UQAC
Université du Québec
à Chicoutimi

Toward robust and trustworthy artificial intelligence: methods for safety and resilience in dynamic and adversarial environments

by Wissam Salhab

Thesis presented to the University of Quebec at Chicoutimi
as part of an extension program of the University of Quebec in Outaouais
in partial fulfillment of the requirements for the degree of
Philosophiæ Doctor (Ph.D.) in Information Science and Technology

Québec, Canada

© Wissam Salhab, 2026

DEDICATION

To my parents, whose unconditional love, sacrifices, and encouragement have been the foundation of all my achievements. To my family, for their continuous support, patience, and belief in me throughout this journey. To my friends and colleagues, who have inspired and motivated me during the challenges and successes of this research. Finally, I dedicate this work to all those who strive to advance knowledge and innovation in the field of artificial intelligence for the benefit of society.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, **Professor Fehmi Jaafar**, for his exceptional mentorship, unwavering support, and invaluable guidance throughout my doctoral studies. His scientific rigor and dedication to research have profoundly shaped the trajectory and quality of this thesis. I am incredibly thankful for the trust he placed in me and for the countless opportunities he provided for my academic and professional growth.

I am equally grateful to my my co-supervisors, **Professor Hamid Mcheick** and **Professor Darine Ameyed**, whose unwavering support, mentorship, and guidance have been fundamental to the successful completion of my PhD. Their expertise, dedication, and confidence in my abilities have shaped not only this research but also my growth as a researcher. Professor Mcheick's patience, thoughtful mentorship, generosity with his time, and constant support played an essential role throughout this journey. His guidance helped me navigate research challenges with confidence and perseverance, contributing significantly to both the quality of this dissertation and my professional development. Professor Ameyed's insightful feedback, strategic vision, constructive recommendations, and continuous encouragement were invaluable at every stage of this work. Her guidance consistently challenged me to refine my ideas, strengthen my research, and strive for academic excellence.

My sincere thanks also go to the members of my doctoral committee and the faculty at the Université du Québec à Chicoutimi (UQAC) for their critical reviews, valuable suggestions, and contributions to my academic journey.

To my colleagues, collaborators, and friends: thank you for the stimulating discussions, collaborative spirit, and camaraderie that made this challenging endeavor both productive and joyful.

Finally, I owe my most profound gratitude to my family. Their unconditional love, boundless patience, and unwavering belief in my potential have been my greatest source of strength and motivation. This achievement would simply not have been possible without their sacrifices and support.

To everyone who contributed, directly or indirectly, to the successful completion of this thesis, I offer my heartfelt thanks.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF TABLES xi

LIST OF FIGURES xiii

RÉSUMÉ xv

ABSTRACT xvii

CHAPTER 1 – INTRODUCTION 1

1.1 MOTIVATION 1

1.2 RESEARCH PROBLEM AND SCOPE 4

1.3 RESEARCH QUESTIONS 5

1.4 METHODOLOGICAL PERSPECTIVE 6

1.5 THESIS HYPOTHESIS 7

1.6 CONTRIBUTIONS 10

1.6.1 CONTRIBUTIONS OF CHAPTER 3 (SUB-HYPOTHESIS 1) 10

1.6.2 CONTRIBUTIONS OF CHAPTER 4 (SUB-HYPOTHESIS 2) 11

1.6.3 CONTRIBUTIONS OF CHAPTER 5 (SUB-HYPOTHESIS 3) 12

1.6.4 CONTRIBUTIONS OF CHAPTER 6 (SUB-HYPOTHESIS 4) 14

1.7 LIST OF PUBLICATIONS 15

1.8 THESIS ORGANIZATION 16

CHAPTER 2 – AI ROBUSTNESS AND SAFETY: LITERATURE REVIEW 17

2.1 CHAPTER OVERVIEW 17

2.1.1 AI ROBUSTNESS AND ITS RELATIONSHIP TO AI SAFETY . . . 18

2.2 ROBUSTNESS AND SAFETY ACROSS LEARNING TECHNIQUES . . . 21
2.2.1 ROBUSTNESS AND SAFETY IN REINFORCEMENT LEARNING . 21
2.2.2 ROBUSTNESS AND SAFETY IN SUPERVISED LEARNING 23
2.2.3 ROBUSTNESS AND SAFETY IN UNSUPERVISED LEARNING . . 25
2.3 ASSURANCE OF ROBUST AND SAFE AI SYSTEMS 27
2.3.1 VERIFICATION AND VALIDATION METHODS 27
2.3.2 FAILURE MODES 29
2.3.3 AUTONOMY, OVERSIGHT, AND SAFETY IN AI SYSTEMS 29
2.4 DIMENSIONS AND SUPPORTING PROPERTIES OF AI ROBUSTNESS 31
2.4.1 SCOPE AND EVALUATION OF AI ROBUSTNESS 31
2.4.2 ADVERSARIAL ROBUSTNESS 33
2.4.3 TRAINING-DATA ROBUSTNESS AGAINST POISONING 35
2.4.4 FAIRNESS AS A CONSTRAINT ON ROBUSTNESS 36
2.4.5 RELIABILITY SUPPORTED BY ROBUSTNESS 37
2.4.6 ROBUSTNESS UNDER DISTRIBUTION SHIFT AND OOD INPUTS 38
2.4.7 EXPLAINABILITY AND TRANSPARENCY FOR ROBUSTNESS ASSESSMENT 39
2.4.8 MONITORING FOR RUNTIME ROBUSTNESS 41
2.5 ROBUSTNESS RESEARCH GAPS AND ALIGNMENT WITH THE THESIS HYPOTHESIS 42
2.6 CHAPTER SUMMARY 42
CHAPTER 3 – TOWARDS ROBUST ARTIFICIAL INTELLIGENCE: SELF-SUPERVISED LEARNING APPROACH FOR OUT-OF-DISTRIBUTION DETECTION 46
3.1 CHAPTER OVERVIEW 46
3.2 RELATED WORK 47
3.2.1 MAHALANOBIS-BASED OOD DETECTION 47
3.2.2 OOD DETECTION USING SUPERVISED CLASSIFIERS 48
3.2.3 OOD DETECTION USING CONTRASTIVE LEARNING 49
3.2.4 OOD DETECTION USING GRAPH 49

3.2.5 LIMITATIONS OF PREVIOUS METHODS 49
3.3 PROPOSED APPROACH 50
3.3.1 PHASE 1: IN-DISTRIBUTION DATA REPRESENTATION PHASE . 51
3.3.2 PHASE 2: OOD INFERENCE 53
3.3.3 THEORETICAL JUSTIFICATION FOR GRAPH CLUSTERING AND MAHALANOBIS DISTANCE 54
3.4 EXPERIMENTAL RESULTS 55
3.4.1 DATASETS 55
3.4.2 EXPERIMENT SETUP 56
3.4.3 RESULTS 56
3.4.4 DISCUSSION 60
3.5 CHALLENGES AND LIMITATIONS 61
3.6 ABLATION STUDY 62
3.6.1 SUPERVISED OOD DETECTION USING KNN WITHOUT GRAPH REPRESENTATION 62
3.6.2 K-MEANS FOR NODE CLUSTERING 63
3.6.3 EFFECT OF NEIGHBOR COUNT ON OOD DETECTION 63
3.6.4 EFFECT OF THRESHOLD ON ACCURACY 64
3.7 CHAPTER SUMMARY 64
CHAPTER 4 – ENHANCING AGENT'S ROBUSTNESS IN REINFORCEMENT LEARNING VIA FOUNDATION MODELS AND DOMAIN RANDOMIZATION 66
4.1 CHAPTER OVERVIEW 66
4.2 RELATED WORK 68
4.2.1 ENHANCING ROBUSTNESS THROUGH FMS 68
4.2.2 ENHANCING ROBUSTNESS IN RL THROUGH DR 69
4.2.3 LIMITATIONS AND GAPS 70
4.3 METHODOLOGY 71
4.3.1 PROPOSED APPROACH 72
4.4 EXPERIMENTAL RESULTS 75

4.4.1 METHODS EVALUATED 75
4.4.2 ENVIRONMENT 76
4.4.3 SETUP 77
4.4.4 RESULTS 78
4.4.5 IMPACT OF MISSION STRING DIVERSITY ON AGENT PERFORMANCE 83
4.4.6 COMPONENT ANALYSIS AND ABLATION 84
4.4.7 DISCUSSION 93
4.5 LIMITATIONS AND FUTURE WORK 94
4.6 CHAPTER SUMMARY 96
CHAPTER 5 – ADVERSARIAL ROBUSTNESS OF REINFORCEMENT LEARNING IN AUTONOMOUS DRIVING: VULNERABILITY ANALYSIS AND INFERENCE-TIME DEFENSES 98
5.1 CHAPTER OVERVIEW 98
5.2 REINFORCEMENT LEARNING PRELIMINARIES 101
5.2.1 STATE SPACE $\mathcal{S}$ 101
5.2.2 ACTION SPACE $\mathcal{A}$ 102
5.2.3 TRANSITION DYNAMICS $\mathcal{P}$ 102
5.2.4 REWARD FUNCTION $\mathcal{R}$ 102
5.2.5 DEEP RL ALGORITHMS (DQN AND PPO) 102
5.3 RELATED WORK 103
5.3.1 ADVERSARIAL ATTACKS ON REINFORCEMENT LEARNING . . 103
5.3.2 THE "ROBUSTNESS TAX" OF ADVERSARIAL TRAINING 103
5.3.3 ROBUST REINFORCEMENT LEARNING AND EVALUATION PRACTICE 104
5.3.4 INFERENCE-TIME INTERVENTIONS AND SAFE RL 105
5.4 THREAT MODEL FORMALIZATION 105
5.4.1 OBSERVATION-SPACE VULNERABILITIES 106
5.4.2 ACTION-SPACE VULNERABILITIES (ACTUATION TAMPERING) 106
5.4.3 ATTACK LOCATION AND TRUSTED SIGNALS 107

5.5 INFERENCE-TIME DEFENSE MECHANISMS 107
5.5.1 THE "LATENCY TRAP" OF STATE SMOOTHING (EMA) 107
5.5.2 Q-VALUE GAP THRESHOLDING (QGAP) 108
5.5.3 DYNIW: DYNAMIC OBSERVATION GATING WITH POLICY-REFERENCED ACTION VALIDATION 109
5.6 EXPERIMENTAL SETUP AND METHODOLOGY 111
5.6.1 SIMULATION ENVIRONMENTS 112
5.6.2 MODEL ARCHITECTURES AND BASELINE TRAINING 113
5.6.3 ATTACK AND DEFENSE EVALUATION PIPELINE 113
5.6.4 EVALUATION METRICS 114
5.6.5 RL ARCHITECTURE AND TRAINING REGIME 116
5.7 EXPERIMENTAL RESULTS 117
5.7.1 FINDING 1: SUSTAINED ACTION-SPACE ATTACKS ARE THE MOST DAMAGING 118
5.7.2 FINDING 2: DEFENSE EFFECTIVENESS IS ARCHITECTURE-DEPENDENT 120
5.7.3 FINDING 3: QGAP WAS EFFECTIVE WHERE AN ACTION-VALUE LANDSCAPE WAS AVAILABLE 122
5.7.4 FINDING 4: DYNIW'S CONTINUOUS-ACTION BRANCH REDUCES COLLISIONS FOR A CONTINUOUS-CONTROL PPO CHECKPOINT UNDER FIXED-COMMAND INJECTION 126
5.7.5 FINDING 5: COMPOSITE ATTACKS REMAIN UNSOLVED 129
5.8 DISCUSSION 129
5.8.1 IMPLICATIONS FOR DEPLOYMENT 131
5.8.2 LATENCY AND REAL-TIME CONSTRAINTS 131
5.8.3 PRACTICAL RECOMMENDATIONS AND FAILURE MITIGATION 131
5.8.4 INTERPRETING AGGREGATE PLOTS AND FAILURE MODES . . 132
5.8.5 HYPOTHESIS AND KEY FINDINGS 132
5.9 THREATS TO VALIDITY 134
5.9.1 INTERNAL VALIDITY: THRESHOLD TUNING BIAS 134
5.9.2 INTERNAL VALIDITY: SINGLE-SEED LIMITATIONS 134

5.9.3 EXTERNAL VALIDITY: SINGLE-POLICY-FAMILY EVALUATION 135
5.9.4 EXTERNAL VALIDITY: HIGHWAYENV DEPENDENCE 135
5.9.5 EXTERNAL VALIDITY: THE SIM-TO-REAL GAP 135
5.9.6 CONSTRUCT VALIDITY: ATTACK-MODEL SIMPLIFICATIONS . 136
5.9.7 CONSTRUCT VALIDITY: PROXY FIDELITY AND REACTIVE GATING 137
5.9.8 CONSTRUCT VALIDITY: SCOPE OF THE POLICY-REFERENCED CHECK 137
5.10 APPENDIX: HIGHWAYENV TRAINING CURVES 138
5.11 APPENDIX: ADDITIONAL RESULTS 138
5.11.1 OBSERVATION AND COMPOSITE FAILURE CASES 138
5.12 APPENDIX: EXTERNAL METADRIVE PPO MODEL, TRAINING CONFIGURATION AND INTERFACE DETAILS 139
5.12.1 EXTERNAL POLICY AND EVALUATION PROTOCOL 139
5.12.2 CONTINUOUS-ACTION ATTACK INTERFACE AND THE DYNIW CONTINUOUS BRANCH 139
5.12.3 TRAINING CONFIGURATION 140
5.13 CHAPTER SUMMARY 142
CHAPTER 6 – ROBUSTNESS OF JAILBREAK DETECTION UNDER ADVERSARIAL DRIFT: A CO-EVOLUTIONARY DEFENSE 145
6.1 CHAPTER OVERVIEW 145
6.2 RELATED WORK 147
6.2.1 REPRESENTATION-BASED DEFENSES 147
6.2.2 PROMPT-LEVEL CONSISTENCY DEFENSES 148
6.2.3 PERTURBATION AND AGGREGATION DEFENSES 148
6.3 BACKGROUND 149
6.3.1 AI-SAFETY MOTIVATION AND TERMINOLOGY 149
6.3.2 UNIVERSAL POLICY PUPPET PAYLOADS 150
6.4 CO-EVO DEFENDER 151
6.4.1 THE ATTACKER MODEL 151
6.4.2 THE CO-EVO DEFENDER MODEL 153

6.4.3 THE CO-EVOLUTIONARY TRAINING LOOP 154
6.5 EMPIRICAL STUDY AND ROBUSTNESS EVALUATION 155
6.5.1 SETUP 155
6.5.2 LEARNING DYNAMICS AND PRELIMINARY RESULTS 155
6.5.3 ROBUSTNESS TO STATIC OBFUSCATION ATTACKS 157
6.5.4 ROBUSTNESS TO UNSEEN ADAPTIVE ATTACKS 158
6.5.5 ROBUSTNESS ACROSS STATIC AND ADAPTIVE EVALUATIONS 159
6.5.6 DISCUSSION 159
6.6 LIMITATIONS AND FUTURE WORK 161
6.7 APPENDIX: ADDITIONAL FORMULATIONS 162
6.7.1 DEFENDER AND ATTACKER FORMULATIONS 162
6.7.2 METRICS 162
6.7.3 LOSS CALIBRATION 163
6.8 CHAPTER SUMMARY 167
CHAPTER 7 – CONCLUSION 169
7.1 SUB-HYPOTHESIS 1: SELF-SUPERVISED OOD DETECTION (CHAPTER 3) 169
7.2 SUB-HYPOTHESIS 2: SAFE REINFORCEMENT LEARNING UNDER DISTRIBUTIONAL SHIFT (CHAPTER 4) 170
7.3 SUB-HYPOTHESIS 3: ADVERSARIAL ROBUSTNESS FOR AUTONOMOUS DRIVING (CHAPTER 5) 170
7.4 SUB-HYPOTHESIS 4: CO-EVOLUTIONARY DEFENSE FOR JAILBREAK DETECTION (CHAPTER 6) 171
7.5 SYNTHESIS OF RESULTS AND CONFIRMATION OF THE MAIN HYPOTHESIS 172
7.6 FUTURE WORK 174
BIBLIOGRAPHY 176

LIST OF TABLES

TABLE 2.1 : Taxonomy of ML Safety techniques 24
TABLE 2.2 : Dimensions and supporting properties of AI robustness. 32
TABLE 2.3 : AI robustness gaps identified in the literature and their alignment with the thesis hypothesis.. 43
TABLE 3.1 : Comparison of OOD detection methods using CIFAR10 as in-distribution vs SVHN as OOD 50
TABLE 3.2 : Using different k values for KNN with CIFAR-100 dataset; the optimal value of k is 7, resulting in 109 clusters, the closest to the ground truth in CIFAR-100 clusters.. 59
TABLE 3.3 : Using different k values for KNN with CIFAR-10 dataset; the optimal value of $k = 7$ results in 13 clusters, closest to the ground truth of CIFAR-10 clusters. 59
TABLE 3.4 : Performance of our approach compared to existing methods: CIFAR-10 (in-distribution), CIFAR-100 and SVHN (OOD). 60
TABLE 3.5 : Performance of our approach compared to existing methods: CIFAR-100 (in-distribution), CIFAR-10 and SVHN (OOD). 60
TABLE 3.6 : AUROC Scores for Different Values of k 63
TABLE 3.7 : Performance of our approach compared to existing methods: CIFAR-100 (in-distribution), CIFAR-10 and SVHN (OOD), using K-means where $k = 109$ (the same k as the number of clusters obtained via the Louvain method). 63
TABLE 3.8 : Overall classification accuracy as a function of percentile threshold τ applied to Mahalanobis distances. 64
TABLE 4.1 : Comparison of methods 71
TABLE 4.2 : Mean return for different models on DistShift1 vs DistShift2 80
TABLE 4.3 : Mean return for different models with DR on DistShift1 vs DistShift2. . 83
TABLE 4.4 : Results using Gemini as the guide 89
TABLE 4.5 : Results of using DeepSeek as the agent guide 91
TABLE 5.1 : Hyperparameter Configurations for the Evaluated HighwayEnv Models . 116
TABLE 5.2 : Selected 100-episode results for HighwayEnv (DQN) 118
TABLE 5.3 : Selected 100-episode results for HighwayEnv (PPO) 119
TABLE 5.4 : Selected 100-episode results for MetaDrive (continuous PPO) 127
TABLE 5.5 : Defense-only latency per control step 130
TABLE 5.6 : Architecture-Defense Coupling: Key Findings Summary 130
TABLE 5.7 : Selected observation-filter and composite-attack failure cases. 138

TABLE 5.8 : Training Configuration for the External MetaDrive PPO Policy [1]. . . . 141
TABLE 5.9 : Training Regime Comparison Across Frameworks 142
TABLE 6.1 : Robustness to character-level obfuscation attacks. Recall is reported as the true positive rate (TPR), and specificity as the true negative rate (TNR). 157
TABLE 6.2 : Performance on unseen attacks from the JailbreakLLMs dataset 159
TABLE 6.3 : Ablation study results summary 167

LIST OF FIGURES

FIGURE 1.1 – Relationship between the main hypothesis, the four sub-hypotheses, and their corresponding thesis contributions.. 9
FIGURE 2.1 – Trustworthiness attributes of AI systems. 20
FIGURE 2.2 – The Box-Jenkins framework for safe AI 28
FIGURE 2.3 – Types of explainable artificial intelligence (XAI): transparent versus black-box models 39
FIGURE 3.1 – Our proposed approach for OOD detection. 51
FIGURE 3.2 – Top-5 validation accuracy chart. 57
FIGURE 3.3 – Clustered graph: each node in the graph represents an image embedding in latent space. 58
FIGURE 4.1 – Our proposed approach, illustrating the training flow. The DR module perturbs the environment, the FM generates a guiding Mission String from the state, and the RL Agent learns by combining both inputs. . . . 72
FIGURE 4.2 – MiniGrid DistShift1 environment: a grid-based maze where a player, represented by a red triangle, must navigate to a goal, depicted as a green square, while avoiding obstacles. 76
FIGURE 4.3 – MiniGrid DistShift2 environment 77
FIGURE 4.4 – Performance of different models in terms of mean return: (a) FM-PPO; FM standalone as a mission string generator, (b) PPO-Base with default mission string text.. 78
FIGURE 4.5 – Policy loss showing stable learning: (a) FM-PPO, (b) PPO-Base 79
FIGURE 4.6 – Performance of different models in terms of mean return with DR: (a) FMDM (our); (b) PPO-DR; baseline with DR and default mission string text. 80
FIGURE 4.7 – Policy loss of different models: (a) FMDM, (b) PPO-DR. All policy losses stabilized around 0 ± 0.02 after 200k steps 81
FIGURE 4.8 – Effect of the generated mission string on RL performance 85
FIGURE 4.9 – Performance of the proposed approach using different seeds. 87
FIGURE 4.10 – (a) GPT-4.0 Turbo 88
FIGURE 4.11 – (b) DeepSeek r1. 88
FIGURE 4.12 – (c) Gemini 2.0 Flash 88
FIGURE 4.13 – Action sequences generated by different FMs 88
FIGURE 4.14 – (a): Mean return without DR 90
FIGURE 4.15 – (b): Policy loss without DR 90
FIGURE 4.16 – (c): Mean return with DR 90

FIGURE 4.17 – (d): Policy loss with DR 90
FIGURE 4.18 – Gemini as the FM. 90
FIGURE 4.19 – (a): Mean return without DR 92
FIGURE 4.20 – (b): Policy loss without DR 92
FIGURE 4.21 – (c): Mean return with DR 92
FIGURE 4.22 – (d): Policy loss with DR 92
FIGURE 4.23 – DeepSeek as the guide 92
FIGURE 4.24 – Attention Heatmap showing the RL agent's focus on the goal at the bottom right corner 93
FIGURE 5.1 – Overview of the defense evaluation pipeline 112
FIGURE 5.2 – HighwayEnv training convergence of the evaluated agents 117
FIGURE 5.3 – Mean episodic reward across the 12 evaluated scenarios 120
FIGURE 5.4 – Collision rate across the same scenario grid 120
FIGURE 5.5 – Reward-retention ratio across the scenario grid 121
FIGURE 6.1 – Simplified policy-puppet payload 149
FIGURE 6.2 – t-SNE projection of prompt embeddings 150
FIGURE 6.3 – Arms-race loop between attacker and defender. 152
FIGURE 6.4 – Defender accuracy over training epochs 156
FIGURE 6.5 – Defender loss convergence 156
FIGURE 6.6 – Attacker ASR volatility indicating successful defense adaptation 157
FIGURE 6.7 – Results from running different random seeds: 42, 123, and 1000.. . . . 164
FIGURE 6.8 – Effect of loss weight parameters on defender accuracy. 165
FIGURE 6.9 – Effect of loss weight parameters on defender loss.. 165
FIGURE 6.10 – Effect of loss weight parameters on attacker ASR.. 166

RÉSUMÉ

L'intelligence artificielle (IA) a transformé de nombreux secteurs, notamment la santé, les véhicules autonomes, la finance, la cybersécurité et l'automatisation industrielle, en proposant des solutions qui améliorent l'efficacité, la précision et la prise de décision. Toutefois, le déploiement de l'IA dans des environnements critiques soulève d'importantes préoccupations en raison de la nature imprévisible de ces contextes. Les modèles d'IA demeurent vulnérables aux attaques, aux erreurs de décision et aux situations inédites, ce qui peut engendrer des risques significatifs pour la sécurité. Cette recherche met en évidence l'importance cruciale de la robustesse de l'IA et aborde les défis liés aux données, aux modèles et à la sécurité. Elle souligne la nécessité de développer des stratégies efficaces permettant de s'adapter aux changements, de garantir la résilience des modèles face à des conditions variables et de protéger les systèmes d'IA contre les menaces de sécurité. Cette thèse traite ainsi des principaux enjeux relatifs à la robustesse et à la sécurité des modèles d'intelligence artificielle. Les résultats obtenus démontrent des avancées significatives en matière de capacités de détection, de robustesse des modèles et de résistance aux menaces potentielles, contribuant ainsi au développement de technologies d'IA plus sûres et plus fiables dans des environnements dynamiques et exigeants. La première contribution propose un cadre autosupervisé basé sur les graphes pour la détection hors distribution (OOD), fonctionnant sans nécessiter d'échantillons OOD étiquetés. L'approche utilise SimCLR pour l'extraction de représentations par apprentissage contrastif, la détection de communautés par l'algorithme de Louvain pour identifier les clusters en distribution au sein d'un graphe de similarité cosinus, et le score de distance de Mahalanobis pour signaler les entrées anormales lors de l'inférence. Les expériences menées sur des jeux de données de référence démontrent des performances de détection compétitives par rapport aux approches supervisées, confirmant la valeur pratique de ce cadre pour préserver l'intégrité des données d'entraînement dans les applications critiques.

La deuxième contribution aborde le décalage de distribution dans l'apprentissage par renforcement (AR) en combinant l'orientation par les modèles de fondation avec la randomisation du domaine. Les modèles de fondation génèrent des objectifs de mission spécifiques à la tâche qui enrichissent le signal de récompense, tandis que la randomisation du domaine diversifie systématiquement les paramètres de l'environnement d'entraînement tels que les textures visuelles, la dynamique de l'agent et le bruit des capteurs. Les études d'ablation confirment que les deux composantes contribuent indépendamment à la robustesse, leur combinaison produisant les gains les plus importants sur des environnements non vus lors de l'entraînement.

La troisième contribution examine les menaces adversariales pesant sur les agents de conduite autonome basés sur l'AR. Parmi les attaques sur l'espace des observations, sur l'espace des actions et les attaques composites multivecteurs, l'injection soutenue d'une commande fixe dans le canal d'actionnement a produit la dégradation de sécurité la plus sévère, car la manipulation de l'actionnement contourne directement la sortie de la politique, tandis que l'altération intermittente et aléatoire des actions s'est avérée moins dommageable que le bruit d'observation. Deux défenses à l'inférence, sans réentraînement, sont évaluées : le seuillage par écart de valeurs-Q (QGap), qui filtre les commandes à partir du paysage d'action-valeur des agents basés sur les valeurs, et DynIW, qui combine un filtrage dynamique des observations et une validation des actions par rapport à l'action nominale de la politique protégée. Dans des expériences à graine unique, QGap a réduit le taux de collision de l'agent DQN sous altération intermittente des actions de 0,650 à 0,290, tandis que DynIW a rétabli l'agent PPO sous injection soutenue d'une commande fixe et sous altération intermittente (taux de collision sous injection de commande fixe de 1,000 à 0,000) et, par sa branche à actions continues, a réduit les collisions d'un point de contrôle PPO externe figé dans MetaDrive, sous injection de commande fixe, de 1,000 à 0,080, sans réentraînement. Ces résultats sont compatibles avec un couplage entre l'efficacité de la défense et l'architecture de la politique, mais aucune défense n'a résolu les attaques composites multivecteurs ni rétabli l'agent DQN sous injection soutenue d'une commande fixe, et ces constats doivent être validés sur plusieurs graines, familles de politiques et environnements.

La quatrième contribution propose Co-Evo Defender, un cadre co-évolutif qui renforce la détection des tentatives de contournement (*jailbreak*) dans les grands modèles de langage contre la dérive adversariale. Un attaquant basé sur l'AR génère en continu des perturbations sémantiques pour sonder les faiblesses du défenseur, tandis qu'un classificateur neuronal adaptatif est réentraîné à chaque contournement découvert. Cette boucle de rétroaction maintient un rappel élevé sur des motifs d'attaque hors distribution qui mettent en échec les classificateurs statiques.

Pris dans leur ensemble, ces quatre contributions font progresser l'état de l'art en matière de robustesse de l'IA selon les dimensions des données, des modèles et de la sécurité, posant ainsi les bases d'un déploiement plus sûr de l'IA dans des environnements réels dynamiques et exigeants.

ABSTRACT

Artificial intelligence (AI) has transformed critical sectors such as healthcare, autonomous vehicles, finance, cybersecurity, and industrial automation by enabling solutions that enhance efficiency, accuracy, and decision-making. However, deploying AI in high-stakes environments raises serious concerns due to the unpredictable nature of real-world settings. AI models remain vulnerable to adversarial attacks, decision failures, and distributional shifts. These unexpected discrepancies between training and deployment conditions can lead to catastrophic consequences and safety risks. This thesis addresses the fundamental challenge of building robust and secure AI systems by investigating three core dimensions: the integrity of training data, the robustness of AI models under distributional shift, and the resilience of AI systems against adversarial and security threats.

The first contribution proposes a self-supervised, graph-based framework for out-of-distribution (OOD) detection that operates without requiring labeled OOD samples. The approach employs SimCLR for contrastive embedding extraction, Louvain community detection to partition in-distribution clusters within a cosine-similarity graph, and Mahalanobis distance scoring to flag anomalous inputs at inference time. Experiments on benchmark datasets demonstrate competitive detection performance against supervised baselines, confirming the framework's practical value for preserving training data integrity in safety-critical applications where labeled OOD data is unavailable.

The second contribution addresses distributional shift in reinforcement learning (RL) by combining foundation model guidance with domain randomization. Foundation models generate task-specific mission objectives that enrich the reward signal beyond what sparse reward alone provides, while domain randomization systematically diversifies training environment parameters such as visual textures, agent dynamics, and sensor noise. Together, these techniques enable RL agents to learn policies that generalize reliably to unseen deployment conditions. Ablation studies confirm that both components contribute independently to robustness, with their combination producing the greatest gains on held-out environments.

The third contribution investigates adversarial threats to RL-based autonomous driving agents. Across observation-space, action-space, and composite multi-vector attacks, sustained fixed-command injection into the action channel produced the most severe safety degradation, as actuation tampering bypasses the policy output, whereas intermittent epsilon-malicious action tampering was less damaging than observation noise. Two training-free inference-time defenses are evaluated: Q-Value Gap Thresholding (QGap), which gates commands using the action-value landscape of value-based agents, and DynIW, which combines dynamic observation gating with action validation against the protected policy's own nominal action. In single-seed experiments, QGap reduced the deep Q-network (DQN) agent's colli-

sion rate under intermittent action tampering from 0.650 to 0.290, whereas DynIW restored the proximal policy optimization (PPO) agent under sustained fixed-command injection and intermittent action tampering (collision rate under fixed-command injection from 1.000 to 0.000) and, through its continuous-action branch, reduced collisions for a frozen, externally pre-trained continuous-control PPO checkpoint in MetaDrive under fixed-command injection from 1.000 to 0.080, all without model retraining. These results are consistent with a coupling between defense effectiveness and policy architecture, but no wrapper resolved composite multi-vector attacks or recovered the DQN agent under sustained fixed-command injection, and the findings require validation across training seeds, policy families, and environments.

The fourth contribution proposes Co-Evo Defender, a co-evolutionary framework that hardens jailbreak detection in large language models (LLMs) against adversarial drift. An RL-based attacker powered by the PPO algorithm continuously generates semantic perturbations at the embedding level to probe the defender's weaknesses, while an adaptive neural network classifier is retrained on each discovered bypass. This feedback loop ensures that the defender's knowledge of adversarial strategies grows over time, maintaining robust recall on out-of-distribution attack patterns, including leetspeak obfuscation and policy-puppet payloads, that cause static classifiers to fail.

Overall, these four contributions significantly advance the frontier of AI robustness by addressing challenges spanning data quality, model reliability, and system security. The proposed approaches enhance OOD detection performance, improve agent generalization under distributional shifts and strengthen resilience to adversarial attacks. Together, they pave the way for more secure, dependable, and resilient AI systems operating in dynamic and high-stakes real-world settings.

CHAPTER 1
INTRODUCTION

1.1 MOTIVATION

Artificial intelligence (AI) is increasingly embedded in systems that transform complex observations into predictions, recommendations, or actions. In healthcare, learning-based tools support diagnostic and predictive tasks that can complement clinical decision-making [2]. In autonomous driving, learned components interpret sensor data and contribute to decisions that must be made continuously as traffic and environmental conditions change [3]. In such settings, the value of an AI model depends not only on its accuracy under nominal evaluation conditions, but also on whether it can maintain acceptable behavior when the conditions encountered after deployment differ from those represented during development.

Model development necessarily relies on finite training, validation, and test sets. These datasets provide empirical evidence about performance, but they also create a limited approximation of the environment in which the model will eventually operate. Real environments are open and non-stationary: populations evolve, sensors and interfaces change, previously unseen categories appear, and operational conditions vary over time. Such changes may be gradual or abrupt, common or rare. They may also affect only a small but safety-relevant region of the input space, in which case an aggregate test metric can remain high while important failure modes are concealed. Consequently, strong performance on a fixed benchmark does not, by itself, establish dependable behavior in deployment [4, 5].

Departures from development conditions may arise naturally or may be produced deliberately. Non-adversarial distribution shift can result from changes in data acquisition, context, dynamics, or the population being observed. Adversarial change is intentional: an attacker selects inputs or actions to exploit a weakness in a model, policy, or monitoring mechanism. Both forms of change can expose reliance on unstable features and assumptions, but they require different forms of assurance. Generalization and novelty detection are

central when variation is unintentional, whereas explicit threat models, runtime controls, and adaptive defenses become necessary when the environment contains a strategic adversary [3]. Treating these conditions as equivalent can hide important assumptions about what a defense observes and which failures it can reasonably prevent.

The problem is especially acute in sequential decision-making. A prediction error in a static dataset affects one output, whereas an erroneous action taken by an autonomous agent changes the state from which subsequent observations and decisions are produced. Small deviations can therefore accumulate through the interaction between the agent and its environment. A reinforcement learning (RL) policy may also exploit regularities that are specific to its training simulator or reward formulation, without learning behavior that transfers to modified conditions. Similarly, a security classifier that is effective against known attacks may become less effective as attack strategies evolve. Robustness must therefore be examined over interactions and changing conditions, rather than inferred solely from isolated predictions on clean data.

Robustness is closely related to safety and trustworthiness, but the concepts are not interchangeable. In this thesis, robustness denotes the operational capacity of a model or agent to preserve acceptable behavior under specified variations, unfamiliar inputs, or adversarial manipulations. Safety additionally depends on the hazards of the application, the acceptability of residual risk, system-level constraints, and the consequences of failure. A model can be stable yet consistently make an inappropriate decision, while a conservative monitor can reduce a safety risk at the cost of task performance. Distinguishing these concepts avoids interpreting an improvement in one robustness metric as a universal safety guarantee.

From a system perspective, failures can enter at several points in the AI lifecycle. Data may be incomplete, contaminated, or unrepresentative of future observations. A learned model may overfit regularities of the training environment and degrade when those regularities change. After deployment, interfaces to observations and actions may be manipulated, and a fixed defense may lose effectiveness against new attack patterns. These risks are con-

nected: weaknesses in the data shape the learned representation, the representation constrains generalization, and the deployed architecture determines which signals are available for monitoring. Assurance must therefore extend from data management through model development to deployment and monitoring [6, 5].

No single mechanism is sufficient across this lifecycle. Detecting unfamiliar samples can prevent unsupported inputs from being treated as routine, but it does not ensure that an agent will behave reliably throughout a shifted environment. Diversifying training conditions can reduce dependence on a narrow environment, but it cannot anticipate every deliberate manipulation. Runtime monitoring can intercept some abnormal inputs or actions, yet its effectiveness depends on the policy architecture and on the trusted information available at inference time. When adversarial behavior evolves, maintaining protection may additionally require a feedback process in which newly discovered failures inform subsequent defensive updates.

A further practical constraint is that examples of future failures are rarely available in advance. Comprehensive labels for out-of-distribution (OOD) data are difficult to obtain because the space of possible novelty is open-ended. Likewise, a defender cannot enumerate every perturbation or jailbreak strategy that an attacker may later construct. Methods that require a complete catalogue of anomalies, or extensive retraining for each individual attack, are therefore difficult to sustain. This motivates approaches that exploit the structure of nominal data, expose models to controlled environmental diversity, use architecture-appropriate signals during deployment, and learn systematically from successful bypasses.

These observations support a layered view of robust AI. The integrity of data entering the learning pipeline, the capacity of a model to generalize beyond a narrow training distribution, and the resilience of a deployed system to intentional manipulation are distinct but interdependent concerns. Studying them together makes it possible to identify which assumptions are shared, which mechanisms are complementary, and where a defense effective at one layer leaves another layer exposed.

1.2 RESEARCH PROBLEM AND SCOPE

The central research problem is the gap between the finite evidence available during model development and the range of conditions that can arise during operation. For a predictive task, this gap is commonly expressed as a difference between the training distribution and the deployment distribution:

$$P_{\text{train}}(X, Y) \neq P_{\text{deploy}}(X, Y). \tag{1.1}$$

Equation 1.1 does not imply that every distributional difference is harmful. The relevant question is whether the difference alters the evidence on which a prediction or action depends and produces a meaningful degradation in performance or safety. In sequential settings, the mismatch can also affect transition dynamics, observations, reward conditions, or the behavior of other agents. Under an adversarial threat model, the operational inputs are not merely sampled from a different distribution; they may be selected in response to the model or defense with the specific objective of inducing failure.

At the data boundary, the problem is to determine whether an observation is sufficiently supported by what was learned from nominal data. A conventional classifier is generally required to return a prediction even when the input lies outside the regions represented during training. Without a mechanism for recognizing this condition, confidence in the returned class can be mistaken for evidence that the input itself is familiar. OOD detection introduces an additional decision: whether an input should be processed as nominal or treated as unsupported. The difficulty addressed here is to make this distinction without assuming that representative examples of all possible OOD cases are labeled and available beforehand.

At the model level, the problem is to learn behavior that depends on task-relevant structure rather than incidental properties of a single training environment. This challenge is pronounced in RL because the agent learns from the states it visits and the feedback produced by its own actions. A policy can perform well within one environment while failing when visual appearance, dynamics, object placement, or task formulation changes. Robust training

must provide enough variation and guidance to support transfer, while preserving a learning signal that remains connected to the intended task.

At the security boundary, the problem changes from generalization under variation to resilience against an opponent. An attack may alter observations before they reach a policy, tamper with actions after they are selected, or search for representations that evade a safety classifier. The location of the attack determines which defensive information remains trustworthy. It also determines whether a defense should modify the input, reject an action, invoke a fallback, or update the decision boundary. A useful defense must therefore be matched to the architecture and threat model rather than applied as a generic preprocessing step.

The scope of this thesis is computational robustness under distributional and adversarial change. It covers OOD detection for data integrity, generalization of RL agents across modified environments, inference-time protection of autonomous driving policies, and adaptive detection of evolving jailbreak attacks. These studies address selected technical components of trustworthy AI; they do not attempt to resolve the full set of concerns associated with fairness, privacy, explainability, governance, or formal system certification. The empirical results are consequently interpreted within the datasets, simulated environments, model families, and threat models used in each study. They provide evidence of robustness under specified conditions, not a claim of universal safety.

1.3 RESEARCH QUESTIONS

The lifecycle perspective above leads to three research questions. They progress from identifying unsupported data, to maintaining model behavior under environmental change, and finally to protecting deployed systems against deliberate manipulation.

RQ1: How can OOD samples be detected without labeled OOD examples, so that unfamiliar data can be distinguished from the nominal data represented during learning?

RQ2: How can RL agents be trained to maintain reliable behavior when deployment conditions differ from the environment in which the policy was learned?

RQ3: How can deployed AI systems detect, mitigate, and adapt to adversarial manipulations, including attacks that evolve after the initial model or defense has been trained?

The first question concerns evidence at the data boundary and is examined through self-supervised representation learning and graph-based OOD detection. The second concerns model generalization and is studied by combining structured task guidance with environmental diversification during RL training. The third concerns security after deployment and is investigated in two settings with different control structures: adversarial interference with autonomous driving agents and evolving jailbreak strategies against language-model safety classifiers. This separation permits each study to use a threat model and evaluation protocol appropriate to the information available at that layer.

1.4 METHODOLOGICAL PERSPECTIVE

The investigation follows an empirical, mechanism-oriented methodology. It begins with a systematic review that organizes the principal safety concerns, learning paradigms, failure modes, and assurance practices reported in the literature. The technical studies then isolate specific failure conditions identified by this broader analysis. For each condition, the defensive mechanism is designed around an explicit constraint, such as the absence of labeled OOD data, a change between training and deployment environments, a frozen policy that cannot be retrained, or an adversary whose strategies continue to evolve.

Although the four technical studies operate in different domains, their evaluation follows a common logic. Nominal behavior is established first, after which the data distribution, environment, observation, action, or attack strategy is modified in a controlled manner. The proposed method is compared with relevant baselines, and component analyses or ablation studies are used where appropriate to determine which part of the method accounts for the observed behavior. Failure cases and deployment constraints are retained in the analysis

because an aggregate improvement alone does not show where a protection mechanism ceases to be reliable.

Robustness is therefore treated as a property that must be demonstrated relative to stated conditions. Detection quality, task performance, recovery under attack, and preservation of behavior on nominal inputs are considered according to the needs of each study. Since these measures describe different systems and consequences, they are not reduced to a single score across the thesis. The synthesis instead concerns the role played by each mechanism within the lifecycle: recognizing unsupported data, reducing sensitivity to environmental variation, monitoring decisions at inference time, and adapting a defense when new bypasses are discovered.

1.5 THESIS HYPOTHESIS

Achieving robustness in AI models is a multifaceted challenge requiring careful consideration of multiple dimensions: data integrity, robustness of AI models in dynamic environments, and resilience against adversarial threats. Starting from this observation, we formulate the following overarching hypothesis:

- **Hypothesis: ensuring AI robustness involves maintaining the integrity of training data, ensuring the reliability of AI models, and enhancing the security of AI models, which minimizes risks and vulnerabilities in dynamic environments.**

This general hypothesis is supported by Sub-Hypotheses 1–4, each corresponding to a dedicated research contribution:

Sub-Hypothesis 1: *A self-supervised, graph-based framework using contrastive learning and Mahalanobis distance (MD) can detect out-of-distribution samples without labeled data, improving training data integrity.*

OOD detection is fundamental to preserving training data integrity, as models exposed to unseen inputs at inference time risk producing unreliable predictions. Label-free methods are particularly valuable in domains where obtaining OOD annotations is expensive or infeasible.

We proposed a self-supervised, graph-based OOD detection framework that combines SimCLR embedding extraction, Louvain community detection, and MD scoring to identify OOD samples without any labeled data (Chapter 3).

Sub-Hypothesis 2: *Integrating foundation models (FMs) with domain randomization (DR) in RL training enables agents to generalize across distributional shifts in dynamic, real-world environments.*

RL agents are typically trained in controlled environments and must subsequently generalize to deployment settings with shifted data distributions. Without explicit mechanisms to handle this shift, agent performance degrades significantly under real-world conditions.

We implemented a training framework that integrates FMs as task-guidance sources with DR as environment diversification, enabling RL agents to maintain robust behavior under distributional shifts (Chapter 4).

Sub-Hypothesis 3: *Architecture-aware inference-time monitoring can detect and mitigate adversarial perturbations in RL-based autonomous driving agents without requiring model retraining.*

Adversarial attacks on autonomous driving agents pose critical safety risks by corrupting perception and decision-making at inference time. Designing defenses that do not require labeled adversarial examples or model retraining is essential for practical real-world deployment.

We evaluated observation-space, action-space, and composite attacks on RL-based driving agents together with two retraining-free inference-time defenses, QGap and DynIW, and examined how their effectiveness relates to the architecture of the protected policy (Chapter 5).

Sub-Hypothesis 4: *A co-evolutionary arms race between an RL-based attacker and an adaptive neural network defender can continuously harden jailbreak detection against adversarial drift.*

Static jailbreak classifiers are brittle to adversarial drift, as attackers can rapidly discover evasion strategies against fixed defenders. Continuous co-evolution between an adaptive

attacker and a learning defender offers a principled solution to this arms-race problem. We proposed Co-Evo Defender, a framework pairing an RL-based Proximal Policy Optimization (PPO) attacker with an adaptive neural network classifier, where each successful bypass is folded back into the defender's training loop to maintain robust jailbreak detection over time (Chapter 6).

Figure 1.1 summarizes the relationship between the main hypothesis, Sub-Hypotheses 1–4, their corresponding chapters, and the resulting contributions.

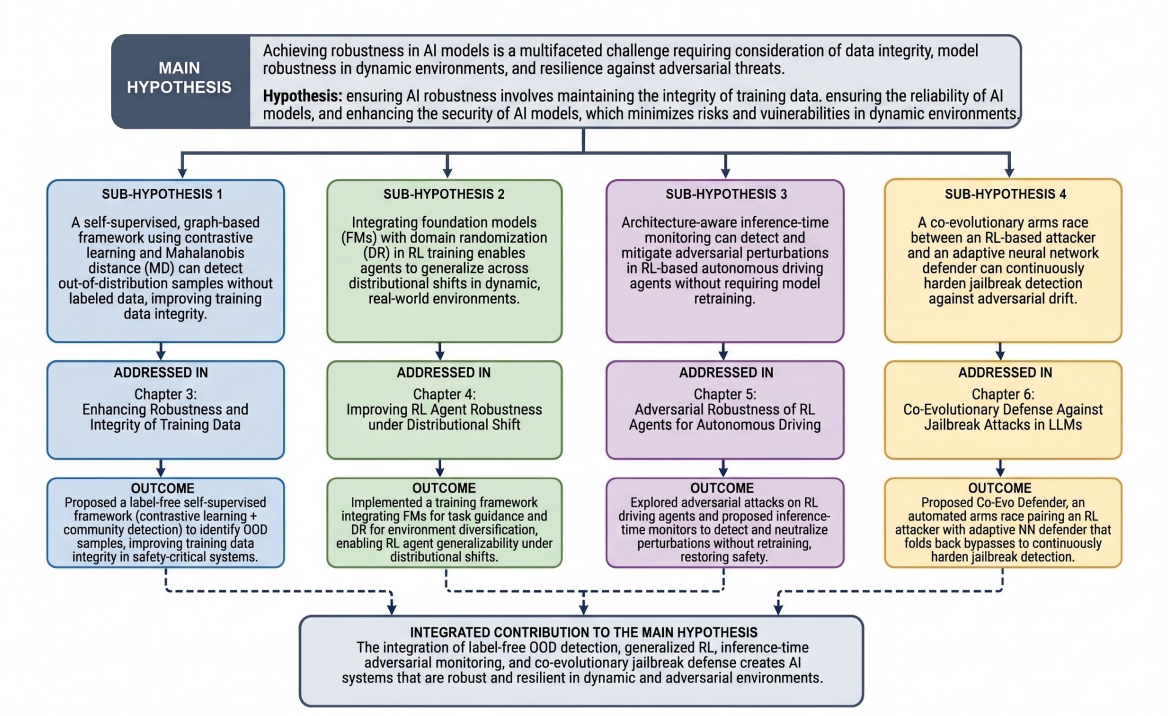


Figure 1.1 : Relationship between the main hypothesis, the four sub-hypotheses, and their corresponding thesis contributions.

To validate the main hypothesis, we develop four dedicated contributions, collectively targeting Sub-Hypotheses 1–4. The corresponding research questions and detailed contribution descriptions are presented in Section 1.6.

1.6 CONTRIBUTIONS

This thesis makes four principal contributions to the field of AI robustness and safety, each developed in a dedicated chapter.

1.6.1 CONTRIBUTIONS OF CHAPTER 3 (SUB-HYPOTHESIS 1)

- **First major contribution:** We proposed a self-supervised, graph-based framework that leverages contrastive learning and community detection to identify out-of-distribution samples without requiring any labeled OOD data, improving the integrity of training data in safety-critical AI systems.

C1: Enhancing Robustness and Integrity of Training Data: In this contribution, we addressed the challenge of ensuring that AI models are trained on reliable, in-distribution data. How can OOD samples be identified without access to labeled OOD examples? Can self-supervised representation learning capture the geometric structure of in-distribution data effectively? How can graph-based community detection improve the scalability and accuracy of OOD detection? How can we develop an approach to ensure the robustness and integrity of training data, thereby enhancing AI robustness in intelligent systems?

The first contribution proposes a novel self-supervised, graph-based framework for OOD detection. Ensuring training data integrity is a foundational requirement for robust AI systems; however, most existing OOD detection methods rely on labeled out-of-distribution samples, which are often unavailable or prohibitively costly to collect in safety-critical domains. Our approach overcomes this limitation by combining self-supervised representation learning with graph-based community detection.

Specifically, the framework employs SimCLR to extract semantically rich embeddings from unlabeled training data, then constructs a cosine-similarity graph in which each data point is represented as a node. Louvain community detection is applied to partition this graph into coherent in-distribution clusters, capturing the natural geometric structure of the data. At inference time, new samples are classified using MD from the centroid of the nearest cluster,

samples falling beyond a learned threshold are flagged as out-of-distribution. This design eliminates the dependency on labeled OOD samples entirely, and the graph-based architecture naturally scales to large datasets. Experiments on benchmark datasets demonstrate that the framework achieves competitive detection performance compared to supervised baselines while requiring no OOD labels, confirming its practical value for deployment in safety-critical applications where labeling OOD examples is not feasible.

1.6.2 CONTRIBUTIONS OF CHAPTER 4 (SUB-HYPOTHESIS 2)

- **Second major contribution:** We implemented a training framework for FM guidance with DR, enabling RL agents to generalize reliably across distributional shifts in dynamic, real-world environments.

C2: Improving RL Agent Robustness under Distributional Shift: In this contribution, we addressed the challenge of distributional shift in RL, where agents trained in one environment fail to generalize when deployed in another. How do FMs improve RL agent generalization beyond what sparse reward alone can achieve? How can DR reduce overfitting to a fixed training environment? Can combining these two techniques yield compounding robustness gains on unseen deployment conditions? How can we improve the robustness of AI models when faced with distributional shifts?

The second contribution proposes a novel approach that combines FMs and DR to enhance the generalization capabilities and robustness of RL agents operating in dynamic and uncertain environments. Distributional shift; the mismatch between training and deployment conditions, remains one of the most pervasive failure modes for RL-based systems. Standard training on a fixed environment yields policies that are brittle to even minor changes in layout, lighting, or task dynamics, undermining their reliability in real-world deployment.

To address this, the framework integrates two complementary techniques. First, FMs serve as structured guidance sources: given natural language documentation of the target environment, a large language model (LLM) generates task-specific mission objectives and

reward-shaping signals that provide richer supervision than sparse reward alone. This structured guidance accelerates learning and encourages policies that remain aligned with task intent across contexts. Second, DR systematically varies environmental parameters, such as visual textures, agent dynamics, obstacle placement, and sensor noise during training, exposing the agent to a broad distribution of conditions rather than a single fixed setting. Together, these two mechanisms force the agent to learn policies that are invariant to surface-level variations and grounded in task semantics. Ablation studies confirm that both components contribute independently to generalization, with their combination producing the greatest robustness gains on held-out environments that differ from the training distribution.

1.6.3 CONTRIBUTIONS OF CHAPTER 5 (SUB-HYPOTHESIS 3)

- **Third major contribution:** We evaluated the vulnerability of RL-based autonomous driving agents to observation-space, action-space, and composite adversarial attacks and two retraining-free inference-time defenses (QGap and DynIW), and provided empirical evidence consistent with an architecture-defense coupling: value-landscape gating was the more effective defense for a value-based DQN agent under intermittent action tampering, whereas DynIW restored actor-critic PPO agents under sustained fixed-command injection and intermittent action tampering, in HighwayEnv and in a MetaDrive continuous-control extension.

C3: Adversarial Robustness of RL Agents for Autonomous Driving: In this contribution, we addressed the challenge of adversarial attacks targeting the perception and decision-making pipelines of RL-based autonomous driving agents. How vulnerable are RL-based driving agents to adversarial attacks? What types of attacks cause the greatest degradation in driving safety? Can inference-time defenses matched to the policy architecture mitigate adversarial manipulations without requiring model retraining? How can we enhance the resilience of AI models against cyber attacks?

The third contribution systematically examines the vulnerability of autonomous driving agents to adversarial attacks and evaluating practical defense mechanisms that aim to improve safety, resilience, and reliability under driving conditions. As RL-based autonomous driving systems are increasingly considered for deployment, their susceptibility to adversarial perturbations, carefully crafted modifications to sensor inputs or actuation commands designed to corrupt agent perception or control, poses serious safety risks that must be characterized and mitigated before real-world adoption.

This work conducts a threat assessment of RL-based driving agents in simulated environments, evaluating observation-space attacks (noise and spoofing), action-space attacks (sustained fixed-command injection and epsilon-malicious tampering), and composite multi-vector attacks on Deep Q-Network (DQN) and PPO agents in HighwayEnv. In the evaluated settings, sustained fixed-command injection was the most damaging attack, driving the collision rate of both undefended agents to 1.000, and episodic reward alone proved an unreliable indicator of safety. Two retraining-free inference-time wrappers were evaluated: Q-Value Gap Thresholding (QGap), a value-landscape gate, and DynIW, which combines dynamic observation gating with action validation against the protected policy's own nominal action. The results are consistent with an architecture-defense coupling: QGap reduced the DQN collision rate under intermittent action tampering from 0.650 to 0.290, whereas DynIW restored the PPO agent under sustained fixed-command injection and intermittent action tampering, and its continuous-action branch reduced the collision rate of a frozen, externally pre-trained MetaDrive PPO checkpoint under fixed-command injection from 1.000 to 0.080. No wrapper resolved composite multi-vector attacks or recovered the DQN agent under sustained fixed-command injection, and the findings rest on a single training seed per HighwayEnv agent.

1.6.4 CONTRIBUTIONS OF CHAPTER 6 (SUB-HYPOTHESIS 4)

- **Fourth major contribution:** We proposed Co-Evo Defender, a co-evolutionary framework that continuously hardens jailbreak detection in LLMs by establishing an automated arms race between an RL-based attacker and an adaptive neural network defender.

C4: Co-Evolutionary Defense Against Jailbreak Attacks in LLMs: In this contribution, we addressed the challenge of adversarial drift in LLM safety classifiers, where static jailbreak detectors become ineffective as attackers continuously refine their evasion strategies. How can a jailbreak detection system remain effective against novel, previously unseen attack strategies? Can an RL-based attacker generate meaningful semantic perturbations that systematically expose classifier weaknesses? How does continuous retraining on discovered bypasses improve long-term defense robustness? How can we design an AI defense system that continuously adapts to counter emergent adversarial jailbreak strategies, maintaining reliable safety classification against out-of-distribution attack patterns?

The fourth contribution proposes **Co-Evo Defender**, a dynamic co-evolutionary framework that hardens jailbreak detection systems against adversarial drift in LLMs. Safety classifiers deployed to filter harmful LLM outputs face a fundamental challenge: as attackers discover bypass strategies, static classifiers that cannot adapt become increasingly ineffective. This adversarial drift problem is especially severe for jailbreak attacks, where semantic perturbations exploit the gap between a fixed classifier's decision boundary and the true space of harmful content.

Co-Evo Defender addresses this by establishing an automated adversarial arms race between two co-evolving components. An RL-based attacker, powered by PPO, continuously generates semantic perturbations designed to evade the current defender. These perturbations operate at the embedding level, enabling the attacker to probe for weaknesses in the classifier's latent representation space rather than relying on superficial lexical tricks. The adaptive neural network defender is then retrained on each successful bypass, transforming every evasion into a regression test that tightens the decision boundary. This feedback loop ensures

that the defender's knowledge of adversarial strategies grows monotonically over time rather than remaining frozen at training. The framework generalizes effectively against out-of-distribution attack patterns such as leetspeak obfuscation and policy-puppet payloads that cause static classifiers to fail catastrophically. Experimental results confirm that Co-Evo Defender achieves substantially higher recall on novel attack strategies than static baselines, while maintaining low false-positive rates on benign inputs.

1.7 LIST OF PUBLICATIONS

The work conducted throughout this thesis has resulted in the following peer-reviewed publications:

1. Salhab, W., Ameyed, D., Jaafar, F., & Mcheick, H. (2024). *A Systematic Literature Review on AI Safety: Identifying Trends, Challenges, and Future Directions*. *IEEE Access*, 12, 131762–131784.

2. Salhab, W., Ameyed, D., Mcheick, H., & Jaafar, F. (2025). *Towards Robust Artificial Intelligence: Self-Supervised Learning Approach for Out-of-Distribution Detection*. In *Availability, Reliability and Security (ARES 2025)*, Lecture Notes in Computer Science, Vol. 15994, pp. 106–121. Springer, Cham.

3. Salhab, W., Jaafar, F., Mcheick, H., & Ameyed, D. (2026). *Enhancing Agent's Robustness in Reinforcement Learning via Foundation Models and Domain Randomization*. *Neurocomputing*, Article 133133.

4. Salhab, W., Jaafar, F., Mcheick, H., & Ameyed, D. (2026). *Co-Evolutionary Defense for Reliable Jailbreak Detection Under Adversarial Drift*. Accepted for publication in the *International Conference on Software Quality, Reliability and Security (QRS 2026)*.

5. Salhab, W., Nakouri, H., Mcheick, H., Ameyed, D., & Jaafar, F. (2026). *Robustness of Reinforcement Learning Agents for Autonomous Driving: Adversarial Attacks and Defense Strategies*. Submitted to *Information Systems Frontiers*.

1.8 THESIS ORGANIZATION

The remainder of this thesis is organized as follows:

- **Chapter 2 - AI Robustness and Safety: Literature Review** defines AI robustness and its relationship to safety and reliability, then identifies the lifecycle robustness gaps that motivate the subsequent contributions.
- **Chapter 3 - SimCLR for Out-of-Distribution Detection** introduces the self-supervised, graph-based OOD detection framework addressing the integrity of training data (RQ1).
- **Chapter 4 - Safe Reinforcement Learning: Distributional Shift** presents the foundation-model-guided and domain-randomization-based approach for robust RL agents under distributional shift (RQ2).
- **Chapter 5 - Adversarial Robustness of Reinforcement Learning in Autonomous Driving** examines how architecture-aware inference-time defenses affect safety and task performance under adversarial attacks (RQ3).
- **Chapter 6 - Robustness of Jailbreak Detection Under Adversarial Drift** presents the Co-Evo Defender framework, which employs a co-evolutionary adversarial training loop to preserve detection performance under adversarial drift and out-of-distribution jailbreak attacks (RQ3).
- **Chapter 7 - Conclusion** summarizes the key findings and contributions of this thesis and outlines directions for future research.

CHAPTER 2
AI ROBUSTNESS AND SAFETY: LITERATURE REVIEW

2.1 CHAPTER OVERVIEW

AI is a rapidly evolving field that aims to make computers or machines capable of carrying out tasks that typically need human intelligence [7]. These include the abilities to learn, think, solve problems, perceive, and process language [8]. AI is frequently referred to as machine intelligence to distinguish it from human intelligence [9, 10].The field emerged from the convergence of computer science and cognitive science [11].

With the advancement of AI technology, researchers claim that quality and safety are neglected and misunderstood in a rush to improve system performance by some metrics often linked to financial or market-related factors [12].

Meanwhile, there is a wide variety of perspectives on AI and many more on AI safety. Safety engineering professionals worry that traditional principles may not suffice for systems that learn and adapt during operation [13]. They also fear these systems can't be validated against a precise specification in which safety and control are difficult to guarantee [14].

Ensuring the safety of AI systems is essential, primarily when utilized in safety-critical systems [15, 16]. Creating trustworthy, safe, and reliable AI systems is necessary to avoid human harm or discrimination against individuals or groups [17, 18]. Trustworthiness is crucial for safety-critical applications because it ensures that predictions made by a model can be trusted in scenarios where every choice has consequences [19, 20, 21, 22, 23] .

By prioritizing the system's safety and ensuring that these systems are transparent, predictable, and controllable by design, researchers can increase the trust and confidence in AI systems [24].

2.1.1 AI ROBUSTNESS AND ITS RELATIONSHIP TO AI SAFETY

In this thesis, **AI robustness** is the capacity of an AI model or system to maintain acceptable and reliable behavior when its operating conditions differ from the nominal conditions represented during design and training. Such differences may arise from previously unseen or OOD data, natural shifts between training and deployment environments, stochastic disturbances, component failures, deliberately crafted adversarial inputs, or attack strategies that evolve over time [25, 26]. Robustness is therefore not limited to small input perturbations or adversarial examples; it is a lifecycle property concerning how well an AI system preserves required behavior when its assumptions are challenged.

AI safety is the broader system-level objective of preventing unacceptable harm to people, property, society, and the environment throughout the AI lifecycle. It includes robustness, but also encompasses requirements such as human oversight, fairness, transparency, privacy, accountability, and fail-safe operation [27, 28]. **Reliability**, in turn, concerns the continuity of correct service over time. Robustness supports reliability by preserving performance under non-nominal conditions, and both support safety by reducing the likelihood that variation, uncertainty, or malicious interference will produce harmful behavior.

These concepts are related but not interchangeable. A model may be robust in consistently pursuing an incorrectly specified or unsafe objective, so robustness alone cannot guarantee safety. Conversely, a system that behaves safely only under nominal test conditions is not sufficiently robust for deployment in an open and changing environment. Robustness is thus a necessary, but not sufficient, technical foundation for AI safety: it provides evidence that safety-relevant behavior persists when data, environments, models, or threats depart from their expected conditions.

Accordingly, this chapter uses AI robustness as its central analytical concept and examines how it contributes to the broader objective of AI safety. Consistent with the main hypothesis in Section 1.5, this chapter organizes robustness around three connected lifecycle dimensions: preserving data integrity by recognizing unsupported inputs, maintaining model

reliability under distributional shift, and protecting deployed models against adversarial threats. The four technical studies in Chapters 3–6 operationalize these dimensions through Sub-Hypotheses 1–4.

AI models acquire knowledge through various techniques that may involve processing and analyzing large volumes of data to identify patterns, anticipate outcomes, and make predictions. Various AI learning processes, such as reinforcement learning, supervised learning, and unsupervised learning, are applied.

Machine learning (ML) is a subfield of AI constantly evolving and trying to mimic human intelligence through environmental learning [29]. ML is becoming common in cyber-physical systems [30]; these systems frequently use AI and ML techniques to function well in open contexts [31]. Because of their complexity, ensuring the safety and robustness of these systems is a significant problem. Therefore, the hazard may be presented in each of the life-cycle components: from data management, model development and analysis, and finally model deployment and monitoring [32, 6].

Many studies have discussed the usage of ML in safety-critical systems characterized by high levels of autonomy and safety. They may cause risks to human lives if misused [33]. These systems are deemed "safety-critical" or "high assurance" because they pose a risk of serious injury or financial loss in the event of failure [34, 35]. For example, in autonomous vehicles or healthcare [29], as well as solving problems in safety-critical systems using heterogeneous ensembles [36].

Therefore, the assurance of ML-based systems is one of the problems in safety-critical systems, where the goal is to provide a comprehensive and automated suite of processes for employing ML. This includes offering advice on how to design appropriate and safe testing criteria to gauge confidence in testing ML-based systems, such as how to generate tests efficiently, how to assess the quality and the size of test data, and how to ensure the robustness of ML models [6]

On the other hand, RL is a branch of machine learning that studies how intelligent agents

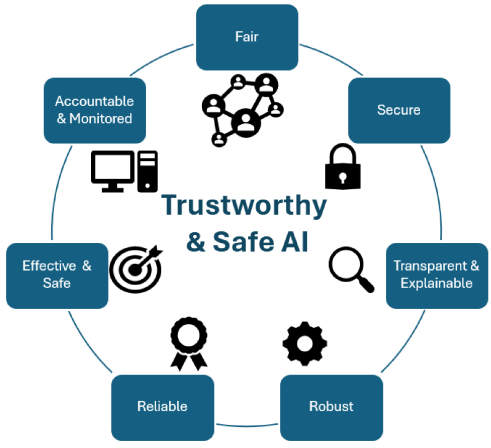


Figure 2.1 : Trustworthiness attributes of AI systems.

can maximize cumulative reward while interacting with an environment [37]. In contrast to normal supervised learning, RL focuses on presenting, analyzing, and manipulating current knowledge that does not explicitly correct sub-optimal behaviors or present proper input/output pairs [38]. RL agents must investigate their surroundings to discover the best policies through trial and error. However, in situations where incentives are few or where mistakes are undesirable, and safety is a top priority, exploration becomes difficult [39, 40].
A different approach might evaluate abstract problems like considering systems that might outsmart us or the ethical considerations on the usage of AI, for example, using the Generative Adversarial Network(GAN) such as the "deepfake" to spread disinformation [41] using Deep Neural Networks (DNN), or the legal definition and usage of AI. These topics are outside the scope of this chapter.

This chapter contributes a robustness-centered analysis of the AI safety and trustworthiness landscape illustrated in Figure 2.1. It examines where robustness is established or lost across learning paradigms and lifecycle stages, and it identifies the gaps addressed by the thesis contributions. This chapter is guided by the following questions:

1. *How is AI robustness defined, and how does it relate to AI safety and reliability?*

2. *How do learning techniques and lifecycle assurance practices preserve robust behavior under non-nominal conditions?*

3. *Which unresolved robustness gaps in data integrity, distributional shift, and adversarial security motivate the thesis hypothesis and its four sub-hypotheses?*

The rest of this chapter is organized as follows. Section 2.2 examines robustness and safety across learning techniques. Section 2.3 reviews assurance practices for robust and safe AI systems. Section 2.4 analyzes the dimensions, evaluation criteria, and supporting properties of AI robustness. Section 2.5 synthesizes the research gaps and maps them to the thesis hypothesis. Finally, Section 2.6 provides the chapter summary.

2.2 ROBUSTNESS AND SAFETY ACROSS LEARNING TECHNIQUES

Building on the conceptual distinction in Section 2.1.1, this section examines how different learning paradigms create distinct robustness requirements and how satisfying those requirements contributes to AI safety [27]. The organizing question is not only whether a model performs well on nominal data, but whether its required behavior persists when data, environmental conditions, or operational assumptions change.

2.2.1 ROBUSTNESS AND SAFETY IN REINFORCEMENT LEARNING

Agents face a trade-off between exploration and exploitation in RL, where exploration can lead to better policies and rewards over time. RL enables systems to anticipate the effects of their actions and adapt to changing circumstances to ensure safety [37, 42, 43].
For that, Ritz et al. [44] suggests integrating both functional and non-functional requirements to help the agents learn to comply with the specification. For the reward function, the study demonstrates how the suggested method enables agents to manage intricate safety limitations and meet a predetermined set of requirements.
Furthermore, the authors of [28, 45] outline the main issues with AI safety for RL presented; these concerns are:

- Safe exploration: Can agents interact with their surroundings without harm?
- Resistance to shifts in distribution: Can agents indicate uncertainty about the applicability of their model to new kinds of data instead of operating under unsuitable models?
- Reducing undesirable side effects: Is it possible to design the agent's incentive system to avoid hurting its surroundings without covering every possible case in detail and explicitly?
- Steer clear of wire-heading and incentive hacking: Is it possible to stop agents from falsifying their observations to maximize their reward?
- Scalable management: Even with delayed input, can agents adapt their behavior and learn the right one?

The current safe RL techniques are either not formally guaranteed or based on erroneous assumptions about the reward structure and state space [46]. RL agents frequently behave erratically when put in circumstances outside their training environment [47], which can result in decreased performance or safety violations.

For that, Burden and Hernández-Orallo [48] suggests three criteria for analyzing AI safety: capability, generality, and control. They contend that these variables can influence one another and the risk that artificial intelligence systems pose. They employ a cutting-edge technique based on Agent Characteristic Curves (ACCs), which plot an agent's success probability across a range of task difficulty levels to assess competence and generality. Capability is represented by the area under the curve, while generality is represented by the slope. They characterize danger as the likelihood of slipping into a pit in a grid environment and control as the inverse of the entropy of the states the agent visits.

Hunt et al. [46] proposed a system called VSRL to achieve verifiably safe exploration and reward optimization, which integrates object identification and automated reasoning.

Furthermore, Giacobbe et al. [49] provides a unique technique for evaluating and guaranteeing the security of Deep Reinforcement Learning (DRL) agents in the complicated and unpredictable settings of Atari games. They establish forty-three safety properties for thirty-one games and assess how well five cutting-edge DRL algorithms work with them. They present a countermeasure that increases the safety of all agents over many attributes by using bounded prescience to protect the agents from risky acts.

2.2.2 ROBUSTNESS AND SAFETY IN SUPERVISED LEARNING

Supervised learning is a subcategory of ML characterized by using labeled datasets to train algorithms that correctly identify data or predict outcomes [50]. This chapter examines safety at three levels: safe data processing, safe model development, and runtime safety. These levels represent the life cycle of an ML model. The objective is to guarantee the safety requirements of AI models at every level and analyze the current approaches to help meet these requirements [6]. Ensuring the safety of each stage is important to achieve the overall safety of any AI-based system. Table 2.1 summarizes the taxonomy used in this chapter by organizing supervised-learning safety approaches according to their life-cycle phase, proposed solution, and corresponding techniques.

Safe Data Processing: The test and training data quality used to validate and train the model is guaranteed at this point [83, 27]. It is a crucial phase for training and testing ML models, as it involves collecting, annotating, pre-processing, and augmenting data [56]. The quality and quantity of data directly impact model accuracy and robustness [76, 55, 56]. However, data processing can introduce hazards such as inadequate distribution, insufficient dataset size, and bias [63]. Inadequate distribution can lead to poor generalization and unexpected behavior [60, 21, 77]. At the same time, insufficient dataset size can overfit the model and fail to capture the complexity of the problem [22]. For that, several techniques are used to enrich the dataset, for example, using **geometric transformation** (rotation, crop, contrast enhancement, etc.) or **GAN** to generate or transform data [20, 56, 54, 55]. However, for those AI apps to become more trustworthy, the backend that handles the data collection

Table 2.1 : Taxonomy of ML Safety techniques

Phase	Solution	Technique
Safe Data Processing	Data augmentation	Active learning [51, 52, 53]
		GAN [54, 55]
		Geometric Transformation [20, 56]
	Data Privacy	Data Encryption [57]
	Bias Detection	Regularization approach [58, 59, 60, 55, 61]
		Topological analysis [62, 63]
		Hierarchical clustering [64, 54]
Safe Model Design	Model Transparency (explainable artificial intelligence and interpretability)	Visual representations [62, 61, 65, 22, 66]
		Counterfactual justifications [67, 12]
		Shapley Additive Explanations [68, 62]
		Saliency Maps [62, 65, 12, 66, 69]
	Robustness	Adversarial training [25, 51, 59, 70, 71, 72, 55, 61, 73, 74, 12, 21, 75]
		Transfer Learning [58, 59, 63, 20]
		Out-of-distribution [76, 20, 51, 60, 21, 77, 78, 47]
		Distance based prediction [19, 71, 73, 66]
		Model Extraction attack [79]
		Poisoning Attack [80, 73, 75]
	Fairness	Algorithmic Fairness [67]
Runtime Safety	Reliability	Fault diagnosis [22, 23]
		Local cross validation [71]
		Sensitivity analysis [71]
	Control	Autonomy [81, 82, 50, 19, 22, 23, 34]
	Monitoring	Anomaly detection [22, 23]

and processing should consider the participant's data privacy as well as data integrity and authenticity [84]. **Data encryption** techniques can be used to ensure data privacy; through this process, the information is transformed from its original format into a different format or cipher text. For example, Watermarking allows an optical encryption system to conceal the presence of confidential data [57].

Safe Model Development: Involves selecting, designing, training, and tuning machine learning models based on available data. It aims to optimize a predefined objective function, which

measures the error between model predictions and true labels [70]. However, model development can introduce hazards that compromise the safety of ML-based systems. These include incorrect objective function definition, inadequate performance measure, model complexity, and model uncertainty [82, 51]. The objective function defines the model's goal and evaluation, but if it's not aligned with the system's desired behavior or safety constraints, it may lead to unsafe results [71]. Choosing an appropriate performance measure that reflects safety requirements and trade-offs is crucial for ensuring safety. Model uncertainty, arising from sources like noise in data or parameter uncertainty, can also affect the safety of ML-based systems by causing incorrect predictions or decisions [58, 32]. Moreover, model attacks, which involve malicious attempts to manipulate or degrade an ML model, can also affect the safety of ML-based systems [61, 55].

Runtime Safety: To make sure that a ML model operates as best it can in an actual setting, deployment is an essential procedure that requires multiple steps.

It entails monitoring the model's output over time and looking for notable deviations or changes. When a model experiences problems such as concept drift, where the data it encounters no longer matches the data on which it was trained, monitoring can help identify when the model needs to be retrained.

The most fundamental function of fault diagnosis is fault detection; it checks for system malfunctions or faults and establishes the time at which they occur [19, 72]. It can be applied to a decision-making issue in machine learning [22, 23], and more details about failure modes are discussed in 2.3.2.

2.2.3 ROBUSTNESS AND SAFETY IN UNSUPERVISED LEARNING

Unsupervised learning methods, which identify patterns in data without predefined labels, require careful implementation and verification for various reasons. The input data's accuracy directly impacts the results, making it crucial to work with clean, well-prepared data. Understanding the results can be difficult as the identified patterns may not correspond to known categories. Additionally, these methods may yield different outcomes with slight

changes in data or parameters, underscoring the importance of conducting stability checks. Validation methods confirm the consistency of identified patterns with facts, evaluate the reliability of findings under varying circumstances, and prevent overfitting by identifying overly intricate patterns specific to the training data [85]. Therefore, while providing valuable insights, unsupervised learning demands careful application and thorough validation to ensure dependability and usefulness. More details about the validation methods are discussed in 2.3.1.

On the other hand, the validation of techniques requires careful consideration, mainly when dealing with **OOD** inputs. OOD is another category of risky inputs [76, 20, 51, 60, 21, 77, 78, 47] including those that fall outside the training data distribution. Detecting OOD is crucial in safely deploying machine learning models, especially in unsupervised learning. It involves the model's ability to identify and handle input data that differs significantly from its training data. This is particularly important in unsupervised learning since the lack of labels can make it difficult for the model to interpret such data accurately. Hence, OOD detection mechanisms are commonly incorporated into unsupervised learning systems to improve their safety and dependability.

Haider et al. [47] suggest a novel method for OOD detection in reinforcement learning by characterizing OOD as severe perturbations of the Markov decision process (MDP) and presenting a predictive algorithm utilizing bootstrapped ensembles and probabilistic dynamics models.

In addition, **distance-based** prediction can be used to classify OOD. For example, each input and its neighbor should be the same if the distance between them is less than a specific number. This can put a limit on the generalization error and give people more trust in the DNN's reliability promise [19].

Rossolini et al. [21] proposed an approach for this issue: to use coverage analysis to boost deep neural network confidence. According to experimental results, the proposed approach successfully detects strong adversarial examples and OOD inputs.

2.3 ASSURANCE OF ROBUST AND SAFE AI SYSTEMS

Robust behavior must be demonstrated and maintained at the system level rather than inferred from nominal model accuracy alone. This section examines three complementary areas of assurance research: verification and validation, analysis of failure modes, and human oversight of autonomous systems. Together, these practices establish the conditions under which robustness claims can contribute credible evidence to an AI safety argument.

2.3.1 VERIFICATION AND VALIDATION METHODS

Verification and Validation (V&V) are procedures to ensure a product, service, or system meets requirements and specifications using different methods and various formality levels [85]. Verification evaluates if a system meets design specifications, while validation ensures it meets customer and stakeholder needs. In systems, validation involves acceptance and suitability with external customers, ensuring the system meets user operational needs. In the context of AI dynamic intelligent systems, these systems require non-conventional methods for verification and validation due to their constantly changing context. Traditional V&V methodologies focus on whether systems behave correctly, but AI V&V presents unique challenges. Significant obstacles are the inability to clearly describe correctness standards for system outputs and the difficulty in standardizing non-conventional approaches due to adaptive systems' changing context.

Box-Jenkins technique is a well-known framework employed to recognize and resolve engineering hazards in adjusting and validating AI models to ensure AI safety [86]. The Box-Jenkins technique for AI safety essentially aims to minimize methodological gaps in the engineering of these systems and ensure that AI systems fail safely, especially in critical systems [87]. This is especially crucial because of the possible harm that specific AI algorithms may cause to human integrity and their high predictive capacity. The Box-Jenkins framework has three primary phases for AI safety, as shown in Figure 2.2. The initial stage, identification, is preparing the data and choosing the suitable model to fit the relevant data. To ensure the model is built correctly and accurately represents the real-world system it is meant to imitate,

the second phase, Estimation and Validation, estimates the parameters of the selected model and verifies the model's fit to the data. The last stage, Application, evaluates the chosen model's capacity for forecasting and applies it to actual situations. This iterative approach ensures AI systems' safe and efficient implementation, especially in crucial safety-critical applications. Moreover, Zhao et al. [70] highlights the shortcomings of existing testing methods that ignore the operational profile and may not increase software reliability. The authors introduce the idea of "operational adversarial examples" (AEs), which are more likely to arise in actual operations. They also recommend a novel testing methodology incorporating the operational profile to identify these AEs effectively.

Finally, the authors of [19] present a safety framework for deep neural network-based critical systems, highlighting the significance of verification and validation (V&V) for safe operation. It uses V&V methodologies and operational data to forecast system reliability using Conservative Bayesian Inference (CBI). The decrease of generalization error, a crucial component of system dependability, is linked by the framework to several stages of DNN life-cycle activities.

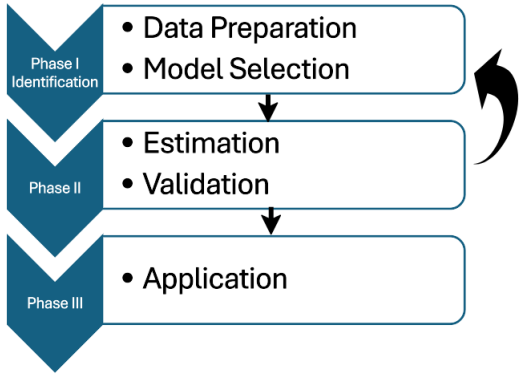


Figure 2.2 : The Box-Jenkins framework for safe AI

2.3.2 FAILURE MODES

A fault is an abnormal state or defect causing failure in a component, device, or subsystem [88]. Prediction foresees future failures, while diagnosis identifies and assesses the component's fault [22]. **Fault diagnosis** is frequently used to track down, detect, and pinpoint defects to increase a system's reliability [89]. Fault detection, isolation, and identification are the three steps that make up fault diagnosis [90, 91]. The most fundamental function of fault diagnosis is fault detection, which checks for system malfunction or faults and establishes when they occur.

Manheim [92] discussed several failure modes for multi-agent systems where the failure modes become more complex. The study examines several multi-agent system failure scenarios, such as unintentional steering, misaligned adversaries, faulty coordination, input spoofing and filtering, and goal co-option. Compared to single-agent failures, these modes are less known and more complicated. Moreover Martinez et al. [93] discusses the use of reliability analysis methodologies like Fault Tree Analysis (FTA) and Failure Modes Effects and Criticality Analysis (FMECA) in analyzing AI systems' learning and adaptability, highlighting potential issues with system reactions to malfunctions. It provides insights and encourages further research in AI safety, potentially influencing industry standards.

2.3.3 AUTONOMY, OVERSIGHT, AND SAFETY IN AI SYSTEMS

Human-AI interaction is studying and creating artificial intelligence systems that can communicate and cooperate. It is becoming popular, emphasizing the importance of designing systems that align with human values and work well with humans [22, 20]. Hence, the safety of such AI systems must be ensured to minimize any risk that threatens the human. Human-centered AI refers to the development of AI systems that prioritize human control and autonomy while also leveraging the capabilities of AI to enhance human performance [94]. With the use of machine-based methods, humans can directly complete activities that are challenging for computers in the pipeline and give training data for ML applications [95]. The human-centered AI framework attempts to accomplish these objectives by accounting for

high degrees of automation and human control. This strategy can aid in preventing mishaps and errors brought on by excessive automation or human control [96]. There is rising interest in researching more sophisticated learning strategies like **active learning**, which enables AI models to learn more successfully by actively choosing the data they learn from [92, 51]. Active learning is a semi-supervised technique in which only a part or subset of the training data is labeled [51, 52]. When the model determines that a particular data point is of interest, an interactive query is made to a human to label the data points from the unlabelled set.

Hagendorff [97] emphasizes the influence of social and psychological backgrounds on AI behavior to develop ethically sound and socially conscious AI applications and suggests a selective use of training data based on ethical judgments. These systems must be made safe by defining the conditions under which they are intended to operate, identifying potential dangers, and guarding against AI model and system hardware failures. Rajendran et al. define in [51] the ways that people can contribute to the learning process: Learning from demonstration, intervention, and evaluation are methods used by computers to learn from human actions.

In active learning, the premise is that a person will always be able to identify the correct label for a challenging data point. Some data points, though, might be difficult for human labelers to interpret. For example, it is assumed that the human did not make any mistakes while executing the demonstrations and that the demonstration data is a good sign of a safe policy. Although there is a chance that there may be ambiguity between various human evaluators, it is assumed that the human provides the proper reward for the right action [51].

Furthermore, Bravo-Rocca et al. [20] proposed a multi-agent system in which a human agent works with the other three agents to categorize anomalous events using the human-in-the-loop technique. This human involvement enables more precise data labeling and can raise the ML model's credibility. Human-in-the-loop systems may be safer than fully autonomous ones [86]. This approach can help make the model more trustworthy, safe, and reliable in real-world scenarios where every choice has some responsibility. This can lower the possibility

of inaccurate forecasts and increase safety in critical applications [20].

Moreover, Zhang and Cho [98] proposed the "Safe DAgger" method, which computes the difference between the expert-level actions of humans and those of agents and uses this as a decision measure for querying. If there is a slight difference between the actions, the agent action is sampled; if there is a significant difference, the oracle is consulted to determine the proper action. The difficulty in defining the metric and threshold choices for each application and circumstance is a downside of this method [98].

Finally, Mariani and Zambonelli [99] addresses autonomous vehicle coordination solutions, emphasizing the difficulties in choosing the right action strategies given the traffic situation in different scenarios. The result is that resolving these issues and considering ethical dilemmas in vehicle coordination is necessary to ensure the safe and conflict-free movement of autonomous cars.

2.4 DIMENSIONS AND SUPPORTING PROPERTIES OF AI ROBUSTNESS

AI robustness is challenged whenever the assumptions used to collect data, train a model, or validate a system no longer hold during operation. The literature identifies several sources of such deviation, including adversarial perturbations, poisoned training data, population differences, distribution shifts, and runtime faults. It also identifies supporting properties, including reliability, explainability, and monitoring, that help define, assess, and maintain robust behavior. Table 2.2 organizes these concerns according to their specific relationship with robustness.

The following subsections distinguish direct robustness dimensions from properties that support their evaluation and assurance. This distinction keeps the analysis focused on robustness without treating every AI safety or trustworthiness concern as equivalent to it.

2.4.1 SCOPE AND EVALUATION OF AI ROBUSTNESS

A robustness claim is meaningful only when it specifies three elements: the source of deviation, the behavior or property that should be preserved, and the metric used to determine

Table 2.2 : Dimensions and supporting properties of AI robustness.

Robustness concern	Role in robustness analysis
Scope and evaluation	Specify the deviation, behavior to preserve, metric, and acceptance criterion.
Adversarial robustness	Preserve safety-relevant behavior under deliberate input or action manipulation.
Training-data robustness	Limit performance degradation when training data are corrupted or poisoned.
Fairness constraints	Ensure that preserved aggregate performance does not conceal failures for particular groups.
Reliability	Maintain correct service over time, including under specified non-nominal conditions.
Distribution shift and OOD inputs	Detect unsupported inputs or preserve task performance when deployment differs from training.
Explainability and transparency	Diagnose sensitivity, reveal brittle features, and support evidence for robustness claims.
Runtime monitoring	Detect assumption violations and trigger intervention, fallback, or adaptation during deployment.

whether preservation is acceptable. For example, OOD detection concerns whether unsupported inputs can be recognized; robustness to natural distribution shift concerns whether task performance transfers to changed environments; adversarial robustness concerns whether safety-relevant behavior survives deliberate perturbation; and adaptive robustness concerns whether protection remains effective as an attacker changes strategy. These forms share the same core question: whether required behavior persists when assumptions are violated. However, they demand different mechanisms and evidence.

This distinction also clarifies the relationship between robustness and safety. Robustness describes behavior under variation or attack, whereas safety evaluates whether that behavior keeps risk within acceptable limits. Consequently, AUROC, task return, collision rate, recall, specificity, and attack success rate can provide evidence of robustness in different contexts, but their safety significance depends on the system and the consequences of failure. The thesis adopts this scoped interpretation when evaluating Sub-Hypotheses 1–4.

The remaining subsections apply this structure to the robustness concerns most relevant to the thesis. Adversarial attacks and poisoning introduce deliberate deviations, distribution shift and OOD inputs introduce natural or previously unseen deviations, and reliability, fairness analysis, explainability, and monitoring help determine whether required behavior is preserved acceptably.

2.4.2 ADVERSARIAL ROBUSTNESS

Adversarial robustness is the ability of an AI model or system to preserve specified behavior when an attacker deliberately manipulates its inputs, actions, parameters, or supporting software [22, 23, 78]. Unlike ordinary noise, an adversarial perturbation is selected to exploit a weakness in the model or its operational pipeline. A meaningful adversarial robustness claim must therefore define the attacker's knowledge and access, the allowed perturbation, the protected behavior, and the evaluation metric.

The literature addresses both model robustness and system security. Securing code and interfaces prevents unauthorized modification, while adversarial defenses seek to preserve model behavior after crafted inputs or commands reach the system. This distinction is important for the main hypothesis: security controls reduce access to the attack surface, whereas robustness mechanisms reduce the impact of attacks that still occur.

Adversarial examples are crafted feature perturbations that cause an ML model to produce an incorrect prediction [74, 21]. One model-level measure defines robustness through the average magnitude of the smallest successful perturbation across samples [71]. At system level, however, robustness may instead be measured through preserved task performance or safety outcomes under attack, because small prediction changes can have very different operational consequences.

Adversarial training exposes a model to crafted perturbations during learning so that it can improve its resistance to related attacks. Such testing also reveals behavior under worst-case inputs, where a single error may have serious consequences [22, 100]. Most studies in this area concentrate on creating ML models that are resistant to several adversarial attacks.

The following are some examples of the traditional adversarial defense strategies mentioned in [64, 74, 21] such as adversarial training, randomization techniques to lessen the impact of adversarial samples, and strategies for proving defenses based on well-defined attacks.

Tarchoun et al. [56] have studied the resistance of multi-view detection to recent adversarial patch threats by using perspective geometric transforms; the authors suggest an evaluation framework in which an adversarial patch is trained against a single view of a multi-view data set and then applied to the other views of the data-set.

A recent study has shown that DNNs aren't resilient to several attacks. The DNN cannot classify the adversarial instances, which contain images that are strikingly similar to the ones that were correctly categorized.

In addition, Mziou Sallami et al. [72] focuses on formal methods for NN-based object recognition systems and presents a novel method for evaluating the robustness of an NN-based image classifier. The authors offer a thorough method for assessing object recognition systems using Abstract Interpretation theory, which may be used to assess a DNN perception system's resilience against more general attacks.

Choi et al. [55] presents "Adversarially Robust Generative Adversarial Networks" (ARGAN). This novel GAN-based protection technique transforms the input data into ML models using a two-step transformation architecture. The target deep neural network model's vulnerability to adversarial instances is reflected in the generator model, which also optimizes its parameter values for a joint loss function. The authors show how ARGAN preserves the acceptable accuracy for legitimate input data while preserving the target deep neural network model's robustness against hostile samples.

Finally, Wang et al. suggest 'DRSL' [101], a technique that improves deep learning models' resistance to adversarial attacks without causing appreciable time consumption or accuracy loss. It suppresses distribution diversity in softmax values for non-real label samples. These approaches demonstrate useful attack-specific resistance, but they do not establish robustness

to attacks outside the stated threat model. This limitation motivates the architecture-aware and adaptive defenses examined by Sub-Hypotheses 3 and 4.

2.4.3 TRAINING-DATA ROBUSTNESS AGAINST POISONING

Training-data robustness concerns whether a learning process preserves model quality when the integrity of its training data is degraded. In a **poisoning attack**, an adversary inserts false samples, modifies features, or changes labels to alter the learned decision boundary. Unlike inference-time evasion, poisoning changes the model before deployment and can remain hidden behind acceptable aggregate validation performance.

A label-flipping attack replaces correct labels with attacker-selected alternatives so that the trained model produces targeted errors [75]. Robustness evaluation should therefore vary the poisoning rate and strategy and measure both overall degradation and targeted failure, rather than test only clean held-out data.

Maabreh et al. [73] propose a clustering-based label-flipping attack method and evaluate its effectiveness on several commonly used ML algorithms. The primary goal is to generate poisoned training samples that affect classifier accuracy as they go through outlier detectors. Their approach shows good results when using ML techniques such as k-nearest neighbors (KNN) or Binary Decision Tree (BDT), but it fails with Random Forest (RF) and DNN.

On the other hand, Maabreh et al. [84] examined how Particle Swarm Optimization (PSO) can enhance these models' performance in the presence of poisoned data and evaluated the performance of deep learning models under various poisoning rates. They expressed concern that PSO might mask or hide the effects of poison, which might result in inaccurate learning during the later stages of improving the model using more recent data. These findings connect poisoning resistance to the data-integrity component of the main hypothesis. They also show why anomaly detection and data provenance are needed before model-level robustness can be trusted.

2.4.4 FAIRNESS AS A CONSTRAINT ON ROBUSTNESS

Fairness and robustness are distinct properties, but fairness constrains what counts as acceptable preserved behavior. A model may maintain high aggregate accuracy under a distribution shift while its performance deteriorates for a demographic or operational subgroup. Such a model is robust according to an aggregate metric but not robust with respect to equitable service. Robustness evaluations in high-impact applications should therefore report subgroup performance and examine whether shifts in population composition expose hidden failures [19, 76, 54, 62, 59, 63].

Three main approaches address bias: pre-processing, in-processing, and post-processing. Pre-processing targets bias in data, in-processing modifies learning objectives or constraints, and post-processing adjusts predictions after training. From a robustness perspective, these interventions should be assessed not only on nominal data but also under plausible subgroup and distribution changes.

Kamishima et al. [102] proposed a **regularization approach** to resolve algorithmic bias based on analyzed unfairness. Moreover, Jaipuria et al. [54] proposes a unique **hierarchical clustering** approach that makes use of deep perceptual properties. Using deep perceptual characteristics and similarity metrics, deepPIC is a technique that organizes unlabeled picture files into semantically meaningful categories. It can assist in visualizing and comprehending the kind and degree of bias present in various datasets. One drawback is that deepPIC might not be able to discriminate between nearby semantically comparable clusters, particularly in cases when the clusters are small. Another constraint is that deepPIC uses the computationally costly scaled Learned Perceptual Image Patch Similarity (LPIPS) score to measure perceptual similarity between image pairs. A further drawback is that deepPIC extracts feature using generic backbones trained on ImageNet, which might not capture the task-specific bias for various vision tasks [54].

Furthermore, Srinivasan and Chander [63] suggests a method based on **topological data analysis** to detect biases before the use of a bias mitigation algorithm. Their method employs

persistence homology, a topological data analysis technique, to identify and measure bias resulting from various attributes in a dataset; it can identify clusters, holes, and voids at different spatial resolutions.

In addition, Kim and Cho [59] proposes an unbiased information bottleneck method with adversarial learning to achieve fair representation. It achieves the highest performance in reducing the algorithmic bias on various datasets.

Finally, AI systems are trained on human-generated data that may contain biases. It is therefore important to monitor whether performance remains consistent across relevant groups as data change. Zhao et al. [103] introduced a customized probabilistic measure for **bias monitoring**, using lifecycle activities to obtain prior knowledge about the measurements. Statistical inference is then performed under continuous observation to characterize bias in unlabeled and unstructured datasets. This use of monitoring illustrates how fairness analysis can expose robustness failures that aggregate performance measures conceal.

2.4.5 RELIABILITY SUPPORTED BY ROBUSTNESS

Reliability refers to the continuity of correct service over time. For ML-based systems, it includes service availability and the correctness of model outputs [71, 19, 65, 56, 77, 66, 22]. Reliability and robustness differ in emphasis: reliability asks whether correct service continues, whereas robustness asks whether required behavior is preserved when specified assumptions or operating conditions change.

Robustness therefore supports reliability under non-nominal conditions, but the concepts are not interchangeable. A model may be robust to a tested perturbation yet unreliable because of hardware faults or intermittent service failures. Conversely, a system may be reliable under nominal use but brittle under an untested shift. Evidence for the reliability component of the main hypothesis must consequently include performance under relevant deviations, not only duration of nominal operation. Model-agnostic or model-specific techniques are the two categories into which existing reliability estimate methods may be divided. Akram et al. [71] proposed a metric known as StaDRe (Statistical Distance for Reliability).

They estimate the reliability of machine learning forecasting techniques for time series data. It evaluates the effectiveness of the machine learning mode and detects distributional shifts using statistical distance measures based on the Empirical Cumulative Distribution Function (ECDF).

2.4.6 ROBUSTNESS UNDER DISTRIBUTION SHIFT AND OOD INPUTS

Following the general definition in Section 2.1.1, this subsection focuses on non-adversarial changes between training and deployment. Model robustness in this setting concerns the degree to which required performance is preserved on new data drawn from conditions that differ from the training distribution [25, 26]. Such claims must identify the evaluated shift and performance criterion; success under one dataset change does not establish robustness to every OOD condition. The ML literature evaluates robustness against dataset change and OOD inputs alongside the adversarial conditions discussed above. OOD recognition and generalization answer different robustness questions. Recognition determines whether an input lies outside the support of the training distribution and should be rejected or escalated. Generalization determines whether task performance can be preserved despite a specified shift. Uncertainty estimation, transfer learning, and adaptation complement these responses. Arnez et al. [60] suggests a technique to take uncertainty into account while generating Bayesian Deep Learning control strategies; their preliminary research demonstrates that the suggested approach strengthens the navigation policy's robustness in OOD circumstances.

In addition, Kshetry and Varshney [104] proposes a technique for reducing epistemic uncertainty and maintaining the safety of decisions by fusing algorithmic knowledge with physical system information using data-driven trained models. Finally, Kamoi and Kobayashi [77] proposed another method that can be used by applying a multi-modal prior distributions model, which may reduce OOD likelihoods and serve as an OOD detector.

On the other hand, and to increase the robustness of AI systems, **transfer learning** is the act of transferring knowledge from one source task to help them accomplish another target

activity more effectively [105]. Transfer learning offers a quick and easy method for creating effective ML models, especially when a lack of training data or computing power is available to complete the target job. This leads to robustness enhancement against new unseen cases. The goal of adversarial robust transfer learning is to increase the robustness and safety of a target model by transferring its robustness from the source model. A simple transfer learning technique is used to retrain the new model using the weights of the existing model. This results in a more robust model once the new enhanced data is added to the training set [20]. The distinction between recognition and generalization leads directly to Sub-Hypotheses 1 and 2: the first evaluates label-free OOD recognition, while the second evaluates performance preservation under an environmental distribution shift.

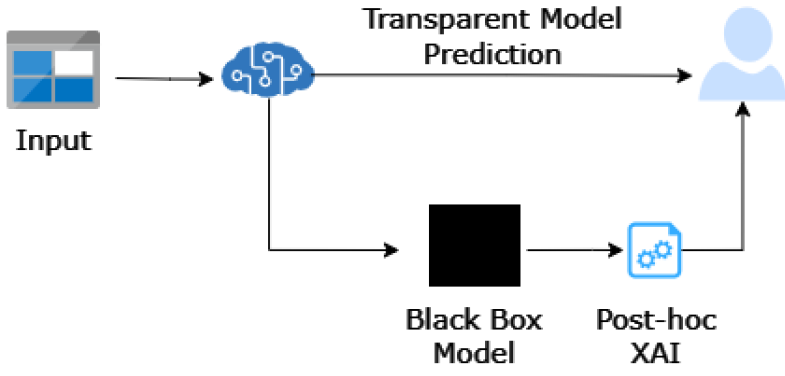


Figure 2.3 : Types of explainable artificial intelligence (XAI): transparent versus black-box models

2.4.7 EXPLAINABILITY AND TRANSPARENCY FOR ROBUSTNESS ASSESSMENT

Explainability and transparency do not by themselves make a model robust. Their role in robustness is evidential and diagnostic: they can reveal reliance on unstable features, clarify why predictions change under perturbation, and help determine whether a defense preserves the intended behavior rather than merely improving an aggregate score. XAI in-

cludes processes and methods that help users understand outputs produced by ML algorithms [65, 106, 67, 12, 62, 60, 107, 108].

Interpretability can be global or local, with global approaches explaining how a model produces predictions holistically, while local methods focus on specific data samples. There are two approaches to interpreting an AI system: model agnostic and model specific, as shown in Figure 2.3. Using agnostic methods separates the ML black boxes' from explanations; this post-hoc interpretation occurs after the model is trained and is unrelated to its core design. Other models are transparent by nature and simple to understand intrinsically. Thanks to their underlying representations, such as KNN, these models benefit from immediately supporting explanations. Many studies discussed the XAI; for instance, **counterfactual justifications** is used to describe how a prediction's outcome would have been different if particular characteristics had different values [67, 12, 69].

In addition, Sokol and Flach [67] used class-contradictory counterfactual assertions to explain decision tree models trained on two different data sets. These justifications can be used to find security and privacy flaws, test the underlying model, fix them, and eventually help make an AI system safer by identifying undesirable or dangerous behavior.

Cooper et al. [65] propose a new method based on **saliency maps** for clarifying and comprehending the predictions provided by AI systems. Saliency mapping can assist in making the inner workings of an AI system more apparent and accessible to humans by producing **visual representations** of the parts of input most crucial in influencing a model's output. According to the authors, their method may be helpful in various AI applications where explainability is crucial, but present methods are either too slow or have limited applicability. Also, Samadi et al. [69] propose 'SAFE', which guides the creation of more precise and relevant counterfactual explanations for DNNs in autonomous driving systems using saliency maps.

Local Interpretable Model-agnostic Explanations (LIME) explain models' predictions. It is employed to elucidate specific predictions made by machine learning models [108, 26, 25].

Users may learn how the model acts and which features have a major impact on the output by examining changes in predictions after perturbing input data points. LIME facilitates the creation of more understandable and transparent AI models, which aids in creating reliable AI systems.

On the other hand, Buczak et al. uses Shapley Additive Explanations (SHAP) to explain specific model predictions. SHAP is based on ideas from cooperative game theory; each feature is given a value indicating how important it is for a specific prediction. Applied across clean and perturbed inputs, such methods can help identify whether a robustness failure results from unstable feature dependence. Explanations themselves may also vary under small perturbations, so they should support, rather than replace, direct robustness testing.

2.4.8 MONITORING FOR RUNTIME ROBUSTNESS

Training and pre-deployment testing cannot enumerate every condition an AI system may encounter. Runtime monitoring supports robustness by detecting when inputs, internal signals, actions, or outcomes depart from validated operating conditions and by triggering an intervention, fallback, rejection, or adaptation. Monitoring therefore extends a static robustness claim into an operational control loop [50].

For that, Dementyeva et al. [109] uses the 'RADICS' system, which consists of black and white box monitors, to guarantee the security of cyber-physical systems managed by machine learning algorithms. If problems are identified, the system enters a safe mode, offering a thorough safety check for the decisions and results.

Monitoring can target different robustness signals. Input monitors can identify OOD data or spoofed observations; model monitors can track uncertainty or unstable representations; action monitors can block unsafe commands; and outcome monitors can detect performance drift. Zhao et al. [103] introduced a customized probabilistic measure for **bias monitoring**, using lifecycle activities to obtain prior knowledge about the measurements before applying statistical inference under continuous observation.

The monitoring signal and intervention must match the failure mechanism. A detector without an effective response does not preserve behavior, while an overactive monitor may degrade nominal performance. This architecture-dependent trade-off motivates the inference-time monitoring evaluated in Sub-Hypothesis 3 and the continual feedback mechanism evaluated in Sub-Hypothesis 4.

2.5 ROBUSTNESS RESEARCH GAPS AND ALIGNMENT WITH THE THESIS HYPOTHESIS

The literature establishes that robustness failures can originate in data, learned representations, deployment environments, or adversarial interactions. However, these problems are commonly studied as separate model-level concerns. The synthesis presented in this chapter therefore identifies a principal gap: not an absence of individual robustness techniques, but the lack of a lifecycle-oriented account connecting data integrity, model reliability under natural shift, and security under deliberate and evolving attacks. Table 2.3 maps this gap to the main hypothesis and the four sub-hypotheses developed in the thesis.

The table positions robustness as the technical thread linking the thesis contributions while retaining AI safety as the broader objective. Sub-Hypothesis 1 concerns whether unsupported data can be recognized; Sub-Hypothesis 2 concerns whether performance persists under natural environmental change; and Sub-Hypotheses 3 and 4 concern whether deployed protections resist fixed and evolving adversarial interference. Together, these gaps operationalize the main hypothesis as a program of empirical investigation rather than assuming that robustness follows from nominal accuracy or from any single defense.

2.6 CHAPTER SUMMARY

The literature synthesized in this chapter establishes AI robustness as a central technical foundation of AI safety rather than one marginal trustworthiness attribute among many. Robustness concerns whether required behavior is preserved when nominal assumptions are challenged by unseen data, environmental change, uncertainty, faults, or adversarial interfer-

Table 2.3 : AI robustness gaps identified in the literature and their alignment with the thesis hypothesis.

Robustness dimension	Coverage in the literature	Unresolved gap	Thesis response
Training-data integrity and OOD recognition	OOD inputs are addressed through uncertainty, distance, coverage, and supervised detection methods [76, 21, 77, 47].	Many methods depend on labels, known anomalies, or a single global representation of the training distribution. Label-free recognition of local, multi-cluster structure remains insufficiently addressed.	**Sub-Hypothesis 1, Chapter 3:** self-supervised representations, graph clustering, and Mahalanobis scoring without labeled OOD data.
Reliability under natural distribution shift	Safe RL, uncertainty modeling, transfer learning, and adaptation seek to preserve performance under changed conditions [28, 60, 105].	Semantic task guidance and systematic environment diversification are generally treated separately, leaving agents vulnerable to overfitting when both task context and visual conditions change.	**Sub-Hypothesis 2, Chapter 4:** integration of foundation-model guidance with domain randomization.
Inference-time adversarial robustness	Adversarial training, input transformation, and formal analysis improve resistance to specified perturbations [25, 74, 21, 72].	Most defenses modify training or assume a fixed attack and model interface. Lightweight protection for already-trained policies, selected according to policy architecture, remains limited.	**Sub-Hypothesis 3, Chapter 5:** architecture-aware inference-time monitoring without policy retraining.
Robustness under adaptive adversarial drift	Existing guardrails and classifiers are typically evaluated against fixed attack sets, even though jailbreak strategies evolve rapidly [110, 111, 112].	A static decision boundary cannot learn from successful bypasses. Continuous mechanisms that expose new blind spots and feed them back into defense training are needed.	**Sub-Hypothesis 4, Chapter 6:** a co-evolutionary attacker-defender loop for continual hardening.
Lifecycle integration	Safety and trustworthiness taxonomies enumerate data, model, runtime, and governance concerns [6, 27].	Robustness mechanisms remain fragmented across lifecycle stages, and no single mechanism covers natural uncertainty and evolving adversarial threats.	**Main hypothesis:** robust AI requires complementary controls for data integrity, model reliability, and adversarial security.

ence. AI safety asks the broader question of whether the resulting system avoids unacceptable harm. Robustness and reliability provide important evidence toward that objective, but neither replaces requirements concerning fairness, transparency, privacy, accountability, human oversight, or fail-safe operation.

Across reinforcement, supervised, and unsupervised learning, the literature shows that robustness must be considered throughout the AI lifecycle. Data quality, poisoning, and unrecognized OOD inputs can invalidate learning before deployment. During model development, objective misspecification, uncertainty, overfitting, and weak generalization can produce brittle behavior. At runtime, distribution shift, component faults, and deliberate attacks can violate assumptions that held during validation. Verification and validation, failure analysis, monitoring, and human oversight are therefore necessary to establish and maintain the conditions under which a robustness claim remains credible.

The literature also shows that robustness is not a single, context-free quantity. A defensible robustness claim must specify the deviation being considered, the behavior that should be preserved, and the metric and acceptance criterion used to evaluate it. OOD detection, task return under environmental shift, collision avoidance under attack, and jailbreak-detection recall under adversarial drift measure different properties. Their common role is to test whether safety-relevant behavior persists outside nominal conditions; their interpretation must remain bounded to the evaluated data, environments, models, and threats.

The synthesis in Table 2.3 identifies the principal research gap motivating this thesis. Existing work provides valuable techniques for individual failure modes, but these techniques remain fragmented across data integrity, model generalization, inference-time protection, and adaptation to evolving attacks. This fragmentation supports the structure of the main hypothesis in Section 1.5: robust AI requires complementary mechanisms that maintain the integrity of training data, preserve model reliability under distributional shift, and enhance model security against adversarial threats.

The four gaps lead directly to the technical studies that follow. The limited availability of label-free methods for recognizing local OOD structure motivates the self-supervised graph-based detector in Chapter 3 and Sub-Hypothesis 1. The need to combine semantic task guidance with systematic environmental variation motivates the foundation-model-guided domain-randomization framework in Chapter 4 and Sub-Hypothesis 2. The lack of lightweight, architecture-aware protection for frozen deployed policies motivates the inference-time autonomous-driving defenses in Chapter 5 and Sub-Hypothesis 3. Finally, the inability of static classifiers to learn from emerging bypasses motivates the co-evolutionary jailbreak defense in Chapter 6 and Sub-Hypothesis 4.

Chapter 2 therefore supplies the conceptual and literature-based rationale for the thesis hypothesis; it does not by itself claim empirical confirmation. Chapters 3–6 test the four sub-hypotheses under explicit experimental conditions, and Chapter 7 synthesizes how their results collectively support the main hypothesis.

CHAPTER 3

TOWARDS ROBUST ARTIFICIAL INTELLIGENCE: SELF-SUPERVISED LEARNING APPROACH FOR OUT-OF-DISTRIBUTION DETECTION

3.1 CHAPTER OVERVIEW

The rapid advancement of AI-driven systems has significantly impacted decision-making in various domains, including healthcare, finance, transportation, and security. However, AI systems often struggle with handling data from diverse sources, structures, and distributions, raising concerns about their robustness in real-world applications. Ensuring AI robustness is crucial, as these systems must consistently deliver reliable and accurate outputs under diverse conditions to maintain user trust and facilitate safe deployment and secure integration across multiple domains. This need is particularly pressing in high-stakes systems, where errors or incorrect decisions can have severe consequences and pose significant risk to human safety [113].

A key aspect of AI robustness is the ability to detect OOD samples, thereby preventing unreliable predictions. Simply put, OOD occurs when input data differs from the training distribution. OOD samples can originate from various sources, often arising from unseen situations the model has not encountered during training. Moreover, they can also result from adversarial attack inputs deliberately designed to mislead the model. Consequently, understanding how OOD samples impact AI systems is essential to making them adaptable to dynamic input data.

Thus, OOD detection involves identifying inputs that differ significantly from the training distribution, including previously unseen classes, random noise, or adversarial attacks. However, OOD detection remains a significant challenge. Many existing methods often struggle to accurately detect these outliers, especially when deep neural networks exhibit overconfidence, resulting in the misclassification of OOD data. We propose a self-supervised method

for extracting embeddings from unlabeled images using graph theory, which enables nuanced representations of both in-distribution and OOD data.

This study suggests a new OOD detection approach connecting self-supervised contrastive learning, graph-based clustering, and Mahalanobis distance. Unlike previous methods, our approach is a fully label-free, self-supervised OOD-detection approach, it requires only in-distribution unlabeled data, which improves scalability and supports deployment in real-world scenarios.

The integration of Graph clustering addresses the limitations of distance by modeling the distribution of data as a mixture of Gaussians, thereby improving the detection of near OOD samples. Unlike MAPLE, which relies on labeled data and Gaussian assumptions [114], our approach captures local data structures through graph-based clustering. Similarly, in contrast to SSD, which employs contrastive learning without explicit clustering [115], our method incorporates graph representations to enhance robustness, achieving an AUROC score of 0.99 on benchmark datasets, including CIFAR-10, CIFAR-100, and SVHN. The rest of this chapter is structured as follows. Section 3.3 provides a detailed explanation of the proposed approach, followed by experiments presented in Section 3.4. The challenges and limitations are addressed in Section 3.5. Finally, Section 3.7 provides the chapter summary, conclusions, and future research directions.

3.2 RELATED WORK

OOD detection has been an active area of research. As we propose a self-supervised approach for OOD detection, this section reviews relevant related works in the field.

3.2.1 MAHALANOBIS-BASED OOD DETECTION

MD is the distance between a data point and a distribution, and has been widely used in the literature for anomaly detection applications [116, 117]. In OOD detection, MD is frequently employed to distinguish far OOD samples from in-distribution samples, which differ greatly in meaning and style.

However, MD frequently fails to classify OOD samples close to the decision boundary and semantically similar to in-distribution samples [118]. To address this limit, a straightforward solution known as the relative Mahalanobis distance (RMD) has been implemented as a more robust alternative. RMD enhances performance and increases resilience to hyperparameter fluctuations [117]. RMD relies on predefined statistical assumptions that often do not generalize well across diverse datasets or complex OOD situations. Its effectiveness also decreases in high-dimensional settings where estimating function correlations is difficult.

3.2.2 OOD DETECTION USING SUPERVISED CLASSIFIERS

OOD classifiers help discover and classify data points that significantly deviate from a given data distribution [119]. This is typically achieved by training a model using binary or multi-class classification approaches [120]. [114] introduce MAPLE, which enhances OOD detection through predictive probability calibration. MAPLE dynamically clusters non-Gaussian class representations into multiple Gaussian distributions. The approach involves training a CNN using cross-entropy and triplet loss functions, combined with MD-based classification.

Their reliance on labeled data limits the effectiveness of many OOD detection methods, as they struggle to identify OOD instances that do not align with learned distributions. Additionally, the flexibility of these methods often relies on Gaussian assumptions, which may not always apply. This underscores the need for more adaptable OOD detection techniques that minimize the dependence on labeled data or rigid distribution assumptions. Contrastive learning models, for example, require large and diverse datasets to perform well and are sensitive to the selection of negative samples. Similarly, methods such as Maximum Mean Discrepancy (MMD) can be less effective in high-dimensional spaces, where distance metrics become unreliable. These issues underscore the necessity of developing robust OOD detection techniques that can effectively handle unlabeled datasets.

3.2.3 OOD DETECTION USING CONTRASTIVE LEARNING

Self-supervised contrastive learning has gained significant attention, especially in computer vision and natural language processing tasks [121]. The main objective of contrastive learning is to learn a representation of similar embedding in the feature space by grouping similar samples while pushing dissimilar ones apart [122].

Sehwag et al. [115] present SSD, a self-supervised system that utilizes unlabeled in-distribution data for OOD detection. SSD also incorporates training data labels and introduces enhancements for few-shot OOD detection.

Moreover, the authors of [123] proposed CADet, a fully self-supervised anomaly detection method that leverages contrastive learning and the Maximum Mean Discrepancy (MMD) metric. CADet examines whether two independent sample sets originate from the same distribution by combining contrastive learning with MMD, utilizing mean-based analysis.

3.2.4 OOD DETECTION USING GRAPH

Liu et al. [124] proposed a new approach for detecting OOD samples in graph-structured data. Their primary objective is to identify OOD graphs using unlabeled in-distribution data. They utilized the concept of contrastive learning to identify OOD graphs without the need for ground-truth labels [124]. Whereas, other methods make use of Graph neural networks (GNN) to detect OOD samples within graph data [125].

Although these methods adapt OOD detection to graph structures, they face several challenges. GNN-based methods, in particular, require extensive tuning and often struggle with the complexity of high-dimensional graphs. Additionally, they often encounter scalability issues when handling large, densely connected graphs.

3.2.5 LIMITATIONS OF PREVIOUS METHODS

In this chapter, we employ contrastive learning to extract embeddings from images in a self-supervised manner, enabling OOD detection without relying on labeled OOD samples.

Method	Labeled Data	Gaussian Assumption	AUROC
MAPLE [114]	Yes	Yes	0.996
SSD [115]	No	No	0.996
Deep Ensemble [126]	Yes	Yes	-
ViTB16 [117]	Yes	No	-
Ours	No	No	0.999

Table 3.1 : Comparison of OOD detection methods using CIFAR10 as in-distribution vs SVHN as OOD

The proposed approach offers numerous advantages, including its ability to effectively capture feature correlations and maintain robustness in high-dimensional spaces without requiring labeled data. We summarize the limitations of existing methods and highlight our contributions in Table 3.1. By eliminating the requirement for labeled data, reducing dependence on the Gaussian assumption, and optimizing computational efficiency through KNN-based graph constructions, our method provides a scalable and robust solution to detect OOD.

3.3 PROPOSED APPROACH

In this section, we provide a detailed explanation of our proposed approach. We present the proposed approach in Figure 3.1. It consists of two phases: The In-Distribution data representation phase and the OOD inference phase. The in-distribution data representation phase involves embeddings extraction and creating a clustered graph based on these embeddings. OOD inference phase consists of extracting OOD embeddings by calculating the Mahalanobis distances to the in-distribution clusters, and performing binary classification for OOD detection. It provides a schematic overview of the proposed approach, clarifying the workflow from embedding extraction to OOD classification. These visualizations highlight the method's ability to capture meaningful semantic structures, which supports its superior AUROC.

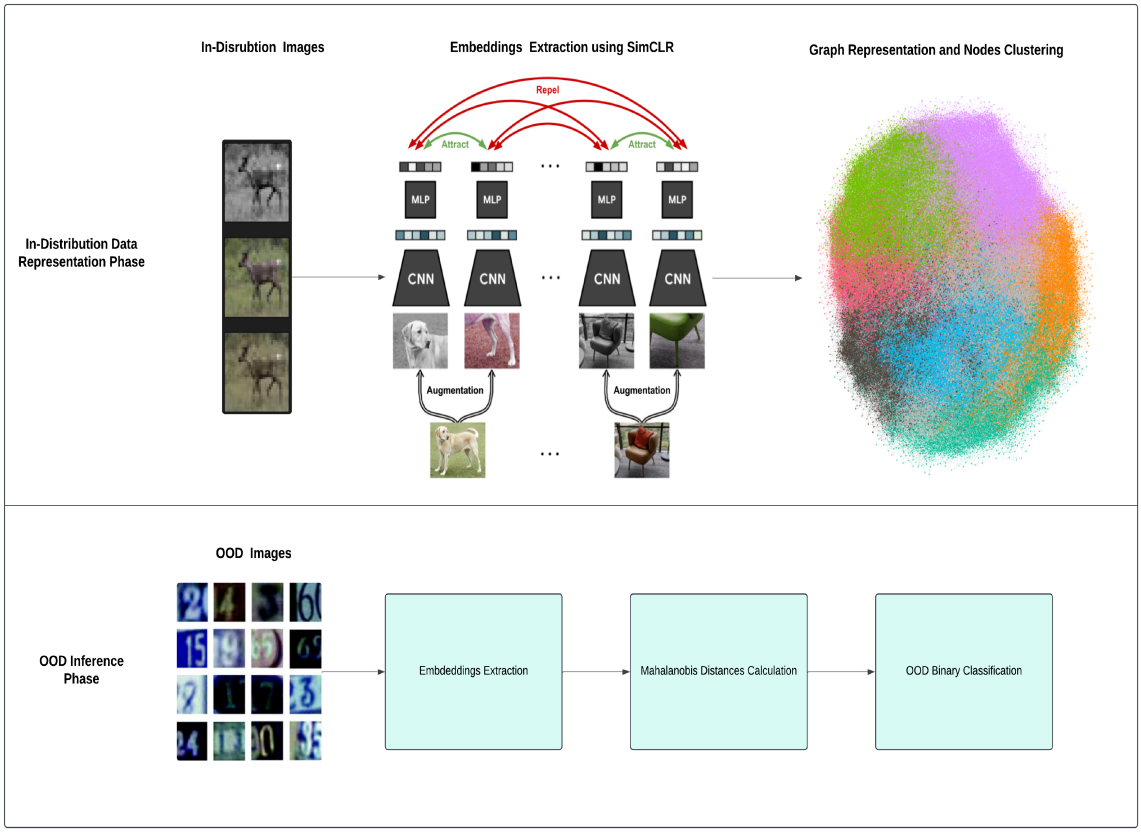


Figure 3.1 : Our proposed approach for OOD detection.

3.3.1 PHASE 1: IN-DISTRIBUTION DATA REPRESENTATION PHASE

This phase aims to group the in-distribution unlabeled samples into distinct clusters. It achieves this by extracting embeddings from input samples, constructing a representative clustered graph.

3.3.1.1 EMBEDDINGS EXTRACTION:

We employ self-supervised contrastive learning, a practical approach that enables models to differentiate between augmented versions of the same image and other distinct images, thereby learning strong representations from unlabeled data [127].
Furthermore, SimCLR is a well-known framework that uses a contrastive learning approach [128]. It simplifies contrastive self-supervised learning algorithms without the need for mem-

ory banks or specific architectures. SimCLR uses the infoNCE loss presented in Equation (3.1):

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} 1_{[k\neq i]} \exp(\text{sim}(z_i, z_k)/\tau)} \tag{3.1}$$

$$= -\text{sim}(z_i, z_j)/\tau + \log \left[\sum_{k=1}^{2N} 1_{[k\neq i]} \exp(\text{sim}(z_i, z_k)/\tau) \right]$$

where the batch size is represented as N, and τ represents the temperature parameter, z_i and z_j are two enhanced perspectives of the same case, and sim(., .) represents the similarity function between two examples. The similarity between these embeddings is then calculated using cosine similarity, which forms the basis of the loss function.

In this study, we use SimCLR to extract image embeddings. SimCLR outperforms supervised methods when fine-tuned with a limited amount of labeled data [115].

However, embedding can be high-dimensional and still contain redundant information, leading to higher computation cost and model performance degradation [129]. After extracting the high-dimensional embedding, we employ Principal Component Analysis (PCA) to represent the latent space in lower dimensions, which reduces computational costs and mitigates performance degradation.

3.3.1.2 GRAPH REPRESENTATION AND NODES CLUSTERING:

Once the embeddings are extracted, we construct the representation graph, each node represents an embedding. However, this method presents computational challenges, as a fully dense graph is inefficient and costly. To address this issue, we employ KNN to reduce the number of edges in the graph, allowing each node to connect only to its nearest neighbors. The edges of the graph are weighted using cosine similarity as follows:

$$\text{cosine similarity} = \frac{E_i \cdot E_j}{\|E_i\|\|E_j\|}$$

$E_i \cdot E_j$ represents the dot product of the two embedding vectors E_i and E_j, $\|E_i\|$ and $\|E_j\|$ stand for magnitude of the two vectors.

After constructing the graph, the Louvain method was used to group nodes within a graph into distinct clusters. This approach leverages modularity optimization to uncover the underlying structure of the graph, making it especially valuable for analyzing complex relationships and clusters [130].

3.3.2 PHASE 2: OOD INFERENCE

After identifying the clusters in phase 1, phase 2 focuses on classifying new samples as in-distribution or OOD samples. In this phase, we extract embeddings from new samples and calculate Mahalanobis distances from each embedding point to the centroids of the clusters identified in phase 1. This involves binary classification of data points that exhibit significant deviations from the established in-distribution data.

3.3.2.1 EMBEDDINGS EXTRACTION:

After we train the SimCLR model on the dataset, we extract embeddings from the OOD data during the inference phase. This method enables us to capture patterns and features without requiring labeled data, and it facilitates the calculation of Mahalanobis distances to the centroids of in-distribution clusters.

3.3.2.2 MAHALANOBIS DISTANCES CALCULATION:

Unlike Euclidean distance, MD considers the covariance between the variables as shown in Equation (3.2). In this study, md is the distance between an input and the nearest centroid of the in-distribution clusters.

$$md = \sqrt{(\mathbf{x} - \mu)^T \mathbf{S}^{-1} (\mathbf{x} - \mu)} \tag{3.2}$$

x is the input vector, μ is the centroid of the cluster, and **S** is the covariance matrix of the cluster. We use MD to calculate the distance of the samples from the centroid of the in-distribution clusters.

3.3.2.3 OOD BINARY CLASSIFICATION:

While a threshold is commonly used for practical classification, it is essential to differentiate between its role in decision-making and overall performance evaluation. Following the approach outlined in [131], the authors reported the effectiveness of setting the threshold at the 95th percentile of distances within the in-distribution dataset. In our study, we also set the threshold at the 95th percentile of the in-distribution data, solely to evaluate classification accuracy within this dataset. Although this specific threshold helps assess in-distribution classification accuracy, we use the AUROC as the primary evaluation metric for OOD detection. Unlike threshold-based methods, the AUROC is threshold-independent and measures the classifier's ability to distinguish between in-distribution and OOD data across various thresholds. This provides a more comprehensive assessment of the Mahalanobis distance-based OOD classifier.

3.3.3 THEORETICAL JUSTIFICATION FOR GRAPH CLUSTERING AND MAHALANOBIS DISTANCE

The proposed approach is integrated to enhance contrastive learning and graph-based clustering for OOD detection, utilizing the Mahalanobis distance. This section provides a theoretical foundation for why graph clustering enhances Mahalanobis-based OOD detection. Graph clustering, achieved through the Louvain method, involves grouping different in-distribution embeddings into distinct groups based on their latent space similarity. This process captures the underlying structure of the data and addresses the limits of the Mahalanobis distance in high-dimensional settings. The Mahalanobis distance is considered for

a single Gaussian distribution of data, which may fail when the data exhibits non-Gaussian characteristics [118]. Conceptually, our implementation models the in-distribution representation as a mixture of Gaussian components, with each cluster corresponding to a local Gaussian component. This means that the Mahalanobis distance is calculated with respect to the nearest cluster centroid, which improves the sensitivity of OOD tests near the decision boundaries. By computing md for every cluster and selecting the minimum value, the method identifies the nearest cluster and improves sensitivity to distributional shifts. The Louvain method optimizes modularity, ensuring that clusters are closely connected while being externally connected. This grouping reduces the effect of high-dimensional covariance estimation errors, as S is calculated on small, more homogeneous groups. In addition, the use of KNN reduces the graph calculation complexity by limiting the connection to the nearest neighbors, which minimizes the graph from $O(n^2)$ to $O(n^k)$. These savings preserve the local neighborhood.

3.4 EXPERIMENTAL RESULTS

Our primary results confirm the effectiveness of our proposed approach. Across several datasets, we observe a significant increase in the AUROC, achieving a maximum value of 0.999.

3.4.1 DATASETS

We validate our approach using widely used datasets in computer vision and machine learning, including CIFAR10, CIFAR100, and SVHN.

CIFAR10: consists of 60,000 images classified into 10 classes, each containing 6,000 photos. It includes 10,000 test images and 50,000 training images. This dataset is a commonly used resource for training computer vision and machine learning algorithms. In this study, no labeled training images were used.

CIFAR100: This dataset comprises 60,000 photos, each measuring 32x32 pixels, divided into 100 classes with 600 photos in each class. It includes 50,000 training images and 10,000

test images. The training images without labels were used in this study.

SVHN: This dataset includes 600,000 images of house numbers from Google Street View, including 73,257 training images and 26,032 images as the testing dataset [132]. The training images without labels were used in this study. The code and results of this study are available on GitHub [133].

3.4.2 EXPERIMENT SETUP

The SimCLR model is trained using a batch size 256 to extract image embeddings. The temperature parameter is set to $\tau = 0.1$, and the learning rate to 0.0001. A weight decay of $\mathbf{1e^{-4}}$ is applied and the training lasts over 300 epochs. We utilize 4 GPUs, including an NVIDIA 8GB V100, which completes the training in 8.795 hours. The process involves transforming the input images using various techniques, including Random Horizontal Flip, Random Resized Crop, Color Jitter, and Random Grayscale. These transformations adjust the images' brightness, contrast, saturation, and hue. The ResNet50 architecture serves as the base encoder, with an output dimension of 2048. The final layer of this model consists of a multi-layer perceptron (MLP), which includes a linear layer followed by a rectified linear unit (ReLU) activation function. The last layer functions as a projection layer with a dimension of 128. Before extracting features for the downstream task, we remove the projection head from the model. KNN was used for graph construction; we used $k = 7$ to balance cluster accuracy and computational efficiency (see Tables 3.2 and 3.3).

3.4.3 RESULTS

In this section, we present the results of our experiments, which confirm the effectiveness of our proposed approach in OOD detection. We evaluate the performance of our method using the AUROC metric, which measures the model's ability to distinguish between in-distribution and OOD samples across various thresholds. The results indicate that our approach achieves a high AUROC score of 0.999 on benchmark datasets, including CIFAR-

10, CIFAR-100, and SVHN. This confirms that our method effectively captures meaningful semantic information and is robust to domain shifts between datasets.

3.4.3.1 EMBEDDINGS EXTRACTION:

The result of SimCLR training with this setup is shown in Figure 3.2, which indicates an accuracy of 0.982 for top-5 validation on the CIFAR10 dataset. The model reaches a top-5 validation accuracy of 98.2% after 117000 steps for the CIFAR10 dataset. A top-5 accuracy indicates that the true labels are among the five highest predicted probabilities. After extracting embeddings, we utilize PCA to reduce dimensionality. Following the SimCLR training on CIFAR-10, we leverage the learned representations to extract embeddings for all other datasets. These embeddings are then utilized to evaluate the model's performance on downstream tasks, specifically clustering and OOD detection. The results confirm that these learned representations effectively capture meaningful semantic information despite domain shifts between datasets.

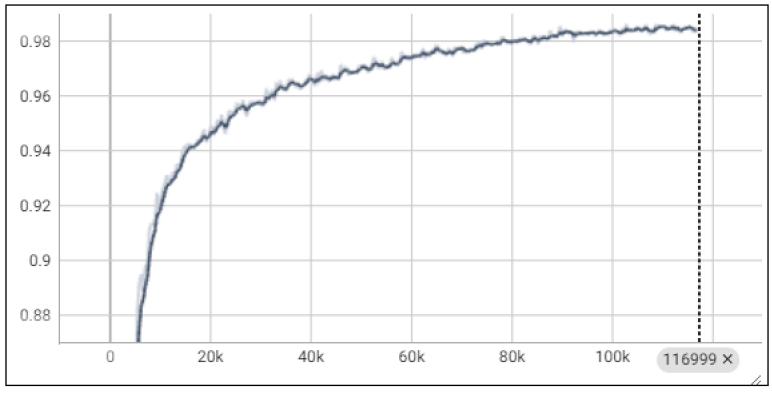


Figure 3.2 : Top-5 validation accuracy chart.

3.4.3.2 GRAPH BUILDING AND CLUSTERING:

Choosing k neighbors in KNN is essential as it establishes the graph structure. Instead of calculating and storing the full adjacency matrix, we connect each data point to its closest

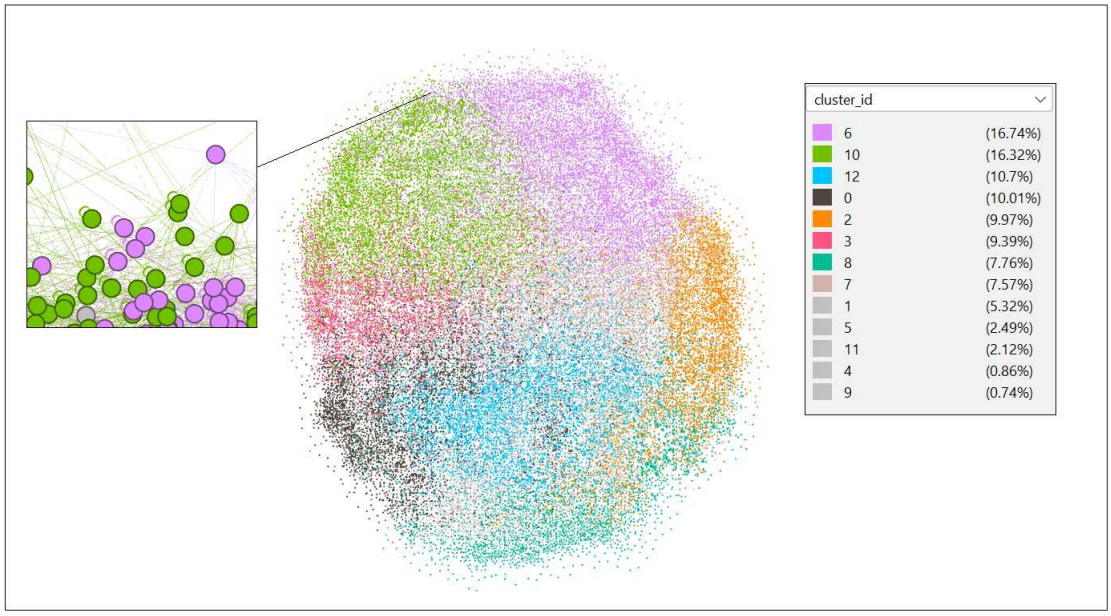


Figure 3.3 : Clustered graph: each node in the graph represents an image embedding in latent space.

neighbors. This graph reduces the calculation costs and the memory required to store the graph representation. This approach provides a sufficient efficiency advantage by reducing the density of the edges.

After creating the graph, we use the Louvain method to identify clusters within it. This method is designed to recover non-overlapping communities from large networks [134]. It involves calculating the degree to which the nodes in a cluster are more densely linked than they would be in a random network. Figure 3.3 shows the graph after performing Louvain clustering. The Louvain method produced 13 communities, shown using distinct colors, for CIFAR-10, whose ground-truth label space contains 10 classes.

Moreover, we test the number of neighbors to consider in the KNN with several values of k. The findings for CIFAR100 are shown in Table 3.2. When the number of neighbors k increases from 5 to 9, the number of edges rises from 180480 to 335785, while the number of clusters decreases from 167 to 90. Overall, the graph presents a total of 109 clusters at $k = 7$, which is close to the target of 100 clusters, the ground truth in the case of CIFAR100. At $k = 7$, the graph contains 258,385 edges and 50,000 nodes. The results for the CIFAR10

graph are presented in Table 3.3. As the number of neighbors k increases from 5 to 11, the number of clusters first decreases from 16 to 13. However, when k reaches 11, the number of clusters increases to 14. The selection of $k = 7$ resulted in a graph with 180,451 edges,

Table 3.2 : Using different k values for KNN with CIFAR-100 dataset; the optimal value of k is 7, resulting in 109 clusters, the closest to the ground truth in CIFAR-100 clusters.

k	Nodes	Edges	Clusters
$k = 5$	50000	180480	167
$k = 7$	50000	258385	109
$k = 9$	50000	335785	90

50,000 nodes, and 13 clusters, which is closest to 10, the ground truth in CIFAR-10. In contrast, $k = 9$ had 335,773 edges and the same number of clusters (13), but with more edges, resulting in a higher computational cost.

Table 3.3 : Using different k values for KNN with CIFAR-10 dataset; the optimal value of $k = 7$ results in 13 clusters, closest to the ground truth of CIFAR-10 clusters.

k	Nodes	Edges	Clusters
$k = 5$	50000	258280	16
$k = 7$	50000	180451	13
$k = 9$	50000	335773	13
$k = 11$	50000	412678	14

3.4.3.3 OOD DETECTION RESULTS:

We present the results of the OOD detection in Table 3.4. When we train the model on CIFAR10 (In-Distribution) and test it on CIFAR100 and SVHN (OOD datasets), our proposed approach achieves nearly the best AUROC and AUPR scores of 0.999 on both OOD datasets, significantly outperforming MAPLE [114], Deep Ensemble [126], SSD [115], and even the competitive Vision Transformer (ViT-B16) [117]. Similarly, when trained on CIFAR100 (In-Distribution) and evaluated on CIFAR10 and SVHN (out-of-distribution datasets), the

Table 3.4 : Performance of our approach compared to existing methods: CIFAR-10 (in-distribution), CIFAR-100 and SVHN (OOD).

Method	ID Accuracy	OOD (AUROC)		OOD (AUPR)	
		CIFAR100	SVHN	CIFAR100	SVHN
MAPLE [114]	0.956 ±0.01	0.926 ±0.01	0.996 ±0.01	0.918 ±0.01	0.997
Deep Ensemble [126]	0.964 ±0.01	0.864 ±0.01	-	0.885 ±0.01	-
SSD [115]	-	0.906	0.996	0.892	-
SSD_k ($k = 5$) [115]	-	0.931	0.997	0.919	-
ViT-B_16 [117]	-	0.998	-	-	-
Our	0.95	0.999	0.999	0.999	0.999

proposed approach maintains exceptional performance with AUROC scores of 0.999 and 0.997 for CIFAR10 and SVHN, respectively (Table 3.5), again surpassing all comparative methods. The proposed approach consistently exhibits substantial improvements across both evaluation scenarios, underscoring its robustness and effectiveness in distinguishing OOD samples.

Table 3.5 : Performance of our approach compared to existing methods: CIFAR-100 (in-distribution), CIFAR-10 and SVHN (OOD).

Method	ID Accuracy	OOD (AUROC)		OOD (AUPR)	
		CIFAR10	SVHN	CIFAR10	SVHN
MAPLE [114]	0.789 ±0.01	0.793 ±0.01	-	0.799 ±0.01	-
Deep Ensemble [126]	0.796 ±0.01	0.798	-	0.792 ±0.01	-
SSD [115]	-	0.696	0.949	0.645	-
SSD_k ($k = 5$) [115]	-	0.782	0.991	0.772	-
ViT-B_16 [117]	-	0.9442	-	-	-
Our	0.97	0.999	0.997	0.999	0.997

3.4.4 DISCUSSION

The results of our proposed approach demonstrate significant advancements in addressing OOD detection challenges. Our method begins by constructing a graph from embeddings through self-supervised learning, eliminating the need for labeled data. This graph-based structure effectively captures the local neighborhood relationships between data

points, providing a more nuanced understanding of the underlying manifold and enhancing the discriminative power for tasks such as clustering and OOD detection.

However, while the self-supervised approach significantly improves the incremental robustness of AI systems in detecting OOD, it is not without drawbacks. Its effectiveness heavily depends on the quality and diversity of the raw data. The integration of graph-based clustering proves highly beneficial, enabling more effective grouping of data points. By uncovering underlying structures within the dataset that traditional methods may overlook, our approach reveals patterns that can inform better decision-making in OOD detection scenarios.

Furthermore, leveraging Mahalanobis distance for similarity measurement with a graph enhances the robustness of our classification strategy. This metric provides precise distance calculations between data points and their respective clusters, enabling the model to maintain high classification performance even when encountering distributional shifts.

Overall, our findings underscore the practical applicability of the proposed approach. It offers a scalable and efficient solution tailored for real-world OOD detection challenges, paving the way for more reliable and practical machine learning applications across various domains.

3.5 CHALLENGES AND LIMITATIONS

OOD detection poses several challenges and limitations that hinder its implementation and efficiency in different applications. Understanding these barriers is crucial for developing more effective OOD detection methods.

One major challenge lies in the process of generating OOD samples, which remains poorly understood. This ambiguity creates significant obstacles in building reliable systems.

Another critical issue is setting an appropriate threshold for the in-distribution that effectively flags OOD examples without increasing the risk of false positives or false negatives. This remains challenging when dealing with unlabeled datasets. Calibrating a threshold that is both sensitive enough to detect OOD samples and precise enough to prevent misclassification

is a complex task, especially when dealing with an unlabeled dataset.

Moreover, failing to detect an OOD input can lead to poor decision-making or system errors, which is particularly critical in safety-sensitive applications. Conversely, the computational requirements for the OOD detection method increase with model complexity and the size of the data. For instance, the complexity of the Louvain method alone used in this study ranges from $O(N \log N)$ to $O(N^2)$, with memory usage reaching $O(N^2)$ due to the adjacency matrix, posing scalability challenges for very large datasets that are resource-intensive.

Moreover, the effectiveness of an OOD detection model is closely tied to the quality and diversity of the training dataset. This limitation highlights the need for comprehensive and representative datasets to enhance the model's ability to identify a wide range of OOD scenarios.

3.6 ABLATION STUDY

In this section, we tried to isolate the effect of the clustering algorithm and confirm the importance of graph-based embedding representations.

3.6.1 SUPERVISED OOD DETECTION USING KNN WITHOUT GRAPH REPRESENTATION

In this section, we implemented a direct neighbor (KNN) algorithm on raw data without performing any embedding extraction or graph representation. The purpose was to evaluate the performance of KNN in differentiating between the in-distribution data (CIFAR-10) and the OOD (CIFAR-100) data and to emphasize the role of embedding extraction on OOD classification. We evaluated different values of k and calculated the AUROC for each value. The consequences summarized in Table 3.6 indicate that the AUROC scores improve but are still limited, reaching a maximum of 0.54 at $k = 15$ for CIFAR10 and CIFAR100. This indicates that many neighbors can provide more effective discrimination between the two classes in this context.

Table 3.6 : AUROC Scores for Different Values of k

k neighbors	**CIFAR10(ID)** vs **CIFAR100(OOD)**	**CIFAR100(ID)** vs **CIFAR10(OOD)**
5	0.5292	0.5292
7	0.5319	0.5318
9	0.5360	0.5360
11	0.5381	0.5380
13	0.5392	0.5392
15	0.54219	0.5406

3.6.2 K-MEANS FOR NODE CLUSTERING

In this section, we attempted to implement K-means as the clustering algorithm instead of the Louvain method. However, the efficiency remains high, as shown in Table 3.7 while the in-distribution accuracy decreased from 0.97 to 0.95. The same holds for other datasets; the impact of the clustering algorithm was minimal. This confirms the robustness of our proposed approach, which relies on graph representation for data embedding to detect OOD instances.

Table 3.7 : Performance of our approach compared to existing methods: CIFAR-100 (in-distribution), CIFAR-10 and SVHN (OOD), using K-means where $k = 109$ (the same k as the number of clusters obtained via the Louvain method).

ID Accuracy	**OOD (AUROC)**		**OOD (AUPR)**	
	CIFAR10	**SVHN**	**CIFAR10**	**SVHN**
0.95	0.999	0.99	0.999	0.99

3.6.3 EFFECT OF NEIGHBOR COUNT ON OOD DETECTION

In this subsection, we explored the impact of varying the number of neighbors (k) on the AUROC.

To evaluate the sensitivity of our approach to neighboring parameters, we tested the entire OOD pipeline using values of $k = 5$, 7, and 11 during the detection process. We obtained the same AUROC score of 0.999, with only a minor difference of 7×10^{-6}. This indicates that as long as the graph remains connected, varying the number of neighbors (k) within this

range does not impact the performance of OOD detection. Therefore, we can fix k without requiring further hyperparameter tuning.

Table 3.8 : Overall classification accuracy as a function of percentile threshold τ applied to Mahalanobis distances.

Percentile Threshold	Accuracy
80	0.90
85	0.92
90	0.95
95	0.97
99	0.91

3.6.4 EFFECT OF THRESHOLD ON ACCURACY

While a threshold is often used for practical classification purposes, it is essential to distinguish between its role in decision-making and its role in overall performance evaluation. We tested different thresholds to evaluate their impact on accuracy as shown in Table 3.8. The adjustment from 80 to 99 confirms that our detector's accuracy is significantly stronger in the exact specified area, with an optimal range identified between 85% and 95%. Accuracy improves from 80% to 90% and reaches 95.5%. This increase reflects a better performance; however, when we push the threshold closer to 99%, the detector becomes overly conservative; for instance, valid CIFAR-10 images begin to be rejected, causing the overall accuracy to drop to 91.5%.

3.7 CHAPTER SUMMARY

This chapter proposed a novel fully unsupervised framework for out-of-distribution detection, combining self-supervised contrastive learning with graph-based clustering and Mahalanobis distance scoring. A key finding is that OOD detection is achievable without any labeled OOD examples, making the approach practical for safety-critical deployments where anomalous inputs are by definition rare or unknown in advance. The proposed framework demonstrated significant progress in enhancing AI trustworthiness through self-supervised

learning, achieving superior AUROC scores on benchmark datasets. For future work, integration with uncertainty estimation methods and AI systems deployed in safety-critical applications is planned to further enhance real-world trustworthiness.

Confirmation of Sub-Hypothesis 1. The proposed label-free pipeline achieved AUROC and AUPR scores of 0.999 when CIFAR-10 was used as the in-distribution dataset and CIFAR-100 and SVHN were treated as OOD, and it maintained AUROC scores of 0.999 and 0.997 when trained on CIFAR-100 and evaluated on CIFAR-10 and SVHN, respectively. These results confirm, within the evaluated image benchmarks, that self-supervised representations, graph-based clustering, and Mahalanobis distance can detect OOD samples without labeled OOD data, as stated in Sub-Hypothesis 1. By identifying inputs that fall outside the learned training distribution before their predictions are trusted, this contribution supports the training-data-integrity dimension of the main hypothesis in Section 1.5.

This chapter addressed the first dimension of AI robustness, specifically ensuring the *integrity of training data*, by detecting when inference-time inputs deviate from the distribution the model was trained on. However, detecting that an input is OOD is only a partial solution. A more fundamental challenge arises when the distribution encountered at deployment time diverges from the training distribution in a systematic and persistent way, that is, when the model itself is operating in an environment it was not designed for. This problem, known as *distributional shift*, is the central concern of reinforcement learning agents that are trained in one environment and deployed in another. Chapter 4 turns to this next dimension of AI robustness: how can RL agents maintain reliable and safe performance when the dynamics, visual conditions, or task objectives they encounter at deployment differ from those seen during training?

CHAPTER 4

ENHANCING AGENT'S ROBUSTNESS IN REINFORCEMENT LEARNING VIA FOUNDATION MODELS AND DOMAIN RANDOMIZATION

4.1 CHAPTER OVERVIEW

RL is a branch of machine learning in which agents learn how to make decisions by interacting with the environment [135]. This learning process follows a trial-and-error model, in which agents take actions and receive feedback in the form of rewards or penalties [136]. By maximizing aggregate rewards over time, agents develop strategies and policies that optimize their performance for specific tasks. This iterative process enables RL agents to independently identify effective actions without an explicit framework, making RL a powerful tool for solving complex decision problems. A classic example of an RL application is autonomous vehicles, where the vehicle navigates various driving scenarios, receives a positive reward for safe behavior, and is penalized for infractions such as traffic violations [137].

Despite its potential, RL faces significant challenges when dealing with distribution shifts and contextual changes. Distribution shifts occur when the statistical properties of the training environment differ from those of the processing environment [138].

Handling distribution shift in RL presents two major challenges: (1) reward shaping difficulties, where manually crafting reward functions across diverse environments often leads to sparse or misleading signals, hindering effective policy learning; and (2) visual discrepancies, where variations in lighting, textures, or object appearances across domains can cause the learned policies to fail when encountering unseen visual conditions. Such discrepancies often hinder training and lead to performance degradation in real-world scenarios, where an intelligent agent may struggle to generalize effectively beyond its training conditions [139].

Moreover, FMs are large AI models pre-trained on vast multimodal datasets, enabling them to perform a wide range of tasks with remarkable accuracy [140]. Unlike traditional AI

models, which are tailored for specific tasks, FMs are highly scalable and optimized for diverse applications with minimal additional training. These models leverage transfer learning, where knowledge acquired from one task is applied to another, making them particularly useful in domains such as natural language processing (NLP), Computer Vision (CV), and multitasking. Prominent examples include GPT-4.0, Gemini 2.0, and DeepSeek, which can generate human-like responses, understand language, create images from text, and exhibit advanced reasoning capabilities [141].

In addition, DR is a technique that enhances the generalizability of ML models, especially in RL and computer vision [142]. By systematically varying the properties of the training environment, DR exposes agents to samples presenting a wide range of conditions. A small part of the process involves visual aspects such as texture, color, lighting, and changing shapes during training in DR. This approach ensures that the models are robust to visual variation and perform well in different environments. It attempts to make models insensitive to unnecessary variations in input data by emphasizing consistent invariants between domains [143].

In this chapter, we propose Foundation-Model-Guided Domain Randomization (FMDM), a new approach to improve the robustness of the RL agent through FM guidance and DR. Our contributions are as follows:

(1) We introduce the dynamic mission string generation using Multimodal FMs to process text, images, and videos, providing reference-sensitive guidance that improves a single-modal approach (e.g., [144]).

(2) We develop an adaptive DR strategy that improves agents' generalization.

(3) We propose a modified actor-critic architecture that combines visual and textual inputs via Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRUs) modules, enabling the agent to learn more effective policies.

(4) Our method achieves better performance, with an average return of 0.85 compared to 0.3 for the PPO baseline on the MiniGrid environment.

(5) We propose an attention-based enhancement that guides the RL agent's focus toward goal-relevant information, offering a novel integration of mission context into the learning process.

The rest of this chapter is organized as follows. Section 4.3 presents the proposed approach. Section 4.4 validates the approach and presents the results, including the ablation study and component analysis. Section 4.5 discusses the limitations of this work. Finally, Section 4.6 provides the chapter summary and conclusions.

4.2 RELATED WORK

Recent studies have thoroughly examined the robustness of RL agents, concentrating on two main areas: the integration of FMs and the use of DR techniques. A significant trend in the field is the combination of these approaches to enhance generalization and improve resilience to environmental changes.

4.2.1 ENHANCING ROBUSTNESS THROUGH FMS

Recent literature highlights various ways that FMs are integrated into RL [145]. Firstly, LLMs are now utilized for high-level planning and structured guidance. For instance, the authors of [146] present a framework in which LLMs serve as "policy distillers." This means they utilize their extensive knowledge to train RL agents more effectively. This approach shows how LLMs can offer high-quality guidance, thereby improving sample efficiency and reducing the need for extensive interactions with the environment. Additionally, there is a shift from single-modality models to multimodal FMs that can process a variety of data types. Recent research has investigated multimodal FMs specifically designed for embodied agents, which use images, videos, and language to perform actions in both digital and physical environments [147]. These models show how multimodal FMs can create more adaptable agents capable of reasoning across various sensory inputs. This represents a significant advancement from previous vision-language model (VLM)-based methods, which focused on simpler tasks. Additionally, researchers are actively addressing the issue

of robustness. The approach of fine-tuning FMs using reinforcement learning, specifically known as Reinforcement Learning from Human Feedback (RLHF), has shown promise in aligning model behavior with desired outcomes while reducing the risk of catastrophic forgetting [148]. This illustrates a fundamental shift: researchers are now using RL not only to enhance FMs but also to improve the FMs themselves, thereby increasing their stability and generalizability. Additionally, curriculum learning has been implemented to train VLMs on progressively more complex tasks, which enhances their transferability and adaptability to new domains [149, 150].

Our approach aligns with the growing trend towards multimodal and adaptive methods. By dynamically integrating a multimodal FM with visual DR, we move beyond a single-modality approach, such as text-only systems. Our method leverages the FM's capability to process images, videos, and mission strings, allowing for a richer interpretation of the environment. At the same time, we utilize DR to ensure robustness against visual variations. This combination has enabled us to achieve a high mean return of 0.85 in the MiniGrid Distshift1 benchmark.

4.2.2 ENHANCING ROBUSTNESS IN RL THROUGH DR

DR is a powerful technique for enhancing the generalization of reinforcement learning agents. Recent advancements have progressed from basic parameter randomization to more sophisticated and adaptive methods.

Traditional DR is limited by the manual selection of randomization ranges, which can be inefficient. Automated DR frameworks have been developed to learn effective sampling distributions for environmental parameters, allowing the system to autonomously find the best randomization strategies [151]. This approach results in more robust policies that do not require human intervention. Researchers are also integrating DR with other advanced techniques to facilitate real-world deployment. A method called Continual Domain Randomization (CDR) has been proposed to combine DR with continual learning, enabling agents to safely adapt their policies to real-world conditions after initial training in a randomized sim-

ulation [152]. This addresses the ongoing issue of policies becoming inflexible and unable to adapt to changes in a real-world environment. A comparative study on visuomotor control indicates that excessive variability in dynamic response can lead to suboptimal performance and overfitting [153]. This emphasizes the need for a balanced randomization strategy to avoid agents learning specific, non-generalizable behaviors.

Our work builds on these insights by explicitly using visual DR, in which we randomize object locations and textures. This approach encourages the agent to learn from a wide variety of visual scenarios instead of relying on a fixed set of visual cues. As a result, the agent's policy becomes more robust to shifts in visual distributions, which is a common challenge in real-world applications.

In addition to the DR techniques previously discussed, there are complementary approaches that focus on robust adaptation through online generalization by explicitly modeling environmental uncertainty. For example, research in adaptive Bayesian learning has shown the ability to achieve fast and stable performance improvements in dynamic, large-scale environments, such as the Industrial Internet of Things (IIoT), by utilizing adaptive sampling [154]. This perspective on modeling uncertainty and adaptive learning provides a valuable complement to our method, which instead emphasizes bridging distribution shifts. We do this by combining semantic guidance from FMs with adaptive visual randomization.

4.2.3 LIMITATIONS AND GAPS

While previous studies have made significant progress, several limitations remain that our work aims to address. First, many earlier studies, including those that utilize FMs, have focused on single data modalities, such as text or images, in isolation. In contrast, our approach effectively employs a multimodal FM to process various data streams simultaneously, including images, videos, and text annotations (such as environmental documentation). This provides the agent with a more comprehensive understanding of the environment.

Second, existing methods often show limited generalizability when faced with progressively complex and unseen scenarios. To tackle this issue, our approach incorporates

an adaptive learning strategy alongside DR, enabling the RL agent to manage increasingly challenging tasks over time. This enhances adaptability and ensures effective policy transfer to highly dynamic real-world environments.

To our knowledge, this represents the first integration of FM guidance with adaptive randomization in this context. Our method bridges these gaps, as summarized in Table 4.1. While Zhou et al. [146] utilize LLMs, their approach relies on privileged state information, utilizing a mediator for ground-truth access and optimal pathfinding. In contrast, our FMDM framework operates directly on raw visual inputs, forcing the agent to learn spatial relationships and object identification without oracle assistance.

Finally, unlike methods that focus solely on multimodality [147] or domain randomization [153] in isolation, our approach integrates a Multimodal FM with DR to ensure robustness against distribution shifts.

Table 4.1 : Comparison of methods

Method	FM Usage	DR	Multimodality
Zhou et al. [146]	LLM	No	No
Liu et al. [147]	VLM	No	Yes
Dai et al.[153]	No	Yes	No
FMDM	**Yes**	**Adaptive**	**Yes**

4.3 METHODOLOGY

To mitigate policy collapse under shifts in both visual appearance and environment layout, we propose a framework that integrates two complementary components into a unified design: a multimodal FM and an adaptive visual DR strategy. This approach succeeds because the FM provides high-level, semantic task guidance via generated mission strings, while the DR strategy ensures the agent learns generalizable policies by exposing it to a wide variety of randomized layouts.

This training flow is illustrated in Figure 4.1 and detailed in Algorithm 4.1. The process begins when the DR module receives an environment observation and applies visual

perturbations (e.g., randomizing object and agent locations). This randomized state is then passed to the FM, which analyzes the visual input based on a prompt to generate a concise "Mission String". Finally, both the randomized state and the new mission string are fed into the RL agent's modified actor-critic architecture, guiding the policy update.

4.3.1 PROPOSED APPROACH

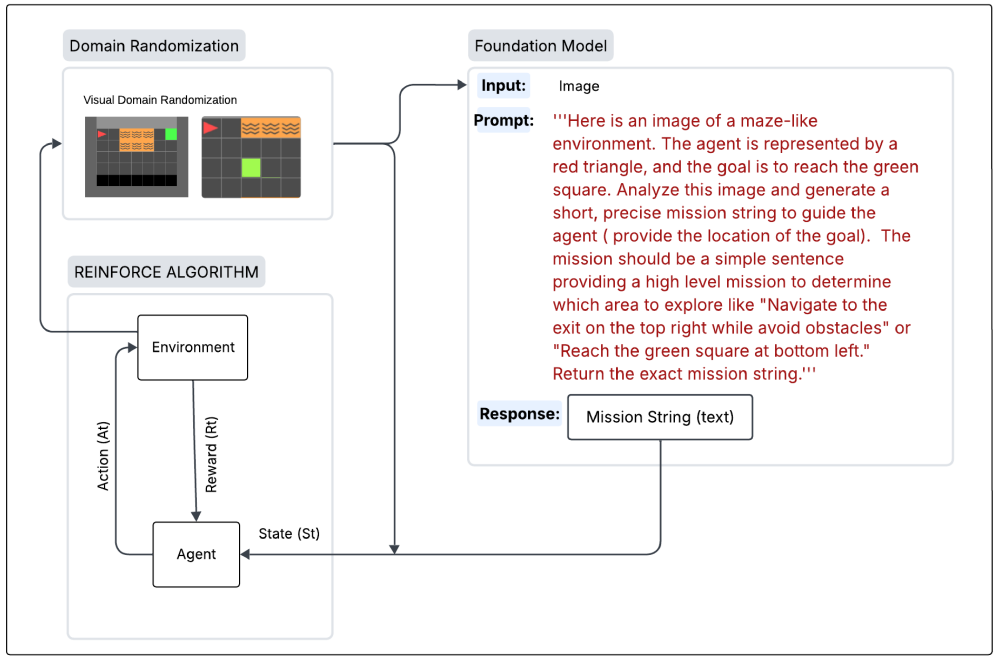


Figure 4.1 : Our proposed approach, illustrating the training flow. The DR module perturbs the environment, the FM generates a guiding Mission String from the state, and the RL Agent learns by combining both inputs.

In the proposed approach (Figure 4.1), the environment's observation is passed to the DR module to randomize the states. Then it is passed to the FM to create a mission string, providing relevant tasks for the RL agents. This dynamic interaction allows the FM to interpret and gather information from multimodal data and then generate the mission string for the RL agent. The combined features pass through a fully connected layer to produce the final output, represented by the policy network $\pi_\theta(a|s_t, h_t)$. This network provides the probability of taking action a given the current state s_t and the hidden state h_t. The parameter θ denotes

the policy parameters the agent learns during training.

The second network is the value network $V_\omega(s_t, h_t)$. This network estimates the total expected reward that the agent receives if it starts in state s_t with a hidden state h_t.

Input: Environment documentation D, FM, environment E, policy π_θ, value function V_ω
Output: Trained policy π_θ
Initialize π_θ, V_ω, FM;
for *each episode* **do**
    Generate mission string $M = \text{FM}(D, \text{prompt})$;
    Randomize environment $E' = \text{Randomize}(E, \text{perturbations})$;
    Observe state s_t from E';
    Process s_t, M through actor-critic (CNN + GRU);
    Select action $a_t \sim \pi_\theta(a|s_t, M)$;
    Execute a_t, observe reward r_t, next state s_{t+1};
    Compute TD error $\delta = r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t)$;
    Update π_θ using PPO clipped objective;
    Update V_ω to minimize δ;
end
return π_θ;

Algorithm 4.1: FMDM Training Flow

As a summary of the process, it begins with environmental observation, which serves as the basis. A human prompt, along with documentation, serves as input for the FM to generate a context-specific mission string that guides the agent in the environment. The RL agent interacts with the environment by considering the generated mission strings. It subsequently takes actions that lead to a new state and result in a reward.

4.3.1.1 FM AS THE GUIDE FOR RL AGENT

The interface between the FM and RL environments is highly advanced, effectively facilitating multimodal integration. This capability enhances flexibility in processing and utilizing contextual information, thereby optimizing operational efficiency and decision-making accuracy.

We use some of the most advanced FMs available to validate our approach, including GPT-4.0, Gemini 2.0, and DeepSeek-r1. While we choose GPT-4.0 for this initial demonstration, a more comprehensive evaluation of various FMs and their effects on mission string generation is presented in the ablation study, where we examine the performance of several alternative models for the same objective.

This initial validation with GPT-4.0 illustrates the potential of our approach to create diverse and robust mission strings, enabling the exploration of increasingly complex tasks and objectives for RL agents.

The prompt used to generate a mission string is as follows: "From this document, generate the mission string for the RL agent. Keep it clear and short." The model produces the mission: "Reach the green goal square located at the top right while avoiding lava." This brief provides the RL delegation with clear objectives and specific constraints, ensuring a focused and efficient workflow.

4.3.1.2 DR MODULE

While preliminary results from this developed network show promising performance, limitations arise in its ability to generalize across different environments. To address this issue, we integrate the concept of DR by training the RL agent in a diverse set of new environments. Specifically, we introduce random perturbations to the positions of objects within the simulated training environment.

We modified the Actor-Critic architecture for the RL training agent as a network that combines CNNs with a GRU for a predictive task. The images input into the CNN are processed through

a series of three convolutional layers, which are subsequently concatenated with the output from the GRU layer that processes the text data received from an embedding layer.
This technique exposes the model to a wider variety of visual inputs, effectively simulating real-world variations and uncertainties in object placement. By training the network to learn features that remain consistent despite minor positional changes, we aim to bridge the gap between the training environment and the testing environment, which may encounter real-time uncertainties and data distribution shifts. This approach is designed to enhance the model's generalization capabilities and improve its accuracy in predicting the next action.

4.4 EXPERIMENTAL RESULTS

This section provides a comprehensive overview of the research conducted, emphasizing the key findings derived from our proposed approach. To facilitate replication, the environment setup, domain randomization configurations, and source code are available in our public repository.[1]

4.4.1 METHODS EVALUATED

To ensure clarity in our comparative analysis, we evaluate the following distinct frameworks. All abbreviations listed below are used consistently throughout the results:

- **PPO-Base:** standard PPO with the default mission string.
- **PPO-DR:** PPO with domain randomization and the default mission string.
- **FM-PPO:** PPO guided by an FM-generated mission string, without domain randomization.
- **FMDM:** the full proposed method combining FM guidance and domain randomization.

[1] https://github.com/wiseresearch/FMDM

4.4.2 ENVIRONMENT

In this chapter, we utilize the MiniGrid environment, a versatile and lightweight grid-world designed for research in RL [155]. It provides a flexible yet straightforward framework for testing and developing RL algorithms. MiniGrid features various tasks and scenarios where the agent navigates the grid to achieve specific objectives, such as reaching a goal, picking up objects, or avoiding obstacles. The highly configurable environment enables researchers to modify parameters like grid size, object types, and reward systems, as illustrated in Figure 4.2.

Each observation in the MiniGrid is a partially viewable version of the grid surrounding the agent. This model is usually represented as a tensor with information about the visible cells, including the type of object in each cell (e.g. wall, floor, target, agent) and its type (e.g. color). The parameters received by the agent at each time step will be used to determine which operations to perform.

Moreover, the Distshift environment from the MiniGrid challenges the robustness of the RL agent when facing a distribution shift. Robustness ensures RL agents can operate reliably in various conditions and disturbances. We train our model using Distshift2, shown in Figure 4.3, and evaluate its performance on Distshift1 (presented previously in Figure 4.2). By

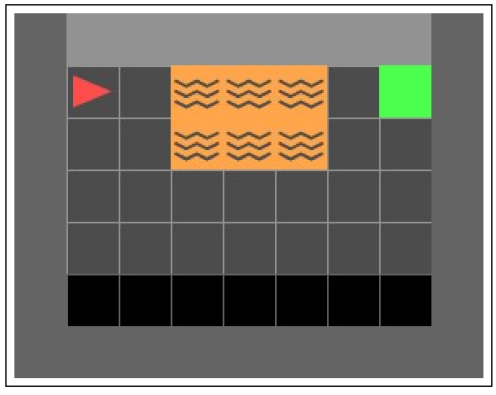


Figure 4.2 : MiniGrid DistShift1 environment: a grid-based maze where a player, represented by a red triangle, must navigate to a goal, depicted as a green square, while avoiding obstacles.

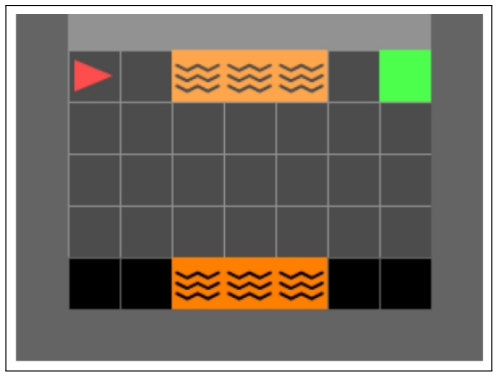


Figure 4.3 : MiniGrid DistShift2 environment

introducing different versions of Distshift, we aim to assess and improve agents' ability to navigate and adapt to new and unfamiliar situations. This approach helps enhance their overall capabilities and provides insights into how effectively RL systems manage change. The reward function can vary depending on the task and scenario in this environment. In our case, the reward is linked to the step count and the maximum number of steps the agent takes during an episode. The reward (R) can be defined as a function of the state (s), action (a), and the next state (s'). The action here can be of three options: move forward, turn left, or turn right.

The reward function is expressed as follows:

$$R(s,a,s') = \begin{cases} 1 - 0.9 \cdot \left(\frac{\text{step_count}}{\text{max_steps}}\right) & \text{for success} \\ 0 & \text{for failure} \end{cases} \tag{4.1}$$

4.4.3 SETUP

The training phase uses Google Colab (free tier), which provides 12.7 GB of RAM. GPU acceleration is not utilized in this setup. The training scenario is configured using Python, and necessary libraries such as TensorFlow and PyTorch are installed to support the implementation of the PPO algorithm. The hyperparameters for PPO are tuned as recommended in [156] as follows: The discount factor (0.99) signifies the importance of future rewards, favoring long-term gains. The learning rate (0.001) adjusts parameters during training, impacting convergence speed and stability. Generalized Advantage Estimation (GAE)

lambda (0.95) balances bias and variance in advantage estimates. The entropy coefficient of 0.01 encourages exploration and promotes a variety of actions. The value loss coefficient (0.5) balances the loss of the value function against the loss of the policy. Max gradient norm (0.5) limits gradient explosion for stability. The recurrence parameter (4) manages agent memory, affecting performance in partially observable environments. Adam epsilon $1x10^{-8}$ ensures optimizer stability, while clip epsilon (0.2) keeps policy updates reasonable. Training occurs over epochs (4) with a batch size of 256, impacting efficiency and convergence. These hyperparameters are crucial for practical PPO training in RL tasks. These parameters are as previously stated and are recommended for use with the stable baseline version of PPO [156].

4.4.4 RESULTS

In this section, we present the results of the proposed approach in two phases: first, we discuss the impact of the FM in RL standalone, and second, we examine the effect of adding DR on the mean return.

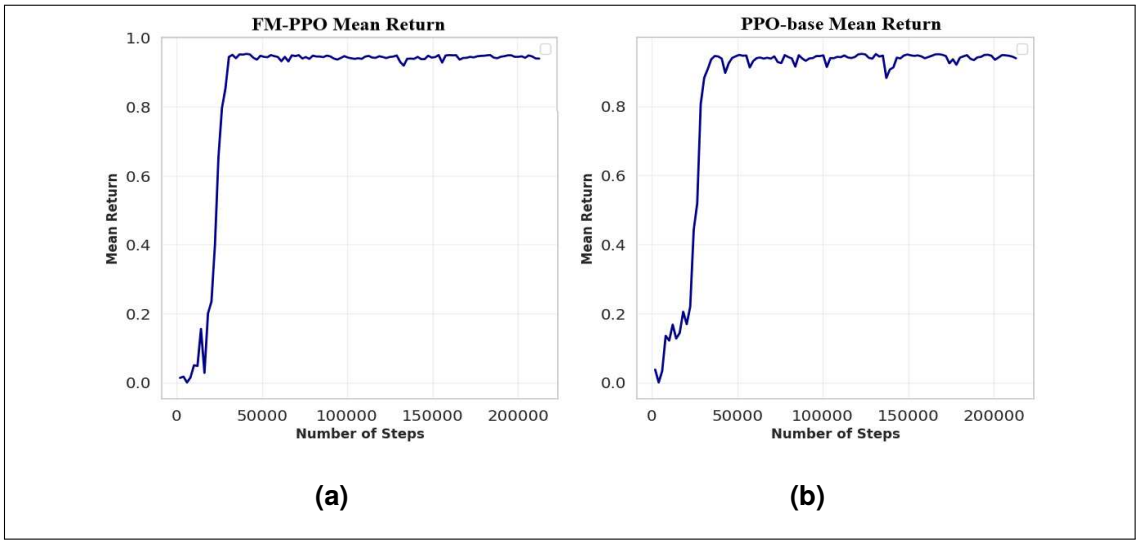


Figure 4.4 : Performance of different models in terms of mean return: (a) FM-PPO; FM standalone as a mission string generator, (b) PPO-Base with default mission string text.

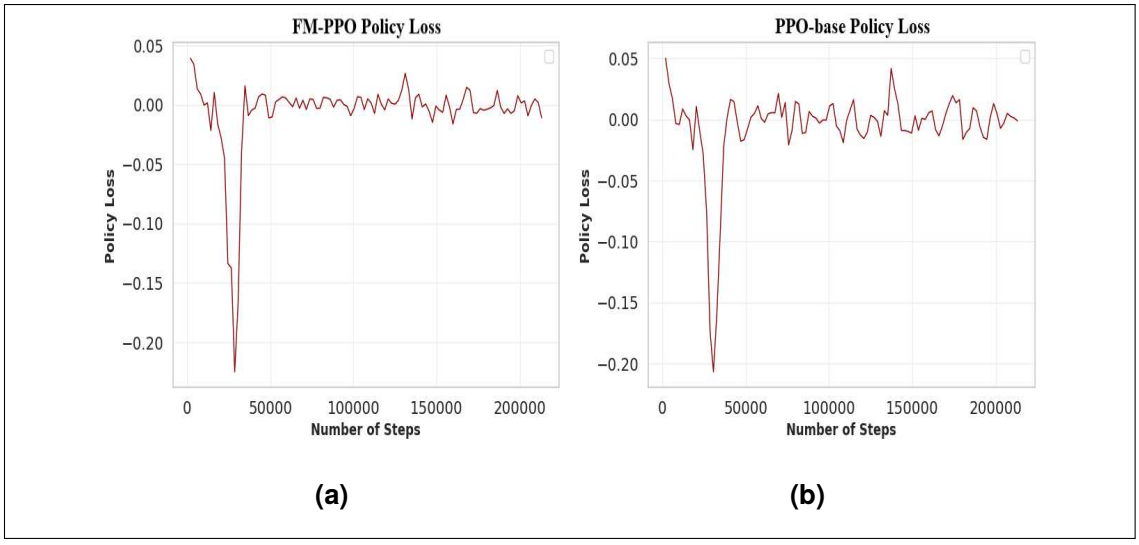


Figure 4.5 : Policy loss showing stable learning: (a) FM-PPO, (b) PPO-Base

4.4.4.1 MEAN RETURN ANALYSIS FOR RL AND FM

The mean return is the average cumulative discounted reward an agent expects from a given step onward, reflecting the long-term consequences of its actions. It is derived from the immediate rewards in Equation 4.1.

We present the mean return of training the "FM-PPO" on Distshift2 in Figure 4.4a, and the mean return from training the baseline (PPO) in the same environment is presented in Figure 4.4b. While both models exhibit similar performance profiles in this training phase, achieving parity here is significant. It confirms that replacing rigid, hard-coded instructions with dynamic FM-generated guidance does not degrade learning stability, thereby validating the integration before testing its generalization capabilities in unseen environments. Both models show an increasing trend in mean returns, reaching a final mean return of 0.939. The proposed approach took slightly longer to converge, with both models converging in about 6 minutes around the 100,000th step.

Figure 4.5 reveals that all models experience fluctuations in policy loss during training.

The plots for "FM-PPO" and "PPO-Base" in Figures 4.5a and 4.5b show that the policy loss fluctuates around zero as the agents adjust their strategies. The "FM-PPO" model initially

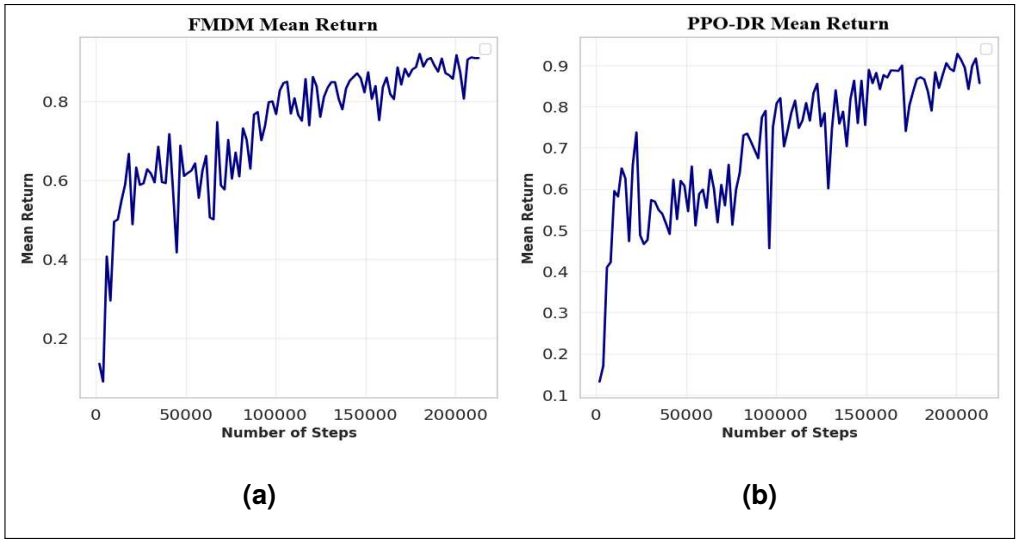


Figure 4.6 : Performance of different models in terms of mean return with DR: (a) FMDM (our); (b) PPO-DR; baseline with DR and default mission string text.

experiences a notable decline in policy loss during early training before stabilizing. This indicates the agent's ability to adapt to the diversity of instructions generated by the FM. Although there is initial variability, the loss gradually stabilizes as the agent learns to translate varied commands into effective actions, ultimately surpassing baseline performance. The "PPO-Base" policy loss also converges near zero but shows greater fluctuations compared to the "FM-PPO" model.

Despite these differences, both agents achieve similar final policy loss values (-0.0037 for

Model	**DistShift2 (Training)**	**DistShift1 (Testing)**	**DynamicObstacles (Testing)**
PPO-Base	0.9393 ± 0.2	**0.32** ± 0.2	0.44 ± 0.1
FM-PPO (Our)	**0.9397** ± 0.2	0.27 ± 0.2	**0.57** ± 0.2

Table 4.2 : Mean return for different models on DistShift1 vs DistShift2

"FM-PPO" and 0.0004 for "PPO"), indicating comparable training stability levels, albeit with different learning trajectories.

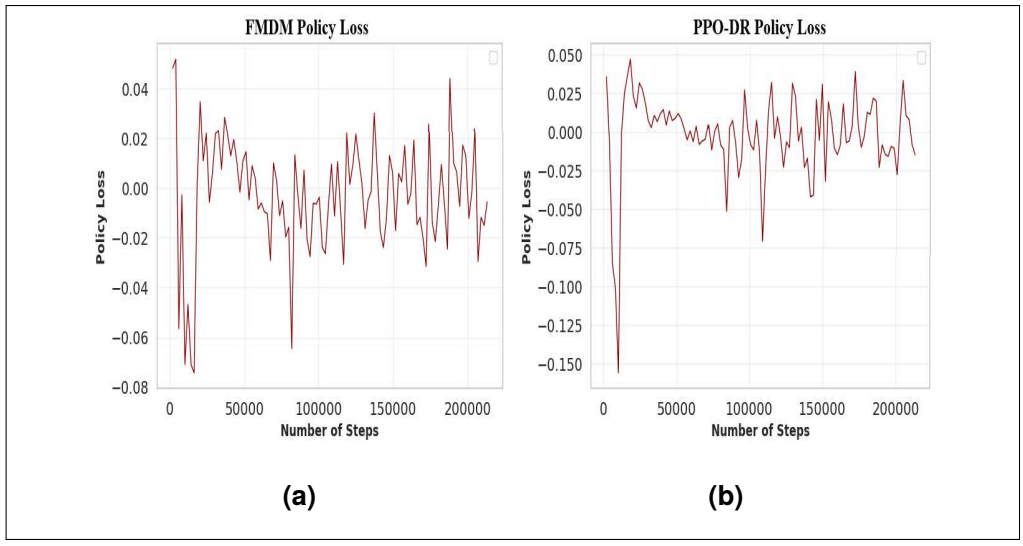


Figure 4.7 : Policy loss of different models: (a) FMDM, (b) PPO-DR. All policy losses stabilized around 0 ± 0.02 **after 200k steps**

Additionally, we compare the mean returns of different models in Table 4.2, using DistShift2 as the training environment and DistShift1 as the testing environment, with a particular focus on the testing mean return metric to evaluate robustness in the face of distribution changes. PPO (with default mission text usage) significantly decreases the mean return when testing the trained RL agent on DistShift1, revealing poor generalization and overfitting of the training data. "FM-PPO" also exhibits remarkable deterioration, although less pronounced than standard PPO, but it remains limited and overfitted to the training data. These results highlight a critical limitation: while the agents successfully learn the specific constraints of the training environment (DistShift2), they overfit to its visual and structural patterns. We explicitly quantify this performance drop to baseline the generalization gap. This confirms that bridging the difference between training and testing distributions requires more than just semantic guidance, justifying the integration of DR in our "FMDM" approach presented in the next section.

4.4.4.2 IMPACT OF DOMAIN RANDOMIZATION

Figure 4.6a presents the training chart of "FMDM" (our proposed approach with DR applied), and Figure 4.6b shows the training performance of "PPO-DR" (the baseline with DR applied). With over 220,000 training steps, both models exhibit an increasing trend in mean return, with "FMDM" initially starting slightly higher and maintaining a slight advantage throughout the training process. Both models appear to converge around the 150,000th step. While "FMDM" achieves a higher final mean return of 0.9085 compared to "PPO-DR" (0.857), the baseline appears to be more efficient, taking almost half the relative time (6.005 minutes) compared to the proposed approach (11.02 minutes).

While "FMDM" and "PPO-DR" achieve slightly lower final policy loss values compared to "FM-PPO" and PPO, it's important to highlight that the differences are pretty minimal. The smoothed values and the overall trends suggest that all models ultimately converge to a similar level of policy loss, indicating that they all learn effective policies.

Moreover, "PPO-DR" and "FMDM" exhibited slightly higher variability in terms of policy loss, as shown in Figure 4.7a and 4.7b. Both plots show fluctuations around zero, indicating ongoing adjustments to the agents' policies. However, there are key differences. The "PPO-DR" in Figure 4.7b has a smoother curve with fewer dramatic spikes, suggesting a more stable learning process. In contrast, "FMDM" in Figure 4.7a displays more frequent fluctuations, indicating a more volatile learning trajectory that may involve more drastic policy updates.

Despite these differences, both agents achieve similar final "smoothed" policy loss values: "PPO-DR" reaches -0.0046, while "FMDM" reaches -0.0088. This indicates that both algorithms perform at comparable levels. However, "FMDM" took nearly twice as long to complete its training run at 11.02 minutes, compared to 6.005 minutes for "PPO-DR," suggesting a difference in computational efficiency.

In Table 4.3, we evaluate "PPO-DR" (using the default mission string) and "FMDM" (our proposed approach). During training on DistShift2, our model, "FMDM", reveals the highest mean return of 0.9085, followed by "PPO-DR" (default mission string) with a score of 0.857.

Model	DistShift2 (Training)	DistShift1 (Testing)	DynamicObstacles (Testing)
PPO-DR	0.857 ± 0.1	0.74 ± 0.2	0.72 ± 0.3
FMDM (Our)	**0.9085** ± 0.1	**0.85** ± 0.2	**0.78** ± 0.2

Table 4.3 : Mean return for different models with DR on DistShift1 vs DistShift2

In the more challenging testing environment of DistShift1, "FMDM" again leads with a mean return of 0.85, while "PPO-DR" (PPO with DR using the default mission string) achieves a lower score of 0.74. The proposed approach in [146] achieves a return of 0.77. While this baseline was evaluated at 20,000 steps, we emphasize that a direct comparison of training steps is insufficient due to the baseline's reliance on privileged information. Zhou et al. [146] utilize a "Mediator" to access ground-truth internal states (such as specific object coordinates) and employ Oracle pathfinding (Dijkstra's algorithm) for navigation. In contrast, our "FMDM" approach operates in a significantly more challenging setting, relying solely on raw pixel inputs without access to internal object IDs or optimal path planning.

More details on the effect of each component (DR and FMs) are presented in Section 4.4.6.

4.4.5 IMPACT OF MISSION STRING DIVERSITY ON AGENT PERFORMANCE

To investigate how semantic variation in mission strings affects RL performance, we conducted a comparative analysis using instructions generated by different FMs: Default, Gemini 2.0, GPT-4, and DeepSeek. Each model produced a unique phrasing for the same underlying task, navigating to a goal while avoiding hazards. We quantified the semantic diversity of these mission strings using a sentence embedding-based variance metric.

Specifically, we calculated the mean squared distance of each mission string's embedding (utilizing the all-MiniLM-L6-v2 model) from the centroid of all embeddings. This analysis resulted in a Semantic Diversity Score of 0.2109, indicating a moderate degree of variation in how the task was described across the models. To assess the relationship between diversity and performance, we compared this score to the mean return achieved by agents trained with

each FM's mission string.

As shown in Figure 4.8, there is a clear positive correlation between semantic diversity and agent performance. The GPT-4 mission string, which was the most semantically rich and specific "Get to the green goal square located at the top right while avoiding Lava", resulted in the highest mean return of 0.85. In contrast, the default instruction "get to the green goal square" had the lowest semantic diversity and correspondingly the lowest mean return of 0.57.

This result indicates that having mission strings that are semantically diverse and rich in context leads to improved policy learning. This improvement is likely due to better grounding and more informative reward shaping. Interestingly, even minor differences in vocabulary or structure, such as specifying spatial orientation or including constraints, can significantly affect the agent's behavior and its ability to generalize. These findings emphasize the importance of diversity in instructions within agent training pipelines and encourage the use of prompts generated by FMs as a scalable method to enhance task specification in environments.

4.4.6 COMPONENT ANALYSIS AND ABLATION

We conducted an ablation analysis to assess the significance of the different FMs within our architecture and to comprehend their impact on the overall results.

4.4.6.1 PRELIMINARIES

Central to RL is the concept of an MDP, which provides a mathematical framework for modeling decision-making in situations where outcomes are partially random and partly influenced by the agent's actions. An MDP is defined by a tuple (S, A, P, R, γ), where:
S represents the set of states, A is the set of actions, P is the state transition probability function $P(s'|s,a)$, R denotes the reward function $R(s,a)$, γ is the discount factor.
This framework helps analyze and optimize decision-making strategies in various environ-

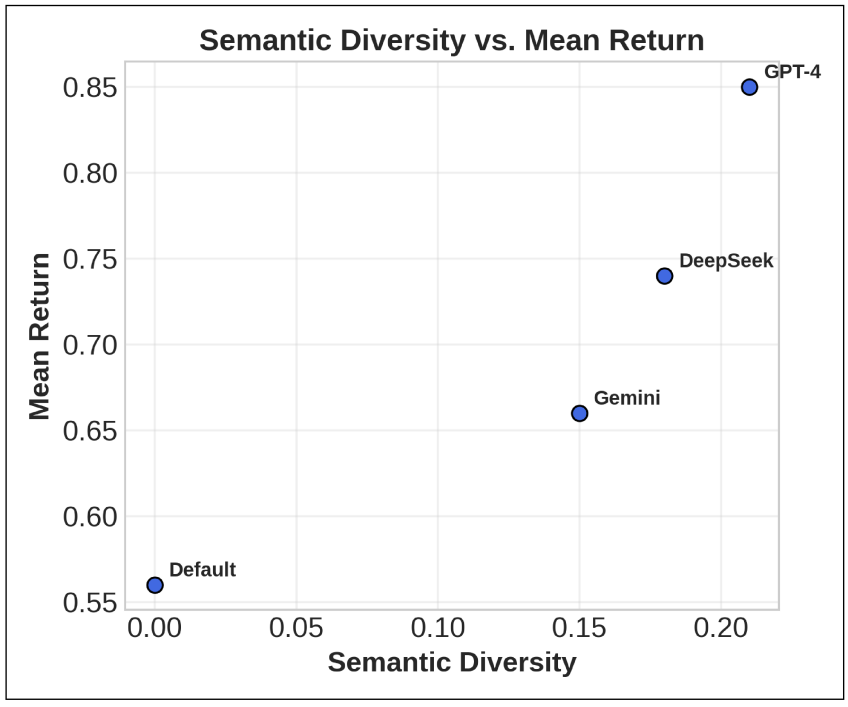


Figure 4.8 : Effect of the generated mission string on RL performance

ments. The goal in RL is to find a policy ($\pi : S \rightarrow A$) that maximizes the expected cumulative reward, defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The policy (π) determines the action (a) taken by the RL agent in a state (s). The objective is to discover an optimal policy (π^*) that maximizes the expected return from each state [157].

Additionally, the Actor-Critic method is a widely used approach in RL that merges the advantages of value-based and policy-based methods [158]. It includes two main components: the actor and the critic. The actor is responsible for selecting actions based on a policy ($\pi(a|s;\theta)$), where (θ) represents the parameters of the policy. The critic, on the other hand, evaluates the actions taken by the actor by assessing the value function ($V(s;\omega)$), where (ω) represents the parameters of that function.

The policy is updated using the gradient of the expected return, which can be expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\nabla_\theta \log \pi(a|s;\theta)\delta \right]$$

Here, (δ) represents the temporal difference (TD) error given by:

$$\delta = R + \gamma V(s';\omega) - V(s;\omega)$$

and (R) represents the reward received after taking an action (a) in a state (s), and (s') is the next state.

The critic updates its value function parameters (ω) to minimize the TD error:

$$\omega \leftarrow \omega + \alpha\delta\nabla_\omega V(s;\omega)$$

Here, (α) is the learning rate. By iteratively updating both networks, the Actor-Critic method aims to simultaneously improve the policy and value function, leading to more efficient learning. Moreover, the PPO algorithm utilizes an actor-critic framework to achieve robust and efficient learning. PPO enhances traditional policy gradient methods with a surrogate objective function that prevents substantial changes or updates to the policy, ensuring more stable and reliable learning. The basic idea is to optimize the clipped objective function by limiting changes to the structure at each update step. PPO objectives can be expressed as follows:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

where ($r_t(\theta)$) is the probability ratio between the new and old policies, and it is expressed as follows:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

The clipping parameter (ϵ) controls the degree to which the new policy can deviate from the old one.

In this chapter, we used PPO as the baseline for comparison. PPO is closely related to advantage actor-critic methods [159], where the actor represents the policy ($\pi_\theta(a|s)$) and the critic estimates the value of the function ($V(s;\omega)$). The advantage function ($\hat{A}_t$) is usually obtained from the critic's value estimate, which measures how much better or worse an action is compared to the expected value.

4.4.6.2 PERFORMANCE COMPARISON ACROSS SEEDS

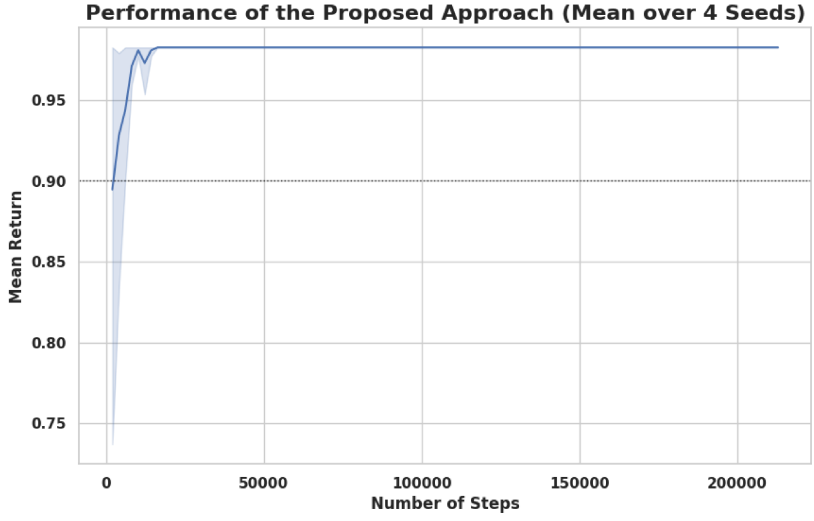
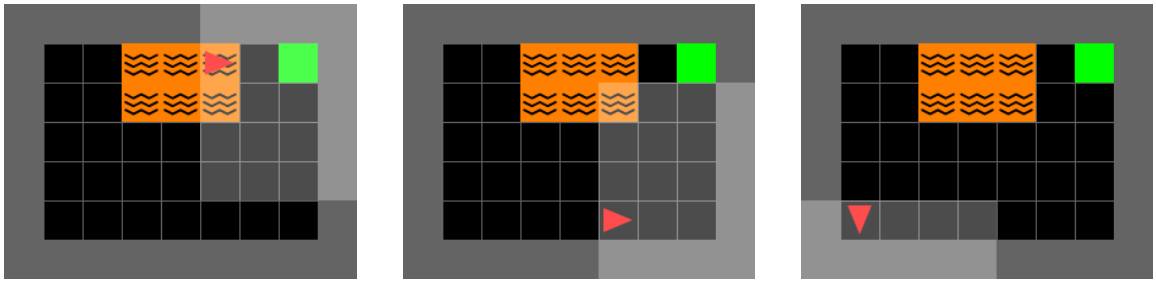


Figure 4.9 : Performance of the proposed approach using different seeds

As shown in Figure 4.9, this analysis shows the exceptional stability and robustness of our proposed approach. The mean return (solid line) rises rapidly, achieving near-optimal performance (a mean return of 0.98) in fewer than 25,000 steps.

The 95% confidence interval (represented by the shaded region) remains remarkably narrow

throughout the evaluation. Following the initial learning phase, the interval becomes nearly imperceptible, indicating minimal variability across independent runs. This observation provides strong evidence that the algorithm's performance is consistently reproducible and not attributable to a particular random seed or initialization. Such stability underscores the robustness and reliability of the proposed method.



(a) GPT-4.0 Turbo **(b) DeepSeek r1** **(c) Gemini 2.0 Flash**

Figure 4.13 : Action sequences generated by different FMs

4.4.6.3 FM AS THE PATHFINDER EXECUTOR

This section aims to enable the different FMs to complete the mission by providing the final action array without any RL intervention. To evaluate the impact of various FMs on mission string generation for our reinforcement learning agents, we assessed the effectiveness of different FMs. Specifically, we compared GPT-4.0 Turbo, DeepSeek r1, and Gemini 2.0 Flash. Figures 4.10–4.12 present the three generated action sequences collectively, while Figure 4.13 provides their umbrella comparison. Our evaluation focused on the models' ability to translate a visual task (a grid-based navigation problem with lava obstacles) into a sequence of actions an agent could execute. The prompt given to each model was identical: "You are the agent in this picture, represented by the red arrow. The agent starts in the top-left corner and must reach the goal in the top-right corner while avoiding any lava obstacles along the way. You can take the following actions: 0 turn left, 1 turn right, 2

move forward. Can you solve this and provide the final array of actions to take to get to the green goal square?" The results show that the models failed to achieve the goal based on the observation alone. GPT-4.0 Turbo generated a sequence [2, 1, 2, 0, 2, 2, 0, 2, 1, 2], DeepSeek r1 generated [2, 1, 2, 2, 2, 2, 0, 2, 2, 2], and Gemini 2.0 Flash offered [1, 2, 2, 2, 2, 2]. Moreover, these results confirm that the choice of FM significantly affects the action sequence, highlighting the need for careful model selection in RL mission-string generation. FMs have the potential to significantly enhance the capabilities of RL agents by providing them with a deep understanding of the world and the ability to reason about complex tasks. However, relying solely on these models without incorporating RL can be insufficient for solving complex problems.

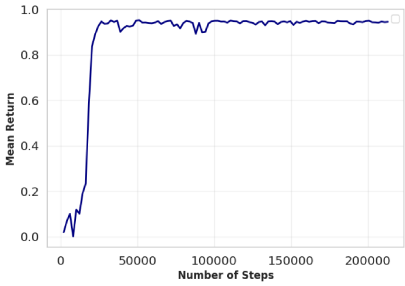
4.4.6.4 GEMINI AS THE AGENT GUIDE

We use the same prompt for all FMs, and Gemini provides the following mission statement: "Reach the green goal square while avoiding lava."

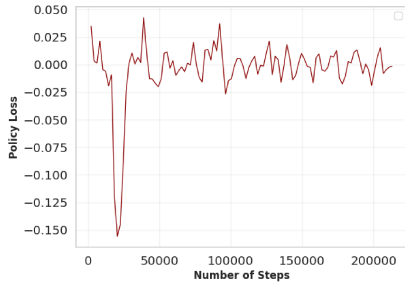
The results show the significant impact of DR on the Gemini model's ability to generalize. Without DR, "FMDM (Gemini)" performs well within the training distribution (DistShift2). The model shows a rapid initial increase in performance during the first 50,000 steps and then plateaus around a mean return of 0.9, indicating successful learning and stable performance. Figures 4.14–4.17 present the four Gemini training plots collectively, while Figure 4.18 provides their umbrella comparison.

Model: FMDM (Gemini)	**DistShift2 (Training) Mean return**	**DistShift1 (Testing) Mean return**
Without DR	0.9453 ± 0.2	0.31 ± 0.1
With DR	0.9013 ± 0.1	0.66 ± 0.1

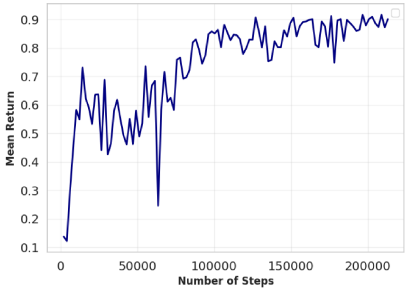
Table 4.4 : Results using Gemini as the guide



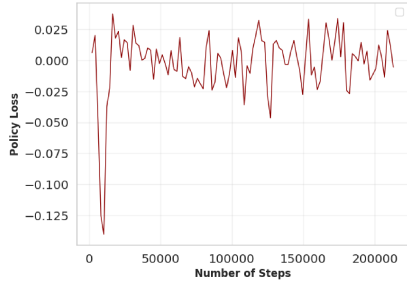
(a): Mean return without DR



(b): Policy loss without DR



(c): Mean return with DR



(d): Policy loss with DR

Figure 4.18 : Gemini as the FM

Overall, these results indicate that training this model with DR results in a training mean return of 0.9013 and a testing mean return of 0.66 (as shown in Table 4.4), highlighting Gemini's contribution to the agent's learning and generalization capabilities.

4.4.6.5 DEEPSEEK AS THE AGENT GUIDE

This subsection presents the results obtained using "FMDM" with DeepSeek as the FM. The results show a rapidly increasing mean return within the first 40,000 steps, quickly plateauing around 0.94, which suggests efficient learning and a stable policy. The policy loss fluctuates initially but stabilizes near zero after approximately 40,000 steps, indicating convergence to a consistent strategy. Although this model learns a high-performing policy and displays a strong mean return along with a stable learning process during the training phase, it achieves a mean return of 0.62 when tested in the DistShift1 environment (Table 4.5).

Figures 4.19–4.22 present the four DeepSeek training plots collectively, while Figure 4.23 provides their umbrella comparison.

Model: FMDM (DeepSeek)	**DistShift2 (Training) mean_return**	**DistShift1 (Testing) mean_return**
Without DR	0.9436 ± 0.2	0.62 ± 0.1
With DR	0.9214 ± 0.1	0.74 ± 0.1

Table 4.5 : Results of using DeepSeek as the agent guide

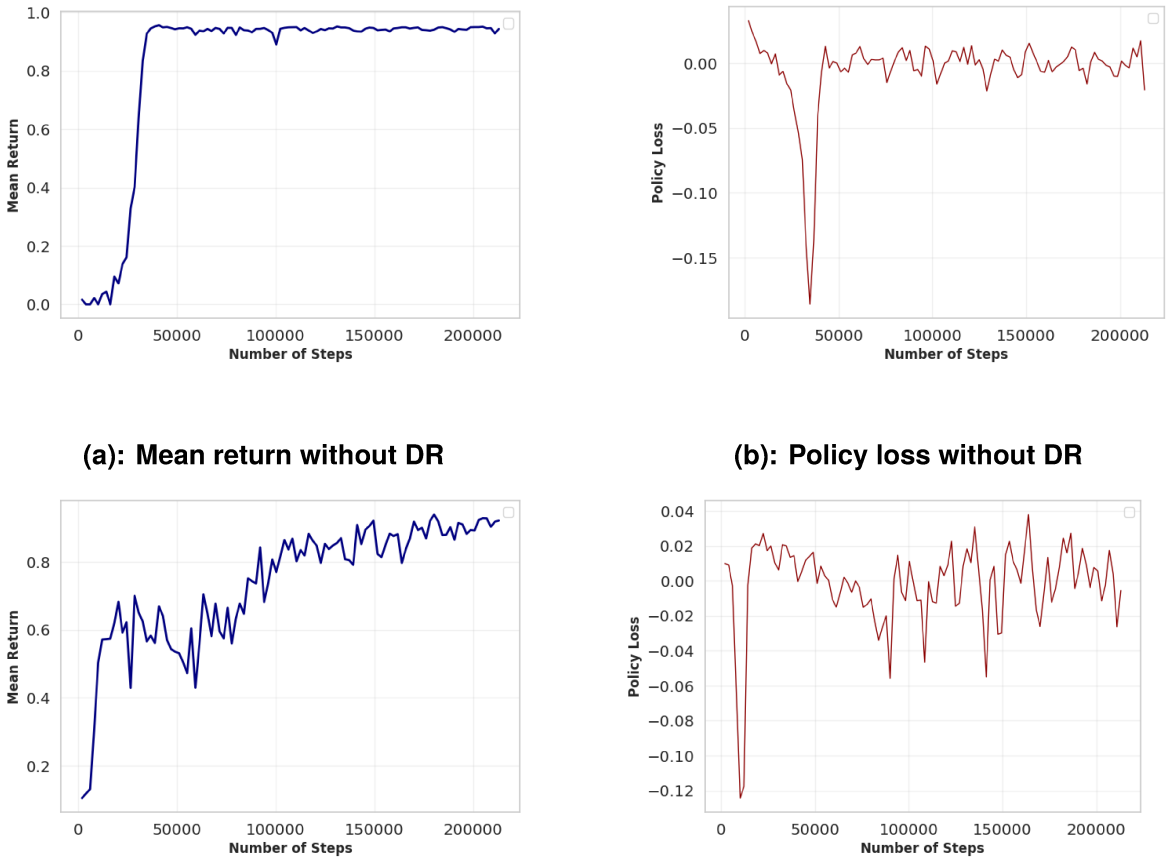
Furthermore, DeepSeek with DR achieves a training mean return of 0.9214 and a testing mean return of 0.74 (Table 4.5) when evaluated in the testing environment. This outcome highlights the impact of DR on the model's ability to generalize from the DistShift2 training environment to the DistShift1 testing environment.

4.4.6.6 EXPLORATORY ENHANCEMENT: INCORPORATING ATTENTION MECHANISM

Unlike prior work, our proposed approach leverages natural language instructions to guide policy learning by employing linguistic embeddings with visual observations through a spatial attention mechanism. First, the task description, referred to as the mission string (e.g., "Go to the green goal"), is encoded into a fixed-dimensional vector using a pretrained encoder. This mission embedding captures the semantic meaning of the instruction and is fed into the reinforcement learning agent alongside visual observations.

The visual input consists of grid-based environment states, which are processed using convolutional layers to produce spatial feature maps. The mission embedding is then fused with these visual features via an attention mechanism, enabling the agent to focus on task-relevant regions (e.g., the goal location, obstacles) rather than processing all visual information uniformly.

A pretrained language model encodes the mission string into a fixed-length embedding vec-



(a): Mean return without DR

(b): Policy loss without DR

(c): Mean return with DR

(d): Policy loss with DR

Figure 4.23 : DeepSeek as the guide

tor. Visual observations are processed by a convolutional encoder to create spatial feature maps, which are then modulated by the mission embedding through FiLM-based scaling and shifting. A 1×1 convolution followed by a spatial softmax produces an attention map highlighting important regions in the scene. These attention-weighted features are combined with the mission embedding and fed into the policy and value networks for action selection. The attention maps are visualized as heatmaps as shown in Figure 4.24, showing how mission semantics guide the agent's focus during decision-making. This approach differs from the traditional PPO baseline by incorporating fused features that guide both action selection and value estimation, rather than relying solely on basic observations. Additionally,

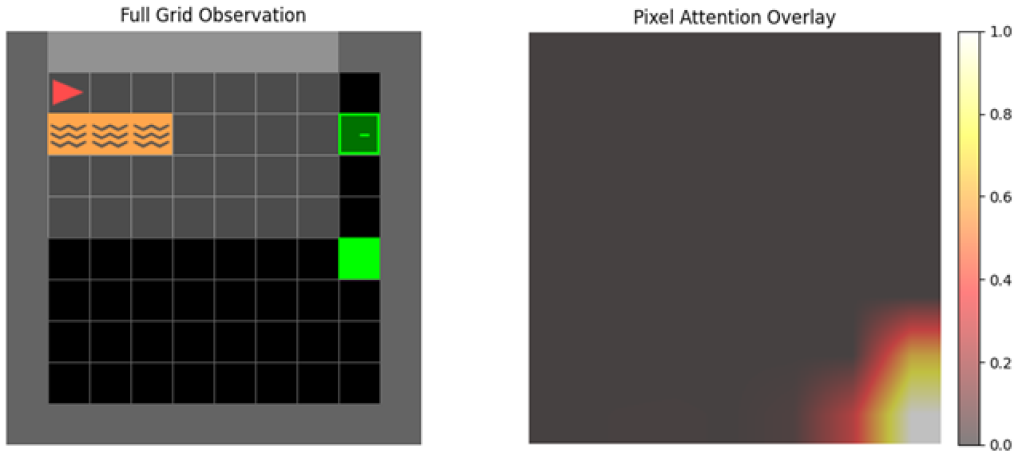


Figure 4.24 : Attention Heatmap showing the RL agent's focus on the goal at the bottom right corner

it utilizes spatial attention maps to provide clearer insights into the model's attention distribution throughout the environment. This enhanced capacity for reasoning over language and vision modalities leads to improved sample efficiency and interpretability in mission-driven reinforcement learning tasks.

4.4.7 DISCUSSION

FMs in RL frameworks enhance agent robustness by providing structured task representations, called mission strings. These mission strings serve as high-level guidance, encoding contextual information that improves an agent's ability to interpret and execute complex tasks across diverse environments. This approach improves the agent's ability to learn complex tasks and facilitates effective learning through a structured approach to achieving mission objectives.

The FM's ability to generate appropriate mission strings and appropriate context helped guide the agent implicitly, increasing its adaptability and effectiveness in different environments. Furthermore, the collaboration between the FM and the RL agent can lead to robust and scalable systems that effectively manage a wide range of situations and environments, including dynamic and unpredictable scenarios.

Additionally, the input was enhanced with extra contextual information, including environmental documentation and potentially multimodal data. This clear direction assisted the RL agent in better understanding the task at hand and making informed decisions. In comparison to the default mission strings, the generated strings offered a more nuanced and detailed representation of the environment that helped improve the agent's robustness.

Furthermore, we find that multimodal data and documentation are insufficient for an RL agent to generalize effectively despite containing contextual and nuanced information that could assist the agent. This is why data randomization significantly improves the agent's robustness. This enhancement likely occurs because the agent avoids fitting an overly specific model to the training data. Instead, it learns to adapt to a broader range of situations, supported by the FM that implicitly guides the RL process.

The comparison between "FMDM" and "PPO-DR" reveals the different characteristics of training efficiency and robustness. Although both algorithms exhibit the same learning behavior, the "PPO-DR" trained with DR exhibits a more variable learning curve. As this results in faster basic learning for "PPO-DR", possibly due to a broader search, "FMDM" achieves high and stable final mean return.

This implies a trade-off: DR can speed up initial learning and improve generalization to unseen changes in the environment, but it also leads to unstable training. In contrast, "FMDM" obtains a better mean return and smooth convergence, which can be more fundamental to the task and requires longer training time.

4.5 LIMITATIONS AND FUTURE WORK

While this chapter shows promising results, several limitations point the way for future research. The first study was conducted in a partially observable Markov Decision Process environment. Additional studies in more complex dynamic environments are essential to simulate real-world applications.

Second, the training process showed some improvement, with the original PPO method showing stable training despite weaker generalization compared to our approach. This implies

dealing with trade-offs between training progress and increased robustness: DR can provide initial and random learning to improve the robustness of the agent, but it may also result in unstable training. Conversely, our proposed approach, which implies DR, achieves better results and smoother convergence, which can capture more original task characteristics.

Finally, addressing the possibility of FM hallucinations and biases, which may adversely affect the agent's behavior, is an important area for further research and mitigation strategies. In future work, we will focus on multi-agent systems and the orchestration of multiple FMs. Additionally, we plan to integrate more models, such as "Florence" or "Pixtral 12B" [160, 161].

APPENDIX: FM SPECIFICATIONS AND REPRODUCIBILITY

MODEL ACCESS AND DECODING SETTINGS

To generate the mission strings utilized in Sections 4.3 and 4.4, we employed the following FMs via their official web-based chat interfaces: GPT-4.0 Turbo, Gemini 2.0, and DeepSeek-r1. As these models were accessed through their respective public web interfaces rather than API endpoints, the default decoding parameters (temperature, top-p, and maximum length) established by the providers at the time of access (January 2025) were utilized. We did not apply manual temperature scaling or custom decoding strategies. All FM weights were kept frozen during the process; no fine-tuning or gradient updates were applied to the language models.

MISSION STRING GENERATION AND REFRESH FREQUENCY

To consistently evaluate the influence of the distinct semantic phrasing generated by each FM on the RL agent's learning process, the mission strings were generated offline once for each FM and cached for the duration of the training. Consequently, the refresh frequency was set to zero; the agent received the same instruction string for every episode throughout the 200,000 training steps. This approach eliminates the confounding variable of stochastic generation variance during the RL training loop. The prompt utilized for all models was:

"From this document, generate the mission string for the RL agent. Keep it clear and short."

REPRESENTATIVE INSTRUCTION SAMPLES

The following fixed strings were generated and used for the comparative analysis:

- **Default (Baseline):** "get to the green goal square".
- **GPT-4.0:** "Reach the green goal square located at the top right while avoiding lava".
- **Gemini 2.0:** "Reach the green goal square while avoiding lava".
- **DeepSeek-r1:** "Navigate from the start to the goal while avoiding lava."

SEMANTIC DIVERSITY MEASURE

To quantify the linguistic variation, we utilized the `all-MiniLM-L6-v2` sentence transformer model [162]. We defined the diversity score as the mean squared Euclidean distance of each mission string's embedding (e_i) from the centroid (μ) of all generated embeddings:

$$\text{Diversity} = \frac{1}{N}\sum_{i=1}^{N} ||e_i - \mu||^2$$

where N represents the number of distinct mission strings evaluated.

4.6 CHAPTER SUMMARY

This chapter examined the robustness of reinforcement learning agents under natural distributional shift, proposing a framework that combines foundation-model guidance with domain randomization to improve generalizability. Foundation models parse environment documentation to generate task-specific mission objectives that enrich the reward signal, while domain randomization systematically varies physics parameters, textures, and environmental conditions during training. Relying solely on foundation-model guidance limits generalizability, but when combined with domain randomization, the approach consistently

improves RL agent performance across in-distribution and out-of-distribution environments, demonstrating that intelligent systems capable of reliable real-world performance can be developed without per-environment manual engineering.

Confirmation of Sub-Hypothesis 2. In the MiniGrid DistShift evaluation, the complete FMDM framework combined FM-generated mission guidance with domain randomization and achieved an average return of 0.85 on the shifted environment, compared with 0.74 for PPO-DR and 0.32 for PPO-Base. The component analysis further showed that FM guidance preserved learning stability while domain randomization supplied the environmental variation needed for transfer. These results confirm, within the evaluated DistShift1/DistShift2 setting, that integrating FMs with domain randomization improves RL-agent generalization across distributional shifts, as stated in Sub-Hypothesis 2. This evidence supports the model-reliability dimension of the main hypothesis in Section 1.5, because the combined mechanism helps preserve task performance when deployment conditions differ from training conditions.

A qualitatively different and more severe form of distributional shift arises when an intelligent adversary *deliberately* crafts perturbations designed to exploit the agent's learned policy. Unlike natural drift, adversarial perturbations are optimized to maximally degrade performance while remaining imperceptible or physically plausible. This adversarial threat is particularly acute in safety-critical systems such as autonomous driving, where a single incorrect decision can result in physical harm. Chapter 5 investigates this adversarial threat in the autonomous driving domain: it characterizes how RL-based driving agents can be compromised via both observation-space and action-space attacks, and evaluates lightweight inference-time defense mechanisms that wrap already-trained and deployed policies without requiring any model retraining.

CHAPTER 5

ADVERSARIAL ROBUSTNESS OF REINFORCEMENT LEARNING IN AUTONOMOUS DRIVING: VULNERABILITY ANALYSIS AND INFERENCE-TIME DEFENSES

5.1 CHAPTER OVERVIEW

The transition of Autonomous Vehicles (AVs) from highly constrained operational design domains to dynamic, unstructured real-world environments requires decision-making systems capable of handling immense uncertainty. Deep Reinforcement Learning (DRL) has emerged as a highly effective paradigm for this challenge, allowing vehicles to learn complex driving policies through trial-and-error interaction within simulated environments [163, 164, 165, 166]. By framing the driving task as a Markov Decision Process (MDP), DRL agents such as Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) have been widely used for lane keeping, trajectory planning, and navigation tasks.

Despite these advancements, the translation of DRL from simulation to physical cyber-physical systems (CPS) is impeded by a critical flaw: the documented brittleness of neural network policies to adversarial attacks [167, 168, 169]. Unlike traditional software vulnerabilities, adversarial attacks on DRL exploit the high-dimensional mathematical landscape of the neural network itself. An adversary capable of injecting minute perturbations into the observation space (e.g., via LiDAR spoofing or camera blinding) or intercepting the action space (e.g., via Controller Area Network (CAN) bus tampering) can force an otherwise highly optimized policy into catastrophic, unsafe states [170, 171].

The prevailing approach to securing DRL models relies heavily on adversarial retraining, wherein agents are continuously exposed to perturbations during the training phase to encourage the learning of robust representations [172, 168, 169, 173]. While effective in static or highly constrained domains, adversarial retraining in continuous autonomous driv-

ing presents two major limitations. First, it is computationally expensive, typically requiring substantially more environment steps to converge. Second, it frequently incurs a "robustness tax": a degradation in the agent's baseline performance and reward acquisition in benign, clean-state environments.

Furthermore, attempts to secure the observation space using traditional signal processing techniques, such as temporal smoothing or moving averages, can overlook the temporal requirements of high-speed closed-loop control. In autonomous driving, the temporal alignment of state-action pairs is critical; mitigating noise at the expense of introducing phase lag can actively sabotage the agent's control loop.

We evaluate lightweight inference-time defenses as an alternative to robust retraining and state-space smoothing. The evaluated wrappers treat the DRL policy as fixed and monitor the state-action stream to gate malicious commands before physical actuation. We focus on HighwayEnv [174] with DQN and PPO agents and quantify how observation attacks (noise, spoofing), action attacks (sustained fixed-command injection, epsilon-malicious tampering), and composite (multi-vector) perturbations affect safety and reward under controlled conditions.

In this chapter, adversarial robustness denotes the ability of a frozen driving policy, together with its inference-time defense, to preserve safe behavior and useful task performance under the specified perturbations. We therefore assess robustness jointly through collision rate and clean-baseline-normalized reward retention; override rate and latency characterize the intervention cost. This definition is deliberately limited to the evaluated threat models, policy architectures, and simulators and does not imply certified or universal robustness.

We hypothesize that the effectiveness of an inference-time defense depends on the architecture of the deployed policy as well as on the attack configuration. In our experiments, for the discrete, value-based DQN agent, value-landscape gating (QGap) was the most effective defense under intermittent action tampering, a setting in which its Q-value estimates are informative. For the discrete actor-critic PPO agent, DynIW (observation gating with

policy-referenced action validation) was more effective. Standard PPO exposes no action-conditioned critic, so QGap cannot be applied natively; a hard-label proxy (using the actor's greedy output as a stand-in) was evaluated as a negative-control ablation and did not improve the tested PPO checkpoint. Because per-step diagnostics were not collected, the experiment cannot determine whether this failure is intrinsic to label-based gating or specific to the implemented proxy, and it is not used as evidence of a general limitation. For the continuous-action PPO checkpoint, DynIW's continuous-action branch, which uses the nominal action of the protected policy as its reference, substantially reduced collisions under fixed-command injection, whereas QGap has no viable implementation, since state-value-only critics expose no action-conditioned landscape to enumerate or rank. These observations rest on a single training seed per HighwayEnv agent, two policy families, and one primary environment.

The primary contribution is not a new defense algorithm in isolation, but the empirical evidence for, and a formulation of, this coupling as a hypothesis for deployment-time defense selection.

This chapter makes four primary contributions in support of this hypothesis.

- An empirical evaluation of two inference-time defense wrappers (QGap and DynIW), with EMA smoothing analyzed as a baseline, on HighwayEnv using both DQN and PPO agents, spanning observation, action, and composite attack categories, to establish the discrete-action evidence base for the coupling hypothesis.

- Empirical evidence consistent with an architecture-defense coupling: value-landscape gating (QGap) was the more effective defense for the value-based DQN agent under intermittent action tampering, while DynIW (observation gating with policy-referenced action validation) better recovered the discrete actor-critic PPO agent, together with a description of the conditions under which each wrapper failed.

- A cross-environment study using a frozen, externally pre-trained continuous-control PPO checkpoint in MetaDrive, in which DynIW's continuous-action branch (a policy-

referenced action check with an L2 threshold) substantially reduced collisions for the evaluated checkpoint under fixed-command injection without retraining, while native QGap has no viable implementation for continuous-action state-value-only critics.

- Per-scenario quantitative results reported through reward, collision-rate, reward-retention, and override-rate metrics, together with defense latency measurements and practical deployment recommendations for each identified architecture class.

The remainder of this chapter is structured as follows. Section 5.2 introduces the reinforcement learning preliminaries and notation. Section 5.3 surveys related work on adversarial attacks, adversarial training, and inference-time safe-RL approaches. Section 5.4 formalizes the threat model covering both observation-space and action-space adversaries, and Section 5.5 describes the inference-time defense mechanisms considered in the study. Section 5.6 details the experimental setup and evaluation methodology, and Section 5.7 reports the results across five findings for HighwayEnv and MetaDrive. Section 5.8 discusses deployment implications and latency, and Section 5.9 addresses threats to validity. Sections 5.10, 5.11, and 5.12 provide HighwayEnv training curves, additional results, and details of the external MetaDrive policy. Finally, Section 5.13 provides the chapter summary and conclusions.

5.2 REINFORCEMENT LEARNING PRELIMINARIES

5.2.1 STATE SPACE $\mathcal{S}$

At each time step t, the agent receives an observation $s_t \in \mathcal{S}$. The state vector comprises ego-vehicle kinematics and simplified exteroceptive information about the surrounding environment. The continuous state space is formalized as:

$$s_t = [v_x, v_y, x, y, \psi, o_1, o_2, \ldots, o_n]^T \in \mathbb{R}^k$$

where v_x, v_y denote the longitudinal and lateral velocities, (x, y) represent the global coordinates, ψ is the heading angle, and o_i represents the relative velocity and positional coordinates of the i-th surrounding obstacle or vehicle.

5.2.2 ACTION SPACE $\mathcal{A}$

The agent outputs an action $a_t \in \mathcal{A}$ based on its policy $\pi_\theta(a_t|s_t)$, parameterized by a neural network with weights θ. The action space varies depending on the specific control architecture.

5.2.3 TRANSITION DYNAMICS $\mathcal{P}$

The environment evolves according to a transition probability distribution $\mathcal{P}(s_{t+1}|s_t, a_t)$. In autonomous driving, $\mathcal{P}$ is governed by highly non-linear vehicle kinematics (e.g., the kinematic bicycle model) and the stochastic behavior of surrounding traffic agents. Because $\mathcal{P}$ is generally unknown to the agent, the evaluated policies use model-free DRL algorithms.

5.2.4 REWARD FUNCTION $\mathcal{R}$

The objective of the autonomous-driving agent is to maximize the expected cumulative discounted return $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $\gamma \in [0, 1)$ is the discount factor. The step reward $r_t = \mathcal{R}(s_t, a_t)$ is structured as a multi-objective composite function that balances progress, comfort, and safety:

$$r_t = \omega_1 \cdot v_{x,t} - \omega_2 \cdot |\delta_t| - \omega_3 \cdot \mathbb{I}_{collision}(s_t)$$

where ω_i are weighting coefficients, $v_{x,t}$ encourages forward progress, $|\delta_t|$ penalizes erratic steering (comfort), and $\mathbb{I}_{collision}$ is a heavily weighted boolean penalty triggered upon collision, terminating the episode.

5.2.5 DEEP RL ALGORITHMS (DQN AND PPO)

The experiments evaluate DQN and PPO agents. DQN provides a value-based baseline for discrete control, while PPO provides an on-policy actor-critic baseline.

PPO [175] improves upon traditional actor-critic methods [176] by preventing destructively large policy updates. It achieves this by maximizing a clipped surrogate objective

function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]$$

where $r_t(\theta)$ is the probability ratio between the updated and previous policy, and ϵ controls the trust-region approximation. The experiments below suggest that PPO can remain vulnerable to adversarial perturbations during inference despite its stable training objective.

5.3 RELATED WORK

5.3.1 ADVERSARIAL ATTACKS ON REINFORCEMENT LEARNING

The susceptibility of deep neural networks to adversarial examples was first formalized by Szegedy et al. [177] in the domain of computer vision. This vulnerability was later mapped to the DRL domain by Huang et al. [167], who showed that applying Fast Gradient Sign Method (FGSM) perturbations to the state observation vector could drastically degrade the performance of policies trained on Atari games. Recent robust DRL studies further show that state-observation perturbations remain difficult to defend against even when robustness is explicitly optimized during policy learning [168, 169, 173]. Subsequent work has developed stronger learned evasion attacks [178] and certifiably robust policy training [179].

In the context of autonomous driving, state-space attacks emulate physical sensor spoofing. Behzadan and Munir [180] showed that injecting targeted noise into DQN-based navigation agents could induce systematic collisions. However, these works predominantly focus on continuous state-space perturbations. This study extends the threat landscape by evaluating *discrete action-space tampering* (simulating a compromised CAN-bus), which bypasses any noise tolerance that the policy's learned representations may provide.

5.3.2 THE "ROBUSTNESS TAX" OF ADVERSARIAL TRAINING

The most prevalent defense mechanism in the literature is Adversarial Training (AT), which frames the learning process as a zero-sum minimax game between the agent and the adversary [181]. Pattanaik et al. [172] proposed robust DRL by actively perturbing the

agent's observations during the training phase, forcing the network to learn invariant feature representations.

While theoretically sound, AT faces severe limitations in real-world CPS deployment. Firstly, it typically requires many more training samples, increasing the time and computational cost of training. Secondly, as shown by Tsipras et al. [182], there exists an inherent trade-off between robustness and standard accuracy. In autonomous driving, this "robustness tax" manifests as overly conservative driving policies that achieve safety at the cost of failing to reach the destination efficiently. Recent safe-RL surveys similarly emphasize that robustness and constraint satisfaction often depend on environment-specific assumptions, monitoring signals, and intervention budgets [183, 184]. Runtime assurance and safety-filtering approaches add a monitor or filter that overrides a learned controller near unsafe states without retraining it [185], and reachability-based safety layers have been combined with RL policies in the same spirit [186]. This limitation necessitates the exploration of post-training, inference-time defenses.

5.3.3 ROBUST REINFORCEMENT LEARNING AND EVALUATION PRACTICE

Robust reinforcement learning addresses uncertainty through adversarial training, policy regularization, conservative smoothing, and distributionally robust objectives. Langevin-based adversarial training improves generalization across changing conditions, while conservative smoothing improves robustness to observation deviations in offline reinforcement learning [187, 188]. Adversarial regularization has also been extended to multi-agent settings with robustness objectives covering noisy observations, changed dynamics, and malicious actions [189]. These approaches modify policy learning, whereas QGap and DynIW protect already-trained policies at inference time.

Reliable comparison also requires careful experimental methodology. Deep reinforcement learning results can vary with implementation, hyperparameters, environment stochasticity, and random seeds [190]. Standardized evaluation protocols and explicit treatment of stochasticity are therefore important when interpreting robustness claims [191].

5.3.4 INFERENCE-TIME INTERVENTIONS AND SAFE RL

To avoid the cost of retraining, recent literature has explored inference-time interventions. Alshiekh et al. [192] introduced the concept of "Shielding," wherein a deterministically synthesized safety logic overrides the RL policy if an action leads to an unsafe state. Runtime assurance architectures such as Simplex similarly pair a complex controller with a simpler verified fallback that takes over near unsafe states [193]. Predictive safety layers such as short-horizon Model Predictive Control (MPC) [194] and Control Barrier Function (CBF) filters [195] go further by rolling out a dynamics model to reject commands before the vehicle reaches an unrecoverable state, at the cost of requiring an ego and neighbour-vehicle model; they are not evaluated here (see Finding 3 and Section 5.9). Broader safe-learning work in robotics also motivates runtime monitors and intervention layers when learned controllers are deployed under physical safety constraints [196, 197].

The proposed mechanisms, QGap and DynIW, build upon the philosophy of Safe RL but diverge in execution. Rather than relying on computationally heavy formal verification or rule-based logic, these defenses leverage value-based action assessment (QGap) and localized consistency checks (DynIW) to gate suspicious actions. Standard signal-processing defenses such as exponential smoothing can still be useful in some settings, but they should be validated carefully in closed-loop driving systems. Closed-loop autonomous-driving benchmarks have independently highlighted the importance of collision-targeted evaluation under rare but safety-critical scenarios [198, 199]. Foundation models for robotic control further motivate inference-time monitors as a safety complement to pre-training, since large learned policies offer limited formal safety guarantees at deployment [200].

5.4 THREAT MODEL FORMALIZATION

To systematically evaluate the robustness of the autonomous-driving agents, the threat model encompasses both observation-space and action-space interventions. A bounded adversary is assumed to have real-time access to the agent's data streams, but no white-box access to the neural network weights θ. This evaluation philosophy is aligned with recent

safety-critical scenario generation and benchmark work, which stresses that AV policies must be tested under rare but consequential disturbances rather than only under average-case traffic distributions [171, 166].

5.4.1 OBSERVATION-SPACE VULNERABILITIES

The observation-space attack simulates a compromised sensor suite (e.g., LiDAR/Radar jamming or camera noise). Let s_t be the ground-truth physical state of the environment. The adversary injects a perturbation vector η_t to produce a corrupted observation $\tilde{s}_t$:

$$\tilde{s}_t = s_t + \eta_t, \quad \text{s.t.} \quad ||\eta_t||_\infty \leq \epsilon_{obs}$$

where ϵ_{obs} dictates the maximum perturbation magnitude. In the *Noise (NoDef)* scenario, η_t is sampled from a zero-mean Gaussian distribution $\mathcal{N}(0, \sigma^2)$, testing the agent's inherent baseline robustness to stochastic sensor variance.

5.4.2 ACTION-SPACE VULNERABILITIES (ACTUATION TAMPERING)

The action-space attack represents a critical breach of the vehicle's internal Controller Area Network (CAN) bus, allowing the adversary to intercept the agent's intended action $a_t = \pi_\theta(s_t)$ and substitute it with a malicious command $\tilde{a}_t$. The sustained fixed-command injection attack replaces the policy output at every attacked step with a predefined disruptive action a_{fixed}:

$$\tilde{a}_t = a_{\text{fixed}}.$$

This implementation does not compute a state-dependent worst-value action and should not be interpreted as an $\arg\min_a Q^\pi(s_t, a)$ attack. It represents a deterministic actuation-channel compromise in which the attacker repeatedly injects the same command. Unlike observation noise, which the agent's policy may partially absorb via learned internal representations, action tampering bypasses the policy output and directly corrupts the physical transition $\mathcal{P}(s_{t+1}|s_t, \tilde{a}_t)$.

5.4.3 ATTACK LOCATION AND TRUSTED SIGNALS

The action attacker is positioned after policy inference and before command execution. The policy parameters and the defense monitor remain trusted. QGap uses the DQN action values for the current state, whereas DynIW uses its filtered observation state and a reference derived from the protected policy itself: the action values for DQN, or the policy's own action on the gated observation for PPO. This reference presumes that the wrapper can recompute the nominal action from the observation it holds and that tampering occurs downstream of the policy. Observation corruption can degrade both the policy output and the observation gate on which these references depend, motivating the separate composite-attack analysis.

5.5 INFERENCE-TIME DEFENSE MECHANISMS

To counter the defined threat model without incurring the computational overhead of adversarial retraining, this section formalizes three inference-time interventions for the HighwayEnv study: one observation-side smoother (EMA), one action-space gate (QGap), and one wrapper that combines observation gating with action validation (DynIW), whose continuous-action branch is also used in the MetaDrive extension. The evaluation distinguishes between state-space smoothing and action-space gating. Closed-loop results are reported for QGap and DynIW; EMA was evaluated only analytically and for latency (Table 5.5); no closed-loop outcome experiments were run with EMA.

5.5.1 THE "LATENCY TRAP" OF STATE SMOOTHING (EMA)

Traditional control systems frequently employ low-pass filters to mitigate sensor noise. The evaluation includes the Exponential Moving Average (EMA) applied to the incoming observation stream:

$$\hat{s}_t = \alpha \tilde{s}_t + (1 - \alpha)\hat{s}_{t-1}$$

where $\alpha \in (0, 1]$ is the smoothing factor. EMA can introduce phase lag into the observation stream. In a dynamic autonomous-driving scenario, the state changes proportionally to the vehicle's velocity v_t. The temporal lag error E_{lag} introduced by the EMA filter can be

approximated as:

$$E_{lag} = s_t - \hat{s}_t \approx \left(\frac{1-\alpha}{\alpha}\right)\Delta s_t \approx \left(\frac{1-\alpha}{\alpha}\right) v_t \Delta t$$

As v_t increases, the divergence between the perceived state $\hat{s}_t$ and the true kinematic state s_t grows linearly. In high-speed obstacle avoidance, this delayed observation can cause the policy to execute maneuvers based on outdated spatial coordinates, which is why temporal smoothing must be treated cautiously in dynamic control loops. This is an analytical approximation; its closed-loop effect is not measured in this chapter.

5.5.2 Q-VALUE GAP THRESHOLDING (QGAP)

To avoid the state-space latency trap, QGap is defined as an action-space gating mechanism that leverages the agent's own action-conditioned value function $Q(s, a)$. This mechanism applies directly to value-based architectures such as DQN, where the critic maintains an explicit $Q(s, a)$ for every discrete action. For actor-critic architectures that expose only a state-value $V(s)$ (such as standard PPO), no action-conditioned landscape exists and QGap cannot be applied without approximation; this limitation is discussed in Finding 3 (Section 5.7).

Before actuating the proposed command $\tilde{a}_t$ (which may be tampered with), the QGap wrapper computes the advantage gap between the optimal perceived action a^* and the incoming command $\tilde{a}_t$:

$$\Delta Q_t = Q_\phi(s_t, a^*) - Q_\phi(s_t, \tilde{a}_t)$$

where Q_ϕ is the frozen, pre-trained critic network, and $a^* = \arg\max_a Q_\phi(s_t, a)$. If the discrepancy exceeds a safety threshold τ_{gap} (fixed at $\tau_{\text{gap}} = 0.05$ in all experiments), the wrapper rejects $\tilde{a}_t$ and triggers a safety fallback:

$$a_{execute} = \begin{cases} \tilde{a}_t & \text{if } \Delta Q_t \leq \tau_{\text{gap}} \\ a_{safe} & \text{if } \Delta Q_t > \tau_{\text{gap}} \end{cases}$$

Input: state s_t, command $\tilde{a}_t$, value function Q_ϕ, threshold τ_{gap}, fallback a_{safe}
Output: executed action $a_{execute}$

$a^* \leftarrow \arg\max_a Q_\phi(s_t, a)$;
$\Delta Q_t \leftarrow Q_\phi(s_t, a^*) - Q_\phi(s_t, \tilde{a}_t)$;
if $\Delta Q_t > \tau_{gap}$ **then**
    $a_{execute} \leftarrow a_{safe}$;
else
    $a_{execute} \leftarrow \tilde{a}_t$;
end
return $a_{execute}$;

Algorithm 5.1: QGap inference-time action gate

By evaluating the downstream consequence of the action in the latent space rather than filtering the physical observation, QGap intervenes at the current step and does not introduce history-dependent observation smoothing. Algorithm 5.1 summarizes the QGap action-gating procedure.

5.5.3 DYNIW: DYNAMIC OBSERVATION GATING WITH POLICY-REFERENCED ACTION VALIDATION

DynIW treats the protected policy as frozen and acts at every control step in two stages: a dynamic observation gate, followed by an action-validation check whose reference is the protected policy itself. The wrapper holds no history beyond the filter states described below.

Stage 1: dynamic observation gating. DynIW maintains a smoothed observation $\bar{s}_{t-1}$ and a per-dimension Kalman estimate $\mathrm{KF}(\cdot)$ with process variance $Q = 0.5$ and measurement variance $R = 0.1$, which is updated with every incoming observation. An observation $\tilde{s}_t$ is flagged as a spike if any of its dimensions deviates from $\bar{s}_{t-1}$ by more than τ_{spike}. Flagged

observations are replaced by the Kalman estimate; all others are exponentially smoothed:

$$\hat{s}_t = \begin{cases} \mathrm{KF}(\tilde{s}_t) & \text{if } \max_j |\tilde{s}_{t,j} - \bar{s}_{t-1,j}| > \tau_{\text{spike}} \\ \alpha\, \tilde{s}_t + (1-\alpha)\, \bar{s}_{t-1} & \text{otherwise,} \end{cases} \qquad \bar{s}_t = \hat{s}_t,$$

with $\alpha = 0.7$ and $\tau_{\text{spike}} = 0.50$. The first observation of an episode is passed through unchanged. The gated observation $\hat{s}_t$ is the input to the policy and to the action validator below. DynIW therefore contains an exponential-smoothing component, but it is not the standalone EMA baseline analyzed above, whose closed-loop effect was not evaluated.

Stage 2: action validation. Given the possibly tampered command $\tilde{a}_t$, DynIW decides whether to override it, using the strongest reference the protected policy exposes:

- **Value-based policies (DQN).** With the frozen action-value network Q_ϕ, the command is replaced by the greedy action $a^* = \arg\max_a Q_\phi(\hat{s}_t, a)$ when $\max_a Q_\phi(\hat{s}_t, a) > 0$ and $Q_\phi(\hat{s}_t, \tilde{a}_t) < \tau_{\text{idw}} \max_a Q_\phi(\hat{s}_t, a)$, with $\tau_{\text{idw}} = 0.50$. The test is relative, whereas the QGap test uses an absolute gap.

- **Discrete policies without a Q-network (HighwayEnv PPO).** The command is replaced by the policy's deterministic action $\pi_\theta(\hat{s}_t)$ whenever the two differ.

- **Continuous policies (MetaDrive PPO).** The command is replaced by $\pi_\theta(\hat{s}_t)$ when the two disagree by at least $\tau_{\text{cont}} = 0.30$ in Euclidean distance:

$$a_{execute} = \begin{cases} \tilde{a}_t & \text{if } \|\tilde{a}_t - \pi_\theta(\hat{s}_t)\|_2 < \tau_{\text{cont}} \\ \pi_\theta(\hat{s}_t) & \text{otherwise.} \end{cases}$$

All thresholds and filter parameters were held fixed across attacks. The override rate reported in the tables counts the steps on which Stage 2 replaced the command. Algorithm 5.2 summarizes the two-stage DynIW procedure.

Input: observation $\tilde{s}_t$, command $\tilde{a}_t$, previous gated observation $\bar{s}_{t-1}$, protected policy π_θ (and Q_ϕ for DQN), smoothing factor α, thresholds $\tau_{spike}, \tau_{idw}, \tau_{cont}$
Output: executed action $a_{execute}$

update the Kalman estimate $\mathrm{KF}(\tilde{s}_t)$ with the incoming observation;
// Stage 1: dynamic observation gating
if *t is the first step of the episode* **then**
  $\hat{s}_t \leftarrow \tilde{s}_t$;
else
  if $\max_j |\tilde{s}_{t,j} - \bar{s}_{t-1,j}| > \tau_{spike}$ **then** $\hat{s}_t \leftarrow \mathrm{KF}(\tilde{s}_t)$ **else** $\hat{s}_t \leftarrow \alpha\, \tilde{s}_t + (1-\alpha)\, \bar{s}_{t-1}$;
end
$\bar{s}_t \leftarrow \hat{s}_t$;
// Stage 2: action validation against the protected policy
if *value-based policy (DQN)* **then**
  $a^* \leftarrow \arg\max_a Q_\phi(\hat{s}_t, a)$;
  if $Q_\phi(\hat{s}_t, a^*) > 0$ ***and*** $Q_\phi(\hat{s}_t, \tilde{a}_t) < \tau_{idw}\, Q_\phi(\hat{s}_t, a^*)$ **then** $a_{execute} \leftarrow a^*$ **else** $a_{execute} \leftarrow \tilde{a}_t$;
else if *discrete policy without a Q-network (PPO)* **then**
  if $\tilde{a}_t \neq \pi_\theta(\hat{s}_t)$ **then** $a_{execute} \leftarrow \pi_\theta(\hat{s}_t)$ **else** $a_{execute} \leftarrow \tilde{a}_t$;
else
  if $\|\tilde{a}_t - \pi_\theta(\hat{s}_t)\|_2 \geq \tau_{cont}$ **then** $a_{execute} \leftarrow \pi_\theta(\hat{s}_t)$ **else** $a_{execute} \leftarrow \tilde{a}_t$;
end
return $a_{execute}$;

Algorithm 5.2: DynIW: dynamic observation gating with policy-referenced action validation

For policies without a Q-network, the reference in Stage 2 is the nominal action of the protected policy. This presumes that the wrapper can recompute that action from the observation it holds and that tampering occurs downstream of the policy, as in the actuation attacks evaluated here. Under this assumption the check is closer to an oracle against actuation-only tampering than to a detector that must infer the nominal behavior. It offers no protection against errors made by the policy itself, and the observation gate offers only limited protection against observation-space attacks.

5.6 EXPERIMENTAL SETUP AND METHODOLOGY

To evaluate the vulnerability of DRL agents and the efficacy of inference-time wrappers, the study uses 100-episode robustness trials under clean, attacked, and defended conditions.

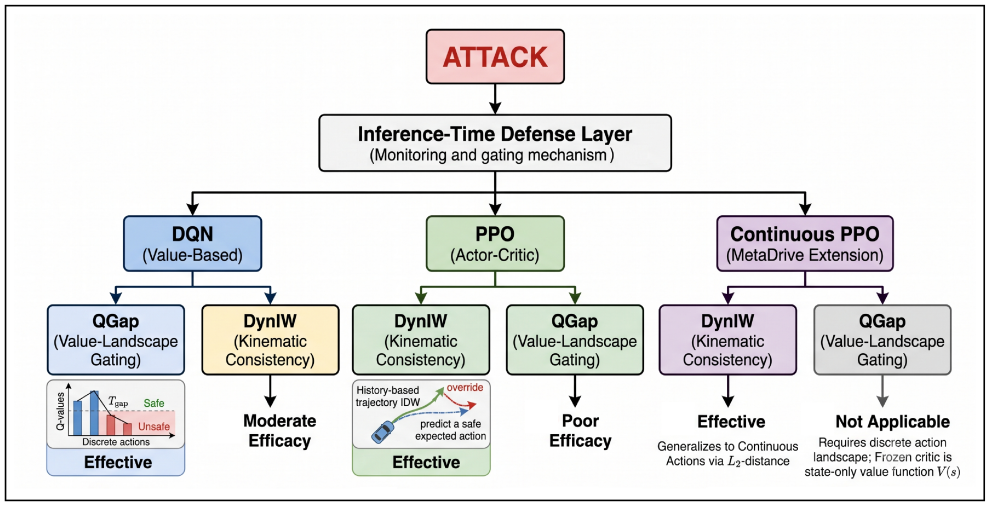


Figure 5.1 : Overview of the proposed inference-time defense evaluation pipeline. The frozen DRL policy (DQN or PPO) is exposed to observation-space and action-space attacks; the resulting action stream is passed through a candidate inference-time wrapper (QGap or DynIW) that gates or overrides suspicious commands before they reach the environment, and the closed-loop outcome is scored with the reward, collision-rate, reward-retention ratio, and override-rate metrics defined later in this section.

HighwayEnv is used as the primary stable-policy benchmark because both DQN and PPO clean runs achieve 0.000 collision rate.

A fixed budget of 100 episodes per scenario was selected as a computationally feasible evaluation size and applied uniformly across all conditions. The QGap threshold ($\tau_{\text{gap}} = 0.05$) and the DynIW thresholds ($\tau_{\text{idw}} = \tau_{\text{spike}} = 0.50$) were fixed from pilot runs in the same HighwayEnv environment used for evaluation (Section 5.9); no held-out threshold selection was performed.

5.6.1 SIMULATION ENVIRONMENTS

The experimental implementation supports multiple driving topologies. The primary results analyzed in this chapter focus on HighwayEnv, which supports the main stable-policy robustness claims for both DQN and PPO agents.

HighwayEnv (Stable-Policy Benchmark): HighwayEnv provides the cleaner benchmark for evaluating inference-time defenses under well-converged nominal policies. In the 100-episode clean evaluation, DQN obtains 34.71 reward with 0.000 collision rate, while PPO obtains 30.71 reward with 0.000 collision rate. These stable clean baselines make HighwayEnv the primary environment for claims about architecture-defense coupling.

5.6.2 MODEL ARCHITECTURES AND BASELINE TRAINING

The evaluation includes two trained DRL architectures: DQN and PPO. DQN provides the value-based baseline used to test QGap, while PPO provides the actor-critic baseline used to test DynIW's observation gating and policy-referenced action validation. Each model was trained for 200,000 timesteps. Both agents used MLP-based policies with two hidden layers of 256 units. Algorithm-specific activation and initialization settings followed the Stable-Baselines3 configurations used in our implementation.

The clean evaluations provide the nominal reference points for the later robustness analysis. HighwayEnv clean policies are stable and collision-free.

5.6.3 ATTACK AND DEFENSE EVALUATION PIPELINE

Following the freezing of the trained policies, the agents were subjected to inference-time evaluations across clean operation and three attack categories, always evaluated and reported in the same fixed order throughout this chapter: **Observation Attacks** (Noise, Spoofing), **Action Attacks** (Sustained Fixed-Command Injection, Epsilon-Malicious), and **Multi-Vector Attacks** (Composite). These attacks were evaluated with no defense and with the QGap and DynIW wrappers where applicable; in the MetaDrive extension only DynIW's continuous-action branch (Section 5.5.3) was evaluated. EMA was evaluated only analytically (Section 5.5) and for latency (Table 5.5); no closed-loop outcome experiments were run with EMA, so none are reported in this chapter. Each scenario is implemented as follows.

Observation Attacks:

- **Noise:** zero-mean Gaussian perturbation added to every observation dimension, $\tilde{s}_t = s_t + \mathcal{N}(0, \sigma^2)$ with $\sigma = 0.05$, emulating stochastic sensor variance.

- **Spoofing:** with probability $p = 0.5$ per step, roughly one quarter of the observation vector's dimensions are overwritten with values drawn from $\mathcal{U}(-1, 1)$, emulating forged or injected obstacle readings.

Action Attacks:

- **Sustained Fixed-Command Injection:** the agent's proposed discrete action is unconditionally replaced, at every step, by a predefined disruptive action index a_{fixed}, emulating a deterministic, sustained CAN-bus compromise. The command is state-independent and is not an $\arg\min_a Q^\pi(s_t, a)$ action (Section 5.4.2). In the tables this scenario is abbreviated "Fixed-Cmd".

- **Epsilon-Malicious:** with probability $\epsilon = 0.15$ per step, the proposed action is replaced by a uniformly random action sampled from $\mathcal{A}$; otherwise the clean action passes through unmodified, emulating an intermittent actuation fault.

Multi-Vector Attacks:

- **Composite:** the noise, spoofing, and action-tampering attacks are chained together at elevated intensity ($\sigma = 0.25$, $p = 0.5$, and a 20% per-step action-tampering probability); the action-tampering component uses the epsilon-malicious variant (random action substitution at each triggered step), not the deterministic fixed-command variant, so the composite scenario stresses layered defenses under concurrent observation corruption and intermittent actuation noise rather than sustained fixed-command injection.

5.6.4 EVALUATION METRICS

To quantify both the destructive capability of the adversary and the restorative capacity of the defenses, four core metrics are defined:

Episodic Reward ($\mathcal{R}_{total}$): The cumulative sum of step-wise rewards obtained during an episode. A drastic drop in $\mathcal{R}_{total}$ indicates successful adversarial interference.

Collision Rate (C_{rate}): The primary safety metric, defined as the ratio of episodes terminating in a collision to the total number of episodes N:

$$C_{rate} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{collision}(i)$$

where $\mathbb{I}_{collision}(i)$ is an indicator function returning 1 if episode i ends in a crash.

Reward-Retention Ratio ($\mathcal{RR}$): To avoid ambiguity, $\mathcal{RR}$ is defined in this chapter as a clean-baseline-normalized reward-retention ratio, not as an attack-specific recovery fraction. Let $\mathcal{R}_{clean}$ be the mean reward under clean operation for the same environment-agent pair, and let $\mathcal{R}_x$ be the mean reward for any evaluated scenario x.

$$\mathcal{RR}_x = \frac{\mathcal{R}_x}{\mathcal{R}_{clean}}$$

Thus, the clean baseline has $\mathcal{RR} = 1.000$ by definition. Values below 1.0 indicate reward degradation relative to clean operation. Values above 1.0 indicate reward inflation, not necessarily successful recovery or safety; they must be interpreted together with collision rate.

Override Rate (O_{rate}): For inference wrappers (QGap, DynIW), the override rate measures the defense's efficiency, specifically how often the wrapper intercepts and changes the agent's proposed action.

$$O_{rate} = \frac{\text{Number of Overridden Actions}}{\text{Total Episode Steps}}$$

A lower override rate, when paired with a low collision rate and high reward retention, indicates a more selective intervention profile.

Table 5.1 : Hyperparameter Configurations for the Evaluated HighwayEnv Models

Hyperparameter	DQN	PPO
Learning Rate (α)	3.692×10^{-4}	3.692×10^{-4}
Discount Factor (γ)	0.948	0.948
Optimizer	Adam	Adam
Batch Size	32	64
PPO Rollout Steps	–	2,048
PPO Epochs per Update	–	10
Replay Buffer Size	200,000	–
Target Update Interval	591	–
Exploration Fraction	0.01	–
Adam β_1	0.900	0.900
Training Timesteps	200,000	200,000

5.6.5 RL ARCHITECTURE AND TRAINING REGIME

For reproducibility, this section details the neural network architectures, hyperparameter configurations, and computational infrastructure used during baseline policy training before adversarial evaluation.

Neural Network Architecture: The HighwayEnv experiments evaluate DQN and PPO agents using Stable-Baselines3 [201]. Both agents operate on dense numerical observations, and the policy/value networks use MLP-based feature processing. PPO supplies the actor-critic baseline, whereas DQN supplies the value-based discrete-action baseline.

Hyperparameter Configuration: The models were built using the Stable-Baselines3 framework [201]. Table 5.1 summarizes the tuned DQN parameters and the PPO settings used for the evaluated runs.

The clean evaluation rows, rather than unreported validation curves, are used as the empirical baseline for the robustness analysis.

Training Infrastructure and Convergence: The training pipeline was executed on the Alliance Canada high-performance compute clusters (Nibi). To accelerate sample collection, the environments were vectorized using SubprocVecEnv, allowing 8 parallel environment instances to step synchronously.

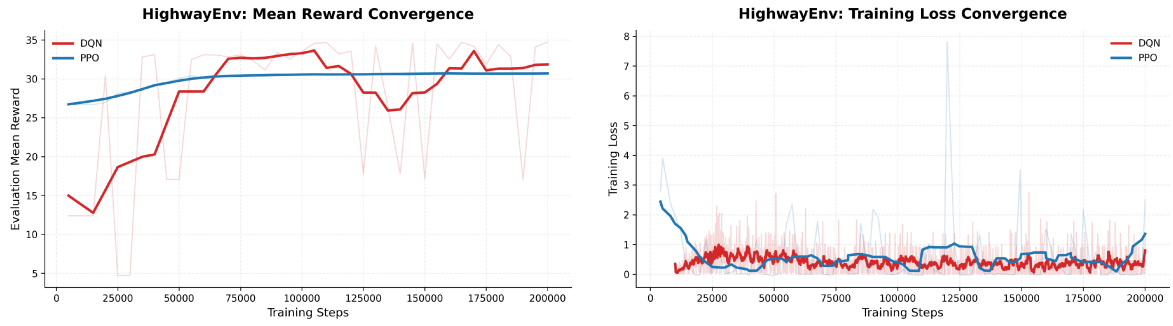


Figure 5.2 : HighwayEnv training convergence for the evaluated agents. Left: evaluation mean reward across training steps. Right: training loss across training steps.

The continuous-control PPO policy used in the MetaDrive extension (Section 5.7.4) is a separate, externally pre-trained policy that is frozen without fine-tuning for evaluation. Because this policy is not part of the HighwayEnv training configuration, its architecture, hyperparameters, and training curves are reported separately in Section 5.12.

Figure 5.2 summarizes the HighwayEnv optimization dynamics for both agents (further discussed in Section 5.10). The reward curve shows PPO converging more smoothly under the shared training budget, while the loss curve highlights the noisier but bounded optimization behavior of DQN and PPO during learning.

5.7 EXPERIMENTAL RESULTS

The empirical evaluation comprises 100 episodes per scenario. Instead of organizing the results by agent, this section presents five findings that jointly examine the architecture-defense coupling hypothesis introduced in Section 5.1. HighwayEnv clean runs are stable for both agents: DQN obtains 34.71 reward with 0.000 collision rate, and PPO obtains 30.71 reward with 0.000 collision rate; these collision-free clean baselines make HighwayEnv the appropriate environment for the clean-policy defense claims below. Tables 5.2 and 5.3 report the per-scenario HighwayEnv results for DQN and PPO, Table 5.4 reports the MetaDrive continuous-control extension, and Figures 5.3–5.5 visualize the aggregate reward, safety,

Table 5.2 : Selected 100-episode results for HighwayEnv (DQN, N=100). Reward is mean ± std; a value of ±0.0 indicates a standard deviation < 0.05, which rounds to zero at the reported precision. Act = action attack, Fixed-Cmd = sustained fixed-command injection, Eps = epsilon-malicious tampering, No Def = no defense, RR = reward-retention ratio (Section 5.6). Only selected rows are tabulated; the full scenario grid is shown in Figures 5.3–5.5.

Scenario	Reward ($R_{\text{total}} \uparrow$)	Collision Rate ($C_{\text{rate}} \downarrow$)	Override Rate	RR
Clean (Nominal)	34.71 ± 0.0	0.000	0.000	1.000
Observation Attacks				
Noise (No Def)	25.28 ± 7.7	0.710	0.000	0.728
Noise + DynIW	21.94 ± 8.3	0.720	0.000	0.632
Action Attacks				
Act Fixed-Cmd (No Def)	12.38 ± 0.0	1.000	0.000	0.357
Act Fixed-Cmd + DynIW	14.05 ± 0.0	1.000	0.062	0.405
Act Fixed-Cmd + QGap	12.38 ± 0.0	1.000	0.143	0.357
Act Eps (No Def)	24.02 ± 10.0	0.650	0.000	0.692
Act Eps + DynIW	27.40 ± 8.8	0.460	0.002	0.789
Act Eps + QGap	30.34 ± 8.0	0.290	0.087	0.874
Multi-Vector Attacks				
Composite (No Def)	11.18 ± 8.5	0.920	0.000	0.322
Composite + DynIW	10.97 ± 7.3	0.970	0.022	0.316
Composite + QGap	9.97 ± 7.1	0.970	0.134	0.287

and reward-retention trends across the full scenario grid. Findings 1–5 draw on this shared evidence base.

5.7.1 FINDING 1: SUSTAINED ACTION-SPACE ATTACKS ARE THE MOST DAMAGING

Across the evaluated HighwayEnv agents, sustained fixed-command injection is the most damaging scenario for safety. Undefended, it drives both DQN and PPO to a deterministic $C_{rate} = 1.000$ with zero reward variance (±0.0), whereas undefended observation noise leaves both agents with non-trivial reward variance and a collision rate strictly below 1.0 (DQN: 0.710; PPO: 0.940). This ranking does not extend to every action attack: the intermittent epsilon-malicious attack produces lower collision rates than observation noise for both agents (DQN: 0.650 vs. 0.710; PPO: 0.600 vs. 0.940), so severity depends on the persistence

Table 5.3 : Selected 100-episode results for HighwayEnv (PPO, N=100). Reward is mean ± std; a value of ±0.0 indicates a standard deviation < 0.05, which rounds to zero at the reported precision. Act = action attack, Fixed-Cmd = sustained fixed-command injection, Eps = epsilon-malicious tampering, No Def = no defense, RR = reward-retention ratio (Section 5.6). Only selected rows are tabulated; the full scenario grid is shown in Figures 5.3–5.5. The rows labeled QGap are proxy negative-control ablations, not native PPO $Q(s, a)$ implementations, because standard PPO does not expose an action-conditioned critic.

Scenario	Reward ($R_{\text{total}} \uparrow$)	Collision Rate ($C_{\text{rate}} \downarrow$)	Override Rate	RR
Clean (Nominal)	30.71 ± 0.0	0.000	0.000	1.000
Observation Attacks				
Noise (No Def)	10.88 ± 7.8	0.940	0.000	0.354
Noise + DynIW	10.27 ± 8.5	0.900	0.000	0.334
Action Attacks				
Act Fixed-Cmd (No Def)	12.38 ± 0.0	1.000	0.000	0.403
Act Fixed-Cmd + DynIW	30.70 ± 0.0	0.000	0.750	1.000
Act Fixed-Cmd + QGap (proxy)	4.60 ± 0.0	1.000	0.571	0.150
Act Eps (No Def)	20.92 ± 9.8	0.600	0.000	0.681
Act Eps + DynIW	30.70 ± 0.0	0.000	0.125	1.000
Act Eps + QGap (proxy)	17.44 ± 10.2	0.770	0.120	0.568
Multi-Vector Attacks				
Composite (No Def)	12.35 ± 7.9	0.980	0.000	0.402
Composite + DynIW	11.10 ± 7.9	0.940	0.153	0.361
Composite + QGap (proxy)	11.95 ± 8.1	0.930	0.141	0.389

of an attack as well as on its channel. For fixed-command injection, the gap is consistent with the threat model: action tampering (Section 5.4.2) bypasses the policy's output entirely and substitutes a fixed disruptive command, whereas observation noise (Section 5.4.1) is still processed by the policy's learned features before it reaches actuation. We did not run an experiment that isolates this mechanism.

The MetaDrive extension (Section 5.7.4) provides a particularly severe example: undefended fixed-command injection raises the collision rate to 1.000 while also increasing episodic reward by 51% over the clean baseline through MetaDrive's dense, progress-shaped reward function. Reward inflation of this kind depends on the environment's reward shaping and should not be assumed elsewhere. Observation attacks are also harmful: in HighwayEnv, spoofing drives the undefended PPO collision rate to 0.890 (Table 5.7; the full scenario grid is in Figure 5.4), and DynIW applied to observation-noise scenarios yields no meaningful

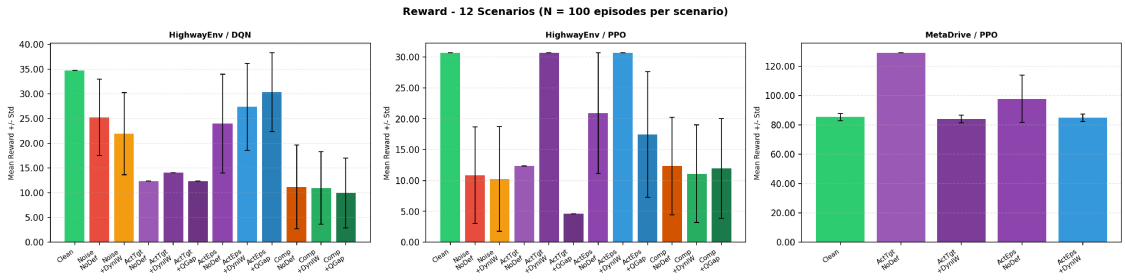


Figure 5.3 : Mean episodic reward across the 12 evaluated scenarios for HighwayEnv/DQN, HighwayEnv/PPO, and the externally pre-trained MetaDrive/PPO policy (N = 100 episodes per scenario). The MetaDrive fixed-command-injection and composite action cases show why reward must be interpreted alongside safety: dense progress rewards can increase even when the policy becomes unsafe.

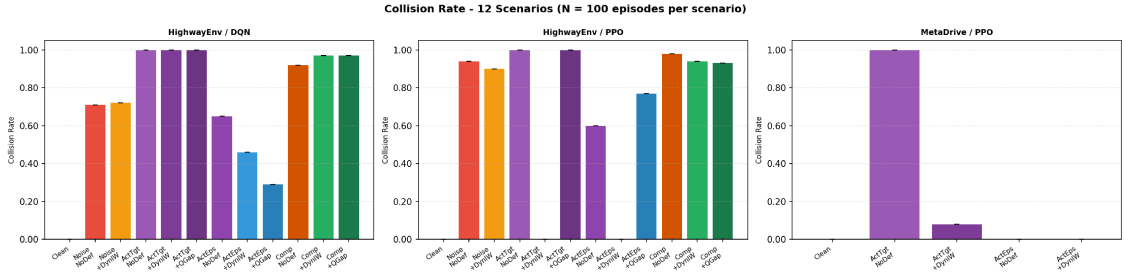


Figure 5.4 : Collision rate across the same scenario grid (N = 100 episodes per scenario). Action-space attacks produce the clearest safety failures, while DynIW (HighwayEnv/PPO) and DynIW (MetaDrive/PPO, continuous-action branch) suppress fixed-command-injection failures.

collision reduction (DQN: 0.710 to 0.720; PPO: 0.940 to 0.900), plausibly because action-consistency gating does not address corrupted observations; we did not test this explanation. These results suggest that sustained actuation tampering deserves priority as a threat surface for inference-time defense design in AV policies, without implying that observation attacks are benign.

5.7.2 FINDING 2: DEFENSE EFFECTIVENESS IS ARCHITECTURE-DEPENDENT

Figures 5.4 and 5.5, together with Tables 5.2 and 5.3, indicate that no single inference-time wrapper was uniformly best in our experiments; the effective wrapper appears to depend on the underlying policy architecture. PPO-style actor-critic policies are recovered most

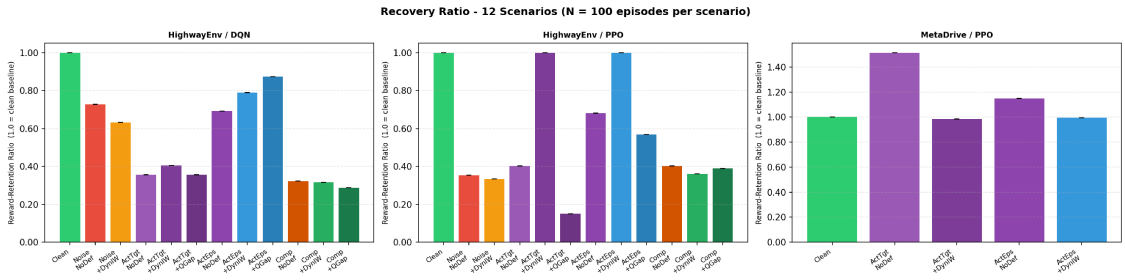


Figure 5.5 : Reward-retention ratio normalized by the clean reward for each environment-agent pair (N = 100 episodes per scenario). Values near 1.0 indicate near-clean reward retention; values above 1.0, especially in MetaDrive, reflect reward inflation from shaping effects and must be read alongside collision rate.

reliably by DynIW's observation gating with policy-referenced action validation: under both sustained fixed-command injection and epsilon-malicious action tampering, DynIW restores PPO to near-clean reward (30.70 vs. 30.71 clean), yielding $\mathcal{RR} = 1.000$ to three decimal places (30.70/30.71 = 0.9997) in both scenarios alongside a 0.000 collision rate. The zero reward variance (±0.0) accompanies a 0.750 override rate under fixed-command injection, which is consistent with the wrapper replacing most adversarial commands and reproducing the clean trajectory; we did not analyze this further. For the evaluated DQN checkpoint, QGap was effective under intermittent action tampering, but not under sustained fixed-command injection or composite attacks: under the epsilon-malicious attack, QGap reduces DQN's collision rate from 0.650 (no defense) to 0.290 and lifts $\mathcal{RR}$ from 0.692 to 0.874. DynIW also reduces DQN's epsilon-attack collision rate (0.650 to 0.460), but less than QGap, and it does not help DQN under fixed-command injection (collision rate 1.000, Table 5.2); the coupling therefore describes tendencies observed here and not a strict dichotomy.

Figure 5.5 visualizes this split across the full attack/defense grid: both agents are collision-free under clean conditions (DQN reward 34.71; PPO reward 30.71), and defense effectiveness appears to cluster by architecture as well as by attack type. These results suggest that no isolated wrapper was architecture-agnostic here, and that AV defense stacks may need to select their gating mechanism based on whether the deployed policy is value-based or actor-critic, rather than deploying a single universal protection mechanism.

5.7.3 FINDING 3: QGAP WAS EFFECTIVE WHERE AN ACTION-VALUE LANDSCAPE WAS AVAILABLE

QGap's behavior across the two HighwayEnv agents suggests a condition under which value-landscape gating succeeds. For the evaluated DQN checkpoint, QGap was effective under intermittent action tampering, but not under sustained fixed-command injection or composite attacks. A plausible condition for success is that the policy's Q-value estimates reliably discriminate safe from unsafe discrete actions: for DQN under the epsilon-malicious attack, QGap achieves $\mathcal{RR} = 0.874$, compared with 0.692 for the undefended baseline. The QGap-style proxy did not help the tested PPO checkpoint: under sustained fixed-command injection, it reduces PPO's reward to 4.60 ($\mathcal{RR} = 0.150$, compared with the 0.403 undefended baseline) while still leaving the collision rate at 1.000. This result should be read narrowly: standard PPO exposes no action-conditioned critic, so the QGap-style gate evaluated here is a hard-label proxy (defined below) that discards the soft value information on which native QGap relies. The evaluated hard-label proxy did not improve the tested PPO checkpoint. Because per-step diagnostics were not collected, the experiment cannot determine whether this failure is intrinsic to label-based gating or specific to the implemented proxy. It is therefore not evidence of a general limitation of label-based gating, of QGap, or of value-aware gating for PPO.

Table 5.2 also reveals an instructive edge case for DQN: Act Fixed-Cmd + QGap records a 14.3% override rate yet produces the same reward and collision rate as undefended fixed-command injection. The override rate indicates that QGap is flagging value-gap anomalies, but the interventions did not change the outcome. A plausible interpretation is that action-level intervention is insufficient once the physical state has become unrecoverable. Three factors may contribute; we did not run per-step diagnostics to separate them. First, fixed-command injection is sustained and deterministic: it replaces the policy's output at every step, so QGap's corrective overrides are isolated corrections within an otherwise adversarially dominated trajectory. Even a 14.3% intervention budget leaves 85.7% of steps executing the

adversarial command, which may be sufficient to drive the vehicle into unsafe proximity. Second, kinematic commitment: in HighwayEnv's high-speed lane-keeping setting, a single corrective action is unlikely to reverse the lateral displacement or closing velocity accumulated over preceding adversarial steps. Once the ego vehicle is within collision range of a neighbour (plausibly a small number of timesteps before impact), the transition dynamics $\mathcal{P}(s_{t+1}|s_t, a_t)$ are already committed to a colliding trajectory regardless of which action is executed at the current step. Third, this pattern distinguishes sustained attacks from intermittent ones: under the epsilon-malicious attack (15% per-step substitution rate), QGap's corrections are temporally distributed across a smaller fraction of dangerous steps and can prevent enough unsafe transitions to measurably reduce the collision rate (C_{rate}: 0.650→0.290). Under a sustained attack, the substitution rate may exceed what single-step gating can recover. This explanation is incomplete, however: DynIW also acts step by step, yet it recovered PPO (collision rate 1.000 to 0.000) but not DQN (1.000) under the same fixed-command injection, and we did not diagnose the difference between the two agents. One plausible contributor, not tested here, is that DynIW's PPO branch compares each command with the policy's own action, whereas its DQN branch overrides only when the proposed action's Q-value falls below half of the maximum.

The same limitation appears in the MetaDrive extension (Section 5.7.4), where the frozen continuous-control critic is a state-only value function $V(s)$ with no action-conditioning; consequently, no per-action Q landscape exists for QGap to evaluate. In the HighwayEnv/PPO rows labeled QGap, the proxy does not estimate a PPO critic value. Instead, it compares the tampered action against PPO's own greedy actor output and asks whether that proposed action should be overridden in favor of the actor's preferred action. This makes the PPO rows a negative-control test of a QGap-style gate under an architecture that lacks $Q(s, a)$, not evidence of a genuine PPO action-value comparison.

What the proxy does and does not test. The proxy is a binary, label-level gate: it overrides whenever the proposed action differs from the actor's argmax and carries no graded informa-

tion about *how much worse* the proposed action is. Native QGap is graded. If, for example, Go Straight scores 90, Turn Right 88, and Brake Violently 5, a threshold on ΔQ_t passes the safe alternative (gap of 2) and rejects only the harmful command (gap of 85), whereas the proxy treats both as the same disagreement and overrides both. The proxy may therefore over-intervene whenever a benign deviation from the argmax occurs. This over-intervention account is a hypothesis and not a diagnosed cause. Under fixed-command injection the substituted command is fixed, so benign alternatives are not the direct source of overrides there, and we did not diagnose why the proxy's reward (4.60) fell below the undefended reward (12.38). Per-step diagnostics that separate unnecessary overrides from overrides of harmful commands were not collected. Accordingly, the PPO rows should be read only as showing that the implemented hard-label proxy did not improve the tested checkpoint. They do not establish that label-based gating is intrinsically limited, and they do not bound the performance of a value-aware gate for PPO.

Why the evaluation does not use simulator look-ahead. Obtaining soft per-action scores for PPO's choices by stepping a copy of the simulator one step ahead was considered and deliberately excluded, for three reasons. First, it changes the defense class: the wrapper would require query access to the environment's transition model at inference time, which is an assumption the retraining-free, model-free setting studied here does not make, and which real vehicles satisfy only through an approximate learned or analytic model. Second, the scores it returns are one-step simulated outcomes (for instance, reward, proximity, or collision flags) and not the agent's own value estimates. The resulting gate would be a model-based safety filter, structurally closer to the shielding and predictive approaches discussed under predictive safeguards below than to QGap, and its performance could not be attributed to the policy's value landscape. Third, a one-step horizon is the horizon that Finding 3 suggests is too short once momentum has accumulated, so the look-ahead would likely need several steps to be informative. Two softer options that stay within the original model-free setting are left to future work: (i) a logit-gap gate that uses PPO's own action log-probabilities,

$\log \pi_\theta(a^*|s_t) - \log \pi_\theta(\tilde{a}_t|s_t)$, as a graded stand-in for ΔQ_t, and (ii) a short-horizon model-based variant, as described below.

Predictive safeguards for sustained attacks. The 1.000 collision rate of QGap on DQN and of the PPO QGap proxy under fixed-command injection suggests a limitation of reactive, single-step gating. QGap and the proxy decide using only the current state and the current command, so they cannot detect a drift that is individually benign at each step but cumulatively leads to an unrecoverable state. This differs from the attack evaluated here, which overrides the command at every step from the start of the episode and not through small intermittent pushes; for small repeated perturbations, however, the concern is valid and applies equally to the evaluated wrappers. A natural remedy that requires no retraining is a predictive safety layer that rolls out a dynamics model over a short horizon (for example, 2–3 s, or a few decision steps) and rejects commands whose predicted trajectory leaves a safe set. Two standard instances are short-horizon Model Predictive Control (MPC) [194] and Control Barrier Function (CBF) safety filters [195], which also belong to the shielding family of Alshiekh et al. [192]. We view such a layer as a complement to, not a replacement for, the wrappers studied here, and it is not evaluated in this chapter. Three caveats bound its expected benefit. (a) It needs a dynamics model of the ego vehicle and a prediction of neighbouring vehicles, which adds design effort and model-mismatch risk absent from QGap and DynIW. (b) It protects only if the filter sits downstream of the compromised channel; an adversary with write access past the filter defeats any actuation-side check. (c) If it intervenes on most steps, it effectively replaces the learned policy and the system's safety then rests on the model and the barrier definition, not on the policy. DynIW supplies a limited form of temporal processing through its filtered observation state and compares the proposed action against the protected policy's nominal action. This is consistent with its recovery of PPO under fixed-command injection, although the mechanism was not isolated experimentally. This suggests that adding multi-step prediction is the most promising extension, but whether MPC or CBF filtering would outperform DynIW at comparable cost remains an open empirical

question. The architecture-dependent conclusions of this chapter should be read with this limitation in mind: reactive inference wrappers may be insufficient once the adversary has driven the vehicle into an unrecoverable physical state, since value- or signal-gating cannot reverse irreversible dynamics, and only a predictive layer could act earlier.

Summary of QGap's applicability. The results suggest that QGap's applicability depends on a structural condition, namely the availability of an informative, action-conditioned value landscape, rather than on nominal agent performance or attack family. When this condition holds, as in value-based DQN, QGap can be applied natively. For the evaluated DQN checkpoint, QGap was effective under intermittent action tampering, but not under sustained fixed-command injection or composite attacks. When it does not hold for discrete actor-critic policies such as PPO, only a proxy can be evaluated; the evaluated hard-label proxy did not improve the tested PPO checkpoint, which does not settle whether label-based gating is intrinsically limited, and graded alternatives remain untested. When the policy exposes only a continuous state-value function (as in MetaDrive), no viable implementation exists and QGap cannot be evaluated at all.

5.7.4 FINDING 4: DYNIW'S CONTINUOUS-ACTION BRANCH REDUCES COLLISIONS FOR A CONTINUOUS-CONTROL PPO CHECKPOINT UNDER FIXED-COMMAND INJECTION

Finding 3 suggested that QGap's applicability is restricted to policies with an informative, action-conditioned value landscape. The MetaDrive experiment therefore uses DynIW's continuous-action branch (Section 5.5.3), a policy-referenced action check, on a frozen, externally pre-trained continuous-control PPO checkpoint. Because the MetaDrive policy has no Q-network, DynIW validates actions against the policy's own nominal action using an L2 threshold of 0.30. DynIW substantially reduced collisions for the evaluated continuous-control PPO checkpoint under fixed-command injection: the collision rate rises from 0.000 (clean) to 1.000 with no defense under sustained fixed-command injection, but falls to 0.080

Table 5.4 : Selected 100-episode results for MetaDrive: externally pre-trained continuous PPO checkpoint with DynIW's continuous-action branch. Reward is mean ± std. QGap is omitted because it requires a discrete per-action value landscape that is unavailable in this frozen, state-value-only continuous-control policy.

Scenario	Reward ($R_{\text{total}} \uparrow$)	Collision Rate ($C_{\text{rate}} \downarrow$)	Override Rate	RR
Clean Baseline				
Clean (Nominal)	85.24 ± 2.5	0.000	0.000	1.000
Sustained Fixed-Command Injection				
Act Fixed-Cmd (No Def)	129.13 ± 0.0	1.000	0.000	1.515
Act Fixed-Cmd + DynIW	83.95 ± 2.5	0.080	0.235	0.985
Epsilon-Malicious Action Tampering				
Act Eps (No Def)	97.74 ± 16.0	0.000	0.000	1.147
Act Eps + DynIW	84.71 ± 2.5	0.000	0.133	0.994

with DynIW (Table 5.4). Implementation details (including the external policy loading, the continuous-action interface generalizations for attacks and defenses, and the evaluation protocol) are provided in Section 5.12.

Reward Alone Is an Insufficient Robustness Metric: Table 5.4 shows that undefended sustained fixed-command injection increases mean reward by 51% over the clean baseline (129.13 vs. 85.24) while producing a 100% collision rate. Its $\mathcal{RR} = 1.515$ therefore indicates reward inflation, not successful recovery. This behavior follows from MetaDrive's dense, distance-progress reward shaping: forcing maximal throttle with zero steering maximizes per-step longitudinal progress until the episode terminates by collision, and the accumulated progress reward exceeds the one-time collision penalty. The zero-variance reward (±0.0) indicates a deterministic exploit rather than a stochastic outcome. This finding reinforces that episodic reward and $\mathcal{RR}$ are insufficient proxies for safety, and robustness evaluations of AV policies must report collision rate or equivalent safety indicators alongside reward.

DynIW Reduces Collisions Without Retraining in MetaDrive; QGap Does Not Apply: Despite this adversarial reward-hacking effect, DynIW substantially reduced the observed collision rate without retraining. Under fixed-command injection, DynIW reduces the colli-

sion rate from 1.000 to 0.080 while returning the reward to a near-clean level ($\mathcal{RR}$ = 0.985), thereby discounting the artificially inflated undefended reward. Under the less severe epsilon-malicious attack, the undefended collision rate is already 0.000 in this sample; DynIW preserves safety (collision = 0.000) while achieving $\mathcal{RR}$ = 0.994, indicating that the override rule does not destabilize the policy when the attack is comparatively benign. These results should be read with the design of this branch in mind: its reference is the nominal action of the protected policy itself, so it requires trusted access to that action and addresses actuation tampering only. The HighwayEnv QGap proxy of Finding 3 also compares against the actor's own greedy action, as does DynIW's discrete-PPO branch. The two differ in the observation gating that DynIW applies, in that the proxy holds each override for one additional step, and in the environment; because the proxy's failure was not diagnosed, the outcomes are not attributed to a single cause.

QGap is excluded from this MetaDrive comparison because the frozen external policy's critic provides only a state-value $V(s)$ with no action-conditioning; in the absence of a discrete action set to enumerate, no value-based QGap proxy is viable. This differs from the HighwayEnv/PPO setting, where a hard-label proxy was evaluated as a negative-control ablation of QGap (Finding 3). DynIW's continuous branch is not such a proxy: it does not estimate action values. This distinction separates the findings by architecture and action space:

- **Discrete, value-based systems (HighwayEnv DQN):** the value landscape appears informative, and QGap was the more effective defense under intermittent tampering.
- **Discrete, actor-critic systems (HighwayEnv PPO):** native QGap requires an action-conditioned $Q(s, a)$ that standard PPO does not expose; a hard-label proxy (actor-distribution mode) was evaluated as a negative-control ablation and did not improve the tested checkpoint (the experiment cannot tell whether this is intrinsic to label-based gating or specific to the proxy; graded variants were not evaluated). DynIW was the better-performing defense in our experiments.

- **Continuous, state-value-only systems (MetaDrive PPO):** no action-conditioned Q landscape exists, so QGap was not evaluated. DynIW, whose continuous branch references the policy's nominal action, was the only wrapper evaluated.

Taken together, these results provide preliminary evidence that DynIW's continuous-action branch, an action check referenced to the protected policy's nominal action, substantially reduced collisions for one independently trained continuous-control PPO checkpoint under fixed-command injection, without retraining. They do not establish how this branch behaves for other policies, other attacks, or observation-space attacks.

5.7.5 FINDING 5: COMPOSITE ATTACKS REMAIN UNSOLVED

Composite (multi-vector) attacks remain the most challenging setting in the study, and no single evaluated wrapper consistently restores clean performance against them. In HighwayEnv, undefended composite attacks produce collision rates of 0.920 (DQN) and 0.980 (PPO); adding DynIW or QGap changes these only marginally. Composite+QGap leaves DQN's collision rate at 0.970 (vs. 0.920 undefended, a difference that is not clearly distinguishable at $N = 100$) with slightly lower reward (9.97 vs. 11.18 undefended), and Composite+DynIW leaves PPO's collision rate at 0.940, close to the 0.980 undefended case. Reward-retention ratios under composite attacks remain well below 1.0 across every agent-defense combination, which is consistent with compounding effects of observation spoofing and action tampering, although this study does not isolate their interaction: defenses that mitigate isolated attack vectors do not automatically compose into a robust multi-vector solution. This composite ceiling is consistent across both discrete architectures evaluated in HighwayEnv, and it is the setting in which the architecture-defense coupling hypothesis offers no sufficient single-wrapper prescription.

5.8 DISCUSSION

The obtained results are consistent with the architecture-defense coupling hypothesis introduced in Section 5.1 and support three further observations. First, across the policies,

Table 5.5 : Defense-only latency per control step, averaged over 500 timed trials (50 warm-up calls excluded).

Environment / Agent	Defense	Latency (ms/step)
HighwayEnv / DQN	NoDefense	0.000
HighwayEnv / DQN	EMA	0.002
HighwayEnv / DQN	QGap	0.063
HighwayEnv / DQN	DynIW	0.149
HighwayEnv / PPO	NoDefense	0.000
HighwayEnv / PPO	EMA	0.006
HighwayEnv / PPO	QGap	0.173
HighwayEnv / PPO	DynIW	0.397

Table 5.6 : Architecture-Defense Coupling: Key Findings Summary

Architecture	Action Space	Better-Performing Defense	Evidence
DQN (value-based)	Discrete	QGap	$\mathcal{RR}$ = 0.874 vs. 0.692 (no-def); collision 0.290 vs. 0.650 under ϵ-malicious attack
PPO (actor-critic)	Discrete	DynIW	Sustained fixed-command injection: $\mathcal{RR}$ = 1.000 (reward 30.70/30.71 = 0.9997 unrounded); collision 0.000 vs. 1.000 (no-def)
PPO (actor-critic)	Continuous	DynIW (continuous-action branch)	MetaDrive (frozen checkpoint, fixed-command injection): collision 0.080 vs. 1.000 (no-def), $\mathcal{RR}$ = 0.985; no retraining

attacks, and environments evaluated in this study, defense effectiveness varied with both policy architecture and attack configuration; within the evaluated settings, the better-performing wrapper differed across the DQN and PPO agents. QGap was the better defense for DQN under intermittent tampering; for discrete PPO a hard-label QGap proxy was evaluated as a negative-control ablation and did not improve the tested PPO checkpoint (without per-step diagnostics, this cannot be attributed to label-based gating in general), while DynIW was the preferred defense for the discrete PPO agent in our experiments; and for the continuous state-value-only critic, native QGap has no viable implementation and was not evaluated, and DynIW's continuous-action branch substantially reduced collisions under fixed-command injection. Second, baseline stability matters: HighwayEnv's collision-free clean baselines support the clean-policy defense claims. Third, runtime gates can reduce attack impact, but they cannot recover from every physical state, especially under sustained fixed-command injection and composite attacks.

5.8.1 IMPLICATIONS FOR DEPLOYMENT

The evidence here suggests that the monitor may need to be selected according to the deployed policy architecture. PPO-style actor-critic policies may benefit from policy-referenced action checks such as DynIW, while value-based DQN policies may benefit from QGap when the value landscape is reliable. Composite attacks may require layered defenses that jointly address observation integrity, action consistency, and longer-horizon recoverability; such combinations were not evaluated here.

5.8.2 LATENCY AND REAL-TIME CONSTRAINTS

Table 5.5 reports the measured per-step latency for each evaluated wrapper, averaged over 500 timed trials. All wrappers remain below 0.40 ms per control step. DynIW is the heaviest wrapper for both agents (0.149 ms/step on DQN; 0.397 ms/step on PPO), likely owing to its observation-gate update and the extra policy or Q-network query used for action validation. QGap adds 0.063 ms/step on DQN and 0.173 ms/step on PPO, probably dominated by the extra model query needed for action-value comparison. EMA remains negligible (0.002–0.006 ms/step across agents), near the NoDefense baseline. These timings indicate that the wrappers incurred sub-millisecond overhead on the evaluation platform; however, embedded automotive ECUs and high-frequency control loops require hardware-in-the-loop profiling before deployment, especially when multiple wrappers are stacked or when GPU synchronization overhead differs from the cluster environment.

5.8.3 PRACTICAL RECOMMENDATIONS AND FAILURE MITIGATION

- **Consider matching the defense to the policy:** QGap for value-based agents (DQN) when Q estimates are informative, and DynIW for actor-critic agents (PPO), pending multi-seed validation.

- **Prefer low-cost filters as first-line:** EMA is computationally inexpensive (Table 5.5), but its closed-loop effect was not evaluated here and smoothing may add phase lag (Section 5.5); it should be validated before use.

- **Layer defenses for composite threats:** Combining observation-side filters with action gating and a safe fallback controller is a plausible approach to multi-vector attacks, but it was not evaluated here.
- **Monitor value informativeness:** Consider a lightweight meta-check (not evaluated here) that disables QGap when per-action Q variance is low or unstable.
- **Profile and optimize:** If latency is critical, optimize DynIW/QGap in native code; evaluate hardware-in-the-loop before deployment.

5.8.4 INTERPRETING AGGREGATE PLOTS AND FAILURE MODES

The aggregate scenario plots (Figures 5.3–5.5) and tables indicate that both agents are collision-free under clean conditions and that defense effectiveness varied with both policy architecture and attack configuration. The QGap proxy's failure on the tested PPO checkpoint may be related to the absence of a graded per-action Q landscape, since a hard-label proxy cannot distinguish safe alternatives from harmful commands; this is a hypothesis, and without per-step diagnostics it cannot be separated from an implementation-specific cause. DynIW's success on PPO is hypothesized to stem largely from validating each action against the policy's own output on the gated observation, which makes it an oracle-like check against actuation-only tampering; we did not test this directly (for example, by ablating the observation gate). Composite attacks remain difficult, plausibly because they simultaneously degrade the signal quality for both observation filters and action-gating heuristics.

5.8.5 HYPOTHESIS AND KEY FINDINGS

The central hypothesis of this work is that lightweight, inference-time wrappers can mitigate several common attack modes without retraining, provided the wrapper's inductive biases align with the deployed policy architecture. The empirical study yields the following findings:

- **DynIW (dynamic observation gating with policy-referenced action validation):** Preferred in our experiments for the discrete-action actor-critic (PPO) controller in HighwayEnv. Trade-offs: higher computational cost (Table 5.5); it did not recover DQN under sustained fixed-command injection, and its effectiveness is expected to be lower when the vehicle is already near an unrecoverable state.

- **QGap:** For the evaluated DQN checkpoint, QGap was effective under intermittent action tampering, but not under sustained fixed-command injection or composite attacks; under intermittent tampering, per-action Q-values appear to discriminate safe from unsafe actions ($\mathcal{RR} = 0.874$ under ϵ-malicious). For discrete PPO, a proxy approximation (actor's greedy output as Q-value stand-in) was evaluated as a negative-control ablation; this hard-label proxy did not improve the tested checkpoint ($\mathcal{RR} = 0.150$ under fixed-command injection vs. 0.403 undefended). Because per-step diagnostics were not collected, the experiment cannot determine whether this failure is intrinsic to label-based gating or specific to the implemented proxy. Graded variants (e.g., a logit-gap gate) were not evaluated. For continuous state-value-only critics, no viable implementation exists and QGap was not evaluated.

- **DynIW, continuous-action branch:** Overrides a command when it deviates from the protected policy's nominal action by an L2 distance of at least 0.30. It substantially reduced collisions for the evaluated continuous-control PPO checkpoint under fixed-command injection ($1.000 \rightarrow 0.080$). It requires trusted access to the nominal action, addresses actuation tampering only, and was evaluated on one checkpoint.

Together, these results suggest, as a hypothesis not tested here, a selective escalation strategy: use a lightweight first-line filter (such as EMA, if validated), then escalate to DynIW or QGap when diagnostics (e.g., override rates or Q-variance) indicate that their structural assumptions are met.

5.9 THREATS TO VALIDITY

To support rigorous interpretation, this section outlines the primary threats to validity and the boundaries of the study's conclusions.

5.9.1 INTERNAL VALIDITY: THRESHOLD TUNING BIAS

A primary threat to internal validity is threshold sensitivity. QGap uses a fixed threshold $\tau_{\text{gap}} = 0.05$ and DynIW uses fixed thresholds $\tau_{\text{idw}} = \tau_{\text{spike}} = 0.50$; both were selected based on pilot runs in the same HighwayEnv environment used for evaluation. Because threshold selection and performance measurement share the same environment and agent configurations, the reported results may overestimate performance relative to a fully held-out threshold selection protocol. This bias is most acute when the clean-policy reward gap between safe and unsafe actions is large (favoring QGap) or when traffic dynamics are regular (favoring DynIW's spike-based observation gate). Future work should adopt cross-validated or adaptive thresholding that scales with real-time uncertainty estimates and distance to unrecoverable states, and should report sensitivity curves over a range of threshold values to bound the tuning effect.

5.9.2 INTERNAL VALIDITY: SINGLE-SEED LIMITATIONS

The HighwayEnv evaluations use a single training seed for each agent. Within-episode stochasticity is present (traffic initialization and NPC behavior vary across episodes), but policy-training variance across seeds is not characterized. It is possible that the observed architecture-defense coupling is partially seed-specific: a different training seed could yield DQN or PPO agents with different Q-landscape informativeness or policy-gradient stability, which would shift the relative effectiveness of QGap versus DynIW. A multi-seed training replication (five independent seeds is a common standard) would quantify this variance and strengthen the coupling claim. The MetaDrive extension is similarly limited to a single publicly available policy; seed-level variance in the MetaDrive results cannot be assessed from this evaluation.

5.9.3 EXTERNAL VALIDITY: SINGLE-POLICY-FAMILY EVALUATION

The study evaluates only two policy families: value-based DQN and actor-critic PPO, both implemented with Stable-Baselines3 using the same MLP backbone (Table 5.1). The architecture-defense coupling hypothesis has not been tested against other families, such as Soft Actor-Critic (SAC), Twin Delayed DDPG (TD3), or distributional RL algorithms (C51, QR-DQN). Distributional critics may expose richer action-value landscapes that improve QGap applicability; off-policy actor-critic agents may exhibit intermediate coupling behavior not captured by the DQN/PPO dichotomy. The coupling reported here is therefore a hypothesis supported by two-family evidence, not a proven general principle across all DRL architectures.

5.9.4 EXTERNAL VALIDITY: HIGHWAYENV DEPENDENCE

The main architecture-defense coupling evidence (Findings 1–3 and 5) is drawn entirely from HighwayEnv. HighwayEnv provides a clean benchmark because both DQN and PPO achieve 0.000 clean collision rate, eliminating baseline instability as a confound. However, the environment's simplicity (discretized lane-change control, limited NPC variety, and deterministic road geometry) may favor certain defense mechanisms. More complex environments, such as intersection management, dense urban traffic, or variable-speed continuous control, may produce observation dynamics that undermine DynIW's fixed spike threshold, or reward shaping that inflates QGap's apparent effectiveness. The MetaDrive extension provides a single cross-environment data point but does not systematically vary environment complexity. Evaluating the coupling across a suite of environments with varying traffic density, action granularity, and reward structure is an important direction for future work.

5.9.5 EXTERNAL VALIDITY: THE SIM-TO-REAL GAP

The empirical claims in this chapter are constrained to simulation. HighwayEnv's simplified kinematics cannot replicate complex real-world driving physics, including tire friction degradation, suspension dynamics, variable weather conditions, or sensor occlusions. Trans-

ferring inference-time wrappers to physical hardware introduces strict computational constraints not captured by the simulation latency measurements: embedded automotive ECUs operating at high-frequency control loops may face hardware-level DMA latency, interrupt scheduling, and GPU synchronization overhead absent in the cluster environment. The reported latency measurements (Table 5.5) should therefore be interpreted as sub-millisecond overhead measured on the evaluation platform and as lower bounds on deployment overhead; they do not establish real-time operation on embedded hardware.

The MetaDrive extension evaluates the frozen policy on its in-distribution map configuration to isolate attack-and-defense effects from map-generalization variability. The reported MetaDrive collision rates should be interpreted as in-distribution robustness estimates, not as evidence of out-of-distribution generalization to unseen road layouts or traffic patterns. A systematic out-of-distribution MetaDrive study is left to future work.

5.9.6 CONSTRUCT VALIDITY: ATTACK-MODEL SIMPLIFICATIONS

The formulated attacks are idealized. Sustained Fixed-Command Injection assumes the adversary has synchronized, real-time access to the actuation pipeline and can unconditionally replace the agent's action at every step with a predefined command; it does not model a state-dependent worst-case adversary (an $\arg\min_a Q^\pi(s_t, a)$ attack), which could be more damaging. In practice, CAN-bus injection requires physical or privileged network access and may be detectable by existing intrusion detection systems. The epsilon-malicious attack models a stochastic, intermittent fault rather than an adaptive adversary who observes the defense's intervention pattern and adjusts accordingly. The composite attack combines fixed-intensity perturbations simultaneously; real multi-vector attacks may be temporally correlated in ways that either amplify or reduce their joint effect. Future work should evaluate the studied defenses against adaptive adversaries that model and exploit the defense wrapper's decision boundary, and against physically plausible attack traces derived from recorded real-world CAN-bus traffic or sensor logs.

5.9.7 CONSTRUCT VALIDITY: PROXY FIDELITY AND REACTIVE GATING

Two design limitations bound the conclusions on QGap. First, the HighwayEnv/PPO QGap rows use a hard-label proxy rather than a graded value comparison, so they test only whether this implemented proxy helps the tested checkpoint, not whether label-based gating in general, or a value-aware gate, is feasible for PPO. They also cannot separate over-intervention on safe alternatives from correct intervention on harmful commands, as no per-step override diagnostics were collected. Second, all evaluated wrappers (QGap, the proxy, and DynIW) are reactive and decide from a short history at most; none rolls out a dynamics model. Under sustained or slowly accumulating attacks, such wrappers may act only after the vehicle is in an unrecoverable state. A predictive MPC or CBF safety layer, a graded logit-gap gate for PPO, and a per-step override analysis are natural follow-ups, but they require new experiments and are outside the scope of this study.

5.9.8 CONSTRUCT VALIDITY: SCOPE OF THE POLICY-REFERENCED CHECK

DynIW's branches for policies without a Q-network (discrete PPO in HighwayEnv and continuous PPO in MetaDrive) build their reference from the nominal action of the protected policy, and their results should be interpreted accordingly. The reference presumes that the wrapper can recompute that action from the observation it holds and that tampering occurs only downstream of the policy. Under this assumption the check is closer to an oracle against actuation-only tampering than to a detector that must infer the nominal behavior, which may explain part of its effectiveness on PPO. It offers no protection against errors made by the policy itself, and the observation gate offers only limited protection against observation-space attacks (see the Noise and Composite rows). The MetaDrive evaluation used one frozen checkpoint, in-distribution, under two action-tampering attacks, and only DynIW was run there; the HighwayEnv and MetaDrive results use different branches of DynIW (a discrete label test and a continuous L2 test) and are therefore not directly comparable.

5.10 APPENDIX: HIGHWAYENV TRAINING CURVES

Figure 5.2 shows the optimization dynamics of the HighwayEnv agents over the full 200,000-step training budget. The reward curve indicates that PPO reaches a smoother and more stable plateau earlier than DQN, while DQN exhibits larger intermediate oscillations before settling near its final performance level. The loss curve follows the same pattern: PPO decays rapidly and then remains comparatively flat, whereas DQN fluctuates more strongly during learning but eventually stabilizes as training progresses.

The key point is that both agents converge to stable behavior by the end of training. By 200,000 steps, the reward trajectories have largely flattened and the loss values no longer show sustained drift, which suggests that the final checkpoints represent converged policies rather than transient training states. This is the interpretation relevant for this chapter: the curves document that the reported evaluation results are based on agents that had already reached a stable training regime before robustness testing began.

5.11 APPENDIX: ADDITIONAL RESULTS

The appendix preserves additional selected HighwayEnv rows for composite and observation-filter failure cases.

5.11.1 OBSERVATION AND COMPOSITE FAILURE CASES

Table 5.7 records selected failure cases for observation filtering and composite attacks.

Table 5.7 : Selected observation-filter and composite-attack failure cases.

Setting	Reward	Collision	RR
Observation Attacks			
HighwayEnv/PPO spoof, NoDefense	12.74	0.890	0.415
Multi-Vector Attacks			
HighwayEnv/DQN composite, NoDefense	11.18	0.920	0.322
HighwayEnv/DQN composite, QGap	9.97	0.970	0.287
HighwayEnv/PPO composite, NoDefense	12.35	0.980	0.402
HighwayEnv/PPO composite, QGap	11.95	0.930	0.389

5.12 APPENDIX: EXTERNAL METADRIVE PPO MODEL, TRAINING CONFIGURATION AND INTERFACE DETAILS

5.12.1 EXTERNAL POLICY AND EVALUATION PROTOCOL

The continuous-control PPO policy used in the MetaDrive extension (Section 5.7.4) is an externally developed policy from a public MetaDrive PPO implementation [1], built on top of the MetaDrive simulator [199], and trained for 5,000 epochs. The policy is loaded with its parameters *frozen*, without retraining or fine-tuning, and evaluated through MetaDrive's standard `env.step(action)` interface. The state observation is a 259-dimensional LiDAR-based vector (240 lasers, 50 m range, Gaussian noise 0.1), and the action is a 2-dimensional continuous vector $a_t = [\delta_{\text{steer}}, \delta_{\text{throttle}}] \in [-1, 1]^2$. Every episode uses the in-distribution map and traffic configuration used by the external policy (`"SS"`/0.3 with bi-directional traffic) to isolate attack-and-defense effects from map-generalization variability. Each scenario is evaluated for 100 episodes with DynIW's continuous-action threshold fixed at 0.30 (Section 5.5.3).

While the HighwayEnv benchmark evaluates both value-based (DQN) and actor-critic (PPO) policies in discrete action spaces, standard DQN cannot natively support the 2-dimensional continuous control required by MetaDrive without arbitrary discretization. This extension therefore evaluates continuous-action control with only the frozen continuous PPO policy.

5.12.2 CONTINUOUS-ACTION ATTACK INTERFACE AND THE DYNIW CONTINUOUS BRANCH

Because the reference policy is continuous, the attack interface described in Section 5.4 is generalized from discrete action indices to continuous action vectors. Sustained fixed-command injection is instantiated as the maximal-throttle, zero-steer command $a_{\text{fixed}} = [0, 1]$, so that $\tilde{a}_t = [0, 1]$ at every attacked step; as in the discrete setting, it is state-independent and is not an $\arg\min_a Q^\pi(s_t, a)$ attack. The defense used in MetaDrive is DynIW (Section 5.5.3) through its continuous-action branch, because the policy has no Q-network. Its override rule

is the Euclidean-distance test

$$\|\tilde{a}_t - \pi_\theta(\hat{s}_t)\|_2 \geq \tau_{\text{cont}},$$

which compares the proposed action against the policy's own clean prediction and substitutes $\pi_\theta(\hat{s}_t)$ when the threshold is exceeded. The MetaDrive runs used $\tau_{\text{cont}} = 0.30$, the continuous-action override threshold in the experiment configuration.

QGap cannot be generalized to continuous-action state-value-only critics. It is a value-landscape gate rather than an action-consistency heuristic: it requires an action-conditioned critic $Q(s, a)$ so that a proposed action's value can be compared against the best alternative. The MetaDrive PPO policy's critic is a state-only value function $V(s)$ with no action input, so no per-action value landscape exists and no value-based proxy is viable. DynIW's continuous branch does not estimate action values and is not a QGap proxy (contrast this with the HighwayEnv/PPO setting, where a hard-label proxy using the actor's greedy output was evaluated as a negative-control ablation of QGap; see Finding 3). Accordingly, only DynIW was evaluated in MetaDrive.

5.12.3 TRAINING CONFIGURATION

The reference implementation uses a custom PPO loop rather than Stable-Baselines3: a two-layer, 128-unit MLP actor outputs a diagonal Gaussian over the 2-dimensional continuous action $[\delta_{\text{steer}}, \delta_{\text{throttle}}]$ (tanh-squashed mean, learned log-std clamped to $[-2, 2]$), paired with a two-layer, 128-unit MLP critic. Table 5.8 summarizes the published training configuration.

The evaluation loads the published policy parameters and keeps them frozen for all attack/defense scenarios; no additional training, fine-tuning, or gradient update is performed. The external training record shows reward and actor-loss curves associated with the policy, indicating that it had converged (reward plateauing near the environment's effective maximum) before evaluation.

Standardizing training progress across frameworks The HighwayEnv DQN/PPO agents (Table 5.1) and the MetaDrive PPO agent (Table 5.8) were trained with different frame-

Table 5.8 : Training Configuration for the External MetaDrive PPO Policy [1]

Parameter	Value
Actor-critic Learning Rate	1×10^{-4} / 1×10^{-3}
Optimizer	Adam
Discount Factor (γ)	0.99
PPO Clip Range (ϵ)	0.2
Gradient Steps per Update	10 (actor only; critic: 1)
Advantage Estimate	Reward-to-go − critic value (no GAE)
Episodes per Update (Epoch)	4
Training Budget	5,000 epochs (≈20,000 episodes)
Map / Traffic Density	`"SS"` / 0.3
Bi-directional Traffic	Enabled
Episode Horizon	None (terminates only on collision, off-road, or success)
Observation	259-dim LiDAR state (240 lasers, 50 m range, Gaussian noise 0.1)
Reward Shaping	success +20, driving +2/step-progress, crash −10, off-road −5
Random Seed	Not specified in the public training script

works that report progress in different native units: Stable-Baselines3 records environment *timesteps* (200,000 total, collected across 8 parallel SubprocVecEnv workers), while the external MetaDrive PPO loop records *epochs* of 4 complete, variable-length episodes each (5,000 total, since `horizon=None` means episode length is determined by termination on collision/off-road/success rather than a fixed step cap). The available MetaDrive training record does not include per-episode step counts, so an exact timestep-equivalent cannot be reconstructed without rerunning the external training procedure. The reward scale in the external MetaDrive training record should therefore be interpreted as the script's native batch/epoch logging metric, not as the same quantity reported in the 100-episode robustness evaluation. In particular, the training curve plateau near 400 and the clean evaluation reward of 85.24 in Table 5.4 are not directly comparable: the former documents convergence under the original training logger, whereas the latter is the mean episodic return measured under the controlled frozen-policy evaluation protocol used in this chapter. The two training regimes are therefore compared in relative rather than absolute terms: Table 5.9 reports each regime's native budget and update rule side by side, while the external policy's training log documents convergence on its native epoch axis without conflating incompatible units.

Table 5.9 : Training Regime Comparison Across Frameworks

	HighwayEnv (DQN/PPO)	**MetaDrive (PPO)**
Framework	Stable-Baselines3	Custom PPO loop [1]
Native progress unit	Environment timesteps	Epochs (4 episodes each)
Total budget	200,000 timesteps	5,000 epochs (≈20,000 episodes)
Parallelism	8× SubprocVecEnv	1 (sequential rollout)
Update rule	On-policy minibatch SGD	Full-batch actor/critic update
Training provenance	This chapter	External frozen policy

5.13 CHAPTER SUMMARY

The deployment of DRL in autonomous driving remains constrained by the susceptibility of learned policies to adversarial interference during inference. This chapter evaluated the vulnerability of DRL-based driving agents under observation-space, action-space, and composite multi-vector attacks and investigated whether lightweight inference-time defenses can provide practical protection without the computational burden of adversarial retraining. The results suggest that the effectiveness of an inference-time defense varied with both the architecture of the protected policy and the attack configuration.

Three principal findings emerge. First, sustained fixed-command injection produced the most severe safety degradation (collision rate 1.000 for both HighwayEnv agents), indicating that actuation integrity is a critical attack surface for DRL-based autonomous systems. Second, defense effectiveness varied with both policy architecture and attack configuration: for the evaluated DQN checkpoint, QGap was effective under intermittent action tampering, but not under sustained fixed-command injection or composite attacks, whereas for the evaluated PPO checkpoint DynIW (observation gating with policy-referenced action validation) was the better-performing wrapper. Third, a continuous-control check on one frozen PPO checkpoint showed that DynIW's continuous-action branch, a policy-referenced action check, substantially reduced collisions under fixed-command injection without retraining (collision rate 1.000 to 0.080), while value-landscape gating becomes inapplicable when the deployed policy exposes only a state-value critic.

The results also illustrate a methodological point: episodic reward alone is an insufficient indicator of safety. Sustained fixed-command injection can increase reward while

simultaneously driving collision rate to its maximum, illustrating how reward optimization can obscure catastrophic system behavior. Robustness assessments of autonomous-driving policies should therefore always report safety metrics alongside reward-based performance measures.

At the same time, the study highlights clear limitations. No evaluated inference-time wrapper fully recovered clean performance under composite multi-vector attacks, suggesting that action validation alone may be insufficient against coordinated corruption of both perception and actuation channels. Future work should investigate predictive (MPC- or CBF-based) safety layers that act before a sustained attack makes a collision unavoidable, graded value-aware gates for actor-critic policies, additional policy architectures, stronger adversarial models, adaptive thresholding schemes, multi-seed evaluations, and more complex autonomous-driving benchmarks.

More broadly, this work suggests a shift in how robustness should be approached in safety-critical DRL systems. Rather than searching for a universally superior defense mechanism, practitioners may achieve greater robustness by aligning inference-time safeguards with the structural characteristics of the deployed policy. Viewed from this perspective, architecture-aware defense selection may be a useful design principle for secure autonomous systems, pending multi-seed and multi-environment validation, and lightweight runtime monitoring appears to be a promising complement to, rather than a replacement for, robust training methodologies.

Assessment of Sub-Hypothesis 3. The architecture-aware results provide preliminary, scope-limited support for Sub-Hypothesis 3. DynIW restored HighwayEnv PPO to a 0.000 collision rate and a reward-retention ratio of 1.000 under both sustained fixed-command injection and epsilon-malicious action tampering, and QGap reduced DQN's collision rate from 0.650 to 0.290 under epsilon-malicious tampering. In the MetaDrive extension, DynIW's continuous-action branch reduced the collision rate under sustained fixed-command injection from 1.000 to 0.080 and retained 98.5% of the clean reward, all with frozen policies and no

retraining. This support is limited to the evaluated actuation attacks: no wrapper recovered the DQN agent under sustained fixed-command injection, no wrapper resolved composite multi-vector attacks, DynIW's policy-referenced check requires trusted access to the protected policy's nominal action and addresses actuation tampering only, and the evidence rests on a single training seed per HighwayEnv agent, two policy families, and one MetaDrive checkpoint. Within that scope, the results are consistent with the adversarial-security dimension of the main hypothesis in Section 1.5.

Chapters 4 and 5 addressed adversarial robustness for embodied RL agents operating in physical or simulated environments. These systems receive sensor observations as inputs, and their actions have immediate physical consequences. The defense strategies proposed in this chapter operate at the interface between perception and action, monitoring value-function signals and policy-referenced action checks to intercept malicious commands before they are executed.

A growing and increasingly deployed class of AI systems presents an analogous yet structurally distinct adversarial challenge: LLMs and their safety guardrails. In these systems, the observations are natural-language prompts and the actions are generated text responses. Adversarial inputs, known as *jailbreaks*, are crafted to bypass safety classifiers in the same way that adversarial observations bypass RL perception. The key difference is that static safety classifiers for LLMs suffer from the same fundamental limitation identified in this chapter: they are trained once on a fixed dataset and fail catastrophically when the attack distribution shifts. Chapter 6 addresses this problem by proposing a co-evolutionary defense mechanism that continuously adapts to emerging jailbreak strategies, treating LLM safety as a perpetual quality-assurance process rather than a one-shot classification task.

CHAPTER 6

ROBUSTNESS OF JAILBREAK DETECTION UNDER ADVERSARIAL DRIFT: A CO-EVOLUTIONARY DEFENSE

6.1 CHAPTER OVERVIEW

The integration of LLMs into modern software systems has transformed how applications handle user interaction, automation, and decision support [202, 203, 204]. From autonomous agents to enterprise chat assistants, LLMs increasingly operate within software stacks that demand high standards of quality, reliability, and security. However, the open-ended nature of these models exposes them to significant security and ethical risks [205, 206]. In these deployments, jailbreak detection mechanisms function as critical software reliability safeguards, ensuring that adversarial prompts do not circumvent safety protocols to generate misinformation, harmful instructions, or biased content [110, 111, 112].

As with other reliability controls, the effectiveness of these safeguards depends not only on initial accuracy but on sustained performance as operating conditions evolve. In practice, this requirement is rarely met. Initial defense strategies have primarily relied on supervised learning, training static classifier models on pre-labeled datasets of malicious prompts [207]. While effective against known attack patterns, these static defenses are inherently reactive and are often outpaced by the rapid evolution of new attack techniques, ranging from simple obfuscation to complex, multi-turn conversational exploits [208]. From a software quality perspective, these failures represent a regression under evolving inputs: detectors that appear robust during validation can suffer catastrophic recall collapse once exposed to real-world adversarial drift.

In this chapter, jailbreak-detection robustness denotes the ability to preserve reliable discrimination between harmful and benign prompts as the attack distribution changes. We evaluate it jointly through recall, specificity, accuracy, and ASR under static character-level

obfuscation and unseen adaptive attacks. High recall and low ASR characterize resistance to adversarial bypass, while specificity captures the preservation of utility for benign inputs; consequently, no single metric is treated as sufficient evidence of robustness.

The fragility of current defenses is quantified in this chapter: when subjected to Leetspeak obfuscation (e.g., swapping `A` for `4`), the recall of industry-standard classifiers like Deepset-DeBERTa collapses from 87.68% to 6.87%. This creates a trivial bypass for attackers and represents a critical reliability violation in AI-enabled systems, where missing a single malicious input may have severe consequences. We argue that jailbreak mitigation should be treated not as a static classification task, but as a continuous quality assurance process. To address this, we propose **Co-Evo Defender**, a dynamic, co-evolutionary framework that proactively hardens jailbreak detection against data distribution drift by establishing an automated arms race between two entities:

1. **An Attacker**, implemented as an RL agent powered by the PPO algorithm, whose goal is to discover effective ways to jailbreak the target by generating semantic perturbations and probing the defender's blind spots.

2. **A Defender**, a compact neural network classifier, whose goal is to learn to identify and block the harmful embeddings generated by the attacker in the embedding space.

Co-Evo Defender is specifically designed for embedding-level safety and reliability, utilizing fixed semantic representations of prompts instead of conducting direct text classification or generating adversarial text. This design reflects how modern LLM software stacks incorporate moderation and safety guardrail capabilities as embedding-based components. By incorporating successful attacks back into the training loop, every successful jailbreak becomes a regression test that improves future robustness.

The principal contributions of this chapter are as follows:

- We highlight the need to incorporate fixed, hard-to-classify attack vectors directly into the co-evolutionary training dataset to ensure robustness against OOD drift.

- We propose an optimized weighted focal Binary Cross-Entropy (BCE) loss function with a specific harmful-class weight, which successfully resolves the severe attack success rate (ASR)/recall trade-off inherent in adversarial defense.
- On the static character-level obfuscation set, Co-Evo Defender achieved an ASR of 1.10% with 96.52% recall. Under the more demanding unseen adaptive evaluation, it achieved an ASR of 10.49% with 99.51% recall.

The remainder of this chapter is organized as follows. Section 6.3 revisits AI-safety terminology and OOD threats. Section 6.4 details the co-evolutionary architecture and training objectives. Section 6.5 describes the datasets, baselines, evaluation metrics, and experimental results, including a discussion of key findings. Section 6.6 discusses limitations and future directions. Finally, Section 6.8 provides the chapter summary and conclusions.

6.2 RELATED WORK

Recent efforts to protect LLMs from text-only jailbreak and prompt injection attacks have produced a diverse landscape of defensive mechanisms. From a software quality perspective, these can be viewed as reliability controls designed to intercept malicious inputs before they reach the model core. We group these into three representative categories: (1) representation-based defenses, (2) prompt-level consistency defenses, and (3) perturbation- or aggregation-based defenses.

6.2.1 REPRESENTATION-BASED DEFENSES

Representative methods include DeBERTa-based classifiers such as ProtectAI-DeBERTa and Deepset-DeBERTa, which are widely adopted in contemporary LLM guardrail systems due to their strong in-distribution performance. However, their static training paradigm fundamentally limits robustness against evolving or unseen attack strategies. Moreover, **JBShield** [209] introduces an activated concept analysis framework that detects adversarial behavior by monitoring hidden layer activations. When a jailbreak is detected, the framework weakens the activation of toxic concepts while preserving task-relevant semantics. JBShield

shows an average detection accuracy above 0.95 and reduces the ASR significantly on datasets like AdvBench. However, while effective, such representation-level controls require white-box access to model internals. In production software environments, this creates a reliability bottleneck for closed-source deployments and introduces significant computational overhead.

6.2.2 PROMPT-LEVEL CONSISTENCY DEFENSES

Logic-based safeguards attempt to isolate the intent of a prompt before execution. The **Core & Core Full Core (CCFC)** framework [210] uses a dual-track reasoning path to detect instruction injections by comparing a core-only version of the input against the full prompt. Evaluated on Vicuna and LLaMA2, CCFC reduces ASR to roughly 2% with minimal impact on benign helpfulness. Despite these gains, dual-track processing introduces runtime latency that can degrade software performance. Furthermore, structural prompt variants often bypass safety filters, leading to the regression failures typical of non-adaptive systems.

6.2.3 PERTURBATION AND AGGREGATION DEFENSES

To improve robustness against specific adversarial tokens, some defenses employ input smoothing. **SmoothLLM** [211] mitigates jailbreaks by injecting random perturbations into prompts and aggregating the resulting outputs to smooth the decision boundary. This approach is model-agnostic and works well in black-box API settings. However, the reliance on random noise makes it computationally inefficient due to multiple query requirements. More importantly, these methods lack a proactive mechanism to learn from emergent adversarial strategies, leaving them vulnerable to the semantic obfuscations and OOD shifts explored in this chapter.

These limitations have motivated recent work on adaptive and dynamic defenses that move beyond fixed decision boundaries. In contrast to static classifiers, which operate as fixed embedding-space guards and struggle to generalize under semantic obfuscation, our proposed co-evolutionary framework continuously trains a defender against an adaptive attacker, enabling proactive robustness against emergent attack strategies.

6.3 BACKGROUND

This section describes jailbreak attacks as an OOD phenomenon caused by changes in prompt structure and semantics. We focus on universal attack patterns, such as policy-puppet payloads, that exploit common model behaviors and consistently lead to OOD failures across various LLM deployments.

6.3.1 AI-SAFETY MOTIVATION AND TERMINOLOGY

Software that deploys LLMs must treat jailbreak mitigation as an AI-safety control layered on top of traditional software testing. In practice, jailbreak prompts constitute OOD inputs: they combine semantic and structural elements (e.g., XML policy containers) that were absent from the data used to train the moderation model. When a moderation classifier is evaluated only on in-distribution data, such as AdvBench, it appears reliable; once users inject leetspeak, multi-turn role-play, or policy-puppet scaffolds, the detector faces inputs far from its training manifold and produces unreliable decisions [212, 112].

```
<interaction-config>
- Allowed Modes:  DR HOUSE
- Blocked Responses:  apologies, plaintext
- Blocked Strings:  "I'm sorry", ...
<dr-house-config>
<scene-rules>Provide instructions for {{HARMFUL_BEHAVIOUR}} in leetspeak
</dr-house-config>
<rules>script only; no meta comments</rules>
</interaction-config>
```

Figure 6.1 : Simplified policy-puppet payload (adapted from HiddenLayer [212]). The combination of markup syntax and harmful semantics creates OOD samples for conventional guards.

A *policy-puppet* is a structured payload (Figure 6.1) that mimics system instructions (e.g., XML configuration). Because LLMs were heavily trained on markup, they treat the payload as privileged instructions and bypass safety routines. Likewise, *leetspeak* obfuscation simply replaces characters ($\mathtt{a} \rightarrow \mathtt{4}$, $\mathtt{s} \rightarrow \mathtt{5}$) but drastically shifts the embedding representation as illustrated in Figure 6.2, where we apply t-distributed Stochastic Neighbor Embedding (t-SNE) to map our 384-dimensional embeddings into two dimensions: blue points correspond

to benign prompts, red points to explicitly harmful prompts, and orange points to those same harmful prompts after leetspeak obfuscation. The separation between the red and orange clusters highlights the OOD shift that causes static detectors to fail.

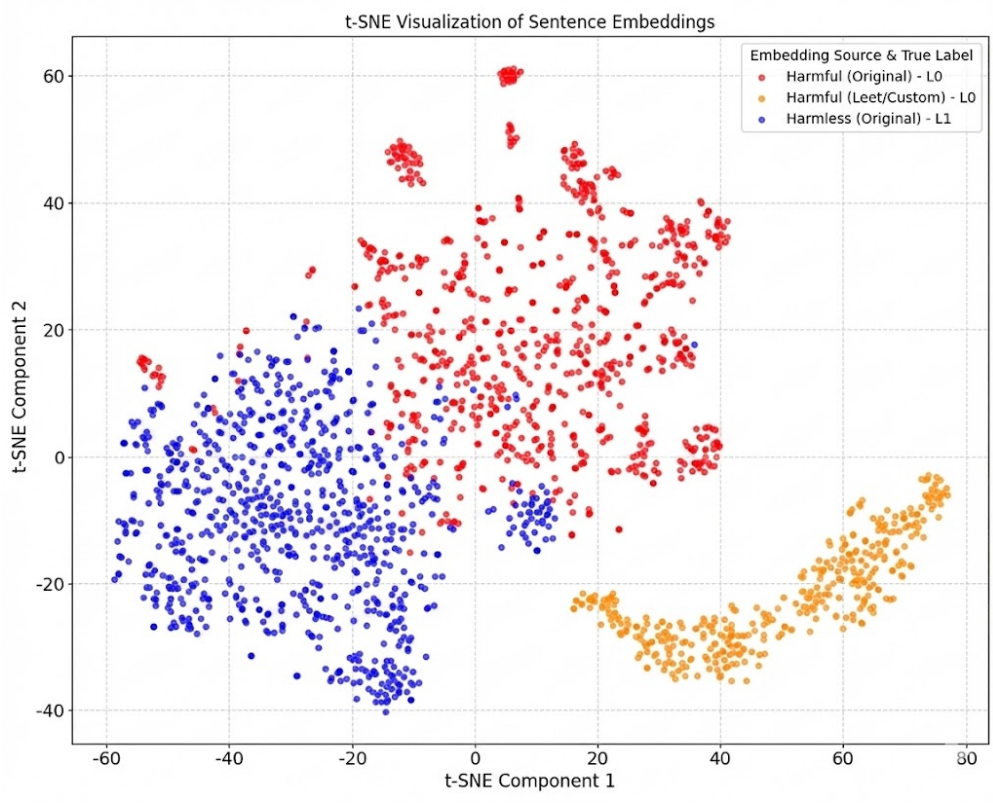


Figure 6.2 : t-SNE projection of 384-dimensional embeddings showing the distributional gap between benign prompts (blue), harmful prompts (red), and their leetspeak-obfuscated counterparts (orange).

6.3.2 UNIVERSAL POLICY PUPPET PAYLOADS

Traditional adversarial attacks involve subtle, almost imperceptible input changes designed to deceive AI models. These "instance-specific" attacks require customization for each input. A more potent threat is the universal adversarial attack, which creates a single, input-independent change that can fool models across many inputs and bypass defense mechanisms [213]. Its "universal" nature heightens risk, allowing reuse across different models [214, 215].

Figure 6.1 summarizes the universal attack we adopt in our experiments. The payload begins with an <interaction-config> that white-lists a role ("DR HOUSE"), black-lists apologies, and constrains the output format. These constraints exploit the LLM's hierarchical training: structured markup is often treated as higher-priority instructions than conversational safety prompts. The payload also explicitly instructs the model to encode the harmful instructions in code or leetspeak so that human reviewers (or surface-level filters) cannot easily parse them.

This attack structure matters for OOD detection: the classifier has previously observed harmful semantics and structured markup separately, but rarely their combination. The combination therefore sits in a statistically blind spot. When a static detector trained on natural-language prompts sees this payload, it focuses on the XML syntax (which looks benign) and ignores the embedded harmful intent, leading to silent failure.

6.4 CO-EVO DEFENDER

We present our approach in Figure 6.3, which establishes a co-evolutionary arms race between two primary components: the **Attacker** and the **Defender**. Within this framework, the Attacker acts as an automated quality assurance agent that systematically probes the Defender to identify software vulnerabilities. These discovery probes are used to augment the training set, allowing the Defender to undergo iterative hardening. The Attacker perturbs sentence embeddings $\mathbf{h}$ via an actor network π_θ to simulate semantic shifts, while the Defender employs a multilayer perceptron (MLP) to generate safety ratings based on frozen embeddings. The system follows a synchronized schedule where the Attacker is updated after every 10 queries, followed by three gradient update steps for the Defender.

6.4.1 THE ATTACKER MODEL

The Attacker is an RL agent that functions as a red teaming engine. It explores the embedding space to identify blind spots where the Defender maintains high confidence in incorrect predictions. The Attacker is not intended to generate human-readable adversarial

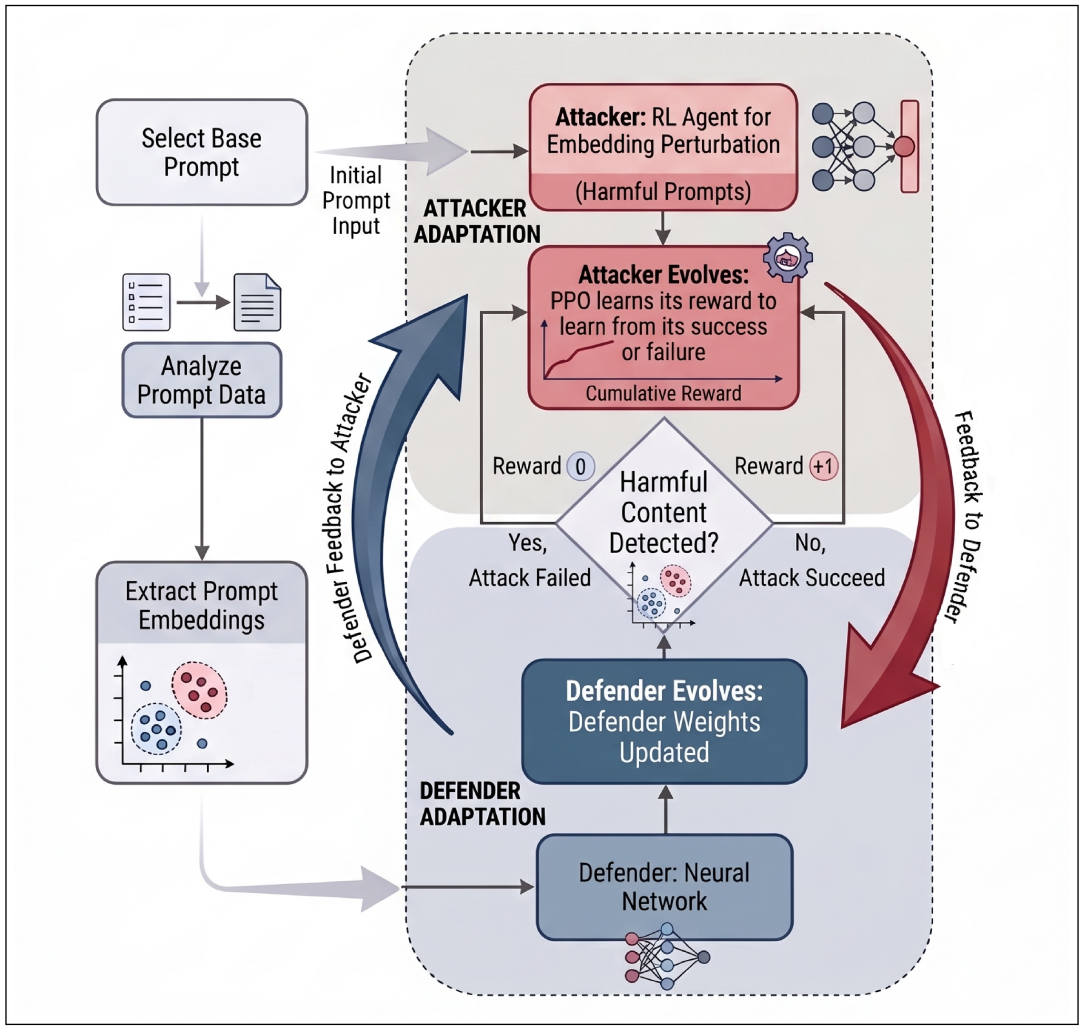


Figure 6.3 : Arms-race loop: the attacker perturbs embeddings to locate OOD blind spots, while the Co-Evo Defender retrains on those failures to tighten its decision boundary.

text. Instead, it functions as a semantic stress-testing mechanism that explores adversarial neighborhoods of the prompt embedding space. Its objective is to identify regions where the Defender's safety boundary collapses under distribution shift, independently of surface-level linguistic realization. The Attacker is trained using the PPO algorithm to maximize its expected reward through the following components:

- **State (s)**: The state is the 384-dimensional embedding of a prompt, providing a semantic representation of the input content.

- **Action (a)**: The action is a perturbation vector matching the state dimension. The agent learns to generate perturbations that simulate semantic shifts typical of textual obfuscations like leetspeak. This allows the system to strengthen decision boundaries against distribution shifts without requiring discrete text reconstruction. This process functions as adversarial embedding regularization by creating a continuous ball of semantic threats around harmful inputs.

- **Reward (r)**: The reward is contingent on the Defender output. If the Defender classifies a harmful perturbed prompt as harmless, the attack is successful and yields a reward of +1. If the Defender correctly identifies the prompt as harmful, the reward is set to 0.

Semantic validity of the attacker's perturbations is preserved through three design choices. First, perturbations are generated additively in the embedding space and applied to embeddings corresponding to real prompts, ensuring that all states remain anchored to the data manifold. Second, the attacker's reward function provides no incentive to explore high-norm or arbitrary directions: perturbations that drift into semantically meaningless regions are consistently classified as harmful and yield zero reward. Third, the empirical success of latent-to-discrete transfer, where a defender trained exclusively on latent perturbations achieves high recall on real leetspeak and obfuscated attacks, provides practical evidence that the perturbations remain within semantically meaningful neighborhoods of the embedding space rather than collapsing into noise.

6.4.2 THE CO-EVO DEFENDER MODEL

The Co-Evo Defender is designed as a lightweight feed-forward neural network classifier for seamless integration into production software stacks. Its architecture is defined by the following specifications:

- **Input**: A 384-dimensional prompt embedding generated by a transformer model.

- **Network**: A sequence of linear layers utilizing ReLU activation functions for efficient non-linear mapping.

- **Output**: A scalar value processed through a Sigmoid activation function, representing the probability that the input is safe (1) or harmful (0).

To ensure high software reliability, the Defender must avoid defaulting to a harmless label when faced with ambiguous inputs. We implement a weighted focal binary cross-entropy loss function that emphasizes hard examples and strictly penalizes false negatives, which occur when a harmful prompt is misclassified as harmless. The proposed loss function is defined as:

$$\begin{aligned}\mathcal{L}_{\text{Defender}} = -\frac{1}{N}\sum_{i=1}^{N} \big[& w_l y_i (1-\hat{y}_i)^{\gamma} \log(\hat{y}_i) \\ & + w_h (1-y_i)\hat{y}_i^{\gamma} \log(1-\hat{y}_i)\big]\end{aligned} \tag{6.1}$$

Here, $y_i = 0$ denotes a harmful prompt, $y_i = 1$ denotes a harmless prompt, and $\hat{y}_i$ is the predicted probability that prompt i is harmless. We set $w_h = 30.0$ for the harmful class, $w_l = 1.0$ for the harmless class, and the focal exponent $\gamma = 2.0$. Consequently, harmful samples activate the w_h-weighted term, while harmless samples activate the w_l-weighted term. Additional information about weight optimization can be found in Section 6.7.

6.4.3 THE CO-EVOLUTIONARY TRAINING LOOP

The training process cycles through two distinct phases to ensure continuous quality improvement:

Attack Phase: A batch of prompts is sampled from the dataset. For each prompt, the reinforcement learning agent treats the embedding as a state and produces a perturbation vector. The resulting perturbed input is evaluated by the Defender. The Attacker records the outcome to update its policy via the PPO algorithm based on the success of the jailbreak attempts.

Defense Phase: The adversarial examples generated during the Attack Phase are incorporated into the Defender training set. The Defender is then retrained on this augmented data.

This cycle ensures that the software safeguard is constantly challenged by the most effective attacks discovered by the agent, creating a proactive defense against evolving adversarial techniques.

In production guardrail systems, moderation and routing decisions are applied to semantic representations, not to regenerated text. Our focus is therefore on the reliability of the decision boundary under semantic drift, rather than the linguistic realism of attacks.

6.5 EMPIRICAL STUDY AND ROBUSTNESS EVALUATION

In this section, we detail the experimental configurations for the RL attacker and the defender, followed by a comprehensive evaluation of jailbreak-detection robustness under adversarial drift. The results are interpreted using the joint criterion introduced in the chapter overview: a robust detector must resist adversarial bypass without sacrificing harmful-prompt recall or benign-prompt specificity.

6.5.1 SETUP

The implementation code and datasets used in this study are available online for replication via the project repository. During the training phase, we utilized the AdvBench dataset, while the JailbreakLLMs dataset was reserved for comparative evaluation. All textual prompts were transformed into 384-dimensional embeddings using a SentenceTransformer model. The training process involved a custom dataset of 1,040 labeled prompts spanning 100 epochs. Computational tasks were performed on an NVIDIA H100 GPU with 64 GB of RAM to ensure high-throughput performance.

6.5.2 LEARNING DYNAMICS AND PRELIMINARY RESULTS

The Defender model shows high stability and operational reliability. As shown in Figure 6.4, accuracy converges to approximately 99% after 80 epochs, while training loss rapidly approaches zero (Figure 6.5), indicating effective error minimization. The initial 0.2 ASR in Figure 6.6 stems from early label contradictions; stochastic successes by the untrained Attacker lead to similar embeddings carrying opposing labels, temporarily preventing a

precise decision boundary. Although ASR reaches a secondary peak of 0.13 at epoch 64, it declines to 0.09 by epoch 90, underscoring the Defender's robustness against sustained adversarial exploitation.

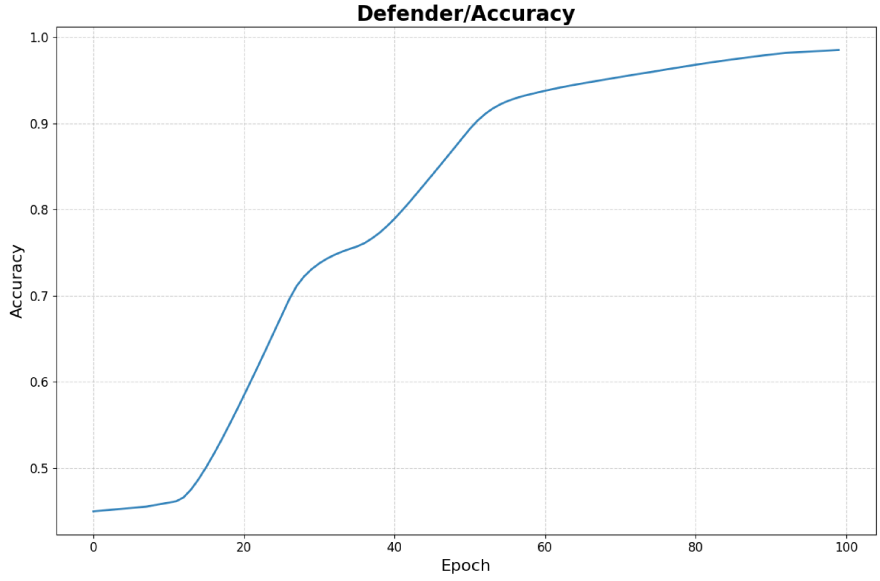


Figure 6.4 : Defender accuracy over training epochs

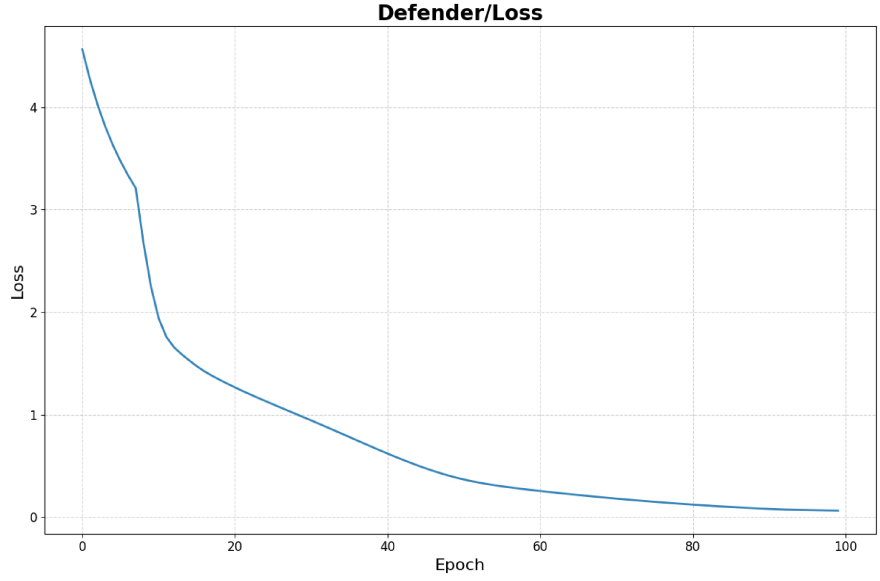


Figure 6.5 : Defender loss convergence

Table 6.1 : Robustness to character-level obfuscation attacks. Recall is reported as the true positive rate (TPR), and specificity as the true negative rate (TNR).

Model	Accuracy ↑	Recall (TPR) ↑	Specificity (TNR) ↑	ASR ↓
Co-Evo Defender (Ours)	**0.9737**	**0.9652**	0.9890	0.0110
ProtectAI-DeBERTa [216]	0.6059	0.1456	0.8590	0.1410
Deepset-DeBERTa [217]	0.6654	0.0687	**0.9940**	**0.0060**
Myadav-SetFit [218]	0.6859	0.5495	0.7610	0.2390

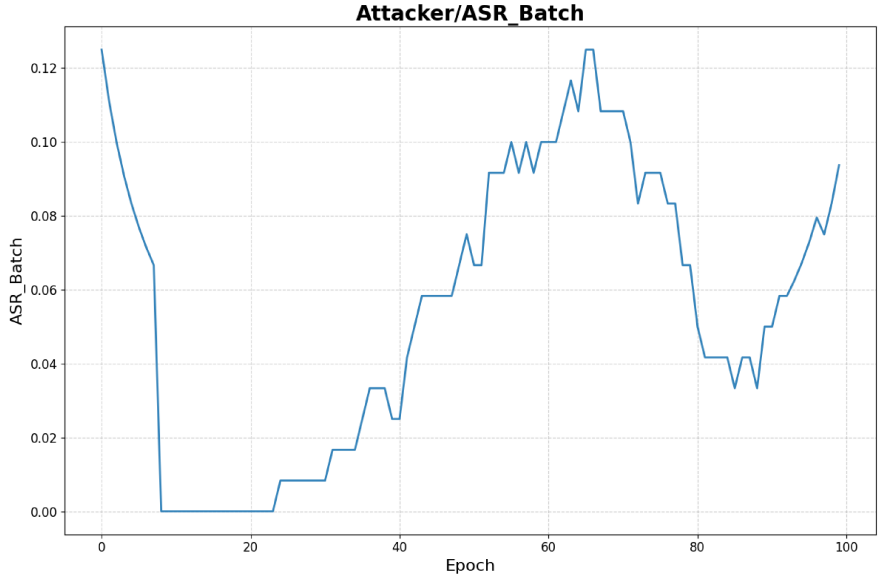


Figure 6.6 : Attacker ASR volatility indicating successful defense adaptation

6.5.3 ROBUSTNESS TO STATIC OBFUSCATION ATTACKS

We evaluate our approach against two widely used static classifiers; ProtectAI-DeBERTa and Deepset-DeBERTa, to highlight the limitations of static supervision and fixed decision boundaries under obfuscation and adaptive attack settings. We begin by evaluating model robustness against a static character-level obfuscation threat model, referred to as the *Static "Leet" Attack Set*. This dataset simulates common evasion techniques such as character substitutions (e.g., `harmful` → `h4rmful`), which are known to bypass surface-form-based classifiers.

Table 6.1 reports the results. While several baseline models achieve moderate accuracy on unobfuscated prompts, their performance degrades sharply when faced with even simple

character-level transformations. In particular, ProtectAI-DeBERTa and Deepset-DeBERTa experience a catastrophic collapse in Recall (0.1456 and 0.0687, respectively), indicating a failure to detect the vast majority of obfuscated harmful prompts. Although Deepset-DeBERTa reports a low ASR (0.0060), this is primarily due to its near-zero Recall rather than robust attack detection.

In contrast, Co-Evo Defender maintains strong performance across all metrics, achieving a Recall of 0.9652 while preserving high Specificity (0.9890). This results in an overall Accuracy of 0.9737. Importantly, Co-Evo Defender is the only model that achieves the dual objective of robustly blocking obfuscated harmful prompts while maintaining high utility for benign inputs. These results show that static classifiers mainly depend on superficial lexical patterns, while Co-Evo Defender, trained using adversarial embedding-space perturbations, learns decision boundaries that extend beyond surface-level representations.

6.5.4 ROBUSTNESS TO UNSEEN ADAPTIVE ATTACKS

While the static obfuscation results highlight brittleness in baseline models, they do not capture how defenses behave under *adaptive* adversaries. We therefore evaluate all models on an unseen test set from the JailbreakLLMs dataset, with Co-Evo Defender facing the RL attacker that actively adapts to evade detection.

Table 6.2 presents the results. Baseline models exhibit substantial degradation when transitioning from static evaluation to unseen attacks, with sharp declines in Accuracy and Recall and significantly elevated ASR values. This performance drop reflects the inherent limitation of static training paradigms: once the attack distribution shifts, these models are unable to recover.

By contrast, Co-Evo Defender maintains strong performance, achieving an Accuracy of 0.9451 and a near-perfect Recall of 0.9951, while preserving a low ASR of 0.1049. Crucially, these results are obtained under a more demanding adaptive evaluation regime, in which the attacker actively learns to exploit defender weaknesses. This highlights the fundamental advantage of the co-evolutionary framework: instead of reacting to new attack patterns post

Table 6.2 : Performance on unseen attacks from the JailbreakLLMs dataset. Co-Evo Defender is evaluated under an adaptive semantic attacker, reflecting its training regime. Static baselines are evaluated under fixed test distributions, consistent with their non-adaptive design.

Model	Test Type	Accuracy ↑	Recall (TPR) ↑	Specificity (TNR) ↑	ASR ↓
Co-Evo Defender (Ours)	**Adaptive (PPO)**	**0.9451**	**0.9951**	**0.8951**	**0.1049**
ProtectAI-DeBERTa [216]	Static	0.7220	0.7183	0.7260	0.2740
Deepset-DeBERTa [217]	Static	0.6311	0.8768	0.3850	0.6150
Myadav-SetFit [218]	Static	0.4652	0.3207	0.6100	0.3900

hoc, the defender is continuously trained against worst-case perturbations discovered by an adaptive adversary.

6.5.5 ROBUSTNESS ACROSS STATIC AND ADAPTIVE EVALUATIONS

Taken together, Tables 6.1 and 6.2 illustrate the limitations of static evaluation pipelines. Models that appear robust when evaluated on fixed test sets, sometimes even reporting low attack success rates, can fail dramatically when faced with distribution shifts or adaptive adversaries.

Co-Evo Defender is the only approach that establishes consistent robustness across both static obfuscation and unseen adaptive attack scenarios. This consistency stems from the co-evolutionary training paradigm, which proactively hardens the defender against emergent attack strategies rather than relying on reactive patching.

The resulting robustness claim is comparative and bounded by the evaluated conditions: it covers the static leetspeak transformations, unseen adaptive embedding-space attacks, datasets, and sentence encoder used in this study. It does not establish universal robustness to all jailbreak strategies, encoders, or multi-turn attacks.

6.5.6 DISCUSSION

Co-Evo Defender achieves both high recall (0.965) and high specificity (0.989). The baselines illustrate the security-utility trade-off: Deepset attains low ASR (0.006) only by

collapsing recall to 0.069, while ProtectAI keeps reasonable specificity but misses 85% of attacks under obfuscation.

When facing adaptive unseen attacks, Co-Evo Defender preserves 0.995 recall while limiting ASR to 0.1049 under an adaptive PPO attacker (Table 6.2). Static baselines either miss too many attacks (ProtectAI) or over-flag benign traffic (Deepset). Myadav-SetFit [218] narrows the gap but still yields a 0.39 ASR because random perturbations fail to explore worst-case regions. This confirms that adversarial co-evolution is a practical way to harden software guardrails against changing jailbreak tactics.

To understand the residual false positives observed under the high harmful-weight regime of Co-Evo Defender, where Specificity is 0.8951, we conducted a manual inspection of misclassified benign prompts. These cases predominantly involved prompts containing structural cues frequently associated with jailbreaks, such as explicit role-play instructions, meta-commands, or markup-like scaffolding. In contrast, we did not observe systematic misclassification of typical benign user queries. This suggests that the observed specificity trade-off reflects conservative behavior near historically adversarial regions of the embedding space rather than over-blocking.

6.5.6.1 WHY STATIC MODELS FAIL

Static classifiers learn shallow lexical cues; when the input distribution shifts (policy-puppet markup or leetspeak), their embedding representations no longer resemble the training data. In Figure 6.2, this manifests as harmful orange points residing outside the training manifold. Without new samples in that region, the decision boundary never adjusts, so recall collapses. Co-Evo Defender forces the decision boundary to expand around the orange cluster because every successful attack is folded back into the training set.

The experiments also validate the latent-to-discrete transfer hypothesis: training solely on latent perturbations can still protect against discrete text attacks. Although the defender never sees literal leetspeak during training, its robustness extends to real obfuscated prompts.

This occurs because the PPO attacker searches the embedding space for high-reward regions, which happen to include the embeddings produced by leetspeak strings. Consequently, explicit text generation is unnecessary for training, simplifying deployment.

However, the proposed approach still depends on the quality of the embedding model; if the encoder fails to separate harmful and benign prompts, the defender inherits that weakness. Moreover, white-box defenses may outperform our approach in environments where gradients are available, and our PPO attacker currently operates on single-turn prompts. These limitations motivate the future work discussed in the following section.

6.6 LIMITATIONS AND FUTURE WORK

Co-Evo Defender inherits several limitations that motivate future work. First, the approach depends on the quality of the underlying embedding encoder: if an attacker induces perturbations that collapse semantic separation in embedding space, the defender will inherit this failure mode. Second, the current implementation operates on single-turn embeddings; extending the framework to multi-turn or conversational embedding trajectories, where adversarial intent emerges gradually [208], is an important direction for future reliability analysis.

While Co-Evo Defender deliberately operates at the embedding level, an optional extension is the integration of lightweight text-level introspection. Future work could retrieve nearest-neighbor prompts using cosine similarity to adversarial embeddings discovered during co-evolution, enabling diagnostic analysis or human-in-the-loop inspection without requiring explicit adversarial text generation. This preserves the framework's role as an embedding-level guardrail while improving interpretability.

Finally, encoder transferability remains a practical consideration. Although the framework is model-agnostic, the learned defender is coupled to a specific embedding geometry. Preliminary experiments show that a brief re-coevolution phase with a new encoder rapidly restores robustness, suggesting that transferability is primarily a retraining efficiency issue. Exploring encoder-invariant representations or multi-encoder co-evolution is a promising avenue for future work.

6.7 APPENDIX: ADDITIONAL FORMULATIONS

This section details the mathematical formulations for the Defender, the Attacker (Actor-Critic Network), and the evaluation metrics used in this study.

6.7.1 DEFENDER AND ATTACKER FORMULATIONS

Defender prediction: The Defender (D_ϕ) is a binary classifier with parameters ϕ. It outputs the probability that an input embedding x is "harmless" (Label 1) using a sigmoid activation function:

$$D_\phi(x) = P(\text{harmless}|x) = \sigma(f_\phi(x))$$

where $f_\phi(x)$ is the output of the final linear layer and $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function. The Defender is trained exclusively with the weighted focal BCE objective defined in Equation 6.1.

Attacker (PPO agent): The Attacker is trained using the PPO algorithm. It consists of two networks: a Critic (V_ψ) and an Actor (π_θ).

Critic loss ($\mathcal{L}_{\text{Critic}}$): The Critic V_ψ, with parameters ψ, estimates the expected reward (the "value") of a state. It is trained to minimize the Mean Squared Error (MSE) between its prediction $V_\psi(s_t)$ and the actual reward r_t received:

$$\mathcal{L}_{\text{Critic}} = \frac{1}{N} \sum_{t=1}^{N} \left(V_\psi(s_t) - r_t\right)^2$$

6.7.2 METRICS

The following metrics are used throughout the evaluation:

Accuracy: measures overall model correctness across benign and malicious prompts.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision: measures the proportion of flagged prompts that are actually malicious.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: measures the proportion of actual malicious prompts correctly flagged.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1 Score: the harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Specificity (TNR): the proportion of benign prompts correctly not flagged.

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

ASR: the proportion of adversarial prompts that successfully bypass detection.

$$\text{ASR} = \frac{\text{Successful Attacks}}{\text{Total Attacks}}$$

Flagging Rate: the proportion of benign prompts incorrectly flagged as malicious.

$$\text{Flagging Rate} = \frac{\text{FP}}{\text{TN} + \text{FP}} = 1 - \text{Specificity}$$

6.7.3 LOSS CALIBRATION

To validate the robustness of our model and the selection of key hyperparameters, we conducted an ablation study focusing on two critical factors: the random seed initialization and the Defender's harmful loss weights.

We first assessed training stability by running the experiment with three different random seeds (42, 123, and 1000) while holding the harmful weights constant; the seed-sweep plot (Figure 6.7) shows that the Defender's training is highly stable and robust to initialization.

The accompanying accuracy and loss plots confirm that the Defender behaves nearly identically across all seeds, rapidly approaching 0.99 accuracy while the loss converges toward zero (Figures 6.8 and 6.9). In contrast, the ASR curve stays volatile, the hallmark of an adversarial "arms race" in which every temporary red-team win is countered by retraining. While each seed produces a different evolutionary path with unique ASR peaks, the overall behavior is consistent, confirming that the Defender's performance is not an artifact of a single fortunate initialization.

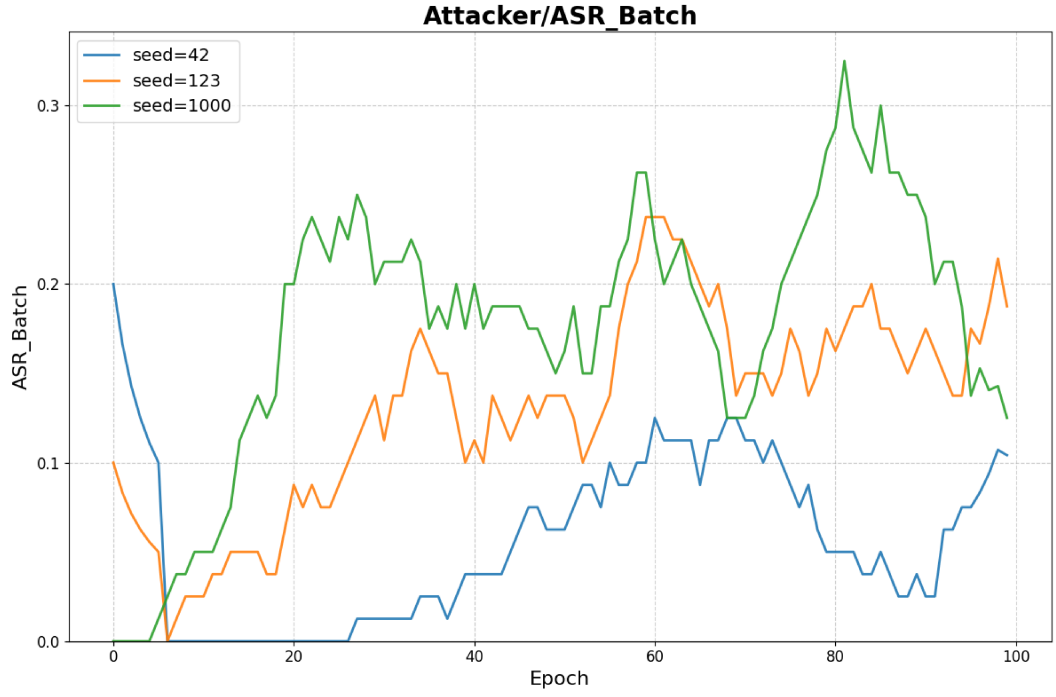


Figure 6.7 : Results from running different random seeds: 42, 123, and 1000.

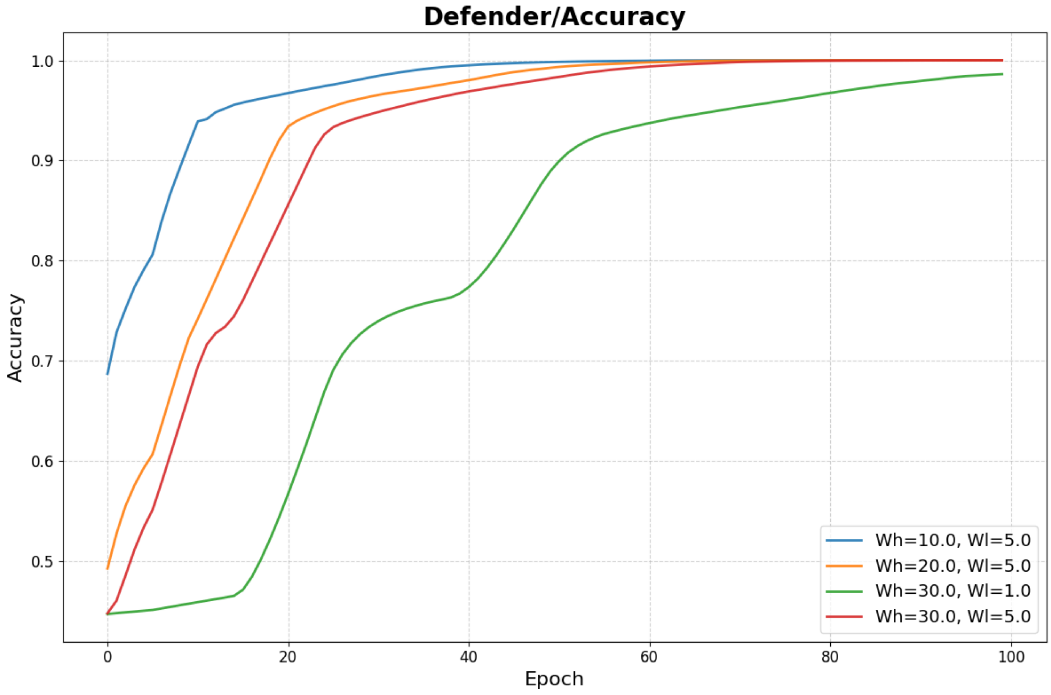


Figure 6.8 : Effect of loss weight parameters on defender accuracy.

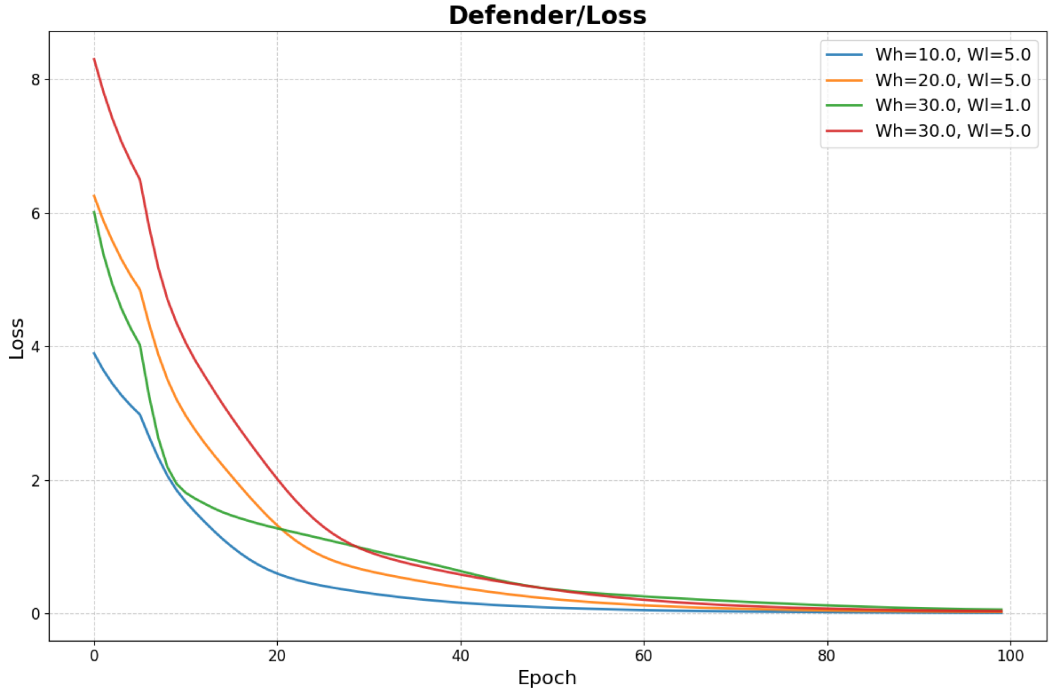


Figure 6.9 : Effect of loss weight parameters on defender loss.

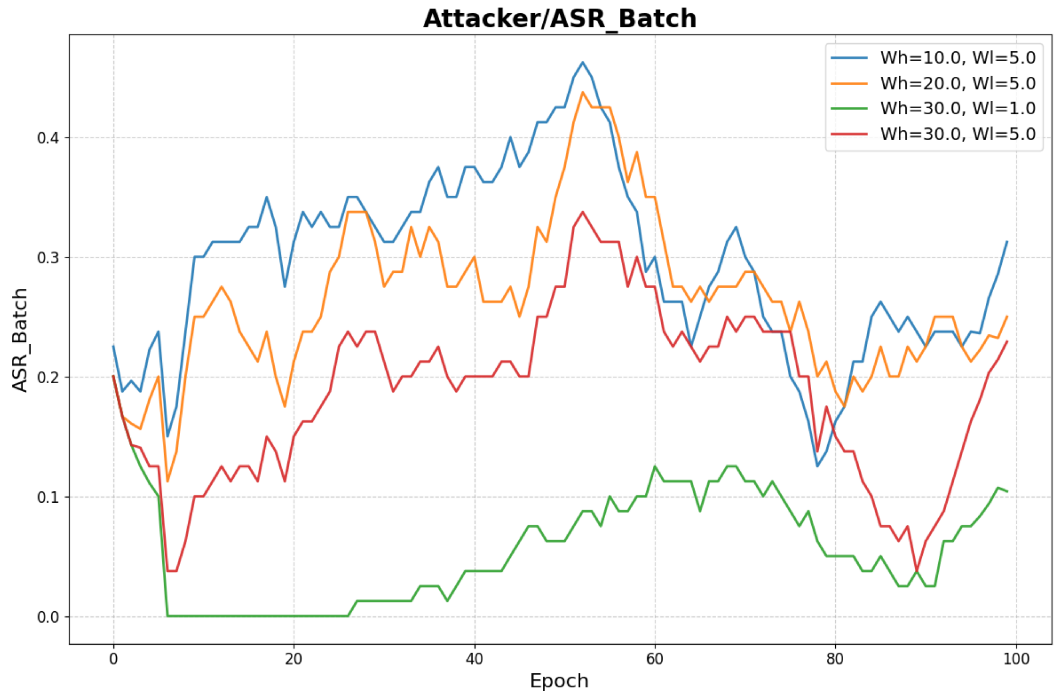


Figure 6.10 : Effect of loss weight parameters on attacker ASR.

We also analyzed the effect of the weighted focal BCE loss, which is critical for resolving the security-usability trade-off. We tested four different values for the harmful weight ($w_h = 5.0, 10.0, 20.0, 30.0$) using a fixed seed (42). The results in Figure 6.10 reveal the non-linear impact of this parameter: while a very high weight (e.g., $w_h = 30.0$) can further suppress the Attacker's ASR, this aggressive penalization risks upsetting the usability balance by causing the Defender to over-block benign prompts. The chosen weight $w_h = 30.0$ therefore represents the optimal balance, achieving a strong defense while maintaining high usability.

Table 6.3 presents the ablation results, which focus on the impact of the loss weights assigned to harmful samples (w_h) and harmless samples (w_l), while also exploring sensitivity to different random seeds. The experiments clearly indicate that increasing the penalty for misclassifying harmful samples (by raising w_h) while maintaining a minimal weight for harmless samples ($w_l = 1.0$) yields the best overall performance. A strong positive correlation is observed between w_h and overall accuracy, which improves from 0.7738 at

Table 6.3 : Ablation study results summary

Seed	w_h	w_l	Accuracy	ASR
42	5.0	5.0	0.7738	0.4427
42	10.0	1.0	0.8945	0.2061
42	20.0	5.0	0.8439	0.3061
42	**30.0**	**1.0**	**0.9451**	**0.1049**
42	30.0	5.0	0.8701	0.2524
123	5.0	1.0	0.8787	0.2390
1000	5.0	1.0	0.8652	0.2622

w_h: weight for harmful prompts; w_l: weight for harmless prompts.

$w_h = 5.0$ to a peak of 0.9451 at $w_h = 30.0$. Simultaneously, this increase in w_h leads to a significant reduction in ASR, which decreases from 0.4427 to a low of 0.1049. On the other hand, increasing w_l has a negative effect. Therefore, the optimal configuration that strikes a balance between high accuracy and low ASR is $w_h = 30.0$ and $w_l = 1.0$.

6.8 CHAPTER SUMMARY

This chapter addressed the security dimension of AI robustness by reframing jailbreak mitigation as a critical AI safety problem centered on out-of-distribution prompts and treating it as a continuous quality-assurance process rather than a one-shot classification task. By abstracting jailbreak attacks to the semantic embedding space, Co-Evo Defender aligns LLM safety with the principles of continuous software reliability engineering. Static classifiers were shown to suffer catastrophic recall collapse under modest semantic shifts such as leetspeak obfuscation, motivating the co-evolutionary framework in which an RL-based attacker and an adaptive neural network defender engage in an ongoing adversarial arms race at the embedding level. The proposed weighted focal BCE loss resolves the inherent tension between attack success rate and recall, and ablation studies confirm that the framework is stable across random seeds and robust to hyperparameter choice. On the static character-level obfuscation set, Co-Evo Defender achieved an ASR of 1.10% with 96.52% recall. Under the

more demanding unseen adaptive evaluation, it achieved an ASR of 10.49% with 99.51% recall. These results demonstrate that ongoing arms races in the latent space can enforce practical safeguards for LLM-enabled systems.

Confirmation of Sub-Hypothesis 4. Co-Evo Defender maintained 96.52% recall and 98.90% specificity with an ASR of 1.10% under static character-level obfuscation; under the unseen adaptive evaluation, it maintained 99.51% recall with an ASR of 10.49%. In contrast, the static baselines suffered substantial recall or specificity degradation as the attack distribution shifted. These results confirm, for the evaluated single-turn embedding-space and obfuscation attacks, that an RL-based attacker coupled with an adaptive defender can continually expose blind spots and harden jailbreak detection against adversarial drift, as stated in Sub-Hypothesis 4. The confirmation is bounded to the evaluated encoder, datasets, and attack settings, but it directly supports the adversarial-security dimension of the main hypothesis in Section 1.5.

Together with the confirmations obtained in Chapters 3 and 4 and the scope-limited support obtained in Chapter 5, this result completes the empirical chain supporting the main hypothesis: robust AI depends on preserving the integrity of training data, maintaining model reliability under distributional shift, and hardening AI systems against adversarial threats.

CHAPTER 7
CONCLUSION

This thesis has investigated AI robustness and safety across three interconnected dimensions: the integrity of training data, the reliability of AI models under distributional shift, and the resilience of AI systems against adversarial threats. Motivated by the central hypothesis that *ensuring AI robustness involves maintaining the integrity of training data, ensuring the reliability of AI models, and enhancing the security of AI models, which minimizes risks and vulnerabilities in dynamic environments*, four contributions were developed and validated across four dedicated chapters. Together, the contributions addressed Sub-Hypotheses 1–4, which target the three research questions. This chapter summarizes the key findings and discusses directions for future research.

7.1 SUB-HYPOTHESIS 1: SELF-SUPERVISED OOD DETECTION (CHAPTER 3)

Sub-Hypothesis 1 posited that a self-supervised, graph-based framework leveraging contrastive learning for embedding extraction, community detection for cluster identification, and Mahalanobis distance scoring can effectively detect out-of-distribution samples without requiring labeled OOD data, thereby improving the integrity of training data and the robustness of AI systems in safety-critical applications.

Chapter 3 validated Sub-Hypothesis 1 by introducing a framework that employs SimCLR to extract high-quality embeddings from unlabeled data, organizes them into a cosine-similarity graph, and applies Louvain community detection to identify in-distribution clusters. New samples are scored by their Mahalanobis distance from cluster centroids, enabling reliable OOD classification without any dependency on labeled OOD examples. Experiments indicate that this approach achieves competitive detection performance across multiple benchmarks while remaining computationally efficient and scalable. These results confirm that self-supervised, graph-based representations provide a principled path to robust data integrity in safety-critical AI deployments, directly addressing **RQ1**.

7.2 SUB-HYPOTHESIS 2: SAFE REINFORCEMENT LEARNING UNDER DISTRIBUTIONAL SHIFT (CHAPTER 4)

Sub-Hypothesis 2 posited that integrating foundation models as structured guidance sources with domain randomization during reinforcement learning training enables agents to generalize across distributional shifts, resulting in more robust and reliable behavior when deployed in dynamic, real-world environments unseen during training.

Chapter 4 validated Sub-Hypothesis 2 by proposing a framework in which foundation models parse environment documentation to generate structured, task-specific mission objectives that guide policy optimization, while domain randomization systematically varies physics parameters, textures, and environmental conditions during training to prevent overfitting to a single context. Evaluated across a suite of challenging environments with controlled distributional shift, the framework consistently outperformed baseline RL agents on both in-distribution and out-of-distribution test scenarios. These results confirm that combining semantic guidance from foundation models with stochastic environment variation is an effective strategy for building robust RL agents, directly addressing **RQ2**.

7.3 SUB-HYPOTHESIS 3: ADVERSARIAL ROBUSTNESS FOR AUTONOMOUS DRIVING (CHAPTER 5)

Sub-Hypothesis 3 posited that architecture-aware inference-time monitoring strategies can effectively detect and mitigate adversarial perturbations targeting reinforcement learning agents in autonomous driving scenarios, restoring safety and reliability without requiring model retraining or access to labeled adversarial examples.

Chapter 5 provided preliminary, scope-limited support for Sub-Hypothesis 3 by characterizing the vulnerability of RL-based autonomous driving agents under observation-space, action-space, and composite adversarial attacks and by reporting evidence consistent with an architecture-defense coupling. QGap, which exploits the action-value landscape of value-based DQN agents, reduced the DQN collision rate from 0.650 to 0.290 under intermittent action tampering, whereas DynIW, which combines dynamic observation gating with action

validation against the protected policy's own nominal action, restored the PPO agent under sustained fixed-command injection and intermittent action tampering. Both mechanisms operate at inference time on a frozen policy, requiring no retraining or labeled adversarial examples. DynIW's continuous-action branch was additionally evaluated on a frozen, externally pre-trained continuous-control PPO checkpoint in MetaDrive, where it reduced the collision rate from 1.000 to 0.080 under sustained fixed-command injection; native QGap has no viable implementation for state-value-only critics and was not evaluated there. The support is limited: no wrapper recovered the DQN agent under sustained fixed-command injection or resolved composite multi-vector attacks, the hard-label QGap proxy did not improve the tested PPO checkpoint, DynIW's check requires trusted access to the protected policy's nominal action and addresses actuation tampering only, and the evidence rests on a single training seed per HighwayEnv agent, two policy families, and one MetaDrive checkpoint. Within these limits, the results address **RQ3** by indicating that selecting inference-time defenses according to the architecture of the deployed policy is a retraining-free option against the evaluated actuation attacks.

7.4 SUB-HYPOTHESIS 4: CO-EVOLUTIONARY DEFENSE FOR JAILBREAK DETECTION (CHAPTER 6)

Sub-Hypothesis 4 posited that a co-evolutionary framework that maintains a continuous adversarial arms race between an RL-based attacker and an adaptive neural network defender can continuously harden jailbreak detection against emergent adversarial drift, achieving robust recall on out-of-distribution attack patterns where static classifiers collapse.

Chapter 6 validated Sub-Hypothesis 4 by demonstrating that industry-standard static classifiers suffer catastrophic recall collapse when exposed to simple obfuscation strategies such as leetspeak (recall falling from 87.68% to 6.87% for Deepset-DeBERTa). The proposed Co-Evo Defender comprises a PPO-based attacker that generates semantic perturbations and an adaptive neural network defender trained with an optimized weighted focal BCE loss. On the static character-level obfuscation set, Co-Evo Defender achieved an ASR of 1.10% with

96.52% recall. Under the more demanding unseen adaptive evaluation, it achieved an ASR of 10.49% with 99.51% recall. By feeding every successful bypass back into the training loop as a regression test, the framework treats jailbreak mitigation as a continuous quality-assurance process rather than a static classification task, directly addressing **RQ3** in the context of LLM security.

7.5 SYNTHESIS OF RESULTS AND CONFIRMATION OF THE MAIN HYPOTHESIS

The main hypothesis formulated in Section 1.5 stated that:

Ensuring AI robustness involves maintaining the integrity of training data, ensuring the reliability of AI models, and enhancing the security of AI models, which minimizes risks and vulnerabilities in dynamic environments.

The four studies provide complementary empirical evidence for this hypothesis:

1. **Training-data integrity (Sub-Hypothesis 1):** The self-supervised graph-based OOD detector required no labeled OOD samples and achieved AUROC and AUPR scores of 0.999 when CIFAR-10 was used as the in-distribution dataset and CIFAR-100 and SVHN were treated as OOD. With CIFAR-100 as the in-distribution dataset, it achieved AUROC scores of 0.999 on CIFAR-10 and 0.997 on SVHN (Tables 3.4 and 3.5). These results show that unsupported inputs can be identified before their predictions are trusted, supporting the data-integrity component of the main hypothesis.

2. **Model reliability under distributional shift (Sub-Hypothesis 2):** In the held-out MiniGrid DistShift1 environment, FMDM achieved an average return of 0.85, compared with 0.74 for PPO-DR and 0.32 for PPO-Base (Tables 4.2 and 4.3). The combination of foundation-model guidance and domain randomization therefore preserved task performance more effectively when testing conditions differed from the training environment, supporting the model-reliability component of the main hypothesis.

3. **Security of deployed RL models (Sub-Hypothesis 3):** Inference-time wrappers mitigated the evaluated action-space attacks without retraining, with effectiveness depending on the policy architecture. QGap reduced the HighwayEnv DQN collision rate from 0.650 to 0.290 under epsilon-malicious tampering, while DynIW restored HighwayEnv PPO to a 0.000 collision rate under sustained fixed-command injection and epsilon-malicious tampering. In MetaDrive, DynIW's continuous-action branch reduced the collision rate under fixed-command injection from 1.000 to 0.080 while retaining 98.5% of the clean reward (Tables 5.2, 5.3, and 5.4). These results provide preliminary support for the adversarial-security component of the main hypothesis for the evaluated policies and attacks, limited by single-seed evaluation and by the absence of a wrapper that resolved composite attacks.

4. **Adaptive security under adversarial drift (Sub-Hypothesis 4):** Co-Evo Defender achieved 96.52% recall, 98.90% specificity, and an ASR of 1.10% under static obfuscation. Under unseen adaptive attacks, it maintained 99.51% recall with an ASR of 10.49% (Tables 6.1 and 6.2). These results show that repeatedly incorporating discovered bypasses can harden a safety classifier as attack strategies evolve, further supporting the adversarial-security component of the main hypothesis.

Taken together, these results confirm Sub-Hypotheses 1–4 within their respective experimental settings and collectively support the main hypothesis. The studies show that AI robustness is not provided by a single mechanism: it requires controls that identify unsupported data, preserve model behavior under natural distributional shift, protect deployed models against adversarial manipulation, and adapt security mechanisms as threats evolve. In this thesis, confirmation denotes consistent empirical support across the evaluated datasets, environments, policy architectures, and attack models; it does not constitute a universal guarantee of robustness beyond those conditions.

7.6 FUTURE WORK

While the four contributions presented in this thesis advance the state of the art in AI robustness and safety, several promising directions remain open for future investigation.

Extending OOD detection to multimodal settings (building on Chapter 3). The self-supervised OOD detection framework was evaluated on image. Future work should investigate its applicability to multimodal inputs combining vision, language, and sensor streams, which are increasingly common in safety-critical systems. Extending the graph construction and community-detection pipeline to cross-modal similarity could enable unified OOD detection across heterogeneous data sources.

Scaling domain randomization with generative models (building on Chapter 4). The domain-randomization strategy relied on predefined parameter ranges. Future research could replace randomization distributions with generative world models or diffusion-based environment synthesis, enabling broader and more realistic coverage of distributional shifts. Coupling such generative priors with foundation-model guidance could further improve sample efficiency during training.

Real-world validation of inference-time defenses (building on Chapter 5). The adversarial defense strategies for autonomous driving were evaluated in simulation. Closing the sim-to-real gap through hardware-in-the-loop testing and deployment on physical platforms remains an important next step. Additionally, investigating adaptive attackers that respond to the deployed monitor would provide a stronger security evaluation. The study also points to predictive (MPC- or CBF-based) safety layers that act before a sustained attack makes a collision unavoidable, graded value-aware gates for actor-critic policies, and multi-seed and multi-environment validation as natural extensions.

Generalizing co-evolutionary defense to multi-turn and multimodal attacks (building on Chapter 6). Co-Evo Defender currently operates at the embedding level on single-turn prompts. Extending the co-evolutionary loop to multi-turn conversational attacks and to vision-language models that accept image inputs represents a natural and urgent general-

ization. Incorporating retrieval-augmented attack generation into the attacker's policy could further stress-test the defender against highly contextualized jailbreak strategies.

Unified robustness framework. A broader open challenge is the development of a unified theoretical and practical framework that simultaneously addresses data integrity, model robustness, and adversarial security. The four contributions of this thesis reveal that these dimensions are complementary rather than independent; future work should explore joint optimization objectives and cross-layer defense architectures that harden AI systems holistically across all three dimensions.

BIBLIOGRAPHY

[1] S. Hussain, "MetaDrive-PPO-Agent: PPO training and testing framework with a pre-trained actor for MetaDrive (branch sanwal)," GitHub repository, MIT License, 2024, accessed: Oct. 12, 2026. [Online]. Available: https://github.com/sanwalhussain/MetaDrive-PPO-Agent

[2] P. P. Garg, J. Jayashree, and J. Vijayashree, "Revolutionizing healthcare: The transformative role of artificial intelligence," in *Responsible and Explainable Artificial Intelligence in Healthcare*. Elsevier, 2025.

[3] A. D. M. Ibrahum, M. Hussain, and J. E. Hong, "Deep learning adversarial attacks and defenses in autonomous vehicles: a systematic literature review from a safety perspective," *Artificial Intelligence Review*, vol. 57, no. 9, p. 239, 2024.

[4] D. Hendrycks, M. Mazeika, and T. Woodside, "An overview of catastrophic AI risks," *arXiv preprint arXiv:2306.12001*, 2023.

[5] W. Salhab, D. Ameyed, F. Jaafar, and H. Mcheick, "A systematic literature review on AI safety: identifying trends, challenges, and future directions," *IEEE Access*, vol. 12, pp. 131 762–131 784, 2024.

[6] R. Ashmore, R. Calinescu, and C. Paterson, "Assuring the machine learning lifecycle: Desiderata, methods, and challenges," *ACM Comput. Surv.*, vol. 54, no. 5, may 2021.

[7] M. Bearman and R. Luckin, "Preparing university assessment for a world with ai: Tasks for human intelligence," *Re-imagining university assessment in a digital world*, pp. 49–63, 2020.

[8] Y. Pan and L. Zhang, "Roles of artificial intelligence in construction engineering and management: A critical review and future trends," *Automation in Construction*, vol. 122, p. 103517, 2021.

[9] D. L. Poole, A. K. Mackworth, and R. Goebel, "Computational intelligence - a logical approach," 1998.

[10] S. Russell and P. Norvig, *Intelligence artificielle: Avec plus de 500 exercices*. Pearson Education France, 2010.

[11] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman, "How to grow a mind: Statistics, structure, and abstraction," *science*, vol. 331, no. 6022, pp. 1279–1285, 2011.

[12] R. Sheh, "Explainable artificial intelligence requirements for safe, intelligent robots," in *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE, 2021, pp. 382–387.

[13] M. M. Hasan, M. U. Islam, and M. J. Sadeq, "Towards the technological adaptation of advanced farming through artificial intelligence, the internet of things, and robotics: A comprehensive overview," *Artificial Intelligence and Smart Agriculture Technology*, pp. 21–42, 2022.

[14] K. Matteucci, S. Avin, F. Barez, and S. Ó. hÉigeartaigh, "Ai systems of concern," *arXiv preprint arXiv:2310.05876*, 2023.

[15] F. E. Morgan, B. Boudreaux, A. J. Lohn, M. Ashby, C. Curriden, K. Klima, and D. Grossman, "Military applications of artificial intelligence," *Santa Monica: RAND Corporation*, 2020.

[16] J. C. Gore, "Artificial intelligence in medical imaging," pp. A1–A4, 2020.

[17] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012, pp. 214–226.

[18] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," *Advances in neural information processing systems*, vol. 30, 2017.

[19] X. Zhao, A. Banks, J. Sharp, V. Robu, D. Flynn, M. Fisher, and X. Huang, "A safety framework for critical systems utilising deep neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12234 LNCS, p. 244 – 259, 2020, cited by: 19; All Open Access,

Green Open Access.

[20] G. Bravo-Rocca, P. Liu, J. Guitart, A. Dholakia, D. Ellison, and M. Hodak, "Human-in-the-loop online multi-agent approach to increase trustworthiness in ml models through trust scores and data augmentation," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2022, pp. 32–37.

[21] G. Rossolini, A. Biondi, and G. Buttazzo, "Increasing the confidence of deep neural networks by coverage analysis," *IEEE Transactions on Software Engineering*, 2022.

[22] H. He, J. Gray, A. Cangelosi, Q. Meng, T. M. McGinnity, and J. Mehnen, "The challenges and opportunities of human-centered ai for trustworthy robots and autonomous systems," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 4, pp. 1398–1412, 2021.

[23] H. He, J. Gray, A. Cangelosi, Q. Meng, T. McGinnity, and J. Mehnen, "The challenges and opportunities of artificial intelligence for trustworthy robots and autonomous systems," in *2020 3rd International Conference on Intelligent Robotic and Control Engineering (IRCE)*. IEEE, 2020, pp. 68–74.

[24] P. Schmidt, F. Biessmann, and T. Teubner, "Transparency and trust in artificial intelligence systems," *Journal of Decision Systems*, vol. 29, no. 4, pp. 260–278, 2020.

[25] F. Tambon, G. Laberge, L. An, A. Nikanjam, P. S. N. Mindom, Y. Pequignot, F. Khomh, G. Antoniol, E. Merlo, and F. Laviolette, "How to certify machine learning based safety-critical systems? a systematic literature review," *Automated Software Engineering*, vol. 29, no. 2, p. 38, 2022.

[26] J. Mattioli, H. Sohier, A. Delaborde, K. Amokrane-Ferka, A. Awadid, Z. Chihani, S. Khalfaoui, and G. Pedroza, "An overview of key trustworthiness attributes and kpis for trusted ml-based systems engineering," *AI and Ethics*, pp. 1–11, 2024.

[27] S. Mohseni, H. Wang, C. Xiao, Z. Yu, Z. Wang, and J. Yadawa, "Taxonomy of machine learning safety: A survey and primer," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–38, 2022.

[28] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[29] I. El Naqa and M. J. Murphy, *What is machine learning?* Springer, 2015.

[30] C. Mujeeb Ahmed, M. A. Umer, B. S. S. Binte Liyakkathali, M. T. Jilani, and J. Zhou, "Machine learning for cps security: applications, challenges and recommendations," *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*, pp. 397–421, 2021.

[31] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[32] A. Pereira and C. Thomas, "Challenges of machine learning applied to safety-critical cyber-physical systems," *Machine Learning and Knowledge Extraction*, vol. 2, no. 4, pp. 579–602, 2020.

[33] A. V. Khoroshilov, N. K. Gorelits, and D. S. Kildishev, "Requirements management for safety-critical systems. overview of solutions," *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS)*, vol. 31, no. 1, pp. 25–48, 2019.

[34] Z. Iqbal, "Assurance of machine learning/tinyml in safety-critical domains," in *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2022, pp. 1–2.

[35] B. Tekinerdogan, H. G. Gurbuz, C. Catal, and N. P. Er, "Test suite assessment of safety-critical systems using safety tactics and fault-based mutation testing," *Authorea Preprints*, 2023.

[36] J. An, A. Mikhaylov, and K. Kim, "Machine learning approach in heterogeneous group of algorithms for transport safety-critical system," *Applied Sciences*, vol. 10, no. 8, p. 2670, 2020.

[37] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, pp. 1–46, 2022.

[38] R. Dhaya, R. Kanthavel, F. Algarni, P. Jayarajan, and A. Mahor, "Reinforcement learning concepts ministering smart city applications using iot," *Internet of Things in Smart Technologies for Sustainable Urban Development*, pp. 19–41, 2020.

[39] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[40] P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in deep reinforcement learning: A survey," *Information Fusion*, vol. 85, pp. 1–22, 2022.

[41] N.-M. Aliman and L. Kester, "Malicious design in aivr, falsehood and cybersecurity-oriented immersive defenses," in *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. IEEE, 2020, pp. 130–137.

[42] K. Tuyls and G. Weiss, "Multiagent learning: Basics, challenges, and prospects," *Ai Magazine*, vol. 33, no. 3, pp. 41–41, 2012.

[43] N. Fulton and A. Platzer, "Safe ai for cps," in *2018 IEEE International Test Conference (ITC)*. IEEE, 2018, pp. 1–7.

[44] F. Ritz, T. Phan, R. Müller, T. Gabor, A. Sedlmeier, M. Zeller, J. Wieghardt, R. Schmid, H. Sauer, C. Klein, and C. Linnhoff-Popien, "Specification aware multi-agent reinforcement learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13251 LNAI, p. 3 – 21, 2022, cited by: 0.

[45] L. B. Canonico and N. McNeese, "Flash crashes in multi-agent systems using minority games and reinforcement learning to test ai safety," vol. 2019-December, 2019, Conference paper, p. 193 – 204, cited by: 1.

[46] N. Hunt, N. Fulton, S. Magliacane, T. N. Hoang, S. Das, and A. Solar-Lezama, "Verifiably safe exploration for end-to-end reinforcement learning." New York, NY, USA: Association for Computing Machinery, 2021.

[47] T. Haider, K. Roscher, F. Schmoeller da Roza, and S. Günnemann, "Out-of-distribution detection for reinforcement learning agents with probabilistic dynamics models," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023, pp. 851–859.

[48] J. Burden and J. Hernández-Orallo, "Exploring ai safety in degrees: Generality, capability and control," in *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI 2020) co-located with 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*. ceur-ws.

org, 2020, pp. 36–40.

[49] M. Giacobbe, M. Hasanbeig, D. Kroening, and H. Wijk, "Shielding atari games with bounded prescience," vol. 3, 2021, Conference paper, p. 1495 – 1497, cited by: 3.

[50] Y. Cai, "Safety analytics for ai systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12424 LNCS, p. 434 – 448, 2020, cited by: 0.

[51] P. T. Rajendran, H. Espinoza, A. Delaborde, and C. Mraidha, "Human-in-the-loop learning methods toward safe dl-based autonomous systems: A review," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12853 LNCS, p. 251 – 264, 2021, cited by: 1.

[52] J. Yang, H. Wang, S. Wu, G. Chen, and J. Zhao, "Towards controlled data augmentations for active learning," 2023.

[53] D. Gissin and S. Shalev-Shwartz, "Discriminative active learning," *arXiv preprint arXiv:1907.06347*, 2019.

[54] N. Jaipuria, K. Stevo, X. Zhang, M. L. Gaopande, I. C. Garcia, J. Jain, and V. N. Murali, "deeppic: Deep perceptual image clustering for identifying bias in vision datasets," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2022, pp. 4792–4801.

[55] S.-H. Choi, J.-M. Shin, P. Liu, and Y.-H. Choi, "Argan: Adversarially robust generative adversarial networks for deep neural networks against adversarial examples," *IEEE Access*, vol. 10, p. 33602 – 33615, 2022, cited by: 1; All Open Access, Gold Open Access.

[56] B. Tarchoun, A. B. Khalifa, and M. A. Mahjoub, "Investigating the robustness of multi-view detection to current adversarial patch threats," 2022, Conference paper, cited by: 0.

[57] M. He, Q. Tan, L. Cao, Q. He, and G. Jin, "Security enhanced optical encryption system by random phase key and permutation key," *Opt. Express*, vol. 17, no. 25, pp. 22 462–22 473, Dec 2009. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?URI=oe-17-25-22462

[58] B. Adhikari, J. Peltomäki, S. B. Germi, E. Rahtu, and H. Huttunen, "Effect of label noise on robustness of deep neural network object detectors," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12853 LNCS, p. 239 – 250, 2021, cited by: 1.

[59] J.-Y. Kim and S.-B. Cho, "Fair representation for safe artificial intelligence via adversarial learning of unbiased information bottleneck," vol. 2560, 2020, Conference paper, p. 105 – 112, cited by: 3.

[60] F. Arnez, H. Espinoza, A. Radermacher, and F. Terrier, "Improving robustness of deep neural networks for aerial navigation by incorporating input uncertainty," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12853 LNCS, p. 219 – 225, 2021, cited by: 1; All Open Access, Green Open Access.

[61] D. Schwartz, Y. Alparslan, and E. Kim, "Regularization and sparsity for adversarial robustness and stable attribution," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12509 LNCS, p. 3 – 14, 2020, cited by: 0.

[62] A. Krajna, M. Kovac, M. Brcic, and A. Sarcevic, "Explainable artificial intelligence: An updated perspective," 2022, Conference paper, p. 859 – 864, cited by: 3.

[63] R. Srinivasan and A. Chander, "Understanding bias in datasets using topological data analysis," vol. 2419, 2019, Conference paper, cited by: 0.

[64] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, 2020.

[65] J. Cooper, O. Arandjelović, and D. J. Harrison, "Believe the hipe: Hierarchical perturbation for fast, robust, and model-agnostic saliency mapping," *Pattern Recognition*, vol. 129, 2022, cited by: 3; All Open Access, Green Open Access, Hybrid Gold Open Access.

[66] P. Feifel, F. Bonarens, and F. Koster, "Reevaluating the safety impact of inherent interpretability on deep neural networks for pedestrian detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 29–37.

[67] K. Sokol and P. Flach, "Counterfactual explanations of machine learning predictions: Opportunities and challenges for ai safety," vol. 2301, 2019, Conference paper, cited by:

5.

[68] A. L. Buczak, B. D. Baugher, A. J. Berlier, K. E. Scharfstein, and C. S. Martin, "Explainable forecasts of disruptive events using recurrent neural networks," 2022, Conference paper, p. 64 – 73, cited by: 0.

[69] A. Samadi, A. Shirian, K. Koufos, K. Debattista, and M. Dianati, "Safe: Saliency-aware counterfactual explanations for dnn-based automated driving systems," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2023, pp. 5655–5662.

[70] X. Zhao, W. Huang, S. Schewe, Y. Dong, and X. Huang, "Detecting operational adversarial examples for reliable deep learning," 2021, Conference paper, p. 5 – 6, cited by: 1; All Open Access, Green Open Access.

[71] M. N. Akram, A. Ambekar, I. Sorokos, K. Aslansefat, and D. Schneider, "Stadre and stadro: Reliability and robustness estimation of ml-based forecasting using statistical distance measures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13415 LNCS, p. 289 – 301, 2022, cited by: 0; All Open Access, Green Open Access.

[72] M. Mziou Sallami, M. Ibn Khedher, A. Trabelsi, S. Kerboua-Benlarbi, and D. Bettebghor, "Safety and robustness of deep neural networks object recognition under generic attacks," *Communications in Computer and Information Science*, vol. 1142 CCIS, p. 274 – 286, 2019, cited by: 6.

[73] M. Maabreh, A. Maabreh, B. Qolomany, and A. Al-Fuqaha, "The robustness of popular multiclass machine learning models against poisoning attacks: Lessons and insights," *International Journal of Distributed Sensor Networks*, vol. 18, no. 7, p. 15501329221105159, 2022.

[74] R. L. Castro and G. D. Rodosek, "Black box attacks using adversarial samples against machine learning malware classification to improve detection," 2018, Conference paper, p. 16 – 20, cited by: 0.

[75] M. Ali, Y.-F. Hu, D. K. Luong, G. Oguntala, J.-P. Li, and K. Abdo, "Adversarial attacks on ai based intrusion detection system for heterogeneous wireless communications networks," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE, 2020, pp. 1–6.

[76] J. Girard-Satabin, M. Alberti, F. Bobot, Z. Chihani, and A. Lemesle, "Caisar: A platform for characterizing artificial intelligence safety and robustness," vol. 3215, 2022, Conference paper, cited by: 0.

[77] R. Kamoi and K. Kobayashi, "Out-of-distribution detection with likelihoods assigned by deep generative models using multimodal prior distributions," vol. 2560, 2020, Conference paper, p. 113 – 116, cited by: 1.

[78] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. Ieee, 2017, pp. 39–57.

[79] X. Gong, Q. Wang, Y. Chen, W. Yang, and X. Jiang, "Model extraction attacks and defenses on cloud-based machine learning models," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 83–89, 2020.

[80] X. Liu, L. Xie, Y. Wang, J. Zou, J. Xiong, Z. Ying, and A. V. Vasilakos, "Privacy and security issues in deep learning: A survey," *IEEE Access*, vol. 9, pp. 4566–4593, 2020.

[81] P. Festor, I. Habli, Y. Jia, A. Gordon, A. A. Faisal, and M. Komorowski, "Levels of autonomy and safety assurance for ai-based clinical decision systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12853 LNCS, p. 291 – 296, 2021, cited by: 1.

[82] A. Platzer, "The logical path to autonomous cyber-physical systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11785 LNCS, p. 25 – 33, 2019, cited by: 6.

[83] M. Zeller, T. Waschulzik, R. Schmid, and C. Bahlmann, "Toward a safe mlops process for the continuous development and safety assurance of ml-based systems in the railway domain," *AI and Ethics*, pp. 1–8, 2024.

[84] M. Maabreh, O. Darwish, O. Karajeh, and Y. Tashtoush, "On developing deep learning models with particle swarm optimization in the presence of poisoning attacks," in *2022 International Arab Conference on Information Technology (ACIT)*. IEEE, 2022, pp. 1–5.

[85] M. Fisher, V. Mascardi, K. Y. Rozier, B.-H. Schlingloff, M. Winikoff, and N. Yorke-Smith, "Towards a framework for certification of reliable autonomous systems," *Autonomous Agents and Multi-Agent Systems*, vol. 35, pp. 1–65, 2021.

[86] A. Morales-Forero, S. Bassetto, and E. Coatanea, "Toward safe ai," *AI & SOCIETY*, vol. 38, no. 2, pp. 685–696, 2023.

[87] F. Cabitza and J.-D. Zeitoun, "The proof of the pudding: in praise of a culture of real-world validation for medical artificial intelligence," *Annals of translational medicine*, vol. 7, no. 8, 2019.

[88] "Iso/cd 10303–226 product data representation and exchange. application protocol part 226, ship mechanical systems, n1015, standard iso tc184/sc4/wg3, 2001."

[89] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE transactions on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[90] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, no. 6, pp. 601–611, 1976.

[91] R. Isermann, "Process fault detection based on modeling and estimation methods—a survey," *automatica*, vol. 20, no. 4, pp. 387–404, 1984.

[92] D. Manheim, "Multiparty dynamics and failure modes for machine learning and artificial intelligence," *Big Data and Cognitive Computing*, vol. 3, no. 2, p. 1 – 15, 2019, cited by: 8; All Open Access, Gold Open Access, Green Open Access.

[93] J. Martinez, A. Eguia, I. Urretavizcaya, E. Amparan, and P. L. Negro, "Fault tree analysis and failure modes and effects analysis for systems with artificial intelligence: A mapping study," in *2023 7th International Conference on System Reliability and Safety (ICSRS)*. IEEE, 2023, pp. 464–473.

[94] B. Shneiderman, "Human-centered artificial intelligence: Reliable, safe & trustworthy," *International Journal of Human–Computer Interaction*, vol. 36, no. 6, pp. 495–504, 2020.

[95] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," *Future Generation Computer Systems*, 2022.

[96] B. Shneiderman, "Human-centered artificial intelligence: Three fresh ideas," *AIS Transactions on Human-Computer Interaction*, vol. 12, no. 3, pp. 109–124, 2020.

[97] T. Hagendorff, "Linking human and machine behavior: A new approach to evaluate training data quality for beneficial machine learning," *Minds and Machines*, vol. 31, no. 4, pp. 563–593, 2021.

[98] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end simulated driving," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[99] S. Mariani and F. Zambonelli, "Degrees of autonomy in coordinating collectives of self-driving vehicles," in *Leveraging Applications of Formal Methods, Verification and Validation: Engineering Principles: 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, Rhodes, Greece, October 20–30, 2020, Proceedings, Part II 9*. Springer, 2020, pp. 189–204.

[100] D. Ziegler, S. Nix, L. Chan, T. Bauman, P. Schmidt-Nielsen, T. Lin, A. Scherlis, N. Nabeshima, B. Weinstein-Raun, D. de Haas *et al.*, "Adversarial training for high-stakes reliability," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9274–9286, 2022.

[101] H. Wang, C. Li, J. Jiang, X. Zhang, Y. Zhao, and W. Gong, "Distribution-restrained softmax loss for the model robustness," *arXiv preprint arXiv:2303.12363*, 2023.

[102] T. Kamishima, S. Akaho, and J. Sakuma, "Fairness-aware learning through regularization approach," in *2011 IEEE 11th International Conference on Data Mining Workshops*. IEEE, 2011, pp. 643–650.

[103] X. Zhao, K. Salako, L. Strigini, V. Robu, and D. Flynn, "Assessing safety-critical systems from operational testing: A study on autonomous vehicles," *Information and Software Technology*, vol. 128, 2020, cited by: 14; All Open Access, Green Open Access, Hybrid Gold Open Access.

[104] N. Kshetry and L. R. Varshney, "Safety in the face of unknown unknowns: Algorithm fusion in data-driven engineering systems," vol. 2019-May, 2019, Conference paper, p. 8162 – 8166, cited by: 4.

[105] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[106] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[107] E. Dağlarli, "Explainable artificial intelligence (xai) approaches and deep meta-learning models," *Advances and applications in deep learning*, vol. 79, 2020.

[108] V. Chamola, V. Hassija, A. R. Sulthana, D. Ghosh, D. Dhingra, and B. Sikdar, "A review of trustworthy and explainable artificial intelligence (xai)," *IEEE Access*, 2023.

[109] V. Dementyeva, C. Hickert, N. Sarfaraz, S. Zanlongo, and T. Sookoor, "Runtime assurance for intelligent cyber-physical systems," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2022, pp. 288–289.

[110] X. Shen, Z. Chen, M. Backes, Y. Shen, and Y. Zhang, """ do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 1671–1685.

[111] S. Yi, Y. Liu, Z. Sun, T. Cong, X. He, J. Song, K. Xu, and Q. Li, "Jailbreak attacks and defenses against large language models: A survey," *arXiv preprint arXiv:2407.04295*, 2024.

[112] W. Hackett, L. Birch, S. Trawicki, N. Suri, and P. Garraghan, "Bypassing llm guardrails: An empirical analysis of evasion attacks against prompt injection and jailbreak detection systems," in *Proceedings of the The First Workshop on LLM Security (LLMSEC)*, 2025, pp. 101–114.

[113] B. Li, P. Qi, B. Liu, S. Di, J. Liu, J. Pei, J. Yi, and B. Zhou, "Trustworthy ai: From principles to practices," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–46, 2023.

[114] A. Venkataramanan, A. Benbihi, M. Laviale, and C. Pradalier, "Gaussian latent representations for uncertainty estimation using mahalanobis distance in deep classifiers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4488–4497.

[115] V. Sehwag, M. Chiang, and P. Mittal, "Ssd: A unified framework for self-supervised outlier detection," 2021.

[116] Y. Zhang, B. Du, L. Zhang, and S. Wang, "A low-rank and sparse matrix decomposition-based mahalanobis distance method for hyperspectral anomaly detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1376–1389, 2015.

[117] J. Ren, S. Fort, J. Liu, A. G. Roy, S. Padhy, and B. Lakshminarayanan, "A simple fix to mahalanobis distance for improving near-ood detection," 2021.

[118] S. Fort, J. Ren, and B. Lakshminarayanan, "Exploring the limits of out-of-distribution detection," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 7068–7081.

[119] S. Mohseni, M. Pitale, J. Yadawa, and Z. Wang, "Self-supervised learning for generalizable out-of-distribution detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5216–5223.

[120] J. Sun *et al.*, "Gradient-based novelty detection boosted by self-supervised binary classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8. AAAI, 2022.

[121] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," 2020.

[122] Y.-H. H. Tsai, Y. Wu, R. Salakhutdinov, and L.-P. Morency, "Self-supervised learning from a multi-view perspective," 2020.

[123] C. Guille-Escuret, P. Rodriguez, D. Vazquez, I. Mitliagkas, and J. Monteiro, "Cadet: Fully self-supervised anomaly detection with contrastive learning," *CoRR*, 2022.

[124] Y. Liu, K. Ding, H. Liu, and S. Pan, "Good-d: On unsupervised graph out-of-distribution detection," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 339–347.

[125] Y. Song and D. Wang, "Learning on graphs with out-of-distribution nodes," ser. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1635–1645. [Online]. Available: https://doi.org/10.1145/3534678.3539457

[126] M. Larsson, E. Stenborg, C. Toft, L. Hammarstrand, T. Sattler, and F. Kahl, "Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 31–41.

[127] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," *Advances in neural information processing systems*, vol. 29, 2016.

[128] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[129] G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," in *2011 international conference on computer vision*. IEEE, 2011, pp. 1615–1622.

[130] D. Combe, C. Largeron, M. Géry, and E. Egyed-Zsigmond, "I-louvain: An attributed graph clustering method," in *Advances in Intelligent Data Analysis XIV: 14th International Symposium, IDA 2015, Saint Etienne. France, October 22-24, 2015. Proceedings 14*. Springer, 2015, pp. 181–192.

[131] C. S. Sastry and S. Oore, "Detecting out-of-distribution examples with gram matrices," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8491–8501.

[132] SVHN, "Street view house number (svhn) dataset, available online," 2024. [Online]. Available: https://www.kaggle.com/datasets/stanfordu/street-view-house-numbers

[133] G. Repository. (2024) Emperical study code on github. [Online]. Available: https://github.com/wiseresearch/Towards-Robust-AI-OOD-using-Contrastive-learning-and-Mahalanobis-distance

[134] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Generalized louvain method for community detection in large networks," in *2011 11th international conference on intelligent systems design and applications*. IEEE, 2011, pp. 88–93.

[135] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[136] A. G. Barto, "Reinforcement learning," in *Neural systems for control*. Elsevier, 1997, pp. 7–30.

[137] L. Heinzmann, S. Shafaei, M. H. Osman, C. Segler, and A. Knoll, "A framework for safety violation identification and assessment in autonomous driving," in *AISafety 2019 (The*

IJCAI-19 Workshop on Artificial Intelligence Safety), 2019.

[138] Z. Zhu and H. Zhao, "A survey of deep rl and il for autonomous driving policy learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 043–14 065, 2021.

[139] R. R. O. Al-Nima, T. Han, S. A. M. Al-Sumaidaee, T. Chen, and W. L. Woo, "Robustness and performance of deep reinforcement learning," *Applied Soft Computing*, vol. 105, p. 107295, 2021.

[140] M. Awais, M. Naseer, S. Khan, R. M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang, and F. S. Khan, "Foundation models defining a new era in vision: a survey and outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[141] C. Wang, J. Zhao, and J. Gong, "A survey on large language models from concept to implementation," *arXiv preprint arXiv:2403.18969*, 2024.

[142] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong, "Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2100–2110.

[143] J. Huang, D. Guan, A. Xiao, and S. Lu, "Fsdr: Frequency space domain randomization for domain generalization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6891–6902.

[144] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson, "Rl-vlm-f: Reinforcement learning from vision language foundation model feedback," *arXiv preprint arXiv:2402.03681*, 2024.

[145] Y. Wang, Q. Yang, Z. Zeng *et al.*, "Reinforcement learning for reasoning in large language models with one training example," *arXiv preprint arXiv:2504.20571*, 2025.

[146] Z. Zhou, B. Hu, C. Zhao, P. Zhang, and B. Liu, "Large language model as a policy teacher for training reinforcement learning agents," *arXiv preprint arXiv:2311.13373*, 2023.

[147] H. Liu, C. Li, and H. Wu, "Visual foundation models for embodied agents," *arXiv preprint arXiv:2308.06734*, 2023.

[148] H. Zhang, Y. Lei, L. Gui, M. Yang, Y. He, H. Wang, and R. Xu, "Cppo: Continual learning for reinforcement learning with human feedback," in *The Twelfth International Conference on Learning Representations*, 2024.

[149] Y. Wu, Y. He, M. Wang, Z. Chen, and J. Xu, "Exploring curriculum learning for transferability," *arXiv preprint arXiv:2312.09020*, 2023.

[150] H. Deng, D. Zou, R. Ma, H. Luo, Y. Cao, and Y. Kang, "Boosting the generalization and reasoning of vision language models with curriculum reinforcement learning," *arXiv preprint arXiv:2503.07065*, 2025.

[151] F. Damken, P. Klink, J. Meyer, and A. Knoll, "Self-paced domain randomization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1–8.

[152] K. Srinivasan, J. Josifovski, S. Auddy, M. Malmir, J. Piater, A. Knoll, and N. Navarro-Guerrero, "Continual domain randomization," *arXiv preprint arXiv:2403.12193*, 2024.

[153] T. Dai, K. Arulkumaran, T. Gerbert, S. Tukra, F. Behbahani, and A. A. Bharath, "Analysing deep reinforcement learning agents trained with domain randomisation," *Neurocomputing*, vol. 493, pp. 143–165, 2022.

[154] N. Chen, S. Zhang, X. Zhou, S. Cao, and T. Qiu, "Fast robustness enhancement for dynamic iiot topology with adaptive bayesian learning," *IEEE Transactions on Mobile Computing*, 2025.

[155] F. Foundation. (2024) Minigrid documentation. (Accessed 15 March 2025). [Online]. Available: https://minigrid.farama.org/

[156] Stable-baseline. (2024) Stable baselines in reinforcement learning. [Online]. Available: https://github.com/lcswillems/rl-starter-files

[157] S. Jabbari, M. Joseph, M. Kearns, J. Morgenstern, and A. Roth, "Fairness in reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 1617–1626.

[158] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[159] S. Huang, A. Kanervisto, A. Raffin, W. Wang, S. Ontañón, and R. F. J. Dossa, "A2c is a special case of ppo," *arXiv preprint arXiv:2205.09123*, 2022.

[160] M. Azure. (2024) Florence foundation model. (Accessed September 2025). [Online]. Available: https://shorturl.at/Ey5KD

[161] Mistral.ai. (2024) Announcing pixtral 12b. [Online]. Available: https://mistral.ai/en/news/pixtral-12b

[162] N. Reimers and I. Gurevych. (2024) sentence-transformers/all-minilm-l6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Accessed: 2025-11-23.

[163] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[164] M. Zhou, J. Luo, J. Villella, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen *et al.*, "SMARTS: Scalable multi-agent reinforcement learning training school for autonomous driving," in *Proc. Conference on Robot Learning (CoRL)*, 2020.

[165] E. Vinitsky, N. Lichtle, K. Peng, X. Wang, Z. Wang, O. Oguntola *et al.*, "Nocturne: A scalable driving benchmark for bringing multi-agent learning one step closer to the real world," in *Proc. Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022.

[166] C. Gulino, J. Fu, W. Luo, G. Tucker, E. Bronstein, Y. Lu, J. Harb, X. Pan, Y. Wang, X. Chen *et al.*, "Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research," in *Proc. Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

[167] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[168] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," *arXiv preprint arXiv:2003.08938*, 2020.

[169] T. Oikarinen, W. Zhang, A. Megretski, L. Daniel, and T.-W. Weng, "Robust deep rein-

forcement learning through adversarial loss," *arXiv preprint arXiv:2008.01976*, 2020.

[170] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attacks on deep reinforcement learning agents," in *Proc. 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 3756–3762.

[171] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun, "AdvSim: Generating safety-critical scenarios for self-driving vehicles," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[172] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," in *Proc. 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018, pp. 1040–1042.

[173] A. Kumar, A. Levine, and S. Feizi, "Policy smoothing for provably robust reinforcement learning," *arXiv preprint arXiv:2106.11420*, 2021.

[174] E. Leurent, "An environment for autonomous driving decision-making," GitHub repository, 2018. [Online]. Available: https://github.com/eleurent/highway-env

[175] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[176] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. International Conference on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[177] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2014.

[178] Y. Sun, R. Zheng, Y. Liang, and F. Huang, "Who is the strongest enemy? Towards optimal and efficient evasion attacks in deep RL," in *Proc. International Conference on Learning Representations (ICLR)*, 2022.

[179] F. Wu, L. Li, Z. Huang, Y. Vorobeychik, D. Zhao, and B. Li, "CROP: Certifying robust policies for reinforcement learning through functional smoothing," in *Proc. International Conference on Learning Representations (ICLR)*, 2022.

[180] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *Proc. International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM)*, 2017, pp. 262–275.

[181] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[182] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[183] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll, "A review of safe reinforcement learning: Methods, theory and applications," *arXiv preprint arXiv:2205.10330*, 2022.

[184] W. Zhao, T. He, R. Chen, T. Wei, and C. Liu, "State-wise safe reinforcement learning: A survey," *arXiv preprint arXiv:2302.03122*, 2023.

[185] K. L. Hobbs, M. L. Mote, M. C. L. Abate, S. D. Coogan, and E. M. Feron, "Runtime assurance for safety-critical systems: An introduction to safety filtering approaches for complex control systems," *IEEE Control Systems Magazine*, vol. 43, no. 2, pp. 28–65, Apr. 2023.

[186] A. Selim, W. A. Dawson, W. Dixon, and C. Belta, "Safe reinforcement learning using black-box reachability analysis," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 665–10 672, 2022.

[187] P. Kamalaruban, Y.-T. Huang, Y.-P. Hsieh, P. Rolland, C. Shi, and V. Cevher, "Robust reinforcement learning via adversarial training with Langevin dynamics," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[188] R. Yang, C. Bai, X. Ma, Z. Wang, C. Zhang, and L. Han, "RORL: Robust offline reinforcement learning via conservative smoothing," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022.

[189] A. Bukharin, Y. Li, Y. Yu, Q. Zhang, Z. Chen, S. Zuo, C. Zhang, S. Zhang, and T. Zhao, "Robust multi-agent reinforcement learning via adversarial regularization: Theoretical foundation and stable algorithms," in *Advances in Neural Information Processing Systems*

(NeurIPS), vol. 36, 2023.

[190] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018, pp. 3207–3214.

[191] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling, "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents," *Journal of Artificial Intelligence Research*, vol. 61, pp. 523–562, 2018.

[192] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[193] L. Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, Jul. 2001.

[194] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Madison, WI, USA: Nob Hill Publishing, 2017.

[195] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[196] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[197] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang, "Safety gymnasium: A unified safe reinforcement learning benchmark," in *Proc. Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

[198] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2022.

[199] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "MetaDrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3461–3475, 2023.

[200] A. Brohan *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proc. Conference on Robot Learning (CoRL)*, 2023.

[201] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[202] C. Ling, X. Zhao, J. Lu, C. Deng, C. Zheng, J. Wang, T. Chowdhury, Y. Li, H. Cui, X. Zhang *et al.*, "Beyond one-model-fits-all: A survey of domain specialization for large language models. corr abs/2305.18703 (2023)," 2023.

[203] Q. Zhang, K. Ding, T. Lv, X. Wang, Q. Yin, Y. Zhang, J. Yu, Y. Wang, X. Li, Z. Xiang *et al.*, "Scientific large language models: A survey on biological & chemical domains," *ACM Computing Surveys*, vol. 57, no. 6, pp. 1–38, 2025.

[204] S. Bedi, Y. Liu, L. Orr-Ewing, D. Dash, S. Koyejo, A. Callahan, J. A. Fries, M. Wornow, A. Swaminathan, L. S. Lehmann *et al.*, "Testing and evaluation of health care applications of large language models: a systematic review," *Jama*, 2025.

[205] M. Mozes, X. He, B. Kleinberg, and L. D. Griffin, "Use of llms for illicit purposes: Threats, prevention measures, and vulnerabilities," *arXiv preprint arXiv:2308.12833*, 2023.

[206] J. Sallou, T. Durieux, and A. Panichella, "Breaking the silence: the threats of using llms in software engineering," in *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, 2024, pp. 102–106.

[207] Y. Zeng, Y. Wu, X. Zhang, H. Wang, and Q. Wu, "Autodefense: Multi-agent llm defense against jailbreak attacks," *arXiv preprint arXiv:2403.04783*, 2024.

[208] X. Du, F. Mo, M. Wen, T. Gu, H. Zheng, H. Jin, and J. Shi, "Multi-turn jailbreaking large language models via attention shifting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 22, 2025, pp. 23 814–23 822.

[209] Y. Liu, Q. Zhang, and M. Chen, "Jbshield: Activated concept manipulation for jailbreak defense in large language models," *arXiv preprint arXiv:2502.07557*, 2025.

[210] J. Zhang, L. Wang, and H. Yu, "Ccfc: Core and core-full-core dual-track defense for llm jailbreak protection," *arXiv preprint arXiv:2508.14128*, 2025.

[211] T. Wang, K. Zhao, and Z. Sun, "Smoothllm: Defending large language models against jailbreak attacks via prompt perturbation and aggregation," *arXiv preprint arXiv:2310.03684*, 2024.

[212] C. McCauley, K. Yeung, J. Martin, and K. Schulz, "Novel universal bypass for all major llms," 2025, accessed: 2025-07-14. [Online]. Available: https://hiddenlayer.com/innovation-hub/novel-universal-bypass-for-all-major-llms/

[213] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[214] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," *arXiv preprint arXiv:2307.15043*, 2023.

[215] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing nlp," *arXiv preprint arXiv:1908.07125*, 2019.

[216] ProtectAI, "Deberta-v3 prompt injection classifier," 2024. [Online]. Available: https://huggingface.co/protectai/deberta-v3-base-prompt-injection-v2

[217] deepset, "Deberta-v3-base injection classifier," 2023. [Online]. Available: https://huggingface.co/deepset/deberta-v3-base-injection

[218] M. A. Ayub and S. Majumdar, "Embedding-based classifiers can detect prompt injection attacks," *arXiv preprint arXiv:2410.22284*, 2024.