



AFRL-RI-RS-TR-2022-122

LEVERAGING SYMBOLIC REPRESENTATIONS FOR SAFE AND ASSURED LEARNING

GALOIS, INC.

AUGUST 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2022-122 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

STEVEN L. DRAGER
Work Unit Manager

/ S /

GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED	
AUGUST 2022		FINAL TECHNICAL REPORT		START DATE JUNE 2019	END DATE DECEMBER 2021
4. TITLE AND SUBTITLE LEVERAGING SYMBOLIC REPRESENTATIONS FOR SAFE AND ASSURED LEARNING					
5a. CONTRACT NUMBER FA8750-19-C-0092		5b. GRANT NUMBER N/A		5c. PROGRAM ELEMENT NUMBER 62303E	
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER R2S0	
6. AUTHOR(S) Aditya Zutshi, Chris Lockett, Thomas Dietterich, Alan Fern, Suresh Jagannathan, Ufuk Topcu, Swarat Chaudhuri					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Galois, Inc. 421 SW 6th Ave Ste 300 Portland OR 97204				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITA DARPA 525 Brooks Road 675 North Randolph St Rome NY 13441-450 Arlington, VA 22203-2114			10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RI-RS-TR-2022-122
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. PA#: AFRL-2022-3920 Date Cleared: 16 August 2022					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Report developed under contract FA8750-19-C-0092: Leveraging Symbolic Representations for Safe and Assured Learning. This research effort targets development of novel tools, algorithms and methodologies to improve the assurance of Autonomous, Learning Enabled Cyber Physical Systems (LE-CPSs). These systems exhibit a rich set of behaviors due to higher levels of autonomy, and interaction between cyber components and the physical environment. This effort summarizes advances in symbolic system testing, model extraction, anomaly detection, learning unknown dynamics and formal approaches to verify these systems. Efforts were integrated within the Controls Systems Analysis Framework and applied to a high fidelity F16 model as the challenge problem.					
15. SUBJECT TERMS Cyber Physical Systems, Machine Learning, Model Extraction, Digital Engineering, Symbolic Testing, Formal Verification, F16, Air Collision Avoidance, Run Time Assurance, Anomaly Detection, Open World Classification, Learning Unknowns System Dynamics					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT		18. NUMBER OF PAGES
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR		107
19a. NAME OF RESPONSIBLE PERSON STEVEN L. DRAGER					19b. PHONE NUMBER (Include area code) N/A

TABLE OF CONTENTS

Section	Page
LIST OF FIGURES	iii
LIST OF TABLES	iv
LIST OF EQUATIONS	v
1.0 SUMMARY	1
1.1 Assured Learning Challenge Problem	2
1.2 Control Systems Analysis Framework	2
2.0 INTRODUCTION	3
2.1 Symbolic System Testing.....	3
2.2 Assured Learning Challenge Problems	3
2.3 Approaches for Training, Robustness and Formal Verification of CPS Systems.....	4
2.4 Anomaly Detection Investigation by OSU	5
2.5 Learning System Dynamics On-The-Fly (Physics Aware Learning)	5
2.6 Reinforcement Learning Methods for Learning Programmatic Policies	6
3.0 METHODS, ASSUMPTIONS, AND PROCEDURES.....	7
3.1 Challenge Problem and Integration Efforts	7
3.2 Novel Second Order Non-Convex Optimizers for Efficient and Effective Training of ML Models.....	11
3.3 Open-Category Detection and Out of Distribution Dynamics	25
3.4 Data Driven Algorithms for Analysis and Control of Unknown Dynamical Systems	29
3.5 Reinforcement Learning Methods for Learning Programmatic Policies	33
4.0 RESULTS AND DISCUSSIONS	35
4.1 Assured Learning Milestones and Deliverables.....	35
4.2 Results for Efficient and Scalable Training of ML Models for CPS Systems.....	43
4.3 Open Category Detection and Anomaly Detection Results	60
4.4 Results from Learning Unknown System Dynamics	64
4.5 Results from Reinforcement Learning Methods for Learning Programmatic Policies.....	68
4.6 Transition Results and Open Source Release of Assured Learning.....	70
5.0 CONCLUSIONS.....	72
5.1 F16 Challenge Problem Conclusions.....	72
6.0 RECOMMENDATIONS	74
7.0 REFERENCES	75

APPENDIX A: KEY ACCOMPLISHMENTS	81
APPENDIX B: CSAF CAPABILITY SUMMARY	83
APPENDIX C: TOWARDS PRINCIPLED AUTONOMY, A WHITE PAPER.....	85
BIBLIOGRAPHY.....	95
LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS.....	96

LIST OF FIGURES

Figure 1 Assured Learning Team Poster	1
Figure 2 Overview of Control Systems Analysis Framework	2
Figure 3 Phase I F16 Challenge Problem and Testing Framework	8
Figure 4 Fuzzy learned LQR Controller for F-16 Inner Loop	9
Figure 5 F16 RTA Test Harness with FlightGear Integration	10
Figure 6 Phase I F-16 Challenge Problem Demonstration	11
Figure 7 CNN Model and Layer Composition	15
Figure 8 Reward functions using FITRE and Proximal Policy Optimization (PPO)	18
Figure 9 Demonstration of the Actor-Critic-FITRE and PPO RL controller on the F16	18
Figure 10 Purdue's Algorithm 2: Shield Defense Outline.....	20
Figure 11 Approach Overview for Scalable Synthesis for Verified Controllers	22
Figure 12 Assured Learning Milestones (Planned and Completed)	36
Figure 13 Phase 1 Assured Learning Schedule.....	36
Figure 14 Phase 2 Assured Learning Schedule.....	37
Figure 15 Exemplar Jupyter Notebook Developed on Assured Learning	41
Figure 16 Summary of Assured Learning Tools, Algorithms and Frameworks Developed	42
Figure 17 Comparison of Multiple Inner Loops for F16	42
Figure 18 Classical vs NN Controller for Inverted GCAS	43
Figure 19 The FITRE Algorithm and Method.....	44
Figure 20 Reward functions from trajectories for Actor-Critic Reinforcement Learning using FITRE and Proximal Policy optimization (PPO).....	52
Figure 21 Demonstration of the RL controller on the F16 simulator using Actor-Critic-FITRE and Proximal Policy optimization (PPO) RL controllers	53
Figure 22 Average Reward for Original Policy and Shielded Policy	55
Figure 23 Mean Task Completion Rates on Specifications ϕ_1, ϕ_2, ϕ_3 and ϕ_4	58
Figure 24 Specification satisfaction percentages (left) for ϕ_{ex} . Mean task completion (right) rates for the 3D Navigation experiments on specification ϕ_a	60
Figure 25 How changing the horizon affects anomaly detection in terms of AUC.....	64
Figure 26 Plot of Immediate Results over 100 Episodes.....	65
Figure 27 On-The-Fly-Control Results for F16.....	66
Figure 28 Numerical Results for the Double Pendulum.....	67
Figure 29 Trajectories for 'acc' Modeling Adaptive Cruise Control.....	70

LIST OF TABLES

Table 1 Description of Notation Employed	14
Table 2 Assured Learning Milestone Deliverable Summary.....	37
Table 3 CDRL Deliverables for Phases 1 and 2	40
Table 4 Comparison of VGG16 Using ImageNet Dataset.....	45
Table 5 Comparison of VGG16 with a quasi-Newton Method.	48
Table 6 SGD with various learning rates on VGG Networks.....	49
Table 7 Reparameterization Invariance results for SGD and FITRE methods.....	50
Table 8 Experimental Results from Benchmarks	53
Table 9 Effectiveness of the safety planner on training safe and performant networks	57
Table 10 Task Completion Percentage on Convergence for N=3	59
Table 11 Example experimental results in CFAR100.	61
Table 12 Example experimental results in CFAR10	62
Table 13 Example experimental results in CFAR100	62
Table 14 Anomaly detection results	63
Table 15 Performance Results in TORCS	69
Table 16 Generalization results in TORCS	69
Table 17 Assured Learning DISTAR Requests and Status	71

LIST OF EQUATIONS

Equation 1 Local Quadratic Approximation.....	13
Equation 2 Matrix Equations for FITRE Algorithm.....	15
Equation 3 FITRE Linear Transformation and Activation Function.....	16
Equation 4 Relationship of Gradient and Hessian of a Function (f).....	16
Equation 5 Hessian Vector Product	16
Equation 6 Computation of Gradient Loss Function	16
Equation 7 Training Objective Function.....	17
Equation 8 Loss Given by Conditional Distribution.....	17
Equation 9 Local Predicates for States	25
Equation 10 Counterparts to Predicates that Act on Global State	25
Equation 11 reach_{lo} Semantics	25
Equation 12 reach_{gl} Semantics	25
Equation 13 Control-Affine Nonlinear System	30
Equation 14 Function Encoding Prior Knowledge on Model.....	32
Equation 15 Dynamics of the double pendulum.....	67

The views, opinions, and/or findings contained in this report are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Air Force Research Laboratory or the Department of Defense.

1.1 Assured Learning Challenge Problem

The team identified the F-16 flight control system¹ [75] as a Department of Defense (DoD) relevant benchmark for the Assured Autonomy Program. This supported the program Phase II challenge problem 2.5 “Ground Collision Avoidance” and Phase III challenge problem 3.2 “Air Collision Avoidance”. We developed the Controls Systems Analysis Framework (CSAF), an integrated toolchain to demonstrate the effectiveness of the techniques and tools developed by the team. We used DevOps style automation to impellent continuous integration and goal directed testing for LE-CPS. Finally, we demonstrated the viability of Runtime Assurance (RTA) to protect the system against LE-CPS failures.

1.2 Control Systems Analysis Framework

Galois developed the Control Systems Analysis Framework, an integrated tool chain to model, test and formally verify LE-CPS. CSAF brings state-of-the-art analysis tools into a digital engineering framework supported by DevOps style continuous integration to accelerate the digital design of safety critical systems. CSAF enables continuous verification via rigorous automated testing of learning enabled components. Figure 2 illustrates how CSAF integrates with existing and emerging tools to provide automated, goal directed testing, run-time assurance verification into a Continuous Integration/Continuous Deployment (CI/CD) environment.

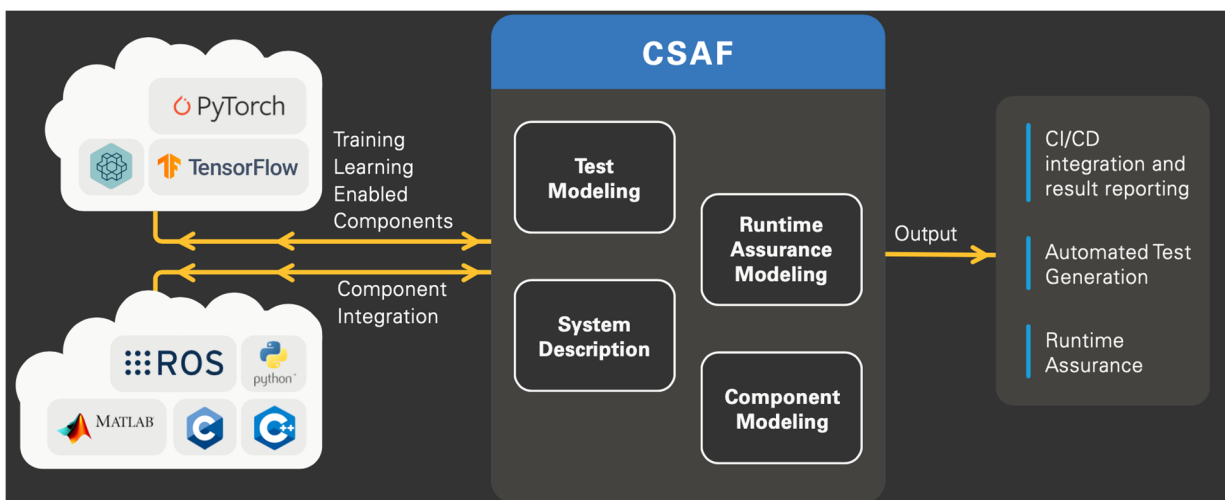


Figure 2 Overview of Control Systems Analysis Framework

¹ DISTRIBUTION A: Approved for public release; distribution unlimited. Case Number: 2018-0257 (original case number(s): MSC/PA-2018-0056; 88ABW-2018-1310), completed on 18 Apr 2018.

2.0 INTRODUCTION

The tremendous advances that have been made in the last decade on applying machine learning techniques to cyber-physical systems has led to the development of learning-enabled autonomous cyber-physical systems; these systems have seen promising initial deployment in air, ground, and deep-sea environments. Machine learning has enabled robust and adaptive approaches to image processing, decision-making, and control. However, wider adoption and broader acceptance of these systems is hindered by the lack of strong assurance guarantees on their operational behavior and the large amounts of data needed by many training approaches.

2.1 Symbolic System Testing

Phase 1 of this effort explored how formal verification and principled testing techniques developed for traditional software systems can be repurposed for verification, safety and improved training times of learning enabled systems. Specifically, we targeted these techniques on a real-world, complex, heterogeneous system (F-16 model) consisting of a mix of traditional algorithms, adaptive control, and learning-enabled components. Our techniques incorporated relevant domain knowledge as part of model generation where in phase 2 we produced high-coverage system-level tests. This approach provides high assurance in the safety of deployed machine learning models, even in the face of unexpected environment disturbances. Initial results in Phase 1 successfully demonstrated the use of formal models to effectively guide learning, improve training times, increase performance and enable a more efficient use of data. As a result, we expect this will significantly expand the scope of system components to which deep learning approaches can be applied.

We combined the above formal verification and testing of learning-enabled systems with work on run-time protection approaches to ensure the correctness of a learning enabled controller (LEC) algorithm. We leverage advanced Machine Learning (ML) anomaly detection methods to assess the quality of the LEC output. We provide alternate, deterministic programs that can be used both to steer a LEC from an undesirable behavior as well as provide an alternate abstraction of the LEC behavior suitable for formal verification. Finally, we provide methods to leverage our testing results to increase the efficiency and robustness of our re-trained LECs.

While the adaptation of control laws via learning is important to enable more flexible vehicle systems, adaptation to changes in dynamics, e.g. due to degradation of components or vehicle damage is equally important. Phase 1 advanced prior work in developing novel approaches to the learning of system dynamics that use knowledge of the underlying system physics to deliver substantially more trustworthy and efficient results.

2.2 Assured Learning Challenge Problems

Galois's efforts, led by Dr. Aditya Zutshi, began with a collection of benchmark challenge problems including Boeing's TAXINET/X-Plane simulator, Car Learning to Act (CARLA), and academic benchmarks used in control theory research. The F-16 AeroBench benchmark [76] was selected to align our work with the air domain challenge

problem set and perception work underway with Boeing's TAXINET/X-plane efforts using perception-based learning techniques. Phase 1 culminated in the early integration of our tools and techniques into a modularized learning enabled controller test harness using the F-16 model to demonstrate assured learning for the F-16 Ground Collision Avoidance System (GCAS) maneuver. We demonstrated successful GCAS maneuver as part of Phase 1 Challenge Problem 2.5.

In Phase 2, the team continued development of the Control Systems Analysis Framework and additional tools focused on the F-16 Air Collision Avoidance (ACA) challenge problems. In Phase 2, we demonstrated single and multi-agent F-16 Air Collision Avoidance as part of the Program's Challenge Problem 3.2.

2.3 Approaches for Training, Robustness and Formal Verification of CPS Systems

The Purdue team, led by Drs. Suresh Jagannathan and Ananth Grama's efforts, were directed toward six core components of assured learning-enabled controllers:

- (i) next generation of efficient and effective training procedures for learning models;
- (ii) use of new training procedures in reinforcement learning (RL) controllers;
- (iii) formal approaches to assuring performance of controllers and safety of the physical plant;
- (iv) robustness techniques applied to CPS systems;
- (v) scalable verification of controllers; and;
- (vi) new specification formalisms and compilation techniques for high-assurance multi-agent autonomous systems;

2.4 Anomaly Detection Investigation by OSU

The Oregon State University (OSU) team, led by Drs. Tom Dietterich and Alan Fern, investigated anomaly detection (AD) for two types of data that span many application areas. The motivation for studying anomaly detection is that assured autonomy for machine-learned systems must be able to detect when a system is in a new "operating regime" compared to the training regime.

The first AD investigation studied the problem of open-category detection for deep image classification. This problem involves developing approaches that can determine if a trained deep image classifier encounters an input image after deployment that is from a novel class with respect to the classes in the training data. Our work focused on using "oracle techniques" to analyze the interaction of deep representations and anomaly detection methods for this problem. The results suggest the relative head-room for improvement from improved representations versus improved anomaly detection signals based on those representations.

The second AD investigation studied the problem of out-of-distribution dynamics (OODD) detection. This problem involves determining when a temporal process, e.g. a deployed learned control system, is experiencing dynamics that are different than those it was trained in. This is a problem that has received little attention in the deep reinforcement learning (DRL) literature, despite the practical importance. Our first contribution was to design and develop the first set of OODD benchmarks and metrics based on commonly used DRL domains. Our second contribution was to design, develop, and evaluate a strong baseline OODD approach, which provided non-trivial performance, but also leaves room for improvement on the benchmarks. This sets the stage for future progress on OODD detection.

2.5 Learning System Dynamics On-The-Fly (Physics Aware Learning)

The University of Texas (UT) Austin team, led by Dr. Ufuk Topcu, focused work to develop data-driven algorithms for reachability analysis and control of unknown dynamical systems under very severe data limitations. This work is motivated by scenarios in which significant and unexpected changes in the dynamics of a system occur. For the Assured Learning Program, these efforts were applied to scenarios in which the F16 model was put in scenarios requiring learning system dynamics on the fly as a result in operating conditions, failures and in extreme ground collision avoidance scenarios in which traditional methods failed to provide assurances.

2.6 Reinforcement Learning Methods for Learning Programmatic Policies

The UT Austin team (originally at Rice University), led by Dr. Swarat Chaudhuri, developed Imitation-Projected Programmatic Reinforcement Learning (PROPEL), a reinforcement learning method for learning programmatic policies that can be expressed in a predefined symbolic language (a paper on this work, "Imitation-Projected Programmatic Reinforcement Learning", was published at the 2019 NeurIPS conference) [48]. A symbolic policy is a decision program, such as "if (the road is clear) accelerate, else slow-down", that could be conceivably written by a human engineer. As opposed to traditional deep reinforcement learning, in which learned policies are opaque neural networks, programmatic policies are auditable and simplify formal correctness verification. This policy learning problem was solved using a mirror descent approach that maintains an additive combination of a neural and a programmatic policy, iteratively updating both components using a mix of inductive program synthesis and gradient-based optimization (more on this in Section 3.5). The method was evaluated on a simulated driving task as well as a range of classic control tasks. The empirical results show that this method discovered human-interpretable policies without significantly compromising performance.

The team also developed an approach to reinforcement learning in which every intermediate policy constructed during exploration is formally verified, presented at NeurIPS 2020 "Neurosymbolic Reinforcement Learning with Formally Verified Exploration" (REVEL). Algorithmically, REVEL learns neural policies that are guarded by shields that ensure that every action is safe. The learning algorithm uses a form of mirror descent to iteratively update both the shield and the neural component of the policy while preserving safety, without requiring direct neural network verification. The approach was evaluated on several benchmarks from control and robotics that are analogous to simplified self-driving applications. It was shown that REVEL could learn policies with respectable performance while guaranteeing worst-case safety during the learning process, whereas other state-of-the-art reinforcement learning methods often violated safety constraints during the learning process.

In the last stage of the project, the team explored mechanisms for learning symbolic policies that operate on scenes perceived as pixels. Here, a neural network is used to identify the interesting entities in a scene. These entities are then treated as named variables, and a program that has these variables as programs is automatically discovered.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Challenge Problem and Integration Efforts

In December of 2019, Galois identified the F-16 open source challenge problem as the focus for our team's development efforts. Galois made progress in the development challenge problem work and integration efforts.

1. Modularized the python F-16 model into key architectural components.
2. Architected and performed initial implementation of the RTA assurance monitor and safe controller complex.
3. Completed development of the Piecewise hybrid representation of Neural Network (NN) controller.
4. Developed the initial library of motion primitive and encoded them using the CoPilot domain specific language.
5. Integrated the FlightGear simulation environment into the RTA test harness for demonstration of air domain challenge problems.

In Figure 3, we illustrate the re-architecture of the original F-16 model into the F-16 Learning Enabled Controller Test Harness

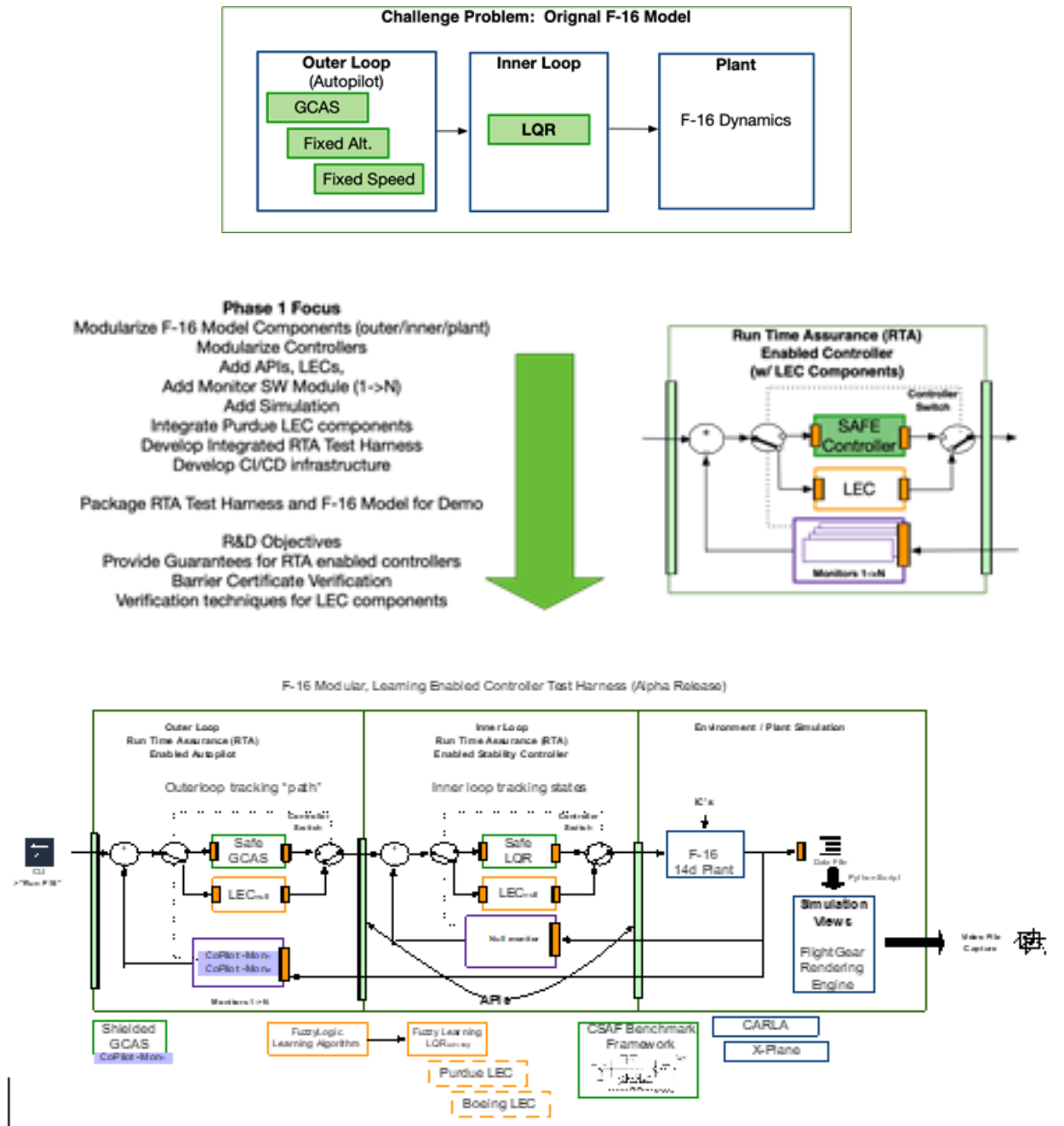
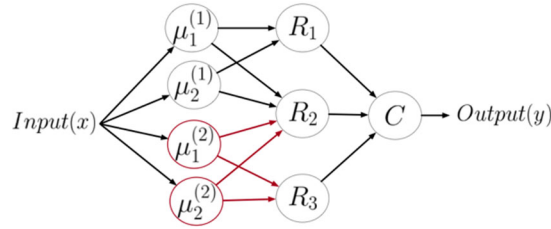


Figure 3 Phase I F16 Challenge Problem and Testing Framework

We created a general approach to learning linear controllers using a Fuzzy learning method to generate rule weights of the piecewise polynomial inner loop linear quadratic regulator (LQR) controller for use within the F-16 RTA test harness. This work is detailed in the following publication: Framework for Mining Hybrid Automata from a Constrained Machine Learning Architecture” [49]. Figure 4 illustrates that this approach achieves a match to the original LQR controller within 3 standard deviations. We integrated this into the F-16 RTA test harness as an inner loop controller.



Equivalent - Constrained Fuzzy NN Architecture

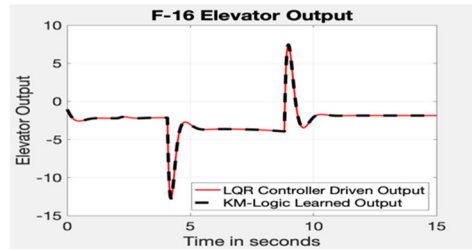


Figure 4 Fuzzy learned LQR Controller for F-16 Inner Loop

3.1.1 Motion Primitives and CoPilot Monitors. Motion primitives were modeled in the CoPilot domain specific language and compiled into C software monitor modules for implementation into the F-16 RTA Test Harness. The CoPilot monitor supplies a form of run-time assurance for the F16 by monitoring the state vector of the system to see if it is getting too close to unsafe states. If so, the CoPilot monitor engages a *recovery maneuver* to move the system away from the unsafe states. Each recovery maneuver is designed for a particular scenario in order to get back to a safe state in the most efficient way for that scenario. A recovery maneuver is built by composing a sequence of individual *motion primitives* that each perform a specific control action, such as pulling up or rolling to wings level.

This design methodology of breaking a recovery maneuver into individual motion primitives enables an approach of *compositional verification* to ensure that the recovery maneuvers are *correct*, that is, that they are guaranteed to avoid unsafe states in the particular scenarios for which they were designed. More specifically, each motion primitive is assigned a *precondition* describing what states it is designed to be run in and a *postcondition* describing what states will result after it is run. Each recovery maneuver is then verified for correctness by checking with a Satisfiability Modulo Theories (SMT) solver that the following conditions hold. First, that the *trigger condition* for each maneuver implies the precondition of the first motion primitive meaning that this motion primitive is only performed in states for which it was designed. Second, that the postcondition of each motion primitive implies the precondition of the next, meaning that each motion primitive leaves the aircraft in a state that the next motion primitive is designed for; and that the final postcondition implies that the aircraft is again in a safe state. Together, these conditions imply that the entire recovery maneuver is correct.

The copilot monitor currently uses the following recovery maneuvers to avoid collision with the ground, all of which are triggered when the altitude drops below 4000 feet. Each

To support a realistic demonstration, we have selected the controlled air space (R-2505)³ near Edwards Air Force Base as our geographic testing location which is approved for flight testing from ground to an unlimited altitude. **Figure 6** shows our Phase I F16 Challenge problem demonstration video with Flight Gear integration in R-2505 air space.

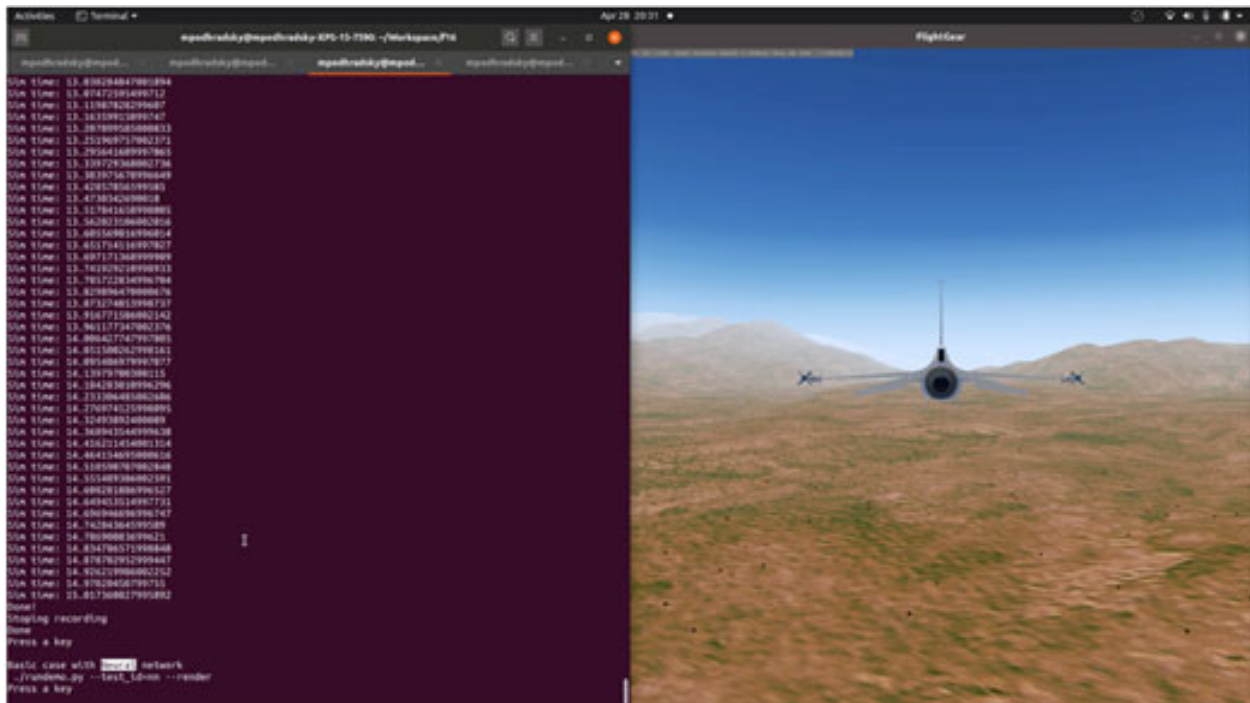


Figure 6 Phase I F-16 Challenge Problem Demonstration

3.2 Novel Second Order Non-Convex Optimizers for Efficient and Effective Training of ML Models

Deep neural networks are critical components of learning-enabled controllers. Enhancing their performance through improved training procedures has tremendous potential impact. First-order methods in general, and stochastic gradient descent in particular, have been the workhorse methods for a large subset of these problems. As part of this project, we have developed an efficient new second order method, Fisher Informed Trust REgion method (FITRE) for training deep neural networks. Our method leverages advantages of both trust-region and natural gradient methods, by employing the natural gradient direction as a way to approximately solve the trust-region sub-problems. We show that our method performs

2021.pdf?ver=dHD_HmYr01FAfgV6_X36FA== and <https://www.edwards.af.mil/Portals/50/documents/R-2508%20MACA%20Pamphlet.pdf>

³ Same sources as footnote 2

significantly better than well-tuned first-order and quasi-Newton methods in both generalization error and wall-clock times on a range of deep network architectures. In particular, our method converges much faster than the alternative methods in terms of data efficiency and iterations, yielding orders of magnitude of relative speedup. We further demonstrate the robustness of our method to different hyperparameters, which results in an easy-to-tune method in practice. We provide an open-source graphics processing unit (GPU) accelerated Compute Unified Device Architecture (CUDA) implementation of our solver for use in general deep network training.

Training of deep networks corresponds to finding a set of model parameters (weights) that minimize the loss function (a measure of error) associated with the training set. This optimization problem iterates over the (large) training set multiple times and learning model parameters until some predefined criteria of convergence is achieved. These processes may take hours for each iteration over the complete dataset, which translates to learning procedures that can take up to weeks. For instance, on the ImageNet dataset [1], which contains about 1.4 million samples, stochastic gradient descent (SGD) can take upwards of 30 hours for each pass over the dataset and weeks to train a deep neural network such as residual neural network (ResNet) [16] or Visual Geometry Group (VGG)Net [35] on workstation class computers. In spite of this significant training time, SGD and its distributed variants are the methods of choice in training of deep learning models.

The popularity of SGD, to a great extent, is attributed to its computationally inexpensive model parameter updates. Indeed, SGD's iterations involve computing the gradient of the objective function on a mini-batch, using back-propagation [21, 27], followed by scaling by a predetermined learning rate, possibly accelerated by momentum [28]. The simplicity of SGD allows for its application to a wide variety of learning tasks, e.g., auto-encoders [9, 39] and reinforcement learning [24, 33]. However, even though, in theory, it has been argued that SGD can avoid undesirable saddle-points [7], realizing this in practice is more involved [41]. In fact, without significant fine-tuning in terms of parameters and hyperparameters for learning rate, initialization, and mini-batch size, SGD's performance (with or without momentum) can diverge significantly from idealized theoretical bounds.

We have developed a new stochastic trust-region method that utilizes Kronecker Factored Approximate Curvature (KFAC) [25] to solving the trust-region subproblem, to obtain a Fisher Informed Trust REgion method that is shown to be well-suited for optimization of deep learning models in general, and convolutional neural networks (CNN) in particular. We have also leveraged the power of GPUs in accelerating various steps of FITRE to deliver excellent computing performance. This work has resulted in the following important contributions:

1. A novel second order stochastic variant of the trust region method, in which the approximations to the sub-problems are informed by directions obtained from KFAC.
2. Our method has excellent robustness as well as invariance to re-parameterizations.

3. Our optimized GPU implementation performs better than highly-tuned SGD and quasi-Newton alternatives, in terms of generalization errors, convergence rates, and wall-clock times.
4. Our method is resilient to the choice of batch size and tuning of the underlying hyperparameters, compared to conventional first-order methods.

We provide an open-source GPU accelerated CUDA optimization framework for CNNs with a first-of-its-kind R-operator for Hessian-vector computations.

3.2.1 FITRE: Overview, Algorithms and Implementation.

Notation: Vectors, $\mathbf{v}$, and matrices, $\mathbf{V}$, are denoted by bold lower- and upper-case letters, respectively. $\nabla f(\mathbf{x})$ and $\nabla^2 f(\mathbf{x})$ represent the gradient and the Hessian of f at $\mathbf{x}$, respectively. The superscript, e.g., $\mathbf{x}(k)$, denotes iteration count. S denotes a collection of indices drawn from the set $\{1, 2, \dots, n\}$, with potentially repeated items, and its cardinality is denoted by $|S|$. Following Matlab notation, $[\mathbf{v}; \mathbf{w}] \in \mathbb{R}^{2p}$ denotes vertical stacking of two column vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^p$, whereas $[\mathbf{v}, \mathbf{w}] \in \mathbb{R}^{p \times 2}$ denotes a p by 2 matrix whose columns are formed from the vectors $\mathbf{v}$ and $\mathbf{w}$. Vector l_2 norm is denoted by $\|\mathbf{x}\|$. For a boolean variable, $x \in \{\text{True}, \text{False}\}$, the indicator function $1(x)$ evaluates to one if $x = \text{True}$, and zero otherwise. $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v}$ denotes the dot product of vectors $\mathbf{u}$ and $\mathbf{v}$, and $\mathbf{A} \odot \mathbf{B}$ represents element-wise multiplication of matrices $\mathbf{A}$ and $\mathbf{B}$. $\mathbf{e}$ is used to indicate an all-ones vector, and $\mathbf{e}_i$ is used to indicate a vector in which the i th component is set to 1 while rest of the components are 0's.

Our method, FITRE, is inspired by [26] and [40, 42], and is formally described in Algorithm 1 detailed in section 4.2.1 (see Figure 19). At the heart of FITRE lies the stochastic trust-region method using a local quadratic approximation shown in Equation 1:

$$\min_{\|\mathbf{s}\| \leq \Delta_t} m_t(\mathbf{s}) = \langle \mathbf{g}_t, \mathbf{s} \rangle + \frac{1}{2} \langle \mathbf{s}, \mathbf{H}_t \mathbf{s} \rangle \quad (1)$$

Equation 1 Local Quadratic Approximation

We use a stochastic estimation of the gradient $\mathbf{g}_t$ and Hessian $\mathbf{H}_t$. The application of Hessian estimate, $\mathbf{H}_t$, which contains information regarding the curvature of the optimization landscape, has been shown to offer many advantages, including resilience to hyperparameter tuning and problem ill-conditioning [40, 41]. The step-length, which is governed by the trust-region radius Δ_t is automatically adjusted based on the quality of the quadratic approximation and the amount of descent in the objective function. In practice, (1) is approximated by restricting the problem to lower dimensional spaces, e.g., the Cauchy point, which amounts to searching in a one-dimensional space spanned by the gradient. Here, we do the same, however, by restricting the sub-problem to the space spanned by the KFAC direction, or its combination with the gradient.

Our choice is motivated by the following observation: when the objective function involves probabilistic models, as is the case in learning-enabled controllers, natural gradient

direction amounts to the steepest descent direction among all possible directions inside a ball measured by Kullback-Leibler Divergence (KL-divergence) between the underlying parametric probability densities. On the contrary, the (standard) gradient represents the direction of steepest descent among all directions constrained in a ball measured by the Euclidean metric [13], which is less informative than the former, though it is much easier to compute. To alleviate the computational burden of working with the Fisher information matrix and its inverse, Kronecker-product based approximations are shown to simultaneously preserve desirable properties of the exact Fisher matrix, such as invariance to reparameterization and resilience to large batch sizes.

Table 1 Description of Notation Employed

Description	Symbol
A sample point from the dataset and corresponding output	$(\mathbf{x}, y)$
Total number of classes in the dataset	c
Function of the network	$f(\mathbf{x}, y)$
Prediction of the network w.r.t $\mathbf{x}$	z
Loss function	$\mathcal{L}$
Gradient of the network	$\mathbf{g}$
Hessian of the network	$\mathbf{H}$
Fisher information matrix of the network	$\mathbf{F}$
Natural gradient direction	$\mathbf{r}$
Number of layers in the neural network	$\ell, (1, \dots, \ell)$
Input to layer l	$\mathbf{a}_{l-1}$
Mini-batch input to layer l	$\tilde{\mathbf{A}}_{l-1}$
Input to layer l , augmented with homogeneous coordinate	$\tilde{\mathbf{a}}_{l-1}$
Output of layer l	$\mathbf{a}_l$
Weights of layer l	$\mathbf{W}_l$
Bias of layer l	$\mathbf{b}_l$
Weights and bias folded into single parameter of layer l	$\tilde{\mathbf{W}}_l$
Parameters of the network, $\boldsymbol{\theta}$	$[\text{vec}(\tilde{\mathbf{W}}_1)^\top, \dots, \text{vec}(\tilde{\mathbf{W}}_\ell)^\top]^\top$
Output of the convolution function (and input to the activation function) of layer l	$\mathbf{s}_l$
Output of activation function (and input to pool function) of layer l	$\mathbf{p}_l$
Output of pool function of layer l	$\mathbf{a}_l$
Output of the neural network	$\mathbf{a}_{out} (= \mathbf{a}_\ell)$
Output of the loss function	$\mathbf{a}_{loss}$
Derivative of the loss function w.r.t a parameter, $\boldsymbol{\theta}$	$\mathcal{D}\boldsymbol{\theta} = \frac{d\mathcal{L}}{d\boldsymbol{\theta}}$
Gradient terms propagated from the loss function	$\mathbf{g}_{loss} = (\mathcal{D}\mathbf{a}_{out})$
Gradient terms propagated from the pool function of layer l	$\mathbf{g}_l^p (= \mathcal{D}\mathbf{p}_l)$
Gradient terms propagated from the activation function of layer l	$\mathbf{g}_l^a (= \mathcal{D}\mathbf{s}_l)$
Gradient terms propagated from the convolution function of layer l	$\mathbf{g}_l^{conv} (= \mathcal{D}\mathbf{a}_l)$

Computational Model: Table 1 describes symbols we use and their meaning in the context of deep neural networks. We demonstrate our solver on a typical CNN architecture, shown in Figure 7a. We assume that the network contains ℓ layers, and each layer can be either a convolution layer or a linear layer. A convolutional layer is composed of convolution and activation functions. It may also optionally contain pooling functions and batch-normalization layers, as shown in Figure 7b. Likewise, a linear layer is composed of a linear function and optionally can have activation functions within it as well. These layers are stacked together, so that the output of one layer forms the input of the next layer. The

last layer, i.e., l th layer, is connected to the loss function. In this illustration, we use softmax cross-entropy for our loss function, denoted as L .

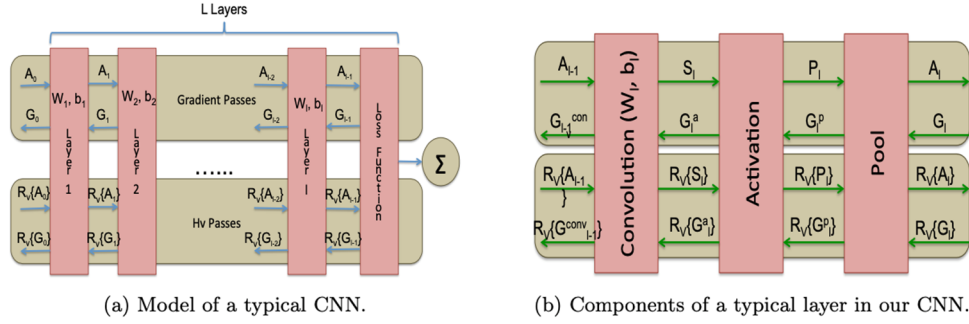


Figure 7 CNN Model and Layer Composition

Gradient computation is performed in two passes over the network, shown in the top half of the Figure 7a, using the back-propagation algorithm. In the forward pass for the gradient computation, a_{l-1} and a_l are, respectively, the input to and the output from the layer l . We bundle a set of sample points, also known as mini-batch, which forms the input to the first layer, A_0 . Please note that for analysis purposes all the equations in this report use one sample point for l th-convolution layers (i.e., A_l , a matrix of dimensions samples $\times$ channels) as well as m th-linear layers (i.e., a_m , a vector). The following equations summarize the operations performed during the forward pass for each of the layers in the network, as shown in Figure 7b:

$$\text{Convolution:} \quad C_l = \bar{W}_l \bar{a}_{l-1} \quad (2a)$$

$$\text{Activation:} \quad P_l = \phi(C_l) \quad (2b)$$

$$\text{Pool:} \quad A_l = \mathcal{P}(P_l). \quad (2c)$$

Equation 2 Matrix Equations for FITRE Algorithm

The convolution operation is represented as a matrix multiplication, as described in (2a), where W_l and A_{l-1} are the matrix of weights and input sample point. Note that the weights matrix, W_l and bias vector, b_l , associated with the l th layer of the network are folded into a single matrix $\bar{W}_l (= [W_l b_l])$ by appending the column vector b_l into W_l . The output of the convolution operation, C_l , forms the input to the non-linearity function ϕ and its output, P_l , is passed to the down sampling function, $\mathcal{P}$. The output of the pool function, A_l , forms the input to the next layer $l+1$ in the model shown in Figure 7a. During the backward pass of the gradient computation, the partial derivatives with respect to inputs to each layer are passed in the backward (opposite) direction through the network as shown in Figure 7a. Partial derivatives of layer l , G_{conv} , are fed to the preceding layer $l-1$ through the network. Inside layer l , the incoming gradient terms G_{conv} are passed through the $l+1$ pooling, non-linearity and convolution functions sequentially, backpropagating outputs G_p , G_a and G_{conv} respectively; see Figure 7b. Along similar lines, the equations for linear layer are as follows:

$$\text{Linear transformation:} \quad \mathbf{s}_l = \bar{\mathbf{W}}_l \bar{\mathbf{a}}_{l-1}^\top \quad (3a)$$

$$\text{Activation:} \quad \mathbf{a}_l = \phi(\mathbf{s}_l) \quad (3b)$$

Equation 3 FITRE Linear Transformation and Activation Function

Equations 3a and 3b represent the linear transformation and activation function performed by the linear layer during the forward pass of the network. During the backward pass, we use g_{l-1}^d and g_l^a represent the gradient terms propagated backwards by the linear transformation and activation function within the linear layer. The framework developed for gradient computations can also be adapted to compute the Hessian-vector products by using the ‘‘R-Operator’’ approach, first introduced in [32].

Specifically, the gradient and Hessian of a function, f , are related by:

$$\nabla f(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}) + \nabla^2 f(\boldsymbol{\theta})\delta\boldsymbol{\theta} + o(\|\delta\boldsymbol{\theta}\|^2). \quad (4)$$

Equation 4 Relationship of Gradient and Hessian of a Function (f)

By choosing $\delta\boldsymbol{\theta} = r\mathbf{v}$ for some $r \in \mathbb{R}$, Hessian-vector product, $\nabla^2 f(\boldsymbol{\theta})\mathbf{v}$, can be obtained using:

$$\nabla^2 f(\boldsymbol{\theta})\mathbf{v} = \lim_{r \rightarrow 0} \frac{\nabla f(\boldsymbol{\theta} + r\mathbf{v}) - \nabla f(\boldsymbol{\theta})}{r} = \left. \frac{d}{dr} \nabla f(\boldsymbol{\theta} + r\mathbf{v}) \right|_{r=0}. \quad (5)$$

Equation 5 Hessian Vector Product

Now, by defining

$$\mathcal{R}_{\mathbf{v}} \{\mathbf{u}(\boldsymbol{\theta})\} = \left. \frac{d}{dr} \mathbf{u}(\boldsymbol{\theta} + r\mathbf{v}) \right|_{r=0}, \quad (6)$$

Equation 6 Computation of Gradient Loss Function

for any vector valued function $\mathbf{u}$, we have $\nabla^2 f(\boldsymbol{\theta})\mathbf{v} = \mathcal{R}_{\mathbf{v}} \{\nabla f(\boldsymbol{\theta})\}$. Therefore, by applying $\mathcal{R}_{\mathbf{v}}$ to all the equations evaluated during the gradient computation of the given network, we can compute $\nabla^2 f(\boldsymbol{\theta})\mathbf{v}$ of the loss function associated with the given network. The forward and backward passes through the neural network associated with Hessian-vector computation are shown in the bottom half of Figure 7a and Figure 7b.

Define $\boldsymbol{\theta} = [\text{vec}(\mathbf{W}^{-1})\mathbf{I}, \dots, \text{vec}(\mathbf{W}^{-L})\mathbf{I}]^\top$, which is a vector of all the network’s parameters, concatenated together, and the vec operator is used to flatten the matrices by stacking columns together. The loss function, denoted by $L(\mathbf{y}, \mathbf{z})$, is used to measure the disagreement between a prediction $\mathbf{z}$ and a target $\mathbf{y}$ corresponding to the input-

output pair (x, y) . The training objective function $h(\theta)$ is the average of losses $L(y, f(x, \theta))$ over all input-target pairs (x, y) , i.e.,

$$h(\theta) \triangleq \frac{1}{n} \sum_{i=1}^n \mathcal{L}(y_i, f(\mathbf{x}_i, \theta)). \quad (7)$$

Equation 7 Training Objective Function

For a given (x, y) , its corresponding loss is given by the negative log likelihood associated with a conditional distribution of y given $z = f(x, \theta)$, as

$$\mathcal{L}(y, z) = -\log r(y|f(\mathbf{x}, \theta)) \triangleq -\log p(y|\mathbf{x}, \theta). \quad (8)$$

Equation 8 Loss Given by Conditional Distribution

Here, $p(y|x, \theta)$ is the conditional distribution of y given x that is implied by the neural network and parameterized by θ . Here the parameters are not assumed to be random, but one can extend this model to include priors on θ and effectively have a Bayesian model. Minimizing the objective function $h(\theta)$ is identical to maximizing the likelihood $p(y|x, \theta)$ over the training dataset.

Algorithm: Algorithm 1 (see section 4.2.1, Figure 19) describes FITRE, a realization of our proposed method in trust-region settings. First, the natural gradient direction, $\mathbf{p}_t$ is computed and used in determining the step-size. This is done by using the quadratic approximation of the objective function at $\mathbf{p}_t$, whose closed form solution is $(\Delta / \|\mathbf{H}_t \mathbf{p}_t + \mathbf{g}_t\|) (\mathbf{H}_t \mathbf{p}_t + \mathbf{g}_t)$. (Note that gradient, $\mathbf{g}_t$, can also be used to estimate the step size, and it may yield a better descent direction in some cases). Once the step-size, η , is determined, $\mathbf{p}_t$ is computed over the same mini-batch to determine the trust-region radius as well as the iterate update. These steps are repeated until desired generalization is achieved. Note that we can compare the efficiency of the natural gradient direction with that of the standard gradient and use the appropriate one at each iteration. This is referred to as “KFAC + gradient” in this algorithm.

3.2.2 Robustness to Adversarial Attacks in Learning Enabled.

Summary: Learning-enabled controllers used in cyber-physical systems are known to be susceptible to adversarial attacks. Such attacks manifest as perturbations to the states generated by the controller’s environment in response to its actions. We consider state perturbations that encompass a wide variety of adversarial attacks and describe an attack scheme for discovering adversarial states. To be useful, these attacks need to be natural, yielding states in which the controller can be reasonably expected to generate a meaningful response. We consider shield-based defenses as a means to improve controller robustness in the face of such perturbations.

Our defense strategy allows us to treat the controller and environment as black-boxes with unknown dynamics. We developed a two-stage approach using FITRE and Proximal Policy Optimization (PPO) to generate reward functions (Figure 8) to construct this defense and

have demonstrated its effectiveness through a range of experiments on realistic continuous control domains such as the navigation control-loop of an F16 aircraft (Figure 9) and the motion control system of humanoid robots.

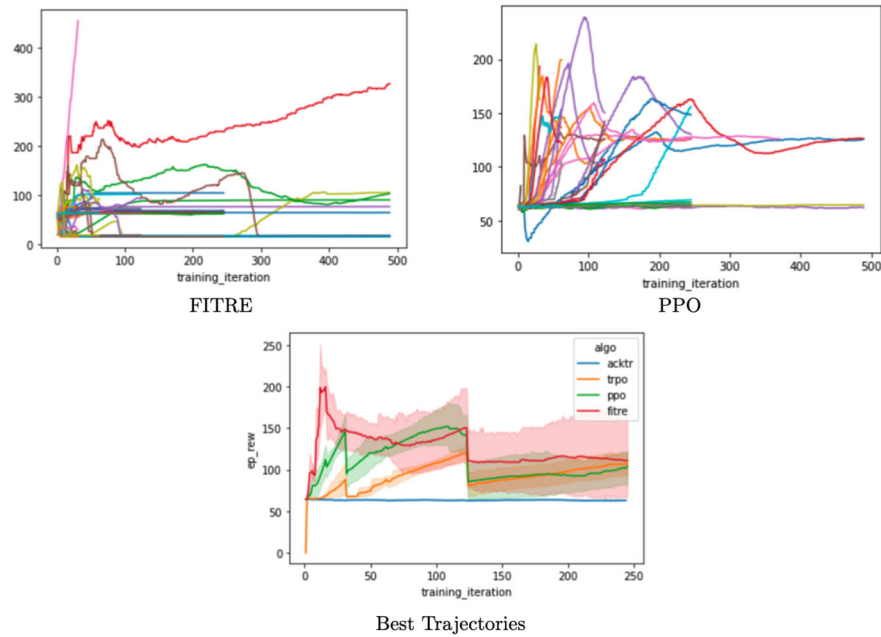


Figure 8 Reward functions using FITRE and Proximal Policy Optimization (PPO)

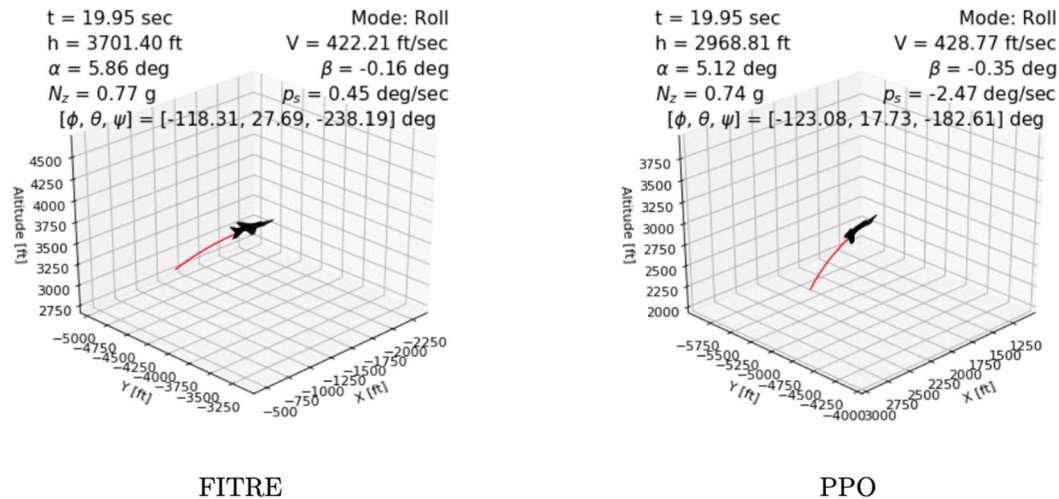


Figure 9 Demonstration of the Actor-Critic-FITRE and PPO RL controller on the F16

Background. Deep reinforcement learning approaches have shown promise in synthesizing high-quality controllers for sophisticated cyber-physical domains such as

autonomous vehicles and robotics [5, 8]. However, because these domains are characterized by complex environments with large feature spaces, they have proven vulnerable to various kinds of adversarial attacks [12] that trigger violations of safety conditions the controller must respect. For example, a safe controller for an unmanned aerial navigation system must account for a myriad of factors with respect to weather, topography, obstacles, etc., that may maliciously affect safe operation, only a fraction of which are likely to have been considered during training. Ensuring that controllers are robust in the face of adversarial attacks is therefore an important ongoing challenge.

Although the state space over which the controller’s actions take place is intractably large, the need to conform to physical realism greatly reduces the attack surface available to an adversary, in practice. For example, an adversarial attack on a Unmanned Aerial Vehicle (UAV) cannot generate an obstacle from thin air, suspend gravity, or instantaneously eliminate prevailing wind conditions. A meaningful attack is thus expected to generate states that are natural [12], i.e., states that can be derived from perturbations of realizable states as defined by the physics of the application domain under consideration. We are interested in identifying such states along a trajectory representing a rollout of the controller’s learnt policy.

Successful attacks manifest as a safety failure in this policy. While these kinds of falsification methods are certainly important [11] to assess the overall safety of a controller, devising a defensive strategy that prevents unsafe operation resulting from such attacks is equally valuable. In black-box, model-free RL settings where a policy’s internal structure is unknown, shield-based defenses are typically required [2].

Approach. We consider the generation of such defenses that comprise three distinct components: (1) a mechanism to efficiently explore adversarial attacks within a realistic bounded area of the states found in simulated and seemingly safe trajectories; (2) a detector policy that predicts safe and unsafe states based on previously observed successful attacks; and, (3) an auxiliary policy that is triggered whenever the detector identifies an unsafe state; the role of the policy is to propose a new action that re-establishes a safe trajectory. To be practical, each of these components requires new insights on adversarial attacks and defense in RL. While an effective defense must be tuned with respect to the detector to protect against identified attacks, any useful methodology must also be able to generalize the shield to successfully prevent unseen attacks, i.e., attacks for which the detector has not been trained against.

Consider a continuous-state system M where user-defined safe (S_s) and unsafe states (S_u) form a partition over the entire state space S of M . A rollout of a trained agent policy π_0 on M yields a state trajectory $\tau_0 = (s_0, \dots, s_T)$. Let $T(\pi_0, M, S_0)$ be the set of safe trajectories formed by rollouts of policy π_0 in the system M with initial state space S_0 (i.e. trajectories in this set do not contain any states in S_u).

Define a ϵ -state perturbation of a state s to be another state $s' \in S$ that is within ϵ distance away from s by varying dimensions in s . We choose the L_∞ norm to measure state distances. Given a state s , denote $P_\epsilon(s)$ to be the set of all possible perturbations on s .

We define an Adversarial State Attack $\text{Atte}(\tau_0)$ to be a set of states in which each state $s \in \text{Atte}(\tau_0)$ is an ε -state perturbation on a state in the trajectory τ_0 that causes the new trajectory to be unsafe. In other words, $s \in \text{Atte}(\tau_0)$ if, for some k and s' , $0 \leq k \leq T$, $s' \in P_\varepsilon(s_k)$, and executing the policy π_0 from s' leads to a state in S_u . This can be thought of as a state perturbation at an arbitrary time index along the trajectory that results in safety failure. These perturbations can be minute, a small shift of a robot due to a bump on the ground, or mild turbulence affecting the positioning of a jet in midair. If ε is unconstrained, perturbations can be unrealistic, unlikely to occur in practice, or impossible to defend against given existing environment transition dynamics.

We aim to find a way to defend against adversarial state attacks leading to a more robust control strategy that is still performant. In real-world black-box scenarios, we do not have access to either controller internals (underlying control policy) or environment conditions. Thus, an effective defense must treat the original policy π_0 as a black box and make no assumptions about the (hidden) environment dynamics. With these constraints in mind, we divide this problem into two stages. The first step is to detect adversarial state attacks, knowledge of which can be used to act differently, by shielding π_0 from malicious state perturbations (the second step).

One defense outline is to define a safety shield that modulates actions taken by π_0 based on the observation provided by the environment. We structure a simple two-level function to represent the shield shown in Figure 10 as Algorithm 2. The upper level detects that an Atte attack has occurred while the lower level specifies a policy to bring the system back to a safe state. We call the upper level function the detector and the lower level policy the auxiliary policy (π_{aux}).

Algorithm 2: Shield Defense Outline

```

Function Shield(state):
    Given detector,  $\pi_o$ ,  $\pi_{aux}$ 
    if detector (state) then
        | return  $\pi_{aux}$ 
    else
        | return  $\pi_o$ 

```

Figure 10 Purdue’s Algorithm 2: Shield Defense Outline

With this setup, the constructed defense then uses the detector to decide whether to follow the original policy π_0 or switch to π_{aux} . If these two functions are accurate, this approach is robust to adversarial state attacks for a given ε . We have developed novel techniques to learn these two functions keeping in mind the restrictions placed on the defense scheme.

3.2.3 Scalable Synthesis of Verified Controllers

Summary. There has been significant recent interest in devising verification techniques for learning-enabled controllers that manage safety-critical systems. Given the opacity and lack of interpretability of the neural policies that govern the behavior of such controllers, many existing approaches enforce safety properties through the use of shields (as discussed above), a dynamic monitoring-and-repairing mechanism that ensures a LEC does not emit actions that would violate desired safety conditions. These methods, however, have been shown to have scalability limitations because verification costs grow as problem dimensionality and objective complexity increase. To address this issue, we have developed an automated verification pipeline capable of synthesizing high-quality safety shields even when the problem domain involves hundreds of dimensions, or when the desired objective involves stochastic perturbations, liveness considerations, and other complex non-functional properties. Our key insight involves separating safety verification from neural controller training, instead of using pre-computed verified safety shields to constrain the training process.

Background. Deep Reinforcement Learning has proven to be a powerful tool to implement autonomous controllers for various kinds of cyber-physical systems [14, 23, 34]. Since these learning-enabled controllers are intended to operate in safety-critical environments, there has been significant recent interest in developing verification methods that ensure their behavior conforms to desired safety properties [2, 3, 17, 22, 38, 43]. While these different approaches all provide strong guarantees on controller safety, scaling their techniques, both with respect to problem dimensionality as well as objective complexity, has proven to be challenging. Approaches that attempt to statically verify that a neural controller always preserves desired safety guarantees may reject high-quality controllers that may only infrequently violate safety [17, 38]. Alternatively, techniques that dynamically monitor controller actions, triggering a safety shield when these actions may lead to an unsafe state, require the shield's behavior to align closely with the neural controller's, which is typically trained with various performance objectives, in addition to safety [3, 43]. Balancing these competing goals of ensuring safety on the one hand and maximizing objective reward on the other poses its own set of challenges that can compromise verifiability, performance, and safety.

Approach. Similar to other shielding-based approaches [2, 3, 43], our work does not statically verify a neural controller directly. Instead, it generates a family of linear controllers from known system dynamics and the problem specification; this family collectively serves as a shield that can dynamically enforce the safe operation of the system. Controllers are synthesized using linear-quadratic regulators that yield different instantiations based on a cost function expressed in terms of different parameter values associated with states and control inputs. Our verification strategy searches for a combination of different controller instances that satisfy a desired safety property in the aggregate; this search is driven by a probabilistic reachability analysis tailored for this purpose. Different controllers in this aggregation are optimized for different time steps in possible system trajectories. Not all combinations of different controllers can be verified to be safe, however. Nonetheless, even combinations that are possibly unsafe can be used to provide strong probabilistic guarantees on their safety efficacy, which the verifier can

leverage as part of its subsequent search. Notably, our technique considers safety verification independently of neural controller behavior and neural network internal structure, and thus enables scalability with respect to objective complexity and problem dimensionality. In our experiments, for example, we demonstrate successful verification of CPS benchmarks with over 800 dimensions, a scale that is significantly higher than reported with existing approaches.

Because these linear controllers are generated based only on safety considerations, they are not intended to serve as the primary mechanism for governing the actual operation of the system, which must also take into account other performance-related objectives. To ensure the safety of a neural controller, tailored with optimal execution in mind, with a synthesized linear controller family, developed with safety as its only goal, our pipeline additionally integrates the controller family as part of the DRL training process. This integration biases the training process of the neural controller towards verified safety characteristics. However, because neural controller objectives do not impose safety requirements on the training procedure, our technique allows neural policies to be learnt with complex objectives (e.g., achieving different goals in a specific temporal order, ensuring various kinds of liveness properties, etc.). A well-trained neural controller is expected to be generally well-behaved, but is not guaranteed to be safe. However, safety can be enforced by operating the neural controller in conjunction with the verified linear controller family. Our work thus demonstrates a fully automated safety verification pipeline for DRL-based learning-enabled controllers trained with complex objectives that has very positive scalability characteristics.

Our overall approach, shown in Figure 11 ensures a reinforcement learning system’s safety by inserting a safety planner in the loop that governs the interaction between a neural controller and its environment.

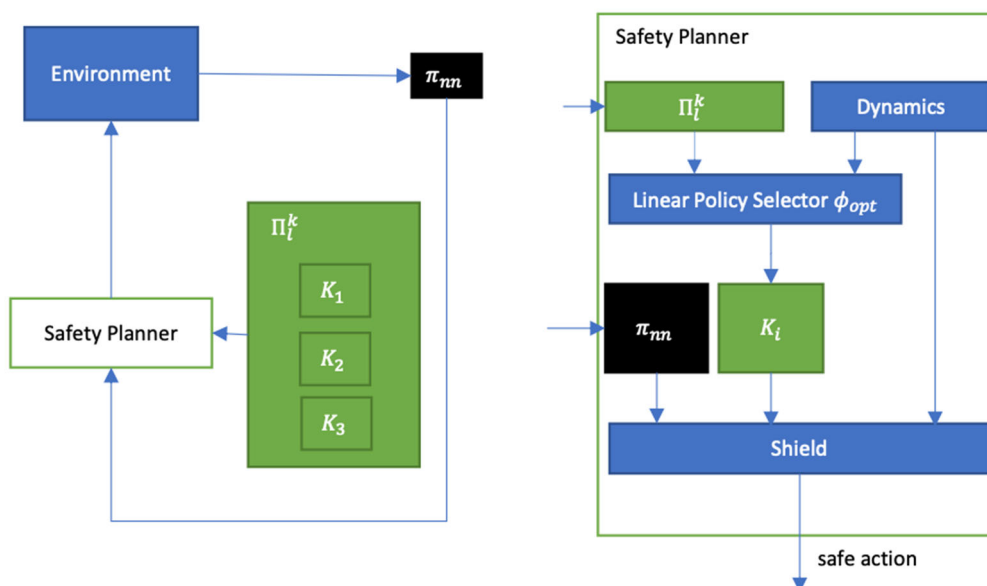


Figure 11 Approach Overview for Scalable Synthesis for Verified Controllers

The planner takes the neural network controller and a linear controller family as input. A selector ϕ picks up a linear controller from the linear controller family every k steps of a simulated trajectory. Given a linear controller picked by ϕ , the distribution of an initial state set, and characterization of (bounded) noise, the safety planner computes all states reachable by the controller in the next k steps of the trajectory, as well as a lower bound of the probability density of landing in these states. These states comprise a bounded reachable set for the selected controller. The verification algorithm aims to generate an optimal selector ϕ_{opt} such that it maximizes the overlap between the reachable set of selected linear policies and the safe state space. A safety shield takes the neural controller, the selected linear controller, and the k -step reachable set of the dynamics as input. It monitors a neural network's action and ensures that the proposed action yields a state within the reachable set of the linear policy, applying the action proposed by the safe linear controller when it does not. Because we maximize the selected linear controller's reachable set's overlapping part with the safe state space, we can provide a high probabilistic and oftentimes exact safety guarantee.

3.2.4 DistSPECTRL: Distributing Specifications in a Multi-Agent World

Summary. While notable progress has been made in specifying and learning objectives for general cyber-physical systems, applying these methods to distributed multi-agent systems still pose significant challenges. Among these are the need to (a) craft specification primitives that allow expression and interplay of both local and global objectives, (b) tame explosion in the state and action spaces to enable effective learning, and (c) minimize coordination frequency and the set of engaged participants for global objectives. To address these challenges, we have developed a novel specification framework that allows natural composition of local and global objectives used to guide training of a multi-agent system. Our technique enables learning expressive policies that allow agents to operate in a coordination-free manner for local objectives, while using a decentralized communication protocol for enforcing global ones. Experimental results support our claim that sophisticated multi-agent distributed planning problems can be effectively realized using specification-guided learning.

Background. Reinforcement Learning can be used to learn complex behaviors in many different problem settings. A main component of RL is providing feedback to an agent via a reward signal. This signal should encourage desired behaviors, and penalize undesirable ones, enabling the agent to eventually proceed through a sequence of tasks and is designed by the programmer beforehand. A commonly used technique to encode tasks in a reward signal is the sparse method of providing zero reward until a task is completed upon which a non-zero reward is given to the agent. Because this procedure has the significant shortcoming of delaying generating a useful feedback signal for a large portion of the agent-environment interaction process, a number of alternative techniques have been proposed [4, 29].

Formulating a reward signal that reduces the sparsity of this feedback is known as reward shaping. Often, this is done manually but a more general, robust method would be to automatically shape a reward given a specification of desired behavior. Spectrl [18]

proposes a reward shaping mechanism for a set of temporal logic specifications on a single-agent task that uses a compiled finite-state automaton called a task monitor. Reward machines [6, 36, 37] are another objective-specifying method for RL problems that also define a finite automaton akin to the ones used in Spectrl, with some subtle differences such as the lack of registers (used by the task monitor for memory).

Distributed multi-agent applications however, introduce new challenges in automating this reward shaping process. Agents have their own respective goals to fulfill as well as coordinated goals that must be performed in cooperation with other agents. While the expressiveness of the language in Spectrl lends itself, with minor extensions, to specifying these kinds of goals, we require new compilation and execution algorithms to tackle inherent difficulties in multi-agent reinforcement learning (MARL); these include credit assignment of global objectives and the presence of large state and action spaces that grow as the number of agents increases.

Learning algorithms for multi-agent problems have often encouraged distribution as a means of scaling in the presence of state and action space explosion. This is because purely centralized approaches have the disadvantages of not only requiring global knowledge of the system at all times but also induce frequent and costly synchronized agent control.

Approach. To address these issues, we have developed a new specification-guided distributed multi-agent reinforcement learning framework. Our approach has four main features. First, we introduce two classes of predicates (viz. local and global) to capture tasks in a multi-agent world. Second, we develop a new procedure for generating composite task monitors using these predicates and devise new techniques to distribute these monitors over all agents to address scalability and decentralization concerns. Third, we efficiently solve the introduced problem of subtask synchronization among agents via synchronization states in the task monitors. Fourth, we develop a specification framework capable of expressing a wide class of specification structures amenable to scaling in the number of agents and provide a means to perform such scaling. By using these components in tandem, we provide the first solution to composing specifications and distributing them among agents in a scalable fashion within a multi-agent learning scenario over continuous state and action spaces.

Specifications. Unlike the single agent case, multi-agent problems have two major classes of objectives. Agents have individual goals to fulfill as well as collective goals that require coordination and/or global system knowledge. These individual goals are often only dependent on the agent-specific state s_i while collective goals require full system knowledge s .

Consequently, for a multi-agent problem, we see the need for two types of predicates viz. local and global. Local predicates are of the form $p_{lo} : S_A \rightarrow B$ whereas global predicates have the form $p_{gl} : S \rightarrow B$ where B is the Boolean space. We introduce two simple extensions of reach [18] to demonstrate the capabilities of this distinction.

Local predicates are defined with respect to each agent and represent our individual goals. As an example, closely related to the problems observed in Spectrl, we introduce the following local predicates for a state $s_a \in S_A$,

$$\llbracket \text{reach}_{lo} x \rrbracket(s_a) = (\|s_a - x\|_\infty < 1) \quad (9)$$

Equation 9 Local Predicates for States

which represents reaching near location x in terms of the L_∞ norm. Now to enforce global restrictions, we introduce counterparts to these predicates that act on a global state $\bar{s} \in S$.

$$\llbracket \text{reach}_{gl} \bar{x} \rrbracket(\bar{s}) = (\|\bar{s} - \bar{x}\|_\infty < 1) \quad (10)$$

Equation 10 Counterparts to Predicates that Act on Global State

where we now have a set of locations $\bar{x} \in S$. As in Spectrl, each of these predicates b require quantitative semantics $\llbracket b \rrbracket_q$ to facilitate our reward shaping procedure. We define these semantics as follows:

- reach_{lo} has the same semantics as reach in [18] yet is defined on space S_A .

$$\llbracket \text{reach}_{lo} x \rrbracket_q(s_a) = 1 - d_\infty(s_a, x) \quad (11)$$

Equation 11 reach_{lo} Semantics

where $d_\infty(a, b)$ represents the L_∞ distance between a and b with the usual extension to the case where b is a set.

- reach_{gl} is defined on the state space S as

$$\llbracket \text{reach}_{gl} \bar{x} \rrbracket_q(\bar{s}) = 1 - d_\infty(\bar{s}, \bar{x}) \quad (12)$$

Equation 12 reach_{gl} Semantics

3.3 Open-Category Detection and Out of Distribution Dynamics

3.3.1 Open-Category Detection for Deep Image Classifiers. This project made two methodological contributions for analyzing open category detection in the domain of deep image classification. The first was to introduce the notion of oracle anomaly detection for quantifying the information available in a learned latent representation. The second is to introduce oracle representation learning, which produces a representation that is guaranteed to be sufficient for accurate anomaly detection.

These two techniques help to separate the quality of the learned deep representation from the performance of the anomaly scoring mechanism to help identify avenues for improvement. The methods also provide an upper bound on how much open category

detection can be improved through better anomaly scoring mechanisms. The combination of the two oracles gives an upper limit on the performance that any open category detection method could achieve.

Our problem setup assumes a training set of images that are labeled by one of K known classes. At test time, our system will be provided with a sequence of images which can be from one of the K known classes or one of U unknown classes that were not present at training time. The goal is to learn a scoring function $g(X)$ that produces an anomaly score that relates to whether the test image X is from an unknown class. By setting a threshold on the value of $g(X)$ we can arrive at an anomaly detector and measure the false positive and false negative rates. Alternatively, the quality of g can be measured in terms of threshold agnostic metrics such as Area Under the Curve (AUC).

In this work, we will divide $g(X)$ into an encoder $E(X)$, which produces a vector encoding of the input image, followed by an anomaly scoring function S that assigns an anomaly score to image encodings. Thus, we have that $g(X) = S(E(X))$. As one example of this architecture, we can start with any deep image classifier $f(X)$ trained on the K classes and let $E(X)$ correspond to the activations at some hidden layer of $f(X)$. In turn, $S(E(X))$ might be the entropy of the softmax activations of $f(X)$, which measures the uncertainty of $f(X)$ about the class label.

In our study, we considered three different types of representation encoders: 1) the penultimate layer of a discriminatively trained deep image classifier trained on the K known classes, 2) the encoder of a deep auto-encoder trained to minimize reconstruction, where we use a deterministic auto-encoder with bottleneck L2 regularization, 3) a hybrid encoder that is jointly trained as the penultimate layer of a deep classifier and the input to a decoder for image reconstruction.

Our study also considered a number of scoring functions S of encoded images. These included: 1) entropy of the predictive distribution learned on top of the given representation, 2) negative of the maximum softmax probability over classes, 3) decoder reconstruction error when using the encoded representation, 4) the score produced by the Isolation Forest (IF) anomaly detection algorithm applied to the encoded representation, 5) the score produced by the Local Outlier Factor (LOF) anomaly detection algorithm applied to the encoded representation.

A key part of our study was to evaluate the use of oracle encoders and oracle scoring functions. The motivation for this was to provide insight into how much room there is to improve performance via changes to the encoder versus scoring functions. To arrive at an oracle encoder, we trained a deep image classifier on representative data from both the K known and U unknown classes. Given an accurate classifier we know that the latent representation is sufficient for discriminating knowns from unknowns and hence can be viewed as an oracle representation. Similarly, given an encoder $E(X)$ we can obtain an oracle anomaly scoring function by training a supervised classifier to discriminate between known and unknown classes and using the probability of the unknown class as the anomaly signal. The oracle anomaly detector serves as an upper bound on how well an anomaly detector can do when using a particular encoder $E(X)$. Note that for both the oracle encoder

and detector we are allowing access to data and labels from the unknown classes, which is not available in a non-oracle setting.

3.3.2 OODD Detection: Benchmarks and Strong Baseline. Out-of-distribution dynamics detection involves detecting when the dynamics of a temporal process change compared to the training-distribution dynamics. This is relevant to applications in control, reinforcement learning, and multi-variate time-series, where changes to test time dynamics can impact the performance of learning controllers/predictors in unknown ways. This problem is particularly important in the context of deep RL, where learned controllers often overfit to the training environment. Currently, however, there is a lack of established OODD benchmarks for the types of environments commonly used in RL research. Our work designed and developed a set of OODD benchmarks derived from common RL environments with varying types and intensities of OODD. In addition, we designed, developed, and evaluated a strong OODD detection baseline, called the recurrent implicit quantile networks (RIQNs), which monitors autoregressive prediction errors for OODD detection.

Our OODD benchmarks are derived from RL benchmark environments; classic control from OpenAI Gym [Bro+16] (Acrobot, Cartpole, and LunarLander), physics simulations from Bullet [CB21] (Ant, Hopper, HalfCheetah, and Walker2D), and an F16 flight simulation including autopilots for fixed airspeed and fixed altitude, and a ground collision avoidance system. In all environments, the observations used for OODD are continuous environment state features, which vary in dimension depending on the environment, e.g., CartPole, Ant, and F16 have 4, 28, and 17 features, respectively. To obtain controllers for the environments, we trained policies via Deep Q-Network (DQN) [Mni+13] and Twin Delayed DDPG (TD3) [FHM18] RL algorithms for OpenAI Gym and Bullet, respectively. For the F16 setting, we used the provided autopilots and GCAS. Using these controllers, we generated a set of 10,000 nominal trajectories using the default environment settings. These trajectories are intended to represent the training environment to be used to train OODD detectors.

For each environment, we generate various types of anomalous trajectories which follow the nominal dynamics until a random time point upon which there is a switch to anomalous dynamics. The anomalous dynamics correspond to different ways of modifying the state features provided to the controllers through sensors. Note that such modifications can impact the behavior of the controllers, which in turn can impact the evolution of the trajectories. Importantly, we avoid modifications that cause a controller to completely fail (e.g., falling or crashing) soon after the anomaly is introduced. This is because such failures are easy to detect and do not allow testing the ability to detect anomalies before “disasters”, which is a primary goal. To achieve this, we only modify a small percentage of the features (at most 20%) chosen for each environment based on tests with the trained controllers. Each anomalous trajectory is generated by randomly selecting the appropriate size subset of features to modify, which provides diversity across the anomalous trajectories. We consider the following four types of anomalous trajectories, which differ in the way they modify the selected features.

- **Independent and Identically Distributed (IID) Noise.** Gaussian noise is added to the features. For each environment, the mean and standard deviation is selected to avoid near-term failure of the controller.
- **Sensor Shutdown.** The output of the sensor changes to zero immediately after the anomaly and stays at zero for the rest of the run. If applied to the most important features for a policy, it could easily degrade the performance. It resembles the case where a sensor fails and needs to be replaced.
- **Sensor Calibration Failure.** It multiplies the sensor’s output by a constant throughout the anomalous run. Similar to the IID noise, for each case study, there is a specific pre-determined constant.
- **Sensor Drift.** Relative to the time step in a trajectory, a small amount of noise is injected into the chosen sensor. As the trajectory progresses, the magnitude of the injected noise increases.

Further details of the parameters used for the different types of anomalies can be found in the full paper [DF21]. Table 14 (see section 4.3.2) gives the average reward of 1,000 trajectories achieved by controllers in the nominal and anomalous environments. We see a decrease in reward for anomalous environments but not catastrophic levels of decrease.

Our OOD detection baseline RIQN is based on *anomaly detection via prediction errors*, which involves the following steps: 1) train a dynamics model that can predict future observations based on past observations, 2) at each time point t use the model to make autoregressive predictions of observations at time $t + \Delta$, possibly for multiple values of Δ , 3) generate an anomaly signal based on the difference between the predictions and actual observations. Note that when used for real-time anomaly detection, the value of Δ places a fundamental lower-bound on how much delay there must be before the onset of an anomaly is detected (i.e., a delay of at least Δ). However, for “slower moving” anomalies (e.g., subtle sensor drift), it may be necessary to use larger values of Δ to detect an anomaly’s impact. Below we first describe our predictive model, followed by how it generates anomaly signals for OOD.

To capture aleatoric uncertainty, RIQNs predict a distribution over future observations given the observation history. For this purpose, RIQNs are based on Implicit Quantile Networks (IQNs) [Dab+18], which were proposed to represent value-distributions in distributional RL. IQNs learn an implicit representation of a value distribution using the quantile Huber loss. At a functional level, once trained, it can be used to generate samples from the learned distribution conditioned on the input. Our RIQN baseline includes 3 primary modifications of IQNs: 1) since we are interested in dynamics prediction in problems where observations do not fully capture the state, we add recurrent memory to the IQN network via a Gated Recurrent Unit (GRU) [Cho+14] layer resulting in the recurrent IQN model, 2) rather than predicting action values from state-action pairs as input, we train an RIQN for each observation feature to predict the feature distribution at time t based on the previous values of the features, and 3) rather than training via noisy RL-derived target values, we follow a supervised learning approach.

More specifically, for supervised learning, we minimize an asymmetric variant of the Huber loss (from prior work [Dab+18]) on predictions of each feature value at time t given all observation before time t . In addition, since we use the RIQN to generate auto-regressive predictions of future observation sequences, we potentially face the problem of compounding prediction error. That is, since during auto-regressive prediction, the network's outputs are fed back to the input, error accumulation can occur at test time. To address this, we use the Scheduled Sampling approach [Ben+15], which conducts training using inputs from ground truth and auto-regressive samples.

Given an RIQN model and a sequence of observations $f_{1:t}$ up to time t , we can auto-regressively sample a future predicted sequence of length H , $\hat{f}$. First, sample $\hat{f}_{t+1:t+H}$ given $f_{1:t}$, followed by sampling $\hat{f}$ given $\hat{f}$, and so on until sampling a value for $\hat{f}$. By repeating this $t+2 \ 1:t+1 \ t+H$ process we can generate multiple sampled sequences that for any time $t + \Delta$ yields a set of samples $\{\hat{f}(i)\}$ that can be used to represent the predictive distribution at time $t+\Delta$.

The above auto-regressive sampling approach provides a way of capturing aleatoric uncertainty in a dynamic process using a single RIQN model. To account for epistemic uncertainty due to limited nominal data, we train an ensemble of RIQN models, each of which can be used to sample predicted observation sequences. Each model is trained starting from different initial conditions and different hyperparameter settings, attempting to encourage diversity. In this work, we did not attempt to optimize the ensemble training approach, nor did we attempt to quantify how well the ensembles capture epistemic uncertainty.

Given an ensemble of e RIQN models, an observation sequence $f_{1:H}$, and a detection horizon Δ , we can now define a simple anomaly score to assign to any time $t \in \{\Delta, \dots, H\}$, denoted by A_t . First, use each RIQN model to produce a set of M auto-regressive samples at time t given the subsequence $f_{1:t-\Delta}$, resulting in a set $\{\hat{f}(1), \dots, \hat{f}(M \times e)\}$ of $M \times e$ samples. Given these samples the anomaly score is given by: $A_t = \frac{1}{M \times e} \sum_i |f_t - \hat{f}_i|$ which is just the average L1 distance between the samples and the actual observation at time t . There are other ways to define an anomaly score given an RIQN ensemble, and we leave a deeper comparison for future work. Given an anomaly score for each time point, one can consider various mechanisms for selecting an alarm threshold over which an OOD alarm is raised. In our experiments, we consider different choices for Δ and compare the OOD performance.

3.4 Data Driven Algorithms for Analysis and Control of Unknown Dynamical Systems

We develop data-driven algorithms for reachability analysis and control of unknown dynamical systems under very severe data limitations. The work is motivated by scenarios in which significant and unexpected changes in the dynamics of a system occur. The changes in the dynamics are such that the a priori known model cannot be used, and there is a need to learn the new dynamics on the fly. Thus, the system has access to data from only its current trajectory and needs to retain a certain degree of control. Under such a severe limitation on the amount of available data, learning can be performed efficiently only by incorporating already known invariant properties or qualitative properties of the

dynamical system derived from elementary laws of physics. We refer to such extra knowledge as side information and propose two complementary approaches to incorporate a wide variety of side information into data-driven models and control of dynamical systems. The approaches are extensively evaluated in high-physics robotics simulators and in the F-16 aircraft simulator.

3.4.1 On-The-Fly Control from a Single and Ongoing Trajectory. We develop data-driven algorithms, DaTaReach and DaTaControl, for the reachability analysis and control of systems with a priori unknown dynamics. DaTaReach exploits the data from a given trajectory of the system to compute an over-approximation of the set of states it may reach. DaTaControl incorporates the computed over-approximation into a constrained optimal control problem, which is then solved on the fly.

DaTaReach and DaTaControl can work with possibly noisy data from only a single finite-horizon trajectory of the system and take advantage of various forms of side information on the dynamics. Specifically, the data include finite (noisy) samples of the states, the derivatives of the states, and the control signals applied. The side information may be a priori knowledge of the regularity of the dynamics, bounds on the vector field locally, monotonicity of the vector field, decoupling in the dynamics among the states, algebraic constraints on the states, or knowledge of parts of the dynamics. For example, such side information may be due to known elementary laws of physics or may be directly extracted from the data.

Problem Setting: More specifically, in [74], we consider *control-affine* nonlinear systems,

$$\dot{x} = f(x) + G(x)u, \quad (13)$$

Equation 13 Control-Affine Nonlinear System

where $f: X \mapsto \mathbb{R}^n$ and $G: X \mapsto \mathbb{R}^{n \times m}$ are *unknown* vector-valued and matrix-valued functions, respectively, for $X \in \mathbb{R}^n$. Then, we assume that f and G are locally Lipschitz-continuous functions. That is, each component of f and G has a finite Lipschitz constant on each subset of $\mathbb{R}^n$. This assumption is common in the frameworks of reachability analysis and receding-horizon control. Since the domain $X \in \mathbb{R}^n$ is bounded, f and G are globally Lipschitz-continuous on X . We exploit such global Lipschitz continuity on X by assuming known upper bounds on the Lipschitz constants. That is, we have access to $L \in \mathbb{R}^n$ and $L \in \mathbb{R}^{n \times m}$ with $(L_f)_k = L_{fk}$ and $(L_G)_{k,l} = L_{Gk,l}$ as known upper bounds on the Lipschitz constants of $(f)_k$ and $(G)_{k,l}$ for all $k \in N_{[1,n]}$ and $l \in N_{[1,m]}$. We emphasize that the Lipschitz bounds can be directly estimated from data at the expense of weakening some of the guarantees in [74].

Let $N \in \mathbb{N}$, $N \geq 1$. Let $T_N = \{(x^i, \dot{x}^i, u_i)\}_{i=1}^N$ denote a single finite-horizon trajectory containing N noisy samples of the exact state $x_i = x(t_i)$, the derivative $\dot{x}^i = \dot{x}(t_i)$ of the state, and the control signal $u_i = u(t_i)$ from a trajectory of the system. We build on the widely-used *bounded* noise assumption and consider that the bounds $|x(t) - \tilde{x}(t)| \leq \eta$, $|\dot{x}(t) - \dot{\tilde{x}}(t)| \leq \eta^-$ hold for all $t \in \mathbb{R}_+$ and for some vector values $\eta, \eta^- \in \mathbb{R}^{n+}$. Here the absolute value and the comparison are conducted elementwise.

Problem Statement. Given the current noisy measurements of the state $\tilde{x}^{N+1} = \tilde{x}(t_{N+1})$, the trajectory T_N , a set $V \subseteq U$ of admissible control signals, a time step size $\Delta t > 0$, and a maximum number $T > N$ of time steps, compute an over-approximation of the reachable set at time $t_i = t_N + (i - N)\Delta t$ for all $i \in N[N+1, T]$. Further, compute a solution to the one-step optimal control problem where the desired behavior of the system is generally encoded as a *one-step cost function* $c : X \times U \times X \rightarrow \mathbb{R}$ that assigns preferences over the tuple of the current state, the current control input, and the corresponding state at the next time step.

Data-Driven Reachable Set Computation. DaTaReach provides closed-form expressions for over-approximations of the reachable set of unknown dynamical systems. It first uses the available data and the given side information to construct differential inclusions that contain the unknown vector field. Unlike existing work that provides state-independent differential inclusions, the constructed differential inclusions account for dependencies on the states and control signals. Then, DaTaReach provides closed-form expressions for over-approximations of the reachable set associated with these data-driven differential inclusions. Specifically, it builds on interval Taylor-based methods to over-approximate the reachable set of dynamics described by the differential inclusions. The closed-form expressions can incorporate the available side information, and increasing amount of data and richer side information provide tighter over-approximations.

Convex-Optimization-Based Data-Driven Control. The closed-form expressions enable convex-optimization-based, near-optimal, real-time, and on-the-fly control of unknown dynamical systems. DaTaControl computes *approximate solutions* to the one-step optimal control problem (2) through convex optimization relaxations. The convex relaxations are obtained by replacing the unknown future state with a control-affine linearization of the corresponding over-approximation of the reachable set. More specifically, DaTaControl achieves near-optimal and real-time control of the system through two convex relaxations of the one-step optimal control problem. The first relaxation, called optimistic control problem, minimizes the one-step cost over all possible trajectories in the over-approximation of the reachable set. The second relaxation, called idealistic control problem, minimizes the one-step cost over a specific trajectory, idealistically picked as the unknown evolution of the state, inside the over-approximation of the reachable set. We provide bounds on the suboptimality of the solutions of the optimistic and idealistic control problems with respect to the solutions of the one-step optimal control problem if the dynamics were known. Then, we theoretically show that the obtained bounds on the suboptimality become tighter with increasing amount of data and richer side information. We also provide a bound on the number of primitive operations required to terminate.

In our recent work [50], we generalize the above problem statement by solving N -step optimal control problem instead of the one-step control problem. Besides, we consider the larger class of nonlinear dynamics with polynomial dependency in the control inputs instead of control-affine. We also provide comparisons with state-of-the-art reinforcement algorithms in the Multi-Joint Dynamics with Contact (MuJoCo) simulation environment.

3.4.2 Physics-Constrained Neural Ordinary Differential Equations for Dynamical System Modeling. The solution approach described in the previous paragraph assumes certain regularity of the unknown dynamics given by the Lipschitz bounds. Due to such assumption, the learned model and the reachable set computation can sometimes be conservative. We propose to reduce the conservativeness of the prior approach and the requirement of regularity of the underlying dynamics via deep neural networks.

The central thesis of this line of work is that effective inclusion of a-priori knowledge into the training of deep neural network models of dynamical systems can greatly improve data efficiency and model generalization to previously unseen regions of the state space, while also ensuring that the learned model respects physical principles.

We consider arbitrary non-linear dynamics that can be expressed in terms of an ordinary differential equation (ODE) of the form $\dot{x} = h(x, u)$. The state $x: \mathbb{R}_+ \mapsto X$ is a continuous-time signal, the control input $u: \mathbb{R}_+ \mapsto U$ is a (possibly non-continuous) signal of time, and the vector field $h: X \times U \mapsto \mathbb{R}^n$ is assumed to be unknown and possibly non-linear. Here, n denotes the dimension of the state. We note that it is straightforward to explicitly include time dependency in the ODE representing the system dynamics, however, we omit it for clarity and notational simplicity. We assume that a finite dataset D of system trajectories – time-series data of states and control inputs – is available, in lieu of the system’s model. That is, we are given a set $D = \{\tau_1, \dots, \tau_{|D|}\}$ of trajectories $\tau = \{(x_0, u_0), (x_1, u_1), \dots, (x_T, u_T)\}$, where $x_i = x(t_i)$ is the state at time t_i , $u_i = u(t_i)$ is the control input applied at time t_i , and $t_0 < t_1 < \dots < t_T$ is an increasing sequence of points in time. We seek to learn a function to predict future state values $x_{k+1}, \dots, x_{k+nr+1}$, given the current state value x_k and a sequence of control inputs $u_k, \dots, u_{k+nr}$.

We propose to parameterize and to learn the unknown vector field h , as opposed to learning a prediction function that maps directly from a history of states and control inputs to the predicted next state. This choice to parametrize a model of the vector field is primarily motivated by the two following points: (1) A-priori knowledge derived from physical principles is typically most easily expressed in terms of the system’s vector field; (2) Recent work has demonstrated that parametrizing the vector field results in models that are able to approximate time-series data with more accuracy than approaches that directly estimate the next state. We thus build a framework capable of learning the vector field h from the dataset D of trajectories, while also taking advantage of a wide variety of potential sources of a-priori knowledge. The physics-based knowledge is incorporated as inductive bias in the model of the vector field via two distinct mechanisms.

Physics-Informed Model Structure. We represent the system’s vector field as compositions of unknown terms that are parametrized as neural networks and known terms that are derived from a-priori knowledge. We represent the vector field of the dynamical system as the composition of a known function – derived from a-priori knowledge – and a collection of unknown functions that must be learned from data.

$$\dot{x} = h(x, u) = F(x, u, g_1(\cdot), \dots, g_d(\cdot)) \quad (14)$$

Equation 14 Function Encoding Prior Knowledge on Model

Here, F is a known differentiable function encoding available prior knowledge on the system's model. The functions $g_1, \dots, g_d$ encode the unknown terms within the underlying model. The inputs to these functions could themselves be arbitrary functions of the states and control inputs, or even of the outputs of the other unknown terms.

Using the available training data, we learn the unknown functions $g_{\theta_1}, \dots, g_{\theta_d}$ in an end-to-end fashion. We parameterize $g_1, \dots, g_d$ by a collection of neural networks $g_{\theta_1}, g_{\theta_2}, \dots, g_{\theta_d}$, where $\theta_1, \theta_2, \dots, \theta_d$ are the parameter vectors of each of the individual networks. For notational simplicity, we define $\Theta = (\theta_1, \dots, \theta_d)$. For fixed values of the parameters Θ , we integrate the current estimate of the vector field of the dynamics using a differentiable ODE solver to obtain predictions of the future state, which are used to compute the loss function. Finally, assuming that F is differentiable, we can train the weights in Θ by automatically back-propagating through both ODEsolve and the function F via the adjoint sensitivity method.

Physics-Informed Constraints. In addition to using a-priori physics-based knowledge to dictate the structure of the neural network, we also use such knowledge to derive constraints on the outputs and the internal states of the model. Such constraints could, for example, encode known system equilibria, invariants, or symmetries in the dynamics. We note that while only a limited number of data points may be available for supervised learning, constraints derived from a-priori knowledge will hold over large sub-sets of the state space, and potentially over the state space in its entirety. Ensuring that the learned model satisfies the relevant constraints throughout the state space, while also fitting the available trajectory data, leads to a semi-supervised learning scheme that generalizes the available training data to the unlabeled portions of the state space. During the model's training, we enforce the physics-informed constraints on a collection of points in the state space, both labeled and unlabeled, via the augmented Lagrangian method.

3.5 Reinforcement Learning Methods for Learning Programmatic Policies

The two highlights of our work on this project were: (i) the PROPEL algorithm for programmatic RL, and (ii) the REVEL framework for RL with formally verified exploration.

3.5.1 PROPEL Algorithm. The PROPEL framework is based on three insights that enables integrating and building upon state-of-the-art approaches for policy gradients and program synthesis. First, we view programmatic policy learning as a constrained policy optimization problem, in which the desired policies are constrained to be those that have a programmatic representation. This insight motivates utilizing constrained mirror descent approaches, which take a gradient step into the unconstrained policy space and then project back onto the constrained space. Second, by allowing the unconstrained policy space to have a mix of neural and programmatic representations, we can employ well-developed deep policy gradient approaches to compute the unconstrained gradient step. Third, we define the projection operator using program synthesis via imitation learning, in order to recover a programmatic policy from the unconstrained policy space.

Concretely, PROPEL learns using neurosymbolic policies $h(x)$ of the form $f(x) + g(x)$, where $f(x)$ is a neural policy and $g(x)$ is a symbolic policy expressible in a domain-specific language. The set of symbolic policies is viewed as forming a proper subset of the set of neurosymbolic policies, and the discovery of an optimal symbolic policy is framed as a constrained optimization task. The learning process, an instance of *approximate functional mirror descent*, starts with an initial, suboptimal neurosymbolic policy. In each learning step, it iterates between:

- Applying a series of gradient updates to the current neurosymbolic policy. Importantly, these updates are based on the gradient of the neural component of the policy, and is thus an approximation to the true gradient in the space of neurosymbolic policies.
- Using a *projection* operation to distill this neurosymbolic policy into a programmatic policy, which is then lifted back into the neurosymbolic space. This projection operation is framed as an imitation learning problem that reduces to inductive program synthesis (a well-studied problem in the formal methods community).

3.5.2 Expanding PROPEL to the REVEL Framework. The subsequent REVEL framework expands PROPEL to accommodate monitoring mechanisms that check whether the policy violates a formal requirement, and shielding mechanisms that execute a failsafe policy if the learned policy is about to execute an unsafe action. In contrast to prior shielding-based approaches, the monitor and the shield in REVEL are updated as learning progresses. A key feature of the approach is that we repeatedly invoke a formal verifier from within the learning loop. Doing this is challenging because of the high computational cost of verifying neural networks. We overcome this challenge using a neurosymbolic policy representation in which the shield and the monitor are expressed in an easily-verifiable symbolic form, whereas the normal-mode policy is given by a neural network. The algorithm views the set of shields as being obtained by imposing a constraint on the general policy space. Starting with a safe but suboptimal shield, it alternates between: (i) safely lifting the current shield into the unconstrained policy space by adding a neural component; (ii) safely updating this neurosymbolic policy using approximate gradients; and (iii) using a form of imitation learning to project the updated policy back into the constrained space. Overall, the method admits efficient gradient-based learning as well as efficient updates to both the shield and monitor, and ensures safety without requiring expensive direct verification of neural networks.

4.0 RESULTS AND DISCUSSIONS

We have completed all project milestones with significant demonstrative progress toward our goal of developing and applying new methods, techniques and approaches to improve the assurance of learning enabled systems. With the incorporation of the F-16 challenge problem, we applied novel optimization techniques for learning machine learning models of an inner loop neural network controller with quantitative improvements in solver performance (300%) and test accuracy (20%). Using fuzzy learning methods, we generated highly accurate models of the LQR inner loop controller integrating both of these into our F16 Test Harness infrastructure.

Progress in the area of anomaly detection includes the identification of major weaknesses in the accuracy of autoencoder-based anomaly detection in images as a result of experimentation with multiple configurations of trained representations. We believe these results will contribute broadly in helping the research community better understand limitations and how to focus research efforts to improve the accuracy of anomaly detection. Early results using a novel “choice function” framework shows significant promise to develop robust, online safety policies developed neural network and fuzzy logic applied symbolic systems testing and model extraction techniques for verification for directed testing and challenge.

A framework for on-the-fly control of autonomous systems with unknown dynamics shows promise to provide next step, optimized recovery control of an autonomous system using only a few seconds of data. These algorithms may provide critical safety assurance for learning enabled systems. New approaches to learning policies that satisfy provable a temporal property have been explored in phase 1 and demonstrated on the CARLA challenge problem. We anticipate that this work can be applied to develop the F-16 challenge problem in phase 2.

We re-architected the F-16 model into a modular, component-based test harness to enable component and system level test and verification of multiple learning enabled and safety controllers with various plant and environmental models. We have begun to integrate new methodologies, algorithms and techniques into the test harness framework demonstrating promising results for the complex F-16 challenge problem. Our phase 1 effort concluded by demonstrating the successful application of these techniques to protect a learning enabled F-16 from a known faulty, initial condition scenario where a GCAS maneuver is initiated.

4.1 Assured Learning Milestones and Deliverables

Figure 12 summarizes our milestones and month they were delivered on the project.

Milestone		Month	Done
M1	Identify sample systems to use to drive experimental work.	6	6
M2	Perform goal-directed testing of the sample systems and collect analytical results.	18	18
M3	Model extraction support for the learning components in the sample system.	12	12
M4	Initial support for anomaly detection for sample CPS system.	12	12
M5	Support for open-world classification systems as part of sample CPS system.	30	28
M6	Support for learning some aspects of system dynamics in a physics-aware manner.	12	12
M7	Physics-aware dynamics learning integrated with other verification and testing work.	30	28
M8	Initial support for PPS-based assurance cases and inference rules.	12	12
M9	Support for more complex inference in PPS and additional types of evidence.	30	29
M10	Support for incorporating testing and run-time data into PPS.	18	18
M11	Support for automatically generating PPS proofs for system designs in an existing modeling language.	30	19
M12	Support for computing more complex safety metrics from PPS proofs.	30	29
M13	Initial demonstration of CSAF extensions for multi-agent simulation, goal directed testing and visualization	28	28
M14	Demonstration of CP 3.2 Collision avoidance challenge problem	28	28
M15	Demonstration of CP 3.2 Multi-Agent collision avoidance challenge problem	30	29
Planned Milestone Completion (Month)			
Milestone completion (Month)			

Figure 12 Assured Learning Milestones (Planned and Completed)

The project was organized into two phases that were aligned with the Assured Autonomy Program Challenge Problems. In our first milestone, M1, we identified the F16 sample system as the target of our challenge problems. This aligned our work with the Assured Autonomy “Air Domain” efforts in which we collaborated with TA-4 performer Boeing on setting up challenge problems to achieve program objectives. In phase 1 we selected the Ground Collision Avoidance Scenario challenge problem that was best aligned with Boeing’s Challenge Problem 2.5 Ground Collision Avoidance. Figure 13 provides the Phase 1 milestone schedule.

Task	Description	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
		2019						2020						
		1	2	3	4	5	6	7	8	9	10	11	12	13
1.0	Symbolic System Testing						M1							
2.0	Model Extraction for Verification												M3	
3.0	Anomaly Detection and Open World Classification												M4	
4.0	OPTION: Learning System Dynamics												M6	
5.0	PPS Semantics and Inference Rules												M8	
6.0	Integrate Data into PPS													
7.0	OPTION: Generation of PPS from Existing Design Artifacts													
8.0	OPTION: Advanced Measures and Metrics													
9.0	Program Meetings, Reporting and Travel						PI Mtg		PI Mtg				PI Mtg	

Figure 13 Phase 1 Assured Learning Schedule

In phase 2, following the Air Domain Workshop, DARPA, Galois and Boeing agreed to pursue the air collision avoidance challenge problem (CP) 3.2, which was extended to include multi-agent air collision avoidance scenarios. Figure 14 provides our phase 2 milestone schedule.

Task	Description	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
		2020						2021											
		13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1.0	Symbolic System Testing						M2										M13		M15
2.0	Model Extraction for Verification																		
3.0	Anomaly Detection and Open World Classification																		M5
4.0	OPTION: Learning System Dynamics																		M7
5.0	PPS Semantics and Inference Rules																		M9
6.0	Integrate Data into PPS						M10												
7.0	OPTION: Generation of PPS from Existing Design Artifacts																		M11
8.0	OPTION: Advanced Measures and Metrics																M14		M12
9.0	Program Meetings, Reporting and Travel							PI Mtg									PI Mtg		Brief

Figure 14 Phase 2 Assured Learning Schedule

We delivered phase 1 software and documentation deliverables in June 2020 along with interim briefings and demonstrations showing the team's progress. During Phase 2, DARPA established the Assured Autonomy Tool Portal for team members to aggregate and post tools developed on the program. In addition, per DARPA request, Galois submitted a Distribution Statement A Request (DISTAR) request to open source CSAF, which was approved. Table 2 summarizes the milestone deliveries of software and documentation for the entire project. All the software has been delivered as open source and provided in a packaged virtual machine along with demonstration videos and necessary documents for delivery to DARPA.

Table 2 Assured Learning Milestone Deliverable Summary

Milestone	Description	File Name / Link / Location
	Result / Deliverable	
M1	Identify sample systems to use to drive experimental work.	See Phase 1 Demo Briefing #1: AssuredLearningPhase2CSAF
Galois	Control Systems Analysis Framework AeroBench F-16 Model	Demo1.pptx CSAF ¹ F16 Model ¹
M2	Perform goal-directed testing of the sample systems and collect analytical results.	See CSAF Beta Demonstration CSAFBetaDemo_Sep2020.pptx
Galois Purdue	CP 2.5 GCAS results (Demo 1 video) Abstraction Refinement- Guided Training (ART) for Provably Correct Neural Networks	Galois_2021-01.25_AssuredLearning_Phase2_CSAF Demo_1.pptxA
M3	Model extraction support for the learning components in the sample system	FITRE ^{1,2} : Trust region method for training DNN

Purdue	FITRE: Fisher Informed Trust-region Method for Training Deep Neural Network LEC for F16 Inner Loop	Binary components for the FITRE F16 LEC ^{1,2} AA Major Accomplishment #1 (Appendix A)
M4	Initial support for anomaly detection for sample CPS system.	See slides 11-25 of June 2020 Monthly report: FA8750-19-C-0092_Galois_Monthly_Jun_2020.pptx AA Major Accomplishment #2 (Appendix A)
OSU	Anomaly Detection results for Cessna Grand Caravan CPS systems	
M5	Support for open-world classification systems as part of sample CPS system.	RIDAD ¹ Jupyter Notebook
OSU	<i>Recurrent network Implicit Distributions for Anomaly Detection (RIDAD): OODD anomaly detection for discrete events (F16 Autopilot)</i>	
M6	Support for learning some aspects of system dynamics in a physics-aware manner.	Phase 1 release of On-the-Fly-Control (Data-driven C2Opt controller for F16 elevator)
UT Austin	CONstrained Global Optimization of Lipschitz smooth functions (congol)	
M7	Physics-aware dynamics learning integrated with other verification and testing work.	Phase 2 release of On-the-Fly-Control ^{1,2} Integrated with CSAF/Jupyter NB See slides #29-34 of final briefing FA8750-19-0092_FinalBriefing_Jan22.pptx AA Major Accomplishment #4 (Appendix A)
UT Austin	On-the-Fly-Control integrated into CSAF	
M8	Initial support for Probabilistic Proof Spaces (PPS) based assurance cases and inference rules.	Phase 1 release of REVEL and PROPEL ¹
	REVEL framework for neurosymbolic reinforcement learning PROPEL Imitation-Projected Policy Gradient (IPPG), to the discovery of	

Rice /UT	formally assured programmatic policies.	
M9	Support for more complex inference in PPS and additional types of evidence.	See slides #35-42 of final briefing FA8750-19-0092_FinalBriefing_Jan22.pptx Phase 2 release of PROPEL ¹ AA Major Accomplishment #3 (Appendix A)
Rice/UT	End-to-End Differentiable learning of a programmatic model; learning a program that controls RL agent	
M10	Support for incorporating testing and run-time data into PPS	See Phase 1 Demo Briefing #2: IntegratedLECTestingPhase2 CSAFDemo2
Rice/UT	Used Core-RL to compute neural policy for F-16 GCAS Autopilot	
M11	Support for automatically generating PPS proofs for system designs in an existing modeling language.	See Galois_2021-01.25_AssuredLearning_Phase2_CoPilot Demo_3.pptx COPILOT ³
Galois	Augmented CoPilot with what4 library to prove properties about LEC streams	
M12	Support for computing more complex safety metrics from PPS proofs.	See slides #22-28 of final briefing FA8750-19-0092_FinalBriefing_Jan22.pptx DISTSpectrl loaded on VM
Purdue	DISTSpectrl Hierarchical RL Framework for multi-agent and adversarial robustness (Purdue)	
M13	Initial demonstration of CSAF extensions for multi-agent simulation, goal directed testing and visualization	See slides 3-6 of November 2021 Principal Investigator (PI) Briefing: FA8750-19-C-0092_Galois_PI_Meeting_October2021
Galois	CP 3.2 Multi-agent Airborne Collision Avoidance System (ACAS) Demo and Briefing	
M14	Demonstration of CP 3.2 Collision avoidance challenge problem	See slides 3-6 of November 2021 PI Briefing: FA8750-19-C-0092_Galois_PI_Meeting_October2021 See Jupyter NBs for all scenarios CSAF ¹ . F16 Model ¹
Galois	CP 3.2 ACAS demo with scenario generation	

M15 Galois	Demonstration of CP 3.2 Multi-Agent collision avoidance challenge problem	See slides #4-7 of final briefing FA8750-19-0092_FinalBriefing_Jan22.pptx
	CP 3.2 ACAS multi-agent demo (F16 intruder + balloon) with ACAS-Xu and FITRE LECs	See Jupyter NBs for all scenarios CSAF ¹ F16 Model ¹
Notes	¹ Open source on Github per DISTAR request # 34620 ² Integrated into CSAF on Github ³ Transitioned to existing Open Source repository on Github maintained by The National Aeronautics and Space Administration (NASA)	

In Table 3, a summary of the project Contract Data Requirements List (CDRL) deliverables is provided.

Table 3 CDRL Deliverables for Phases 1 and 2

Program Phase	CDRL Deliverables
Phase 1 (June 2020)	FA8750-19-C-0092_(A004 SP) - Software_Product_Specification_V1_June2020_v0.1 FA8750-19-C-0092_(A005 Commercial Off-the-Shelf (COTS)) - DI- TMSS-80527C_COTS_V1_June2020 FA8750-19-C-0092_(A006 SP) - Phase 1 Technical Report_June2020vFinal Phase 1 Software Delivery VM: assured_autonomy.ova
Phase 2 (December 2021)	FA8750-19-C-0092_(A004 SP) - Software_Product_Specification_Dec2021.docx FA8750-19-C-0092_(A006 SP) CSAF Documentation_Dec2021.docx FA8750-19-C-0092_(A005 COTS) - DI-TMSS- 80527C_COTS_FINAL_Dec2021.docx FA8750-19-0092_FinalBriefing_Jan22.ppts FA8750-19-C-0092_(A007 SP) - Final Technical Report_Jan2022_FINAL.docx Phase 2 Software Delivery VM: assured_autonomyFinal.ova

In discussion with DARPA, Galois incorporated the use of Jupyter Notebooks, in addition to videos as a convenient method to document tools and to provide both written and graphical results in a convenient location. Figure 15 provides an example Jupyter notebook delivered with the software and documented in the FA8750-19-C-0092_(A006 SP) CSAF Documentation_Dec2021 file.

```

Falsify Neural Network Controlled F16 GCAS

Based on demo in https://github.com/caffett/falsify\_neural\_f16gcas

import os
import sys

import numpy as np
from numpy import pi, deg2rad
from skopt import gp_minimize

# CSAF Imports
import csaf.system as csys
import csaf.config as cconf

# Analysis Objects/Functions
from falsify_bopt import F16SimulationFunction, EvaluatedFunction, attack, initial_space

# create a csaf configuration out of toml
my_conf = cconf.SystemConfig.from_toml("/csaf-system/f16_shield_config.toml")

# create pub/sub components out of the configuration
my_system = csys.System.from_config(my_conf)

def ground_collision_condition(cname, outs):
    """ground collision premature termination condition"""
    return cname == "plant" and outs["states"][11] <= 0.0

def gcas_simulation(initial_state, initial_time, tmax=35):
    """simulate GCAS system"""
    tspan = (initial_time, tmax)
    my_system.reset()
    my_system.set_state("plant", initial_state[:13])
    my_system.set_state("controller", initial_state[13:])
    trajs, passed = my_system.simulate_tspan(tspan, show_status=True,
        terminating_conditions=ground_collision_condition,
        return_passed=True)
    tlen = min(len(trajs["plant"].states), len(trajs["controller"].states))
    return trajs["plant"].times, np.hstack((trajs["plant"].states[:tlen], trajs["controller"].states[:tlen])), passed

## simulate
l, h = initial_space()
initial_state = np.random.uniform(l, h)
times, states, passed = gcas_simulation(initial_state, 0)

## attack
attack_traj = np.array(states)[0:1]
delta = (h - l)/2
attack_spaces = [attack_traj[0]-delta, attack_traj[0]+delta]
unsafe_states = attack(EvaluatedFunction(F16SimulationFunction(0, gcas_simulation)), attack_spaces, n_calls=10)

## unbind/disconnect system
my_system.unbind()

## print number of unsafe states found
print(f"Number of unsafe states found: {len(unsafe_states)}")

```

Figure 15 Exemplar Jupyter Notebook Developed on Assured Learning

4.1.1 Assured Learning Algorithms, Tools and Methodologies Results:

Figure 16 provides a comprehensive view of all the efforts developed in the Assured Learning project.

Assured Learning Integrated Toolchain (CSAF)

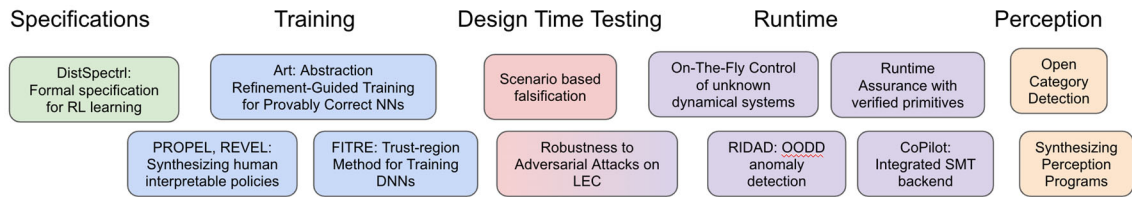


Figure 16 Summary of Assured Learning Tools, Algorithms and Frameworks Developed

4.1.2 F16 Challenge Problem 2.5 Inverted GCAS Test Results. We trained a NN to perform the shielded control of a GCAS maneuver using the F-16 test harness infrastructure. Initiating a GCAS from an initial condition in which the F-16 is inverted is a known failure mode for this particular controller. We demonstrated that applying new techniques can generate runtime shields to enforce safety for the GCAS maneuver of the Air Domain Challenge Problem 2.5. Figure 17 and Figure 18 highlight early phase 1 test results.

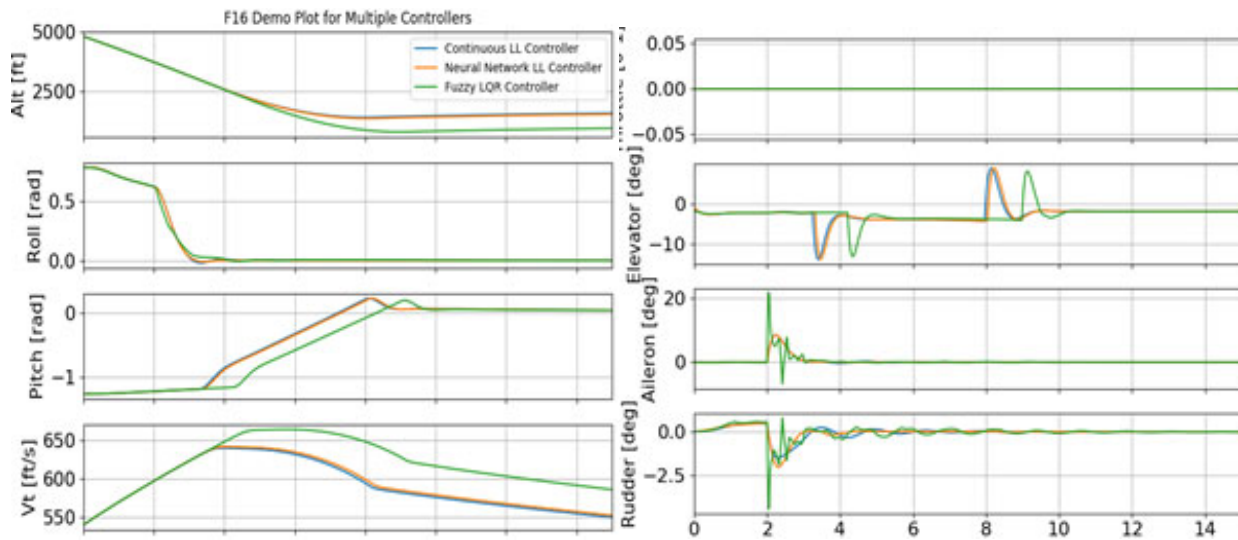


Figure 17 Comparison of Multiple Inner Loops for F16

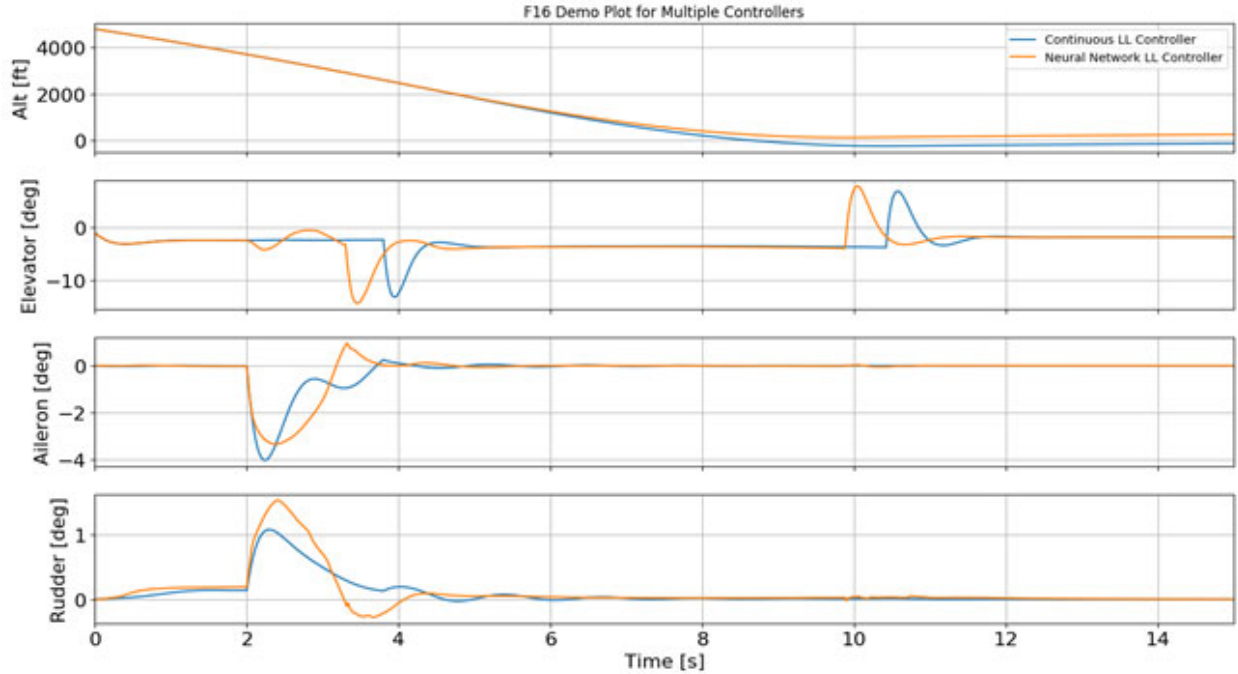


Figure 18 Classical vs NN Controller for Inverted GCAS

4.2 Results for Efficient and Scalable Training of ML Models for CPS Systems

4.2.1 FITRE Results and Experiments. We present results from our experiments with the proposed FITRE method (see Figure 19). We provide an in-depth analysis about the behavior of our proposed method, and compare it with well-tuned Nesterov- accelerated SGD as well as a quasi-Newton method, in the context of well-known CNNs.

Algorithm 1: FITRE Method

```
Input      :  
             - Starting point  $\mathbf{x}_0$   
             - Initial trust-region radius:  $0 < \Delta_0 < \infty$   
             - KFAC parameters: damping parameter ( $\gamma \geq 0$ ), moving average ( $0 < \theta < 1$ )  
Result:  $\mathbf{x}_t$  - direction to be used to update model parameters.  
foreach  $t = 0, 1, \dots$  do  
    Set the approximate gradient  $\mathbf{g}_t$  and Hessian  $\mathbf{H}_t$   
    /* Compute the approximated Inverse Fisher  $\times$  gradient, a.k.a  
       natural-gradient */  
    Obtain natural-gradient direction  $\mathbf{r}_t$ , as described in [13, 26]  
    Case 1: KFAC  
         $\eta_t = \arg \min_{\|\eta \mathbf{r}_t\| \leq \Delta_t} m(\eta \mathbf{p}_t) = \eta \mathbf{g}_t^T \mathbf{r}_t + \frac{\eta^2}{2} \mathbf{r}_t^T \mathbf{H}_t \mathbf{r}_t$   
         $\mathbf{s}_t = \eta_t \mathbf{r}_t$   
    Case 2: KFAC + Gradient  
         $\eta_t = \arg \min_{\|\eta \mathbf{r}_t\| \leq \Delta_t} m(\eta \mathbf{p}_t) = \eta \mathbf{g}_t^T \mathbf{r}_t + \frac{\eta^2}{2} \mathbf{r}_t^T \mathbf{H}_t \mathbf{r}_t$   
         $\alpha_t = \arg \min_{\|\alpha \mathbf{g}_t\| \leq \Delta_t} m(\alpha \mathbf{g}_t) = \alpha \mathbf{g}_t^T \mathbf{g}_t + \frac{\alpha^2}{2} \mathbf{g}_t^T \mathbf{H}_t \mathbf{g}_t$   
         $\mathbf{s}_t = \arg \min_{\mathbf{s} \in \{\eta_t \mathbf{r}_t, \alpha_t \mathbf{g}_t\}} m(\mathbf{s})$   
    Set  $\rho_t \triangleq \frac{h_t(\theta_t) - h_t(\theta_t + \mathbf{s}_t)}{-m(\mathbf{s}_t)}$ , ( $h_t(\cdot)$  are evaluated on the same mini-batch as  $\mathbf{g}_t$  and  $\mathbf{H}_t$  ).  
    if  $\rho_t \geq 0.75$  then  
        |  $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{s}_t$  and  $\Delta_{t+1} = \min\{2\Delta_t, \Delta_{max}\}$   
    end  
    else if  $\rho_t \geq 0.25$  then  
        |  $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{s}_t$  and  $\Delta_{t+1} = \Delta_t$   
    end  
    else  
        |  $\mathbf{w}_{t+1} = \mathbf{w}_t$  and  $\Delta_{t+1} = \Delta_t/2$   
    end  
end
```

Figure 19 The FITRE Algorithm and Method

Hardware and Software Platform for Experiments: All our simulations are performed on NVIDIA’s Tesla V100 GPUs configured with 16 Gigabytes (GB) global Random Access Memory (RAM) on CUDA 9.0 runtime platform. These machines are configured with Intel Xeon Central Processing Units (CPUs) with 192 GB of RAM. Our code is implemented in C++, and we primarily used cublas for General Matrix-to-Matrix Multiply (GEMM) operations. SGD results are executed on pyTorch 1.0.0 installation with python 3.1 as the front-end. The code base is available for download at [19].

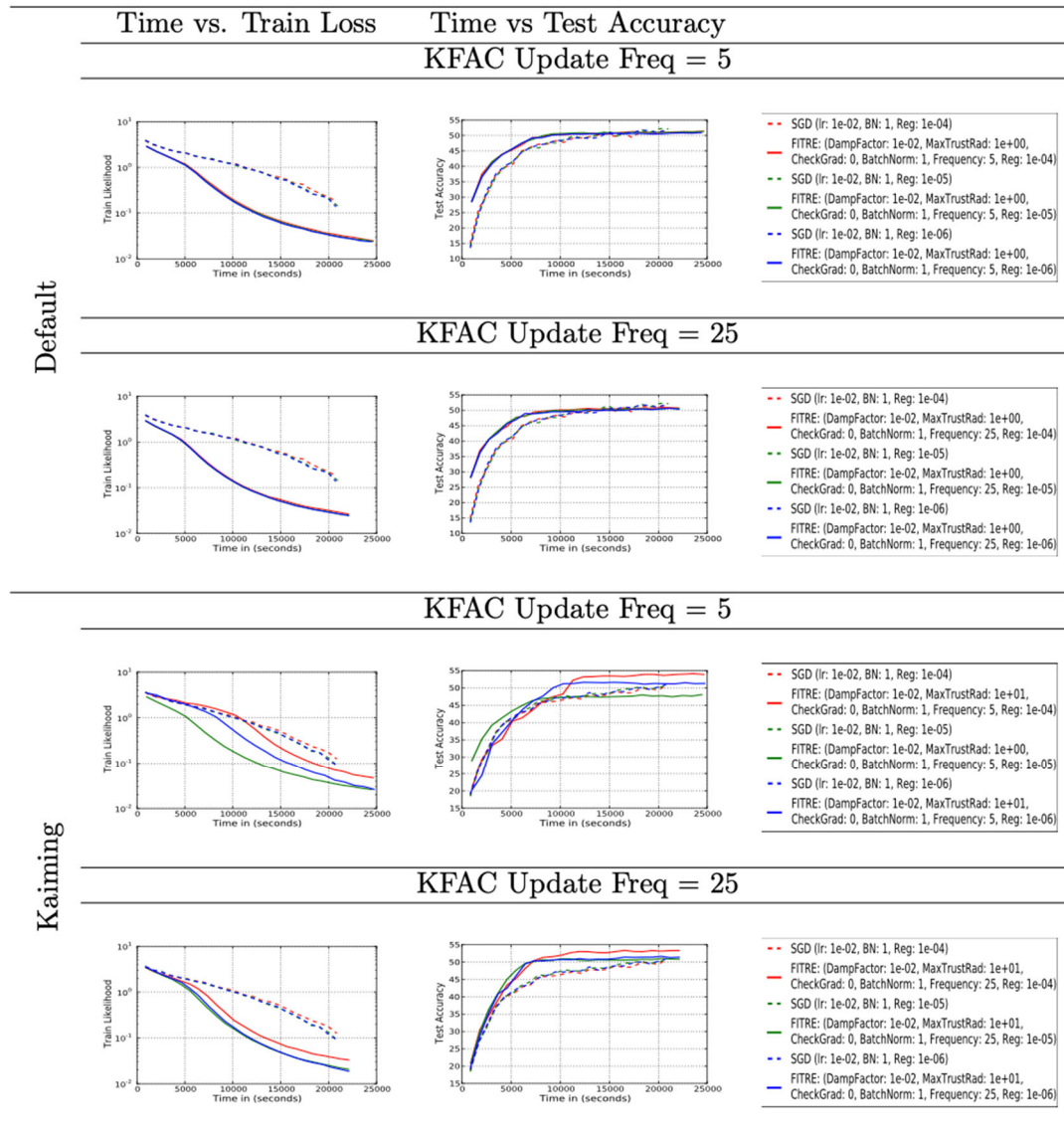
Datasets. We present results here on ImageNet [1] datasets for validating our method. Extensive experimental results on a wide range of networks and datasets are presented in our reports. ImageNet has 12288 features, 100K training samples and 9K testing samples.

Hyperparameter Tuning. Our proposed method has two easily tunable parameters—the damping parameter (γ) and maximum trust-region radius (δ). All experiments presented here use $\gamma \in \{1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^0, 1e^1, 1e^2\}$ and $\delta \in \{1e^{-2}, 1e^{-1}, 1e^0, 1e^1, 1e^2\}$.

Regularization term (λ) is set to the following values $\{1e^{-4}, 1e^{-5}, 1e^{-6}\}$. Trust-region radius, Δ , is capped by δ . It is doubled when ρ , the ratio of observed model reduction and expected model reduction, is ≥ 0.75 ; and it is halved when it is < 0.25 ; otherwise it remains the same. SGD uses learning rate as the hyper-parameter, which is in the following range $\{1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^0, 1e^1, 1e^2\}$. Both methods use the Nesterov-accelerated momentum term as 0.9. All the experiments are run for 50 epochs except for ImageNet dataset, where maximum number of epochs is set to 25. Mini-batch size is set to 200 for both the methods.

Plots presented in this section (Table 4) are selected as follows: SGD curves in all the plots always use the learning rate that yields highest test accuracy among all the learning rates. FITRE curves always use the selection of hyperparameters that yield the maximum average test accuracy.

Table 4 Comparison of VGG16 Using ImageNet Dataset



Convolutional Neural Networks. We report our results for VGG16 CNN for our validation purposes. Results on other networks (VGG11, VGG19) are available in our report. These are standard test networks and their details are available in literature.

Results on the ImageNet Dataset. Table 4 shows the plots for the ImageNet dataset using VGG16. We show the test errors plotted against wall-clock time and against number of epochs and we show the training loss against wall-clock time and against number of epochs. KFAC update frequency is set to 5 (mini-batches) for the first row; and it is set to 25 for the second row. All plots (in the first two rows of Table 4) use Default initialization, as defined in pyTorch, which is a uniform distribution. Corresponding results using Kaiming initialization [15] (on a high-level, this initialization is based on random Gaussian distribution) are shown in Table 4.

The following conclusions can be made from the plots in Table 4: (i) FITRE method minimizes the likelihood function to a significantly smaller value, compared to well-tuned SGD, and at any given wall-clock instance (FITRE method yields better Negative Log-Likelihood (NLL) value, compared to SGD), (ii) Kaiming initialization yields superior generalization errors, compared to Default initialization of the CNNs, (iii) contrary to expectations, KFAC update frequency of 25 yields better generalization errors, relative to more frequent updates, (iv) with increasing network complexity, VGG16 compared to VGG11, FITRE method yields significantly better generalization errors compared to SGD, showcasing its superior scaling characteristics compared to SGD; and (v) Default initialization is relatively immune to l2 regularization compared to Kaiming initialization.

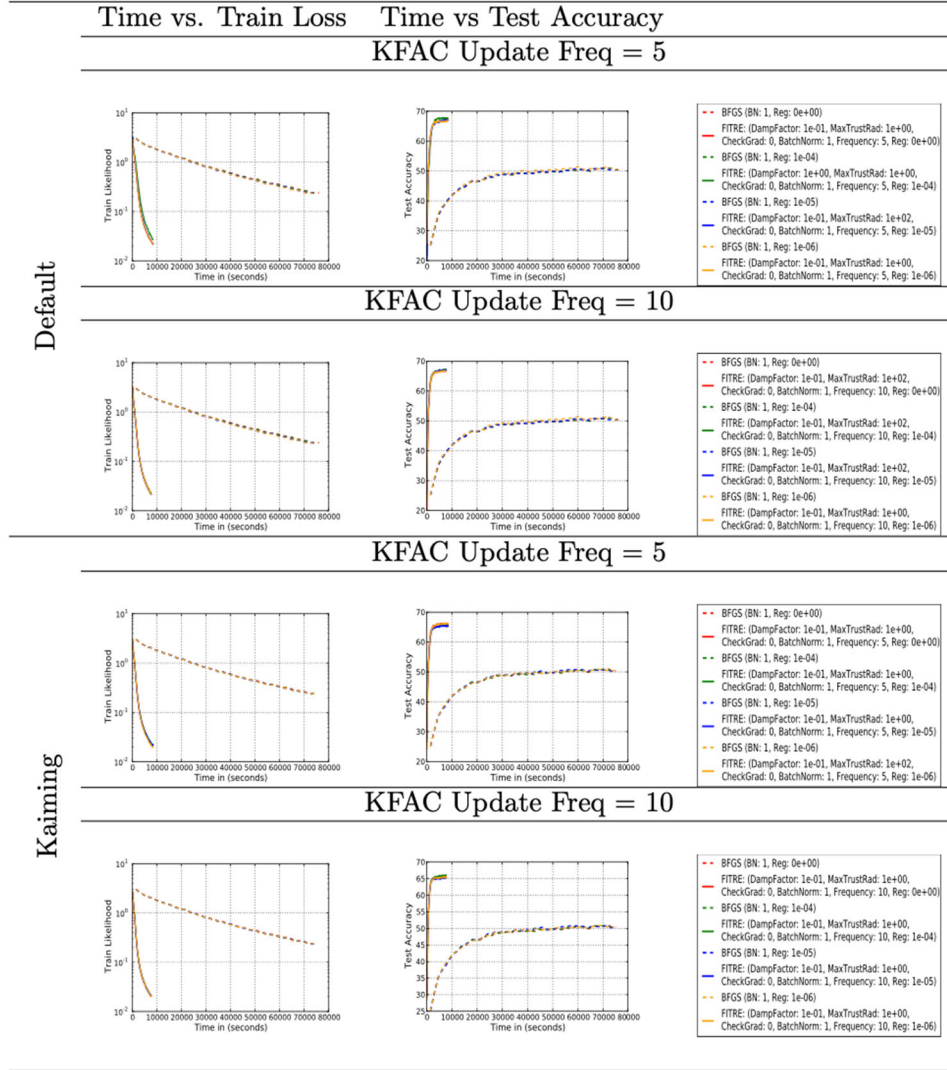
For VGG16 network with Kaiming initialization and KFAC update frequency of 25 we observe that to attain 50% test accuracy FITRE (with $1e-6$ regularization) takes ≈ 6500 seconds compared to ≈ 20500 seconds for SGD (for all regularizations used); a speedup of 3.2 over SGD. Furthermore, when regularization is set to $1e-4$, FITRE achieves 53.5% test accuracy whereas SGD fails to obtain similar accuracy. Similar arguments can be made for the VGG11 network as well. This shows that even though FITRE is computationally more expensive on a per-iteration basis, it yields significantly better results in shorter time, compared to SGD. This can be attributed to better descent direction (SGD’s gradient versus FITRE’s natural gradient) and an adaptive second-order approximated learning rate computation within the trust-region framework used by the FITRE method. Contrary to expectations, we notice that for VGG11 CNN and Default initialization, FITRE’s execution of 50 epochs takes less time compared to SGD for KFAC update frequency 25. FITRE makes two passes over the network (one forward and backward pass for gradient computation and another pass for Hessian-vector product computation used to compute the learning rate in the trust-region framework). One would expect that SGD is at least twice as fast as FITRE on the wall-clock time (on a per-iteration basis). We note that SGD’s pyTorch implementation uses auto-differentiation to compute the gradient of the given network, whereas our implementation of the FITRE method is R-operator based (as proposed by Perlmutter [32]). At a finer level, we note from our previous experience with the pyTorch platform [10, 20], that memory management on the GPU is not efficient. For example, pyTorch allocates and frees memory very often, and it tends to persist very little information on the device. Even though FITRE makes two passes over the network and computes inverses of smaller matrices at each layer of the network (for computing the

inverse of the KFAC block matrices) our implementation persists relevant information on the GPU memory. Coupled with our efficient implementation of the R-operator based Hessian-vector product, we can significantly reduce the computation cost associated with each mini-batch. In addition, our proposed method is a true stochastic online method in which there is no dependence on any part of the dataset other than the current mini-batch during its entire execution time, compared to state-of-the-art existing second-order methods [27, 30].

We notice that Default initialization is immune to regularization for both methods (FITRE and SGD). These two methods show negligible changes in training loss values (as well as generalization errors), while the FITRE method yields superior results compared with SGD for significant part of the execution. At the end of the execution, SGD tends to achieve similar generalization errors compared to FITRE, but on minimizing the training objective function (i.e. training loss) FITRE always achieves superior results. However, when using Kaiming initialization, based on random Gaussian distribution, for both the networks, we notice that regularization helps in achieving superior generalization errors for FITRE method compared to SGD. In all cases, FITRE method yields superior results when the underlying model does not use any regularization. Compared to the FITRE method, SGD is relatively immune to Kaiming initialization as well, as shown in plots in columns 1 and 2 of KFAC update frequency is a hyperparameter used to control the frequency with which the block matrix inverses are computed at each layer of the network. These block inverses are used to compute the natural gradient direction eventually for each mini-batch. Since these blocks approximate the Fisher matrix of the loss function, they are updated once every few mini-batches. Martens et. al. [13, 26] argues that more frequent updates of these block inverses makes them too rigid and may lead to overfitting. Using larger values for this update frequency has the effect of a regularizer on the underlying model, and it helps in avoiding overfitting. As an added advantage, this dependence of the FITRE method reduces its computation cost (note also that the computation of block inverses can be delegated to slave processing units, if available, further reducing the computation cost thereby decreasing the time for processing each mini-batch). This is also one of the reasons why our proposed method scales well with increasing network complexity. We note that for VGG16 (with Kaiming initialization), a larger and more complex network compared to VGG11, FITRE method yields superior generalization errors as well as minimizing objective function compared to SGD.

Comparison with quasi-Newton Method In this paragraph, we elaborate on the comparison of FITRE method against Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS), a widely known quasi-Newton method. Our Broyden Fletcher Goldfarb Shanno (BFGS) implementation is in pyTorch, and it uses a history of 20 prior gradients in generating the Hessian approximation used in computing the descent direction. For step size estimation we use cubic interpolation method as described in Section 3.4 of the book of Nocedal and Wright [31]. In the plots shown in Table 5, we describe the comparison results between FITRE and L-BFGS methods. For FITRE method, we keep the KFAC update frequency and regularization constant, and we select the best performing set of hyperparameters, yielding the highest test accuracy for comparison against the BFGS method.

Table 5 Comparison of VGG16 with a quasi-Newton Method.



From the plots in Table 5, we clearly notice that FITRE method is orders of magnitude faster, compared to BFGS method, $\approx 8\times$ faster in almost all cases. This is because FITRE uses our CUDA framework while BFGS method is implemented in pyTorch. Both these methods use GPUs for computation, but in a significantly contrasting manner. pyTorch is a general-purpose framework which is optimized for multi-user multi-process environment which inherently optimizes the GPU device usage for simultaneous use GPUs for many users. In that process, all the operations on GPUs are executed in a transactional manner. This means that whenever GPU computation is deemed necessary all the associated data is moved on to the GPU device and computation is initiated and once completed the results are moved back to the CPU memory space. Because of this reason, which is repeated hundreds of times over the course of the simulation, we notice a significant deterioration with the pyTorch version of BFGS solver (resulting in $8\times$ slower compared to FITRE method). In addition, FITRE method uses better tuning of thread block size for individual CUDA kernels, highly optimized implementation of convolution, activation and pooling

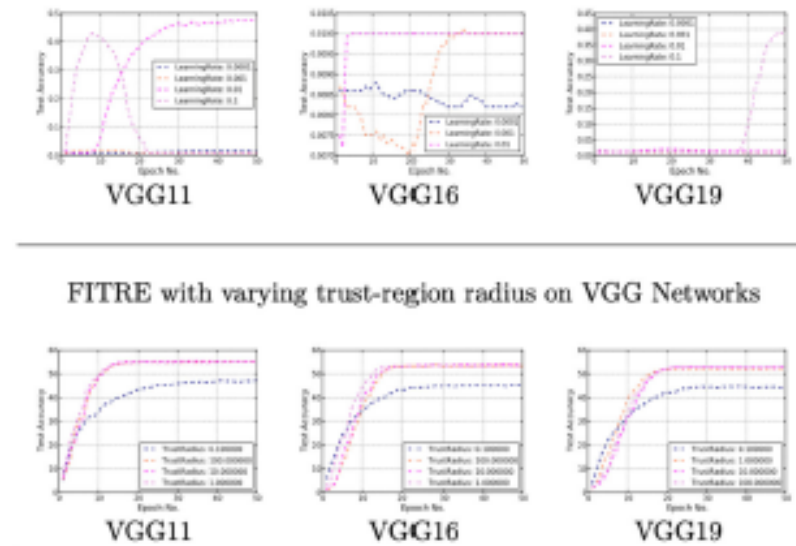
functions along with their first- and second-derivatives which are used during the computation of Fisher information statistics as well as Hessian-vector products.

In terms of test accuracy, we clearly notice that (irrespective of parameter initialization method and KFAC update frequency) FITRE method achieves significantly superior test accuracy compared to BFGS method. This can be attributed to the quality of natural gradient as estimated by the Kronecker approximated Fisher information matrix as well as the crude estimation of the Hessian used in estimation of the descent direction during the optimization of the objective function. Apart from the above-mentioned differences between BFGS and FITRE, we also notice that regularization term does not play a significant role in either improving the generalization error or time consumed during the course of the simulation for either of the optimizers.

Robustness Results. The FITRE method’s two main hyperparameters—damping term and maximum trust-region radius—can be easily estimated. Typical values for trust-region radius are between $1e-1$ and 10 . Similarly, damping parameter’s well-behaved region is between $1e-2$ and 10 . For most of the combinations, the FITRE method behaves in a consistent manner, and can yield almost identical results, as shown in Table 5. This behavior contributes significantly to the resilient behavior of the FITRE method w.r.t to minor changes in the hyperparameters **Error! Reference source not found..** However, we notice that SGD’s behavior changes significantly with slight changes in learning rate. This is one of the major challenges that a developer runs into with first-order methods, namely hyperparameter space that is difficult to tune, which we do not encounter with our FITRE method.

Table 6 shows the behavior of FITRE and SGD without regularization on Canadian Institute for Advanced Research (CIFAR)100 dataset. FITRE method uses an update frequency of 5 and “KFAC + gradient” option is turned off in these set of simulations. VGG networks in this table does not use batch-normalization function.

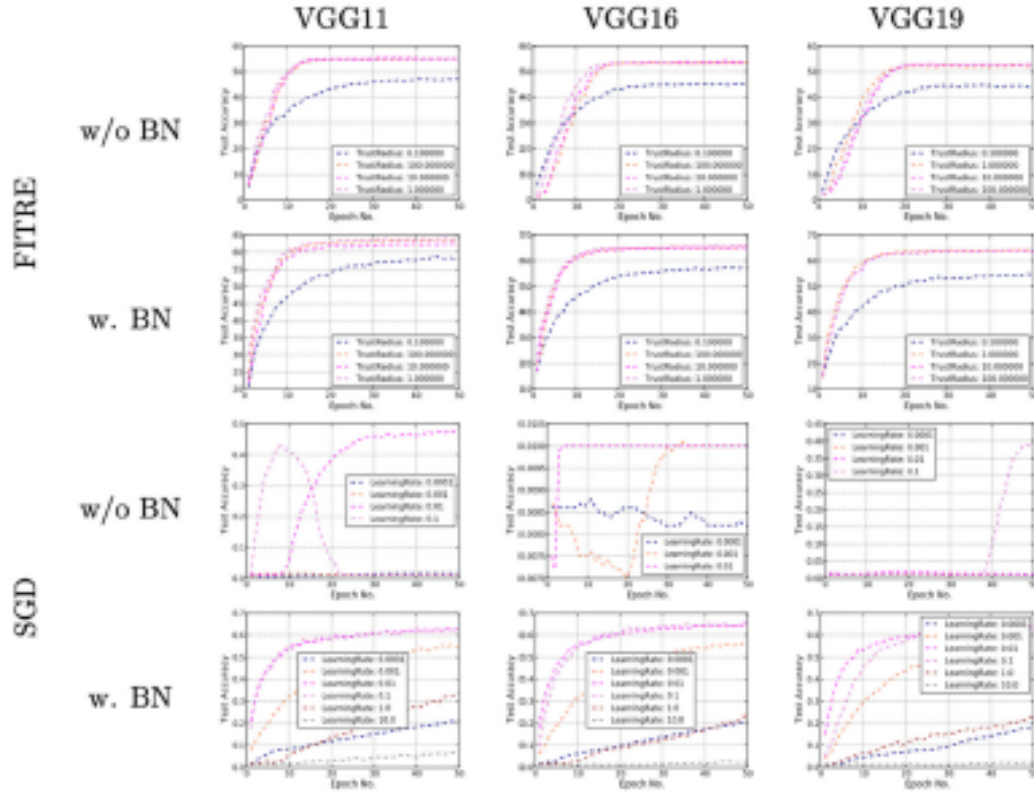
Table 6 SGD with various learning rates on VGG Networks



Invariance to Re-parameterization. With the computation of block inverses and ultimately the natural gradient itself, expectation of the inputs and outputs of the convolution layer is deeply embedded into the KFAC approximation itself. Because of this, additional batch normalizations are irrelevant to the FITRE method’s performance in minimizing the objective function. However, first-order methods like SGD need to have batch normalization layers embedded into the underlying model itself (typically right after the convolution layer). These results are presented in Table 7 (with and without Batch Normalization (BN)). We notice that without the use of batch normalization layers, the SGD method tends to diverge or does not make significant progress in optimizing the likelihood of the underlying model.

VGG networks in these experiments use 0 regularization. And for FITRE method we use the KFAC update frequency of 5 and use only the natural gradient direction as the descent direction (KFAC + gradient option is not used in these experiments).

Table 7 Reparameterization Invariance results for SGD and FITRE methods.



The FITRE method is robust, insensitive to small changes in its hyperparameters, as well as invariant to re-parameterization. Typically, implementation of batch normalization is computationally expensive, and as the network complexity increases, its execution time contributes significantly to the overall simulation time. There is no need for such layers in case of the FITRE method, which significantly helps reduce the processing time for each mini batch (and simulation time for the network).

FITRE In Learning Enabled Controllers. We demonstrate the use of our FITRE training procedure on a reinforcement learning based controller. Conventional RL controllers rely on techniques such as Proximal Policy Optimization and Trust Region Policy Optimization (TRPO) for optimizing reward in an RL framework. Another commonly used class of methods is called Actor-Critic, which uses a “Critic” to estimate the value function (either the action-value (the Q value) or state-value (the V value)) and an “Actor” to update the policy distribution. In this context, both the actor and critic are deep networks. Other methods in RL include Actor Critic with Kronecker-factored Trust Region (ACKTR), which combines actor-critic methods with trust region optimization, and Kronecker factorization.

In our RL implementation, we adopt an Actor-Critic (AC) framework, with FITRE used to train the actor and critic networks. We apply our RL technique to the F16 simulator and test it in different settings. In Figure 20 we show the performance of various methods. We first show various trajectories for each method and observe that FITRE-AC yields the highest reward (over 400) in a small number of iterations. In contrast, the commonly used PPO technique achieve a best-case reward of under 250 in a larger number of iterations.

We also compare FITRE-AC with other state of the art methods (ACKTR, PPO, and TRPO) and observe that FITRE-AC (red curve) yields the highest reward in this experiment, in the fewest iterations. This establishes FITRE not just as an excellent optimizer for training deep networks, but that its superior generalizability leads to better controllers in RL frameworks.

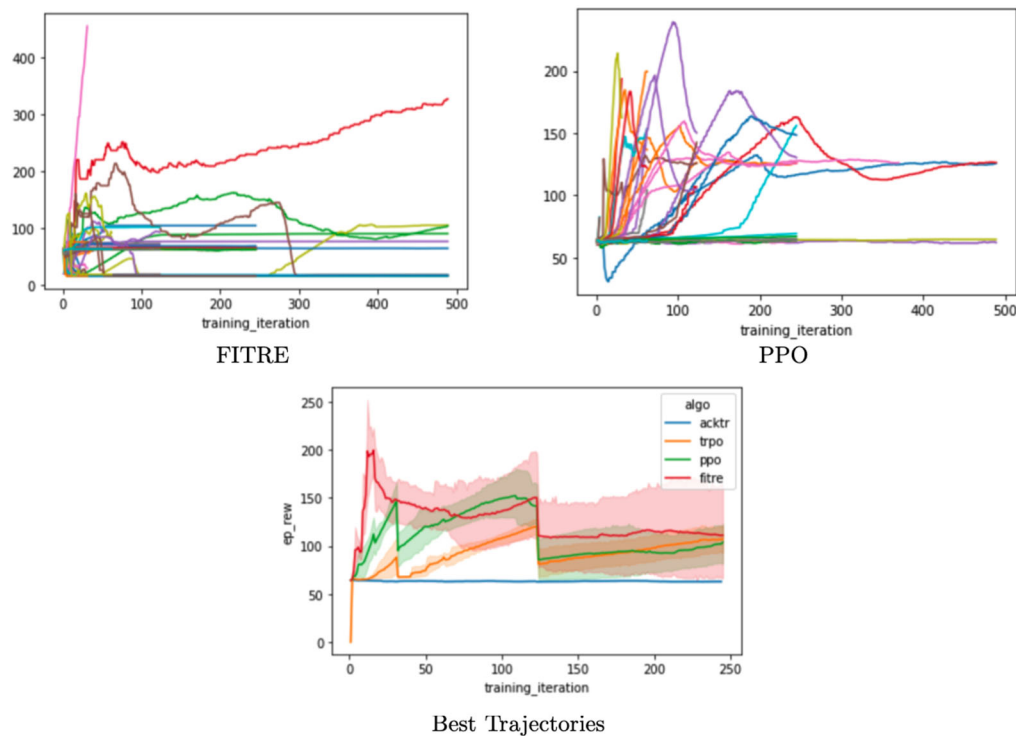


Figure 20 Reward functions from trajectories for Actor-Critic Reinforcement Learning using FITRE and Proximal Policy optimization (PPO).

We illustrate a snapshot from the trajectory of a simulated avoidance maneuver of a simulated F16 using FITRE-AC and PPO in Figure 21 the complete trajectory is available online.

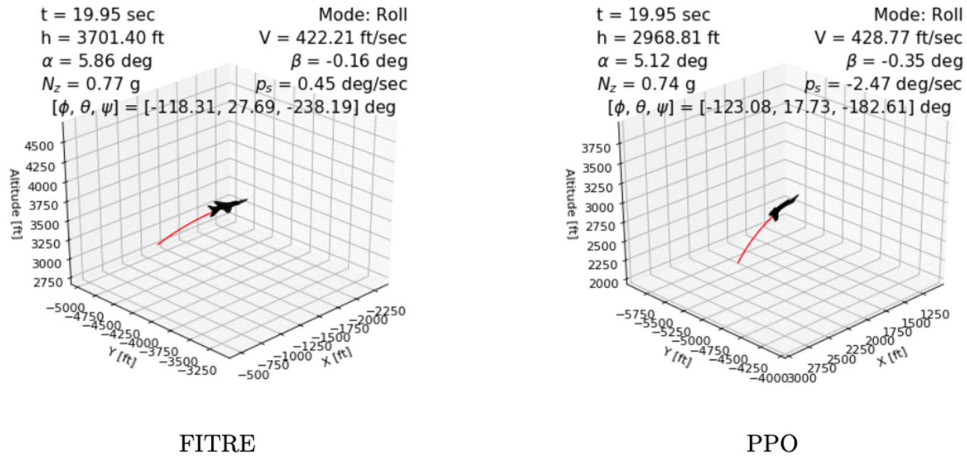


Figure 21 Demonstration of the RL controller on the F16 simulator using Actor-Critic-FITRE and Proximal Policy optimization (PPO) RL controllers

4.2.2 Robustness to Adversarial Attacks in LECs Results

Evaluation: We have considered a number of challenging problems ranging from autonomous robot navigation to F16 flight control. Table 8 shows two attack strategies and the effectiveness of our defense strategy. The first, random attack indicates the percentage of safety violations detected when initializing the state randomly around a sampled trajectory with respect to the chosen ϵ . For each benchmark, we randomly sampled 40K states within the bounds determined by ϵ and initialized the simulations accordingly. Column Bayesian Optimization (BO) attack shows the attack success rate when using BO to discover attacks; BO is a well-studied learning technique in black-box environments.

In Table 8 benchmarks are of the form, E-A where E is the Environment and A is the type of controller attacked.

Table 8 Experimental Results from Benchmarks

Benchmarks	dims (obs/state)	simu. traj. length	Attack ϵ	rand. attack	BO attack	defense succ. rate	attack improvement using shielded policy
Hopper-a2c	15/12	1000	0.001	1.29%	2.18%	91.40%	43.51%
Hopper-ppo				0.41%	5.27%	97.80%	27.97%
Hopper-trpo				1.71%	1.97%	97.90%	48.16%
HalfCheetah-a2c	26/17	1000	0.02	0.36%	9.43%	91.40%	34.46%
HalfCheetah-acktr				0.90%	14.54%	92.80%	65.23%
HalfCheetah-ddpg				1.43%	11.84%	88.20%	47.27%
HalfCheetah-ppo				0.38%	4.68%	97.80%	50.53%
HalfCheetah-sac				4.40%	5.88%	97.90%	62.12%
HalfCheetah-trpo				0.77%	4.83%	97.90%	49.02%
Ant-a2c	28/29	1000	0.0075	0.17%	1.26%	82.60%	46.20%
Ant-ddpg				0.42%	4.43%	94.30%	93.57%
Ant-ppo				1.47%	11.78%	88.70%	91.39%
Ant-sac				2.82%	14.83%	96.00%	94.02%
Ant-td3				5.79%	12.79%	93.60%	72.52%
Humanoid-ppo	44/47	1000	0.001	0.38%	2.00%	91.90%	22.93%
Pendulum-ddpg	2/2	200	*	0.03%	8.72%	100.00%	100.00%
4carploom-ddpg	7/7	1000	*	0.01%	3.82%	100.00%	94.71%
8carploom-ddpg	15/15	2000	*	0.72%	4.13%	100.00%	100.00%
Helicopter-ddpg	28/28	2000	*	0.00%	0.26%	100.00%	100.00%
F16-ppo	8/16	2000	*	0.01%	2.60%	96.40%	96.69%

* Use allowed variance of initial state which is given by the benchmark itself

Defense Results. We trained detectors with the states generated by our attack and learned auxiliary policies with the detector and corresponding safety reward. We measured the quality of our defense in two ways. First, we want to know how successful our detector and auxiliary policies are in preventing attacks on the original policy that would otherwise lead to a safety violation. Second, given a shielded policy, i.e., a policy that integrates the original, defense, and auxiliary policy, we measure how effective this combined policy is in reducing the number of unsafe states available to an attack, compared to the original policy.

To answer the first question, we run BO attacks on the original policy, employing our detector and shield policy to detect and avoid unsafe trajectories. Note that we may still find unsafe trajectories because not all attacks are necessarily preventable for reasons discussed below. The defense success rate on the PyBullet benchmarks ranges from 82.6% to 100%. On the simpler classical control benchmarks, the defense success rate can achieve 100% while the F16GCAS benchmark has a 96.4% defense success rate.

These results support our claim that ϵ -state perturbations used in adversarial attacks in these environments can be effectively mitigated. We note that our defense method does not provide verifiable guarantees of safety. There are two primary reasons for this: a) the possible existence of states in our attack which are not recoverable by any policy (e.g., a robot whose forward momentum will cause it to fall regardless of any possible shield action that may be taken); and, b) covariate shift due to our black-box setting that does not provide insight into environment dynamics, making it impossible to provide guarantees on detector accuracy during testing since the detector and auxiliary policy are trained only with the adversarial states discovered by previously seen BO attacks. In other words, even if the detector is accurate, it is also untenable to assert that the auxiliary policy (π_{aux}) trained by RL is always reliable due to variations while training.

The attack improvement using shielded policy column shows the percentage reduction in unsafe states discovered by a BO attack when applied to the shielded policy as compared to the original. The trajectories admitted by the shielded policy are more constrained than the original since the policy is derived as a mixture of both the original and the auxiliary (safety) policy. Consequently, we would expect a fewer number of unsafe states to be discoverable under this new policy when compared against a policy in which safety was not taken into account. Indeed, the results shown in the column justify this intuition. For all benchmarks, the use of a shielded policy reduces the number of unsafe states found by a BO attack by at least 22.93%. The use of this blended policy on the three Classical Control tasks demonstrates 100% empirical robustness, while the F16 benchmark is very close behind at 96.69%. Several of the PyBullet benchmarks such as Ant-ddpg and Ant-sac show similar improvement.

4.2.3 Performance of the Shielded Policy. The experiments in Figure 22 investigate the impact of considering safety (and thus shield interventions) on performance. We randomly initialized the simulation around the attacked trajectory within a range bounded by ϵ . For each benchmark, we run a simulation 1000 times and count the (normalized) average return yielded with the original policy and shielded policy. The y-axis is the trajectory return scaled by a constant shared among each environment’s benchmarks. It represents the performance of the corresponding policy (i.e., moving fast with low electricity cost). The result in Figure 22 supports our claim that the cost of shield interventions is typically modest, on average only 5.78% variance compared to the performance exhibited by the original.

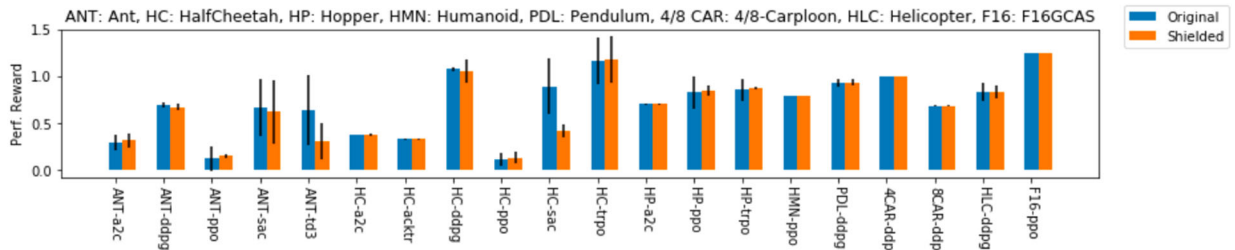


Figure 22 Average Reward for Original Policy and Shielded Policy

4.2.4 Results for Scalable Synthesis of Verified Controllers

Evaluation. We have applied the verification algorithm on various stochastic transition systems, whose dimensionality ranges from 2 to 896. We associate safety constraints with each benchmark, synthesizing a verified linear controller family that seeks to guarantee these properties hold, and use that family to train a neural network with additional performance (a.k.a. liveness) objectives. The resulting system consists of a performance-sensitive neural network trained with the awareness of safety constraints, coupled with a safety shield represented by the linear controller family. We train the neural network controller using Proximal Policy Optimization [34], a standard training technique used in reinforcement learning.

We evaluate our algorithm on 24 benchmarks. There are 6 base benchmarks - Pendulum, Cartpole, Cartpole, Carplatoon, and Helicopter, DroneInWind. The DroneInWind environment is time-variant because we allow the angle and the strength of the wind in the environment to change over time. We also consider stacking environment variants of these benchmarks named as n-B for assessing the effectiveness of our approach as dimensionality increases; here, n is the stacking depth and B is one of the six base benchmarks. To make the experiments not simply exploit the safety characteristics discovered for the base program, each stacked layer is defined with a randomly injected offset that makes every stacked element different from every other one.

environment to change over time. We also consider stacking environment variants of these benchmarks named as n-B for assessing the effectiveness of our approach as dimensionality increases; here, n is the stacking depth and B is one of the six base benchmarks. To make the experiments not simply exploit the safety characteristics discovered for the base program, each stacked layer is defined with a randomly injected offset that makes every stacked element different from every other one.

We train a neural controller using the safety guarantees captured by our verified linear controller family. Table 9 demonstrates the effectiveness of our approach on overall network performance. First, although the LQR controller is verified to be safe, it can perform poorly when performance objectives are taken into account. However, using it as a shield for a performant neural controller can realize both performance and safety benefits. The table also presents normalized performance characteristics of the shielded controller relative to the base PPO algorithm without augmentation of a safety planner and the LQR family that is implemented without performance objectives. Thus, numbers greater than one in the column labeled Shield/PPO indicate that the controller trained in conjunction with the safety planner outperformed the PPO-only trained algorithm. A similar interpretation holds for the column labeled Shield/LQR. While it is not surprising that controllers trained with both safety and performance (Shield) would outperform those that are only aware of safety (LQR), it is notable that the shielded controller has a higher performance reward than the PPO-trained controller on 19 of the 24 benchmarks. The last two columns indicate the number of safety violations encountered - PPO trained networks exhibited a non-negligible number of safety violations on every benchmark; since our verification algorithm was able to generate a provably safe shield for each benchmark, the safety-augmented controller exhibited no violation in any of the benchmarks we considered.

Table 9 Effectiveness of the safety planner on training safe and performant networks

Benchmarks	Abbr.	State dim	Action dim	Shield/PPO	Shield/LQR	PPO vio.	Shield vio.
Pendulum	PDL	2	1	2.65	8.59	1437	0
Cartpole	CTP	4	1	1.36	3.54	959	0
DronelnWind	DIW	6	2	3.35	4.54	864467	0
Carploon	CPL	15	8	1.83	30.58	69	0
Oscillator	OSC	18	2	1.37	6.79	3	0
Helicopter	HLC	28	6	1.04	1.49	30	0
2-Pendulum	2-PDL	4	2	1.77	2.84	2375	0
2-Cartpole	2-CTP	8	2	0.64	2.59	1775	0
2-DronelnWind	2-DIW	12	4	3.14	4.00	863053	0
2-Carploon	2-CPL	30	16	0.76	11.84	1137	0
2-Oscillator	2-OSC	36	4	1.18	3.33	46	0
2-Helicopter	2-HLC	56	12	1.07	1.48	277	0
4-Pendulum	4-PDL	8	4	2.68	2.34	4736	0
4-Cartpole	4-CTP	16	4	0.60	2.21	3529	0
4-DronelnWind	4-DIW	24	8	3.61	4.08	1748560	0
4-Carploon	4-CPL	60	32	0.60	15.94	1863	0
4-Oscillator	4-OSC	72	8	2.20	3.77	150	0
4-Helicopter	4-HLC	112	24	1.17	1.33	405	0
8-Pendulum	8-PDL	16	8	1.28	1.93	11305	0
8-Cartpole	8-CTP	32	8	1.17	2.55	12680	0
8-DronelnWind	8-DIW	48	16	2.11	2.19	3551103	0
8-Carploon	8-CPL	120	64	0.75	12.50	7250	0
8-Oscillator	8-OSC	144	16	1.99	3.29	579	0
8-Helicopter	8-HLC	224	48	1.33	1.24	1388	0

4.2.5 DistSPECTRL Results.

Evaluation. Our experiments aim to validate the claim that the use of a distributed task monitor helps achieve synchronization during the deployment of multiple agents on a range of specifications. In addition, to emphasize the need for distribution of task monitors to alleviate the state space explosion caused by mixing local and global specifications, we include experiments with Spectrl applied to a centralized controller. As a baseline comparison, we choose to run our algorithm without giving policies access to the monitor state (no mon). These are trained with the same shaped reward as DistSpectrl and monitor transitions are set to cycle between all transitions until an available one is found. This enables the baseline to traverse as deep into the task monitor as possible without needing to decide between transitions.

Environment. Our first set of experiments are done on a Two-Dimensional (2D) Navigation problem with $N = 3$ agents. The observations ($S \in \mathbb{R}^2$) used are coordinates with respect to a goal dependent on the agent. The action space ($A \in \mathbb{R}^2$) provides the velocity of the agent in the space. Agents are initialized in a line below their reference goals.

The second set of experiments were built in a similar fashion to modern Three-Dimensional (3D) benchmarks and represent particle motion in space. We train multiple agents ($N = 3$) in the 3D space ($S \in \mathbb{R}^3$) with a 3D action space ($A \in \mathbb{R}^3$) to show the scaling potential of our framework. Agents are initialized below the goal in the X-Y plane and with a random Z coordinate sampled between (2, 3).

2D Navigation. The evaluated specifications are a mix of local and global objectives.

$$\begin{aligned}
\phi_1 &= \text{reach}_{g1}(5, 0); \text{reach}_{g1}(0, 0) \\
\phi_2 &= \phi_1; \text{reach}_{g1}(3, 0) \\
\phi_3 &= \text{reach}_{lo}(5, 0); \text{reach}_{g1}(0, 0); \text{reach}_{g1}(3, 0) \\
\phi_4 &= [\text{reach}_{lo}(3, 0) \text{ or } \text{reach}_{lo}(5, 10)]; \phi_3 \\
\phi_{ex} &= \text{reach}_{gl}(10, 10); \\
&\quad \text{or } \text{reach}_{lo}(3, 0); [\text{reach}_{lo}(10, 10) \text{ or } \text{reach}_{gl}(5, 0)]
\end{aligned}$$

3D Environment: the following specification is considered for X-Y-Z coordinates:

$$\phi_a = \text{reach}_{lo}(5, 0, 0); \text{reach}_{g1}(0, 0, 0); \text{reach}_{g1}(3, 0, 0)$$

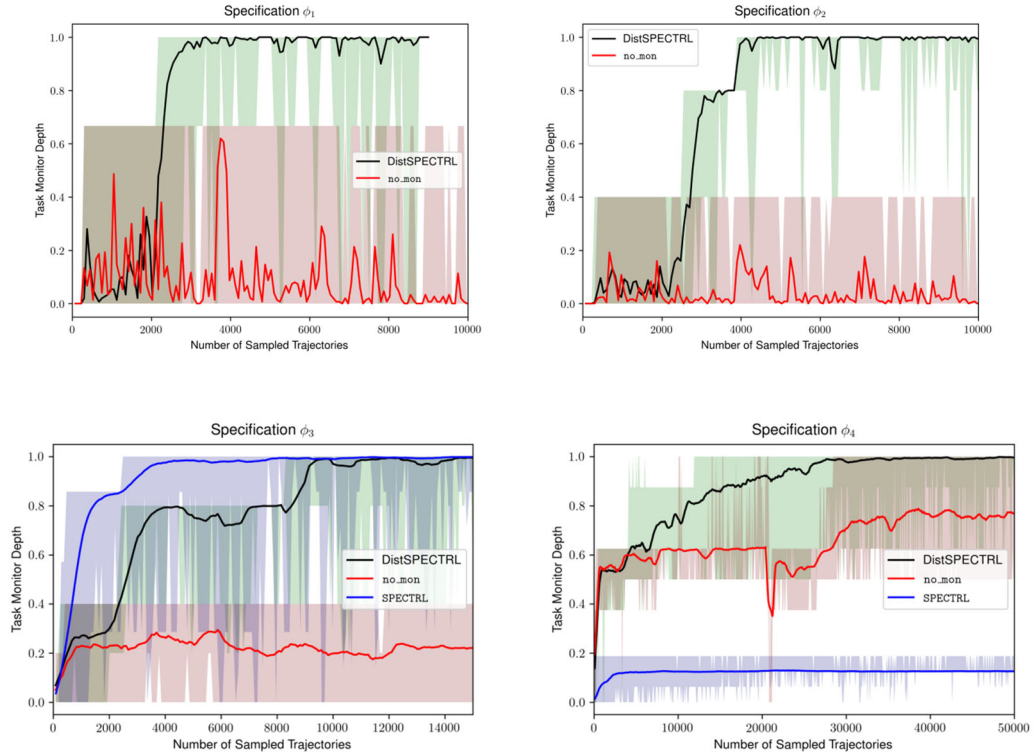


Figure 23 Mean Task Completion Rates on Specifications ϕ_1, ϕ_2, ϕ_3 and ϕ_4

Results. The experiments in Figure 23 demonstrate execution when the task monitor predicates have access to the entire system state. This provides agents with information sufficient to calculate global predicates for task monitor transitions. The normalized task monitor depth is reported with value from 0 being no sub-task completed to 1.0 being the entire specification satisfied. For ϕ_{ex} the overall satisfaction percentage is shown since depth-based metrics are not ideal for branching specifications of different lengths.

While Spectrl has been shown to be more effective [18] than existing methods for task specification, the further utility of the monitor state in enhancing coordination between agents is clearly evident in a distributed setting. For example, our baseline for ϕ_3 in Figure 23 is unable to get beyond the first of the three sequential tasks whereas the monitor state enables the policy to learn how to switch objectives. We see that the task monitor state is essential for coordination as our baseline is able to complete the local tasks (by exhaustively going through the list of possible transitions) but global task completion requires higher levels of synchronization.

From Table 10 we see that upon convergence of the learning algorithm, the agent is able to maintain a nearly 100% task completion rate for our tested specifications, a significant improvement in comparison to the no mon case, showing the importance of the task monitor as part of a multi- agent policy.

Table 10 Task Completion Percentage on Convergence for N=3

Spec.	DistSPECTRL	no_mon	SPECTRL
ϕ_1	99.62	00.93	-
ϕ_2	99.30	01.46	-
ϕ_3	98.63	64.77	99.63
ϕ_4	99.39	58.64	12.59
ϕ_{ex}	93.16	74.07	-
ϕ_a	80.16	36.92	-

Benefits of Distribution Over Centralization. The centralized Spectrl graphs (blue curves in Figure 23) show that while distribution may not be necessary for certain specifications with few local portions (ϕ_3), concatenating them (ϕ_4) will quickly lead to learning difficulties. This difficulty is due in large part to state space explosion of the task monitor in these cases as is apparent by the significantly better performance of our distributed algorithm.

Scaling to Larger State Spaces. The results in Figure 24 show promise that the DistSpectrl framework can be scaled up to larger dimension tasks as well. The 3D environment results exhibit similar behavior to the 2D case with the no mon showing difficulty in progressing beyond the local tasks in the larger state space with sparser predicates.

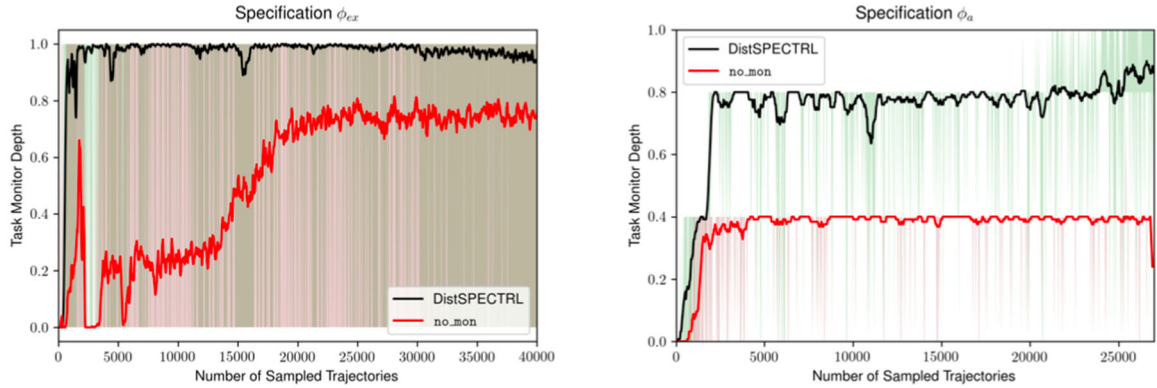


Figure 24 Specification satisfaction percentages (left) for ϕ_{ex} . Mean task completion (right) rates for the 3D Navigation experiments on specification ϕ_a

4.3 Open Category Detection and Anomaly Detection Results

Below we overview the main results of our two investigations into anomaly detection. These results address project milestones M1, M4, and M5.

4.3.1 Open Category Detection for Deep Image Classification. Our experiments were designed to shed light on three primary research questions (RQ).

- RQ1: Which methods of representation learning give the best representations (as measured by oracle scoring function)?
- RQ2: How big is the gap between oracle AD and existing anomaly scoring methods?
- RQ3: How big is the gap between oracle representations (measured with oracle scoring function) and existing representation learning methods (also measured with oracle scoring function)?

We evaluate open category performance on the Modified National Institute of Standards and Technology (MNIST), CIFAR10 and CIFAR100 datasets. The MNIST digit data set consists of ten classes of hand-written digits, and each instance is a 28x28 monochrome image. The CIFAR10 and CIFAR100 datasets consist of 10 and 100 classes of real-world objects. Each image is a 3-channel 32x32 RGB image. For MNIST and CIFAR10, we select 6 classes to be known and 4 classes to be unknown. For CIFAR100, we select 80 classes to be known and 20 classes to be unknown. All experiments were averaged over 5 runs, where each run used a different random partition of known versus unknown classes.

Table 11 shows an example experiment for CFAR100 for oracle scoring functions on top of different learned representations. In this example, we see that the discriminative representation contains significantly more information for open-category detection compared to the generative representation. This was somewhat surprising, since it is reasonable to expect a discriminative representation to filter out information that could be useful for distinguishing unknown classes. We do see a small amount of improvement for

the combined/hybrid representation that includes both the discriminative and generative representations. However, most of the power appears to come from the discriminative portion. Similar results were observed for both CFAR10 and MNIST. This provides evidence for RQ1 that discriminative representations appear to be more powerful than the generative methods that we have tried so far.

In Table 11 Table 11 Each row corresponds to an oracle scoring function (i.e. anomaly detector) applied to the representation learned by a discriminative classifier (deep network), a generative model (autoencoder), and the combination of discriminative/generative representations. The numbers Acc./AUC correspond to the accuracy of detecting unknown classes and the area under the Receiver Operating Characteristic (ROC) curve respectively.

Table 11 Example experimental results in CFAR100.

	Discriminative Acc./AUC	Generative Acc./AUC	Combined Acc./AUC
Random Forest	0.77 / 0.86	0.62 / 0.76	0.78 / 0.88
K nearest neighbors	0.78 / 0.87	0.67 / 0.77	0.79 / 0.88
Multilayer Perceptron (MLP)	0.67 / NA	0.64 / NA	0.74 / NA

Table 13 and Table 13 give example experiments for CFAR10 and CFAR100 respectively. These experiments all involve the combined/hybrid representation and different scoring functions. Each row corresponds to a different combination of oracle vs. non-oracle (nominal) representations and scoring functions. For CFAR10 we see that including an oracle scoring function but keeping a nominal representation leads to an AUC improvement from 0.787 to 0.983, which is a significant improvement. Further, by adding an oracle representation to the oracle scoring function we increase performance further to 0.977. For CFAR100, the corresponding sequence of improvements is from 0.724 to 0.828 to 0.857.

For both datasets we see that there can be big gains from using an oracle scoring function, indicating there is significant room for improvement over nominal methods. In addition, there are non-trivial additional gains from using an oracle representation. Similar observations hold for MNIST, but all performance measures at higher AUC levels due to the easier nature of that dataset. These results provide evidence for RQ2 and RQ3 that suggests there is significant headroom to improve.

Table 12 Example experimental results in CFAR10

Representation	CIFAR10		Z_{hybrid}	
	AD Treatment	Anomaly Measure	AUROC	$\pm$ 95% CI
Nominal	Nominal	Max $C_{\theta}(x)$	0.787	0.009
Nominal	Oracle	$g(x)$ (binary)	0.983	0.004
		$h(x)$ (multiclass)	0.851	0.011
Oracle	Oracle	$g(x)$ (binary)	0.997	0.001
		$h(x)$ (multiclass)	0.933	0.008

In Table 12 example experimental results in CFAR10 for different combinations of oracle and non-oracle (nominal) settings. The Representation (AD Treatment) column indicates whether an oracle or nominal representation (scoring function) is used.

Table 13 Example experimental results in CFAR100

Representation	CIFAR100		Z_{hybrid}	
	AD Treatment	Anomaly Measure	AUROC	$\pm$ 95% CI
Nominal	Nominal	Max $C_{\theta}(x)$	0.724	0.009
Nominal	Oracle	$g(x)$ (binary)	0.825	0.012
		$h(x)$ (multiclass)	0.828	0.014
Oracle	Oracle	$g(x)$ (binary)	0.845	0.012
		$h(x)$ (multiclass)	0.857	0.011

In Table 13 example experimental results in CFAR100 for different combinations of oracle and non-oracle (nominal) settings. The Representation (AD Treatment) column indicates whether an oracle or nominal representation (scoring function) is used.

4.3.2 Results from OOD Detection: Benchmarks and Strong Baseline.

Unless otherwise specified, our experiments use an ensemble of 5 RIQN models and a sampling size of 8 per ensemble member. We average all of our performance metrics over 1,000 test trajectories for each anomaly type investigated.

AUC Performance. Once anomaly scores have been calculated for each time point of a test sequence as described above, we can compute the AUC of those scores with respect to the ground truth knowledge of when an anomaly occurred. Table 14 shows results for detection horizons $\Delta = 1$ and $\Delta = 10$ for the different environments averaged over the 4 different anomaly types. We see that, with respect to AUC averaged over anomaly types, the performance for $\Delta = 1$ and $\Delta = 10$ are quite similar and that the baseline RIQN approach is quite strong for these benchmarks. However, there is still apparent room for non-trivial performance improvements via other techniques.

Table 14 Anomaly detection results

Environment	Policy	Policy Performance			$\Delta = 1$		$\Delta = 10$		
		Nominal							
Env	Anomalous								
Env	AUC	Delay	FAR	AUC	Delay	FAR			
Acrobot	DQN	-78.78	-95.8	95.1	2.5	0.024	93.2	11.4	0.029
CartPole	DQN	500	430.8	91	2.3	0.035	93.9	11	0.04
LunarLander	DQN	213.55	139.2	91	3.3	0.048	91.8	12	0.05
Ant	TD3	3297.21	2163.6	94.4	2.1	0.03	96	11	0.037
HalfCheetah	TD3	2820.91	2062	92	2.3	0.05	91.6	11.1	0.048
Hopper	TD3	2678.05	1876.6	91.6	2	0.04	91	11	0.041
Walker2D	TD3	2240.81	1634.2	92.2	2	0.044	90.5	11.1	0.04
Fixed Airspeed	-	201.25	192.04	92.3	4.2	0.03	93.3	15.1	0.038
Fixed Altitude	-	265.99	185.28	91.1	3.7	0.07	93.2	14	0.068
GCAS	-	225.16	201.42	91	4.5	0.04	90.4	13.3	0.043

Detection Delay. In many applications, it is important to not just provide anomaly scores, but to also raise an alarm when an anomalous shift in dynamics is suspected. The alarms should have minimal delay to allow for time-critical interventions when necessary. A small delay, however, must be traded-off with the false alarm rate (FAR), which must be sufficiently low for practical applications. Here, we use the sequence of anomaly scores for an observation sequence to produce alarms using the classic cumulative sum (CUSUM) method. We used a CUSUM threshold of 0.01 and drift of 0.0018. The threshold sets the amplitude value for the change in the data, and the drift term prevents any change detection in the absence of change. Quantitative results for the average delay and FAR across anomaly types for each environment are reported in Table 14. Note that larger Δ values tend to have larger delays unless they provide significantly better anomaly scores. This is the case when considering performance averaged across anomaly types where $\Delta = 1$ has a much lower delay than $\Delta = 10$. However, we will see below that for certain types of anomalies larger Δ values do pay off.

The FAR values are also similar for the different detection horizons and range from 3% to a maximum of 6.8%. These FAR rates are likely too large for many practical applications as they would result in too many false alarms. Thus, there appears to be significant potential to improve these baseline FAR results while maintaining similar or better delays.

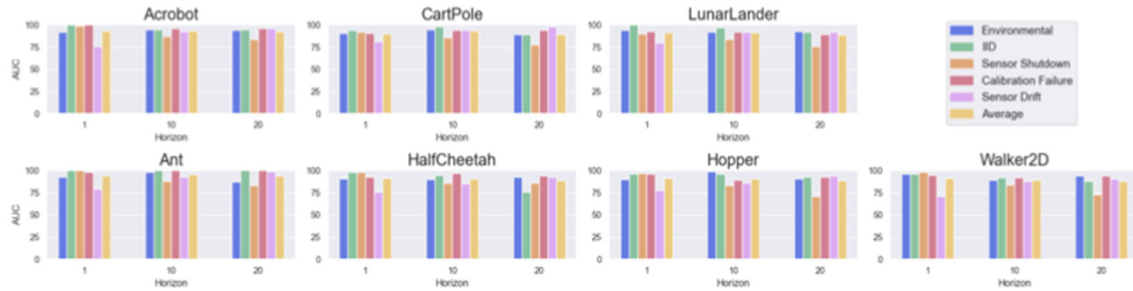


Figure 25 How changing the horizon affects anomaly detection in terms of AUC

Impact of Horizon. Depending on the anomaly, one could choose to perform ODD detection on an arbitrary horizon. Here, we consider the performance of our method for the individual types of anomalies when the horizon is changed. To do so, we have provided AUC results for $\Delta = 1, 10, 20$ in Figure 25Figure 1 the different anomaly types. For the sensor-drift anomalies, which are slowly evolving, we might expect longer horizons to perform better up to a point. The results agree with this expectation and show an advantage for the larger horizons. In contrast, for sensor shutdown, which has a sudden impact, it is expected that shorter horizons will perform better. Figure 25 shows that $\Delta = 1$ has a much better AUC compared to $\Delta = 10, 20$. For other types of anomalies, results among different horizons are quite similar. These results indicate the importance of varying the qualitative types of anomalies in ODD benchmarks to highlight differences in approaches and parameters.

Analysis of additional aspects of the approach can be found in the full paper [DF21], which includes an analysis of the impact of using ensembles and impact of recurrent memory, both of which are determined to be important to performance.

4.4 Results from Learning Unknown System Dynamics

We illustrate the effectiveness of taking advantage of available side information for learning to control and model unknown dynamical systems under scarce amount of data. We provide more details on the experiments in the paper [47].

4.4.1 Results: On-The-Fly control From a Single and Ongoing Trajectory. We empirically demonstrate that DaTaControl, using data from only the current trial and the least amount of side information necessary to learn, can achieve performance comparable to the highly-tuned performant reinforcement learning algorithms Distributed Distributional Deep Deterministic Policy Gradient (DDPG) (D4PG) and Soft Actor Critic (SAC) trained over *ten million* of interactions with the environments. We emphasize that the comparison is unfair to our algorithm since, at each evaluating episode, it learns from only the *thousand* data obtained during the episode, while D4PG and SAC were pre-trained using *ten million* data before the evaluation. Further, we show in the F-16 aircraft simulator, 13-states and 4-control inputs nonlinear dynamics with polynomial control, that the algorithm can meet real-time requirements.

Data-Driven Control in MuJoCo. The equations of motion for multi-joint dynamical systems in the MuJoCo environment are as follows: $M(q)\ddot{q} + b(\dot{q}, q) = h(u) + JcT(q)Fc(q, \dot{q}, u)$, where q is the system's state, $M(q)$ is the inertial matrix, $b(\dot{q}, q)$ contains coriolis, centrifugal, gravitational and passive forces, $JcT(q)$ is the contact Jacobian matrix, and $Fc(q, \dot{q}, u)$ is the contact force.

We seek to achieve high performance in control tasks such as driving to a target a two-link planar reacher (Reacher) and a 3-link planar swimmer (Swimmer) based on highly-accurate Reynolds fluid drag model. Besides, we seek to maximize the speed of a running planar biped (Cheetah). For each environment, the cost function is provided by MuJoCo and we perform numeric differentiation in order to find its gradient.

The Lipschitz bounds are under-estimated using only 1000 data points obtained prior to the on-the-fly control. The Reacher environment does not consider any side information other than the Lipschitz bounds, while Swimmer and Cheetah consider that $M(q)$ is known in order to start learning. Indeed, without such side information, our algorithm fails to learn to control due to large over-approximations of reachable sets. Further, we reduce the over-approximation of Fc by considering the Coulomb law's of friction. That is, we impose the constraints $Fc1 \geq 0$ and $Fc1 \geq \sqrt{(Fc2)^2\mu_1 + (Fc3)^2\mu_2}$ at each contact point, where $Fc1$ is the normal force value, $Fc2$ and $Fc3$ are the tangential forces, and μ_1, μ_2 are the friction coefficients.

In Figure 26 from left to right, we plot the (average) *immediate reward* over 100 episodes. The experiments show that our algorithm can yield performance comparable to D4PG and SAC pre-trained over *ten million* of iterations with the environment and evaluated over the 100 episodes.

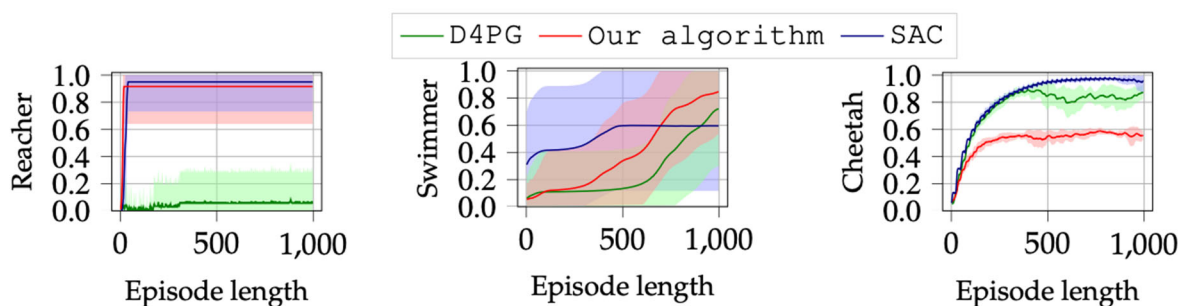


Figure 26 Plot of Immediate Results over 100 Episodes

Figure 26 demonstrates that it is possible to learn to control with only data from a single episode by leveraging side information. In Cheetah, more side information can improve our algorithm's performance. We reduced the time-step value of Reacher to accommodate our algorithm and observed that D4PG was unable to learn the task solely due to such a change, while SAC was not affected.

Data-Driven Control of an F-16 Aircraft. We consider a scenario involving an F-16 aircraft diving towards the ground and show how our algorithm can prevent a ground collision using only the measurements obtained during the dive and elementary laws of physics as side information. We compare our algorithm with the tuned linear-quadratic regulator algorithm of the simulator.

The F-16's flight control system is described by a hierarchical feedback control loop consisting of an autopilot and a low-level controller. Inside the simulator, the underlying dynamics of the F-16 capture the (6-Degrees of Freedom (DOF)) movement of an aircraft through the standard aerodynamic equations.

In Figure 27 our algorithm successfully enables the F-16 to avoid the ground collision while the embedded LQR controller fails to avoid the crash. Further, it can be applied in real time since the compute time is less than the *control time step* enforced by the simulator.

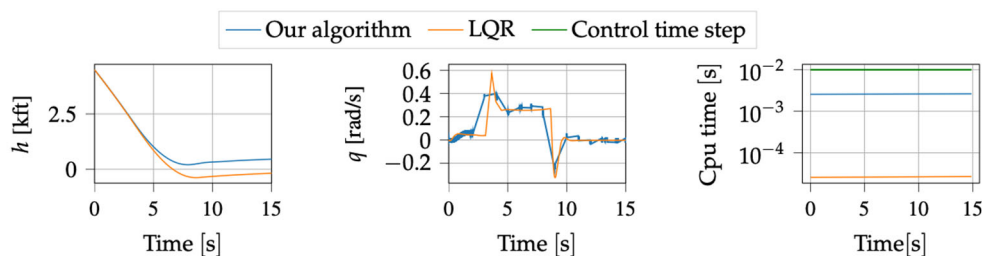


Figure 27 On-The-Fly-Control Results for F16

The dynamics describe the evolution of the system's states, namely velocity v_t , angle of attack α , sideslip β , altitude h , attitude angles: roll ϕ , pitch θ , yaw ψ , and their corresponding rates p , q , r , engine *power* and two more states for translation along north and east. The plant model uses several *linearly interpolated lookup tables* to incorporate wind tunnel data describing the engine model, the various coefficients including force and moment coefficients, and the moments due to the control surfaces.

Our algorithm takes as side information the knowledge of the rigid-body dynamics of a 6-DOF, while assuming that the aerodynamics forces and moments interpolated via lookup tables are *unknown*. Thus, the response of the system to control inputs is unknown and needs to be learned on the fly. The Lipschitz bounds were estimated using 1000 data collected before the scenario.

4.4.2 Results: Physics-Constrained Neural ODE's for Dynamical Systems Modeling. We apply the proposed framework to a suite of controlled rigid multi-body systems simulated in Brax [73]. However, as an initial illustrative example of how the framework might incorporate arbitrary a-priori within another context, we begin by numerically experimenting with the dynamics of a double pendulum.

Learning the Dynamics of a Double Pendulum. The dynamics of the double pendulum are described by Equation 15

$$\ddot{\phi}_1 = (g_1 - \alpha_1 g_2) / (1 - \alpha_1 \alpha_2); \quad \ddot{\phi}_2 = (-\alpha_2 g_1 + g_2) / (1 - \alpha_1 \alpha_2) \quad (15)$$

Equation 15 Dynamics of the double pendulum

Here, ϕ_1 and ϕ_2 specify the angular position of the first and second links of the pendulum, $\alpha_1(\phi_1, \phi_2) \propto \cos(\phi_1 - \phi_2)$, $\alpha_2(\phi_1, \phi_2) \propto \cos(\phi_2 - \phi_1)$, and $g_1(\phi_1, \phi_2, \dot{\phi}_1, \dot{\phi}_2)$, $g_2(\phi_1, \phi_2, \dot{\phi}_1, \dot{\phi}_2)$ are both complicated trigonometric functions of the state variables. By setting $x_1 = \phi_1$, $x_2 = \phi_2$, $x_3 = \dot{\phi}_1$, and $x_4 = \dot{\phi}_2$ we can thus express the dynamics as $\dot{x} = h(x)$, where $h_1(x) = x_3$, $h_2(x) = x_4$, and $h_3(x)$, $h_4(x)$ are both given by (4). We consider the following baselines.

- *Defining a Baseline for Comparison.* As a baseline, we parametrize the vector field h by a single neural network $g\theta_1$. In terms of the framework outlined in the previous sections, we thus have $\Theta = \theta_1$ and $F(x, G\Theta) = g\theta_1(x)$.

Figure 28 Numerical results for the double pendulum. The model is trained using a single trajectory consisting of 3 seconds worth of observation data.

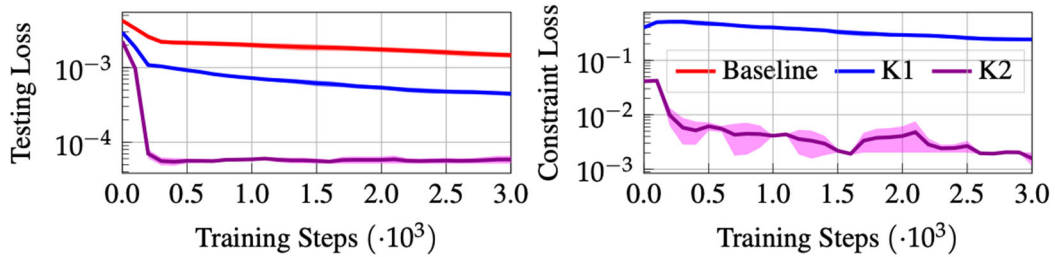


Figure 28 Numerical Results for the Double Pendulum

- *(K1) Incorporating Knowledge of Equation 15.* In order to demonstrate how arbitrary knowledge surrounding the system's vector field can be incorporated into the dynamics model, we assume that Equation 15 is known, along with $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$. However, we assume $g_1(\cdot)$ and $g_2(\cdot)$ are unknown. We parametrize g_1 and g_2 by two neural networks $g\theta_1$ and $g\theta_2$, respectively. The proposed model for the vector field may thus be expressed as $F_1(x, G\Theta) = x_3$, $F_2(x, G\Theta) = x_4$, and with $F_3(x, G\Theta)$, $F_4(x, G\Theta)$ both being functions of α_1 , α_2 , $g\theta_1$, $g\theta_2$, as in Equation 15.
- *(K2) Incorporating Symmetry Constraints on g_1 and g_2 .* To demonstrate that the proposed training algorithm properly enforces constraints on the system's dynamics, we impose equality constraints on $g\theta_1$, $g\theta_2$ derived from symmetries in the vector field. In particular, we impose four separate equality constraints, an

example of which is as follows: $g_1(x_{1:2}, x_{3:4}) = -g_1(-x_{1:2}, x_{3:4})$. These equality constraints are enforced at a collection of points Ω that are sampled uniformly throughout the state space.

Experimental Setup. Each neural network $g\theta_i(x)$ is a multilayer perceptron with Rectified Linear Unit (ReLU) activation functions and two hidden layers with 128 nodes each. We use a rollout horizon of $nr = 5$ when defining the loss function and we randomly sample $|\Omega| = 10000$ points throughout the state space X at which we enforce the above equality constraints.

Experimental Results. Figure 28 plots the results of training the various physics-informed models using only a single trajectory of data, which is equivalent to 3 seconds worth of observations. We observe that in terms of prediction losses, (K1) performs better than the baseline, and (K2) performs better than both the baseline and (K1); its testing loss at convergence is roughly an order of magnitude lower than that of (K1). Furthermore, we observe that the constraint loss of (K2) – which measures the average value of the model’s violation of the equality constraints – is roughly two orders of magnitude lower than that of (K1). These results demonstrate the proposed framework’s effectiveness at incorporating a-priori system knowledge into the dynamics model. As more detailed knowledge is included into the model, the testing loss at convergence is significantly reduced. Furthermore, the gap in constraint loss between (K1) and (K2) demonstrates the effectiveness of the proposed framework in learning to fit the training data while simultaneously enforcing constraints.

4.5 Results from Reinforcement Learning Methods for Learning Programmatic Policies

We demonstrated the effectiveness of PROPEL in synthesizing programmatic controllers in three continuous control environments, including The Open Racing Car Simulator (TORCS), a challenging car-racing environment with multiple tracks. Table 15 and Table 16 show the performance on the considered TORCS tracks (see the paper for more details). Here, PROPELPROG and PROPELTREE are two instantiations of PROPEL, and Neurally Directed Program Search (NDPS) and Verifiability via Iterative Policy ExtRaction (VIPER) are two programmatic RL algorithms from previous work. We see that PROPELPROG and PROPELTREE consistently outperform the NDPS and VIPER baselines, respectively. While DDPG outperforms PROPEL on some tracks, its volatility causes it to be unable to learn in some environments, and hence to crash the majority of the time.

In Table 15 performance results in TORCS over 25 random seeds. Each entry is formatted as Lap-time / Crash-ratio, reporting median lap time in seconds over all the seeds (lower is better) and ratio of seeds that result in crashes (lower is better). A lap time of C_R indicates the agent crashed and could not complete a lap for more than half the seeds.

Table 15 Performance Results in TORCS

LENGTH	G-TRACK 3186M	E-ROAD 3260M	AALBORG 2588M	RUUDSKOGEN 3274M	ALPINE-2 3774M
PRIOR	312.92 / 0.0	322.59 / 0.0	244.19 / 0.0	340.29 / 0.0	402.89 / 0.0
DDPG	78.82 / 0.24	89.71 / 0.28	101.06 / 0.40	CR / 0.68	CR / 0.92
NDPS	108.25 / 0.24	126.80 / 0.28	163.25 / 0.40	CR / 0.68	CR / 0.92
VIPER	83.60 / 0.24	87.53 / 0.28	110.57 / 0.40	CR / 0.68	CR / 0.92
PROPELPROG	93.67 / 0.04	119.17 / 0.04	147.28 / 0.12	124.58 / 0.16	256.59 / 0.16
PROPELTREE	78.33 / 0.04	79.39 / 0.04	109.83 / 0.16	118.80 / 0.24	236.01 / 0.36

Table 16 provides generalization results in TORCS, where rows are training and columns are testing tracks. Each entry is formatted as PROPELProg / DDPG, and the number reported is the median lap time in seconds over all the seeds (lower is better). CR indicates the agent crashed and could not complete a lap for more than half the seeds.

Table 16 Generalization results in TORCS

	G-TRACK	E-ROAD	AALBORG	RUUDSKOGEN	ALPINE-2
G-TRACK	-	124 / CR	CR / CR	CR / CR	CR / CR
E-ROAD	102 / 92	-	CR / CR	CR / CR	CR / CR
AALBORG	201 / 91	228 / CR	-	217 / CR	CR / CR
RUUDSKOGEN	131 / CR	135 / CR	CR / CR	-	CR / CR
ALPINE-2	222 / CR	231 / CR	184 / CR	CR / CR	-

To compare the ability of the controllers to perform on tracks not seen during training, we executed the learned policies on all the other tracks. We observe that DDPG crashes significantly more often than PROPELPROG. This demonstrates the generalizability of the policies returned by PROPEL. Finally, we found that the policies generated by PROPELTREE and PROPELPROG are easier to automatically verify using standard tools than DDPG policies.

Our evaluation for REVEL used 10 benchmarks that include classic control problems, robotics applications, and benchmarks from prior work. We demonstrated that:

- The performance of REVEL is competitive with (or even better than) DDPG, a state-of-the-art deep RL algorithm that we augmented with a reward function that penalizes safety violations, for 7 out of the 10 benchmarks. REVEL achieves significantly better reward than DDPG in one benchmark, and reward is only slightly worse for 2 benchmarks.
- REVEL has better performance than Constrained Policy Optimization (CPO), a state-of-the-art on 4 out of the 10 benchmarks and only performs slightly worse on 2. Furthermore, the cost incurred by CPO is significantly worse on 2 benchmarks.
- REVEL outperforms traditional shielding-based approach on 4 out of 10 benchmarks.

As for safety, DDPG and CPO both exhibit safety violations in 8 out of the 10 benchmarks. In contrast, training in REVEL is guaranteed to be safe.

Figure 29 shows trajectories for “acc”, which models an adaptive cruise control system where the goal is to follow a lead car as closely as possible without crashing into it. The lead car can

apply an acceleration to itself at any time. The x-axis shows the distance to the lead car while the y-axis shows the relative velocities of the two cars. Here, all three trajectories start by accelerating to close the gap to the lead car before slowing down again. The traditional, statically shielded (and most conservative) policy is the first to slow down. The DDPG and CPO policies fail to slow down soon enough or quickly enough and crash into the lead car (the red region on the right side of the figure). In contrast, the REVEL policy can more quickly close the gap to the lead car and slow down later while still avoiding a crash.

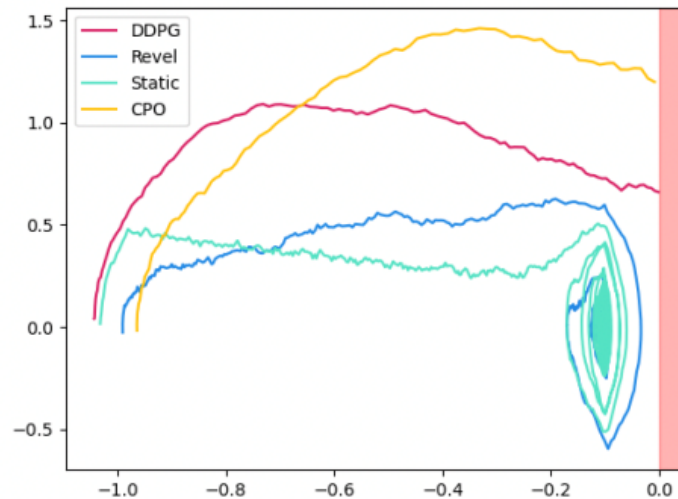


Figure 29 Trajectories for 'acc' Modeling Adaptive Cruise Control

4.6 Transition Results and Open Source Release of Assured Learning

The team has open sourced our tools and methodologies in accordance with program priorities. We provide a summary below of these and other transition efforts completed.

4.6.1 CSAF Open source and Tool Transition to Assured Autonomy Repository

CSAF Github: <https://github.com/GaloisInc/csaf>

4.6.2 COPILOT Transition to NASA

Galois has transitioned the what4 library (M11 Probabilistic Proof Space efforts) into NASA's CoPILOT repository: <https://github.com/copilot-language/copilot>

4.6.3 Assured Autonomy Tool Portal

Galois Assured Learning tools have been posted to DARPA's Assured Autonomy tool portal: <https://aavo-staging.isis.vanderbilt.edu/tools>

4.6.4 DISTAR Summary

DARPA requested Galois to open source all software tools on the Assured Learning project. Table 17 summarizes our approved DISTAR requests.

Table 17 Assured Learning DISTAR Requests and Status

DISTAR Case #	Description	Notes
34620 Approved	CSAF Public Release Approved Distribution A	https://github.com/GaloisInc/csaf
33355 Approved	Project Page: Leveraging Symbolic Representations for Safe and Assured Learning and Blog post	https://galois.com/project/leveraging-symbolic-representations-for-safe-and-assured-learning/

5.0 CONCLUSIONS

5.1 F16 Challenge Problem Conclusions

In Phase 1, we demonstrated that the approaches, techniques and algorithms under development showed promise to provide a new level of assurance for learning enabled systems. These approaches provided new mechanisms to verify learning enabled systems at both the component and system level. Using compositional verification and testing techniques in a modular framework, one can now begin the process of model based, directed learning and testing of these systems to generate verification artifacts that provide a higher level of safety guarantees for the future.

In Phase I and II we integrated our assured learning tools into CSAF and demonstrated results on the F16 benchmark for Ground Collision Avoidance (CP 2.5) and Air Collision Avoidance (CP 3.2). We demonstrated F16 Air collision avoidance in both single agent and multi-agent scenarios.

5.1.1 Assured Learning Algorithms, Tools and Methods Conclusions

CSAF Contributions: CSAF is the integrated toolchain that was developed to prototype algorithms for high assurance LECs and evaluate them on LECs and autonomous systems. For modeling, CSAF provides a component-oriented framework that lets the user model complex systems like F-16 flight control system. CSAF provides seamless connections with state-of-the-art tooling for prototyping LECs and algorithms. Finally, CSAF provides a test architecture to easily plug in falsification and testing methods, describe scenarios, specify goals/safety properties and integrates them with CI/CD and existing unit test frameworks. The algorithms and techniques developed on the program have been included in CSAF and demonstrated on the challenge problem which were also developed in CSAF.

Algorithms. We have developed several algorithms that address the capability gaps in assuring mission critical autonomous systems. These algorithms are applicable at various stages of development including design, verification and validation and at runtime. We succinctly list the algorithms and their highlights below.

Specifications: DistSpectrl was developed to address the need for formally specifying the correctness and performance goals for an ML policy. The DistSpectrl framework provides the tooling to automatically translate these specifications into a loss function that reinforcement learning methods can use to train a policy. DistSpectrl is specialized to work with multi-agent systems and has shown initial success in the multi-agent version of the F-16 ACAS challenge problem.

Training: The three training relevant tools FITRE, PROPEL (and Reinforcement Learning with Verified Exploration (REVEL)) and ART address the challenges in training high assurance autonomous CPS systems. FITRE provides a high performance and efficient training using second order optimization techniques, PROPEL is focused on increasing the interpretability of the learnt policy and ART is concerned with correctness of the learnt policy. Both FITRE and PROPEL have been show to scale on the F-16 challenge problems, however ART needs to be developed further in order to scale it to

complex problems. A future research challenge will also include the combination of these three different research directions to achieve a correct, high performance training solution.

Design Time Testing: Goal directed testing is the key technique applicable here. Using a combination of high-level scenario description and goal, it uses automatic falsification algorithms to find if the goals are satisfied. Each satisfied goal is represented with a test cases encapsulating the initial conditions. In this way, tests are automatically generated and are repeatable. The results have been successfully demonstrated on the challenge problems where multiple failures for both GCAS and ACAS scenarios were automatically found.

Runtime Assurance: The techniques here include runtime monitoring and approaches for designing verified fail-safes and infrastructure to implement them in relevant Air Force Research Laboratory (AFRL) platforms. With RIDAD, we are spearheading the research into anomaly detection of out of distribution dynamics. RIDAD provides a strong baseline approach to detect i.i.d. sensor noise, sensor failures, drift and calibration failures. Once detected, RTA with verified primitives can be used to design assured recovery maneuvers. In extreme cases where such pre-designed recovery policies cannot be used, on-the-fly control can be used. The success of these have been demonstrated on the F-16 challenge problems.

6.0 RECOMMENDATIONS

See Appendix C “Towards Principled Autonomy”, a white paper capturing recommendations for future research.

7.0 REFERENCES

- [1] Tiny imagenet dataset. <https://tiny-imagenet.herokuapp.com/>, October 2019. URL <https://tiny-imagenet.herokuapp.com/>.
- [2] Mohammed Alshiekh, Roderick Bloem, Ruediger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), 2017. URL <http://arxiv.org/abs/1708.08611>.
- [3] Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. Neurosymbolic reinforcement learning with formally verified exploration. *Advances in Neural Information Processing Systems*, 33, 2020.
- [4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL <https://proceedings.neurips.cc/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf>.
- [5] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, 2018. URL <https://openreview.net/forum?id=Sy0GnUxCb>.
- [6] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6065–6073, 2019.
- [7] Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv:1406.2572v1*, 2014.
- [8] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. CARLA: an open urban driving simulator. In 1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, *Proceedings*, pages 1–16, 2017. URL <http://proceedings.mlr.press/v78/dosovitskiy17a.html>.
- [9] Thierry Dumas, Aline Roumy, and Christine Guillemot. Autoencoder based image compression: Can the learning be quantization independent? *arXiv preprint arXiv:1802.0937*, 2018.
- [10] Chih-Hao Fang, Sudhir B Kylasa, Farbod Roosta-Khorasani, Michael W Mahoney, and Ananth Grama. Distributed second-order convex optimization. *arXiv preprint arXiv:1807.07132*, 2018.

- [11] Shromona Ghosh, Felix Berkenkamp, Gireeja Ranade, Shaz Qadeer, and Ashish Kapoor. Ver- ifying Controllers Against Adversarial Examples with Bayesian Optimization. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 7306–7313, 2018. doi: 10.1109/ICRA.2018.8460635. URL <http://arxiv.org/abs/1802.08678>.
- [12] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, 2020. URL <https://openreview.net/forum?id=HJgEMpVFwB>.
- [13] Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. arXiv:1602.01407v2, 2016.
- [14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International Conference on Machine Learning, pages 1861–1870. PMLR, 2018.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852, 2015.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.
- [17] Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. ACM Transactions on Embedded Computing Systems (TECS), 18(5s):1–22, 2019.
- [18] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A composable specification language for reinforcement learning tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d AlcheBuc, E. Fox, and R. Garnett, editors, Advances in Neural Information Process- ing Systems 32, pages 13041–13051. Curran Associates, Inc., 2019. URL <http://papers.nips.cc/paper/9462-a-composable-specification-language-for-reinforcement-learning-tasks.pdf>.
- [19] Sudhir B Kylasa. Cuda non-convex framework for fitre optimizer. <https://github.com/kylasa/NewtonCG>, 10 2019.
- [20] Sudhir B Kylasa, Fred Roosta Khorasani, Michael W. Mahoney, and Ananth Y Grama. Gpu accelerated sub-sampled newton’s method for convex classification problems. In SIAM, editor, Proceedings of the 2019 SIAM International Conference on Data Mining, pages 702–710. SIAM, 2019.
- [21] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klas-Robert Muller. Efficient backprop. Neural Networks: tricks of the trade, 1998.
- [22] Shuo Li and Osbert Bastani. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 7166–7172. IEEE, 2020.

- [23] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.
- [24] Ziuju Lu, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He. Recurrent reinforcement learning: A hybrid approach. arXiv preprint arXiv:1509.03044, 2015.
- [25] James Martens. New insights and perspectives on the natural gradient method. arXiv preprint: arXiv:1412.1193v9, 2017.
- [26] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. arXiv:1503.05671v6, 2016.
- [27] Gregoire Montavon, Genevieve B. Orr, and Klas-Robert Muller. Neural Networks: Tricks of the Trade. Springer, 2nd edition, September 2012.
- [28] Yurii Nesterov. Introductory lectures on convex optimization, volume 87. Springer Science & Business Media, 2004.
- [29] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In In Proceedings of the Sixteenth International Conference on Machine Learning, pages 278–287. Morgan Kaufmann, 1999.
- [30] Jorge Nocedal. Updating quasi-Newton matrices with limited storage. Mathematics of computation, 35(151):773–782, 1980.
- [31] Jorge Nocedal and Stephen Wright. Numerical optimization. Springer Science & Business Media, 2006.
- [32] Barak A. Pearlmutter. Fast exact multiplication by the hessian. Neural Computation, 1993.
- [33] Yongming Rao, Jiwen Lu, and Jie Zhou. Attention-aware deep reinforcement learning for video face recognition. The IEEE International Conference on Computer Vision (ICCV), pages 3931–3940, 2017.
- [34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arxiv 2017. arXiv preprint arXiv:1707.06347, 2017.
- [35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2015.
- [36] Rodrigo Toro Icarte, Torny Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In Proceedings of the 35th International Conference on Machine Learning (ICML), pages 2112–2121, 2018.

- [37] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. arXiv preprint arXiv:2010.03950, 2020.
- [38] Hoang-Dung Tran, Xiaodong Yang, Diego Manzananas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T Johnson. Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. arXiv preprint arXiv:2004.05519, 2020.
- [39] Dehao Wu, Maziar Nekovee, and Yue Wang. Deep learning based autoencoder for interference channel. arXiv preprint arXiv:1902.06841, 2019.
- [40] Peng Xu, Fred Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, pages 1–36. doi: 10.1007/s10107-019-01405-z.
- [41] Peng Xu, Farbod Roosta-Khorasani, and Michael W. Mahoney. Second-Order Optimization for Non-Convex Machine Learning: An Empirical Study. arXiv preprint arXiv:1708.07827, 2017.
- [42] Zhewei Yao, Peng Xu, Farbod Roosta-Khorasani, and Michael W Mahoney. Inexact non-convex Newton-type methods. arXiv preprint arXiv:1802.06925, 2018.
- [43] He Zhu, Zikang Xiong, Stephen Magill, and Suresh Jagannathan. An inductive synthesis frame- work for verifiable reinforcement learning. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 686–701, 2019.
- [44] Xuankang Lin, He Zhu, Roopsha Samanta, Suresh Jagannathan “Art: Abstraction Refinement-Guided Training for Provably Correct Neural Networks”
- [45] Zikang Xiong, Joe Eappen, He Zhu, Suresh Jagannathan, “Robustness to Adversarial Attacks in Learning-Enabled Controllers. Submitted to NeurIPS 2020
- [46] Sudhir B. Kylasa, Chih-Hao Fang, Peng Xu, Fred Roosta, Micahel Mahoney, Ananth Grama. “FITRE: Fisher Informed Trust-region Method for Training Deep Neural Networks”, DASC 2020
- [47] M. Ornik, S. Carr, A. Israel and U. Topcu. “Control-Oriented Learning on the Fly”. To appear in the *Transactions on Automatic Control*, 2020.
- [48] Abhinav Verma, Hoang M. Le, Yisong Yue, Swarat Chaudhuri. “Imitation-Projected Policy Gradient for Programmatic Reinforcement Learning”, NeurIPS 2019.
- [49] Mathew Clark, Kuldeep Rattan, “Framework for Mining Hybrid Automata from a Constrained Machine Learning Architecture”, submitted FUZZ-IEEE 2020
- [50] Djeumou, F., Vinod, A., Goubault, E., Putot, S., and Topcu, U., "On-The-Fly Reachability and Control of Unknown Smooth Systems", in preparation to American Control Conference, 2021.

- [51] Ruff, L., Kauffmann, J., Vandermeulen, R., Montavon, G., Samek, W., Kloft, M. Dietterich, T., Müller, K-R. "A Unifying Review of Deep and Shallow Anomaly Detection", Under review for Proceedings of the IEEE 2021.
- [52] Chou Yi, Aditya Zutshi, Matt Clark, "High Assurance Run-Time Monitoring Architecture for Autonomous Control". To appear in AIAA Scitech 2021 Forum.
- [53] [Chen2019] Chen, Ricky TQ, et al. "Residual flows for invertible generative modeling." *arXiv preprint arXiv:1906.02735* (2019).
- [54] [Friston2010] Friston, Karl. "The free-energy principle: a unified brain theory?." *Nature reviews neuroscience* 11.2 (2010): 127-138.
- [55] [Ha2018] Ha, David, and Jürgen Schmidhuber. "World models." *arXiv preprint arXiv:1803.10122* (2018).
- [56] [Woods2019] Woods, Walt, Jack Chen, and Christof Teuscher. "Adversarial explanations for understanding image classification decisions and improved neural network robustness." *Nature Machine Intelligence* 1.11 (2019): 508-516.
- [57] [Tenenbaum2011] Tenenbaum, Joshua B., et al. "How to grow a mind: Statistics, structure, and abstraction." *science* 331.6022 (2011): 1279-1285.
- [58] [Mengistu2016] Mengistu, Henok, et al. "The evolutionary origins of hierarchy." *PLoS computational biology* 12.6 (2016): e1004829.
- [59] [Sifakis2019] Sifakis, Joseph. "Autonomous systems—an architectural characterization." *Models, Languages, and Tools for Concurrent and Distributed Programming*. Springer, Cham, 2019. 388-410.
- [60] [Nachum2018] Nachum, Ofir, et al. "Data-efficient hierarchical reinforcement learning." *arXiv preprint arXiv:1805.08296* (2018).
- [61] [Lesort2020] Lesort, Timothée, et al. "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges." *Information fusion* 58 (2020): 52-68.
- [62] [Toulmin2003] Toulmin, Stephen E. *The uses of argument*. Cambridge university press, 2003.
- [63] [NRC1999] National Research Council. 1999. *Trust in Cyberspace*. Washington, DC: The National Academies Press. <https://doi.org/10.17226/6161>
- [64] [Perrow1984] Perrow, Charles (1984) "Normal Accidents". Princeton University Press.
- [65] [Dietterich2019] Dietterich, T. G. (2019). Robust Artificial Intelligence and Robust Human Organizations. *Frontiers of Computer Science*, 13(1): 1-3. [SharedIt Link to Publisher's Page DOI 10.1007/s11704-018-8900-4](https://doi.org/10.1007/s11704-018-8900-4).

- [66] [Taleb2012] Taleb, Nassim Nicholas (2012) “Antifragile: Things That Gain from Disorder”. Random House
- [67] [Pearl2011] Pearl, J. and Bareinboim, E. (2011). Transportability of causal and statistical relations: a formal approach. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 247–254. AAAI Press.
- [68] [Subbaswamy2019] Subbaswamy, A., Schulam, P., and Saria, S. (2019). Preventing failures due to dataset shift: Learning predictive models that transport. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3118–3127. PMLR
- [69] [Subbaswamy2020], A. and Saria, S. (2020). I-spec: An end-to-end framework for learning transportable, shift-stable models. *arXiv preprint arXiv:2002.08948*
- [70] [CoSTAR] [How JPL's Team CoSTAR Won the DARPA SubT Challenge: Urban Circuit Systems Track](#)
- [71] Stanley Bak, Sergly Bogomolov, Niklas Kochdumper, Ethan Lew and Kostiantyn Potomkin, Reachability of Koopman Linearized Systems Using Random Fourier Feature Observables and Polynomial Zonotope Refinement, CAV 2022 (submitted).
- [72] Chuchu Fan, Xin Qin, Yuan Xia, Aditya Zutshi, Jyotirmoy Deshmukh, Statistical Verification of Cyber-Physical Systems using Surrogate Models and Conformal Inference, ICCPS 2022, (submitted).
- [73] [Brax] Freeman, C. & Frey, Erik & Raichuk, Anton & Girgin, Sertan & Mordatch, Igor & Bachem, Olivier. (2021). Brax -- A Differentiable Physics Engine for Large Scale Rigid Body Simulation.
- [74] Ferlez, J. & Shoukry, Y. (2021) Assured Neural Network Architectures for Control and Identification of Nonlinear systems.
- [75] P. Heidlauf, A. Collins, M. Bolender, S. Bak. (2018) Verification Challenges in F-16 Ground Collision Avoidance and Other Automated Maneuvers, 5th International Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH 2018).
- [76] <https://github.com/stanleybak/AeroBenchVVPython>

APPENDIX A: KEY ACCOMPLISHMENTS

1. May 2020: FITRE Algorithm to Optimize Training of ML Models

DARPA Assured Autonomy developed a new optimization technique for training ML models that surpasses the state-of-the-art gradient descent techniques. In comparison to the best-existing techniques and their implementations in optimized public packages such as TensorFlow and PyTorch, this technique is up to 300% faster (in training time), and has better empirical generalizability (up to 20% better test accuracy) on standard benchmarks (CIFAR10, CIFAR100, VGG16, VGG19). Assured Autonomy optimized GPU implementations of technique are made available in open source over the public domain. (Purdue: A. Grama)

2. June 2020: Performance bounds for Deep Learning Approaches (Anomaly Detection)

DARPA Assured Autonomy has demonstrated upper performance bounds on common types of deep learning approaches to detecting novel object categories in images. Extensive experiments decomposed the sources of error by measuring the performance impact of using ideal oracles for image representation learning and/or anomaly detection using the representations. The results show that for the best current techniques there are significant drops in AUC when removing the detection oracle (0.94 to 0.85) and then removing the representation oracle (0.85 to 0.77). These methods provide researchers with principled ways to identify how proposed improvements address each performance gap and how much room there is to improve within this framework. (OSU- T. Dietterich)

3. January 2021: Reinforcement Learning Method to Learn Symbolic Policies

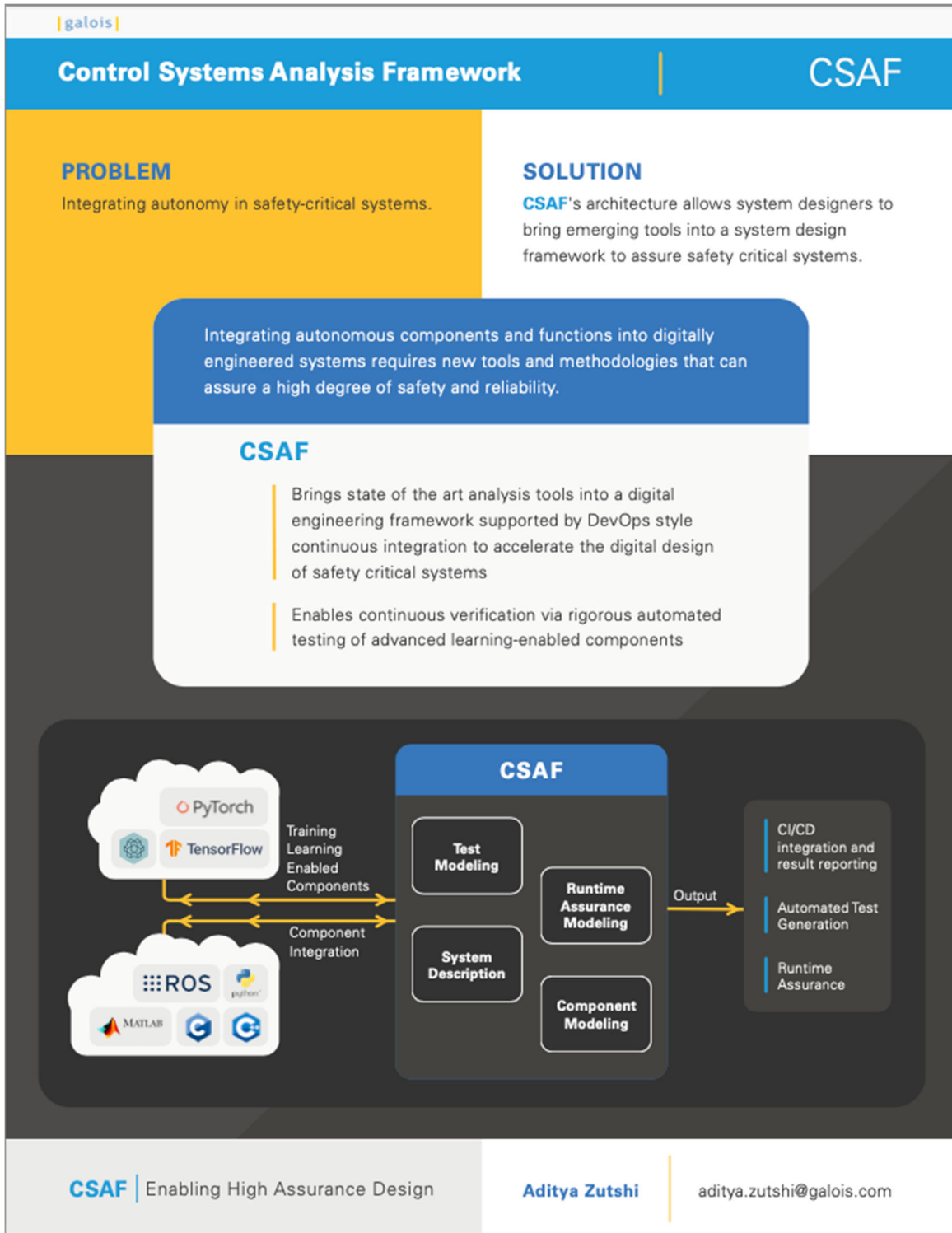
DARPA Assured Autonomy program developed a new reinforcement learning method that learns symbolic policies. As opposed to classical reinforcement learning where learned policies are opaque, symbolic policies are auditable and can be verified for correctness. The method has been evaluated on 12 examples from control and robotics that are analogous to simplified self-driving applications. The empirical results show that this method enforced safe exploration without compromising performance, whereas other state-of-the-art reinforcement learning methods resulted in violation of safety constraints. (UT Austin-S Chaudhuri)

4. February 2021 On-The-Fly-Control Using Data Driven Algorithms

The DARPA Assured Autonomy program developed and demonstrated a data-driven algorithm for the on-the-fly control of systems with unknown nonlinear dynamics. The algorithm merges data from only the current trajectory of the system and, if available, takes advantage of various forms of side information on the underlying dynamics. It achieves real-time and convex-optimization-based control of the system through a data-driven over-approximation of the set of states the system may reach. Our experiments show that the algorithm outperforms, in terms of optimality of the control and computation time, state-of-the-art data-driven approaches based on

system identification with model predictive control, and contextual optimization-based approaches. The algorithm was demonstrated in a 14-dimensional physics model of an F-16 ground collision avoidance simulation. In a scenario involving an F-16 aircraft simulation falling from an extremely low pitch angle and low altitude (-85deg and 3500ft), we show that when considering the dynamics of the F-16 to be unknown, the algorithm enables to prevent its crash using only the measurements obtained while falling and some elementary laws of physics as side information. (UT Austin – U. Topcu)

APPENDIX B: CSAF CAPABILITY SUMMARY

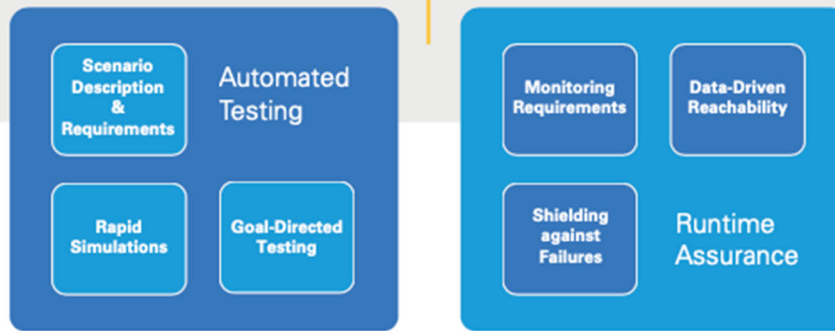


How does CSAF achieve High Assurance Design?

CSAF supports prototyping and integrating state-of-the-art training methods, modeling and specification formalism, and testing and runtime analysis.

CSAF increases component and system assurance using:

- + **Digital engineering support: model-based training, design, and testing.**
- + **Automated testing: Goal-Directed Testing.**
- + **Runtime assurance: designing Runtime Assurance solutions using monitors and formally verified fail-safes.**



CSAF Features

Highlights and Achievements

- + Developed on DARPA's Assured Autonomy program to tackle the challenge of assuring the correctness of neural networks used in safety critical systems.
- + Applied to the F-16's flight control system: Modeled the components of the F-16's flight control system and associated physics.
- + Applied to two different functions: Ground and Air Collision Avoidance (GCAS and ACAS).

APPENDIX C: TOWARDS PRINCIPLED AUTONOMY, A WHITE PAPER

Dr. Aditya Zutshi, Mr. David Burke, Dr. Walt Woods, Dr. John Launchbury, Galois, Inc
Dr. Tom Dietterich, OSU

Introduction: Ideally, a high assurance autonomous system should be able to function acceptably in real-world within some range of operating conditions that are well understood. It should be easy to train and retrain with enough robustness to allow for successful transfer to the real-world and its actions should be easy to debug and explain. It should also have a notion of its own limitations (known-knowns, known-unknowns, and unknown unknowns).

Significantly, as the level of autonomy increases, the assurance objectives shift dramatically from individual components to how well the overall system functions in various contexts. Mission success becomes the focus. But as we push the frontiers of automation, the problem space gets increasingly complex. One reason for this is the gap between the real-world and the state of existing theory and engineering solutions. Often, we find ourselves designing systems as best as we can, hoping that engineering practices will bridge the gap to reality. For autonomous systems, this process is far from satisfactory, and even though we have made large strides in understanding the nature of learned statistical models, we need additional structure to realize the vision of safe autonomous systems.

In this white paper, we outline a selection of research challenges in high assurance autonomous system design and summarize promising directions, together with some potential example scenarios. We dissect the above-mentioned gap into the following subcategories:

- Assurance in nominal operating conditions
 - [Envelope-Based Assurance](#)
- Assurance in off-nominal operating conditions
 - Dealing with [Anomaly and Surprise](#)
 - [Explicitly Steering Failures](#) so systems fail in (human) understandable ways
- We also speculate about [Architecture for Autonomous Systems](#) that can help achieve properties desirable in safety critical scenarios.

Autonomy is often conflated with machine learning and deep learning (DL). However, in this white paper we take the view that the state-of-the-art ML and DL techniques by themselves are not yet sophisticated enough to take the next step in autonomy due to the difficulty in encoding context, reasoning, adaptability, and real-world concepts like causality. Moreover, DL algorithms are still not well understood and their output can only be evaluated empirically. This creates additional challenges to their real world deployability where novelty or off-nominal conditions often result in poor performance or unexpected failures.

Hierarchical and Modular Architecture for Autonomous Systems. In nature, complex systems often exhibit modular and hierarchical organization, generally attributed to the evolutionary processes [Mengistu2016]. The systems are complex but still exhibit robustness and adaptability

[Tenenbaum2011], both desirable properties which we often struggle to engineer. In these systems, the different hierarchical layers are often associated with different levels of abstractions (time scales, granularity of information). Modularity on the other hand, is usually concerned with separation of concern (with respect to objectives and constraints).

In a similar vein, system designers use hierarchical and modular designs to engineer complex systems to reduce the cognitive cost arising from designing and reasoning about their specifications and requirements and maintaining them. Such decompositions are subjective and rely on the designers' collective experience. Even though substantial research has improved post-hoc testing and formal verification of such systems, we lack a mature process for designing and modifying complex systems with the aim of testability, verification, resilience and robustness. Looking towards complex autonomous systems, it is then natural to question the existence of common architectures (or common design aspects) that optimize for properties desirable in high assurance systems.

Sifakis captures some broad architectural ideas and a set of modules/functionalities essential in an autonomous system [Sifakis2019]: *Perception* to interpret inputs, *Reflection* for building and updating world model, *Goal* selection, *Planning* to achieve chosen goals and *Self-adaptation* to modify behavior through continuous learning and reasoning. Furthermore, Sifakis discusses the trustworthiness of autonomous systems, an overarching aspect in which he includes both nominal and off-nominal performance of a system. In light of this, it is important to explore a design process that co-optimizes for functionality and correctness in nominal conditions, along with continuous learning and adaptability to support acceptable performance (or degradation) in off nominal conditions. To support this, we now outline five disparate promising research directions.

Co-design and mixed reasoning. Existing architectures for autonomy are rigid and designed to realize specific functions. Architectures (for co-design) that facilitate systematic refinement of world-view by exchanging information between modules and across time can lead to more *intelligent* autonomous behaviors. A good example of such a co-design is that of simultaneous localization and mapping (SLAM).

Hybrid architectures that combine different forms of reasoning (including numerical, probabilistic, and rule-based reasoning) can yield the best of current advances in ML and the older symbolic Artificial Intelligence (AI) techniques. This can enable high assurance design at a coarse granularity by selectively using symbolic techniques for the critical aspects of an autonomous system, but also by advancing the state of the art in explicitly characterizing and propagating uncertainties through the modules and layers of the system. Team CoSTAR in the DARPA SubT challenge [CoSTAR] scratched the surface of the topic to satisfy the challenging requirements of the program: robust autonomous operation in extreme uncertainty and degraded conditions.

Modular architecture for Deep Learning. Architecture itself encodes implicit information and imposes constraints, it is important to explore these in a systematic fashion. For example, multiple architectures for artificial neural networks exist for different applications (Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNNs), CNNs, Transformers). This brings up the question if there are certain architectural choices that are more amenable to the design of critical systems? For instance, systems designed using hierarchical reinforcement learning [Nachum2018]

are modular, and hence more amenable to analysis when compared to monolithic end-to-end learning designs.

Architectures that enable continuous learning can provide robustness and adaptability required when realizing autonomy in the real world, especially in the domain of robotics [Lesort2020]. This is in stark contrast to the existing rigid designs for autonomous systems. Reasoning about correctness of such evolving designs would be even harder, but this might be offset by assurance on their constant improvements and convergence to optimality.

Layered assurance and argument models. Modular design makes it easier to reason about individual modules, but it does not by itself help in reasoning about the overall system behaviors. This is due to the fact that large gaps can exist between requirements and specifications; and top-level system properties often do not decompose over individual modules. This is especially true for systems where the modules are tightly coupled and share dense connections resulting in a rich set of emergent behaviors due to interaction between the modules. In such cases assurance arguments, such as Toulmin structures [Toulmin2003] can fill the gaps. Note that it is a mistake to think of assurance as a system property. While we can create assurance artifacts as evidence for claims we are making, the assurance itself is not an object – it is a belief held by system stakeholders, i.e., assurance is something that an organization believes, not something that an organization has.

Failure engineering — Explicitly Steering Failures. A 1999 National Academies of Science report entitled “Trust in Cyberspace” [NRC1999] offered up “Three Axioms of Insecurity”.

1. Insecurity exists
2. Insecurity cannot be destroyed
3. Insecurity can be moved around

This recognition allows people to think more about engineering Trust itself, rather than trying to eliminate all points of Trust in a system (note that Trust is a security liability, not a security asset).

Now let’s apply these ideas to failures in learning systems. In the book “Normal Accidents” [Perrow1984], Charles Perrow argued that it is myopic to look at failures as simply instances of machines malfunctioning. Instead, failures are the inevitable result of complex systems that are tightly coupled (hence the title). Perrow opines that modern complex systems should be modeled as socio-technical systems, because much of the time, it is organizations that are the primary causes of failures. As he wrote “Our ability to organize does not match the inherent hazards of our organized activities”.

Therefore, failure is not so much a malfunction as it is the result of the inability of a system to cope with complexity. A dynamic, not well-understood environment acting upon a tightly coupled system almost inevitably outstrips the abilities of the system to adapt (the human element in particular) [Dietterich2019].

To express these insights, we could take the Insecurity axiomatic principles and recast them in terms of failures. This gives us three corresponding axioms of Failure Engineering:

1. Failures happen
2. Failure-proof systems cannot be designed
3. Failures can (and should) be managed

We envision Failure Engineering (axiom 3) as the art/craft/science of designing a system so that failures are both explicit and managed. A useful way to look at this is the “Swiss Cheese” metaphor of risk management, which is similar to a defense-in-depth strategy. The idea is that systems can be thought of being like a stack of cheese slices – each single slice may have a hole in it (representing a potential or existing failure) – but if they are stacked together in the right way, no hole extends completely across the whole stack.

The metaphorical challenge of stacking cheese slices translates into the challenge of selecting components and interfaces that work well together. And the important thing to observe is that this is a contextual decision – whether a hole in a particular slice is “dangerous” depends on which other slices are included in the stack. This translates into knowing the relationships and interactions between different components – metaphorically, this means designing to minimize “common holes”.

From our Axioms of Failure Engineering above, we know that failures are going to occur (i.e., we should not expect to have slices without any holes), and so the objective is to make the operational context explicit enough (i.e., which slices need to be put in the stack), and designing an arrangement so that if a hole is unavoidable, we retain the control as to where it shows up.

Trying to comprehensively define the performance envelope for the system (i.e., all possible environments it could find itself in) is not generally feasible, either. In safety engineering, a distinction is made between:

1. Regular threats – these are the situations that come up often enough that a well-engineered system is designed to address them.
2. Irregular threats – these are the (extreme) low probability events that are inside the current model space, but may or may not be addressed in the current system design.
3. Unforeseen threats – these represent events/situations that are truly “off the map” because the current conceptual/model space doesn’t include them.

The real challenge is addressing category (3). We offer some potential research avenues towards that goal:

- *Targeted Formal Methods*: Formal methods are necessary, but not sufficient. As Donald Knuth once wrote to a correspondent: “Beware of bugs in the above code: I have only proved it correct, not tried it.” One way to think about the role of formal methods is by analogy to the famous Eisenhower quote about planning: “Plans are useless. But planning is indispensable.” In the context of formal methods, we have: “Formal proofs are useless. But applying formal methods is indispensable.” That’s because the process of working through formal methods forces us to surface hidden assumptions and bring clarity

and precision to our claims and arguments. The research challenge here is gaining a better understanding of how to use targeted formal methods in the design of complex systems.

- *Steering Failures*: The “normal accidents” paradigm tells us that tightly coupled subsystems are the primary triggers for catastrophic failure. By paying more attention to interfaces through explicit models of them, we create possibilities for designing loose coupling in a system, giving the system (human operators, in particular) more time to respond to unforeseen circumstances. This also opens up the potential of using interfaces to “steer” failures into subsystems that are more manageable (which speaks to the “Axioms for Failure” above).
- *Design for Antifragility*: Robust systems are better than fragile systems. Taleb, in [Taleb2012], introduces a third class of systems which he coins “antifragile”. A robust system is resilient when stressors are applied, but an antifragile system takes robustness one step further, and becomes stronger and better under stress. A desirable aspect of antifragility is that the system responds positively to stressors, even when it doesn’t understand them.

Envelope-Based Assurance. For assurance, we need machine learning approaches that do not rely on the assumption that the runtime distribution (of states, of sensor inputs) is the same as the training distribution. This will require the machine learning field to move beyond purely statistical learning. Here are two potential directions that might allow this.

The first direction is causal modeling. Building on the work by Pearl and Bareinboim [Pearl2011], Subbaswami, Saria, and their collaborators [Subbaswamy2019, Subbaswamy2020] have developed a causal modeling approach to “transportability” in which the model designer defines a parameterized causal model and annotates it to indicate those aspects that are invariant and those that could change at run time. By analyzing these models, Subbaswami et al. derive a statistical model that is guaranteed to generalize correctly over the aspects marked as potentially varying. Provided that the qualitative structure of the causal model is correct, this gives assurances. However, in many cases, the qualitative structure *cannot* be verified, because it concerns assertions about the absence of unobserved confounding variables. Such assertions cannot be tested without measuring additional quantities, so we cannot rely on runtime monitors to check the qualitative structure. As a design discipline, this may lead us to add sensing of those variables so that they can be monitored.

The second direction is to combine smoothness assertions with dense sampling within an envelope of the state space. Smoothness assertions would specify that the fitted function (e.g., for sensor interpretation or system dynamics) has a small Lipschitz constant, bounded derivatives, bounded Fourier coefficients, and so on. From this, a sufficient sampling density can be determined that will guarantee the accuracy of the fitted model throughout the state space.

An alternative to dense sampling might be adversarial (robust) optimization in which an adversary attempts to identify under-sampled regions of the state space where the fitted model is bad. If the adversary can be guaranteed to find such points, then those can be sampled, and the fitted model can be improved until the adversary cannot find any more failure regions. This is related to model checking, and perhaps SMT solvers can be used.

This second approach must confront the curse of dimensionality. This might be achieved by providing an expressive language for describing the envelope in the hopes that we can keep it as small as possible. If we could bound the degree of interaction among state variables, we might be able to employ fractional factorial sampling designs instead of sampling all possible combinations of the values of state variables. This suggests combining explicit causal models with smoothness assumptions.

Anomaly and Surprise. Even though well-engineered systems operate optimally in normal parameters, real-world circumstances often push them outside of their operating region. A first step toward principled autonomy is recognizing extraordinary observations as a means of understanding when the system might be operating sub-optimally. Even better, when the system is capable of computing possible future trajectories (forward modeling), these sub-optimal regions might be avoided or mitigated.

Broadly, anomaly detection and *surprisal* both rely on some distance metric between data available at training time and data experienced during operation. We note that even in online systems which might use a model of their world to compute this distance metric, real-world sensory noise forces such models to require constant maintenance. This relates to Friston's body of work, [Friston2010], wherein it is hypothesized that brains are constantly trying to evaluate their environment by making predictions and observations, and minimizing the *surprisal* between the prediction and the observation. Therefore, the accuracy of any anomaly detection system relies on this distance metric being defined across task-relevant metrics in a way that generalizes and can be updated with some robustness.

For example, a simple positioning system attempting to answer the question, "Have I been here before?" would be able to use Euclidean distance metrics between the current location and training locations. However, perhaps the global positioning system (GPS) receiver is degrading over time, leading eventually to measurements which are useless in isolation. To compensate for this sensory drift, a principled automaton might substitute knowledge of its own locomotion to correct the small-scale location updates which can no longer be retrieved from individual GPS measurements.

The problem of anomaly detection gets more challenging when such a distance function is not directly expressible. For instance, in ML, it is common that a system might want to see if it recognized its location from camera data alone. That is, the "location" might be an abstract idea rather than a set of Euclidean coordinates. In these cases, a system demonstrating principled autonomy has two issues:

First, camera data is unreliable given e.g. changes in lighting conditions. Therefore, a robust system would likely need to either have or learn a model of its environment, perhaps using a system like [Ha2018] to reduce the observed camera data down to a denser, more salient latent space. This presumes availability of sufficiently diverse data, which might be difficult to assess. Thus, the value of the latent space in isolation is difficult to compute. We note that this system might technically be skipped, relying wholly on a resolution to the second issue.

For the second issue, any direct distance function across such a latent space would likely be mismatched to the latent space's dynamics. One could attempt to condition the latent space such that a standard distance function would work using something like a variational autoencoder

(VAE), though research on that is limited. ML methods have notoriously crumpled manifolds; these may be smoothed via techniques like [Woods2019], however, this often reduces the effective resolution of the latent space, and thus the accuracy. Another option would be to map the latent space to a density estimate explicitly, using something like residual flows [Chen2019]. A benefit of approaches like residual flows is that they force the ML method to intrinsically learn a compact representation amenable to distance computation.

Ideally, a machine with principled autonomy would further be able to identify new modalities within the data and expand its ability to deal with deploy-time changes while keeping its overall objectives in scope. This was one of the ideas behind DARPA's Lifelong Learning Machines (L2M) program.

Example Challenge Problems. We outline four possible example scenarios: lightweight medical procedures, autonomous driving, network cyber protection, and advisory AI for decision making. Each of the examples demonstrate (a) the need for assurances, and (b) the value of combining rich model structures with learning.

Lightweight Medical Procedures: Healthcare has tremendous opportunity for automation, but the consequences of failures are also high. Hence, use cases in the domain can provide appropriate challenges for assurance in autonomous CPS. We consider lightweight medical procedures that are routine, low complexity and have low risk of extreme failure. The considered examples can provide a range of autonomy (similar to the levels of vehicle autonomy) and allow for adjusting the challenge. These examples can include Coronavirus disease (COVID-19) testing, intramuscular and subcutaneous vaccination administration and blood draws. A Medical kiosk (medical version of an Automated Teller Machine (ATM)) can be envisioned with different levels of autonomy.

- Level 1 – Assisted automation, where the medical staff helps in positioning and validating all parameters and operating conditions.
- Level 2 – Advisory and partial automation.
- Level 3 – Multiple kiosks operated by a medical personal with minimal oversight. Limited intervention required.
- Level 4 – Fully autonomous.

Possible failure modes include

- Mis-identifying the application site: detection of site, for example detecting and classifying an arm and its orientation.
- Not detecting novel equipment failures like broken needles, obstructed tubing.
- Failing to recognize when something is going wrong: Is the swab inserted too far into a nasal cavity? Is the blood failing to clot?

Broadly, failure modes can be categorized either as well understood and included as specific case studies in medical training, or, as 'obvious' and never specified formally. To prevent failure, successful automation in this space will require to deal with both and might use the following mechanisms:

- Interactive verbal communication with the patient on a semantic level.
- Logical modeling of failure modes and their appropriate integration, with appropriate mechanisms for escalation in an emergency or when human intervention is required.
- A Physical or a verbal failsafe for human interventions to prevent or recover the system from unexpected failure modes.
- Explainability will be desirable in gaining confidence and debugging.
- Continuous learning for improving with every experience and failure encountered.

This application may be appropriate when using probabilistic programming models. For instance, a generative model can be used to match the visual and other sensory processing (finding the nasal cavity, detecting depth and pressure on the swab).

Autonomous Driving: The application for autonomy to driving provides us with a rich set of scenarios where contextual reasoning is required and often essential to prevent catastrophic failures. The challenges come from

- Physical malfunctions or non-nominal performance: sensors, tires, breaks, engine.
- Failures and errors in algorithms such as perception.
- Unforeseen circumstances.

To solve these, contextual reasoning and awareness is required as different problems require a different response in different driving conditions. For example, using different models for driving under different traction conditions. When a skid occurs, the appropriate models kick in to identify the cause (water on the road, ice, high wind, tire failure, etc), which then introduce alternative driving models to manage the vehicle (bring it safely to the shoulder, or regain control to return to normal conditions).

Certain failures or abnormal conditions are often transient, for example a small patch of ice under the shade of a bridge on an otherwise perfectly sunny day, requiring a different response. This might involve a temporary switch to more cautious driving mode but might also require a proactive response.

Network Cyber Protection: Networks are continuously evolving environments whose gatekeepers require adaptability, continuous learning and fusion of knowledge from different sources.

A network monitor or gatekeeper needs to balance false positives and negatives. It seems impossible for a gatekeeper to perfectly learn every anomalous behavior and characterize as to whether it is likely an attack, or component within an advanced persistent threat (APT), or whether it is just a new behavior of the users of the network.

Quantifying the risks of failure is important here. On the one hand, it is valuable to be sure that certain kinds of attacks will be noticed and prevented. At the same time, it is equally important to quantify the likely interruptions to service when legitimate activity occurs, especially if an attacker could learn how to game the system and engineer a Denial of Service (DOS) attack.

There has been significant work already in trying to learn legitimate usage patterns of people (in the context of usage models). These tend to have a hard time when a new behavior emerges collectively: is the new pattern because of a coordinated attack, or was there an out-of-band coordination event, such as an email from a supervisor instructing a particular new activity? One possible source of innovation could be to learn patterns across organizations (perhaps that meet a similarity measure).

Advisory AI for Decision Making: Real-world decision-making can get extremely complex. AI systems that maintain objectivity while speeding up the computation would have obvious benefits. Such systems are likely to be domain specific but they share common needs, requiring breakthroughs in learning and representing constraints and objectives, and in methods for maintaining transparency and building trust.

They also need general methods for dealing with the noisiness of the real world. Aircraft collision avoidance decision systems like Traffic Alert and Collision Avoidance System (TCAS) and ACAS-X demonstrate how complex problem-spaces can become. Although these advisory systems are designed to operate only in very specific conditions, they are still highly complex and took years to design and build trust. This is in part due to the gap between the complexity of real-world decision making processes and our existing engineering methods.

When a problem can be represented mathematically, solutions appear to be easy to design. Unfortunately, the real world is rarely so clean. In theory there is no difference between theory and practice, but in practice there is. So with collision systems. In most scenarios, avoiding a collision involves a simple negotiation across two agents. But in practice, we have to deal with partial observability, noisy data, unknown or hard to represent objectives and ‘rules of engagement’.

References:

[Chen2019] Chen, Ricky TQ, et al. "Residual flows for invertible generative modeling." *arXiv preprint arXiv:1906.02735* (2019).

[Friston2010] Friston, Karl. "The free-energy principle: a unified brain theory?." *Nature reviews neuroscience* 11.2 (2010): 127-138.

[Ha2018] Ha, David, and Jürgen Schmidhuber. "World models." *arXiv preprint arXiv:1803.10122* (2018).

[Woods2019] Woods, Walt, Jack Chen, and Christof Teuscher. "Adversarial explanations for understanding image classification decisions and improved neural network robustness." *Nature Machine Intelligence* 1.11 (2019): 508-516.

[Tenenbaum2011] Tenenbaum, Joshua B., et al. "How to grow a mind: Statistics, structure, and abstraction." *science* 331.6022 (2011): 1279-1285.

[Mengistu2016] Mengistu, Henok, et al. "The evolutionary origins of hierarchy." *PLoS computational biology* 12.6 (2016): e1004829.

- [Sifakis2019] Sifakis, Joseph. "Autonomous systems—an architectural characterization." *Models, Languages, and Tools for Concurrent and Distributed Programming*. Springer, Cham, 2019. 388-410.
- [Nachum2018] Nachum, Ofir, et al. "Data-efficient hierarchical reinforcement learning." *arXiv preprint arXiv:1805.08296* (2018).
- [Lesort2020] Lesort, Timothée, et al. "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges." *Information fusion* 58 (2020): 52-68.
- [Toulmin2003] Toulmin, Stephen E. *The uses of argument*. Cambridge university press, 2003.
- [NRC1999] National Research Council. 1999. *Trust in Cyberspace*. Washington, DC: The National Academies Press. <https://doi.org/10.17226/6161>
- [Perrow1984] Perrow, Charles (1984) "Normal Accidents". Princeton University Press.
- [Dietterich2019] Dietterich, T. G. (2019). Robust Artificial Intelligence and Robust Human Organizations. *Frontiers of Computer Science*, 13(1): 1-3. SharedIt Link to Publisher's Page DOI 10.1007/s11704-018-8900-4.
- [Taleb2012] Taleb, Nassim Nicholas (2012) "Antifragile: Things That Gain from Disorder". Random House
- [Pearl2011] Pearl, J. and Bareinboim, E. (2011). Transportability of causal and statistical relations: a formal approach. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 247–254. AAAI Press.
- [Subbaswamy2019] Subbaswamy, A., Schulam, P., and Saria, S. (2019). Preventing failures due to dataset shift: Learning predictive models that transport. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3118–3127. PMLR
- [Subbaswamy2020], A. and Saria, S. (2020). I-spec: An end-to-end framework for learning transportable, shift-stable models. *arXiv preprint arXiv:2002.08948*
- [CoSTAR] How JPL's Team CoSTAR Won the DARPA SubT Challenge: Urban Circuit Systems Track

BIBLIOGRAPHY

Heidlauf, P., Collins, A., Bolender, M., & Bak, S. (2018, September). Verification Challenges in F-16 Ground Collision Avoidance and Other Automated Maneuvers. In ARCH@ ADHS (pp. 208-217).

E. Kaiser, J. N. Kutz, and S. L. Brunton. "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit". *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

<i>2D</i>	<i>Two-Dimensional</i>
<i>3D</i>	<i>Three-Dimensional</i>
<i>AA</i>	<i>Assured Autonomy</i>
<i>AC</i>	<i>Actor-Critic</i>
<i>ACA</i>	<i>Air Collision Avoidance</i>
<i>ACAS</i>	<i>Airborne Collision Avoidance System</i>
<i>ACKTR</i>	<i>Actor Critic with Kronecker-factored Trust Region</i>
<i>AD</i>	<i>Anomaly Detection</i>
<i>AFRL</i>	<i>Air Force Research Laboratory</i>
<i>AI</i>	<i>Artificial Intelligence</i>
<i>APT</i>	<i>Advanced Persistent Threat</i>
<i>ART</i>	<i>Abstraction Refinement- Guided Training</i>
<i>ATM</i>	<i>Automated Teller Machine</i>
<i>AUC</i>	<i>Area Under the Curve</i>
<i>BFGS</i>	<i>Broyden Fletcher Goldfarb Shanno</i>
<i>BN</i>	<i>Batch Normalization</i>
<i>BO</i>	<i>Bayesian Optimization</i>
<i>CARLA</i>	<i>Car Learning to Act, Open source simulator for autonomous driving research</i>
<i>CDRL</i>	<i>Contract Data Requirements List</i>
<i>CI/CD</i>	<i>Continuous Integration/Continuous Deployment</i>
<i>CIFAR</i>	<i>Canadian Institute for Advanced Research</i>
<i>CNN</i>	<i>Convolutional Neural Network</i>
<i>CONGOL</i>	<i>CONstrained Global Optimization of Lipschitz smooth functions</i>
<i>COTS</i>	<i>Commercial Off-the-Shelf</i>
<i>COVID-19</i>	<i>Coronavirus disease</i>
<i>CP</i>	<i>Challenge Problem</i>
<i>CPO</i>	<i>Constrained Policy Optimization</i>
<i>CPS</i>	<i>Cyber Physical Systems</i>
<i>CPU</i>	<i>Central Processing Unit</i>
<i>CSAF</i>	<i>Controls Systems Analysis Framework</i>
<i>CUDA</i>	<i>Compute Unified Device Architecture</i>

<i>CUSUM</i>	<i>Cumulative Sum</i>
<i>D4PG</i>	<i>Distributed Distributional DDPG</i>
<i>DARPA</i>	<i>Defense Advanced Research Projects Agency</i>
<i>DDPG</i>	<i>Deep Deterministic Policy Gradient</i>
<i>DISTAR</i>	<i>Distribution Statement A Request</i>
<i>DL</i>	<i>Deep Learning</i>
<i>DRL</i>	<i>Deep Reinforcement Learning</i>
<i>DoD</i>	<i>Department of Defense</i>
<i>DOF</i>	<i>Degrees of Freedom</i>
<i>DOS</i>	<i>Denial of Service</i>
<i>DQN</i>	<i>Deep Q-Network</i>
<i>DTED</i>	<i>Digital Terrain Elevation Data</i>
<i>FAR</i>	<i>False Alarm Rate</i>
<i>FITRE</i>	<i>Fisher Informed Trust REgion method</i>
<i>GB</i>	<i>Gigabyte</i>
<i>GCAS</i>	<i>Ground Collision Avoidance System</i>
<i>GEMM</i>	<i>General Matrix-to-Matrix Multiply</i>
<i>GPS</i>	<i>Global Positioning System</i>
<i>GPU</i>	<i>Graphics Processing Unit</i>
<i>GRU</i>	<i>Gated Recurrent Unit</i>
<i>IF</i>	<i>Isolation Forest</i>
<i>IID</i>	<i>Independent and Identically Distributed</i>
<i>IPPG</i>	<i>Imitation-Projected Policy Gradient</i>
<i>IQN</i>	<i>Implicit Quantile Networks</i>
<i>KFAC</i>	<i>Kronecker Factored Approximate Curvature</i>
<i>KL-divergence</i>	<i>Kullback-Leibler Divergence</i>
<i>L-BFGS</i>	<i>Limited-memory Broyden Fletcher Goldfarb Shanno</i>
<i>L2M</i>	<i>Lifelong Learning Machines</i>
<i>LE</i>	<i>Learning Enabled</i>
<i>LEC</i>	<i>Learning Enabled Controller</i>
<i>LE-CPS</i>	<i>Learning Enabled Cyber Physical System</i>
<i>LOF</i>	<i>Local Outlier Factor</i>
<i>LQR</i>	<i>Linear Quadratic Regulator</i>

<i>LSTM</i>	<i>Long Short-Term Memory</i>
<i>MARL</i>	<i>Multi-Agent Reinforcement Learning</i>
<i>ML</i>	<i>Machine Learning</i>
<i>MLP</i>	<i>Multilayer Perceptron</i>
<i>MNIST</i>	<i>Modified National Institute of Standards and Technology</i>
<i>MuJoCo</i>	<i>Multi-Joint Dynamics with Contact</i>
<i>NASA</i>	<i>The National Aeronautics and Space Administration</i>
<i>NDPS</i>	<i>Neurally Directed Program Search</i>
<i>NLL</i>	<i>Negative Log-Likelihood</i>
<i>NN</i>	<i>Neural Network</i>
<i>ODE</i>	<i>Ordinary Differential Equation</i>
<i>OODD</i>	<i>Out Of Distribution Dynamics</i>
<i>OSU</i>	<i>Oregon State University</i>
<i>PI</i>	<i>Principal Investigator</i>
<i>PPO</i>	<i>Proximal Policy Optimization</i>
<i>PPS</i>	<i>Probabilistic Proof Spaces</i>
<i>PROPEL</i>	<i>Imitation-Projected Programmatic Reinforcement Learning</i>
<i>RAM</i>	<i>Random Access Memory</i>
<i>REVEL</i>	<i>Reinforcement Learning with Verified Exploration</i>
<i>ReLU</i>	<i>Rectified Linear Unit</i>
<i>ResNet</i>	<i>Residual Neural Network</i>
<i>RIDAD</i>	<i>Recurrent network Implicit Distributions for Anomaly Detection</i>
<i>RIQNs</i>	<i>Recurrent Implicit Quantile Networks</i>
<i>RL</i>	<i>Reinforcement Learning</i>
<i>RNN</i>	<i>Recurrent Neural Network</i>
<i>ROC</i>	<i>Receiver Operating Characteristic</i>
<i>RQ</i>	<i>Research Questions</i>
<i>RTA</i>	<i>Runtime Assurance</i>
<i>SAC</i>	<i>Soft Actor Critic</i>
<i>SGD</i>	<i>Stochastic Gradient Descent</i>
<i>SLAM</i>	<i>Simultaneous Localization and Mapping</i>
<i>SMT</i>	<i>Satisfiability Modulo Theories</i>
<i>TCAS</i>	<i>Traffic Alert and Collision Avoidance System</i>

<i>TD3</i>	<i>Twin Delayed DDPG</i>
<i>TORCS</i>	<i>The Open Racing Car Simulator</i>
<i>TRPO</i>	<i>Trust Region Policy Optimization</i>
<i>UAV</i>	<i>Unmanned Aerial Vehicle</i>
<i>UT</i>	<i>University of Texas, Austin</i>
<i>VAE</i>	<i>Variational Autoencoder</i>
<i>VGG</i>	<i>Visual Geometry Group</i>
<i>VIPER</i>	<i>Verifiability via Iterative Policy ExtRaction</i>