Know the Unknowns: Addressing Disturbances and Uncertainties in Autonomous Systems

Invited Paper

Qi Zhu[1], Wenchao Li[2], Hyoseung Kim[3], Yecheng Xiang[3], Kacper Wardega[2], Zhilu Wang[1], Yixuan Wang[1], Hengyi Liang[1], Chao Huang[1], Jiameng Fan[2], Hyunjong Choi[3]

[1] Northwestern University [2] Boston University [3] University of California, Riverside

ABSTRACT

Future autonomous systems will employ complex sensing, computation, and communication components for their perception, planning, control, and coordination, and could operate in highly dynamic and uncertain environment with safety and security assurance. To realize this vision, we have to better understand and address the challenges from the "unknowns" – the unexpected disturbances from component faults, environmental interference, and malicious attacks, as well as the inherent uncertainties in system inputs, model inaccuracies, and machine learning techniques (particularly those based on neural networks). In this work, we will discuss these challenges, propose our approaches in addressing them, and present some of the initial results. In particular, we will introduce a cross-layer framework for modeling and mitigating execution uncertainties (e.g., timing violations, soft errors) with weakly-hard paradigm, quantitative and formal methods for ensuring safe and time-predictable application of neural networks in both perception and decision making, and safety-assured adaptation strategies in dynamic environment.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**.

KEYWORDS

Autonomous systems, uncertainty, disturbance, weakly-hard, neural networks, safety verification, adaptation

1 INTRODUCTION

At the heart of future autonomous systems is their ability to perceive the environment, reason about the situation, and make decisions accordingly. This is a highly challenging process, given the many *unknowns* involved – from unexpected disturbances (e.g., environmental interference, adversarial attacks, component faults), to inherent uncertainties (e.g., uncertain environment input, randomness in software and hardware operations), and to the lack of understanding (e.g., inaccuracies in physical process models, unpredictability of neural network output). To ensure safe, secure, robust, and efficient operation of these autonomous systems, it is critical to *know the unknowns*, i.e., to model the various uncertainties mathematically, analyze their impact on system properties, and design mitigation strategies that can either directly accommodate the uncertainties or adapt the system configuration to address them, such as lead the system to a fail-operational or fail-safe mode.

Consider the example of a self-driving vehicle. During its operation, there is often constant interference (noise) on its sensing of the environment and communication with other vehicles and infrastructures. There could be security attacks from a variety of interfaces [9, 46], and faults of cyber and physical components that are either transient or permanent. There is significant uncertainty in sensing input, with the vehicle operating in a highly dynamic environment under different road, weather, and traffic conditions. The wide adoption of deep neural networks (DNNs) in perception and to some extent in prediction, navigation, planning and control, together with the lack of full understanding of those DNNs' behavior, further increases the uncertainty in vehicle's response. There are also broad usage of model-based methods for control and decision making in general, where models of the physical environment and mechanical components are built for analysis and optimization. However, given the complexity of most physical processes, creating physical models that are both accurate and sufficiently efficient for runtime control remains a challenging task.

Given this broad range of unknown factors and the safety-critical and mission-critical nature of many autonomous systems, it is clear that new *uncertainty-aware* methodologies, algorithms, and tools are needed in the design and operation of autonomous systems. In this paper, we will discuss in-depth the challenges from some of those uncertainties and present our approaches in addressing them. Note that for simplicity, we will use the term *uncertainty* here to generally refer to the various unknown factors that may be caused by external disturbances, inherent uncertainties/randomness, or the lack of understanding in current methods.

In the rest of the paper, we will discuss several specific types of uncertainties, and introduce our approaches in modeling, analysis and mitigation of them. In particular, in Section 2, we will consider execution uncertainties that may be captured by the *weakly-hard* constraints, and present a cross-layer framework that we have been developing to address them, including formal methods at the functional layer, co-design algorithms across functional and software layers, and scheduling methods at the OS layer. In Section 3, we will focus on the uncertainties that arise from the adoption of neural networks in perception and decision making, and present

ICCAD '20, November 2–5, 2020, Virtual Event, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415768

our approaches for addressing them. These include timing uncertainty in DNN inference tasks and our *DART* framework that offers deterministic and bounded response time analysis of those tasks; uncertainty from adversarial input attacks and our approach that provides bounded output range analysis of neural networks for a potentially-perturbed input space; and uncertainty in the behavior of neural-network controlled systems (NNCSs) and our *ReachNN** tool that performs reachability analysis and safety verification of NNCSs. In Section 4, we will introduce our safety-assured adaptation methods that adapt the system under disturbances or other changing requirements. We offer concluding remarks and future directions in Section 5.

2 MITIGATING EXECUTION UNCERTAINTY WITH WEAKLY-HARD PARADIGM

One major type of uncertainty during the operation of autonomous systems is *timing uncertainty* in the execution of system functions, i.e., how much time it takes for certain function to complete and whether that meets the deadline. Traditionally, there are two main approaches to manage such uncertainties. In hard real-time systems, designers remove the consideration of all timing uncertainties in execution through exhaustive modeling to achieve strict bounds on the timing behavior of the system, and do not allow any deadlines to be missed. Soft real-time systems, on the other hand, permit the system to arbitrarily violate bounds and miss deadlines; in this case, system designers can achieve at best probabilistic guarantees on the system's behaviors. However, only using hard deadlines often leads to over-pessimistic designs or over-provisioning of system resources, while soft deadlines cannot provide the safety guarantees needed by many autonomous systems.

To address these challenges, we advocate to develop systems with a cross-layer *weakly-hard* paradigm, where deadline misses are allowed in a bounded manner [6]. This is motivated by the fact that many system components can tolerate a certain degree of timing uncertainties and deadline violations and still satisfy their functional and extra-functional properties such as safety, stability and performance, as long as those violations are bounded and properly managed. Fig. 1 highlights the concept of our cross-layer weakly-hard system design. At the functional layer, verification and validation of functional properties, such as correctness, safety and stability, are conducted with consideration of potential deadline misses. At the software layer, system functionalities are synthesized into the form of software tasks and their communication mechanisms, while the weakly-hard timing behavior (e.g., deadline miss pattern) is analyzed and evaluated against the requirements at the functional layer. At the OS layer, scheduling algorithms and runtime mechanisms are provided to ensure correct execution of weakly-hard tasks in the presence of deadline misses.

Previous works in weakly-hard systems focus on either schedulability analysis at the software layer or control stability analysis at the functional layer. We believe that, however, it is important to take a cross-layer approach and address functional properties, software implementation, and OS support in a holistic framework, as the weakly-hard constraints set for managing timing uncertainties have to ensure that 1) functional properties can indeed be satisfied under the allowed deadline misses, and 2) software implementation and OS support can bound the deadline misses as specified.

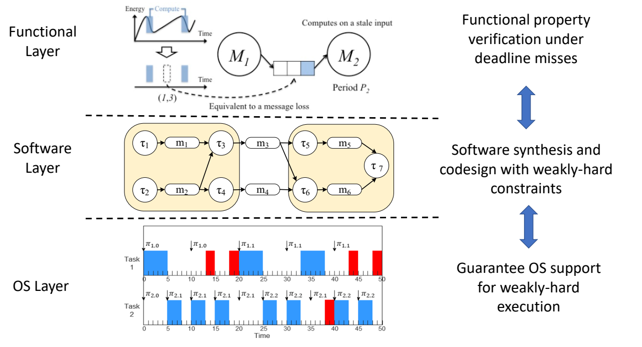
Functional Layer
Software Layer
OS Layer
Energy
Compute
Time
(1,3)
Equivalent to a message loss
Computes on a stale input
M_1
M_2
Period P_2
τ_1 m_1 τ_3 m_3 τ_5 m_5 τ_7
τ_2 m_2 τ_4 m_4 τ_6 m_6
Task 1
Task 2
Time
Functional property verification under deadline misses
Software synthesis and codesign with weakly-hard constraints
Guarantee OS support for weakly-hard execution

Figure 1: Overview of our cross-layer weakly-hard design framework for the modeling, analysis and mitigation of execution uncertainties.

Note that our framework can be applied to other types of execution uncertainties beyond timing violations. For instance, the most common form of weakly-hard constraints is the (m, K) model [6, 27], which simply specifies that among any K consecutive executions, at most m instances can miss their execution deadlines. In our framework, such misses may be caused by timing uncertainties, or are the results of soft errors [44], or are proactively decided by the system (i.e., the executions are skipped [33, 34]).

2.1 Functional Analysis and Verification under Weakly-Hard Constraints

At the functional layer, the key issue for leveraging weakly-hard paradigm is to evaluate precisely to what degree the systems can tolerate deadline misses, e.g., as described by the (m, K) model. We will model and analyze various functional properties, such as safety, stability, and performance under such weakly-hard constraints.

Safety. One of the most important functional properties is safety. In our weakly-hard framework, we consider whether the system with (m, K) constraints will ever enter a pre-specified unsafe state set.

In literature, the work in [23] models a weakly-hard system with linear dynamic as a hybrid automaton, whose reachability is then verified by SpaceEx [24]. In [17], the behavior of a linear weakly-hard system is transformed into a program, and whether its unsafe specification can be met is checked by program verification techniques such as abstract interpretation and SMT solvers. The work in [61] considers discrete-time systems described as labelled transition systems, and explores logical relationships among weakly-hard constraints with various m and K values. The approach improves the verification efficiency by only checking the satisfaction boundary, rather than the whole configuration space of (m, K).

Our framework addresses the safety of nonlinear weakly-hard systems for the first time in literature. In [32], our approach derives a safe initial set for any given (m, K) constraint, that is, starting from any initial state within such set, the system will always stay within the same safe state set under the given weakly-hard constraint. Specifically, we first convert the infinite-time safety problem into a finite one by finding a set satisfying both *local safety* and *inductiveness*. Local safety ensures that the system stays in the safe region within K steps, and inductiveness guarantees that the system will go back to the initial set after K steps. Thus the set that satisfies both local safety and inductiveness is theoretically guaranteed to be a safe initial set. To make estimating such a set tractable, we make two assumptions – exponential stability of the system without deadline misses and Lipschitz continuity of system dynamics – to help bound the system behavior under different situations. Then we can abstract the problem as a one-dimensional problem and use linear programming (LP) to obtain a certified safe initial set.

We observe that in practice, the assumptions in [32] are sometimes hard to satisfy and the parameters of exponential stability are difficult to obtain. Moreover, while the scalar abstraction provides high efficiency, experiments indicate that the estimation is often overly conservative. Thus, we further relax the aforementioned assumptions by leveraging state space discretization and graph theory in [29]. Sepcifically, we first discretize the safe state set X into grids, and then try to find the grid set that satisfies both local safety and inductiveness. For each property, we build a directed graph, where each node corresponds to a grid and each directed edge represents the mapping between grids with respect to reachability. We are then able to leverage dynamic programming and inverse search algorithms to construct the initial safe set. This approach is implemented in an open-source tool called *SAW*[1].

Control stability and performance. From the perspective of system resilience, we can measure a system's ability to tolerate disturbances in terms of weakly-hard constraints, e.g., under what kind of (m, K) constraints the system controllers can remain stable and how much their performance is affected. For instance, in literature, the work in [25] provides an analytical bound for the deadline miss ratio that can ensure the stability of a distributed embedded controller. The work in [48] presents a more general framework to analyze the control performance with respect to a specific sequence of deadline miss pattern. The work in [57] studies the worst-case control performance of an LQR controller under deadline misses.

In our framework, we consider linear time-invariant (LTI) systems implemented in Logical Execution Time (LET) paradigm [28]. In [45], we quantify control performance as the capability of a controller to bring the system back to the equilibrium state after a disturbance. Assuming that a zero control input is applied if the control task misses its deadline, we model the closed-loop system as a switched system depending on the deadline hit/miss pattern. The stability of the switched system is checked by finding whether the eigenvalues of the transition matrix of a hyper-period lie inside the unit circle. In [44], we use the typical worst-case analysis (TWCA) [63] to bound the number of deadline misses in the presence of transient fault and approximate control performance by exploring valid deadline hit/miss patterns. We also build an event-based simulator to capture the exact deadline hit/miss pattern and use it to obtain the worst-case control performance.

Network properties. Many safety-critical wireless networked systems such as those in the industrial wireless, connected vehicles, and infrastructure monitoring domains are subject to difficult-to-model external disturbances and internal uncertainties [33, 66]. For example, transient communication faults can occur due to radio interference, mobile units changing the network topology, nodes locally deciding to shut off their radios to conserve energy or perform other tasks, or malicious network attacks such as jamming and flooding. Furthermore, wireless communication protocols typically need to manage a tradeoff between the real-time performance and the resource usage of communication. Taken together, it becomes clear that the hard real-time analysis of a wireless networked system would be infeasible. In [33], we motivate the weakly-hard paradigm to model disturbances and uncertainties in wireless networked systems as a promising tool to obtain the deterministic guarantees required to prove safety and liveness properties. Our first attempt at such an analysis considers real-time applications such as environmental monitoring and industrial control applications running on wireless networked systems [59]. Each inter-task message that needs to be sent over wireless medium is treated as a task in its own right; each message is transmitted by one-to-all Glossy floods as part of a statically-scheduled low-power wireless bus round [21, 22]. The parameters of the underlying Glossy flood for a given message determine the weakly-hard behavior, the duration, and the power consumption of a given message transmission task. We provide a scheduler that determines not only task release times, but also the required Glossy flood parameters, in order to meet the real-time constraints.

2.2 Software Synthesis and Codesign with Weakly-Hard Constraints

At the software layer in our framework, the key issue is to analyze the weakly-hard behavior of software implementations for system functionalities (e.g., deadline miss patterns in their execution) based on schedulability analysis that will be introduced later in Section 2.3, and then evaluate whether such weakly-hard behavior can meet the requirements on functional properties using the techniques introduced above in Section 2.1. For instance, for a control function, we will analyze the deadline miss pattern of its software implementation, i.e., what (m, K) constraints it can satisfy, considering execution resources and other software tasks in schedulability analysis. We will then evaluate whether such (m, K) constraints can ensure the safety, stability, and performance requirements of this control function. Moreover, based on such cross-layer analysis, we will explore the various software implementation options (i.e., perform software synthesis) in a codesign process, with holistic consideration of functional properties (e.g., safety, stability, performance, security, fault tolerance) and platform properties (e.g., schedulability, energy consumption).

In our prior work [70], we have presented a series of codesign methods for traditional hard real-time systems, exploring the design space in a quantitative and automated manner. Our weakly-hard design framework introduced in Fig. 1 extends this vision and particularly focuses on leveraging the additional *slack* from allowed deadline misses to improve various system objectives. For instance, in [45], we develop a codesign approach for adding security monitoring tasks to resource-limited automotive electronic systems by finding feasible weakly-hard constraints on existing control tasks and exploring the allocation, priority, and period assignment of

[1]https://github.com/551100kk/SAW.git

added security monitoring tasks. Intuitively, adding more monitoring tasks and running them more frequently (i.e., with smaller periods) can help improve the system security level, but may lead to more deadline misses for the existing control tasks. Therefore, our codesign approach quantitatively trades off between security and control performance, while ensuring that the deadline misses will not lead to instability of the control functions.

In [44], we similarly find feasible weakly-hard constraints on certain control tasks, and leverage their execution slacks to add fault tolerance techniques [65] for soft error detection and correction (e.g., by embedding detection techniques in tasks or applying redundant task execution). We develop a codesign approach that trades off between system control performance and error coverage, while ensuring the stability of control tasks. Our results demonstrate that leveraging weakly-hard paradigm can significant improve the error coverage over traditional hard deadlines.

2.3 OS Support for Weakly-Hard Paradigm

For practical use and general acceptance of the weakly-hard paradigm, we need system software and OS-level support to maintain functional and temporal correctness at runtime. Research on new systems mechanisms and algorithms is sought for the efficient and reliable execution of weakly-hard tasks, including scheduling policies leveraging weakly-hard models for better resource utilization and predictability, runtime mechanisms for handling deadline-missed jobs for safe and robust operations, and support for multi-core architectures for scalability.

Scheduling algorithms. Task scheduling with given (m, K) constraints has been conducted extensively, starting with Bernat et al. [6] under traditional task-level fixed-priority scheduling. Extensions of this schedulability work have been studied, such as for bi-modal execution [7, 51] and non-preemptive tasks [43], by making strong assumptions on task timing behavior, e.g., fixed initial release offset and fixed period with no release jitter. However, we believe that such assumptions limit their applicability to recent CPS applications, especially autonomous systems that require flexibility and adaptability. Recent work [54, 63] relaxes some of these assumptions but at the expense of high analysis complexity, making them difficult to use at runtime for online admission control in adaptive systems. Moreover, existing scheduling policies cannot take full advantage of the flexibility allowed by weakly-hard constraints.

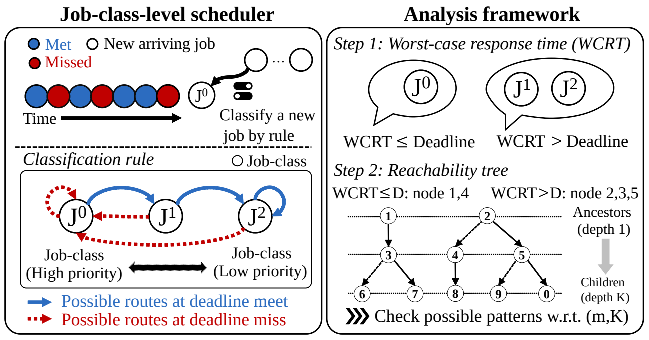
Job-class-level scheduler
Analysis framework
Met
New arriving job
Missed
Time
Classify a new job by rule
Classification rule
Job-class
J0
J1
J2
Job-class (High priority)
Job-class (Low priority)
Possible routes at deadline meet
Possible routes at deadline miss
Step 1: Worst-case response time (WCRT)
WCRT ≤ Deadline
WCRT > Deadline
Step 2: Reachability tree
WCRT≤D: node 1,4
WCRT>D: node 2,3,5
Ancestors (depth 1)
Children (depth K)
1 2 3 4 5 6 7 8 9 0
Check possible patterns w.r.t. (m,K)

Figure 2: Job-class-level scheduling for weakly-hard tasks.

As an effort to address these issues, we have developed new scheduling algorithms for weakly-hard systems. In [13], we proposed *job-class-level scheduling*, which significantly improves the scheduling efficiency and flexibility of weakly-hard real-time tasks. The key to this work is in the classification of jobs of each task, called *job-classes*, and the assignment of priority to each class of jobs (Fig. 2). Unlike traditional task-level priority scheduling, a task can have as many priority levels as the number of job-classes it has, and the priority of each job is determined by the priority of its corresponding job-class. This approach also enables decomposing the complex weakly-hard schedulability problem into two sub-problems that are easier to solve: (i) analysis of the worst case response time for individual job-classes, and (ii) finding all possible scheduling patterns, which can be modeled as a reachability tree. Based on experiments, this new scheduler outperforms prior work, with as much as 56% higher taskset schedulability on a single processor. The analysis running time under our new scheduler is much faster than that under conventional fixed-priority schedulers [54], thereby making it applicable to runtime admission control.

Deadline miss handling. Resource efficiency and design flexibility of a system can vary depending how deadline missed-jobs are handled. Various assumptions have been made in the literature of weakly-hard systems, but their effects have not been comparably studied. In [12], we classified possible deadline handling schemes into four categories: (1) *job abort* terminates a job immediately when the job misses its deadline, (2) *delayed completion* allows a deadline-missed job to continue to run until it completes, (3) *job pre-skip* determines whether to execute a job or not at its release time, based on available slack time or predetermined patterns, and (4) *job post-skip* allows a deadline-missed job to continue over the period but skips the next job. Based on this classification, we developed a unified runtime framework that supports all these schemes. Although it was originally implemented in the Linux kernel, the framework design is easily applicable to other OSs, and analyzing the four schemes under various conditions is an interesting research direction.

Multi-core support. Despite the popularity of multi-core processors in recent embedded platforms, studies for weakly-hard systems are still in their early stages. Task scheduling in multi-core systems is typically classified into partitioned, semi-partitioned, and global scheduling. Among them, partitioned scheduling allows direct application of existing single-core schedulers to multi-core platforms on a per-core basis, but it may not be able to efficiently utilize the slack made available by weakly-hard tasks. On the other hand, semi-partitioned and global scheduling have the potential to improve resource utilization without sacrificing the safety of weakly-hard systems. We are extending our job-class-level scheduler [13] with the semi-partitioned approach. The end-to-end latency of a chain of tasks executing across multiple CPU cores is also an important topic. In our recent work [11], we presented a *chain-based scheduler* for multi-core platforms, which improves end-to-end latency by taking advantage of permissible deadline misses of intermediate tasks. Lastly, shared memory resources, such as caches, memory buses, and DRAM banks, are critical sources of timing unpredictability in recent multi-core platforms [3, 37, 39], but to the best of our knowledge, no prior work has studied these issues for weakly-hard systems. Addressing these issues is part of our future work.

3 ADDRESSING UNCERTAINTY FACING NEURAL NETWORKS

Neural networks have been widely applied in autonomous systems, both in perception and decision making. Fig. 3 shows a typical example. The physical plant describes the system behavior and is represented by an ordinary differetial equation (ODE). At each sampling instant t, raw data is sampled from various sensors, e.g. camera, Lidar, Radar. Then the perception module runs a convolutional neural network to extract the state information from the raw data. The following decision making module computes the control input based on the system state by a fully-connected neural network. Finally the control input is actuated in the next period to drive the system evolution.

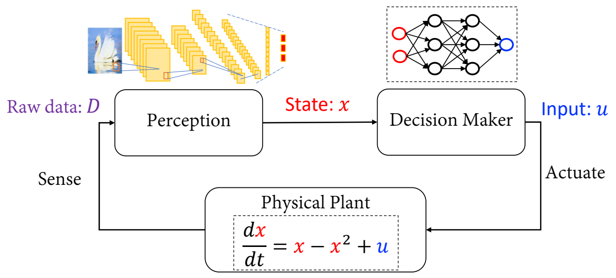

Figure 3: Illustration of a neural-network involved system.

Due to the high complexity and numerous parameters, neural networks bring inherent uncertainties to the modern autonomous systems. In this section, we introduce our recent works on addressing the uncertainties of neural networks across both perception and decision making components.

3.1 Runtime System for Time-Predictable DNN

While DNNs have been widely applied in autonomous systems and much research has been conducted to optimize their structure, limited attention has been given to mitigating the *timing uncertainties* of executing multiple DNNs, specifically when they are integrated into resource-constrained heterogeneous platforms. For instance, existing modern DNN frameworks, such as TensorFlow and PyTorch, only provide sequential execution patterns with no priority support. Those frameworks do not take into account timing interference among different DNN tasks due to the contention on shared computing resources. Thus, response time of those tasks may become unpredictably long in the worst case while leaving system resources underutilized. Researchers have proposed several frameworks such as S^3DNN [67], Neurosurgeon [36] and DeepMon [35] that target on lowering the DNN inference latency. However, none of those frameworks bound or mitigate the inference time uncertainty. In other words, they are not amenable to real-time schedulability analysis and do not offer deterministic guarantees on the response time of DNN tasks.

To address those limitations, we developed DART [62] as shown in Fig. 4, a real-time DNN framework that offers deterministic and bounded response time to real-time DNN inference tasks and concurrent execution of various types of DNN model on heterogeneous multi-core and GPU-integrated platforms. DART introduces new

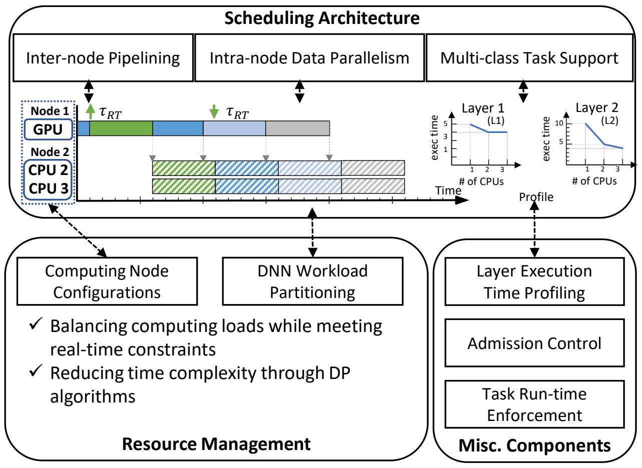
Scheduling Architecture
Inter-node Pipelining
Intra-node Data Parallelism
Multi-class Task Support
Node 1
GPU
Node 2
CPU 2
CPU 3
τ_RT
τ_RT
Time
Layer 1 (L1)
Layer 2 (L2)
exec time
# of CPUs
Profile
Computing Node Configurations
DNN Workload Partitioning
✓ Balancing computing loads while meeting real-time constraints
✓ Reducing time complexity through DP algorithms
Resource Management
Layer Execution Time Profiling
Admission Control
Task Run-time Enforcement
Misc. Components

Figure 4: Real-time DNN execution framework for heterogeneous embedded platforms.

abstractions to deal with the different resource requirements of individual layers of DNNs and to facilitate the co-utilization of CPU and GPU in inference job execution. Specifically, it (i) creates the pipeline stages of each task in a way to balance the contention across a given set of processors, (ii) configures a set of computing nodes considering heterogeneity and parallelism levels, and (iii) allocates computing resources to meet timing constraints and to minimize task response time. In addition, the design of DART enables a systematic formulation of the real-time DNN scheduling problem into a distributed acyclic scheduling problem. Our analysis captures DNN job execution over the proposed abstractions of stages, nodes, and workers, and takes into account system overheads including inter-node communication, GPU preemption, and data copy time. DART has demonstrated its ability to bound and mitigate the execution time uncertainties on both popular ARM and x86 platforms. Our experimental results indicate that DART significantly outperforms existing frameworks, by up to 98.5% shorter worst-case response time for real-time tasks while simultaneously achieving up to 17.9% higher throughput for best-effort tasks.

There are several research directions that can be built upon DART. First, distributed IoT devices can be modeled as nodes of the DART pipeline to offload workloads and mitigate uncertainties. Second, sophisticated real-time GPU schemes [38, 47] can be applied for better resource utilization and other types of hardware accelerators, such as FPGAs, can be co-used for larger DNNs. Third, shared memory resources and their timing interference [55] are worth investigating to minimize timing uncertainties and provide better performance isolation between real-time and best-effort DNNs.

3.2 Adversarial Attacks and Output Range Analysis for Neural Networks

With the rising use of emerging DNN applications in safety-critical systems, much attention has been given to the reliability and trustworthiness of DNN inference output against disturbances such as malicious adversarial attacks. Adversarial examples were first founded in [56]. Intuitively, an adversarial example could cause the network to make a false prediction with small perturbations that are unrecognizable by humans. To achieve better robustness against adversarial examples, adversarial training and output range analysis are two main topics studied recently.

Dependability against adversarial attacks. As the output of DNNs is determined by input data, trained weights, and intermediate results stored in memory, adversarial *fault data injection* attacks such as physical laser beam [8] and row hammer attacks [50] can easily manipulate these parameters and have DNNs to generate different output. Fig. 5 shows an example adversarial DNN attack. The model is expected to infer red traffic light and stop sign from the driving scene. However, malicious attacker can launch a fault injection attack that alters the critical model parameters and intermediate computation results of DNNs running in an untrusted and/or uncertified software environment, which in turn leads to a false classification result and threatens the safety of the system. For dependable DNN execution, it is imperative to protect the parameters critical to output correctness from malicious data modifications. The leakage of the critical parameters including data structures and location in memory should also be prevented because such information is required by fault injection attacks to analyze the weakness of DNN models. Even if attacks happen, the degree of faulty output should be contained within an acceptable and predictable range.

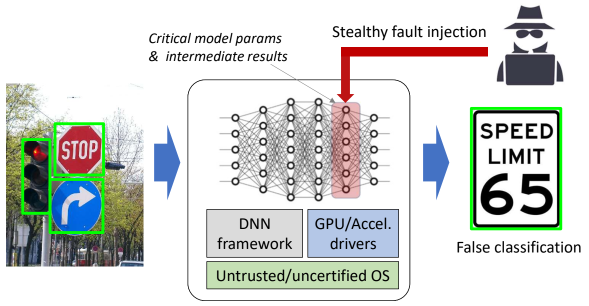

Figure 5: Adversarial fault injection attacks to DNNs.

Recently much work has focused on the implications of external disturbances to the outputs of DNNs and tried to guide protection strategies. TensorFI [42], BinFI [10], and Ares [52] are fault injection tools that analyze DNN models and identify critical bits, and they are useful not only to assess the impact of soft errors but also to identify weaknesses. However, those studies does not provide a concrete protection mechanism for a variety of fault injection attacks. There are also studies [18, 26, 40] on improving the privacy of DNN inference by executing the entire DNN model inside Intel SGX enclaves, but due to the performance limitations of SGX, those approaches are not suitable for real-time applications.

We are currently investigating to address these limitations and to achieve dependable real-time DNN execution environment. Our on-going work leverages secure SGX enclaves but protects only the critical part of real-time DNN tasks which are vulnerable to potential fault injection attacks. In order to choose the right set of layers for protection while satisfying real-time requirements, we are developing a dynamic-programming based approach to find a layer protection configuration for each task based on layer-wise DNN time and SDC (Silent Data Corruption) profiling mechanisms. We utilize a machine-learning based SDC prediction method to reduce the time for estimating SDC rates for all possible layer protection configurations. Our approach is not limited to SGX; with additional integrity checks, it can be applied to trusted hypervisors [16] and other hardware security extensions, e.g., ARM TrustZone.

Bounding output range for certifiable DNNs. Beyond adversarial protections, output range analysis provides the certified bound of a neural network with a given input space, which theoretically evaluates the robustness of a neural network.

Specifically, output range analysis solves the following problem: given a neural network f and the input range $\mathcal{X}$, compute the output range of $f(\mathcal{X})$. Due to the highly nonlinearity of neural networks, it is generally difficult to compute the exact range. In most cases, we use an overapproximation $\overline{\mathcal{Y}}$ such that $f(\mathcal{X}) \subseteq \overline{\mathcal{Y}}$. Such overapproximation can provide an explicit bound for determining whether the neural network output falls into an unwanted region. In the context of adversarial robustness, for a data point x with the concerned error bound ϵ, we can define the input space as the small L_∞-norm box $[x-\epsilon, x+\epsilon]$ and estimate the output range as $[\underline{l}_1, \bar{l}_1] \times \cdots \times [\underline{l}_g, \bar{l}_g] \cdots \times [\underline{l}_n, \bar{l}_n]$ for n labels, where $\underline{l}_g$ ($\bar{l}_g$) denotes the lower (upper) bound of the output in terms of the ground truth. If $\underline{l}_g \geq \bar{l}_i$, for $i = 1, \cdots, n$, we can safely say the the neural network is locally robust on x with respect to the perturbation ϵ.

In our work [31], we propose a layer-wise refinement method that bridges propagation-based methods with mixed-integer linear programming (MILP) by using sliding windows. Specifically, we use a convex polygonal relaxation (over-approximation) of the activation functions to cope with the nonlinearity. This allows us to encode the relaxed problem into a mixed integer linear program (MILP), and control the tightness of the relaxation by adjusting the number of segments in the polygon. Starting with a segment number of 1 for each neuron, which coincides with a linear programming (LP) relaxation, our approach heuristically selects neurons layer by layer to iteratively refine this relaxation. To tackle the increase of the number of integer variables with tighter refinement, we bridge the propagation-based method and the programming-based method by dividing and sliding the layer-wise constraints: given a length of sliding window s, for the neuron in layer l, we only encode the constraints of the layers between $l-s$ and l. With these methods, we can effectively manage the size of MILP and handle deep networks.

3.3 Reachability Analysis and Safety Verification of NNCSs

Neural networks are not only widely used for perception, but have been increasingly applied to control and general decision making. Learning-enabled autonomous systems, especially neural-network controlled systems (NNCSs) have recently become the subject of intense research and demonstrated great promises. An NNCS is essentially a closed-loop system controlled by a neural network. Unlike the output range analysis, verification of an NNCS will need to capture not only the behavior of the neural-network controller but the interaction between system dynamics and the neural-network controller. In [30], we propose a new reachability analysis approach, ReachNN, to verify NNCSs via Bernstein polynomial approximation. Specifically, given an input state space, ReachNN constructs an error-bounded approximation based on Bernstein polynomial for the neural-network controller and casts the NNCS into a tractable closed-loop system. This allows us to verify properties of the NNCS's reachable space by using existing reachability tools and handle general neural networks. One advantage of using a Bernstein polynomial-based approximation is that it allows us to establish a bound on the approximation error based on the Lipschitz constant of the neural network.

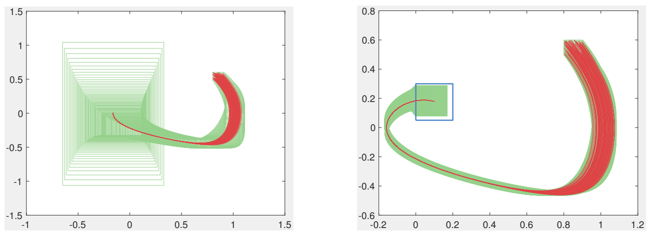
(a) Output Range Analysis **(b) ReachNN***

Figure 6: Reachability analysis result computed by output range analysis only and by ReachNN* on an NNCS. The controller is a heterogeneous neural network that has ReLU and tanh activation functions. The red curves are simulation traces. The green boxes represent the reachable sets and the blue box is the target region.

We implement this technique as an open-source tool called ReachNN* [20] [2], with enhanced efficiency and scalability by GPU acceleration in the error estimation, which brings about 7× to 422× performance improvement. In Fig. 6, we show that applying only output range analysis techniques to verify NNCSs can result in rapid growth of the reachable set overapproximation. On the other hand, ReachNN* can provide a much tighter estimation for all control steps and successfully prove the reachability property.

In the ReachNN* work, we also observe that Lipschitz continuity of a neural network plays a major role in the construction of reachable sets for NNCSs. Larger value of the Lipschitz constant leads to higher computational cost and looser reachable set overapproximations. This observation inspires us to further explore approaches that ease the verification through training. Our work [19] develops a novel verification-aware knowledge distillation (KD) framework that transfers the knowledge of a trained network to a new and easier-to-verify network by reducing the value of Lipschitz constant. We form a two-objective optimization problem that considers knowledge distillation and the value of Lipschitz constant jointly. We show that our technique enables state-of-the-art verification tools for NNCSs to achieve tighter reachable sets and reduce their computation time. This work opens up the possibility of reducing verification complexity by influencing how a system is trained.

4 SAFETY-ASSURED ADAPTATION IN DYNAMIC ENVIRONMENT

Autonomous systems are often deployed in highly dynamic and uncertain environment that could put changing requirements on system objectives. For instance, a robot may need to strengthen its security protection in an adversarial environment, to improve soft error tolerance in radioactive surroundings, or to enhance control performance in difficult-to-navigate terrains. To address these changing scenarios, intelligent and safe adaptation of the system is needed.

In the literature, *safe learning*, i.e., the additional consideration of safety requirements during training or deployment of machine learning components, has garnered attention from multiple research communities [1, 2, 14, 69]. From a system's point of view, a well-known approach to ensure safety at runtime is the *Simplex* architecture [53]. In this architecture, a safety controller is used to ensure stabilization of the physical system in a *known* domain of the system state space, in addition to a baseline controller and an advanced controller. Recently, [49] extends this idea to autonomous systems, where a neural or AI controller is used as the advanced controller to generate high-performance control actions. Frequent and intermittent switching between the safety controller and the neural controller, however, can lead to undesirable behavior and reduced performance. The authors in [68] propose to reduce such control switching by *repairing* the neural controller, i.e., making it safe in states that would have required the safety controller to intervene, by using control actions generated by the safety controller at runtime.

In our work, we have been focusing on *safe adaptation and switching* among multiple controllers or multiple modes. For instance, a simple controller may only handle a portion of the possible scenarios and fail for the rest, while a robust controller may be able to handle more scenarios but is too expensive or excessive for those simple scenarios. By considering both the performance and the efficiency, adaptively selecting a proper controller under different scenarios may greatly benefit the system [4]. Traditional adaptive control algorithms dynamically adjust the parameters of the feedback controller for better control performance. For instance, gain scheduling [41] selects different control parameters at different operation points to eliminate the uncertainty introduced by linearization. Model reference adaptive control (MRAC) [5] attempts to minimize the error between the actual system output and a desired one generated by a reference model. Dual adaptive control [60] not only provides good control performance, but also minimizes the uncertainty of the model parameter estimation. Such an idea can be extended to multiple control modes in autonomous systems. For instance in [15], a shorter sampling period (fast mode) in control loops can make faster response to external disturbances while consume more computation/control resource than a longer sampling period (slow mode). Adaptation between these two modes is promising for stability and resource saving. While these adaptive control laws are designed to tackle the disturbance and uncertainty, the control performance is usually the design objective under the constraint of stability. However, the disturbance may cause the stable controller entering unsafe state space.

Our approach considers an autonomous system that is equipped with multiple controllers to handle different situations, and different from most methods in the literature, ours formally guarantees system safety during adaptation. At a sampling instant, to meet the system objectives at the time, an adapter can choose an appropriate controller to compute the control input, or skip the control input

[2]https://github.com/JmfanBU/ReachNNStar

computation and apply zero input. A key issue here is how to design the adapter to guarantee the system safety while identifying the best controller to choose for the objectives. In [34], we make the first attempt, where we consider the adaptation between a model-based controller (e.g., model predictive control) and zero input for a dynamical system under disturbance. To guarantee safety, we first compute a *strengthened safe set* based on the notion of *robust control invariant* and *backward reachable set* of the underlying safe controller. Intuitively, the strengthened safety set represents the states at which the system can accept any control input at the current step and be able to stay within safe states, with the underlying safe controller applying input from the next step on. We then develop a monitor to check whether the system is within such strengthened safe set at each control step. Whenever it is found that the system state is out of the strengthened safe set, the monitor will require the system to apply the underlying safe controller for guaranteeing system safety. To achieve a better control performance, we leverage a deep reinforcement learning (DRL) approach to learn the mapping from the current state and the historical characteristics to the skipping choices, which implicitly reflects the impact of specific operation context and environment that denoted by disturbance.

In [58], we extend the approach from [34] to handle systems equipped with multiple neural-network controllers, which can be generated by different design methods or design hyper-parameters. In this work, the system safety is ensured by control invariant, which differs from [34]. Specifically, we first approximate each controller by Bernstein polynomials, which follows the idea in [30], but with state space partitioning to obtain a more precise hybrid controller. Combining the hybrid controllers with the dynamical system, we obtain a hybrid system with bounded disturbance as an over-approximation of the original system. Then the robust invariant set for each controller is obtained via semi-definite programming [64]. Intuitively, the invariant is a safe set that ensure every possible controlled trajectory starting from it will never leave it. The union of these invariant sets then build a safe configuration space, within which we design a DRL agent to learn a run-time switching strategy with with a safe guard rule. Experiments show that our approach can ensure system safety while significantly reduce the overall energy consumption.

Our ongoing work extends the idea of safety-assured adaptation to a cross-layer framework, as illustrated in Fig. 7. We consider architecture platforms where the software implementations of multiple functions are scheduled to share computation and communication resources. During runtime adaptation, resources may be re-allocated from certain tasks (e.g., by reducing their execution periods, skipping their execution instances as in [34], or choosing more efficient controllers for them as in [58]) to other tasks that are more in need at the time. The key is to ensure that under such adaptation, functions meet their functional layer requirements such as safety, stability, and performance; *and* their implementations on the architecture layer satisfy constraints on schedulability, resource usage, energy consumption, etc. We address such adaptation by formulating it as a multi-objective optimization problem with constraints on safety, schedulability, and other functional and architectural constraints. The objectives may include saving control energy consumption (same as in [34] and [58]), optimizing the control performance, improving the performance of other functions

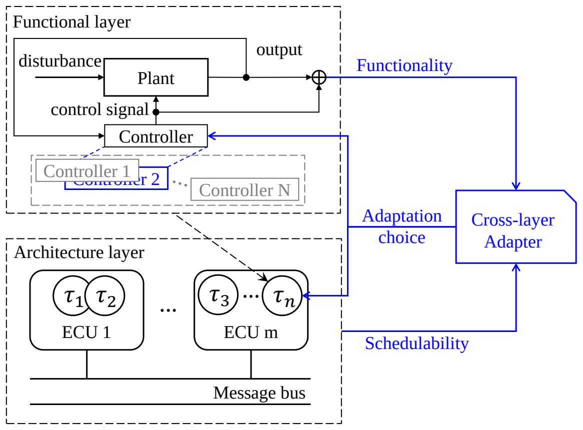

Figure 7: Cross-layer adaptation framework. The adapter cuts across the functional layer and the architecture layer. At the functional layer, control functions interact with their corresponding physical plants (one control function with multiple candidate controllers is shown). There could be other types of functions as well (e.g., sensing functions; not shown in the figure). At the architecture layer, multiple software tasks implementing the functions are scheduled to share computation and communication resource. During runtime, the adapter may choose different controllers for control functions at the functional layer, or adjust their software implementations at the architecture layer, to meet the system needs while ensuring safety.

(e.g., sensing functions), enhancing security with monitoring tasks as in [45], or improving fault tolerance as in [44].

5 CONCLUDING REMARKS

Dependable operation of autonomous systems requires proper handling of external disturbances and inherent uncertainties. In this paper, we lay out various aspects of this challenge and potential solutions, from uncertainty modeling and mitigation using a weakly-hard paradigm, to formal verification, robust execution, and intelligent adaptation of neural network based components. We posit that a cross-layer approach that cuts across functional, software and OS layers is essential to mitigate and manage uncertainties. Initial results from domains such as connected and autonomous vehicles demonstrate the effectiveness of our approaches in coping with uncertainties that arise in a variety of scenarios, and adapting the system for better resource usage while preserving safety. Several future directions such as predictive monitoring and runtime adaptation can be explored to further the capabilities of autonomous systems in dealing with uncertainties, and we believe such efforts can be built upon our initial endeavor presented in this paper.

ACKNOWLEDGMENTS

We gratefully acknowledge the support from National Science Foundation (NSF) grants 1646497, 1834701, 1834324, 1839511, 1724341, and Office of Naval Research (ONR) grant N00014-19-1-2496.

REFERENCES

[1] J. Achiam et al. Constrained policy optimization. In *International Conference on Machine Learning*, 2017.

[2] M. Alshiekh et al. Safe reinforcement learning via shielding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] B. Andersson et al. Schedulability analysis of tasks with corunner-dependent execution times. *ACM TECS*, 17(3):1–29, 2018.

[4] K. Åström and B. Wittenmark. *Adaptive Control: Second Edition.* Dover Books on Electrical Engineering. Dover Publications, 2013.

[5] I. Barkana. Simple adaptive control – a stable direct model reference adaptive control methodology – brief survey. *International Journal of Adaptive Control and Signal Processing*, 28(7-8):567–603, 2014.

[6] G. Bernat et al. Weakly hard real-time systems. *IEEE transactions on Computers*, 50(4):308–321, 2001.

[7] G. Bernat and R. Cayssials. Guaranteed on-line weakly-hard real-time systems. In *IEEE Real-Time Systems Symposium (RTSS)*, 2001.

[8] J. Breier et al. Deeplaser: Practical fault attack on deep neural networks. *arXiv preprint arXiv:1806.05859*, 2018.

[9] S. Checkoway et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Conference on Security*, 2011.

[10] Z. Chen et al. BinFI: an efficient fault injector for safety-critical machine learning systems. In *SC*, pages 1–23, 2019.

[11] H. Choi et al. Chain-based fixed-priority scheduling of loosely-dependent tasks. In *IEEE International Conference on Computer Design (ICCD)*, 2020.

[12] H. Choi and H. Kim. Work-in-progress: A unified runtime framework for weakly-hard real-time systems. In *Brief Presentations of RTAS*, 2019.

[13] H. Choi et al. Job-class-level fixed priority scheduling of weakly-hard real-time systems. In *IEEE RTAS*, 2019.

[14] Y. Chow et al. A lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8092–8101, 2018.

[15] X. Dai et al. A dual-mode strategy for performance-maximisation and resource-efficient cps design. *ACM TECS*, 18(5s):1–20, 2019.

[16] D. De Niz et al. Mixed-trust computing for real-time systems. In *IEEE RTCSA*, 2019.

[17] P. S. Duggirala and M. Viswanathan. Analyzing real time linear control systems using software verification. In *RTSS*, pages 216–226. IEEE, 2015.

[18] T. Elgamal and K. Nahrstedt. Serdab: An IoT framework for partitioning neural networks computation across multiple enclaves. *arXiv:2005.06043*, 2020.

[19] J. Fan et al. Towards verification-aware knowledge distillation for neural-network controlled systems. In *International Conference on Computer-Aided Design (ICCAD)*, 2019.

[20] J. Fan et al. Reachnn*: A tool for reachability analysis of neural-network controlled systems. In *ATVA*, 2020.

[21] F. Ferrari et al. Low-power wireless bus. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2012.

[22] F. Ferrari et al. Efficient network flooding and time synchronization with glossy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 73–84, 2011.

[23] G. Frehse et al. Formal analysis of timing effects on closed-loop properties of control software. In *IEEE Real-Time Systems Symposium*, pages 53–62, Dec 2014.

[24] G. Frehse et al. Spaceex: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pages 379–395. Springer, 2011.

[25] D. Goswami et al. Relaxing signal delay constraints in distributed embedded controllers. *IEEE Trans. on Control Systems Technology*, 22(6):2337–2345, 2014.

[26] K. Grover et al. Privado: Practical and secure DNN inference with enclaves. *arXiv preprint arXiv:1810.00602*, 2018.

[27] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *IEEE Transactions on Computers*, 44(12), 1995.

[28] T. A. Henzinger et al. Giotto: a time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, Jan 2003.

[29] C. Huang et al. Saw: A tool for safety analysis of weakly-hard systems. In *Computer Aided Verification (CAV)*, pages 543–555, 2020.

[30] C. Huang et al. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s), Oct. 2019.

[31] C. Huang et al. Divide and slide: Layer-wise refinement for output range analysis of deep neural networks. *IEEE TCAD*, 2020.

[32] C. Huang et al. Formal verification of weakly-hard systems. In *ACM Conference on Hybrid Systems: Computation and Control (HSCC)*, 2019.

[33] C. Huang et al. Exploring weakly-hard paradigm for networked systems. In *Workshop on Design Automation for CPS and IoT (DESTION)*, 2019.

[34] C. Huang et al. Opportunistic intermittent control with safety guarantees for autonomous systems. In *Design Automation Conference (DAC)*, 2020.

[35] L. N. Huynh et al. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *MobiSys*, 2017.

[36] Y. Kang et al. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

[37] H. Kim et al. Bounding memory interference delay in cots-based multi-core systems. In *IEEE RTAS*, 2014.

[38] H. Kim et al. A server-based approach for predictable gpu access with improved analysis. *Journal of Systems Architecture*, 88:97–109, 2018.

[39] H. Kim and R. Rajkumar. Predictable shared cache management for multi-core real-time virtualization. *ACM TECS*, 17(1):1–27, 2017.

[40] T. Lee et al. Occlumency: Privacy-preserving remote deep-learning inference using SGX. In *MobiCom*, 2019.

[41] D. J. Leith and W. E. Leithead. Survey of gain-scheduling analysis and design. *International Journal of Control*, 73(11):1001–1025, 2000.

[42] G. Li et al. TensorFI: A configurable fault injector for tensorflow applications. In *IEEE Symposium on Software Reliability Engineering Workshops*, 2018.

[43] J. Li et al. Providing real-time applications with graceful degradation of QoS and fault tolerance according to (m, k)-firm model. *IEEE Transactions on Industrial Informatics*, 2(2):112–119, 2006.

[44] H. Liang et al. Leveraging weakly-hard constraints for improving system fault tolerance with functional and timing guarantees. In *2020 International Conference on Computer-Aided Design (ICCAD)*, 2020.

[45] H. Liang et al. Security-driven codesign with weakly-hard constraints for real-time embedded systems. In *International Conference on Computer Design (ICCD)*, pages 217–226, 2019.

[46] C. Lin et al. Security-Aware Design Methodology and Optimization for Automotive Systems. *ACM TODAES*, 21(1):18:1–18:26, December 2015.

[47] P. Patel et al. Analytical enhancements and practical insights for mpcp with self-suspensions. In *IEEE RTAS*, 2018.

[48] P. Pazzaglia et al. Beyond the weakly hard model: measuring the performance cost of deadline misses. In *ECRTS*, 2018.

[49] D. T. Phan et al. Neural simplex architecture. In *NASA Formal Methods Symposium*, pages 97–114. Springer, 2020.

[50] A. S. Rakin et al. Bit-flip attack: Crushing neural network with progressive bit search. In *International Conference on Computer Vision*, pages 1211–1220, 2019.

[51] P. Ramanathan. Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):549–559, Jun 1999.

[52] B. Reagen et al. Ares: A framework for quantifying the resilience of deep neural networks. In *ACM/IEEE Design Automation Conference (DAC)*, 2018.

[53] D. Seto et al. The simplex architecture for safe online control system upgrades. In *American Control Conference (ACC)*, 1998.

[54] Y. Sun and M. D. Natale. Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. *ACM TECS*, 16(5s):171, 2017.

[55] N. Suzuki et al. Coordinated bank and cache coloring for temporal protection of memory accesses. In *IEEE CSE*, 2013.

[56] C. Szegedy et al. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[57] E. P. van Horssen et al. Performance analysis and controller improvement for linear systems with (m, k)-firm data losses. In *ECC*, pages 2571–2577, 2016.

[58] Y. Wang et al. Energy-efficient control adaptation with safety guarantees for learning-enabled cyber-physical systems. In *International Conference on Computer-Aided Design (ICCAD)*, 2020.

[59] K. Wardega and W. Li. Application-Aware Scheduling of Networked Applications over the Low-Power Wireless Bus. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2020.

[60] B. Wittenmark. Adaptive dual control methods: An overview. In C. BÁNYÁSZ, editor, *Adaptive Systems in Control and Signal Processing 1995*, IFAC Postprint Volume, pages 67 – 72. Pergamon, Oxford, 1995.

[61] S.-L. Wu et al. Efficient system verification with multiple weakly-hard constraints for runtime monitoring. In *20th International Conference on Runtime Verification (RV)*, 2020.

[62] Y. Xiang and H. Kim. Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference. In *RTSS*, pages 392–405, 2019.

[63] W. Xu et al. Improved deadline miss models for real-time systems using typical worst-case analysis. In *ECRTS*, pages 247–256, 2015.

[64] B. Xue and N. Zhan. Robust invariant sets computation for switched discrete-time polynomial systems. *arXiv preprint arXiv:1811.11454*, 2018.

[65] B. Zheng et al. Analysis and Optimization of Soft Error Tolerance Strategies for Real-Time Systems. In *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, October 2015.

[66] B. Zheng et al. Design and analysis of delay-tolerant intelligent intersection management. *ACM Transaction on Cyber-Physical Systems*, 2019.

[67] H. Zhou et al. S3DNN: Supervised streaming and scheduling for GPU-accelerated real-time DNN workloads. In *IEEE RTAS*, 2018.

[68] W. Zhou et al. Runtime-safety-guided policy repair. In *20th International Conference on Runtime Verification (RV)*, 2020.

[69] W. Zhou and W. Li. Safety-aware apprenticeship learning. In *International Conference on Computer Aided Verification (CAV)*, pages 662–680, 2018.

[70] Q. Zhu and A. Sangiovanni-Vincentelli. Codesign methodologies and tools for cyber–physical systems. *Proceedings of the IEEE*, 106(9):1484–1500, 2018.