Towards Self-Aware and Safe Autonomous Systems

Nikil Dutt
Dutt Research Group
UC Irvine

Acknowledgements: DRG and IPF team

Diversity in Emerging Systems and Applications

Abstraction Layers

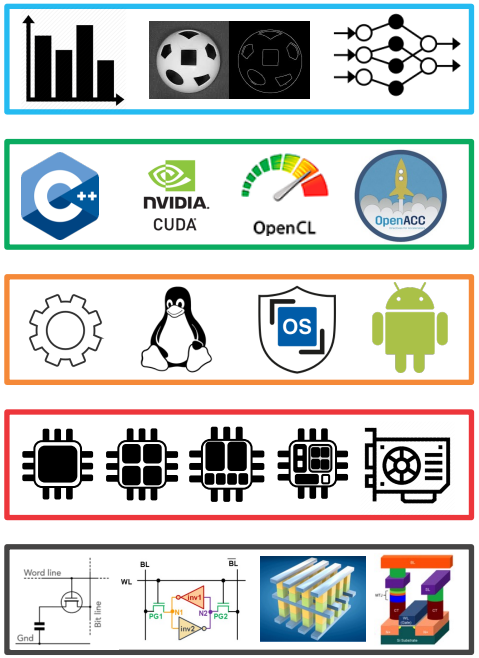
Applications

Vendor Libraries

Kernel

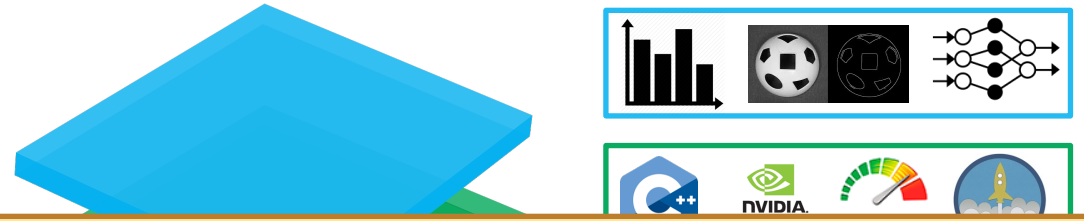
Architecture

Device Tech



Diversity in Emerging Systems and Applications

Abstraction Layers



- **"Triple Whammy" of changes:**
  1. **Platforms (e.g., process variability)**
  2. **Environment (e.g., temperature)**
  3. **Applications (e.g., processing, memory…)**
- **Static design techniques can't cope with changes**
- **Increasing need for runtime adaptivity**
- **Deploy Computational Cognitive Intelligence (CCI) principles**

Computational Cognitive Intelligence (CCI)

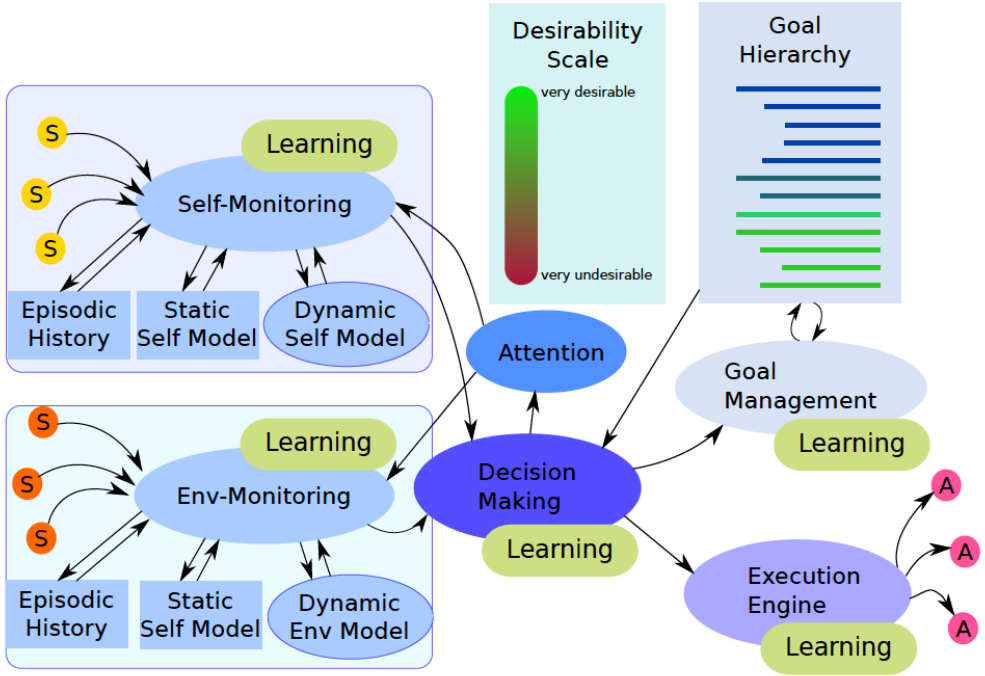
CCI: biologically inspired, but adapted for computing systems

- Explicit models of ***self*** and ***environment*** for adaptivity/autonomy
- Computational abstraction of biological cognition: self- and environmental-models; goal management; understanding & reasoning; contextualization; and history
- Use **CCI reference architecture** for driving adaptive system design
- Demonstrated utility of CCI in several embedded, CPS, and IoT use-cases (see DAC 2023 Position Paper)

Exemplar CCI Reference Architecture

CCI's bio-inspired abstractions:

- self- and environmental-models
- history
- goal management
- decision making
- attention & contextualization
- learning

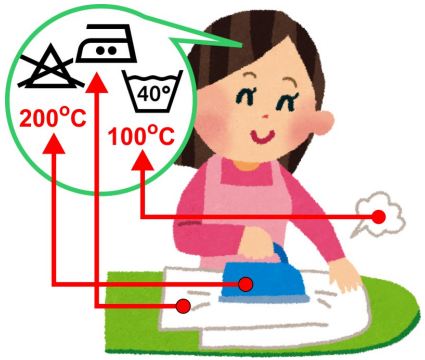


Reflex vs Reflect

Reflexive, Reactive

- Actions driven solely on external feedback
  - E.g., our autonomic nervous systems

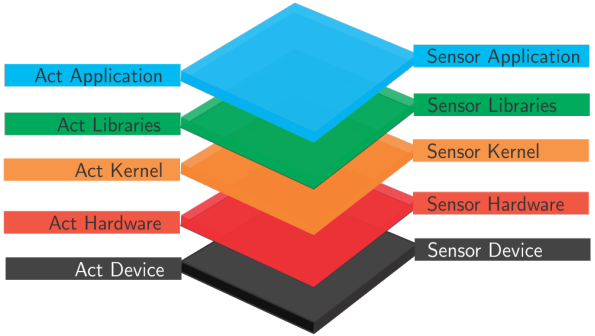
Reflection, Introspection



- Consider past and future outcomes
  - E.g., planning, strategies, policies, …

What does an open-loop system look like ?

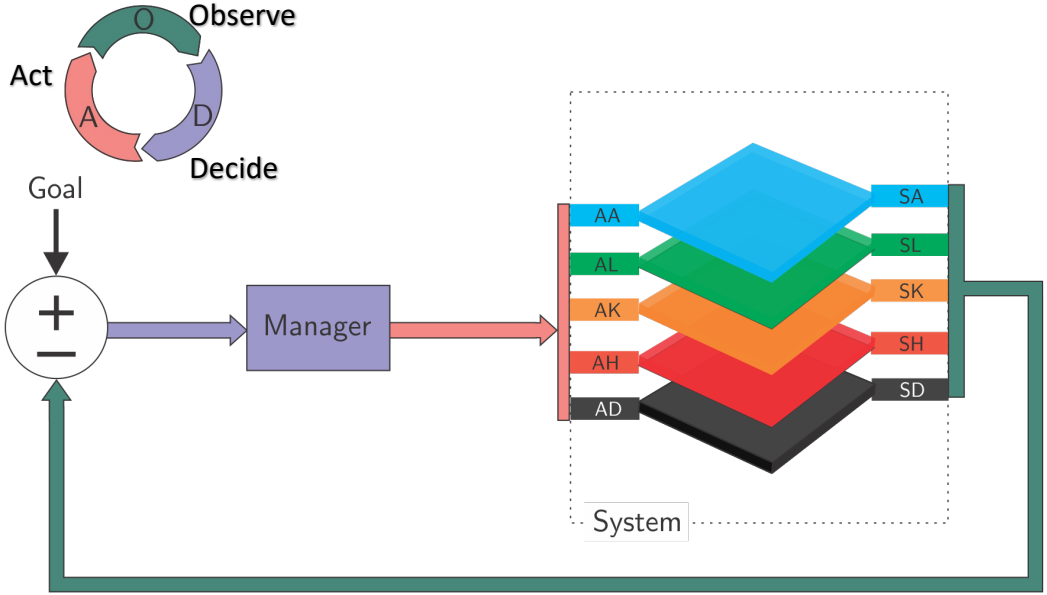
Abstraction Layers



System

- no awareness, but potential

What does a closed-loop system look like ?



System

- no awareness, but potential

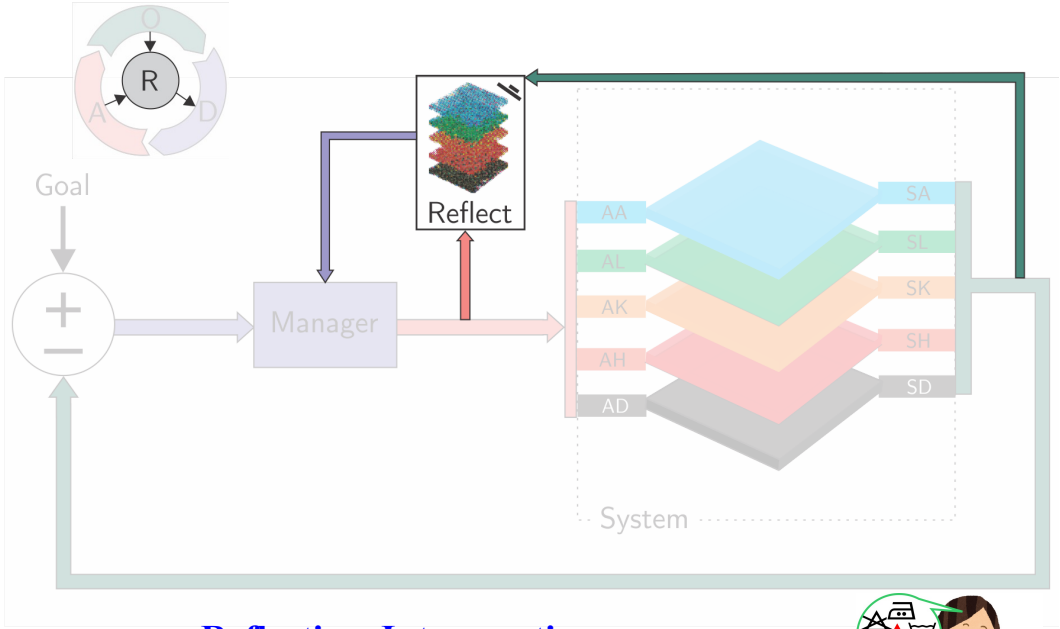
Reactive System

- reasoning, but still not self-aware

Reflexive, Reactive

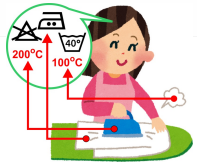
- **Actions driven solely on external feedback**
  - **E.g., our autonomic nervous system**

What does a self-aware system look like ?



Reflection, Introspection

- **Consider past and future outcomes**
  - **E.g., planning, strategies, policies, …**



System

- no awareness, but potential

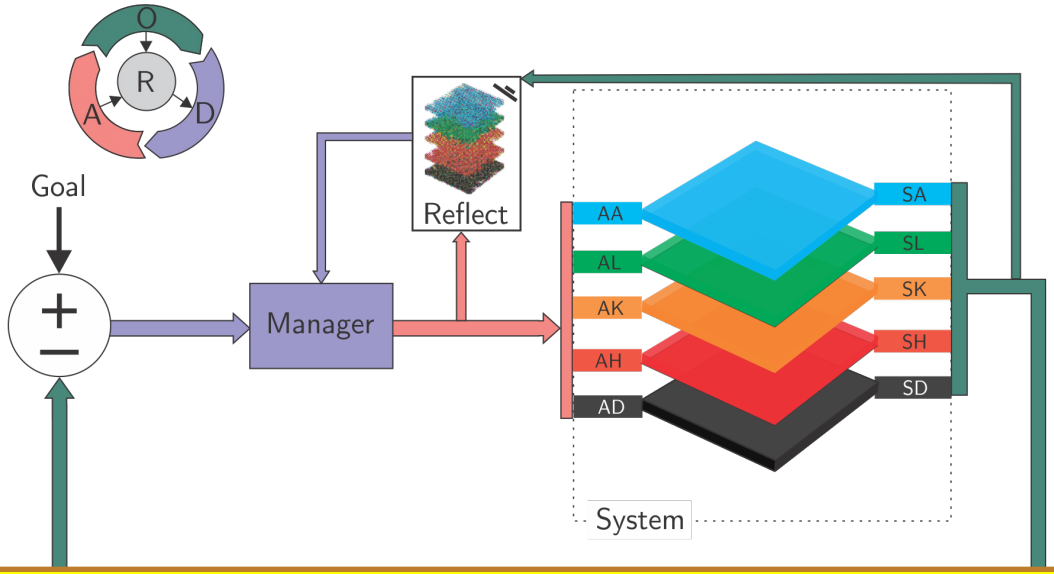
Reactive System

- reasoning, but still not self-aware

Reflective System

- self-aware reasoning

What does a self-aware system look like ?



System

- no awareness, but potential

Reactive System

- reasoning, but still not self-aware

Reflective System

Computational SA: Observe-Reflect-Decide-Act

Our Recent Work on Computational SA for Memory Mgt

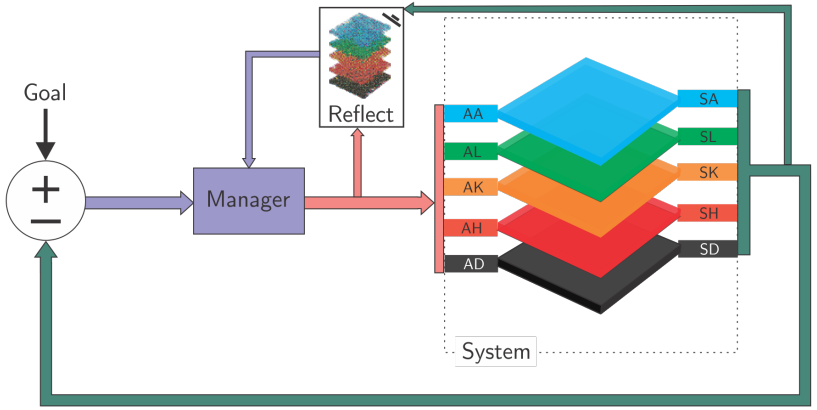
Architectural-level

ESL 2020

Self-adaptive memory approximation: A formal control theory approach

TECS 2022

SEAMS: Self-Optimizing Runtime Manager for Approximate Memory Hierarchies



System-level

ESWEEK 2019

HESSLE-FREE: Heterogeneous Systems Leveraging Fuzzy Control for Runtime Resource Management

Ongoing

Scalable deployment of MARS in datacenter racks for capacity crisis mitigation. (joint work with Meta)

Application-level

IESS 2019

Workload characterization for memory management in emerging embedded platforms

ESWEEK 2021

Chauffeur: Benchmark Suite for Design and End-to-End Analysis of Self-Driving Vehicles on Embedded Systems.

NAS 2022

ProSwap: Period-aware Proactive Swapping to Maximize Embedded Application Performance

Challenges in deploying CCI

Computational Cognitive Intelligence (CCI): critical for adaptive systems

Many open challenges:

- ❑ Incorporate symbolic/human understanding
- ❑ Enable self-monitoring systems for dynamic runtime verification/validation
- ❑ Achieve what-ifs via low-overhead reflection and introspection
- ❑ Safe, interpretable decision making
- ❑ Dealing with unknown-unknowns: empower systems to handle new experiences

UCIrvine

Trust, But Verify: Towards Self-Aware, Safe, Autonomous Systems

Minjun Seo, Caio de Melo, Ahmed Nassar*, Saehanseul Yi, Jong-Chan Kim**, Biswadip Maity, Bryan Donyanavard***, Rachid Karami, Walaa Amer, **Nikil Dutt**, Fadi J. Kurdahi

Center for Embedded & Cyber-physical Systems
University of California
Irvine, CA USA

*NVIDIA Corp., San Jose, CA
**Kookmin Univ., Korea
*** San Diego State Univ.

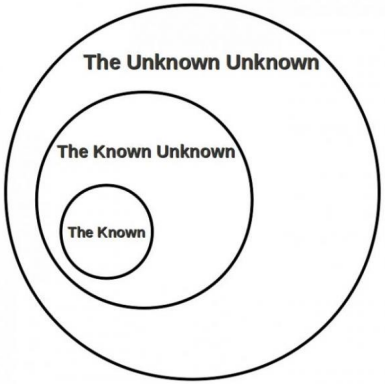
The (un)Known (un)Knowns

… there are *known knowns*; there are things we know we know. We also know there are *known unknowns*; that is to say we know there are some things we do not know. But there are also *unknown unknowns*—the ones we don't know we don't know…
Donald Rumsfeld (former US Sec. Defense), Feb 2000

Your Application

Awareness \ Knowledge	yes	no
yes	**Known Knowns** *Things we are aware of and understand.*	**Known Unknowns** *Things we are aware of but don't understand.*
no	**Unknown Knowns** *Things we understand but are not aware of.*	**Unknown Unknowns** *Things we are neither aware of nor understand.*

System bugs, failures, AI unexplainability, other disruptive events that we aware of, but can occur unpredictably



???

Unforeseen events that we do not know how to solve at design time. **Emergent behavior**

[Courtesy Fadi Kurdahi]

Sources of Exceptional Behavior

Imperfect Software

bugs and vulnerabilities, etc…

Imperfect Hardware

Harsh Environment, Variability, Aging, etc…

Imperfect Models

under- or over-specification, AI unpredictability, etc…

Need for Runtime Verification (RV)

- Can't fully verify a system at design time
  - Doesn't guarantee correctness at runtime due to environmental and dynamic nature
  - Catastrophic failures can result from uncaught errors or unpredicted dynamic behaviors/conditions

Incorrect code reuse

Arithmetic error

Thread deadlock error

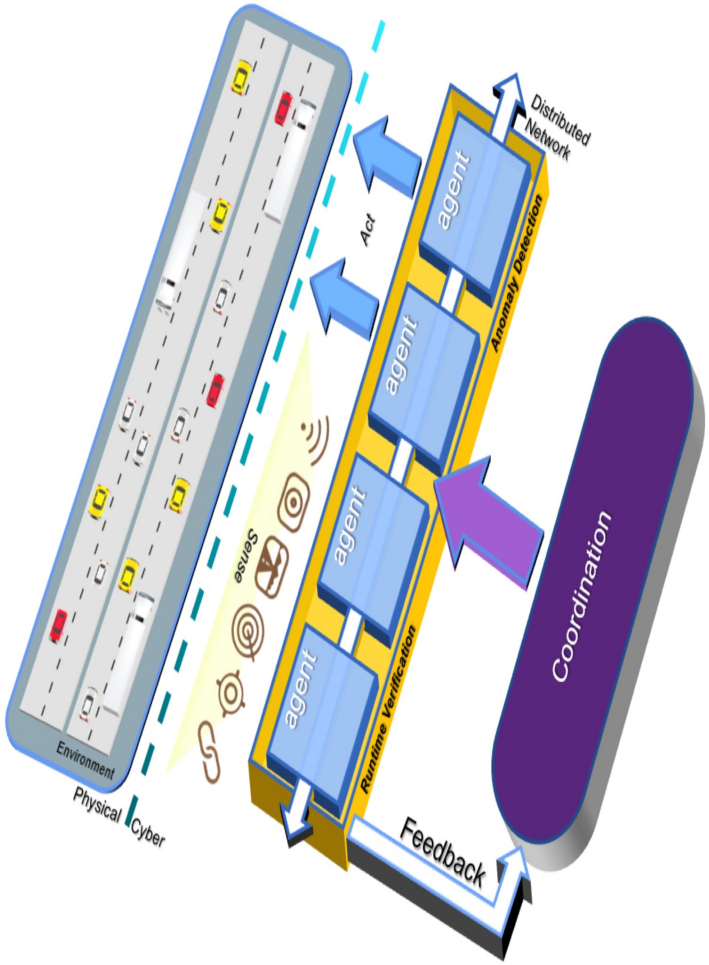
Uber's self-driving crash

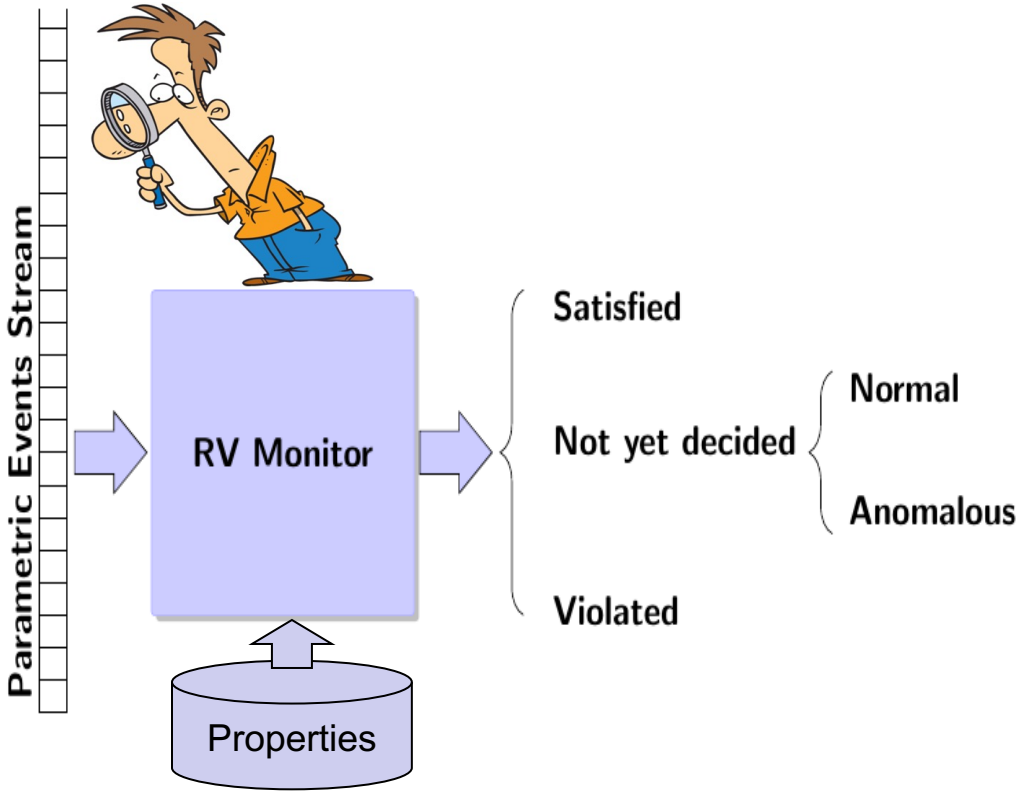
- In-situ (on site) runtime verification needed
  - Continuously monitor systems to ensure adherence to system requirements

[Courtesy Fadi Kurdahi]

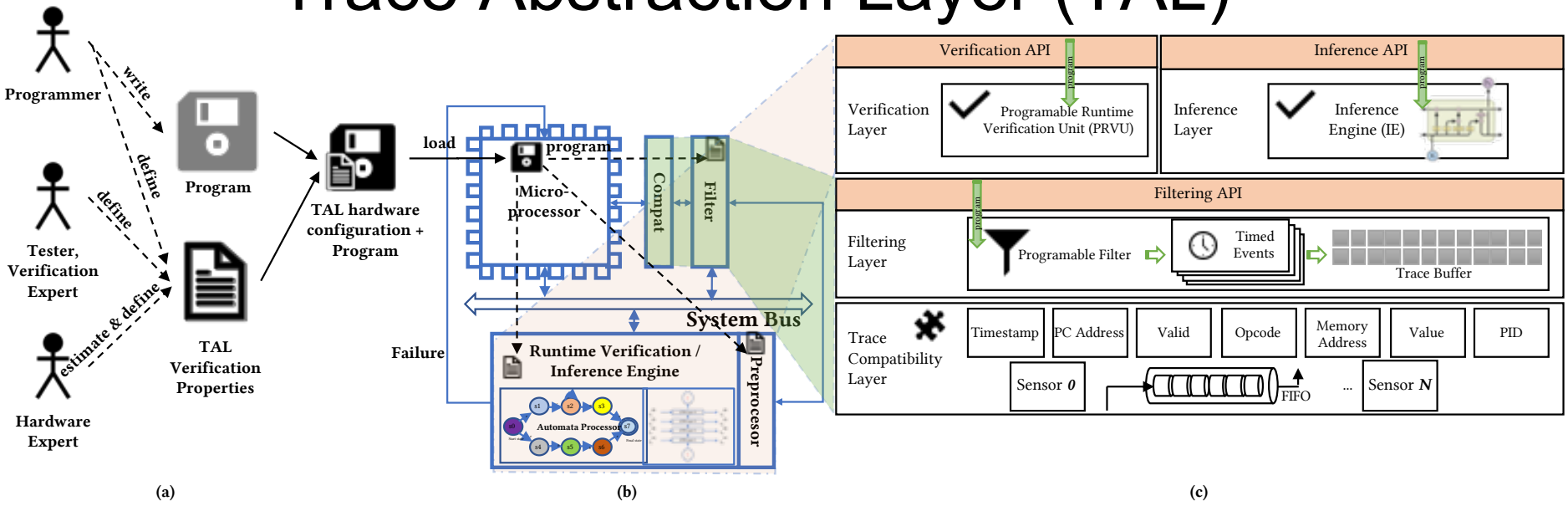
Runtime Verification

- In-situ (on site) runtime verification
  - Continually monitor system to ensure adherence to system requirements



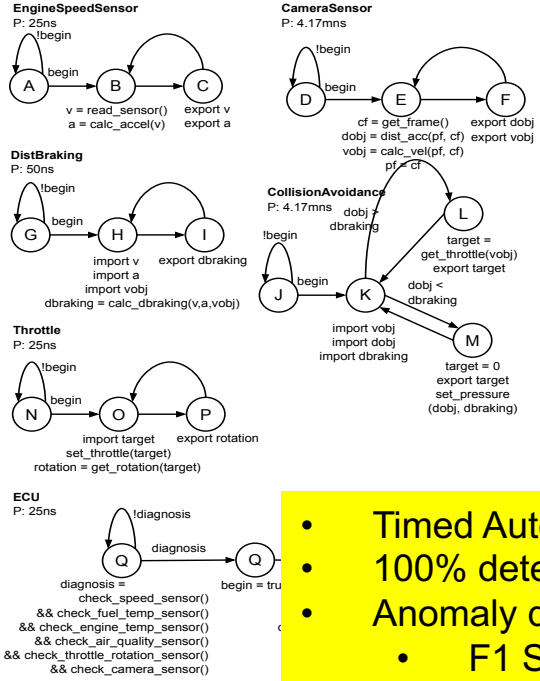


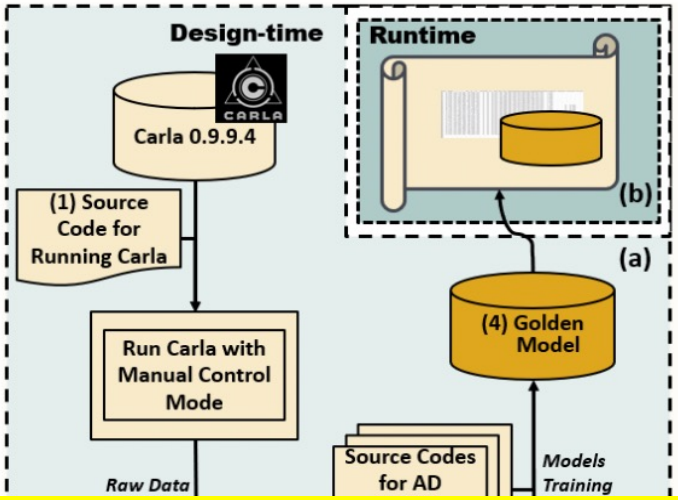
Trace Abstraction Layer (TAL)

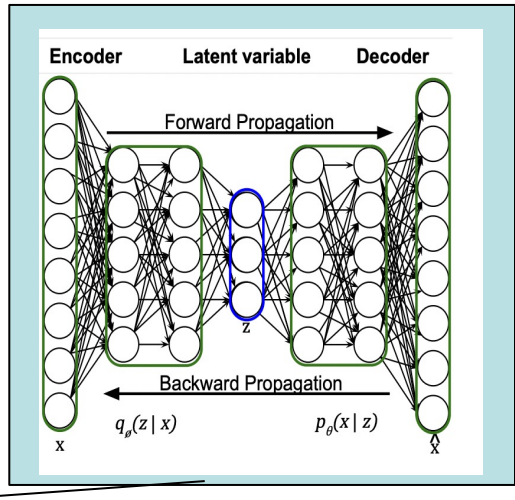


- Non-Intrusive
  - Probe effect
  - Separation of implementer and verifier
- Minimal Overhead
  - Hardware vs software instrumentation (10-1000x speedup)
- Timely detection
  - To give sufficient time for the system to respond
- Comprehensive monitoring
  - Processors
  - Soc Components (memory, buses, interfaces, etc..)
  - Sensors

Using TAL in a Collision Avoidance Subsystem







- Timed Automata modeling of properties
- 100% detection of properties failures (known unknows).
- Anomaly detection (unknown unknowns)
  - F1 Score: 77% (Carla Simulator)
- Hardware-assisted detection
  - very fast (20 cycles vs ~ 900K cycles in software)

Safety

Functional Correctness

Velocity (*v*) must be obtained before calculating acceleration (*a*)	Acceleration distance and velocity calculation must be executed after acquiring current frame (*cf*) and must have previous frame (*pf*)	
Value of each variables, ***v, a, vobj, dobj, distBraking, targetThrottle, injectionValue***, must have 0 or positive value		
Distance for braking must be calculated after calculation of current velocity (v) and acceleration (a) and target vehicle's velocity (v_{obj})		
If distance for braking is less than the distance of target vehicle, throttling must be re-calculated	If distance for braking is greater than and equal to the distance of target vehicle, throttling must be set to 0	
Speed sensor must read sensor value every 25 *ns*	Camera sensor must capture a frame every 4.17 *ms*	DistBraking and CollisionAvoidance must be executed every 50 *ns*	ECU must run at a rate of 25 *ns*

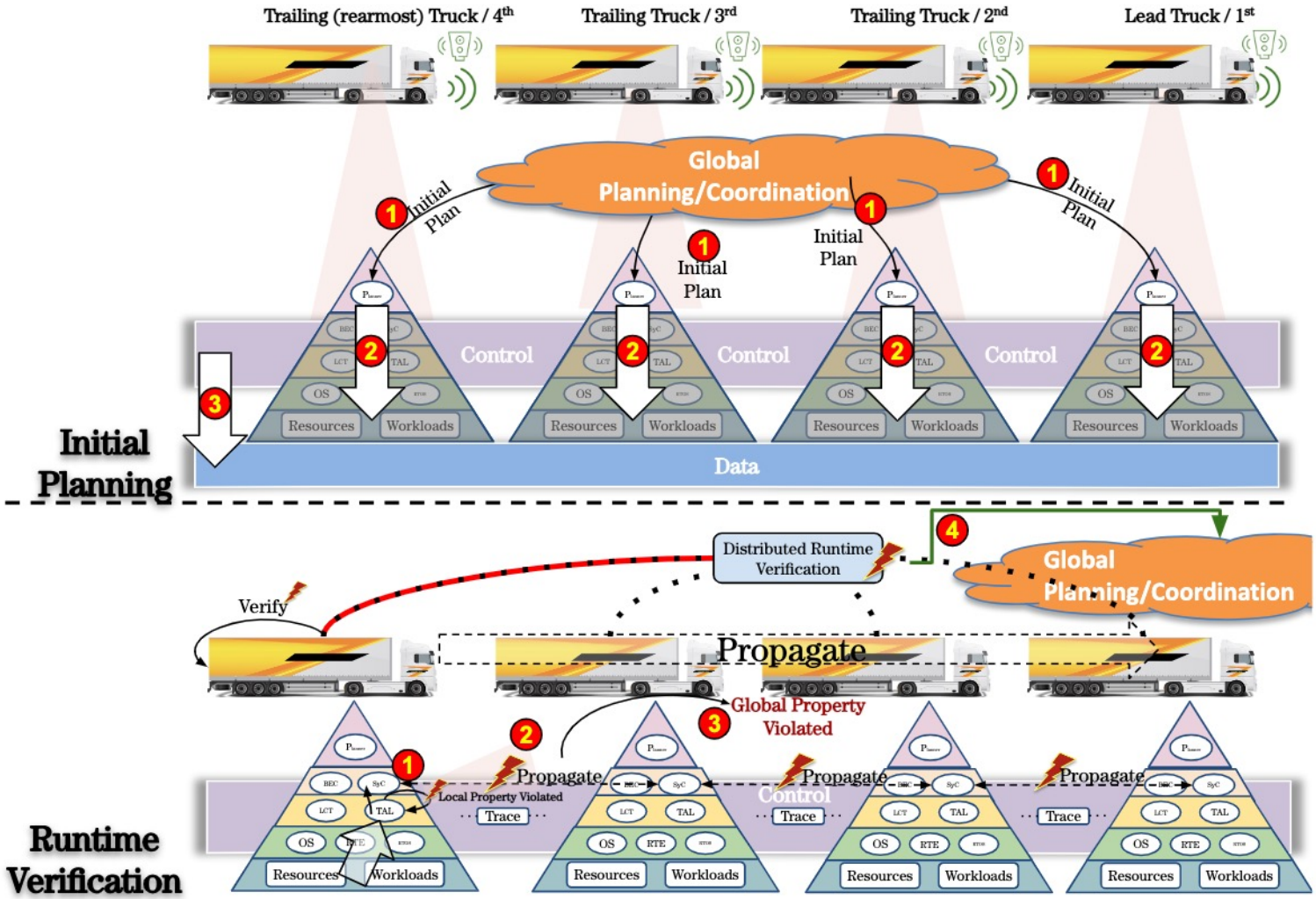
System Correctness

Timers must be triggered every 25 *ns*	Synchronization checking

Table 5. F1 scores for Carla Experiments

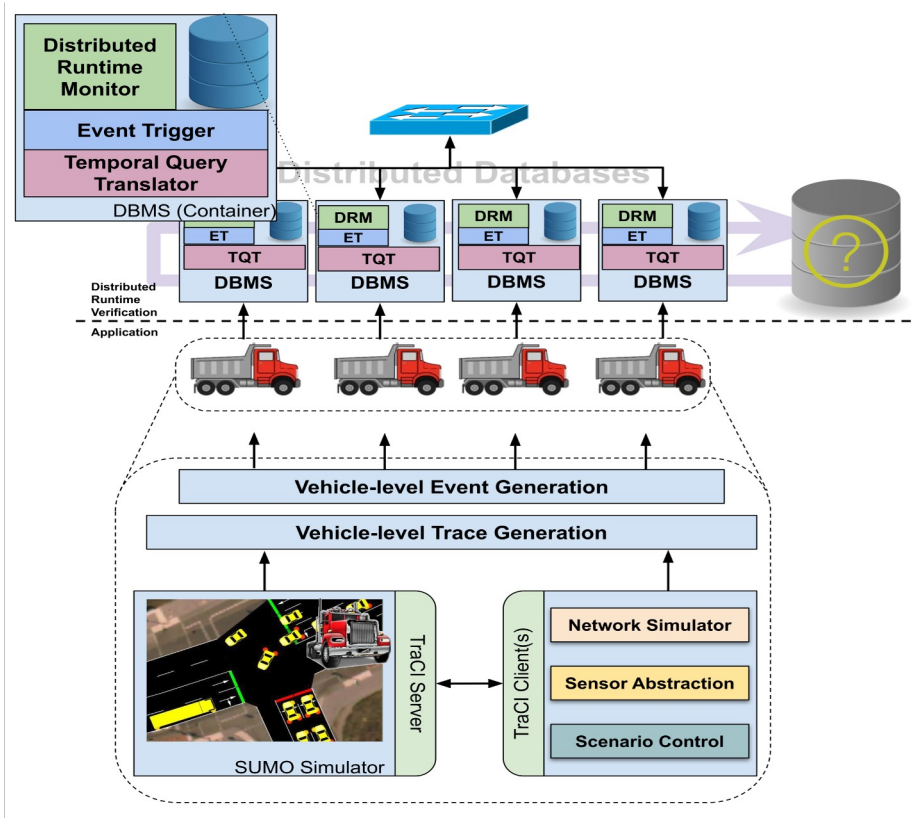
Algorithm	Training Data	Test Data
Auto-Encoder	75%	66%
KNN	67%	64%
PCA	74%	64%
VAE	83%	77%

Generalization to Multiple, Distributed IPF Systems (IPF 2.0)



DRIVE: Distributed Runtime Verification for Collective Autonomous Systems

- Distributed data storage across a platoon of Self-driving trucks
- Vehicles benefit from perception of other members in the platoon
- Allow for verification of global properties through aggregation of local data
- Results in faster responses to hazards

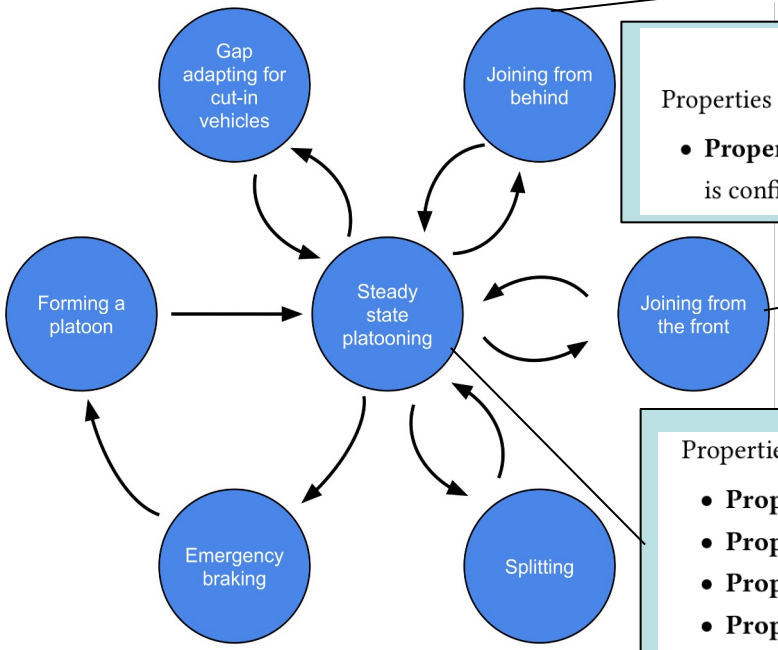


DRIVE: Platoon Behavior (Properties)

- Platoon should follow defined states
- Properties extracted to be verified at runtime (SAE standard)
- Spatio-temporal metrics are used to verify correctness
- TLTL grammar

Properties for joining from behind:

- **Property 5**. Time gap between $T[n]$ and $T[new]$ should be within t_{min} and t_{max} for k seconds before the join is confirmed;
- **Property 6**. Lateral offset difference between $T[n]$ and $T[new]$ should be within r_{min} and r_{max} for k seconds before the join is confirmed.



Properties for joining from the front:

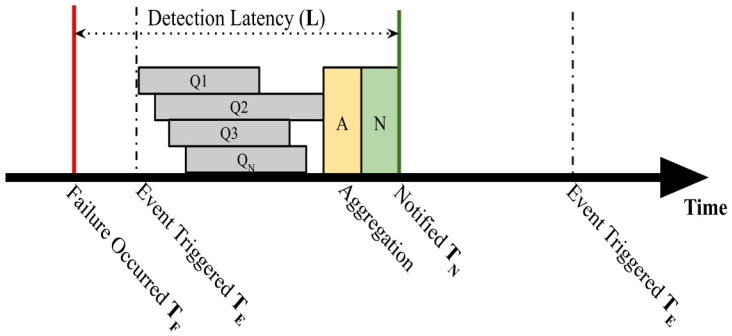
- **Property 7**. Time gap between $T[new]$ and $T[1]$ should be within t_{min} and t_{max} for k seconds before the join is confirmed.

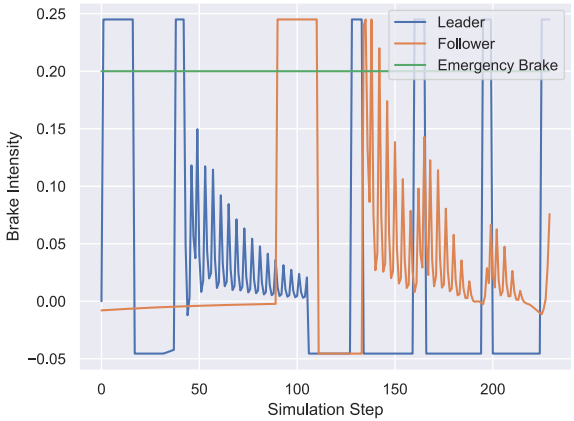
Properties for steady-state platooning:

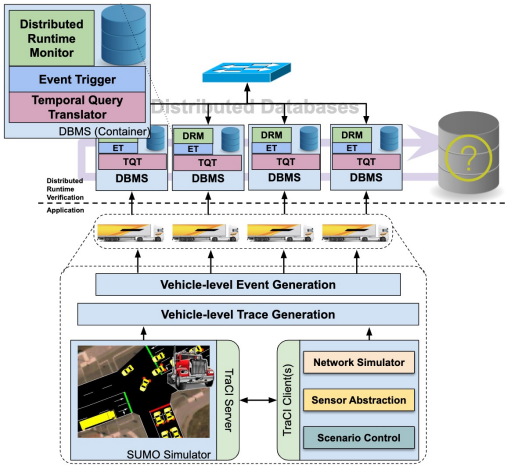
- **Property 1**. Time gaps between trucks should be within t_{min} and t_{max};
- **Property 2**. Platoon velocity v should be within v_{min} and v_{max};
- **Property 3**. Platoon acceleration a should be within a_{min} and a_{max};
- **Property 4**. Lateral offset difference between adjacent trucks should be within r_{min} and r_{max}.

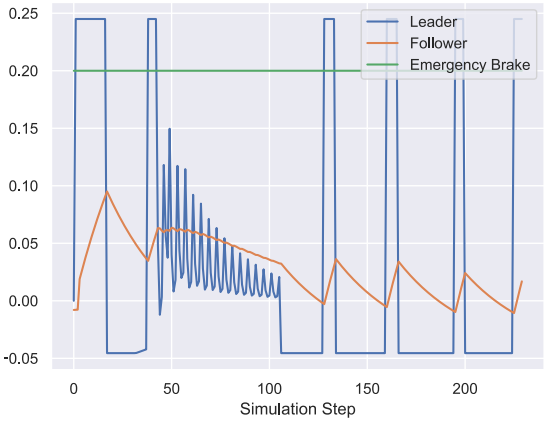
IPF 2.0: Distributed Runtime Verification

- Fast detection (avg <1ms, max ~10ms)
- Does not consider communication latency/reliability







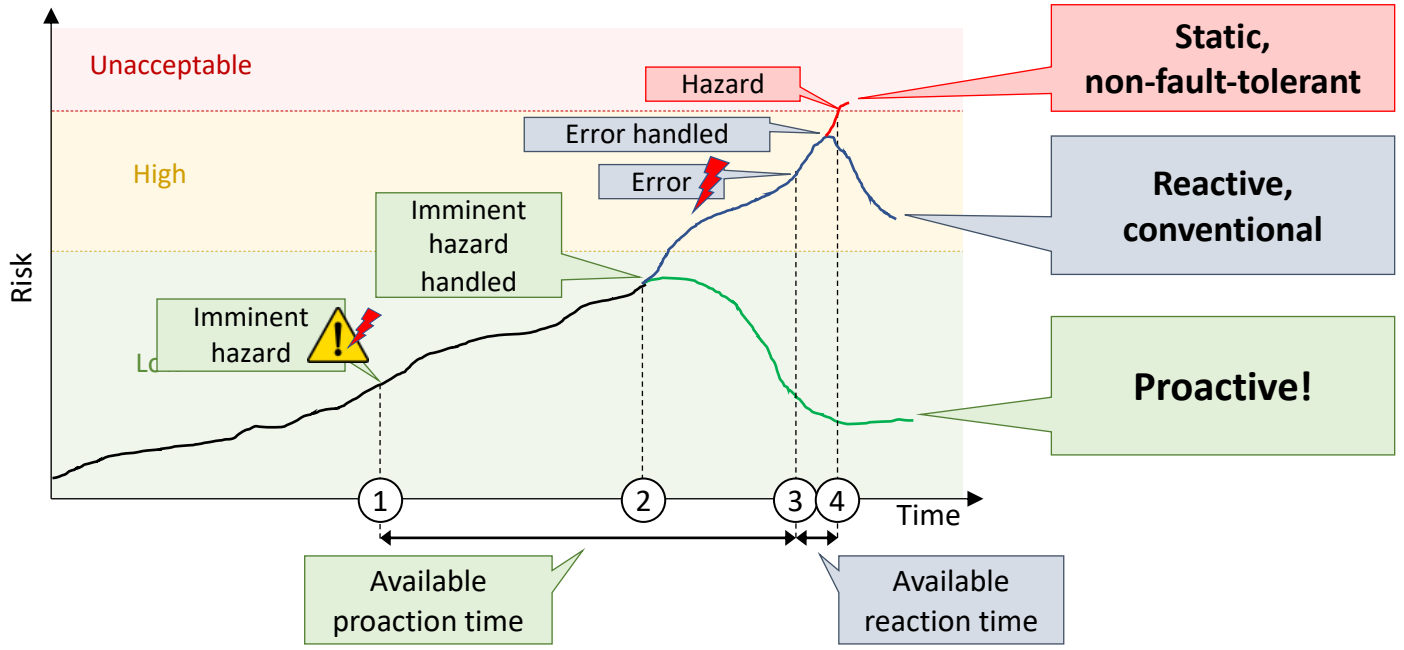


Significant improvement in brake intensity for follower vehicles

- Improves brake lifetime
- Improves regenerative braking energy

Can we do better?
Proactive self-diagnosis – imminent hazard

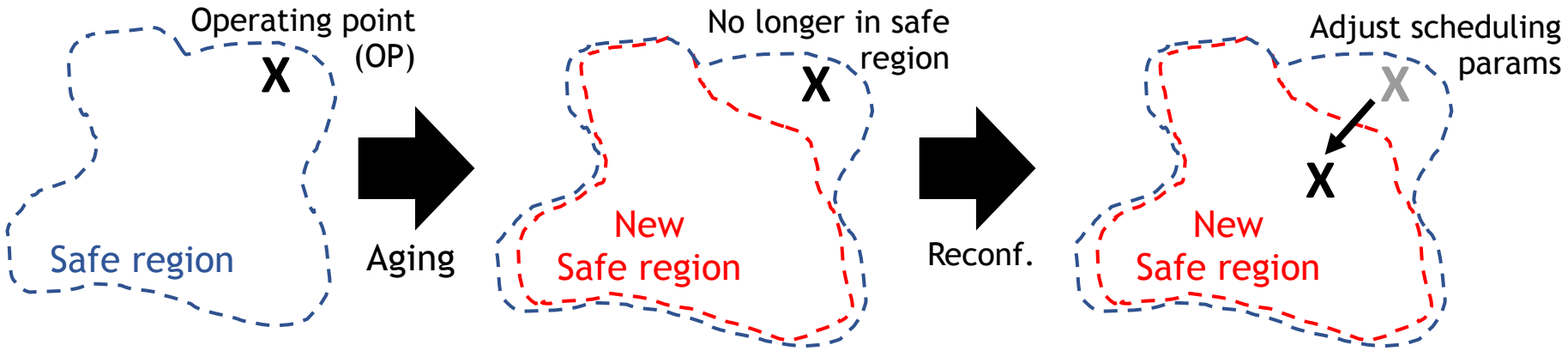
- Detect increased risk of errors **on time**
  - Focus on errors with unacceptable risk of **hazards** (critical functions only)
- Difficult to achieve with conventional, reactive error detection mechanisms
  - Reaction must finish **before** a hazard occurs and leads to a requirement violation



Intellectuals solve problems, geniuses prevent them
– Albert Einstein

Why is Proactive important?

- ❑ Enables continuous optimization & reconfiguration of the system (future work)
- ❑ Operation point(OP) results from optimization
- ❑ BoostIID can detect the fault + collect data for further runtime optimization



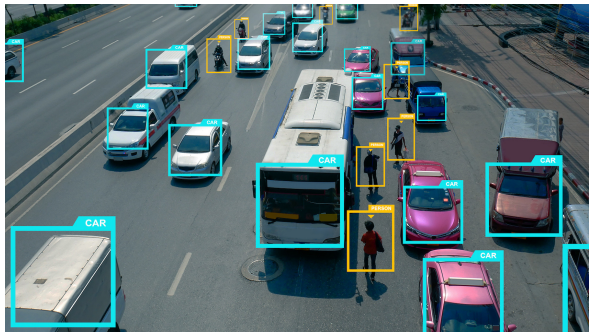
BoostIID: Fault-agnostic Online Detection of WCET Changes in Autonomous Driving

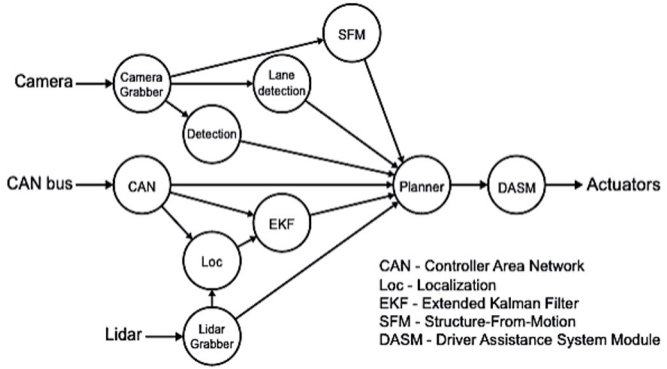
Saehanseul Yi[*], Nikil Dutt[*]

[*]University of California, Irvine, USA

Motivation

- **Autonomous Driving (AD)**
  - **Real-time (RT) Systems**: stringent timing guarantees → deterministic schedule
  - **Safety over *design-time* analyses:** Worst-Case Exec. Time (WCET)
  - **Tight margin**: demanding workloads in both computation and data volume
  - Small errors may lead to catastrophic results

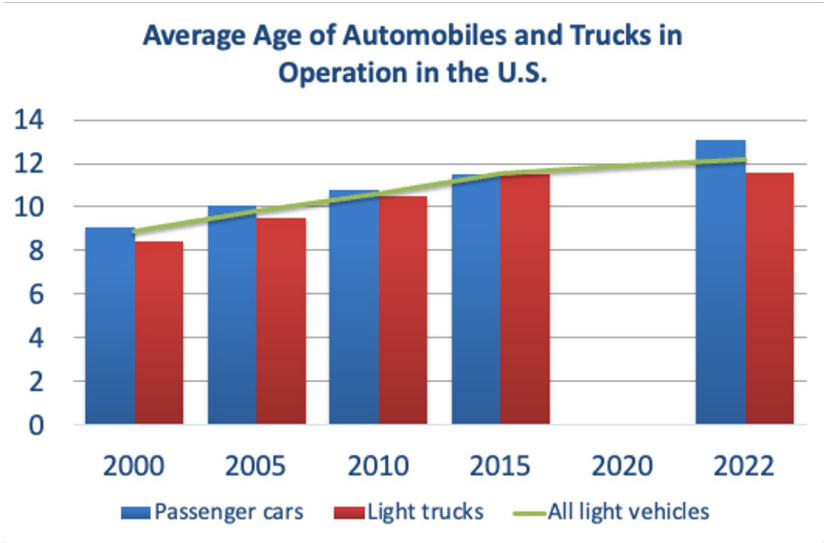




<Example AD Workloads>

Image courtesy of https://www.7wdata.be/digital-transformation/self-driving-cars-whos-winning-and-why

Motivation



- Passenger cars lasts 13.1 years
- Electric vehicles are expected to have longer lifespan than combustion engine vehicles
- ***Will WCETs stay the same during vehicle's lifetime?***

Chart from Bureau of Transportation Statistics 2022

Motivation: Aging of Computer Systems

- Various aging and permanent faults
  - **DRAM**: Bit Error Rate (BER) increases → re-execution
  - **Cache (SRAM)**: p_{fail} increases → re-partition
  - **SSD, NVMe**: cell wear-out → fail-slow
- Aging may unpredictably affect WCETs that were used in the system design
- Safety-critical systems must be safe when designed, and continue to be safe as conditions change

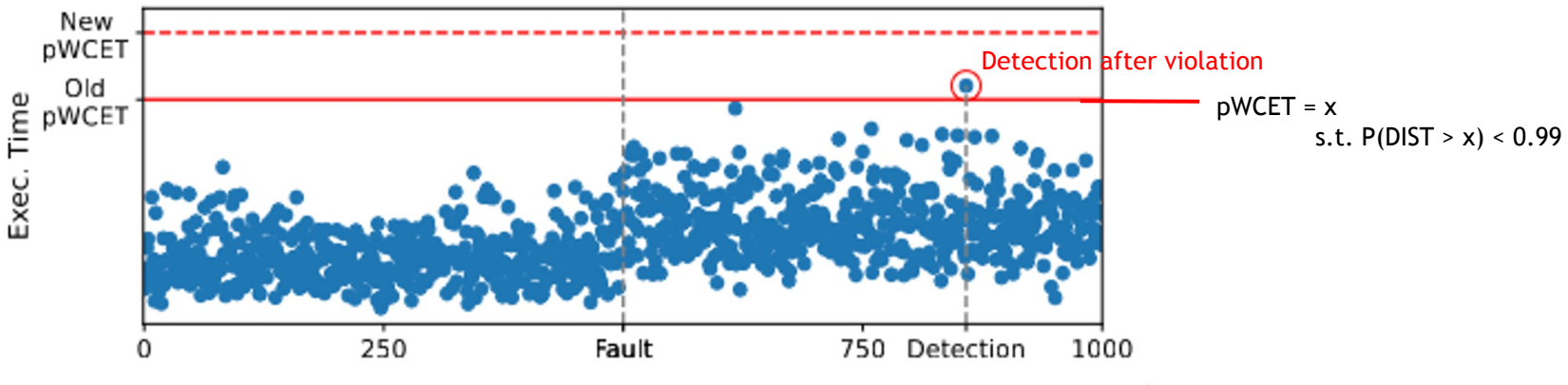
First step: how to proactively detects these safety threats?

Related Work

- **Design-time fault tolerance [… citation]**
  - Add *safety margin* to WCET
  - (-) Fault-specific analyses to measure potential impact
  - (-) Safety margin incurs energy inefficiency during normal operation
  - (+) Does not require recovery actions

- **Run-time fault tolerance [… citations]**
  - Control-Flow Checking (CFC): monitors a critical region of a program
  - (-) Heavy overhead; typically requires hardware support
  - (-) Detects after actual violation
  - (+) Does not require safety margin

Our Idea



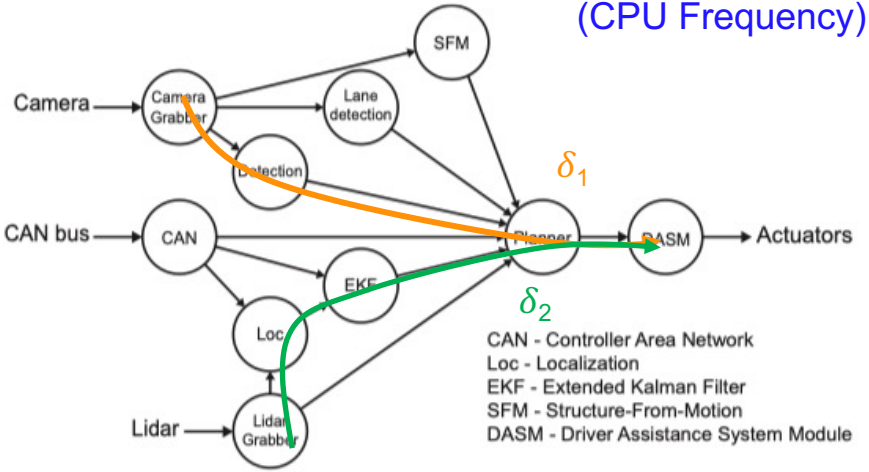
<Example fault detection scenario>

- View faults as a statistical distribution change of execution times
- ***(+) Proactively* detect distribution change using *independent & identical distribution (i.i.d) test***
- (+) No need for fault-specific analyses
- (+) Less pessimistic WCET estimation

- ❑ (+) Run-time method with low overhead
- ❑ (-) Does not recover from fault

Background: Example RT System Design

- Non-blocking periodic tasks
  - Async buffer between tasks
  - Decision variables: p_i : task period
    s_i : per-task speed factor (CPU Frequency)



A DAG of tasks from Bosch WATERS Industrial Challenge 2019

- Constraints (for each mode)
  - **Schedulability** (assuming EDF)

    Can we schedule tasks without violating periods?

$$U = \sum_i \frac{e_i}{p_i s_i} \leq 100\%$$

e_i: WCET at s_i=1
U : system utilization

  - **End-to-end deadline**

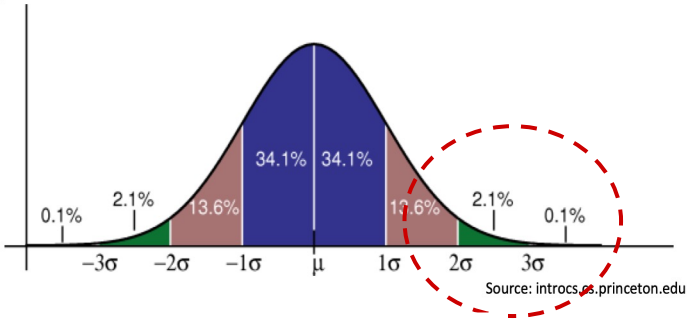
    Timing constraint from sensors to actuators

$$\sum_{i \in \delta} 2p_i \approx \sum_{i \in \delta} (p_i + r_i) \leq d$$

d : end-to-end deadline
r_i: worst-case delay

Background: Probabilistic WCET (pWCET)

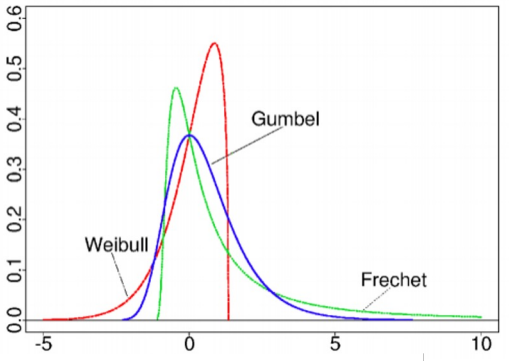
- Central Limit Theorem (CLT): good for estimating mean (center)
- **Extreme Value Theory (EVT)**: extreme value (worst case) distribution follows one of the three forms: *Gumbel*, the *Weibull*, or the *Frechet*



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

μ: mean
σ: std dev

<Normal Dist.>

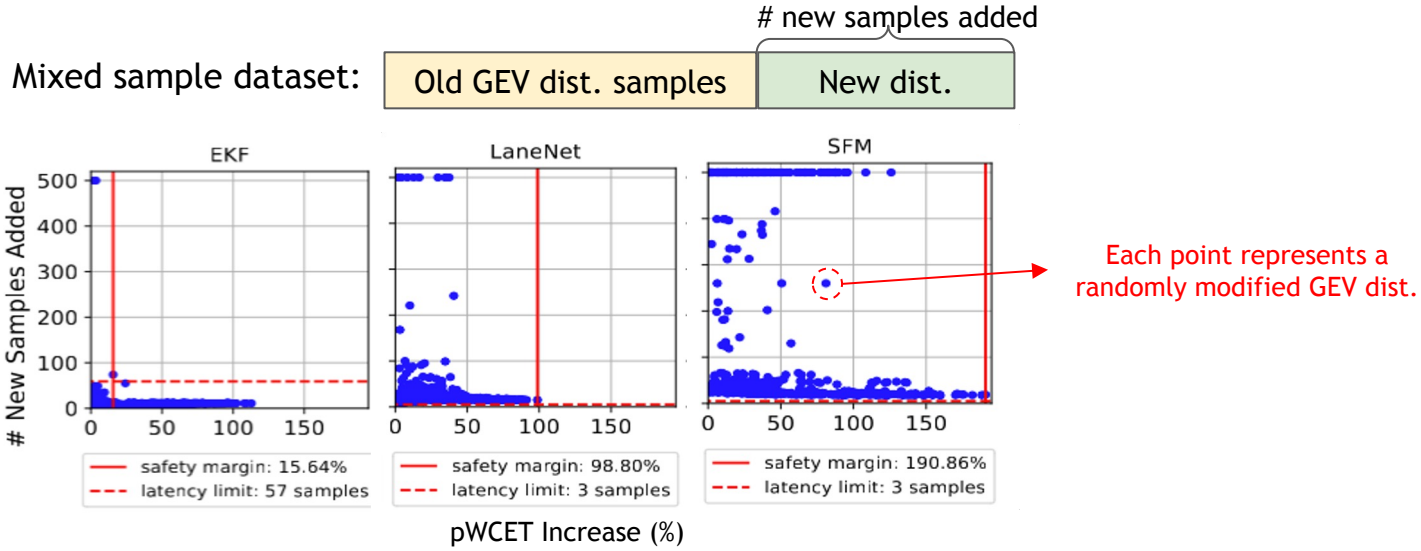


$$G(x) = \begin{cases} e^{-e^{\frac{x-\mu}{\sigma}}} & \xi = 0 \\ e^{-[1+\xi(\frac{x-\mu}{\sigma})]^{-1/\xi}} & \xi \neq 0. \end{cases}$$

μ: location
σ: scale
ξ: **shape**

<Generalized EV (GEV) Dist.>

Detection with a single i.i.d test



- KPSS i.i.d test as a boolean classifier
  - Are the samples from the same distribution? *True*/*False*
- Latency limit (dashed horizontal): within *k* new samples, the WCET change is detected (configurable)
- Safety margin (solid vertical): detector's blind spot; incorporate it to WCET
- KPSS performs well for some tasks

Detection with other i.i.d tests (Sensitivity)

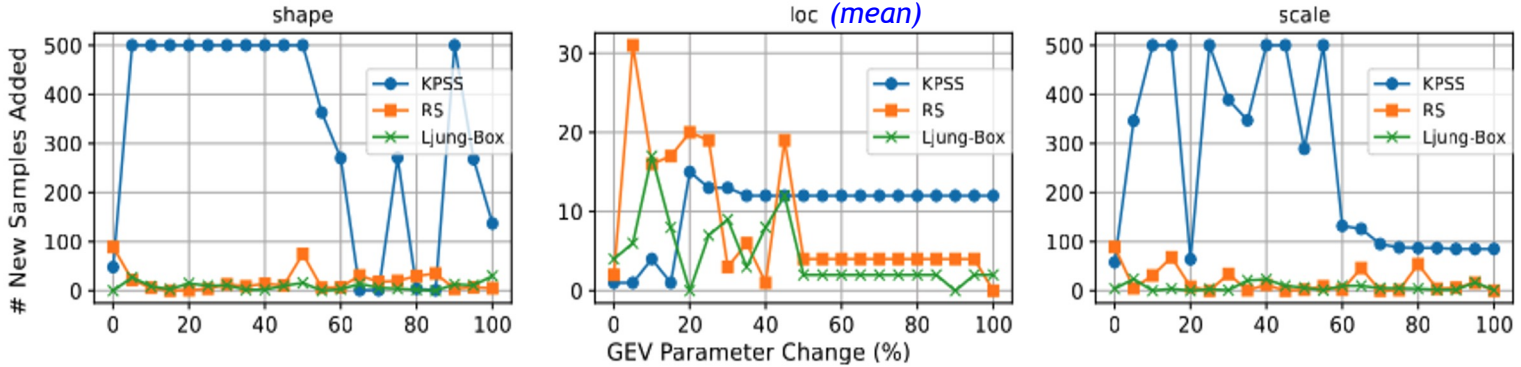
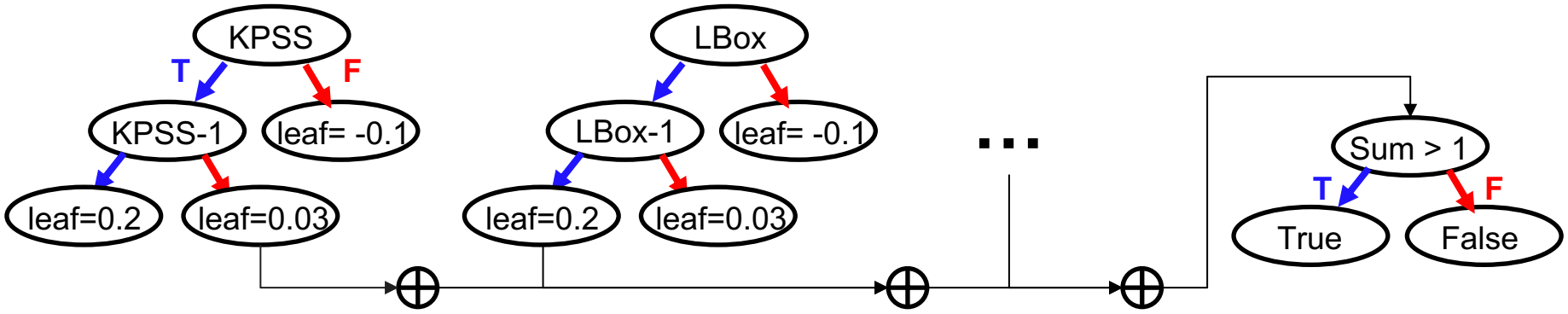


Fig. 4. Characterization of i.i.d tests with GEV parameter sensitivity using Darknet

- Three i.i.d tests: KPSS, R/S, and Ljung-Box
- Different i.i.d tests respond differently to each GEV parameter change
- Difficult to define situations to prioritize a certain i.i.d test
- Accuracy is fluctuating

Boosting



<Boosting example>

- Treat each i.i.d test as a weak classifier
- **eXtreme Gradient Boosting (XGBoost)**: combine multiple *weak classifiers* with weights to create a *strong predictive model*
- XGBoost input: 3 i.i.d tests + their history (previous 5 predictions)

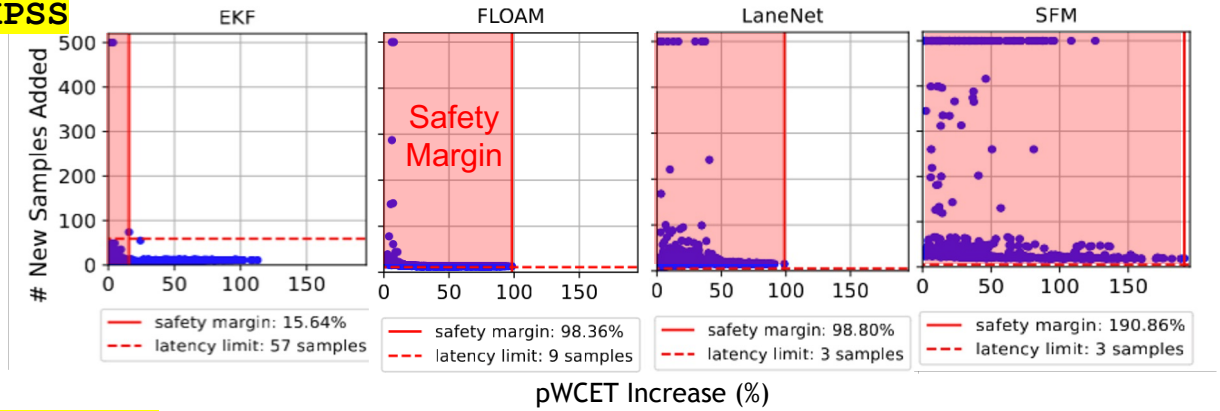
Experimental Setup

- NVIDIA Jetson TX2 platform
  - 6 task implementation from *Chauffeur*: autonomous driving benchmark suite
    EKF, Hybrid A*, FLOAM, LaneNet, Darknet, and SFM
  - i.i.d test implementation from *Chronovise*: a C++ framework for MBPTA
- pWCET
  - pWCET estimation using *Chronovise* with a prob. 10^{-4} & cross-checked with MATLAB's *gevcdf*
  - pWCET for KPSS / RS/ L-Box: 1.16 ms / 0.63 ms / 1.03 ms
- GEV parameter estimation
  - Maximum Likelihood Estimation (MLE) from *Chronovise* on 500 samples
- GEV parameter random modification
  - Each parameter 0~200%; uniform distribution
  - Generated by using MATLAB 2022b's *gevrnd*
- XGBoost python package with 100k detection dataset

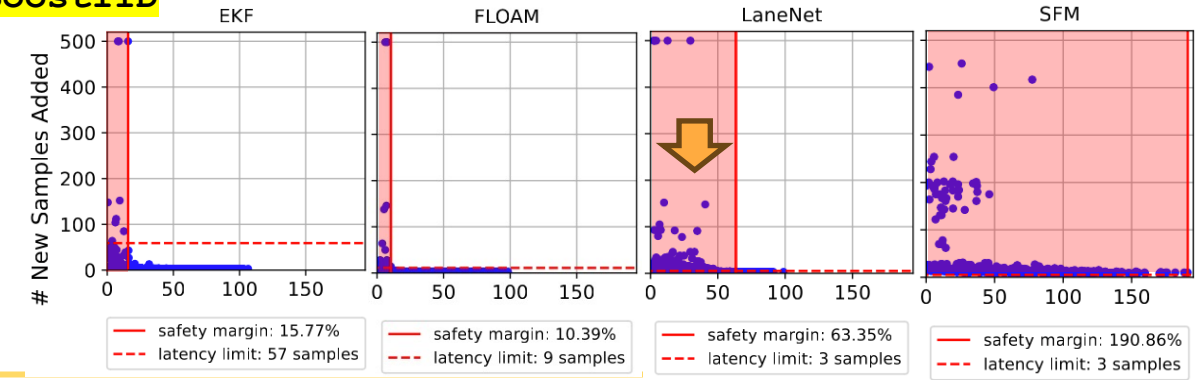
Measurement-Based
Probabilistic
Timing
Analysis

Experimental Results: Overview

KPSS

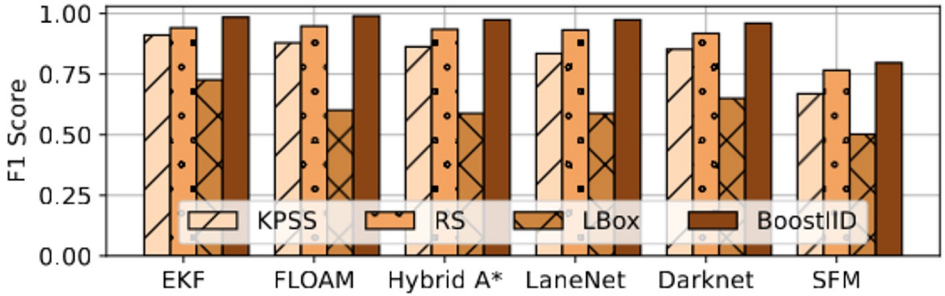


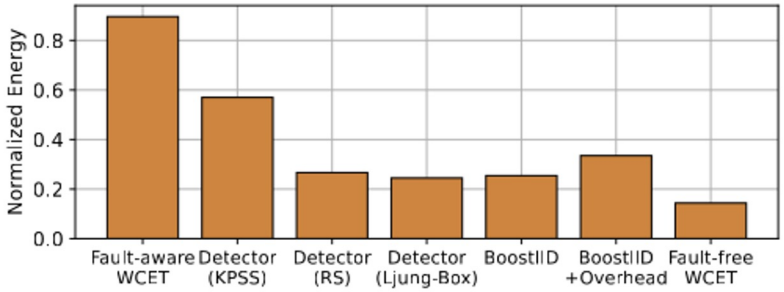
BoostIID



- Detection latency ↓ (# samples added)
  - → Safety margin ↓
  - → Energy efficiency ↑
- Huge improvement in FLAOM and LaneNet
- Poor results on EKF and SFM
  - EKF: already good enough with single i.i.d test
  - SFM: distribution is too difficult for the current set of i.i.d tests.

Experimental Results: F1 Score & Energy Efficiency





$$F_1 = \frac{2TP}{2TP + FP + FN}$$

False positive

False negative

- Except for SFM, the F1 score ranges from 0.96 to 0.99 on 100k datasets
- 62.6% energy reduction compared to classical Fault-aware WCET technique (=100% safety margin)
- BoostIID is fault-agnostic and improves energy efficiency by reducing the safety margin
- But cannot tolerate faults as the Fault-aware approach does

Conclusion

- Novel usage of i.i.d test for runtime detection of execution time change
- BoostIID alleviates pessimistic safety margin in WCET with previous fault-aware WCET methods
- As a result, BoostIID achieved 62.6% average dynamic power reduction in an example RT system with Autonomous Driving workloads
- Proactiveness of our method provides time for further runtime reconfiguration
- In the future, we plan to extend our work to recover from faults after detection using BoostIID

Key Takeaways

- (lack of) Trust is not just about AI
  - The Unknown knowns:
    - Hardware failure
    - Software bugs,
    - Security
    - Over/under specifications
    - … and AI un{predictability, explainability, bounded behavior}
  - The Unknown Unknowns
    - Emergent Behavior
    - Unforeseen scenarios
    - Zero day attacks
- Need to make systems self-aware, and ….Trust, but verify …

Questions?

Acknowledgements:

- Dutt Research Group (DRG)
- Fadi Kurdahi/ UCI
- Bryan Donyanavard/ SDSU
- Andreas Herkersdorf / TU Munich
- Rolf Ernst/ TU Braunschweig
- Jason Jong-Chan Kim, Kookmin Univ
- Axel Jantsch/ TUWien

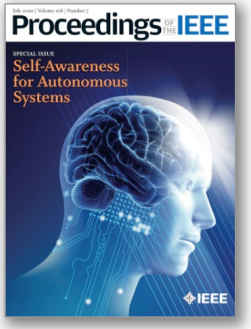
Research Support:

- National Science Foundation (NSF)
- DFG
- Facebook (Meta)
- Samsung

Adverts…..

ACM TCPS Special Issue on Self-Awareness in CPS
Guest Editors: *Axel Jantsch, Nikil Dutt, Peter Lewis*
August 2020

Proceedings of the IEEE Special Issue on Self-Awareness for Autonomous Systems
Guest Editors: *Nikil Dutt, Carlo Regazzoni, Bernhard Rinner, Xin Yao*
July 2020



Self-Awareness for Autonomous Systems

Volume 108, Issue 7 | July 2020

- Guest Editors
- Special Issue Papers

Guest Editors:

Nikil Dutt
Carlo S. Regazzoni
Bernhard Rinner
Xin Yao

Publications

Architectural-level

ESL 2020

Self-adaptive memory approximation: A formal control theory approach

Biswadip Maity, Bryan Donyanavard, Anmol Surhonne, Amir Rahmani, Andreas Herkersdorf, and Nikil Dutt. "**SEAMS: Self-Optimizing Runtime Manager for Approximate Memory Hierarchies**," in ACM Transactions on Embedded Computing Systems 20, 5, Article 48 (July 2021), 26 pages. DOI: https://doi.org/10.1145/3466875

TECS 2022

SEAMS: Self-Optimizing Runtime Manager for Approximate Memory Hierarchies

Biswadip Maity, Majid Shoushtari, Amir M. Rahmani and Nikil Dutt, "**Self-adaptive Memory Approximation: A Formal Control Theory Approach**," in IEEE Embedded Systems Letters, DOI: https://www.doi.org/10.1109/LES.2019.2941018.

Publications

System-level

ESWEEK 2019

HESSLE-FREE: Heterogeneous Systems Leveraging Fuzzy Control for Runtime Resource Management

Ongoing

Scalable deployment of MARS in datacenter racks for capacity crisis mitigation.
(joint work with Meta)

Kasra Moazzemi, Biswadip Maity, Saehanseul Yi, Amir M. Rahmani, and Nikil Dutt. 2019. **HESSLE-FREE: Heterogeneous Systems Leveraging Fuzzy Control for Runtime Resource Management**. ACM Transactions on Embedded Computing Systems (TECS), Article 74 (October 2019), 18.5s 19 pages. DOI:https://doi.org/10.1145/3358203

E. A. Rambo, Bryan Donyanavard, Minjun Seo, Florian Maurer, Thawra M. Kadeed, Caio Batista De Melo, Biswadip Maity, Anmol Surhonne, Andreas Herkersdorf, Fadi Kurdahi, Nikil D Dutt, Rolf Ernst, "**The Self-Aware Information Processing Factory Paradigm for Mixed-Critical Multiprocessing**," in IEEE Transactions on Emerging Topics in Computing, DOI: https://www.doi.org/10.1109/TETC.2020.3011663.

Tiago Mück, Bryan Donyanavard, Biswadip Maity, Kasra Moazzemi, and Nikil Dutt. "**MARS: Middleware for Adaptive Reflective Computer Systems**," in arXiv:2107.11417 [cs.DC]

Publications

Application-level

IESS 2019

Workload characterization for memory management in emerging embedded platforms

D. Seo, B. Maity, P. Chen, D. Yun, B. Donyanavard, N. Dutt. "**ProSwap: Period-aware Proactive Swapping to Maximize Embedded Application Performance**"
to appear in 16th International Conference on Networking, Architecture, and Storage (NAS 2022).

ESWEEK 2021

Chauffeur: Benchmark Suite for Design and End-to-End Analysis of Self-Driving Vehicles on Embedded Systems.

Biswadip Maity, Saehanseul Yi, Dongjoo Seo, Leming Cheng, Sung-Soo Lim, Jong-Chan Kim, Bryan Donyanavard, and Nikil Dutt. 2021. **"Chauffeur: Benchmark Suite for Design and End-to-End Analysis of Self-Driving Vehicles on Embedded Systems"**. ACM Transactions on Embedded Computing Systems 20, 5s, Article 74 (October 2021), 22 pages. DOI:https://doi.org/10.1145/3477005

NAS 2022

ProSwap: Period-aware Proactive Swapping to Maximize Embedded Application Performance

B. Maity, B. Donyanavard and N. Dutt. "**Self-aware Memory Management for Emerging Energy-efficient Architectures**," in 11th International Green and Sustainable Computing Workshops (IGSC), 2020, pp. 1-8, doi: 10.1109/IGSC51522.2020.9291086.

Publications – IPF Project Related

M. Ji, S. Yi, C. Koo, S. Ahn, D. Seo, N. Dutt, and J. Kim, “Demand Layering for Real-Time DNN Inference with Minimized Memory Usage,” Proceedings of the 26th IEEE International Real-Time Systems Symposium (RTSS 2022), December 5-8, 2022, to appear.

S. Yi, T. -W. Kim, J. -C. Kim and N. Dutt, "Energy-Efficient Adaptive System Reconfiguration for Dynamic Deadlines in Autonomous Driving," 2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC), 2021, pp. 96-104, doi: 10.1109/ISORC52013.2021.00023.

W. Jang, H. Jeong, K. Kang, N. Dutt and J. -C. Kim, "R-TOD: Real-Time Object Detector with Minimized End-to-End Delay for Autonomous Driving," 2020 IEEE Real-Time Systems Symposium (RTSS), 2020, pp. 191-204, doi: 10.1109/RTSS49844.2020.00027

A. Miele, A. Kanduri, K. Moazzemi, D. Juhász, A. Rahmani, N. Dutt, P. Liljeberg and A. Jantsch (2019), “**On-Chip Dynamic Resource Management**”, Foundations and Trends in Electronic Design Automation: Vol. 13, No. 1-2, pp 1–144. DOI: 10.1561/1000000055.

Minjun Seo and Fadi Kurdahi. **Efficient Tracing Methodology Using Automata Processor**. *ACM Transactions on Embedded Computing Systems (TECS)*, Article 80, (October 2019), 18.5s: 1-18. DOI:https://doi.org/10.1145/3358200

Tianyi Zhang, Minjun Seo, Bryan Donyanavard, Nikil Dutt, and Fadi J. Kurdahi, "**Predicting Failures in Embedded Systems using Long Short-Term Inference**," in IEEE Embedded Systems Letters, DOI: https://www.doi.org/10.1109/LES.2020.3007361.

Publications – IPF Project Related

H. Hoffman, A. Jantsch, and N. Dutt, "**Embodied Self-Aware Computing Systems**," Proceedings of the IEEE, Special issue on Self-Awareness, Volume 108, Issue 7, July 2020. DOI: 10.1109/JPROC.2020.2977054

K. Bellman, N. Dutt, L. Esterle, A. Herkersdorf, A. Jantsch, C. Landauer, P. Lewis, M. Platzner, N. Taherinejad, and K. Tammemäe, "**Self-aware Cyber-Physical Systems**," ACM Transactions on Cyber-Physical Systems, Article 38, June 2020. DOI: https://doi.org/10.1145/3375716

F. Maurer, B. Donyanavard, A. Rahmani, N. Dutt, and A. Herkersdorf, "**Emergent Control of MPSoC Operation by a Hierarchical Supervisor / Reinforcement Learning Approach**," Proceedings of the 2020 Conference on Design, Automation and Test in Europe (DATE 2020), March 2020. DOI: 10.23919/DATE48585.2020.9116574

B. Donyanavard, T. Mück, A. M. Rahmani, N. Dutt, A. Sadighi, F. Maurer, and A. Herkersdorf. 2019. **SOSA: Self-Optimizing Learning with Self-Adaptive Control for Hierarchical System-on-Chip Management**. In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '52). Association for Computing Machinery, New York, NY, USA, 685–698. DOI:https://doi.org/10.1145/3352460.3358312

M. Seo and F. Kurdahi – **Efficient Tracing Methodology Using Automata Processor**. Proceedings of the International Symposium on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2019), New York, NY, October 2019. DOI: 10.1145/3358200