Toward predictable AI-enabled Real-Time Systems

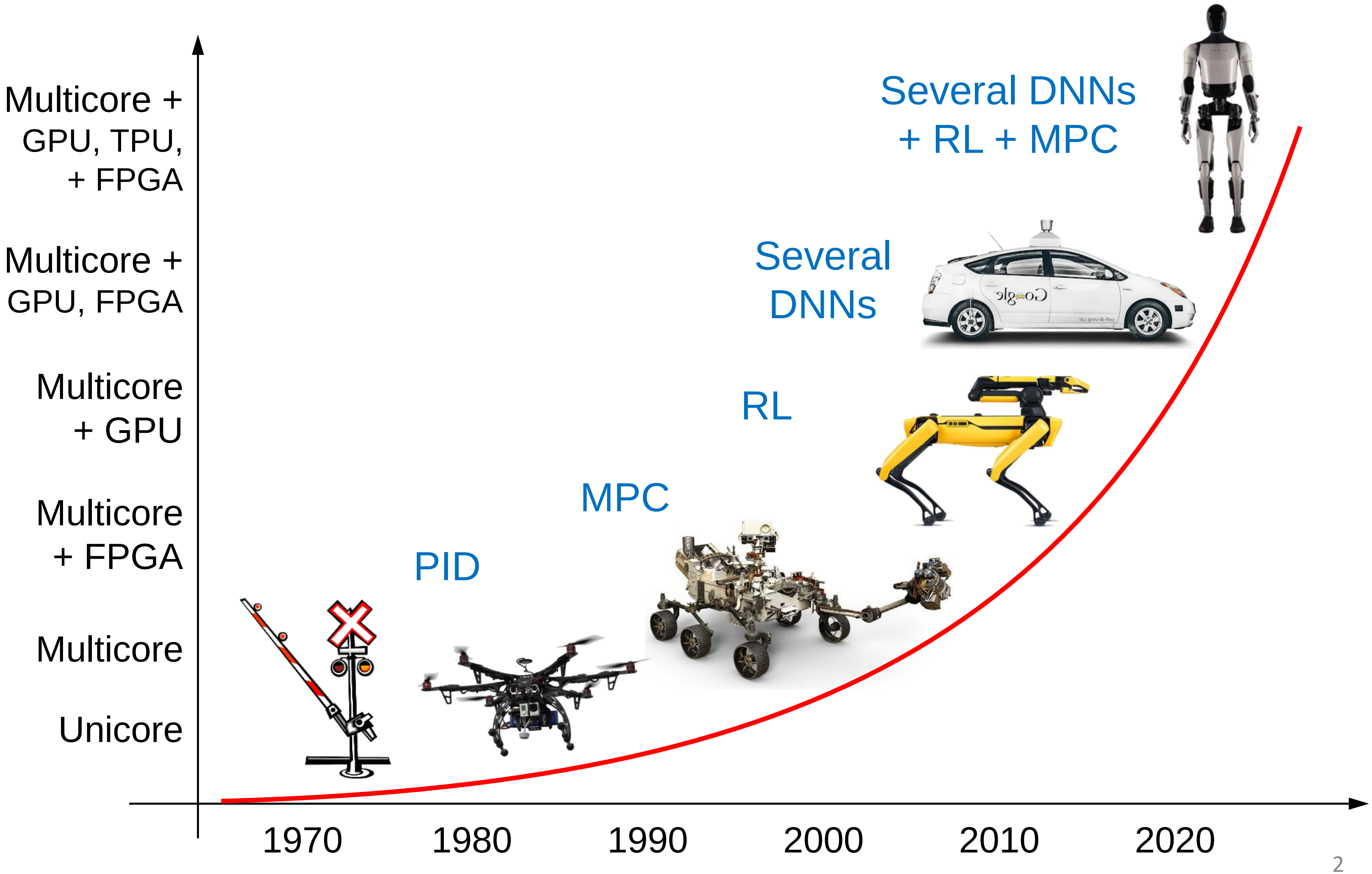
Giorgio Buttazzo

giorgio.buttazzo@santannapisa.it

Sant'Anna School of Advanced Studies

Real-Time Systems Laboratory

Increasing complexity



Retis
Real-Time Systems Laboratory

Features & Requirements

Typical features		Requirements
▪ Perceive complex scenes	⇨	AI & deep learning components
▪ Real-time performance	⇨	RTOS, efficient resource manag.
▪ Mixed criticality and req.	⇨	Hypervisors, component isolation
▪ Large code size	⇨	Security, Intrusion detection
▪ Safety-critical	⇨	Fault/anomaly detection
▪ Distributed	⇨	RT Cloud, RT middleware (DDS)

⇩

several challenges

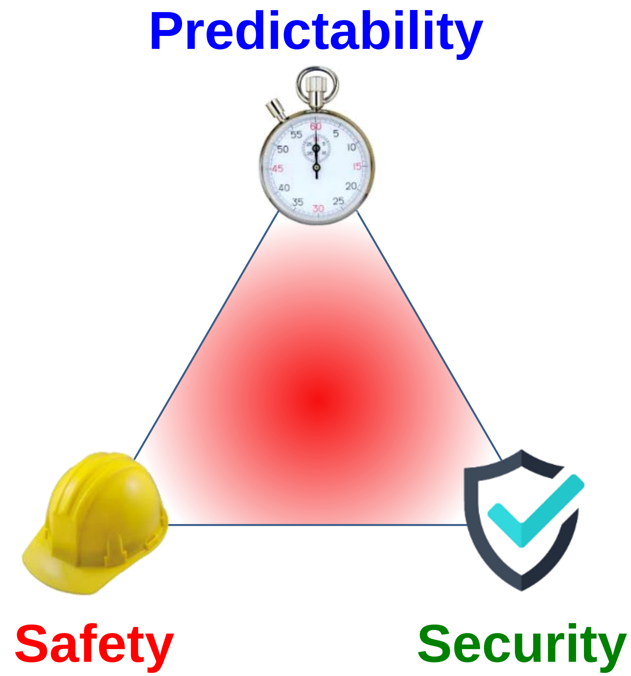
Major challenges

AI algorithms

Middleware support

Real-Time software support

Heterogeneous hardware



Major challenges

In this talk, I will focus on two main aspects, illustrating problems and potential solutions:

Real-Time Issues

- Mixed criticality
- Hypervisor
- Predictability
- Analysis
- Optimization

AI issues

- Main problems
- Acceleration
- Optimization
- Safety
- Security

Real-Time issues

Types of computations

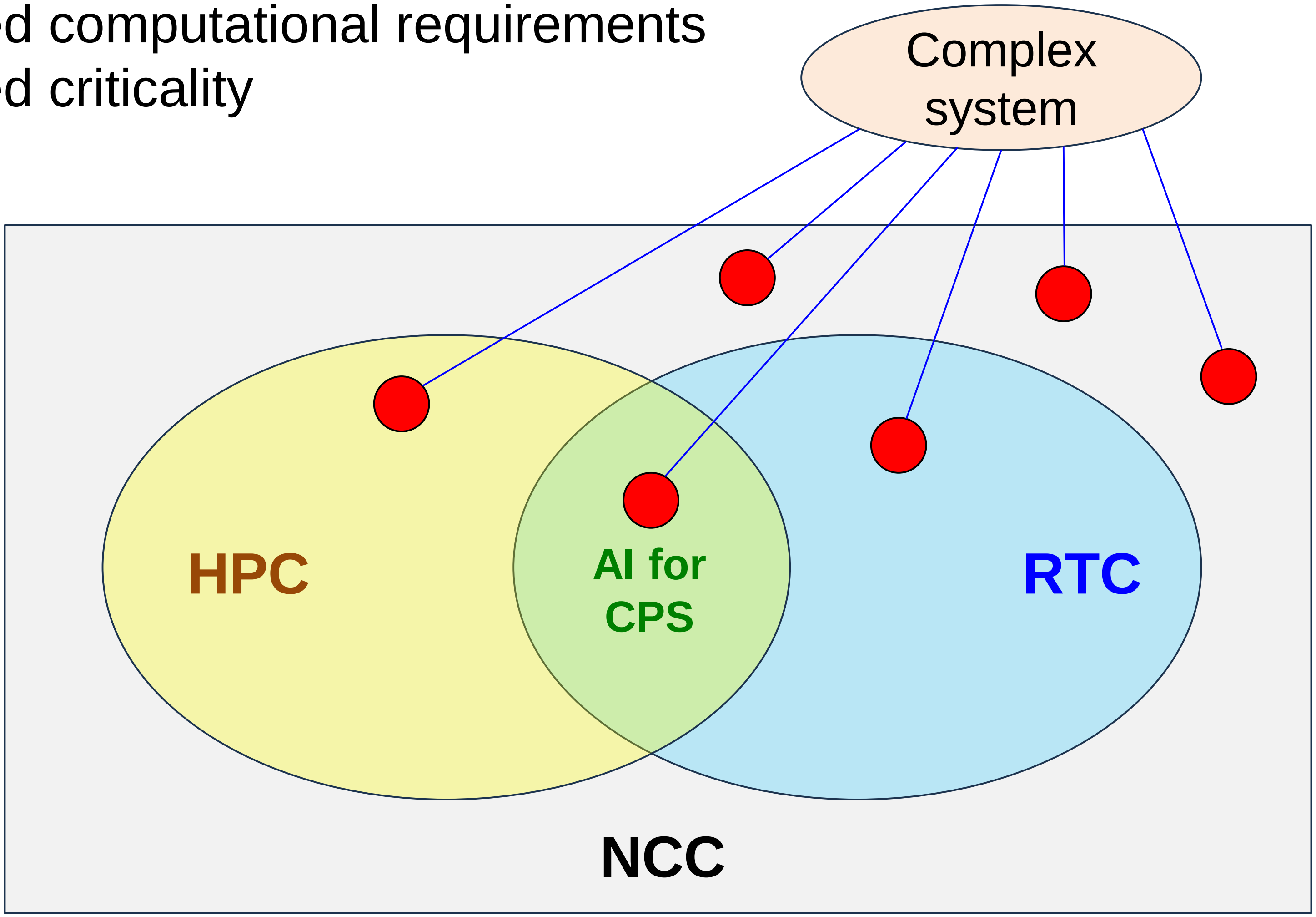
- High-Performance (**HPC**): Computationally intensive, a lot of memory
- Real-Time (**RTC**): Reactive, periodic, timing guarantees
- Non Critical (**NCC**): neither HP nor RT (functionally correct)

	HPC	**RTC**	**NCC**
Examples	train DNNs, simulate virtual worlds	visual tracking, ABS, robot control	comfort functions, user interface
Objective	run faster, reduce avg. response time	guarantee WCRT & bounded delays	correct functionality
SW support	Rich OS (Linux, QNX, VxWorks)	RTOS (FreeRTOS, Erika)	Rich OS (Linux, QNX, VxWorks)
HW support	parallel arch, GPUs, specialized HW	single core or multi core CPUs	single core or multi core CPUs

Types of computations

Complex systems normally require all types of software components:

Mixed computational requirements
Mixed criticality



Mixed criticality

Consider for example a self-driving car.

Not certifiable SW
Large attack surface

Perception, tracking, localization need to be managed by a **rich OS** to exploit device drivers, libraries, and AI development frameworks.

Steering, throttle modulation, braking, and engine control are highly critical and must be managed by a **certified RTOS**.

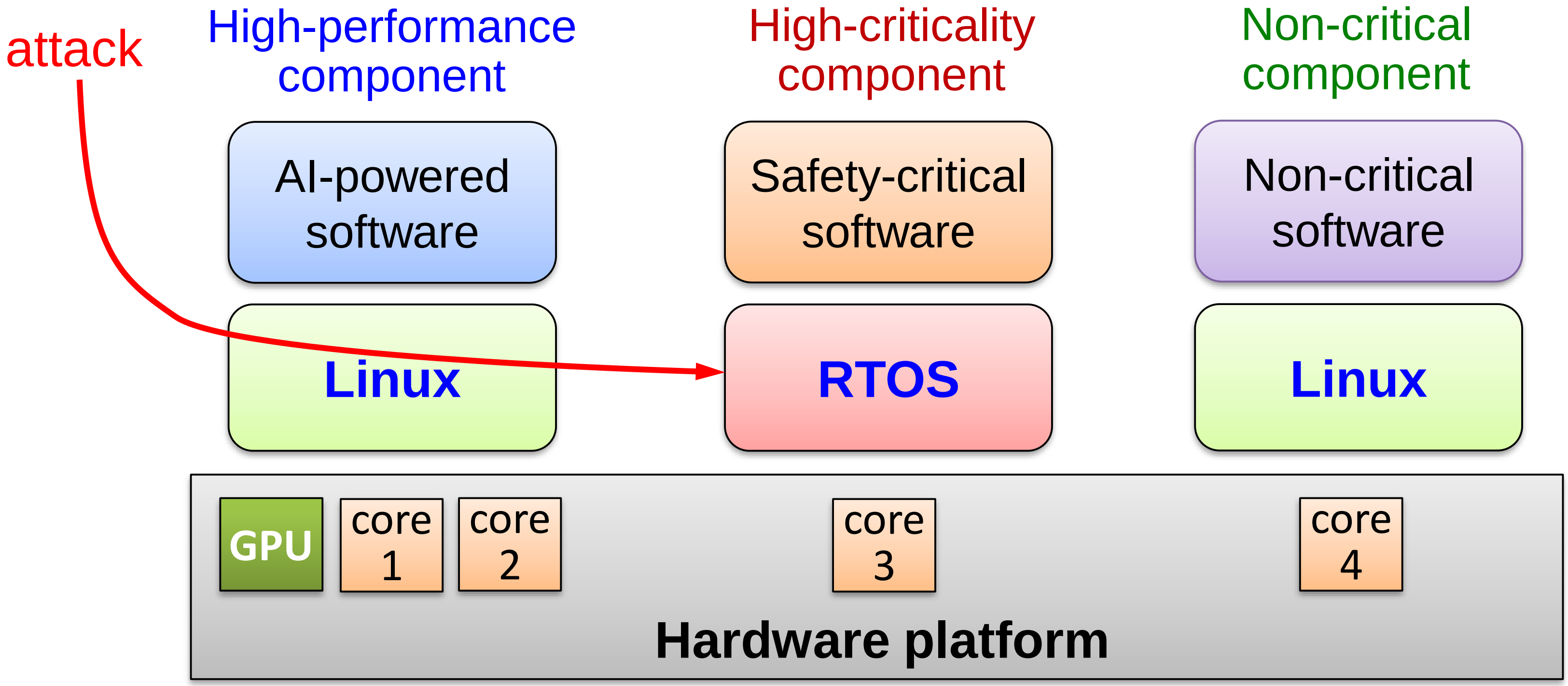
Rich OS

RTOS

Problems of mix-critical appl^s

Interference: low-critical tasks can delay highly-critical ones due to interference among share resources (memory, bus)

Security: an attack to a component can propagate to others

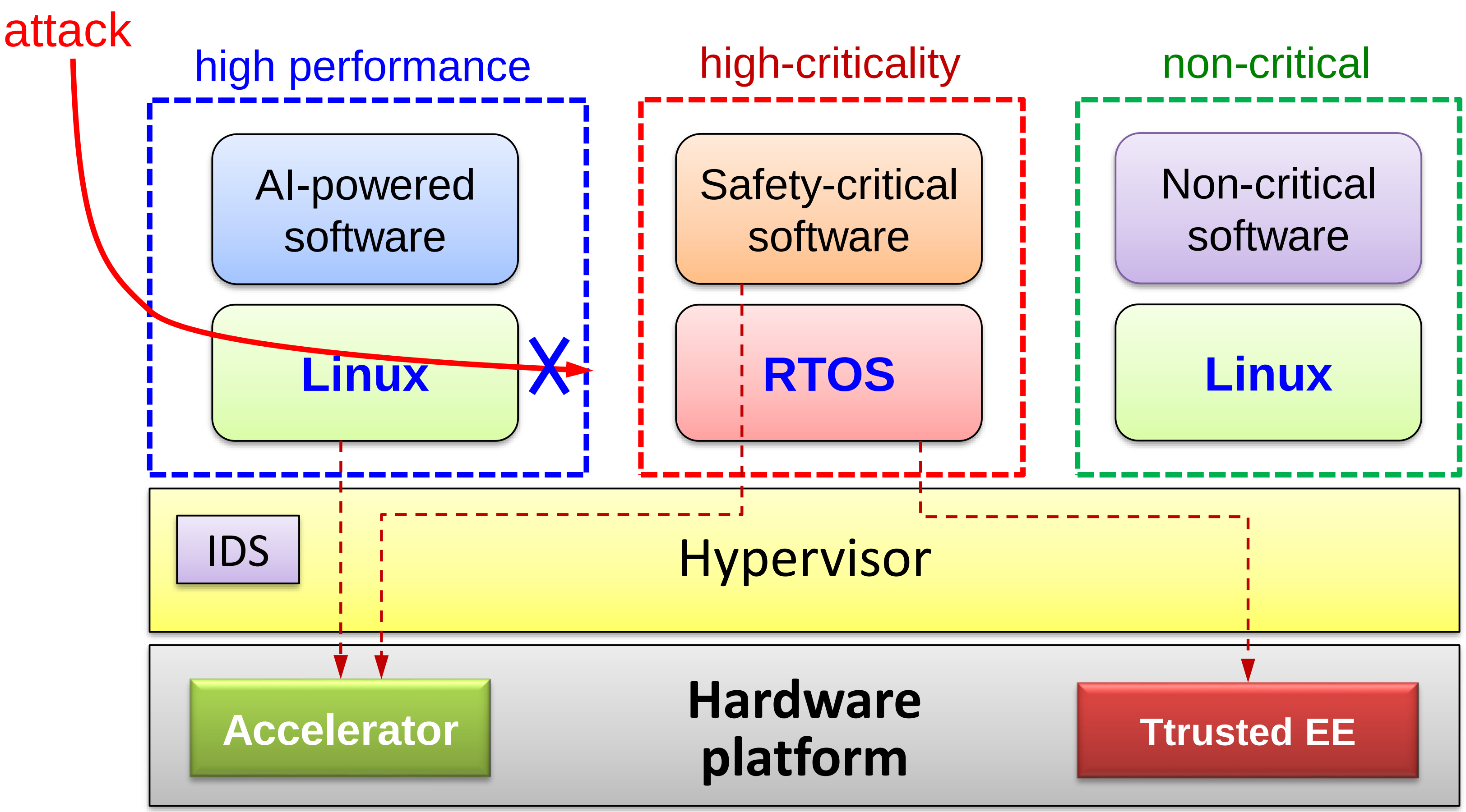


Security is a serious issue

In 2015, a **Jeep Cherokee** was remotely attacked by exploiting a vulnerability of the infotainment system. The hackers gained control of the car, including steering, braking, turning on the wipers, blasting the radio, and stopping the engine.

Achieving mix-criticality

A safe solution is to **isolate** the different software components by a **Type 1 bare-metal hypervisor** with **security** and **real-time** features:



Hypervisor features

1. **Strong temporal & spatial isolation** among execution domains by secure cache partitioning, CPU/memory reservations & virtualization

2. **Hard real-time scheduling** of execution domains

3. **I/O virtualization** to efficiently share resources among domains

4. **Deterministic inter-domain communication**: zero-copy & wait-free shared-memory paradigms, cyclic async buffers, bounded latency …

5. **Security mechanisms** against denial-of-service and side-channel attacks, run-time security monitoring, address space layout randomization, control flow Integrity, ISO 21434 qualification, …

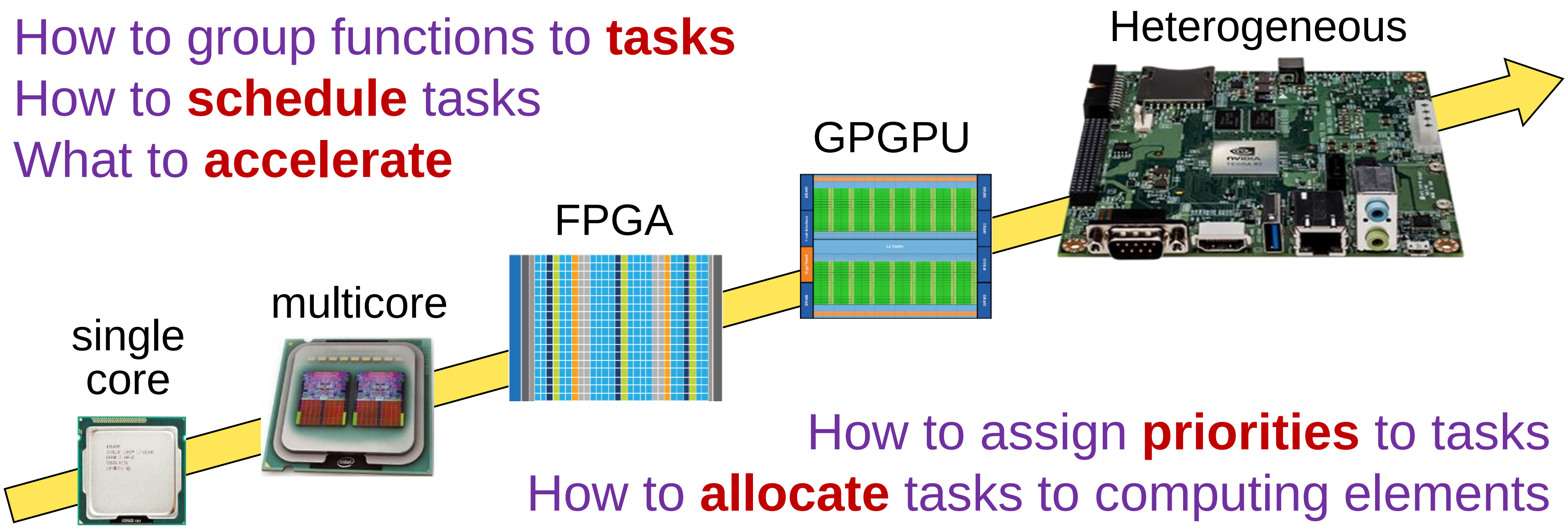
6. **Safety**: totally static, MISRA compliance, ISO 26262 qualification, VM-level health-monitoring, …

https://accelerat.eu/clare/

Optimizing RT software

With the growing complexity of computing platforms, optimizing software became quite challenging!

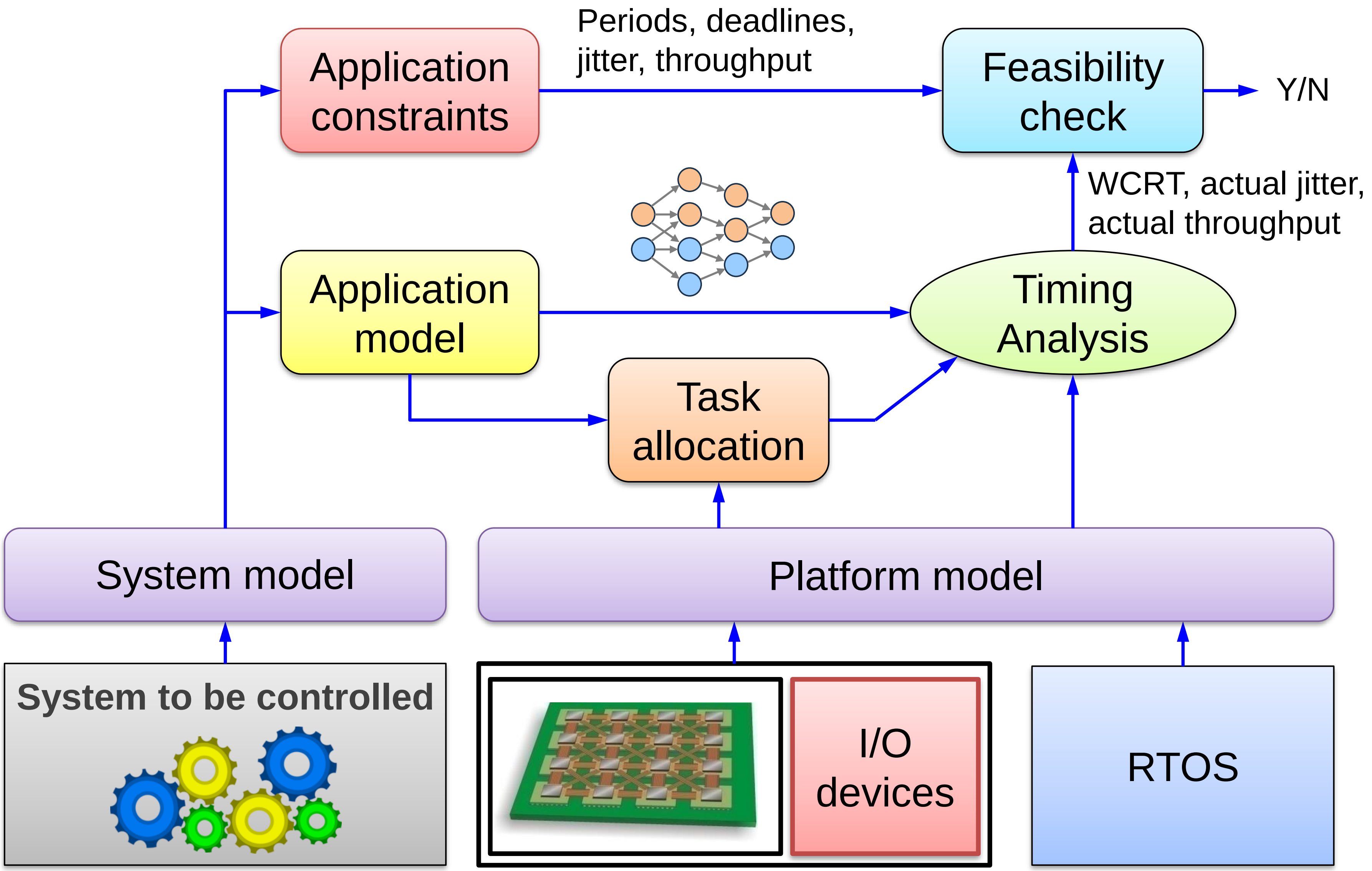
How to group functions to **tasks**
How to **schedule** tasks
What to **accelerate**



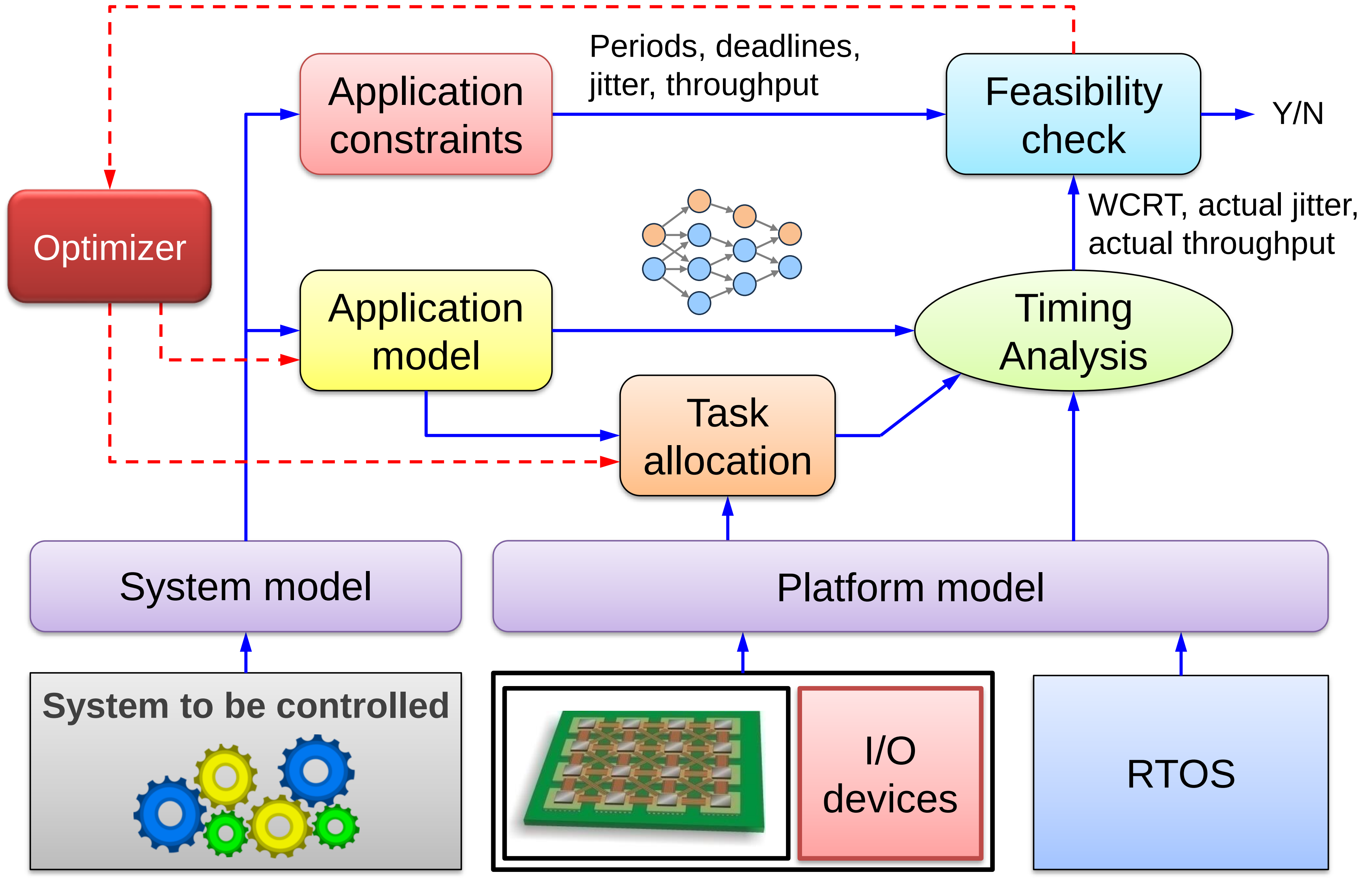
How to assign **priorities** to tasks
How to **allocate** tasks to computing elements
How to **synchronize** the access to shared resources

Such an **optimization process** requires a **precise timing analysis** to predict the response times of various interacting SW tasks.

Timing analysis

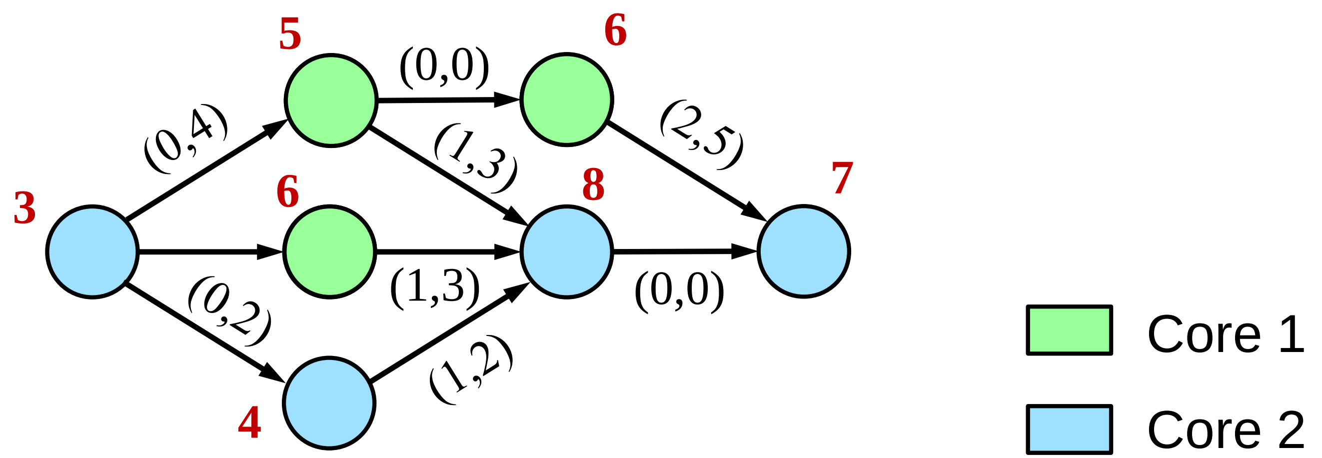


Optimization



Model and Analysis

Thus, the application is modeled as a **directed acyclic graph** (**DAG**) where each node has a WCET and each edge has a (min, max) delay range:

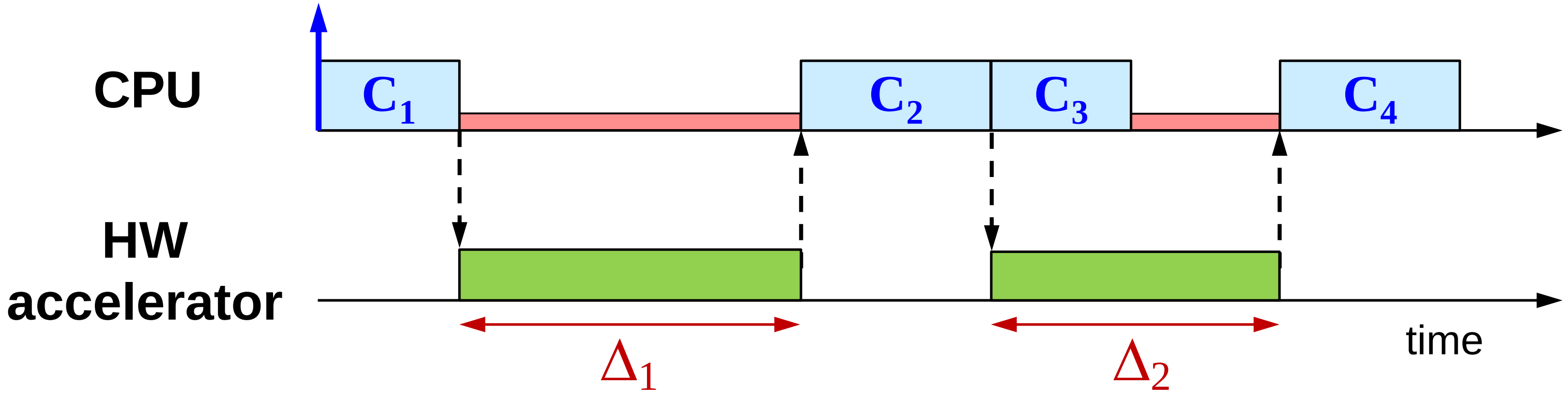


In addition, each node can be manually allocated to a different core or the **best allocation** is automatically found by optimization.

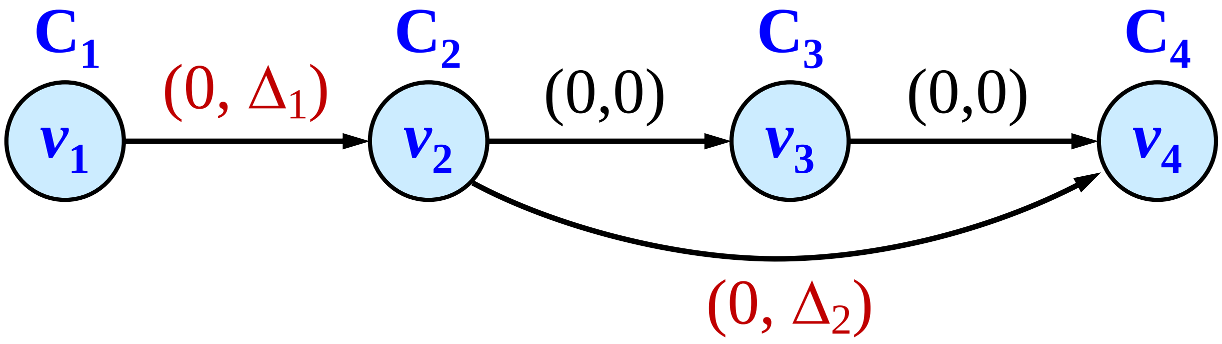
Reference paper

F. Aromolo, A. Biondi, G. Nelissen, and G. Buttazzo, "Event-Driven Delay-Induced Tasks: Model, Analysis, and Applications", Proc. of the IEEE RTAS 2021.

Application model



Application model



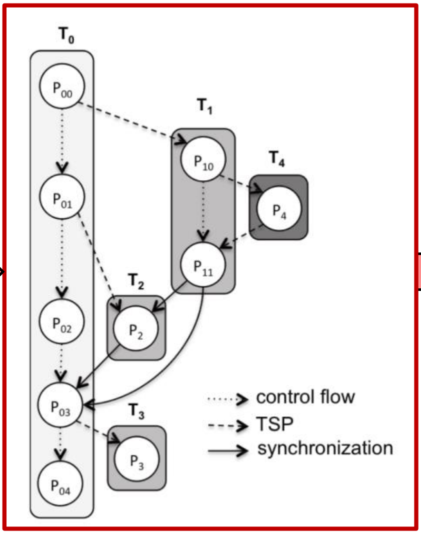
From code to analysis

DAG and analysis can directly be derived from the application code (e.g., OpenMP parallel code):

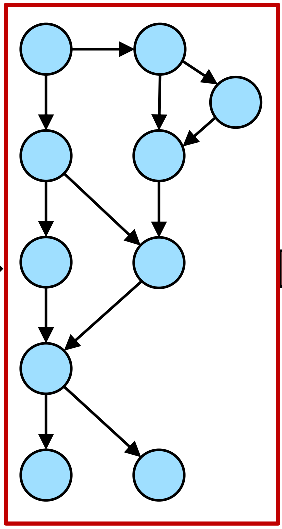
Program code

```
#pragma omp parallel num_threads(10) {
#pragma omp master {
#pragma omp task { // T0
   part00
   #pragma omp task depend(out:x) // T1
                    final(true)
   {
      part10
      #pragma omp task { part4 } // T4
      part11
   }
   part01
   #pragma omp task depend(in:x) // T2
   { part2 }
   part02
   #pragma omp taskwait
   part03
   #pragma omp task { part3 } // T3
   part04
}}}
```

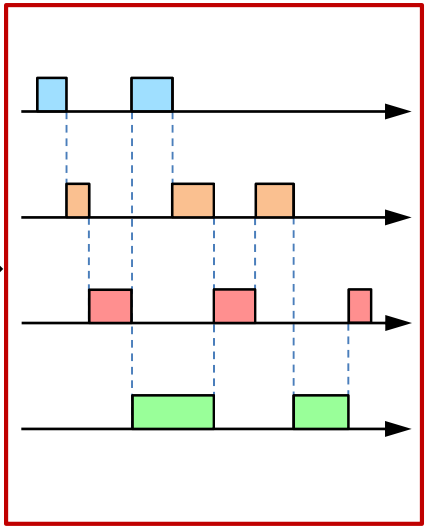
Code structure



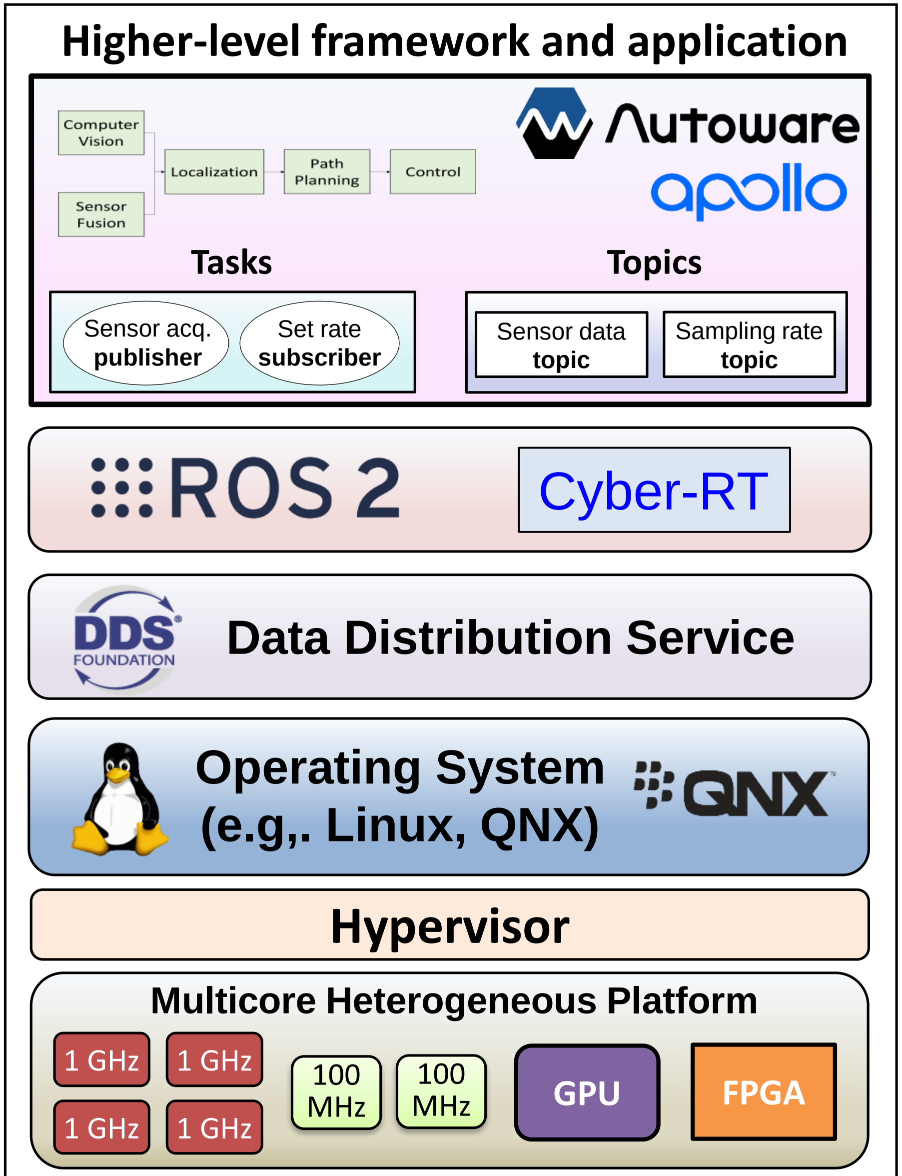
DAG model



Timing analysis



DDS-enabled RT systems



Often, applications needs to deal with multiple levels of scheduling:

- Deep learning frameworks (TensorFlow, Pythorch)
- Communication middleware (ROS 2, DDS)
- Operating System
- Hypervisor

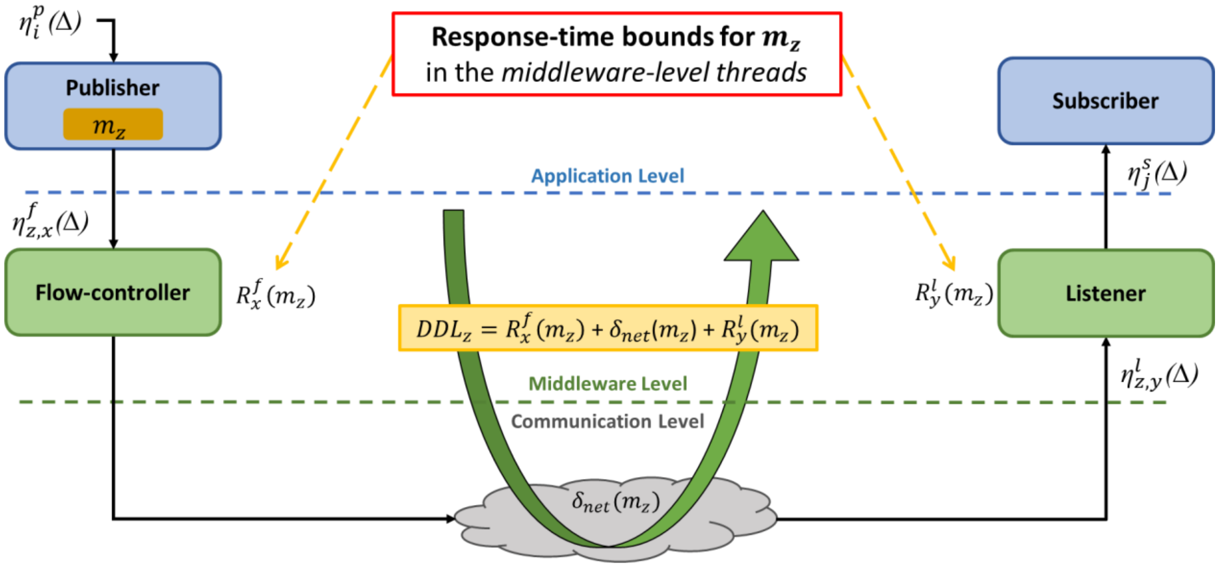
Such scheduling levels have substantial effects on the timing behavior of the final application.

End-to-end latency analysis

RETIS Lab developed

- a compositional model for **DDS-enabled RT systems**
- a specific instance for **FastDDS**
- a fine-grained **response-time analysis** for FastDDS messages

Main benefit: **validate the timing requirements of** complex DDS-based systems



Reference paper

G. Sciangula, D. Casini, A. Biondi, C. Scordino, M. Di Natale, "Bounding the Data-Delivery Latency of DDS Messages in Real-Time Applications", Proc. of the Euromicro Conference on Real-time Systems (ECRTS 2023), Vienna, Austria, July 11-14, 2023.

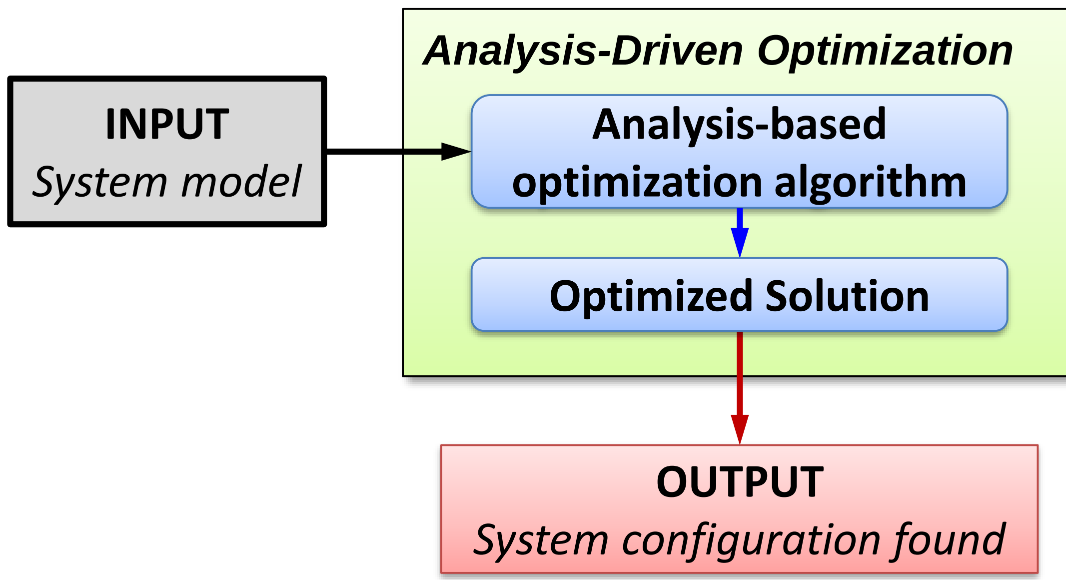
Optimization of DDS RT systems

Real-Time Systems Laboratory

RETIS Lab developed

- **Analysis-driven optimization** for automatic design-space exploration of FastDDS-based RT systems.
- Case study evaluation based on **Autoware Reference System**.

Main benefit: **helping** designers in **configuring** DDS-enabled real-time systems



Reference paper

G. Sciangula, D. Casini, A. Biondi, C. Scordino, "End-to-End Latency Optimization of Thread Chains Under the DDS Publish/Subscribe Middleware", Proc. of the Design, Automation, and Test in Europe Conference (DATE 2024), Valencia, Spain, March 25-27, 2024.

AI issues

Real-Time Systems Laboratory

Problems of current AI

1. AI models are **computationally intensive**: HW acceleration

2. HP-HW **not always available** in embedded systems to run in RT: model compression (quantization, pruning, distillation, optimization)

3. Even if available, **GPUs are unpredictable**: FPGAs are more predictable and consume less energy

4. AI models are **not trustworthy**: prediction score ≠ confidence: methods to detect anomalous inputs and derive confidence.

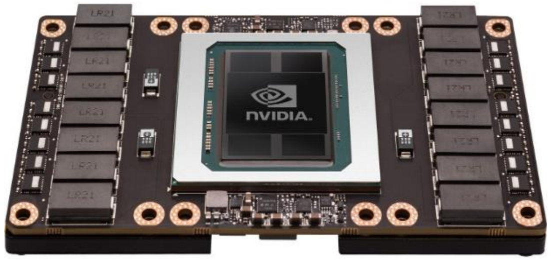
5. AI models are prone to **adversarial attacks**, also in the real world: detection and defense mechanisms

AI acceleration

DNN acceleration

To be used in real time, the inference of modern DNN models requires hardware acceleration. This is usually done by

General purpose GPUs (**GPGPUs**)

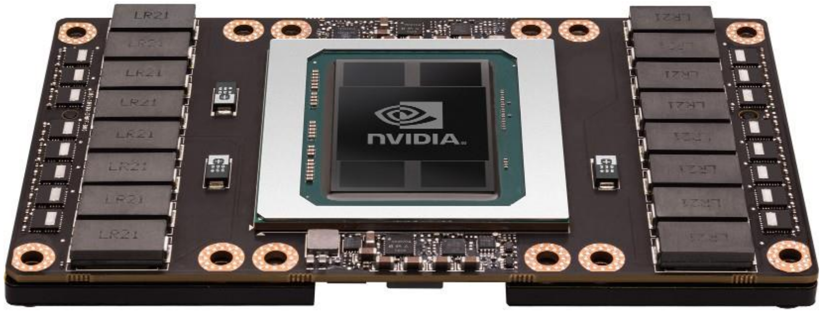


Programmable logic (**FPGA**)

Both solutions have **pro** & **cons**
both requires DNN optimization

GPU acceleration

GPGPUs are the most used to accelerated DNNs, because of two main **advantages**:



- ✓ **Response time** can be reduced by two orders of magnitude;
- ✓ Development is supported by **standard frameworks**.

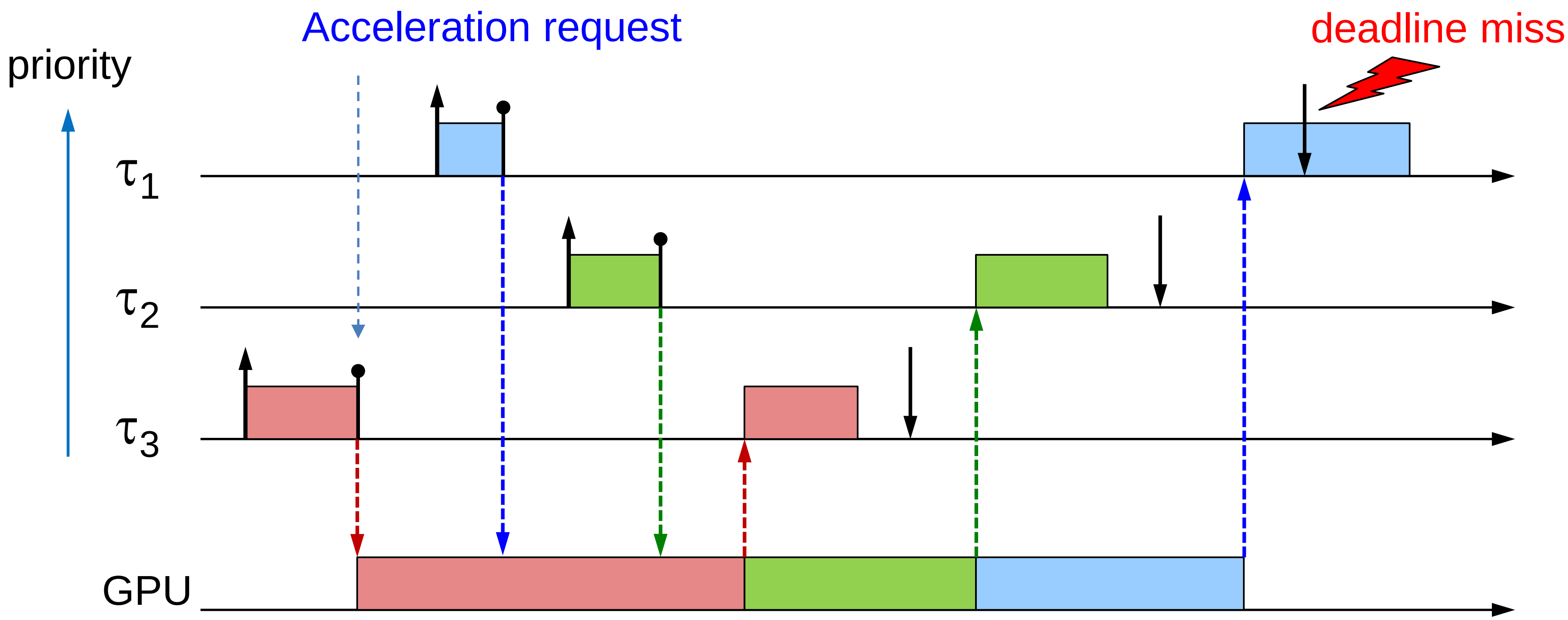
On the other hand, there are serious **disadvantages**:

- ✗ Concurrent tasks are executed in **non-preemptive** fashion;
- ✗ Significant **power consumption**, **weight**, and **encumbrance**.

This prevents their usage in small embedded systems:

GPU + TensorRT

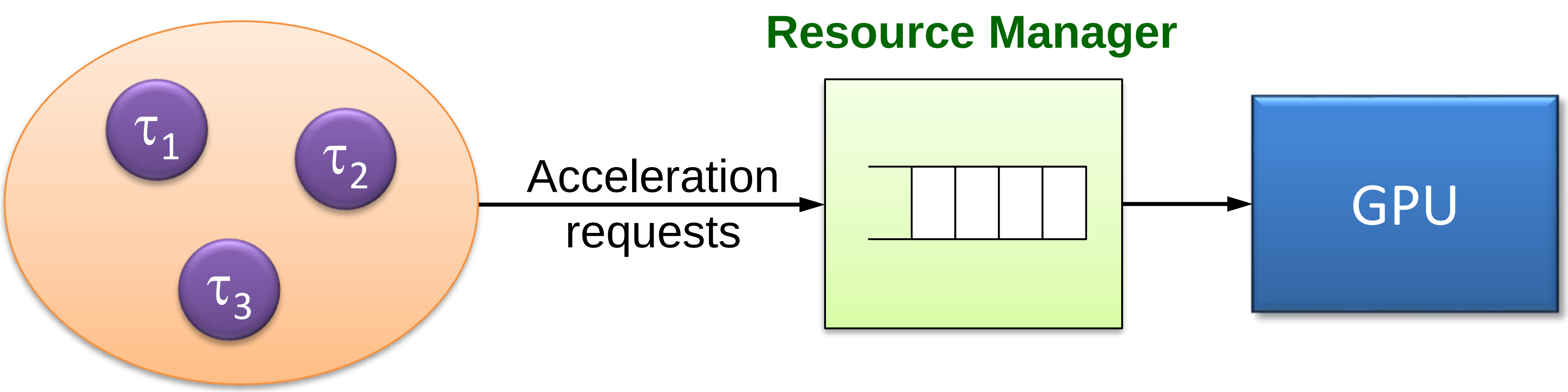
Since the execution of GPU requests is **non-preemptive**, high-priority requests cannot preempt lower-priority ones:



Note that GPU requests may not be served by FCFS due to internal memory constraints.

GPU + TensorRT

To solve this problem, an external **Resource Manager** must be implemented to properly schedule the acceleration requests coming from the application tasks:



FPGA acceleration

On the other end, FPGAs have the following advantages:

- ✓ They exhibit a highly **predictable** behavior in terms of execution times.
- ✓ They consume much **less power** with respect to GPUs.
- ✓ Commercial boards have **lower weight**, **encumbrance**, & **cost**.

Hence, they are ideal for battery-operated systems, as space robots, satellites, and UAVs. But…

- ✗ **No FPU** is available, unless explicitly programmed by the user (but consuming a fraction of the available fabric).
- ✗ **Difficult programming** (efficient coding requires a deep knowledge of low-level architecture details).

Retis
Real-Time Systems Laboratory

FPGA acceleration

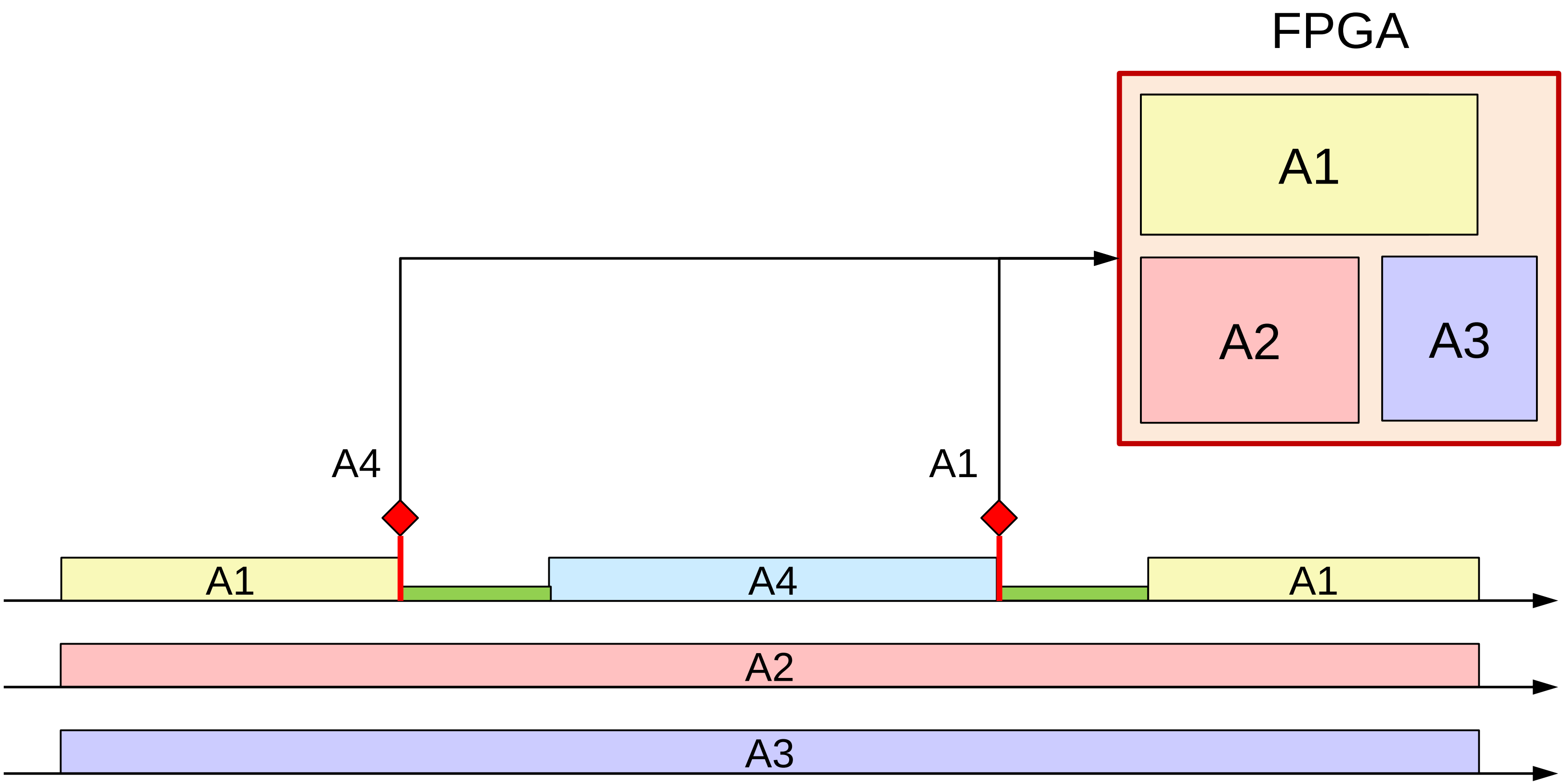
Deploy the **full DNN** on the FPGA → ✓ Faster, ✗ less flexible, ✗ DDN may not fit

Accelerate DNN operations by a coprocessor (**DPU**) → ✗ Slower, ✓ More flexible

We considered both approaches providing solutions for both of them.

The FRED framework

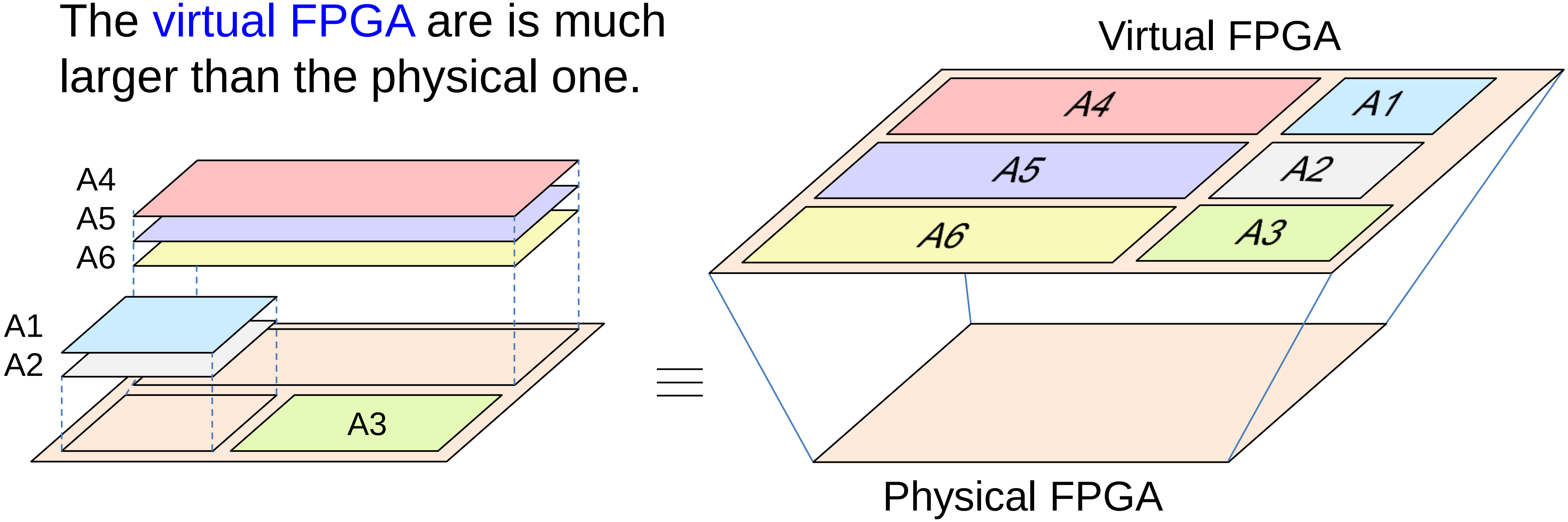
Dynamic partial reconfiguration (**DPR**) allows reprogramming a portion of the FPGA while the rest is still running:



FPGA virtualization

RETIS Lab developed a programming framework (**FRED**) that exploits **dynamic partial reconfiguration** (**DPR**) to virtualize the FPGA area:

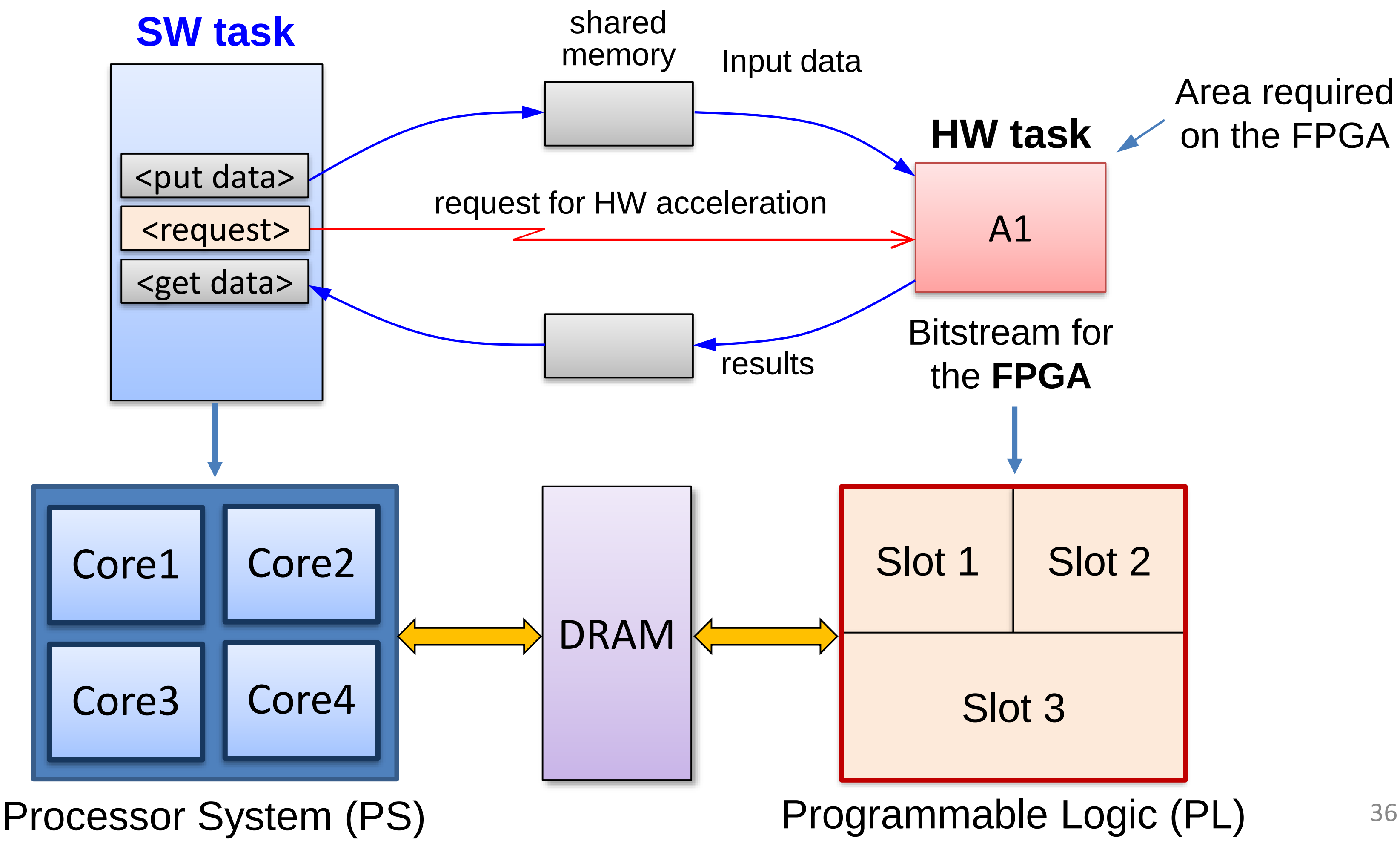
The virtual FPGA are is much larger than the physical one.



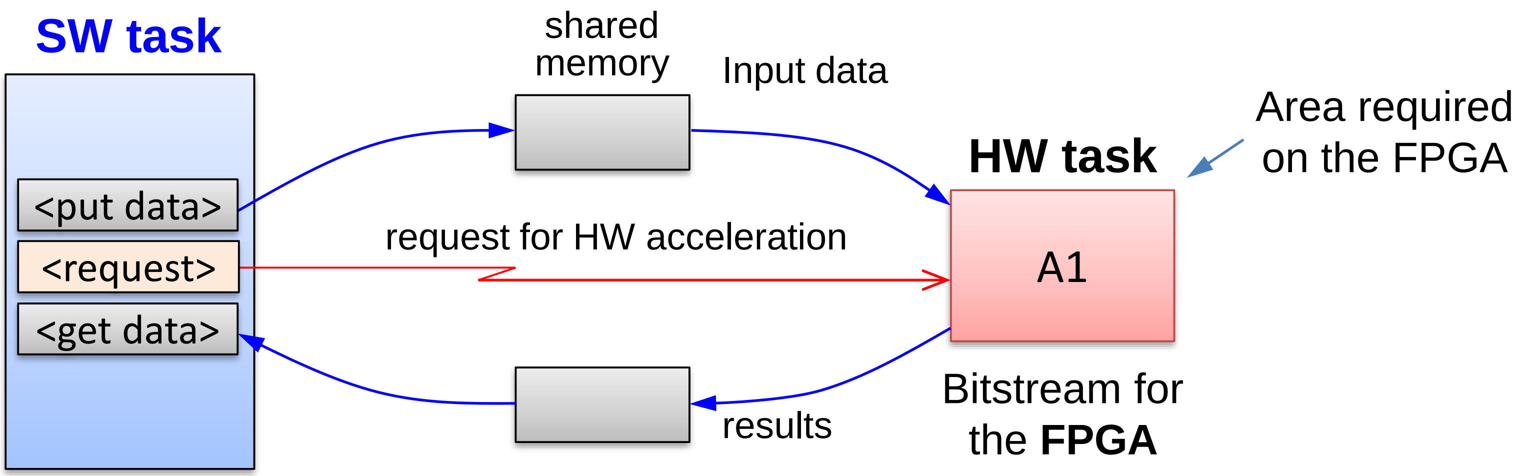
Timesharing is possible if HW accelerators do not run continuously, but execute periodically with $T_i > C_i$ (which is normally the case).

Task model

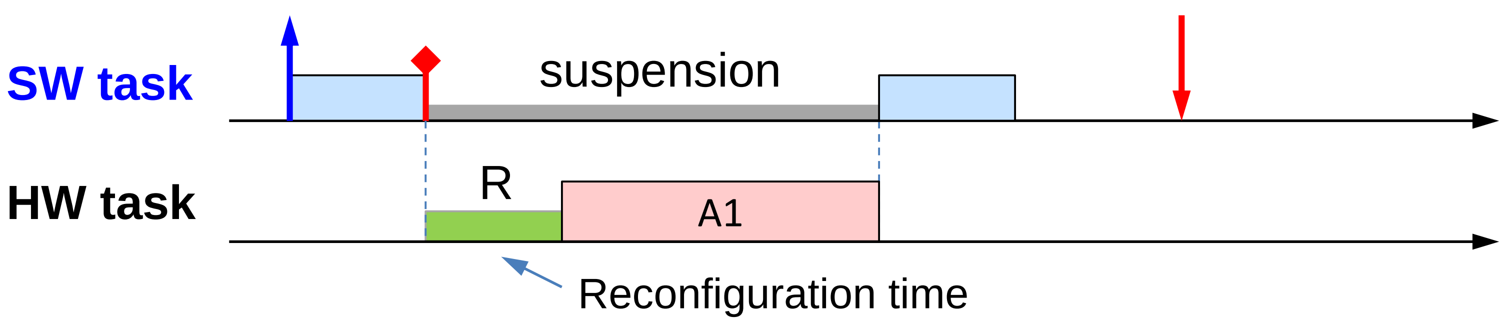
FRED applications consist of **SW-tasks** (running on the PS) and **HW-tasks** (running on the PL):



Task model

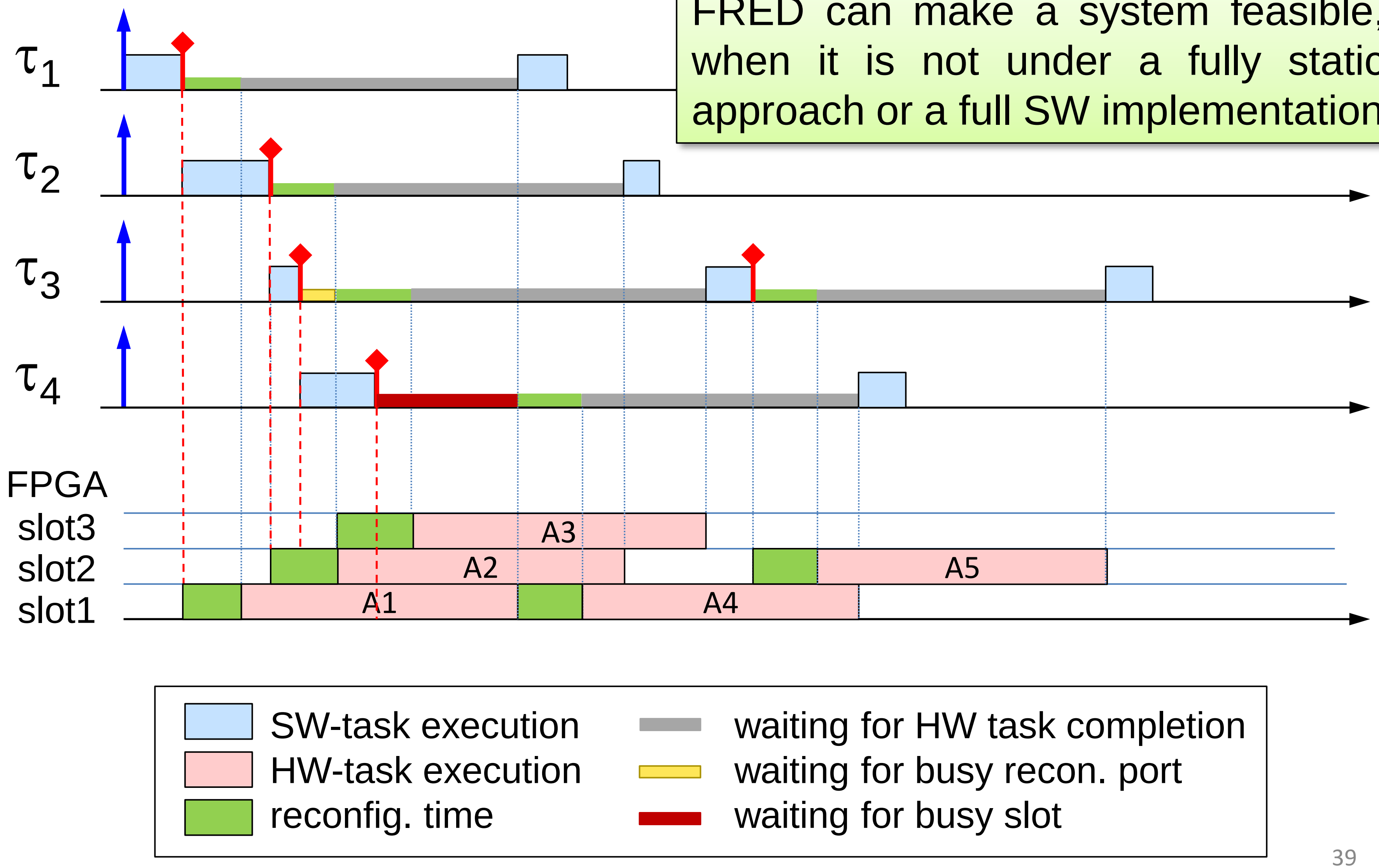


After issuing a **request for acceleration**, a SW task is suspended until the results are produced.



Example of schedule

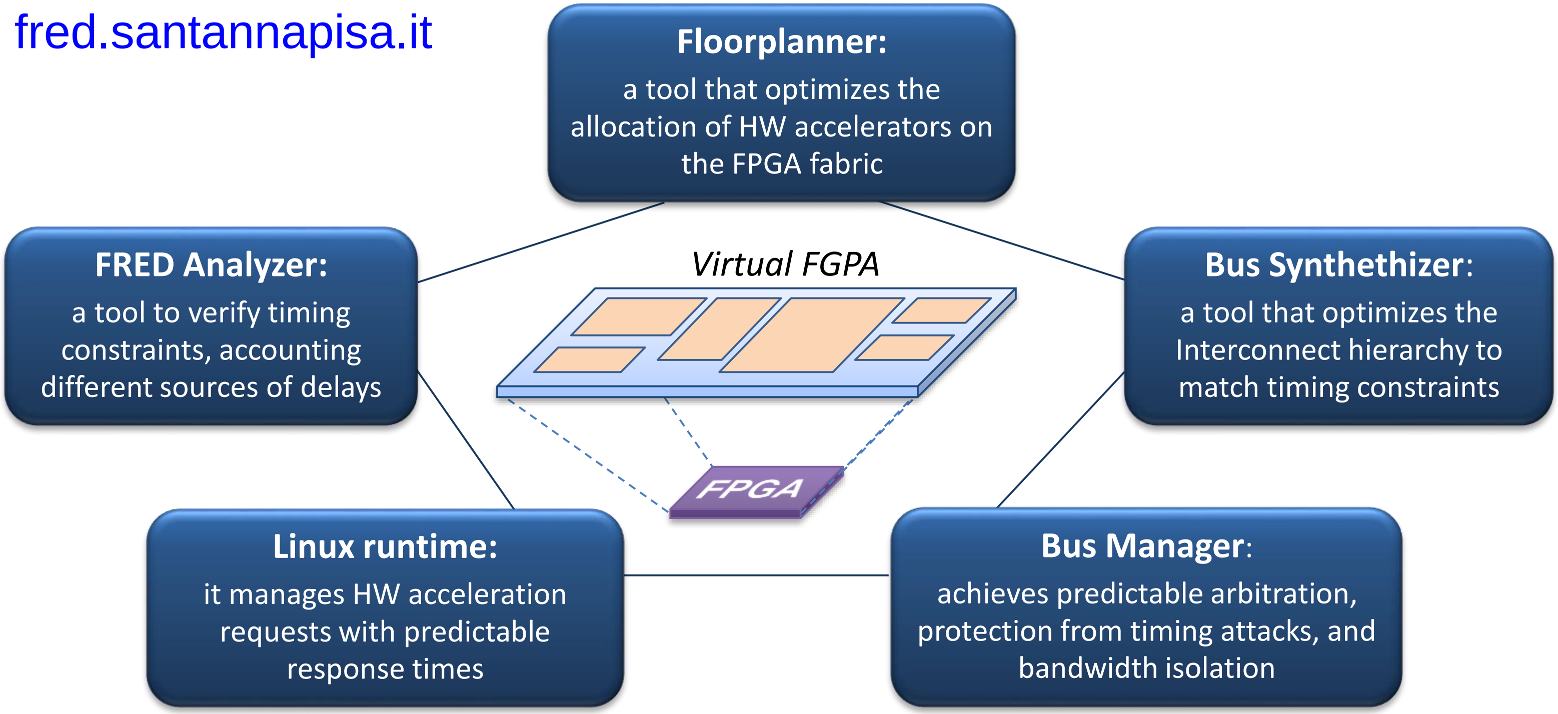
FRED can make a system feasible, when it is not under a fully static approach or a full SW implementation



The FRED framework

FRED includes a set of tools:

URL: fred.santannapisa.it

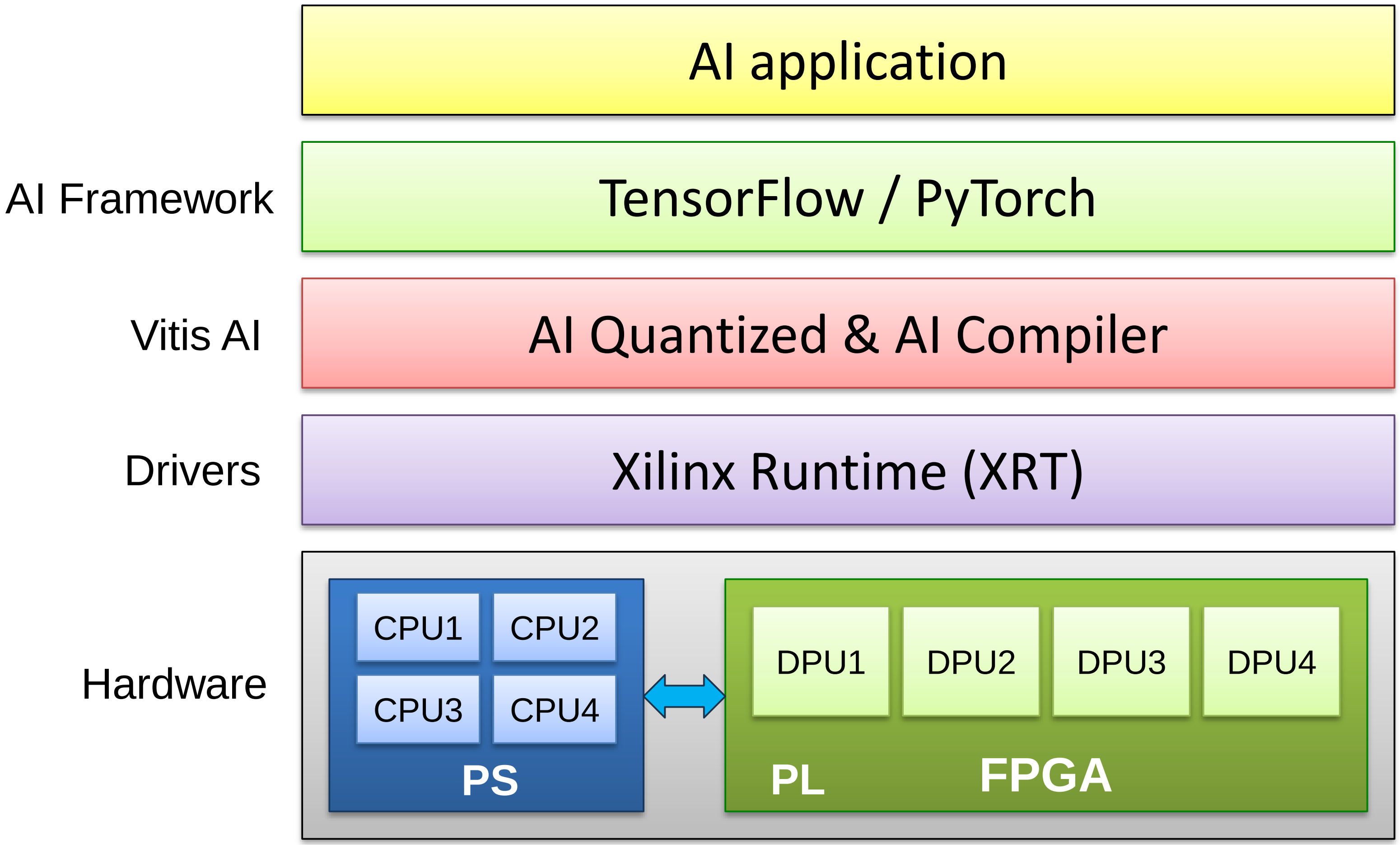


FRED Paper

A. Biondi et al., "A Framework for Supporting Real-Time Applications on Dynamic Reconfigurable FPGAs", Proc. of the IEEE Real-Time Systems Symposium, 2016.

Xilinx DPU

A more flexible way to accelerate AI models is by a proper **softcore coprocessor**, as the Xilinx **deep learning processing unit** (**DPU**):



DNN optimization

To meet real-time constraints, other optimization steps are usually needed on trained DNNs:

- Weight quantization (convert floats to n-bit integers)
- Pruning (remove redundant nodes/weights)
- Layer fusion (e.g., merging conv-bias-relu)

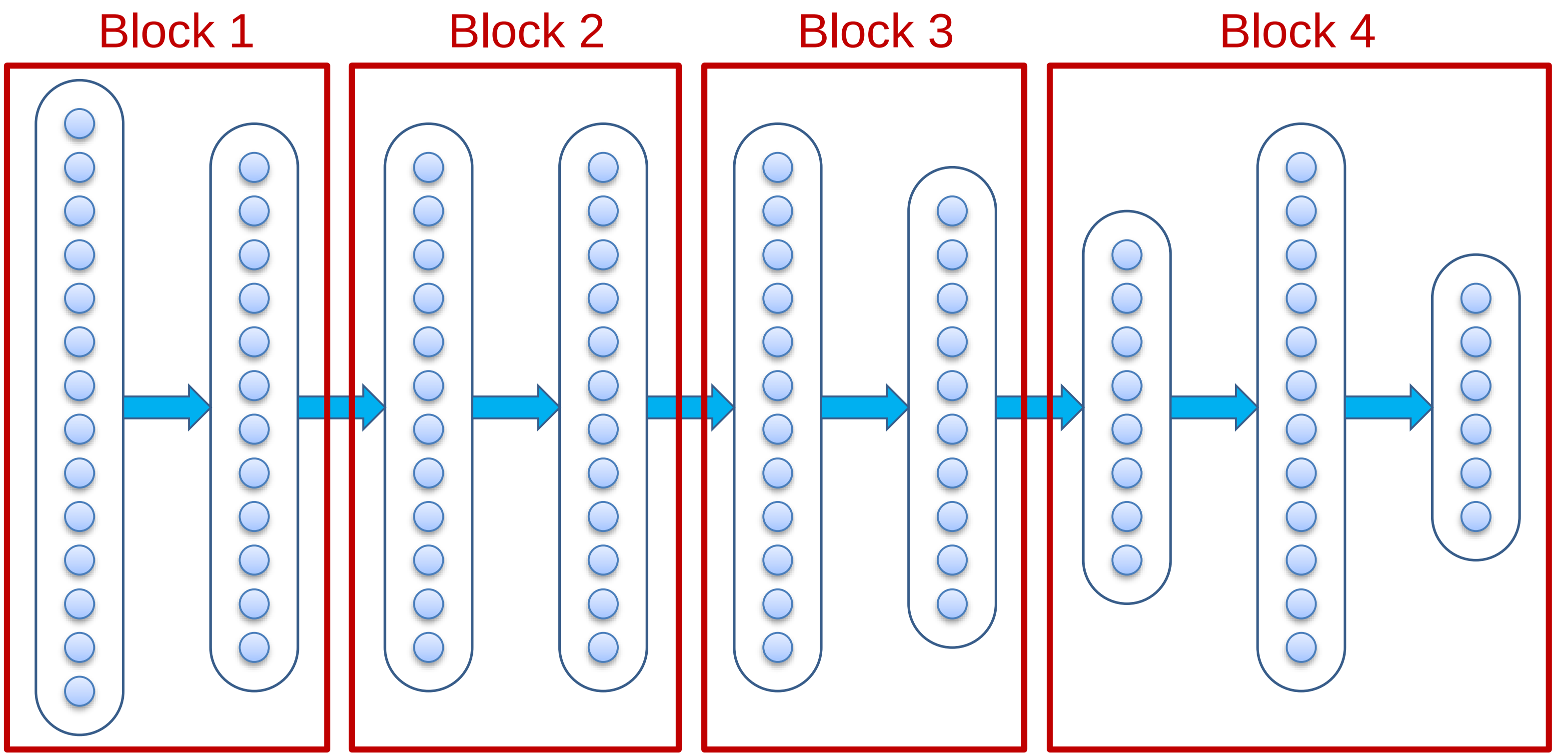
This allows several benefits, such as reducing

- computation times
- memory footprint
- energy consumption

while keeping almost the same accuracy.

DNN splitting

In complex CPS using multiple DNNs, a network can be split into several blocks to enable preemption and improve response times of higher-priority DNNs:



Choosing the best split points is an optimization process.

Projects on AI acceleration

RETIS Lab has two projects on AI accelerations funded by the *Italian Ministry of Research*

1. **OPERAND: reconfigurable platform for AI inference on the edge**

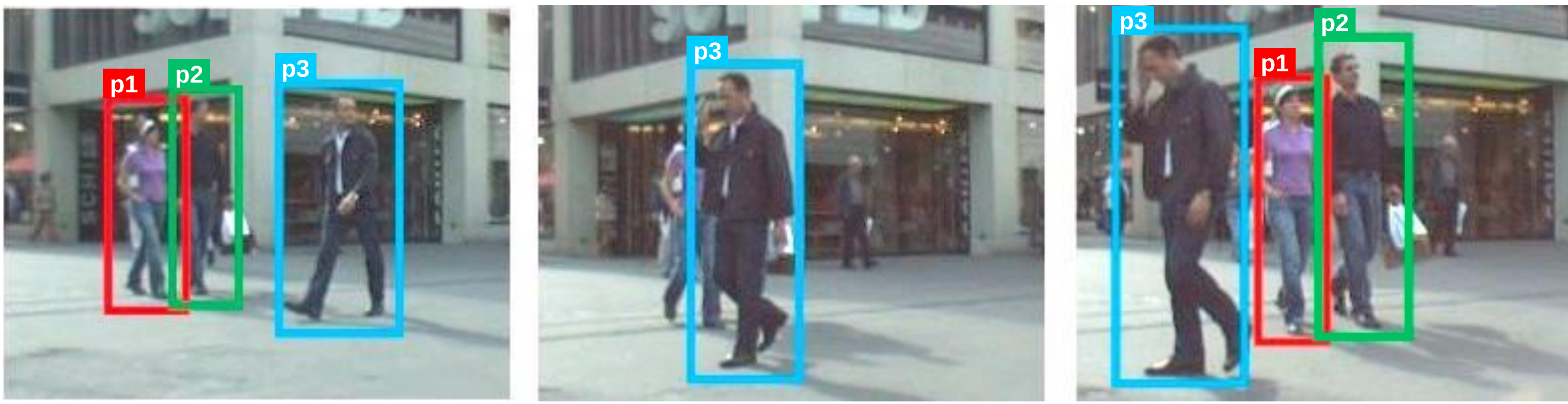
   Objective: develop a predictable **AI accelerator** for safety-critical systems with built-in support for redundancy and voting.

2. **RETICULATE: Real-time & secure acceleration framework for AI**

   Objective: develop a secure and deterministic AI acceleration framework for FPGA using Vitis AI framework and the DPUs.

Optimized real-time tracking

Real-time object tracking, requires tracking multiple objects even in the presence of **occlusions**:



To do that, **neural trackers** exploit three main methods:

Object detection
CNN (e.g., YOLO)

Kalman filter to predict motion during occlusions

→

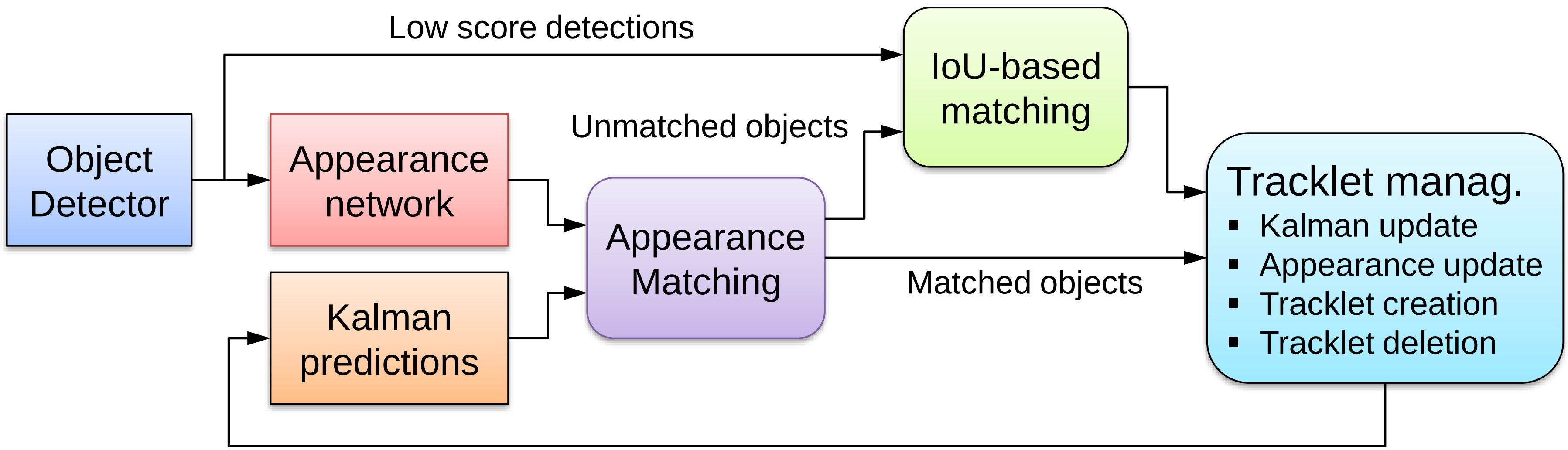
Optimization algorithm
to best match predictions with detections (using both IoU and appearance)

Optimized real-time tracking

We optimized the entire tracking pipeline by:

- accelerating CNNs on multiple DPUs on FPGA
- accelerating image pre- and post-processing on FPGA
- parallelizing the matching algorithm on multiple cores

Xilinx
Ultrascale++
ZCU104
Kria

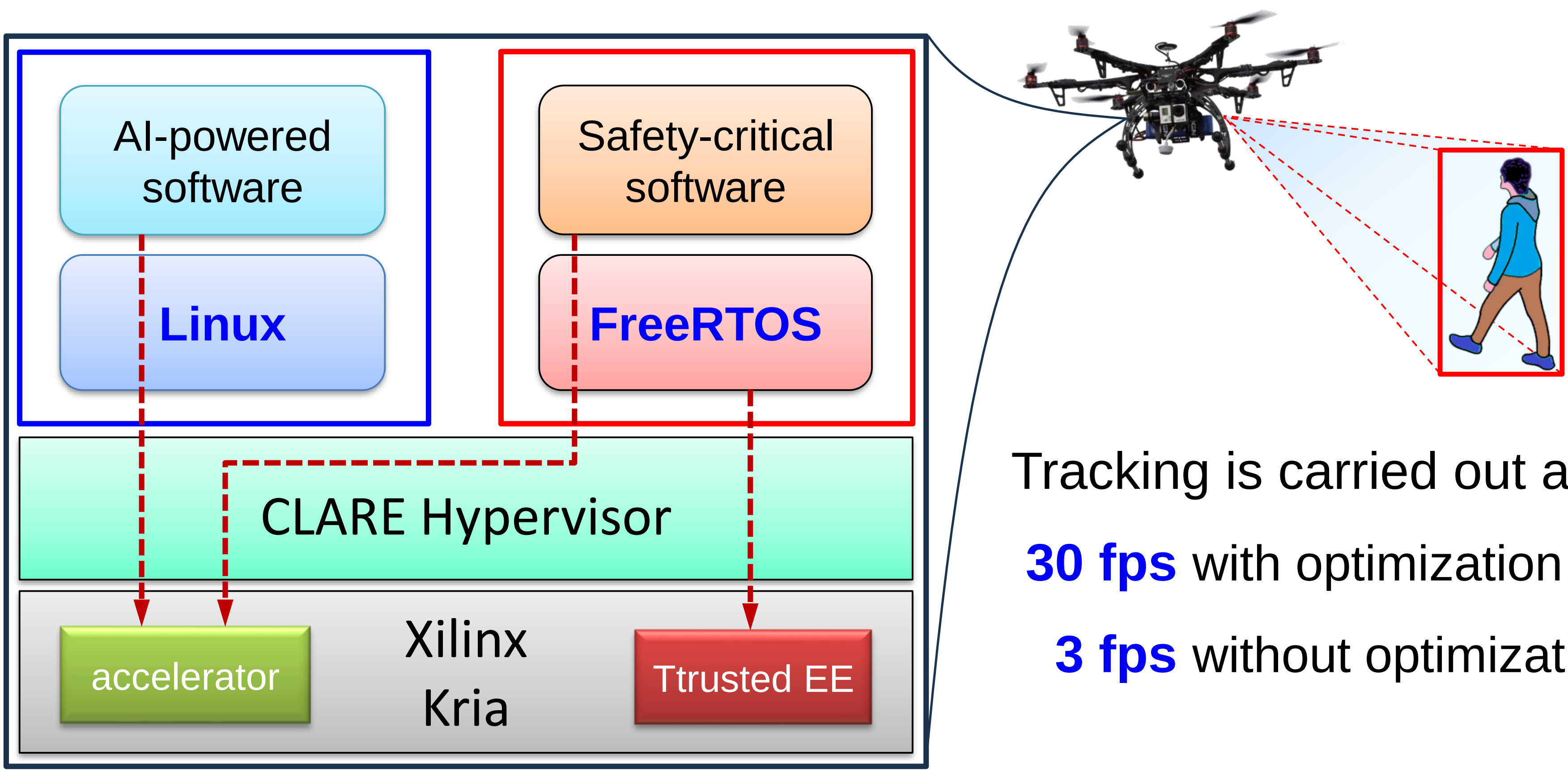


Reference paper

E. Cittadini, M. Marinoni, A. Biondi, G. Cicero, G. Buttazzo, "Supporting AI-Powered Real-Time Cyber-Physical Systems on Heterogeneous Platforms via Hypervisor Technology", Real-Time Systems, 59(4):609-635, 2023.

Optimized real-time tracking

The system was implemented to track persons by a quadrotor, using two execution domains isolated by the **CLARE hypervisor**:



Tracking is carried out at

30 fps with optimization

3 fps without optimization

AI safety issues

Real-Time Systems Laboratory

Can we trust a NN?

Training set

Can we trust a DNN on inputs that are quite different from those shown in the training set?

→ **Neural Network** → Speed limit 50

Can we trust a NN?

Training set

Can we trust a DNN on inputs that are quite different from those shown in the training set?

Neural Network

?

Out-of-distribution inputs

Can a DNN recognize such images?

Accidents due to AI

***23 March 2018**: A Tesla X missed to recognize lanes and crashed into a concrete lane divider at 70 miles per hour.*

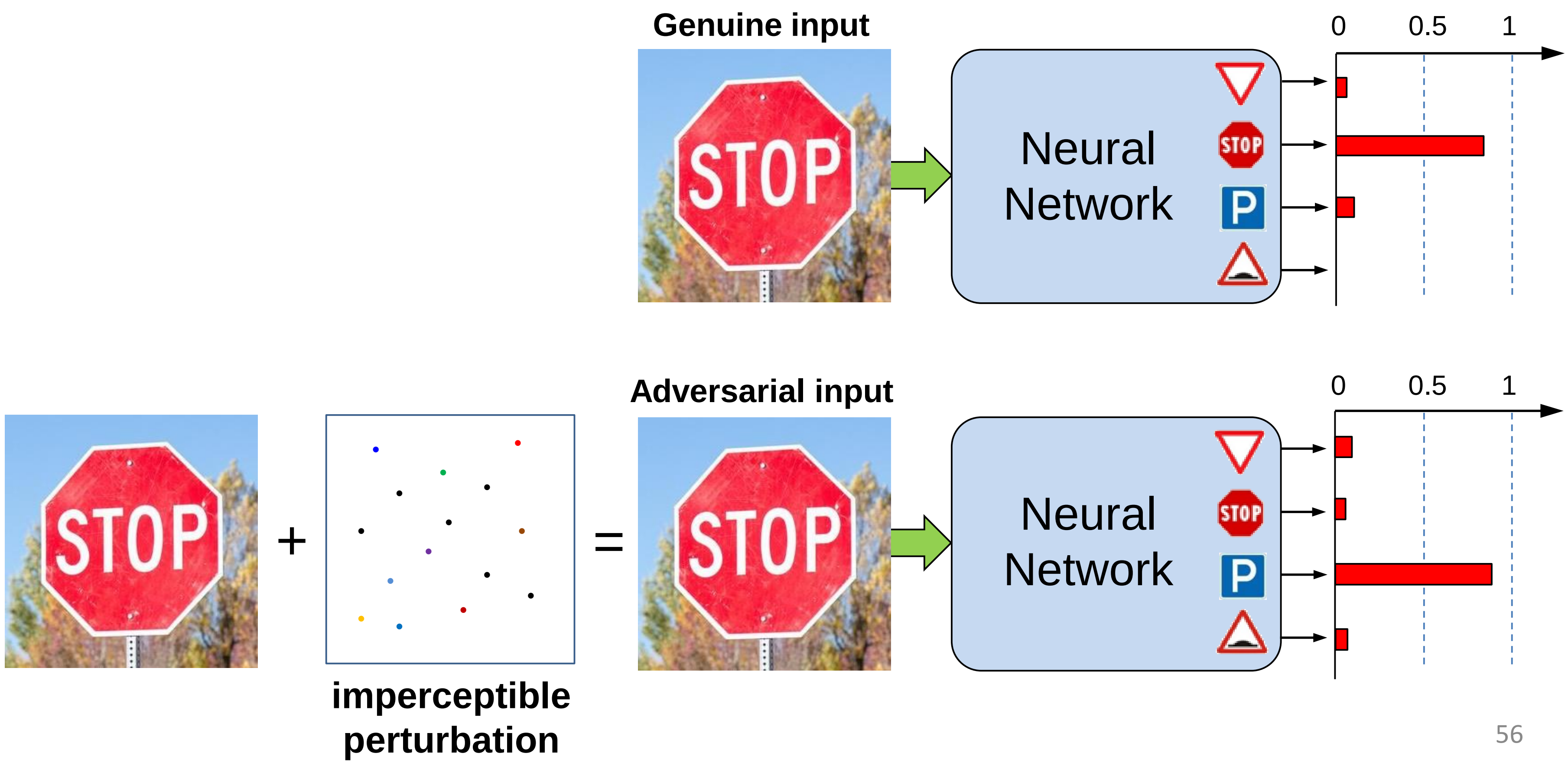
Accidents due to AI

June 1, 2020*: A model 3 Tesla missed to recognize an overturned truck on a highway in Taiwan and crashed into it at 68 mph.*

AI security issues

Cyber-attacks to DNNs

Neural networks are prone to **adversarial attacks**, i.e., malicious inputs containing imperceptible perturbations that can make a neural network to make wrong predictions.



Real-world attacks

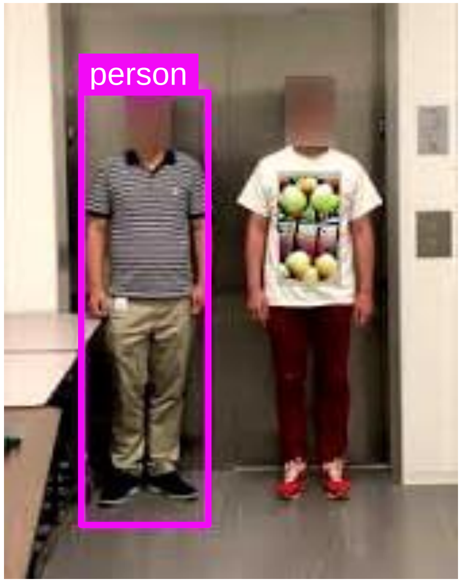
Classic adversarial inputs must have access to the AI system (DNN input, memory, or camera) to modify the image.

Real-world Adversarial attacks are directly applied to objects in the physical world, without accessing the AI system.

PARKING (92%)

BRAD PITT (93%)

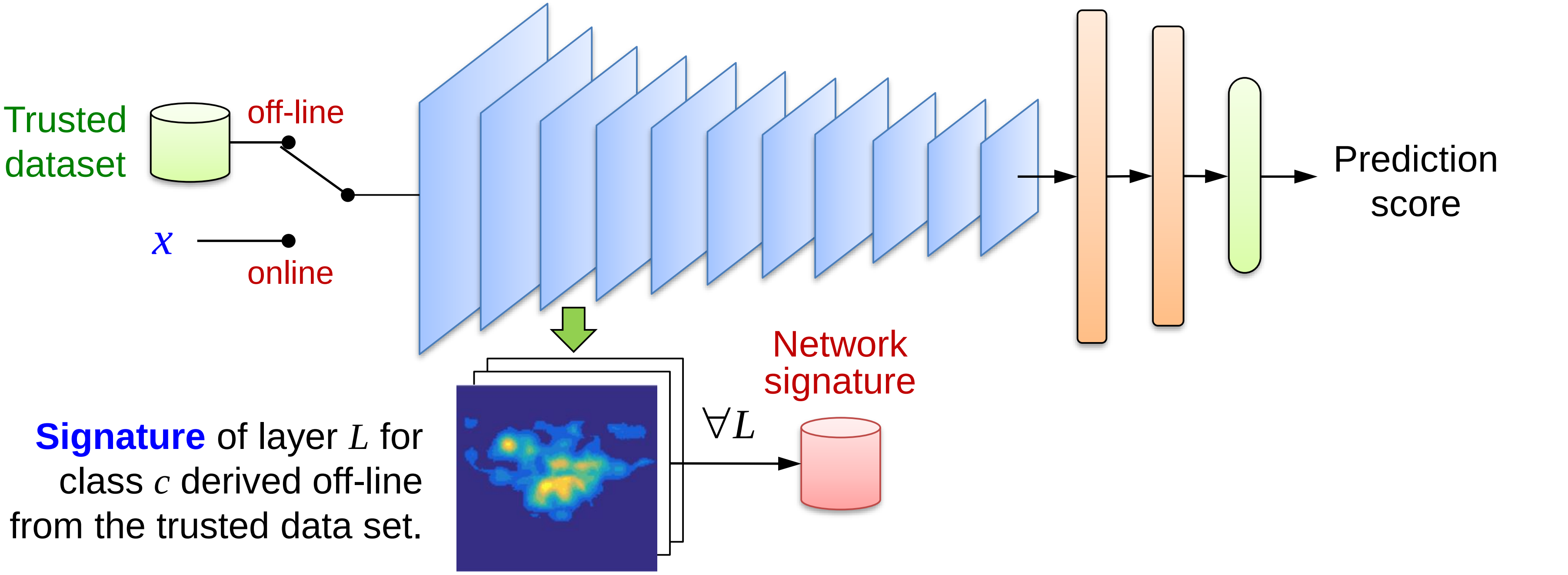
RIFLE (91%)



NO DETECTION

Coverage analysis

RETIS Lab proposed an efficient method to analyze the internal activations of a neural network to detect both **anomalous** and **adversarial inputs** through a **confidence score**:

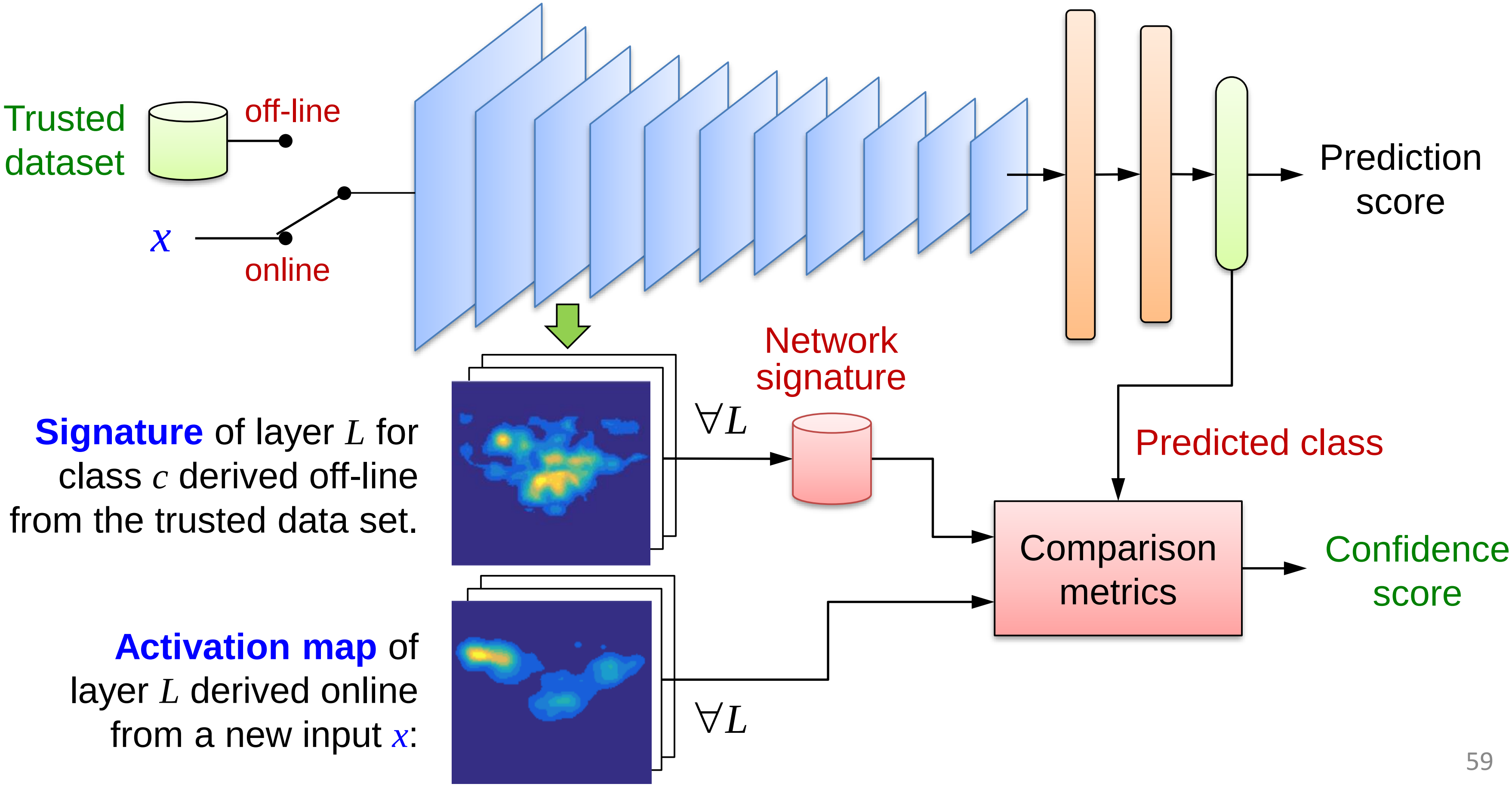


Paper

G. Rossolini, A. Biondi, G. Buttazzo, "Increasing the Confidence of Deep Neural Networks by Coverage Analysis", *IEEE Trans. on Software Engineering*, 49(2):802-815, 2023.

Coverage analysis

For a new input x, the current activation state is compared with the stored **signature** corresponding to the predicted class. The higher the matching, the higher the confidence:

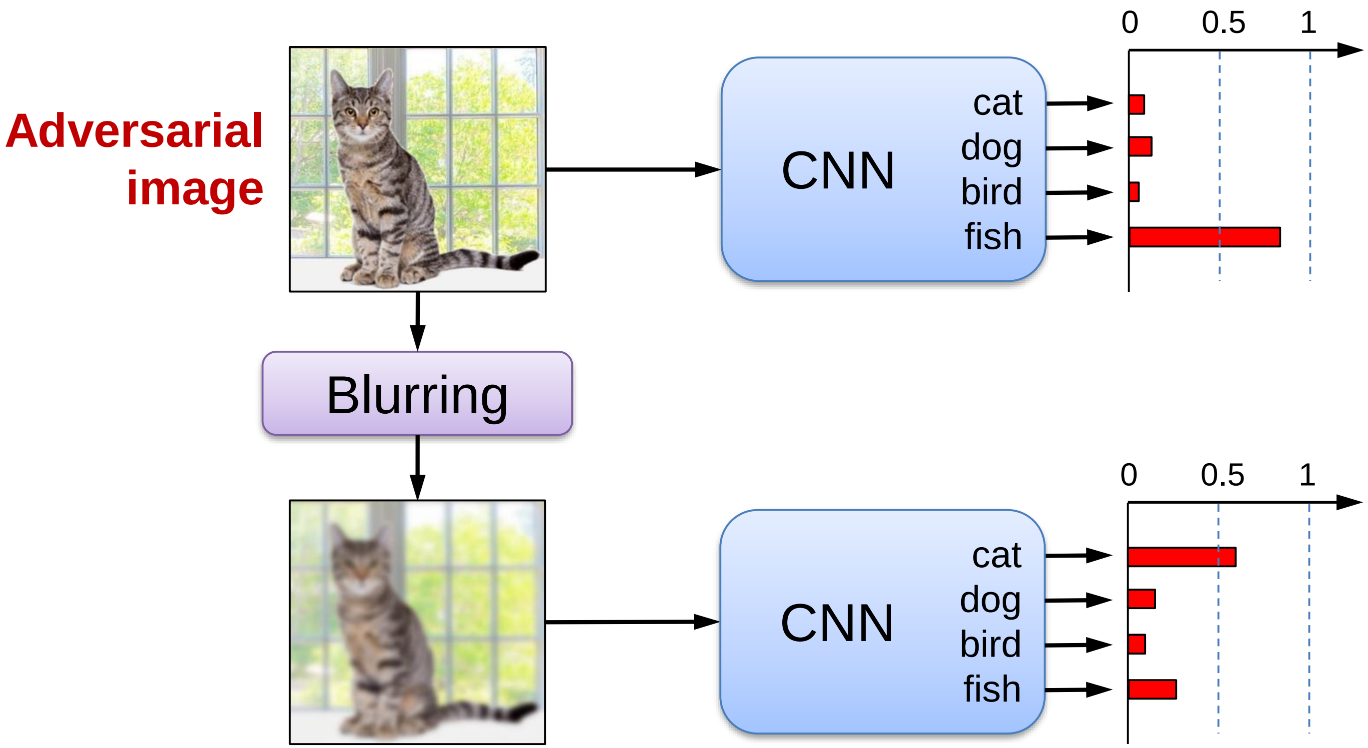


Signature of layer L for class c derived off-line from the trusted data set.

Activation map of layer L derived online from a new input x:

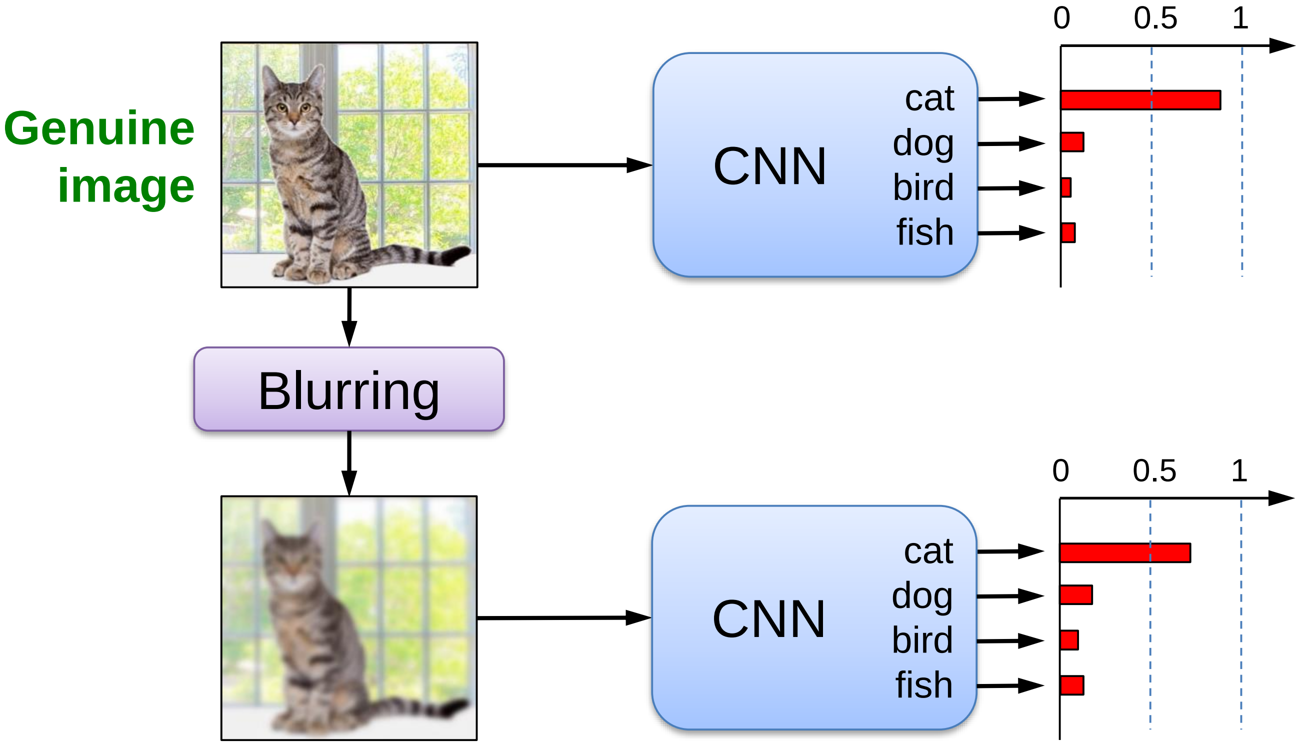
Input transformations

Another approach exploits the fact that standard AEs loose their effect when they are subject to certain **input transformations** (e.g., blurring, translation, rotations):



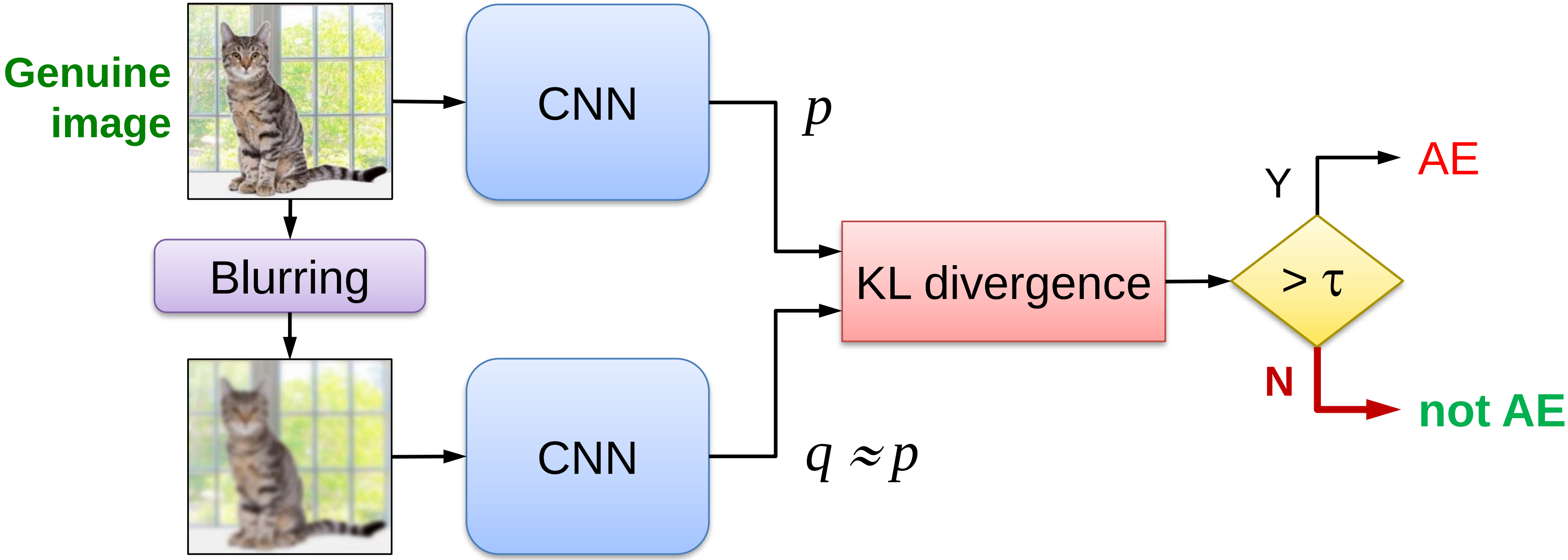
Input transformations

For genuine images, the same transformations do not cause a strong degradation in the prediction:



Input transformations

RETIS Lab proposed a detection method that compares the two distributions using a KL-divergence: a sample is considered to be AE if the two predictions are "distant" from each other:

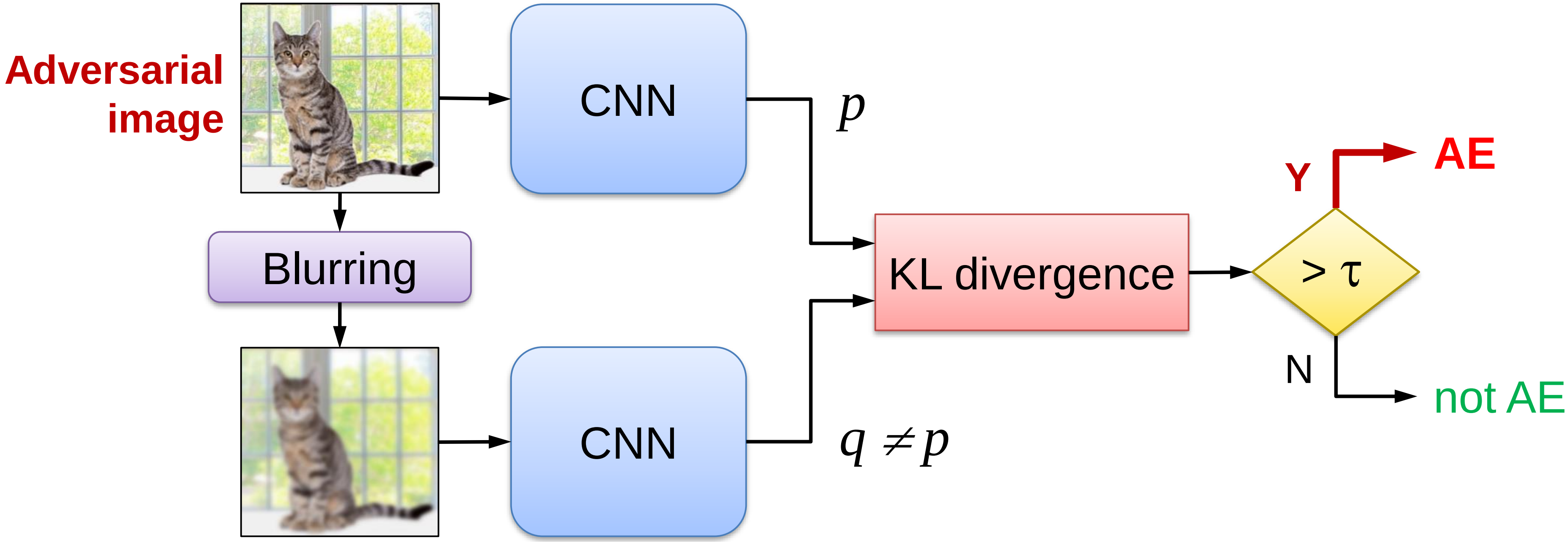


Paper

F. Nesti, A. Biondi, and G. Buttazzo, "Detecting Adversarial Examples by Input Transformations, Defense Perturbations, and Voting", *IEEE Trans. on Neural Networks and Learning Systems*, 34(3):1329-1341, March 2023.

Input transformations

RETIS Lab proposed a detection method that compares the two distributions using a KL-divergence: a sample is considered to be AE if the two predictions are "distant" from each other:



Paper

F. Nesti, A. Biondi, and G. Buttazzo, "Detecting Adversarial Examples by Input Transformations, Defense Perturbations, and Voting", *IEEE Trans. on Neural Networks and Learning Systems*, 34(3):1329-1341, March 2023.

Real-world adv. attacks

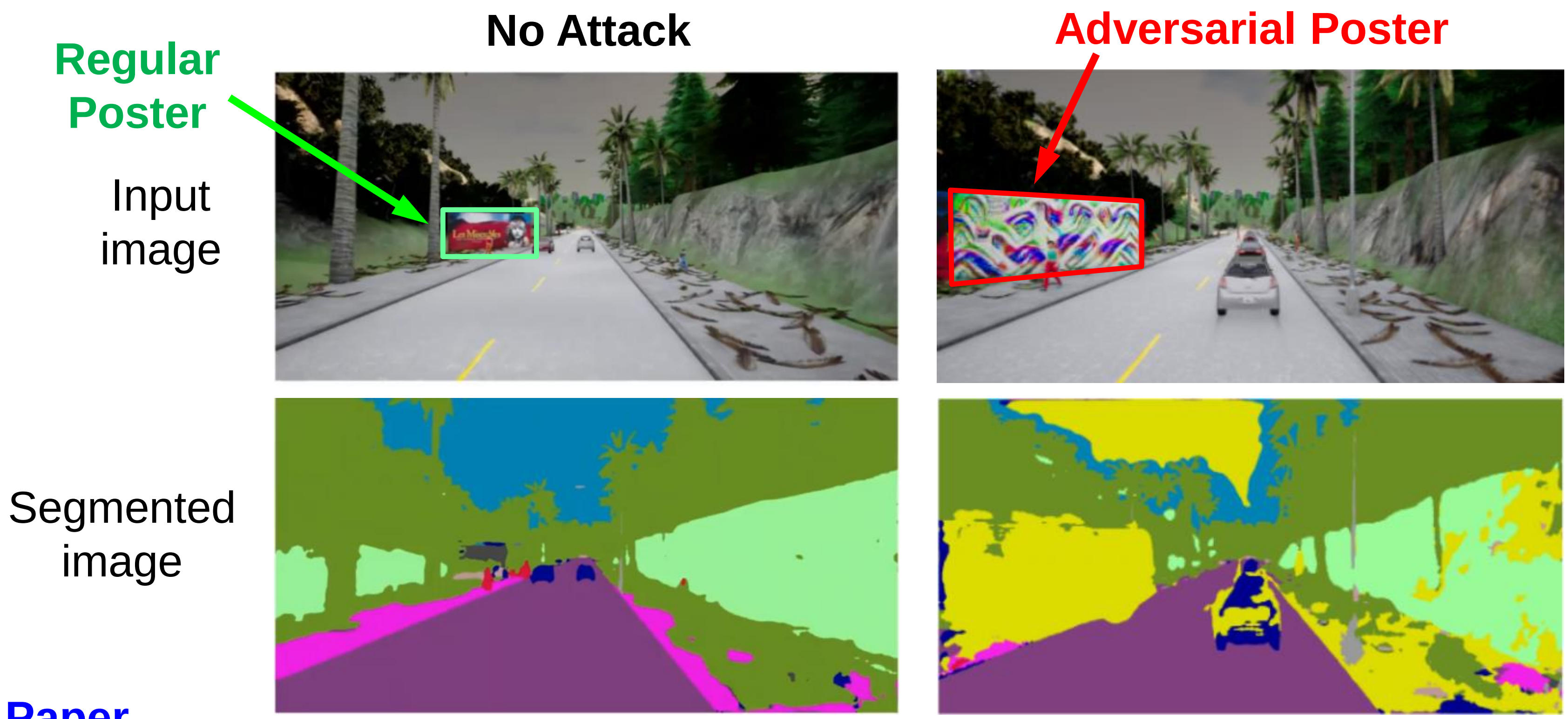
An extensive experimental study has been performed to evaluate the robustness of **segmentation networks** against real-world attacks, based on patches and physical posters:

on billboards

behind trucks

Real-world adv. attacks

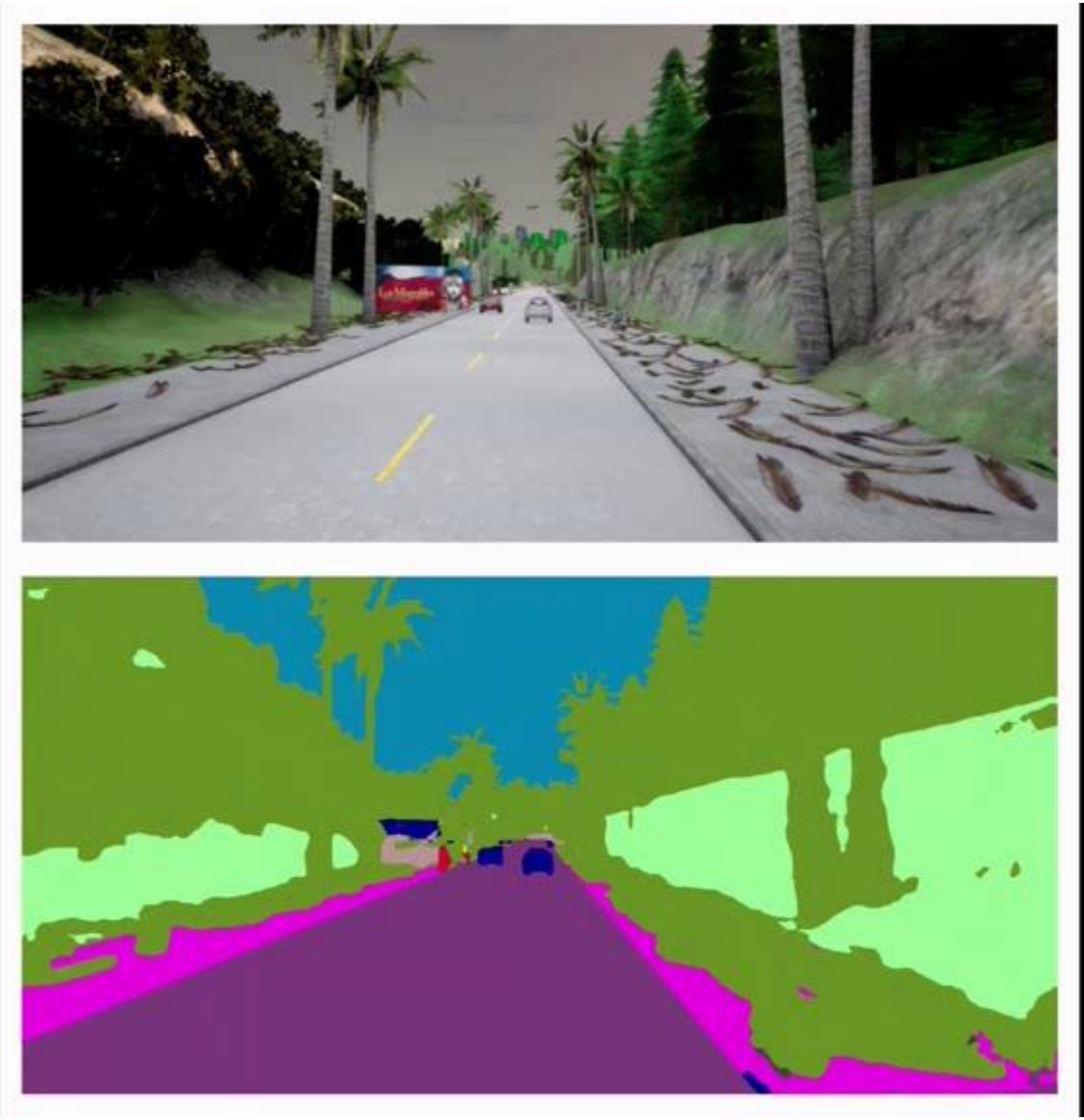
Experiments on the CARLA simulator highlighted that some semantic segmentations networks are more **sensitive to adversarial attacks**:



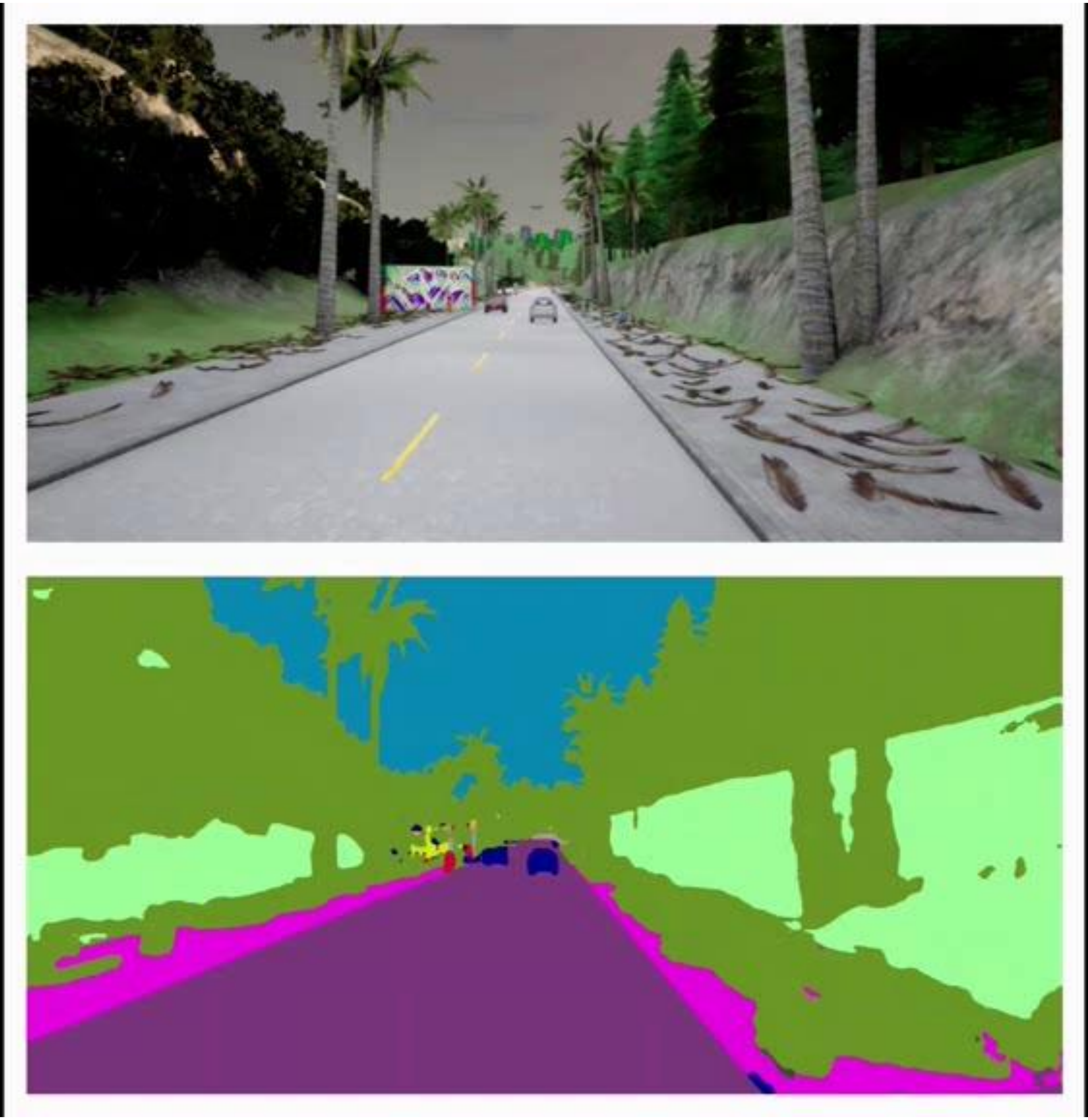
Paper

F. Nesti, G. Rossolini, S. Nair, A. Biondi, and G. Buttazzo, "Evaluating the Robustness of Semantic Segmentation for Autonomous Driving against Real-World Adversarial Patch Attacks", Proc. of WACV 2022.

Normal Poster

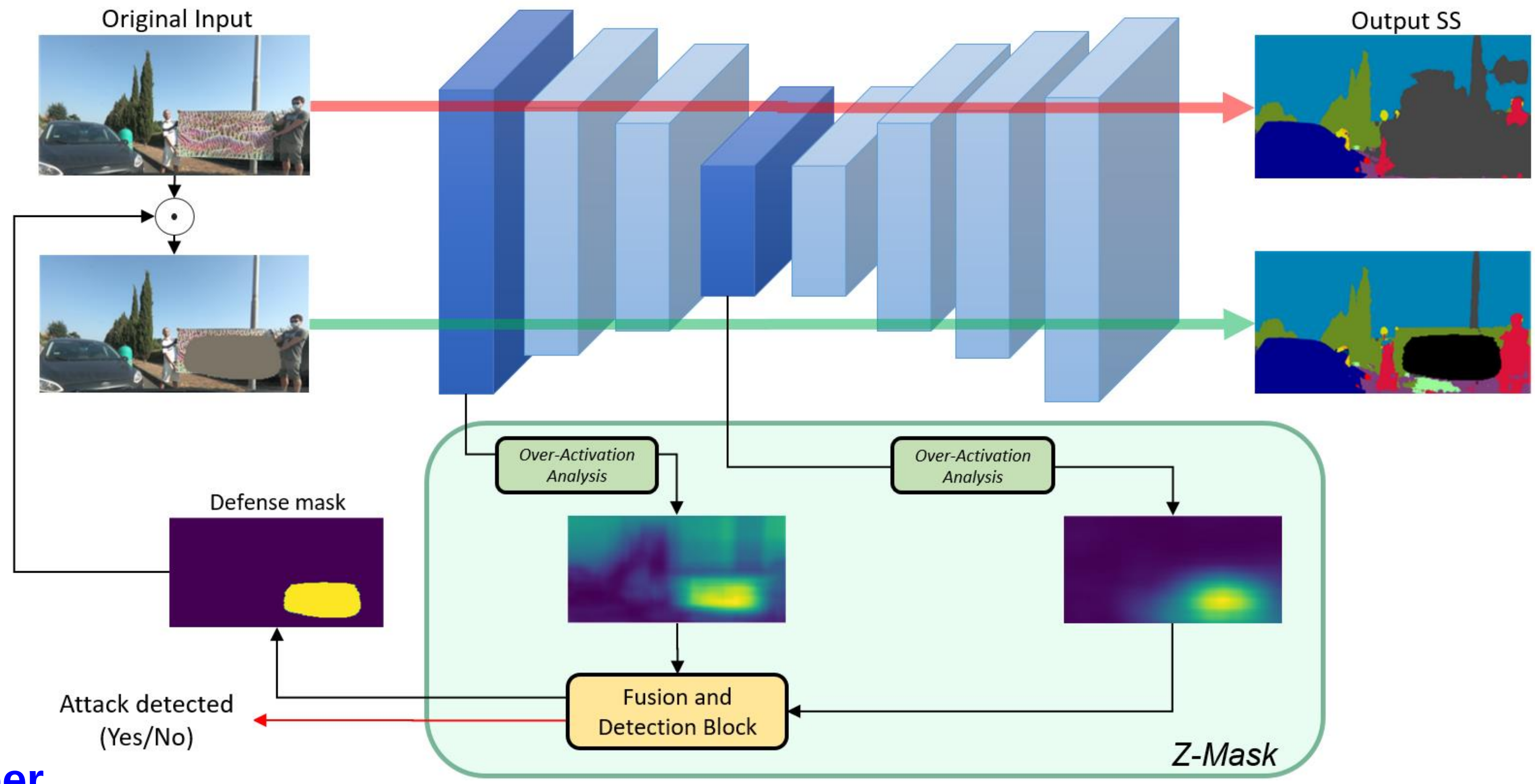


Adversarial Poster



Z-mask defense

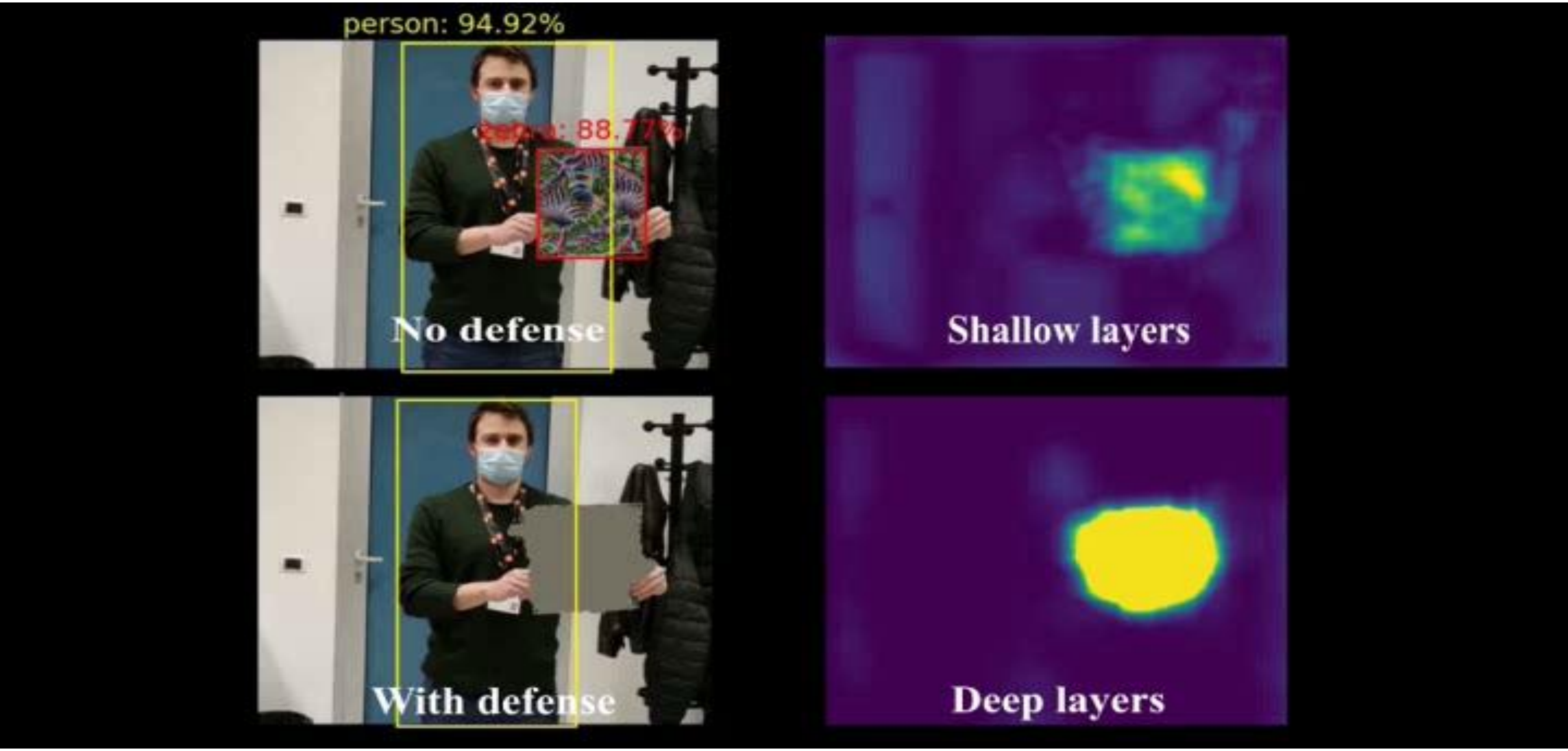
A new defense method to identify and **mask the adversarial region**:



Paper

G. Rossolini, F. Nesti, F. Brau, A. Biondi, and G. Buttazzo. "Defending from physically-realizable adversarial attacks through internal over-activation analysis", Proc. of the 37th AAAI Conf. on Artificial Intelligence, Washington, DC, USA, February 7-14, 2023.

Z-mask in action



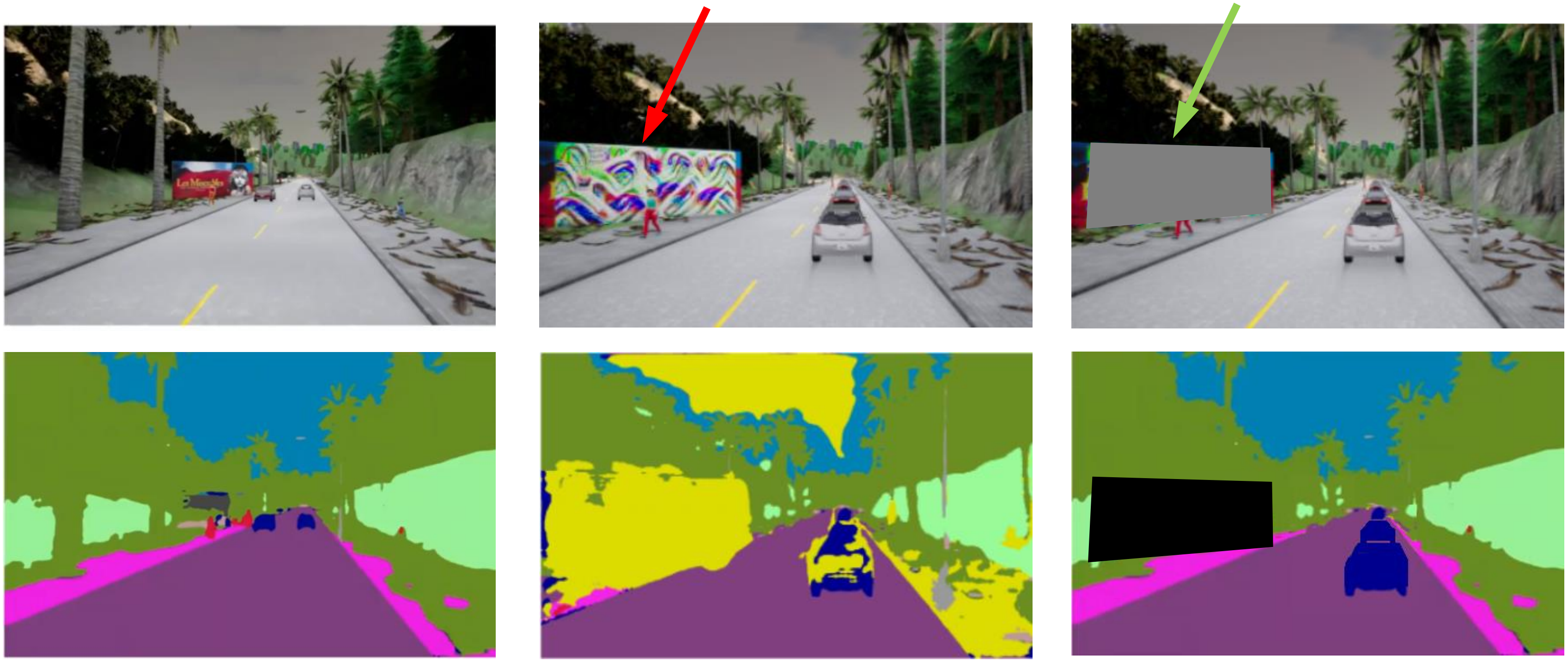
Z-mask defense

Z-mask applied on CARLA to neutralize an adversarial poster:

No attack | **Adversarial Poster** | **Defense Mask**

Input image

Seg. image



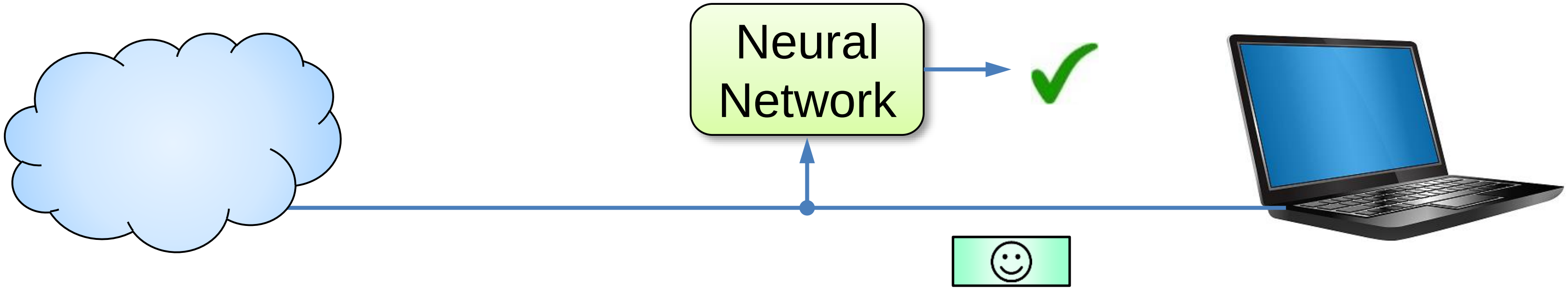
Paper

G. Rossolini, F. Nesti, F. Brau, A. Biondi, and G. Buttazzo. "Defending from physically-realizable adversarial attacks through internal over-activation analysis", Proc. of the 37th AAAI Conf. on Artificial Intelligence, Washington, DC, USA, February 7-14, 2023.

AI for security

Intrusion detection by AI

RETIS Lab developed a **universal IDS** able to detect not only a few types of malicious packets, but all anomalous packets.



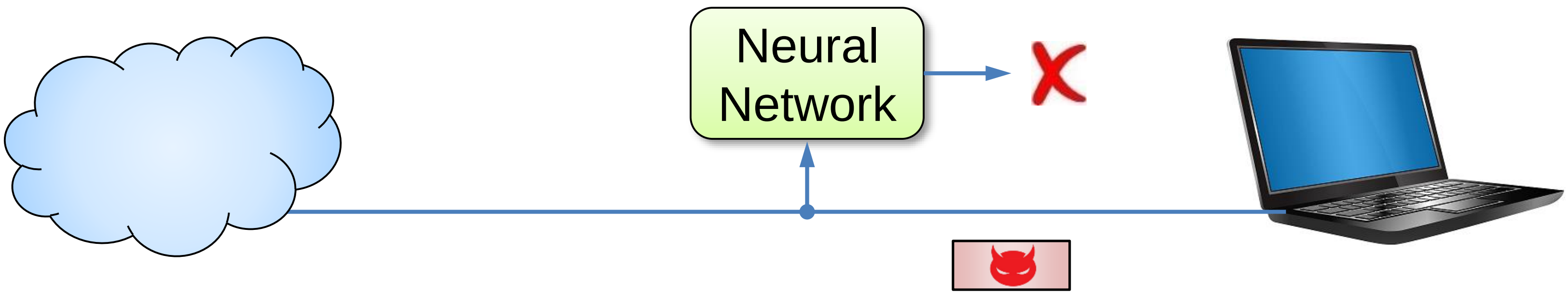
A special Multi-State Memory Autoencoder (MSM-AE) is used to recognize and reconstruct packets from regular traffic, raising a warning on anomalous ones.

Paper

N. Borgioli, L.T. Xuan Phan, F. Aromolo, A. Biondi, G. Buttazzo, "Real-Time Packet-based Intrusion Detection on Edge Devices", Proc. of the Workshop on Real-time and Intelligent Edge Computing (RAGE), San Antonio, TX, May 9th, 2023 (**Best Paper**).

Intrusion detection by AI

RETIS Lab developed a **universal IDS** able to detect not only a few types of malicious packets, but all anomalous packets.



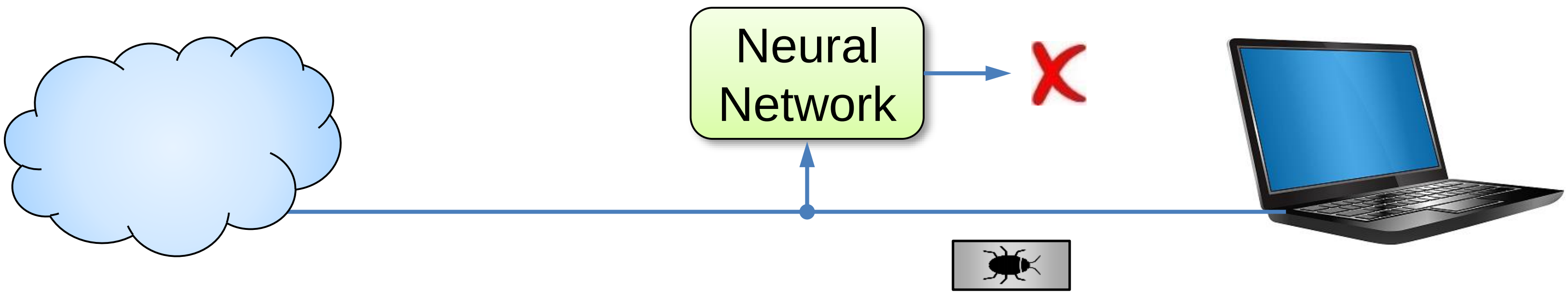
A special Multi-State Memory Autoencoder (MSM-AE) is used to recognize and reconstruct packets from regular traffic, raising a warning on anomalous ones.

Paper

N. Borgioli, L.T. Xuan Phan, F. Aromolo, A. Biondi, G. Buttazzo, "Real-Time Packet-based Intrusion Detection on Edge Devices", Proc. of the Workshop on Real-time and Intelligent Edge Computing (RAGE), San Antonio, TX, May 9th, 2023 (**Best Paper**).

Intrusion detection by AI

Being attack agnostic, new type of attacks are also detected:



Detection time

True positive rate: $TPR = \frac{TP}{TP + FN} = 99.83\ \%$

False positive rate: $FPR = \frac{FP}{FP + TN} = 0.18\ \%$

190 ms	50 ms
on CPU Cortex A78 12 cores	on GPU NVIDIA Jetson AGX Orin

Paper

N. Borgioli, L.T. Xuan Phan, F. Aromolo, A. Biondi, G. Buttazzo, "Real-Time Packet-based Intrusion Detection on Edge Devices", Proc. of the Workshop on Real-time and Intelligent Edge Computing (RAGE), San Antonio, TX, May 9th, 2023 (**Best Paper**).

Concluding remarks

So what about AI in CPS?

We have seen that AI models have **intrinsic weaknesses** in terms of

- timing predictability, safety, security, and certifiability.

Does it mean that we cannot use AI in complex CPS?

We cannot prevent AI algorithms from being attacked or producing wrong results, but we can take a number of countermeasures to prevent them from harming.

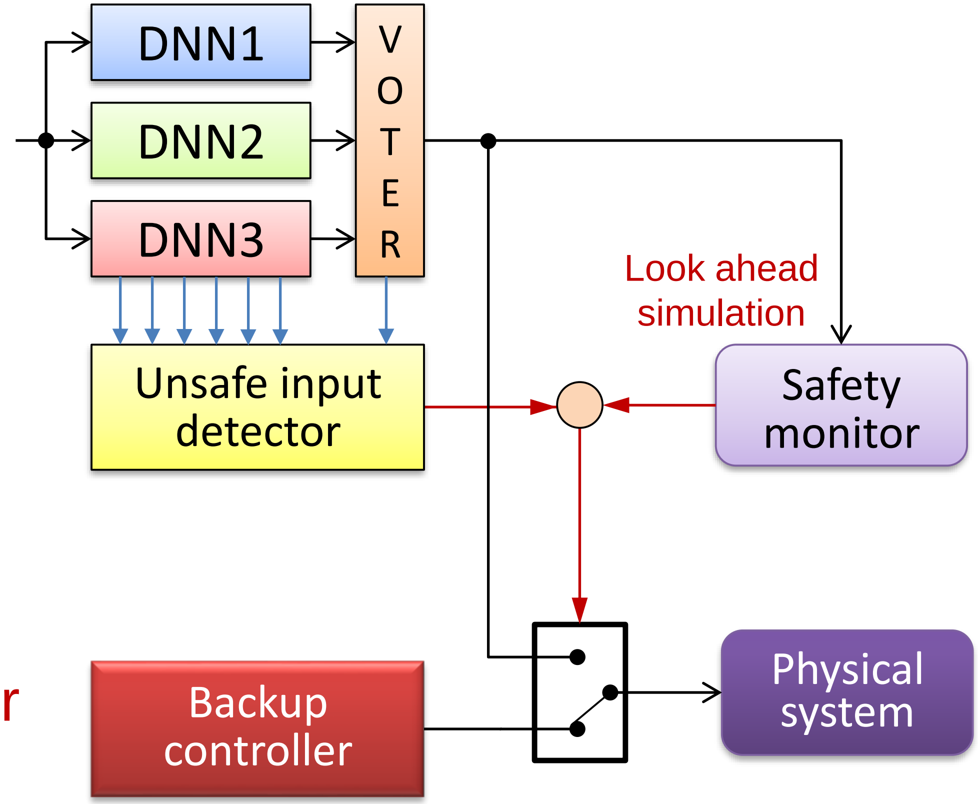
Some solutions already exist, but more research is needed to

- Increase **predictability** when accelerating AI models
- Reduce response times by compression, distillation, & optimization
- Increase **safety** by detecting faults and anomalous inputs
- Increase **security** by proper defense mechanisms

Safe architecture

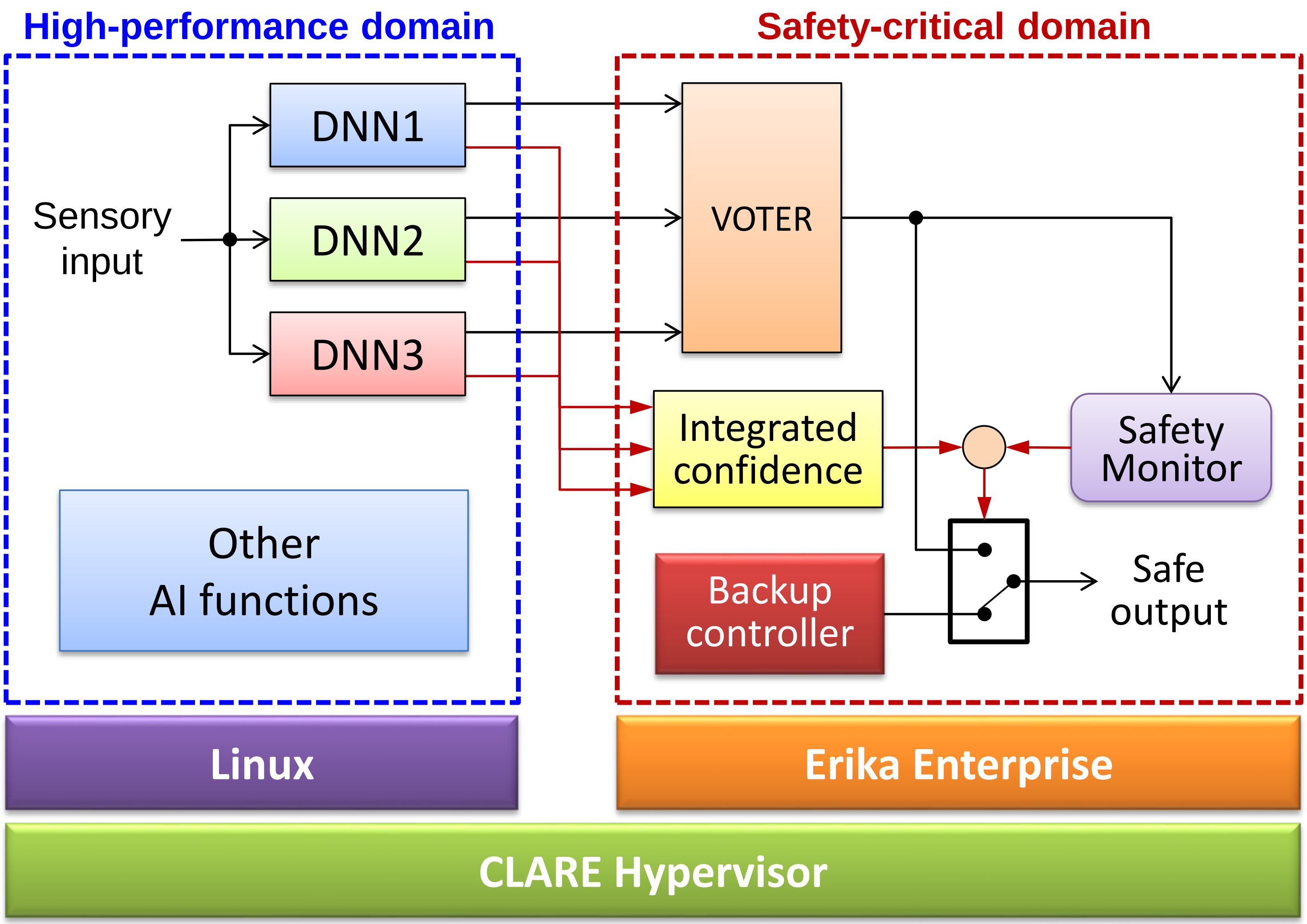
Act on the **architecture** to implement **fault detection & exclusion**:

- Achieve **fault-tolerance** by replication + voting
- Detect **anomalous inputs** and **adversarial attacks**
- Detect dangerous outputs by **safety monitoring**
- Switch to a back-up controller in anomalous conditions



Overall architecture

- Ensure **security** and **isolation** by a hypervisor.



Thank you