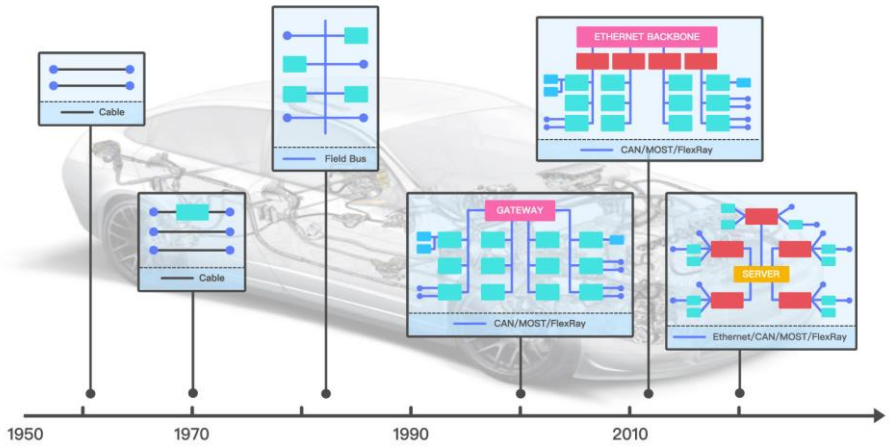
Intrusion detection on vehicle CAN networks: implementation on edge ECU and distributed architecture for product cybersecurity

Automotive SPIN Italia
24° WORKSHOP on AUTOMOTIVE SOFTWARE & SYSTEMS – 21/05/2026

Presenter: Stefano Mazzetti
(stefano.mazzetti@esws.tech, ESWS S.r.l. – Embedded SoftWare Systems)

Canino Nicasio, Pierpaolo Dini, Daniele Rossi, Sergio Saponara
(Department of Information Engineering – University of Pisa)

Evolution of E/E architecture of vehicles



- **Growth of comfort & safety** services but **increase of *Security Vulnerabilities*:**

R155 Regulation

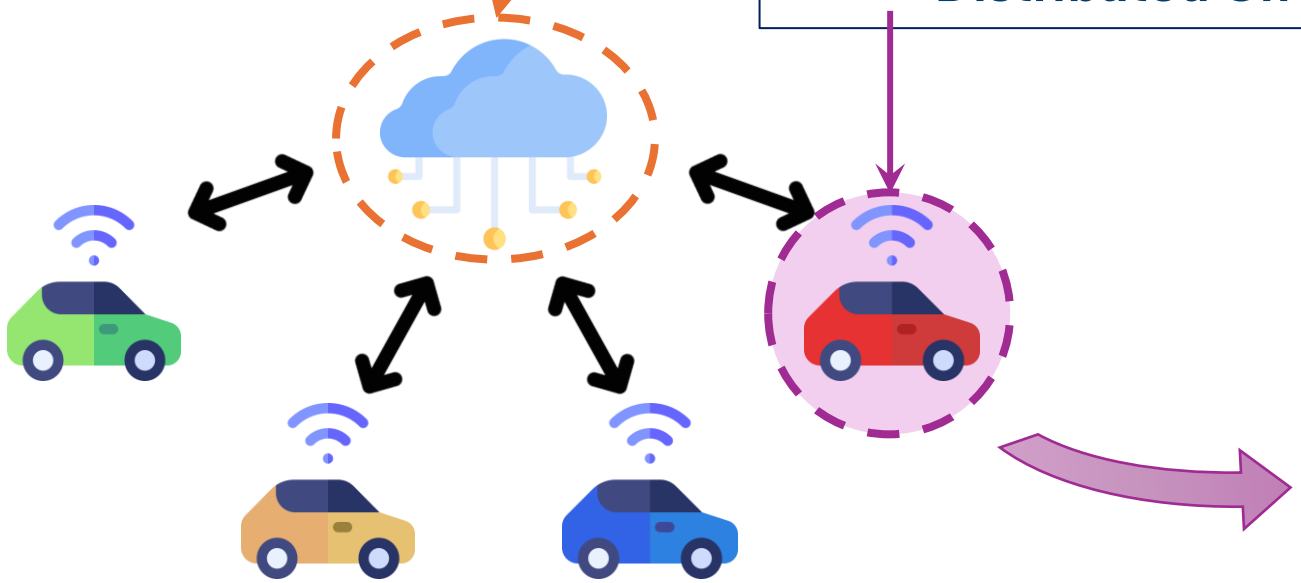
21434 Standard

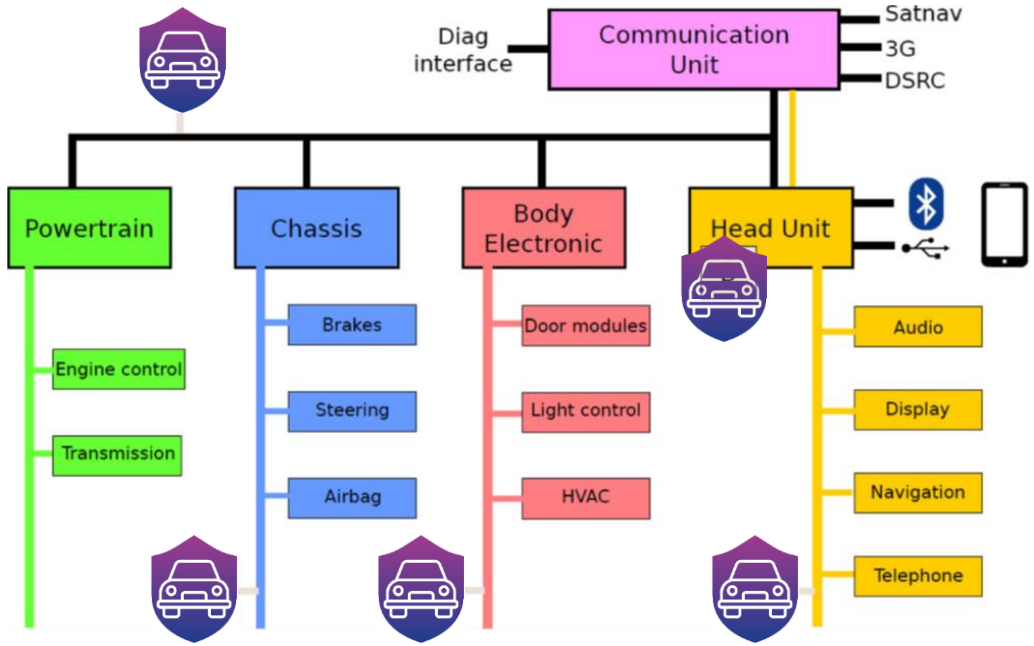
Connected systems are experiencing exponential growth in complexity, making the early detection of anomalies, malfunctions, and cyberattacks increasingly challenging.

ESWS, together **DII (UNI_PI), developed "Argus"** , recently **tested** with **EGICON** support, an innovative anomaly detection and cybersecurity platform designed to address this challenge through a distributed architecture that combines embedded on-edge monitoring modules with advanced Cloud-based Artificial Intelligence.

***Argus* concept combines two monitoring levels:**

- **Cloud-based AI engine** *(under dev.)*
- **Distributed On-edge Monitors (IDS)**





Physical/ Wireless Attack surface → Direct/Indirect-ECU Attack Vector → Altered characteristics of IVN traffic

Edge IDS Monitor

Deployment of IDS on Edge Nodes must satisfy:

Requirements

- **High accuracy**, with low False-Positives rate
- **High coverage of attack** categories
- **Low detection latency** of attack conditions
- **Detection** of **unknown attack** patterns
- **Frame-by-Frame monitoring**

Technical Challenges

- **ECU's constrained memory**
- **ECU's constrained computation capabilities**
- **Reduced IDS computation time to not loose CAN frames**

Preliminary analysis:

- Definition of **physical test** infrastructure requirements
- Definition of **CAN traffic dataset** characteristics
- Generation of **realistic CAN traffic** (DBC, unique IDs, timing periodicity, etc.) on physical infrastructure

Characterization of IDS modules for generic CAN traffic:

- **Offline**: training and testing of Machine Learning models in the Matlab environment
- **Run-Time:** porting of IDS modules onto the NXP S32K344 connected to the test infrastructure

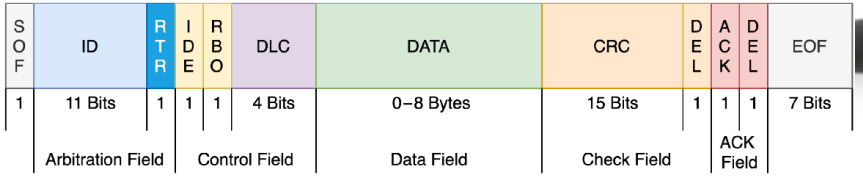
Characterization of IDS modules for specific Use-Cases:

- **Sensor** signals: physical correlations vs. ML models
- **Infotainment signals**: ML models

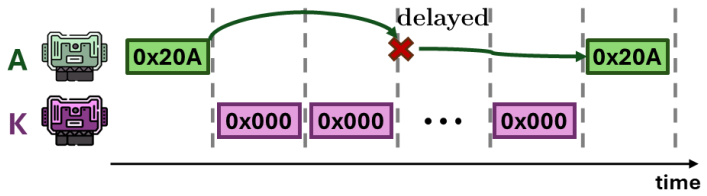
CAN Cybersecurity Vulnerabilities

CAN Protocol vulnerabilities

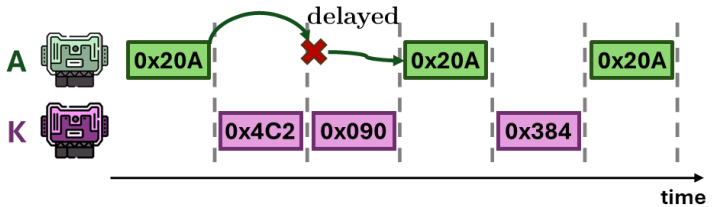
1. NO data encryption
2. NO source information
3. NO message or node authentication



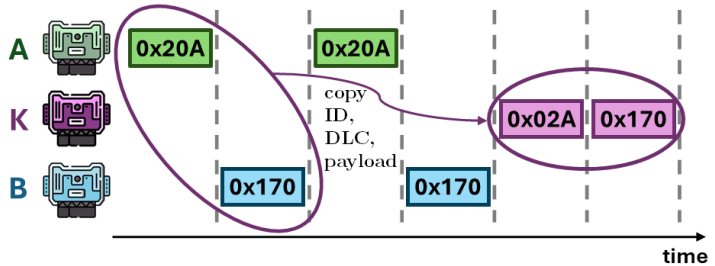
CAN Attack Categories



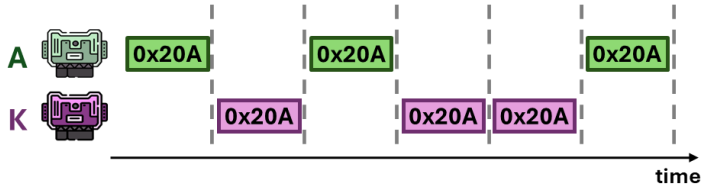
(a) DoS Attack



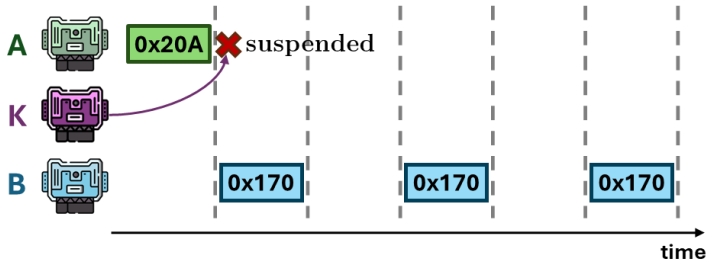
(b) Fuzzy Attack



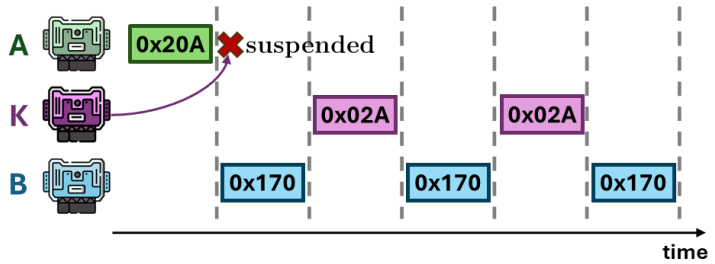
(c) Replay Attack



(d) Spoof Attack



(e) Suspension Attack



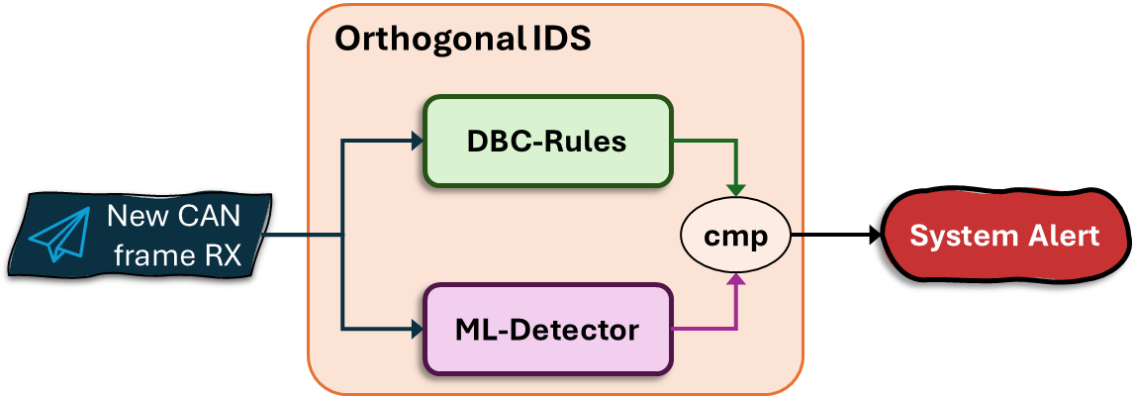
(f) Masquerade Attack

IDS architecture “frame-by-frame”

- **DBC-rules** → Given the DBC of the monitored CAN network, each CAN frame is checked for its ID (whether it is in the whitelist), and its inter-arrival time is verified to be within ±20% of the nominal period.

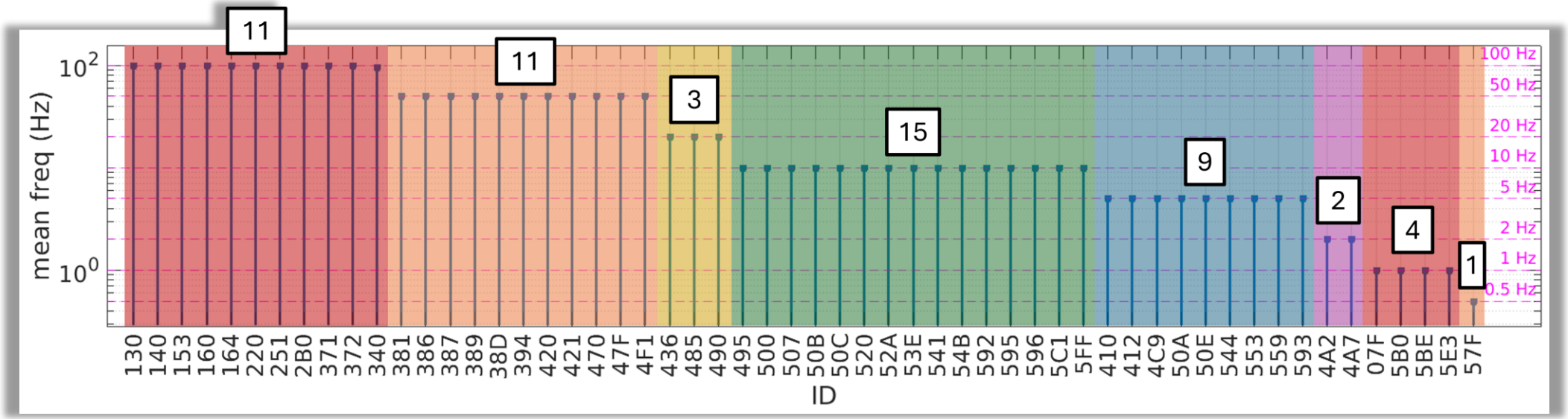
- **ML-based** → Based on a classifier (ML model) that uses inter-arrival time, ID, and payload of received CAN frames to determine whether a frame is benign or malicious.

- **Frame-by- frame** monitoring and response, in order to have the result of the analysis of the past frame before the arrival of the next one.

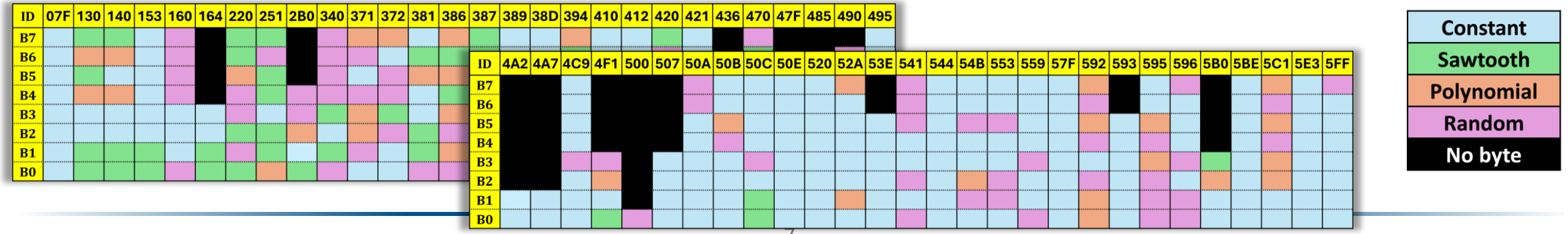


Information on data-link & application CAN layers

- Whitelist of benign ID
- Each ID has a nominal period

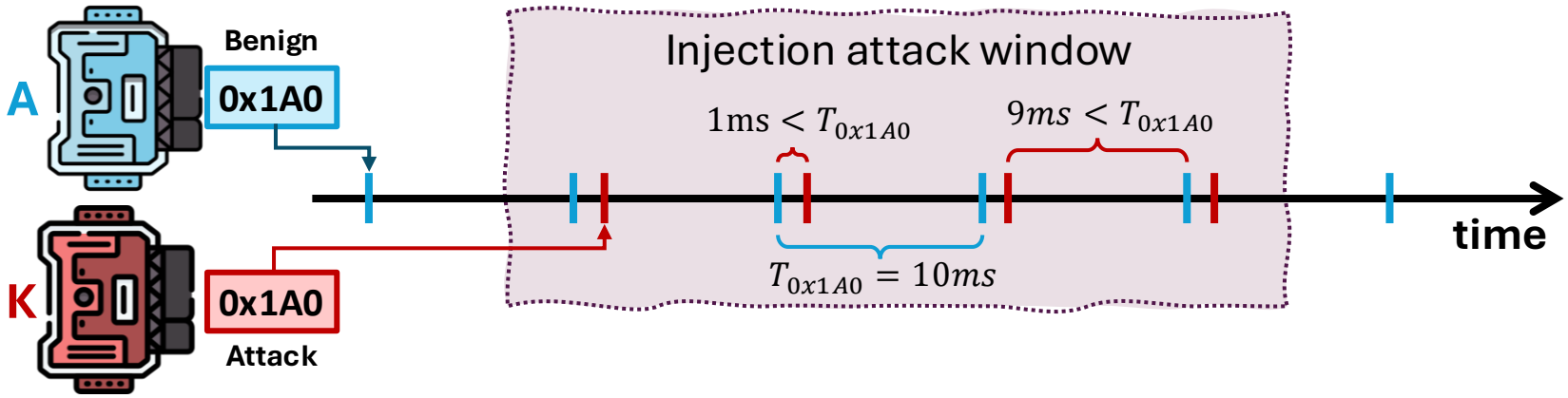


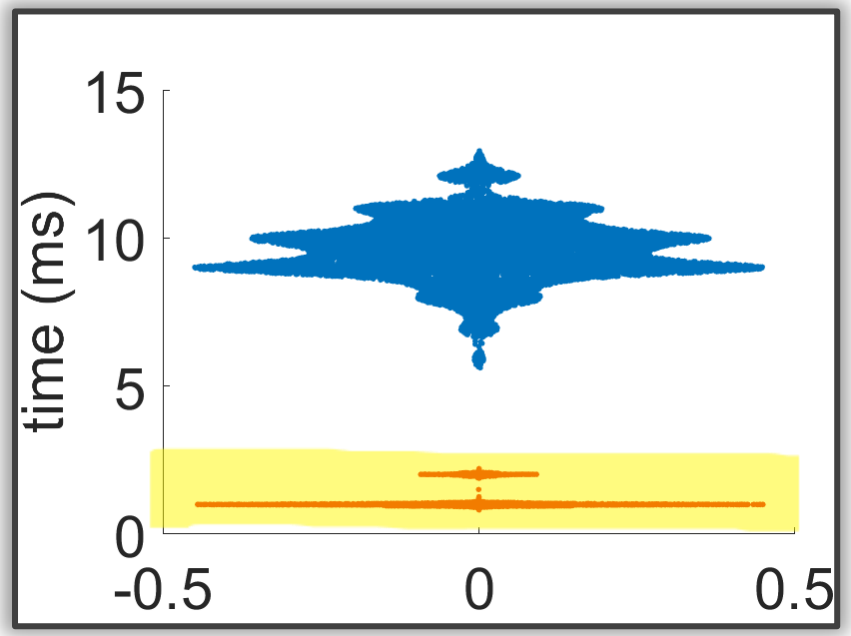
- Payload data fields with characteristic patterns vs. time
- Payload data of fixed length (DLC)



Effects of an injection attack to CAN traffic

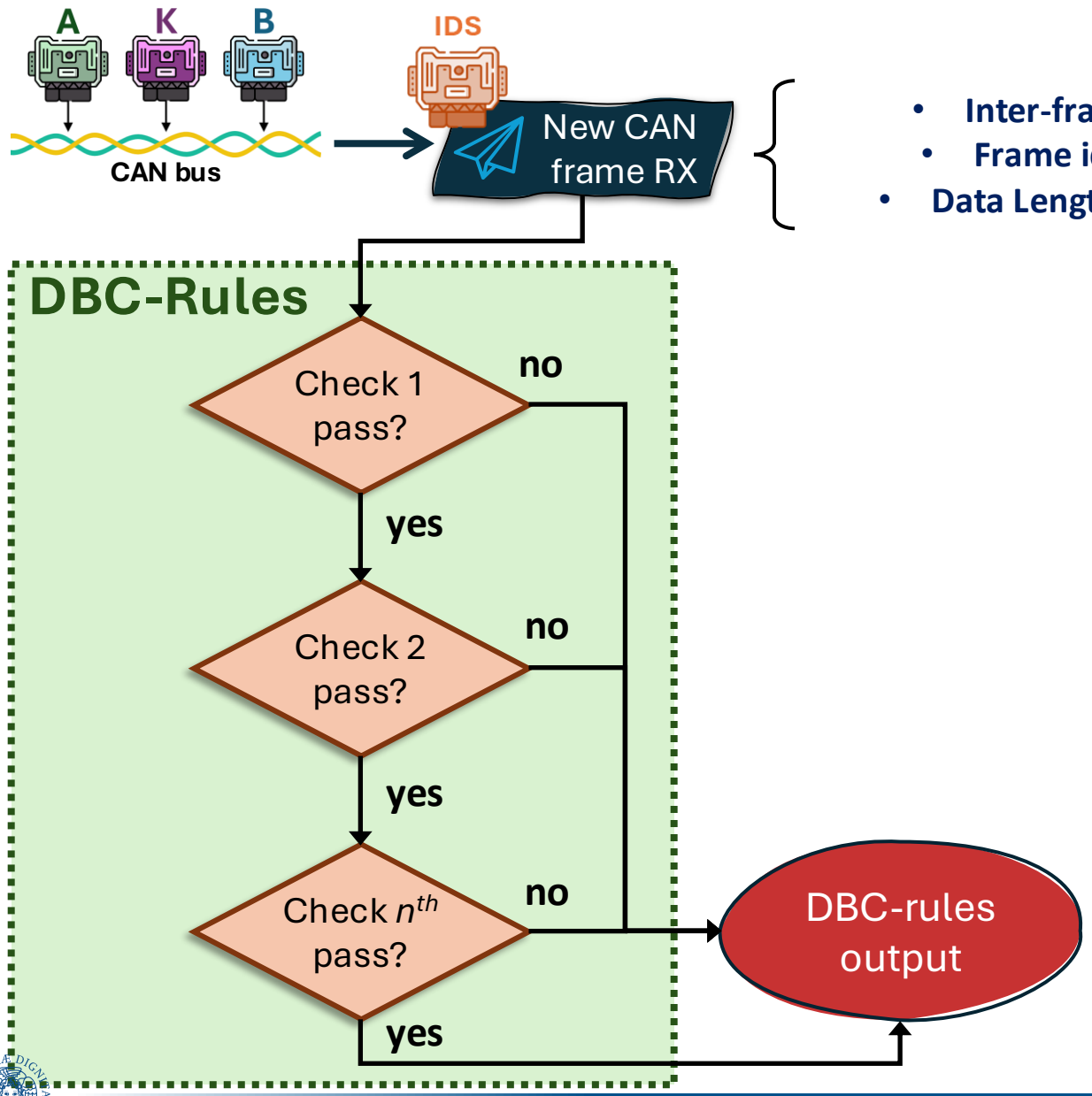
➢ **Injection Attack**





Real distribution of inter-frame times for the attacked ID**:**

DBC-Rules Method



- **Inter-frame time (δt)**
- **Frame identifier (ID)**
- **Data Length Code (DLC)**

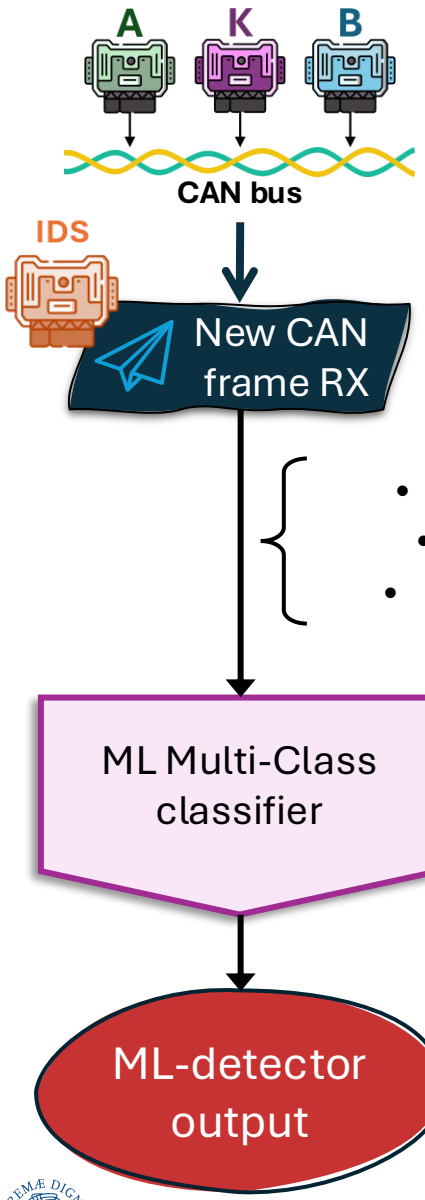
Pros →

- Extremely lightweight method
- Explainable & deterministic

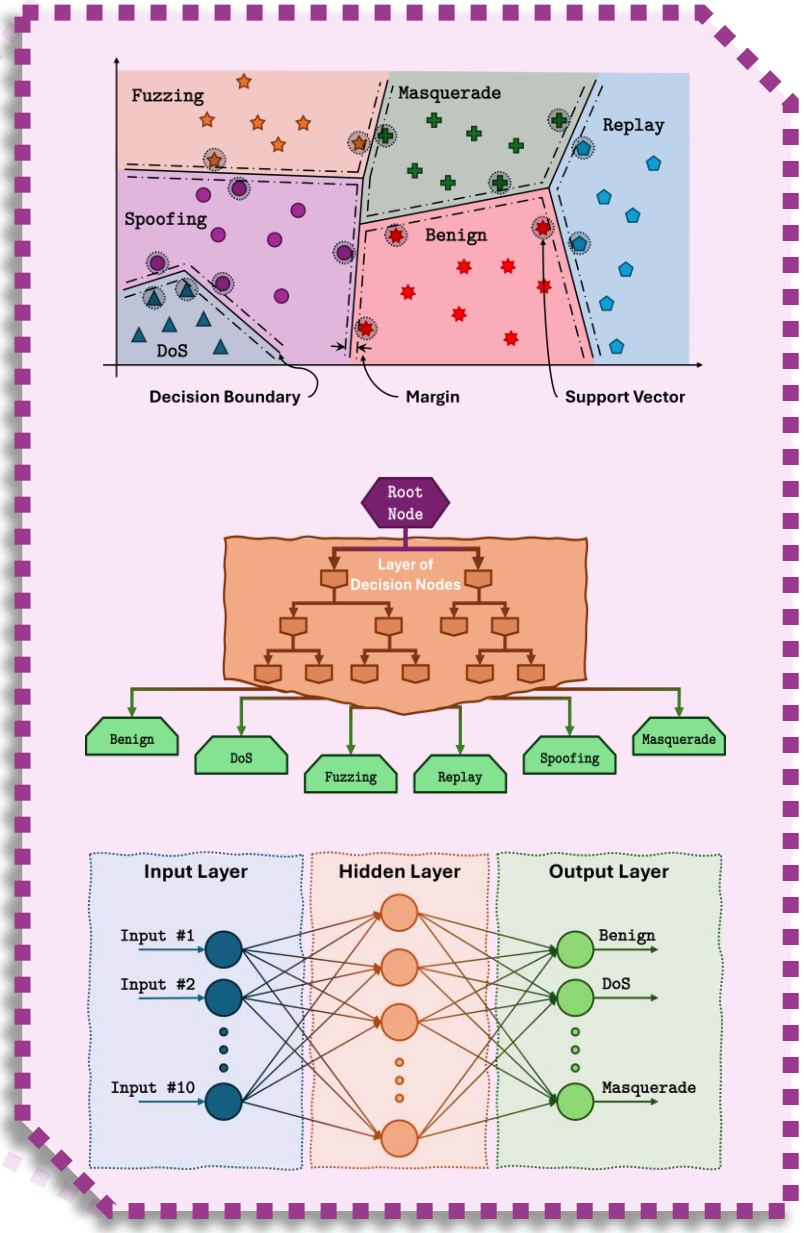
Cons →

- DBC or reverse engineering required
- Limited to injection attacks
- Cannot detect Masquerade

ML-based Method



- **Inter-Frame Time (δt)**
- **Frame Identifier (ID)**
- **8-Byte Payload (DATA)**



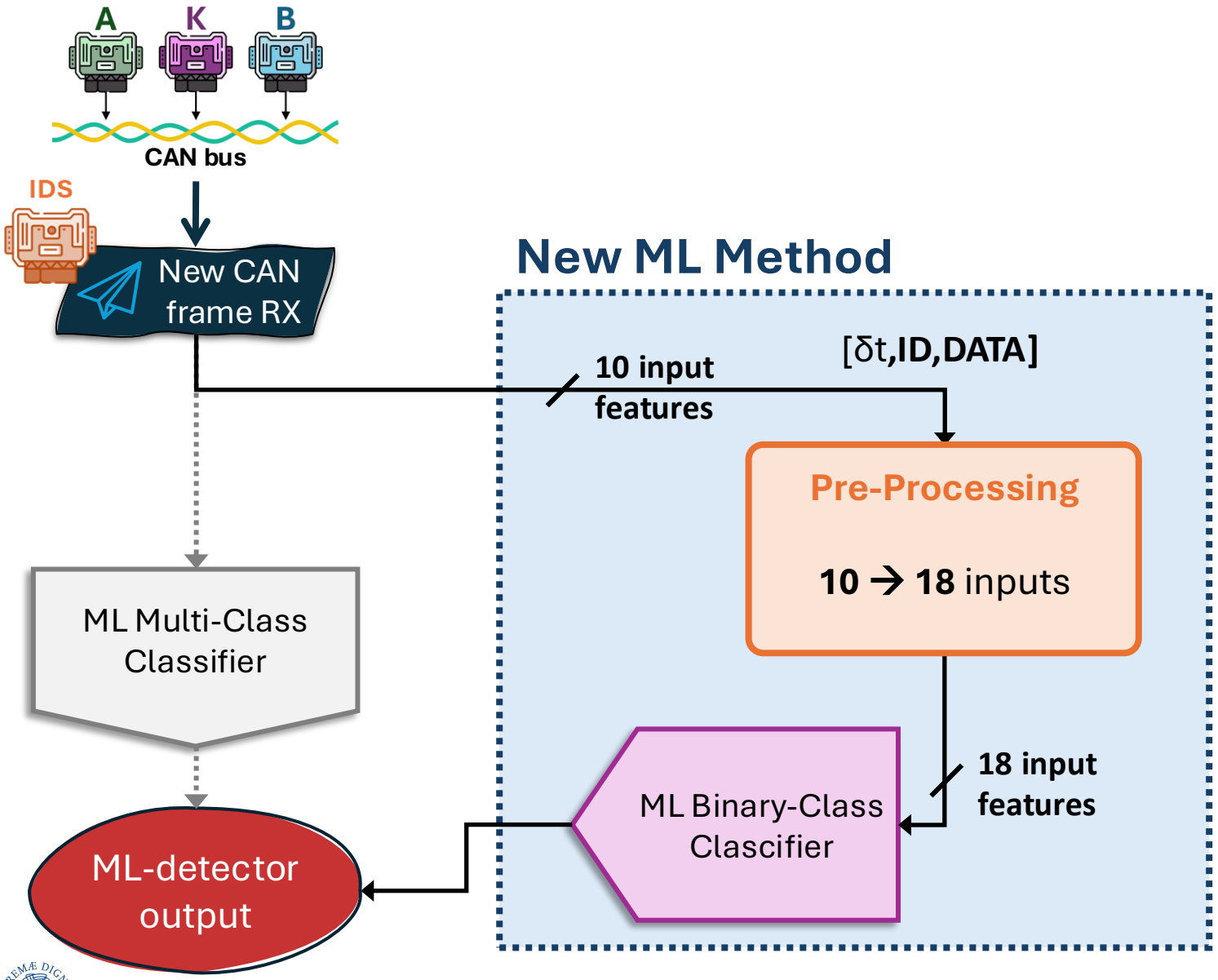
Pros →

- Classification of each attack category
- Lightweight zero pre-processing

Cons →

- High FP in real CAN bus
- Issue with real DoS and Fuzzing attacks
- Vulnerable to small δt & ID shifts

Adjusted version – ML-based Method



Pros →

- Low FP in real CAN bus
- High accuracy with all attack categories
- Robust to δt & ID shifts

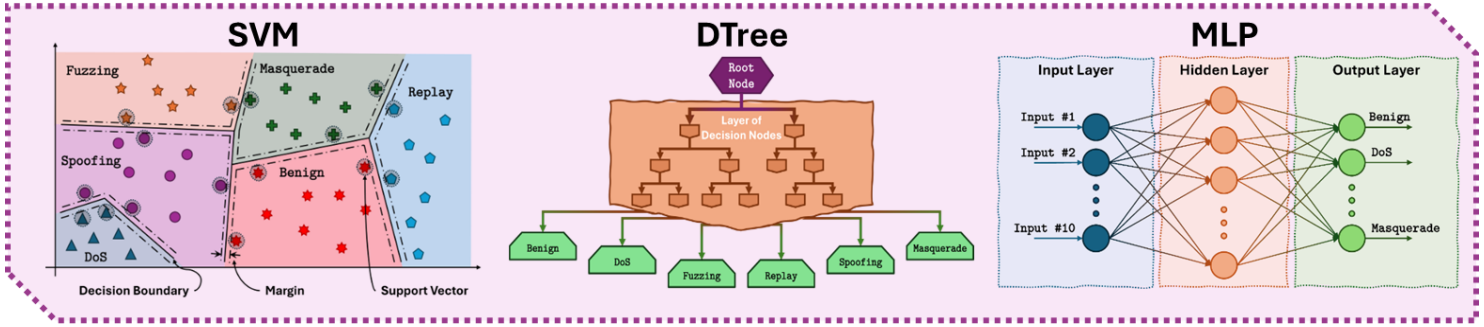
Cons →

- Binary output attack/benign
- On-Board required pre-processing
- Requires minimal tweak before training

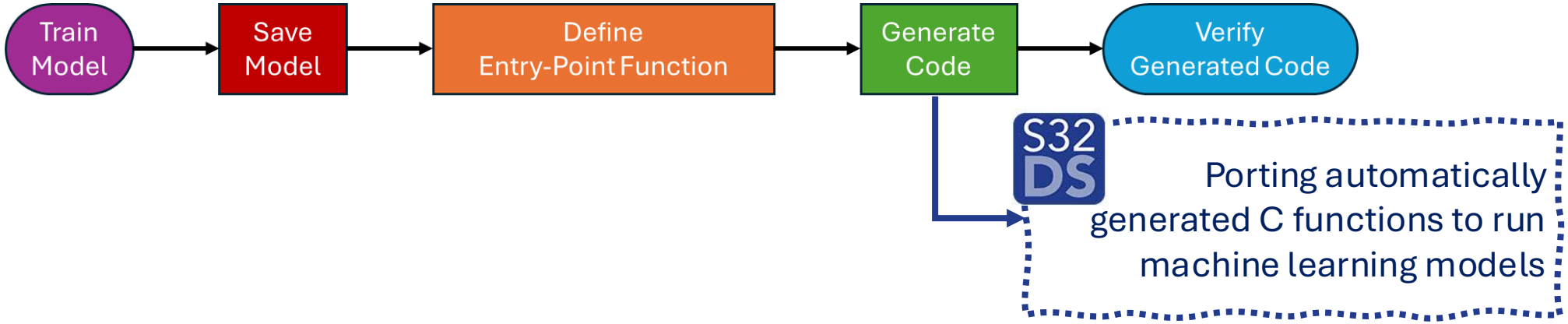
IDS for CAN traffic → Offline validation

Matlab Workflow:

- the selected ML classifier models were trained, validated and tested using the dataset considered.
- A binary classifier (benign/attack) was chosen instead of a multi-class one. The type of attack was not important, only its presence at this stage.

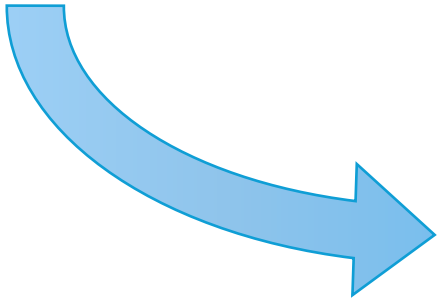


Given the selected ML models, Matlab C Coder was used to auto-generate the C code for integration into S32DS



Results – ML-based method

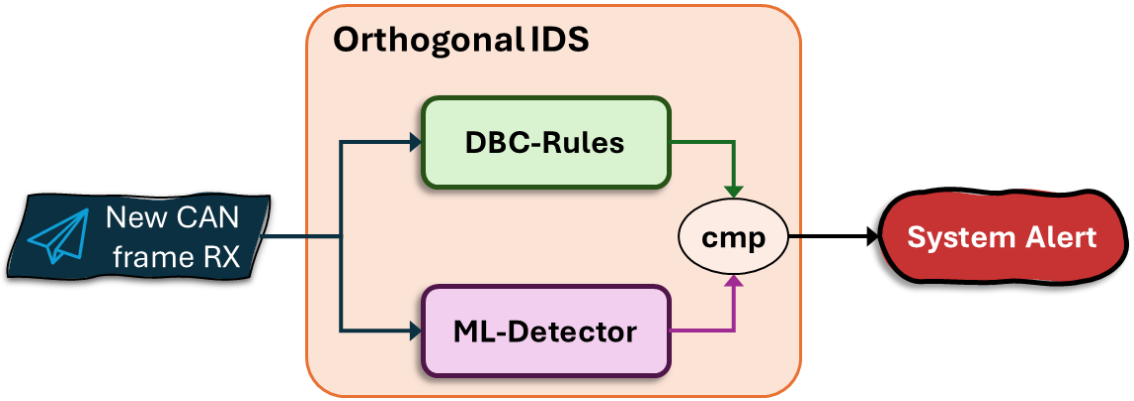
ML classifier	Accuracy [%]	Classification time		Memory Footprint
		μT [μs]	σT [μs]	
Decision Tree	99,71	301,6	6,6	31,7
MLP	99,76	64,5	6,1	3,6
Linear SVM	87,05	66,8	8,1	1,8
Polynomial SVM	99,3	353,0	61,1	169,1
Random Forest	99,78	6466,5	43,8	822,5



MLP resulted to be the best option from preliminary off-line results

"Orthogonal" Multi-Method IDS

Frame-by-Frame continuous **monitoring** of CAN network with **two methods:**



- **DBC-Rules** →
  - Requires DBC file of that network
  - Low false positives
  - Deterministic output
- **ML-Detector** →
  - Boolean output
  - Effective for sophisticated attacks
  - No need of deep knowledge of monitored network

Pros →

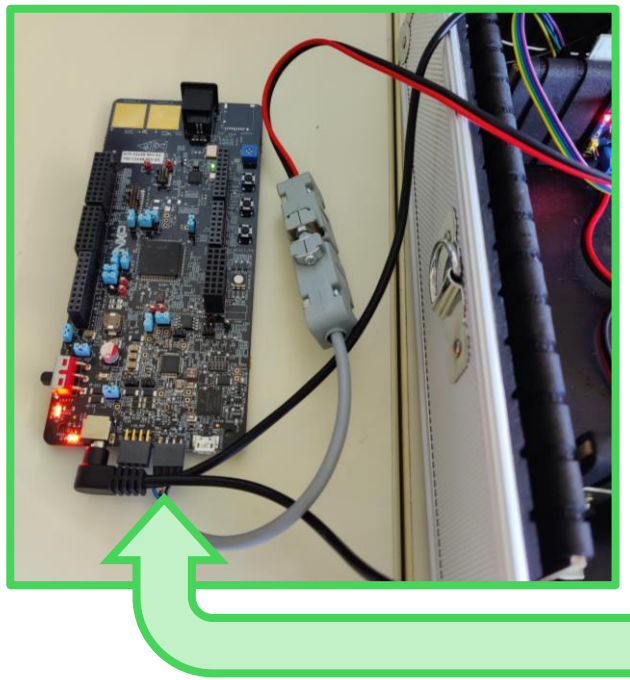
- High accuracy for both simple and sophisticated attacks

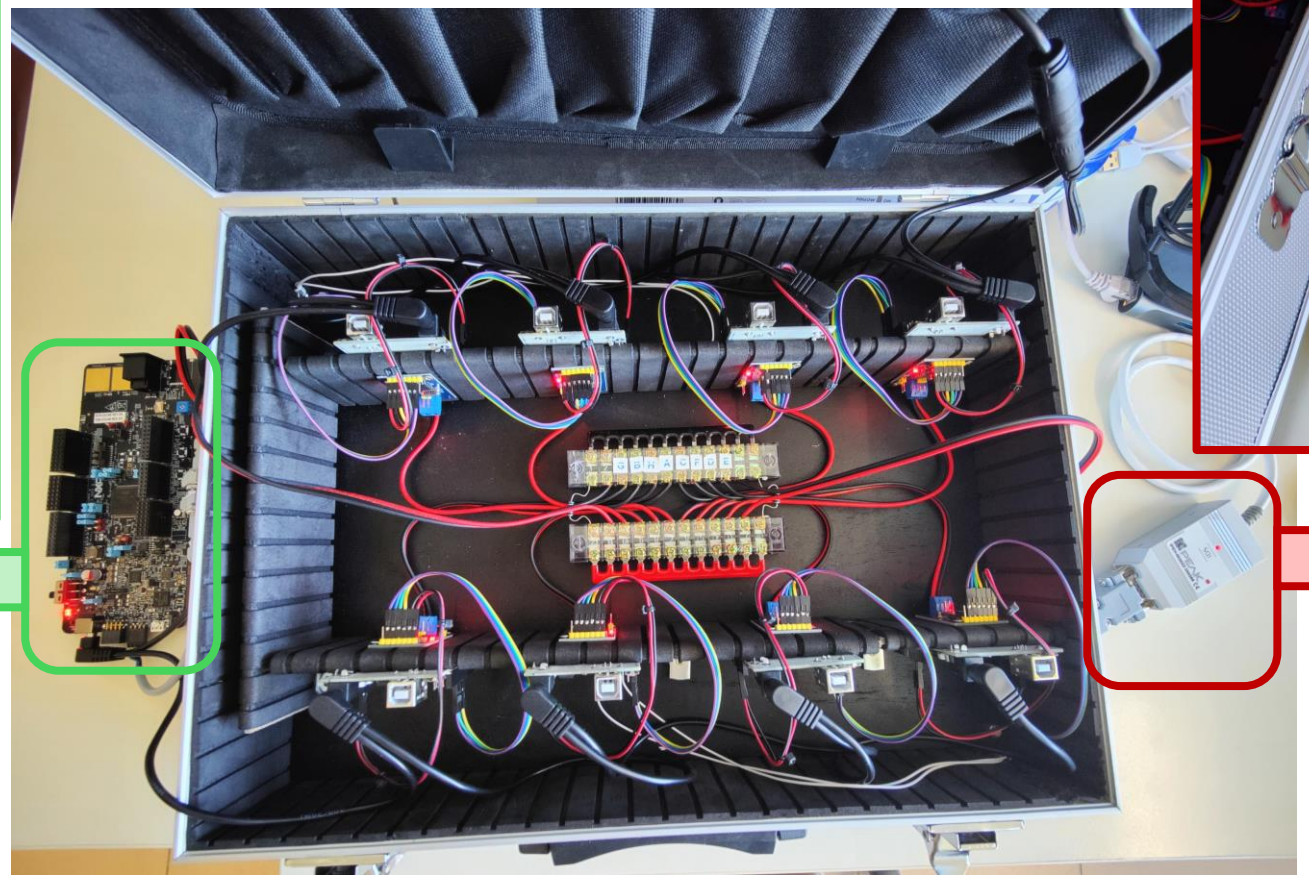
Cons →

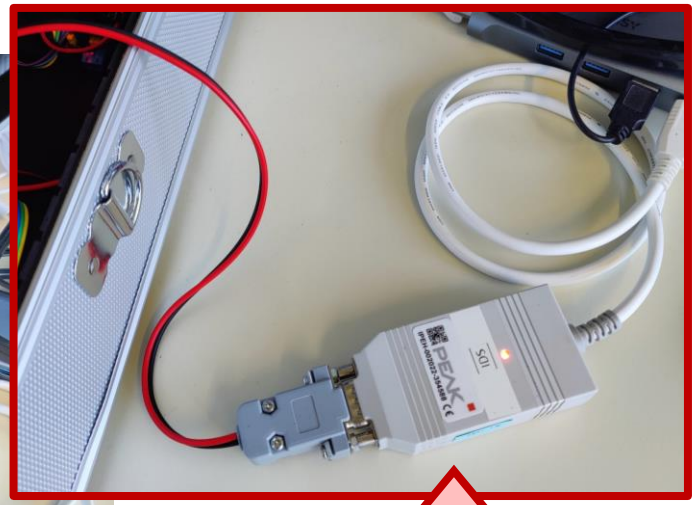
- Higher memory & computation cost (combination of each)
- Different output granularity to be combined

Proto test setup – Orthogonal Multi-Method IDS

EGICON Reliable engineering was interested in a proof of concept based on NXPS32K3







IDS Node

- µC NXP S32K3
- Arm® Cortex-M7
- Clock 160MHz

Attacker Node

- PCAN-USB
- USB to DB9 interface
- CAN 2.0A and 2.0B

CAN network (8 benign nodes)

- Arduino® Uno-R3
- MCP2515 CAN module

Test setup – CANinoApp

Windows application (Python + Qt) to control PCAN-USB:

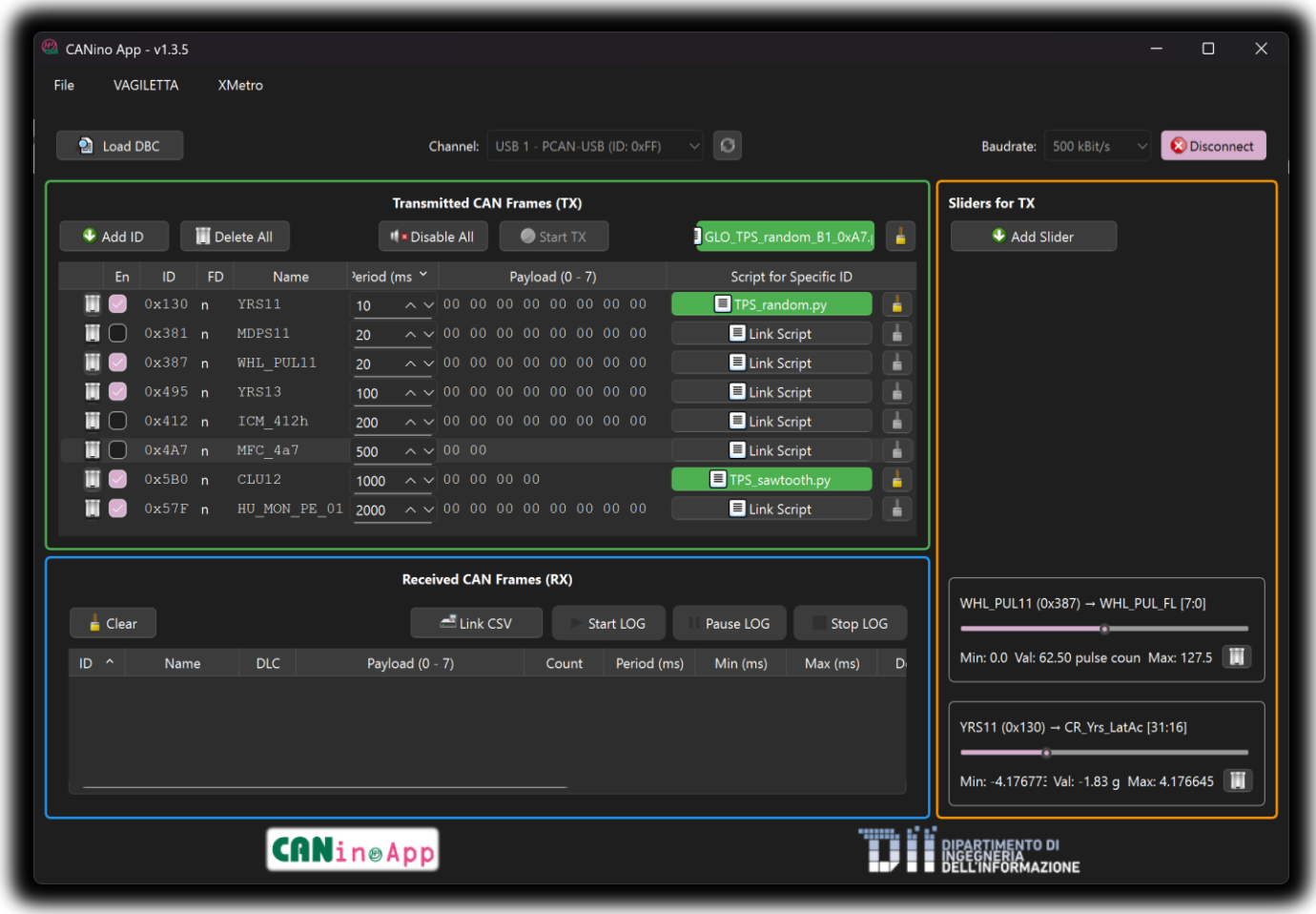
✓ https://github.com/NickCanino/CANino_app.git

✓ **The app allows to:**

- Transmit custom CAN messages with custom payloads.
- Receive and view CAN messages info and statistics in real time.
- Load DBC files for automatic traffic decoding.
- Connect custom Python scripts for dynamic payload generation.
- Export logs of received messages in CSV format.

- Display gauges with DBC decoded signals

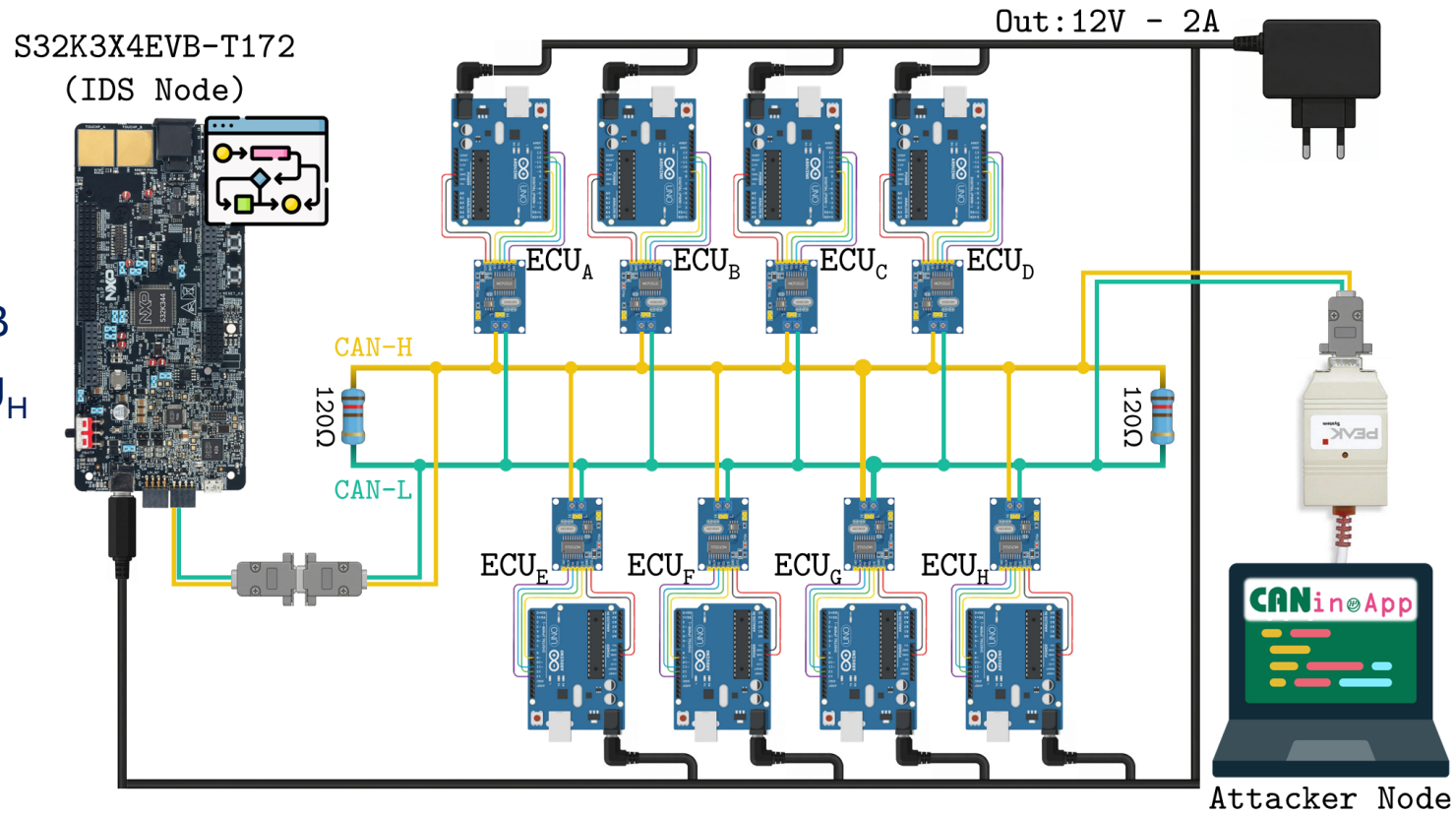
- Flash Arduino C code to configure benign nodes



Proto test setup – Schematical view

Schematic of the proto test setup:

- **IDS Node** → NXP S32K3X4EVB-172
- **Attacker Node** → CANinoApp via PCAN-USB
- **Benign ECUs** → 8 nodes from ECU_A to ECU_H
- **Common CAN bus**
- **Common power source**



Proto test: KPI of DBC-Rules + ML-based Methods

Injection attack on whitelisted ID → **ALTERS inter-time**

F-by-F check of that ID → **Frame Classified as Attack**

Good Accuracy
Perfect Recall

Scenario	Attack Frames (P_{tot})	ML-Detector				DBC-Rules				Full IDS Scores [%]	
		TN	FN	TP	FP	OK	ID_{bad}	T_{bad}	DBC_{bad}	Accuracy	Recall
Baseline (No Attack)	0	48 836	0	0	1164	49 226	0	774	0	97.63	–
#1 ↦ DoS, Fuzzing BB	4069	44 476	0	4069	1455	45 076	4069	855	0	97.00	100
#2 ↦ Fuzzing GB Full	9263	34 304	740	8523	6433	34 404	0	11 502	4094	86.48	100
#3 ↦ Fuzzing GB Half	10 564	32 750	460	10 104	6686	33 059	0	16 941	0	86.34	100
#4 ↦ Replay, Spoofing	7192	37 663	435	6757	5145	37 933	0	12 067	0	89.18	100
#5 ↦ Masquerade	10 605	37 517	2846	7759	1878	49 390	0	610	0	90.61	73.56

Mean Accuracy:

- **Only benign → ~97,6%**
- **All attacks → ~89,9%**

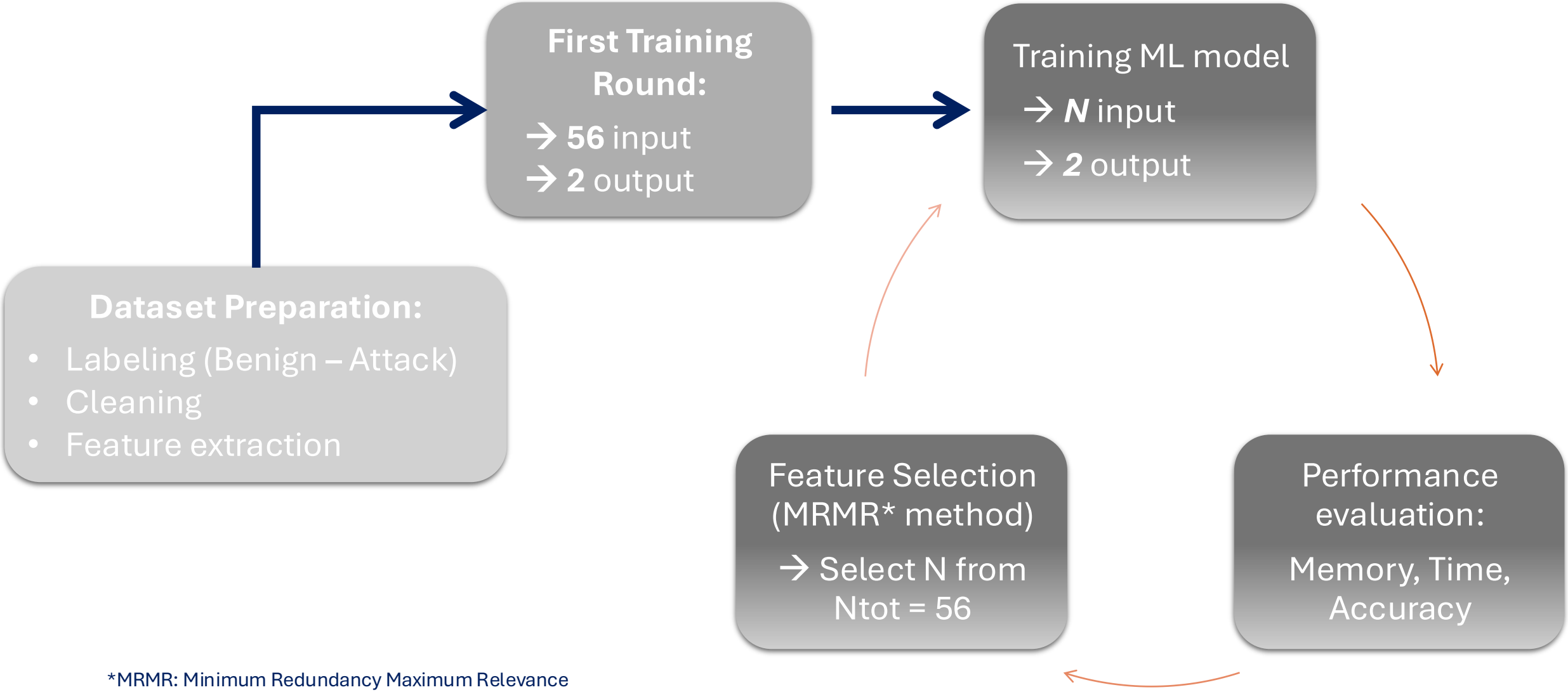
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

$$Accuracy = \frac{Correct\ classification}{All\ frames}$$

$$Recall = \frac{True\ Attack}{All\ Attack\ frames}$$

Infotainment Dataset – ML Classifier



*MRMR: Minimum Redundancy Maximum Relevance

Infotainment Dataset – Results over MRMR Iterations

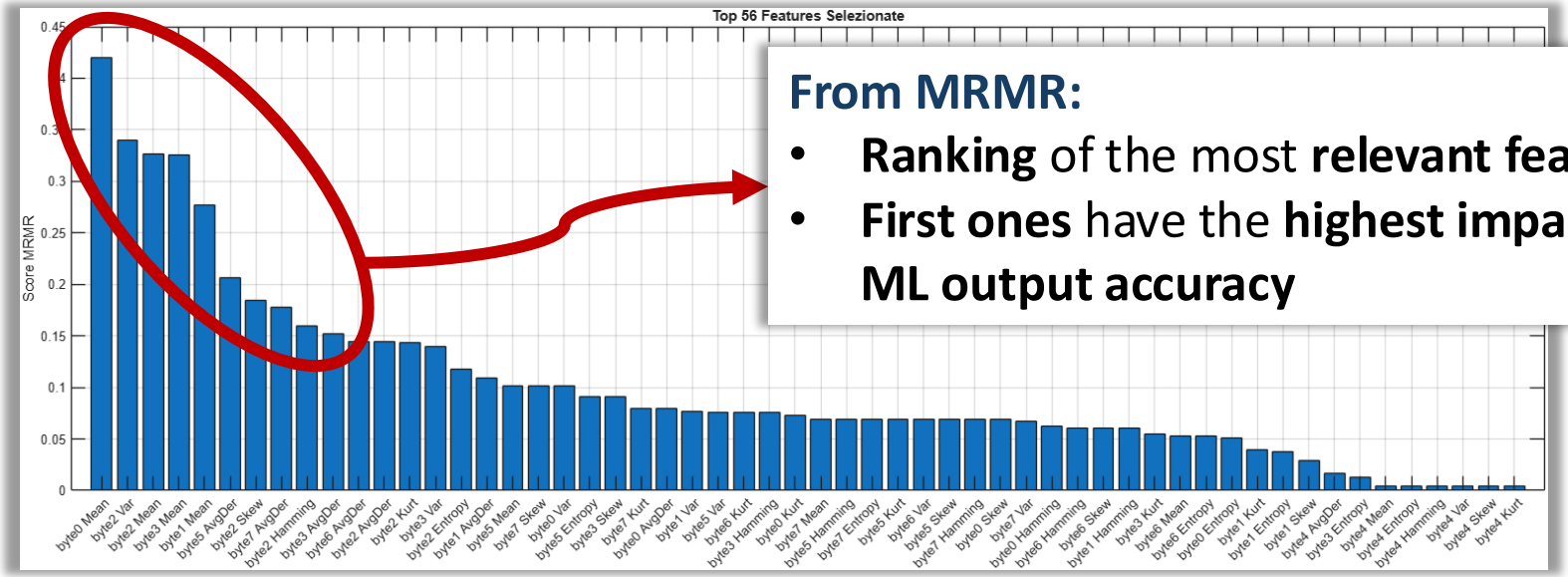
Pros →

- Reduced Pre-Processing → less inputs to prepare
- Reduced Memory Footprint → smaller ML model
- Reduced Computation Time → smaller ML model

Cons →

- Reduced Model Accuracy → less inputs to be used to produce output

<Matlab>	56 input	30 input	20 input	10 input	Trend
Accuracy	96,9%	95,0%	94,1%	93,8%	↓
Memory Footprint	35 kB	30 kB	27 kB	22 kB	↓



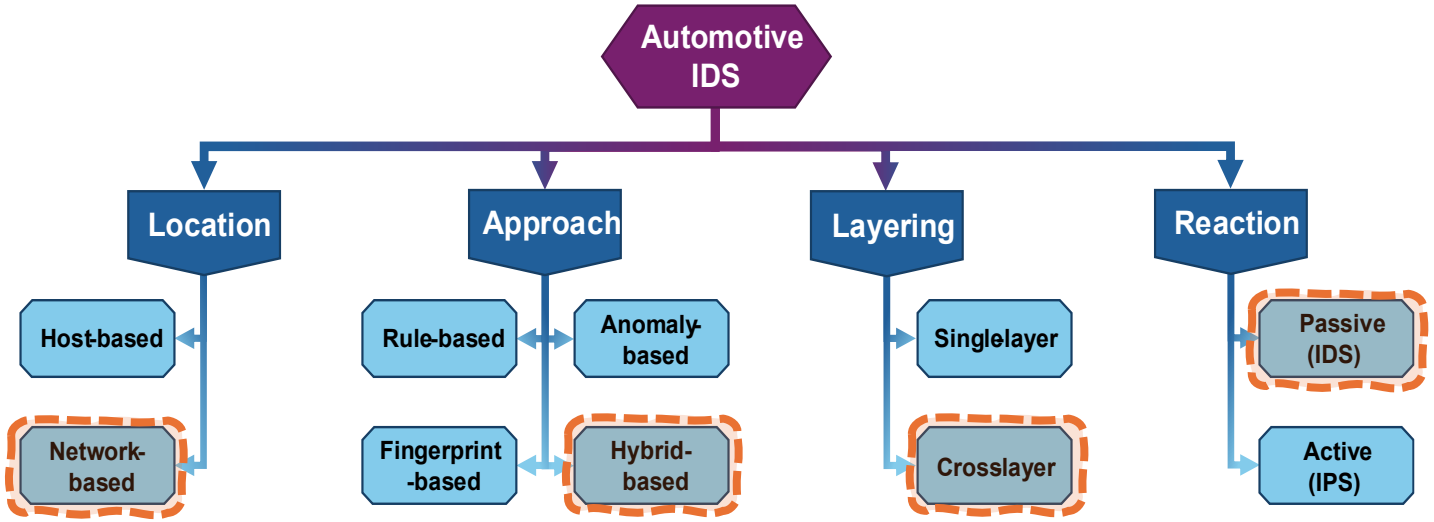
From MRMR:

- **Ranking** of the most **relevant features**
- **First ones** have the **highest impact** on **ML output accuracy**

Conclusion

Characteristics of developed IDS deployed in On-Edge Monitor:

- **Location** → Network-based
- **Type** → Hybrid-based (Rule + Anomaly)
- **Layering** → Physical & Data-Link ISO/OSI layers
- **Reaction** → Passive



Metric	Value
Accuracy	~90%
Computation Time	~38,8 µs
Memory Footprint	~3,2 KB

Results further optimizable for:

- **Accuracy, or**
- **Time & Memory**

Conclusion

Originally conceived for the automotive sector, one of the most demanding environments due to complex ECU networks, embedded software, and critical digital functions, ***Argus*** learns the normal operational behavior of a machine and identifies deviations using advanced Machine Learning techniques.

This enables the **early detection of abnormal patterns**, **intrusions** and system **irregularities**, including previously unknown threats.

Rather than directly managing containment actions, ***Argus*** **focuses on rapid recognition and situational awareness**, while **response strategies remain delegated to the host vehicle** or machine control systems **according to operational context**.

embedded software systems

Università di Pisa

Dipartimento di Ingegneria dell'Informazione

Thank you !