NATWORK

Net-Zero self-adaptive activation of distributed self-resilient augmented services

D5.2 Energy efficient AI-based security processes

Lead beneficiary	ELTE	Lead author	Péter Vörös
Reviewers	Gürkan Gür (ZHAW), Roberto González (NEC)		
Type	R	Dissemination	PU
Document version	V1.0	Due date	31/12/2025

Co-funded by the European Union

6GSNS

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

UK Research and Innovation

Project information

Project title	Net-Zero self-adaptive activation of distributed self-resilient augmented services
Project acronym	NATWORK
Grant Agreement No	101139285
Type of action	HORIZON JU Research and Innovation Actions
Call	HORIZON-JU-SNS-2023
Topic	HORIZON-JU-SNS-2023-STREAM-B-01-04 Reliable Services and Smart Security
Start date	01/01/2024
Duration	36 months

Document information

Associated WP	WP5
Associated task(s)	T5.2
Main Author(s)	Péter Vörös (ELTE), Gergely Sárközi (ELTE)
Author(s)	Brais Ares, Adriana Febles (GRAD), Francesco Paolucci, Michelangelo Guaitolini, Abdul H. Khan (CNIT), Nasim Nezhadsistani (UZH), Virgilios Passas, Nikolaos Makris, Donatos Stavropoulos, Dimitrios Manolakis, Antonios Lalas, Anastasios Drosou (CERTH)
Reviewers	Gürkan Gür (ZHAW), Roberto González (NEC)
Type	R — Document, report
Dissemination level	PU — Public
Due date	M24 (31/12/2025)
Submission date	31/12/2025

Co-funded by the European Union
Project funded by
UK Research and Innovation

Document version history

Version	Date	Changes	Contributor (s)
v0.1	15/07/2025	Initial table of contents	Péter Vörös (ELTE)
v0.2	15/09/2025	First drafts added	Gergely Sárközi, Péter Vörös (ELTE)
v0.3	08/10/2025	Contributions for sections 4.1, 4.2, 4.4	Adriana Febles (GRAD)
v0.4	15/10/2025	Contributions for section 4.3	Brais Ares (GRAD)
v0.4.5	31/10/2025	Section drafts ready	All authors
v0.5	05/11/2025	Section 0: Minor fixes and updated acronym list	Brais Ares (GRAD)
v0.5.5	11/11/2025	Introduction, conclusion added	Péter Vörös (ELTE)
V0.6	17/11/2025	Corrections before internal review	All authors
v0.6.5	18/11/2025	Version ready for review	Péter Vörös (ELTE)
v0.7	05/12/2025	Document review	Gürkan Gür (ZHAW), Roberto González (NEC)
v0.7.5	05/12/2025	Corrections after internal review	Péter Vörös (ELTE)
v0.8	05/12/2025	Refinement based on review comments	All authors
v0.8.5	15/12/2025	Quality check	Anna Padee (HES-SO)
v0.9	16/12/2025	Addressing quality check comments	Péter Vörös (ELTE)
v0.9.5	23/12/2025	Final review and refinements	Antonios Lalas (CERTH) and CERTH team
v1.0	31/12/2025	Final version for submission	Antonios Lalas (CERTH)

Disclaimer

Copyright message

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

Contents

List of acronyms and abbreviations 8

List of figures 10

List of tables 11

Executive summary 12

1. Introduction 13

1.1 Purpose and structure of the document 13

1.2 Intended Audience 14

1.3 Interrelations 14

2. State of the art 16

2.1 Disaggregated Net-Zero machine learning 16

2.2 Jamming detection algorithms on MPSoC platforms 17

2.2.1 Acceleration frameworks 18

2.3 Energy efficient and latency-aware workload scheduling 18

2.4 Energy-efficient RDMA transfer 19

2.5 Net-Zero machine learning for anomaly detection 20

3 Disaggregated Net-Zero machine learning 23

3.1 Proposed model and system architecture 23

3.1.1 Model disaggregation and utilization of previously unused computation capacities 24

3.1.2 Federated learning and support for network slices 25

3.2 Implementation 26

3.2.1 Extraction of flow features 26

3.2.2 Model training and as-soon-as-possible inference 26

3.2.3 Achieving seamless model switchovers 27

3.3 Preliminary results 28

3.3.1 Optimal ratio of monitored flows 31

3.3.2 Time required for model deployment 31

4 Jamming detection algorithms on MPSoC platforms 33

4.1 System architecture 33
4.2 Proposed model and framework 34
4.3 Implementation 34
4.3.1 Firmware 34
4.3.2 Vitis AI workflow 38
4.4 Preliminary results 40
5 Energy efficient and Latency-aware workload scheduling 42
5.1 Proposed model and system architecture 42
5.2 Implementation 44
5.2.1 Scaphandre 44
5.2.2 Latency Module 44
5.2.3 Workload Energy Consumption Predictor 45
5.2.4 Custom Kubernetes scheduler based on RL-agent 45
5.3 Preliminary results 47
6 Energy-efficient RDMA transfer 48
6.1 Proposed model and system architecture 48
6.2 Implementation 49
6.3 Preliminary results 49
7 Net-Zero machine learning for anomaly detection 53
7.1 Proposed model and system architecture 53
7.2 Implementation 53
7.2.1 Impact of DDoS and Jamming Attacks on Energy Consumption 55
7.2.2 Motivated design and energy model for XGBoost-on-SmartNIC (DDoS) and CSI-driven near-RT RIC xApp (Jamming) 56
7.2.3 Conclusion (net energy savings with detection and blocking) 58
7.3 Preliminary results 58
8 Conclusions 61
References 63

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
UK Research and Innovation

List of acronyms and abbreviations

Abbreviation	Description
AI	Artificial Intelligence
ALU	Arithmetic Logic Unit
API	Application Programming Interface
ARM	Advanced RISC Machines
AXI	Advanced eXtensible Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DDoS	Distributed Denial-of-Service
DDR	Double Data Rate
DL	Deep Learning
DPU	Deep Learning Processing Unit
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
E-RTA	Energy-Aware RDMA Transfer Agent
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
FPS	Frames Per Second
GPU	Graphics Processing Unit
I/O	Input/Output
IDS	Intrusion Detection System
ILP	Integer Linear Programming
IP	Intellectual Property
IRQ	Interrupt Request
KNN	K-Nearest Neighbors
ML	Machine Learning
MPSoC	Multi-Processor System-on-Chip
NAS	Neural Architecture Search
PTQ	Post-Training Quantization
QEMU	Quick Emulator
RDMA	Remote Direct Memory Access
RF	Random Forest
RL	Reinforcement Learning
RoCE	RDMA over Converged Ethernet
SLCR	System Level Configuration Registers
SoC	System-on-Chip
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VART	Vitis AI Runtime Library

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

Abbreviation	Description
XIR	AMD Intermediate Representation

List of figures

Figure 1: An overview of the proposed model 24

Figure 2: An example of disaggregated model deployment. The goal of the disaggregation is to fit the classification model into the unused segments of existing networking infrastructure. ... 25

Figure 3: Our proposed setup supporting multiple network slices using federated learning. The confidential raw flow data does not cross network slice boundaries. 26

Figure 4: Accuracy of in-network DDoS attack detection over time 30

Figure 5: Accuracy of in-network portscan detection over time 30

Figure 6: Accuracy of in-network detection of concurrent DDoS and portscan attacks 31

Figure 7: System architecture 33

Figure 8: Model architecture 34

Figure 9: Overview of FPGA design 35

Figure 10: Utilization summary of FPGA resources (left: device layout; right: usage percentages) 36

Figure 11: Block diagram of all pieces needed to run an AI model in a SoC 37

Figure 12: Optimizer methodology 38

Figure 13: Workflow comparison of Iterative Pruning, One-Step Pruning and One-Shot NAS methods 39

Figure 14: Quantization and compilation workflow 40

Figure 15: ELAWS system architecture 43

Figure 16: Architecture of the RL agent for energy and latency-aware workload placement decisions........ 43

Figure 17: Scaphandre deployment and interconnection in Kubernetes Cluster 44

Figure 18: Experimental results for node energy consumption 47

Figure 19 E-RTA functional architecture and interfaces 49

Figure 20 E-RTA Testbed setup 50

Figure 21 Analysis of energy consumption with TCP and E-RTA 50

Figure 22 Prometheus /Grafana dashboard used to retrieve E-RTA parameters 51

Figure 23: Nebula system workflow 55

List of tables

Table 1: Fraction of monitored flows versus the achieved F1 score 31

Table 2: Combined results of baseline and pruned ResNet18 models on GPU and ZCU102 MPSoC 41

Table 3: Acceleration results of the Jamming Detection on GPU and ZCU102 MPSoC............... 41

Table 4: Energy consumption used by different transfer operations 51

Table 5: Energy consumption by E-RTA with different file dimensions and buffer sizes............ 52

Table 6: Distribution of Energy Consumption and Carbon Emissions per Phase (CIC-IDS2017, 10 nodes, 20 rounds) 59

Table 7: Carbon Intensity and Training Emissions per Node in a 5G-NIDD Scenario 59

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
UK Research and Innovation

Executive summary

The purpose of this deliverable D5.2 “Energy efficient AI-based security processes” is to present the design principles, implementation approaches, and preliminary results of energy-efficient, AI-based security mechanisms developed within the NATWORK project. These mechanisms aim to strengthen the resilience and sustainability of next-generation network infrastructures through intelligent data processing and adaptive learning, aligning with the project’s goal of enabling Net-Zero, secure, and self-adaptive 6G services.

The document introduces multiple complementary approaches for achieving energy-efficient security. It details the development of **disaggregated Net-Zero machine learning models** for in-network anomaly detection, the **acceleration of jamming detection algorithms on Multiprocessor System-on-Chip (MPSoC) platforms**, and **reinforcement learning-based workload scheduling** for optimizing latency and energy consumption across the edge–cloud continuum. Furthermore, it explores **energy-aware RDMA transfer techniques** and **sustainability-driven machine learning** for anomaly detection, showcasing how distributed AI solutions can enhance both performance and energy efficiency.

The deliverable defines the methodologies, architectures, and preliminary experimental validations supporting these innovations. It demonstrates how the integration of AI-assisted data plane intelligence, hardware acceleration, and energy-aware orchestration contributes to achieving sustainable cybersecurity for programmable and distributed 6G infrastructures.

The results of this work establish the foundation for large-scale testing, integration, and cross-layer validation activities planned in upcoming phases of NATWORK, directly supporting the project’s objectives of security, energy efficiency, and autonomous network operation.

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

1. Introduction

The NATWORK D5.2 “Energy Efficient AI-based Security Processes” deliverable presents the progress and results achieved within Task 5.2, “Net-zero ML and intrinsic security of AI,” focusing on the development and validation of AI-driven, energy-efficient security mechanisms designed for next-generation programmable network infrastructures. The increasing complexity of modern networked environments — characterised by rising traffic volumes, highly distributed workloads, and a growing attack surface spanning cloud, edge, and programmable data planes — requires new approaches to cybersecurity that are both adaptive and sustainable. Traditional security mechanisms struggle to keep pace with evolving threats without incurring substantial computational and energy overheads, creating a need for more efficient, intelligent, and context-aware solutions.

This deliverable addresses these challenges by investigating how intrinsic security capabilities can be embedded directly into the network and compute fabric through machine learning–based detection, hardware-accelerated analytics, and energy-aware decision-making. The work aims to demonstrate how intelligent data processing and adaptive learning can enhance cybersecurity while maintaining or reducing energy consumption, thereby contributing to the overall Net-Zero objectives of the NATWORK project. The mechanisms developed in Task 5.2 exploit programmable switches, edge accelerators, and distributed ML pipelines to achieve real-time, low-overhead protection aligned with the project’s broader vision of secure, autonomous, and environmentally responsible network infrastructures.

1.1 Purpose and structure of the document

The purpose of this document is to detail the design principles, implementation approaches, and experimental validation of the developed AI-based security processes. It consolidates the technical contributions of multiple partners toward energy-efficient, intelligent, and sustainable security functions within programmable and distributed network environments.

Following the Introduction, which defines the purpose, audience, and interrelations within the project framework, the structure of the deliverable is as follows:

Sections:

- **Section 2 – State of the Art:** Reviews the current research landscape and technologies relevant to energy-efficient AI-based security mechanisms.
- **Section 3 – Disaggregated Net-Zero Machine Learning:** Describes the proposed distributed AI models for in-network anomaly detection and mitigation.

- **Section 4 – Jamming Detection on MPSoC Platforms:** Details the hardware acceleration of neural network models for energy-efficient signal analysis and security.
- **Section 5 – Energy-efficient and Latency-aware Workload Scheduling:** Presents the design and implementation of adaptive workload scheduling mechanisms combining latency and energy optimization.
- **Section 6 – Energy-efficient RDMA Transfer:** Explains methods for optimizing data transfer processes to reduce energy usage in high-speed network environments.
- **Section 7 – Net-Zero Machine Learning for Anomaly Detection:** Summarizes approaches for sustainable machine learning applied to network security analytics.
- **Section 8 – Conclusions:** Summarizes the key findings and outlines the next steps in advancing the project’s research objectives.

1.2 Intended Audience

The NATWORK D5.2 “Energy Efficient AI-based Security Processes” deliverable is devised for public use in the context of the research, development, and dissemination activities of the NATWORK consortium, comprising members, project partners, and affiliated stakeholders. This document mainly focuses on the design, implementation, and evaluation of AI-based mechanisms for enhancing network security and improving energy efficiency within the project framework, thereby serving as a referential tool throughout the project's lifespan.

1.3 Interrelations

The NATWORK consortium integrates a multidisciplinary spectrum of competencies and resources from academia, industry, and research sectors, focusing on user-centric service development, robust economic and business models, cutting-edge cybersecurity, seamless interoperability, and comprehensive on-demand services. The project integrates a collaboration of fifteen partners from ten EU member states and associated countries (UK and CH), ensuring a broad representation for addressing security requirements of emerging 6G Smart Networks and Services in Europe and beyond.

NATWORK is categorized as a "Research Innovation Action - RIA" project and is methodically segmented into 7 WPs, further subdivided into tasks. With partners contributing to multiple activities across various WPs, the structure ensures clarity in responsibilities and optimizes communication amongst the consortium's partners, boards, and committees. The interrelation framework within NATWORK offers smooth operation and collaborative innovation across the consortium, ensuring the interconnection of the diverse expertise from the various entities (i.e., Research Institutes, Universities, SMEs, and Large industries), enabling scientific, technological, and security advancements in the realm of 6G. The D5.2 “Energy Efficient AI-based Security

Processes" deliverable addresses activities of the NATWORK project related to energy-efficient security, with concrete integration points across multiple work packages and deliverables. It leverages architectural definitions and requirements from WP2—particularly D2.2: "6G Use Case Scenarios and Requirements" and D2.3: "NATWORK Architecture, Interfaces & Specifications"—to ensure that the proposed AI-based security processes comply with the NATWORK architecture, interface definitions, and system constraints established early in the project.

D5.2 also builds directly on the security, orchestration, and data-plane innovations produced in WP3. For example, the mechanisms introduced in D3.1: "Secure-by-design orchestration & data-plane offloading.r1" and D3.3: "Intent-based service security and adaptive security techniques.r1" provide the foundational context in which energy-aware AI-driven security can operate. D5.2 incorporates these capabilities to ensure that security functions are not only adaptive but also optimally placed across the network with energy consumption considerations in mind.

Finally, the outcomes of D5.2 feed directly into WP6 integration and validation activities. The energy-efficient processes proposed in D5.2 are incorporated into the testbeds as outlined in D6.2: "System Integration on the testbeds, Pilot installations and implementations" and evaluated within the NATWORK pilot scenarios defined in D6.1: "Definition of the evaluation framework & Pilot specifications". Their performance will be assessed and reported in D6.3: "System validation, End-user evaluation & Lessons Learnt", ensuring that the developed mechanisms demonstrate measurable improvements in both security and energy consumption under realistic 6G-aligned conditions.

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

2. State of the art

2.1 Disaggregated Net-Zero machine learning

Disaggregation of data plane programs has been widely studied [1],[2]. Notably, a framework named DINC [3] has also been created, which is capable of automatically disaggregating in-network, random forest-based traffic classifiers, among other programs. The authors emphasize the framework's ability to overcome resource constraints through program distribution, which is crucial in our plan to achieve net-zero classification.

In-network traffic classification using random forests (RFs) is a thoroughly researched topic. Various methods have been presented showing how random forests can be efficiently encoded into the programmable pipeline of switches. For example, the authors of pForest [4] embed each depth of each decision tree (that makes up the RFs) into a distinct match-action table, and each tree node has a respective entry in a match-action table. This approach can be implemented without relying on sparse TCAM memory, but it creates long chains of computation that may not be supported by all hardware. It's also important to note that pForest utilizes as-soon-as-possible inference, which means that flows are classified as soon as a high confidence classification verdict can be made, as opposed to only being classified after they have ended or after they have reached a predetermined packet count. The authors of [5] rely on a different method to embed RFs into the data plane. They encode each decision tree into a respective match-action table by using the features directly as table keys and mapping feature value ranges to the predicted label. This approach doesn't create long chains of computation, but uses ranges of values as table keys, which may not be supported by all switches.

The utilization of online learning techniques to aid in-network inference has also been studied. For example, the authors of [6] execute inference within the data plane but forward all the extracted features to the control plane for online learning. The controller is responsible for training and deploying improved classifiers into the data plane. Two sets of match-action tables are utilized to make sure the updating process does not interfere with ongoing classification: the secondary tables can be updated while the primary tables are being used. HALIDS [7] resorts to a different approach: it only forwards feature to the control plane when in-network classification is not possible with high confidence. This reduces the amount of data that must be transferred through the slow path, while still enabling online learning. The features are sent to an intrusion detection system (IDS), which can classify flows better than in-network classifiers (because it doesn't suffer from data plane limitations), but also introduces considerable latency into the classification process. The control plane uses the predictions from the IDS to improve the in-network classifier, reducing the number of flows it needs to forward to the IDS in the future.

Taking into account the research bibliography, there are several works regarding disaggregated inference and in-network traffic classification, respectively. The main building blocks have exhaustively been discussed, but, to the best of our knowledge, their integration has not been thoroughly attempted before.

2.2 Jamming detection algorithms on MPSoC platforms

Deep Neural Networks (DNNs) have gained prominence due to their remarkable accomplishments across various domains, including telecommunications and security. DNN-based algorithms, such as Convolutional Neural Networks (CNNs), have proven their great performance in tasks such as the detection and classification of radio frequency signals. These algorithms usually have a high computing cost, since they require vast amounts of data for training the deep layers that comprise their architecture.

Jamming detection models are usually integrated into decision-making systems which are often deployed in resource-constrained environments that typically have at least one of the following requirements:

- **Real-time operation**. This is the case of critical use cases, like critical infrastructure protection, security-related applications, or applications that require a quick response, which may be the case in defense use cases.
- **Low energy consumption**. This may be an important issue for applications that are deployed in remote locations, which need to rely on batteries; or in environments where there are no affordable power sources available.
- **Small form factor and restricted weight requirements**. This is the case of embedded devices, which may require to be deployed on portable devices; or in unmanned aerial vehicles which cannot carry heavy payloads.

A Multiprocessor System-on-Chip (MPSoC) is a single integrated circuit that contains multiple processors (CPUs) and programmable logic (FPGAs) to meet the demands of real-time performance and low-power consumption. MPSoCs are predominantly used in embedded applications that require significant processing capabilities, such as military and defense systems, autonomous robots, and advanced multimedia systems.

In NATWORK project, the ZCU102 Evaluation Board with MPSoC AMD Zynq UltraScale+ is used to test the acceleration of DNN algorithms in MPSoCs, as it is compatible with state-of-the-art acceleration frameworks.

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

2.2.1 Acceleration frameworks

The process to accelerate DNN Networks on FPGAs requires a series of steps that range from network optimization, weights and bias quantization and compilation to translate to a hardware language. AMD is one of the leading manufacturers of FPGAs, and it provides Vitis AI [8], which is a specialized framework to deploy models in its own hardware.

HLS4ML is another competitive framework, particularly considering its accessibility, open-source licensing, support for a variety of neural network architectures and compatibility with a range of tested hardware platforms.

2.3 Energy efficient and latency-aware workload scheduling

The challenge of optimizing both energy consumption and latency in cloud-native and edge-distributed workloads continues to evolve rapidly. Early works focused on minimizing energy use through heuristic or Integer Linear Programming (ILP) based scheduling strategies within datacenters, while later research expanded these methods to the fog and edge layers, introducing resource heterogeneity, mobility, and stricter QoS constraints. Several trends have emerged recently that push this research frontier forward and are presented below.

A major shift has been the adoption of deep reinforcement learning (DRL) and attention-based DQN methods to dynamically balance latency and energy across edge and fog resources. In [9], the authors propose an energy-efficient edge scheduler using an attention-enhanced DQN, which autonomously learns offloading policies that minimize both energy consumption and response time compared to heuristic baselines. The authors in [10] apply a DQN framework to IoT–fog–cloud environments, jointly optimizing latency and energy efficiency in large-scale deployments with realistic network conditions. These approaches represent a movement from static or rule-based heuristics toward adaptive, data-driven control for heterogeneous, dynamic edge infrastructures.

Beyond RL, hybrid optimization techniques are also gaining traction. In [11], the authors integrate the Boltzmann probabilistic modeling with Bayesian optimization to continuously refine scheduling decisions under varying load conditions. Their method dynamically adjusts resource allocation to reduce both power draw and end-to-end latency in edge environments. Such approaches complement learning-based schedulers by providing lightweight, explainable optimization mechanisms suitable for constrained devices.

Recently, Kubernetes has become the standard orchestration platform for distributed workloads, including ML inference. Recent studies focus on network-topology-aware and latency-driven scheduling plugins for multi-cluster and multi-zone K8s environments. The works [12] and [13] demonstrate that incorporating real-time inter-node latency metrics into Kubernetes' scheduling decisions significantly improves service response times. Their implementations show that topology-aware scheduling reduces latency bottlenecks without increasing system overhead, validating the feasibility of latency-driven orchestration at cluster scale.

A key enabler for energy-aware scheduling is the availability of accurate energy telemetry at the container level. The authors in [14] have introduced the Kepler framework that calculates the energy consumption of applications running on containers. The work in [15] validates Cloud Native Computing Foundation Kepler's (Kubernetes-based Efficient Power Level Exporter) capability to estimate container-level power consumption with sufficient accuracy for real-time scheduling feedback. Such observability allows schedulers to operate on measured rather than estimated energy data, supporting fine-grained power-aware orchestration across workloads.

The research trajectory indicates a steady transition from heuristic scheduling to intelligent, telemetry-driven, and learning-based orchestration. Energy metrics are now being exposed natively within Kubernetes via tools such as Kepler, while latency awareness has evolved from static modeling to real-time measurement and topology sensitivity.

2.4 Energy-efficient RDMA transfer

Remote Direct Memory Access (RDMA) technologies (e.g., InfiniBand or RoCE) allow data to move directly between the memory of two systems with minimal CPU intervention, in contrast to conventional TCP/IP, which incurs kernel and copy overheads. This kernel-bypass approach yields significantly lower latency and higher throughput per transfer, while also reducing CPU load on hosts. For example, one-sided RDMA operations (Reads/Writes) avoid CPU involvement entirely and show very low end-to-end latency, whereas TCP and even two-sided RDMA Send/Receive involve more CPU processing and higher latency. These efficiencies can translate into lower energy per byte transferred for RDMA, since less CPU work (and associated power) is needed to achieve the same data movement. However, RDMA's advantages come with caveats. It requires specialized NIC hardware and a lossless network environment. If either endpoint lacks RDMA capability or if network conditions are unfavorable, a system must fall back on standard TCP transfers. Moreover, RDMA networks demand careful congestion management (e.g., priority flow control in RoCE) to avoid performance issues, whereas TCP's mature congestion control and ubiquitous support offer greater generality at the cost of higher CPU utilization and overhead.

Recent research has focused on making both RDMA- and TCP-based transfers more adaptive and energy-efficient. One direction leverages fine-grained telemetry and intelligent control to tune data flows in real time. For instance, Pan and You introduce a multi-path RDMA framework that embeds in-band network telemetry into RDMA's flow control signals, enabling real-time link utilization feedback and dynamic load balancing across paths [16]. Such telemetry-driven optimization aligns network usage with current conditions, improving throughput while preventing wasted energy on under-utilized links.

Other approaches emphasize software-defined networking (SDN) control to manage energy at the network level. Brito et al. present an SDN-based scheme where a central controller monitors traffic and selectively powers down idle switches, achieving near energy-proportional network operation with minimal performance impact. This demonstrates that programmability in the data plane (e.g., using P4-programmable switches or SmartNICs) can autonomously adjust resource usage – shutting off, downclocking, or rerouting as needed – to save power while maintaining service quality [17]. Similarly, machine-learning-driven agents have been proposed to sit between the application and transport layers, dynamically choosing transfer modes and parameters based on telemetry (e.g., congestion signals, CPU load, temperature) to balance speed and energy consumption.

RDMA significantly reduces CPU load and latency in network communications, for example, boosting the connectivity between the cloud and the edge while guaranteeing a reduced energy consumption footprint. There are, in fact, multiple RDMA operations with distinct energy footprint. For example, Guaitolini et al. provided a measurement of the different consumptions done for RDMA Send, RDMA Receive, RDMA Write, and RDMA Read with files of different sizes [18].

These innovations highlight key trade-offs in performance vs. energy. RDMA offloads work from CPUs, yielding high throughput and low latency, but its efficacy depends on hardware support and network conditions. TCP is universally compatible and adaptive, though less efficient in CPU and energy usage per transfer. Telemetry- and SDN-enhanced solutions can bridge the gap by adjusting behavior on the fly: they improve adaptability and can reduce energy draw (for example, by idling hardware during low load), at the cost of added system complexity and slight control overhead. Overall, the state-of-the-art envisions a smart orchestration of data transfers where protocols and network devices collaboratively minimize energy consumption while still meeting performance demands.

2.5 Net-Zero machine learning for anomaly detection

Several studies have concentrated on machine learning-driven anomaly detection in 5G networks and on sustainable (or Net-Zero) machine learning. These domains establish the basis for attaining environmentally aware anomaly detection in 5G systems. Despite considerable advancements in the development of energy-efficient IDS, the establishment of universal sustainability Key Performance Indicators (KPIs), and the benchmarking of carbon accounting for machine learning independently, the direct amalgamation of these methodologies, particularly via decentralized or federated techniques, is still a largely unexamined domain [19],[20]. The following paragraphs will describe the most prominent current works and concepts within these distinct yet converging domains.

Anomaly detection in 5G networks using machine learning is a vital and rapidly evolving domain, addressing the security and operational challenges arising from the inherent complexity of these networks. This complexity encompasses elements such as NFV, SDN, network slicing for diverse service requirements, edge computing implementations, and significant IoT integration. The present state of the art employs a diverse array of machine learning approaches. Commonly utilized supervised learning techniques include Support Vector Machines (SVM), Decision Trees, K-Nearest Neighbors (KNN), Logistic Regression, Naive Bayes, and robust ensemble approaches such as Random Forests and XGBoost [21],[22]. These approaches frequently attain great precision in categorizing recognized assault patterns or anomalies. Conversely, unsupervised methods, including clustering algorithms and autoencoders, are crucial for identifying novel or zero-day threats that lack preset fingerprints. They detect anomalies from established normative behavior, aiding in the identification of potential dangers. Deep learning models hold considerable importance in this domain. CNNs excel in detecting spatial patterns and can be utilized for analyzing network traffic represented as images. Recurrent networks, such as Long Short-Term Memory (LSTM) networks, are crucial in the analysis of time-series data. Both categories of deep learning architectures exhibit remarkable proficiency in capturing the complex, high-dimensional, and temporal dependencies inherent in 5G data.

The concurrent advancement of sustainable or Net-Zero machine learning emphasizes the need to comprehend and proactively mitigate the environmental impact of AI operations, including energy usage and related carbon emissions. Achieving Net-Zero entails equilibrating the emitted greenhouse gases with those extracted; a principle that is gaining prominence in Information and Communication Technology (ICT) and AI. Fundamental efforts in this domain entail the development of comprehensive frameworks and standardized key performance indicators for assessing and reporting sustainability, rendering it a pragmatic operational issue [23]. Frameworks like GreenDFL provide approaches to measure energy consumption (in kilowatt-hours) and corresponding CO_2 emissions across various stages of the machine learning lifecycle, specifically training, inference, and communication in distributed environments [20]. These calculations require comprehensive telemetry data, encompassing hardware specifications such as CPU Thermal Design Power (TDP) and real-time GPU power consumption, hardware utilization metrics, execution durations, data transfer volumes, energy costs of communication channels, and, crucially, the carbon intensity of the electricity used. The intensity may fluctuate by location and can be mitigated by local renewable energy sources. Instruments such as CodeCarbon and CarbonTracker facilitate the automation of these measurements [24],[25].

Strategies for mitigating environmental effects while preserving performance depend on precise measurement and sophisticated optimization methods. Initially, meticulous hardware selection prioritizes energy-efficient accelerators like GPUs or specialized chips over less efficient CPUs for demanding tasks. Secondly, algorithmic efficiency is enhanced through the development of simpler models and the implementation of techniques such as model pruning, quantization, knowledge distillation, and dynamic early stopping predicated on convergence criteria to eliminate unnecessary computations [26]. Third, carbon-conscious deployment and scheduling allocate computing jobs to data centers or edge nodes situated in areas with low-carbon power grids, or time-intensive functions for periods of high renewable energy availability. Ultimately, efficient distributed learning minimizes overhead by optimizing communication patterns, utilizing effective data-transfer protocols, and investigating federated or decentralized learning (DFL) on platforms like Nebula [27]; nevertheless, the energy trade-off between communication and computation requires meticulous assessment.

In addition, sustainability-conscious orchestration in DFL involves the dynamic selection of participating clients and adjustment of aggregation strategies based on reported energy and carbon metrics. Benchmarking studies highlight the importance of evaluating accuracy alongside emissions, showing that substantial carbon reductions of approximately 40-45% can often be attained through careful model selection, such as XGBoost, potentially providing greater efficiency than Random Forest for comparable accuracy and hyperparameter optimization. This holds true even for security analytics tasks such as DDoS detection. The integration of these varied measurement and optimization techniques paves the way for achieving Net-Zero Machine Learning in demanding applications, such as 5G anomaly detection.

Co-funded by the European Union
6GSNS
Project funded by
UK Research and Innovation

3 Disaggregated Net-Zero machine learning

The objective of the disaggregated net-zero machine learning is to provide a green solution for in-network attack detection and mitigation. Our approach is based on the disaggregation of tree-based ensemble methods (i.e., random forests), the low-latency in-network inference of these disaggregated segments, and their continuous improvement in a zero-touch manner through online learning, allowing it to react to emerging network threats dynamically and in a timely fashion. We have described our main achievements in [28].

3.1 Proposed model and system architecture

Our proposed model comprises multiple components, which are detailed in the subsequent paragraphs. An overview of the proposed system architecture, showing the distinct components and their interconnections, is shown in Figure 1.

The key components of our proposed model are the programmable switches responsible for identifying network flows, extracting flow features from individual packets (such as TCP/UDP ports, and maximum packet length), and executing as-soon-as-possible in-network inference of random forests using the extracted features. To enable monitoring and online learning, the switches select a small portion of the network flows and send the network traffic along with the predicted labels associated with these flows to the control plane. Application-specific hardware, such as programmable hardware switches, can execute these steps with low latency, high throughput, and increased energy-efficiency compared to general-purpose processors.

The controller, situated within the control plane, is responsible for supervising the in-network inference process and for continuously improving it through online learning. The controller receives the cloned traffic from the switches and subsequently classifies it within the control plane, assisted by an off-the-shelfIDS. By comparing the classification results received from the switches and the IDS, the controller can estimate the accuracy of the in-network classification and take appropriate steps to improve it in a zero-touch manner: by saving the features of the most recent flows and the labels predicted by the IDS, the controller is able to automatically train improved classification models and deploy them onto the programmable switches.

The previously mentioned IDS is a simple off-the-shelf IDS that runs within the control plane. Because it does not suffer from the limitations of in-network solutions, this IDS can classify flows with greater accuracy compared to programmable switches. It is essential to note that this IDS is only required to process a small fraction of the network traffic used for monitoring and online learning: the majority of the traffic is only classified by the programmable switches. The high accuracy of the control plane IDS enables us to treat it as a source of truth during the training of improved classification models, as its predicted labels can be considered true labels, which are required for the training of improved classification models.

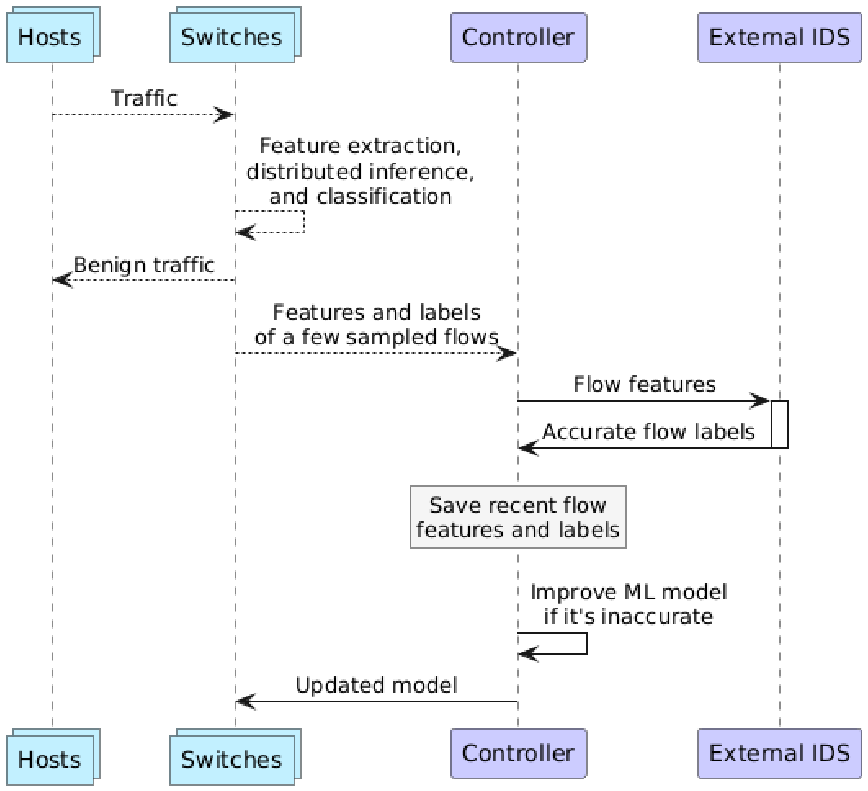

Hosts
Switches
Controller
External IDS
Traffic
Feature extraction, distributed inference, and classification
Benign traffic
Features and labels of a few sampled flows
Flow features
Accurate flow labels
Save recent flow features and labels
Improve ML model if it's inaccurate
Updated model
Hosts
Switches
Controller
External IDS

Figure 1: An overview of the proposed model

3.1.1 Model disaggregation and utilization of previously unused computation capacities

Network traffic is forwarded through multiple switches as it is traverses the internal network. If the switches possess available, unused resources (unused memory and computational capacity), then these free resources can be utilized to execute parts of the random forest inference process. Disaggregated inference is achieved through the coordination of the controller component, which segments the traffic classification model into distinct parts and deploys each part on a separate switch. The segmentation of random forests can be implemented in a straightforward manner because the decision trees forming the RF can be independently inferred; therefore, they can easily be placed in separate segments. Once each decision tree has been inferred, then the verdict of the entire random forest can be computed. If the segments formed during the disaggregation process fit inside the unused resources of the switches already deployed within the network, net-zero in-network inference is achieved.

This is because the power consumption of hardware switches is largely independent of the amount of per-packet processing they execute, provided that the processing fits within the constraints of the hardware. Figure 2 provides an example of the model disaggregation.

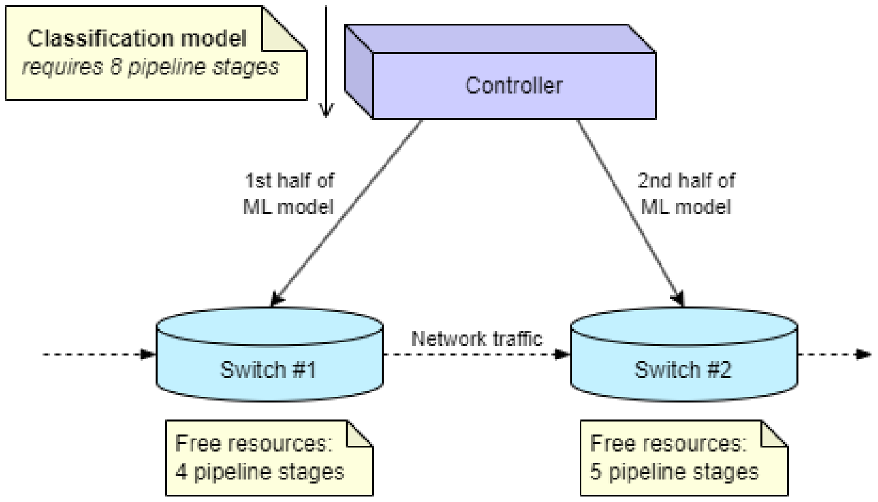


Figure 2: An example of disaggregated model deployment. The goal of the disaggregation is to fit the classification model into the unused segments of existing networking infrastructure.

3.1.2 Federated learning and support for network slices

The in-network attack mitigation system was designed to be compatible with network slices being used within the network. In this deployment scenario, the different network slices might belong to different network operators; therefore, it is imperative not to share sensitive data such as raw flow features between them, or even to export such data to a third party. To satisfy these constraints, while still maximizing the attack detection capabilities of the system, we envision a setup utilizing secure data aggregation (federated learning) and a centralized coordinator component, as depicted in Figure 3. Each network slice has its own IDS and controller, but they share their trained models with the coordinator, which merges them and provides an improved model to all controllers to be deployed on the switches. It's important to note that the implementation of the secure data aggregation functionality required to realize this feature has not been completed.

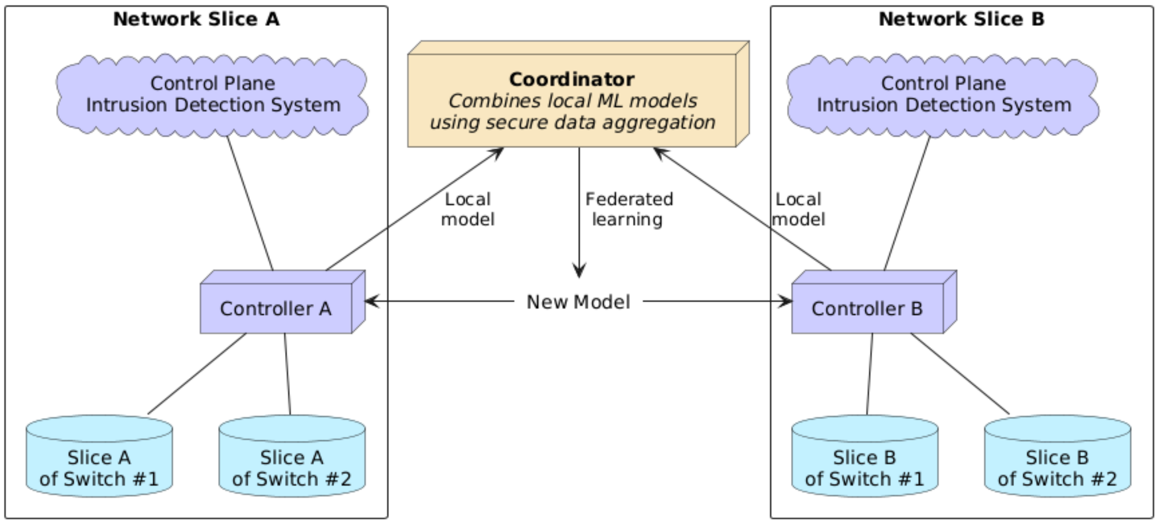


Figure 3: Our proposed setup supporting multiple network slices using federated learning. The confidential raw flow data does not cross network slice boundaries.

3.2 Implementation

The in-network attack classification system has been implemented to support multiple types of programmable switches, a wide array of possible features to be used for inference, and capabilities such as as-soon-as-possible inference (introduced by pForest [4]) or seamless model switchovers. The subsequent paragraphs provide a detailed description of these aspects.

3.2.1 Extraction of flow features

Random forests classify network flows as either benign or attack (malicious) by comparing the various features of the flows to thresholds computed during the training of the classification model. Some of these features can be directly extracted from individual network packets (such as the length of the packet), but most features (e.g., maximum packet length or TCP flag counts) require flow-specific data to be persisted. We implemented flow-based persistence in P4 programmable switches by indexing registers with the hash of the flow's identifier (5-tuple). This is a simple and efficient method for storing flow-specific data, but it is susceptible to hash collisions, which would cause distinct flows to attempt to use the same registers, corrupting the stored features of both flows. The risk of hash collisions is minimized by not persisting the data of flows that have been inactive for at least 30 seconds: they are considered finished, and their register slots are reset when a new flow is assigned to the same slot.

3.2.2 Model training and as-soon-as-possible inference

The programmable switches select a small fraction of the network flows and forward the extracted features and the assigned classification labels belonging to these flows to the controller. The flows are randomly selected, based on the hash of their respective flow identifiers. To support as-soon-as-possible inference, the extracted features are sent after each processed packet, rather than only sending them once the flow has ended. The controller saves the last few minutes of flow features and flow labels received from the switches. The controller also forwards the features to the external IDS and saves the highly accurate flow labels found in the response. The in-network classifier's performance can be assessed by treating the labels received from the switches as the predicted labels, the labels received from the IDS as the true labels and then computing the accuracy or the F1 score metric.

The controller periodically trains new classification models using the received features and the highly accurate flow labels. To classify flows as soon as a high confidence classification can be made (i.e., to achieve as-soon-as-possible inference), a model training process is used that is like the one described in pForest [4]. A separate random forest is attempted to be trained on the first N packets of each flow, where N is a number that starts at one and increases by one at every iteration, until no unclassified flows remain. The verdict of a random forest is used only if the confidence of its verdict surpasses a predetermined threshold; otherwise, the classifier will wait for the next packet(s) to be received. Furthermore, a random forest is only retained if its accuracy meets the acceptance criteria when evaluated using the data cached by the controller: the random forest is tested against the stored flow features, and the accuracy of its high-confidence classifications is measured. As a result, the final classification model will attempt to classify the network flows at various flow lengths (e.g., after the 3rd, 4th, and 10th packet) but will only accept the classification verdict if the confidence is high enough. This allows easily classifiable flows to be labeled after just a few packets, while more complex flows will stay unlabeled until a high-confidence verdict can be made.

Finally, the performance of the currently used and the new classification model is compared. If the new model outperforms the old one (e.g., its accuracy is at least 1% greater), then the new model is deployed into the programmable switches and will be used for subsequent in-network classification.

3.2.3 Achieving seamless model switchovers

As described in the previous subsections, the classification models deployed into the network are frequently replaced by newer, improved models. For this reason, it is imperative for the deployment process not to interfere with the ongoing classification of the active flows. Without proper precautions, the deployment process may update a random forest while it is being traversed, potentially leading to incorrect classifications and reduced accuracy. These issues can be avoided by making the switchover process (switching from the old to the new model) atomic.

The deployment of a classification model involves updating several match-action tables across multiple switches, which cannot be executed as a single atomic operation for several reasons:

each switch can only be updated separately, and most switches don't support the atomic updating of multiple tables, either. The following steps are taken to overcome this challenge and achieve atomic switchovers within a single switch:

- Each random forest is assigned a numeric identifier. Before an RF is embedded into a switch, it is given an identifier that currently identifies no other RF within the switch.
- Within each switch, a match-action table mapping flow lengths (packet counts) to RF identifiers is used to select which RF should be used for inference.
- The first step of the deployment process is the embedding of each decision tree of each random forest into the data plane, without overwriting the RFs that are currently in use. Previously deployed but no longer used RFs may be overwritten. This also means that the switches must reserve enough memory for at least two sets of RFs (two classification models) to be simultaneously deployed (one active model and one inactive model).
- After all decision trees have been embedded, the match-action table mapping flow lengths to RF identifiers is updated to map packet counts to the newly deployed RFs.

Prior to the last step being executed, the switches only use the old RFs for inference, which are kept unchanged during the deployment process. This guarantees atomic switchovers within individual switches but does not solve the issue across the entire network due to the disaggregation of classification models: the last step cannot be executed atomically across all switches; therefore, different switches may try to use different RFs to execute their part of the disaggregated inference. To overcome this issue, only the first switch to process the packet reads which RF should be used for inference and saves this RF's identifier into the packet header supporting the disaggregated inference, and subsequent switches obtain the RF identifier from this header field. The header contains the intermediate results of the disaggregated inference (the verdicts of the previously executed decision trees) and is only kept while the packet is within the internal network: it is transparent to end-hosts and the outside network.

3.3 Preliminary results

The proposed in-network attack detection system has been evaluated using the improved version of the CIC-IDS-2017 [29],[30] dataset, a widely used data source for the assessment of attack detection systems. The evaluation was performed using two different kinds of programmable switches: the Intel Tofino hardware switch and a P4 programmable, eBPF-based software switch called NIKSS [31]. Three eBPF switches were used, arranged in a chain topology, each supporting the deployment of 6 decision trees with a maximum depth of 7. While larger models could have also been used, these values were found to be sufficient. The Tofino switch supported the deployment of 3 decision trees, each with a maximum depth of 5. The control plane (controller and IDS) was identical for all switch types. During the evaluation, the IDS was replaced by a mock implementation returning the known true classification of each flow (which were published as part of the CIC-IDS-2017 dataset) to avoid a third-party IDS affecting the evaluation results.

As part of the evaluation, we measured the in-network attack detection system's ability to identify flows that belong to a DDoS attack, a port scan, or either: the system was required to classify each flow as either an attack (malicious) flow or as a benign flow. We extracted packet capture segments from the CIC-IDS-2017 dataset and replayed the relevant parts (containing benign traffic as well as the DDoS attack, the port scan, or both) on a host machine connected to the switch(es). The system's ability to identify DDoS attacks was measured using eBPF switches; the final F1 score that was achieved is 0.96. The accuracy over time can be seen in Figure 4. The figure primarily uses the accuracy metric because there are some periods within the evaluation run where there are no attack flows, in which cases the F1 score would be undefined.

Figure 4 also shows the number of active benign and attack flows, as well as the timestamps when the controller deployed a new, improved traffic classification model into the data plane. Our in-network classifiers rely solely on online learning (rather than pre-training on a dataset); therefore, an empty classification model is embedded into the switches at the start of each evaluation run, which classifies all flows as benign. This resulted in an initial accuracy score of 1.0 because the start of the packet capture segment that was being replayed only contained benign flows. The first attack flows appeared after around 150 seconds, at which point the classifier's accuracy dropped significantly. This happened because the initially used (empty) classification model incorrectly classified the attack flows as benign.

The controller swiftly detected the drop in accuracy and started training improved models using the data (flow features and labels) received from the switches. The figure shows that the controller executed multiple model deployments in quick succession. This is because the first few improved models are trained on a very small sample of attack flows (because the attack flows have just appeared in the network). As the features of more attack flows become available to the controller, it can train better and more generalized models. After some time has passed, the training of new models will become unnecessary, as the new models won't be able to perform better than the already deployed models. It's also important to note that the deployment of classification models does not interfere with ongoing classification processes; therefore, there is no need to circumvent the frequent replacement of classification models. Doing so would only delay the deployment of improved classification models, hindering the system's attack detection ability.

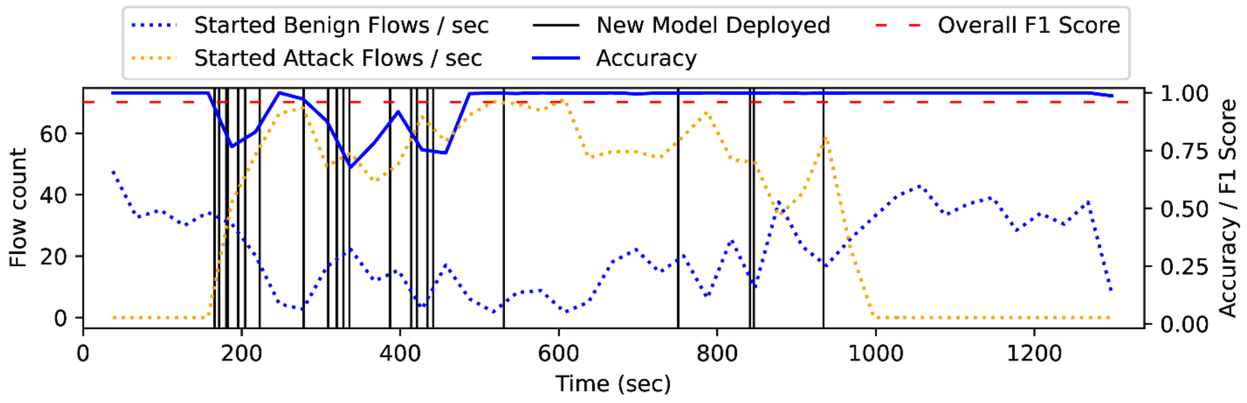


Figure 4: Accuracy of in-network DDoS attack detection over time

The system's flow classification capabilities were also measured against port scan attacks, found in a different segment of the CIC-IDS-2017 packet capture. In this scenario, the Intel Tofino hardware switch was used, and an F1 score of 0.98 was achieved. The accuracy of the system over time is shown in Figure 5. Finally, we also executed an evaluation run when the same traffic classification model had to detect the two distinct attack types concurrently, because both were included in the replayed packet capture segment. This evaluation was conducted on eBPF switches, and the results are presented in Figure 6. An overall F1 score of 0.95 was achieved, which is only slightly lower compared to the scenarios where only a single attack type was present.

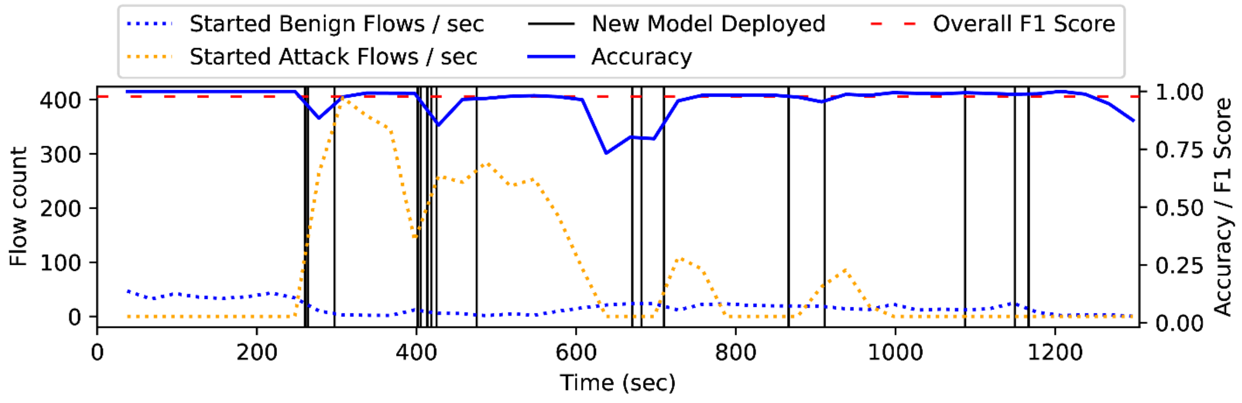


Figure 5: Accuracy of in-network portscan detection over time

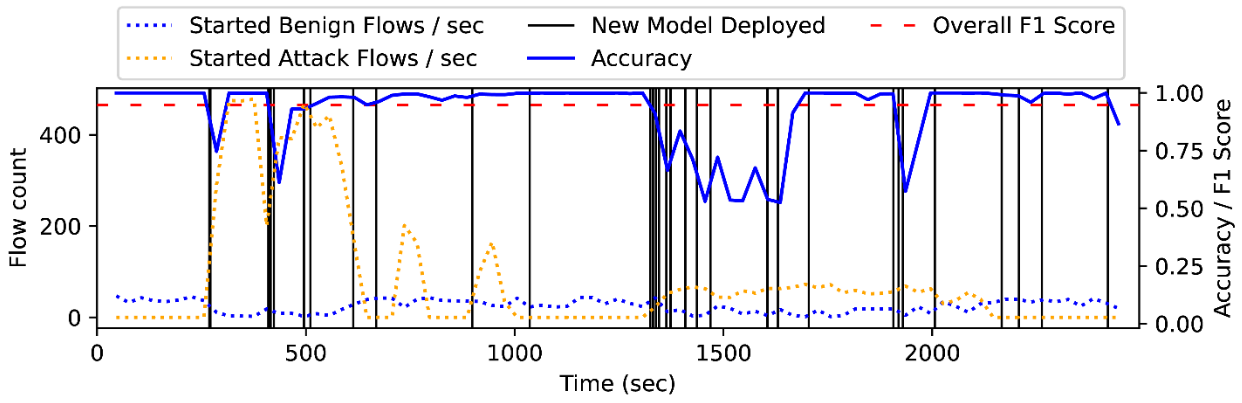


Figure 6: Accuracy of in-network detection of concurrent DDoS and portscan attacks

3.3.1 Optimal ratio of monitored flows

Measurements have been conducted to determine the optimal ratio of flows whose features (and predicted labels) should be forwarded to the control plane. The challenge lies in maximizing learning effectiveness while minimizing controller workload. The more data the controller receives, the more data it has available to execute effective online learning. On the other hand, forwarding too much data to the controller could put excessive strain on the CPU, or even overload the controller or the network link, introducing additional latency and packet loss into the data stream. While some related works, such as [6], forward 100% of the network flows to the control plane, we have found that even a small portion (e.g., 5%) of the flows is adequate for effective online learning. The results of our measurements are presented in Table 1. These results were achieved using eBPF-based switches used for DDoS detection.

Table 1: Fraction of monitored flows versus the achieved F1 score

Sampling ratio	1%	3%	5%	10%	25%	50%	100%
F1 score	0.94	0.96	0.96	0.95	0.94	0.91	0.95

3.3.2 Time required for model deployment

When the controller trains a new model as part of its online learning process, and it determines that the new model outperforms the previous (currently used) one, it must embed the new model into the programmable switches. The deployment process is composed of various table entries being updated, added, or removed, with the exact number of operations depending on the complexity (size) of the model being deployed. The less time it takes to complete the deployment process, the sooner the new classification model will be used, resulting in more accurate in-network classifications.

Our measurements show that the type of switch being updated greatly influences the time required for deployment: models can be embedded over 20 times quicker into Tofino switches compared to eBPF-based switches. Tofino deployments take a few hundred milliseconds, while eBPF deployments take multiple seconds. We believe this difference stems from the fact that the different switches have different control plane APIs, which are not optimized to the same extent and support different features. For example, while the Tofino control plane API supports batch actions (executing multiple table entry updates as part of a single operation), the nikss-ctl [31] used for updating eBPF switches currently requires a separate shell command to be executed for each table entry update.

4 Jamming detection algorithms on MPSoC platforms

The objective of accelerating the jamming detection neural network on an MPSoC is to enable an energy-efficient hardware implementation that facilitates the evaluation of key performance metrics, including inference speed, power consumption, and classification accuracy.

4.1 System architecture

The Vitis AI tool from AMD has been employed as the framework for accelerating neural networks on FPGAs. This tool provides support for all the operators involved in the jamming detection network, which primarily consists of CNNs. The selected edge device is the ZCU102 Evaluation Board based on the Zynq UltraScale+™ XCZU9EG-2FFVB1156E MPSoC, which executes the computational graph of the model. Figure 7 illustrates the main components of the proposed system, which are described in detail in the following sections.

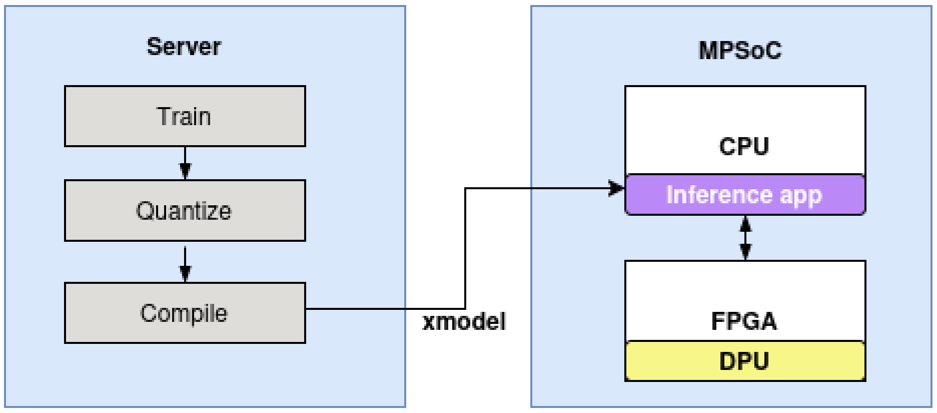


Figure 7: System architecture

The process starts on a high-performance server, where the model is first trained and then converted into a hardware-friendly format. After training, the network undergoes quantization to reduce numerical precision and improve computational efficiency. The quantized model is then compiled into a format known as an XMODEL, which is specifically designed for execution on FPGA-based devices. Once the XMODEL is generated, it is transferred to the target hardware. On this platform, the CPU runs an inference application that coordinates data handling and control. At the same time, the FPGA hosts a Deep Learning Processing Unit (DPU) responsible for executing the neural network operations. The inference application on the CPU communicates with the DPU to perform real-time predictions efficiently.

Co-funded by the European Union

4.2 Proposed model and framework

As the primary objective of this task is to study the hardware acceleration of neural networks in MPSoCs, rather than to develop a model and carry out an extensive training and validation process, a pre-trained custom model from Gradiant was reused. The model is defined using PyTorch and follows a CNN architecture that processes the Fast Fourier Transform (FFT) of I/Q signal samples to perform jamming classification as shown in Figure 8. As the DPU does not support the FFT operation, this computation was offloaded to the preprocessing stage, allowing the forward of the model class to consist exclusively of supported layers.

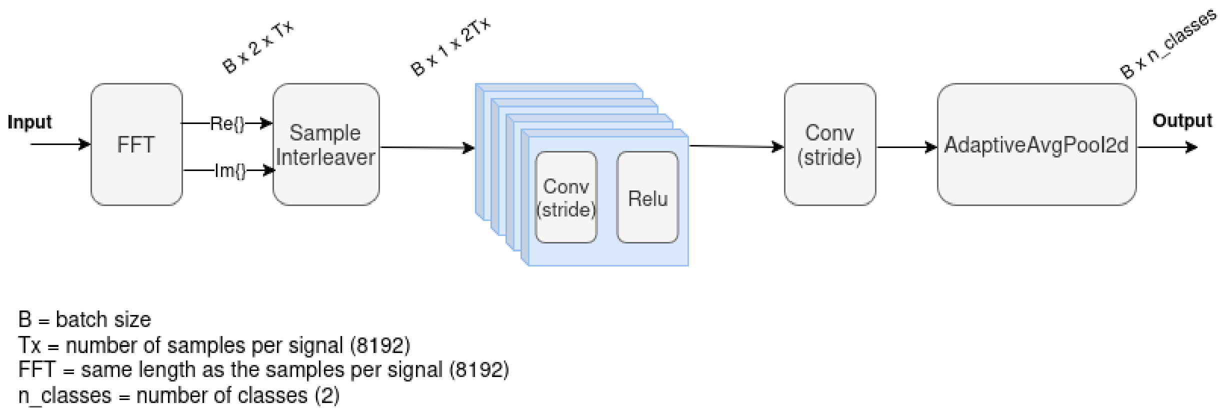


Figure 8: Model architecture

4.3 Implementation

4.3.1 Firmware

To support model inference within the MPSoC, the DPU must first be instantiated in the FPGA design and configured according to the project's requirements (such as expected performance) and the specific needs of the model to be run. Furthermore, this DPU must be interfaced with the processor unit (ARM), where the software application runs, to enable bidirectional data flow. This configuration is considered the firmware of the design, which is loaded into the FPGA every time the platform boots. Once this is complete, the platform is ready, waits for a model to be loaded and then proceeds with the actual inference.

Figure 9 shows the firmware block design, created using the Xilinx Vivado tool, featuring two main entities along with interconnection and control blocks.

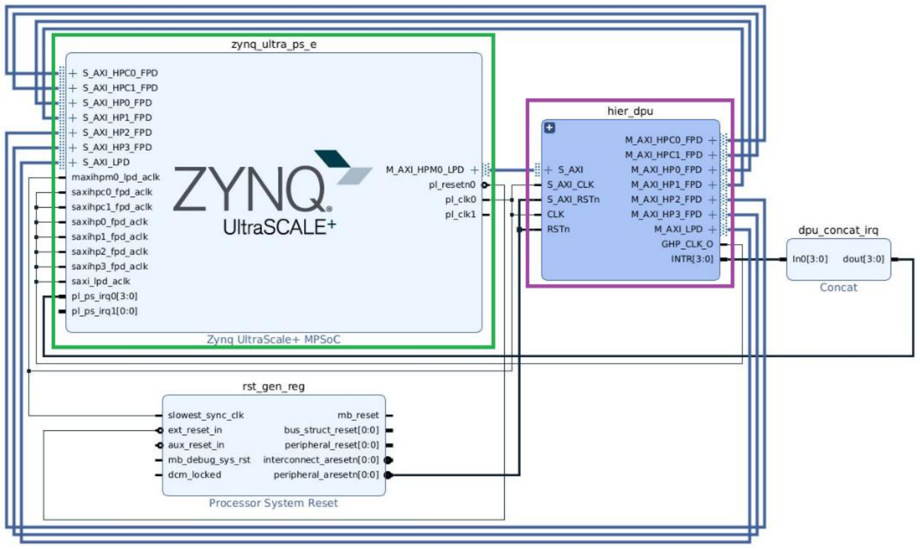


Figure 9: Overview of FPGA design

The two main entities of firmware design are:

- **MPSoC IP Core** (highlighted in green) serves as the logical bridge connecting the Processing System (ARM) and Programmable Logic. It manages I/O peripherals, AXI interfaces, multiplexed inputs/outputs, clocks, interrupts, and resets. Additionally, it handles System Level Configuration Registers (SLCRs), configures high-speed interfaces, DDR memory, and isolation settings.
- **DPU** (highlighted in purple) is a programmable engine and parameterizable IP core optimized for deep neural networks. Pre-implemented on the hardware, it requires no place-and-route and accelerates inference workloads for computer vision applications, such as image classification, segmentation, and object detection. It utilizes the Vitis AI instruction set and an efficient tensor-level instruction set to support various popular CNNs, including VGG, ResNet, YOLO, and MobileNet.

4.3.1.1 DPU configuration and resulting design

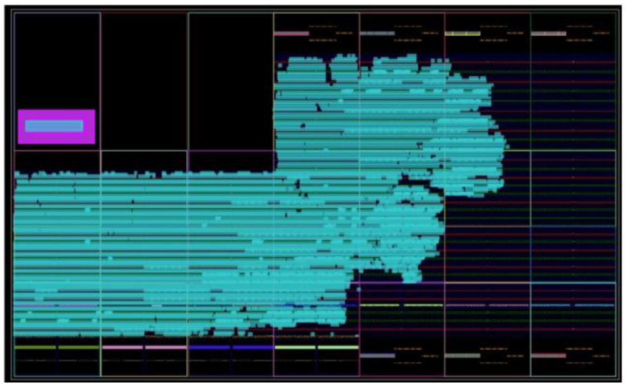
The instantiated DPU offers several key parameters to optimize its performance and resource utilization for a specific application. Chief among these is the degree of parallelism, which is configured via the architectures (to choose among B512, B800, B1024, B1152, B1600, B2304, B3136, B4096), determining the number of concurrent operations. Designers can also configure the number of DPU cores instantiated, the clock frequency, and the memory interface width. Furthermore, it allows for the selection of supported data types (e.g., INT8) and the enabling of specific functions like depth-wise convolution. These parameters collectively enable a tailored balance between processing speed, power consumption, and FPGA logic resource usage.

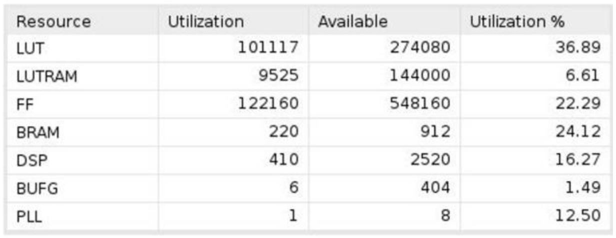
Co-funded by the European Union
6GSNS
Project funded by
UK Research and Innovation

All these configuration choices must be made before compiling the FPGA design, which is a time-consuming process (it might take several hours on a standard computer). Therefore, it is essential to know the requirements beforehand to avoid repeated recompilations. For this project, a good trade-off was found to be:

- Architecture: B4096
- Number of cores: 3
- Ram usage: low
- Conv RelU Type: ReLU + LeakyReLU + ReLU6
- Alu parallel: 4

Once the design is compiled, Vivado returns not only the resulting bitstream (the file used to program the FPGA) but can also generate reports on resource usage, power consumption, and several other metrics. Figure 10 shows some of these outputs, including a view of the logic resource placement on the device (left) and a table with specific resource utilization numbers (right).





Resource	Utilization	Available	Utilization %
LUT	101117	274080	36.89
LUTRAM	9525	144000	6.61
FF	122160	548160	22.29
BRAM	220	912	24.12
DSP	410	2520	16.27
BUFG	6	404	1.49
PLL	1	8	12.50

Figure 10: Utilization summary of FPGA resources (left: device layout; right: usage percentages)

Finally, Vivado can estimate power consumption based on the logical resources that constitute the design. The resulting power estimation report shows a total on-chip power of 8.63 W, of which only 1.2% is attributed to the Processing System (ARM).

4.3.1.2 Petalinux as an Operating System

Obtaining the FPGA compiled design, which includes the DPU, was only the first step in preparing the platform to run AI models on it. Figure 11 shows what has been covered so far by compiling the design (highlighted in purple), as well as what is yet needed to finally run software on the platform (highlighted in green). This missing piece is the operating system that will run on the device. In this case, a Petalinux [32] distribution is chosen as it is the one recommended by Xilinx and widely supported by Vitis AI applications and models.

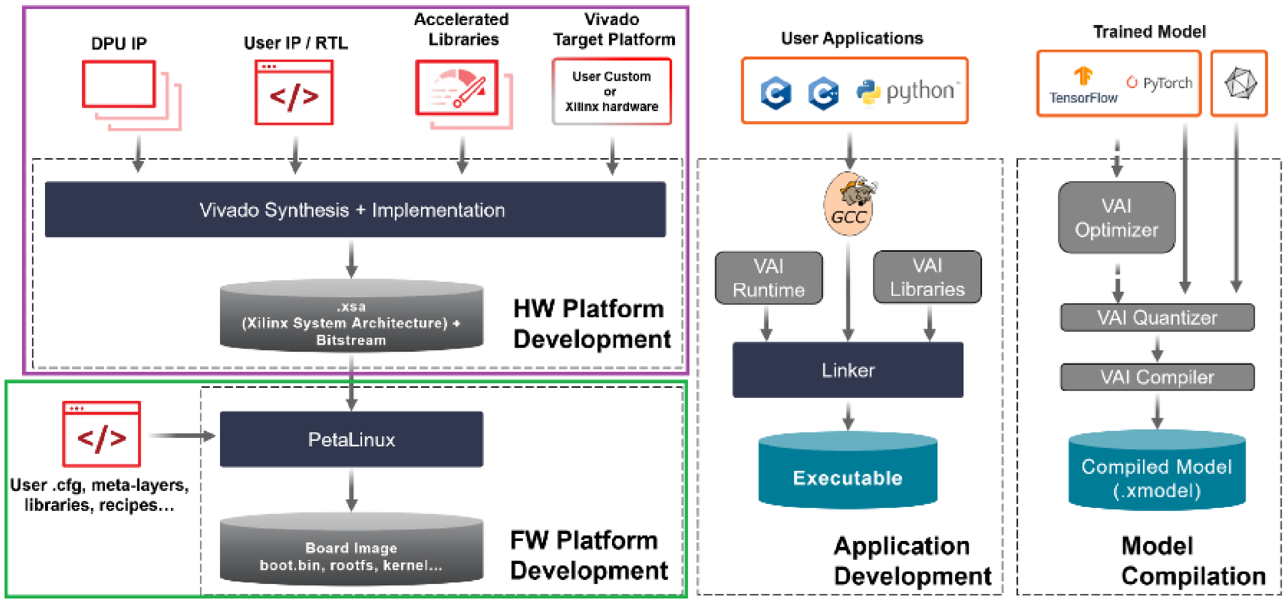


Figure 11: Block diagram of all pieces needed to run an AI model in a SoC

PetaLinux is a Linux development environment specifically for Xilinx FPGAs and adaptive SoCs. It provides the necessary tools to build, customize, and embed a complete Linux distribution, including the kernel, device drivers, and bootloader, tailored to run on a specific hardware design. In this case, the following Petalinux version is used: *honister-2022.2_release_S10071807*, fully compatible with Vitis-AI 3.5.

Since the operating system requires specific libraries and tools tailored for the project, it must be configured and built from the source code. Vivado provides the PetaLinux Tools to simplify this process. The most relevant details of the operating system configuration used are listed below:

- Enable the drivers for the DPU, which are installed at the kernel level, allowing software applications to communicate directly with the DPU resident in the FPGA.
- Add the Vitis-AI, OpenCL, and OpenCV libraries.
- Install other typical Linux operating system utilities: dnf, nfs, resize, cmake, x11, gstreamer, etc.
- Some other minor tweaks are useful for development.

After configuration, Petalinux is built, and the following files are generated:

- BOOT.BIN: Initial bootloader containing FSBL (First Stage Bootloader), bitstream (to program FPGA), and U-Boot.
- boot.scr: U-Boot script defining boot commands and environment variables.
- Image: Compressed Linux kernel image.
- system.dtb: Device Tree Blob describing hardware configuration for the MPSoC system.

Co-funded by the European Union 6GSNS Project funded by UK Research and Innovation

- rootfs: Root Filesystem containing operating system directories, libraries, applications, and user space components.

All these files are copied to an SD card and used to boot the target device. Once the image is loaded, the system is ready to run XMODELs compiled by Vitis AI, as detailed in the following section. This image creation has been validated using both QEMU (a processor emulation system) and physical ZCU hardware.

4.3.2 Vitis AI workflow

The Vitis AI Optimizer (VAI_OPTIMIZER) is a framework designed to enhance neural network models by eliminating redundant components, such as unnecessary convolutional kernels or channels. Its primary objective is to reduce computational complexity and memory usage during inference, while preserving model accuracy.

As shown in Figure 12, the optimizer includes a single tool, the Pruner. Pruned models can subsequently undergo quantization and be deployed on AMD hardware platforms, including FPGAs, SoCs, and adaptive SoCs.

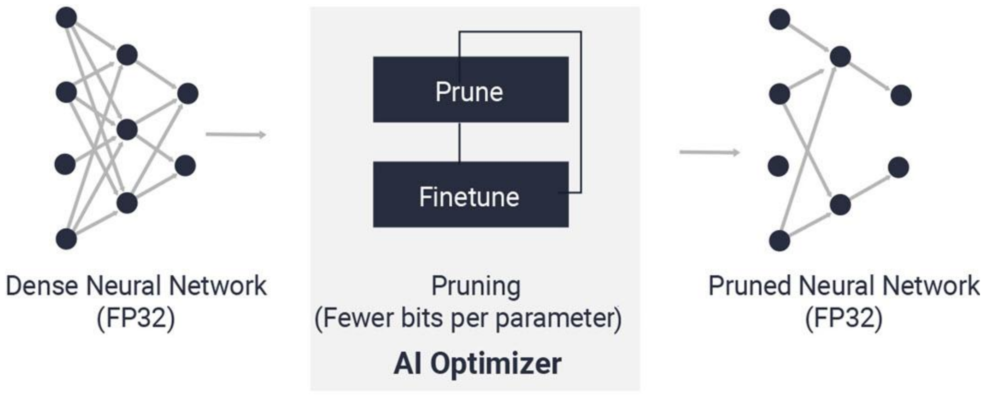


Figure 12: Optimizer methodology

The Vitis AI Optimizer supports two main families of pruning approaches: coarse-grained pruning (also known as channel pruning) and Neural Architecture Search (NAS)–based methods. In coarse-grained pruning, entire filters, channels, or convolutional kernels are removed rather than individual weights, providing a balanced compromise between compression ratio, inference speed, and model accuracy.

Coarse-grained pruning in Vitis AI can be performed using two alternative strategies:

- Iterative Pruning, which progressively removes parameters through multiple prune–retrain cycles.
- One-Step Pruning, which performs pruning and fine-tuning in a single, more efficient step.

As illustrated in Figure 13, these two coarse-grained methods differ from One-Shot NAS, which integrates pruning within a single supernet search process. The figure provides a comparative view of the workflow and execution order for each pruning strategy.

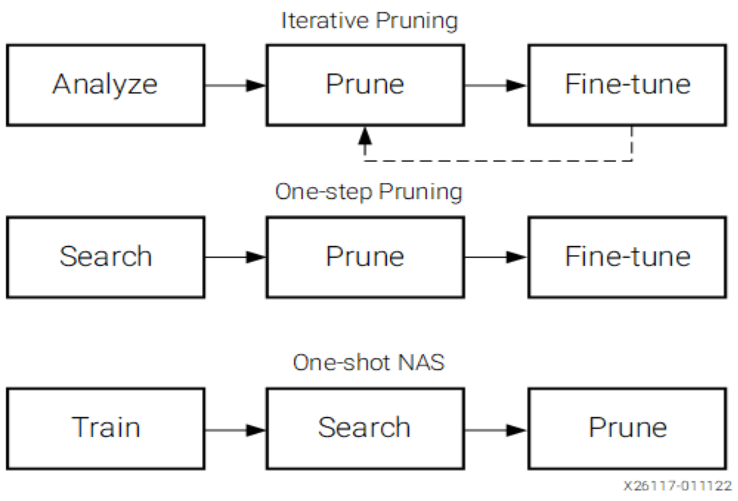


Figure 13: Workflow comparison of Iterative Pruning, One-Step Pruning and One-Shot NAS methods

Vitis AI also provides a quantizer that supports model quantization, calibration, and fine-tuning. The Vitis AI quantizer significantly reduces computational complexity while preserving the prediction accuracy nearly unchanged by converting 32-bit floating-point weights and activations to fixed-point formats like INT8. This transformation yields a fixed-point network model that requires less memory bandwidth, resulting in faster processing speed and improved power efficiency compared to the floating-point model. The quantization method implemented in this task was Post-Training Quantization (PTQ), with VAI_Q_PYTORCH, which requires multiple iterations of inference for calibrating the activations to enhance the accuracy of quantized models and capture activation statistics. This necessitates a calibration image dataset as input. Typically, the quantizer works effectively with 100–1000 calibration images, as backpropagation is unnecessary, and an unlabeled dataset serves the purpose.

The Vitis AI compiler maps the quantized AI model to an efficient instruction set and dataflow model. As shown in Figure 14, it performs optimizations, such as layer fusion, instruction scheduling, and reuses on-chip memory. This step generates an XMODEL file, which is used to initialize the set of DPU instructions corresponding to the model architecture. The compiler (VAI_C) serves as a unified interface for a family of compilers to optimize neural network computations for variousDPUs. Vitis AI supports several DPUs for different platforms and applications. The DPUCZDX8G, in particular, is designed for the AMD Zynq™ UltraScale+™ MPSoC platforms.

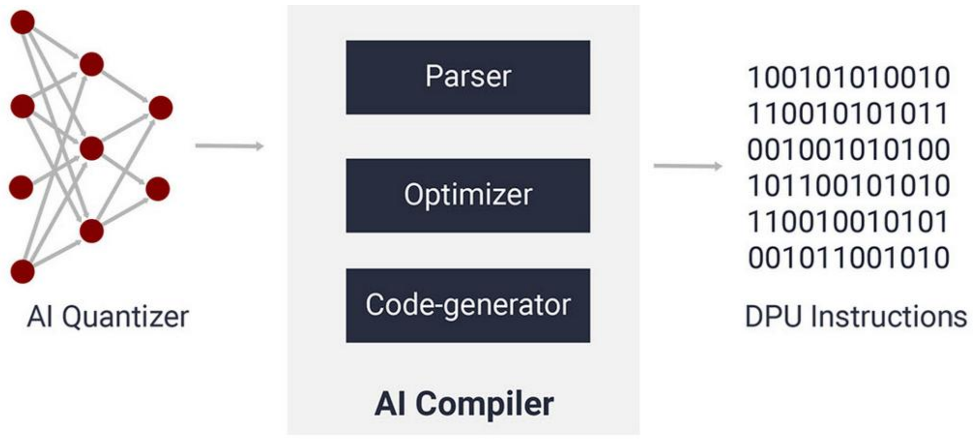


Figure 14: Quantization and compilation workflow

AMD Intermediate Representation (XIR) is a graph-based intermediate representation of AI algorithms designed for compilation and efficient deployment of the DPU on the FPGA platform. XIR includes the Op, Tensor, Graph, and Subgraph libraries, which provide a flexible representation of the computational graph. XIR has in-memory and file formats for different uses. The in-memory format XIR is a graph object, and the file format is an XMODEL.

For model deployment on the ZCU102 platform, the Vitis AI Runtime Library (VART) facilitates execution by providing a high-level API to initialize and control a DPU runner. This runner manages the interaction with the DPU, including input feeding and output retrieval. Although inference is offloaded to the DPU, preprocessing and postprocessing steps are performed on the CPU. The application was implemented in Python, utilizing the runtime Python API; equivalent functionality is also available through its C++ interface.

4.4 Preliminary results

To study the One-Step pruning, a ResNet model from the Vitis AI tutorial “ResNet18 in PyTorch from Vitis AI Library” was selected as it represents a validated and hardware-supported baseline for deployment. By starting from an officially supported model, full compatibility with the Vitis AI workflow was ensured, and framework-related issues were minimized, allowing the evaluation to focus exclusively on the effects of pruning and fine-tuning. Once the tutorial was successfully implemented and verified, the One-Step Pruning method was applied since ResNet-18 is a conventional CNN architecture with Batch Normalization in all residual blocks.

This study evaluated various configurations of pruned models, with sparsity levels ranging from 0.4 to 0.7, which indicate the percentage of the model parameters that were removed through pruning. The "Subnetworks" column in Table 2 shows the number of subnetworks evaluated before the prune step.

Table 2 summarizes the impact of pruning on the ResNet-18 model in terms of disk size after optimization, classification accuracy, the actual Frames per Second (FPS), and power consumption on the ZCU102 board.

Table 2: Combined results of baseline and pruned ResNet18 models on GPU and ZCU102 MPSoC

Model	Sparsity	Subnetworks	Size (MB)	Accuracy (%)	FPS	Power (W)
ResNet18 Baseline	-	-	42.74	88.50	418.02	33.86
ResNet_0.4	0.4	20	22.78	87.85	460.37	32.58
ResNet_0.5	0.5	20	16.77	86.63	539.00	33.09
ResNet_0.6	0.6	20	14.07	87.34	576.08	31.70
ResNet_0.7	0.7	20	8.28	86.76	637.98	31.30

Regarding the acceleration of the Jamming Detection model described in Section 4.2 on an MPSoC, the results are illustrated in Table 3. The workflow only considered the Quantization, Compilation, and deployment steps, as the Optimization study overlapped with this task in time. The tests on the MPSoC board reveal improved performance when the parallel capacity of the DPU is exploited, i.e., with four execution threads in the DPU. Although the power consumption increases slightly, this is the cost of achieving a higher frames per second rate. When analyzing the power consumed per frames in one second, it is noticed that with the ZCU102, the most significant energy savings take place.

Table 3: Acceleration results of the Jamming Detection on GPU and ZCU102 MPSoC

Hardware	Accuracy (%)	FPS	P (W)	mW/FPS
GPU	86.75	817	50.99	416.59
ZCU102	86.75	1135	25.64	22.59

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI
UK Research and Innovation

5 Energy efficient and Latency-aware workload scheduling

5.1 Proposed model and system architecture

The Energy efficient and Latency-Aware Workload Scheduling (ELAWS) proposes a system that can schedule cloud workloads across a multi-tier cloud infrastructure with different resource sites (Cloud, Fog, Edge, Device), aiming to minimize the energy consumption of workload execution while respecting latency constraints between user devices and service endpoints. Each cloud site is characterized by distinct computational capabilities and latency performance, as summarized below:

- Cloud – offers the highest computational resources, suited for heavy workloads and low execution time, but with high latency to the end-devices.
- Fog – provides moderate resources near the network core, balancing performance and latency.
- Edge – enables low-latency networking since it is placed closer to end-devices, but has limited computing capacity.
- Device – includes end-user or IoT nodes with minimal computational resources.

The system can monitor the power consumption of each node, as well as the individual container power consumption, via a popular energy monitoring tool, namely the Scaphandre [33]. These metrics are then exported to the control & decision plane of the cloud orchestrator via the Prometheus exporter.

Additionally, a custom latency monitoring service operating in real-time was designed and developed to measure the end-to-end latency between user devices and the various nodes across the different cloud sites.
Based on these stored metrics, a historical workload execution dataset was constructed, capturing the power consumption and the deployment context (co-located workloads on each node) across the various nodes in the infrastructure. Then, a machine learning (ML) model is pre-trained on this dataset to predict the energy required for the upcoming workload batches that need to be scheduled.

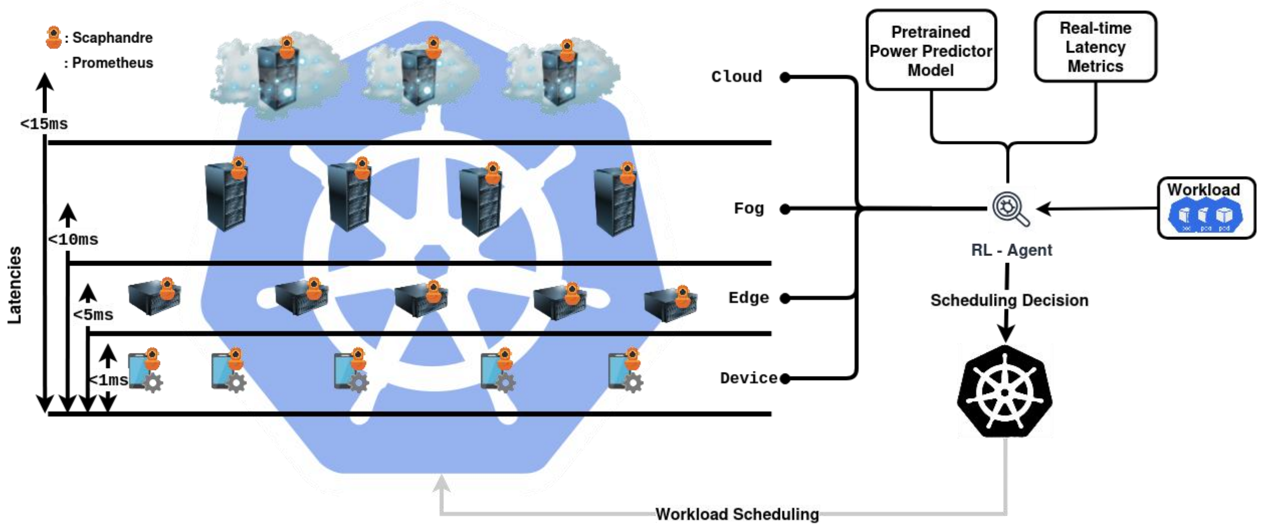


Figure 15: ELAWS system architecture

A Reinforcement Learning (RL)-based agent (Figure 16) taking as an input the output of the Energy predictor model for the workload batches and the real-time latency measurements acts as the central decision-maker, which decides on which cloud-site (Resource Level) and which node of each cloud-site (Latency constraint) each workload will be placed by informing with the generated decision to a custom Kubernetes-scheduler. The custom Kubernetes scheduler applies the scheduling decisions made by the RL agent, executing the selected workload placements in the cluster.

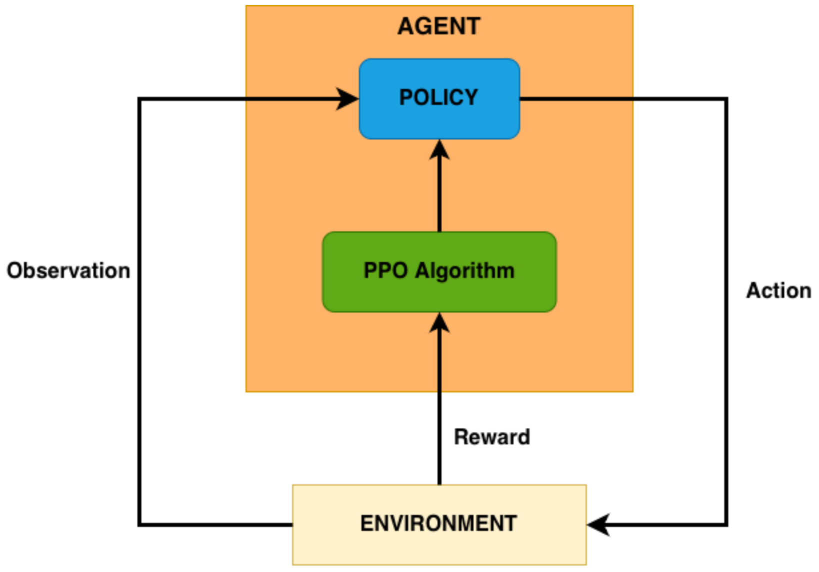


Figure 16: Architecture of the RL agent for energy and latency-aware workload placement decisions

5.2 Implementation

5.2.1 Scaphandre

For the tracking of the power consumption of scheduled workloads, ELAWS utilizes Scaphandre, which provides power monitoring capabilities in Linux environments, leveraging the RAPL (Running Average Power Limit) sensor to continuously measure the overall energy consumption of the host system. A Scaphandre agent needs to be installed on each bare-metal node. Scaphandre is able to estimate the power consumption of each process from the total CPU power measured by RAPL, with low overhead and high accuracy over time. To obtain the per-container (pod-level) power consumption, ELAWS aggregates the estimated process-level power metrics for all processes belonging to each container. A Scaphandre deployment (Figure 17) is created in the cluster, which spawns a Scaphandre pod on each node. Each pod communicates with the node's local Scaphandre agent and exports the aggregated metrics through the Prometheus exporter, enabling real-time collection and analysis by the cloud orchestrator.

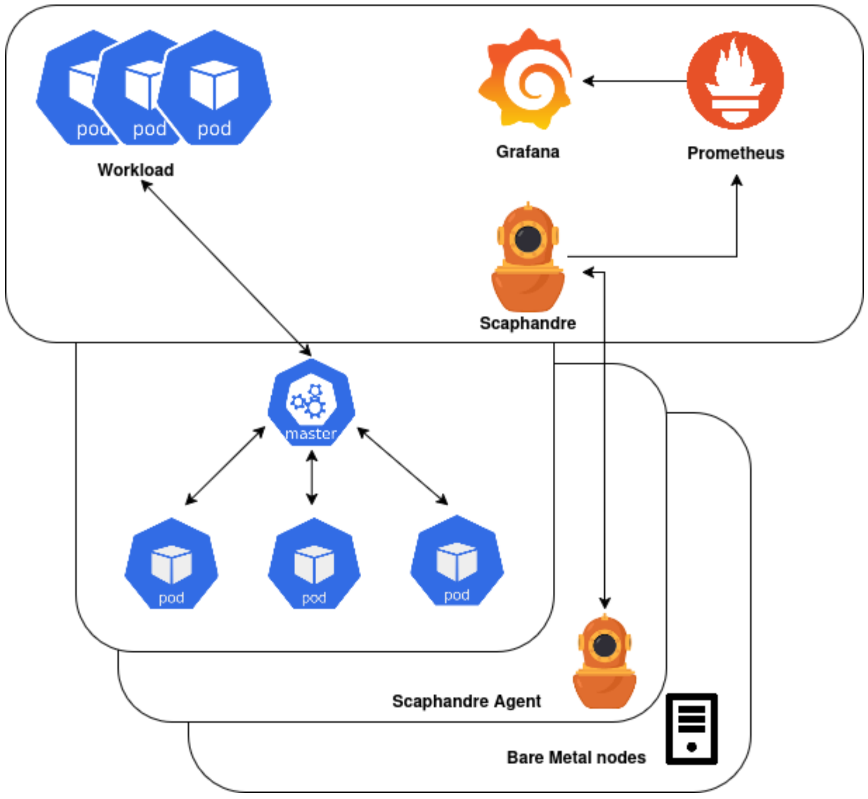


Figure 17: Scaphandre deployment and interconnection in Kubernetes Cluster

5.2.2 Latency Module

To implement the custom Latency Monitoring Module within the Kubernetes cluster, we deployed a pod on every node in the cluster responsible for either generating or responding to

ICMP packets to measure the Round-Trip Time (RTT) among the cluster nodes. Each Latency Measurement Pod is configured via the Multus CNI plugin to attach a secondary network interface (net1) with a statically assigned IP address, ensuring deterministic Layer-2 connectivity between nodes across different cloud sites. The key operational behavior of this module is that only the pods deployed on the end-devices ping all the non-end-device nodes (Edge, Fog, and Cloud) to generate ICMP echo requests and measure the RTT. These measurements are executed at a 1-second interval, and the resulting RTT values are aggregated locally within each pod. The collected RTT measurements are aggregated in a 10-second time window, computing the mean RTT and then exposing it through the Prometheus exporter, which supports the real-time latency monitor required by the RL-agent.

5.2.3 Workload Energy Consumption Predictor

For the RL agent to make optimal placement decisions for the workloads, the expected energy consumption of all candidate deployments must be provided as input. To estimate these values across multiple workload scenarios prior to scheduling, we implemented a machine learning model based on an XGBoost regressor. As already mentioned, the dataset used to train the XGBoost ML model contained metrics from past workload execution scenarios. Each training sample includes the workload type (pod_type), CPU allocation (cpu_millicores), node identifier (node_name), and the type and number of co-located pods running on the same node. Using these features, the model can predict the expected energy consumption (pod_energy_joules) for any workload–node pair. The predictor achieved a Mean Absolute Percentage Error (MAPE) of approximately 10%. The predicted energy values are then provided to the RL-based scheduling agent, which incorporates them into its energy–latency optimization policy for workload placement.

5.2.4 Custom Kubernetes scheduler based on RL-agent

In order to support intelligent workload placement decisions that jointly optimize for energy efficiency and latency performance, a custom Kubernetes scheduler was designed and integrated within the ELAWS framework. This scheduler is governed by an RL agent, which operates as the central decision-making component. The RL agent is responsible for determining optimal workload-to-node assignments by considering both predicted energy consumption and real-time latency constraints, translating these into scheduling decisions executed within the Kubernetes control plane.

Upon the submission of a new batch of workloads, the custom Kubernetes scheduler initiates a placement request to the RL agent through a dedicated API interface. The agent processes a rich system state composed of the following elements:

- Predicted energy consumption values for each workload-node pair, inferred via the XGBoost-based energy predictor.
- Real-time latency measurements between user endpoints and cluster nodes, collected by the latency monitoring module.
- Node-level resource utilization metrics (CPU, MIPS, co-located workloads), exposed via Prometheus and processed as part of the system context.

This aggregated system state is provided as input to the policy network of the RL agent, which has been trained using the Proximal Policy Optimization (PPO) algorithm. The agent outputs a discrete action representing the selected target node for each workload. The custom scheduler then applies these actions by binding the pods to the designated nodes, enforcing the decision through Kubernetes' native scheduling primitives.

The design enforces a clear separation of concerns: the RL agent handles the control logic and decision-making, while the custom scheduler executes the final placements. This modular architecture facilitates retraining or substituting the agent independently of the orchestrator.

To continuously adapt to dynamic cluster conditions, the RL agent is trained with a reward function that jointly captures energy efficiency and latency constraint satisfaction. The reward function R for each scheduling decision is computed as follows:

$$R = \begin{cases} -\alpha \cdot E - \beta \cdot D & \text{if } D \leq L_{\text{req}} \\ -\gamma & \text{otherwise} \end{cases}$$

where:

- E is the estimated energy consumed for the workload execution,
- D is the total end-to-end latency (execution time + network RTT),
- L_{req} is the maximum latency requirement,
- α, β are tunable weights for energy and latency penalties,
- γ is a fixed penalty applied when the latency constraint is violated.

This reward formulation incentivizes the agent to minimize energy consumption and latency while avoiding placements that lead to SLA violations. The reward is computed using the post-placement metrics exported by Scaphandre and the latency monitoring service, ensuring realistic and high-fidelity feedback.

A schematic representation of the RL-based scheduling architecture is depicted in Figure 15, outlining the flow from workload submission, metric collection, energy prediction, RL-based decision-making, and final scheduling action enforcement via the Kubernetes control plane.

5.3 Preliminary results

To evaluate the performance of the Energy Predictor ML model, we conducted a series of experiments. We explicitly trained and tested the XGBoost model on a node located in the Cloud capable of executing multiple workload types. For this purpose, we implemented an ML-driven application that performs object detection over video streams using pre-trained neural network models. Two distinct workload types were defined, corresponding to object detection on video inputs of 30 frames per second (fps) and 60 fps, representing low- and high-intensity inference tasks, respectively. A randomized sequence of workload batches combining these two workload types was generated and sequentially deployed on the target node to emulate heterogeneous and dynamic execution patterns.

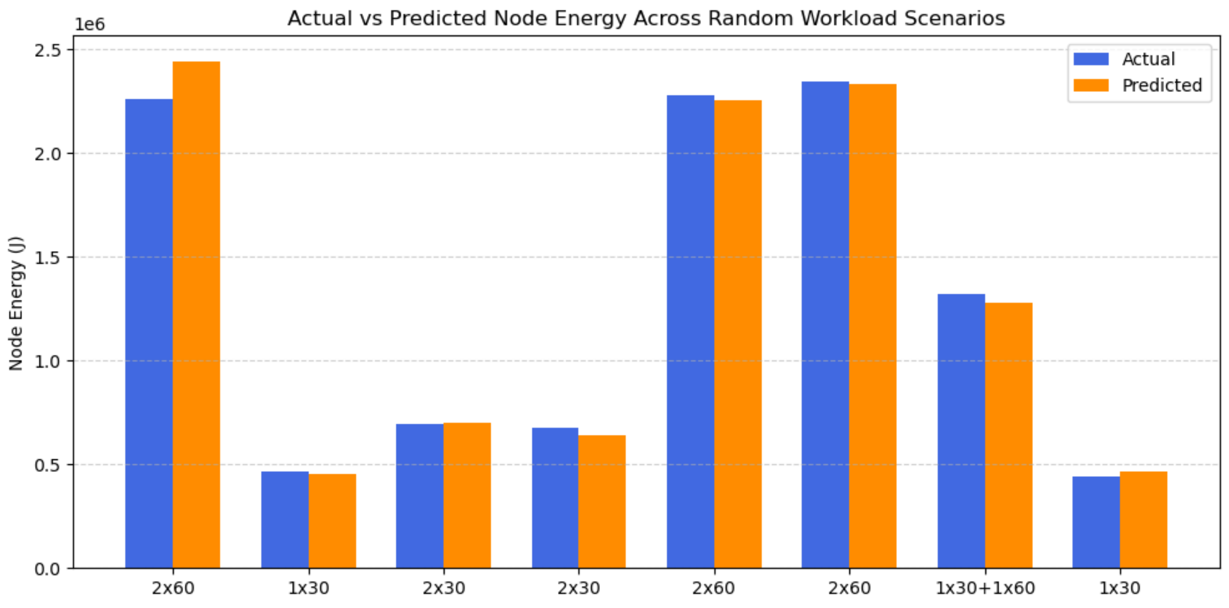


Figure 18: Experimental results for node energy consumption

In Figure 18, the comparison between the actual (sourced from the scaphandre monitoring system) and the predicted node energy consumption across a sequence of randomly generated workload scenarios is illustrated. As we observe, the model is able to estimate the energy consumption of the deployed applications, succeeding with MAPE < 5%.

6 Energy-efficient RDMA transfer

This section presents the Energy-Aware RDMA Transfer Agent (E-RTA), a software module developed to optimize RDMA transfers in terms of both performance and energy efficiency. The motivation behind this work stems from the growing need for low-latency and energy-conscious data transfer mechanisms across large-scale and heterogeneous computing environments. RDMA over Converged Ethernet (RoCE) enables direct memory access between nodes with minimal CPU involvement, but its efficiency depends heavily on runtime conditions and system configuration.

The proposed E-RTA acts as an intelligent middleware that dynamically detects RDMA-capable endpoints, adapts transfer parameters such as buffer sizes, and estimates the energy consumption associated with each transfer. The following subsections describe the agent's design, implementation, and preliminary evaluation results obtained in an RDMA-enabled edge computing testbed.

6.1 Proposed model and system architecture

The Energy-Aware RDMA Transfer Agent (E-RTA) is conceived as an intelligent intermediary layer between network orchestrators and the underlying RDMA-capable infrastructure. Its purpose is to enhance data transfer efficiency by combining adaptive configuration with real-time telemetry feedback. Upon receiving a transfer request, the agent analyzes the incoming file to determine its size and verifies whether both source and destination nodes possess RDMA-capable network interface cards (NICs). If RDMA support is available, the agent initiates an optimized RDMA-based transfer; otherwise, it falls back to TCP. The E-RTA dynamically adjusts buffer sizes and transfer settings based on file characteristics, ensuring optimal utilization of available resources and minimal latency. Once the transfer is complete, the agent estimates the energy consumed using telemetry data — CPU load and thermal metrics — collected through Prometheus and displayed on Grafana dashboards.

This architecture allows closed-loop optimization, integrating northbound communication with controllers and southbound interaction with RDMA-enabled nodes, as illustrated in Figure 19. The result is a flexible and responsive framework for adaptive, energy-aware data transfers across high-performance and edge computing environments.

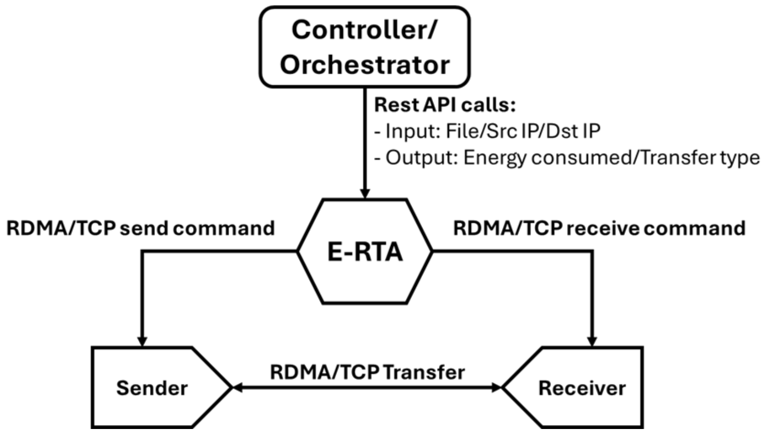


Figure 19 E-RTA functional architecture and interfaces

6.2 Implementation

The E-RTA was implemented in Python 3 and executed as a command-line tool on a dedicated control node. It coordinates transfers between endpoints by remotely executing the sender and receiver applications via secure SSH sessions. User inputs include the transfer protocol (RDMA or TCP), file path, and endpoint credentials.

For RDMA operations, the agent triggers custom-built binaries (rdma_client, rdma_server), while for TCP transfers, it uses alternate binaries (sender, receiver). Execution is carried out with elevated privileges, and timing data are recorded to monitor performance. The system supports automatic fallback to TCP when RDMA hardware is not available, ensuring operational compatibility.

Energy estimation is based on a lightweight telemetry-driven model, using real-time CPU temperature and CPU utilization values collected via Prometheus. The estimated energy consumption is derived from the following empirical relationship:

$$Est.\,en. = avg(T_{CPU}) \cdot CPU\ utilization\ rate\ \cdot 0.5$$

This formulation enables relative energy comparisons between transfer modes (TCP vs RDMA), without requiring direct hardware metering. The methodology follows validated approaches [34] that demonstrate the reliability of thermal and CPU activity metrics as energy proxies.

6.3 Preliminary results

The evaluation was conducted on a network testbed shown in Figure 20, using two Dell PowerEdge R760 servers, each featuring Intel Xeon Gold 6438Y+ processors (128 cores, 256 GB RAM) and NVIDIA BlueField-2 DPUs for RDMA capability. Both nodes ran Ubuntu Linux (low-latency kernel 5.15), while a local control node hosted the E-RTA agent. Prometheus continuously collected telemetry, visualized through Grafana, to correlate transfer performance and system energy trends.

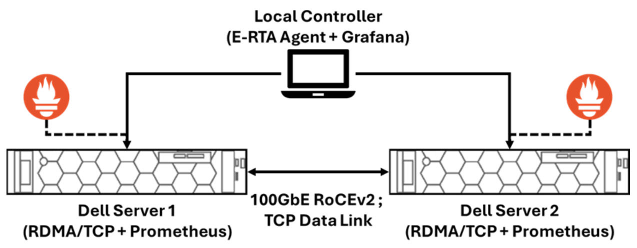


Figure 20 E-RTA Testbed setup

Three protocols were compared: RDMA Read, RDMA Write, and TCP. Across file sizes ranging from 5 MB to 50 GB, RDMA Write consistently achieved the lowest latency, followed by RDMA Read and TCP. Consequently, RDMA Write was used for further evaluations.

Subsequent experiments examined the impact of buffer size (1 MB to 1 GB) on transfer efficiency. Results showed that static buffer allocation strategies lead to inconsistent performance, with certain buffer sizes suitable only for specific file ranges. The E-RTA dynamically adjusts buffer sizes according to file characteristics, achieving more stable throughput and lower estimated energy consumption. As illustrated in Figure 21, E-RTA consumes less energy than TCP for all file sizes, particularly beyond 1 GB.

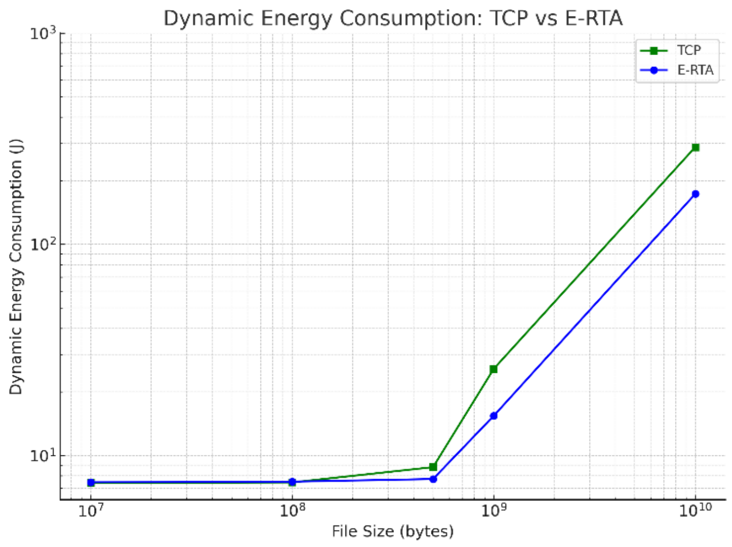
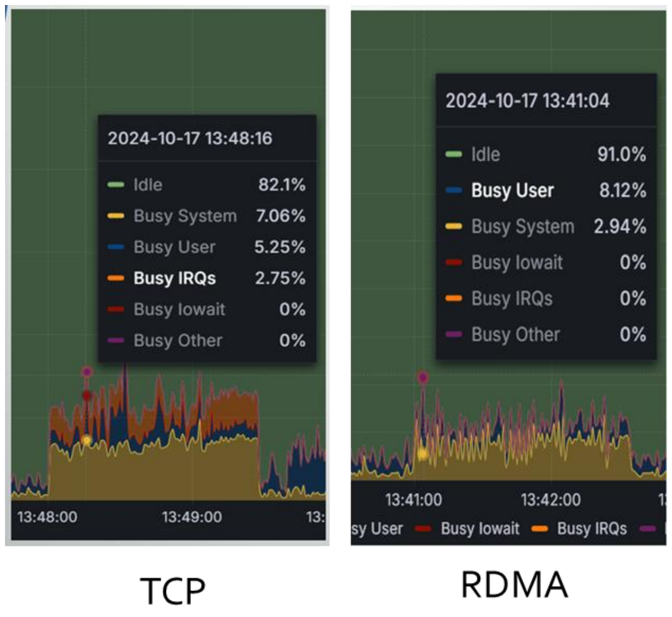


Figure 21 Analysis of energy consumption with TCP and E-RTA

The Prometheus-Grafana dashboard used to fill the parameters into the E-RTA decision model is shown in Figure 22. In particular, it shows the busy resources during TCP and RDMA transfer of the same file. The dashboard shows that RDMA minimizes the system and the IRQ resources, reducing context-based operations at the kernel level.



2024-10-17 13:48:16

Idle	82.1%
Busy System	7.06%
Busy User	5.25%
Busy IRQs	2.75%
Busy Iowait	0%
Busy Other	0%

2024-10-17 13:41:04

Idle	91.0%
Busy User	8.12%
Busy System	2.94%
Busy Iowait	0%
Busy IRQs	0%
Busy Other	0%

Figure 22 Prometheus /Grafana dashboard used to retrieve E-RTA parameters

Table 4 quantifies the energy savings across transfer modes, while Table 5 highlights E-RTA's optimal buffer selection logic, confirming its superior adaptability.

Table 4: Energy consumption used by different transfer operations

Estimated Energy (J)		**Transfer operations (TCP or RDMA)**			
		TCP	**R-W**	**R-R**	**E-RTA**
File size	**10MB**	7.4	7.5	7.6	7.5
	100MB	7.5	8.3	8.5	7.5
	500MB	8.8	9.2	9.8	7.7
	1GB	25.7	19.7	21.2	15.4
	10GB	288.7	280.5	284.5	173.2
	50GB	3032.8	2258.0	2271.3	1293.7

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI
UK Research and Innovation

Table 5: Energy consumption by E-RTA with different file dimensions and buffer sizes

Estimated Energy (J)	File Size					
	10MB	100MB	500MB	1GB	10GB	50GB
TCP	7.4	7.5	8.8	25.7	288.8	3032.8
1MB	7.5	8.3	9.1	19.7	280.5	2258.0
10MB	7.5	7.5	7.7	16.3	247.5	1503.2
100MB	7.8	8.5	8.2	15.4	224.1	1581.6
1GB	7.6	7.8	7.9	15.6	173.2	1293.7
E-RTA optimal buffer	1MB	10MB	10MB	100MB	1GB	1GB

The E-RTA was also deployed in a distributed edge computing scenario supporting a "smart mirror" application [35] where a front-end node handles user interactions and a back-end node processes images and clothing overlays. Using RDMA-based communication, the distributed system achieved latency comparable to a single-server configuration, while the E-RTA ensured efficient and energy-aware data transfer between components.

The experiments show that the proposed E-RTA outperforms traditional TCP transfers, especially for medium to large files. E-RTA achieves lower energy consumption by dynamically adjusting buffer sizes and using RDMA Write operations, which cut CPU usage and transfer time. The energy savings grow significantly for files exceeding 1 GB, demonstrating E-RTA's scalability and efficiency. Overall, the results confirm the benefits of intelligent, energy-aware data transfer mechanisms for data centers and edge systems.

Future work will focus on integrating DPUs and DOCA libraries to further boost RoCE performance through hardware acceleration and offloading, thereby enhancing both scalability and energy efficiency.

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
UK Research and Innovation

7 Net-Zero machine learning for anomaly detection

A Net-Zero Machine Learning framework is introduced for anomaly and attack detection in networked systems. This framework focuses on two main goals: (a) ensuring strong performance in intrusion detection on realistic datasets, and (b) reducing the energy consumption and carbon emissions linked to the learning process. The implementation is carried out on Nebula, a decentralized training platform, and is evaluated with two key cybersecurity datasets: CIC-IDS2017 and 5G-NIDD. CIC-IDS2017 includes labeled records of network flows that represent both benign and malicious traffic, while 5G-NIDD covers traffic and signaling behaviors pertinent to 5G/MEC deployments. Together, these datasets reflect both traditional IP network intrusion scenarios and the environments of next generation 5G edge computing.

7.1 Proposed model and system architecture

The entire system operates in a decentralized, peer-to-peer way for collaborative learning. Each node in Nebula trains an anomaly detection model on its own private data. This keeps raw traffic on the node, which is crucial for preserving data privacy and regulatory compliance. Nodes only share model parameters or gradients instead of sending data. As a result, the system works with the real-world limits of telecommunications and security systems, where it is often not possible to centralize data.

The learning process consists of three iterative phases that recur throughout various global rounds: (i) Local Training, (ii) Model Communication, and (iii) Model Aggregation. During the Local Training phase, each Nebula node refines a local classifier to differentiate between attacks and benign traffic, applying its specific subset of CIC-IDS2017 or 5G-NIDD. This is the most resource-intensive stage, which typically results in the largest energy consumption and, consequently, the greatest carbon emissions. During the Model Communication phase, nodes share their current model parameters or updates with neighboring nodes in the Nebula topology, which may be arranged in a ring, partial mesh, or other decentralized configuration. The energy expense during this phase primarily depends on the volume of data transmitted and the technology used for the connections (Wi-Fi, fiber, 5G backhaul). In the Model Aggregation phase, each node combines the received models with its own. Although this step is less demanding than local training, it still utilizes CPU/GPU resources, thus contributing to a lesser extent to the total carbon emissions.

7.2 Implementation

The proposed framework is implemented within Nebula in a manner that does not require modification of the underlying datasets and only minimal assumptions about the infrastructure. Nebula is organized into three primary components: (i) a Frontend that sets up the experiment, (ii) a Controller that initiates and manages the participating nodes, and (iii) a Core runtime that carries out decentralized learning phases (local training, model sharing, and aggregation). The sustainability module is featured in both the Frontend and the Core (Figure 23).

In the Frontend, each node is set up with its hardware specifications, the type of communication link it uses (such as Wi-Fi, 5G backhaul, or fiber), the percentage of renewable energy accessible at that location, and the estimated carbon intensity of the local power grid. The renewable energy share and the carbon intensity of the local grid determine the grams of CO2 produced for every kilowatt-hour consumed by the node. The type of communication link influences the energy cost per byte for model sharing. Collectively, these parameters enable Nebula to assess not only the energy consumption of each node but also the associated climate costs of that energy.

In the Core runtime, Nebula measures, per round and per node: (1) local training time, (2) CPU and GPU utilization, (3) instantaneous GPU power draw (in watts), (4) data volume sent and received during model exchange, and (5) the additional CPU/GPU work needed for aggregation. Using these quantities, Nebula computes training energy, communication energy, and aggregation energy for every node. For communication, the energy model multiplies the total bytes exchanged (bytes sent plus bytes received) by an energy-per-byte factor specific to the node link type. For aggregation, the model accounts for CPU and GPU usage over the aggregation interval, scaled by each node's hardware efficiency [20].

Concerning the anomaly detection models, each Nebula node is responsible for training a lightweight classifier tailored for intrusion and attack detection. The CIC-IDS2017 model evaluates per-flow statistical data to ascertain whether a flow is benign or malicious, identifying possible threats such as DDoS attacks, brute force attempts, and botnet activities. For the 5G-NIDD, each node acts as a vantage point within the 5G/MEC framework, such as an edge site or a gNodeB-like access point and focuses on training with signaling or data-plane observations that pertain to 5G-specific attack patterns, such as anomalous signaling floods. The data distribution across the nodes is intentionally non-IID; for instance, one node may primarily detect DDoS-type behavior, while another node may focus on authentication of brute-force attempts. This heterogeneity mirrors the realistic conditions found in multi-site telecom and critical infrastructure networks.

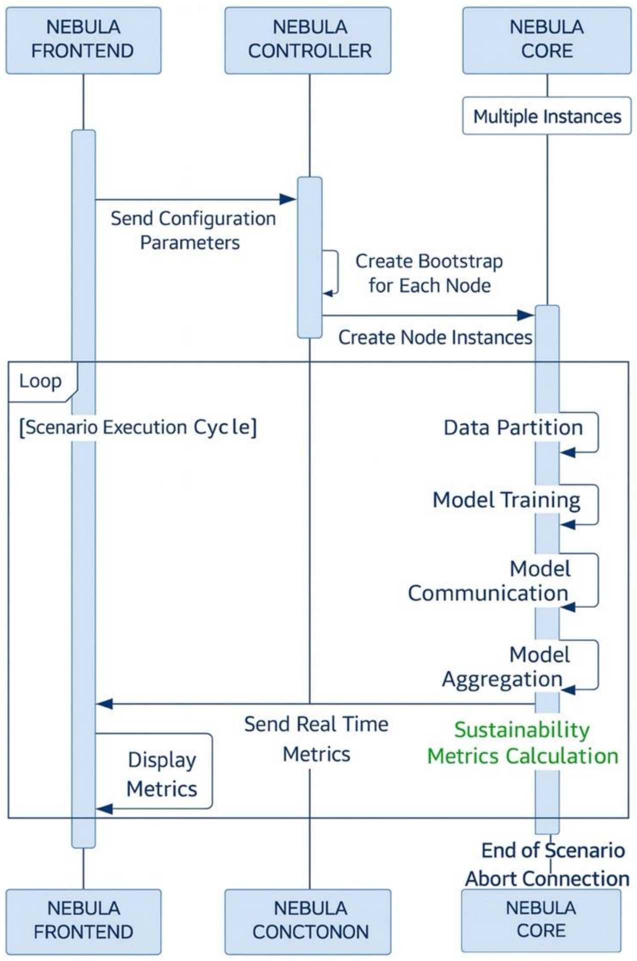


Figure 23: Nebula system workflow

7.2.1 Impact of DDoS and Jamming Attacks on Energy Consumption

Distributed Denial-of-Service (DDoS) and jamming attacks operate at different layers yet converge on a common thermodynamic outcome: they force computing, network, and radio subsystems into energetically unfavorable operating points. At the logical layer, DDoS sustains high rates of protocol work and prevents entry into low-power states; quantitative and modeling evidence shows energy-oriented DoS can roughly double CPU and memory power by keeping servers out of idle residency and driving cache/memory churn, thereby lifting cluster draw from nominal into sustained high-power regimes [36]. In constrained IoT and healthcare devices, the same pattern appears at a smaller scale: protocol choice, payload size, and port state materially affect per-request energy because malformed or repeated handshakes trigger cryptographic and error-handling paths whose cost compounds under flood conditions [37]. At the physical layer, reactive or barrage jamming achieves a favorable energy leverage because a comparatively small interferer can hold many receivers in high-duty carrier sensing and failed decoding, increase retransmissions, and prevent efficient PRB scheduling; optimization analyses of jamming confirm that careful timing maximizes the victim's retransmission cost per jammer watt [38], while control-systems studies show that jamming-induced channel mismatch and packet losses provoke repeated re-initialization and sustained estimator/controller work with non-trivial energy overheads that accumulate over minutes of instability [39]. In O-RAN/fronthaul, this dynamic manifests as RU/DU timing instability detectable in near-real-time telemetry; practical near-RT RIC stacks and xApp exemplars demonstrate sub-second anomaly detection on CQI/RSRP/CSI streams with modest platform overhead, enabling corrective PRB or beam actions before interference drives the cell into prolonged high-duty waste [40][41]. The net effect, absent mitigation, is that a compute cluster that idles around a kilowatt can be driven to multi-kilowatt sustained draw under DDoS (consistent with the "~2× dynamic" increases measured at component level) [36], while a cell that would otherwise schedule sparsely is coerced into repeated resynchronization and high-duty reception—both pathways converting adversarial activity into real kilowatt-hours and elevating the risk of thermal stress and power-budget excursions that early EC-DoS models anticipated [36].

7.2.2 Motivated design and energy model for XGBoost-on-SmartNIC (DDoS) and CSI-driven near-RT RIC xApp (Jamming)

The design choice is to put the detector where the attack must flow and on hardware whose incremental power is orders of magnitude smaller than the energy we avoid. For DDoS, a gradient-boosted tree ensemble (XGBoost) over tabular flow/packet features is a strong option because boosted trees combine high detection fidelity with very low inference cost and map efficiently to FPGA pipelines. Multiple studies demonstrate that FPGA realizations of tree ensembles deliver order-of-magnitude higher throughput than CPUs and dramatically lower energy per inference when tree nodes are placed in on-chip memory; speedups up to ~20× over a 10-threaded CPU on CPU-FPGA platforms and substantial energy reductions versus a high-performance CPU have been reported, with microsecond-class latency suitable for in-line inspection [42][43][44]. From a power-budget standpoint, modern P4-programmable/FPGA SmartNICs are designed to remain within the PCIe slot envelope for standard datapaths and scale into the "tens of watts" for complex images; vendor documentation shows low-profile FPGA SmartNIC images under roughly 25 W with auxiliary 12 V connectors available for heavier loads, which is the correct ballpark for 100 GbE in-NIC analytics [44][49]. Crucially, the literature on DPU/SmartNIC offloads provides measured deltas that anchor the benefit side: offloading OVS or IPsec from host CPUs to NIC/DPU silicon reduces server power by ~127 W and up to ~247 W, respectively, by eliminating kernel dataplane churn and crypto hot paths during heavy traffic [45]. These measurements are directly relevant because energy-oriented DDoS raises compute-plane work in precisely those subsystems; catching and filtering at the NIC avoids pushing host cores into the high-power region.

A conservative, transparent energy model follows. Consider a compact XGBoost model with 200 trees of depth 6 compiled into an FPGA pipeline on the SmartNIC. Results from CPU-FPGA tree-ensemble inference justify treating per-inference energy as far smaller than CPU traversal due to pointer-free comparators in BRAM/DSP and streaming execution [42][43]. At 100 GbE, the incremental SmartNIC power for the detector and necessary match-action stages is plausibly in the additional 5–15 W range beyond the base NIC image, consistent with FPGA/DPU card documentation and empirical ranges for custom datapaths [44][49]. On the savings side, EC-DoS measurements indicate ~2× dynamic power elevation at the component level when low-power residency is lost [36], and DPU offload studies show hundreds of watts saved under equivalent dataplane work [45]. If detection and enforcement occur within 5–10 s of onset, the avoided draw for a mid-size cluster is on the order of 1.3–1.8 kW for the remainder of the event; for a 30-minute flood, this yields roughly 40–55 kWh avoided versus at most a few hundredths of a kWh to keep the on-NIC classifier continuously active, an energy return that lies comfortably in the “thousands-to-one” range given these inputs [36][45]. Facility-side effects further amplify this because cooling and conversion overheads typically add 10–30% to IT power; DOE and utility best-practice guides codify this relationship via PUE, implying that each IT-side kilowatt avoided yields roughly 1.1–1.3 kW at the meter [46].

For jamming, the detector belongs in the near-RT RIC so it can ingest CSI/KPM streams and actuate scheduler-level changes well inside the near-RT control budget. A compact time-series model (for example, a GRU/LSTM or change-point detector over CSI eigen-structure and CQI trajectories) can flag narrowband or reactive interference within tens of milliseconds. The near-RT RIC/xApp platform is designed for such closed loops over E2, and cellular security exemplars report <2% CPU and <100 MB memory overhead while inspecting L3/L2 telemetry at line rate; on a 200 W server, that maps to ~4 W per xApp instance, which is negligible relative to jam-induced waste [50]. Actuation through PRB reallocation, beam nulling, or adaptive carrier selection prevents prolonged interference lock-in. Orthogonal O-RAN studies aimed at energy efficiency (rather than security) empirically validate that the same actuation channel materially changes the RAN power trajectory: xApps that govern PRB occupancy and radio-card sleep report substantial power reductions in simulation and testbeds, and operator-data-driven PoCs show double-digit energy savings on capacity layers without harming accessibility, establishing that near-RT policy has the leverage needed to meaningfully alter energy consumption under disturbance [47][48]. The physical-layer motivation is that reactive jamming can keep many UEs in high-duty receive and provoke repeated RU/DU resynchronizations with measurable energy overheads per cycle; detecting within 100 ms and shifting PRBs or beams collapses the loss to a few transients while the continuous compute cost of the xApp remains in the “few-watts” regime

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

[38][39][40]. Moreover, recent work shows that under certain massive-MIMO uplink regimes the system can harvest part of the jammer's RF energy, yielding non-zero capacity and improving the net energy balance during mitigation windows—further evidence that the defended system can claw back energy even in hostile spectra.

7.2.3 Conclusion (net energy savings with detection and blocking)

The energy calculus is decisively asymmetric. On the wireline side, energy-oriented DDoS transforms spare compute capacity into heat by defeating power management, while SmartNIC/FPGA-resident XGBoost adds only single-digit to low-tens of watts to halt the escalation; measured offloads show hundreds of watts saved even for individual dataplane functions, and EC-DoS models explain the remaining kilowatt-scale savings once low-power residency is preserved [36],[45]. On the radio side, a small jammer can impose a multi-device, multi-node energy penalty, but a CSI-driven near-RT xApp can restore normal PRB utilization within the near-RT control loop at a cost of only a few watts of compute per cell; empirical O-RAN studies of energy-aware xApps corroborate that such near-RT control materially reduces RAN power, and security xApps demonstrate that the platform overhead is minimal [47],[48],[50]. Accounting for facility PUE, each avoided IT kilowatt translates into roughly 1.1–1.3 kW avoided at the meter, so the "watts-to-save-kilowatts" conclusion is not only mechanistically sound but also economically persuasive under realistic data-center and RAN deployments [46].

7.3 Preliminary results

This section provides a summary of initial experimental findings utilizing Nebula on CIC-IDS2017 (traditional IP network intrusion detection) and 5G-NIDD (5G/MEC-oriented anomaly detection). The initial step of the research examines the comparative contributions of each learning phase to total energy consumption and CO_2 emissions. Table 6 displays the energy consumption and carbon emissions for a 10-node, 20-round experiment performed on the CIC-IDS2017 dataset. The Local Training phase constitutes the predominant portion of total energy consumption (about 70% or more) and total carbon emissions (about 75% or more). In contrast, the Communication and Aggregation phases contribute significantly less. This observation suggests that the primary environmental challenge in decentralized intrusion detection arises from the computationally intensive local training process, rather than from overhead communication. In practical terms, the predominant climate impact arises from the conditions under which training is conducted, rather than from the exchange of model parameters.

Table 6: Distribution of Energy Consumption and Carbon Emissions per Phase (CIC-IDS2017, 10 nodes, 20 rounds)

Phase	Total Energy E (kWh)	Share of Total Energy (%)	CO_2 Emissions (g)	Share of Total CO_2 (%)
Local Training	4.80	72.7	910	75.4
Model Communication	0.95	14.4	130	10.8
Model Aggregation	0.85	12.9	167	13.8
Total	**6.60**	**100**	**1207**	**100**

These results support the architectural choice to bias training toward lower-carbon nodes. If training dominates CO_2 emissions, then minimizing training on high-carbon nodes directly reduces the total footprint. Communication and aggregation continue to contribute, but mainly as secondary factors.

Table 7: Carbon Intensity and Training Emissions per Node in a 5G-NIDD Scenario

Node	Carbon Intensity CI_k (gCO_2/kWh)	Training Energy (kWh)	Training CO_2 (g)	Renewable Share R_k (%)
N1	420	0.55	231	10
N2	180	0.60	108	55
N3	95	0.48	45	80
N4	510	0.50	255	5
N5	160	0.52	83	60

The second part of the analysis explores how the physical and electrical context of each node influences carbon emissions. Table 7 encapsulates five exemplary Nebula nodes within a 5G-NIDD deployment. Each node represents a specific 5G/MEC site characterized by a unique energy composition. The table presents carbon intensity (gCO_2/kWh), energy consumption for training, resultant CO_2 emissions, and the proportion of renewable energy. The data indicate that two nodes executing analogous training tasks can produce significantly different emissions, contingent upon the carbon intensity of their respective energy sources. Conversely, a node that utilizes significant energy may nevertheless be classified as low-carbon if its operations are predominantly supported by renewable energy sources.

These findings highlight that net-zero anomaly detection depends not only on computational efficiency but also on location awareness. Two geographically distributed training sites performing identical tasks can impose dramatically different climate impacts depending on the

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

grid composition. This aspect is especially significant in 5G and MEC environments, where training and inference are closely linked to diverse power infrastructures.

8 Conclusions

This deliverable presented the design, development, and preliminary validation of energy-efficient AI-based security mechanisms within Task 5.2 of the NATWORK project. The work addressed multiple layers of the programmable network and compute continuum — from data-plane ML inference and hardware-accelerated signal analysis to orchestration-level workload scheduling and energy-aware data transfer — contributing to NATWORK's objective of enabling secure, autonomous, and Net-Zero-aligned 6G services.

Key contributions include in-network anomaly detection using disaggregated Net-Zero machine learning, demonstrating that programmable switches can execute ML inference with minimal additional energy by leveraging unused hardware resources. The approach integrates online learning, seamless model updates, and distributed inference, achieving high accuracy while minimizing control-plane overhead.

Neural network acceleration for jamming detection on MPSoC platforms illustrated that FPGA-based DPUs can substantially improve throughput and energy efficiency compared to GPU processing. Pruning and quantization further reduced resource consumption without compromising classification accuracy, confirming the suitability of MPSoCs for low-power security analytics at the network edge.

An energy- and latency-optimized orchestration framework was developed for heterogeneous cloud–fog–edge infrastructures. By combining real-time latency monitoring, container-level energy telemetry, ML-based energy prediction, and RL-driven scheduling, the framework enables adaptive workload placement that reduces energy consumption while maintaining strict QoS constraints.

Energy-aware transport-layer mechanisms were introduced to enable dynamic selection between RDMA and TCP transfers, runtime parameter tuning, and energy estimation through lightweight telemetry models. These mechanisms reduce CPU overhead and energy waste in high-performance cloud-edge communication.

Methodologies were also established to measure, model, and mitigate the environmental footprint of ML-based security analytics. This includes defining sustainability KPIs, quantifying the carbon and energy cost of training, inference, and distributed learning, and applying optimization techniques such as pruning, quantization, efficient model selection, and carbon-aware scheduling. Federated-learning-compatible designs ensure that distributed AI-based security functions meet both accuracy and sustainability targets.

The results demonstrate that energy-efficient AI-based security can be achieved through distributed, context-aware intelligence across the network and compute hierarchy. Implemented within programmable switches, MPSoCs, orchestration frameworks, and sustainable ML pipelines, these approaches show that high-performance security and Net-Zero objectives can be realized concurrently.

These outcomes provide a solid foundation for upcoming integration and large-scale WP6 validation activities. Future work will focus on interoperability, multi-domain experimentation, and consolidation of sustainability metrics to strengthen NATWORK's contributions to secure, autonomous, and environmentally responsible 6G infrastructures.

References

[1] Sultana, N., Sonchack, J., Giesen, H., Pedisich, I., Han, Z., Shyamkumar, N., ... & Loo, B. T. (2021). Flightplan: Dataplane disaggregation and placement for p4 programs. In 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21) (pp. 571-592).

[2] Liu, H., Chen, X., Huang, Q., Zhou, H., Zhang, D., & Wu, C. (2020). SRA: Switch resource aggregation for application offloading in programmable networks. In GLOBECOM 2020-2020 IEEE Global Communications Conference (pp. 1-6). IEEE.

[3] Zheng, C., Tang, H., Zang, M., Hong, X., Feng, A., Tassiulas, L., & Zilberman, N. (2023). DINC: Toward distributed in-network computing. Proceedings of the ACM on Networking, 1(CoNEXT3), 1-25.

[4] Busse-Grawitz, C., Meier, R., Dietmüller, A., Bühler, T., & Vanbever, L. (2019). pforest: In-network inference with random forests. arXiv preprint arXiv:1909.05680.

[5] Wang, S. Y., & Wu, Y. H. (2023). Supporting large random forests in the pipelines of a hardware switch to classify packets at 100-gbps line rate. IEEE Access, 11, 112384-112397.

[6] Zang, M., Zheng, C., Dittmann, L., & Zilberman, N. (2023). Toward continuous threat defense: in-network traffic analysis for iot gateways. IEEE Internet of Things Journal, 11(6), 9244-9257.

[7] Brandino, B., Grampin, E., Dietz, K., Wehner, N., Seufert, M., Hoßfeld, T., & Casas, P. (2024). HALIDS: A hardware-assisted machine learning IDS for in-network monitoring. In IEEE 8th TMA Conference.

[8] Xilinx. (2024). Vitis AI: accelerated AI inference on Xilinx hardware. https://github.com/Xilinx/Vitis-AI.

[9] Wen, M., Liu, X., Ning, X., Liu, C., Chen, X., Nian, J. and Cheng, L., Deep Reinforcement Learning for Energy-Efficient Workflow Scheduling in Edge Computing, Computer Networks, 2025, 111790.

[10] Benaboura, A., Bechar, R., Kadri, W., Ho, T. D., Pan, Z., & Sahmoud, S. (2025). Latency-Aware and Energy-Efficient Task Offloading in IoT and Cloud Systems with DQN Learning. Electronics, 14(15), 3090.

[11] Sahu, D., Nidhi, Chaturvedi, R., Prakash, S., Yang, T., Rathore, R. S., & Alsolbi, I. (2025). Optimizing energy and latency in edge computing through a Boltzmann driven Bayesian framework for adaptive resource scheduling. Scientific Reports, 15(1), 30452.

[12] Centofanti, C., Tiberti, W., Marotta, A., Graziosi, F., & Cassioli, D. (2023). Latency-aware kubernetes scheduling for microservices orchestration at the edge. In 2023 IEEE 9th International Conference on Network Softwarization (NetSoft) (pp. 426-431). IEEE.

[13] Furnadzhiev, R. (2025). An Experimental Evaluation of Latency-Aware Scheduling for Distributed Kubernetes Clusters. Engineering Proceedings, 100(1), 25.

Co-funded by the European Union

Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI

UK Research and Innovation

[14] Amaral, M., Chen, H., Chiba, T., Nakazawa, R., Choochotkaew, S., Lee, E. K. and Eilam, T., (2023), "Kepler: A Framework to Calculate the Energy Consumption of Containerized Applications," 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, 2023, pp. 69-71.

[15] Pijnacker, B., Setz, B. and Andrikopoulos, V. (2025). Container-level Energy Observability in Kubernetes Clusters.

[16] Pan, Y., & You, J. (2025). MP-CreditINT: Enhancing multi-path RDMA transport with credit-based congestion control and in-band network telemetry. Computer Networks, 111665.

[17] Brito, J. A., Moreno, J. I., & Contreras, L. M. (2025). Energy-Aware Edge Infrastructure Traffic Management Using Programmable Data Planes in 5G and Beyond. Sensors, 25(8), 2375.

[18] Guaitolini, M., Khan, A. H., Le Rouzic, E., Paolucci, F., & Cugini, F. (2024). Virtual Try-On Application leveraging RoCE in Low-latency Edge Computing Networks. In 2024 24th International Conference on Transparent Optical Networks (ICTON) (pp. 1-4). IEEE.

[19] Rakine, I., Oukaira, A., El Guemmat, K., Atouf, I., Ouahabi, S., Talea, M., & Bouragba, T. (2025). Comprehensive Review of Intrusion Detection Techniques: ML and DL in Different Networks. IEEE Access.

[20] Feng, C., Celdrán, A. H., Cheng, X., Bovet, G., & Stiller, B. (2025). GreenDFL: a Framework for Assessing the Sustainability of Decentralized Federated Learning Systems. arXiv preprint arXiv:2502.20242.

[21] Bouke, M. A., & Abdullah, A. (2024). An empirical assessment of ML models for 5G network intrusion detection: A data leakage-free approach. e-Prime-Advances in Electrical Engineering, Electronics and Energy, 8, 100590.

[22] Maiga, A. A., Ataro, E., & Githinji, S. (2024). XGBoost and Deep Learning Based-Federated Learning for DDoS Attack Detection in 5G Core Network VNFs. In 2024 6th International Conference on Computer Communication and the Internet (ICCCI) (pp. 128-133). IEEE.

[23] Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green ai. Communications of the ACM, 63(12), 54-63.

[24] Schmidt, V., et al. (2021). CodeCarbon: Estimate and track carbon emissions from machine learning computing. https://github.com/mlco2/codecarbon

[25] Anthony, L. F. W., Kanding, B., & Selvan, R. (2020). Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. arXiv preprint arXiv:2007.03051.

[26] Budennyy, S. A., Lazarev, V. D., Zakharenko, N. N., Korovin, A. N., Plosskaya, O. A., Dimitrov, D. V. E., ... & Zhukov, L. E. E. (2022). Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai. In Doklady mathematics (Vol. 106, No. Suppl 1, pp. S118-S128). Moscow: Pleiades Publishing.

[27] Beltrán, E. T. M., Bovet, G., Pérez, G. M., & Celdrán, A. H. (2025). NEBULA-Decentralized Federated Learning for Heterogeneous Networks. In Proceedings of the ACM SIGCOMM 2025 Posters and Demos (pp. 149-151).

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

[28] Gergely, Sarkozi, et al. (2025) "In-Network Attack Detection Using Disaggregated Inference & Online Learning" GLOBECOM 2025-2025 IEEE Global Communications Conference. IEEE, 2025. [Accepted]

[29] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp, 1(2018), 108-116.

[30] Liu, L., Engelen, G., Lynar, T., Essam, D., & Joosen, W. (2022). Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018. In 2022 IEEE Conference on Communications and Network Security (CNS) (pp. 254-262). IEEE.

[31] Osiński, T., Palimąka, J., Kossakowski, M., Tran, F. D., Bonfoh, E. F., & Tarasiuk, H. (2022). A novel programmable software datapath for software-defined networking. In Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies (pp. 245-260).

[32] Xilinx (2024). *PetaLinux tools: Embedded Linux system development*. https://github.com/Xilinx/PetaLinux

[33] Petit, B., (2024). Scaphandre, https://github.com/hubblo-org/scaphandre

[34] M. Fahad, A. Shahid, R. R. Manumachu and A. Lastovetsky "A comparative study of methods for measurement of energy of computing", Energies, 2019, 12(11), 2024.

[35] M. Guaitolini, A. H. Khan, E. Le Rouzic, F. Paolucci and F. Cugini "Virtual try-on application leveraging RoCE in low-latency edge computing networks", in Proc. 24th international conference on transparent optical networks (ICTON), IEEE, 2024.

[36] Palmieri, F., Ricciardi, S., & Fiore, U. (2011). Evaluating network-based DoS attacks under the energy consumption perspective: new security issues in the coming green ICT area. In 2011 international conference on broadband and wireless computing, communication and applications (pp. 374-379). IEEE.

[37] Alwaisi, Z., Soderi, S., & Nicola, R. D. (2023). Energy cyber attacks to smart healthcare devices: a testbed. In International Conference on Bio-inspired Information and Communication Technologies (pp. 246-265). Cham: Springer Nature Switzerland.

[38] Amuru, S., & Buehrer, R. M. (2014). Optimal jamming strategies in digital communications—Impact of modulation. In 2014 IEEE Global Communications Conference (pp. 1619-1624). IEEE.

[39] Knorn, S., & Teixeira, A. (2019). Effects of jamming attacks on a control system with energy harvesting. IEEE Control Systems Letters, 3(4), 829-834.

[40] srsRAN (2025). Near-RT RIC/xApp notes (E2/KPM/RC service models). https://docs.srsran.com/projects/project/en/latest/tutorials/source/near-rt-ric/source/index.html

[41] RIMEDO Labs (2023). Anomaly detection in O-RAN: Jamming. https://rimedolabs.com/blog/anomaly-detection-in-o-ran-jamming/

Co-funded by the European Union 6GSNS Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI UK Research and Innovation

[42] Owaida, M., Zhang, H., Zhang, C., & Alonso, G. (2017). Scalable inference of decision tree ensembles: Flexible design for CPU-FPGA platforms. In 2017 27th International Conference on Field Programmable Logic and Applications (FPL) (pp. 1-8). IEEE.

[43] Alcolea, A., & Resano, J. (2021). FPGA accelerator for gradient boosting decision trees. Electronics, 10(3), 314.

[44] Summers, S., Di Guglielmo, G., Duarte, J., Harris, P., Hoang, D., Jindariani, S., ... & Wu, Z. (2020). Fast inference of boosted decision trees in FPGAs for particle physics. Journal of Instrumentation, 15(05), P05026.

[45] NVIDIA (2022). Increasing data center power efficiency with the BlueField DPU (White paper, OVS −127 W, IPsec −247 W). https://dcmag.fr/wp-content/uploads/2022/11/nvidia-dpu-power-efficiency-white-paper-2508650.pdf

[46] U.S. Department of Energy, Federal Energy Management Program. (2024). Energy efficiency in data centers. https://www.energy.gov/femp/energy-efficiency-data-centers

[47] Liang, X., Al-Tahmeesschi, A., Wang, Q., Chetty, S., Sun, C., & Ahmadi, H. (2024). Enhancing energy efficiency in O-RAN through intelligent xApps deployment. In 2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 1-6). IEEE.

[48] Keysight, ONF, Rimedo, Juniper, & AirHop. (2024). Energy saving and traffic steering by O-RAN RIC xApp/rApp. RitiRAN Workshop 2024. https://ritiran.com/assets/pdf/keysight-energy.pdf

[49] Silicom. (2023). N6010 Agilex FPGA SmartNIC datasheet (thermal/power domain). https://www.silicom.dk/wp-content/uploads/2023/08/Silicom-FPGA-SmartNIC-N6010-Data-Sheet-v1.1.pdf

[50] Wen, H., Porras, P., Yegneswaran, V., Gehani, A., & Lin, Z. (2024). 5G-spector: An O-RAN compliant layer-3 cellular attack detection service. In Proceedings of the 31st Annual Network and Distributed System Security Symposium, NDSS (Vol. 24).

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI
UK Research and Innovation