A Benchmark Study of RF Anomaly Detection Models on NVIDIA Jetson Orin Nano

Nicholas D. Redmond*, Mohd Hasan Ali†, Dipankar Dasgupta*, Myounggyu Won*
*Department of Computer Science, University of Memphis, Memphis, TN, USA
{ndrdmond, dasgupta, mwon}@memphis.edu
†Department of Electrical And Computer Engineering, University of Memphis, Memphis, TN, USA
mhali@memphis.edu

***Abstract*—Radio frequency (RF) communication has become a cornerstone of modern critical infrastructure, supporting essential sectors including healthcare, transportation, and government services. To safeguard critical infrastructure from cybersecurity threats, numerous machine learning (ML)–based solutions have been developed. However, due to their high computational demands, many of these models are executed on cloud platforms, which may introduce latency and privacy concerns. With the rapid advancement of AI edge computing, there is a growing opportunity to deploy such solutions directly on edge devices. Despite this potential, the performance of ML models for RF anomaly detection on cutting-edge AI edge platforms like the NVIDIA Jetson Orin Nano remains largely unexplored. To address this gap, this paper presents the first benchmark study evaluating ML models for RF anomaly detection on the NVIDIA Jetson Orin Nano platform. Our results demonstrate that the device offers sufficient computational power to support real-time RF anomaly detection, achieving strong performance across multiple metrics, including latency, throughput, energy efficiency, memory usage, and CPU/GPU utilization.**

***Index Terms*—RF Communication Security, Anomaly Detection, Edge Devices, NVIDIA Jetson Orin Nano**

I. Introduction

Radio frequency (RF) communication forms the backbone of modern infrastructure, powering technologies such as 5G networks, Wi-Fi, and satellite links that underpin critical sectors including healthcare, transportation, and government systems [1]. As reliance on wireless connectivity continues to grow across these sectors, securing wireless networks has become a critical priority [2]. However, safeguarding RF communication networks is increasingly challenging with the rise of advanced machine learning (ML) techniques and the widespread availability of low-cost software-defined radio (SDR) technologies [3].

To address the challenge and strengthen the security of wireless networks, a wide range of machine learning-based solutions have been developed [4][5][6]. However, many of these ML models are computationally intensive and often rely on cloud computing resources, which can hinder their applicability in real-time scenarios due to latency and resource constraints. Recently, AI edge platforms such as NVIDIA Jetson Orin Nano [7], Google Coral [8], and Intel Neural Compute Stick 2 [9] are emerging, opening the possibility of bringing cloud-based machine learning solutions to edge devices, thereby enabling robust support for various real-time applications.

In December 2024, NVIDIA introduced JetPack 6.1, a software upgrade for the original Jetson Orin Nano Developer Kit that enhanced performance by up to 70%, significantly improving AI processing speed and memory utilization—all without requiring any hardware modifications [10]. However, to the best of our knowledge, no prior benchmark study has evaluated the performance of ML models for RF anomaly detection on this platform, likely due to its relatively recent release. To bridge this knowledge gap, this paper presents the first benchmark study of ML model performance for RF anomaly detection using the NVIDIA Jetson Orin Nano.

Our contributions are summarized as follows.

- To the best of our knowledge, this is the first benchmark study evaluating ML models for RF anomaly detection on the NVIDIA Jetson Orin Nano platform.
- Numerous machine learning models were benchmarked in this study, including the Autoencoder (AE) [11], Adversarial Autoencoder (AAE) [12], Convolutional Neural Network–based Autoencoder (CNN-AE) [13], Long Short-Term Memory–based Autoencoder (LSTM-AE) [14], and Residual Network–based Autoencoder (ResNet-AE) [15].
- A comprehensive evaluation was carried out, assessing critical performance metrics such as latency, throughput, energy efficiency, memory footprint, and resource utilization.

Our paper is organized as follows. In Section II, we review prior work on benchmark studies involving the NVIDIA Jetson Orin Nano. Section III provides an overview of the device, followed by implementation details in Section IV. Section V presents the benchmark results, and Section VI concludes the paper.

II. Related Work

In this section, we review experimental studies conducted with the NVIDIA Jetson Orin Nano. The device has been evaluated in a wide range of applications, including object

detection [16][17], surveillance [18], and speech recognition [19].

Scalcon *et al.* evaluated the NVIDIA Jetson Orin series in the context of AI and edge computing tasks, using a parking lot surveillance scenario implemented with the CVEDIA-RT platform [18]. Their study compared the Orin NX and Nano models by analyzing resource utilization metrics such as RAM, GPU, and CPU usage. Similarly, Chakraborty *et al.* provided a detailed characterization of resource behavior in NVIDIA Jetson edge devices under concurrent vision inference workloads [20]. Rey *et al.* explored the deployment of object detection models—specifically YOLOv8n and YOLOv8s—on resource-constrained edge platforms, including the NVIDIA Jetson Orin Nano, Orin NX, and Raspberry Pi 5 (RPI5) [16]. Their evaluation measured detection accuracy, inference latency, energy consumption, and the impact of post-training quantization (PTQ).

Alqahtani *et al.* benchmarked several object detection models—YOLOv8 (Nano, Small, Medium), EfficientDet Lite (Lite0, Lite1, Lite2), and SSD variants (SSD MobileNet V1, SSDLite MobileDet)—on the Jetson Orin Nano, focusing on key performance indicators such as inference latency, energy consumption, and mean average precision (mAP) [17]. Duggan *et al.* analyzed power consumption patterns across multiple convolutional neural network (CNN) architectures—including DenseNet, EfficientNet, MobileNet, ResNet, ConvNeXt, and RegNet—while executing image processing tasks on the Jetson Orin Nano [21]. In the domain of speech processing, Chakravarty evaluated the effects of quantization, memory demand, and energy consumption on the inference efficiency of automated speech recognition (ASR) models deployed on the same platform [19].

While these studies offer valuable benchmark results across a range of application domains, to the best of our knowledge, no prior work has benchmarked the performance of ML models for RF anomaly detection on the NVIDIA Jetson Orin Nano.

III. Preliminaries

This section presents an overview of the NVIDIA Jetson Orin Nano (Fig. 1). The device is designed to accelerate entry-level edge AI applications such as AI-powered robots, smart drones, and intelligent cameras. The device integrates an Ampere-architecture GPU and Arm Cortex-A78AE CPU cluster, coupled with LPDDR5 memory. It supports up to 40 trillion operations per second (TOPS) of AI performance under INT8 precision [23]. The Jetson Orin Nano Developer Kit includes interfaces for high-speed I/O, storage expansion, and multimedia support, making it a practical platform for deploying and benchmarking ML workloads [23].

The NVIDIA Jetson Orin Nano is available in two configurations: 4GB and 8GB. The 8GB variant features a 1024-core GPU based on the Ampere architecture with 32 Tensor Cores and a peak GPU clock speed of 625 MHz [23]. In contrast, the 4GB version includes a 512-core Ampere GPU with 16 Tensor Cores, also operating at a peak clock speed of 625 MHz [23]. For this study, we utilized the 8GB Jetson Orin Nano Developer Kit along with JetPack 6.1 to ensure compatibility with the target machine learning frameworks.

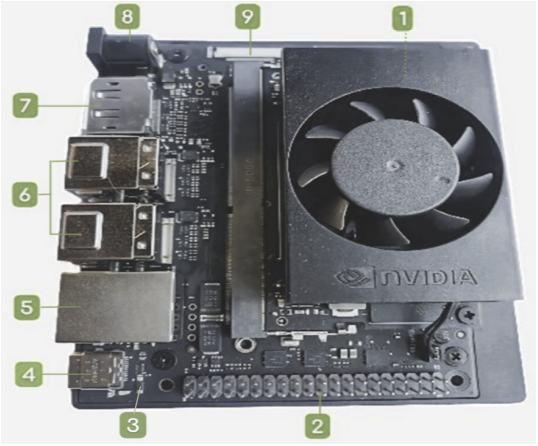

Fig. 1: NVIDIA Jetson Orin Nano Developer Kit [22]: (1) microSD card slot, (2) 40-pin expansion header, (3) power indicator, (4) USB-C port, (5) Ethernet port, (6) USB 3.1 Type A ports, (7) Display port, (8) DC Barrel jack for 14v input, (9) MIPI CSI camera connector.

IV. Implementation

This section presents implementation details related to device setup, ML models, and datasets used in this study.

A. NVIDIA Jetson Orin Nano

At the time of writing, JetPack SDK 6.2.1 is the latest production release in the JetPack 6 series. It includes Jetson Linux 36.4.4, which features the Linux Kernel 5.15 and an Ubuntu 22.04-based root file system [24]. In this study, we used JetPack 6.1 due to compatibility concerns. Specifically, several external tools and drivers were not yet updated to support JetPack 6.2, resulting in incompatibility issues that hindered stable deployment.

For our experiment, we utilized the MAXN SUPER power mode [25]. NVIDIA Jetson Linux 36.4.3 introduces support for the new high-power Super Mode on NVIDIA Jetson Orin Nano and Jetson Orin NX production modules. With Super Mode enabled, the Jetson Orin NX series achieves up to a 70% increase in AI TOPS, while the Jetson Orin Nano series delivers similar improvements in AI TOPS along with a 50% increase in memory bandwidth. This enhanced configuration enables up to 2× higher generative AI inference performance on Jetson Orin modules [25].

We also utilized the proprietary NVIDIA TensorRT engine [26] on the Jetson platform to optimize select models for high-performance inference. However, only certain models can be converted to TensorRT due to a variety of technical,

architectural, and software support constraints inherent to TensorRT's design and optimization scope [27]. The process of converting a PyTorch deep learning model into a TensorRT engine involves three main steps (Fig. 2). First, the model is exported to the ONNX (Open Neural Network Exchange) format using the torch.onnx.export() function, which traces the model execution with a given input tensor and generates a corresponding ONNX representation [28]. The ONNX model is then parsed and compiled into a TensorRT engine—a serialized form ready for optimized deployment. TensorRT can often yield significant improvements in inference throughput, especially for specific model architectures. Throughput typically increases with batch size until it reaches the hardware limits [29].

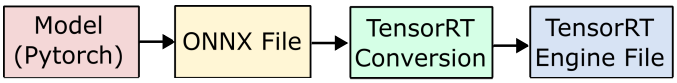

Fig. 2: The process of convering a Pytorch model into TensorRT engine file.

The ML models for RF anomaly detection were executed using one of two configurations: (1) the native PyTorch implementation and (2) a TensorRT-optimized deployment. Baseline performance measurements were obtained using standard PyTorch inference with CUDA acceleration. For TensorRT optimization, models were automatically converted to TensorRT engines as part of the experimental pipeline. This conversion process involved exporting the model to an ONNX (Open Neural Network Exchange) intermediate representation, followed by TensorRT engine compilation with workspace and layer optimization for hardware-accelerated inference.

B. ML Models

At the core of ML's contribution to real-time systems is its ability to generalize from past experiences and rapidly adapt to unseen inputs. Such dynamic adaptability is especially critical in domains like RF communication, where preprogrammed logic often fails to capture the nuance and unpredictability of real-world behavior. Autoencoders have been widely recognized for this capability, particularly in spectrum imaging and time-series analysis [30]. To evaluate model performance on the Nvidia Jetson Orin Nano, we selected five representative architectures—Autoencoder (AE) [11], Adversarial Autoencoder (AAE) [12], Convolutional Neural Network–based Autoencoder (CNN-AE) [13], Long Short-Term Memory–based Autoencoder (LSTM-AE) [14], and Residual Network–based Autoencoder (ResNet-AE) [15]. These architectures are well-established in the anomaly detection literature, and prior studies consistently demonstrate that the autoencoder family achieves high accuracy across both time-series and spectrum-based data, making them an appropriate choice for this experimental study.

C. Dataset

To train and evaluate the ML models, we generated two synthetic datasets using GNU Radio [31]. Each dataset consists of 1,000 samples, produced by synthesizing single-channel baseband signals modulated with 5G-standard quadrature phase-shift keying (QPSK). The signals incorporate realistic pulse shaping and interference patterns to emulate characteristics representative of modern 5G communication systems. For reproducibility, fixed seed values were used during generation.

Each sample contains 1,024 complex-valued in-phase and quadrature (I/Q) samples, captured at a 1 MHz sampling rate, corresponding to a time duration of approximately 1.024 milliseconds. The signals were generated at baseband (0 Hz carrier frequency) with unit power prior to normalization. This approach aligns with common digital communications simulation practices, where operations are performed in the baseband domain for computational efficiency, followed by normalization during preprocessing for model input.

All signals were generated using 4 samples per symbol and passed through a root-raised cosine pulse shaping filter with an excess bandwidth factor of 0.35. This configuration is standard in practice for producing spectrally efficient and distortion-minimized signals from digital data streams.

V. Experimental Results

A. Experimental Setup

In practical settings, RF anomaly detection is a continuous, real-time task. To emulate this operational behavior, we evaluated each model using sustained input streams rather than isolated or batch-based inference tests. For statistical robustness, each inference experiment was repeated ten times, with each run comprising 300 seconds of uninterrupted inference. To mitigate thermal effects and account for performance fluctuations during extended operation, a cooldown period was enforced between model runs. During this interval, system power consumption was monitored to ensure that the device returned to its thermal baseline before initiating the next evaluation.

After ensuring consistent thermal and computational conditions across experiments, we collected key performance metrics, including latency, throughput, energy efficiency, memory footprint, and resource utilization. These measurements were obtained using the Tegrastats utility provided by the JetPack SDK [32], which reports real-time memory usage, CPU load, and GPU activity for Tegra-based devices. To ensure statistical reliability, results were averaged over multiple runs, and standard deviations were calculated to quantify measurement variability.

B. Latency

Latency is defined as the time required for a model to process a single input sample and produce an output, measured in milliseconds. We evaluated the latency of all models,

including a simple feedforward (FF) network, which serves as a baseline reference for comparison. To assess the impact of optimization, selected models were also executed using TensorRT, highlighting the performance acceleration capabilities of the NVIDIA Jetson Orin Nano platform.

Fig. 3 presents the latency results for all evaluated models. The basic autoencoder (AE) recorded the lowest latency at 2.2 ms, which was further reduced to 0.76 ms when optimized using TensorRT. In contrast, the LSTM-based autoencoder exhibited the highest latency at 7.1 ms, primarily due to its sequential nature and recurrent computations. Notably, all models achieved inference latencies below 10 ms—well exceeding common real-time processing benchmarks such as 30 FPS (33.33 ms) and 60 FPS (16.67 ms) [33]—thereby demonstrating the Jetson Orin Nano's strong suitability for real-time RF anomaly detection.

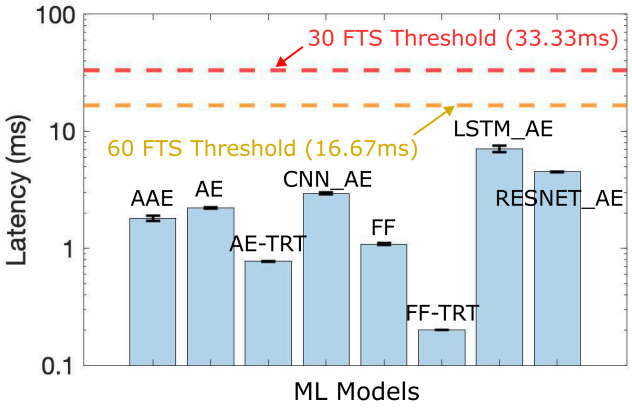

Fig. 3: Inference latency.

C. Throughput

Throughput refers to the number of input samples processed per second and is reported in frames per second (FPS). This metric is critical for evaluating the system's ability to handle continuous data streams in real time. Higher throughput translates to enhanced capacity for monitoring multiple RF channels concurrently or for analyzing high-bandwidth signals.

We evaluated the throughput of all models, with results shown in Fig.4. Consistent with the latency analysis, the TensorRT-optimized basic autoencoder (AE) achieved the highest throughput at 397.2 FPS, while the LSTM-based AE recorded the lowest at 111.4 FPS due to its sequential computation overhead. Notably, all models exceeded the real-time threshold of 60 FPS. The fact that most models sustained throughput rates above 200 FPS highlights the Jetson Orin Nano's capability to support high-volume RF data processing—an essential requirement for time-sensitive spectrum monitoring tasks in 5G environments, such as rapid jamming or spoofing detection.

D. Energy Efficiency

We measured average power consumption, defined as the mean rate of energy usage over time during model inference,

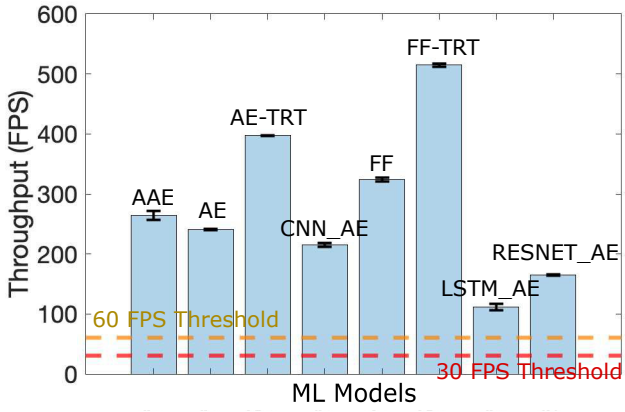

Fig. 4: Throughput.

expressed in watts (W). This metric is particularly important for edge deployment scenarios, where power budgets are limited and effective thermal management is essential.

Fig. 5 depicts the results. Among the evaluated models, the adversarial autoencoder (AAE) exhibited the highest energy efficiency, with average power consumption 11.9% lower than the TensorRT-optimized AE. This efficiency may be attributed to the AAE's architecture, which achieves comparable reconstruction performance with reduced parameter activity or lower memory access overhead. In contrast, more complex models—such as LSTM_AE and RESNET_AE—demonstrated reduced energy efficiency. In particular, LSTM-based models consumed similar power but delivered lower throughput, due to their sequential processing nature and recurrent computations, which limit parallelism and increase inference time. These results underscore the importance of selecting model architectures that balance accuracy with energy efficiency for deployment in resource-constrained edge environments. Overall, all models' average power consumption stayed below the 5W threshold, which ensures optimal thermal performance and extended battery life in mobile applications. according to the Jetson Orin Series Developer's Guide [34].

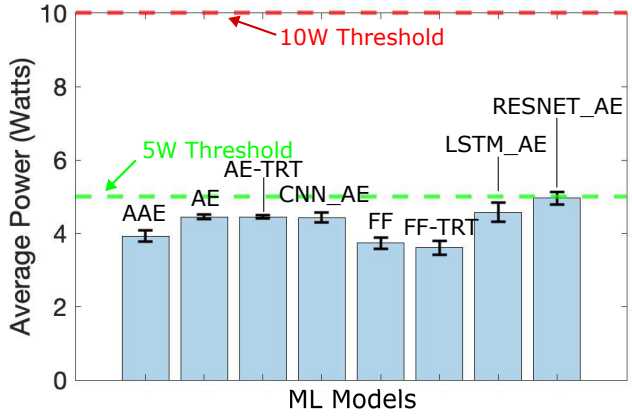

Fig. 5: Power consumption.

Fig. 6 illustrates the performance–power trade-off across all evaluated models. Notably, the adversarial autoencoder (AAE) stands out for its energy efficiency, maintaining high throughput while drawing less power than its counterparts. Interestingly, while the AE model optimized with TensorRT achieves a substantial increase in throughput, it does not exhibit a proportional improvement in energy efficiency. Overall, all models exceed the target throughput threshold while remaining below the average power consumption limit, demonstrating that each configuration is viable for real-time RF anomaly detection on edge devices within constrained power budgets.

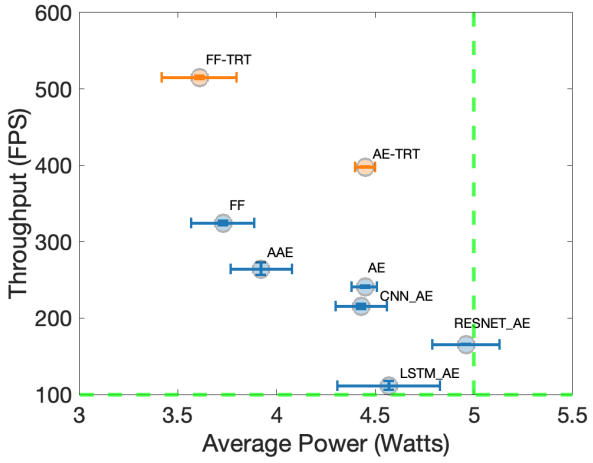

Fig. 6: Performance vs power trade-off.

E. Memory Footprint

We evaluated the GPU memory usage of all models on the Jetson Orin Nano device, with results presented in Fig. 7. Notably, both the basic AE and CNN-AE models exhibit peak memory usage around 17 MB. Interestingly, the TensorRT-optimized AE (AE-TRT) also maintains a similarly high memory footprint, suggesting that TensorRT optimization, while improving throughput, does not necessarily reduce GPU memory usage. This may be due to the allocation of additional intermediate buffers or workspace memory required during TensorRT's optimization and execution phases. On the other hand, the LSTM-based autoencoder achieves the lowest combined memory footprint among all models, despite its higher computational complexity. This result implies that recurrent architectures may benefit from efficient parameter reuse and reduced memory overhead during sequential processing, making them more memory-efficient for deployment on constrained edge platforms.

Regarding system memory usage, we observed relatively stable consumption across all models. Among them, the TensorRT-optimized AE model demonstrated the lowest system memory usage at approximately 2.2 GB. This suggests that while TensorRT does not significantly reduce GPU memory demands, it may enhance host-side efficiency through optimized memory allocation and reduced overhead on the CPU side. Importantly, all models remained well within the Jetson Orin Nano's 8 GB system memory capacity, confirming the feasibility of deploying these models on the device without risking memory exhaustion or requiring additional memory management strategies.

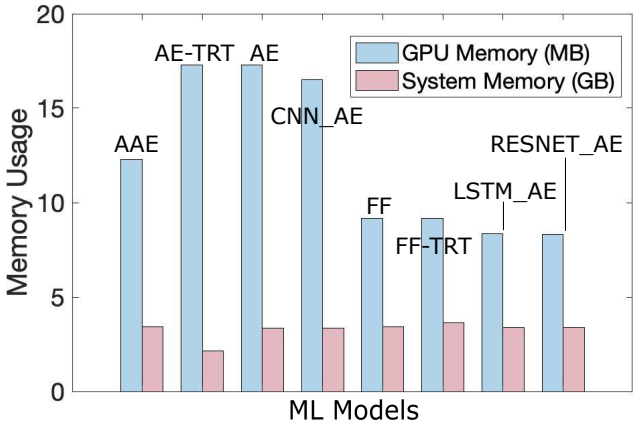

Fig. 7: Memory footprint.

F. Resource Utilization

We evaluated the resource utilization of all models with results presented in Fig. 8. The observed patterns provide valuable architectural insights. GPU utilization varied significantly across models, with the LSTM autoencoder reaching approximately 65% due to its inherently sequential processing, which limits parallelism and places greater demands on GPU resources. In contrast, most other autoencoder variants maintained GPU usage between 20–25%, reflecting better parallelization efficiency. CPU utilization remained consistently low across all models, ranging from 7–14%, with the TensorRT-optimized feedforward (FF) model exhibiting the lowest CPU usage at approximately 11%. This low CPU overhead suggests ample room for concurrent tasks such as preprocessing, data buffering, or system monitoring. Notably, the basic AE model achieved an optimal resource balance with just 7% CPU and 20% GPU utilization, making it well-suited for deployment in multi-model systems. The elevated GPU demand of the LSTM models also suggests they could benefit from quantization to improve efficiency in production environments. Overall, the moderate resource footprint across most models confirms the Jetson Orin Nano's suitability for running multiple real-time RF detection algorithms concurrently without compromising system responsiveness.

VI. Conclusion

With the rapid evolution of AI edge computing platforms and their latest capabilities, a notable gap remains in benchmarking the performance of ML models for RF anomaly detection on such platforms. In this study, we addressed this

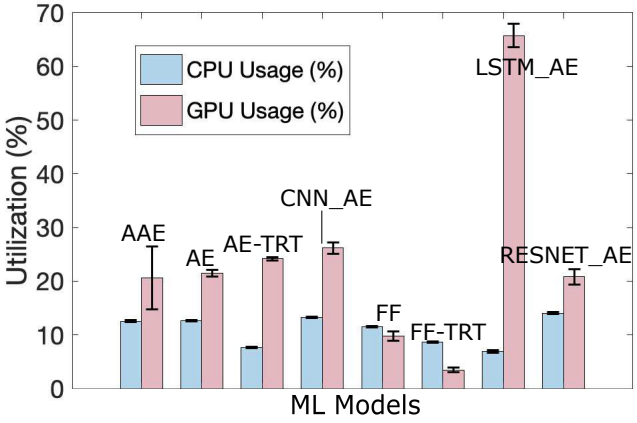

Fig. 8: Resource utilization.

gap by presenting a comprehensive benchmark analysis of various ML models deployed on the NVIDIA Jetson Orin Nano. Our results demonstrate that the device is well-equipped to support RF anomaly detection tasks, achieving strong performance across key metrics including latency, throughput, energy efficiency, and resource utilization. We expect that these findings serve as a valuable reference for both researchers and practitioners aiming to deploy ML-based RF anomaly detection solutions on edge devices.

REFERENCES

[1] B. Paul, A. R. Chiriyath, and D. W. Bliss, "Survey of rf communications and sensing convergence research," *IEEE Access*, vol. 5, pp. 252–270, 2016.

[2] H. Alipour, Y. B. Al-Nashif, P. Satam, and S. Hariri, "Wireless anomaly detection based on ieee 802.11 behavior analysis," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2158–2170, 2015.

[3] V. L. Thing, "IEEE 802.11 network anomaly detection and attack classification: A deep learning approach," in *Proc. of WCNC*, 2017.

[4] L. Cerdà-Alabern, G. Iuhasz, and G. Gemmi, "Anomaly detection for fault detection in wireless community networks using machine learning," *Computer Communications*, vol. 202, pp. 191–203, 2023.

[5] Y. Li, J. Yang, S.-L. Shih, W.-T. Shih, C.-K. Wen, and S. Jin, "Efficient iot devices localization through wi-fi csi feature fusion and anomaly detection," *IEEE Internet of Things Journal*, vol. 11, no. 24, pp. 39 306–39 322, 2024.

[6] K. Gusain, Z. Hassan, D. Couto, M. A. Malek, V. K. Shah, L. Zheng, and J. H. Reed, "Automated and blind detection of low probability of intercept RF anomaly signals," in *Proc. of Mobicom*, 2024.

[7] NVIDIA. (2022, September) Jetson orin nano sets new standard for entry-level edge ai and robotics with 80x performance leap. Accessed: 2025-08-28. [Online]. Available: https://nvidianews.nvidia.com/news/nvidia-jetson-orin-nano-sets-new-standard-for-entry-level-edge-ai-and-robotics-with-80x-performance-leap

[8] Google. (2023) Coral: Building on-device ai with edge tpu. Accessed: 2025-08-28. [Online]. Available: https://coral.ai/products/

[9] I. Corporation. (2018) Intel® neural compute stick 2. Accessed: 2025-08-28. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/neural-compute-stick/overview.html

[10] NVIDIA. (2024, December) Jetpack 6.1 boosts performance and security through camera stack optimizations and introduction of firmware tpm. Accessed: 2025-08-28. [Online]. Available: https://developer.nvidia.com/blog/nvidia-jetpack-6-1-boosts-performance-and-security-through-camera-stack-optimizations-and-introduction-of-firmware-tpm/

[11] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proc. of ICCV*, 2019.

[12] X. Chen, L. Deng, Y. Zhao, and K. Zheng, "Adversarial autoencoder for unsupervised time series anomaly detection and interpretation," in *Proc. of WSDM*, 2023.

[13] S. Liu, N. Zhou, Z. Bai, and Y. Wu, "The study on multivariate time series anomaly detection via convolutional variational autoencoder," in *Proc. of AUTEEE*, 2023.

[14] G. Zhu, L. Huang, D. Li, and L. Gong, "Anomaly detection for multivariate times series data of aero-engine based on deep lstm autoencoder," in *Proc. of ECNCT*, 2024.

[15] D. Cheng, Y. Fan, S. Fang, M. Wang, and H. Liu, "Resnet-ae for radar signal anomaly detection," *Sensors*, vol. 22, no. 16, p. 6249, 2022.

[16] L. Rey, A. M. Bernardos, A. D. Dobrzycki, D. Carramiñana, L. Bergesio, J. A. Besada, and J. R. Casar, "A performance analysis of you only look once models for deployment on constrained computational edge devices in drone applications," *arXiv preprint arXiv:2502.15737*, 2025.

[17] D. K. Alqahtani, M. A. Cheema, and A. N. Toosi, "Benchmarking deep learning models for object detection on edge computing devices," in *International Conference on Service-Oriented Computing*. Springer, 2024, pp. 142–150.

[18] F. P. Scalcon, R. Tahal, M. Ahrabi, Y. Huangfu, R. Ahmed, B. Nahid-Mobarakeh, S. Shirani, C. Vidal, and A. Emadi, "Ai-powered video monitoring: Assessing the nvidia jetson orin devices for edge computing applications," in *2024 IEEE Transportation Electrification Conference and Expo (ITEC)*. IEEE, 2024, pp. 1–6.

[19] A. Chakravarty, "Deep learning models in speech recognition: Measuring gpu energy consumption, impact of noise and model quantization for edge deployment," *arXiv preprint arXiv:2405.01004*, 2024.

[20] A. Chakraborty, W. Tavernier, A. Kourtis, M. Pickavet, A. Oikonomakis, and D. Colle, "Profiling concurrent vision inference workloads on nvidia jetson–extended," *arXiv preprint arXiv:2508.08430*, 2025.

[21] A. Duggan, T. Scully, N. Smith, and A. Giltinan, "Profiling power consumption for deep learning on resource limited devices," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2023, pp. 129–141.

[22] NVIDIA, "Getting started with jetson orin nano developer kit," https://developer.nvidia.com/embedded/learn/get-started-jetson-orin-nano-devkit, 2023, accessed: 2025-08-24.

[23] ——, "Jetson orin nano," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/, Jan. 2023, accessed: 2025-08-21.

[24] ——, "Jetpack sdk," https://developer.nvidia.com/embedded/jetpack, accessed: 2025-08-24.

[25] ——, "Jetpack 6.2 release notes," https://docs.nvidia.com/jetson/archives/jetpack-archived/jetpack-62/release-notes/index.html, accessed: 2025-08-24.

[26] ——, "Tensorrt developer guide," https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html, 2024, accessed: 2025-08-29.

[27] ——, "Supported layers and limitations in tensorrt," https://docs.nvidia.com/deeplearning/tensorrt/support-matrix/index.html, 2024, accessed: 2025-08-29.

[28] T. Prasanna Swaminathan, C. Silver, and T. Akilan, "Benchmarking deep learning models on nvidia jetson nano for real-time systems: An empirical investigation," *arXiv e-prints*, pp. arXiv–2406, 2024.

[29] Y. Zhou and K. Yang, "Exploring tensorrt to improve real-time inference for deep learning," in *Proc. of HPCC*, 2022.

[30] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. of KDD*, 2017.

[31] GNU Radio Project, "GNU Radio: The Free and Open Software Radio Ecosystem," https://www.gnuradio.org/, 2024, accessed: 2025-08-29.

[32] NVIDIA, "Jetson linux developer guide: tegrastats utility," https://docs.nvidia.com/jetson/archives/r36.2/DeveloperGuide/text/SD/Performance/tegrastats.html, 2024, accessed: 2025-08-29.

[33] J. Nielsen, *Usability engineering*. Morgan Kaufmann, 1994.

[34] NVIDIA Corporation, *Jetson Orin Series — Developer Guide*, https://docs.nvidia.com/jetson/archives/r36.2/DeveloperGuide/SO/JetsonOrinSeries.html\#power-modes-profiles, Jun. 2024, power Modes (Profiles).