Internet Technology Letters

WILEY

SPECIAL ISSUE ARTICLE OPEN ACCESS

AI-Native Networking for Resource Constrained Edge or IoT

Performance Evaluation of Privacy Models for Data Streams on the Edge

Ilir Murturi[1] | Boris Sedlak[2] | Reza Farahani[3] | Schahram Dustdar[4,5]

[1]Mechatronics, University of Prishtina, Prishtina, Kosovo | [2]Distributed Intelligence and Systems-Engineering Lab, UPF, Barcelona, Spain | [3]Institute of Information Technology, University of Klagenfurt, Klagenfurt, Austria | [4]Distributed Systems Group, TU Wien, Vienna, Austria | [5]ICREA, Barcelona, Spain

Correspondence: Boris Sedlak (boris.sedlak@upf.edu)

Received: 5 November 2025 | **Revised:** 14 January 2026 | **Accepted:** 19 February 2026

Keywords: data streams | edge computing | IoT | performance | privacy

ABSTRACT

Recent advances in edge computing enable data stream privacy enforcement directly on resource-constrained devices, reducing latency and the exposure of sensitive information. In this paper, we extend and validate our previously proposed privacy-enforcing framework, which allows high-level privacy policies to be expressed as chains of triggers and transformations, executed at the edge. To assess its practical viability, we conduct a comprehensive performance profiling of multiple privacy models across heterogeneous edge hardware platforms. Six privacy-model chains, ranging from basic face detection to combined face-and-person anonymization, are evaluated across three representative edge devices. Key performance metrics (i.e., execution time, CPU utilization, memory usage, and power consumption) are measured to inform optimal placement of privacy transformations. Our evaluation offers critical insights into the effectiveness of the privacy-enforcing framework on resource-constrained devices, thereby guiding practitioners in selecting suitable deployment targets for privacy-preserving data stream analytics on the edge.

1 | Introduction

Digital privacy has long been a core concern in distributed computing systems. Early solutions mainly focused on protecting static data in centralized and cloud-based architectures, while the rise of cyber-physical systems and smart environments has shifted privacy challenges toward continuous, real-time data streams generated by heterogeneous and resource-constrained devices. In such settings, sensitive information is often processed and transmitted across multiple system layers, making traditional cloud-centric privacy protection approaches increasingly inadequate. Furthermore, the proliferation of IoT and camera-equipped devices in smart environments (e.g., traffic monitoring, smart factories, etc.) has intensified concerns over digital privacy in continuous data streams [1]. On the other side, transferring large volumes of video and sensor data to cloud servers for further processing can result in high latency, increased bandwidth costs, and a greater risk of privacy breaches.

Edge computing has emerged as a compelling alternative by shifting computation closer to data sources [2]. By processing streams on nearby edge devices, application developers can reduce network latency and decrease their dependence on cloud services. More importantly, performing privacy-sensitive analytics on the edge enhances data protection, as raw sensitive content can be transformed or filtered before leaving the device. To address such challenges in emerging environments, we proposed an approach in our previous work to enforce privacy policies at the edge through a flexible privacy model specification [3, 4]. In the proposed framework, data streams are intercepted at an edge gateway and checked against rule-based privacy policies. Any detected privacy violations (e.g., faces, objects, screens) trigger a transformation on the data before it is forwarded. For example, filtering a security camera stream in real-time (e.g., blurring faces or screens) by a nearby edge node prevents sensitive information from ever reaching untrusted recipients. The framework enables domain experts to define high-level privacy rules in a

human-readable format, consisting of chains of conditions and actions. At runtime, the edge-based system interprets these policies and executes the corresponding trigger-detection and data transformation functions on streaming data.

An initial evaluation demonstrated the feasibility of this approach on edge devices by measuring the network latency introduced by on-the-fly transformations. However, an open question remains regarding the computational performance and scalability limits of such privacy-enforcement chains across different edge platforms. For example, where should a given privacy transformation be deployed? A lightweight IoT gateway may handle basic anonymization but struggle with complex video analytics, while a more powerful edge accelerator can process intensive privacy filters in real-time, albeit with higher energy consumption. To make informed placement decisions, it is crucial to profile the performance of various privacy-preserving models on heterogeneous edge devices under realistic workloads and resource constraints. To address these gaps, our core contributions within this paper are as follows:

- We systematically profile the computational performance of privacy-enforcement chains across three widely used edge devices with distinct computational capabilities.
- We evaluate six representative privacy-model configurations, ranging from basic face detection to combined face-and-person anonymization, within a unified privacy framework.
- For each model–device combination, we provide detailed measurements of execution time, CPU utilization, memory consumption, and power usage during operation.
- Results provide insights into the performance–energy trade-offs of deploying privacy transformations on heterogeneous edge platforms.

The rest of this paper is structured as follows: Section 2 reviews background and related work. Section 3 describes the methodology, experimental setup, privacy models, and evaluation metrics. Section 4 presents the experimental results, and Section 5 discusses the key findings, conclusions, and future directions.

2 | Background

This section provides essential background for evaluating the privacy-enforcing framework by outlining its existing architecture and operations, as described in our previous works [3, 4]. Second, we summarize related works that analyze privacy enforcement models and their operation and highlight the gap that this research paper addresses.

2.1 | Edge-Based Privacy Enforcement Framework

Our introduced framework [3, 4] enables privacy enforcement directly at the network edge, ensuring that data streams are transformed near their source to comply with predefined privacy policies. As shown in Figure 1, Edge devices, such as IoT gateways and intelligent cameras, act as *Selective Forwarding Units* (SFUs)

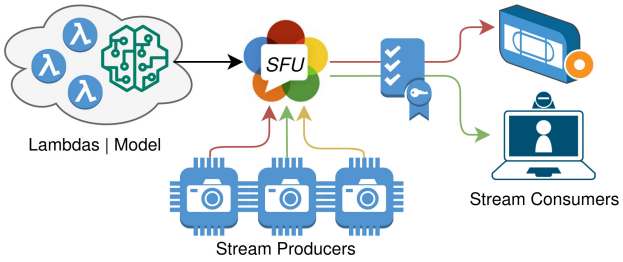

FIGURE 1 | The privacy-enforcing framework for transforming data streams on the edge; results are streamed or recorded by consumers [3, 4].

that mediate between data producers and consumers. Instead of sending raw data to the cloud, the SFU intercepts each stream, detects privacy-relevant patterns, and applies the required transformations before forwarding the processed data to its destination. This design minimizes the risk of sensitive information leakage and reduces latency compared to centralized cloud-based processing.

The data flow begins when both producers and consumers establish peer connections with the SFU through a signaling process based on the *Session Description Protocol* (SDP)[1]; this process is shown in Figure 2. Once a peer connection is established, producers transmit continuous data streams, such as video or audio frames, to the SFU, which forwards them to the consumers. The SFU incorporates a media relay component that guarantees all consumers receive synchronized data streams with consistent frame rates. To mitigate network fluctuations, a frame buffer ensures smooth video playback and consistent timing between input and output streams. This mechanism allows the SFU to maintain low-latency communication even under unstable network conditions.

At the core of the framework lies the capability to identify privacy-sensitive information within data streams and transform it before it is exposed. To achieve this, we employ *Convolutional Neural Networks* (CNNs) to detect patterns such as human faces, bodies, and other identifiable features. The framework allows for integrating a wide variety of pre-trained models converted to the *Open Neural Network eXchange* (ONNX)[2] standard, thus ensuring interoperability across different edge devices that feature the ONNX runtime. The ONNX runtime executes these models efficiently on CPUs or GPUs, providing hardware acceleration through NVIDIA CUDA when available. Figure 3 illustrates the process of converting a trained model into the ONNX format and deploying it for real-time inference on edge devices.

Each ONNX model is encapsulated within a *trigger function* that detects privacy violations by producing labels, bounding boxes, and confidence probabilities. Once a trigger activates, a corresponding *transformation function* modifies the detected region using techniques such as pixelation, Gaussian blurring, or area masking, all of which are implemented with the OpenCV library. Both triggers and transformations follow a standardized interface, enabling them to be flexibly composed into executable *privacy chains*. A privacy chain specifies how a data stream should be processed through a sequence of operations to satisfy privacy requirements. Each chain begins with the media source and

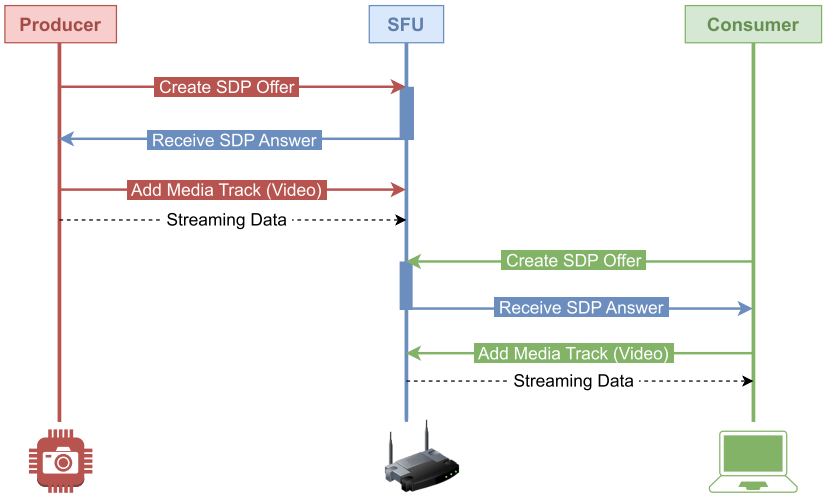

FIGURE 2 | Establishing streaming connections between the data producer, the *Selective Forwarding Unit* (SFU), and the data consumer [3].

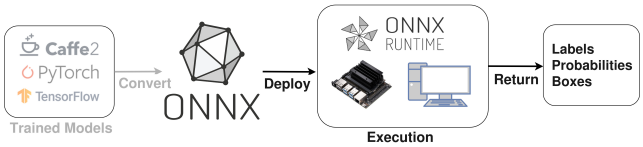

FIGURE 3 | Lifecycle of an ONNX model used for object detection: Conversion, integration, and execution on the edge device [3, 4].

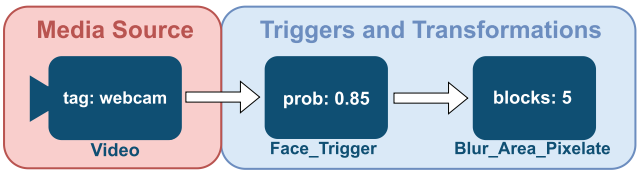

FIGURE 4 | Privacy chain representation for face blurring [3].

connects a series of triggers and transformations. For instance, a model for anonymizing human faces in a video feed can be expressed as follows:

```
video:{'tag':'webcam'} → Face_Trigger:{} →
    Blur_Area_Pixelate:{'blocks':5}
```

Such a chain describes how video frames from a webcam are first analyzed for face patterns and subsequently anonymized through pixelation. Parameters associated with each function, such as the number of pixelation `'blocks'`, allow fine-tuning the functions; for example, the anonymization strength relative to the cost of running the transformation. Figure 4 visualizes the logical flow of such a privacy chain.

Our modular architecture allows privacy rules to be dynamically defined, deployed, and updated without interrupting ongoing stream processing. Because all functions conform to a common interface, new models or transformations can be seamlessly integrated. The framework also enforces a security-first design, that is, when a stream type lacks an active privacy chain, it cannot be consumed, ensuring that no unprocessed data is transmitted unintentionally. While the framework itself is hardware-agnostic, its runtime performance strongly depends on the computational

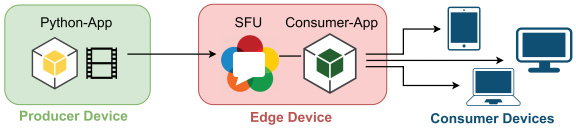

FIGURE 5 | Streaming architecture for prototype evaluation [3].

capacity of the hosting edge device. Resource-limited devices can efficiently handle lightweight models, such as face detection and pixelation, whereas computationally intensive tasks, like full-body detection or multi-object tracking, benefit from GPU-accelerated devices. Evaluating these trade-offs between model complexity, hardware capability, and energy consumption forms the core of our experimental analysis, which is presented in the following sections. Figure 5 illustrates the streaming architecture used for prototype evaluation.

2.2 | Related Work

Enforcing privacy on streaming data has been studied from various angles in the literature. Anonymization at the data source is one common strategy. For example, Wang et al. [5] proposed an anonymized data collection scheme for IoT sensors that obscures the identity of the data source. Similarly, Hajihassani et al. [6] demonstrate an edge-based method that removes or transforms sensor data attributes using representation learning, eliminating specific private attributes before the data leaves the device. However, this technique is limited to predefined attributes and cannot handle more complex or contextual privacy rules.

Another line of research focuses on privacy-preserving transformations for streaming data under formal privacy models. Khavkin et al. [7] and Jha et al. [8] explored mechanisms for ensuring differential privacy or z-anonymity in data streams with minimal latency overhead. These works illustrate approaches for injecting noise or suppressing identifiers on the fly, although maintaining privacy guarantees under concept drift remains a major challenge. Baniya et al. [9] investigated policy-driven edge architectures for privacy, focusing on the application of

role-based access filters at the edge broker level. From a performance and scalability perspective, prior systems have begun addressing how edge devices can keep up with intensive data processing. Yi et al. [10] proposed LAVEA, a latency-aware edge video analytics platform that dynamically scales the number of processing threads or nodes to handle surges in workload. Alqahtani et al. [11] analyzed the performance of object detection models on edge devices—an objective that invites more specific focus on privacy-preserving models, as presented in this work.

Contributions beyond the state-of-the-art: While many of these efforts confirm the viability of executing privacy or analytics tasks at the edge, our work provides quantitative insight into how different edge hardware platforms perform under a unified privacy enforcement framework. This complements existing research by highlighting practical deployment boundaries, for example, identifying which privacy models are too computationally demanding for a lightweight micro-computer, and conversely, whether high-end edge devices can efficiently perform advanced anonymization in real time.

3 | Methodology

We selected three edge devices representing a spectrum of performance and resource capacity commonly found in IoT deployments: *(i)* Raspberry Pi 3 Model B (RPi3), *(ii)* Raspberry Pi 4 Model B (RPi4), and *(iii)* NVIDIA Jetson Xavier NX. All devices ran under a Linux-based operating system, that is, Raspberry Pi OS on the RPi3 and RPi4, and NVIDIA JetPack (Ubuntu) on the Jetson. To isolate the computational performance of the privacy models, we assumed perfect streaming conditions from the producers and clients to the SFU.

The benchmark was designed to execute consistently across all test devices and model variants, allowing direct comparison of computational behavior under identical input conditions. It evaluates six predefined model configurations, labeled *Model A* through *Model F*. Each model represents a distinct version of the privacy-preserving processing pipeline, differing in detection method, transformation type, and computational complexity.

- *Model A: Basic Face Detection Using a Lightweight Classifier*. Performs basic face detection with a lightweight classifier, ensuring rapid processing suitable for scenarios requiring high-performance demands, such as real-time surveillance.
- *Model B: Face Detection with Gaussian Blur*. Extends *Model A* by anonymizing detected faces using Gaussian blur, providing enhanced privacy protection in video streams.
- Model C: Face Detection with Pixelation. Similar to *Model B*, it applies face detection but uses pixelation instead of a Gaussian blur, offering an alternative anonymization approach that retains a recognizable visual structure while protecting identity.
- Model D: Multi-Face Tracking with Temporal Smoothing. Incorporates multi-face tracking in addition to face detection, applying temporal smoothing techniques to improve the accuracy and consistency of face tracking across frames, as shown in Figure 6.

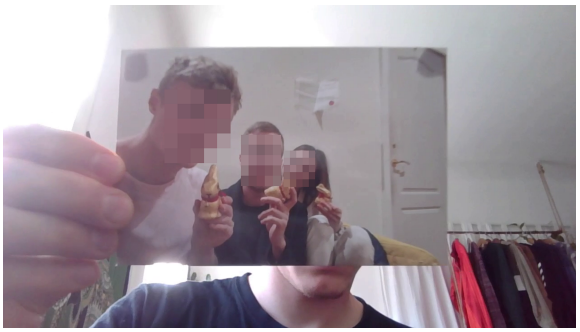

FIGURE 6 | Resulting stream with multiple faces transformed [3].

- Model E: Full-Body Detection with Anonymization. Extends the detection to full-body recognition, anonymizing detected individuals and providing stronger privacy protection in crowded environments.
- Model F: Combined Face and Person Detection with Adaptive Filters. Combines face and person detection followed by adaptive filtering, dynamically adjusting transformation strength based on real-time conditions, optimizing processing efficiency.

The same pre-recorded Full HD (1920 × 1080) video was used for all experiments. The video features dynamic human motion and varied lighting conditions, reflecting realistic usage scenarios. This setup enables each model to be evaluated under identical frame sequences and content, ensuring reliable comparative metrics. Furthermore, the benchmark captures the following performance metrics:

- *Execution Time (s)*: Total time required to process the entire video.
- *Average CPU Usage (%)*: Measured every second using the `psutil` library.
- *Average Memory Usage (MiB)*: Resident Set Size (RSS) memory consumed by the process.
- *Power Consumption (W)*: Measured with device-specific monitoring tools or sensors (when available).

For the RPi3 and RPi4, power consumption was measured using the UM25C USB Meter tool.[3] The powermeter device was placed between the power source and the RPi to log real-time voltage and current, from which the power draw in watts was calculated. On the NVIDIA Jetson Xavier NX, power consumption was measured using the built-in `tegrastats` utility[4] to monitor system power draw, along with other runtime statistics, such as CPU, GPU, and memory usage. This setup allowed software-based power estimation directly on the device.

4 | Experimental Results

To complement the theoretical exploration of privacy-enhancing video processing, a comprehensive benchmark was conducted to evaluate the performance of the implemented privacy-enforcing framework on three representative edge devices. The objective of this evaluation is to quantify the computational and resource

TABLE 1 | Summary of benchmark results by model and edge device.

Device	Model	Time (s)	Avg. CPU (%)	Avg. Mem (MiB)	Power (W)
Jetson	model_A	125.4	355.0	274.8	6.32
Jetson	model_B	104.2	349.0	283.6	6.40
Jetson	model_C	132.2	299.5	285.3	6.73
Jetson	model_D	342.7	359.7	298.1	6.77
Jetson	model_E	341.0	357.8	308.4	6.75
Jetson	model_F	103.8	349.9	307.1	5.00
RPi 3	model_A	529.5	356.7	281.4	2.50
RPi 3	model_B	536.9	353.4	285.4	2.42
RPi 3	model_C	641.9	312.4	284.1	2.70
RPi 3	model_D	1321.4	381.6	293.8	2.70
RPi 3	model_E	1316.2	382.8	303.8	2.74
RPi 3	model_F	519.4	359.5	303.0	2.23
RPi 4	model_A	153.5	327.7	285.1	4.25
RPi 4	model_B	154.4	326.6	296.4	4.25
RPi 4	model_C	171.5	302.5	296.2	4.89
RPi 4	model_D	328.4	386.5	307.3	5.22
RPi 4	model_E	332.8	383.7	318.4	5.59
RPi 4	model_F	156.7	329.0	317.4	4.35

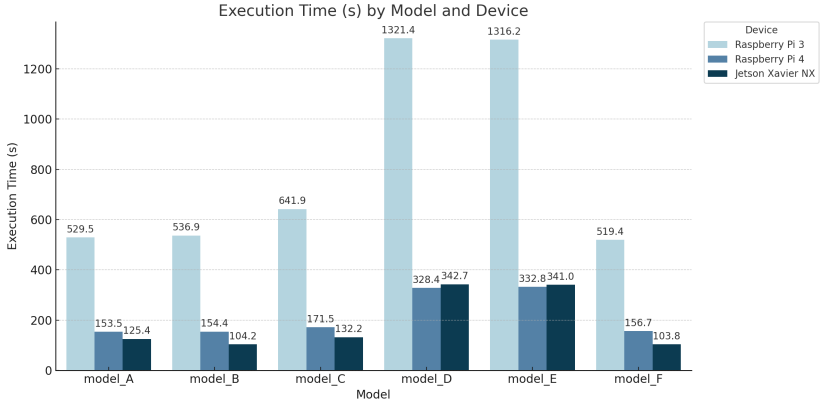

FIGURE 7 | Average execution time across devices and models.

demands of the privacy-preserving pipeline under realistic operating constraints. Specifically, the benchmark measures CPU utilization, memory consumption, power consumption, and total execution time across a set of predefined privacy models and a standard Full HD video input. These metrics provide insight into the feasibility and scalability of performing on-device video processing in resource-constrained environments. Table 1 summarizes the key performance metrics for each model and device.

4.1 | Execution Time

As shown in Figure 7, the *Jetson Xavier NX* achieved the shortest inference times, followed by the *Raspberry Pi 4*; the *Raspberry Pi 3* lagged considerably, particularly on complex models. The Jetson processed most workloads in under 350 s, while the RPi4 required 150–330 s, and the RPi3 up to 1320 s for heavier models. This demonstrates that GPU acceleration and parallelism drastically improve processing latency for advanced anonymization chains.

4.2 | CPU Usage

To pinpoint performance limits in the privacy model operation, CPU usage is a helpful indicator for operations running on the CPU. As shown in Figure 8, the *RPi3* saturated the CPU for all models (near 100% on all four cores), indicating full utilization even for simpler models. Heavier models (D, E) averaged ~382% across four cores versus ~350% for lighter ones, consistent with their higher computational load. The *RPi4* also showed high CPU engagement (300%–380% of 400%), but completed tasks much faster due to its stronger cores. Even for Models D/E, the RPi4 stayed below 400%, implying partial single-threaded behavior. In contrast, the *Jetson Xavier NX* exhibited lower CPU utilization (typically 300%–360%), as its six-core CPU offloaded substantial work to the GPU. This headroom indicates better parallel efficiency and no CPU bottleneck. In summary, both RPis operated near full CPU capacity, while the Jetson achieved comparable or superior performance.

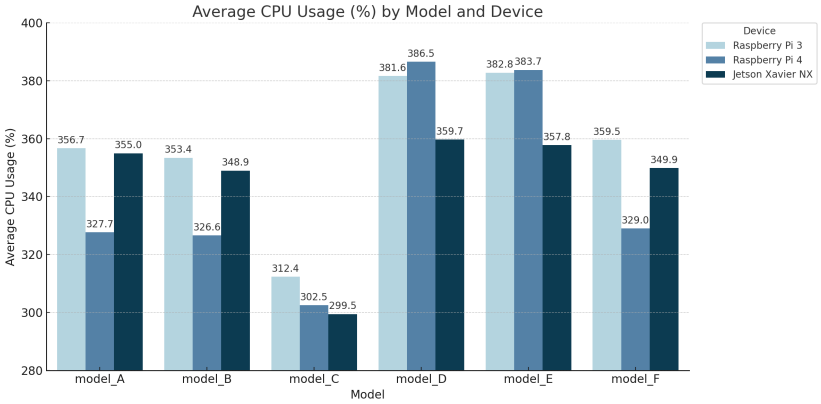

FIGURE 8 | Average CPU usage across devices and models.

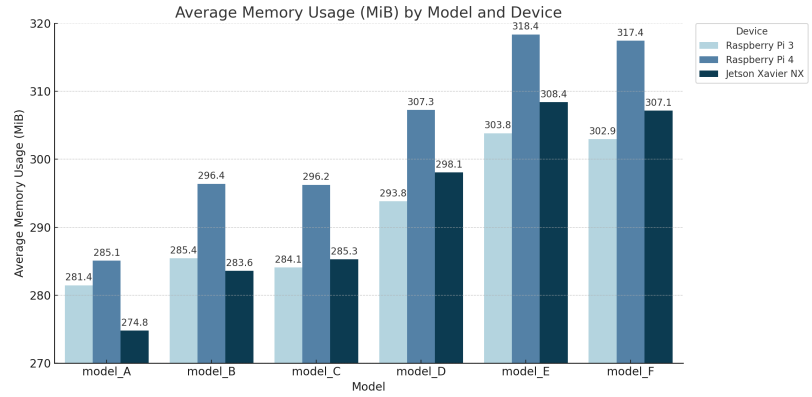

FIGURE 9 | Average memory usage across edge devices and models.

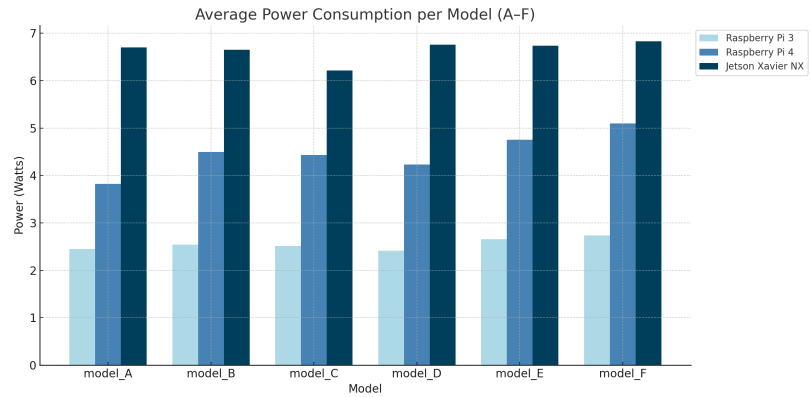

FIGURE 10 | Average power usage across devices and models.

4.3 | Memory Consumption

Memory appeared not to be a critical bottleneck in our tests, though some differences emerged across devices. As shown in Figure 9, the *RPi3*'s memory usage remained moderate even for the largest models, staying below ~304 MiB (out of 1 GB). This suggests CPU, not RAM, was the primary constraint, with no swapping observed. The *RPi4* (4 GB RAM) showed slightly higher usage for heavier models (~318 MiB for Model E), still under 8% of total memory. The *Jetson Xavier NX* had the highest absolute usage (up to ~308 MiB for Model E), but this was trivial relative to its 8 GB capacity (~4%). The higher Jetson values likely reflect GPU driver and buffer allocations. Overall, none of the devices exhibited memory pressure or thrashing, suggesting that for these video-based privacy models, memory consumption (i.e., a few hundred MiB) is not the limiting factor; instead, it is processing performance (CPU/GPU) that poses the challenge. Only multi-stream or larger model workloads might challenge the 1 GB RPi3.

4.4 | Energy Usage

Especially for battery-operated edge deployments, energy usage is a critical parameter. We observed that power draw correlates with device capability: the *Jetson Xavier NX* consumed the most power on average, followed by the *RPi4*, then the *RPi3*. As illustrated in Figure 10, the Jetson's average power ranged roughly from

TABLE 2 | Overall device suitability per model.

Model	RPi 3	RPi 4	Jetson Xavier NX
Model A	Unsuitable	Suitable	Excellent
Model B	Borderline	Suitable	Excellent
Model C	Unsuitable	Suitable	Excellent
Model D	Unsuitable	Borderline	Suitable
Model E	Unsuitable	Borderline	Suitable
Model F	Borderline	Suitable	Excellent

5 to 7 W across the models, peaking for the heavy models (D, E). The RPi4 averaged around 4–5 W, and the RPi3 was around 2.3–2.7 W. These figures are expected given the hardware differences and were relatively stable during the runs (the RPi values fluctuated by only ±0.2 W in most cases). While the Jetson NX uses more power, it performed the work much faster, which leads to an interesting perspective: *performance per watt*. The Jetson often finished a task in a fraction of the time of an RPi, so even though it draws ∼2× the power of an RPi4, it might actually use less total energy for a given job because of its speed.

4.5 | Summary

Across all metrics, the *Jetson Xavier NX* consistently outperformed the other devices, confirming its suitability for real-time, GPU-accelerated privacy enforcement. The *RPi4* offers a balanced mid-tier solution (i.e., energy-efficient yet capable of handling lightweight to moderate workloads). In contrast, the *RPi3* is only appropriate for basic or non-real-time processing. Table 2 summarizes device suitability for each privacy model based on measured performance. Overall, the benchmarking results highlight clear trade-offs between computational power, latency, and energy efficiency. GPU-accelerated devices such as the Jetson are essential for real-time privacy-preserving analytics, while mid-range platforms like the RPi4 provide an optimal cost–efficiency compromise for moderate workloads. Nevertheless, the benchmarking setup provided consistent offline measurements, network conditions and concurrent workloads were excluded. Future evaluations should extend to real-time streaming scenarios to assess end-to-end latency and throughput under operational load. Additionally, incorporating thermal monitoring and GPU utilization metrics would enable a more holistic understanding of long-term performance stability on edge devices.

5 | Conclusion

We evaluated a privacy-enforcing framework for video data streams on resource-constrained edge devices. By testing six privacy-preserving model configurations across three hardware platforms, we analyzed their efficiency, energy consumption, and scalability under realistic conditions. The results demonstrated that hardware heterogeneity strongly influences the feasibility of privacy-preserving analytics on the edge. These findings also confirmed that edge-based privacy enforcement is not only viable but also highly dependent on the computational capabilities of the underlying device. Future work will explore dynamic and adaptive deployment strategies that build upon the presented insights for designing novel workload schedulers capable of placing and migrating privacy tasks across heterogeneous edge nodes.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Endnotes

[1]https://www.rfc-editor.org/rfc/rfc4566.html.

[2]https://onnx.ai/.

[3]Ruideng UM25C USB Meter, https://www.manualslib.com/products/Ruideng-Um25c-10243666.html.

[4]Nvidia TegraStats Utility Documentation.

References

1. I. Murturi, P. K. Donta, V. C. Pujol, A. Morichetta, and S. Dustdar, "Learning-Driven Zero Trust in Distributed Computing Continuum Systems," in *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)* (IEEE; 2023, 44–49).

2. C. Avasalcai, I. Murturi, and S. Dustdar, "Edge and Fog: A Survey, Use Cases, and Future Challenges," in *Fog Computing: Theory and Practice* (Wiley, 2020), 43–65.

3. B. Sedlak, I. Murturi, P. K. Donta, and S. Dustdar, "A Privacy Enforcing Framework for Data Streams on the Edge," *IEEE Transactions on Emerging Topics in Computing* 12, no. 3 (2023): 852–863.

4. B. Sedlak, I. Murturi, and S. Dustdar, "Specification and Operation of Privacy Models for Data Streams on the Edge," in *2022 IEEE 6th International Conference on Fog and Edge Computing (ICFEC)* (IEEE; 2022, 78–82).

5. T. Wang, Y. Mei, W. Jia, X. Zheng, G. Wang, and M. Xie, "Edge-Based Differential Privacy Computing for Sensor–Cloud Systems," *Journal of Parallel and Distributed Computing* 136 (2020): 75–85.

6. O. Hajihassani, O. Ardakanian, and H. Khazaei, "Anonymizing Sensor Data on the Edge: A Representation Learning and Transformation Approach," *arXiv Preprint* (2021) ArXiv:2011.08315.

7. M. Khavkin and M. Last, "Preserving Differential Privacy and Utility of Non-Stationary Data Streams," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (2018, 29–34).

8. N. Jha, T. Favale, L. Vassio, M. Trevisan, and M. Mellia, "z-Anonymity: Zero-Delay Anonymization for Data Streams," in *2020 IEEE International Conference on Big Data (Big Data)* (2020, 3996–4005).

9. P. Baniya, G. Bajaj, J. Lee, A. Bastani, C. Francis, and M. Agumbe Suresh, "Towards Policy-Aware Edge Computing Architectures," in *2020 IEEE International Conference on Big Data (Big Data)* (2020, 3464–3469).

10. S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: Latency-Aware Video Analytics on Edge Computing Platform," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing. SEC '17* (Association for Computing Machinery, 2017), 1–13.

11. D. K. Alqahtani, M. A. Cheema, and A. N. Toosi, "Benchmarking Deep Learning Models for Object Detection on Edge Computing Devices," in *Service-Oriented Computing*, ed. W. Gaaloul, M. Sheng, Q. Yu, and S. Yangui (Springer Nature Singapore, 2025).