EdgeTraffic: An EdgeAI Dataset Integrating Road Traffic Dynamics with System Telemetry

Moysis Symeonides
msymeo03@ucy.ac.cy
University of Cyprus
Nicosia, Cyprus

Demetris Trihinas
trihinas.d@unic.ac.cy
University of Nicosia
Nicosia, Cyprus

Nicolae Cleju
ncleju@etti.tuiasi.ro
"Gheorghe Asachi" Technical University of Iasi
Iasi, Romania

Abstract

Video stream analytics has emerged as a pivotal contributor to smart traffic monitoring services, yet the systems community lacks publicly described datasets that jointly capture real-world traffic dynamics and the corresponding computational overhead. This paper introduces EdgeTraffic, a dataset derived from a live traffic monitoring service deployed at a road intersection in Iași, Romania. The setup features high-definition IP cameras streaming video over a private 5G network to a GPU-powered Multi-access Edge Computing (MEC) node hosting an AI model serving pipeline for continuous inference. Uniquely, the dataset temporally aligns model outputs (e.g., vehicle counts) with fine-grained system telemetry, including GPU/CPU utilization, memory footprint, power draw, and network activity. This alignment enables end-to-end analysis linking physical events to infrastructure behavior. To enhance utility and reproducibility, we augment the live traces with controlled replay logs of the captured footage across heterogeneous hardware targets and YOLOv8 model variants. We demonstrate the dataset's value through preliminary analyses that reveal diurnal traffic patterns, quantify systematic under-counting when adopting smaller model variants, and characterize the device-dependent correlations between inference latency, resource saturation, and energy consumption.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Distributed artificial intelligence**; • **General and reference** → **Experimentation**; • **Computer systems organization** → **Cloud computing**.

Keywords: Internet of Things, Edge Computing, Machine Learning

ACM Reference Format:
Moysis Symeonides, Demetris Trihinas, and Nicolae Cleju. 2026. EdgeTraffic: An EdgeAI Dataset Integrating Road Traffic Dynamics with System Telemetry. In *4th International Workshop on Testing Distributed Internet of Things Systems (TDIS '26), April 27–30, 2026, Edinburgh, Scotland Uk.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3802513.3803482

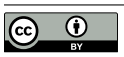
1 Introduction

The scalable deployment of IoT devices and HD video cameras in urban environments, coupled with the localized processing power of Multi-access Edge Computing (MEC), is fundamental to realizing the vision of smart cities [13]. Although modern cellular networks (5G and emerging 6G) offer high-throughput connectivity, the overwhelming volume of raw video data renders centralized cloud processing impractical due to bandwidth costs, latency constraints, and data privacy restrictions [18]. Consequently, intelligence is being pushed to the network edge, where Deep Learning (DL) models are embraced for continuous inference close to the data sources to realize EdgeAI services such as urban observability and real-time traffic management [5, 11].

For such deployments, operators run edge-hosted DL model serving pipelines (e.g., vehicle detection, car counting) and forward only the resulting analytics to the cloud for storage, visualization, and long-term analysis [12]. Although assembling and deploying such pipelines can be relatively straightforward architecturally, optimizing them is not. Operators face complex trade-offs when selecting hardware platforms (e.g., embedded vs. discrete GPUs), choosing model variants (e.g., YOLO nano vs. large), and provisioning resource capacity to meet strict latency and energy constraints [10]. Addressing these decisions requires representative data that capture both the stochastic nature of real-world inputs and the corresponding system behavior. Without aligned traces of model outputs and resource metrics, it is difficult to quantify trade-offs, validate assumptions, or perform reliable capacity planning [2].

This need for holistic data spans two distinct research communities. From a smart-city perspective, urban planners and data scientists require metrics that reflect real traffic dynamics rather than synthetic patterns to train predictive models for traffic congestion evolution, travel time estimation, and traffic signal optimization. At the same time, from an IoT and systems perspective, researchers require datasets that act as workload generators and benchmarking traces. To enable reproducible benchmarking, the community needs traces that allow for comparative studies across different MEC configurations and hardware targets under identical and realistic input conditions. However, to the best of our knowledge, no open and publicly available dataset jointly captures a real traffic monitoring workload and the corresponding end-to-end system behavior. Existing datasets fail to meet this dual need: they either focus on structural and synthetic information without runtime execution metrics (e.g., Topo4MEC [17], DATA7 [3]), or they provide telemetry tailored for security and intrusion detection rather than the performance characterization of video stream processing (e.g., TON_IoT [1], IoT-23 [7]).

To address this gap, we introduce *EdgeTraffic*, a publicly available traffic monitoring dataset that integrates time-aligned model outputs with fine-grained system metrics from a real-world deployment. The dataset is derived from an operational 5G-enabled smart city pipeline in Iași, Romania, performing continuous car detection and counting on a MEC node. We provide traces of application-level measurements (e.g., vehicle counts, inference latency) synchronized with infrastructure metrics (e.g., CPU/GPU utilization,



TDIS '26, Edinburgh, Scotland Uk

https://doi.org/10.1145/3802513.3803482

memory footprint, power draw). To increase data plurality and support system-aware analysis, we complement the live traces with logs generated by re-running captured video streams on heterogeneous GPU-enabled platforms and multiple YOLOv8 model variants. Finally, we demonstrate the dataset's utility through preliminary analyses that characterize diurnal traffic dynamics and expose how model selection and hardware platform alter the resource footprint of edge video analytics. The dataset and respective analysis code are open-sourced and made publicly available[1].

The rest of this paper is organized as follows: Section 2 discusses related work, while Section 3 presents the use-case and implementation. Moreover, Section 4 describes the dataset and Section 5 reports preliminary analyses and replay-based experiments. Finally, Section 6 concludes the paper and introduces our future directions.

2 Related Work

The systems community has long relied on cloud datacenter datasets. Notable examples include cluster traces from Google [14] and Alibaba (MLaaS GPU cluster) [16], which capture job scheduling, task duration, and resource usage in hyperscale environments. Similarly, Cortez et al. [6] characterize VM workloads on Azure Cloud over a 3 month period, creating a dataset capturing VM lifetime, size, and utilization patterns. While these datasets are foundational for cloud resource management, they are less applicable to (mobile) edge computing scenarios. Specifically, they abstract application logic, for instance, one can observe that a CPU spike occurred, but not the physical event of cause. Furthermore, the assumptions of hyperscale datacenters (e.g., infinite computing power, stable cooling, homogeneous hardware) do not hold for geo-distributed edge computing deployments.

For edge computing and IoT, researchers have developed and open-sourced dedicated datasets. A notable example is the dataset Topo4MEC [17], which focuses on network topology, featuring randomly generated MEC graphs and a realistic topology based on OpenCellID data from Milan. However, it is limited to structural data (nodes, edges, bandwidth) and lacks runtime execution metrics. A popular open dataset, DATA7 [3], combines mobile tower positions with synthetic mobility traces generated via SUMO to simulate vehicle movement in Pisa. While valuable for simulation, the workload is derived from algorithmic models rather than empirical observation. In the security domain, the TON_IoT [1] and IoT-23 [7] datasets capture telemetry and network traffic from realistic IoT testbeds (including smart home devices). While these include extensive logs, they are tailored for intrusion detection and packet analysis rather than the performance characterization of video stream processing pipelines.

In turn, the Intelligent Transportation Systems (ITS) domain offers datasets with a rich environmental context but limited system visibility. For example, the CN+ dataset [9] captures vehicle flows at a Bremen intersection (25k vehicles over 32 hours) alongside meteorological data. Similarly, the Austin, Texas camera traffic counts dataset [4] captures the number of vehicles passing through a road intersection along with their respective direction. Additionally, the ETH dataset [8] provides fine-grained traffic signal states in Zurich. Finally, UA-DETRAC [15] is a computer vision dataset that features over 10 hours of video from roadside surveillance cameras in China under challenging weather conditions. These datasets describe the input to a smart city system but do not capture any information relevant to the computational processing cost (GPU/CPU usage), inference latency, memory footprint, or energy required to analyze these flows in real-time.

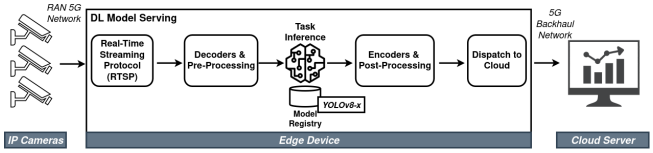


Figure 1. EdgeTraffic's EdgeAI Smart Traffic Analytic Service

The evaluation of AI-enabled geo-distributed IoT systems requires data that reflects the complexity of both the computational infrastructure and the physical environment. While researchers have access to high-quality datasets in isolation that range from datacenter traces to traffic flows, there is a notable absence of datasets that couple real-world physical inputs with the resulting impact on the underlying edge computing infrastructure.

3 Use-Case Architecture and Methodology

The EdgeTraffic dataset originates from a production-grade smart traffic monitoring service deployed at the Podu Roș intersection in Iași, Romania. Access to this infrastructure was secured through the European Union's TriasNet Horizon-JU-SNS-2022 Research and Innovation Programme, via an open call granting third-party researchers access to B5G testbeds across Europe. A high-level view of the architecture is depicted in Figure 1. The sensing layer consists of 6 HD IP cameras streaming video at 25 frames per second (FPS) via the Real-Time Streaming Protocol (RTSP) to a nearby MEC node. Connectivity is provided by a commercial private 5G network operated by Orange Romania, configured to support both Standalone (SA) and Non-Standalone (NSA) modes. To sustain the high throughput required for concurrent HD video streams, the system utilizes a dedicated Enhanced Mobile Broadband (eMBB) network slice. Camera traffic is forwarded through a Nokia FastMile 5G Customer Premises Equipment (CPE) unit, which handles traffic differentiation via multiple VLANs on a single SIM with distinct Data Network Name (DNN) profiles.

The software pipeline is developed by adopting Savant AI[2], which provides Python abstractions on top of the Nvidia DeepStream SDK[3] to implement GPU-accelerated computer vision and video analytics services. Incoming video streams are decoded via hardware acceleration (NVDEC) and converted to raw frames for analysis. Each frame is forwarded to a YOLOv8 object detector[4] (input resolution 640×640 px), executed through a TensorRT-optimized ONNX runtime. Post-inference processing extracts object bounding boxes and computes per-frame and aggregate counts (e.g., car density), with internal components communicating via an asynchronous pub-sub messaging model. These metrics serve as the critical input for traffic operators to monitor real-time mobility patterns, aiming to parameterize congestion models and optimize traffic signal timing. Inference outputs are serialized and exported to a Kafka queue for downstream consumption. During the live field trials, operational privacy constraints prevented the storage

[1] https://github.com/unic-ailab/EdgeTraffic

[2] https://savant-ai.io/

[3] https://developer.nvidia.com/deepstream-sdk

[4] https://yolov8.com/

Field	Short description
fseqno	Frame sequence number for the video stream
timestamp	Unix timestamp associated with the inference execution
car_count	Number of cars detected by the vision model for each frame
inf_latency	Per-frame inference latency measured at the edge pipeline
gpu_util	GPU compute utilization during pipeline execution
gpu_power	GPU power draw during inference and processing
network	Outgoing network traffic size from edge to cloud (in bytes)
gpu_mem	GPU memory usage while serving the model
cpu_power	CPU power draw during pipeline execution
cpu_util	Host CPU utilization during pipeline execution
cpu_mem	Host memory usage during pipeline execution

Table 1. EdgeTraffic Dataset fields

of raw video content for making data publicly available. Consequently, the primary dataset comprises of derived analytic insights and time-aligned system logs rather than archived footage.

To characterize the resource demands of the DL model serving pipeline, we instrumented the edge infrastructure to synchronize application-level performance with low-level system telemetry. The field trials were conducted over a 5-day period in late January 2025, ensuring the capture of traffic analytics across diverse diurnal windows and model configurations. For precise latency characterization, we utilized Savant's metadata processors to inject timestamps immediately preceding and following the computer vision model execution. With this, the delta between these markers yields the specific per-frame inference latency. In parallel, we deployed NetData[5] and Prometheus[6] to log CPU, RAM, and Network I/O, while a wrapper around the nvidia-smi interface[7] polls GPU-specific metrics, including compute utilization, memory footprint, and instantaneous power draw.

The live deployment of the field tests ran on a carrier-grade MEC server equipped with an Intel Xeon Silver 4210 CPU (2.20 GHz), 32 GB RAM, and an Nvidia V100 GPU (32 GB VRAM). Beyond the live traces, we introduce a "replay" video stream to enable controlled comparisons across heterogeneous hardware. For this, we recorded a 1-hour reference video stream and replayed it offline through the same software pipeline on two additional devices: an edge server with an Intel Xeon Gold 6230 and Nvidia T4 GPU, and an embedded Nvidia Jetson AGX Orin (12-core Arm Cortex, Ampere GPU, 64 GB RAM). For the Orin, we modified the monitoring exporter to parse telemetry from the tegra-stats utility as it lacks nvidia-smi support. Across these replay experiments, we also varied the model complexity by deploying, other than the YOLOv8-X variant, the YOLOv8-N and YOLOv8-M variants, to allow researchers to quantify the trade-offs between detection accuracy, inference latency, and energy consumption.

4 Dataset Structure and Context

The EdgeTraffic dataset is organized as a collection of directories, where each corresponds to a distinct experimental trial executed on a specific hardware platform and DL model variant. Each trial directory contains two primary files: a CSV file (`data.csv`) containing the high-frequency time-series measurements, and a YAML file (`metadata.yaml`) documenting the execution context.

[5] https://www.netdata.cloud/
[6] https://prometheus.io/
[7] https://docs.nvidia.com/deploy/nvidia-smi/index.html

```yaml
trial_id: "v100_yolov8x_live_001"
video_type: "live"
duration: 120
date: "28-01-2025"
device:
  name: "Server"
  gpu_model: "NVIDIA V100"
model:
  family: "YOLOv8"
  variant: "X"
  input_size: [640, 640]
pipeline:
  framework: "Savant"
  sdk: "NVIDIA DeepStream"
  fps: 25
time:
  start: "1738041386"
  end: "1738048586"
output:
  csv_file: "data.csv"
  fields:
    - "fseqno"
    - "timestamp"
    - "car_count"
    - "inf_latency"
    - "gpu_util"
    - "..."
```

Figure 2. YAML Metadata File

Table 1 summarizes the schema of the `data.csv` logs. The frame sequence number (`fseqno`) serves as the unique index for the video stream. The `timestamp` field records the Unix epoch time (in seconds) for each inference event. We note that, since the pipeline processes video at 25 fps, approximately 25 distinct data points share the same second-granularity timestamp. Application-level performance is captured by `car_count` (the number of objects detected per frame) and `inf_latency`. The latter reports the pure model inference time (in seconds), omitting overheads such as batching, and message serialization, as these were negligible and stable across runs. To characterize system resource behavior, the dataset includes two categories of telemetry. GPU metrics include utilization (`gpu_util`, 0–100%), power drawn (`gpu_power`, in Watts), and memory footprint (`gpu_mem`, in GB). Host metrics capture the cumulative CPU utilization (`cpu_util`, 0–100%), CPU power draw (`cpu_power`, in Watts), system memory usage (`cpu_mem`, in Bytes), and network activity (representing outgoing traffic from edge-to-cloud in Bytes).

To facilitate reproducibility, we provide a `metadata.yaml` file for every trial run with Figure 2 depicting an example. This file records the unique `trial_id` and the `video_type` (e.g., `live` or `replay`), alongside the collection `date` and trial `duration`. It formally documents the hardware specifications (e.g., Server with Nvidia V100), the software stack (Savant over DeepStream), and the model configuration (e.g., YOLOv8-X, 640x640px resolution). By adhering to this generic schema, the dataset remains extensible, allowing future contributions of new devices or workloads that can be integrated seamlessly.

We note that metric availability varies by hardware target due to driver limitations. The V100-enabled MEC server provides the full suite of metrics detailed in Table 1. However, the T4 edge

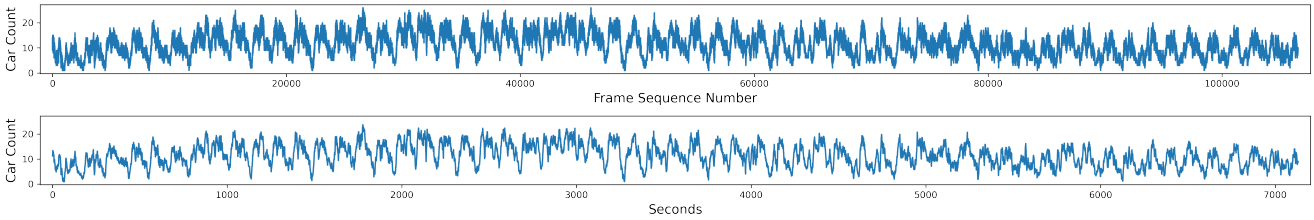


Figure 3. Car count over a two-hour period shown per frame in the upper plot and averaged per second in the lower plot.

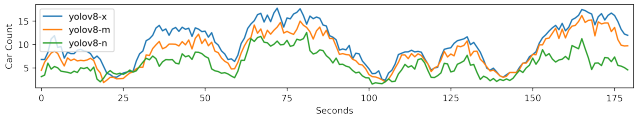


Figure 4. Timeline results of different YOLOv8 variations.

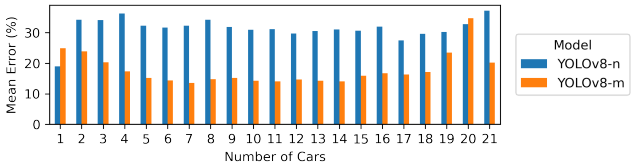


Figure 5. Errors for car count for YOLOv8-M and YOLOv8-N.

server and Jetson AGX Orin provide a subset: `fseqno`, `timestamp`, `car_count`, `inf_latency`, `gpu_util`, and `gpu_power`. In turn, trial durations vary based on the deployment environment. For the MEC server, we provide approx 120min traces for the production model (YOLOv8-X) captured during live deployment. Additional trials for smaller YOLOv8 variants on the live MEC server are limited to 15min windows to minimize interference with the production service. Nonetheless, for the replay experiments, we provide complete 60min traces embracing the YOLOv8-M and YOLOv8-N variants run on the T4-enabled server and the Jetson AGX Orin box.

5 Dataset Utility and Preliminary Evaluation

In this section, we demonstrate the utility of the EdgeTraffic dataset through preliminary analysis and controlled experimentation to provide system-level characterization of accuracy, latency, and energy trade-offs. The following sub-sections detail these findings across diverse hardware and model configurations.

5.1 Characterizing Car Count Dynamics

Figure 3 illustrates the variability of traffic intensity captured during the live deployment, utilizing the `car_count` metric extracted from the YOLOv8-X detector. The upper plot presents the raw per-frame car count across a full 2-hour capture, indexed by frame sequence number (`fseqno`). This granular view highlights rapid fluctuations driven by immediate scene changes, such as stochastic vehicle arrivals, departures, and transient occlusions. To mitigate high-frequency noise, the lower plot aggregates these measurements via a 1-second sliding window average. This smoothing reveals a clearer trajectory while preserving the underlying short-term dynamics. With this, the metric exhibits a distinct periodic structure, with peaks and troughs recurring at a consistent cadence. This behavior aligns with the operational logic of the intersection, reflecting traffic signal cycles that release vehicle platoons during green intervals, followed by phases of relative inactivity. While this periodicity is obscured at the per-frame resolution by fine-grained volatility, it becomes pronounced after temporal aggregation. These observations confirm that the workload is characterized by burstiness at short time scales yet remains structured and partially predictable over longer intervals. The aforementioned are interesting insights for researchers attempting to correlate visual scene activity with system resource demands.

5.2 Comparative Analysis of Model Variants

Next, we quantify the degradation in detection accuracy across YOLOv8 model variants. For context, these models span a wide range of computational complexity. Specifically, the lightweight Nano variant (YOLOv8-N) contains approx. 3.2M parameters, the Medium (YOLOv8-M) scales to 25.9M, and the Extra-Large (YOLOv8-X) comprises 68.2M parameters. Using a 1-hour replay video trace, we executed an identical input stream through the DL model serving pipeline on the T4-equipped edge server while varying only these detection models. Figure 4 reports the temporal evolution of the `car_count` emitted by each model, aggregated into 1-minute averages to attenuate frame-level noise and highlight persistent trends. The blue curve represents the production-based baseline YOLOv8-X, the orange curve corresponds to YOLOv8-M, and the green curve to YOLOv8-N. Across the full interval, all three models reconstruct the same macroscopic temporal patterns, indicating consistent reactivity to changes in scene activity. However, a clear systematic offset in magnitude is evident. Specifically, the YOLOv8-X consistently reports higher counts than the M variant, and substantially higher counts than the N variant. This suggests that the smaller architectures miss a significant fraction of vehicles that is likely due to reduced representational capacity when handling small objects or challenging lighting, with these resulting in frequent false negatives. While some error is expected relative to the production-based model, the persistent under-counting by the M and especially N variant, highlight the accuracy penalty associated with lightweight models.

This motivates an explicit trade-off analysis between the model variants. To this end, Figure 5 summarizes the Mean Absolute Percentage Error (MAPE) of the lighter variants relative to YOLOv8-X. Methodologically, for each reference `car_count` (bucket) on the x-axis as determined by YOLOv8-X, we compute the average percentage deviation exhibited by the M and N variants. Two distinct behaviors emerge from this analysis. YOLOv8-N exhibits consistently higher error across almost all traffic densities, remaining strictly above 30% except for the very first bucket (where it drops below 20%). This flat error profile confirms that the under-counting is systemic rather than situational. In contrast, YOLOv8-M is generally more accurate, maintaining a stable error below 20% for typical load conditions. However, its performance degrades noticeably at the edges of the distribution. Specifically, the error rises both when the scene is sparse and when it approaches peak congestion (approx.

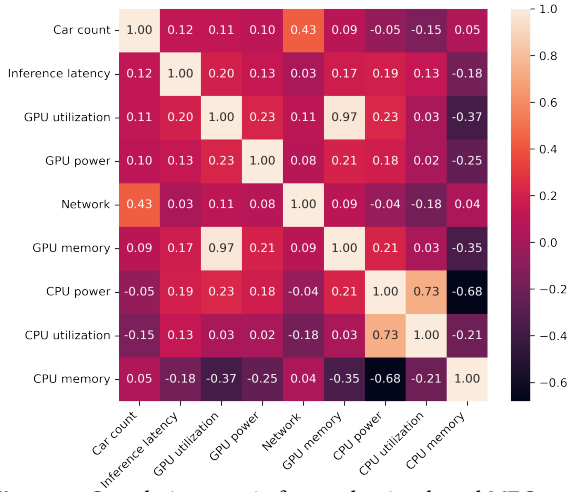


Figure 6. Correlation matrix for production-based MEC node

19–21 detected cars). This indicates that, for this specific dataset, the M variant struggles to maintain parity with the X model during dense traffic phases.

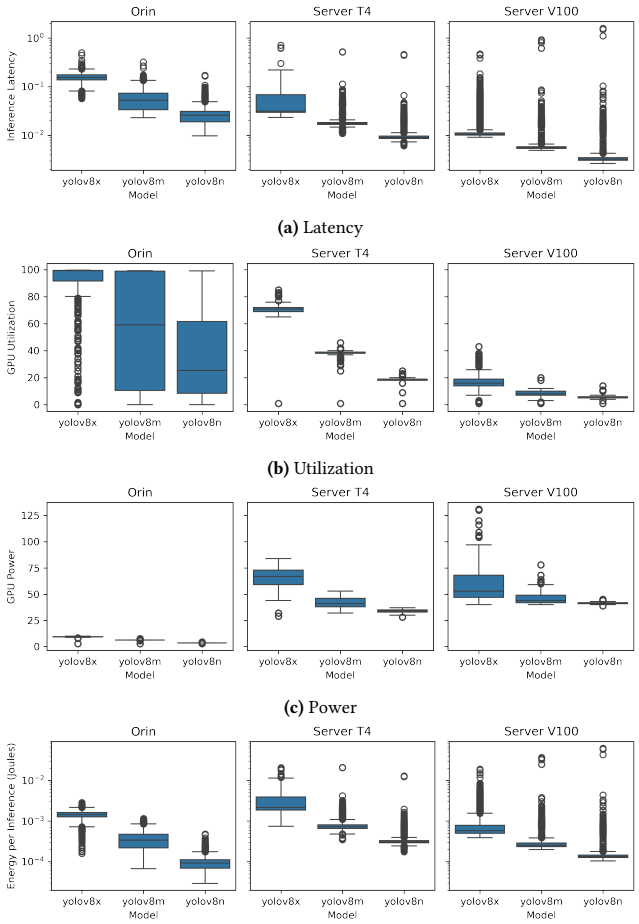
5.3 Impact of Model Complexity on Resource Utilization

To understand the coupling between application and infrastructure dynamics, we analyze the pairwise Pearson correlation coefficients among app metrics and system telemetry for the live production-based deployment (MEC node with Nvidia V100 GPU), as shown in Figure 6. To ensure precise temporal alignment between heterogeneous data sources, we aggregate all feature values into 1-second windows before performing the analysis as inference logs occur per-frame while system metrics are sampled per-second.

Several distinct performance signatures emerge from the correlation matrix. First, GPU utilization is effectively synchronized with GPU memory footprint (r=0.97). This indicates that for the YOLOv8-X workload, periods of high compute intensity impose a proportionally high demand on GPU VRAM, likely due to the dynamic allocation of tensor workspaces during complex frame processing. Second, host metrics adhere to expected physical constraints, where CPU power exhibits a strong positive correlation (r=0.73) with CPU utilization. In turn, inference latency shows a weak correlation with CPU utilization. This lack of coupling confirms that the critical path for performance is effectively offloaded to the GPU acceleration and host-side processing (decoding, message passing) does not bottleneck the pipeline. Consequently, latency behavior is driven primarily by GPU dynamics rather than host resource contention. Fourth, the Car Count metric shows a positive correlation (r=0.43) with network egress traffic. This validates the pipeline's architectural logic where detecting more objects generates a larger volume of metadata (bounding boxes and classification tags), directly increasing the size of the serialized messages transmitted to the cloud dashboard. Finally, we observe a counter-intuitive negative correlation between host memory usage and CPU power (r=-0.68). This implies that the workload is compute-bound rather than memory-bound, where phases of intense CPU processing by Savant do not coincide with higher host memory pressure.

5.4 Device-Dependent Performance Profiles

Figure 7 presents a cross-device performance characterization of the video analytics pipeline when adopting the 3 different YOLOv8



(a) Latency

(b) Utilization

(c) Power

(d) Energy

Figure 7. System-level metrics for different deployments

model variants. Each panel depicts a set of boxplots, where the line inside each box is the median, the box spans the interquartile range, and the whiskers capture the spread of the remaining samples. The first plot in each panel corresponds to the Jetson AGX Orin box, the second to the edge server powered by an Nvidia T4 GPU, and the third, the production-based MEC server with an Nvidia V100 GPU. We note that to improve plot visibility due to many outliers, we opt for a logarithmic scale for the y-axis for the per-frame latency and the per-frame energy metrics.

Figure 7 (a) reports inference latency of all devices. Specifically, the latency decreases monotonically from YOLOv8-X to M and then to N, reflecting the reduced computational cost of the smaller models. The absolute latency is highest on Orin and lowest on V100, while T4 sits in between. This highlights a clear hardware effect, where stronger accelerators reduce not only the median latency but also its variability. Moreover, Figure 7 (b) highlights GPU utilization during pipeline execution. Orin shows the highest GPU pressure, particularly for YOLOv8-X, and the widest spread for the smaller models. This is consistent with a pipeline where GPU side stages such as decoding, batching, and pre- and post-processing become more visible when the GPU work per frame decreases. On the two servers, GPU utilization is substantially lower and becomes progressively smaller from X to N, indicating that the pipeline

becomes less GPU intensive as the model footprint shrinks and the overall per frame processing path shortens.

For the GPU power draw (Figure 7 (c)), the server GPUs operate at much higher absolute power than Orin, and within each server the power draw increases with model size. This aligns with expectations since larger models drive higher sustained GPU activity. Orin exhibits much lower GPU power across all variants, reflecting the different power envelope of an embedded accelerator.

Lastly, Figure 7 (d) shows energy per inference, which captures the combined effect of power and latency. For every device, moving from YOLOv8-X to M and N sharply reduces energy per inference, showing that model downsizing yields substantial energy savings in addition to latency gains. A key takeaway is that higher power hardware does not necessarily imply higher energy per inference. The V100 MEC server often achieves low energy per inference because its substantially shorter latency compensates for its higher instantaneous power. Overall, the figure shows the expected accuracy independent trade-off space at the system-level, where model size and hardware jointly determine latency, resource pressure, and energy cost per frame.

6 Conclusions & Future Work

In this paper, we presented EdgeTraffic, a real-world dataset that integrates video analytics inference outputs with fine-grained system telemetry from an operational smart traffic management service deployed on a MEC server. The dataset aligns application metrics, such as vehicle counts and inference latency, with system-level logs that include CPU and GPU utilization, memory footprint, power consumption, and network I/O. To facilitate reproducibility and comparative analysis, we complemented the live traces with controlled replay experiments across heterogeneous hardware targets and YOLOv8 model variants. Our preliminary analysis demonstrates that real-world traffic flows are characterized by short-term burstiness overlaid on periodic structures driven by intersection signaling. Furthermore, our cross-device benchmarks reveal critical system trade-offs: we quantified the systematic under-counting inherent to lightweight models and highlighted that high-power server-grade accelerators can yield lower energy-per-inference due to significantly reduced latency. Ultimately, EdgeTraffic provides researchers and urban planners with the empirical ground truth necessary for benchmarking adaptive middleware, modeling workload capacity, and designing energy-efficient edge systems.

Our future work targets a three-fold extension of the dataset. First, we aim to broaden the environmental diversity by incorporating longitudinal data covering distinct seasonal variations, lighting conditions, and weather events. Second, we intend to enrich the telemetry fidelity to support deeper root cause analysis of performance anomalies across heterogeneous hardware. Third, we will leverage EdgeTraffic as a testbed for evaluating adaptive orchestration and selective inference policies, focusing on minimizing energy and latency without compromising the accuracy of downstream traffic analytics.

Acknowledgment. This work is part of AdaptoFlow, receiving funding from the European Union's Horizon Europe research and innovation action programme, via the TRIALSNET Open Call issued and executed under the TrialsNet project (Grant Agreement no. 101017141). Certain language refinements were performed at the sentence level using Gemini and ChatGPT. All original content and ideas are solely those of the authors.

References

[1] Abdullah Alsaedi, Nour Moustafa, Zahir Tari, Abdun Mahmood, and Adnan Anwar. 2020. TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access* 8 (2020), 165130–165150. doi:10.1109/ACCESS.2020.3022862

[2] Mohammad S Aslanpour, Sukhpal Singh Gill, and Adel N Toosi. 2020. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things* 12 (2020).

[3] Emanuele Carlini, Massimo Coppola, Patrizio Dazzi, Luca Ferrucci, Hanna Kavalionak, and Matteo Mordacchini. 2023. DATA7: A Dataset for Assessing Resource and Application Management Solutions at the Edge. In *Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge* (Orlando, FL, USA) *(FRAME '23)*. Association for Computing Machinery, New York, NY, USA, 3–6. doi:10.1145/3589010.3595652

[4] City of Austin. 2025. Austin Texas Camera Traffic Counts. https://data.austintexas.gov/Transportation-and-Mobility/Camera-Traffic-Counts/sh59-i6y9/about_data.

[5] Nicolae Cleju, Carlos Pascal, Ciprian-Romeo Comsa, Constantin-Florin Caruntu, Iulian B. Ciocoiu, Cristian Patachia-Sultanoiu, and Razvan Mihai. 2024. Towards Efficient Urban Mobility: Deployment Strategies for Smart Traffic Management and Crowd Monitoring Systems. In *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, Antwerp, Belgium, 997–1002. doi:10.1109/EuCNC/6GSummit60053.2024.10597063

[6] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 153–167. doi:10.1145/3132747.3132772

[7] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. 2020. *IoT-23: A labeled dataset with malicious and benign IoT network traffic*. Zenodo. doi:10.5281/zenodo.4743746

[8] Alexander Genser, Michail A. Makridis, Kaidi Yang, Lukas Abmühl, Monica Menendez, and Anastasios Kouvelas. 2023. A traffic signal and loop detector dataset of an urban intersection regulated by a fully actuated signal control system. *Data in Brief* 48 (2023), 109117. doi:10.1016/j.dib.2023.109117

[9] Thenuka Karunathilake, Meyo Zongo, Dinithi Amarawardana, and Anna Förster. 2024. CN+: vehicular dataset at traffic light regulated intersection in Bremen, Germany. *Scientific data* 11, 1 (2024), 665.

[10] Michalis Kasioulis, Moysis Symeonides, Giorgos Ioannou, George Pallis, and Marios D. Dikaiakos. 2024. Energy modeling of inference workloads with AI accelerators at the Edge: A benchmarking study. In *2024 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, Paphos, Cyprus, 189–196. doi:10.1109/IC2E61754.2024.00028

[11] Demetris Trihinas, Panagiotis Michael, and Moysis Symeonides. 2024. Towards low-cost and energy-aware inference for edgeai services via model swapping. In *2024 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 168–177.

[12] Demetris Trihinas, Moysis Symeonides, Nicolae Cleju, George Pallis, and Marios Dikaiakos. 2026. FakeInf: Selective Deep Neural Network Inference for Latency and Energy-Aware Model Serving Pipelines. In *Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing (UCC '25)*. Association for Computing Machinery, New York, NY, USA, Article 15, 10 pages. doi:10.1145/3773274.3774270

[13] Amin Ullah, Syed Myhammad Anwar, Jianqiang Li, Lubna Nadeem, Tariq Mahmood, Amjad Rehman, and Tanzila Saba. 2024. Smart cities: The role of Internet of Things and machine learning in realizing a data-centric smart environment. *Complex & Intelligent Systems* 10, 1 (2024), 1607–1637.

[14] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems* (Bordeaux, France) *(EuroSys '15)*. Association for Computing Machinery, New York, NY, USA, Article 18, 17 pages. doi:10.1145/2741948.2741964

[15] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. 2020. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding* 193 (2020), 102907. doi:10.1016/j.cviu.2020.102907

[16] Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. 2022. MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 945–960.

[17] Bin Xiang, Jocelyne Elias, Fabio Martignon, and Elisabetta Di Nitto. 2021. A dataset for mobile edge computing network topologies. *Data in Brief* 39 (2021), 107557. doi:10.1016/j.dib.2021.107557

[18] Minrui Xu, Hongyang Du, Dusit Niyato, Jiawen Kang, Zehui Xiong, Shiwen Mao, Zhu Han, Abbas Jamalipour, Dong In Kim, Xuemin Shen, et al. 2024. Unleashing the power of edge-cloud generative AI in mobile networks: A survey of AIGC services. *IEEE Communications Surveys & Tutorials* 26, 2 (2024), 1127–1170.