Design and Deployment of an Edge-Aware MLOps System for Multimodal Sensor Data

Lukasz Karbowiak
Faculty of Computer Science and Artificial Intelligence, Czestochowa University of Technology
Czestochowa, Poland *lkarbowiak@icis.pcz.pl*

Jacek Piatkowski
Faculty of Computer Science and Artificial Intelligence, Czestochowa University of Technology
Czestochowa, Poland *jacek.piatkowski@icis.pcz.pl*

Lukasz Kuczynski
Faculty of Computer Science and Artificial Intelligence, Czestochowa University of Technology
Czestochowa , Poland *lukasz.kuczynski@icis.pcz.pl*

Abstract

The article presents the design and deployment of a production-grade MLOps infrastructure that integrates edge computing with cloud-based resources for efficient machine learning operations. The system connects vehicle-mounted sensors (cameras and LiDAR) with a centralized data lake and a Kubernetes-based cloud environment. A fully automated pipeline, developed using Apache NiFi, manages continuous data acquisition, preprocessing, metadata registration, and model lifecycle orchestration. Data collected on edge devices are stored on a shared NAS and seamlessly transferred to the cloud for large-scale training and evaluation. The system allows for safe access, flexibility, and processing based on metadata, making it easy to scale and repeat machine learning workflows that meet current MLOps standards. The system has been validated in real-world deployments, confirming its applicability to safety-critical scenarios.
Keywords: MLOps, Edge Computing, Automated Machine Learning Pipeline

1. Introduction

With the increasing availability of data and computing power, there is growing pressure to seek solutions that enhance the processes of data management, machine learning, and software development. A broad discussion on this topic can be found, among others, in the works [1], [3], [5], [8]. The authors of the work [5] emphasize at the same time that a large part of, perhaps not always well-organized, machine learning projects do not end in success and move on to the production phase. In this context, the MLOps (Machine Learning Operations) paradigm is currently indicated as the most promising. The most important principles of this model are pointed out as: (i) automation of the continuous integration and continuous delivery (CI/CD) process, i.e. automation of the process of delivering new code to the development repository and running tests; (ii) workflow orchestration, (iii) reproducibility including versioning of data, model, and code; (iv)collaboration; (v) continuous Machine Learning (ML) training and evaluation; (vi) ML metadata tracking and logging; (vii) continuous monitoring; and (viii) use of feedback loops.

This article presents the practical implementation of an MLOps-compliant architecture for a hazardous situation detection system at pedestrian crossings using data collected from sensors mounted on our research vehicle. These data include images and depth maps. The whole system

guarantees low-latency detection and scalable model training and data storage by combining edge devices with cloud storage and processing.
In designing one of its main components, the Synchronized Data Acquisition System (SDAS) described in [7], we used an approach consistent with the rules of ontology-driven conceptual modeling (ODCM) [2]. According to the principles of this modeling method, we formulated the functional assumptions of the SDAS as follows: (i) the recorded information can come from any sensors that can provide data in any formats in any cycles (e.g. video frames, LIDAR point cloud data, GPS coordinates, etc.); (ii) the length of the data recording session can be any - i.e. one set of data can be a snapshot of an event, while the next set can cover several or several dozen minutes of observation of any phenomenon or process.
SDAS was written by us in C++, which provides powerful mechanisms supporting the ODCM philosophy, i.e. tools designed for generic programming – see Tab. 1.

Ontology-driven conceptual modeling	**C++ generic programming**
Operating on entities and their properties in an instance-independent manner.	Operating on type parameters (generic types) without specifying them.
Definition of classes of entities, their properties and relationships.	Definition of parameterized function templates, class templates and generalized algorithms.
Multiple uses of conceptual models.	Multiple use of templates (models).
Ontological axioms and relationship constraints.	Constraints on syntactic and semantic requirements.
Conceptual abstraction, hierarchy and relationships, extensibility, modularity and reuse, application to complex systems.	

Table 1. The main features of ODCM and capabilities of C++ generic programming tools.

2. System Architecture in the MLOps Context

The implemented architecture can be divided into three core components, described below.

Edge Server. All devices communicate through a local network created within the vehicle, ensuring synchronized data acquisition and efficient coordination. Instead of storing data locally on each edge device, the system is configured to store all sensor output data in a centralized network-attached storage (NAS) unit mounted inside the vehicle. This approach ensures unified access to the data and simplifies downstream processing and transmission.
The edge server deploys Apache NiFi, which continuously monitors the NAS directory structure. It automates the retrieval of new files, performs basic data filtering and transformation tasks, and initiates the transmission of data and related metadata to the cloud infrastructure. The project supports both online mode and store-and-forward mode, enabling reliable operation even with intermittent connectivity.
In addition, inference tasks (e.g., danger situation detection) can be performed locally, enabling low-latency decision making, which is critical for real-time applications such as pedestrian behavior monitoring or obstacle detection. The edge layer thus acts as the first stage in the MLOps pipeline, providing both computational capabilities and a reliable mechanism for data transmission to the cloud.

Data Lake. The object storage layer-compliant with the Amazon S3 API—serves as the long-term repository for all sensor data, training artifacts, model binaries, and result files. It ensures durability, scalability, and support for large, unstructured datasets such as images, point clouds, and video streams.
MongoDB functions as a dynamic metadata registry. Each file saved in S3 comes with a struc-

tured metadata document, which includes important details like when it was acquired, the type of sensor used, the spatial resolution, the location, and any notes about the objects. This rich metadata layer facilitates advanced queries and filtering during data discovery, allowing training data sets to be programmatically assembled based on complex criteria.
To support real-time workflows, metadata entries are automatically created or updated by custom Apache NiFi processors (e.g., KMDPutMongo, KMDPutMetadataToMongo). These ensure that every data asset is traceable and reusable, reinforcing reproducibility and regulatory compliance in machine learning workflows.

Cloud Infrastructure. The Kubernetes-based cloud environment provides a scalable and resilient foundation for running workloads intensive compute-intensive machine learning. It enables elastic resource allocation, fault-tolerant execution of long-running jobs, and parallel processing of independent tasks across multiple nodes.
Apache NiFi, deployed as a managed application within this cluster, orchestrates the entire data science workflow: from identifying relevant datasets in the metadata registry to packaging them for training, triggering model execution jobs, and finally storing the output back in the S3 compatible data lake. NiFi's visual flow-based programming interface simplifies the coordination of complex multi-stage pipelines while maintaining traceability and modularity.
Model training and inference jobs are submitted as Kubernetes jobs, often with GPU requirements. These jobs are dynamically scheduled and executed in containerized environments, ensuring consistency and isolation. The system currently supports training architectures like YOLOv7-YOLOv12, PointCNN, PointRCNN and Frustrum PointNet with extensibility for custom models through script-based execution.
Access to cloud resources is governed by Keycloak, which provides a centralized identity and access management system. Integrated with NiFi via OpenID Connect (OIDC), it ensures secure authentication, role-based authorization, and user auditability. This setup supports multi-tenant usage scenarios, where different research teams or applications can securely share infrastructure while maintaining data isolation.

3. MLOps Workflow in the Use Case

The system we developed (see Fig. 1) supports a fully automated MLOps pipeline covering data acquisition, pre-processing, training, deployment and monitoring.

The use of data from many different types of sensors makes it possible to better map the observed phenomena, but it becomes crucial here to ensure proper synchronization of the acquired data. The problem is to ensure that different types of data, collected at different intervals and stored in different data sets, always represent the same events.

We have adopted a data fusion strategy at the raw data level. Having the raw data at our disposal, we gained the ability to analyze and process it later using different techniques and algorithms. We made an a priori assumption that none of the measuring devices would provide data (samples) exactly at the moments when (theoretically) these samples should be provided, regardless of the reasons causing such disruptions. Therefore, we developed our own, previously mentioned, data acquisition system (SDAS) responsible for synchronizing measurement data from different types of sensors.

The SDAS is a distributed system (see Fig. 1) – with one master controller (MC) and many sensor controllers (SCs) operating as separate processes [7]. The MC synchronizes the operation of the SCs within the SDAS. Individual SCs control data received from sensors. If necessary, they perform sample alignment relative to the expected (theoretical) delivery time. Each SC (according to instructions from the MC) divides the measurement data stream into smaller chunks and stores them in a local repository.

Data is stored on a central network-attached storage (NAS) and monitored by an edge-deployed processing system. Files are transferred to the cloud, where metadata is extracted

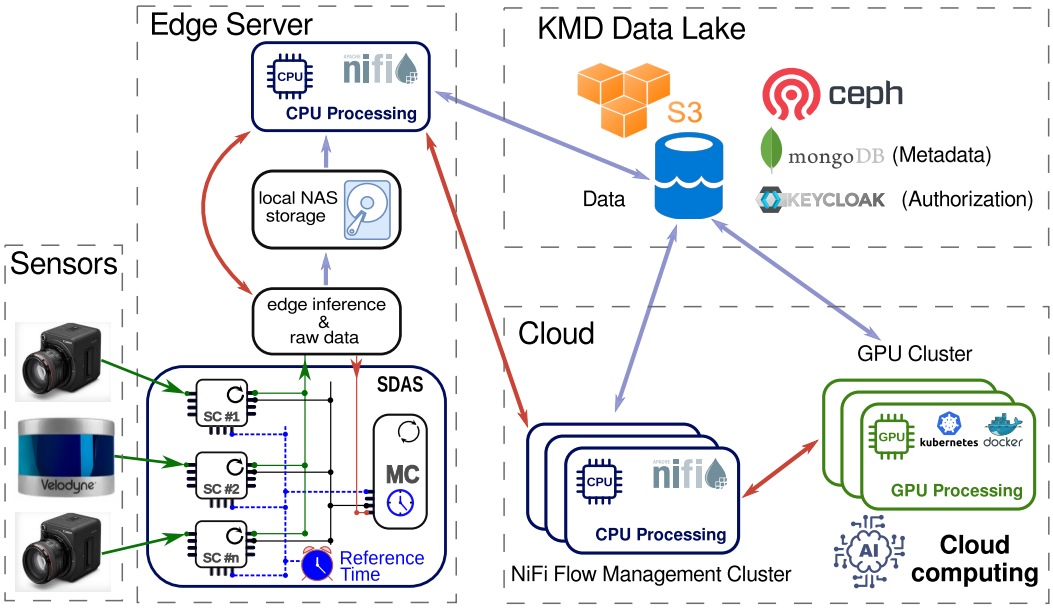

Fig. 1. Workflow of the implemented MLOps lifecycle.

and saved in a dedicated database. Edge-side pre-processing includes object detection using optimized deep learning models.

Dataset Management. The system supports automated preparation of training datasets based on structured metadata stored in a cloud-based registry. Users can query and filter data simultaneously from multiple sensor modalities, such as RGB images, LiDAR scans, and GPS coordinates, using a unified metadata layer. This enables coherent, multimodal dataset creation aligned with specific use case criteria. In addition to automated retrieval, the platform supports manual validation and labeling workflows, allowing users to inspect, refine, or annotate data samples prior to training. The final data sets, including sensor data, annotations, and descriptors, are versioned and uploaded to cloud storage to ensure traceability and reproducibility.

Model Training. Training is conducted in a containerized Kubernetes environment equipped with GPU nodes (NVIDIA A100), supporting parallel execution of workloads across multiple users. The system facilitates resource sharing through role-based quotas and dynamic scheduling. Users can leverage a library of predefined training templates that encapsulate best practices for various architectures, including YOLO (v7–v12), PointRCNN, and Frustum PointNet. These templates automate data loading, augmentation, checkpointing, and metric logging. After training, models are automatically uploaded to the cloud registry and, if approved, can be seamlessly deployed to the edge system. The newly trained model replaces the previously deployed version in the research vehicle, enabling continuous improvement of inference at the edge. The system has been successfully validated during real-world deployments in urban conditions, demonstrating high reliability and responsiveness to detection of hazardous situations [4].

Monitoring and Observability. The platform continuously monitors both training and inference stages, collecting data on GPU utilization, job duration, convergence patterns, and inference latency on edge devices. Logs and performance metrics are persisted in the metadata registry and cloud storage for audit, analysis, and debugging. The system includes mechanisms for detecting data drift and performance degradation, which can trigger retraining workflows. Access to all components and pipelines is secured through integration with a centralized identity provider using OpenID Connect, ensuring multi-tenant isolation, auditability, and secure collaboration among research teams.

4. Conclusion

This paper presents an implementation of a system that integrates vehicle-mounted sensors with cloud-based infrastructure, scalable model learning, and edge processing for real-time inference. The fully automated data feed, built on Apache NiFi, enables continuous data ingestion, metadata enrichment, model versioning, and training orchestration on GPU-enabled Kubernetes clusters.

Our proprietary solution is a software-implemented multispectral data acquisition and synchronization system. Thanks to this system, we were able to effectively create extensive data sets that were used to build and test diverse inference models [4]. The source codes for the components of this system, made available as open source, can be found on the site [6]. We are still working on its development so that in the future we can present its next more advanced versions.

Acknowledgements

This work was carried out as one of the tasks titled: "Edge computing services and applications" under the project: POIR.04.02.00-00-D010/20 „Krajowy Magazyn Danych. Uniwersalna infrastruktura dla składowania i udostępniania danych oraz efektywnego przetwarzania dużych wolumenów danych w modelach HPC, BigData i sztucznej inteligencji”.

References

[1] Filom, S., Amiri, A.M., Razavi, S.: Applications of machine learning methods in port operations–a systematic literature review. Transportation Research Part E: Logistics and Transportation Review 161, pp. 102722 (2022)

[2] Fonseca, C.M., Porello, D., Guizzardi, G., Almeida, J.P.A., Guarino, N.: Relations in ontology-driven conceptual modeling. In: Conceptual Modeling: 38th International Conference, ER 2019, Salvador, Brazil, November 4–7, 2019, Proceedings 38. pp. 28–42. Springer (2019)

[3] Hewage, N., Meedeniya, D.: Machine learning operations: A survey on mlops tool support. arXiv preprint arXiv:2202.10169 (2022)

[4] Karbowiak, Ł.: System wykrywania sytuacji nietypowych w obrębie pojazdów z wykorzystaniem obrazów oraz map głębi, bazujący na uczeniu głębokim. Phd thesis, Faculty of Computer Science and Artificial Intelligence, Czestochowa University of Technology, Czestochowa (2024), https://cloud.icis.pcz.pl/s/89k9dSwrgkF5aSR

[5] Kreuzberger, D., Kühl, N., Hirschl, S.: Machine learning operations (mlops): Overview, definition, and architecture. IEEE access 11, pp. 31866–31879 (2023)

[6] Piatkowski, J., Karbowiak, Ł., Depta, F.: Synchronized data acquisition system. https://github.com/karbol93/Synchronized-Data-Acquisition-System (2024), accessed: 2025-06-26

[7] Piatkowski, J., Karbowiak, Ł., Depta, F.: Synchronized data acquisition system (sdas)-a software approach for synchronizing data recording from multiple sensors. In: International Conference on Information Systems Development (ISD). No. 32nd (2024)

[8] Zhong, R., Salehi, C., Johnson Jr, R.: Machine learning for drilling applications: A review. Journal of Natural Gas Science and Engineering 108, pp. 104807 (2022)