Platform-Agnostic MLOps on Edge, Fog and Cloud Platforms in Industrial IoT

Alexander Keusch[1], Thomas Blumauer-Hiessl[2], Alireza Furutanpey[1], Daniel Schall[2] and Schahram Dustdar[1]
[1]*TU Vienna, Austria*
[2]*Siemens Technology, Austria*

Keywords: IoT, Edge Intelligence, Machine Learning, MLOps.

Abstract: The proliferation of edge computing systems drives the need for comprehensive frameworks that can seamlessly deploy machine learning models across edge, fog, and cloud layers. This work presents a platform-agnostic Machine Learning Operations (MLOps) framework tailored for industrial applications. A novel framework enables data scientists in an industrial setting to develop and deploy AI solutions across diverse deployment modes while providing a consistent experience. We evaluate our framework on real-world industrial data by collecting performance metrics and energy measurements on training and prediction runs of two ML workflows. Then, we compare edge, fog, and cloud deployments and highlight the advantages and limitations of each deployment mode. Our results emphasize the relevance of the introduced platform-agnostic MLOps frameworks in enabling flexible and efficient AI deployments.

1 INTRODUCTION

The vast amounts of data constantly being generated by industrial systems(Trabesinger et al., 2020) that are often analyzed using Artificial Intelligence (AI) or machine learning (ML) methods, as well as the emergence of edge computing systems and AI accelerators, lead to the rise of edge intelligence. (Rausch and Dustdar, 2019; Ding et al., 2022).

Edge computing meets the demands of industrial systems by offering low latency and a privacy-aware approach for processing and analyzing the data (Calo et al., 2017). Despite the availability of powerful edge devices, computational resources, such as memory or processing power are limited(Mendez et al., 2022). The constrained resources limit the volume of accessible data for analysis on edge devices, extend the duration of the analysis process, and restrict the complexity of applicable algorithms. In contrast, significantly more computational resources are available at the cloud and fog levels. However, task offloading requires sufficient bandwidth and reduces privacy (Nastic et al., 2022; Mendez et al., 2022). These observations underscore the inherent advantages and limitations of edge, cloud, and fog deployments. Consequently, there is a need for a comprehensive framework capable of facilitating the development and deployment of AI solutions in a seamless end-to-end Machine learning Operations (MLOps) lifecycle across these diverse layers. Several frameworks encompassing the entire MLOps lifecycle—from data preprocessing and model training and validation to model execution and monitoring—have been developed by leading cloud companies such as AWS and Microsoft Azure. Nonetheless, these tools predominantly center around cloud-based workflows, leaving a gap in the seamless integration of the fog and edge layers(Kemnitz et al., 2023; Rausch et al., 2019).

To fill this gap between the cloud- and the fog- as well as edge layer, this paper presents an MLOps Framework, that provides an end-to-end MLOps workflow on all of these layers. The design goals of this framework are:

- **Platform Agnosticism:** The framework must function seamlessly across edge, fog, and cloud environments. This extends the capabilities of MLOps lifecycle management, which are typically confined to frameworks provided by major cloud providers, to also encompass edge and fog layers.
- **Consistent Experience:** The framework must ensure a uniform user experience for data scientists and operators across all layers. This entails that all layers should be utilized in the same manner, regardless of the specific deployment environment.
- **Data Management:** The framework should support a wide variety of data sources as input for

Keusch, A., Blumauer-Hiessl, T., Furutanpey, A., Schall, D. and Dustdar, S.
Platform-Agnostic MLOps on Edge, Fog and Cloud Platforms in Industrial IoT.
DOI: 10.5220/0012977500003825

In *Proceedings of the 20th International Conference on Web Information Systems and Technologies (WEBIST 2024)*, pages 71-79
ISBN: 978-989-758-718-4; ISSN: 2184-3252

models.

- **Model Management:** Models should be encapsulated in deployment packages that can be uploaded, utilized, and managed consistently across all layers.
- **Integration with Existing Systems:** The framework should seamlessly integrate with existing systems by allowing the results to be pushed back into these systems.

We present a framework that meets these criteria and evaluate its performance through experiments conducted in setups that simulate real-life environments. Specifically, the Framework is deployed on all three deployment modes, and two ML-based data analysis workflows are executed using real-world industrial data. Metrics are then collected for each run of the workflows, which are used to compare edge, fog, and cloud executions and discuss the advantages and limitations that the deployment modes result in.

In summary, our contributions are

- introducing a novel platform-agnostic MLOps framework
- demonstrating the practicability of the general solution approach with two representative industrial ML workflows
- contrasting the differences between deployment on edge, fog, and cloud layers

Section 2 discusses relevant related work. Section 3 introduces the core MLOps framework. Section 4 describes the evaluation methodology and experiments. Lastly, Section 5 concludes the work with a brief discussion on limitations and future outlook.

2 RELATED WORK

Kreuzberger et al.(Kreuzberger et al., 2023) discuss the challenges in automating and operationalizing machine learning projects for production, leading to many ML endeavors failing to meet expectations. To address these challenges, the concept of Machine Learning Operations (MLOps) is introduced, focusing on automating and managing ML workflows effectively.

MLOps emphasizes collaboration, automation and continuous integration and delivery, drawing parallels with the DevOps paradigm in software engineering. By leveraging insights from DevOps experiences, MLOps seeks to improve the efficiency and reliability of ML systems. Their work concludes by emphasizing the importance of MLOps in bridging the gap between ML research and practical deployment, ultimately increasing the number of ML proofs of concept that transition to production.

This work extends on the MLOps paradigm by providing a framework that employs the key principles in a platform-agnostic manner.

John et al.(John et al., 2020) present an architectural framework for edge computing in AI. The authors present five architectures that facilitate the deployment of AI models on edge devices using edge-cloud collaboration. They conducted a qualitative interview to evaluate the proposed architectures. The validation study reveals insights from experts in different industries regarding the applicability and challenges of each architectural alternative. Companies prefer architectures that balance centralized and decentralized approaches with data collection, model optimization, and cost-effectiveness considerations. However, the authors do not provide a platform-agnostic MLOps framework, which is the focus of our work.

Raj et al. (Raj et al., 2021) propose an edge MLOps framework capable of automatically deploying and managing machine learning models on edge devices. They focus on scalability and automatization of the deployment process. They again focus on synergizing edge and cloud computing. This paper, in contrast, focuses on providing a framework that can be deployed on cloud, fog and edge level individually, without the need to use cloud resources at all.

Kemnitz et al.(Kemnitz et al., 2023) introduce a framework for building and operating AI models at the industrial edge, addressing challenges in deploying and managing AI applications in such scenarios. It introduces the concept of model artifacts and discusses three industrial AI model use cases: energy efficiency monitoring, predictive maintenance, and parameter forecasting. The framework aims to streamline AI model deployment, operation, and management, catering to various user roles, including data scientists, automation engineers, and service technicians. Key contributions include requirements elicitation based on user roles, framework design, and qualitative evaluation based on implemented use cases. The framework seeks to enable seamless integration of edge resources into AI workflows, focusing on scalability, ease of deployment, and operationalization without requiring software engineering expertise. While their work focuses exclusively on processing at the edge, our approach includes intermediate fog and cloud nodes to consider an edge-cloud compute continuum.

3 SYSTEM DESIGN

This section describes the architecture of the platform-agnostic MLOps framework. It shows, how the framework can be used to deploy machine learning models on different platforms, while providing a consistent interface for data scientists and operators.

3.1 System Architecture

Fig. 1 shows the system architecture. The MLOps Framework can be deployed on docker-capable devices, which can be a cloud server, a fog node or an industrial edge device.

The asset, whose data is used for training and later analyzed, is typically located on the field level. The edge device is located on this level as well.

The asset can either output its data (examples of such data is described in Section 4.2) directly to an MQTT broker or an industrial edge connector can be used to access the asset data and convert it to a standardized format which is then forwarded to the MQTT broker. Each deployment has an MQTT broker which has to be accessible from the field level, in order to connect the framework to the asset. MQTT is chosen as a protocol, as it offers a lightweight and easy to use messaging system, however, also other messaging solutions could be chosen.

Once the asset data is pushed to the MQTT broker it is ingested by the MLOps framework. This then uses the data for training a machine learning model or for generating predictions using the trained model. These predictions might assign classes to data points, as described in Section 4.2, try to predict future anomalies of the asset operation or do any other kind of analysis a data scientist might come up with.

These predictions can be pushed back to the MQTT broker, which makes them available on the field level. For example, to integrate them into the Siemens Industrial Edge Platform, which offers i.e. a dashboard for visualizing the data or connectors to use the data for further controlling of the asset.

The MLOps framework also offers a consistent view of the system to data scientists and operators on all levels. Operators can use the web interface to configure the data ingest via MQTT and monitor the system's status. Data scientists can use it to deploy their machine learning models, packaged as zip files containing the Python sources and the required libraries. After uploading the model, the data scientists can configure the model's training, monitor its performance and manage and monitor the prediction stage.

3.2 Application Architecture

Fig. 2 shows the internal structure of the MLOps framework. On the left are the assets connected to the industrial edge platform, which pushes their data to the MQTT broker. They do this by directly connecting to the MQTT broker or using an industrial edge connector. Industrial edge connectors are part of the Siemens Industrial Edge Platform and exist for many commonly used industrial protocols. The connectors interface with the asset and read its data via the respective protocol, converting the data to a standardized format [1] and push it to the MQTT broker.

Data in this standardized format can be ingested directly by the backend-controller service of the MLOps framework. To directly connect to assets that push data to the MQTT broker without using an industrial edge connector, High-Level Drivers (HLD) are used. These custom modules can be implemented and added to the MLOps framework. They manage data in any arbitrary format the asset provides and modify it for ingestion by the backend-controller. The connections can be configured via the web interface of the MLOps framework. Examples of such High-Level Drivers are the GENICAM HLD, which enables communication with cameras supporting the Genicam protocol, or the OPCUA HLD which accesses data via OPCUA.

The backend-controller is the core of the MLOps framework. As described above, it handles the data ingest before the data is stored in an internal database. Furthermore, the backend-controller handles deployment, training, monitoring, and management of machine learning workflows. Workflows can be uploaded in a package format in the web interface, making them available for use in the MLOps framework. After uploading the ML workflows, training can be started via the web interface. For this, data is selected from the previously collected asset data, which is then sent together with the ML workflow package to the runtime, which executes the training routine defined by the workflow on the selected data. The trained model artifact is then returned to the backend-controller, and the workflow is ready for use in the prediction stage. The prediction stage can be initiated via the web interface. In this stage, the backend-controller sends the ML workflow package and the trained model artifact to the runtime. The runtime executes the routine defined by the workflow package for each relevant data point and generates a prediction using the trained model. These predictions are

[1] https://github.com/industrial-edge/common-databus-payload-format/blob/main/docs/payload-format/PayloadFormat.md

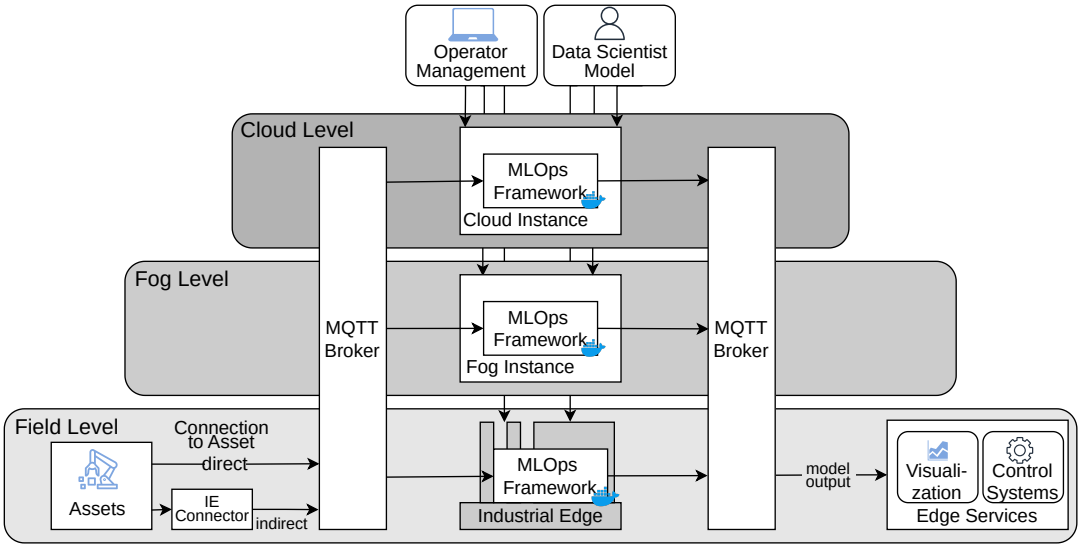

Figure 1: System Architecture.

collected by the backend-controller and visualized via the web interface. The predictions are also pushed back to the MQTT broker, where they are accessible to the industrial edge platform. The industrial edge platform uses the predictions, i.e. for further visualizations or for controlling the assets.

4 EVALUATION

This section shows how different deployment modes of the MLOps framework perform on real-world industrial data. For this, we deploy the application to machines on the cloud, fog and edge layers and execute two ML workflows using real-world industrial data. The experiments should illustrate the differences between individual deployment modes. Our hypothesis is that the three deployment modes—cloud, fog, and edge—each have their own advantages and disadvantages, with no clear overall winner, which emphasizes the necessity for a platform-agnostic approach to MLOps.

4.1 Experiment Setup

For running the MLOps framework, a representative machine exhibiting typical performance characteristics has been selected for each designated location. The edge level is represented by a Siemens industrial PC (SIMATIC IPC427E), equipped with an Intel Xeon CPU E3-1505L v5 @ 2.00GHz and 16 GiB of RAM. The fog and cloud locations use virtual machines based on OpenStack and AWS. The Fog VM features 8 vCPUs based Intel Xeon SP Gold 6230 20C/ 40T - 2,1GHz/ 3,9GHz and 8 GiB of RAM, representing a medium-sized OpenStack instance (c1.medium). On the cloud level, an AWS EC2 instance with 2 vCPUs based on the AMD EPYC 7000 series and 8 GiB of RAM corresponds to a large instance (t3a.large). A large instance was chosen on the cloud level, compared to a medium instance on the fog level, to represent the greater availability of computing resources in the cloud.

4.2 ML Workflows

Two ML workflows that analyze industrial data are used to run the experiments and collect metrics. The following sections provide a quick overview about these workflows:

4.2.1 PCB Quality Inspection

This ML-based data analysis workflow analyzes data from a printed circuit board (PCB) manufacturing process. In this process, solder paste is placed on PCBs, a procedure prone to error. During this process, 16 numerical measurements are collected, which include e.g.: height, area, offset, x- & y-positions. The ML model uses these measurements to classify the PCBs into the classes `OK`, `error`, and `pseudo-error`. The ML model uses a supervised learning approach with labels by human inspectors for training.

4.2.2 Inertial Measurement Classification

This ML workflow detects human actions in an industrial setting from sequenced events. These actions include hammering, screwing, sawing, etc. To classify a

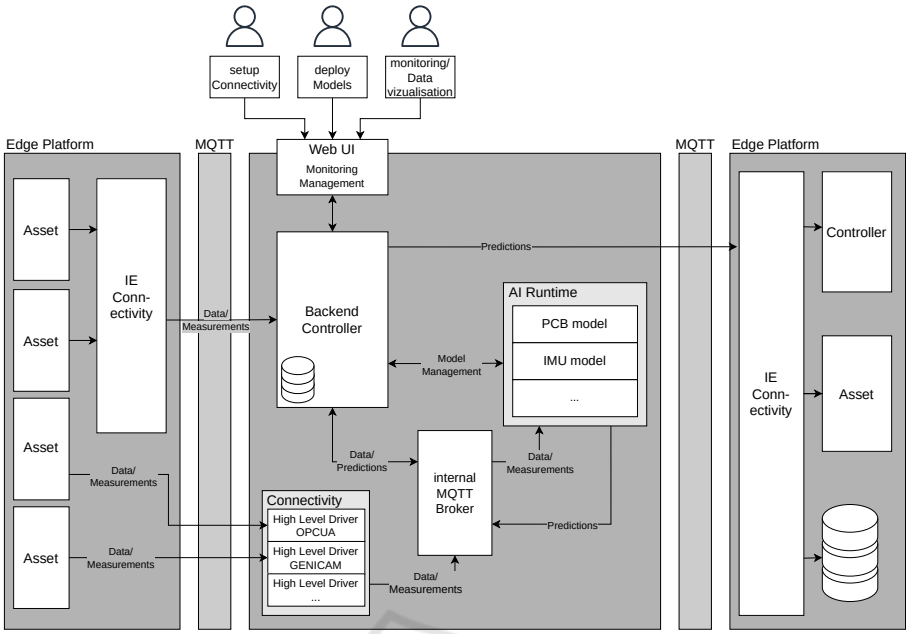

Figure 2: Application Architecture.

person's actions, timeseries data is collected using inertial measurement units (IMU) placed on the left and right hands of workers. The timeseries data is manually labeled for training, while the prediction stage should assign a category to the timeseries data automatically.

4.3 Experiment Runs

For evaluation, both ML workflows described above are executed on the edge, fog and cloud deployments of the MLOps framework. During each of two runs, measurements are collected and used to compare the layers. The collected measurements are as follows:

- Training Time: the time a machine needs to complete the training procedure of the respective ML workflow
- Prediction Time: the time needed to ingest and process a data sample to generate a prediction/-classification based on the previous training. This is split into the time required for the system to ingest the data point and the time that it needs to calculate the result.
- Communication Latency: latency of the asset to the respective machine
- Memory Utilization: maximum memory utilization of the MLOps framework while executing the ML workflows
- CPU Utilization: maximum CPU usage of the ML workflow
- Energy Consumption: the amount of energy the machines use for training and generating predictions. This is generated by code carbon[2], a Python package that estimates the amount of energy a device uses to execute a program. The values generated by code carbon are estimates based on CPU utilization, CPU type, time of calculations as well as CPU power tracking if available[3].

The experiment setup can be seen in Fig. 3. The edge device is located directly on a shop floor in Vienna (AUT), the fog instance (openstack-VM) is hosted in a data center nearby located in the same building. And the cloud instance (EC2-VM) is hosted in the nearest AWS data center in Frankfurt (GER).

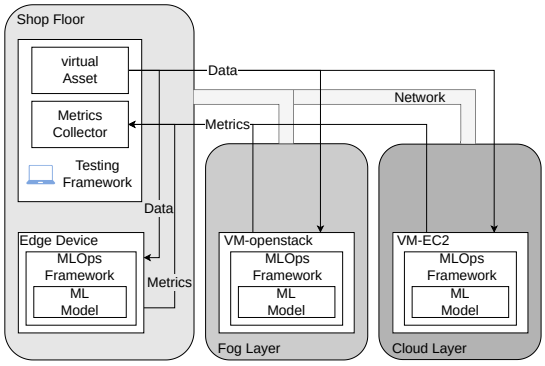

Figure 3: Experiment Design.

[2]https://codecarbon.io/

[3]https://mlco2.github.io/codecarbon/methodology.html#power-usage

A Testing Framework we developed is executed on a machine located on the same shop floor as the edge device for testing the setup. The framework provides a virtual asset corresponding to the type of asset expected by the respective ML workflow. e.g.: PCB data and IMU data The testing framework also automatically starts training runs on all deployments and collects training time, CPU & memory utilization, and energy consumption as described above. After the training, a set of ten prediction cycles is started. For this, a data point is generated by the virtual asset and ingested into the system. A prediction is then generated, and metrics like prediction time, energy consumption and CPU & memory consumption are calculated. Finally, communication latency is measured by doing a series of pings.

4.4 Results

For each workflow, two training sessions and 20 prediction cycles were performed. The collected metrics are as follows:

Table 1 shows the system measurements that have been collected. The first column shows the communication latency from the asset on the shop floor to the respective machines. This clearly shows the advantage of the edge device being in the same room, causing a latency of only 8.5ms. In contrast, the fog and cloud machines exhibit a much greater latency.

The second column shows the maximum CPU utilization of a training/prediction cycle of the PCB workflow. Here 100% corresponds to the complete utilization of a single CPU core. The measurements show a high utilization of the edge device due to the weaker hardware, whereas fog and cloud have a lower utilization due to more powerful hardware.

The last column shows the maximum memory utilization of the PCB workflow runs. These do not show any significant differences. As long as the application does not exceed the available memory on the given machine, no swapping occurs, which could influence the memory footprint.

CPU and memory utilization were only collected for the PCB workflow. All further measurements are collected for both workflows and are shown in the following figures.

4.4.1 Training

Fig. 4 shows the training time of the PCB and IMU workflows on the cloud, fog and edge machines. The bar plots show the mean of the two executed runs, and the error bars show the standard deviation.

The green bars show the training time of the PCB workflow. Despite the different hardware equipment,

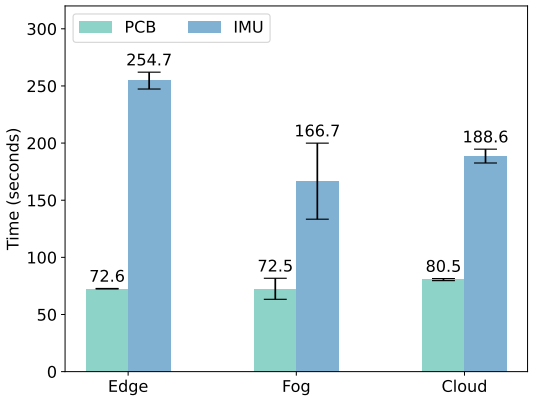

Figure 4: Training Time.

the training time of the PCB workflow shows almost no differences between the deployment modes. This is because the training algorithm of the PCB workflow requires minimal effort. Most of the time comes from deploying the workflow via a zip file (e.g. from the framework extracting the file and adding it to the AI runtime). The duration of this process seems to depend very little on the machine's hardware, as the compression implementation does not utilize the full hardware potential.

The blue bar shows the training time of the IMU workflow. The training algorithm takes a much larger share of the overall training effort in this workflow. The training on the edge device takes the longest, which is due to the weaker hardware. The training on the fog and cloud machines is faster, however there is only minimal difference. That the difference is not greater is possibly due to the not fully parallelized training algorithm. Because of this lack of parallelization, the time depends primarily on the single core performance of the machines, which while similar on both machines, still differs due to the different architectures of AMD and Intel CPUs.

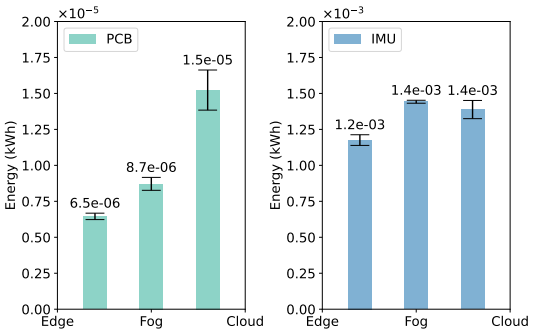

Figure 5: Training Energy Consumption.

Fig. 5 shows the energy consumption of the training runs of the PCB and IMU workflows. Again, the bar plots show the mean of the two executed runs and

Table 1: System measurements.

platform	latency [ms]	CPU utilization [%]	memory consumption [MiB]
edge	8.5	313.5	1144
fog	173.8	150.7	1138
cloud	241.9	146	1169

the error bars show the standard deviation. The plot is split into two diagrams, due to the different scales for each workflow. The left plot shows the energy consumption of the PCB workflow in kWh. The energy consumption is very low because the training algorithm is not very demanding. Nevertheless, it shows that the edge device consumes the least energy, due to the weaker hardware, while the fog machine consumes a little more than the edge device. The cloud machine consumes the most energy, due to being the most powerful hardware.

The right plot shows the energy consumption of the IMU workflow. Here the difference between the edge device, which consumes the least energy, and the fog and cloud machines, which consume more and roughly the same amount of energy, is less pronounced.

This is due to the weaker hardware of the edge device, which on the one hand consumes less energy, but on the other hand also takes longer to complete the training, which in turn raises the total amount of spent energy again.

The cloud and fog machines consume roughly the same amount of energy, as they also consume a very similar amount of time for training.

4.4.2 Prediction

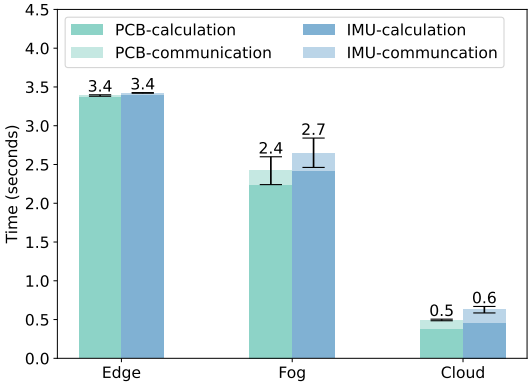

Figure 6: Prediction Time.

Fig. 6 shows the prediction time of the PCB and IMU workflows. The bar plots show the mean of 20 runs that we executed, and the error bars show the standard deviation.

The bars are further split into two sections. The darker section shows the time that the respective system uses for the calculation of the model output, while the lighter section on top shows the time that is added by the system to ingest the data point, therefore representing the communication latency. Both sections combined show the total time that is needed to generate the model output.

Both workflows show a very similar pattern. The measurements of the edge device show that the latency is very low but due to the weaker hardware, the time that is needed for the calculation of the model output is the highest. Therefore, the total time needed for the prediction is the highest, as the low latency has only a minuscule impact on the overall prediction time. On the fog machine, the time needed for calculation of the model output is lower, due to the more powerful hardware, but the latency is higher due to the greater physical distance to the asset. So the impact of the latency on the overall prediction time is higher. However, the overall prediction time is still lower than on the edge device. The cloud machine offers the lowest overall prediction time, even though the latency has a greater relative impact on the overall time than on the other devices. Due to the much faster calculation of the model output, the overall prediction is still the fastest, however.

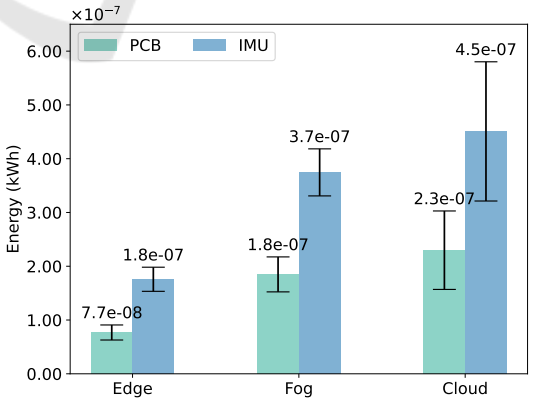

Figure 7: Prediction Energy Consumption.

Fig. 7 shows the energy consumption of the prediction runs of both workflows. The bar plots show the mean of the energy consumption of 20 runs in kWh and the error bars show the standard deviation.

Both workflows show a similar pattern. However, the IMU workflow shows on all platforms approximately double the energy consumption of the PCB workflow, due to the more demanding algorithm.

The edge device consumes the least energy, due to the weaker hardware. While the fog and cloud machines consume more energy, with the cloud machine having the highest energy consumption.

4.5 Discussion

As demonstrated in Section 4, there are several differences when running machine learning tasks on various deployment modes. The individual implications of the results provided are discussed as follows:

Resource Availability: The measurements of the training, as well as the prediction time show that the higher levels in the deployment hierarchy can benefit significantly from the computational resources available on these levels. The resources available on the cloud and fog level greatly exceed the capabilities of typical edge devices. This can benefit particularly computationally demanding tasks such as machine learning. We showed that the performance advantage of cloud and fog machines can even exceed the latency advantage provided by edge devices due to their proximity to the assets.

Energy Consumption: The measurements conducted within this paper show that due to the weaker hardware resources of edge devices, they can execute the ML tasks with lower overall power consumption. This makes the usage of edge devices economical and environmentally friendly.

The energy measurements are however limited, as they rely on a software package that tracks CPU and memory usage. This provides reasonably accurate energy consumption estimates, but the concrete kWh values might not be entirely correct. However, as the same method is used for all measurements, the relative difference between the measurements is still valid.

Qualitative Aspects: Other differences between the deployment levels are qualitative aspects, such as security or privacy. With edge computing and possibly fog computing, the data is processed on the edge device or fog instance, which can be beneficial for privacy, as the data does not have to be transmitted to a remote server possibly located in a different country(Mendez et al., 2022; Satyanarayanan, 2017). This can be especially important for industrial applications, where the data might be sensitive and strict regulations may apply.

Relevance of Platform-Agnostic MLOps: These findings underline the relevance of a platform-agnostic MLOps framework that enables data scientists and operators to deploy machine learning models on different platforms while providing a consistent experience. Platform-agnostic MLOps frameworks enable users to choose the platform that best fits their requirements without having to adapt their workflows to the selected platform.

Feedback for the Operator: The experiments above show that the Testing Framework provides useful metrics for deploying an ML task. An operator can use the Testing Framework to get crucial insight for assessing the performance and viability of a deployment location for a specific ML scenario. These insights can be used to inform optimal deployment decisions tailored to the needs of the given ML task.

Generalizability: The results of our experiments are expected to be generalizable across various machine learning tasks. Performance differences among the cloud, fog, and edge layers are anticipated to be existent for most ML tasks, indicating that different tasks may be better suited to different layers depending on specific requirements and settings. This suggests that a flexible, platform-agnostic approach is crucial for optimizing MLOps across diverse scenarios.

Hybrid Deployments: Deployment is often not a binary choice; for instance, training might be best performed in the cloud or fog, while inference is more suitable for the edge. Our experimental results highlight differences between the layers, suggesting that hybrid deployments might be advantageous. However, these hybrid approaches are not considered in the current study and are suggested for future research.

5 CONCLUSION

This work presented a platform-agnostic MLOps framework, which can be deployed on the cloud, fog and edge level. The framework enables users to choose the platform that fits their requirements best, without the need to adapt workflows and practices to a specific deployment mode. We also evaluated the framework on real-world ML tasks and discussed the implications that the deployment mode has on the given task. These results underline the need for platform-agnostic MLOps.

Future research should explore the potential of hybrid deployments, where different stages of the MLOps lifecycle are distributed across cloud, fog, and edge environments. Additionally, incorporating

automatic orchestration mechanisms for detecting the most suitable platform for each task and deploy models accordingly would enhance efficiency and performance. This would involve developing intelligent systems capable of dynamically optimizing deployment strategies based on task requirements and resource availability. These advancements could significantly improve the flexibility and effectiveness of MLOps frameworks.

REFERENCES

Calo, S. B., Touna, M., Verma, D. C., and Cullen, A. (2017). Edge computing architecture for applying AI to IoT. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3012–3016.

Ding, A. Y., Peltonen, E., Meuser, T., Aral, A., Becker, C., Dustdar, S., Hiessl, T., Kranzlmüller, D., Liyanage, M., Maghsudi, S., Mohan, N., Ott, J., Rellermeyer, J. S., Schulte, S., Schulzrinne, H., Solmaz, G., Tarkoma, S., Varghese, B., and Wolf, L. (2022). Roadmap for edge AI: A Dagstuhl perspective. *ACM SIGCOMM Computer Communication Review*, 52(1):28–33.

John, M. M., Holmström Olsson, H., and Bosch, J. (2020). AI on the Edge: Architectural Alternatives. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 21–28.

Kemnitz, J., Weissenfeld, A., Schoeffl, L., Stiftinger, A., Rechberger, D., Prangl, B., Kaufmann, T., Hiessl, T., Holly, S., Heistracher, C., and Schall, D. (2023). An Edge Deployment Framework to Scale AI in Industrial Applications. In *2023 IEEE 7th International Conference on Fog and Edge Computing (ICFEC)*, pages 24–32.

Kreuzberger, D., Kühl, N., and Hirschl, S. (2023). Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access*, 11:31866–31879.

Mendez, J., Bierzynski, K., Cuéllar, M. P., and Morales, D. P. (2022). Edge Intelligence: Concepts, Architectures, Applications, and Future Directions. *ACM Transactions on Embedded Computing Systems*, 21(5):48:1–48:41.

Nastic, S., Raith, P., Furutanpey, A., Pusztai, T., and Dustdar, S. (2022). A Serverless Computing Fabric for Edge & Cloud. In *2022 IEEE 4th International Conference on Cognitive Machine Intelligence (CogMI)*, pages 1–12.

Raj, E., Buffoni, D., Westerlund, M., and Ahola, K. (2021). Edge MLOps: An Automation Framework for AIoT Applications. In *2021 IEEE International Conference on Cloud Engineering (IC2E)*, pages 191–200.

Rausch, T. and Dustdar, S. (2019). Edge Intelligence: The Convergence of Humans, Things, and AI. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pages 86–96.

Rausch, T., Hummer, W., Muthusamy, V., Rashed, A., and Dustdar, S. (2019). Towards a Serverless Platform for Edge {AI}. In *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*.

Satyanarayanan, M. (2017). The Emergence of Edge Computing. *Computer*, 50(1):30–39.

Trabesinger, S., Butzerin, A., Schall, D., and Pichler, R. (2020). Analysis of High Frequency Data of a Machine Tool via Edge Computing. *Procedia Manufacturing*, 45:343–348.