Quality Assurance in MLOps Setting: An Industrial Perspective

Ayan Chatterjee[1], Bestoun S Ahmed[1,2], Erik Hallin[3] and Anton Engman[3]

[1]*Department of Mathematics and Computer Science, Karlstad University, Universitetsgatan 2, Karlstad, 651 88, Sweden*

[2]*Department of Computer Science, FEE, Czech Technical University in Prague, Czechia*

[3]*Uddeholms AB, Uvedsvägen, Hagfors, 683 33, Värmlands län, Sweden*

Abstract

Today, machine learning (ML) is widely used in industry to provide the core functionality of production systems. However, it is practically always used in production systems as part of a larger end-to-end software system that is made up of several other components in addition to the ML model. Due to production demand and time constraints, automated software engineering practices are highly applicable. The increased use of automated ML software engineering practices in industries such as manufacturing and utilities requires an automated Quality Assurance (QA) approach as an integral part of ML software. Here, QA helps reduce risk by offering an objective perspective on the software task. Although conventional software engineering has automated tools for QA data analysis for data-driven ML, the use of QA practices for ML in operation (MLOps) is lacking. This paper examines the QA challenges that arise in industrial MLOps and conceptualizes modular strategies to deal with data integrity and Data Quality (DQ). The paper is accompanied by real industrial use-cases from industrial partners. The paper also presents several challenges that may serve as a basis for future studies.

Keywords

Quality assurance, machine learning, automated software engineering, software testing, data integrity, data quality, MLOps.

1. Introduction

Traditionally, the software development life cycle includes quality assurance (QA) practices and automated tools to assess the quality of the software system. Due to the increasing functionalities of recent developments in cyber-physical systems in Industry 4.0 and human-centered development in Industry 5.0, ML programs that have core functionality are becoming increasingly ML software systems. ML software development in operation is an ongoing development in automated software engineering that sets up automated pipelines for continuous training and deployment of ML software. Recent advances toward a continuous development cycle of ML software are known as MLOps [1] or AIOps [2]. An MLOps pipeline, for example, in Amazon Web Services (AWS) [3] and Microsoft Azure [4], generally consists of ML software development and automated deployment and monitoring (or Ops). Then, trained ML software is deployed for real-time prediction and classification tasks and an action. Inspired by such architectures, we conceptualize an infographic diagram of the overview of the MLOps architecture in Figure 1, which shows the contin-

uous and automated development of ML software. Such ML software, which does not follow the conventional way of authoring software code, is a black box [5] and is data-driven [6]. Thus, shifting the requirements for QA, which previously relied on exposure to software code for Software Quality Assurance (SQA) and Data Quality (DQ) professionals and tools for data QA [7].

Data and SQA ensure compliance with the level of accuracy, data security, and performance scripted in an organization's QA policies and enforced by the SQA plan in classical software development in operation (DevOps) [8]. Such a QA plan in MLOps requires robust software testing approaches and data QA in a software development pipeline. Recent developments in SQA of ML are enabled by methods such as domain-specific software testing [9] and mutation testing [10] that attempt to detect faults in ML software by systematically modifying the test data. When it comes to data QA, data integrity, trustworthiness, and DQ, they are measured by individual data dimensions such as timeliness, uniqueness, relevancy, and location intelligence [11]. However, the dimensions for a given software pipeline are pre-determined for a given application, and we discovered a lack of implementation for industrial MLOps in our search.

This paper attempts to address the following research questions (RQs):

- **RQ 1: What are the industrial QA challenges when conventional software testing and QA are not viable for MLOps?**
  We collaborated with partners in the manufacturing and utilities sector to discover and outline the
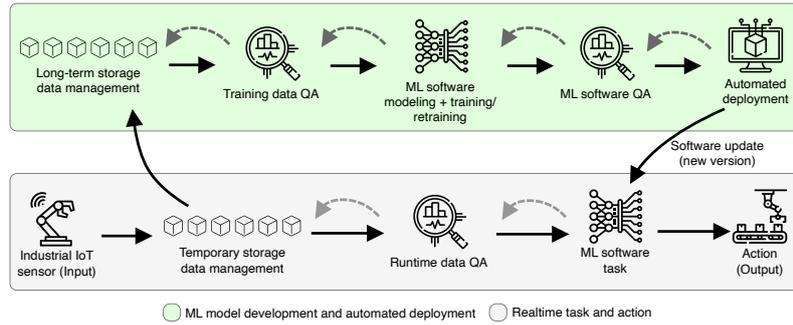
Figure 1: A high-level infographic overview of QA in MLOps architecture, inspired from AWS [3] and Azure [4] architectures. (The straight lines in the figure represent data flow and the dashed lines represent feedback.)

QA challenges in industrial MLOps for real Use Case (UC)s.

- **RQ 2: What approach can we take when dealing with QA in industrial MLOps?**
  We propose concept for a modular approach to data QA, where each component of data QA is an ML software in the industrial MLOps architecture mentioned in Figure 1, with automated training, selection and deployment of data dimensions. Furthermore, we conceptualize a practical and applicable MLOps architecture to enable QA and the continuous delivery of ML software.

2. MLOps software architecture and QA challenges

2.1. Architecture

Network architecture has undergone significant advances in recent years [12, 13], making it possible for modern industries and factories of the future to offer a backbone for MLOps setup, minimize latency, and enable time-sensitive hard and soft deadline tasks. As a result, we have identified three network architectures, each of which depends on how far Internet of Things (IoT) data travel for ML software tasks. The architecture categories are as follows:

- **End-to-end on-device processing:** The first is an end-to-end software pipeline where data acquisition, QA, ML software processing and all actions occur on the same device. These are relatively lightweight, usually a single action tiny ML task. Even if the ML software, in this case, is initially delivered externally, adaptation and retraining of the model occur on the fly. Due to the lack of processing power and devices running on a battery, QA is challenging in this situation.

- **Data processing on edge and fog nodes:** In contrast to the previous design, this architecture is built on local networks to handle computational operations on edge or fog nodes. Although this architecture has increased latency compared to on-device processing, this has the advantage of performing data analytics and action using information from multiple sources in the network.

- **IoT-edge/fog-cloud architecture:** The third architecture utilizes cloud services such as Amazon Web Services, Google Cloud, and Microsoft Azure, in addition to processing on-device, edge, and fog. This architecture has the highest latency among others, but it has the ability to execute collective analytics from all connected devices and historical data and is not resource constrained. Additionally, this enables centralized, decentralized, and federated ML applications.

Although the architecture categories differ for each application, they combine to form a hybrid network architecture for a suite of automated ML applications. Furthermore, because the hardware resources and time constraints available on the IoT device, the edge or fog network, and the cloud are vastly different, this creates a challenge for QA for industrial MLOps.

2.2. QA Challenges

Working in collaboration with industrial partners, we have identified the following challenges in relation to QA that are applicable to any architecture:

- **Modelling challenges:** External factors such as anomaly and noise deviate from the sensor data or signals from its ideal measurements. Furthermore, QA for ML must automatically accommodate class imbalance and drifts [14], where the current instance of the containerized ML application has observed changes in the test data

compared to the data on which it was trained. There is a lack of robust AutoML strategies and algorithms for such external factors in automated QA.

- **Resource, time, and scalability:** While some industrial processes are sparse, others are frequent enough to produce big data. For big data, it is challenging to perform the necessary data QA stages in a timely manner. In addition, industries scale up frequently, and system integration of new hardware may differ from existing machinery. The rapid adoption of new hardware in existing QA processes poses a scalability issue.

- **Architectural constraints:** Industrial MLOps architectures have network components, each with its capabilities and limitations. Network components, such as routers and switches, have capabilities such as the number of data packets they can handle at a given time, network scheduling, and routing. Additionally, these components are subject to network attacks, such as distributed denial of service, for example [15]. This contributes to the total latency that a data packet needs to traverse from IoT sensors to QA microservices running on edge/fog and is a challenge for a robust and scalable architecture.

- **Lack of production data in manufacturing processes:** Some manufacturing operations are time-intensive or infrequent/sparse, typically occurring on average once or twice a day. One such example is electroslag remelting in the steel industries [16]. This indicates that such operations will have a few hundred manufacturing events over the course of a year, and the lack of data is a challenge for the ML software. Although data augmentation or a principled approach to generating synthetic data has been used to fill the gaps, it poses a QA challenge to strategically split the limited ground truth data for robust training and testing [17].

- **Compliance with regulatory, export control, and ISO standards:** Data is often subject to regulatory requirements from government entities. For example, with medical equipment that requires additional regulatory requirements for increased safety, user data is subject to the General Data Protection Regulation (GDPR[1]) in Europe and defense data are subject to export control. These regulations are local to a region and are not universally applicable. In automated and continuous ML software development, adhering to such regional regulations is challenging for automated QA.

---
[1]https://gdpr-info.eu/

3. Modular QA for industrial MLOps

To accommodate both lightweight and cloud-based ML software pipelines for industrial applications, we designed a modular QA solution. The modular architecture is predicated on the notion that every data QA step needs its unique collection of dimensions. And the set of data QA dimensions for each data QA step is determined by the answer to the following question in an organization's SQA plan: *What are the automated steps taken as a result of data QA?*

3.1. ML software architecture

Knowledge of automated steps or actions serves as the foundation for QA strategy. To make this behavior possible, we divided the overall architecture of QA for ML into three phases: (i) definition and formulation, (ii) dimension selection, and (iii) QA model training, each answering the following questions:

- **Definition and formulation:** *How does an organization define and formulate QA dimensions such as trustworthiness, relevancy, and privacy?*
- **Dimension selection:** *What minimum QA dimensions are necessary to achieve the QA actions?*
- **QA model training:** *How the selected QA dimensions can be trained and weighed automatically in industrial MLOps?*

3.1.1. Definition and formulation

In the first stage, given the objectives and QA policies of an organization, formal definitions and mathematical formulations need to be developed for each QA dimension. While the rest of the QA for the ML strategy is automated, all QA dimensions require a definition. The definition changes as the organization's policy shifts. Although some QA dimensions are universal, others depend on the objectives of a company. Sensor noise, for example, is an ubiquitous component that affects all streaming data from industrial IoT sensors. Other QA indicators, such as contextual DQ and data integrity depend on the QA strategy of the organization. Furthermore, for MLOps, it is also important to define QA actions. For example, a QA action is to identify whether the streaming data is relatively clean of noise and anomalies with intrinsic DQ metrics and applicable to current ML software in production with contextual DQ metrics.

3.1.2. Dimension selection

Following the definition of all dimensions of QA, the next step is to determine which dimensions are relevant to a
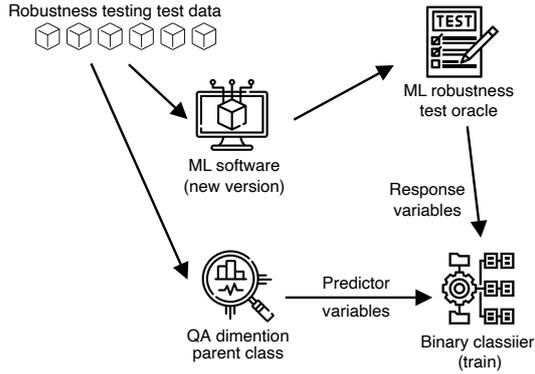
Figure 2: Robustness testing of each parent class in the model training phase.

given set of QA actions. The QA dimensions are then arranged in a connected graph structure with parent-child relationships. For example, in the literature, these parent-child relationships are structured as follows: data integrity (parent) with DQ, data integration, data enrichment, and location intelligence (children). Data completeness, timeliness, and relevancy (children) are aspects of DQ (parent), further classified into intrinsic and contextual subcategories. In QA for ML, the selection of dimensions is achieved by first calculating all dimensions of DQ from the test data and then minimizing the number of child nodes required. We have combined dimension selection and model training, which can be done iteratively as one process.

3.1.3. QA model training

The final step before deployment is QA for the ML model training. An approach to this problem is to iteratively perform dimension selection and model training, where the QA for ML model is trained with dimensions fixed, and dimension selection is performed to promote sparsity in the QA for ML graph while keeping the model fixed. Both steps are performed iteratively until the QA for ML model converges. For each parent in the model training process, robustness testing, a software testing methodology that examines the boundary conditions of a software [18], is carried out with the test data for a binary pass/fail score, as shown in Figure 2.

The outcomes of each parent's robustness testing are then considered for the desired automated actions, allowing action-based QA for ML. For example, a score of "pass" for "intrinsic DQ" and "fail" for "contextual DQ" means that the streaming data are relatively clean and usable without the need of data cleaning (i.e., less noise, anomaly-free, and without NaN/missing values) but is not relevant for the current ML software in production. Still, it might be relevant for other ML software. In this

case, with QA actions, the streaming data are set up to be sent to the QA for ML strategy in a different software pipeline or offline storage for future study.

3.2. Real-time QA classification and action

After training and testing, the next stage is to deploy QA for ML model using automated software deployment methods. Docker[2] and kubernetes[3] are common tools for the deployment of ML software in a containerized manner. One such roadmap is to use Python scripts for model training and then generate a Docker container. The Docker container is then pushed to the Docker Hub, where it is then pulled to the Kubernetes edge nodes for real-time processing of the streaming data and action.

To support industrial UCs and QA in MLOps, we designed an applicable and practical IoT-edge-cloud architecture for the continuous delivery of ML software. UCs in industrial MLOps require support for two types of data transmission across the network: application and monitoring data. The incoming streams of IoT sensor data necessary for routine industrial applications are known as application data. The network data reflect the monitoring information, such as CPU and memory use in real time. To accommodate both types of data in the architecture, we designed the architecture in figure 3 that handles both real-time and offline data by combining microservices on the edge node with cloud services for ML training and offline applications. The architecture supports the following:

- **Automated ML model retraining**: ML software, including AutoML software, once trained in 'cloud services' is automatically deployed to the edge as an Edge AI software on the 'edge node'. Once deployed for the first time, data flows from the network to the delivery and collection microservices to the cloud API layer. Once model degradation has occurred, the ML software is retrained or adapted in the cloud. A new version of the ML software is automatically deployed on the edge node.
- **Real-time and automated decision and action**: Data transfer from the network to the delivery and collection microservices to the microservices for UCs. Microservices for UCs are then sent to Action Services to execute the appropriate action. The action signal is sent back to the network.
- **Automated QA assessment**: Similar to real-time and automated decision and action, data
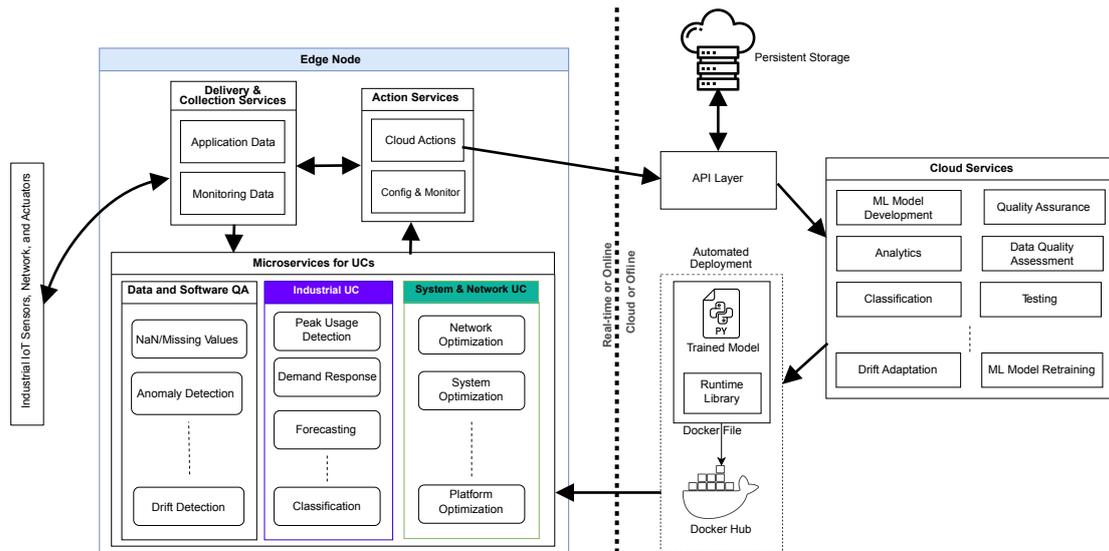
Figure 3: An architectural view of the proposed MLOps utilizing IoT, edge, and cloud for continuous ML software and QA for ML model training, testing, deployment, and real-time analytics and action.

flows from the network to microservice delivery and collection to microservices for UCs and Action Services. The microservices for UCs here are containers that evaluate the quality of the incoming data. Furthermore, the quality of the containerized ML software for model degradation. Further, the action signal from the action microservice is sent to the Cloud API layer for the execution of the appropriate cloud service(s).

- **Message forwarding for cloud services**: In this scenario, for long-term storage, the message or data are forwarded from the network to the delivery and collection microservice to the cloud API layer for persistent storage.

4. Future directions

The modular QA and the MLOps architecture collectively enable real-time execution of real-time ML and action supported by data and SQA processes. However, research in this area is still in its infancy, and we have identified the following vectors for future direction:

- **Software testing:** The principles of Chaos Engineering state that systems react differently based on surroundings and traffic patterns. This is true for software, and automated software engineering and software testing procedures, such as shadow testing, allow the software to be analyzed in terms of how it performs in different environments. The log data of the software in various environments

is then analyzed. The use of SQA opens the possibility of potential research to automate shadow testing and other automated software tests. An industrial application where automated software testing with QA is useful is in compressed air. Compressing air is an energy-intensive process that is used in industries to remove dust and cooling [19]. In addition to wasting air/gas and increasing operational expenses, leaks can also be a point of entry for contaminants to enter the system. Therefore, the process must be continuously monitored for leaks to reduce overall energy and operational cost. Streaming data in compressed air are often subject to anomaly and noisy measurements. Monitoring and predicting future air pressure readings based on historical data is one such UC.

- **Automated regularization, weights, and parameter estimation of ML software:** ML models in ML software often need to be fine-tuned with regularization, weights, and other parameters to obtain a more accurate convergence. Furthermore, ML models can get caught in local minima, and one of the best practices is to train multiple times to use the trained model with the optimum performance. In a machine learning software configuration with continuous training and automatic deployment, it is not always possible to perform this repeated training and fine tuning. In modern implementations, systematic parameter search looks for the best match among a prede-

fined set of parameters by the user. Thus, there is potential for automated and principled parameter optimization.

- **Automated ML software container fault detection:** Failures can occur in containerized ML software running on edge and fog nodes. One of these flaws is that containers can suddenly shut down in the middle of a task. This can be caused by a variety of reasons, including an external attack. Currently, platforms such as Kubernetes provide a minimal container monitoring configuration based on the input of a network administrator [20]. With the QA for ML, it is possible to implement automatic monitoring to discover and classify faults.

- **Drift detection and adaptation:** Streaming data is often subject to data and concept drift. The standard practice in this case is to retrain an ML software periodically or whenever a drift is found. An adaptation technique that retrains and redistributes weights on the fly without having to produce a new software version is a future direction in this research. An example of this is in the utilities sector with electricity data from smart meters. To manage electricity supply and demand, data from smart meters from individual and business users are used to estimate load forecasting and peak usage. When users install their solar panels or buy an electric vehicle, their normal pattern of electricity demand changes and drift occurs. As a result, with the help of QA for ML, drift adaptation can be implemented.

Acknowledgments

References

[1] S. Alla, S. K. Adari, What Is MLOps?, Apress, Berkeley, CA, 2021, pp. 79–124.

[2] Y. Dang, Q. Lin, P. Huang, Aiops: Real-world challenges and research innovations, in: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 2019, pp. 4–5.

[3] Build mlops workflows with amazon sagemaker projects, gitlab, and gitlab pipelines, https://aws.amazon.com/blogs/machine-learning/build-mlops-workflows-with-amazon-sagemaker-projects-gitlab-and-gitlab-pipelines/, 2022.

[4] Mlops for python with azure machine learning - azure reference architectures | microsoft docs, https://docs.microsoft.com/en-gb/azure/architecture/reference-architectures/ai/mlops-python, 2022.

[5] S. Shams Amiri, S. Mottahedi, E. R. Lee, S. Hoque, Peeking inside the black-box: Explainable machine learning applied to household transportation energy consumption, Computers, Environment and Urban Systems 88 (2021) 101647.

[6] M. Hesenius, N. Schwenzfeier, O. Meyer, W. Koop, V. Gruhn, Towards a software engineering process for developing data-driven applications, in: 2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2019, pp. 35–41.

[7] Y. W. Lee, D. M. Strong, Knowing-why about data processes and data quality, Journal of Management Information Systems 20 (2003) 13–39.

[8] W. Y. Wong, T. H. Sam, C. W. Too, W. F. Pok, Software quality assurance plan: Setting quality assurance checkpoints within the project life cycle and system development life cycle, in: 2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA), IEEE, 2022, pp. 214–219.

[9] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, D. Zufferey, Paracosm: A test framework for autonomous driving simulations, in: International Conference on Fundamental Approaches to Software Engineering, Springer, Cham, 2021, pp. 172–195.

[10] A. Panichella, C. C. S. Liem, What are we really testing in mutation testing for machine learning? a critical reflection, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), 2021, pp. 66–70.

[11] S. K. Balan, Data Integrity and Quality, IntechOpen, Rijeka, 2021.

[12] D. Kreuzberger, N. Kühl, S. Hirschl, Machine learning operations (mlops): Overview, definition, and architecture, 2022.

[13] B. W. Ford, Z. Zong, A cost effective framework for analyzing cross-platform software energy efficiency, Sustainable Computing: Informatics and Systems 35 (2022) 100661.

[14] F. Bayram, B. S. Ahmed, A. Kassler, From concept drift to model degradation: An overview on performance-aware drift detectors, Knowledge-Based Systems 245 (2022) 108632.

[15] M. Carugi, Z. Lou, Future industrial networks: Requirements, challenges, research and standardization needs, in: 2021 ITU Kaleidoscope: Connecting

Physical and Virtual Worlds (ITU K), 2021, pp. 1–9.

[16] J. B. Boštjan Arh, Bojan Podgornik, Electroslag remelting: A process overview, Materiali in tehnologije 50 (2016) 971–979.

[17] A. Chatterjee, B. S. Ahmed, E. Hallin, A. Engman, Testing of machine learning models with limited samples: An industrial vacuum pumping application, arXiv preprint arXiv:2208.04062 (2022).

[18] J. M. Zhang, M. Harman, L. Ma, Y. Liu, Machine learning testing: Survey, landscapes and horizons, IEEE Transactions on Software Engineering 48 (2020) 1–37.

[19] M. Benedetti, F. Bonfa, I. Bertini, V. Introna, S. Ubertini, Explorative study on compressed air systems' energy efficiency in production and use: First steps towards the creation of a benchmarking system for large and energy-intensive industrial firms, Applied energy 227 (2018) 436–448.

[20] Tools for monitoring resources, https://kubernetes.io/docs/tasks/debug/debug-cluster/resource-usage-monitoring/, 2022.