White Paper
Data Center
June 2022

IT@Intel: Autonomous Quality in AI Model Productization: A Journey

Enable efficient quality gates of the AI model lifecycle by moving toward autonomous quality in AI model productization

Author
Ziv Shapira
QA Architect IT-AI, Intel IT

Table of Contents

Executive Summary

Intel IT's artificial intelligence (AI) group works across Intel to transform critical work, optimize processes, eliminate scalability bottlenecks and generate significant business value (more than USD 1.5B return on investment in 2021). Our efforts unlock the power of data to make Intel's business processes smarter, faster and more innovative, from product design to manufacturing to sales and pricing.

Our machine learning operations (MLOps) methodology requires embedding quality-control principles into the heart of the AI model productization process to ensure the scalability of our industrialization of the AI model production pipeline. Our principles provide many advantages, including the ability to achieve the following:

- Apply quality principles to different projects by abstracting the quality-control strategy elements.
- Quickly implement specific quality controls with reusable building blocks and shared components (this task used to take weeks).
- Minimize the cost and effort required to maintain the hundreds of AI models in production by adhering to systematic quality-control metrics.

Our dedication to improve autonomous quality in AI model productization led us to develop Microraptor, a set of MLOps capabilities that enable this operation at scale. MLOps is the practice of efficiently developing, testing, deploying and maintaining ML in production.

Contributor

Moty Fania, Principal Engineer, CTO and Director of Machine Learning Engineering, Intel IT

Acronyms

AI	artificial intelligence
BDD	behavior-driven development
CI/CD	continuous integration/continuous delivery
ML	machine learning
MLOps	machine learning operations
QA	quality assurance

Background

Intel IT AI is a group of over 200 data scientists, machine learning (ML) engineers and AI product experts. Our work covers a wide span of Intel's core verticals to deliver AI solutions that optimize operations and enable scalability. We provide high business impact and transform Intel's inner processes with AI, including engineering, manufacturing, hardware validation, sales and performance. In the past decade, we have deployed 500 AI models, including more than 100 ML solutions in the last year. Our tolerance to model quality degradation is extremely low because our solutions have evolved to become intrinsic, integral parts of Intel's business-critical activities. Our dedication to reducing model quality degradation applies to releasing products to production and to ongoing execution in production.

Embedding and enabling AI across Intel's vital business operations allowed us to deliver over USD 1.56B in value during 2021. Intel IT AI has won the Intel Achievement Award for five successive years. We attained many advantages by integrating quality in the pipeline. Here are a few examples:

- Efficiently added end-to-end and model-level **automated testing** across the delivery pipeline.
- Collected numerous **quality metrics and indicators** to start applying real-time alerts on model quality degradation.
- Optimized **AI product team processes** to help define new projects and monitor their impact daily using dashboards and pipelines.

Intel is already reaping significant business value from AI throughout the entire enterprise. Our team has learned that to scale AI effectively, we cannot view projects on an individual level. To truly achieve AI model quality, we must thoroughly understand the AI quality lifecycle and solve the challenges of maintaining the quality of AI models at scale, from experiments through productization and the maintenance phase (post-deployment to production).

AI Model Quality Lifecycle

The **quality lifecycle** of an AI model consists of several steps (see figure 1):

- **Experimental Stage** is used to develop a high-quality model that addresses specific business requirements and uses high-quality data to prove its successful strategy for solving a specific business problem. Easily compare different models' results and performance parameters.
- **Productization** of the model using a robust continuous integration/continuous delivery (CI/CD) pipeline containing unit tests, integration tests, model tests, end-to-end tests and validation of quality indicators.
- **Quality Monitoring and Automated Actuations** (such as retrain) should be followed when the model is deployed and is in the maintenance phase, to help prevent model degradation and potential harm to the business.

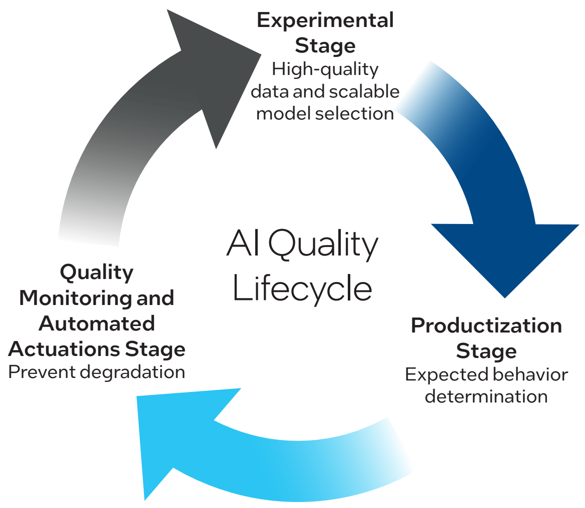
Experimental Stage
High-quality data and scalable model selection
AI Quality Lifecycle
Productization Stage
Expected behavior determination
Quality Monitoring and Automated Actuations Stage
Prevent degradation

Figure 1. The AI Quality Lifecycle includes three stages for maintaining quality: Experimental, Productization and Quality Monitoring and Automated Actuations.

While maintaining quality for a few models is easy, once scale comes into play, the challenge grows significantly due to modifications in AI models that involve more than just a code change.[1] Our primary concern is that maintaining the quality along the lifecycle becomes so time-consuming to the point that eventually our teams will not be able to address any new projects. Embedding industry-standard quality-control principles and tools into our AI pipeline allows quality to become ubiquitous and near-autonomous for the AI model across development, deployment and execution.

We aim to shorten the experiment phase and productize models in minutes by using full automation, quality gates and adapting CI/CD strategies common in the software development domain. Our approach to maintaining model quality combines innovative research and development along with MLOps to implement and integrate AI model quality into all the delivery pipelines.

AI Models Lifecycle: Quality Challenges

Maintaining the quality of AI-based software is highly challenging. It requires substantial investments in time, effort and specialized quality assurance (QA) teams. In many cases, a model may never make it to production; over 87% of AI projects stay on the shelf.[2]

Adopting a specific quality approach for each AI-based software product is convenient, fast and results in a quick ad hoc win. Specialized open-source building blocks and dedicated wrappers provide a safety net of tests, often constructed from "frozen" datasets, specific test cases and periodic checks on the input and output data. However, productizing the test suite can take up to a week, adding to the model productization time because it is firmly coupled with the implementation.

Our years of experience with AI products and the quality of traditional software solutions provided us with insights into why the above approach is not scalable when building an AI software products factory. Some of the reasons are:

- **Lack of standardization**. A failure to outline how the product should perform can lead to an inefficient approach in defining and executing the quality-control strategy in specific cases.
- **Limited or no reuse**. The one-project-at-a-time approach to quality creates redundancies instead of promoting code reuse. Different applications may need a similar quality-related code that potentially could have been reused.
- **High cost to maintain quality-related tests and datasets**. Managing the quality codebase with hundreds of unmanaged AI applications spread across the enterprise makes it challenging to monitor and track the health of many models when quality is not standardized. Tracking how to maintain each model is also a costly burden.
- **Explainability and debugging**. These actions become difficult because model behavior can change over time, based on retraining or self-improvement heuristics.
- **Lack of high-quality data, labeled data or any data can hinder model quality**. AI models require high-quality data for testing and validation before a project goes to production. Frequently, the existing data at the start and during project development is either non-existent, very scarce, low-quality (due to the processes that create it) or missing labels for the model to use.
- **Environment replication can be difficult**. Traditional software can easily replicate an environment for testing purposes (a server or Kubernetes cluster, for example). The task is much more difficult for AI products since the environment types (such as clusters or a set of IoT sensors, or hardware and firmware), configuration (such as amounts of data hyperparameters or bandwidth) and setup cost (security and privacy) hinder the scale needed for testing.

Our AI Model Quality Vision and Principles

Connecting our quality-control principles to our vision for autonomous AI model quality in manufacturing is essential to achieve our goals.

Vision

Our ongoing vision, partly led by the elements described in this paper, is about achieving autonomous software quality. For us, autonomous quality means the ability of an AI solution or software solution to organically and independently understand the quality state of its functions and respond to that state without any human intervention. Such behavioral changes can be accomplished by retraining the model to recognize gradual degradation, stopping recommendations by identifying the imminent negative impact on the customer, retuning hyperparameters due to changing data detection, or changing the heuristics.

A central part of the vision includes the following guidelines:

- Apply **behavior-driven development (BDD)** to our models and software so the expected behavior is clear to the entire team. BDD is an established industry standard for defining behavior and describing the acceptance use cases.
- Ensure the model or application in production continuously transmits **data or metrics**, so indicators and alerts are applied based on rules or AI (model degradation).
- Use **bugs analysis and continuous production monitoring** to apply the correct autonomous actuations, thus increasing the quality autonomously at a large scale (projects, models, data and configurations).

Principles

Our principles for software quality in general and their application to AI-based software development are derived from over ten years of experience in the domain. Here is a summary of the guiding principles that are the backbone of the vision:

- Enable quality to be easily embedded as part of the model or solution code.
- Support scale of handling quality issues, ideally without human intervention.
- Base decisions and projects on high-quality data.
- Focus on bringing high value through quality investment: customer satisfaction, reliability, efficiency and agile feedback loops.
- Support efficient, highly automated work within domains (product AI, data scientists and ML engineers) and across domains.

Intel IT MLOps Solution: Model Quality Aspects

For years, Intel IT has been using top industry standards for quality and testing to streamline project and software delivery with added automated quality gates. To avoid issues associated with the custom approach to AI quality and testing, we applied our learnings and infrastructure to enable and improve machine learning operations (MLOps), specifically by adding tests, indicators and datasets along the pipeline.

We built an AI productization platform for each business domain we work with, such as manufacturing or sales, to enable MLOps. Models and AI services are delivered, deployed, maintained and managed on top of the AI platforms with the ability to add several quality gates along the way.

Our ML engineers and data scientists construct the AI workflows and infrastructure required to productize AI models and ensure their quality. Their central duties are the following:

- Add datasets for AI testing at scale, using extraction of production data or generated datasets.
- Implement testing interfaces for AI models and other engineering components (data transformers, output).
- Facilitate MLOps for CI/CD and test automation.
- Build and sustain AI models through dashboards and automated actuations.

Common Quality Aspects of All AI Platforms

Every business domain's AI platform works with several types of data, various data extraction processes and different integration points into the applicable business workflow. And yet, our AI platforms encourage reuse because they share multiple quality objectives:

- **Behavior** of the application and the models needs to be continuously validated.
- **Data and hyperparameters** need to be continuously monitored since they impact the AI-based software, even if the code has not changed.
- **Enable CI/CD** for AI-based software code and its quality.
- Achieve **separation of concerns** and mandate fewer hand-offs or rework between data scientists, product experts and ML engineers regarding the implementation of quality gates.
- Enable **maintainability and quality** of the models from deployment to production.

Microraptor is a platform of MLOps capabilities that are reused in all our AI platforms. To fulfill our standard requirements for quality in Microraptor, we included the capability to track the AI model's quality-control metrics using a built-in *model quality indicator system*. In addition, we enabled model testing using unit tests and datasets prepared by the data scientists and end-to-end tests using data prepared by the product analysts and ML engineers.

Tracking Model Quality-Control Metrics

Once AI models deploy to production, quality and performance often degrade over time. Predictions become less precise and degradation may harm the business processes that rely on those predictions. Therefore, it is critical to track models in production, monitor their health, and respond to problems as they arise.

We incorporated a subsystem in Microraptor that enables the development, deployment and management of all model quality-control metrics. We use it for alerting and actuation to trigger an automatic retrain or, if necessary, human attention.

We designed the subsystem (see Figure 2) to permit the development of indicators using only Python skills, allowing data scientists to develop indicators and quality-control metrics independently without the assistance of ML engineers.

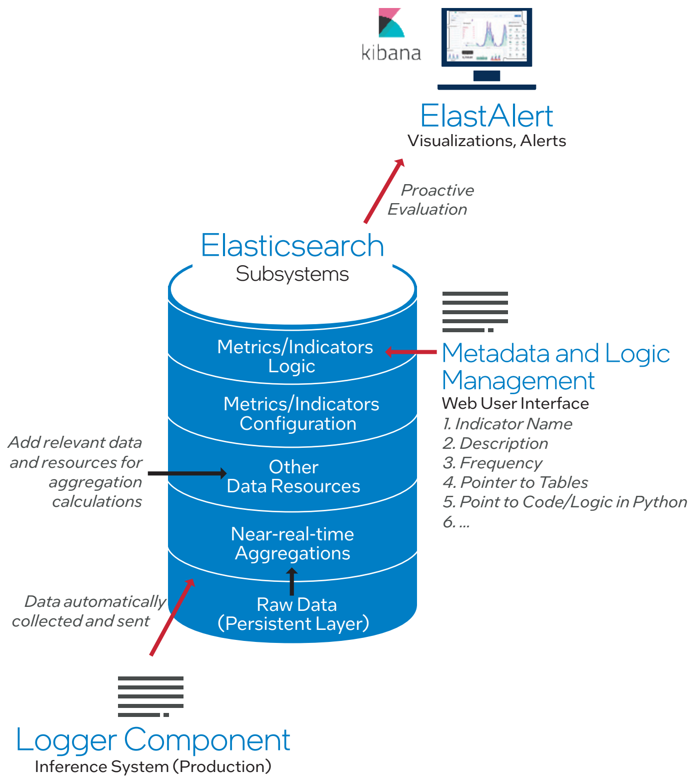
kibana
ElastAlert
Visualizations, Alerts
Proactive Evaluation
Elasticsearch
Subsystems
Metrics/Indicators Logic
Metrics/Indicators Configuration
Other Data Resources
Near-real-time Aggregations
Raw Data (Persistent Layer)
Metadata and Logic Management
Web User Interface
1. Indicator Name
2. Description
3. Frequency
4. Pointer to Tables
5. Point to Code/Logic in Python
6. ...
Add relevant data and resources for aggregation calculations
Data automatically collected and sent
Logger Component
Inference System (Production)

Figure 2. Microraptor includes subsystems to develop, deploy and manage model-quality metrics. Microraptor streamlines the process of developing indicators and quality-control metrics independently of ML engineers.

To accomplish this objective, Microraptor provides these layers:

- A **logger** component records inputs and outputs for each inference request. It automatically collects raw data and sends it to Elasticsearch.
- In addition to logging data, the system computes **near-real-time aggregations** that summarize data. These summaries contain counts, averages, standard deviations and additional data.
- It is easy to **add relevant data and sources** that could be required for the aggregation computations. This data might include information like statistical distribution, labels and training data metadata.
- Data scientists can use custom logic and rules to calculate the indicators using Python. The subsystem provides easy access to the data generated by previous layers stored in Elasticsearch.
- User feedback (prediction labels) can be loaded into the indicators system and be **matched with the prediction results** to assess the model quality based on actual feedback.

After the data from the layers is available in Elasticsearch, we use it to proactively evaluate the health of all models. Kibana is an open user interface used to visualize the indicators and quality-control metrics stored in Elasticsearch. Microraptor uses ElastAlert to define rules and more sophisticated Python logic for calculating metrics and triggering the required actuation.

In addition to the capability to create indicators, we included industry-standard model metrics in Microraptor.

The following are available out-of-the-box:

- **Identify concept drift**. Detect shifts in the distribution of production elements.
- **Assess numeric stability**. Examine the model's prediction stability and classification classes frequency stability.
- **Skew monitoring**. For specific AI model types—like ensembles, A/B and graphs—monitor several models with production inputs and compare metrics to analyze the stability of each one.
- **Monitor predictions versus labels**. These metrics, also known as offline proxy metrics, are used for models where true labels are known. Some metric examples in this category include normalized discounted cumulative gain (NDCG), log loss, square error and confusion metrics.

Product Ops

Product Ops is an emerging trend that supports a consolidated set of practices and tools for organizations, such as our AI team, where all the teams have a common core. In our case, the processes of the IT AI product teams have a commonality for looking for signals in customers' data and analyzing production to either find model degradation or to show a positive impact of our solution on Intel's IT AI customers' business.

Our infrastructure, best practices and shared code form the foundation for product ops within the product teams. Therefore, it is a natural step to apply the holistic quality for AI model productization strategy to the unique role of Product Ops within the team and produces AI-based solutions.

The layers of our Product Ops infrastructure include:

- Personal on-demand environments to implement the analysis pipeline, based on Jupyter Notebooks or DSRaptor (an enhanced pipeline technology targeted to accelerate model development and improve productivity and quality).
- Scalability of memory, computing and storage resources.
- Inner-source reusable components and Python libraries for fast ramp-up.
- A dashboarding service based on Streamlit for exposing results to stakeholders.
- Collaboration with developers and data scientists using GitHub and DSRaptor.
- Persistence layer of analysis results for comparison over time.

Our product teams' reusable code, independence from relying on data scientists and developers, and Python skills contribute to the efficiency and speed of the product teams to automate their processes and increase quality by eliminating waste. Using industry-standard tools and dashboarding capabilities smooths the collaboration between team roles through code and real-time dashboards.

Data Quality (Including Big Data)

Maintaining data quality is achieved using a set of metrics and supporting tools that enable a project-level and organization-level code to validate a dataset. This especially applies in autonomous quality in AI model productization where challenges are amplified for big data and AI projects. Validation occurs before either an AI-based application consumes the dataset or when a team member reviews and interprets the dataset's quality. Infrastructure, best practices and shared code serve as the basis for data quality within our teams.

Data quality comes in various stages along the model pipeline, from exploration to post-production:

- Initial datasets from customers. Evaluate the project.
- Processing datasets (cleanup, enrichment and value replacement, for example). This requires revalidating the data quality.
- Input dataset to the model. A quality gate is required, because the team that develops data preparation code is often different than the data scientists who develop the model code, where data quality is expected to be high.
- Splitting data to train and test. Validate the two datasets in terms of the expected quality.

We construct our data quality infrastructure layers to support the transition from data profiling and understanding to mature indicators that continuously become part of the post-production quality gatekeepers. This infrastructure is under active development, based on industry-leading solutions and internal enhancements to support visibility scale and automation.

Overview of the Open-Source Technologies

As illustrated in Figure 3, the quality indicators component uses several open-source technologies to enable the full MLOps quality gates while abstracting the complexity of these technologies from the user. Users do not have to know anything about Kubernetes, Helm, Jenkins, or Elasticsearch. They can focus on discovering or building the best AI model and validating its quality.

After the model is ready, a data scientist registers the model to MLflow (an open-source platform for managing the end-to-end AI lifecycle) while adhering to basic coding standards. Other tasks—from building to testing to deploying—occur automatically. The model deploys while all the quality gates execute pre- and post-production.

Here are some of the additional open-source products we started adopting specifically for enhancing the quality indicators component of Microraptor:

- **Data quality library profiling and understanding.** Integrating tools such as great_expectations and pandas_profiling.
- **Visualizations.** Users can visualize data within an inner-source Python library that interfaces with visualization standard libraries like Plotly, Bokeh and Plotly go.
- **Dashboarding.** Streamlit is an open-source framework that generates web applications and dashboards with functionality—such as filtering from Notebook or Python code—that use the Streamlit API.

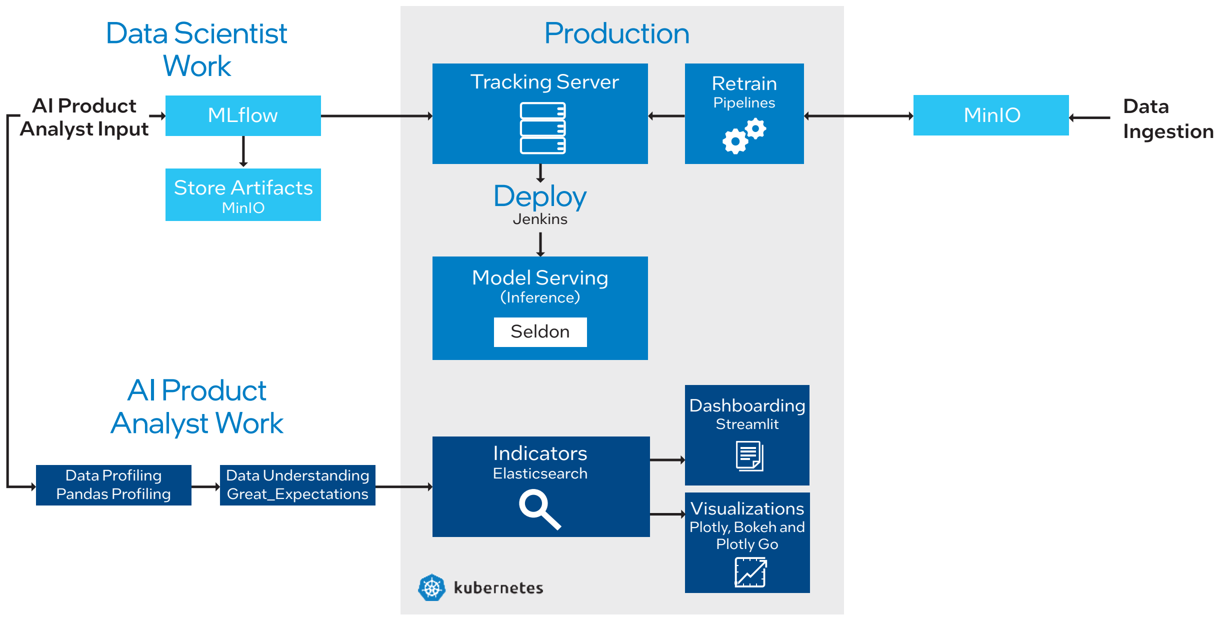

Figure 3. The Microraptor build is comprised of multiple, customized open-source resources. Open-source technologies simplify the AI development and deployment stages for the user with embedded autonomous quality using the indicators.

Results

Our AI platforms, equipped with advanced, reusable MLOps capabilities, prove highly successful in automating and accelerating the quality development and maintenance of AI models through their lifecycle.

By applying the following IT-AI principles for software quality we saw the following results:

- **Enable quality to be easily embedded as part of the model/solution code.** In many cases, the data scientist can achieve production quality monitoring independently, eliminating rework and unnecessary hand-offs by enhancing tests and test data. These new capabilities are now actively used in various AI projects in the Sales AI platform, manufacturing AI product development and other domains.
- **Support scale of handling quality issues, ideally without human intervention.** It used to take up to several weeks to test a single model. Most deployed models are now subject to proactive monitoring using Microraptor's indicator system to track model health and respond proactively and sometimes autonomously to a model's degradation and other quality items tracked by automated quality indicators.
- **Base decisions and projects on high-quality data.** By applying quality indicators to datasets, whether they are explored for signals or delivered in production, teams ensure that data retains the same high quality as initially expected, and as defined using the indicators.
- **Support efficient, highly automated work within domains (product AI, data scientists and ML engineers) and across domains.** Some product teams have transformed their work from offline and static communication with their stakeholders to online dynamic pipelines and dashboards, which are scalable in terms of compute and availability without further work through self-updating dashboards.
- **Focus on bringing high value through quality investment with customer satisfaction, reliability, efficiency and agile feedback loops.** Minimal to no intervention is required by product teams, because stakeholders can access a dashboard that reflects the data and insights at any time.

Conclusion

Data is a transformational force, and we use AI to propel product innovation and improve Intel's business process execution. Our MLOps solution has enabled Intel IT to develop, deploy and support hundreds of AI model quality gates with low cost and effort.

We have significantly automated model quality components along the model lifecycle. Autonomous quality gates allow most of our AI group to concentrate on developing new AI solutions, while only ten percent of our total resources are devoted to model maintenance and quality updates.

We continue to refine our strategy and platform applications to materialize our vision for autonomous quality in AI model productization. As we implement, we plan to continue working with Intel's business units and design teams to implement our principles and put automation, AI, and data to work to support Intel's quality to steadily expand to support our scale in business.

Related Content

If you liked this paper, you may also be interested in these related stories:

- Building an AI Center of Excellence blog
- Push-button Productization of AI Models white paper

For more information on Intel IT best practices, visit **intel.com/IT**.

[1] Research Gate, "Quality Assurance for AI-based Systems: Overview and Challenges," https://www.researchgate.net/publication/349195521_Quality_Assurance_for_AI-based_Systems_Overview_and_Challenges

[2] Intel IT, "Push-Button Productization of AI Models," intel.com/content/www/us/en/it-management/intel-it-best-practices/push-button-productization-of-ai-models-paper.html