QualAI: Continuous Quality Improvement of AI-based Systems*

Nicole Novielli[1], Rocco Oliveto[2], Fabio Palomba[3], Fabio Calefato[1], Giuseppe Colavito[1], Vincenzo De Martino[3], Antonio Della Porta[3], Giammaria Giordano[3], Emanuela Guglielmi[2], Filippo Lanubile[1], Luigi Quaranta[1], Gilberto Recupito[3], Simone Scalabrino[2], Angelica Spina[2] and Antonio Vitale[2]

[1]*University of Bari, Italy*
[2]*University of Molise, Italy*
[3]*University of Salerno, Italy*

Abstract

QualAI is a two-year project that aims to define a set of recommenders to continuously monitor, assess, and improve the quality of AI-based systems, with a particular focus on ML-based systems. Quality assurance will be guaranteed from different perspectives and during both the development and operations phases. We will define recommenders for the quality assurance of both data and ML models to enable practitioners to mitigate technical debt. Emphasis will be given to communication issues that could arise in hybrid teams including data scientists and software developers. In this paper, we present the project outline, provide an executive summary of the research activities, and present the expected project results.

Keywords

Software Engineering, Machine Learning, Quality Assurance, Recommender Systems

1. Introduction and Motivation

In 2020, Google Health released an extremely accurate AI software for identifying diabetic retinopathy in pictures of patients' eyes. The classifier achieved over 90% accuracy and provided a diagnosis in less than 10 minutes. Unfortunately, when deployed for use in hospitals, the AI-based classifier experienced a drop in performance compared to the lab setting. Also, the system often failed to provide an outcome: being trained with high-resolution pictures, it discarded over one-fifth of images due to their low quality. This caused delays of up to months to obtain a diagnosis, resulting in complaints from patients [1]. This accident shows how assessing performance in the lab might not be enough to ensure the quality of AI-based systems, as the success of a machine learning (ML) model does not consist exclusively of its accuracy. Special attention should be devoted to users' needs and context of action as well as to the integration of ML models with non-ML software as part of a large AI-based system,

Joint Proceedings of RCIS 2024 Workshops and Research Projects Track, May 14-17, 2024, Guimarães, Portugal
nicole.novielli@uniba.it (N. Novielli); rocco.oliveto@unimol.it (R. Oliveto); fpalomba@unisa.it (F. Palomba); fabio.calefato@uniba.it (F. Calefato); giuseppe.colavito@uniba.it (G. Colavito); vdemartino@unisa.it (V. D. Martino); adellaporta@unisa.it (A. D. Porta); giagiordano@unisa.it (G. Giordano); emanuela.guglielmi@unimol.it (E. Guglielmi); filippo.lanubile@uniba.it (F. Lanubile); luigi.quaranta@uniba.it (L. Quaranta); grecupito@unisa.it (G. Recupito); simone.scalabrino@unimol.it (S. Scalabrino); a.spina5@studenti.unimol.it (A. Spina); a.vitale8@studenti.unimol.it (A. Vitale)

In a typical software system, given the requirements, the behavior is always specified by the developers. In an ML-based system, instead, data scientists define the operationalization of constructs playing a role in the addressed problem. Also, they build a training set, and identify the envisaged ML technique , which then defines the system behavior (ML models). Such systems require maintenance and quality assurance like any other system but special attention should be devoted to the typical issues affecting the quality of data and ML models [2]. As such, assessing and improving the quality of ML-based systems presents unique challenges involving different aspects, which we discuss in the following. First, quality issues can be found in the ML *models* that the system uses to build it or its parameters, as well as in the *data* used for training them. For example, the historical data once used to train the ML model cannot be used blindly because they may become outdated and no longer reflect the status quo, due to a concept drift that might be occurring. Second, communication issues might arise as the teams working on ML-based systems are intrinsically heterogeneous. Several peculiar quality issues may arise, related, for example, to team *communication* and technological gaps, e.g., data scientists and software developers may use incompatible technologies. Finally, further issues might arise at the level of *deployment* and *operations*. The automated build process of some modules of an ML-based system and the construction of container images often require training one or more ML models. Specific quality issues may occur in this phase.

In essence, developers and data scientists are now confronted with the challenge of being more agile and adaptive. More specifically, new methods and strategies are needed for keeping ML-based systems responsive, monitored, and dependent on reliable variables. **MLOps**[1] is an ML engineering culture and practice that aims at dealing with the above challenges. MLOps unifies the ML system development (Dev) and ML system operation (Ops) advocating for automation and monitoring at all steps of ML system construction, including integration, testing, releasing, deployment, and infrastructure management, thus representing an umbrella for best practices and guiding principles around machine learning.

The above considerations motivate this project proposal. QualAI aims to define a set of recommenders that can be used to continuously monitor, assess, and improve the quality of AI-based systems, with a particular focus on ML-based systems. Quality assurance will be guaranteed from different perspectives and during both the development and operations phases. We will define recommenders for the quality assurance of both data and ML models. Results will allow practitioners to mitigate technical debt [2]. Emphasis will be given to communication issues that could arise between data scientists and software developers. Finally, we will define approaches to (i) identify quality issues in the CI/CD pipeline; and (ii) monitor the quality of the system during the operations phase. QualAI will, both, facilitate the analysis of the recommendations (thanks to their explainability) and the planning of the corrective operations suggested by QualAI (thanks to the cost-effective analysis). A web platform integrating the recommenders for assessing the quality of ML-based systems will be released to produce quality badges summarizing the quality of a given AI-based system.

QualAI is a two-year project that has been funded in July 2023 by the European Union - NextGenerationEU through the PRIN 2022 call for projects of the Italian Ministry of University and Research for projects. In the following we provide a description of the research goals and

[1]MLOps, https://ml-ops.org, 2020

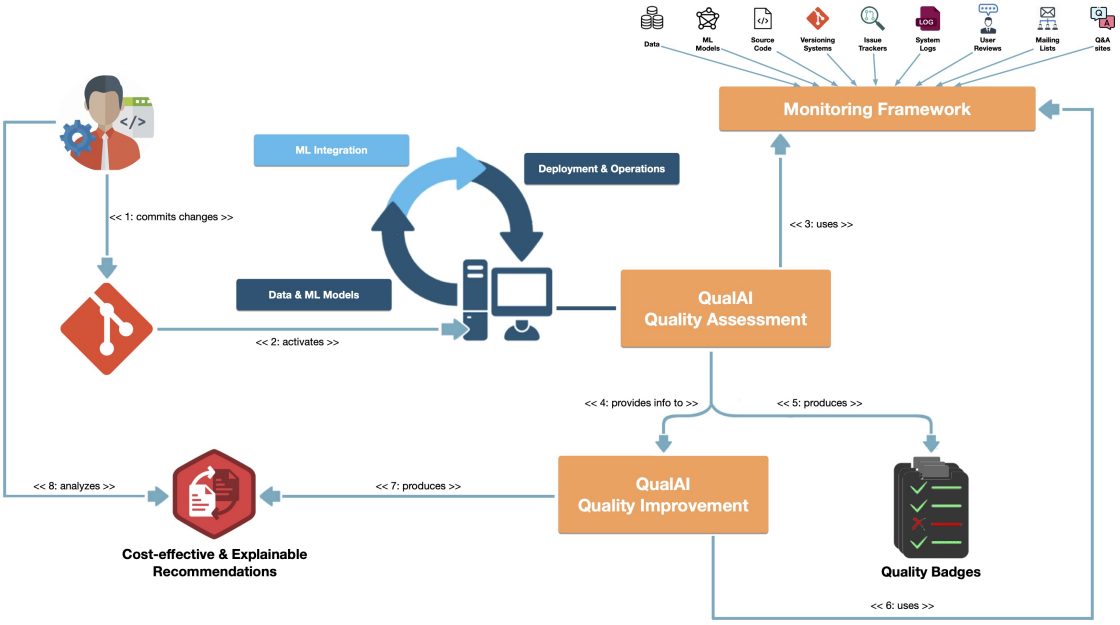

Figure 1: The Workflow of the QualAI framework.

an executive summary, by also positioning this research in the frame of related work.

2. Goals and Expected Results

Project Goal and Final Outcome The goal of QualAI is to define a set of recommenders to improve the quality of AI-based systems, in general, and of ML-systems, in particular, from different perspectives: data and ML models, ML integration, and deployment and operations. This goal will be achieved by monitoring the quality of the ML-based system during its whole life-cycle aiming at collecting useful information to automatically assess its level of quality. Once a quality issue, i.e., technical debt, has been identified, corrective operations will be suggested to remove the technical debt and improve the overall quality of the ML-based system.

All the QualAI recommendations will have a cost-effective and explainable connotation. They will be designed to rank the identified issues or the identified corrective operations based on the ratio between the potential costs that developers should spend to address the issue, e.g., change the ML model, and the potential benefits that their removal might provide to the overall quality of the ML-based system. Also, each recommendation is enriched with a human-readable explanation, in textual or visual form, see for instance Bellini et al. [3], providing the rationale behind the identified issues or the identified corrective actions. Such properties will increase the practitioners' confidence in the recommendations received and make informed decisions. The QualAI recommenders can be properly designed to be easily integrated into a Continuous Integration/Continuous Deployment (CI/CD) pipeline aiming at continuously improving the quality of ML-based systems.

The *final outcome* of the QualAI projects is represented by a set of recommenders able to

assess and improve the quality of ML-based systems. Figure 1 shows the overall workflow of QualAI. The recommenders composing QualAI are activated when a developer or data scientist commits a change to the ML-based system. Then, QualAI analyzes the quality of the new version of the ML-based system from different perspectives: data and ML models, ML integration, and deployment and operations. The pipeline of QualAI recommenders can be easily integrated into the original CI/CD pipeline of the ML-based system to allow continuous quality assurance. In this respect, QualAI also provides specific recommenders to optimize and improve the CI/CD pipeline. The quality assessment is performed by the *Quality Assessment component* of QualAI. At the end of the analysis, the QualAI Quality Assessment component provides as output a set of quality badges (one for each quality dimension analyzed) that summarize the quality of the system and emphasize specific quality issues. These badges could be used by a project manager to simply analyze the quality of the system or as a support to certify that the ML system has a certain level of quality. The QualAI Quality Assessment component also provides information (e.g., the quality problem identified and its location) to the Quality Improvement component of QualAI. The *Quality Improvement component* is in charge to identify corrective actions (i.e., refactoring operations) aiming at removing the identified issues and thus improving the overall quality of the system. Each recommendation is accompanied by a description in a human-comprehensible format as well as an analysis of the cost-benefits for each proposed operation. Such analyses will facilitate the planning and the schedule of the proposed operations (e.g., the software analyst could decide to focus the attention on the most critical issues and postpone the others).

Both the Quality Assessment and Quality Improvement components rely on the monitoring framework of QualAI, i.e., the shared knowledge base which all the recommenders are based on. Such a knowledge base is continuously and automatically updated and contains resources *internal* (e.g., source code, ML models, training data, issues, logs, mailing lists, user reviews) to the ML system under analysis and *external* to the system (e.g., source code and related artifacts of other ML-based software projects, question and answer sites).

The accuracy of the QualAI recommenders will be empirically evaluated. We plan to conduct mixed-method research that combines (1) the mining of data science projects, which aims at establishing the accuracy of the recommenders; and (2) survey- and interview-based studies with developers to get feedback on the effectiveness of the proposed recommenders. In the context of the study, we will define guidelines for conducting such empirical studies and for creating and sharing replication packages. We also plan to apply the QualAI recommenders on a set of industrial software systems and involve practitioners by exploiting the collaboration with our industrial partners.

Objectives and Expected Results To the overall goal of our research project, we will address the following objectives.

- *OB1: Definition of a monitoring framework for knowledge management.* As a shared preliminary objective, we will define what data sources should be considered and what formats should be used to represent the data. More specifically, in this project we will analyze developers' communication (e.g., on collaboration platforms), user feedback (e..g, through application reviews), source code and notebooks, and build logs through

continuous integration tools, application logs from monitoring tools. The commonly used ML process models will be reviewed and synthesized as a preliminary step, to ensure that the approaches defined as an outcome of the other objectives provide adequate support for most of the realistic application scenarios of ML-based systems. The expected result of this activity is a common framework that will be used in all the following phases.

- *OB2: Definition of approaches for assessing and improving the quality of data and ML models.* We will consider the causes leading to the degradation of several properties of ML systems, including robustness, efficiency, privacy, interpretability, fairness, and reproducibility. As a result, we plan to build a comprehensive catalog of the issues affecting the above-mentioned properties as well as the mitigation strategies that can improve them. To this aim, we will define novel approaches to identify issues in the data used for training the models, in the machine learning techniques used to build them, and in their configuration, based on, both, static and dynamic analysis. In this respect, we plan to propose recommenders that balance the cost needed to address the issues identified and the associated effectiveness. Also, all the recommendations will be explainable, in an effort of facilitating the identification of more critical issues to address. The second expected result is a set of cost-effective recommendation techniques that can automatically improve data and model quality.
- *OB3: Definition of approaches for assessing and improving the quality of the integration between the underlying ML models and the rest of the system.* We will focus on several relevant aspects, including team communication, technical gap, and system security. The first expected result is a set of cost-effective techniques that can automatically detect quality issues at integration and system level. To achieve this goal, we will define novel approaches for detecting quality issues both in the integration (process-oriented) and in the resulting system (product-oriented). Such approaches will be mostly based on static analysis techniques (e.g., detection of community and code smells). Finally, novel approaches will be defined for automatically improving the quality of the integration (e.g., techniques for automatically adapting the technologies used by data scientists to production-ready code) and of the resulting system (e.g., ML-based system-specific refactoring operations). We also plan to devise approaches based on data-driven techniques, which will still follow an explainable and cost-effective philosophy. The second expected result is a set of cost-effective approaches to recommend operations for fixing the quality issues at the ML integration level.
- *OB4: Definition of approaches for assessing and improving the quality of deployment and operation of ML-based systems.* We will focus on the CI/CD philosophy and, specifically, on the configuration of the pipelines for building the final product and checking its quality. Indeed, suboptimal configurations of such pipelines may hinder the quality of the final product. We will also focus on virtualization and/or containerization and, specifically, on the composition of the images describing the execution environments of the system components. Finally, we will focus on the software log quality. The first expected result is a set of techniques that can automatically detect quality issues at the deployment and operation levels. To achieve this goal, we will define novel approaches based on static analysis techniques (e.g., detection of configuration smells for Docker files) and dynamic analysis techniques (e.g., analysis of the execution logs). The second expected result is a

set of cost-effective approaches that can recommend operations to fix the quality issues at the deployment and operation levels. Especially, new approaches will be defined for automatically improve the quality of deployment and operation.

3. State of the Art

In the following, we overview the literature on the three pillars of QualAI.

Data and ML Models Studies were conducted to describe issues affecting data and ML model quality. Sculley et al. [4, 2] identified design issues that threaten robustness, relevance, and efficiency. Recently, taxonomies and causes of bugs for deep learning applications were also developed [5, 6]. Bugs were generally related to wrong configuration of ML models, which impacts their robustness, or to misinterpretation of the ML model, leading data scientists to not understand its predictions. Zhang et al. [7] elicited open challenges in ML testing showing that the most critical issues affecting the reliability of ML systems concern their robustness, fairness, and correctness. Brun and Meliou [8] urged SE researchers to address the challenges of designing fair software. Further studies [9, 10, 11] described the challenges of reproducing computational notebooks, i.e., tools designed to make data analysis easier to document and reproduce. Recent studies proposed tools to detect anomalies or inefficiencies in datasets before feeding them into ML pipelines [12, 13]. All these studies highlight the importance of data and model quality for building successful AI systems. However, the few available studies represent a call for further research on investigating quality issues related to AI systems and defining recommenders to improve their overall quality.

ML Integration Quality assurance of ML integration is challenging due to the different backgrounds of data scientists, who build ML models, and software developers, who make the ML models available in the system [14]. Recommenders were proposed to detect such social smells that occur, for example, when communication lacks between teams working on different system components [15]. Sculley et al. [2] highlighted that cultural debt may arise when teams with different skills collaborate, and process management debt may accrue when many ML models are run in the same system, leading to problems with resource management and the model maintenance. Kim (2020) described roles and responsibilities that different stakeholders should have when debugging and testing ML models at different development stages. Zhang et al. [7] highlighted that security problems in ML systems may appear not only in the model in isolation but also in the integration with the rest of the system. Indeed, ML systems can be vulnerable to unique attacks, such as model stealing or data poisoning, which might compromise their integrity and confidentiality. The literature mostly focuses on understanding issues related to ML integration quality. Only recently, researchers have started investigating communication issues in multidisciplinary teams for AI-based software development [16]. We plan to further investigate communication challenges in the development of ML systems.

Deployment and Operations A few studies investigated how to appropriately deploy AI systems, especially concerning how to set up CI/CD pipelines. Recent research pointed out

the need for ML-specific pipelines that consider common needs, like the availability of models with good accuracy or suitable training data - thus supporting the idea of establishing quality control mechanisms for ML systems. Karlas et al. [17] defined a tool for integrating ML tools within existing CI/CD pipelines. Humbatova et al. [6]identified further issues related to model configuration, e.g., API-related issues, which call for additional tools. Cito et al. [18] analyzed common quality issues of Dockerfiles in open-source projects, while Wu et al. [19] defined a proper catalog of configuration smells for such files. No previous studies specifically addressed the problem of quality assurance for ML system containerization. The literature does not provide enough support to specialists in properly deploying ML systems. We also found no techniques for monitoring ML systems in production.

Acknowledgments

This research was funded by the European Union - NextGenerationEU through the Italian Ministry of University and Research, Projects PRIN 2022 ("QualAI: Continuous Quality Improvement of AI-based Systems", grant n. 2022B3BP5S, CUP: H53D23003510006).

References

[1] E. Beede, E. Baylor, F. Hersch, A. Iurchenko, L. Wilcox, P. Ruamviboonsuk, L. M. Vardoulakis, A human-centered evaluation of a deep learning system deployed in clinics for the detection of diabetic retinopathy, in: Proc. of the 2020 CHI Conf. on Human Factors in Computing Systems, CHI '20, ACM, 2020, p. 1–12. doi:10.1145/3313831.3376718.

[2] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, D. Dennison, Hidden technical debt in machine learning systems, in: Proc. of the 28th Int'l Conf. on Neural Information Processing Systems - Volume 2, NIPS'15, MIT Press, 2015, p. 2503–2511.

[3] V. Bellini, A. Schiavone, T. Di Noia, A. Ragone, E. Di Sciascio, Knowledge-aware autoencoders for explainable recommender systems, in: Proc. of the 3rd Workshop on Deep Learning for Recommender Systems, DLRS 2018, ACM, 2018, p. 24–31. doi:10.1145/3270323.3270327.

[4] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, Machine learning: The high interest credit card of technical debt, in: SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop), 2014.

[5] Y. Zhang, Y. Chen, S.-C. Cheung, Y. Xiong, L. Zhang, An empirical study on tensorflow program bugs, in: Proc. of the 27th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis, ISSTA 2018, ACM, 2018, p. 129–140. doi:10.1145/3213846.3213866.

[6] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, P. Tonella, Taxonomy of real faults in deep learning systems, 2020 IEEE/ACM 42nd Int'l Conf. on Software Engineering (ICSE) (2019) 1110–1121.

[7] J. M. Zhang, M. Harman, L. Ma, Y. Liu, Machine learning testing: Survey, landscapes and horizons, IEEE Trans. on Softw. Eng. 48 (2022) 1–36. doi:10.1109/TSE.2019.2962027.

[8] Y. Brun, A. Meliou, Software fairness, in: Proc. of the 2018 26th ACM Joint Meeting on

European Software Engineering Conf. and Symposium on the Foundations of Software Engineering, ESEC/FSE 2018, ACM, 2018, p. 754–759. doi:10.1145/3236024.3264838.

[9] J. F. Pimentel, L. Murta, V. Braganholo, J. Freire, Understanding and improving the quality and reproducibility of jupyter notebooks, Empirical Software Engineering 26 (2021) 65. doi:10.1007/s10664-021-09961-9.

[10] S. , I. Prasad, A. Z. Henley, A. Sarma, T. Barik, What's wrong with computational notebooks? pain points, needs, and design opportunities, in: Proc. of the 2020 CHI Conf. on Human Factors in Computing Systems, CHI '20, ACM, 2020, p. 1–12. doi:10.1145/3313831.3376729.

[11] J. Wang, T.-y. Kuo, L. Li, A. Zeller, Assessing and restoring reproducibility of jupyter notebooks, in: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering, ASE '20, ACM, 2021, p. 138–149. doi:10.1145/3324884.3416585.

[12] E. Breck, N. Polyzotis, S. Roy, S. Whang, M. Zinkevich, Data validation for machine learning, in: A. T. et al. (Ed.), Proc. of MLSys 2019, 2019.

[13] N. Hynes, D. Sculley, M. Terry, The data linter: Lightweight, automated sanity checking for ml data sets, in: NIPS MLSys Workshop, volume 1, 2017.

[14] N. Nahar, S. Zhou, G. Lewis, C. Kästner, Collaboration challenges in building ml-enabled systems: communication, documentation, engineering, and process, in: Proc. of the 44th Int'l Conf. on Software Engineering, ICSE '22, ACM, 2022, p. 413–425. doi:10.1145/3510003.3510209.

[15] D. A. Tamburri, F. Palomba, A. Serebrenik, A. Zaidman, Discovering community patterns in open-source: a systematic approach and its evaluation, Empirical Softw. Engg. 24 (2019) 1369–1417. doi:10.1007/s10664-018-9659-9.

[16] D. Piorkowski, S. Park, A. Y. Wang, D. Wang, M. Muller, F. Portnoy, How ai developers overcome communication challenges in a multidisciplinary team: A case study, Proc. ACM Hum.-Comput. Interact. 5 (2021). doi:10.1145/3449205.

[17] B. Karlaš, M. Interlandi, C. Renggli, W. Wu, C. Zhang, D. Mukunthu Iyappan Babu, J. Edwards, C. Lauren, A. Xu, M. Weimer, Building continuous integration services for machine learning, in: Proc. of the 26th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining, KDD '20, ACM, New York, NY, USA, 2020, p. 2407–2415. doi:10.1145/3394486.3403290.

[18] J. Cito, G. Schermann, J. E. Wittern, P. Leitner, S. Zumberi, H. C. Gall, An empirical analysis of the docker container ecosystem on github, in: 2017 IEEE/ACM 14th Int'l Conf. on Mining Software Repositories (MSR), 2017, pp. 323–333. doi:10.1109/MSR.2017.67.

[19] Y. Wu, Y. Zhang, T. Wang, H. Wang, Characterizing the occurrence of dockerfile smells in open-source software: An empirical study, IEEE Access 8 (2020) 34127–34139. doi:10.1109/ACCESS.2020.2973750.