

Towards Architecting Sustainable MLOps: A Self-Adaptation Approach

Hiya Bhatt^{*† §}, Shrikara Arun^{†§}, Adyansh Kakran[†] and Karthik Vaidhyanathan[†]

^{*}Manipal University Jaipur (MUJ) [†]Software Engineering Research Center, IIIT Hyderabad
hiyabhatt2002@gmail.com, shrikara.a@students.iiit.ac.in, adyansh.kakran@research.iiit.ac.in,
karthik.vaidhyanathan@iiit.ac.in

Abstract—In today’s dynamic technological landscape, sustainability has emerged as a pivotal concern, especially with respect to architecting Machine Learning enabled Systems (MLS). Many ML models fail in transitioning to production, primarily hindered by uncertainties due to data variations, evolving requirements, and model instabilities. Machine Learning Operations (MLOps) offers a promising solution by enhancing adaptability and technical sustainability in MLS. However, MLOps itself faces challenges related to environmental impact, technical maintenance, and economic concerns. Over the years, self-adaptation has emerged as a potential solution to handle uncertainties. This paper introduces a novel approach employing self-adaptive principles integrated into the MLOps architecture through a MAPE-K loop to bolster MLOps sustainability. By autonomously responding to uncertainties, including data, model dynamics, and environmental variations, our approach aims to address the sustainability concerns of a given MLOps pipeline identified by an architect at design time. Further, we implement the method for a Smart City use case to display the capabilities of our approach.

Index Terms—Sustainability, Self-Adaptation, MLOps

I. INTRODUCTION

The concern about the sustainability of software systems has been exacerbated by the emergence of Machine Learning-enabled systems (MLS), which are software systems that incorporate ML models. This is due to their computational complexity and uncertainties arising from their probabilistic nature. Studies like those by Gartner [1] show that almost half of the ML models do not successfully transition from prototype to production because they are not sustainable. Sustainability encompasses four dimensions: environmental, technical, social and economical [2]. Environmental concerns, including energy consumption and carbon emissions, are particularly pertinent as ML models often require significant computational resources while training, retraining and deployment. Technical concerns involve ensuring the maintainability [3] and reliability of both the ML models (the core of MLS) and the pipelines (the sequences of data processing and learning steps), which can be challenging given the rapidly evolving nature of ML. Moreover, MLS faces multiple Social concerns like fairness, privacy, explainability, and broader issues like ethics and legislation. Economic concerns are tied to the cost of training, testing, and inference.

Machine Learning Operations (MLOps) [4] aims to enhance the technical sustainability of an MLS by architecting MLOps

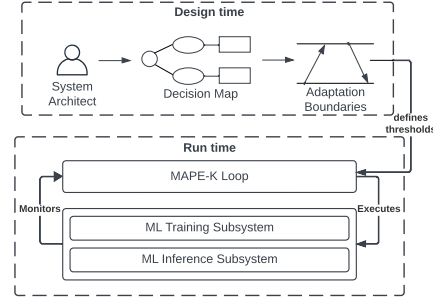


Figure 1. Flow diagram for our approach

pipelines, which include the development, deployment, and maintenance of ML models. However, MLOps needs to be enhanced to address the other sustainability concerns. These sustainability concerns for the MLOps pipelines can be identified by an architect at design time and visually represented in a Decision Map (DM) [5].

Although a DM helps capture the sustainability concerns at design time, some of the concerns arise from runtime uncertainties inherent to MLOps pipelines. These include drifts in data quality, model quality, evolving business needs, and the dynamic environmental context, as put forward by [6] [7]. Over the years, self-adaptation has emerged as a potential solution to handle runtime uncertainties [8]. However, not much work has been done on integrating self-adaptation with MLOps to make it sustainable.

The research question we pose is “(How) can self-adaptation be used in MLOps to improve sustainability?” To this end, this paper proposes a novel approach that uses self-adaptation in MLOps pipelines through a MAPE-K loop [9] to improve their sustainability. As in Figure 1, the system architect creates a DM at design time, from which adaptation boundaries are realised. This information is stored in the Knowledge base of MAPE-K and enables run time self-adaptation of the MLOps pipelines to enhance sustainability across dimensions. We further demonstrate the practicality of our approach using a Smart City case study to predict Air Quality.

II. RELATED WORK

There has been significant effort related to engineering MLS through MLOps, including [4], [10], [11]. Traditional approaches do not address sustainability through self-adaptation. The closest work we find is [12] describing a self-adaptive

[§]Equal contribution

approach to handle drift in an industrial process. Our approach differs since it deals not only with model drift but also with other sustainability aspects. What little work on self-adaptive MLOps that exists fails to address the concern of sustainability, across such dimensions mentioned in this paper, such as [13], who apply a MAPE-K loop, and deal entirely with the method and effects of self-adaptation in MLS. Whereas [14] discusses MLOps and AI Software sustainability, it is not through the lens of runtime self-adaptation. The concept of runtime goal management is described by [15] by expressing adaptation intent as a sustainability goal. They propose an approach that uses decision maps to make sustainability-driven decisions for adaptation in a systematic way but does not deal with MLS and the added uncertainties they add due to their probabilistic nature. Switching the lens to the field of software sustainability, [5] describes decision maps for software sustainability and categorizes concerns into social, technical, environmental, and economic; and discusses immediate, enabling, and systemic impacts. This, too, does not deal with MLS. Differently from the above works, we propose an approach that aims to enhance the sustainability of MLOps pipelines and thereby MLS by taking into consideration design time goals through runtime self-adaptation powered by MAPE-K loop.

III. CASE STUDY

The Smart City Living Lab of IIITH¹ is a research platform and test bed for smart city applications comprising more than 300 IoT nodes spanning various domains such as air quality, water quality and quantity, solar power monitoring, home automation, etc. Due to increasing vehicular traffic and pollution, air quality monitoring (involves monitoring of Air Quality Index, AQI) has emerged as one of the key domains as this can contribute to the establishment of environmental regulations in urban India. As part of this domain, there are 10 outdoor and 5 indoor sensor nodes deployed in the campus which collect air quality information such as PM2.5, PM10 levels, temperature and humidity at a target frequency of once every second. These data are then used by ML pipelines to forecast AQI [16]. However, these pipelines do not take into consideration the sustainability concerns arising from model and data drifts (technical), increased cost and energy consumption due to frequent retraining (economical and environmental). To this end, we envision architecting a self-adaptive MLOps pipeline to enhance sustainability. We use this case study in the remainder of this paper to explain our approach and the results obtained through our preliminary evaluations.

IV. DEFINING OUR APPROACH

Our architecture utilizes the MAPE-K framework for self-adaptation to respond to detected uncertainties. The three components of our approach are: the Managed System, which is the MLS, the Managing System, which is responsible for monitoring the system and its environment, detecting uncertainties at runtime, planning and executing adaptations; and

the Decision Map, which captures the sustainability concerns of the system at design time.

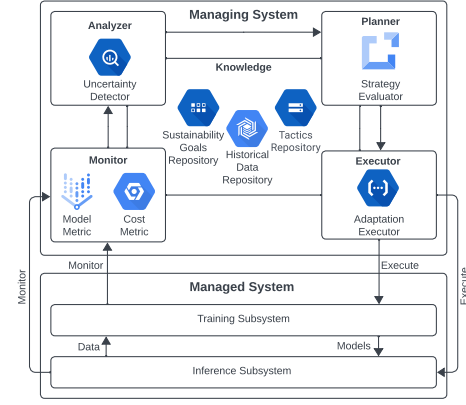


Figure 2. Approach

A. Managed System

The Managed System encompasses the entirety of the MLOps pipeline, consisting of two components, as in [17]:

- 1) *Training Subsystem* is responsible for developing and refining models based on available data. It handles model and data versioning to track changes and improvements.
- 2) *Inference Subsystem* serves trained models obtained from the training subsystem to deliver online/batch predictions.

B. Decision Map

A Decision Map (DM), as shown in Figure 3, is a visual representation of sustainability concerns across the Social, Environmental, Technical and Economic dimensions [5]. A system architect decides the sustainability goals for each concern during design time, setting the stage for runtime adaptation. The architect also defines and modifies adaption boundaries which dictate the acceptable quality of the system across sustainability dimensions [15]. For instance, (rt_{max}, rt_{min}) , where rt_{min} and rt_{max} denote the minimum and maximum allowed response time of the system.

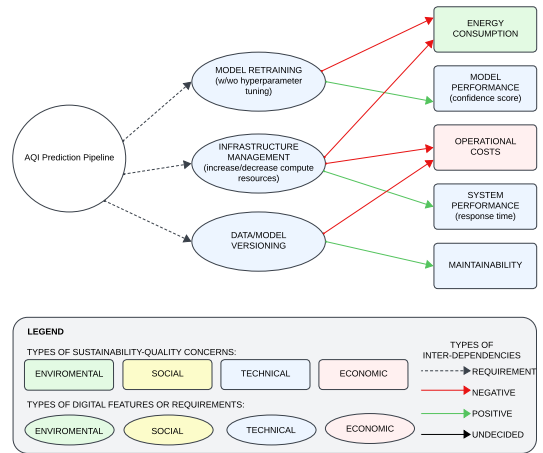


Figure 3. Decision Map for AQI Prediction Pipeline

The DM in Figure 3 describes the sustainability concerns for our Case Study, a) *Model Retraining*: We would need

¹<https://smartcitylivinglab.iiit.ac.in> (Last accessed 10 Feb, 2024)

Table I
SELF-ADAPTATION UNCERTAINTIES, TACTICS AND STRATEGIES FOR SUSTAINABLE MLOPS
(THOSE IN BOLD HAVE BEEN IMPLEMENTED IN THE CASE STUDY)

Uncertainty	Concern	Impact			Tactics	Strategy
		Immediate	Enabling	Systemic		
1. Model Drift	Technical	Reduced quality of predictions	Reduced user trust	Reduced social benefits	1. Model switch	Switch to a model with better performance but higher energy use
					2. Retrain	a. Conduct incremental learning b. Conduct transfer learning c. Conduct complete retraining (with/without different hyperparameters)
2. High energy consumption	Environmental	Increased costs	Increased environmental impact	Reduced economic competitiveness	Model switch	Switch to a model with lower energy consumption
3. Future goals changes	Technical	Increased uncertainty: system's purpose	Difficulty in maintaining the system	Reduced public trust in AI	Update model objectives	Change thresholds for: a. Business metrics b. Cost metrics c. Model metrics
4. Rise in cost	Economic	Increased costs	Difficulty in deploying and maintaining the system	Reduced ability to compete in the market	Optimize the system to reduce costs	a. Reduce the amount of resources used by the system b. Use more cost-effective resources

to retrain our models with the latest data to account for shifts in distribution of AQI caused due to seasons, increased emissions and weather. b) *Infrastructure Management*: The hardware infrastructure used to train the model and conduct inference can affect response time, cost and energy utilization. c) *Data/Model Versioning*: Improves maintainability and reproducibility, but can increase operational costs.

Considering these concerns, we identify some uncertainties from [6] that affect them, as well as their *immediate*, *enabling* and *systemic impacts* in Table I. Table I also contains the tactics to mitigate identified uncertainties and the strategies that can be employed to carry out the tactics, more of which can be created by the architect.

C. Managing System

Knowledge: The Knowledge base, as in Figure 2 is divided into three subsections: 1) *Sustainability Goals Repository* stores the sustainability goals defined by the system architect during the design phase, and the acceptable threshold for metrics derived from the DM which are defined by the adaptation boundaries. 2) *Historical Data Repository* stores the version history of models and data. 3) *Tactics Repository* stores mitigation tactics and the corresponding strategies.

Monitor: The Monitor component, as in Figure 2 continuously collects data about the managed system's state, including cost metrics and model metrics. The *Cost Metrics* include the cost of compute resources, data storage, model training, and inference. The *Model Metrics* evaluate the model's performance, including its confidence/accuracy and energy consumption.

Analyzer: The core of the Analyzer component is the *Uncertainty Detector* (refer Figure 2), which analyzes the metrics collected by the Monitor for anomalies or trends that might signal uncertainties. These uncertainties, as defined in Table I, are flagged based on thresholds in the *Sustainability Goals Repository*. The *Uncertainty Detector* assesses the potential impact of these uncertainties on the managed system to deter-

mine if an adaptation is necessary. These adaptation boundaries delineate the acceptable quality of the system [15]. These boundaries are not fixed and can be adjusted based on factors such as the system's current state, changing requirements, and environmental conditions. This makes it more effective at detecting anomalies and other changes that may indicate uncertainty since our definition of uncertainty, too, may evolve. For instance, assume that in our case study (refer Section III), the energy consumed for performing the forecasts, E , violates the sustainability goals defined for energy consumption such that $E \leq E_{min}$ or $E \geq E_{max}$, then we can say that the quality of the system has deteriorated, and the Planner will be triggered to decide the adaption strategy.

Planner: When the Analyzer detects an uncertainty, the Planner component determines the optimal strategy to address it using the *Strategy Evaluator*. It decides on the tactic and strategy the Managing system should employ by mapping the detected uncertainties to their respective tactic and strategy. For instance, in our case study, if the trigger for the planner is due to violations of energy, then the planner may switch the model (refer Table I). Additionally, more sophisticated planner strategies can incorporate Reinforcement Learning (RL) [18], model checkers [19] or clustering [20] methods in combination with the aforementioned methods.

Executor: The *Adaptation Executor* in the Executor implements the strategy selected by the Planner component. It interacts with both the training and the inference subsystem. For instance, it can trigger a (re)training job in the training subsystem or can switch to a different model in the inference subsystem based on the selected strategy.

V. PRELIMINARY RESULTS

We implemented our approach (by addressing the model drift & high energy consumption uncertainties which affect the technical & environmental sustainability concerns respectively) on the AQI forecasting pipeline (Section III). We utilize

the DM in Figure 3 to realize adaptation boundaries, which are mentioned in the GitHub repository². The forecasting pipeline consists of two different models, one which uses Linear Regression (lightweight model with low response time & prediction quality) & one which makes use of a Long Short Term Memory network (LSTM) (heavier model with high prediction quality & response time.)

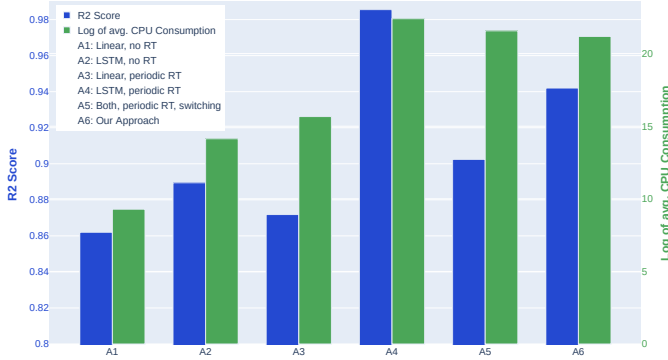


Figure 4. R^2 Score & Log of Average CPU Consumption over 10s (μJ). RT=retraining

We further evaluated our approach against five baseline approaches: two that use Linear Regression & LSTM, each without retraining (RT) (A1 & A2), two that use periodic RT [21] (A3 & A4), and one that uses both models with periodic RT & switching (A5). We switch between the models based on CPU consumption. As shown in Figure 4, our approach (A6) strikes a balance between performance, measured by R^2 score and average CPU consumption over the past 10s, measured in μJ . Unlike A3, A4 & A5, we retrain the models only when model drift is detected. We calculate the model drift using KL divergence, which quantifies the disparity between the distribution of training data and real-time encountered data. While using only the LSTM with periodic retraining (A3) offers the best R^2 score, it consumes significantly more energy than other approaches. Our approach, as compared to periodically retraining both models and switching between them (A5), improves R^2 score from 0.90 to 0.94 and reduces average CPU consumption by 32%.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose an architecture for a sustainable MLOps pipeline by utilizing self-adaptation through a MAPE-K loop. During design time, a system architect identifies adaptation boundaries through the creation of a decision map to allow for runtime self-adaptation. Preliminary evaluations in our case study show that our approach strikes a balance between performance and CPU consumption. We believe that a self-adaptive MLOps architecture can pave the way to increase sustainability of MLOps pipelines. Future work includes evaluating the generalizability of our approach to different domains and identifying more uncertainties and alleviation tactics, especially in the Social dimension. While we focus on supervised ML tasks, work also needs to be

done to address unsupervised and reinforcement learning tasks. The rapid landscape of generative AI also presents promising avenue for further research.

REFERENCES

- [1] K. Costello and M. Rimol, "Gartner identifies the top strategic technology trends for 2021," 2020.
- [2] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, "Sustainability design and software: The karlskrona manifesto," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, 2015, pp. 467–476.
- [3] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," *Advances in neural information processing systems*, vol. 28, 2015.
- [4] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas, "Mlops - definitions, tools and challenges," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 2022.
- [5] P. Lago, "Architecture design decision maps for software sustainability," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 2019, pp. 61–64.
- [6] J. Cámara, J. Troya, A. Vallecillo, N. Bencomo, R. Calinescu, B. H. Cheng, D. Garlan, and B. Schmerl, "The uncertainty interaction problem in self-adaptive systems," *Software and Systems Modeling*, vol. 21, no. 4, 2022.
- [7] M. Casimiro, P. Romano, D. Garlan, and L. Rodrigues, "Towards a framework for adapting machine learning components," in *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, 2022, pp. 131–140.
- [8] D. Weyns, *An introduction to self-adaptive systems: A contemporary software engineering perspective*. John Wiley & Sons, 2020.
- [9] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.
- [10] S. Shankar, R. Garcia, J. M. Hellerstein, and A. G. Parameswaran, "Operationalizing machine learning," *University of California, Berkeley*.
- [11] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019, pp. 291–300.
- [12] F. Bayram, B. S. Ahmed, E. Hallin, and A. Engman, "A drift handling approach for self-adaptive ml software in scalable industrial processes," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–5.
- [13] M. Casimiro, P. Romano, D. Garlan, G. A. Moreno, E. Kang, and M. Klein, "Self-adaptation for machine learning based systems," in *ECSC (Companion)*, 2021.
- [14] D. A. Tamburri, "Sustainable mlops: Trends and challenges," in *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNAS)*, 2020, pp. 17–23.
- [15] I. Gerostathopoulos, C. Raibulet, and P. Lago, "Expressing the adaptation intent as a sustainability goal," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 36–40.
- [16] N. Niles, I. Patwardhan, J. Narang, and S. Chaudhari, "Tot-based aqi estimation using image processing and learning methods," in *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*. IEEE, 2022, pp. 1–5.
- [17] "Uber Michelangelo ML Platform," 9 2017. [Online]. Available: <https://www.uber.com/blog/michelangelo-machine-learning-platform/>
- [18] A. Metzger, C. Quinton, Z. Á. Mann, L. Baresi, and K. Pohl, "Realizing self-adaptive systems via online reinforcement learning and feature-model-guided exploration," *Computing*, 2022.
- [19] C. Stevens and H. Bagheri, "Reducing run-time adaptation space via analysis of possible utility bounds," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020, p. 1522–1534.
- [20] S. Kulkarni, A. Marda, and K. Vaidyanathan, "Towards self-adaptive machine learning-enabled systems through qos-aware model switching," *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*, 2023.
- [21] R. Nazir, A. Bucaioni, and P. Pelliccione, "Architecting ml-enabled systems: Challenges, best practices, and design decisions," *Journal of Systems and Software*, vol. 207, p. 111860, 2024.

²<https://github.com/sa4s-serc/sustainableMLOps>