From Concept to Implementation: The Data-Centric Development Process for AI in Industry

Paul-Philipp Luley*, Jan M. Deriu*, Peng Yan*†, Gerrit A. Schatte‡, Thilo Stadelmann*§
{lule, deri, yanp, stdm}@zhaw.ch, gerrit.schatte@kistler.com
*Centre for Artificial Intelligence, ZHAW School of Engineering, Winterthur, Switzerland
†Institute of Neuroinformatics, ETH & University of Zurich, Zurich, Switzerland
‡Innovation Lab, Kistler Instrumente AG, Winterthur, Switzerland
§European Centre for Living Technology (ECLT), Ca' Bottacin, Venice, Italy
§Fellow, ECLT (European Centre for Living Technology, Venice, Italy) and Senior Member, IEEE

***Abstract*—We examine the paradigm of data-centric artificial intelligence (DCAI) as a solution to the obstacles that small and medium-sized enterprises (SMEs) face in adopting AI. While the prevalent model-centric approach emphasizes collecting large amounts of data, SMEs often suffer from small datasets, data drift, and sparse ML knowledge, which hinders them from implementing AI. DCAI, on the other hand, emphasizes to systematically engineer the data used to build an AI system. Our contribution is to provide a concrete, transferable implementation of a DCAI development process geared towards industrial application, specifically in machining and manufacturing, and demonstrate how it enhances data quality by fostering collaboration between domain experts and ML engineers. This added value can place AI at the disposal of more SMEs. We provide the necessary background for practitioners to follow the rationale behind DCAI and successfully deploy the provided process template.**

***Index Terms*—MLOps, ML pipeline, data preparation**

I. Introduction

The development of AI systems based on machine learning (ML) fundamentally revolves around extracting knowledge from data and materializing that knowledge within a model. This is predominantly done by model-centric approaches [1]. In model-centric AI (MCAI), the model is treated as the first-class citizen, i.e., development revolves around engineering the model while keeping the data fixed after an often one-time collection and preparation effort. Based on the assumption that a model performs better when trained with more data [2], the only emphasis given to the data itself is to collect as much data as possible, which is thoroughly studied and unanimously recommended for shallow and deep learning [3]–[6]. MCAI excels in several disciplines and has led to undeniably important advances (cp. [7], [8]). Thus, especially consumer internet companies, which are able to serve both the demand for sufficient data, the corresponding compute capacities, and employ the necessary expertise, rank among the main beneficiaries of AI [9], [10].

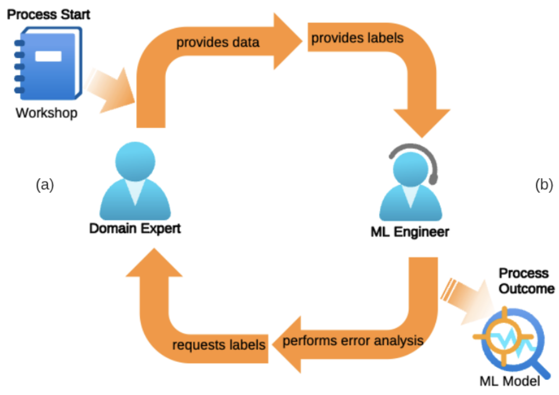
At the same time, these conditions often do not reflect the reality of other companies like SMEs. Often, they suffer from (i) data drift [11], (ii) small datasets [12], and (iii) sparse ML knowledge [13], as part of the long-tail problem [14]. This excludes many from AI adoption, despite its benefits [15]. A solution to the listed obstacles can be found in the paradigm of data-centric AI (DCAI) [11], [16]. It "is the discipline of systematically engineering the data used to build an AI system" [1]. For this, it utilizes common engineering tools [17] like data cleaning, feature selection, data transformation, feature engineering, and dimensionality reduction, alongside systematic error analyses [18], in order to enhance data quality through well-defined and tool-supported processes.

In this paper, based on a systematic literature review, our contribution is a concrete, novel, and transferable template for a data-centric development process derived from an AI application in the machining industry. Equipped with role descriptions (to foster collaboration between data-providing experts and ML engineers), list of process steps (to adapt to the application at hand), and tooling overview (for concrete implementation of the process), the template enables practitioners with a hands-on example to follow off-the-shelf, supporting the more wide-spread adoption of AI [19].

II. Related Work

Since the inception of the term in 2021, the concept of DCAI has sparked widespread interest. Hamid [20] wraps up the evolution from MCAI to DCAI and motivates a complementary understanding of both terms, where DCAI builds on top of pre-trained models like foundation models [8]. Whang et al. [21] address DCAI implementation with established tools. Jarrahi et al. [22] formulate six DCAI principles, but stay vague with their advice for practitioners. Jakubik et al. [23] span the arc from definition to framework, but their framework does not go beyond a graphical representation of instructions. Rather, they focus on the impact on information systems. Summarizing, these overviews provide valuable information about DCAI but lack practical utility.

The following substantial implementations deal with the practical impact of DCAI on label consistency. Hao et al. [24] address which subset of data to improve and demonstrate the effectiveness of a pre-trained model supported by an active learning loop for a computer vision (CV) task. Their approach can be fully automated and does not rely on a human in the loop, whereas Nguyen et al. [25] concentrate on human integration into the labeling process (for 3D objects). They utilize a pre-trained model for a preliminary annotation in 3D



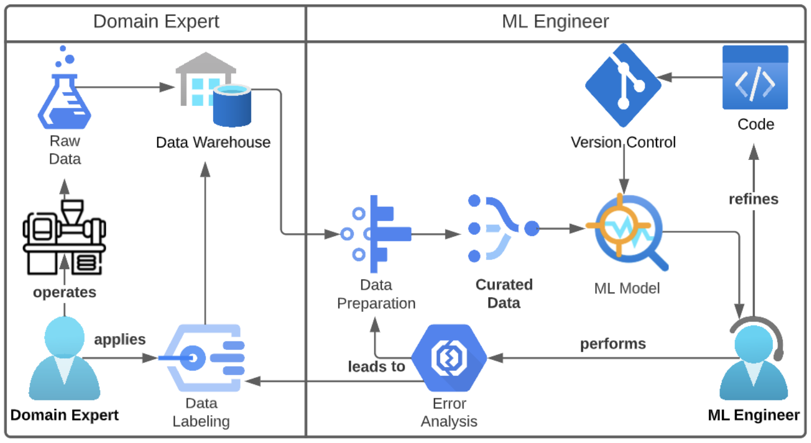


Fig. 1: (a) The proposed data-centric development process with roles and activities; (b) the MLOps pipeline enabling it.

and reduce the amount of annotator clicks to one with a good software integration. Further examples of how the labeling or re-labeling process can be made more efficient and/or effective to ultimately improve labeling quality are presented at the same workshop in [26]–[28]. All authors agree to use a pre-trained model as starting point for DCAI approaches.

There are a number of interesting application examples for DCAI. Yin et al. [29] try to close the model performance gap between training and real production for an entity resolution system. They augment the data with different human-in-the-loop strategies, improving the F1 score from 95.07 to 98.79. Liu et al. [30] release AutoDC as a pendant to AutoML [31]. Whereas AutoML facilitates building custom ML applications in an MCAI way, AutoDC seeks to automate outlier detection, label correction, edge case selection, and data augmentation for users without particular programming skills, e.g., domain experts. Outliers are implicitly considered incorrectly labeled or edge cases. Accordingly, in the next steps, users are asked to correct the labels, followed by selecting the edge cases, each of which is aided by AutoDC. Bai et al. [32] obtain an unsupervised representation of their CV data by contrastive learning methods. In addition, based on a nearest neighbor graph, they add pseudo-labels to enrich the dataset. The work of Huang et al. [33] is an example of what a truly seamless DCAI integration can look like for a software company. They tackle the mentioned problems of label scarcity and label bias and facilitate the continuous training induced by DCAI as a means for concept and data drift detection. Several other works [34]–[36] explore the use of novel DCAI frameworks for their use cases, tackling label noise or sparsity and promoting continuous learning. However, their frameworks are either highly customized or poorly maintained, hindering adoption. The literature thus reveals a gap between concrete DCAI applications for specific use cases and a framework ready for adoption between the conceptual and the implementation level—a data-centric development process.

III. A Real-Life Development Process for DCAI

A typical development process for MCAI solutions is straightforward: Let domain experts (potentially after some schooling on ML) gather data, and pass this dataset "as is" to the ML engineers for further pre-processing and modeling; feedback loops are not foreseen [37], and data problems are mitigated with modeling approaches [38]. How should a development process look like that is data-centric, i.e., considers the data as the main object of manipulation?

Process Overview Our proposed data-centric process is depicted in Figure 1(a). At the core of the process, we propose a tight collaboration between the domain expert and the machine learning engineer. This lies in contrast to the MCAI approach of a one-shot exchange between the two roles. The iterative process tackles the main issues that arise from the more specialized proposals surveyed above. First, since data is scarce and annotations are costly, the samples to be labeled are chosen by the machine learning expert so that their impact is maximized. Second, the domain expert is in charge of interpreting the data, which is often ignored in the MCAI setting where the machine learning engineers simply use the data at their discretion. Lastly, data and concept shifts are anticipated and handled inherently by the iterative process.

Roles and Responsibilities The communication between the two roles is defined as follows: Whenever domain experts provide new data and/or labels, this triggers the automatic model building pipeline of training and evaluation. The respective results prompt the ML engineers to inspect the outcome, analyze error cases, and annotate the samples in question with comments and requests to the domain experts. Thus, the ML engineers are responsible for the analysis of the causes of inadequate model performance and issue requests; domain experts review requests (potentially in real time) and provide answers in form of data, labels and/or explanations. Both roles have the advantage of getting important information quicker than with a classical (MCAI) development process. Turnaround times for model iteration are reduced, giving domain experts a clearer picture on where their AI solution is heading, and providing ML engineers with direct feedback whether their work is improving practical performance.

Thus, in the DCAI setting the main focus is to provide an integrated set of tools that facilitate the communication between the domain expert and the machine learning engineer. This includes tools that are specific for each role, tools for the communication, and fostering the trust rooted in a common understanding of each other's work as well as the final model. In the next part, we describe such a MLOps pipeline.

IV. Tooling Landscape

Figure 1(b) depicts the types of tools needed for the seamless end-to-end integration of the whole DCAI process with a MLOps pipeline. There are three types of tools that are needed: (i) Tools for the domain expert, (ii) tools for the ML engineer, and (iii) tools for the orchestration and communication. We set the focus on open-source tools, which allow higher flexibility when adapting to novel use cases.

Tool Support for the Domain Expert Role The domain expert essentially needs two types of tools: The data labeling tool, and a tool for providing new raw data. The latter is highly domain specific and must be provided by the domain expert. For instance, the apparatus to collect and provide astrophysical data is different from ways to supply injection molding sensor data. The list of labeling tools under consideration was narrowed down to Diffgram [39], Label Studio [40] and TRAINSET [41] due to the ability to also label time series (TS) data. All are open-source, but TRAINSET is poorly maintained. Diffgram and Label Studio are supported by a professional community, but Label Studio is more popular and provides more TS-specific functionality, hence it is selected. It has a thought-through and user-friendly front end that does not require any specialized ML knowledge. Thus, it can be operated autonomously by the domain expert, reducing the complexity of ML tasks and the reliance on an ML engineer.

Tool Support for the ML Engineer Role Disentangled from the data acquisition process, the ML engineer can tune and improve the model as part of the cyclic ML workflow: (i) tune data and hyperparameters, (ii) train the model, and (iii) error analysis. For this, the ML engineer needs tools for data preparation, a version control for models and code, and an error analysis component (tools focusing on model development are purposefully left out here but discussed elsewhere [42]). While the landscape for the other types of tools is already vast (e.g., Neptune [43], DVC, MLMD [44], GitHub), there are only a few choices supporting error analysis.

Error analysis is needed to identify the specific samples of data that degrade the model's performance. That is, when tuning the model does not lead to further progress, our development process motivates the ML engineer to inspect the data. Most error analysis tools are highly specific depending on the use case. For computer vision (CV) tasks, Domino [45] and Spotlight [46] are open-source options which offer less customization, LandingLens [47], on the other hand, provides a data-centric pipeline but is offered as a closed-source commercial web interface. For natural language processing tasks, SEAL [48] has a sophisticated GUI but comes only with clustering as a means for outlier detection. The most mature tool is called Error Analysis [49]. It supports all types of numeric inputs, is open-source, actively maintained, and provides measures for identifying malicious samples using decision tree and heatmap, and for examining data and feature distributions to evaluate a sample's impact.

Implementation Backbone and Communication Orchestration of the whole process, bringing together the tools for the domain expert and machine learning engineer, is implemented by two components: (i) Communication tools transfer the labeled data from the domain expert to the machine learning engineer, and send the samples identified by error analysis back to the domain expert; (ii) orchestration tools provide the platform on which all these tools are deployed and integrated. For orchestration there exist many solutions such as Airflow [50], [51] or more commercial services like Microsoft Azure, Amazon Sagemaker, and Google Cloud Platform, that offer similar functionality pre-built at the expense of less control.

A more integrated orchestration leads to a more integrated process. For instance, a change in a label of a sample by the domain expert is automatically communicated to the machine learning engineer. This is done via a data warehouse, for which there exist many solutions already (e.g., MySQL, SQL Server, and PostgreSQL). Most importantly, a consistent structure of the data for a certain period of time is ensured, which facilitates the work of the ML engineer. The orchestration can be automated so far that as soon as new data arrives, an event is triggered that initiates a new build of the most recent ML model pulled from version control. To keep track of the data changes, DVC [52] can be used, whereas MLflow [53] keeps track of the model lifecycle. DVC works on top of Git [54] and is therefore convenient to use. Compared to its competitors (Pachyderm [55], Neptune) it stands out as an open-source, light-weight, atomic data versioning tool. MLflow complements it nicely as it keeps track of every run's model information independent of Git commits. The main advantage over its competitors (Neptune, Amazon Sagemaker, Verta AI, Azure ML) is its open-source nature.

V. Conclusions and Future Work

In this paper, we introduced a development process and supporting MLOps pipeline that implements the vision of data-centric artificial intelligence, which is better suited for small-and-medium-sized enterprises. At the core lies the tight and iterative coupling of the domain expert and the machine learning engineer. This lies in contrast to the model-centric view, which only considers a one-time interaction between the two roles. We gave an overview of the types of tools needed to facilitate this process and presented concrete tools that exist already. We focus on open-source tools, which allow maximal control and adaptability to the use case at hand. However, we found that for the crucial step of error analysis the tooling landscape is still scarce, and more general solutions need to be developed. Thus, when applying our process to novel problems, custom tools may be needed for certain steps in the pipeline, leaving a fully general solutions as an open problem.

A natural continuation of our work is the integration of automated data augmentation since synthesizing data holds enormous prospects for applied AI, albeit being challenging. Also, wrapping up the MLOps pipeline underlying the development process as an integrated package is left for further development. It would be insightful to compare different pipeline configurations with respect to their efficacy in facilitating AI system development in practice.

Acknowledgement This work has been supported financially by Innosuisse grant 101.787 IP-ENG "DISTRAL".

References

[1] DeepLearningAI, "A chat with Andrew on MLOps: From model-centric to data-centric AI." Available online: https://www.youtube.com/watch?v=06-AZXmwHjo, Mar. 2021.

[2] A. L'Heureux et al., "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[3] G. Foody et al., "The effect of training set size and composition on artificial neural network classification," *Int J Remote Sens*, vol. 16, no. 9, pp. 1707–1723, 1995.

[4] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Inf Sci*, vol. 179, no. 8, pp. 1040–1058, 2009.

[5] P. Tsangaratos and I. Ilia, "Comparison of a logistic regression and naïve bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size," *CATENA*, vol. 145, pp. 164–179, 2016.

[6] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Comput Electron Agric*, vol. 153, pp. 46–53, 2018.

[7] L. Tuggener et al., "Is it enough to optimize CNN architectures on ImageNet?," *Front Comput Sci*, vol. 4, 2022.

[8] R. Bommasani et al., "On the opportunities and risks of foundation models," *arXiv:2108.07258*, 2021.

[9] S. Makridakis, "The forthcoming artificial intelligence (AI) revolution: Its impact on society and firms," *Futures*, vol. 90, pp. 46–60, 2017.

[10] N. Ahmed and M. Wahed, "The de-democratization of AI: Deep learning and the compute divide in artificial intelligence research," *arXiv:2010.15581*, 2020.

[11] E. Strickland, "Andrew Ng, AI minimalist: The machine-learning pioneer says small is the new big," *IEEE Spectr*, vol. 59, no. 4, pp. 22–50, 2022.

[12] T. Stadelmann et al., "Deep learning in the wild," in *Proc. ANNPR*, pp. 17–38, Springer, 2018.

[13] Databricks, "Data centric AI development from big data to good data Andrew Ng." Available online: https://www.youtube.com/watch?v=avoijDORAlc, July 2022. Last accessed Feb 27, 2023.

[14] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[15] T. H. Davenport and R. Ronanki et al., "Artificial intelligence for the real world," *Harv Bus Rev*, vol. 96, no. 1, pp. 108–116, 2018.

[16] T. Stadelmann, T. Klamt, and P. H. Merkt, "Data centrism and the core of data science as a scientific discipline," *Archives of Data Science, Series A*, vol. 8, no. 2, 2022.

[17] J. Brownlee, *Data preparation for machine learning*. Machine Learning Mastery, 2020.

[18] A. Ng, "Machine Learning Yearning," *URL: http://www. mlyearning. org/(96)*, vol. 139, 2017.

[19] B. Allen et al., "Democratizing AI," *JACR*, vol. 16, no. 7, pp. 961–963, 2019.

[20] O. H. Hamid, "From model-centric to data-centric AI: A paradigm shift or rather a complementary approach?," in *Proc. ITT*, pp. 196–199, 2022.

[21] S. E. Whang et al., "Data collection and quality challenges in deep learning: A data-centric AI perspective," *VLDB J*, pp. 1–23, 2023.

[22] M. H. Jarrahi et al., "The principles of data-centric AI (DCAI)," *arXiv:2211.14611*, 2022.

[23] J. Jakubik et al., "Data-centric artificial intelligence," *arXiv:.2212.11854*, 2022.

[24] H. Hao et al., "Highly efficient representation and active learning framework for imbalanced data and its application to covid-19 X-ray classification," in *Proc. NeurIPS DCAI Workshop*, 2021.

[25] T. Nguyen et al., "Single-click 3D object annotation on LiDAR point clouds," in *Proc. NeurIPS DCAI Workshop*, 2021.

[26] J. Rehm et al., "Utilizing driving context to increase the annotation efficiency of imbalanced gaze image data," in *Proc. NeurIPS DCAI Workshop*, 2021.

[27] I. Jang et al., "Decreasing annotation burden of pairwise comparisons with human-in-the-loop sorting: Application in medical image artifact rating," in *Proc. NeurIPS DCAI Workshop*, 2021.

[28] A. Borovac et al., "Influence of human-expert labels on a neonatal seizure detector based on a convolutional neural network," in *Proc. NeurIPS DCAI Workshop*, 2021.

[29] W. Yin et al., "Combining data-driven supervision with human-in-the-loop feedback for entity resolution," in *Proc. NeurIPS DCAI Workshop*, 2021.

[30] Z. Y.-C. Liu et al., "AutoDC: Automated data-centric processing," in *Proc. NeurIPS DCAI Workshop*, 2021.

[31] L. Tuggener et al., "Automated machine learning in practice: state of the art and recent results," in *Proc. SDS*, pp. 31–36, 2019.

[32] H. Bai et al., "Self-supervised semi-supervised learning for data labeling and quality evaluation," in *Proc. NeurIPS DCAI Workshop*, 2021.

[33] X. Huang et al., "Challenges and solutions to build a data pipeline to identify anomalies in enterprise system performance," in *Proc. NeurIPS DCAI Workshop*, 2021.

[34] C. C. Moreno et al., "Contrasting the profiles of easy and hard observations in a dataset," in *Proc. NeurIPS DCAI Workshop*, 2021.

[35] N. Polyzotis and M. Zaharia, "What can data-centric AI learn from data and ML engineering?," in *Proc. NeurIPS DCAI Workshop*, 2021.

[36] D. Kang et al., "Finding label and model errors in perception data with learned observation assertions," in *Proc. SIGMOD*, p. 496–505, 2022.

[37] T. Stadelmann et al., "Beyond ImageNet: deep learning in industrial practice," in *Applied data science*, pp. 205–232, Springer, 2019.

[38] N. Simmler et al., "A survey of un-, weakly-, and semi-supervised learning methods for noisy, missing and partial labels in industrial vision applications," in *Proc. SDS*, pp. 26–31, 2021.

[39] Diffgram Inc, "Diffgram." Available online: https://github.com/diffgram/diffgram, Feb. 2023.

[40] Heartex, "Label Studio: Data labeling software." Available online: https://github.com/heartexlabs/label-studio, Feb. 2023.

[41] Geocene Inc., "TRAINSET." Available online: https://github.com/Geocene/trainset, Nov. 2022.

[42] G. Bravo-Rocca et al., "Scanflow: A multi-graph framework for machine learning workflow management, supervision, and debugging," *Expert Syst Appl*, vol. 202, 2022.

[43] neptune.ai, "neptune.ai." Available online: https://github.com/neptune-ai/neptune-client, Feb. 2023.

[44] Google, "ML Metadata." Available online: https://github.com/google/ml-metadata, Dec. 2022.

[45] S. Eyuboglu et al., "Domino: Discovering systematic errors with cross-modal embeddings," *arXiv:2203.14960*, 2022.

[46] G. d'Eon et al., "The spotlight: A general method for discovering systematic errors in deep learning models," in *2022 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1962–1981, 2022.

[47] Landing AI, "LandingLens." Available online: https://landing.ai/platform/, Feb. 2023.

[48] N. Rajani et al., "Seal: Interactive tool for systematic error analysis and labeling," *arXiv:2210.05839*, 2022.

[49] Microsoft, "Error Analysis." Available online: https://github.com/microsoft/responsible-ai-toolbox, Feb. 2023.

[50] Apache Software Foundation, "Airflow." Available online: https://github.com/apache/airflow, Feb. 2023.

[51] D. Kreuzberger et al., "Machine learning operations (mlops): Overview, definition, and architecture," *arXiv preprint arXiv:2205.02302*, 2022.

[52] Iterative.ai, "DVC." Available online: https://github.com/iterative/dvc, Feb. 2023.

[53] LF Projects, LLC, "MLFlow." Available online: https://github.com/mlflow/mlflow, Feb. 2023.

[54] Y. Zhao, "MLOps and data versioning in machine learning project." Available online: https://staff.fnwi.uva.nl/a.s.z.belloum/LiteratureStudies/Reports/2020-Internship_report-Yizhen.pdf, 2020.

[55] Pachyderm, "Pachyderm." Available online: https://github.com/pachyderm/pachyderm, Feb. 2023.