Design of a Cloud-Based Data Platform for Standardized Machine Learning Workflows with Applications to Transport Infrastructure

2022 IEEE 21st Mediterranean Electrotechnical Conference (MELECON) | 978-1-6654-4280-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/MELECON53508.2022.9843138

Andrea Bartezzaghi*, Ioana Giurgiu*, Chiara Marchiori*, Mattia Rigotti*, Rizal Sebastian†, Cristiano Malossi*
**IBM Research*, Zürich, Switzerland
†*TNO - Netherlands Organisation for Applied Scientific Research*, Delft, The Netherlands

***Abstract*—This paper describes the IBM One-Click Learning (OCL) platform, an ongoing internal effort in IBM Research aimed at providing end-to-end support for the full data science process. The design decisions behind the platform are presented, specifically with regard to promoting adoption and applicability within application domains where the potential of machine learning (ML) is recognized but still not fully harnessed. We focus on the representative case study of applying ML to civil engineering, a domain of growing interest also thanks to the EU coordination and support action to improve the European standards for inspection, monitoring and maintenance of bridges, tunnels and other types of transport infrastructures. This example is illustrative of the major roadblocks for adoption of ML in new application domains: the diversity in user profiles and familiarity with data science among domain practitioners; the variety in available hardware infrastructure and computing needs; and the heterogeneity, specificity and unsettled evolution of the use case landscape. Removing these roadblocks requires designing a platform explicitly geared towards usability goals of broad accessibility on one hand, and extensibility and specialization on the other one. We elaborate on a set of functionalities to support these usability goals and thereby enable the design of a task agnostic end-to-end platform to drive ML adoption and workflow standardization in new dynamic application domains. Finally, we present the IBM OCL platform as a proof-of-concept implementation of these functionalities and validate it in a use case where computer vision models are deployed to aid visual inspection of a bridge.**

***Index Terms*—Machine Learning, cloud data platform, standardization, transport infrastructure, inspection and monitoring**

I. Introduction

Adequate information from inspection and monitoring of constructions is crucial to take the right decisions on maintenance and safety of bridges and tunnels. In Europe, these transport infrastructures are vital for the functioning and growth of the economy and society. Unfortunately, these infrastructures are ageing; in the last two decades major failures of bridges and tunnels in Europe caused hundreds of casualties.

Digitalization is key to obtaining, analyzing and sharing accurate information about the condition and maintenance of transport infrastructures while they are coping with increasing traffic loads and resilience threats. Therefore, digitalization has become an important aspect in updating the existing European standards and developing new standards for inspection, monitoring, maintenance and safety of transport infrastructures. The standardization-oriented effort is performed in the EU Coordination and Support Action titled "IM-SAFE" [1].

This paper describes the IBM One-Click Learning (OCL) platform, an ongoing effort internal at IBM Research aimed at providing end-to-end support for the full data science process. In particular, we will focus on the functionalities of the IBM OCL platform that help implement the standardization goals of IM-SAFE and address some of the challenges arising in Big Data and Data Analytics in general, and within the use specifications of inspection and structural health monitoring (SHM) of transport infrastructures in particular. The focus will therefore be on the general design considerations behind the IBM OCL platform that enable a unified standardized treatment of generic Machine Learning use cases and workflows.

In the next section, a brief overview of the existing data and machine learning (ML) platforms is presented. We then delineate the design considerations behind IBM OCL in terms of functionalities that enable a standardized unified workflow accommodating generic ML workloads and pipelines. We then explain how these functional requirements are specifically implemented by the architecture of OCL. Then, in order to validate the applicability of the proposed solution, we will present a case study demonstrating a computer vision object segmentation task applied to the identification of defects for the inspection of a bridge. Finally, conclusions on the achievements and lessons-learned are drawn, and recommendations for further development and standardization are addressed.

II. Related and Relevant ML Platforms

There have been several important works in the past aimed at creating cloud-based solutions that would broaden user accessibility. An important way to achieve that is to free the user from having to worry about how to provision compute infrastructure by *automating resource allocation*. This challenge has been tackled for instance by [2]–[4]. This line of work was technically supported by studies that investigated

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958171. IBM and IBM OCL are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

the suitability specifically of cloud infrastructure for specific ML workloads [5]–[11].

Other important developments to lower the barrier for entry to large-scale machine learning were to provide generic ML as a Service (MLaaS) ([12]), and to allow users to access a prediction service through an API and a Graphical User Interface as demonstrated by [13].

Some of these considerations have been implemented in several existing commercial platforms among which some of the most popular are: IBM Cloud Pak for Data [14] (an end-to-end ML platform available on IBM Cloud which integrates IBM Watson Studio for model automation); Google Cloud AI [15], Azure Machine Learning [16] and Amazon SageMaker [17] (enterprise-grade platforms for the end-to-end ML lifecycle from Google, Microsoft and Amazon, respectively); H2O.ai [18] (an open source end-to-end ML platform).

All mentioned commercial platforms provide Graphical User Interfaces that abstract away resource allocation as well as several aspects of the selection of the appropriate algorithms and model hyperparameters for a particular task of interest. That helps considerably lowering the barrier of entry into the utilization of ML methods to practitioners that are not expert in data science and managing its supporting hardware infrastructure.

The main additional feature that differentiates between these commercial platforms and the OCL platform that we are presenting in this article is the focus on a modular design that enables high interoperability between tasks and workloads along the whole ML pipeline, sharing access to these among users, and the quick integration of novel elements (models, data preprocessing or training procedure, etc) from prototypes and R&D assets at the bleeding edge of the ML development cycle. This design choice was motivated by the decision to guarantee two core usability goals of 1. *broad accessibility*, irrespective of user expertise with coding, ML or hardware, and 2. *extensibility and specialization* of the platform in order to quickly incorporate evolving practices and needs within the application domain. In the section below we elaborate on this design choices and the functional requirements implemented in the IBM OCL platform that enable them.

III. Essential functional requirements to enable workflow standardization

Behind the design of OCL are several functional requirements that have been taken into consideration in order to enable workflow standardization across different use cases. These have shaped architectural and implementation decisions.

The main roadblock to adoption and standardization of ML workflows within specific application domains and industries is first of all the large diversity in user profiles and degrees of familiarity with data science among domain practitioners. This challenge is further compounded by the difficulties that potential adopters might face while negotiating their computing needs within a varied and fractured hardware infrastructure landscape. At the same time, when defining the scope of the use cases that an ML platform should cover to promote adoption, one quickly comes to the realization that in many application domains like civil engineering the use of ML is relatively novel, meaning that application scenarios are currently limited and will possibly be expanded as new data, analysis techniques and opportunities for value creation become available.

Taking these considerations into account we set out to design the OCL platform based on a set of requirements to simultaneously support the following two *usability goals*:

- *Broad accessibility:* the platform should be *suitable for everybody* in order to enable practitioners with no coding, ML, hardware deployment or framework-specific expertise to explore their data, synthesize models, analyze results of training and inference;
- *Extensibility and specialization:* the platform should be useful for data science and ML researchers as well, both as working environment to conduct research itself, and as a means to quickly integrate research products in order to *extend it with the latest cutting-edge features* and *specialize it to novel emerging needs* within an application domain of interest.

In many cases, guaranteeing *broad accessibility* translates into automating and abstracting away tasks and implementation details that commonly arise in ML. In addition, the goal of accommodating *extensibility and specialization* emphasizes an architectural design focused on *modularity*, and *ease of development and integration of new features*. In the next sections we describe how these considerations play out in the design choices of the OCL platform in terms of UI/UX, data management, hardware resources provisioning and management, and compute jobs scheduling and orchestration.

A. UI/UX

In order to guarantee our first usability goal of enabling *broad accessibility*, we want the end user to be able to fully operate the platform from an intuitive UI, similarly to the majority of other commercial platforms. However, handling completely different tasks within a fixed user interface does not guarantee the best user experience: for example, exploring a civil infrastructure dataset composed of many annotated images organized in a hierarchical structure, is more efficient with the availability of specific navigation tools which exploit this particular structure, if compared to exploring a generic object detection dataset with standard tools. Therefore, *modularity and extensibility* of the framework are important also regarding the user interface itself.

Another important requirement of the first point is that, by controlling simple high level *budget* parameters (e.g. computing time to consume, resources to use, wanted optimization level, etc.), the system should help taking decisions on how to obtain the best model it can, based on the available data and under the requested budget constraints.

In addition, experienced users that require programmatic access to the platform are provided with a REST API that gives control on all parameters and enables programmatic interaction with the platform, which can be for instance necessary when clients need to integrate the system in existing data pipelines.

B. Data management

The primary and most important asset of any ML-based system is data. Performance of any kind of model directly depends on the availability and quality of the data which can be used for training. Therefore, in OCL we have adopted a data-centric approach in the organization of the functionalities of the platform. Particular care has been taken into offering to the user data exploration capabilities, and many internal decisions are automatically taken based on the type of data, its quantity and its quality. Thus, the architecture of the system allows very topic-specific customizations on data analysis and exploration tools, necessary to give the best user experience, while still serving as a base for completely different use cases (e.g. civil infrastructure object detection vs. NLP pipelines). This can be accomplished with a layered approach to data management, based on decoupling the component that manages generic data, common to all ML fields, from the components that deal with features specific of a single topic. Researchers can easily incorporate new topic-specific components: once they are automatically discovered and registered by the platform, the new features become available. These can include new customizations available from the UI, new operational steps in the data pipelines, or also capabilities for handling completely new data types and associated models. All these building on top of an abstraction layer providing management of a common base of generic data structures, letting the researcher be free to focus on the topic-specific functionalities. This common base layer is completely agnostic of the actual content of the data and provides common functionalities such as transparent access to heterogeneous data sources (e.g. cloud objects storage, shared filesystems, etc.), ingestion and transfer of data among the components of the system, and the extremely important aspect of user authentication, data security and compliance.

C. Hardware Resources Provisioning

An important but tedious task that any ML work needs to tackle is handling computing resources. As data becomes bigger and models become deeper and more complex, it becomes a must to make use of GPU resources, especially for computer vision or NLP tasks. These can, however, be considered as a *scarce* resource, due to their cost. Therefore, an important aspect to automate is the provisioning of computational resources. This brings both the benefit of optimal allocation of computing power among the pending tasks, providing also fairness of usage between users of the platform, together with potential cost savings when resources can be dynamically allocated (e.g. for cloud provisioning).

Due to the scarcity nature of GPUs, we specifically included in the design of OCL flexibility in the usage and management of computational resources. Goal is to be able to exploit all potential computational power we have access to: this includes cloud resources, potentially from different cloud vendors, but also more traditional HPC clusters. For example, a client with strong requirements on data security and locality could have already invested on an on-prem hardware solution; researchers could already have access to institutional shared HPC clusters; clients could have deals with cloud vendors managing their data. Flexibility in management of computational resources allows users to organize the hardware infrastructure in the best way it fits their use cases.

D. Jobs Scheduling

With *job* we refer to all those units of computation that are performed on given input data, such as preprocessing a dataset, or training a model, and so on. Given the flexibility offered in the potentially usable computing resources, having a strong abstraction around the generic *computing job* is necessary. For cloud-based solution, *containers* are powerful tools which bring the needed decoupling between the task that is being performed and the underlying deployment. However, they are not always available to be used in more traditional HPC clusters, thus requiring this abstraction to be implemented at code-level as well.

Once we have the definition of a generic computing job, next important step for the platform is to manage multiple jobs, potentially issued by different users. This task, performed by the *job scheduler*, includes:

- management of job queuing, taking into account priorities of the users and their access to computing resources;
- handling of failures, automatic restarts of jobs past wall-time limits, real-time feedback on running tasks;
- management of computational resources, differentiating between CPU and GPU-intensive jobs.

These components described above together form a platform built around extensibility and modularity, which is ready to offer a simple way for accommodating integration of prototypes and ML research assets. To begin with, an early access portal to R&D prototypes enables clients to test out the use of cutting-edge research features, even before they reach full maturity. This allows to provide feedback, speeds up the development-deployment cycle by favouring an agile development process of features before integration in more stable products. At the same time, it keeps researchers as free as possible from the constraints of stability and maturity that productization requires, which is, in our opinion, a key point in the ever-evolving landscape of AI development.

IV. Backend implementations in IBM OCL platform

In the previous section, we described the high level requirements we observed when taking the design decisions for our OCL platform. In the following we briefly introduce its architecture, sketched in Fig. 1.

The system is divided into two main logically separated parts: the *head* and the *hands*.

The *head* is the main center of operations. It includes all components which are not computationally-intensive, which deal with:

- management of users and authentication providers;

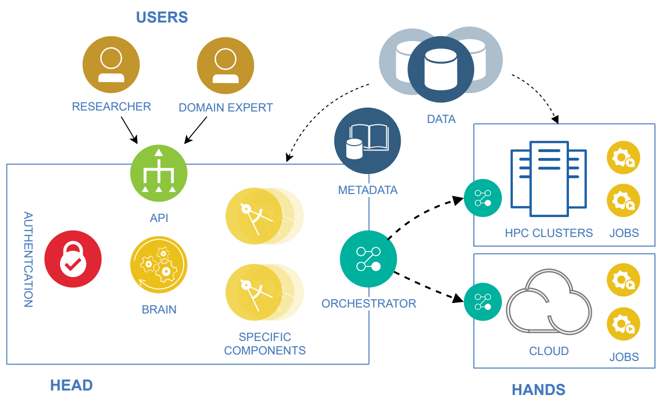


Fig. 1. Diagram sketching the OCL architecture.

- high-level management of projects and data;
- organization and access to the computing resources;
- scheduling of jobs;
- user interface and REST API.

The architecture is based on micro-services and is usually deployed in the cloud, for ensuring high availability. Internally, we devised a set of basic *entities* which are common to all AI workflows and are passed around the various services (among the others: *datasets*, *annotations*, *models*, *tasks*, *runs*, *users*, *projects*, etc.). Use case-specific assets can define specialized entities on top of the basic ones: examples in the civil engineering domain are *civil engineering asset*, *defect*, and so on. All components talk to each other by means of a centralized document-based NoSQL database, where metadata regarding all *entities* is stored. The database contains no data by itself, but just information on how to access it as pointers to external sources. A convenient API layer provides the means to then access the data by UUID or name, transparently from its actual physical location. All micro-services are stateless, to lower their complexity while ensuring scalability. Specifically, we have two sets of services:

- the *server*: the main entry point of the system, providing the REST API layer offering access to all functionalities and hosting the user interface;
- the *orchestrators*: multiple services handling the organization and scheduling of jobs.

With *hand* we refer instead to the logical part running computationally-intensive jobs. This component can have special requirements in terms of access to the hardware: it may need one or multiple GPUs, exclusive access to multiple cores, extensive amounts of RAM, or big local storage as cache. Each hand can be linked to different computational resources (being on cloud, or on other clusters). In OCL multiple *hands* can be registered simultaneously and it is up to the *head* to provide transparent access to them, to handle data transfers in and out and to manage the execution of jobs. In particular, jobs are scheduled in two stages:

- A high level *orchestrator* checks the presence of new jobs to run from a global queue; these are usually pipelines divided into stages (or *runs*). Based on the pipeline needs in terms of hardware and the accessibility of the user to the available computational resources, the orchestrator decides to which hand to submit the runs.
- A hand-specific orchestrator takes the runs from its own specific queue, prepares the container or environment in which to execute them (based on the type of deployment to use) and submits them for execution.

This last step depends on the actual computing environment the job is destined to run on. In case of cloud-based deployment, jobs are handled as *kubeflow pipelines* [19]. When dealing with HPC clusters, a specialized orchestrator connects to the login node, prepares the environment and data, and submits the job using the cluster-specific job queuing system (e.g. IBM Spectrum LSF). It is the duty of the orchestrator services to keep track of all hands and to report real-time information about running jobs to the head.

V. Case Study: Visual Inspection of Transport Infrastructure

In this section, we showcase the application of OCL to the visual inspection of transport infrastructures, taking the particular example of bridge inspection. This specific use case emphasizes the user- and data-centric aspects of OCL, in particular its modular design and extensibility that allowed it to integrate in a traditional ML workflow the management of customized application-specific assets and analysis tools in terms of specialized entities that are of interest to civil engineering users and domain expert.

Until recently bridge inspection was exclusively a manual process conducted by reliability engineers. Not only is this dangerous due to the complexity of the structure and the fact that some parts are hardly accessible, but it is also estimated to cost \$50B and \$2B man-hours annually. The main objective of the inspection is to assess the condition of an asset and determine whether repair or further maintenance operations are needed. Specifically, engineers make such decisions by analyzing the surfaces and detecting defects, such as cracks, spalling, rust or algae, and assessing their severity, relative to the defect size and location in the structure.

The advances in drone technology and its falling costs have recently pushed this laborious process of manual inspection progressively towards automation. Flying drones around a structure and using embedded high-resolution cameras to collect visual data from all angles not only speeds up the inspection process, but it also removes the human from potentially dangerous situations. In addition, thanks to the power of artificial intelligence capabilities, defects can be detected and localized with high precision automatically and presented to the reliability engineer for further analysis. Our OCL platform is at the center of providing experts with such advanced capabilities. By taking a data-centric approach, it focuses the user experience around the management of users' assets, providing specific features for:

- powerful data exploration of large datasets and high-resolution images with corresponding detected damage;

- detection, characterization and measurement of damages;
- reconstruction of infrastructure elements and localization of damages with automated image stitching;
- quick extraction of actionable results;
- empowering engineers and infrastructure managers to use pre-trained AI models for everyday inspections.

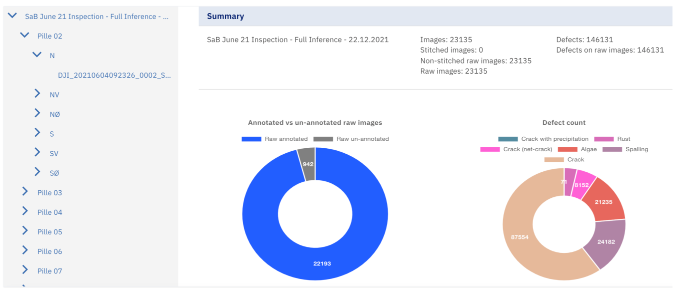
A compelling visual inspection use case is that of the Storebælt East suspension bridge, owned and operated by Sund & Bælt [20]. The bridge is part of the Great Belt Fixed Link [21] in Denmark. At 254 metres above sea level, the East Bridge has a length of 6790 metres and a free span of 1624 metres. A visual inspection of all 22 pillars was conducted in June 2021 with Matrix DJI 300 RTK drones. More than 23k high-resolution images were collected, where each image consists of $\approx$ 6k $\times$ 4k pixels (24M pixels). The OCL platform is used to store, analyze and visualize these images.

First, as seen in Fig. 2, general statistics are presented to the engineer. On the left side, a progressive view of the bridge that dives into the pillars, their orientations and corresponding images, allow the user to understand the hierarchy of the structure and quickly locate data of interest. On the right side, a summarized view of the images collected during drone inspection and the defects detected and classified during the inference of the AI model is provided. The defects are classified into categories specific to the use case (i.e., 6 here: crack, rust, spalling, algae, net-crack and crack with precipitation). Below the summary section, more detailed statistics are presented, such as average area (in m^2) per defect type and distribution of defects across the dataset.

Additionally, the platform provides filtering options to enable an intuitive and fast navigation through the defects. Specifically, one can select one or more defect categories and provide ranges of interest for attributes like area (in pixels or m^2), confidence score as resulting from the model inference or severity rating. This will return the set of images satisfying this criteria. The user can navigate through the selected images and visualize all defects detected in a specific image, as seen in Fig. 3. By hovering with the mouse over a defect, more details of interest are provided, such as type of defect, area in pixels and m^2, prediction score and severity level.

While the functionality described so far applies per individual image, it already provides great benefits to reliability engineers. However, decisions around repair and maintenance take into consideration the locations of defects in a bridge structure as well. Powered by image stitching algorithms, OCL provides an overall view of each bridge pillar. These stitched images are reconstructed automatically from the raw high resolution photos taken by the drone, by using image rectification and location reconstruction algorithms, and combine all defects detected in the individual images that were attached together, as shown in Fig. 4. It is always possible for the engineer to go from the overview image to the underlying raw photos for further detailed inspection (e.g., by clicking on the red highlighted area in Fig. 4). Intuitive navigation through the raw photos is possible by means of a minimap (shown in Fig. 5), which allows also to have an overview of where in the pillar structure each defect is located.

These stitched images are extremely big in size, as they preserve the full resolution of the already detailed raw photos. In order to deliver a smooth navigation, in the civil infrastructure component in OCL, images are divided in multi-resolution tiles and streamed to the browser from cloud object storage on demand. This is just one of the challenges that we had to overcome when dealing with such a big amount of data. Nevertheless, thanks to the flexible architectural design of the platform and the scalability of cloud-based solutions, we are able to deliver a smooth user experience in these complex use cases also.

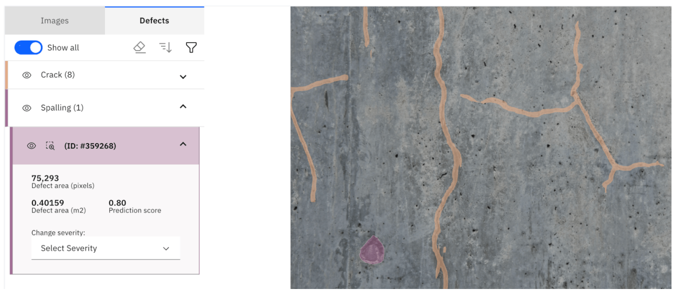

Summary

SaB June 21 Inspection - Full Inference - 22.12.2021

Images: 23135
Stitched images: 0
Non-stitched raw images: 23135
Raw images: 23135

Defects: 146131
Defects on raw images: 146131

Fig. 2. Hierarchical view of assets (left); summary of dataset and associated defects, as detected and classified into 6 categories during AI-model inference.



75,293
Defect area (pixels)

0.40159
Defect area (m2)

0.80
Prediction score

Change severity:
Select Severity

Fig. 3. Visualization of defects in specific image with associated confidence score as computed by the AI-model.

VI. Discussion

In this paper we presented the design of the IBM OCL platform, an end-to-end platform that aims to be task-agnostic and thereby providing a basis for standardization of ML workflows in technical domains that are still in the process of adopting and refining data science practices. In these domains, the main roadblock to adoption is the need to design a platform that can accommodate as much of the already established tasks and practices of current domain practitioners, while at the same time being flexible enough to adapt to evolving needs and growing application modes.

The way the IBM OCL platforms negotiates this tension between *broad accessibility* and *extensibility and specialization* is with a modular design thanks to which all components in the system can be flexibly reused and recombined for the

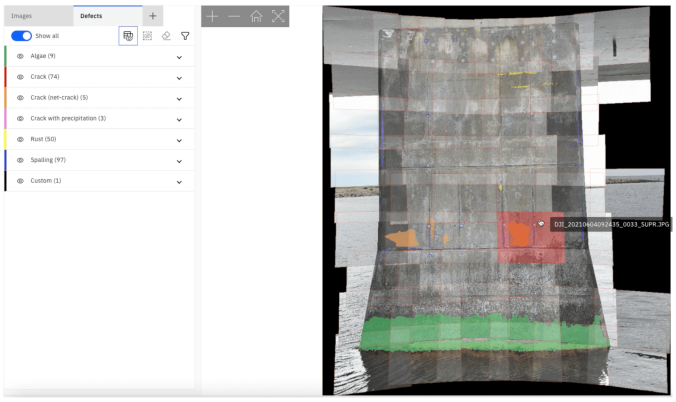


Fig. 4. Overall view of a bridge pillar, after image stitching algorithm was applied. Summary of defects is provided on the left and different categories are distinguished by color in the stitched image.

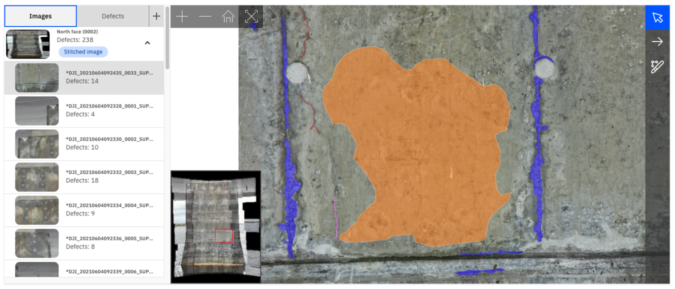


Fig. 5. Highlight on a defect; a minimap with the overview stitched image is showing the corresponding location of the defect on the full pillar.

different use cases, while remaining as decoupled as possible. By allowing for seamless swapping in/out of features and refactorings of capabilities, such modularity in design also has the added benefit of enabling the typical agile workflow of an R&D setting. In turn, doubling as an environment for R&D of novel ML assets that are bespoke to specific application domains endows the IBM OCL platforms with the advantages of a rapid development cycle affording it unprecedented customizability and potential for specialization in such domains. At the same time, because of the decoupled nature of its modular elements, OCL is able to mitigate the problem of overengineering of the structure, one of the major pitfalls that tends to happen when trying to develop a one-fits-all solution in advance.

Last but not least, we stipulate that the development of a shared data science platform that can be modularly extended and customized to evolving needs and application scenarios should be considered as a useful stepping stone towards a standardized, reliable and replicable use of data analytics that could serve as a basis to support robust and safe critical infrastructure, like for instance transport infrastructure in Europe, as illustrated by the bridge inspection use case that we examined.

Acknowledgment

The authors would like to thank Sund & Bælt Holding A/S for providing the bridge inspection data showcased in the case study of this paper, and in particular Finn Bormlund and Svend Gjerding. The authors would also like to thank the whole OCL team at IBM Research Zurich for their invaluable contribution in the development of the OCL platform.

References

[1] Harmonised transport infrastructure monitoring in europe for optimal maintenance and safety, Horizon 2020 EU grant. [Online]. Available: https://im-safe-project.eu/

[2] F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, and G. M. Sarné, “A trust-aware, self-organizing system for large-scale federations of utility computing infrastructures,” *Future Generation Computer Systems*, vol. 56, pp. 77–94, 2016, publisher: Elsevier.

[3] M. Caballer, S. Zala, Á. L. García, G. Moltó, P. O. Fernández, and M. Velten, “Orchestrating complex application architectures in heterogeneous clouds,” *Journal of Grid Computing*, vol. 16, no. 1, pp. 3–18, 2018, publisher: Springer.

[4] D. Salomoni, I. Campos, L. Gaido, J. M. de Lucas, P. Solagna, J. Gomes, L. Matyska, P. Fuhrman, M. Hardt, and G. Donvito, “INDIGO-DataCloud: A platform to facilitate seamless access to e-infrastructures,” *Journal of Grid Computing*, vol. 16, no. 3, pp. 381–408, 2018, publisher: Springer.

[5] S. Pallickara, J. Ekanayake, and G. Fox, “Granules: A lightweight, streaming runtime for cloud computing with support, for map-reduce,” in *2009 IEEE International Conference on Cluster Computing and Workshops*. IEEE, 2009, pp. 1–10.

[6] F. O. Catak and M. E. Balaban, “CloudSVM: training an SVM classifier in cloud computing systems,” in *Joint International Conference on Pervasive Computing and the Networked World*. Springer, 2012, pp. 57–68.

[7] K. Li, C. Gibson, D. Ho, Q. Zhou, J. Kim, O. Buhisi, D. E. Brown, and M. Gerber, “Assessment of machine learning algorithms in cloud computing frameworks,” in *2013 IEEE Systems and Information Engineering Design Symposium*. IEEE, 2013, pp. 98–103.

[8] X. Lin, P. Wang, and B. Wu, “Log analysis in cloud computing environment with Hadoop and Spark,” in *2013 5th IEEE International Conference on Broadband Network & Multimedia Technology*. IEEE, 2013, pp. 273–276.

[9] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, and S. Owen, “Mllib: Machine learning in apache spark,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016, publisher: JMLR. org.

[10] G. Preethi, P. V. Krishna, M. S. Obaidat, V. Saritha, and S. Yenduri, “Application of deep learning to sentiment analysis for recommender system on cloud,” in *2017 International conference on computer, information and telecommunication systems (CITS)*. IEEE, 2017, pp. 93–97.

[11] V. Ishakian, V. Muthusamy, and A. Slominski, “Serving deep learning models in a serverless platform,” in *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2018, pp. 257–262.

[12] M. Ribeiro, K. Grolinger, and M. A. Capretz, “Mlaas: Machine learning as a service,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 896–902.

[13] S. Chan, T. Stone, K. P. Szeto, and K. H. Chan, “Predictionio: a distributed machine learning server for practical software development,” in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 2493–2496.

[14] “IBM Cloud Pak for Data as a Service.” [Online]. Available: https://www.ibm.com/products/cloud-pak-for-data/as-a-service

[15] “Google Cloud AI.” [Online]. Available: https://cloud.google.com/products/ai

[16] “Azure Machine Learning.” [Online]. Available: https://azure.microsoft.com/en-us/services/machine-learning/

[17] “Amazon SageMaker.” [Online]. Available: https://aws.amazon.com/sagemaker/

[18] “H2O.ai.” [Online]. Available: https://www.h2o.ai

[19] E. Bisong, “Kubeflow and kubeflow pipelines,” in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 2019, pp. 671–685.

[20] “Sund & bælt.” [Online]. Available: https://sundogbaelt.dk/en

[21] “Great belt fixed link.” [Online]. Available: https://en.wikipedia.org/wiki/Great_Belt_Fixed_Link