Digital Object Identifier 10.1109/ACCESS.2017.DOI

A cloud-based framework for machine learning workloads and applications

ÁLVARO LÓPEZ GARCÍA[1], JESÚS MARCO DE LUCAS[1], MARICA ANTONACCI[3], WOLFGANG ZU CASTELL[9][10], MARIO DAVID[2], MARCUS HARDT[6], LARA LLORET IGLESIAS[1], GERMÁN MOLTÓ[7], MARCIN PLOCIENNIK[4], VIET TRAN[8], ANDY S. ALIC[7], MIGUEL CABALLER[7], ISABEL CAMPOS PLASENCIA[1], ALESSANDRO COSTANTINI[4], STEFAN DLUGOLINSKY[8] DOINA CRISTINA DUMA[4], GIACINTO DONVITO[3], JORGE GOMES[2], IGNACIO HEREDIA CACHA[1], KEIICHI ITO[9], VALENTIN Y. KOZLOV[6], GIANG NGUYEN[8], PABLO ORVIZ FERNÁNDEZ[1], ZDENĚK ŠUSTR[11], PAWEL WOLNIEWICZ[4]

[1]IFCA (CSIC-UC), Santander, Spain
[2]Laboratory of Instrumentation and Experimental Particle Physics (LIP), Lisbon, Portugal
[3]INFN Bari, Bari, Italy
[4]INFN CNAF, Bologna, Italy
[5]Poznan Supercomputing and Networking Center, IBCh PAS, Poznan, Poland
[6]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[7]Instituto de Instrumentación para Imagen Molecular (I3M), CSIC - Universitat Politècnica de València, Valencia, Spain
[8]Institute of Informatics, Slovak Academy of Sciences (IISAS), Bratislava, Slovakia
[9]Helmholtz Zentrum München, Deutsches Forschungszentrum für Gesundheit
[10]Department of Mathematics, Technische Universität München, Germany
[11]CESNET, Prague, Czech Republic

Corresponding author: Álvaro López García (e-mail: aloga@ifca.unican.es).

This work has been supported by the project DEEP-Hybrid-DataCloud "Designing and Enabling E-infrastructures for intensive Processing in a Hybrid DataCloud" that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 777435.

ABSTRACT In this paper we propose a distributed architecture to provide machine learning practitioners with a set of tools and cloud services that cover the whole machine learning development cycle: ranging from the models creation, training, validation and testing to the models serving as a service, sharing and publication.

In such respect, the DEEP-Hybrid-DataCloud framework allows transparent access to existing e-Infrastructures, effectively exploiting distributed resources for the most compute-intensive tasks coming from the machine learning development cycle. Moreover, it provides scientists with a set of Cloud-oriented services to make their models publicly available, by adopting a serverless architecture and a DevOps approach, allowing an easy share, publish and deploy of the developed models.

INDEX TERMS Cloud Computing, Computers and information processing, Distributed Computing, Machine Learning, Deep Learning, Serverless architectures.

I. INTRODUCTION

The impact of emerging computing techniques together with an increasing dimension of large datasets and the availability of more and more performing and accessible computing resources is transforming many research areas. This opens the door to new opportunities to tackle unprecedented research challenges. Over the last decade there has been a boost on the usage of machine learning techniques in most of the research areas, and recently it has even improved with the adoption of deep learning techniques, e.g. LeCun et al. [1]. Although the basic components of the techniques are well known, recent advances arouse the interest from the scientific community towards this area, and it has already become a state-of-the-art technology in many fields, from computer vision to speech recognition.

The performance increase in the existing computing technologies and the availability of specialized computing devices played an important role in the advent of deep learning. In particular, the availability of more efficient and affordable Graphical Processing Units (GPU), more suited than Central Processing Units (CPU) for the kind of workloads that characterize the deep learning training tasks, allowed the

development of the so called neural networks in an extremely efficient way. Moreover, hardware vendors are actually developing new architectures with specific focus on neural networks and deep learning, implementing specific features for these kind of workloads, such as specialized cores to accelerate large matrix operations of specialized devices like Tensor Processing Units (TPUs) developed by Google [2].

In parallel, current technological advancements related to Cloud computing have been adopted by many research communities to provision compute and storage resources, due to the well known advantages of flexibility, elasticity and economy that it can offer [3]. Cloud computing is now considered a common computing model among scientists and, as a matter of fact, large scale distributed and pan-European e-Infrastructures are offering Cloud computing services [4] focused on scientific users. However, even if Cloud and its related services had acquired an increasing interest among the scientific computing ecosystem, the adoption of Cloud computing infrastructures is still limited. In fact, the Infrastructure as a Service (IaaS) offers are considered too complex and heterogeneous [5] to be efficiently exploited by end users.

Consequently, a step ahead from low-level IaaS offerings is needed to effectively exploit the potential of the Cloud computing model. It is required to provide users with high level tools, like Platform and Software as a Service stacks (PaaS and SaaS respectively), able to implement the needed flexibility to deliver solutions that can be tailored on purpose to satisfy a given communities' needs, providing also transparent exploitation of the infrastructure resources [6].

In such respect, Machine learning "as-a-Service" can clearly be considered as one of the most demanded services when large-scale computing infrastructures are adopted. With the increasing availability of the amount of data, the machine learning pipelines for large-scale learning tasks can be exploited to provide an additional level of challenge. With the need for a proper design of the learning task at hand, additional tasks result in the need to organize large-scale data. Furthermore, provision of necessary computing power and storage capacity as well as orchestration of various infrastructure components at different places have to be managed, since large-scale data is commonly distributed.

It is a common opinion that such tasks cannot be managed by all users. Most of them, in fact, have a domain knowledge in specific fields, lacking technological or infrastructure knowledge. Therefore, a support by the infrastructure layer must break down the complexity of the task and allow scientists to focus on their respective activities, i.e. modelling of the problem, evaluating and interpreting the results of the algorithms. Starting from the above assumption, understanding the requirements of the user communities becomes mandatory for infrastructure as well as resource and technology providers in order to combine different technologies and provide easy-to-use services for the end users.

To address this goal, the "Designing and Enabling E-infrastructures for intensive Processing in Hybrid Data Clouds" (DEEP-Hybrid-DataCloud)[1] project (hereinafter referred as DEEP) developed and implemented a framework to enable the transparent training, sharing, and serving of machine learning models over distributed and hybrid e-Infrastructures. In this work we present the DEEP approach and architecture, showcasing how it enables researchers to develop and train their machine learning models in a distributed manner. Using the DEEP framework, any data scientist will be able to take all the steps of the learning cycle, exploiting the whole potential of distributed e-Infrastructures. The DEEP project contributions and highlights can be summarized as follows:

- We provide transparent access to last generation computing and storage distributed e-Infrastructures.
- We have developed a standardized API for machine learning models, allowing to expose their functionality with known semantics, regardless of the underlying model.
- We provide an easy path to deploy the developed modules as services, following a serverless approach and applying DevOps techniques, easing the transition from development to production.
- We have build tools that allow the effective sharing of the developed machine learning models and their associated metadata, fostering a knowledge exchange in the machine learning area.

The remainder of the paper is structured as follows. Section II describes the related work in the area. In Section III we elaborate on the motivation that is driving this development. In Section IV we describe the proposed framework and reference architecture that is being developed within the DEEP project. In Section V we provide a brief description of some of the early results obtained from different use cases exploiting the DEEP framework. Closing this work, Section VI presents the conclusions and potential future work.

II. RELATED WORK REGARDING MACHINE LEARNING PLATFORMS

Although machine learning systems have obtained limited attention by research works and academia, there are still similar cloud-based initiatives and solutions, aimed to lower the entry barrier for users that want to leverage usage of machine learning models.

Abdelaziz et al. developed a cloud based model to provide machine learning services based on cloud computing with focus on healthcare services [7]. In their work they provide an algorithm to improve the virtual machine selection to optimize the performance of some machine learning based healthcare applications. However, this work focuses on the selection of VMs based on an optimization model, rather than providing a comprehensive framework for the development machine learning applications over cloud infrastructures and it is too focused on specific healthcare applications. Also focused on specific applications, Wang et al. developed cloud

based framework to tackle the resource allocation problem for wireless communications.

Resource allocation is a complex task that should not be performed directly by users, unless they have the knowledge and they are skilled to do so. This is a problem that has been tackled beforehand by the authors [3], [5], [6]. In this line, Messina et al. [8] developed a Hypertrust, a framework for trustworthy resource allocation problems in utility computing and cloud computing, providing a decentralized solution to support trusted resource finding and allocation into federated cloud infrastructures.

Li et al. present in their work [9], [10] the parameter server. This project is a framework to solve distributed machine learning problems, where globally shared parameters can be used as local sparse vectors or matrices to perform linear algebra operations with local training data. Kraska et al. developed MLBase [11], providing a simple declarative way to specify machine learning tasks so that they can be applied to a given data set exploiting an optimized runtime. Systems like these present an advancement on scalable machine learning, however users are tied to specific frameworks and languages, rather than being able to choose their preferred framework from the existing offer [12].

There is a vast number of authors that have performed a significant quantity of studies [13]–[20] about the cloud suitability and its associated performance for the development and training of specific machine learning methods. However, these studies are mostly exploiting infrastructure resources or big data and analytics tools that are deployed on top of those resources. The work presented in this paper goes forward, providing a complete framework for the training, sharing and deployment of machine learning applications, exploiting cloud-based services.

In [21], Ishakian et al. provide an assessment of the feasibility of serving deep learning models exploiting serverless frameworks, where a serverless function provides an image classification engine.

Rivero, Grolinger and Capretz proposed an architecture for generic Machine Learning as a Service (MLaaS) [22], together with an initial open source implementation. In their work, the MLaaS service relied on different machine learning algorithms to be specifically implemented for that particular service, as well as data processing tools and auxiliary processes. Similarly, Chan et al. [23] presented another architecture for MLaaS named PredictionIO. They integrated several machine learning models into a prediction service, accessible through an API and a Graphical User Interface.

Li et al. [24] presented a design and implementation of a scalable feature computing engine and store. Their framework provided with tools to manage and train a hierarchy of models as a single logical entity, together with an automated deployment system and scalable real-time serving service. Baldominos et al. [25] presented an online service for real time big data analysis based on machine learning.

Besides, some open source tools do exist as well. Polyaxon [26] aims to provide an enterprise-grade platform for agile, reproducible, and scalable machine learning. Polyaxon aims to be deployed on premises, on top of Kubernetes, and it provides tools for building, training, and monitoring large scale deep learning applications. Despite being an on premises solution, focused on enterprise applications, Polyaxon does not provide sharing or publishing capabilities.

Other existing platforms have a clearly targeted user audience. Kipoi is a model repository (*model zoo*) for genomics [27] that allows sharing and exchanging models. It covers key predictive tasks in genomics. Kipoi provides data handling, Python and R APIs, leveraging online tools and services (such as GitHub and CircleCI for storing models and performing nightly tests, respectively). Similarly, ModelHub.ai [28] is a repository of deep learning models pre-trained for a wide variety of medical applications. ModelHub.ai provides a framework in which users can plug their model, as well as pre- and post-processing code. The framework provides a runtime environment and helper functions (like image conversion) as well as APIs to access the model.

DLHub [29] puts the focus on publishing, sharing, and reusing machine learning models and their associated data, capturing its provenance and providing credit to authors. DLHub also provides a model serving framework for executing inference and other tasks over arbitrary data. DLHub has been exploited by several research projects with success, such as [30], [31].

A common fact of Kipoi, ModelHub.ai and DLHub is is their science centric vision, with strong focus on fostering open source and positioning themselves as machine learning repositories that contribute to the reproducibility of scientific computation.

Moreover, several commercial platforms exist, providing different levels of functionality. In this regard, Yao et al. [32] performed an evaluation of the most popular commercial machine learning platforms at the time of writing the article (2017). They focused exclusively on the classification performance and not on any other aspects and phases of the machine learning development cycle. They found that, as expected, with more user control over the platform there are more performance gains, as well as a higher underperforming risk due to bad configuration settings. A brief overview of those platforms and services is given in what follows.

- Google Cloud AI [33] and Google Hub AI [34] with their large catalogue of services including AutoML, Computer Vision, Video Intelligence, Speech-to-Text and Natural Language.
- Azure Machine Learning Service [35] and Studio [36] which allow tracking and auto-scaling experiments to GPU clusters.
- Model Asset eXchange [37] from IBM which offers a range of services including computer vision and audio classification.
- AWS Marketplace - Machine Learning [38] where users and companies can develop their models and sell them as services running on AWS. Their extensive catalogue includes hundreds of services.

- H2O [39] tools cover financial services, insurance, health care, manufacturing, marketing, retail and telecommunications.
- Floydhub [40] aims at accelerating Deep Learning development by relieving the user from the need to take care of infrastructure, version control, environments and deployment, based on exploiting resources from commercial clouds.
- Deep Cognition [41] offers users to develop a model with a visual editor, train and deploy models using a REST API.

For the sake of completeness other similar platforms include BigML [42], ParallelDots [43], DeepAI [44], MLjar [45], Datarobot [46], Valohai [47], Vize [48], Seldon [49], PredictronLabs [50], Gavagai [51] and Cloudsignt [52].

Most of the previously mentioned commercial platforms include the ability for users to access resources for developing, training, and testing models, as well as providing a unified API to access their models. Others just provide Deep Learning services that users can access through APIs.

Whilst many of the described services focus their activity on a subset of the phases (mostly sharing and serving) of the machine learning cycle (Figure 1), our work proposes a comprehensive framework that comprises a set of services and tools aimed at covering all the relevant phases of the development and deployment of a machine learning application: from the model creation (or update) to the serving and sharing phases, passing trough the training (over distributed computing infrastructures), testing and evaluation phases. One distinct feature of the DEEP framework is its ability to exploit computing resources from production science e-Infrastructures, commercial clouds or on-premises computing clusters. Moreover, the framework is designed so that users are not tied to a specific learning tool or library, being able to chose the framwework of their choice (e.g. Keras, TensorFlow, Scikit-learn, PyTorch, etc [12])).

III. MOTIVATION FOR A MACHINE LEARNING AS A SERVICE PLATFORM

The typical machine learning development cycle as proposed by Miao et al. [53] (adapted in Figure 1) starts with a set of data which is split into training data and test data. After creating a model, normally starting from a reference one, this model is then trained over the corresponding data. Afterwards, the test data is used to estimate the quality of the model with respect to its learning tasks. The latter can typically be related to either classification or regression. After the model has been trained and its accuracy has been evaluated as satisfactory (this may imply repeating the create/update, train and evaluate loop several times), the readily trained model can be used in applications for classification or regression.

From a computing perspective the various steps of this learning cycle are not all equal, and the cycle can be repeated several times (also when new data is available). While training is commonly computing intensive and relying on large amounts of data being available, validation and classification are much cheaper in terms of resources. An extreme case of moving through the cycle very often is given by online-learning algorithms. Similarly, data requirements for the various steps within the cycle are different: Whilst some algorithms need a large amount of data being available right from the beginning at the training phase, others work through the data in batches. Finally, depending on the learning algorithm being used, various ways of distributed computing and computing architecture are needed.

Moving ahead from the infrastructure and resources point of view, different use cases rely on different levels of knowledge in machine learning technology on the side of the user communities. The users can therefore be classified in three categories depending on their machine learning knowledge and the use they want to make of the platform:

- Basic users just want to use the tools (i.e. models) that have already been developed and apply them to their data. They therefore need almost no machine learning knowledge.
- Intermediate users want to retrain the available tools to perform the same task but fine tuning them to their own data. They still may work without detailed knowledge on modelling of machine learning problems, but typically they need basic programming skills to prepare their own data into the appropriate format. Nevertheless, they can re-use the knowledge that is captured in a trained network and tune it by re-training the network on their own dataset (transfer learning). From the point of view of infrastructure providers, this variant relies on a powerful computing infrastructure to execute re-training and host datasets of users.
- Advanced users, instead, are those developing their own machine learning tools. They therefore need to be competent in machine learning. They will then need to design their own neural network architecture, potentially re-using parts of the code from other tools. From the infrastructure point of view this is the most resource demanding case.

Machine learning specialists are normally keen on covering the create/update, training, and model evaluation phases. Often, the final phase of sharing and publication of a model is either neglected or it is assumed that enough knowledge exists elsewhere to perform this action. Publication of models is, however, a critical step, because only public models allow the effective exploitation of their functionality by others (i.e. external communities and users can benefit from such models).

Publication as suggested in the DEEP project is done by offering models as a service. Sharing and reusing should be key in any machine learning system, as this is a driving factor for disseminating knowledge in science and fostering collaboration across research areas.

During the DEEP project activities, carried out with the support of different scientific communities [54] belonging to the DEEP consortium, constraints, gaps and expected gains have been identified thanks to the feedback provided by the

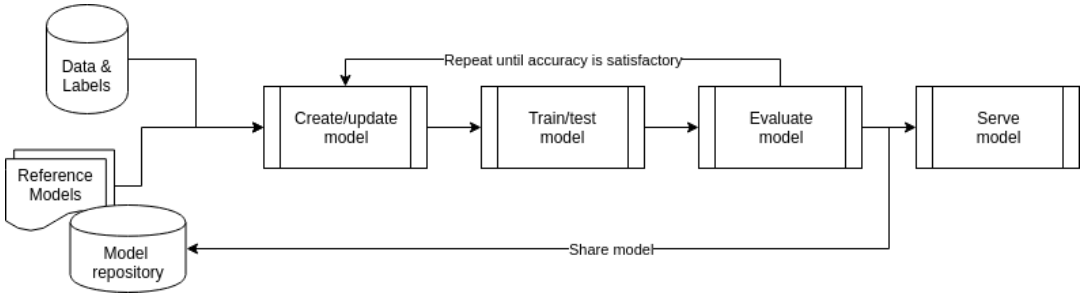

Figure 1. Learning cycle for a machine learning application.

users and related to possible limitations in their current work and chances of improvements.

In order to perform an efficient training of a given model, **access to computing power and storage** resources is needed. Even if some communities can have direct access to the resources, this is generally not the case. Researchers need to leverage distributed e-Infrastructures such as those provided by the European Open Science Cloud ecosystem. Although these facilities can compensate the lack of resources problem to an extent, they impose other kind of challenges: users need to deal with infrastructure details, that sometimes are complex and hamper their research. Cloud technologies allow to avoid these difficulties enabling to address machine learning tasks in new ways fostering a general push in the field of machine learning applications.

The use of e-Infrastructures and the cloud computing paradigm, however, comes with limitations as well as the requirement for users to understand usage and management of such technologies. Even if restrictions related to the structure of specific resource centres require detailed expert knowledge, a certain degree of abstraction must be provided so that users offload the infrastructure interactions to platform level tools.

Once a user has designed, trained, and validated a model on a given dataset, it can be shared so that other interested users can exploit this trained model on their data. This sharing should be considered as a highly important key task in the scientific machine learning life cycle due to the fact that it is the only way to effectively **enable collaboration** and break knowledge silos. Once the model is shared publicly, it is possible to include these assets into the scholarly process leading to a publication. However, there is no common way to share and make public a developed model without creating a heterogeneous and fragmented ecosystem and preventing the reuse of a model: users may need to face different methods and tools for different published models.

Modern science is performed in a distributed and collaborative way, therefore **serving** a developed model over the network, either to close collaborators or to a more general audience, has to be considered as a key task that should become part of the learning cycle applied to science [55] Sharing a model over the network requires technological knowledge (API implementation, security, etc.) that many scientific users may not have.

The DEEP framework is offering solutions aimed at covering all the aforementioned aspects. The framework is providing also an environment where to improve reproducibility of experiments carried out within the framework, as scientists can provide a running endpoint where their results can be tested and validated, linked with all the assets that take part on the experiment: dataset, model's source code, model configuration and weights, etc. Reproducibility is defined as the ability of a researcher to duplicate the results of a prior study using the same data and means as were used by the original investigator. The scientific community, in fact, has gained awareness about the importance of following some basic reproducibility principles on their research [56], increasing thus the rigour and quality of scientific outputs and leading to greater trust in science. These principles do not only include the sharing of code and data but also publishing their provenance, and storing them together with a meaningful set of metadata and execution instructions.

Developing a comprehensive, reliable and secure source code is of paramount importance in every software development project. For such reason, the DEEP project implemented for any aspect of its software life cycles a **Software Quality Assurance** (SQA) processes acting as an enabler for accomplishing the aforementioned overarching goals of machine learning models' re-use and share.

IV. THE DEEP-HYBRID-DATACLOUD FRAMEWORK

The DEEP framework is designed for allowing scientists to develop machine learning and deep learning models on distributed e-Infrastructures. It provides a comprehensive framework that covers the whole machine learning development cycle, as explained in Section III. For this different high level components have been designed, depicted in Figure 2 and described in what follows:

- The **DEEP Open Catalogue**[2], a marketplace where the users and user communities can browse, share, store and download ready to use machine learning and deep learning modules. This includes working and ready to use applications (such as image classification tools, network

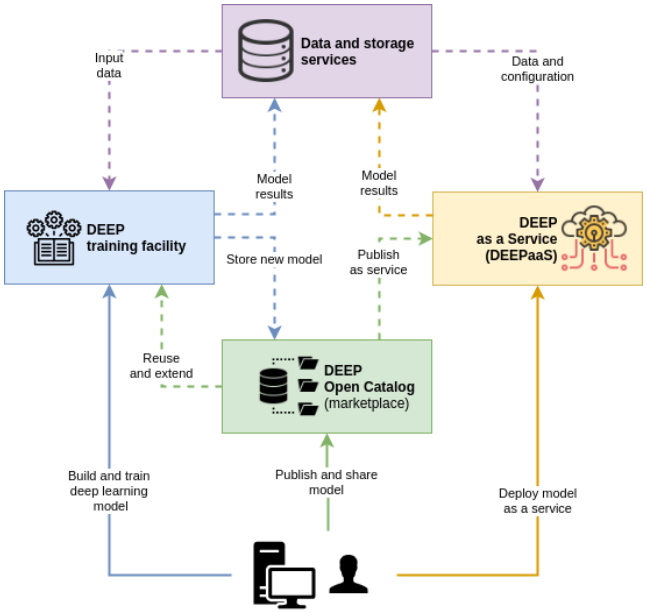

Figure 2. DEEP high level overview.

anomaly detection engines, etc.) as well as more general purpose and generic models (such as image segmentation or image super resolution tools). Moreover, the marketplace also comprises additional components, like complex application topologies (for example for data ingestion), and all the model and application associated metadata.

- The **DEEP learning facility**[3] that coordinates and orchestrates the overall model training, testing and evaluation, choosing the appropriate Cloud resources according to the computing and storage resources.
- The **DEEP as a Service**[4] solution (DEEPaaS), that provides a way to deploy and serve an already trained model, stored in the catalogue, as a service, to provide third users with the model functionality and acquired knowledge.
- The **storage and data services** where the user data, training and validation results, as well as any other data assets are stored.

A. DEEP ARCHITECTURE

The detailed DEEP architecture is depicted in Figure 3. The framework is divided into the main blocks already described in Figure 2.

In the proposed approach a user may either start developing a model from scratch or reuse one of the existing models that are published in the DEEP Open Catalog. With both approaches the model is encapsulated inside a Docker container along with all the required libraries and the DEEPaaS API [57] component. The DEEPaaS API is a REST endpoint, based on the OpenAPI specification [58] that provides a thin layer over a machine learning model, thus ensuring consistency across all the modules that are published into the marketplace. Moreover, this API has bindings to allow seamless execution and serving, exploiting different serverless frameworks such as OpenWhisk [59]. After the model is encapsulated into a Docker container, all the user interaction with the model, either locally or remotely, is performed through this component via a convenient simple web interface that is available whenever the model is loaded.

Once the model has been initially developed on the scientist's workstation, the user will submit it for training over a distributed e-Infrastructure. To this aim we leverage the INDIGO-DataCloud PaaS Orchestrator [6] together with the Infrastructure Manager (IM) [60]. With these two components it is possible to submit a training task to an existing computing cluster registered in the orchestrator database (such as Apache Mesos or an HPC system) or to deploy any complex application topology from scratch on top of any bare IaaS deployment: the orchestrator will dispatch the tasks to the most suited compute environment depending on the requirements declared in the user request, like for instance number of GPUs. Both approaches allow the user workload to access the existing distributed storage systems (either on premises or commercial) to obtain access to the data repository used for the training. The topologies and workloads to be deployed are described using the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) standard [61], and are handled transparently for the users that do not need to deal with the TOSCA details. To also provide advanced users with a tool to graphically build and deploy application topologies, we are adopting and extending Alien4Cloud [62], an open source tool that simplifies the composition of applications for the Cloud.

After the deployment is made, the training and monitoring, as well as the testing and evaluation tasks are performed by interacting with the DEEPaaS API, using the aforementioned REST endpoint or the simple web interface provided via Swagger [58]. Again, this provides consistency for the users, since the interaction with the model is the same in their local workstations or when being executed on a distributed e-Infrastructure.

If the evaluation phase is successful, the user will store all the model components (like the trained model, training weights, hyperparameters) in the storage systems, and will submit a request to include the model in the DEEP Open Catalog. To maintain a coherent marketplace, DEEP has defined a JSON schema [63]. This allows unified characterization and representation of the modules in the Open Catalog. The DEEP schema [64] includes both basic –author, license, date, name, description– and more specific metadata relevant for the specific model execution and deployment. The metadata is maintained in the user space, as an individual file under the model's repository code, so that the model's developer is fully responsible and comprehensively controls what information is being rendered and displayed in the

[3] https://learn.deep-hybrid-datacloud.eu

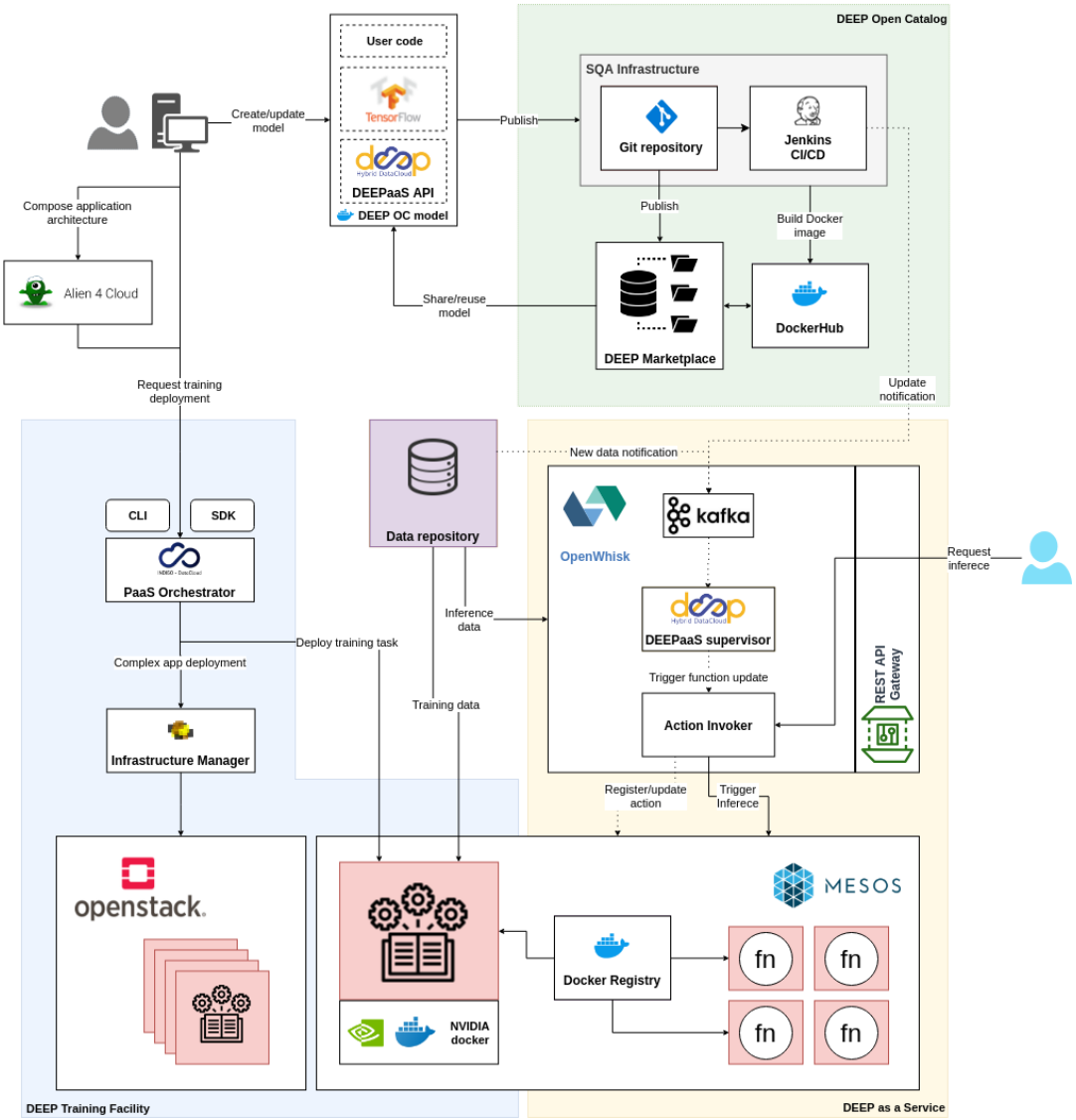

Figure 3. The DEEP detailed architecture (dotted lines represent automated notifications and transitions based on them).

DEEP marketplace. The inclusion request within the Open Catalog can be performed at any stage in the development phase. It is granted only after the successful completion of a quality-aware DevOps pipeline. The pipeline covers all tasks related with the testing, delivery, and deployment of the new model into the DEEP as a Service (DEEPaaS) component.

DEEPaaS is based on a serverless architecture where the user model is encapsulated as a function (or action), providing the model's final functionality (e.g. prediction, inference, etc.). The framework we have built is based on OpenWhisk, using Mesos as the execution framework for the user defined actions (i.e. the user models). A dedicated supervisor is deployed on DEEPaaS as well, to react on the updates of the existing models. DEEPaaS allows directly triggering an inference task, as well as reactions to automated classification tasks based on storage events that are published by the storage systems. By leveraging OpenWhisk as the serving layer the proposed solution can rely on it for the management of horizontal scalability, thus providing users with a convenient solution to deploy and serve machine learning models.

Our system is designed for extensibility, therefore taking great care in designing a framework which can be updated easily and where any component can be replaced with a new one. We follow a so called "share-nothing architecture". We have chosen to utilize Docker containers for executing the user applications. This makes it possible to deliver customized environments tailored to the user needs, and facilitating maximal portability of the runtime environments.

Besides, by using the DEEPaaS API component to serve the model functionality, a module can be deployed on any computing infrastructure. The supported platforms range from user workstations, computing servers all the way to

HPC systems, and Container Orchestration Engines (such as Kubernetes or Mesos) or serverless frameworks such as OpenWhisk. This way, the very same container image can be executed regardless of the underlying computing infrastructure. This ensures the highest possible level of portability and results in reproducibility. In cases in which Docker is not available, it is possible to execute the model by using the udocker [65] tool.

B. ENSURING SOFTWARE QUALITY AND INSTANT READINESS OF MACHINE LEARNING MODULES

The list of modules that are reachable through the Open Catalog is generated by an automated process that ensures their viability in terms of exploitation. Module owners only need to care about the development process, and incorporate new features that the automation system will receive as input. This generates the essential assets to make them transparently available in the production DEEPaaS service. Along the way, new features and/or metadata are validated to avoid disruptions, software is encapsulated in containers and published in online repositories, and entries in the Open Catalog are updated to reflect changes. Figure 4 outlines the series of automated transitions that start from the initial commit of a change in the model's metadata or source code, and eventually result in the public exploitation of the associated model application. In the following, we describe this process in detail.

Machine learning developers are encouraged to adopt software development best practices [66], which are applied with the guidance of the SQA team, in a DevOps-like environment. Thus, relevant changes in the source code are automatically validated through the execution of a series of static analysis tests, which represent the Continuous Integration (CI) stage. The diversity and scope of such tests differs among the existing modules in the project, tailored to the developer needs. Nevertheless, there exist two essential requirements, i.e. compliance with standard code styles and security analysis, that are common to all modules.

As a consequence the resulting modules can be distributed more safely and shared within the community, with the additional characteristic of being made up of readable and understandable source code. The latter is particularly beneficial for the sustainability of the modules as it positively affects their maintainability and usability by the external users. Additionally, whenever the change involves modification of the metadata, the CI stage includes a supplementary check to validate its structure according to the defined DEEP JSON schema. Once all these tests pass successfully, the new model's data will appear in the Open Catalog.

The DevOps scenario combines the above described CI implementation with a subsequent software delivery stage. Since DEEP modules are containerized applications, the DevOps pipeline automatically packages each updated model version into a container image, given successful validation of the changes. The resulting image is then published to online production repositories, readily available for consumption.

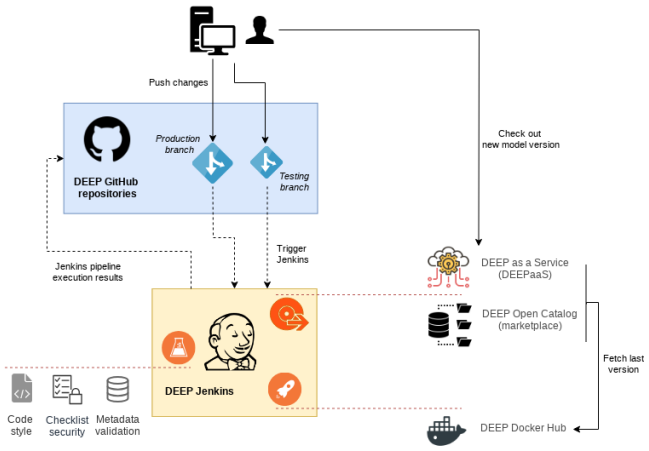

Figure 4. DevOps implementation covering the testing, delivery and deployment of DEEP model applications (dotted lines represent automated transitions).

Following the project's software best practices, the modules' coding workflow uses a branching approach where production or stable versions are separated from the testing ones. New container images are created based on changes made in those branches. The latest versions for testing and production are seamlessly pushed to the container repositories, and hence accessible through the Open Catalog. The provision of Docker images that contain the latest stable versions of a module is crucial for the appropriate exploitation of the modules, i.e. not only changes in the model's code repository trigger new Docker image builds, but also new released versions of DEEPaaS. Hence, the system ensures that no disruption is induced in the model's performance.

The final stage deploys the new version of the model, exposing it to the outside world through DEEPaaS. The SQA system leverages the OpenWhisk event-driven capabilities — through a Kafka service— to acknowledge the presence of a new model version. OpenWhisk will then process the request for a container's image replacement by either superseding the running one with the newly released or by adding a new model, according to the previous existence of the given model in the DEEPaaS offerings. The main outcome of the described Continuous Deployment (CD) implementation is the immediate availability of a new model version into the production DEEPaaS system. Similarly to the CI case, this stage also covers the processing and publication of the metadata. If changes are detected in the metadata file, those will be reflected in the DEEP marketplace. Through a separate pipeline, the SQA system controls which models need to be present in the marketplace. The setup relies on an index file containing the full list of supported entries, and rebuilds the marketplace accordingly. Only if a given model is present in the marketplace, this step wil be carried out in the CD stage.

V. USE CASES

This section illustrates two different use cases that have adopted the DEEP-Hybrid-DataCloud solutions to train an deliver deep learning services on the Cloud. These use cases have exploited the resources available in the DEEP-Hybrid-DataCloud preview testbed, but the framework can exploit other production computing infrastructures, including commercial cloud providers.

A. PHYTOPLANKTON CLASSIFICATION ENGINE

The Belgian LifeWatch Observatory the Flanders Marine Institute (VLIZ), part of the LifeWatch ERIC, harvest monthly samples of phytoplankton at nine sampling stations in the Belgian part of the North Sea. Phytoplankton [67] is a key part of oceans and an indicator of the general state of the marine ecosystem. Changes in the phytoplankton community have an affect on a wide range of other marine species. As such, monitoring plankton communities provides valuable information to assess ecosystem health and to research ecosystem functioning. The collected data is then manually processed by a taxonomy expert in order to classify the species present in the samples, requiring from the time-consuming task of identifying and counting the different specimens.

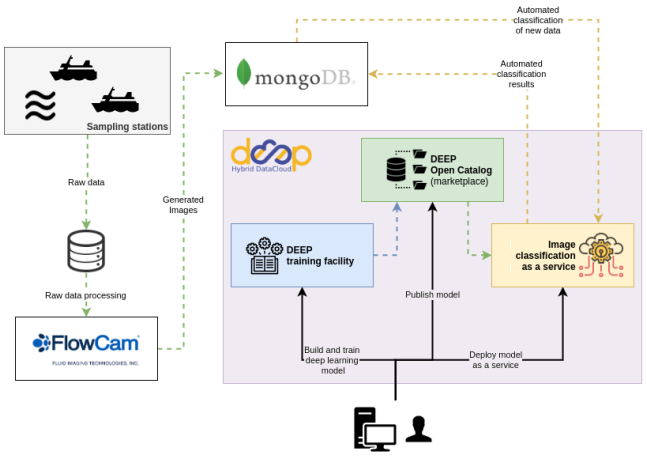

Figure 5. Architecture of the solution to provide a deep learning classification engine as a service applied to Phytoplankton classification using the DEEP-Hybrid-DataCloud stack.

Moreover, the sampling is being done in a semi-automated manner, leveraging specialized devices like FlowCAM [68]. The samples are collected and passed through the FlowCAM, resulting in multiple large collections of (10^5 - 10^6) plankton images. This increase in the number of digital image datasets make manual classification tasks even more time-consuming than before, therefore the development of an automated classification system seems like a natural task. Nowadays there is proliferation of automatic classification methods providing reproducible and standardized results. Deep learning techniques, and more specifically, Deep Neural Networks (DNN) are showing great achievements for detection and identification in such problems [69].

In this context we are providing VLIZ users with the architecture depicted in Figure 5. The basic idea behind this view is to provide scientists with tools to deploy a Phytoplankton automated classification engine as a service leveraging the whole DEEP stack, so that users do not need to deal with infrastructure allocation details.

In the aforementioned architecture, the raw collected data is stored in an mongoDB instance, an open source data base management system (DBMS) using a document-oriented database model supporting several types of data. Starting from a generic image classification model present in the marketplace, labelled data (from manual classifications previously done by taxonomists) was fetched form the database and fed directly to build and train the CNN model using the DEEP training facility. Once the model was trained and its accuracy was evaluated, it is deployed as an image classification service allowing to perform inference in a fast and horizontally scalable way.

Figure 6. Execution time for 100 consecutive requests for the default behaviour and the pre-warming of actions.

In order to evaluate this performance and scalability, we have performed an initial test of performing 100 consecutive requests to the same endpoint, evaluating the time required to provider a user with a response. As it is seen in Figure 6 the first request took approximately 45 seconds to complete, with the subsequent requests being completed in a fraction of this time, always below 3 seconds. This is due to the fact that, for the first invocation of the function, the serverless framework needs to download the Docker image into the local registry and spawn it before satisfying the request. All the subsequent requests are able to reuse the same Docker container, therefore the time to provide a user with a response is only the time of the processing, not of the aforementioned preparatory phase. This is the common behaviour in serverless frameworks, that normally maintain a warm pool of resources to satisfy the user requests in a reasonable amount of time. In order to alleviate event more this allocation time, the DEEPaaS supervisor implements a pre-warming of all the registered functions: whenever it registers a new function, it

performs an empty request so that subsequent requests do not need to wait for this warming time, as shown in Figure 6.

B. IMAGE CLASSIFICATION FOR BIODIVERSITY

We have developed a pipeline [70] that allows us to train an image classifier to solve a new problem with practically no code changes. Therefore we have trained several classifiers focused on solving different biodiversity problems, as will be detailed next.

All these applications showcase how easy it is to train new classifier without only very small variations, starting from an existing module from the marketplace. The most time consuming task when training the new classifiers was, by far, gathering and cleaning the new datasets. No time at all was spent on dealing and adapting the deep learning code.

Moreover, as all these classifier derive from the same codebase and expose the same API is trivial to build services and tools around them, like classification assistants or mobile applications for each new trained classifier.

these tools at the infrastructure and platform levels. Among the objectives of the project, there is the support to user communities to develop their machine learning applications and services in a way that encapsulates technical details the end user does not have to deal with. The areas covered by the research communities that are part of the DEEP consortium are: biological and medical science, computing security, physical sciences, citizen science and earth observation. DEEP is a community driven project, this means that they are the main stakeholders of DEEP and have been the key for the design of the whole computing framework.

The next steps foreseen for the platform are the inclusion of additional Machine Learning and Deep Learning modules, providing crowd-sourcing capabilities and working with scientific communities and industry to leverage the framework for their research. To this aim, the simplification and the automation of the inclusion process is needed, working together with user communities to streamline the platform adoption to improve the SQA process to strengthen the development and deployment of their machine learning modules. Moreover, pursuing the implementation of parallel and distributed training across several node instances is intended as a way to provide a better efficiency. Marketplace Improvements are also expected, allowing advanced search queries exploiting the full potential of the module metadata, as well as the promotion of the metadata schema. Lastly, future activities will be devoted on the integration of the DEEP framework into existing production e-Infrastructures, with special focus on supporting different distributed storage and data management systems.