ACTA TECHNICA JAURINENSIS
Vol. 15, No. 1, pp. 1-6, 2022
10.14513/actatechjaur.00581

Research Article

MLOps approach in the cloud-native data pipeline design

István Pölöskei[1,*]

[1]Adesso Hungary Kft.
Infopark sétány 1, 1117 Budapest, Hungary
*e-mail: istvan.poeloeskei@adesso.eu

Submitted: 25/12/2020 Accepted: 21/03/2021 Published online: 09/04/2021

Abstract: The data modeling process is challenging and involves hypotheses and trials. In the industry, a workflow has been constructed around data modeling. The offered modernized workflow expects to use of the cloud's full abilities as cloud-native services. For a flourishing big data project, the organization should have analytics and information-technological know-how. MLOps approach concentrates on the modeling, eliminating the personnel and technology gap in the deployment. In this article, the paradigm will be verified with a case-study in the context of composing a data pipeline in the cloud-native ecosystem. Based on the analysis, the considered strategy is the recommended way for data pipeline design.

Keywords: MLOps; cloud-native; data pipeline; machine learning

I. INTRODUCTION

Classical algorithms are apparent; they obey a clear logic in a human-readable manner. A machine learning model results from the training process by using unusual methods to reach the best result. The model evaluation might answer what the best result indicates. The way to the model is challenging, comprises assumptions and experiments.

The data and its structure has been evolving. In an enterprise, a workflow has been built around data modeling. This base helps mine the information from the data swiftly and efficiently, providing the organization's agility. The proposed modern workflow requires to use of the cloud's full capabilities. Planning a workflow according to the infrastructure, makes the operation more affordable and the implementation more powerful. Since the public cloud providers serve on-demand invoicing, the reserved resources should be connected to the running tasks [1].

II. CLOUD-NATIVE

1. Cloud benefits

The brand-new cloud technology innovations make computing more affordable and manageable than in the on-premised environment [2]. The training process of a deep learning model takes some time. It could occur that the local settings are not able to manipulate massive datasets. The training quality can often be efficiently increased by committing more resources, like attaching computation-intensive hyperparameter optimization measures [3].

2. Containers

Container technology is a state-of-the-art virtualization platform [4]. This approach presents an isolated, transportable bundle of the application with its dependencies. The currently applied (Docker) containers are originated from Linux primitives, implementing a process level separation [4].

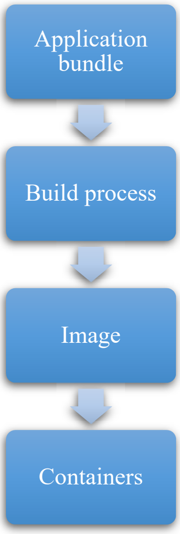

***Figure 1**. Container building process in Docker*

Docker containers are formed (**Fig. 1**) from layered immutable images based on their descriptor (Dockerfile) [5]. In the enterprise where containers are started and stopped dynamically by the runtime environment, it is advised to reduce the startup time; the container should be light-weight for better usability in the cloud-native circumstances [5]. In application development, the building process can be supported by assigned containers; the production container contains only the binaries with the runtime environment.

According to the 12 factors application principle [6], the conventional container-level application design must utilize the cloud-native's potential opportunities. The terms of the pattern, like stateless processes, are the primary entry points for cloud-native use-cases.

3. Cloud-native benefits

The Cloud Native Computing Foundation was established in 2015 to promote innovative container technologies [7]. According to their interpretation: the cloud-native is the set of technologies for providing scalable applications in the cloud, including ideas like micro services, DevOps, and agile [8]. Furthermore, cloud-native is more than architecture; it is a radical change in providing and developing a service. The cloud-native applications use the cloud's traits with its full potential.

Containers and micro services are the fundamental characteristics of cloud-native applications [8]. The functionalities are broken down into tiny autonomous bits, including an API-based interface. The pieces can be operating individually; each part has its deployment and version. The containers' concept proves each service process-wise sovereignty and lessens the entire system's complexity [8]. By wholly practicing cloud principles, it is feasible to serve compliant and scalable software.

Implementation's rate is vital. Fleet delivering features for the business is a strategic advantage. Introduction with a DevOps mindset and Agile, state-of-the-art deployment system can accomplish the business's demands more efficiently than before [8].

By the usage of the cloud-native best practices, the suggested architecture makes the application reliable and fault-tolerant. The framework replaces the broken element with a fresh instance, operating like a self-healing system. It preserves the functionalities by allocating more instances process-wise; the resources' dynamic usage is possible [9].

The state-of-the-art cloud-native solution is the Kubernetes, presented by Google [10]. In the Kubernetes, the containers are merged into POD, the atomic primitive in the Kubernetes ecosystem. Through the POD approach, the lifecycle of the processes can be executed through the orchestrator.

III. Data pipeline

1. Pipeline

The data pipeline is an adhesive code between the data sources and machine learning algorithms [11]. The code segment, as mentioned earlier, is continuously growing because the data composition strongly influences it. The data needs to be reconstructed and conveyed before the training commences. A workflow is a feasible solution, gathering and standardizing the required data, for providing high-level input for the machine learning algorithms. Since the entire process is divided into numerous levels, parallelized computing procedures are used [12].

The pipeline workflow design is not an uncomplicated task. Realizing the whole data processing requires some boilerplate codes. Workflow builders standardize the orchestration of data pipelines by using workflow engines and frameworks.

2. MLOps

Some prototypes are not deployed in the machine learning use-cases in the production situation because they may have concerns with the more prominent data-load or scaling, scheduling, or the integration to the data sources. For a successful big data project, the organization should have analytics and infrastructure expertise; they need to have a comprehensive plan about deploying the model and the production dataset's training cycles.

As a bridge linking the data scientist and software engineer, a unique responsible role is a viable solution to tackle machine learning models to production as soon as possible. This specialist needs to have a global picture of the model lifecycle, like organizing each stage's main interactions [13]. Infrastructural coordination is necessary for having efficient architectural usage and automatized, high-quality input data from diversified sources. It makes the deployment cycles of the models regular.

The role of MLOps as a Machine Learning branch of the DevOps is accepted in the machine learning domain [13]. It is a compound of operation, machine learning, and business understanding. All of them are distinct competencies. The DevOps methodology is for the speed and effectiveness of development and deployment, providing the project's adaptability in line with the agile [13]. MLOps is the same approach but focuses on the model development for eliminating the personnel and technology gap in the deployment. It concentrates on model formulation, evaluation, deployment, and monitoring. The new cloud-based solutions promote this approach vigorously.

3. Data platform

The data solution performs some steps in the data use-case, like the data-preparation, training, model verification, feature selection [12]. Achieving a data solution is challenging because it involves some perspectives like mathematics, informatics, and business [14]. It was stimulating to present a solution for data scientists in the last years, granting the possibility to deploy their model without code modifications in a productive environment. The complexity should be diminished through the modern data frameworks, giving a portable, scalable, and efficient infrastructure for the data experts. Managing the existing, same locally tested code, in the cloud environment, without an extensive perception of the cluster is an immense business value. The team can focus on the business, not on the cloud-native infrastructure or deployment. The effort to the productive data solution has been simplified, but the actual performance is to build an integrated fail-tolerant data infrastructure.

If the model has been constructed, the business should apply it. Conventional components also support the deployment section, like the TensorFlow Serving library [15]. The implementation unit or product support should permanently observe the usage of the model. Sometimes, the prototype should be aligned with the current data structure or obscure data patterns (**Fig. 2**). The model should be retrained for being up-to-date.

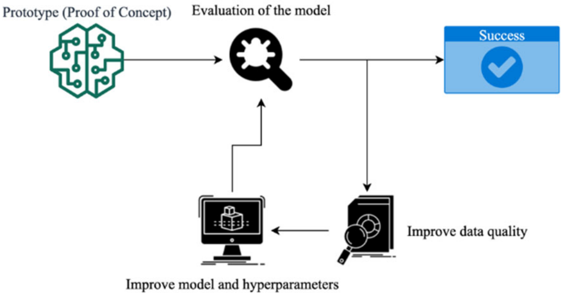
Prototype (Proof of Concept)
Evaluation of the model
Success
Improve data quality
Improve model and hyperparameters

Figure 2. *Prototyping process*

A CI/CD pipeline like a data platform is an added advantage because it accommodates the rapid prototyping process. Data science can explore their new idea directly in production. The new model can be deployed smoothly through some automatisms, producing business value as soon as possible in the continuously varying business environment. According to the MLOps principle, the Machine Learning tasks and artifacts can be combined with mainstream DevOps instruments [13].

4. Cloud-native pipelines

Kubernetes and TensorFlow are the prime open-source brands that were started by Google [16]. By them, some modern infrastructure has been established for giving improved solutions for novel problems. As their union, the KubeFlow is a state-of-the-art toolkit in a self-managed cloud-native ecosystem [17].

Kubeflow is an open-source cloud-native platform for developing data pipelines and workflows. It is a potent mixture of pipeline approach with cloud-native foundation [17]. It clarifies and normalizes the whole machine learning sequence in the cloud.

Machine learning and deep learning should be used with active data-processing in the cloud to determine contemporary data problems. According to the use-case discussed earlier, the KubeFlow may work as a fundamental part of its cloud-native strategy.

5. Workflow as graph

A pipeline can operate on the cluster. This procedure is cost-effective because the scaling-out is more affordable than the scaling-up [18]. Since the workflow is composed as a DAG (directed acyclic graph), each job can be executed as a DAG node [19].

Each task can be grouped by components (like python functions). The components, like regular atomic elements, can interact through their inputs and outputs. The low-level segments are not responsible for the cloud-naiveness. In the framework, the orchestrator is responsible for the supervision of the elements. The data specialists must not implement the infrastructure; they can only focus on their business processes [19].

Other pipelines can re-use each operator or component. By them, the entire pipeline can be controlled on the tasks level. The flow is visualized graphically, and each part has its log-stream. That facilitates the bug finding and fixing in the production.

For the coding, the KubeFlow accommodates jupyter notebooks as a standard coding context [17]. It is a popular conventional solution in the machine-learning society, strongly supports feature engineering and model fabricating. Through the built-in visualization, the data scientist can adequately decipher the information of the data. By a jupyter service provision, the most popular dependencies and libraries can be included by default; the resource quotas can also be configured. The configured runtimes can use GPU assistance as well (**Fig. 3**).

6. Infrastructure based on the pipeline

Establishing the infrastructure on the POD level of the nodes is complex. New frameworks, like Kubeflow, allow constructing pipelines at the code level without explaining them in any descriptor files. The script can control the whole infrastructure; the designed program can be scaled based on the load in a portable style [16]. This is propitious for machine learning use-cases when the training cycles are more resource-demanding than the prediction use-cases. The billing is based on consumption in the cloud; the corporation should not keep any additional resources when it is not necessary.

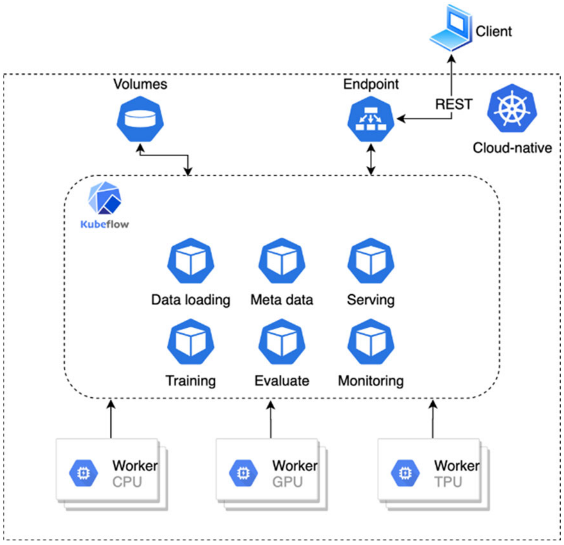

***Figure 3.** Kubeflow landscape*

The critical advantage of cloud-native infrastructure, the built-in features for the provision of the services. ReplicaSets, monitoring, heartbeat. All these factors and patterns are primary for consistent and fail-tolerant infrastructure [10]. The abstraction and standardization of the frameworks enhance the security and the overall maintainability.

7. GPU based pipelines

The data pipelines in the industry expect regular training cycles. This is resource consuming, but this resource should be granted only on demand. Cloud computing is a valid option for this use-case.

The workflows are compute-intensive but present a choice for distributed computing. Deep learning is a potent use-case of GPU programming [20]. By the native integration of GPU, the training can be accomplished more adequately without complicated GPU scripts.

8. TFX

Tensorflow is the state-of-the-art deep learning and computational framework [21]. Tensorflow Extended (TFX) library and specification help the Tensorflow model implementations in the production. A proper pipeline can be built around the deep learning model through its elements, like an end-to-end solution (**Fig. 4**). It is providing a high-performance application in the cloud-native environment.

Through Tensorflow Extended, a standardized TensorFlow pipeline can be created for particular orchestrators like Airflow and Kubeflow [21]. Having a multi-architectural approach has an advantage because it cannot be guaranteed that the project can obey the cloud-native principles. Integration and workflow management is a necessity, but the target system does not always use Kubernetes. In that case, traditional alternatives (like Airflow) must be used.

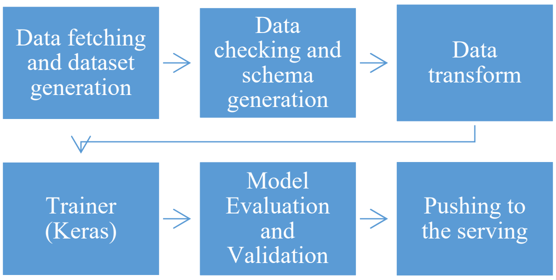
Data fetching and dataset generation
Data checking and schema generation
Data transform
Trainer (Keras)
Model Evaluation and Validation
Pushing to the serving

***Figure 4.** Tensorflow Extended pipeline*

IV. Case study: Data platform in manufacturing

1. Situation faced

Data science's fundamental duty is the observation and optimization of business processes. This journey is like a discovery of the production. The base-line situation is the progressed scrapping rate in manufacturing; the selected data scientist should investigate and analyze the root causes.

For the production-ready data-analytics result, the data team wants to constitute a data platform. The platform should utilize the public cloud approach; natively, it needs GPU support besides high availability. The cloud infrastructure should be sized according to the actual load; more resources should be involved in the exacting training steps. This architecture should support the data science department's standard data tools like Jupyter notebooks and python with deep learning support. The integration layer should be capable of loading the required data sources in the organization without any difficulty.

2. Actions taken

The MLOps approach adds the DevOps toolstack and cloud-native philosophy to the data workloads. An MLOps engineer represents the way how the model should be applied and deployed. Some orchestration difficulties have already been resolved through the introduction of KubeFlow in the cloud-native environment (**Fig. 5**).

The chosen strategy was shifting from the simple to more complex features. The base-line model can be improved for better accuracy through hyperparameter optimization and feature engineering. With a simplistic proof of concept, the data platform architecture can be validated for usability. Based on the evaluation outcome, the model's overall performance has been improved and verified by mainstream measurements (like accuracy, F1-score, etc.) [22]. If the model's correctness meets the requirements, the deployment begins.

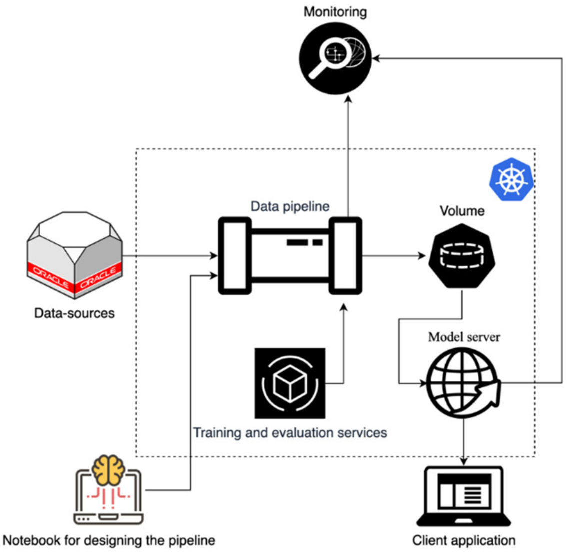

***Figure 1.** Realized pipeline. Cloud services have processed the input data. The created model has been published*

Through the prototype has been adapted several times, a version controlling is required. The model selection is also essential because the existing models could have more excellent performance than the new release. A basis artifact library can serve the previously built best model.

The workflow can be altered; it triggers container level cloud-native fundamental changes. The produced components can manage the complete training and evaluation workflow. Monitoring and logging operations are assigned to primary activities. The pipeline steps in details:

- Establishing the pipeline cloud-native architecture based on the repository
- Data pre-processing and data digestion
- Machine learning (classical or deep learning approach) in dedicated PODs
- Evaluation and model selection
- Deployment to the REST endpoint

The data scientists have built a promising offline model. The deployment and extension of the pipeline remain challenging. It takes time to reach the production level.

3. Results achieved

It is justified to set up a pilot project for a reduced scope for experiment purposes. The immediate feedback can solidify the concept shortly (**Fig. 6**). A production-level prototype verifies the theoretical approach and achieves the management's acceptance. The testing charge is lower in that case, and the design can be examined without higher production risks.

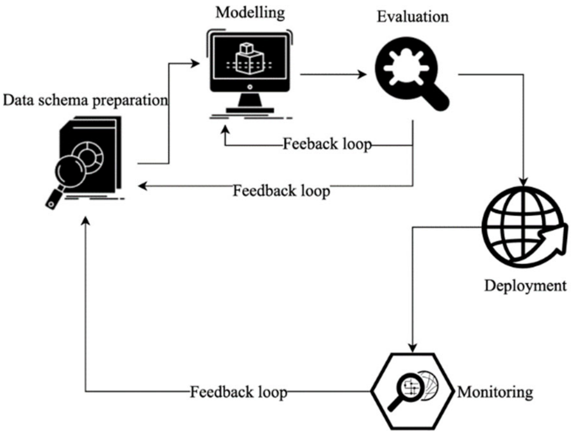

***Figure 2.** The quick feedback loop in production*

By acceptance of the MLOps mindset, the research loops become more expeditious. Through continuous integration, the pipeline and model can be arranged rapidly in production. Model optimization, like hyperparameters harmonizing, claims some trials and failures. The prompt reactions yield real business content in production.

The MLOps gives the dynamic construction of event-based workflows, with the native support of distributed data processing frameworks, implementing scale-out opportunities on the top of the cloud-native design.

4. Lessons learned

At the first iteration, the pipeline was composed of Jenkins jobs. It was complicated and not resembling the data science mindset. Between the prototyping and deployment, there was a considerable gap. KubeFlow with Tensorflow Extended resolved this concern. The framework accommodates a firm context for modeling without narrowing the data experts' opportunities. The utilized technology grants high-level automatization possibilities by the usage of elements that implement specific subsequent responsibilities. The Tensorflow Extended represents a current, comprehensive ML pipeline, from implementing the new business scenarios to managing the traditional machine learning activities.

V. Conclusion

Some designs never reach production in the machine learning area. A pipeline, like a structure, supports loading and digesting the data for machine learning procedures. The scalability is crucial because the training expenses are soaring in the latest neural network-based architectures; the complete workflow should be performed in an on-demand manner in the cloud. Cloud-native seems to be a convenient and future-proof solution. Based on the study, the examined strategy is the advised way for data pipeline configuration.

ACKNOWLEDGEMENT

The publishing of this paper was supported by XY.

AUTHOR CONTRIBUTIONS

I. Pölöskei: Conceptualization, Experiments, Writing

REFERENCES

[1] R. Gao, L. Wang, R. Teti, D. Dornfeld, S. Kumara, M. Mori, M. Helu, Cloud-enabled prognosis for manufacturing. CIRP Annals - Manufacturing Technology 64 (2) (2015) pp. 749-772. https://doi.org/10.1016/j.cirp.2015.05.011

[2] D. A. Tamburri, M. Miglierina, E. Di Nitto, Cloud applications monitoring: An industrial study. Information and Software Technology 127 (2020) 106376. https://doi.org/10.1016/j.infsof.2020.106376

[3] L. Franceschi, M. Donini, P. Frasconi, M. Pontil, On hyperparameter optimization in learning systems. 5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings.

[4] M. De Benedictis, A. Lioy, Integrity verification of Docker containers for a light-weight cloud environment. Future Generation Computer Systems 97 (2019) pp. 236-246. https://doi.org/10.1016/j.future.2019.02.026

[5] A. Martin, S. Raponi, T. Combe, R. Di Pietro, Docker ecosystem – Vulnerability Analysis. Computer Communications 122 (2018) pp. 30-43. https://doi.org/10.1016/j.comcom.2018.03.011

[6] M. Mohamed, R. Engel, A. Warke, S. Berman, H. Ludwig, Extensible persistence as a service for containers. Future Generation Computer Systems 97 (2019) pp. 10-20. https://doi.org/10.1016/j.future.2018.12.015

[7] D. Gannon, R. Barga, N. Sundaresan, Cloud-Native Applications. IEEE Cloud Computing, 4 (2017) pp. 16-21. https://doi.ieeecomputersociety.org/10.1109/MCC.2017.4250939

[8] S. Peltonen, L. Mezzalira, D. Taibi, Motivations, Benefits, and Issues for Adopting Micro-Frontends: A Multivocal Literature Review. https://arxiv.org/abs/2007.00293

[9] M. Malawski, A. Gajek, A. Zima, B. Balis, K. Figiela, Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions. Future Generation Computer Systems 110 (2020) pp. 502-514. https://doi.org/10.1016/j.future.2017.10.029

[10] S. Kho Lin, U. Altaf, G. Jayaputera, J. Li, D. Marques, D. Meggyesy, S. Sarwar, S. Sharma, W. Voorsluys, R. Sinnott, A. Novak, V. Nguyen, K. Pash, Auto-Scaling a Defence Application across the Cloud Using Docker and Kubernetes. Proceedings - 11th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion 2018. https://doi.org/10.1109/UCC-Companion.2018.00076

[11] D. Wu, L. Zhu, X. Xu, S. Sakr, D. Sun, Q. Lu, Building pipelines for heterogeneous execution environments for big data processing. IEEE Software, 33 (2) (2016) pp. 60-67. https://doi.org/10.1109/MS.2016.35

[12] Z. Peng, Stocks Analysis and Prediction Using Big Data Analytics. Proceedings - 2019 International Conference on Intelligent Transportation, Big Data and Smart City, ICITBS 2019. https://doi.org/10.1109/ICITBS.2019.00081

[13] I. Karamitsos, S. Albarhami, C. Apostolopoulos, Applying DevOps Practices of Continuous Automation for Machine Learning. Information 11 (7) (2020) 363. https://doi.org/10.3390/info11070363

[14] J. S. Saltz, S. Yilmazel, O. Yilmazel, Not all software engineers can become good data engineers. Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016. https://doi.org/10.1109/BigData.2016.7840939

[15] D. Baylor, K. Haas, K. Katsiapis, S. Leong, R. Liu, C. Menwald, M. Trott, H. Miao, M. Zinkevich, N. Polyzotis, Continuous training for production ML in the tensorflow extended (TFX) platform. Proceedings of the 2019 USENIX Conference on Operational Machine Learning, OpML 2019. https://www.usenix.org/conference/opml19/presentation/baylor

[16] Google, Kubeflow [cited 2020-12-25]. https://www.kubeflow.org/

[17] Z. Li, R. Chard, L. Ward, K. Chard, T. J. Skluzacek, Y. Babuji, A. Woodard, S. Tuecke, B. Blaiszik, M. J. Franklin, I. Foster, DLHub: Simplifying publication, discovery, and use of machine learning models in science. Journal of Parallel and Distributed Computing 147 (2021) pp. 64-76. https://doi.org/10.1016/j.jpdc.2020.08.006

[18] C. Avci Salma, B. Tekinerdogan, I. N. Athanasiadis, Domain-Driven Design of Big

DISCLOSURE STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Systems Based on a Reference Architecture. Software Architecture for Big Data and the Cloud (2017) pp. 49-68. https://doi.org/10.1016/b978-0-12-805467-3.00004-1

[19] R. Mitchell, L. Pottier, S. Jacobs, R. F. Da Silva, M. Rynge, K. Vahi, & E. Deelman, Exploration of Workflow Management Systems Emerging Features from Users Perspectives. Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019. https://doi.org/10.1109/BigData47090.2019.9005494

[20] F. Rouzbeh, P. Griffin, A. Grama, M. Adibuzzaman, Collaborative Cloud Computing Framework for Health Data with Open Source Technologies. https://doi.org/10.1145/3388440.3412460

[21] Google, Tensorflow [cited 2020-12-25]. https://www.tensorflow.org/

[22] M. Abdar, W. Książek, U. R. Acharya, R. S. Tan, V. Makarenkov, P. Pławiak, A new machine learning technique for an accurate diagnosis of coronary artery disease. Computer Methods and Programs in Biomedicine 179 (2019) 104992. https://doi.org/10.1016/j.cmpb.2019.104992