AI-TOOLKIT: A MICROSERVICES ARCHITECTURE FOR LOW-CODE DECENTRALIZED MACHINE INTELLIGENCE

Vincenzo Lomonaco[*], *Valerio De Caro*[*], *Claudio Gallicchio*[*], *Antonio Carta*[*], *Christos Sardianos*[†], *Iraklis Varlamis*[†], *Konstantinos Tserpes*[†], *Massimo Coppola*[‡], *Mina Marmpena*[⋆], *Sevasti Politi*[⋆], *Erwin Schoitsch*[∓], *Davide Bacciu*[*]

[*]Università di Pisa, [†]Harokopio University of Athens, [⋆]Information Technology for Market Leadership, [∓]Austrian Institute of Technology, [‡]Consiglio Nazionale delle Ricerche

ABSTRACT

Artificial Intelligence and Machine Learning toolkits such as Scikit-learn, PyTorch and Tensorflow provide today a solid starting point for the rapid prototyping of R&D solutions. However, they can be hardly ported to heterogeneous decentralised hardware and real-world production environments. A common practice involves outsourcing deployment solutions to scalable cloud infrastructures such as Amazon SageMaker or Microsoft Azure. In this paper, we proposed an open-source microservices-based architecture for decentralised machine intelligence which aims at bringing R&D and deployment functionalities closer following a low-code approach. Such an approach would guarantee flexible integration of cutting-edge functionalities while preserving complete control over the deployed solutions at negligible costs and maintenance efforts.

Index Terms— Artificial Intelligence, Microservices, Decentralized Learning and Inference, Pervasive Computing

1. INTRODUCTION

Artificial Intelligence and Machine Learning solutions are invaluable cutting-edge technologies that enable products and services even unthinkable a few decades ago, from autonomous vehicles to digital personal agents, running in our smartwatches. However, most of those solutions are based on three main assets: i) a large quantity of high-quality, labeled data; ii) a scalable integration and deployment infrastructure for efficient continuous training; iii) strong expertise in AI, data science, and engineering. Those needs often prevent large-scale adoption of these technologies from small and medium enterprises which can neither afford to outsource to large centralized cloud infrastructures their AI solutions nor face the frictions of in-house development.

Open-source R&D libraries such as Scikit-learn [1], PyTorch [2], and Tensorflow [3] have significantly lowered the barriers to the development of machine-learning based R&D projects. However, they still require specialized human resources to prototype and then deploy optimized solutions depending on the desired target use case. In this paper we propose a way to bridge this gap by providing the following key original contributions:

1. We propose a microservice-based architecture for decentralised machine intelligence (Section 2).
2. We discuss the development of off-the-shelf AI methodologies, functional building blocks and low-code approaches for maximizing easiness of deployment and system autonomy (Section 3).
3. We show initial results of the application of such tech-

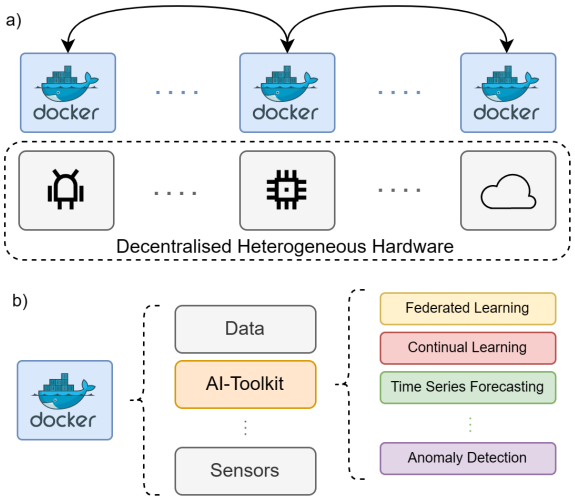

Fig. 1. General architecture of the TEACHING platform based on Dockerized micro-services (a). Images for different architectures and hardware are easily downloadable through Docker-Hub[2] with off-the-shelf AI functionalities made available by the AI-Toolkit library (b), as one of the key components of the platform.

This research was supported by the TEACHING project, funded by the EU Horizon 2020 research and innovation programme under GA n. 871385.

nology and methodologies in a real-world stress prediction from physiological signals task (Section 4.1).

4. Finally, we release the open-source implementation of the library (hereafter denoted as the "*AI-Tookit*") freely accessible on GitHub[3].

2. OVERALL ARCHITECTURE DESIGN

The AI-Toolkit has been developed within the European project TEACHING [4] to support the development of trustworthy autonomous cyber-physical applications through Human-Centred Intelligence. An initial blueprint and demonstration of the AI-Toolkit has been described in [5]. Such a toolkit leverages the "TEACHING platform" [4], a distributed computing infrastructure based on dockerized microservices as shown in Fig. 1. A message broker allows the dynamic definition of data workflows that may run on any platform that supports Docker containers (e.g. desktops, servers, embedded boards, etc.). Computational workflows can be deployed across multiple platforms. The key idea is that application developers should be able to use an abstract language to describe the data workflows that they want to deploy, and the system should be able to implement it transparently. In this context, a TEACHING *Application* is essentially modeled as a data workflow that is meant to close the loop between the machine and the human. Such a workflow may contain AI-based components which will be better discussed in Section 3. The TEACHING platform has been designed to support two types of end users for the TEACHING Platform: the core functionalities developers and the application developer/provider. The core developer manages the code through a GitHub[4] project and an automated Continuous Integration and Deployment (CI/CD) pipeline builds the Docker images to be deployed for the platform or each application. The Application developer then defines the application model as a data workflow and deploys it in one or more target systems. The Platform design focuses primarily on meeting the requirements of the application developer and encompasses four key objectives: monitoring, quantification, integration, and programmability. Monitoring and quantification are achieved by introducing data workflows that aggregate human and system monitoring data sources as well as ML models (quantification). Integration is possible through the inclusion of components at the ends of the workflow that can interact with the target AI-based system that is outside of TEACHING's control. Finally, programmability is achieved using architecture patterns that allow the creation of dynamic workflows over abstract declarations provided by the end users, with minimal intervention in their code. Towards that end, several features have been developed. In the next section, we focus on the AI-Toolkit, that is a specific library within the TEACHING platform supporting its AI functionalities.

[3]https://github.com/EU-TEACHING/teaching-ai-toolkit

[4]https://github.com

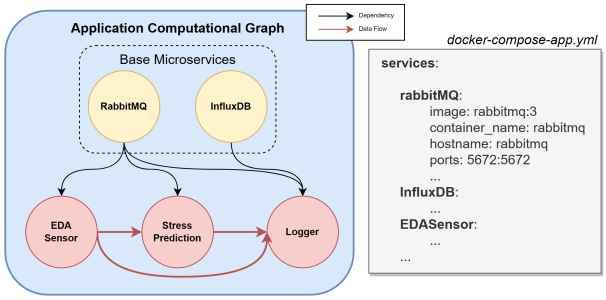

Fig. 2. Low-code definition of the *application computational graph*. AI pre-processing, training, and inference nodes can be easily instantiated with logical data flow and dependency connections as defined through a docker-composed *yaml* configuration file with its parameters.

3. AI-TOOLKIT AND OFF-THE-SHELF NODES

The *AI-Toolkit* collects and implements the AI microservices for a TEACHING application. As we mentioned above, a TEACHING application defines a *computational graph* as a docker-compose application (as shown in Fig. 2). The graph is composed of several nodes (microservices) that act as data *Producers*, *Consumers*, or both. The communication (exchange of JSON-based DataPackets) is handled by RabbitMQ [6] in a completely transparent way. The AI-Toolkit collects all the nodes that can be used for defining learning and inference modules implementing the overall AI features of any TEACHING app. The AI-Toolkit reflects the microservice-based nature of the TEACHING platform. As such, it does not force any constraints apart from the basic API for communication, training, inference, and data/configuration handling. This means that the AI-Toolkit, while providing significant support to the development of the AI functionalities within the app, can be tailored for the specific use cases and hardware platform. In particular, the AI-toolkit offers native *off-the-shelf support* for:

1. *Sequence learning*: Learning from structured sequential data is a key property of intelligence [7, 8]. It inherently relates to the ability to predict the future and know *"what is next"*. Within the AI-Toolkit we offer native support for recurrent neural networks for time-series forecasting and classification, which can work within weakly supervised or unsupervised regimes.
2. *Stress-Prediction*: built on top of our sequence learning algorithms we offer a state-of-the-art generally applicable predictor for stress recognition based on physiological data, such as Electrodermal Activity (EDA) or Photoplethysmographic (PPG) signals [9].
3. *Human-Centric Personalization via Reinforcement*

Learning: While many applications can be built based on labeled data and supervised learning, it is also important to consider tasks and environments where learning comes through intrinsic or extrinsic rewards. Within the AI-Toolkit we offer native support to predictors personalization based on reinforcement learning [10].

4. *Anomaly Detection and Cybersecurity*: the ability to identify irregularities is crucial for any autonomous system. By utilizing the Long Short-Term Memory AutoEncoder architecture [11, 12], this module can effectively facilitate semi-supervised learning from sequential data. The module can act as an intrusion detection system, thereby enabling AI-toolkit to offer support for cybersecurity measures.
5. *Dependable and Privacy Estimation*: Finally, we focused on the offering of dependable applications that can work within defined guarantees and are aware of the privacy leaked through learning, potentially offering a better tunable trade-off between privacy and performance [13, 14].

Without the pretense of completeness, such off-the-shelf functionalities constitute just a variegate example of what it is possible to build on top of the AI-Toolkit and the TEACHING platform microservices infrastructure.

3.1. Lower Level Functionalities

While off-the-shelf modules offer AI-Tookit *nodes* that can be instantiated right away through low-code solutions, the AI-Toolkit is hierarchically structured in basic utilities that can be used to define new nodes at ease:

- *Base Python Classes*: At the moment, the AI-Toolkit offers a set of base Python classes that can be inherited to simplify the development of new nodes. Figure 3 shows the basic API of every node. While we offer simplified Python interfaces, this does not limit the ability of the application developer to define heterogeneous nodes custom other dependencies.
- *Training and Inference*: It is important to note that each node in the application computational graph can be instantiated with different parameters which define the microservice behaviours. Each AI node can then be used for inference (prediction only), training, or both.
- *Reservoir computing*: Learning on constrained hardware and edge devices can be very demanding. Within the AI-Toolkit we offer basic utilities to train and use (partially) randomized neural networks exploiting their inner complex temporal dynamics. Such networks are generally more efficient to train, smaller and less energy-demanding. General Echo State Networks (ESN) for sequential classification and time-series forecasting are implemented in TensorFlow.

```python
class AIModule(object):

    def __init__(self):
        # method called once the process is created
        pass

    @TEACHINGNode(produce=True, consume=True)
    def __call__(self, input_fn):
        # method call evey time a new data package
        # it sent to the listened input channels
        pass

    def _build(self):
        # method run right the before the module
        # first time usage
        pass
```

Fig. 3. Main Application Program Interface (API) for every AI-toolkit microservice (node). Each node has to implement at least three main functions to build, initialise and call its main offered functionality as a new data package arrives.

- *Pre-processing*: Nodes in the computational graph are not necessarily related to predictive models, but may be also microservices offering data preprocessing and synchronization functionalities.
- *Federated Learning*: An application computational graph is generally intended as running on a single machine even though services (our nodes) can be easily distributed across several machines. Such applications may leverage "clones" of the computational graph, which are running in a distributed setting and perform federated learning. The AI-Toolkit offers transparent support for it, with the basic indication of a web server to contact to create clone groups.
- *Continual Learning*: Mixing learning and inference is key for any autonomous application facing real-world ever-changing conditions. However, learning from non-stationary data streams may pose some issues related to catastrophic interference. Within the AI-Toolkit we offer basic utilities to support training in a continual setting with experience replay [15, 16].

3.2. Defining Custom Nodes

The AI-Toolkit has been designed to be as modular and flexible as possible, allowing for easy customization. In addition to the off-the-shelf nodes, it also supports the definition and development of custom nodes. Implementing a new node can be fairly straightforward. Let us consider Figure 3 as a reference. Overall, we just need to define three key methods: `__init__`, `_build` and `__call__`. The combination of the `__init__` and the `_build` methods allows to per-

form the lazy initialization of the node: the former performs the parametrization of the execution environment, while the latter initializes the node during the first call of the internal logic. The `__call__` method implements the rationale of the node. To do so, it relies on the `TEACHINGNode` decorator, which determines whether the node is a producer, a consumer or both with respect to the other components in the system. In particular, if consume=True, then the `__call__` method's header must be equipped with an input generator input_fn which provides the DataPackets from nodes upstream. If produce=True then the `__call__` method needs to yield a DataPacket: this will be downstreamed automatically to the (eventual) consumer nodes.

4. APPLICATION DEFINITION AND DEPLOYMENT

A TEACHING application can be deployed on any operating system supporting Docker Compose (e.g. Windows, Mac OS, or Linux). Using Windows, it can be easily installed with Windows Subsystem for Linux (WSL2) and Docker Desktop[5]. Once the environment is properly configured, running the computational graph for an application is as simple as running docker-compose directly within your WSL image (e.g. Ubuntu based). Once you have defined your *my-application.yml* you can run it simply by running a few lines of code as shown in Figure 4.

```
git clone --recurse-submodules
https://github.com/EU-TEACHING/teaching-app

docker-compose -f teaching-app/my-
application.yml up
```

Fig. 4. Deploying a TEACHING application leveraging the AI-tookit is as simple as running a Docker-Compose application as defined within the *"my-application.yml"* definition file.

4.1. Stress Level Prediction using Echo State Networks

In Section 4 we described how we can easily set up and deploy an AI application leveraging the AI-Toolkit and the TEACHING platform. Such application is indeed describing the implementation of a basic use-case for human-state-monitoring, i.e. real-time stress-level predictions based on Electrodermal Activity (EDA; sometimes known as galvanic skin response, or GSR) [17]. EDA signals can be collected with non-invasive wearable sensors (e.g. smart gloves or smart watches) and real-time predictions can be computed directly on the edge for better privacy and personalization. For such reason, the off-the-shelf stress-prediction node leveraging Echo-State-Networks (ESNs) can be used. In Figure 5 a logging trace example is reported for a stress prediction node trained on real-world data of a subject placed in an autonomous driving simulator based on Carla [18]. In particular, the subject was wearing Shimmer Sensors[6] in a controlled setting and a simulated stress ground truth was generated based on the simulated vehicle lateral acceleration and velocity. If treated as a classification problem (as for classifying the level of stress in highly stressed, normal, and relaxed) at any point in time during the sequence, the model was able to reach about 94% in accuracy.

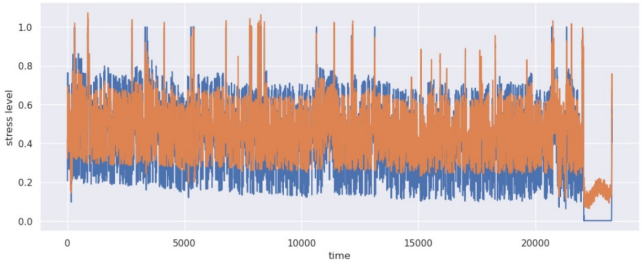

Fig. 5. Example trace of the stress prediction (orange) against the ground truth (blue) for a subject placed in an autonomous driving simulator based on Carla [18].

While this represents just a simple application of the AI-Toolkit functionalities, we can envision its usage in other trustworthy autonomous cyber-physical applications with humans in the loop such as avionics, another key use-case of the TEACHING project [4].

5. CONCLUSIONS & FUTURE WORKS

In this paper, we proposed a low-code microservices architecture for decentralised machine intelligence. Such a prototype, developed within the scope of the TEACHING project, provides a solid starting point to make our AI systems more pervasive and autonomous, reducing the friction in their prototyping and deployment in heterogeneous, distributed settings. In future works we plan to extend this toolkit and platform on three main aspects: i) provide a visual low-code solution to easily create AI applications without any knowledge about docker-compose; ii) expand the number of off-the-shelf nodes and docker images; iii) expand the support for low-level utilities for custom functionalities.

[5]More instructions and FAQ about the deploying procedure can be found in our documentation: https://github.com/EU-TEACHING/teaching-ai-toolkit

[6]`https://shimmersensing.com`

6. REFERENCES

[1] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[3] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "Tensorflow: a system for large-scale machine learning.," in *Proc. of 12th USENIX OSDI*. Savannah, GA, USA, 2016, pp. 265–283.

[4] Davide Bacciu, Siranush Akarmazyan, Eric Armengaud, Manlio Bacco, George Bravos, Calogero Calandra, Emanuele Carlini, Antonio Carta, Pietro Cassarà, Massimo Coppola, et al., "Teaching-trustworthy autonomous cyber-physical applications through human-centred intelligence," in *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*. IEEE, 2021, pp. 1–6.

[5] Valerio De Caro, Saira Bano, Achilles Machumilane, Alberto Gotta, Pietro Cassarà, Antonio Carta, Rudy Semola, Christos Sardianos, Christos Chronis, Iraklis Varlamis, et al., "Ai-as-a-service toolkit for human-centered intelligence in autonomous driving," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, 2022, pp. 91–93.

[6] David Dossot, *RabbitMQ essentials*, Packt Publishing Ltd, 2014.

[7] Benjamin A Clegg, Gregory J DiGirolamo, and Steven W Keele, "Sequence learning," *Trends in cognitive sciences*, vol. 2, no. 8, pp. 275–281, 1998.

[8] Davide Maltoni and Vincenzo Lomonaco, "Semi-supervised tuning from temporal coherence," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2509–2514.

[9] Jorge Rodríguez-Arce, Liliana Lara-Flores, Otniel Portillo-Rodríguez, and Rigoberto Martínez-Méndez, "Towards an anxiety and stress recognition system for academic environments based on physiological features," *Computer methods and programs in biomedicine*, vol. 190, pp. 105408, 2020.

[10] Christos Chronis, Christos Sardianos, Iraklis Varlamis, Dimitrios Michail, and Konstantinos Tserpes, "A driving profile recommender system for autonomous driving using sensor data and reinforcement learning," in *25th Pan-Hellenic Conference on Informatics*, 2021, pp. 33–38.

[11] Pankaj Malhotra, Lovekesh Vig, Gautam M. Shroff, and Puneet Agarwal, "Long short term memory networks for anomaly detection in time series," in *The European Symposium on Artificial Neural Networks*, 2015.

[12] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam M. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *ArXiv*, vol. abs/1607.00148, 2016.

[13] Georg Macher, Siranush Akarmazyan, Eric Armengaud, Davide Bacciu, Calogero Calandra, Herbert Danzinger, Patrizio Dazzi, Charalampos Davalas, Maria Carmela De Gennaro, Angela Dimitriou, et al., "Dependable integration concepts for human-centric ai-based systems," in *Computer Safety, Reliability, and Security. SAFECOMP 2021 Workshops: DECSoS, MAPSOD, DepDevOps, USDAI, and WAISE, York, UK, September 7, 2021, Proceedings 40*. Springer, 2021, pp. 11–23.

[14] Davide Bacciu, Antonio Carta, Daniele Di Sarli, Claudio Gallicchio, Vincenzo Lomonaco, and Salvatore Petroni, "Towards Functional Safety Compliance of Recurrent Neural Networks," in *Proceedings of the 1st International Conference on AI for People: Towards Sustainable AI, CAIP 2021, 20-24 November 2021, Bologna, Italy*, Dec. 2021.

[15] Vincenzo Lomonaco, "Continual learning with deep architectures," 2019.

[16] Andrea Cossu, Davide Bacciu, Antonio Carta, Claudio Gallicchio, and Vincenzo Lomonaco, "Continual learning with echo state networks," *European Symposium on Artificial Neural Networks (ESANN)*, 2021.

[17] Federico Matteoni, Andrea Cossu, Claudio Gallicchio, Vincenzo Lomonaco, and Davide Bacciu, "Continual learning for human state monitoring," *European Symposium on Artificial Neural Networks (ESANN)*, 2021.

[18] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.