Continual-Learning-as-a-Service (CLaaS): On-Demand Efficient Adaptation of Predictive Models

Rudy Semola[1,*], Vincenzo Lomonaco[1] and Davide Bacciu[1]

[1]*Department of Computer Science, University of Pisa*

Abstract

Predictive machine learning models nowadays are often updated in a stateless and expensive way. The two main future trends for companies that want to build machine learning based applications and systems are *real-time inference* and *continual updating*. Unfortunately, both trends require a mature infrastructure that is hard and costly to realize on-premise. This paper defines a novel software service and model delivery infrastructure termed Continual Learning-as-a-Service (CLaaS) to address these issues. Specifically, it embraces continual machine learning and continuous integration techniques. It provides support for model updating and validation tools for data scientists without an on-premise solution and in an efficient, stateful and easy-to-use manner. Finally, this CL model service is easy to encapsulate in any machine learning infrastructure or cloud system. This paper presents the design and implementation of a CLaaS instantiation, called Continual Brain, evaluated in two real-world scenarios. The former is a robotic object recognition setting using the CORe50 dataset while the latter is a named category and attribute prediction using the DeepFashion-C dataset in the fashion domain. Our preliminary results suggest the usability and efficiency of the Continual Learning model services and the effectiveness of the solution in addressing real-world use-cases regardless of where the computation happens in the continuum Edge-Cloud.

1. Introduction

Machine Learning (ML) has become a fast-growing, trending approach in solution development in practical scenario. Deep Learning (DL) which is a subset of ML, learns using deep neural networks to simulate the human brain. Nowadays, the latter is widely adopted in a variety of applications, especially in Computer Vision (CV) [1] [2] and Natural Language Processing (NLP) [3] [4]. Both academia and industrial research are investing significant efforts in developing ML-as-a-Service (MLaaS) tools to build and monitor a variety of smart applications [5] [6].

Current technical issues related to software development and delivery in organizations that work on ML projects have brought novel practicals and concepts. In particular, the integration of Machine Learning practices that support data engineering, with the Development and Operations (DevOps) practices based software development, has resulted in Machine Learning Model Operationalization Management (MLOps) [7]. MLOps principles have been proposed and used to deploy and maintain machine learning models in production reliably and efficiently. It incorporates ML models for solution development and maintenance with continuous integration (CI) to provide efficient and reliable service. Different roles such as data scientists, DevOps engineers, and IT professionals are involved in different MLOps process.

The accuracy of the predictions made by ML applications depends on many factors such as data type, training algorithm, hyperparameters, learning rate and optimizers. Different edge or cloud based applications need the latest real-time data and are retrained frequently to produce more accurate and precise predictions. Thus, the training models should be retrained without human intervention using reproducible and automating pipelines in a continuous manner. It is challenging to automate these decisions making processes using current MLOps. Moreover, these MLOps toolkits should be user-friendly, reliable, and efficient to use in industrial and practical scenarios.

Currently, we note also that ML serving systems are not able to handle the dynamic and non-stationary production environments adequately. The principal causes come from the concept drift [8] issue of real-life data. One of the main consequences is decay in the performance of the model. The second is monitoring data to understand when a drift distribution happens. To overcome these issues, we note that the companies innovative trend is toward real-time inferences and Continual Learning (CL) update models [9]. The first to generate more accurate predictions, and the latter to adapt models to changing non-stationary production environments [10].

To support the generation of CL solutions for research and practical applications, several systems have been developed, as summarized in Table 1. However, most of

Proceedings of the 1st International Workshop on Computational Intelligence for Process Mining (CI4PM) and the 1st International Workshop on Pervasive Artificial Intelligence (PAI), co-located with the IEEE World Congress on Computational Intelligence (WCCI), Padua, Italy, 18–23 July 2022

*Corresponding author.
r.semola@phd.unipi.it (R. Semola); vincenzo.lomonaco@unipi.it (V. Lomonaco); davide.bacciu@unipi.it (D. Bacciu)
https://github.com/rudysemola (R. Semola); https://www.vincenzolomonaco.com (V. Lomonaco); http://pages.di.unipi.it/bacciu (D. Bacciu)

Table 1
Comparison of several (continual) machine learning operations toolkits. HPO: hyper parameter optimization in training stage; MLOps: Model Operationalization Management; CM: Continuous Monitoring; CT: Continuous Training.

Tools	**Drift data detection**	**CL zoo algorithm**	**CL metrics**	**CM (MLOps)**	**CT (MLOps)**	**Portability (edge-cloud)**	**auto HPO**
Continual [11] AutoML (AWS)	⊠	-	-	⊠	⊠	-	⊠
AWS Sagemaker [12]	⊠	-	-	⊠	⊠	⊠	⊠
Avalanche [13]	-	⊠	⊠	-	-	-	-
ModelCI-e [14]	⊠	⊠	-	⊠	⊠	⊠	-
CLaaS	⊠	⊠	⊠	⊠	⊠	⊠	⊠
Continual Brain	-	⊠	⊠	-	⊠	⊠	-

these solutions do not support CL strategies for continuous training (CT). Several CL tools are widely used by research communities, but few of them meet the requirements of real industrial scenarios.

1.1. Proposal Solution and Scientific Value

In this paper, we propose a new model service paradigm named Continual-Learning-as-a-Service (CLaaS) with the following guidelines. First, CLaaS should be easily integrated with existing ML serving systems and MLOps toolkit either on-primes edge or cloud-based. Second, we are interested not only for research purposes but also in fast R&D prototype projects. Therefore, it allows for training and updating ML models in existing serving systems in an efficient, scalable and adaptable manner. Third, this model service can allow the building and maintaining of low-cost smart infrastructure without particular knowledge of Continual Learning.

We start to define CLaaS paradigm and compare it to existing related continual ML model services (Section 3). Then, the architecture and system implementation details of a possible CLaaS instance we called *Continual Brain* are described (Section 4). Some preliminary studies, comparing the Cumulative (when a new batch of data becomes available, prediction models are re-trained from scratch) [10] approach and well-known CL strategies, are also executed in two particular practical domains to demonstrate the potential advantages in terms of efficiency (Section 5). To sum up, future improvements and optimization directions are presented (Section 6).

2. Related Work and Background

This section first summarizes MLOps methodologies with a brief overview of the tools support. Then, it describes Continual Learning and its effectiveness in real-world contexts.

2.1. Machine Learning Operations (MLOps) and Tool support

MLOps is a set of scientific principles, tools, and techniques of Machine Learning and traditional Software Engineering to design and build complex computing systems. It encompasses all stages from data collection, to model building, to making the model in SW production system. MLOps emerges from the understanding that separating the ML model development from the process that delivers it, named ML operations, lowers the quality, transparency, and agility of the whole intelligent software [7]. For More detail, challenge and commercially available MLOps tool support in software development are well described in [15] and [16].

Typical workflow for machine learning-based software development includes three primary subjects. They are data, ML model, and code and this workflow consists of three main phases

- *Data Engineering*, concerned with data acquisition and data preparation
- *ML Model Engineering*, in which the process starts from model training, evaluation and serving
- *Code Engineering*, which it is integrated the ML model into the final product

There are three main problems that influence the value of ML models once they are in production. First, ML

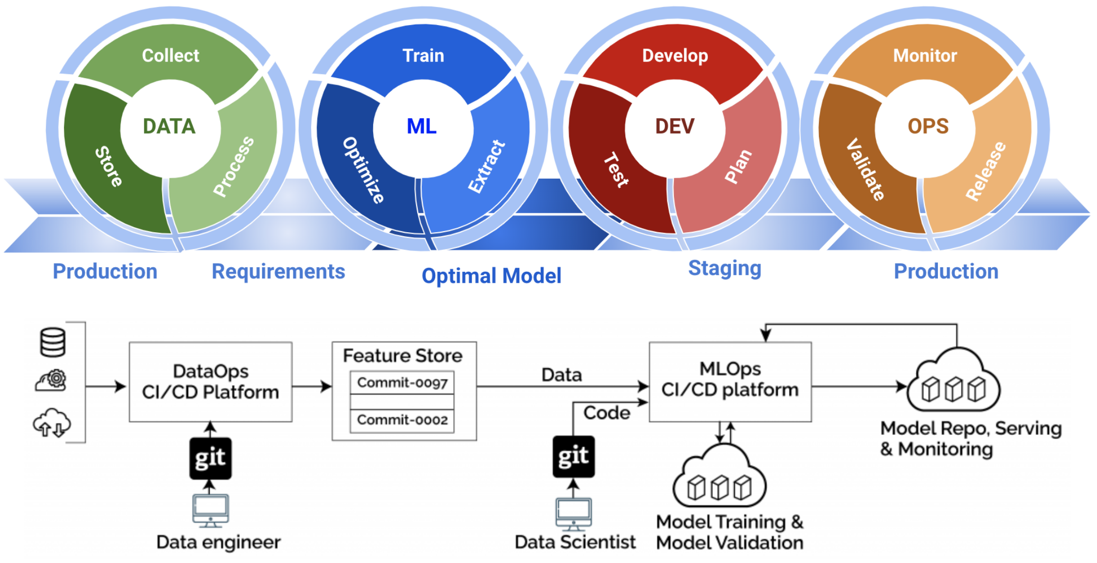


Figure 1: Graphical representation of MLOps, an end-to-end Machine Learning life-cycle management. It is an Iterative-Incremental Process mainly based on 3 stages: Data, Machine Learning (ML) and DevOps. The latter has two principles named Continuous Integration (CI) and Continuous Delivery (CD). MLOps adds other two practices named Continuous Monitoring (CM) and Continuous Training (CT). This work is focused on the Machine Learning development stage to train and validate the models with MLOps practices and Continual Learning methodologies.

models are sensitive to the semantics, amount and completeness of incoming data (data quality). Second is the performance degradation of ML models in production over time (model decay). In fact, real-life data are non-stationary and most of the time they have not been seen during the model training. Third, when transferring ML models to new business customers, these models, which have been pre-trained on different user demographics, might not work correctly according to quality metrics (locality).

MLOps try to establish effective practices and processes around designing, building, and deploying ML models into production. Currently, MLOps tools range from using a machine-learning platform to implementing an on-premise solution by composing open-source libraries.

MLOps platforms should support languages, frameworks and libraries in an easy-to-use unique environment. Although cloud service providers have similar platforms, they are costly and are not addressing the ML problem itself through a single dashboard. In addition to that, some of the platforms do not offer free licenses to use as embedded systems. The accuracy of the predictions made by DL models depends on many factors and some applications need the latest real-time data that are retrained frequently to produce more accurate and precise predictions. Thus, the training models should be retrained without human intervention using reproducible pipelines and in a continuous manner. It is challenging to automate these decisions making processes using the current MLOps tools. Continuous training and evaluation techniques and strategies have to take into account the non-stationarity nature of the data. Continual Learning methodologies could be beneficial for these MLOps tools.

2.2. Continual Learning and Real-World Applications

In a classical Continual learning problem, a single neural network model needs to sequentially learn a series of tasks. During training, only data from the current task is available and the tasks are assumed to be clearly separated. Continual learning methodologies deal with ML problems in a non-stationary data setting. They try to train models in a sequence of tasks to acquire new knowledge without forgetting what has been trained in the past. This problem has been actively studied in recent years and many methods for alleviating catastrophic forgetting have been proposed [17]. Unlike closed or simulation environments the data does not follow a stationary distribution in real applications. In a real-world application, we have a constant flow of information, where the dis-

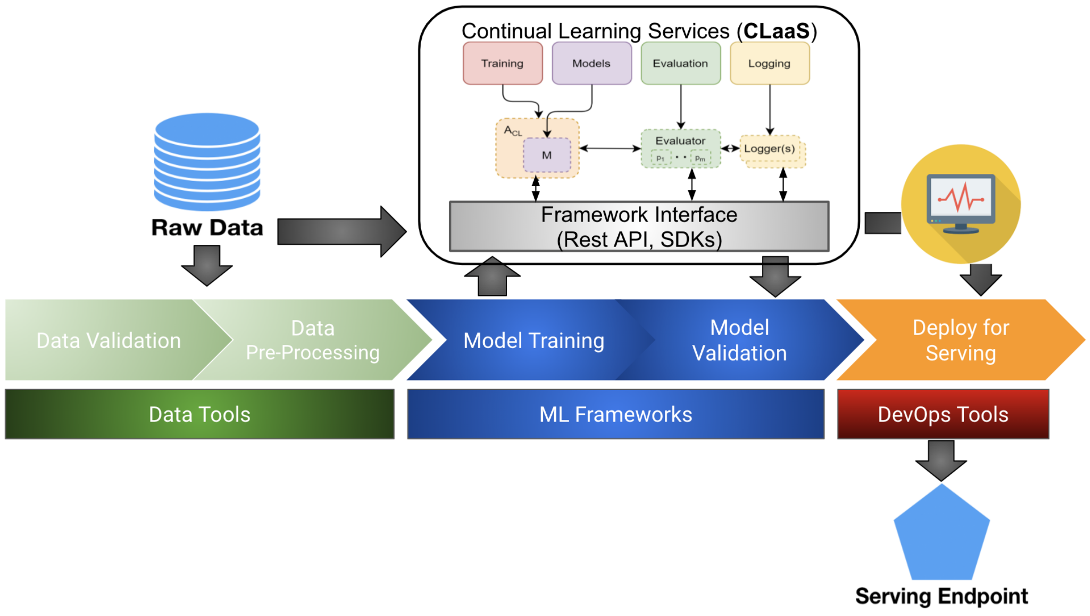


Figure 2: Graphical representation of a typical Machine Learning pipeline. This approach includes three procedures: (1) collection, selection and preparation of data to be used in model training, (2) finding and selecting the most efficient model after validation and (3) sending the selected model to the serving system. CLaaS is a service model approach that helps data scientists and engineers to monitor and update continually and efficiently the models over time.

tribution can change due to various external or internal factors. This problem creates the need to update the model continually efficient and adaptively manner.

Recent and relevant continual learning studies for different applications have been done. From surveillance videos, robotics and machine vision, clinical and medical sector, to the industry with edge and cloud computing [18] [19] [20]. These works demonstrate the usability and the effectiveness of CL in practical several domains.

At the same time, there are also many real-world and application-research directions unexplored with Continual Learning. Currently, companies interested to explore CL for their business purpose are dealing with costly and quite challenging approaches. It is required a research team and toolkit for expert of the CL domain. Furthermore, enabling CL in production is still a challenging problem. These problems motivated the design of a new service paradigm that embrace Continual learning Methodologies and tool in a cheaper and easy-to-use manner for real-world scenarios.

2.3. Avalanche, End-to-End Library for Continual Learning

In [13] it is proposed Avalanche, an open-source end-to-end library for continual learning research based on PyTorch. Avalanche provides a shared and collaborative codebase for fast prototyping, training, and reproducible evaluation of continual learning algorithms [13].

Avalanche is designed with five main principles important to CL research and real-world applications. These principles are *Comprehensiveness and Consistency*;*Ease-of-Use*; *Reproducibility and Portability*; *Modularity and Independence*; *Efficiency and Scalability*. At the current stage of development, the library is organized into five main modules: *Benchmarks*, *Training*, *Evaluation*, *Models* and *Logging*.

Avalanche implements a system of *Plugins* to facilitate the customization of strategies, metrics and logging. This is used by strategies, metrics, and loggers. It allows them to interact with the training loop and execute their code at the correct points using a simple interface. In particular, *SupervisedPlugin* is the base class for plugins for a supervised scenario, from which loggers inherit, and PluginMetric, a base class for metrics.

3. Continual-Learning-as-a-Service Paradigm

Continual-Learning-as-a-Service is a service model paradigm mainly based on Continual Learning methodologies to continuously monitor data distribution shifts and update the model in an efficient fashion. As presented in Figure 3, CLaaS are based on MLOps principles. In particular, Continuous Training (CT) and Continuous Monitoring (CM) are Continual Learning based. It is easy to use in an on-demand manner for both edge and cloud-based systems. This Section start from a comparative description of (continual) machine learning operations toolkits and follow with the motivations and benefits of the use of the CL features in this paradigm

3.1. Comparison of (continual) machine learning operations toolkits

The main features and differences between the other (continual) Machine Learning tools for MLOps are summarized in Table 1. Most of monitoring solutions are focused on analyzing statistics of a feature and alert when significant changes in these statistics happen. The level of automation of the ML pipeline defines the maturity of the ML process, which reflects the velocity of training new models given new data or training new models given new implementations. However, most of these tools do stateless retraining, where the model is trained from scratch each time without leveraging efficient and adaptable ways to continually evaluate models.

Instead, CLaaS infrastructures are set up to do *stateful* training. The latter is when you continue training your model on new data instead of retraining your model from scratch. Therefore, instead of updating your models based on a fixed schedule, continually update your model whenever data distributions shift and the performance of the model decay. The training frequency is triggered by the drift detector. Continual Learning strategies are applied to re-training the model in an efficient and adaptable manner. Furthermore, with the Continual Learning metrics, we can monitor model performance over time.

3.2. Continual Learning featured in CLaaS

The increasing demand for overcome classical machine learning process is leveraging the emergence of new solutions. For a company, CLaaS is a set of toolkit tools aimed to support the daily work of data scientists and data engineers in the machine learning development process. In particular, CLaaS provide features thought to manage the designing, building and managing of reproducible, testable ML-powered software. It is specialized to guarantee Continuous Monitoring and Training using Continual Learning solutions. In this subsection, it is described how CL methodologies are featured in the CLaaS paradigm with a brief review.

Detection of data drift distribution. In the classical ML, the data used in training follows a distribution supposed stationary. Based on this assumption, the generalization of model can be limited by the distribution of data on which it was trained. Nevertheless, In a real-world application, we have a constant flow of information, where the distribution can change due to various external or internal factors. These changes in the input data distribution cause issues in previously trained models, mainly because their weights are not prepared to face the drift concerning the training data. This problem creates the need to detect distribution shifts and update the model continually.

There are many types of concept drift that have been identified [21], each of which can affect model performance. Different approaches were developed for the different kinds of shifts. The CLaaS toolkit exploits both base and advanced approaches. In fact, the user can select among two main categories, supervised and unsupervised or build custom data drift approaches (for more detail of data drift solutions see Appendix in [11]).

Continual Learning zoo algorithms. There are many continual learning strategies in the literature developed for the neural network models. For a more in-depth overview, we refer the reader to the recent overviews in [10] and [22]. The latter additionally exposes the bio-inspired aspects of existing continual approaches. To roadmap in the CL strategies, it is useful to classify them into three main groups. First, the Memory-based Continual Learning Methods, in which gather all methods that save raw samples as the memory of past tasks in episodic memory. Second, Architecture-based Continual Learning Methods, in which they use the model to overcome catastrophic forgetting and learn over time. Typically, these classes of strategies exploit the dynamic neural network architecture changes that can be explicit or implicit. Finally, Regularization-based Continual Learning Methods consist in modifying the update of weights when learning in order to keep the memory of previous knowledge. The literature on CL strategies propose also hybrid solutions [10].

All these CL strategies allow the update of the model over time avoiding catastrophic forgetting and are in CLaaS. In fact, the user of the CLasS can select among these different continuous learning approaches for different practical scenarios. Most of real-world use cases, Memory-based Methods can be performant at the cost of the major memory usage. If data preservation is a constraint, the user could select CL strategies that are Memory-free like Architectural or Regularization strategies. Furthermore, in the edge computing context, the user can select efficient on-device CL strategies like [18] and [23].

Continual Learning metrics. A core feature not present in machine learning tools for the MLOps process is the use of the Continual Learning metric to detect model of the performance decay over time. It is crucial

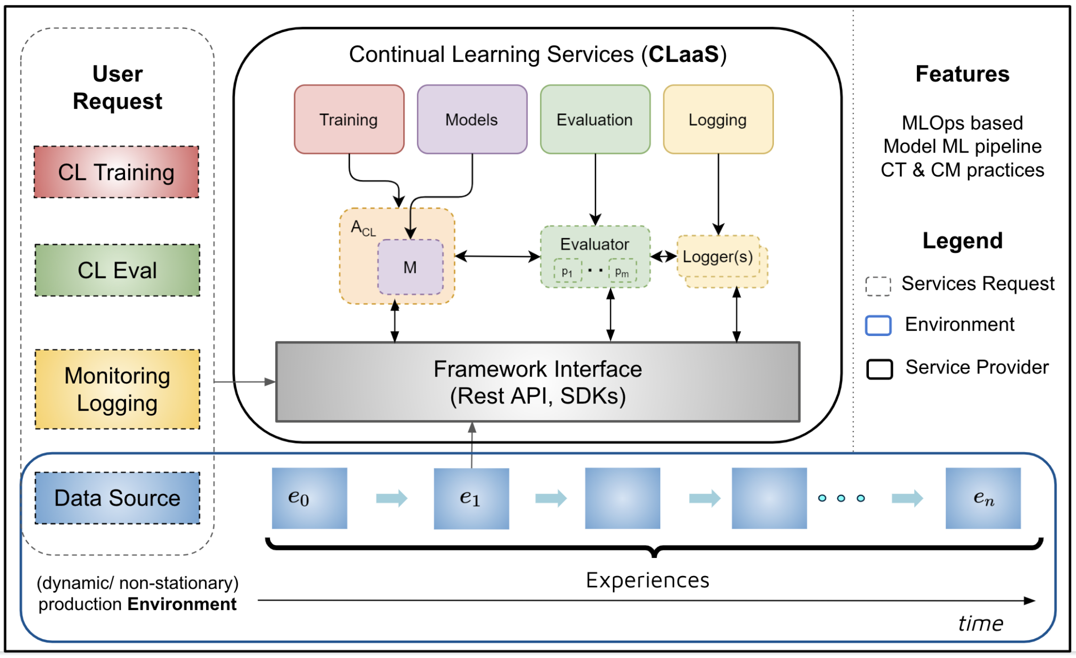


Figure 3: Graphical representation of the CLaaS paradigm. It is mainly based on MLOps practices and Continual Learning methodologies.

to have a set of good evaluation metrics to monitor the model performance, i.e. the model forgets the past or does not learn new skills. Detailed on CL evaluation protocols are well described in [10] [24] [25].

Using Continual Learning strategies in CLaaS, it is possible to train models in a sequence of tasks and acquire new knowledge without forgetting what has been trained in the past. The evaluation criteria have to cover the whole aspect of the full (continual) learning problem. It is not enough to observe good final accuracy on an algorithm to know if it is transferable to a serving system. We should also evaluate how fast it learns and forgets, if the algorithm is able to transfer knowledge from one task to another and if the algorithm is stable and efficient while learning and predict.

4. Continual Brain: System Architecture and Implementation

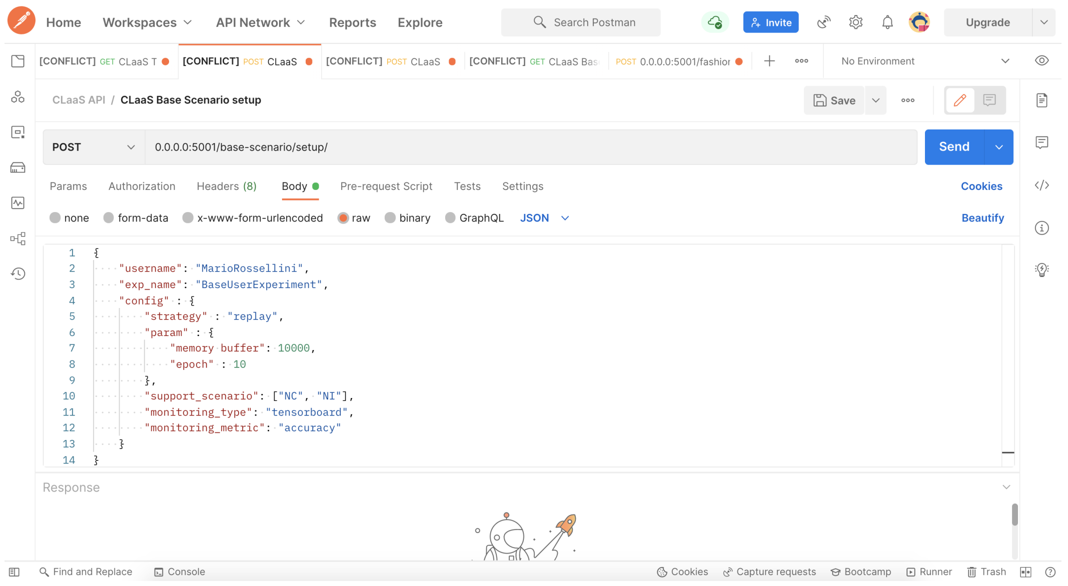
4.1. Design Principles

Continual Brain starts with a simple idea: extend the library Avalanche as a service model with a set of design guidelines: *modular and independent Building-Block view, backend flexible and expandable, portability, ease-of-use, efficiency and scalability*. We believe that these principles are important first steps for reliable Continual Learning tools in real-world applications.

Modular and independent Building-Block view The main design principle for Continual Brain follows from the concept of modular building blocks, the idea of providing a set of services independent of each other and a baseline for further improvements. This particular focus on module independence is maintained to guarantee the stand-alone individual module services. Moreover, the user can pick up a particular set of services and make use of a customize the others. For instance, the user can select CL strategy services and customize services of CL metrics or CL models.

Backend flexible and expandable Continual Brain is based on the principle of the API interface independent of the backend and easy to extend. Therefore, it is possible for the maintainers to expand the backend and add more services API exposed to the user. In this way, it allows also to employ of different environments, technologies and methods and guarantees flexibility of execution.

Portability A critical design objective of Continual Brain is to allow experimental results to be seamlessly portable in both edge and cloud resources. As the first step, we have decided to allow data scientist and engineer to simply integrate their own research and code into



```
POST 0.0.0.0:5001/base-scenario/setup/
```

```
{
    "username": "MarioRossellini",
    "exp_name": "BaseUserExperiment",
    "config" : {
        "strategy" : "replay",
        "param" : {
            "memory buffer": 10000,
            "epoch" : 10
        },
        "support_scenario": ["NC", "NI"],
        "monitoring_type": "tensorboard",
        "monitoring_metric": "accuracy"
    }
}
```

Figure 4: Example of user's request Restful API in JSON body (POSTMAN interface). Part of the configuration file to set up a typical experimental task in Continual Brain.

the platform to speed up the development of original continual learning solutions in practical scenarios.

Efficiency and scalability These two designing principles are fundamental in modern DL research experiments. Like current DL frameworks, we offer the end-user a seamless and transparent experience regardless of the use case or the hardware platform that the platform is run on.

Ease-of-use The last principle presented is the focus on simplicity. All the Continual Learning services in Avalanche are given in a simple set of calls. In particular, our efforts were focused on the design of an intuitive Application Programming Interface (API) and SW Development Kit (SDK) for data scientists and engineers.

4.2. System Workflow

The complete workflow of Continual Brain system can be split into two phases, the offline preparation phase and the online execution and monitoring phase.

In the offline phase, researchers first leverage the built-in APIs or SDK of the CL zoo strategies to use or customize a CL model. Meanwhile, engineers can prepare a configuration file to set up the trigger rule and the frequency of the retrain/fine-tuning according to our provided template.

In the online monitoring phase, the system follows the predefined rules in the configuration file to schedule model updating tasks. The main operation of monitoring model performance is based on CL metrics and it is executed in an automated manner with the user's configuration rules.

4.3. Architecture

The architecture of Continual Brain can be split into three main layers: *Interface*, *Middleware*, *CL-Backend*. Further internal components and frameworks model the overall structure, while the middle-layer endpoints are linked to the outermost components.

Interface, written in Flask python micro-framework, and helps the data scientist to interact with the CL-Backend. In particular, this layer manages communication with clients in RESTful APIs way or through an SDK (see Figure 4). This top layer interfaces with custom user code to run experiments or evaluate the performance with data tests.

Middleware, a Python package that manages the monitoring and update stages, and controls model versioning. Moreover, this middle layer is set up as a proxy to access external storage services or for lower-level storage. In this way, the logic of the lower-level storage is decoupled and flexible from the user storage preferences, making it possible also multiple storage services for the same user. These middle-level components allow also services external to the application. The latter can be either integrated into the application or completely external. For example, data storage services and the database can be provided in the CL-backend or can be out cloud services. Here, it is not used libraries but a custom codebase

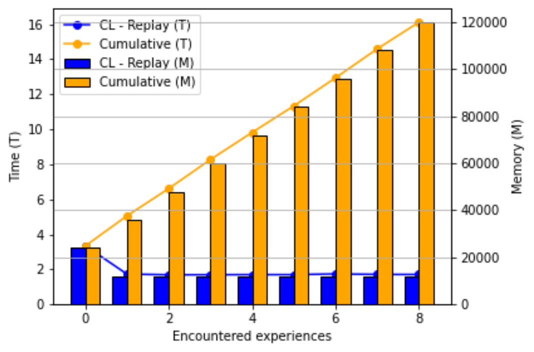


Figure 5: Cumulative and Replay CL-based strategy comparison (averaged on 3 runs) on CORe50 dataset: Computation in terms of time (min); memory in terms of the number of image patterns.

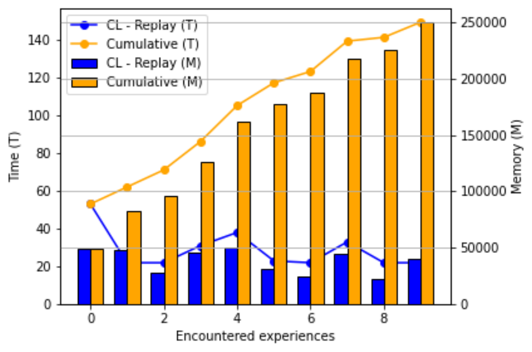


Figure 6: Cumulative and Replay CL-based strategy comparison (averaged on 3 runs) on DeepFashion-C dataset: Computation in terms of time (min); memory in terms of the number of image patterns.

implementation.

CL-Backend, leverages the Avalanche CL library [13] and PyTorch machine learning framework. The functionality of the Avalanche library in the Continual Brain service used are the CL zoo strategies, the CL models, the CL evaluation protocols and the easy set-up of CL scenarios (Table 1).

Finally, Continual Brain was designed as a microservice architecture and implemented using the Docker Compose tool. The motivation is into the advantage of the architecture and tool for managing the application as a set of containers, giving the possibility to create multiple instances of the same service. The latter is important to guarantee the Continuous Training property in an MLOps manner.

5. Empirical Evaluation

Computer Vision (CV) and Deep Learning in robotics vision and fashion domain have become hot topics and received a great deal of attention in both academia and the industrial landscape [10] [26]. This section illustrates the advantage of CL as a model service in automating continuous model updating. In particular, we explore both object recognition in robotics and fashion real-world applications [27], by discussing system efficiency when using a Continual Brain implementation.

5.1. Experiment Setup

All experiments are conducted with the following settings. We use common CL benchmark named CORe50 [28] for Computer Vision object recognition application. We also employ an instance of fashion analysis named categories prediction to evaluate both performance and efficiency. The latter is an enabling task for several fashion applications like visual search or visual recommendation [27]. We use the DeepFashion-C dataset [29] to build a new CL-benchmark in the fashion domain. After splitting the dataset in train and test set, to simulate the CL process, we build a New Classes (e.g. NC, also known as *Class-Incremental Learning*) scenario with 10 experiences. Therefore, the first experiences contain 10 classes while the rest contain 4 classes. The metric that is used to evaluate the clothing recognition models is the top-k accuracy. The state-of-the-art methods to solve category and attribute prediction task in DeepFashion-C are summarized in [27].

To demonstrate the efficiency of the CL strategies in these practical domains, we compare a Cumulative approach with the Replay CL-based strategies [22]. Cumulative is a stateless and time-consuming update process where all the data encountered through time are used to train the model in an offline manner. We ran the experiments on a Multi-GPU NVIDIA-SMI server with 80-core Intel Xeon CPU E5-2698 v4 and 4 Tesla V100 GPUs 11.2 CUDA Version.

5.2. Results

As Figures 7 and 8 show, the strategies based on Continual Learning are very efficient at the cost of a small decay in predictive performance. As predicted, the Cumulative strategies achieve similar results of the upper bound but the cost grows with the number of encountered experiences. In Figure 5 we note that the time and memory saved by the CL approaches are up to a ×6 factor for a stream of 9 experiences. In Figure 6 this nears an ×8 factor in 10 experiences. The performance-efficiency trade-off is evident for the Replay CL strategy. Finally, the Replay Continual Learning strategy has a stable training time in the growth of experiences. For this CL strategy the value 15000 is used for the *memory_size* hyper-parameter in DeepFashion-C dataset. For CORe50 dataset the value of the *memory_size* are setting to 5000.

These results demonstrate that the automation of the ML pipeline can be achieved efficiently and in an adaptable manner with this CL-tool. Therefore, in non-stationary production environments CL strategies might prove essential to attain efficient stateful training.

6. Conclusion and Discussion

In this paper, we have described CLaaS, a novel software as a service and licensing model mainly based on a continual learning approach. A first version, named Continual Brain, has been implemented by extending the functionalities of Avalanche as a set of micro-services. We have demonstrated the usability and efficiency of the system with representative case studies in machine vision and fashion domains.

Nowadays,the industrial and company applications could help to direct new CL research directions toward more practical scenarios. There are various practical fields that researchers and companies have investigated with Continual Learning. The literature has proposed different CL real-world applications. For instance, in MLOps, surveillance videos, robotics and machine vision, clinical and medical sector, industry and edge computing. Anyway, there are also many real-world and application-research directions unexplored with Continual Learning.

Prevalent scenarios of Continual Learning are Class Incremental Learning (CIL). They assume disjoint sets of classes as tasks but are less realistic in a real-world application. Recent works [30] and [31] focus the attention to address a more realistic CL setup which occurs frequently in real world AI deployment scenarios. The request to practical CL setting seems also start from the companies that use CL for core business.

Finally, deploying CL to real-world applications is still quite challenging. The absence of CL tools in ML system workflow and the poor investigation of real-world applications are the main reasons. We believe the CLaaS paradigm can narrow the gap between research and its industrial application, and speed up companies innovative trend towards Continual Learning.

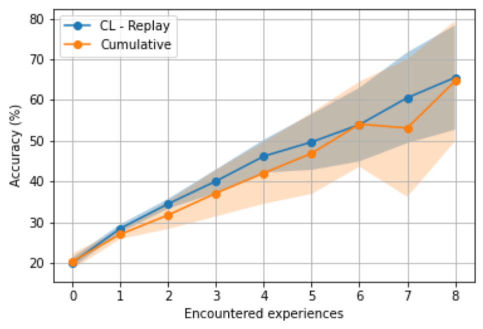


Figure 7: Cumulative and Replay CL-based strategy comparison (averaged on 3 runs) on CORe50 dataset: predictive performance in term of Accuracy using the entire test data for each experience.

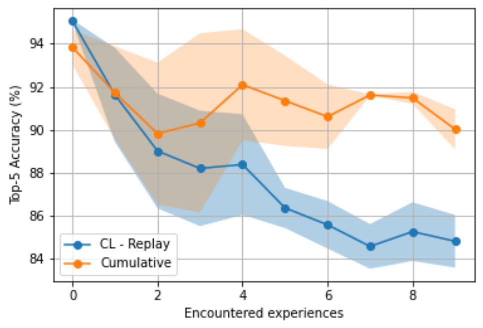


Figure 8: Cumulative and Replay CL-based strategy comparison (averaged on 3 runs) on DeepFashion-C dataset: predictive performance in term of Top-k accuracy with $k = 5$ accumulating the test stream during the experiences.

References

[1] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, cnn architectures, challenges, applications, future directions, Journal of Big Data 8 (2021).

[2] Z. Zou, Z. Shi, Y. Guo, J. Ye, Object detection in 20 years: A survey, ArXiv abs/1905.05055 (2019).

[3] D. W. Otter, J. R. Medina, J. K. Kalita, A survey of the usages of deep learning for natural language processing, IEEE Transactions on Neural Networks and Learning Systems 32 (2021) 604–624. doi:10.1109/TNNLS.2020.2979670.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, CoRR abs/1706.03762 (2017). URL: http://arxiv.org/abs/1706.03762. arXiv:1706.03762.

[5] L. E. Li, E. Chen, J. Hermann, P. Zhang, L. Wang, Scaling machine learning as a service, in: C. Hardgrove, L. Dorard, K. Thompson, F. Douetteau (Eds.), Proceedings of The 3rd International Conference on Predictive Applications and APIs,

volume 67 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 14–29. URL: https://proceedings.mlr.press/v67/li17a.html.

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

[7] Machine learning operations, 2022. URL: https://ml-ops.org/.

[8] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proceedings of the national academy of sciences 114 (2017) 3521–3526.

[9] C. Huyen, Real-time machine learning: challenges and solutions, https://huyenchip.com/2022/01/02/real-time-machine-learning-challenges-and-solutions.html, ????

[10] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, N. Díaz-Rodríguez, Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges, Information fusion 58 (2020) 52–68.

[11] T. Diethe, T. Borchert, E. Thereska, B. Balle, N. Lawrence, Continual learning in practice, arXiv preprint arXiv:1903.05202 (2019).

[12] Amazon sagemaker, ???? URL: https://aws.amazon.com/it/pm/sagemaker.

[13] V. Lomonaco, L. Pellegrini, A. Cossu, A. Carta, G. Graffieti, T. L. Hayes, M. De Lange, M. Masana, J. Pomponi, G. M. Van de Ven, et al., Avalanche: an end-to-end library for continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3600–3610.

[14] Y. Huang, H. Zhang, Y. Wen, P. Sun, N. B. D. TA, Modelci-e: Enabling continual learning in deep learning serving systems, arXiv preprint arXiv:2106.03122 (2021).

[15] N. Hewage, D. Meedeniya, Machine learning operations: A survey on mlops tool support, arXiv preprint arXiv:2202.10169 (2022).

[16] D. Kreuzberger, N. Kühl, S. Hirschl, Machine learning operations (mlops): Overview, definition, and architecture, arXiv preprint arXiv:2205.02302 (2022).

[17] G. M. van de Ven, A. S. Tolias, Three scenarios for continual learning, CoRR abs/1904.07734 (2019). URL: http://arxiv.org/abs/1904.07734. `arXiv:1904.07734`.

[18] T. L. Hayes, C. Kanan, Online continual learning for embedded devices, arXiv preprint arXiv:2203.10681 (2022).

[19] K. Doshi, Y. Yilmaz, Continual learning for anomaly detection in surveillance videos, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2020.

[20] D. Kiyasseh, T. Zhu, D. A. Clifton, Clops: Continual learning of physiological signals, arXiv preprint arXiv:2004.09578 (2020).

[21] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, N. D. Lawrence, Dataset shift in machine learning, Mit Press, 2008.

[22] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Networks 113 (2019) 54–71.

[23] L. Pellegrini, G. Graffieti, V. Lomonaco, D. Maltoni, Latent replay for real-time continual learning, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 10203–10209.

[24] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, J. van de Weijer, Class-incremental learning: survey and performance evaluation on image classification, arXiv preprint arXiv:2010.15277 (2020).

[25] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A continual learning survey: Defying forgetting in classification tasks, IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).

[26] W.-H. Cheng, S. Song, C.-Y. Chen, S. C. Hidayati, J. Liu, Fashion meets computer vision: A survey, ACM Computing Surveys (CSUR) 54 (2021) 1–41.

[27] X. Gu, F. Gao, M. Tan, P. Peng, Fashion analysis and understanding with artificial intelligence, Information Processing & Management 57 (2020) 102276.

[28] V. Lomonaco, D. Maltoni, Core50: a new dataset and benchmark for continuous object recognition, in: Conference on Robot Learning, PMLR, 2017, pp. 17–26.

[29] Z. Liu, P. Luo, S. Qiu, X. Wang, X. Tang, Deepfashion: Powering robust clothes recognition and retrieval with rich annotations, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 1096–1104.

[30] J. Bang, H. Kim, Y. Yoo, J.-W. Ha, J. Choi, Rainbow memory: Continual learning with a memory of diverse samples, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8218–8227.

[31] H. Koh, D. Kim, J.-W. Ha, J. Choi, Online continual learning on class incremental blurry task configuration with anytime inference, arXiv preprint arXiv:2110.10031 (2021).