Avalanche: A PyTorch Library for Deep Continual Learning

Antonio Carta  
*University of Pisa*  
antonio.carta@unipi.it

Lorenzo Pellegrini  
*University of Bologna*  
L.Pellegrini@unibo.it

Andrea Cossu  
*Scuola Normale Superiore*  
ANDREA.COSSU@SNS.IT

Hamed Hemati  
*University of St. Gallen*  
HAMED.HEMATI@UNISG.CH

Vincenzo Lomonaco  
*University of Pisa*  
VINCENZO.LOMONACO@UNIPI.IT

Editor: Joaquin Vanschoren

Abstract

Continual learning is the problem of learning from a nonstationary stream of data, a fundamental issue for sustainable and efficient training of deep neural networks over time. Unfortunately, deep learning libraries only provide primitives for offline training, assuming that model’s architecture and data are fixed. Avalanche is an open source library maintained by the ContinualAI non-profit organization that extends PyTorch by providing first-class support for dynamic architectures, streams of datasets, and incremental training and evaluation methods. Avalanche provides a large set of predefined benchmarks and training algorithms and it is easy to extend and modular while supporting a wide range of continual learning scenarios. Documentation is available at [https://avalanche.continualai.org](https://avalanche.continualai.org).

**Keywords:** Continual Learning, lifelong learning, PyTorch, reproducibility, machine learning software

1. Introduction

Learning continually from non-stationary data streams is a long-sought goal in Artificial Intelligence. While most deep learning methods are trained offline, there is a growing interest in Deep Continual Learning (CL) (Lesort et al., 2020) to improve learning efficiency, robustness and adaptability of deep networks. Deep learning libraries such as PyTorch and Tensorflow are designed to support offline training, making it difficult to implement continual learning methods. Avalanche, initially proposed in Lomonaco et al. (2021), provides a comprehensive library to support the development of research-oriented continual learning methods. The library is maintained by the ContinualAI non-profit organization. Compared to existing continual learning libraries (Douillard and Lesort, 2021; Wolczyk et al., 2021; Normandin et al., 2022; Mirzadeh and Ghasemzadeh, 2021; Masana et al., 2022), Avalanche offers a comprehensive set of tools and primitives for continual learning.

1. Official Avalanche website: [https://avalanche.continualai.org](https://avalanche.continualai.org)

©2023 Antonio Carta, Lorenzo Pellegrini, Andrea Cossu, Hamed Hemati, Vincenzo Lomonaco.

License: CC-BY 4.0, see [https://creativecommons.org/licenses/by/4.0/](https://creativecommons.org/licenses/by/4.0/) Attribution requirements are provided at [http://jmlr.org/papers/v24/23-0130.html](http://jmlr.org/papers/v24/23-0130.html)
2022), Avalanche supports a larger number of methods and scenarios. Avalanche is a library with a growing user base, an increasing set of features, and a strong focus on reproducibility.

2. What can you do with Avalanche?

Avalanche is a library built on top of PyTorch (Paszke et al., 2019), designed to provide simple and stable components with everything that you need to execute continual learning experiments. The library is split into 5 modules: benchmarks, training, models, evaluation, and loggers. Figure 1 shows a high-level overview of the library and its major components.

**Benchmarks** provides standard benchmark definitions (doc), high-level benchmark generators (doc), and low-level utilities to define new benchmarks by manipulating stream of experiences and datasets (doc).

**Training** provides standard training algorithms. Training strategies can be easily extended and combined with each other. Many continual learning techniques are available to be used out-of-the-box (doc). For replay methods, custom storage policies can be implemented, and many options such as balancing methods and reservoir sampling are available (doc). Most training methods can be combined together to create hybrid strategies.

**Models** provides CL architectures and first-class support for dynamic architectures, multi-task models, and optimizer update (doc). Dynamic Modules implement growing architectures such as multi-head classifiers and progressive networks (Rusu et al., 2016).

**Evaluation** provides CL metrics (doc). Metrics are defined with a declarative API and computed automatically. Avalanche provides an extensive set of metrics to measure the model’s performance and to keep track of system’s metrics such as memory occupation and CPU usage. All the metrics can be computed at different granularities (e.g. minibatch,

---

2. https://github.com/ContinualAI/avalanche
experience, stream). Custom metrics can be defined or they can be easily computed from the results of the existing metrics.

Logging  metrics are collected and serialized automatically by the logging system. The EvaluationPlugin (doc) connects training strategies, metrics, and loggers, by collecting all the metrics and dispatching them to all the registered loggers. TensorBoard, Weights and Biases, CSV files, text files, and standard output loggers are available (doc), but the logging interface can be easily extended with new loggers for custom needs.

Core Utilities  Avalanche offers a checkpointing functionality, allowing to pause and resume experiments. All Avalanche components are serializable.

3. API and Design
At the high level, Avalanche provides ready-to-use strategies, which can be instantiated and trained with a minimal amount of code (example). Internally, the training and eval loops implement a callback system that supports external plugins. Plugins allow to build on top of existing strategies with minimal changes and to combine different strategies together, a critical feature that other CL libraries do not support. Thanks to the plugin system, many strategies can be easily reused in different scenarios or combined together without any change to their code.

Benchmarks, Streams, Experiences  Benchmarks in Avalanche provide the data needed to train and evaluate CL models. Benchmarks are a collection of streams (e.g., a train and test stream for SplitMNIST (Lomonaco et al., 2021)). Streams are sequences of Experiences, where an experience stores all the information available at a certain point in time. Experiences provide all the necessary information for training, evaluation and logging. For example, in supervised CL experiences provide a dataset, while in reinforcement learning they provide an environment. Streams and experiences have private unique identifiers for logging purposes.

Dataset and Stream Manipulation  AvalancheDataset extends PyTorch datasets with the ability to add attributes values (such as task labels and other metadata) at the example granularity, manage different groups of transformations, and define custom collate functions. AvalancheDatasets can be subsampled and concatenated, providing a simple API to define and manipulate streams and replay buffers. In addition, Avalanche offers flexible data loaders that control balancing and joint sampling from multiple datasets.

Dynamic and Standalone Components  Avalanche extends many PyTorch static components into dynamical objects. For example, CL strategies may require changing the model’s architecture, optimizer, losses, and datasets during training. In PyTorch, these are static objects that are not easy to update during training (e.g., the architecture of nn.Modules is fixed). Avalanche DynamicModules provide a simple API to update the model’s architecture, ExemplarsBuffer manage replay buffers, regularization plugins update the loss function after each experience, and optimizer are also updated before each experience. Every component is automatically managed by Avalanche strategies, but they can also be used standalone in a custom training loop (example).
Inside an Avalanche Strategy  Figure 2 shows a high-level overview of an Avalanche strategy. *Templates* define the structure of the training and evaluation loops, providing a set of callbacks that can be used to execute code at any point of the loop. Strategies combine templates to define a complex training loop (e.g., base, SGD, online, supervised, meta-learning). *Plugins* use callbacks to access the strategy state and execute code at specific moments of the loop. Finally, a *Strategy* is an implementation of a set of templates with a list of plugins (e.g., Naive finetuning with ReplayPlugin, EWCPlugin, EvaluationPlugin). Most features in Avalanche can be used as a plugin: training utilities, CL methods, the evaluation and logging system. The advantage of this approach is that any plugin can be used with any template that supports its required callbacks and attributes, making it easy to write general components that can be reused across many experimental settings.

Testing  Avalanche is thoroughly tested with a battery of unit tests. Each pull request is tested on a subset of the unit tests by the continuous integration pipeline on Github. A subset of continual-learning-baselines (link) is executed with a regular cadence to ensure that Avalanche baselines are in line with expected results from the literature.

4. Conclusion

Currently, Avalanche v0.3.1 constitutes the largest software library for deep continual learning. Its main focus on fast prototyping, re-producibility and portability makes it the perfect candidate for research-oriented projects. The library is a result of more than two years of development effort involving more than fourteen different research organizations across the world. The MIT licensed software and the support of ContinualAI ensure continuity and alignment with the continual learning research community at large. In the future, we plan to increase the number of available benchmarks and methods, keeping a strong focus on reproducibility and continual-learning-baselines(link). We also plan to provide better support for reinforcement learning(link), distributed, and federated training support, while bringing the toolkit to maturity and its first stable official release v1.0.0.
Acknowledgments

Antonio Carta acknowledge support from the Ministry of University and Research (MUR) as part of the FSE REACT-EU - PON 2014-2020 “Research and Innovation” resources – Innovation Action - DM MUR 1062/2021.

References


