Continual Learning: On Machines that can Learn Continually

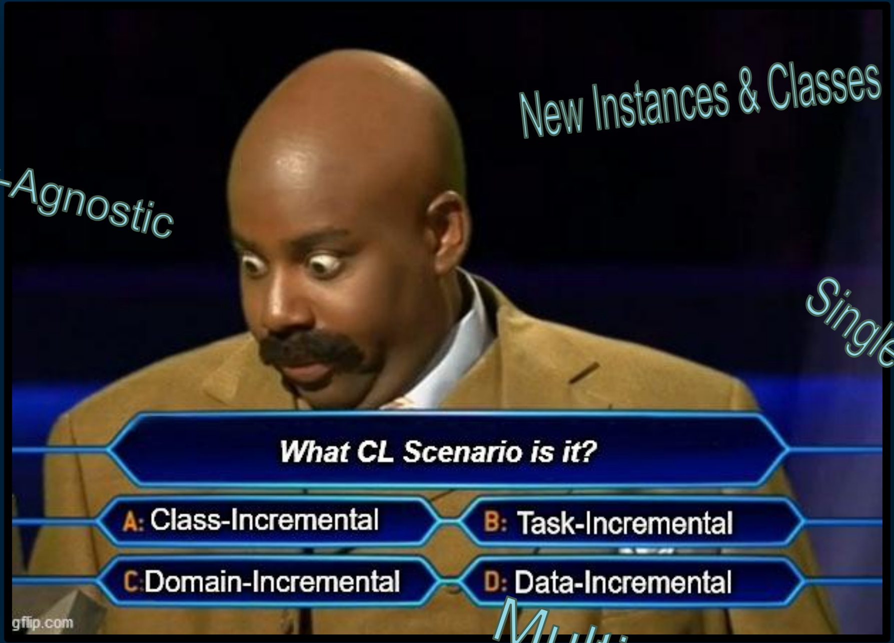
Official Open-Access Course @ University of Pisa, ContinualAI, AIDA

Lecture 3: Scenarios & Benchmarks

Vincenzo Lomonaco
University of Pisa & ContinualAI
vincenzo.lomonaco@unipi.it

TABLE OF CONTENTS

Continual Learning Scenarios



Task-Agnostic

New Instances & Classes

Task-Free

Single-Incremental-Task

Multi-Task

Dataset Shift in Machine Learning

Objectives:

- We want to learn f: X -> Y

Types of Shift:

- Shift in the independent variables (**Covariate Shift**): P(X)
- Shift in the target variable (**Prior probability Shift**): P(Y)
- Shift in the relationship between the independent and the target variable (**Concept Shift**): P(Y|X)

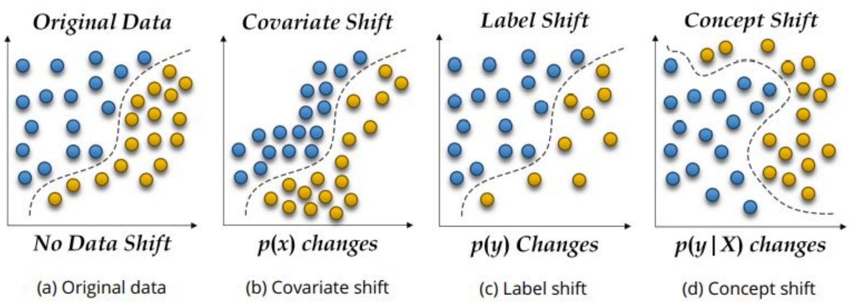
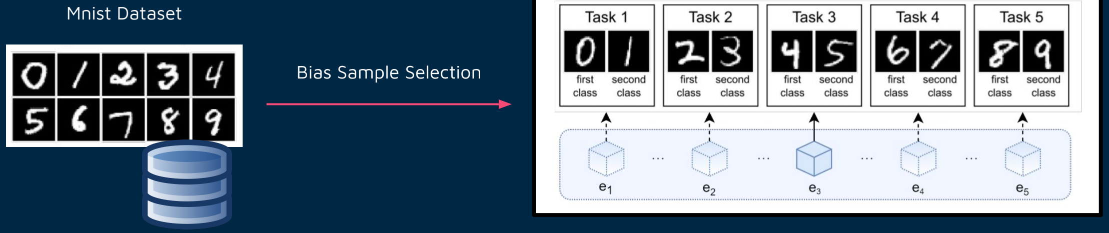
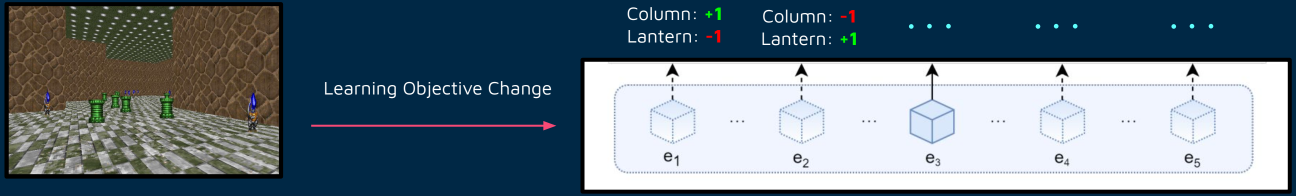


Figure 1: Dataset shift illustration (Similar to Fig. 1 in [4, 5].) Note that the dotted line is the decision boundary between the two classes; i.e., the blue and yellow data points. Here, x represents the input data and y represents the output we aim to predict.

Dataset Shift in Machine Learning, Joaquin Quiñonero-Candela et al, MIT Press Book, 2008.
Understanding Dataset Shift and Potential Remedies, Mehdi Ataei et al. Vector Institute Tech Report, 2021.

Real vs Virtual Shift





Non-stationary Assumptions:

- **Real Shift: the learning objective is changing** (more studied in online learning and AutoML)
- **Virtual Shift: sample selection bias** (Continual Learning today main focus)

no worries: we can just look for **feedback (labels? rewards? heuristics?..)** and approximate the (eventually shifting) learning objective the best way we can.

Dataset Shift in Machine Learning, Joaquin Quiñonero-Candela et al, 2008.
Incremental learning algorithms and applications. Gepperth et al. ESANN, 2016.

Apparently Overwhelming Scenario Proposals

Different Objectives:

- Learn a sequence of **well-defined tasks** in a sequence
- Learn from non-i.i.d data, **small batches**
- Learn **one pattern at a time**
- ....

Different Assumptions:

- Different **Train and Testing Settings**
- **Amount of Supervision** (labels?, task labels?, rewards?)
- **Experiences content** (new classes?, new instances?, ...)
- ...

Common Assumptions

- **Shift is only virtual** (forgetting is not needed, accumulation of knowledge is enough).
- **No conflicting evidence** (we are modeling **mathematical functions**, i.e. to one x there's only one valid y).
- **Unbounded time between two experiences** (you can train as much as you want)
- **Data in each experience can be processed together** (you can shuffle them, process them multiple times, etc.)

Key-Settings and Scenarios

1. **Availability of Task/Distribution Labels**: during training and/or testing
2. **Task/Shift Boundaries**: during training and/or testing
3. **Experience Content**: examples of [same|new] classes
4. **Classification Problem**: [Unique|Partitioned]

	Name	Task Labels	Boundaries	Classes	Problem
3	***Class-Incremental***	no	yes	new	unique
2	***Task-Incremental***	yes	yes	new	partitioned
2	***Domain-Incremental***	no	yes	same	unique
	Task-Free	no	no	any	unique
	Task-Agnostic	no	no	any	partitioned
	…	…	…	…	…

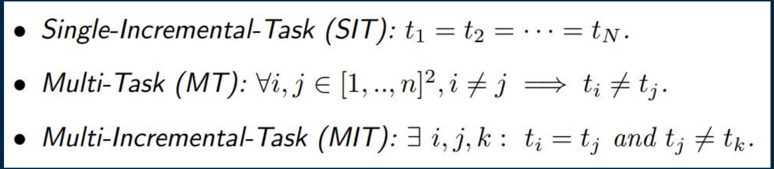
…any combination is possible: check for these assumptions!

A Possible Categorization

Experience content type

Task Labels

	New Instances (NI)	New Classes (NC)	New Instances and Classes (NIC)
Multi-Task	-	Task Incremental	-
Single-Incremental-Task	Domain-Incremental	Class-Incremental	Data-Incremental
Multiple-Incremental-Task	?	?	?



- *Single-Incremental-Task (SIT)*: $t_1 = t_2 = \cdots = t_N$.
- *Multi-Task (MT)*: $\forall i, j \in [1, .., n]^2, i \neq j \implies t_i \neq t_j$.
- *Multi-Incremental-Task (MIT)*: $\exists\ i, j, k :\ t_i = t_j \ and \ t_j \neq t_k$.

- **Defining the notion of a scenario based on what the agent sees <x, y, …, t>.**
- **Unexplored areas** (see "?").
- **Still not comprehensive enough**: *what if you have multiple tasks for one experience? t should be rather a tensor |t| == |y|*

Common Benchmarks

Dataset vs Scenario vs Benchmark

Mnist Dataset

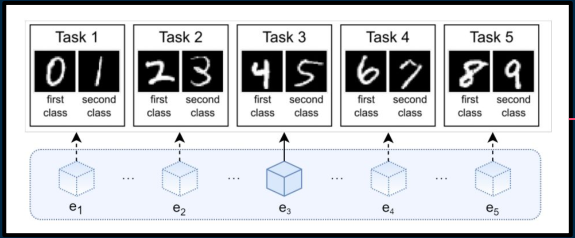
Class-Incremental Learning Scenario

Settings:

1. *Each* ***e*** *contains only examples of new classes never seen before (clear boundaries)*
2. *No* ***t*** *available during train or test.*
3. *Unique Classification problem*

$$+$$

Split Mnist Benchmark



Benchmark Instances

Exact specific sequences and composition of e.
For example:

S1 = e1, e2
S2 = e2, e1

represents two possible **Benchmark Instances** of *Split MNIST*.

Avalanche: An End-to-End Library for Continual Learning. Lomonaco et al. CLVision Workshop at CVPR 2021.

Common CL benchmarks

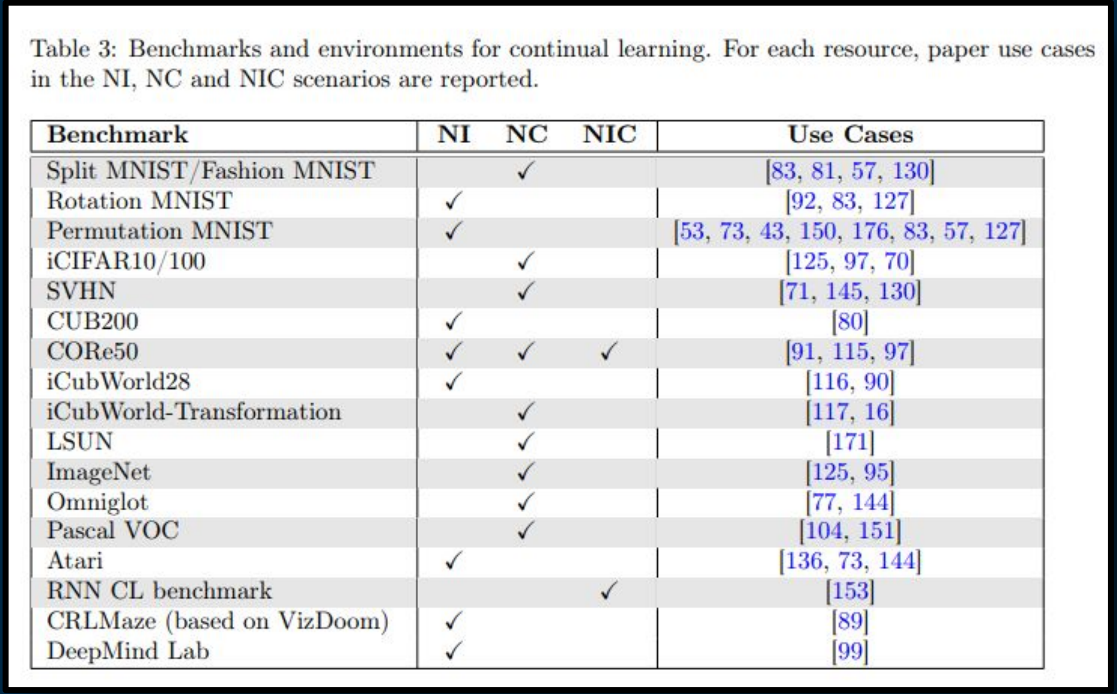


Table 3: Benchmarks and environments for continual learning. For each resource, paper use cases in the NI, NC and NIC scenarios are reported.

Benchmark	NI	NC	NIC	Use Cases
Split MNIST/Fashion MNIST		✓		[83, 81, 57, 130]
Rotation MNIST	✓			[92, 83, 127]
Permutation MNIST	✓			[53, 73, 43, 150, 176, 83, 57, 127]
iCIFAR10/100		✓		[125, 97, 70]
SVHN		✓		[71, 145, 130]
CUB200	✓			[80]
CORe50	✓	✓	✓	[91, 115, 97]
iCubWorld28	✓			[116, 90]
iCubWorld-Transformation		✓		[117, 16]
LSUN		✓		[171]
ImageNet		✓		[125, 95]
Omniglot		✓		[77, 144]
Pascal VOC		✓		[104, 151]
Atari	✓			[136, 73, 144]
RNN CL benchmark			✓	[153]
CRLMaze (based on VizDoom)	✓			[89]
DeepMind Lab	✓			[99]

Past Focus

- **Multi-Task** (Often with Task Supervised Signals)
- **I.I.D by Parts**
- **Few Big Tasks**
- Unrealistic / Toy Datasets
- Mostly Supervised
- Accuracy

Current Focus

- **Class-Incremental Learning**
- **I.I.D by parts**
- **Dozens of experiences**
- Mostly unrealistic / toy datasets
- Mostly supervised
- Accuracy

Is Class-Incremental Enough for Continual Learning?

Why can't we revisit previously seen classes?

- Real-world environments **may naturally include repetition**
- **More repetition □ less forgetting** (replay / rehearsal)
  - CL ≠ no forgetting (many other objectives)
- **Repetition** may be used as **source of information**
  - what is important and what not based on frequencies of occurrences
- Usually repetition allows for longer streams (more experiences)

TL;DR: Class-Incremental with repetition is interesting

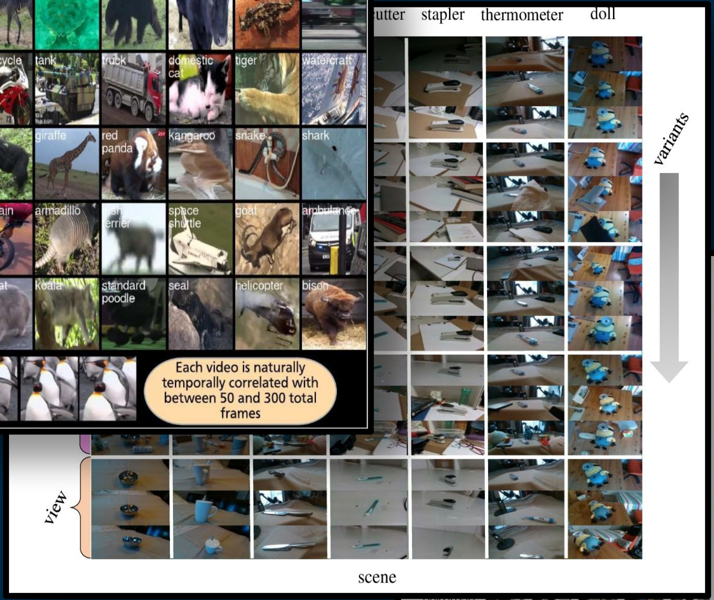
Is Class-Incremental Enough for Continual Learning? Cossu et al, to be published.

What's Next?

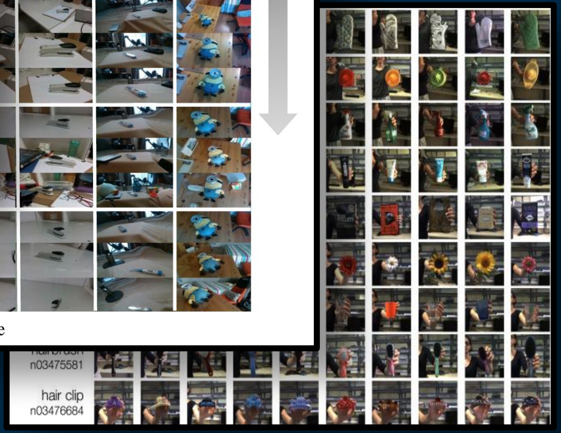
- **Single-Incremental-Task**
- **High-Dimensional Data Streams** (highly non-i.i.d.)
- Natural / Realistic Datasets
- Mostly **Unsupervised**
- **Scalability** and **Efficiency**

Natural Video Benchmarks: the Path Forward?

Stream-51



OpenLoris

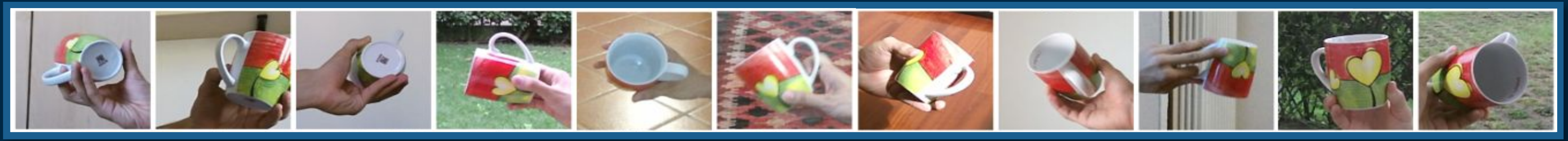


iCub-Transformation

(Roady et al, 2020; She et al, 2020; Pasquale et al, 2019)

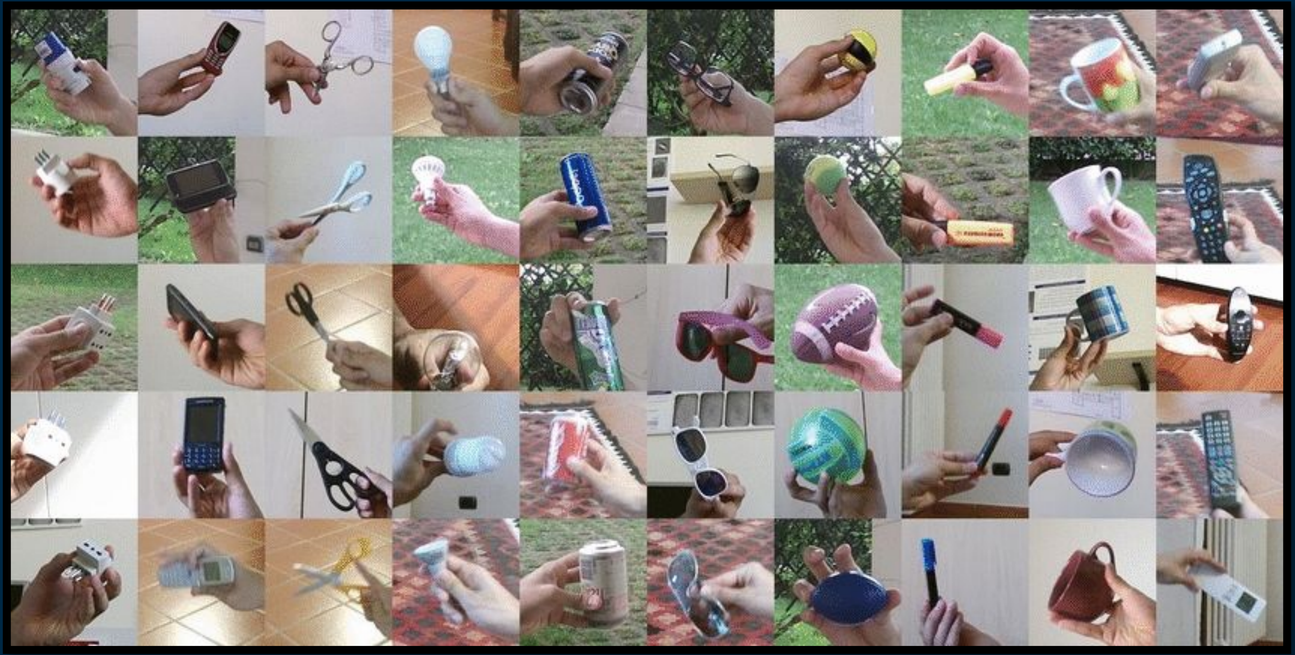
Not only Data Streams but Sequences!

Continual Learning needs the presence of multiple (temporal coherent and unconstrained) views of the same objects taken in different sessions.



Lomonaco V. and Maltoni D. ***CORe50: a New Dataset and Benchmark for Continuous Object Recognition***. CoRL2017.

CORe50: a Video Benchmark for CL and Object Recognition/Detection/Segmentation



Lomonaco V. and Maltoni D. ***CORe50: a New Dataset and Benchmark for Continuous Object Recognition***. CoRL2017.

CORe50: a Video Benchmark for CL and Object Recognition/Detection/Segmentation

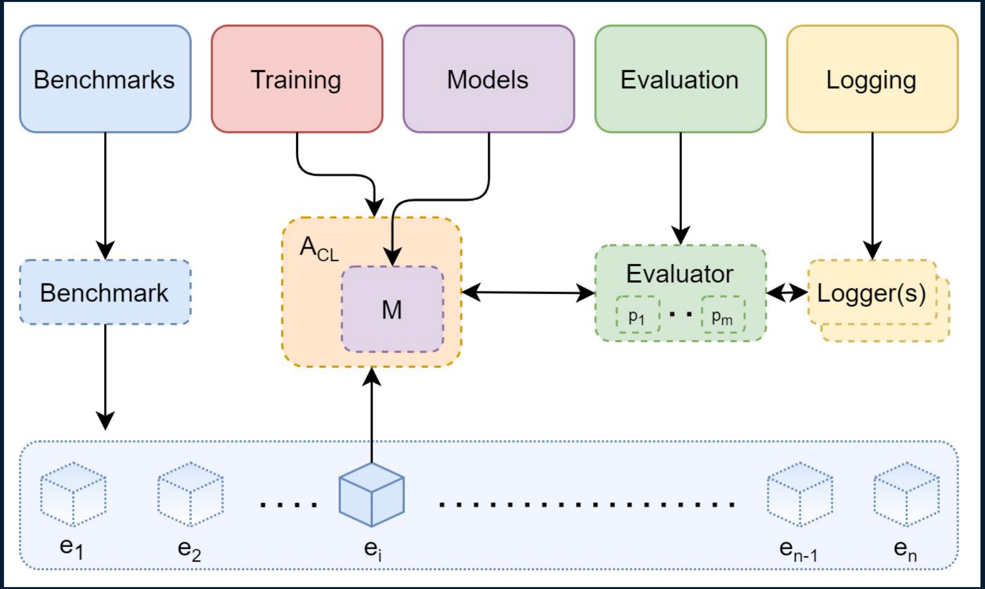
# Images	164,866
Format	RGB-D
Image size	350x350 128x128
# Categories	10
# Obj. x Cat.	5
# Sessions	11
# img. x Sess.	~300
# Outdoor Sess.	3
Acquisition Sett.	Hand held

Lomonaco V. and Maltoni D. ***CORe50: a New Dataset and Benchmark for Continuous Object Recognition***. CoRL2017.

Avalanche Benchmarks

Benchmarks Module

- **Stand-alone**, independent from other modules
- Many out-of-the-box tools
- **Maximum flexibility**
- Exceptional time saver

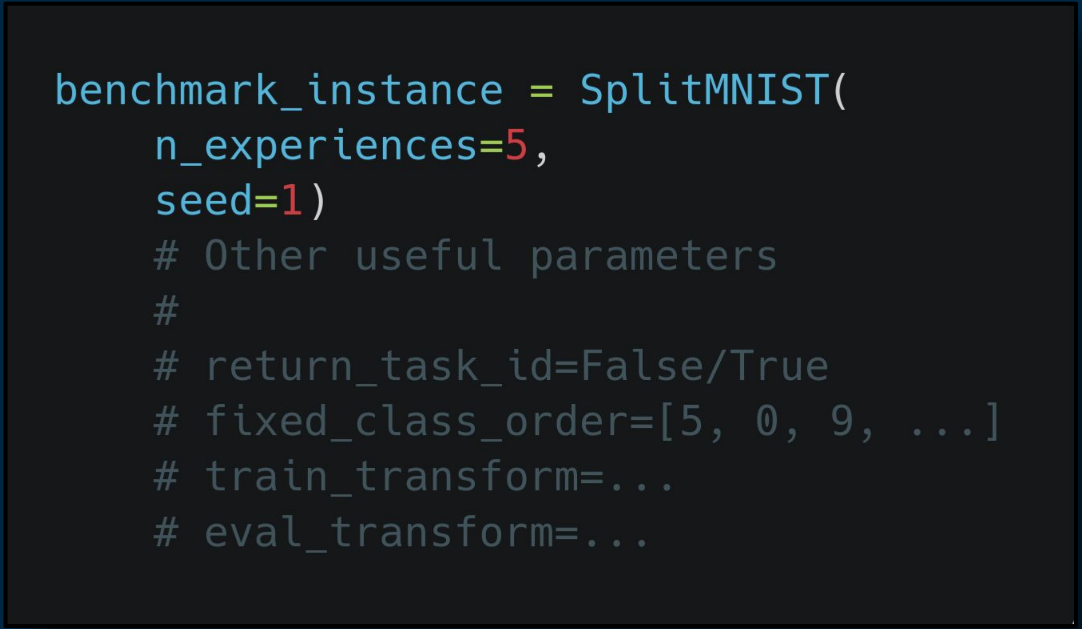


V. Lomonaco et al. ***Avalanche: an End-to-End Library for Continual Learning***. CLVision Workshop at CVPR 2021.

Benchmarks Module

- The benchmarks module offers many tools!
- **Data loading procedures**
- **Generation of data streams**
  - Streams of Experiences
- A lot of out-of-the-box **“classic” benchmarks**
  - SplitMNIST, CIFAR, ImageNet, CUB-200, Stream-51, CORe50, …
- Creation of **custom benchmarks**
  - Maximum compatibility with TorchVision datasets

“Classic” Benchmarks



```
benchmark_instance = SplitMNIST(
    n_experiences=5,
    seed=1)
    # Other useful parameters
    #
    # return_task_id=False/True
    # fixed_class_order=[5, 0, 9, ...]
    # train_transform=...
    # eval_transform=...
```

Streams and Experiences

A **benchmark instance** may be composed of many streams

- Always available: “**train**” and “**test**” streams
- Support for **custom streams**!
    - for instance: validation (A-GEM), out-of-distribution (Steam-51), …

Stream of Experiences, each carrying

- A PyTorch dataset
- Task labels
- Any benchmark-specific data

Benchmark Instance: Basic Loop

```python
train_stream = benchmark_instance.train_stream
test_stream = benchmark_instance.test_stream

for idx, experience in enumerate(train_stream):
    dataset = experience.dataset

    print('Train dataset contains',
        len(dataset), 'patterns')

    for x, y, t in dataset:
        ...

    test_experience = test_stream[idx]
    cumulative_test = test_stream[:idx+1]
```

Custom Benchmarks

Higher-level **Benchmark Generators**: ready to use utilities

- **“New Classes”** (for Class-/Task-Incremental settings)
- **“New Instances”** (for Domain-Incremental settings)

Lower-level Generators: from …

- … Tensors
- … list of files
- … Caffe-style filelists
- … custom PyTorch datasets

Higher Level API: SplitMNIST

```
# Nearly all datasets from torchvision are supported

mnist_train = MNIST('./mnist', train=True)
mnist_test = MNIST('./mnist', train=False)

benchmark_instance = nc_benchmark(
    train_dataset=mnist_train,
    test_dataset=mnist_test,
    n_experiences=n_experiences,
    task_labels=True/False)
```

Benchmarks: Maximum Flexibility

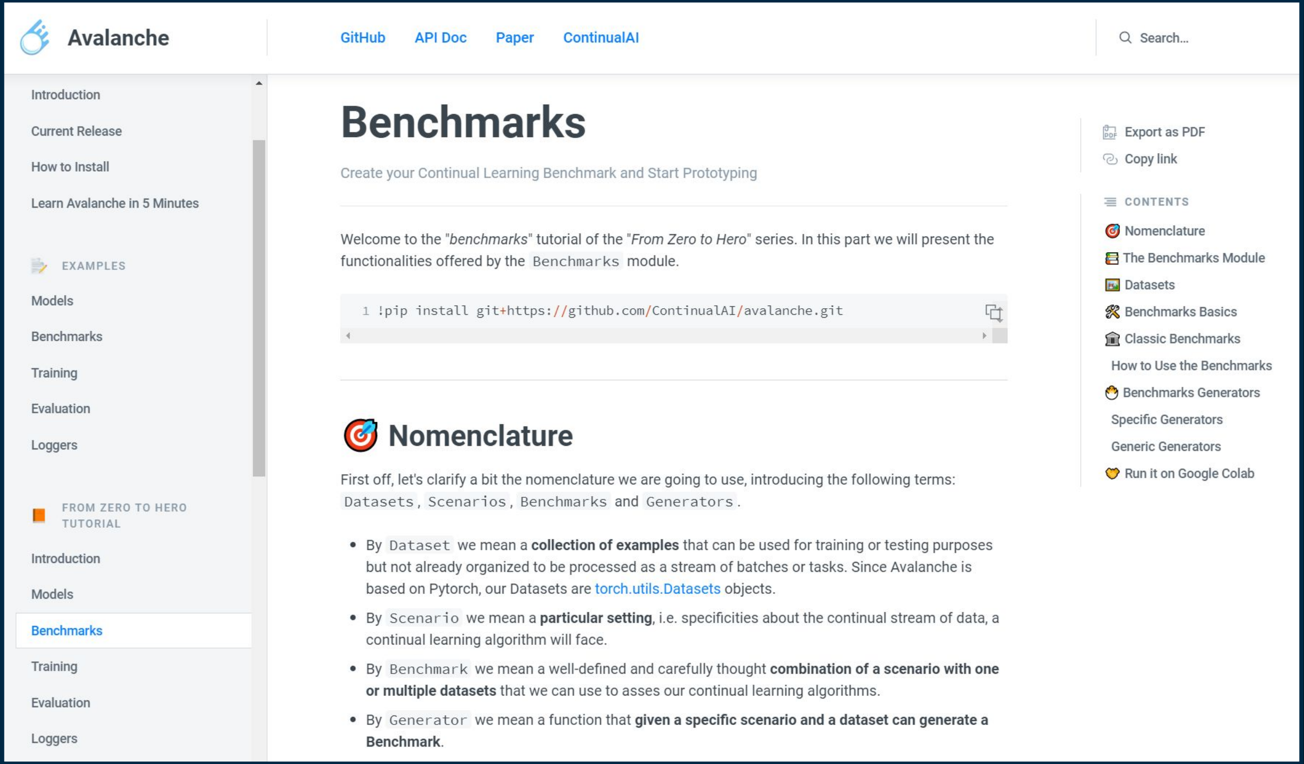
- Mechanisms, internal aspects, name of components are independent w.r.t. the presence of task labels
  - **No forced nomenclature**
- Choices regarding task labels are left to the benchmark creator
- Task labels can be defined at pattern granularity
- Easy to create **complex setups in a simple way**

Benchmarks: Next Steps

- Integration of **new classic benchmarks** (contributions are welcome!)
- Not only classification: Regression, segmentation
- Not only Vision Datasets
- Object Detection (on their way)
- Even **more tools for defining custom benchmarks**

Avalanche Benchmarks

Demo Session!



Avalanche

GitHub API Doc Paper ContinualAI

Search...

Introduction
Current Release
How to Install
Learn Avalanche in 5 Minutes

EXAMPLES
Models
Benchmarks
Training
Evaluation
Loggers

FROM ZERO TO HERO TUTORIAL
Introduction
Models
Benchmarks
Training
Evaluation
Loggers

Benchmarks

Create your Continual Learning Benchmark and Start Prototyping

Welcome to the "*benchmarks*" tutorial of the "*From Zero to Hero*" series. In this part we will present the functionalities offered by the `Benchmarks` module.

```
!pip install git+https://github.com/ContinualAI/avalanche.git
```

Nomenclature

First off, let's clarify a bit the nomenclature we are going to use, introducing the following terms: `Datasets`, `Scenarios`, `Benchmarks` and `Generators`.

- By `Dataset` we mean a **collection of examples** that can be used for training or testing purposes but not already organized to be processed as a stream of batches or tasks. Since Avalanche is based on Pytorch, our Datasets are torch.utils.Datasets objects.
- By `Scenario` we mean a **particular setting**, i.e. specificities about the continual stream of data, a continual learning algorithm will face.
- By `Benchmark` we mean a well-defined and carefully thought **combination of a scenario with one or multiple datasets** that we can use to asses our continual learning algorithms.
- By `Generator` we mean a function that **given a specific scenario and a dataset can generate a Benchmark**.

Export as PDF
Copy link

CONTENTS

Next:

Evaluation & Metrics

Do you have any questions?

vincenzo.lomonaco@unipi.it
vincenzolomonaco.com
University of Pisa

THANKS