Continual Learning: On Machines that can Learn Continually

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

Lecture 4: Evaluation & Metrics

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

TABLE OF CONTENTS

Evaluation Protocols

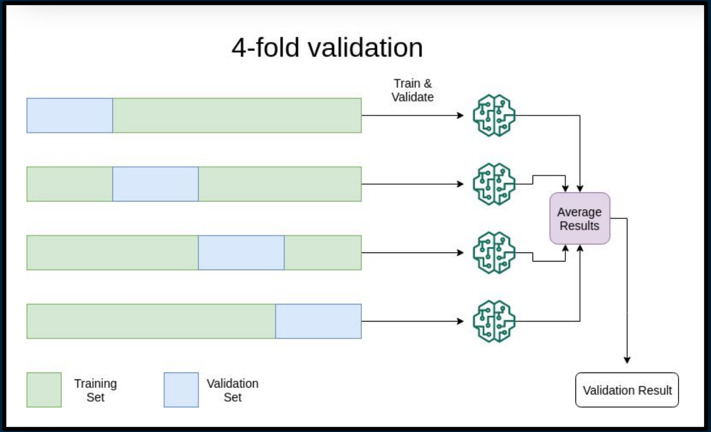
Classic ML Evaluation

Train - Validation - Test split

- **Model selection**: train on training set, eval on validation set
- **Model assessment**: train on training (+ validation) set, eval on test set

Variations allowed

- **K-fold** Cross-Validation
- Leave-one-out
- ...



Test, training and validation sets (brainstobytes.com)

Evaluating Machine Learning Models, by Alice Zheng, 2015.

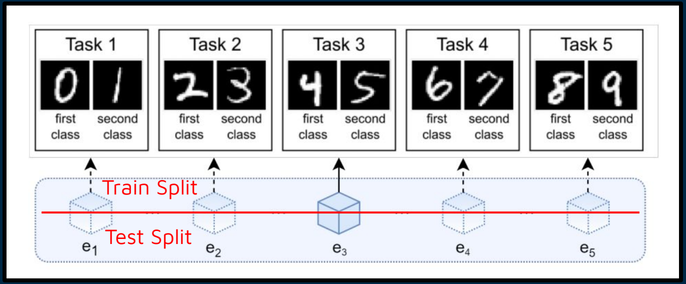
Basic CL Evaluation Protocol

Different Data

- Classic *Machine Learning* -> **static dataset**
- *Continual Learning* -> **stream of datasets** (experiences)

A Simple Extension to CL

- **Split by patterns**: one train-(validation)-test per experience (or *parallel streams* of experiences)
- This is the **simplest** and **most common** evaluation protocol



The objective of a CL algorithm is to minimize the loss $\mathcal{L}_S$ over the entire stream of data S:

$$\mathcal{L}_S(f_n^{CL}, n) = \frac{1}{\sum_{i=1}^{n} |\mathcal{D}_{test}^i|} \sum_{i=1}^{n} \mathcal{L}_{exp}(f_n^{CL}, \mathcal{D}_{test}^i) \tag{2}$$

$$\mathcal{L}_{exp}(f_n^{CL}, \mathcal{D}_{test}^i) = \sum_{j=1}^{|\mathcal{D}_{test}^i|} \mathcal{L}(f_n^{CL}(\boldsymbol{x}_j^i), y_j^i), \tag{3}$$

where the loss $\mathcal{L}(f_n^{CL}(\boldsymbol{x}), y)$ is computed on a single sample $\langle \boldsymbol{x}, y \rangle$, such as cross-entropy in classification problems.

Split by Patterns

- **Training phase**: train the model on **training sets** of each experience, sequentially
- **Test phase**: evaluate the model on **the sets** of the experiences (order does not matter)
- Examples in the training and test sets **are disjoint**!
- We may have a **single test set** or **one for each experience**
- **Multiple evaluation streams are possible** (Valid, Test, Out-of-Distribution, etc.)
- **Cross-Validation** & **Hyper-parameters** selection can be operated based on the final aggregate metric at the end of the training.

When and What to Test On

When to test?

- **At the end of each experience**, usually.
- A finer granularity is always possible (*epochs*, *iterations*, etc.)

On what to test?

- **Current** experience
- **Future** experiences
- **Past** experiences
- **All** experiences
- …

…depending also on the metrics you want to use!

Growing vs Fixed Test Set

Growing Test Set

- We consider only the test set of the **current and previously encountered** experiences
- Compute the performance metrics average over those

Fixed test set

- Common for some benchmarks
- Clear view on **overall system performance**
- Recover experience-wise performance, if needed

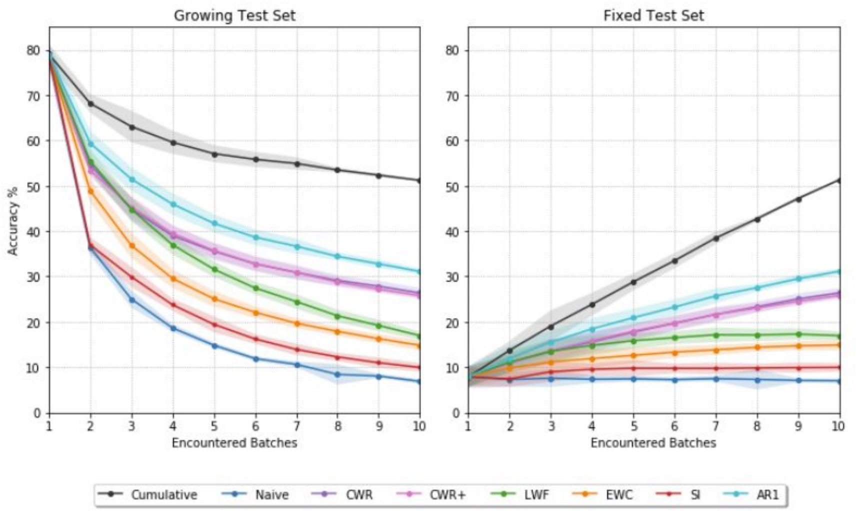
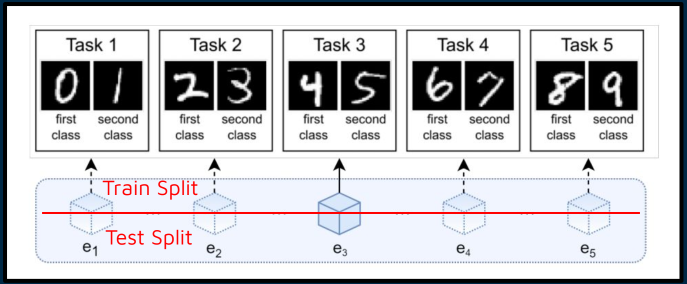


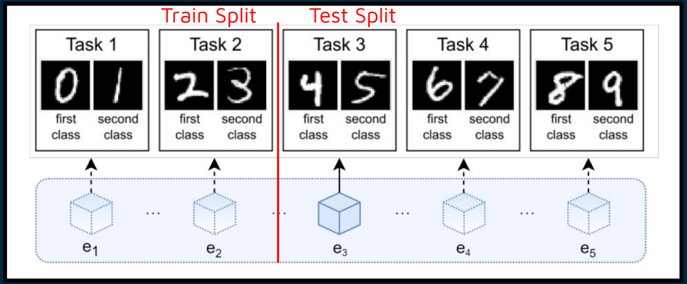
Figure 8: Accuracy on iCIFAR-100 with 10 batches (10 classes per batch). Results are averaged on 10 runs: for all the strategies hyperparameters have been tuned on run 1 and kept fixed in the other runs. The experiment on the right, consistently with CORe50 test protocol, considers a fixed test set including all the 100 classes, while on the left we include in the test set only the classes encountered so far (analogously to results reported in [28]). Colored areas represent the standard deviation of each curve. Better viewed in color.

Continuous Learning in Single-Incremental-Task Scenarios. Maltoni & Lomonaco, Neural Networks Journal 2019.

Is it Enough for CL?

- **Split by patterns**: one train-validation-test per experience (or *parallel streams* of experiences)
- *But is it enough for Continual Learning? ->* we would like a way to evaluate if we are **actually able** to learn continually!
- **Split by experiences**: **model selection** on a first set of experiences, **model assessment** on a second set of experiences
- Model assessment **should also involve training**.





Efficient Lifelong Learning with A-GEM Chaudhry et. al. ICLR, 2019.

Hyper-parameters Selection for CL

- We mentioned **Hyper-parameters** selection can be operated based on the final aggregate metric at the end of the training
- But this may be seen as **a form of cheating**: we select the best hyperparameters that maximize the the performance on a specific sequence of training experiences
- We may partially solve this with **several runs** with a **random order** of the training experiences
- This may be still **unfair**: we should calibrate hyper-parameters on a **limited set of experiences**

Class-incremental learning: survey and performance evaluation on image classification. Masana et al. 2020.
A continual learning survey: Defying forgetting in classification tasks. De Lange et al, 2019.

A more Articulated Protocol: An Example

- **Model selection**: train the model on a first split of experiences, select best *hyperparameters* with a cross-validation scheme.

- **Model assessment**: train & evaluate the CL algorithm on a second split of experiences

Algorithm 1 Learning and Evaluation Protocols

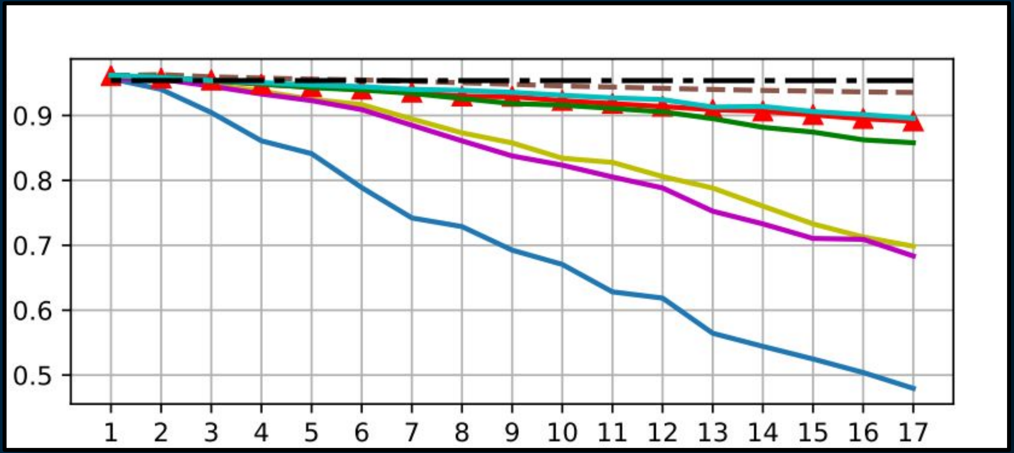
for h **in** hyper-parameter list **do** ▷ Cross-validation loop, executing multiple passes over $\mathcal{D}^{CV}$
  for $k = 1$ **to** T^{CV} **do** ▷ Learn over data stream $\mathcal{D}^{CV}$ using h
    for $i = 1$ **to** n_k **do** ▷ Single pass over $\mathcal{D}_k$
      Update f_θ using $(\mathbf{x}_i^k, t_i^k, y_i^k)$ and hyper-parameter h
      Update metrics on test set of $\mathcal{D}^{CV}$
    end for
  end for
end for
Select best hyper-parameter setting, h^*, based on average accuracy of test set of $\mathcal{D}^{CV}$, see Eq. 1.
Reset f_θ.
Reset all metrics.
for $k = T^{CV} + 1$ **to** T **do** ▷ Actual learning over datastream $\mathcal{D}^{EV}$
  for $i = 1$ **to** n_k **do** ▷ Single pass over $\mathcal{D}_k$
    Update f_θ using $(\mathbf{x}_i^k, t_i^k, y_i^k)$ and hyper-parameter h^*
    Update metrics on test set of $\mathcal{D}^{EV}$
  end for
end for
Report metrics on test set of $\mathcal{D}^{EV}$.

Efficient Lifelong Learning with A-GEM Chaudhry et. al. ICLR, 2019.

Continual Learning
Metrics

What to Monitor?

- Performance on **current experience**
- Performance on **past experiences**
- Performance on **future experiences**
- **Resource consumption**
  (Memory / CPU / GPU / Disk usage)
- **Model size growth**
  (with respect to the first model)
- Execution **time**
- **Data efficiency**
- …



Gradient Episodic Memory for Continual Learning, Lopez-Paz et al. NIPS, 2017.
Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges, Lesort et al. Information Fusion, 2020.

Accuracy

Q: How accurate is my model?

In many different sauces

- *Accuracy on the current experience*
- *Accuracy on previous experiences (plus the current one)*
- *Accuracy on future experiences (plus the current one)*

ACC Metric

- After training on all experiences, **average accuracy** over all the test experiences.

A Metric

- Average of the accuracy on all experiences **at any point in time**.

R	Te_1	Te_2	Te_3
Tr_1	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$
Tr_2	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$
Tr_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$

$$\textbf{Average Accuracy: } \mathrm{ACC} = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}$$

$$A = \frac{\sum_{i=1}^{N}\sum_{j=1}^{i} R_{i,j}}{\frac{N(N+1)}{2}}$$

Don't forget, there is more than forgetting: new metrics for Continual Learning, Rodriguez-Diaz et al. CL Workshop @ NeurIPS, 2018.
Gradient Episodic Memory for Continual Learning. Lopez-Paz & Ranzato, NeurIPS 2017.

Forward Transfer

Q: How much learning the current experience improves my performance on future experiences?

FWT Metric

- Accuracy on experience *i* after training on last experience Minus
- Accuracy on experience *i* before training on the first experience (model init)
- Averaged over $i=2,\dots,T$

R	Te_1	Te_2	Te_3
Tr_1	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$
Tr_2	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$
Tr_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$

$$\text{FWT} = \frac{1}{T-1}\sum_{i=2}^{T} R_{i-1,i} - \bar{b}_i.$$

Gradient Episodic Memory for Continual Learning. Lopez-Paz & Ranzato, NeurIPS 2017.

Backward Transfer

Q: How much learning the current experience improves my performance on previous experiences?

BWT Metric

- Accuracy on experience *i* after training on experience T Minus
- Accuracy on experience *i* after training on experience *i*
- Averaged over i=1,...,T-1

FORGETTING = - BWT

R	Te_1	Te_2	Te_3
Tr_1	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$
Tr_2	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$
Tr_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$

$$\text{BWT} = \frac{1}{T-1} \sum_{i=1}^{T-1} R_{T,i} - R_{i,i}$$

Gradient Episodic Memory for Continual Learning. Lopez-Paz & Ranzato, NeurIPS 2017.

Memory

Not only performance

- **How much space does your model occupy**? (MB, # of params, etc.)
- **What is the increment** in space required for each new experience?
- How much space do you require for **additional information** (replay buffer, past models…)?

$$MS = min(1, \frac{\sum_{i=1}^{N} \frac{Mem(\theta_1)}{Mem(\theta_i)}}{N})$$

$$SSS = 1 - min(1, \frac{\sum_{i=1}^{N} \frac{Mem(M_i)}{Mem(D)}}{N})$$

Don't forget, there is more than forgetting: new metrics for Continual Learning, Rrodriguez-Diaz et al. CL Workshop @ NeurIPS, 2018.

Computation

Not only performance

- What is the **computational overhead** during training? (# MACs, Running Time, GPU/CPU time, ...)
- What about its **increment over time**?
- What is the computational overhead **during inference**?

$$CE = min(1, \frac{\sum_{i=1}^{N} \frac{Ops\uparrow\downarrow(Tr_i)\cdot\varepsilon}{1+Ops(Tr_i)}}{N})$$

Don't forget, there is more than forgetting: new metrics for Continual Learning, Rrodriguez-Diaz et al. CL Workshop @ NeurIPS, 2018.

Don't Forget: There is More than Forgetting!

A plethora of other possible metrics

- *Accuracy vs offline baseline*
- *Model Robustness*
- *Model Plasticity & Capacity*
- *...*

More complex Score Functions

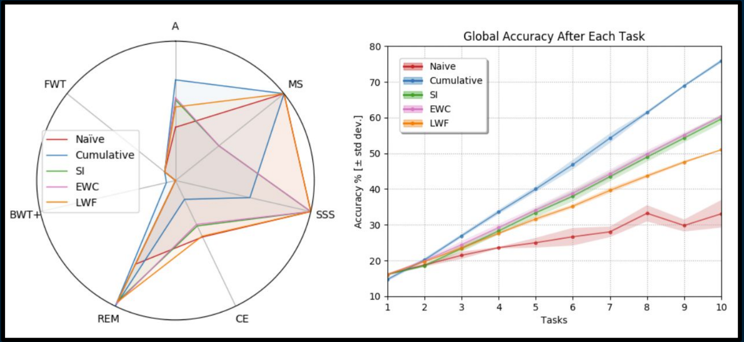
- Additional, more informative **derived metrics** can be devised as well.
- They can be **tuned depending on the specific application goals**.

$$CL_{score} = \sum_{i=1}^{\#\mathcal{C}} w_i c_i$$

$$CL_{stability} = 1 - \sum_{i=1}^{\#\mathcal{C}} w_i stddev(c_i)$$

Table 1: CL metrics and CL_{score} for each CL strategy evaluated (higher is better).

Strategy	A	REM	BWT⁺	FWT	MS	SSS	CE	CL_{score}	$CL_{stability}$
Naïve	0.3825	0.6664	0.0000	0.1000	**1.0000**	**1.0000**	**0.4492**	0.5140	**0.9986**
Cumul.	**0.7225**	**1.0000**	**0.0673**	0.1000	**1.0000**	0.5500	0.1496	0.5128	0.9979
EWC	0.5940	0.9821	0.0000	0.1000	0.4000	**1.0000**	0.3495	0.4894	0.9972
LWF	0.5278	0.9667	0.0000	0.1000	**1.0000**	**1.0000**	0.4429	**0.5768**	**0.9986**
SI	0.5795	0.9620	0.0000	0.1000	0.4000	**1.0000**	0.3613	0.4861	0.9970



[Don't forget, there is more than forgetting: new metrics for Continual Learning](), Rrodriguez-Diaz et al. CL Workshop @ NeurIPS, 2018.

Summing Up

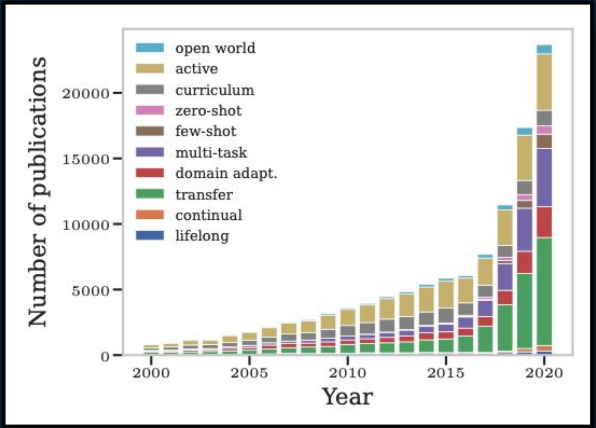
- Choose an evaluation protocol and **declare** it (no *standard*, yet)
- Choose the metrics you monitor wisely (**what are you interested in?**)
- **Do not focus exclusively on performance metrics**, if possible

Q: you can achieve low/zero forgetting by occupying a lot of space. How?

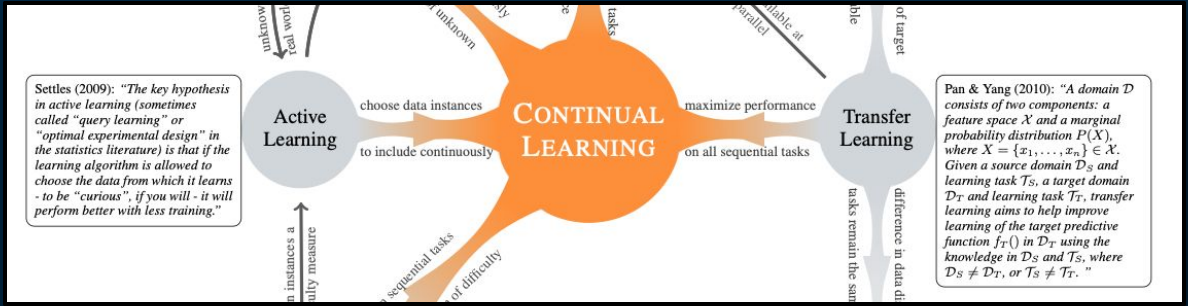
Q: which metrics would you monitor to evaluate a continual learner deployed and trained on the edge on image classification tasks?

CLEVA-Compass: A Continual Learning EValuation Assessment Compass

Recall lecture 1: there are various machine learning formulations that have continuous components



Depending on **where inspiration** has been drawn from, continual learning setups and **evaluation can vary dramatically**.



(a small snapshot from the overall paradigm relationships)

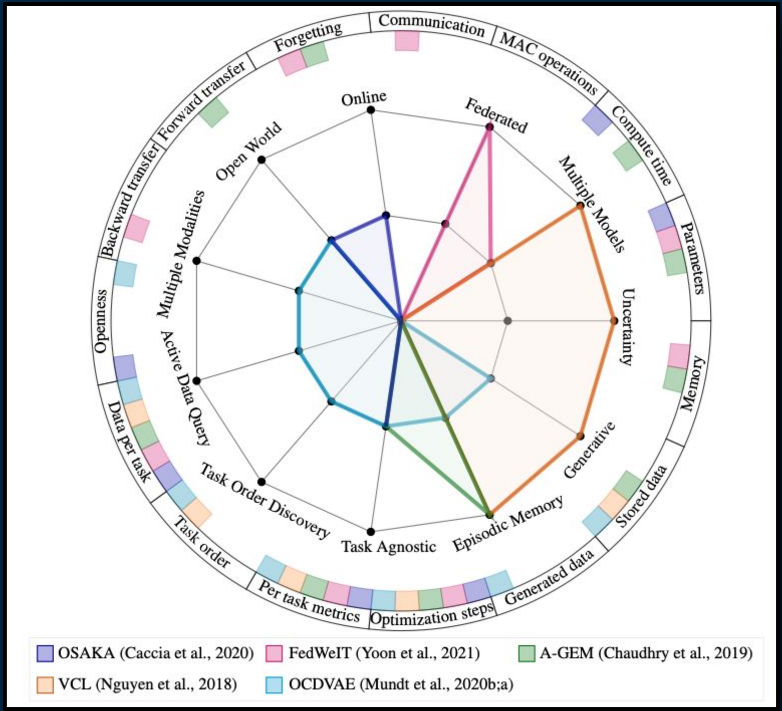
CLEVA-Compass: A Continual Learning EValuation Assessment Compass to Promote Research Transparency and Comparability, arXiv, 2021.

CLEVA-Compass: A Continual Learning EValuation Assessment Compass

Existence of various scenarios is not a problem, but actually meaningful because different applications can desire different things!

But **reproducibility & comparability can be problematic**, which is a constant subject in the scientific literature.

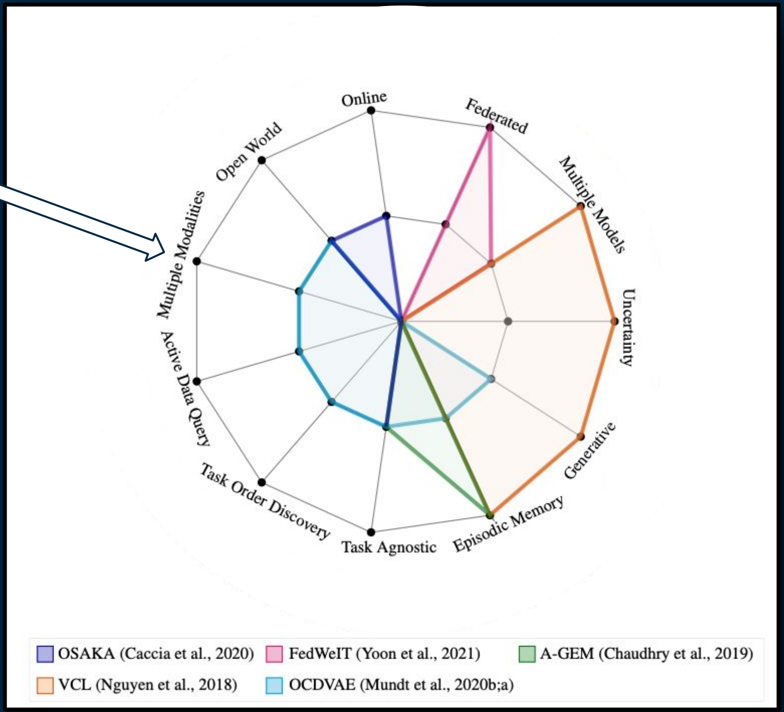
Recently, the **CLEVA-Compass** has been introduced to **promote transparency & comparability**



CLEVA-Compass: A Continual Learning EValuation Assessment Compass to Promote Research Transparency and Comparability, arXiv, 2021.

CLEVA-Compass: A Continual Learning EValuation Assessment Compass

Inner compass level (star plot):
indicates related paradigm inspiration & continual setting configuration (assumptions)

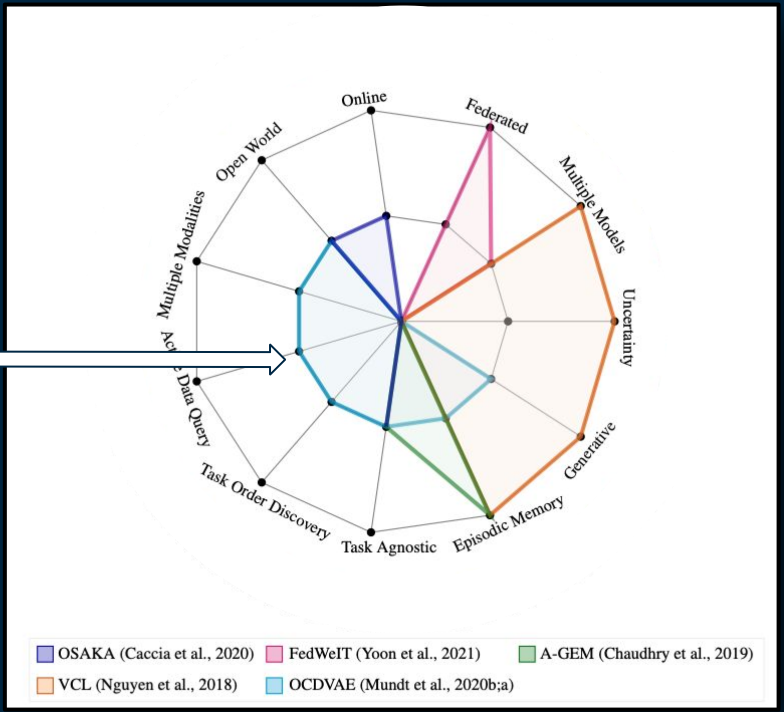


CLEVA-Compass: A Continual Learning EValuation Assessment Compass to Promote Research Transparency and Comparability, arXiv, 2021.

CLEVA-Compass: A Continual Learning EValuation Assessment Compass

Inner compass level (star plot):
indicates related paradigm inspiration & continual setting configuration (assumptions)

Inner compass level of supervision:
"rings" on the star plot indicate presence of supervision. Importantly: supervision is individual to each dimension!



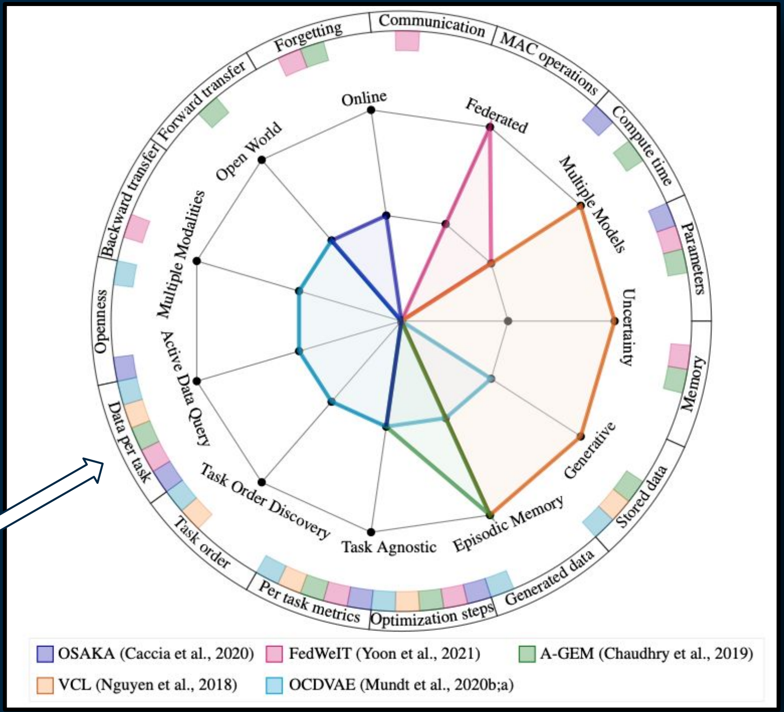
CLEVA-Compass: A Continual Learning EValuation Assessment Compass to Promote Research Transparency and Comparability, arXiv, 2021.

CLEVA-Compass: A Continual Learning EValuation Assessment Compass

Inner compass level (star plot):
indicates related paradigm inspiration & continual setting configuration (assumptions)

Inner compass level of supervision:
“rings” on the star plot indicate presence of supervision. Importantly: supervision is individual to each dimension!

Outer compass level:
Contains a comprehensive set of practically reported measures



CLEVA-Compass: A Continual Learning EValuation Assessment Compass to Promote Research Transparency and Comparability, arXiv, 2021.

Avalanche Metrics & Loggers

How to Monitor Experiments?

Evaluation module provides

- **Metrics** (accuracy, forgetting, CPU Usage…) - *you can create your own!*
- **Loggers** to report results in different ways - *you can create your own!*
- **Automatic integration in the training and evaluation loop** through the Evaluation Plugin
- **A dictionary with all recorded metrics** always available for custom use

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

Let’s Track our Experiments

```python
from avalanche.logging import InteractiveLogger, TextLogger, \
    TensorboardLogger
from avalanche.training.plugins import EvaluationPlugin
from avalanche.evaluation.metrics import ExperienceForgetting, \
    accuracy_metrics, loss_metrics, cpu_usage_metrics


eval_plugin = EvaluationPlugin(
  accuracy_metrics(minibatch=True, stream=True),
  loss_metrics(epoch=True, experience=True),
  ExperienceForgetting(),
  cpu_usage_metrics(stream=True),
  # add as many metrics as you need
  loggers=[TextLogger(open('out.txt', 'w')),
           InteractiveLogger(),
           TensorboardLogger()])

# just pass the evaluation plugin to the strategy
# strategy = EWC(..., evaluator=eval_plugin)

# {'metric name': [x values], [metric values]}
# empty since we have not trained, yet
metric_dict = eval_plugin.get_all_metrics()
```

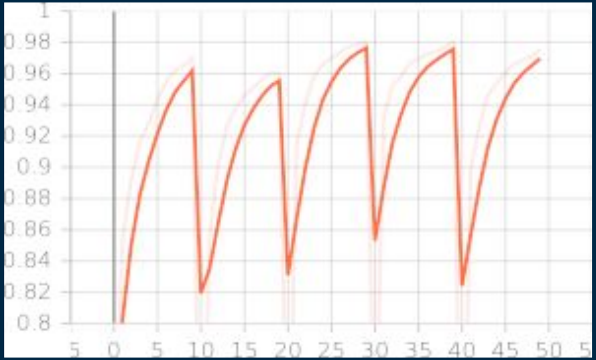
V. Lomonaco et al. ***Avalanche: an End-to-End Library for Continual Learning***. CLVision Workshop at CVPR 2021.

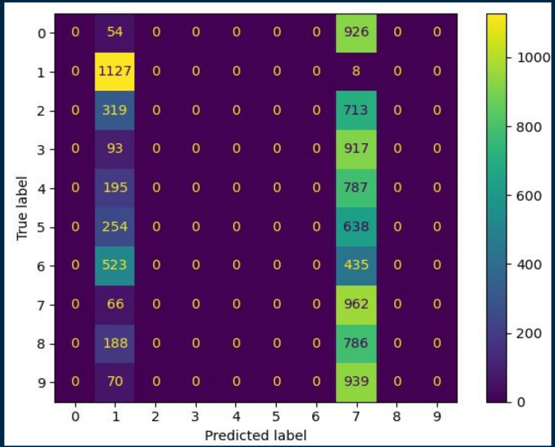
Interactive Logger Output

```
-- >> Start of training phase << --
-- Starting training on experience 0 (Task 0) from train stream --
Epoch 0 ended.
  Loss_Epoch/train_phase/train_stream/Task000 = 1.1099
  Top1_Acc_Epoch/train_phase/train_stream/Task000 = 0.8926
...
-- >> End of training phase << --
-- >> Start of eval phase << --
-- Starting eval on experience 0 (Task 0) from test stream --
> Eval on experience 0 (Task 0)  from test stream ended.
     Loss_Exp/eval_phase/test_stream/Task000/Exp000 = 0.0208
     Top1_Acc_Exp/eval_phase/ test_stream/Task000/Exp000 = 0.9981
...
-- >> End of eval phase << --
     Loss_Stream/eval_phase/test_stream = 4.4492
```

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

Tensorboard Logger in Action





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

Standalone Metrics

```python
import torch
from avalanche.evaluation.metrics import Accuracy

# create an instance of the standalone Accuracy metric
acc_metric = Accuracy()
print("Initial Accuracy: ", acc_metric.result())  # output 0

# two consecutive metric updates
real_y = torch.tensor([1, 2]).long()
predicted_y = torch.tensor([1, 0]).float()
acc_metric.update(real_y, predicted_y)
acc = acc_metric.result()
print("Average Accuracy: ", acc)  # output 0.5

predicted_y = torch.tensor([1,2]).float()
acc_metric.update(real_y, predicted_y)
acc = acc_metric.result()
print("Average Accuracy: ", acc)  # output 0.75

# reset accuracy to 0
acc_metric.reset()
print("After reset: ", acc_metric.result())  # output 0
```

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

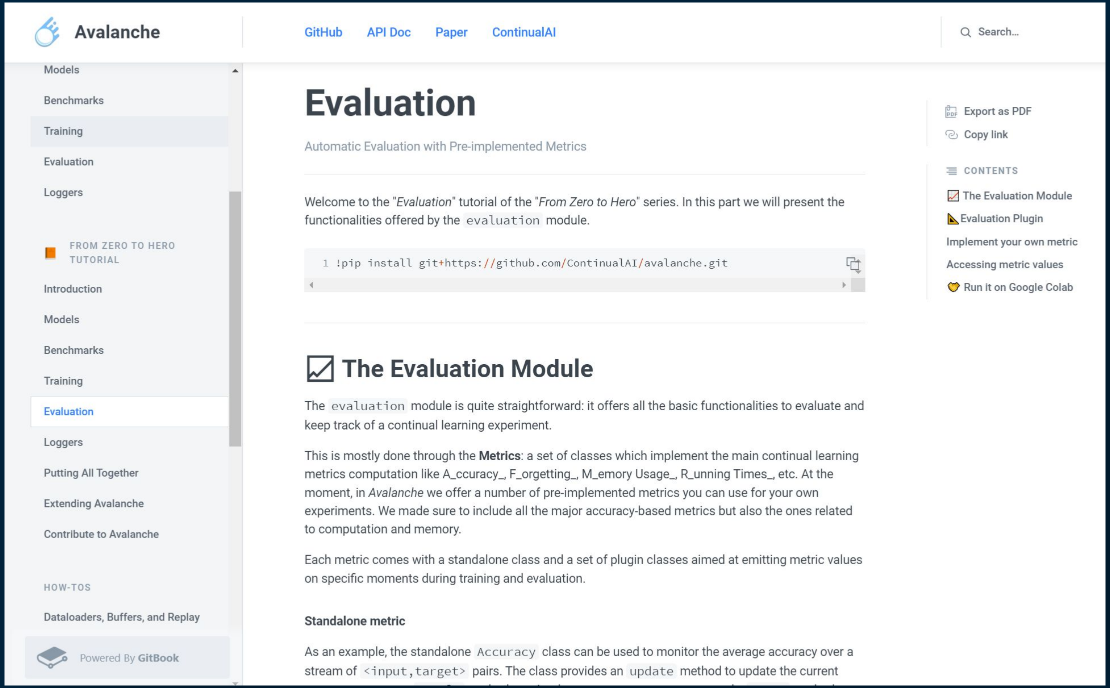
What's Next?

- Evaluation of a CL algorithm is not only about metrics and loggers.
- More support for the definition of **training and evaluation protocols**
    - How to perform **cross validation** in CL?
    - How to evaluate **multiple runs**?
- The objective of a **shared protocol** is possible only with the help of the community

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

Avalanche Evaluation Module

Demo Session!



Avalanche

GitHub API Doc Paper ContinualAI

Search...

Models
Benchmarks
Training
Evaluation
Loggers

FROM ZERO TO HERO TUTORIAL

Introduction
Models
Benchmarks
Training
Evaluation
Loggers
Putting All Together
Extending Avalanche
Contribute to Avalanche

HOW-TOS

Dataloaders, Buffers, and Replay

Powered By GitBook

Evaluation

Automatic Evaluation with Pre-implemented Metrics

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

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

The Evaluation Module

The `evaluation` module is quite straightforward: it offers all the basic functionalities to evaluate and keep track of a continual learning experiment.

This is mostly done through the **Metrics**: a set of classes which implement the main continual learning metrics computation like A_ccuracy_, F_orgetting_, M_emory Usage_, R_unning Times_, etc. At the moment, in *Avalanche* we offer a number of pre-implemented metrics you can use for your own experiments. We made sure to include all the major accuracy-based metrics but also the ones related to computation and memory.

Each metric comes with a standalone class and a set of plugin classes aimed at emitting metric values on specific moments during training and evaluation.

Standalone metric

As an example, the standalone `Accuracy` class can be used to monitor the average accuracy over a stream of `<input,target>` pairs. The class provides an `update` method to update the current

Export as PDF
Copy link

CONTENTS

The Evaluation Module
Evaluation Plugin
Implement your own metric
Accessing metric values
Run it on Google Colab

[Avalanche, From Zero To Hero tutorial: Evaluation], [Avalanche, From Zero To Hero tutorial: Loggers]

Next:

Methodologies [Part 1]

Do you have any questions?

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

THANKS