Continual Learning: On Machines that can Learn Continually

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

Lecture 2: Understanding Catastrophic Forgetting

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

TABLE OF CONTENTS

What's Forgetting?

A Concrete Example

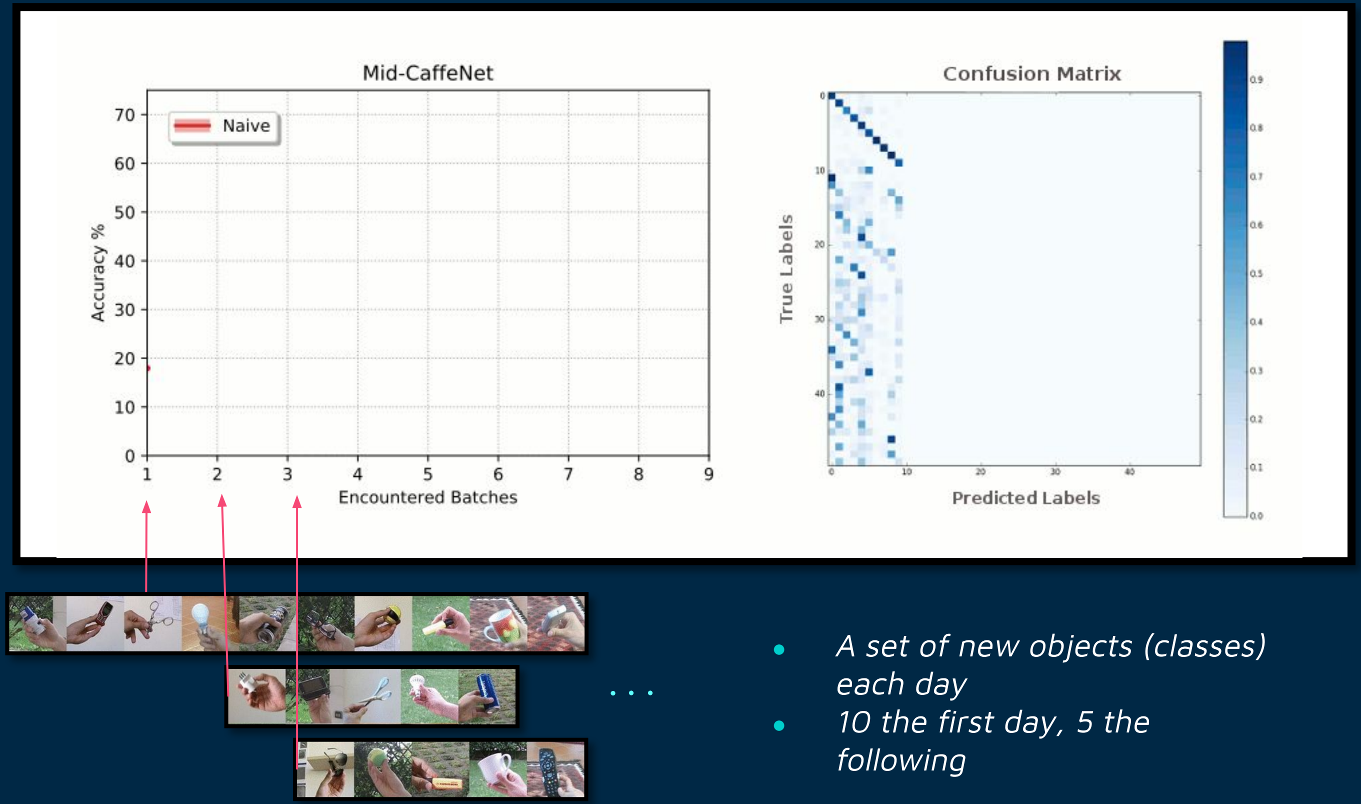
- **50GB/s** streaming data.
- **~30240 TB of data** after only a week.
- **Impossible** to re-train the mini-spot brain from scratch and to **adapt fast**.

[Spot® - The Agile Mobile Robot | Boston Dynamics]

R1 Example from *Istituto Italiano di Tecnologia*

Teaching iCub to recognize objects using deep Convolutional Neural Networks. Giulia Pasquale et al., IROS 2016.

Continual Learning: what’s the problem?



- *A set of new objects (classes) each day*
- *10 the first day, 5 the following*

[CORe50: a new Dataset and Benchmark for Continuous Object Recognition](), V. Lomonaco & D. Maltoni. Conference on Robot Learning (CoRL), 2017.

The Stability-Plasticity Dilemma

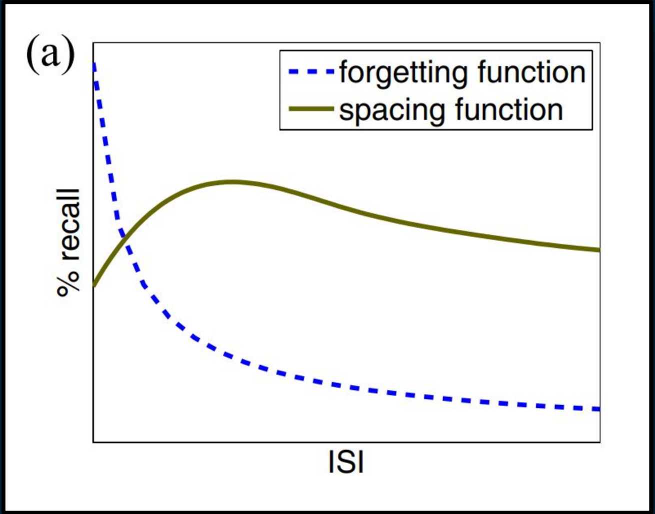
Stability-Plasticity Dilemma:

- **Remember past concepts**
- **Learn new concepts**
- **Generalize**

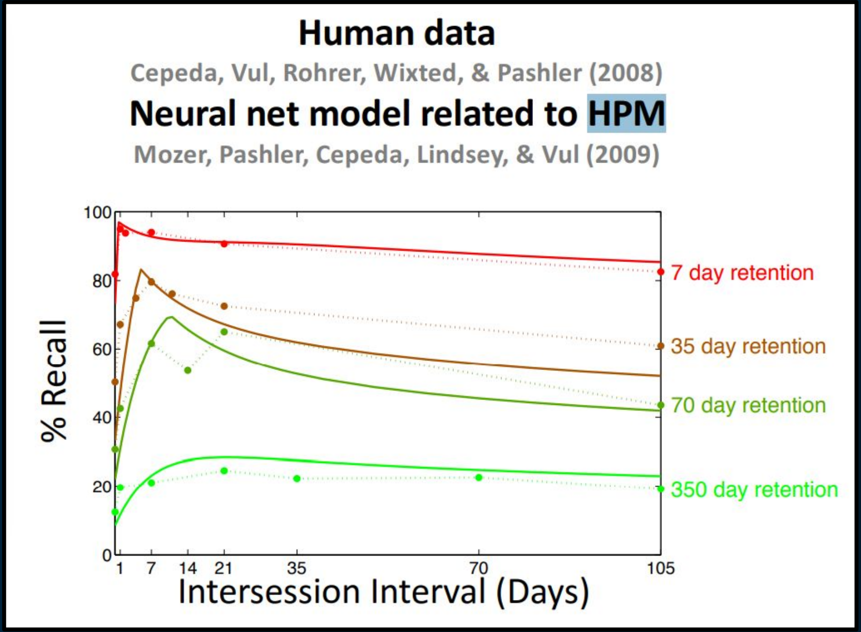
First Problem in Deep Learning:

- **Catastrophic Forgetting**

Forgetting in Humans



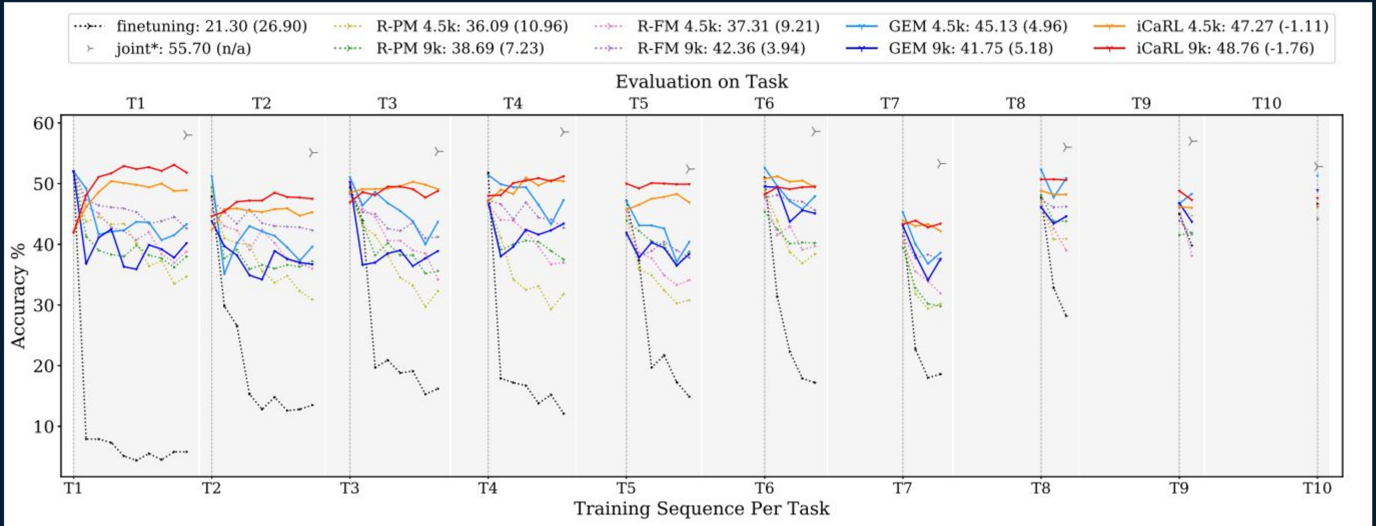
The spacing function (solid line) depicts recall at test following two study sessions separated by a given intersession interval (ISI); the forgetting function (dashed line) depicts recall as a function of the lag between study and test.



Predicting the optimal spacing of study: A multiscale context model of memory. M. Mozer et al. NeurIPS, 2009.

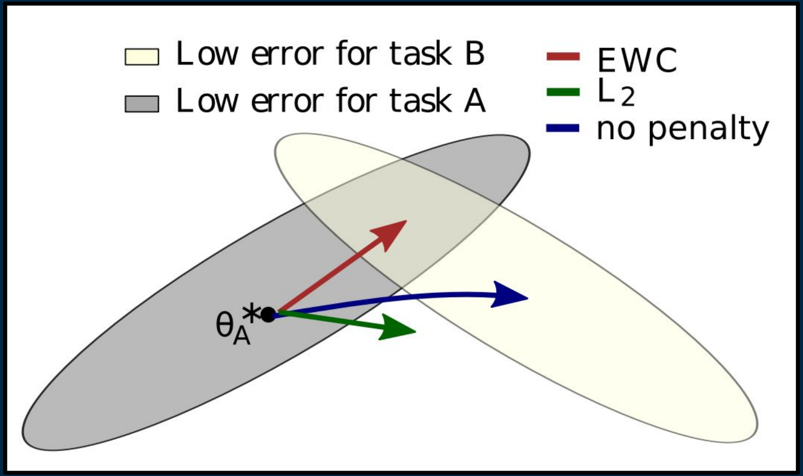
Forgetting in Machines

*Catastrophic interference, also known as **catastrophic forgetting**, is the tendency of an artificial neural networks to completely and abruptly forget previously learned information upon learning new information. ->* Mostly due to Gradient Descent.



A continual learning survey: Defying forgetting in classification tasks. Matthias Delange et al. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

Forgetting in Machines



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

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

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

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.

1. *We don't have access to previously encountered data.*
2. *$\mathcal{L}_S$ can only be approximated.*

Overcoming catastrophic forgetting in neural networks. James Kirkpatrick et al. Proceedings of the national academy of sciences, 2017.

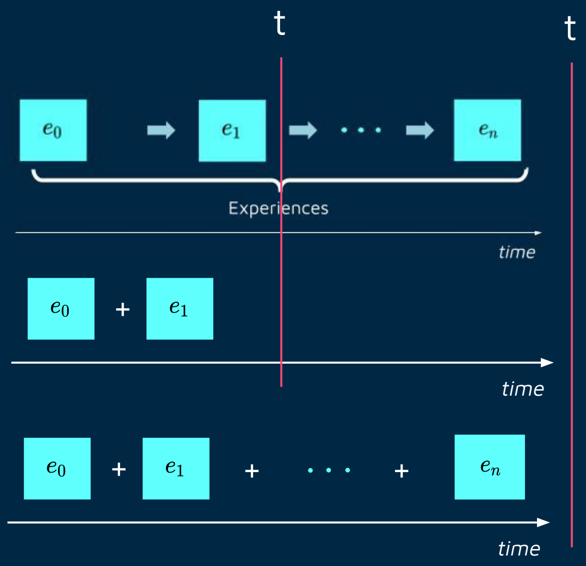
Classic ML vs CL

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

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

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

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.

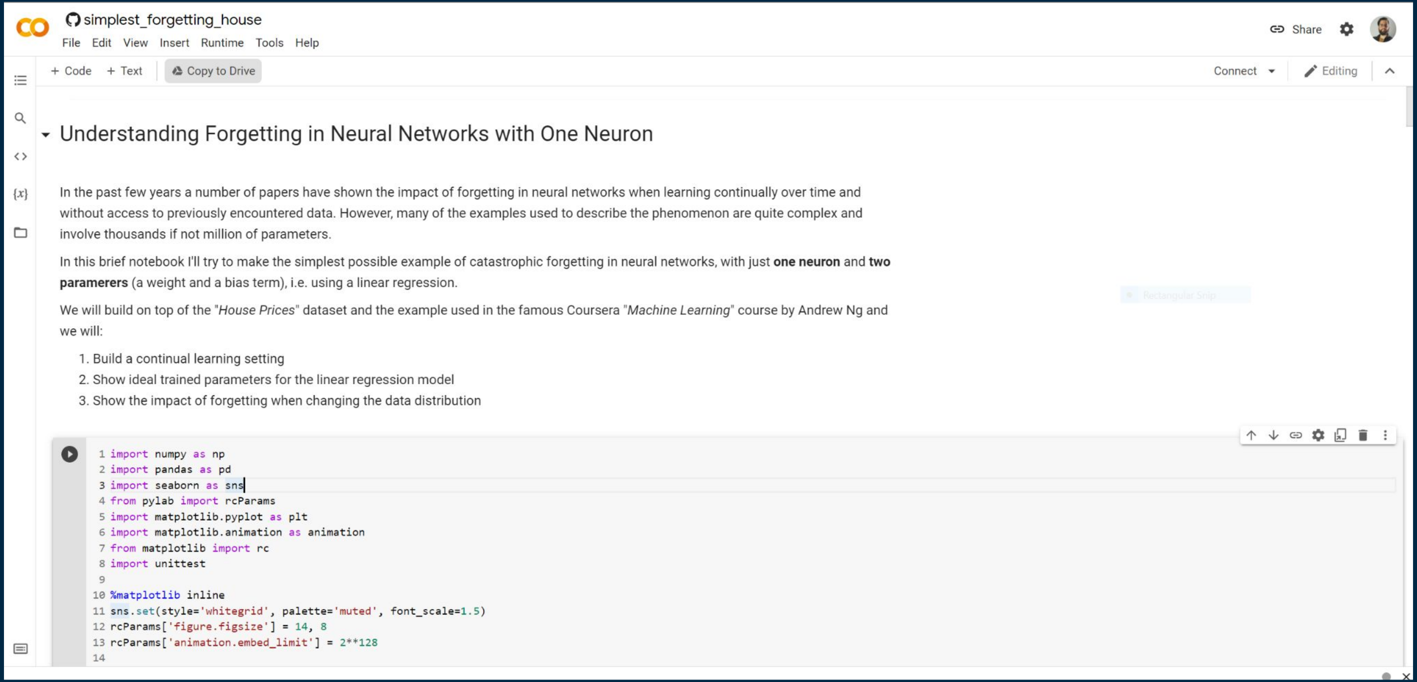


- *Given a time t, you can always transform the continual learning problem in a static one.*
- *Continual Learning is mostly about leveraging previously learned knowledge, since t is unbounded.*
- *More efficiency (low & bounded memory and computation) -> more forgetting.*

Forgetting with 1 Neuron

House Prices Estimation Example

Demo Session!



Understanding Forgetting in Neural Networks with One Neuron

In the past few years a number of papers have shown the impact of forgetting in neural networks when learning continually over time and without access to previously encountered data. However, many of the examples used to describe the phenomenon are quite complex and involve thousands if not million of parameters.

In this brief notebook I'll try to make the simplest possible example of catastrophic forgetting in neural networks, with just **one neuron** and **two paramerers** (a weight and a bias term), i.e. using a linear regression.

We will build on top of the "*House Prices*" dataset and the example used in the famous Coursera "*Machine Learning*" course by Andrew Ng and we will:

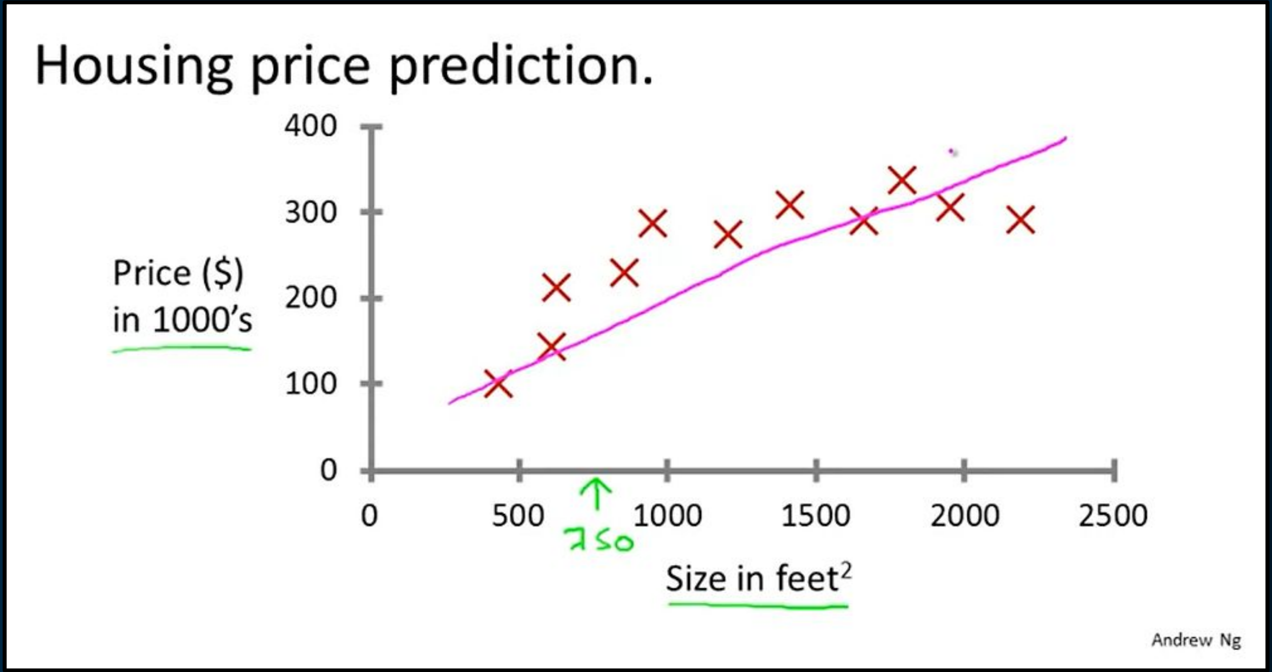
1. Build a continual learning setting
2. Show ideal trained parameters for the linear regression model
3. Show the impact of forgetting when changing the data distribution

```python
import numpy as np
import pandas as pd
import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import rc
import unittest

%matplotlib inline
sns.set(style='whitegrid', palette='muted', font_scale=1.5)
rcParams['figure.figsize'] = 14, 8
rcParams['animation.embed_limit'] = 2**128

```

https://github.com/ContinualAI/colab

House Prices Estimation Example

Demo Session!



House Prices Estimation Example

Demo Session!

$$y = \sum_j w_j x_j + b.$$

$$\mathcal{L}(y, t) = \frac{1}{2}(y - t)^2.$$

$$\begin{aligned}
\mathcal{E}(w_1, \ldots, w_D, b) &= \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y^{(i)}, t^{(i)}) \\
&= \frac{1}{2N} \sum_{i=1}^{N} \left(y^{(i)} - t^{(i)}\right)^2 \\
&= \frac{1}{2N} \sum_{i=1}^{N} \left(\sum_j w_j x_j^{(i)} + b - t^{(i)}\right)^2
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{E}}{\partial w_j} &= \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)} (y^{(i)} - t^{(i)}) \\
\frac{\partial \mathcal{E}}{\partial b} &= \frac{1}{N} \sum_{i=1}^{N} y^{(i)} - t^{(i)}.
\end{aligned}$$

Just add a “dummy” input x0 which always takes the value 1; then the weight w0 acts as a bias.

$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{E}}{\partial w_j}.$$

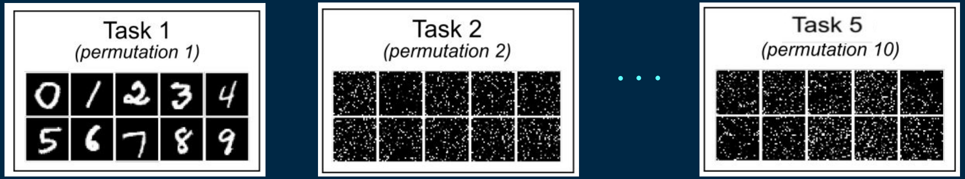
$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \mathbf{X}^\top (\mathbf{y} - \mathbf{t}),$$

Linear Regression notes by Roger Grosse, University of Toronto, 2017.

MNIST: a Classic Computer Vision Example

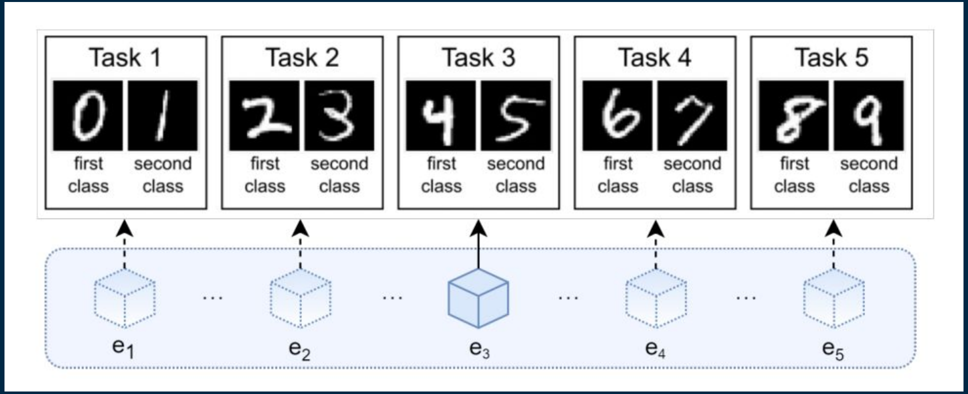
Permuted MNIST and SplitMINIST

Demo Session!



Permuted MNIST

or



Split MNIST

https://github.com/ContinualAI/colab

How to Solve Forgetting?

How to Solve Forgetting?

Brainstorming Session!

- Can we store just a portion of the previously encountered data?
- Can we make an ensemble of models?
- **Can we freeze some parameters of the network?**
- Have the model parameters the same importance? Can we leverage this to avoid forgetting?
- How model topology effects forgetting? What about pre-trained networks?
- ...?

CL: Not Only Forgetting

- Catastrophic forgetting is just a self-evident prominent failure of current ML systems
- **Continual Learning is much more!**
- **Efficiency of Learning (memory, computation)**
- **Backward and Forward transfer**
- **Compositionality**
- **Robustness**
- **Learning to Learn**
- **...**

Intro to Avalanche

Avalanche: an End-to-End Library for CL

Avalanche is an **End-to-End Continual Learning Library** based on PyTorch, born within **ContinualAI** with the unique goal of providing a **collaborative** and community-driven open-source (MIT licensed) codebase for **fast prototyping**, **training** and **reproducible evaluation** of continual learning algorithms.

Avalanche can help Continual Learning researchers and practitioners in several ways:

- *Write less code, prototype faster & reduce errors.*
- *Improve reproducibility.*
- *Improve modularity and reusability.*
- *Increase code efficiency, scalability & portability.*
- *Augment impact and usability of your research products.*

Lomonaco V. et al., ***Avalanche: an End-to-End Library for Continual Learning***, CLVision Workshop at CVPR2021.

Avalanche Design Principles

1. Comprehensiveness and Consistency
2. Ease-of-Use
3. Low Constraints / Assumptions
4. Reproducibility and Portability
5. Modularity and Independence
6. Efficiency and Scalability
7. Community-driven

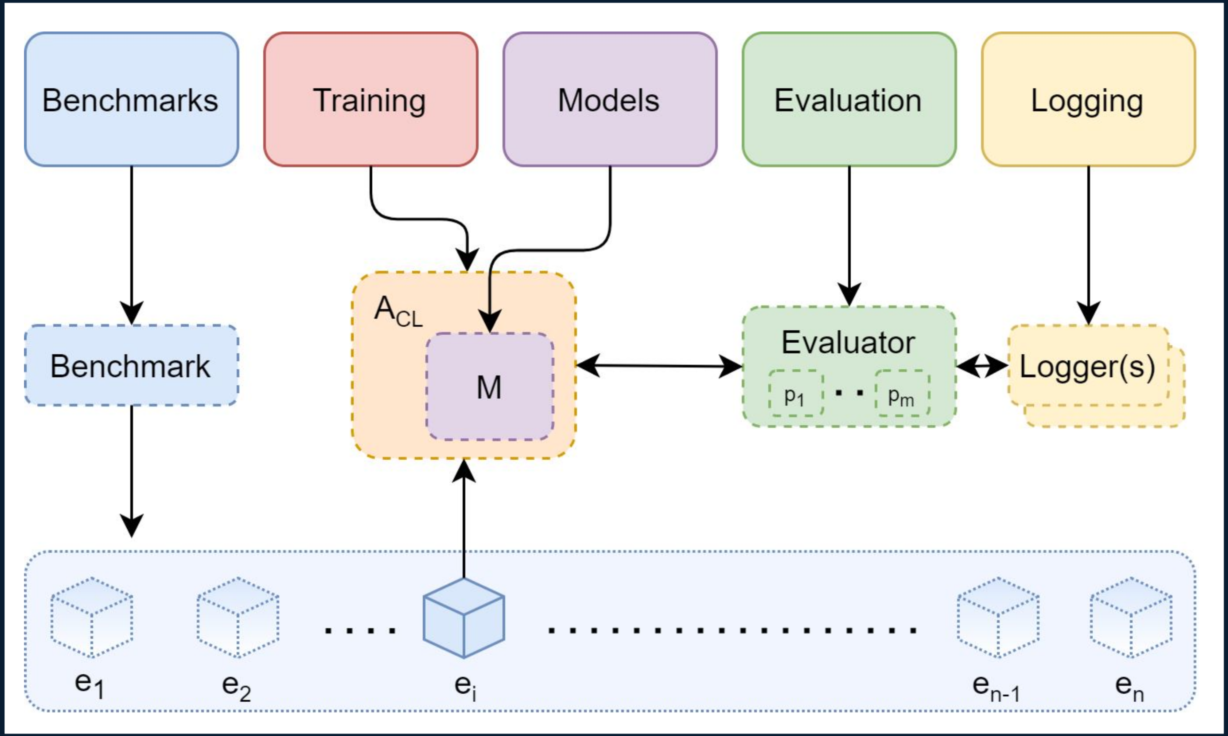
Lomonaco V. et al., ***Avalanche: an End-to-End Library for Continual Learning***, CLVision Workshop at CVPR2021.

Introduction to Avalanche

- 5 main maintainers (Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti)
- 40+ contributors and top CL players around the world (more on the people here)
- 15+ organizations and leading CL labs from Europe, USA, China.
- Integrating different codebases (like FACIL)
- Huge Sponsorships on the way…

Lomonaco V. et al., ***Avalanche: an End-to-End Library for Continual Learning***, CLVision Workshop at CVPR2021.

Avalanche in a Nutshell



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

Avalanche: an End-to-End Library for CL

Avalanche Key Links

- ***Avalanche Official Website***: https://avalanche.continualai.org
- ***Avalanche GitHub***: https://github.com/ContinualAI/avalanche
- ***Avalanche API-DOC***: https://avalanche-api.continualai.org
- ***Avalanche ContinualAI Slack***: #avalanche channel

With Avalanche | Without Avalanche

```python
import torch
from torch.nn import CrossEntropyLoss
from torch.optim import SGD

from avalanche.benchmarks.classic import PermutedMNIST
from avalanche.extras.models import SimpleMLP
from avalanche.training.strategies import Naive

# Config
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

# model
model = SimpleMLP(num_classes=10)

# CL Benchmark Creation
perm_mnist = PermutedMNIST(n_experiences=3)
train_stream = perm_mnist.train_stream
test_stream = perm_mnist.test_stream

# Prepare for training & testing
optimizer = SGD(model.parameters(), lr=0.001, momentum=0.9)
criterion = CrossEntropyLoss()

# Continual learning strategy
cl_strategy = Naive(
    model, optimizer, criterion, train_mb_size=32, train_epochs=2,
    eval_mb_size=32, device=device)

# train and test loop
results = []
for train_task in train_stream:
    cl_strategy.train(train_task, num_workers=4)
    results.append(cl_strategy.eval(test_stream))
```

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

Next:

Scenarios & Benchmarks

Do you have any questions?

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

THANKS