Continual Learning

Scenarios, evaluation, and metrics

Antonio Carta

antonio.carta@unipi.it

Outline

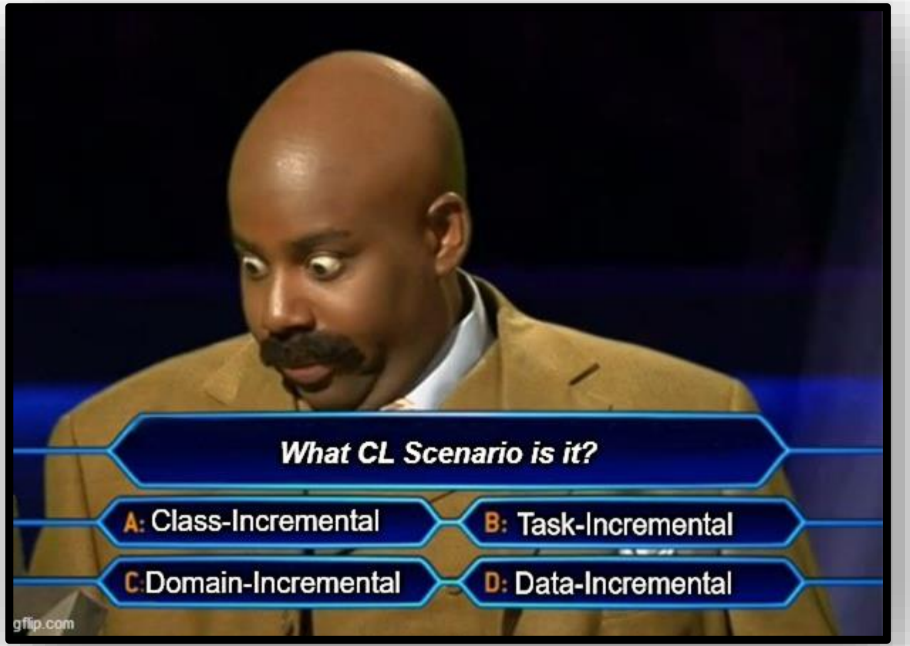
CL Scenarios
- CL assumptions
- Types of drifts
- Nomenclature
- Benchmarks examples

Evaluation
- CL eval vs prequential
- Hyperparameter selection

Metrics
- What and when to monitor
- Accuracy
- Forgetting
- Backward/forward transfer
- Computational performance

Task-Agnostic

New Instances & Classes



Task-Free

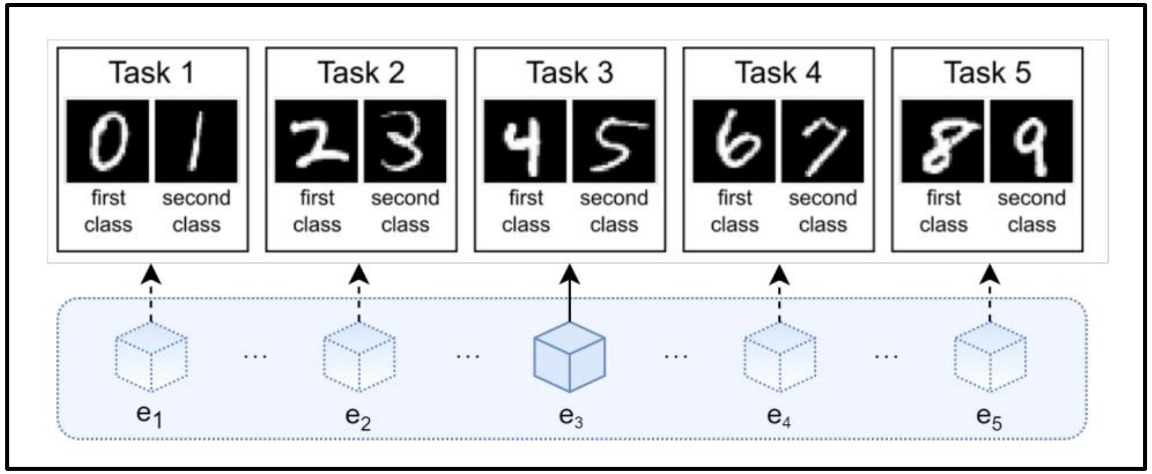
Single-Incremental-Task

Multi-Task

What is a CL Scenario?

A common nomenclature that:

- A set of **CL metrics** that we want to optimize
- A set of constraints that the **learning algorithm** must satisfy
- A restricted form of access to the data through a sequential **data stream**



Given the type of scenario and its constraints we can identify a proper strategy to learn it.

Continual Learning - Definition

In continual learning (CL) data arrives in a streaming fashion as a (possibly infinite) sequence of learning experiences $S = e_1, \dots, e_n$. For a supervised classification problem, each experience e_i consists of a batch of samples $\mathcal{D}^i$, where each sample is a tuple $\langle x_k^i, y_k^i \rangle$ of input and target, respectively, and the labels y_k^i are from the set $\mathcal{Y}^i$, which is a subset of the entire universe of classes $\mathcal{Y}$. Usually $\mathcal{D}^i$ is split into a separate train set $\mathcal{D}^i_{train}$ and test set $\mathcal{D}^i_{test}$.

A continual learning algorithm $\mathcal{A}^{CL}$ is a function with the following signature:

$$\mathcal{A}^{CL} : \langle f_{i-1}^{CL}, \mathcal{D}^i_{train}, \mathcal{M}_{i-1}, t_i \rangle \rightarrow \langle f_i^{CL}, \mathcal{M}_i \rangle \quad (1)$$

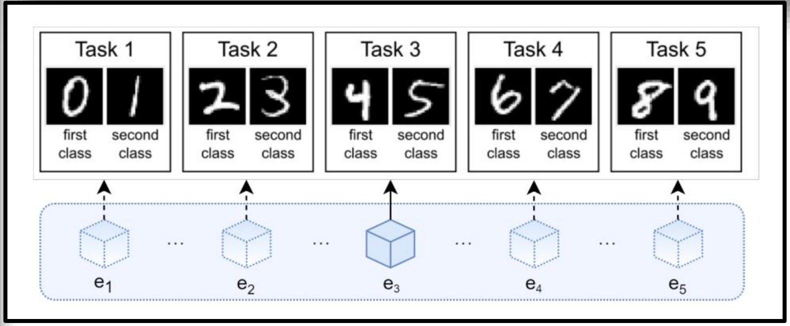
where f_i^{CL} is the model learned after training on experience e_i, $\mathcal{M}_i$ a buffer of past knowledge, such as previous samples or activations, stored from the previous experiences and usually of fixed size. The term t_i is a task label which may be used to identify the correct data distribution.

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

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

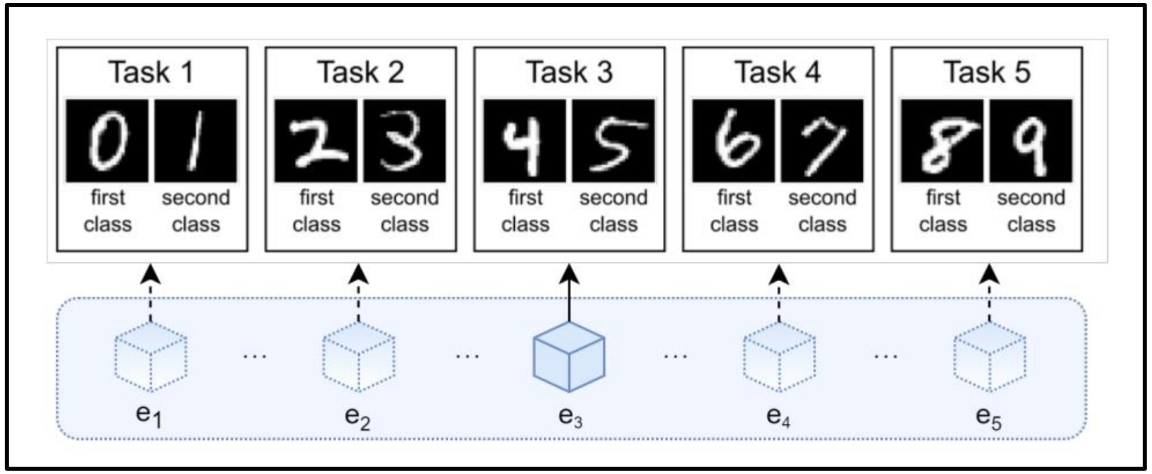
$$\mathcal{L}_{exp}(f_n^{CL}, \mathcal{D}^i_{test}) = \sum_{j=1}^{|\mathcal{D}^i_{test}|} \mathcal{L}(f_n^{CL}(\boldsymbol{x}_j^i), y_j^i), \quad (3)$$

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.



Scenarios Nomeclature: what we need

- **How much data** do we have in each experience?
- Do we know the **type of shifts**?
- Do we know **when** the shifts happen?
- Do we have **task labels** at training/inference time?



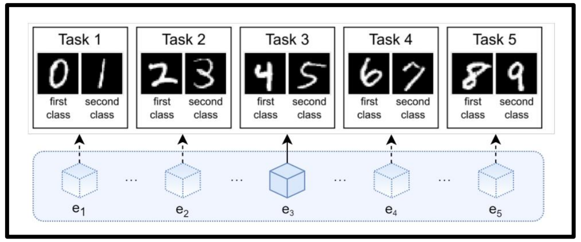
Real vs Virtual Shift

- **Real shift**:
  - We have seen it in OML
  - The world changes, either abruptly (covid lockdown) or continuously (weather, financial markets). $p(x,\ y)$ is changed.
  - In some application you care only about the present and future and can forget the past
- **Virtual Shift**
  - Common CL assumption
  - The «world» is fixed. Shifts are «virtual» and due to sample selection bias
  - The data changes because the prior $p(x)$ is changing. $p(y|x)$ is fixed.
  - You don't want to forget anything because you may encounter the old data again in the future

Real vs Virtual Shift

Mnist Dataset

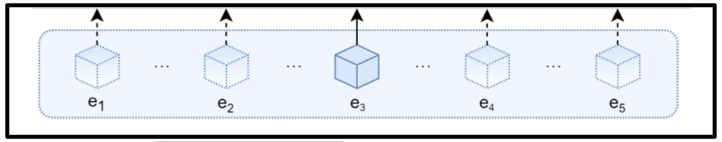
Non iid experience split

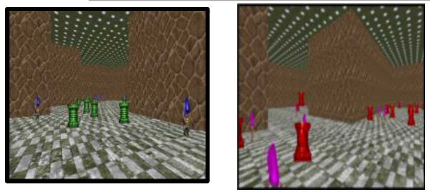


Change of environment

Column: **-1**
Lantern: **+1**

INVERTED
Column: **+1**
Lantern: **-1**





Non-stationarity Assumptions:

- **Real Shift**: the world is changing (more studied in OML).
- **Virtual Shift**: the world is fixed but there is a sample selection bias.

We will often assume virtual shifts.

REMINDER: Dataset Shift Nomenclature

Dataset Shift: $p_{tra}(x,y) \neq p_{tst}(x,y)$ Informally: any change in the distribution is a shift

Covariate shift: happen in X→Y problems when

- $p_{tra}(y|x) = p_{tst}(y|x)$ and $p_{tra}(x) \neq p_{tst}(x)$
- informally: the input distribution changes, the input->output relationship does not

Prior probability shift: happen in Y→X problems when

- $p_{tra}(x|y) = p_{tst}(x|y)$ and $p_{tra}(y) \neq p_{tst}(y)$
- Informally: output->input relationship is the same but the probability of each class is changed

Concept shift:

- $p_{tra}(y|x) \neq p_{tst}(y|x)$ and $p_{tra}(x) = p_{tst}(x)$ in X→Y problems.
- $p_{tra}(x|y) \neq p_{tst}(x|y)$ and $p_{tra}(y) = p_{tst}(y)$ in Y→X problems.
- Informally: the «concept» (i.e. the class)

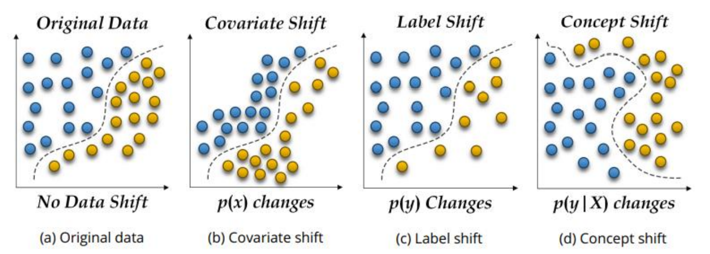


Figure 1: Dataset shift illustration (Similar to Fig. 1 in [4, 5].) Note that the dotted line is the decision boundary between the two classes; i.e., the blue and yellow data points. Here, x represents the input data and y represents the output we aim to predict.

Image: Dataset Shift in Machine Learning, Joaquin Quiñonero-Candela et al, MIT Press Book, 2008.

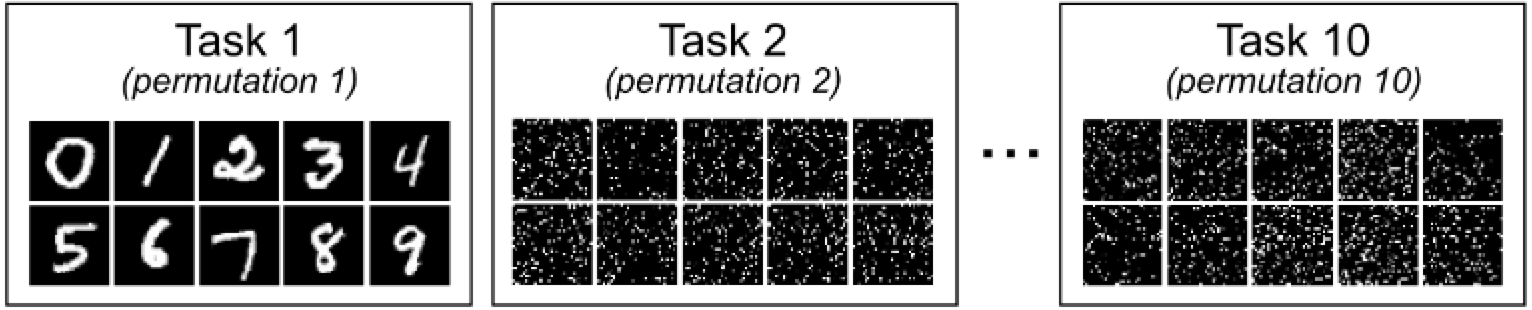
Common Assumptions

- **Shift is only virtual:** we do not want to forget, we need to accumulate knowledge.
- **No labeling errors/conflicting information:** targets are always correct (but possibly noisy).
- **Unbounded time:** No hard latency requirements. We may have computational constraints.
- **Data in each experience can be freely processed:** you can shuffle them, process them multiple times, etc. like you would do during offline training.

Common Types of Shifts

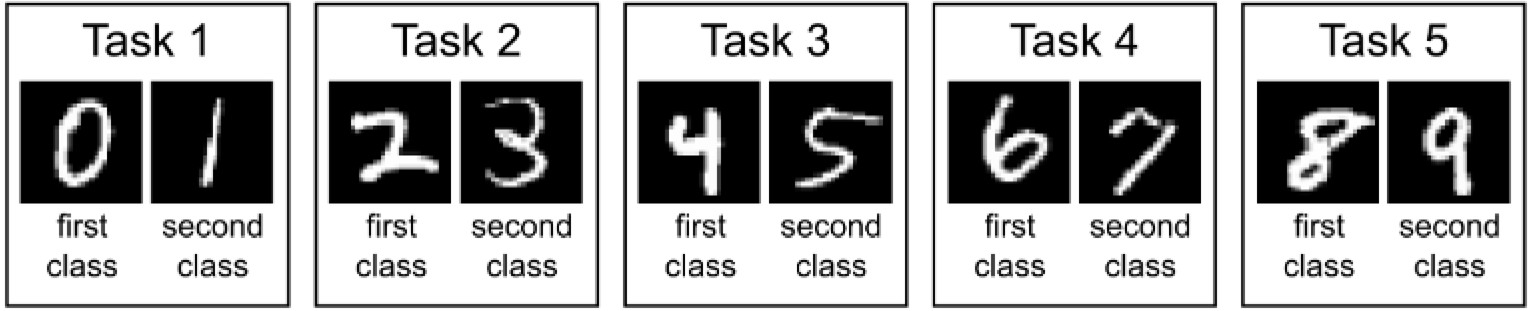
- **New Instances**: each experience provides new instances for old classes. Old instances are never seen again (in the training stream).

Permuted MNIST



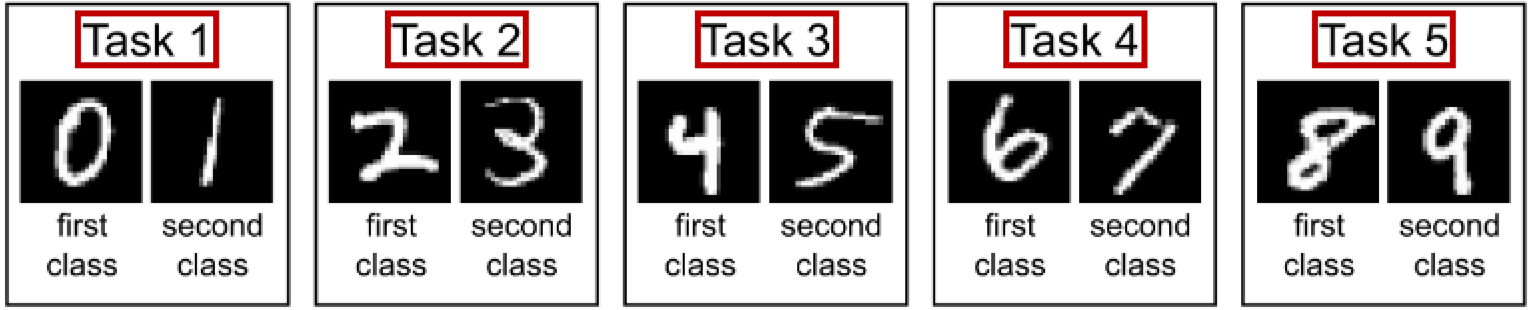
- **New Classes**: each experience provides new classes. Old classes are never revisited (in the training stream).

Split MNIST



Presence of Task Labels

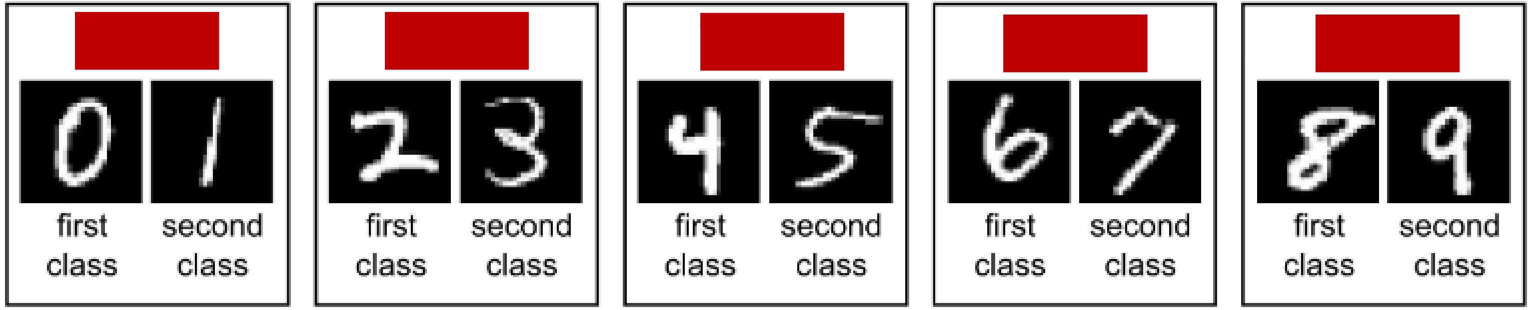
- **Task-aware**: task labels are available during training and inference



NOTE: some references use the term «task» to denote experiences even in task-agnostic scenarios

- **Task-agnostic**: task labels are not available

Task labels can change the output space (single vs multi head)



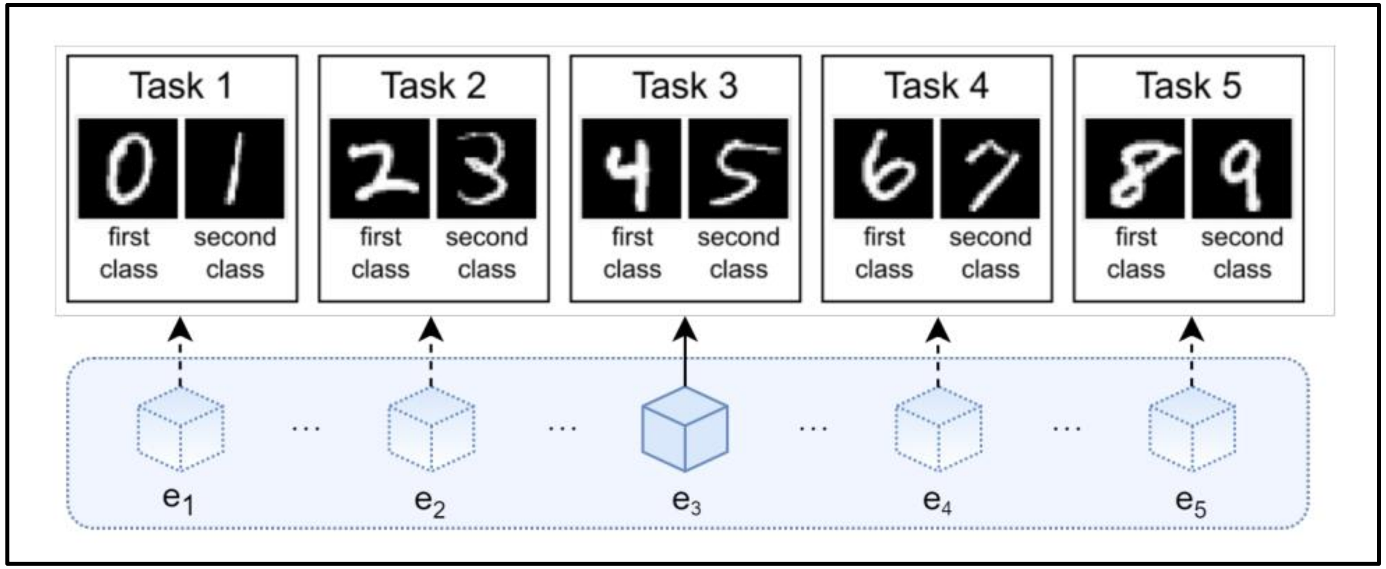
Tasks in Continual Learning

- Task labels simplify the problem
  - We can use multi-task models that take task labels as an explicit argument
    - Modularity also helps to prevent interference (it may limit forward transfer)
  - Output space is smaller:
    - 100 classes divided in 10 tasks -> 10-way classification
    - 100 classes in a single task -> 100-way classification
- Notice: the term task in CL is a bit overloaded
  - Sometimes, experiences are called tasks even when there are no explicit labels or other mechanisms to disambiguate different tasks
  - Often, tasks are actually domains (same problem, different $p(x)$)

Batch vs Online/Streaming CL

How much data for each experience?

- **Online CL (OCL)** / **Streaming CL**: Single example/small minibatch
- **Batch CL**: Large batch, no constraint on the size of the experience



Knowledge of Distribution Shifts

- **Task-free**: the model doesn't know when the shift happens (as in OML)
  - Notice: we don't have task labels AND we don't know WHEN the shift happens
- No common term for the «shift-aware» version
- In a batch scenario the typical assumption is that each experience is the result of a distribution shift
- In OCL knowledge of task boundaries is more useful (because the stream is much longer) but all the methods assume that they don't have access to it (more realistic)

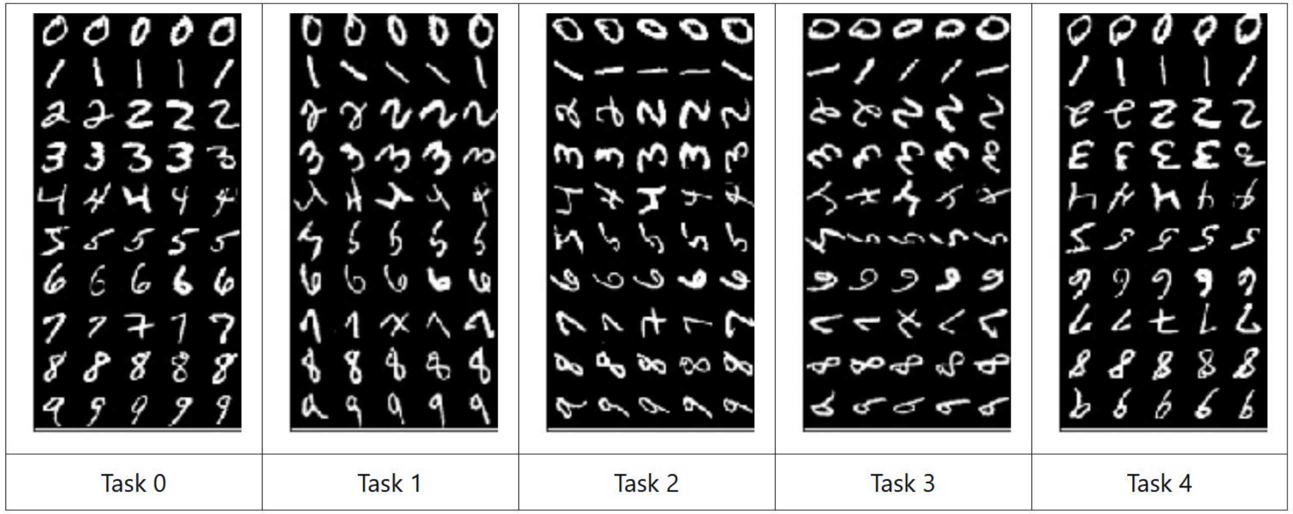
Sharp vs Blurry Shifts

- **Sharp Shifts**: drift happen abruptly
- **Blurry/Gradual Shifts**: drift happen slowly

Most CL methods deal with sharp drifts

Example of gradual shift: Rotated MNIST

RotatedMNIST:



Remember the assumption about «no conflicting information»? We may want to remove 6 or 9 here

img source: https://github.com/Continvvm/continuum

Nomenclature for Common Scenarios

- **Availability of Task/Distribution Labels**: during training and/or testing
- **Task/Shift Boundaries**: during training and/or testing
- **Experience Content**: examples of [same|new] classes
- **Output Space**: [Shared/Separate]
- NOT an exhaustive classification

Name	Task Labels	Boundaries	Classes	Output
Class-Incremental	No	Yes	New	Shared
Task-Incremental	Yes	Yes	New	Separate
Domain-Incremental	No	Yes	Same	Shared
(Online) Task-Free	No	No	Any	Shared

Task-Labels/Shift-Type Categorization

Alternative 2D categorization:
- Presence of task labels
- Type of shift (class/instance)

- *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)*: $\exists\ i, j, k :\ t_i = t_j\ and\ t_j \neq t_k$.

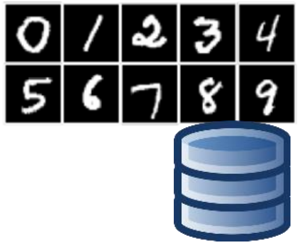
- **NEW:** repetitions of concepts
- **Limitation**: each experience has a single task label

	New Instances (NI)	New Classes (NC)	New Instance and Classes (NIC)
Multi-Task (MT)	-	Task Incremental	-
Single-Incremental Task (SIT)	Domain-incremental	Class-incremental	Data-incremental
Multiple-Incremental-Task (MIT)	-	-	-

Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. Lesort et al, Information Fusion, 2020.

Dataset, Scenarios, Benchmarks

MNIST Dataset



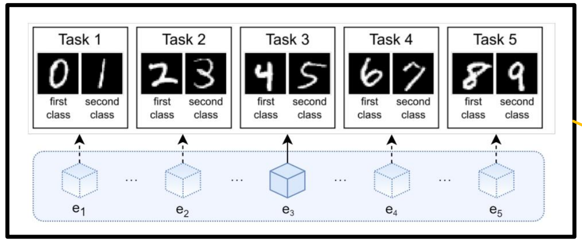
Class-Incremental Learning Scenario

Settings:

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

+

Split MNIST Benchmark



Benchmark Instances

Exact specific sequences and composition of e. For example:

S1 = {class 0 & 1}, {class 2 & 3}
S2 = {class 5 & 7}, {class 0 & 8}

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.

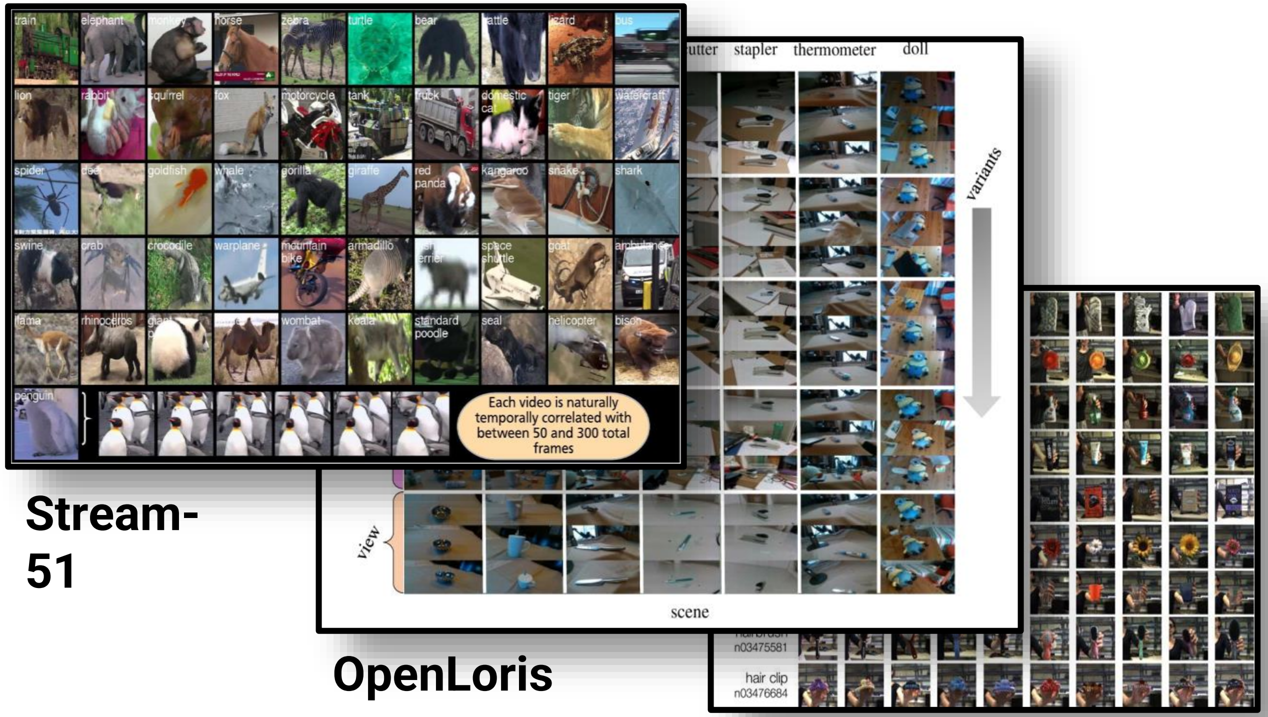
Some examples

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]

Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. Lesort et al, Information Fusion, 2020.

Natural Video Benchmarks



Stream-51

OpenLoris

iCub-Transformation

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

Example: CORe50

- Continuous Object Recognition
  - 50 classes
  - Short videos of object manipulation with different background
  - Temporal coherence from videos
- Many scenarios: batch, online, with repetitions.

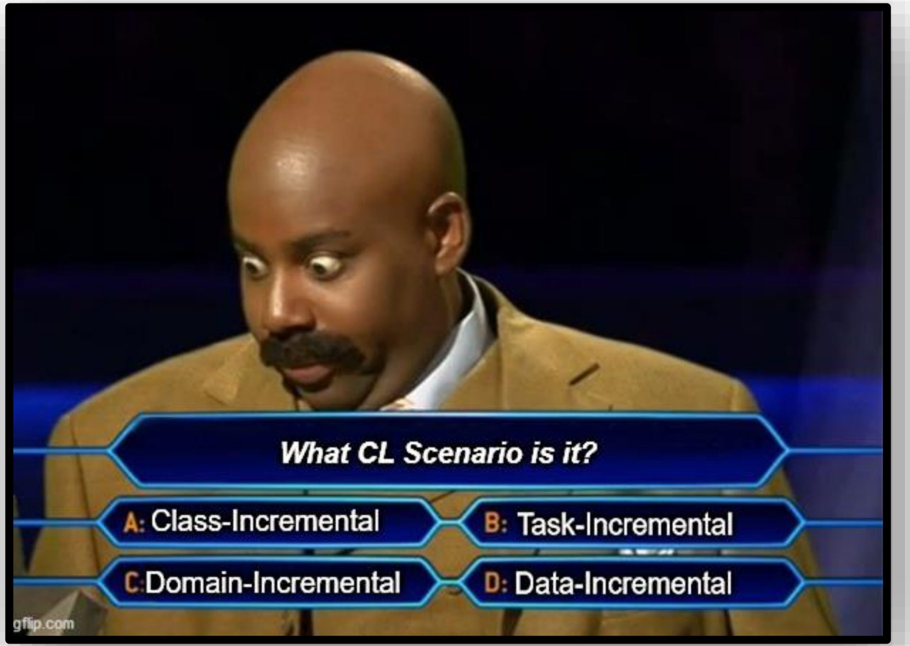
Lomonaco V. and Maltoni D. CORe50: a New Dataset and Benchmark for Continuous Object Recognition. CoRL2017.

CORe50

# 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

Task-Agnostic

New Instances & Classes



Task-Free

Single-Incremental-Task

Multi-Task

CL – Evaluation

Average Stream Accuracy

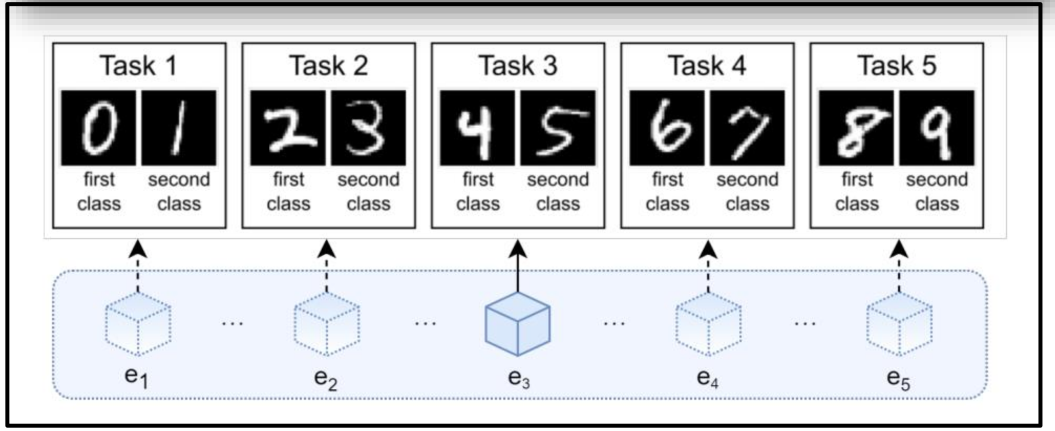
- The model is evaluated on the average accuracy on the entire (test) stream
- It must remember how the classify data from old experiences

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.



Offline Model Selection

- We assume to have access to three parallel stream
  - Train/Validation/Test streams
  - At time t, they all have data from the same distribution
- Offline Model Selection
  - Train several models on the train stream
  - Select best on the entire validation stream
- Final evaluation on the test stream
- Simple to implement but unrealistic. Assume to have access to the entire stream at the end of training for model selection purposes

Early Model Selection

- Use the first k experiences for model selection
  - Train sequentially on the training splits (until time=k)
  - Evaluate on the validation splits
  - Select the best model on the first k expereriences of the validation stream
- Continue training the best model on the rest of the training stream
- Model selection is still offline, but only for the first part of the stream

Continual Hyperparameter Selection

- Can we do the model selection without access to the old data?
- We have two objectives:
  - Maximize plasticity (learning new experiences)
    - Accuracy on current data
    - Is to estimate given the current validation experience
  - Minimize forgetting (of older experiences)
    - Accuracy on past experiences (for the current model)
    - We don’t have the data to evaluate this objective
- Let’s assume that we have only two hyperparameters:
  - One controls plasticity
  - The other controls stability (forgetting)

Continual Hyperparameter Selection

Example hyperparameters:

- Plasticity: learning rate
- Stability: regularization strength

PSEUDOCODE:

- For each experience:
  - STEP 1: Find optimal plasticity hyperparameters
    - This step will find the max accuracy you can get at the expense of stability
    - fix them
  - STEP 2: Find stability hyperparameters
    - Start with maximal stability
    - Decrease stability hyperparameters until you have a good enough accuracy
    - This a stability-plasticity tradeoff. If you stop too soon you have low plasticity. If you stop too late, you have too much forgetting.

Continual Hyperparameter Selection

ALGORITHM:

- for each experience:
    - finetune on new data, coarse grid search on lr, get acc A
    - train on new data with CL method (lr from prev step), get acc A*
    - while performance on new data is too low (A - A* too big)
        - Decrease stability hparams
        - train on new data, get acc A*

INTUITIVELY: after finding optimal plasticity hparams, decrease forgetting hparams until the performance is close enough to the optimal plasticity (tradeoff)

Algorithm 1. Continual Hyperparameter Selection Framework

input $\mathcal{H}$ hyperparameter set, $\alpha \in [0,1]$ decaying factor, $p \in [0,1]$ accuracy drop margin, D^{t+1} new task data, Ψ coarse learning rate grid
require θ^t previous task model parameters
require CLM continual learning method

`//Maximal Plasticity Search`

$A^* = 0$
for $\eta \in \Psi$ **do**
  $A \leftarrow \text{Finetune}(D^{t+1}, \eta; \theta^t)$ ▷ Finetuning accuracy
  if $A > A^*$ **then**
    $A^*, \eta^* \leftarrow A, \eta$ ▷ Update best values

`//Stability Decay`

do
  $A \leftarrow CLM(D^{t+1}, \eta^*; \theta^t)$
  if $A < (1-p)A^*$ **then**
    $\mathcal{H} \leftarrow \alpha \cdot \mathcal{H}$ ▷ Hyperparameter decay
while $A < (1-p)A^*$

CL vs OML

Deep CL

- (possibly) Large experiences
- Virtual drift
- Domains: Vision, NLP, speech
- Evaluation: average accuracy on the full stream

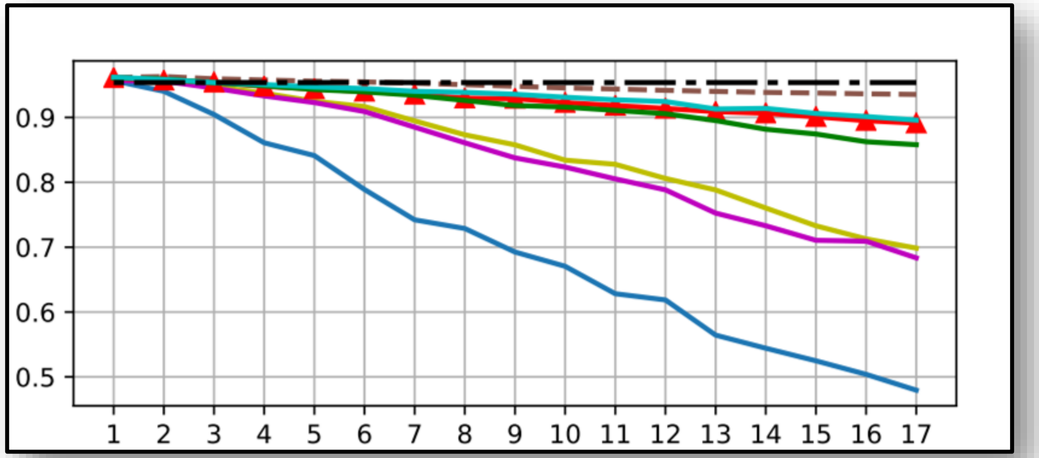
Online ML

- One sample at a time
- Real drift
- Domains: time series data
- Evaluation: prequential accuracy

CL – Metrics

What to monitor?

- Performance on current/past/future experience
- Resource consumption: Memory, CPU, Disk usage
- Model size growth
- Execution time and latency
- 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

Let N be the stream length, T the current timestep, $R_{t,i}$ be the accuracy on the experience i at time t

Time – which model to use:

- Last model: $R_{T,T}$
- Averaged over time: $\frac{1}{T+1}\sum_{t=0}^{T}\sum_{i=0}^{t} R_{t,i}$

Data – which data to use:

- Current experience: $R_{t,t}$
- Data seen up to now: $\frac{1}{T+1}\sum_{i=0}^{T} R_{T,i}$
- Full stream: $\frac{1}{N}\sum_{i=0}^{N-1} R_{T,i}$

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	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$

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

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

Zero-shot Forward Transfer

Q: Is continual learning improving the performance on future experiences?

- FWT Metric compares
  - Accuracy on future experience i+k after training on experience i
- against
  - $\bar{b}_i$ Accuracy on experience i of a model trained with a random initialization
  - Averaged over i=2,...,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.

Evaluating Representation Learning with Linear Probing

- FWT assumes that the model can predict future experiences
  - Most models can't predict unseen classes
  - Only makes sense for new instances, not new classes
- Alternative solution: Evaluate whether the latent representation helps learning unseen tasks
- **Linear Probing**:
  - Learn a linear classifier on top of the learned representation
  - Compare against random feature extractor and previous models
- Measures whether the learned features transfer to the new data

Backward Transfer

Q: Is continual learning improving the performance on **OLD** 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	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$

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

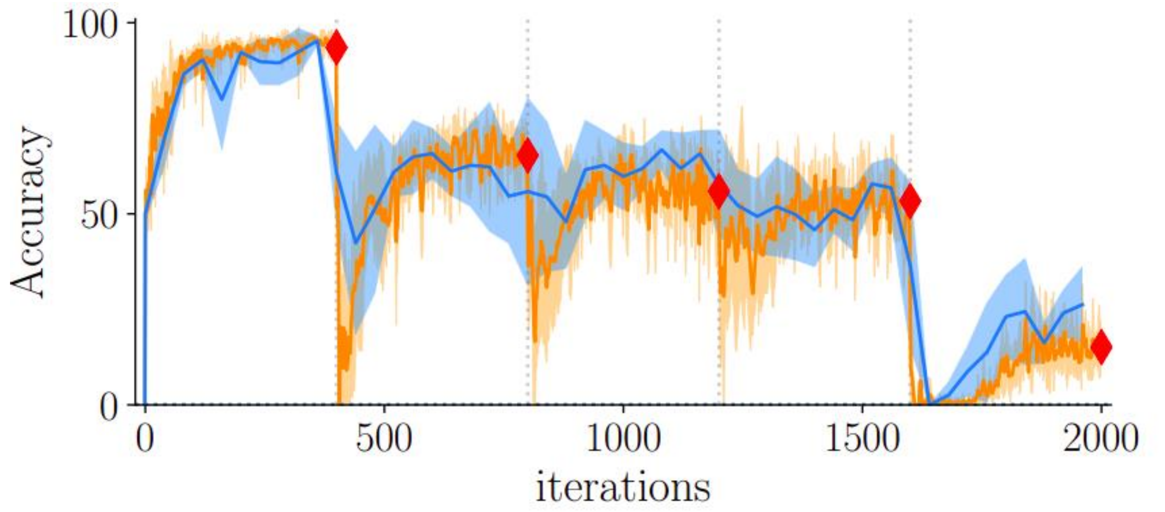
Gradient Episodic Memory for Continual Learning. Lopez-Paz & Ranzato, NeurIPS 2017.

Online CL – Desiderata

Evalution in Online CL is more difficult because the stream is longer and task boundaries are unknown.

- **Knowledge Accumulation**: the model should improve over time
  - At any point in time
  - High average accuracy but also fast adaptation
- **Continual Stability**: the model should not forget previous knowledge
  - At any point in time
  - We often assume virtual drifts when measuring stability
- **Representation Quality**: the latent representations should improve over time
  - A weaker form of knowledge accumulation/stability
  - Can be evaluated on out-of-distribution data or self-supervised models

Red diamonds = task boundaries

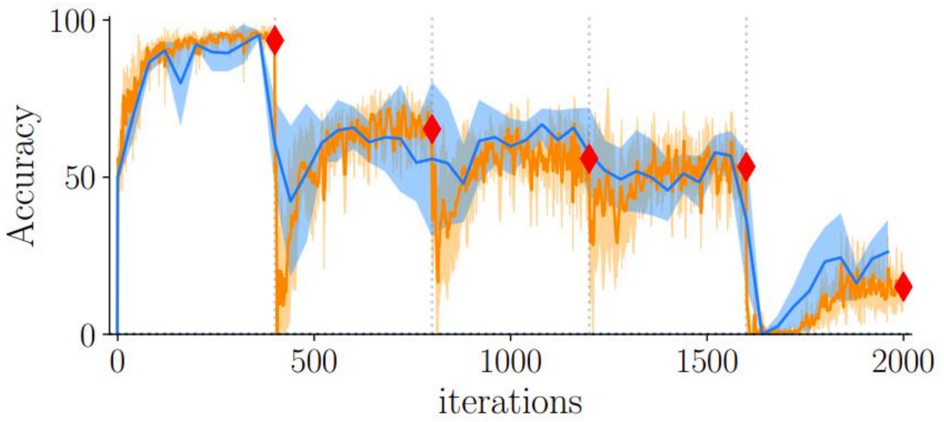


A. Soutif et al. "A Comprehensive Empirical Evaluation on Online Continual Learning" 2023

Knowledge Accumulation

Red diamonds = task boundaries

- **Average Anytime Accuracy**: accuracy along the entire curve.
- Do not confuse with
  - Avg accuracy at the end of training (final diamond)
  - Avg at task boundaries (avg of diamonds)



Notation:
- f_i model at time i
- E_i experience i
- $A(E_i, f_i)$ accuracy of model f_i for experience E_i

$$\mathrm{AAA}_t = \boxed{\frac{1}{t}} \sum_{j=1}^{t} \boxed{\frac{1}{k} \sum_{i=1}^{k} \mathbf{A}(E_i, f_j)}$$

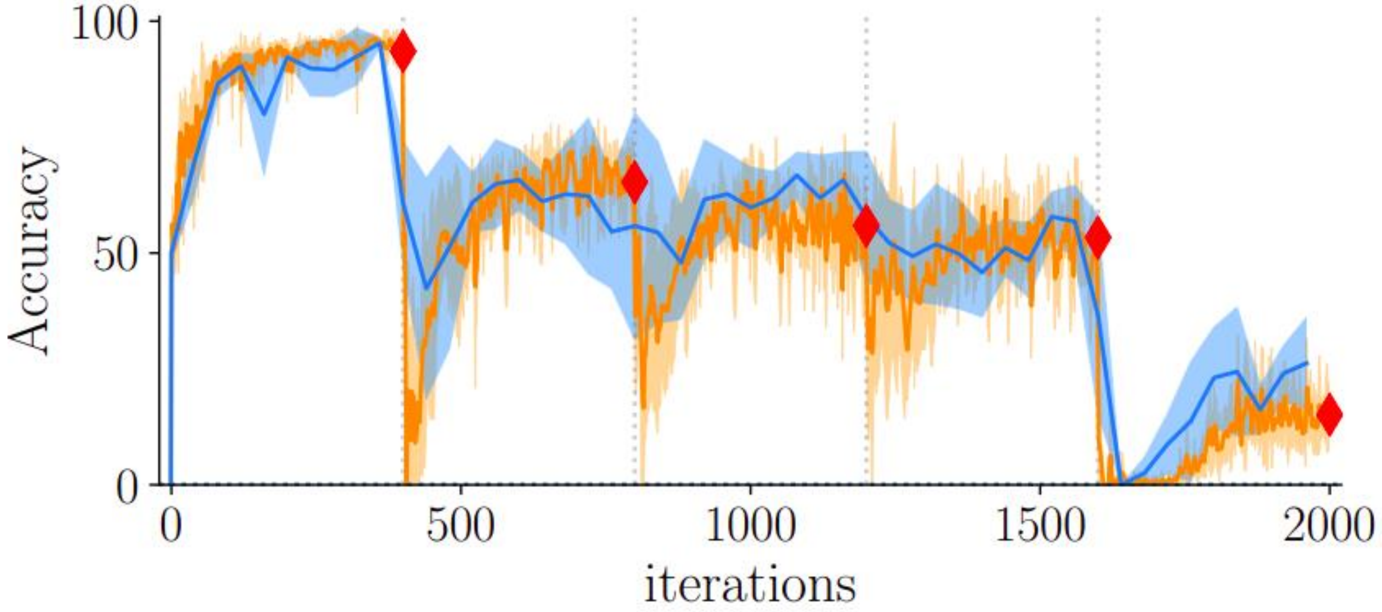
Average along the training curve

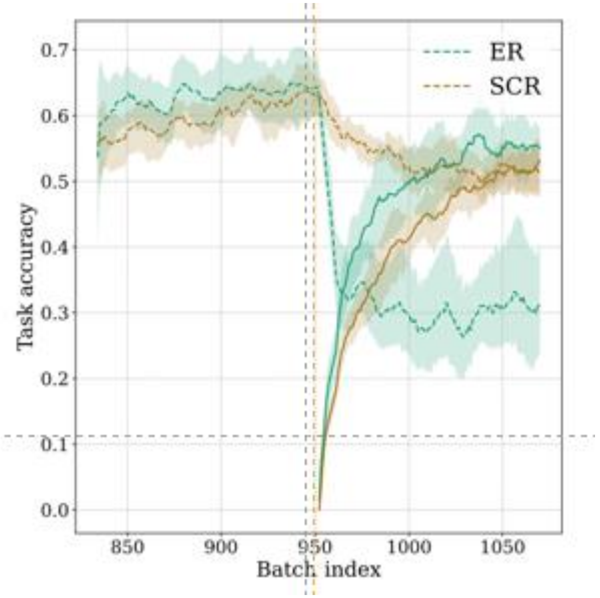
Average accuracy of data seen up to now

Lucas Caccia et. al, *New Insights on Reducing Abrupt Representation Change in Online Continual Learning*, ICLR 2022

Continual Stability

- Observe the behavior of the accuracy during training (curve from one diamond to the next)
- CL methods forget and re-learn old experiences during training
- This phenomenon is masked with the typical metrics measured only at boundaries (red diamonds)





[1] Mathias Delange et. al, Continual Evaluation for Lifelong Learning: Identifying the stability gap, ICLR 2023
[2] Lucas Caccia et. al, New Insights on Reducing Abrupt Representation Change in Online Continual Learning, ICLR 2022
[3] A. Soutif et al. "A Comprehensive Empirical Evaluation on Online Continual Learning" 2023

Memory Usage

- **Model Size (MS)**: How much space does your model occupy? (MB, # of params, etc.)
- Scalability over time: What is the increment in space required for each new experience?
- **Samples Storage Size (SSS)**: 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

Q: What is the computational overhead during training and inference?

- #MAC – Multiply and Accumulate
- Running Time, CPU/GPU time
- scalability over time

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

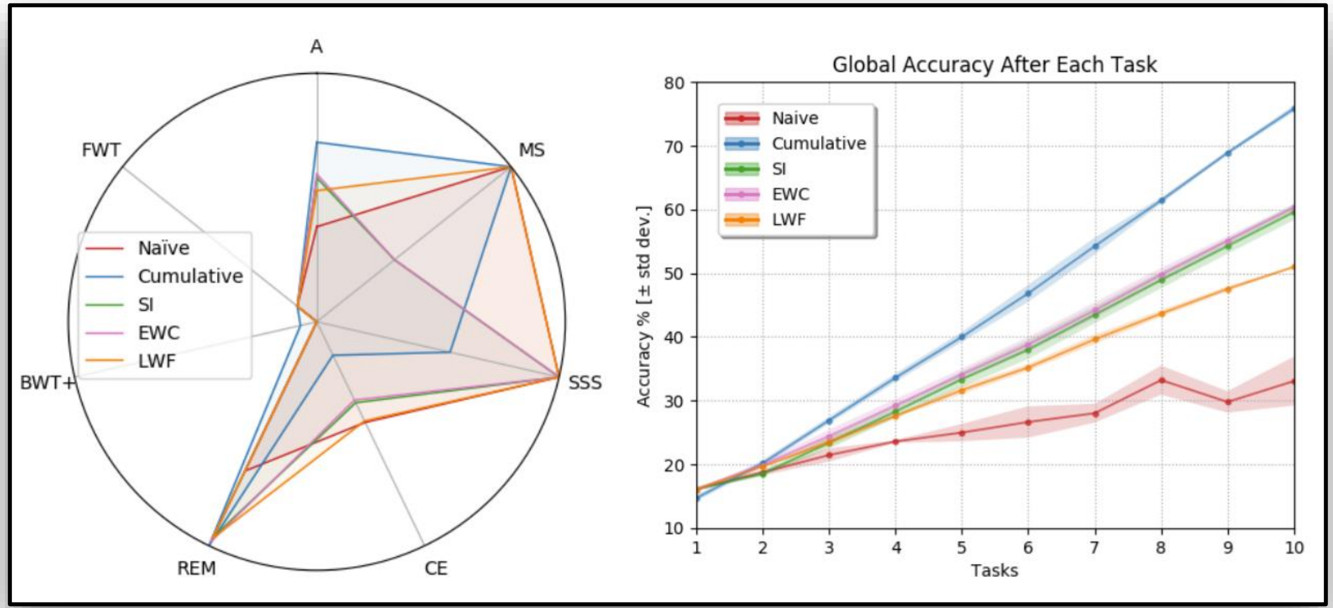
NOTE: evaluating GPU performance is tricky because you have to consider CPU<->GPU communication, synchronization costs, parallelization. #MAC can be misleading.

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

In general, multiple objectives!

Each application is different and multiple objectives may interfere with each other:

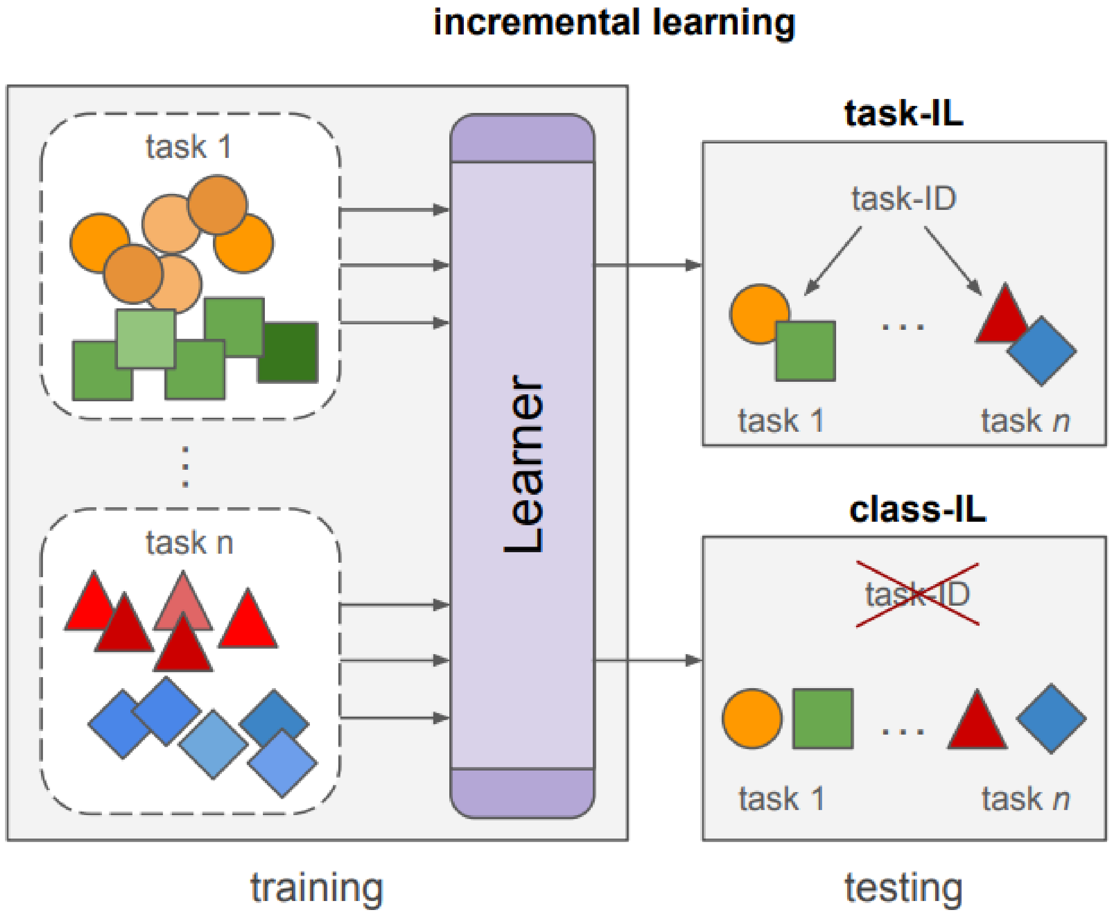
- Computational constraints
- Privacy Constraints
- Accuracy and Forgetting



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

Forgetting in CL Scenarios

Task-Incremental vs Class-Incremental



Masana, Marc, et al. "Class-incremental learning: survey and performance evaluation on image classification." IEEE TPAMI 2022

Task-Incremental vs Class-Incremental

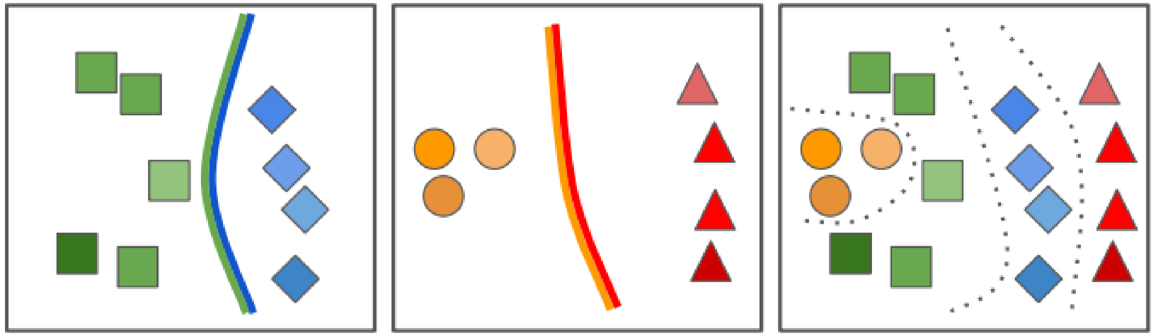


Fig. 2: A network trained continually to discriminate between task 1 (left) and task 2 (middle) is unlikely to have learned features to discriminate between the four classes (right). We call this problem *inter-task confusion*.

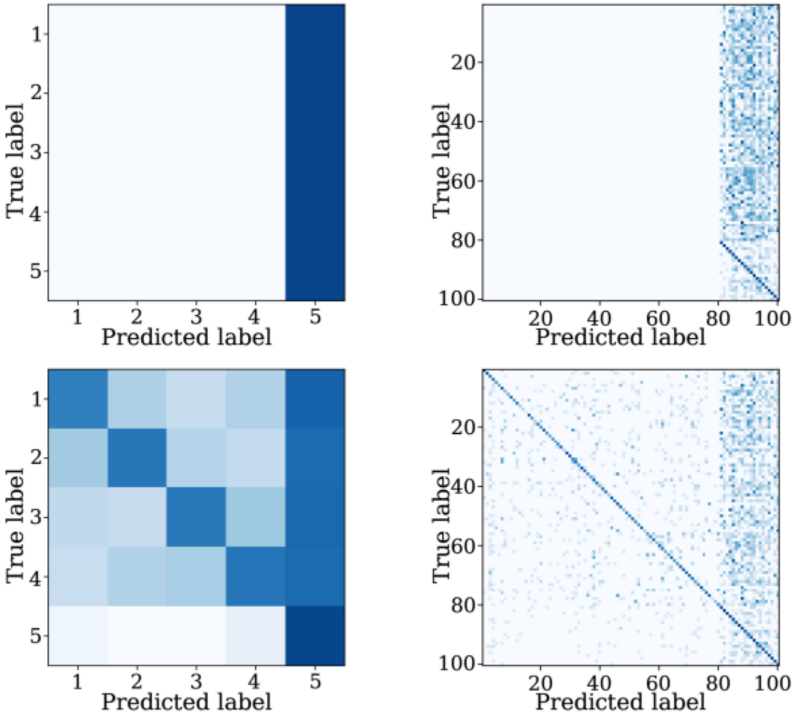


Fig. 3: Examples of task and class confusion matrices for Finetuning (top row) and Finetuning with 2,000 exemplars (bottom row) on CIFAR-100. Note the large bias towards the classes of the last task for Finetuning. By exploiting exemplars, the resulting classifier is clearly less biased.

Classifier Bias in CIL

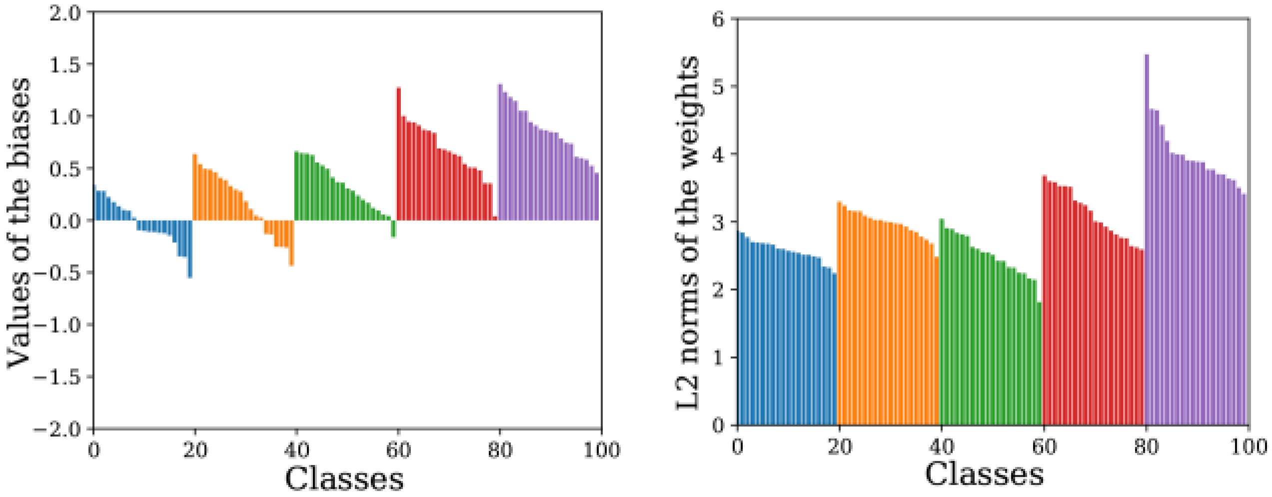


Fig. 4: Bias and weight analysis for iCaRL with 2,000 exemplars on CIFAR-100. We show the ordered biases and norm of the last classification layer of the network for different tasks. Note how the bias and the norm of the weights are higher for the last tasks. This results in a *task-recency bias*.

Replay does not fix the task-recency bias

Masana, Marc, et al. "Class-incremental learning: survey and performance evaluation on image classification." IEEE TPAMI 2022

Self-Supervised Models

- SSL methods are more robust to continual training
- We evaluate the representation (linear probing)

- More realistic because we don't need labels during training
- We still need labels to finetune the classifier (linear probing), otherwise we can't use the model
- SSL methods are slow to converge and require large amounts of data

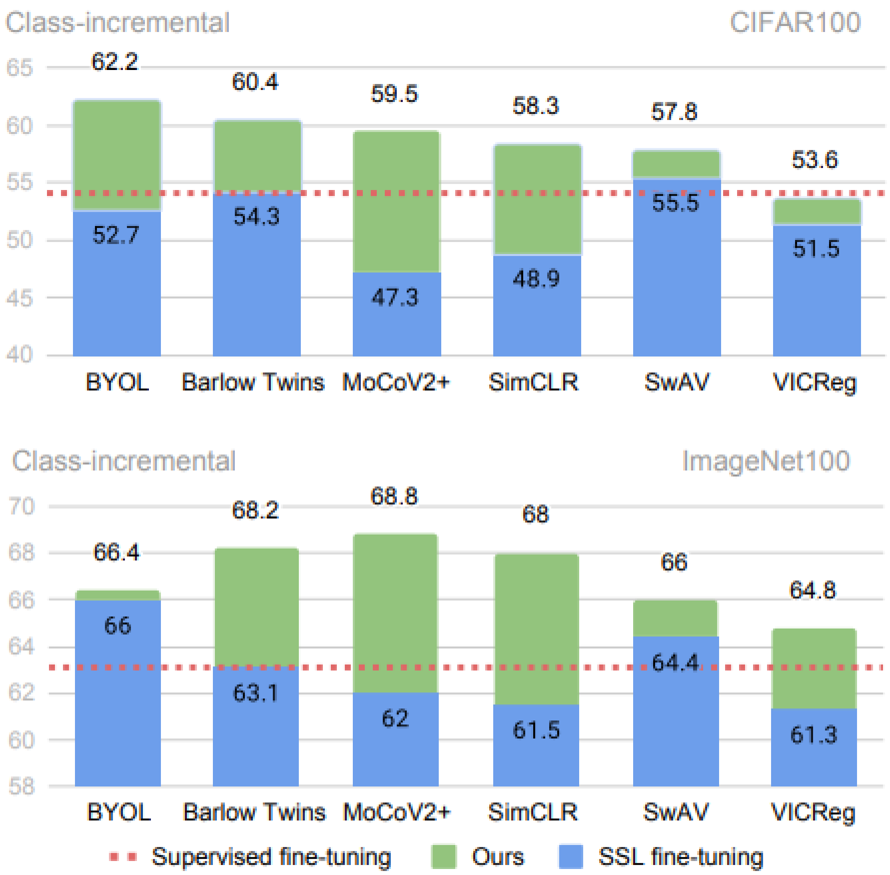


Figure 1. Linear evaluation accuracy of representations learned with different self-supervised methods on class-incremental CIFAR100 and ImageNet100. In blue the accuracy of SSL fine-tuning, in green the improvement brought by CaSSLe. The red dashed line is the accuracy attained by supervised fine-tuning.

Fini, Enrico, et al. "Self-supervised models are continual learners." CVPR. 2022.

Continual Pretraining

Continual Pretraining

- SSL pre-training, incrementally over time
- Finetuning on the downstream tasks using the last model

Again, SSL > supervised, both in vision and text

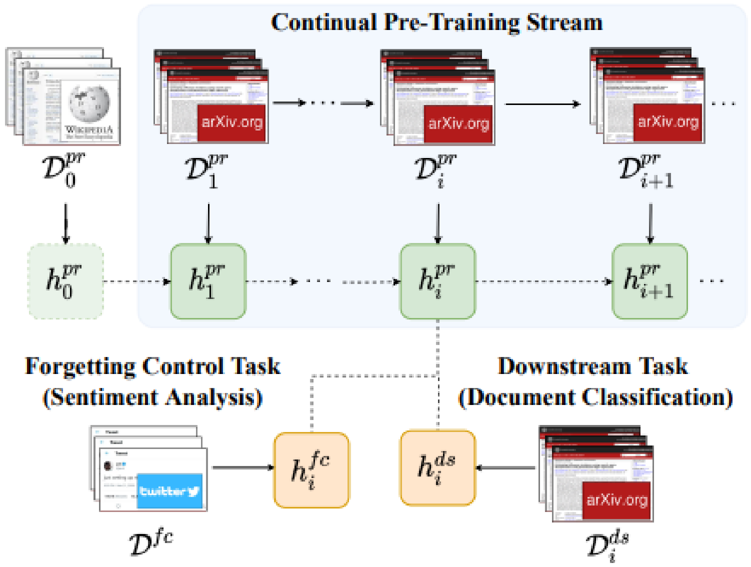


Figure 1: The Continual Pre-training scenario. During each stage (experience) i of continual pre-training (top), the model h_i^{pr} is pre-trained (center) on the dataset $\mathcal{D}_i^{pr}$ (e.g., *scientific abstracts*). Subsequently (bottom), the model is fine-tuned against one (or more) downstream task $\mathcal{D}_i^{ds}$ (e.g. *scientific abstracts* classification). Forgetting is measure by fine-tuning on $\mathcal{D}^{fc}$ (e.g. *sentiment analysis*). At each stage, only the current pre-trained and downstream datasets/models are available.

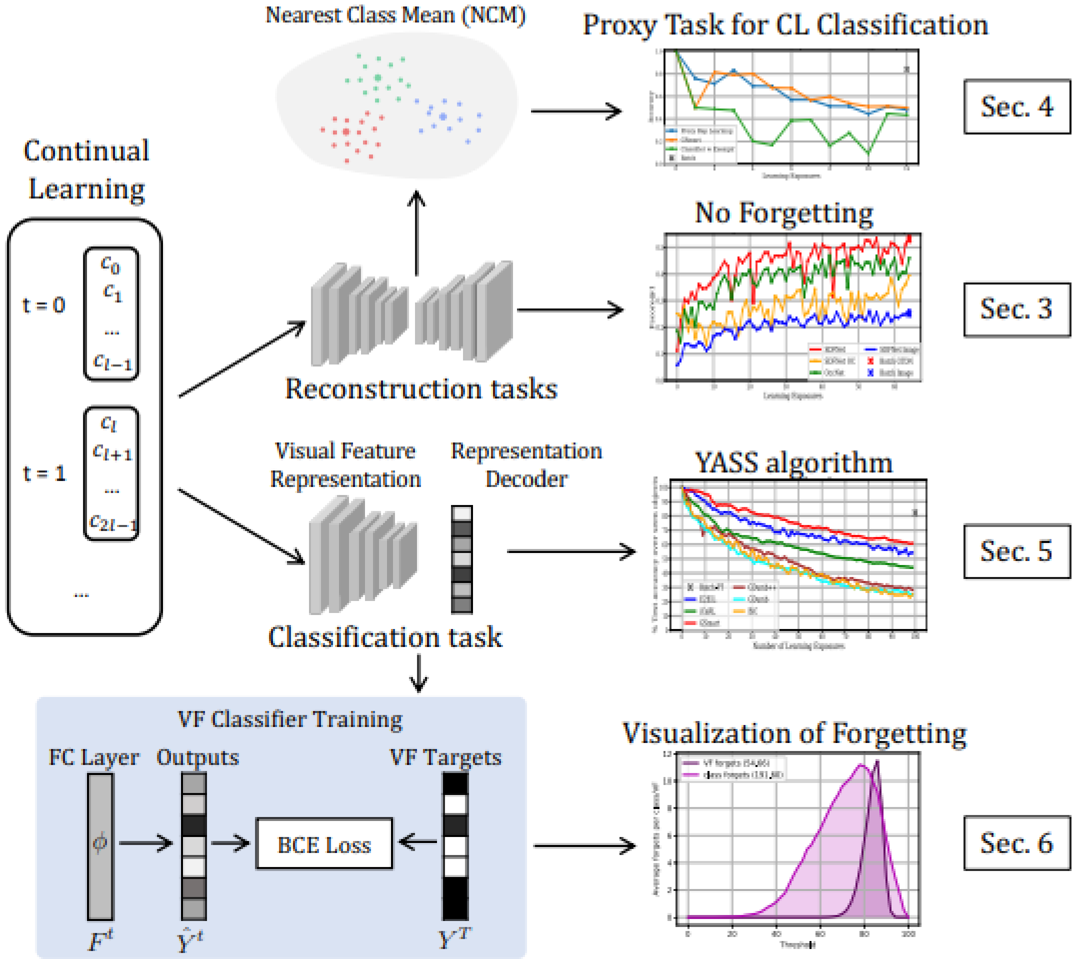
Cossu, Andrea, et al. "Continual pre-training mitigates forgetting in language and vision." arXiv preprint arXiv:2205.09357 (2022).

Forgetting in Different Scenario

Some tasks are much more robust to CL than others

- Incremental classification results in catastrophic forgetting
- SSL methods are more robust
- Tasks such as reconstruction are very robust

- Forgetting will also depend on the drifts (iid vs class vs domain vs gradual…)



Thai, Anh, et al. "Does continual learning= catastrophic forgetting." arXiv preprint arXiv:2101.07295 (2021).

Take-Home Messages

While DCL is similar to OCL, there are many differences:

- Domains: vision, NLP, …
- Drift: real vs virtual
- Evaluation: avg. accuracy vs prequential
  - Forgetting is a major concern which we don't have in the prequential setting
  - We can have forward transfer with deep learning models
- DCL methods are much more expensive than OCL. The main challenge is avoiding forgetting, not fast adaptation

Next Lecture

We start looking at the algorithms:

- Baselines for CL
- Replay Methods