Continual Learning

Collective Machine Intelligence

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

Next Lectures

- Intro to Continual Learning (today)
    - Motivations and problem definition
    - Benchmarks and methods
    - Open research questions
- Learning multiple task and multiple agents
- Continual and Multi-Task Learning in the LLM era

Motivations

- Understanding the problems of
  - Learning over time
  - Sharing knowledge between different ML agents
  - Generalizing to novel task
- Highlighting open research questions

Resources

- Video lectures phd course on CL: https://course.continualai.org/
- Notebooks master course: https://github.com/AntonioCarta/continual-learning-course

Avalanche

- CL library built on top of Pytorch
- Currently the most extensive collection of CL benchmarks and algorithms
- Used by the CL community for research, new benchmarks, challenges and courses

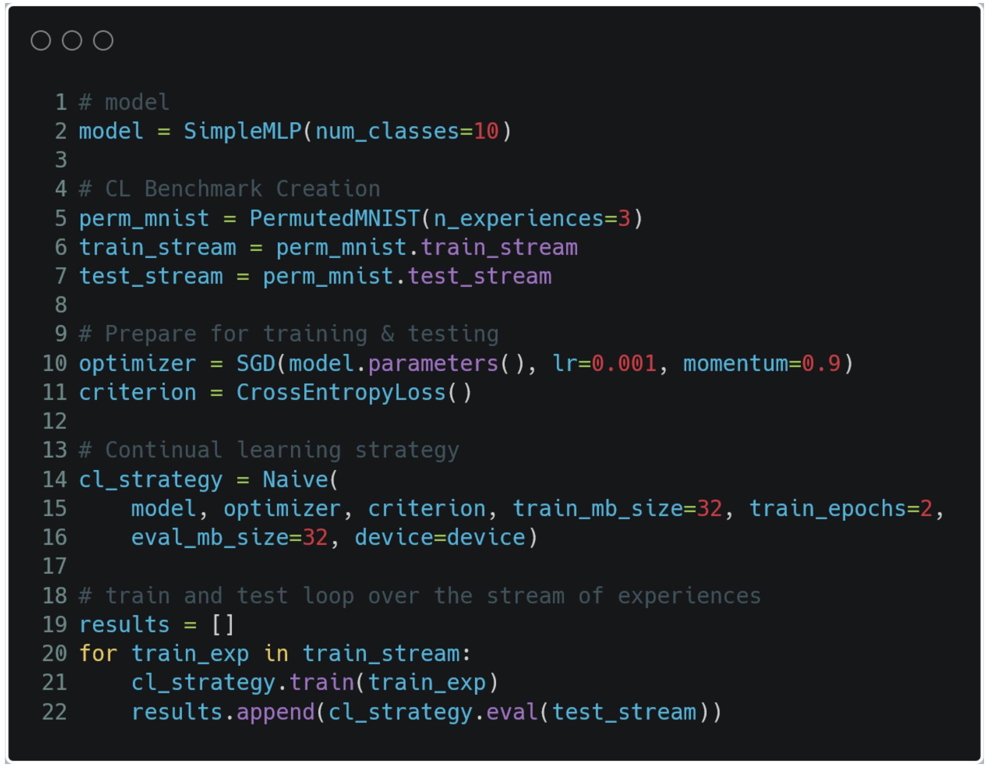
Website: avalanche.continualai.org/

CL-baselines:

https://github.com/continualAI/continual-learning-baselines/

Avalanche-demo:
https://github.com/AntonioCarta/avalanche-demo

Avalanche powered by ContinualAI



```python
# 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 over the stream of experiences
results = []
for train_exp in train_stream:
    cl_strategy.train(train_exp)
    results.append(cl_strategy.eval(test_stream))
```

Deep Continual Learning

Lifelong Learning Artificial Agents

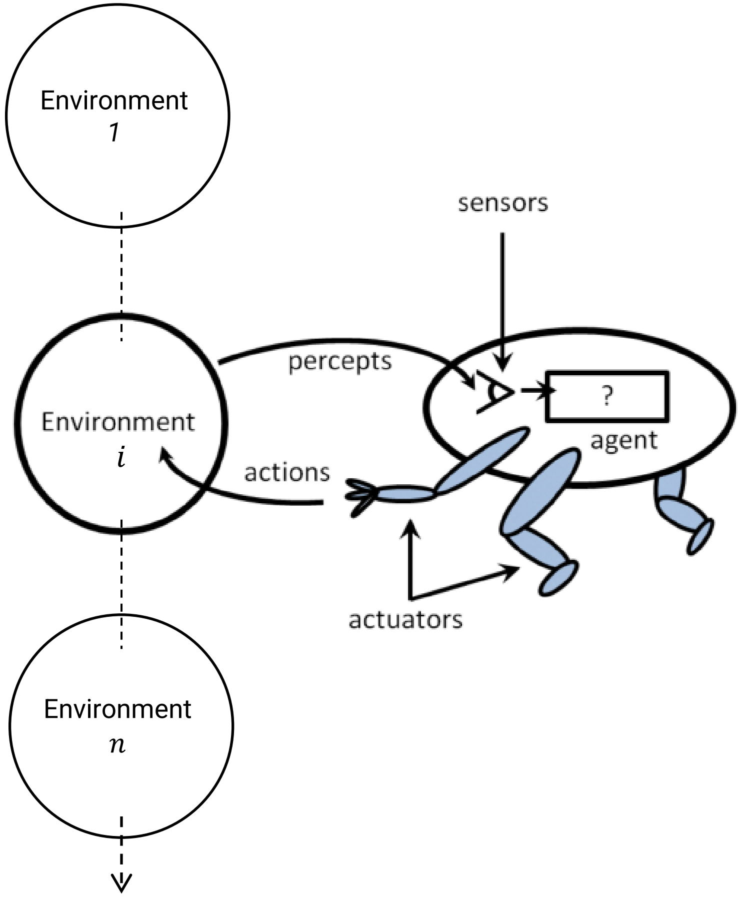
Our goals:

1. **Incremental Learning**: knowledge and skills accumulation and re-use. Learn new skills + don’t forget.

2. **Fast Adaptation**: adapt to ever-changing environments

3. **Generalization**: adapt to unseen environments and improve learning algorithm

AI Agent Architecture
(Russel & Norvig, 1995 - 2022)



A Long-Desired Objective

- Incremental learning with rule-based systems (**Diederich, 1987**)
- Forgetting in Neural Networks (**French, 1989**)
- Incremental learning with Kernel Machines (**Tat-Jun, 1999**)
- Continual Learning (**Ring, 1998**)
- Lifelong Learning (**Thrun, 1998**)
- Dataset Shift (**Quiñonero-Candela, 2008**)
- Never-Ending Learning (**Mitchell, 2009**)
- Concept Drift Adaptation (**Ditzler, 2015**)
- Deep Continual Learning (**Kirkpatrick, 2016**)
- Lifelong (Language) Learning (**Liu, 2018**)

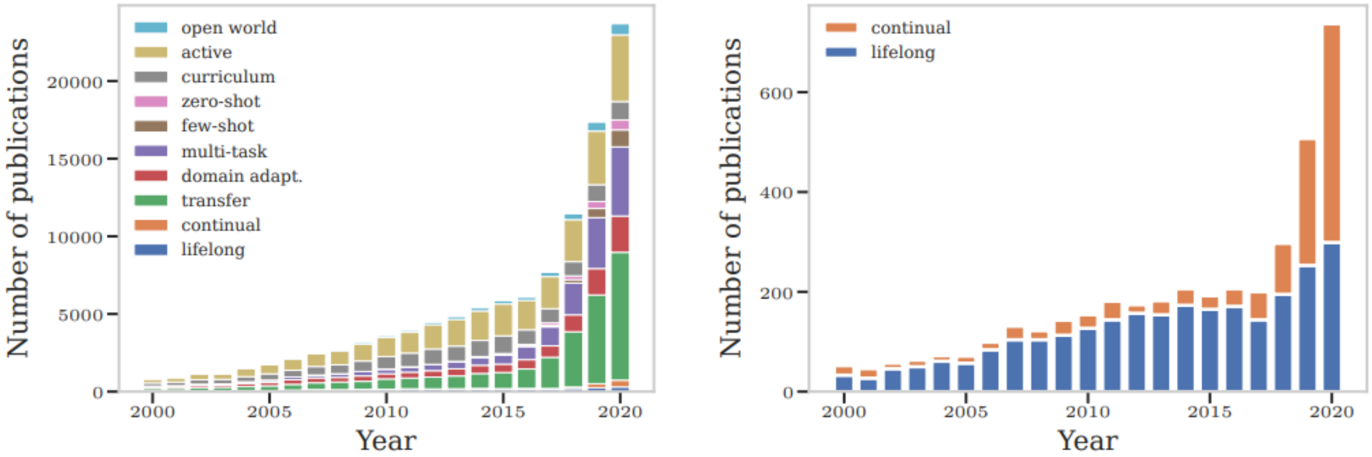


Figure 1: Per year machine learning publications. Left: cumulative amount of papers across keywords with continuous components that influence continual learning practice, see Section 2. Right: increasing use of "continual" machine learning, demonstrating a shift in use of terminology with respect to the preceding emphasis on the term "lifelong". Data queried using the Microsoft Academic Graph utilities (Sinha et al., 2015) based on keyword occurrence in the abstract.

CLEVA-Compass: A Continual Learning EValuation Assessment Compass to Promote Research Transparency and Comparability, Martin Mundt et al. preprint, 2021

CL and Neuroscience

“We are not looking for incremental improvements in state-of-the-art AI and neural networks, but rather paradigm-changing approaches to machine learning that will enable systems to continuously improve based on experience.”

— Hava Siegelmann, 2018

Practical Motivations

- Training is expensive
- Data is always changing
- We want to reuse previous models

Dealing with Non-Stationary Environments

“The world is changing and we must change with it" - Ragnar Lothbrok

What is Concept Drift (CD)?

What it is:

- A change in the real world
- Affects the input/output distribution
- Disrupt the model's predictions

What it's not:

- It's not noise
- It's not outliers

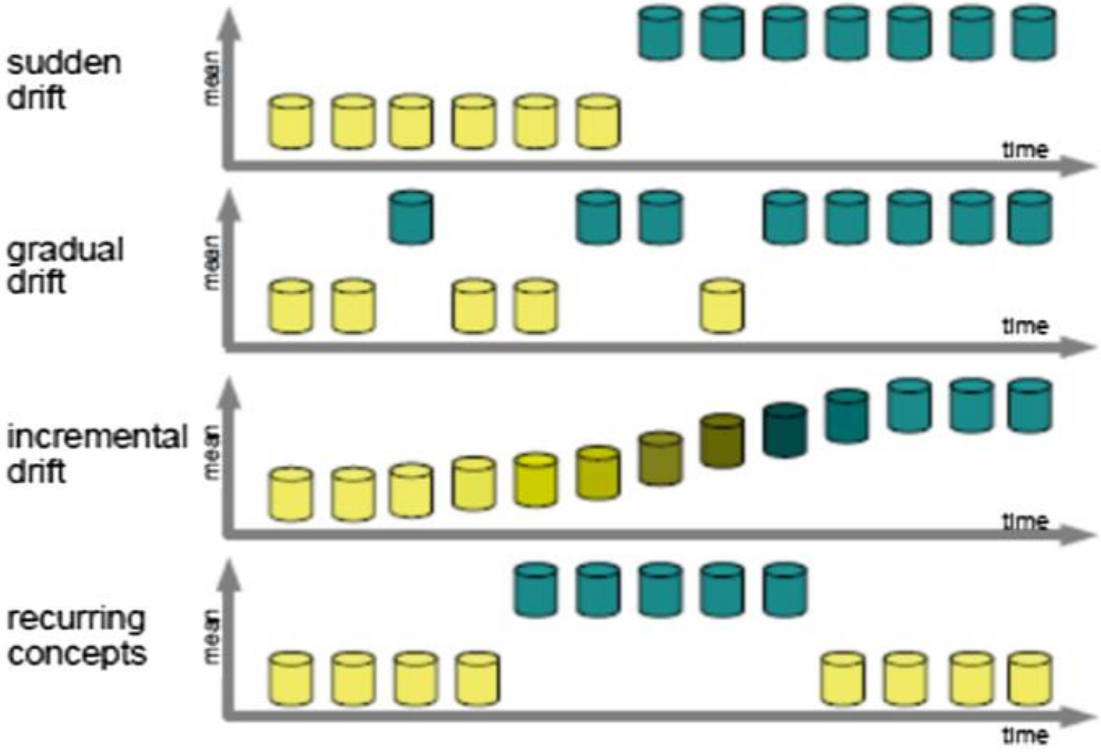


Fig. 3. Types of concept drift

Figure: V. Lemaire et al. «A Survey on Supervised Classification on Data Streams"

CD – A Probabilistic Definition

- Given an input $x_1, x_2, \ldots, x_t$ of class y we can apply bayes theorem:

$$p(y|x_t) = \frac{p(y)p(x_t|y)}{p(x_t)}$$

- $p(y)$ is the prior for the output class (concept)
- $p(x_t|y)$ the conditional probability
- Why do we care?
  - Different causes for changes in each term
  - Different consequences (do we need to retrain our model?)

Dataset Shift Nomenclature

Notation:

- x covariates/input features
- y class/target variable
- p(y, x) joint distribution
- sometimes the x→ y relationship is referred with the generic term "concept"

The nomenclature is based on **causal assumptions**:

- x→y problems: class label is causally determined by input. Example: credit card fraud detection
- y→x problems: class label determines input. Example: medical diagnosis

Moreno-Torres, Jose G., Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. "A Unifying View on Dataset Shift in Classification." Pattern Recognition 45, no. 1 (January 2012): 521–30. https://doi.org/10.1016/j.patcog.2011.06.019.

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: happens 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: happens 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

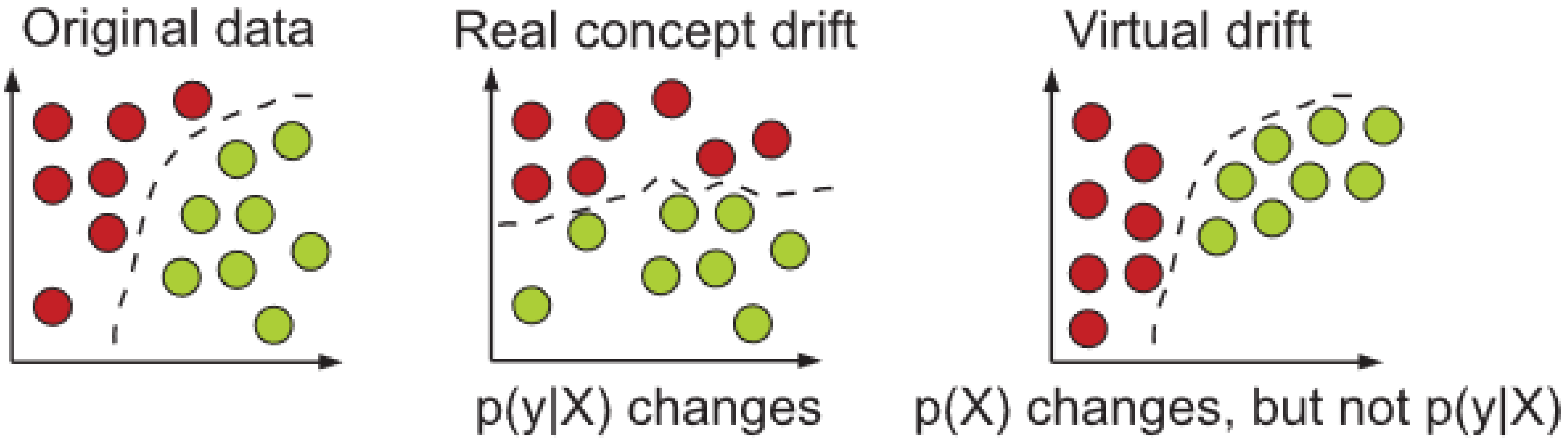
Concept shift:

- $p_{tra}(y|x) \neq p_{tst}(y|x)$ 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)

Moreno-Torres, Jose G., Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. "A Unifying View on Dataset Shift in Classification." Pattern Recognition 45, no. 1 (January 2012): 521–30. https://doi.org/10.1016/j.patcog.2011.06.019.

Dataset Shift in Machine Learning, J. Qui˜nonero-Candela et al. MIT Press, 2008.

Real vs Virtual Drift



Gama, João, et al. A survey on concept drift adaptation. ACM computing surveys (CSUR) 46.4 (2014): 1-37.

Causes of Shifts

Sampling bias:

- The world is fixed but we only see a part of it
- The «visible part» changes over time, causing a shift
- We will also call it virtual drift
- Examples: bias in polls, limited observability of environments, change of domain…

Non-stationary environments:

- The world is continuously changing
- We will also call it real drift
- Examples: weather, financial markets, …

Deep Continual Learning has been mostly focused on "virtual drifts" and with **knowledge accumulation rather than adaptation**.

The Stability-Plasticity Dilemma

Stability-Plasticity Dilemma:

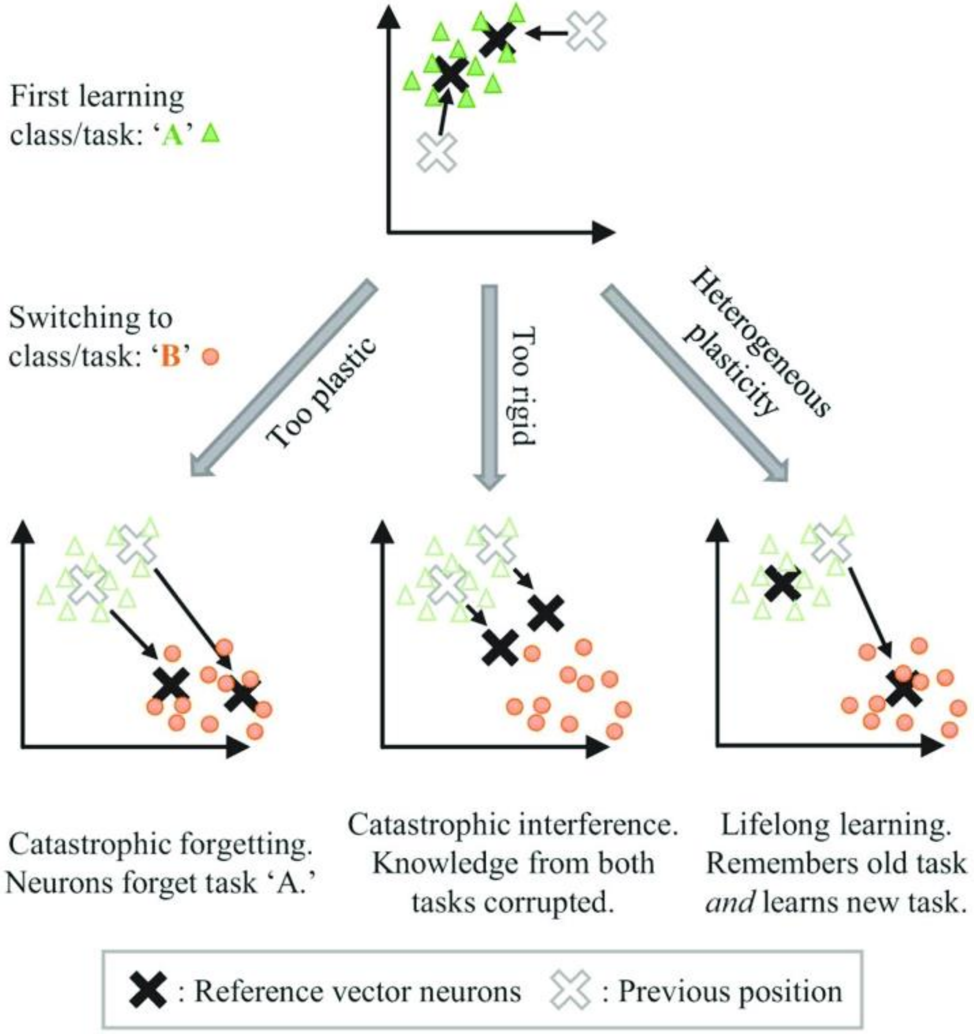
- stability: remember past concepts
- plasticity: learn new concepts

The two objectives often interfere with each other!

- I can freeze the network to prevent forgetting
- I can do a naive finetuning (or even randomly initialize) to have optimal plasticity

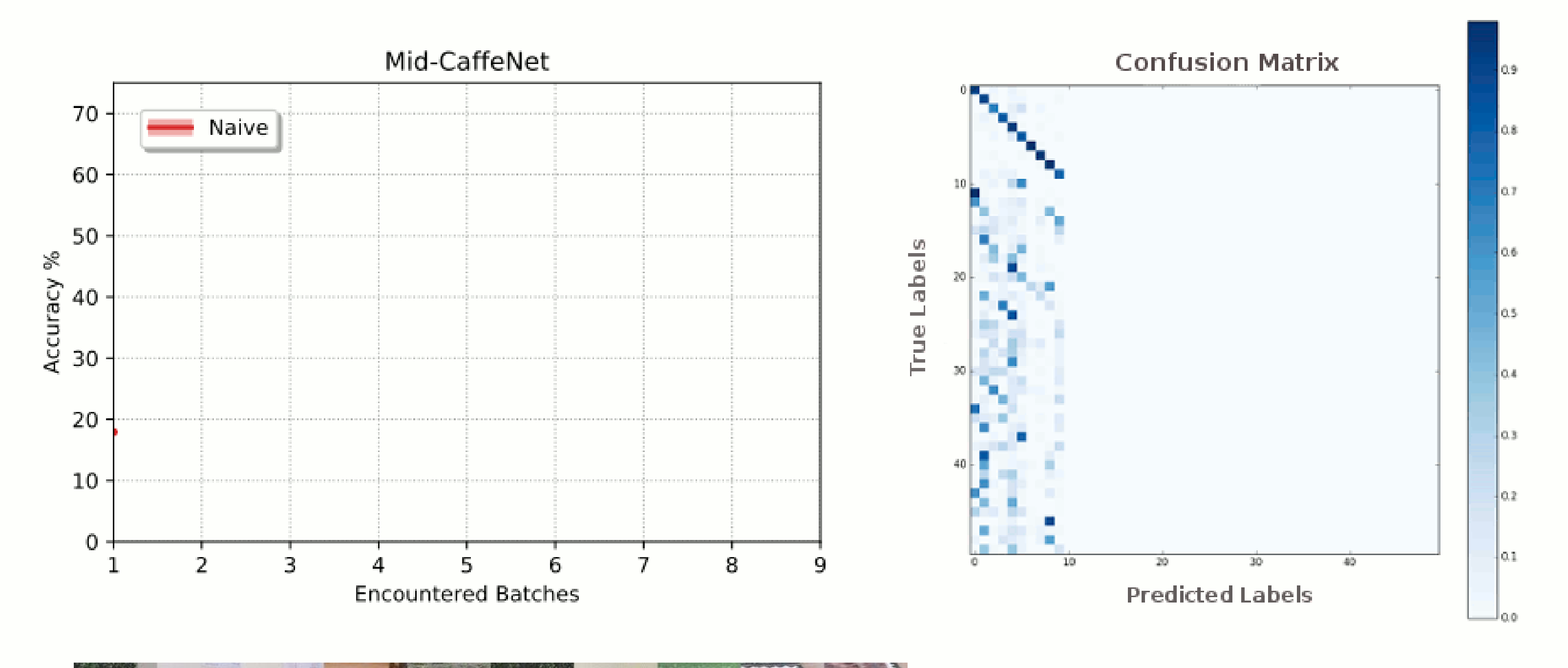
First Problem in Deep Learning: Catastrophic Forgetting

- It's not an unsolvable problem. Most DNN have enough capacity the learn past, current, and future data. We just have to design a proper learning method.



First learning class/task: 'A' △
Switching to class/task: 'B' ●
Too plastic
Too rigid
Heterogeneous plasticity
Catastrophic forgetting. Neurons forget task 'A.'
Catastrophic interference. Knowledge from both tasks corrupted.
Lifelong learning. Remembers old task *and* learns new task.
✖ : Reference vector neurons
✖ : Previous position

Catastrophic Forgetting



- *A set of new objects (classes) each day*

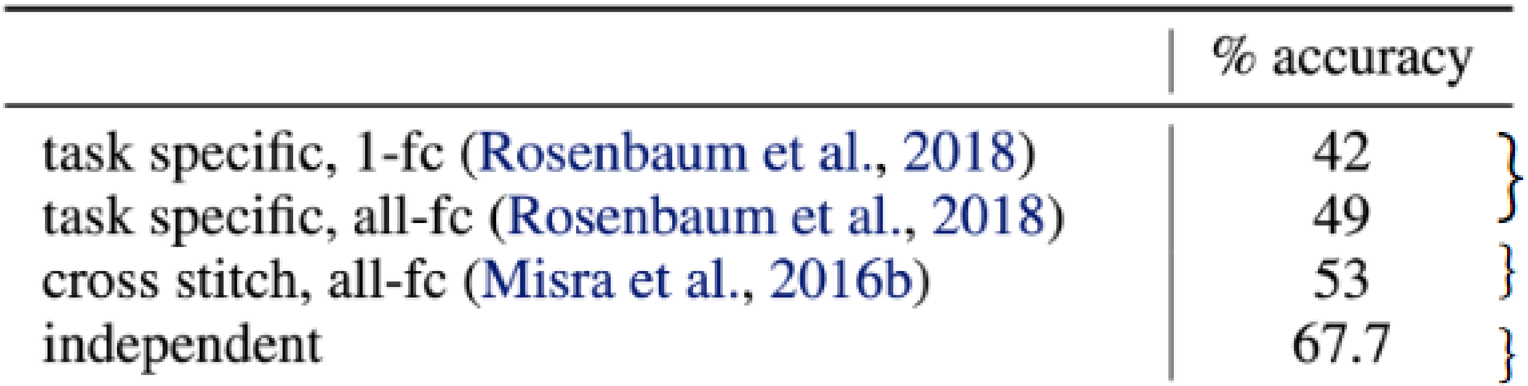
- *10 the first day, 5 the following*

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

Transfer and Interference

Sometimes, forgetting is not caused by incremental training. Even multi-task models have problems with interference!

- **Positive Transfer**: training tasks jointly (i.e. sharing weights) improves the performance on the single tasks
  - if the tasks are small the joint solution is more robust and less prone to overfitting
- **Negative Transfer**:
  - Sometimes independent models are better
  - cross-task interference, different rates of learning
  - representational capacity, MT nets need to be bigger



	% accuracy	
task specific, 1-fc (Rosenbaum et al., 2018)	42	} multi-head architectures
task specific, all-fc (Rosenbaum et al., 2018)	49	
cross stitch, all-fc (Misra et al., 2016b)	53	} cross-stitch architecture
independent	67.7	} independent training

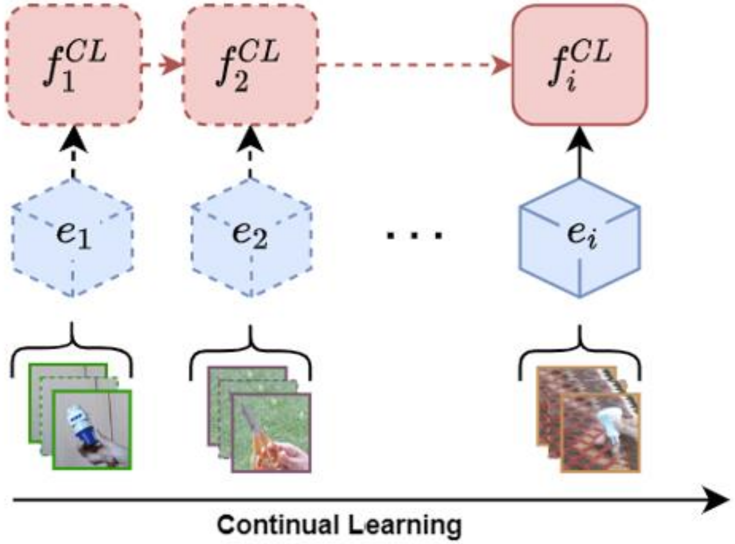
Yu et al. Gradient Surgery for Multi-Task Learning. 2020

Deep Continual Learning

Definition, Objectives, Desiderata

Continual Learning

CL = Incremental Learning from a non-stationary stream
+ **environment information**: task labels, task boundaries, …
+ **constraints**: computational/memory limits, privacy, …
+ **metrics**: minimize forgetting, maximize transfer, …



Suggested review:
T. Lesort et al. "Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges." *Information Fusion*. https://doi.org/10.1016/j.inffus.2019.12.004.

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.

Continual Learning Objectives and Desiderata

In continual learning (CL) data arrives in a streaming fashion as a (possibly infinite) sequence of learning experiences $S = e_1, \ldots, e_n$. 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:

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

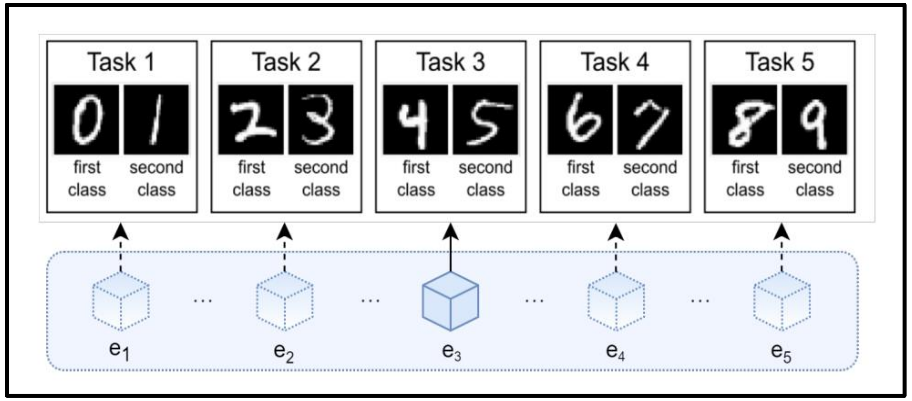
where f^{CL}_i is the model learned after training on experience e_i, $\mathcal{M}_i$ a buffer of past knowledge, such as previous samples or activations, stored from the previous experiences and usually of fixed size. The term t_i is a task label which may be used to identify the correct data distribution.

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

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

$$\mathcal{L}_{exp}(f^{CL}_n, \mathcal{D}^i_{test}) = \sum_{j=1}^{|\mathcal{D}^i_{test}|} \mathcal{L}(f^{CL}_n(\boldsymbol{x}^i_j), y^i_j), \qquad (3)$$

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

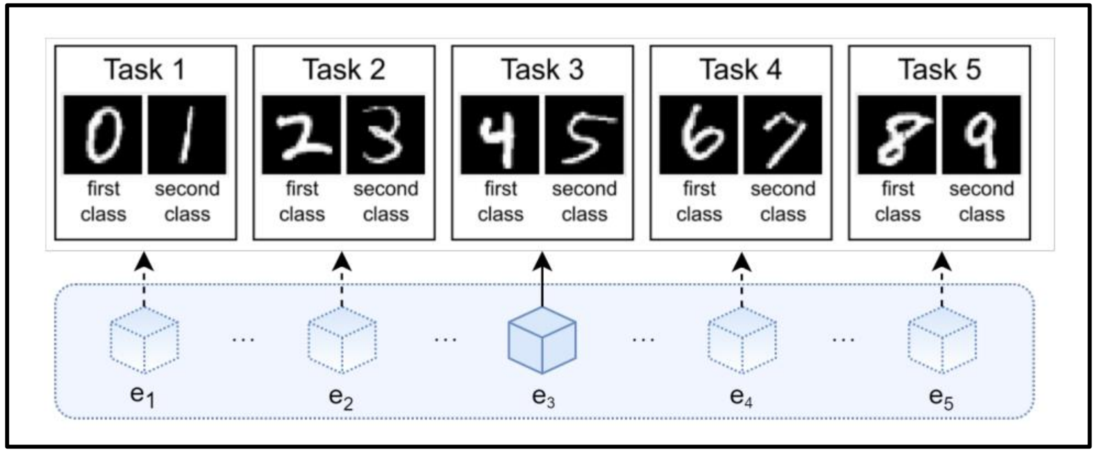


Ex-Model: Continual Learning from a Stream of Trained Models, Carta et al, 2021.

Environment Information – Nomenclature

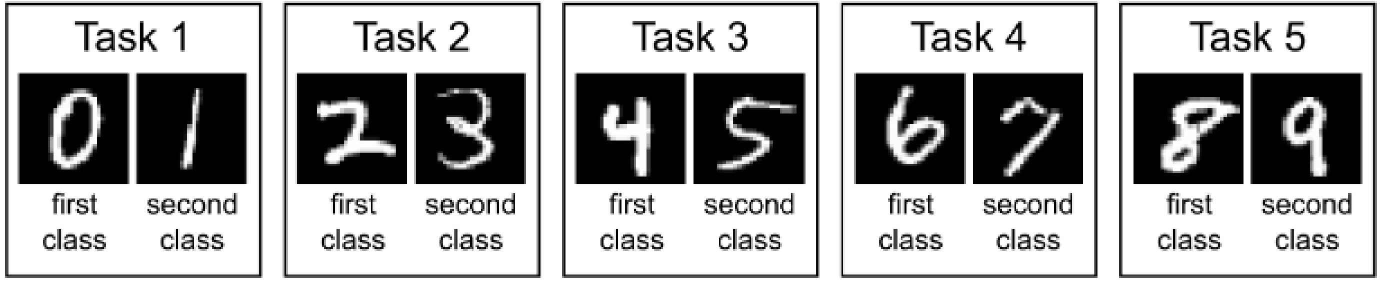
Different streams require different methods

- **Stream**: A list of experiences, each providing a batch of data and some additional information (e.g. task labels)
- **Batch/Online: How much data** do we have in each experience?
- **Class/Domain-incremental**: Do we know the **type of shifts**?
- Do we know **when** the shifts happen?
- Do we have **task labels** at training/inference time?

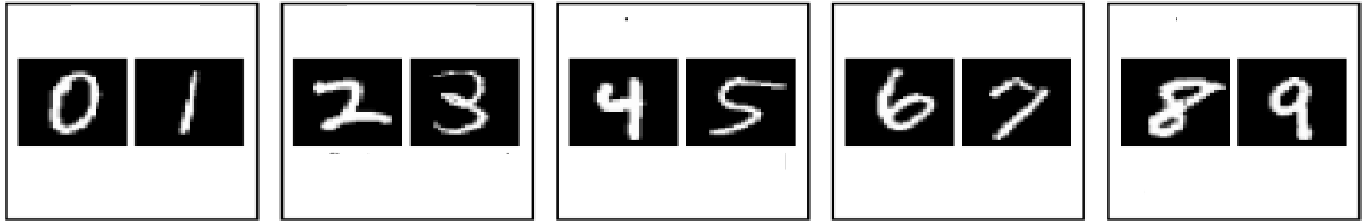


Continual Learning – Toy Scenarios

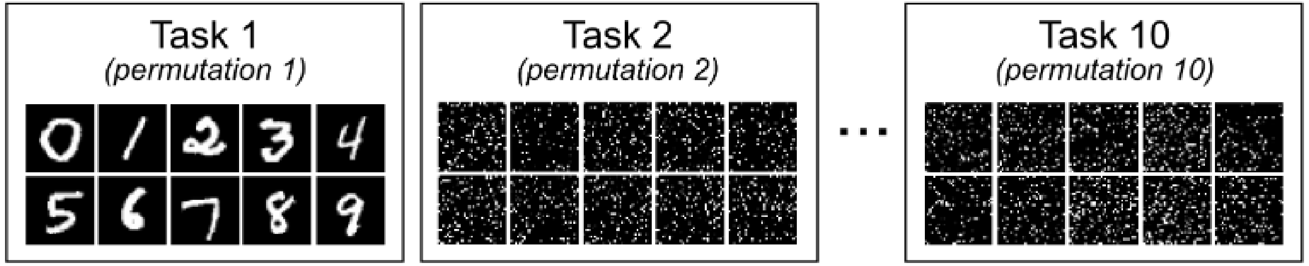
1. **Task-Incremental**: every experience is a different task.



1. **Class-Incremental**: every experience contains examples of different classes of a unique classification problem.



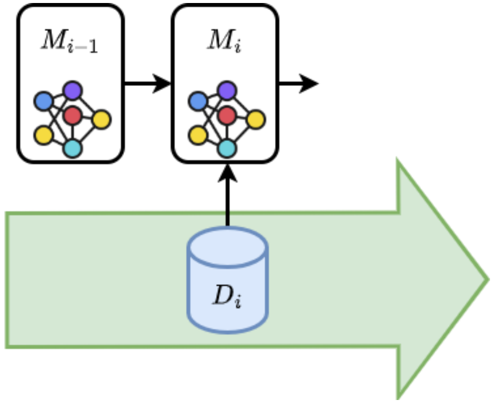
1. **Domain-Incremental**: every experience contains examples (from a different domain) of the same classes.



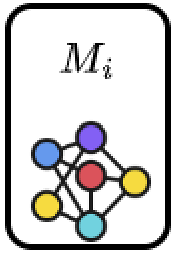
Three scenarios for continual learning, Van de Ven, 2019

Baselines: Naive Finetuning

Train: sequential SGD, each time using only the current data.



Inference: use last model (M_i)

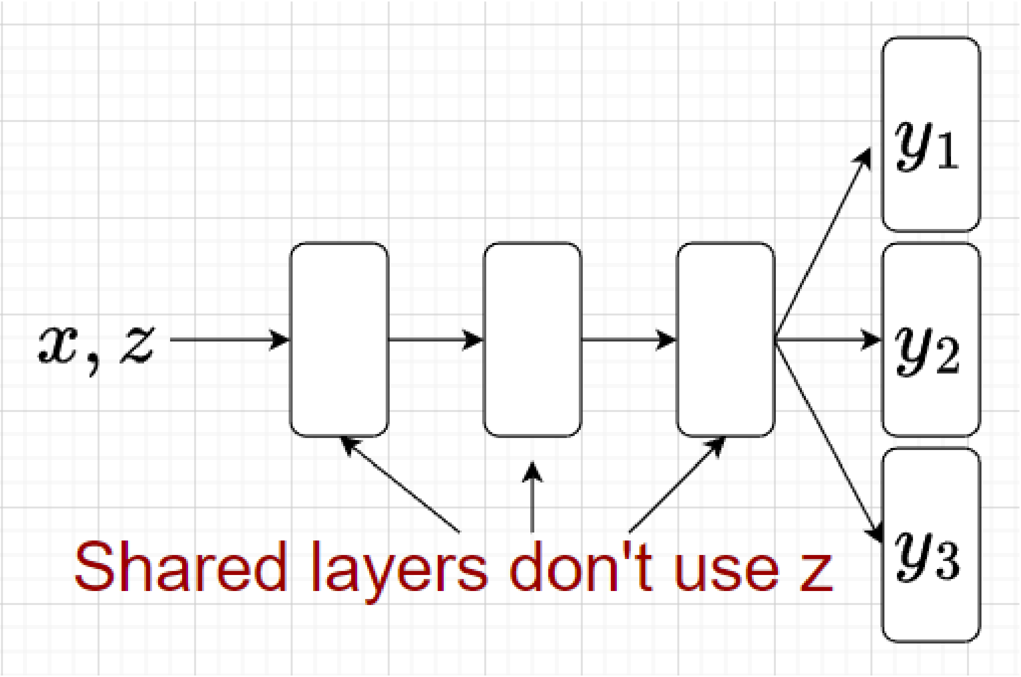


Note: Naive finetuning often results in catastrophic forgetting. CL methods should always beat the Naive baseline

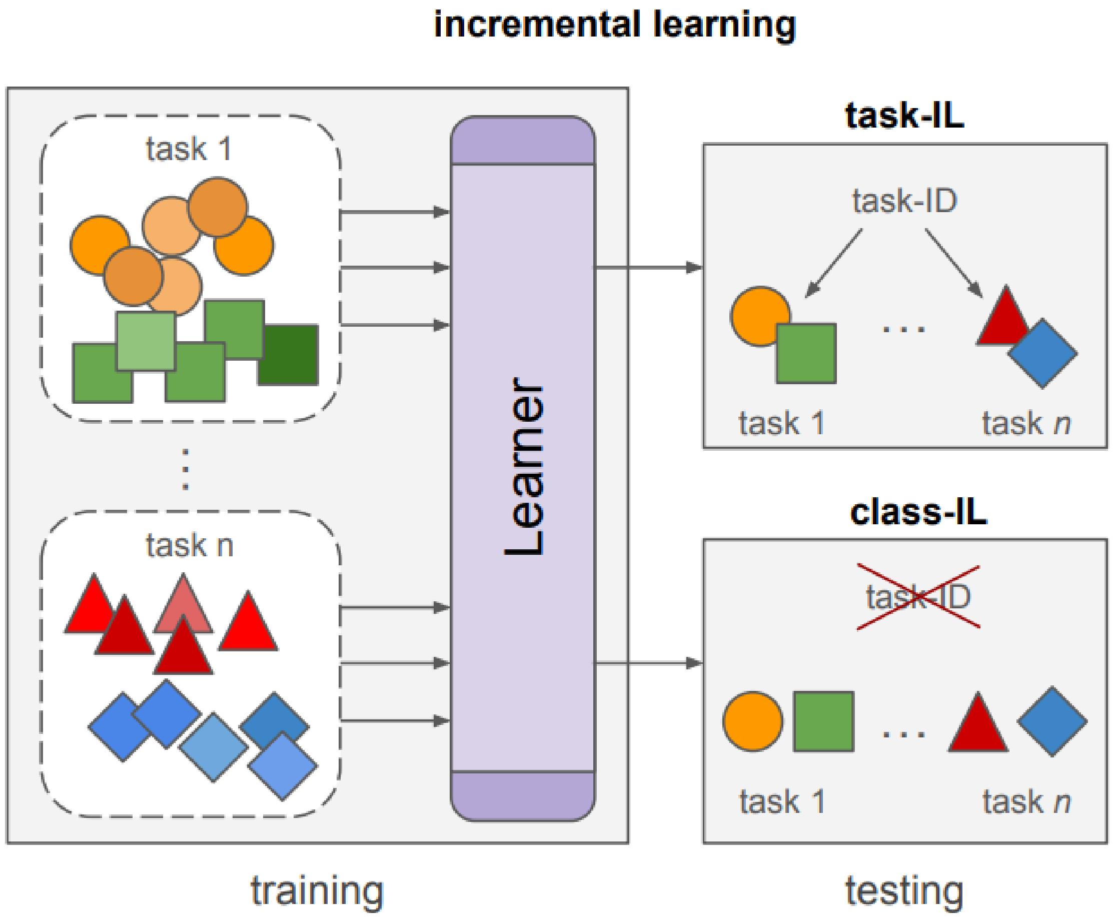
Task Labels and Multi-Head

Multi-Head architectures have:

- a shared feature extractor
- a separate linear classifier (head) for each task
- the correct head is selected for each example via multiplicative gating
- The multi-head architecture is one of the big advantages of having task labels.
- We can also have task-dependent hidden layers (architectural methods)



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

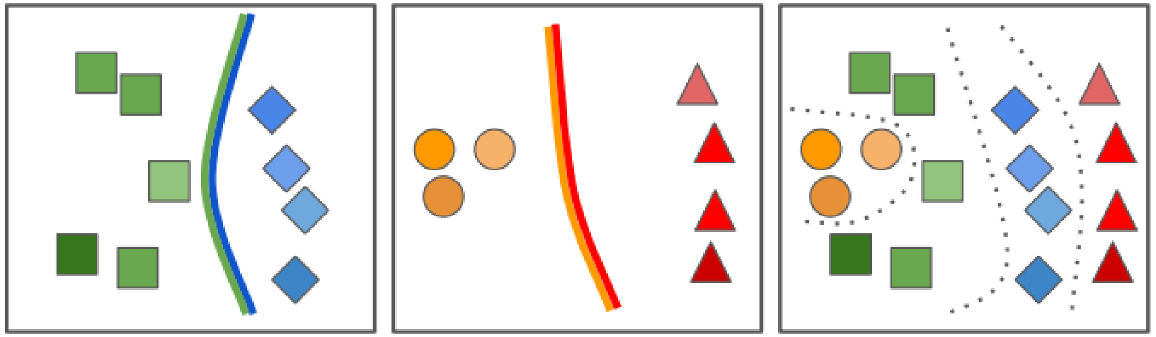


Fig. 2: A network trained continually to discriminate between task 1 (left) and task 2 (middle) is unlikely to have learned features to discriminate between the four classes (right). We call this problem *inter-task confusion*.

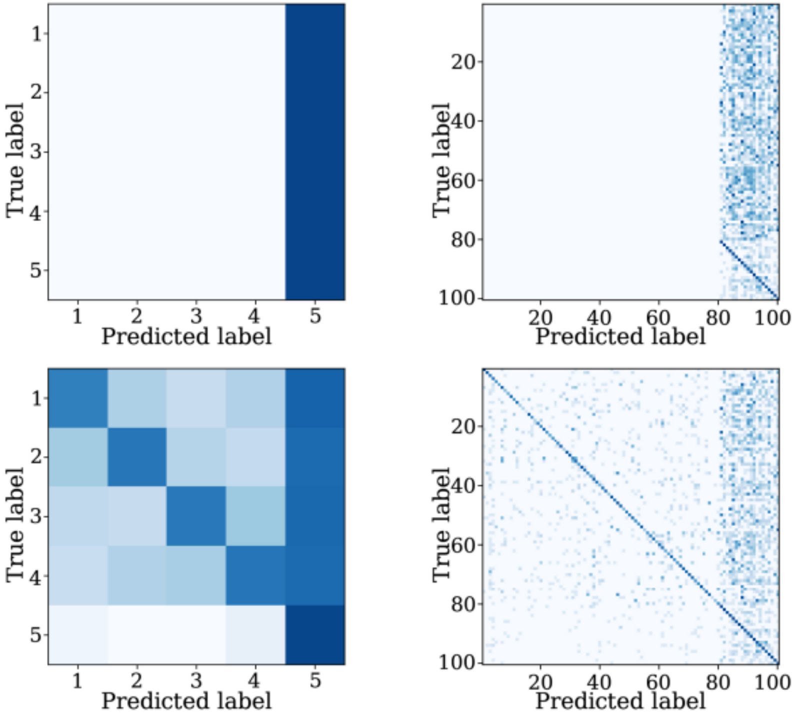


Fig. 3: Examples of task and class confusion matrices for Finetuning (top row) and Finetuning with 2,000 exemplars (bottom row) on CIFAR-100. Note the large bias towards the classes of the last task for Finetuning. By exploiting exemplars, the resulting classifier is clearly less biased.

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

Classifier Bias in CIL

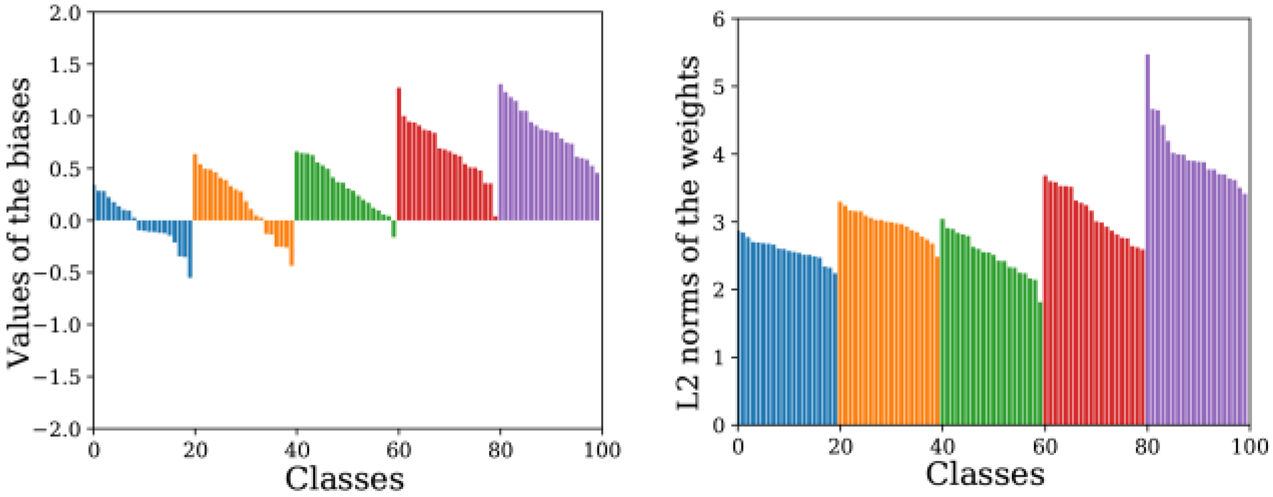


Fig. 4: Bias and weight analysis for iCaRL with 2,000 exemplars on CIFAR-100. We show the ordered biases and norm of the last classification layer of the network for different tasks. Note how the bias and the norm of the weights are higher for the last tasks. This results in a *task-recency bias*.

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

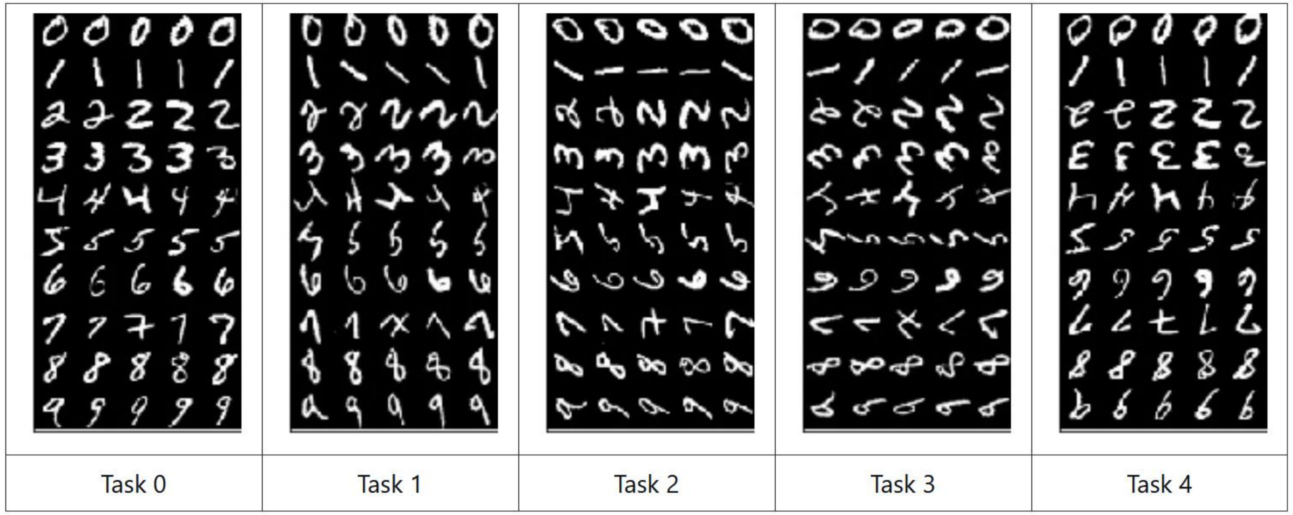
Sharp vs Blurry Shifts

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

Most CL methods deal with sharp drifts

Example of gradual shift: Rotated MNIST

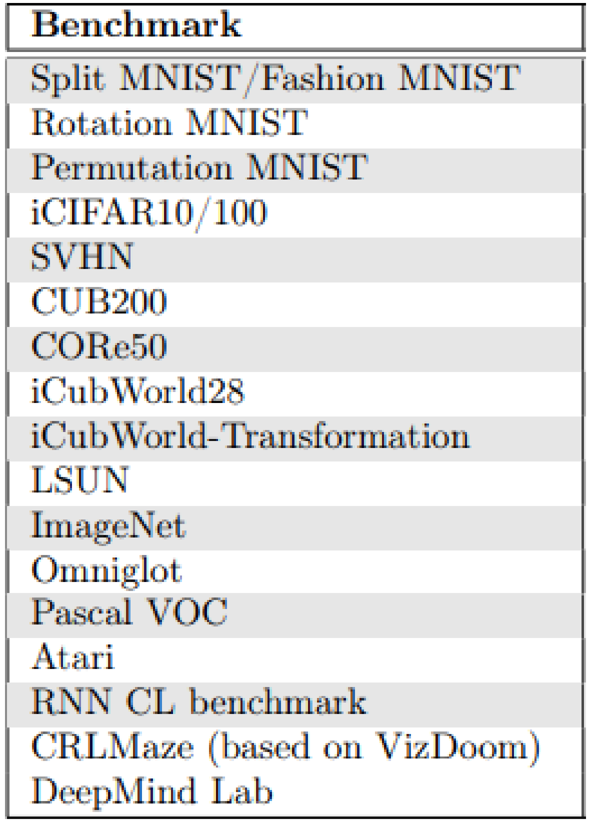
RotatedMNIST:



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

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

Continual Learning Benchmarks



Benchmark
Split MNIST/Fashion MNIST
Rotation MNIST
Permutation MNIST
iCIFAR10/100
SVHN
CUB200
CORe50
iCubWorld28
iCubWorld-Transformation
LSUN
ImageNet
Omniglot
Pascal VOC
Atari
RNN CL benchmark
CRLMaze (based on VizDoom)
DeepMind Lab

Current Focus

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

Recent Growing Trend

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

Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges, Lesort et al. Information Fusion, 2020.

Evaluation

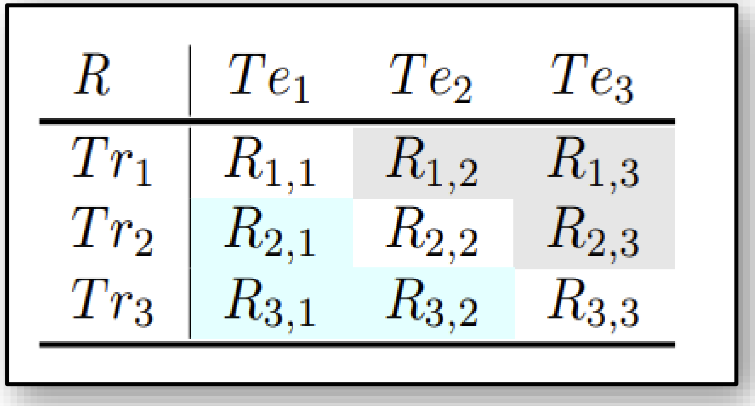
Let N be the stream length, T the current timestep, $R_{t,i}$ be the accuracy on the experience i at time t

Time – which model to use:

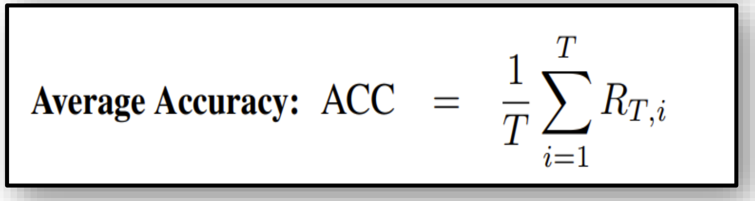
- Last model: $R_{T,T}$
- Averaged over time: $\frac{1}{T+1}\sum_{t=0}^{T}\sum_{i=0}^{t} R_{t,i}$

Data – which data to use:

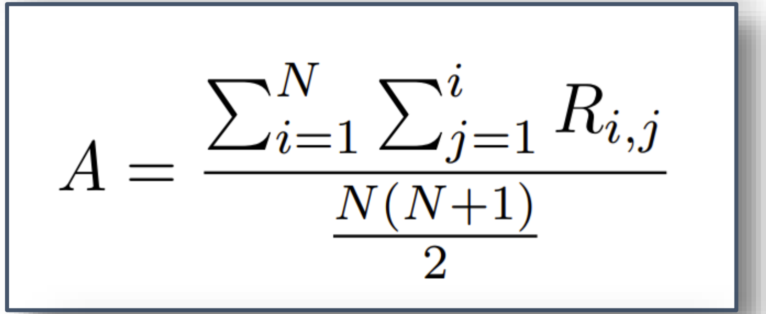
- Current experience: $R_{t,t}$
- Data seen up to now: $\frac{1}{T+1}\sum_{i=0}^{T} R_{T,i}$
- Full stream: $\frac{1}{N}\sum_{i=0}^{N-1} R_{T,i}$
- PAST/PRESENT/FUTURE data
    - Forgetting = past_acc – current_acc
    - Forgetting = - Backward Transfer(BWT)



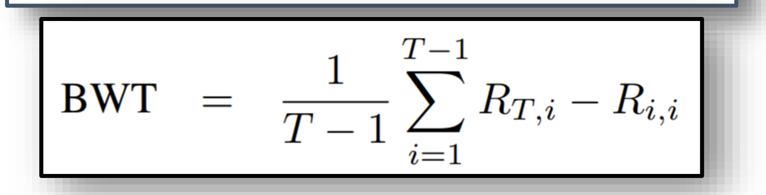
R	Te_1	Te_2	Te_3
Tr_1	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$
Tr_2	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$
Tr_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$



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



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



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

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

Evaluating Representation Learning with Linear Probing

- 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

Continual Learning Approaches

Desiderata

- Replay-Free Continual Learning
- Memory and Computationally Bounded
- Task-free Continual Learning
- Online Continual Learning

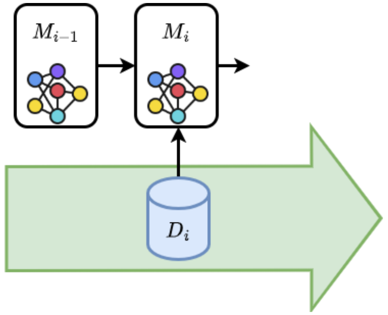
Baselines

- **Finetuning**: sequential SGD, each time using only the current data. Catastrophic forgetting.
- **Ensemble**: Train one independent model for each experience
- **JointTraining / Offline**: Concatenate all the data (keeping task labels) and train *starting from a random initialization*.
- **Cumulative**: for every experience, accumulate all data available up to now $(\bigcup_{k=0}^{i} D_k)$ and re-train *starting from the previous model*.

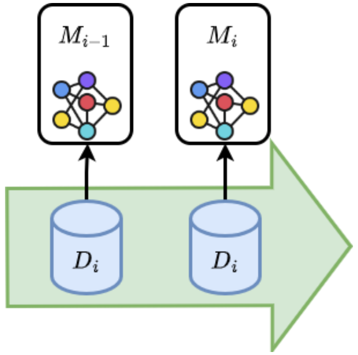
General considerations:

- Given enough data, starting from scratch always achieves a higher performance with enough epochs.
- Starting from the previous model achieves faster convergence than training from scratch.

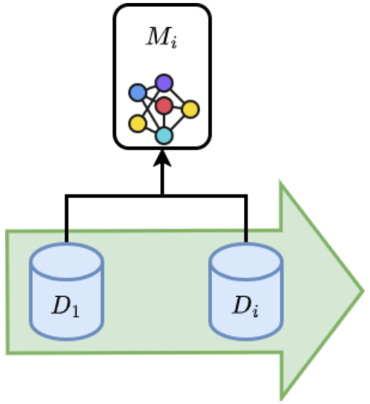
Finetuning (lower bound)



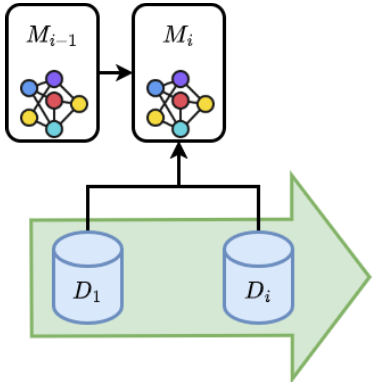
Ensemble + task labels (upper bound)



Offline Joint Training (upper bound)



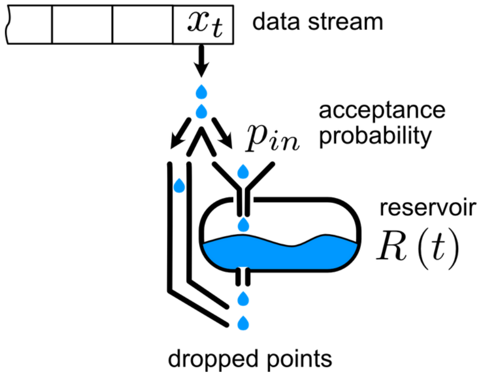
Cumulative (upper bound)



Methodologies

Replay

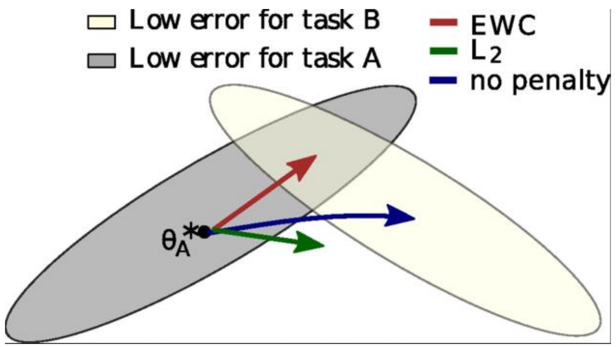
- Keep a buffer of old samples
- Rehearse old samples



Reservoir Sampling

Regularization

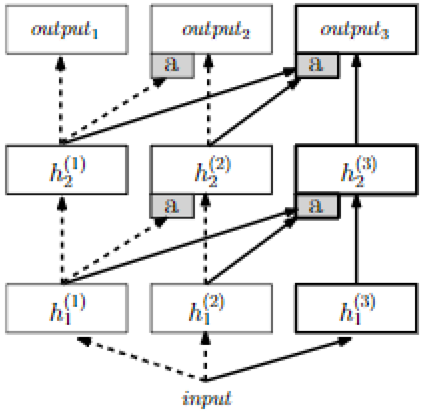
- Regularize the model to balance learning and forgetting



Elastic Weight Consolidation

Architectural

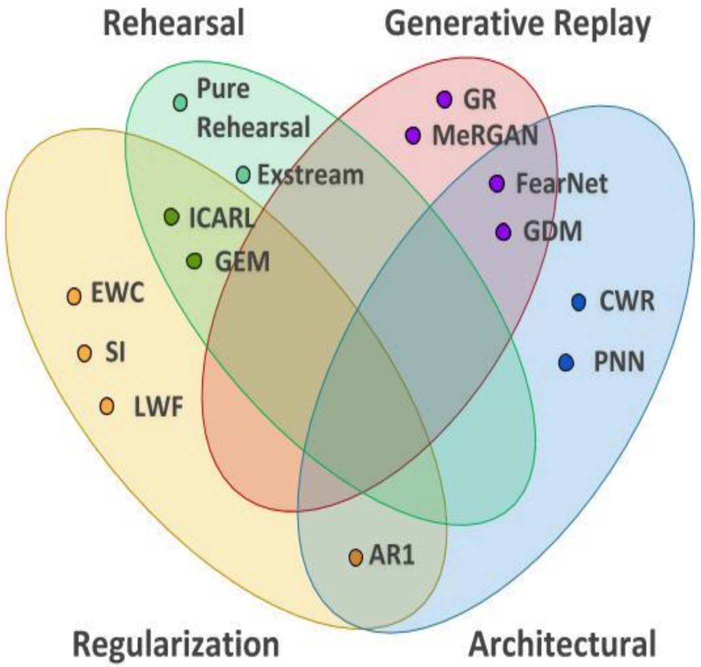
- Expand the model over time with new units/layers

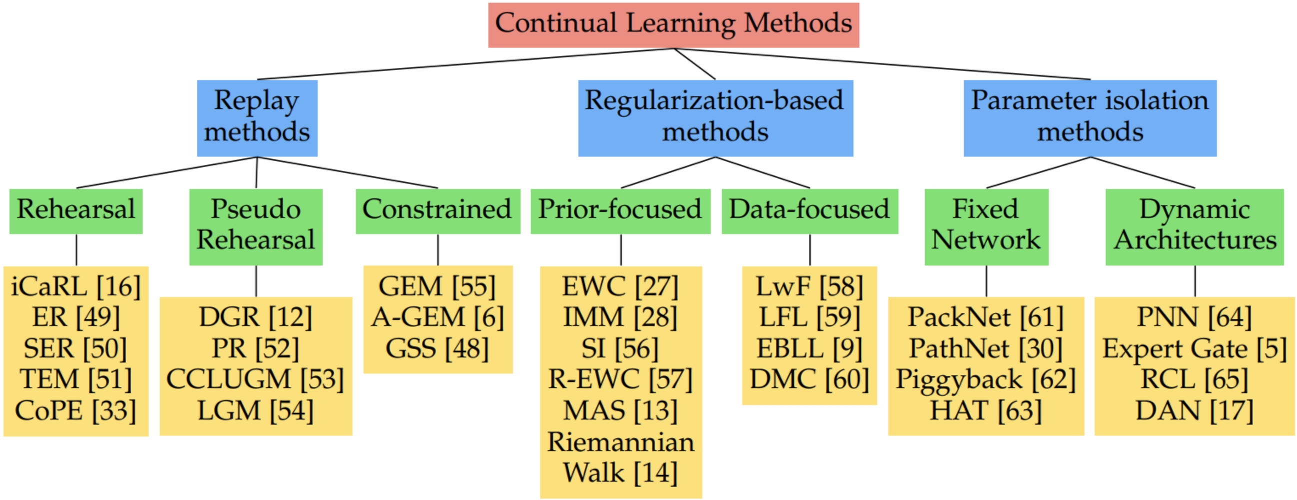


Progressive Neural Networks

Image from https://towardsdatascience.com/reservoir-sampling-for-efficient-stream-processing-97f47f85c11b

Continual Learning Approaches





Continual Learning Methods

- Replay methods
  - Rehearsal: iCaRL [16], ER [49], SER [50], TEM [51], CoPE [33]
  - Pseudo Rehearsal: DGR [12], PR [52], CCLUGM [53], LGM [54]
  - Constrained: GEM [55], A-GEM [6], GSS [48]
- Regularization-based methods
  - Prior-focused: EWC [27], IMM [28], SI [56], R-EWC [57], MAS [13], Riemannian Walk [14]
  - Data-focused: LwF [58], LFL [59], EBLL [9], DMC [60]
- Parameter isolation methods
  - Fixed Network: PackNet [61], PathNet [30], Piggyback [62], HAT [63]
  - Dynamic Architectures: PNN [64], Expert Gate [5], RCL [65], DAN [17]

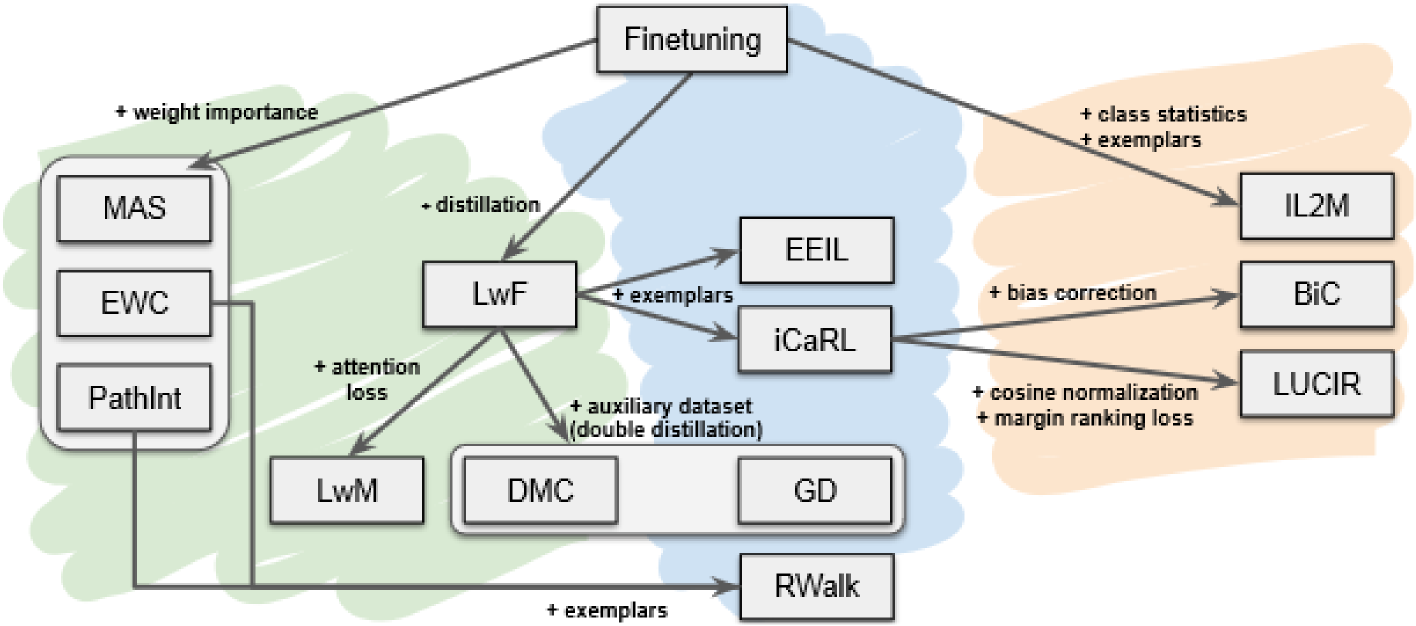
Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges, Lesort et al. Information Fusion, 2020.

A continual learning survey: Defying forgetting in classification tasks. De Lange et al, TPAMI 2021.

CL Strategy Components

CL methods can be combined together

- Regularization +
- Replay +
- Architectural +
- Bias correction: methods for output layer



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

Continual Learning Objective

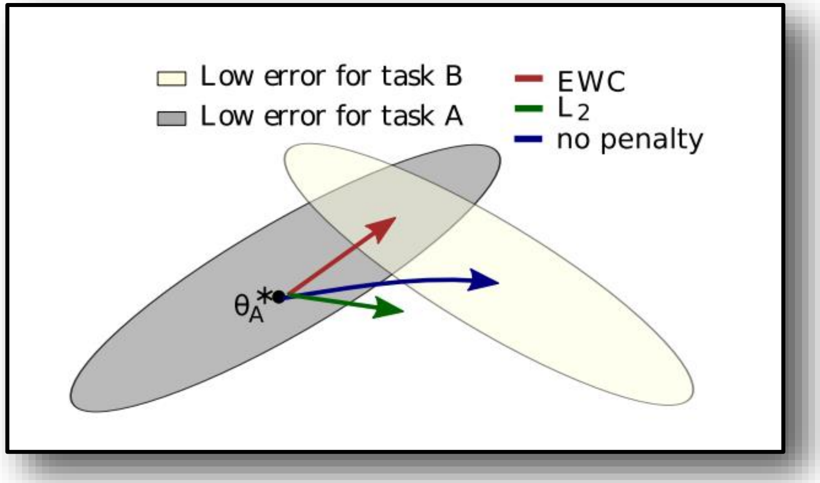
CL Loss:

$$\mathcal{L}^{\text{tot}}(\theta) = \mathcal{L}^{\text{new}}(\theta) + \mathcal{L}^{\text{old}}(\theta)$$

- We estimate $\mathcal{L}^{\text{new}}(\theta)$ from new data (easy)
- How do we estimate $\mathcal{L}^{\text{old}}(\theta)$?
  - We don’t have access to the old data
  - We need to approximate it

Importance-Based Methods

- An example of **prior-focused** method: Model loss function of previous tasks with surrogate losses.
- **IDEA**: important weights should not change. We can change unimportant weights only.
  - Deep networks are overparameterized. This means that we should have a lot of free capacity.
  - Each task should have a small number of important weights.
- **PROBLEM**: how do we find the important weights? Fisher Information!



Overcoming catastrophic forgetting in neural networks, Kirkpatrick et al, PNAS 2017.

CL with Bayesian Updates

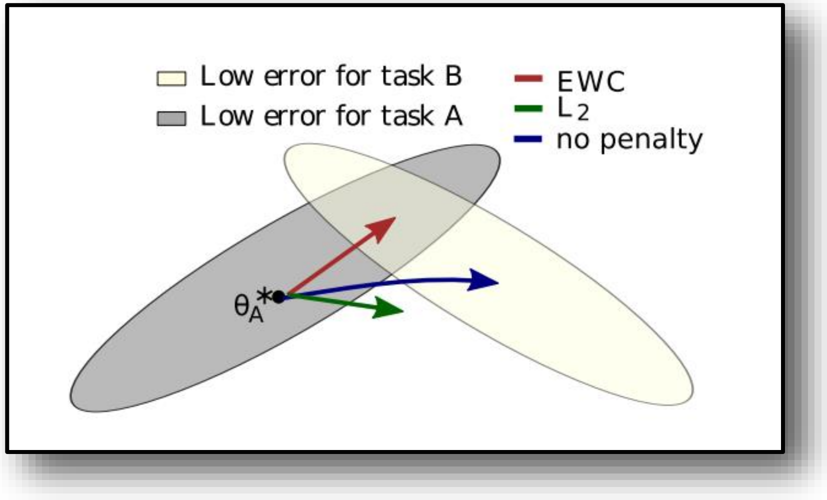
Bayesian Learning:

- Starting from a prior $p(\theta)$
- We learn from data D_a
- We obtain a posterior $p(\theta|D_a)$

Sequential Bayesian Update:

- Given the current model θ_t trained up to experience t
- The posterior $p_t(\theta_t \mid D_{t-1}, \dots, D_1)$ becomes the new prior
- Approximation: the posterior will be approximated, which will introduce errors over time
- Can we compute the DNN posterior? Not really, but we can approximate it

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D}_{new} \mid \theta) + \log p(\theta \mid \mathcal{D}_{old}) - \log p(\mathcal{D}_{new})$$



$$\begin{array}{ll}
p(\theta \mid \mathcal{D}_A, \mathcal{D}_B) = & \text{Bayes' rule} \\
\dfrac{p(\mathcal{D}_A, \mathcal{D}_B \mid \theta) p(\theta)}{p(\mathcal{D}_A, \mathcal{D}_B)} = & \text{conditional independence} \\
\dfrac{p(\mathcal{D}_A \mid \theta) p(\mathcal{D}_B \mid \theta) p(\theta)}{p(\mathcal{D}_A) p(\mathcal{D}_B)} = & \text{Bayes' rule on first numerator term} \\
\dfrac{\frac{p(\theta \mid \mathcal{D}_A) p(\mathcal{D}_A)}{p(\theta)} p(\mathcal{D}_B \mid \theta) p(\theta)}{p(\mathcal{D}_A) p(\mathcal{D}_B)} = & \text{some terms cancel out} \\
\dfrac{p(\theta \mid \mathcal{D}_A) p(\mathcal{D}_B \mid \theta)}{p(\mathcal{D}_B)}. &
\end{array}$$

EWC – Laplace Approximation

Laplace Approximation:

- True posterior $p(\theta \mid \mathcal{D}_{old})$ is intractable
- We approximate it with a gaussian with mean θ_t (MAP solution) and covariance $F(\theta_t)$
  - Where F is the Fisher information
- We need the optimal model (enough data and iterations to converge)

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D}_{new} \mid \theta) + \log p(\theta \mid \mathcal{D}_{old}) - \log p(\mathcal{D}_{new})$$

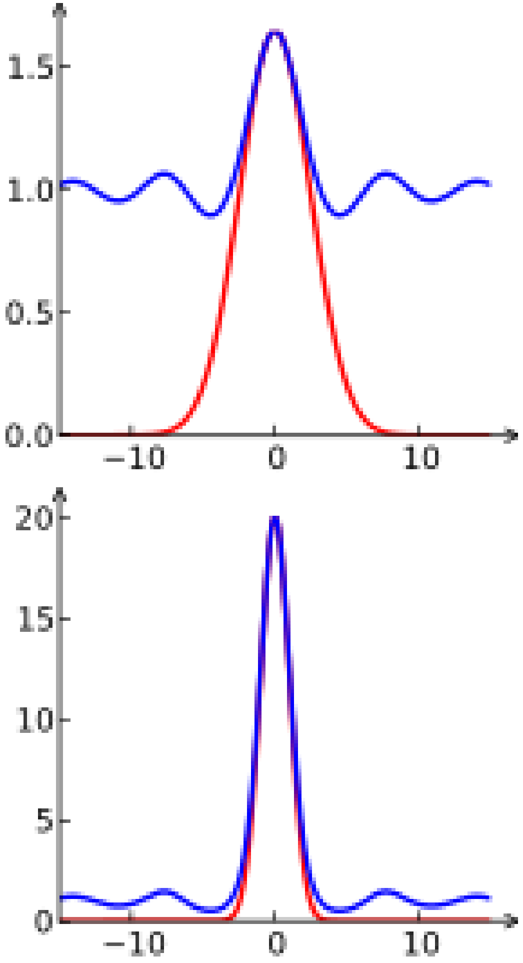


Image source: wikimedia

Fisher Information

- **Fisher Information**:$\mathrm{F}(\theta^*) = \mathrm{E}\left[\left(\frac{\partial}{\partial\theta}\ln f(X;\theta^*)\right)^2\right]$

- If θ^* are the optimal parameters, the Fisher information is equivalent to the curvature of the $KL(\theta\ |\theta^*)$
    - This is approximately true at the task boundaries, assuming we have enough data (and iterations) to train the model until convergence and that the DNN is large enough
- The KL is not the loss function, but it’s a good distance measure and often close to the loss function (e.g. crossentropy)
- The Fisher information $\mathrm{F}(\theta)$ measures how sensitive the distribution is wrt changes to each parameter
- Easy to compute: we only need the gradients
- In our setting, the expectation is taken by sampling $p(x)$ from the data and $p(y|x)$ from the model (not the data!)

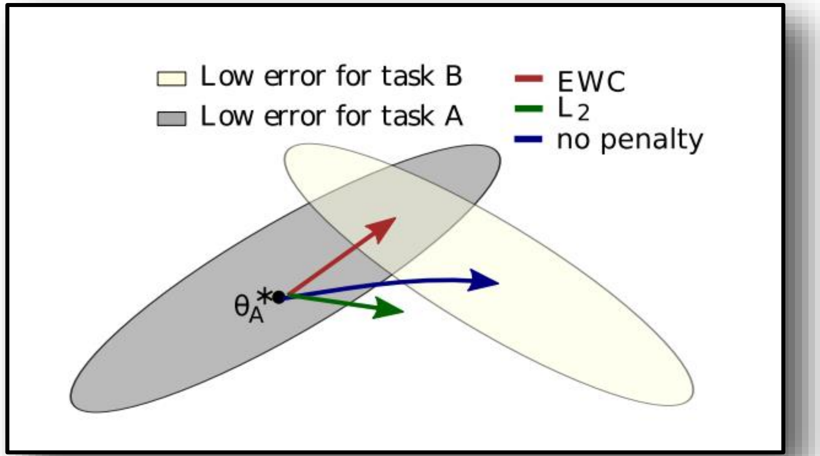
Elastic Weights Consolidation (EWC)

Key Idea: after training, store tuple of $< \theta^{i-1}, F^{i-1} >$ and use the Fisher Information F^{i-1} as an importance value.

EWC loss is like the l2 loss with:

- F^{i-1}, the Fisher diagonal coefficients as weights for each parameter
- The previous weights θ^{i-1} as the "center" of the loss (i.e. zero loss point)

Elastic: weights are pulled towards the previous solution (like the l2 decay), weighted by their importance.



$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

Overcoming catastrophic forgetting in neural networks, Kirkpatrick et al, PNAS 2017.

Importance weights follow a logarithmic distribution

This is a good news for us because having few important parameters means that we can use a strong regularization coefficient for them while having enough free capacity to learn new tasks.

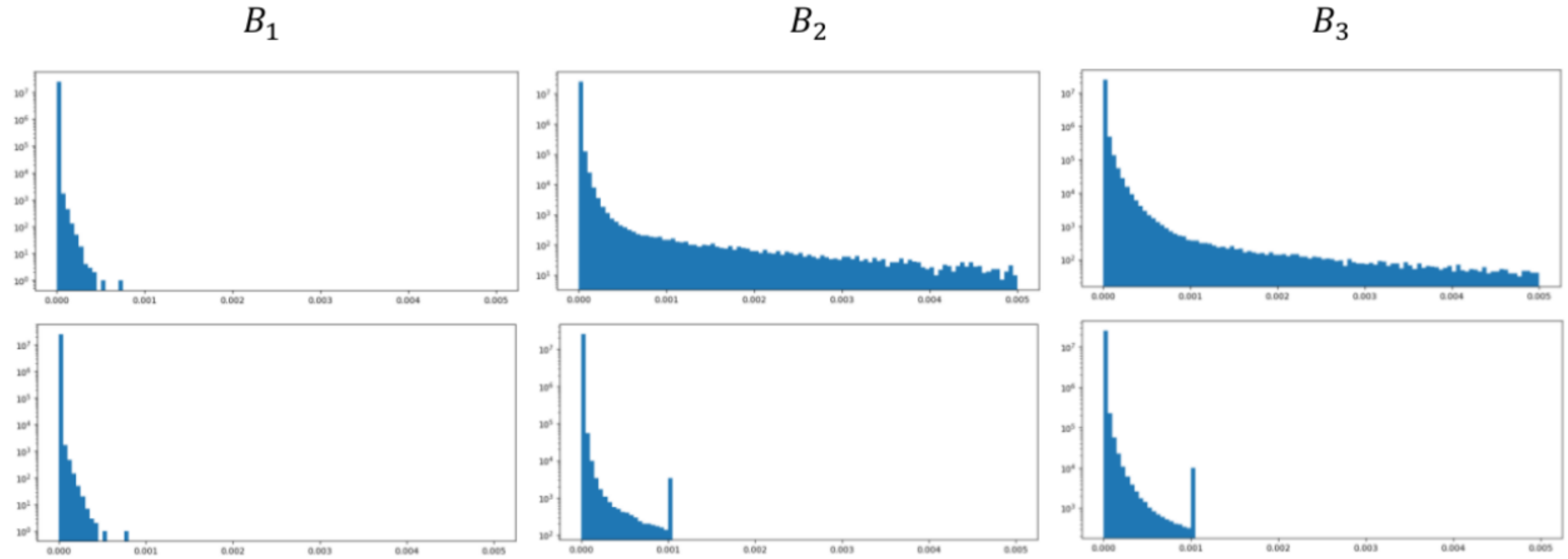


Figure 4: CaffeNet trained by EWC on CORe50 SIT (details in Section 4.2). The first row shows F values distribution denoting a long tail on the right: considering the logarithmic scale, the number of weights values taking high values in F is quite limited. The second row shows the normalized matrix $\hat{F}$ obtained by averaging F values and max clipping to 0.001. Saturation to 0.001 is well evident, but after B_3 the fraction of saturated weights is small (about 1/1000).

Continuous Learning in Single-Incremental Tasks, Maltoni & Lomonaco, Neural Networks, 2019.

Recap

- EWC provides a good proxy loss for past experiences
- Computing the Fisher requires only the gradients (computed at boundaries)
- Many extensions of EWC with a better (non-diagonal) approximation of the Fisher or with estimates of the curvature of the loss function (e.g. Synaptic Intelligence)

Problems:

- Needs task boundaries and large batches, which makes it useless in online settings
- In «separate» mode it has a linear cost in the number of tasks (one Fisher and one model stored for each task)

Online CL and Replay

Online CL

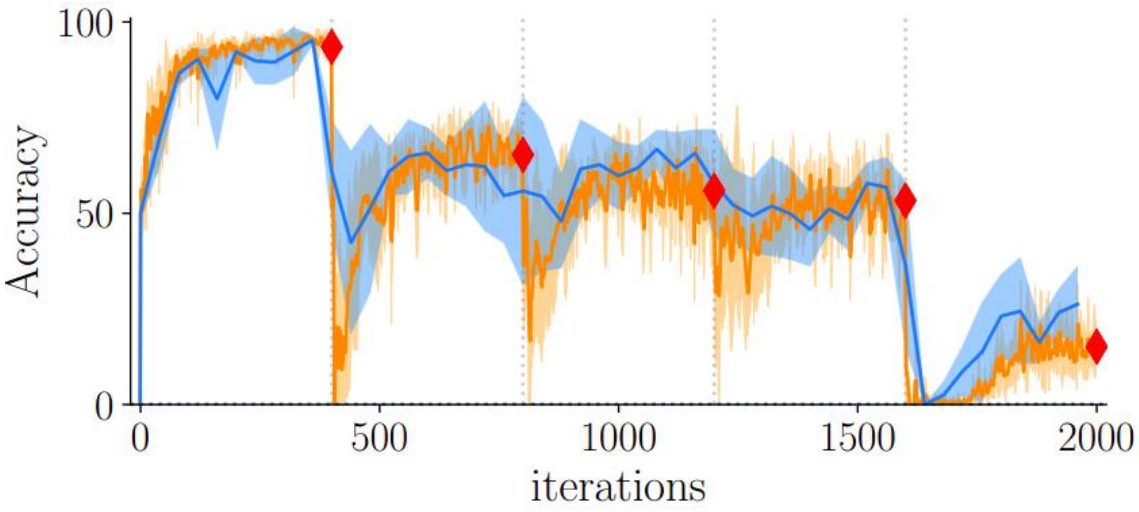
- DNN learns one (small) mini-batch at a time
- Drifts happen over time
- No access to task labels
- Ideally, no knowledge about task boundaries (many methods still require them)
- Replay is allowed

Online CL – Desiderata

Evalution in Online CL is more difficult because the stream is longer, we have less data at each experience, 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

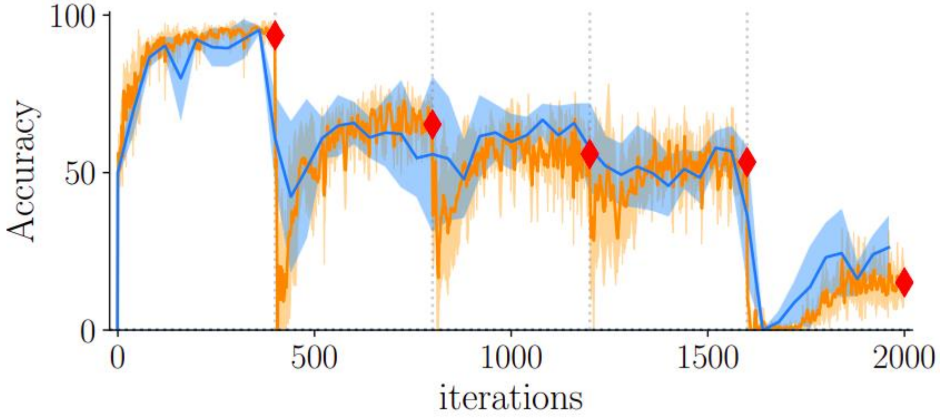
Knowledge Accumulation

- **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

Red diamonds = task boundaries



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

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

Replay Algorithm

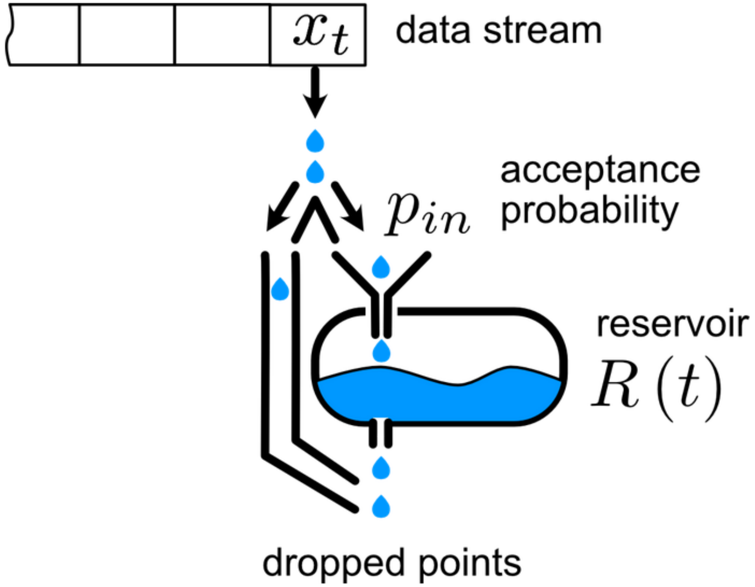
Parameters: memory size

During training (finetuning + rehearsal):

- Sample from the current data
- Sample from the buffer using sampling policy
- Do SGD step on the concatenated mini-batch

After each experience (buffer update):

- Use insertion policy to choose data from the current experience
- Add example to the buffer
- Use removal policy if the buffer is too big



Replay

Good News: Replay is a simple, general and effective strategy for CL.

- Approximates an i.i.d distribution
- Approximate cumulative training
- Relatively cheap in terms of computations

Bad News:

- Memory limitations or privacy constraints
- Scaling: for long streams we may need to store a large buffer. Memory increases over time

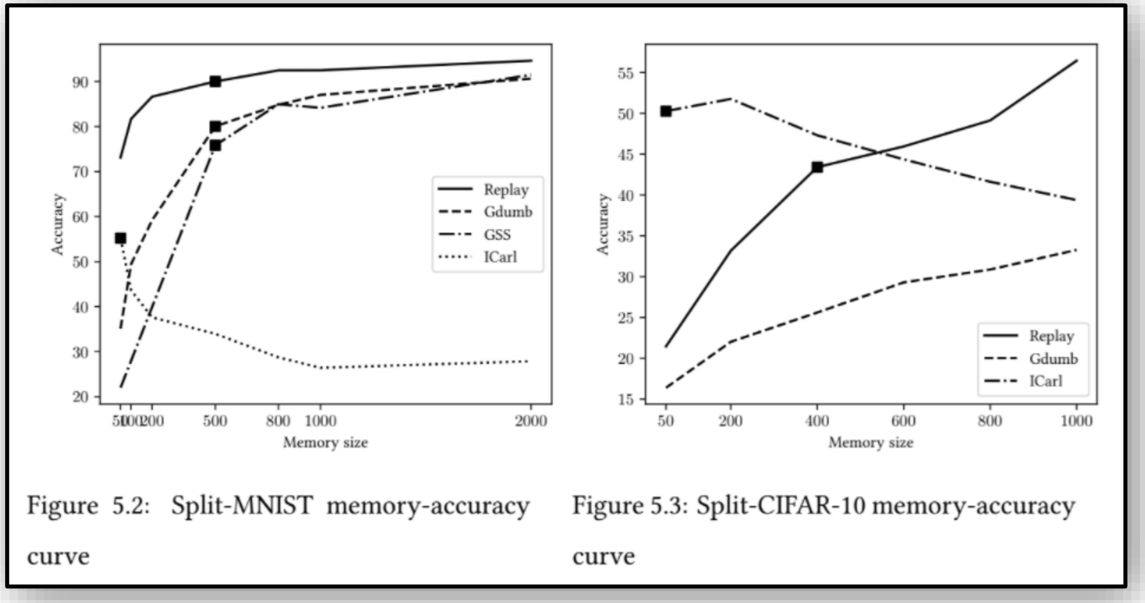


Figure 5.2: Split-MNIST memory-accuracy curve

Figure 5.3: Split-CIFAR-10 memory-accuracy curve

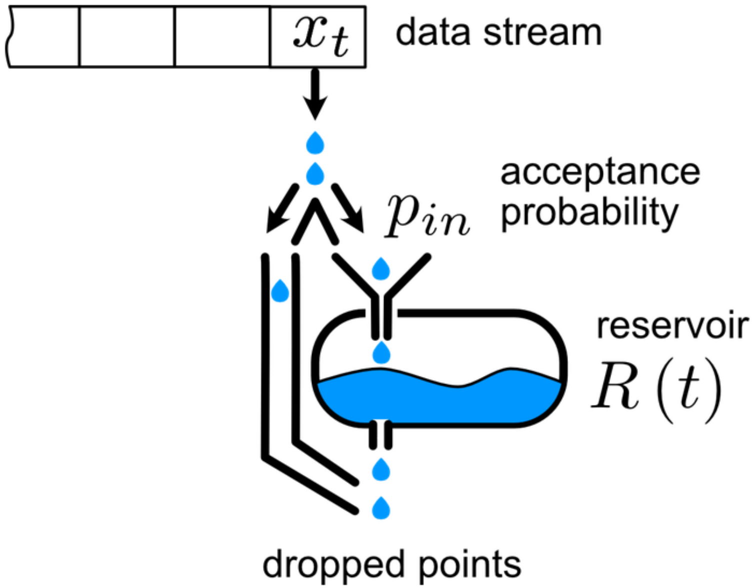
Reservoir Sampling

Reservoir sampling: uniform random sampling, without replacement, of K items from an infinite stream S.

At time N, we want $p(x_t \in M) = \frac{K}{N}, \forall t \leq N$

Class-Balanced Reservoir Sampling (CBRS):

One RS buffer for each class

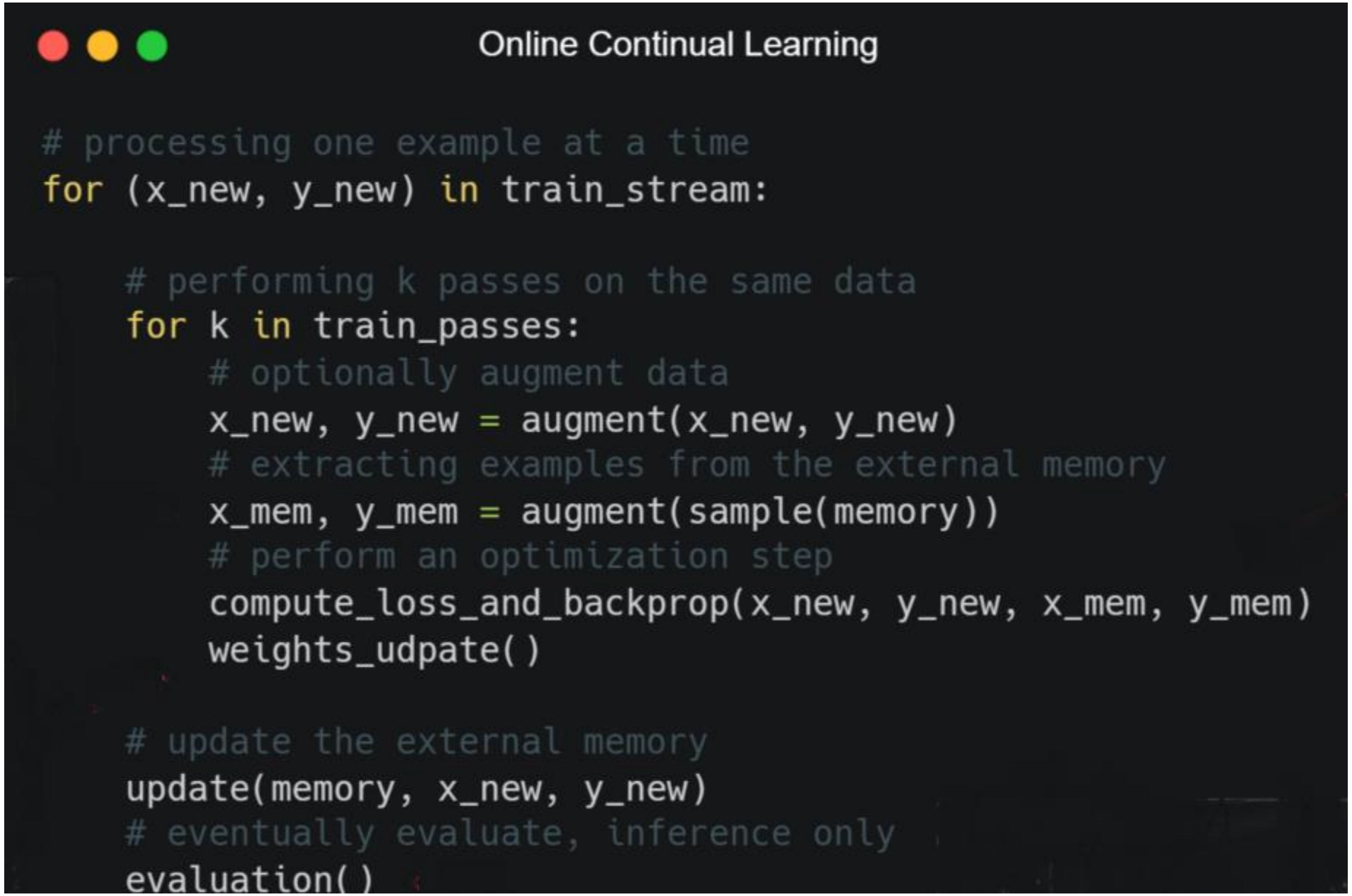


```
(* S has items to sample, R will contain the result *)
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i := 1 to k
      R[i] := S[i]

  // replace elements with gradually decreasing
probability
  for i := k+1 to n
    (* randomInteger(a, b) generates a uniform integer
       from the inclusive range {a, ..., b} *)
    j := randomInteger(1, i)
    if j <= k
        R[j] := S[i]
```

Online Continual Learning with Replay

Notice that we apply different augmentations at each step!



```
# processing one example at a time
for (x_new, y_new) in train_stream:

    # performing k passes on the same data
    for k in train_passes:
        # optionally augment data
        x_new, y_new = augment(x_new, y_new)
        # extracting examples from the external memory
        x_mem, y_mem = augment(sample(memory))
        # perform an optimization step
        compute_loss_and_backprop(x_new, y_new, x_mem, y_mem)
        weights_udpate()

    # update the external memory
    update(memory, x_new, y_new)
    # eventually evaluate, inference only
    evaluation()
```

Soutif-Cormerais, Albin, et al. "A comprehensive empirical evaluation on online continual learning." *ICCV Workshops*. 2023.

Latent Replay

- **PROBLEM**: Replay in the input space is inefficient and biologically implausible. Also memory intensive with high resolution images
- **SOLUTION**: replay latent activations
  - Good Accuracy-Memory-Computation trade-offs.
  - There is no obvious choice for the layer. Middle layers can be very wide.
  - If we allow lossy storage activations can be compressed a lot.
- **Algorithm**:
  - Store latent representation
  - Forward new samples up to latent replay layer
  - Concatenate new and stored representations
  - Forward to the output layer

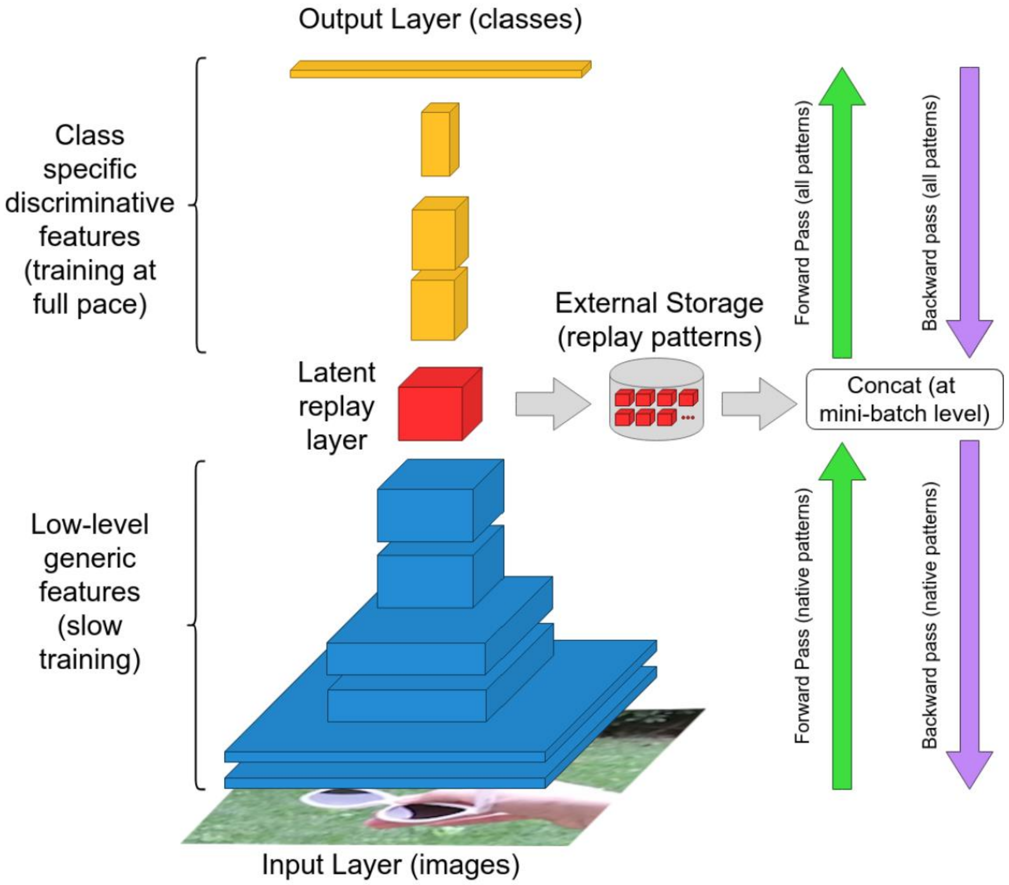

Output Layer (classes)
Class specific discriminative features (training at full pace)
Latent replay layer
External Storage (replay patterns)
Concat (at mini-batch level)
Forward Pass (all patterns)
Backward pass (all patterns)
Low-level generic features (slow training)
Forward Pass (native patterns)
Backward pass (native patterns)
Input Layer (images)

Figure 1: Architectural diagram of Latent Replay.

Latent Replay for Real-Time Continual Learning. Pellegrini et al. IROS, 2019.

Freezing + Latent Replay

- Low layers are trained early during training. They don’t change much afterward.
- We can freeze them at some point.
- Improves latent replay. If we don’t freeze the latent representation in the buffer will become outdated over time

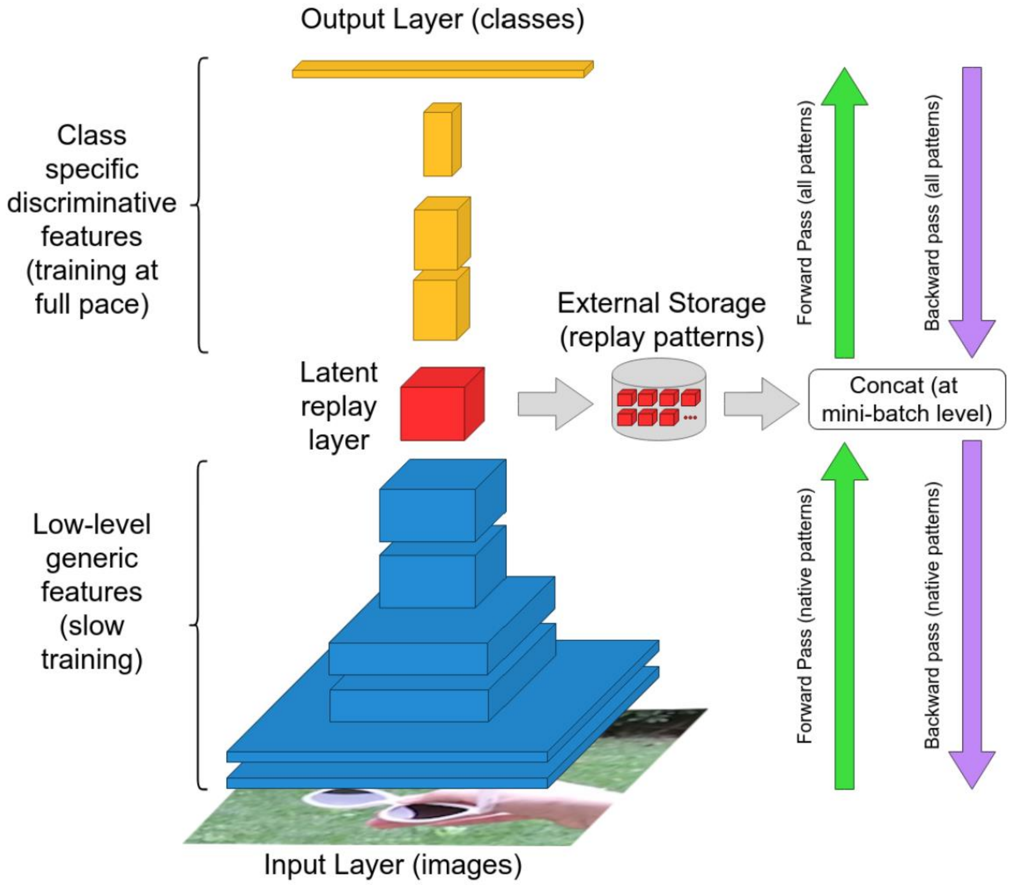
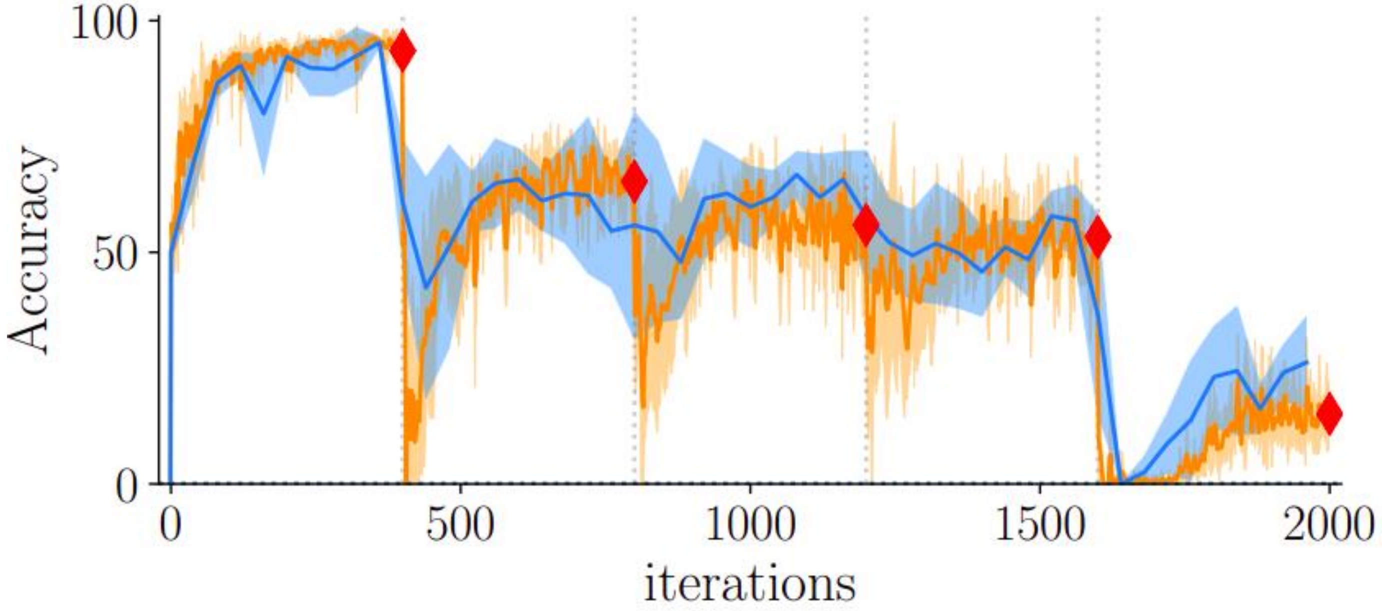
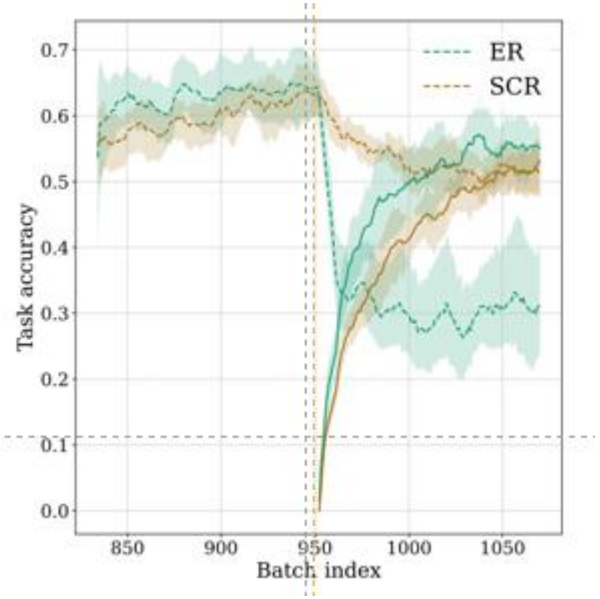

Output Layer (classes)
Class specific discriminative features (training at full pace)
Latent replay layer
External Storage (replay patterns)
Forward Pass (all patterns)
Backward pass (all patterns)
Concat (at mini-batch level)
Low-level generic features (slow training)
Forward Pass (native patterns)
Backward pass (native patterns)
Input Layer (images)

Figure 1: Architectural diagram of Latent Replay.

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)

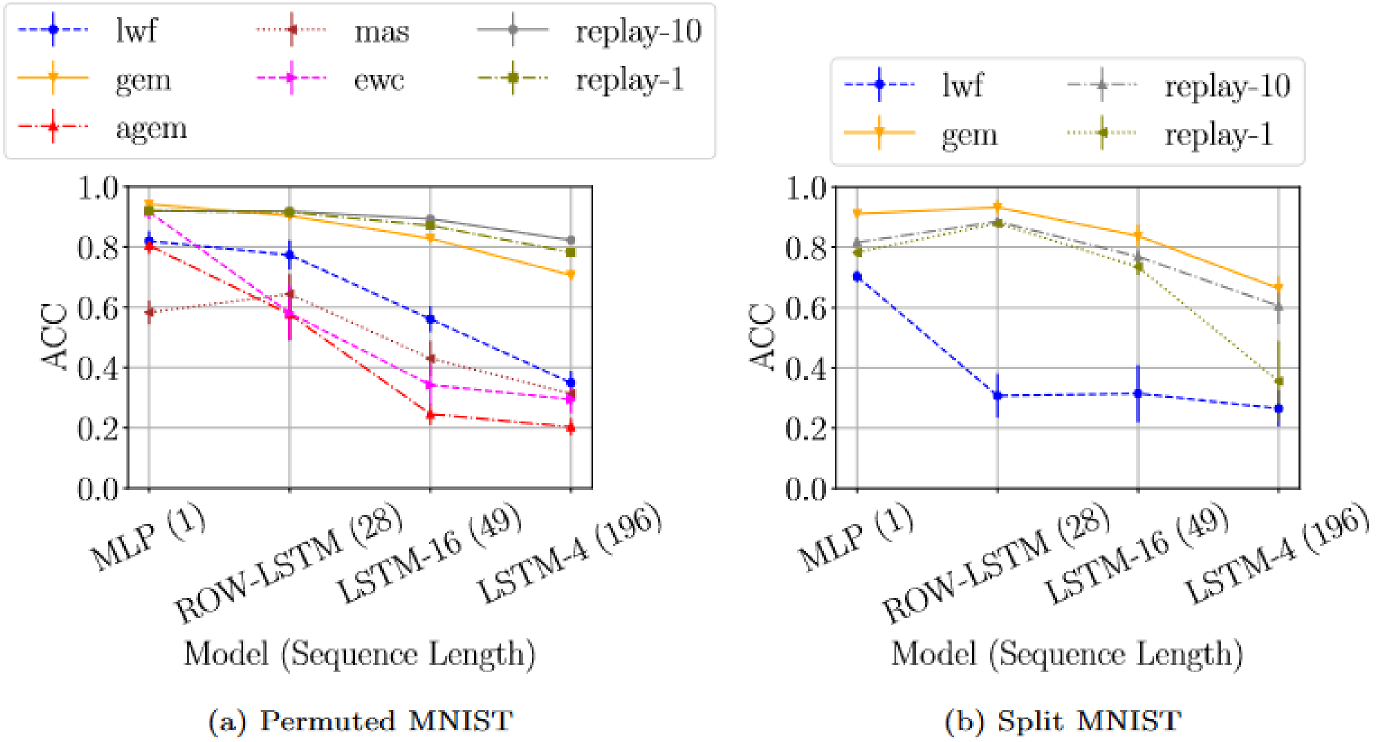




[1] Mathias Delange et. al, Continual Evaluation for Lifelong Learning: Identifying the stability gap, ICLR 2023
[2] Lucas Caccia et. al, New Insights on Reducing Abrupt Representation Change in Online Continual Learning, ICLR 2022
[3] A. Soutif et al. "A Comprehensive Empirical Evaluation on Online Continual Learning" 2023
[4] A. Soutif et al. «Improving Online Continual Learning Performance and Stability with Temporal Ensembles" CoLLAs '23

CL for Time Series – CL with RNNs

- Sequence length increases forgetting
- Replay is still the best method
- Lack of a common benchmark for CL on time series



(a) Permuted MNIST

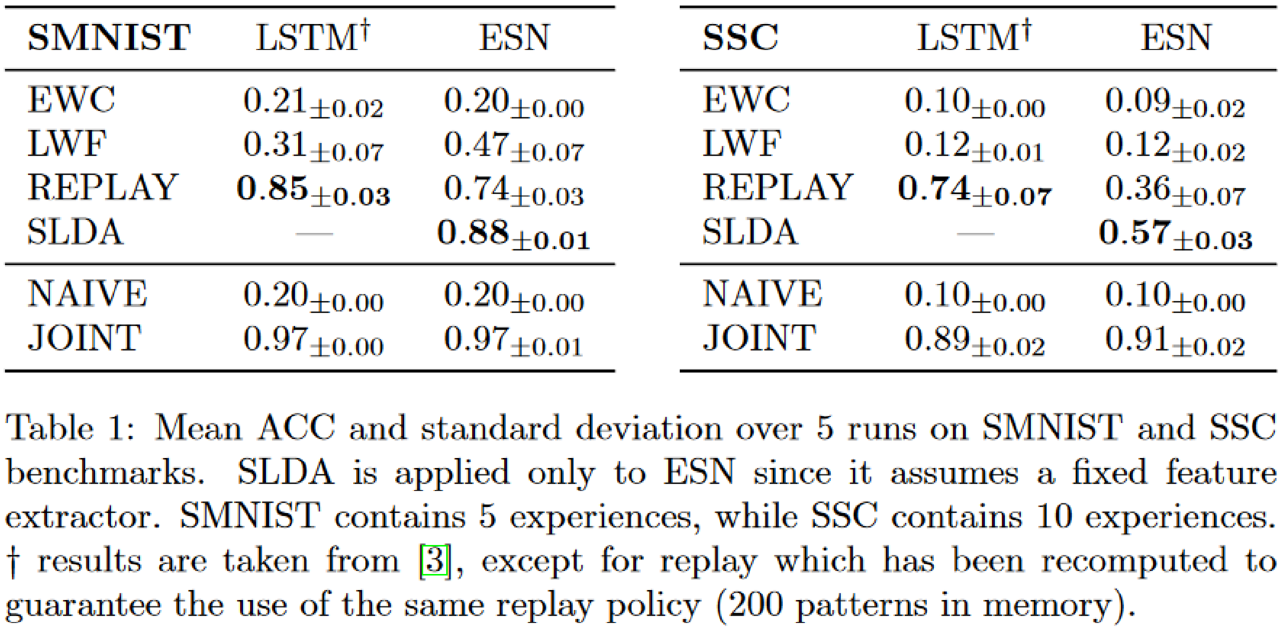
(b) Split MNIST

Figure 6: Average ACC on all steps for different sequence lengths and different CL strategies. Sequence length causes a decrease in performances among all strategies. Best viewed in color.

A. Cossu et al. "Continual Learning for Recurrent Neural Networks: An Empirical Evaluation." *Neural Networks*. http://arxiv.org/abs/2103.07492.

CL for Time Series – CL with ESNs

- Alternative to pretraining for RNNs
- Methods that train only the classifier can be used (SLDA)
- Efficient
- rehersal-free
- Applicable for Online CL



SMNIST	LSTM†	ESN
EWC	$0.21_{\pm 0.02}$	$0.20_{\pm 0.00}$
LWF	$0.31_{\pm 0.07}$	$0.47_{\pm 0.07}$
REPLAY	$\mathbf{0.85}_{\pm \mathbf{0.03}}$	$0.74_{\pm 0.03}$
SLDA	—	$\mathbf{0.88}_{\pm \mathbf{0.01}}$
NAIVE	$0.20_{\pm 0.00}$	$0.20_{\pm 0.00}$
JOINT	$0.97_{\pm 0.00}$	$0.97_{\pm 0.01}$

SSC	LSTM†	ESN
EWC	$0.10_{\pm 0.00}$	$0.09_{\pm 0.02}$
LWF	$0.12_{\pm 0.01}$	$0.12_{\pm 0.02}$
REPLAY	$\mathbf{0.74}_{\pm \mathbf{0.07}}$	$0.36_{\pm 0.07}$
SLDA	—	$\mathbf{0.57}_{\pm \mathbf{0.03}}$
NAIVE	$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$
JOINT	$0.89_{\pm 0.02}$	$0.91_{\pm 0.02}$

Table 1: Mean ACC and standard deviation over 5 runs on SMNIST and SSC benchmarks. SLDA is applied only to ESN since it assumes a fixed feature extractor. SMNIST contains 5 experiences, while SSC contains 10 experiences. † results are taken from [3], except for replay which has been recomputed to guarantee the use of the same replay policy (200 patterns in memory).

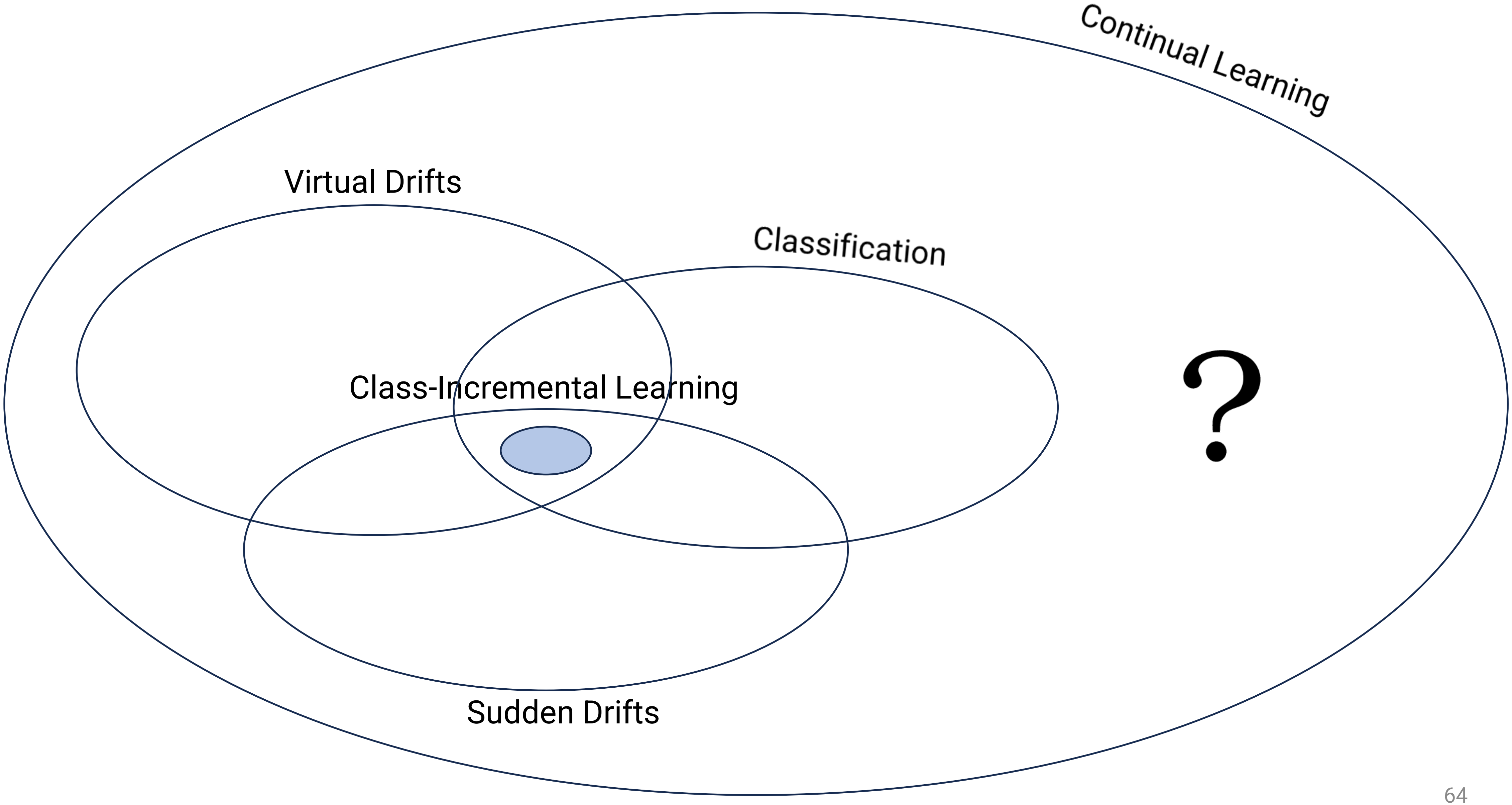
A. Cossu et al. "Continual Learning with Echo State Networks," ESANN 2021. http://arxiv.org/abs/2105.07674.

Open Research Question

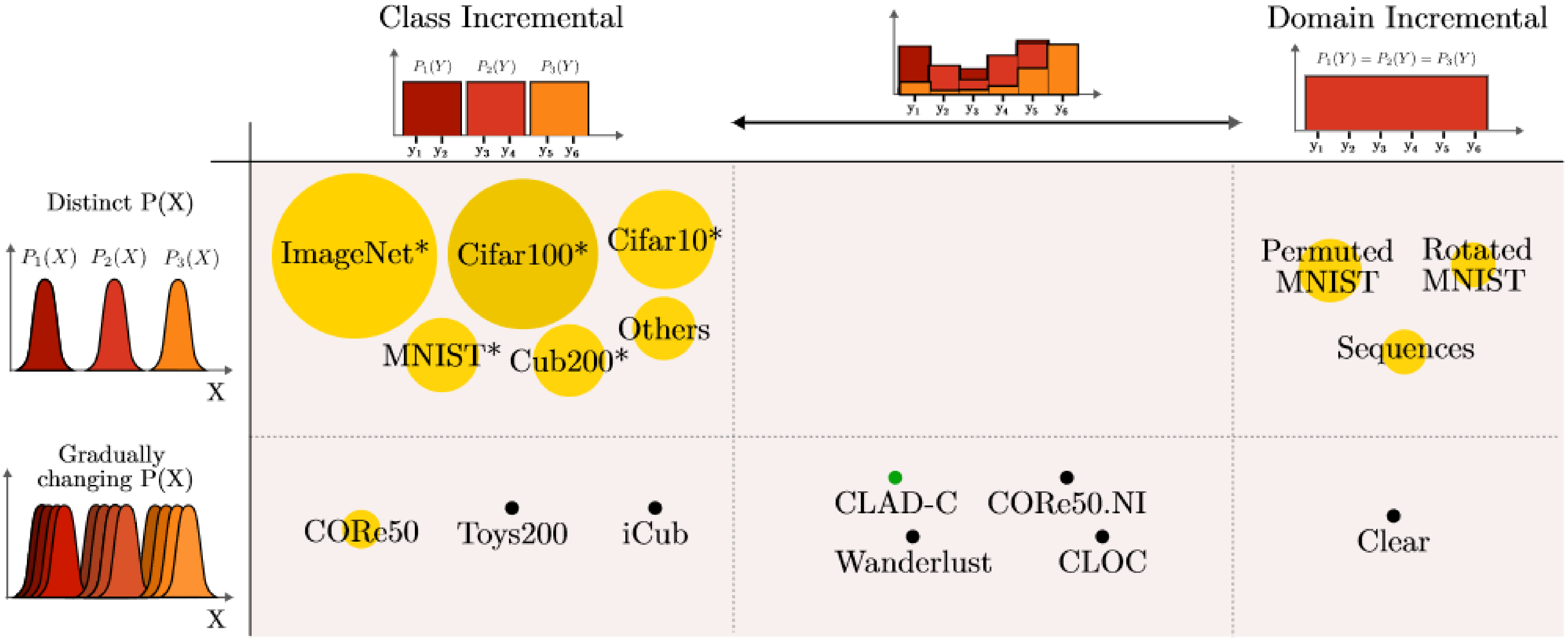
Beyond Class-Incremental Learning

«Is Class-Incremental Enough For Continual Learning?», Cossu et al, Frontiers in CS, 2021

Beyond Class-Incremental Learning



Classic CL Benchmarks



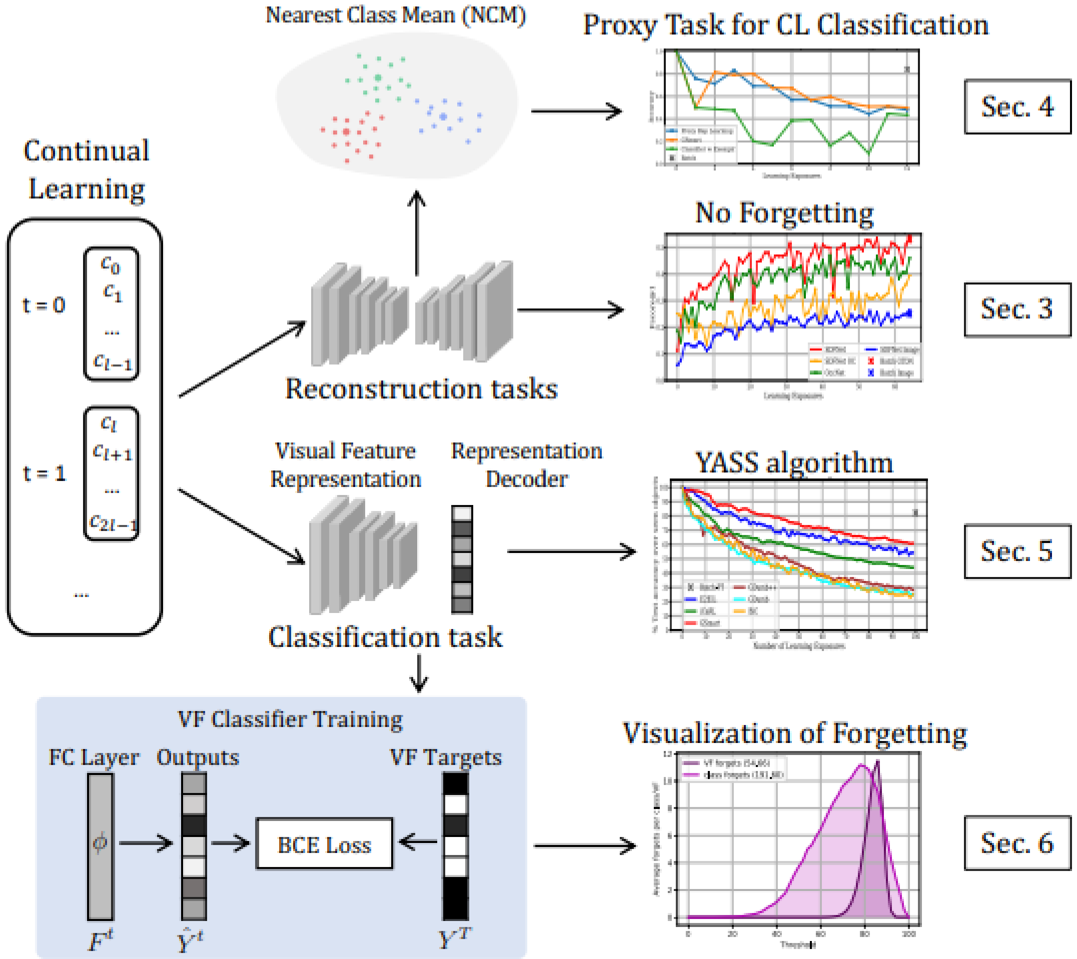
Eli Verwimp et al. 2022. "CLAD: A Realistic Continual Learning Benchmark for Autonomous Driving."

Forgetting in Different Scenarios

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).

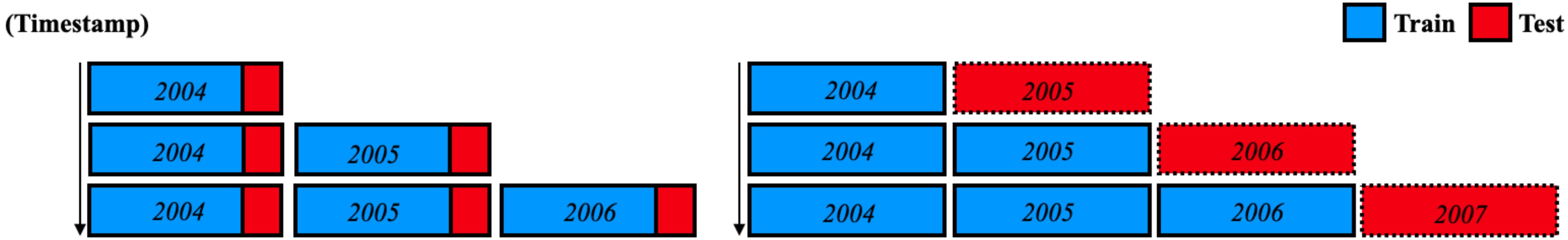
Consequences of Realistic Streams

- **Gradual drifts**: methods can’t easily freeze old components, task/domain inference is more difficult.
- **New domains**: new classes implicitly provide labels, domains don’t.
- **Repetitions**: methods can’t easily freeze old components.
- **Imbalance**: reservoir sampling mimics the unbalance in the stream.
- **Real drift**: Replay data may be incorrect.

Evaluation with Real vs Virtual Drifts

Example: Data ordered by class (0,0,0,0,0,1,1,1,1…)
- Persistent classifier (predict previous class) is optimal
- The model can (and should) exploit temporal consistency!

- **Virtual drift**
  - sampling bias
  - Evaluation on a static test set
  - a.k.a. most of the CL research
- **Real drift**
  - Concept drift. Example: politician roles and affiliations to political party
  - Evaluation on the next data (e.g. **prequential evaluation**)
  - Not a lot of research in CL right now



IID Protocol: *Train today, test on today*

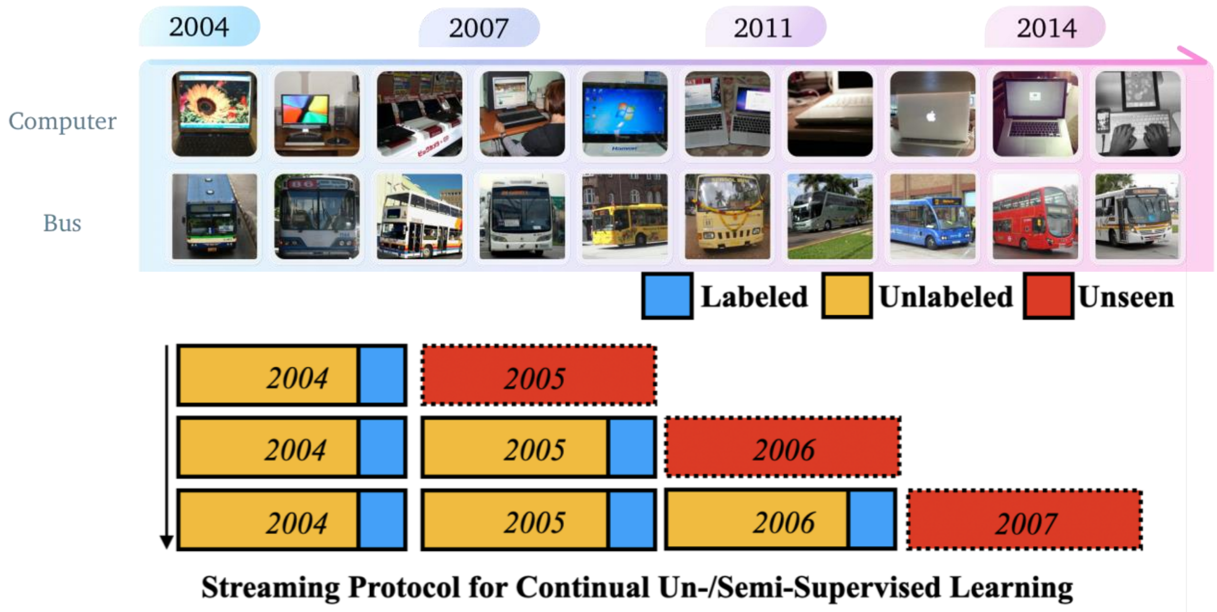
Streaming Protocol: *Train today, test on tomorrow*

Image from CLEAR paper

Real Drift - CLEAR / Wild-Time

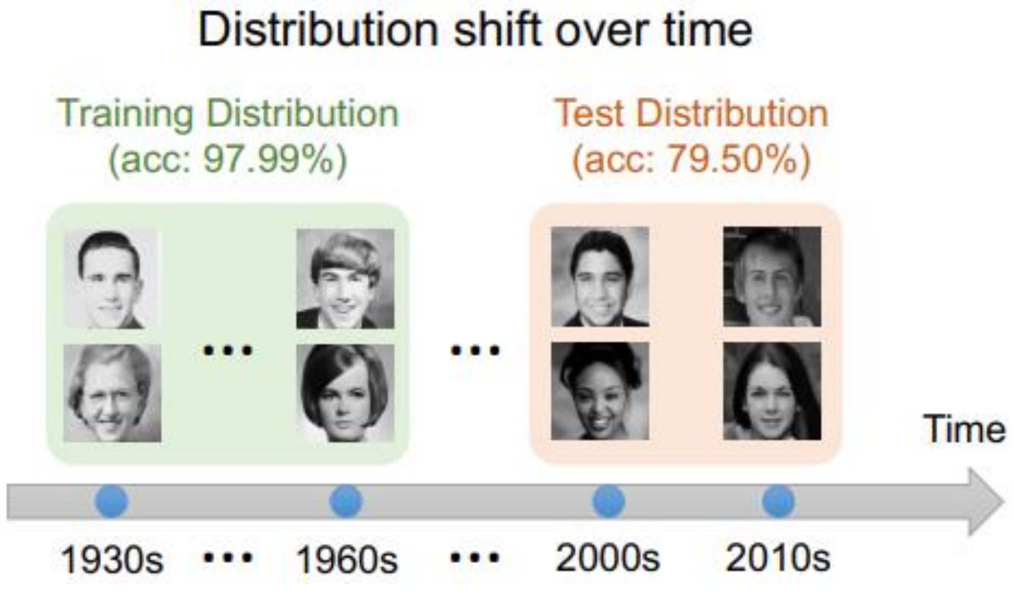
CLEAR

- real-world images with smooth temporal evolution
- Large unlabeled dataset (~7.8M images)
- Prequential evaluation
- Scenario: domain-incremental and semi-supervised



Wild-Time

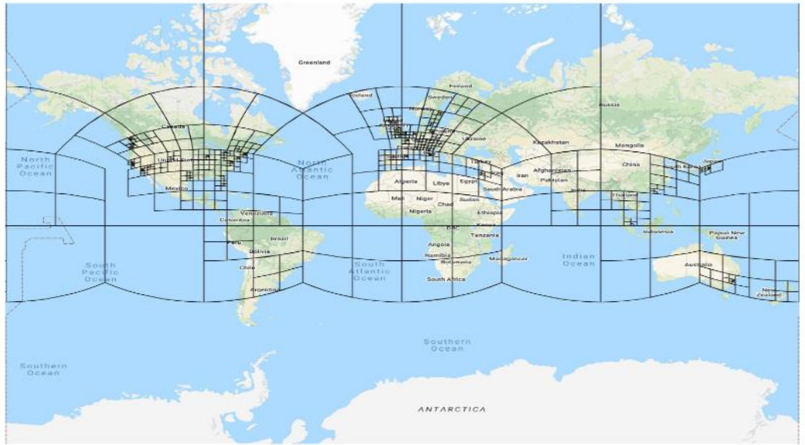
- 5 datasets with temporal distribution drifts (real drift)
- Temporal metadata
- **Eval-Fix**: evaluation on static test data
- **Eval-Stream**: evaluate on the next K timestamps



Z. Lin et al. "The CLEAR Benchmark: Continual LEArning on Real-World Imagery" 2021
Yao, Huaxiu et al. "Wild-Time: A Benchmark of in-the-Wild Distribution Shift over Time." NeurIPS 2022

Real Drift - CLOC – Continual Localization

- Images with geolocalization and timestamps
  - 9 years of data
  - 39M images
  - 2M for offline preprocessing
  - 712 classes (localization regions)



(a) S2 Cells in our dataset

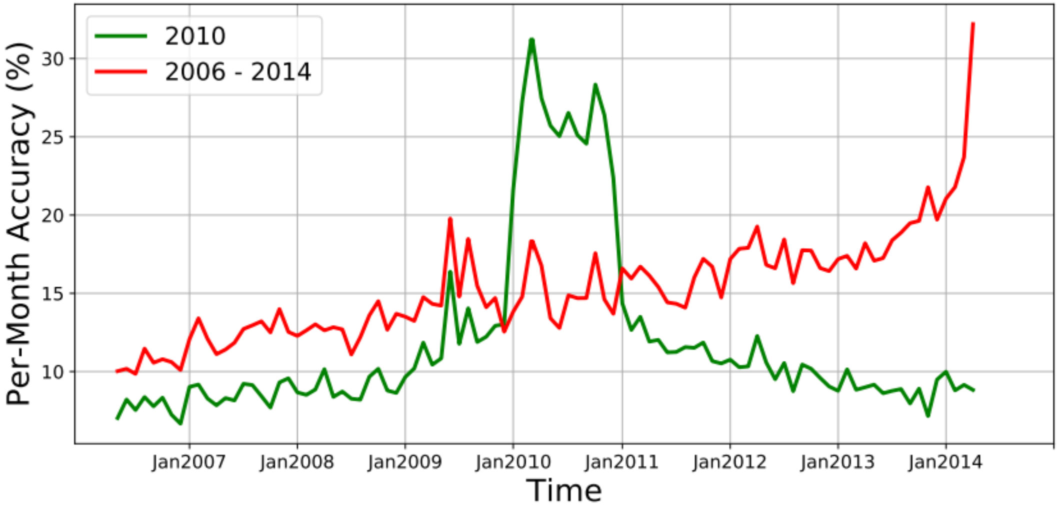


Figure 2. **Distribution shift in CLOC.** We train two supervised models, one using data from the entire temporal range and the other only on data from the year 2010. We evaluate both models on the full temporal range using the validation set (not seen during training). Due to non-stationarity in the data, the performance of the 2010 model drops sharply on data from other times.

Z. Cai et al. "Online Continual Learning with Natural Distribution Shifts: An Empirical Study with Visual Data." ICCV '21

Temporal Coherence - CoRE50

- **Temporally coherent** streams
- Domain-incremental, class-incremental, and repetitions
- **CL on-the-edge application**:
  - Given a pretrained model
  - Take a short video of a new object
  - Finetune the model

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.

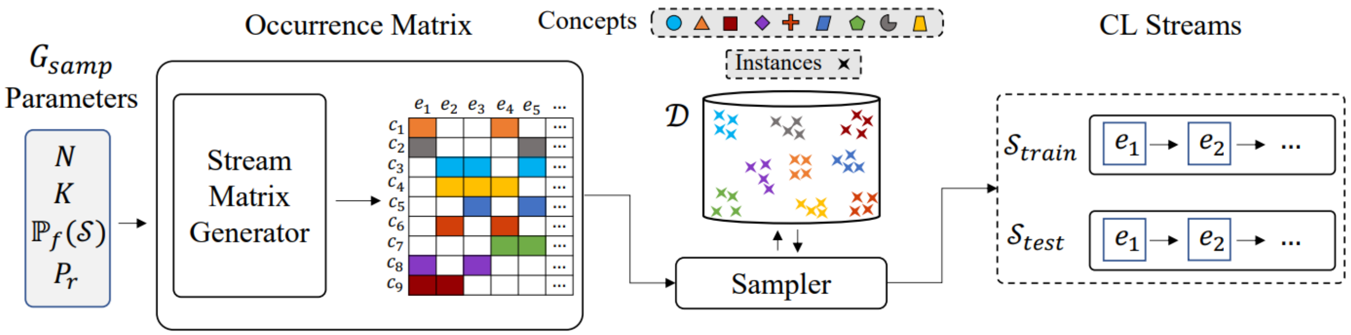
Simulators and Synthetic Data

Driving simulation

- **Parameters**:
  - new classes
  - weather
  - illumination changes
- **Temporal consistency**

CIR Synthetic Generator

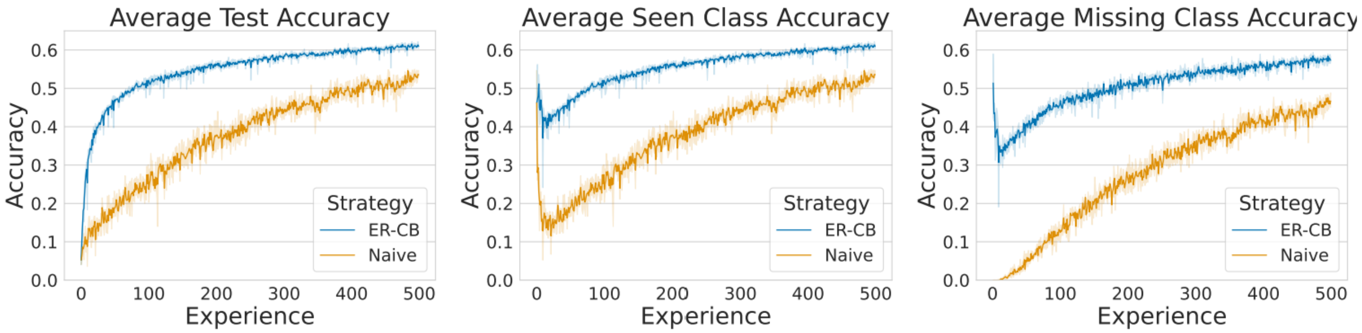
- Start from a static dataset (e.g. CIFAR100)
- Define distribution parameters: stream length, class balancing, repetitions, …
- Sample stream with the desired probability
- You can tweak the difficulty of the benchmark and check how different methods perform under different conditions



T. Hess et al. "A Procedural World Generation Framework for Systematic Evaluation of Continual Learning." 2021
H. Hemati et al. "Class-Incremental Learning with Repetition." CoLLAs '23

Effect of Natural Repetitions

Naive finetuning approaches replay for long streams with repetitions



Missing class accuracy improves over time, even for naive finetuning

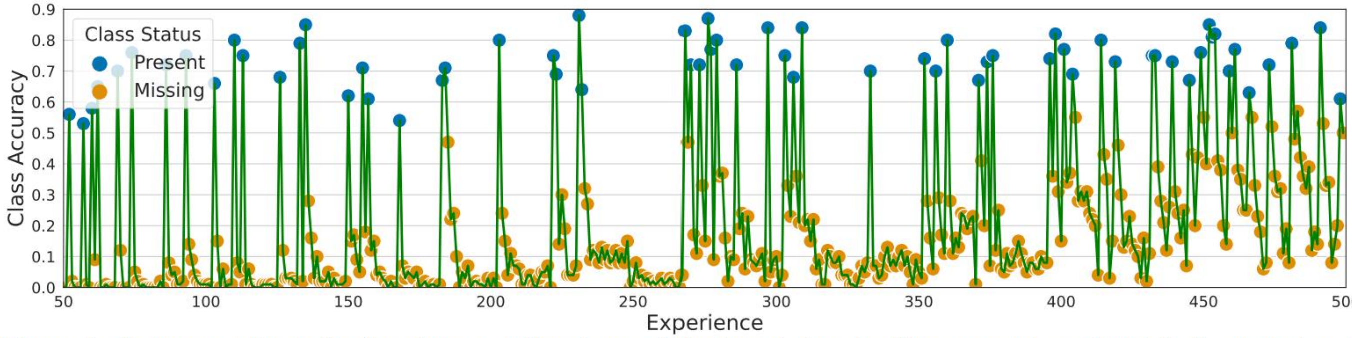


Figure 6: Accuracy of a particular class over the stream. The target class is either present or absent in the experiences indicated by the blue and orange points, respectively.

In unbalanced streams, class-balanced buffers and reservoir sampling are not effective

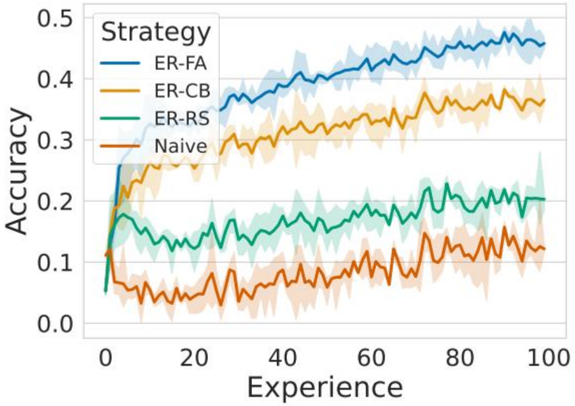


Figure 10: Accuracy of Infrequent Classes.

H. Hemati et al. “Class-Incremental Learning with Repetition.” CoLLAs ‘23

Recap

- **Benchmarks desiderata**: gradual drifts, new domains and classes, repetitions, temporal coherence, real drift
- **Real drifts**: Wild-Time, CLEAR, CLOC. Prequential evaluation for real drifts
- **Streaming data**: CoRE50 (and many others)
- **Simulators and synthetic generators**: allow to control drift and evaluate over many different configurations

Open Research Questions

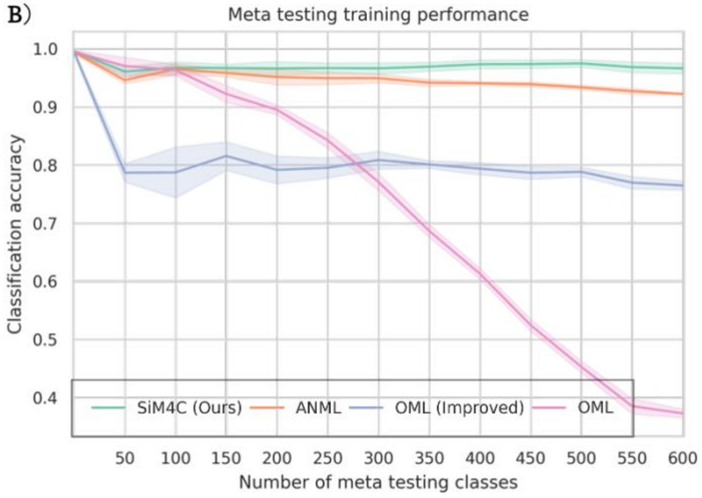
CL Methods Robustness

Real-time CL

Robust CL Methods

- **Most methods cheat during hyperparameter selection by optimizing over the whole stream!**
- **Alternative to Continual Hyperparameter Selection**: design robust models!
- **Example: SiM4C**
  - Efficient meta-continual learning
  - Use a single inner update step
  - Use exact gradient instead of first-order approximation
- **Results**:
  - Higher accuracy
  - No need for additional hyperparameter selection
  - Easy to plug into existing methods
  - Works in continual-meta and meta-continual learning

Omniglot



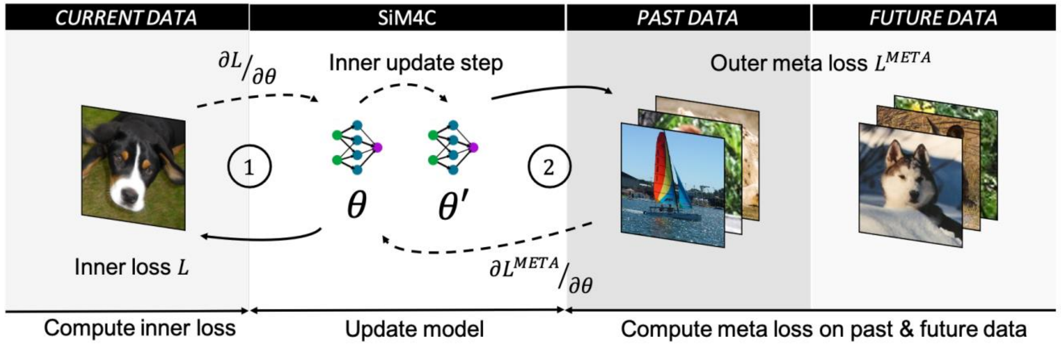
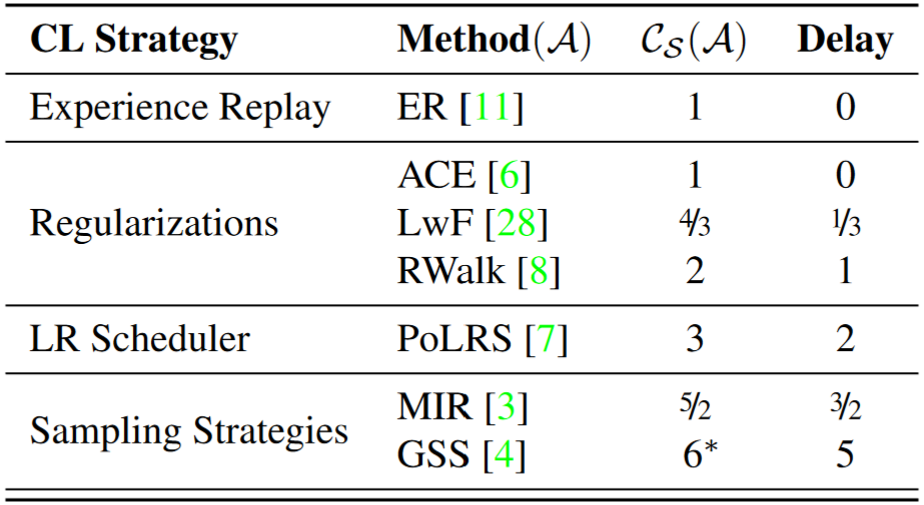

CURRENT DATA
SiM4C
PAST DATA
FUTURE DATA
$\partial L / \partial \theta$
Inner update step
Outer meta loss L^{META}
1
2
θ
θ'
Inner loss L
$\partial L^{META} / \partial \theta$
Compute inner loss
Update model
Compute meta loss on past & future data

Figure 1. Schematic depiction of SiM4C, after a single inner optimization step the proposed meta-objective optimizes for forward and backward transfer by utilizing seen *past data* from previous tasks and unseen *future data* of the current task.

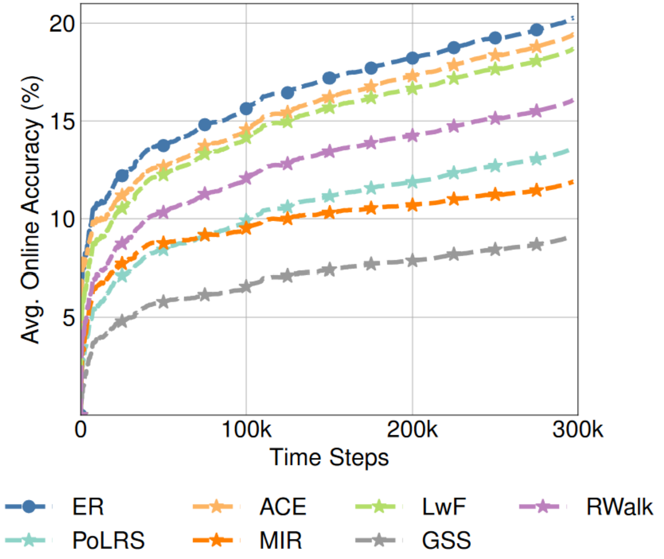
E. Cetin et al. «A Simple Recipe to Meta-Learn Forward and Backward Transfer ", ICCV '23

Real-Time / Infinite Memory / Finite Compute

- **Memory is cheap, compute is expensive**
  - CL methods are designed for finite memory usage. Often unrealistic
  - The "privacy argument" is not very strong, because trained models can leak data
- **Alternative: real-time, infinite memory, bounded computational cost**
  - Real-time constraints. Methods need to skip data if they are not fast enough
- **Results: Experience Replay outperforms CL methods**



CL Strategy	Method$(\mathcal{A})$	$\mathcal{C}_{\mathcal{S}}(\mathcal{A})$	Delay
Experience Replay	ER [11]	1	0
Regularizations	ACE [6]	1	0
	LwF [28]	$4/3$	$1/3$
	RWalk [8]	2	1
LR Scheduler	PoLRS [7]	3	2
Sampling Strategies	MIR [3]	$5/2$	$3/2$
	GSS [4]	6*	5



Y. Ghunaim et al. "Real-Time Evaluation in Online Continual Learning: A New Hope." CVPR '23
A. Prabhu et al. "Computationally Budgeted Continual Learning: What Does Matter?" CVPR '23

Conclusion

Main Message

The goal of Continual Learning is to understand how to design machine learning models that learn over time

- on a constrained budget (memory/compute/real-time requirements)
- with non-stationary data

- Wide applications to real problems and foundational ML problems
  - Learning on-the-edge
  - Fast adaptation, knowledge accumulation and lifelong learning

Open Challenges

- **CL Benchmarks**:
  - Real drifts and prequential evaluation
  - Exploitation of temporal coherence
  - Real-time training with infinite memory
  - Compute-bounded lifelong learning
- **CL Robustness to**
  - stream parameters
  - (continual) hyperparameter selection
  - stability gap