



**POLITECNICO**  
MILANO 1863

# Continual Learning

Federico Giannini

Ph.D. Student  
DEIB – Politecnico di Milano  
[federico.giannini@polimi.it](mailto:federico.giannini@polimi.it)

## QUESTION:

Which are the two main problems that Streaming Machine Learning must manage?



- ▶ The Traditional ML approach is unrealistic:
  - ▶ It supposes that the knowledge is completely represented by a bounded training set.
  - ▶ It supposes that data will not change its distribution over time.
- ▶ In the real world, data:
  - ▶ Becomes available over time
  - ▶ Can suffer changes in its distribution.
- ▶ We cannot train a model once on a data set and think it will remain appropriate forever.
- ▶ We must **continuously** train our model to keep it updated with new knowledge and data changes.

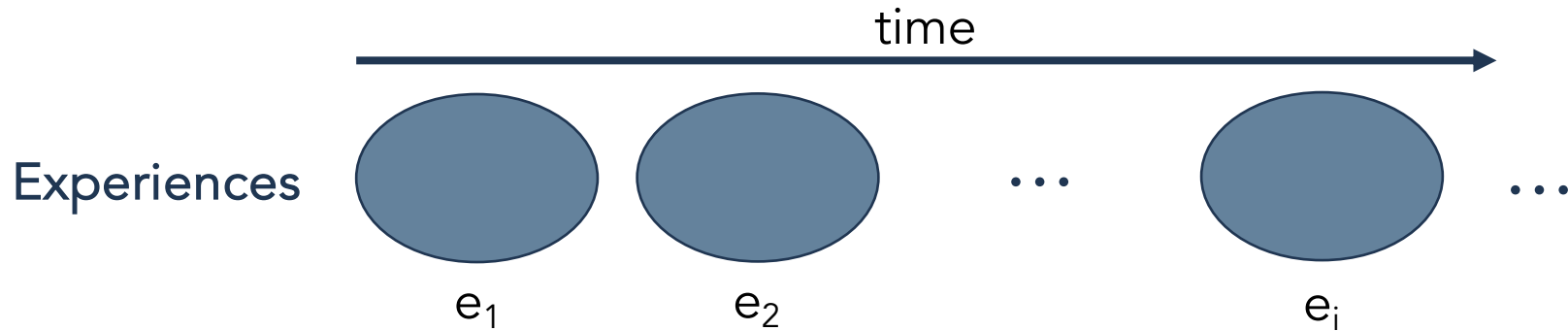


- ▶ Similarly to SML, the CL goal is to improve the model based on experience and keep it updated to changes.
- ▶ It usually applies Deep Learning techniques.
- ▶ But, as we will see, it focuses on remembering the past knowledge when learning new one.

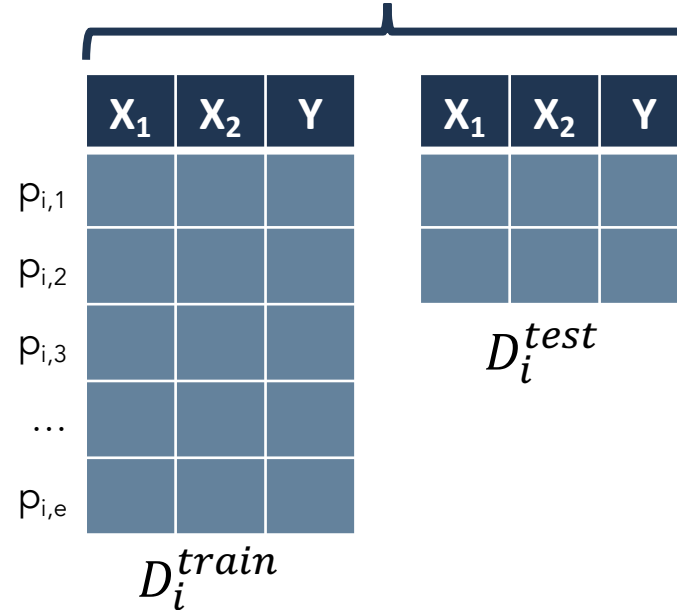


# Problem Setting

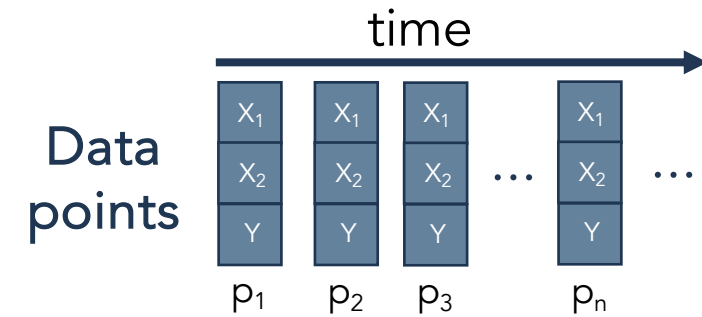
## CONTINUAL LEARNING



- ▶ A datastream  $S$  is split into **experiences**.
- ▶ Each experience  $e_i$  consists of a **large batch of samples**  $D_i$ .
- ▶  $D_i$  is usually split into  $D_i^{train}$  and  $D_i^{test}$ .
- ▶ Each experience is available at once and the model can be trained on it for several epochs.
- ▶ Unbounded time between two experiences.



## STREAMING ML



- ▶ Single data points or mini-batches arrive over time.
- ▶ No distinction between train and test (Prequential evaluation).
- ▶ The model must be able to provide online answers at any time (only one pass, low time/memory)



$$A^{CL}: (f_{i-1}^{CL}, M_{i-1}, D_i^{train}, t_i) \rightarrow (f_i^{CL}, M_i)$$

- ▶  $f_i^{CL}$ : model learned after the  $i$ -th experience.
- ▶  $M_i$ : knowledge until the  $i$ -th experience.
- ▶  $t_i$ : **task** (optional). It could be seen as a time interval in which the data distribution and the objective function may be fixed (similar to SML's concept).

CL goal is to optimize the loss computed on the different  $D_i^{test}$  **over the entire data stream**.

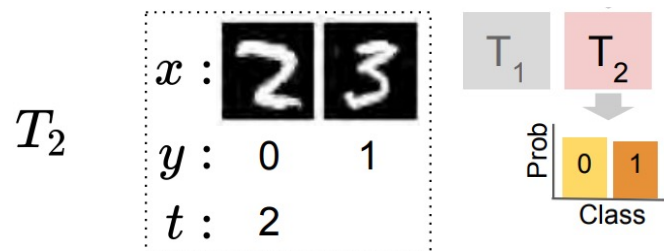
A CL strategy must perform well on the **current experience** and all the **experience seen so far**.

# Scenarios

A new experience usually introduces a change. (Each experience is supposed to be i.i.d.)

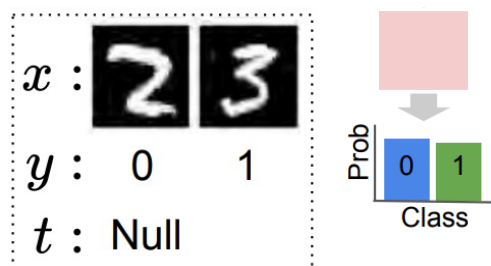
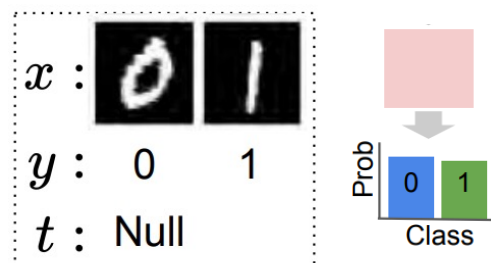
→ No need to apply change detectors during training.

## Incremental Task Learning



Task labels during train & test.

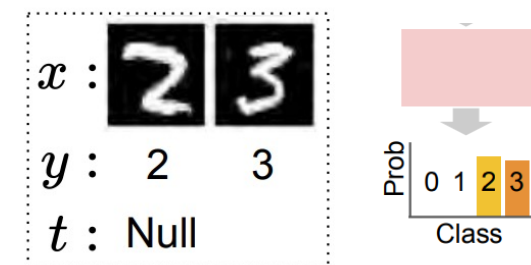
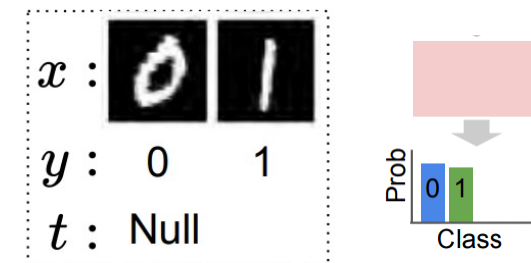
## Incremental Domain Learning



No task labels.

Each experience has its own class labels.

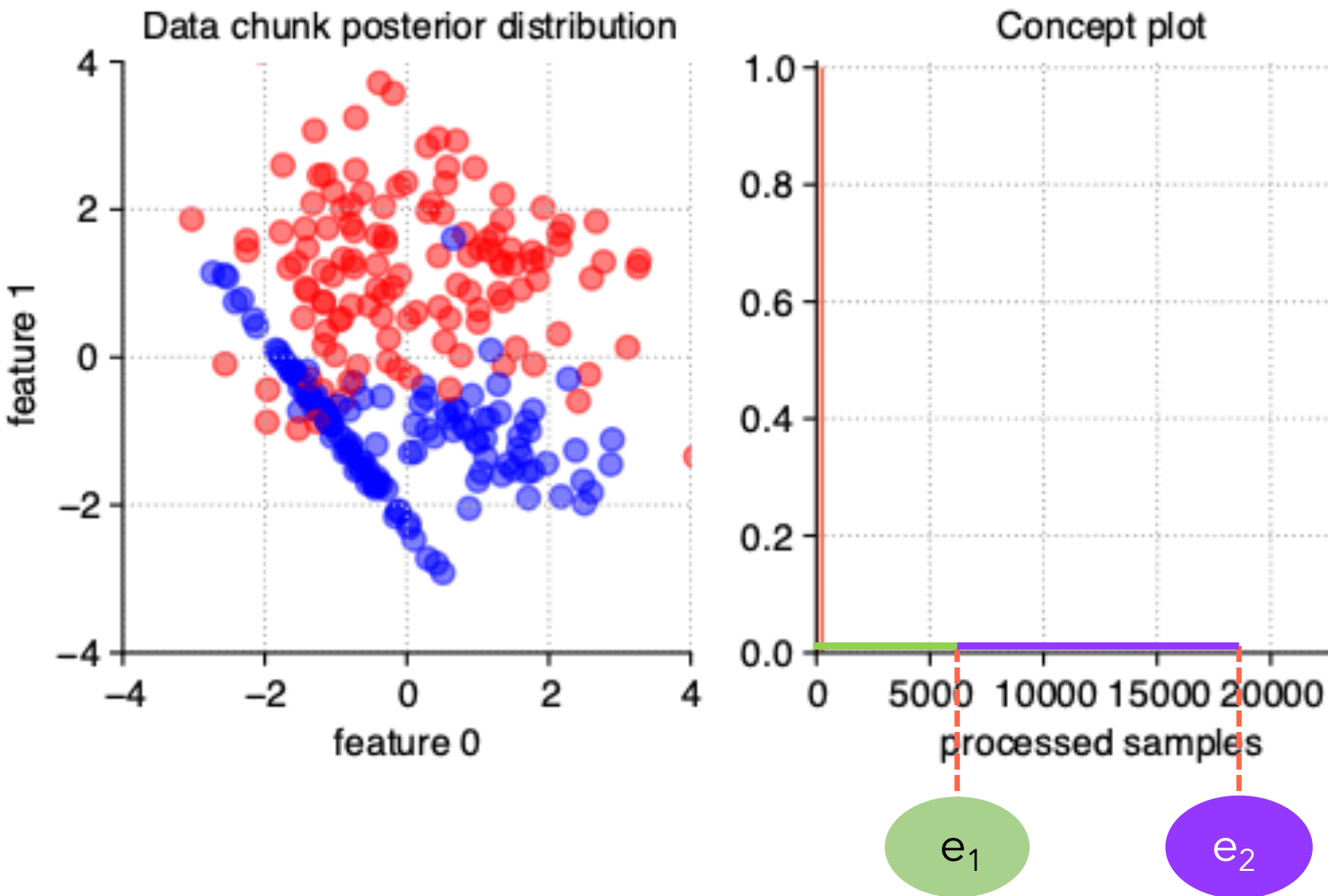
## Incremental Class Learning



No task label

(different class labels during train)

# What is a CL experience?



We can imagine to **accumulate** the data points of each concept in a **batch** (experience).

When the concept finishes (an abrupt concept drift arises) we input the experience to the CL model.

Each new experience can introduce:

- New classes (**Class Incremental**)
- New Instances of the same classes with a different distribution (**Domain Incremental**).

We have the label of the distribution in **Task Incremental**.



# Stability-plasticity Dilemma – 1/4

A model should achieve a trade-off between stability and plasticity.

- ▶ **Plasticity:** ability to acquire new knowledge and keep learning over time.

Too much plasticity → forgetting past knowledge.

- ▶ **Stability:** ability to remember the past acquired knowledge over time.

Too much **stability** → difficulties in learning new knowledge.

CL focuses on the following:

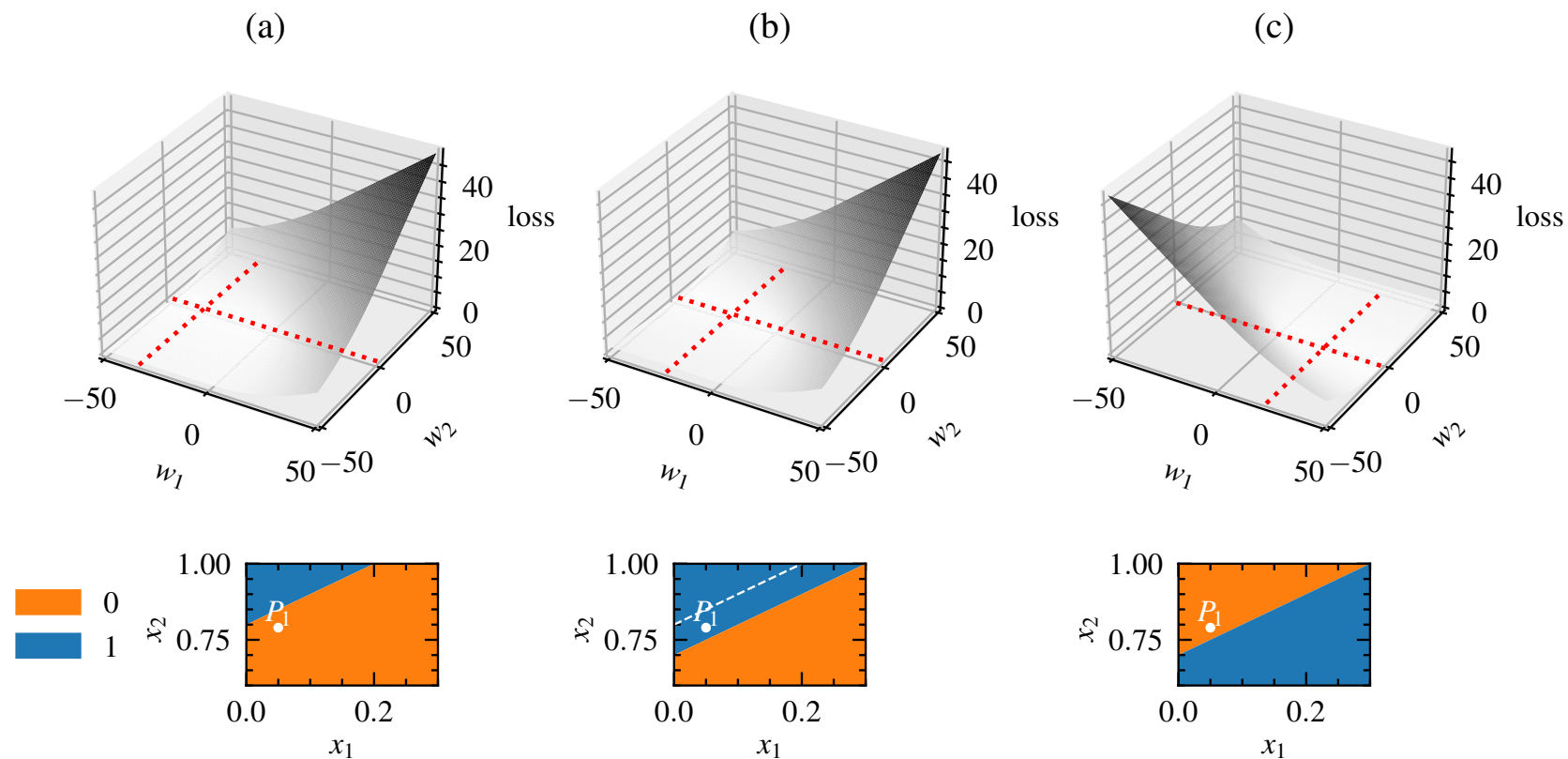
- ▶ Learning new knowledge.
- ▶ Remembering past knowledge.
- ▶ Generalizing on future instances.

Evaluation strategies and different methodologies are associated with these goals.



# Stability-plasticity Dilemma – 2/4: Catastrophic Forgetting (CF)

- ▶ CF is the problem associated with too much plasticity.
- ▶ NN models can easily suffer from forgetting past knowledge because of the SGD algorithm.
- ▶ When there's a drift in the objective function, the model's parameters are set to reach the new loss function's minimum.
- ▶ Without a specific strategy, the previous task is forgotten.

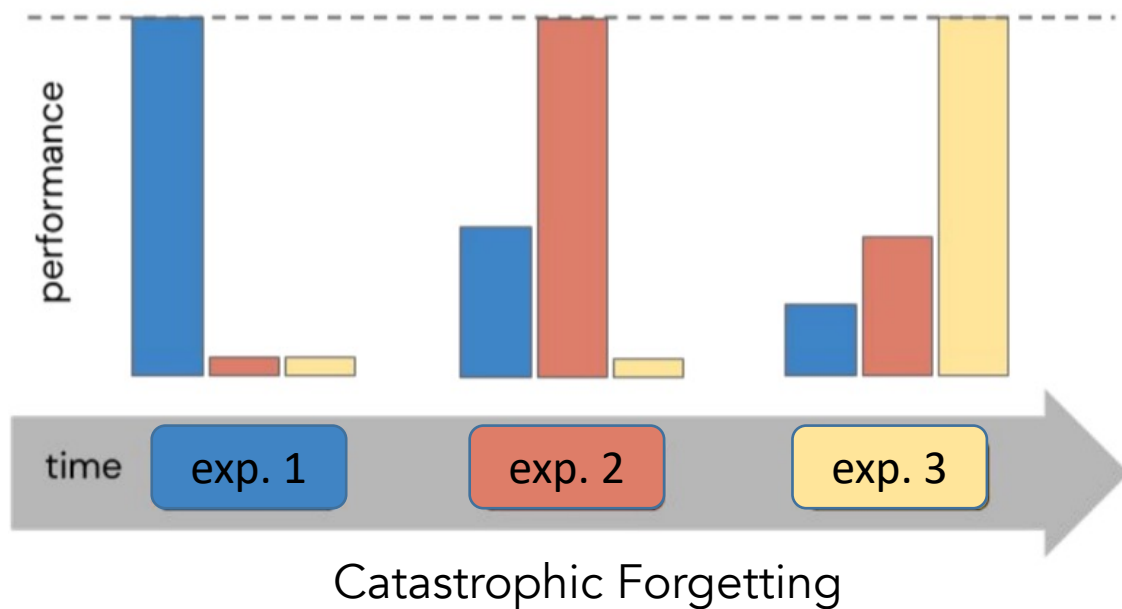


*(Incremental task scenario)*

# Stability-plasticity Dilemma 3/4

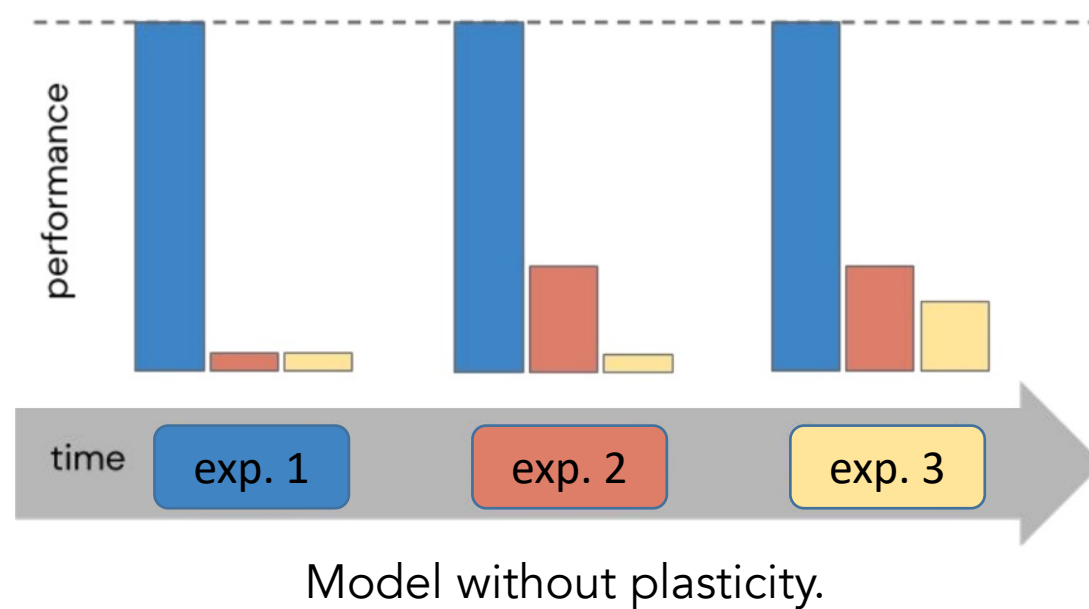
## Avoiding CF

Training on new experiences should not reduce performance on previously learned experiences.



## Maintaining Plasticity

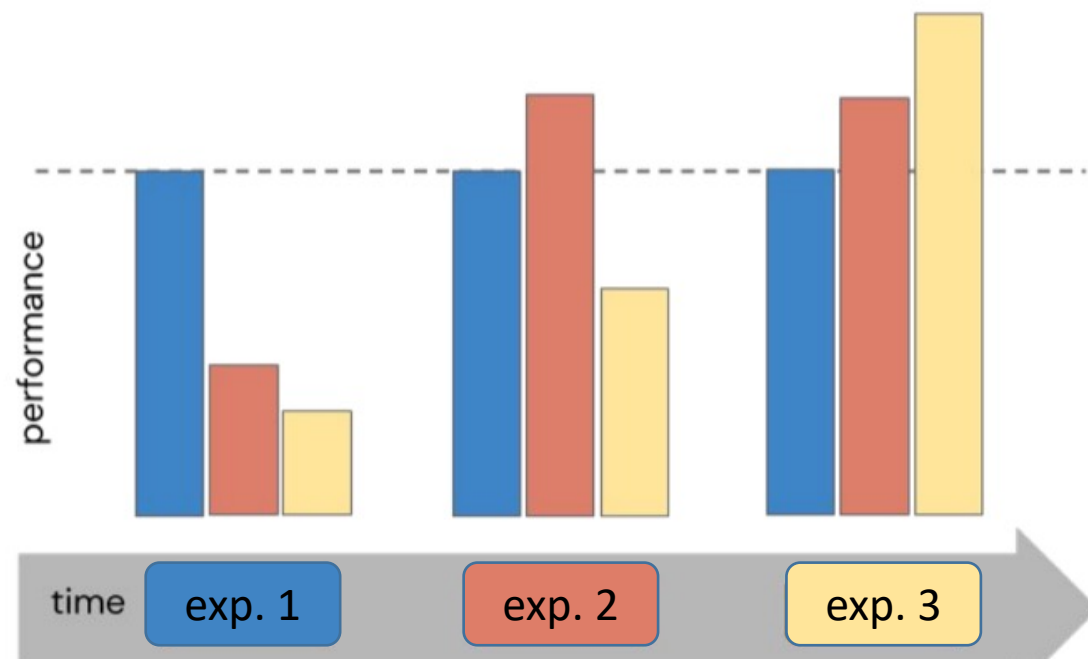
The model should be able to keep learning as new experiences are observed.



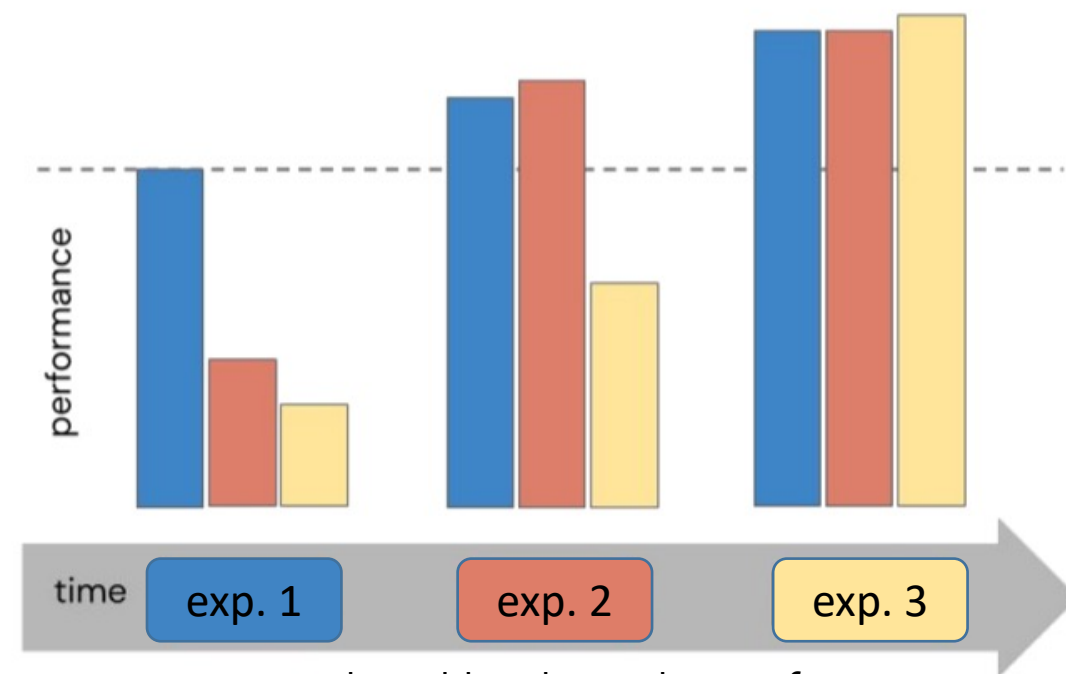
# Stability-plasticity Dilemma 4/4

## Maximizing Forward and Backward transfer

Learning an experience should improve both past and future experiences.

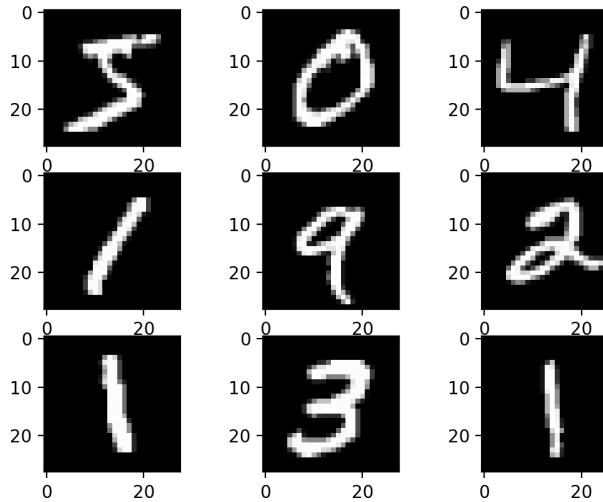


Forward transfer and no catastrophic forgetting



Forward and backward transfer

- ▶ Most CL applications are still on toy datasets (computer vision like MNIST, CIFAR) and supervised problems.
- ▶ The most studied scenario in the past was Incremental Task Learning with few big tasks.
- ▶ Currently, the focus is on Incremental Class Learning with dozens of experiences.



# Evaluation: Average Accuracy and A metric

- ▶ Usually, each experience is split into **train** and **test** sets.
- ▶ After each experience's training, we compute the metrics on **all** the experiences' test sets (we suppose to have all the test sets available).

	Test e <sub>1</sub>	Test e <sub>2</sub>	Test e <sub>3</sub>
Train e <sub>1</sub>	R <sub>1,1</sub>	R <sub>1,2</sub>	R <sub>1,3</sub>
Train e <sub>2</sub>	R <sub>2,1</sub>	R <sub>2,2</sub>	R <sub>2,3</sub>
Train e <sub>3</sub>	R <sub>3,1</sub>	R <sub>3,2</sub>	R <sub>3,3</sub>

- ▶ **Average accuracy:**  $ACC = \frac{1}{T} \sum_{i=1}^T R_{T,i}$

Mean of final performances on all the experiences' test sets.

- ▶ **A metric:**  $A = \frac{\sum_{i=1}^N \sum_{j=1}^i R_{i,j}}{\frac{N(N+1)}{2}}$

For each experience, we consider the current and the previous ones.

	Test e <sub>1</sub>	Test e <sub>2</sub>	Test e <sub>3</sub>
Train e <sub>1</sub>	R <sub>1,1</sub>	R <sub>1,2</sub>	R <sub>1,3</sub>
Train e <sub>2</sub>	R <sub>2,1</sub>	R <sub>2,2</sub>	R <sub>2,3</sub>
Train e <sub>3</sub>	R <sub>3,1</sub>	R <sub>3,2</sub>	R <sub>3,3</sub>

To measure the impact of the current experience's training on future/previous experiences, we introduce two other metrics:

- **Forward Transfer Metric:**  $\frac{1}{T-1} \sum_{i=2}^T R_{i-1,i} - \bar{b}_i$

It measures how the current training improves performance for the next experience.

- **Backward Transfer Metric:**  $\frac{1}{T-1} \sum_{i=1}^{T-1} R_{T,i} - R_{i,i}$

It measures how the final model has improved or decreased the performances on the previous experiences.

**A negative value indicates forgetting!**

# Evaluation: FWT and BWT example

	Test $e_1$	Test $e_2$	Test $e_3$
Train $e_1$	0.7	0.65	0.6
Train $e_2$	0.5	0.8	0.65
Train $e_3$	0.4	0.3	0.9

Negative Backward Transfer

	Test $e_1$	Test $e_2$	Test $e_3$
Train $e_1$	0.7	0.05	0.02
Train $e_2$	0.75	0.8	0.07
Train $e_3$	0.8	0.85	0.9

Negative Forward Transfer

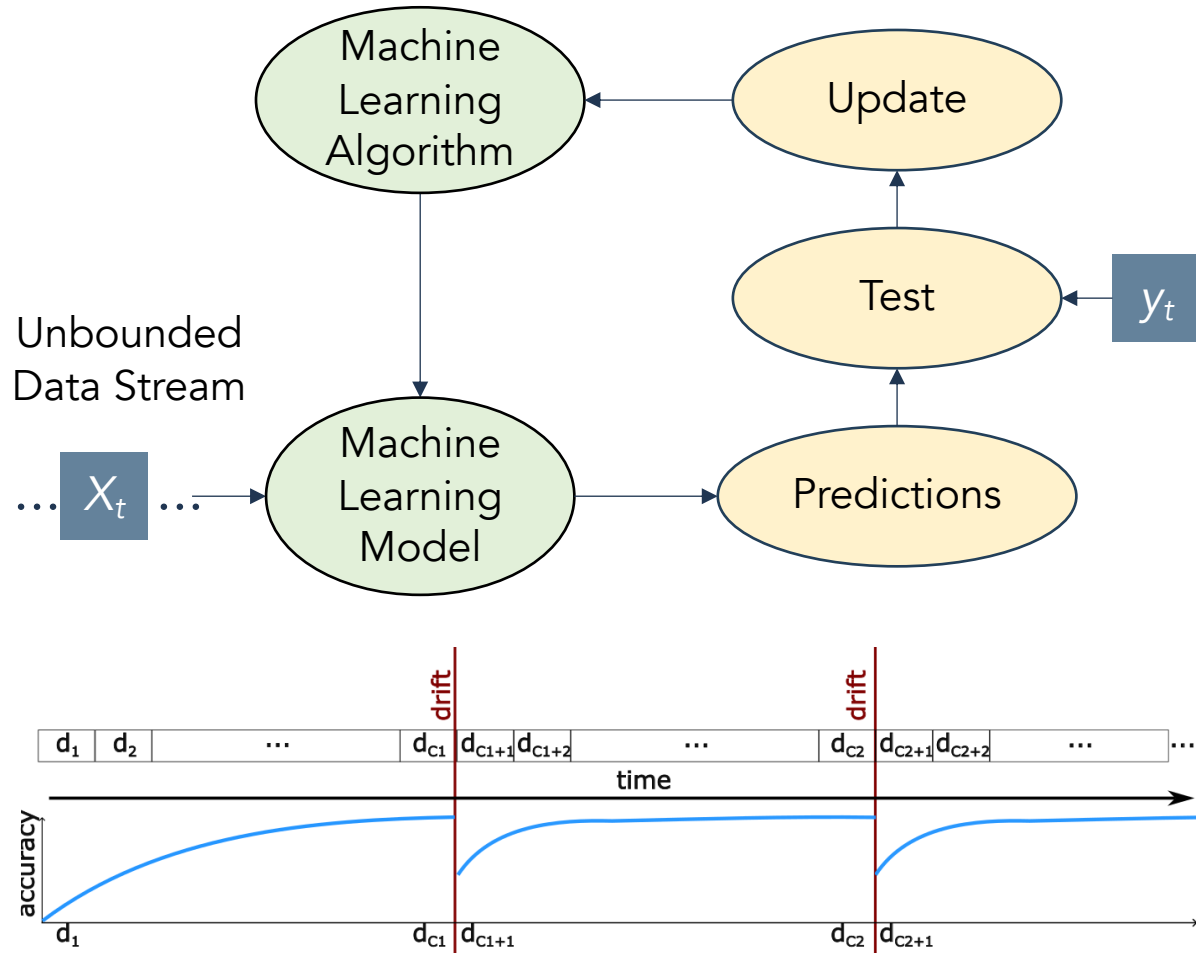
	Test $e_1$	Test $e_2$	Test $e_3$
Train $e_1$	0.7	0.65	0.6
Train $e_2$	0.75	0.8	0.68
Train $e_3$	0.8	0.85	0.9

Positive Backward and Forward Transfer

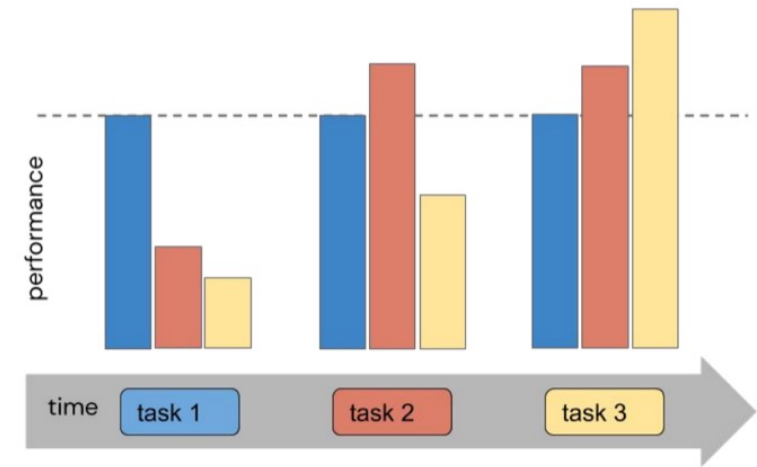


# Evaluation: SML vs CL

## Streaming Machine Learning



## Continual Learning



	Test $e_1$	Test $e_2$	Test $e_3$
Train $e_1$	$Acc_{1,1}$	$Acc_{1,2}$	$Acc_{1,3}$
Train $e_2$	$Acc_{2,1}$	$Acc_{2,2}$	$Acc_{2,3}$
Train $e_3$	$Acc_{3,1}$	$Acc_{3,2}$	$Acc_{3,3}$

# Evaluation: Other problems

Performance is not enough!

Metrics on **memory occupation** and **computational overhead** have been introduced.

**Hyperparameters tuning** and **model selection** are still open issues in CL.

- ▶ Use a first set of experiences.

After having selected the best configurations, the selected model is reset and a new training starts from the following experience.

- ▶ It does not learn the first experiences.
- ▶ Hyperparameters tuned and selected model on the first experiences could not fit the followings.
- ▶ Model selection and hyperparameters tuning on the same stream and test sets of the evaluation (violation of Statistical Learning Theory principles).



- ▶ CL aims to learn new tasks without losing the prediction ability on old ones.
- ▶ The goal is to minimize the loss on all the test sets.
- ▶ A CL strategy is applied on top of a **base learner**.
- ▶ Each Neural Network model can be used as a base learner.
- ▶ In this lectures we will focus on the logic of the strategies.

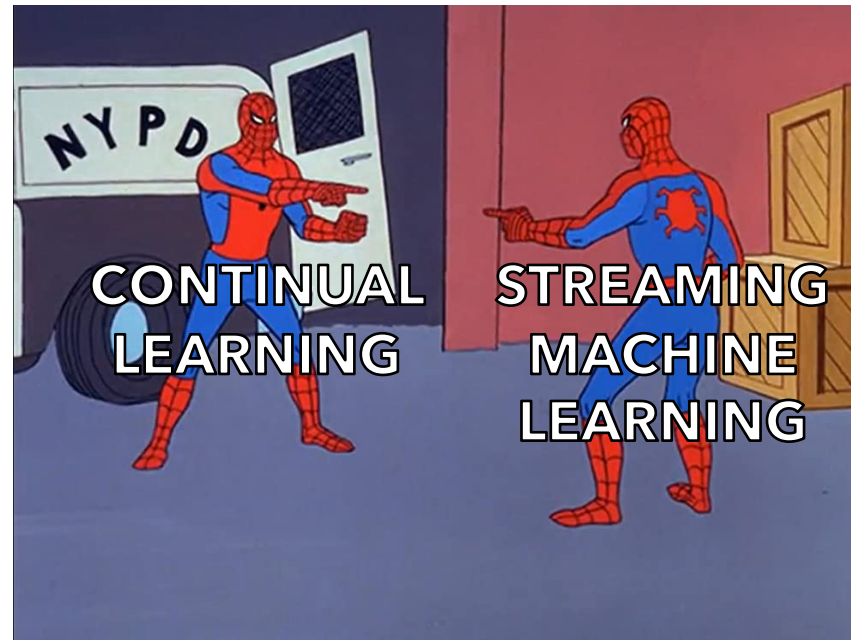


- ▶ **Replay**: during each experience, they mix old examples with new ones to not forget old tasks. Generative replay strategies generate examples associated with old tasks. The training set's size increases with the number of tasks.
- ▶ **Regularization**: they change the loss function by adding regularization terms to the new task's problem and not losing the old ones.
- ▶ **Architectural**: they introduce new parameters to the network (e.g. shared layers between all tasks and tasks specific layers, transfer learning or weight masks). For some strategies, the number of parameters could increase with the number of tasks. Usually they require the task labels.

- ▶ CL focuses on avoiding catastrophic forgetting, while SML is more focused on quick adaptation on new concepts and providing online answers at any time (one concept at a time).
- ▶ CL has a distinction between Train and Test set, while SML usually applies Prequential Evaluation.
- ▶ The presented CL scenarios are not enough and do not represent the real world.
- ▶ CL constraints on problem settings are unrealistic.
- ▶ CL strategies are an interesting way to manage multiple concepts/tasks learning and could be applied on SML scenarios.

- ▶ Future CL works will have to propose solutions for:
  - ▶ New scenarios: Re-occurring of already seen classes / tasks + Mixed experiences
  - ▶ Manage of non-i.i.d. experiences.
  - ▶ Applications on real dataset / Unsupervised problems.
  - ▶ Alternatives to SGD.
- ▶ A new paradigm of CL is arising: **Online Continual Learning**. It reduces the size of the experiences (mini-batches of 10 data points), a more similar approach to SML.
- ▶ SML and CL communities are trying to co-operate and mix methodologies. In this direction, the CL community is applying SML concept drift detectors to CL scenarios.

# Thank you!



The presented material is inspired by Continual Learning Course.

See more at: [Continual AI site](#)