Online Continual Learning

With CapyMOA

Anton Lee
Victoria University of Wellington

June 9, 2025

1897 VICTORIA UNIVERSITY OF WELLINGTON TE HERENGA WAKA NEW ZEALAND

PAKDD 2025
Sydney 10-13 June

https://github.com/adaptive-machine-learning/PAKDD2025

Anton Lee (me):

Anton Lee is a PhD student in AI and a research assistant at the University of Wellington, where they study continual learning. As a research assistant, they maintain the CapyMOA open-source data-stream learning project.

Heitor Murilo Gomes

Senior lecturer at the Victoria University of Wellington (VuW) in New Zealand. Leads the CapyMOA open source library for data stream learning, and provide support for MOA (Massive Online Analysis). `https://heitorgomes.com/`

Nuwan Gunasekara

Nuwan Gunasekara research interests include stream learning, online continual learning, and online streaming continual learning.

Online Continual Learning

Combines online learning and continual learning into a single framework inheriting properties from both.

1. Online Learning

- 1.1 Online vs Batch
- 1.2 Challenges

2. Continual Learning

- 2.1 Tasks and Concepts
- 2.2 Challenges

3. Online Continual Learning

- 3.1 Challenges
- 3.2 Related Paradigms

4. Evaluation

5. Strategies

- 5.1 Replay
- 5.2 Regularisation
- 5.3 Prototype
- 5.4 Others

6. Conclusion

Online Learning

Online Learning

Online Machine Learning (also known as data stream learning) focuses on developing machine learning models capable of performing **inference at any time**, learning from data streams, and adapting to changing and new concepts.

Data Streams

A data stream is a **sequence of examples**, possibly infinite, with a **temporal order**.

- Online learning is machine learning for **data streams**—as opposed to batch learning for datasets.
- Data examples arrive one by one or in mini-batches, and we want to **build and maintain models**, such as patterns or predictors, of these examples in real time (or near real time).
- The underlying relationships may **change** over time.

"No man ever steps in the same river twice." (Heraclitus)

Examples

- **Sensor data (IoT):** energy demand prediction, environmental monitoring, traffic flow.
- **Marketing and e-commerce:** product recommendation, click stream analysis, sentiment analysis (social networks).
- **Cybersecurity:** malware detection, spam detection, intrusion detection.
- Many more exist!
- Not every problem should be treated as a stream learning problem!

Data Streams

When should we abstract data as a continuous “data stream”?

- Cannot store all the data.
- Should not store all the data.

Data Streams: Can't Store

Why **can't** we store all the data?

- Storing all the data may **exceed the available storage** capacity or cause practical limitations.
- The **volume or velocity** of incoming data may be too high to store and process fully.

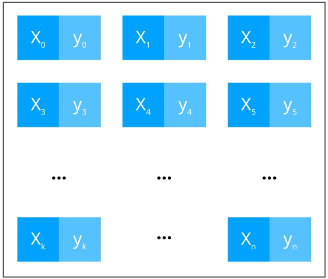
Data Streams: Shouldn't Store

Why **shouldn't** we store all the data?

- Storing all the data may not be desirable due to **privacy concerns**, **compliance requirements**, or the nature of the problem.
- For example, if we are only interested in real-time analysis or immediate decision-making.

Online Learning/Online vs Batch

Batch

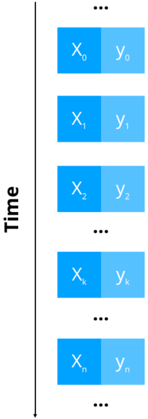


This is regular machine learning:

- Stationary dataset.
- Data is shuffled and independent and identically distributed (**IID**).
- Training and inference occurs in phases: train → test.
- Train phase can be arbitrarily long.
- Training has random access to data.

Challenges: noise, missing data, high-dimensionality—among others.

Online

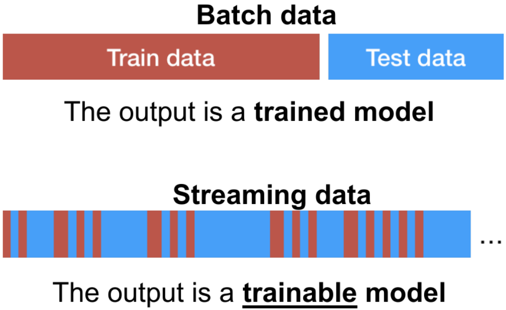


- Data flows continuously.
- There is limited time to inspect data points.
- Training and inference occurs in a cycle: ... → test → train → test → train → ...
- Relationships evolve (non-IID)—concept drift and concept evolution occur.
- **Challenges**: Those of batch, adapting to changes, concept drift, concept evolution, strict memory/processing requirements—among others.

Online Learning/Challenges

Anytime Inference

Online learners should give a good prediction at anytime.



Batch data

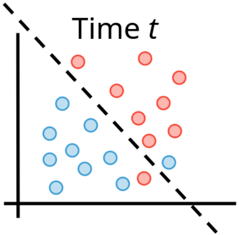
The output is a **trained model**

Streaming data

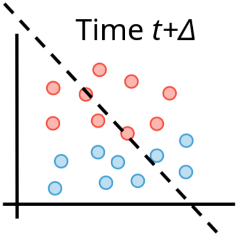
The output is a **trainable** **model**

Concept Drift

We can build a simple linear model to separate the two classes.



What if the data distribution changes?



The model is no longer accurate.

Concept drift is a change in the relationships present in data (concepts).

Online Learning Summary

An online learner predicts and learns from a data stream. A data stream is a forward-only, ordered dataset that contains evolving concepts.

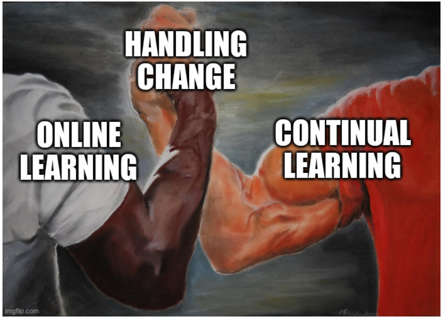
Key challenges:

- **Single Pass**
- **Anytime Inference**
- **Concept Drift**
- **Resource Constraints**

Continual Learning

Continual Learning

Continual Learning—also known as Lifelong Learning or Incremental Learning—focuses on developing *artificial neural networks* that can learn new concepts and changing concepts without forgetting.



Continual Learning/Tasks and Concepts

Tasks

- In continual learning, a “task” or “experience” is equivalent to a “concept” in stream learning.
- A task is a set of examples that contain specific concepts or relationships, such as a set of classes or data from a domain—essentially, it is a small dataset.
- A task is what the learner should learn.

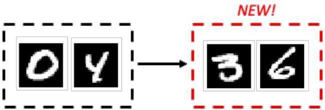
Example

- Consider a robot tasked with recognising and interacting with objects.
- Initially, the robot trains to recognise basic objects (cups, books, and pens).
- This initial training is the first "**task**".
- As the robot operates, it encounters new objects not in its initial training set (new tasks).
- The robot must adapt to these new objects and learn about them, retaining prior knowledge.

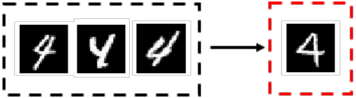
Figure: Source [Lomonaco and Maltoni, 2017]

New and Changing Tasks

Learn to classify new classes:

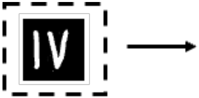


Update model to accommodate changes within existing classes:



Still a 4, but it is different

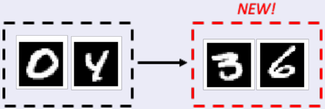
Forget what is no longer needed:



May not be used anymore

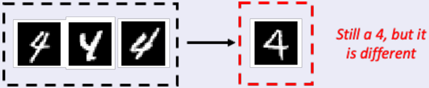
Class Incremental

Learn to classify new classes.



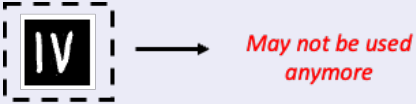
Domain Incremental

Update model to accommodate changes within existing classes.



Graceful Forgetting or Selective Forgetting

Forget what is no longer needed.



Continual Learning vs Batch Learning

- **Status Quo:** Deep learning models (artificial neural nets) learn by sampling from a single training set that contains all tasks.
- Deep learning assumes a **closed world**.
- What happens when the model must learn a new task?
  - Train from scratch, using data from **all** tasks.
  - Continue training, using data from **all** tasks.
  - **Continue training, using only new tasks**.
- This is an **open world**.

Continual Learning/Challenges

Catastrophic Forgetting

Continual learning aims to overcome catastrophic forgetting.

- **What is it?** Artificial Neural Networks (ANNs) tend to forget previously learned tasks when trained on a new sequence of tasks.
- **Why is it catastrophic?** The term “catastrophic“ is used for historic reasons to convey a severity not present in biological forgetting.
- **Why does it happen?** It happens because of distributed representations. Any new learning impacts all network parameters to some degree, overwriting previous knowledge.

Stability and Plasticity

Learning requires change, but change can destroy existing knowledge. This fundamental challenge is known as the plasticity-stability trade-off [Mermillod et al., 2013], balancing:

- **Plasticity:** The ability to learn new things. An overly plastic network experiences catastrophic forgetting.
- **Stability:** The ability to remember old things. A completely stable network cannot learn anything new.

This trade-off is a recurring theme in continual learning.

Forward and Backwards Transfer

Transfer learning occurs when learning one task improves other tasks.

In continual learning, transfer has two directions:

- **Forward Transfer:** Forward transfer occurs when a model learning a task improves performance on, or the ability to learn, future tasks. It relates to curriculum learning.
- **Backwards Transfer:** Backwards transfer occurs when a model learning a task improves performance on past tasks.

Continual Learning Summary

Continual learning tackles catastrophic forgetting in artificial neural networks.

Key challenges:

- Catastrophic forgetting and the stability-plasticity dilema.
- Forward Transfer.
- Backwards Transfer.

Online Continual Learning

Online Continual Learning

Online continual learning focuses on developing artificial neural networks that can perform inference at any time, learn from datastreams, and adapt to new and changing tasks without forgetting.

It is composed of:

- **Continual Learning:** Incrementally learns a sequence of tasks **without forgetting** previously learned ones.
- **Online Learning:** Learns and **adapts** to a non-stationary data stream, performing **anytime inference** at any point.

Online Continual Learning/Challenges

The Stability Gap

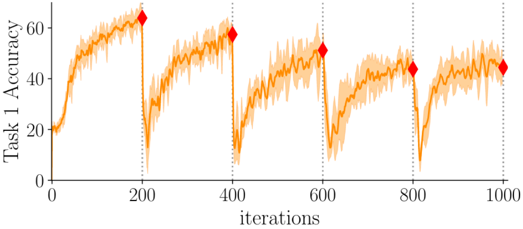
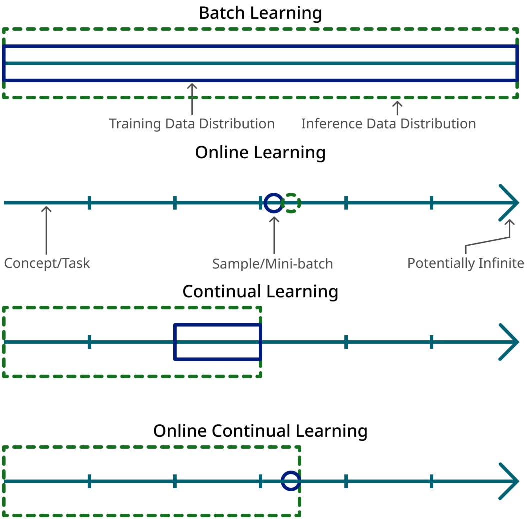


Figure: Source [Lange et al., 2023]

The Stability Gap is a temporary drop in performance on past tasks after a transition to new tasks. The stability gap is a problem for OCL because inference may occur at any time, potentially during the stability gap.

Online Continual Learning/Related Paradigms



Batch Learning

Training Data Distribution

Inference Data Distribution

Online Learning

Concept/Task

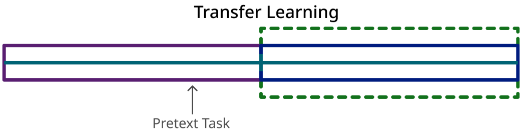
Sample/Mini-batch

Potentially Infinite

Continual Learning

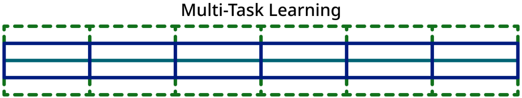
Online Continual Learning

Transfer Learning



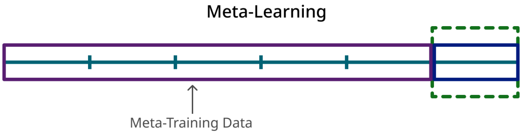
In transfer learning, a model trained on one task (pretext) is reused on another to improve performance on the new task. If we continued to care about performance on the pretext task, then it would be continual learning.

Multi-Task Learning



In multi-task learning, a single model is trained to perform multiple related tasks simultaneously. This allows the model to leverage shared information and improve performance on individual tasks, even with limited data.

Meta-learning



In meta-learning or learning-to-learn a model learns to acquire a general learning algorithms or prior knowledge.

Online Continual Learning Summary

Online continual learning combines continual learning and online learning.

Key challenges:

- **From Online Learning:** single pass, anytime inference, concept drift, and resource constraints.
- **From Continual Learning:** catastrophic forgetting, and the stability-plasticity dilemma.
- **Unique Challenge:** stability gap.

Evaluation

Tasks

Continual learning evaluation typically assumes we ground truth knowledge of tasks. A popular method to simulate incremental continual learning problems is to split a dataset into tasks.

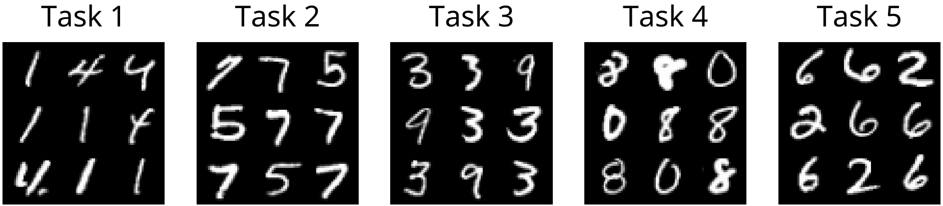


Figure: SplitMNIST10/5 is MNIST's ten classes split into five tasks, each containing two classes.

Scenarios

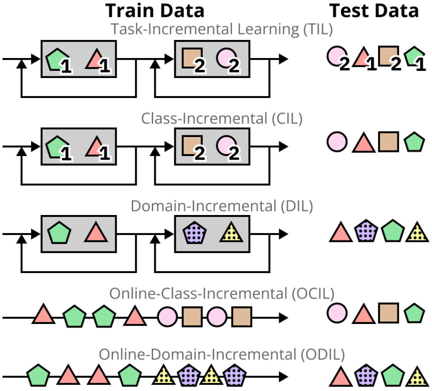

Train Data
Test Data
Task-Incremental Learning (TIL)
1
1
2
2
2
1
2
1
Class-Incremental (CIL)
1
1
2
2
Domain-Incremental (DIL)
Online-Class-Incremental (OCIL)
Online-Domain-Incremental (ODIL)

Figure: Shapes are classes, patterns are domains, and subscripts are task labels.

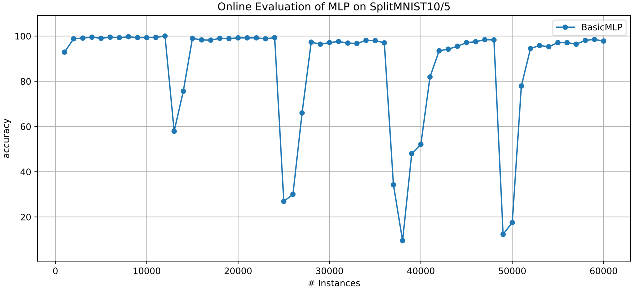
	TIL	CIL	DIL	OCIL	ODIL
No test-time task labels		✓	✓	✓	✓
No train-time task labels				✓	✓
Unknown task boundaries				✓	✓
New classes	✓	✓		✓	
New examples			✓		✓
Single pass				✓	✓
Anytime inference				✓	✓

Table: Different continual learning scenarios have different assumptions and limitations. The **"O"** prefix stands for online settings.

Online Evaluation

Interleaved test-then-train–also known as cumulative evaluation–predicts first, then trains on each example in the data stream.

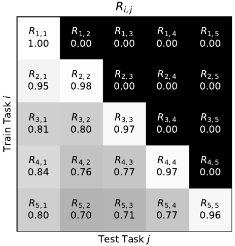
- It uses no holdout set.
- This is valid because examples are never used for testing after training.
- Windowed variants obtain a local average of model accuracy.



Continual Learning Evaluation Loop

Algorithm 1 Train Test Loop

```
for $i \in 1 \ldots T$ do
    for $k \in 1 \ldots N_i$ do
        Train on $S_{i,k}$
    end for
    for $j \in 1 \ldots T$ do
        $R_{i,j}$ = test acc. on $S'_j$
    end for
end for
```

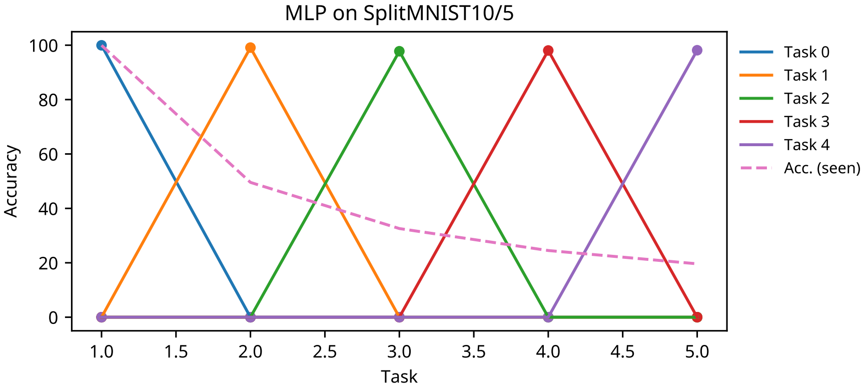


$R_{i,j}$

Train Task i	Test Task j				
	$R_{1,1}$ 1.00	$R_{1,2}$ 0.00	$R_{1,3}$ 0.00	$R_{1,4}$ 0.00	$R_{1,5}$ 0.00
	$R_{2,1}$ 0.95	$R_{2,2}$ 0.98	$R_{2,3}$ 0.00	$R_{2,4}$ 0.00	$R_{2,5}$ 0.00
	$R_{3,1}$ 0.81	$R_{3,2}$ 0.80	$R_{3,3}$ 0.97	$R_{3,4}$ 0.00	$R_{3,5}$ 0.00
	$R_{4,1}$ 0.84	$R_{4,2}$ 0.76	$R_{4,3}$ 0.77	$R_{4,4}$ 0.97	$R_{4,5}$ 0.00
	$R_{5,1}$ 0.80	$R_{5,2}$ 0.70	$R_{5,3}$ 0.71	$R_{5,4}$ 0.77	$R_{5,5}$ 0.96

- T is the number of tasks.
- N_i is the number of samples in task i.
- $S_{i,k}$ is the kth sample from task i.
- $R_{i,j}$ is the accuracy of the model after task i on task j.

Continual Learning Evaluation Metrics



This ignores performance within tasks. This is significant because online learners are expected to perform classification at any time.

Anytime Evaluation Loop

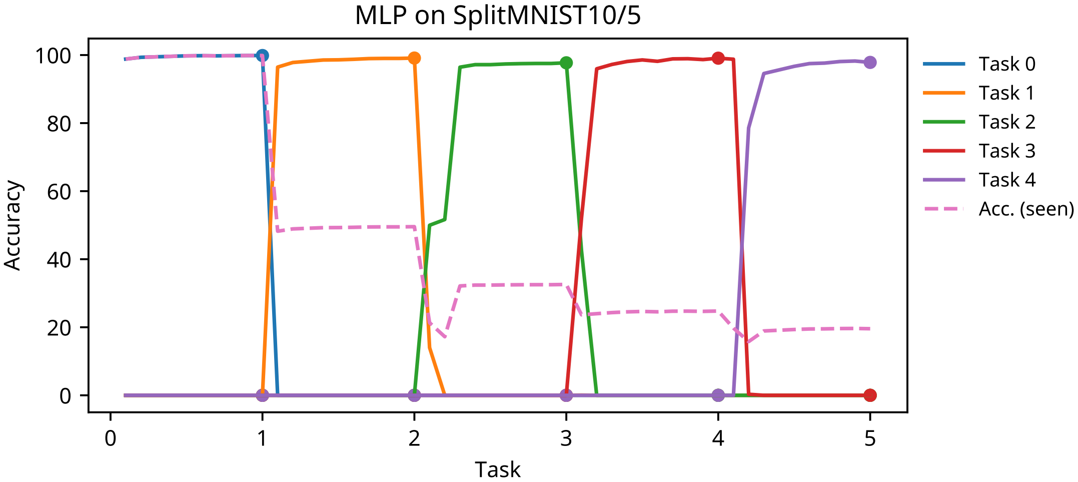
Algorithm 2 Train Test Loop

```
for $i \in 1 \ldots T$ do
    for $h \in 1 \ldots H$ do
        for $k \in \frac{(h-1)N_i}{H} \ldots h\frac{N_i}{H}$ do
            Train on $S_{i,k}$
        end for
        for $j \in 1 \ldots T$ do
            $A_{i,h,j} =$ test acc. on $S'_j$
        end for
    end for
end for
```

To evaluate anytime inference performance, intra-task evaluation is needed.

- H is the number of times within a task that the model is evaluated.
- T is the number of tasks.
- N_i is the number of samples in task i.
- $S_{i,k}$ is the kth sample from task i.
- $A_{i,h,j}$ is the accuracy of the model after intra-task step h after task i on task j.

Anytime Evaluation Metrics



Evaluation Notebook

https://github.com/adaptive-machine-learning/PAKDD2025/notebooks/00_evaluation.ipynb

CapyMOA Summary

- Provides easy access to MOA (online learning), PyTorch (deep learning), and Scikit-learn learners (batch learning).
- Supports coding in Python, Java, or both.
- Standardises evaluation for:
  - Online (or data stream) learning.
  - **Online continual learning**.
  - Semi-supervised online learning.
  - Concept drift detection.
  - Online clustering.
- Offers 20+ classifiers, 8 regressors, 11 drift detectors, and 4 anomaly detectors.

capymoa.org

Strategies

Strategies

OCL strategies are additions on top of existing artificial neural networks architectures. In practice, one often uses an existing architecture (MLP, ResNet, CNN, etc.) as the model, an optimiser (SGD, Adam, etc.), and a strategy to address continual learning.

Taxonomy

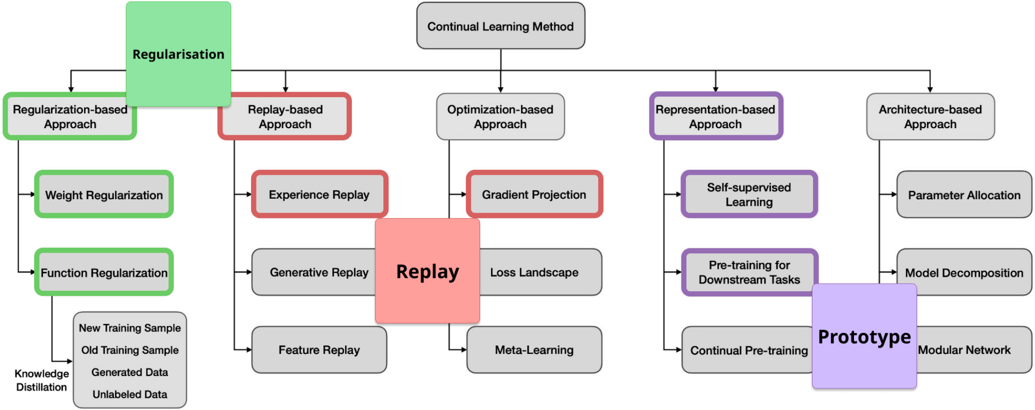
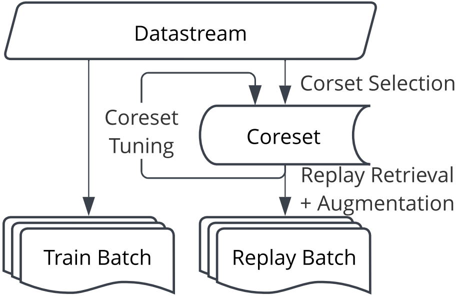


Figure: Source [Wang et al., 2023]. My annotations are in colour.

Strategies/Replay

Replay[1]



- Replay strategies store a subset of the data stream in a coreset or replay buffer.
- Like flashcards.
- During optimisation, the learner samples from the coreset to recall previously learnt tasks.
- When the buffer is small, the learner can over-fit on the replay buffer.
- Which samples should be in the coreset? (coreset selection)
- Which samples should be used from the coreset? (replay retrieval)

[1] also known as rehearsal

Experience Replay (ER)

Experience replay is a baseline method that uses reservoir sampling to uniformly sample from a buffer of past examples.

- **Coreset Selection**: Reservoir sampling selects a fixed-size buffer of past examples.
- **Coreset Retrieval**: Uniformly random.
- **Coreset Exploitation**: The learner trains on both the current batch of examples and the sampled buffer examples.

GDumb [Prabhu et al., 2020]

GDumb is a greedy sampler combined with a “dumb” learner.

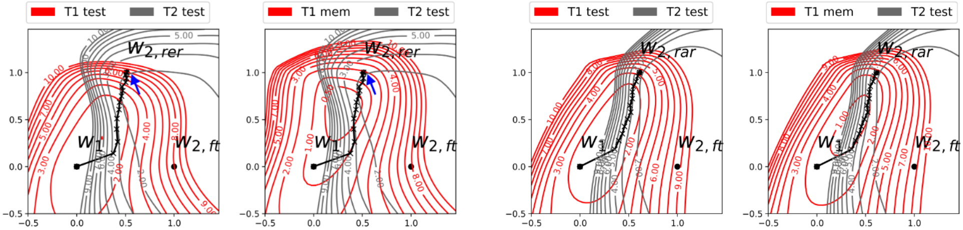
- **Coreset Selection**: It greedily stores samples in a class-balanced buffer.
- **Coreset Exploitation**: Before inference, the learner resets and trains only on the stored samples.

Since online learners do not have an “inference time,” GDumb is an offline algorithm, but it remains a useful baseline. Essentially, GDumb downsamples the dataset and trains offline.

GDumb outperforms more sophisticated algorithms, and the authors conclude that this raises questions about progress in continual learning.

Repeated Augmented Rehearsal (RAR) [Zhang et al., 2022]

- **Coreset Exploitation**: The learner trains on the current batch of examples, performing **repeated** optimisation steps with a randomly sampled replay batch and random **augmentations**.



(a) RER ($K = 10$)

(b) RAR ($K = 10$)

Figure: Source [Zhang et al., 2022]. Loss contours with (RAR) and without RER augmentations. When using augmentations, the loss contours of the coreset more closely resemble the test set contours.

Averaged Gradient Episodic Memory (A-GEM) [Chaudhry et al., 2019]

- **Coreset Exploitation**: Coreset examples **constrain optimisation using gradient projection**, ensuring learning does not increase the loss on any previously learned coreset examples.

$$\left\langle \frac{\partial \ell\left(f_\theta(x,t),y\right)}{\partial \theta}, \frac{\partial \ell\left(f_\theta,\mathcal{M}_k\right)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k.$$

This means the gradient from a new sample (left-hand side of the dot product) must have a non-negative dot product with the gradient from any memory sample (right-hand side). This ensures the new gradient does not interfere with old knowledge.

Gradient-Based Sample Selection (GSS) [Aljundi et al., 2019b]

- **Coreset Selection**: GSS reframes coreset selection as a constraint selection problem, identifying a subset of samples that imposes constraints similar to the full dataset. In practice, GSS selects samples that **maximise the diversity of gradient directions** in the coreset.

  To select the coreset $\hat{\mathcal{M}}$ from available samples $\mathcal{M}$, GSS uses:

$$\hat{\mathcal{M}} \leftarrow \text{argmin}_{\hat{\mathcal{M}}} \sum_{i,j \in \hat{\mathcal{M}}} \frac{\langle g_i, g_j \rangle}{\|g_i\| \, \|g_j\|}$$

$$\text{s.t. } \hat{\mathcal{M}} \subset \mathcal{M}; |\hat{\mathcal{M}}| = M$$

  For practicality, they have a greedy algorithm for solving this online.

Maximally Interfered Retrieval (MIR) [Aljundi et al., 2019a]

- **Coreset Retrieval**: MIR selects top-k samples from a replay buffer based on how much their loss changes after a gradient update from the current training batch. Samples in memory $(\mathbf{x}, y) \in \mathcal{M}$ are ranked using:

$$s(\mathbf{x}, y) = \ell(f(\mathbf{x}, \theta_v), y) - \ell(f(\mathbf{x}, \theta), y)$$

where $\theta_v = \theta - \alpha \nabla \mathcal{L}(f(\mathbf{X}_i, \theta), \mathbf{Y}_i)$ and $(\mathbf{X}_i, \mathbf{Y}_i)$ are the incoming training samples.

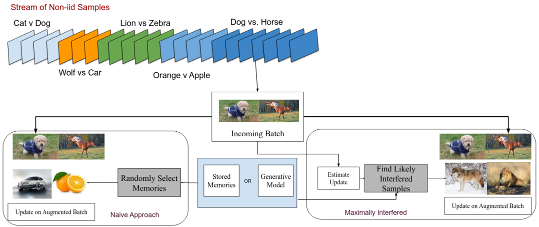


Figure: Source [Aljundi et al., 2019a].

Replay Notebook

https://github.com/adaptive-machine-learning/PAKDD2025/notebooks/01_replay.ipynb

Strategies/Regularisation

Regularisation

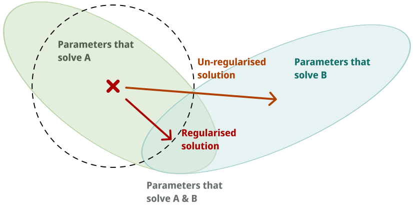
Regularisation strategies add a term to the loss function. This encourages the model to remain similar to a previous model.

There are two main approaches:

- **Weight Regularisation:** This adds a loss term that encourages the model's parameters to align with those of previous tasks.
- **Functional Regularisation:** This adds a loss term to keep the network's behaviour consistent with a previous model.

Regularisation strategies tend to perform poorly on class-incremental learning problems because they have no way to learn the differences between classes from different tasks.

Weight Regularisation



Many weight regularisation methods follow the same pattern:

$$\mathcal{L}(\theta) = \mathcal{L}_{ce}(\theta) + \lambda \sum_i \Omega_i(\theta_i - \theta_i^*)^2$$

where:

- θ is the model that is being trained.
- θ^* is a historical model containing knowledge that the method wants to remember.
- λ is the strength of the regularisation.
- Ω measures parameter importance. In L2, $\Omega = 1$. In EWC, it is the diagonal of the Fisher information matrix [Kirkpatrick et al., 2017].

Functional Regularisation (LWF) [Li and Hoiem, 2016]

$$\mathcal{L}(\theta) = \mathcal{L}_{ce}(\theta) + \text{kd-distance}(f_\theta(x,y), f_\theta^*(x,y))$$

- $f_\theta(x,y)$ is the output of the current model.
- $f_\theta^*(x,y)$ is the output of a past model.
- $\text{kd-distance}(f_\theta(x,y), f_\theta^*(x,y))$ is a knowledge distillation loss component. It encourages the current model to **mimic the behaviour** of the older model on previous tasks, thereby preserving learned representations.

Regularisation Notebook

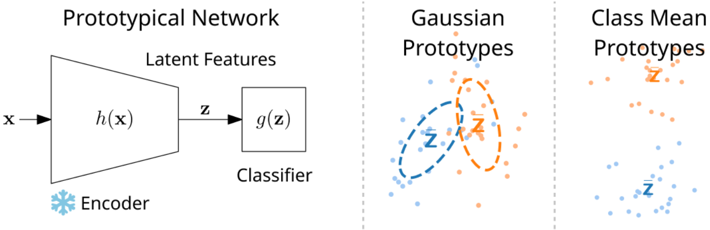
https://github.com/adaptive-machine-learning/PAKDD2025/notebooks/02_regularisation.ipynb

Strategies/Prototype

Pretrained Prototypes

Neural networks often divide into two components:

- **Backbone**: extracts features (e.g., ViT, ResNet).
- **Head**: classifies using the features (e.g., linear layer, NCM, KNN).



These approaches avoid catastrophic forgetting by using a frozen pretrained backbone and a head that is immune to catastrophic forgetting. If the backbone was trained on something similar to the downstream task, it often performs well.

Nearest Class Mean (NCM) [Mensink et al., 2013]

- NCM “learns” by calculating the **mean of each class in the embedding space**—creating a prototype for each class.
- During inference, the algorithm measures the **distance of an instance’s embedding to each class prototype**, assigning the instance to the class with the closest prototype.
- NCM resembles k-Nearest Neighbors (kNN) but stores only one prototype per class instead of all training samples.
- The frozen backbone provides nowhere for forgetting to occur.
- When the encoder is not frozen, its representation changes, invalidating the class means of NCM.

NCM’s effectiveness stems from a quality feature extractor/backbone.

Streaming Linear Discriminant Analysis (SLDA) [Hayes and Kanan, 2020]

- Moving beyond NCM, Gaussian discriminant analysis-based methods, such as streaming linear discriminant analysis, consider each class's spread and variance in the embedding space.
- SLDA "learns" by incrementally calculating the mean and **covariance** of each class in the embedding space—creating a prototype for each class.
- Others have explored other types of streaming/incremental Gaussian discriminant analysis for continual learning [McDonnell et al., 2023, Goswami et al., 2023, Prabhu et al., 2024].
- The frozen backbone provides nowhere for forgetting to occur.

Incremental Classifier and Representation Learning (iCaRL) [Rebuffi et al., 2017]

- **Encoder:** Trained online.
- **Prototypes:** Computed as the mean of exemplars in the coreset at inference time.
- **Coreset Selection**: Constructed *offline* to approximate the class mean. Ordered such that the mean of the coreset rapidly converges to dataset mean. *Online* implementations often use reservoir sampling.
- **Coreset Exploitation**: Train on batch and replay batch with a classification and distillation term, like LWF.

Supervised Contrastive Replay (SCR) [Mai et al., 2021]

Supervised Contrastive Replay uses replay and self-supervised learning to update the encoder and recompute class mean prototypes using the replay buffer.

- Contrastive learning learns to encode augmented views of data nearby each other, and different data far from each other.
- SCR uses no pretrained encoder.
- Forgetting occurs in the encoder.
- The NCM head needs constant updates.

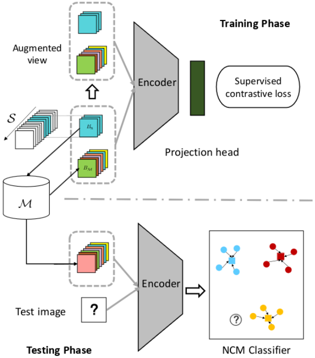


Figure: [Mai et al., 2021]

Prototype Notebook

https://github.com/adaptive-machine-learning/PAKDD2025/notebooks/03_prototype.png

Strategies/Others

Others

1. **Meta-learning:** Meta-learning applies machine learning to tune the hyper-parameters of optimisation algorithms (learning-to-learn). You can treat OCL as a meta-learning problem—to tune an initial configuration or tune regularisation importance parameters.
2. **Architecture:** One can add or group parameters to avoid catastrophic forgetting. This includes ensemble methods for avoiding forgetting.
3. **Generative Replay:** Instead of a coreset, generative replay uses a generative model to sample from earlier tasks.

Conclusion

Conclusion

We have learnt that:

1. Online continual learning (OCL) combines continual learning with online learning, inheriting the objectives of both.
2. OCL adapts to changes in data distribution–tasks–without forgetting previous knowledge.
3. Catastrophic forgetting occurs in neural networks when learning a new task causes a loss of performance on tasks learned previously.
4. In OCL, performance at any time is important, not only the final result.
5. Replay methods store a subset of the stream to remind the model of past data.
6. Regularisation methods add a loss term. This term encourages the model to behave as it did in the past.
7. Pretrained prototype methods avoid forgetting and are often effective.

Questions?

capymoa.org

CapyMOA Team

- Heitor Murilo Gomes (project lead)[1]
- Anton Lee[1]
- Nuwan Gunasekara[2]
- Yibin Sun[2]
- Guilherme Cassales[2]
- Marco Heyden[3]
- Justin Liu[2]
- Jesse Read[4]
- Maroua Bahri[5]
- Marcus Botacin[6]
- Vitor Cerqueira[7]
- Albert Bifet,9[2]
- Bernhard Pfahringer[2]
- Yun Sing Koh[8]
- And other contributors.

[1] Victoria University of Wellington, New Zealand

[2] University of Waikato, New Zealand

[3] KIT, Germany

[4] École polytechnique, IP Paris, France

[5] Sorbonne Université, France

[6] Texas A&M Engineering, USA

[7] Porto University, Portugal

[8] University of Auckland, New Zealand

[9] Télécom Paris, IP Paris, France

References I

Aljundi, R., Caccia, L., Belilovsky, E., Caccia, M., Lin, M., Charlin, L., and Tuytelaars, T. (2019a).
Online Continual Learning with Maximally Interfered Retrieval.
arXiv:1908.04742 [cs, stat].
arXiv: 1908.04742.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019b).
Gradient based sample selection for online continual learning.
In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019, NeurIPS 2019, december 8-14, 2019, vancouver, BC, canada*, pages 11816–11825.
947 citations (google-scholar/DOI) [2025-01-22].

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019).
Efficient Lifelong Learning with A-GEM.
arXiv:1812.00420 [cs, stat].

References II

Goswami, D., Liu, Y., Twardowski, B., and Weijer, J. v. d. (2023).
FeCAM: Exploiting the Heterogeneity of Class Distributions in Exemplar-Free Continual Learning.
In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Hayes, T. L. and Kanan, C. (2020).
Lifelong Machine Learning with Deep Streaming Linear Discriminant Analysis.
In *CLVision Workshop at CVPR 2020*, pages 1–15.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017).
Overcoming catastrophic forgetting in neural networks.
Proceedings of the National Academy of Sciences, 114(13):3521–3526.
Publisher: National Academy of Sciences Section: Biological Sciences.

References III

Lange, M. D., Ven, G. v. d., and Tuytelaars, T. (2023).
Continual evaluation for lifelong learning: Identifying the stability gap.
arXiv:2205.13452 [cs].

Li, Z. and Hoiem, D. (2016).
Learning without forgetting.
CoRR, abs/1606.09282.
arXiv: 1606.09282 tex.bibsource: dblp computer science bibliography, https://dblp.org tex.biburl: https://dblp.org/rec/journals/corr/LiH16e.bib tex.timestamp: Thu, 31 Dec 2020 11:34:47 +0100.

Lomonaco, V. and Maltoni, D. (2017).
CORe50: a New Dataset and Benchmark for Continuous Object Recognition.
In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 17–26. PMLR.
ISSN: 2640-3498.

References IV

Mai, Z., Li, R., Kim, H., and Sanner, S. (2021).
Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning.
In *IEEE conference on computer vision and pattern recognition workshops, CVPR workshops 2021, virtual, june 19-25, 2021*, pages 3589–3599. Computer Vision Foundation / IEEE.
49 citations (opencitations/DOI) [2025-01-22].

McDonnell, M. D., Gong, D., Parvaneh, A., Abbasnejad, E., and van den Hengel, A. (2023).
RanPAC: Random projections and pre-trained models for continual learning.
In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in neural information processing systems 36: Annual conference on neural information processing systems 2023, NeurIPS 2023, new orleans, LA, USA, december 10 - 16, 2023*.
tex.bibsource: dblp computer science bibliography, https://dblp.org tex.timestamp: Fri, 01 Mar 2024 16:26:19 +0100.

References V

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2013).
Distance-based image classification: Generalizing to new classes at near-zero cost.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(11):2624–2637.
tex.bibsource: dblp computer science bibliography, https://dblp.org tex.timestamp: Tue, 26 Apr 2022 17:22:56 +0200.

Mermillod, M., Bugaiska, A., and BONIN, P. (2013).
The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects.
Frontiers in Psychology, 4:504.

Prabhu, A., Sinha, S., Kumaraguru, P., Torr, P. H. S., Sener, O., and Dokania, P. K. (2024).
RanDumb: A Simple Approach that Questions the Efficacy of Continual Representation Learning.
arXiv:2402.08823 [cs].

References VI

Prabhu, A., Torr, P. H. S., and Dokania, P. K. (2020).
GDumb: A Simple Approach that Questions Our Progress in Continual Learning.
In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 524–540, Cham. Springer International Publishing.
209 citations (Crossref/DOI) [2025-01-24].

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017).
iCaRL: Incremental Classifier and Representation Learning.
In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, L., Zhang, X., Su, H., and Zhu, J. (2023).
A Comprehensive Survey of Continual Learning: Theory, Method and Application.
arXiv:2302.00487 [cs].

References VII

Zhang, Y., Pfahringer, B., Frank, E., Bifet, A., Lim, N. J. S., and Jia, Y. (2022).
A simple but strong baseline for online continual learning: Repeated Augmented Rehearsal.
In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in neural information processing systems 35: Annual conference on neural information processing systems 2022, NeurIPS 2022, new orleans, LA, USA, november 28 - december 9, 2022*.
57 citations (google-scholar/DOI) [2025-01-22].