Dark Experience for General Continual Learning

Pietro Buzzega Matteo Boschini Angelo Porrello Davide Abati Simone Calderara
AImageLab - University of Modena and Reggio Emilia, Modena, Italy

Antonio Impalà Kristian Di Blasi Giuseppe Germano

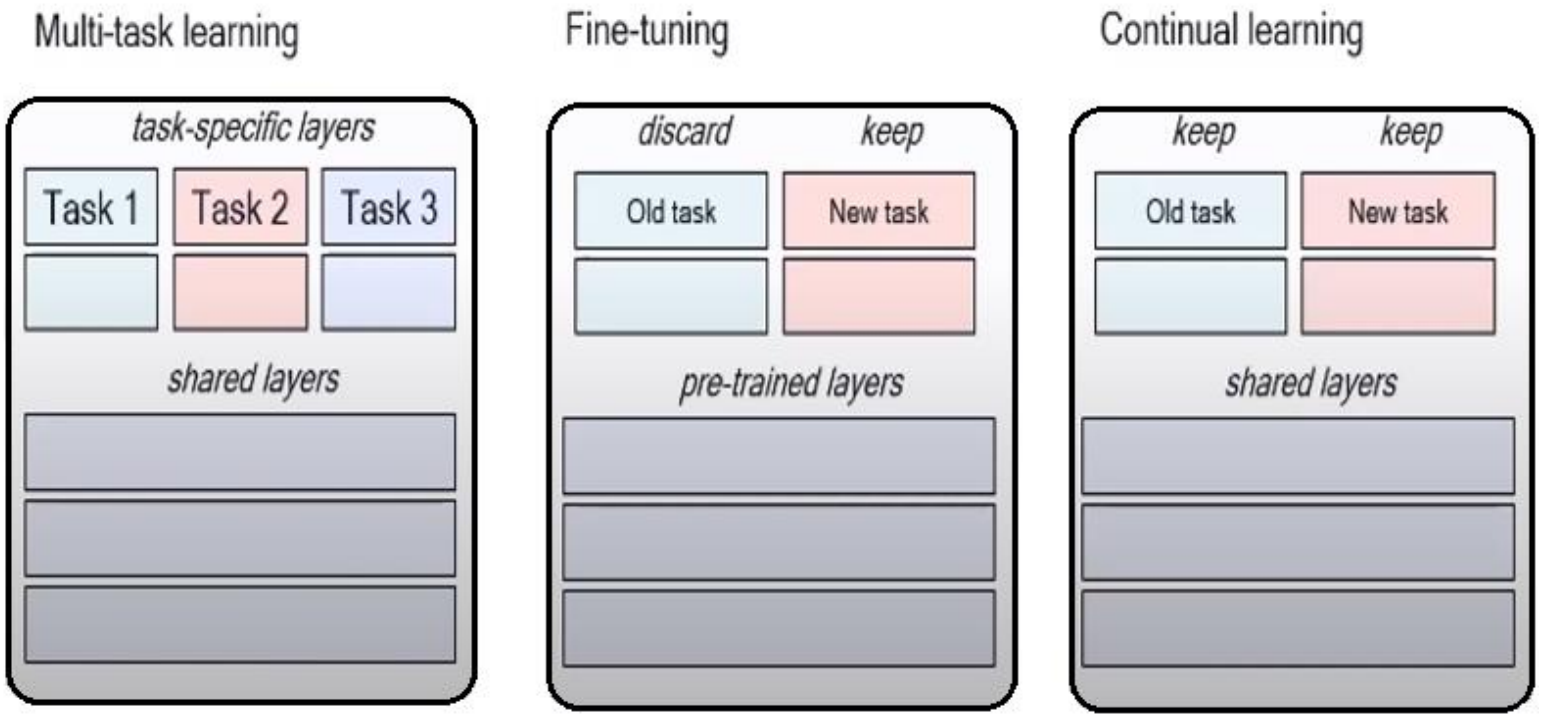
Continual Learning problem

- Continual Learning, also known as Lifelong learning, is built on the idea of learning continuously about the external world in order to enable the autonomous, incremental development of ever more complex skills and knowledge.

- A Continual learning system can be defined as *an adaptive algorithm capable of learning from a continuous stream of information, with such information becoming progressively available over time and where the number of tasks to be learned (e.g. membership classes in a classification task) are not predefined. Critically, the accommodation of new information should occur without catastrophic forgetting or interference.*

Key concept: TASK

Each task is a layer specific, e.g. for classification, from which the model learns a knowledge.
In CL each knowledge, hence each task, as to be preserved while computing another one and usually initial layers are shared.



Continual Learning Desiderata

1. **Avoid forgetting**
   - Performance over previous tasks should not decrease
2. **Fixed memory and compute**
   - If not possible, grow sub-linearly with tasks
3. **Enable forward transfer**
   - Knowledge acquired over previous tasks should help learning future tasks
4. **Enable backward transfer**
   - While learning the current task, performance in previous tasks may also increase
5. **Do not store examples**
   - Or store as few as possible

Approaches to CL

PARAMETER ISOLATION

Explicitly identify important parameters for each task

MODEL GROWING

Increase the model capacity for every new task

KNOWLEDGE DISTILLATION

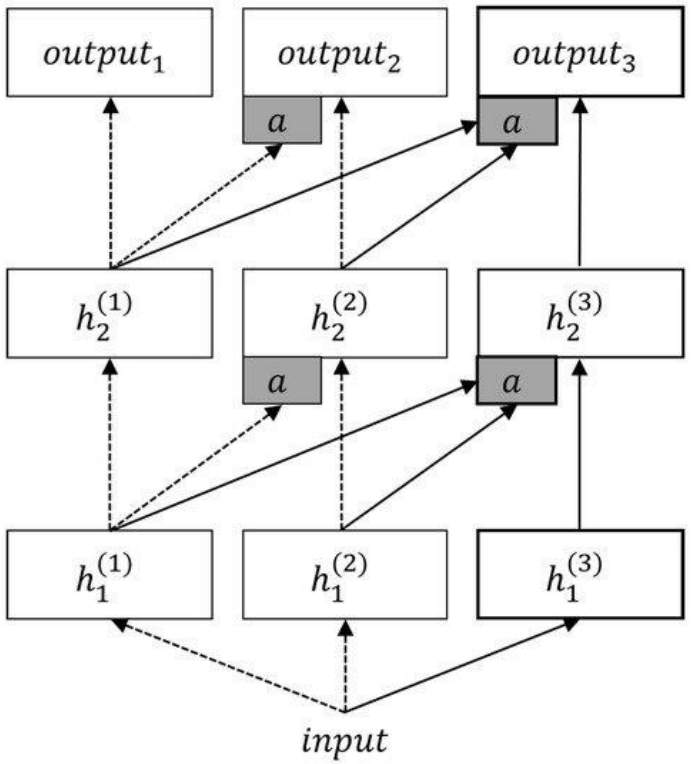
Use the model in a previous training state as a teacher

REGULARIZATION

Penalize (some) parameter variations

REHEARSAL

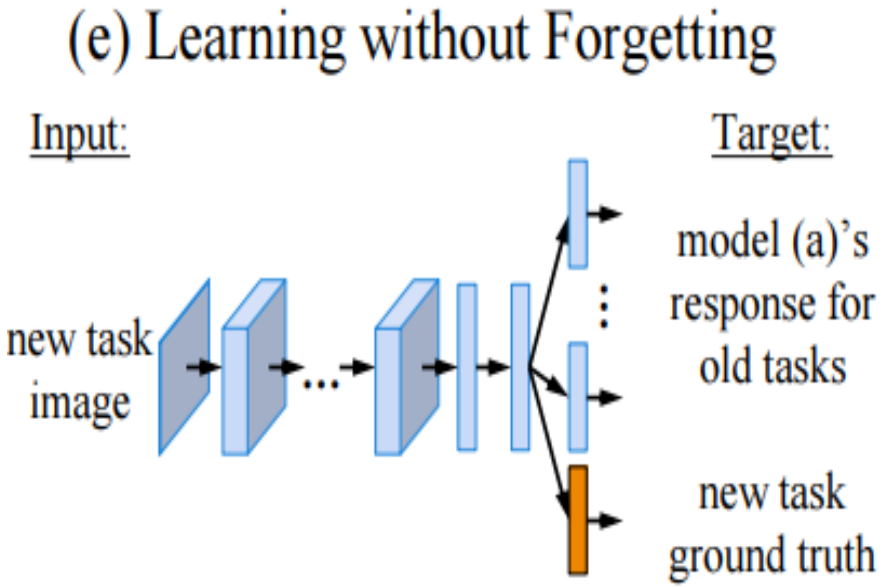
Store old inputs and replay them to the model.



Approaches to CL Model growing methods

E.g. PNN:

- A progressive neural network (prognets) is a neural algorithm developed by Deepmind in their paper *Progressive Neural Networks* (Rusu et al., 2016). Prognets are a simple, powerful, and creative solution to transfer learning — to quote the paper abstract, they "are immune to forgetting and can leverage prior knowledge via lateral connections to previously learned features"

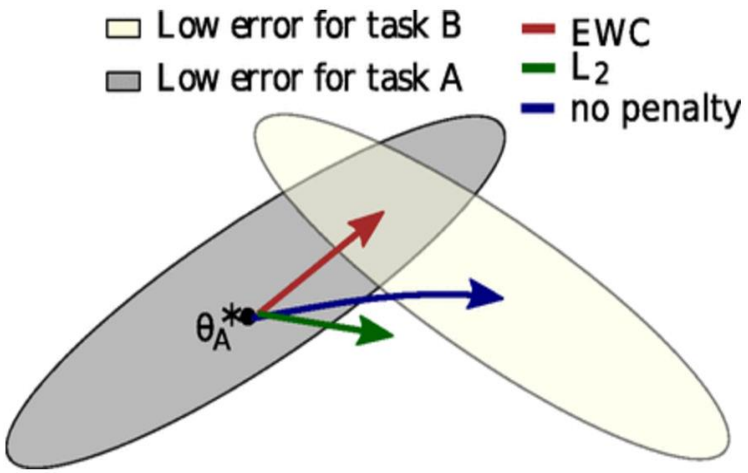


Approaches to CL Knowledge distillation methods

E.g. Learning without forgetting:

- When building a unified vision system or gradually adding new capabilities to a system, the usual assumption is that training data for all tasks is always available. However, as the number of tasks grows, storing and retraining on such data becomes infeasible. A new problem arises where we add new capabilities to a Convolutional Neural Network (CNN), but the training data for its existing capabilities are unavailable. The model uses only new task data to train the network while preserving the original capabilities.

Approaches to CL
Regularization methods



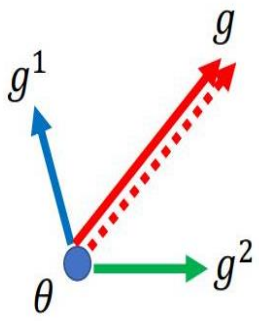
$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

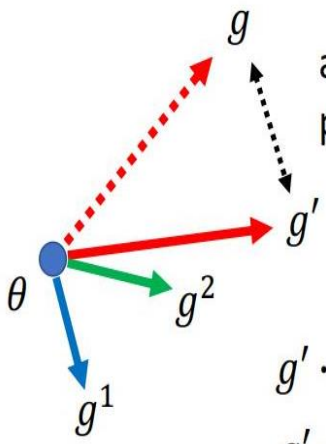
E.g. Elastic weight consolidation:

- EWC ensures task A is remembered whilst training on task B.
  Training trajectories are illustrated in a schematic parameter space, with parameter regions leading to good performance on task A (gray) and on task B (cream). After learning the first task, the parameters are at θ * A . If we take gradient steps according to task B alone (blue arrow), we will minimize the loss of task B but destroy what we have learnt for task A.
  On the other hand, if we constrain each weight with the same coefficient (green arrow) the restriction imposed is too severe and we can only remember task A at the expense of not learning task B. EWC, conversely, finds a solution for task B without incurring a significant loss on task A (red arrow) by explicitly computing how important weights are for task A.

Gradient Episodic Memory (GEM)

- Constraint the gradient to improve the previous tasks





as close as possible

$g' \cdot g^1 \geq 0$

$g' \cdot g^2 \geq 0$

: negative gradient of current task

negative gradient of previous task

: update direction

Need the data from the previous tasks

Approaches to CL Rehearsal methods

E.g. (GEM) Gradient Episodic Memory:

- Here models keeps a buffer of experiences, an episodic memory.
- Hence during training process, not only models tries to select the best parameters focusing on loss of input stream, but also uses the buffer to constrain the update step by considering also the loss related to buffer memory (past experiences).

Dark Experience Replay

DER work towards General Continual Learning (GCL), where task boundaries blur and the domain and class distributions shift either gradually or suddenly. Method address it through mixing rehearsal with knowledge distillation and regularization; Dark Experience Replay, matches the network's logits sampled throughout the optimization trajectory, thus promoting consistency with its past. By conducting an extensive analysis on both standard benchmarks and a novel GCL evaluation setting (MNIST-360)

Dark Experiences Replay

- Considering task t ∈ {1, ..., T} and input (x, y) from an i.i.d. distribution D_t.
- A function f, with parameters θ, is optimized on one task at a time in a sequential manner.
- The goal is to learn how to correctly classify, at any given point in training, examples from any of the observed tasks up to the current one t ∈ {1, . . . , t_c}
- output logits: $h_\theta(x)$
- probability distribution: f_θ(x) , Softmax($h_\theta(x)$)

$$\underset{\theta}{\operatorname{argmin}} \sum_{t=1}^{t_c} \mathcal{L}_t, \quad \text{where} \quad \mathcal{L}_t \triangleq \mathbb{E}_{(x,y)\sim D_t}\left[\ell(y, f_\theta(x))\right].$$

Dark Experiences Replay

Ideally, we look for parameters that fit the current task well while approximating the behavior observed in the old ones: effectively, we encourage the network to mimic its original responses for past samples. To preserve the knowledge about previous tasks, we seek to minimize the following objective:

$$\mathcal{L}_{t_c} + \alpha \sum_{t=1}^{t_c-1} \mathbb{E}_{x \sim D_t} \left[D_{KL}(f_{\theta_t^*}(x) \,||\, f_\theta(x)) \right]$$

where $\theta * t$ is the optimal set of parameters at the end of task t, and α is a hyper-parameter balancing the trade-off between the terms. This objective, which resembles the teacher-student approach, would require the availability of D_t for previous tasks. To overcome such a limitation, we introduce a replay buffer M_t holding past experiences for task t. Differently from other rehearsal-based methods, we retain the network's logits z , $h_\theta t(x)$, instead of the ground truth labels y.

$$\mathcal{L}_{t_c} + \alpha \sum_{t=1}^{t_c-1} \mathbb{E}_{(x,z) \sim \mathcal{M}_t} \left[D_{KL}(\mathrm{softmax}(z) \,||\, f_\theta(x)) \right].$$

As a focus on General Continual Learning, it is intentionally avoided to rely on task boundaries to populate the buffer as the training progresses. Therefore, in place of the common task-stratified sampling strategy, it's adopted a reservoir sampling: this way, selected |M| random samples from the input stream, guaranteeing that they have the same probability |M|/|S| of being stored in the buffer, without knowing the length of the stream S in advance. We can rewrite previous formula as follows:

$$\mathcal{L}_{t_c} + \alpha \, \mathbb{E}_{(x,z)\sim\mathcal{M}} \left[D_{KL}(\text{softmax}(z) \,||\, f_\theta(x)) \right].$$

Such a strategy implies picking logits z during the optimization trajectory, so potentially different from the ones that can be observed at the task's local optimum. Even if counter-intuitive, we empirically observed that this strategy does not hurt performance, while still being suitable without task boundaries. Furthermore, we observe that the replay of sub-optimal logits has beneficial effects in terms of flatness of the attained minima and calibration

Under mild assumptions, the optimization of the KL divergence in Eq. 4 is equivalent to minimizing the Euclidean distance between the corresponding pre-softmax responses (i.e. logits). The work strategy opts for matching logits, as it avoids the information loss occurring in probability space due to the squashing function (e.g., softmax). With these considerations in hands, Dark Experience Replay optimizes the following objective:

$$\mathcal{L}_{t_c} + \alpha \, \mathbb{E}_{(x,z)\sim\mathcal{M}} \left[\|z - h_\theta(x)\|_2^2 \right].$$

DER and DER++ algorithm

Algorithm 1 - Dark Experience Replay

Input: dataset D, parameters θ, scalar α, learning rate λ

$\mathcal{M} \leftarrow \{\}$
for (x, y) **in** D **do**
  $(x', z', y') \leftarrow \mathit{sample}(\mathcal{M})$
  $x_t \leftarrow \mathit{augment}(x)$
  $x'_t \leftarrow \mathit{augment}(x')$
  $z \leftarrow h_\theta(x_t)$
  $reg \leftarrow \alpha \|z' - h_\theta(x'_t)\|_2^2$
  $\theta \leftarrow \theta + \lambda \cdot \nabla_\theta[\ell(y, f_\theta(x_t)) + reg]$
  $\mathcal{M} \leftarrow \mathit{reservoir}(\mathcal{M}, (x, z))$
end for

Algorithm 2 - Dark Experience Replay ++

Input: dataset D, parameters θ, scalars α and β, learning rate λ

$\mathcal{M} \leftarrow \{\}$
for (x, y) **in** D **do**
  $(x', z', y') \leftarrow \mathit{sample}(\mathcal{M})$
  $(x'', z'', y'') \leftarrow \mathit{sample}(\mathcal{M})$
  $x_t \leftarrow \mathit{augment}(x)$
  $x'_t,\ x''_t \leftarrow \mathit{augment}(x'),\ \mathit{augment}(x'')$
  $z \leftarrow h_\theta(x_t)$
  $reg \leftarrow \alpha \|z' - h_\theta(x'_t)\|_2^2 + \beta\, \ell(y'', f_\theta(x''_t))$
  $\theta \leftarrow \theta + \lambda \cdot \nabla_\theta[\ell(y, f_\theta(x_t)) + reg]$
  $\mathcal{M} \leftarrow \mathit{reservoir}(\mathcal{M}, (x, z, y))$
end for

Experiments: key concept

Task sequence adhere the following three settings:

- Task Incremental Learning (Task-IL) and Class Incremental Learning (Class-IL):

  The training samples are splitted into partitions of classes (tasks).

  Although similar, the former provides task identities to select the relevant classifier for each example, whereas the latter does not

  Task-IL and Class-IL are the easiest and hardest scenarios among the three. In practice, **CIFAR-10 and Tiny ImageNet are splitted in 5 and 10 tasks**, each introduces **2 and 20 classes** respectively. We show all the classes in the same fixed order across different runs.

- Domain Incremental Learning (Domain-IL):

  feeds all classes to the network during each task, but applies a task-dependent transformation to the input; task identities remain unknown at test time

  Permuted MNIST and Rotated MNIST are used. They both require the learner to classify all MNIST digits for **20 subsequent tasks**, but the former applies a random permutation to the pixels, whereas the latter rotates the images by a random angle in the interval [0, π).

Evaluation Protocol

- **Architecture:**

  For tests we conducted on variants of the MNIST dataset, is used a fully-connected network with two hidden layers, each one comprising of 100 ReLU units. For CIFAR-10 and Tiny ImageNet, is used ResNet18 [15] (not pre-trained).

- **Augmentation:**

  For CIFAR-10 and Tiny ImageNet, we apply random crops and horizontal flips to both stream and buffer examples. Competitors use the same approace for fairness. It is worth noting that combining data augmentation with regularization objective enforces an implicit consistency loss, which aligns predictions for the same example subjected to small data transformations.

- **Hyperparameter selection:**

  Hyperparameters are selected by performing a grid-search on a validation set, the latter obtained by sampling 10% of the training set. For the Domain-IL scenario, we make use of the final average accuracy as the selection criterion. Differently, we perform a combined grid-search for Class-IL and Task-IL, choosing the configuration that achieves the highest final accuracy averaged on the two settings

- **Training:**

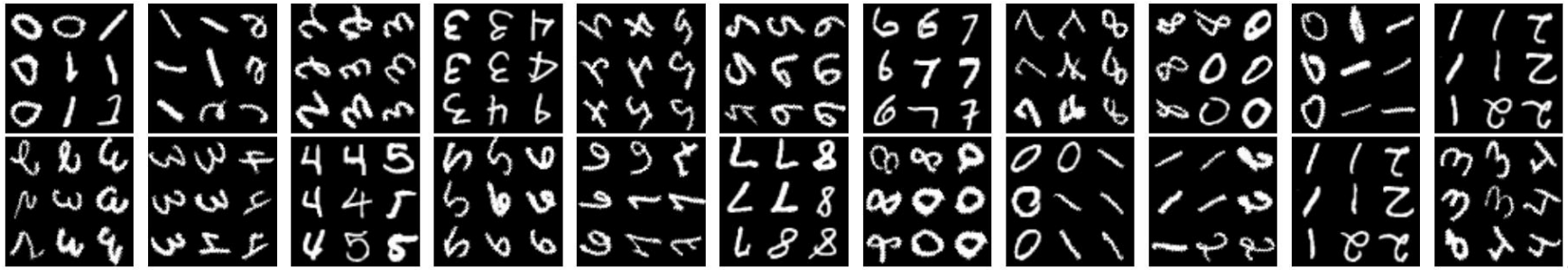
  To provide a fair comparison among CL methods, all networks are trained using the Stochastic Gradient Descent (SGD) optimizer. Despite being interested in an online scenario, with no additional passages on the data, it is then necessary to set the number of epochs per task in relation to the dataset complexity.

  For MNIST-based settings, one epoch per task is sufficient. Conversely, for Sequential CIFAR-10 and Sequential Tiny ImageNet the number of epochs is increased to 50 and 100 respectively.

A study case: MNIST-360 for GCL

To address the General Continual Learning desiderata, paper propose a novel protocol: MNIST-360. It models a stream of data presenting batches of two consecutive MNIST digits at a time (e.g. {0, 1}, {1, 2}, {2, 3} etc.). By rotating each example of the stream by an increasing angle and, after a fixed number of steps, switch the lesser of the two digits with the following one. As it is impossible to distinguish 6's and 9's upon rotation, we do not use 9's in MNIST-360. The stream visits the nine possible couples of classes three times, allowing the model to leverage positive transfer when revisiting a previous task. In the implementation, it's guaranteed that:

- each example is shown once during the overall training.
- two digits of the same class are never observed under the same rotation.



MNIST-360 training/test details implementations (backbone)

- Training:

  For Training purposes, we build batches using exemplars that belong to two consequent classes at a time, meaning that 9 pairs of classes are possibly encountered: (0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), and (8, 0). Each pair is shown in this order in R rounds (R = 3 in our experiments) at changing rotations. This means that MNIST-360 consists of $9 \cdot R$ pseudo-tasks, whose boundaries are not signaled to the tested method. We indicate them with Ψ (d1,d2) r where $r \in \{1, \ldots, R\}$ is the round number and d1, d2 are digits forming one of the pairs listed above.

  As every MNIST digit d appears in $2 \cdot R$ pseudo-tasks, we randomly split its example images evenly in 6 groups Gd i where $i \in \{1, \ldots, 2 \cdot R\}$.

- Test:

  As no task boundaries are provided, evaluation on MNIST-360 can only be carried out after the training is complete.

  The order with which digits are shown is irrelevant, therefore no specific batching strategy is necessary and we simply show one digit at a time.

Accuracy measurements on Datasets using DER/DER++

Here is showed the confidence interval for each dataset and setting, using DER and DER ++, using the following parameters:

- confidence interval: 95% -> 1.960
- Number of runs: 10

Furthermore, test on seq-Cifer10 and seq-Tinyimagenet were substituted with seq-MNIST as it is computationally feaseable.

Buffer 200	S-MNIST Class-IL (%)	S-MNIST Task-IL (%)	P-MNIST Domain-IL (%)	R-MNIST Domain-IL (%)
DER	84.80 +/- 1.73	98.75 +/- 0.15	81.19 +/- 0.41	90.57 +/- 0.77
DER++	85.58 +/- 1.06	98.84 +/- 0.10	83.51 +/- 0.55	90.75 +/- 0.93
DER(paper)	84.55 +/- 1.64	98.80 +/- 0.15	81.74 +/- 1.07	90.04 +/- 2.61
DER++(paper)	85.61 +/- 1.40	98.76 +/- 0.28	83.58 +/- 0.59	90.43 +/- 1.87
Buffer 500	S-MNIST Class-IL (%)	S-MNIST Task-IL (%)	P-MNIST Domain-IL (%)	R-MNIST Domain-IL (%)
DER	91.54 +/- 1.59	98.86 +/- 0.08	87.39 +/- 0.20	92.48 +/- 0.72
DER++	91.55 +/- 0.41	99.00 +/- 0.05	87.99 +/- 0.44	92.57 +/- 0.46
DER(paper)	90.54 +/- 1.18	98.84 +/- 0.13	87.29 +/- 0.46	92.24 +/- 1.12
DER++(paper)	91.00 +/-1.49	98.94 +/- 0.27	88.21 +/- 0.39	92.77 +/- 1.05
Buffer 5120	S-MNIST Class-IL (%)	S-MNIST Task-IL (%)	P-MNIST Domain-IL (%)	R-MNIST Domain-IL (%)
DER	94.42 +/- 0.94	99.29 +/- 0.10	91.62 +/- 0.06	94.08 +/- 0.15
DER++	95.20 +/- 0.60	99.48 +/- 0.05	92.11 +/- 0.26	94.56 +/- 0.35
DER(paper)	94.90 +/- 0.57	99.29 +/- 0.11	91.66 +/- 0.11	94.14 +/- 0.31
DER++(paper)	95.30 +/- 1.20	99.47 +/- 0.07	92.26 +/- 0.17	94.65 +/- 0.33

Other tests on MNIST-360 for GCL, under same conditions

Accordingly, they define the General Continual Learning setting (GCL) by proposing a series of desiderata for CL methods to be applicable in practice. Most importantly:

- no task boundaries: do not rely on boundaries between tasks during training, as they may not exist in practice;
- no test time oracle: do not require task identifiers at inference time;
- constant memory: have a bounded memory footprint throughout the entire training phase.

Buffer	DER (%)	DER++ (%)	DER (%) (paper)	DER++ (%) (paper)
200	55.21 +/- 1.58	55.04 +/- 1.54	55.22 +/- 1.67	54.16 +/- 3.02
500	69.08 +/- 1.25	69.38 +/- 1.32	69.11 +/- 1.66	69.62 +/- 1.59
1000	75.97 +/- 1.04	75.48 +/- 1.24	75.97 +/- 2.08	76.03 +/- 1.61

Testing with others parameters

	LR = 0.001 Batch size = 10 N epochs = 1 Minibatch size = 128 Alpha = 1		LR = 0.001 Batch size = 8 N epochs = 10 Minibatch size = 8 Alpha = 1	
Buffer 500	S-MNIST Class-IL (%)	S-MNIST Task-IL (%)	S-MNIST Class-IL (%)	S-MNIST Task-IL (%)
DER	63.98 +/- 1.52	96.6 +/- 0.28	90.39 +/- 0.21	98.58 +/- 0.05

In the first test we have changed only the learning rate onto best args. The results show a degradation of accuracy.
In other tests, we have changed the remaining parameters. In this case, the results obtained almost reached those of the paper. In fact, the confidance interval of the Class-IL overlaps the one obtained with the best args, unlike the C.I. of Task-IL are very close but do not overlap.