Cascaded Scaling Classifier: class incremental learning with probability scaling

Anonymous Authors[1]

Abstract

Humans are capable of acquiring new knowledge and transferring learned knowledge into different domains, incurring a small forgetting. The same ability, called Continual Learning, is challenging to achieve when operating with neural networks due to the forgetting affecting past learned tasks when learning new ones. This forgetting can be mitigated by replaying stored samples from past tasks, but a large memory size may be needed for long sequences of tasks; moreover, this could lead to overfitting on saved samples. In this paper, we propose a novel regularisation approach and a novel incremental classifier called, respectively, Margin Dampening and Cascaded Scaling Classifier. The first combines a soft constraint [1] and a knowledge distillation approach to preserve past learned knowledge while allowing the model to learn new patterns effectively. The latter is a gated incremental classifier, helping the model modify past predictions without directly interfering with them. This is achieved by modifying the output of the model with auxiliary scaling functions. We empirically show that our approach performs well on multiple benchmarks against well-established baselines, and we also study each component of our proposal and how the combinations of such components affect the final results.

1. Introduction

Building an agent capable of continuously learning over time from a stream of tasks remains a fundamental challenge. This ability is called Continual Learning (CL), a key property enabling autonomous real-world agents. Traditional machine learning models often struggle when faced

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

[1]We call it soft because we would like to be satisfied during the training. A hard one, instead, is a constraint we need to satisfy.
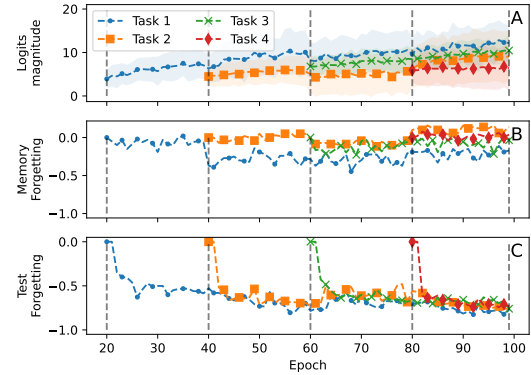

Figure 1. We train a ResNet-20 model with a simple rehearsal approach and a memory size of 500, on CIFAR10 divided into 5 tasks. A) Magnitude of ground truth logits on the replay samples; B) Forgetting on the replay samples; C) Forgetting on test scores for each past task.

with sequential and non-stationary data due to Catastrophic Forgetting (CF), which often leads to a drastic performance decrease due to the overwriting of past learned knowledge while learning new one (Parisi et al., 2019). The desired balance between remembering past learned patterns and correctly acquiring new knowledge is often called *stability-plasticity* trade-off (Wang et al., 2023).

The most straightforward way to mitigate CF is by using rehearsal-based approaches, which store and replay a small subset of previous task samples, but how such samples must be saved and retrieved is an open question. It is challenging to keep generally discriminative past samples because of overfitting that arises from replaying them (Verwimp et al., 2021; Prabhu et al., 2020; Buzzega et al., 2020; Bonicelli et al., 2022). In such scenarios, additional complexity comes from the intrinsically unbalanced nature of the training procedure, in which newer patterns have more samples than the ones in the memory. Consider the example shown in Fig. 1. Overfitting on rehearsal samples leads to a growth of their ground truth logits during the whole training procedure. At the same time, the forgetting on the test samples is higher than the one achieved on rehearsal samples. Intuitively, these two quantities should be similar, showing that this approach is not fighting the CF but only overfitting patterns present in the external memory. This is

evident by looking at scores achieved for tasks 3 and 4, in which forgetting on replay samples is nearly zero across all the training, and often becomes positive, while the accuracy on the corresponding test sets rapidly decreases. Despite such limitations, using past samples as training ones is one of the most used approaches to fight the CF.

Usually, a rehearsal approach is coupled with a regularisation strategy, in which past samples are used to calculate a regularisation term added to the training loss to alleviate CF further. In such scenarios, the true nature of the forgetting still needs to be understood. Some works claim it could arise from the classifier (Ahn et al., 2021; Wu et al., 2019; Zhao et al., 2020; Li & Hoiem, 2017), and most of the time the output of the model is directly regularised to mitigate this. The regularisation is usually carried out by forcing the classifier to produce the same output distribution as in the past only for rehearsal samples. This combination creates two losses that are in contrast, because training on past samples leads to overfitting while regularising on them forces the model to produce the same output as in the past. Such approaches usually require a careful selection of the hyperparameters to balance these two aspects. This behaviour is shown in Appendix A.

Motivated by these observations, we propose a novel CL approach composed of a regularisation term and a novel incremental classifier head. The former modifies the probabilities of previous classes up to a certain margin which satisfies a given constraint while, at the same time, regularising all the classes using a Knowledge Distillation (KD) approach calculated over the whole output of the model. The latter is a hybrid classifier composed of smaller task-wise classifiers, which are scaled and combined to produce the final output distribution. The combination of such components is capable of correctly regularising past learned knowledge without training on past samples, avoiding overfitting of such samples. At the same time, current patterns are correctly learned without interfering with past learned ones. To demonstrate the effectiveness of our proposals, we conduct comprehensive experiments on multiple CL benchmarks, as well as in-depth exploratory experiments to analyse the effectiveness of core design choices of both the regularisation approach, called Margin Dampening (MD), as well as the classification head, which we call Cascaded Scaling Classifier (CSC). For space reasons, an analysis of related works can be found in Appendix B.

2. Continual Learning definition

In a Class Incremental Learning (CIL) scenario, a generic model $f(\cdot)$ is trained on a sequence of classification tasks $\mathbf{T} = \{\mathcal{T}_i\}$, with $i = 1, \ldots, T$, having complete access only to the current training task t, with the additional possibility of storing a subset of past tasks' samples. A generic task

consists of a tuple $\mathcal{T}_i = (\mathcal{D}^i, \mathrm{Y}^t)$, where $\mathcal{D}^i = \{(x_i, y_i)\}$ is the dataset containing the samples and $y_i \in \mathrm{Y}^t$ is the label space of the task t, such that $\bigcap_{t=1,\ldots,T} \mathrm{Y}^t = \varnothing$; no other assumptions are made on the labels' space except that $|\mathrm{Y}^t| \geq 2$ for each task t.[2] Moreover, by definition, in a CIL scenario task identities are present only during training but not in the inference phase. This key difference makes CIL harder than another CL scenario called Task Incremental Learning (TIL), because having access to the task identities during inference allows for better separation between tasks, drastically reducing CF effects. For this reason, TIL can be easily solved using an architectural approach (Golkar et al., 2019; Wortsman et al., 2020) or even employing a regularisation approach without additional external memory (Pomponi et al., 2022).

Formally, we aim to fit a model $f : \mathcal{X} \rightarrow \mathcal{Y}^{\mathrm{T}}$, where $\mathcal{Y}^{\mathrm{T}} = \bigcup_{t=1,\ldots,T} \mathrm{Y}^t$, which minimises the expected risk:

$$f^* = \underset{f}{\arg\min} \underset{(x,y) \sim \bigcup_{i=1}^{\mathrm{T}} \mathcal{D}^i}{\mathbb{E}} \mathcal{L}(f(x), y) \qquad (1)$$

where $\mathcal{L}$ stands for the loss (e.g. cross-entropy loss) of correctly classifying sample x having label y using the model f. A generic model f used in a CL scenario can be further decomposed into a backbone $b(\cdot)$ and a classification head $h^t(\cdot)$, which can be conditioned to the task t,[3] such that $f^t(x) = h^t(b(x))$. No assumptions are made about the model, but usually, in a CIL scenario the backbone is given and its structure is fixed, while the output of the model is adapted each time a new task is retrieved. This is done by modifying the output of the classifier such that its output size is expanded to match the total number of classes, including the new ones; we also operate in this setting.

The goal of the minimisation process is to find the optimal model capable of minimising the loss over all the tasks. However, when training on a given task t we cannot access the whole past tasks $t' < t$, making this minimisation infeasible. To overcome this constraint while training on a task $t > 1$ we can use a regularisation function $\mathcal{R}$, an external memory $\mathcal{M}$ containing a small portion of past samples, or combine these two approaches, as such:

$$f^* = \underset{f}{\arg\min} \underset{(x,y) \sim \mathcal{D}^t \cup \mathcal{M}}{\mathbb{E}} \mathcal{L}(f(x), y) + \mathcal{R}(x, y) \qquad (2)$$

A well-crafted CL approach term prevents the model from forgetting past learned knowledge while leaving room for learning new patterns from the current task. This is an important trade-off which must be considered when operating in a CL scenario, and it is called *stability-plasticity* trade-off (Wang et al., 2023).

---

[2]We also assume that the classes are in a sequential order starting from zero, but this is just a scaling factor that has no impact on the scenario itself.

[3]If the classifier cannot be conditioned the task is ignored.

(a) Margin Dampening
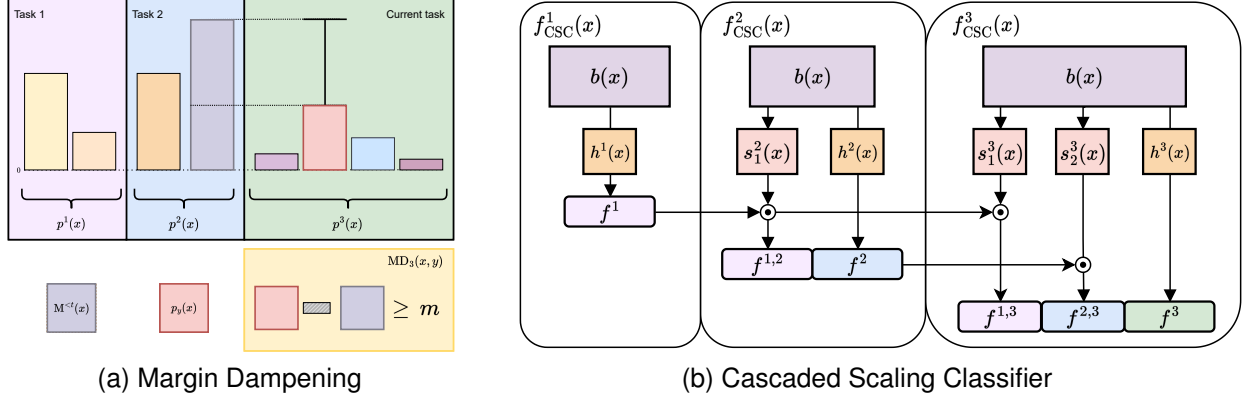
(b) Cascaded Scaling Classifier

Figure 2. On the left we show the probability dampening schema (MD, Section 3.1), in which past probabilities are decreased up to a certain margin, resulting in a regularisation approach based on a soft constraint. On the right, we visualise the cascaded procedure (CSC, Section 3.2), which combines task-wise outputs to build the final prediction, which in our case is the third. Better viewed in colours.

3. Proposed training schema

Our proposal is composed of a custom head classifier $h^t(\cdot)$ called Cascaded Scaling Classifier (CSC), and a rehearsal regularisation approach, named Margin Dampening (MD). Combining these two components produces a method capable of achieving a better stability-plasticity trade-off, leading to better results.

3.1. Margin Dampening (MD)

For a given tuple $(x, y) \in \mathcal{T}_t$, with $t > 1$, the core idea of our training schema is to decrease past probabilities while training only the output logits related to the current task, by forcing the maximum probability from past tasks to be lower than the ground truth probability, plus a safety margin. To this end, we first denote as $p^{1:t}(x) \in \mathbb{R}^{|\mathcal{Y}^t|}$ the vector containing the probabilities calculated over all classes seen up to the task t included. Using this formulation, we calculate the maximum past probability using the function $\mathrm{M}^{<t}(x) = \max p^{1:t-1}(x)$, and the probability associated with the true training class y as $p_y^{1:t}(x)$, and we want the minimization procedure to satisfy the following inequality:

$$p_y^{1:t}(x) - \mathrm{M}^{<t}(x) \geq m \qquad (3)$$

where m is the margin value which controls the safe distance between the two quantities. The constraint translates into the Margin Dampening (MD) loss, defined as:

$$\mathrm{MD}_t(x, y) = \max\left(0, \mathrm{M}^{<t}(x) - p_y^{1:t}(x) + m\right) \qquad (4)$$

which is summed to the cross-entropy loss calculated only over classes' probability belonging to the current task t, $p^t(x) \in \mathbb{R}^{|\mathrm{Y}^t|}$ [4]. The final loss while training on a task

[4] since the probabilities are calculated only over the current task, we have $\sum p^t(x) = 1$

$t > 1$ is:

$$\mathcal{L}_t(x, y) = \lambda \mathrm{MD}_t(x, y) + \mathcal{L}_{ce}(p^t(x), y) \qquad (5)$$

with λ being a scaling value to balance the two terms, controlling the stability-plasticity trade-off. This loss works in a bidirectional manner: at the same time, the correct probability grows due to the minimisation of the cross-entropy loss calculated only over current classes, without interfering with past classes, while past probabilities are pushed down up to a certain point which is defined by the margin. Past this point, the regularisation loss becomes zero, creating a smoother and easier-to-optimise loss function. The proposed regularisation approach can be visually inspected in Figure 2a.

In addition to the training loss just exposed, which is calculated only over samples from the current task, we use past samples to calculate a KD regularisation term over rehearsal samples, which enforces the stability of the model. In literature, the KD is divided, usually, into General KD (Zhao et al., 2020), which aggregates together logits belonging to the classes from all the previous tasks, and Task-wise approaches (Ahn et al., 2021), which treat classes within each task separately. Based on the intuition that newly added classes are randomly produced since the corresponding classifiers are randomly initialised, resulting in logits which are lower than the already trained ones, we propose a third way. Our regularisation term operates on the output of the model as a whole, without taking into account task boundaries. By doing so, we remove the necessity of training on past samples along with the ones from the current task, preventing the rise of overfitting of such samples.

To do that, we store a subset of past samples in an external memory $\mathcal{M}$ containing a fixed number of samples

3

during the whole training, which is adapted by removing random portions of past samples when new ones must be stored.[5] The samples are drawn in a class-balanced way and are used to regularise the output of the model using the Kullback-Leibler (Kl) divergence, as $\mathcal{R}(x) = \text{KL}(p^{1:t}(x) \,||\, \overline{p}^{1:t}(x))$, where $x \sim \mathcal{M}$, $\overline{p}^{1:t}(x) \in \mathbb{R}^{|\mathcal{Y}^t|}$. The latter can be seen as a teacher model with the same output as the model at the beginning of task t before starting the training process, but after adapting the classification layer to match the total number of classes.

Selection of the margin: regarding the choice of the margin, we rely on the observation that a CIL scenario is defined in such a way that the number of classes grows with the number of tasks, which implies that the maximum probability obtainable decreases after each task. Following this intuition, the margin m in Eq (4) must be scaled accordingly. To do that, we use a margin value which is based on how many classes have been encountered up to the current task t: $m^t = \frac{1}{|\mathcal{Y}^t|-1}$. By doing so, we avoid manually selecting the margin, which could result in either a disruptive or a negligible MD loss when not properly done. Not only does the margin value adapt while training, but it also allows the model to self-adapt based on the complexity of the training, removing the necessity to tune an additional hyperparameter.

Final loss: for a given tuple $(x, y) \in \mathcal{D}^t$ coming from the current task t, the final loss we propose is the following:

$$\mathcal{L}_t^{\text{MD}}(x, y, x_m) = \begin{cases} \mathcal{L}_t(x,y) + \mathcal{R}(x_m) & \text{if } t > 1 \\ \mathcal{L}_{ce}(p^1(x), y) & t = 1 \end{cases}$$

in which x_m are the samples drawn from the external memory $\mathcal{M}$ in a class-balanced way, $x_m \sim \mathcal{M}$, such that $|x_r| = |x|$.

3.2. Cascaded Scaling Classifier (CSC)

As opposed to standard heads used in literature, our approach is an ensemble of small task-wise heads, which are scaled and combined to produce the final output of the model. In this sense, our approach is a hybrid between a multi-head approach usually employed in TIL scenarios, and the single incremental one, which is the standard one used in the literature when dealing with CIL scenarios. The core idea is to help MD to decrease past classes' probabilities for samples coming from a new task, without operating directly on the associated classifier heads, allowing for more plasticity. To do that, past logits are scaled using additional scaling heads that operate as gating mechanisms.

When a new task $t > 1$ is retrieved, we not only add the corresponding task-wise head $h^t(b(x)) \in \mathbb{R}^{|Y^t|}$ into the classi-

fier module, but also a set of scaling functions as $S^t(x) = \{s_i^t(x) \in \mathbb{R}^{|Y^i|}\}$ for $i = 1, \ldots, t-1$. These additional modules are composed of a linear layer $g(\cdot)$, followed by a scaled Sigmoid such that $s_i^t(x) = \text{Sigmoid}(\gamma \cdot g(b(x)) + \beta)$, in which we set $\gamma = 1$ and $\beta = 10$.[6] Scaling the logits simplifies the regularisation of past classes without interfering directly with past heads. Mathematically, given the current training task index t and a past task $i < t$, the output associated to i is calculated as:

$$f^{i,t}(x) = f^i(x) \cdot \prod_{j=i+1}^{t} s_i^j(x) \tag{6}$$

with $f^{t,t}(x) = f^t(x)$. The final output of the model is produced by concatenating all such vectors:

$$f_{\text{CSC}}^t(x) = [f^{1,t}(x), f^{2,t}(x), \ldots, f^t(x)] \in \mathbb{R}^{|\mathcal{Y}^t|} \tag{7}$$

where t is the last task seen during the training procedure, which is set to the last one during the inference. Our final proposal is the combination of the just proposed head classifier, which can be visually inspected in Figure 2b, and the Margin Dampening regularisation proposed in Section 3.1.

4. Experimental Analysis

In this section, we define the experimental setup and show the results against well-established baselines. Then, we proceed with an in-depth analysis of our proposal and its components. The overall code is based on the Avalanche library (Lomonaco et al., 2021) and it is available on the official repository[7].

4.1. Experimental settings

Benchmarks: To evaluate our proposal, we use a variety of established CL benchmarks. Each one is built from a vision classification dataset, which is split into T disjoint sets, each one containing $\frac{C}{T}$ classes, where C is the number of classes in the original dataset.[8] Usually, in literature, the classes are grouped incrementally. However, how the classes are divided is crucial and can radically change the obtained results. To cover a wider spectrum of possible scenarios, as well as study the stability of each CL approach, we group the classes randomly each time a scenario is built, resulting in a different scenario with its complexity. We evaluate our proposal on three different datasets, used to create scenarios having growing difficulty, which are:

---

[5]When the training on a task is over, a random subset of it is saved into the memory.

[6]In the experimental section we study how these values affect the results.

[7]https://github.com/jaryP/Cascaded-Scaling-Classifier

[8]These subsets follow the rules exposed in Section 2.

	C10-5				C100-10				TyM-10			
Naive	18.62|-96.25				8.0|-82.4				6.2|-59.3			
Cumulative	86.8|-2.7				61.4|-3.4				30.0|-4.6			
Memory Size	200	500	1K	2K	200	500	1K	2K	500	1K	2K	5K
Replay	38.8|-69.4	50.2|-53.6	61.1|-38.8	61.1|-31.0	12.6|-74.0	17.6|-66.6	25.8|-56.1	32.6|-46.1	6.4|-51.9	7.0|-50.4	8.2|-46.8	12.2|-38.2
GEM (Lopez-Paz & Ranzato, 2017)	18.8|-88.9	20.8|-89.7	21.8|-82.2	22.3|-82.9	17.6|-68.1	18.4|-59.1	19.1|-54.3	–	8.1|-46.7	11.2|-41.1	–	–
DER (Buzzega et al., 2020)	43.1|-64.4	55.1|-48.4	63.7|-36.4	70.4|-26.1	12.7|-74.7	19.8|-64.7	27.4|-54.9	36.4|-43.2	5.8|-54.4	6.6|-51.7	8.9|-51.5	11.2|-49.2
GDUMB (Prabhu et al., 2020)	24.5|-24.2	31.1|-23.5	36.8|-26.2	45.2|-25.0	3.6|-47.8	6.7|-28.6	9.5|-14.7	14.1|-17.4	2.0|-4.7	3.2|-6.8	5.1|-9.0	9.8|-10.3
RPC (Pernici et al., 2021)	38.0|-70.2	50.4|-53.1	58.4|-42.2	69.6|-24.9	12.6|-73.8	18.5|-65.8	25.2|-56.7	32.2|-46.6	6.4|-52.9	7.0|-50.7	8.6|-46.6	11.8|-39.3
SS-IL (Ahn et al., 2021)	41.2|-25.4	45.6|-21.4	47.3|-21.2	52.1|-14.8	19.3|-34.1	24.2|-27.3	27.5|-20.7	28.8|-16.6	8.1|-5.0	8.2|-3.7	8.6|-3.7	8.5|-3.4
ER-ACE (Caccia et al., 2022)	51.5|-22.1	57.5|-17.5	63.4|-11.2	67.5|-6.7	19.4|-45.0	26.1|-37.3	31.2|-30.9	36.3|-24.1	10.2|-27.8	11.7|-24.9	12.6|-21.8	15.5|-18.7
ER-LODE (Liang & Li, 2023)	49.3|-14.4	56.9|+4.4	58.5|+13.1	60.4|+17.0	19.4|-40.6	28.5|-33.7	33.3|-20.4	38.2|-19.0	10.1|-34.4	11.1|-25.4	12.4|-23.7	15.6|-20.7
LD	55.9|-36.8	58.9|-32.4	57.9|-33.4	55.9|-32.5	19.6|-50.6	29.6|-30.7	30.1|-27.0	35.1|-19.3	6.9|-38.0	7.8|-14.4	10.1|-14.1	11.4|-17.1
MD-CSC (ours)	61.4|-27.9	65.5|-35.7	69.4|-28.3	75.68|-17.2	21.8|-53.9	30.3|-41.1	35.6|-21.3	41.3|-25.7	11.4|-29.1	13.8|-19.4	15.8|-23.7	18.8|-12.3

Table 1. The results obtained for each scenario, formatted as ACC($\uparrow$)|BWT($\downarrow$) averaged over 5 runs; the standard deviations are also shown. The best results for each combination of scenario and memory are highlighted. For a better comparison, the second best results are also highlighted. Better viewed in colors.

CIFAR10 (**C10**): it contains 10000 32×32 sized images, divided into 10 classes. Using this dataset we build the benchmark C10-5, in which we have 5 tasks, each one having 2 classes.

CIFAR100 (**C100**): the number and size of the images are the same as in CIFAR10, but we have 100 classes, resulting in fewer samples per class. Following the same approach used to create C10-5, we define C100-10.

TinyImageNet (**TyM**): it is a subset of 200 classes from ImageNet (Deng et al., 2009) which contains 64×64 sized images. Each class has 500 training and 50 testing images, and we used this dataset to build scenarios in which each task contains 20 classes, named TyM-10.

Architectures and training details: we use a ResNet-20 model (He et al., 2016) trained from scratch for all the experiments. To have easy-to-read comparisons, we trained all the models using the same number of epochs, batch size, and optimizer, which is SGD with a learning rate of 0.01 and a momentum of 0.8. For C10-5 we train each task for 20 epochs, using a batch size of 32. We train for the same number of epochs also for C100-10, but with a batch size of 16. Regarding TyM-10, we train the model for 30 epochs with a batch size of 32. Moreover, we use the same augmentation schema for each dataset (a random cropping, followed by a horizontal flipping with a probability of 0.5 and the normalization step), and no scheduling schemes or further regularisation methods are used.

For each combination of scenario and memory size, we run 5 experiments, each time incrementally setting the random seed (from 0 to 4), resulting, for each approach, in the same starting model and the same scenario, built up by randomly grouping the classes.

Hyperparameters: for each CL approach, we select the best set of hyperparameters through a grid search. The approaches are evaluated over a portion of the training data split (10%) which is used only for evaluation purposes.

The hyperparameters giving the best results on this split, after training on the train set in a CIL scenario, are the ones used in the final experimental evaluation. We do not rely on the hyperparameters selected in each paper to have better comparable results, all obtained under the same unified experimental environment. Each combination of approach, memory size, and scenario is evaluated once To have the same starting point, a single shared seed is used for all the experiments.. The evaluated hyperparameters for each approach, as well as the best ones, can be found in Appendix D.

Metrics: to evaluate the efficiency of a CL method, we use two widely-used metrics (Díaz-Rodríguez et al., 2018). The first one, called Accuracy (ACC), measures the final accuracy obtained across all the tasks' test splits, while the second one, called Backward Transfer (BWT), tells us how much of that past accuracy is lost during the training on upcoming tasks; both metrics are averaged over all tasks once the training on all of them is over. The first one tells us the final accuracy obtained on each task, while the latter how much of such information is lost after the training on a task is completed. Both metrics are important, and a trade-off must be achieved to balance plasticity (high accuracy on current task) and stability (low forgetting). However, even if both are important to compare CL approaches, the accuracy usually has a bigger weight in the overall evaluation. Appendix E contains the details on how such metrics are calculated.

Baselines: for a fair comparison we select only CL algorithms for which we can explicitly control the size of the external memory. The selected ones are: **Experience Replay** (Chaudhry et al., 2019; Buzzega et al., 2021), **Greedy Sampler and Dumb learner** (GDumb) (Prabhu et al., 2020), **Experience Replay with Asymmetric Cross-Entropy** (ER-ACE) (Caccia et al., 2022), **Separated Softmax for Incremental Learning** (SS-IL) (Ahn et al., 2021), **Gradient Episodic Memory** (GEM) (Lopez-Paz & Ranzato, 2017), **Dark experience replay** (DER)

(Buzzega et al., 2020), in its DER++ version, **Regular Polytope Classifier** (RPC) (Pernici et al., 2021), and **Rehearsal Loss Decoupling** (ER-LODE) (Liang & Li, 2023). In addition to such methods, we also propose a simple baseline called **Logits Distillation** (LD), in which an external class balanced memory contains images used to regularize the logits with the KL divergence (weighted with a constant α), without involving any classification loss over past samples. To do so, a copy of the model, used as the teacher model, is created at the beginning of the training and used to regularise the whole output, as in our main proposal Margin Dampening (Section 3.1). This baseline is useful for studying the effect of regularising the output without involving any classification loss over past samples. In addition, we also used Naive as the lower bound, in which the model is trained sequentially without any CF mitigation technique, and the upper bound called Cumulative, which trains the models on all the tasks up to the current one combined into a single training dataset. A more detailed overview of each baseline approach can be found in Appendix C.

4.2. Main results

By examining Table 1 we can observe that the simple Replay approach often achieves higher scores than more elaborated ones. It is clear when comparing Replay and GEM: the first reaches, on average, higher results than the latter, even when the BWT is worst. Moreover, it scales better with the memory size, while GEM seems to saturate once a limit is reached.[9]

This is also true if we compare Replay against methods presented as good to fight the class imbalance issue. This can be seen when comparing it with SS-IL and RPC. Such methods also saturate the results when the memory grows, and this happens because the approaches fight the CF by imposing very strong regularisation constraints leading to better BTW scores at the expense of training loss over the current tasks, which leads to lower accuracy scores. This shows that by strongly constraining the training procedure the final achieved accuracy is lower, but the past learned knowledge is well preserved. These results show that imposing strong regularisation terms prioritises stability at the expense of plasticity. Prioritising it could work when the number of tasks is limited, but its capacity to produce competitive results is negatively influenced when the number of tasks increases.

DER, which simultaneously regularises the logits and trains over past samples, achieves better results than the already-analysed approaches. It overcomes SS-IL in all the experiments and scales better with the memory and the

---

[9]Some GEM results are omitted due to the huge amount of time required to complete a training procedure.

Memory size	0.025	0.05	0.1	0.25	0.5	0.75	1
200	53.1	59.7	61.6	58.5	53.6	51.1	49.6
500	46.4	61.8	65.5	66.6	64.7	62.5	62.3
1000	48.4	54.5	64.5	71.3	69.6	67.7	71.6
2000	49.5	57.7	63.7	71.8	74.1	75.7	73.1
4000	42.5	52.9	63.0	71.2	75.0	74.4	75.6
6000	41.3	55.1	63.9	71.5	76.5	74.2	75.9
10000	39.7	50.1	64.9	73.0	76.0	74.6	76.8

Margin regularization

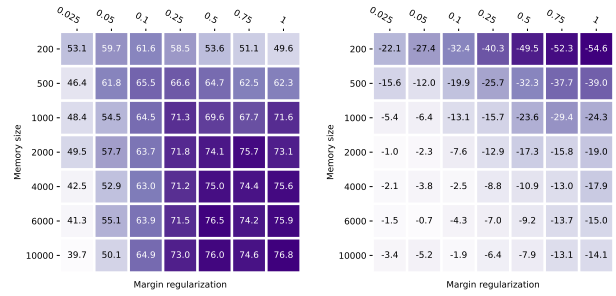| Memory size	0.025	0.05	0.1	0.25	0.5	0.75	1
200	-22.1	-27.4	-32.4	-40.3	-49.5	-52.3	-54.6
500	-15.6	-12.0	-19.9	-25.7	-32.3	-37.7	-39.0
1000	-5.4	-6.4	-13.1	-15.7	-23.6	-29.4	-24.3
2000	-1.0	-2.3	-7.6	-12.9	-17.3	-15.8	-19.0
4000	-2.1	-3.8	-2.5	-8.8	-10.9	-13.0	-17.9
6000	-1.5	-0.7	-4.3	-7.0	-9.2	-13.7	-15.0
10000	-3.4	-5.2	-1.9	-6.4	-7.9	-13.1	-14.1

Margin regularization

Figure 3. How the accuracy (left) and the BTW (right) scores are affected when varying the memory and the past margin regularisation. The results were obtained on ResNet-20 trained on C10-5.

scenario difficulty. However, DER constantly scores worse than a much simpler approach, which is ER-ACE. This approach is the second-best baseline overall, which is surprising due to the absence of a regularisation term. This tells us that a carefully studied method to reduce class unbalancing, rising from the usage of past samples as training ones, better fights the CF and that most of the regularisation approaches are just making the forgetting slightly slower by focusing on the stability, without taking into account the plasticity of the model. A similar approach, ER-LODE, which decouples the training loss into two terms, is the baseline achieving the best results when dealing with harder scenarios. By looking at the BWT scores obtained on C10-5, it is clear that it lacks plasticity, which is recovered when training on the next task when rehearsal samples are used for training. However, it performs better than ER-ACE in harder scenarios. This could be related to the proportional nature of the weights used to balance the decoupled terms.

Surprisingly, Logit Distillation (LD), overcomes many approaches when the scenario contains simple images (C-10 and C-100) but struggles otherwise. This shows that by just regularising the logits good results can be achieved, but when operating on harder scenarios a more sophisticated regularisation schema, which takes into account also the plasticity of the model, is needed.

In the end, our main proposal constantly reaches better results. When the scenario is easy, such as C-10, it overcomes AR-ACE by 10 percentage points when using a small memory (200) and, even if the gap is not preserved over all the experiments, the results are constantly better and scale better with the size of the external memory. To further understand how our proposal behaves, we proceed to an in-depth analysis of the components of the approach.

4.3. Proposal analysis

4.3.1. STABILITY-PLASTICITY TRADE-OFF

Here, we analyse how our approach controls the trade-off between plasticity and stability. The only parameter of our regularisation schema is λ, which is combined with the size of the memory to achieve the desired trade-off. Figure 3 shows the results of training a ResNet-20 on the C10-5 scenario.

Even when operating with small memories, it is possible to combine these values to achieve the best stability-plasticity trade-off. For example, combining a small memory (e.g. 200) with a high regularisation term (higher than 0.1) leads to bad results, since the memory does not contain enough samples to balance the forgetting. However, better results can be achieved when the regularisation term is 0.1 or lower, giving the best results for such a combination of memory size and the regularisation term, and showing that a good trade-off can be easily obtained. These findings suggest that increasing the size of the memory allows for a stronger regularisation term, and such a combination improves the results. These findings are confirmed by the results shown in Figure 3.

On the other hand, having a low regularisation term (e.g., 0.01) leads to a training schema focused on the stability of the model, which is incapable of properly learning current classes when using any memory size. Such combinations lead to a low BWT as well as a low accuracy, symptoms that current tasks are not learned properly due to the high focus on stability preservation.

In the end, such results show that, as opposed to other approaches, the choice of the regularisation term λ and the memory size affect the achievable results predictably. These two quantities are easy to balance, resulting in better results overall.

4.3.2. MD REGULARISATION EFFECTS

Figure 4 shows how the probabilities change when training the model using the proposed Margin Dampening regularisation approach. As we can see in the top image, the ground truth probabilities constantly increase while, at the same time, the maximum probabilities from past tasks decrease. The regularisation loss reaches zero after a few epochs, showing the regularisation ability of the proposed approach. Despite that, the maximum past value associated with current training samples keeps reducing even after the loss reaches zero (due to the soft constraint imposed by the Equation (4)). Such decreasing does not affect the output produced for past samples since the constraint imposed by the regularisation term is already respected, zeroing the regularisation term.
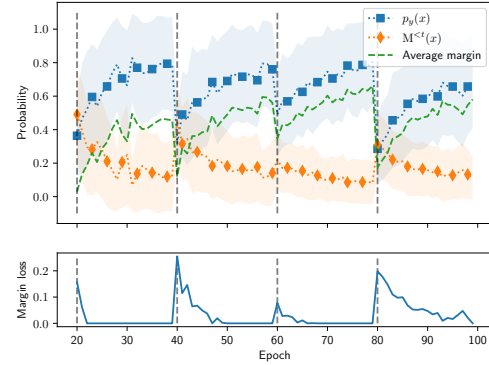

Figure 4. The figure (up) shows how the past logits, as well as the ground truth ones and the margin loss (down), vary during the training from the second task onward. Results were obtained on ResNet-20 trained on C10-5, with 500 samples saved in the memory and our proposal as the regularisation method.

4.3.3. REGULARISING FUTURE CLASSES

Using past samples as training ones seems important to avoid learning current classes Y^t using only current task samples, called positive samples, which inevitably leads to CF. Moreover, we know that using such samples to augment the training procedure could lead to overfitting, increasing forgetting. Instead, we advocate that training on such samples is unnecessary and that future classes, concerning a past class $t' < t$, can be easily regularised instead of trained, that improves both the plasticity and the stability. Here, we show this aspect by evaluating how the predictive ability of the model changes while training on newer tasks. Each time a new task t is collected, we calculate the distribution $p_0^t(x) \in \mathbb{R}^{|\mathcal{Y}^t|}$, for all $x \in \mathcal{D}_{\text{test}}^{t'}$ and for each $t' < t$. Then, we evaluate how such distributions change while training on t using the KL divergence between the current distributions and $p_0^t(x)$.

Figure 5 shows such results, obtained on C10-5 trained using a memory size of 500, for DER, Replay, LD, and our proposal. DER, which uses a fixed number of output classes, regularised and trained simultaneously, diverges. Such behaviour is unsurprising as the approach trains all the future classes using only positive samples and then tries to keep the output of the model fixed while training it, negatively impacting the stability. Consequently, it must strike a delicate and hard-to-tune balance to work efficiently. Even then, ER-ACE and ER-LODE achieve better results. Since we observe the same divergence when using Replay, we conclude that DER regularisation has a negligible effect on future classes. On the other hand, LD can preserve the model's predictive ability by keeping the divergence low during training. However, as seen in the main
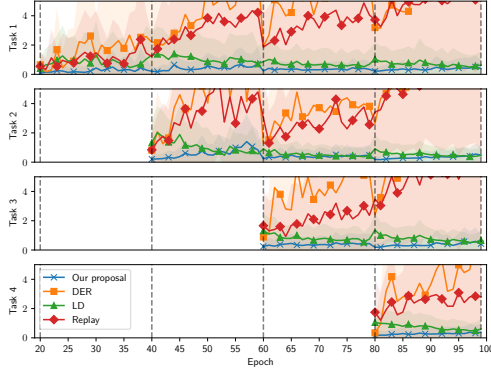
Figure 5. The images show the KL distance between the distributions predicted for past samples when a new task is collected and while training on it. The results are calculated over the whole test set of the C10-5 scenario, using the best hyperparameters and a memory size of 500.
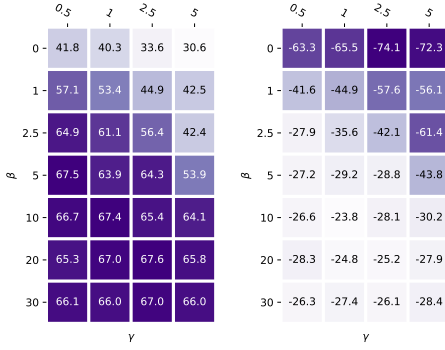

Figure 6. How the accuracy (left) and the BTW (right) scores are affected when varying the parameters of the scaler function proposed in Section 3.2. The results were obtained on ResNet-20 trained on C10-5 with a memory size set to 500.

results, it lacks plasticity, resulting in low scores. Instead, our main proposal removes the overfitting by not training on past samples and by using a regularisation schema that allows for more plasticity without detriment to the stability. Ultimately, our main proposal is the only rehearsal-regularisation approach capable of preserving the model's predictive capability while allowing for a higher degree of plasticity. These aspects combined lead to higher results overall, as already shown in the main results.

4.3.4. ON SCALING THE SIGMOID FUNCTION

In this section, we analyse how the scale (γ) and the offset (β) values used in the scaling function, proposed in Section 3.2, affect the results. Intuitively, a strong offset β is necessary, since a high value will preserve past logits by not gating them at the beginning of the training, creating a model which is capable of outputting a distribution that

|	C10-5				C100-10			
Memory Size	200	500	1K	2K	200	500	1K	2K
MD-CSC (ours)	61.41	65.52	69.48	75.68	21.82	30.34	35.63	41.37
CSC (sh)	58.01	62.92	66.96	69.4	19.13	27.69	34.07	38.72
I-MD	55.40	62.70	65.23	67.19	21.00	27.82	34.72	36.29
ER-ACE	50.31	56.30	62.51	67.19	19.50	26.02	31.10	36.33
CSC-ER-ACE	52.73	61.36	63.11	68.33	21.44	28.24	32.49	35.08

Table 2. The accuracies obtained on C10-5 and C100-10 when varying the components of the classifier layer. MD-CSC is our main proposal, while CSC (sh) and I-MD are, respectively, our approach when a single scaling head is used instead of the whole scaling architecture, and when the standard Incremental Head replaces CSC. CSC-ER-ACE is the standard ER-ACE coupled with the head we proposed. The best results for each combination of scenario and memory are highlighted.

resembles the one obtained by the model before starting the training on a new task. As shown in Fig. 6, for the C10-5 scenario using a memory size of 500, a low offset value leads to a lower accuracy, which is partially recovered when the scaling value is lower than 1. Overall, all the results stop improving when the offset value is higher or equal to 20. This is expected since a low offset does not guarantee that the model is regularised using the correct output distributions. Regarding the scaling value, which controls the smoothness of the Sigmoid curve, it behaves like a balance factor when the offset is below or equal to 2.5, but it becomes less impacting as the offset gets higher, having a negligible impact when the latter reaches 20. For this reason, in our experiment, we fixed the offset to 10 and the scale to 1.

4.3.5. CASCADED SCALING CLASSIFIER ABLATIONS

In this section, we analyse how much the proposed classifier CSC affects the results. To this end, we compare our proposal when the scaling functions in the set $S^t(x)$ are used (as in the main experiments) or not (which is, basically, an incremental classifier). Additionally, we use also a version of CSC in which the scaling head is just one and no cascaded component is used. Moreover, to understand if CSC could improve other approaches, we compare also the results achieved by ER-ACE when using or not the CSC classifier.

Table 2 contains the results of both experiments, obtained on C10-5 and C100-10 . Looking at the ER-ACE results, we can see that our approach is capable of marginally improving the accuracy with respect to the counterpart that uses the incremental head classifier, especially when the memory size is contained; this effect is less visible when its dimension grows. Regarding our approach, the results are worse when the logits are not scaled, regardless of the memory size. However, the results are always better than ER-ACE. Instead, when using a single scaling head, the
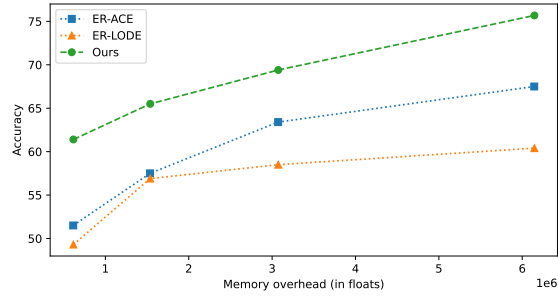
(a) C10-5 scenario.
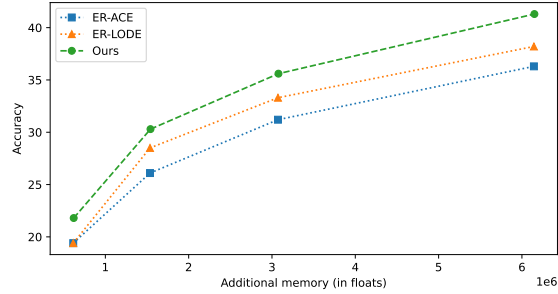
(b) C100-10 scenario.

Figure 7. The additional floats required by the approaches against the achieved accuracy. As additional floats, we count all the pixels in the memory, as well as additional parameters that a method requires. Results obtained by different baselines, as well as our approach, on C10-5 and C100-10 are displayed.

achieved results are competitive but lower than the ones obtained using multiple scaling heads. Intuitively, this happens because the scaling approach helps to mitigate the class unbalancing issue by giving the training procedure two ways to decrease past logits: by directly decreasing them or by decreasing the scaling value. This improves the plasticity without negatively affecting the stability, leading to better results. These aspects make our proposal competitive over all the benchmarks selected, with all possible combinations of memory size.

4.3.6. MEMORY OVERHEAD

Here, we compare the required additional memory of our proposal with other baselines. To do that, we count the number of additional floats (pixels plus parameters) that a method requires to mitigate CF, with respect to the model used in a Naive training approach. In all rehearsal approaches we used the additional floats are the number of pixels in the memory, while in our approach we also have all the additional parameters that CSC requires; the details on how such floats are calculated are present in Appendix F. The results are in Figure 7, which shows the required floats against the achieved accuracy for multiple methods.

The images show that our approach is capable of achieving better results than the others, using the same number of additional floats. This happens because the additional scaling heads used in CSC require, for the models we used, a negligible memory overhead when compared to the rehearsal memory.

5. Conclusion

We proposed a novel rehearsal-regularisation approach which combines a constraint-based regularisation schema and a scaled classifier head, which builds the final prediction vector using a cascaded approach. Combining these components creates a CL method which achieves better results than the baselines. Our approach takes advantage of a soft constraint, which allows for smoother regularisation and less forgetting, even when an external memory contains few samples per class. We also extensively evaluated our proposal, to understand how and why the components affect the results. In the future, we will delve more into the theoretical analysis of class unbalancing mitigation in CL, which we empirically observed and shown in the paper. We will also extend the approach for CL scenarios in which the task boundaries are not well defined, as well as Online CL scenarios.

6. Impact Statement

This paper presents work whose goal is to advance the field of Continual Learning. Although our work could have potential societal consequences, we feel that none of these must be specifically highlighted here.

References

Ahn, H., Kwak, J., Lim, S., Bang, H., Kim, H., and Moon, T. Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International conference on computer vision*, pp. 844–853, 2021.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. *Memory Aware Synapses: Learning What (not) to Forget*, pp. 144–161. Springer International Publishing, 2018.

Bang, J., Kim, H., Yoo, Y., Ha, J.-W., and Choi, J. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8218–8227, 2021.

Bonicelli, L., Boschini, M., Porrello, A., Spampinato, C., and Calderara, S. On the effectiveness of lipschitz-driven rehearsal in continual learning. *Advances in Neural Information Processing Systems*, 35:31886–31901, 2022.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Buzzega, P., Boschini, M., Porrello, A., and Calderara, S. Rethinking experience replay: a bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2180–2187. IEEE, 2021.

Caccia, L., Aljundi, R., Asadi, N., Tuytelaars, T., Pineau, J., and Belilovsky, E. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Chaudhry, A., Gordo, A., Dokania, P., Torr, P., and Lopez-Paz, D. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6993–7001, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Díaz-Rodríguez, N., Lomonaco, V., Filliat, D., and Maltoni, D. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.

Douillard, A., Cord, M., Ollion, C., Robert, T., and Valle, E. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference*, pp. 86–102. Springer, 2020.

Frascaroli, E., Benaglia, R., Boschini, M., Moschella, L., Fiorini, C., Rodolà, E., and Calderara, S. Casper: Latent spectral regularization for continual learning. *arXiv preprint arXiv:2301.03345*, 2023.

Gao, R. and Liu, W. Ddgr: Continual learning with deep diffusion-based generative replay. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Golkar, S., Kagan, M., and Cho, K. Continual learning via neural pruning. In *Real Neurons & Hidden Units: Future directions at the intersection of neuroscience and artificial intelligence @ NeurIPS 2019*, 2019.

Gomez-Villa, A., Twardowski, B., Yu, L., Bagdanov, A. D., and van de Weijer, J. Continually learning self-supervised representations with projected functional regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3867–3877, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, 2017. ISSN 1091-6490.

Lao, Q., Jiang, X., Havaei, M., and Bengio, Y. Continuous domain adaptation with variational domain-agnostic feature replay. *arXiv preprint arXiv:2003.04382*, 2020.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Liang, Y.-S. and Li, W.-J. Loss decoupling for task-agnostic continual learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Lomonaco, V., Pellegrini, L., Cossu, A., Carta, A., Graffieti, G., Hayes, T. L., De Lange, M., Masana, M., Pomponi, J., Van de Ven, G. M., et al. Avalanche: an end-to-end library for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3600–3610, 2021.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Mallya, A. and Lazebnik, S. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

Mundt, M., Pliushch, I., Majumder, S., Hong, Y., and Ramesh, V. Unified probabilistic deep continual learning through generative replay and open set recognition. *Journal of Imaging*, 8(4), 2022.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Pernici, F., Bruni, M., Baecchi, C., Turchini, F., and Del Bimbo, A. Class-incremental learning with pre-allocated fixed classifiers. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 6259–6266. IEEE, 2021.

Pham, Q., Liu, C., and Hoi, S. C. Continual learning, fast and slow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Pomponi, J., Scardapane, S., and Uncini, A. Structured ensembles: An approach to reduce the memory footprint of ensemble methods. *Neural Networks*, 144:407–418, 2021. ISSN 0893-6080.

Pomponi, J., Scardapane, S., and Uncini, A. Centroids matching: an efficient continual learning approach operating in the embedding space. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Pomponi, J., Scardapane, S., and Uncini, A. Continual learning with invertible generative models. *Neural Networks*, 164:606–616, 2023.

Prabhu, A., Torr, P. H., and Dokania, P. K. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 524–540. Springer, 2020.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Van de Ven, G. M., Siegelmann, H. T., and Tolias, A. S. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1): 4069, 2020.

Verwimp, E., De Lange, M., and Tuytelaars, T. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2021.

Wang, L., Zhang, X., Su, H., and Zhu, J. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*, 2023.

Wang, Z., Zhan, Z., Gong, Y., Yuan, G., Niu, W., Jian, T., Ren, B., Ioannidis, S., Wang, Y., and Dy, J. Sparcl: Sparse continual learning on the edge. *Advances in Neural Information Processing Systems*, 35:20366–20380, 2022.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 374–382, 2019.

Xu, Z., Tang, X., Shi, Y., Zhang, J., Yang, J., Chen, M., and Wei, X. Continual learning via manifold expansion replay. *arXiv preprint arXiv:2310.08038*, 2023.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

Zhai, J.-T., Liu, X., Bagdanov, A. D., Li, K., and Cheng, M.-M. Masked autoencoders are efficient class incremental learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19104–19113, 2023.

Zhao, B., Xiao, X., Gan, G., Zhang, B., and Xia, S.-T. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13208–13217, 2020.

A. Regularising and learning on past samples

We hypothesize that a hybrid approach which, at the same time, uses rehearsal samples as training samples as well as regularisation ones is sub-optimal. Figure 8 shows how the gradients diverge when using DER (Buzzega et al., 2020), when training Resnet-20 on C10-5, with a memory having a size of 500. The image shows the difference, calculated over the output channels, between the gradients obtained using the standard cross entropy and the one obtained for the knowledge distillation regularisation, both calculated over samples in the memory.

We can see that the method produces classification layer's gradients giving negative similarity, suggesting that the two terms of the training loss try to move the model towards two different directions: in the first one the model satisfies the cross entropy (with the risk of overfitting over past samples), while the second one tries to keep the logits fixed. Even if competitive results are achievable by carefully balancing the loss terms, we advocate that the final results are highly sub-optimal.
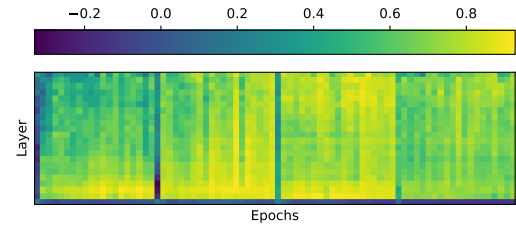

Figure 8. The cosine distance between gradients produced by the loss calculated on rehearsal samples and the ones related to the Knowledge Distillation regularisation. The last layer (last row of the heatmap) is the one which constantly has the two gradients pointing toward different directions.

B. Related work

Existing CL approaches can be mainly categorised into three categories, even if, most of the time, an approach can belong to multiple categories at the same time. Regularisation approaches (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Ahn et al., 2021; Chaudhry et al., 2021; Frascaroli et al., 2023; Gomez-Villa et al., 2022) introduce additional terms in the loss function to force the model to preserve knowledge that is crucial to keep solving past learned tasks while learning to solve newer ones. Architectural approaches (Rusu et al., 2016; Mallya & Lazebnik, 2018; Golkar et al., 2019; Douillard et al., 2020; Pomponi et al., 2021; Zhai et al., 2023; Wortsman et al., 2020) directly operate on the model itself, by isolating weights or dynamically expanding the model capacity. Rehearsal-based approaches implement an external memory containing examples from prior tasks and usually train the model jointly with the current task (Buzzega et al., 2020; Chaudhry et al., 2019; Prabhu et al., 2020; Pham et al., 2023; Wang et al., 2022; Xu et al., 2023; Bang et al., 2021; Pomponi et al., 2021). A subset of such methods is called Pseudo-rehearsal, in which a generative model takes the place of the memory (Van de Ven et al., 2020; Lao et al., 2020; Mundt et al., 2022; Gao & Liu, 2023; Pomponi et al., 2023). Although generative models are susceptible to issues like forgetting and mode collapse, they overcome the lack of diversity present in bounded memory buffers.

C. Baselines

In this section, we list all the baselines we compared our proposal with. The selected baselines are:

Naive: it is used as the lower bound, and it trains the models on the tasks without any strategy to fight the CF.

Cumulative: this approach cumulatively trains the models using a dataset which is built up by merging all the tasks up to the current one. It represents the upper bound since all the training samples are saved into the external memory.

Experience Replay (Chaudhry et al., 2019; Buzzega et al., 2021): it is a simple rehearsal approach, in which samples from the memory are used along with the ones from the current dataset to train the model. It has a fixed-sized memory populated with samples from the current task once it is over, discarding past samples to make room for newer ones.

Greedy Sampler and Dumb learner (GDumb) (Prabhu et al., 2020): it was proposed for questioning the advantages of CL, and it simply avoids training the model when a new task is collected, which is just used to fill up the memory. The just-filled memory is then used to train a new model from scratch when needed.

Experience Replay with Asymmetric Cross-Entropy (ER-ACE) (Caccia et al., 2022): initially proposed for contrasting CF in an Online CL scenario, it was also adapted for CIL. It uses a disjointed cross-entropy loss to leverage class unbalancing.

Separated Softmax for Incremental Learning (SS-IL) (Ahn et al., 2021): it mitigates the CF using Knowledge Distillation on a task-wise basis, in addition to a modified cross-entropy loss to learn patterns from the current task.

Gradient Episodic Memory (GEM) (Lopez-Paz & Ranzato, 2017): it uses the external memory to calculate gradients associated with past tasks and uses them to move the current one in a region of the space that satisfies both the current task as well as past ones. It does so by minimizing a quadratic problem, whose computational complexity scales exponentially with the number of samples in the memory.

Dark experience replay (DER) (Buzzega et al., 2020): the model is regularised by, at the same time, augmenting the current batch using past samples and regularizing the logits using the MSE distance loss between the logits obtained using the current model and the ones from the past model. The approach implements a classifier with a fixed number of classes, trained all at the same time.

Regular Polytope Classifier (RPC) (Pernici et al., 2021): the idea is to fight the CF by using a fixed number of equidistant and not learnable classifiers, and to learn only the backbone. To avoid setting the number of classes in advance we add a projection layer, which produces a vector containing 1000 features, between the backbone and the classifier layer, fixing the number of learnable classes to 1001.

Loss Decoupled (ER-LODE) (Liang & Li, 2023): it decouples the classification loss creating two components conditional on whether the samples are from the current task or not. It uses an external memory and combines the losses by weighting them using a scaling factor which depends on the number of seen classes.

Logits Distillation (LD): we also propose a simple baseline called Logits Distillation, in which an external class-balanced memory contains images used to regularize the logits with a KL distance (weighted by a constant α). To do so, a copy of the model, used as the teacher model, is created at the beginning of the training. Moreover, we regularize future logits as in our proposed method (Section 3.1). This baseline is useful for studying the effect of future regularisation without involving any classification loss over past samples.

D. Hyperparameters selection

Dataset	Method	Memory size				
		200	500	1000	2000	5000
C10-5	DER	$\alpha = 0.1, \beta = 0.5$	$\alpha = 0.5, \beta = 0.8$	$\alpha = 0.5, \beta = 0.1$	$\alpha = 0.2, \beta = 0.8$	
	LODE	$\rho = 0.2$	$\rho = 0.5$			
	LD	$\alpha = 0.1$				
	MARGIN	$\lambda = 0.1$	$\lambda = 0.25$		$\lambda = 0.5$	
C100-10	DER	$\alpha = 0.1, \beta = 0.8$	$\alpha = 0.1, \beta = 1$	$\alpha = 0.1, \beta = 0.8$		
	LODE	$\rho = 0.1$	$\rho = 0.2$		$\rho = 0.5$	
	LD	$\alpha = 1$		$\alpha = 0.75$		
	MARGIN	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 0.25$	$\lambda = 1$	
TyN-10	DER	$\alpha = 0.1, \beta = 0.5$	$\alpha = 0.1, \beta = 0.1$	$\alpha = 0.1, \beta = 0.5$	$\alpha = 0.1, \beta = 0.2$	
	LODE	$\rho = 0.1$			$\rho = 0.2$	$\rho = 0.5$
	LD	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 1$		$\alpha = 0.75$
	MARGIN	$\lambda = 0.05$			$\lambda = 0.25$	

Table 3. The best combination of hyper-parameters for each approach.

To select the best hyperparameters, we fixed the number of epochs, the batch size, and the optimizer parameters. Then, we trained a model for each combination of hyperparameters, and we selected the best set by evaluating the results on a development split containing 10% of the training samples. For each method, the hyperparameters are:

- DER (Buzzega et al., 2020): this approach has two hyperparameters. The first controls the classification loss of past samples, $\alpha : [0.1, 0.2, 0.5, 0.8, 1.0]$, while the second controls the logits distillation loss $\beta : [0.1, 0.2, 0.5, 0.8, 1.0]$.

- LODE (Liang & Li, 2023): the only parameter ρ controls the weight of the proportion between the number of new classes over the number of past ones. The resulting value is used to weigh one of the losses. Based on the findings shown in the paper, the search space is $\rho : [0.1, 0.2, 0.5, 0.8, 1]$.

- LD: it has just one parameter, which controls the strength of the logit regularisation. The search space is $\alpha : [0.1, 0.25, 0.5, 0.75, 1]$

- MD: our proposal has only one hyperparameter, which controls the strength of margin regularisation, and the search space is $\lambda : [1, 0.5, 0.25, 0.1, 0.05, 0.025, 0.01]$.

The best parameters, for each combination of scenario and memory, are shown in Table 3.

E. Metrics

To evaluate the efficiency of a CL method, we use two different metrics proposed in (Díaz-Rodríguez et al., 2018). The first one, called Accuracy, shows the final accuracy obtained across all the tasks' test splits, while the second one, called Backward Transfer (BWT), measures how much of that past accuracy is lost during the training on upcoming tasks. To calculate the metrics, we use a matrix $\mathbf{R} \in \mathbb{R}^{N \times N}$, in which an entry $\mathbf{R}_{i,j}$ is the test accuracy obtained on the test split of the task j when the training on the task i is over. Using the matrix $\mathbf{R}$ we calculate the metrics as:

$$\text{ACC} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{R}_{N,j} \qquad\qquad \text{BWT} = \frac{\sum_{i=2}^{N} \sum_{j=1}^{i-1} (\mathbf{R}_{i,j} - \mathbf{R}_{j,j})}{\frac{1}{2}N(N-1)} .$$

Both metrics are important to evaluate a CL method since a low BWT does not imply that the model performs well, especially if we also have a low Accuracy score because, in that case, it means that the approach regularizes too much during the training, leaving no space for learning new tasks. In the end, the combination of the metrics is what we need to evaluate.

F. Memory overhead

As additional parameters, we consider the ones that must be stored in addition to the base model and the tasks' head.

For a generic rehearsal approach, the number of additional floats to store is given by $O = W \times H \times C \times |\mathcal{M}|$, in which (C, H, W) are the sizes of the images in the dataset and $\mathcal{M}$ is the rehearsal memory.

For our proposal, we need to also count the additional parameters in the classifier head CSC, which are added to O. Having the input of the head size equal to I and the current task T, we have:

$$O_{CSC}^{T} = O + \text{I} \times \sum_{i=1}^{T} (i-1)|\text{Y}^i| \tag{8}$$

where $|\text{Y}^i|$ is the number of classes in task i. For example, when measuring O_{CSC}^{T} at the end of the final C10-5 task, we have that $O_{CSC}^{T} = O + \text{I} \times 2 \times 15$, since $|\text{Y}^i| = 2$ for each task.

G. CSC time overhead

In this section, we analyse the time overhead which the proposed head CSC has. To do that, we generated 20 virtual tasks, meaning that no real images are used, but the growing number of tasks is simulated to adapt the models. To measure the time, in milliseconds, we used the Event functionality in the Pytorch framework [10]. To have better statistics, we simulate a forward pass in the whole model, and multiple forward (100) passes into the head we are testing. This process is repeated for each task and the average elapsed time is saved.

Figure 9 shows that, for our approach, such time indeed grows, but this growth is linear with the number of tasks and not quadratic as the number of heads. Moreover, being the time in milliseconds, we can say that the time overhead introduced by our proposal is contained.
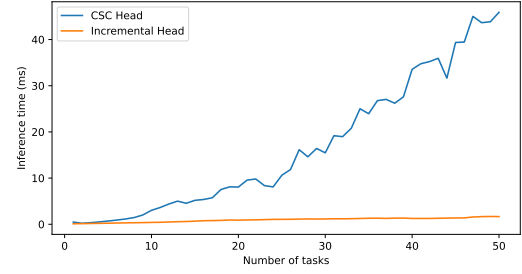

Figure 9. The empirically estimated inference time for our proposal CSC and the standard Incremental Head when varying the number of classes.