Calibration of Continual Learning Models

Lanpei Li*
University of Pisa and ISTI-CNR
lanpei.li@isti.cnr.it

Elia Piccoli*
University of Pisa
elia.piccoli@phd.unipi.it

Andrea Cossu
University of Pisa
andrea.cossu@di.unipi.it

Davide Bacciu
University of Pisa
davide.bacciu@unipi.it

Vincenzo Lomonaco
University of Pisa
vincenzo.lomonaco@unipi.it

Abstract

Continual Learning (CL) focuses on maximizing the predictive performance of a model across a non-stationary stream of data. Unfortunately, CL models tend to forget previous knowledge, thus often underperforming when compared with an offline model trained jointly on the entire data stream. Given that any CL model will eventually make mistakes, it is of crucial importance to build calibrated *CL models: models that can reliably tell their confidence when making a prediction. Model calibration is an active research topic in machine learning, yet to be properly investigated in CL. We provide the first empirical study of the behavior of calibration approaches in CL, showing that CL strategies do not inherently learn calibrated models. To mitigate this issue, we design a continual calibration approach that improves the performance of post-processing calibration methods over a wide range of different benchmarks and CL strategies. CL does not necessarily need perfect predictive models, but rather it can benefit from reliable predictive models. We believe our study on continual calibration represents a first step towards this direction.*

1. Introduction

In offline machine learning, models learn from a fixed data distribution and they are tested on new examples from the same distribution (the *iid* assumption). Unfortunately, machine learning models never achieve perfect predictive accuracy unless the task is very simple or created ad-hoc.
If a perfect predictive model is unrealistic for offline machine learning, it is even more unlikely in Continual Learning (CL) [15], where the model learns over time from a sequence of non-stationary data distributions. Due to the forgetting phenomenon [9], the predictive performance of a CL model can degrade as the model is trained on new distributions.
So far, most of the efforts in CL have been dedicated to designing approaches that mitigate forgetting and increase predictive performance [7]. While these remain fundamental challenges for the advancement of CL research, they also mainly aim at reducing the gap with respect to a perfect predictive model. We do not expect this gap to be ever fully closed, hence *we need to learn how to deal with imperfect models that make mistakes*.

The objective of this paper is to understand how to build CL systems that can be trusted. For example, being able to tell *in advance* when a model might be wrong can make applications more robust and reliable: a user could discard predictions that do not match a predefined level of trust and only accept those that are marked as safe by the model itself (as in the learning to reject paradigm [6]). To this extent, we leverage the *calibration* paradigm, a well-known research topic in machine learning that aims at learning a proper *confidence* measure related to the predictions of a model [10, 21, 26].
Intuitively, the confidence tells how likely the model is, on average, to provide a correct answer on a given example. For this reason, calibrated models are extremely useful in many practical scenarios, from finance and healthcare to computer vision and robotics. The more autonomous the application, the more risky it is to rely on the predictions of uncalibrated models. Although it could be of extreme use for practical purposes, calibration is currently disregarded in CL (with the notable exception of a brief mention in [3, 4], that we also consider in our work).
We believe calibration to be a fundamental challenge for CL as it is unlikely to achieve reliable CL models for real-world applications without a strong notion about their robustness (Figure 1). Here, we provide our contribution towards calibrated CL models:

*Corresponding authors.

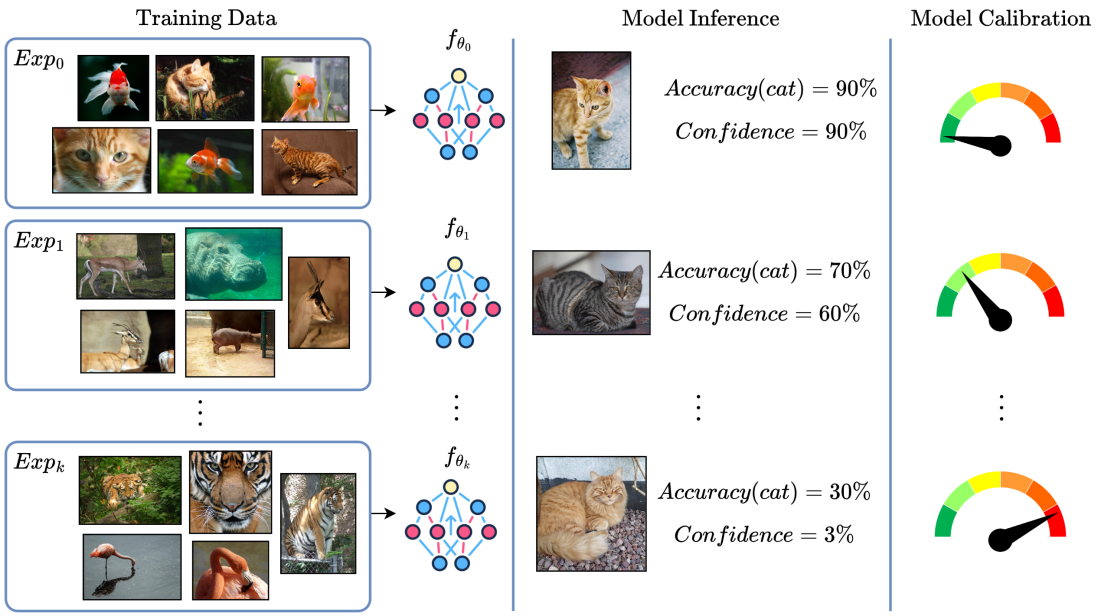

Figure 1. A CL model f_θ is trained on a sequences of k experiences (or tasks). The model accuracy on the class "cat" decreases over time. Its confidence decreases much faster. Therefore, the model becomes less calibrated over the course of its learning phase. A calibrated CL model, which is the objective of this paper, should output a confidence which is equal to the average accuracy. A calibrated model knows what to expect, on average, as a result of its predictions.

1. We ran extensive experiments across 4 CL benchmarks, 3 CL strategies, and 5 calibration methods. We include both popular CL benchmarks as well as others datasets aimed at testing calibration in scenarios beyond computer vision (supervised action prediction from Atari) and real-world scenarios (land use detection from satellite images). To the best of our knowledge, this is the first comprehensive study on continual calibration.
2. We discovered that calibration methods only partially work when applied to non-stationary data streams. Even when equipped with CL strategies, the resulting models are not necessarily well calibrated, especially when compared with the same model trained offline on the entire data stream.
3. We design Replayed Calibration, a continual calibration method that is compatible with a large family of calibration approaches (the post-processing calibration approaches introduced in Section 2.1). Our approach improves the performance of calibration methods by large margins.

2. Calibration background

Calibration has been studied for the offline machine learning setup [24, 26]. We provide a brief overview of calibration mainly intended for continual learning researchers who are interested in applying or studying calibration.

We focus on the calibration of neural network models trained on supervised classification tasks [10]. Most of this discussion generalizes to other types of models as well.

A model f_θ parameterized by $\theta \in \mathbb{R}^d$ is trained on a dataset $\mathcal{D} = \{(x_j, y_j)\}_{j=1,\dots,M}$, where each example is composed by an input-target pair (x_j, y_j) and the target represents the class associated with the input. For each input example x_j the model returns a probability vector $\hat{y}_j$, containing one probability per class, *and* a confidence value $\hat{c}_j \in \mathbb{R}$. Formally, $\hat{y}_j, \hat{c}_j = f_\theta(x_j)$. The probability vector is obtained by passing the logits z_j through a softmax function: $\hat{y}_j = \text{softmax}(z_j)$. The logits are computed by the last layer of the model $z_j = h_\theta(x_j)$, where h_θ computes the pre-softmax output. The model is trained by minimizing a loss function $\mathcal{L}(\hat{y}, y)$ (e.g., cross-entropy).

Definition 1. A model f_θ is *calibrated* when $P(\hat{y} = y|\hat{c} = c) = c$, $\forall c \in [0, 1]$.

Definition 1 states that a model is calibrated when the probability of predicting the correct class is equal to the confidence, for any given value of the confidence. The calibration objective cannot be computed exactly since the joint distribution $P(\hat{Y}, \hat{C})$ is taken over the predictions and confidence random variables, respectively, which are continuous variables. Given a dataset $\mathcal{D}$ with M examples, we use the Expected Calibration Error (ECE) and the reliability

diagrams as approximations of the calibration objective of Definition 1. The reliability diagram reports the histogram of accuracy against confidence. The histogram collects all M model predictions and confidence values. Then, it partitions the predictions in K equally-spaced bins based on the confidence value. It finally computes the average accuracy of the predictions within each bin.

Definition 2. A reliability diagram with K equally-spaced confidence bins reports the average accuracy over a generic bin I_b as $\overline{a}_b = \frac{1}{|I_b|} \sum_{c_j \in I_b} \mathbb{1}(\hat{y}_j = y_j)$. Correspondingly, the average confidence within a bin I_b by $\overline{c}_b = \frac{1}{|I_b|} \sum_{\hat{c}_j \in I_b} \hat{c}_j$.

A perfectly calibrated model would return a reliability diagram equal to the identity function: $\overline{a}_b = \overline{c}_b, \ \forall I_b$. We use reliability diagrams in our experiments (see Figure 6 for an example). The distance from a perfectly calibrated model is computed by the Expected Calibration Error, which summarizes the information contained within a reliability diagram in a single value.

Definition 3. The Expected Calibration Error (ECE) for a given model on a dataset $\mathcal{D}$ is computed as ECE $= \sum_{b=1}^{K} \frac{|I_b|}{M} |\overline{a}_b - \overline{c}_b|$, where M is the total number of examples in $\mathcal{D}$.

Notice how ECE is a scalar metric between 0 and 1, hence it can be reported as a percentage value, with 0% representing a perfectly calibrated model and 100% its opposite, not calibrated counterpart.

2.1. Calibration of neural networks

Although calibration has been studied for years in machine learning [26], there are only a few techniques available that are compatible with neural networks *and* multi-class classification tasks (with more than 2 classes) [10, 21].
For this paper, we follow [26] and divide calibration methods into two main families: *post-processing calibration methods* and *self-calibration methods*.
Post-processing calibration methods are applied after the model training phase and they rely on a held-out validation set to tune or learn some calibration (hyper)parameters. The same validation set can also be used for model selection. Many post-processing calibration methods are available for binary classification tasks. Since in a CL environment, new classes often appear over time, it is unrealistic to consider binary classification tasks. Therefore, we focus on existing extensions to the multi-class case.
Self-calibration methods operate directly during model training, without requiring a separate calibration phase.

Temperature scaling (TS). TS [10] is a post-processing calibration method that adapts the softmax temperature applied after the output layer to compute "softer" probability distributions. Peaked distributions are often associated with over-confidence in the prediction. TS computes the confidence on an example x_j as $\hat{c}_j = \max \text{softmax}(\frac{z_j}{T})$, where the logits are divided by the scalar temperature T and the maximum is computed across the resulting probability vector after the softmax. The temperature is learned by minimizing the Negative Log Likelihood on the validation set, which is associated with the entropy and therefore measures how peaked the distribution is. Since TS only changes the temperature T, the output classes predicted by the model remain the same before and after the calibration phase.

Matrix/Vector scaling. Matrix scaling (MS) and Vector scaling (VS) [10] are post-processing methods that learn an additional linear projection parameterized by W, b during the calibration phase. The model predictions on a generic example x_j are updated as $\hat{y}_j = \text{softmax}(W z_j + b)$ and the confidence is obtained by $\hat{c}_j = \max \hat{y}_j$ (like TS, the maximum is computed across the probability vector returned by the softmax). The parameters W, b are optimized with respect to the Negative Log Likelihood on the validation set. In MS, W is any matrix, while in VS W is a diagonal matrix (for efficiency purposes).

Entropy regularization (HR). Instead of promoting high-entropy distributions via post-processing methods like TS and MS/VS, HR [21] operates directly during model training. The loss used at training time is augmented with a regularization term of the form $-\lambda H(\hat{y}_j)$, where H is the entropy of the probability distribution computed by the model on x_j. The optimization process strives to minimize the loss, hence to maximize the entropy (and prevent peaked distributions). The regularization is controlled by the hyperparameter λ.

3. Continual Calibration

Our objective is to i) understand how to apply calibration methods in a CL setup, ii) assess the behavior of calibration approaches on non-stationary data streams and iii) extend existing approaches backed by intuitions from CL strategies. Figure 2 provides a compact representation of all three points.

Calibration of CL models is especially challenging, since the data distribution faced during training changes over time. All the calibration methods we presented in Section 2.1 are designed for a data distribution that does not change between the training and the calibration phase. In CL, we have a stream of experiences (or tasks) $\mathcal{S} = (e_1, e_2, \dots)$ [16]. Each experience e_i contains a dataset $\mathcal{D}_i$ with M_i examples: $\mathcal{D}_i = \{(x_j, y_j)\}_{j=1,\dots,M_i}$. We are still considering the supervised classification setup for CL. The stream $\mathcal{S}$ be-

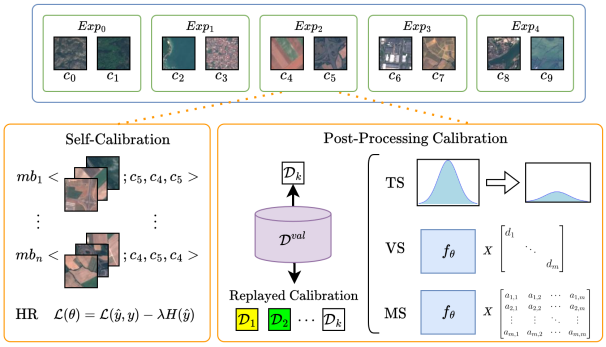

Figure 2. Continual calibration is performed on a stream of experiences (top) by applying either self-calibration (bottom left) or post-processing calibration (bottom right). Self-calibration approaches like Entropy Regularization (HR) regularize the training loss at each minibatch. Post-processing calibration like Temperature Scaling (TS) and Matrix/Vector scaling (MS/VS) are applied only at the end of each experience. Our Replayed Calibration approach is applicable alongside any post-processing methods.

comes available over time and the model is continuously trained on each experience sequentially. Importantly, the data distribution changes between one experience and the other, making the stream non-stationary [8]. Since the content of each experience cannot be entirely stored for later reuse, the model needs to learn new experiences without forgetting previous ones. That is, the predictive performance on previous experiences should not decrease.

Self-calibration techniques like HR are already compatible with a CL setup since they do not require a separate calibration phase. Post-processing calibration methods, instead, only operate *at the end* of the training phase. While this makes sense in an offline learning setup, where all data is available at once, post-processing calibration methods are not directly applicable in CL, where the model could be potentially trained on an infinite sequence of experiences.

Post-processing continual calibration. When considering finite data streams, one possibility would be to apply the post-processing calibration method only once at the end of training on all experiences. Unfortunately, since in CL we cannot store the entire content of previous experiences, the post-processing calibration would be applied only to the validation set associated with the last experience. Therefore, the model would not be calibrated on any examples coming from previous experiences.

Instead, we add a calibration phase at the end of training on each experience. Calibration is performed on the validation set $\mathcal{D}^{\text{val}}$ associated with the current experience, where $\mathcal{D}_i = \mathcal{D}_i^{\text{train}} \cup \mathcal{D}_i^{\text{val}}$. The test set $\mathcal{D}_i^{\text{test}}$ associated with each experience is assumed to be always available. The test sets are never used neither to train the model nor to calibrate it, but only for evaluation purposes.

The post-processing calibration methods we considered either add a new layer after the original classifier (VS, MS) or they change the default softmax temperature from 1 to a learned value (TS). In our CL setup, these changes are first introduced after training on the first experiences. To comply with the CL setup, in the following experiences we did not revert the changes made by post-processing calibration and we train continuously the resulting CL model (either with an extra output layer or with a learned temperature).

Replayed Calibration (RC). Our adaptation of post-processing calibration for CL does not completely solve the issue of calibrating on an incomplete portion of the data. During each calibration phase, the model only sees data coming from the current experience. Therefore, when a new experience arrives (a new data distribution), we have no guarantee that the previously calibrated model will remain calibrated on previous distributions. We extend post-processing calibration methods with CL approaches based on replay [11]. Many CL applications allow to store a (small) subset of previous data. Usually, replay techniques leverage the external buffer at training time by training the model on data coming from the current experience and from the memory buffer, to improve model stability and mitigate forgetting. Inspired by this approach, we do the same during the calibration phase. The external memory buffer contains examples from the validation sets of previous experiences. The model is then calibrated on both the content of the buffer *and* the validation set of the current experience. We call this post-processing calibration approach *Replayed Calibration (RC)*. RC can be combined with any of the existing post-processing calibration methods.

3.1. Empirical evaluation

We study calibration of CL models trained with Naive finetuning, Experience Replay [23] and Dark Experience Replay (DER) [4], in its DER++ version[1]. Naive simply trains the model continuously over the data stream, minimizing the classification loss. Experience Replay keeps a fixed-size buffer in which to store examples from previous experiences. We use reservoir sampling to fill the buffer. DER++ is a state-of-the-art CL method that combines replay and distillation. In addition to input-target pairs, the replay memory M of DER++ also stores the logits computed by the model when the example was first added to the memory. The distillation loss is computed on examples sampled from the memory and it reads $\alpha\ \mathbb{E}_{(x,z)\sim M}\|z - h(x)\|_2^2 + \beta\ \mathbb{E}_{(x',y')\sim M}\mathcal{L}(\hat{y}, y')$, where for simplicity $\hat{y}$ in the last term denotes the model prediction computed on x'.

[1]The code to reproduce the experiments is available at https://github.com/lilanpei/Continual-Calibration and as supplementary material.

Table 1. Average accuracy and standard deviation on the test set of all experiences computed at the end of training. Bold highlights the best result for each CL strategy and Joint Training. Bold and underline highlights overall best across CL strategies.

Accuracy (%)	***Split MNIST***	***Split CIFAR100***	***EuroSAT***	***Atari***
Joint	93.00 ± 1.08	64.37 ± 7.22	91.48 ± 4.05	55.10 ± 0.61
HR	$\mathbf{94.91 \pm 1.10}$	62.20 ± 4.76	94.51 ± 4.26	54.82 ± 0.71
TS	93.92 ± 0.71	64.67 ± 3.18	95.12 ± 3.75	$\mathbf{55.38 \pm 0.42}$
VS	94.21 ± 1.76	$\mathbf{68.30 \pm 5.92}$	93.74 ± 3.34	55.32 ± 0.30
MS	94.24 ± 3.32	62.90 ± 2.90	$\mathbf{95.31 \pm 4.69}$	39.27 ± 2.04
DER++	92.74 ± 0.38	35.18 ± 2.86	77.39 ± 8.44	32.35 ± 0.17
HR	92.56 ± 0.39	37.41 ± 2.70	79.86 ± 2.76	32.09 ± 0.34
TS	$\underline{\mathbf{94.79 \pm 0.21}}$	32.75 ± 10.43	77.11 ± 1.21	$\underline{\mathbf{33.14 \pm 0.67}}$
VS	91.92 ± 0.34	23.30 ± 2.56	70.60 ± 2.65	25.19 ± 2.75
MS	91.95 ± 0.19	19.01 ± 10.98	53.20 ± 18.72	25.03 ± 3.74
TS + RC	94.74 ± 0.20	$\mathbf{41.79 \pm 0.84}$	$\mathbf{80.81 \pm 4.34}$	33.03 ± 0.87
VS + RC	92.24 ± 0.27	34.26 ± 6.08	57.75 ± 21.32	26.21 ± 1.42
MS + RC	92.23 ± 0.02	37.30 ± 5.82	75.07 ± 3.38	25.56 ± 2.16
Replay	90.97 ± 0.66	40.39 ± 4.00	80.57 ± 1.06	29.30 ± 0.66
HR	90.71 ± 0.63	40.36 ± 1.38	81.03 ± 0.74	$\mathbf{29.87 \pm 0.50}$
TS	$\mathbf{94.20 \pm 0.55}$	23.22 ± 7.56	78.02 ± 0.67	28.29 ± 0.51
VS	75.31 ± 2.31	35.00 ± 2.90	73.08 ± 2.39	28.45 ± 0.19
MS	74.18 ± 5.54	34.73 ± 3.33	79.37 ± 4.43	28.98 ± 0.38
TS + RC	93.87 ± 0.81	15.26 ± 8.59	81.73 ± 0.76	29.03 ± 0.42
VS + RC	90.19 ± 0.85	$\underline{\mathbf{46.82 \pm 0.87}}$	$\underline{\mathbf{84.95 \pm 2.01}}$	27.83 ± 0.93
MS + RC	90.77 ± 0.32	46.42 ± 0.93	83.72 ± 0.91	28.34 ± 0.76
Naive	19.86 ± 0.07	7.90 ± 0.65	19.84 ± 0.22	20.57 ± 1.76
HR	20.46 ± 0.45	8.29 ± 0.70	19.36 ± 0.35	19.95 ± 0.49
TS	21.40 ± 0.35	7.97 ± 0.58	19.58 ± 0.41	20.96 ± 0.94
VS	34.85 ± 0.05	7.98 ± 0.24	19.90 ± 0.17	20.66 ± 0.62
MS	19.58 ± 0.31	8.10 ± 0.44	19.41 ± 0.16	19.49 ± 0.25
TS + RC	19.72 ± 0.16	8.21 ± 0.26	19.71 ± 0.40	21.40 ± 0.02
VS + RC	34.85 ± 11.13	10.57 ± 0.38	15.32 ± 5.15	20.87 ± 1.01
MS + RC	$\mathbf{37.80 \pm 5.26}$	$\mathbf{11.96 \pm 1.03}$	$\mathbf{19.93 \pm 1.87}$	$\mathbf{21.96 \pm 1.06}$

DER++ uses two hyper-parameters α and β to control the contribution of each regularizer. Intuitively, the regularizer controlled by α promotes stability of the output distribution, while the regularizer controlled by β prevents a drop in the predictive performance on previous examples (since $\mathcal{L}$ is the classification loss).

Interestingly, DER++ is known to result in calibrated models. However, the original paper [3, 4] did not consider any calibration methods. We combined DER++ with 5 calibration methods, including our RC and we verified whether we can improve DER++ calibration.

We compare all the methods with the offline learning model jointly trained on the dataset resulting from the concatenation of all experiences: $\mathcal{D} = \cup_{i=1}^{N} e_i$, for a stream with N experiences. Ideally, we would like CL models to achieve a similar calibration than the offline learning models. As expected, due to the continuous training and the presence of drifts between experiences the CL models under-perform with respect to the offline models. All CL strategies are coupled with various calibration methods, including self-training HR, three post-processing calibration techniques (TS, VS, and MS), and our Replayed Calibration RC.

Benchmarks. We assess the performance of calibration methods on 4 CL benchmarks: Split MNIST [25], Split CIFAR100 [17, 22], EuroSAT [13, 14] and Atari [2, 18]. Split MNIST is obtained by splitting the MNIST dataset into 5 experiences, each of which contains examples from 2 classes. Similarly, Split CIFAR100 is obtained by splitting the CIFAR100 dataset into 10 experiences, each of which contains examples from 10 classes. Both benchmarks are class-incremental benchmarks [22].

EuroSAT is a publicly available dataset for land use and land cover classification from Sentinel-2 satellite images. We adopted this dataset as it represents an interesting ex-

Table 2. Average ECE (10 bins) and standard deviation on the test set of all experiences computed at the end of training. Bold highlights the best result for each CL strategy and Joint Training. Bold and underline highlights overall best across CL strategies.

ECE (%)	*Split MNIST*	*Split CIFAR100*	*EuroSAT*	*Atari*
Joint	4.53 ± 0.40	14.60 ± 6.23	4.45 ± 2.13	2.20 ± 1.83
HR	2.85 ± 1.04	15.91 ± 3.00	3.12 ± 3.34	1.90 ± 0.79
TS	$\mathbf{1.56 \pm 0.32}$	7.80 ± 4.12	2.57 ± 1.78	1.52 ± 0.71
VS	1.70 ± 0.27	$\mathbf{5.75 \pm 2.03}$	$\mathbf{2.46 \pm 1.81}$	$\mathbf{1.38 \pm 0.20}$
MS	38.08 ± 2.05	27.96 ± 4.71	39.67 ± 2.97	23.50 ± 2.97
DER++	1.96 ± 0.42	36.45 ± 4.09	11.29 ± 1.67	12.51 ± 0.65
HR	1.96 ± 0.20	38.70 ± 1.33	10.69 ± 0.78	11.69 ± 1.64
TS	$\underline{\mathbf{1.25 \pm 0.13}}$	28.42 ± 2.42	13.00 ± 2.68	7.25 ± 0.29
VS	4.86 ± 0.35	34.64 ± 0.59	17.93 ± 4.07	7.67 ± 1.97
MS	4.08 ± 0.16	37.27 ± 0.29	28.75 ± 8.81	8.98 ± 4.33
TS + RC	1.43 ± 0.13	27.52 ± 1.96	$\mathbf{9.00 \pm 5.13}$	6.13 ± 0.69
VS + RC	3.63 ± 0.77	9.59 ± 1.16	15.45 ± 8.85	4.69 ± 0.49
MS + RC	3.77 ± 0.26	$\underline{\mathbf{8.51 \pm 0.67}}$	10.86 ± 2.90	$\underline{\mathbf{3.61 \pm 1.16}}$
Replay	3.77 ± 0.32	38.42 ± 5.95	11.07 ± 2.09	48.94 ± 0.68
HR	3.85 ± 0.35	39.31 ± 1.76	9.46 ± 0.71	48.68 ± 1.24
TS	2.86 ± 0.69	23.72 ± 4.35	12.96 ± 2.65	49.96 ± 2.28
VS	8.21 ± 3.05	50.11 ± 2.84	15.06 ± 0.22	34.78 ± 5.09
MS	8.99 ± 4.26	49.76 ± 1.82	12.91 ± 3.53	41.13 ± 2.62
TS + RC	$\mathbf{2.24 \pm 0.25}$	$\mathbf{14.63 \pm 3.35}$	8.23 ± 0.96	41.12 ± 1.87
VS + RC	4.29 ± 0.62	26.04 ± 1.25	$\underline{\mathbf{4.70 \pm 0.64}}$	13.75 ± 1.23
MS + RC	3.76 ± 0.45	27.12 ± 1.61	9.46 ± 0.62	$\mathbf{13.46 \pm 3.40}$
Naive	70.31 ± 0.79	72.44 ± 3.15	76.57 ± 1.61	34.60 ± 5.35
HR	67.05 ± 3.95	70.40 ± 3.53	74.88 ± 2.25	34.51 ± 4.20
TS	71.94 ± 2.08	67.07 ± 2.24	76.31 ± 0.55	47.10 ± 3.33
VS	73.64 ± 0.84	65.89 ± 1.07	78.17 ± 0.31	26.35 ± 8.56
MS	72.73 ± 4.09	63.56 ± 1.81	75.70 ± 2.64	35.77 ± 5.71
TS + RC	65.57 ± 2.31	61.26 ± 2.99	74.58 ± 1.24	42.03 ± 4.61
VS + RC	30.31 ± 5.75	25.73 ± 1.29	35.69 ± 4.99	29.20 ± 3.00
MS + RC	$\mathbf{28.07 \pm 3.40}$	$\mathbf{21.99 \pm 1.35}$	$\mathbf{31.27 \pm 2.27}$	$\mathbf{13.87 \pm 5.29}$

ample of a CL application in a resource-constrained environment. The CL agent can operate directly on the satellite in an autonomous way. Therefore, it needs to provide robust, calibrated predictions. We created a class-incremental benchmark by splitting the dataset into 5 experiences, each of which contains JPEG-encoded RGB images from 2 classes describing the land type.

For Atari we used the replay buffer data released in [1] to pair the game frames with the optimal action chosen by a trained DQN agent. We combined the data from 5 different games (VideoPinball, Boxing, Breakout, StarGunner, Atlantis) to define our own domain-incremental benchmark [25] with one game per experience. Each experience contains 200k randomly sampled stacks of four consecutive game frames paired with the optimal action from the last replay buffer. In this scenario, we can treat the problem of learning the policy $\pi(a|s)$ (that predicts the action a given the state s) as a supervised task.

In our Atari benchmark, the output layer is fixed since the action space is defined by Atari. However, the optimal action distribution changes across games, with some actions never being selected in some of them or their frequency drifting from one experience to the other.

Experimental setup. On each benchmark, we conducted a model selection for each calibration and CL strategy. For our RC, we kept the same configuration of the hyperparameters found during model selection on the corresponding calibration strategy (e.g., we performed model selection for TS and applied the same configuration to TS + RC).

We report the complete set of optimal values found by model selection in the Appendix. On Split MNIST, we used a one-hidden-layer MLP trained with SGD. On Split CIFAR100 and EuroSAT, we used a ResNet110 and ResNet50 [12], respectively. Both models are trained with AdamW.

On Atari we chose the DQN [20] with full Atari action space optimized with Adam.
The reliability diagrams and the corresponding ECEs are computed from 10 equally-spaced bins. The first bin spans the $[0, 0.1]$ confidence interval, the second bin the $(0.1, 0.2]$ confidence interval and so on up until the last bin spanning the $(0.9, 1.0]$ confidence interval.
We used the Avalanche library [5] to run all the experiments. Our experiments do not use task labels. This means that at test time the model needs to distinguish between all classes learned during training.

4. Results

Table 1 and 2 report the average accuracy and ECE, respectively, on the entire data stream at the end of training. The runs are averaged over 3 random seeds. We now highlight the main results found in our empirical evaluation.

Joint Training calibration. Calibration strategies do not hurt predictive accuracy in Joint Training, except with MS on Atari. MS also achieves the worst ECE in Joint Training in all benchmarks. These results are in line with the original MS paper [10], where MS was not able to consistently train calibrated models. We will see how this behavior changes in CL setup. Interestingly, although VS performs the same type of post-processing as MS (but with a learned diagonal matrix instead of a full matrix), it shows a much better calibration in Joint Training. Again, this is aligned with the results presented in [10].

RC mitigates forgetting. Both the calibration and the accuracy are heavily impacted by the CL training. As expected, Naive finetuning causes catastrophic forgetting of previous knowledge on all class-incremental benchmarks. Due to its domain-incremental nature, the Atari benchmark shows a softer forgetting. The fixed output space enjoys better stability and prevents the accuracy from dropping to the level of a random classifier. When combined with MS, our RC approach is able to improve the accuracy (Figure 3) as well as the ECE on all benchmarks (Figure 4). Importantly, RC is not equal to replay since the examples come from the validation set and are not used to maximize the predictive accuracy, but rather the calibration.

RC improves calibration. We found RC to be beneficial even when paired with the Replay and DER++ strategies (Figure 5 and Figure 6, respectively). Although there is no unique post-processing strategy that consistently performs better than the others, our RC always improves calibration on all 4 benchmarks, often by large margins. For example, on EuroSAT, post-processing strategies alone achieve an ECE between 12% (best case) and 15% (worse). After

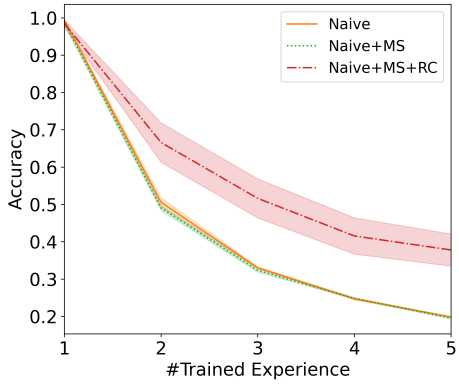

Figure 3. Accuracy of Naive on Split MNIST.

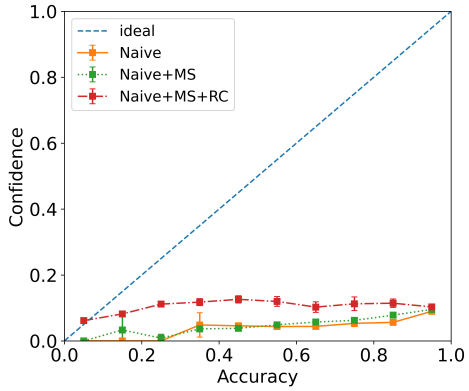

Figure 4. Calibration diagram for Naive on Split CIFAR100.

applying RC, we are able to reduce the gap to 4% and 9%, respectively.
In some cases, a calibrated model results in a decrease in accuracy. For example, TS on Split CIFAR100 drops the average accuracy from 40% with Replay to 23% with TS and 15% with TS+RC. Still, in terms of calibration TS outperforms the other approaches. This trade-off needs to be carefully considered: whether to prefer a less accurate, but calibrated model, or vice versa a more accurate but less calibrated model. However, it is also important to note that calibration does not necessarily causes a drop in accuracy. For example, Replay with VS+RC on EuroSAT results in an improvement in both accuracy and ECE.

Calibration techniques boost DER++. DER++ is one of the most interesting strategies in terms of calibration. DER++ achieves the best calibration on the very challenging Split CIFAR100 and Atari. The original paper [4] also showed the effectiveness of DER++ in calibrating CL models. However, the paper did not leverage any calibra-

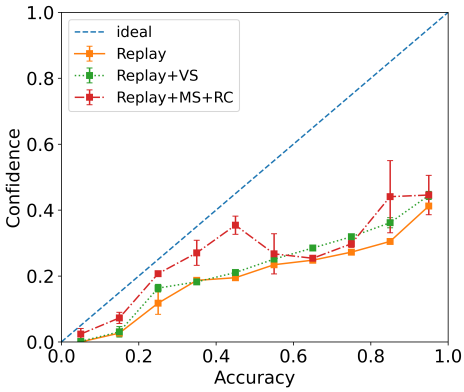

Figure 5. Calibration diagram for Replay on Atari.

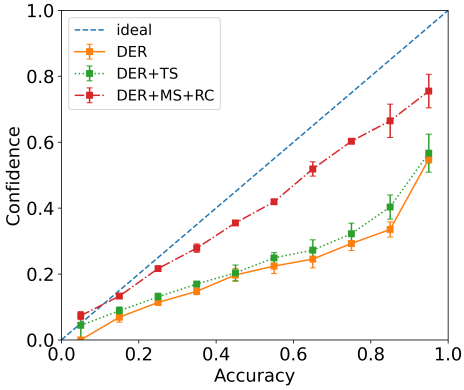

Figure 6. Calibration diagram for DER++ on Split CIFAR100.

tion methods and did not report precise calibration values. In our experiments, we show how DER++ is indeed very effective, but we also point out how its calibration ability can easily be improved when coupled with calibration strategies. In particular, our RC strategy outperforms all other combinations when coupled with MS (Figure 6), without a substantial decrease in accuracy.

Calibration of CL models remains less effective than that of Joint Training models. However, we showed how CL strategies and calibration strategies can operate together, resulting in better calibrated models. Even a state-of-the-art strategy like DER++ enjoys clear improvements in calibration with post-processing techniques.
Our results apply to a diverse set of CL benchmarks, some of which are especially promising in terms of real-world applications (EuroSAT) and generalization to other kinds of problems beyond pure pattern recognition (Atari).

5. Conclusion and Future Work

We start from the assumption that perfect predictive models do not exist. CL models inevitably make mistakes. Instead of only pursuing a better predictive model, we argue that it is equally relevant to design robust models that can be trusted. Calibration allows the model itself to learn a meaningful notion of confidence about its predictions. In particular, the confidence expresses the expected average accuracy on that kind of examples. Calibrated models can operate more autonomously than uncalibrated ones since they can detect when they are likely to make a mistake and call for external help (e.g., a human).

We provided the first empirical evaluation on continual calibration and we show how, on one side, CL models are not naturally calibrated and that, on the other side, post-processing calibration and self-calibration are effective when combined with CL strategies. Our Replayed Calibration improved the performance of post-processing calibration methods across different calibration and CL techniques. We hope our work can increase the attention towards continual calibration, and we highlight some promising research directions.
There are only a few self-calibration techniques currently available for multi-class classification with neural networks [21]. However, self-calibration techniques are inherently compatible with a CL setup, since they operate online during training, without requiring a separate calibration phase. Efforts in designing new self-calibration techniques could directly benefit their CL application.
Calibration mostly considers supervised classification tasks. It is not entirely clear how to frame calibration in other types of tasks, like Reinforcement Learning. A recent work on the topic shed some light on this very challenging research direction in a model-based setting [19]. Being non-stationary by design, discoveries in calibration for reinforcement learning would likely have a strong impact on CL as well.
Finally, our empirical evaluation considered several CL benchmarks, including real-world datasets like EuroSAT and action classification from Atari. We are planning to extend our experiments by also including Natural Language Processing benchmarks. We look forward to future works tackling the continual calibration challenge.

Acknowledgements

Work supported by EU EIC project EMERGE (Grant No. 101070918) and by PNRR- M4C2- Investimento 1.3, Partenariato Esteso PE00000013- ”FAIR- Future Artificial Intelligence Research”- Spoke 1 ”Human-centered AI”, funded by the European Commission under the NextGeneration EU programme.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 104–114. PMLR, 2020. 6

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. 5

[3] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-Incremental Continual Learning into the eXtended DER-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2022. 1, 5

[4] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and SIMONE CALDERARA. Dark Experience for General Continual Learning: A Strong, Simple Baseline. In *Advances in Neural Information Processing Systems*, pages 15920–15930. Curran Associates, Inc., 2020. 1, 4, 5, 7

[5] Antonio Carta, Lorenzo Pellegrini, Andrea Cossu, Hamed Hemati, and Vincenzo Lomonaco. Avalanche: A PyTorch Library for Deep Continual Learning. *Journal of Machine Learning Research*, 24(363):1–6, 2023. 7

[6] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with Rejection. In *Algorithmic Learning Theory*, pages 67–82, Cham, 2016. Springer International Publishing. 1

[7] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 1

[8] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in Nonstationary Environments: A Survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015. 4

[9] Robert French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. 1

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pages 1321–1330, Sydney, NSW, Australia, 2017. JMLR.org. 1, 2, 3, 7

[11] Tyler L. Hayes, Giri P. Krishnan, Maxim Bazhenov, Hava T. Siegelmann, Terrence J. Sejnowski, and Christopher Kanan. Replay in Deep Learning: Current Approaches and Missing Biological Elements. *Neural computation*, 33(11):2908–2950, 2021. 4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[13] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207. IEEE, 2018. 5

[14] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 5

[15] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020. 1

[16] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M. van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas S. Tolias, Simone Scardapane, Luca Antiga, Subutai Ahmad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: An End-to-End Library for Continual Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3595–3605. IEEE, 2021. 3

[17] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 5

[18] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018. 5

[19] Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. Calibrated Model-Based Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4314–4323. PMLR, 2019. 8

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015. 7

[21] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. Regularizing Neural Networks by Penalizing Confident Output Distributions. In *ICLR Workshop*, 2017. 1, 3, 8

[22] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 5

[23] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Contin-

ual Learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 4

[24] Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas Schön. Evaluating model calibration in classification. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 3459–3467. PMLR, 2019. 2

[25] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *Continual Learning Workshop NeurIPS*, 2018. 5, 6

[26] Xu-Yao Zhang, Guo-Sen Xie, Xiuli Li, Tao Mei, and Cheng-Lin Liu. A Survey on Learning to Reject. *Proceedings of the IEEE*, 111(2):185–215, 2023. 1, 2, 3