DATS: DISTANCE-AWARE TEMPERATURE SCALING FOR CALIBRATED CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual Learning (CL) is recently gaining increasing attention for its ability to enable a single model to learn incrementally from a sequence of new classes. In this scenario, it is important to keep consistent predictive performance across all the classes and prevent the so-called Catastrophic Forgetting (CF). However, in safety-critical applications, predictive performance alone is insufficient. Predictive models should also be able to reliably communicate their uncertainty in a calibrated manner – that is, with confidence scores aligned to the true frequencies of target events. Existing approaches in CL address calibration primarily from a data-centric perspective, relying on a *single* temperature shared across all tasks. Such solutions overlook task-specific differences, leading to large fluctuations in calibration error across tasks. For this reason, we argue that a more principled approach should adapt the temperature according to the distance to the current task. However, the unavailability of the task information at test time/during deployment poses a major challenge to achieve the intended objective. For this, we propose Distance-Aware Temperature Scaling (DATS), which combines prototype-based distance estimation with distance-aware calibration to infer task proximity and assign adaptive temperatures without prior task information. Through extensive empirical evaluation on both standard benchmarks and real-world, imbalanced datasets taken from the biomedical domain, our approach demonstrates to be stable, reliable and consistent in reducing calibration error across tasks compared to state-of-the-art approaches.

1 INTRODUCTION

The steep improvement of the predictive capabilities of modern neural architectures has led practitioners to increasingly deploy neural networks into critical decision-making systems. In these contexts, however, predictive models must not only be accurate but also calibrated – i.e., able to reliably communicate via uncertainty estimates when they are likely to be incorrect. Yet, despite their accuracy, neural networks often show under- or over-confidence, especially under distribution shifts. Model calibration offers a way to ensure that a model's predicted confidence levels are statistically consistent with the empirical frequency of correct predictions. For instance, among all predictions assigned a confidence level of 85%, the model should yield correct outputs in approximatively 85% of the cases. Consequently, model calibration has become progressively more investigated over the years to enhance the reliability and trustworthiness of neural architectures in standard single-task settings (Guo et al., 2017; Zhang et al., 2020; Gupta et al., 2021; Tomani et al., 2022).

However, in many real-world applications – ranging from predicting virus variants to product recommendation in e-commerce –, the environment is not static but may vary over time. For this reason, Continual Learning (CL) has gained increasing attention for its ability to enable a single model to learn incrementally from a sequence of new classes. In this scenario, due to the tendency of the model to focus on the most recent information, it is important to keep consistent predictive performance across all the classes and thus mitigate the so-called Catastrophic Forgetting (CF) (McCloskey & Cohen, 1989; Ratcliff, 1990). Reducing the forgetting effect in CL has been largely investigated in recent years (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Tao et al., 2020; Bang et al., 2021; Hurtado et al., 2023; Serra et al., 2025) but remains a major challenge yet to be completely solved.

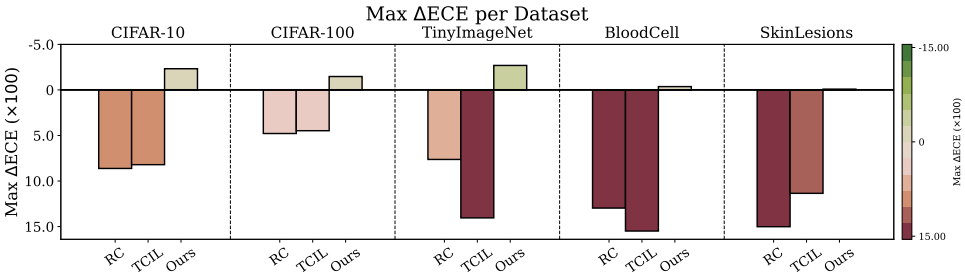

Figure 1: Comparison of the worst-case difference in Expected Calibration Error (Max ΔECE) after post-hoc re-calibration. Positive values (red) indicate worsened calibration (higher CE than before re-calibration), while negative ones (green) indicate improved calibration (lower calibration error than before re-calibration). We observe that state-of-the-art methods can substantially increase calibration error for certain tasks, whereas our approach consistently reduces miscalibration.

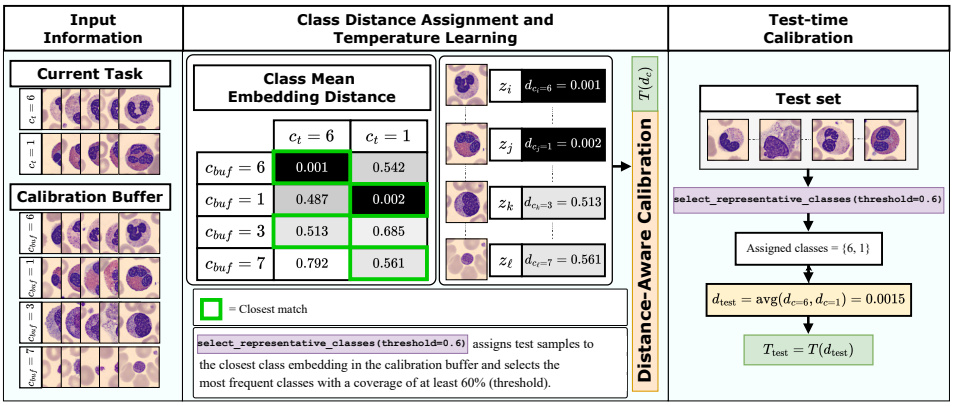

Figure 2: Schematic overview of the proposed framework. Given the validation set of the current task $D_{t,val}$ and the calibration buffer $\mathcal{B}_t$, our method first aligns each class in $\mathcal{B}_t$ (i.e., c_{buf}) with the closest class in $D_{t,val}$ (i.e., c_t). Each sample in $\mathcal{B}_t$ (e.g., x_i) is then assigned a distance score $d_{c_{buf}=c_i}$. During calibration, both the logit vector $\boldsymbol{z}_i$ and the corresponding distance score serve as inputs to the distance-aware optimisation phase. At test time, samples from $D_{t,test}$ are mapped to their nearest counterparts in $\mathcal{B}_t$; the assigned classes provide a reference for computing d_{test} and guide the final distance-aware temperature scaling.

Considering the forgetting effect and the dynamic characteristics of the setting, CL models are intrinsically more prone to make wrong predictions than single-task models. Thus, it is even more important to ensure that CL models reliably communicate their uncertainty. Let's consider a practical example as a running example throughout this work; in one hospital, a neural network is trained continuously to detect skin lesions. The model will initially be trained on the most common types of skin lesions (e.g., melanocytic nevi) but, as more data are collected, new and rarer classes (e.g., vascular lesions, dermatofibroma) will be encountered during training. For the deployment of such model in critical scenarios, the model is required to be accurate but, more importantly, should be able to reliably communicate its uncertainty such that human intervention can be requested when complex cases are encountered. For such uncertainty estimates to be useful in practice, they need to be calibrated. In standard single-task classification scenarios, model calibration can be substantially improved by employing post-hoc uncertainty calibration approaches (Platt et al., 1999; Zadrozny & Elkan, 2002; Zhang et al., 2020; Tomani et al., 2023) - that is, after training model outputs are transformed, most commonly by dividing logits by a learnt temperature (temperature scaling), such that they better match the true likelihood.

Despite the relevance of the problem, post-hoc uncertainty calibration, is largely under-explored in CL. The limited literature available (Li et al., 2024; Hwang et al., 2025) tackles the problem from a data-centric perspective by learning a *single* temperature shared across *all* the involved tasks. How-

ever, due to catastrophic forgetting, predictive performance often differs substantially between tasks, indicating that such task-agnostic strategies fail to capture the dynamic nature of CL problems. This motivates us to shift towards *task-aware* calibration methods in order to account for the variability across tasks.

In an ideal setting, each task could be re-calibrated by using its own validation set. Yet, this is impractical during deployment – and in class-incremental learning (CIL) – where the task information is not available at any time. Thus, to solve this challenge and introduce task-awareness into the re-calibration process, we propose Distance-Aware Temperature Scaling (DATS), a method that a) infers the proximity to the current task via a prototype-based distance criterion and b) exploits such information to inject task-awareness during calibration optimisation and, in the context of temperature-based approaches, learns how to assign a temperature based on the estimated distance. This mitigates miscalibration due to temperature variability across tasks.

The contributions of this work can be summarised as follows:

- We demonstrate that, given the dynamic characteristics of CL (Figure 3), calibration should move from task-agnostic to *task-aware* and *adaptive* re-calibration methods and expose the limitations of current task-agnostic data-centric approaches (Figure 4).
- We propose a distance-aware calibration framework that shifts the focus from data selection and injects task-awareness directly into the re-calibration algorithm. Our approach allows us to assign a distance score to each test batch, enabling *task-aware* re-calibration without explicit task information.
- We show the effectiveness of our method on standard benchmarks and more probing real-world scenarios from the medical domain.

2 Preliminaries and Related Work

Preliminaries Following the notation proposed in Hwang et al. (2025), we assume to have a dataset $D_t = \{(x_i, y_i)\}$ for each task t, such that each class label $y_i \in D_t$ belongs to a disjoint set of classes C_t. The dataset is split in $D_{t,train}$, $D_{t,val}$, and $D_{t,test}$. We also assume to have a memory buffer $\mathcal{M}$, which stores a subset of samples from the previously encountered tasks. We denote with C a set of classes and with $\boldsymbol{\mathcal{C}}$ the union of set of classes. Thus, let $\boldsymbol{\mathcal{C}}_{t-1} = \bigcup_{k=1}^{t-1} C_k$ denote the set of all previously seen classes up to task $t-1$; then, the memory $\mathcal{M}_{t-1}$ contains a limited set of labelled samples (x_j, y_j) such that $y_j \in \boldsymbol{\mathcal{C}}_{t-1}$. The model is trained on D_t and uses $\mathcal{M}_{t-1}$ for replay. At the end of the training of task t , the memory $\mathcal{M}_t$ is updated with a portion of samples taken from $D_{t,train}$. Finally, following Li et al. (2024), we also assume to have a calibration buffer $\mathcal{B}_t$ which contains a subset of samples from the validation sets encountered up to task t.

Let f_θ be a classifier parametrised by θ, producing a logit vector $\boldsymbol{z}_i = f_\theta(x_i)$ for an input x_i. The function maps input data to K output classes, where K is the size of $\boldsymbol{\mathcal{C}}_t$ (i.e., the total number of classes at time t). The probability vector $\boldsymbol{p}_i$ is obtained by passing the logits $\boldsymbol{z}_i$ through a softmax function such that $\boldsymbol{p}_i = \text{softmax}(\boldsymbol{z}_i)$. Then, we denote with $\hat{y}_i = \arg\max_k \boldsymbol{p}_{i,k}$ and $\hat{p}_i = \max_k \boldsymbol{p}_{i,k}$ the predicted class and the confidence of the prediction respectively.

Using the notion of calibration, we can define *perfect calibration* (Guo et al., 2017) as:

$$\mathbb{P}(\hat{y}_i = y_i | \hat{p}_i = p) = p \quad \forall p \in [0, 1]. \tag{1}$$

It naturally follows the definition of Calibration Error (CE) (Naeini et al., 2015) which measures the gap between predicted confidence and actual accuracy:

$$\mathbb{E}_{(x_i,y_i)\sim\mathcal{P}} \left[|\mathbb{P}(\hat{y}_i = y_i \mid \hat{p}_i) - \hat{p}_i| \right]. \tag{2}$$

However, in practical scenarios with a finite number of samples, the probability defined in Equation 2 cannot be computed exactly, and binning-based estimators are commonly used. Let us divide the interval $[0, 1]$ in B equally-spaced bins b. For each bin B_b, we can compute the empirical average accuracy and confidence via $\text{acc}_b = \frac{1}{|B_b|} \sum_{i \in b} \mathbb{1}(\hat{y}_i = y_i)$ and $\text{conf}_b = \frac{1}{|B_b|} \sum_{i \in b} \hat{p}_i$ respectively. Finally, we can estimate the expected calibration error (ECE) via

$$\text{ECE} = \sum_{b=1}^{B} \frac{|B_b|}{N} |\text{acc}_b - \text{conf}_b|, \tag{3}$$

where N represents the total number of samples in the considered dataset D.

Post-hoc calibration methods Among existing calibration strategies, post-hoc methods have gained particular popularity due to their ease of application. Unlike approaches that require modifying the classifier's training procedure (Müller et al., 2019; Moon et al., 2020; Ghosh et al., 2022; Liu et al., 2022; Noh et al., 2023), post-hoc calibration is applied after training, making it flexible and computationally efficient. Several post-hoc techniques have been proposed, including Platt Scaling (Platt et al., 1999), Isotonic Regression (IR) (Zadrozny & Elkan, 2002), and Spline Calibration (Gupta et al., 2021). Following prior work in continual learning (Li et al., 2024; Hwang et al., 2025), we focus on temperature scaling (TS, Guo et al. (2017)). TS learns a single scalar parameter T on a validation set to rescale the classifier's logits: predictions are softened if overconfident ($T > 1$) or sharpened if underconfident ($T < 1$). The method is order-preserving and efficient, as it requires learning only one parameter. More advanced variants of TS extend this idea, for example through ensembling (ETS, Zhang et al. (2020)) or parameterised temperature scaling (PTS, Tomani et al. (2022)), where T is modelled sample-wise by a multi-layer perceptron (MLP). In all these standard TS approaches, calibration relies only on the logit vector $\boldsymbol{z}_i$ of each sample. Taking inspiration from the literature in out-of-distribution detection, where Sun et al. (2022) introduce a non-parametric density estimation based on k-nearest neighbour (KNN) distance, Tomani et al. (2023) propose to exploit the information contained in the inner layers of the classifier (up to the penultimate one) as an additional input for KNN. This density-based information is then used, together with $\boldsymbol{z}$, to learn sample-wise temperatures for uncertainty calibration in OOD scenarios.

Calibration in CL The challenge of post-hoc uncertainty calibration in CL has only been recently investigated. Li et al. (2024) present the first study of this problem in class-incremental learning (CIL) in the context of memory-based approaches. The authors introduce Replayed Calibration (RC), an approach that leverages an additional buffer (namely, a calibration buffer) populated with samples taken from the validation set of each task for calibration optimisation. Despite its simplicity, this work opens a new research direction and highlights the importance of exploiting information from both current and past tasks for calibration. More recently, Hwang et al. (2025) propose T-CIL, a new temperature scaling (TS) approach for CIL. Starting from the observation that using the memory buffer for temperature optimisation is not effective due to its usage during training, the authors propose to perturb the samples contained in the memory buffer to create synthetic samples for calibration. In particular, the magnitude and the direction of the perturbations is adjusted based on the difficulty of the samples taken into consideration – samples from old tasks are perturbed more strongly than the current task's ones.

RC and T-CIL primarily emphasise the choice of the data used for calibration rather than the calibration mechanism itself. For instance, T-CIL generates synthetic samples for calibration starting from the memory buffer and the current validation set, but its generative nature introduces substantial computational overhead and significantly increases the execution time (see Table 5). Ultimately, both methods rely on the same principle of learning a single temperature across tasks while avoiding direct use of the memory buffer. Such task-agnostic approach, however, fails to account for task-specific variability and thus appears insufficient to mitigate miscalibration in CL. In contrast, we depart from this task-agnostic and data-centric perspective (i.e., which data to use for calibration) and are the first to explicitly consider the dynamic nature of CL, introducing task-awareness into post-hoc calibration to better account for the variations across tasks.

3 METHODOLOGY

As argued above, the evolving nature of CL motivates the need for a *task-aware* re-calibration approach. Catastrophic forgetting causes predictive performance to vary substantially across tasks, and we hypothesize that this discrepancy is reflected in the confidence scores of the model. We investigate this behaviour in Figure 3, which reports the average confidence score per task for two different datasets. In both cases, the most recent task exhibit significantly higher confidence than earlier ones. This observation supports our hypothesis: although accuracy for the last task is typically higher, a *task-agnostic* approach cannot yield well-calibrated predictions across all tasks.

To make this point explicit, we translate the *task-agnostic* vs. *task-aware* dilemma in the context of temperature scaling. In a synthetic experiment (Figure 4), we compute the ideal temperature per task by using its corresponding validation set. The results clearly show that the optimal temperature

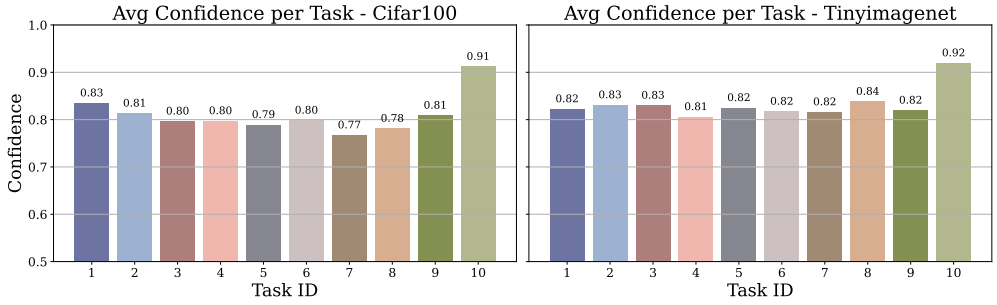

Figure 3: Visualisation of the average confidence score for each task at the end of the training procedure. A clear difference is noticeable between the average confidence of the last task and the one of the past tasks.

per task fluctuates across past tasks and decreases for the most recent ones, demonstrating that a *single* temperature approach is *not sufficient* for effective calibration in CL.

These findings motivate us to move beyond task-agnostic strategies and develop *adaptive*, *task-aware* post-hoc calibration methods. In the context of temperature scaling, this means moving from learning a *single* temperature for all tasks to learning *different* temperatures depending on the task at hand. In this way, we can instil task-awareness into the re-calibration phase and overcome the limitations of current task-agnostic state-of-the-art approaches. In the ideal scenario presented in Figure 4, one could calibrate each task with its validation set, but this is impractical in class-incremental learning and during deployment where task labels are not available at any time. To address this challenge, we propose to infer the task information indirectly through prototype-based distance information and use this signal to tune the temperature accordingly.

In the remainder of this section, we describe in more details the modules constituting our approach. First, in Section 3.1, we describe how to compute and assign distance scores to classes in the calibration buffer. Then, in Section 3.2, we illustrate the use of the learned scores for task-aware temperature optimisation procedure. Finally, in Section 3.3, we outline the strategy to assign a distance score to the test set without task knowledge and to adapt the temperature accordingly. A summary of the whole method is described in Appendix A.6 - Algorithm 1.

3.1 Class Distance Score Assignment

The main objective in this phase is to identify the closest class pairs between the current classes (contained in $D_{t,val}$) and those in the calibration buffer (i.e., all seen classes). For this purpose, let C_t be the set of classes for the current task t, and $\mathcal{C}_{buf} = \mathcal{C}_t = \bigcup_{k=1}^{t} C_k$ be the set of buffer classes containing all classes seen up to and including task t.

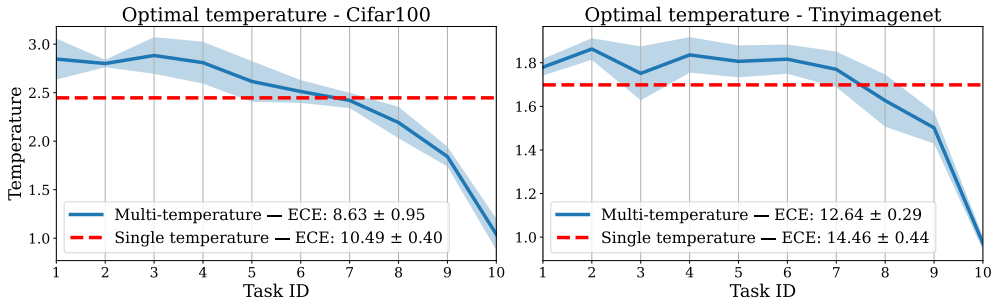

Figure 4: Comparison between the ideal temperature for each task learned by exploiting the corresponding validation set $D_{t,val}$ (blue line) and the optimal single-temperature learned by concatenating the validation information of all tasks (red dashed line).

To quantify the relationship between classes in the current task and those stored in the calibration buffer, we represent each class by a prototype embedding $\boldsymbol{\mu}_c$, computed as the average of the latent representations at the penultimate layer $\ell - 1$ of the neural network:

$$\boldsymbol{\mu}_c = \frac{1}{|D_c|} \sum_{(x_i, y_i) \in D_c} f_\theta^{\ell-1}(x_i), \tag{4}$$

where D_c denotes the set of samples belonging to class c, and $f_\theta^{\ell-1}(x_i)$ is the output of the penultimate layer for input x_i. For the current task classes, these prototypes are computed from $D_{t,val}$, while for buffer classes they are computed from the samples stored in $\mathcal{B}_t$.

Then, to assess the similarity between buffer and validation classes, we compute the pairwise cosine similarity matrix $\boldsymbol{S} \in \mathbb{R}^{|C_t| \times |\boldsymbol{\mathcal{C}}_{buf}|}$:

$$\boldsymbol{S}(c_t, c_{buf}) = \frac{\boldsymbol{\mu}_{c_t}^\top \boldsymbol{\mu}_{c_{buf}}}{\|\boldsymbol{\mu}_{c_t}\|_2 \, \|\boldsymbol{\mu}_{c_{buf}}\|_2}, \quad c_t \in C_t, \; c_{buf} \in \boldsymbol{\mathcal{C}}_{buf}. \tag{5}$$

Finally, the score for each buffer class c_{buf} is defined as the minimum distance (based on cosine similarity) to any current class:

$$d(c_{buf}) = \min_{c_t \in C_t} (1 - \boldsymbol{S}(c_t, c_{buf})) \quad \forall c_{buf} \in \boldsymbol{\mathcal{C}}_{buf}. \tag{6}$$

The scores are then normalised via MinMaxScaler. Intuitively, we expect $d(c_{buf}) \approx 0$ when $c_{buf} \in C_t$, and $d(c_{buf}) \gg 0$, otherwise. The score is assigned back to the examples in the buffer such that the ones from the same class c share the same score $d(c_{buf} = c)$. An illustration of this assignment is depicted in Figure 2; the learned distance scores are assigned back to samples in $\mathcal{B}_t$ according to the corresponding class.

3.2 Task-Aware Calibration

Once we assign a distance-per-class score $d(c_{buf})$ to each sample in the calibration buffer $\mathcal{B}_t$, we leverage these scores as an additional signal during the calibration optimisation phase. Here, $d(c_{buf})$ acts as a proxy for the distance between a buffer sample and the current task, thereby introducing task-awareness into the calibration process. While the logit information $\boldsymbol{z}$ already provide fine-grained, sample-specific information, we assign the same distance score to all samples of a given class c. This choice reduces the information burden during temperature optimisation and encourages the model to capture class-level temperature dynamics. In contrast, using per-sample distances would introduce unnecessary noise and instability, since the variability within a class may impede learning the broader class-level trends that are most relevant for inducing *task-aware* calibration.

For a given class c, the temperature is defined as

$$T(d_c) = w_c d_c + T_{base}, \tag{7}$$

where T_{base} is the global base temperature and w_c controls how strongly the distance score d_c modulates the adjustment. In this way, our method can learn a temperature for each seen class (i.e., for each class $c \in \boldsymbol{\mathcal{C}}_{buf}$). We learn T_{base} and w_c by minimising the Brier score based on the logits, distance scores and labels from $\mathcal{B}_t$ (see Equation 9). More details about the optimisation of our task-aware re-calibration approach are reported in Appendix A.1.

3.3 Test-time Calibration

At test time, our goal is to apply $T(d_c)$ to $D_{t,test}$ for all tasks up to t. Since class and task information are not available in this phase, we proceed as follows. First, each test sample is assigned to the nearest class embedding $\boldsymbol{\mu}_c$ in the calibration buffer $\mathcal{B}_t$. Then, we retain only the most frequent classes covering at least 60% of the assignments. This filtering step ensures that the selected classes reliably describe the test set while reducing the risk of including spurious classes. For this, considering the catastrophic forgetting effect, we adopt a 60% threshold as a practical trade-off between coverage and representativeness of each task: higher thresholds increase the likelihood of incorporating misclassified classes, thereby amplifying discrepancies between assigned and true classes (see Appendix A.5, Table 6).

Through this class assignment, we approximate the dominant classes in the test set and use them to compute a representative score for the entire set. Specifically, given the set of assigned classes $\hat{C}_{\text{test}}$ and the distance score obtained from the calibration buffer, we define the test set distance d_{test} as

$$d_{\text{test}} = \frac{1}{|\hat{C}_{\text{test}}|} \sum_{c \in \hat{C}_{\text{test}}} d(c_{buf} = c). \tag{8}$$

A numerical example of this procedure is illustrated in Figure 2. Suppose we are processing the test set of the current task with classes 6 and 1 and, with a threshold of 0.6, the class assignment step successfully retrieves the two classes. Then, d_{test} is computed via Equation 8 and used as input to calculate the temperature according to Equation 7. Intuitively, when the class assignment is (almost) correct, we expect to have $d_{\text{test}} \approx 0$ when dealing with the current task and $d_{\text{test}} > 0$ otherwise.

4 Experiments

4.1 Datasets and Settings

Datasets In line with related work (Li et al., 2024; Hwang et al., 2025), we evaluate our method on three popular datasets: CIFAR10, CIFAR100 (Krizhevsky et al., 2009), and TinyImageNet (Le & Yang, 2015). To reproduce the class-incremental scenario, we divide each dataset into disjoint tasks. For CIFAR10, we randomly assign two classes to each task (5 tasks). For CIFAR100 and TinyImageNet, we respectively assign 10 and 20 classes to every task (10 tasks). This setup allows the evaluation of baselines independently of task composition. As anticipated in Section 1, we next evaluate our approach under more challenging and realistic settings. Instead of using CIFAR-like datasets (e.g., ImageNet) or artificial tasks (e.g., EMNIST), we conduct validation on biomedical image analysis datasets. Beyond domain differences in image statistics, these datasets also introduce an additional challenge: class imbalance. To better imitate realistic scenarios, where newer tasks typically provide fewer samples due to limited collection time or due to class rarity, we assign classes to tasks according to their relative sizes. This setting represents a realistic and common scenario in AI-assisted medicine where one hospital may encounter different pathology subtypes with different frequencies. For this purpose, we focus on two biomedical datasets, both annotated by expert clinical pathologists; BloodCell (Acevedo et al., 2020; Yang et al., 2021), with 8 classes of microscopic blood cell images, and SkinLesions (Tschandl et al., 2018; Codella et al., 2019; Yang et al., 2021), with 7 classes of dermoscopic images for skin cancer detection. Statistics are reported in Appendix A.2.

Experimental settings In all the main experiments, we use a slim version of ResNet18 (He et al., 2016) as done in previous CL works (Lopez-Paz & Ranzato, 2017; Kumari et al., 2022; Hurtado et al., 2023; Serra et al., 2025) and use the SGD optimizer with a learning rate of 0.1. For replay, we use Experience Replay (ER) (Chaudhry et al., 2019). In order to achieve competitive results on each dataset and have a meaningful baseline, the size of the memory buffer $\mathcal{M}$ is adapted according to the dataset in consideration. It is important to notice that, since our approach is post-hoc, the initial training procedure can be changed as desired as long as the trained model achieves reasonable predictive results. We ablate the use of a different architecture in Appendix A.5 - Table 8. For the composition of the calibration buffer $\mathcal{B}$, following Li et al. (2024), we reserve with random selection a percentage of the validation set $D_{t,val}$ of each processed task. We ablate the percentage value in Appendix A.5 - Table 7. A detailed list of the hyperparameter selection can be found in the supplementary material. We run the experiments on three random seeds such that each time the class-per-task assignment is different. Each experiment was run on a Linux machine using a single Quadro RTX 5000 with 16 GB RAM.

Baselines In line with prior work (Li et al., 2024; Hwang et al., 2025), we first benchmark our method against standard temperature scaling (TS) techniques (i.e., TS, Ensemble TS (ETS), and Parametrised TS (PTS)) which rely only on the current validation set. This comparison highlights the inefficacy of approaches that ignore information from previous tasks in CL. Then, we compare DATS against RC (Li et al., 2024) and T-CIL (Hwang et al., 2025) to assess the effectiveness of our approach in comparison with task-agnostic approaches specifically tailored for class-incremental learning (CIL).

Table 1: Comparison of average negative log-likelihood (NLL), average expected calibration error (ECE) and average delta last ECE (ΔLECE) on standard benchmarks.

	CIFAR10 (Acc: 63.58 ± 2.86)			CIFAR100 (Acc: 45.19 ± 0.89)			TinyImageNet (Acc: 22.79 ± 0.38)		
	NLL (▼)	AECE (▼)	ΔLECE (▼)	NLL (▼)	AECE (▼)	ΔLECE (▼)	NLL (▼)	AECE (▼)	ΔLECE (▼)
Uncal	3.38 ± 0.54	30.77 ± 3.01	✗	4.02 ± 0.07	35.55 ± 0.47	✗	4.27 ± 0.13	38.09 ± 1.23	✗
TS	2.33 ± 0.28	27.94 ± 2.95	-1.14 ± 0.25	4.03 ± 0.59	35.38 ± 2.64	0.08 ± 1.93	4.36 ± 0.09	39.45 ± 0.50	0.55 ± 0.35
ETS	2.31 ± 0.34	28.26 ± 2.75	-1.00 ± 0.18	3.90 ± 0.60	35.67 ± 2.00	0.18 ± 1.62	4.36 ± 0.09	39.45 ± 0.51	0.55 ± 0.35
PTS	✗	✗	✗	5.96 ± 1.50	34.92 ± 2.06	1.48 ± 2.34	5.10 ± 1.08	37.50 ± 2.83	-0.09 ± 0.31
RC	1.12 ± 0.06	12.18 ± 2.33	7.68 ± 1.24	2.32 ± 0.07	10.03 ± 0.32	6.44 ± 5.98	3.52 ± 0.04	14.45 ± 0.59	7.24 ± 2.87
T-CIL	1.12 ± 0.06	12.33 ± 2.68	7.34 ± 2.01	2.32 ± 0.08	9.56 ± 1.29	7.11 ± 3.70	3.52 ± 0.02	**10.48 ± 0.78**	16.77 ± 3.84
Ours	**1.08 ± 0.04**	**8.80 ± 0.93**	**-2.31 ± 0.75**	**2.29 ± 0.06**	**7.63 ± 0.84**	**-1.26 ± 1.21**	**3.50 ± 0.02**	11.84 ± 0.58	**-2.66** ± 1.67

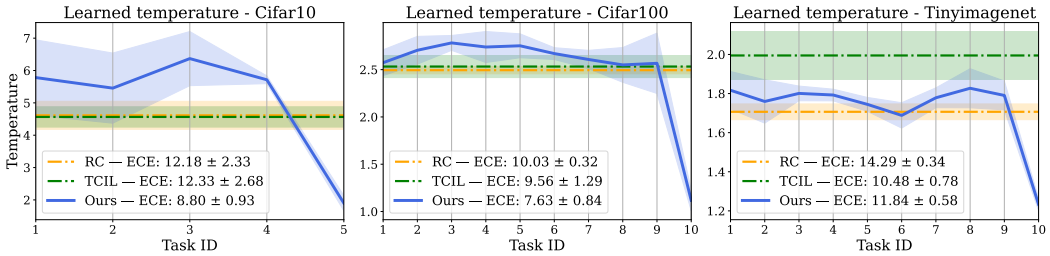

Figure 5: Visual comparison of the learned temperatures across tasks between task-agnostic (RC and T-CIL) and task-aware (ours) calibration methods for CL.

Evaluation metrics Following standard practice in CIL settings, we report the *average* metric. For instance, the average accuracy (*Acc* in our tables) report the average of the accuracy across tasks at the end of the training procedure. Similarly, we report the average negative log-likelihood (NLL) and the average expected calibration error (AECE). We adopt both ECE and NLL as complementary measures of calibration quality. While NLL assesses the overall quality of the predictive distribution, ECE directly measures the alignment between predicted confidence and empirical accuracy. Furthermore, we introduce two additional metrics to evaluate calibration in CL settings: the difference in ECE before and after calibrating the last task (ΔLECE) and the worst-case difference in ECE across tasks after re-calibration ($\max$ ΔECE). We propose ΔLECE to show the effect of the adopted temperature on the current task; considering the change in the confidence distribution between the current and previous tasks, this metric gives an idea of the effect of the calibration on the currently evaluated task. Instead, $\max$ ΔECE provides insight into the stability and robustness of a calibration method. For both metrics, negative values mean that the considered approach effectively reduces the calibration error, while positive values indicate increased ECE. A detailed definition of the metrics is provided in Appendix A.4.

4.2 Empirical Results

From the results reported in Table 1, we can observe that the proposed approach consistently reduces miscalibration in most cases in terms of NLL and ECE. Looking at the difference between the calibration error before and after calibrating the last task (ΔLECE), it is evident that existing solutions produce an undesirable behaviour: miscalibration becomes even more severe than when no re-calibration is applied. The learned temperatures shown in Figure 5 further illustrate these dynamics, where it is clear that a task-agnostic fixed temperature does not capture the dynamic properties of CIL settings. Noticeably, while RC and T-CIL perform similarly on CIFAR datasets, T-CIL assign a much higher temperature on TinyImageNet. This results in a smaller average ECE compared to our method, but at the cost of a substantially larger calibration error on the last task. Importantly, we also demonstrate that our approach remains effective and stable on imbalanced, real-world datasets from the biomedical domain, as reported in Table 2.

In practical deployments, and especially in critical domains as our running example described in Section 1, our method proves safer and more reliable by avoiding large fluctuations across tasks and

Table 2: Comparison of average negative log-likelihood (NLL), average expected calibration error (ECE) and average delta last ECE (ΔLECE) on BloodCell and SkinLesions.

	Blood Cell (Acc: 77.90 ± 4.46)			Skin Lesions (Acc: 49.14 ± 0.67)		
	NLL (▼)	AECE (▼)	ΔLECE (▼)	NLL (▼)	AECE (▼)	ΔLECE (▼)
Uncal	1.25 ± 0.24	16.47 ± 3.54	✗	1.57 ± 0.08	16.48 ± 0.88	✗
RC	0.82 ± 0.12	9.91 ± 1.04	12.53 ± 1.63	1.44 ± 0.02	14.26 ± 1.58	14.77 ± 2.31
T-CIL	0.83 ± 0.11	10.08 ± 1.24	13.31 ± 2.74	1.43 ± 0.03	13.09 ± 0.76	9.56 ± 1.56
Ours	**0.80** ± 0.11	**9.87** ± 1.40	**0.21** ± 0.66	**1.42** ± 0.09	**12.53** ± 2.68	**-1.90** ± 1.81

preventing severe degradation on individual ones. This is illustrated in Figure 1, where we show the worst-case ECE improvement ($\max \Delta$ECE) across tasks for each baseline. A positive delta (in red) indicates that the ECE of a given task is increased after calibration, while a negative delta (in green) represents decreased ECE. From the results, we can observe that, in the worst case, our approach does not change or marginally improves the calibration of a task, while current state-of-the-art approaches exacerbate the problem in all the considered datasets. This property is particularly important in safety-critical settings, such as biomedical applications, where rare or newly introduced classes are at risk of being severely miscalibrated by approaches like RC or T-CIL. In such cases, our method offers a more stable and trustworthy calibration strategy across tasks.

5 Discussion and Conclusion

Execution time Due to space constraints, we report the runtime of the calibration phase in seconds in Appendix A.5 - Table 5. The results show that, while DATS introduces some overhead compared to RC, it remains competitively fast compared to RC and considerably faster than T-CIL.

Comprehensive evaluation tailored to the CL setting Beyond standard metrics, we introduce two new evaluation measures: 1) ΔLECE to assess the impact of calibration on the most recently learned task (given the change in its predictive performance and confidence distribution), and 2) $\max \Delta$ECE which captures whether re-calibration degrades calibration on any task by tracking the worst-case change in ECE across the sequence. In line with our shift from a task-agnostic to a task-aware re-calibration perspective, these metrics provide a more nuanced understanding of re-calibration effectiveness beyond average metrics.

Ablation studies In Appendix A.5, we analyse the impact of the most relevant hyperparameters on the performance of our method. From the results (Tables 6, 7, and 8), our approach remains stable and robust within a broad range of settings.

Limitations The approach assumes that the representative classes selected at test-time are a reliable proxy of the processed task. However, under highly non-stationary streams, this assumption may be weakened.

Conclusions In this work, we tackle the challenges of uncertainty calibration in CL, a largely under-explored problem. By investigating the behaviour of CL models (Figure 3), we argue that post-hoc calibration techniques in CL should move from a task-agnostic to a task-aware perspective. Current state-of-the-art approaches, primarily focusing on the data to use for calibration, overlook task variability and propose a task-agnostic single temperature approach which appears to be limited in CL settings (Figure 4). This choice, in fact, leads to large fluctuations in ECE across tasks (Figure 1), an undesirable behaviour for the deployment of CL models in safety-critical settings. For this reason, we depart from such task-agnostic, data-centric view and are the first to propose an adaptive, task-aware calibration method for CL. To integrate task-awareness into the calibration phase, we propose DATS, an elegant solution that, in the context of TS, is able to adapt the task temperature without prior information about it via a combination of prototype-based distance estimation and distance-aware calibration. The empirical results show that our approach delivers robust calibration across tasks on a variety of datasets, ranging from standard benchmark (Table 1) to real-world, imbalanced biomedical ones (Table 2). Additional ablation studies demonstrates that DATS is simple to tune and efficient, providing an easy-to-integrate tool for post-hoc recalibration of CL models. Overall, DATS offers a lightweight, principled, and effective solution for *task-aware calibration* in class-incremental learning, enabling safer deployment of CL models in safety-critical domains.

Reproducibility Statement

Code availability. We provide the full implementation of our method, along with all scripts necessary to reproduce the experiments. The code is submitted as supplementary material and will be made publicly available upon acceptance.

Data accessibility. All datasets employed in our experiments are publicly available. We include detailed instructions on dataset usage, and any required preprocessing steps are automated via scripts included in the code repository.

Hyperparameters. Complete training configurations and hyperparameter values are specified in the main text. The effect of critical hyperparameters is further examined in the ablation study (see Appendix A.5).

Hardware and runtime. Experiments were conducted on a Linux server with a single NVIDIA Quadro RTX 5000 GPU and 16 GB RAM. Training times are reported in Table 5.

Experiment instructions. The repository includes a README file with step-by-step instructions for reproducing the results. For clarity, we also provide pseudocode for the main components of our method in Appendix A.6.

References

Andrea Acevedo, Anna Merino, Santiago Alférez, Ángel Molina, Laura Boldú, and José Rodellar. A dataset of microscopic peripheral blood cell images for development of automatic recognition systems. *Data in Brief*, 30:105474, 2020.

Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. 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.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, P Dokania, P Torr, and M Ranzato. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019.

Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

Arindam Ghosh, Thomas Schaaf, and Matthew Gormley. Adafocal: Calibration-aware adaptive focal loss. *Advances in Neural Information Processing Systems*, 35:1583–1595, 2022.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=eQe8DEWNN2W.

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*, pp. 770–778, 2016.

Julio Hurtado, Alain Raymond-Sáez, Vladimir Araujo, Vincenzo Lomonaco, Alvaro Soto, and Davide Bacciu. Memory population in continual learning via outlier elimination. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3481–3490, 2023.

Seong-Hyeon Hwang, Minsu Kim, and Steven Euijong Whang. T-cil: Temperature scaling using adversarial perturbation for calibration in class-incremental learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15339–15348, 2025.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *online*, 2009.

Lilly Kumari, Shengjie Wang, Tianyi Zhou, and Jeff A Bilmes. Retrospective adversarial replay for continual learning. *Advances in Neural Information Processing Systems*, 35:28530–28544, 2022.

Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Lanpei Li, Elia Piccoli, Andrea Cossu, Davide Bacciu, and Vincenzo Lomonaco. Calibration of continual learning models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4160–4169, 2024.

Bingyuan Liu, Ismail Ben Ayed, Adrian Galdran, and Jose Dolz. The devil is in the margin: Margin-based label smoothing for network calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 80–88, 2022.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. Confidence-aware learning for deep neural networks. In *international conference on machine learning*, pp. 7034–7044. PMLR, 2020.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 1, 2015.

Jongyoun Noh, Hyekang Park, Junghyup Lee, and Bumsub Ham. Rankmixup: Ranking-based mixup training for network calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1358–1368, 2023.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

Giuseppe Serra, Ben Werner, and Florian Buettner. How to leverage predictive uncertainty estimates for reducing catastrophic forgetting in online continual learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=dczXe0S1oL.

Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International conference on machine learning*, pp. 20827–20840. PMLR, 2022.

Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pp. 254–270. Springer, 2020.

Christian Tomani, Daniel Cremers, and Florian Buettner. Parameterized temperature scaling for boosting the expressive power in post-hoc uncertainty calibration. In *European conference on computer vision*, pp. 555–569. Springer, 2022.

Christian Tomani, Futa Kai Waseda, Yuesong Shen, and Daniel Cremers. Beyond in-domain scenarios: Robust density-aware calibration. In *International Conference on Machine Learning*, pp. 34344–34368. PMLR, 2023.

Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018.

Jiancheng Yang, Rui Shi, and Bingbing Ni. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pp. 191–195, 2021.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 694–699, 2002.

Jize Zhang, Bhavya Kailkhura, and T Yong-Jin Han. Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *International conference on machine learning*, pp. 11117–11128. PMLR, 2020.

A Appendix

A.1 Post-hoc Calibration Optimisation

Given the logits $\boldsymbol{z}$, the distance scores $d(c_{\text{buf}})$, and the labels from the calibration buffer $\mathcal{B}_t$, we fit our task-aware, class-specific temperature scaling parameters for the trained classifier f_θ by minimizing the Brier score using the L-BFGS optimizer:

$$L_{\text{cal}} = \sum_{i=1}^{N} \sum_{c \in \boldsymbol{\mathcal{C}}_{buf}} \left(I_{i,c} - \text{softmax}((\boldsymbol{z}_i^c / T(d_c))\right)^2 \tag{9}$$

where N is the number of samples in $\mathcal{B}_t$, $I_{i,c}$ is an indicator variable equal to 1 if $y_i = c$ and 0 otherwise, and $T(d_c)$ denotes the temperature assigned to class c as a function of its distance score d_c.

A.2 Dataset Statistics

In Table 3, we report the statistics of the datasets used for the main experiments. Following related work, we use standard benchmarks (CIFAR10, CIFAR100, TinyImagenet), and datasets from different domains (SkinLesions and BloodCell). In particular, the two biomedical datasets pose additional challenges as they reflect more realistic conditions (less data, imbalanced).

Table 3: Statistics of the datasets.

Number of	Classes	Samples	Tasks	Image size
CIFAR10	10	60000	5	32
CIFAR100	100	60000	10	32
TinyImageNet	200	100000	10	64
SkinLesions	7	10015	3	64
BloodCell	8	17092	4	28

A.3 Hyperparameter Selection

In Table 4, we report the hyperparameters used for each dataset. The choices follow standard practice in CL literature to obtain a backbone configuration that achieves reasonable predictive performance. Specifically, NF denotes the number of features in the ResNet backbone, $|\mathcal{M}|$ is the memory buffer size, Patience controls early stopping during training, and and Val. inclusion (%) indicates the percentage of each task's validation set that is included in the calibration buffer for calibration optimisation. Values for Val. inclusion (%) were chosen per dataset to reflect dataset size and class balance; small datasets or imbalanced biomedical datasets use larger fractions to ensure sufficient

calibration samples. For standard benchmarks, the value is chosen to match the size of the memory buffer. An ablation study for the percentage value is reported in Table 7.

Table 4: Hyperparameter selection for each dataset.

	NF	$\lvert\mathcal{M}\rvert$	Patience	Val. inclusion (%)
CIFAR10	20	1000	10	10
CIFAR100	32	4000	20	40
TinyImageNet	64	1000	50	20
SkinLesions	64	200	20	50
BloodCell	20	200	50	50

A.4 Evaluation Metrics

We adopt a set of standard metrics for CL together with additional measures specifically designed to assess calibration in class-incremental learning (CIL) settings. Following standard practice, we report the *average* values of the considered metrics across tasks at the end of training. In particular:

- Accuracy (*Acc*) evaluates predictive performance in terms of correct classifications.
- Negative Log-Likelihood (*NLL*) and Expected Calibration Error (*ECE*) serve as complementary measures of calibration quality: NLL captures the overall quality of the predictive distribution, while ECE directly measures the alignment between predicted confidence and empirical accuracy.
- ΔLECE and $\max \Delta$ECE are additional metrics we introduce to assess the effect and robustness of calibration in CL settings.

A detailed definition of each metric follows.

- *Average Accuracy (Acc)*: Let $\text{acc}^{t,i}$ denote the accuracy on task i after learning task t. Given the total number of tasks $\mathcal{T}$, the average accuracy Acc is defined as:

$$Acc = \frac{1}{\mathcal{T}} \sum_{i=1}^{\mathcal{T}} \text{acc}^{\mathcal{T},i}. \tag{10}$$

  This metric summarizes the predictive performance across all tasks at the end of the training procedure.

- *Average Negative Log-Likelihood (NLL)*: The negative log-likelihood measures the quality of the full predictive distribution. Lower values indicate that the predicted probabilities are well aligned with the true labels, penalizing both overconfidence and underconfidence. Let $\text{nll}^{t,i}$ denote the negative log-likelihood on task i after learning task t. The average NLL is defined as:

$$NLL = \frac{1}{\mathcal{T}} \sum_{i=1}^{\mathcal{T}} \text{nll}^{\mathcal{T},i}. \tag{11}$$

- *Average Expected Calibration Error (AECE)*: The expected calibration error measures the discrepancy between predicted confidence and empirical accuracy, typically estimated via binning. Lower values correspond to better-calibrated models. Let $\text{ece}^{t,i}$ denote the ECE on task i after learning task t computed via Equation 3 with $B = 10$. The average ECE (AECE) is defined as:

$$AECE = \frac{1}{\mathcal{T}} \sum_{i=1}^{\mathcal{T}} \text{ece}^{\mathcal{T},i}. \tag{12}$$

- *Delta Last ECE (ΔLECE)*: To assess the effect of the adopted calibration procedure on the last task, we define the change in last-task ECE. Let $\text{ece}^{t,t}$ and $\text{ece-cal}^{t,t}$ denote the ECE on task t before and after calibration, respectively. Then:

$$\Delta\text{LECE} = \text{ece-cal}^{\mathcal{T},\mathcal{T}} - \text{ece}^{\mathcal{T},\mathcal{T}}. \tag{13}$$

  This metric quantifies the direct impact of calibration on the most recently learned task.

- *Worst-case Delta ECE (*$\max \Delta$*ECE)*: To evaluate the stability and robustness of a calibration method across tasks, we consider the worst-case change in ECE. Let $\Delta\text{ece}^{\mathcal{T},i} = \text{ece-cal}^{\mathcal{T},i} - \text{ece}^{\mathcal{T},i}$ be the change in ECE for task i after calibration at the end of training. The worst-case delta ECE is defined as:

$$\max \Delta\text{ECE} = \max_{i \in \{1,\dots,\mathcal{T}\}} \Delta\text{ece}^{\mathcal{T},i}. \tag{14}$$

  This metric captures the most adverse calibration effect across tasks, highlighting whether re-calibration destabilizes the calibration performance across tasks.

For both ΔLECE and $\max \Delta$ECE, negative values indicate that the considered approach reduces miscalibration (improved calibration), while positive values indicate an increase in miscalibration.

A.5 Ablation Study

In this section, we ablate different experimental choices to investigate the effectiveness and robustness of our approach to different settings.

Execution time Table 5 reports the runtime of the calibration phase in seconds. As expected, RC represents the lower bound since it simply applies TS on a calibration buffer without introducing additional computation. Despite the extra steps required by our method, DATS remains competitively fast compared to RC and considerably faster than T-CIL. Notably, the generative nature of T-CIL induces substantial computational overhead leading to longer execution times across datasets.

Table 5: Execution time (in seconds) of the calibration phase after training. After the first task, the value is the sum of the calibration procedure for each task up to the current one.

	CIFAR10	CIFAR100	TinyImageNet	Blood Cell	Skin Lesions
	Runtime (▼, in seconds)				
RC	19.09 ± 2.06	23.76 ± 0.12	376.02 ± 18.11	3.37 ± 0.04	3.92 ± 0.00
T-CIL	230.88 ± 1.46	510.80 ± 21.49	1949.55 ± 55.38	38.52 ± 0.16	59.15 ± 0.36
Ours	22.15 ± 0.12	54.36 ± 7.87	479.61 ± 14.03	7.50 ± 0.17	9.88 ± 0.04

Coverage threshold As described in Section 3.3, at test time, we assign test samples to the closest class embedding $\boldsymbol{\mu}_c$ in the calibration buffer $\mathcal{B}$. Considering that CL models are more prone to be inaccurate (especially for past tasks), we decide to set the coverage threshold to 0.6 for all the experiments as a good compromise between representativeness and coverage. In Table 6, we ablate the coverage threshold (i.e, the percentage of samples in the test set that are used to infer the representative classes). From the results, we can notice a clear trend; reducing the threshold improves the final ECE in most of the cases. This effect is expected since, by reducing the threshold, we are filtering out the least present classes and, most probably, the mismatch between the real and assigned classes on the test set. This is particularly true for more challenging datasets like CIFAR100 and TinyImageNet, where the overall accuracy is less than 50%. In general, we believe a value between 40% and 60% to be a good range for most of the cases.

Table 6: Effect of varying the coverage threshold on the calibration performance (ECE ▼). Best results are typically obtained for thresholds between 40% and 60%

Threshold	0.4	0.5	0.6	0.7	0.8
	ECE (▼)				
CIFAR10	9.53 ± 0.60	8.80 ± 0.93	8.80 ± 0.93	9.40 ± 1.80	12.49 ± 2.040
CIFAR100	6.52 ± 0.62	6.73 ± 0.49	7.63 ± 0.84	8.38 ± 0.99	8.70 ± 0.35
TinyImageNet	10.31 ± 0.32	11.29 ± 0.48	11.84 ± 0.58	11.97 ± 0.40	12.32 ± 0.39

Validation inclusion percentage In Table 7, we analyse the effect of the size of the calibration buffer $\mathcal{B}$ in terms of the percentage of each task's validation set $D_{t,val}$ included in $\mathcal{B}$ (Val. inclusion (%)). For each validation set, samples are drawn uniformly at random. From the results, the percentage of samples included from $D_{t,val}$ has only a minor impact on the calibration performance of our approach.

Table 7: Effect of varying the percentage of each task's validation set (Val. inclusion (%)) on the calibration performance (ECE ▼). Results in terms of ECE do not change considerably when changing the percentage of samples selected from $D_{t,val}$.

Val. inclusion (%)	20	30	40	50
	ECE (▼)			
CIFAR10	8.85 ± 0.81	8.99 ± 0.68	9.02 ± 0.67	8.83 ± 0.75
CIFAR100	7.50 ± 0.55	7.44 ± 0.95	7.63 ± 0.84	7.81 ± 0.67
TinyImageNet	11.84 ± 0.58	11.56 ± 0.85	11.67 ± 0.81	11.46 ± 0.82

Backbone architecture In our experiments, we use a slim version of ResNet18 (SlimResNet) for all the dataset. In Table 8, we report the results on the benchmark datasets when using a different architecture for training, i.e., ResNet32 (He et al., 2016). We can see that changing the backbone architecture does not affect the capabilities of our approach and the behaviour of all the considered methods.

Table 8: Calibration performance when using ResNet32 as backbone architecture. Results are consistent with those obtained using SlimResNet, confirming that our approach is not sensitive to the choice of the backbone architecture.

	CIFAR10 (Acc: 65.43 ± 2.09)			CIFAR100 (Acc: 44.81 ± 2.43)			TinyImageNet (Acc: 20.53 ± 1.45)		
	NLL (▼)	ECE (▼)	ΔLECE (▼)	NLL (▼)	ECE (▼)	ΔLECE (▼)	NLL (▼)	ECE (▼)	ΔLECE (▼)
Uncal	2.12 ± 0.26	27.39 ± 2.20	✗	2.93 ± 0.26	4.63 ± 0.07	✗	4.63 ± 0.07	41.41 ± 0.30	✗
RC	1.09 ± 0.07	10.70 ± 1.42	4.73 ± 1.55	2.24 ± 0.11	10.50 ± 1.10	4.20 ± 2.53	3.73 ± 0.07	13.80 ± 0.48	8.66 ± 2.98
T-CIL	1.09 ± 0.06	10.27 ± 1.01	5.33 ± 0.81	2.23 ± 0.10	9.14 ± 1.93	10.05 ± 7.17	3.74 ± 0.08	**10.23 ± 1.34**	19.52 ± 1.53
Ours	**1.08 ± 0.06**	**8.38 ± 1.51**	**-0.78 ± 0.24**	**2.21 ± 0.11**	**8.38 ± 0.64**	**-1.07 ± 1.16**	**3.71 ± 0.07**	12.18 ± 0.26	**-4.17 ± 1.66**

A.6 PSEUDOCODE OF DATS

Algorithm 1 DATS training procedure.

Input f_θ: trained model, $D_{t,val}$: validation set of task t, $\mathcal{B}_t$: calibration buffer.
Notation t: current task, T: number of tasks; C_t: set of classes in the validation set of current task; $\boldsymbol{\mathcal{C}}_{buf}$: set of classes in the calibration buffer, $\boldsymbol{S}$: distance matrix.

Class Distance Score Assignment
for $c_{buf} \in \boldsymbol{\mathcal{C}}_{buf}$ **do** ▷ For each class in $\mathcal{B}_t$
  $\boldsymbol{\mu}_{c_{buf}} \leftarrow \text{GETCLASSPROTOTYPE}(f_\theta, c_{buf})$ ▷ Compute class prototype with Eq. 4
end for
for $c_t \in C_t$ **do** ▷ For each class in C_t
  $\boldsymbol{\mu}_{c_t} \leftarrow \text{GETCLASSPROTOTYPE}(f_\theta, c_t)$ ▷ Compute class prototype with Eq. 4
end for
for $c_t \in C_t$ **do**
  for $c_{buf} \in \boldsymbol{\mathcal{C}}_{buf}$ **do**
    $\boldsymbol{S}[c_t, c_{buf}] \leftarrow \text{DISTANCE}(\boldsymbol{\mu}_{c_t}, \boldsymbol{\mu}_{c_{buf}})$ ▷ Compute pairwise distance with Eq. 5
  end for
end for
for $c_{buf} \in \boldsymbol{\mathcal{C}}_{buf}$ **do** ▷ For each class in $\mathcal{B}_t$
  $d_{c_{buf}} \leftarrow \text{ASSIGNSCORE}(c_{buf})$ ▷ Assign class distance score with Eq. 6
end for

Distance-Aware Calibration
$T(d_c) \leftarrow \text{LEARNTEMPERATURE}(d_c, \boldsymbol{z}_c)$ ▷ Compute class-wise temperatures using Eq. 7

Test-Time Calibration
$\hat{C}_{\text{test}} \leftarrow \text{ASSIGNNEARESTCLASSES}(D_{t,test}, \{\boldsymbol{\mu}_{c_{buf}}\})$ ▷ Assign test sample to nearest $\boldsymbol{\mu}_{c_{buf}}$
$\hat{C}_{\text{test}} \leftarrow \text{KEEPFREQUENTCLASSES}(\hat{C}_{\text{test}}, 0.6)$ ▷ Keep frequent classes with $\geq 60\%$ coverage
$d_{\text{test}} \leftarrow \text{COMPUTETESTDISTANCE}(\hat{C}_{\text{test}}, \{d_c\})$ ▷ Compute test distance with Eq. 8
$\text{APPLYCALIBRATION}(D_{t,test}, d_{\text{test}})$ ▷ Calibrate test logits based on d_{test}