IMPLICIT TARGET SHIFT IN ONLINE LEARNING: CHARACTERIZATION AND CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Online learning from a stream of data is a defining feature of intelligence, yet modern machine learning systems often struggle in this setting, especially under distributional shift. To understand its basic properties, we study the relationship between online and offline learning in the context of kernel regression by deriving a closed-form expression for the function learned by online kernel regression. We reveal that online kernel regression is equivalent to offline regression with shifted, inaccurate target outputs. Conversely, we show that by compensating for this implicit target shift in the teaching signal through target correction, online kernel-based learning can provably learn the same predictor as its offline counterpart. We derive both a closed-form expression for this target correction and an iterative form that can be applied sequentially. Applying this framework to continual image classification tasks on domain-incremental Split CIFAR-10 and CORe50, we show that online stochastic gradient descent with iteratively corrected targets outperforms learning with the true targets. This work therefore provides a basic framework for analyzing and improving online learning in non-stationary environments from the perspective of implicit target shift.

1 INTRODUCTION

Humans and animals can adaptively learn new skills and knowledge from streams of sensory inputs and environmental feedback. This online form of learning in non-stationary environments is becoming increasingly important in machine learning applications, including large language model fine-tuning (Luo et al., 2025; Carta et al., 2023), robotics (Lesort et al., 2020; Wołczyk et al., 2021), and medical diagnosis (Lee & Lee, 2020). However, neural network-based models typically require many epochs of updates over the same static dataset, using gradients averaged over many samples. Consequently, in the presence of large distributional shifts or continual changes in the task objective, neural networks often struggle to learn in an online manner and resulting catastrophic forgetting (McCloskey & Cohen, 1989; Hadsell et al., 2020; Luo et al., 2025).

Substantial empirical and theoretical progress has been made in understanding learning under distribution shift (Zinkevich, 2003; Gama et al., 2004; Besbes et al., 2015), particularly in the context of continual learning (French, 1991; Kirkpatrick et al., 2017; Zenke et al., 2017; Rebuffi et al., 2017; Lee et al., 2021). However, much of continual learning theory considers explicitly defined sequences of tasks (Bennani et al., 2020; Lee et al., 2021; Doan et al., 2021; Evron et al., 2022; Lin et al., 2023), leaving the relationship between sample-by-sample online learning and its offline counterpart under arbitrary sample order less well understood. This distinction is especially important in online continual learning, where data are processed sequentially, often in a single pass (Lopez-Paz & Ranzato, 2017; Aljundi et al., 2019; Mai et al., 2022). Classical online learning theory provides rigorous guarantees for sequential prediction and optimization (Zinkevich, 2003; Cesa-Bianchi et al., 2004; Rakhlin et al., 2011; Dieuleveut & Bach, 2016), but such analyses typically rely on stationarity and convexity, and thus do not directly capture the learning dynamics of neural networks under distribution shift. Consequently, we still lack a mechanistic understanding of how sequential updates under a changing data distribution bias the learned predictor relative to an offline solution trained jointly on the same data, and whether this bias can be systematically characterized and corrected.

Here, we address why online continual learning is structurally difficult, and how to mitigate this difficulty, by asking: *what kind of offline learning is an online learner actually performing?* We

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

address this question primarily in the context of kernel regression (Schölkopf & Smola, 2002; Hofmann et al., 2008) and its online formulation (Freund & Schapire, 1998; Cauwenberghs & Poggio, 2000; Kivinen et al., 2004; Engel et al., 2004; Smale & Yao, 2006). This provides a theoretically grounded framework that is intrinsically linked to the training dynamics of wide deep neural networks via the Neural Tangent Kernel (NTK) (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019), and also employed in data-limited machine learning applications (Cervantes et al., 2020; Deisenroth et al., 2013; Wilson et al., 2016).

First, we rewrite online kernel regression as a closed-form offline expression. This highlights the directional, causal sample-to-sample interactions inherent in online learning, contrasting with the bi-directional inter-sample interactions of offline batch learning. Importantly, this formulation allows us to compare online and offline learning directly. We reveal that online kernel regression is mathematically equivalent to offline kernel regression trained on shifted targets, and we provide a closed-form expression for these implicit shifts.

Second, we reveal that it is possible to reverse this implicit target shift by actively correcting the teaching signals used during online learning. Specifically, we prove that there exists a target correction that allows online kernel regression to exactly achieve the performance of its offline counterpart. While this exact correction requires a priori knowledge of all future samples, we derive an iterative approximation that respects causality. We further prove that the online-offline gap under this iterative correction is bounded by the cumulative offline learning loss and converges at $\mathcal{O}(1/n)$ rate in an independent and identically distributed (i.i.d.) setting.

Finally, we apply this target correction framework heuristically to non-linear models outside the strict kernel regime. By using corrected targets based on the evolving empirical NTK as training signals for mini-batch stochastic gradient descent (SGD), we evaluate deep neural networks on the Split CIFAR-10 (Krizhevsky, 2009) and CORe50 (Lomonaco & Maltoni, 2017) datasets. We show that in continual learning settings, target correction significantly outperforms vanilla SGD, providing a proof of principle for a counterintuitive conclusion from our theory: online continual learning can be improved by training on corrected target labels rather than the ground-truth targets.

2 Related Work

The online implementation of kernel regression has been studied extensively (Kivinen et al., 2004; Freund & Schapire, 1998; Cauwenberghs & Poggio, 2000; Engel et al., 2004; Smale & Yao, 2006). Various data sub-sampling and dictionary-building methods to keep these algorithms memory-efficient have also been proposed (Dekel et al., 2008; Richard et al., 2008; Liu et al., 2009; Dai et al., 2014). Theoretical analyses of convergence rates have demonstrated that, in the absence of temporal distribution shifts, online kernel regression is competitive with its offline counterpart under appropriate step size choices (Ying & Pontil, 2008; Tarres & Yao, 2014; Dieuleveut & Bach, 2016; Wen et al., 2025; Zhang et al., 2025). However, these works primarily focus on i.i.d. settings. In contrast, our work explicitly addresses non-stationary environments, including class-incremental learning. While online learning in non-stationary environments has been explored, particularly in online convex optimization settings (Zinkevich, 2003; Besbes et al., 2015), our approach provides a fine-grained sample level analysis of the sub-optimality, through the lens of implicit target shift.

The concept of an implicit target shift is conceptually adjacent to influence functions, a diagnostic tool used to estimate the impact of individual training points on model predictions (Hampel, 1974; Koh & Liang, 2017). The key distinction lies in the direction of the analysis: while influence functions measure how the model's error changes when a data point is perturbed, our implicit target shift measures how the data itself is effectively perturbed by the online learning process.

Our target correction framework also finds parallels in label smoothing (Szegedy et al., 2016; Müller et al., 2019) and knowledge distillation (Hinton et al., 2015), where training targets are softened to improve generalization. Unlike global target modifications in these methods, here we provide sample-level vectorized target correction to exactly reverse the bias stemming from online learning.

In the context of continual learning, our target correction is related to weight regularization methods (Kirkpatrick et al., 2017; Zenke et al., 2017) and gradient sparsification/orthogonalization methods (French, 1991; Serra et al., 2018; Bennani et al., 2020). However, the proposed target correction framework offers a conceptually different perspective: rather than constraining the weights or gra-

dients directly, we modulate the target outputs to naturally align the online trajectory with the multi-task (offline) ideal. This is also related to Dark Experience Replay (DER) (Buzzega et al., 2020), in which past predictions, rather than ground-truth targets, are replayed. Our framework provides a mathematical basis for understanding how non-ground-truth targets improve online learning and how to construct those targets.

3 Implicit Target Shift in Online Kernel Regression

To understand the nature and limitations of online learning, we focus on the comparison between online and offline learning. The offline learning solution provides a baseline that is independent of sample order, allowing us to isolate the effect of sample-by-sample learning from factors shared by the two settings, such as the hypothesis class and learning objective. In the context of continual learning, joint multi-task training (the offline ideal) is widely regarded as a theoretical upper bound for sequential learning (Hadsell et al., 2020; Mai et al., 2022; Peng et al., 2023), further motivating an understanding of how online continual learning deviates from this offline ideal. In particular, below, we investigate the following question: What is the offline learning equivalent of an online learning process?

We address this in the context of kernel regression, where the offline predictor admits a closed-form solution. By utilizing the properties of inverse triangular matrices, we reformulate the online learning rule as a modified offline learning. This reformulation reveals three key insights: (1) Online learning incurs a performance penalty because it relies on a directional kernel as opposed to the symmetric kernel employed in offline regression. (2) Online kernel regression is equivalent to offline learning with a target shift, suggesting that online updates effectively transform the teaching signals. (3) The evolution of this implicit target shift follows a simple error-based update rule: the larger the error on a new sample, the greater the label shift it induces on neighboring samples.

3.1 Offline Expression of Online Kernel Regression

Consider online learning from a sequence of inputs and target outputs $\{\boldsymbol{x}_t, \boldsymbol{y}_t\}_{t=1}^n$, where $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ and $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$. For brevity, we define the $d_x \times n$ input matrix $X_n \equiv [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ and the $d_y \times n$ target matrix $Y_n \equiv [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]$. We first focus on linear regression using a feature mapping $\phi(\boldsymbol{x})$, where $\boldsymbol{y} = W\phi(\boldsymbol{x})$. More general scenarios are discussed in Section 5. Given a mean-squared error (MSE) loss with L_2 regularization:

$$\ell(W; \boldsymbol{x}_t, \boldsymbol{y}_t) = \tfrac{1}{2}\|\boldsymbol{y}_t - W\phi(\boldsymbol{x}_t)\|^2 + \tfrac{\gamma}{2}\|W\|_F^2, \tag{1}$$

it is well known (Hofmann et al., 2008) that offline batch learning yields the following predictor for an arbitrary input $\boldsymbol{x}_*$:

$$f_{\text{off}}(\boldsymbol{x}_*; X_n, Y_n) = Y_n \left(\gamma I + K(X_n, X_n)\right)^{-1} k(X_n, \boldsymbol{x}_*), \tag{2}$$

where $k(\boldsymbol{x}, \boldsymbol{x}') \equiv \phi(\boldsymbol{x})^T\phi(\boldsymbol{x}')$ is the kernel function, $K(X_n, X_n)$ is the $n \times n$ Gram matrix with elements $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $k(X_n, \boldsymbol{x}_*)$ is the n-dimensional kernel vector, and I is the identity matrix. In contrast, an iterative sample-by-sample gradient update in the weight space follows:

$$W_{t+1} = W_t - \eta\left((W_t\phi(\boldsymbol{x}_t) - \boldsymbol{y}_t)\phi(\boldsymbol{x}_t)^T + \gamma W_t\right), \tag{3}$$

where η is a fixed learning rate. Applying this update rule recursively, we obtain:

$$W_{t+1} = \sum_{i=1}^{t} \eta \boldsymbol{y}_i \boldsymbol{\phi}_i^T \prod_{j=i+1}^{t} \left[(1-\eta\gamma)I - \eta\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T\right] + W_1 \prod_{j=1}^{t}\left[(1-\eta\gamma)I - \eta\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T\right], \tag{4}$$

where $\boldsymbol{\phi}_i \equiv \phi(\boldsymbol{x}_i)$ and W_1 is the initial weight matrix, and we used the convention $\prod_{j=a}^{b} A_j = A_a A_{a+1} \ldots A_b$. Using the properties of the inverse of an upper triangular matrix, we derive a closed-form solution for the predictor resulting from this online rule (see Appendix A.2 for the derivation).

Lemma 3.1. *Given a sequence of data (X_n, Y_n), the online learning predictor $f_{on}(\boldsymbol{x}_*; X_n, Y_n) \equiv W_{n+1}\phi(\boldsymbol{x}_*)$ after n update steps from $W_1 = 0$ follows:*

$$f_{on}(\boldsymbol{x}_*; X_n, Y_n) = Y_n D_n \left(\tfrac{1}{\eta} I + \tfrac{1}{1-\eta\gamma} K^U(X_n, X_n)\right)^{-1} k(X_n, \boldsymbol{x}_*), \tag{5}$$

where $K^U(X_n, X_n)$ is the strictly upper-triangular portion of the Gram matrix $K(X_n, X_n)$, and D_n is a diagonal matrix with $[D_n]_{ii} = (1-\eta\gamma)^{n-i}$.

Note that, the iterative formulation of online kernel regression is well established (Cauwenberghs & Poggio, 2000; Kivinen et al., 2004; Engel et al., 2004; Smale & Yao, 2006). This lemma instead derives an offline-equivalent closed-form expression for the predictor learned by online kernel regression.

Equation 5 implies that regularization in the online setting is primarily driven by the learning rate factor $1/\eta$, as shown previously (Ying & Pontil, 2008; Barrett & Dherin, 2020). Hence, for simplicity, we set $\gamma = 0$ for online learning in the following analysis. The expression then becomes:

$$f_{\text{on}}(\boldsymbol{x}_*) = Y_n \left(\tfrac{1}{\eta} I + K^U(X_n, X_n)\right)^{-1} k(X_n, \boldsymbol{x}_*). \tag{6}$$

Comparing Eq. 6 with Eq. 2 highlights the fundamental distinction between online and offline regimes. In batch regression, the inverse term depends on the full similarity matrix $K(X_n, X_n)$. Conversely, the online predictor uses only the strictly upper-triangular components $K^U(X_n, X_n)$. This is intuitively interpretable: since an online model lacks access to future samples, it can only incorporate similarities between the current sample and its predecessors, resulting in a directional kernel. In other words, for sample x_t, $k(x_t, x_{t'})$ is available for $t' \leq t$ (past samples), but not for $t' > t$ (future samples). This lack of bidirectional information flow provides a clear mathematical intuition for why online learning is inherently more constrained.

Finally, as shown in Appendix A.3, this framework extends to mini-batch learning with size b:

$$f_b(\boldsymbol{x}_*) = Y_n \left(\tfrac{1}{\eta} I + K^{bU}(X_n, X_n)\right)^{-1} k(X_n, \boldsymbol{x}_*), \tag{7}$$

where K^{bU} is a strictly upper-triangular matrix where the $b \times b$ block-diagonal components are also zeroed out. This reflects the fact that the model cannot leverage correlations between samples within the same mini-batch during a single update step.

3.2 Implicit Target Shift

The structural similarity between the online kernel regression expression (Eq. 6) and its offline counterpart (Eq. 2) allows us to introduce the concept of an implicit target shift. We define the effective targets Y_n^e as follows:

$$Y_n^e \equiv Y_n \left(\tfrac{1}{\eta} I + K^U(X_n, X_n)\right)^{-1} \left(\gamma I + K(X_n, X_n)\right). \tag{8}$$

Using this definition, we establish the following equivalence:

Theorem 3.2. *Assuming $\eta, \gamma > 0$, the predictor f_{on} obtained through online gradient descent $W_{t+1} = W_t - \eta(W_t\phi(\boldsymbol{x}_t) - \boldsymbol{y}_t)\phi(\boldsymbol{x}_t)^T$ with $W_1 = 0$ on data (X_n, Y_n) is identical to the offline kernel regression predictor f_{off} trained on the data with modified target (X_n, Y_n^e). That is:*

$$f_{on}(\boldsymbol{x}; X_n, Y_n) = f_{off}(\boldsymbol{x}; X_n, Y_n^e). \tag{9}$$

Proof: By substituting the definition of Y_n^e from Eq. 8 into the online predictor formula (Eq. 6):

$$\begin{aligned} f_{\text{on}}(\boldsymbol{x}_*; X_n, Y_n) &= Y_n \left(\tfrac{1}{\eta} I + K^U(X_n, X_n)\right)^{-1} k(X_n, \boldsymbol{x}_*) \\ &= Y_n^e \left(\gamma I + K(X_n, X_n)\right)^{-1} k(X_n, \boldsymbol{x}_*) \\ &= f_{\text{off}}(\boldsymbol{x}_*; X_n, Y_n^e). \end{aligned} \tag{10}$$

This theorem demonstrates that the sub-optimality of online kernel learning can be fully captured by a transformation of the target outputs from Y_n to Y_n^e. Note that the online predictor f_{on} in Theorem 3.2 is defined without an explicit regularizer (Eq. 6). The parameter γ in Eq. 8 originates solely from the offline predictor f_{off}. Consequently, the effective targets Y_n^e depend directly on the regularization strength γ of the specific offline baseline that the online model effectively mimics.

Using the effective target Y_n^e, the online-offline gap is written as $f_{\text{on}}(\boldsymbol{x}; X_n, Y_n) - f_{\text{off}}(\boldsymbol{x}; X_n, Y_n) = (Y_n^e - Y_n)(\gamma I + K(X_n, X_n))^{-1} k(X_n, \boldsymbol{x})$. Thus, the discrepancy is upper bounded by the target shift: $\|f_{\text{on}}(\,\cdot\,; X_n, Y_n) - f_{\text{off}}(\,\cdot\,; X_n, Y_n)\|_{\mathcal{H}} \leq \frac{1}{2\sqrt{\gamma}} \|Y_n^e - Y_n\|_F$, where $\mathcal{H}$ is the Reproducing Kernel Hilbert Space (RKHS) induced by kernel k.

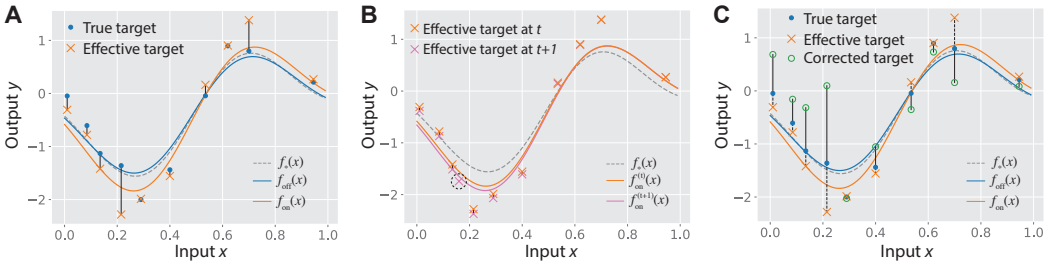


Figure 1: Illustration of implicit target shift and its correction. **(A)** Example of implicit target shift. Blue and orange points correspond to the true target y_i and effective target y_i^e, respectively. **(B)** Shift in the effective target from one-step update (with the sample marked by a dashed black circle). Pink points on the left side shift downward from orange points, while they overlap on the right side. **(C)** Target correction (green circles) for the implicit target shift (orange x-marks) illustrated in panel A. See main text and Appendix C.1 for details.

Figure 1A illustrates this phenomenon in a one-dimensional regression task using data generated from a Gaussian Process (GP; see Appendix C.1). While the offline predictor (blue line) successfully recovers the true underlying function (gray line) from noisy samples (blue points), the online predictor (orange line) exhibits significant deviations. The effective target shift provides a lens for interpreting these errors: the orange crosses represent the effective targets y_i^e. In regions where $f_{\text{on}}(x)$ diverges from the batch solution, the effective targets have been shifted away from the true labels. Because the online predictor is mathematically equivalent to a batch regressor trained on these "distorted" targets, the resulting function inherits these deviations.

3.3 One-step Update of Implicit Target Shifts

To understand the mechanism by which these distortions emerge, we analyze the dynamics of the target shift. Let $\boldsymbol{y}_i^e(n)$ denote the effective target of the i-th sample after n updates. Upon receiving the $(n+1)$-th sample, the effective targets evolve as follows (see Appendix A.4 for the derivation):

$$\boldsymbol{y}_i^e(n+1) = \begin{cases} \boldsymbol{y}_i^e(n) - \eta \boldsymbol{e}_{n+1} k(\boldsymbol{x}_{n+1}, \boldsymbol{x}_i) & i \leq n \\ \boldsymbol{y}_{n+1} - (\eta[\gamma + k_{n+1}] - 1)\boldsymbol{e}_{n+1} & i = n+1, \end{cases} \tag{11}$$

where $k_{n+1} = k(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1})$ and $\boldsymbol{e}_{n+1} \equiv f_{\text{on}}(\boldsymbol{x}_{n+1}; X_n, Y_n) - \boldsymbol{y}_{n+1}$ is the prediction error on $\boldsymbol{x}_{n+1}$ from data (X_n, Y_n). For previous samples ($i \leq n$), the shift is proportional to the prediction error $\boldsymbol{e}_{n+1}$ and the kernel similarity $k(\boldsymbol{x}_{n+1}, \boldsymbol{x}_i)$. This indicates that a poor prediction on a new point biases the effective targets of its neighbors in the feature space.

Figure 1B visualizes this one-step shift. When a new sample (marked by a dashed circle) is added to the dataset from Fig. 1A, it induces a downward shift in the effective targets of its neighbors (left side), while leaving distant points unaffected (right side). This framework thus allows us to precisely localize where and how online predictors deviate from the offline ideal, providing a potential diagnostic tool for online learning performance.

4 Target Correction

The concept of implicit target shift introduced in the previous section helps identify which samples accumulate excess error during online learning, but it also raises a question of how to actively correct this shift. In this section, we investigate the correction of the implicit target shift. First, we demonstrate the existence of a set of corrected targets that allows an online learner to exactly recover the offline regression predictor, providing its closed-form expression. We then derive an iterative formulation of this target correction that stabilizes the learning dynamics and restores practical causality.

By inverting the relationship defining the implicit target shift (using Eqs. 2 and 6), we define the corrected targets Y_n^c as:

$$Y_n^c \equiv Y_n \left(\gamma I + K(X_n, X_n)\right)^{-1} \left(\tfrac{1}{\eta} I + K^U(X_n, X_n)\right). \tag{12}$$

Theorem 4.1. *Assuming $\eta, \gamma > 0$, the predictor obtained via online kernel-based learning $W_{t+1} = W_t - \eta(W_t\phi(\boldsymbol{x}_t) - \boldsymbol{y}_t^c)\phi(\boldsymbol{x}_t)^T$ starting from $W_1 = 0$ on the data (X_n, Y_n^c) is mathematically equivalent to the offline learning predictor trained on the original data (X_n, Y_n). Namely:*

$$f_{on}(\boldsymbol{x}; X_n, Y_n^c) = f_{off}(\boldsymbol{x}; X_n, Y_n). \tag{13}$$

The proof follows straightforwardly from a parallel argument to Theorem 3.2. This result implies that if the true targets are known a priori, one can construct a set of "distorted" targets Y_n^c with which an online learner can provably match offline performance. Counterintuitively, this means that an online model can achieve better performance by training on intentionally altered targets rather than the ground-truth targets.

Figure 1C illustrates this correction for the 1D regression task previously shown in Figure 1A. In the left-hand region, where the effective targets (orange crosses) are erroneously shifted downward by the online process, the corrected targets (green open circles) introduce an upward bias to cancel the induced error. Conversely, on the right side, the corrected targets impose a downward correction. Theorem 4.1 ensures that these counter-biases exactly align the online learning trajectory with the offline baseline, meaning that the online predictor constructed with green corrected points exactly matches the blue curve, $f_{\text{off}}(x)$. As with target shift, we can also analyze the step-wise evolution of these corrections to understand their directionality (see Appendix A.5).

4.1 Iterative Target Correction

While the exact target correction guarantees offline performance, its construction (Eq. 12) requires full access to the entire dataset a priori, breaking the causal requirement of online learning. Furthermore, enforcing this exact correction can result in highly unstable learning dynamics (as will be demonstrated in Fig. 2C and D). To address these limitations, we derive an iterative formulation.

We first cast the corrected targets (Eq. 12) as the solution to a minimization problem. Given an arbitrary set of targets $Z \in \mathbb{R}^{d_y \times n}$, the online prediction is $f_{\text{on}}(\boldsymbol{x}; X, Z) = Z(\frac{1}{\eta}I + K^U(X,X))^{-1}k(X,\boldsymbol{x})$. Thus, the discrepancy between the online and offline predictors is minimized when:

$$Z = \arg\min_{Z} \|Z(\tfrac{1}{\eta}I + K^U(X,X))^{-1}k(X,\cdot) - Y(\gamma I + K(X,X))^{-1}k(X,\cdot)\|_{\mathcal{H}}^2. \tag{14}$$

Solving this recovers the original target correction (Eq. 12; see Appendix A.6).

To adapt this to an iterative, causal setting, we optimize targets in sequential chunks. Consider the optimization of novel targets $Z_{\text{new}} = \{\boldsymbol{z}_{N-b+1}, \ldots, \boldsymbol{z}_N\}$ for incoming inputs $X_{\text{new}} = \{\boldsymbol{x}_{N-b+1}, \ldots, \boldsymbol{x}_N\}$, conditioned on past inputs $X_{\text{past}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-b}\}$ and their frozen corrected targets $Z_{\text{past}} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N-b}\}$. Here, b is the look-ahead horizon (or mini-batch size) over which targets are simultaneously optimized. While we still assume a sample-by-sample online update for the actual weight learning, the targets themselves are generated iteratively in blocks of size b.

Denoting the union of inputs as $X_{\text{tot}} = X_{\text{past}} \cup X_{\text{new}}$, the block-wise loss function becomes:

$$\begin{aligned}\ell(Z_{\text{new}}; Z_{\text{past}}, X_{\text{tot}}) = \frac{1}{2}\left\|[Z_{\text{past}}, Z_{\text{new}}](\tfrac{1}{\eta}I + K_{\text{tot}}^U)^{-1}k(X_{\text{tot}},\cdot) - Y_{\text{tot}}(\gamma I + K_{\text{tot}})^{-1}k(X_{\text{tot}},\cdot)\right\|_{\mathcal{H}}^2 \\ + \frac{\gamma_o}{2}\left\|[Z_{\text{past}}, Z_{\text{new}}](\tfrac{1}{\eta}I + K_{\text{tot}}^U)^{-1}\right\|_F^2,\end{aligned} \tag{15}$$

where $Y_{\text{tot}} (= Y_{\text{past}} \cup Y_{\text{new}})$ represents the true labels for the accumulated data, and the second term is a Tikhonov regularizer ($\gamma_o > 0$) introduced to enforce stability. Minimizing this objective yields the following update rule:

Proposition 4.2. *Minimization of Eq. 15 yields a unique solution for Z_{new} given $\gamma, \eta, \gamma_o > 0$:*

$$Z_{new} = Y_{new} + (Y_{new} - f_{on}(X_{new}; X_{past}, Z_{past}))C_{on} + (f_{off}(X_{new}; X_{past}, Y_{past}) - Y_{new})C_{off}, \tag{16}$$

where the coefficient matrices are defined as:

$$C_{on} \equiv \left((\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K_{nn}^U) - I\right), \tag{17a}$$

$$C_{off} \equiv \gamma\left([\gamma I + K_{nn}] - K_{np}[\gamma I + K_{pp}]^{-1}K_{pn}\right)^{-1}(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K_{nn}^U), \tag{17b}$$

with $K_{pp} = K(X_{past}, X_{past})$ and $K_{np} = K(X_{new}, X_{past})$.

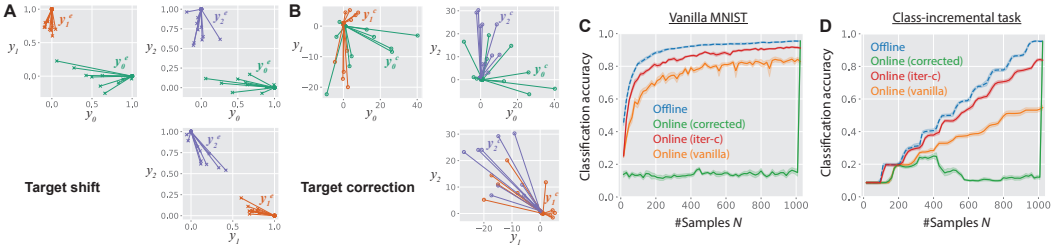


Figure 2: Effective target shifts and their correction in NTK regression applied to MNIST. **A, B)** Visualization of effective target shifts (A) and their correction (B). Three panels depict the first three dimensions of the output space corresponding to labels 0, 1, and 2. For instance, green lines in the top-left panel in A depict how the one-hot vector of label 0, $\boldsymbol{y}_0 = [1, 0, 0, ...]$, exhibits effective shifts under online NTK regression (10 lines represent 10 training samples belonging to label 0). **C)** Learning curves of offline (blue) and online kernel regression with the true (orange), corrected (green), and iteratively-corrected (red) targets. **D)** The same as C, but in a class-incremental setting. In panels C and D, accuracy was measured by test accuracy, and shaded areas are the standard error of the mean (SEM) over 10 random seeds. See Appendix C.2 for details.

See Appendix A.7 for the full derivation. This formulation disentangles two mechanisms of correction. The first term, weighted by C_{on}, acts as a sample-wise learning rate modulation driven by the online prediction error. This can be seen by rearranging Eq. 16 to highlight the modified residual:

$$Z_{\text{new}} - f_{\text{on}} = (Y_{\text{new}} - f_{\text{on}})(I + C_{\text{on}}) + (f_{\text{off}} - Y_{\text{new}})C_{\text{off}}, \tag{18}$$

where f_{on} and f_{off} represent the online and offline predictions on X_{new}, respectively. The second term, weighted by C_{off}, injects the offline estimation structure into the targets. Crucially, the offline predictor f_{off} in this formulation relies only on X_{past}, X_{new}, and Y_{past}, circumventing the causality bottleneck, and allowing for real-time target shifts without requiring future labels.

Although the iterative target correction no longer guarantee the exact recovery of the offline predictor, the following two results hold on the discrepancy between online and offline predictors under this iterative correction.

Proposition 4.3. *Consider iterative target correction with data* $X_{1:n} = [X_1, ..., X_n]$ *and* $Y_{1:n} = [Y_1, ..., Y_n]$. *Then, under* $\gamma > 0$ *and* $\gamma_o > 0$, *the online-offline gap is bounded by:*

$$\|f_{on}(\,\cdot\,; X_{1:n}, Z_{1:n}) - f_{off}(\,\cdot\,; X_{1:n}, Y_{1:n})\|_{\mathcal{H}} \leq \frac{1}{\sqrt{\gamma}} \sum_{i=1}^{n} \|Y_i - f_{off}(X_i; X_{1:i-1}, Y_{1:i-1})\|_F \,. \tag{19}$$

The proof is provided in Appendix A.10. Thus, even for an arbitrary data sequence, the online–offline gap under iterative target correction is bounded by the cumulative prediction error of predictors trained on all preceding samples. By contrast, for naive online learning, the corresponding gap depends on the minimum singular value of the directional kernel, $\sigma_{\min}^{-1}\left(\gamma I + K^U(X_{1:n}, X_{1:n})\right)$, which need not be lower bounded by $1/\gamma$ due to non-normality (see Eq. 85 in Appendix).

Moreover, under an i.i.d. setting with additional constraints on data and kernel (see Assumptions A.11), the online-offline gap converges at $\mathcal{O}(1/n)$ rate, supporting the soundness of the algorithm:

Proposition A.5 (informal): Under Assumptions A.11, the norm of online-offline gap follows,

$$\mathbb{E}[\|f_{on}(\,\cdot\,; X_{1:n}, Z_{1:n}) - f_{\text{off}}(\,\cdot\,; X_{1:n}, Y_{1:n})\|_{\mathcal{H}}^2] \leq C\left(\frac{1}{n} + \frac{b}{2}(1-\alpha)^{n/2}\right) \tag{20}$$

where C and α $(0 < \alpha < 1)$ are constants that depends on the kernel, probability distribution, and the target function. The full statement and the proof are provided in Appendix A.11.

4.2 Example: Conv-NTK Regression on MNIST

Figure 2 demonstrates iterative target correction on the MNIST image recognition task, utilizing the NTK of a shallow, infinite-width convolutional neural network (Novak et al., 2020). Following

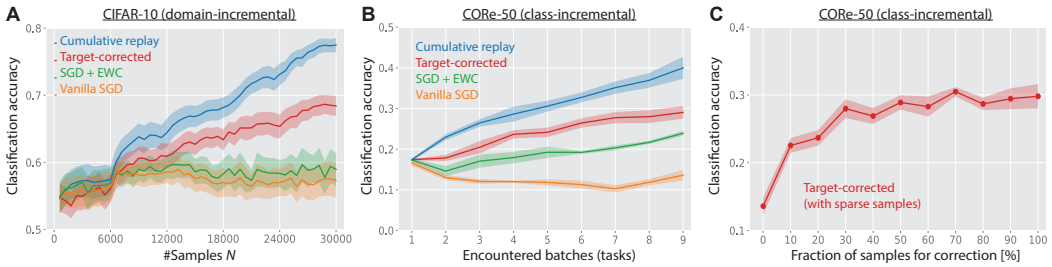


Figure 3: Application of iterative target correction to SGD training of nonlinear neural networks. **(A)** Domain-incremental setting where 10 categories from CIFAR-10 are split into five binary tasks. We used a single head for all five tasks to make the task challenging. **(B)** CORe-50 dataset, a class-incremental continual learning task with 50 categories in total, split into 9 tasks. **(C)** Final accuracy on CORe50 as a function of the fraction of samples used for target correction. See Appendix C.3 for implementation details.

standard NTK practice, we framed the classification as a regression problem via mean-squared error (Jacot et al., 2018; Arora et al., 2019; Liu et al., 2020).

Visualizing the implicit target shift in the 3D subspace spanned by the digits '0', '1', and '2', we observed that online learning consistently causes a shrinkage of the target vectors toward the origin (Fig. 2A; filled circles are true one-hot targets, x-marks are effectively shifted targets after 1024 online updates). The target correction compensates for this shrinkage by proactively pushing the target vectors apart (Fig. 2B). Interestingly, the correction flips the signs of certain sample targets entirely, highlighting the non-trivial geometry of optimal label correction.

When evaluating test accuracy (Fig. 2C), vanilla online learning underperformed the offline NTK baseline (orange vs. blue lines), as expected. Training sequentially with the exact corrected labels (Eq. 12) eventually matches the offline NTK (green line), but suffered from instability during intermediate learning phases. In contrast, the iterative target correction ("iter-c"; red line) maintained near-offline stability and performance throughout the entire learning trajectory. Note that both offline γ and online η were optimized for peak final accuracy (see Fig. 5 in Appendix), confirming that the iterative correction's advantage stems from the sample-wise modulation via C_{on} and C_{off} rather than global hyperparameter tuning.

Finally, we evaluated target correction in a highly non-stationary class-incremental learning setting, presenting all samples belonging to label '0' (approx. 100 samples) before moving to label '1', and so on. Under this adversarial trial order, vanilla SGD struggled to learn the task effectively due to forgetting. However, learning under iterative target correction performed close to offline levels (Fig. 2D), demonstrating its resistance to temporal distribution shifts.

5 Application of Target Correction to Non-linear Models

Building on the offline-equivalent expression of online kernel regression, we introduced the concept of implicit target shift and showed that iterative target correction can improve online learning performance. We now extend this framework beyond the linear kernel regime to online mini-batch SGD in non-linear models. Because features evolve during deep network training (Fort et al., 2020; Geiger et al., 2020; Radhakrishnan et al., 2024), a closed-form, static target correction is no longer theoretically guaranteed. Nevertheless, we hypothesized that applying the correction heuristically using an evolving empirical kernel approximation could still improve online learning, particularly in continual learning settings.

To adapt target correction to non-linear networks, we computed the empirical NTK, $\nabla_{\boldsymbol{w}} f^T \nabla_{\boldsymbol{w}} f$, directly from the network and substituted it into Eq. 16 to calculate corrected targets. To account for feature learning while limiting computational cost, we updated the empirical NTK only at the beginning of each new task and, for multi-epoch learning, at the beginning of each within-task epoch. We used mini-batch SGD throughout (see Appendix C.3 for full implementation details).

We first evaluated iterative target correction in a domain-incremental continual learning setting based on CIFAR-10 (Krizhevsky, 2009), where the dataset was split into five binary classification tasks solved sequentially with a single head. Online learning with iterative target correction significantly outperformed both vanilla SGD and the Elastic Weight Consolidation (EWC) baseline (Kirkpatrick et al., 2017) (Fig. 3A; red vs. orange and green lines), although its performance was naturally bounded below that of cumulative replay, where the model was trained jointly on all previously seen data (red vs. blue). Learning rates and regularization strengths were optimized independently for each condition, supporting the robustness of this advantage (see Fig. 7 in Appendix). Target correction with a fixed NTK also outperformed vanilla SGD and EWC in this setting (Fig. 6A).

To test whether this observation extends beyond the artificial CIFAR-10 task decomposition, we next evaluated the method on the CORe50 continual learning benchmark (Lomonaco & Maltoni, 2017) (Fig. 3B). While the absolute performance was limited by our simplified architectural setup, and the performance degradation was observed in vanilla SGD (orange line) due to task difficulty as noted in the original CORe50 benchmark (Lomonaco & Maltoni, 2017), the relative advantage of the corrected targets remained consistent. Additionally, target correction with sparse sampling of past data (30%) was sufficient (Fig. 3C), mitigating the $\mathcal{O}(n^3)$ computational cost of updating and inverting the empirical NTK over past data.

The performance gain of target correction was not explained by the soft-labeling effect. Moreover, the combination of target correction with DER improved replay performance over vanilla DER (Table 1 in Appendix), supporting the benefit of sample-level vectorized target correction beyond vanilla online learning. Single-pass learning with sparse target correction did not outperform replay-based methods that use the same samples directly for replay rather than for target correction (Table 1), but required only half as many network weight updates because it does not perform replay.

Together, these results indicate that iterative target correction can improve online learning in nonlinear networks across multiple continual learning settings. We emphasize, however, that in its current formulation, iterative target correction is not intended as a practical, drop-in replacement for state-of-the-art continual learning algorithms. Rather, these experiments serve as a proof of principle for our theoretical insight from kernel regression: an online model can achieve better predictive performance when trained on corrected targets than when trained directly on the ground-truth targets.

6 Discussion

Overall, this work presents a fundamental framework to analyze the discrepancy between online and offline learning in feature-based learning settings, providing two core theorems on online-to-offline conversion. Furthermore, it proposes target correction methods for efficient online learning with theoretical bounds on the online–offline gap, which we applied to continual image classification tasks as a proof of principle.

While Theorem 4.1 guarantees that the online learner can exactly match the offline predictor, this naturally implies that generalization performance remains bottlenecked by the quality of the chosen offline regularizer, γ. More broadly, strictly enforcing exact offline equivalence removes the implicit regularization of mini-batch SGD noise, which helps networks escape sharp minima and achieve better generalization (Keskar et al., 2016; Zhu et al., 2018b).

Nevertheless, beyond conceptual interest, the target correction framework may be beneficial, particularly when modifying the network weights is the primary bottleneck and replay-based methods are therefore costly. For example, in an edge-computing setting (Chen & Ran, 2019), corrected targets could be computed on a central server with greater computational resources, while the edge device performs only an online single-pass SGD update using the corrected target supplied by the server. Similar advantage is expected in a machine teaching setting where the teacher needs to optimize its feedback to the student (Zhu et al., 2018a), and also in *in situ* training of optical neural networks where weight update requires a physical reconfiguration of the system (Buckley et al., 2023).

The exact formulation of effective target shift and correction also holds promising potential for applications in learning process diagnosis (Koh & Liang, 2017), knowledge distillation (Hinton et al., 2015), and reward shaping (Dorigo & Colombetti, 1994; Ng et al., 1999). More generally, our work establishes a new mathematical lens through which to analyze online learning, opening up a promising line of both theoretical and empirical investigation.

AI USE STATEMENT

We used generative AI for improving the clarity and readability of the manuscript and to assist with literature survey. We also used generative AI tools to assist with code editing and debugging. We have reviewed all AI-assisted work. All AI-assisted code was reviewed and tested by the authors before use in the reported experiments. We take responsibility for the final content of this work, including text, code, or claims produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide complete proofs of the theoretical results in the appendix. Detailed experimental settings, including model architectures, hyperparameters, kernel and NTK construction, and additional implementation details, are also provided in the appendix. All experiments are conducted using publicly available datasets. URL of anonymized source code for reproducing the main experiments is provided in the Appendix.

REFERENCES

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

David GT Barrett and Benoit Dherin. Implicit gradient regularization. *arXiv preprint arXiv:2009.11162*, 2020.

Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. *arXiv preprint arXiv:2006.11942*, 2020.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Non-stationary stochastic optimization. *Operations research*, 63 (5):1227–1244, 2015.

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*, 45(5):5497–5512, 2022.

Sonia Mary Buckley, Alexander N Tait, Adam N McCaughan, and Bhavin J Shastri. Photonic online learning: a perspective. *Nanophotonics*, 12(5):833–845, 2023.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International conference on machine learning*, pp. 3676–3713. PMLR, 2023.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, 13, 2000.

Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408: 189–215, 2020.

Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8): 1655–1674, 2019.

Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. *Advances in neural information processing systems*, 27, 2014.

Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2008.

Aymeric Dieuleveut and Francis Bach. Nonparametric stochastic approximation with large step-sizes. *The Annals of Statistics*, pp. 1363–1399, 2016.

Thang Doan, Mehdi Abbana Bennani, Bogdan Mazoure, Guillaume Rabusseau, and Pierre Alquier. A theoretical analysis of catastrophic forgetting through the ntk overlap matrix. In *International Conference on Artificial Intelligence and Statistics*, pp. 1072–1080. PMLR, 2021.

Marco Dorigo and Marco Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial intelligence*, 71(2):321–370, 1994.

Yaakov Engel, Shie Mannor, and Ron Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on signal processing*, 52(8):2275–2285, 2004.

Itay Evron, Edward Moroshko, Rachel Ward, Nathan Srebro, and Daniel Soudry. How catastrophic can catastrophic forgetting be in linear regression? In *Conference on Learning Theory*, pp. 4028–4079. PMLR, 2022.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

Robert M French. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. In *Proceedings of the 13th annual cognitive science society conference*, volume 1, pp. 173–178, 1991.

Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 209–217, 1998.

Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pp. 286–295. Springer, 2004.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, 2020.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning1. *The Annals of Statistics*, 36(3):1171–1220, 2008.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Jyrki Kivinen, Alexander J Smola, and Robert C Williamson. Online learning with kernels. *IEEE transactions on signal processing*, 52(8):2165–2176, 2004.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Cecilia S Lee and Aaron Y Lee. Clinical applications of continual learning machine learning. *The Lancet Digital Health*, 2(6):e279–e281, 2020.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

Sebastian Lee, Sebastian Goldt, and Andrew Saxe. Continual learning in the teacher-student setup: Impact of task similarity. In *International Conference on Machine Learning*, pp. 6109–6119. PMLR, 2021.

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.

Sen Lin, Peizhong Ju, Yingbin Liang, and Ness Shroff. Theory on forgetting and generalization of continual learning. In *International Conference on Machine Learning*, pp. 21078–21100. PMLR, 2023.

Chaoyue Liu, Libin Zhu, and Misha Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33:15954–15964, 2020.

Weifeng Liu, Il Park, and Jose C Principe. An information theoretic approach of designing sparse kernel adaptive filters. *IEEE transactions on neural networks*, 20(12):1950–1961, 2009.

Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on robot learning*, pp. 17–26. PMLR, 2017.

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

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

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.

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

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL `https://github.com/google/neural-tangents`.

Liangzu Peng, Paris Giampouras, and René Vidal. The ideal continual learner: An agent that never forgets. In *International Conference on Machine Learning*, pp. 27585–27610. PMLR, 2023.

Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

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*, pp. 2001–2010, 2017.

Cédric Richard, José Carlos M Bermudez, and Paul Honeine. Online prediction of time series data with kernels. *IEEE Transactions on Signal Processing*, 57(3):1058–1067, 2008.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pp. 4548–4557. PMLR, 2018.

Steve Smale and Yuan Yao. Online learning algorithms. *Foundations of computational mathematics*, 6(2): 145–170, 2006.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Pierre Tarres and Yuan Yao. Online learning as stochastic approximation of regularization paths: Optimality and almost-sure convergence. *IEEE Transactions on Information Theory*, 60(9):5716–5735, 2014.

Hongwei Wen, Annika Betken, and Hanyuan Hang. Optimal learning of kernel logistic regression for complex classification scenarios. In *The Thirteenth International Conference on Learning Representations*, 2025.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

Maciej Wołczyk, Michał Zajac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 28496–28510, 2021.

Yiming Ying and Massimiliano Pontil. Online gradient descent learning algorithms. *Foundations of Computational Mathematics*, 8(5):561–596, 2008.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

Haihan Zhang, Weicheng Lin, Yuanshi Liu, and Cong Fang. Learning curves of stochastic gradient descent in kernel regression. *arXiv preprint arXiv:2505.22048*, 2025.

Tong Zhang. Leave-one-out bounds for kernel methods. *Neural computation*, 15(6):1397–1437, 2003.

Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018a.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. *arXiv preprint arXiv:1803.00195*, 2018b.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

A PROOFS AND DERIVATIONS

A.1 MATHEMATICAL NOTATION

Throughout the manuscript, we used lower-case italic letters for scalar variables, lower-case bold-italic letters for vectors, and upper-case letters for matrices. Vectors are defined as column vectors; thus $\boldsymbol{a}^T\boldsymbol{b}$ and $\boldsymbol{a}\boldsymbol{b}^T$ represent the inner and outer products of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. For a vector $\boldsymbol{v}$, $[\boldsymbol{v}]_i$ represents the i-th element of the vector, and for a matrix M, $[M]_{ij}$ represents the (i,j)-th element of the matrix.

A.2 PROOF OF LEMMA 3.1

From Eq. 4, the function obtained by online learning from data (X_n, Y_n) obeys

$$\begin{aligned} f_{on}(\boldsymbol{x}_*; X_n, Y_n) &= W_{n+1}\phi(\boldsymbol{x}_*) \\ &= \sum_{i=1}^{n} \eta \boldsymbol{y}_i \boldsymbol{\phi}_i^T \prod_{j=i+1}^{n} \left[(1-\eta\gamma)I - \eta\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T\right]\boldsymbol{\phi}_* + W_1 \prod_{j=1}^{n}\left[(1-\eta\gamma)I - \eta\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T\right]\boldsymbol{\phi}_*, \end{aligned} \tag{21}$$

where $\boldsymbol{\phi}_* \equiv \phi(\boldsymbol{x}_*)$. The first term is rewritten as

$$\begin{aligned} &\boldsymbol{\phi}_i^T \prod_{j=i+1}^{n}\left[(1-\eta\gamma)I - \eta\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T\right]\boldsymbol{\phi}_* \\ &= (1-\eta\gamma)^{n-i}k(\boldsymbol{x}_i,\boldsymbol{x}_*) + \sum_{\ell=1}^{n-i}(1-\eta\gamma)^{n-i-\ell}(-\eta)^\ell \sum_{i+1\le j_1<\ldots<j_\ell\le n} k(\boldsymbol{x}_i,\boldsymbol{x}_{j_1})k(\boldsymbol{x}_{j_1},\boldsymbol{x}_{j_2})\ldots k(\boldsymbol{x}_{j_\ell},\boldsymbol{x}_*) \\ &= (1-\eta\gamma)^{n-i}k(\boldsymbol{x}_i,\boldsymbol{x}_*) + \sum_{\ell=1}^{n-i}(1-\eta\gamma)^{n-i-\ell}(-\eta)^\ell \left[\left(K^U(X_n,X_n)\right)^\ell k(X_n,\boldsymbol{x}_*)\right]_i, \end{aligned} \tag{22}$$

where $[\boldsymbol{v}]_i$ represents the i-th element of vector $\boldsymbol{v}$. In the last line, we define an $n\times n$ matrix $K^U(X_n,X_n)$ as the strictly upper-triangular component of the kernel matrix:

$$\left[K^U(X_n,X_n)\right]_{ij} = \begin{cases} k(\boldsymbol{x}_i,\boldsymbol{x}_j) & (i<j) \\ 0 & (i\ge j). \end{cases} \tag{23}$$

The last line follows because K^U satisfies

$$\begin{aligned} &\left[\left(K^U(X_n,X_n)\right)^\ell k(X_n,\boldsymbol{x}_*)\right]_i \\ &= \sum_{j_1=1}^{n}\sum_{j_2=1}^{n}\ldots\sum_{j_\ell=1}^{n} K^U_{ij_1}K^U_{j_1j_2}\ldots K^U_{j_{\ell-1}j_\ell}k(\boldsymbol{x}_{j_\ell},\boldsymbol{x}_*) \\ &= \sum_{j_1=i+1}^{n}\sum_{j_2=j_1+1}^{n}\ldots\sum_{j_\ell=j_{\ell-1}+1}^{n} k(\boldsymbol{x}_i,\boldsymbol{x}_{j_1})k(\boldsymbol{x}_{j_1},\boldsymbol{x}_{j_2})\ldots k(\boldsymbol{x}_{j_{\ell-1}},\boldsymbol{x}_{j_\ell})k(\boldsymbol{x}_{j_\ell},\boldsymbol{x}_*) \\ &= \sum_{i+1\le j_1<j_2<\ldots<j_\ell\le n} k(\boldsymbol{x}_i,\boldsymbol{x}_{j_1})k(\boldsymbol{x}_{j_1},\boldsymbol{x}_{j_2})\ldots k(\boldsymbol{x}_{j_{\ell-1}},\boldsymbol{x}_{j_\ell})k(\boldsymbol{x}_{j_\ell},\boldsymbol{x}_*). \end{aligned} \tag{24}$$

Therefore, the coefficient of $\boldsymbol{y}_i$ becomes

$$\begin{aligned}
&\boldsymbol{\phi}_i^T \prod_{j=i+1}^{n} \left[(1-\eta\gamma)I - \eta\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T\right] \boldsymbol{\phi}_* \\
&= (1-\eta\gamma)^{n-i}\left(k(\boldsymbol{x}_i,\boldsymbol{x}_*) + \sum_{\ell=1}^{n-i}\left(\tfrac{-\eta}{1-\eta\gamma}\right)^{\ell}\left[\left(K^U(X_n,X_n)\right)^{\ell}k(X_n,\boldsymbol{x}_*)\right]_i\right) \\
&= (1-\eta\gamma)^{n-i}\sum_{\ell=0}^{n-i}\left(\tfrac{-\eta}{1-\eta\gamma}\right)^{\ell}\left[\left(K^U(X_n,X_n)\right)^{\ell}k(X_n,\boldsymbol{x}_*)\right]_i \\
&= (1-\eta\gamma)^{n-i}\sum_{\ell=0}^{n}\left[\left(\tfrac{-\eta}{1-\eta\gamma}K^U(X_n,X_n)\right)^{\ell}k(X_n,\boldsymbol{x}_*)\right]_i \\
&= (1-\eta\gamma)^{n-i}\left[\left(I+\tfrac{\eta}{1-\eta\gamma}K^U(X_n,X_n)\right)^{-1}k(X_n,\boldsymbol{x}_*)\right]_i . \qquad (25)
\end{aligned}$$

In the third line, we used that, $\left[\left(K^U(X_n,X_n)\right)^{\ell}k(X_n,\boldsymbol{x}_*)\right]_i$ is zero for $\ell > n-i$. The last line follows because, for an arbitrary square matrix A,

$$(I+A)\sum_{\ell=0}^{n}(-A)^{\ell} = I + (-1)^n A^{n+1}, \qquad (26)$$

and a size n strictly upper triangular matrix K^U satisfies $(K^U)^{n+1}=0$. Because eigenvalues of a triangular matrix are its diagonal components, $I+\frac{\eta}{1-\eta\gamma}K^U(X_n,X_n)$ is invertible (unless $1=\eta\gamma$).

Let us denote a size n diagonal matrix D_n by

$$[D_n]_{ij} = \begin{cases}(1-\eta\gamma)^{n-i} & (i=j)\\ 0 & (i\neq j),\end{cases} \qquad (27)$$

then under $W_1=0$, f_{on} follows

$$\begin{aligned}
f_{on}(\boldsymbol{x}_*;X_n,Y_n) &= \eta\sum_{i=1}^{n}\boldsymbol{y}_i(1-\eta\gamma)^{n-i}\left[\left(I+\tfrac{\eta}{1-\eta\gamma}K^U(X_n,X_n)\right)^{-1}k(X_n,\boldsymbol{x}_*)\right]_i \\
&= \eta Y_n D_n\left(I+\tfrac{\eta}{1-\eta\gamma}K^U(X_n,X_n)\right)^{-1}k(X_n,\boldsymbol{x}_*) \\
&= Y_n D_n\left(\tfrac{1}{\eta}I+\tfrac{1}{1-\eta\gamma}K^U(X_n,X_n)\right)^{-1}k(X_n,\boldsymbol{x}_*). \qquad (28)
\end{aligned}$$

A.3 Offline expression of mini-batch online kernel regression

Our results for sample-by-sample learning in Lemma 3.1 can be extended to a mini-batch online learning setting straightforwardly. Let us denote the mini-batch size as b, and the sample set for i-th iteration by (X_i,Y_i) where $X_i\in\mathbb{R}^{d_x\times b}$ and $Y_i\in\mathbb{R}^{d_y\times b}$. For brevity, we write $\phi(X_i)=\Phi_i$, $X=[X_1,...,X_n]$, and $Y=[Y_1,...,Y_n]$. Under an online gradient learning without an L2 regularizer, the weight update follows:

$$W_{t+1} = W_t - \eta(W_t\Phi_t - Y_t)\Phi_t^T. \qquad (29)$$

By applying it recursively from $W_1=0$, we have

$$W_{t+1} = \sum_{i=1}^{t}\eta Y_i\Phi_i^T\prod_{j=i+1}^{t}\left(I-\eta\Phi_j\Phi_j^T\right), \qquad (30)$$

and thus the predictor $f_b^{(n)}(X_*)$ after n-th updates obeys

$$f_b^{(n)}(X_*) = \sum_{i=1}^{n}\eta Y_i\Phi_i^T\prod_{j=i+1}^{n}\left(I-\eta\Phi_j\Phi_j^T\right)\Phi_*, \qquad (31)$$

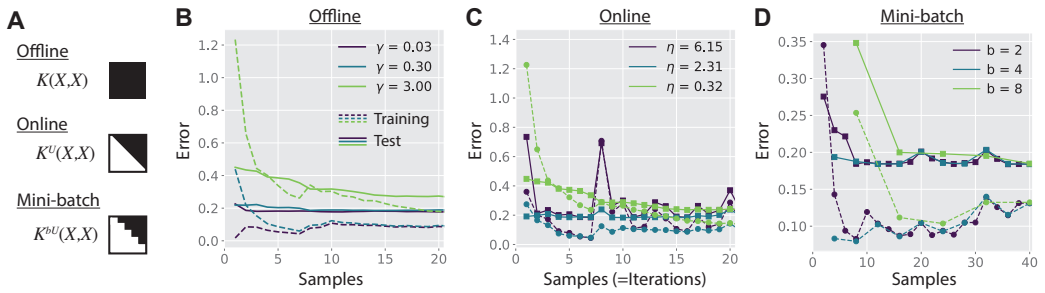


Figure 4: Illustration of learning curves under a random projection kernel $k(x, x') = \phi_J(x)^T \phi_J(x')$ with $\phi_J(x) = \tanh(J\boldsymbol{x})$. **(A)** Visualization of kernels K, K^U, and K^{bU}. **(B)** Offline learning curves under various regularization amplitude γ. **(C)** Online kernel regression learning curves. Points are error under iterative SGD updates $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta(\boldsymbol{w}\phi_J(x) - y)\phi_J(x)^T$, while lines are offline-like expressions from Eq. 6. Learning rate η were chosen as $\eta = 1/(\gamma + k_o)$ where $k_o \equiv E[k(x,x)]$ for $\gamma = \{0.03, 0.3, 3.0\}$ for comparison with panel B. **(D)** Mini-batch learning curves under mini-batch size $b = \{2, 4, 8\}$. Points are mini-batch SGD update while lines are offline-like expressions obtained from Eq. 33. In B-D, each trajectory represents a single learning curve (without any averaging). We used the same random GP task illustrated in Fig. 1, and we generated $\phi_J(x)$ by randomly sampling elements of a 100-dimensional vector J from a Gaussian distribution $\mathcal{N}(0, 1)$.

We introduce a block-wise upper triangular matrix $K^{bU} \in \mathbb{R}^{nb \times nb}$ by

$$\left[K^{bU}(X,X)\right]_{block(i,j)} = \begin{cases} K(X_i, X_j) & (i < j) \\ 0 & (i \geq j), \end{cases} \tag{32}$$

where $\left[K^{bU}\right]_{block(i,j)}$ is a submatrix of K^{bU} consists of $(i-1)b+1$ to ib-th rows and $(j-1)b+1$ to jb-th columns. This is a block-wise extension of K^U introduced in Eq. 23. Fig. 4A contrasts K^{bU} against full kernel K and upper-triangular kernel K^U. Using K^{bU}, $f_b^{(n)}$ is written as follows:

Proposition A.1. *The predictor $f_b^{(n)}(X_*) = W_{n+1}\phi(X_*)$ obtained from n-steps of mini-batch learning with Eq. 29 using data $X = [X_1, ..., X_n]$ and $Y = [Y_1, ..., Y_n]$ follows*

$$f_b^{(n)}(X_*) = Y\left(\frac{1}{\eta}I + K^{bU}(X,X)\right)^{-1} K(X, X_*), \tag{33}$$

where K^{bU} is block-wise upper triangular matrix defined in Eq. 32.

Proof. From a parallel arguments with Lemma 3.1 (under $\gamma = 0$), we have

$$\Phi_i^T \prod_{j=i+1}^{n} \left(I - \eta\Phi_j\Phi_j^T\right)\Phi_* = \left[\left(I + \eta K^{bU}\right)^{-1} K(X, X_*)\right]_{block(i)} \tag{34}$$

Thus, we have

$$\begin{aligned} f_b^{(n)}(X_*) &= \sum_{i=1}^{n} \eta Y_i \Phi_i^T \prod_{j=i+1}^{n} \left(I - \eta\Phi_j\Phi_j^T\right)\Phi_* \\ &= Y\left(\frac{1}{\eta}I + K^{bU}(X,X)\right)^{-1} K(X, X_*). \end{aligned} \tag{35}$$

□

Notably, unlike the sample-by-sample online learning, the upper triangular components in the block diagonal components are also missing under the mini-batch setting. It makes intuitive sense because samples learned together in a mini-batch cannot interact with each other.

A.4 Dynamics of target shift in online kernel regression

From the definition (Eq. 8), the target shift after $(n+1)$-th update follows

$$\begin{aligned} Y^e_{n+1} &= Y_{n+1}\left(\tfrac{1}{\eta}I + K^U_{n+1}\right)^{-1}(\gamma I + K_{n+1}) \\ &= (Y_n \quad \boldsymbol{y}_{n+1})\begin{pmatrix} \frac{1}{\eta}I + K^U_n & k(X_n, \boldsymbol{x}_{n+1}) \\ 0 & 1/\eta \end{pmatrix}^{-1}\begin{pmatrix} \gamma I + K_n & k(X_n, \boldsymbol{x}_{n+1}) \\ k(X_n, \boldsymbol{x}_{n+1})^T & \gamma + k_{n+1} \end{pmatrix}. \end{aligned} \tag{36}$$

For brevity, here we defined $K^U_n \equiv K^U(X_n, X_n)$ and $k_{n+1} \equiv k(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1})$. Using the block matrix inverse property, we have

$$\begin{pmatrix} \frac{1}{\eta}I + K^U_n & k(X_n, \boldsymbol{x}_{n+1}) \\ 0 & 1/\eta \end{pmatrix}^{-1} = \begin{pmatrix} \left(\frac{1}{\eta}I + K^U_n\right)^{-1} & -\eta\left(\frac{1}{\eta}I + K^U_n\right)^{-1}k(X_n, \boldsymbol{x}_{n+1}) \\ 0 & \eta \end{pmatrix}. \tag{37}$$

Thus, Y^e_{n+1} is written as

$$\begin{aligned} Y^e_{n+1} &= \left(Y_n\left(\tfrac{1}{\eta}I + K^U_n\right)^{-1}, \quad -\eta\left[f^{(n)}_{on}(\boldsymbol{x}_{n+1}) - \boldsymbol{y}_{n+1}\right]\right)\begin{pmatrix} \gamma I + K_n & k(X_n, \boldsymbol{x}_{n+1}) \\ k(X_n, \boldsymbol{x}_{n+1})^T & \gamma + k_{n+1} \end{pmatrix} \\ &= \left(Y^e_n - \eta\left[f^{(n)}_{on}(\boldsymbol{x}_{n+1}) - \boldsymbol{y}_{n+1}\right]k(\boldsymbol{x}_{n+1}, X_n), \quad \boldsymbol{y}_{n+1} - (\eta[\gamma + k_{n+1}] - 1)\left[f^{(n)}_{on}(\boldsymbol{x}_{n+1}) - \boldsymbol{y}_{n+1}\right]\right), \end{aligned} \tag{38}$$

where $f^{(n)}_{on}(\boldsymbol{x}_{n+1}) \equiv f_{on}(\boldsymbol{x}_{n+1}; X_n, Y_n) = Y_n(\frac{1}{\eta}I + K^U_n)^{-1}k(X_n, \boldsymbol{x}_{n+1})$ from Eq. 6. By denoting the prediction error of the online predictor by

$$\boldsymbol{e}_{n+1} \equiv f^{(n)}_{on}(\boldsymbol{x}_{n+1}) - \boldsymbol{y}_{n+1}, \tag{39}$$

we obtain Eq. 11 in the main text.

The update for the new sample ($i = n+1$) reveals an interesting limit. As $\eta \to 0$, the effective target $\boldsymbol{y}^e_{n+1}(n+1)$ converges to $f_{\text{on}}(\boldsymbol{x}_{n+1}; X_n, Y_n)$. Intuitively, if the learning rate is negligible, the model does not "accept" the new label $\boldsymbol{y}_{n+1}$; instead, its effective target is simply its own current prediction, resulting in no change to the model parameters. Conversely, the $(n+1)$-th sample experiences no target shift ($\boldsymbol{y}^e_{n+1} = \boldsymbol{y}_{n+1}$) when the learning rate is tuned to $\eta = 1/(\gamma + k_{n+1})$, effectively matching the online and offline diagonal regularization.

A.5 Dynamics of target correction in online kernel regression

With a parallel calculation with Appendix A.4, we can derive the dynamics of target correction Y^c_n. Corrected target in the presence of $(n+1)$ data points X_{n+1}, Y_{n+1} is

$$\begin{aligned} Y^c_{n+1} &= Y_{n+1}(\gamma I + K_{n+1})^{-1}\left(\tfrac{1}{\eta}I + K^U_{n+1}\right) \\ &= (Y_n \quad \boldsymbol{y}_{n+1})\begin{pmatrix} \gamma I + K_n & k(X_n, \boldsymbol{x}_{n+1}) \\ k(\boldsymbol{x}_{n+1}, X_n) & \gamma + k_{n+1} \end{pmatrix}^{-1}\begin{pmatrix} \frac{1}{\eta}I + K^U_n & k(X_n, \boldsymbol{x}_{n+1}) \\ 0 & 1/\eta \end{pmatrix}, \end{aligned} \tag{40}$$

where $K_n = K(X_n, X_n)$ and $k_{n+1} = k(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1})$ as before. Let us introduce scalar variables q and ρ by

$$q \equiv k(\boldsymbol{x}_{n+1}, X_n)(\gamma I + K_n)^{-1}k(X_n, \boldsymbol{x}_{n+1}), \quad \rho \equiv \frac{1}{\gamma + k_{n+1} - q}. \tag{41}$$

Then, the inverse matrix is rewritten as

$$\begin{aligned} &\begin{pmatrix} \gamma I + K_n & k(X_n, \boldsymbol{x}_{n+1}) \\ k(\boldsymbol{x}_{n+1}, X_n) & \gamma + k_{n+1} \end{pmatrix}^{-1} \\ &= \begin{pmatrix} (\gamma I + K_n)^{-1}\left(I + \rho k(X_n, \boldsymbol{x}_{n+1})k(\boldsymbol{x}_{n+1}, X_n)(\gamma I + K_n)^{-1}\right) & -\rho(\gamma I + K_n)^{-1}k(X_n, \boldsymbol{x}_{n+1}) \\ -\rho k(\boldsymbol{x}_{n+1}, X_n)(\gamma I + K_n)^{-1} & \rho \end{pmatrix}. \end{aligned} \tag{42}$$

Therefore, the corrected target follows

$$Y_{n+1}^c = \left(Y_n(\gamma I + K_n)^{-1} + \rho \boldsymbol{e}_{n+1}^{\text{off}} k(\boldsymbol{x}_{n+1}, X_n)(\gamma I + K_n)^{-1}, \quad -\rho \boldsymbol{e}_{n+1}^{\text{off}}\right) \begin{pmatrix} \frac{1}{\eta} I + K_n^U & k(X_n, \boldsymbol{x}_{n+1}) \\ 0 & 1/\eta \end{pmatrix}$$
$$= \left(Y_n^c + \rho \boldsymbol{e}_{n+1}^{\text{off}} k(\boldsymbol{x}_{n+1}, X_n)(\gamma I + K_n)^{-1}(\tfrac{1}{\eta} I + K_n^U), \quad \boldsymbol{y}_{n+1} + [1 + \rho(q - 1/\eta)]\, \boldsymbol{e}_{n+1}^{\text{off}}\right), \tag{43}$$

where $\boldsymbol{e}_{n+1}^{\text{off}}$ is the prediction error of the offline estimator defined by

$$\boldsymbol{e}_{n+1}^{\text{off}} \equiv f_{\text{off}}(\boldsymbol{x}_{n+1}; X_n, Y_n) - \boldsymbol{y}_{n+1}. \tag{44}$$

Denoting

$$c_q \equiv \frac{\gamma + k_{n+1} - 1/\eta}{\gamma + k_{n+1} - q}, \quad C_K \equiv (\gamma I + K_n)^{-1}(\tfrac{1}{\eta} I + K_n^U), \tag{45}$$

we have

$$\boldsymbol{y}_i^c(n+1) = \begin{cases} \boldsymbol{y}_i^c(n) + \rho \boldsymbol{e}_{n+1}^{\text{off}} \left[k(\boldsymbol{x}_{n+1}, X_n) C_K\right]_i & i \le n \\ \boldsymbol{y}_{n+1} + c_q \boldsymbol{e}_{n+1}^{\text{off}}, & i = n+1 \end{cases} \tag{46}$$

Thus, the change in target correction depends on the prediction error of the offline predictor on the new sample $(\boldsymbol{x}_{n+1}, \boldsymbol{y}_{n+1})$, $\boldsymbol{e}_{n+1}^{\text{off}}$. If the offline regressor $f_{\text{off}}^{(n)}$ predicts the next data perfectly, there is no additional target correction, whereas in the presence of non-zero error, previous samples experience correction that depends on its similarity with the new input $k(\boldsymbol{x}_{n+1}, \boldsymbol{x}_i)$.

A.6 Discrepancy minimization formulation of target correction

We consider an optimization problem where we aim to optimize the target for online learning Z such that the learned function from online learning becomes close to the function learned in the offline manner with the true target Y. Because online and offline estimators are written as

$$f_{on}(\boldsymbol{x}; X, Z) = Z \left(\tfrac{1}{\eta} I + K^U(X, X)\right)^{-1} k(X, \boldsymbol{x}), \tag{47a}$$

$$f_{\text{off}}(\boldsymbol{x}; X, Y) = Y \left(\gamma I + K(X, X)\right)^{-1} k(X, \boldsymbol{x}), \tag{47b}$$

minimization of the distance between the two functions is written as:

$$\ell(Z) = \frac{1}{2} \left\| Z(\tfrac{1}{\eta} I + K^U(X, X))^{-1} k(X, \cdot) - Y(\gamma I + K(X, X))^{-1} k(X, \cdot) \right\|_{\mathcal{H}}^2, \tag{48}$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the RKHS norm. Assuming that $K(X, X)$ is invertible, the global optimization of Z with respect to this loss yield

$$Z = Y \left(\gamma I + K(X, X)\right)^{-1} \left(\tfrac{1}{\eta} I + K^U(X, X)\right), \tag{49}$$

which recovers the target correction solution (Eq. 12).

A.7 Proof of Proposition 4.2

Optimization of the loss (Eq. 15) with respect to Z_{new} yields

$$Z_{new} = (G - Z_{past} B)\, C^{-1}, \tag{50}$$

where matrices $B \in \mathbb{R}^{(n-b)\times b}$, $C \in \mathbb{R}^{b \times b}$, and $G \in \mathbb{R}^{d_y \times b}$ are defined as

$$\begin{pmatrix} * & B \\ B^T & C \end{pmatrix} \equiv \left(\tfrac{1}{\eta} I + K_{tot}^U\right)^{-1} (\gamma_o I + K_{tot}) \left(\tfrac{1}{\eta} I + K_{tot}^L\right)^{-1} \tag{51a}$$

$$(* \quad G) \equiv Y(\gamma I + K_{tot})^{-1} K \left(\tfrac{1}{\eta} I + K_{tot}^L\right)^{-1}, \tag{51b}$$

where we introduced $K_{tot} = K(X_{tot}, X_{tot})$, $K_{tot}^U = K^U(X_{tot}, X_{tot})$, and $K_{tot}^L = K^L(X_{tot}, X_{tot})$ for brevity. $K^L = (K^U)^T$ is the strictly lower-triangular matrix generated from K, and $*$ terms in

the matrices are terms we don't need to derive. Using the properties of block matrix inverse, from a straightforward linear algebra (see Appendix A.8), B, C, and G are written as

$$B = \left(\tfrac{1}{\eta}I + K^U_{pp}\right)^{-1} K_{pn}\left(I - (\tfrac{1}{\eta}I + K^U_{nn})^{-1}(\gamma_o I + K_{nn})\right)\left(\tfrac{1}{\eta}I + K^L_{nn}\right)^{-1}, \tag{52a}$$

$$C = \left(\tfrac{1}{\eta}I + K^U_{nn}\right)^{-1}(\gamma_o I + K_{nn})\left(\tfrac{1}{\eta}I + K^L_{nn}\right)^{-1}, \tag{52b}$$

$$G = \left(Y_{new} + \gamma\left[Y_{past}(\gamma I + K_{pp})^{-1}K_{pn} - Y_{new}\right]Q^{-1}\right)(\tfrac{1}{\eta}I + K^L_{nn})^{-1}, \tag{52c}$$

where $K_{pp} = K(X_{past}, X_{past})$, $K_{pn} = K(X_{past}, X_{new})$, and $K_{nn} = K(X_{new}, X_{new})$. Matrix Q in the third line is defined as

$$Q \equiv (\gamma I + K_{nn}) - K_{np}(\gamma I + K_{pp})^{-1}K_{pn}. \tag{53}$$

Let us further denote X_{past} as X_p and X_{new} as X_n for brevity, and use the same notation for Y and Z. Then, substituting matrices B, C, and G in $Z_{new} = (G - Z_p B)C^{-1}$ with the expressions above, we have

$$\begin{aligned}
Z_{new} &= (G - Z_p B)C^{-1} \\
&= \left(Y_n + \gamma[Y_p(\gamma I + K_{pp})^{-1}K_{pn} - Y_n]Q^{-1}\right)(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K^U_{nn}) \\
&\quad - Z_p(\tfrac{1}{\eta}I + K^U_{pp})^{-1}K_{pn}\left(I - (\tfrac{1}{\eta}I + K^U_{nn})^{-1}(\gamma_o I + K_{nn})\right)(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K^U_{nn}) \\
&= \left(Y_n + \gamma\left[f_{\text{off}}(X_n; X_p, Y_p) - Y_n\right]Q^{-1}\right)(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K^U_{nn}) \\
&\quad - f_{on}(X_n; X_p, Z_p)\left[(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K^U_{nn}) - I\right] \\
&= Y_n + \gamma\left[f_{\text{off}}(X_n; X_p, Y_p) - Y_n\right]Q^{-1}(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K^U_{nn}) \\
&\quad + (Y_n - f_{on}(X_n; X_p, Z_p))\left[(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K^U_{nn}) - I\right].
\end{aligned} \tag{54}$$

In the third line, we used Eqs. 2 and 6 to get

$$Y_p(\gamma I + K_{pp})^{-1}K_{pn} = f_{\text{off}}(X_n; X_p, Y_p), \quad Z_p(\tfrac{1}{\eta}I + K^U_{pp})^{-1}K_{pn} = f_{on}(X_n; X_p, Z_p). \tag{55}$$

Let us define C_{off} and C_{on} by

$$C_{on} \equiv \left((\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K^U_{nn}) - I\right), \tag{56a}$$

$$C_{\text{off}} \equiv \gamma\left([\gamma I + K_{nn}] - K_{np}[\gamma I + K_{pp}]^{-1}K_{pn}\right)^{-1}(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta}I + K^U_{nn}). \tag{56b}$$

Then, we obtain the simple expression from the main text:

$$Z_{new} = Y_n + (Y_n - f_{on}(X_n; X_p, Z_p))C_{on} + (f_{\text{off}}(X_n; X_p, Y_p) - Y_n)C_{\text{off}}. \tag{57}$$

A.8 Derivation of matrices B, C, G

Here, we detail the derivation of matrices B, C, and G for completeness. From the block matrix inverse formula, the inverse of the upper triangular matrix $\frac{1}{\eta}I + K^U$ follows

$$\begin{aligned}
\left(\tfrac{1}{\eta}I + K^U\right)^{-1} &= \begin{pmatrix} \frac{1}{\eta}I + K^U_{pp} & K_{pn} \\ 0 & \frac{1}{\eta}I + K^U_{nn} \end{pmatrix}^{-1} \\
&= \begin{pmatrix} (\frac{1}{\eta}I + K^U_{pp})^{-1} & -(\frac{1}{\eta}I + K^U_{pp})^{-1}K_{pn}(\frac{1}{\eta}I + K^U_{nn})^{-1} \\ 0 & (\frac{1}{\eta}I + K^U_{nn})^{-1} \end{pmatrix},
\end{aligned} \tag{58}$$

where $K_{pp} \in \mathbb{R}^{(n-b)\times(n-b)}$ is the kernel matrix of previous $n - b$ samples, $K_{nn} \in \mathbb{R}^{b\times b}$ is the kernel matrix of the new b samples, and $K_{pn} \in \mathbb{R}^{(n-b)\times b}$ is the cross term. Similarly, the inverse of the lower triangular matrix $\frac{1}{\eta}I + K^L$ is decomposed as

$$\begin{aligned}
\left(\tfrac{1}{\eta}I + K^L\right)^{-1} &= \begin{pmatrix} \frac{1}{\eta}I + K^L_{pp} & 0 \\ K_{np} & \frac{1}{\eta}I + K^L_{nn} \end{pmatrix}^{-1} \\
&= \begin{pmatrix} (\frac{1}{\eta}I + K^L_{pp})^{-1} & 0 \\ -(\frac{1}{\eta}I + K^L_{nn})^{-1}K_{np}(\frac{1}{\eta}I + K^L_{pp})^{-1} & (\frac{1}{\eta}I + K^L_{nn})^{-1} \end{pmatrix}.
\end{aligned} \tag{59}$$

Therefore, matrices B and C follow

$$\begin{aligned}\begin{pmatrix} * & B \\ B^T & C \end{pmatrix} &= \left(\tfrac{1}{\eta}I + K^U\right)^{-1} (\gamma_o I + K) \left(\tfrac{1}{\eta}I + K^L\right)^{-1} \\ &= \left(\tfrac{1}{\eta}I + K^U\right)^{-1} \begin{pmatrix} * & K_{pn}\left(\tfrac{1}{\eta}I + K^L_{nn}\right)^{-1} \\ * & (\gamma_o I + K_{nn})\left(\tfrac{1}{\eta}I + K^L_{nn}\right)^{-1} \end{pmatrix} \\ &= \begin{pmatrix} * & \left(\tfrac{1}{\eta}I + K^U_{pp}\right)^{-1} K_{pn}\left(I - (\tfrac{1}{\eta}I + K^U_{nn})^{-1}(\gamma_o I + K_{nn})\right)(\tfrac{1}{\eta}I + K^L_{nn})^{-1} \\ * & (\tfrac{1}{\eta}I + K^U_{nn})^{-1}(\gamma_o I + K_{nn})(\tfrac{1}{\eta}I + K^L_{nn})^{-1} \end{pmatrix},\end{aligned} \tag{60}$$

where $*$ represents terms we omitted, as we do not need to estimate for the iterative correction estimation. Similarly, on G term, denoting

$$Q \equiv (\gamma I + K_{nn}) - K_{np}(\gamma I + K_{pp})^{-1} K_{pn}, \tag{61}$$

$(\gamma I + K)^{-1}$ is rewritten as

$$\begin{aligned}(\gamma I + K)^{-1} &= \begin{pmatrix} \gamma I + K_{pp} & K_{pn} \\ K_{np} & \gamma I + K_{nn} \end{pmatrix}^{-1} \\ &= \begin{pmatrix} (\gamma I + K_{pp})^{-1}\left(I + K_{pn}Q^{-1}K_{np}(\gamma I + K_{pp})^{-1}\right) & -(\gamma I + K_{pp})^{-1}K_{pn}Q^{-1} \\ -Q^{-1}K_{np}(\gamma I + K_{pp})^{-1} & Q^{-1} \end{pmatrix},\end{aligned} \tag{62}$$

and $K(\frac{1}{\eta}I + K^L)^{-1}$ follows

$$\begin{aligned}K\left(\tfrac{1}{\eta}I + K^L\right)^{-1} &= \begin{pmatrix} K_{pp} & K_{pn} \\ K_{np} & K_{nn} \end{pmatrix} \begin{pmatrix} (\tfrac{1}{\eta}I + K^L_{pp})^{-1} & 0 \\ -(\tfrac{1}{\eta}I + K^L_{nn})^{-1}K_{np}(\tfrac{1}{\eta}I + K^L_{pp})^{-1} & (\tfrac{1}{\eta}I + K^L_{nn})^{-1} \end{pmatrix} \\ &= \begin{pmatrix} * & K_{pn}(\tfrac{1}{\eta}I + K^L_{nn})^{-1} \\ * & K_{nn}(\tfrac{1}{\eta}I + K^L_{nn})^{-1} \end{pmatrix}.\end{aligned} \tag{63}$$

Thus, G in $(*, G) = Y(\gamma I + K)^{-1}K(\frac{1}{\eta}I + K^L)$ is rewritten as

$$\begin{aligned}G = &\left[Y_p(\gamma I + K_{pp})^{-1}\left(I + K_{pn}Q^{-1}K_{np}(\gamma I + K_{pp})^{-1}\right) - Y_n Q^{-1}K_{np}(\gamma I + K_{pp})^{-1}\right] K_{pn}(\tfrac{1}{\eta}I + K^L_{nn})^{-1} \\ &+ \left(-Y_p(\gamma I + K_{pp})^{-1}K_{pn} + Y_n\right) Q^{-1}K_{nn}(\tfrac{1}{\eta}I + K^L_{nn})^{-1}.\end{aligned} \tag{64}$$

While the expression of matrix G appears complicated, we can simplify the expression by first rearranging the equations by separating Y_p and Y_n dependent terms:

$$\begin{aligned}G &= Y_p(\gamma I + K_{pp})^{-1}\left(K_{pn} - K_{pn}Q^{-1}\left[K_{nn} - K_{np}(\gamma I + K_{pp})^{-1}K_{pn}\right]\right)(\tfrac{1}{\eta}I + K^L_{nn})^{-1} \\ &\quad + Y_n Q^{-1}\left[K_{nn} - K_{np}(\gamma I + K_{pp})^{-1}K_{pn}\right](\tfrac{1}{\eta}I + K^L_{nn})^{-1} \\ &= Y_p(\gamma I + K_{pp})^{-1}K_{pn}\gamma Q^{-1}(\tfrac{1}{\eta}I + K^L_{nn})^{-1} + Y_n(I - \gamma Q^{-1})(\tfrac{1}{\eta}I + K^L_{nn})^{-1} \\ &= \left(Y_n + \gamma\left[Y_p(\gamma I + K_{pp})^{-1}K_{pn} - Y_n\right]Q^{-1}\right)(\tfrac{1}{\eta}I + K^L_{nn})^{-1}.\end{aligned} \tag{65}$$

In the third line, we used $K_{nn} - K_{np}(\gamma I + K_{pp})^{-1}K_{pn} = Q - \gamma I$.

A.9 Online-offline gap under iterative target correction

Under the iterative target correction, while stability and causality are restored, equivalence between online and offline predictors (Theorem 4.1) no longer holds in general. This motivates us to analyze the progression of the difference between online and offline predictors f_{on} and f_{off} under the iterative algorithm. Below, we first note that the gap has a simple expression. We then show that in the non-i.i.d. online continual learning setting, the gap is bounded by the cumulative error under the iterative offline learning.

As in the main text, we denote a set of previous samples as X_p, Y_p, a set of b samples newly introduced for iterative correction as (X_n, Y_n), and the total samples as (X_{tot}, Y_{tot}) (i.e., $X_{tot} = [X_p, X_n]$ and $Y_{tot} = [Y_p, Y_n]$). The online/offline gap after a one-step iterative correction is then written as follows.

Lemma A.2 (Online/offline gap expression)**.** *Consider a one-step iterative target correction with Prop. 4.2 from (X_p, Y_p, Z_p) using new samples (X_n, Y_n). The online/offline gap under one-step update follows:*

$$f_{on}(\,\cdot\,; X_{tot}, Z_{tot}) - f_{\mathit{off}}(\,\cdot\,; X_{tot}, Y_{tot}) \\ = \left[g_p(\,\cdot\,) - g_p(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,)\right] - E_n Q^{-1}\left[r_n(\cdot) - r_n(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,)\right], \tag{66}$$

where $K_{nn} \equiv k(X_n, X_n)$, Matrices E_n, Q are defined as

$$E_n \equiv Y_n - f_{\mathit{off}}(X_n; X_p, Y_p) \tag{67a}$$

$$Q \equiv (\gamma I + K_{nn}) - K_{np}(\gamma I + K_{pp})^{-1} K_{pn}, \tag{67b}$$

and functions $r_n(\,\cdot\,)$, $g_p(\,\cdot\,)$ in the RKHS are defined as

$$r_n(\,\cdot\,) \equiv k(X_n, \,\cdot\,) - K_{np}(\gamma I + K_{pp})^{-1} k(X_p, \,\cdot\,) \tag{68a}$$

$$g_p(\,\cdot\,) \equiv f_{on}(\,\cdot\,; X_p, Z_p) - f_{\mathit{off}}(\,\cdot\,; X_p, Y_p). \tag{68b}$$

Proof. From Eq. 58, $f_{on}(\,\cdot\,; X_{tot}, Z_{tot})$ is rewritten as

$$f_{on}(\,\cdot\,; X_{tot}, Z_{tot}) = f_{on}(\,\cdot\,; X_p, Z_p) + \left(Z_n - Z_p(\tfrac{1}{\eta} I + K_{pp}^U)^{-1} K_{pn}\right)(\tfrac{1}{\eta} I + K_{nn}^U)^{-1} k(X_n, \,\cdot\,) \\ = f_{on}(\,\cdot\,; X_p, Z_p) + f_{on}\left(\,\cdot\,; Z_n - f_{on}(X_n; X_p, Z_p)\right). \tag{69}$$

Similarly, from Eq. 62, $f_{\text{off}}(\,\cdot\,; X_{tot}, Y_{tot})$ is rewritten as

$$f_{\text{off}}(\,\cdot\,; X_{tot}, Y_{tot}) \\ = f_{\text{off}}(\,\cdot\,; X_p, Y_p) + \left(Y_n - Y_p(\gamma I + K_{pp})^{-1} K_{pn}\right) Q^{-1}(k(X_n, \,\cdot\,) - K_{np}(\gamma I + K_{pp})^{-1} k(X_p, \,\cdot\,)) \\ = f_{\text{off}}(\,\cdot\,; X_p, Y_p) + E_n Q^{-1} r_n(\,\cdot\,), \tag{70}$$

where Q, E_n, and $r_n(\,\cdot\,)$ are defined above. Using the definition of Z_n (in Eqs. 16 - 17),

$$Z_n - f_{on}(X_n; X_p, Z_p) \\ = \left([Y_n - f_{on}(X_n; X_p, Z_p)] + [f_{\text{off}}(X_n; X_p, Y_p) - Y_n]\gamma Q^{-1}\right)(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta} I + K_{nn}^U) \\ = \left(E_n(I - \gamma Q^{-1}) - g_p(X_n)\right)(\gamma_o I + K_{nn})^{-1}(\tfrac{1}{\eta} I + K_{nn}^U), \tag{71}$$

where, in the second line, we used $g_p(X_n) = f_{on}(X_n; X_p, Z_p) - f_{\text{off}}(X_n; X_p, Y_p)$. By replacing $Z_n - f_{on}(X_n; X_p, Z_p)$ in Eq. 69 with the equation above and use $I - \gamma Q^{-1} = Q^{-1}(Q - \gamma I) = Q^{-1} r_n(X_n)$, we have

$$f_{on}(\,\cdot\,; X_{tot}, Z_{tot}) = f_{on}(\,\cdot\,; X_p, Z_p) + \left(E_n Q^{-1} r_n(X_n) - g_p(X_n)\right)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,). \tag{72}$$

Therefore, $f_{on}(\,\cdot\,; X_{tot}, Z_{tot}) - f_{\text{off}}(\,\cdot\,; X_{tot}, Y_{tot})$ is rewritten as

$$f_{on}(\,\cdot\,; X_{tot}, Z_{tot}) - f_{\text{off}}(\,\cdot\,; X_{tot}, Y_{tot}) \\ = \left[g_p(\,\cdot\,) - g_p(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,)\right] - E_n Q^{-1}\left[r_n(\cdot) - r_n(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,)\right] \tag{73}$$

□

A.10 Online-offline gap under iterative target correction in the general non-i.i.d setting (Proposition 4.3)

We first consider the upper bound of the online/offline gap under a general non-i.i.d. setting, which is relevant to online continual learning scenarios with distributional shift over samples. In this setting, the online/offline gap generally doesn't converge to zero under iterative target correction. It underlies the challenge of online continual learning where previously learned samples may not help or even impede subsequent learning. Nevertheless, we show that the online/offline gap under the iterative target correction is bounded by the error of the iteratively-updated offline predictor. This is in contrast with the gap under naive online learning with the true target, where the gap may explode depending on the sample order.

Proposition 4.3 (restated): Consider iterative target correction with data $X_{1:n} = [X_1, ..., X_n]$ and $Y_{1:n} = [Y_1, ..., Y_n]$. Then, under $\gamma > 0$ and $\gamma_o > 0$, the online/offline gap ϵ^g is bounded by:

$$\epsilon_n^g \equiv \|f_{on}(\,\cdot\,; X_{1:n}, Z_{1:n}) - f_{\text{off}}(\,\cdot\,; X_{1:n}, Y_{1:n})\|_{\mathcal{H}} \leq \frac{1}{\sqrt{\gamma}} \sum_{i=1}^{n} \|Y_i - f_{\text{off}}(X_i; X_{1:i-1}, Y_{1:i-1})\|_F \,. \tag{74}$$

Proof. We first consider one-step bound at $i = n$. Let us denote $X_p = [X_1, ..., X_{n-1}]$, $Y_p = [Y_1, ..., Y_{n-1}]$, and $Z_p = [Z_1, ..., Z_{n-1}]$. From the lemma above, the gap is decomposed into two terms. From triangular inequality,

$$\begin{aligned}\epsilon_n^g \leq & \left\| \left[g_p(\,\cdot\,) - g_p(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,) \right] \right\|_{\mathcal{H}} \\ & + \left\| E_n Q^{-1} \left[r_n(\cdot) - r_n(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,) \right] \right\|_{\mathcal{H}} .\end{aligned} \tag{75}$$

The first term that depends on the previous discrepancy g_p satisfies

$$\|g_p(\,\cdot\,) - g_p(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,)\|_{\mathcal{H}} \leq \|g_p\|_{\mathcal{H}}. \tag{76}$$

This is because denoting the sampling operator $\hat{S}_n : \mathcal{H} \to \mathbb{R}^n$ by $\hat{S}_n g = g(X_n)^T$, we have

$$\begin{aligned} g_p(\cdot) - g_p(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \cdot) &= \left(I - \hat{S}_n^T (\gamma_o I + \hat{S}_n \hat{S}_n^T)^{-1} \hat{S}_n \right) g_p(\cdot) \\ &= \gamma_o (\gamma_o I + \hat{S}_n^T \hat{S}_n)^{-1} g_p(\cdot), \end{aligned} \tag{77}$$

and $\gamma_o(\gamma_o I + \hat{S}_n^T \hat{S}_n)^{-1}$ is a shrinking operator. By definition, $\|g_p(\,\cdot\,)\|_{\mathcal{H}}$ follows

$$\|g_p(\,\cdot\,)\|_{\mathcal{H}} = \|f_{on}\left(\,\cdot\,; X_{1:n-1}, Z_{1:n-1}\right) - f_{\text{off}}\left(\,\cdot\,; X_{1:n-1}, Y_{1:n-1}\right)\|_{\mathcal{H}} = \epsilon_{n-1}^g. \tag{78}$$

This means that, the discrepancy originated from previous samples do not increase under the iterative target correction.

However, the second term that scales with the current offline predictor error E_n potentially increases the gap. From a parallel argument with Eq. 77, this term is rewritten as

$$\left\| E_n Q^{-1} \left[r_n(\,\cdot\,) - r_n(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,) \right] \right\|_{\mathcal{H}}^2 \leq \left\| E_n Q^{-1} r_n(\,\cdot\,) \right\|_{\mathcal{H}}^2 \tag{79}$$

From the definition of $r_n(\,\cdot\,)$, we have

$$\begin{aligned} & \left\| E_n Q^{-1} r_n(\,\cdot\,) \right\|_{\mathcal{H}}^2 \\ & \leq \text{tr}\left[E_n Q^{-1} \left(K_{nn} - K_{np}(\gamma I + K_{pp})^{-1} K_{pn} - \gamma K_{np}(\gamma I + K_{pp})^{-2} K_{pn} \right) Q^{-1} E_n^T \right] \\ & \leq \text{tr}\left[E_n Q^{-1} \left(Q - \gamma K_{np}(\gamma I + K_{pp})^{-2} K_{pn} - \gamma I \right) Q^{-1} E_n^T \right] \\ & \leq \text{tr}\left[E_n Q^{-1} E_n^T \right] \leq \left\| Q^{-1} \right\|_2 \left\| E_n \right\|_F^2 \end{aligned} \tag{80}$$

In the third line, we used $Q = \gamma I + \left[K_{nn} - K_{np}(\gamma I + K_{pp})^{-1} K_{pn} \right]$, and in the last line, we used $Q - \gamma K_{np}(\gamma I + K_{pp})^{-2} K_{pn} - \gamma I \preceq Q$. The norm $\left\| Q^{-1} \right\|_2$ is upper-bounded by

$$\left\| Q^{-1} \right\|_2 = \left\| \left(\gamma I + \left[K_{nn} - K_{np}(\gamma I + K_{pp})^{-1} K_{pn} \right] \right)^{-1} \right\|_2 \leq \frac{1}{\gamma}. \tag{81}$$

The last inequality holds because $K_{nn} - K_{np}(\gamma I + K_{pp})^{-1} K_{pn}$ is positive semi-definite. Therefore, we have

$$\left\| E_n Q^{-1} \left[r_n(\,\cdot\,) - r_n(X_n)(\gamma_o I + K_{nn})^{-1} k(X_n, \,\cdot\,) \right] \right\|_{\mathcal{H}} \leq \frac{1}{\sqrt{\gamma}} \left\| E_n \right\|_F . \tag{82}$$

Combining the upper bounds on the two terms and apply them recursively, we get

$$\begin{aligned} \|f_{on}(\,\cdot\,; X_{1:n}, Z_{1:n}) - f_{\text{off}}(\,\cdot\,; X_{1:n}, Y_{1:n})\|_{\mathcal{H}} & \leq \frac{1}{\sqrt{\gamma}} \sum_{i=1}^{n} \left\| E_i \right\|_F \\ & \leq \frac{1}{\sqrt{\gamma}} \sum_{i=1}^{n} \left\| Y_i - f_{\text{off}}\left(X_i; X_{1:i-1}, Y_{1:i-1}\right) \right\|_F . \end{aligned} \tag{83}$$

□

Thus, the online-offline discrepancy under iterative target correction is upper-bounded by the error of iterative offline learning, $\|Y_i - f_{\text{off}}(X_i; X_{1:i-1}, Y_{1:i-1})\|_F$. In general, this bound does not converge to zero in the $n \to \infty$ limit under arbitrary sample orderings. This is an inevitable feature of the non-i.i.d. setting, in which new samples may be adversarial to old ones, as exemplified by domain- and class-incremental continual learning.

By contrast, under vanilla online learning, from Eq. 8, the online-offline gap satisfies

$$\|f_{\text{on}}(\,\cdot\,; X, Y) - f_{\text{off}}(\,\cdot\,; X, Y)\|_{\mathcal{H}} = \left\|(Y^e - Y)(\gamma I + K(X,X))^{-1} k(X, \cdot)\right\|_{\mathcal{H}} \leq \frac{1}{2\sqrt{\gamma}} \|Y^e - Y\|_F \,. \tag{84}$$

Furthermore, we have

$$\begin{aligned} \|Y^e - Y\|_F &\leq \left\| Y(\tfrac{1}{\eta} I + K^U(X,X))^{-1}(\gamma I + K(X,X)) \right\|_F + \|Y\|_F \\ &\leq \|Y\|_F \left(\frac{\sigma_{\max}(K(X,X)) + \gamma}{\sigma_{\min}(\frac{1}{\eta} I + K^U(X,X))} + 1 \right), \end{aligned} \tag{85}$$

where $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$ denote the largest and smallest singular values of a matrix M, respectively. Notably, because $\frac{1}{\eta} I + K^U_{nn}$ is non-normal, its smallest singular value is not bounded below by $1/\eta$. Instead, it can decay exponentially with the number of samples, which underlies the instability of vanilla online learning. In contrast, under iterative target correction, the upper bound no longer scales with $\sigma^{-1}_{\min}(\frac{1}{\eta} I + K^U(X,X))$, supporting robust learning dynamics.

A.11 Online-offline gap under iterative target correction in an i.i.d setting

We next consider the i.i.d. case to further examine the properties of iterative target correction. Although our framework primarily concerns online continual learning, where samples are drawn in a non-i.i.d. manner, convergence in the i.i.d. setting provides additional support for the validity of the iterative target correction algorithm.

Assumptions A.11 In addition to i.i.d. sampling assumption, in this subsection, but only in this subsection, we introduce the following set of assumptions. We set $d_y = 1$ and consider a finite dimensional feature $\phi : \mathcal{X} \to \mathbb{R}^{d_h}$ with $d_h < \infty$ for kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$. We further assume that the features are normalized (i.e., $\|\phi(\boldsymbol{x})\|_2 = 1$ for all $\boldsymbol{x} \in \mathcal{X}$) and for the probability distribution $p(x)$ from which we pick samples, $E_{\boldsymbol{x} \sim p(\boldsymbol{x})}[\phi(\boldsymbol{x})\phi(\boldsymbol{x})^T] = \Sigma \succeq \rho I_h$ with $\rho > 0$. Finally, target y_i at trial i are generated by $y_i = \boldsymbol{w}_*^T \phi(\boldsymbol{x})$.

Before deriving the upper bound of the offline-online gap, we note that the error of the offline kernel regressions, $E_t = Y_t - f_{\text{off}}(X_t; X_{1:t-1}, Y_{1:t-1})$, converges at $\mathcal{O}(1/t)$ rate in the i.i.d. setting. Note that, under $d_y = 1$, E_t is a b-dimensional vector.

Lemma A.3. *Under the Assumptions A.11 stated above, the following upper bound holds for the error under the offline kernel regression:*

$$\mathbb{E}_{X_{1:t+1}} \left[||E_{t+1}||^2 \right] \leq b \, \|\boldsymbol{w}_*\|^2 \, \frac{(1+\gamma)^2}{\gamma(bt+1)}. \tag{86}$$

Proof. The proof follows from the leave-one-out bounds in Zhang (2003) which we describe for completeness. Let us denote

$$B \equiv \gamma I + \sum_{j=1}^{bt+1} \phi(\boldsymbol{x}_j)\phi(\boldsymbol{x}_j)^T, \quad A_{-i} \equiv \gamma I + \sum_{j=1, j\neq i}^{bt+1} \phi(\boldsymbol{x}_j)\phi(\boldsymbol{x}_j)^T = B - \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T, \tag{87}$$

and $\phi_i \equiv \phi(\boldsymbol{x}_o)$ for brevity. If we exchange the i-th sample with the $bt+1$-th sample, the error is written as

$$\epsilon_i \equiv \left(\boldsymbol{w}_*^T \phi(\boldsymbol{x}_i) - f_{\text{off}}(\boldsymbol{x}_i; X_{-i}, Y_{-i}) \right)^2 = \left(\gamma \boldsymbol{w}_*^T A_{-i}^{-1} \phi(\boldsymbol{x}_i) \right)^2, \tag{88}$$

where (X_{-i}, Y_{-i}) are the sets of training samples except the i-th sample. Using ϵ_i, from exchangeability of i.i.d. samples, the error is rewritten as

$$\mathbb{E}_{X_{1:t},\boldsymbol{x}_*}\left[(y_* - f_{\text{off}}(\boldsymbol{x}_*; X_{1:t}, Y_{1:t}))^2\right] = \frac{1}{bt+1}\mathbb{E}_{X_{1:t},\boldsymbol{x}_{bt+1}}\left[\sum_{i=1}^{bt+1}\epsilon_i\right] \tag{89}$$

From $A_{-i}^{-1} = (B - \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T)^{-1} = B^{-1} + \frac{B^{-1}\phi_i\phi_i^T B^{-1}}{1-\phi_i^T B^{-1}\phi_i}$, we have

$$\epsilon_i = \left(\gamma \boldsymbol{w}_*^T A_{-i}^{-1}\phi(\boldsymbol{x}_i)\right)^2 = \frac{\gamma^2(\boldsymbol{w}_*^T B^{-1}\phi_i)^2}{\left(1-\phi_i^T B^{-1}\phi_i\right)^2}, \tag{90}$$

and because $B \succeq \gamma I + \phi_i\phi_i^T$,

$$\phi_i^T B^{-1}\phi_i \leq \phi_i^T\left(\gamma I + \phi_i\phi_i^T\right)^{-1}\phi_i \leq \frac{1}{1+\gamma}. \tag{91}$$

Therefore,

$$\epsilon_i = \frac{\gamma^2(\boldsymbol{w}_*^T B^{-1}\phi_i)^2}{(1-\phi_i^T B^{-1}\phi_i)^2} \leq (1+\gamma)^2\left(\boldsymbol{w}_*^T B^{-1}\phi_i\right)^2. \tag{92}$$

Lastly, taking summation over i

$$\begin{aligned}\sum_{i=1}^{bt+1}\epsilon_i &\leq (1+\gamma)^2\sum_{i=1}^{bt+1}\left(\boldsymbol{w}_*^T B^{-1}\phi_i\right)^2 = (1+\gamma)^2\boldsymbol{w}_*^T B^{-1}\left(\sum_{i=1}^{bt+1}\phi_i\phi_i^T\right)B^{-1}\boldsymbol{w}_* \\ &= (1+\gamma)^2\left(\boldsymbol{w}_*^T B^{-1}\boldsymbol{w}_* - \gamma\left\|B^{-1}\boldsymbol{w}_*\right\|^2\right) \leq (1+\gamma)^2\boldsymbol{w}_*^T B^{-1}\boldsymbol{w}_* \leq \frac{(1+\gamma)^2}{\gamma}\left\|\boldsymbol{w}_*\right\|^2.\end{aligned} \tag{93}$$

Substituting ϵ_i in Eq. 89 in the expression above, we obtain

$$\mathbb{E}_{X_{1:t},\boldsymbol{x}_*}\left[(y_* - f_{\text{off}}(\boldsymbol{x}_*; X_{1:t}, Y_{1:t}))^2\right] \leq \frac{(1+\gamma)^2\left\|\boldsymbol{w}_*\right\|^2}{\gamma(bt+1)}. \tag{94}$$

Because $E_t = Y_t - f_{\text{off}}(X_t; X_{1:t-1}, Y_{1:t-1})$ is defined batch-wise (with batch size b), by multiplying the bound above with b, we obtain Eq. 86. □

We also introduce the following Lemma on the expected step-wise shrinkage of the gap:

Lemma A.4. *Let us denote $R^{(b)}$ by*

$$R^{(b)} \equiv \gamma_o\left(\gamma_o I_h + \sum_{i=1}^{b}\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T\right)^{-1}, \tag{95}$$

where $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_b]$ are i.i.d. samples from $p(\boldsymbol{x})$ and I_h is the size d_h identity matrix. Then, denoting the largest eigenvalue of matrix M by $\lambda_{\max}[M]$, we have

$$\lambda_{\max}(\mathbb{E}_X[(R^{(b)})^2]) \leq 1 - \frac{\rho}{1+\gamma_o} \tag{96}$$

Proof. Because $R^{(b)}$ satisfies $0 \preceq R^{(b)} \preceq I$, $(R^{(b)})^2 \preceq R^{(b)}$. Additionally, because $\sum_{i=1}^{b}\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T$ is positive semi-definite, we have

$$R^{(b)} = \gamma_o\left(\gamma_o I_h + \sum_{i=1}^{b}\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T\right)^{-1} \preceq \gamma_o\left(\gamma_o I_h + \phi(\boldsymbol{x}_1)\phi(\boldsymbol{x}_1)^T\right)^{-1} = R^{(1)}, \tag{97}$$

meaning that the operator norm of $R^{(b)}$ under arbitrary batch size $b \geq 1$ is upper bounded by the norm of $R^{(1)}$, which corresponds to the single sample update. Under $b = 1$, from Sherman-Morrison formula,

$$\mathbb{E}_{x_1}\left[\gamma_o\left(\gamma_o I_h + \phi(\boldsymbol{x}_1)\phi(\boldsymbol{x}_1)^T\right)^{-1}\right] = \mathbb{E}_{x_1}\left[I_h - \frac{\phi(\boldsymbol{x}_1)\phi(\boldsymbol{x}_1)^T}{\gamma_o + \left\|\phi(\boldsymbol{x}_1)\right\|^2}\right] = I_h - \frac{\Sigma}{\gamma_o+1}. \tag{98}$$

In the last line we used the assumptions, $\mathbb{E}_x[\phi(\boldsymbol{x})\phi(\boldsymbol{x})^T] = \Sigma$ and $\|\phi(\boldsymbol{x})\|^2 = 1$. Thus, for any $b \geq 1$, the largest eigenvalue $\lambda_{\max}\left(E_X[(R^{(b)})^2]\right)$ is bounded by

$$\lambda_{\max}\left(E_X[(R^{(b)})^2]\right) \leq \lambda_{\max}\left(E_X[R^{(b)}]\right) \leq \lambda_{\max}\left(E_{\boldsymbol{x}_1}[R^{(1)}]\right) \leq \lambda_{\max}\left(I_h - \frac{\Sigma}{\gamma_o + 1}\right) \leq 1 - \frac{\rho}{1+\gamma_o}. \tag{99}$$

□

Proposition A.5. *Under the Assumptions A.11, the expectation of the norm of offline-online gap over i.i.d. samples satisfies*

$$\mathbb{E}_{X_{1:t}}\left[\|g_t(\cdot)\|_{\mathcal{H}}^2\right] \leq \frac{8b(1+\gamma)^2(1+\gamma_o)^2\|\boldsymbol{w}_*\|^2}{\gamma^2\rho^2}\left(\frac{2}{bt} + \left(1 - \frac{\rho}{1+\gamma_o}\right)^{t/2}\right). \tag{100}$$

Proof. The offline-online gap $g_t(\cdot)$ follows

$$\begin{aligned} g_t(\cdot) &= f_{on}(\cdot; X_t, Z_t) - f_{\text{off}}(\cdot; X_t, Y_t) \\ &= \left[g_{t-1}(\cdot) - g_{t-1}(X_t)(\gamma_o I + K_{tt})^{-1}k(X_t, \cdot)\right] - E_t Q_t^{-1}\left[r_t(\cdot) - r_t(X_t)(\gamma_o I + K_{tt})^{-1}k(X_t, \cdot)\right], \end{aligned} \tag{101}$$

where $K_{tt} = k(X_t, X_t)$ is a $b \times b$ gram matrix. The first term is rewritten as

$$\begin{aligned} \left[g_{t-1}(\cdot) - g_{t-1}(X_t)(\gamma_o I + K_{tt})^{-1}k(X_t, \cdot)\right] &= \left[I_h - \phi(X_t)\left(\gamma_o I_b + \phi(X_t)^T\phi(X_t)\right)^{-1}\phi(X_t)^T\right]\boldsymbol{g}_{t-1} \\ &= \left[I_h - \left(\gamma_o I_b + \phi(X_t)\phi(X_t)^T\right)^{-1}\phi(X_t)\phi(X_t)^T\right]\boldsymbol{g}_{t-1} \\ &= \gamma_o\left(\gamma_o I_h + \phi(X_t)\phi(X_t)^T\right)^{-1}\boldsymbol{g}_{t-1}, \end{aligned} \tag{102}$$

where vector $\boldsymbol{g}_{t-1}$ was defined by $g_{t-1}(\boldsymbol{x}) = \boldsymbol{g}_{t-1}^T\phi(\boldsymbol{x})$. We used push-through identity in the second line. For brevity, let us denote the operator term in the equation above and the second term of the online-offline gap equation by

$$R_t \equiv \gamma_o\left(\gamma_o I_h + \phi(X_t)\phi(X_t)^T\right)^{-1}, \tag{103a}$$

$$h_t(\cdot) \equiv -E_t Q_t^{-1}\left[r_t(\cdot) - r_t(X_t)(\gamma_o I + K_{tt})^{-1}k(X_t, \cdot)\right], \tag{103b}$$

Here, R_t is $R^{(b)}$ defined with t-th sample set X_t (we omitted the superscript for brevity). Then, applying it recursively, we have

$$\begin{aligned} \boldsymbol{g}_t &= R_t\boldsymbol{g}_{t-1} + \boldsymbol{h}_t \\ &= R_t R_{t-1}\boldsymbol{g}_{t-2} + R_t\boldsymbol{h}_{t-1} + \boldsymbol{h}_t = \ldots = \sum_{s=0}^{t-1}\left(\prod_{m=0}^{s-1} R_{t-m}\right)\boldsymbol{h}_{t-s}. \end{aligned} \tag{104}$$

Here, we defined the matrix product in a descending order: $\prod_{m=0}^{s-1} R_{t-m} \equiv R_t R_{t-1}..R_{t-(s-1)}$. Therefore,

$$\begin{aligned} \left(\mathbb{E}_{X_{1:t}}\left[\|g_t(\cdot)\|_{\mathcal{H}}^2\right]\right)^{1/2} &= \left(\mathbb{E}_{X_{1:t}}\left[\left\|\sum_{s=0}^{t-1}\left(\prod_{m=0}^{s-1} R_{t-m}\right)h_{t-s}(\cdot)\right\|_{\mathcal{H}}^2\right]\right)^{1/2} \\ &\leq \sum_{s=0}^{t-1}\left(\mathbb{E}_{X_{1:t}}\left[\left\|\left(\prod_{m=0}^{s-1} R_{t-m}\right)h_{t-s}(\cdot)\right\|_{\mathcal{H}}^2\right]\right)^{1/2}. \end{aligned} \tag{105}$$

The term inside the summation is further written as

$$\begin{aligned} &\mathbb{E}_{X_{1:t}}\left[\left\|\left(\prod_{m=0}^{s-1} R_{t-m}\right)\boldsymbol{h}_{t-s}\right\|_{\mathcal{H}}^2\right] \\ &= \mathbb{E}_{X_{1:t-s}}\left[\boldsymbol{h}_{t-s}^T\mathbb{E}_{x_{t-s+1}}\left[R_{t-s+1}^T...R_{t-1}^T\mathbb{E}_{x_t}[R_t^T R_t]R_{t-1}...R_{t-s+1}\right]\boldsymbol{h}_{t-s}\right] \\ &\leq \lambda_{\max}\left(\mathbb{E}_{x_t}[R_t^T R_t]\right)\mathbb{E}_{X_{1:t-s}}\left[\boldsymbol{h}_{t-s}^T\mathbb{E}_{x_{t-s+1}}\left[R_{t-s+1}^T...\mathbb{E}_{x_{t-1}}[R_{t-1}^T R_{t-1}]...R_{t-s+1}\right]\boldsymbol{h}_{t-s}\right] \\ &\leq \left(\prod_{m=0}^{s-1}\lambda_{\max}\left(\mathbb{E}_{x_{t-m}}[R_{t-m}^2]\right)\right)\mathbb{E}_{X_{1:t-s}}\left[\|h_{t-s}(\cdot)\|_{\mathcal{H}}^2\right]. \end{aligned} \tag{106}$$

The second line holds because R_t only depends on the instantaneous sample x_t and h_{t-s} is a function of all samples up to $t-s$.

The former term, $\lambda_{\max}\left(\mathbb{E}_{x_{t-m}}[R_{t-m}^2]\right)$, is upper bounded by Lemma A.4, while regarding the latter term, from Eq. 80 in Appendix A.10, $\|h_{t-s}(\cdot)\|_{\mathcal{H}}^2 \leq \frac{1}{\gamma}\|E_{t-s}\|^2$, thus, we have

$$\mathbb{E}_{x_{1:t-s}}\left[\|h_{t-s}(\cdot)\|_{\mathcal{H}}^2\right] \leq \frac{1}{\gamma}\mathbb{E}_{X_{1:t-s}}\left[\|E_{t-s}\|^2\right], \tag{107}$$

which can be upper bounded by Lemma A.3. Therefore, we have

$$\mathbb{E}_{X_{1:t}}\left[\left\|\left(\prod_{m=0}^{s-1} R_{t-m}\right) h_{t-s}(\cdot)\right\|_{\mathcal{H}}^2\right] \leq \left(1-\frac{\rho}{1+\gamma_o}\right)^s \frac{b(1+\gamma)^2\|\boldsymbol{w}_*\|^2}{\gamma^2(b(t-s-1)+1)}, \tag{108}$$

and thus

$$\left(\mathbb{E}_{X_{1:t}}\left[\|g_t(\cdot)\|_{\mathcal{H}}^2\right]\right)^{1/2} \leq \sum_{s=0}^{t-1}\left(1-\frac{\rho}{1+\gamma_o}\right)^{s/2} \frac{\sqrt{b}(1+\gamma)\|\boldsymbol{w}_*\|}{\gamma\sqrt{b(t-s-1)+1}} \tag{109}$$

Let us introduce $\alpha \equiv \rho/(1+\gamma_o)$ for brevity, and split the summation into two parts by $s < t/2$ and $s \geq t/2$. Excluding the constant term, the first half follows

$$\sum_{s=0}^{t/2-1}\frac{(1-\alpha)^{s/2}}{\sqrt{b(t-s-1)+1}} \leq \frac{1}{\sqrt{bt/2+1}}\sum_{s=0}^{t/2-1}(1-\alpha)^{s/2} = \frac{1}{\sqrt{bt/2+1}}\frac{1-(1-\alpha)^{t/4}}{1-\sqrt{1-\alpha}} \leq \sqrt{\frac{2}{bt}}\frac{2}{\alpha}. \tag{110}$$

In the last inequality, we used $1-\sqrt{1-\alpha} \geq \alpha/2$. Similarly, the latter half follows

$$\sum_{s=t/2}^{t-1}\frac{(1-\alpha)^{s/2}}{\sqrt{b(t-s-1)+1}} \leq (1-\alpha)^{t/4}\sum_{s=0}^{t/2-1}(1-\alpha)^{s/2} = (1-\alpha)^{t/4}\frac{1-(1-\alpha)^{t/4}}{1-\sqrt{1-\alpha}} \leq \frac{2}{\alpha}(1-\alpha)^{t/4}. \tag{111}$$

By combining the two terms,

$$\left(\mathbb{E}_{X_{1:t}}\left[\|g_t(\cdot)\|_{\mathcal{H}}^2\right]\right)^{1/2} \leq \frac{2\sqrt{b}(1+\gamma)\|\boldsymbol{w}_*\|}{\gamma\alpha}\left(\sqrt{\frac{2}{bt}}+(1-\alpha)^{t/4}\right). \tag{112}$$

Using $(a+b)^2 \leq 2(a^2+b^2)$, we have

$$\mathbb{E}_{X_{1:t}}\left[\|g_t(\cdot)\|_{\mathcal{H}}^2\right] \leq \frac{8b(1+\gamma)^2\|\boldsymbol{w}_*\|^2}{\gamma^2\alpha^2}\left(\frac{2}{bt}+(1-\alpha)^{t/2}\right). \tag{113}$$

Inserting $\alpha = \rho/(1+\gamma_o)$, we obtain the proposition. □

Because $0 < \alpha < 1$, at a large t limit, the error converges with $\mathcal{O}(\frac{1}{t})$ rate. $b > 1$ does not accelerate the convergence here because the shrinkage factor under arbitrary b is bounded by $b = 1$ case in Lemma A.4.

B Additional numerical results

B.1 Application to a Standard Image Recognition Task

Here we ask whether the same benefit we observed in the continual learning scenario appears in a standard randomly ordered CIFAR-10 image-recognition task. In contrast to the continual-learning settings, corrected targets did not clearly improve single-pass mini-batch SGD. However, an advantage emerged under multi-epoch training (Fig. 6B), suggesting that target correction is most beneficial in the presence of structured distributional shifts.

B.2 Comparison with Soft-Labeling

Soft-labeling (Szegedy et al., 2016; Müller et al., 2019), where a smoothed label is used instead of the one-hot target label, can be interpreted as a form of target correction. To dissociate the potential impact of this soft-label effect in the iterative target correction algorithm, we also implemented SGD with soft labeling on the two datasets we tested. As depicted in Table 1, the performance gain from soft labeling over SGD was around 1% in CIFAR-10, and slightly negative in CORe50, indicating that soft labeling doesn't explain the gain observed by the target correction in non-linear networks.

Table 1: Final test accuracy (%, mean ± SEM over 10 seeds on CIFAR-10 and 5 seeds on CORe50). Memory-based methods use $M = 0.1N_{\text{train}}$ (CIFAR-10: $M = 3000$; CORe50: $M = 600$) and replay batch $m = 4$. In TC and DER+TC, NTK was updated at the beginning of each task.

METHOD	CIFAR-10 / DOMAIN-IL	CORe50 / CLASS-IL
Vanilla SGD	57.19 ± 2.61	13.59 ± 1.27
Soft Label (SL)	58.05 ± 2.45	11.23 ± 0.74
ER	69.49 ± 1.36	28.29 ± 1.40
DER	66.64 ± 1.32	12.04 ± 0.60
DER++	$\mathbf{69.90 \pm 1.45}$	27.59 ± 1.61
TC (with 10% samples)	64.57 ± 1.92	22.54 ± 1.09
DER+TC	67.64 ± 1.44	$\mathbf{32.04 \pm 1.21}$

B.3 Dark Experience Replay with Target Correction

In Dark Experience Replay (DER), previous network outputs, rather than the ground-truth targets, are replayed (Buzzega et al., 2020), suggesting that non-ground-truth targets can be beneficial. This motivated us to combine iterative target correction (TC) with DER and ask whether corrected targets provide better replay targets than earlier network outputs.

We implemented this variation by applying target correction to each sample newly added to the memory buffer for DER replay, using the samples already stored in the buffer. We then used the corrected target z^c as the replay target instead of the prediction $\hat{y}$ obtained when the sample was added to the memory buffer (see Appendix C.3).

Table 1 compares the performance of this DER + TC method with vanilla DER and several other baselines. DER + TC improves prediction accuracy over DER on both the domain-incremental CIFAR-10 task and the class-incremental CORe50 task. On the latter dataset, it also outperforms DER++ (Boschini et al., 2022). These results suggest that the target correction framework is potentially useful for modifying targets in replay buffers, not just for new samples in online learning.

C Implementation details

Below we provide numerical experiment details. Please also see the corresponding repository `https://anonymous.4open.science/r/target_correction-A829` for further details.

C.1 Toy model (Figs. 1 and 4)

We generated the one-dimensional data, used for illustration of effective target shift and its correction, using a Gaussian Process (GP) with radial basis function (RBF) kernel. Specifically, we defined X_{grid} and $X_{grid} = [0.0, 0.005, ..., 1.0]$ and sampled the true curve $f_*(x)$ by $f_*(X_{grid}) \sim \mathcal{N}(0, k_{\text{rbf}}(X_{grid}, X_{grid}))$, where $k_{\text{rbf}}(x, x') \equiv \exp(-(x - x')^2/\sigma^2_{\text{rbf}})$ with $\sigma^2_{\text{rbf}} = 0.1$. We then sub-sampled mutually exclusive training and test sets X_{train} and X_{test} randomly from X_{grid}, with $|X_{\text{train}}| = 40$ and $|X_{\text{test}}| = 160$. The targets Y_{train} and Y_{test} were generated by adding Gaussian noise to $f_*(x)$. For $x_i \in X_{\text{train}}$, the corresponding target is $y_i = f_*(x) + \sigma_{tn}\xi$, where $\sigma_{tn} = 0.3$ and ξ is a random variable sampled independently from a Gaussian distribution $\mathcal{N}(0, 1)$.

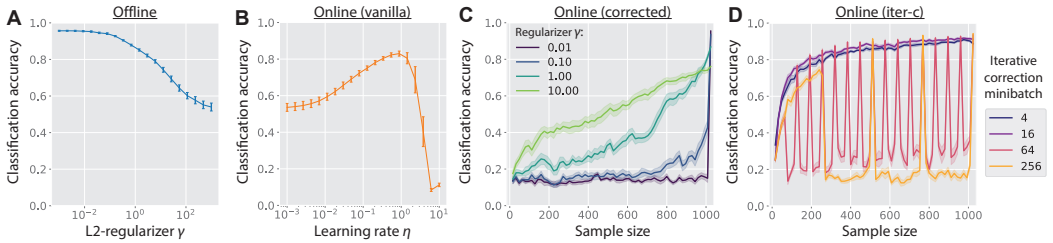


Figure 5: Offline and online regression using NTK applied to MNIST. **A)** Offline learning performance as a function of the L2-regularizer γ. **B)** Vanilla online learning performance as a function of the fixed learning rate η. **C)** Online learning with corrected targets under various regularizers γ of the target offline predictor. Note that the online learning process itself was implemented without a regularizer as noted in Theorem 4.1. **D)** Learning curves of iterative target correction with various minibatch sizes for the iterative correction (Eq. 16) under $\gamma = 0.01$ and $\eta = 1.0$. In all the curves, weight updates were implemented in a fully online, sample-by-sample manner. In panels C and D, the test accuracy was measured after every 16 weight updates. Error bars and shaded areas represent SEMs over 10 random seeds.

In Fig. 1, we used the same RBF kernel with the generative model for offline and online learning model, meaning that the kernel is defined by $k(x, x') = \exp(-(x-x')^2/\sigma_{\text{rbf}}^2)$. We set the regularizer amplitude of the offline learning γ to $\gamma = 1.0$, and the learning rate of online learning η to $\eta = 0.5$. We picked the latter, η, to make the diagonal terms of $\gamma I + K(X, X)$ and $\frac{1}{\eta} I + K^U(X, X)$ the same (i.e., $\gamma + 1 = 1/\eta$).

In Fig. 4, the data was generated in the same way as Fig. 1, but we instead used a finite-size kernel defined by $k(x, x') = \tanh(Jx)^T \tanh(Jx')$ to illustrate the exact match between iterative online and mini-batch updates, and their offline formulations (Eqs. 6 and 33, respectively). We set J to be a 100-dimensional vector, where each element was sampled independently from a Gaussian distribution $\mathcal{N}(0, 1)$.

C.2 Neural Tangent Kernel (NTK) Regression Models (Figs. 2 and 5)

For NTK regression, we used a NTK of an infinite-width convolutional neural network with two convolutional layers and one dense layer. We set the size of receptive field to (3,3) and used average pooling with (2,2) window and (2,2) strides for both convolutional layer, and set the activation function to ReLU. We estimated NTK analytically using Python `neural_tangents` library (Novak et al., 2020). The MNIST images were whitened at each pixel over samples, and presented to the predictor. We sub-sampled 1024 training data and 256 test data to focus on sparse sample regime. In the class-incremental task setting depicted in Fig. 2D, we ordered 1024 training data from label 0 to label 9.

In Figs. 2A-C, we set $\gamma = 0.01$ and $\eta = 1.0$ based on the parameter search in Figs. 5A and B. We set γ_o from Eq. 15 to $\gamma_o = 0$ throughout the numerical experiment because K_{nn} was full rank. Similarly, from parameter search, in Fig. 2D, we used $\gamma = 0.001$ for the offline learning (blue line) and $\eta = 0.001$ for the vanilla online learning (orange), and used $(\gamma, \eta) = (0.001, 0.001)$ for the online learning with corrected targets (green). We instead used $(\gamma, \eta) = (1.0, 0.3)$ for the iterative target correction. In the iterative target correction lines in Fig. 2C and D, we set the mini-batch size for the iterative target correction to $b = 16$, but weight updates were implemented in a fully online, sample-by-sample manner. A large regularizer stabilizes the learning curve of the online learning with corrected targets (Eq. 12), but lowers the final accuracy. Similarly, as depicted in 5D, when the minibatch for iterative correction is large, the final accuracy improves, but the learning curve becomes less stable.

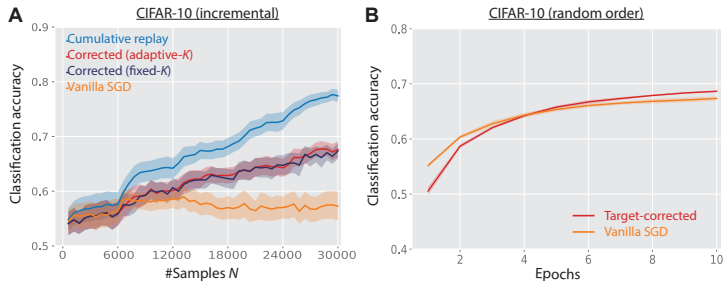


Figure 6: Learning curves under alternative implementations. **(A)** Mini-batch SGD with target correction estimated using a fixed initial NTK (purple line) achieves performance comparable to target correction using an adaptive NTK updated at the beginning of each task (i.e., every 6000 samples; red line) in the domain-incremental CIFAR-10 task. **(B)** Comparison of the learning curves for target correction and vanilla SGD on the standard CIFAR-10 task with randomly ordered samples (i.e., samples from all classes appear in random order). Error bars are SEM across 10 random seeds.

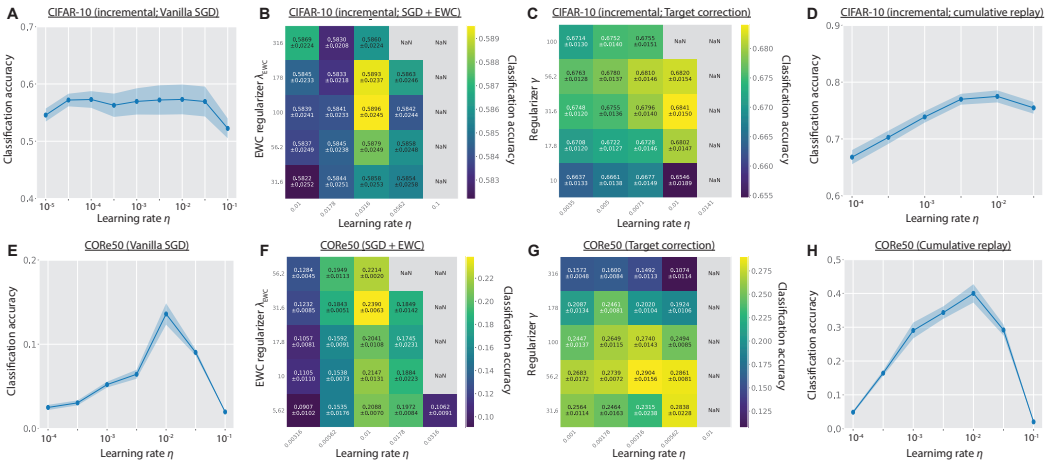


Figure 7: Hyperparameter dependence of nonlinear model training. **(A)** Learning-rate η dependence of vanilla SGD performance in the domain-incremental CIFAR-10 task. **(B-C)** Classification accuracy of EWC (B) and target correction (C) in the domain-incremental CIFAR-10 task under various learning-rate η and the regularizer strength γ **(D)** Performance of SGD with cumulative relpay in the domain-incremental CIFAR-10 task under various learning rate η **(E-H)** The same as panels A-D, but under the class-incremental CORe50 task. Error bars are SEMs over random seeds.

C.3 Application to Non-linear Models (Figs. 3, 6, and 7)

Network Architecture

We used a small convolutional neural network (CNN) for the image classification experiments on CIFAR-10. The network consisted of two convolutional layers followed by a single fully connected output layer. The first convolutional layer used 32 filters with 3×3 kernels and padding of 1, followed by a ReLU activation and average pooling with a 2×2 window and stride 2. The second convolutional layer used 64 filters with 3×3 kernels and padding of 1, followed by another ReLU activation. Adaptive average pooling was then used to obtain a 7×7 spatial feature map, which was flattened into a 64×7×7-dimensional feature vector. For classification, this feature vector was mapped directly to the output classes using a single linear layer. All convolutional and linear layers were used without bias terms to suppress rank-1 correlations in empirical NTK. Unless otherwise specified, convolutional and linear layers were initialized using the default PyTorch initialization.

For the CORe50 dataset, we used a pre-trained ResNet-18 model as the feature extractor. The parameters of the pre-trained feature extraction layers were kept frozen throughout training. On top of the frozen ResNet-18 features, we added a two-layer fully connected classifier head, consisting of a 100-dimensional hidden layer with a ReLU nonlinearity and a 50-dimensional output layer corresponding to the 50 target classes in CORe50. Only the parameters of these two fully connected layers were trained and updated. Both linear layers were used without bias terms.

DATASETS

For Fig. 3A, the 10 classes in CIFAR-10 were randomly partitioned into five ordered binary classification tasks and presented to the model sequentially. We randomly sub-sampled half of the training samples to keep the size of the kernel matrix moderate; as a result, each task consisted of 6,000 samples presented in a single pass with mini-batches. For testing, we used the full 10^4 test samples from the CIFAR-10 dataset.

For Fig. 3B, we followed the CORe50 task setting used in Lomonaco & Maltoni (2017). This is a 50-class class-incremental object recognition setting in which new object classes are introduced sequentially across batches. The first batch, or task, contains 10 classes, while each of the remaining eight batches contains 5 classes, resulting in 50 classes in total. Following Lomonaco & Maltoni (2017), the classes were selected using a biased policy that promotes broad categorical representation by spreading objects from the same category across different batches. The only difference from the original setting is that, instead of using video frames sampled at 20 frames per second, we used 1 frame per second to reduce computational cost.

For Fig. 6B, we evaluated the standard 10-class image classification on CIFAR-10 with a random sample order. Here, instead of strictly online single-pass learning, we considered a multi-pass learning process, since a single pass was not sufficient to achieve decent generalization performance. The order of training samples was randomly shuffled once at the beginning of training and then repeated across epochs.

LEARNING ALGORITHMS

We trained the networks by minimizing the mean-squared error (MSE) loss with stochastic gradient descent (SGD) without momentum using various regularizers and corrected targets as specified below. In all update rules below, we set the mini-batch size of the weight update to 4.

Vanilla SGD The vanilla SGD baseline was trained sequentially on samples in the prescribed order, without target correction, replay, or additional regularization. We used $\eta = 0.01$ in Figs. 3A and B, and $\eta = 0.03$ in Fig. 6B. See Figs. 7A and E for the learning rate dependence.

SGD with iterative target correction For the iterative target correction method, corrected targets were computed using Eq. 16, but with a block-wise upper-triangular matrix K^{bU} instead of the full upper-triangular matrix K^U, to reflect the mini-batch SGD update. We set the size of iterative target correction to $b = 20$, and set $\gamma_o = 0$. We coarsely optimized η and γ numerically for each task setting. The block size of K^{bU} was set to 4, matching the mini-batch size used in SGD. The corrected targets were then used in place of the original one-hot labels when computing the MSE loss and the corresponding gradient. Note that the parameter η in Eq. 17, which is used in Eq. 16, denotes the learning rate used for the SGD weight update.

The empirical NTK was computed with respect to the trainable parameters of the current network. For CORe50, where the pre-trained ResNet-18 feature extractor was frozen, the empirical NTK was computed only with respect to the trainable fully connected classifier head. The specific schedule for updating the empirical NTK depended on the experimental setting.

For Fig. 3A, corresponding to the CIFAR-10 domain-incremental binary classification setting, the empirical NTK was updated at the beginning of each binary task, i.e., every 6,000 training samples. We used $\eta = 0.01$ and $\gamma = 31.6$; see Figs. 7C. In the fixed-K variant, shown in Fig. 6A, the empirical NTK was computed only once using the initial network parameters, and the same corrected targets were reused across all epochs.

For Fig. 3B, corresponding to the CORe50 class-incremental setting, each encountered batch was trained for 10 epochs. After each epoch within an encountered batch, we recomputed the empirical NTK using the current model parameters and recalculated the corrected targets. For later encountered batches, target correction was computed based on data from previously encountered batches together with the current batch, following the sequential structure of the CORe50 setting. The hyperparameters were set to $\eta = 0.00316$ and $\gamma = 56.2$; see Figs. 7G.

For Fig. 6B, corresponding to the standard CIFAR-10 10-class classification with a fixed shuffled order, we updated the kernel at the beginning of each epoch, as we used multi-epoch training. We used a learning rate of $\eta = 0.01$ and regularizer $\gamma = 30.0$.

Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) For the SGD+EWC baseline, we added a quadratic regularization term to the MSE loss to penalize changes in parameters that were important for previously learned tasks or encountered batches. After each task or encountered batch τ, we stored the trainable parameters $\boldsymbol{w}^{*,(\tau)}$ and estimated the parameter importance $F_i^{(\tau)}$ using a diagonal approximation, where each diagonal entry was computed from the squared gradient of the loss with respect to parameter $\boldsymbol{w}_i$. During training on the current task or encountered batch t, the objective was

$$\mathcal{L}_{\mathrm{EWC}}^{(t)}(\boldsymbol{w}) = \mathcal{L}_{\mathrm{MSE}}^{(t)}(\boldsymbol{w}) + \frac{\lambda_{\mathrm{EWC}}}{2} \sum_{\tau<t} \sum_i F_i^{(\tau)} \left(\boldsymbol{w}_i - \boldsymbol{w}_i^{*,(\tau)} \right)^2 . \tag{114}$$

where $\boldsymbol{w}_i$ denotes the current parameter, $\boldsymbol{w}_i^*$ denotes the stored parameter value after previous training, and F_i is the estimated importance weight. We used $(\eta, \lambda_{\mathrm{EWC}}) = (0.00316, 100)$ for Figs. 3A and $(0.01, 31.6)$ fo Figs. 3B (see Figs. 7B and F for parameter dependence).

Cumulative Replay For the cumulative replay baseline, the model was trained using all samples encountered so far. That is, after a new task or encountered batch was introduced, the training set was expanded to include both the new data and all previously observed data. This baseline therefore provides a reference for performance when previous data remain available for replay. In both continual CIFAR-10 and CORe50 tasks, we set the learning rate $\eta = 0.01$.

Soft Label (SL) In the soft label method implemented in Table 1, one-hot labels are replaced by softer targets. The one-hot target $y \in \{0,1\}^C$ of each sample is replaced by

$$\tilde{y} = (1-\varepsilon)\, y + \frac{\varepsilon}{C}\mathbf{1}, \tag{115}$$

where C is the number of output classes and $\varepsilon \in [0,1]$ is the smoothing strength. Training minimizes the MSE between the network outputs and $\tilde{y}$. When $\varepsilon = 0$, $\tilde{y}$ reduces to the original one-hot label.

Memory-based Methods

All memory-based methods use the same sample-count budget $M = 0.1 N_{\mathrm{train}}$. For ER, DER, DER++, and DER + TC this is the replay-buffer capacity; for sparse TC it is the number of stored anchors used to construct corrected targets.

Let $(x_t, y_t)_{t\geq 1}$ be the stream and $\mathcal{M}$ a buffer of capacity M. After seeing the t-th sample, reservoir sampling updates $\mathcal{M}$ by

$$\mathcal{M} \leftarrow \begin{cases} \mathcal{M} \cup \{e_t\}, & \text{if } t \leq M, \\ \mathcal{M} \setminus \{e\} \cup \{e_t\} \text{ with probability } M/t, & \text{if } t > M, \end{cases} \tag{116}$$

where e (the sample dropped from the buffer) is drawn uniformly from the current buffer and e_t is the samples stored for sample t. If $t > M$ and the sample is not accepted, $\mathcal{M}$ is left unchanged. This keeps $\mathcal{M}$ an approximately uniform subset of the examplesseen so far.

On multi-epoch streams (CORe50), the same rule is applied to *unique* samples only: each original index is considered at most once, on its first occurrence. Repeated views in later epochs neither re-enter the buffer nor increment t.

Matching M matches the number of stored inputs, not bytes: DER-style methods additionally store a teaching vector per slot.

ER, DER, and DER++ ER stores (x, y) and replays hard labels. DER (Buzzega et al., 2020) stores the stop-gradient prediction $\hat{y} = f_{\theta_t}(x_t)|_{\text{stopgrad}}$ at insertion time and replays that frozen output. DER++ (Buzzega et al., 2020) stores $(x, y, \hat{y})$ and uses both replay terms:

$$\mathcal{L}_{\text{ER}} = \frac{1}{|\mathcal{B}_t|} \sum_{(x,y)\in\mathcal{B}_t} \ell(x, y) + \frac{\beta}{|\mathcal{B}_\mathcal{M}|} \sum_{(x',y')\in\mathcal{B}_\mathcal{M}} \ell(x', y'), \tag{117a}$$

$$\mathcal{L}_{\text{DER}} = \frac{1}{|\mathcal{B}_t|} \sum_{(x,y)\in\mathcal{B}_t} \ell(x, y) + \frac{\alpha}{|\mathcal{B}_\mathcal{M}|} \sum_{(x',\hat{y}')\in\mathcal{B}_\mathcal{M}} \ell(x', \hat{y}'), \tag{117b}$$

$$\mathcal{L}_{\text{DER++}} = \mathcal{L}_{\text{DER}} + \frac{\beta}{|\mathcal{B}_\mathcal{M}|} \sum_{(x',y')\in\mathcal{B}_\mathcal{M}} \ell(x', y'). \tag{117c}$$

Thus $\beta = 0$ in DER++ recovers DER, and $\alpha = 0$ recovers ER.

Target Correction (TC) with Sparse Samples For Fig. 3C, we used sparse subset of samples for target correction. For each task already encountered, a fraction of one training epoch is kept as anchors. Within the task currently being learned, correction is applied to the samples of the current epoch in order, so samples earlier in that epoch are also anchors for the samples that follow. A fraction of 0 uses no samples from previous tasks (i.e., vanilla SGD); a fraction of 100% uses one full epoch from every previous task. In this setting, the reservoir $\mathcal{M}$ of pairs (x, y) corresponds to the anchor set for the iterative correction in Eq. 16. Admission into the buffer $\mathcal{M}$ followed Eq. 116.

DER + TC DER+TC keeps DER's current-batch loss on y and the same reservoir loop, but the replay target is a corrected label z^c rather than a frozen prediction $\hat{y}$. When a new block is considered for insertion, z^c is computed from Eq. 16 using the *current* reservoir true labels $Y_\mathcal{M}$ as the prefix; previously stored z^c values are not reused as KRR targets. Accepted entries store (x, y, z^c). The objective is

$$\mathcal{L}_{\text{DER+TC}} = \frac{1}{|\mathcal{B}_t|} \sum_{(x,y)\in\mathcal{B}_t} \ell(x, y) + \frac{\alpha}{|\mathcal{B}_\mathcal{M}|} \sum_{(x',z^{c\prime})\in\mathcal{B}_\mathcal{M}} \ell(x', z^{c\prime}). \tag{118}$$

DER and DER+TC therefore share the replay mechanism and differ only in the stored teaching signal.

IMPLEMENTATION

The small CNN used in the CIFAR-10 experiments was implemented using torch.nn, whereas the pre-trained ResNet-18 model used for CORe50 was constructed from torchvision.models. The experiments were conducted on NVIDIA H100 GPUs and on additional GPUs. Execution time of the nonlinear models was around a few hours per run.