Hyperbolic Displacement-Constrained Adaptation for CLIP-Based Class-Incremental Learning

Quan Cheng, Zhen-Hao Xie, Lan Li, Da-Wei Zhou, Lijun Zhang*
State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China
School of Artificial Intelligence, Nanjing University, Nanjing, China
{chengq, xiezh, lil, zhoudw, zhanglj}@lamda.nju.edu.cn

Abstract

Class-incremental learning aims to learn a stream of new classes while preserving knowledge from prior tasks, but suffers from catastrophic forgetting when the model is adapted across sessions. CLIP-based approaches have recently gained popularity in this setting due to the strong transferable representations of vision-language pretrained models. Most of them resist forgetting by retaining data-derived class memory such as replay samples, visual prototypes, or class distribution information. In contrast, we ask whether stable adaptation can be achieved by constraining adapter-induced feature displacement without retaining any data-derived class memory. We propose **H**yperbolic **D**isplacement-**C**onstrained **A**daptation (HDCA), which keeps the direction of each adapter output but compresses its norm, so small outputs remain almost unchanged while large or accumulated ones are suppressed. At the per-task level, a hyperbolic low-rank adapter compresses the output of each task adapter. At the multi-task level, a cumulative compression stage further suppresses the summed output of all task adapters. On the classifier side, HDCA introduces a parameter-free Poincaré distance classifier that preserves cosine-based ranking for fixed adapted features but reshapes the training gradient in high-similarity regions. We conduct experiments on five standard class-incremental datasets, and the results comprehensively validate the effectiveness of HDCA without retaining any data-derived class memory during training and inference.

1 Introduction

Class-incremental learning (CIL) trains a model on a stream of new classes while preserving discrimination over previously seen ones [Zhou et al., 2024c]. Its central difficulty is catastrophic forgetting, where updates that improve adaptation to the current session overwrite or distort the decision structure needed for earlier sessions, especially when old training data cannot be revisited [Wang et al., 2024, Zhou et al., 2024b]. Vision-language pretrained models such as CLIP [Radford et al., 2021] provide an attractive foundation for CIL, because their image-text alignment yields transferable visual representations and a broad open-vocabulary classifier over class names. Since prediction depends on the relative geometry between image and text features, incremental updates must preserve the image-text alignment inherited from pretraining, not only the visual representation. Many existing CLIP-based CIL methods, such as PROOF [Zhou et al., 2025b] and RAPF [Huang et al., 2024], mitigate forgetting by retaining data-derived class memory, including replay samples, visual prototypes, or class distribution information. Such memory reintroduces old-class constraints during incremental training and helps preserve prior decision regions, but the stability mechanism remains external to the adaptation module itself. This raises a natural question: can CLIP be adapted stably across incremental sessions without storing any data-derived class memory?

*Lijun Zhang is the corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

We approach this question through adapter-based parameter-efficient fine-tuning (PEFT) [Han et al., 2024], which keeps the pretrained CLIP backbone frozen while lightweight task-specific modules absorb per-session changes. Freezing the backbone and earlier adapters, however, does not freeze the features: the outputs of all task adapters act jointly in every forward pass and shift the final image and text features away from their pretrained counterparts. We call this shift adapter-induced feature displacement. As it accumulates over sessions, it can push old-class features off the pretrained image-text geometry and thus become a source of forgetting. This motivates making the adapter function itself the source of stability, leaving small adapter outputs nearly intact to preserve plasticity while structurally suppressing large or accumulated ones.

This displacement-control objective is nontrivial under standard additive adapter designs, which typically insert LoRA-style low-rank adapters into frozen backbone layers and sum their outputs $\delta_k(x) = B_k A_k x$ as $\sum_k \delta_k(x)$ [Hu et al., 2022]. Optimization-level factors such as weight decay, learning-rate schedules, and step counts keep individual updates well behaved but impose no functional constraint on how the outputs of different task adapters combine. When these outputs align in direction, additive aggregation amplifies the accumulated norm across sessions and can distort the CLIP image-text geometry. Controlling this accumulation therefore calls for a structural constraint built into the adapter function itself, rather than regulation through training dynamics alone.

Recent advances in parameter-efficient tuning offer a natural starting point. HypLoRA [Yang et al., 2025] reformulates low-rank adaptation on the Lorentz manifold, avoiding the map-cancellation issue of tangent-space hyperbolic operations and introducing norm-dependent higher-order terms that improve single-task PEFT. We draw inspiration from HypLoRA and bring hyperbolic geometry into CLIP-based CIL, because its Lorentz construction acts as a norm compression on the low-rank output that is nearly the identity for small outputs, logarithmic for large ones, and smooth enough to preserve useful gradients, which together match the structural constraint we seek. HypLoRA, however, is designed for single-task PEFT and leaves two questions open in our setting: how to control the accumulation of adapter outputs when multiple task adapters are combined across sessions, and how to expose the adapted features to a compatible hyperbolic training signal.

We propose **H**yperbolic **D**isplacement-**C**onstrained **A**daptation (HDCA) to address these two questions. On the feature side, HDCA keeps the direction of each adapter output but compresses its norm at two levels. At the per-task level, a hyperbolic low-rank adapter compresses the output of each task adapter. At the multi-task level, cumulative compression is applied again to the sum of all adapter outputs. As a result, small adapter outputs pass almost unchanged, whereas large outputs, or outputs that accumulate along similar directions, are suppressed. On the classifier side, HDCA replaces the standard cosine classifier with a Poincaré distance classifier, which supplies this hyperbolic training signal. For adapted features, this classifier preserves the cosine-induced class ranking, but it reshapes the training gradient by emphasizing high-similarity regions where semantically close text features tend to cluster. HDCA thus extends HypLoRA from single-task PEFT to CLIP-based CIL by jointly addressing the accumulation of adapter outputs across sessions and hyperbolic distance-based training.

Our main contributions are summarized below.

- We propose HDCA, a CLIP-based CIL framework that constrains adapter-induced feature displacement through two-level compression, applied first to the output of each task adapter and then to the sum of adapter outputs across sessions.
- We introduce a parameter-free Poincaré distance classifier that preserves cosine-based ranking for adapted features but reshapes the training gradient in high-similarity regions.
- Across five standard class-incremental benchmarks under 10 protocol settings, HDCA surpasses prior CLIP-based continual learning methods in most settings without retaining any data-derived class memory.

2 Related Work

2.1 Class-Incremental Learning with Vision-Language Models

Class-incremental learning (CIL) [Wang et al., 2024, Zhou et al., 2024c] studies how a model absorbs a non-stationary stream of classes while preserving discrimination on those it has already seen. Forgetting usually appears as representation drift, interference with old decision boundaries, or

prediction bias toward recently learned classes. Existing CIL approaches address this stability problem along three lines. Replay-based methods rehearse a small buffer of past samples [Rebuffi et al., 2017, Liu et al., 2020b, Sun et al., 2022], optimization-based methods constrain update directions to reduce interference with prior knowledge [Farajtabar et al., 2020, Saha et al., 2021, Wang et al., 2021], and architecture-based methods grow or partition the network for new tasks [Yoon et al., 2018, Li et al., 2019, Liang and Li, 2023]. These mechanisms are effective for controlling forgetting, but they do not directly address the image-text geometry that becomes central in CLIP-based classification.

With pretrained models such as ViT [Dosovitskiy et al., 2021], CIL shifts from training a representation from scratch to adapting an existing one [Zhou et al., 2024b]. Two main lines have emerged. Prompt-based methods learn task-conditioned prompts on a frozen ViT [Wang et al., 2022b,a, Smith et al., 2023, Lu et al., 2024], while adapter-based methods route plasticity through small task-specific modules [Zhou et al., 2023, 2024a, Liang and Li, 2024, Wu et al., 2025]. Freezing most pretrained weights reduces parameter overwriting, but visual-only prompt or adapter mechanisms do not guarantee that adapted image features remain aligned with the text side of CLIP.

CLIP-based CIL adds the constraint that adaptation must preserve the image-text alignment inherited from CLIP pretraining [Radford et al., 2021]. Existing methods preserve alignment through learnable prompts, task-specific parameters, textual guidance, or constrained updates [Zhou et al., 2022, Huang et al., 2024, Zhou et al., 2025a,b], but their stability typically relies on retained data-derived class memory. HDCA instead constrains the adapter function itself. The output of each task adapter is compressed, the sum of adapter outputs is compressed again across sessions, and the Poincaré distance classifier provides a compatible hyperbolic training signal. Old classes enter only at inference, through their class names; HDCA retains no old images, visual prototypes, class distribution information, or stored text embeddings.

2.2 Hyperbolic Learning

Hyperbolic learning represents data in spaces of constant negative curvature, whose exponential volume growth provides a natural inductive bias for hierarchical and tree-like semantic structures [He et al., 2025]. Early work introduced hyperbolic embeddings in the Poincaré ball and the Lorentz model for learning symbolic and continuous hierarchies [Nickel and Kiela, 2017, 2018]. Subsequent studies developed manifold-aware neural layers and Riemannian optimization algorithms [Ganea et al., 2018, Bécigneul and Ganea, 2019]. In computer vision, hyperbolic geometry has been used for image classification [Khrulkov et al., 2020], zero-shot recognition [Liu et al., 2020a], few-shot recognition [Gao et al., 2021], and semantic segmentation [Atigh et al., 2022]. These works show that hyperbolic spaces can improve visual representations and decision boundaries when category semantics have hierarchical structure.

Hyperbolic geometry has also been explored in vision-language learning. MERU and related hyperbolic vision-language models extend CLIP-style alignment to hyperbolic spaces to better capture visual-semantic hierarchies between images and text [Desai et al., 2023, Ibrahimi et al., 2024, Peng et al., 2025]. These methods mainly modify the representation space, similarity metric, or pretraining objective. Hyperbolic geometry has more recently been connected to parameter-efficient adaptation. Closest to HDCA is HypLoRA [Yang et al., 2025], which reformulates low-rank adaptation on the Lorentz manifold, avoids the map-cancellation issue of tangent-space hyperbolic operations, and introduces norm-dependent nonlinear effects for single-task fine-tuning. HDCA repurposes the norm compression induced by this Lorentz construction to control adapter-induced feature displacement in sequential CLIP-based CIL. Unlike single-task hyperbolic PEFT, HDCA studies how the outputs of multiple frozen task adapters combine over time, adds cumulative compression to suppress their aligned accumulation, and uses a Poincaré distance classifier to shape the training signal while keeping classification tied to CLIP text features.

3 Methodology

3.1 Preliminaries and Overview

We consider a standard class-incremental stream of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_N$, where task $\mathcal{T}_t$ introduces a disjoint class set $\mathcal{C}_t$. During session t, the model is trained only on the current dataset $\mathcal{D}_t$. After session t, the model is evaluated on all classes seen so far, $\mathcal{C}_{\leq t} = \bigcup_{k \leq t} \mathcal{C}_k$, using only their class

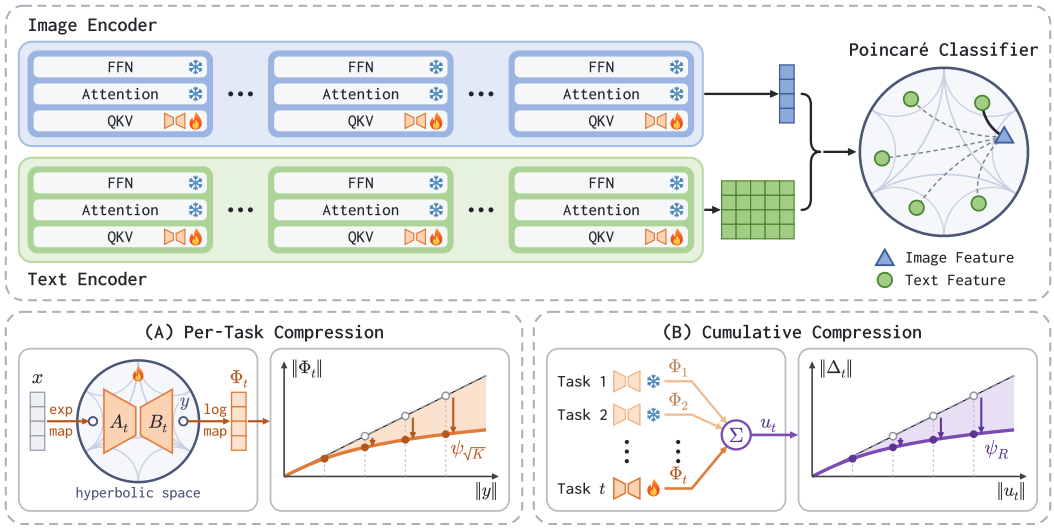

Figure 1: Overview of HDCA. In each session, a low-rank adapter is attached to the query and value projections of the frozen CLIP image and text encoders, and the adapted image and text features are compared in the Poincaré ball. (A) Per-Task Compression: each input token x is mapped into hyperbolic space by the exponential map, transformed there by the low-rank adapter (A_t, B_t), and mapped back by the logarithmic map. Mapping back compresses the norm of the low-rank output y by $\psi_{\sqrt{K}}$, little for small outputs and more for large ones (dashed: identity). (B) Cumulative Compression: the compressed outputs Φ_k of the frozen earlier adapters ($k < t$) and Φ_t of the trainable current adapter are summed into u_t, whose norm can grow with the number of sessions when the outputs align. The map ψ_R compresses it to Δ_t, which is added to the frozen projection output Wx.

names; no data-derived memory from previous sessions is retained. We start from a CLIP backbone with a frozen image encoder f_v^0 and a frozen text encoder f_t^0, both producing d-dimensional features that are normalized to unit length. For an image x and a prompted class name τ_c, the original CLIP classifier scores class c by

$$\ell_c^{\cos}(x) = \gamma \cdot \cos\big(f_v^0(x),\, f_t^0(\tau_c)\big), \tag{1}$$

where γ is the pretrained CLIP logit scale.

A common way to adapt the frozen backbone is to insert LoRA-style adapters into linear projections. For a projection W and input token x, the adapter of task t adds a low-rank output

$$Wx \;\mapsto\; Wx + \delta_t(x), \qquad \delta_t(x) = M_t x, \quad M_t = \frac{\alpha}{q} B_t A_t, \tag{2}$$

where q is the adapter rank and α/q is the LoRA scale. LoRA adapters are parameter-efficient and leave the CLIP weights frozen, but their outputs still displace the features used for image-text alignment. Let $f_v^{(t)}$ and $f_t^{(t)}$ denote the encoders after all adapter pairs learned up to session t have been attached, and let $z_v^{(t)}$ and $e_c^{(t)}$ be their unit-normalized image and text features. A single small adapter output may be harmless, but outputs introduced by different sessions can align and accumulate, gradually moving the adapted features away from the geometry inherited from CLIP pretraining. As illustrated in Figure 1, HDCA therefore separates the problem into two coupled design choices, namely how to keep adapter outputs from accumulating into large feature displacement and how to train the adapted features with a compatible hyperbolic distance signal.

HDCA keeps the intervention on CLIP minimal. On the feature side, we insert low-rank adapters into the query and value projections of each transformer block in the image encoder and apply the same construction to the text encoder. We adapt the text encoder because class names define the decision geometry in CLIP-based CIL; text features, however, are recomputed from names rather than stored, and the same two-level compression is applied to keep text features from drifting excessively. Adapter outputs pass through two compression stages: the first compresses the output of each task adapter, and the second compresses the sum of these outputs across sessions. Both stages are nearly the identity for small outputs and logarithmic for large ones, so they preserve plasticity near the

pretrained representation while suppressing large outputs. On the classifier side, we replace the cosine logit in Eq. (1) with a parameter-free Poincaré distance logit. Given $z_v^{(t)}$ and $e_c^{(t)}$, the classifier projects both features into the Poincaré ball with a fixed map P and predicts with

$$\ell_c(x) = -\sigma \cdot d_H\big(P(z_v^{(t)}),\, P(e_c^{(t)})\big), \tag{3}$$

where d_H is the hyperbolic geodesic distance on the ball and $\sigma > 0$ is a fixed scale. Across sessions, HDCA adds one task-specific low-rank adapter pair at each selected projection while keeping the CLIP backbone and earlier adapters frozen. The following subsections define the two-level adapter and the Poincaré classifier.

3.2 Hyperbolic Adapter with Two-Level Compression

We now formalize the two-level adapter introduced above. HDCA uses the hyperbolic adapter mainly as a smooth norm compressor. Given an adapter output y, it keeps the direction of y and replaces its norm $r = \|y\|$ with $a\,\mathrm{arcsinh}(r/a)$. This map behaves like the identity near zero and grows only logarithmically for large r, preserving small adapter outputs while suppressing large ones. The Lorentz construction below yields this map from a direct hyperbolic low-rank transformation.

Consider a frozen query or value projection W inside a multi-head attention block. For task t, the adapter matrices satisfy $A_t \in \mathbb{R}^{q\times \dim(x)}$ and $B_t \in \mathbb{R}^{\dim(Wx)\times q}$, with $M_t = \frac{\alpha}{q} B_t A_t$ as in Eq. (2). With ordinary LoRA, the projection would use $Wx + \sum_{k\le t} M_k x$, and the summed output could grow linearly when the outputs of different tasks align. HDCA keeps the same low-rank factors but changes how their outputs are generated and combined.

Hyperbolic lift and return. We use the Lorentz model with curvature parameter $K > 0$ (constant curvature $-1/K$), given by

$$\mathbb{L}_K^m = \{y = (y_0, y_{\mathrm{sp}}) \in \mathbb{R}^{m+1} : \langle y, y\rangle_L = -K,\ y_0 > 0\}, \qquad \langle y, z\rangle_L = -y_0 z_0 + y_{\mathrm{sp}}^\top z_{\mathrm{sp}}. \tag{4}$$

The origin is $o_K = (\sqrt{K}, 0)$, and we identify a Euclidean vector $v \in \mathbb{R}^m$ with the tangent vector $(0, v) \in T_{o_K}\mathbb{L}_K^m$. The exponential map from the tangent space to the Lorentz manifold is

$$\Pi_{\exp}^K(v) = \exp_{o_K}^K(0, v) = \big(\sqrt{K}\cosh(\|v\|/\sqrt{K}),\ \sqrt{K}\sinh(\|v\|/\sqrt{K})\, v/\|v\|\big), \tag{5}$$

with the spatial component defined by continuity at $v = 0$. Conversely, for $y = (y_0, y_{\mathrm{sp}}) \in \mathbb{L}_K^m$, the spatial coordinate of the logarithmic map at o_K is

$$\Pi_{\log}^K(y) = \frac{\sqrt{K}\ \mathrm{arccosh}(y_0/\sqrt{K})}{\|y_{\mathrm{sp}}\|} y_{\mathrm{sp}} = \frac{\sqrt{K}\ \mathrm{arcsinh}(\|y_{\mathrm{sp}}\|/\sqrt{K})}{\|y_{\mathrm{sp}}\|} y_{\mathrm{sp}}, \tag{6}$$

again understood by continuity when $y_{\mathrm{sp}} = 0$. Thus a Euclidean token is lifted from the tangent space at o_K to the Lorentz manifold, and the transformed point is returned to Euclidean coordinates by $\Pi_{\log}^K$.

Direct Lorentz low-rank transformation. Following the direct Lorentz construction of HypLoRA [Yang et al., 2025], HDCA applies the low-rank matrix to the spatial component and recomputes the time component so the output remains on the Lorentz manifold. Let $x^H = \Pi_{\exp}^K(x) = (x_0^H, x_{\mathrm{sp}}^H)$. We define

$$\mathrm{LLR}(M_t, x^H) = \big(\sqrt{\|M_t x_{\mathrm{sp}}^H\|^2 + K},\ M_t x_{\mathrm{sp}}^H\big). \tag{7}$$

This point lies in $\mathbb{L}_K^{\dim(Wx)}$ because $-(\|M_t x_{\mathrm{sp}}^H\|^2 + K) + \|M_t x_{\mathrm{sp}}^H\|^2 = -K$. The output of the task-t adapter is

$$\Phi_t(x) = \Pi_{\log}^K\big(\mathrm{LLR}\big(M_t, \Pi_{\exp}^K(x)\big)\big). \tag{8}$$

Expanding the logarithmic map gives the closed form used by the adapter,

$$\Phi_t(x) = \frac{\sqrt{K}\ \mathrm{arcsinh}\big(\|M_t x_{\mathrm{sp}}^H\|/\sqrt{K}\big)}{\|M_t x_{\mathrm{sp}}^H\|} M_t x_{\mathrm{sp}}^H. \tag{9}$$

Eq. (9) is the first compression stage (Figure 1A). It preserves the direction of $M_t x_{\mathrm{sp}}^H$ and changes only its norm.

Properties of the compression map. To characterize this stage, define the general compression map

$$\psi_a(y) = \frac{a \operatorname{arcsinh}(\|y\|/a)}{\|y\|} y, \qquad a > 0, \tag{10}$$

with the value at $y = 0$ given by continuity. Eq. (9) is exactly $\Phi_t(x) = \psi_{\sqrt{K}}(M_t x_{\mathrm{sp}}^H)$. For $r = \|y\|$, its output norm is

$$\|\psi_a(y)\| = a \operatorname{arcsinh}(r/a). \tag{11}$$

Since $\operatorname{arcsinh}(s) \le s$ for $s \ge 0$, the map never increases the norm, $\|\psi_a(y)\| \le \|y\|$. It also has two useful regimes,

$$\psi_a(y) = \left(1 - \frac{\|y\|^2}{6a^2} + O\left(\frac{\|y\|^4}{a^4}\right)\right) y \quad (\|y\| \ll a), \tag{12}$$

and

$$\|\psi_a(y)\| = a \log(2\|y\|/a) + O(a^3/\|y\|^2) \quad (\|y\| \gg a). \tag{13}$$

Thus small outputs pass nearly unchanged, while large outputs affect the projection only logarithmically. Unlike hard clipping or $\tanh$ saturation, the $\operatorname{arcsinh}$ form avoids flat saturated regions and keeps useful gradients for moderately large outputs.

Cumulative compression. During session t, each adapter location contains $t-1$ frozen low-rank pairs and one trainable pair. Direct composition would sum their compressed outputs,

$$u_t(x) \;=\; \sum_{k=1}^{t} \Phi_k(x). \tag{14}$$

This Euclidean sum can still grow quickly because outputs learned on related representations need not point in unrelated directions. When they align, $\|u_t(x)\|$ can increase roughly linearly with the number of sessions, even though each output has already been compressed.

HDCA therefore applies the same compression map, with its own scale R, to this sum (Figure 1B),

$$\Delta_t(x) \;=\; \psi_R\big(u_t(x)\big), \qquad R > 0, \tag{15}$$

with final projection output $Wx + \Delta_t(x)$. The scale R controls when the summed output leaves the near-identity regime. If the uncompressed sum grows as $\|u_t(x)\| = O(t)$ along an aligned direction, Eq. (13) gives $\|\Delta_t(x)\| = O(R \log t)$. The two stages therefore operate at different levels: $\sqrt{K}$ sets the compression of each task output before summation, and R sets the compression of the sum added to the frozen projection.

The outer compression intentionally couples tasks. Although earlier adapter parameters (A_k, B_k) for $k < t$ are frozen, the scale factor in $\psi_R(u_t(x))$ depends on the norm of the summed output $\|u_t(x)\|$. A new task can therefore slightly rescale the contributions of older adapters at inference, which is the mechanism that prevents aligned outputs from accumulating linearly. This coupling introduces a stability-plasticity trade-off, where the parameters of old adapters are fixed yet their effective contribution can change through the shared scale factor. During session t, only the current pair (A_t, B_t) is trainable, and gradients through the aggregator are stopped for all earlier Φ_k, avoiding direct optimization of past modules while the norm of the final output remains controlled at inference. We apply the same two-level structure to the text encoder with the same K and R.

The adapter specifies how features move. The next subsection specifies the compatible classifier used to compare the adapted image features with text features.

3.3 Poincaré Distance Classifier

Rather than introducing a learnable hyperbolic multinomial classifier, we keep the matching rule parameter-free and replace the cosine logit with a Poincaré distance-based score. The parameter-free design keeps the classifier tied to CLIP text features while exposing the adapters to a hyperbolic training signal.

We project the unit-normalized image and text features into the Poincaré ball with a fixed scaling

$$P(z) \;=\; \rho \cdot z, \qquad \rho \in (0, 1), \tag{16}$$

where ρ keeps all points strictly inside the open unit ball. Since both image and text features are unit-normalized before projection, every projected point has norm ρ. The hyperbolic geodesic distance between $P(z_v^{(t)})$ and $P(e_c^{(t)})$ is therefore

$$d_H\big(P(z_v^{(t)}), P(e_c^{(t)})\big) \;=\; \operatorname{arccosh}\Big(1 + \frac{2\,\|P(z_v^{(t)}) - P(e_c^{(t)})\|^2}{(1-\rho^2)^2}\Big), \tag{17}$$

which is plugged into Eq. (3) with the fixed scale $\sigma > 0$.

A useful consequence is that the Poincaré classifier does not introduce an opaque change in the inference rule. For any fixed $\rho \in (0, 1)$, Eq. (17) is a strictly decreasing function of the cosine similarity $s = \cos(z_v^{(t)}, e_c^{(t)})$, so the Poincaré logit gives the same class ranking as a cosine logit applied to the same adapted features. The main effect is instead to reshape the training gradient. Differentiating the distance with respect to s shows that its magnitude grows sharply as $s \to 1$, so the Poincaré logit amplifies gradients in the high-similarity region where CLIP text features from semantically related classes often cluster. The closed-form gradient and the numerical-stability details are deferred to Appendix B.

Both ρ and σ are scalar hyperparameters fixed at initialization and never updated. Training uses the same logits in a cross-entropy loss over the classes introduced in the current session,

$$\mathcal{L}_t \;=\; -\,\mathbb{E}_{(x,y)\sim\mathcal{D}_t}\Big[\log \frac{\exp(\ell_y(x))}{\sum_{c\in\mathcal{C}_t}\exp(\ell_c(x))}\Big], \tag{18}$$

where the logits ℓ_c are defined by Eq. (3). Classes from earlier sessions do not enter the training loss. No previous image, feature, prototype, class distribution information, or stored text embedding is reused. At inference, all adapter pairs collected so far are active, image features are produced by the adapter-augmented image encoder, and text features are recomputed from class names through the adapter-augmented text encoder for $c \in \mathcal{C}_{\le t}$ and then discarded. The prediction is the argmax of Eq. (3) over $\mathcal{C}_{\le t}$. HDCA is therefore memory-free with respect to data-derived class information, while still growing parameters linearly by adding one low-rank adapter pair per task at each selected projection.

4 Experiments

We evaluate HDCA on five class-incremental learning datasets under 10 protocol settings against representative pretrained-model and CLIP-based baselines. We then dissect each design choice through component ablations and examine how compression relates to feature displacement and forgetting.

4.1 Experimental Setup

Datasets and Protocols. Following prior work [Zhou et al., 2025a], we evaluate on five datasets that differ in their distribution gap from CLIP's pretraining data, namely CIFAR100 [Krizhevsky, 2009], ObjectNet [Barbu et al., 2019], ImageNet-R [Hendrycks et al., 2021], StanfordCars [Krause et al., 2013], and Food101 [Bossard et al., 2014]. We use 100 classes for CIFAR100, StanfordCars, and Food101 and 200 classes for ObjectNet and ImageNet-R, matching the splits in [Zhou et al., 2025b]. We adopt the B-m Inc-n convention [Zhou et al., 2024a] with two configurations per dataset, where B0 Inc-n presents all classes incrementally and B-m Inc-n places half of the classes in the base session, yielding 10 settings. The class order is fixed by a single random seed across all baselines.

Baselines and Metrics. We compare with prompt-based CIL methods L2P [Wang et al., 2022b], DualPrompt [Wang et al., 2022a], CODA-Prompt [Smith et al., 2023], CoOp [Zhou et al., 2022], and with prototype-retaining CLIP-based methods RAPF [Huang et al., 2024] and MG-CLIP [Huang et al., 2025]. We additionally include three exemplar-based methods, iCaRL [Rebuffi et al., 2017], MEMO [Zhou et al., 2023], and PROOF [Zhou et al., 2025b], on a subset of datasets, each storing 20 images per class (Table 2). The zero-shot CLIP classifier (ZS-CLIP) [Radford et al., 2021] and naive sequential fine-tuning (Finetune) serve as a memory-free reference and a lower bound. All baselines share the same pretrained CLIP backbone. Following [Rebuffi et al., 2017, Zhou et al., 2025a], we report the final accuracy $\mathcal{A}_B$ after the last session and the average accuracy $\bar{\mathcal{A}} = \frac{1}{B}\sum_{b=1}^{B}\mathcal{A}_b$, where $\mathcal{A}_b$ is the top-1 accuracy on all classes seen through session b.

Table 1: Comparison of class-incremental learning methods on five benchmarks. The top sub-table reports the B0 Inc-n protocol where all classes are presented incrementally, and the bottom sub-table reports the B-m Inc-n protocol where half of the classes are placed in the base session. We report average accuracy $\bar{\mathcal{A}}$ and final accuracy $\mathcal{A}_B$. Best results are highlighted in **bold** and second-best results are underlined. All methods are initialized from the same pretrained CLIP model to ensure fair comparison.

Method	CIFAR100 B0 Inc-10		Cars B0 Inc-10		Food B0 Inc-10		ImageNet-R B0 Inc-20		ObjectNet B0 Inc-20	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	7.84	4.44	3.14	1.10	3.49	1.71	1.37	0.43	1.34	0.47
CoOp	47.00	24.24	36.46	21.65	36.01	14.18	60.73	37.52	21.24	6.29
ZS-CLIP	81.81	71.38	82.60	76.37	87.86	81.92	83.37	77.17	38.43	26.43
L2P	82.74	73.03	76.63	61.82	85.66	77.33	75.97	66.52	51.40	39.39
DualPrompt	81.63	72.44	76.26	62.94	84.92	77.29	76.21	66.65	52.62	40.72
CODA-Prompt	82.43	73.43	80.21	66.47	86.18	78.78	77.69	68.95	46.49	34.13
RAPF	86.14	78.04	82.89	62.85	88.57	81.15	81.26	70.48	48.67	27.43
MG-CLIP	**89.74**	**82.78**	88.21	79.73	88.59	82.35	83.18	77.20	51.41	40.90
HDCA (Ours)	89.28	81.57	**90.66**	**81.01**	**88.73**	**82.52**	**87.69**	**82.30**	**65.99**	**54.42**

Method	CIFAR100 B50 Inc-10		Cars B50 Inc-10		Food B50 Inc-10		ImageNet-R B100 Inc-20		ObjectNet B100 Inc-20	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	5.30	2.46	1.54	1.13	2.14	1.52	1.01	0.88	0.69	0.54
CoOp	41.23	24.12	37.40	20.87	33.13	18.67	54.20	39.77	16.21	6.82
ZS-CLIP	76.49	71.38	78.32	76.37	84.75	81.92	79.57	77.17	31.12	26.43
L2P	81.14	73.61	76.37	65.64	80.42	73.13	72.82	66.77	48.91	42.83
DualPrompt	80.12	72.57	76.88	67.55	80.00	72.75	73.22	67.58	49.08	42.92
CODA-Prompt	78.69	71.58	75.06	64.19	80.98	74.13	73.71	68.05	40.57	34.13
RAPF	82.17	77.93	75.87	63.19	85.53	81.17	76.10	70.23	39.28	28.73
MG-CLIP	85.62	**81.26**	84.58	79.62	28.86	12.51	50.47	35.37	25.00	12.39
HDCA (Ours)	**85.68**	81.03	**88.83**	**83.41**	**86.59**	**82.50**	**85.36**	**82.53**	**60.75**	**52.64**

Implementation Details. Adapters are inserted into the query and value projections of all attention blocks in both encoders. By default, we set $q = 20$, $K = 0.01$, $R = 1.0$, $\rho = 0.7$, and $\sigma = 30$. Algorithm 1 in Appendix A summarizes the training procedure.

4.2 Experimental Results

Table 1 reports $\bar{\mathcal{A}}$ and $\mathcal{A}_B$ for HDCA and the eight compared methods across the 10 settings.

HDCA achieves the best score on 17 of the 20 metric-protocol cells and the second-best on the remaining 3, namely CIFAR100 $\bar{\mathcal{A}}$ and $\mathcal{A}_B$ under B0 Inc-10 and $\mathcal{A}_B$ under B50 Inc-10, where MG-CLIP leads by 0.46, 1.21, and 0.23 points. HDCA's lead depends strongly on the gap between the target classes and the CLIP pretraining distribution. On CIFAR100, Cars, and Food, where CLIP's open-vocabulary classifier already covers the test distribution well, the best memory-retaining baseline stays close to HDCA, leading on most CIFAR100 cells and trailing by sub-point to mid-single-digit margins on Cars and Food. On the domain-shifted ImageNet-R and ObjectNet, the gaps widen considerably, with HDCA exceeding the strongest trainable competitor by 4.51 / 5.10 points on ImageNet-R B0 Inc-20, 9.26 / 12.30 points on B100 Inc-20, 13.37 / 13.52 points on ObjectNet B0 Inc-20, and 11.67 / 9.72 points on B100 Inc-20. This pattern is consistent with our motivation: the more the adapters must change to fit a shifted distribution, the more their outputs can accumulate into feature displacement, which is the regime that two-level compression targets.

A second pattern is the sensitivity of MG-CLIP to the B-m protocol. Although it is the strongest competitor on B0 Inc-10 of CIFAR100, Cars, and Food, it drops to 28.86 / 12.51 on Food B50 Inc-10, 50.47 / 35.37 on ImageNet-R B100 Inc-20, and 25.00 / 12.39 on ObjectNet B100 Inc-20, suggesting that its prototype-based stabilization is sensitive to a large base session followed by small increments. HDCA instead varies smoothly across the two protocols, with $\mathcal{A}_B$ differing by at most 2.40 points

Table 2: Comparison with exemplar-based CIL methods. HDCA uses no exemplars, while iCaRL, MEMO, and PROOF each retain 20 images per class; their results are taken from Zhou et al. [2025b].

Method	CIFAR100 B0 Inc-10		ImageNet-R B0 Inc-20	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
iCaRL	79.91	63.94	72.22	54.38
MEMO	84.67	74.98	80.00	74.07
PROOF	86.70	79.05	85.34	80.10
HDCA (Ours)	**89.28**	**81.57**	**87.69**	**82.30**

Table 3: Component ablation of HDCA's three modules on CIFAR100 B0 Inc-10 and ImageNet-R B0 Inc-20. The first three columns indicate whether Per-Task Compression, Cumulative Compression, and the Poincaré Classifier are enabled (✓) or disabled (✗).

Per-Task Compression	Cumulative Compression	Poincaré Classifier	CIFAR100		ImageNet-R	
			$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
✗	✗	✗	59.19	39.55	30.66	4.48
✓	✗	✗	78.55	66.99	79.27	69.33
✓	✓	✗	81.47	72.14	81.58	74.75
✓	✗	✓	87.92	78.88	86.37	78.47
✓	✓	✓	**89.28**	**81.57**	**87.69**	**82.30**

between B0 and B-m. These observations suggest that controlling adapter-induced displacement can stabilize CLIP adaptation without class memory; Section 4.4 examines the relation between compression, displacement, and forgetting directly.

Beyond the exemplar-free baselines in Table 1, we additionally compare HDCA with the three exemplar-based methods on CIFAR100 B0 Inc-10 and ImageNet-R B0 Inc-20. iCaRL, MEMO, and PROOF each store 20 images per class, while HDCA uses no exemplars at any point in training or inference. Their results are taken from Zhou et al. [2025b], which uses the same CLIP weights and class order as our setting. As reported in Table 2, HDCA outperforms all three replay-equipped competitors, supporting our central claim that constraining adapter-induced feature displacement during training can serve as an alternative to data-based replay as a source of stability.

4.3 Ablation Study

We isolate HDCA's three modules on CIFAR100 (B0 Inc-10) and ImageNet-R (B0 Inc-20), namely Per-Task Compression (Eq. (9)), Cumulative Compression (Eq. (15)), and the Poincaré Classifier (Eq. (3)), each replaced in turn by the plain LoRA output, a plain sum of adapter outputs across tasks, and the cosine logit of Eq. (1), respectively.

Per-Task Compression alone rescues HDCA from the all-disabled collapse where $\mathcal{A}_B$ falls to 4.48 on the 10-session ImageNet-R, yet a clear gap remains. Adding Cumulative Compression raises $\mathcal{A}_B$ by 5.15 and 5.42 points on the two benchmarks, roughly twice its gain in $\bar{\mathcal{A}}$, consistent with suppressing accumulation that surfaces toward the end of the stream. Substituting the Poincaré Classifier for the cosine logit yields a larger gain of 7 to 12 points on top of Per-Task Compression, against 2 to 5 points from Cumulative Compression. Compression thus provides the main protection against forgetting, while the classifier complements it (see also Section 4.4), and the full model outperforms each two-module variant in Table 3 on both benchmarks. Appendix D shows that varying each hyperparameter within the tested range changes $\mathcal{A}_B$ by at most 1.23 points.

4.4 Displacement and Forgetting

We now measure feature displacement and forgetting directly under different levels of compression. On ImageNet-R, we compare three variants over 10 sessions (B0 Inc-20) and 20 sessions (B0 Inc-10): no compression, Per-Task Compression only, and both levels of compression (HDCA). All three use the Poincaré Classifier and, within each stream, share the class order, adapter rank and placement,

Table 4: Feature displacement and forgetting on ImageNet-R over 10 sessions (B0 Inc-20) and 20 sessions (B0 Inc-10). The first two columns indicate whether Per-Task Compression and Cumulative Compression are enabled; all variants use the Poincaré Classifier, and the last row is HDCA. $D_{\cos}$ is the cosine displacement of the final image features from those of the original CLIP (lower is better); higher BWT indicates less forgetting.

Per-Task Compression	Cumulative Compression	10 sessions $\mathcal{A}_B$	10 sessions BWT	10 sessions $D_{\cos}$	20 sessions $\mathcal{A}_B$	20 sessions BWT	20 sessions $D_{\cos}$
✗	✗	9.13	-63.44	0.73	2.88	-53.94	0.79
✓	✗	78.47	-9.38	0.18	72.60	-12.83	0.23
✓	✓	**82.30**	$\mathbf{-6.31}$	**0.10**	**80.52**	$\mathbf{-7.77}$	**0.11**

number of trainable parameters, and optimization settings. Forgetting is measured by backward transfer (BWT), the average change in accuracy on each earlier task from the session in which it was learned to the final session, so a higher BWT means less forgetting. Displacement is measured by $D_{\cos} = 1 - \cos(\hat{f}, \hat{f}^0)$ between the final normalized image features of the adapted model and of the original CLIP, averaged over test images. Appendix C gives the formal definitions and additionally reports text-side displacement, adapter output norms, Euclidean displacement, and old- and new-class accuracy.

Table 4 shows that displacement and forgetting decrease together as compression is added, in both streams. Without compression, the image features move far from their pretrained counterparts and almost all earlier classes are forgotten. Relative to this variant, Per-Task Compression alone reduces image displacement by 75.46% and 71.50% over 10 and 20 sessions, and adding Cumulative Compression raises the reduction to 85.87% and 86.33%. Text features follow the same ordering in both streams (Appendix C). Forgetting decreases accordingly: over 10 sessions, BWT improves from -63.44 to -9.38 with Per-Task Compression and to -6.31 with both levels, and Cumulative Compression raises the final old- and new-class accuracy from 77.70 / 85.66 to 81.77 / 87.22. The classifier alone cannot account for these gains: the variant without compression differs from the first row of Table 3 only in the classifier, which raises $\bar{\mathcal{A}}$ / $\mathcal{A}_B$ from 30.66 / 4.48 to only 38.24 / 9.13.

The benefit of Cumulative Compression grows with the stream length, from 3.83 points in $\mathcal{A}_B$ over 10 sessions to 7.92 points over 20 sessions. When the stream doubles, the $\bar{\mathcal{A}}$ and $\mathcal{A}_B$ of HDCA decrease by 0.37 and 1.78 points, and its BWT changes from -6.31 to -7.77. These controlled comparisons show that lower displacement accompanies less forgetting and that Cumulative Compression contributes beyond Per-Task Compression, although they do not rule out other sources of forgetting.

5 Conclusion

We introduced Hyperbolic Displacement-Constrained Adaptation (HDCA), a CLIP-based class-incremental learning framework that seeks stability by constraining adapter-induced feature displacement rather than by retaining class memory. It does so through two-level compression: the output of each task adapter is compressed, and the sum of adapter outputs is compressed again across sessions. It further uses a parameter-free Poincaré distance classifier to keep classification tied to CLIP text features while reshaping the training signal in high-similarity regions. Across five benchmarks under 10 protocol settings, HDCA improves over representative pretrained-model and CLIP-based continual learning baselines in most settings without retaining any data-derived class memory. Controlled diagnostics further show that lower feature displacement accompanies less forgetting, with a larger benefit from cumulative compression on longer streams. We validate HDCA only on CLIP-style vision-language models in this work, and leave its extension to other foundation-model architectures, including pretrained language models, as future work.

Acknowledgments and Disclosure of Funding

This work was partially supported by NSFC (U23A20382), the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118), and the Fundamental Research Funds for the Central Universities (2026300271).

References

Mina Ghadimi Atigh, Julian Schoep, Erman Acar, Nanne Van Noord, and Pascal Mettes. Hyperbolic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4453–4462, 2022.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, pages 446–461. Springer, 2014.

Karan Desai, Maximilian Nickel, Tanmay Rajpurohit, Justin Johnson, and Ramakrishna Vedantam. Hyperbolic image-text representations. In *International Conference on Machine Learning*, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3762–3773, 2020.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Zhi Gao, Yuwei Wu, Yunde Jia, and Mehrtash Harandi. Curvature generation in curved spaces for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8691–8700, 2021.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024.

Neil He, Hiren Madhu, Ngoc Bui, Menglin Yang, and Rex Ying. Hyperbolic deep learning for foundation models: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6021–6031, 2025.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8320–8329. IEEE Computer Society, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Linlan Huang, Xusheng Cao, Haori Lu, and Xialei Liu. Class-incremental learning with CLIP: Adaptive representation adjustment and parameter fusion. In *European Conference on Computer Vision*, pages 214–231. Springer, 2024.

Linlan Huang, Xusheng Cao, Haori Lu, Yifan Meng, Fei Yang, and Xialei Liu. Mind the gap: Preserving and compensating for the modality gap in CLIP-based continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3777–3786, 2025.

Sarah Ibrahimi, Mina Ghadimi Atigh, Nanne Van Noord, Pascal Mettes, and Marcel Worring. Intriguing properties of hyperbolic embeddings in vision-language models. *Transactions on Machine Learning Research*, 2024.

Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6428, 2020.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

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

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934. PMLR, 2019.

Yan-Shuo Liang and Wu-Jun Li. Adaptive plasticity improvement for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7816–7825, 2023.

Yan-Shuo Liang and Wu-Jun Li. InfLoRA: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024.

Shaoteng Liu, Jingjing Chen, Liangming Pan, Chong-Wah Ngo, Tat-Seng Chua, and Yu-Gang Jiang. Hyperbolic visual embedding learning for zero-shot recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9273–9281, 2020a.

Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020b.

Yue Lu, Shizhou Zhang, De Cheng, Yinghui Xing, Nannan Wang, Peng Wang, and Yanning Zhang. Visual prompt tuning in null space for continual learning. In *Advances in Neural Information Processing Systems*, 2024.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018.

Zelin Peng, Zhengqin Xu, Zhilin Zeng, Changsong Wen, Yu Huang, Menglin Yang, Feilong Tang, and Wei Shen. Understanding fine-tuning CLIP for open-vocabulary semantic segmentation in hyperbolic space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4562–4572, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5533–5542, 2017.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. CODA-Prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11909–11919, June 2023.

Qing Sun, Fan Lyu, Fanhua Shang, Wei Feng, and Liang Wan. Exploring example influence in continual learning. In *Advances in Neural Information Processing Systems*, 2022.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022b.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. SD-LoRA: Scalable decoupled low-rank adaptation for class incremental learning. In *International Conference on Learning Representations*, 2025.

Menglin Yang, Ram Samarth B B, Aosong Feng, Bo Xiong, Jihong Liu, Irwin King, and Rex Ying. Hyperbolic fine-tuning for large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *International Conference on Learning Representations*, 2023.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 2024a.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8363–8371, 2024b.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9851–9873, 2024c.

Da-Wei Zhou, Kai-Wen Li, Jingyi Ning, Han-Jia Ye, Lijun Zhang, and De-Chuan Zhan. External knowledge injection for CLIP-based class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025a.

Da-Wei Zhou, Yuanhan Zhang, Yan Wang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6):4489–4504, 2025b.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 2022.

Algorithm 1 HDCA training over a class-incremental stream

Require: Task stream $\{(\mathcal{D}_t, \mathcal{C}_t)\}_{t=1}^{N}$, frozen CLIP encoders f_v^0, f_t^0, selected projection set $\mathcal{S}$, scales K, R, ρ, σ.

for $t = 1, \ldots, N$ **do**
  Freeze f_v^0, f_t^0, and all previous adapter parameters $\{\Theta_k\}_{k<t}$.
  Add current trainable adapters $\Theta_t = \{A_t^\ell, B_t^\ell : \ell \in \mathcal{S}\}$ with $M_t^\ell = \frac{\alpha}{q} B_t^\ell A_t^\ell$.
  for each minibatch $\mathcal{B} \subset \mathcal{D}_t$ **do**
    for each selected projection $\ell \in \mathcal{S}$ in the image and text encoders **do**
      For every active adapter $k \le t$, compute its compressed output $\Phi_k^\ell(h) = \psi_{\sqrt{K}}\big(M_k^\ell h_{\mathrm{sp}}^H\big), \quad h^H = \Pi_{\exp}^K(h)$.
      Sum the outputs of all active adapters and apply cumulative compression $u_t^\ell(h) = \sum_{k=1}^{t} \Phi_k^\ell(h), \quad \Delta_t^\ell(h) = \psi_R\big(u_t^\ell(h)\big)$.
      Use the adapted projection output $W^\ell h + \Delta_t^\ell(h)$, with stop-gradient through Φ_k^ℓ for all $k < t$.
    end for
    Encode images and the class names of the current session, then normalize features $z_i^{(t)} = \mathrm{norm}\Big(f_v^{(t)}(x_i)\Big), \quad e_c^{(t)} = \mathrm{norm}\Big(f_t^{(t)}(\tau_c)\Big), \; c \in \mathcal{C}_t$.
    Compute Poincaré distance logits over the current classes $\ell_{i,c} = -\sigma\, d_H\Big(\rho z_i^{(t)}, \rho e_c^{(t)}\Big), \quad c \in \mathcal{C}_t$.
    Update only Θ_t by minimizing $\mathcal{L}_t = -\frac{1}{|\mathcal{B}|} \sum_{(x_i,y_i)\in\mathcal{B}} \log \frac{\exp(\ell_{i,y_i})}{\sum_{c\in\mathcal{C}_t} \exp(\ell_{i,c})}$.
  end for
  Retain Θ_t for future sessions and discard the recomputed text features.
end for

A Training Procedure

Algorithm 1 summarizes HDCA training with the main computations made explicit. The algorithm is written for one selected projection index ℓ but is applied to all selected query and value projections in both encoders. At inference after session t, the same forward computation is applied with the class names of all seen classes $\mathcal{C}_{\le t}$.

B Gradient Analysis of the Poincaré Classifier

This appendix supplies the gradient analysis omitted from Section 3.3, which justifies why the Poincaré logit reshapes the training signal in high-similarity regions while preserving cosine-based ranking at inference.

Let $s = \cos(z_v^{(t)}, e_c^{(t)})$. Because both projected points have norm ρ, their squared Euclidean distance reduces to

$$\|P(z_v^{(t)}) - P(e_c^{(t)})\|^2 \;=\; 2\rho^2(1-s). \tag{19}$$

Substituting Eq. (19) into Eq. (17) shows that d_H is a strictly decreasing function of s for any fixed $\rho \in (0,1)$, so the Poincaré logit and the cosine logit induce the same class ranking on the same adapted features.

Differentiating d_H with respect to s gives

$$\frac{\partial d_H}{\partial s} \;=\; -\frac{\beta}{\sqrt{(1+\beta(1-s))^2 - 1}}, \qquad \beta \;=\; \frac{4\rho^2}{(1-\rho^2)^2}, \tag{20}$$

whose magnitude grows sharply as $s \to 1$. As ρ approaches the boundary of the ball, β grows and small angular differences translate into larger hyperbolic distance changes, which is the mechanism that amplifies gradients between semantically close text features. To avoid numerical issues when s is close to 1, we clamp the argument of $\mathrm{arccosh}$ in Eq. (17) to remain strictly above 1.

C Displacement and Forgetting Metrics

This appendix gives the formal definitions behind Section 4.4 and reports the complete measurements. Let B be the number of sessions and $\mathrm{Acc}_{j,i}$ the accuracy on the test classes of task i after session j. Backward transfer is

$$\mathrm{BWT} = \frac{1}{B-1} \sum_{i=1}^{B-1} \big(\mathrm{Acc}_{B,i} - \mathrm{Acc}_{i,i}\big), \tag{21}$$

which weights earlier tasks equally. Old- and new-class accuracy are computed after the final session on the test images of classes from earlier sessions and from the final session, respectively.

For displacement, let $\hat{f}$ and $\hat{f}^0$ denote the final unit-normalized features of the adapted model and of the original CLIP. We report the cosine displacement $D_{\cos} = 1 - \cos(\hat{f}, \hat{f}^0)$ and the Euclidean displacement $D_2 = \|\hat{f} - \hat{f}^0\|_2$. Image-side values are averaged over the 6,000 ImageNet-R test images. Text-side values are computed on class prototypes, obtained by averaging the normalized embeddings of all prompt templates and normalizing the result, and are averaged over the 200 classes. The adapter output norm is the norm of the total adapter output added to a query or value projection, averaged over tokens within each forward pass and then equally over all such forward passes on the image or text side.

Table 5: Complete displacement measurements on ImageNet-R after the final session. Img and Txt denote the image and text sides; all variants use the Poincaré Classifier.

Stream	Per-Task Compression	Cumulative Compression	Adapter output norm Img	Adapter output norm Txt	$D_{\cos}$ Img	$D_{\cos}$ Txt	D_2 Img	D_2 Txt
10 sessions	✗	✗	41.73	47.93	0.73	0.66	1.20	1.15
	✓	✗	3.47	3.44	0.18	0.31	0.59	0.79
	✓	✓	**1.95**	**1.93**	**0.10**	**0.26**	**0.45**	**0.72**
20 sessions	✗	✗	42.39	60.52	0.79	0.44	1.26	0.94
	✓	✗	4.73	4.77	0.23	0.34	0.67	0.82
	✓	✓	**2.27**	**2.26**	**0.11**	**0.24**	**0.46**	**0.69**

Table 6: Accuracy and forgetting on ImageNet-R over 10 sessions (B0 Inc-20). Old and New denote the final accuracy on classes from earlier sessions and from the final session.

Per-Task Compression	Cumulative Compression	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	BWT	Old	New
✗	✗	38.24	9.13	−63.44	5.28	45.25
✓	✗	86.37	78.47	−9.38	77.70	85.66
✓	✓	**87.69**	**82.30**	**−6.31**	**81.77**	**87.22**

Table 5 shows that, within each stream, all three measures decrease from the variant without compression to Per-Task Compression and further to HDCA, on both the image and text sides. Per-Task Compression reduces the adapter output norm by roughly an order of magnitude, and Cumulative Compression roughly halves it again. Table 6 shows that Cumulative Compression improves both old- and new-class accuracy, so the reduction in forgetting does not come at the expense of learning new classes. Over 20 sessions, HDCA reaches an average accuracy of 87.32 and final old- and new-class accuracy of 80.40 / 83.03.

D Hyperparameter Sensitivity

We conduct a one-factor-at-a-time study on ImageNet-R B0 Inc-20 over all 10 sessions, varying one hyperparameter while keeping the others at their defaults: adapter rank $q = 20$, curvature parameter $K = 0.01$, cumulative compression scale $R = 1.0$, projection radius $\rho = 0.7$, and classifier scale $\sigma = 30$. Adapters are inserted into the query and value projections of all attention blocks. Table 7 reports the final accuracy $\mathcal{A}_B$.

Table 7: Sensitivity of HDCA to its hyperparameters on ImageNet-R B0 Inc-20. One hyperparameter is varied at a time; default values are shown in bold.

Hyperparameter	Values	$\mathcal{A}_B$
Adapter rank q	5 / 10 / **20** / 40	81.80 / 82.12 / **82.30** / 82.53
Curvature parameter K	0.005 / **0.01** / 0.02	82.83 / **82.30** / 81.42
Cumulative compression scale R	0.5 / **1.0** / 2.0	82.83 / **82.30** / 81.07
Projection radius ρ	0.5 / **0.7** / 0.9	81.93 / **82.30** / 82.23
Classifier scale σ	20 / **30** / 40	81.17 / **82.30** / 81.98

Across all tested configurations, $\mathcal{A}_B$ ranges from 81.07 to 82.83, compared with 82.30 at the default setting, and no value leads to an abrupt degradation. The defaults are not optimal for every hyperparameter; for example, $R = 0.5$ and $K = 0.005$ both reach 82.83. These results suggest that HDCA does not require delicate tuning within the tested ranges.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please refer to Section 3 and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Please refer to Section 3.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Please refer to Section 4 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code will release after acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to Section 4 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Please refer to Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Section 4 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have read the NeurIPS Code of Ethics and ensured that our research conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper is about continual learning and does not involve societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper is about continual learning and does not have a risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly cited all data, code, and models used in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: This research does not involve LLMs as any important, original, or nonstandard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.