Hierarchical Semantic Tree Anchoring for CLIP-based Class-Incremental Learning

Tao Hu[1,2] **Lan Li**[1,2] **Zhen-Hao Xie**[1,2] **Da-Wei Zhou**[1,2(✉)]
[1] School of Artificial Intelligence, Nanjing University
[2] State Key Laboratory for Novel Software Technology, Nanjing University
{hut, lil, wenzh, zhoudw}@lamda.nju.edu.cn

Abstract

Class-Incremental Learning (CIL) enables models to learn new classes continually while preserving past knowledge. Recently, vision-language models like CLIP offer transferable features via multimodal pre-training, making them well-suited for CIL. However, real-world visual and linguistic concepts are inherently hierarchical: a textual concept like "dog" subsumes fine-grained categories such as "Labrador" and "Golden Retriever," and each category is associated with corresponding images. Nevertheless, existing CLIP-based CIL methods fail to explicitly capture this inherent hierarchy, leading to feature drift of fine-grained classes during incremental updates and ultimately to catastrophic forgetting. To address this challenge, we propose HASTEN (Hierarchical Semantic Tree Anchoring), a hierarchy-aware framework that uses semantic structure to stabilize CLIP-based CIL. Rather than treating classes as isolated labels, HASTEN leverages external hierarchical knowledge as structured supervision to organize visual and textual features in hyperbolic space, helping maintain parent–child relations as new tasks arrive and mitigating feature drift. Since the shared hyperbolic mapper is updated across tasks, we further stabilize it by constraining its updates to a null space induced by prior-task features, reducing interference with previous mappings while retaining adaptability to new classes. Extensive experiments on nine benchmarks show that HASTEN consistently outperforms existing methods while reducing feature drift and catastrophic forgetting. Code is available at: https://github.com/LAMDA-CL/NeurIPS2026-HASTEN.

1 Introduction

Recent advancements in deep learning [28, 12, 38, 26, 24] have driven progress in vision and language tasks, but real-world applications face challenges with non-stationary streaming data, requiring adaptation to new data while retaining prior knowledge [1, 39]. Class-Incremental Learning (CIL) [58, 86, 47] enables incremental learning of new classes without forgetting earlier knowledge, yet suffers from catastrophic forgetting [20, 60, 19, 61], where new tasks interfere with old ones. Recent pre-trained models (PTMs) like CLIP [56] offer generalizable, transferable features from large-scale pre-training [92, 91, 30, 10], providing a strong foundation for continual learning and fast adaptation to new tasks via embedded prior knowledge.

Recent studies have attempted to address catastrophic forgetting [82, 14, 73, 37] by freezing the CLIP backbone to preserve general knowledge while adapting lightweight components (*e.g.*, adapters [30, 88] or prompts [76, 90]). Although these approaches mitigate forgetting to some extent, natural classes are inherently *hierarchical*. For example, "dog" subsumes "Labrador" and "Golden Retriever," and each fine-grained class has its own image set. However, current CLIP-based CIL methods often overlook this hierarchical signal because CLIP's Euclidean feature space is ill-suited to modeling hierarchy [13, 23]. Its uniform metric treats all points alike, so placing a generic concept

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

near many related items compresses distances and inadvertently pulls in unrelated specific concepts. As a result, broad concepts become too close to many specifics, blurring boundaries between hierarchical levels and weakening the representation of structure.

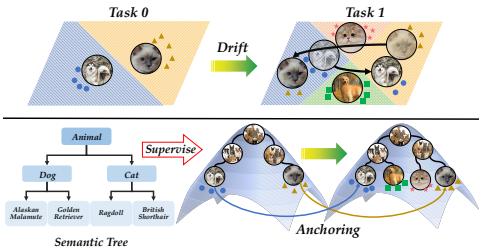


Figure 1: Effect of feature hierarchy. **Top:** Without hierarchy, fine-grained features tend to drift as new classes emerge. **Bottom:** With a hierarchical constraint, features are better anchored and relations are more stable during incremental learning.

This hierarchy is especially useful in incremental learning, where preserving relations among old and new classes can help mitigate catastrophic forgetting. As illustrated in Figure 1 (Top), without hierarchy, fine-grained class features (*e.g.*, specific dog and cat breeds) gradually drift [83] across incremental tasks. As new classes emerge, previous class features will shift from their original positions and even incorporate features of unrelated classes. In contrast, Figure 1 (Bottom) shows that with explicit hierarchical constraints, features can be anchored in a structured hyperbolic space, thereby suppressing drift during continual updates.

These observations motivate a stronger hierarchy-aware inductive bias. Without hierarchical constraints, early-class features lack effective protection and are gradually eroded by training on new classes. An ideal incremental learner should provide stable anchors for latent parent–child relations, keeping siblings compact yet separable and preventing broad concepts from being crowded by fine-grained ones. Meanwhile, its updates should respect the existing hierarchical organization while remaining plastic to new classes. Such a design can stabilize accumulated knowledge and mitigate catastrophic forgetting.

Based on this insight, we propose HASTEN (HierArchical Semantic TreE Anchoring), a hierarchy-aware framework that incorporates hierarchical semantic structure into CLIP-based continual learning. Concretely, HASTEN first uses an external hierarchical semantic tree as structured supervision to model text–text and image–text relations through two hierarchy-aware modules. The enhanced features are then mapped by a shared hyperbolic mapper into hyperbolic space, where hierarchical relations can be more naturally encoded and maintained across tasks. However, because this mapper bridges all tasks and modalities while being continuously updated, unconstrained updates may disturb previously learned mappings. We therefore constrain its updates to the null space induced by prior-task features, reducing interference with previous mappings while retaining adaptability to new classes. In this way, HASTEN maintains hierarchical structure while stabilizing the shared mapper, jointly mitigating feature drift and catastrophic forgetting throughout incremental learning.

2 Related Work

Class-Incremental Learning (CIL): CIL enables models to incrementally learn new classes without forgetting previously acquired knowledge [47, 11], and traditional approaches can be broadly categorized into several groups. Replay-based methods preserve past knowledge by storing and revisiting samples from previous tasks [58, 7, 8, 45], though this may come at the cost of memory usage and data privacy. Regularization-based methods estimate parameter importance to constrain updates [34, 85, 2, 3] or use knowledge distillation to preserve information at the logit [58, 42], feature [29, 44, 51], or group level [17, 22, 15]. Architecture-based methods mitigate forgetting through feature or module expansion [80, 71, 72, 87] and dynamically introduced task-specific tokens [18], balancing knowledge retention and adaptability at the cost of additional model complexity. Bias rectification methods address prediction and classifier biases [62, 86, 77].

CIL with Pre-Trained Models: The paradigm of CIL has been revitalized by powerful pre-trained models (PTMs) like Vision Transformers [16] and CLIP [56], which offer strong, generalizable representations [76]. A prevalent strategy is to freeze the PTM backbone and introduce lightweight, learnable modules for adaptation. For vision-centric PTMs, research has explored visual prompt learning [76, 64, 33, 40], adapter-based tuning [57, 9, 82, 55], and prototype-based classifiers that match features to class prototypes in embedding space [88, 48, 65]. Building on this, a significant trend leverages vision-language models like CLIP to exploit cross-modal alignment [41]. This is achieved through various means, including learning multimodal or textual prompts [75, 74, 92],

adapting the architecture with new projection heads as in PROOF [90], or using parameter-efficient adapters as in RAPF [30] to balance stability and plasticity. Recent studies have also explored leveraging external models to extract and incorporate fine-grained representations. ENGINE [89] extracts and aligns fine-grained features through a dual-branch architecture, while AREA [79] learns to aggregate external features without forgetting.

Hyperbolic Geometry for Hierarchical Representation: Hyperbolic geometry provides a natural inductive bias for hierarchical modeling, as its negative curvature and exponential volume growth enable embeddings of tree-like or partial-order structures with substantially lower distortion than in Euclidean spaces [25, 6]. In representation learning, a line of work has exploited hyperbolic manifolds (e.g., Poincaré ball and Lorentz model) to learn continuous hierarchies and taxonomies, showing strong advantages on structured data such as graphs and knowledge bases [49, 50, 21]. Beyond distance-based embeddings, order- and entailment-aware formulations further connect hierarchy to geometry by enforcing partial-order relations via specialized losses or cone-based constraints, enabling more explicit supervision of parent–child semantics [69, 21]. These ideas have been generalized to deep models, including hyperbolic neural networks and hyperbolic graph representation learning, and have been systematically reviewed in recent surveys [53]. More recently, hyperbolic representations have also been explored in vision-language modeling to better organize multimodal semantics under hierarchical or safety-aware structures [54].

3 Preliminaries

Class-Incremental Learning (CIL): CIL aims to build a classifier that learns new classes from sequential tasks without forgetting previously learned ones [58]. We consider a sequence of tasks $\{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^B\}$, where each task $\mathcal{D}^b = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_b}$ contains n_b instances. Here, $\mathbf{x}_i \in \mathbb{R}^D$ represents the feature vector, and $y_i \in Y_b$ is the class label for task b. The label sets across tasks are disjoint, *i.e.*, $Y_b \cap Y_{b'} = \varnothing$ for any $b \neq b'$. The exemplar-free CIL setting [93, 76] prohibits storing or replaying data from previous tasks, which means the model can only access the data from the current task $\mathcal{D}^b$ during training. The goal is to learn a unified model f that generalizes to all seen classes $\mathcal{Y}_b = Y_1 \cup \cdots \cup Y_b$ and minimizes the empirical risk:

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}}\, \mathbb{E}_{(\mathbf{x},y)\in \mathcal{D}^1 \cup \cdots \cup \mathcal{D}^b} \left[\mathbb{I}(y \neq f(\mathbf{x}))\right] , \tag{1}$$

where $\mathcal{H}$ is the hypothesis space, and $\mathbb{I}(\cdot)$ is the indicator function. In this paper, following recent work [90, 82, 30], we build our model $f(\mathbf{I})$ upon a pre-trained CLIP model [56]. CLIP consists of a visual encoder g_v and a text encoder g_t. For a given image $\mathbf{I}$, the visual embedding is $\mathbf{e}_v = g_v(\mathbf{I})$. For each class c, we construct a class-specific prompt $\mathbf{t}_c$ (e.g., "a photo of a $[\text{CLASS}]_c$"), and the textual embedding is $\mathbf{e}_t^c = g_t(\mathbf{t}_c)$. The image $\mathbf{I}$ is classified by comparing the cosine similarity between $\mathbf{e}_v$ and $\mathbf{e}_t^c$ of all seen classes $\mathcal{Y}_b$:

$$P_c(\mathbf{I}) = \frac{\exp\left(\cos(\mathbf{e}_v, \mathbf{e}_t^c)/\tau\right)}{\sum_{c' \in \mathcal{Y}_b} \exp\left(\cos(\mathbf{e}_v, \mathbf{e}_t^{c'})/\tau\right)}, \tag{2}$$

where τ is a temperature parameter that controls the softness of the distribution.

Adapting CLIP for CIL: A common approach in CLIP-based CIL is to freeze the pre-trained visual and text encoders (g_v and g_t) and insert a lightweight adaptation module h_v after the visual encoder [30, 90]. In this setup, the adapted image feature is represented as:

$$\mathbf{z}_v = h_v(\mathbf{e}_v) , \tag{3}$$

where $\mathbf{e}_v = g_v(\mathbf{I})$ is the frozen visual embedding. This adapted feature is then substituted for $\mathbf{e}_v$ in Eq. 2 to compute classification probabilities, which are subsequently used to calculate the contrastive loss during training. This strategy allows the model to learn downstream knowledge while mitigating catastrophic forgetting effects by restricting the number of trainable parameters.

Discussion: Although Eq. 3 mitigates forgetting via lightweight tuning, the adapter-only tuning approach is intrinsically fragile under sequential updates. Continual revisions across tasks will induce class-feature drift [83], which progressively erodes the text–image correspondence and breaks the cross-modal alignment required by Eq. 2. This accumulation of unreliable logits and visual–textual mismatches ultimately results in catastrophic forgetting. To counteract this, an effective CLIP adaptation must explicitly encode an ordered hierarchical structure to organize and anchor visual and textual features across all tasks, thus curbing drift and preserving crucial cross-modal alignment.

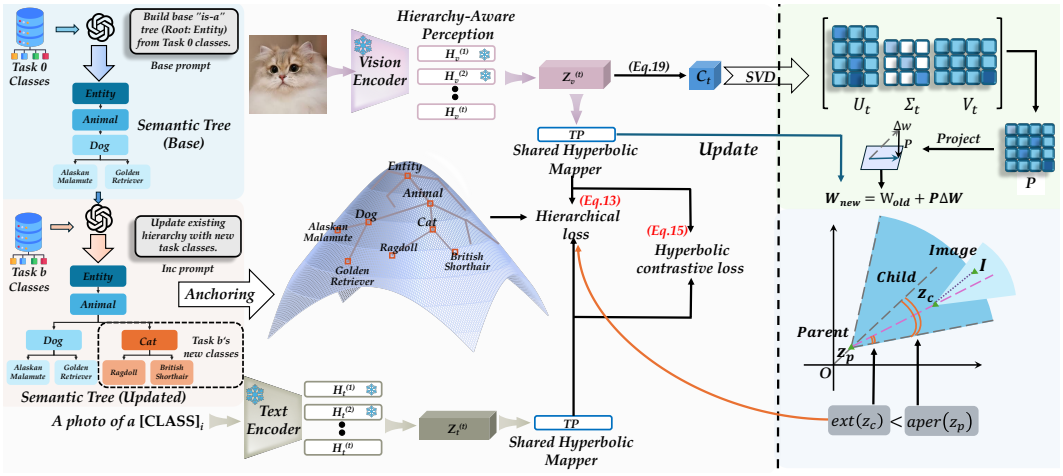


Figure 2: Illustration of HASTEN. **Left**: Hierarchical semantic tree building and hyperbolic projection. We use GPT-5 to generate a tree-structured semantic hierarchy and design task-specific hierarchical perception modules to meet downstream knowledge requirements. Euclidean features are projected into hyperbolic space via a global hyperbolic mapping layer. **Top-Right**: Null space projection of the TP layer, ensuring TP does not interfere with the outputs of old tasks during updates. **Bottom-Right**: Illustration of the entailment loss in $\mathcal{B}^2$. This loss pushes the embedding $\mathbf{z}_c$ within an imaginary cone projected by its paired parent embedding $\mathbf{z}_p$.

4 HASTEN: Hierarchical Semantic Tree Anchoring

Motivated by feature drift in CLIP-based CIL, we propose HASTEN, a hierarchy-aware framework that uses semantic structure to organize visual and textual features for mitigating catastrophic forgetting. HASTEN first builds an externally generated hierarchical semantic tree as structured supervision, allowing classes to be modeled through parent-child relations rather than isolated labels. The hierarchy-aware perception modules and shared hyperbolic mapper then embed visual and textual features into hyperbolic space to maintain hierarchical relations across tasks. To stabilize this shared mapper, we project its gradients onto an approximate null space induced by prior-task features, reducing interference with previous mappings while preserving adaptability to new classes.

4.1 Hierarchical Semantic Tree Construction

In the real world, visual and linguistic concepts naturally form hierarchies, where broad concepts subsume fine-grained subclasses and images correspond to leaf-level instances. To model this structure under the CIL protocol, we construct a dataset-conditioned semantic tree in an incremental manner, using only the class names available at each stage and without accessing future classes.

Base Tree Initialization ($t = 0$)**:** At the initial stage, we use GPT-5 to build an *is-a* hierarchy from the base classes, with a virtual root "entity". GPT-5 may introduce abstract virtual classes to connect dataset-native real classes into a coherent tree. For example:

> **Q**: Build a complete *is-a* hierarchy with a single root "entity" from the base class list. You may introduce abstract parent classes if needed. Input list: ["ragdoll", "british shorthair"]
> **A**: {"entity": ["animal"],
> "animal": ["cat"],
> "cat": ["ragdoll", "british shorthair"]}

Incremental Tree Expansion ($t > 0$)**:** For later stages, we expand the existing hierarchy by appending only the newly observed classes. Existing parent-child edges are kept fixed, and new virtual parents are added only when necessary to connect new classes to existing branches.

> **Q**: Given an existing taxonomy tree, integrate the new classes without removing or altering existing nodes. Existing tree: {"entity": ["animal"], "animal": ["cat"], "cat": ["ragdoll", "british shorthair"]}. New classes: ["golden retriever", "alaskan

malamute"]
A: {"entity": ["animal"],
"animal": ["cat", "dog"],
"cat": ["ragdoll", "british shorthair"],
"dog": ["golden retriever", "alaskan malamute"]}

Through this process, we obtain a model-agnostic hierarchical supervision signal at each learning stage. We denote the current semantic tree as $\mathcal{K} = (\mathcal{C}, \mathcal{E}, r)$, where $\mathcal{C} = \mathcal{C}_{\text{real}} \cup \mathcal{C}_{\text{virt}}$ contains real and virtual classes, $r =$ "entity" is the root, and $\mathcal{E} \subseteq \mathcal{C} \times \mathcal{C}$ contains directed parent→child edges. This tree defines the text-to-text hierarchy used for subsequent hierarchy-aware feature learning.

4.2 Hierarchy-Aware Anchoring

With the GPT-generated hierarchical semantic tree in hand, we leverage it during training so that classes explicitly inherit parent–child structure. This choice aligns with hyperbolic geometry. Theoretically, hyperbolic space can embed tree-structured data with minimal distortion [59, 13], making it a natural fit for representing and preserving hierarchical semantics throughout learning.

Hyperbolic Geometry Setup: Concretely, we adopt the Lorentz model $\mathcal{B}^d$ (embedded in $\mathbb{R}^{d+1}$) to instantiate our hyperbolic representation, where d matches the Euclidean embedding dimension. A point $\mathbf{p} \in \mathcal{B}^d$ is parameterized as $[\mathbf{p}_s, p_t]$, where $p_t = \sqrt{\frac{1}{c} + \|\mathbf{p}_s\|^2}$. The resulting geodesic distance $d_{\mathcal{B}}(\mathbf{p}, \mathbf{q})$ in $\mathcal{B}^d$ is:

$$d_{\mathcal{B}}(\mathbf{p}, \mathbf{q}) = \frac{1}{\sqrt{c}} \cosh^{-1}\big(-c(\langle \mathbf{p}_s, \mathbf{q}_s \rangle - p_t q_t)\big). \tag{4}$$

Here, $c > 0$ denotes the curvature, which is set to 1 for stability. Negative curvature provides exponential volume growth and a depth–radius correspondence, naturally matching tree-structured data; a detailed justification is given in Appendix A.2. In contrast, Euclidean volume grows only polynomially and may cause crowding near generic concepts. In our model, $d_{\mathcal{B}}$ serves as the task metric for similarity, margins, and hierarchy constraints. We use the spatial component of the exponential map at the origin $\mathbf{o} = [\mathbf{0}, 1]$ to map a Euclidean tangent vector $\mathbf{v}$:

$$\exp\mathrm{m}_{\mathbf{o}}(\mathbf{v}) = \frac{\sinh(\|\mathbf{v}\|)}{\|\mathbf{v}\|}\mathbf{v}. \tag{5}$$

These components provide the necessary hyperbolic similarities and geodesics that respect hierarchical geometry, which we utilize for subsequent alignment and drift control.

Hierarchy-Aware Perception and Aggregation: To encode hierarchical information while retaining CLIP's generalization power, we introduce two *task-specific* linear hierarchy-aware modules, H_v^b (for visual features) and H_t^b (for textual features), immediately following the frozen CLIP encoders g_v and g_t for each incremental task b. The modules map from $\mathbb{R}^d$ to $\mathbb{R}^d$. Given a text $\mathbf{t}$ and an image $\mathbf{I}$, the features in Euclidean space are $\mathbf{e}_t$ and $\mathbf{e}_v$. During task b, only the new modules $\{H_v^b, H_t^b\}$ are trained, while all earlier modules $\{H_v^1, \ldots, H_v^{b-1}\}$ and $\{H_t^1, \ldots, H_t^{b-1}\}$ are frozen to preserve prior knowledge. We cumulatively aggregate the outputs of all modules to obtain the task-aware Euclidean features before mapping to the hyperbolic space:

$$\tilde{\mathbf{z}}_v = \sum_{i=1}^{b} H_v^i(\mathbf{e}_v), \quad \tilde{\mathbf{z}}_t = \sum_{i=1}^{b} H_t^i(\mathbf{e}_t), \tag{6}$$

This placement and aggregation preserves old-task information through frozen components while enabling the new module to specialize for the current task b.

Global Hyperbolic Mapping Layer: To map all aggregated visual and textual features from Euclidean space onto a shared hyperbolic manifold, we utilize a task-shared linear layer, $\mathrm{TP}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$. It maps an aggregated Euclidean feature $\tilde{\mathbf{z}}$ (*e.g.*, $\tilde{\mathbf{z}}_v$ or $\tilde{\mathbf{z}}_t$) to the spatial component $\mathbf{z}_s$ of the Lorentz model $\mathcal{B}^d$ via $\mathbf{z}_s = \mathrm{TP}(\tilde{\mathbf{z}})$. The resulting hyperbolic representations used for training are:

$$\mathbf{z}_v = \mathrm{TP}\left(\tilde{\mathbf{z}}_v\right), \quad \mathbf{z}_t = \mathrm{TP}(\tilde{\mathbf{z}}_t). \tag{7}$$

For notational simplicity, $\mathbf{z}_v$ and $\mathbf{z}_t$ also denote the corresponding full Lorentz points. To balance projection fidelity and adaptation flexibility, we regularize TP toward the exponential map at the origin:

$$\mathcal{L}_{\text{tp}} = \left\|\mathrm{TP}(\tilde{\mathbf{z}}) - \exp\mathrm{m}_{\mathbf{o}}(\tilde{\mathbf{z}})\right\|_2^2. \tag{8}$$

Overall, task-shared TP maps aggregated module outputs into a common hyperbolic space, producing text ($\mathbf{z}_t$) and image ($\mathbf{z}_v$) features for hierarchy-aware alignment and stable incremental training.

Entailment Loss for Partial-Order Supervision: We leverage the semantic tree $\mathcal{K} = (\mathcal{C}, \mathcal{E}, r)$ to encode its partial order in hyperbolic space: for any edge $(p, c) \in \mathcal{E}$, the child embedding must lie inside a parent-centered cone of the parent. We use the hyperbolic text embedding $\mathbf{z}_t^u$ for any node $u \in \mathcal{C}$ and the hyperbolic visual embedding $\mathbf{z}_v$ for an image $\mathbf{I}$. The cone half-aperture for a hyperbolic point $\mathbf{p}$ is defined as: $\mathrm{aper}(\mathbf{p}) = \sin^{-1}\Big(\frac{2\kappa}{\sqrt{c}\,\|\mathbf{p}\|}\Big)$, where $\kappa = 0.1$ unless noted. Deeper parents (larger $\|\mathbf{p}\|$) induce tighter cones, which aligns with increasing semantic specificity. The exterior angle $\mathrm{ext}(\mathbf{p}, \mathbf{q})$ measures the child $\mathbf{q}$'s deviation from the cone boundary centered at parent $\mathbf{p}$:

$$\mathrm{ext}(\mathbf{p}, \mathbf{q}) = \cos^{-1}\left(\frac{q_t + p_t\,(c\langle \mathbf{p}, \mathbf{q}\rangle_{\mathcal{B}})}{\|\mathbf{p}_s\|\sqrt{(c\langle \mathbf{p}, \mathbf{q}\rangle_{\mathcal{B}})^2 - 1}}\right), \tag{9}$$

where $\mathbf{p}, \mathbf{q} \in \mathcal{B}^d$. To ensure that the learned hyperbolic space represents the semantic hierarchy, we impose the geometric constraint that each child node must lie within an aperture cone centered at its parent, formalized as $\mathrm{ext}(\mathbf{p}, \mathbf{q}) \leq \mathrm{aper}(\mathbf{p})$. As visualized in Figure 2 (Bottom-Right), this constraint encourages child nodes to lie in parent-centered descendant regions, while the separation term in Eq. 11 further promotes a radial hierarchy between parents and children. We penalize violations using the entailment loss:

$$\mathcal{L}_{\mathrm{entail}}(\mathbf{p}, \mathbf{q}) = \mathrm{SP}\big(\mathrm{ext}(\mathbf{p}, \mathbf{q}) - \mathrm{aper}(\mathbf{p})\big), \tag{10}$$

with the soft-plus operation $\mathrm{SP}(z) = \log(1 + \mathrm{e}^z)$. In practice, we apply $\mathcal{L}_{\mathrm{entail}}$ to: (1) Text–Text pairs $(\mathbf{z}_t^p, \mathbf{z}_t^c)$ for $(p, c) \in \mathcal{E}$ and (2) Text–Image pairs $(\mathbf{z}_t^c, \mathbf{z}_v)$, effectively pulling all children into their respective parents' cones and instantiating $\mathcal{K}$'s hierarchical relation in $\mathcal{B}^d$.

Hierarchical Supervision: Building on the entailment cones, we supervise both the text–text hierarchy and text–image anchoring with two complementary terms. For a parent-child edge $(p, c) \in \mathcal{E}$, the text–text hierarchy term is:

$$\mathcal{L}_{\mathrm{hier}}^{\mathrm{txt}\to\mathrm{txt}} = \mathcal{L}_{\mathrm{entail}}(\mathbf{z}_t^p, \mathbf{z}_t^c) + \mathrm{SP}\left(\delta - d_{\mathcal{B}}(\mathbf{z}_t^p, \mathbf{z}_t^c)\right). \tag{11}$$

In Eq. 11, the second term enforces a minimum parent–child separation δ to avoid embedding collapse. The text–image hierarchy term aligns the visual embedding $\mathbf{z}_v$ with its corresponding class text node $\mathbf{z}_t^c$:

$$\mathcal{L}_{\mathrm{hier}}^{\mathrm{txt}\to\mathrm{img}} = \mathcal{L}_{\mathrm{entail}}\big(\mathbf{z}_t^c,\, \mathbf{z}_v\big). \tag{12}$$

We combine the two terms in Eqs. 11 and 12 to form the hierarchical supervision objective:

$$\mathcal{L}_{\mathrm{hier}} = \lambda_1 \mathcal{L}_{\mathrm{hier}}^{\mathrm{txt}\to\mathrm{txt}} + \lambda_2 \mathcal{L}_{\mathrm{hier}}^{\mathrm{txt}\to\mathrm{img}}. \tag{13}$$

Eq. 13 stabilizes features by anchoring children within parent-centered cones, ensuring a hierarchy-respecting geometry across tasks with sufficient separation.

Hyperbolic Contrastive Loss: Complementing the cone-based hierarchical supervision, we further tighten cross-modal alignment with a hyperbolic contrastive loss. Using the negative hyperbolic distance from Eq. 4 as the similarity metric, we adopt the standard symmetric contrastive loss framework (similar to the multi-class N-pair loss [66, 56]).

Specifically, for the j-th positive image-text pair $(\mathbf{I}_j, \mathbf{t}_j)$ in the current batch, with hyperbolic embeddings $\mathbf{z}_{v,j}$ and $\mathbf{z}_{t,j}$, the match probabilities used in the contrastive loss are:

$$P_j^{\mathrm{I}} = \frac{\exp(-d_{\mathcal{B}}(\mathbf{z}_{v,j}, \mathbf{z}_{t,j})/\tau)}{\sum_{i=1}^{N} \exp(-d_{\mathcal{B}}(\mathbf{z}_{v,j}, \mathbf{z}_{t,i})/\tau)}, \quad P_j^{\mathrm{T}} = \frac{\exp(-d_{\mathcal{B}}(\mathbf{z}_{t,j}, \mathbf{z}_{v,j})/\tau)}{\sum_{i=1}^{N} \exp(-d_{\mathcal{B}}(\mathbf{z}_{t,j}, \mathbf{z}_{v,i})/\tau)}. \tag{14}$$

The hyperbolic contrastive loss $\mathcal{L}_{\mathrm{contrast}}$ is defined as:

$$\mathcal{L}_{\mathrm{contrast}} = -\frac{1}{2N} \sum_{j=1}^{N} \left(\log P_j^I + \log P_j^T\right). \tag{15}$$

Eq. 15 pulls matched pairs together and separates mismatches in $\mathcal{B}^d$, thereby stabilizing cross-task features and reinforcing cross-modal clusters.

Drift Mitigation across Tasks: The hierarchical supervision in Eq. 13 serves as the anchoring objective that preserves the semantic tree structure in the hyperbolic space and helps resist feature drift. To keep these hierarchical anchors stable across incremental stages, after each task we cache and freeze the hyperbolic text embeddings of all virtual nodes. In the next task, we only instantiate missing ancestors, reuse the cached virtual nodes, and compute the textual hierarchy loss over the resulting union, so that new classes are attached to existing parents without shifting previous anchors. Meanwhile, to maintain old class regions without storing exemplars, we follow [93] to draw class-level features from $\mathcal{N}(\mu_k, \Sigma_k)$, where μ_k and Σ_k are derived from the frozen encoder g_v. These surrogates provide old-class references in the hyperbolic space and help reduce cross-modal feature drift during training.

4.3 Stabilizing the Shared TP

Since TP is shared across tasks and modalities, its updates may shift previously learned visual–textual mappings in the hyperbolic space. To mitigate forgetting, we restrict them to directions with limited response on past pre-TP features. At the beginning of task b, we compute the projection subspace from the accumulated covariance $C^{(b-1)}$ of previous-task pre-TP features:

$$C^{(b-1)} = U\,\Sigma\,V^\top, \quad \Sigma = \mathrm{diag}(\sigma_1 \geq \cdots \geq \sigma_d). \tag{16}$$

Let r be the smallest index whose cumulative energy $\sum_{k=1}^{r}\sigma_k^2 / \sum_{k=1}^{d}\sigma_k^2$ exceeds a preset threshold; this yields a data-adaptive approximation to the null space rather than an exact one. We define the orthogonal complement as $V_\perp = [\mathbf{v}_{r+1}, \ldots, \mathbf{v}_d]$ and project the raw TP gradient ΔW onto this space:

$$\Delta W_{\mathrm{proj}} = V_\perp V_\perp^\top \Delta W. \tag{17}$$

Since $C^{(b-1)} \propto X^\top X$ for past features X, the columns of $V_\perp$ span directions of near-zero variance. Thus $X^o V_\perp \approx 0$ for old-task features X^o, leading to:

$$X^o \Delta W_{\mathrm{proj}} = X^o V_\perp V_\perp^\top \Delta W = (X^o V_\perp)(V_\perp^\top \Delta W) \approx 0, \tag{18}$$

which suggests that the projected update has limited response on old-task features and reduces the drift of previous TP mappings while still enabling learning in new directions.

After completing task b, we update the covariance used for subsequent tasks:

$$C^{(b)} = \frac{\alpha_{b-1}\,C^{(b-1)} + N_b\,C_{\mathrm{new}}^{(b)}}{\alpha_b}, \quad \alpha_i = \sum_{j=1}^{i} N_j, \tag{19}$$

where N_j is the number of samples in task j, and $C_{\mathrm{new}}^{(b)}$ denotes the uncentered second-moment matrix of the current-task pre-TP features.

Discussion: The purpose of this step is to stabilize the shared TP across tasks and modalities. Projecting gradients onto the approximate null space induced by past pre-TP features reduces interference from previous tasks while keeping TP plastic for new ones. Specifically, under an idealized gradient descent step, the parameter update becomes $\theta_{\mathrm{TP}} \leftarrow \theta_{\mathrm{TP}} - \eta(V_\perp V_\perp^\top \Delta W)$. The energy threshold that selects r tunes the stability–plasticity trade-off, where larger r protects more high-variance old-task directions while smaller r allows faster adaptation. We update $C^{(b)}$ from pre-TP Euclidean features after each task and refresh $V_\perp$ for subsequent tasks to control cost. In this way, the projection limits disturbance to previously mapped regions while allowing TP to learn new mappings.

Summary of HASTEN: In HASTEN, we leverage hierarchical knowledge for continual learning via a semantic tree. We design hierarchy-aware perception with hyperbolic mapping and stabilize the shared mapper via projected updates. The training objective is defined by combining Eqs. 13, 15, and 8:

$$\mathcal{L} = \mathcal{L}_{\mathrm{hier}} + \mathcal{L}_{\mathrm{contrast}} + \beta\mathcal{L}_{\mathrm{tp}}\,. \tag{20}$$

During inference, we enhance robustness by ensembling predictions from two distinct feature spaces. The first component, the original space probability $P_{\mathrm{orig},k}(\mathbf{x})$, is computed via cosine similarity in the original Euclidean space between the image feature $\mathbf{e}_v$ and class centers μ_k. The second, the hyperbolic space probability $P_{\mathrm{hyp},k}(\mathbf{x})$, leverages the learned hierarchy, using the negative hyperbolic distance $-d_{\mathcal{B}}(\mathbf{z}_v, \mathbf{z}_t^c)$ between the hierarchy-aware image feature $\mathbf{z}_v$ and the hyperbolic text prototype $\mathbf{z}_t^c$ as the logit. These two distributions are then fused to form the final class probability:

$$P_k(\mathbf{x}) = P_{\mathrm{orig},k}(\mathbf{x}) + P_{\mathrm{hyp},k}(\mathbf{x})\,. \tag{21}$$

The final predicted class $\hat{y}$ is determined by selecting the class with the maximum fused probability: $\hat{y} = \arg\max_k P_k(\mathbf{x})$.

5 Experiments

In this section, we evaluate HASTEN on nine benchmarks against state-of-the-art methods. Furthermore, we provide comprehensive analyses, including ablation studies on hierarchical supervision, key hyperparameters, and t-SNE visualizations comparing embeddings with and without preserving hierarchical structure. Finally, we visualize image-to-root geodesic traversals to explicitly illustrate how the hyperbolic representation preserves hierarchical semantics from specific to generic concepts.

Table 1: Average and last performance comparison of different methods. The best performance is shown in bold. **All methods are initialized with the same pre-trained CLIP for a fair comparison.**

Method	Aircraft				CIFAR100				Cars			
	B0 Inc10		B50 Inc10		B0 Inc10		B50 Inc10		B0 Inc10		B50 Inc10	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	3.16	0.96	1.72	1.05	7.84	4.44	5.30	2.46	3.14	1.10	1.54	1.13
SimpleCIL [88]	59.24	48.09	53.05	48.09	84.15	76.63	80.20	76.63	92.04	86.85	88.96	86.85
ZS-CLIP [56]	26.66	17.22	21.70	17.22	81.81	71.38	76.49	71.38	82.60	76.37	78.32	76.37
L2P [76]	47.19	28.29	44.07	32.13	82.74	73.03	81.14	73.61	76.63	61.82	76.37	65.64
DualPrompt [75]	44.30	25.83	46.07	33.57	81.63	72.44	80.12	72.57	76.26	62.94	76.88	67.55
CODA-Prompt [64]	45.98	27.69	45.14	32.28	82.43	73.43	78.69	71.58	80.21	66.47	75.06	64.19
PROOF [90]	63.81	56.14	59.47	57.10	86.77	79.11	83.32	79.73	90.74	86.51	88.00	85.58
RAPF [30]	50.38	23.61	40.47	25.44	86.14	78.04	82.17	77.93	82.89	62.85	75.87	63.19
CLG-CBM [84]	66.05	55.93	59.26	55.39	81.82	74.57	68.61	65.55	78.94	74.26	80.11	73.62
HASTEN	**67.23**	**56.32**	**61.47**	**57.19**	**88.64**	**82.51**	**85.45**	**82.38**	**93.06**	**88.93**	**90.23**	**88.46**

Method	ImageNet-R				CUB				UCF			
	B0 Inc20		B100 Inc20		B0 Inc20		B100 Inc20		B0 Inc10		B50 Inc10	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	1.37	0.43	1.01	0.88	2.06	0.64	0.56	0.47	4.51	1.59	1.21	0.80
SimpleCIL [88]	81.06	74.48	76.84	74.48	83.81	77.52	79.75	77.52	90.44	85.68	88.12	85.68
ZS-CLIP [56]	83.37	77.17	79.57	77.17	74.38	63.06	67.96	63.06	75.50	67.64	71.44	67.64
L2P [76]	75.97	66.52	72.82	66.77	70.87	57.93	75.64	66.12	86.34	76.43	83.95	76.62
DualPrompt [75]	76.21	66.65	73.22	67.58	69.89	57.46	74.40	64.84	85.21	75.82	84.31	76.35
CODA-Prompt [64]	77.69	68.95	73.71	68.05	73.12	62.98	73.95	62.21	87.76	80.14	83.04	75.03
PROOF [90]	83.84	78.40	81.20	78.92	82.31	76.64	79.20	76.37	94.58	91.10	93.58	90.91
RAPF [30]	81.26	70.48	76.10	70.23	79.09	62.77	72.82	62.93	92.28	80.33	90.31	81.55
CLG-CBM [84]	79.19	72.03	75.20	70.50	78.11	72.14	78.17	73.03	95.04	91.36	94.17	91.85
HASTEN	**85.73**	**80.70**	**82.58**	**80.53**	**86.22**	**80.96**	**82.86**	**80.20**	**96.49**	**92.69**	**95.55**	**92.91**

Method	SUN				Food				ObjectNet			
	B0 Inc30		B150 Inc30		B0 Inc10		B50 Inc10		B0 Inc20		B100 Inc20	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	4.51	1.59	0.78	0.72	3.49	1.71	2.14	1.52	1.34	0.47	0.69	0.54
SimpleCIL [88]	82.13	75.58	78.62	75.58	87.89	81.65	84.73	81.65	52.06	40.13	45.11	40.13
ZS-CLIP [56]	79.42	72.11	74.95	72.11	87.86	81.92	84.75	81.92	38.43	26.43	31.12	26.43
L2P [76]	82.82	74.54	79.57	73.10	85.66	77.33	80.42	73.13	51.40	39.39	48.91	42.83
DualPrompt [75]	82.46	74.40	79.37	73.02	84.92	77.29	80.00	72.75	52.62	40.72	49.08	42.92
CODA-Prompt [64]	83.34	75.71	80.38	74.17	86.18	78.78	80.98	74.13	46.49	34.13	40.57	34.13
PROOF [90]	83.89	77.25	80.15	76.54	90.04	84.73	87.52	84.74	56.07	43.69	48.90	43.62
RAPF [30]	82.13	72.47	78.04	73.10	88.57	81.15	85.53	81.17	48.67	27.43	39.28	28.73
CLG-CBM [84]	84.85	78.09	81.58	77.79	89.12	83.05	86.76	83.85	58.52	45.11	49.80	43.56
HASTEN	**86.30**	**80.40**	**82.99**	**80.25**	**90.67**	**85.70**	**88.18**	**85.75**	**58.63**	**46.18**	**52.03**	**45.88**

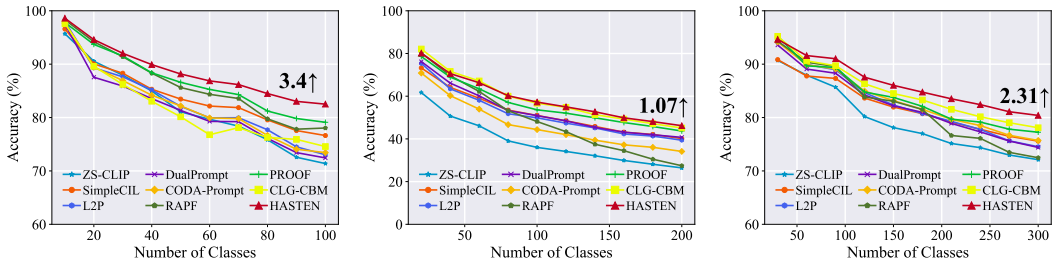
5.1 Implementation Details

Dataset: We adopt the evaluation protocol from [89, 90, 92] and evaluate on nine benchmarks that exhibit domain shifts from CLIP's pre-training data: CIFAR100 [36], CUB200 [70], ObjectNet [4], ImageNet-R [27], FGVCAircraft [46], StanfordCars [35], Food101 [5], SUN397 [78], and UCF101 [67]. Following [90], we use sampled subsets for practical partitioning: 100 classes each from CIFAR100, FGVCAircraft, StanfordCars, Food101, and UCF101; 200 classes each from CUB200, ObjectNet, and ImageNet-R; and 300 classes from SUN397.

Dataset split: Following [58, 89, 90], we use 'B-m Inc-n' for CIL class splitting: m denotes the number of classes in the first stage, and n the number of classes in each subsequent stage. Consistent with [58], we randomly shuffle the class order with seed 1993, which is kept consistent across all methods.

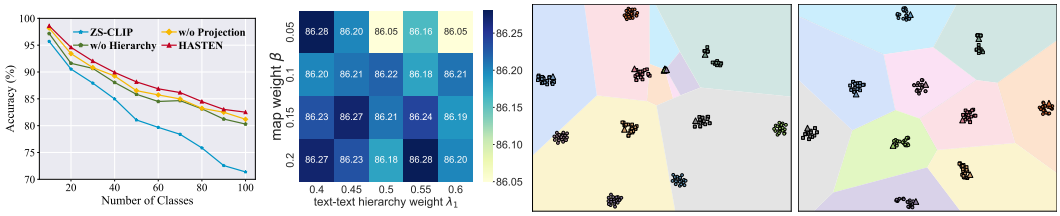
Comparison methods: We compare our method with SOTA pre-trained model-based CIL algorithms, such as L2P [76], DualPrompt [75], CODA-Prompt [64], and SimpleCIL [88]. We also include SOTA CLIP-based CIL approaches, including PROOF [90], RAPF [30], and CLG-CBM [84]. The baseline of finetuning CLIP for incremental tasks is denoted as Finetune. All methods are initialized with the same CLIP model for a fair comparison.

Training details: All experiments are run on NVIDIA A100 GPUs with PyTorch [52]. Following [90, 30], we evaluate CLIP ViT-B/16 pre-trained on LAION-400M [31] across all methods to ensure a fair comparison. For vision-only methods that cannot use textual prompts, we initialize them with CLIP's visual encoder. In HASTEN, we employ the AdamW [43] optimizer with a batch size of 64 to train the model for 10 epochs. The learning rate starts at 0.001 and follows cosine annealing. We set $\lambda_1 = 0.5$ and $\lambda_2 = 0.1$ for the text–text and text–image hierarchical losses, $\beta = 0.1$ for the TP regularizer, and $\delta = 0.2$. For the fixed class order determined by seed 1993, the stage-wise hierarchical semantic trees are incrementally generated using GPT-5 [63] before training, and the full prompts are provided in the supplementary.



(a) CIFAR100 Base0 Inc10 (b) ObjectNet Base0 Inc20 (c) SUN Base0 Inc30

Figure 3: Incremental performance of different methods. We report the performance gap after the last incremental stage of HASTEN and the runner-up method at the end of the line. All methods use the same pre-trained CLIP weights. More results are in the supplementary.



(a) Ablation study (b) Parameter sensitivity (c) w/o preserving hierarchy (d) w/ preserving hierarchy

Figure 4: **(a)** Ablation study and **(b)** parameter sensitivity analysis. **(c)(d)** t-SNE [68] visualizations of visual and textual features on CIFAR100 B0 Inc5. We show the feature distributions of old classes and new classes without (c) and with (d) hierarchy.

Evaluation metric: Following [58, 90], we denote the model's top-1 accuracy after the b-th stage by $\mathcal{A}_b$. We report $\mathcal{A}_B$ as the final-stage accuracy and $\bar{\mathcal{A}} = \frac{1}{B}\sum_{b=1}^{B}\mathcal{A}_b$ as the mean accuracy across incremental stages.

5.2 Benchmark Comparison

We compare HASTEN with state-of-the-art methods on benchmark datasets, with representative results reported in Table 1 and Figure 3, and complete results on all nine datasets provided in Appendix F. As shown, HASTEN consistently achieves state-of-the-art performance across the evaluated settings, demonstrating robust generalization and broad applicability. Finetuning performs the worst, indicating severe catastrophic forgetting. Visual prompt methods fall behind as they fail to leverage textual information. Compared to strong CLIP-based methods such as RAPF and PROOF, HASTEN's gains highlight its anti-forgetting capability while retaining the advantages of cross-modal representations.

5.3 Further Analysis

Ablation study: We conduct ablations on CIFAR100 B0 Inc10 in Figure 4a. **HASTEN** denotes the full model. **w/o Hierarchy** removes the hierarchical supervision. **w/o Projection** removes the TP null-space projection. **ZS-CLIP** is the zero-shot baseline and performs worse due to the downstream pre-training distribution gap. The curves show that HASTEN stays highest across all stages; the gap over baselines widens as the number of classes grows. Removing hierarchy yields a larger and persistent drop, indicating increased drift without hierarchical supervision. Removing projection incurs a smaller early drop but hurts later stages as interference accumulates. These trends confirm that hierarchical supervision and TP stabilization provide complementary benefits for strong incremental performance.

Parameter robustness: We evaluate the sensitivity to two hyperparameters, including the text–text hierarchical weight λ_1 in Eq. 13 and the hyperbolic mapping weight β in Eq. 20. On the CUB B0 Inc20 setting, we sweep λ_1 among $\{0.40, 0.45, 0.50, 0.55, 0.60\}$ and β among $\{0.05, 0.10, 0.15, 0.20\}$. Figure 4b reports the average validation accuracy, showing that HASTEN maintains stable performance across a broad range of hyperparameter choices. Additional sensitivity results and details are provided in Appendix D.



Ours	CLIP
Tai Chi	*Tai Chi*
martial_art	↓
a practitioner flowing through slow, mindful tai chi forms in a park.	↓
a martial arts drill emphasizing timing, guard, and footwork.	↓
"entity"	[ROOT]



Ours	CLIP
Parallel Bars	*Parallel Bars*
a gymnast supporting the body and swinging between the parallel bars.	↓
a gymnast balancing inverted and walking on hands across a mat.	↓
"entity"	[ROOT]

Figure 5: **Image traversals with HASTEN and CLIP.** HASTEN's path through hyperbolic space reveals a smooth transition from specific to generic text descriptions, demonstrating a learned hierarchy. In contrast, CLIP's flat representation typically yields fewer descriptions along the path.

Visualizations: We use t-SNE [68] to visualize cross-modal features on CIFAR100 B0 Inc5 in Figure 4c and Figure 4d, comparing models with and without preserving hierarchical structure across tasks. First-task visual features are dots (◯) and second-task ones are squares (□). Text embeddings are triangles (△), and shaded regions denote decision areas induced by the text points. As shown in Figure 4c, without preserving hierarchical structure, visual clusters drift away from their text counterparts and even enter other classes' regions. As shown in Figure 4d, preserving hierarchical structure suppresses drift and keeps image–text features co-located for old and new tasks. These results indicate that HASTEN uses hierarchical supervision to preserve the learned structure and maintain cross-modal alignment during incremental training.

Image traversals: Following [13], we visualize hierarchical consistency on UCF101 by traversing from each image embedding to the root ("entity") along the geodesic in our hyperbolic space, and along the image-to-`[ROOT]` path for CLIP. We sample 100 equally spaced points on each path; ours retrieves the nearest ancestor node, while CLIP retrieves the highest-similarity text from a text set that includes the root. As shown in Figure 5, our method yields descriptions that become increasingly generic as the path approaches the root, indicating that the hyperbolic representation captures the semantic tree. Conversely, CLIP's flat feature space typically produces a single description. More details and examples are in Appendix G.

6 Conclusion

This paper presents HASTEN, a hierarchy-aware framework for CLIP-based CIL in hyperbolic space. HASTEN leverages external hierarchical knowledge as structured supervision to organize visual and textual features into a hierarchically structured representation, instead of isolated class labels. This representation helps maintain semantic relations and suppress feature drift during incremental learning. To further reduce forgetting, null-space projection stabilizes the shared hyperbolic mapper by reducing interference with prior mappings. Extensive experiments show consistent gains across diverse benchmarks, improved resistance to feature drift, and reduced catastrophic forgetting.
Limitations: This paper uses a GPT-generated semantic tree for hierarchical supervision, which may be imperfect in specialized domains with ambiguous or fine-grained class relations. Future work will explore curated taxonomies, data-driven hierarchy induction, and light human verification for more reliable domain-specific deployment.

Acknowledgments

This work was supported in part by the Basic Research Program of Jiangsu (BK20251251), NSFC (62506160), JSTJ-2025-147, Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118), the 111 Center (No. B26023), and the Collaborative Innovation Center of Novel Software Technology and Industrialization. The authors declare no competing interests.

References

[1] Charu C Aggarwal. A survey of stream clustering algorithms. In *Data Clustering*, pages 231–258. Chapman and Hall/CRC, 2018.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[3] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11254–11263, 2019.

[4] 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. *Advances in neural information processing systems*, 32, 2019.

[5] 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.

[6] James W. Cannon, William J. Floyd, Richard Kenyon, Walter R. Parry, and Mathematical Sciences Research Institute. *Hyperbolic Geometry*, page 59–116. Cambridge University Press, 1997.

[7] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019.

[9] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.

[10] Marco D'Alessandro, Alberto Alonso, Enrique Calabrés, and Mikel Galar. Multimodal parameter-efficient few-shot class incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 3393–3403, 2023.

[11] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Karan Desai, Maximilian Nickel, Tanmay Rajpurohit, Justin Johnson, and Shanmukha Ramakrishna Vedantam. Hyperbolic image-text representations. In *International Conference on Machine Learning*, pages 7694–7731. PMLR, 2023.

[14] Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024.

[15] Songlin Dong, Xiaopeng Hong, Xiaoyu Tao, Xinyuan Chang, Xing Wei, and Yihong Gong. Few-shot class-incremental learning via relation knowledge distillation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1255–1263, 2021.

[16] 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.

[17] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European conference on computer vision*, pages 86–102. Springer, 2020.

[18] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9285–9295, 2022.

[19] Ethan Dyer, Aitor Lewkowycz, and Vinay Ramasesh. Effect of scale on catastrophic forgetting in neural networks. In *International conference on learning representations*, 2022.

[20] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4): 128–135, 1999.

[21] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in neural information processing systems*, 31, 2018.

[22] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *European Conference on Computer Vision*, pages 423–439. Springer, 2022.

[23] Yuting Gao, Jinfeng Liu, Zihan Xu, Jun Zhang, Ke Li, Rongrong Ji, and Chunhua Shen. Pyramidclip: Hierarchical feature alignment for vision-language model pretraining. *Advances in neural information processing systems*, 35:35959–35970, 2022.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[25] Mikhael Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987.

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

[27] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. 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 8340–8349, 2021.

[28] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[29] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839, 2019.

[30] 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.

[31] Gabriel Ilharco, Mitchell Wortsman, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, et al. Openclip. *Zenodo*, 2021.

[32] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision*, pages 709–727. Springer, 2022.

[33] Dahuin Jung, Dongyoon Han, Jihwan Bang, and Hwanjun Song. Generating instance-level prompts for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11847–11857, 2023.

[34] 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.

[35] 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.

[36] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[37] Guannan Lai, Yujie Li, Xiangkun Wang, Junbo Zhang, Tianrui Li, and Xin Yang. Order-robust class incremental learning: Graph-driven dynamic similarity grouping. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4894–4904, 2025.

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[39] Li Li, Jiawei Peng, Huiyi Chen, Chongyang Gao, and Xu Yang. How to configure good in-context sequence for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26710–26720, 2024.

[40] Lan Li, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Addressing imbalanced domain-incremental learning through dual-balance collaborative experts. In *International Conference on Machine Learning*, 2025.

[41] Lan Li, Tao Hu, Da-Wei Zhou, Jia-Qi Yang, Han-Jia Ye, and De-Chuan Zhan. Bofa: Bridge-layer orthogonal low-rank fusion for clip-based class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 22967–22975, 2026.

[42] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

[43] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

[44] Yichen Lu, Mei Wang, and Weihong Deng. Augmented geometric distillation for data-free incremental person reid. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7329–7338, 2022.

[45] Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11371–11380, 2023.

[46] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[47] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2023.

[48] Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton Van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36:12022–12053, 2023.

[49] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[50] 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.

[51] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13698–13707, 2021.

[52] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[53] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12): 10023–10044, 2022.

[54] Tobia Poppi, Tejaswi Kasarla, Pascal Mettes, Lorenzo Baraldi, and Rita Cucchiara. Hyperbolic safety-aware vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4222–4232, 2025.

[55] Zhi-Hong Qi, Da-Wei Zhou, Yiran Yao, Han-Jia Ye, and De-Chuan Zhan. Adaptive adapter routing for long-tailed class-incremental learning. *Machine Learning*, 114(3):68, 2025.

[56] 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.

[57] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

[58] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[59] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International symposium on graph drawing*, pages 355–366. Springer, 2011.

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

[61] Guangyuan Shi, Jiaxin Chen, Wenlong Zhang, Li-Ming Zhan, and Xiao-Ming Wu. Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima. *Advances in neural information processing systems*, 34:6747–6761, 2021.

[62] Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip HS Torr, Song Bai, and Vincent YF Tan. Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16722–16731, 2022.

[63] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

[64] 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, 2023.

[65] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[66] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[67] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[68] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[69] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[70] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[71] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision*, pages 398–414. Springer, 2022.

[72] Fu-Yun Wang, Da-Wei Zhou, Liu Liu, Han-Jia Ye, Yatao Bian, De-Chuan Zhan, and Peilin Zhao. BEEF: Bi-compatible class-incremental learning via energy-based expansion and fusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[73] 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.

[74] Runqi Wang, Xiaoyue Duan, Guoliang Kang, Jianzhuang Liu, Shaohui Lin, Songcen Xu, Jinhu Lü, and Baochang Zhang. Attriclip: A non-incremental learner for incremental knowledge learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3654–3663, 2023.

[75] 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*, pages 631–648. Springer, 2022.

[76] 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, 2022.

[77] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382, 2019.

[78] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[79] Zhen-Hao Xie, Yu-Cheng Shi, and Da-Wei Zhou. Area: Attribute extraction and aggregation for clip-based class-incremental learning. In *International Conference on Machine Learning*, 2026.

[80] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3014–3023, 2021.

[81] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[82] Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23219–23230, 2024.

[83] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6982–6991, 2020.

[84] Lu Yu, Haoyu Han, Zhe Tao, Hantao Yao, and Changsheng Xu. Language guided concept bottleneck models for interpretable continual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14976–14986, 2025.

[85] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. Pmlr, 2017.

[86] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13208–13217, 2020.

[87] Bowen Zheng, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Task-agnostic guided feature expansion for class-incremental learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10099–10109, 2025.

[88] 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*, 133(3):1012–1032, 2025.

[89] 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*, pages 3314–3325, 2025.

[90] 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, 2025.

[91] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16816–16825, 2022.

[92] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International journal of computer vision*, 130(9):2337–2348, 2022.

[93] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5871–5880, 2021.

Appendix

In this appendix, we provide supplementary materials for HASTEN, including additional method details, semantic-tree construction protocols, comprehensive experimental results, robustness analyses, and efficiency studies.

Section A provides more details of HASTEN, including an overview of the training algorithm and a geometric justification for using Lorentz hyperbolic space to model hierarchical structures.

Section B presents the prompts and protocols used to dynamically construct hierarchical semantic trees with GPT-5.

Section C provides supplementary experimental results and analyses, including component analysis, forgetting analysis, generalization studies, robustness evaluations, and descriptions of the compared methods.

Section D supplements the parameter robustness study with sensitivity analyses of additional hyperparameters, including λ_2 for text-to-image hierarchy, δ for parent–child separation, the curvature c, and the cone half-aperture parameter κ.

Section E analyzes computational efficiency, including trainable parameters, running time, and additional overhead.

Section F reports the full incremental learning curves under different benchmark settings.

Section G provides more image traversal examples and details the construction of the text set used for hierarchical consistency evaluation.

Section H quantitatively verifies the learned hierarchical structure across multiple datasets, showing that the hyperbolic representations preserve radial expansion and semantic topology after the final incremental stage.

Section I discusses the broader impacts and practical considerations.

A More Details of HASTEN

A.1 Overview of Training Algorithm

The training process of HASTEN is summarized in Algorithm 1. At the beginning of the b-th incremental task, HASTEN first constructs or expands the hierarchical semantic tree using only the currently observed classes, without accessing future classes (Lines 2–6). Then, all previously learned hierarchy-aware modules and cached virtual-node embeddings are frozen, and the current hierarchy-aware modules are initialized for the new task (Lines 7–8). During training, frozen CLIP encoders extract visual and textual features, which are processed by the accumulated hierarchy-aware modules and mapped into the Lorentz hyperbolic space through the shared TP layer (Lines 16–18). The model is optimized with the hierarchical loss, hyperbolic contrastive loss, and TP regularization (Lines 20–23). To stabilize the shared TP, its gradient is projected onto the null space induced by prior-task features before the parameter update (Lines 25–26). After finishing the task, HASTEN updates the class statistics, covariance matrix, projection subspace, and cached virtual-node embeddings for subsequent tasks (Lines 28–31).

A.2 Geometric Justification of Hyperbolic Hierarchy Modeling

We provide a geometric justification for why the Lorentz hyperbolic space used in HASTEN is suitable for representing hierarchical structures.

Proposition 1. *The Lorentz hyperbolic space $\mathcal{B}^d$ with curvature parameter $c > 0$ provides exponentially increasing surface capacity with respect to the geodesic radius. Specifically, the surface area of a geodesic sphere with radius ρ grows as*

$$S_{\mathcal{B}^d}(\rho) = \Theta\left(e^{(d-1)\sqrt{c}\rho}\right), \quad \rho \to \infty. \tag{22}$$

Thus, it naturally matches the exponential branching pattern of tree-structured data and supports a depth-radius representation of hierarchical concepts.

Algorithm 1 Training HASTEN for CLIP-based CIL

Input: Incremental task dataset $\mathcal{D}^b$; class set $\mathcal{Y}_b$; frozen CLIP encoders g_v, g_t; previous semantic tree $\mathcal{K}^{(b-1)}$; previous modules $\{H_v^i, H_t^i\}_{i=1}^{b-1}$; shared hyperbolic mapper TP; previous covariance $C^{(b-1)}$; cached virtual-node embeddings.
Output: Updated model, semantic tree, covariance, projection subspace, and cached virtual-node embeddings.

if $b = 1$ **then**
  Construct the base semantic tree $\mathcal{K}^{(b)}$ from $\mathcal{Y}_b$ using GPT-5;
else
  Expand $\mathcal{K}^{(b-1)}$ with new classes $\mathcal{Y}_b$ without modifying existing edges;
end if
Obtain the current semantic tree $\mathcal{K}^{(b)} = (\mathcal{C}, \mathcal{E}, r)$;
Freeze previous hierarchy-aware modules $\{H_v^i, H_t^i\}_{i=1}^{b-1}$ and cached virtual-node embeddings;
Initialize current hierarchy-aware modules H_v^b and H_t^b;
if $b > 1$ **then**
  Compute the projection basis $V_\perp^{(b-1)}$ from $C^{(b-1)}$ via SVD;
else
  Set $V_\perp^{(b-1)}$ to the identity projection;
end if
for each training iteration on $\mathcal{D}^b$ **do**
  Sample a mini-batch $(\mathbf{I}, y)$ from $\mathcal{D}^b$;
  Extract frozen CLIP features $\mathbf{e}_v = g_v(\mathbf{I})$ and $\mathbf{e}_t = g_t(\mathbf{t}_y)$;
  Aggregate hierarchy-aware features: $\tilde{\mathbf{z}}_v = \sum_{i=1}^{b} H_v^i(\mathbf{e}_v)$, $\tilde{\mathbf{z}}_t = \sum_{i=1}^{b} H_t^i(\mathbf{e}_t)$;
  Map features into hyperbolic space: $\mathbf{z}_v = \mathrm{TP}(\tilde{\mathbf{z}}_v)$, $\mathbf{z}_t = \mathrm{TP}(\tilde{\mathbf{z}}_t)$;
  Sample old-class surrogates from $\mathcal{N}(\mu_k, \Sigma_k)$ for previous classes when needed;
  Compute the text–text and text–image hierarchical loss $\mathcal{L}_{\mathrm{hier}}$ using $\mathcal{K}^{(b)}$ and cached virtual nodes;
  Compute the hyperbolic contrastive loss $\mathcal{L}_{\mathrm{contrast}}$;
  Compute the TP regularization loss $\mathcal{L}_{\mathrm{tp}}$;
  Compute the total loss $\mathcal{L} = \mathcal{L}_{\mathrm{hier}} + \mathcal{L}_{\mathrm{contrast}} + \beta\mathcal{L}_{\mathrm{tp}}$;
  Back-propagate $\mathcal{L}$ to obtain gradients for H_v^b, H_t^b, and TP;
  Project the TP gradient: $\Delta w_{\mathrm{proj}} = V_\perp^{(b-1)}(V_\perp^{(b-1)})^\top \Delta w$;
  Update H_v^b, H_t^b, and update TP with Δw_{proj};
end for
Estimate class statistics $\{\mu_k, \Sigma_k\}$ for classes in $\mathcal{Y}_b$ using frozen visual features;
Compute $C_{\mathrm{new}}^{(b)}$ from pre-TP features and update $C^{(b)}$;
Recompute the SVD of $C^{(b)}$ and update the projection subspace for future tasks;
Cache and freeze hyperbolic text embeddings of virtual nodes in $\mathcal{K}^{(b)}$;
return Updated HASTEN.

Proof: Following the notation in the main paper, we use the Lorentz model $\mathcal{B}^d$ with curvature parameter $c > 0$. For a point $\mathbf{p} = [\mathbf{p}_s, p_t] \in \mathcal{B}^d$, the Lorentz constraint is

$$\langle \mathbf{p}, \mathbf{p} \rangle_{\mathcal{B}} = \|\mathbf{p}_s\|^2 - p_t^2 = -\frac{1}{c}, \quad p_t > 0. \tag{23}$$

The geodesic distance used in the main paper is

$$d_{\mathcal{B}}(\mathbf{p}, \mathbf{q}) = \frac{1}{\sqrt{c}} \cosh^{-1}(-c\langle \mathbf{p}, \mathbf{q} \rangle_{\mathcal{B}}). \tag{24}$$

Let $\rho = d_{\mathcal{B}}(\mathbf{o}, \mathbf{p})$ denote the geodesic radius from the origin. For general c, the origin is $\mathbf{o} = [\mathbf{0}, 1/\sqrt{c}]$, which reduces to $[\mathbf{0}, 1]$ when $c = 1$. Any point at radius ρ can be parameterized by a direction $\mathbf{u} \in \mathbb{S}^{d-1}$ as

$$\Phi(\rho, \mathbf{u}) = \left[\frac{\sinh(\sqrt{c}\rho)}{\sqrt{c}} \mathbf{u}, \frac{\cosh(\sqrt{c}\rho)}{\sqrt{c}} \right]. \tag{25}$$

This parameterization lies on $\mathcal{B}^d$ because

$$\left\|\frac{\sinh(\sqrt{c}\rho)}{\sqrt{c}}\mathbf{u}\right\|^2 - \left(\frac{\cosh(\sqrt{c}\rho)}{\sqrt{c}}\right)^2 = \frac{\sinh^2(\sqrt{c}\rho) - \cosh^2(\sqrt{c}\rho)}{c} = -\frac{1}{c}. \tag{26}$$

Moreover, $\Phi(\rho, \mathbf{u})$ is indeed at geodesic distance ρ from the origin, since

$$-c\langle \mathbf{o}, \Phi(\rho, \mathbf{u})\rangle_{\mathcal{B}} = \cosh(\sqrt{c}\rho), \tag{27}$$

and therefore

$$d_{\mathcal{B}}(\mathbf{o}, \Phi(\rho, \mathbf{u})) = \frac{1}{\sqrt{c}}\cosh^{-1}\big(\cosh(\sqrt{c}\rho)\big) = \rho. \tag{28}$$

We next derive the volume-growth property of $\mathcal{B}^d$. Differentiating the above parameterization gives

$$d\mathbf{p}_s = \cosh(\sqrt{c}\rho)\mathbf{u}\, d\rho + \frac{\sinh(\sqrt{c}\rho)}{\sqrt{c}} d\mathbf{u}, \quad dp_t = \sinh(\sqrt{c}\rho) d\rho. \tag{29}$$

Since $\|\mathbf{u}\| = 1$, we have $\mathbf{u}^\top d\mathbf{u} = 0$. Thus, the induced metric becomes

$$\begin{aligned} ds^2 &= \langle d\mathbf{p}, d\mathbf{p}\rangle_{\mathcal{B}} \\ &= \|d\mathbf{p}_s\|^2 - dp_t^2 \\ &= \big(\cosh^2(\sqrt{c}\rho) - \sinh^2(\sqrt{c}\rho)\big) d\rho^2 + \left(\frac{\sinh(\sqrt{c}\rho)}{\sqrt{c}}\right)^2 d\Omega^2 \\ &= d\rho^2 + \left(\frac{\sinh(\sqrt{c}\rho)}{\sqrt{c}}\right)^2 d\Omega^2, \end{aligned} \tag{30}$$

where $d\Omega^2$ is the metric on the unit sphere $\mathbb{S}^{d-1}$. Hence, the surface area of a geodesic sphere with radius ρ is

$$S_{\mathcal{B}^d}(\rho) = \omega_{d-1}\left(\frac{\sinh(\sqrt{c}\rho)}{\sqrt{c}}\right)^{d-1}, \tag{31}$$

where ω_{d-1} is the surface area of the unit $(d-1)$-sphere. Since

$$\sinh(\sqrt{c}\rho) = \frac{e^{\sqrt{c}\rho} - e^{-\sqrt{c}\rho}}{2}, \tag{32}$$

we obtain

$$S_{\mathcal{B}^d}(\rho) = \Theta\Big(e^{(d-1)\sqrt{c}\rho}\Big), \quad \rho \to \infty. \tag{33}$$

Therefore, the available surface capacity at radius ρ grows exponentially with ρ. This is fundamentally different from Euclidean space, where the surface area grows only polynomially:

$$S_{\mathbb{R}^d}(\rho) = \omega_{d-1}\rho^{d-1}. \tag{34}$$

This exponential growth matches the structure of a tree. For a balanced b-ary tree, the number of nodes at depth ℓ is b^ℓ, which also grows exponentially with depth. If tree depth is represented by geodesic radius and nodes at depth ℓ are placed around radius $\rho_\ell = \gamma\ell$, then the available hyperbolic surface capacity grows as

$$S_{\mathcal{B}^d}(\rho_\ell) = \Theta\Big(e^{(d-1)\sqrt{c}\gamma\ell}\Big). \tag{35}$$

Choosing

$$\gamma = \frac{\log b}{(d-1)\sqrt{c}} \tag{36}$$

makes this growth proportional to b^ℓ. Thus, hyperbolic space can allocate capacity for exponentially many nodes using only a linearly increasing geodesic radius. In contrast, since the surface area of a Euclidean sphere grows as R^{d-1}, matching b^ℓ nodes at depth ℓ requires

$$R_\ell = \Omega\Big(b^{\ell/(d-1)}\Big), \tag{37}$$

which grows exponentially with ℓ. This mismatch explains the crowding issue when many fine-grained concepts must be represented around broad parent concepts in Euclidean space.

The Lorentz distance also naturally reflects both hierarchical depth and branch separation. Consider two points $\mathbf{p} = \Phi(\rho, \mathbf{u})$ and $\mathbf{q} = \Phi(\eta, \mathbf{v})$, where $\mathbf{u}^\top \mathbf{v} = \cos\theta$. Substituting the parameterization into the Lorentz distance formula gives

$$\cosh\big(\sqrt{c}\, d_{\mathcal{B}}(\mathbf{p}, \mathbf{q})\big) = \cosh(\sqrt{c}\rho)\cosh(\sqrt{c}\eta) - \sinh(\sqrt{c}\rho)\sinh(\sqrt{c}\eta)\cos\theta. \tag{38}$$

When $\theta = 0$, the two points lie on the same radial direction, and the above equation reduces to

$$\cosh\big(\sqrt{c}\, d_{\mathcal{B}}(\mathbf{p}, \mathbf{q})\big) = \cosh\big(\sqrt{c}(\rho - \eta)\big)\,, \tag{39}$$

which implies

$$d_{\mathcal{B}}(\mathbf{p}, \mathbf{q}) = |\rho - \eta|. \tag{40}$$

Thus, movement along the same radial direction corresponds directly to changes in semantic depth, making it natural to place more generic concepts closer to the origin and more fine-grained concepts farther away. When $\theta > 0$, the distance increases with angular separation because

$$\frac{\partial}{\partial\theta}\left[\cosh(\sqrt{c}\rho)\cosh(\sqrt{c}\eta) - \sinh(\sqrt{c}\rho)\sinh(\sqrt{c}\eta)\cos\theta\right] = \sinh(\sqrt{c}\rho)\sinh(\sqrt{c}\eta)\sin\theta \geq 0. \tag{41}$$

Since $\cosh^{-1}(\cdot)$ is monotonic, larger angular separation yields larger geodesic distance. Therefore, radial coordinates can encode hierarchical depth, while angular directions can separate different semantic branches.

This geometry is also consistent with the entailment cones used in our hierarchical objective. Recall that the cone aperture is defined as

$$\mathrm{aper}(\mathbf{p}) = \sin^{-1}\left(\frac{2\kappa}{\sqrt{c}\|\mathbf{p}\|}\right). \tag{42}$$

The norm appearing in the aperture increases as a point moves outward in geodesic radius. For example, under the Lorentz parameterization, the spatial norm satisfies

$$\|\mathbf{p}_s\| = \frac{\sinh(\sqrt{c}\rho)}{\sqrt{c}}, \tag{43}$$

which is monotonically increasing in ρ. The full Euclidean norm of $\mathbf{p}$ is also monotonically increasing in ρ. Therefore, larger-radius points induce smaller cone apertures. To see this explicitly, for fixed c and κ, let $s = \|\mathbf{p}\|$. When the aperture is well-defined, $s > \frac{2\kappa}{\sqrt{c}}$, and

$$\frac{\partial \mathrm{aper}(\mathbf{p})}{\partial s} = -\frac{2\kappa}{\sqrt{c}s^2} \cdot \frac{1}{\sqrt{1 - \left(\frac{2\kappa}{\sqrt{c}s}\right)^2}} < 0. \tag{44}$$

Hence, higher-level concepts can have broader descendant regions, while deeper and more specific concepts induce tighter cones. The constraint

$$\mathrm{ext}(\mathbf{p}, \mathbf{q}) \leq \mathrm{aper}(\mathbf{p}) \tag{45}$$

therefore turns a parent–child relation into a geometric inclusion relation in $\mathcal{B}^d$.

Overall, the Lorentz hyperbolic space is suitable for hierarchical representation because it provides exponential capacity for tree-like branching, a natural depth-radius correspondence, and a distance geometry that separates different branches while supporting ancestor-descendant organization. These properties support our use of hyperbolic space for modeling hierarchical semantic structures in CLIP-based class-incremental learning.

B Generating Hierarchical Semantic Tree

In this section, we present the full prompts used for hierarchical semantic tree generation and illustrate them with a CIFAR-100 example under the B50 Inc10 incremental setting. To strictly adhere to the class-incremental learning paradigm and prevent any leakage of future classes, our method dynamically constructs the hierarchy *incrementally* in two stages:

1. **Base construction**: build an initial taxonomy from the first task's classes only;
2. **Incremental update**: progressively integrate each subsequent task's new classes while *strictly preserving* all existing parent–child edges.

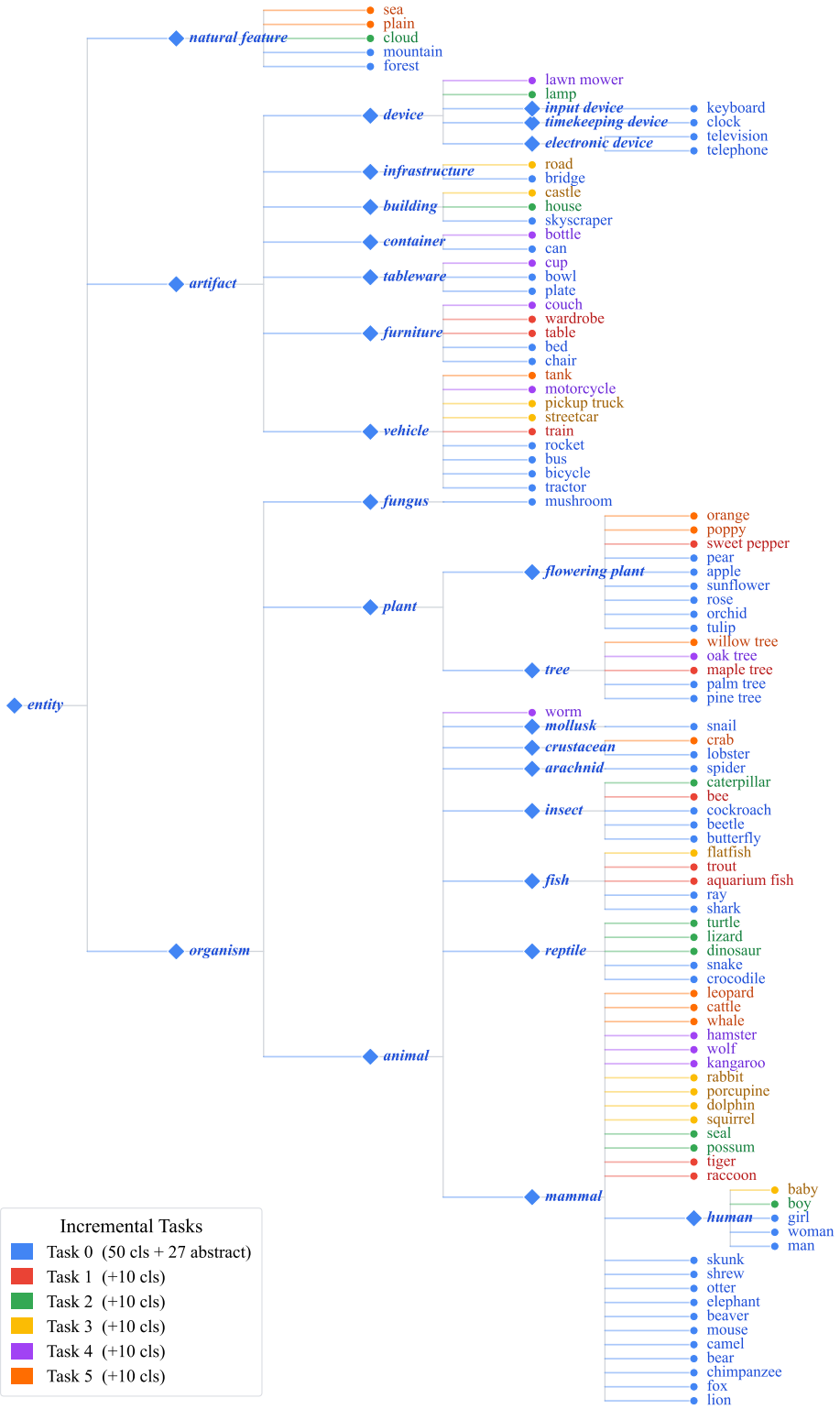


Figure 6: Visualization of the semantic tree structure derived from the output JSON on CIFAR100. The root node "entity" branches into abstract categories (e.g., "organism"), which further subdivide until reaching the specific leaf classes from the input list.

B.1 Stage 1: Base Taxonomy Construction

The base prompt takes only the class names available at Task 0 and produces a complete "is-a" hierarchy rooted at `entity`.

Prompt *(Base Construction)*

You are a meticulous and highly accurate taxonomist and ontologist. Your goal is to construct an initial hierarchical "is-a" taxonomy from a flat list of class names.

Non-negotiable Rules

1) **Total Inclusion** – The final JSON must contain ALL class names from the input list. Each class appears exactly once. No omissions, no duplicates.
2) **Name Preservation** (Critical) – Keep every input class name *exactly* as given (spelling, spacing, case). Abstract parent nodes you create MAY use `snake_case`.
3) **Single Root** – All branches connect to one root node: `"entity"`. No disconnected subtrees.
4) **True "is-a" Relationships** – Every parent → child must be a genuine "is-a-type-of" relation. Avoid vague placeholders (`"object"`, `"thing"`, `"item"`) unless logically necessary.
5) **Minimal but Coherent Abstraction** – Introduce abstract parents only when necessary. Prefer semantically meaningful intermediate categories. No unnecessary multi-layer chains.
6) **Structure Constraints** – Single valid JSON object. Keys = parent nodes, values = lists of direct children. Leaf nodes appear only in child lists. No empty lists, no cycles, no self-loops.

Self-check before output: ✓ Every input class appears exactly once ✓ All nodes reachable from `"entity"` ✓ All edges are valid "is-a" relations ✓ No input class renamed

CLASS LIST (Task 0, 50 classes):

```
["lion", "tulip", "butterfly", "pear", "tractor", "chair", "fox", "rocket",
"chimpanzee", "bear", "snail", "forest", "orchid", "telephone", "mountain",
"spider", "woman", "camel", "can", "plate", "man", "pine tree", "keyboard",
"bridge", "palm tree", "mouse", "crocodile", "television", "lobster", "snake",
"cockroach", "apple", "beetle", "mushroom", "beaver", "shark", "bicycle", "girl",
"bowl", "bus", "rose", "shrew", "elephant", "otter", "sunflower", "bed", "ray",
"skunk", "clock", "skyscraper"]
```

Response *(Base Taxonomy – Task 0)*

```
{
"entity": ["organism", "artifact", "natural_feature"],
"organism": ["animal", "plant", "fungus"],
"animal": ["mammal", "reptile", "fish", "insect",
           "arachnid", "crustacean", "mollusk"],
"mammal": ["lion", "fox", "chimpanzee", "bear", "camel",
           "mouse", "beaver", "elephant", "otter", "shrew",
           "skunk", "human"],
"human": ["man", "woman", "girl"],
"reptile": ["crocodile", "snake"],
"fish": ["shark", "ray"],
"insect": ["butterfly", "beetle", "cockroach"],
"arachnid": ["spider"],
"crustacean": ["lobster"],
"mollusk": ["snail"],
"plant": ["tree", "flowering_plant"],
"tree": ["pine tree", "palm tree"],
"flowering_plant": ["tulip", "orchid", "rose", "sunflower",
                    "apple", "pear"],
"fungus": ["mushroom"],
"natural_feature": ["forest", "mountain"],
"artifact": ["vehicle", "furniture", "tableware", "container",
             "building", "infrastructure", "device"],
"vehicle": ["tractor", "bicycle", "bus", "rocket"],
"furniture": ["chair", "bed"],
"tableware": ["plate", "bowl"],
```

```
"container": ["can"],
"building": ["skyscraper"],
"infrastructure": ["bridge"],
"device": ["electronic_device", "timekeeping_device",
           "input_device"],
"electronic_device": ["telephone", "television"],
"timekeeping_device": ["clock"],
"input_device": ["keyboard"]
}
```

B.2 Stage 2: Incremental Taxonomy Update

When a new task arrives, we feed the *current* hierarchy together with the new class list into the incremental prompt. The key constraint is **no rewiring**: all existing edges are frozen, and the model may only *append* new children or introduce new parent keys.

Prompt *(Incremental Update)*

You are a meticulous and highly accurate taxonomist and ontologist. Your goal is to update an existing hierarchical "is-a" structure by integrating a list of NEW class names into it.

Non-negotiable Rules

1) **Total Inclusion** – The final JSON must retain ALL strings from the existing hierarchy AND include ALL new classes. Do not delete anything.
2) **Name Preservation** (Critical) – Keep every new class name exactly as given. New abstract nodes you invent MAY use `snake_case`.
3) **Stability / No Rewiring** (Critical) – Do **not** remove or change any existing parent→child edge. Do **not** move an existing child between parents. You may **only**: (a) append new children to existing parent lists; (b) add new parent keys; (c) add new high-level children under `"entity"`.
4) **Minimal New Abstract Nodes** – Create the minimum number of new grouping nodes required.
5) **Fallback Bucket** – If a new class cannot be confidently placed, put it under `"unclassified"` (reachable from `"entity"`).

Self-check: ✓ Every new class appears exactly once ✓ Every existing string still present ✓ No existing edge removed

EXISTING HIERARCHY:

{ ... the Task 0 JSON shown above ... }

NEW CLASSES (Task 1, 10 classes):

`["maple tree", "sweet pepper", "train", "aquarium fish", "bee", "table", "wardrobe", "raccoon", "tiger", "trout"]`

Response *(Updated Taxonomy – after Task 1)*

```
{
"entity": ["organism", "artifact", "natural_feature"],
"organism": ["animal", "plant", "fungus"],
"animal": ["mammal", "reptile", "fish", "insect",
           "arachnid", "crustacean", "mollusk"],
"mammal": ["lion", "fox", "chimpanzee", "bear", "camel",
           "mouse", "beaver", "elephant", "otter", "shrew",
           "skunk", "human", "raccoon", "tiger"],
"human": ["man", "woman", "girl"],
"reptile": ["crocodile", "snake"],
"fish": ["shark", "ray", "aquarium fish", "trout"],
"insect": ["butterfly", "beetle", "cockroach", "bee"],
"arachnid": ["spider"],
"crustacean": ["lobster"],
"mollusk": ["snail"],
"plant": ["tree", "flowering_plant"],
"tree": ["pine tree", "palm tree", "maple tree"],
"flowering_plant": ["tulip", "orchid", "rose", "sunflower",
```

```
                    "apple", "pear", "sweet pepper"],
"fungus": ["mushroom"],
"natural_feature": ["forest", "mountain"],
"artifact": ["vehicle", "furniture", "tableware", "container",
             "building", "infrastructure", "device"],
"vehicle": ["tractor", "bicycle", "bus", "rocket", "train"],
"furniture": ["chair", "bed", "table", "wardrobe"],
"tableware": ["plate", "bowl"],
"container": ["can"],
"building": ["skyscraper"],
"infrastructure": ["bridge"],
"device": ["electronic_device", "timekeeping_device",
           "input_device"],
"electronic_device": ["telephone", "television"],
"timekeeping_device": ["clock"],
"input_device": ["keyboard"]
}
```

Observations:

- All 10 new classes are correctly inserted into existing parent lists.
- No existing parent→child edge has been removed or rewired.
- New classes are appended to semantically appropriate parents: `"raccoon"` and `"tiger"` → `mammal`; `"train"` → `vehicle`; `"maple tree"` → `tree`; etc.
- Every path still connects to `"entity"`.
- This process repeats identically for Tasks 2–5, each time feeding the latest accumulated hierarchy and the new batch of 10 classes.

Figure 6 visualizes the final hierarchical semantic tree for CIFAR-100 under the B50 Inc10 setting, constructed through the above two-stage prompting process. Nodes are color-coded by the incremental task that introduced them: **blue** denotes Task 0 (the 50 base classes and all abstract grouping nodes), while subsequent tasks are shown in **red** (Task 1), **green** (Task 2), **yellow** (Task 3), **purple** (Task 4), and **orange** (Task 5). Within each parent node, children added by later tasks are appended below existing ones, visually reflecting the incremental growth of the taxonomy. Notably, the no-rewiring constraint ensures that all blue edges established at Task 0 remain unchanged throughout, while colored branches extend outward as new classes arrive.

Validation and fallback. After generation, we check whether all seen classes appear exactly once, all nodes are reachable from the root, and no previous edge is removed or rewired. If the output violates these constraints, we re-query the model; unresolved classes are placed under the "unclassified" node.

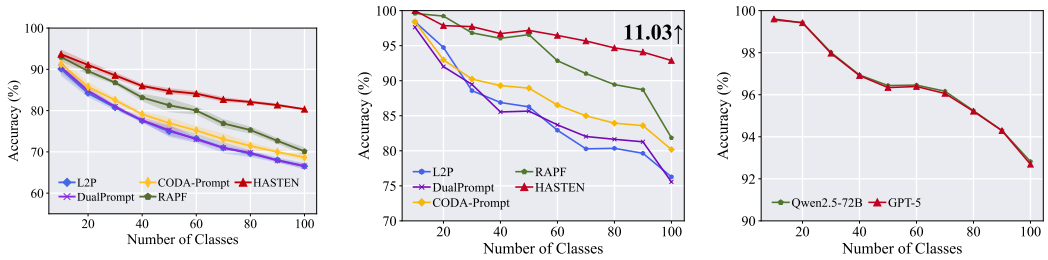
C Supplementary Results and Methods

C.1 Evaluation Across Multiple Random Seeds

In the main paper, all experimental results follow the standard CIL protocol with a fixed class order. To further assess statistical robustness, we additionally repeat the experiments under multiple random class-order splits. Specifically, we conduct five runs with different class-order seeds and report the mean accuracy together with the standard deviation. Figure 7a shows the results on ImageNet-R B0 Inc20, where the solid lines denote the mean performance and the shaded bands indicate one standard deviation across runs. The results show that HASTEN consistently outperforms the baselines under different class orders, demonstrating strong robustness to randomness in class splitting.

C.2 Results with Different Backbones

In the main paper, our experiments mainly use CLIP ViT-B/16 with LAION-400M pre-trained weights [31]. To further examine the generality of our approach, we additionally report results of our method using the OpenAI CLIP pre-trained weights on UCF B0 Inc10, as shown in Figure 7b. Across different backbone initializations, HASTEN consistently outperforms competing methods, demonstrating strong robustness to variations in pre-trained models.

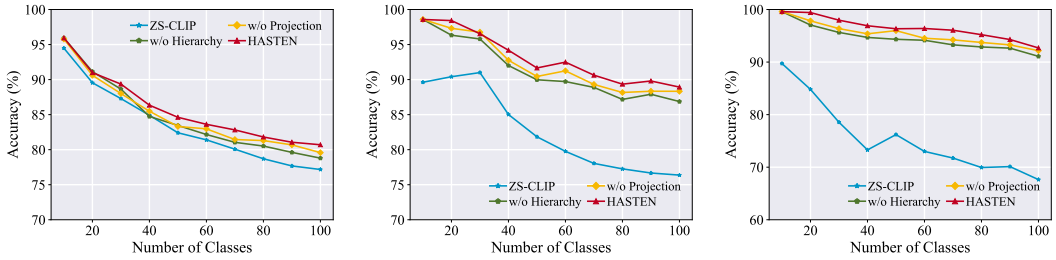


(a) Results on ImageNet-R B0 Inc20 with multiple runs. HASTEN consistently outperforms other methods by a substantial margin.

(b) Experiments when using OpenAI weights on UCF B0 Inc10. HASTEN consistently outperforms other methods across different backbone weights.

(c) Results on UCF B0 Inc10 with different LLMs for hierarchical semantic tree. HASTEN is robust and compatible with various LLMs.

Figure 7: Comprehensive analysis of HASTEN on different benchmarks. Our method demonstrates consistent superiority and robustness across various random seeds, pre-trained checkpoints, and LLMs.



(a) ImageNet-R B0 Inc20 (b) Cars B0 Inc10 (c) UCF B0 Inc10

Figure 8: Additional ablation studies on ImageNet-R, Cars, and UCF. The full HASTEN model consistently outperforms variants without hierarchical supervision or TP projection, confirming the complementary roles of hierarchy preservation and shared-mapper stabilization.

C.3 Component Analysis

To further investigate the contribution of individual components in HASTEN, we conduct additional ablation studies on the hierarchy-aware objectives, the parent–child separation term, and the TP regularizer. For individual loss ablations, all other components remain unchanged. We also construct a Euclidean counterpart by replacing the Lorentz hyperbolic representation with a Euclidean projection, the geodesic distances in the hyperbolic contrastive loss with Euclidean distances, and the entailment-cone constraints with Euclidean distance-based parent–child margins, while keeping the remaining experimental settings unchanged. All experiments are conducted under the B0 Inc10 setting, and we report both average accuracy $\bar{\mathcal{A}}$ and final-stage accuracy $\mathcal{A}_B$.

As shown in Table 2, removing either $\mathcal{L}_{\text{hier}}^{\text{txt}\to\text{txt}}$ or $\mathcal{L}_{\text{hier}}^{\text{txt}\to\text{img}}$ consistently degrades performance across all three datasets, indicating the complementary benefits of hierarchical semantic supervision and visual-textual alignment. Removing the parent-child separation term has a relatively minor impact, with slight performance decreases on Cars and UCF101 and marginal improvements on CIFAR100, suggesting that its contribution varies across datasets. Meanwhile, removing $\mathcal{L}_{\text{tp}}$ leads to substantial performance drops, supporting its role in stabilizing the shared hyperbolic mapper during continual updates. Finally, the Euclidean counterpart consistently underperforms the full model across all datasets, suggesting that the combined hyperbolic representation and geometry-aware objectives are beneficial for modeling hierarchical semantic relations. Overall, these results highlight the importance of hierarchy-aware supervision, TP regularization, and hyperbolic modeling in HASTEN.

C.4 Additional Ablation Studies

To further validate the contribution of each component beyond CIFAR100, we conduct additional ablation studies on three representative benchmarks: ImageNet-R B0 Inc20, Cars B0 Inc10, and UCF

Table 2: Component analysis of HASTEN under the B0 Inc10 setting. Each entry reports average accuracy $\bar{\mathcal{A}}$ and final-stage accuracy $\mathcal{A}_B$.

Variant	Cars		CIFAR100		UCF101	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
HASTEN (Full)	93.06	88.93	88.64	82.51	96.49	92.69
w/o $\mathcal{L}_{\text{hier}}^{\text{txt}\to\text{txt}}$	91.99	87.56	86.75	80.48	94.21	91.80
w/o $\mathcal{L}_{\text{hier}}^{\text{txt}\to\text{img}}$	92.38	88.25	87.37	81.33	95.84	92.54
w/o parent–child separation term	92.93	88.73	88.68	82.62	96.16	92.57
w/o $\mathcal{L}_{\text{tp}}$	90.98	86.25	85.70	80.55	94.06	90.46
Euclidean counterpart	90.14	85.86	84.58	78.41	93.25	88.57

Table 3: Forgetting measure F_B of different methods under different incremental settings. Lower F_B indicates less forgetting, and the best result in each column is highlighted in bold.

Method	Food		Cars		UCF		SUN	
	B0 Inc10	B50 Inc10	B0 Inc10	B50 Inc10	B0 Inc10	B50 Inc10	B0 Inc30	B150 Inc30
ZS-CLIP [56]	5.22	**2.84**	6.41	3.14	10.77	4.74	10.16	4.97
SimpleCIL [88]	6.22	3.76	5.11	2.51	4.86	**2.31**	8.12	**3.51**
L2P [76]	24.69	20.26	7.45	4.22	8.15	6.22	22.87	17.57
DualPrompt [75]	24.28	20.77	7.35	4.71	9.26	7.95	23.54	17.63
CODA-Prompt [64]	25.82	22.41	6.14	4.59	9.80	9.21	22.71	17.69
RAPF [30]	8.16	7.83	28.27	29.30	15.89	20.50	14.58	9.36
CLG-CBM [84]	11.79	9.38	6.86	4.86	8.25	6.87	14.62	12.26
PROOF [90]	16.58	16.94	4.48	**2.30**	4.22	4.85	18.33	15.71
HASTEN (Ours)	**4.86**	2.86	**4.30**	2.68	**4.12**	3.22	**7.32**	3.66

B0 Inc10. The results are shown in Figure 8. Across all datasets, the full HASTEN model consistently achieves the best performance throughout incremental learning. Removing the hierarchical supervision leads to clear and persistent degradation, especially in later stages, indicating that the learned hierarchy is important for mitigating feature drift and preserving cross-modal structure. Removing the TP projection also reduces performance, showing that stabilizing the shared hyperbolic mapper helps suppress accumulated interference from new tasks. Compared with ZS-CLIP, both ablated variants still benefit from adaptation, but neither matches the full model. These results demonstrate that hierarchical structure preservation and TP stabilization provide complementary benefits across datasets with different domains and granularities.

C.5 Forgetting Analysis

We further evaluate the forgetting behavior of different methods in Table 3. Following the standard CIL protocol [58, 90], we compute the forgetting measure after the final incremental stage as

$$F_B = \frac{1}{B-1} \sum_{b=1}^{B-1} \max_{l \in \{b,\dots,B-1\}} \big(\mathcal{A}_{l,b} - \mathcal{A}_{B,b}\big), \tag{46}$$

where $\mathcal{A}_{l,b}$ denotes the accuracy on task b after learning stage l, and $\mathcal{A}_{B,b}$ denotes the accuracy on task b after the final stage B. A lower F_B indicates that the method better preserves previously learned task knowledge.

As shown in Table 3, HASTEN achieves the lowest forgetting in all B0 settings on Food, Cars, UCF, and SUN, demonstrating its ability to preserve earlier knowledge when learning starts without base classes. In the half-base settings, HASTEN remains competitive and obtains forgetting scores close to the best methods, while also achieving stronger overall accuracy as reported in the main benchmark results. Compared with prompt-based methods such as L2P, DualPrompt, and CODA-Prompt, HASTEN consistently shows much smaller forgetting across datasets, indicating that hierarchical supervision and TP stabilization effectively reduce accumulated interference during incremental learning. Although ZS-CLIP and SimpleCIL show low forgetting in some settings, they perform limited or no task-specific adaptation and mainly serve as reference baselines. Overall, these results show that HASTEN improves incremental performance while maintaining strong resistance to forgetting across diverse datasets and class splits.

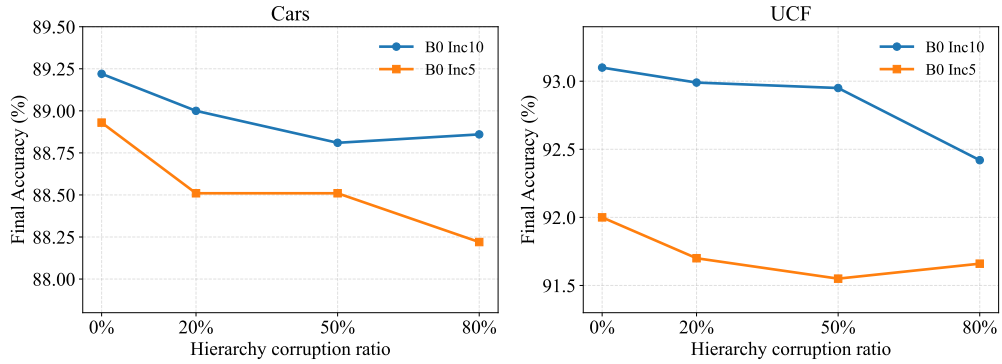


Figure 9: Robustness to noisy hierarchical supervision on Cars and UCF. We corrupt parent-child relations in the semantic tree and report final-stage accuracy. Clean hierarchies consistently perform best, showing the benefit of accurate hierarchical supervision.

C.6 Robustness to Different Tree Constructors

In the main paper, we use GPT-5 to generate the hierarchical semantic tree. To examine whether HASTEN depends on a specific LLM, we replace GPT-5 with Qwen2.5-72B [81] for tree construction and evaluate the resulting hierarchy on UCF B0 Inc10. As shown in Figure 7c, the two tree constructors yield highly consistent performance across incremental stages. This indicates that HASTEN does not rely on GPT-5 itself, but on the hierarchical structure used as supervision. Therefore, the semantic tree can also be generated by other LLMs or replaced by curated taxonomies and domain ontologies when available.

Robustness to noisy hierarchies. To further examine the dependence of HASTEN on the quality of the semantic tree, we intentionally corrupt the parent-child relations and report the final-stage accuracy in Figure 9. For each task-wise cumulative tree, we randomly reassign 20%, 50%, or 80% of existing parent-child edges to other valid internal parent nodes while preserving a valid DAG structure and keeping the numbers of nodes and edges unchanged. As shown in the figure, the clean hierarchy consistently achieves the best final accuracy across both datasets and both incremental settings. As the corruption ratio increases, the final accuracy generally decreases, indicating that correct parent-child relations contribute meaningful supervision for stabilizing incremental learning. This experiment is different from the 'w/o Hierarchy' ablation: the noisy-tree setting still keeps hierarchical supervision active but injects incorrect structural relations, whereas 'w/o Hierarchy' removes the semantic-tree supervision entirely. Therefore, the results show that hierarchical supervision itself is useful, and its effectiveness is further strengthened when the semantic tree provides more accurate structural relations.

C.7 Generalization of the Frozen CLIP Backbone

HASTEN builds its hierarchy-aware adaptation on top of frozen CLIP visual and textual encoders. The original CLIP backbone is not updated during continual training, so its pre-trained representation and zero-shot matching ability remain directly accessible. The hierarchy-aware modules and the shared hyperbolic mapper serve as additional adaptation layers for organizing seen incremental classes with semantic structure, rather than replacing the original CLIP feature space. Therefore, when evaluating unseen datasets or open-vocabulary categories, one can still directly use the original frozen CLIP pathway. This design allows HASTEN to improve class-incremental learning with hierarchy-aware adaptation while keeping CLIP's original generalization ability available.

C.8 Descriptions of Compared Methods

In this section, we provide details of the methods compared in the main paper. To ensure a fair comparison, all methods are evaluated using the same pre-trained model. The methods listed in Table 1 are described as follows:

- **Finetune**: initializes with a pre-trained CLIP model and finetunes it for each new task. Consequently, it experiences significant catastrophic forgetting on previous tasks.
- **SimpleCIL [88]:** utilizes only the pre-trained image encoder, excluding the text encoder. As a result, we remove the text branch in the pre-trained CLIP and evaluate the model using

just the visual branch. The frozen image encoder is used to generate class prototypes for each new class, with a cosine classifier employed for classification. Since the model isn't updated with backpropagation, this method highlights the generalizability of the pre-trained vision encoder for downstream tasks.

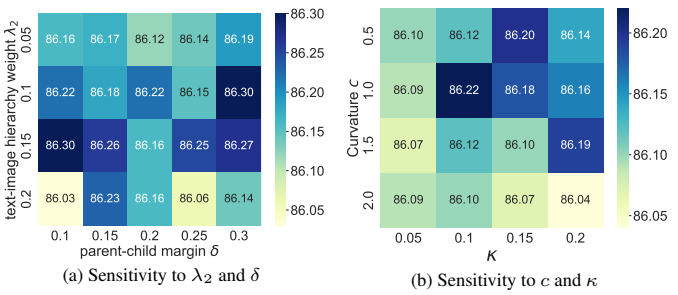
- **ZS-CLIP [56]:** freezes the pre-trained CLIP model and predicts the logits of incoming classes based on cosine similarity. It serves as a benchmark for evaluating the performance of the pre-trained CLIP on downstream tasks.
- **L2P [76]**: utilizes only the visual branch of CLIP. During the update process, it freezes the pre-trained weights and applies visual prompt tuning [32] to adapt to new task features. It generates instance-specific prompts through a prompt pool, constructed via key-value mapping.
- **DualPrompt [75]**: extends L2P by introducing two types of prompts, namely general and expert prompts. The rest of the procedure remains identical to L2P, including the use of a prompt pool to create instance-specific prompts. This approach also relies solely on the visual branch of CLIP.
- **CODA-Prompt [64]**: addresses the limitations of instance-specific prompt selection; this method replaces the prompt selection process with prompt reweighting. It uses attention-based recombination of prompts instead, while still involving only the visual branch of CLIP.
- **RAPF [30]**: enhances the continual learning capacity of CLIP by combining a hard class separation loss with decomposed parameter fusion to integrate new knowledge into the CLIP model.
- **PROOF [90]**: aims to improve CLIP's continual learning capabilities by learning expandable projection layers and a cross-modal fusion module. The historical prototypes of visual and textual features are passed through the cross-modal fusion module for further alignment.
- **CLG-CBM [84]:** utilizes a concept bottleneck layer to align semantics with CLIP, enabling the learning of human-understandable and cross-task generalizable concepts.

D Hyperparameter Sensitivity Analysis

In the main paper, we presented a robustness analysis for the text-to-text hierarchical weight λ_1 and the hyperbolic mapping regularizer weight β. This section provides complementary sensitivity analyses for additional key hyperparameters. For the sensitivity analyses, we use a validation split from the original training data. Specifically, the training data are split into training and validation subsets with a ratio of $4{:}1$, and all sensitivity results are reported as average accuracies across incremental stages on the validation subset.

First, we analyze the *text-to-image hierarchical loss weight* λ_2 from Eq. 13, which controls how strongly visual features are anchored to their corresponding class text nodes within the parent cone, and the *parent–child separation margin* δ from Eq. 11, which enforces a minimum geodesic distance between parent and child text embeddings to prevent representation collapse. We evaluate on the CUB B0 Inc20 setting, sweeping λ_2 over $\{0.05, 0.10, 0.15, 0.20\}$ and δ over $\{0.10, 0.15, 0.20, 0.25, 0.30\}$. Figure 10a reports the average validation accuracy. The results show that HASTEN remains stable under different choices of λ_2 and δ.

Furthermore, to examine whether our framework is sensitive to geometric configurations across different semantic granularities, we study the foundational geometric parameters: the *curvature* c and the *cone half-aperture parameter* κ. The curvature controls the volume expansion rate of the Lorentz model, while κ determines the strictness of the partial-order entailment cones. We sweep $c \in \{0.5, 1.0, 1.5, 2.0\}$ and $\kappa \in \{0.05, 0.10, 0.15, 0.20\}$. Figure 10b reports the average validation accuracy, which remains tightly bounded between 86.04% and 86.22% across all configurations. These results indicate that HASTEN does not require exhaustive dataset-specific tuning and remains stable under a broad range of hyperparameter choices.



(a) Sensitivity to λ_2 and δ

(b) Sensitivity to c and κ

Method	Trainable Params
L2P	161,330
DualPrompt	333,412
CODA-Prompt	3,916,900
PROOF	1,310,720
RAPF	262,144
HASTEN	786,432

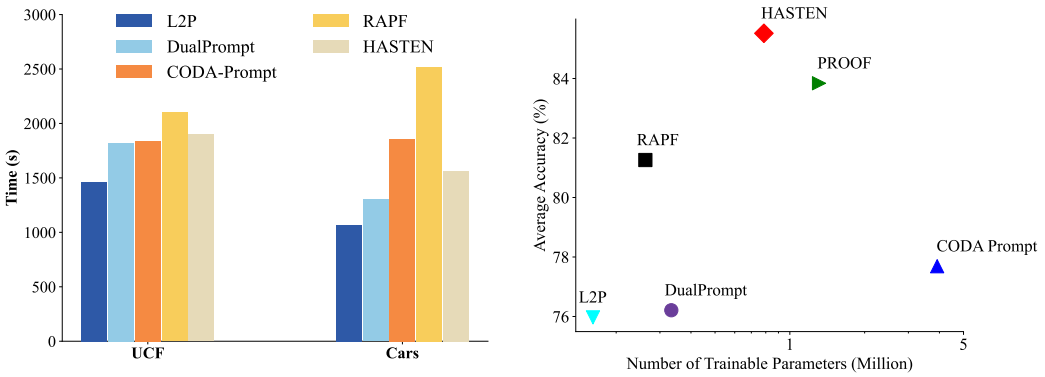
(c) Trainable parameters

Figure 10: Parameter sensitivity analysis on CUB B0 Inc20. **(a)** Impact of the text-to-image hierarchical loss weight λ_2 and the parent-child separation margin δ. **(b)** Impact of the foundational geometric priors: hyperbolic curvature c and cone aperture κ. **(c)** Number of trainable parameters on ImageNet-R B0 Inc20.

E Efficiency and Overhead Analysis

E.1 Trainable Parameter Analysis

In this paper, we design HASTEN by extending a hierarchy-aware module per task and incorporating a task-shared hyperbolic mapping layer. During inference, the features are aggregated into the final Euclidean feature. Specifically, when learning task b, we freeze the hierarchy-aware modules from all previous tasks $\{1, \ldots, b-1\}$ and only train the newly added module for task b. Therefore, the number of *active* (trainable) parameters at every stage remains constant: $2 \times d \times d$ for the current hierarchy-aware modules (H_i^b and H_t^b), plus $d \times d$ for the global hyperbolic mapping layer TP, yielding $3 \times d \times d$ in total. Moreover, as illustrated in Section 4.2 of the main paper, because all hierarchy-aware modules are linear layers, the frozen modules accumulated from previous tasks can be *reparameterized* by simply summing their weights, *i.e.*, $\sum_{p=1}^{b-1} H_i^p$ and $\sum_{p=1}^{b-1} H_t^p$, each collapsing into a single $d \times d$ matrix. This means the *stored* parameter size of the hierarchy-aware modules stays at $2 \times d \times d$ regardless of how many tasks have been learned, rather than growing linearly with the number of tasks. Combined with the $d \times d$ parameters of TP, the total parameter footprint of HASTEN is always $3 \times d \times d$. We further report the number of trainable parameters for representative methods in Figure 10c. In addition, we visualize the relationship between trainable parameter size and average accuracy in Figure 11b, showing that HASTEN achieves strong performance with a comparable parameter budget. As shown in the table, HASTEN has a comparable number of trainable parameters to other competitors, while achieving the best performance.



(a) Running time comparison. HASTEN achieves the best performance while maintaining a training time comparable to other compared methods.

(b) Comparison of accuracy and trainable parameters of different methods on ImageNet-R B0 Inc20.

Figure 11: Efficiency analysis. **(a)** Running time comparison across methods. **(b)** Trade-off between accuracy and the number of trainable parameters on ImageNet-R B0 Inc20.

Table 4: Overhead comparison on **CUB B0 Inc20**. All methods start from the same pre-trained CLIP backbone and are evaluated under the same hardware environment, using a **single RTX 3090 GPU** per method. "Total Time" denotes the end-to-end running time from the start of continual training to the completion of the final evaluation. "Peak GPU" denotes the peak training GPU memory of the current process, and "Infer." reports the average evaluation latency per test image aggregated over all sessions. Lower is better for all overhead metrics, while higher is better for $\bar{\mathcal{A}}$.

Method	$\bar{\mathcal{A}}$	Total Time (min)↓	Train Time (min)↓	Eval Time (min)↓	Peak GPU (GB)↓	Infer. (ms/img)↓
SimpleCIL [88]	83.81	**1.24**	**0.55**	**0.69**	**0.92**	**3.20**
DualPrompt [75]	69.49	61.77	59.82	1.95	6.95	9.06
CODA-Prompt [64]	73.27	66.43	64.26	2.17	15.53	10.09
L2P [76]	70.57	73.66	71.60	2.07	2.85	9.60
RAPF [30]	79.26	16.95	16.08	0.87	12.25	4.06
PROOF [90]	82.86	46.15	44.57	1.58	1.30	7.34
CLG-CBM [84]	79.33	35.44	34.75	0.70	1.17	3.23
HASTEN (Ours)	**86.18**	32.11	30.73	1.38	1.31	6.41

E.2 Running Time Comparison

In this section, we present a running time comparison of different methods. To ensure a fair evaluation, all methods are evaluated under the same experimental setting. Figure 11a reports the total running time on UCF B0 Inc10 and Cars B0 Inc10, including both training and inference/evaluation. As shown in the figure, HASTEN achieves the best performance while maintaining competitive running time compared with baselines such as CODA-Prompt and RAPF.

E.3 Overhead Analysis

Table 4 reports the computational overhead on CUB B0 Inc20. This supplementary efficiency study is conducted on a single NVIDIA RTX 3090 GPU for each method, while the main experiments follow the hardware setting described in the main paper. All methods use the same pre-trained CLIP backbone and are evaluated under the same hardware environment for a fair comparison. Among the compared methods, HASTEN achieves the best average accuracy (86.18%), while keeping the overall computational cost within a practical range.

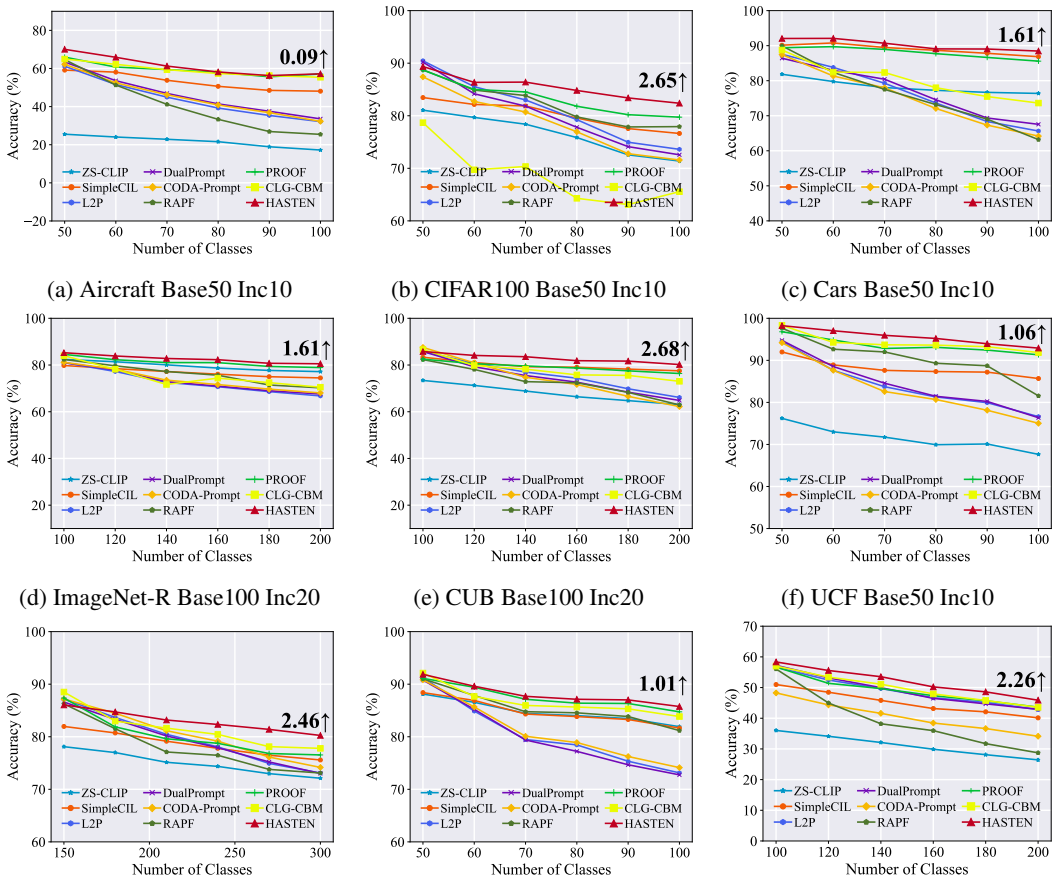
Total runtime. HASTEN completes the continual learning process in 32.11 minutes, including both training and evaluation. Although it is slower than lightweight baselines such as SimpleCIL (1.24 min) and RAPF (16.95 min), it is faster than CLG-CBM (35.44 min), PROOF (46.15 min), DualPrompt (61.77 min), CODA-Prompt (66.43 min), and L2P (73.66 min). This indicates that the hierarchy-aware training and TP stabilization introduce moderate training overhead while improving accuracy.

Inference cost. During evaluation, HASTEN requires 6.41 ms per image on average. This is higher than SimpleCIL (3.20 ms), CLG-CBM (3.23 ms), and RAPF (4.06 ms), but lower than PROOF (7.34 ms), DualPrompt (9.06 ms), L2P (9.60 ms), and CODA-Prompt (10.09 ms). Therefore, the additional hierarchy-aware mapping and hyperbolic logits introduce a moderate inference cost compared with lightweight baselines, while remaining efficient among several optimization-based CLIP CIL methods.

Memory usage. For peak training GPU memory, HASTEN uses 1.31 GB, which is close to PROOF (1.30 GB) and CLG-CBM (1.17 GB), and much lower than L2P (2.85 GB), DualPrompt (6.95 GB), RAPF (12.25 GB), and CODA-Prompt (15.53 GB). Although it uses slightly more memory than SimpleCIL (0.92 GB), its memory footprint remains lightweight among the compared methods.

Overall, these results indicate that the additional hierarchy-aware modeling in HASTEN does not introduce prohibitive computational cost, and the method maintains a practical efficiency profile while achieving stronger incremental learning performance.

Additional overhead. Beyond the CUB overhead table, we further examine the SVD cost in the B0 Inc10 settings used in Figure 11a. Each B0 Inc10 benchmark contains 10 incremental tasks, and HASTEN performs SVD only once after each task to refresh the projection subspace. For CLIP ViT-B/16, the pre-TP covariance matrix is 512×512. Benchmarking SVD on random covariance-like matrices of this size gives an average cost of 51.82 ± 4.41 ms per decomposition, resulting in about 0.52 s total SVD overhead. This accounts for only 0.027% of the total running time on UCF and



(a) Aircraft Base50 Inc10 (b) CIFAR100 Base50 Inc10 (c) Cars Base50 Inc10

(d) ImageNet-R Base100 Inc20 (e) CUB Base100 Inc20 (f) UCF Base50 Inc10

(g) SUN Base150 Inc30 (h) Food Base50 Inc10 (i) ObjectNet Base100 Inc20

Figure 12: Incremental performance of different methods on half-base setting. We report the performance gap after the last incremental stage of HASTEN and the runner-up method at the end of the line. All methods use the same CLIP pre-trained weights.

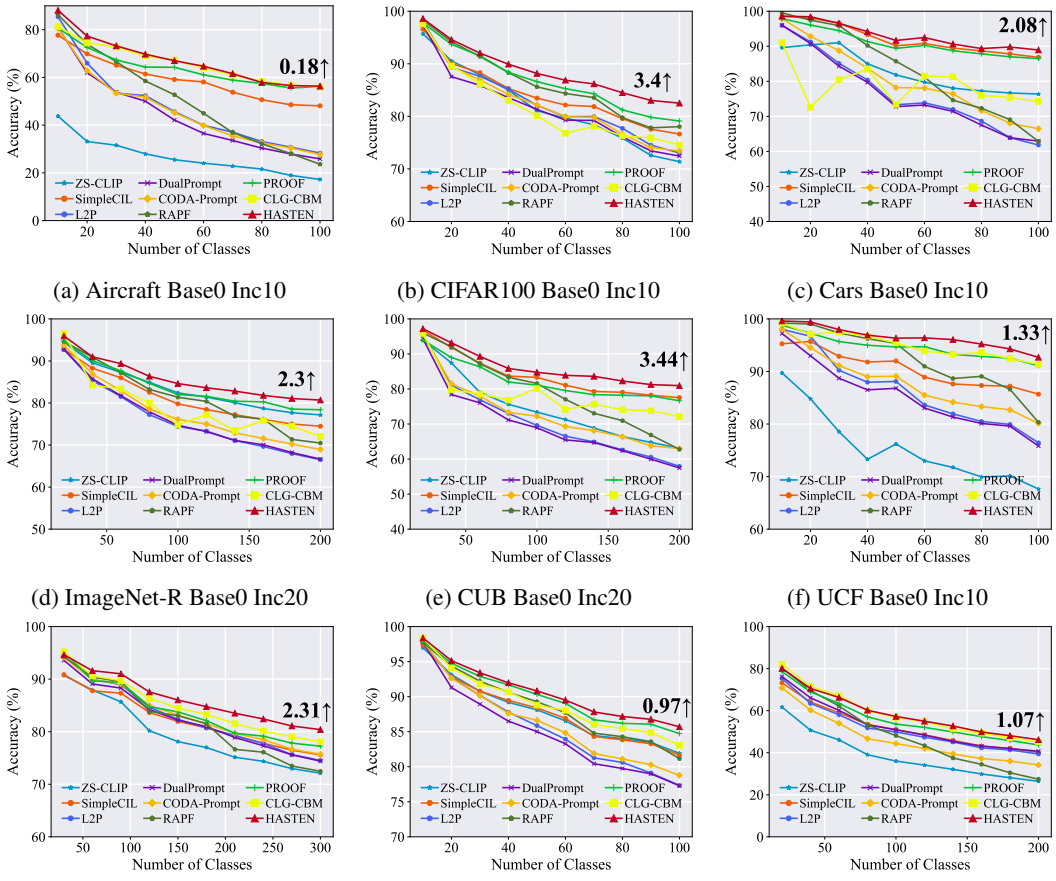
0.033% on Cars, indicating that the projection update is computationally negligible. Separately, for the given class order, the stage-wise semantic trees are constructed or expanded outside the training loop and used as fixed hierarchical supervision; thus, LLM querying introduces no training-loop or inference overhead.

F Full Results on Benchmarks

In this section, we provide the complete benchmark results for all compared methods. In addition to the complete quantitative comparison in Table 1, we present the full set of incremental learning curves. While the main paper reports three representative learning trends, here we include results for all benchmark settings. Specifically, Figure 13 shows the incremental performance under the B0 setting, and Figure 12 shows the results under the half-base setting. Across datasets and splits, HASTEN consistently outperforms competing approaches.

G Details of Image Traversals and Text Set Construction

In this section, we present additional image traversal examples on the UCF101 dataset and provide a detailed explanation of the traversal protocol and the text sets used in these visualizations. All traversals are computed on images randomly sampled from the UCF101 test split, with qualitative results shown in Figure 14.



(a) Aircraft Base0 Inc10 (b) CIFAR100 Base0 Inc10 (c) Cars Base0 Inc10

(d) ImageNet-R Base0 Inc20 (e) CUB Base0 Inc20 (f) UCF Base0 Inc10

(g) SUN Base0 Inc30 (h) Food Base0 Inc10 (i) ObjectNet Base0 Inc20

Figure 13: Incremental performance of different methods on B0 setting. We report the performance gap after the last incremental stage of HASTEN and the runner-up method at the end of the line. All methods use the same CLIP pre-trained weights.

Text Set Construction: To properly evaluate the semantic transition from specific to generic concepts, we construct a candidate text pool. For each UCF101 class, we generate a subset of text that includes the original class name, along with short descriptions and related concepts generated by GPT-5. These textual descriptions are intentionally curated to span multiple levels of semantic granularity—ranging from highly specific visual actions to broader categorical terms. The global text pool used for the retrieval during traversal is the union of the text subsets from all classes, supplemented by the universal root node ("entity" for HASTEN, and `[ROOT]` for CLIP).

Traversal and Retrieval Protocol: As described in the main paper, we build an image-to-root traversal in each model's feature geometry: the Lorentz manifold $\mathcal{B}^d$ for HASTEN and the Euclidean unit hypersphere for CLIP. We uniformly sample 100 points along the trajectory. At each sampled point $\mathbf{l}_t$, we retrieve one text from a hierarchical text set $\mathcal{T}$ (including `[ROOT]`). For HASTEN, retrieval is structure-constrained. A text node is a valid ancestor candidate only if it satisfies the entailment-cone constraint:

$$\mathcal{P}(\mathbf{l}_t) = \{\mathbf{p} \in \mathcal{T} \mid \mathrm{ext}(\mathbf{p}, \mathbf{l}_t) \leq \mathrm{aper}(\mathbf{p})\}\,. \tag{47}$$

We then select the nearest valid ancestor in hyperbolic space:

$$\mathbf{p}_t^* = \arg\min_{\mathbf{p} \in \mathcal{P}(\mathbf{l}_t)} d_{\mathcal{B}}(\mathbf{l}_t, \mathbf{p}). \tag{48}$$

For CLIP, no structural constraint is used; we directly retrieve

$$\mathbf{u}_t^* = \arg\max_{\mathbf{u} \in \mathcal{T}} \cos(\mathbf{l}_t, \mathbf{u}). \tag{49}$$



Figure 14: Image traversals with HASTEN and CLIP: HASTEN's path through hyperbolic space reveals a smooth transition from specific to generic text descriptions, reflecting a more systematic and fine-grained visual-textual semantic hierarchy than CLIP.

Therefore, HASTEN performs geometry-aware parent retrieval, while CLIP performs unconstrained nearest-text retrieval.

Analysis of the Results: The retrieved sequences show how semantics change as an embedding moves toward the root. In HASTEN, image and text embeddings are organized by parent-centered entailment cones, so each sampled point is matched only to entailment-valid ancestors. This geometry enforces a partial order and yields a gradual shift from specific to increasingly generic concepts. In CLIP, retrieval is unconstrained: points are matched only by cosine similarity in a flat Euclidean space. Because CLIP is trained for global image-text alignment rather than ancestor-descendant ordering, intermediate points often remain closest to the original leaf-level concept and switch only near a decision boundary (often close to [ROOT]). Thus, the downward arrows (↓) indicate repeated nearest-neighbor outcomes over many consecutive samples, rather than stable intermediate ancestors. This exposes its lack of inherent hierarchical properties, a semantic gap that HASTEN effectively resolves via its hierarchy-aware hyperbolic representation.

Below, we provide two concrete examples illustrating how the class-specific text subsets are generated via GPT-5:

Hammer Throw:

> **Q**: Could you analyze "Hammer Throw" by providing a series of descriptions that start as very specific and become progressively more general? Alongside this,

please also provide a list of related adjectives, similarly ordered from specific to general.
A: An athlete spinning within a circle and releasing the hammer for maximum distance. An athlete accelerating down the runway and launching in a field event. A person practicing a sport outdoors. A person performing a physical activity. Adjectives: powerful, explosive, technical, coordinated, solo, athletic, outdoor.

Knitting:

Q: Could you analyze "Knitting" by providing a series of descriptions that start as very specific and become progressively more general? Alongside this, please also provide a list of related adjectives, similarly ordered from specific to general.
A: Hands looping yarn with knitting needles to form a patterned fabric. Hands crafting with yarn, fabric, or small tools. Routine indoor activity using common tools. A person engaged in everyday human activity. Adjectives: precise, controlled, rhythmic, handheld, quiet, indoor, recreational.

H Quantitative Verification of Hierarchical Structure

To quantitatively verify that the learned hyperbolic embeddings faithfully capture the intended hierarchical structure, we analyze the geometric properties of the representations on the test sets of CIFAR-100 (B0 Inc10), CUB-200 (B0 Inc20), and UCF-101 (B0 Inc10). Crucially, all features are extracted after the model has completed the final incremental learning stage, demonstrating HASTEN's robustness against catastrophic forgetting across the entire task sequence.

Following the Lorentz model adopted in Section 4.2, we define the *radius* of an embedding $\mathbf{x} \in \mathcal{B}^d$ as its geodesic distance to the origin $\mathbf{o} = [\mathbf{0}, 1]$ via Eq. 4, *i.e.*, $r(\mathbf{x}) = d_{\mathcal{B}}(\mathbf{x}, \mathbf{o})$. In a well-structured hyperbolic hierarchy, the entailment cone mechanism constrains each child to lie within the cone of its parent, which geometrically pushes children to higher-radius regions. Consequently, concepts should expand outward radially as they become more specific, and semantically related nodes should cluster tightly.

As reported in Table 5, HASTEN strictly preserves these geometric expectations across all datasets, even after the full sequence of incremental updates:

1. **Radial Expansion by Specificity**: The average radius increases with tree depth across all datasets. For instance, on the deep CUB-200 hierarchy the radius expands monotonically from 0.10 at Depth 1 to 0.83 at Depth 5; UCF-101 smoothly expands from 0.14 at Depth 1 to 0.78 at Depth 6. Furthermore, decomposing the radius by node type reveals a consistent ordering: Virtual parent nodes (abstract categories) reside closest to the origin, Real leaf-class prototypes lie at intermediate radii, and Image-level sample embeddings are mapped nearest to the boundary (*Virtual* $<$ *Real* $<$ *Image* across all datasets). This directly reflects the entailment cone geometry, where the cone aperture $\mathrm{aper}(\mathbf{x})$ shrinks with increasing $\|\mathbf{x}\|$, causing deeper and more specific nodes to occupy progressively higher-radius regions of $\mathcal{B}^d$.
2. **Semantic Topology Preservation**: Beyond the radial ordering, we verify structural integrity by comparing the average pairwise geodesic distance between sibling nodes (sharing the same parent in the semantic tree) and non-relative nodes. Across all datasets, siblings are consistently embedded closer together than non-relatives (e.g., 0.27 vs. 0.32 on CIFAR-100; 0.42 vs. 0.47 on UCF-101), confirming that the learned Lorentz embedding preserves the semantic topology throughout the incremental learning process.

These results confirm that HASTEN effectively translates the discrete partial-order semantic tree into a continuous, geometrically faithful hierarchical representation, and successfully maintains this structure across sequential tasks without collapsing.

I Broader Impacts

HASTEN studies how hierarchical information can be used to improve class-incremental learning, enabling models to learn new classes while better preserving previously learned knowledge. Since the method is exemplar-free, it avoids storing past training samples, which may benefit privacy-sensitive or storage-constrained continual learning scenarios. By using semantic hierarchies as structured

Table 5: Quantitative hierarchy verification computed after the final incremental task. The radius is defined as the geodesic distance to the Lorentz origin $\mathbf{o} = [\mathbf{0}, 1]$ via Eq. 4. Structural distances compare the pairwise geodesic proximity of sibling nodes versus non-relative nodes. (A dash (–) indicates that the dataset tree does not reach this depth.)

Dataset	**Average Radius by Tree Depth**						**Average Radius by Node Type**			**Structural Distance**	
	D 1	D 2	D 3	D 4	D 5	D 6	Virtual	Real	Image	Sibling	Non-Rel
CIFAR-100	0.13	0.29	0.29	0.67	0.68	–	0.26	0.30	0.68	0.27	0.32
CUB-200	0.10	0.12	0.48	0.80	0.83	–	0.32	0.53	0.82	0.51	0.54
UCF-101	0.14	0.17	0.35	0.72	0.75	0.78	0.25	0.44	0.75	0.42	0.47

supervision, HASTEN provides a way to incorporate class relationships into continual learning without requiring manually stored exemplars. The main practical consideration is that the semantic hierarchy should be reliable for the target domain, since inaccurate hierarchy construction may affect downstream predictions. In real deployments, this can be mitigated by validating the generated hierarchy, using curated domain taxonomies when available, or incorporating light human review for high-stakes applications.

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: We state the contributions and scope in the abstract and introduction, and discuss limitations in Section 6.

   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: We included it in the paper.

   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: We state a standard geometric property of Lorentz hyperbolic space and provide its derivation in Appendix A.2 to justify its use for hierarchical representation.

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: We provide all the necessary details to reproduce the results. Detailed algorithmic descriptions are provided in Appendix A.1.

   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: [Yes]

Justification: The source code is publicly available at: `https://github.com/LAMDA-CL/NeurIPS2026-HASTEN`.

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: The experiment details are available in the paper.

   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: We report mean accuracy with one-standard-deviation bands over multiple random class-order splits in the appendix. The variability reflects different independent runs under different class orders.

   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: The information of computer resources is provided.

   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 research conforms to the NeurIPS Code of Ethics.

   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: [Yes]

    Justification: We discuss both positive and negative societal impacts of the work in the appendix.

    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: No data or models with a high risk of misuse are released.

    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: All the datasets used are publicly available, discussed, and properly cited in Section 5 and Appendix C.8.

    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 datasets, models, or benchmarks.

    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 human-subject research.

    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: [Yes]

    Justification: We describe the use of LLMs for semantic tree construction in Section 4.1 and Appendix B.

    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.