

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 multi-modal 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 entails its images. But existing CLIP-based CIL methods fail to explicitly capture this inherent hierarchy, leading to fine-grained class features drift during incremental updates and ultimately to catastrophic forgetting. To address this challenge, we propose HASTEN (Hierarchical Semantic Tree Anchoring) that anchors hierarchical information into CIL to reduce catastrophic forgetting. First, we employ an external knowledge graph as supervision to embed visual and textual features in hyperbolic space, effectively preserving hierarchical structure as data evolves. Second, to mitigate catastrophic forgetting, we project gradients onto the null space of the shared hyperbolic mapper, preventing interference with prior tasks. These two steps work synergistically to enable the model to resist forgetting by maintaining hierarchical relationships. Extensive experiments show that HASTEN consistently outperforms existing methods while providing a unified structured representation.

1. Introduction

Recent advancements in deep learning [12, 22, 23, 25, 36] 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, 37]. Class-Incremental Learning (CIL) [43, 51, 75] enables incremental learning of new classes without forgetting earlier knowledge, yet suffers from catastrophic forgetting [19, 49, 53, 54], where

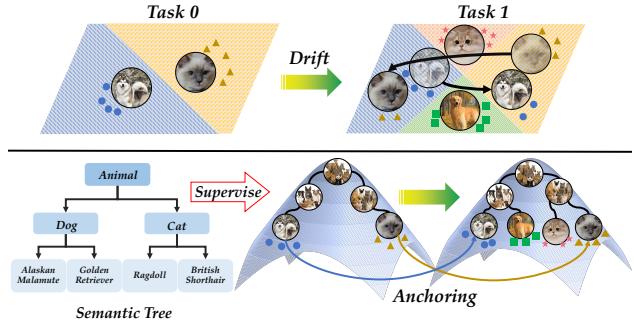


Figure 1. Effect of feature hierarchy. **Top:** Without hierarchy, fine-grained features will drift as new classes emerge. **Bottom:** With a hierarchical constraint, features stay anchored and relations can be preserved in the incremental learning process.

new tasks interfere with old ones. Recent pre-trained models (PTMs) like CLIP [48] offer generalizable, transferable features from large-scale pre-training [10, 27, 79, 80], 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 [14, 35, 64, 71] by freezing the CLIP backbone to preserve general knowledge while adapting lightweight components (e.g., adapters [27, 77] or prompts [67, 78]). 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, 21]. Its uniform metric treats all points alike, so placing a generic concept 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.

Correspondingly, in incremental learning, such hierarchical characteristics are beneficial for addressing catastrophic forgetting. As illustrated in Figure 1 (Top),

without hierarchy, fine-grained class features (*e.g.*, specific dog and cat breeds) gradually drift [73] across incremental tasks. As new classes emerge, previous classes 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, keeping them stable without drift across continuous updates.

These observations motivate a stronger inductive bias toward hierarchy. In the absence of hierarchical constraints, features of early classes lack effective protection and are gradually eroded by training on new classes. An ideal incremental learner should maintain a latent hierarchy, provide stable anchors for parent–child relations, and preserve previously learned decision structure while remaining plastic to new classes. It should keep siblings compact yet separable, avoid crowding broad concepts with specific ones, and ensure that updates respect the existing hierarchical organization. With these properties, the model can retain accumulated knowledge and use the hierarchy to stabilize features, thereby mitigating catastrophic forgetting.

To address this challenge, we propose HASTEN (Hierarchical Semantic TreE Anchoring), a novel framework that preserves hierarchical semantic structures for effective continual learning. Guided by the insight that preserving hierarchical relationships is key to mitigating feature drift, our approach proceeds in two steps. First, for each task, we train two hierarchy-aware modules to learn relations among texts and between images and texts, supervised by an external hierarchical semantic tree. A globally shared hyperbolic mapper then embeds these enhanced features in hyperbolic space, which naturally represents hierarchies. Second, because this mapper is updated across tasks, we project each parameter update onto the null space of features from previous tasks. This projection preserves the mapper’s outputs on old tasks and enables the accumulation of task-specific mapping patterns. Together, these components preserve hierarchical structures, keep features stable without drift, and ultimately alleviate catastrophic forgetting.

2. Related Work

Class-Incremental Learning (CIL): CIL enables models to incrementally learn new classes without forgetting previously acquired knowledge [11, 43], and traditional approaches can be broadly categorized into several groups. Replay-based methods preserve past knowledge by storing and revisiting samples from previous tasks [7, 8, 41, 51], though this may come at the cost of memory usage and data privacy. Regularization-based methods estimate parameter importance to penalize critical updates [2, 3, 32, 74] or use knowledge distillation to align outputs at logit [38, 51], feature [26, 40, 46], or group level [15, 17, 20]. Parameter-isolation methods assign task-specific parameters via net-

work expansion [62, 63, 70, 76] or masking, effectively preventing interference but slightly increasing model complexity. Bias rectification methods address prediction and classifier biases [55, 68, 75], while model expansion dynamically grows capacity through neuron, backbone, or adapter extension [18], thereby integrating new knowledge without overwriting prior information.

CIL with Pre-Trained Models: The paradigm of CIL has been revitalized by powerful pre-trained models (PTMs) like Vision Transformers [16] and CLIP [48], which offer strong, generalizable representations [67], and a prevalent strategy is to freeze the PTM backbone and introduce lightweight, learnable modules for adaptation. For vision-centric PTMs, research has focused on prompt-based learning, which steers model behavior by learning visual prompts [31, 56, 67], and adapter-based tuning, where small, trainable modules are inserted into the frozen network [9, 50, 72]. A distinct approach involves prototype-based methods that build classifiers directly in embedding space by matching features to class prototypes [44, 57, 77]. Building on this, a significant trend leverages vision-language models like CLIP, exploiting their cross-modal alignment. This is achieved through various means, including learning multi-modal or textual prompts [65, 66, 80], or adapting the architecture with new projection heads as in PROOF [78]. Parameter-efficient tuning is also applied, with methods such as RAPF [27] using modular adapters to balance stability and plasticity.

3. Preliminaries

3.1. Class-Incremental Learning

Class-Incremental Learning (CIL) aims to build a classifier that learns new classes from sequential tasks without forgetting previously learned ones [51]. We consider a sequence of tasks $\{\mathcal{D}^1, \mathcal{D}^2, \dots, \mathcal{D}^B\}$, 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.*, $Y_b \cap Y_{b'} = \emptyset$ for any $b \neq b'$. The exemplar-free CIL setting [67, 81] 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 $\mathcal{Y}_b = Y_1 \cup \dots \cup Y_b$ and minimizes the empirical risk:

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

where \mathcal{H} is the hypothesis space, $\mathbb{I}(\cdot)$ is the indicator function. In this paper, following recent work [27, 72, 78], we build our model $f(\mathbf{I})$ upon a pre-trained CLIP model [48]. 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 [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 :

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

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

3.2. 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 [27, 78]. In this setup, the adapted image feature is represented as:

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

where $\mathbf{e}_v = g_v(\mathbf{I})$ is the frozen visual embedding. This adapted feature \mathbf{z} 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.

Discussions: 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 [73], 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.

4. HASTEN: Hierarchical Semantic Tree Anchoring

Motivated by the feature drift in CLIP-based CIL, we propose HASTEN, a framework that integrates hierarchical structure modeling into incremental learning to organize visual and textual features, thus mitigating catastrophic forgetting. The core design of HASTEN has three components. First, a GPT-5-generated [45] hierarchical semantic tree provides structural supervision. Second, task-specific hierarchical perception modules, together with a global hyperbolic mapper, anchor hierarchical knowledge and embed features in hyperbolic space, thereby counteracting feature drift. Moreover, to mitigate forgetting, we project gradient updates onto a null space from prior-task features, preserving old-task outputs while learning new mappings.

4.1. Hierarchical Semantic Tree Construction

In the real world, classes exhibit intrinsic hierarchical relationships, *i.e.*, broader concepts subsume fine-grained subclasses, with images serving as leaf nodes. To explicitly model this structure while reducing manual hierarchical annotation, we use GPT-5 to generate a tree-structured semantic tree for every dataset. Firstly, a top-level virtual class “entity” acts as the root to encompass all classes. Next, we construct a hierarchical path from each dataset leaf class to the root “entity,” which forms the backbone of the semantic tree. Finally, for classes lacking direct parent–child connections, we introduce abstract virtual classes that are not present in the original dataset to complete the hierarchy. We refer to dataset-native classes as real classes to distinguish them from GPT-5-generated virtual ones. This process yields a dataset-specific hierarchical semantic tree that guides subsequent feature learning. Specifically, we utilize GPT-5 to generate a hierarchical semantic tree by:

Q: You are a precise taxonomist. Given a list of class names, build a complete *is-a* hierarchy with a single root “entity”. You may introduce abstract parent classes that are not included in the input list as needed to create a coherent hierarchy. Input list: [“alaskan malamute”, “ragdoll”, “golden retriever”, “british shorthair”]

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

In this way, we obtain a dataset conditioned, model-agnostic hierarchical supervision signal, denoted as semantic tree $\mathcal{K} = (\mathcal{C}, \mathcal{E}, r)$. We will use it to model class hierarchy. Here $\mathcal{C} = \mathcal{C}_{\text{real}} \cup \mathcal{C}_{\text{virt}}$ is the set of classes with $\mathcal{C}_{\text{real}}$ real classes and $\mathcal{C}_{\text{virt}}$ virtual classes, $r = \text{“entity”}$ is the root, and $\mathcal{E} \subseteq \mathcal{C} \times \mathcal{C}$ contains only directed parent→child edges. This supervision specifies text-to-text hierarchy, which we will use 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 [13, 52], 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

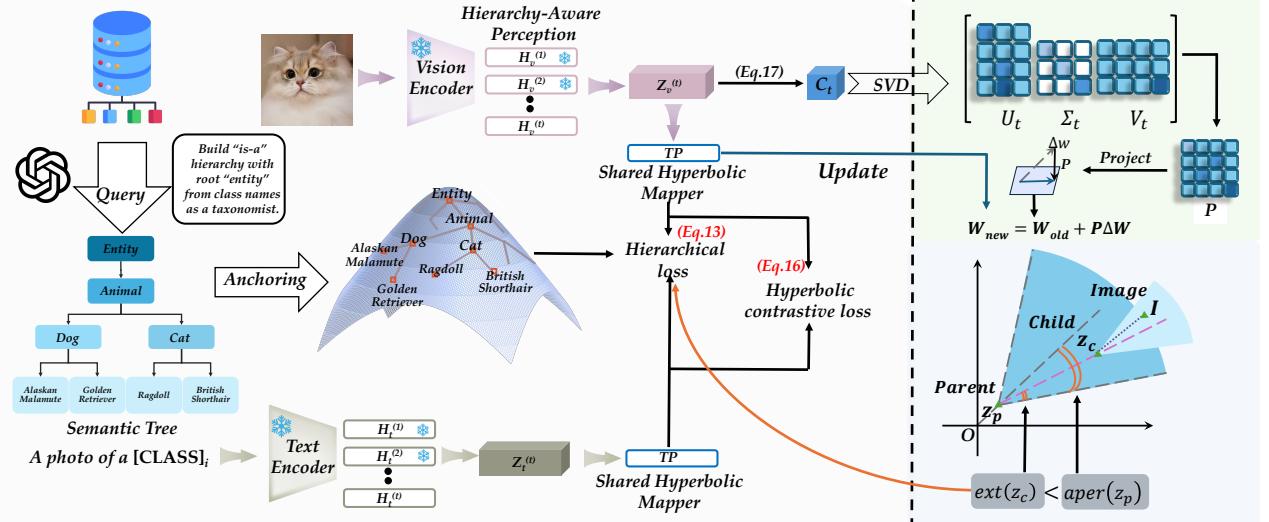


Figure 2. Illustration of HASTEN. **Left:** Hierarchical semantic tree building and hyperbolic projection. We use GPT-5 to generate a tree-structured semantic hierarchy and design task-specific hierarchical perception modules to meet downstream knowledge requirements. Euclidean features are projected 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}^d . This loss pushes the embedding \mathbf{z}_c within an imaginary cone projected by its paired parent embedding \mathbf{z}_p .

distance $d_{\mathcal{B}}(\mathbf{p}, \mathbf{q})$ in \mathcal{B}^d is:

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

Here, $c > 0$ denotes the curvature, which is set to 1 for stability. Negative curvature enables exponential volume growth and a depth-radius correspondence, causing distances to expand across hierarchy levels while remaining relatively tight among siblings, which naturally aligns with tree-structured data. In contrast, Euclidean volume grows only polynomially, leading to crowding when mapping many specific concepts near a generic one. In our model, $d_{\mathcal{B}}$ serves as the task metric for similarity, margins, and hierarchy constraints. We can map Euclidean (tangent) vector \mathbf{v} to the hyperbolic manifold using the exponential map at the origin $\mathbf{o} = [\mathbf{0}, 1]$:

$$\text{exp}_{\mathbf{o}}(\mathbf{v}) = \frac{\sinh(\|\mathbf{v}\|)}{\|\mathbf{v}\|} \mathbf{v}. \quad (5)$$

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 $\{H_v^1, \dots, H_v^{b-1}\}$ and $\{H_t^1, \dots, H_t^{b-1}\}$ 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:

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

This placement and aggregation effectively 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, $\text{TP}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$. 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 $\mathbf{z}_s = \text{TP}(\tilde{\mathbf{z}})$. The resulting hyperbolic representations used for training are:

$$\mathbf{z}_v = \text{TP}(\tilde{\mathbf{z}}_v), \quad \mathbf{z}_t = \text{TP}(\tilde{\mathbf{z}}_t). \quad (7)$$

To balance projection fidelity and adaptation flexibility, we regularize TP toward the exponential map at the origin:

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

Overall, the task-shared TP maps aggregated module outputs into a common hyperbolic space, yielding unified 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: $\text{aper}(\mathbf{p}) = \sin^{-1}\left(\frac{2\kappa}{\sqrt{c}\|\mathbf{p}\|}\right)$, where

$\kappa = 0.1$ unless noted. Deeper parents (larger $\|\mathbf{p}\|$) induce tighter cones, which aligns with increasing semantic specificity. The exterior angle $\text{ext}(\mathbf{p}, \mathbf{q})$ measures the child \mathbf{q} 's deviation from the cone boundary centered at parent \mathbf{p} :

$$\text{ext}(\mathbf{p}, \mathbf{q}) = \cos^{-1} \left(\frac{q_t + p_t (c \langle \mathbf{p}, \mathbf{q} \rangle_B)}{\|\mathbf{p}_s\| \sqrt{(c \langle \mathbf{p}, \mathbf{q} \rangle_B)^2 - 1}} \right), \quad (9)$$

where $\mathbf{p}, \mathbf{q} \in \mathcal{B}^d$. To ensure the learned hyperbolic space faithfully 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 $\text{ext}(\mathbf{p}, \mathbf{q}) \leq \text{aper}(\mathbf{p})$. As visualized in Figure 2(Bottom-Right), this single constraint inherently places child nodes radially farther from the origin than their parents, thus preserving the desired hierarchical structure. We penalize the violations by the entailment loss:

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

with the soft-plus operation $\text{SP}(z) = \log(1 + e^z)$. In practice, we apply $\mathcal{L}_{\text{entail}}$ 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 Loss: Building on the entailment cones, we supervise both the text–text hierarchy and the text–image anchoring with two complementary terms. The Text-to-Text Hierarchical Loss for a parent-child edge $(p, c) \in \mathcal{E}$ is:

$$\mathcal{L}_{\text{hier}}^{\text{txt} \rightarrow \text{txt}} = \mathcal{L}_{\text{entail}}(\mathbf{z}_t^p, \mathbf{z}_t^c) + \text{SP}(\delta - d_{\mathcal{B}}(\mathbf{z}_t^p, \mathbf{z}_t^c)). \quad (11)$$

In Eq. (11), the second term enforces a minimum parent-child separation δ to avoid embedding collapse. The Text-to-Image Hierarchical Loss aligns the visual embedding \mathbf{z}_v with its corresponding class text node \mathbf{z}_t^c :

$$\mathcal{L}_{\text{hier}}^{\text{txt} \rightarrow \text{img}} = \mathcal{L}_{\text{entail}}(\mathbf{z}_t^c, \mathbf{z}_v). \quad (12)$$

We then combine Eq. (11) and Eq. (12) to build the overall hierarchical objective considering both modalities:

$$\mathcal{L}_{\text{hier}} = \lambda_1 \mathcal{L}_{\text{hier}}^{\text{txt} \rightarrow \text{txt}} + \lambda_2 \mathcal{L}_{\text{hier}}^{\text{txt} \rightarrow \text{img}}. \quad (13)$$

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

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 symmetrical contrastive loss framework (similar to the multi-class N-pair loss [48, 58]).

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:

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

$$P_j^{\text{T}} = \frac{\exp(-d_{\mathcal{B}}(\mathbf{z}_{t,j}, \mathbf{z}_{v,j})/\tau)}{\sum_{i=1}^N \exp(-d_{\mathcal{B}}(\mathbf{z}_{t,j}, \mathbf{z}_{v,i})/\tau)}. \quad (15)$$

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

$$\mathcal{L}_{\text{contrast}} = -\frac{1}{2} (\log(P_j^{\text{I}}) + \log(P_j^{\text{T}})). \quad (16)$$

Eq. (16) pulls matched pairs together and separates mismatches in \mathcal{B}^d , thereby stabilizing cross-task features and reinforcing cross-modal clusters.

Hierarchical Anchoring: To anchor the hierarchy in the hyperbolic space, we utilize virtual-class anchoring. Following 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 cache, and calculate the textual hierarchy loss against this union, thereby anchoring new classes to their parents. Meanwhile, to represent past classes k , we follow [81] to draw features from $\mathcal{N}(\mu_k, \Sigma_k)$ (derived from the frozen encoder g_v) into the hyperbolic space. This process creates class-level surrogates that mitigate cross-modal feature drift during training.

4.3. Gradient Projection for TP

Since TP is shared across tasks and modalities, its updates can overwrite prior mappings. To mitigate forgetting, we restrict them to directions orthogonal to the span of past features. During task b , we maintain an uncentered covariance $C^{(b)}$ of the Euclidean features before TP, as:

$$C^{(b)} = \frac{\alpha_{b-1} C^{(b-1)} + N_b C_{\text{new}}^{(b)}}{\alpha_b}, \quad \alpha_i = \sum_{j=1}^i N_j, \quad (17)$$

where N_j is the number of samples in task j . We then compute an SVD for $C^{(b)}$:

$$C^{(b)} = U \Sigma V^{\top}, \quad \Sigma = \text{diag}(\sigma_1 \geq \dots \geq \sigma_d). \quad (18)$$

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 choice that *approximates* the null space rather than fixing a rank. We define the orthogonal complement as $V_{\perp} = V[:, r : d]$. We then project the raw gradient Δw of TP onto this approximate null space:

$$\Delta w_{\text{proj}} = V_{\perp} V_{\perp}^{\top} \Delta w. \quad (19)$$

Since $C^{(b)} \propto X^{\top} X$ for the past features X , the columns of V_{\perp} span directions with near-zero variance, hence $X^o V_{\perp} \approx 0$ for old-task features X^o . Therefore, we have:

$$X^o \Delta w_{\text{proj}} = X^o V_{\perp} V_{\perp}^{\top} \Delta w = (X^o V_{\perp})(V_{\perp}^{\top} \Delta w) \approx 0,$$

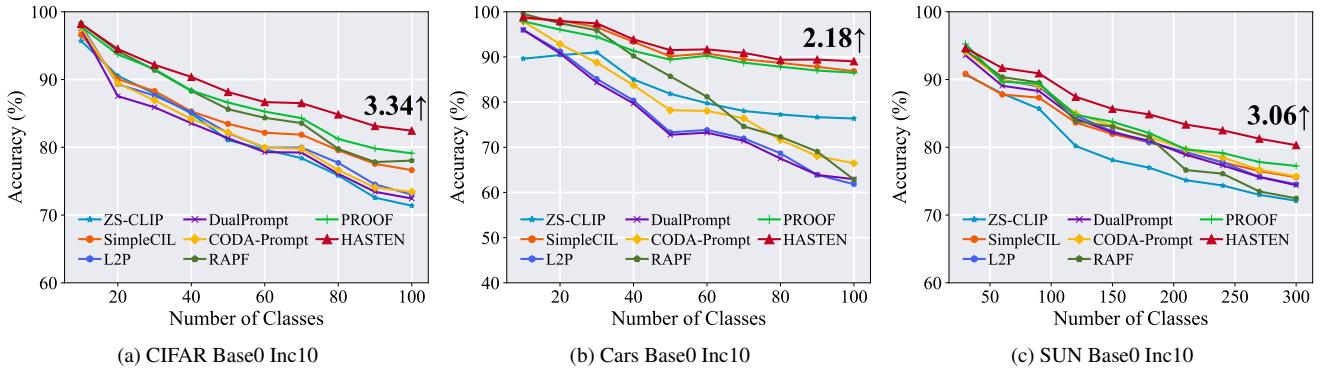


Figure 3. Incremental performance of different methods. We report the performance gap after the last incremental stage of HASTEN and the runner-up method at the end of the line. All methods utilize the same CLIP pre-trained weight. More results are in the supplementary.

which shows that updates to TP preserve old predictions while still enabling learning in new directions.

Discussions: Projecting gradients onto the approximate null space reduces interference from past tasks while keeping TP plastic for new ones. Specifically, the parameter update becomes $\theta_{\text{TP}} \leftarrow \theta_{\text{TP}} - \eta(V_{\perp} V_{\perp}^T g)$. The energy threshold that selects r tunes the stability–plasticity trade-off, where larger r protects more old variance while smaller r adapts faster. We compute $C^{(b)}$ from pre-TP Euclidean features and update V_{\perp} periodically to control cost. This projection prevents new updates from disturbing previously mapped regions.

4.4. 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 anchor gradient update to preserve knowledge. The training objective is defined by combining Eq. (13), Eq. (16), and Eq. (8):

$$\mathcal{L} = \mathcal{L}_{\text{hier}} + \mathcal{L}_{\text{contrast}} + \beta \mathcal{L}_{tp}. \quad (20)$$

During inference, we enhance robustness by ensembling predictions from two distinct feature spaces. The first component, the original space probability $P_{\text{orig},k}(\mathbf{x})$, 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 $P_{\text{hyp},k}(\mathbf{x})$, leverages the learned hierarchy, using the negative hyperbolic distance $-d_B(\mathbf{z}_v, \mathbf{z}_t^c)$ 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:

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

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. We report incremental

learning curves, ablations on hierarchical tree anchoring, key hyperparameters, and t-SNE visualizations comparing embeddings with vs. without anchoring. We also visualize image-to-root hierarchy traversals, learned embeddings, and logit shifts after anchoring to illustrate the effect.

5.1. Implementation Details

Dataset: We adopt the evaluation protocol from [67, 78, 80] and evaluate on nine benchmarks that exhibit domain shifts from CLIP’s pre-training data: CIFAR100 [34], CUB200 [61], ObjectNet [5], ImageNet-R [24], FGVCAircraft [42], StanfordCars [33], Food101 [6], SUN397 [69], and UCF101 [59]. Following [78], 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. Additional details are provided in the supplementary material.

Dataset split: Following [51, 67, 78], 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 [51], we randomly shuffle the class order with seed 1993, which is kept consistent across all compared methods.

Comparison methods: We compare our method with SOTA pre-trained model-based CIL algorithms, such as L2P [67], DualPrompt [66], CODA-Prompt [56], and SimpleCIL [77]. We also include SOTA CLIP-based CIL approaches, including PROOF [78], RAPF [27], and MG-CLIP [28]. 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 [47]. Following [27, 78], we evaluate CLIP ViT-B/16 pre-trained on LAION-400M [29] 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 AdamW [39] 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 $\lambda_1 = 0.5$

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										
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 [77]	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 [48]	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 [67]	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 [66]	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 [56]	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 [78]	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 [27]	50.38	23.61	40.47	25.44	86.14	78.04	82.17	77.93	82.89	62.85	75.87	63.19
MG-CLIP [28]	48.33	32.34	26.28	13.02	89.74	82.78	85.62	81.26	88.21	79.73	84.58	79.62
HASTEN	67.20	57.16	61.66	57.13	88.70	82.45	85.50	82.56	93.00	89.03	90.20	88.49
Method	ImageNet-R				CUB				UCF			
	B0 Inc20		B100 Inc20		B0 Inc20		B100 Inc20		B0 Inc10		B50 Inc10	
	$\bar{\mathcal{A}}$	\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 [77]	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 [48]	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 [67]	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 [66]	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 [56]	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 [78]	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 [27]	81.26	70.48	76.10	70.23	79.09	62.77	72.82	62.93	92.28	80.33	90.31	81.55
MG-CLIP [28]	83.18	77.20	50.47	35.37	74.20	64.25	53.47	34.78	87.74	80.83	75.45	59.42
HASTEN	85.52	80.23	82.52	80.23	86.06	80.20	82.44	80.24	96.08	92.61	95.30	92.88
Method	SUN				Food				ObjectNet			
	B0 Inc30		B150 Inc30		B0 Inc10		B50 Inc10		B0 Inc20		B100 Inc20	
	$\bar{\mathcal{A}}$	\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 [77]	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 [48]	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 [67]	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 [66]	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 [56]	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 [78]	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 [27]	82.13	72.47	78.04	73.10	88.57	81.15	85.53	81.17	48.67	27.43	39.28	28.73
MG-CLIP [28]	53.71	41.62	26.58	12.64	88.59	82.35	28.86	12.51	51.41	40.90	25.00	12.39
HASTEN	86.27	80.31	83.06	80.29	90.65	85.93	88.16	85.79	58.40	46.03	52.04	45.96

and $\lambda_2 = 0.1$ for the textual and image to text hierarchical losses, $\beta = 0.1$ for the TP regularizer, and $\delta = 0.1$. The hierarchical semantic tree is generated using GPT-5 [45], and we report the full prompt in the supplementary.

Evaluation metric: Following [51, 78], 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, and the results are reported in Table 1 and Figure 3. As shown, HASTEN delivers state-of-the-art performance, outperforming all prior methods on 8 out of the nine evaluated benchmark settings. The sole exception is CIFAR-100, where MG-CLIP performs best. However,

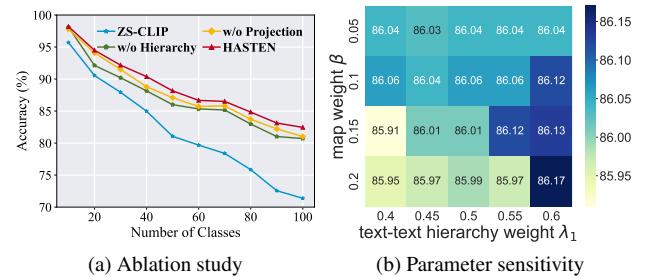


Figure 4. Ablation study and parameter sensitivity analysis.

HASTEN’s superior results across the other diverse datasets highlight its consistently robust performance and broader applicability. Finetuning performs the worst, indicating severe catastrophic forgetting. Visual prompt methods fall behind as they fail to leverage textual information. Compared to other strong CLIP-based methods like RAPF and PROOF, HASTEN’s significant gains underscore its robust

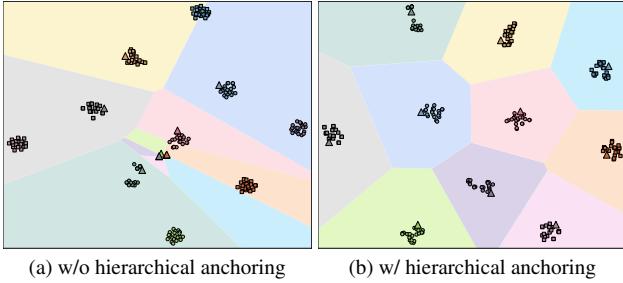


Figure 5. t-SNE [60] visualizations of visual and textual features on CIFAR100 B0 Inc5. We show the feature distributions of old classes and new classes without (left) and with (right) hierarchy.

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 cone-based hierarchical loss $\mathcal{L}_{\text{hier}}$.

w/o Projection removes the TP null-space projection. **ZS-CLIP** is the zero-shot baseline. 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 parent-centered anchors. Removing projection incurs a smaller early drop but hurts later stages as interference accumulates. ZS-CLIP degrades the fastest. These trends confirm that hierarchy supplies stable anchors and TP projection preserves prior mappings; both are needed for strong performance.

Parameter robustness: We conduct a robustness study on two hyperparameters, including the text–text hierarchical weight λ_1 and the hyperbolic mapping weight β in Eq. (13). On CUB B0 Inc20 setting, we sweep λ_1 among $\{0.30, 0.45, 0.50, 0.55, 0.60\}$ and β among $\{0.05, 0.10, 0.15, 0.20\}$. Figure 4b reports the final-stage accuracy, showing stable performance across a broad range. We adopt $\lambda_1 = 0.5$ and $\beta = 0.1$ as defaults. Additional results for other parameters are provided in the appendix.

Visualizations: We use t-SNE [60] to visualize cross-modal features on CIFAR100 B0 Inc5 in Figure 5, comparing models with and without hierarchical anchoring across the tasks. First-task visual features are dots (\circ) and second-task ones are squares (\square). Text embeddings are triangles (\triangle), and shaded regions denote decision areas induced by the text points. As shown in Figure 5a, visual clusters drift away from their text counterparts and even enter other classes’ regions. As shown in Figure 5b, drift is suppressed and image–text features remain co-located for old and new tasks. These results indicate that HASTEN aligns modalities via hierarchical anchors and preserves previously learned structure during incremental training.

Image traversals: Following [13], we visualize hierarchical consistency on the UCF101 dataset via geodesic



Ours	CLIP	Ours	CLIP
Tai Chi	Tai Chi	Parallel Bars	Parallel Bars
<i>martial_art</i>	↓	<i>a gymnast</i>	↓
<i>a practitioner</i>	↓	<i>supporting the</i>	
<i>flowing through</i>		<i>body and</i>	
<i>slow, mindful tai</i>		<i>swinging</i>	
<i>chi forms in a</i>		<i>between the</i>	
<i>park.</i>		<i>parallel bars.</i>	
<i>a martial arts</i>	↓	<i>a gymnast</i>	↓
<i>drill emphasizing</i>		<i>balancing</i>	
<i>timing, guard,</i>		<i>inverted and</i>	
<i>and footwork.</i>		<i>walking on hands</i>	
“entity”	[ROOT]	“entity”	[ROOT]

Figure 6. 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 path yields fewer descriptions.

traversals from an image to the root (“entity”) in our hyperbolic model and [ROOT] for CLIP. Along an image-to-root path, ancestors correspond to the shortest paths to the root. We interpolate 100 equally spaced points along the geodesic from the image embedding to the root and, at each point, retrieve the nearest neighbor from a text set that includes the root. As shown in Figure 6, our method yields descriptions that become increasingly generic as the path approaches the root, indicating that the hyperbolic representation captures the semantic tree, whereas CLIP typically produces a single description. More examples are provided in the appendix.

6. Conclusion

This paper tackles CIL with a hierarchy-aware framework in hyperbolic space. HASTEN builds a dataset-conditioned semantic tree and uses task-specific hierarchy-aware modules with a shared hyperbolic mapper to map features onto the hyperbolic space. Parent–child order is enforced with entailment cones and a margin, and a hyperbolic contrastive loss strengthens text–image alignment. Virtual-class anchoring removes the need for exemplars, and projecting hyperbolic mapper updates into an approximate null space preserves prior predictions. Experiments show consistent gains, less drift, and reduced forgetting.

Limitations: This paper relies on a GPT-generated semantic tree; when GPT-5 lacks sufficient domain coverage or inadvertently introduces bias, the resulting hierarchical structure can be noisy or incomplete. Future work will incorporate curated knowledge graphs and data-driven hierarchy induction with light human expert validation and oversight.

References

- [1] Charu C Aggarwal. A survey of stream clustering algorithms. In *Data Clustering*, pages 231–258. Chapman and Hall/CRC, 2018. 1
- [2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 2
- [3] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *CVPR*, pages 11254–11263, 2019. 2
- [4] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023. 14
- [5] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *NeurIPS*, pages 9448–9458, 2019. 6
- [6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101-mining discriminative components with random forests. In *ECCV*, pages 446–461, 2014. 6
- [7] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Gui, Cordelia Schmid, and Kartek Alahari. End-to-end incremental learning. In *ECCV*, pages 233–248, 2018. 2
- [8] Arslan Chaudhry, Marc’Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a gem. In *ICLR*, 2018. 2
- [9] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, pages 16664–16678, 2022. 2
- [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*, pages 3393–3403, 2023. 1
- [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, 2021. 2
- [12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 1
- [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. 1, 3, 8
- [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. 1
- [15] Songlin Dong, Xiaopeng Hong, Xiaoyu Tao, Xinyuan Chang, Xing Wei, and Yihong Gong. Few-shot class-incremental learning via relation knowledge distillation. In *AAAI*, pages 1255–1263, 2021. 2
- [16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 2
- [17] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pages 86–102, 2020. 2
- [18] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR*, pages 9285–9295, 2022. 2
- [19] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1
- [20] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-DFCIL: relation-guided representation learning for data-free class incremental learning. In *ECCV*, pages 423–439, 2022. 2
- [21] 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. 1
- [22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 1
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2015. 1
- [24] 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 *ICCV*, pages 8340–8349, 2021. 6
- [25] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006. 1
- [26] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 2
- [27] Linlan Huang, Xusheng Cao, Haori Lu, and Xialei Liu. Class-incremental learning with clip: Adaptive representation adjustment and parameter fusion. In *ECCV*, pages 214–231, 2024. 1, 2, 3, 6, 7, 16

- [28] Linlan Huang, Xusheng Cao, Haori Lu, Yifan Meng, Fei Yang, and Xialei Liu. Mind the gap: Preserving and compensating for the modality gap in clip-based continual learning. In *ICCV*, pages 3777–3786, 2025. 6, 7, 16
- [29] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hanneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 6, 14
- [30] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022. 15
- [31] Dahuin Jung, Dongyoon Han, Jihwan Bang, and Hwanjun Song. Generating instance-level prompts for rehearsal-free continual learning. In *ICCV*, pages 11847–11857, 2023. 2
- [32] 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. 2
- [33] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop*, pages 554–561, 2013. 6
- [34] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6
- [35] Guannan Lai, Yujie Li, Xiangkun Wang, Junbo Zhang, Tianrui Li, and Xin Yang. Order-robust class incremental learning: Graph-driven dynamic similarity grouping. In *CVPR*, pages 4894–4904, 2025. 1
- [36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1
- [37] Li Li, Jiawei Peng, Huiyi Chen, Chongyang Gao, and Xu Yang. How to configure good in-context sequence for visual question answering. In *CVPR*, pages 26710–26720, 2024. 1
- [38] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017. 2
- [39] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6
- [40] Yichen Lu, Mei Wang, and Weihong Deng. Augmented geometric distillation for data-free incremental person reid. In *CVPR*, pages 7329–7338, 2022. 2
- [41] Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *CVPR*, pages 11371–11380, 2023. 2
- [42] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 6
- [43] 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, 2022. 1, 2
- [44] Mark D McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *NeurIPS*, 2023. 2
- [45] OpenAI. Gpt-5 system card, 2025. 3, 7
- [46] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *ICCV*, pages 13698–13707, 2021. 2
- [47] 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. In *NeurIPS*, pages 8026–8037, 2019. 6
- [48] 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 *ICML*, pages 8748–8763, 2021. 1, 2, 5, 7, 15
- [49] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *ICLR*, 2021. 1
- [50] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NIPS*, pages 506–516, 2017. 2
- [51] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 1, 2, 6, 7, 14
- [52] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International symposium on graph drawing*, pages 355–366. Springer, 2011. 3
- [53] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, pages 4548–4557. PMLR, 2018. 1
- [54] 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. In *NeurIPS*, pages 6747–6761, 2021. 1
- [55] 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 *CVPR*, pages 16722–16731, 2022. 2
- [56] 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 *CVPR*, pages 11909–11919, 2023. 2, 6, 7, 16
- [57] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4080–4090, 2017. 2
- [58] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016. 5

- [59] 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. 6
- [60] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008. 8
- [61] 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. 6
- [62] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, pages 398–414, 2022. 2
- [63] 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 *ICLR*, 2023. 2
- [64] 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. 1
- [65] Runqi Wang, Xiaoyue Duan, Guoliang Kang, Jianzhuang Liu, Shaohui Lin, Songcen Xu, Jinhua Lü, and Baochang Zhang. Attriclip: A non-incremental learner for incremental knowledge learning. In *CVPR*, pages 3654–3663, 2023. 2
- [66] 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 *ECCV*, pages 631–648, 2022. 2, 6, 7, 16
- [67] 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 *CVPR*, pages 139–149, 2022. 1, 2, 6, 7, 15
- [68] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 2
- [69] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010. 6
- [70] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021. 2
- [71] Jiazu 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 *CVPR*, pages 23219–23230, 2024. 1
- [72] Jiazu 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 *CVPR*, pages 23219–23230, 2024. 2
- [73] Lu Yu, Bartłomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, pages 6982–6991, 2020. 2, 3
- [74] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017. 2
- [75] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pages 13208–13217, 2020. 1, 2
- [76] Bowen Zheng, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Task-agnostic guided feature expansion for class-incremental learning. In *CVPR*, pages 10099–10109, 2025. 2
- [77] 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*, 133:1012–1032, 2025. 1, 2, 6, 7, 15
- [78] 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. 1, 2, 3, 6, 7, 16
- [79] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, pages 16816–16825, 2022. 1
- [80] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 130(9):2337–2348, 2022. 1, 2, 6
- [81] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, pages 5871–5880, 2021. 2, 5

Appendix

In this appendix, we provide additional information about HASTEN, including implementation specifics and extended experimental results.

Section A presents the exact prompt used with GPT-5 to generate the hierarchical semantic trees.

Section B presents the complete set of experimental curves and detailed descriptions of the compared methods. Furthermore, it includes extended robustness evaluations across multiple random seeds, alternative pre-trained backbones, and different LLMs.

Section C presents additional image traversal examples and explains how the corresponding text set is constructed.

Section D supplements the parameter robustness study with sensitivity analyses of additional key hyperparameters: λ_2 (text-to-image hierarchical loss weight) and δ (parent-child separation margin).

Section E analyzes the computational efficiency of our method, reporting the comparisons of trainable parameters and running times against other baselines.

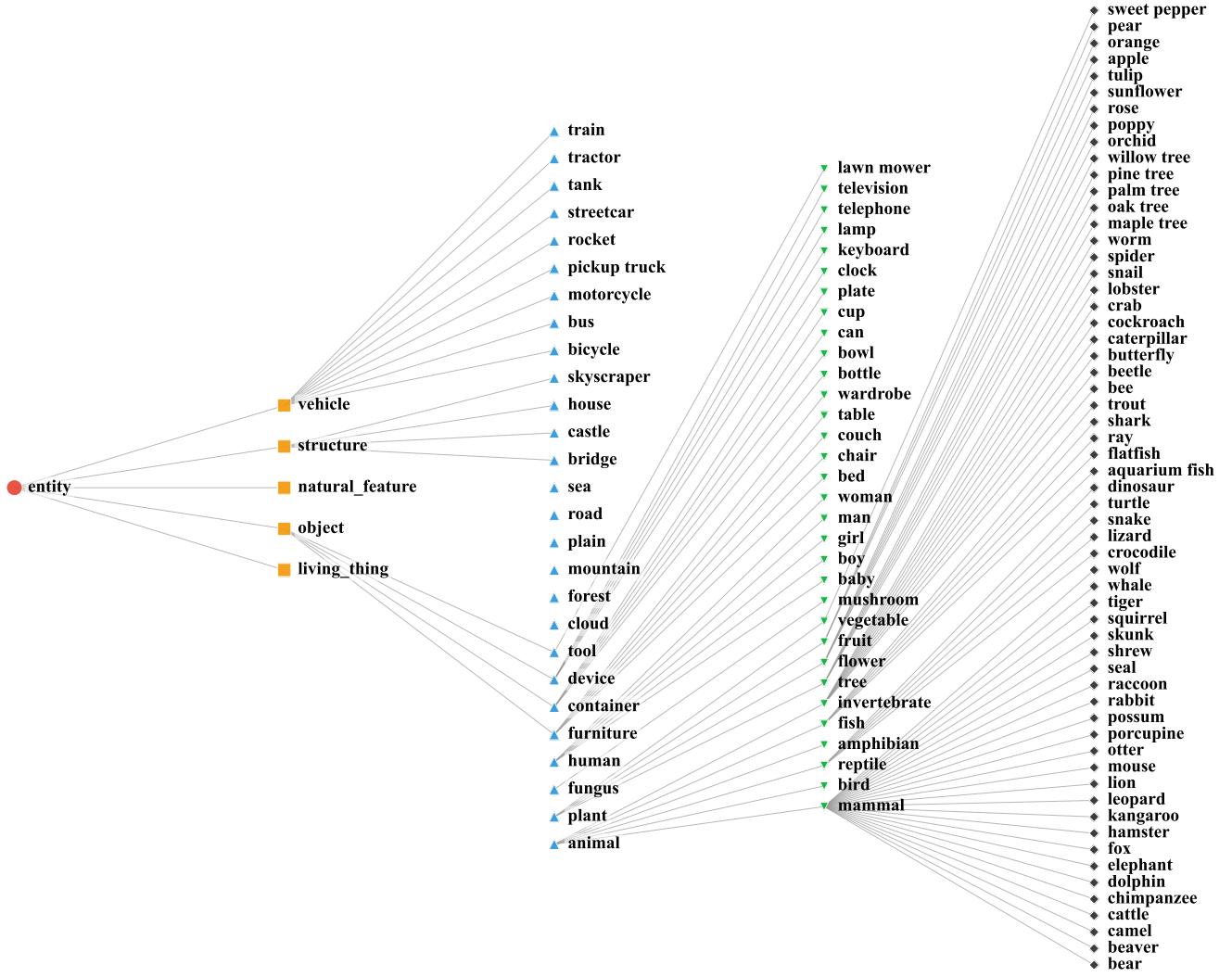


Figure 7. Visualization of the semantic tree structure derived from the output JSON on CIFAR100. The root node “entity” branches into abstract categories (e.g., “living_thing”), which further subdivide until reaching the specific leaf classes from the input list.

A. Generating Hierarchical Semantic Tree

In this section, we present the full prompt used for hierarchical semantic tree generation and illustrate it with a CIFAR100 example.

Q: You are a meticulous and highly accurate taxonomist and ontologist. Your goal is to organize a given list of class names into a complete hierarchical “is-a” structure (parent-child relationships), ensuring that no class from the input list is omitted.

Primary objectives

Total inclusion: Every single class name from the input list must appear in the final JSON output, either as a child (in a list) or as a parent key

(if it has children). No omissions are allowed.

Single root node: All branches of the hierarchy must eventually connect to a single root node named “entity”.

Abstract parents allowed: You may introduce new abstract grouping nodes (not from the input list) to build a logical and coherent hierarchy (for example, “animal”, “vehicle”, “fruit”).

Structure rules

No parent-child duplicates: A class name may not appear both as a key and as one of its own direct children. Parent and child names must always be different.

No empty lists: Only classes or abstract nodes that actually have children should appear as dic-

tionary keys. If a node has no children, it should only appear as a child in another node's list.

Preserve original names: Use the input names exactly as given (keep case, spaces, and hyphens). Only abstract grouping nodes may be newly created, and those should use lowercase words with underscores (for example, "living_thing").

Tree shape only: The structure must form a directed acyclic tree: no cycles and no duplicate parentage.

Direct relationships only: Each key's value list must represent its *immediate* children, not grandchildren.

Output format

The result must be a single valid JSON object (dictionary). Keys are parent class names (each must have at least one child). Values are lists of the direct child class names (strings). Do not include comments, trailing commas, or explanatory text: only the pure JSON.

Self-check before finalizing

Before you output the JSON, verify the following conditions: Every class name from the input list appears at least once in the JSON. No key has a child identical to itself. No key has an empty list as its value. Only classes with children appear as keys. Every branch leads back up to "entity". The JSON syntax is valid and contains only one object.

Example:

Input: ["alaskan malamute", "ragdoll", "golden retriever", "british shorthair"]

Correct output:

```
{“entity”:[“animal”],“animal”:[“cat”, “dog”],  
“cat”:[“ragdoll”, “british shorthair”],  
“dog”:[“golden retriever”, “alaskan malamute”]}
```

Notes: All input items appear exactly once. No key equals any of its children. There are no empty lists. Every path connects to "entity".

Now, process the following list. Remember: complete inclusion is mandatory.

Input list: ["apple", "aquarium fish", "baby", "bear", "beaver", "bed", "bee", "beetle", "bicycle", "bottle", "bowl", "boy", "bridge", "bus", "butterfly", "camel", "can", "castle", "caterpillar", "cattle", "chair", "chimpanzee", "clock", "cloud", "cockroach", "couch", "crab", "crocodile", "cup", "dinosaur", "dolphin", "elephant", "flatfish", "forest", "fox", "girl", "hamster", "house", "kangaroo", "keyboard", "lamp", "lawn mower", "leopard", "lion", "lizard", "lobster", "man",

"maple tree", "motorcycle", "mountain", "mouse", "mushroom", "oak tree", "orange", "orchid", "otter", "palm tree", "pear", "pickup truck", "pine tree", "plain", "plate", "poppy", "porcupine", "possum", "rabbit", "raccoon", "ray", "road", "rocket", "rose", "sea", "seal", "shark", "shrew", "skunk", "skyscraper", "snail", "snake", "spider", "squirrel", "streetcar", "sunflower", "sweet pepper", "table", "tank", "telephone", "television", "tiger", "tractor", "train", "trout", "tulip", "turtle", "wardrobe", "whale", "willow tree", "wolf", "woman", "worm"]

A: Output JSON: {“entity”:[“living_thing”, “object”, “natural_feature”, “structure”, “vehicle”], “living_thing”:[“animal”, “plant”, “fungus”, “human”], “animal”:[“mammal”, “bird”, “reptile”, “amphibian”, “fish”, “invertebrate”], “mammal”:[“bear”, “beaver”, “camel”, “cattle”, “chimpanzee”, “dolphin”, “elephant”, “fox”, “hamster”, “kangaroo”, “leopard”, “lion”, “mouse”, “otter”, “porcupine”, “possum”, “rabbit”, “raccoon”, “seal”, “shrew”, “skunk”, “squirrel”, “tiger”, “whale”, “wolf”], “reptile”:[“crocodile”, “lizard”, “snake”, “turtle”, “dinosaur”], “fish”:[“aquarium fish”, “flatfish”, “ray”, “shark”, “trout”], “invertebrate”:[“bee”, “beetle”, “butterfly”, “caterpillar”, “cockroach”, “crab”, “lobster”, “snail”, “spider”, “worm”], “plant”:[“tree”, “flower”, “fruit”, “vegetable”], “tree”:[“maple tree”, “oak tree”, “palm tree”, “pine tree”, “willow tree”], “flower”:[“orchid”, “poppy”, “rose”, “sunflower”, “tulip”], “fruit”:[“apple”, “orange”, “pear”], “vegetable”:[“sweet pepper”], “fungus”:[“mushroom”], “human”:[“baby”, “boy”, “girl”, “man”, “woman”], “object”:[“furniture”, “container”, “device”, “tool”], “furniture”:[“bed”, “chair”, “couch”, “table”, “wardrobe”], “container”:[“bottle”, “bowl”, “can”, “cup”, “plate”], “tool”:[“lawn mower”], “device”:[“clock”, “keyboard”, “lamp”, “telephone”, “television”], “vehicle”:[“bicycle”, “bus”, “motorcycle”, “pickup truck”, “rocket”, “streetcar”, “tank”, “tractor”, “train”], “structure”:[“bridge”, “castle”, “house”],

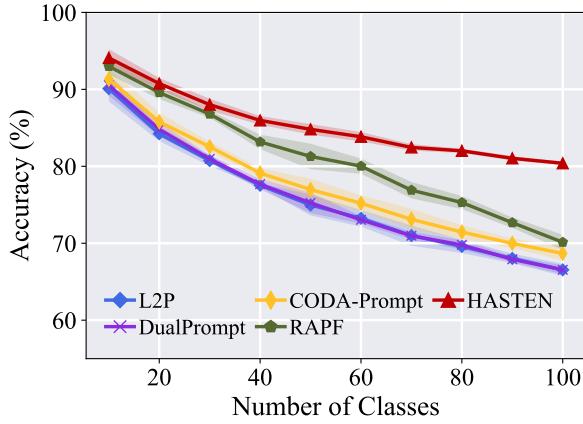


Figure 8. Results on ImageNet-R B0 Inc20 with multiple runs. HASTEN consistently outperforms other methods by a substantial margin.

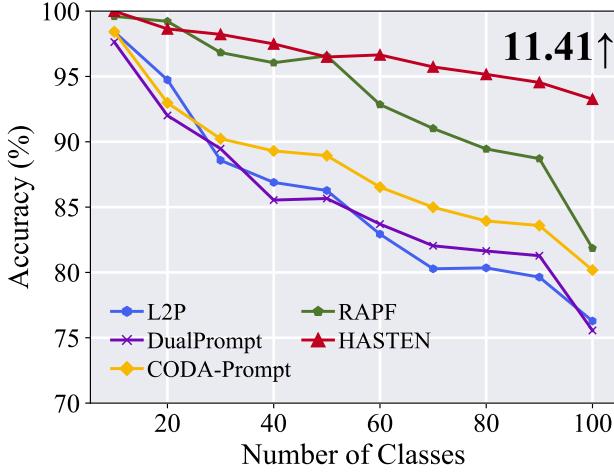


Figure 9. Experiments when using OpenAI weights on UCF B0 Inc10. HASTEN consistently outperforms other methods across different backbone weights.

```

“skyscraper”],
“natural_feature”:[“cloud”,“forest”,“mountain”,
“plain”,“road”,“sea”]}

```

Figure 7 visualizes the hierarchical structure derived from the above JSON output for CIFAR100, providing an intuitive view of the parent-child relationships between classes.

B. Supplementary Results and Methods

B.1. Evaluation Across Multiple Random Seeds

In the main paper, experimental results are obtained based on class splitting with the random seed 1993, adhering to the standard protocol in CIL [51]. To investigate the ro-

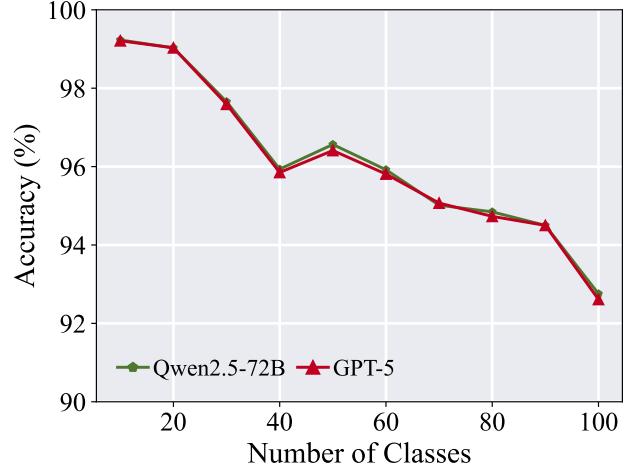


Figure 10. Results on UCF B0 Inc10 with different LLMs for hierarchical semantic tree. HASTEN is robust and compatible with various LLMs.

bustness of different methods, we extend our evaluation by conducting multiple independent trials. Specifically, we execute the class splitting process using a set of five distinct random seeds 1993, 1994, 1995, 1996, 1997 and calculate the average performance along with the standard deviation. The results on ImageNet-R B0 Inc20, presented in Figure 8, demonstrate that HASTEN exhibits superior robustness compared to the baselines, as it consistently outperforms competing methods across these repeated runs.

B.2. Results with Different backbones

In the main paper, our experiments mainly use CLIP ViT-B/16 with LAION-400M pre-trained weights [29]. 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 9. Across different backbone initializations, HASTEN consistently outperforms competing methods, demonstrating strong robustness to variations in pre-trained models.

B.3. Different LLMs

In the main paper, we use GPT-5 to generate the hierarchical semantic tree. As HASTEN is a general framework compatible with diverse LLMs, we also adopt Qwen2.5-72B [4] for hierarchical semantic tree generation and carry out experiments on the UCF B0 Inc10. The performance comparison between GPT-5 and Qwen2.5-72B is presented in Figure 10. As illustrated, the results from the two LLMs are highly consistent, demonstrating the robustness of HASTEN across different LLMs.

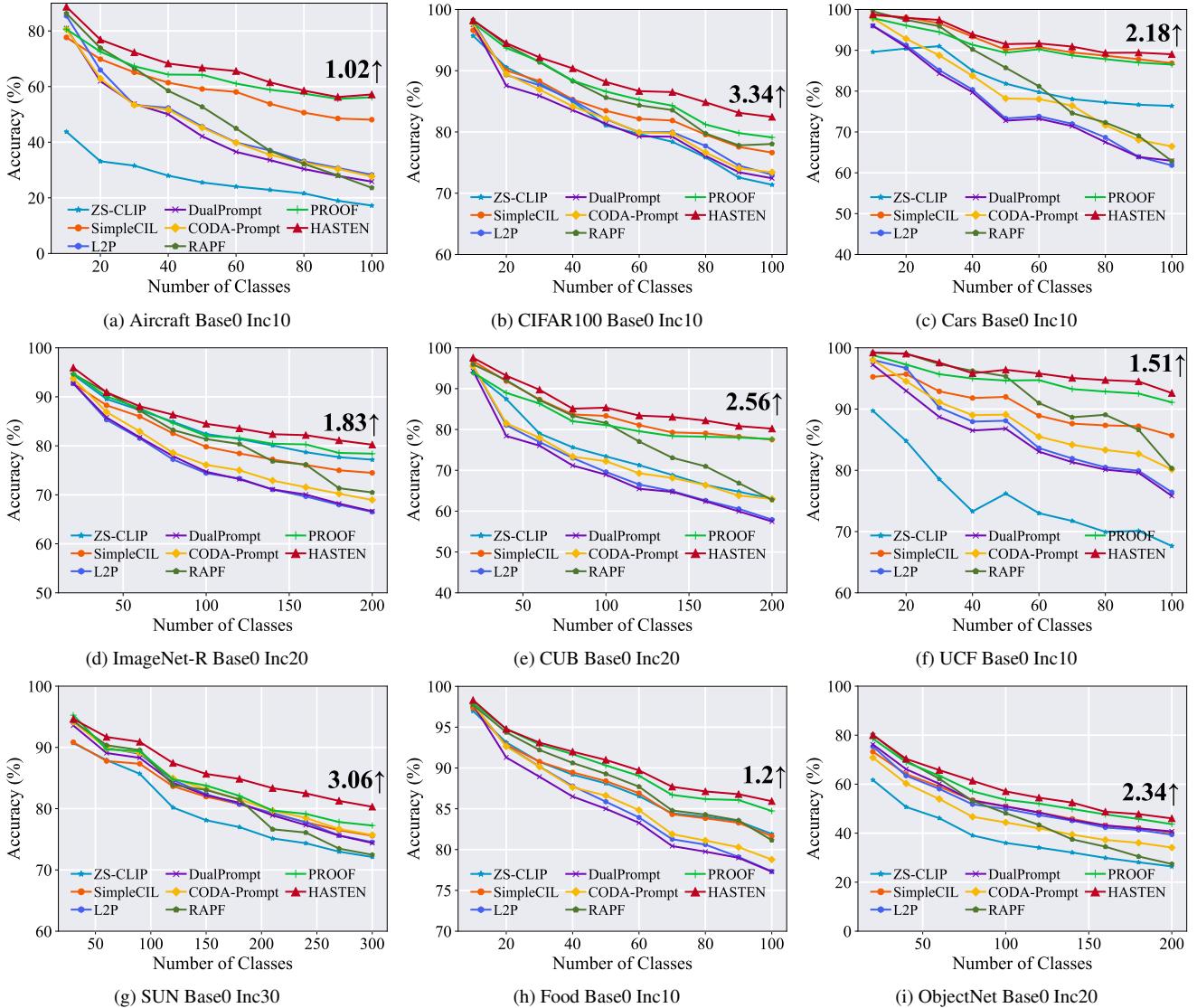


Figure 11. Incremental performance of different methods on B0 setting. We report the performance gap after the last incremental stage of HASTEN and the runner-up method at the end of the line. All methods utilize the same CLIP pre-trained weight.

B.4. 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** [77]: 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 en-

coder 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** [48]: 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** [67]: utilizes only the visual branch of CLIP. During the update process, it freezes the pre-trained weights and applies visual prompt tuning [30] to adapt to new task features. It generates instance-specific prompts through a prompt pool, constructed via key-value mapping.

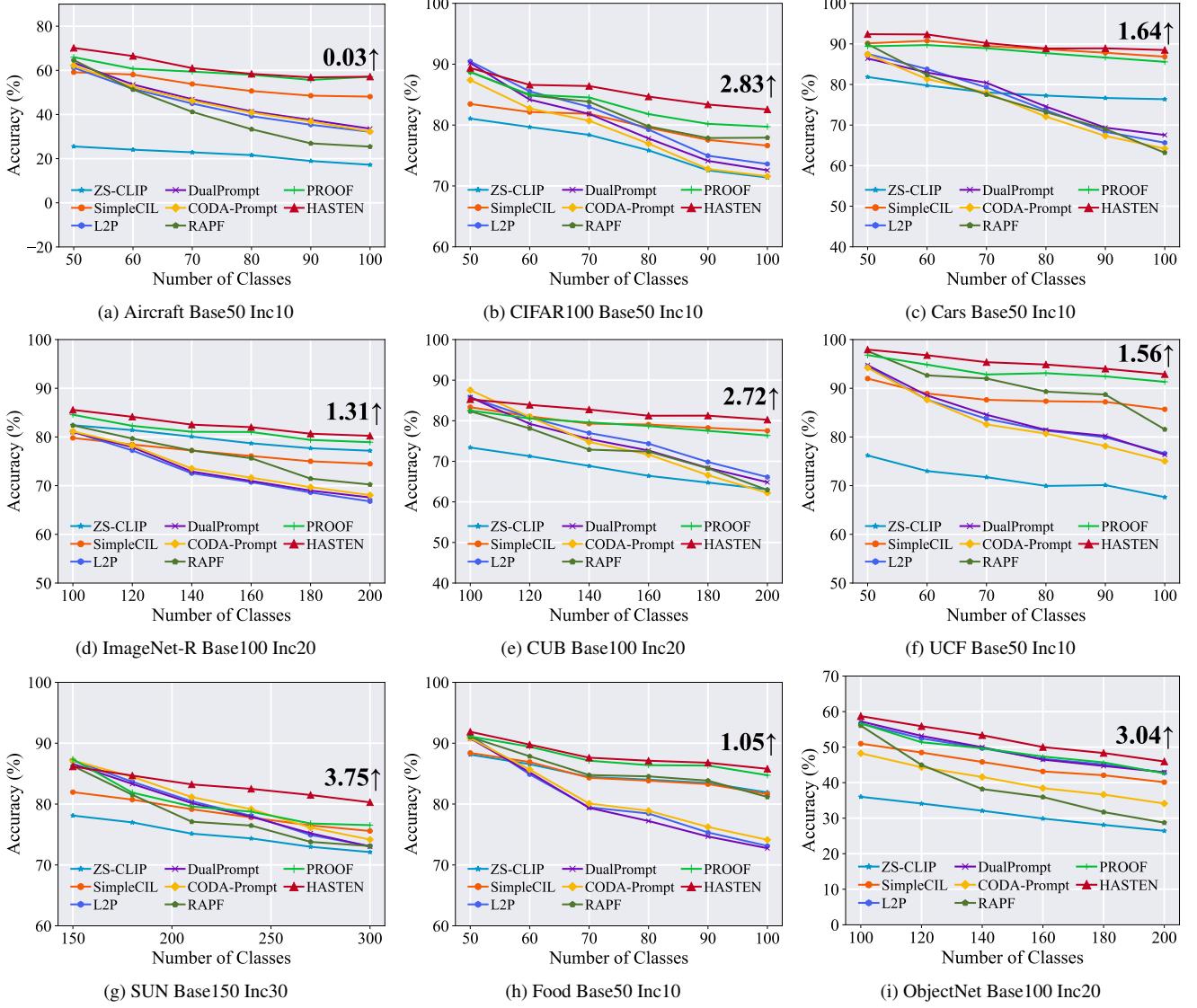


Figure 12. Incremental performance of different methods on half-base setting. We report the performance gap after the last incremental stage of HASTEN and the runner-up method at the end of the line. All methods utilize the same CLIP pre-trained weight.

- **DualPrompt [66]**: 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 [56]**: 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 [27]**: enhances the continual learning capacity of CLIP by combining a hard class separation loss with de-

composed parameter fusion to integrate new knowledge into the CLIP model.

- **PROOF [78]**: 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.
- **MG-CLIP [28]**: addresses continual learning by preserving CLIP’s inherent modality gap between image and text features to maintain pre-trained knowledge and mitigate forgetting. It introduces an auxiliary visual-space classifier to compensate for the gap’s limitations and enhance the model’s adaptability to new tasks.

B.5. Additional Experimental Results

In this section, we present the complete set of experimental curves for all compared methods. The main paper only includes three representative learning trends; here, we provide the full results corresponding to Table 1. Specifically, we plot the incremental performance of different methods with 0 base classes in Figure 11 and with half base classes in Figure 12. Across all datasets and data splits, HASTEN consistently achieves superior performance over competing approaches.

C. More Details of Image Traversals

In this section, we present additional image traversal examples on the UCF101 dataset and describe the text sets used in these visualizations. All traversals are computed on images from the UCF101 test split, with qualitative results shown in Figures 16. For each UCF101 class, we construct a text set that includes the original class name together with short descriptions and related adjectives generated by GPT-5, arranged from very specific to increasingly generic. The complete text set used for the image traversal is formed by combining the text sets from all classes. Below, we provide two examples that illustrate how such text sets are generated:

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.

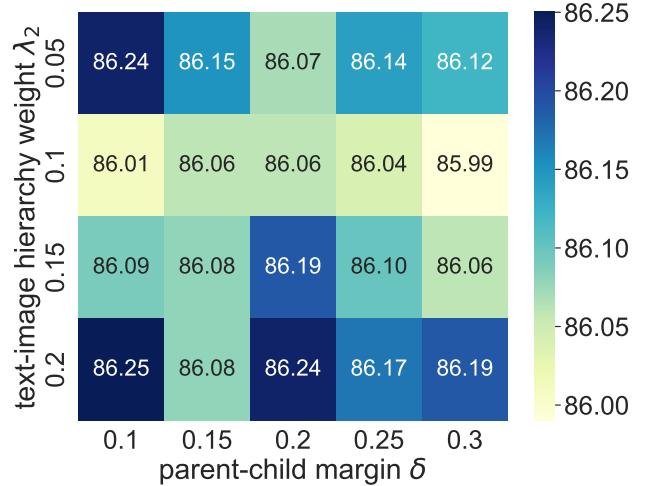


Figure 13. Parameter sensitivity

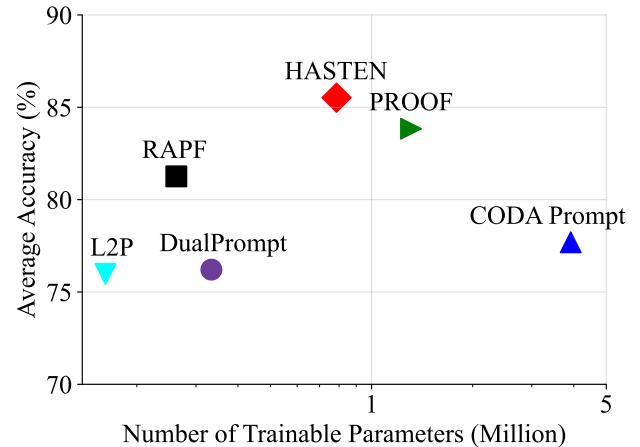


Figure 14. Comparison of accuracy and trainable parameters of different methods on ImageNet-R B-10 Inc-20.

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 a complementary sensitivity analysis for two additional key hyperparameters to further demonstrate the stability of HASTEN. Specifically, 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. We also study 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 in the hyperbolic space. We follow the same protocol as in the main paper, evaluating on the CUB B0 Inc20 set-

Table 2. Number of trainable parameters on ImageNet-R B0 Inc20 setting.

Method	Trainable Parameters
L2P	161330
DualPrompt	333412
CODA-Prompt	3916900
PROOF	1310720
RAPF	262144
HASTEN	786432

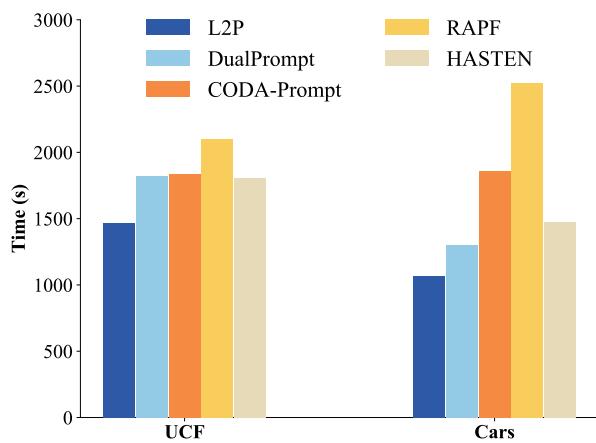


Figure 15. Running time comparison. HASTEN achieves the best performance while maintaining a training time comparable to other compared methods.

ting. We sweep λ_2 over $\{0.05, 0.10, 0.15, 0.20\}$ and δ over $\{0.10, 0.15, 0.20, 0.25, 0.30\}$, and adopt $\lambda_2 = 0.10$ and $\delta = 0.20$ as defaults while keeping all other hyperparameters fixed. The results, shown in Figure 13, indicate that the performance of HASTEN is robust to changes in both λ_2 and δ . This confirms that our framework is not overly sensitive to these hyperparameter choices, making it practical and easy to deploy.

E. Trainable Parameters and Running Time

E.1. Trainable Parameter Analysis

In this paper, we design HASTEN by extending hierarchy-aware module per task and incorporating a task-shared hyperbolic mapping layer. During inference, the features are aggregated into the final Euclidean feature. As illustrated in Section 4.2 of the main paper, we can reparameterize these hierarchy-aware modules by adding the weights since they are linear layers, *i.e.*, $\sum_p H_t^p$ and $\sum_p H_t^p$. Hence, the parameter size of hierarchy-aware modules can be squeezed from $2 \times b \times d \times d$ to $2 \times d \times d$. In addition to the hierarchy-aware modules, we introduce the global hyperbolic map-

ping layer TP, which adds another set of parameters, with a total of $d \times d$ trainable parameters. As a result, the total parameter size is $3 \times d \times d$. We further report the number of trainable parameters for each compared method in Table 2. In addition, we visualize the relationship between trainable parameter size and average accuracy in Figure 14, 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.

E.2. Running Time Comparison

In this section, we present a running time comparison of different methods. To ensure a fair evaluation, we conduct all experiments under the same experimental setting and report the results in Figure 15. As inferred from the figure, HASTEN achieves the best performance while maintaining a competitive running time relative to baselines like CODA-Prompt and RAPF, effectively verifying the efficiency and effectiveness of our approach.



Ours	CLIP
Basketball Dunk	↓
an athlete sprinting to the rim and slamming the ball with a two-handed dunk.	↓
a skateboarder rolling down a street and popping an ollie over a curb.	↓

“entity” [ROOT]



Ours	CLIP
Cricket Bowling	individual_sport
individual_sport	↓
a person enjoying a light outdoor pastime.	↓

“entity” [ROOT]



Ours	CLIP
Playing Tabla	Playing Tabla
a tabla player producing distinct bols with fingertips and palms.	↓
Playing Sitar	↓

“entity” [ROOT]



Ours	CLIP
Rope Climbing	Rock Climbing Indoor
Rock Climbing	↓
an athlete climbing a thick rope using legs and arms in a gym.	↓

“entity” [ROOT]



Ours	CLIP
Soccer Penalty	Soccer Penalty
Field Hockey Penalty	↓
a player taking a penalty stroke and aiming low into the corner of the goal.	↓
a person practicing a sport indoors.	↓

“entity” [ROOT]



Ours	CLIP
Pommel Horse	Pommel Horse
gymnastics	Balance Beam
Balance Beam	↓
apparatus_-gymnastics	↓
a gymnast holding a balanced pose on apparatus.	↓

“entity” [ROOT]



Ours	CLIP
Salsa Spin	Salsa Spin
a dancer spinning quickly while maintaining salsa timing with a partner.	a person jumping with arms and legs spreading in a rhythmic exercise.
a person jumping with arms and legs spreading in a rhythmic exercise.	↓

“entity” [ROOT]



Ours	CLIP
Shotput	Shotput
Javelin Throw	↓
a person warming up before exercise.	↓
an athlete practicing a sport on a field or court.	↓

“entity” [ROOT]

Figure 16. 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.