Hier-COS: Making Deep Features Hierarchy-aware via Composition of Orthogonal Subspaces

Supplementary Material

We structure this supplementary material as follows, providing relevant references (in blue) to the main paper. In Sections 7 and 8, we prove that our proposed formulation of Hier-COS results in an HAVS and is thereby hierarchically consistent (as mentioned in Sec. 3.2 of the main paper). We then discuss a few important geometric properties of Hier-COS in Section 9. In Section 10, we provide details about our experimental setup, dataset details, comparative baselines and implementation details (as mentioned in the Sec. 5 of the main paper). In Section 11, we discuss the results on tieredImageNet-H (as mentioned in Sec. 5 of the main paper) and provide an empirical validation of our model's hierarchical consistency. We also present the results of our ablation study, sensitivity analysis and qualitative results demonstrating the quality of feature representations learned by Hier-COS in comparison to previous methods. In Section 12, we review the existing metrics used to evaluate hierarchical classification, information retrieval and ranking systems (as mentioned in Sec. 4.1 of the main paper). We formally define HOPS in Section 13, and show its computation with an example (as mentioned in Sec. 4.2 of the main paper). We define and discuss the limitations of each metric. In Section 14, we provide additional discussion on our framework, its complexity and some of the future research directions. We also provide additional supplementary files with our submission, which we summarize in Section 15.

7. Proof of Theorem 1

Theorem 2. *Consider a vector space $V_{\mathcal{T}}$ and its subspaces for fine-grained classes $\{V_{y_1}, \cdots, V_{y_K}\}$ spanned by $\{\mathcal{E}_{y_1}, \cdots, \mathcal{E}_{y_K}\}$ defined using the hierarchy tree $\mathcal{T}$, as discussed above. For all $\mathbf{x} \in V_{\mathcal{H}}$, if $\mathbf{x} \in V_{y_i}$ and $\langle \mathbf{x}, e_i \rangle^2 > 0$, $\forall\, e_i \in \mathcal{E}_{y_i}$, then $V_{\mathcal{H}}$ is an n-dimensional HAVS induced by an LCA-based tree distance function $D_{\mathcal{T}}$.*

Proof. To prove $V_{\mathcal{T}}$ is an HAVS, we must satisfy Def. 1. Let $\mathbf{x}$ be a feature vector for class y_i such that the theorem's conditions hold. We must show that for any two other classes $y_j, y_k \in \mathcal{Y}$:

$$\begin{aligned} \text{if} \quad & D_{\mathcal{T}}(y_i, y_j) < D_{\mathcal{T}}(y_i, y_k) \\ \text{then} \quad & |D_i - D_j| < |D_i - D_k| \end{aligned}$$

where, $D_{\mathcal{T}}$ is an LCA-based distance function, which implies that $D_{\mathcal{T}}(y_i, y_j) < D_{\mathcal{T}}(y_i, y_k)$ if and only if $|f_a(v_i) \cap f_a(v_j)| > |f_a(v_i) \cap f_a(v_k)|$.

First, by the theorem's assumption $\mathbf{x} \in V_{y_i}$, the projection of $\mathbf{x}$ onto its own subspace is $\mathbf{x}$ itself. Thus, its distance to its own subspace is zero:

$$D_i = \|\mathbf{x} - \mathbb{P}_{y_i}(\mathbf{x})\| = 0 \tag{7}$$

The HAVS condition simplifies to proving:

$$\begin{aligned} \text{if} \quad & D_{\mathcal{T}}(y_i, y_j) < D_{\mathcal{T}}(y_i, y_k) \\ \text{then} \quad & D_j < D_k \end{aligned}$$

From the main paper's definition (Eq. 2), the squared distance from $\mathbf{x}$ to any subspace V_{y_j} is the sum of squared components of $\mathbf{x}$ on the orthogonal complement basis $\neg\mathcal{E}_{y_j}$:

$$D_j^2(\mathbf{x}) = \sum_{e_m \in \neg\mathcal{E}_{y_j}} \langle \mathbf{x}, e_m \rangle^2 \tag{8}$$

Now, we use the first assumption, $\mathbf{x} \in V_{y_i}$. This implies that $\mathbf{x}$ has no components outside of V_{y_i}. Formally, $\langle \mathbf{x}, e_m \rangle = 0, \forall e_m \in \neg\mathcal{E}_{y_i}$, i.e., $\mathbf{x}$ is orthogonal to all the basis vectors not in $\mathcal{E}_{y_i}$. We can substitute this into Eq. 8. The only basis vectors $e_m \in \neg\mathcal{E}_{y_j}$ that can contribute to the sum are those that are also in $\mathcal{E}_{y_i}$.

$$D_j^2(\mathbf{x}) = \sum_{e_m \in \neg\mathcal{E}_{y_j} \cap \mathcal{E}_{y_i}} \langle \mathbf{x}, e_m \rangle^2$$

This set is equivalent to the set difference $\mathcal{E}_{y_i} \setminus \mathcal{E}_{y_j}$. Thus, the distance is the energy of $\mathbf{x}$ projected onto the basis vectors that are in y_i's path but not in y_j's path:

$$D_j^2(\mathbf{x}) = \sum_{e_m \in \mathcal{E}_{y_i} \setminus \mathcal{E}_{y_j}} \langle \mathbf{x}, e_m \rangle^2 \tag{9}$$

Similarly, for class y_k:

$$D_k^2(\mathbf{x}) = \sum_{e_m \in \mathcal{E}_{y_i} \setminus \mathcal{E}_{y_k}} \langle \mathbf{x}, e_m \rangle^2 \tag{10}$$

Since we know that $|f_a(v_i) \cap f_a(v_j)| > |f_a(v_i) \cap f_a(v_k)|$, i.e., the set of shared ancestors for y_i and y_j is larger than for y_i and y_k, therefore:

$$|\mathcal{E}_{y_i} \cap \mathcal{E}_{y_j}| > |\mathcal{E}_{y_i} \cap \mathcal{E}_{y_k}|$$

Since $\mathcal{E}_{y_i} \setminus \mathcal{E}_{y_j}$ is equivalent to $\mathcal{E}_{y_i} \setminus (\mathcal{E}_{y_i} \cap \mathcal{E}_{y_j})$, a larger intersection implies a smaller set difference. Therefore:

$$|\mathcal{E}_{y_i} \setminus \mathcal{E}_{y_j}| < |\mathcal{E}_{y_i} \setminus \mathcal{E}_{y_k}|$$

Furthermore, because the LCA path is shared, the set difference $\mathcal{E}_{y_i} \setminus \mathcal{E}_{y_j}$ is a strict subset of $\mathcal{E}_{y_i} \setminus \mathcal{E}_{y_k}$:

$$(\mathcal{E}_{y_i} \setminus \mathcal{E}_{y_j}) \subset (\mathcal{E}_{y_i} \setminus \mathcal{E}_{y_k}) \tag{11}$$

Finally, we use the second assumption of the theorem: $\langle \mathbf{x}, e \rangle^2 > 0$ for all $e \in \mathcal{E}_{y_i}$. Since the sum for $D_j^2(\mathbf{x})$ (Eq. 9) is over a strict subset of the basis vectors used for $D_k^2(\mathbf{x})$ (Eq. 10), and all terms are strictly positive, the sum over the smaller set must be smaller.

$$D_j^2(\mathbf{x}) < D_k^2(\mathbf{x}) \implies D_j(\mathbf{x}) < D_k(\mathbf{x})$$

Combining this with $D_i(\mathbf{x}) = 0$ (Eq. 7), we have:

$$|D_i(\mathbf{x}) - D_j(\mathbf{x})| < |D_i(\mathbf{x}) - D_k(\mathbf{x})|$$

This satisfies Def. 1, proving that $V_{\mathcal{T}}$ is an HAVS. □

8. Proof of Proposition 1

Lemma 1. *For any given node $v_j \in \mathcal{V}_{\mathcal{T}}$ and its parent $v_p = parent(v_j)$, $V_j \subset V_p$.*

Proof. By definition,

$$\mathcal{E}_j = \mathcal{E}_j^a \cup \{e_j\} \cup \mathcal{E}_j^d$$
$$\mathcal{E}_p = \mathcal{E}_p^a \cup \{e_p\} \cup \mathcal{E}_p^d$$

The ancestors of v_j are v_p and all of v_p's ancestors, i.e.

$$\mathcal{E}_j^a = \mathcal{E}_p^a \cup \{e_p\}$$

The descendants of v_p include v_j and all of v_j's descendants, i.e.

$$\mathcal{E}_p^d \supset \{e_j\} \cup \mathcal{E}_j^d$$

Substituting these into the definition of $\mathcal{E}_p$

$$\begin{aligned}\mathcal{E}_p &= \mathcal{E}_p^a \cup \{e_p\} \cup \mathcal{E}_p^d \\ &\supset \mathcal{E}_j^a \cup (\{e_j\} \cup \mathcal{E}_j^d) \\ &\supset \mathcal{E}_j\end{aligned}$$

Thus, $\mathcal{E}_j \subset \mathcal{E}_p$, which implies $V_j \subset V_p$. □

Proposition 2. *The label hierarchy predicted using Hier-COS is consistent across all the levels, i.e., $\{\hat{y}^{(1)}, \ldots, \hat{y}^{(l)}, \ldots, \hat{y}^{(H)}\}$ is a valid path in the tree.*

Proof. Hierarchical consistency requires that for any predicted node $\hat{y}^{(l)}$ at level $l > 1$, its predicted parent $\hat{y}^{(l-1)}$ must be its true parent, i.e., $\hat{y}^{(l-1)} = \text{parent}(\hat{y}^{(l)})$.

The prediction at any level l is the node $v_j \in \mathcal{V}_{\mathcal{T}}^{(l)}$ that minimizes the distance $D_j^2(\mathbf{x})$:

$$\hat{y}^{(l)} = \arg \min_{v_j \in \mathcal{V}_{\mathcal{T}}^{(l)}} D_j^2(\mathbf{x}) \tag{12}$$

We prove consistency by showing that for an ideal feature vector $\mathbf{x}$ corresponding to a single leaf class $\hat{y}_{leaf}$, the distance to any true ancestor is 0, while the distance to any non-ancestor is strictly positive.

Let $\hat{y}_{leaf}$ be the predicted leaf node. In an ideal setting, $\mathbf{x} \in V_{\hat{y}_{leaf}}$. This implies $\langle \mathbf{x}, e_k \rangle = 0, \forall e_k \in \neg\mathcal{E}_{\hat{y}_{leaf}}$. Let $\hat{y}^{(l)}$ be any true ancestor of $\hat{y}_{leaf}$ at level l. By recursive application of Lemma 1, we have $V_{\hat{y}_{leaf}} \subset V_{\hat{y}^{(l)}}$. Since $\mathbf{x} \in V_{\hat{y}_{leaf}}$, it follows that $\mathbf{x} \in V_{\hat{y}^{(l)}}$. Therefore, the distance from $\mathbf{x}$ to any of its true ancestors is zero:

$$D_{\hat{y}^{(l)}}^2(\mathbf{x}) = \left\| \mathbf{x} - \mathbb{P}_{\hat{y}^{(l)}}(\mathbf{x}) \right\| = 0 \tag{13}$$

Now, let v_q be any *other* node at level l (i.e., $v_q \neq \hat{y}^{(l)}$). Since v_q is at the same level as $\hat{y}^{(l)}$ but is not the same node, v_q cannot be an ancestor of $\hat{y}_{leaf}$. By the definition of the subspace V_q, its basis $\mathcal{E}_q$ consists of $\mathcal{E}_q^a \cup \{e_q\} \cup \mathcal{E}_q^d$. Since v_q is not an ancestor of $\hat{y}_{leaf}$, $e_{\hat{y}_{leaf}} \notin \mathcal{E}_q^a \cup \{e_q\}$. Since $\hat{y}_{leaf}$ is not a descendant of v_q, $e_{\hat{y}_{leaf}} \notin \mathcal{E}_q^d$. Therefore, the basis vector $e_{\hat{y}_{leaf}}$ is not in $\mathcal{E}_q$:

$$e_{\hat{y}_{leaf}} \in \neg\mathcal{E}_q$$

Now consider the distance to this "incorrect" node v_q:

$$D_q^2(\mathbf{x}) = \sum_{e_k \in \neg\mathcal{E}_q} \langle \mathbf{x}, e_k \rangle^2$$

This sum includes the term $\left\langle \mathbf{x}, e_{\hat{y}_{leaf}} \right\rangle^2$. The condition in Theorem 1 is designed to ensure the feature vector $\mathbf{x}$ has non-zero energy on its leaf node, i.e., $\left\langle \mathbf{x}, e_{\hat{y}_{leaf}} \right\rangle^2 > 0$.

Since $D_q^2(\mathbf{x})$ is a sum of non-negative terms (squared inner products) and includes at least one strictly positive term ($\left\langle \mathbf{x}, e_{\hat{y}_{leaf}} \right\rangle^2$), we have:

$$D_q^2(\mathbf{x}) \geq \left\langle \mathbf{x}, e_{\hat{y}_{leaf}} \right\rangle^2 > 0 \tag{14}$$

Comparing Eq. 13 and 14, at any level l, the distance to the true ancestor $\hat{y}^{(l)}$ is 0, while the distance to any other node v_q is strictly positive.

Therefore, the prediction at level l must be the true ancestor:

$$\hat{y}^{(l)} = \arg \min_{v_j \in \mathcal{V}_{\mathcal{T}}^{(l)}} D_j^2(\mathbf{x}) = \hat{y}^{(l)}$$

This holds for all levels, and thus $\hat{y}^{(l-1)} = \text{parent}(\hat{y}^{(l)})$ for all l. The predicted path is guaranteed to be consistent.

□

9. Properties of Hier-COS

1. **Isometry between the subspace distance and LCA-based tree distance:** The sorted order of pairwise distances between the subspaces V_{y_i} and V_{y_j} $\forall j \in [K]$, as defined in Eq. 8, results in an order which is exactly

the same as the partial preference order obtained using the LCA-based tree distance metric. Intuitively, with the increase in the number of common ancestors between any two leaf nodes, the overlap of the subspaces also increases, thereby reducing their distance. Therefore, feature vectors in Hier-COS are inherently consistent with the semantic similarity obtained from the tree.

2. **Isometry between Grassmannian distance and LCA-based tree distance:** For any two distinct leaf nodes y_i and y_j, the squared Grassmannian distance between their subspaces is directly proportional to the LCA-based tree distance between them in the hierarchy tree $\mathcal{T}$. Intuitively, with orthogonal basis vectors as in Hier-COS, the principal angles between any two subspaces can either be exactly zero (common ancestors) or $\pi/2$ (distinct nodes). Since the Grassmannian distance between two subspaces is directly proportional to the sum of their principal angles, therefore, it is directly proportional to the LCA-based tree distance.
3. **Hierarchy-Adaptive Capacity:** The dimensionality of the subspace V_y adapts to the semantic complexity of the class y. Superclasses have higher dimensionality to capture diverse features, while fine-grained classes have constrained dimensionality for specificity. $\mathcal{E}_d$ for a superclass provides the necessary capacity to represent the union of features from all diverse sub-categories (e.g., V_{Animal} must accommodate features of both birds and fish). Whereas $\mathcal{E}_a$ acts as a constraint, forcing the model to learn a specific representation for leaf class that is a refinement of its ancestors, without the excess degrees of freedom found in superclasses.
4. **Unified Classifier for Hierarchy-Aware and Hierarchical Multi-Label Classification:** The distance of the feature vector $\mathbf{x} \in V_{\mathcal{T}}$ can be computed from any subspace $V_i \subset V_{\mathcal{T}}$. This property allows computing the pairwise distance of $\mathbf{x}$ from all the subspaces corresponding to each node at a particular height h, therefore, encouraging predicting a class label for each hierarchical level $h \in [H]$. This enables hierarchical multi-label classification.

10. Experimental Setup

Dataset: We evaluate our approach on the test set of four datasets: FGVC-Aircraft [25], CIFAR-100 [20], iNaturalist-19 [36] and tieredImageNet-H [32]. For FGVC-Aircraft, we adopt the original hierarchy provided by the dataset. For CIFAR-100, we use the hierarchy tree provided by [21], while for iNaturalist-19 and tieredImageNet-H, we use the hierarchies provided by [6]. The statistics of these datasets are summarized in Table 5.

Baseline Methods: We compare our method with [3, 6, 7, 13, 23, 31] and a *flat* cross-entropy-based classification approach. Following [23], we also include baselines without the transformation layer (I) and with the transformation layer (II). For a fair comparison with the best competing methods, we re-did the experiments using the official codebase provided by the authors. For methods denoted with an asterisk (*), we simply use the metrics reported by [13]. Among all the baseline methods, [7, 13] train an additional classifier for each level of granularity. We compare our results with them to evaluate hierarchical consistency. Although cross entropy and [23] do not explicitly allow classification across all the hierarchical levels, we compute the probabilities for a super-class as the sum of probabilities of all its children to compare the FPA on CIFAR-100 [13]. None of the previous methods evaluate the hierarchical performance of a transformer-based method. We create a simple baseline where we only fine-tune the output layer of a pre-trained ViT using the cross-entropy loss.

Dataset	H	K	n	Train	Val	Test
FGVC-Aircraft	3	100	200	3.3K	3.3K	3.3K
CIFAR-100	5	100	134	45K	5K	10K
iNaturalist-19	7	1010	1189	187K	40K	40K
tieredImageNet-H	12	608	842	425K	15K	15K

Table 5. Statistics of the datasets while excluding the root node. H is the height of the hierarchy tree, K is the number of fine-grained classes and n is the total number of nodes in the tree. The rightmost three columns represent the approximate number of samples in train, val and test sets, respectively.

Backbone Architectures: Following the same strategy as [13, 23], we use WideResNet architecture for CIFAR-100 and ResNet-50 for FGVC-Aircraft, iNaturalist-19 and tieredImageNet-H. In addition to the CNN-based feature extractors, we compare our method using a transformer-based backbone. We use a pre-trained ViT-MAE for iNaturalist-19 and a ViT-B16 for tiered-ImageNet-H datasets as the feature extractor and freeze the weights, i.e., we only learn a transformation map from the learned embedding space to Hier-COS.

Evaluation Metrics: We follow the same evaluation strategy as [6, 13, 18, 23] and report the top-1 accuracy, MS and AHD@$\{1, 5, 20\}$ for all the baseline methods. Similar to [23], we report the mean and 95% confidence interval derived from the t-distribution with four degrees of freedom for five different seeds. Additionally, we report the proposed HOPS, HOPS@$\{5, 20\}$ for the competing methods. To quantify hierarchical consistency, we also report the full-path accuracy (FPA) [28] for FGVC-Aircraft, CIFAR-100 and iNaturalist-19. We do not report FPA for tieredImageNet-H because the leaf classes in the corresponding hierarchy tree are not at the same level. Unlike previous methods, we do not artificially extend the leaf nodes to be at the same level, therefore, classes at levels $l \in [H]$ are not well defined.

Method	Accuracy (↑)	MS (↓)	AHD@1 (↓)	AHD@5 (↓)	AHD@20 (↓)	HOPS (↑)	HOPS@5 (↑)	HOPS@20 (↑)
Cross Entropy	73.63 ± 0.312	6.95 ± 0.021	1.83 ± 0.023	5.66 ± 0.008	7.29 ± 0.010	0.58 ± 0.001	0.23 ± 0.003	0.14 ± 0.002
Barz & Denzler*	60.27 ± 0.240	6.80 ± 0.019	2.70 ± 0.022	5.48 ± 0.271	6.21 ± 0.005	-	-	-
YOLO-v2*	66.02 ± 0.099	6.99 ± 0.011	2.38 ± 0.012	5.05 ± 0.001	6.17 ± 0.001	-	-	-
HXE+CRM*	73.54 ± 0.150	6.89 ± 0.027	1.82 ± 0.016	4.82 ± 0.006	6.03 ± 0.004	-	-	-
Soft-labels (β=30)*	69.31 ± 0.125	6.99 ± 0.007	2.15 ± 0.008	4.95 ± 0.001	6.11 ± 0.001	-	-	-
Soft-labels (β=4)*	17.28 ± 0.079	7.54 ± 0.001	6.24 ± 0.005	6.94 ± 0.005	7.25 ± 0.002	-	-	-
Flamingo-I	72.27 ± 0.209	6.96 ± 0.016	1.93 ± 0.017	5.77 ± 0.010	7.42 ± 0.012	0.58 ± 0.001	0.22 ± 0.001	0.13 ± 0.002
Flamingo-II	65.72 ± 1.551	7.07 ± 0.036	2.42 ± 0.122	5.79 ± 0.023	7.33 ± 0.018	0.58 ± 0.001	0.22 ± 0.004	0.14 ± 0.002
HAFeat-I	73.49 ± 0.218	6.92 ± 0.027	1.83 ± 0.016	5.55 ± 0.020	6.98 ± 0.017	0.65 ± 0.003	0.28 ± 0.002	0.21 ± 0.003
HAFeat-II	67.89 ± 1.772	7.05 ± 0.016	2.26 ± 0.128	5.62 ± 0.036	6.97 ± 0.015	0.65 ± 0.004	0.26 ± 0.011	0.20 ± 0.007
HAFrame	73.70 ± 0.284	6.90 ± 0.007	1.82 ± 0.019	4.96 ± 0.013	6.16 ± 0.009	0.87 ± 0.003	0.56 ± 0.008	0.60 ± 0.009
Hier-CoS	72.22 ± 0.211	6.70 ± 0.009	1.86 ± 0.012	4.81 ± 0.006	6.02 ± 0.002	0.94 ± 0.001	0.71 ± 0.003	0.76 ± 0.002
ViT-Cross Entropy	75.10 ± 0.079	6.77 ± 0.015	1.69 ± 0.005	5.44 ± 0.002	7.00 ± 0.003	0.62 ± 0.000	0.31 ± 0.001	0.21 ± 0.001
ViT-Hier-CoS	74.71 ± 0.082	6.60 ± 0.023	1.67 ± 0.009	4.82 ± 0.007	6.07 ± 0.003	0.92 ± 0.001	0.65 ± 0.004	0.70 ± 0.004

Table 6. Results comparing the performance of hierarchical classification on the test set of tieredImageNet-H.

10.1. Implementation Details

Inspired by HAFrame [23], the transformation module comprises of 5 linear layers, each with hidden units same as the number of nodes n, along with batch-norm layer and PReLU activation feeding to a fully connected layer with fixed weights $\{e_1 \ldots, e_n\}$ (Fig. 3). A small distinction from [23] is that we define the transformation module f_θ as the Transformation + Linear module of HAFrame. We re-emphasize that the weight vectors (equivalently, the basis set $\mathcal{E}$) can be arbitrary but are always orthonormal and fixed for the fifth linear layer.

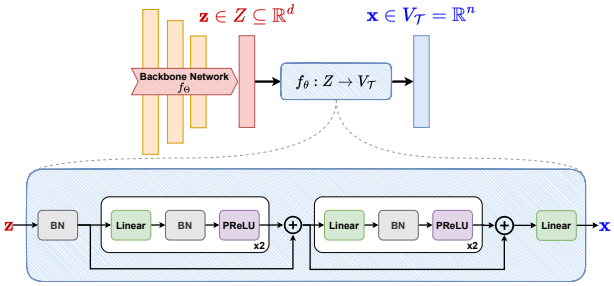


Figure 3. Illustration of the transformation module $f_\theta : Z \to V_\mathcal{T}$. The components of f_θ are inspired from HAFrame [23]

10.2. Training Configuration

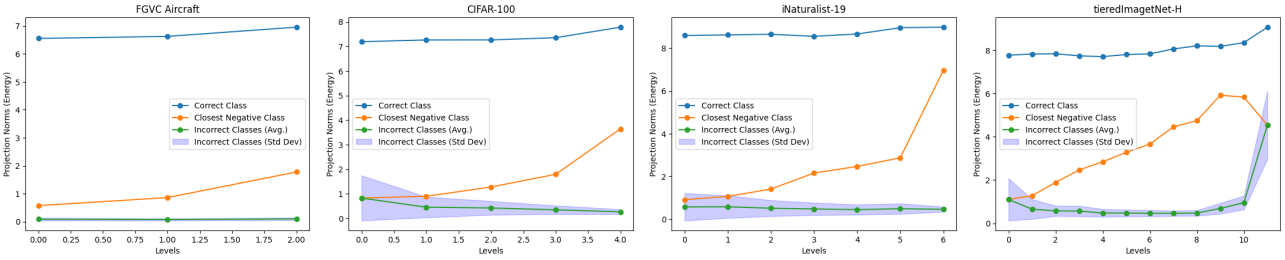
We use the same training configurations as [23]. We obtained the best results on FGVC-Aircraft, CIFAR-100, iNaturalist-19 and tieredImageNet-H using α as $(0.1, 0.05, 0.001, 0.0001)$, respectively. To compare our method using a transformer-based backbone, we freeze the weights of the backbone and only fine-tune f_θ for 10 epochs with $\alpha = 1e-4$.

11. Additional Experiments and Results

In this section, we analyze the performance of Hier-COS on tieredImageNet-H (Table 6) and provide an empirical validation of our model's hierarchical consistency. Here, we observe a reduction of 1.48% in the top-1 accuracy as compared to SOTA, while we observe significant improvements in MS, AHD@$\{5, 20\}$, HOPS and HOPS@$\{5, 20\}$. AHD@1 is slightly reduced because of the degraded top-1 performance. We emphasize that tieredImageNet has a complex hierarchical structure (visualized in `Results and Analysis.pdf`) with leaf nodes at different levels of the hierarchy. This makes learning algorithms difficult to optimize. This is also evident in SOTA. We observe that while HAFrame improves the top-1 performance, its overall performance is not better than the baseline HXE+CRM. The AHD@k metrics for HXE+CRM are significantly better than HAFrame, while we show that with a slight reduction in top-1 performance, Hier-COS is able to achieve better performance across all the hierarchical metrics.

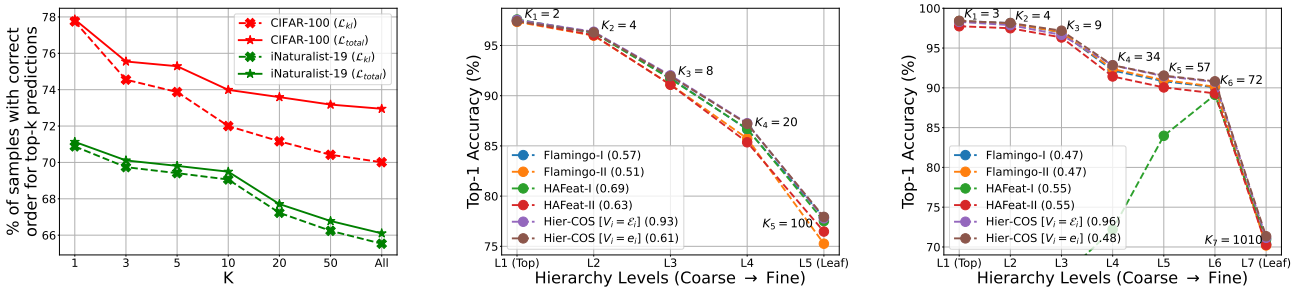
Fig. 1 from the main paper, depicts the improvement in hierarchical performance. We observe a large drop in the percentage of samples with correct prediction orders using all the previous methods for all $k > 1$ in Fig. 1. Although the drop in accuracy is not justified because, unlike [6], we demonstrated via other experiments that there is no trade-off between top-1 and hierarchical performance, we speculate that this occurs due to the design of a simple loss function that does not account for the imbalance in the height of subtrees. Specifically, each class in tieredImageNet-H spans a different number of dimensions, while all the classes in other datasets span exactly H dimensions. We hypothesize that accounting for the dimensionality of subspaces should also improve the top-1 accuracy.

Finally, while Hier-COS theoretically promotes positivity, we validate this property empirically in Fig. 4. By plotting the distribution of projection magnitude (energy) onto the correct basis vectors from root to leaf, we demonstrate that Hier-COS maintains consistent positive energy across all levels, ensuring the structural integrity of the learned hierarchical representations.



(a) (b) (c) (d)

Figure 4. Level-wise projections/energy (zoom-in for clarity)



(a) Impact of $\mathcal{L}_{reg}$ (b) Level-wise accuracy for CIFAR-100 (c) Level-wise accuracy for iNaturalist-19

Figure 5. Ablative analysis examining (a) the impact of the regularization term and (b, c) the effect of adding extra dimensions to V_i. HOPS metric is mentioned in parenthesis in (b, c).

11.1. Ablation Study

In this section, we examine the (i) impact of the regularization term (Fig. 5a), (ii) the effect of adding extra dimensions to the vector space (Fig. 5b and 5c), (iii) the influence of α on the hierarchical performance (Table 7), and (iv) the influence of w_l on the classification performance (Table 8. In Fig. 5a, we plot the percentage of samples with correct order for the top-k predictions against k. We observe that when $\mathcal{L}_{reg}$ is not used (dotted line), the hierarchical performance reduces with k, however, it is still better than the previous methods shown in Fig. 1 of the main paper. Including $\mathcal{L}_{reg}$ minimizes the norm of the projection of a feature vector corresponding to a class y_i onto its complementary subspace $V_i^{\perp} = \text{span}(\mathcal{E} \setminus \mathcal{E}_{y_i})$. Therefore, we would expect the feature vector to be closer to the subspace corresponding to the correct class y_i. We validate this on CIFAR-100 by calculating the cosine similarity between $\mathbf{x}$ and $\mathbb{P}_{\mathcal{E}_{y_i}}\mathbf{x}$. The cosine similarity obtained with and without $\mathcal{L}_{reg}$ is 0.97 and 0.87, respectively.

In Fig. 5b and 5c, we observe that when no additional dimensions are added to the vector space, i.e., $V_i = \text{span}(\{e_i\})$, the level-wise accuracy is competitive with the proposed Hier-COS framework (i.e., $V_i = \text{span}(\mathcal{E}_i)$); however, the HOPS metric is deteriorated significantly. This suggests that extra dimensions help in learning diverse and discriminative features across all the hierarchical levels. Further, we emphasize that, unlike [7, 13], a single classifier is used to predict classes across all the levels. This demonstrates that learned feature representations are generalized across all the hierarchical levels. We discuss this property further in Section 14.

11.2. Sensitivity Analysis

We evaluate the performance of Hier-COS in Table 7 with different values of α on CIFAR-100. We observe that the performance is not too sensitive to α and maintains the same hierarchical performance across all the values with slight variations in top-1 accuracy.

α	Accuracy	MS	AHD@5	AHD@20	HOPS
0	77.51 ± 0.288	2.21 ± 0.026	1.10 ± 0.005	2.17 ± 0.002	0.93 ± 0.001
5e-2	77.79 ± 0.145	2.21 ± 0.017	1.09 ± 0.005	2.17 ± 0.003	0.93 ± 0.001
1e-2	77.39 ± 0.231	2.21 ± 0.049	1.10 ± 0.010	2.18 ± 0.004	0.93 ± 0.002
5e-3	77.43 ± 0.072	2.21 ± 0.008	1.10 ± 0.002	2.18 ± 0.002	0.93 ± 0.001
1e-3	77.48 ± 0.371	2.22 ± 0.020	1.10 ± 0.006	2.18 ± 0.004	0.93 ± 0.002
1e-4	77.41 ± 0.217	2.23 ± 0.019	1.11 ± 0.005	2.18 ± 0.004	0.93 ± 0.001

Table 7. Sensitivity Analysis of α on CIFAR-100

In Table 8, we analyze the impact of distributing the weights w_l such that (i) $w_l > w_{l+1}$, i.e., projection norm is concentrated towards the coarser classes, (ii) $w_l = w_{l+1}$, i.e., projection norm is concentrated equally across all the

levels, and (iii) $w_l < w_{l+1}$, i.e., projection norm is concentrated towards the finer classes. To obtain weights $w_l > w_{l+1}$, we simply reverse the order of the level-wise weights obtained using the equation mentioned in the main paper, i.e., $w_l = \exp\left(\frac{1}{h+1-l}\right)$. Obtaining the uniform weights for all the levels is straightforward and done using $w_l = \frac{1}{h}$. Empirical evidence supports our discussion in Sec. 3.3 of the main paper, that the leaf classes become indistinguishable as the concentration of the projection norm becomes high towards the coarser classes. This also justifies our design choice of a monotonically increasing weight function as we move from the root node's basis vectors to the leaf.

Weights	Accuracy	MS	AHD@1	AHD@5	AHD@20
$w_l > w_{l+1}$	51.08 ± 19.681	1.92 ± 0.070	0.93 ± 0.338	1.29 ± 0.156	2.24 ± 0.056
$w_l = w_{l+1}$	70.22 ± 2.860	2.08 ± 0.019	0.62 ± 0.055	1.16 ± 0.029	2.19 ± 0.010
$w_l < w_{l+1}$	77.79 ± 0.145	2.21 ± 0.017	0.49 ± 0.006	1.09 ± 0.005	2.17 ± 0.003

Table 8. Impact of the distribution of weights w_l on Top-1 accuracy and other hierarchical metrics

We further investigate the sensitivity of the HOPS metric by defining a generic decay function $\eta_j = b^{-z_j}$, where $b \in \{1.5, 2, 3, 5\}$ controls the penalty for high-severity mistakes. Additionally, we compare three multi-step decay functions: (i) exp-linear (solid), (ii) exp-log (dashed), and (iii) exp-exp (dotted). As shown in Fig. 6(a,b), the HOPS-based performance ranking of methods remains invariant across different datasets (iNat-19 and tieredImageNet-H) and values of b, with Hier-COS consistently achieving superior performance. These results confirm the stability of relative HOPS scores regardless of the decay constant η_j or the specific tree topology. Similar trends are observed for HOPS@$\{5, 10, 20\}$. Furthermore, Fig. 6(c) suggests that the weights generated by the exp-linear decay are more representative of hierarchical preference, as the exp-log and exp-exp functions reduce weights too abruptly to maintain discriminative value.

11.3. Qualitative Analysis

In this section, we demonstrate that our method learns stronger hierarchical feature representations as compared to the previous methods. Fig. 1 of the main paper, only highlights the overall hierarchical performance of the methods, where even a small mistake would result in an error. For a deeper analysis of the learned representations, for all the samples belonging to a class y_i, we visualize the average of the log probabilities for the complete prediction vector comprising of K classes. A higher log probability denotes a higher preference (or a low inter-class distance). We explain these plots using the ground truth LCA-based log probabilities on the test set of CIFAR-100 in Fig. 7, representing the best hierarchical performance, and demonstrate the hierarchical performance across all the datasets in Fig. 8. We

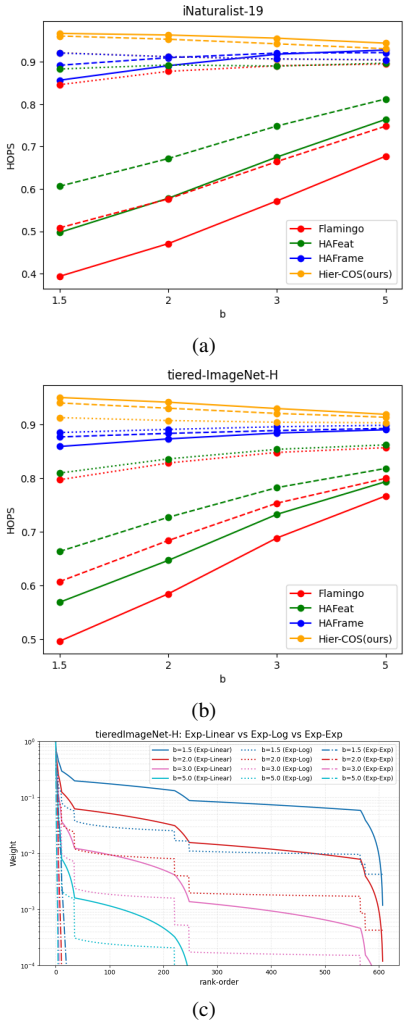


Figure 6. HOPS Sensitivity Analysis (zoom-in for clarity)

observe that Hier-COS is able to learn better hierarchical representations than any of the other baseline methods.

11.4. Time and Memory Complexity Analysis

To evaluate the computational efficiency, we provide a comparative analysis of the training time and memory overhead. As observed in our empirical profiling, Hier-COS demonstrates a significantly reduced temporal footprint. Specifically, it requires only $0.48\times$ and $0.66\times$ the training time of HAFrame on the CIFAR-100 and iNaturalist-19 datasets, respectively. For hardware resource allocation, we monitor peak utilization ratios for both VRAM and RAM. For CIFAR-100, Hier-COS maintains a (VRAM, RAM) ratio of $(0.97\times, 1.59\times)$ relative to the baseline, while on the larger-scale iNaturalist-19 dataset, the ratio is $(1.12\times, 0.67\times)$. These results indicate that while our features are higher-dimensional, we remain computationally efficient and scalable across varying dataset complexities.

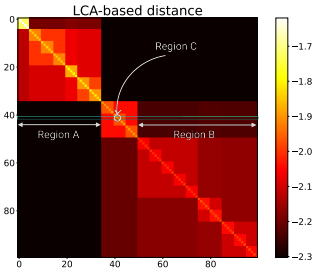
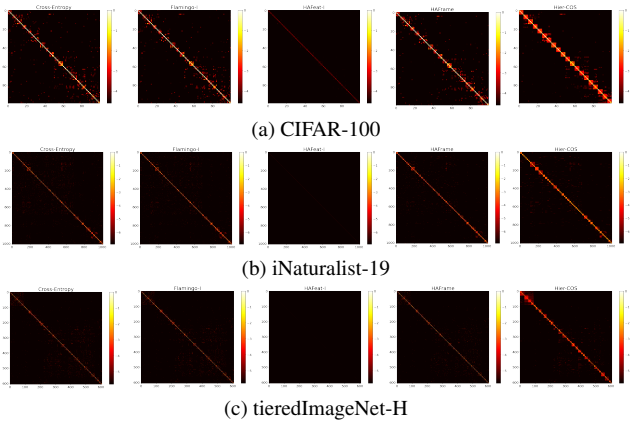


Figure 7. Ground truth LCA-based log probabilities on the test set of CIFAR-100 representing the best hierarchical performance. Consider the row highlighted by the rectangular box, say class y_c. 'Region A' contains the classes farthest from the class y_c because they have the least log probability in that row. Similarly, 'Region B' contains all the classes closer to y_c than the classes in 'Region A' but are farther than all the other classes that are not in 'Region A' and 'Region B'. 'Region C' contains only one class, i.e., y_c, and represents the average log probability of being correct. The color scale of log probabilities helps in understanding the hierarchical preference order. Therefore, for learned hierarchy-aware feature representation, we would expect the log probabilities to follow a similar preference order.



(a) CIFAR-100

(b) iNaturalist-19

(c) tieredImageNet-H

Figure 8. **(Zoom in for better clarity)** Plots showing the obtained order from different classification methods on the test set of (a) CIFAR-100, (b) iNaturalist-19 and (c) tieredImageNet-H. The rows and columns represent the K classes. Each row contains the softmax-based log probabilities for all the K classes averaged over all the samples belonging to that class. All the classes are grouped according to the LCA distance.

12. A Review of Evaluation Metrics

This section discusses some of the widely used and related evaluation metrics for hierarchical classification methods. This also includes metrics used for evaluating information retrieval and recommendation systems.

12.1. Definitions

Mistake Severity: The severity of a mistake is often quantified using the hierarchical distance of a mistake, i.e., the LCA distance between the ground truth and the predicted class when the input is misclassified [6].

$$MS = \frac{1}{FP + FN} \sum_{i=1}^{|\mathcal{X}|} height(LCA(y_i, \widehat{y}_i))$$

where y_i and $\widehat{y}_i$ are the ground truth and the predicted class label at the finest level of granularity for a sample i in the dataset and $|\mathcal{X}|$ is the total number of samples in the dataset $\mathcal{X}$. The MS is averaged over all misclassifications, i.e., the total number of false positives (FP) and false negatives (FN). It is important to note that the hierarchical distance for correct predictions is zero, and hence, averaging it over only FP and FN makes it equivalent to saying that MS is computed for misclassified samples only.

Average Hierarchical Distance @k: A major drawback of the MS metric was pointed out by [18] that it only considers the misclassified samples and hence needs to be paired with top-1 accuracy for comparing different methods. A simple extension of MS is to average over all the samples $|\mathcal{X}|$ instead of only the misclassified ones (FP + FN) [6].

$$AHD = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} height(LCA(y_i, \widehat{y}_i))$$

In the case of hierarchy-aware feature representations, we would expect not only lower MS and AHD values but also smaller LCA distances between the top predictions – an aspect that neither MS nor AHD adequately captures. Therefore, [6] also proposes to compute the average AHD for all the top-k predictions, defined as:

$$AHD@k = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \frac{1}{k} \sum_{j=1}^{k} height(LCA(y_i, \widehat{y}_{ij}))$$

where $\widehat{y}_{ij}$ is the top-j^{th} ranked prediction for sample i.
Hierarchical Precision and Recall: For evaluating hierarchical classification methods, hierarchical precision (hP) is used, which is the proportion of correctly predicted classes among all the predicted classes at all the hierarchical levels,

$$hP = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \frac{|\tilde{f}_a(v_{\widehat{y}_i}) \cap \tilde{f}_a(v_{y_i})|}{|\tilde{f}_a(v_{\widehat{y}_i})|}$$

where y_i and $\widehat{y}_i$ are the ground truth and predicted class labels, respectively, v_{y_i} and $v_{\widehat{y}_i}$ are the corresponding nodes

in the tree and $\tilde{f}_a(v_i)$ is the set containing all the ancestors of v_i and itslef, i.e., $\{f_a(v_i) \cup v_i\}$ [17].

Similarly, for any sample i, hierarchical recall (hR) is the proportion of correctly predicted classes among all the ground truth classes at all the hierarchical levels.

$$hR = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \frac{|\tilde{f}_a(v_{\hat{y}_i}) \cap \tilde{f}_a(v_{y_i})|}{|\tilde{f}_a(v_{y_i})|}$$

In simpler words, we can say that hR is analogous to hP but focuses on how well the true hierarchical structure is retrieved rather than the correctness of the predicted structure.

Total Preference Ordering: We often deal with information expressed as partial orderings (PO). For PO based on a hierarchy tree, the highest preference is given to the ground truth class, the second highest to all its siblings, the third highest to all its first cousins, and so on, resulting in a total preference ordering (TPO) [10]. There could also be other TPOs, such as TPO based on the prediction probabilities. [10] presents a metric to compare TPOs using a pair-wise Preference-Score Matrix (PSM), M, for each TPO, defined as:

$$M(i,j) = \begin{cases} 1, & \text{if } x_i \succ x_j, \\ -1, & \text{if } x_i \prec x_j, \\ 0, & \text{if } x_i = x_j. \end{cases}$$

where x_i and x_j denote the preference scores of a classes i and j, respectively, and $M(i,j)$ is the component of PSM representing whether the classes i and j are in the correct preference order. Finally, [10] defines the distance between the two TPOs using the Frobenius distance as follows:

$$\begin{aligned} d_F(M_1, M_2) &= \|M_1 - M_2\|_F \\ &= \sqrt{\text{Tr}\left((M_1 - M_2)^T (M_1 - M_2)\right)} \end{aligned}$$

Mean Reciprocal Ratio and Normalized Rank: A popular evaluation metric used in information retrieval and ranking systems is Mean Reciprocal Rank (MRR). We present its definition with respect to our problem setup. As the name suggests, it measures the mean of the reciprocal of the ranks for the correct class. Specifically, rank can be considered as the position or index at which the correct class appears in an ordered list of predictions. The lowest rank, zero, signifies the prediction with the highest confidence.

$$MRR = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \frac{1}{rank_i}$$

Recently, Tian et al. [34] proposed an alternative to MRR for hierarchical classification called Mean Normalized Rank (MNR). We present its definition with respect to multi-class classification below:

$$MNR = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \left(\frac{1}{H} \sum_{l=1}^{H} \left(\frac{rank_i^l}{K_l} \right) \right)$$

In simpler words, MNR is the average rank over all the hierarchical levels normalized by the number of classes at each level.

Normalised Discounted Cumulative Gain: Normalized Discounted Cumulative Gain (NDCG) is a standard metric that evaluates the quality of recommendation and information retrieval systems. NDCG measures the ability of a method to sort items based on relevance. The relevance score is inversely proportional to ranks, i.e., the higher relevance score is better. The NDGC@k computes the NDCG for top-k predictions, given for each sample i by:

$$\begin{aligned} NDCG_i@k &= \frac{DCG_i@k}{IDCG_i@k} \\ DCG_i@k &= \sum_{j=1}^{k} \frac{rel_{i,j}}{log_2(j+1)} \end{aligned}$$

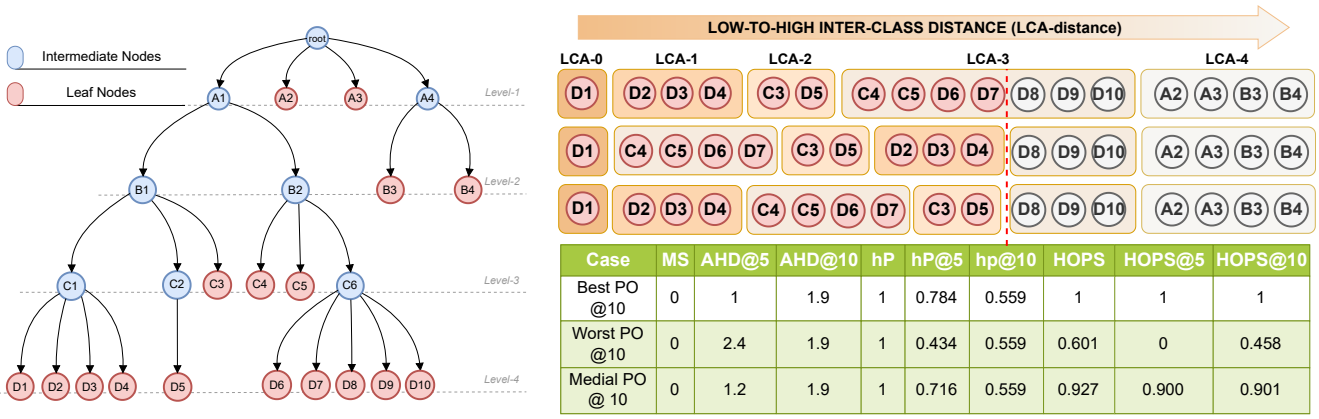
where, $IDCG@k$ is the DCG@k for the ideal ranking. Recently, Tian et al. [34] presented an approach to extend this metric to hierarchical classification, where relevance between classes i and j is defined based on the structure of the hierarchy tree as:

$$rel_{i,j} = 1 - \frac{d(i, lca(i,j)) + d(j, lca(i,j))}{D_{\mathcal{T}}}$$

where, i is the ground truth class, j is the predicted class, $lca(i,j)$ is the lowest common ancestor between i and j, $d(i,j)$ is the number of edges between the nodes i and j, and $D_{\mathcal{T}}$ is the length of the longest path between any two leaf nodes.

12.2. Limitations

Mistake Severity: We emphasize that MS only considers the average cost of mistakes and is biased towards methods that make *more* mistakes in numbers but less in severity [18]. Therefore, it can only be compared when coupled with the top-1 accuracy. This is also evident for the 'Soft-labels $\beta = 4$' and 'Barz & Denzler*' methods in Tables 2 and 3 of the main paper, respectively. Moreover, the metric is dependent on the properties of the tree and is not normalized. Because the MS is unnormalized, this is often interpreted as the average LCA distance for a mistake. However, for a given class, the lowest MS for misclassification is dependent on the LCA distance of the nearest sibling, which



Case	MS	AHD@5	AHD@10	hP	hP@5	hp@10	HOPS	HOPS@5	HOPS@10
Best PO @10	0	1	1.9	1	0.784	0.559	1	1	1
Worst PO @10	0	2.4	1.9	1	0.434	0.559	0.601	0	0.458
Medial PO @ 10	0	1.2	1.9	1	0.716	0.559	0.927	0.900	0.901

Figure 9. (Left) The same taxonomy $\mathcal{T}$ shown in Fig. 2 of the main paper. (Right) Three examples with correct class prediction but varying the predicted preference order for top-10 predictions.

results in a higher MS for imbalanced trees, even for less severe mistakes. For instance, whenever classes $A2$ and $A3$ of the taxonomy $\mathcal{T}$ in Fig 9 are misclassified, the MS is expected to be high because most incorrect classes are at a higher LCA distance. Therefore, we discover that MS depends on the leaf node's depth, tree's branching factor and imbalance, and it is challenging to interpret the MS score without due consideration to the tree structure.

AHD@k: AHD@1 can be viewed as a straightforward extension of MS that does not require pairing with top-1 accuracy. However, as highlighted by [13], this metric tends to be biased towards higher top-1 accuracy and fails to effectively measure the severity of mistakes. Moreover, similar to MS, AHD@k is also unnormalized and suffers from the limitations discussed above. For instance, although the predictions are correct, AHD@$\{5, 10\}$ is non-zero for all the cases in the Fig. 9, which depends on the structure of the tree. Moreover, the average operation is permutation invariant; therefore, any random permutation of the same top-k prediction results in the same AHD@k. For instance, AHD@10 for any random permutation of the top-10 predictions is always 1.9, as shown in Fig. 9. Therefore, we discover that AHD@k also depends on the properties of the tree and is invariant to the order of predictions.

hP and hR: Similar to AHD@k, we can also use the average hP and hR of the top-k predictions to get hP@k and hR@k, respectively. Both hP@k and hR@k can be viewed as alternatives to AHD@k that are normalized. Observe that $|\tilde{f}_a(v_{\widehat{y}_i}) \cap \tilde{f}_a(v_{y_i})|$ measures the LCA similarity between the ground truth and the predicted classes, i.e., as the number of common nodes increases the similarity increases; therefore, a higher value represents better performance. However, because of their resemblance to AHD@k, both possess similar limitations, i.e., both depend on the tree's properties and are permutation invariant. Moreover, for a balanced tree having the leaf nodes at the same level, we know that $|\tilde{f}_a(v_{\widehat{y}_i})| = |\tilde{f}_a(v_{y_i})|$, therefore, $hP = hR$. In Fig 9, it is evident that hP@k is a normalized version of AHD@k.

TPO: This metric fundamentally depends on the definition of TPOs and their corresponding PSMs. We can use the hierarchy tree to define a ground truth PSM, say M_1. However, determining an order of preferences from the prediction is not well-defined when we have multiple classes at the same preference value, which is almost always the case with hierarchies. Therefore, although [10] presents a promising direction to compare the TPOs, it is not yet feasible for hierarchical evaluations. Even if we assume a method exists to determine the order of preferences from the prediction, using this method for hierarchical evaluation is still challenging. The construction of PSM considers all misclassifications equally severe. Moreover, deciding which classes (based on ground truth TPO) are more important when considering a hierarchy with a huge number of leaf nodes is often important.

MRR and MNR: MRR only considers the rank of the correct prediction, while ignoring all the negative classes. Therefore, it is analogous to AHD@1 and, hence, suffers from the same problems. Similarly, MNR is also computed over the correct prediction at all levels and has the same limitations. Further, MNR requires a ranking over all the hierarchical levels. Hence, it can not be used for methods that predict for a single level, like Cross-Entropy and HAFrame.

NDCG@k: The definition of the relevance given by [34] does not hold for trees that do not have leaf nodes at the same level. According to them, $d(D1, A2) < d(D1, D6)$ in Fig 9, however, we can see that $D1$ is closer to $D6$ than $A2$. Moreover, the discounting factor, $\frac{1}{log_2(j+1)}$ for each class with rank j, is independent of the hierarchical structure. Therefore, NDCG@k does not capture the properties of the hierarchy tree. Specifically, for a class that has all the nearest-k neighbors at the same LCA distance, if the pre-

dicted class at $(k-1)^{\text{th}}$ rank has a higher LCA distance, the penalty is decayed significantly ($\frac{1}{log_2(k)}$) without considering that all the top-k predictions are equally relevant.

13. HOPS: Technical Description & Example

Let $\mathcal{U}_{d,y_c}$ be the set of classes at the finest level that are at LCA distance d from y_c. Note that $\mathcal{U}_{0,y_c}$ is a singleton set with the only element being $\{y_c\}$, i.e., the correct class. We emphasize that the cost of misclassifying y_c as any of the classes in $\mathcal{U}_{d,y_c}$ is the same, i.e., d. Therefore, an order on $\mathcal{U}_{d,y_c}$ can be used to define the preferences of classes where any order within a set $\mathcal{U}_{d,y_c}$ is equally preferred, i.e., the preferred ordering will be $\{\mathcal{U}_{0,y_c}, \cdots, \mathcal{U}_{H,y_c}\}$. In the case when a node only has one child, for some value $d = l, \mathcal{U}_{l,y_c}$ will be empty and will be of no significance. In order to ignore such empty sets, we use ranks instead of absolute LCA distances to represent the ordering. Specifically, we use an index set $\mathbb{I}$ to rank all the non-empty sets in $\{\mathcal{U}_{d,y_c}\}_{d=0}^{d=H}$. We define the index set $\mathbb{I} = \{0, 1, 2, \ldots, H\}$ for indexing the elements in the ordered set $\mathbb{S}_{y_c}$, defined as

$$\mathbb{S}_{y_c} = \{\mathbb{S}^i_{y_c} \mid i \in \mathbb{I}\} := \{\mathcal{U}_{d,y_c} \mid \mathcal{U}_{d,y_c} \neq \phi\}_{d=0}^{d=H}. \quad (15)$$

Now, we define the *desired order* z for a class y_c as:

$$z = [0, \underbrace{1, \ldots, 1}_{|\mathbb{S}^1_{y_c}| \text{ times}}, \cdots, \underbrace{k, \ldots, k}_{|\mathbb{S}^k_{y_c}| \text{ times}}] \quad (16)$$

Note that this preference order is conditioned on the true class y_c. $|\mathbb{S}^i_{y_c}|$ denotes the number of classes at rank i; therefore, z encodes the properties of the tree via ranks.

Let $\Pi = [\pi_{y_1}, \cdots, \pi_{y_K}]$ be the vector of probabilities corresponding to all the leaf classes and $\hat{\Pi} = \arg\text{sort}(\Pi)$ be the vector of indices (class labels) that would sort Π in descending order. Assume $rank(y_c, y_j)$ returns the index $i \in \mathbb{I}$ of the set $\mathcal{U}_{d,y_c}$ that contains the class y_j. Finally, we define the *predicted order* $\widehat{z}$ as follows:

$$\widehat{z} = [rank(y_c, \hat{\Pi}_j)]_{j=1}^{K} \quad (17)$$

By defining a metric on the *desired* and *predicted* preference orders, z and $\widehat{z}$ respectively, we can derive a hierarchical metric that takes into account the branching factor and depth (or height) of the tree. Specifically, we compute the following for a single sample having the true class y_c as:

$$s = \sum_{j=1}^{K} \eta_j \cdot |z_j - \widehat{z}_j| \quad (18)$$

where, η_j's are defined using a multi-step exponential-linear decay function. Specifically, we use the exponential decay when the preferred rank changes, i.e., $\eta_j = 2^{-z_j}$ when j is the first occurrence of rank z_j, and for each subsequent values j for rank=z_j, we use a linear decay weight by linearly interpolating between 2^{-z_j} and $2^{-(z_j+1)}$. An exponential decay ensures that the classes with higher LCA distances, when ranked closer to the predicted class, have a relatively high negative impact on the score; thereby accounting for the imbalance of the tree. Meanwhile, the linear decay accounts for the branching factor by reducing the weights for classes at the same rank as we move away from the correct class. The rationale behind the choice of this decay function is detailed in the supplementary.

Note that we get $s_{min} = 0$ when $z = \widehat{z}$, and we use the worst possible predicted order, i.e., the reverse order of z, to get the maximum achievable score s_{max}. The final measure of HOPS is given by:

$$\text{HOPS} = 1 - \frac{s}{s_{max}} \quad \in [0, 1] \quad (19)$$

HOPS@k. We are often interested in only the top-k performance of a classifier, particularly, when there are a large number of classes. We define the HOPS@k variants, simply by modifying the notion of the worst achievable score and adapting the weights. For HOPS@k, we define the worst ranked vector $z^*_r = [z_{k:-1:1}, z_{k+1:1:K}]$ [3] yielding the score as s^k_{max} which is then used to compute HOPS@k:

$$\text{HOPS@}k = \max\left(0, 1 - \frac{s^k}{s^k_{max}}\right) \quad (20)$$

where s^k is computed using (18), but with $\eta_{j>k} = 0$. The scores averaged over all samples are used for reporting the classifiers performance. It is worth pointing out that for $k = 1$, the HOPS@k is equal to the top-1 accuracy. This is aligned with the design of this metric, where the HOPS@k concerns with the ranking performance of up to the k^{th} ranked prediction, while the HOPS metric concerns the complete ranking over all classes.

13.1. Example

As discussed in the previous section, for a given ground-truth class y_c, we get a *preference order* based on the hierarchy tree by constructing z as:

$$z = [0, \underbrace{1, \ldots, 1}_{\times n_1}, \cdots, \underbrace{k, \ldots, k}_{\times n_k}] \quad , \; k \leq H \quad (21)$$

where, n_k denotes the number of $(k-1)^{th}$-*nearest* higher order cousins of class y_c and $n_0 = 1$ (self). This implies that z is the *desired* preference ordering where all the classes are arranged according to the hierarchy-based preferences such that a lower value of z_j denotes a higher preference (similar to ranks). We then propose to construct $\widehat{z}$ using

[3] In $z_{a:c:b}$, a is the start, b is the stop and c is the step-size

the predictions such that it contains the preference values of the predicted classes based on z. For instance, z and $\widehat{z}$ corresponding to the second row in Fig 9 will be:

$$z = [0, 1, 1, 1, 2, 2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4]$$
$$\widehat{z} = [0, 3, 3, 3, 3, 2, 2, 1, 1, 1, 3, 3, 3, 4, 4, 4, 4]$$

14. Additional Discussions

Computational Overhead: In the worst case, the complexity for a complete b-ary tree is $\mathcal{O}(\frac{bK-1}{b-1}) \approx \mathcal{O}(K)$. However, in the real world, the complexity is much lower because the hierarchy trees are rarely complete. The values of (K, n) for all the datasets is given in table Table 5. Moreover, unlike Flamingo and HAFeat, we do not append dummy nodes to have all the leaf nodes at the same level. Additionally, we do not train separate classifiers for each hierarchical level. Hence, we are resource-efficient when compared to these methods.

Reducing Complexity when Severity of Mistakes is not important: The ablation experiments suggested that the level-wise accuracy for all the datasets is maintained even when the subspaces are defined using $V_i = \text{span}(\{e_i\})$, while the severity of mistakes is compromised. This is because all the leaf classes are orthogonal and have no hierarchical awareness. As mentioned earlier, the Hier-COS framework learns a single classifier that is capable of classifying at all levels, we can reduce the complexity of the network if the severity of mistakes is not conccerned. Specifically, for predicting classes at level l, we can prune the network by removing all the weights corresponding to $\{\mathcal{E} \setminus \mathcal{E}^{(l)}\}$.

Extending to DAGs and Arbitrary Groups: Previous methods [7, 13] learn the semantic relationship between the leaf classes by learning a separate classifier for every level in the hierarchy tree using a multi-class classification loss with additional constraints. However, in case the relationship is not represented as a tree [2, 29, 33], these methods can not be used and are non-trivial to extend. HAFrame [23] fixes the frames based on the LCA distance between leaf nodes. However, in the case of a Directed Acyclic Graph (DAG), there can be multiple lowest common ancestors that makes this formulation ambiguous and non-trivial to apply in such a setting. In contrast, we formulate Hier-COS based on the number of common ancestors between any two leaf nodes, which is unambiguous for DAGs or any arbitrary grouping that defines the hierarchical structure.

Designing Kernel Tricks: Kernel methods have been widely used in various domains as they allow embeddings to be learned in a very high dimensional space while actually operating in a lower dimensional feature space. At the core, kernel methods use kernel functions that compute the inner product in a lower-dimensional feature space while implicitly transforming the feature embeddings into a higher-dimensional vector space where patterns become easier to recognize. We re-iterate that Hier-COS is a framework that defines a vector space $V_{\mathcal{T}}$ which implicitly captures the properties of the hierarchical structure. Therefore, we hypothesize that kernel tricks can be designed for implicitly transforming the feature vectors from Z into $V_{\mathcal{T}}$ without operating in the n-dimensional vector space. This will especially be beneficial when the number of leaf nodes is very high and the given semantic relationship is complex.

15. Additional Supplementary

This section summarizes all the supplementary files shared with the main paper. In `Results and Analysis.pdf`, we share pre-computed results of our analysis on CIFAR-100. It includes visualization of hierarchy trees for all the datasets (demonstrating the complexity, imbalance, etc.), error & metric analysis, visualization of the weights $\mathbf{w}$ used for HOPS and HOPS@k (depicting the impact of proposed weight decay function), and results of FPA computed based on the approach presented by HAFeat. We will release the code upon acceptance.