Hierarchical Image Classification using Entailment Cone Embeddings

Ankit Dhall[1], Anastasia Makarova[1], Octavian Ganea[2], Dario Pavllo[1], Michael Greeff[1], Andreas Krause[1]
[1]ETH Zurich [2]MIT
adhall@ethz.ch, anastasiia.makarova@inf.ethz.ch, oct@mit.edu
dario.pavllo@inf.ethz.ch, michael.greeff@usys.ethz.ch, krausea@ethz.ch

Abstract

Image classification has been studied extensively, but there has been limited work in using unconventional, external guidance other than traditional image-label pairs for training. We present a set of methods for leveraging information about the semantic hierarchy embedded in class labels. We first inject label-hierarchy knowledge into an arbitrary CNN-based classifier and empirically show that availability of such external semantic information in conjunction with the visual semantics from images boosts overall performance. Taking a step further in this direction, we model more explicitly the label-label and label-image interactions using order-preserving embeddings governed by both Euclidean and hyperbolic geometries, prevalent in natural language, and tailor them to hierarchical image classification and representation learning. We empirically validate all the models on the hierarchical ETHEC dataset.

1. Introduction

In deep learning, classification is typically performed by independently predicting class-probabilities (e.g., using a linear-softmax layer) and predicting the highest scoring label. Such an approach by default assumes mutually exclusive, unstructured labels. Contrary to this assumption, in many common datasets, labels have an underlying latent organization, potentially allowing hierarchical clustering into progressively more abstract concepts. Relatively few previous works use hierarchical information in the context of computer vision. Among them, in [2] the label-hierarchy from WordNet [3] is used to consolidate data across datasets. [4] show how to optimize the trade-off between accuracy and fine-grained-ness of the predicted label, but their proposed method only considers the semantic similarity and disregards visual similarity. [5] use relation graph information to improve performance over a strong baseline in a zero-shot learning setting.

Incorporating the hierarchy in the model would improve generalization on classes for which training data is scarce, by leveraging shared features among hierarchically-related classes, e.g. “truck” and “car” both have wheels in their shared superclass “vehicle”. As is the case with *few-shot learning* approaches, sharing information and parameters among the long tail of leaf labels helps overcome this data scarcity problem.

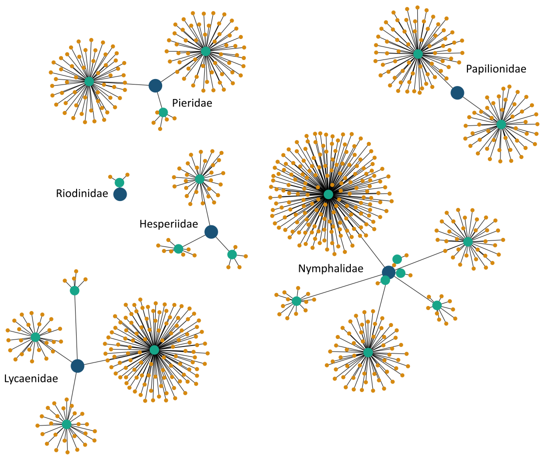


Figure 1: Hierarchy of labels from the ETHEC dataset [1] across 4 levels: family (blue), sub-family (aqua), genus (brown) and species. For clarity, this visualisation depicts only the first 3 levels. The name of the family is displayed next to its sub-tree. Edges represent direct relations.

Uncovering the black-box model. If a human is tasked with classifying an image, the natural way to proceed is to identify the membership of the image to abstract labels and then move to more fine-grained labels. Even if an untrained eye cannot tell apart an *Alaskan Malamute* from a *Siberian Husky*, it is more likely to at least get the concept of “animal” and its sub-concept “dog” correct.

Using the label hierarchy to guide the classification models we are able to bridge one gap in the way machines and humans deal with visual understanding. Incorporating such auxiliary information improves explainability and interpretability of image understanding models.

Leveraging label-label interactions. Usually, image classifiers perform flat N-way classification solely by learning to discriminate between visual signals. These models capture the label-image interactions but do not use additional information available about the inter-label interaction that could boost performance and interpretability.

Long-tailed data distributions. Real-world data is commonly characterized by imbalance. Class labels form a hierarchy and can be viewed as directed acyclic graph (DAG), where abstract labels have finer-grained descendants. Abstract levels have fewer labels and more images per label compared to their fine-grained descendants. The converse is true for fine-grained labels resulting in a long-tailed data distribution. Shallow classifiers benefit from balanced datasets, and generalize worse when classes are imbalanced. We show that image classifiers can exploit information naturally shared across data from different levels and labels.

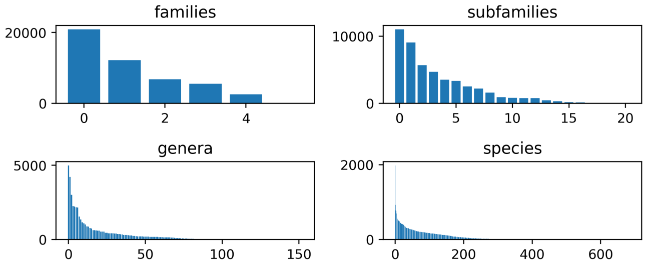


Figure 2: Long-tailedness is evident from the image distribution across labels from the 4 levels of our hierarchy: 6 *family*, 21 *sub-family*, 135 *genus* and 550 *species*. x-axis: number of images for a particular label; y-axis: label. Genus and species labels have been omitted for clarity.

Visual similarity does not imply semantic similarity. Visual models rely on image-based features to distinguish between different objects. But, often, semantically related classes might exhibit marked visual dissimilarity. Sometimes it might even be the case that the intra-class variance of visual features for a single label is larger than the inter-class variance (we show an example in the Appendix, Fig. 14). In such scenarios learned representations for two instances with different visual appearance would be coerced away from each other, indirectly affecting the image understanding capability of the model.

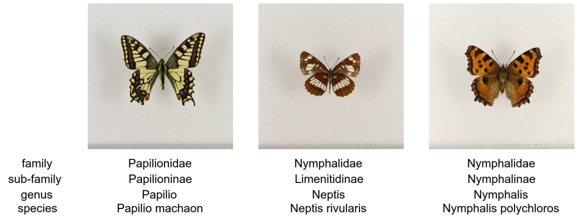


Figure 3: Sample images and their 4-level labels from the ETHEC dataset [1]. The dataset consists of 47,978 butterfly specimens with 723 labels spread across 4 levels.

Labels with varying levels of abstraction may also be beneficial for further downstream tasks involving both natural language and computer vision such as image captioning, scene graph generation and visual-question answering (VQA). This work exploits semantic information available in the form of hierarchical labels. We show that visual models trained with such guidance outperform a hierarchy-agnostic model. We also show how these models can be more interpretable when using more explicit representations via embeddings for the task of image classification.

Our work. We propose and compare multiple approaches for incorporating hierarchical information in state-of-the-art CNN classifiers. To this end, we first compare baselines where the hierarchy is exploited in the loss function (hierarchical softmax, marginalization classifier), and then propose a set of *embedding-based* approaches where images and labels are embedded in a common space. These are more flexible as they allow for entailment prediction tasks and hierarchy-based retrieval. Our embeddings are based on entailment cones, which can be embedded both in Euclidean geometry and in hyperbolic geometry. We compare these and show that the hyperbolic case has empirical advantages over the Euclidean case, while being backed up by theoretical advantages.

We summarize our contributions: (1) applying order-preserving embeddings to image classification, where both images and labels are embedded in a common space that enforces transitivity, (2) providing a set of methods to incorporate entailment cones in CNN-based classifers, including effective optimization techniques. (3) comparing entailment cones in different geometries (Euclidean and hyperbolic), highlighting their strengths and weaknesses, (4) comparing embedding-based approaches to non-embedding-based approaches, under uniform settings.

2. Related Work

Embedding-based models for text. One way to model semantic hierarchies is to use *order-preserving embeddings*, which enforce transitivity among hierarchically-related concepts by imposing a structure on the latent space. For instance, *order-embeddings* [6] learn hierarchical word embeddings on WordNet [3]. As an alternative to common symmetric distances (e.g. Euclidean, Manhattan, or cosine), the work proposes an asymmetric distance resulting in the formation of a transitive embedding space as shown in Fig. 4. As opposed to the distance-preserving nature, the order-preserving nature of order-embeddings ensures that anti-symmetric and transitive relations can be captured well without having to rely on physical closeness between points. However, the distance function in [6] is limited as each concept occupies a large volume in the embedding space irrespective of its volume needs and suffers from heavy orthant intersections. This ill-effect is amplified especially in extremely low dimensions such as $\mathbb{R}^2$. To this end, [7] proposes *Euclidean entailment cones* which generalizes order-embeddings by substituting translated orthants with more flexible convex cones. Furthermore, [8] generalizes order-embeddings [6] and entailment cones [7] for embedding DAGs with an exponentially-increasing number of nodes.

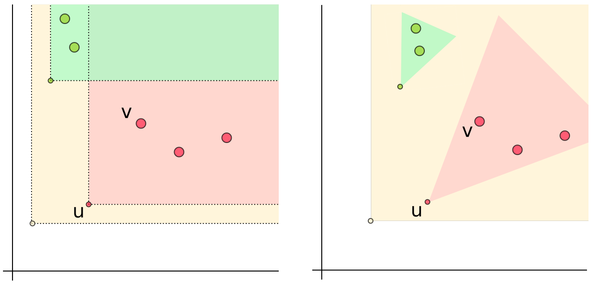
More general and flexible methods where the embedding space is not necessarily Euclidean have also been explored. [7] leverage non-Euclidean geometry by learning embeddings defined by *hyperbolic cones* for hypernymy prediction in the WordNet hierarchy [3]. In hyperbolic space, the volume of a ball grows exponentially with the radius as compared to polynomially in Euclidean space, allowing to embed exponentially-growing hierarchies in low-dimensional space. Lately, [9] combined the idea of Hearst patterns to create a graph and hyperbolic embeddings to infer and embed hypernyms from text. *Hyperbolic neural networks* [10] are feed-forward neural networks parameterized in hyperbolic space that allow using hyperbolic embeddings for NLP tasks more naturally and boost the performance.

Other non-Euclidean embeddings include embeddings on surfaces, generalized multidimensional scaling on the sphere and probability embeddings [11, 12] which generalize point embeddings.

Embedding-based models for images. Visual-semantic embeddings, proposed in [13], define a similarity measure instead of an explicit classification and return the closest concept in the embedding space for a given query. They use an LSTM and a CNN and map to a joint embeddings space through a linear mapping and measure similarity for cross-modal image-caption retrieval. [14] maps images onto class embeddings and use dot product to measure similarity. A drawback of such an approach is that the label embeddings are fixed when training on the image embeddings. The labels might be embedded properly however they might not be arranged in a way that puts visually similar labels together. Furthermore, these approaches are based on Euclidean geometry.

In contrast to general CNNs for image classification, the work done in [15] exploits unannotated text in addition to the images labels. They use embeddings and transfer knowledge from the text-domain to a model for visual recognition and perform zero-shot classification on an extended ImageNet dataset [16].

Non-embedding-based approaches. While this work focuses on embedding-based approaches, there has also been work on incorporating label hierarchies in the model architecture *or* in loss function. [17, 18, 19] discuss hierarchical approaches not based on the concept of order-preserving embeddings. While these approaches can effectively exploit label hierarchies to improve performance, their hierarchies are typically fixed, integrated in the architecture of the model, and tailored to one specific downstream task (e.g. classification). On the other hand, embedding-based approaches allow for flexible hierarchies and retrieval tasks using parent-child queries.



(a) In OE, if v is u, it lies within an orthant at u.

(b) In EC, if v is u, it lies within a cone at u.

Figure 4: Comparing embedding space for OE and EC.

3. Background

Order-embeddings (OE). Order-embeddings [6] preserves the *order* between objects rather than distance. From a set of ordered-pairs $\mathcal{P}$ and unordered-pairs $\mathcal{N}$ the goal is to determine if an arbitrary pair is ordered. They use a reversed product order on $\mathbb{R}^N$: $y \preceq x$ if and only if $\bigwedge_{i=1}^{N} y_i \geq x_i$ and *approximate* order-violation minimization.

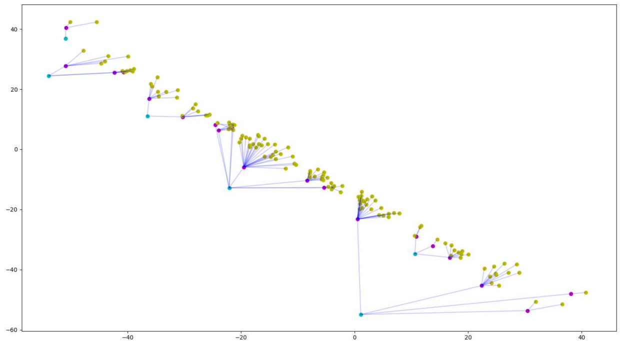


Figure 5: Visualization of the label-hierarchy embedded using OE in $\mathbb{R}^2$. Node colors - cyan: *family*, magenta: *subfamily*, yellow: *genus*. Last level omitted for clarity.

$$\mathscr{L}=\sum_{(u,v)\in\mathcal{P}} E\big(f(u), f(v)\big)+\sum_{(u',v')\in\mathcal{N}} r\big(\alpha-E(f(u'), f(v'))\big) \quad (1)$$

where $r(\cdot)=\max(0,\cdot)$, $\mathcal{P}$ and $\mathcal{N}$ represent positive and negative edges respectively, $\alpha \in \mathbb{R}_+$ is a margin, f is a function that maps a concept to its embedding. $E(f(u), f(v))$ is the energy that defines the severity of the order-violation for a given pair (u, v) and is given by $E(x, y) = ||\max(0, x - y)||$. According to the energy $E(x, y) = 0 \iff y \preceq x$.

For positive pairs where y is-a x, one would like embeddings such that $E(x, y) = 0$. a is-a b implies that a is a sub-concept of b.

Euclidean Cones (EC). Euclidean cones [7] are a generalization of order-embeddings [6]. For each vector x in $\mathbb{R}^N$, the aperture of the cone is based solely on the Euclidean norm of the vector, $||x||$, [7] and is given by $\psi(x) = \arcsin(K/||x||)$ where K is a hyper-parameter. The cones can have a maximum aperture of $\pi/2$ [7]. To ensure continuity and transitivity, the aperture should be a smooth, non-increasing function. To satisfy properties mentioned in [7], the domain of the aperture function has to be restricted to $(\varepsilon, 1]$ for some ε. $\varepsilon = f(K)$. Eq. (2) computes the minimum angle between the axis of the cone at x and the vector y. $E(x, y) = \max(0,\ \Xi(x, y) - \psi(x))$ measures the cone-violation which is the minimum angle required to rotate the axis of the cone at x to bring y into the cone.

$$\Xi(x, y) = \arccos\left(\frac{||y||^2 - ||x||^2 - ||x-y||^2}{2\ ||x||\ ||x-y||}\right) \tag{2}$$

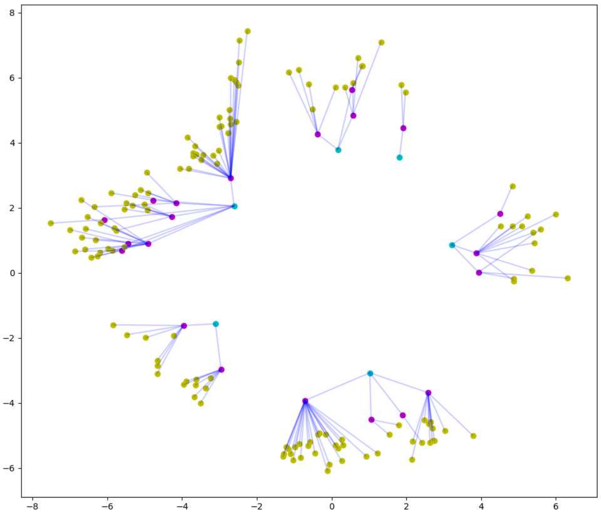


Figure 6: Visualization of the label-hierarchy using Euclidean cones in 2 dimensions. Color coding follows Fig. 5. *genus+species* nodes are omitted to visualize better.

Hyperbolic Cones (HC). The Poincaré ball is defined by the manifold $\mathbb{D}^N = \{x \in \mathbb{R}^N : ||x|| < 1\}$. The distance between two points $x, y \in \mathbb{D}^N$ and the norm are

$d_{\mathbb{D}}(x, y) = \operatorname{arccosh}(1+2(||x-y||^2)/((1-||x||^2)(1-||y||^2)))$ and $||x||_{\mathbb{D}} = d_{\mathbb{D}}(0, x) = 2 \operatorname{arctanh}(||x||)$ where we use $||.||$ for Euclidean norm, $\langle ., . \rangle$ for dot-product and $\hat{x} = x/||x||$ for a unit vector. The angle between two tangent vectors $u, v \in T_x\mathbb{D}^n$ is given by $\cos(\angle(u, v)) = \langle u, v\rangle/(||u||\ ||v||)$. The aperture of the cone is $\psi(x) = \arcsin(K(1 - ||x||^2)/||x||)$. $\Xi(x, y)$ computes the minimum angle between the axis of the cone at x and the vector y.

$$\Xi(x, y) = \arccos\left(\frac{\langle x, y\rangle(1+||x||^2) - ||x||^2(1+||y||^2)}{\omega\sqrt{1+||x||^2||y||^2 - 2\langle x, y\rangle}}\right) \tag{3}$$

$E(x, y) = \max(0,\ \Xi(x, y) - \psi(x))$ measures the cone-violation which is the minimum angle required to rotate the axis of the cone at x to bring y into the cone. $\omega = ||x||\ ||x-y||$

Optimization in hyperbolic space. For parameters living in hyperbolic space, Riemannian stochastic gradient descent (RSGD) [7] is used. An update $u \leftarrow \exp_u(\eta\ \nabla_u^R \mathscr{L})$ involves Rimannian gradient (RG) $\nabla_u^R \mathscr{L}$ for parameter u. RG is computed by rescaling the Euclidean gradient by $\nabla_u^R \mathscr{L} = (1/\lambda_u)^2 \nabla_u \mathscr{L}$ where $\lambda_u = 2/(1 - ||u||^2)$ [7]. *Exponential-map* at a point x, $\exp_x(v) : T_x\mathbb{D}^n \to \mathbb{D}^n$, maps a point v in the tangent space to the hyperbolic space:

$$\exp_x(v) = (x(c\lambda_x + s\langle x, \hat{v}\rangle))/q + (\hat{v}s)/q \tag{4}$$

where $\lambda'_x = (\lambda_x - 1)$ and $q = 1 + \lambda'_x c + \lambda_x s\langle x, \hat{v}\rangle$, $s = \sinh(\lambda_x ||v||)$, $c = \cosh(\lambda_x ||v||)$.

4. Approach

4.1. CNN classifiers

We do not focus on specifically designed CNN components but on different ways to formulate probability distributions to pass hierarchical information.

Hierarchy-agnostic baseline classifier (HAB). As a baseline, we use SOTA residual network for image classification [20]. The baseline is agnostic to any label hierarchy in the dataset. The model performs N_t-way classification (see Fig. 9). $N_t = \sum_{i=1}^{L} N_i$ represents labels across all L levels and N_i are the number of distinct labels on the *i-th* level. It uses the one-versus-rest strategy for each of the N_t labels. We minimize multi-label soft-margin loss,

$$\mathscr{L}(x, y) = \frac{1}{N_t}\sum_{j=1}^{N_t}(a_j + b_j) \tag{5}$$

$x \in \mathbb{R}^{N_t}$, $y \in \{0, 1\}^{N_t}$. $a_j = y_j \log((1 + \exp(-x_j))^{-1})$ and $b_j = (1-y_j) \log(\exp(-x_j)/(1+\exp(-x_j)))$. $\mathcal{F}(\mathcal{I}) = x$, where x are the logits (normalized as a probability distribution) from the last layer of a model $\mathcal{F}$ which takes as input image $\mathcal{I}$. From empirical analysis we found that choosing a single threshold for all labels is better as it is less prone to over-fitting than choosing a per-class decision boundary. Refer to Appendix 7.4.

Per-level classifier (PLC). Instead of a single N_t-way classifier we replace it with L N_i-way classifiers where each of the L classifiers handles all the N_i labels present in level L_i (Fig. 10). We use the multi-label soft-margin loss: $\mathscr{L}(x, \tau) = \sum_{i=1}^{L} \mathscr{L}_i(x_i, \tau_i)$.

$$\mathscr{L}_i(x_i, \tau_i) = -x_i[\tau_i] + \log(\sum_{j=1}^{N_i} \exp(x_i[j])) \tag{6}$$

where, τ_i is the true label for the *i-th* level. $x_i \in \mathbb{R}^{N_i}$, $\tau \in \mathbb{I}_{+}^{L}$. $\mathcal{F}(\mathcal{I}) = x$ where, x are the logits from the last layer of $\mathcal{F}$. x_i is a continuous sub-sequence of the predicted logits x, i.e. $x_i = (x_i[N_{i-1}+1], x_i[N_{i-1}+2], ..., x_i[N_{i-1}+N_i])$.

Marginalization classifier (MC). The notion of L levels is built into the per-level classifier but it is still unaware of the relationship between nodes across levels. Here, a single classifier outputs a probability distribution over the final level in the hierarchy. Instead of having classifiers for the remaining $(L-1)$ levels, we compute the probability distribution over each one of these by summing the probability of the children nodes. Although, the network does not explicitly predict these scores, the models is still penalized for incorrect predictions across the L levels. We minimize $\mathscr{L}(x, \tau) = \sum_{i=1}^{L} \mathscr{L}_i(x_i, \tau_i) = -\sum_{i=1}^{L} \log(p_i[\tau_i])$ where, τ_i is the true label for the *i-th* level. $x_i \in \mathbb{R}^{N_i}$, $\tau \in \mathbb{I}_{+}^{L}$. $\mathcal{F}(\mathcal{I}) = x$ where, x are the logits from the last layer of $\mathcal{F}$.

$$p_i[j] = P(v_i^j|\mathcal{I}) = \sum_{c \in \text{childrenOf}(v_i^j)} P(c|\mathcal{I}) \tag{7}$$

$\forall i \in \{1, 2, ..., (L-1)\}$ where, v_i^j is the *j-th* vertex in the *i-th* level. All but the last level use this to compute the probabilities for their labels. For the final level, we compute the probabilities over the leaf nodes by directly using the logits from the model $\mathcal{F}$, using $p_L[j] = P(v_L^j|\mathcal{I}) = \exp(x_j)/(\sum_{k=1}^{N_L} \exp(x_k))$. Once p_L is determined, p_{L-1} can be calculated in a bottom up fashion as seen in Fig. 11.

Masked Per-level classifier (M-PLC). On the upper levels of the hierarchy one has more data per label and fewer labels to choose from. Naturally, this makes classifying relatively accurate closer to the root of the hierarchy. This model exploits knowledge about the parent-child relationship between nodes in a top down manner.

Here, we have L-classifiers, one for each level. For level l_i, the models belief about upper level is leveraged i.e. it's prediction for level l_{i-1}. Instead of naively predicting the label with the highest score for level l_i (comparing among all possible logits), all nodes except the children of the predicted label for the previous level l_{i-1} are masked (see Fig. 12). The label for l_i is the highest scoring unmasked node. The loss is computed over a subset of the original nodes for any level l_i which is possible due to the availability of the parent-child relationship. This assumes that the parent label is correct. Due to less labels and more data, classification in upper levels is more accurate and since we perform this in a top down fashion, this is a reasonable assumption. Another work has shown this to be the case [21].

While training, even if the model predicts the parent incorrectly, we still use the ground truth to penalize its prediction for the children. For data with unknown ground truth i.e. during evaluation, the model uses the predictions from level l_{i-1} to infer about level l_i by masking nodes that correspond to labels that are not possible as per the hierarchy. We minimize $\mathscr{L}(x, \tau) = \sum_{i=1}^{L} \mathscr{L}_i(x_i, \tau_i)$, where

$$\mathscr{L}(x_i, \tau_i) = -x_i[\tau_i] + \log(\sum_{j \in C} \exp(x_i[j])) \tag{8}$$

τ_i is the true label for the *i-th* level. $x_i \in \mathbb{R}^{N_i}$, $\tau \in \mathbb{I}_{+}^{L}$, $C = \text{childrenOf}(v_{i-1}^{\tau_{i-1}})$. v_i^j is the *j-th* vertex (node) in the *i-th* level and consequently, $v_{i-1}^{\tau_{i-1}}$ is the node corresponding to the ground-truth on level $(i-1)$. $\mathcal{F}(\mathcal{I}) = x$ where, x are the logits from the last layer model $\mathcal{F}$. x_i is a continuous sub-sequence of the predicted logits x, i.e. $x_i = (x_i[N_{i-1}+1], x_i[N_{i-1}+2], ..., x_i[N_{i-1}+N_i])$.

Hierarchical Softmax (HS). HS model predicts logits for every node in the hierarchy. There are dedicated linear layers for each group of sibling nodes leading to a separate (conditional) probability distribution over them. This is probability conditioned on the parent node i.e. $p(v_i^{j_i}|v_{i-1}^{j_{i-1}}), \forall v_i^{j_i} \in C$, such that $C = \text{childrenOf}(v_{i-1}^{j_{i-1}})$.

To reduce computation over large vocabularies, [22, 23] propose similar ideas for NLP. In the context of computer vision it is relatively unexplored and we propose to predict conditional distributions for each set of direct descendants to exploit the label-hierarchy.

$$p(v_i^{j_i}|v_{i-1}^{j_{i-1}}) = \exp(x_{v_{i-1}^{j_{i-1}}}[j_i])/(\sum_{k \in C} \exp(x_{v_{i-1}^{j_{i-1}}}[k])) \tag{9}$$

$\forall v_i^{j_i} \in C$, $x_{v_{i-1}^{j_{i-1}}} \in \mathbb{R}^{|C|}$. The vector $x_{v_{i-1}^{j_{i-1}}}$ represents the logits that exclusively correspond to all the children of node $v_{i-1}^{j_{i-1}}$. With this in place, for each set of children of a given node, a conditional probability distribution is output by $\mathcal{F}$. $\mathcal{F}(\mathcal{I}) = p(\cdot)$ where, $p(\cdot)$ is the conditional probability for every child node given the parent, $p(v_i^{j_i}|v_{i-1}^{j_{i-1}})$. In order to calculate the joint distribution over the leaves, probabilities along the path from the root to each leaf are multiplied as $p(v_1^{j_1}, v_2^{j_2}, ..., v_{(L-1)}^{j_(L-1)}, v_L^{j_L}) = p(v_1^{j_1})p(v_2^{j_2}|v_1^{j_1})...p(v_L^{j_L}|v_{(L-1)}^{j_(L-1)})$ where, $v_i^{j_i}$ is the parent node of $v_{i+1}^{j_{(i+1)}}$. The nodes belonging to the i-*th* level and the (i+1)-*st* level respectively.

The cross-entropy loss is computed only over the leaves but since the distribution is calculated using internal nodes, all levels are optimized implicitly. $\mathscr{L}(x, \tau) = -\log(p(v_1^{j_1}, v_2^{j_2}, ..., v_{(L-1)}^{j_(L-1)}, v_L^{\tau_L})) = -\log(p(v_1^{\tau_1}, v_2^{\tau_2}, ..., v_{(L-1)}^{\tau_{L-1}}, v_L^{\tau_L}))$, where, τ_i is the true label for the *i-th* level. $x_i \in \mathbb{R}^{N_i}$, $\tau \in \mathbb{I}_{+}^{L}$.

4.2. Embedding Classifiers

We treat our label hierarchy as a directed-acyclic graph, more specifically as a directed tree graph. The dataset $\mathcal{X}$ consists of entailment relations (u, v) connected via a directed edge from u to v. (following the definition in [7]). These directed edges or hypernym links convey that v is a sub-concept of u.

4.2.1 Label and Image Representations

Label embeddings. For our implementation of the HC, the label-embeddings live in the hyperbolic space $\mathbb{D}^N$ and are optimized using the RSGD as per Section 3. RSGD is implemented by modifying the SGD gradients in PyTorch[24] as it is not a part of the standard library.

Image embeddings. For images, features from the final layer of the backbone of the best performing CNN-based model are used ($\in \mathbb{R}^{2048}$). In order to map them to $\mathbb{D}^N$ we use a linear transform $W \in \mathbb{R}^{2048 \times N}$ and then apply a projection into $\mathbb{D}^N$ via the exponential-map at zero which is equivalent to $\exp_0(x)$. This bring the image embeddings to the hyperbolic space with Euclidean parameters. This allows for optimizing the parameters with well know optimization schemes such as Adam [25].

4.2.2 Embedding Label-Hierarchy

We begin by learning to represent the taxonomical hierarchy alone. Considering only the label-hierarchy and momentarily excluding the images we model this problem as hypernym prediction where a hypernym pair represents two labels (x, y) such that y `is-a` x. Embeddedings for the label-hierarchy with OE and EC are shown in Fig. 5 and Fig. 6.

Data splitting. We use the tree to form the "basic" edges for which the transitive closure can be fully recovered. If these edges are not present in the *train* set, the information about them is unrecoverable and therefore they are always included in the *train* set. Now, we randomly pick edges from the transitive closure [26] minus the "basic" edges to form a set of "non-basic" edges. We use the "non-basic" edges to create *val* (5%) and *test* (5%) splits and a proportion of the rest are reserved for training.

Training details. We follow the training details in [7]. We augment both the validation and test set by 5 negative pairs each for (x, y): of the type (x', y) and (x, y') with a randomly chosen edge that is not present in the full transitive closure of the graph. Generating 10 negatives for each positive. We report performance on different training set sizes. We vary the training set to include 0%, 10%, 25%, 50% of the "non-basic" edges selected randomly. We train for 500 epochs with a batch size of 10. We run two sets of experiments: one, we fix $\alpha = 1.0$ as mentioned in [6] and two, tune α based on the F1-score on the *val* set [7].

***Pick-per-level* strategy.** During the experiments, instead of sampling a negative edge (x', y) uniformly from candidate x', we pick each x' from a different level in the hierarchy. This serves a dual purpose. 78.24% of the nodes belong to the final level in the hierarchy and uniform negative sampling would result in edges where x' is from the last level majority of the times, making convergence slow. Secondly, this strategy samples hard negatives edges from the same level as the non-corrupted node y, helping embeddings to disentangle and spread out in space.

Optimization details. We use Adam optimizer [25] for order-embeddings and Euclidean cones. For hyperbolic cones we use RSGD [7]. $lr = 0.01$. We also embed synthetic trees of varying height and branching factor using OE and EC. The final embeddings are visualized in Fig. 13.

4.2.3 Jointly Embedding Images with Label-Hierarchy

In order-embeddings [6], the images are put on the lower-level and the captions on the upper level as images are more detailed while captions represent concepts more abstract than the image itself. For jointly embedding the images together with the labels we use the hypernym loss from Eq. (1). We modify it such that now in addition to the labels, $\mathcal{G}$ (the graph representing the hierarchy) also contains images as nodes as leaves at the lowest level. $\mathcal{G}$ constitutes of two types of edges: an edge (u, v) can be such that $u, v \in$ labels or $u \in$ labels, $v \in$ images. The embeddings are computed differnently for images and labels but in the end, both f_i and f_l map respective inputs to the same space.

Multi-label Classification with Embeddings Since our problem does not concern hypernym prediction but rather assigning multiple labels to an image; instead of performing edge prediction (as the case would be in a hypernym prediction task) we use the embeddings for the task of classification. To classify an image we compute the order-violation energy E between the given image and each label and pick the label corresponding to the minimum violation, $\arg\min_{l} E(f_l(l), f_i(i)), \forall l \in$ labels.

Generating Label and Image Embeddings To generate image embeddings we use the best performing CNN model trained on the ETHEC dataset and extract *fc7*-features from the penultimate layer. We use a learnable linear transformation, a matrix W, on top of the *fc7*-features to be able to adjust the *fc7*-features and map them into the joint embedding space: $f_i(i) = W * \text{CNN}(i) \in \mathbb{R}^N$. CNN($i$) represent the *fc7*-features from our best performing CNN model and W is a matrix. The weights of the CNN are frozen to calculate the *fc7*-features with only W that can be learned. For the labels, $f_l(l)$ is just a lookup table that stores vectors in $\mathbb{R}^N$. The embedding are in $\mathbb{R}^N$ for Euclidean models and $\mathbb{D}^N$ for hyperbolic models (Poincaré disk).

Data splitting. We split the data the same way as for the CNN models: *train* (80%), *val* (10%) and *test* (10%) based solely on the images. The graph $\mathcal{G}$ contains directed edges from each label to the image that it "describes" as well as edges between related labels.

Training details. Let $\mathcal{G}$ represent the graph to be embedded. All edges in $\mathcal{G}_{tc}$, the transitive closure of $\mathcal{G}$, are considered as positive edges. To obtain negative edges, $\mathcal{G}_{neg}$ is constructed by removing the edges in $\mathcal{G}_{tc}$ from a fully-connected di-graph with the same nodes as $\mathcal{G}$.

While training, we generate negative pairs as mentioned in Section 4.2.2 with the *pick-per-level* strategy. We make sure that we do not sample a negative edge (u', v') such that both u and v are images. This ensures that no two images are forced apart unless their labels require them to do so. For validation and testing, we measure the model's classification the *val* and *test* set images respectively.

Graph reconstruction task. In addition to the classification task, we also check the quality of reconstruction of the label-hierarchy itself. Here, all the edges in $\mathcal{G}$ that correspond to edges between labels are treated as positive edges, while the the edges in $\mathcal{G}_{neg}$ that correspond to edges between labels are treated as negative edges. We compute $E(u, v)$ $\forall e \in \mathcal{P} \cup \mathcal{N}$ where $e = (u, v)$ and choose a threshold to classify edges as positive and negative using that yields the best F1-score on this label-hierarchy reconstruction task. This task does not use any edges that have an image on any side to check the quality of reconstruction.

For W we use a linear transformation, a matrix $\mathbb{R}^{2048 \times N}$. Non-linearity is not applied to the output that maps to the embedding space.

Optimization details. For jointly embedding labels and images, we empirically found using Adam [25] optimizer instead of the RSGD. The label embeddings are parameterized in the Euclidean space and we use the $\exp_0(v)$ to map them to the hyperbolic space. This is observed to be more stable and helps better converge the joint embeddings. Also, with this implementation of the hyperbolic cones, for both labels and joint embeddings, it was not necessary to initialize the embeddings with the Poincaré embeddings [27] as suggested in [7]. However, a performance boost is obtained when initialized with values from embedding only the label-hierarchy. EC: 200 epochs, $lr_{labels} = 10^{-2}$, $lr_{im} = 10^{-3}$. HC: 100 epochs, lr_{labels}=10^{-4}, lr_{im}=10^{-3}, Initialization from label-embeddings only model. Adam and α=1.

5. Experiments

Data. We empirically evaluate our work on the real-world ETH Entomological Collection (ETHEC) dataset [1] comprising images of *Lepidoptera* specimens with their taxonomy tree. The real-world dataset has variations not only in terms of the images per category but also a significant imbalance in the structure of the taxonomical tree. In Fig. 2 we illustrate the data distribution for each label in the ETHEC hierarchy.

5.1. Hierarchical Classification Performance

To perform image classification using embeddings, the least violating energy $E(f_l(l), f_i(i))$ for a given image across all possible labels in a given level in the hierarchy is considered as the predicted label. The CNN models use Adam [25] for 100 epochs with 224 x 224 RGB images and batch size=64. For HAB, PLC: lr$= 10^{-2}$; MC, M-PLC, HS: lr$=10^{-5}$. We empirically found ResNet-50 for HAB, PLC, MC, M-PLC and ResNet-152 for HS among ResNet 50, 101, 152 variants.

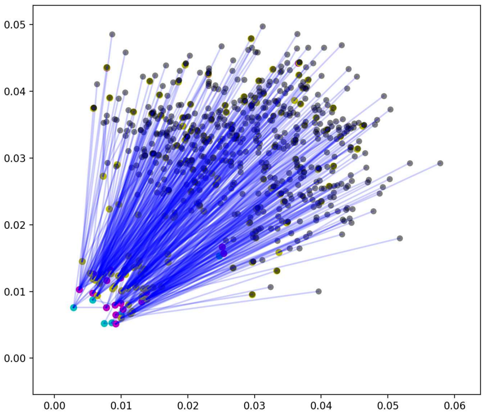


Figure 7: Label-only embeddings with HC $\mathbb{D}^{1000}$ projected to 2D. The embeddings organize themselves such that more generic concepts are closer to the origin while the most specific concepts form the periphery. Color coding as Fig. 5.

	classify *test* set images			graph reconstruction		
Model	**m-F1**	hit@3	hit@5	TPR	TNR	full-F1
			Euclidean Cones			
$d = 10$	0.780	0.889	0.920	0.805	0.998	0.704
$d = 10^2$	0.835	0.902	0.943	**0.963**	**0.999**	**0.821**
$d = 10^3$	0.801	0.897	0.928	0.815	0.998	0.707
			Hyperbolic Cones			
$d = 10^2$	**0.840**	**0.920**	**0.939**	0.642	0.998	0.576
$d = 10^3$	0.805	0.902	0.928	0.523	0.997	0.483

Table 1: The table summarizes the embedding model performance when used to classify images for the ETHEC dataset [1]. The joint image and label embeddings live in $\mathbb{R}^d$ or $\mathbb{D}^d$. m-F1 is the critical metric for image classification performance. We also report the quality of the reconstruction for the label-hierarchy after the joint embedding.

Table 2 shows that the hierarchy-agnostic baseline is outperformed by all models that use any kind of hierarchical information. Embeddings: a completely different class of models, used widely in context of natural language but are relatively unexplored for image classification, also outperform HAB.

W's model capacity. We use a matrix W that transforms *fc7* image features to the embedding space. A more elaborate 4-layer feed-forward neural network was also used but performed worse and was hard to optimize. Jointly training the complete CNN was also over-fitting.

Negative edge frequency. For joint-embedding the ETHEC dataset [1], since the images (around 50,000) out-

Model	m-F1	L_1	L_2	L_3	L_4
CNN-based methods					
HAB	0.8147	0.9417	0.9446	0.8311	0.4578
PLC	0.9084	0.9766	0.9661	0.9204	0.7704
MC	**0.9223**	**0.9887**	**0.9758**	**0.9273**	**0.7972**
M-PLC	0.9173	0.9828	0.9701	0.9233	0.7930
HS	0.9180	0.9879	0.9731	0.9253	0.7855
Order-preserving (joint) embedding models					
EC d=100	0.8350	0.9728	0.9370	0.8336	0.5967
HC d=100$*$	0.7627	0.9695	0.9205	0.7523	0.4246
HC d=100	**0.8404**	**0.9800**	**0.9439**	**0.8477**	**0.5977**

Table 2: Both EC and HC exploit hierarchical information and outperform the hierarchy-agnostic classifier baseline. We include the overall m-F1 in addition to the separate m-F1 across the 4 levels in the ETHEC dataset [1]. All joint-embeddings models are initialized using labels-only embeddings. $*$=random initalization, best overall model, **best model in category**.

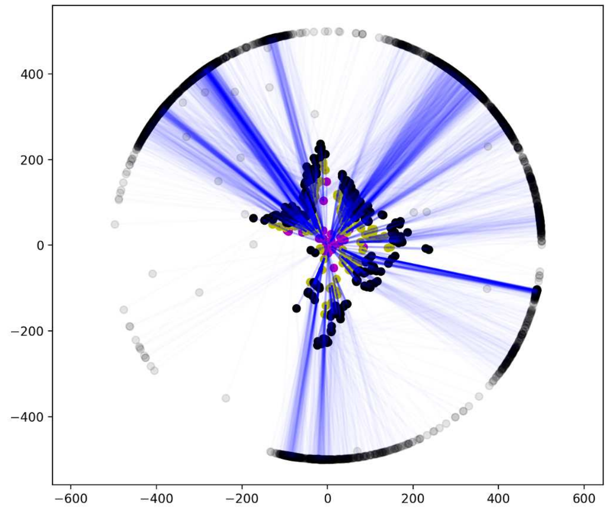


Figure 8: Jointly embedding labels and images using EC in $\mathbb{R}^2$. Color coding follow Fig. 5, grey: images. The images are accumulated around the periphery, away from the origin.

number the labels (723) we thought it might be useful to randomly sample negative edges such that the ratio of negative nodes have a proportion to be 50%:50% for images:label ratio however, the original strategy works better.

Choice of Optimizer. Initial experiments for the hyperbolic cones (HC) used the RSGD optimizer as it seemed to work for labels-only embeddings hyperbolic cones. When using the same to optimize over the labels for the joint-embedding model, we noticed that the label hierarchy moves towards the image labels and ends up collapsing from a very good initialization (taken from the labels-only embeddings). The collapse leads to entanglement between nodes from different labels and images, which leads it to a point of no return and the performance worsens due to the label-hierarchy becoming disarranged and its inability to recover. We believe that the reason for its inability to re-arrange is due to there being a two different types of objects being embedded (and also being computed differently) and it compounded by using different optimizers.

In our experiments we obtain best results when using the Adam optimizer even if it means the update step for parameters living in hyperbolic space has to be performed in an approximate manner. Adam optimizer with an approximate update step works better in practice than RSGD with its mathematically more precise update step.

Label initialization for joint-embeddings Using RSGD we observed that if the labels are not initialized with the labels-only embedding then the joint model finds it difficult to disentangle the label embeddings and eventually this effect is cascaded to the images causing the image classification performance to not improve.

With the RSGD replaced by the Adam optimizer, in experiments where we randomly initialized the label-embeddings, we observed them to disentangle and form entailment cones even with the images being involved and making the optimization more complex. The joint-model still works well with random label initialization and achieves an image classification m-F1 score of 0.7611 and even outperforms the hierarchy-agnostic CNN in the m-F1 L_1. [7] recommends to use Poincaré embeddings [27] to initialize the hyperbolic cones model. The fact that the joint model as well as the labels-only hyperbolic cones have great performance without any special initialization scheme is interesting. We conjecture that this could be because of using an approximate yet better optimizer.

6. Conclusion

We propose an embedding-based approach for image classification using *entailment cones*, a recently proposed type of *order-preserving* embeddings. In particular, we compare these both in the Euclidean geometry setting and in the hyperbolic setting, and show that hyperbolic geometry provides an empirical advantage over Euclidean geometry. We also propose and compare a set of simple hierarchical classifier baselines where the hierarchy is incorporated in the loss function. Although these tend to perform slightly better than embedding-based approaches, they are less flexible as they assume that the hierarchy is fixed, and are more limited in terms of downstream tasks (e.g. they do not allow for hierarchy-based retrieval). Finally, we evaluate our methods on the real-world ETHEC dataset [1], and show that exploiting hierarchical information always leads to an improvement over a shallow CNN classifier.

References

[1] A. Dhall, "Eth entomological collection (ethec) dataset [palearctic macrolepidoptera, spring 2019] https://www.research-collection.ethz.ch/handle/20.500.11850/365379," 2019.

[2] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[3] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[4] J. Deng, J. Krause, A. C. Berg, and L. Fei-Fei, "Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3450–3457, IEEE, 2012.

[5] C. Samplawski, J. Wolff, T. Klein, and M. Nabi, "Learning graph-based priors for generalized zero-shot learning," *Workshop on Deep Learning on Graphs: Methodologies and Applications*.

[6] I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun, "Order-embeddings of images and language," *arXiv preprint arXiv:1511.06361*, 2015.

[7] O.-E. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic entailment cones for learning hierarchical embeddings," *arXiv preprint arXiv:1804.01882*, 2018.

[8] R. Suzuki, R. Takahama, and S. Onoda, "Hyperbolic disk embeddings for directed acyclic graphs," *arXiv preprint arXiv:1902.04335*, 2019.

[9] M. Le, S. Roller, L. Papaxanthos, D. Kiela, and M. Nickel, "Inferring concept hierarchies from text corpora via hyperbolic embeddings," *arXiv preprint arXiv:1902.00913*, 2019.

[10] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *Advances in neural information processing systems*, pp. 5345–5355, 2018.

[11] X. Li, L. Vilnis, D. Zhang, M. Boratko, and A. McCallum, "Smoothing the geometry of probabilistic box embeddings," in *International Conference on Learning Representations*, 2019.

[12] B. Muzellec and M. Cuturi, "Generalizing point embeddings using the wasserstein space of elliptical distributions," 2018.

[13] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler, "Vse++: Improving visual-semantic embeddings with hard negatives," *arXiv preprint arXiv:1707.05612*, 2017.

[14] B. Barz and J. Denzler, "Hierarchy-based image embeddings for semantic image retrieval," *arXiv preprint arXiv:1809.bib09924*, 2018.

[15] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, *et al.*, "Devise: A deep visual-semantic embedding model," in *Advances in neural information processing systems*, pp. 2121–2129, 2013.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[17] S. Kumar and R. Zheng, "Hierarchical category detector for clothing recognition from visual data," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2306–2312, 2017.

[18] T. Chen, W. Wu, Y. Gao, L. Dong, X. Luo, and L. Lin, "Fine-grained representation learning and recognition by exploiting hierarchical semantic embedding," *arXiv preprint arXiv:1808.04505*, 2018.

[19] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam, "Large-scale object classification using label relation graphs," in *European conference on computer vision*, pp. 48–64, Springer, 2014.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[21] T. Kjosev, "Deep learning for generating template pictorial and textual representations," *Thesis*, 2018.

[22] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model.," in *Aistats*, vol. 5, pp. 246–252, Citeseer, 2005.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *Software Library*, 2017.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] Wikipedia contributors, "Transitive closure — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Transitive_closure&oldid=926668384`, 2019. [Online; accessed 16-February-2020].

[27] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in neural information processing systems*, pp. 6338–6347, 2017.

[28] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Resampling or reweighting: A comparison of boosting implementations," in *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, vol. 1, pp. 445–451, IEEE, 2008.