Hierarchical classification at multiple operating points

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many classification problems consider classes that form a hierarchy. Classifiers that are aware of this hierarchy may be able to make confident predictions at a coarse level despite being uncertain at the fine-grained level. While it is generally possible to vary the granularity of predictions using a threshold at inference time, most contemporary work considers only leaf-node prediction, and almost no prior work has compared methods at multiple operating points. We present an efficient algorithm to produce operating characteristic curves for any method that assigns a score to every class in the hierarchy. Applying this technique to evaluate existing methods reveals that top-down classifiers are dominated by a naïve flat softmax classifier across the entire operating range. We further propose two novel loss functions and show that a soft variant of the structured hinge loss is able to significantly outperform the flat baseline. Finally, we investigate the poor accuracy of top-down classifiers and demonstrate that they perform relatively well on out-of-distribution examples.

1 Introduction

Many classification problems involve classes that can be recursively grouped into a hierarchy of progressively larger superclasses. This hierarchy can be represented by a directed graph where the nodes are classes and the edges define a superset-of relation. Knowledge of the hierarchy can be useful in many different respects. For example, mistake severity can be quantified using distance in the graph [12, 3], classifiers can make predictions at a coarse level to avoid an error at the fine-grained level [14, 51], classes with few labels in a long-tailed distribution can benefit from the examples of similar classes [51], and a cascade can be used to reduce inference time [35, 21, 18].

Whereas many recent works consider only leaf-node prediction, we are interested in the setting where the classifier may predict any class in the hierarchy, including internal nodes. In this setting, there is an inherent trade-off between specificity and correctness: more general predictions contain less information but have a better chance of being correct. It is important to measure both aspects during evaluation, and we refer to the particular balance that a classifier achieves as an operating point. Rather than evaluate the model at a single operating point, we will show that it is possible to construct an operating characteristic *curve*, analogous to ROC or precision-recall curves, enabling a fairer and more complete comparison of different methods.

Given the effectiveness of deep learning, we focus on loss functions for training differentiable models. Specifically, we will compare the trade-offs realized by different loss functions using the proposed technique. Experiments on the iNat21 [47] and ImageNet-1k [13] datasets for image classification reveal that a naïve flat softmax classifier *dominates* recently proposed top-down classifiers, obtaining better accuracy at all operating points. We introduce two novel loss functions and show that a soft

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

version of the classical structured prediction loss can realize a better trade-off than the flat softmax baseline at all operating points.

While it was already known that top-down approaches provide worse *leaf-node* predictions than a flat softmax [41, 3], we did not expect this to hold for non-leaf predictions. We hypothesize that the top-down approaches obtain worse accuracy than the "bottom-up" flat softmax because the coarse classes can be highly diverse, and thus better learned by a union of distinct classifiers when the fine-grained labels are available. To support this hypothesis, we show that training a flat softmax classifier at a lower level and testing at a higher level provides better accuracy than training at the higher level. Finally, we consider a synthetic *out-of-distribution* problem where a hierarchical classifier that explicitly assigns scores to higher-level classes may be expected to have an advantage. The top-down methods are more competitive for unseen classes, while a recent method based on "parameter inheritance" [51] is the most effective.

2 Related work

There are several different types of hierarchical classification problem. In the terminology of Silla and Freitas [44], we consider problems with tree structure, Single-Path Labels (SPL) and Non-Mandatory Leaf-Node Prediction (NMLNP). *Tree structure* means that there is a unique root node and every other node has exactly one parent, *Single-Path Labels* means that a sample cannot belong to two separate classes (unless one is a superclass of the other) and *Non-Mandatory Leaf-Node Prediction* means that the classifier may predict any class in the hierarchy, not just leaf nodes. While a significant body of work has considered the problem of *learning* the hierarchy, we consider the distinct problem of instilling a classifier with knowledge of an established hierarchy.

MLNP with deep learning. Several recent works have sought to leverage a class hierarchy in the aim of improving leaf-node prediction. Wu et al. [50] trained a softmax classifier by optimizing a "win" metric that comprised a weighted combination of likelihoods on the path from the root to the label. Bertinetto et al. [3] proposed a similar Hierarchical Cross-Entropy (HXE) loss that instead considered the conditional likelihood of each node given its parent, as well as a Soft Label loss that generalized label smoothing [36] to incorporate the hierarchy. Karthik et al. [26] demonstrated that a flat softmax classifier remained highly effective, and applied Conditional Risk Minimization (CRM) at inference time. Chang et al. [8] partitioned the final feature vector into separate parts for each level of the hierarchy and blocked gradients between levels. Numerous works have proposed *architectures* that reflect the hierarchy [53, 37, 1, 20, 55, 27, 30]. Guo et al. [23] introduced a hybrid CNN-RNN in which the RNN traverses down from the root node. We are primarily interested in comparing different loss functions while keeping the architecture constant.

Non-MLNP methods. Comparatively few works have entertained hierarchical classifiers that can predict aribtrary nodes. Deng et al. [14] highlighted the trade-off between specificity and accuracy, and sought to obtain the most-specific classifier that would satisfy a maximum error rate. Davis et al. [10, 11] re-calibrated a flat softmax classifier within a hierarchy using a held-out set and then performed inference by starting at the most-likely leaf-node and climbing the tree until a threshold was met, considering fixed thresholds of 90 and 95%. Wu et al. [51] proposed the Deep Realistic Taxonomic Classifier (Deep RTC), which obtains logits for each node by summing over its ancestors (following [43]) and whose loss is evaluated at random cuts of the tree. They perform inference by traversing down from the root until a logit threshold is crossed. In the YOLO-9000 detector, Redmon and Farhadi [41] introduced a conditional softmax classifier that bore a resemblance to the efficiency-motivated approach of Morin and Bengio [35]. Their model outputs a concatenation of logit vectors that each parameterize (via softmax) the conditional distribution of children given a parent. It was noted to provide elegant degradation but the hierarchical predictions were not rigorously evaluated. Brust and Denzler [6] generalized the conditional approach to multi-path labels in a DAG hierarchy by replacing the softmax with a sigmoid for each class given each of its parents. Inference was performed by seeking the class with the maximum likelihood excluding its children, which may predict non-leaf nodes. This was later extended to handle partial-depth annotation [5].

Hierarchical metrics. There are many ways to measure the accuracy of hierarchical classifiers [9, 28]. For MLNP, the hierarchy is typically used to quantify error severity. Russakovsky et al. [42] found that, for the ImageNet challenge, the average edge distance of the top k predictions produced the same ordering as top-1 and top-5 accuracy. Bertinetto et al. [3] measured the average mistake severity

of the misclassified examples, although Karthik et al. [26] cautioned that this rewards methods for additional low-severity errors. Wu et al. [50] evaluated their "win" metric using a one-hot distribution to measure the quality of leaf-node predictions. When considering NMLNP, it is important to capture both correctness and specificity. This is achieved by the Wu-Palmer metrics [52], which measure precision and recall of the set of predicted classes. Deng et al. [14] measured specificity using information, and Zhao et al. [54] modified the Wu-Palmer metrics to use this measure. Davis et al. [10] measured specificity using the fraction of examples that receive non-root classifications. Wu et al. [51] proposed a combined metric for specificity and correctness that measures the fraction of true-negative leaf nodes if the prediction is correct and zero otherwise. Class hierarchies have also become commonplace in object detection, with LVIS [24] and OpenImages [31] including superclasses during evaluation. Hierarchical metrics have also been to evaluate methods that predict embeddings in a continuous label space for zero-shot learning [49, 17, 2, 54].

Structured prediction. Cai and Hofmann [7] applied a linear SSVM (Structured Support Vector Machine [46]) to the problem of hierarchical document classification. Edunov et al. [16] considered both the classical structured hinge loss and a soft version [39, 19] for sequence-to-sequence learning. Menon et al. [34] used the soft structured loss for long-tail classification, encouraging infrequent classes to be separated by a greater margin. In earlier work, Verma et al. [48] applied the structured hinge loss to metric learning with a class hierarchy, and Sun et al. [45] applied it to train a linear model for efficient inference.

3 Problem definition

3.1 Class hierarchy

Let $\mathcal{Y}$ be the set of classes, including both leaf nodes and their superclasses. The hierarchy is defined by a tree with edges $\mathcal{E} \subset \mathcal{Y}^2$. The edges define a transitive, non-strict superset relation $\supseteq$ over the classes. It is helpful to think of the classes as sets of samples. A sample x that belongs to class y also belongs to the ancestors (superclasses) of y; that is, $x \in y$ and $y \subseteq y'$ implies $x \in y'$. Hierarchical classification can be seen as multi-label classification, since samples belong to multiple classes simultaneously. However, samples cannot belong to *arbitrary* subsets of $\mathcal{Y}$, since belonging to one class implies belonging to its superclasses. Furthermore, we consider only class hierarchies in which siblings are *mutually exclusive*; that is, a sample x can only belong to two classes y and y' if one is a superclass of the other. The problem is therefore to assign a single label $y \in \mathcal{Y}$ to an example, with this label signifying membership in class y and all of its superclasses. While any superclass of the ground-truth label is deemed to be a correct classification, it is preferable for the classifier to predict the most-specific correct label.

We briefly introduce our notation for the tree. Let $\mathbf{r} \in \mathcal{Y}$ be the unique root node, let $\pi(y) \in \mathcal{Y}$ be the unique parent for every node $y \in \mathcal{Y} \setminus \{\mathbf{r}\}$, let $\mathcal{C}(y) \subseteq \mathcal{Y}$ be the children of node y, let $\mathcal{L} \subseteq \mathcal{Y}$ be the set of leaf nodes, let $\mathcal{A}(y) \subseteq \mathcal{Y}$ be the ancestors of node y, and let $\mathcal{D}(y) \subseteq \mathcal{Y}$ be the descendants of node y. We define the ancestors and descendants inclusively, such that $\mathcal{D}(y) \cap \mathcal{A}(y) = \{y\}$. It will also be useful to introduce $\mathcal{L}(y) = \mathcal{L} \cap \mathcal{D}(y)$ to denote the set of leaf descendants of node y and $\mathcal{S}(y) = \mathcal{C}(\pi(y))$ to denote its siblings. The superset relation over classes is equivalent to the ancestor relation in the tree; that is, $u \supseteq v$ iff $u \in \mathcal{A}(v)$.

We focus on models that map a sample x to a conditional likelihood $p(y|x)$ over all labels in the hierarchy. If we consider $\mathcal{Y}$ with its superset relation and mutually exclusivity of siblings as an event space, then a function $p : \mathcal{Y} \to [0, 1]$ must satisfy

$$\forall u \in \mathcal{Y} : \quad p(u) \geq \textstyle\sum_{v \in \mathcal{C}(u)} p(v) \tag{1}$$

to be a probability function on $\mathcal{Y}$. If this holds with equality, we say that the children are exhaustive. We generally expect that the root node is uninformative $p(\mathbf{r}) = 1$.

3.2 Metrics

Metrics $M(y, \hat{y})$ measure the quality of predicted label $\hat{y}$ for ground-truth label y. When making non-leaf predictions, it is important to capture both correctness and specificity, either using a pair of metrics or a single combined metric. The simplest such pair are the binary metrics

$$\text{Correct}(y, \hat{y}) = [\hat{y} \supseteq y] \ , \qquad \text{Exact}(y, \hat{y}) = [\hat{y} = y] \ . \tag{2}$$

If the ground-truth label is not a leaf node, then it is possible for the predicted label to be below it, $\hat{y} \subset y$. This is not considered an error, and we replace $\hat{y} \leftarrow y$ where this occurs.

Since the Exact metric is often too strict, it is useful to introduce a non-binary measure of label specificity. Two popular choices are the depth of the node $d(y)$ and its information content $I(y) = -\log p(y) = \log|\mathcal{L}| - \log|\mathcal{L}(y)|$ (assuming a uniform distribution over the leaf nodes [14, 54]). We prefer information as it better accounts for an imbalanced tree; in a perfect k-ary tree, $I(y) \propto d(y)$. Rather than directly measure the specificity of the predictions [14], the specificity measure can be used to define precision and recall metrics [54] using the Lowest Common Ancestor (LCA):

$$\mathrm{Recall}(y, \hat{y}) = \frac{I(\mathrm{lca}(y, \hat{y}))}{I(y)} \ , \qquad \mathrm{Precision}(y, \hat{y}) = \frac{I(\mathrm{lca}(y, \hat{y}))}{I(\hat{y})} \ . \tag{3}$$

Correct and Precision are metrics for correctness (decreasing with depth), whereas Exact and Recall are metrics for specificity (increasing with depth). Each type should not be used without the other.

4 Inference and loss functions

We adopt the paradigm that a differentiable model f with parameters ϕ maps an input x to a real vector $\theta = f(x, \phi)$ that parameterizes the conditional likelihood $p(\cdot|x) = q(\theta) \in [0, 1]^{\mathcal{Y}}$. Learning will be performed using Stochastic Gradient Descent (SGD) to minimize the expected value of a loss function $\ell(y, \theta)$ on a training set. Inference will be performed using a function $\xi : [0, 1]^{\mathcal{Y}} \to \mathcal{Y}$ to obtain $\hat{y} = \xi(q(\theta))$. We consider the training method to be defined by the pair (ℓ, q) and the inference method to be defined by ξ. The dimension of θ depends on the training method.

4.1 Inference functions

In standard flat classification, inference simply selects the most likely class $\xi(p) = \arg\max_{y \in \mathcal{Y}} p(y)$. However, in the hierarchical setting, this would always select the uninformative root node. We can consider **leaf** inference $\xi(p) = \arg\max_{y \in \mathcal{L}} p(y)$, but this will never select an internal node. We propose **confidence threshold** inference, taking the maximum-information node

$$\xi_\tau(p) = \arg\max_{y \in \mathcal{Y}} I(y) \quad \text{subject to} \quad p(y) > \tau \ . \tag{4}$$

When $\tau \geq 0.5$ and p satisfies (1), there is a single path in the tree whose nodes satisfy the threshold, *i.e.* $p(y) > \tau \iff y \in \mathcal{A}(\hat{y})$. Inference can therefore be performed by traversing down from the root and the maximizer is unique assuming that $I(y)$ is strictly increasing on the edges of the tree. While these properties are nice, we also entertain arbitrary p and $\tau < 0.5$ in the remainder of the paper. We refer to the special case of $\tau = 0.5$ as **majority** inference. Rather than consider a single threshold τ, we propose to study the *operating characteristic curve* in the following section.

Two straightforward variants are **plurality** inference, which instead seeks the maximum-information label which is more likely than any other non-ancestor, and **information threshold** inference, which reverses the roles of the terms to instead maximize $p(y)$ subject to $I(y) \geq \zeta$. While the latter can also generate an operating curve, it does not provide adaptive specificity according to confidence.

Deng et al. [14] proposed an inference rule that maximizes the augmented expected information

$$\xi_\lambda(p) = \arg\max_{y \in \mathcal{Y}} \ (I(y) + \lambda)p(y) \tag{5}$$

with the factor $\lambda \geq 0$ chosen by bisection to yield the desired accuracy on a held-out set. Each value of λ represents a particular balance between expected information $I(y)p(y)$ and confidence $p(y)$. As $\lambda \to \infty$, the maximizer eventually becomes the root node, which maximizes $p(y)$. Although $p(y)$ and $I(y)$ are monotonic with respect to depth (decreasing and increasing, respectively), there is no such guarantee for the product $I(y)p(y)$, and therefore an increase in λ is not guaranteed to move the prediction towards the root. For $\lambda = 0$, we call this **expected information** inference.

Karthik et al. [26] and Deng et al. [12] proposed **conditional risk minimization (CRM)** for the MLNP setting. This selects the leaf-node label $\hat{y}$ to minimize the expected cost $\mathbb{E}[C(Y, \hat{y})]$ with $Y \sim p(\cdot|x)$ being a random leaf node whose distribution is the predicted conditional likelihood

$$\xi(p(\cdot|x)) = \arg\min_{y \in \mathcal{L}} \sum_{u \in \mathcal{L}} C(u, y)\, p(u|x) \ . \tag{6}$$

We apply CRM to predict non-leaf nodes by choosing $C(y, \hat{y}) = \mathrm{Correct}(y, \hat{y}) \cdot I(y)$ and maximizing over *all* nodes with the expectation computed using the likelihood of y and *not* its children

$$\xi(p(\cdot|x)) = \arg\min_{y \in \mathcal{Y}} \sum_{u \in \mathcal{Y}} C(u, y) \left\{ p(u|x) - \sum_{v \in \mathcal{C}(u)} p(v|x) \right\} \ . \tag{7}$$

4.2 Loss functions and likelihood parameterizations

We first present existing loss functions before introducing our novel loss functions. The most straightforward approach is a **flat softmax** classifier over the leaf nodes with a parameter vector θ in $\mathbb{R}^{\mathcal{L}}$. The likelihood of internal nodes is obtained as a recursive sum

$$q_y(\theta) = \frac{\exp \theta_y}{\sum_{u \in \mathcal{L}} \exp \theta_y}\,, \quad y \in \mathcal{L} \qquad\qquad q_y(\theta) = \textstyle\sum_{v \in \mathcal{C}(y)} q_v(\theta)\,, \quad y \notin \mathcal{L}\,. \tag{8}$$

For ground-truth labels $y \in \mathcal{L}$, the NLL is the familiar cross-entropy loss over the set $\mathcal{L}$,

$$\ell(y, \theta) = -\log q_y(\theta) = \mathrm{CE}_{\mathcal{L}}(y, \theta) = -\theta_y + \log \textstyle\sum_{u \in \mathcal{L}} \exp \theta_u\,. \tag{9}$$

However, for $y \notin \mathcal{L}$, the NLL is non-convex function and resembles Multiple Instance Learning [29]

$$\ell(y, \theta) = -\log q_y(\theta) = -\log \textstyle\sum_{u \in \mathcal{L}(y)} \exp \theta_u + \log \sum_{u \in \mathcal{L}} \exp \theta_u\,. \tag{10}$$

Bertinetto et al. [3] proposed an alternative loss function for the same parameterization q. Their **Hierarchical Cross Entropy (HXE)** loss places geometrically decreasing weight on the conditional distribution of deeper nodes given their parents. To define the loss, let $\gamma = e^{-\alpha} \leq 1$ be a discount factor and let $\omega_0, \ldots, \omega_{d(y)} \in \mathcal{Y}$ denote the ordered ancestors of y in

$$\ell_\alpha(y, \theta) = \sum_{k=1}^{d(y)} -\gamma^{k-1} \log \hat{p}_\theta(\omega_k | \omega_{k-1}) = \sum_{k=1}^{d(y)} \gamma^{k-1} \left[-\log \sum_{u \in \mathcal{L}(\omega_k)} \exp \theta_u + \log \sum_{u \in \mathcal{L}(\omega_{k-1})} \exp \theta_u \right]. \tag{11}$$

Besides the flat softmax, another naïve baseline which is unaware of the hierarchy is a **multi-label sigmoid** classifier; that is, independent binary logistic regression per node. The likelihoods are parameterized by a vector θ in $\mathbb{R}^{\mathcal{Y}}$ according to $q_y(\theta) = \sigma(\theta_y) = 1/(1 + \exp(-\theta_y))$, which is not guaranteed to satisfy (1). We adopt the Focal Loss [32] because each binary problem is imbalanced

$$\ell_{\alpha,\gamma}(y, p) = \sum_{u \in \mathcal{Y}} \mathrm{FL}_{\alpha,\gamma}([u \subseteq y], \sigma(\theta_u))\,, \quad \mathrm{FL}_{\alpha,\gamma}(y, p) = \begin{cases} -\alpha(1-p)^\gamma \log p\,, & y = 1 \\ -(1-\alpha) p^\gamma \log(1-p)\,, & y = 0\,. \end{cases}$$

Redmon and Farhadi [41] introduced the **conditional softmax** classifier, which uses a separate softmax for the conditional distribution over each internal node's children. The parameter vector θ is in $\mathbb{R}^{\mathcal{Y} \setminus \{\mathbf{r}\}}$ and the likelihood of each node is obtained as a top-down product, with $p(\mathbf{r}|x) = 1$

$$p(v|u, x) = q_{v|u}(\theta) = \frac{\exp \theta_v}{\sum_{v' \in \mathcal{C}(u)} \exp \theta_{v'}}\,, \qquad p(y|x) = q_y(\theta) = \prod_{u \in \mathcal{A}(y) \setminus \{\mathbf{r}\}} q_{u|\pi(u)}(\theta)\,. \tag{12}$$

The NLL of this parameterization is a sum of softmax cross-entropies, convex for any label $y \in \mathcal{Y}$:

$$\ell(y, \theta) = -\log q_y(\theta) = \textstyle\sum_{u \in \mathcal{A}(y) \setminus \mathbf{r}} \mathrm{CE}_{\mathcal{S}(u)}(u, \theta) \tag{13}$$

Brust and Denzler [6] similarly proposed a **conditional sigmoid** classifier for the more general problem of multi-path labels in a DAG hierarchy. We adapt this for a tree by replacing $q_{v|u}(\theta) = \sigma(\theta_v)$ above. This ensures that parents are more likely than each child, but does not guarantee the condition in (1). Their proposed loss function is Binary Cross Entropy for children of ancestors of the label:

$$\ell(y, \theta) = \sum_{u \in \mathcal{A}(y)} \sum_{v \in \mathcal{C}(u)} \mathrm{BCE}([v \supseteq y], \sigma(\theta_v))\,, \quad \mathrm{BCE}(y, p) = \begin{cases} -\log p\,, & y = 1 \\ -\log(1-p)\,, & y = 0\,. \end{cases} \tag{14}$$

Wu et al. [51] proposed the **Deep Realistic Taxonomic Classifier (Deep RTC)** based on the idea of parameter inheritance [43, 45]. A score for each class is obtained as the sum of individual scores of its ancestors $\beta_y = \sum_{u \in \mathcal{A}(y)} \theta_u$, and inference involves traversing down from the root until encountering a node with score less than zero. The model is optimized using softmax cross-entropy. To ensure that internal nodes get a positive score even when all annotations are leaf nodes, the loss is computed at a random cut obtained by terminating at each node with a fixed probability. While the original definition did not provide a normalized conditional distribution $p(y|x)$, we propose a **sigmoid variant** that uses the total score of each node $q_y(\theta) = \sigma(\beta_y)$ and is more faithful to the original, as well as a **softmax variant** that uses the total scores of only the leaf nodes to construct a proper distribution.

We now introduce our new loss functions for hierarchical classification. The first is the **soft-max-descendant** classifier, which is similar to the flat softmax except that it has scores $\theta \in \mathbb{R}^{\mathcal{Y} \setminus \{\mathbf{r}\}}$ for

internal nodes as well as leaf nodes. The motivation is to permit the model to recognise internal nodes in and of themselves, with the evidence for a class varying *like* the maximum over its descendants

$$q_y(\theta) = \frac{\sum_{u \in \mathcal{D}(y)} \exp \theta_u}{\sum_{v \in \mathcal{L}} \exp \theta_v} , \qquad \log q_y(\theta) \propto \log \sum_{u \in \mathcal{D}(y)} \exp \theta_u \ \sim \ \max_{u \in \mathcal{D}(y)} \theta_u \ . \tag{15}$$

If we simply minimize the NLL, the loss would be non-convex and the scores of internal nodes may have little incentive to be increased. We therefore propose the loss function

$$\ell(y, \theta) = \sum_{u \in \mathcal{A}(y)} \frac{1}{|\mathcal{L}(u)|} \mathrm{CE}_{\{u\} \cup \mathcal{N}(u)}(u, \theta) \tag{16}$$

where $\mathcal{N}(y) = \mathcal{Y} \setminus (\mathcal{A}(y) \cup \mathcal{D}(y))$ is the set of negative labels for y; that is, labels which are neither ancestors nor descendants. Normalization is necessary to prevent higher-level nodes from dominating. Finally, inspired by recent work [34], we propose a **soft-max-margin** loss for the same parameterization. This is a soft version of the structured hinge loss [46], defined in [39, 19, 16] as:

$$\ell(y, \theta) = \mathrm{CE}_{\mathcal{Y}}(y, \theta + \alpha C(y, \cdot)) = \log \left(1 + \sum_{y' \in \mathcal{Y} \setminus \{y\}} \exp\{\theta_{y'} - \theta_y + \alpha C(y, y')\} \right) \tag{17}$$

with $C(y, y')$ being the desired margin of y from y'. We use $C(y, y') = \mathrm{Correct}(y, y')$ and $\alpha = 5$ in all experiments. While the original structured hinge loss is also convex and has been used for hierarchical classification [48, 45], we were unable to use it to train a deep model to high accuracy.

5 Operating characteristic curve

Given a classifier that predicts a conditional distribution over the class hierarchy, the set of predictions that can be attained by varying the parameter τ in confidence-threshold inference is the Pareto set

$$\mathcal{H}(p(\cdot|x)) = \{y \in \mathcal{Y} : \nexists u (p(u|x) > p(y|x) \wedge I(u) > I(y))\} \ . \tag{18}$$

Given an example (x, y), let the sequence $\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{K-1}$ denote the elements of $\mathcal{H}(p(\cdot|x))$ ordered such that $p(\hat{y}_k|x) \geq p(\hat{y}_{k+1}|x)$ and $I(\hat{y}_k) \leq I(\hat{y}_{k+1})$, typically starting at $\hat{y}_0 = \mathbf{r}$. Further, let us define the corresponding sequences $c_k = p(\hat{y}_k|x)$ and $z_k = M(y, \hat{y}_k)$ for some metric M, such that z_k is the value of the metric for $c_k > \tau > c_{k+1}$. Finally, let us define $\delta_k = z_k - z_{k-1}$ for $k \geq 1$, with δ_k representing the change in the metric when decreasing τ from $c_k + \epsilon$ to $c_k - \epsilon$.

This suggests an efficient way to obtain the cumulative metrics for a set of examples $\{(x^i, y^i)\}_i$ over all thresholds τ. It suffices to merge the sequences of pairs $\{(c_k^i, \delta_k^i)\}_{k \geq 1}$ of all examples i to obtain a single sequence $(C_k, \Delta_k)_k$ with $C_k \geq C_{k+1}$. The total metric for $C_k > \tau > C_{k+1}$ is given by

$$Z_k = Z_0 + \sum_{j=1}^{k} \Delta_j \ , \quad k \geq 1 \qquad\qquad Z_0 = \sum_i z_0^i \ , \tag{19}$$

which can be obtained for all k using a simple cumulative sum. After obtaining this curve for the precision and recall metrics, we can evaluate methods using the area under the curve or a fixed intercept. The following section will apply this technique to investigate different loss functions.

The ordered Pareto set for a single example can be obtained in $O(|\mathcal{Y}| \log |\mathcal{Y}|)$ time. For arbitrary p and I, the worst-case running time for a dataset of size N is $O(N|\mathcal{Y}|(\log N + \log |\mathcal{Y}|))$, although each Pareto set can be computed independently in parallel. If we assume that the tree is relatively balanced, $p(\cdot|x)$ satisfies (1) and restrict ourselves to $\tau \in [0.5, 1]$, then the size of the set is bounded by the maximum depth of the tree, which is $O(\log |\mathcal{Y}|)$. This results in a running time of $O(N|\mathcal{Y}| \log |\mathcal{Y}|)$ to find the Pareto sets and $O(N \log N \log |\mathcal{Y}|)$ to merge the sequences. Otherwise, ensuring that all leaf nodes have equal information guarantees that the Pareto set contains at most one leaf node, reducing the worst-case merge time by a factor of $1 - |\mathcal{L}|/|\mathcal{Y}|$.

6 Empirical study

We now apply this technique to compare different loss functions. Most experiments consider image classification on the iNat21-Mini dataset [47], containing 50 examples each for 10,000 biological species in a seven-level taxonomy. All examples have leaf-node (full-depth) annotation. For all iNat21 experiments, we use a ResNet-18 model [25] with input images of size 224×224. We start from the Pytorch ImageNet-pretrained checkpoint [38] and train for 20 epochs using SGD with

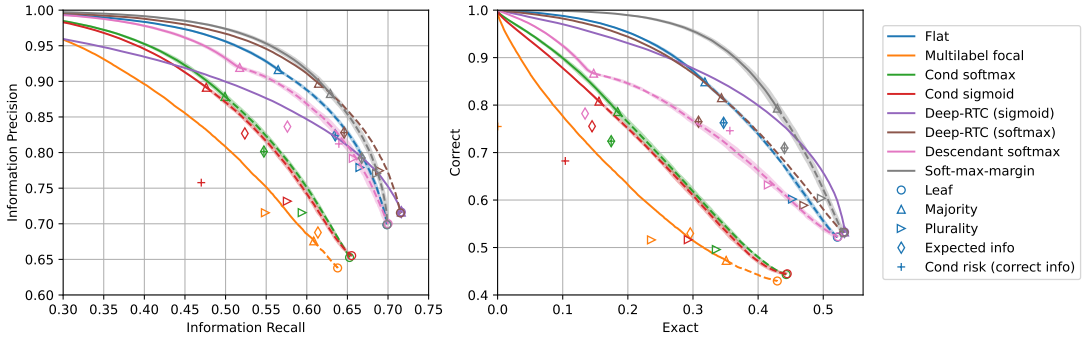


Figure 1: Correctness-specificity trade-offs for different methods and metrics on the iNat21 validation set. Markers indicates operating points obtained by inference functions. Comparing methods using single operating points would not provide a complete perspective. The dashed line is the part of the curve with threshold below 0.5. The shaded areas depict two standard deviations to either side.

Table 1: Overall metrics derived from operating curves on the iNat21 validation set. Average Precision (AP) and Average Correct (AC) are integrals with respect to Recall, while R@XC indicates Recall at $X\%$ Correct. All precision and recall metrics use information as the specificity measure (not depth). The error ranges are two standard deviations.

	From operating curves				Majority	Leaf
%	AP	AC	R@90C	R@95C	F_1	F_1
Flat	66.90 ±0.16	64.71 ±0.18	52.44 ±0.42	45.88 ±0.30	63.68 ±0.19	69.86 ±0.13
Multilabel focal	57.56	53.25	32.84	26.94	63.30	63.80
Cond softmax [41]	60.72 ±0.11	57.47 ±0.13	42.03 ±0.12	35.71 ±0.20	58.54 ±0.20	65.27 ±0.11
Cond sigmoid [6]	60.64 ±0.17	57.13 ±0.16	40.52 ±0.13	34.29 ±0.18	56.73 ±0.07	65.52 ±0.19
Deep RTC [51] (sigmoid)	66.13	60.88	31.50	17.49	**71.64**	**71.64**
Deep RTC [51] (softmax)	**68.78**	**66.76**	*55.53*	*48.99*	*67.96*	*71.53*
Soft-max-descendant	66.46 ±0.37	64.19 ±0.39	49.15 ±0.36	43.29 ±0.56	60.99 ±0.33	70.01 ±0.33
Soft-max-margin	*67.63* ±0.25	*65.97* ±0.26	**56.53** ±0.28	**50.65** ±0.41	67.92 ±0.27	69.89 ±0.24

momentum 0.9, cosine schedule [33], batch size 64, initial learning rate 0.01 and weight decay 0.0003. Most experiments were conducted on a single machine with one Nvidia A6000 GPU. Each epoch of iNat21-Mini takes about 20 minutes (5-8 minibatches per second). To obtain error-bars, we used a second, larger machine with 8 Nvidia V100 GPUs that was shared with other users. The code was implemented using the Pytorch library [38] and will be released under an MIT license.

6.1 Operating characteristic curves

Our main experiment trains and evaluates each of the methods on the iNat21-Mini dataset to obtain operating curves. The results presented here are $\mu \pm 2\sigma$ from five trials[1] with different random seeds. Figure 1 presents curves for precision-vs-recall and correct-vs-exact. The most striking observation is that all methods except Deep RTC [51] and our soft-max-margin loss are effectively dominated by the flat softmax classifier at all operating points. For the precision-recall trade-off, Deep RTC (with softmax) is the best at high recall and soft-max-margin is the best at high precision. The soft-max-margin has a distinct advantage in the correct-exact trade-off. The least effective method is the binary multilabel baseline, followed by the two approaches that learn top-down conditional distributions [41, 6]. The singular operating points achieved by different inference methods are also depicted in the figure, with the operating curve always containing majority inference and leaf inference as special cases. For parameterizations $p(\cdot|x) = q(\theta)$ that respect the hierarchy, specificity and correctness should be monotonic in τ for $\tau > 0.5$, and a dashed line is used for the segment with $\tau < 0.5$. It is observed that plurality inference enables more-specific predictions than majority inference, but it is often below the operating curve. In fact, most inference methods lie on or below the curve. Further enquiry is required to understand the cause of the "knot" observed in the curves for

[1]Only one trial could be performed for the multilabel focal loss and Deep RTC due to compute constraints. However, these fall well outside of the range of the other methods, enabling rejection of the null hypothesis.

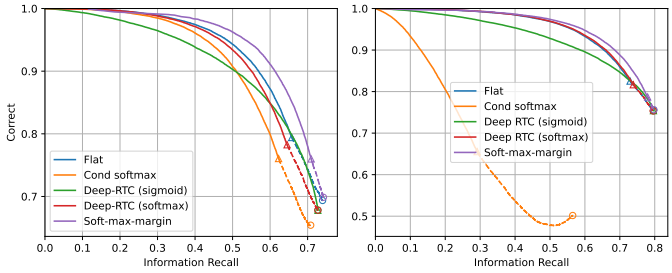


Figure 2: ImageNet-1k operating curves for (left) ResNet-50 from scratch and (right) linear model with CLIP ViT-B/32 features.

Table 2: iNat21-Mini accuracy at level j when training at level i.

%	$j = 3$	4	5	6
$i = 3$	82.7			
4	85.2	60.7		
5	86.6	64.6	50.6	
6	86.9	68.2	56.8	44.6
7	**87.3**	**70.4**	**59.8**	**49.2**

the soft-max-descendant method. Critically, the curves are much more useful than single operating points for the purposes of algorithm development and selection.

Table 1 presents several integral and intercept metrics obtained from the curves. For comparison, the F_1 metrics for majority and leaf inference are shown. Flat softmax, Deep RTC [51] and our proposed methods achieve the best results across all metrics, although Deep RTC is less effective in the intercept metrics, which measure recall at high rates of correctness.

Several of the proposed loss functions have additional hyper-parameters to specify. Figure 4 in the appendix examines the impact of (i) label smoothing in the flat softmax, (ii) focal loss parameters in the multilabel classifier, (iii) exponential discount in HXE [3] and (iii) cut probability in the Deep RTC loss [51]. Label smoothing and HXE include the flat softmax as a special case, and this was the optimal hyper-parameter choice in both cases. For this reason, they were excluded from our main experiment. The optimal cut probability for Deep RTC (sigmoid) was zero, meaning it reduces to a softmax loss with logits obtained by parameter inheritance. This was surprising, as it gives little incentive for the model to increase the scores of the internal nodes.

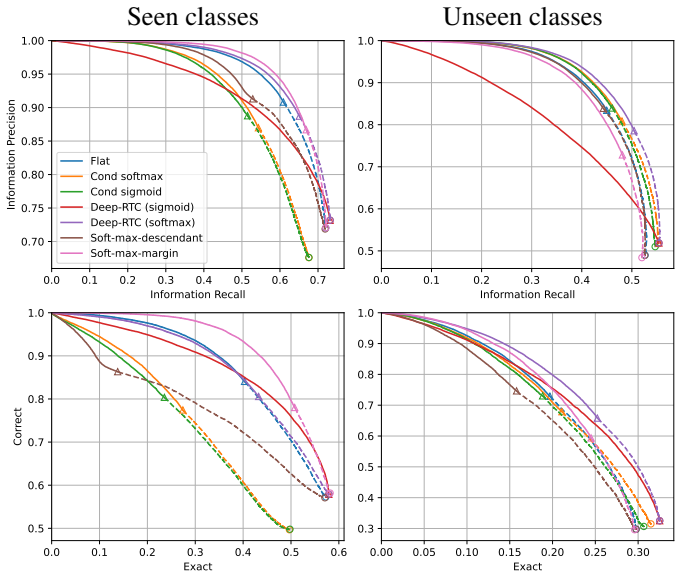
We conducted additional experiments with a subset of loss functions on ImageNet-1k using the hierarchy of [3], training a ResNet-50 model from scratch. Figure 2 presents operating curves for this experiment. The trends are consistent: our soft-max-margin method achieves a better trade-off than existing methods, Deep RTC (sigmoid) is strong at high recall but mediocre for high precision, and the conditional softmax is worse than the flat softmax across the full range. We used the same training configuration except only trained for 15 epochs. Figure 2 also shows the impact of using each loss function to train a *linear* model on top of the feature representation of a CLIP [40] model with ViT-B/32 [15] architecture. The soft-max-margin still provides an improvement over the baseline, whereas the (non-) Deep RTC model with softmax normalization is mathematically equivalent to a flat softmax (except for weight decay). Compared to training from scratch, higher recall is achieved, presumably due the strong feature representation. However, the conditional softmax is much worse, suggesting that the high-level classes are less easily separated in this representation. The linear model was trained for 20 epochs with batch size 256, initial learning rate 0.1 and weight decay 10^{-5}.

6.2 Flat classifiers at different levels

It was unexpected that the conditional softmax would fail to out-perform the flat softmax at *any* operating point. One possible explanation is that the high-level classes are better learned using a union of low-level classifiers than a single high-level classifier because the latter effectively ignores the fine-grained annotations. To investigate this hypothesis, we trained flat models at each level of the hierarchy and compared the classifiers trained at level i with those trained at levels $i + 1$, $i + 2$ and so on, shown in Table 2. Classifiers trained at deeper levels consistently achieved higher accuracy, although with diminishing returns. This motivated the design of our proposed loss functions, which predict scores for low-level and high-level classes.

6.3 Unseen classes

We hypothesized that the parameterizations which explicitly estimate scores for internal nodes might achieve better generalization to examples from "unseen" classes, which belong to an internal node but not a leaf node. To evaluate the ability of models to correctly classify such examples, we randomly



% AP (wrt Recall)	All	Seen	Unseen
Flat	*59.2*	69.3	49.5
Cond softmax	57.3	63.4	*51.6*
Cond sigmoid	56.9	63.1	51.1
Deep RTC (sigmoid)	57.1	68.1	46.0
Deep RTC (softmax)	**61.3**	**70.6**	**52.3**
Soft-max-descendant	58.3	68.1	49.2
Soft-max-margin	59.1	*70.0*	48.3

% AC (wrt Exact)	All	Seen	Unseen
Flat	36.7	50.8	23.5
Cond softmax	31.0	38.9	24.1
Cond sigmoid	30.3	38.4	23.4
Deep RTC (sigmoid)	38.1	*51.2*	*25.3*
Deep RTC (softmax)	*38.2*	51.2	**26.4**
Soft-max-descendant	32.1	44.9	22.0
Soft-max-margin	**38.7**	**54.0**	23.9

Figure 3: Operating curves for the out-of-distribution experiment on iNat21, divided into examples from seen and unseen classes. The top-down methods (conditional softmax, conditional sigmoid) are much more competitive on unseen classes. The soft-max-margin is most effective on seen classes but Deep RTC with softmax normalization is the most robust.

select half of the leaf nodes and remove them from the dataset *and* the hierarchy during training. At test time, examples from these classes were preserved with their labels projected onto the sub-tree.

Figure 3 presents the operating curves and integral metrics for the examples from seen and unseen classes. The conditional softmax [41] and conditional sigmoid [6] are much more competitive for unseen classes, although Deep RTC (softmax) remains the most effective. The soft-max-margin approach is excellent for seen classes and relatively poor for unseen classes, suggesting that it learns a highly tailored model that could be worse in an out-of-distribution setting. The soft-max-descendant approach is uncompetitive in both settings.

7 Conclusion

This work has presented a new approach for evaluating hierarchical classifiers for the prediction of non-leaf nodes. It was shown that the flat softmax classifier, despite having no knowledge of the hierarchy during training, is nonetheless a competitive baseline. Several novel loss functions were proposed, and the soft-max-margin loss was shown to out-perform several existing methods. On the other hand, the proposed soft-max-descendant losses were shown to be relatively ineffective. Surprisingly, top-down classifiers were found to be inferior across the entire operating range. Evidence suggests that this is due to the difficulty of learning a classifier to separate high-level classes without fine-grained supervision. The top-down classifiers were shown to be more effective on a synthetic out-of-distribution experiment, where recognition of internal-node classes is required.

One potential disadvantage of the soft-max-margin loss is that it requires the practitioner to choose the margin. Future work could investigate different margins, direct optimization of the area under the curve [4] or the impact of calibration [22]. It would also be interesting to compare the losses for long-tail learning in a class hierarchy, and to investigate other problems where the soft-max-margin loss could be applied. One limitation of our general approach is that it may not easily scale to millions of classes, where embeddings can be more suitable.

Social impact. While hierarchical classifiers may mitigate some of the risks of misclassification due to their more logical modes of failure, this may enable more widespread use of automatic classification or lead human users to place excessive trust in the system. Caution should always be exercised when employing automatic classification in consequential settings. A class hierarchy with *tree* structure may also encourage an oversimplified view of a problem due to the inability for classes to intersect.

References

[1] K. Ahmed, M. H. Baig, and L. Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision*, pages 516–532. Springer, 2016.

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1425–1438, 2015.

[3] L. Bertinetto, R. Mueller, K. Tertikas, S. Samangooei, and N. A. Lord. Making better mistakes: Leveraging class hierarchies with deep networks. In *CVPR*, pages 12506–12515, 2020.

[4] A. Brown, W. Xie, V. Kalogeiton, and A. Zisserman. Smooth-AP: Smoothing the path towards large-scale image retrieval. In *European Conference on Computer Vision*, pages 677–694. Springer, 2020.

[5] C.-A. Brust, B. Barz, and J. Denzler. Making every label count: Handling semantic imprecision by integrating domain knowledge. In *ICPR*, pages 6866–6873. IEEE, 2020.

[6] C.-A. Brust and J. Denzler. Integrating domain knowledge: using hierarchies to improve deep classifiers. In *Asian Conference on Pattern Recognition*, pages 3–16. Springer, 2019.

[7] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 78–87, 2004.

[8] D. Chang, K. Pang, Y. Zheng, Z. Ma, Y.-Z. Song, and J. Guo. Your "flamingo" is my "bird": Fine-grained, or not. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11476–11485, 2021.

[9] E. Costa, A. Lorena, A. Carvalho, and A. Freitas. A review of performance evaluation measures for hierarchical classifiers. In *Evaluation Methods for Machine Learning II: Papers from the AAAI-2007 workshop*, pages 1–6, 2007.

[10] J. Davis, T. Liang, J. Enouen, and R. Ilin. Hierarchical semantic labeling with adaptive confidence. In *International Symposium on Visual Computing*, pages 169–183. Springer, 2019.

[11] J. Davis, T. Liang, J. Enouen, and R. Ilin. Hierarchical classification with confidence using generalized logits. In *ICPR*, pages 1874–1881. IEEE, 2021.

[12] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision*, pages 71–84. Springer, 2010.

[13] 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*, pages 248–255. Ieee, 2009.

[14] 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*, pages 3450–3457. IEEE, 2012.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[16] S. Edunov, M. Ott, M. Auli, D. Grangier, and M. Ranzato. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364, 2018.

[17] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. DeViSE: A deep visual-semantic embedding model. *Advances in Neural Information Processing Systems*, 26, 2013.

[18] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *2011 International Conference on Computer Vision*, pages 2072–2079. IEEE, 2011.

[19] K. Gimpel and N. A. Smith. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736, 2010.

[20] W. Goo, J. Kim, G. Kim, and S. J. Hwang. Taxonomy-regularized semantic deep convolutional neural networks. In *European Conference on Computer Vision*, pages 86–101. Springer, 2016.

[21] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[22] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[23] Y. Guo, Y. Liu, E. M. Bakker, Y. Guo, and M. S. Lew. CNN-RNN: A large-scale hierarchical image classification framework. *Multimedia Tools and Applications*, 77(8):10251–10271, 2018.

[24] A. Gupta, P. Dollar, and R. Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019.

[25] 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*, pages 770–778, 2016.

[26] S. Karthik, A. Prabhu, P. K. Dokania, and V. Gandhi. No cost likelihood manipulation at test time for making better mistakes in deep networks. In *International Conference on Learning Representations*, 2021.

[27] J. Kim, Y. Park, G. Kim, and S. J. Hwang. SplitNet: Learning to semantically split deep networks for parameter reduction and model parallelization. In *International Conference on Machine Learning*, pages 1866–1874. PMLR, 2017.

[28] A. Kosmopoulos, I. Partalas, E. Gaussier, G. Paliouras, and I. Androutsopoulos. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery*, 29(3):820–865, 2015.

[29] O. Z. Kraus, J. L. Ba, and B. J. Frey. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32(12):i52–i59, 2016.

[30] Z. Kuang, J. Yu, Z. Li, B. Zhang, and J. Fan. Integrating multi-level deep learning and concept ontology for large-scale visual recognition. *Pattern Recognition*, 78:198–214, 2018.

[31] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, et al. The Open Images Dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[33] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[34] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*, 2020.

[35] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, volume 5, pages 246–252. Citeseer, 2005.

[36] R. Müller, S. Kornblith, and G. E. Hinton. When does label smoothing help? *Advances in Neural Information Processing Systems*, 32, 2019.

[37] C. Murdock, Z. Li, H. Zhou, and T. Duerig. Blockout: Dynamic model selection for hierarchical deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2583–2591, 2016.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[39] P. Pletscher, C. S. Ong, and J. M. Buhmann. Entropy and margin maximization for structured output learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 83–98. Springer, 2010.

[40] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[41] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, pages 7263–7271, 2017.

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[43] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR 2011*, pages 1481–1488. IEEE, 2011.

[44] C. N. Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1):31–72, 2011.

[45] M. Sun, W. Huang, and S. Savarese. Find the best path: An efficient and accurate classifier for image hierarchies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 265–272, 2013.

[46] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 104, 2004.

[47] G. Van Horn, E. Cole, S. Beery, K. Wilber, S. Belongie, and O. Mac Aodha. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12884–12893, 2021.

[48] N. Verma, D. Mahajan, S. Sellamanickam, and V. Nair. Learning hierarchical similarity metrics. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2280–2287. IEEE, 2012.

[49] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.

[50] C. Wu, M. Tygert, and Y. LeCun. A hierarchical loss and its problems when classifying non-hierarchically. *PLOS ONE*, 14(12):e0226222, 2019.

[51] T.-Y. Wu, P. Morgado, P. Wang, C.-H. Ho, and N. Vasconcelos. Solving long-tailed recognition with deep realistic taxonomic classifier. In *ECCV*, pages 171–189. Springer, 2020.

[52] Z. Wu and M. Palmer. Verb semantics and lexical selection. *arXiv preprint cmp-lg/9406033*, 1994.

[53] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu. HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2740–2748, 2015.

[54] H. Zhao, X. Puig, B. Zhou, S. Fidler, and A. Torralba. Open vocabulary scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2002–2010, 2017.

[55] X. Zhu and M. Bain. B-CNN: Branch convolutional neural network for hierarchical classification. *arXiv preprint arXiv:1709.09890*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] in the final section.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] in the final section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Code and artifacts will be released after acceptance.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] at the start of the experimental section.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] for the main experiment in Figure 1 and Table 1.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] at the start of the experimental section.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We have cited the iNat21 and ImageNet datasets, the ImageNet class tree, and Pytorch for the library and pretrained models.

(b) Did you mention the license of the assets? [No] The iNaturalist dataset has an MIT license. ImageNet is available to researchers for non-commercial use. The ImageNet class tree of Bertinetto et al. [3] was released under a CC BY-NC-SA 4.0 license.

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We use image datasets without "person" classes or photograph metadata.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]