Large-Scale Object Classification using Label Relation Graphs

Jia Deng[†*], Nan Ding[*], Yangqing Jia[*], Andrea Frome[*], Kevin Murphy[*], Samy Bengio[*], Yuan Li[*], Hartmut Neven[*], Hartwig Adam[*]

University of Michigan[†], Google Inc.[*]

Abstract. In this paper we study how to perform object classification in a principled way that exploits the rich structure of real world labels. We develop a new model that allows encoding of flexible relations between labels. We introduce Hierarchy and Exclusion (HEX) graphs, a new formalism that captures semantic relations between any two labels applied to the same object: mutual exclusion, overlap and subsumption. We then provide rigorous theoretical analysis that illustrates properties of HEX graphs such as consistency, equivalence, and computational implications of the graph structure. Next, we propose a probabilistic classification model based on HEX graphs and show that it enjoys a number of desirable properties. Finally, we evaluate our method using a large-scale benchmark. Empirical results demonstrate that our model can significantly improve object classification by exploiting the label relations.

Keywords: Object Recognition, Categorization

1 Introduction

Object classification, assigning semantic labels to an object, is a fundamental problem in computer vision. It can be used as a building block for many other tasks such as localization, detection, and scene parsing. Current approaches typically adopt one of the two classification models: multiclass classification, which predicts one label out of a set of mutually exclusive labels(e.g. entries in ILSVRC [9]), or binary classifications, which make binary decisions for each label independently(e.g. entries in PASCAL VOC classification competitions [13]).

Both models, however, do not capture the complexity of semantic labels in the real world. Multiclass classification tasks typically assume a set of mutually exclusive labels. Although efforts have been made to artificially constrain the label set in benchmarks (e.g. the ImageNet Challenges [9] select a subset of mutually exclusive labels from WordNet), this assumption becomes increasingly impractical as we consider larger, more realistic label sets. This is because the same object can often be described by multiple labels. An object classified as "husky" is automatically a "dog"; meanwhile it may or may not be a "puppy". Making "husky", "dog", and "puppy" mutually exclusive labels clearly violates real world semantics.

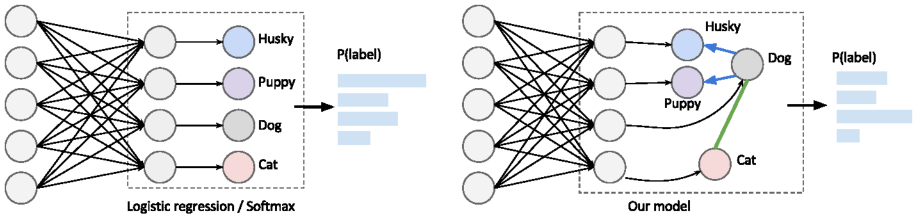

Fig. 1. Our model replaces traditional classifiers such as softmax or independent logistic regressions. It takes as input image features (e.g. from an underlying deep neural network) and outputs probabilities consistent with pre-specified label relations.

Independent binary classifiers, on the other hand, ignore the constraints between labels and can thus handle overlapping labels. But this can lead to inconsistent predictions such as an object being both a dog and a cat, or a husky but not a dog. In addition, discarding the label relations misses the opportunity to transfer knowledge during learning. For example, in practical settings training images are not always annotated to the most specific labels — many Internet images are simply labeled as “dog” instead of “husky” or “German Shepherd”. Intuitively, learning a good model for “dog” should benefit learning breeds of dogs (and vice versa) but training independent binary classifiers will not be able to capitalize on this potential knowledge transfer.

In this paper we study how to perform classification in a principled way that exploits the rich structure of real world labels. Our goal is to develop a new classification model that allows flexible encoding of relations based on prior knowledge, thus overcoming the limitations of the overly restrictive multiclass model and the overly relaxed independent binary classifiers (Fig. 1).

We first introduce Hierarchy and Exclusion (HEX) graphs, a new formalism allowing flexible specification of relations between labels applied to the same object: (1) mutual exclusion (e.g. an object cannot be dog and cat), (2) overlapping (e.g. a husky may or may not be a puppy and vice versa), and (3) subsumption (e.g. all huskies are dogs). We provide theoretical analysis on properties of HEX graphs such as consistency, equivalence, and computational implications.

Next, we propose a probabilistic classification model leveraging HEX graphs. In particular, it is a special type of Conditional Random Field (CRF) that encodes the label relations as pairwise potentials. We show that this model enjoys a number of desirable properties, including flexible encoding of label relations, predictions consistent with label relations, efficient exact inference for typical graphs, learning labels with varying specificity, knowledge transfer, and unification of existing models.

Finally, we evaluate our approach using the ILSVRC2012 [9], a large-scale benchmark for object classification. We also perform experiments on zero-shot recognition. Empirical results demonstrate that our model can significantly improve object classification by exploiting the label relations.

Our main contribution is *theoretical*, i.e. we propose a new formalism (HEX graphs), a new classification model, and a new inference algorithm, all grounded on rigorous analysis. In addition, we validate our approach using large-scale data, showing significant empirical benefits.

2 Related Work

Our approach draws inspirations from various themes explored in prior literature, including hierarchies, multilabel annotation, large-scale classification, and knowledge transfer. The main novelty of our work is unifying them into a single probabilistic framework with a rigorous theoretical foundation.

Exploiting hierarchical structure of object categories has a long history [34]. In particular, label hierarchies have been used to share representations [15, 2, 8, 17] and combine models [18, 38, 27].

Correlations between labels have been explored in multilabel annotation (e.g. [20]), but most prior work addresses contextual relations between co-occurring objects (e.g. [10]), as opposed to our setting of multiple labels on the same object. Lampert et al. and Bi and Kwok studied hierarchical annotations as structured predictions [24, 4, 3]. Chen et al. considered exclusive relations between labels [6]. To our knowledge we are the first to *jointly* model hierarchical *and* exclusive relations. By treating unobserved labels as latent variables, our approach also connects to prior work on learning from partial or incomplete labels [19, 7, 5].

Our model is a generalized multiclass classifier. It is designed to run efficiently and can thus be adapted to work with techniques developed for large-scale classification involving many labels and large datasets [32, 29].

Finally, by modeling the label relations and ensuring consistency between visual predictions and semantic relations, our approach relates to work in transfer learning [31, 30], zero-shot learning [28, 12, 25], and attribute-based recognition [1, 36, 33, 14], especially those that use semantic knowledge to improve recognition [16, 30] and those that propagate or borrow annotations between categories [26, 22].

3 Approach

3.1 Hierarchy and Exclusion (HEX) Graphs

We start by introducing the formalism of Hierarchy and Exclusion (HEX) graphs, which allow us to express prior knowledge about the labels. Due to space limit, all proofs and some lemmas are provided in the supplemental material.

Definition 1. *A HEX graph $G = (V, E_h, E_e)$ is a graph consisting of a set of nodes $V = \{v_1, \ldots, v_n\}$, directed edges $E_h \subseteq V \times V$, and undirected edges $E_e \subseteq V \times V$, such that the subgraph $G_h = (V, E_h)$ is a directed acyclic graph (DAG) and the subgraph $G_e = (V, E_e)$ has no self loop.*

Each node $v \in V$ represents a distinct label. An edge $(v_i, v_j) \in E_h$ is a hierarchy edge, indicating that label i subsumes label j, e.g. "dog" is a parent,

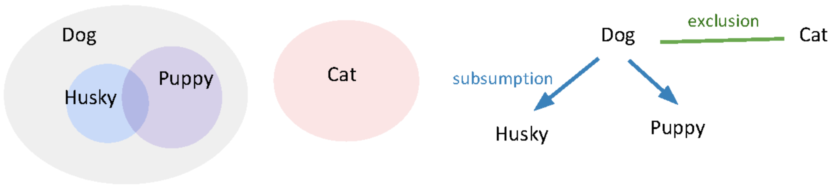

Fig. 2. HEX graphs capture relations between labels applied to the same object.

or superclass of "husky". The subgraph with only those edges form a semantic hierarchy. An edge $(v_i, v_j) \in E_e$ is called an exclusion edge, indicating that label v_i and v_j are mutually exclusive, e.g. an object cannot be dog and cat. If two labels share no edge, it means that they overlap, i.e. each label can turn on or off without constraining the other.

Alternatively one can think of each label as representing a set of object instances and the relations between labels as relations between (distinct) sets (Fig. 2). A hierarchy edge corresponds to one set containing the other. An exclusion edge corresponds to two disjoint sets. No edge corresponds to overlapping sets—the only remaining case.

It is worth nothing that while it is convenient to assume mutually exclusive children in a hierarchy, this is not the case for real world hierarchies, e.g. "child", "male", "female" are all children of "person" in WordNet. Thus we need a HEX graph to express those complexities.

Each label takes binary values, i.e. $v_i \in \{0, 1\}$. Each edge then defines a constraint on values the two labels can take. A hierarchy edge $(v_i, v_j) \in E_h$ means that an assignment of $(v_i, v_j) = (0, 1)$ (e.g. a husky but not a dog) is illegal. An exclusion edge $(v_i, v_j) \in E_e$ means that $(v_i, v_j) = (1, 1)$ (both cat and dog) is illegal. These local constraints of individual edges can thus define legal global assignments of labels.

Definition 2. *An assignment (state) $y \in \{0, 1\}^n$ of labels V in a HEX graph $G = (V, E_h, E_e)$ is legal if for any $(y_i, y_j) = (1, 1)$, $(v_i, v_j) \notin E_e$ and for any $(y_i, y_j) = (0, 1)$, $(v_i, v_j) \notin E_h$. The state space $S_G \subseteq \{0, 1\}^n$ of graph G is the set of all legal assignments of G.*

We now introduce some notations for further development. Let $\alpha(v_i)$ the set of all ancestors of $v_i \in V$ and $\bar{\alpha}(v_i) = \alpha(v_i) \cup \{v_i\}$ (ancestors and the node itself). Let $\sigma(v_i)$ be the set of all descendants of $v_i \in V$ and $\bar{\sigma}(v_i) = \sigma(v_i) \cup \{v_i\}$. Let $\epsilon(v_i)$ be the set of exclusive nodes, those sharing an exclusion edge with v_i. Let $o(v_i)$ be the set of overlapping nodes, those sharing no edges with v_i.

Consistency So far our definition of the HEX graph allows arbitrary placement of exclusion edges. This, however, can result in non-sensible graphs. For example, it allows label v_i and v_j to have both hierarchy and exclusion edges, i.e. v_i subsumes v_j and v_i and v_j are exclusive. This makes label v_j "dead", meaning that it is always 0 and thus cannot be applied to any object instance without causing a contradiction: if it takes 1, then it's parent v_i must take value 1 per the hierarchy edge and also take 0 per the exclusion edge. This demonstrates the need for a concept of consistency: a graph is consistent if every label is "active", i.e. it can take value either 1 or 0 and there always exists an assignment to the rest of labels such that the whole assignment is legal.

Definition 3. *A HEX graph $G = (V, E_h, E_e)$ is consistent if for any label $v_i \in V$, there exists two legal assignments $y, y' \in \{0,1\}^n$ such that $y_i = 1$ and $y'_i = 0$.*

Consistency is in fact solely determined by the graph structure—it is equivalent to the condition that for any label, there is no exclusion edge between its ancestors or between itself and its ancestors:

Theorem 1. *A HEX graph $G = (V, E_h, E_e)$ is consistent if and only if for any label $v_i \in V$, $E_e \cap (\bar{\alpha}(v_i) \times \bar{\alpha}(v_i)) = \emptyset$.*

We thus have an algorithm to check consistency without listing the state space. As will become clear, consistency is very important algorithmically.

3.2 Classification Model

A HEX graph encodes our prior knowledge about label relations. We can thus define a probabilistic classification model based on a HEX graph $G = (V, E_h, E_e)$. Let $x \in \mathcal{X}$ be an input and $f(x; w) : \mathcal{X} \to \mathcal{R}^n$ be a function with parameters w that maps an input image (or bounding box) to a set of scores, one for each label. The form of f is not essential (e.g. it can be a linear model $w_i^T x$ or a deep neural network) so we leave it unspecified. We define a joint distribution of an assignment of all labels $y \in \{0,1\}^n$ as a Conditional Random Field (CRF) [23]:

$$\tilde{P}(y|x) = \prod_i e^{f_i(x;w)[y_i=1]} \prod_{(v_i,v_j)\in E_h} [(y_i, y_j) \neq (0,1)] \prod_{(v_i,v_j)\in E_e} [(y_i, y_j) \neq (1,1)], \tag{1}$$

where $\tilde{P}$ is the unnormalized probability. The probability is then $\Pr(y|x) = \tilde{P}(y|x)/Z(x)$, where $Z(x) = \sum_{\hat{y}} \tilde{P}(\hat{y}|x)$ is the partition function. To compute the probability of a label, we marginalize all other labels. The scores $f_i(x; w)$ can be thought of as raw classification scores (local evidence) for each label and our model can convert them into marginal probabilities.

It is easy to verify a few facts about the model: (1) the probability of any illegal assignment is zero; (2) to compute the probability of a legal assignment, we take all labels with value 1, sum their scores, exponentiate, and then normalize; (3) the marginal label probabilities are always consistent with the label relations: probability of "dog" is always bigger than that of "husky" and probabilities of "dog" and "cat" cannot add to more than 1; (4) the model assumes an open world to gracefully handle unknown categories. For each node on the hierarchy, it is legal to assign itself a value 1 and all its descendants value 0. e.g. an object is a "dog" but none of the known dog subcategories. If the model sees novel dog subcategory, it can produce a large marginal for dog but will not be compelled to assign a large probability to a known subcategory.

Special cases A nice property is that it unifies standard existing models. If we use a HEX graph with pairwise exclusion edges and no hierarchy edges, i.e. all nodes are mutually exclusive, it is easy to verify that we arrive at the popular softmax (or multinomial regression) [1]: $\Pr(y_i = 1|x) = e^{f_i}/(1 + \sum_j e^{f_j})$. Another special case is when the HEX graph has no edges at all, i.e. all labels are independent. Eqn. 1 thus fully decomposes: $\Pr(y|x) = \prod_i e^{f_i[y_i=1]}/(1 + e^{f_i})$, i.e. independent logistic regressions for each label.

Joint hierarchical modeling We highlight another property: our model allows flexible joint modeling of hierarchical categories, thus enabling potential knowledge transfer. It is easy to verify that for all graphs the marginal probability of a label depends the sum of its ancestors' scores, i.e. $\Pr(y_i = 1|x)$ has the term $\exp(f_i + \sum_{v_j \in \alpha(v_i)} f_j)$, because all its ancestors must be 1 if the label takes value 1. Thus the model allows the score for "dog" to influence decisions about "husky". If the score function $f_i(x; w)$ is a linear model $w_i^T x$, then $\Pr(y_i = 1|x) \propto \exp(\tilde{w}_i^T x)$, where $\tilde{w}_i = w_i + \sum_{v_j \in \alpha(v_i)} w_j$, i.e the weights decompose along the hierarchy—the weights for "husky" are a combination of weights for "husky-ness", "dog-ness", and "animal-ness". This enables a form of sharing similar to prior work [8]. Note that depending on the applications, sharing can also be disabled by using constant scores (e.g. zeros).

Conversely, the probability of an internal node of the hierarchy also depends on the probabilities of its descendants because we need to marginalize over all possible states of the descendants. For example, if we have a tree hierarchy with mutually exclusive siblings, it can be shown that the unnormalized probability $\tilde{P}$ of an internal node is a simple recursive form involving its own score, its ancestors' scores, and the sum of unnormalized probabilities of its direct children $c(v_i)$, i.e. $\tilde{P}(y_i = 1|x) = \exp\left(f_i + \sum_{v_k \in \alpha(v_i)} f_k\right) + \sum_{v_j \in c(v_i)} \tilde{P}(y_j = 1|x)$. Again we can use constant scores for internal nodes, in which case the model is a collection of "local leaf models" and we simply sum the probabilities of leaves.

Learning In learning we maximize the (marginal) likelihood of the observed ground truth labels using stochastic gradient descent (SGD). Given training examples $\mathcal{D} = \{(x^{(l)}, y^{(l)}, g^{(l)})\}, l = 1, \ldots, m$, where $y^{(l)} \in \{0, 1\}^n$ is the complete ground truth label vector and $g^{(l)} \subseteq \{1, \ldots, n\}$ is the indices of the observed labels, the loss function is :

$$\mathcal{L}(\mathcal{D}, w) = -\sum_l \log \Pr(y_{g^{(l)}}^{(l)} | x^{(l)}; w) = -\sum_l \log \sum_{y: y_{g^{(l)}} = y_{g^{(l)}}^{(l)}} \Pr(y|x^{(l)}; w) \quad (2)$$

In training we often have incomplete ground truth. Labels can be at any level of the hierarchy. We may only know the object is a "dog" but uncertain about the specific breed. Incomplete ground truth also occurs with labels that are not hierarchical but can still overlap ("husky" and "puppy"). Our model can

[1] There is an additional constant 1 in the denominator because for n labels there are $n + 1$ states. The extra one is "none of the above", i.e. all labels zero. This makes no practical difference. See supplemental material for further discussions.

naturally handle this by treating the unobserved labels as latent variables and marginalizing them in computing the likelihood.

3.3 Efficient Inference

Inference—computing the partition function and marginalizing unobserved labels—is exponential in the number of labels if performed with brute force. Treated as a generic CRF, our model can easily be densely connected and full of loops, especially when there are many mutual exclusions, as is typical for object labels. Thus at first glance exact inference is intractable.

However, in this section we show that exact inference is tractable for a large family of HEX graphs with dense connections, especially in realistic settings. The main intuition is that when the graph is densely connected, the state space can be small due to the special form of our binary potentials—all illegal states have probability zero and they can simply be eliminated from consideration. One example is the standard multiclass setting where all labels are mutually exclusive, in which case the size of the state space is $O(n)$. On the other hand, when a graph is sparse, i.e. has small treewidth, then standard algorithms such as junction trees apply. Our algorithm will try to take the best of both worlds by transforming a graph in two directions: (1) to an equivalent sparse version with potentially small treewidth; (2) to an equivalent dense version such that we can afford to exhaustively list the state space for any subset of nodes. The final algorithm is a modified junction tree algorithm that can be proven to run efficiently for many realistic graphs.

Equivalence Two HEX graphs are *equivalent* if they have the same state space:

Definition 4. *HEX graphs G and G' are equivalent if $S_G = S_{G'}$.*

Intuitively equivalent graphs can arise in two cases. One is due to the transitivity of the subsumption relation—if "animal" subsumes "dog" and "dog" subsumes "husky" then "animal" should be implied to subsume "husky". Thus a hierarchy edge from "animal" to "husky" is redundant. The other case is that mutual exclusion can be implied for children by parents. For example, if "cat" and "dog" are exclusive, then all subclasses of "dog" should be implied to be exclusive with "cat". Thus an exclusion edge between "husky" and "cat" is redundant. Formally redundant edges are those that can be removed or added without changing the state space:

Definition 5. *Given a graph $G = (V, E_h, E_e)$, a directed edge $e \in V \times V$ (not necessarily in E_h) is redundant if $G' = (V, E_h \setminus \{e\}, E_e)$ and $G'' = (V, E_h \cup \{e\}, E_e)$ are both equivalent to G. An undirected edge $e \in V \times V$ (not necessarily in E_e) is redundant if $G' = (V, E_h, E_e \setminus \{e\})$ and $G'' = (V, E_h, E_e \cup \{e\})$ are both equivalent to G.*

For consistent graphs, redundant edges can be found by searching for certain graph patterns: for a directed hierarchy edge (v_i, v_j), it is redundant if and only if there is an alternative path from v_i to v_j. For an undirected exclusion edge

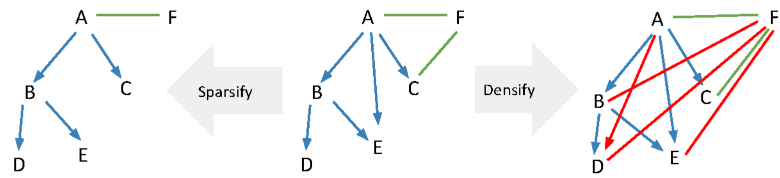

Fig. 3. Equivalent HEX graphs.

(v_i, v_j), it is redundant if and only if there is an another exclusion edge that connects their ancestors (or connects one node's ancestor to the other node).

Lemma 1. *Let $G = (V, E_h, E_e)$ be a consistent graph. A directed edge $e \in V \times V$ is redundant if and only if in the subgraph $G = (V, E_h)$ there exists a directed path from v_i to v_j and the path doesn't contain e. An undirected edge $e = (v_i, v_j) \in V \times V$ is redundant if and only if there exists an exclusion edge $e' = (v_k, v_l) \in E_e$ such that $v_k \in \bar{\alpha}(v_i)$, $v_l \in \bar{\alpha}(v_j)$ and $e \neq e'$.*

Lemma 1 in fact gives an algorithm to "sparsify" or "densify" a graph. We can remove one redundant edge a time until we obtain a *minimally sparse* graph. We can also add edges to obtain a *maximally dense* equivalent. (Fig. 3).

Definition 6. *A graph G is* minimally sparse *if it has no redundant edges. A graph G is* maximally dense *if every redundant edge is in G.*

In fact for a consistent graph, its minimally sparse or maximally dense equivalent graph is unique, i.e. we always arrive at the same graph regardless of the order we remove or add redundant edges:

Theorem 2. *For any consistent graphs G and G' that are both minimally sparse (or maximally dense), if $S_G = S_{G'}$, then $G = G'$.*

Thus given any consistent graph, we can "canonicalize" it by sparsifying or densifying it [2]. This can help us reason about the size of the state space.

Size of State Space If a graph has very dense connections, its state space tends to be tractable, such as the case of pairwise mutually exclusive labels. Intuitively, for labels applied to real world objects, there should be many mutual exclusions. For example, if we randomly pick two labels from the English dictionary, most likely they will have a relation of exclusion or subsumption when applied to the same object. In other words, for the same object, there shouldn't be too many overlapping labels. Otherwise the state space can grow exponentially with the amount of overlap. We can formalize this intuition by first introducing the quantity *maximum overlap*.

Definition 7. *The maximum overlap of a consistent graph $G = (V, E_h, E_e)$ is $\Omega_G = \max_{v \in V} |o_{\bar{G}}(v)|$, where $o_{\bar{G}}(v) = \{u \in V : (u, v) \notin \bar{E}_h \wedge (v, u) \notin \bar{E}_h \wedge (u, v) \notin \bar{E}_e\}$ and $\bar{G} = (V, \bar{E}_h, \bar{E}_e)$ is the maximally dense equivalent of G.*

[2] Note that Lemma 1 and Theorem 2 do not apply to inconsistent graphs because of "dead" nodes. See supplemental materials for more details.

Algorithm 1 Listing state space

```
function ListStateSpace(graph G)
    if G = ∅ then return ∅
    end if
    Let G = (V, E_h, E_e) and n = |V|.
    Pick an arbitrary v_i ∈ V.
    V^0 ← α(v_i) ∪ ϵ(v_i) ∪ o(v_i).
    G^0 ← G[V^0]
    S_{G^0} ← ListStateSpace(G^0)
    S_G^0 ← {y ∈ {0,1}^n : y_i = 0 ⋀ y_{σ(v_i)} = 0 ⋀ y_{V^0} ∈ S_{G^0}}.
    V^1 ← σ(v_i) ∪ o(v_i).
    G^1 ← G[V^1]
    S_{G^1} ← ListStateSpace(G^1)
    S_G^1 = {y ∈ {0,1}^n : y_i = 1 ⋀ y_{α(v_i)} = 1 ⋀ ϵ(v_i) = 0 ⋀ y_{V^1} ∈ S_{G^1}}
    return S_G^0 ∪ S_G^1.
end function
```

That is, we first convert a graph to its maximally dense equivalent and then take the max of the per-node overlap—the number of non-neighbours of a node. In other words, the overlap of a label is the number of other non-superset and non-subset labels you can additionally apply to the same object. We can now use the maximum overlap of a graph to bound the size of its state space:

Theorem 3. *For a consistent graph $G = (V, E_h, E_e)$, $|S_G| \leq (|V| - \Omega_G + 1)2^{\Omega_G}$.*

As an interesting fact, if a HEX graph consists of a tree hierarchy and exclusion edges between all siblings, then it's easy to verify that its maximum overlap is zero and its state space size is exactly $|V| + 1$, a tight bound in this case.

Listing State Space A tractable size of the state space is useful for inference only if the legal states can also be enumerated efficiently. Fortunately this is always the case for HEX graphs. To list all legal assignments for an input, we can first pick an arbitrary node as a "pivot" and fix its value to 1. Also we fix all its parents to 1 and exclusive neighbours to 0 because otherwise we would violate the constraints. We then recursively apply the same procedure to the subgraph induced by the rest of the nodes (children and non-neighbours). For each returned assignment of the subgraph, we generate a new assignment to the full graph by concatenating it with the fixed nodes. Similarly, we fix the pivot node to 0 and fix its children to 0, and then recurse on the subgraph induced by the rest of the nodes. See Alg. 1 for pseudo-code. We can formally prove that if the graph is consistent and maximally dense, this greedy procedure returns all legal assignments and runs in time linear in the size of the state space:

Lemma 2. *If graph $G = (V, E_h, E_e)$ is consistent and maximally dense, Alg. 1 runs in $O\left((|V| + |E_h| + |E_e|)|S_G|\right)$ time and returns the state space S_G.*

Full Inference Algorithm We now describe the full inference algorithm (Alg. 2). Given a graph, we first generate the minimally sparse and maximally dense equivalents. We treat the minimally sparse graph as a generic CRF and generate a junction tree. For each clique of the junction tree, we list its state space using

Algorithm 2 Exact Inference

Input: Graph $G = (V, E_h, E_e)$.
Input: Scores $f \in \mathcal{R}^{|V|}$.
Output: Marginals, e.g. $\Pr(v_i = 1)$.
1: $G^* \leftarrow$ SPARSIFY(G)
2: $\bar{G} \leftarrow$ DENSIFY(G)
3: $T \leftarrow$BUILDJUNCTIONTREE(G^*).
4: For each clique $c \in T$, $S_c \leftarrow$LISTSTATESPACE($\bar{G}[c]$).
5: Perform (two passes) message passing on T using only states S_c for each clique c.

the subgraph induced by the clique on the maximally dense graph. To do inference we run two passes of sum-product message passing on the junction tree, performing computation only on the legal states of each clique.

This algorithm thus automatically exploits dynamic programming for sparse regions of the graph and small state spaces for dense regions of the graph. For example, it is easy to verify that for pairwise mutually exclusive labels, the inference cost is $O(n)$, the same as hand-coded softmax, due to the small state space. For fully independent labels (no edges), the inference cost is also $O(n)$, the same as n hand-coded logistic regressions, because the junction tree is n disconnected cliques, each containing a single label. We can formally bound the complexity of the inference algorithm:

Theorem 4. *The complexity of the exact inference (Line 5 in Alg. 2) for graph* $G = (V, E_h, E_e)$ *is* $O\left(\min\{|V|2^w, |V|^2 2^{\Omega_G}\}\right)$*, where* w *is the width of the junction tree* T *(Line 3).*

Note that this is a worst case bound. For example, a graph can have two disconnected components. One has a large treewidth but small overlap (e.g. many mutual exclusions between object labels). The other has a small treewidth but large overlap (e.g. attribute labels). The bound in Theorem 4 for the whole graph will assume the worst, but Alg. 2 can perform inference for this graph efficiently by automatically treating the two components differently.

4 Experiments

4.1 Implementation

We implement our model as a standalone layer in a deep neural network framework. It can be put on top of any feed-forward architecture. The layer takes as input a set of scores $f(x; w) \in \mathbb{R}^n$ and outputs (marginal) probability of a given set of observed labels. During learning, we use stochastic gradient descent and compute the derivative $\frac{\partial \mathcal{L}}{\partial f}$, where $\mathcal{L}$ is the loss as defined in Eqn. 2 but treated here as a function of the label scores f instead of the raw input x. This derivative is then back propagated to the previous layers represented by $f(x; w)$.

For exact inference as described in Alg. 2, Step 1 to Step 4 (processing the graph, building the junctions trees, listing state space etc.) only depend on the graph structure and are performed offline. Only the message passing on the junction tree (Step 5) needs to be performed for each example online. Given a junction tree and the legal assignments for each clique, we perform a “dry run” of message passing and record the sequence of sum-product operations. Then the online inference for each example simply follows this pre-determined sum-product sequence and thus has negligible extra overhead compared to hand-coded implementations of softmax or independent logistic regressions.

4.2 Object classification on ImageNet

Dataset We evaluate our model using the ILSVRC2012 dataset [9]. It consists of 1.2M training images from 1000 object classes [3]. These 1000 classes are mutually exclusive leaf nodes of a semantic hierarchy based on WordNet that has 820 internal nodes. All images are labeled to the leaf nodes.

WordNet provides hierarchical relations but no exclusive relations. We thus adopt the assumption of “exclusive whenever possible”, i.e. putting an exclusive edge between two nodes unless it results in an inconsistent graph. This means that any two labels are mutually exclusive unless they share a descendant. Note that the WordNet hierarchy is a DAG instead of a tree. For example, “dog” appear under both “domestic animal” and ”canine”.

Setup In this experiment we test the hypothesis that our method can improve object classification by exploiting label relations, in particular by enabling joint modeling of hierarchical categories. To this end, we evaluate the recognition performance in the typical test setting—multiclass classification at the leaf level—but allow the training examples to be labeled at different semantic levels. This setting is of practical importance because when one collects training examples from the Internet, the distribution of labels follow a power law with many more images labeled at basic levels (“dog”,“car”) than at fine-grained levels (“husky”, “Honda Civic”). While it is obvious that with a semantic hierarchy, one can aggregate all training examples from leaf nodes to learn a classifier for internal nodes, the other direction—how to use higher level classes with many training examples to help train fine-grained classes with fewer examples—is not as clear.

Since ILSVRC2012 has no training examples at internal nodes, we create training examples for internal nodes by “relabelling” the leaf examples to their immediate parent(s) (Fig. 4), i.e. some huskies are now labeled as “dog”. Note that each image still has just one label; an image labeled as “husky” is not additionally labeled as “dog”.

We put our model on top of the convolutional neural network (CNN) developed by Krizhevsky et al. [21, 37]. Specifically, we replace the softmax classifier layer (fully connected units with a sigmoid activation function followed by normalization to produce the label probabilities) with our layer — fully connected units followed by our inference to produce marginals for each label (see Fig. 1 for an illustration). We only use fully connected units for the leaf nodes and fix input

[3] Since ground truth for test set is not released and our experiments involve non-standard settings and error measures, we use the validation set as our test images and tune all parameters using cross-validation on the training set.

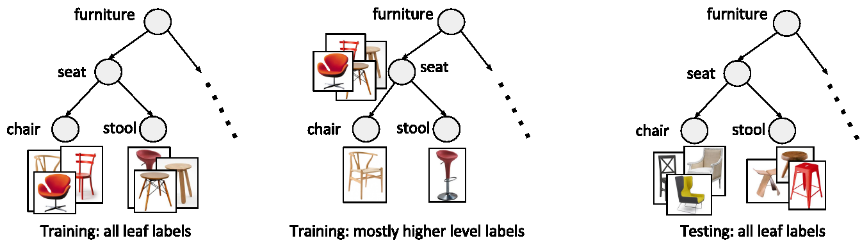

Fig. 4. Instead of training with data all labeled at leaf nodes, we train with examples leveled at different levels but still evaluate classification at leaf nodes during test.

scores (f_i in Eqn. 1) of internal nodes to zero because we found that otherwise it takes longer to train but offer no significant benefits on ILSVRC2012.

We compare our model with three baselines, all on top of the same CNN architecture and all trained with full back propagation from scratch: (1) softmax on leaf nodes only i.e. ignoring examples labeled to internal nodes; (2) softmax on all labels, i.e. treating all labels as mutually exclusive even though "dog" subsumes "husky"; (3) independent logistic regressions for each label [4]. For the logistic regression baseline, we need to specify positive and negative examples individually for each class—we add to positives by aggregating all examples from descendants and use as negatives all other examples except those from the ancestors. During test, for each method we use its output probabilities for the 1000 leaf nodes (ignoring others) to make predictions.

Results Table 1 reports the classification accuracy on the leaf nodes with different amounts of relabelling (50%, 90%, 95%, 99%). As a reference, our implementation of softmax with all examples labeled at leaf nodes (0% relabeling) gives a hit rate (accuracy) of 62.6% at top 1 and 84.3% at top 5 [5].

Our approach outperforms all baselines in all settings except that in the extreme case of 99% relabelling, our approach is comparable with the best baseline. Even with 90% of labels at leaf nodes "weakened" into internal nodes, we can still achieve 55.3% top 1 accuracy, not too big a drop from 62.6% by using all leaf training data. Also independent logistic regressions perform very abysmally, likely because of the lack of calibration among independent logistic regressions and varying proportions of positive examples for different classes.

Interestingly the baseline of softmax on all labels, which seems non-sensible as it trains "dog" against "husky", is very competitive. We hypothesize that this is because softmax treats examples in internal nodes as negatives. It is in fact a mostly correct assumption: the "dog" examples contain some huskies but are mostly other types of dogs. Softmax thus utilizes those negative examples effectively. On the other hand, our model treats labels in internal nodes as weak positive examples because the marginal of an internal node depends on the scores

[4] They are trained jointly but, unlike softmax, without normalization of probabilities.

[5] our model gives the same result—with scores for internal nodes fixed to zero it is equivalent to softmax up to an additional constant in the partition function.

relabeling	softmax-leaf	softmax-all	logistic	**ours**
50%	50.5(74.7)	56.4(79.6)	21.0(45.2)	**58.2(80.8)**
90%	26.2(47.3)	52.9(77.2)	9.3(27.2)	**55.3(79.4)**
95%	16.0(32.2)	50.8(76.0)	5.6(17.2)	**52.4(77.2)**
99%	2.5 (7.2)	**41.5**(68.1)	1.0(3.8)	**41.5(68.5)**

Table 1. Top 1 (top 5 in brackets) classification accuracy on 1000 classes of ILSVRC2012 with relabeling of leaf node data to internal nodes during training.

of its descendants. This is semantically correct, but makes no assumption about the proportion of real positive examples. This suggests a future direction of including stronger assumptions during learning to further improve our model.

In the case of 99% relabelling, softmax-all is comparable to our model (41.5% versus 41.5% top 1 accuracy). This is likely because there are too few images labeled at the leaf nodes (around 10 images per class). During learning, our model "softly" assigns the images labeled at internal nodes to leaves (by inferring the distribution of unobserved labels), which improves the leaf models (weights for generating leaf input scores). However, with too few training examples labeled at leaf nodes, the leaf models can "drift away" with noisy assignment.

4.3 Zero-shot recognition on Animals with Attributes

In this experiment we evaluate whether our HEX graph based model can be used to model relations between objects and attributes, although not by design. We use the Animal with Attributes (AWA) dataset [25] that includes images from 50 animal classes. For each animal class, it provides binary predicates for 85 attributes, e.g. for zebra, "stripes" is yes and "eats fish" is no. We evaluate the zero-shot setting where training is performed using only examples from 40 classes and testing is on classifying the 10 unseen classes. The binary predicates and the names of the unseen classes are both known a priori.

Here we show that our model can be easily adapted to perform zero-shot recognition exploiting object-attribute relations. First we build a HEX graph for all animals and attributes by assuming mutual exclusion between the animal classes and then adding the object-attribute relations: "zebra has stripes" establish a subsumption edge from "stripes" to "zebra"; "zebra doesn't eat fish" means an exclusion edge between them.

We use the same published, pre-computed features x from [25] and use a linear model to map the features to score $f_i(x) = w_i^T \phi(x)$ for label i, where $\phi(x)$ is computed by the Nystrom method [35] with rank 8000 to approximate the Chi-squared kernel used in [25]. In training, we observe the class labels for examples from the first 40 classes. The system is trained to maximize the likelihood of the class labels, but indirectly it learns to also predict the (latent) attributes given the image features. At test time, the class labels are not observed (and are drawn from a distinct set of 10 new labels); however, the model can predict the attributes given the image, and since the mapping from attributes to classes is known (for all 50 classes), the model can also (indirectly) predict the novel class label. This can be done by performing inference in the model and reading out

Fig. 5. We can build a HEX graph using relations between objects and attributes.

the marginals for the 10 unknown classes. We achieve a 38.5% mean accuracy (and 44.2% using the recently released DECAF features [11]) as compared to 40.5% in [25]. Given that our model is not explicitly designed for this task and the kernel approximation involved, this is a very encouraging result.

4.4 Efficiency of Inference

We evaluate the empirical efficiency of inference by counting the number of basic operations (summations and multiplications) needed to compute the marginals for all labels from the scores (not including computation for generating the scores). For the HEX graph used with ILSVRC2012, i.e. a DAG hierarchy and dense mutual exclusions, the inference cost is 6 relative to softmax for the same number of labels. For AWA, the cost is 294 relative to softmax. In both cases, the overhead is *negligible* because while softmax costs $O(n)$, simply computing the scores from d-dimensional inputs costs $O(nd)$ using a linear model ($d = 4096$ for ILSVRC2012 and $d = 8000$ for AWA), let alone multilayer neural networks.

Moreover, it is worth noting that the HEX graph for AWA (Fig. 5) has 85 overlapping attributes with no constraints between them. Thus it has at least 2^{85} legal states. Also it has a large treewidth—the 50 animal classes are fully connected with exclusion edges, a complexity of 2^{50} for a naive junction tree algorithm. But our inference algorithm runs with negligible cost. This again underscores the effectiveness of our new inference algorithm in exploiting both dynamic programming and small state space.

5 Discussions and Conclusions

We now briefly mention a couple of possible future directions. Efficient exact inference depends on the relations being "absolute" such that many states have probability zero. But it does not allow non-absolute relations ("taxi is mostly yellow"). Thus it remains an open question how to use non-absolute relations in conjunction with absolute ones. Another direction is to integrate this model developed for single objects into a larger framework that considers spatial interactions between objects.

To conclude, we have provided a unified classification framework that generalizes existing models. We have shown that it is flexible, theoretically principled, and empirically useful. Finally, we note that although motivated by object classification, our approach is very general in that it applies to scenes, actions, and any other domains with hierarchical and exclusive relations.

References

1. Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C.: Label-embedding for attribute-based classification. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. pp. 819–826. IEEE (2013)
2. Amit, Y., Fink, M., Srebro, N., Ullman, S.: Uncovering shared structures in multiclass classification. In: Proceedings of the 24th international conference on Machine learning. pp. 17–24. ACM (2007)
3. Bi, W., Kwok, J.T.: Multi-label classification on tree-and dag-structured hierarchies. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11). pp. 17–24 (2011)
4. Bi, W., Kwok, J.T.: Mandatory leaf node prediction in hierarchical multilabel classification. In: NIPS. pp. 153–161 (2012)
5. Bucak, S.S., Jin, R., Jain, A.K.: Multi-label learning with incomplete class assignments. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 2801–2808. IEEE (2011)
6. Chen, X., Yuan, X.T., Chen, Q., Yan, S., Chua, T.S.: Multi-label visual classification with label exclusive context. In: Computer Vision (ICCV), 2011 IEEE International Conference on. pp. 834–841. IEEE (2011)
7. Cour, T., Sapp, B., Taskar, B.: Learning from partial labels. The Journal of Machine Learning Research 12, 1501–1536 (2011)
8. Dekel, O., Keshet, J., Singer, Y.: Large margin hierarchical classification. In: Proceedings of the twenty-first international conference on Machine learning. p. 27. ACM (2004)
9. Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., Fei-Fei, L.: Imagenet large scale visual recognition challenge 2012. www.image-net.org/challenges/LSVRC/2012 (2012)
10. Desai, C., Ramanan, D., Fowlkes, C.C.: Discriminative models for multi-class object layout. International journal of computer vision 95(1), 1–12 (2011)
11. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 (2013)
12. Elhoseiny, M., Saleh, B., Elgammal, A.: Write a classifier: Zero-shot learning using purely textual descriptions. ICCV (2013)
13. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision 88(2), 303–338 (2010)
14. Farhadi, A., Endres, I., Hoiem, D.: Attribute-centric recognition for cross-category generalization. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 2352–2359. IEEE (2010)
15. Fergus, R., Bernal, H., Weiss, Y., Torralba, A.: Semantic label sharing for learning with many categories. In: Computer Vision–ECCV 2010, pp. 762–775. Springer (2010)
16. Frome, A., Corrado, G.S., Shlens, J., Bengio, S., Dean, J., Mikolov, T.: Devise: A deep visual-semantic embedding model. Advances in Neural Information Processing Systems pp. 2121–2129 (2013)
17. Hwang, S.J., Sha, F., Grauman, K.: Sharing features between objects and their attributes. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 1761–1768. IEEE (2011)

18. Jia, Y., Abbott, J.T., Austerweil, J., Griffiths, T., Darrell, T.: Visual concept learning: Combining machine vision and bayesian generalization on concept hierarchies. In: Advances in Neural Information Processing Systems. pp. 1842–1850 (2013)
19. Jin, R., Ghahramani, Z.: Learning with multiple labels. In: Advances in neural information processing systems. pp. 897–904 (2002)
20. Kang, F., Jin, R., Sukthankar, R.: Correlated label propagation with application to multi-label learning. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. vol. 2, pp. 1719–1726. IEEE (2006)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. vol. 1, p. 4 (2012)
22. Kuettel, D., Guillaumin, M., Ferrari, V.: Segmentation propagation in imagenet. In: Computer Vision–ECCV 2012, pp. 459–473. Springer (2012)
23. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. pp. 282–289. ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001), http://dl.acm.org/citation.cfm?id=645530.655813
24. Lampert, C.H.: Maximum margin multi-label structured prediction. In: NIPS. vol. 11, pp. 289–297 (2011)
25. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 951–958. IEEE (2009)
26. Lim, J.J., Salakhutdinov, R., Torralba, A.: Transfer learning by borrowing examples for multiclass object detection. In: Neural Information Processing Systems (NIPS) (2011)
27. Marszalek, M., Schmid, C.: Semantic hierarchies for visual object recognition. In: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. pp. 1–7. IEEE (2007)
28. Palatucci, M., Pomerleau, D., Hinton, G.E., Mitchell, T.M.: Zero-shot learning with semantic output codes. In: NIPS. vol. 3, pp. 5–2 (2009)
29. Perronnin, F., Akata, Z., Harchaoui, Z., Schmid, C.: Towards good practice in large-scale learning for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 3482–3489. IEEE (2012)
30. Rohrbach, M., Stark, M., Schiele, B.: Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 1641–1648. IEEE (2011)
31. Rohrbach, M., Stark, M., Szarvas, G., Gurevych, I., Schiele, B.: What helps where–and why? semantic relatedness for knowledge transfer. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 910–917. IEEE (2010)
32. Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J.: Image classification with the fisher vector: Theory and practice. International journal of computer vision 105(3), 222–245 (2013)
33. Sharmanska, V., Quadrianto, N., Lampert, C.H.: Augmented attribute representations. In: Computer Vision–ECCV 2012, pp. 242–255. Springer (2012)
34. Tousch, A.M., Herbin, S., Audibert, J.Y.: Semantic hierarchies for image annotation: A survey. Pattern Recognition 45(1), 333–345 (2012)
35. Williams, C., Seeger, M.: Using the nyström method to speed up kernel machines. In: Advances in Neural Information Processing Systems 13. Citeseer (2001)
36. Yu, F.X., Cao, L., Feris, R.S., Smith, J.R., Chang, S.F.: Designing category-level attributes for discriminative visual recognition. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. pp. 771–778. IEEE (2013)

37. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional neural networks. arXiv preprint arXiv:1311.2901 (2013)
38. Zweig, A., Weinshall, D.: Exploiting object hierarchy: Combining models from different category levels. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. pp. 1–8. IEEE (2007)