Taxonomy-Regularized Semantic Deep Convolutional Neural Networks

Wonjoon Goo[1], Juyong Kim[1], Gunhee Kim[1], Sung Ju Hwang[2]

[1]Computer Science and Engineering, Seoul National University, Seoul, Korea
[2]School of Electrical and Computer Engineering, UNIST, Ulsan, South Korea
{wonjoon,gem0521,gunhee}@snu.ac.kr, sjhwang@unist.ac.kr

Project page: https://github.com/hiwonjoon/eccv16-taxonomy

Abstract. We propose a novel convolutional network architecture that abstracts and differentiates the categories based on a given class hierarchy. We exploit grouped and discriminative information provided by the taxonomy, by focusing on the general and specific components that comprise each category, through the min- and difference-pooling operations. Without using any additional parameters or substantial increase in time complexity, our model is able to learn the features that are discriminative for classifying often confused sub-classes belonging to the same superclass, and thus improve the overall classification performance. We validate our method on CIFAR-100, Places-205, and ImageNet Animal datasets, on which our model obtains significant improvements over the base convolutional networks.

Keywords: deep learning, object categorization, taxonomy, ontology

1 Introduction

Deep convolutional neural networks (CNNs) [12–14, 18] have received much attention in recent years, due to its success on object categorization and many other visual recognition tasks. They have achieved the state-of-the-art performances for challenging categorization datasets such as ImageNet [3], owing to their ability to learn compositional representations for the target tasks, through multiple levels of non-linear transformations. This multi-layer learning is biologically inspired by the human visual system that also processes the visual stimuli through a similar hierarchical cascade.

However, while the deep CNNs closely resemble such low-level human visual processing systems, they pay less attention to the high-level reasoning employed for categorization. When performing categorization, humans do not treat each category as an independent entity that is different from everything else. Rather, they understand each object category in relation to others, performing generalization and specialization focusing on their commonalities and differences, either through observations or by the learned knowledge.

For example, consider the images of animals at the bottom of Figure 1. Each image shows a different animal species (*e.g. cheetah*, *jaguar*, *leopard*). How can

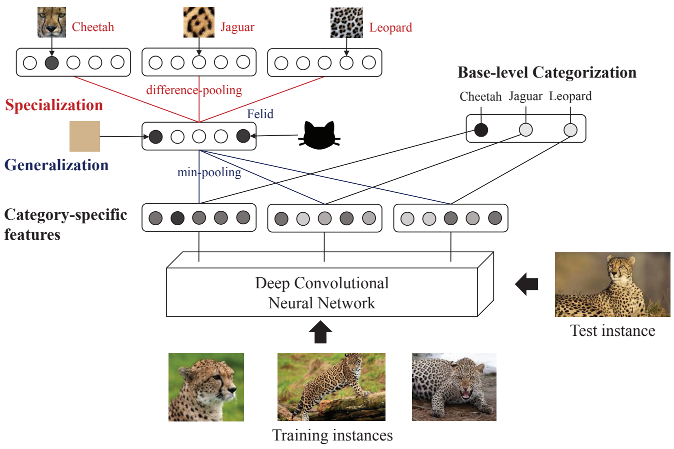

Fig. 1: **Concept:** Our taxonomy-regularized deep CNN learns grouped and discriminative features at multiple semantic levels, by introducing additional regularization layers that abstract and differentiate object categories based on a given class hierarchy. 1) At the generalization step, our network finds the commonalities between similar object categories that help recognize the supercategory, by finding the common components between per-category features. 2) At the specialization step, our network learns subcategory features as different as possible from the supercategory features, to discover unique features that help discriminate between sibling subcategories. These generalization and specialization layers work as regularizers that help the original network learn the features focusing on those commonalities and differences.

we tell them apart? We first notice that all these three animals have distinctive feline features, and have spots (*i.e.* discovery of commonalities). Then, since those common properties are no longer useful to discriminate between the animals, we start focusing on the properties that are specific to each animal, which are disjoint from the common properties that are shared among all the three animals. For example, we notice that they have different shapes of spots, and the leftmost animal has a distinctive tear mark. This fine-grained discrimination is not directly achieved by the low-level visual processing, and requires deliberate observations and reasoning.

How can we then construct a CNN such that it can mimic such high-level human reasoning process? Our idea is to implement the generalization/specialization process as additional regularization layers of the CNN, leveraging the class structure provided by a given taxonomy. Specifically, we add in multiple superclass layers on top of the CNN, which are implemented as channel-wise pooling layers that focus on the components shared by multiple sub-categories, which we refer to as *min-pooling*. After this generalization process, our network performs specialization for each subcategory through *difference-pooling* between it and its superclass. It enforces the network to learn unique discriminative features for each object category (See Figure 1).

These two pooling layers can be readily integrated into any conventional CNN models, to function as regularizers. We validate our taxonomy-regularized CNN on multiple datasets, including CIFAR-100 [11], Places-205 [25], and ImageNet Animal datasets [22], and obtain significant performance gain over the base CNN models such as AlexNet [12] and NIN [14].

Our contributions are threefold:

1. We show that exploiting grouped and discriminative information in a semantic taxonomy helps learn better features for CNN models.
2. We propose novel generalization and specialization layers implemented with min- and difference-pooling, which can be seamlessly integrated into any conventional CNN models.
3. We perform extensive quantitative and qualitative evaluation of our method, and show that the proposed regularization layers achieve significant classification improvement on multiple benchmark datasets such as CIFAR-100 [11], Places-205 [25], and ImageNet Animal datasets [22].

2 Related Work

Using semantic taxonomies for object categorization. Semantic taxonomies have been extensively explored for object categorization. Most existing work [1, 5, 15] exploits the tree structure for efficient branch-and-bound training and prediction, while a few use taxonomies as sources of relational knowledge between categories [24, 6, 9, 2]. Our method shares the same goal with the latter group of work, and especially focuses on the parent-child and sibling-sibling relations.

Deep convolutional neural networks. Deep CNNs [13] have recently gained enormous popularity for their impressive performance on a number of visual recognition tasks. Since Krizhevsky *et al.* [12] won the ImageNet ILSVRC challenge 2012 [3], many variants of this model have been proposed. GoogLeNet [21], and VGGNet [18] focus on increasing the network depth by adding more convolutional layers to the original model. Lin et al. [14], propose a model structure called *Network In Network* (NIN) to train non-linear filters with micro neural networks in convolutional layers, and replace the fully connected layers by global average pooling on per-category feature maps. Our model benefits from these recent advances in deep CNNs, as it can leverage any one of these deep networks as the base model.

Some existing work has explored the tree structure among tasks to regularize the learning of deep neural networks. Salakhutdinov *et al.* [17] propose to learn hierarchical Dirichlet process prior over the top-level features of a deep Boltzmann machine, which enables the model to generalize well even with few training examples. Srivastava and Salakhutdinov [19] further extend this idea to the discriminatively learned CNN, with both predefined and automatically-constructed tree hierarchies using Chinese Restaurant Process. However, these models only work as priors and do not exploit discriminative information in a class hierarchy that our model aims to learn. Recently, Yan *et al.* [23] propose a two-staged CNN architecture named HD-CNN, which leverages the taxonomy to separate the categories into easy coarse-grained ones and confusing fine-grained ones, trained in separate networks. However, such separate learning of coarse and fine grained categories results in larger memory footprints, while our model seamlessly integrates the two with minimal increase in memory usage. The main novelties of our approach in this line of research are twofold. First, we propose a generic regularization layers that can be merged into any types of CNNs. Second and more importantly, we exploit the tree structure to learn discriminative properties not only between supercategories, but also between sibling subcategories belonging to the same parents.

Discriminative feature learning by promoting competition. Some recent work in multitask learning focuses on promoting competitions among tasks to learn discriminative features per each task. Zhou *et al.* [27] introduce an exclusive lasso that regularizes the original least square objective with a ℓ_2-norm over ℓ_1-norm on parameters, which encourages competition for the features among different tasks. The orthogonal transfer proposed in [26] leverages the intuition that the classifiers in parent and child nodes of a taxonomy should be different, by minimizing the inner product of the parameters of a parent and a child classifier. A similar idea is explored in [7] in the context of metric learning, but the approach of [7] selects disjoint features instead of simply making the parameters to be different. This idea is further extended to the case of multiple taxonomies [8], where each taxonomy captures different sets of semantically discriminative features, which are combined in the multiple kernel learning framework. A recent work [9] also proposes a similar constraint, to relate the category embeddings learned for both the parent and child, and the sibling classes. Our idea shares a similar goal for learning unique and discriminative features for each class, but it is implemented with a much simpler means of pooling, which fits well into the CNN framework unlike all the other previous frameworks.

3 Architecture

Our goal is to exploit the class hierarchy information to learn grouped and discriminative features of categories in a deep convolutional neural network (CNN). We tackle this problem by augmenting the base CNN architecture with two additional generalization and specialization layers, which regularize the learning of the network to focus on the general and specific visual properties between similar visual object classes.

Figure 2 illustrates the overview of our network. We assume that a taxonomy of object categories is given as side information, which is either human-defined or constructed from data, and the base network is able to generate a feature map (or a vector) for each category. We will further discuss the details of base models in Section 3.1. Then, leveraging the class structure in the given taxonomy, our model imposes additional layers on top of these per-category feature maps (or vectors), to regularize the learning of the original network.

The first set of layers are generalization layers, which have the same structure with the given hierarchy $\mathcal{T}$. They mimic the human generalization process that learns increasingly more general and abstract classes by identifying the commonalities among the classes. Specifically, our network learns the feature maps

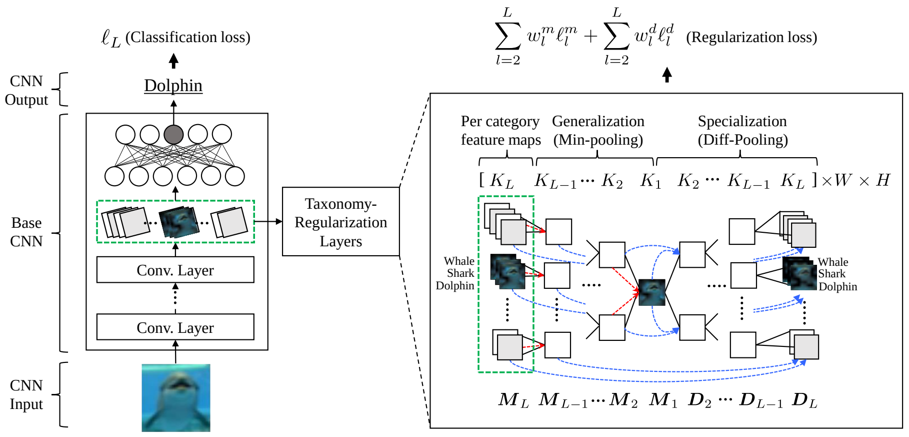
ℓ_L (Classification loss)
$\sum_{l=2}^{L} w_l^m \ell_l^m + \sum_{l=2}^{L} w_l^d \ell_l^d$ (Regularization loss)
CNN Output
Dolphin
Base CNN
Conv. Layer
Conv. Layer
CNN Input
Taxonomy-Regularization Layers
Per category feature maps
Generalization (Min-pooling)
Specialization (Diff-Pooling)
$[K_L \; K_{L-1} \cdots K_2 \; K_1 \; K_2 \cdots K_{L-1} \; K_L] \times W \times H$
Whale Shark Dolphin
Whale Shark Dolphin
$\boldsymbol{M}_L \; \boldsymbol{M}_{L-1} \cdots \boldsymbol{M}_2 \; \boldsymbol{M}_1 \; \boldsymbol{D}_2 \cdots \boldsymbol{D}_{L-1} \; \boldsymbol{D}_L$

Fig. 2: Overview of our taxonomy-regularized CNN. Our network computes per category feature maps (in green dotted boxes) from the base CNN, and feed them into the taxonomy-regularization layers. Then, the regularization sub-network that is organized by the structure of the given taxonomy first learns supercategory feature maps that capture shared features among the grouped classes through min-pooling (generalization), and then learn exclusive feature maps for each child class that are disjoint from its parent class through difference-pooling (specialization). Rectangles indicate feature maps; red and blue arrows denote min- and difference-pooling respectively.

of generalization layers by recursively applying the channel-wise *min-pooling* operation to grouped subclass feature maps guided by the taxonomy $\mathcal{T}$. Thus, each superclass feature map can identify the common activations among its child subclass feature maps. The generalization layers are learned to minimize the loss of superclass classification (*i.e.* classifying each superclass from all the other superclasses on the same level) (Section 3.2).

On top of the generalization layers are the specialization layers, which have the inverse structure of generalization layers (See Figure 2). The specialization layers uniquely identify each object class as a specialization of a more generic object class. These layers learn a unique feature map for each subclass that is not explained by the feature map of its parent through *difference-pooling*, which computes the difference between each subclass feature map and its parent feature map. The specialization layers are learned to minimize the loss of subclass classification (Section 3.3).

Throughout the paper, we use $l = 1, \cdots, L$ to denote the level of taxonomy hierarchy (L for the leaf and 1 for the root), K_l for the number of nodes at level l, n_l^k for the k-th node of the tree at level l, and c_l^k for children nodes of n_l^k.

3.1 Base Network Models

We can use any types of CNN models as our base network. However, instead of directly using it, we make a small modification to the last convolutional

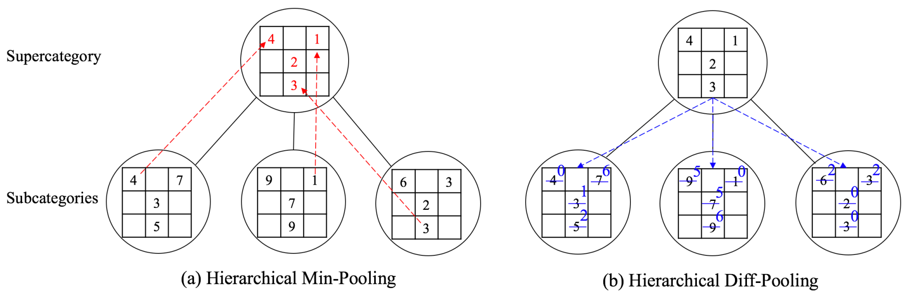

Fig. 3: Illustration of min- and difference-pooling. (a) The *min-pooling* operation computes the elementwise minimums across multiple subcategory feature maps. (b) The *difference-pooling* computes the elementwise differences between the feature maps of each subcategory and its parent.

layer, where we learn a per-class feature map $\boldsymbol{M}_L^k \in \mathbb{R}^{h \times w}$ for each class $k \in \{1, \cdots, C\}$, where h and w are the height and width of the feature map, and L in the subscript denotes that this feature map is for a base-level class. Note that the per-class features are not required to be two-dimensional maps, but can be one-dimensional vectors ($h = 1$). However, for generality we assume that the features are 2D maps, since this assumption is necessary for some CNN architectures (*e.g. Network in Network* (NIN) [14]). For architectures such as *AlexNet* that do not generate per-category feature maps, we can easily get such a network by simply adding convolutional layer having them to the last convolutional layer.

The feature map of each class is linked to the softmax loss layer through a *global average pooling* layer [14], which simply computes a single average value of all entries of an input feature map $\boldsymbol{M}$ (*i.e.* $\frac{1}{h \times w} \sum \boldsymbol{M}(i,j)$). The main role of the global average pooling is to learn the network such that each category-specific feature map produces high response for the input images of that category.

3.2 Min-Pooling for Superclasses

We learn the feature map for each superclass by exploiting the commonalities among the subclasses that belong to it(*e.g.* superclass *big cat* for subclasses *tiger*, *lion*, and *jaguar*). This is implemented by the min-pooling operation across subclass feature maps. The min-pooling simply computes the element-wise minimum over all input siblings' feature maps. Equation 1 and Figure 3.(a) describe the min-pooling:

$$\boldsymbol{M}_{l-1}^k(i,j) = \min\{\boldsymbol{M}_l^{k'}(i,j)\}_{k' \in \mathcal{C}_l^k} \tag{1}$$

where $\boldsymbol{M}_l^k(i,j)$ is the (i,j)-th element of feature map for class node n_l^k, and $\mathcal{C}_l^k$ is the set of its children.

This operation captures features that are common across all children subclasses, but not unique to any of them, which can be captured by difference-pooling in next section. For example, in Figure 1, using min-pooling on the feature maps captures only feline-features, such as the shape of the face and light brown color of their fur, but not their distinctive spots.

We attach the global-average pooling layer and then the softmax layer on top of the min-pooled superclass feature maps. Next we minimize the superclass loss, which learns the superclass feature maps to focus on the common properties of its children, which in turn propagate to lower layers.

Min- vs. max-pooling. Max-pooling is a more widely used downsampling method for object recognition using deep CNNs (*e.g.* [12]). However, it is not useful in our case where we aim to find general components across multiple object categories. For example, applying max-pooling on the category maps of the three animals in Figure 1, would capture unique spot patterns for each animal as well as the general feline features. This helps recognize the superclass *big cat* better, but does not help discriminate between the subclass categories, since the model does not know which are common and which are unique.

3.3 Difference-Pooling between Parent and Child Classes

Between each subcategory and its parent, we focus on finding the features that are as different as possible. Since the feature map of the parent captures the commonality between siblings, its activations may not be so useful for inter-subcategory discrimination. Thus we apply the difference-pooling between the response maps of the parent and its child subcategories. It retrieves the subcategory-specific entries that are not used in its parent response map.

Equation 2 and Figure 3.(b) describe the difference-pooling between a parent and its children. The feature map $\boldsymbol{D}_l^k(i,j)$ for node n_l^k of the difference-pooling layer between a parent and a child is defined as

$$\boldsymbol{D}_l^k(i,j) = \boldsymbol{M}_l^k(i,j) - \boldsymbol{M}_{l-1}^{k'}(i,j), \text{ s.t.} \quad k \in \mathcal{C}_{l-1}^{k'}. \tag{2}$$

The difference-pooling reduces the effect of supercategory-specific features, and thus makes the subcategory discrimination less dependent on supercategory-specific features. It in turn promotes learning the features that are required for fine-grained categorization at lower layers of the CNN.[1] As with the superclass feature maps, we attach the global average pooling and multinomial classification loss layers to the diff-pooling layers. This enforces the network to learn discriminative features that uniquely identifies each object category.

3.4 Unsupervised Construction of a Taxonomy

While the taxonomies for most generic object categories can be obtained from semantic taxonomies such as WordNet [16], such predefined taxonomies may

[1] We also test XOR-pooling that assign 0 to the elements of the children feature maps that are also selected at the parent feature map. However, in our experiments, the XOR-pooling results in a worse performance than diff-pooling, perhaps due to excessive sparsity.

be unavailable for domain-specific data. Furthermore, the semantic taxonomies do not always accurately reflect the feature distributions in the training set. Therefore, we propose a simple taxonomy construction method by examining the activations of the feature maps in the base network. Note that we do not declare this method as our major contribution, but as will be shown in experiments, it performs successfully when no taxonomy is available.

The key idea is to group the classes that have the similar activations of feature maps, because those are the confusing classes that we want to discriminate. First, we learn a base network using the original category labels, as done in normal image classification. We then define activation vector $\boldsymbol{g}^k$ for each category k by averaging the feature response maps for its training images. Next we perform agglomerative clustering on $\{\boldsymbol{g}^k\}_{k=1}^C$ using l_2 distance metric and Ward's linkage criteria. Once we obtain the dendrogram between categories, we can cluster them for any given K number of clusters, which become the superclasses and their members become subclasses. A single application of such agglomerative clustering can generate a two-level taxonomy. We can recursively apply this operation to obtain a multi-level taxonomy.

3.5 Training

We attach a softmax loss layer to every level of min- and diff- pooled layers via global average pooling layers (See Figure 2). With the superclass classification loss, the feature maps generated by min-pooling is learned to preserve spatially consistent information across subclasses that belong to the same superclass. That is, the activations on those feature maps are unique to the group of subclasses, and not possessed by other superclass groups. However, the superclass loss can at the same time hamper the network from learning the representation that discriminates between the subclasses that belong to the same group. Thus we add in an additional loss layer on top of the diff-pooling layer, which is the classification loss over the classes in same level.

The resulting network has multiple loss layers including the loss layers for the base-level classes, subclasses, and superclasses. However, since we are mostly interested in improving on the base-level categorization accuracy, we balance the contribution of each loss by giving different weights so that the additional losses for min-pooled and diff-pooled layers act as regularizers. The combined loss term is described as follows:

$$\ell = \ell_L + \sum_{l=2}^{L} w_l^m \ell_l^m + \sum_{l=2}^{L} w_l^d \ell_l^d. \tag{3}$$

where ℓ_L is the original base-level categorization loss, ℓ_l^m are the losses from min-pooled layers for superclasses, ℓ_l^d are the losses from diff-pooled layers for subclasses, and w_l^m, w_l^d are weights for each loss term.[2]

[2] Our experiments reveal that the network is not sensitive to these balancing parameters, as long as the base-level categorization loss has a higher weight than others. That is, $w_l^m, w_l^d < 1$.

We implement the min- and diff-pooling layers on top of the publicly available Caffe [10] package. Note that the added pooling layers do not introduce any new parameters and the only additional computational burden is on computing the min- and diff-pooling; thus the increase in memory and computational complexity is minor compared to the original model. The increase in space and time complexity depends on an employed tree structure, specifically on the number of internal nodes. If the number of classes at all levels is C and the memory usage of per class feature maps is U, then the increase in the space complexity will be $O(CU)$, where the worst case happens if the given tree is a full binary tree.

In our experiments, the increase in memory usage is less than 3% and the increase in training time is 15% at maximum, compared to those of the base networks. The HD-CNN [23], which is a similar approach that makes use of hierarchical class information, on the other hand, increases the memory usage and the training time by about 50% and 150% each. Thus, our model is more scalable with a much larger number of classes, with a large and complex class hierarchy. Also further speed-up can be achieved with a parallel implementation of the additional layers, although our current implementation does not fully exploit the parallelism on the problem.

4 Experiment

We evaluate the multiclass classification performance of our approach on multiple image datasets. Our main focus is to demonstrate that the taxonomy-based generalization and specialization layers improve the performance of base CNN models, by learning the discriminative features at multiple semantic granularity.

4.1 Dataset

CIFAR-100. The CIFAR-100 [11] consists of 100 generic object categories (*e.g. tiger*, *bed*, *palm*, and *bus*), and has been extensively used for the evaluation of deep neural networks.[3] It consists of 600 images per category (*i.e.* 60,000 images in total) with a size of 32×32, where 500 images are used for training and the remaining 100 images are for testing. We pre-process the images with global contrast normalization and ZCA whitening as done in [14, 23]. For taxonomy, we use the trees provided in [11], [19], and another one discovered using our tree construction method in Section 3.4.

Places-205. The Places dataset [25] contains $2,448,873$ images from 205 scene categories. The set of scene classes includes both indoor scenes (*e.g. romantic bedroom*, *stylish kitchen*) and outdoor scenes (*e.g. rocky coast*, and *wintering forest path*). We use the provided training and test splits by [25] for our experiments. For taxonomy, we use the discovered class hierarchy using our tree construction method since no predefined one exists for this dataset.

[3] `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html`.

Method	Top-1	Top-5
Baseline ConvNet of [19]	62.80	-
Tree-Based Prior [19]	63.15	-
Network in Network (NIN) [14]	65.03	-
dasNet [20]	66.22	-
HD-CNN [23]	67.38	-
NINtri	67.66	89.15
Ensemble of NINtri	70.58	90.67
(`Ours:NINtri+min-only`)	67.74	87.83
(`Ours:NINtri+Tree[11]`)	69.36	89.97
(`Ours:NINtri+TreeClust`)	68.82	89.34
(`Ours:NINtri+Tree[19]`)	68.32	89.44
(`Ours:NINtri+Ensemble`)	**71.81**	**91.46**

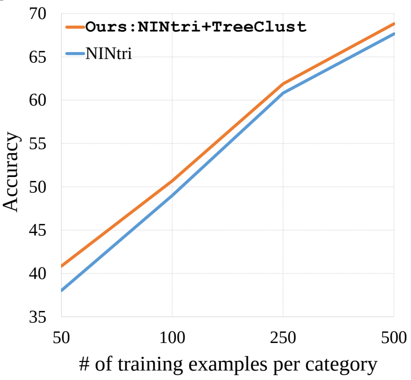

Fig. 4: **Left:** The classification results on the CIFAR-100 dataset. We report top-1 and top-5 accuracy in percentage. **Right:** The classification accuracy with different number of training images per label.

ImageNet Animal. ImageNet 1K/22K Animal datasets, suggested in [22], are subsets of the widely-used ImageNet dataset [3]. For ImageNet 1K Animal dataset, we select all 398 animal classes out of the ImageNet 1K and split the images into 501K training images and 18K test images. For ImageNet 22K Animal dataset, we collect 2,266 animal classes out of all ImageNet 22K classes; we only consider the classes that have more than 100 images and are at leaf nodes of ImageNet hierarchy. The dataset consists of 1.6M training images and 282K test images. Our ImageNet 22K Animal dataset has slightly different number of classes from [22] (2,282 classes), but the difference is less than 1%. For taxonomy, we use the generated class hierarchy instead of the existing ImageNet hierarchy since the ImageNet class hierarchy is largely imbalanced and overly deep.

As for tree depth in automatic hierarchy construction, we experimentally found that the optimal value is around $\log_{10} k$, where k is the number of classes; we used 2-level trees for CIFAR-100 and Places-205, and a 3-level tree for ImageNet 22K-Animals that comes with 2K classes.

4.2 Quantitative Evaluation

Results on CIFAR-100. Figure 4 shows the classification results of our method and the baselines on the CIFAR-100 dataset. As our base model, we use the Network-in-Network-triple denoted by (NINtri), which is the same as the original Network-in-Network model in [14], except that it has three times of the number of filters in the original network. This network is also used as a baseline in [23], in which the HD-CNN results in a lower accuracy than the NIN-triple, perhaps due to the difference in the number of learning parameters. We report the performance of our NIN-triple model regularized with different taxonomies. We use three different class hierarchies from [11], [19], and our tree construction method (denoted by a suffix `+TreeClust`). The performance varies depending on which taxonomy we use, but all of our models outperform the base NIN-triple.

Table 1: Classification results on the Places-205 dataset.

Method	Top-1	Top-5	Method	Top-1	Top-5
Places-AlexNet [25]	50.04	81.10	Places-NIN	43.46	75.00
(`Ours:AlexNet+TreeClust`)	**51.14**	**81.85**	(`Ours:NIN+TreeClust`)	**45.78**	**76.78**

Table 2: Classification results on the ImageNet 1K/22K animal dataset.

Dataset / Method	Xiao *et al.* [22]	AlexNet-pretrained	(`Ours`)
Imagenet 1K Animal	63.2	66.46	**67.53**
Imagenet 22K Animal	51.48	50.82	**51.91**

We obtain the best result using the class hierarchy in [11], which outperforms NIN-triple by 1.7%p. This increase is larger than 0.35%p reported in [19], and we attribute such larger enhancement to the exploitation of discriminative information from the taxonomy, through the proposed two pooling methods.

The performance can be further improved by ensemble learning with multiple taxonomies. We obtain a bagging predictor by simply averaging out the predictions of the models with the three taxonomies, and this ensemble model denoted by (`Ours:NINtri+Ensemble`) achieves 71.81% of accuracy, which is significantly higher than the base network NIN-triple by 4.15%p.

Results on Places-205. Table 1 shows the classification results on the Places dataset. As base networks, we test the NIN [14] and the AlexNet [12] trained on ILSVRC2012 dataset. Since no pre-trained model is publicly available for the Places dataset, we fine-tune those base models on the Places dataset, which we report as Places-Alexnet and Places-NIN. We generate the tree hierarchy using the method in Section 3.4, and then fine-tune each base network with the proposed generalization and specialization layers. Our tree-regularized networks outperform the base networks by 1.10%p (`Ours:Alexnet`) and 2.32%p (`Ours:NIN`), which are significant improvements.

Results on ImageNet Animal. Table 2 shows the classification results on ImageNet Animal datasets. We first pretrain the AlexNet on the ImageNet 1K/22K Animal datasets, reported as AlexNet-pretrained. From the learned base model, we generate the class hierarchy using our tree construction method. We then learn our tree-regularized network by fine-tuning the pretrained base AlexNet with the hierarchy. The resulting network outperforms [22] by 4.33%p in ImageNet 1K Animal, and by 0.43%p in ImageNet 22K Animal dataset. Also our network increases the performance of the base AlexNet model more than 1% in the both datasets.

On all datasets, our method achieves larger improvements in top-1 accuracy rather than in top-5 accuracy, which suggest that the key improvement come from the correct category recognition at the fine-grained level. This may be due to our model's ability to learn features that are useful for fine-grained discrimination from class hierarchy, through min- and diff-pooling.

Table 3: Hierarchical precision@k results on the CIFAR-100 dataset.

Method	hp@1	hp@2	hp@5
Network in Network-triple (NINtri) [23]	67.66	56.64	71.31
Ensemble of NINtri [23]	70.58	55.06	64.82
(`Ours:NINtri+min-only`)	67.74	60.34	79.19
(`Ours:NINtri+Tree[11]`)	69.36	**62.33**	**79.71**
(`Ours:NINtri+TreeClust`)	68.82	59.15	78.18
(`Ours:NINtri+Tree[19]`)	68.32	61.05	77.99
(`Ours:NINtri+Ensemble`)	**71.81**	57.85	67.44

Accuracy as a function of training examples. One can expect that our taxonomy-based regularization might be more effective with fewer training examples; to validate this point, we experiment with different number of training examples per class on CIFAR-100 dataset. We learn our model using 50, 100, 250, and 500 training examples, and plot the accuracy as a function of number of examples in Figure 4 (right). The plot shows that our model becomes increasingly more effective than the base network when using less number of training examples. The largest relative performance gain using our model occurs when using as few as 50 training examples, outperforming the baseline by around 3%p.

Semantic prediction performance using hp@k. To validate that our tree-regularized network can obtain semantically meaningful predictions, we also evaluate with the hierarchical precision@k (hp@k) introduced in [4], which is a measure of semantic relevance between the predicted label and the groundtruth label. It is computed as a fraction of the top-k predictions that are in the correct set, when considering the k nearest classes based on the tree distance. For detailed description of the hp@k measure, please refer to [4]. Table 3 shows the results on the CIFAR-100 dataset.

We observe that our taxonomy-regularized network obtains high hierarchical precisions, outperforming the base network by more than 7%p, using the semantic taxonomy from [11]. The improvement is less when using the constructed tree, but our network still outperforms the non-regularized base network. This performance gain in the semantic prediction mostly comes from the use of min-pooling, which groups the relevant classes together, with the diff-pooling also contributing to some degree with accurate discrimination of fine-grained categories. This point is clearly observed by comparing with the result of min-pooling only with the result of the full model (*i.e.* the third and fourth rows of Table 3).

4.3 Qualitative analysis

Figure 5 shows selected examples of class prediction made using our model and the baseline NIN [14] network on the CIFAR-100 and the Places dataset. We observe that in many cases, our network predicts more semantically relevant categories in the top-5 predictions (See Figure 5.(a-d)). This is even true for the failure case of Figure 5.(e), where the top-5 classes predicted by our model

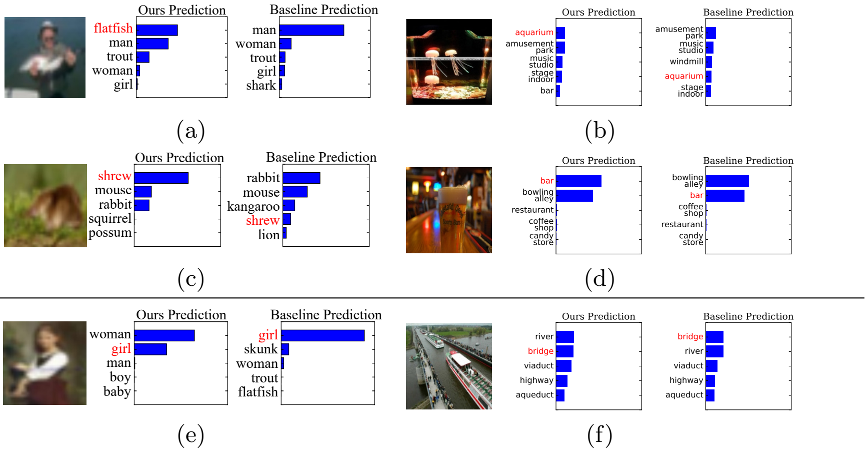

Fig. 5: Example predictions for the CIFAR-100 dataset (left column) and the Places-205 dataset (right column). For each image, we show the top-5 prediction result using our model and the base network (NIN). The last row shows failure cases.

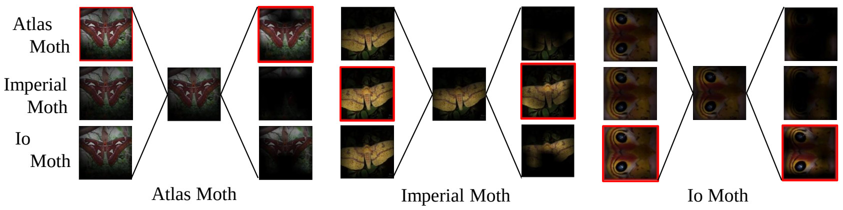

Fig. 6: Response maps for the subclasses that belong to the same superclasses in the ImageNet 22K Animal dataset for a given test instance. We superimpose the per-category, min-pooled, and diff-pooled response maps of these sibling classes on top of each input image. The correct feature map for each input image is highlighted in red.

(*i.e. woman*, *girl*, *man*, *boy*, and *baby*) are all semantically relevant to the correct class *girl*. On the other hand, the results of the base network include semantically irrelevant categories such as *skunk* and *flatfish* in the top-5 predictions. Also, our model is less likely to confuse between similar classes, while the base network is more prone to misclassification between them. Hence, the base network that often lists correct categories in the top-5 predictions still fails to predict the correct top-1 category (See Figure 5.(a-d)).

To further analyze where the improvement in classification performance comes from, we examine the response maps for the subcategories that share the same parent categories, in Figure 6. From ImageNet 22K animal dataset, we select one supercategory, *moth*, for which we select three subcategories. Note that the original feature maps do not represent discriminative activations, but diff-pooled

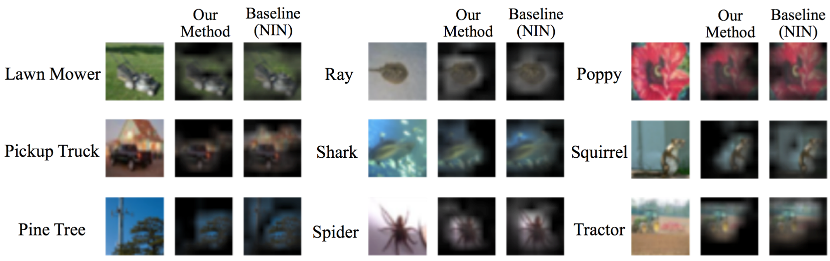

Fig. 7: Image segmentation results on the CIFAR-100 test images. Our network generates tighter segmentation boundaries compared to the base network.

activations clearly capture *discriminative* local properties for each subcategory. For example, in the *moth* subclasses, we can see that the diff-pooled activation maps exclusively focus on the wing patterns which are important for distinguishing different moth species. This example confirms that the hierarchical regularization layers can indeed capture discriminative traits for the subclasses, thus significantly eliminating confusions between subcategories.

We also compare between the feature maps learned by the base model NIN and our approach on the CIFAR-100 dataset, in Figure 7. Since the base network NIN has only convolution layers except for the last classification loss layer, the forward pass can preserve the spatial information. Therefore, we can segment an image by using activations of the feature map on the image. We show segmentation results in Figure 7. The segmentations by our method are qualitatively better, as they focus more on the target objects compared to the base network, which often generates loose and inaccurate segmentations. These results assure our taxonomy-based model's ability to learn unique features for each category, rather than learning features that are generic across all the categories.

5 Conclusion

We propose a regularization method that exploits hierarchical task relations to improve the categorization performance of deep convolutional neural networks. We focus on the task relations between the prediction of a parent and child classes in a taxonomy, and learn features common across semantically related classes through min-pooling, then learn discriminative feature maps for each object class by performing difference-pooling between the feature maps of each child class and its parent superclass. We validate our approach on the CIFAR-100, the Places-205, and the ImageNet Animal datasets, on which it achieves significant improvement over the baselines. We further show that our taxonomy-regularized network makes semantically meaningful predictions, and could be more useful when the training data is scarce.

Acknowledgement. This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1502-03.

References

1. Bengio, S., Weston, J., Grangier, D.: Label Embedding Trees for Large Multi-Class Task. In: NIPS (2010)
2. Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., Adam, H.: Large-scale object classification using label relation graphs. In: ECCV (2014)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009)
4. Frome, A., Corrado, G., Shlens, J., Bengio, S., Dean, J., Ranzato, M., Mikolov, T.: Devise: A deep visual-semantic embedding model. In: NIPS (2013)
5. Gao, T., Koller, D.: Discriminative Learning of Relaxed Hierarchy for Large-Scale Visual Recognition. In: ICCV (2011)
6. Hwang, S.J.: Discriminative Object Categorization with External Semantic Knowledge. Ph.D. Dissertation, The University of Texas at Austin (2013)
7. Hwang, S.J., Grauman, K., Sha, F.: Learning a Tree of Metrics with Disjoint Visual Features. In: NIPS (2011)
8. Hwang, S.J., Grauman, K., Sha, F.: Semantic Kernel Forests from Multiple Taxonomies. In: NIPS (2012)
9. Hwang, S.J., Sigal, L.: A Unified Semantic Embedding: Relating Taxonomies and Attributes. In: NIPS (2014)
10. Jia, Y.: Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding. In: arXiv:1408.5093 (2013)
11. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. Master's thesis, University of Toronto (2009)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: NIPS (2012)
13. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based Learning Applied to Document Recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
14. Lin, M., Chen, Q., Yan, S.: Network In Network. In: ICLR (2014)
15. Marszalek, M., Schmid, C.: Constructing Category Hierarchies for Visual Recognition. In: ECCV (2008)
16. Miller, G.A., Beckwith, R., Fellbaum, C.D., Gross, D., Miller, K.: WordNet: An Online Lexical Database. Int. J. Lexicograph 3(4), 235–244 (1990)
17. Salakhutdinov, R.R., Tenenbaum, J.B., Torralba, A.: Learning to Learn with Compound HD Models. In: NIPS (2011)
18. Simonyan, K., Zisserman, A.: Imagenet classification with deep convolutional neural networks. In: ICLR (2015)
19. Srivastava, N., Salakhutdinov, R.R.: Discriminative Transfer Learning with Tree-based Priors. In: NIPS (2013)
20. Stollenga, M.F., Masci, J., Gomez, F., Schmidhuber, J.: Deep Networks with Internal Selective Attention through Feedback Connections. In: NIPS (2014)
21. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going Deeper with Convolutions. In: CVPR (2015)
22. Xiao, T., Zhang, J., Yang, K., Peng, Y., Zhang, Z.: Error-driven Incremental Learning in Deep Convolutional Neural Network for Large-Scale Image Classification. In: ACM MM (2014)
23. Yan, Z., Zhang, H., Jagadeesh, V., DeCoste, D., Di, W., Yu, Y.: HD-CNN: Hierarchical Deep Convolutional Neural Network for Image Classification. In: ICCV (2015)

24. Zhao, B., Fei-Fei, L., Xing, E.P.: Large-Scale Category Structure Aware Image Categorization. In: NIPS (2011)
25. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning Deep Features for Scene Recognition Using Places Database. In: NIPS (2014)
26. Zhou, D., Xiao, L., , Wu, M.: Hierarchical Classification via Orthogonal Transfer. In: ICML (2011)
27. Zhou, Y., Jin, R., Hoi, S.C.H.: Exclusive Lasso for Multi-task Feature Selection. JMLR 9, 988–995 (2010)