From Categories to Subcategories: Large-scale Image Classification with Partial Class Label Refinement

Marko Ristin[1] Juergen Gall[2] Matthieu Guillaumin[1] Luc Van Gool[1,3]
[1]ETH Zurich [2]University of Bonn [3]KU Leuven

Abstract

The number of digital images is growing extremely rapidly, and so is the need for their classification. But, as more images of pre-defined categories become available, they also become more diverse and cover finer semantic differences. Ultimately, the categories themselves need to be divided into subcategories to account for that semantic refinement. Image classification in general has improved significantly over the last few years, but it still requires a massive amount of manually annotated data. Subdividing categories into subcategories multiples the number of labels, aggravating the annotation problem. Hence, we can expect the annotations to be refined only for a subset of the already labeled data, and exploit coarser labeled data to improve classification. In this work, we investigate how coarse category labels can be used to improve the classification of subcategories. To this end, we adopt the framework of Random Forests and propose a regularized objective function that takes into account relations between categories and subcategories. Compared to approaches that disregard the extra coarse labeled data, we achieve a relative improvement in subcategory classification accuracy of up to 22% in our large-scale image classification experiments.

1. Introduction

Over the last few years, we have witnessed an exponential growth of digital imaging. Possibilities for sharing visual content through social media, such as Facebook and Flickr, together with affordable high-quality cameras have made images ubiquitous in our lives. The semantic organization of images becomes therefore an imperative if one wants to access them effortlessly in a meaningful way. With such a growth, manual categorization is excessively tedious and expensive. As computers became more powerful and better algorithms were developed, automatic image classification has become accurate on smaller datasets with hundreds of categories and thousands of images.

The research community has since then moved to more challenging, larger datasets, such as ImageNet [28], which contain thousands of categories and millions of images. In such datasets, categories are often organized in a hierarchy. The deeper one goes in the hierarchy, the finer the categories are and annotated training data becomes rare. In order to obtain training data for fine subcategories, a natural approach is to search for images of coarser categories and refine the labels. This can be very expensive, especially if the subcategories require expert knowledge [12] (*e.g.*, breeds of dogs, bird or flower species, etc.). In this work, we are interested in such a scenario where only a subset of the training data is annotated with fine subcategory labels while the rest has only coarse category labels (*cf*. Fig. 1).

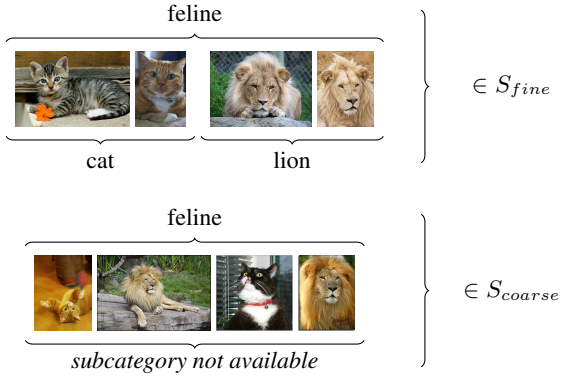


Figure 1: Given training data annotated with a set of categories like "feline", our goal is to refine the classification into subcategories like "cat" and "lion". We assume that the refined labels are available only for a subset of the training data S_{fine}, while for the rest, S_{coarse}, subcategory labels are not available.

In that scenario, we are particularly interested in investigating how the learning of subcategory classification can be improved by the simultaneous presence of training data annotated with coarser labels. To this end, we build on the framework of NCM forests [26], which are classification forests with splitting functions based on nearest class mean (NCM) classifiers [23]. NCM Forests are multiclass classifiers that can be efficiently trained and have shown to per-

form well for large-scale image classification [26]. Our first contribution is a principled approach to automatically learn optimal values for hyper-parameters of the NCM forest during training, namely the number of class means to use at each node. This is done by adding a regularization term to the objective function for training splitting functions. We present this improvement in Section 3. In our experiments, we show that our Regularized NCM forests outperform the NCM forests proposed in [26].

In Section 4, we then present our learning method in more detail. We show how existing state-of-the-art approaches like Stacking [27] can be adapted to our scenario and then describe our proposed model for sharing knowledge between coarse and fine categories. The key aspect of our approach is the introduction of an objective function that takes into account the hierarchical relations of categories and subcategories.

As we show in the experiments in Section 5, our proposed method achieves relative improvement of the classification accuracy at the subcategory level of up to 22% by training forests with an additional training set that only contains category labels. We outperform other methods that also train with this additional training set including the stacking approach of [27], and reach 99% of the performance achieved by a baseline trained with full supervision on the whole training set with refined labels. Finally, we show that an additional improvement can be obtained by combining the forests with the metric learning approach proposed in [23].

2. Related Work

Image classification on large-scale data sets has gained much attention in recent years [2, 13, 28]. While deep convolutional networks (CNN) achieve high classification accuracy [18], they are computationally intensive and take weeks to train. Simpler classifiers, such as nearest class mean classifiers (NCM) combined with a learned metric [23], proved to be a viable alternative with much shorter running times and near-zero costs for integrating new classes. In [26], NCMs were integrated in a random forest framework [9] which increased the performance and enabled fast incremental learning on a large scale. In this work, we address the problem of learning a classifier for the finest category level when only a part of the training data is annotated at that level, while the other training samples have only the labels of a coarser level. This is related to approaches for semi-supervised or transfer learning.

The transfer of knowledge in image classification reduces the amount of labeled data needed to learn a new image class by exploiting the existing labels of the other classes. A straightforward approach extends the training data with unlabeled samples, whose annotation is induced either based on neighborhood structure in the feature space [14] or semantic distance between the categories [15]. For approaches based on SVMs, samples from the semantically related classes can be used either as constraints [33] or regularizers [4] during learning. If the model is based on shapes, features [32] or parts [25] can be shared. Relations between classes can also be formulated as constraints for a neural network [13], or as part of a probabilistic framework [36], which can also be based on attribute similarities [19] or linguistic knowledge bases [27]. Class hierarchy can also be implictly learned to share model parameters between parent and children nodes [29] or to speed up the classification by stacking classifiers [22].

Relations between classes can also be used to build a classifier for various levels of a class hierarchy. Under the assumption that classes on the finer level are more difficult to classify than classes on a higher level, this results in a trade-off between accuracy and specificity. This trade-off was addressed in [11] where specificity was defined by the given class hierarchy, while the authors of [24] try to additionally learn specifity based on colloquial human responses. Image labels can also be leveraged together with appearance, location distribution and context to annotate a large data set with additional annotations such as bounding boxes [16] or segmentation masks [17].

Random forests [9], which have been used for many computer vision applications including image classification [6, 8, 26, 35], have also been used in the context of semi-supervised or transfer learning. The authors of [34] propose to uncover image segmentation masks using image labels as regularizers for both leaf statistics and tree structure. When data is labeled per bags of samples instead of individual instances, random forests have been adapted for multiple instance learning where the trees are trained using deterministic annealing [21]. Space-time consistency learned from weakly related videos is used in [20] to improve an object detector. Training data with accurate labels can be augmented with data with missing [10] or noisy labels [7]. This can be achieved by maximizing a weighted information gain that takes both sources of training data into account. In this work we do not consider additional data as noisy or weakly related, but exploit the relations between categories and subcategories to improve the classification accuracy of the subcategories.

3. Regularized NCM Forests

NCM forests have been shown to provide a good trade-off between training time and accuracy for large-scale image classification [26]. We first briefly discuss NCM forests in Section 3.1. In Section 3.2 we propose a novel objective function for training NCM forests, which takes into account the information gain and the computational cost of a splitting function to automatically set important parameters of NCM forests. In our experiments, we show that this modifi-

cation improves performance. In Section 4, we then proceed to show how their classification accuracy can be increased when only a fraction of the training labels are refined.

3.1. NCM Forests

NCM forests belong to the family of random forests [9], which combine classifiers consisting of T decision trees independently trained in a distributed fashion. Each tree t is trained recursively, starting from a root node, on labeled images S where each image is represented by a d-dimensional feature vector $\vec{x} \in \mathbb{R}^d$. The incoming data S^n at node n is split into two disjoint sets by a splitting function $f^n : \mathbb{R}^d \mapsto \{0,1\}$. The two sets, $S^n_{f=0}$ and $S^n_{f=1}$, are then passed to the left and right child of the node, respectively, and the training continues recursively for each child. In NCM forests, the splitting functions are based on nearest class mean classifiers (NCM) [23]. This consists in computing the means $\{c^n_\kappa\}$ of a subset of the classes $\kappa \in \mathcal{K}^n \subset \mathcal{K}$ observed in the data $\{S^n_\kappa\}$ reaching node n:

$$c^n_\kappa = \frac{1}{|S^n_\kappa|} \sum_{i \in S^n_\kappa} \vec{x}_i. \tag{1}$$

Each class mean c^n_κ is assigned to one of the two children nodes by means of a binary value $e_\kappa \in \{0,1\}$. Hence, the splitting function is defined by:

$$f^n(\vec{x}) = e_{\kappa^*(\vec{x})} \text{ , where } \kappa^*(\vec{x}) = \underset{\kappa \in \mathcal{K}^n}{\operatorname{argmin}} \|\vec{x} - c^n_\kappa\| . \tag{2}$$

In other words, a data point closest to the mean of class κ is passed to the left or right child based on e_κ.

For training, a pool of splitting functions $\mathcal{F}^n$ is generated by randomly sampling the subset of classes $\mathcal{K}^n$ and the binary values $\{e_\kappa\}_{\kappa \in \mathcal{K}^n}$. The splitting function $f^n \in \mathcal{F}^n$ that maximizes the information gain U is then selected and stored at the node n:

$$\begin{aligned} U(f) &= H\left(S^n\right) - \sum_{i \in \{0,1\}} \frac{|S^n_{f=i}|}{|S^n|} H(S^n_{f=i}) \\ H(S^n) &= -\sum_{\kappa \in \mathcal{K}} P(\kappa|S^n) \ln P(\kappa|S^n) \\ f^n &= \underset{f \in \mathcal{F}^n}{\operatorname{argmax}} U(f) , \end{aligned} \tag{3}$$

where H is the class entropy and $P(\kappa|S^n)$ the empirical probability of the class κ within S^n. The training ends when, for the selected splitting function, $|S^n_{f^n=0}| \leq \mu$ or $|S^n_{f^n=1}| \leq \mu$. The node n is then converted into a leaf l and the observed class distribution $P^l(\kappa)$ is stored. Notably, $P^l(\kappa)$ is computed based on at least μ samples.

For classification, an image $\vec{x}$ is passed through each tree t until it reaches a leaf $l^t(\vec{x})$. The individual tree responses are then averaged and the most probable class $\kappa^*(\vec{x})$ is returned as an output:

$$\kappa^*(\vec{x}) = \underset{\kappa}{\operatorname{argmax}} \frac{1}{T} \sum_t P^{l^t(\vec{x})}(\kappa) . \tag{4}$$

3.2. Regularized NCM Forests

As shown in [26], the training and testing times of an NCM forest greatly depend on the number of class means $|\mathcal{K}^n|$. The larger $|\mathcal{K}^n|$, the more comparisons between an image feature $\vec{x}$ and the class means $\{c^n_\kappa\}_{\kappa \in \mathcal{K}^n}$ need to be computed. In [26] it was reported that $|\mathcal{K}^n| = \sqrt{|\mathcal{K}|}$ results in a good trade-off between efficiency and accuracy. But fixing $|\mathcal{K}^n| = \sqrt{|\mathcal{K}|}$ for all nodes of the tree might not be optimal. Indeed, one expects that the optimal number of class means actually depends on the training data arriving at a specific node. We therefore propose to sample splitting functions with variable size of $|\mathcal{K}^n|$ and use an objective function that favors information gain and penalizes large computational cost. To do so, we add a sparsity term in (3):

$$U^\star(f^n) = U(f^n) - \lambda_{reg}|\mathcal{K}^n|, \tag{5}$$

where λ_{reg} is a weighting parameter.

In practice, we sample $\mathcal{K}^n$ such that $2 \leq |\mathcal{K}^n| \leq c\sqrt{|\mathcal{K}|}$. Notably, if $c = 1$, we use $\sqrt{|\mathcal{K}|}$ as upper bound for the number of class means $|\mathcal{K}^n|$ of a splitting function f^n. Mind that λ_{reg} fine-tunes the selection of split functions at the nodes, while the parameter c only impacts the sampling process.

We denote an NCM forest trained with (5) by *Regularized NCM Forest* (RNCMF). In Section 5, we show that the regularized NCMFs outperform the NCMFs [26].

4. From Categories to Subcategories

As already explained, we consider a scenario in which we are provided two disjoint sets of training data S. The first set, S_{coarse}, is annotated with labels of *coarse* classes $\mathcal{K}_{\text{coarse}}$ (*e.g.* "canine", "feline"), while the second set S_{fine} comes also with labels of subcategories, *i.e. fine* classes $\mathcal{K}_{\text{fine}}$ that refine the coarse ones (*e.g.* "wolf", "lion"). The main goal of our system is to classify images into fine classes by exploiting all the available data. This scenario can be seen as a special case of weakly supervised learning at several levels. First, we can consider the coarse and fine classes as completely independent (Section 4.1 and 4.2). We can also view a data point labeled with a coarse category as an uncertain observation or a partially missing label (Section 4.3). Finally, since we know by design the relationship between classes, we can also see the coarse-to-fine link as encoding a constraint on the knowledge transfer (Section 4.4 and 4.5).

4.1. Stacked RNCMF

Among the methods for knowledge sharing considered in [27], the one based on stacking SVM classifiers achieved a remarkable trade-off between simplicity and performance. In such a setting, the outputs of a set of source classifiers are concatenated to the input features for learning the classifiers for the target classes. In our hierarchical scenario, we use coarse classes as source classifiers and the finer ones as target. The stacking strategy is also readily adaptable to RNCM forests. We train a first RNCM forest $\mathcal{T}_1$ on S_{coarse} to classify coarse classes. The samples of S_{fine} are then pushed through $\mathcal{T}_1$ and the resulting probabilities P over coarse classes are normalized ($P' = \alpha P$) and concatenated to the feature vector $\vec{x}$, yielding the new feature vector $\vec{x}_1 = (\vec{x}, P'(\kappa_1), \ldots, P'(\kappa_{|\mathcal{K}_{\text{coarse}}|}))$.

A second forest $\mathcal{T}_2$ is then trained on S_{fine} to distinguish fine classes using the augmented features $\vec{x}_1$. At test time, an image follows the same procedure. First, it is fed into $\mathcal{T}_1$ to obtain $\vec{x}_1$, which is then pushed through $\mathcal{T}_2$ to obtain the probabilities over the fine classes. Stacking has the disadvantage that two forests need to be trained and evaluated for classification, and that its performance depends heavily on the quality of the source classifiers. Instead, we propose in Section 4.4 a novel training approach that learns a single RNCM forest with hierarchical class information.

4.2. Joint RNCMF

Similar to [10], we consider the case of a flat set of labels, where we simply try to learn a RNCMF to perform two independent classification tasks jointly: the first task is to separate the coarse classes, and the second one is to separate the fine classes. We thus denote such a forest as a Joint RNCMF, or J-RNCMF. In order to train a J-RNCMF on S_{coarse} and S_{fine}, the regularized information gain (5) can be computed as a sum of both gains, each one operating on its own level of classes:

$$U^\star(f^n) = U_{\text{fine}}(f^n) + \lambda U_{\text{coarse}}(f^n) + \lambda_{reg}|\mathcal{K}^n|, \tag{6}$$

$$U_{set}(f^n) = H_{set}\left(S^n_{set}\right) - \sum_{i\in\{0,1\}} \frac{|S^n_{f=i,set}|}{|S^n_{set}|} H_{set}(S^n_{f=i,set}),$$

$$H_{set}\left(S\right) = -\sum_{\kappa\in\mathcal{K}_{set}} P\left(\kappa|S\right)\ln P\left(\kappa|S\right),$$

where $S^n_{set} = S^n \cap S_{set}$ and $set \in \{\text{coarse}, \text{fine}\}$.

In other words, we consider the classes $\mathcal{K}_{\text{fine}}$ and $\mathcal{K}_{\text{coarse}}$ to be unrelated, and merely prefer splitting functions that classify well both sets of classes in parallel. Since samples of $\mathcal{K}_{\text{fine}}$ and $\mathcal{K}_{\text{coarse}}$ alike are present at a node, the class means can be computed over both sets of classes. A richer pool of class means improves the performance, as we will see in Section 5, and allows us to integrate the additional knowledge of coarse classes in a straightforward manner.

4.3. NN-RNCMF

Similar to [14], the label of each sample in S_{coarse} can be refined based on a simple classifier, such as nearest neighbors (NN), trained on S_{fine}. Although this approach is as generic as Stacking, it can be easily adapted to exploit our hierarchical setting. This is done by constraining the nearest neighbour search for a sample in $\mathcal{S}_{\text{coarse}}$ only to samples of its corresponding subcategories in $\mathcal{S}_{\text{fine}}$. The coarse labels of S_{coarse} are then replaced by the refined ones and an RNCMF can be trained on $S_{\text{coarse, refined}} \cup S_{\text{fine}}$ using only the labels of the finer categories. We refer to this approach as NN-RNCMF.

4.4. Hierarchical RNCMF

The Joint RNCMF approach described above ignored the relations between categories $\mathcal{K}_{\text{coarse}}$ and subcategories $\mathcal{K}_{\text{fine}}$, by treating the levels of the hierarchy as two independent classification tasks. As a matter of fact, we can easily add a hierarchical flavor to J-RNCMF. We derive for each sample in S_{fine} a category label based on its subcategory according to the hierarchy. To use this additional information, only the term U_{coarse} in (6) needs to be rewritten:

$$U_{\text{coarse}}(f^n) = H_{\text{coarse}}\left(S^n\right) - \sum_{i\in\{0,1\}} \frac{|S^n_{f=i}|}{|S^n|} H_{\text{coarse}}(S^n_{f=i}), \tag{7}$$

i.e. U_{coarse} is not computed only over the data $S^n \cap S_{\text{coarse}}$ with coarse-only labels, but over the entire S^n.

In this way, a classification error of a subcategory in S_{fine} is less penalized if the wrongly predicted subcategory belongs to the same category as the true subcategory, or, in other words, more penalized if the labels also disagree at the coarser level. We refer to this model as Hierarchical RNCMF, or H-RNCMF.

4.5. Our Full Model: NN-H-RNCMF

NN-RNCMF and H-RNCMF are complementary and can be combined. Indeed, while NN-RNCMF provides coarsely labeled data with fine labels, H-RNCMF essentially does the opposite and exploits the derived coarse labels of the finely labeled data. This leads to our full model, which we refer to as NN-H-RNCMF. First, the training samples of S_{coarse} are assigned the label of the closest neighbor in S_{fine} as in NN-RNCMF. Thus, each sample in S is now assigned a class label $\kappa_{\text{coarse}} \in \mathcal{K}_{\text{coarse}}$ as well as $\kappa_{\text{fine}} \in \mathcal{K}_{\text{fine}}$ which refines κ_{coarse}. For learning such a NN-H-RNCMF, (6) is modified by using U_{coarse} as in (7) and computing U_{fine} over S^n using the estimated fine labels for S_{coarse}:

$$U_{\text{fine}}(f^n) = H_{\text{fine}}\left(S^n\right) - \sum_{i\in\{0,1\}} \frac{|S^n_{f=i}|}{|S^n|} H_{\text{fine}}(S^n_{f=i}). \tag{8}$$

In this way, our model exploits both the class hierarchy and the observation that coarsely labeled data should match one of the finer subcategories.

5. Experiments

We perform our experiments on a subset of ILSVRC 2010 [28], a well-established and challenging dataset for large-scale image classification. The hierarchy of ILSVRC 2010 is given as a directed acyclic graph based on WordNet (each node represents a class, also called a *synset*). In this work, we focus on the classes that have a unique parent class. More precisely, we collected all leaf synsets of ILSVRC 2010 as our subcategories and their parents as categories, and class subtrees that overlap are ignored altogether. The remaining parent synsets form our coarse categories $\mathcal{K}_{\text{coarse}}$, while their corresponding children represent the subcategories $\mathcal{K}_{\text{fine}}$. We obtain thus $|\mathcal{K}_{\text{coarse}}|=143$ coarse classes and $|\mathcal{K}_{\text{fine}}|=387$ fine ones.

The original training, validation and test sets of ILSVRC 2010 are reduced to $\mathcal{K}_{\text{fine}}$. The reduced training set consists of 487K images where we have between 1.4K and 9.8K images for each coarse class and between 668 and 2.4K images for each fine class. There are 50 and 150 images per fine class in the validation and the test set, respectively. If not otherwise stated, the performance is measured as top-1 average accuracy over $\mathcal{K}_{\text{fine}}$.

Since in our scenario we assume that subcategory labels are only available for a subset of the training data, we randomly split the training data S in two disjoint sets for each fine class. While the first set S_{coarse} has only the category labels, the second set S_{fine} includes the labels of the subcategories as well. If not otherwise stated, $|S_{\text{fine}}| = |S_{\text{coarse}}| = 0.5|S|$. A detailed list of the classes is part of the supplementary material and the training, evaluation, and testing images are published on-line [1].

In our experiments, we use the publicly available 1k-dim. bag-of-visual-words (BoW) based on densely sampled SIFT features which are provided by [5]. The features are whitened based on the training data.

5.1. Regularized NCM Forest

We evaluate the influence of all the parameters on the validation set. If not stated otherwise, we follow [26] and enforce at least $\mu = 10$ samples at a leaf and restrict $|\mathcal{K}^n| \leq \sqrt{|K|}$. The forests consist of 50 trees. Besides accuracy, we also measure the test runtime by the average number of comparisons per tree and image as in [26].

First, we investigate the influence of parameters on our RNCMF which is trained on the full training set S to classify fine classes. The performance of RNCMF depends on the number of splitting functions $\mathcal{F}^n$ generated during training. To this end, we generate between 10 and 1000 class sets $\mathcal{K}^n$ and, for each set, we sample between 10 and 100 assignment configurations $\{e_\kappa\}_{\kappa\in\mathcal{K}^n}$. The results for both parameters are shown in Fig. 2a) and b). The default values in those experiments are 1000 sets, 50 assignments and $\lambda_{reg}=0.01$ (5). Fig. 2a) shows that the accuracy increases and the test runtime decreases if more sets $\mathcal{K}^n$ are sampled. The decrease of test runtime is caused by the regularizer. The number of random assignments $\{e_\kappa\}_{\kappa\in\mathcal{K}^n}$ has no significant impact. Fig. 2c) shows the impact of the sparseness term in (5). Large values for λ_{reg} reduce test runtime, but also classification accuracy. We fix $\lambda_{reg} = 0.001$ since it reduces test runtime without loss of accuracy.

	a) no metric	b) MET
NCM	7.24	15.71
k-NN	11.20	15.10
Multiclass SVM [3]	18.11	17.60
NCMF [26]	17.58	18.52
Our RNCMF	**19.01**	**19.79**

Table 1: Average accuracy on the test set for classifying fine classes $\mathcal{K}_{\text{fine}}$. The proposed regularized NCMFs outperform NCMFs and the other methods. Using metric learning (MET) as in [23] further improves the accuracy.

In Table 1, we compare RNCMF with NCMF [26] and other multiclass classifiers, namely NCM, k-NN, and multiclass SVM [3]. For all approaches, the parameters are optimized on the validation set. The evaluation is performed on the test set. The results show that the proposed RNCMF outperforms NCMF and the other methods. In addition, we train a distance metric MET as in [23] on the training set and apply it to the features, which reduces their dimensionality to 512 and additionally boosts performance, *cf*. Table 1b).

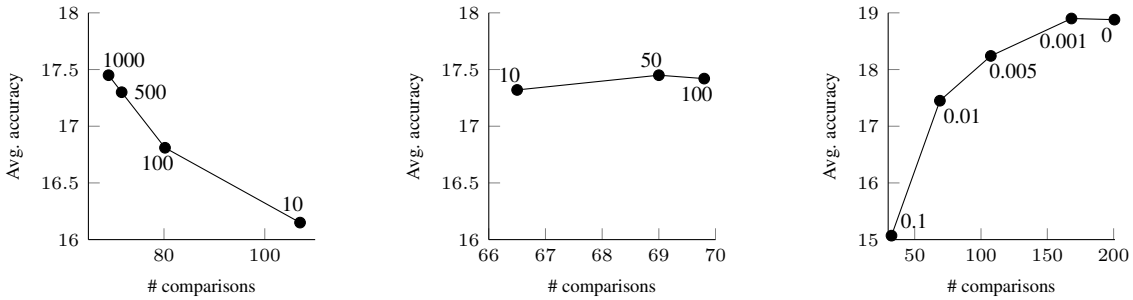
5.2. From Categories to Subcategories

In order to evaluate the performance of the methods discussed in Section 4, we use the training sets S_{fine} and S_{coarse}, and first evaluate the parameters of these methods on the validation set.

5.2.1 Parameters

Stacked RNCMF. Stacked RNCMF consists of two forests where the second forest is trained and applied on the image features and the probabilities over the categories $\mathcal{K}_{\text{coarse}}$ estimated by the first forest. Since $|\mathcal{K}_{\text{coarse}}|=143$, the features have 1143 dimensions in total for the second forest. The probabilities can be normalized to improve the performance. We evaluate the normalizing factors $10^{\{0,1,2,3\}}$ and $\alpha=10$ led to the best performance.

J-RNCMF and H-RNCMF. Both J-RNCMF and H-RNCMF depend on the weighting parameter λ, which steers the impact of U_{coarse} in (6). Fig. 3 shows the accuracy for



a) # sampled sets $\mathcal{K}^n$ b) # sampled $\{e_\kappa\}_{\kappa\in\mathcal{K}^n}$ per set c) regularizer weight λ_{reg}

Figure 2: The influence of **a)** the number of generated class mean subsets $\mathcal{K}^n$ (10, 100, 500, 1000), **b)** the number of generated assignments $\{e_\kappa\}_{\kappa\in\mathcal{K}^n}$ per set (10, 50, 100), and **c)** weighting parameter λ_{reg} (0.1, 0.01, 0.005, 0.001, 0) of the regularization term in (5) on accuracy and test time efficiency, measured by average number of comparisons per tree and image. While the number of generated assignments is not a very sensitive parameter, generating many subsets of class means and preferring smaller subsets through regularization improves both efficiency at test time and classification accuracy.

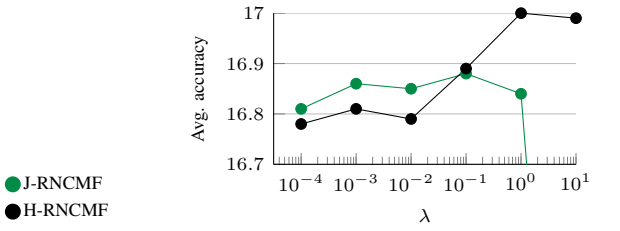


Figure 3: The impact of the weighting factor λ which steers the impact of U_{coarse}. H-RNCMF outperforms J-RNCMF as it also considers hierarchical relations between fine and coarse classes.

		a)	b)	c)
i)	NCM	7.14	14.49	16.30
	k-NN	9.45	12.33	12.58
	Multiclass SVM [3]	17.22	16.24	16.41
	RNCMF (baseline)	16.96	17.43	18.54
ii)	Stacked RNCMF	17.56	17.80	18.83
	J-RNCMF	17.14	17.44	18.39
	NN-RNCMF	17.93	18.26	19.29
	H-RNCMF	18.35	17.93	19.20
	NN-H-RNCMF	**18.46**	**18.48**	**19.55**

Table 2: Comparison of the accuracy when **a)** no metric, **b)** metric $\text{MET}_{\text{coarse}}$ learned on S_{coarse}, and **c)** metric MET_{fine} learned on S_{fine} is applied. Methods in **i)** are solely trained on S_{fine}, while methods in **ii)** are trained on S_{fine} and S_{coarse}. Our full model (NN-H-RNCMF) outperforms the baselines and other approaches and achieves accuracy close to using the entire data as reported in Table 1.

different values of λ. Since U_{coarse} differs for J-RNCMF and H-RNCMF, the optimal λ differs, too. While J-RNCMF achieves the best accuracy (16.86) for $\lambda = 0.1$, H-RNCMF achieves the best performance (17.00) with $\lambda = 1.0$ and outperforms J-RNCMF, which highlights the importance of class relations. For this experiment, we sample $\mathcal{K}^n$ only from $\mathcal{K}_{\text{fine}}$. However, when sampling $\mathcal{K}^n$ from $\mathcal{K}_{\text{coarse}} \cup \mathcal{K}_{\text{fine}}$ the average accuracy for H-RNCMF increases further to 18.10. In the following, we use $\lambda = 0.1$ for J-RNCMF, $\lambda = 1.0$ for H-RNCMF and NN-H-RNCMF, and sample $\mathcal{K}^n$ from $\mathcal{K}_{\text{coarse}} \cup \mathcal{K}_{\text{fine}}$.

NN-RNCMF. We can predict the missing fine labels of S_{coarse} in various ways. One possibility is to train an RNCMF on S_{fine} and use it to predict fine labels of S_{coarse}. This achieves an average accuracy of 14.02. We can also use the fine labels of the nearest neighbours from S_{fine}, ignoring the class relations, which achieves a better accuracy of 14.23. If we discard nearest neighbor matches where subcategory and category do not match, the accuracy improves to 17.03. The best accuracy (17.52) is achieved when the nearest neighbors are only searched among the samples of the corresponding subcategories. This shows that the relations between categories and subcategories are also important for the nearest neighbor approach. We use the third strategy in the following. Discarding nearest neighbor matches based on the distance in the feature space does not improve the performance any further.

Runtime. The training of the RNCM forests takes about 2 hours on a cluster with 800 cores. Finding the nearest neighbor of the samples from S_{coarse} in S_{fine} takes an additional hour. Classification of an image by a single tree takes about $50\mu s$ for 1K-dimensional pre-computed features.

RNCMF (baseline)	12.04 (1.00)	14.35 (1.00)	16.96 (1.00)
Stacked RNCMF	12.92 (1.07)	15.07 (1.05)	17.56 (1.04)
J-RNCMF	12.59 (1.05)	14.89 (1.04)	17.14 (1.01)
NN-RNCMF	13.99 (1.16)	15.48 (1.08)	17.93 (1.06)
H-RNCMF	14.16 (1.18)	16.15 (1.13)	18.35 (1.08)
NN-H-RNCMF	**14.71 (1.22)**	**16.37 (1.14)**	**18.46 (1.09)**
$\lvert S_{\text{fine}}\rvert$	a) $0.1\lvert S\rvert$	b) $0.2\lvert S\rvert$	c) $0.5\lvert S\rvert$

Table 3: Average accuracy of RNCMF as baseline trained on S_{fine} and the other methods trained on $S_{\text{fine}} \cup S_{\text{coarse}}$. The relative performance to the baseline is in the brackets. We fix $|S_{\text{coarse}}| = 0.5|S|$ and set $|S_{\text{fine}}|$ to **a)** $0.1|S|$, **b)** $0.2|S|$ and **c)** $0.5|S|$. Our approaches improve the baseline, even when $|S_{\text{fine}}|$ and $|S_{\text{coarse}}|$ are imbalanced. The improvement is even more pronounced, when there are few fine-labeled samples. Our NN-H-RNCMF outperforms other methods.

RNCMF (baseline)	16.96 (1.00)	16.96 (1.00)	16.96 (1.00)
Stacked RNCMF	17.18 (1.01)	17.21 (1.01)	17.56 (1.04)
J-RNCMF	16.89 (1.00)	16.80 (0.99)	17.14 (1.01)
NN-RNCMF	17.27 (1.02)	17.51 (1.03)	17.93 (1.06)
H-RNCMF	17.59 (1.04)	17.90 (1.06)	18.35 (1.08)
NN-H-RNCMF	**17.87 (1.05)**	**18.13 (1.07)**	**18.46 (1.09)**
$\lvert S_{\text{coarse}}\rvert$	a) $0.1\lvert S\rvert$	b) $0.2\lvert S\rvert$	c) $0.5\lvert S\rvert$

Table 4: Average accuracy of RNCMF as baseline trained on S_{fine} and the other methods trained on $S_{\text{fine}} \cup S_{\text{coarse}}$. The relative performance to the baseline is in the brackets. We fix $|S_{\text{fine}}| = 0.5|S|$ and set $|S_{\text{coarse}}|$ to **a)** $0.1|S|$, **b)** $0.2|S|$ and **c)** $0.5|S|$. The accuracy increases with the quantity of coarse-labeled data, and our NN-H-RNCMF improves most the performance of the baseline.

5.2.2 Comparison

We now evaluate the approaches presented in Section 4, which are trained on both S_{fine} and S_{coarse}, and compare them to an RNCMF that is trained only on S_{fine} to show the benefit of the additional training data S_{coarse}. The results are shown in Table 2. Since we have seen that metric learning improves the accuracy, we evaluate two cases. In the first case, the metric $\text{MET}_{\text{coarse}}$ is trained on S_{coarse} and in the second case MET_{fine} is trained on S_{fine}. Table 2 i) shows that $\text{MET}_{\text{coarse}}$ slightly improves the accuracy, but MET_{fine} improves it even more and is the best choice.

Table 2 ii) shows that the approaches ignoring class relations (Stacked, J) improve the (baseline) RNCMF trained on S_{fine} only slightly, while the approaches exploiting the hierarchy (NN, H, NN-H) outperform the baseline and the other methods.Our full model NN-H-RNCMF performs best with an accuracy of 19.55. Notably, this is almost matching the performance of RNCMF with full supervision, *i.e.*, trained on S with subcategory annotations for all training samples, *cf*. Table 1b), using 50% fewer fine labels.

5.2.3 Impact of Training Size

So far we have used $|S_{\text{coarse}}| = |S_{\text{fine}}| = 0.5|S|$, but obvisouly the relative sizes of those sets matter. Thus, we now evaluate the impact of the training size for S_{coarse} and S_{fine}. We first keep $|S_{\text{coarse}}|$ fixed, but vary $|S_{\text{fine}}| \in \{0.1|S|, 0.2|S|, 0.5|S|\}$ to observe how the methods cope with few fine-labeled data when $|S_{\text{fine}}|$ and $|S_{\text{coarse}}|$ are imbalanced. As baseline, we train RNCMF on S_{fine} and no metric is used. As Table 3 shows, the accuracy improves for all methods when $|S_{\text{fine}}|$ is increased. When there are few fine-labeled samples, the improvement of the baseline becomes larger. For $|S_{\text{fine}}| = 0.1|S|$, our full model NN-H-RNCMF improves the baseline from 12.04 to 14.71, a relative increase of 22%. This is relevant as fine labels are scarcer and more expensive to obtain than coarse ones.

In a second experiment, we fix $|S_{\text{fine}}| = 0.5|S|$ and vary $|S_{\text{coarse}}|$. The results are shown in Table 4. All methods benefit from a larger training set $|S_{\text{coarse}}|$ with coarse labels.

RNCMF (baseline)	24.54 (1.00)
Stacked RNCMF	22.35 (0.91)
J-RNCMF	21.99 (0.90)
NN-RNCMF	24.19 (0.99)
H-RNCMF	23.95 (0.98)
NN-H-RNCMF	**25.15 (1.02)**

Table 5: Comparison of average accuracy in classifying coarse classes $\mathcal{K}_{\text{coarse}}$. The baseline is trained on S for classification of $\mathcal{K}_{\text{coarse}}$, while other models are trained to classify fine classes $\mathcal{K}_{\text{fine}}$. The relative performance to the baseline is in the brackets. The experiment is performed on the test set with 150 images per coarse class. Metric learning is not used and we fix $|S_{\text{coarse}}| = |S_{\text{fine}}| = 0.5|S|$. Our NN-H-RNCMF performs best and even outperforms the baseline due to the additional knowledge of subcategories.

5.3. Classification of Categories

Although it is not our original goal, our classifiers for the subcategories can also be used to classify the categories. This can be seen as a way of exploiting additional information for learning category classifiers. To this end, we only have to convert the class probabilities stored at leaves from $\mathcal{K}_{\text{fine}}$ to $\mathcal{K}_{\text{coarse}}$. To obtain a suitable, balanced test set, we randomly subsample the original test set to contain 150 images per category and measure average accuracy on it. We set $|S_{\text{coarse}}| = |S_{\text{fine}}| = 0.5|S|$ and use no metric. Table 5 shows that the approaches which exploit the hierarchy (NN, H, NN-H) achieve the accuracy of the baseline and outperform other approaches which ignore hierarchical relations

A) $|\mathcal{K}^n| \leq \sqrt{|\mathcal{K}|}$
B) $|\mathcal{K}^n| \leq 5\sqrt{|\mathcal{K}|}$
C) $|\mathcal{K}^n| \leq 10\sqrt{|\mathcal{K}|}$
D) $|\mathcal{K}^n| \leq 20\sqrt{|\mathcal{K}|}$

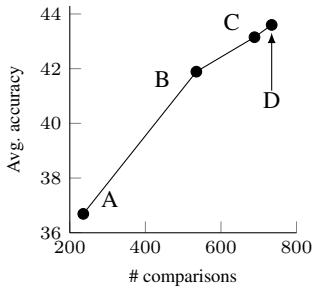


Figure 4: Impact of the upper bound $|\mathcal{K}^n| \leq c\sqrt{|\mathcal{K}|}$. Average accuracy and efficiency at test time measured by the number of comparisons (per tree and image) are reported. The experiments are performed on the validation set using Fisher vectors and a metric MET_{fine} learned on S_{fine}.

NCM [23]	30.22	35.27	42.34
RNCMF (baseline)	30.55	35.99	43.30
H-RNCMF	**31.42**	**36.42**	**43.62**
NN-RNCMF	28.93	32.75	42.26
NN-H-RNCMF	28.25	31.81	41.80
$\lvert S_{\text{fine}}\rvert$	a) $0.1\lvert S\rvert$	b) $0.2\lvert S\rvert$	a) $0.5\lvert S\rvert$

Table 6: Average accuracy of proposed methods and state-of-the-art NCM [23] evaluated on the test set and performed with Fisher vectors and metric learning on S_{fine}. Since metric learning on Fisher vectors [23] does not generalize well to nearest neighbors (NN) approaches. H-RNCMF outperforms NN-H-RNCMF, as well as the baseline and [23].

(Stacked, J). Remarkably, NN-H-RNCMF outperforms the baseline, showing that the additional knowledge of subcategories for 50% of the training data helps to improve the classification of the categories, and that our model makes good use of that extra information.

5.4. High-dimensional features

High-dimensional features like Fisher vectors have shown good performance in image classification [30]. Hence, we also evaluate our methods for classifying subcategories using the same Fisher vectors with metric learning as in [23] instead of BoW features. The metric MET_{fine} is trained with Fisher vectors on S_{fine}. In contrast to the BoW features, the accuracy of RNCMF can be further improved when the upper bound for the number of the class means sampled at each node $|\mathcal{K}^n| \leq c\sqrt{|\mathcal{K}|}$ is increased. We train the forests with $c \in \{1, 5, 10, 20\}$ on S_{fine}, with $|S_{\text{fine}}| = 0.5|S|$, and evaluate them on the validation set. The results in Fig. 4 show that increasing the upper bound on $|\mathcal{K}^n|$ improves the accuracy. In the following experiments, we use $c = 10$, which achieves near optimal performance with a gain in test time complexity.

Finally, we compare our approaches to the state-of-the-art NCM [23]. As in Section 5.1 and 5.2, we optimize the parameters for RNCMF, H-RNCMF, NN-RNCMF and NN-H-RNCMF on the validation set. The parameters do not change except that $\mathcal{K}^n$ is sampled 100 times per node instead of 1000 times and the weighting parameter λ (6), which steers the impact of U_{coarse}, increases to 100. For the evaluation on the test data, the amount of coarse-labeled data is fixed to $|S_{\text{coarse}}| = 0.5|S|$, while $|S_{\text{fine}}|$ varies. We applied the metric MET_{fine} trained on the corresponding S_{fine}. The results are shown in Table 6. In contrast to BoW features, nearest neighbours perform poorly and reduce the performance of NN-RNCMF and NN-H-RNCMF. This is consistent with [23] where it is shown that a metric learned on high-dimensional features does not generalize well to nearest neighbors approaches. For high-dimensional features, H-RNCMF therefore performs best.

We also performed preliminary experiments with 4096-dim. CNN features [31]. Since the features are pre-trained on the fine categories of ILSVRC 2012, the results are not directly comparable to the other experiments. RNCMF as in Table 1 achieves 74.18 accuracy and outperforms multiclass SVM [3] (71.67) and NCM (66.02). Using the protocol as in Table 6, NN-H-RNCMF achieves 69.95 ($0.1|S|$), 71.41 ($0.2|S|$) and 73.43 ($0.5|S|$) accuracy while the accuracy of RNCMF (baseline) is 68.49, 70.49 and 73.07, respectively.

6. Conclusion

In this paper, we have addressed the problem of learning subcategory classifiers when only a fraction of the training data is labeled with fine labels while the rest only has labels of coarser categories. To this end, we proposed to use random forests based on nearest class mean classifiers and extended the method by introducing a regularized objective function for training. We also experimentally showed that the additional training data with the category-only labels improves the classification of sub-categories up to 22% in a large-scale setting. Finally, we have presented experimental evidence that the approaches taking into account the hierarchical relations between categories and subcategories perform better than approaches ignoring these relations.

Acknowledgments. The authors acknowledge financial support from the CTI project (15769.1 PFES-ES), DFG Emmy Noether program (GA 1927/1-1), DFG project (GA 1927/2-2 FOR 1505) and Toyota.

References

[1] http://www.vision.ee.ethz.ch/datasets_extra/mristin/ristin_et_al_cvpr15_data.zip. [Online; accessed 19-Mar.-2015].

[2] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014.

[3] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. *TPAMI*, 2013.

[4] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *ICCV*, 2011.

[5] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010. http://www.image-net.org/challenges/LSVRC/2010, 2010. [Online; accessed 13-Nov.-2014].

[6] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and ferns. In *ICCV*, 2007.

[7] L. Bossard, M. Dantone, C. Leistner, C. Wengert, T. Quack, and L. Van Gool. Apparel classification with style. In *ACCV*, 2012.

[8] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101: Mining Discriminative Components with Random Forests. In *ECCV*, 2014.

[9] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[10] A. Criminisi and J. Shotton. Semi-supervised classification forests. In *Decision Forests for Computer Vision and Medical Image Analysis*, Advances in Computer Vision and Pattern Recognition. Springer, 2013.

[11] J. Deng, J. Krause, A. Berg, and L. Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *CVPR*, 2012.

[12] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *CVPR*, 2013.

[13] J. Deng, D. Nan, J. Yangqing, F. Andrea, M. Kevin, B. Samy, L. Yuan, N. Hartmut, and A. Hartwig. Large-scale object classification using label relation graphs. In *ECCV*, 2014.

[14] K. Driessens, P. Reutemann, B. Pfahringer, and C. Leschi. Using weighted nearest neighbor to benefit from unlabeled data. In *Advances in Knowledge Discovery and Data Mining*, volume 3918 of *Lecture Notes in Computer Science*. Springer, 2006.

[15] R. Fergus, H. Bernal, Y. Weiss, and A. Torralba. Semantic label sharing for learning with many categories. In *ECCV*, 2010.

[16] M. Guillaumin and V. Ferrari. Large-scale knowledge transfer for object localization in ImageNet. In *CVPR*, 2012.

[17] M. Guillaumin, D. Kuettel, and V. Ferrari. ImageNet Auto-annotation with Segmentation Propagation. *IJCV*, 2014.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[19] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

[20] C. Leistner, M. Godec, S. Schulter, A. Saffari, M. Werlberger, and H. Bischof. Improving classifiers with unlabeled weakly-related videos. In *CVPR*, 2011.

[21] C. Leistner, A. Saffari, and H. Bischof. Miforests: Multiple-instance learning with randomized trees. In *ECCV*, 2010.

[22] B. Liu, F. Sadeghi, M. Tappen, O. Shamir, and C. Liu. Probabilistic label trees for efficient large scale image classification. In *CVPR*, 2013.

[23] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *TPAMI*, 2013.

[24] V. Ordonez, J. Deng, Y. Choi, A. C. Berg, and T. L. Berg. From large scale image categorization to entry-level categories. 2013.

[25] P. Ott and M. Everingham. Shared parts for deformable part-based models. In *CVPR*, 2011.

[26] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool. Incremental learning of NCM forests for large-scale image classification. In *CVPR*, 2014.

[27] M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR*, 2011.

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. arXiv:1409.0575, 2014.

[29] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR*, 2011.

[30] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *IJCV*, 2013.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[32] M. Stark, M. Goesele, and B. Schiele. A shape-based object class model for knowledge transfer. In *ICCV*, 2009.

[33] T. Tommasi, F. Orabona, and B. Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *CVPR*, 2010.

[34] A. Vezhnevets and J. M. Buhmann. Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. In *CVPR*, 2010.

[35] B. Yao, A. Khosla, and L. Fei-fei. Combining randomization and discrimination for fine-grained image categorization. In *CVPR*, 2011.

[36] A. Zweig and D. Weinshall. Exploiting object hierarchy: Combining models from different category levels. In *ICCV*, 2007.