2017 14th Conference on Computer and Robot Vision

Weakly Supervised Image Classification with Coarse and Fine Labels

Jie Lei
University of Electronic Science and Technology of China
jayleicn@gmail.com

Zhenyu Guo
Sengled Canada
zhenyu.guo@sengled.com

Yang Wang
University of Manitoba
ywang@cs.umanitoba.ca

***Abstract*—We consider image classification in a weakly supervised scenario where the training data are annotated at different levels of abstractions. A subset of the training data are annotated with coarse labels (e.g. wolf, dog), while the rest of the training data are annotated with fine labels (e.g. breeds of wolves and dogs). Each coarse label corresponds to a superclass of several fine labels. Our goal is to learn a model that can classify a new image into one of the fine classes. We investigate how the coarsely labeled data can help improve the fine label classification. Since it is usually much easier to collect data with coarse labels than those with fine labels, the problem setup considered in this paper can benefit a wide range of real-world applications. We propose a model based on convolutional neural networks (CNNs) to address this problem. We demonstrate the effectiveness of the proposed model on several benchmark datasets. Our model significantly outperforms the naive approach that discards the extra coarsely labeled data.**

I. Introduction

Image classification is one of the most fundamental problems in computer vision. It serves as a building block for many other high level tasks in vision, such as object detection, scene understanding, etc. The performance of image classification systems has increased dramatically in the past few years. The current state-of-the-art image classification system has even surpassed human performance on the ImageNet challenge [1].

The availability of large-scale datasets has been one of the most important driving forces of this tremendous success. For example, the ImageNet [2] dataset contains thounsands of object categories and millions of images. ImageNet has enabled the resurgence of deep convolutional neural networks in computer vision [3]. In such datasets, the categories (called "synsets" in ImageNet) are often organized in a hierarchy. The categories at the top of the hierarchy correspond to coarse classes (e.g. wolf, dog). As one goes deeper in the hierarchy, the categories will correspond to finer classes (e.g. breeds of wolves or dogs).

In order to collect large-scale image datasets, the standard approach is to search images online and ask humans to label them. This can be very expensive and time-consuming, especially for fine-grained categorises. First of all, labeling fine-grained categories often requires expert knowledge (e.g. breeds of wolves or dogs). In addition, if a category is too fine-grained, it might be difficult to get access to enough images of this category. In contrast, it is relatively easy to collect and annotate images of coarse categories (e.g. wolf, dog), since there are more images available and the annotation can be done by non-experts.

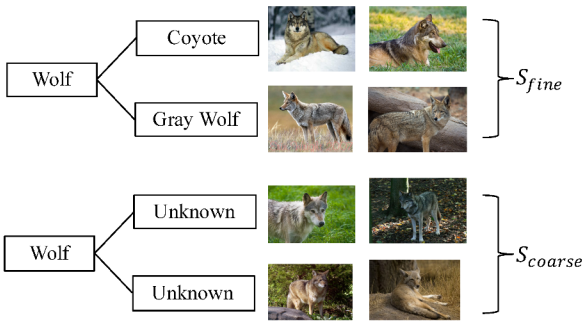

Figure 1: Illustration of the problem considered in this paper. We consider a scenario where the training data consist of two subsets S_{fine} and S_{coarse}. Each image in the first set S_{fine} is annotated with a fine label (e.g. coyote, gray wolf, etc.). We assume that each fine class belongs to exactly one coarse class (e.g. wolf). So we can equivalently think of each image in S_{fine} being annotated with a fine label and the corresponding coarse label (although the coarse label is redundant once we know the fine label). Each image in the second subset S_{coarse} is annotated with only a coarse label (e.g. wolf) and its fine label is unknown. Our goal is to use these two subsets to train a fine label classifier.

In this work, we consider a weakly supervised scenario for image classification where only a subset of training data are labeled with fine classes (e.g. breed of wolves or dogs) and the rest of the training data are labeled with coarse classes (e.g. wolf, dog). Each fine class corresponds to a subcategory of one of the coarse class. From such training data, we learn a system that can classify new images into one of the fine classes. See Fig. 1 for an illustration.

In this scenario, we are interested in learning fine-class classification models whose performance can be improved with the augmentation of additional training data annotated with coarse labels. Ristin *et al.* [4] has considered similar problem setting and proposed a learning method based on the NCM forest [5]. However, convolutional neural network is the current de facto standard in image classification.

978-1-5386-2818-8/17 $31.00 © 2017 IEEE
DOI 10.1109/CRV.2017.21

It is not clear how the method in [4] can be applied in combination with CNNs.

The main contribution of this paper is to develop a CNN-based approach for weakly supervised image classification. Similar to [4], our proposed method can take advantage of training data with coarse labels to improve the performance of fine-class classification. Since our proposed method is based on the powerful CNN framework, perhaps not surprisingly, it significantly outperforms the method in [4].

II. Related Work

In this section, we briefly review several lines of work most relevant to this paper.

Convolutional neural networks: Convolutional neural network (CNN) has shown tremendous success over the past few years. Since AlexNet [3] won the 2012 ImageNet challenge, CNN has quickly become the method of choice for image classification and many CNN variants have been proposed. For example, VGG Net [6] and GoogLeNet [7] add more layers to the original AlexNet to make the network deeper. Network-in-Network (NIN) [8] proposes to replace the linear filters in conventional convolutional networks with micro neural networks. ResNet [9] introduces shortcut connections in the network architecture to learn very deep network. Our proposed method benefits from these recent developments in CNNs. It can use any of these existing deep networks as its base model.

Semantic hierarchy in visual recognition: Our work is related to a line of research on using semantic relations of objects in visual recognition. For example, the object classes in ImageNet[2] are organized as a hierarchical taxonomy with different levels of abstraction. The taxonomy represents the "is-a" relationship of object classes, e.g. *coyote* is a sub-category of *wolf*. Bengio *et al.* [10] and Deng *et al.* [11] use this taxonomy for learning image classification systems that are sublinear in the number of classes. Deng *et al.* [12] use the hierarchy to learn a classifer that can select the appropriate level of abstraction, while trading off specificity for accuracy. Ordonez *et al.* [13] learn to predict *entry level categories* for images. Yan *et al.* [14] and Goo *et al.* [15] incorporate the taxonomy in CNNs to improve image classification performance.

Dealing with limited data: For many applications, it might be difficult to have access to large amount of training data. Our work falls under the general direction of how to deal with such scenario. One approach that has been proven effective is fine-tuning [16]. The idea is to use a model trained for a task where large-scale data are available (e.g. ImageNet classification), then fine-tune the model on a new task where only limited data are available. The work in [4] is the closest to ours. It considers the scenario where a subset of training data are annotated with fine labels, and the rest are annotated with only coarse labels. We consider the same problem setup in this paper.

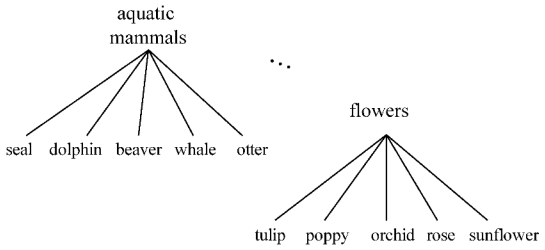

Figure 2: Example taxonomy from the CIFAR100 dataset [17].

III. Problem Setup

Our problem setup is similar to that in [4] and is illustrated in Fig. 1. We consider the scenario where the training data S consists of two disjoint sets S_{coarse} and S_{fine}. Images in S_{coarse} are annotated with their coarse classes (e.g. wolf) and those in S_{fine} are annotated with fine classes (e.g. coyote, gray wolf, etc.). We use N_f and N_c to denote the number of fine and coarse classes, respectively. We assume that there exists a one-to-many mapping between coarse and fine classes – each coarse class contains multiple fine classes, while each fine class belongs to exactly one coarse class. Fig. 2 shows an example of a taxonomy from the CIFAR100 dataset [17].

Our goal is to develop a system to classify images into the fine classes. Of course, a naive solution is to train a standard classification model only based on S_{fine}. However, this is suboptimal since it ignores S_{coarse}. In addition, S_{fine} is usually small in practice, since it is expensive to collect training images with fine class labels. In this paper, we investigate how to improve the performance of fine class prediction when there are additional coarsely labeled training data available. Our problem setup is related to weakly supervised learning. The subset S_{coarse} essentially provides a form of weak supervision for the learning problem. For training images in S_{coarse}, we only have access to their coarse labels without knowing the detailed fine labels, i.e. these images are weakly labeled.

In this paper, we focus on the scenario where the class labels have two levels of abstraction (i.e. coarse labels vs fine labels). But our proposed method can be easily extended to the case where the class labels are organized in a tree-structured taxonomy with multiple levels.

IV. Our Approach

Our goal is to design a model that can exploit the hierarchical relationship between coarse and fine labels in a deep convolutional neural network (CNN). We achieve this by modifying standard CNN architecture (e.g. AlexNet [3]) so that it can predict both coarse and fine labels for a given image. Fig. 3 illustrates the overall architecture of our model. The model will generate a feature map (or a feature vector) for each of the fine classes. Given the taxonomy information

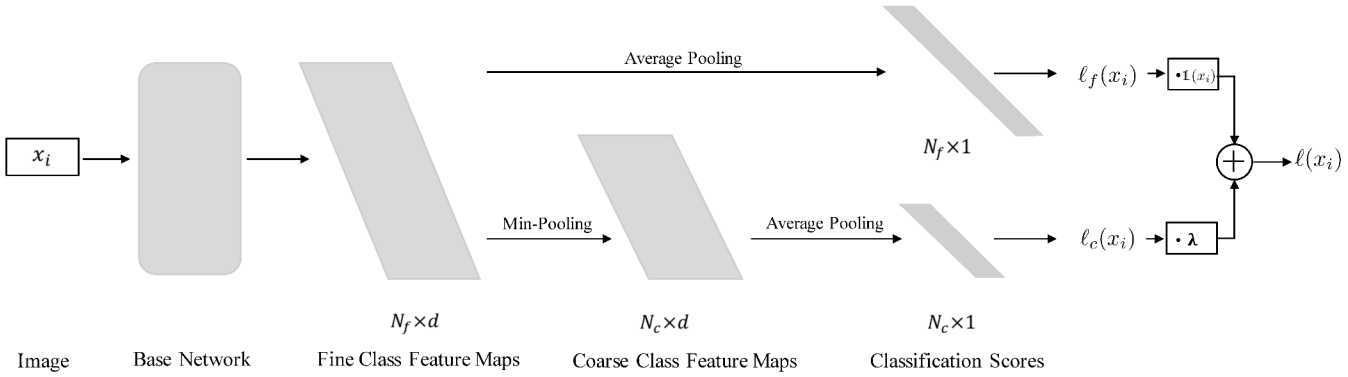

Figure 3: An overview of the architecture of our model. Given an image, our model first generates a feature map for each fine class. The average pooling is applied on these fine class feature maps to obtain the classification score of each fine label. From the fine class feature maps, we use min-pooling to get the coarse class feature maps. Again, the average pooling is applied to obtain the coarse label classification scores. These classification scores are then used to define the loss.

of the classes, the network is able to learn coarse class feature maps through min-pooling. A global average pooling layer is attached on the top of the class feature maps, which simply computes the average for every feature map, for both fine classes and coarse classes.

A. Base Model

We can use any CNN architecture as our base model. The only requirement is that the base model needs to generate a per-category feature map. Most of the popular CNN architectures only require minor modification to produce these per-category feature maps. For CNNs (e.g. Network-in-Network (NIN)) which directly generate a feature map for every class in the dataset, they can be directly adopted in our approach by removing the last softmax layer. For CNNs (e.g. AlexNet) which only produce a high dimensional vector, we can append multiple parallel fully connected layers to generate the class feature maps. Readers are referred to Sec. V for details about those modifications.

In the end, the base model produces a two-dimensional matrix $M_f \in \mathbb{R}^{d \times N_f}$, where each column $M_f^j \in \mathbb{R}^d$ corresponds to a class-wise feature map for each fine class $j \in \{1, 2, ..., N_f\}$. Without loss of generality, we have assumed that the class-wise feature map is a one-dimensional vector with length d. If the base model produces a two-dimensional map (e.g. in the case of NIN), we can always concatenate the values in the map to make a one-dimensional vector.

B. Min-Pooling for Coarse Class

The base model gives N_f feature maps corresponding to each of the fine class. From those feature maps, we generate the feature maps corresponding to coarse classes. For a coarse class (e.g. wolf), the idea is to exploit the commonalities of the fine classes (e.g. coyote, gray wolf, etc.) belonging to this coarse class. Inspired by [15], we

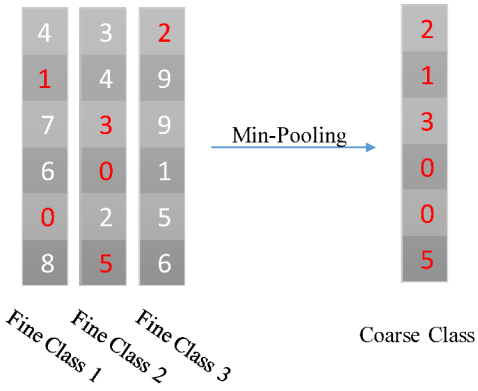

Figure 4: Illustration of the min-pooling operation. Suppose we have a coarse class with three fine classes, the class feature maps for all the three fine classes are available. Through entry-wise min operation, we can obtain a feature map for the coarse class.

implement this by the min-pooling operation across the feature maps of the corresponding fine classes (see Fig. 4 for an illustration). Let k be a coarse class and $\mathcal{C}_k$ be the set of its corresponding fine classes. The feature map M_c^k corresponding to this coarse class is computed as:

$$M_c^k(i) = \min\{M_f^j(i)\}_{j \in \mathcal{C}_k} \quad (1)$$

As suggested in [15], the min-pooling operation captures features that are common across fine classes, but not unique to any of them. So if we perform the min-pooling across fine classes that belong to the same coarse class, the resulting feature map will be able to capture the information about this coarse class.

C. Global Average Pooling

The base model (Sec. IV-A) gives us a set of feature maps M_f^j ($j \in \{1, 2, ..., N_f\}$) for the fine classes. The min-pooling gives another set of feature maps M_c^k ($k \in \{1, 2, ..., N_c\}$) for the coarse classes. From these per-category feature maps, we would like to obtain a classification score for each class. Similar to [8], [15], we get the classification score of a class by attaching a global-average pooling layer to each feature map. In other words, let s_f^j ($j \in \{1, 2, ..., N_f\}$) be the classification score of a fine class and s_c^k ($k \in \{1, 2, ..., N_c\}$) be the classification score of a coarse class. We calculate these scores as:

$$s_f^j = \frac{\sum_{i=1}^{d} M_f^j(i)}{d} \qquad j \in \{1, 2, ..., N_f\} \tag{2}$$

$$s_c^k = \frac{\sum_{i=1}^{d} M_c^k(i)}{d} \qquad k \in \{1, 2, ..., N_c\} \tag{3}$$

Compared with fully connected layers, the main advantage of global average pooling layers is that there are no parameters inside them. This results in less memory footprint and also makes the model less prone to overfitting.

D. Learning

Our main goal is to learn a fine label classifier using images from the two disjoint sets. For images in S_{fine}, both fine and coarse labels are available. However, images in s_{coarse} only have coarse class annotations. For training on those coarse labeled images, the network learns to classify fine labels by exploiting knowledge from the coarse class. The hierarchy architecture of our model is able to do weakly supervised learning. It has two components: an underlying supervised learner (all the way to the fine class feature maps) and a bootstrapping layer (coarse class feature maps) on top of the supervised learner.

For each class (either coarse or fine), the output from the corresponding global average pooling is forwarded to a softmax layer to generate the classification loss. Since we have both fine labels and coarse labels for part of the images, the network will have two loss components, ℓ_f for fine label classification loss, ℓ_c for coarse label classification loss. For a training image x_i with both labels, the loss function can be formulated as:

$$\ell(x_i) = \ell_f(x_i) + \lambda \ell_c(x_i) \quad \text{if } x_i \in S_{fine} \tag{4}$$

where λ is a non-negative hyperparameter used to control the relative importance of the coarse label classification loss. For images with only coarse labels, since we do not have their fine labels to compute ℓ_f, the loss function should be:

$$\ell(x_i) = \lambda \ell_c(x_i) \quad \text{if } x_i \in S_{coarse} \tag{5}$$

To simplify the learning, we introduce the following indicator $\mathbb{1}(x_i)$ to denote whether the training image x_i belongs to S_{fine} or S_{coarse}.

$$\mathbb{1}(x_i) = \begin{cases} 1 & \text{if } x_i \in S_{fine} \\ 0 & \text{if } x_i \in S_{coarse} \end{cases} \tag{6}$$

Then the loss functions defined in Eq. 4 and Eq. 5 can be equivalently written as one loss function:

$$\ell(x_i) = \mathbb{1}(x_i) \cdot \ell_f(x_i) + \lambda \cdot \ell_c(x_i) \tag{7}$$

The loss of the whole training data is simply the summation of the loss of each image in the training set.

To optimize the loss function, we use stochastic gradient descent with momentum of 0.9 and weight decay of 0.0005. The size of the mini-batch used in our experiments varies for different models and different combination of S_{fine} and S_{coarse}. We generally follow two guidelines for setting the mini-batch sizes: 1) keep the ratio of the number of images from S_{fine} and S_{coarse} in each batch equal to $|S_{fine}|/|S_{coarse}|$; 2) use the largest min-batch size that can fit into the GPU memory. We use a Titan X with 12GB memory in the experiments, so the min-batch size varies from 128 to 400. The initial learning rate for all models is set to 0.01 and is further reduced whenever the model performance reaches plateau.

V. Experiments

In this section, we evaluate our proposed model and compare it with other approaches. We first describe some implementation details and the base models we have chosen (Sec. V-A). Then we present results on two datasets: CIFAR100 (Sec. V-B) and a subset of ILSVRC 2010 (Sec. V-C). If not otherwise stated, the results are based on top-1 fine label classification accuracy (%).

A. Implementation Details and Base Models

We consider two existing CNN architectures as our base models. These base models can be used in our experiments with several minor modifications.

NIN: For the CIFAR100 dataset, a 3-layer Network-In-Network (NIN) [8] is used as the base model, this base model is a replication of the one used in the original paper. For the ImageNet dataset, we choose a 4-layer NIN [8], which has more depths and number of parameters compared with the 3-layer model. Both models have also been used as base net for HD-CNN [14] and Taxonomy-Regularized Deep CNN [15]. Since NIN architecture will generate class-wise feature maps by itself, we can directly use it in our experiments with only minor modification (change the output volume of the last convolutional layer to match the number of fine classes and remove the softmax layer).

Note that although we use the same notation *NIN* for the two NINs in the experiments for CIFAR100 and ImageNet, they are different models as we have stated above. This also holds true for the following AlexNet models.

AlexNet: Here we consider a small scale AlexNet introduced in [18] as the second base model for CIFAR100. It differs from the original AlexNet[3] in two ways to make it more suitable to be used for small datasets like CIFAR100. First, it has 3 convolutional layers instead of 5. Second, the

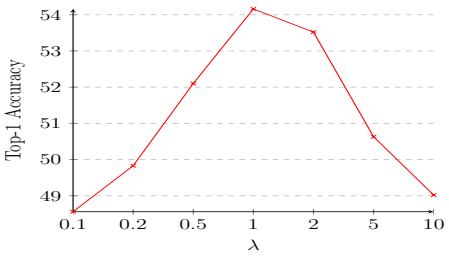

Figure 5: Top-1 validation accuracy on CIFAR100 dataset using NIN as base model. We achieve the best validation performance with $\lambda = 1$.

convolutional layers are followed by one fully connected layer instead of three. We set the dimension of the feature map to be 36 (i.e. $d = 36$). In order to use it for our purpose, we replace the last FC100[1] layer with a FC1024 layer and attach N_f[2] parallel FC36 layers behind it. The resulting model will be able to generate a feature map for each of the 100 fine classes.

For the ImageNet dataset, we use the popular ILSVRC2012 winning model AlexNet[3]. Our implementation of AlexNet is the same as the Caffe AlexNet [19], except that ours is a single column version, which has unrestricted connectivity between layers. For this model, the FC1000 layer is removed and N_f parallel FC36 layers are attached to generate the per-class feature maps.

B. CIFAR100

The CIFAR100 dataset [17] consists of 60000 32x32 color images from $N_f = 100$ fine classes. Each fine class has 500 training images and 100 test images. These 100 fine classes are further grouped into $N_c = 20$ coarse classes, with exactly 5 fine classes in each coarse class. For example, the coarse class *flowers* contains 5 different fine classes: *orchid, poppy, rose, sunflower and tulip*. See Fig. 2 for some example classes on this dataset.

For each fine class, we randomly choose half of the training images to be part of the coarse label subset S_{coarse} and the remaining half to be part of the fine label subset S_{fine}, i.e. $|S_{coarse}| = |S_{fine}| = 0.5|S|$ where $S = S_{coarse} \cup S_{fine}$. We use a small set of images from the training set as validation to set the hyperparameters. Once the optimal hyperparameter is chosen, we reset our model and learn it on the entire training set. All the images are preprocessed with global contrast normalization and ZCA whitening as in [20], [8].

As stated in Sec. IV-D, the weighting parameter λ helps to balance the relative importance of our two loss components. We first conduct a series of experiments to see how the performance varies when λ changes. The result is shown in Fig. 5. Based on this result, we set the hyperparameter $\lambda = 1$ for all the following experiments in this paper.

[1]FC100 denotes a fully connected layer with an output volume of 100.
[2]N_f = 100 for CIFAR100 dataset and 387 for ImageNet subset.

method	NIN	AlexNet
baseline	51.28	45.98
ours	**57.40**	**49.15**
upper bound	64.32[8]	52.80

Table I: Results on the CIFAR100 dataset. We report the overall accuracy of the fine label prediction. We consider two base models: NIN, AlexNet. For each base model, we compare our method (2nd row) with a baseline approach (1st row) which ignores S_{coarse} and learns a standard classification using only S_{fine}. Our method significantly outperforms the baseline in all cases. We also show the result of an oracle model (3rd row). This oracle model is obtained by learning a standard classification model on $S_{fine} \cup S_{coarse}$, but it uses the fine labels for S_{coarse} (while our method uses the coarse labels on S_{coarse}). This oracle model can be considered as an upper bound for our method.

Table I (2nd row) shows the result of our method (trained on $S = S_{fine} \cup S_{coarse}$) using each of the two base models. For comparison, we consider a baseline method that learns the model using only the subset S_{fine}. In other word, this is a standard classification model trained on S_{fine}. The result of this baseline approach is shown in Table I (1st row). Our approach significantly outperforms the baseline method in all cases. For both the baseline and our approach, NIN performs much better than AlexNet on this dataset. This is probably because this dataset is relatively small. Although AlexNet is a powerful model, it is designed for large-scale datasets (e.g. ImageNet) and might be prone to overfitting on small datasets (e.g. CIFAR100).

We also consider an oracle model learned from the entire training data $S = S_{fine} \cup S_{coarse}$, but the learning algorithm has access to the fine labels on S_{coarse} as well. In other words, this is similar to the baseline method (1st row in Table I), but the oracle model is trained on a larger training dataset. The result of this oracle will establish an upper bound for our method, i.e. this is the performance that our model can achieve in the extreme case when the entire training data have fine labels. The result of this oracle model is shown in Table I (3st row).

Fig. 6 shows some qualitative examples of the our method and the baseline NIN model on the CIFAR100 dataset. As we can see from the first two rows, compared with the baseline model, the top-5 predictions made by our model are more semanticly relevant to the ground truth and contains more fine classes from the same coarse class as the ground truth. For example, for the rose (one of the fine classes from the coarse class *flowers*) image in the 2nd row, our model has 5 fine classes belonging to the coarse class *flowers* as its top-5 guesses while the baseline model has only two. This also holds true even for the failure cases in the 3rd row.

In addition, we have found that the base network is easily fooled by objects with similar shape or color from other coarse classes. For example, in the 4th row, the baseline model misclassifies *ray* and *shark* (both belong to the coarse class *fish*) as *dolphin*, which is from another coarse class *aquatic mammals*. This is probably because the example images of *ray* and *shark* highly resemble those of *dolphin*. However, our model is able to avoid this kind of mistakes since it is trained to leverage the knowledge from extra coarsely labeled data.

C. ImageNet

We use a subset of the ILSVRC 2010 dataset [1] as our second dataset and this dataset has also been used in [4] for the same problem. We follow the experiment setup in [4]. The leaf synsets of ILSVRC 2010 are collected as fine classes and their parents as coarse classes, and the class subtrees that overlap are ignored. The original training set are reduced to have $487K$ images for $N_c = 143$ coarse classes and $N_f = 387$ fine classes. Among those 143 coarse classes, 94 of them have 2 fine classes, 26 of them have 3 fine classes, and the rest have 4-9 fine classes. There are between $1.4K$ and $9.8K$ images for each coarse class and between 668 and $2.4K$ images for each fine class. For the validation set and test set, there are 50 and 150 images per fine class respectively. The reduced training set is then split into two disjoint sets for each fine class. The set S_{coarse} has only coarse labels, while the set S_{fine} has both fine labels and coarse labels. We assume $|S_{coarse}| = |S_{fine}| = 0.5|S|$ unless specified otherwise.

During both training and testing time, the original images are resized to have a minimum side of 256 pixels. Randomly cropped and flipped 224x224 patches are forwarded into the learning algorithms. During testing, we follow the 10-view testing in [3]. The predictions of ten 224x224 patches (the center patch and 4 corner patches as well as their horizontal reflections) are averaged as the final prediction.

It is worth noting that we train all the models from scratch. In computer vision, a common practice for learning CNNs is to use some model pre-trained from ImageNet as the starting point and fine-tune the model on the training data [15]. However, this is problematic in our setting. The fine classes on this dataset are a subset of ImageNet. So any model pre-trained on ImageNet would implicitly have seen the fine labels of a large portion of the images from S_{coarse}. Because of this, we choose to learn the model from scratch without using any pre-trained models.

Following [4], we evaluate the performance of our model by varying the relative size of S_{coarse} and S_{fine}. We first fix $|S_{coarse}| = 0.5|S|$ and vary $|S_{fine}| \in \{0.1|S|, 0.2|S|, 0.5|S|\}$. Table II shows how the performance varies with different sizes of $|S_{fine}|$ when using either NIN or AlexNet as the base model. Again, we compare our method with the baseline that only uses S_{fine} and the upper bound performance from the oracle model. We can see that when $|S_{fine}|$ becomes larger, all the models perform better. Both of our models obtain a performance very close to that of the oracle model (65.97% vs 66.93% for NIN, 67.23% vs 68.73% for AlexNet). This clearly shows the ability of our model to utilize the extra coarsely labeled data.

$\|S_{fine}\|$	$0.1\|S\|$	$0.2\|S\|$	$0.5\|S\|$	$0.1\|S\|$	$0.2\|S\|$	$0.5\|S\|$
baseline	32.51	43.90	56.49	39.28	50.58	63.40
ours	**50.77**	**56.70**	**65.97**	**58.43**	**62.27**	**67.23**
upper bound	62.47	64.11	66.93	64.98	65.45	68.73
	NIN			AlexNet		

Table II: Results on ImageNet (with either NIN or AlexNet as the base model) for different S_{fine} sizes. We fix $|S_{coarse}| = 0.5|S|$ and set $|S_{fine}|$ to $0.1|S|$, $0.2|S|$ and $0.5|S|$. Similarly to Table I, we compare our method (2nd row) with the baseline method (1st row) that only uses S_{fine}. We also list the results of the oracle model (3rd row) to provide an upper bound for our method.

$\|S_{fine}\|$	$0.1\|S\|$	$0.2\|S\|$	$0.5\|S\|$
[4]	14.71	16.37	18.46
ours (NIN)	50.77	56.70	65.97
ours (AlexNet)	58.43	62.27	67.23

Table III: Comparison of our methods (with either NIN or AlexNet as the base model) with previous work in [4] for different S_{fine} sizes on ImageNet. We fix $|S_{coarse}| = 0.5|S|$ and set $|S_{fine}|$ to $0.1|S|$, $0.2|S|$ and $0.5|S|$. Again, we compare our method (2nd row) with the baseline method (1st row) that only uses S_{fine}. Our method outperforms the results in [4]. The relative improvement is more significant when the fine labeled dataset is smaller.

For comparison, we also list the our method (with NIN or AlexNet as the base model) with [4] in Table III. Since [4] is not a CNN-based method, it performs much worse than ours. In fact, even our baseline trained only on S_{fine} can significantly outperform [4].

Next we fix $|S_{fine}| = 0.5|S|$ and vary $|S_{coarse}| \in \{0.1|S|, 0.2|S|, 0.5|S|\}$. The results are shown in Table IV. When $|S_{coarse}|$ becomes larger, the performance of our method increases as well. Note that the performance of the baseline stays the same since the baseline only uses $|S_{fine}|$, so increasing $|S_{coarse}|$ will not have any effect on the baseline.

We compare our methods with [4] in Table V when we fix $|S_{fine}|$ and vary $|S_{coarse}|$. Again, our method performs significantly better since [4] is not a CNN-based approach.

Fig. 7 shows some qualitative examples of the our method and the base model (NIN) on the ImageNet dataset. From the examples, we can make observations similar to those on the CIFAR100 dataset. The first three rows show that our model has the ability to make more relevant predictions. The last row shows examples where the baseline model is confused by similar objects from other coarse classes, while

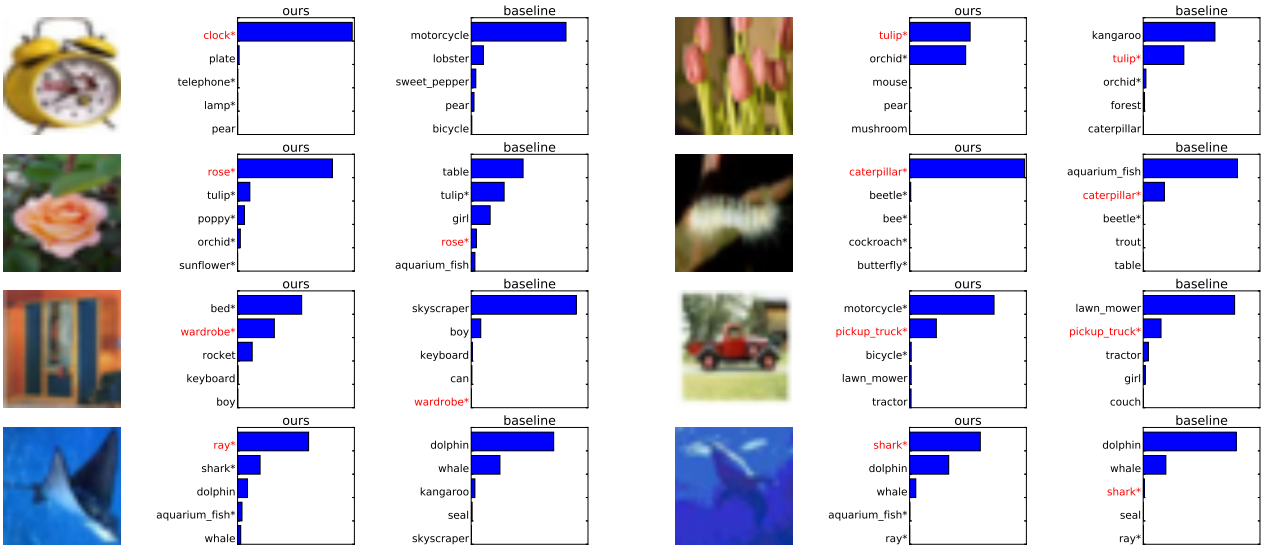

Figure 6: Qualitative examples of baseline NIN model and our model on the CIFAR100 dataset. Ground-truth labels are marked in red. We use $*$ to indicate that the fine classes belong to the same coarse class as the ground-truth. The first three rows show that our model can predict more relevant objects in its top-5 guesses. In addition, the ability of our model to utilize extra coarse labels during training helps it to avoid misclassifying objects into similar objects that belong to another coarse class. Examples are in the 4th row.

$\lvert S_{coarse}\rvert$	$0.1\lvert S\rvert$	$0.2\lvert S\rvert$	$0.5\lvert S\rvert$	$0.1\lvert S\rvert$	$0.2\lvert S\rvert$	$0.5\lvert S\rvert$
baseline	56.49	56.49	56.49	63.40	63.40	63.40
ours	**62.53**	**62.78**	**65.97**	**64.19**	**65.00**	**67.23**
upper bound	63.09	63.54	66.93	65.02	65.94	68.73
		NIN			AlexNet	

Table IV: Results on ImageNet (with either NIN or AlexNet as the base model) for different S_{coarse} sizes. We fix $\lvert S_{fine}\rvert = 0.5\lvert S\rvert$ and set $\lvert S_{coarse}\rvert$ to $0.1\lvert S\rvert$, $0.2\lvert S\rvert$ and $0.5\lvert S\rvert$. The accuracy increases when there are more coarse labeled data available.

$\lvert S_{coarse}\rvert$	$0.1\lvert S\rvert$	$0.2\lvert S\rvert$	$0.5\lvert S\rvert$
[4]	17.87	18.13	18.46
ours (NIN)	62.53	62.78	65.97
ours (AlexNet)	64.19	65.00	67.23

Table V: Comparison of our methods (with either NIN or AlexNet as the base model) with previous work in [4] for different S_{coarse} sizes on ImageNet. We fix $\lvert S_{fine}\rvert = 0.5\lvert S\rvert$ and set $\lvert S_{coarse}\rvert$ to $0.1\lvert S\rvert$, $0.2\lvert S\rvert$ and $0.5\lvert S\rvert$. We compare our method (2nd row) with the baseline method (1st row) that only uses S_{fine}. Our method outperforms the baseline with no surprise.

our model can make the correct predictions.

VI. Conclusion

In this paper, we have investigated the problem of learning image classification when a subset of the training data (i.e. S_{fine}) are annotated with fine labels, while the rest (i.e. S_{coarse}) are annotated with coarse labels. Our goal is to use such weakly labeled data to learn a classifier to predict the fine labels during testing. We have proposed a CNN-based approach to address this problem. The commonalities between fine classes in the same coarse class are naturally captured by min-pooling in our CNN architecture. Our experimental results on CIFAR100 and ImageNet show that our method outperforms the baseline that learns to classify fine labels only based on S_{fine}. Our method also significantly outperforms previous work [4] addressing the same problem.

Acknowledgment

This work was done during Jie Lei's internship at the University of Manitoba funded by the MITACS Globalink program. Yang Wang is supported by grants from NSERC. We thanks NVIDIA for the GPU donations used in this work.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, 2015, arXiv.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

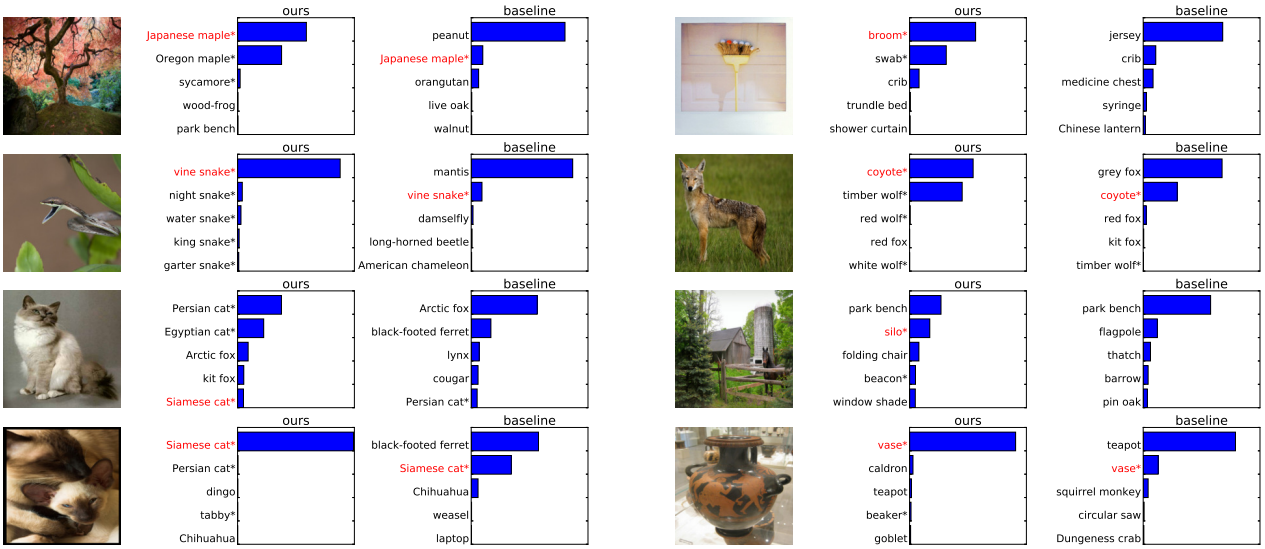

Figure 7: Qualitative examples of base network NIN and our model on the ImageNet dataset. Ground-truth labels are marked in red. We use $*$ to denote that the fine classes belong to the same coarse class as the ground-truth. The first three rows shows that our model can predict more relevant objects in its top-5 guesses. In addition, the ability of our model to utilize extra coarse labels during training helps it to avoid misclassifying objects into similar objects belong to another coarse class. Examples are in the 4th row.

[4] M. Ristin, J. Gall, M. Guillaumin, and L. Van Gool, "From categories to subcategories: Large-scale image classification with partial class label refinement," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[5] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool, "Incremental learning of NCM forests for large-scale image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[8] M. Lin, Q. Chen, and S. Yan, "Network in network," in *International Conference on Learning Representations*, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *Advances in Neural Information Processing Systems*. MIT Press, 2010.

[11] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei, "Fast and balanced: Efficient label tree learning for large scale object recognition," in *Advances in Neural Information Processing Systems*, 2011.

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

[13] V. Ordonez, J. Deng, Y. Choi, A. Berg, and T. L. Berg, "From large scale image categorization to entry-level categories," in *IEEE International Conference on Computer Vision*, 2013.

[14] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu, "HD-CNN: Hierarchical deep convolutional neural networks for large scale visual recognition," in *IEEE International Conference on Computer Vision*, 2015.

[15] W. Goo, J. Kim, G. Kim, and S. J. Hwang, "Taxonomy-regularized semantic deep convolutional neural networks," in *European Conference on Computer Vision*, 2016.

[16] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller, "Recognizing image style," in *British Machine Vision Conference*, 2014.

[17] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.

[18] K. Ahmed, M. H. Baig, and L. Torresani, "Network of experts for large-scale image categorization," in *European Conference on Computer Vision*, 2016.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv:1408.5093*, 2014.

[20] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International Conference on Machine Learning*, 2013.