LEARNING FROM FINE-GRAINED AND LONG-TAILED VISUAL DATA

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yin Cui

August 2019

LEARNING FROM FINE-GRAINED AND LONG-TAILED VISUAL DATA

Yin Cui, Ph.D.

Cornell University 2019

The visual world is fine-grained and long-tailed: many classes are difficult to distinguish; a few classes account for most of the data, while most classes are under-represented. With the remarkable advances in the field of computer vision fueled by large-scale datasets and deep learning, a central question is whether we can quantitatively model such visual data and design deep networks that learn from them.

In this dissertation, we address this question from different perspectives. First, we propose a framework on how to grow a dataset and learn the corresponding model for fine-grained visual recognition with combined human and machine effort. We use deep metric learning to capture the relatively high intra-class variance in fine-grained visual data, assisted by human-annotated hard negatives during the labeling process. We then address the problem of how to design a deep network for fine-grained visual recognition. Specifically, we find nonlinearities in the classifier help the network and thus we explicitly incorporate higher-order nonlinearities into the classifier with our proposed kernel pooling. Further, we focus on methods for fine-grained visual recognition when large-scale, long-tailed data is available. In particular, we show how to measure domain similarity for purposes of selecting a suitable subset from the source domain for improved transfer learning in specific target domains. Next, we present a characterization of long-tailed data distributions based on the effective number of samples, in which we quantify data overlap using a small neighborhood centered around each sample. Finally, we explore how to measure dataset granularity based on clustering theory, as a step toward a more precise definition of “fine-grained.”

BIOGRAPHICAL SKETCH

Yin Cui is a Ph.D. Candidate in the Department of Computer Science at Cornell University working with Prof. Serge Belongie. Before joining Cornell, he recieved his M.S. in Electrical Engineering from Columbia University in 2014 and his B.S. in Electrical Engineering from Beihang University, Beijing, China, in 2012. His research interests are computer vision and deep learning, with a focus on learning from fine-grained and long-tailed visual data. During his Ph.D. study, Yin co-organized COCO Visual Recognition Workshops and Fine-Grained Visual Categorization Workshops at major computer vision conferences. Yin is recipient of the McMullen Fellowship from Computer Science at Cornell University.

To my parents Jiangyu Cui and Lifen Li, and my wife Xingyue Zhou.

ACKNOWLEDGEMENTS

I had a wonderful five-year Ph.D. journey and I am very fortunate to work with my adviser Serge Belongie. First and foremost, I would like to express my gratitude to him, for his countless support through my entire Ph.D. study. Serge always gave me the freedom and courage to explore and is always being very responsive when I needed his help. He has not only provided me guidance on how to conduct research but also helped me grow and become a more independent researcher.

Thank you to my thesis committee members Noah Snavely, Shiri Azenkot and Kilian Weinberger, who have provided invaluable feedback during the whole process of my thesis proposal and defense.

I am lucky to work with so many great researchers at Cornell University. In particular, I want to thank my colleagues Tsung-Yi Lin, Longqi Yang, Andreas Veit, Guandao Yang, Xun Huang, Kimberly Wilber, Menglin Jia, Baoguang Shi, Hani Altwaijry, Zeqi Gu, Isay Katsman and Omid Poursaeed. I really enjoyed spending time with them during my Ph.D. research and life. I also want to extend my gratitude to faculty members Kavita Bala and Bharath Hariharan, for their insightful discussions with me on my research projects.

I had chances to collaborate with amazing researchers from outside Cornell. Among them, I want to especially thank Feng Zhou and Yuanqing Lin for my internship at NEC Labs and Baidu; Yang Song, Chen Sun and Andrew Howard for my internship at Google; Ser-Nam Lim and Laurens van der Maaten for my internship at Facebook; and Grant Van Horn, Oisin Mac Aodha and Pietro Perona from California Institute of Technology.

Last but not least, I want to dedicate my dissertation to my parents Jiangyu Cui and Lifen Li, and my wife Xingyue Zhou. Without their continued love and support, this long and fantastic journey would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In recent years, remarkable progress has been made toward recognizing and understanding the visual world, the primary goal in the field of computer vision. This is largely attributed to the availability of large-scale datasets, remarkable advances in deep learning algorithms and specialized hardware and software. More recently, computer vision algorithms have gained the ability to automatically classify images into thousands of classes by training deep convolutional networks on millions of images and achieve accuracy comparable to humans. Typically, a dataset is designed to have a relatively balanced distribution across classes. However, the visual world is fine-grained and long-tailed: many classes are difficult to distinguish, examples include bird species and car make & model; a few classes claim most of the data, while most classes have very few examples. How to learn from such fine-grained and long-tailed visual data remains challenging for computers. In particular, since many classes are rare in nature and the data collection for them cannot be easily scaled, long-tailed data distribution will be a long-lasting issue in the area of fine-grained visual recognition.

In this dissertation, we present approaches to address the challenge of learning from fine-grained and long-tailed visual data, from the perspectives of quantitatively modeling such visual data and designing deep networks that learn from them. First, we focus on designing deep networks to capture relatively high intra-class variance in the data and incorporate nonlinearities in the classifier, which can improve the performance of the model. Then, we study how to quantitatively model visual data for domain-specific transfer learning in fine-grained target domains and dealing with long-tailed data distribution with class-balanced loss based on effective number of samples. Despite its increasing visibility in our field, "fine-grained" has thus far lacked a precise definition.

In view of this, we conclude with an exploration into how to measure dataset granularity based on clustering theory. We now provide a brief summary of the contributions presented in each chapter.

In Chapter 2, we propose an iterative framework for fine-grained categorization and dataset bootstrapping. Using deep metric learning with humans in the loop, we learn a low dimensional feature embedding for each category to capture intra-class variances and remain discriminative between classes. In each round, images with high confidence scores from our model are sent to humans for labeling. By comparing with exemplar images, labelers mark each candidate image as either a "true positive" or a "false positive." True positives are added into our current dataset and false positives are regarded as "hard negatives" for our metric learning model. Then the model is re-trained with an expanded dataset and hard negatives for the next round.

In Chapter 3, we propose a general pooling framework that captures higher order interactions of features in the form of kernels. Deep convolutional networks with bilinear pooling, initially in their full form and later using compact representations, have yielded impressive performance gains on fine-grained visual categorization. The key to their success lies in the spatially invariant modeling of pairwise (2^{nd} order) feature interactions. We demonstrate how to approximate kernels such as Gaussian RBF up to a given order using compact explicit feature maps in a parameter-free manner. When combined with deep networks, the composition of the kernel can be learned from data in an end-to-end fashion via error back-propagation.

In Chapter 4, we first tackle a problem in large scale fine-grained recognition. Our method won first place in iNaturalist 2017 large-scale species classification challenge. Central to the success of our approach is a training scheme that uses higher image resolution and deals with the long-tailed distribution of training data. Next, we study transfer learning via fine-tuning from large-scale datasets to small-scale, domain-specific FGVC datasets. Transferring the knowledge learned from large-scale datasets (*e.g.*, ImageNet) via fine-tuning offers an effective solution for domain-specific fine-grained visual categorization (FGVC) tasks (*e.g.*, recognizing bird species or car make & model). In such scenarios, data annotation often calls for specialized domain knowledge and thus is difficult to scale. We propose a measure to estimate domain similarity via Earth Mover's Distance and demonstrate that transfer learning benefits from pre-training on a source domain that is similar to the target domain by this measure.

In Chapter 5, we argue that as the number of samples increases, the marginal benefit of a newly added data point will diminish. We introduce a theoretical framework to measure data overlap by associating with each sample a small neighboring region rather than a single point. The effective number of samples is defined as the volume of samples and can be calculated by a simple formula $(1-\beta^n)/(1-\beta)$, where n is the number of samples and $\beta \in [0, 1)$ is a hyperparameter. We design a re-weighting scheme that uses the effective number of samples for each class to re-balance the loss, thereby yielding a *class-balanced loss*. When trained with the proposed class-balanced loss, the network is able to achieve improved performance on long-tailed datasets.

In Chapter 6, building upon clustering theory, we pursue a framework for measuring dataset granularity. We argue that dataset granularity should depend not only on the data samples and their labels, but also on the distance function we choose. We propose an axiomatic framework to capture desired properties for a dataset granularity measure and provide examples of measures that satisfy these properties. We assess each measure via experiments on datasets with hierarchical labels of varying granularity. When measuring granularity in commonly used datasets using our measure with features from deep networks, we find datasets with higher granularities (fine-grained) are more difficult to learn. In addition, we observe that deep features have more difficulty generalizing to datasets with higher granularity. We perform robustness studies and find that dataset granularity is relatively insensitive to changes in network architecture, but sensitive to the choice of pre-trained data. Dataset granularity is also sensitive to nuisance factors arising during image acquisition, including noise and reduced resolution. As a consequence of our study, we find that certain datasets that are widely considered fine-grained in fact have lower granularity than other datasets generally considered coarse-grained.

Chapter 7 concludes the dissertation and discusses future directions.

CHAPTER 2
FINE-GRAINED CATEGORIZATION AND DATASET BOOTSTRAPPING USING DEEP METRIC LEARNING WITH HUMANS IN THE LOOP

Fine-grained visual categorization (FGVC) has received increased interest from the computer vision community in recent years. By definition, FGVC, as a sub-field of object recognition, aims to distinguish subordinate categories within an entry-level category. For example, in fine-grained flower categorization [122, 123, 4], we want to identify the species of a flower in an image, such as "nelumbo nucifera (lotus flower)," "tulip" or "cherry blossom." Other examples include classifying different types of plants [100], birds [17, 16], dogs [89], insects [102], galaxies [39, 31]; recognizing brand, model and year of cars [95, 173, 179]; and face identification [153, 143].

Most existing FGVC methods fall into a classical two-step scheme: feature extraction followed by classification [3, 13, 22, 131]. Since these two steps are independent, the performance of the whole system is often suboptimal compared with an end-to-end system using Convolutional Neural Networks (CNN) that can be globally optimized via back-propagation [16, 187, 96, 110]. Therefore, in this chapter, we focus on developing an end-to-end CNN-based method for FGVC. However, compared with general purpose visual categorization, there are three main challenges arising when using such end-to-end CNN-based systems for FGVC.

Firstly, **lack of training data**. Current commonly used CNN architectures such as AlexNet [99], VGGNet [147], GoogLeNet-Inception [151] and ResNet [66] have large numbers of parameters that require vast amounts of training data to achieve reasonably good performance. Commonly used FGVC databases [123, 17, 89, 95], however, are relatively small, typically with less than a few tens of thousands of training images.

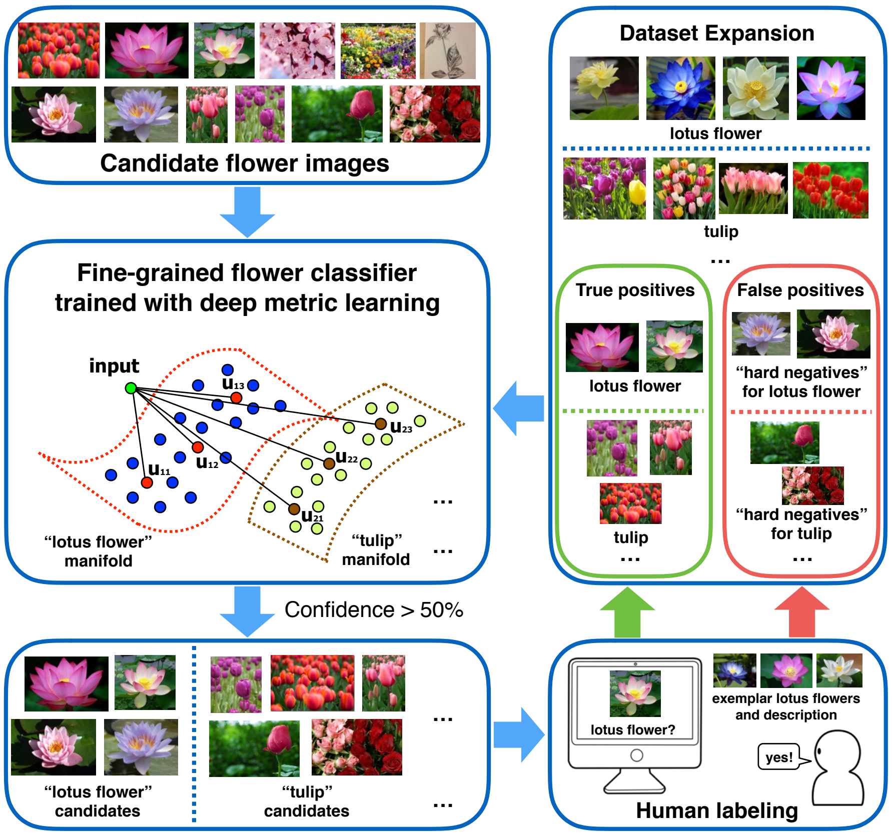


Figure 2.1: Overview of the proposed framework. Using deep metric learning with humans in the loop, we learn a low dimensional feature embedding for each category that can be used for fine-grained visual categorization and iterative dataset bootstrapping.

Secondly, compounding the above problem, FGVC can involve **large numbers of categories**. For example, arguably, it is believed that there are more than $400,000$ species of flowers in the world [83]. As a point of reference, modern face identification systems need to be trained on face images coming from millions of different identities (categories). In such scenarios, the final fully connected layer of a CNN before the softmax layer would contain too many nodes, thereby making the training infeasible.

Lastly, **high intra-class vs. low inter-class variance**. In FGVC, we confront two somewhat conflicting requirements: distinguishing visually similar images from differ-

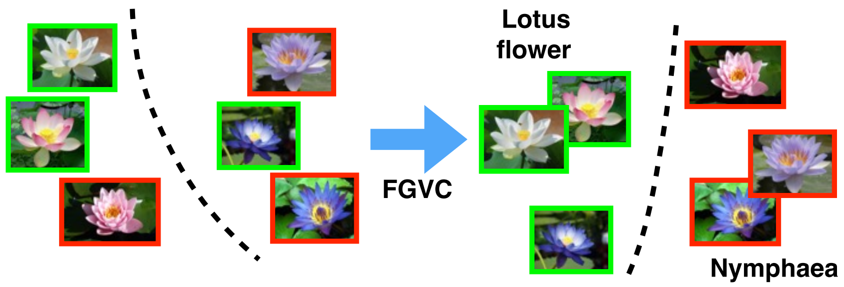


Figure 2.2: Simple appearance based methods will likely find incorrect groups for two visually similar categories. A successful FGVC approach should be able to deal with the challenge of high intra-class vs. low inter-class variance.

ent categories while allowing reasonably large variability (pose, color, lighting conditions, etc.) within a category. As an example illustrated in Figure 2.2, images from different categories could have similar shape and color. On the other hand, sometimes images within same category can be very dissimilar due to nuisance variables. In such a scenario, since approaches that work well on generic image classification often focus on inter-class differences rather than intra-class variance, directly applying them to FGVC could make visually similar categories hard to be distinguished.

In this chapter, we propose a framework that aims to address all three challenges. We are interested in the following question: given an FGVC task with its associated training and test set, are we able to improve the performance by bootstrapping more training data from the web? In light of this, we propose a unified framework using deep metric learning with humans in the loop, illustrated in Figure 2.1.

We use an iterative approach for dataset bootstrapping and model training. In each round, the model trained from last round is used to generate fine-grained confidence scores (probability distribution) for all the candidate images on categories. Only images with highest confidence score larger than a threshold are kept and put into the corresponding category. Then, for each category, by comparing with exemplar images and category definitions, human labelers remove false positives (hard negatives). Images that pass the human filtering will be included into the dataset as new (vetted) data. Finally, we re-train our classification model by incorporating newly added data and also leveraging the hard negatives marked by human labelers. The updated model will be used for the next round of dataset bootstrapping. Although we focus on flower categorization in this chapter, the proposed framework is applicable to other FGVC tasks.

In order to capture within-class variance and utilize hard negatives as well, we propose a triplet-based deep metric learning approach for model training. A novel metric learning approach enables us to learn low-dimensional manifolds with multiple anchor points for each fine-grained category. These manifolds capture within-category variances and remain discriminative to other categories. The data can be embedded into a feature space with dimension much lower than the number of categories. During the classification, we generate the categorical confidence score by using multiple anchor points located on the manifolds.

In summary, the proposed framework handles all three challenges in FGVC mentioned above. Using the proposed framework, we are able to grow our training set and get a better fine-grained classifier as well.

2.1 Related Work

Fine-Grained Visual Categorization (FGVC). Many approaches have been proposed recently for distinguishing between fine-grained categories. Most of them [3, 13, 22, 131] use two independent steps: feature extraction and classification. Fueled by the recent advances in Convolutional Neural Networks (CNN) [99, 57], researchers have gravitated to CNN features [16, 187, 96, 131, 110] rather than traditional hand-crafted features such as LLC [3] or Fisher Vectors [52]. Sometimes, the information from segmentation [96], part annotations [16], or both [22] is also used during the feature extraction. Current state-of-the-art methods [16, 187, 96, 110] all adopt CNN-based end-to-end schemes that learn feature representations from data directly for classification. Although our method also draws upon a CNN-based scheme, there are two major differences. 1) Rather than using softmax loss, we aim to find a low-dimensional feature embedding for classification. 2) We incorporate humans into the training loop, with the human-provided input contributing to the training of our model.

Fine-Grained Visual Datasets. Popular fine-grained visual datasets [123, 167, 89, 95] are relatively small scale, typically consisting of around 10 thousand training images or less. There are some efforts recently in building large-scale fine-grained datasets [159, 179]. We differ from these efforts both in terms of our goal and our approach. Instead of building a dataset from scratch, we aim to bootstrap more training data to enlarge the existing dataset we have. In addition, instead of human labeling, we also use a classifier to help during the dataset bootstrapping. The most similar work in terms of dataset bootstrapping comes from Yu *et al.* [184], which builds a large-scale scene dataset with 10 common categories using deep learning with humans in the loop. However, we are bootstrapping a fine-grained dataset with much more categories (620). Moreover, instead of a dataset, we can also get a model trained with combined human-machine efforts.

Deep Metric Learning. Another line of related work is metric learning with CNNs using pairwise [26, 63] or triplet constraints [168, 143, 70]. The goal is to use a CNN with either pairwise (contrastive) or triplet loss to learn a feature embedding that captures the semantic similarity among images. Compared with traditional metric learning methods that rely on hand-crafted features [176, 58, 171, 25], deep metric learning directly learns from data and achieves much better performance. Recently, it has been successfully applied to variety of problems including face recognition and verification [153, 143], image retrieval [168], semantic hashing [101], product design [10], geo-localization [105] and style matching [165]. In contrast with previous methods, we propose a novel strategy that enables the learning of continuous manifolds. In addition, we also bring humans in the loop and leverage their inputs during metric learning.

2.2 Dataset Bootstrapping

One of the main challenges in fine-grained visual recognition is the scarcity of training data. Labeling of fine-grained categories is tedious because it calls for experts with specialized domain knowledge. This section presents a bootstrapping framework on how to grow a small scale, fine-grained dataset in an efficient manner.

2.2.1 Discovering Candidate Images

In this first step, we wish to collect a large pool of candidate images for fine-grained subcategories under a coarse category, *e.g.*, flowers. The most intuitive way to crawl images could resort to image search engines like Google or Bing. However, those returned images are often iconic, presenting a single, centered object with a simple background, which is not representative of natural conditions.

On the other hand, with the prevalence of powerful personal cameras and social networks, people capture their day-to-day photos and share them via platforms like Instagram or Flickr. Those natural images uploaded by web users offer us a rich source of candidate images, often with tags that hint at the semantic content. So if we search "flower" on Instagram, a reasonable portion of returned images should be flower images. Naturally, we will need a filtering process to exclude the non-flower images.

We first downloaded two million images tagged with "flower" via the Instagram API. To remove the images that clearly contain no flowers, we pre-trained a flower classifier based on GoogLeNet-Inception [151] with 70k images. By feeding all the downloaded images to this classifier, we retained a set of nearly one million images, denoted as $\mathcal{C}$, with confidence score larger than 0.5.

2.2.2 Dataset Bootstrapping with Combined Human-Machine Efforts

Given an initial fine-grained dataset $\mathcal{S}_0$ of N categories and a candidate set $\mathcal{C}$, the goal of dataset bootstrapping is to select a subset $\mathcal{S}$ of the images from $\mathcal{C}$ that match with the original N categories. We divided the candidate set into a list of k subsets: $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_k$ and used an iterative approach for dataset bootstrapping with k iterations in total.

Each iteration consists of three steps. Consider the i-th iteration. First, we trained a CNN-based classifier (see Section 2.3) using the seed dataset $\mathcal{S}_{i-1} \cup \mathcal{H}_{i-1}$, where $\mathcal{H}_{i-1}$ contains the hard negatives from the previous step. Second, using this classifier, we assigned each candidate image $x \in \mathcal{C}_i$ to one of the N categories. Images with confidence score larger than 0.5 form a high quality candidate set $\mathcal{D}_i \subset \mathcal{C}_i$ for the original N categories. Third, we asked human labelers with domain expertise to identify true positives $\mathcal{T}_i$ and false positives $\mathcal{F}_i$, where $\mathcal{T}_i \cup \mathcal{F}_i = \mathcal{D}_i$. Exemplar images and category definitions were shown to the labelers.

Compared to the traditional process requiring the labeler to select one of N categories per image, we asked labelers to focus on a binary decision task which entails significantly less cognitive load. Noting that these false positives $\mathcal{F}_i$ are very similar to ground-truths, we regard them as hard negatives $\mathcal{H}_i \leftarrow \mathcal{H}_{i-1} \cup \mathcal{F}_i$. True positives were also included to expand our dataset: $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \mathcal{T}_i$ for the next iteration.

It is worth mentioning this bootstrapping framework is similar in spirit to the recent work [166, 69] that used semi-automatic crowdsourcing strategy to collect and annotate videos. However, the key difference is we design a deep metric learning method (see Section 2.3) that specifically makes the use of the large number of hard negatives $\mathcal{H}_i$ in each iteration.

2.3 Deep Metric Learning for FGVC

We frame our problem as a deep metric learning task. We choose metric learning for mainly two reasons. First, compared with classic deep networks that use softmax loss in training, metric learning enables us to find a low-dimensional embedding that can well capture high intra-class variance. Second, metric learning is a good way to leverage human-labeled hard negatives. It is often difficult to get categorical labels for these hard negatives. They could belong to flower species outside the dataset, or non-flower images. Therefore, directly incorporating human-labeled hard negatives into a multi-way classification scheme such as softmax is infeasible, while it is quite natrual to include them into the metric learning.

Figure 2.3 illustrates the differences between CNN with softmax and CNN for metric learning in 3-dimensional feature space. In order to minimize softmax loss, we try to map all images within the same category to a single point in feature space, which loses

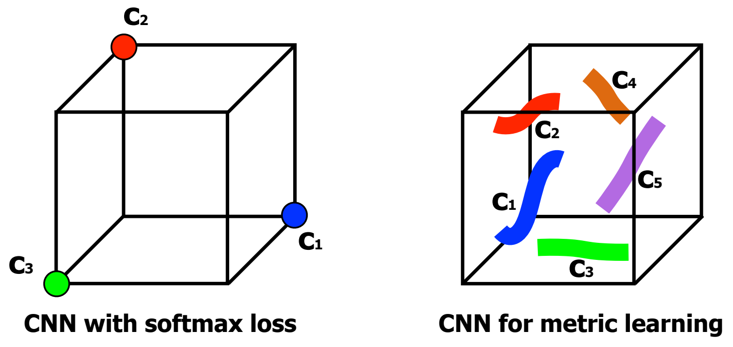


Figure 2.3: Comparison between CNN with softmax and CNN for metric learning in feature space, where c_i denotes images within the same category.

the intra-class variance. In this figure, we try to map category c_1 to $[1, 0, 0]^\top$, c_2 to $[0, 1, 0]^\top$ and c_3 to $[0, 0, 1]^\top$, respectively. We need N nodes in final feature layer to represent N categories. However, in metric learning, we can learn manifolds and the dimensionality of the feature layer could be much smaller than N. In addition, the manifold can preserve useful intra-class variances such as color and pose.

Our goal is to learn a non-linear low-dimensional feature embedding $f(\cdot)$ via CNN, such that given two images x and y, the Euclidean distance between $f(x)$ and $f(y)$ can reflect their semantic dissimilarity (whether they come from same category or not). Typically, people use pairwise or triplet information to learn the feature embedding.

In the pairwise case [26, 63], $f(\cdot)$ is learned from a set of image pairs $\{(x_i, y_i)\}$ with corresponding labels $\{l_i\}$ indicating whether x_i and y_i is similar. In the triplet case [168, 70], $f(\cdot)$ is learned from a set of image triplets $\{(x, x_p, x_n)\}$, which constrains the reference image x to be more similar with the image x_p of the same category compared with any image x_n of different class. We can see triplet constraints offer more fine-grained information: by making use of relative comparisons it is adaptive to differing granularity of similarity while the pairwise counterpart is not. We therefore use triplet information to develop an end-to-end CNN-based approach for FGVC.

2.3.1 Triplet-based Deep Metric Learning

The triplet-based deep metric learning framework is illustrated in Figure 2.4. In each iteration, the input triplet (x, x_p, x_n) is sampled from the training set, where image x is more similar to x_p relative to x_n. Then the triplet of three images are fed into an identical CNN simultaneously to get their non-linear feature embeddings $f(x)$, $f(x_p)$ and $f(x_n)$. The CNN could be any arbitrary architecture such as AlexNet [99], VGGNet [147] or GoogLeNet-Inception [151]. Since we need to compute the distances in feature space, all the features should be normalized to eliminate the scale differences. We use L_2-normalization for this purpose: $f(x) \leftarrow \frac{f(x)}{\sqrt{f(x)^\top f(x)}}$.

We use the triplet loss same as Wang *et al.* [168] used, which can be expressed as

$$\begin{aligned}&\mathcal{L}_{triplet}(x, x_p, x_n) = \\ &\max\left\{0, \left\|f(x) - f(x_p)\right\|_2^2 - \|f(x) - f(x_n)\|_2^2 + m\right\}\end{aligned} \tag{2.1}$$

where m is a hyper-parameter that controls the distance margin after the embedding. This hinge loss function will produce a non-zero penalty of $\left\|f(x) - f(x_p)\right\|_2^2 - \|f(x) - f(x_n)\|_2^2 + m$ if the L_2 distance between x and x_n is smaller than the L_2 distance between x and x_p adding a margin m in feature space: $\|f(x) - f(x_n)\|_2^2 < \left\|f(x) - f(x_p)\right\|_2^2 + m$. The loss will be back propagated to each layer of the CNN and their corresponding parameters are updated through stochastic gradient descent.

2.3.2 Training from Hard Negatives

The most challenging part of training a triplet-based CNN lies in the triplet sampling. Since there are $O(n^3)$ possible triplets on a dataset with n training data, going through all of them would be impractical for large n. A good triplet sampling strategy is needed to make training feasible.

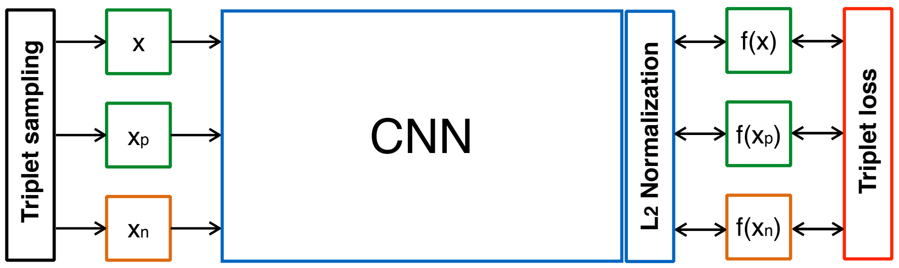


Figure 2.4: Triplet-based deep metric learning. In the input triplet, image x is closer to x_p than it is to x_n. We train a CNN to preserve this relative ordering under feature embedding $f(\cdot)$.

We observed that during training, if we use randomly sampled triplets, many of them satisfy the triplet constraint well and give nearly zero loss in Equation 2.1. That is, those easy triplets have no effect in updating model parameters but we waste our time and resources in passing them through the network. This makes the training process extremely inefficient and unstable: only few examples make contributions to the training within a batch.

Therefore, we use an online hard negatives mining scheme: only train on those triplets that violate the triplet constraint and give non-zero loss will be included into the training. Why not simply train from the hardest negatives, *i.e.*, triplets with the largest $\left\|f(x) - f(x_p)\right\|_2^2 - \|f(x) - f(x_n)\|_2^2$? Because there are noisy data in the training set and trying to satisfy them ruins the overall performance. A similar scenario was also reported in [143].

In our framework, instead of using images coming from categories that are different from the reference image, we also incorporate false positives marked by human labelers as hard negative candidates. Those false positives are all misclassified by our model and thus provide us access to an excellent source of hard negatives.

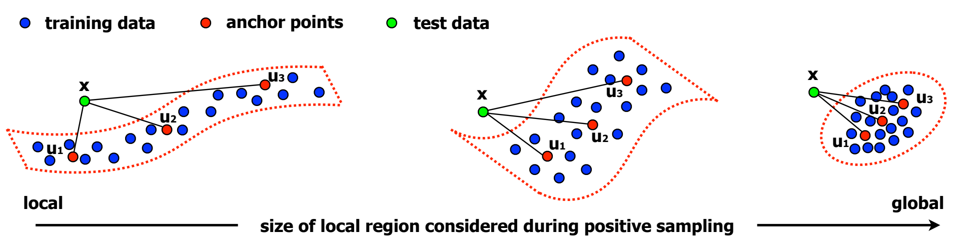


Figure 2.5: Control the shape of manifolds by sampling from local positive region. As the local region considered in positive sampling grows, the learned manifold will be increasingly dense.

2.3.3 Learning Manifolds

Typically, given the reference image x, the positive image x_p is sampled from all images within the same category as x. Suppose we have a training set with n images $\{x_i\}$ with labels $\{C(x_i)\}$ from K categories, where $i = 1, 2, \ldots, n$ and $C(x_i) \in \{1, 2, \ldots, K\}$. In this setting, considering a reference image x within a fine-grained category, suppose the maximum between-class distance for x in feature space is bounded by D. That is, $\|f(x) - f(x_n)\|_2 \leq D$, $\forall\ C(x_n) \neq C(x)$. In order to have 0 triplet loss for the reference image x, we need $\left\|f(x) - f(x_p)\right\|_2^2 \leq \|f(x) - f(x_n)\|_2^2 - m$, $\forall\ C(x_p) = C(x), C(x_n) \neq C(x)$. Therefore, $\forall\ x_i, x_j$ where $C(x_i) = C(x_j) = C(x)$,

$$\begin{aligned} \left\|f(x_i) - f(x_j)\right\|_2^2 &\leq \|f(x) - f(x_i)\|_2^2 + \left\|f(x) - f(x_j)\right\|_2^2 \\ &\leq 2(D^2 - m) \end{aligned} \tag{2.2}$$

The squared within-class pairwise distance is bounded by $2(D^2 - m)$. Thus, by using triplet loss with positives sampled from all images in the same class, we are trying to map all images within that class into a hypersphere with radius $r = \frac{\sqrt{2(D^2-m)}}{2}$. In FGVC, between-class distances could be very small compared with the within-class distances. In such a scenario, $D^2 - m$ could be very close to or even less than 0, which makes the training process very difficult.

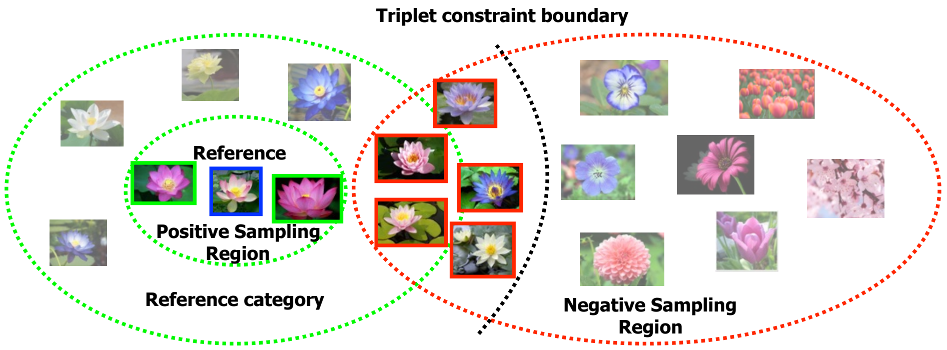


Figure 2.6: Triplet sampling strategy, in which for a reference image, positives are sampled locally and only hard negatives are kept.

However, if we only force positives to be close to the reference locally, we are able to learn an extended manifold rather than a contracted sphere. As illustrated in Figure 2.5, as the considered local positive region grows, the learned manifold will be increasingly contracted, eventually becoming a sphere when using all positives within the same category.

The triplet sampling strategy we used is summarized in Figure 2.6. Given a reference image x (in the blue bounding box) we sample positive images $\{x_p\}$ (in the green bounding boxes) from the local region inside the same category. Negative images $\{x_n\}$ are sampled from different categories but we only keep those hard negatives (marked by red bounding boxes): negatives that violate the triplet constraint with respect to the positives we chose.

2.3.4 Classification

After the manifold learning step, we adopt a soft voting scheme using anchor points on manifolds for classification. For each category, the anchor points are generated by K-means clustering on the training set in feature space. Suppose we have N categories and each category has K anchor points. The j-th anchor point for category i is represented as u_{ij}, where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, K$. Given an input query image x, we first extract its feature embedding $f(x)$ from our network, then the confidence score for category i is generated as

$$p_i = \frac{\sum_{j=1}^{K} e^{-\gamma \|f(x)-u_{ij}\|_2^2}}{\sum_{l=1}^{N} \left(\sum_{j=1}^{k} e^{-\gamma \|f(x)-u_{lj}\|_2^2} \right)} \tag{2.3}$$

The predicted label of x is the category with the highest confidence score: $\operatorname{argmax}_i p_i$. γ is a parameter controlling the "softness" of label assignment and closer anchor points play more significant roles in soft voting. If $\gamma \to \infty$, only the nearest anchor point is considered and the predicted label is "hard" assigned to be the same as the nearest anchor point. On the other hand, if $\gamma \to 0$, all the anchor points are considered to have the same contribution regardless of their distances between $f(x)$.

Notice that during the prediction, the model is pre-trained offline and all the anchor points are calculated offline. Therefore, given a query image, we only need a single forward pass in our model to extract the features. Since we have learned a low-dimensional embedding, computing the distances between features and anchor points in low-dimensional space is very fast.

2.3.5 Learning Anchor Points

As we just described, after metric learning, we use K-means to generate anchor points for representing manifolds and prediction. This could lead to suboptimal performance. In fact, we can go one step further to directly learn anchor points by including soft voting into our triplet-based metric learning model, which is illustrated in Figure 2.7. For simplicity, the data part is not shown.

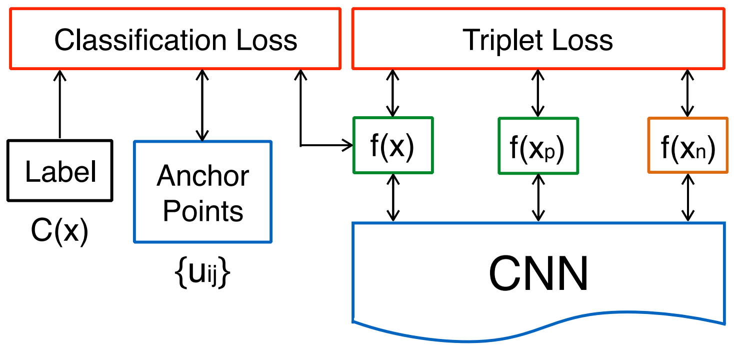


Figure 2.7: Combining anchor points learning into triplet network. The classification loss is used to update the anchor points.

In contrast to the previous model in Figure 2.4 that uses only triplet information, we also leverage the category label $C(x)$ for the reference image x and learn anchor points for classification. We can generate confidence scores p_i for $f(x)$ using anchor points $\{u_{ij}\}$ by soft voting in Equation 2.3. The classification loss we used is logistic loss on top of confidence score:

$$\mathcal{L}_{classification}(x, \{u_{ij}\}, C(x)) = -\log(p_{C(x)}) \tag{2.4}$$

where $p_{C(x)}$ is given in Equation 2.3 by substituting i with $C(x)$. If we have very high confidence score on the true category, $p_{C(x)} \rightarrow 1$, then the loss will be very small: $\mathcal{L}_{classification} \rightarrow 0$.

The overall loss is the weighted sum of triplet and classification loss:

$$\mathcal{L} = \omega \mathcal{L}_{triplet} + (1 - \omega)\mathcal{L}_{classification} \tag{2.5}$$

During training, the loss will be back-propagated to both CNN and anchor points. Anchor point u_{ij} will be updated based on the gradient of the loss with respect to u_{ij}: $\frac{\partial \mathcal{L}}{\partial u_{ij}}$. Since we combine both triplet and categorical information and also learn anchor points directly for classification, we can expect better performance over the triplet-based model.

2.4 Experimental Evaluation

In this section, we present experiments to evaluate the proposed deep metric learning approach against traditional two-step metric learning using deep features and commonly used softmax loss on our flower dataset and another publicly available dataset. We also evaluate the effectiveness of dataset bootstrapping and training with humans in the loop.

2.4.1 Experiments Setup

We compare the performance of the proposed deep metric learning approach with the following baselines: (1) Softmax loss for classification (**Softmax**). The most commonly used scheme in general purpose image classification. The deep network is trained from data with categorical label using softmax loss. We can get label prediction directly from the network output. (2) Triplet loss with naive sampling (**Triplet-Naive**). The architecture illustrated in Figure 2.4 with randomly sampled triplets: given a reference image, the triplet is formed by randomly sampling a positive from same category and a negative from different category. Those triplets are directly fed into triplet network. During testing, we use the classification scheme described in Section 2.3.4. (3) Triplet loss with hard negative mining (**Triplet-HN**). As discussed in Section 2.3.2, instead of feeding all the triplets into the network, we only keep those hard negatives that violate triplet constraint. (4) Triplet loss with manifold learning (**Triplet-M**). As mentioned in Section 2.3.3, the positives are sampled locally with respect to the reference image from same category. (5) Triplet loss with anchor points learning (**Triplet-A**). We combine anchor points learning with triplet network as illustrated in Figure 2.7. During testing, the network directly output label prediction based on confidence scores. In addition, we also compared with state-of-the art FGVC approaches on publicly available dataset.

Since the network is trained via stochastic gradient descent, in order to do online sampling of triplets, we need to extract features on the entire training set, which is certainly inefficient if we do it for each iteration. Therefore, as a trade-off, we adopt a quasi-online sampling strategy: after every $1,000$ iterations, we pause the training process and extract features on the training set, then based on their euclidean distances in feature space, we do triplet sampling (local positives and hard negatives) to generate a list of triplets for next $1,000$ iterations and resume the training process using the newly sampled triplets.

The CNN architecture we used is GoogLeNet-Inception [151], which achieved state-of-the-art performance in large-scale image classification on ImageNet [36]. All the baseline models are trained with fine-tuning using pre-trained GoogleNet-Inception on ImageNet dataset.

We used Caffe [81], an open source deep learning framework, for the implementation and training of our networks. The models are trained on NVIDIA Tesla K80 GPUs. The training process typically took about 5 days on a single GPU to finish $200,000$ iterations with 50 triplets in a batch per each iteration.

Method (feature dimension)	Accuracy (%)
Softmax (620)	65.1
Triplet-Naive (64)	48.7
Triplet-HN (64)	64.6
Triplet-M (64)	65.9
Triplet-A (64)	**66.8**

Table 2.1: Performance comparison on our *flowers-620* dataset.

2.4.2 Deep Metric Learning

We evaluate the baselines on our flower dataset and publicly available CUB-200 Birds dataset [167]. There are several parameters in our model and the best values are found through cross-validation. For all the following experiments on both dataset, we set the margin m in triplet loss to be 0.2; the feature dimension for $f(\cdot)$ to be 64; the number of anchor points per each category K to be 3; the γ in soft voting to be 5. We set $\omega = 0.1$ to make sure that the triplet loss term and the classification loss term in Equation 2.5 have comparable scale. For the size of positive sampling region, we set it to be 60% of nearest neighbors within same category. The effect of positive sampling region size will also be presented later in this section.

Flowers-620. *flowers-620* is the dataset we collected and used for dataset bootstrapping, which contains $20,211$ images from 620 flower species, in which $15,437$ images are used for training. The performance comparison of mean accuracy is summarized in Table 2.1.

From the results, we have the following observations: (1) Triplet-Naive, which uses randomly offline sampling, performed much worse compared with other triplet baselines, which clearly shows the importance of triplet sampling in training. (2) Accuracy increases from Triplet-HN to Triplet-M, showing the effectiveness of learning a bet-

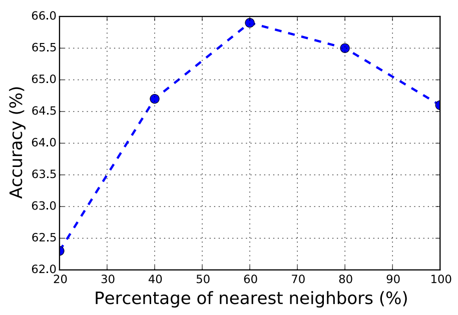
Method (feature dimension)	Accuracy (%)
Alignments [53]	67.0
MsML [131]	67.9
Symbiotic* [22]	69.5
POOF* [13]	73.3
PB R-CNN* [187]	82.0
B-CNN [110]	85.1
PNN* [16]	85.4
Softmax (620)	77.2
Triplet-Naive (64)	61.2
Triplet-HN (64)	77.9
Triplet-M (64)	79.3
Triplet-A (64)	**80.7**

Table 2.2: Performance comparison on *birds-200* dataset. "*" indicates methods that use ground truth part annotations.

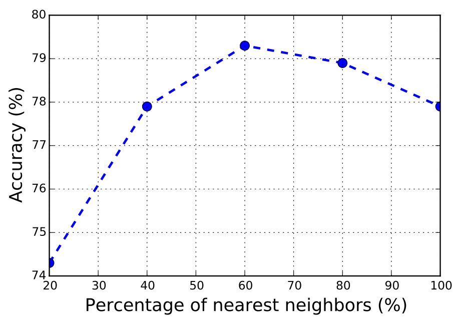
ter manifolds with local positive sampling. (3) Triplet-A performed best and achieved higher accuracy than Softmax. This verifies our intuition that fine-grained categories often have high intra-class difference and such within-class variance can be well captured by learning manifolds with multiple anchor points. In this way, even in a much lower dimensional feature space, the discrimination of the data can still be well preserved. While in Softmax, we are trying to map all the data within a category to a single point in feature space, which fails to capture the within-class structure well.

Birds-200. *birds-200* is the Caltech-UCSD Birds-200-2011 data set for fine-grained birds categorization. There are $11,788$ images from 200 bird species. Each category has around 30 images for training. In training and testing, we use the ground truth bounding boxes to crop the images before feeding them to the network. The performance comparison is summarized in Table 2.2.

Similar to what we just observed in *flowers-620*, experiment results verify the effectiveness of proposed methods. We also compared to recent state-of-the-art approaches



(a) *flowers-620*



(b) *birds-200*

Figure 2.8: Accuracy with varying positive sampling region size.

for fine-grained categorization. Notice that we outperformed MsML [131] by a significant margin, which is a state-of-the-art metric learning method for FGVC. Although our method performed worse than the recent proposed B-CNN [110], we were able to achieve either better or comparable results with those state-of-the-arts using ground truth part annotations during training and testing.

We also evaluate the effect of local positive sampling region size. As we mentioned earlier in Section 2.3.3, the size of local positive sampling region controls the shape of manifolds. We want to learn manifolds that can capture within-class variance well but not too spread out to lose the between-class discriminations.

Figure 2.8 shows the mean accuracy with varying local positive sampling region using Triplet-M. Using 60% of nearest neighbors for positive sampling gives best results on both *flowers-620* and *birds-200*.

2.4.3 Dataset Bootstrapping

During dataset bootstrapping, other than true positives that passed human filtering and included into our dataset, plenty of false positives were marked by human labelers.

Those false positives are perfect hard negatives in our metric learning framework. Therefore, we combined these human labeled hard negatives with negatives from other categories that violate triplet constraint during triplet sampling. We sampled same number of human-labeled hard negatives as the hard negatives from other categories.

With the proposed framework, we included $11,567$ Instagram flower images into our database, which almost doubles the size of our training images to $27,004$. At the same time, we also get $240,338$ hard negatives from labelers. We call this new dataset *flowers-620 + Ins* and will use it for the evaluation of dataset bootstrapping. Notice that the test set in *flowers-620 + Ins* remains same as *flowers-620*.

For best quality, currently we only use in-house labelers. Our framework could be deployed to crowdsourced labeling platforms like Amazon Mechanical Turk, bit with good quality control schemes.

We show that by dataset bootstrapping with humans in the loop, we are able to get a better model using the proposed metric learning approach. For a fair comparison, we also include two baselines that enable hard negatives to be utilized in softmax scheme: (1) SoftMax with all hard negatives as a single novel category (**Softmax + HNS**). The model is trained with one additional hard negative category. (2) SoftMax with hard negatives as multiple novel categories (**Softmax + HNM**). In this setting, instead of mixing all hard negatives as a single category, we regard hard negatives for different flower categories as different novel categories. The model is trained with data from $620\times2 = 1240$ categories, from which 620 of them are category-specific hard negatives. To make the number of flower images and hard negatives to be balanced in each batch during training, the number of epochs we go through on all hard negatives is set to be 10% of 620 flower categories. In testing, only confidence scores from 620 flower categories will be considered for both baselines. The experiment results on *flowers-620*

Method (feature dimension)	Accuracy (%)
Softmax (620)	68.9
Softmax + HNS (621)	70.3
Softmax + HNM (1240)	70.8
Triplet-A (64)	70.2
Triplet-A + HN (64)	**73.7**

Table 2.3: Performance comparison on *flowers-620* + *Ins*.

+ *Ins* are shown in Table 2.3.

Compared with results in Table 2.1, we got 6.9% improvement by dataset bootstrapping. If we look at the breakdown, 3.4% came from the newly added Instagram training images and 3.5% came from human labeled hard negatives, indicating hard negatives has similar importance as positive images. On the other hand, Softmax only gained 1.9% by using hard negatives, which verifies our intuition that the triplet network is a better choice for utilizing hard negatives. The proposed framework fully utilizes combined human-machine efforts to enlarge the dataset as well as train a better model.

2.4.4 Visualization of Embedding

For qualitative evaluation purpose, in Figure 2.9, we show the 2-dimensional embedding of *flower-620* training set using PCA on features extracted from the trained Triplet-A model. Within the zoomed in regions, we can observe the effectiveness of our method in capturing high intra-class variances. For example, flowers from same category with different colors are mapped together in upper right and lower right regions.

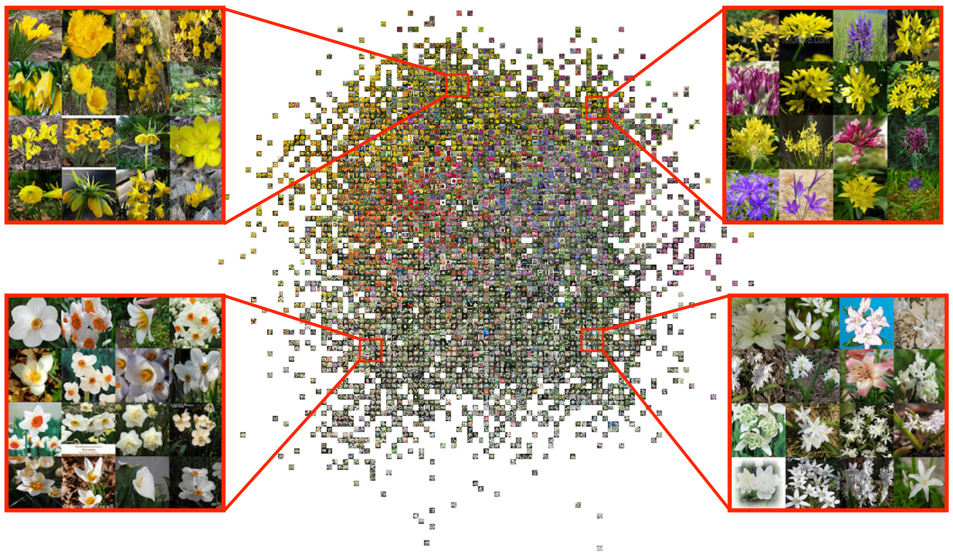


Figure 2.9: 2-D embedding of *flower-620* training set. We can observe that intra-class variance is captured in upper right and lower right regions.

2.5 Conclusion

In this chapter, we have presented an iterative framework for fine-grained visual categorization and dataset bootstrapping based on a novel deep metric learning approach with humans in the loop. Experimental results have validated the effectiveness of our framework.

We train our model mainly based on triplet information. Although we adopt an effective and efficient online triplet sampling strategy, the training process could still be slow, which is a limitation of our method. Some future directions could be discovering and labeling novel categories during dataset bootstrapping with a combined human-machine framework or incorporating more information (*e.g.*, hierarchical information, semantic similarity) into the triplet sampling strategy.

CHAPTER 3

KERNEL POOLING FOR CONVOLUTIONAL NEURAL NETWORKS

The idea of interactions between features has been used extensively as a higher order representation in learning tasks recently [112, 136, 14, 110]. The motivation behind is to make the subsequent linear classifier operates on higher dimensional feature map so that it becomes more discriminative. There are two ways in general to create higher order interactions. The most commonly used one is to *implicitly* map the feature via the kernel trick, like in the case of kernel SVM [162]. The disadvantages are twofold. The storage needed and the evaluation time are both proportional to the number of training data, which makes it inefficient on large datasets. In addition, the construction of the kernel makes it hard to use stochastic learning methods, including Stochastic Gradient Descent (SGD) in the training of CNNs. The other way is to *explicitly* map the feature vector into high dimensional space with products of features (monomials). The drawback of this method is obvious. If we want up to p^{th} order interactions on a d dimensional feature vector, the dimension of the explicit feature map will be $O(d^p)$, which makes it impractical to use in real world applications. A common way to address these issues is to compactly approximate either kernel functions [2, 172] or feature maps [85, 129, 6].

Before the remarkable success of using Convolutional Neural Networks (CNNs) on visual data [99, 147, 151, 66], low-level hand-crafted features (*e.g.*, SIFT [113], HOG [34], Gist [124]) combined with mid-level feature aggregation or pooling methods (*e.g.*, Bag-of-visual-words, Spatial Pyramid Matching [103], Sparse Coding [178], Fisher Vector [128]) were widely adopted as the standard scheme for feature extraction. When learning and applying the subsequent linear classifier on extracted features, kernel methods such as Gaussian RBF or exponential χ^2 kernel are often adopted to capture higher order information and make linear classifier more discriminative. Re-

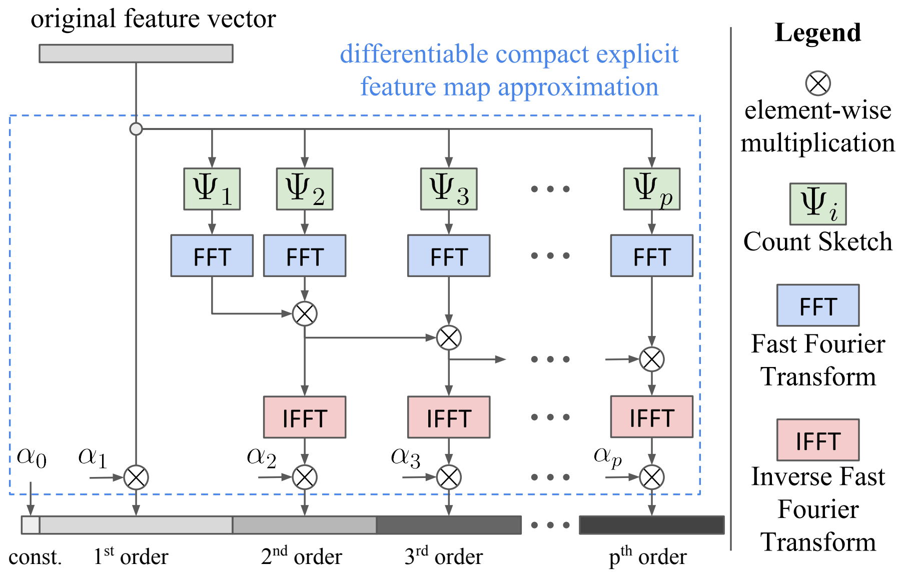


Figure 3.1: The proposed Kernel Pooling method. For a feature vector (i.e., the activation at a spatial location on the feature map, in the case of a CNN), we use Count Sketch [23] to generate a compact explicit feature map up to p^{th} order. After applying kernel pooling, the inner product between two features can capture high order feature interactions as in Equation 3.1. This makes the subsequent linear classifier highly discriminative. The proposed kernel pooling scheme is end-to-end trainable and the composition of the kernel can be learned through the update of coefficients $\{\alpha_i\}_{i=0}^{p}$. The vanilla compact bilinear pooling [49, 48] only use the 2nd order information as the feature vector.

cently, efforts in combining CNNs with 2nd order feature interactions, either by replacing hand-crafted features with CNN features [28] or jointly trained in an end-to-end fashion, yielded impressive performance gains on a wide range of visual tasks. Representative examples include fine-grained visual recognition [110, 49], visual question answering [48], texture representation and synthesis [50, 108], face recognition [27] and style transfer [51]. Notably, both Gao *et al.* [49] and Fukui *et al.* [48] used Tensor Sketch [129] to compactly compress the full bilinear vector by 2 orders of magnitude while preserve the same performance.

In this chapter, we propose a compact and differentiable way to generate explicit

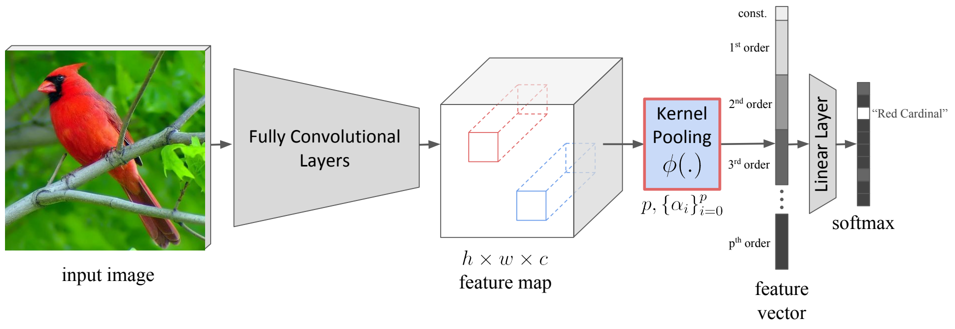


Figure 3.2: End-to-end training with the proposed pooling method. An input image is fed into a series of fully convolutional layers to get the output feature map of size $h \times w \times c$. For the c dimensional feature vector on every single spatial location (*e.g.*, the red or blue bar on the feature map), we apply the proposed kernel pooling method illustrated in Figure 3.1. The final feature vector is average pooled over all locations $h \times w$. Then a linear layer with softmax is used to do the classification. The kernel is defined by the order p and coefficients $\{\alpha_i\}_{i=0}^{p}$, which can be learned from data through back-propagation.

feature maps. We generalize the strategy used in [49, 48] to represent higher order feature interactions. For a feature vector $\mathbf{x}$ of dimension d, we generate its i^{th} order ($i \geq 2$) compact explicit feature map with Count Sketch [23] and circular convolution. In practice, people often operate circular convolution in frequency domain via Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT). It has been proven, both theoretically and practically in [129], that this method is able to compactly approximate polynomial kernels. As illustrated in Figure 3.1, with a stack of Count Sketch, element-wise multiplication, FFT and IFFT units, higher order information can be compactly preserved. The kernel pooling method is applied on every single spatial location on the feature map of a CNN. And the final feature vector is the result of global average pooling across all spatial locations.

Denote the proposed kernel pooling method as ϕ. Then for two feature vectors $\mathbf{x}$ and $\mathbf{y}$, the inner product between $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$ can approximate a kernel up to a certain order p as follows (see Section 3.2 for more details):

$$\phi(\mathbf{x})^\top \phi(\mathbf{y}) \approx \sum_{i=0}^{p} \alpha_i^2 (\mathbf{x}^\top \mathbf{y})^i \approx \mathcal{K}(\mathbf{x}, \mathbf{y}) \tag{3.1}$$

Through the introduction of kernel functions associated with Reproducing kernel Hilbert space, linear classifiers operate on high-dimensional Euclidean space become highly discriminative. Combine the proposed pooling method with a CNN, as shown in Figure 3.2, the model can be trained end-to-end via back-propagation of classification errors. The composition of the kernel, as determined by coefficients $\{\alpha_i\}_{i=0}^p$, can be either predefined to approximate a certain kernel like Gaussian RBF up to order p or learned from data.

To sum up, there are two main contributions in this chapter. Firstly, we propose a general kernel pooling method via compact explicit feature mapping. Using the linear classifier on the feature map is approximately same as applying the kernel trick. Secondly, the proposed kernel pooling is differentiable and can be combined with a CNN for joint optimization. The composition of the kernel can also be learned simultaneously during the training.

3.1 Related Work

The proposed kernel pooling method relies on the existing efforts on low dimensional compact approximation of explicit feature maps. Rahimi *et al.* [133] is one of the first work on using random features for Gaussian and Laplacian kernels. Later, the similar idea was generalized to other kernels such as Maji *et al.* [117] for the histogram intersection kernel and Vedaldi *et al.* [164] for χ^2 kernel. On the compact approximation of polynomial kernels, recent proposed Random Maclaurin by Kar *et al.* [85], Tensor Sketch by Pham *et al.* [129] and Subspace Embedding by Avron *et al.* [6] are the most noticeable representatives. There is also a line of work that tries to learn higher order interactions from the data through optimization [112, 136, 14]. We differ from these work by the combination of Convolutional Neural Networks (CNNs) in an end-to-end fashion. With the joint optimization, we can leverage the powerful off-the-shelf fully convolutional network architectures to learn better features directly from data.

Since the dimension of p^{th} order pooled feature grows exponentially with p, the use of $p > 2$ in real world applications is often limited. In the case of $p = 2$, the model is usually referred as Bilinear models, first introduced by Tenenbaum and Freeman [154]. Bilinear models demonstrate impressive performance on visual tasks applied on both hand-crafted features [21] and learned features [110, 27, 108, 51]. Recently, fueled by compact 2^{nd} order polynomial kernel approximation with Tensor Sketch [23, 129], same visual recognition performances can be preserved with much lower feature dimension [49] and new application on visual question answering is enabled [48]. We differ from these work by generalizing the compact representation from Bilinear models with 2^{nd} order polynomial kernel to p^{th} order Taylor series kernel defined in Section 3.2. The composition of the kernel can also be learned through the end-to-end training with a CNN (see Section 3.2.3).

3.2 Kernel Pooling

We define the concept of "pooling" as the process of encoding and aggregating feature maps into a global feature vector. The architecture of Convolutional Neural Networks (CNNs) can be regarded as fully convolutional layers followed by the subsequent pooling layers and a linear classifier. Table 3.1 summaries pooling strategies adopted in commonly used CNN architectures. Typically people use a stack of fully connected layer with Rectified Linear Unit (ReLU) as in the case of AlexNet [99] and VGG [147].

	AlexNet / VGG	Inception / ResNet	Bilinear	Compact Bilinear	**Ours**
Strategy	$\sigma(W_2\sigma(W_1X))$	$\frac{1}{hw}\sum_{i,j} X_{ij}$	$\frac{1}{hw}\sum_{i,j} X_{ij}X_{ij}^\top$	$\frac{1}{hw}\sum_{i,j} TS(X_{ij})$	$\frac{1}{hw}\sum_{i,j} \phi(X_{ij})$
Dimension	d	c	c^2	d	d
Time	$O(hwcd)$	$O(hwc)$	$O(hwc^2)$	$O(hw(c + d\log d))$	$O(hwp(c + d\log d))$
Space	$O(hwcd)$	0	0	$2c$	pc
Parameters	$O(hwcd)$	0	0	0	0 or p

Table 3.1: A summary of pooling strategies adopted in commonly used CNN architectures. X represent the feature map of size $h \times w \times c$, where h, w and c is the height, width and number of channels; d represents the pre-specified feature dimension for the subsequent linear classifier and p is the order we used for the proposed kernel pooling. $\sigma(.)$, $TS(.)$ and $\phi(.)$ denotes the ReLU unit, Tensor Sketch [129] and the proposed kernel pooling mehtod, respectively.

Fully connected layers often perform well in general but introduce heavy computation and large number of parameters, hence makes the network slow and easy to get overfit. The recently proposed Inception [151] and Residual Learning [66] only use global average pooling on the feature map. This strategy is more computationally efficient but it does not capture higher order feature interactions, which are believed crucial in many visual recognition tasks [110, 27, 108]. The bilinear models [21, 110] explicitly generate the c^2 dimensional feature map for 2nd order polynomial kernel, which is later compactly approximated in [49, 48] using Tensor Sketch [129]. In light of the success of Bilinear models, we propose an approach to go beyond Bilinear models and capture higher order feature interactions. We first define Tayler series kernel and show its explicit feature map can be compactly approximated. Then we demonstrate how to use the compact feature projection of Taylor series kernel to approximate commonly used kernels such as Gaussian RBF.

3.2.1 Explicit feature projection via Tensor product

Suppose the output feature map of a convolution layer is $X \in \mathbb{R}^{h \times w \times c}$ with height h, width w and number of channels c, we denote the c dimensional feature vector of a spatial location on X as $\mathbf{x} = [x_1, x_2, \ldots, x_c]^\top \in \mathbb{R}^c$.

The explicit feature projection $\phi(.)$ of a kernel function $\mathcal{K}(.,.)$ is defined by decomposing the the value of kernel function applied on two feature vectors $\mathbf{x}$ and $\mathbf{y}$ as the inner product between their feature maps:

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y}) \tag{3.2}$$

Commonly used kernel functions include polynomial kernels $(\mathbf{x}^\top \mathbf{y})^p$, Gaussian RBF kernel $\exp(-\gamma||\mathbf{x} - \mathbf{y}||^2)$, χ^2 kernel $\sum_{i=1}^{c} \frac{2x_i y_i}{x_i + y_i}$, *etc.* Notice that some of the kernels may correspond to an infinite dimensional feature projection (*e.g.*, Gaussian RBF).

We introduce the concept of Tensor product and then demonstrate it can be used to get the explicit feature projection of a specific type of kernel called *Taylor series kernel*.

First, we define the 2-level tensor product (*i.e.*, outer product $\mathbf{x}\mathbf{x}^\top$) of $\mathbf{x}$ as:

$$\mathbf{x}^{(2)} = \mathbf{x} \otimes \mathbf{x} = \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_c \\ x_2x_1 & x_2x_2 & \cdots & x_2x_c \\ \vdots & \vdots & \ddots & \vdots \\ x_cx_1 & x_cx_2 & \cdots & x_cx_c \end{bmatrix} \in \mathbb{R}^{c^2} \tag{3.3}$$

Similarly, the p-level tensor product for $p \geq 2$ is defined as:

$$\mathbf{x}^{(p)} = \mathbf{x} \underbrace{\otimes \cdots \otimes}_{\text{p times}} \mathbf{x} \in \mathbb{R}^{c^p} \tag{3.4}$$

We also have $\mathbf{x}^{(0)} = 1$ and $\mathbf{x}^{(1)} = \mathbf{x}$. Figure 3.3 illustrates the original feature vector $\mathbf{x}$ and its 2-level and 3-level tensor product $\mathbf{x}^{(2)}$ and $\mathbf{x}^{(3)}$. It has been shown in [142] that the p-level tensor product is the explicit feature projection of pth order Polynomial kernel:

$$(\mathbf{x}^\top \mathbf{y})^p = (\mathbf{x}^{(p)})^\top (\mathbf{y}^{(p)}) \tag{3.5}$$

We define the Taylor series kernel of order p as follows:

$$\mathcal{K}_{\text{Taylor}}(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{p} \alpha_i^2 (\mathbf{x}^\top \mathbf{y})^i \tag{3.6}$$

Since the non-negative linear combination of kernels is still a kernel [142], the Taylor series kernel is a valid kernel as it can be expressed as non-negative linear combinations of Polynomial kernels.

It is clear to see that the explicit feature projection of Taylor series kernel is given by:

$$\phi_{\text{Taylor}}(\mathbf{x}) = [\alpha_0 (\mathbf{x}^{(0)})^\top, \ldots, \alpha_p (\mathbf{x}^{(p)})^\top]^\top \tag{3.7}$$

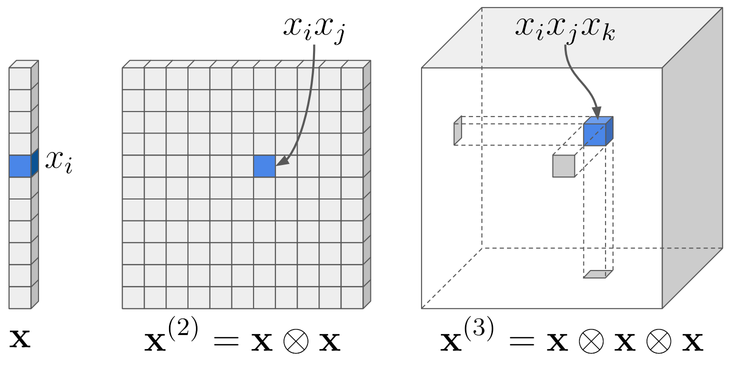

x_i
$x_i x_j$
$x_i x_j x_k$
$\mathbf{x}$
$\mathbf{x}^{(2)} = \mathbf{x} \otimes \mathbf{x}$
$\mathbf{x}^{(3)} = \mathbf{x} \otimes \mathbf{x} \otimes \mathbf{x}$

Figure 3.3: An illustration of tensor product. The p-level tensor product $\mathbf{x}^{(p)}$ of $\mathbf{x} \in \mathbb{R}^c$ is a c^p dimensional vector.

Composed by the concatenation of scaled tensor products $\{\alpha_i \mathbf{x}^{(i)}\}_{i=0}^{p}$, $\phi(\mathbf{x})$[1] is a long feature vector with dimension $O(c^p)$. Even in the case of $c = 512$ and $p = 3$, c^p is still larger than 10^8. Such a high dimension hinders its applications in any real world problems. Therefore, a compact approximation method is needed.

3.2.2 Compact approximation

The compact approximation method is differentiable and has good time and space complexity. There are several recently proposed work on kernel approximation with random feature projections [133, 85, 129, 6]. We build our approximation method on Tensor Sketching [129], because it consumes less time and space compared to [133, 85], and it is easier to implement compared to [6].

Algorithm 1: Count Sketch for Taylor series kernel

```
  Input: $\mathbf{x} \in \mathbb{R}^c,\ p,\ \{d_i\}_{i=2}^p,\ \{\alpha_i\}_{i=0}^p$
  Output: $\phi(\mathbf{x}) \in \mathbb{R}^d$, where $d = 1 + c + \sum_{i=2}^p d_i$, s.t.
          $\phi(\mathbf{x})^\top \phi(\mathbf{y}) \approx \mathcal{K}(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^p \alpha_i^2 (\mathbf{x}^\top \mathbf{y})^i$.
1 Initialization: $\phi(\mathbf{x}) \leftarrow [\alpha_0^2, \mathbf{x}^\top]^\top$, $\mathcal{P} \leftarrow 1$.
2 for $t \leftarrow 1$ to $p$ do
3     Generate 2 independent hash functions $h_t$ and $s_t$. The outputs of $h_t$ and $s_t$ are
        uniformly drawn from $\{1, 2, \ldots, d_t\}$ and $\{+1, -1\}$, respectively.
4     Calculate the Count Sketch of $\mathbf{x}$ as
        $C_t(\mathbf{x}) = [c_1, c_2, \ldots, c_{d_t}]^\top$, where $c_i = \sum_{i:h_t(i)=j} s_t(i)\mathbf{x}_i$.
5     $\mathcal{P} \leftarrow \mathcal{P} \circ \mathrm{FFT}(C_t(\mathbf{x}))$
6     if $t \geq 2$ then
7         $\phi(\mathbf{x}) \leftarrow$ concatenate $(\phi(\mathbf{x}),\ \mathrm{FFT}^{-1}(\mathcal{P}))$
8 return $\phi(\mathbf{x})$
```

Taylor series kernel

To compactly approximate the p-level tensor product $\mathbf{x}^{(p)}$, we define the Count Sketch [23] of $\mathbf{x}$ as:

$$C(\mathbf{x}) = [c_1, c_2, \ldots, c_d]^\top, \text{where } c_i = \sum_{i:h(i)=j} s(i)\mathbf{x}_i \tag{3.8}$$

The Count Sketch $C(\mathbf{x})$ is a d-dimensional vector calculated using 2 hash functions $h(.)$ and $s(.)$. Their outputs are uniformly drawn from $\{1, 2, \ldots, d\}$ and $\{+1, -1\}$, respectively. The p-level tensor product $\mathbf{x}^{(p)}$ can then be approximated as:

$$\tilde{\mathbf{x}}^{(p)} = \mathrm{FFT}^{-1}(\mathrm{FFT}(C_1(\mathbf{x})) \circ \cdots \circ \mathrm{FFT}(C_p(\mathbf{x}))) \tag{3.9}$$

where $C_i(\mathbf{x})$ is the Count Sketch calculated from $2i$ independent hash functions $h_1, h_2, \ldots, h_i$ and $s_1, s_2, \ldots, s_i$, $\circ$ denotes the element-wise multiplication, FFT and FFT^{-1} is the Fast Fourier Transform and its Inverse.

Combining Equation 3.7 and Equation 3.9, the feature map of a Taylor series kernel can be compactly approximated, as described in Algorithm 1. Inputs include the original

[1]For simplicity, unless otherwise specified, we will drop the subscript of $\mathcal{K}_{\text{Taylor}}$ and ϕ_{Taylor} in the remainder of the chapter.

feature vector **x**, the order p of the Taylor series kernel to be approximated, target feature dimensions $d_i (i \geq 2)$ we want to use for estimating $\mathbf{x}^{(i)}$ and its associated coefficient α_i. Compared with the explicit feature map in Equation 3.7, we reduce the feature dimension from exponential to linear. More specifically, from $\sum_{i=0}^{p} c^i$ to $d = 1 + c + \sum_{i=2}^{p} d_i$, where $d \ll c^i,\ \forall i \geq 2$.

It has been proved that $\tilde{\mathbf{x}}^{(p)}$ in Equation 3.9 is an unbiased feature map estimator for pth order Polynomial kernel. The relative estimation error can be bounded by Chebyshev's inequality (see Lemma 7 in [129] for the detailed proof). Similarly, the estimation error of using Algorithm 1 can be bounded as:

$$\mathbf{P}\Big[\Big|\phi(\mathbf{x})^\top \phi(\mathbf{y}) - \mathcal{K}(\mathbf{x}, \mathbf{y})\Big| \geq \epsilon \mathcal{K}(\mathbf{x}, \mathbf{y})\Big] \leq \frac{1}{d_{min}\epsilon^2} \Delta(p) \tag{3.10}$$

where $d_{min} = \min(d_2, \ldots, d_p)$ and

$$\Delta(p) = \begin{cases} 2(p-1), & \text{if } C = \pm 1 \\ \frac{2C^2(C^{2p}-1)}{C^2-1}, & \text{otherwise} \end{cases}$$

$C = \frac{1}{\cos\theta}$ is a constant that equals to the reciprocal of the cosine similarity between two feature vectors **x** and **y**. In our experience, we find higher dimensional feature (large d_{min}) gives better approximation, kernels with larger p introduce larger error, and the error bound also depends heavily on the angle between two feature vectors.

Gaussian RBF kernel

The Taylor expansion of Gaussian RBF kernel [130] can be expressed as:

$$\begin{aligned}\mathcal{K}_{RBF}(\mathbf{x},\mathbf{y}) &= \exp\big(-\gamma\|\mathbf{x}-\mathbf{y}\|^2\big)\\ &= \exp\big(-\gamma(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2-2\mathbf{x}^\top\mathbf{y})\big)\\ &= \beta\exp\big(2\gamma\mathbf{x}^\top\mathbf{y}\big)\\ &= \sum_{i=0}^{\infty}\beta\frac{(2\gamma)^i}{i!}(\mathbf{x}^\top\mathbf{y})^i \qquad (3.11)\end{aligned}$$

where $\beta = \exp\big(-\gamma(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\big)$ is a constant and $\beta = \exp(-2\gamma)$ if $\mathbf{x}$ and $\mathbf{y}$ are ℓ_2-normalized. Compared with Taylor series kernel in Equation 3.6, it is clear that Taylor series kernel can be used to approximate Gaussian RBF term by term up to order p by setting α_i^2 as $\beta\frac{(2\gamma)^i}{i!}$. Other kernels can also be approximated if they have a Taylor expansion in the similar form. Figure 3.4 illustrates the approximation of Gaussian RBF by Taylor series kernel with variant p. The approximation error depends on the inner product value $\mathbf{x}^\top\mathbf{y}$. In general, the closer the value is to 0, the smaller the approximation error. So we need to choose γ carefully based on $\mathbf{x}^\top\mathbf{y}$. With the proper choice of γ, using $p = 4$ would be sufficient to approximate Gaussian RBF. Experiments on kernel approximation error and the effect of γ will be discussed extensively in Section 3.3.2.

3.2.3 Learning kernel composition end-to-end

The proposed kernel pooling method in Algorithm 1 relies on simple computations with a set of fixed hash functions $\{h_t\}$ and $\{s_t\}$, FFT and FFT^{-1}, which are all differentiable. Combined with a CNN, the loss from the softmax layer can go through the proposed kernel pooling layer and be propagated back to the preceding fully convolution layers.

Instead of using fixed pre-defined coefficients to approximate a certain kernel such

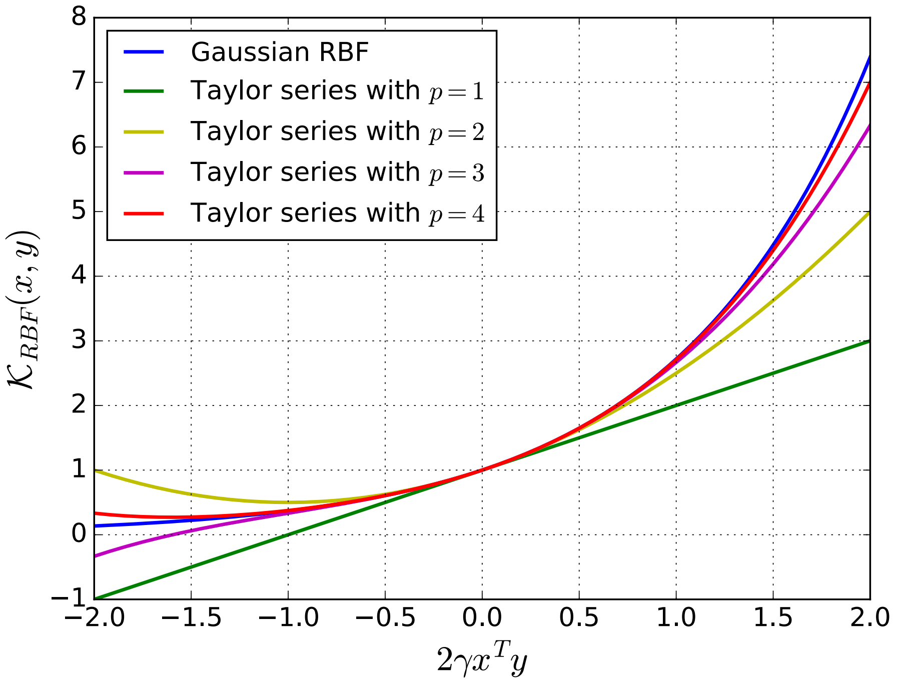


Figure 3.4: Approximating Gaussian RBF kernel by Taylor series kernel with variant p. Without loss of generality, we ignore the constant β when plotting. The approximation error depends on the inner product value $\mathbf{x}^\top\mathbf{y}$ and γ. With the proper choice of γ based on $\mathbf{x}^\top\mathbf{y}$, using $p = 4$ would be sufficient to approximate Gaussian RBF.

as Gaussian RBF, the composition of the kernel can be learned from data, as illustrated in Figure 3.5. Designing and choosing a good kernel is a challenging task because it is hard to probe the underlying distribution of high-dimensional features. Therefore, a kernel function is often chosen empirically or through cross-validation. By jointly learning the kernel composition together with CNN weights in an end-to-end fashion, we argue the learned kernel is more adaptive and suitable to the data we are working on.

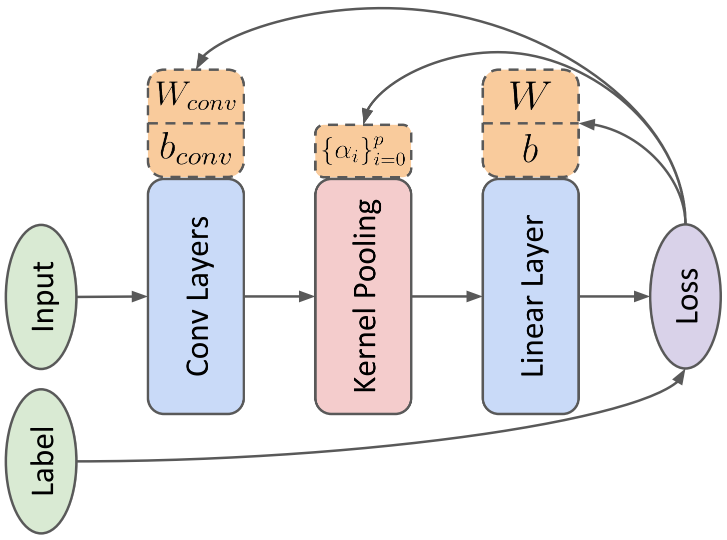


Figure 3.5: Learning kernel composition by end-to-end training with a CNN. The coefficients of the kernel are jointly learned together with weights of other CNN layers via back-propagation of the loss (denoted by outgoing arrows from “Loss”).

3.3 Experimental Evaluations

The proposed kernel pooling method is evaluated in terms of both kernel approximation error and visual recognition accuracy. Section 3.3.1 introduces experiment setup and baseline methods. Then, in Section 3.3.2, we run a comprehensive study of kernel approximation quality on CNN features. We also investigate the configuration such as the choice of feature dimension $\bar{d}$, kernel order p and γ. Section 3.3.3 is the major part of the experiment, in which we present extensive evaluations on various visual recognition tasks, including the recognition of bird [167], car [95], aircraft [118] and food [15]. The proposed kernel pooling method achieves state-of-the-art results on all datasets.

3.3.1 Experiment setup

We evaluate all pooling strategies listed in Table 3.1. For CNN architectures, we use VGG-16 [147] and ResNet-50 [66], both of which achieved state-of-the-art performance on ImageNet [36]. VGG-16 has 13 convolution with ReLU layers and 3 fully connected layers including the final linear layer with softmax for classification. ResNet-50 consists of 49 convolution layers followed by global average pooling and the final linear softmax layer. Both VGG and ResNet reduce the spatial resolution of the input image by a factor of $2^5 = 32$ during the convolution. In the case of Bilinear, Compact Bilinear and our model, we keep the fully convolutional part of the network and use the output feature map from the last convolution layer (*i.e.*, the feature vector $\mathbf{x}$ in Algorithm 1 corresponds to the activation at each spatial location of last layer's feature map). For standard pooling methods, we choose VGG-16 and ResNet-50 [66] as representatives for fully connected pooling and global average pooling, respectively. The performance of VGG-16 and ResNet-50 is reported by fine-tuning the entire network from ImageNet pre-trained weights.

Pooling methods

We compare the performance of kernel pooling methods with the following baselines:

VGG with fully connected pooling (**VGG**): This is the original VGG-16 network proposed in [147]. The architecture of VGG-16 is a generalization of the ground-breaking AlexNet [99]. In AlexNet, only one convolution layer is applied to the input image and the feature map of a specific spatial resolution. In VGG, however, more convolution layers (2 to 3) are applied for each spatial resolution, which achieved state-of-the-art performance on ImageNet Challenge 2014. Both AlexNet and VGG use the same fully connected pooling scheme (a stack of two fully connected with ReLU layers) for the subsequent softmax layer. Due to the fixed number of nodes designed in fully connected layers, VGG requires a fixed input image size of 224×224. For each of the dataset, we replace the last linear layer of VGG to match the number of categories and then fine-tune the whole network from ImageNet pre-trained weights.

Residual Learning with average pooling (**ResNet**): Although the fully connected layer works well in practice, it has several drawbacks including the heavy computation and large storage needed as well as the tend to overfit. Recently proposed deeper networks based on Inception module [151] and Residual module [66] use global average pooling after convolution layers for the subsequent linear classifier. The global average pooling is lightweight, capable of taking input of any size and parameter-free. However, it fails to capture nonlinear information in feature maps. We choose a strong baseline of fine-tuned ResNet as comparison.

Bilinear Pooling (**BP**): We apply full bilinear pooling on top of the conv_{5_3} feature map from VGG-16, which is same as the best-performed B-CNN [D, D] in [110]. The feature dimension of the bilinear vector is $d = 512 \times 512 \approx 260\text{K}$. We don't combine ResNet with bilinear pooling because ResNet has 2048 channels in the final feature map. The brute force bilinear vector has the dimension of $2048 \times 2048 \approx 4.2\text{M}$, which is too large to use in practice.

Compact Bilinear Pooling (**CBP**): We use Tensor Sketch with fixed hash functions to approximate bilinear vector on the feature map of VGG-16 and ResNet-50. Whereas the original paper [49] only used VGG-16. Typically, compact bilinear pooling can achieve same performance as full bilinear pooling with $d \geq 8192$, reducing the original feature dimension by orders of magnitude. For a fair comparison, we set the feature dimension in CBP to be the same as our kernel pooling method in all experiments.

The proposed Kernel Pooling (**KP**): We evaluate the proposed kernel pooling method in the same context as BP and CBP. For the activation $\mathbf{x}$ at each spatial location on the feature map, we apply Algorithm 1 to get the compact feature map $\phi(\mathbf{x})$. Same as BP and CBP, the final feature vector is average pooled across all the spatial locations. The composition of the kernel is evaluated with learned coefficients via back-propagation. The choice of kernel order p, feature dimension d and γ will be discussed in Section 3.3.2.

Implementation

Our implementation follows the commonly used practice in [99, 147, 110, 49]. We have two image input sizes: 224×224 and 448×448. For each image input size $S \times S$, we first subtract it with the pixel-wise image mean, and we resize the original image so that its shorter side is S while keeping its aspect ratio. Then we crop a $S \times S$ square image from the original image. During training, a random square image is cropped. Both the original crop and its horizontal flip are utilized for data augmentation. During inference, the center image is cropped. We pass the original crop and its horizontal flip to the CNN independently. The average of their classification scores is our final classification score.

We follow the post-processing steps in [110, 49] to the feature vector $\mathbf{y}$ before the linear classifier, because the experiments show that it improves fine-grained recognition performance. We apply element-wise signed square root: $\mathbf{y} \leftarrow sign(\mathbf{y})\sqrt{|\mathbf{y}|}$ followed by ℓ_2 normalization: $\mathbf{y} \leftarrow \mathbf{y}/||\mathbf{y}||$ on the compact feature $\mathbf{y}$ vector.

For the sake of faster convergence and better performance, we use pre-trained weights for the neural network. The intial weights of the convolutional layers are pre-trained on ImageNet classification dataset, and the initial weights of the final linear classifier is obtained by training a logistic regression classifier on the compact kernel pooling of pre-trained CNN features. We start the fine-tuing with 10x smaller learning rate (*i.e.* 0.001 for VGG and 0.01 for ResNet) and divide it by 10 after every 30 epochs. We use a momentum of 0.9 and a weight decay of 0.0005 for VGG and 0.0001 for ResNet. The training usually converges at around 50 epochs. The model diverges due to large gradients sometimes. Therefore, gradient clipping [127] is applied to ensure all gradients fall in the range between -1 and $+1$.

We use Tensorflow [1] to implement and train all the models. On a single NVIDIA Tesla K40 GPU, the forward and backward time of both VGG-16 and ResNet-50 with kernel pooling is about 500ms on a 448×448 image and 100ms on a 224×224 image. Kernel pooling requires around 50ms with $d = 4096$ and $p = 4$.

3.3.2 Kernel approximation and configurations

This subsection presents the experiments on kernel approximation error using Algorithm 1 on CNN features. Using VGG-16 trained on ImageNet, we extract conv_{5_3} feature maps on the training set of CUB-200-2011 [167], with input size of 224×224. For each spatial location in the feature map, the feature is a $c = 512$ dimensional vector. Without loss of generality, we use the same feature pooling dimension $\bar{d}$ for each order in kernel pooling (*i.e.*, $d_i = \bar{d}$ for $i \geq 2$). Therefore, the final feature dimension is $d = 1 + c + \sum_{i=2}^{p} \bar{d} = 513 + (p-1)\bar{d}$. Figure 3.6 shows the relative approximation error of Gaussian RBF kernel in log scale, with variant feature pooling dimension $\bar{d}$, order p and γ. The relative approximation error between two feature vector $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$\epsilon = \frac{|\phi(\mathbf{x})^\top \phi(\mathbf{y}) - \mathcal{K}_{RBF}(\mathbf{x}, \mathbf{y})|}{\mathcal{K}_{RBF}(\mathbf{x}, \mathbf{y})} \tag{3.12}$$

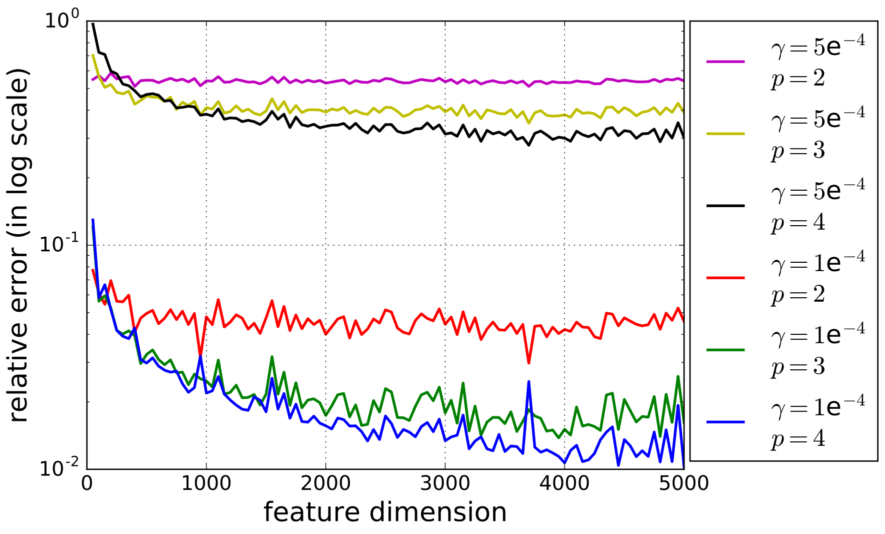


Figure 3.6: Relative approximation error for Gaussian RBF kernel applied on CNN features with variant kernel configurations.

We compare kernel pooing with the feature dimension $\bar{d}$ from 50 to 5000 with the step of 50. Each data point is the averaged error on 100K randomly selected feature pairs.

From Figure 3.6, we have the following observations: higher feature pooling dimension gives better approximation in general; approximation error also goes down with increasing order p; γ plays a key role in the approximation error. The above findings verify the insights from Equation 3.10. In Figure 3.4 we can see that with sufficient feature dimension and order as well as a proper γ, we can achieve close to 1% relative error. In light of this, we use $\bar{d} = 4096$ and $p = 4$ for all the following experiments. The output vector has a dimension of $d = 1 + 512 + 3 \times 4096 = 12801$ for VGG, and $1 + 2048 + 3 \times 4096 = 14337$ for ResNet. The hyper-parameter γ is set as the reciprocal of the mean of inner products between feature vectors in the training set to ensure that $\gamma \mathbf{x}^\top \mathbf{y}$ is small on average and we can get a good kernel approximation.

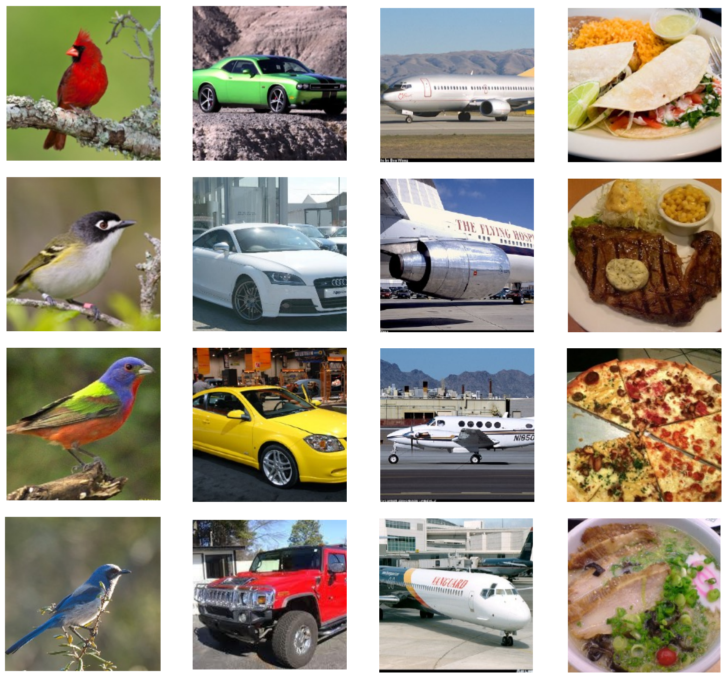


Figure 3.7: Images we used for visual recognition. From left to right, each column contains examples from CUB-200 Bird [167], Stanford Car [95], Aircraft [118] and Food-101 [15].

3.3.3 Visual recognition

We evaluate on the following visual recognition tasks.

Bird species recognition: We use CUB-200 dataset [167] for this task. The dataset consists of $11,788$ images from 200 bird species. Each category has around 30 images for both training and testing.

Car make, model, year classification: The Stanford Car dataset [95] is used for this task. It has $16,185$ images of 196 classes with car make, model and year.

Aircraft classification: The fine-grained aircraft dataset [118] was first introduced in FGComp 2013 challenge, which contains 100 categories and each has 100 images.

Dataset	CNN	Original	BP [110]	CBP [49]	KP	**Others**	
CUB-200	VGG-16	73.1*	84.1	84.3	**86.2**	82.0	84.1
	ResNet-50	78.4	N/A	81.6	84.7	[96]	[78]
Stanford Car	VGG-16	79.8*	91.3	91.2	**92.4**	**92.6**	82.7
	ResNet-50	84.7	N/A	88.6	91.1	[96]	[61]
Aircraft	VGG-16	74.1*	84.1	84.1	**86.9**	80.7	
	ResNet-50	79.2	N/A	81.6	85.7	[61]	
Food-101	VGG-16	81.2	82.4	82.4	84.2	50.76	
	ResNet-50	82.1	N/A	83.2	**85.5**	[15]	

Table 3.2: Performance comparisons among all baselines, where KP is the proposed kernel pooling method with learned coefficients. Following the standard experimental setup, we use the input size of 448×448 for CUB, Stanford Car and Aircraft datasets except the original VGG-16 (marked by an asterisk *), which requires a fixed input size of 224×224. For Food-101, we use the input size of 224×224 for all the baselines.

Food recognition: For this task we use Food-101 dataset [15], which is by far the largest publicly available food recognition dataset to the best of our knowledge. This is a large-scale dataset with $101,000$ images and 1000 images per each category. This dataset is challenging as the training images are noisy and the background is not clean.

Sample images for each task are shown in Figure 3.7. Performance comparison with all the baselines and state-of-the-art methods is presented in Table 3.2. The proposed Kernel Pooling with learned coefficients outperforms all other baselines by a large margin (around 1-3%) on all the datasets.

3.3.4 Discussion

In this subsection, we discuss the relative importance of higher order information for different CNN architectures. We examined learned kernel coefficients on CUB dataset with kernel pooling on VGG and ResNet. We found that high order feature interactions, especially 2nd and 3rd order, are weighted more in VGG compared with ResNet. In ResNet, there is no obvious distinction among first 3 orders. We believe this is due to the difference of the underlying network architectures.

One reason might be that in VGG, the non-linear feature interactions are mainly captured by fully-connected layers. So removing the fully-connected layers significantly degrade the original 1st order feature. Since ResNet only use a global average pooling layer and has a very large receptive field, the features at different locations of the feature map is encouraged to represent similar information. Together with the residual module and a much deeper convolutional architecture, the output convolution feature could implicitly capture more information than VGG. In our experiments, we find that the performance of both VGG-16 and ResNet-50 can be improved when the proposed kernel pooling method is utilized. These experiments verify the effectiveness of using high-order feature interactions in the context of CNN.

3.4 Conclusion

In this chapter, we have introduced a novel deep kernel pooling method as a high-order representation for visual recognition. The proposed method captures high-order and non-linear feature interactions via compact explicit feature mapping. The approximated representation is fully differentiable, thus the kernel composition can be learned together with a CNN in an end-to-end manner. Extensive experiments demonstrate that deep kernel pooling method achieves state-of-the-art performance on various fine-grained recognition tasks.

CHAPTER 4
LARGE SCALE FINE-GRAINED CATEGORIZATION AND DOMAIN-SPECIFIC TRANSFER LEARNING

Fine-grained visual categorization (FGVC) aims to distinguish subordinate visual categories. Examples include recognizing natural categories such as species of birds [167, 159], dogs [89] and plants [123, 170]; or man-made categories such as car make & model [95, 179]. A successful FGVC model should be able to discriminate categories with subtle differences, which presents formidable challenges for the model design yet also provides insights to a wide range of applications such as rich image captioning [5], image generation [9], and machine teaching [82, 115].

Recent advances on Convolutional Neural Networks (CNNs) for visual recognition [99, 147, 151, 66] have fueled remarkable progress on FGVC [110, 33, 191]. In general, to achieve good performance with CNNs, one needs to train networks with vast amounts of supervised data. However, collecting a labeled fine-grained dataset often requires expert-level domain knowledge and therefore is difficult to scale. As a result, commonly used FGVC datasets [167, 89, 95] are relatively small, typically containing around 10k of labeled training images. In such a scenario, fine-tuning the networks that are pre-trained on large scale datasets such as ImageNet [36] is often adopted.

This common setup poses two questions: 1) What are the important factors to achieve good performance on large scale FGVC? Although other large scale generic visual datasets like ImageNet contain some fine-grained categories, their images are usually iconic web images that contain objects in the center with similar scale and simple backgrounds. With the limited availability of large scale FGVC datasets, how to design models that perform well on large scale non-iconic images with fine-grained categories remains an underdeveloped area. 2) How does one effectively conduct transfer learning, by first training the network on a large scale dataset and then fine-tuning it on domain-specific fine-grained datasets? Modern FGVC methods overwhelmingly use ImageNet pre-trained networks for fine-tuning. Given the fact that the target fine-grained domain is known, can we do better than ImageNet?

This chapter aims to answer the two aforementioned problems, with the recently introduced iNaturalist 2017 large scale fine-grained dataset (iNat) [160]. iNat contains 675,170 training and validation images from 5,089 fine-grained categories. All images were captured in natural conditions with varied object scales and backgrounds. Therefore, iNat offers a great opportunity to investigate key factors behind training CNNs for large scale FGVC. In addition, along with ImageNet, iNat enables us to study the transfer of knowledge learned on large scale datasets to small scale fine-grained domains.

In this chapter, we first propose a training scheme for large scale fine-grained categorization, achieving top performance on iNat. Unlike ImageNet, images in iNat have much higher resolutions and a wide range of object scales. We show in Section 4.2.1 that performance on iNat can be improved significantly with higher input image resolution. Another issue we address in this chapter is the long-tailed distribution, where a few categories have most of the images [194, 161]. To deal with this, we present a simple yet effective approach. The idea is to learn good features from a large amount of training data and then fine-tune on a more evenly-distributed subset to balance the network's efforts among all categories and transfer the learned features. Our experimental results, shown in Section 4.2.2, reveal that we can greatly improve the under-represented categories and achieve better overall performance.

Secondly, we study how to transfer from knowledge learned on large scale datasets to small scale fine-grained domains. Datasets are often biased in terms of their content and style [157]. On CUB200 Birds [167], iNat pre-trained networks perform much better

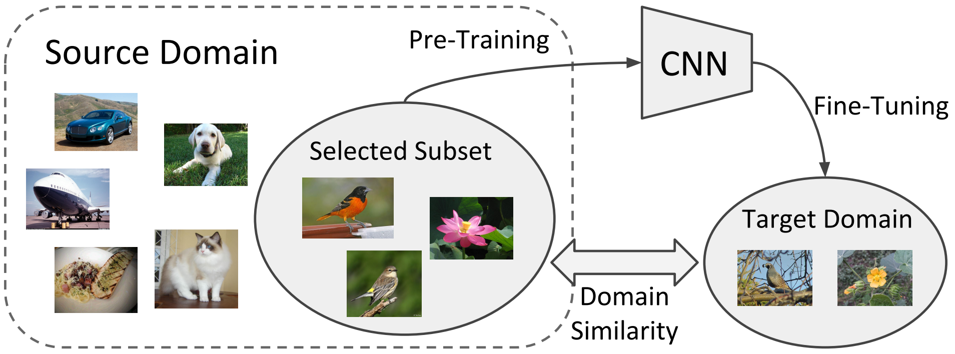


Figure 4.1: Overview of the proposed transfer learning scheme. Given the target domain of interest, we pre-train a CNN on the selected subset from the source domain based on the proposed domain similarity measure, and then fine-tune on the target domain.

than ImageNet pre-trained ones; whereas on Stanford-Dogs [89], ImageNet pre-trained networks yield better performance. This is because there are more visually similar bird categories in iNat and dog categories in ImageNet. In light of this, we propose a novel way to measure the visual similarity between source and target domains based on image-level visual similarity with Earth Mover's Distance. By fine-tuning the networks trained on selected subsets based on our proposed domain similarity, we achieve better transfer learning than ImageNet pre-training and state-of-the-art results on commonly used fine-grained datasets. Figure 4.1 gives an overview of the proposed training scheme.

We believe our study on large scale FGVC and domain-specific transfer learning could offer useful guidelines for researchers working on similar problems.

4.1 Related Work

Fine-Grained Visual Categorization (FGVC). Recent FGVC methods typically incorporate useful fine-grained information into a CNN and train the network end-to-end. Notably, second order bilinear feature interactions was shown to be very effective [110]. This idea was later extended to compact bilinear pooling [49], and then higher order interactions [33, 19, 146]. To capture subtle visual differences, visual attention [174, 47, 191] and deep metric learning [143, 32] are often used. Beyond pixels, we also leverage other information including parts [187, 16, 188], attributes [163, 55], human interactions [17, 37] and text descriptions [134, 68]. To deal with the lack of training data in FGVC, additional web images can be collected to augment the original dataset [32, 97, 177, 55]. Our approach differs from them by transferring the pre-trained network on existing large scale datasets without collecting new data.

Using high-resolution images for FGVC has became increasingly popular [78, 110]. There is also a similar trend in ImageNet visual recognition, from originally 224×224 in AlexNet [99] to 331×331 in recently proposed NASNet [195]. However, no prior work has systematically studied the effect of image resolution on large scale fine-grained datasets as we do in this chapter.

How to deal with long-tailed distribution is an important problem in real world data [194, 161]. However, it is a rather unexplored area mainly because commonly used benchmark datasets are pre-processed to be close-to evenly distributed [36, 107]. Van Horn *et al.* [161] pointed out that the performance of tail categories are much poorer than head categories that have enough training data. We present a simple two-step training scheme to deal with long-tailed distribution that works well in practice.

Transfer Learning. Convolutional Neural Networks (CNNs) trained on ImageNet have been widely used for transfer learning, either by directly using the pre-trained network as a feature extractor [144, 40, 192], or fine-tuning the network [57, 125]. Due to the remarkable success of using pre-trained CNNs for transfer learning, extensive efforts have been made on understanding transfer learning [181, 7, 76, 148]. In particular, some prior work loosely demonstrated the connection between transfer learning and domain similarity. For example, transfer learning between two random splits is easier than natural / man-made object splits in ImageNet [181]; manually adding 512 additional relevant categories from all available classes improve upon the commonly used 1000 ImageNet classes on PASCAL VOC [44]; transferring from a combined ImageNet and Places dataset yields better results on a list of visual recognition tasks [192]. Azizpour *et al.* [7] conducted a useful study on a list of transfer learning tasks that have different similarity with the original ImageNet classification task (*e.g.*, image classification is considered to be more similar than instance retrieval, *etc.*). Our major differences between their work are two-fold: Firstly, we provide a way to quantify the similarity between source and target domain and then choose a more similar subset from source domain for better transfer learning. Secondly, they all use pre-trained CNNs as feature extractors and only train either the last layer or use a linear SVM on the extracted features, whereas we fine-tune all the layers of the network.

4.2 Large Scale Fine-Grained Categorization

In this section, we present our training scheme that achieves top performance on the challenging iNaturalist 2017 dataset, especially focusing on using higher image resolution and dealing with long-tailed distribution.

4.2.1 The Effect of Image Resolution

When training a CNN, for the ease of network design and training in batches, the input image is usually pre-processed to be square with a certain size. Each network architecture usually has a default input size. For example, AlexNet [99] and VGGNet [147] take

Input Resolution	Networks
224×224	AlexNet [99], VGGNet [147], ResNet [66]
299×299	Inception [151, 152, 150]
320×320	ResNetv2 [67], ResNeXt [175], SENet [71]
331×331	NASNet [195]

Table 4.1: Default input image resolution for different networks. There is a trend of using input images with higher resolution for modern networks.

the default input size of 224×224 and this default input size cannot be easily changed because the fully-connected layer after convolutions requires a fixed size feature map. More recent networks including ResNet [66] and Inception [151, 152, 150] are fully convolutional, with a global average pooling layer right after convolutions. This design enables the network to take input images with arbitrary sizes. Images with different resolution induce feature maps of different down-sampled sizes within the network.

Input images with higher resolutions usually contain richer information and subtle details that are important to visual recognition, especially for FGVC. Therefore, in general, higher resolution input image yields better performance. For networks optimized on ImageNet, there is a trend of using input images with higher resolution for modern networks: from originally 224×224 in AlexNet [99] to 331×331 in recently proposed NASNet [195], as shown in Table 4.1. However, most images from ImageNet have a resolution of 500×375 and contain objects of similar scales, limiting the benefits we can get from using higher resolution inputs. We explore the effect of using a wide range of input image sizes from 299×299 to 560×560 in iNat dataset, showing greatly improved performance with higher resolution inputs.

4.2.2 Long-Tailed Distribution

The statistics of real world images is long-tailed: a few categories are highly representative and have most of the images, whereas most categories are observed rarely with only a few images [194, 161]. This is in stark contrast to the even image distribution in popular benchmark datasets such as ImageNet [36], COCO [107] and CUB200 [167].

With highly imbalanced numbers of images across categories in the iNaturalist dataset [160], we observe poor performance on underrepresented tail categories. We argue that this is mainly caused by two reasons: 1) The lack of training data. Around 1,500 fine-grained categories in iNat training set have fewer than 30 images. 2) The extreme class imbalance encountered during training: the ratio between the number of images in the largest class and the smallest one is about 435. Without any re-sampling of the training images or re-weighting of the loss, categories with more images in the head will dominate those in the tail. Since there is very little we can do for the first issue of lack of training data, we propose a simple and effective way to address the second issue of the class imbalance.

The proposed training scheme has two stages. In the first stage, we train the network as usual on the original imbalanced dataset. With large number of training data from all categories, the network learns good feature representations. Then, in the second stage, we fine-tune the network on a subset containing more balanced data with a small learning rate. The idea is to slowly transfer the learned feature and let the network re-balance among all categories. Figure 4.2 shows the distribution of image frequency in iNat training set that we trained on in the first stage and the subset we used in the second stage, respectively. Experiments in Section 4.4.2 verify that the proposed strategy yields improved overall performance, especially for underrepresented tail categories.

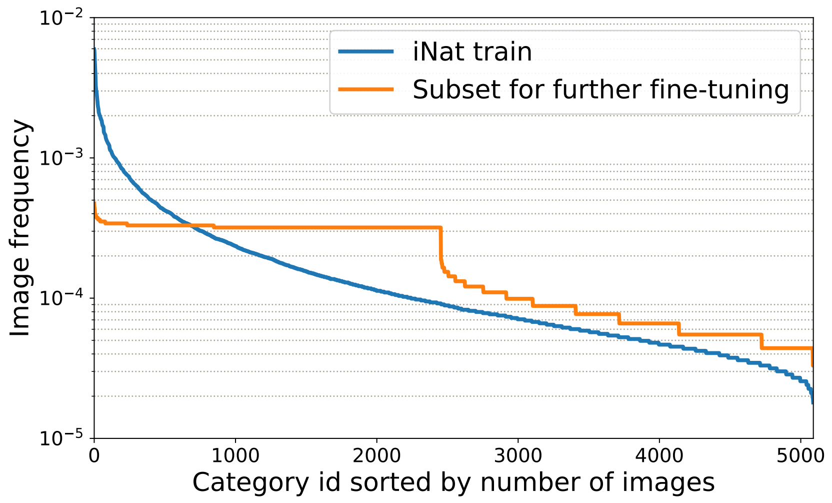


Figure 4.2: The distribution of image frequency of each category in the whole training set we used in the first stage training and the selected subset we used in the second stage fine-tuning.

4.3 Transfer Learning

This section describes transfer learning from the networks trained on large scale datasets to small scale fine-grained datasets. We introduce a way to measure visual similarity between two domains and then show how to select a subset from source domain given the target domain.

4.3.1 Domain Similarity

Suppose we have a source domain $\mathcal{S}$ and a target domain $\mathcal{T}$. We define the distance between two images $s \in \mathcal{S}$ and $t \in \mathcal{T}$ as the Euclidean distance between their feature representations:

$$d(s, t) = ||g(s) - g(t)|| \tag{4.1}$$

where $g(\cdot)$ denotes a feature extractor for an image. To better capture the image similarity, the feature extractor $g(\cdot)$ needs to be capable of extracting high-level information from images in a generic, unbiased manner. Therefore, in our experiments, we use $g(\cdot)$ as the features extracted from the penultimate layer of a ResNet-101 trained on the large scale JFT dataset [148].

In general, using more images yields better transfer learning performance. For the sake of simplicity, in this study we ignore the effect of domain scale (number of images). Specifically, we normalize the number of images in both source and target domain. As studied by Chen *et al.* [148], transfer learning performance increases logarithmically with the amount of training data. This suggests that the performance gain in transfer learning resulting from the use of more training data would be insignificant when we already have a large enough dataset (*e.g.*, ImageNet). Therefore, ignoring the domain scale is a reasonable assumption that simplifies the problem. Our definition of domain similarity can be generalized to take domain scale into account by adding a scale factor, but we found ignoring the domain scale already works well in practice.

Under this assumption, transfer learning can be viewed as moving a set of images from the source domain $\mathcal{S}$ to the target domain $\mathcal{T}$. The work needed to be done by moving an image to another can be defined as their image distance in Equation 4.1. Then the distance between two domains can be defined as the least amount of total work needed. This definition of domain similarity can be calculated by the Earth Mover's Distance (EMD) [132, 139].

To make the computations more tractable, we further make an additional simplification to represent all image features in a category by the mean of their features. Formally, we denote source domain as $\mathcal{S} = \{(s_i, w_{s_i})\}_{i=1}^{m}$ and target domain as $\mathcal{T} = \{(t_j, w_{t_j})\}_{j=1}^{n}$, where s_i is the i-th category in $\mathcal{S}$ and w_{s_i} is the normalized number of images in that cat-

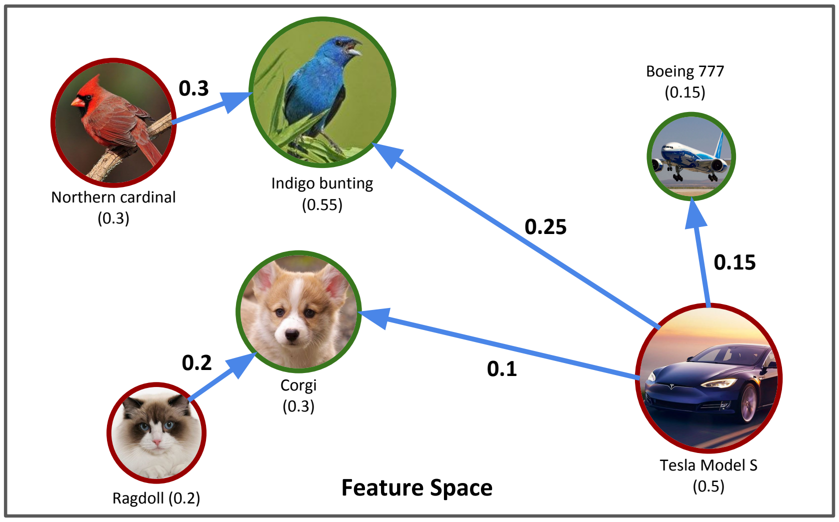


Figure 4.3: The proposed domain similarity calculated by Earth Mover's Distance (EMD). Categories in source domain and target domain are represented by red and green circles. The size of the circle denotes the normalized number of images in that category. Blue arrows represent flows from source to target domain by solving EMD.

egory; similarly for t_j and w_{t_j} in $\mathcal{T}$. m and n are the total number of categories in source domain $\mathcal{S}$ and target domain $\mathcal{T}$, respectively. Since we normalize the number of images, we have $\sum_{i=1}^{m} w_{s_i} = \sum_{j=1}^{n} w_{t_j} = 1$. $g(s_i)$ denotes the mean of image features in category i from source domain, similarly for $g(t_j)$ in target domain. Using the defined notations, the distance between $\mathcal{S}$ and $\mathcal{T}$ is defined as their Earth Mover's Distance (EMD):

$$d(\mathcal{S}, \mathcal{T}) = \text{EMD}(\mathcal{S}, \mathcal{T}) = \frac{\sum_{i=1,j=1}^{m,n} f_{i,j} d_{i,j}}{\sum_{i=1,j=1}^{m,n} f_{i,j}} \tag{4.2}$$

where $d_{i,j} = \|g(s_i) - g(t_j)\|$ and the optimal flow $f_{i,j}$ corresponds to the least amount of total work by solving the EMD optimization problem. Finally, the domain similarity is defined as:

$$\text{sim}(\mathcal{S}, \mathcal{T}) = e^{-\gamma d(\mathcal{S}, \mathcal{T})} \tag{4.3}$$

where γ is set to 0.01 in all experiments. Figure 4.3 illustrates calculating the proposed domain similarity by EMD.

4.3.2 Source Domain Selection

With the defined domain similarity in Equation 4.2, we are able to select a subset from source domain that is more similar to target domains. We use greedy selection strategy to incrementally include the most similar category in the source domain. That is, for each category s_i in source domain $\mathcal{S}$, we calculate its domain similarity with target domain by $\text{sim}(\{(s_i, 1)\}, \mathcal{T})$ as defined in Equation 4.3. Then top k categories with highest domain similarities will be selected. Notice that although this greedy way of selection has no guarantee on the optimality of the selected subset of size k in terms of domain similarity, we found this simple strategy works well in practice.

4.4 Experiments

The proposed training scheme for large scale FGVC is evaluated on the recently proposed iNaturalist 2017 dataset (iNat) [160]. We also evaluate the effectiveness of the our proposed transfer learning by using ImageNet and iNat as source domains, and 7 fine-grained categorization datasets as target domains. Section 4.4.1 introduces experiment setup. Experiment results on iNat and transfer learning are presented in Section 4.2 and Section 4.4.3, respectively.

4.4.1 Experiment setup

Datasets

iNaturalist. The iNatrualist 2017 dataset (iNat) [160] contains 675,170 training and validation images from 5,089 natural fine-grained categories. Those categories belong to

FGVC Dataset	# class	# train	# val
Flowers-102 [123]	102	2,040	6,149
CUB200 Birds [167]	200	5,994	5,794
Aircraft [118]	100	6,667	3,333
Stanford Cars [95]	196	8,144	8,041
Stanford Dogs [89]	120	12,000	8,580
NABirds [159]	555	23,929	24,633
Food101 [15]	101	75,750	25,250

Table 4.2: We use 7 fine-grained visual categorization datasets to evaluate the proposed transfer learning method.

13 super-categories including Plantae (Plant), Insecta (Insect), Aves (Bird), Mammalia (Mammal), and so on. The iNat dataset is highly imbalanced with dramatically different number of images per category. For example, the largest super-category "Plantae (Plant)" has 196,613 images from 2,101 categories; whereas the smallest super-category "Protozoa" only has 381 images from 4 categories. We combine the original split of training set and 90% of the validation set as our training set (iNat train), and use the rest of 10% validation set as our mini validation set (iNat minival), resulting in total of 665,473 training and 9,697 validation images.

ImageNet. We use the ILSVRC 2012 [140] splits of 1,281,167 training (ImageNet train) and 50,000 validation (ImageNet val) images from 1,000 classes.

Fine-Grained Visual Categorization. We evaluate our transfer learning approach on 7 fine-grained visual categorization datasets as target domains, which cover a wide range of FGVC tasks including natural categories like bird and flower and man-made categories such as aircraft. Table 4.2 summarizes number of categories, together with number of images in their original training and validation splits.

Network Architectures

We use 3 types of network architectures: ResNet [66, 67], Inception [151, 152, 150] and SENet [71].

Residual Network (ResNet). Originally introduced by He *et al.* [66], networks with residual connections greatly reduced the optimization difficulties and enabled the training of much deeper networks. ResNets were later improved by pre-activation that uses identity mapping as the skip connection between residual modules [67]. We used the latest version of ResNets [67] with 50, 101 and 152 layers.

Inception. The Inception module was firstly proposed by Szegedy *et al.* in GoogleNet [151] that was designed to be very efficient in terms of parameters and computations, while achieving state-of-the-art performance. Inception module was then further optimized by using Batch Normalization [77], factorized convolution [152, 150] and residual connections [150] as introduced in [66]. We use Inception-v3 [152], Inception-v4 and Inception-ResNet-v2 [150] as representatives for Inception networks in our experiments.

Squeeze-and-Excitation (SE). Recently proposed by Hu *et al.* [71], Sequeeze-and-Excitation (SE) modules achieved the best performance in ILSVRC 2017 [140]. SE module squeezes responses from a feature map by spatial average pooling and then learns to re-scale each channel of the feature map. Due to its simplicity in design, SE module can be used in almost any modern networks to boost the performance with little additional overhead. We use Inception-v3 SE and Inception-ResNet-v2 SE as baselines.

For all network architectures, we follow strictly their original design but with the last linear classification layer replaced to match the number of categories in our datasets.

Implementation

We used open-source Tensorflow [1] to implement and train all the models asynchronously on multiple NVIDIA Tesla K80 GPUs. During training, the input image was randomly cropped from the original image and re-sized to the target input size with scale and aspect ratio augmentation [151]. We trained all networks using the RMSProp optimizer with momentum of 0.9, and the batch size of 32. The initial learning rate was set to 0.045, with exponential decay of 0.94 after every 2 epochs, same as [151]; for fine-tuning in transfer learning, the initial learning rate is lowered to 0.0045 with the learning rate decay of 0.94 after every 4 epochs. We also used label smoothing as introduced in [152]. During inference, the original image is center cropped and re-sized to the target input size.

4.4.2 Large Scale Fine-Grained Visual Recognition

To verify the proposed learning scheme for large scale fine-grained categorization, we conduct extensive experiments on iNaturalist 2017 dataset. For better performance, we fine-tune from ImageNet pre-trained networks. If training from scratch on iNat, the top-5 error rate is $\approx 1\%$ worse.

We train Inception-v3 with 3 different input resolutions (299, 448 and 560). The effect of image resolution is presented in Table 4.3. From the table, we can see that using higher input resolutions achieve better performance on iNat.

The evaluation of our proposed fine-tuning scheme for dealing with long-tailed distribution is presented in Figure 4.4. Better performance can be obtained by further fine-tuning on a more balanced subset with small learning rate (10^{-6} in our experi-

	Inc-v3 299	**Inc-v3 448**	**Inc-v3 560**
Top-1 (%)	29.93	26.51	25.37
Top-5 (%)	10.61	9.02	8.56

Table 4.3: Top-5 error rate on iNat minival using Inception-v3 with various input sizes. Higher input size yield better performance.

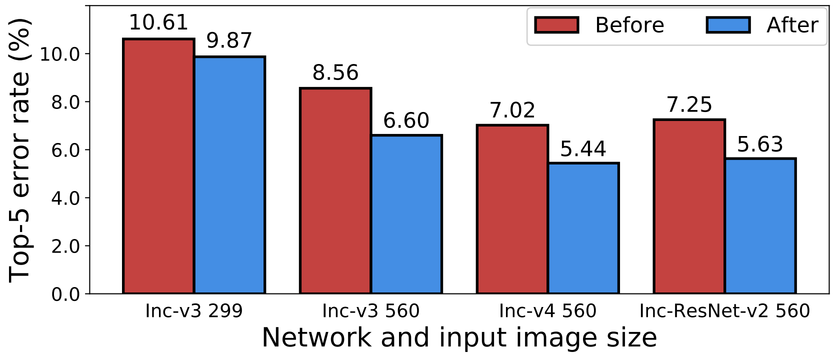


Figure 4.4: Top-5 error rate on iNat minival before and after fine-tuning on a more balanced subset. This simple strategy improves the performance on long-tailed iNat dataset.

ments). Table 4.4 shows performance improvements on head and tail categories with fine-tuning. Improvements on head categories with $\geq$ 100 training images are 1.95% of top-1 and 0.92% of top-5; whereas on tail categories with $<$ 100 training images, the improvements are 5.74% of top-1 and 2.71% of top-5. These results verify that the proposed fine-tuning scheme greatly improves the performance on underrepresented tail categories.

Table 4.5 presents the detailed performance breakdown of our winning entry in the iNaturalist 2017 challenge [1]. Using higher image resolution and further fine-tuning on a more balanced subset are the key to our success.

[1]`https://www.kaggle.com/c/inaturalist-challenge-at-fgvc-2017`

	Before FT		**After FT**	
	Top-1	**Top-5**	**Top-1**	**Top-5**
Head: $\geq$ 100 imgs	19.28	5.79	17.33	4.87
Tail: $<$ 100 imgs	29.89	9.12	24.15	6.41

Table 4.4: Top-1 and top-5 error rates (%) on iNat minival for Inception-v4 560. The proposed fine-tuning scheme greatly improves the performance on underrepresented tail categories.

Network	**Top-1 (%)**	**Top-5 (%)**
Inc-v3 299	29.9	10.6
Inc-v3 560	25.4 (+ 4.5)	8.6 (+ 2.0)
Inc-v3 560 FT	22.7 (+ 2.7)	6.6 (+ 2.0)
Inc-v4 560 FT	20.8 (+ 1.9)	5.4 (+ 1.2)
Inc-v4 560 FT 12-crop	19.2 (+ 1.6)	4.7 (+ 0.7)
Ensemble	18.1 (+ 1.1)	4.1 (+ 0.6)

Table 4.5: Performance improvements on iNat minival. The number inside the brackets indicates the improvement over the model in the previous row. FT denotes using the proposed fine-tuning to deal with long-tailed distribution. Ensemble contains two models: Inc-v4 560 FT and Inc-ResNet-v2 560 FT with 12-crop.

4.4.3 Domain Similarity and Transfer Learning

We evaluate the proposed transfer learning method by pre-training the network on source domain *from scratch*, and then fine-tune on target domains for fine-grained visual categorization. Other than training separately on ImageNet and iNat, we also train networks on a combined ImageNet + iNat dataset that contains 1,946,640 training images from 6,089 categories (*i.e.*, 1,000 from ImageNet and 5,089 from iNat). We use input size of 299 $\times$ 299 for all networks. Table 4.6 shows the pre-training performance evaluated on ImageNet val and iNat minival. Notably, a single network trained on the combined ImageNet + iNat dataset achieves competitive performance compared with two models trained separately. In general, combined training is better than training separately in the

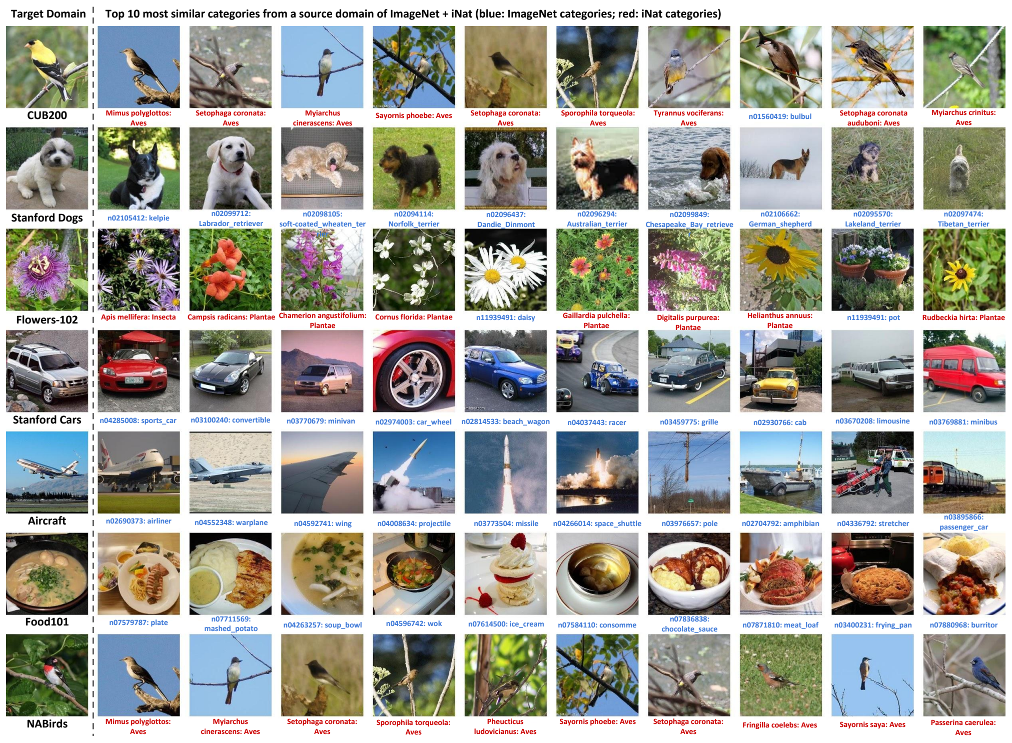


Figure 4.5: Examples showing top 10 most similar categories in the combined ImageNet + iNat for each FGVC dataset, calculated with our proposed domain similarity. The left column represents 7 FGVC target domains, each by a randomly chosen image from the dataset. Each row shows top 10 most similar categories in ImageNet + iNat for a specific FGVC target domain. We represent a category by one randomly chosen image from that category. ImageNet categories are marked in blue, whereas iNat categories are in red.

case of Inception and Inception SE, but worse in the case of ResNet.

Based on the proposed domain selection strategy defined in Section 4.3.2, we select the following two subsets from the combined ImageNet + iNat dataset: **Subset A** was chosen by including top 200 ImageNet + iNat categories for each of the 7 FGVC dataset. Removing duplicated categories resulted in a source domain containing 832 categories. **Subset B** was selected by adding most similar 400 categories for CUB200, NABirds, top 100 categories for Stanford Dogs and top 50 categories for Stanford Cars and Aircraft,

	ImageNet val						iNaturalist minival			
	Original		Separate Train		Combined Train		Separate Train		Combined Train	
	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
ResNet-50 [66, 67]	24.70	7.80	**24.33**	**7.61**	25.23	8.06	**36.23**	**15.67**	36.93	16.49
ResNet-101 [66, 67]	23.60	7.10	**23.08**	7.09	23.39	**7.06**	34.15	14.58	**33.97**	**14.53**
ResNet-152 [66, 67]	23.00	6.70	**22.34**	6.81	22.59	**6.64**	**31.04**	**12.52**	32.58	13.20
Inception-v3 [152]	21.20	5.60	21.73	5.97	**21.52**	**5.87**	31.18	11.90	**30.29**	**11.10**
Inception-ResNet-v2 [150]	19.90*	4.90*	20.33	**5.16**	**20.20**	5.18	**27.53**	9.87	27.78	**9.12**
Inception-v3 SE [71]	-	-	20.98	5.76	**20.75**	**5.69**	30.15	11.69	**29.79**	**10.64**
Inception-ResNet-v2 SE [71]	19.80	4.79	19.77	4.79	**19.56**	**4.61**	27.30	9.61	**26.01**	**8.18**

Table 4.6: Pre-training performance on different source domains. Networks trained on the combined ImageNet + iNat dataset with 6,089 classes achieve competitive performance on both ImageNet and iNat compared with networks trained separately on each dataset. $*$ indicates the model was evaluated on the non-blacklisted subset of ImageNet validation set that may slightly improve the performance.

which gave us 585 categories in total. Figure 4.5 shows top 10 most similar categories in ImageNet + iNat for all FGVC datasets calculated by our proposed domain similarity. It's clear to see that for CUB200, Flowers-102 and NABirds, most similar categories are from iNat; whereas for Stanford Dogs, Stanford Cars, Aircraft and Food101, most similar categories are from ImageNet. This indicates the strong dataset bias in both ImageNet and iNat.

The transfer learning performance by fine-tuning an Inception-v3 on fine-grained datasets are presented in Table 4.7. We can see that both ImageNet and iNat are highly biased, achieving dramatically different transfer learning performance on target datasets. Interestingly, when we transfer networks trained on the combined ImageNet + iNat dataset, performance are in-between ImageNet and iNat pre-training, indicating that we cannot achieve good performance on target domains by simply using a larger scale, combined source domain.

Further, in Figure 4.6, we show the relationship between transfer learning performance and our proposed domain similarity. We observe better transfer learning performance when fine-tuned from a more similar source domain, except Food101, on which the transfer learning performance almost stays same as domain similarity changes. We believe this is likely due to the large number of training images in Food101 (750 training images per class). Therefore, the target domain contains enough data thus transfer learning has very little help. In such a scenario, our assumption on ignoring the scale of domain is no longer valid.

From Table 4.7 and Figure 4.6, we observe that the selected Subset B achieves good performance among all FGVC datasets, surpassing ImageNet pre-training by a large margin on CUB200 and NABirds. In Table 4.8, we compare our approach with existing FGVC methods. Results demonstrate state-of-the-art performance of the prposed trans-

	CUB200	Stanford Dogs	Flowers-102	Stanford Cars	Aircraft	Food101	NABirds
ImageNet	82.84	84.19	96.26	91.31	85.49	88.65	82.01
iNat	89.26	78.46	97.64	88.31	82.61	88.80	87.91
ImageNet + iNat	85.84	82.36	97.07	91.38	85.21	88.45	83.98
Subset A (832-class)	86.37	84.69	97.65	91.42	86.28	88.78	84.79
Subset B (585-class)	88.76	85.23	97.37	90.58	86.13	88.37	87.89

Table 4.7: Transfer learning performance on 7 FGVC datasets by fine-tuning the Inception-v3 299 pre-trained on different source domains. Each row represents a network pre-trained on a specific source domain, and each column shows the top-1 image classification accuracy by fine-tuning different networks on a target fine-grained dataset. Relative good and poor performance on each FGVC dataset are marked by green and red, respectively. Two selected subsets based on domain similarity achieve good performance on all FGVC datasets.

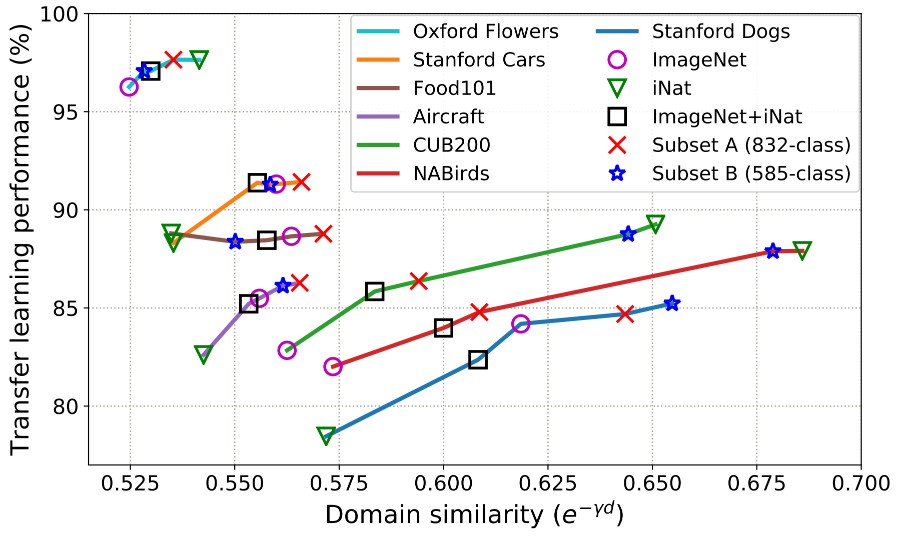


Figure 4.6: The relationship between transfer learning performance and domain similarity between source and target domain. Each line represents a target FGVC dataset and each marker represents the source domain. Better transfer learning performance can be achieved by fine-tuning the network that is pre-trained on a more similar source domain. Two selected subsets based on our domain similarity achieve good performance on all FGVC datasets.

fer learning method on commonly used FGVC datasets. Notice that since our definition of domain similarity is fast to compute, we can easily explore different ways to select a source domain. The transfer learning performance can be directly estimated based on domain similarity, without conducting any pre-training and fine-tuning. Prior to our work in this chapter, the only options to achieve good performance on FGVC tasks are either designing better models based on ImageNet fine-tuning [110, 33, 191] or augmenting the dataset by collecting more images [177, 97]. Our work, however, provides a novel direction of using a more similar source domain to pre-train the network. We show that with properly selected subsets in source domain, it is able to match or exceed those performance gain by simply fine-tuning off-the-shelf networks.

Method	**CUB200**	**Stanford Dogs**	**Stanford Cars**	**Aircrafts**	**Food101**
Subset B (585-class): Inception-v3	**89.6**	86.3	93.1	89.6	90.1
Subset B (585-class): Inception-ResNet-v2 SE	89.3	**88.0**	**93.5**	**90.7**	**90.4**
Krause *et al*. [96]	82.0	-	92.6	-	-
Bilinear-CNN [110]	84.1	-	91.3	84.1	82.4
Compact Bilinear Pooling [49]	84.3	-	91.2	84.1	83.2
Zhang *et al*. [190]	84.5	72.0	-	-	-
Low-rank Bilinear Pooling [92]	84.2	-	90.9	87.3	-
Kernel Pooling [33]	86.2	-	92.4	86.9	85.5
RA-CNN [47]	85.3	**87.3**	92.5	-	-
Improved Bilinear-CNN [109]	85.8	-	92.0	88.5	-
MA-CNN [191]	**86.5**	-	92.8	89.9	-
DLA [183]	85.1	-	**94.1**	**92.6**	**89.7**

Table 4.8: Comparison to existing state-of-the-art FGVC methods. As a convention, we use same 448×448 input size. Since we didn't find recent proposed FGVC methods applied to Flowers-102 and NABirds, we only show comparisons on the rest of 5 datasets. Our proposed transfer learning approach is able to achieve state-of-the-art performance on all FGVC datasets, especially on CUB200 and NABirds.

4.5 Conclusion

In this chapter, we have presented a training scheme that achieves top performance on large scale iNaturalist dataset, by using higher resolution input image and fine-tuning to deal with long-tailed distribution. We further proposed a novel way of capturing domain similarity with Earth Mover's Distance and showed better transfer learning performance can be achieved by fine-tuning from a more similar domain. In the future, we plan to study other important factors in transfer learning beyond domain similarity.

CHAPTER 5

CLASS-BALANCED LOSS BASED ON EFFECTIVE NUMBER OF SAMPLES

The recent success of deep Convolutional Neural Networks (CNNs) for visual recognition [99, 147, 151, 66] owes much to the availability of large-scale, real-world annotated datasets [36, 107, 192, 160]. In contrast with commonly used visual recognition datasets (*e.g.*, CIFAR [98, 158], ImageNet ILSVRC 2012 [36, 140] and CUB-200 Birds [167]) that exhibit roughly uniform distributions of class labels, real-world datasets have skewed [86] distributions, with a *long-tail*: a few dominant classes claim most of the examples, while most of the other classes are represented by relatively few examples. Models trained on such data perform poorly for weakly represented classes [80, 65, 161, 18].

A number of recent studies have aimed to alleviate the challenge of long-tailed training data [12, 126, 72, 161, 169, 56, 189, 180]. In general, there are two strategies: re-sampling and cost-sensitive re-weighting. In re-sampling, the number of examples is directly adjusted by over-sampling (adding repetitive data) for the minor class or under-sampling (removing data) for the major class, or both. In cost-sensitive re-weighting, we influence the loss function by assigning relatively higher costs to examples from minor classes. In the context of deep feature representation learning using CNNs, re-sampling may either introduce large amounts of duplicated samples, which slows down the training and makes the model susceptible to overfitting when over-sampling, or discard valuable examples that are important for feature learning when under-sampling. Due to these disadvantages of applying re-sampling for CNN training, the present chapter focuses on re-weighting approaches, namely, how to design a better class-balanced loss.

Typically, we assign sample weights or re-sample data inversely proportionally to the class frequency. This simple heuristic has been widely adopted [72, 169]. However, recent work on training from large-scale, real-world, long-tailed datasets [120, 116] reveal poor performance when using this strategy. Instead, they propose to use a “smoothed” version that empirically re-samples data to be inversely proportional to the square root of class frequency. These observations suggest an interesting question: how can we design a better class-balanced loss that is applicable to a diverse array of datasets with drastically different scale and imbalance?

We aim to answer this question from the perspective of sample size. As illustrated in Figure 5.1, we consider training a model to discriminate between a major class and a minor class from a long-tailed dataset. Due to highly imbalanced data, directly training the model or re-weighting the loss by inverse number of samples cannot yield satisfactory performance. Intuitively, the more data, the better. However, since there is information overlap among data, as the number of samples increases, the marginal benefit a model can extract from the data diminishes. In light of this, we propose a novel theoretical framework to characterize data overlap and calculate the effective number of samples in a model- and loss-agnostic manner. A class-balanced re-weighting term that is inversely proportional to the effective number of samples is added to the loss function. Extensive experimental results indicate that this class-balanced term provides a significant boost to the performance of commonly used loss functions for training CNNs on long-tailed datasets.

Our key contributions can be summarized as follows: (1) We provide a theoretical framework to study the effective number of samples and show how to design a class-balanced term to deal with long-tailed training data. (2) We show that significant performance improvements can be achieved by adding the proposed class-balanced term to existing commonly used loss functions including softmax cross-entropy, sig-

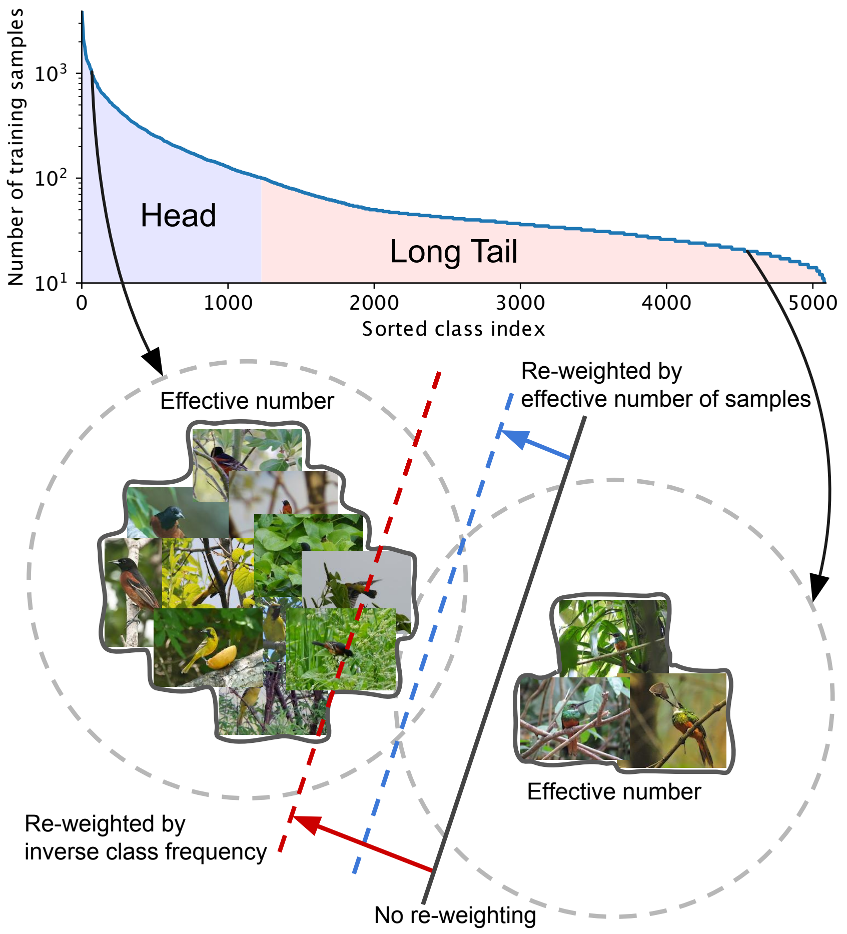


Figure 5.1: Two classes, one from the head and one from the tail of a long-tailed dataset (iNaturalist 2017 [160] in this example), have drastically different number of samples. Models trained on these samples are biased toward dominant classes (black solid line). Re-weighing the loss by inverse class frequency usually yields poor performance (red dashed line) on real-world data with high class imbalance. We propose a theoretical framework to quantify the effective number of samples by taking data overlap into consideration. A class-balanced term is designed to re-weight the loss by inverse effective number of samples. We show in experiments that the performance of a model can be improved when trained with the proposed class-balanced loss (blue dashed line).

moid cross-entropy and focal loss. In addition, we show our class-balanced loss can be used as a generic loss for visual recognition by outperforming commonly-used softmax cross-entropy loss on ILSVRC 2012. We believe our study on quantifying the effective number of samples and class-balanced loss can offer useful guidelines for researchers working in domains with long-tailed class distributions.

5.1 Related Work

Most of previous efforts on long-tailed imbalanced data can be divided into two regimes: re-sampling [145, 56, 18, 196] (including over-sampling and under-sampling) and cost-sensitive learning [156, 193, 72, 88, 141].

Re-Sampling. Over-sampling adds repeated samples from minor classes, which could cause the model to overfit. To solve this, novel samples can be either interpolated from neighboring samples [24] or synthesized [64, 196] for minor classes. However, the model is still error-prone due to noise in the novel samples. It was argued that even if over-sampling incurs risks from removing important samples, under-sampling is still preferred over over-sampling [42].

Cost-Sensitive Learning. Cost-Sensitive Learning can be traced back to a classical method in statistics called importance sampling [84], where weights are assigned to samples in order to match a given data distribution. Elkan *et al*. [43] studied how to assign weights to adjust the decision boundary to match a given target in the case of binary classification. For imbalanced datasets, weighting by inverse class frequency [72, 169] or a smoothed version of inverse square root of class frequency [120, 116] are often adopted. As a generalization of smoothed weighting with a theoretically grounded framework, we focus on (a) how to quantify the effective number of samples and (b) using it to re-weight the loss. Another line of important work aims to study sample difficulty in terms of loss and assign higher weights to hard examples [46, 119, 41, 106]. Samples from minor classes tend to have higher losses than those from major classes as the features learned in minor classes are usually poorer. However, there is no direct connection between sample difficulty and the number of samples. A side effect of assigning higher weights to hard examples is the focus on harmful samples (*e.g.*, noisy data or mislabeled data) [91, 135]. In our work, we do not make any assumptions on the sample difficulty and data distribution. By improving the focal loss [106] using our class-balanced term in experiments, we show that our method is complementary to re-weighting based on sample difficulty.

It is noteworthy to mention other efforts in dealing with data imbalance, including transferring the knowledge learned from major classes to minor classes [12, 126, 169, 30, 180] and designing a better training objective via metric learning [72, 189, 182] or Bayesian uncertainty estimates [87].

Covering and Effective Sample Size. Our theoretical framework is inspired by the random covering problem [79], where the goal is to cover a large set by a sequence of i.i.d. random small sets. We simplify the problem in Section 5.2 by making reasonable assumptions. Note that the effective number of samples proposed in this chapter is different from the concept of effective sample size in statistics. The effective sample size is used to calculate variance when samples are correlated.

5.2 Effective Number of Samples

We formulate the data sampling process as a simplified version of random covering. The key idea is to associate each sample with a small neighboring region instead of a single point. We present our theoretical framework and the formulation of calculating effective number of samples.

5.2.1 Data Sampling as Random Covering

Given a class, denote the set of all possible data in the feature space of this class as $\mathcal{S}$. We assume the volume of $\mathcal{S}$ is N and $N \geq 1$. Denote each sample as a subset of $\mathcal{S}$ that has the unit volume of 1 and may overlap with other samples. Consider the data sampling process as a random covering problem where each subset is randomly sampled from $\mathcal{S}$ to cover the entire set of $\mathcal{S}$. The more data is being sampled, the better the coverage of $\mathcal{S}$ is. The expected total volume of sampled data increases as the number of samples increases and is bounded by N. Therefore, we define:

Definition 1 (Effective Number)**.** The *effective number* of samples is the expected volume of samples.

The calculation of the expected volume of samples is a very difficult problem that depends on the shape of the sample and the dimensionality of the feature space [79]. To make the problem tamable, we simplify the problem by not considering the situation of partial overlapping. That is, we assume a newly sampled data point can only interact with previously sampled data in two ways: either entirely inside the set of previously sampled data with the probability of p or entirely outside with the probability of $1-p$, as illustrated in Figure 5.2. As the number of sampled data points increases, the probability p will be higher.

Before we dive into the mathematical formulations, we discuss the connection between our definition of effective number of samples and real-world visual data. Our idea is to capture the diminishing marginal benefits by using more data points of a class.

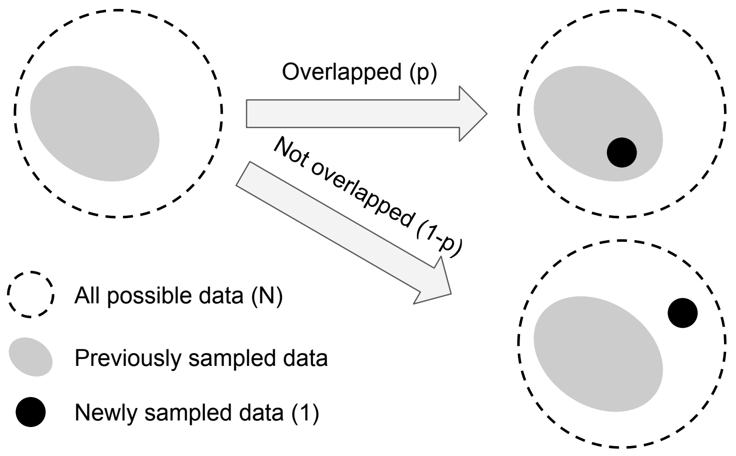


Figure 5.2: Giving the set of all possible data with volume N and the set of previously sampled data, a new sample with volume 1 has the probability of p being overlapped with previous data and the probability of $1 - p$ not being overlapped.

Due to intrinsic similarities among real-world data, as the number of samples grows, it is highly possible that a newly added sample is a near-duplicate of existing samples. In addition, CNNs are trained with heavy data augmentations, where simple transformations such as random cropping, re-scaling and horizontal flipping will be applied to the input data. In this case, all augmented examples are also considered as same with the original example. Presumably, the stronger the data augmentation is, the smaller the N will be. The small neighboring region of a sample is a way to capture all near-duplicates and instances that can be obtained by data augmentation. For a class, N can be viewed as the number of *unique prototypes*.

5.2.2 Mathematical Formulation

Denote the effective number (expected volume) of samples as E_n, where $n \in \mathbb{Z}_{>0}$ is the number of samples.

Proposition 1 (Effective Number)**.** $E_n = (1-\beta^n)/(1-\beta)$, where $\beta = (N-1)/N$.

Proof. We prove the proposition by induction. It is obvious that $E_1 = 1$ because there is no overlapping. So $E_1 = (1-\beta^1)/(1-\beta) = 1$ holds. Now let's consider a general case where we have previously sampled $n-1$ examples and are about to sample the n^{th} example. Now the expected volume of previously sampled data is E_{n-1} and the newly sampled data point has the probability of $p = E_{n-1}/N$ to be overlapped with previous samples. Therefore, the expected volume after sampling n^{th} example is:

$$E_n = pE_{n-1} + (1-p)(E_{n-1}+1) = 1 + \frac{N-1}{N}E_{n-1}. \tag{5.1}$$

Assume $E_{n-1} = (1-\beta^{n-1})/(1-\beta)$ holds, then

$$E_n = 1 + \beta\frac{1-\beta^{n-1}}{1-\beta} = \frac{1-\beta+\beta-\beta^n}{1-\beta} = \frac{1-\beta^n}{1-\beta}. \tag{5.2}$$

□

The above proposition shows that the effective number of samples is an exponential function of n. The hyperparameter $\beta \in [0,1)$ controls how fast E_n grows as n increases.

Another explanation of the effective number E_n is:

$$E_n = (1-\beta^n)/(1-\beta) = \sum_{j=1}^{n}\beta^{j-1}. \tag{5.3}$$

This means that the j^{th} sample contributes β^{j-1} to the effective number. The expected total volume N for all possible data in the class can then be calculated as:

$$N = \lim_{n\to\infty}\sum_{j=1}^{n}\beta^{j-1} = 1/(1-\beta). \tag{5.4}$$

This is consistent with our definition of β in the proposition.

Implication 1 (Asymptotic Properties)**.** $E_n = 1$ if $\beta = 0$ ($N = 1$). $E_n \to n$ as $\beta \to 1$ ($N \to \infty$).

Proof. If $\beta = 0$, then $E_n = (1 - 0^n)/(1 - 0) = 1$. In the case of $\beta \to 1$, denote $f(\beta) = 1 - \beta^n$ and $g(\beta) = 1 - \beta$. Since $\lim_{\beta \to 1} f(\beta) = \lim_{\beta \to 1} g(\beta) = 0$, $g'(\beta) = -1 \neq 0$ and $\lim_{\beta \to 1} f'(\beta)/g'(\beta) = \lim_{\beta \to 1}(-n\beta^{n-1})/(-1) = n$ exists, using L'Hôpital's rule, we have:

$$\lim_{\beta \to 1} E_n = \lim_{\beta \to 1} \frac{f(\beta)}{g(\beta)} = \lim_{\beta \to 1} \frac{f'(\beta)}{g'(\beta)} = n. \tag{5.5}$$

□

The asymptotic property of E_n shows that when N is large, the effective number of samples is same as the number of samples n. In this scenario, we think the number of unique prototypes N is large, thus there is no data overlap and every sample is unique. On the other extreme, if $N = 1$, this means that we believe there exist a single prototype so that all the data in this class can be represented by this prototype via data augmentation, transformations, *etc*.

5.3 Class-Balanced Loss

The *Class-Balanced Loss* is designed to address the problem of training from imbalanced data by introducing a weighting factor that is inversely proportional to the effective number of samples. The class-balanced loss term can be applied to a wide range of deep networks and loss functions.

For an input sample $\mathbf{x}$ with label $y \in \{1, 2, \dots, C\}$ [1], where C is the total number of classes, suppose the model's estimated class probabilities are $\mathbf{p} = [p_1, p_2, \dots, p_C]^\top$, where $p_i \in [0, 1]\ \forall\ i$, we denote the loss as $\mathcal{L}(\mathbf{p}, y)$. Suppose the number of samples for class i is n_i, based on Equation 5.2, the proposed effective number of samples for class i

[1]For simplicity, we derive the loss function by assuming there is exactly one ground-truth label for a sample.

is $E_{n_i} = (1-\beta_i^{n_i})/(1-\beta_i)$, where $\beta_i = (N_i - 1)/N_i$. Without further information of data for each class, it is difficult to empirically find a set of good hyperparameters N_i for all classes. Therefore, in practice, we assume N_i is only dataset-dependent and set $N_i = N$, $\beta_i = \beta = (N-1)/N$ for all classes in a dataset.

To balance the loss, we introduce a weighting factor α_i that is inversely proportional to the effective number of samples for class i: $\alpha_i \propto 1/E_{n_i}$. To make the total loss roughly in the same scale when applying α_i, we normalize α_i so that $\sum_{i=1}^{C} \alpha_i = C$. For simplicity, we abuse the notation of $1/E_{n_i}$ to denote the normalized weighting factor in the rest of this chapter.

Formally speaking, given a sample from class i that contains n_i samples in total, we propose to add a weighting factor $(1-\beta)/(1-\beta^{n_i})$ to the loss function, with hyperparameter $\beta \in [0, 1)$. The class-balanced (CB) loss can be written as:

$$\text{CB}(\mathbf{p}, y) = \frac{1}{E_{n_y}} \mathcal{L}(\mathbf{p}, y) = \frac{1-\beta}{1-\beta^{n_y}} \mathcal{L}(\mathbf{p}, y), \tag{5.6}$$

where n_y is the number of samples in the ground-truth class y. We visualize class-balanced loss in Figure 5.3 as a function of n_y for different β. Note that $\beta = 0$ corresponds to no re-weighting and $\beta \to 1$ corresponds to re-weighing by inverse class frequency. The proposed novel concept of effective number of samples enables us to use a hyperparameter β to smoothly adjust the class-balanced term between no re-weighting and re-weighing by inverse class frequency.

The proposed class-balanced term is model-agnostic and loss-agnostic in the sense that it is independent to the choice of loss function $\mathcal{L}$ and predicted class probabilities $\mathbf{p}$. To demonstrate the proposed class-balanced loss is generic, we show how to apply class-balanced term to three commonly used loss functions: softmax cross-entropy loss, sigmoid cross-entropy loss and focal loss.

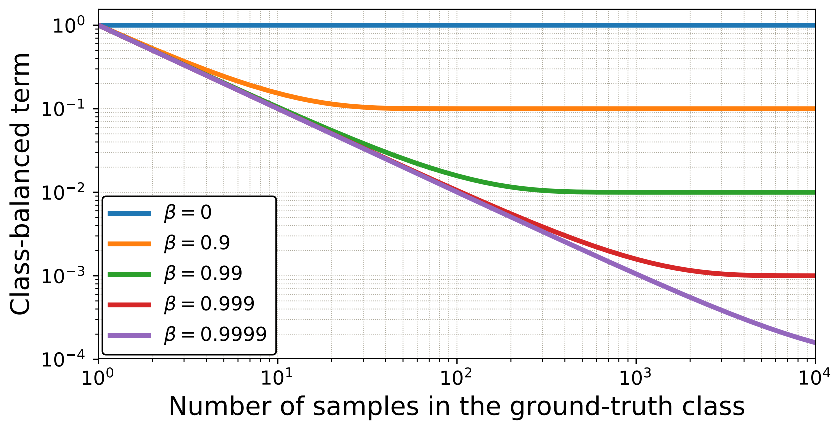


Figure 5.3: Visualization of the proposed class-balanced term $(1-\beta)/(1-\beta^{n_y})$, where n_y is the number of samples in the ground-truth class. Both axes are in log-scale. For a long-tailed dataset where major classes have significantly more samples than minor classes, setting β properly re-balances the relative loss across classes and reduces the drastic imbalance of re-weighing by inverse class frequency.

5.3.1 Class-Balanced Softmax Cross-Entropy Loss

Suppose the predicted output from the model for all classes are $\mathbf{z} = [z_1, z_2, \ldots, z_C]^\top$, where C is the total number of classes. The softmax function regards each class as mutual exclusive and calculate the probability distribution over all classes as $p_i = \exp(z_i)/\sum_{j=1}^{C}\exp(z_j), \forall\ i \in \{1, 2, \ldots, C\}$. Given a sample with class label y, the softmax cross-entropy (CE) loss for this sample is written as:

$$\mathrm{CE}_{\mathrm{softmax}}(\mathbf{z}, y) = -\log\left(\frac{\exp(z_y)}{\sum_{j=1}^{C}\exp(z_j)}\right). \tag{5.7}$$

Suppose class y has n_y training samples, the class-balanced (CB) softmax cross-entropy loss is:

$$\mathrm{CB}_{\mathrm{softmax}}(\mathbf{z}, y) = -\frac{1-\beta}{1-\beta^{n_y}}\log\left(\frac{\exp(z_y)}{\sum_{j=1}^{C}\exp(z_j)}\right). \tag{5.8}$$

5.3.2 Class-Balanced Sigmoid Cross-Entropy Loss

Different from softmax, class-probabilities calculated by sigmoid function assume each class is independent and not mutually exclusive. When using sigmoid function, we regard multi-class visual recognition as multiple binary classification tasks, where each output node of the network is performing a one-vs-all classification to predict the probability of the target class over the rest of classes. Compared with softmax, sigmoid presumably has two advantages for real-world datasets: (1) Sigmoid doesn't assume the mutual exclusiveness among classes, which aligns well with real-world data, where a few classes might be very similar to each other, especially in the case of large number of fine-grained classes. (2) Since each class is considered independent and has its own predictor, sigmoid unifies single-label classification with multi-label prediction. This is a nice property to have since real-world data often has more than one semantic label.

Using same notations as softmax cross-entropy, for simplicity, we define z_i^t as:

$$z_i^t = \begin{cases} z_i, & \text{if } i = y. \\ -z_i, & \text{otherwise.} \end{cases} \tag{5.9}$$

Then the sigmoid cross-entropy (CE) loss can be written as:

$$\begin{aligned} \text{CE}_{\text{sigmoid}}(\mathbf{z}, y) &= -\sum_{i=1}^{C} \log\left(\text{sigmoid}(z_i^t)\right) \\ &= -\sum_{i=1}^{C} \log\left(\frac{1}{1+\exp(-z_i^t)}\right). \end{aligned} \tag{5.10}$$

The class-balanced (CB) sigmoid cross-entropy loss is:

$$\text{CB}_{\text{sigmoid}}(\mathbf{z}, y) = -\frac{1-\beta}{1-\beta^{n_y}} \sum_{i=1}^{C} \log\left(\frac{1}{1+\exp(-z_i^t)}\right). \tag{5.11}$$

5.3.3 Class-Balanced Focal Loss

The recently proposed focal loss (FL) [106] adds a modulating factor to the sigmoid cross-entropy loss to reduce the relative loss for well-classified samples and focus on difficult samples. Denote $p_i^t = \text{sigmoid}(z_i^t) = 1/(1 + \exp(-z_i^t))$, the focal loss can be written as:

$$\text{FL}(\mathbf{z}, y) = -\sum_{i=1}^{C}(1 - p_i^t)^\gamma \log(p_i^t). \tag{5.12}$$

The class-balanced (CB) focal loss is:

$$\text{CB}_{\text{focal}}(\mathbf{z}, y) = -\frac{1-\beta}{1-\beta^{n_y}}\sum_{i=1}^{C}(1 - p_i^t)^\gamma \log(p_i^t). \tag{5.13}$$

The original focal loss has an α-balanced variant. The class-balanced focal loss is same as α-balanced focal loss when $\alpha_t = (1-\beta)/(1-\beta^{n_y})$. Therefore, the class-balanced term can be viewed as an explicit way to set α_t in focal loss based on the effective number of samples.

5.4 Experiments

The proposed class-balanced losses are evaluated on artificially created long-tailed CIFAR [98] datasets with controllable degrees of data imbalance and real-world long-tailed datasets iNaturalist [160]. To demonstrate our loss is generic for visual recognition, we also present experiments on ImageNet data (ILSVRC 2012 [140]). We use deep residual networks (ResNet) [66] with various depths and train all networks from scratch.

5.4.1 Datasets

Long-Tailed CIFAR. To analyze the proposed class-balanced loss, long-tailed versions of CIFAR [98] are created by reducing the number of training samples per class according to an exponential function $n = n_i \mu^i$, where i is the class index (0-indexed), n_i is the original number of training images and $\mu \in (0, 1)$. The test set remains unchanged. We define the imbalance factor of a dataset as the number of training samples in the largest class divided by the smallest. Figure 5.4 shows number of training images per class on long-tailed CIFAR-100 with imbalance factors ranging from 10 to 200. We conduct experiments on long-tailed CIFAR-10 and CIFAR-100.

iNaturalist. The recently introduced iNaturalist species classification and detection dataset [160] is a real-world long-tailed dataset containing 579,184 training images from 5,089 classes in its 2017 version and 437,513 training images from 8,142 classes in its 2018 version [2]. We use the official training and validation splits in our experiments.

ImageNet. We use the ILSVRC 2012 [140] split containing 1,281,167 training and 50,000 validation images.

Table 5.1 summarizes all datasets used in our experiments along with their imbalance factors.

5.4.2 Implementation

Training with sigmoid-based losses. Conventional training scheme of deep networks initializes the last linear classification layer with bias $b = 0$. As pointed out by Lin *et al.* [106], this could cause instability of training when using sigmoid function to get

[2]`https://github.com/visipedia/inat_comp`

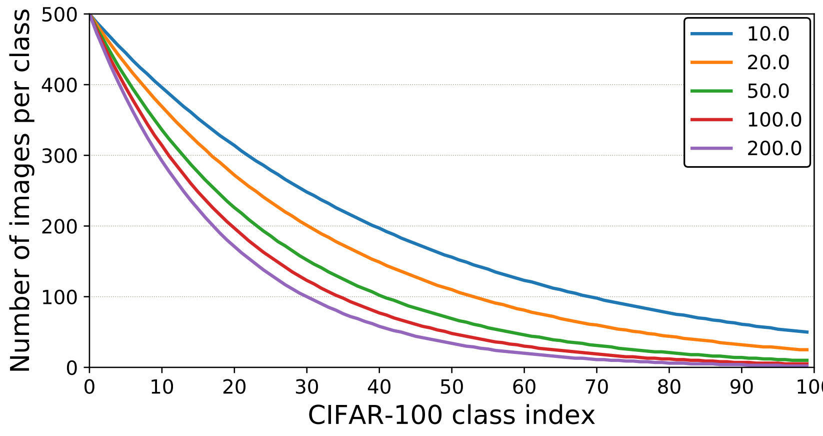


Figure 5.4: Number of training samples per class in artificially created long-tailed CIFAR-100 datasets with different imbalance factors.

Dataset Name	# Classes	Imbalance
Long-Tailed CIFAR-10	10	10.00 - 200.00
Long-Tailed CIFAR-100	100	10.00 - 200.00
iNaturalist 2017	5,089	435.44
iNaturalist 2018	8,142	500.00
ILSVRC 2012	1,000	1.78

Table 5.1: Datasets that are used to evaluate the effectiveness of class-balanced loss. We created 5 long-tailed versions of both CIFAR-10 and CIFAR-100 with imbalance factors of 10, 20, 50, 100 and 200 respectively.

class probabilities. This is because using $b = 0$ with sigmoid function in the last layer induces huge loss at the beginning of the training as the output probability for each class is close to 0.5. Therefore, for training with sigmoid cross-entropy loss and focal loss, we assume the class prior is $\pi = 1/C$ for each class, where C is the number of classes, and initialize the bias of the last layer as $b = -\log((1-\pi)/\pi)$. In addition, we remove the ℓ_2-regularization (weight decay) for the bias b of the last layer.

We used Tensorflow [1] to implement and train all the models by stochastic gradient descent with momentum. We trained residual networks with 32 layers (ResNet-32) to conduct all experiments on CIFAR. Similar to Zagoruyko *et al.* [185], we noticed a disturbing effect in training ResNets on CIFAR that both loss and validation error gradually went up after the learning rate drop, especially in the case of high data imbalance. We found that setting learning rate decay to 0.01 instead of 0.1 solved the problem. Models on CIFAR were trained with batch size of 128 on a single NVIDIA Titan X GPU for 200 epochs. The initial learning rate was set to 0.1, which was then decayed by 0.01 at 160 epochs and again at 180 epochs. We also used linear warm-up of learning rate [62] in the first 5 epochs. On iNaturalist and ILSVRC 2012 data, we followed the same training strategy used by Goyal *et al.* [62] and trained residual networks with batch size of 1024 on a single Cloud TPU. Since the scale of focal loss is smaller than softmax and sigmoid cross-entropy loss, when training with focal loss, we used 2$\times$ and 4$\times$ larger learning rate on ILSVRC 2012 and iNaturalist respectively. Code, data and pre-trained models are available at: `https://github.com/richardaecn/class-balanced-loss`.

5.4.3 Visual Recognition on Long-Tailed CIFAR

We conduct extensive studies on long-tailed CIFAR datasets with various imbalance factors. Table 5.2 shows the performance of ResNet-32 in terms of classification error rate on the test set. We present results of using softmax cross-entropy loss, sigmoid cross-entroy loss, focal loss with different γ, and the proposed class-balanced loss with best hyperparameters chosen via cross-validation. The search space of hyperparameters is {softmax, sigmoid, focal} for loss type, $\beta \in \{0.9, 0.99, 0.999, 0.9999\}$ (Section 5.3), and $\gamma \in \{0.5, 1.0, 2.0\}$ for focal loss [106].

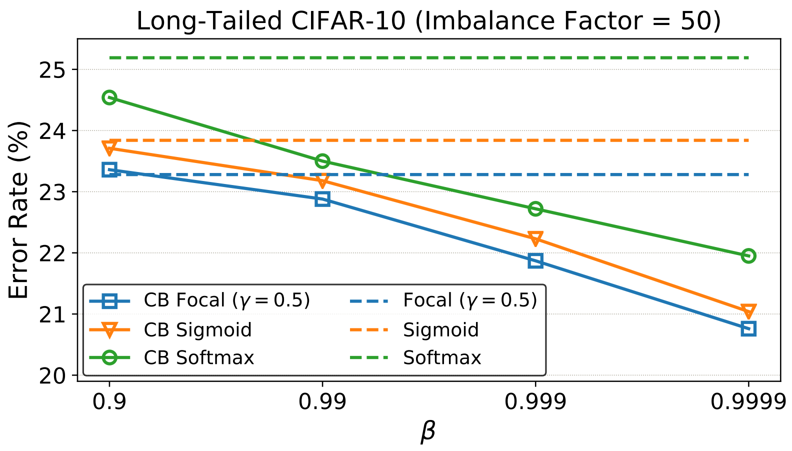
From results in Table 5.2, we have the following observations: (1) With properly selected hyperparameters, class-balanced loss is able to significantly improve the performance of commonly used loss functions on long-tailed datasets. (2) Softmax cross-

Dataset Name	Long-Tailed CIFAR-10						Long-Tailed CIFAR-100					
Imbalance	200	100	50	20	10	1	200	100	50	20	10	1
Softmax	**34.32**	29.64	25.19	17.77	13.61	6.61	65.16	61.68	56.15	48.86	44.29	29.07
Sigmoid	34.51	**29.55**	23.84	**16.40**	**12.97**	**6.36**	64.39	**61.22**	55.85	48.57	44.73	**28.39**
Focal (γ = 0.5)	36.00	29.77	**23.28**	17.11	13.19	6.75	65.00	61.31	55.88	48.90	44.30	28.55
Focal (γ = 1.0)	34.71	29.62	23.29	17.24	13.34	6.60	**64.38**	61.59	**55.68**	**48.05**	**44.22**	28.85
Focal (γ = 2.0)	35.12	30.41	23.48	16.77	13.68	6.61	65.25	61.61	56.30	48.98	45.00	28.52
Class-Balanced	**31.11**	**25.43**	**20.73**	**15.64**	**12.51**	**6.36***	**63.77**	**60.40**	**54.68**	**47.41**	**42.01**	**28.39***
Loss Type	SM	Focal	Focal	SM	SGM	SGM	Focal	Focal	SGM	Focal	Focal	SGM
β	0.9999	0.9999	0.9999	0.9999	0.9999	-	0.9	0.9	0.99	0.99	0.999	-
γ	-	1.0	2.0	-	-	-	1.0	1.0	-	0.5	0.5	-

Table 5.2: Classification error rate of ResNet-32 trained with different loss functions on long-tailed CIFAR-10 and CIFAR-100. We show best results of class-balanced loss with best hyperparameters (SM represents Softmax and SGM represents Sigmoid) chosen via cross-validation. Class-balanced loss is able to achieve significant performance gains. $*$ denotes the case when each class has same number of samples, class-balanced term is always 1 therefore it reduces to the original loss function.

entropy is overwelmingly used as the loss function for visual recognition tasks. However, following the training strategy in Section 5.4.2, sigmoid cross-entropy and focal loss are able to outperform softmax cross-entropy in most cases. (3) The best β is 0.9999 on CIFAR-10 unanimously. But on CIFAR-100, datasets with different imbalance factors tend to have different and smaller optimal β.

To understand the role of β and class-balanced loss better, we use the long-tailed dataset with imbalance factor of 50 as an example to show the error rate of the model when trained with and without the class-balanced term in Figure 5.5. Interestingly, for CIFAR-10, class-balanced term always improves the performance of the original loss and more performance gain can be obtained with larger β. However, on CIFAR-100, only small values of β improve the performance, whereas larger values degrade the performance. Figure 5.6 illustrates the effective number of samples under different β. On CIFAR-10, when re-weighting based on $\beta = 0.9999$, the effective number of samples is close to the number of samples. This means the best re-weighting strategy on CIFAR-10 is similar with re-weighting by inverse class frequency. On CIFAR-100, the poor performance of using larger β suggests that re-weighting by inverse class frequency is not a wise choice. Instead, we need to use a smaller β that has smoother weights across classes. This is reasonable because $\beta = (N - 1)/N$, so larger β means larger N. As discussed in Section 5.2, N can be interpreted as the number of unique prototypes. A fine-grained dataset should have a smaller N compared with a coarse-grained one. For example, the number of unique prototypes of a specific bird species should be smaller than the number of unique prototypes of a generic bird class. Since classes in CIFAR-100 are more fine-grained than CIFAR-10, CIFAR-100 should have smaller N compared with CIFAR-10. This explains our observations on the effect of β.



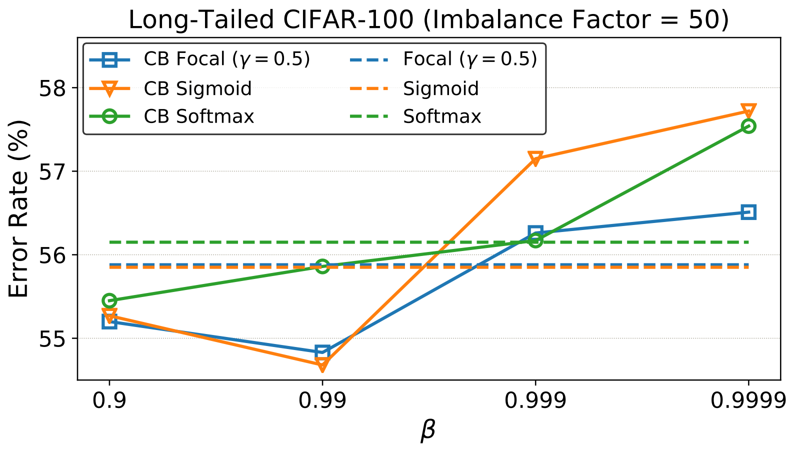


Figure 5.5: Classification error rate when trained with and without the class-balanced term. On CIFAR-10, class-balanced loss yields consistent improvement across different β and the larger the β is, the larger the improvement is. On CIFAR-100, $\beta = 0.99$ or $\beta = 0.999$ improves the original loss, whereas a larger β hurts the performance.

5.4.4 Visual Recognition on Large-Scale Datasets

To demonstrate the proposed class-balanced loss can be used on large-scale real-world datasets, we present results of training ResNets with different depths on iNaturalist 2017, iNaturalist 2018 and ILSVRC 2012.

Table 5.3 summarizes the top-1 and top-5 error rate on the validation set of all datasets. We use the class-balanced focal loss since it has more flexibility and find

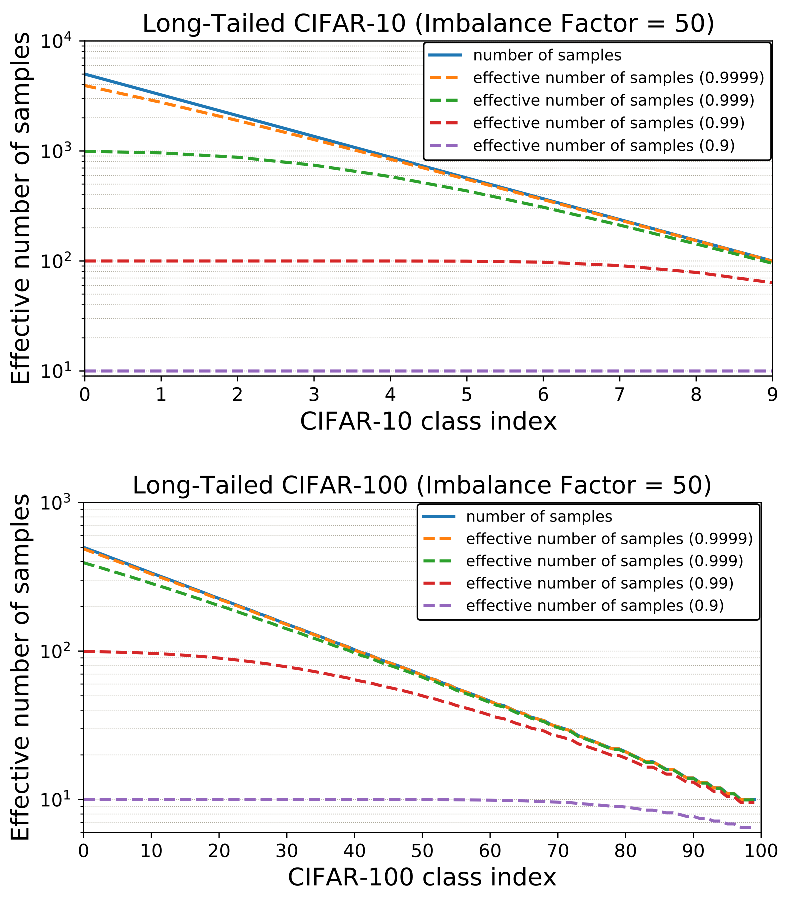
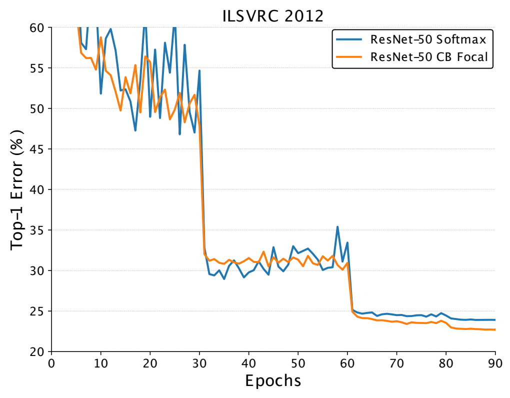


Figure 5.6: Effective number of samples with different β on long-tailed CIFAR-10 and CIFAR-100 with the imbalance of 50. This is a semi-log plot with vertical axis in log-scale. When $\beta \rightarrow 1$, effective number of samples is same as number of samples. When β is small, effective number of samples are similar across all classes.

$\beta = 0.999$ and $\gamma = 0.5$ yield reasonably good performance on all datasets. From results we can see that we are able to outperform commonly used softmax cross-entropy loss on ILSVRC 2012, and by large margins on iNaturalist. Notably, ResNet-50 is able to achieve comparable performance with ResNet-152 on iNaturalist and ResNet-101 on ILSVRC 2012 when using class-balanced focal loss to replace softmax cross-entropy loss. Training curves on ILSVRC 2012 and iNaturalist 2018 are shown in Figure 5.7. Class-balanced focal loss starts to show its advantage after 60 epochs of training.

					iNaturalist 2017		iNaturalist 2018		ILSVRC 2012	
Network	Loss	β	γ	Input Size	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
ResNet-50	Softmax	-	-	224×224	45.38	22.67	42.86	21.31	23.92	7.03
ResNet-101	Softmax	-	-	224×224	42.57	20.42	39.47	18.86	22.65	6.47
ResNet-152	Softmax	-	-	224×224	41.42	19.47	38.61	18.07	21.68	5.92
ResNet-50	CB Focal	0.999	0.5	224×224	41.92	20.92	38.88	18.97	22.71	6.72
ResNet-101	CB Focal	0.999	0.5	224×224	39.06	18.96	36.12	17.18	21.57	5.91
ResNet-152	CB Focal	0.999	0.5	224×224	38.06	18.42	35.21	16.34	20.87	5.61
ResNet-50	CB Focal	0.999	0.5	320×320	38.16	18.28	35.84	16.85	21.99	6.27
ResNet-101	CB Focal	0.999	0.5	320×320	34.96	15.90	32.02	14.27	20.25	5.34
ResNet-152	CB Focal	0.999	0.5	320×320	33.73	14.96	30.95	13.54	19.72	4.97

Table 5.3: Classification error rate on large-scale datasets trained with different loss functions. The proposed class-balanced term combined with focal loss (CB Focal) is able to outperform softmax cross-entropy by a large margin.



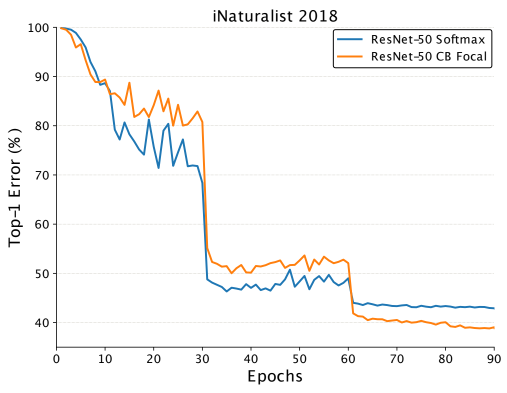


Figure 5.7: Training curves of ResNet-50 on ILSVRC 2012 (upper) and iNaturalist 2018 (lower). Class-balanced focal loss with $\beta = 0.999$ and $\gamma = 0.5$ outperforms softmax cross-entropy after 60 epochs.

5.5 Conclusion

In this chapter, we have presented a theoretically sounded framework to address the problem of long-tailed distribution of training data. The key idea is to take data overlap into consideration to help quantify the effective number of samples. Following this framework, we further propose a *class-balanced loss* to re-weight loss inversely with the effective number of samples per class. Extensive studies on artificially induced long-tailed CIFAR datasets have been conducted to understand and analyze the proposed loss. The benefit of the class-balanced loss has been verified by experiments on both CIFAR and large-scale datasets including iNaturalist and ImageNet.

Our proposed framework provides a non-parametric means of quantifying data overlap, as we don't make any assumptions about data distribution. This makes our loss generally applicable to a wide range of existing models and loss functions. Intuitively, a better estimation of the effective number of samples could be obtained if we know the data distribution. In the future, we plan to extend our framework by incorporating assumptions on the data distribution or designing learning-based, adaptive methods.

CHAPTER 6
MEASURING DATASET GRANULARITY

Recent advances in deep learning [59] have fueled tremendous progress in visual recognition [99, 66]. However, fine-grained recognition on real-world data still remains challenging [160]. Fine-grained recognition is often regarded as a sub-field of object recognition that aims to distinguish subordinate categories within entry-level categories. Examples include recognizing natural categories such as species of birds [167] and breeds of dogs [89]; or man-made categories such as fashion attributes [111] and car make & model [95]. There are also recent efforts focusing on fine-grained recognition in data collected from the real-world that are highly imbalanced [116, 29]. Despite the increasing visibility of fine-grained recognition as a sub-field, it has thus far lacked a precise definition.

Given the scope and the vagueness of the problem, it is unrealistic to expect an explicit definition of fine-grained that is universally applicable to a wide-range of scenarios. Therefore, in this chapter, we instead pursue a framework for quantitatively measuring dataset granularity, wherein high granularity means fine-grained and low granularity means coarse-grained. The notion of dataset granularity only applies when the dataset is *labeled*. For unlabeled data, in extreme cases, we could label all samples in the whole dataset as a generic class, or label each sample as an instance-level class. Other than the data samples and their labels, we argue that dataset granularity should also depend on the distance function we choose. The distance function tells us how to represent the data samples and calculate distances between them. Examples of distance functions include Euclidean distance between features extracted from deep networks, Hamming distance between binary encoding of attributes, *etc*. The granularity of a dataset will be different if we change how we represent samples and calculate their distances. This is aligned with the notion of fine-grained in human perception. Two visually similar bird species are difficult to distinguish for ordinary people but easy for bird experts [159]. People with congenital amusia have difficulty processing fine-grained pitch in music and speech [155]. In this respect, using different distance functions on a dataset is analogous to different levels of domain expertise.

We define a dataset granularity measure as a function that maps a labeled dataset and a distance function to a real number indicating the dataset granularity. Considering there might be multiple reasonable measures, we seek a set of broad, intuitive notions for which we have a consensus on what a dataset granularity measure should capture. Inspired by the seminal work of Kleinberg [90] and Ben-David and Ackerman [11] on clustering theory, we propose an axiomatic framework comprising three desired properties for a dataset granularity measure: scale invariance, isomorphism invariance, and granularity consistency. Then, we show these properties are self-consistent by providing a few examples of dataset granularity measures.

To assess the quality of dataset granularity measures, we use simulated data and datasets with hierarchical labels. We assume every time when we merge subordinate classes into a higher-level super-class, the granularity should be strictly smaller. We use this information as an oracle to empirically evaluate and select the best dataset granularity measure for the rest of our experiments. With the selected measure, we investigate the interplay between dataset granularity and a variety of factors on commonly used datasets. We find it is more difficult to learn on more fine-grained datasets with higher granularities. In addition, dataset granularity correlates well with transfer learning performance, indicating deep features are more difficult to generalize to fine-grained datasets with higher granularities. We also perform robustness studies and find that dataset granularity is relatively insensitive to changes in network architecture, but

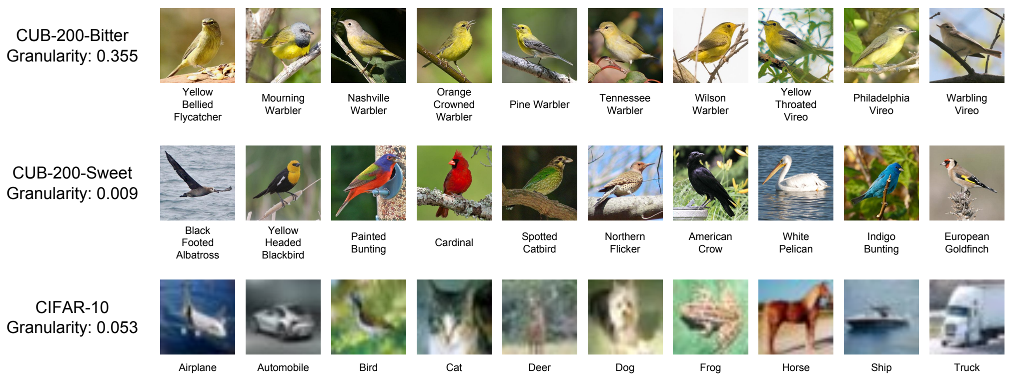


Figure 6.1: Using our measure of dataset granularity, we show two subsets from CUB-200 with 10 classes that are fine-grained (CUB-200-Bitter) and coarse-grained (CUB-200-Sweet). CIFAR-10, which is widely considered coarse-grained, is more fine-grained than CUB-200-Sweet. We represent a class by a randomly selected image from that class in this figure.

sensitive to the choice of pre-trained data. Dataset granularity is also sensitive to nuisance factors arising during image acquisition, including noise and reduced resolution. As a consequence of our study, we show that certain datasets that are widely considered fine-grained in fact have lower granularity than other datasets generally considered coarse-grained. Figure 6.1 shows that based on our proposed dataset granularity measure, a subset from CUB-200 [167] has much lower granularity than CIFAR-10 [98].

Our key contributions can be summarized as follows: (1) We propose a theoretical framework to measure dataset granularity that defines what are the properties that a dataset granularity measure should have. (2) We provide examples of dataset granularity measures and evaluate their quality with hierarchical labeled datasets. (3) We perform a comprehensive empirical study on the characteristics of dataset granularity and their relationship to learning difficulty, generalization and robustness to nuisance factors. Based on our observations, the proposed granularity measure can be used to indicate learning difficulty and transfer learning performance. It can be also used for dataset analysis to find “sweet” and “bitter” subsets with drastically different granularities as shown in Figure 6.1.

6.1 Related Work

Learning from Fine-Grained Visual Data. Recent work on learning from fine-grained visual data focus on designing a deep network in an end-to-end fashion that incorporates modules for better capturing subtle visual differences. Examples of such modules include non-linear feature interactions [110, 49, 33] and attention mechanisms [78, 174, 47, 191, 149]. More recently, efforts have been made to understand the characteristics of learning from fine-grained visual data. Some observations include: (1) deep networks for fine-grained recognition benefit from higher input image resolution [78, 110, 30]; (2) transfer learning to domain-specific fine-grained datasets benefits from networks pre-trained on an adaptively selected subset from the source domain [54, 30, 121]; (3) ImageNet pre-training appears to have only marginal benefits for fine-grained datasets when their labels are not well-represented in ImageNet [93]; (4) Low-shot learning is more difficult on fine-grained dataset, where the dataset granularity is empirically measured by the mean Euclidean distance between class centroids [94]. Our work differs from them by focusing on how to measure dataset granularity and then performing extensive quantitative studies using the proposed measure.

Clustering Theory and Evaluation. The notion of clustering arises naturally whenever one aims to group similar objects together. Despite this intuitively compelling goal, a unified theoretical framework for clustering is difficult to find [90]. However, it is relatively easy to construct a theoretical framework for clustering quality measures [11]. Commonly used metrics for clustering quality evaluation can be divided into two groups: supervised and unsupervised. Supervised clustering measures including Normalized Mutual Information (NMI), the Rand index [74], the V-measure [137] and the Fowlkes-Mallows index (FMI) [45] compare the clustering assignment against ground-truth labels. Unsupervised clustering measures [138, 20, 35, 8, 75] evaluate the intrinsic quality of clustering results. In this chapter, build upon seminal work in [90, 11], we regard a labeled dataset as a clustering assignment and develop a theoretical framework that defines desired properties for a dataset granularity measure. Note that some commonly used unsupervised clustering measures, including the Silhouette index [138], the Calinski-Harabaz index [20] and the Davies-Bouldin index [35], do not satisfy our desired properties. Please refer to Desgraupes [38] for a comprehensive review of clustering evaluation metrics.

6.2 Measuring Dataset Granularity

In this section, we develop an axiomatic framework for measuring dataset granularity. First, we give the problem formulation and mathematical notations. Then we define dataset granularity measure and present three properties as axioms that a dataset granularity measure ought to satisfy. Finally, we show the proposed properties are self-consistent and provide a few examples of dataset granularity measure.

6.2.1 Problem Formulation and Notations

We denote a labeled dataset as $\mathcal{S} = (\mathcal{X}, \mathcal{Y})$, where $\mathcal{X} = \{x_i\}_{i=1}^n$ is a set of samples and $\mathcal{Y} = \{y_i \in \{1, 2, \ldots, k\}\}_{i=1}^n$ is a set of corresponding labels [1]. n denotes the number of samples and k is the number of classes. The set of labels $\mathcal{Y}$ divides samples $\mathcal{X}$ into k

[1]We assume there is exactly one ground-truth label for a sample.

classes $C = \{C_1, C_2, \ldots, C_k\}$, where $C_i = \{x_j \mid y_j = i,\ x_j \in \mathcal{X},\ y_j \in \mathcal{Y}\}$, $C_i \cap C_j = \emptyset$ for $i \neq j$ and $\cup_{i=1}^{k} C_i = \mathcal{X}$.

A distance function $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ defines the similarity between two samples, such that $\forall\ x_i, x_j \in \mathcal{X}$, $d(x_i, x_j) \geq 0$, $d(x_i, x_j) = d(x_j, x_i)$ and $d(x_i, x_j) = 0$ if and only if $x_i = x_j$. Note that we do not require distance functions to be the metrics satisfying the triangle inequality. In our experiments, we use distance functions as the Euclidean distance between features extracted from deep networks pre-trained on different data.

We define a *dataset granularity measure* as a function $g : \mathcal{S} \times d \mapsto \mathbb{R}$ that maps a labeled dataset and a distance function to a real number indicating the dataset granularity. Larger number of $g(\mathcal{S}, d)$ means finer granularity.

6.2.2 Properties of Dataset Granularity Measures

We propose a set of three desired properties as axioms that a dataset granularity measure ought to satisfy. The idea is to characterize what a dataset granularity measure should look like and prevent trivial measures.

The first property we would like to have is the scale invariance with respect to the distance function.

Definition 2 (Scale Invariance)**.** For any distance function d and any $\alpha > 0$, we have $g(\mathcal{S}, d) = g(\mathcal{S}, \alpha d)$.

The scale invariance property requires the dataset granularity measure to be invariant to changes in the units of distance function. This property can be easily satisfied by normalizing the feature vector before calculating distance (*e.g.*, ℓ_2-normalization for Euclidean distance).

Another nice property is that a dataset granularity measure should remain the same when we permute the class indexes. Formally speaking, we define $\mathcal{S}' = (\mathcal{X}', \mathcal{Y}')$ as an *isomorphic transformation* of $\mathcal{S} = (\mathcal{X}, \mathcal{Y})$ if $\mathcal{X}' = \mathcal{X}$ and $\mathcal{Y}' = \{\sigma(y_i)\}_{i=1}^n$, where $\mathcal{Y} = \{y_i \in \{1, 2, \ldots, k\}\}_{i=1}^n$ and $\sigma(.)$ is a permutation of $\{1, 2, \ldots, k\}$. We require the measure to be invariant to any isomorphic transformations.

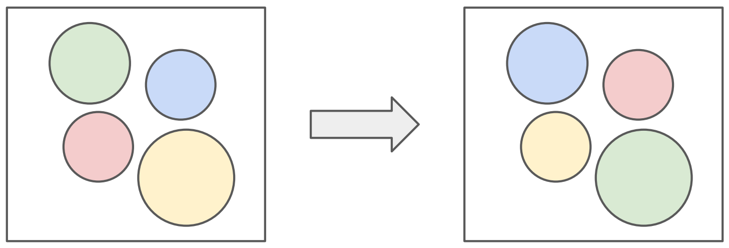
Definition 3 (Isomorphism Invariance)**.** Suppose dataset $\mathcal{S}'$ is a isomorphic transformation of dataset $\mathcal{S}$. For any $\mathcal{S}'$, $g(\mathcal{S}', d) = g(\mathcal{S}, d)$.

Other than the above two invariance properties, we also want to have a property that captures the concept of making a dataset less fine-grained. Intuitively, a labeled dataset is fine-grained if the distance between intra-class samples is relatively large compared with the distance between inter-class samples. Therefore, if we reduce the intra-class distances and enlarge the inter-class distances, the dataset should have smaller granularity. This could be done by using a distance function that better represents the data and makes samples well clustered.

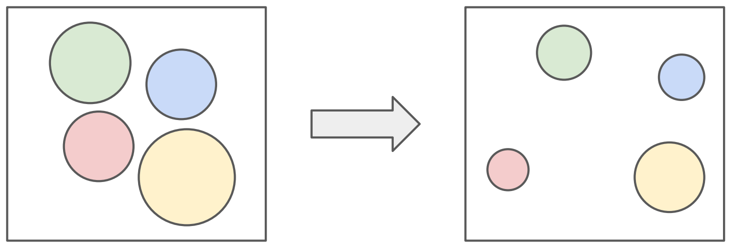
Formally, we define d' as a *granularity consistent transformation* of d if $\forall\ x_i, x_j \in \mathcal{X}$ in the same class of C, we have $d'(x_i, x_j) \leq d(x_i, x_j)$; and $\forall\ x_i, x_j \in \mathcal{X}$ in different classes of C, we have $d'(x_i, x_j) \geq d(x_i, x_j)$.

Definition 4 (Granularity Consistency)**.** Suppose distance function d' is a granularity consistent transformation of distance function d. For any d', $g(\mathcal{S}, d') \leq g(\mathcal{S}, d)$.

The definition of granularity consistency is inspired by the consistency property proposed in Kleinberg [90] and Ben-David and Ackerman [11]. Figure 6.2 illustrates the isomorphic transformation, under which the granularity should remain unchanged, and the granularity consistent transformation, under which the granularity should be smaller.



(a) Isomorphic Transformation



(b) Granularity Consistent Transformation

Figure 6.2: Illustration of isomorphic transformation and granularity consistent transformation on a dataset with 4 classes, denoted by 4 disks with different sizes and colors. We show the original dataset $\mathcal{S}$ on the left and the transformed dataset $\mathcal{S}'$ on the right. In isomorphic transformation, we permute the class indexes. In granularity consistent transformation, we reduce the within-class distances and enlarge the between-class distances.

6.2.3 Examples of Dataset Granularity Measures

To show that scale invariance, isomorphism invariance and granularity consistency are self-consistent, we present examples of dataset granularity measures that satisfy all these desired properties.

The Silhouette index [138] can be calculated as:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{b(x_i)-a(x_i)}{\max\left(a(x_i),b(x_i)\right)}, \tag{6.1}$$

where $a(x_i) = \frac{1}{|C|-1}\sum_{x\in C, x\neq x_i} d(x_i, x)$ is the mean intra-class distance between x_i and other samples in the same class C that x_i belongs to, and $b(x_i)$ is the mean distance between x_i and samples from the nearest class that x_i is not belonging to. Formally, $b(x_i) =$

$\min_{C' \neq C} d(x_i, C')$, where $d(x_i, C') = \frac{1}{|C'|} \sum_{x \in C'} d(x_i, x)$. The silhouette index of a dataset is the mean of silhouette index for all samples in this dataset.

The silhouette index satisfies scale invariance and isomorphism invariance but fails to satisfy granularity consistency. We present a revised version of the silhouette index that satisfies all desired properties.

The Revised Silhouette index (RS). We define the revised silhouette index as the ratio between the mean intra-class distance and the mean nearest-class distance:

$$\mathrm{RS}(\mathcal{S}, d) = \frac{1}{n} \sum_{i=1}^{n} \frac{a(x_i)}{b(x_i)}, \tag{6.2}$$

where $a(x_i)$ and $b(x_i)$ are same as defined in the original silhouette index. RS can be calculated in $O(n^2)$ time.

The Revised Silhouette with Medoids index (RSM). To make the computation of RS faster, we use the medoid to represent each class. The revised silhouette with medoids index is defined as:

$$\mathrm{RSM}(\mathcal{S}, d) = \frac{1}{n} \sum_{i=1}^{n} \frac{d(x_i, c_{x_i})}{d(x_i, c'_{x_i})}, \tag{6.3}$$

where c_{x_i} is the medoid of the ground-truth class of x_i and c'_{x_i} is the medoid of the nearest class other than the ground-truth class of x_i (*i.e.*, $c'_{x_i} = \arg\min_{c, c \neq c_{x_i}} d(c, x_i)$, c is the medoid of a class). RSM can be calculated in $O(nk)$ time, where k is the number of classes.

The Ranking index (Rank). Inspired by the Precision-Recall curve and mean Average Precision used in information retrieval, we define the ranking index as:

$$\mathrm{Rank}(\mathcal{S}, d) = \frac{1}{n} \sum_{i=1}^{n} \frac{n}{n-1} \Big(1 - \sum_{j=1}^{|R_i|} \frac{j}{R_{ij}}\Big), \tag{6.4}$$

where R_i is the list of ranks for samples in the same class of x_i, sorted by their distance between x_i. Specifically, $\sum_{j=1}^{|R_i|} \frac{j}{R_{ij}}$ is the Average Precision (*i.e.*, the area under Precision-

Recall curve) used in information retrieval. We add a term $\frac{n}{n-1}$ to make sure the range of Rank$(\mathcal{S}, d) \in [0, 1]$. Rank can be calculated in $O(n^2 \log n)$ time.

The Ranking with Medoids index (RankM). Similar to the case of RSM, we use the medoid to present each class for faster computation of Rank index. The ranking with medoids index is defined as:

$$\text{RankM}(\mathcal{S}, d) = \frac{1}{n} \sum_{i=1}^{n} \frac{k}{k-1}\Big(1 - \frac{1}{R_{ic}}\Big), \tag{6.5}$$

where R_{ic} is the rank of the x_i's class medoid among all class medoids and k is the number of classes. We add a term $\frac{k}{k-1}$ to make sure the range of RankM$(\mathcal{S}, d) \in [0, 1]$. RankM can be calculated in $O(nk \log k)$ time.

Proposition 2. Dataset granularity measures RS, RSM, Rank and RankM satisfy scale invariance, isomorphism invariance and granularity consistency.

Proof. The proof is given in supplementary material. □

Instead of the aforementioned measures, there exists other measures that satisfy all desired properties, including the Baker-Hubert Gamma index (BHG) [8, 60] for rank correlation between two vectors and the C index [75] for clustering quality assessment. However, the calculations of BHG and the C index have much higher time complexity ($O(n^4)$ for BHG and $O(n^3)$ for the C index) and thus are not scalable on large datasets. For a more comprehensive analysis of other potential dataset granularity measures and proofs showing that they satisfy our proposed properties, please refer to the supplementary material.

6.3 Assessment of Granularity Measures

We perform experiments to assess the quality of dataset granularity measures RS, RSM, Rank and RankM using simulated data and hierarchical labeled CIFAR-100.

6.3.1 Simulation

To better understand the characteristics of proposed granularity measures, we perform a study using simulated datasets with 2 classes, each class has 1000 samples. Samples for each class are drawn from a 2-dimensional multivariate normal distributions with unit covariance matrix and different means in horizontal axis. Formally, $X_1 \sim \mathcal{N}(\Sigma, \mu_1)$ and $X_2 \sim \mathcal{N}(\Sigma, \mu_2)$, where $\Sigma = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, $\mu_1 = (0, 0)^\top$ and $\mu_2 = (m, 0)^\top$. Figure 6.3 illustrates the simulated datasets with different m. The larger the m is, the more separated the two classes are.

We generate simulated datasets with variant m and calculate their granularity measure scores using Euclidean distance as the distance function. We repeat the this process 100 times and show the mean and standard deviation in Figure 6.4. From the figure we can observe that the variance of the scores are small and in general, granularity measures RS, RSM, Rank and RankM are all able to a give lower score when two classes are further separated.

6.3.2 Hierarchical Labeled Dataset

We evaluate the quality of granularity measure in the context of fine-grained class discovery. Consider a dataset $\mathcal{S}$ annotated with coarse-grained labels and a distance func-

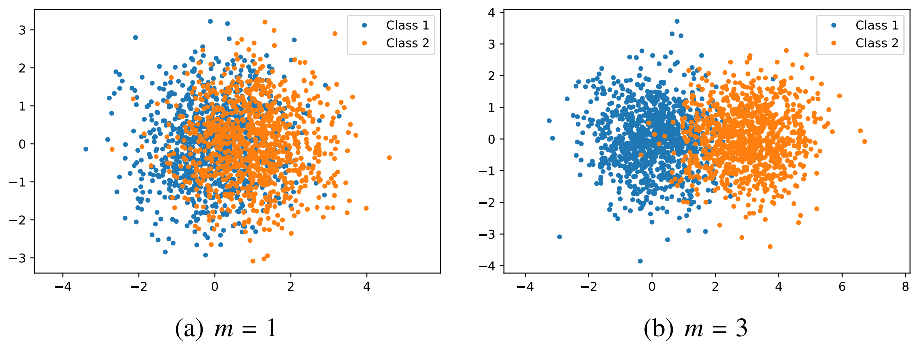


Figure 6.3: Examples of simulated datasets of 2 classes. Samples from each class are draw from a 2-dimensional multivariate normal distributions with unit covariance matrix. m denotes the distance between the means of two classes.

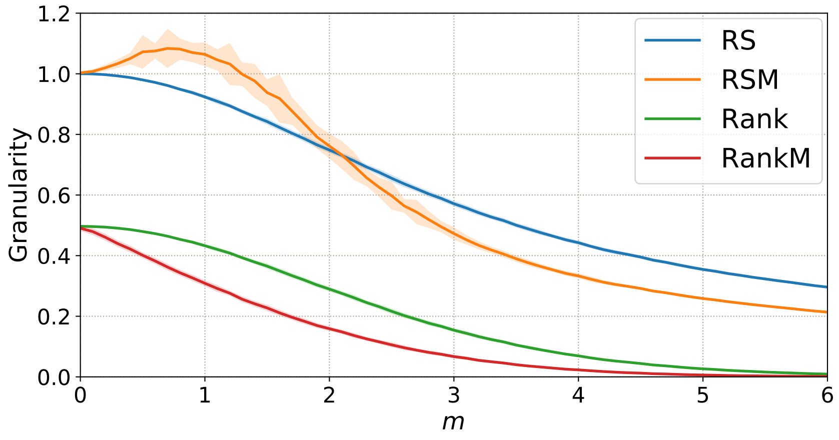


Figure 6.4: Dataset granularity measures RS, RSM, Rank and RankM on simulated dataset with variant m. The solid line denotes mean and the shaded region represents standard deviation.

tion d as the Euclidean distance between features extracted from a deep network, when we re-label samples from a coase-grained class with fine-grained sub-classes, the granularity $g(\mathcal{S}, d)$ should be higher. We build the oracle of relative dataset granularity based on such information and use it to evaluate granularity measures.

We use CIFAR-100 dataset [98] with 20 coarse-grained classes, each containing 5

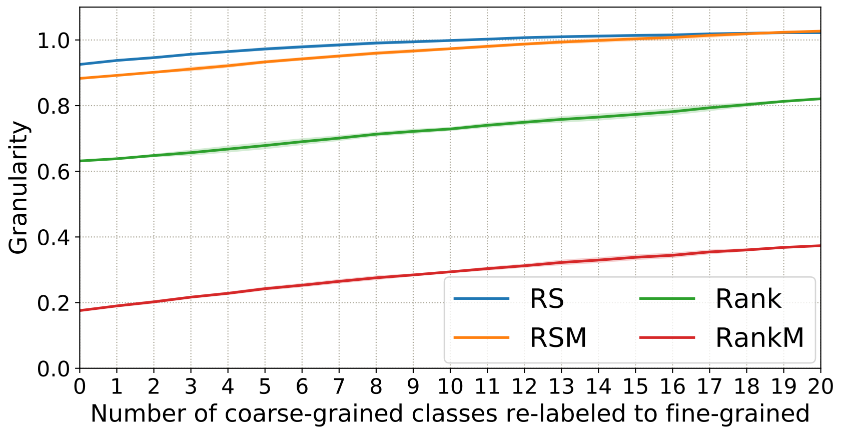


Figure 6.5: Dataset granularity measures RS, RSM, Rank and RankM on CIFAR-100 dataset with increasing number of coarse-grained classes re-labeled to fine-grained. The solid line denotes mean and the shaded region represents standard deviation.

fine-grained classes. We train a ResNet-20 [66] using labels of 20 coarse-grained classes on the CIFAR-100 training set and then extract features on the test set. When calculating granularity scores, we gradually re-label each coarse-grained class until all 20 coarse-grained classes are re-labeled into 100 fine-grained classes. To capture the randomness when deciding the order of coarse-grained class re-labeling, we shuffle the re-labeling order and repeat the experiments 100 times. Figure 6.5 shows the granularity score on CIFAR-100 in terms of how many coarse-grained classes are re-labeled into fine-grained ones. We find scores from all granularity measures are monotonic increasing, consistent with the oracle. Based on results in Figure 6.4 and Figure 6.5, RankM has a smoother change of granularity scores and relatively low computational time complexity compared with others. Therefore, we choose to use *the Ranking with Medoids index (RankM)* as the dataset granularity measure. Unless otherwise stated, we use the term “granularity” of a dataset $\mathcal{S}$ and a distance function d to refer to RankM($\mathcal{S}, d$) in the rest of this chapter.

6.4 Experiments

In this section, we first introduce the datasets and network architectures we used in Section 6.4.1. Then, we discuss the choice of distance function based on pre-trained deep networks in Section 6.4.2.

With a measure of dataset granularity in hand, we can proceed to investigate its interplay with a variety of factors on a range of real-world datasets, including learning difficulty in Section 6.4.3, transferability in Section 6.4.4 and nuisance factors during image capturing in Section 6.4.5.

6.4.1 Experiment Setup

We perform experiments on 9 datasets (Oxford Flowers-102 [123], CUB200-2011 Birds [167], FGVC Aircraft [118], Stanford Cars [95], Stanford Dogs [89], NABirds [159], Food-101 [15], CIFAR-10 and CIFAR-100 [98]) using distance function as the Euclidean distance between features extracted from deep networks. Dataset granularity is calculated using features in the test set. Following the convention, on CIFAR, we use a ResNet-20 trained from scratch on the training set; on the rest of 7 datasets, we use a deep networks pre-trained on large-scale datasets including iNaturalist-2017 [160], ImageNet [36, 140] and ImageNet + iNaturalist, which is the combined dataset of ImageNet and iNaturalist-2017 with 6,089 classes [30]. Table 6.1 summarizes the number of classes, together with number of samples in the training and test splits for all datasets we used.

Dataset	# class	# train / test
Oxford Flowers-102 [123]	102	2,040 / 6,149
CUB200-2011 Birds [167]	200	5,994 / 5,794
FGVC Aircraft [118]	100	6,667 / 3,333
Stanford Cars [95]	196	8,144 / 8,041
Stanford Dogs [89]	120	12,000 / 8,580
NABirds [159]	555	23,929 / 24,633
CIFAR-10 [98]	10	50,000 / 10,000
CIFAR-100 [98]	100	50,000 / 10,000
Food-101 [15]	101	75,750 / 25,250
iNaturalist-2017 [160]	5,089	579,184 / 95,986
ImageNet [36, 140]	1,000	1,281,167 / 50,000
ImageNet + iNaturalist	6,089	1,860,351 / 145,986

Table 6.1: Datasets used in measuring dataset granularity, where iNaturalist-2017 ImageNet and ImageNet + iNaturalist are used to pre-train models for feature extraction.

6.4.2 On the Choice of Distance Function

We calculate distance between samples based on features extracted from a deep network. To understand how sensitive the dataset granularity is in terms of the different choices of distance functions, we perform experiments using deep networks with different architectures pre-trained on ImageNet, including ResNet [66], DenseNet [73] and Inception-V3 [152].

From results in Figure 6.6, we observe that dataset granularity is not very sensitive to the choices of network architecture. In general, using DenseNets yields lower granularity compared with using ResNets and Inception-V3. Choosing different network architecture doesn't change the relative order of dataset granularity. However, dataset granularity is sensitive to the choice of pre-trained data. Using a network pre-trained on different data could drastically change the granularity. For more details on the granularity in terms of pre-trained data, please refer to Section 6.4.4 and Figure 6.10.

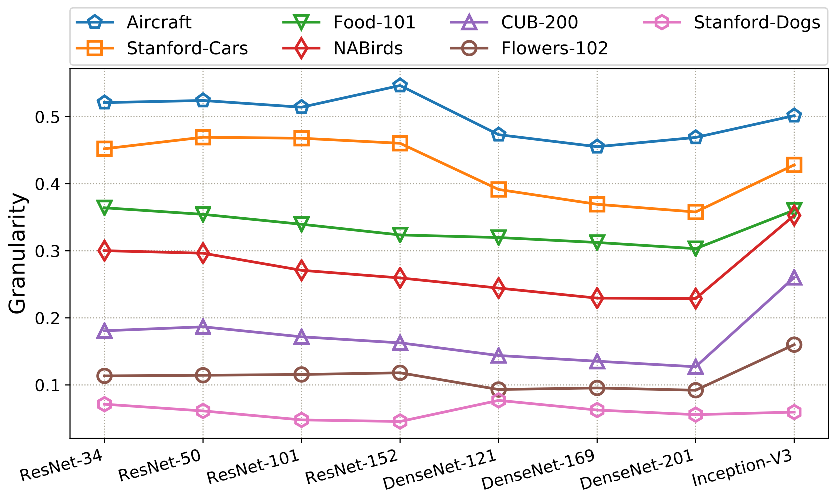


Figure 6.6: Dataset granularity using features extracted from different networks pre-trained on ImageNet.

6.4.3 Granularity versus Learning Difficulty

Intuitively, learning on fine-grained data is more difficult compared with coarse-grained data. To quantitatively understand the relationship between learning difficulty and granularity, we use the training error rate of a linear logistic regression model (LR) trained with deep features as an estimate of training difficulty.

We conduct comprehensive experiments on CIFAR. Specifically, we train a ResNet-20 from scratch on the training set. We then extract features on the test set and use them to train a LR and measure granularity and the training error rate. Figure 6.7 presents results on CIFAR datasets, including CIFAR-10 and CIFAR-100 with variant number of coarse-grained classes re-labeled as fine-grained. Further more, we examine the case of binary classification by forming a dataset with two specific classes for each pair of classes in CIFAR-10 and CIFAR-100, results are shown in Figure 6.8. We observe that dataset granularity is highly correlated with learning difficulty. In addition, we find that machine's perception of granularity makes sense to human: (“cat”, “dog”),

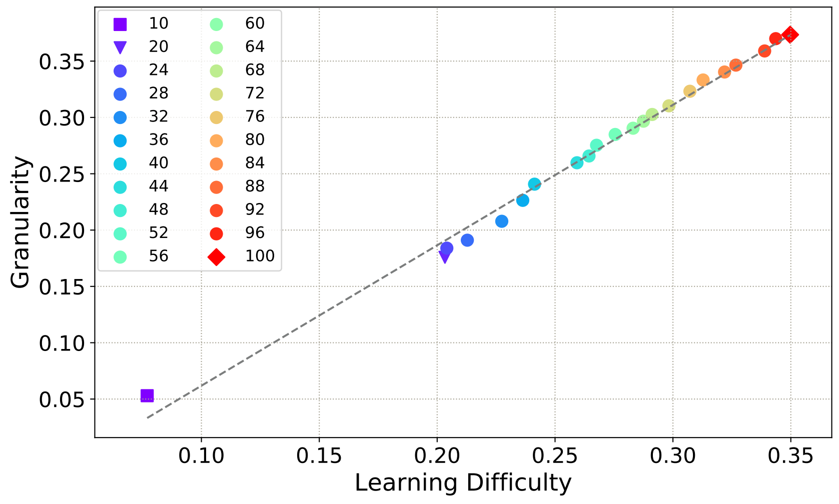
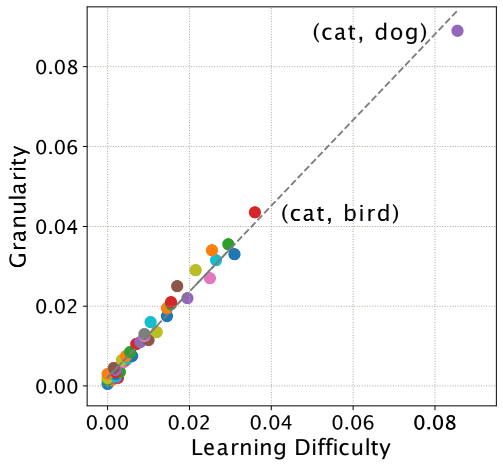


Figure 6.7: Dataset granularity correlates well with difficulty (training error rate using linear logistic regression). The purple square marker represents CIFAR-10 dataset. Other markers represent the CIFAR-100 with different number of classes (by re-labeling coarse-grained classes with fine-grained), where the purple triangle and the red diamond denote CIFAR-100 with 20 and 100 coarse-grained labels respectively.

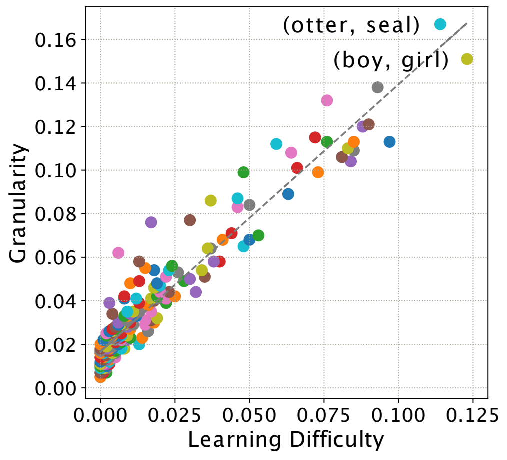
("otter", "seal") and ("boy", "girl") are the most fine-grained pairs of classes and are also relatively difficult to differentiate for human.

Qualitatively, we show t-SNE embeddings [114] of subsets with drastically different granularities from CIFAR-10 and CIFAR-100 in Figure 6.9. Classes with low granularity are well separated, whereas classes with high granularity are mixed together.

Other than linear logistic regression, we also tried to use the training error rate of deep networks to indicate the learning difficulty. However, we find deep networks tend to have close to 0 training error rate on all datasets. As pointed out by Zhang *et al.* [186], deep network can easily fit a training set, especially when the number of parameters exceeds the number of data points as it usually does in practice. In such scenario, if we want to measure the learning difficulty of a deep network, we need to seek for other metrics such as the intrinsic dimension [104] of the network.



(a) CIFAR-10 Pairs (ρ = 0.992)



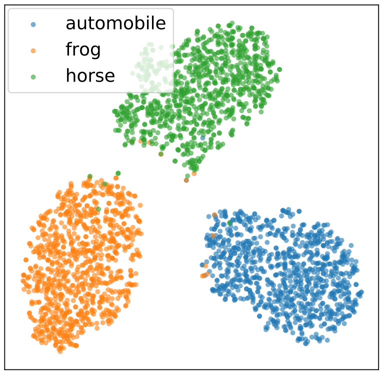
(b) CIFAR-100 Pairs (ρ = 0.958)

Figure 6.8: Training difficulty versus granularity on all pairs of classes from CIFAR. The number in the bracket denotes the Pearson's ρ correlation coefficient.

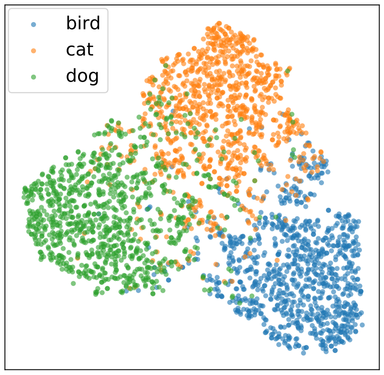
6.4.4 Granularity versus Transferability

Transferring features learned from large-scale datasets to small-scale datasets has been extensively used in our field. To examine the relationship between dataset granularity and feature transferability, we train Inception-V3 networks from scratch on 3 large-scale datasets: ImageNet, iNaturalist-2017 and ImageNet + iNaturalist respectively. Then, we transfer the learned features to 7 datasets. Specifically, for each dataset, we extract features from pre-trained Inception-V3. Then we train a linear logistic regression model using training set features. We evaluate the performance of transfer learning by the error rate on the test set. The dataset granularity is calculated based on test set features.

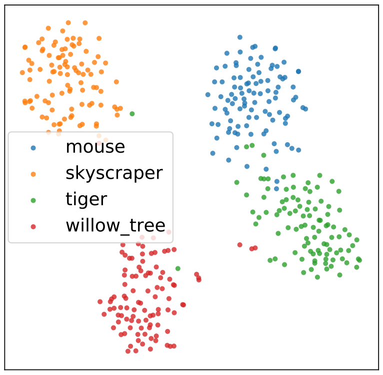
From results in Figure 6.10, we have the following observations: (1) In general, dataset granularity correlates well with transfer learning performance (with Pearson's ρ = 0.980). Therefore, transferring features to a dataset with lower granularity is easier. And on the contrary, transferring to a dataset with higher granularity is harder. Dataset



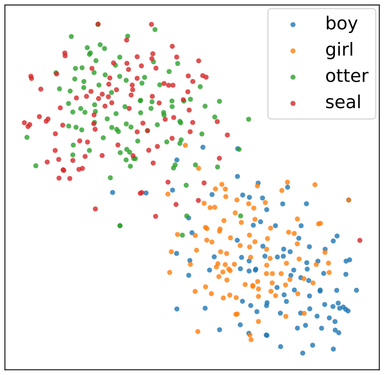
(a) CIFAR-10-Sweet (0.003)



(b) CIFAR-10-Bitter (0.073)



(c) CIFAR-100-Sweet (0.021)



(d) CIFAR-100-Bitter (0.215)

Figure 6.9: t-SNE embedding of subsets from CIFAR-10 and CIFAR-100. The granularity is shown in the bracket.

granularity could be used as an indicator for the transfer learning performance. (2) The choice of pre-trained data has a huge effective on the dataset granularity. For example, the granularity of Stanford-Dogs is about 8× lower when using ImageNet pre-trained features instead of iNaturalist-2017; however, the granularity of NABirds is about 5× lower when using iNaturalist-2017 pre-trained features instead of ImageNet. This is illustrated qualitatively by t-SNE embeddings in Figure 6.11. A key to make a dataset less fine-grained is to use better features extracted from a network pre-trained on more appropriate data. This has beed verified by recent work on domain-specific adaptive transfer learning [54, 30, 121]

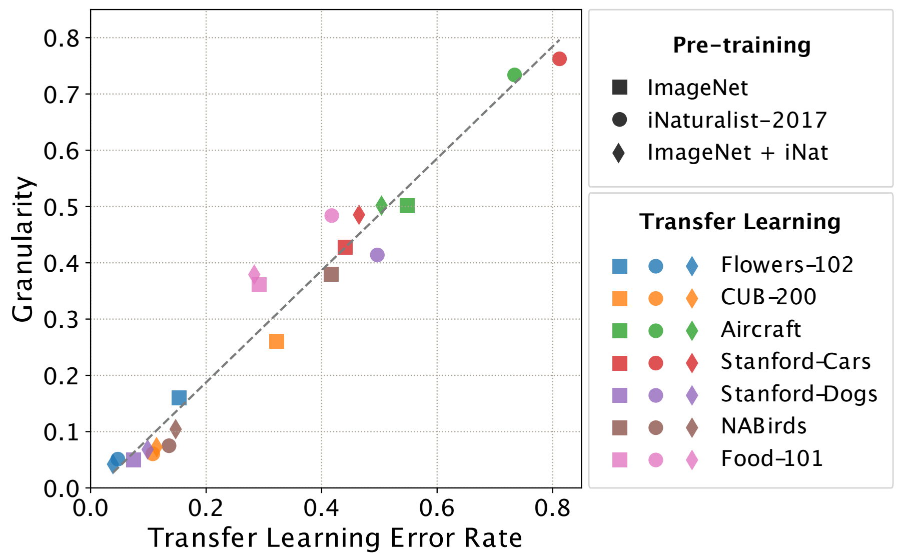
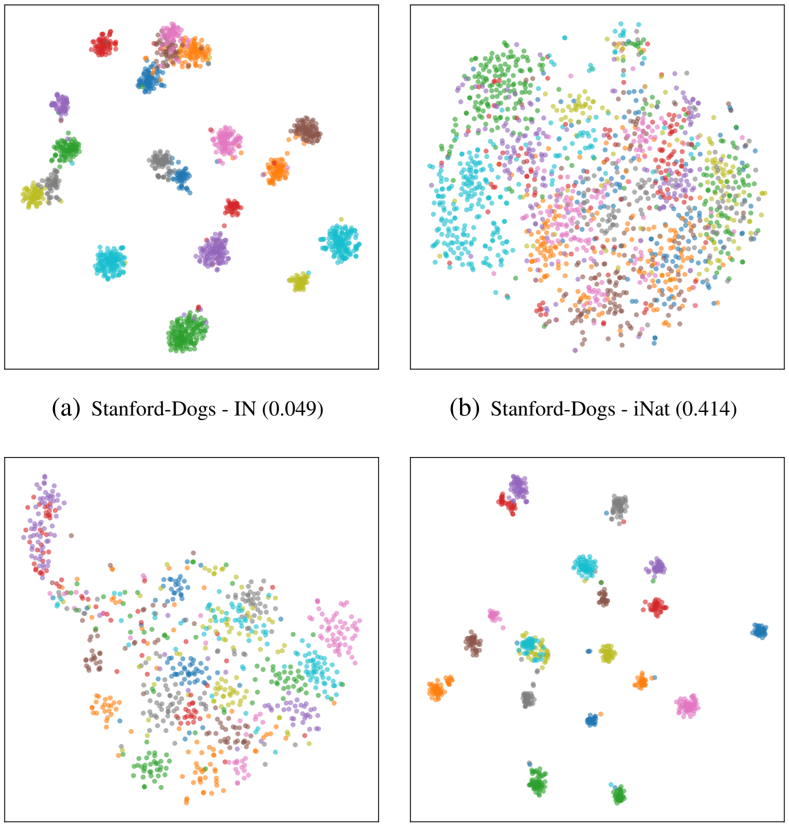


Figure 6.10: Transfer learning performance and granularity on 7 datasets. Each dataset is represented by markers with the same color. We use markers with different shapes to represent pre-training on different datasets.

6.4.5 Granularity versus Nuisance Factors

Ideally, to distinguish classes with subtle visual difference, one could zoom in and leverage information from local discriminative regions. However, nuisance factors such as noise, motion blur, occlusion *etc*. arise during image acquisition in real-world. Furthermore, due to physical constraints of imaging devices, one can not get sharp images with arbitrary high spatial resolution. To examine how nuisance factors affects dataset granularity, we add Gaussian noise and Salt & Pepper noise to images before feature extraction. In addition, we reduce the image resolution by re-sizing the shorter edge of the image and keeping the aspect ratio. Figure 6.12 shows dataset granularity on CUB-200 using ImageNet pre-trained ResNet-50 features. Dataset granularity is very sensitive to noise and reduced resolution. In addition, nuisance factors may change the relative order of granularity. For example, NABirds becomes more fine-grained than



(a) Stanford-Dogs - IN (0.049) (b) Stanford-Dogs - iNat (0.414)

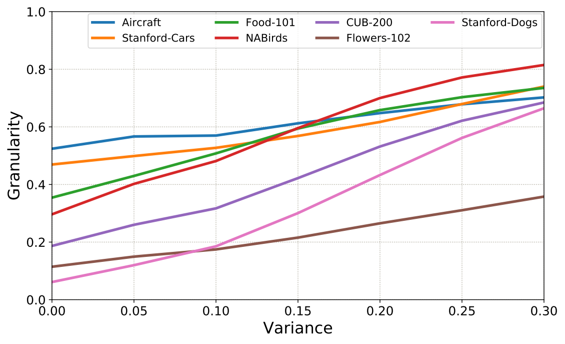
(c) NABirds - IN (0.380) (d) NABirds - iNat (0.075)

Figure 6.11: t-SNE embeddings of first 20 classes from Stanford-Dogs and NABirds, with ImageNet pre-training (IN) on the left and iNaturalist-2017 (iNat) pre-training on the right. The granularity is shown in the bracket.

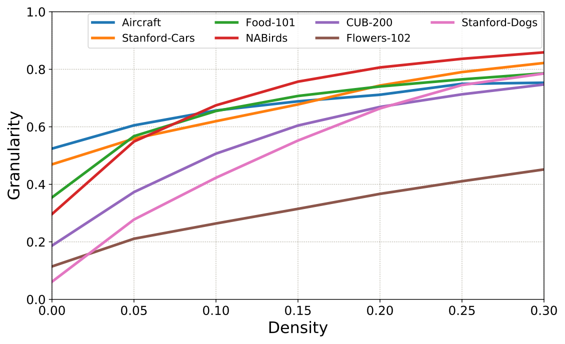
Aircraft, Stanford Cars and Food-101 as we increasing the noise and reducing the image resolution.

6.5 Conclusion

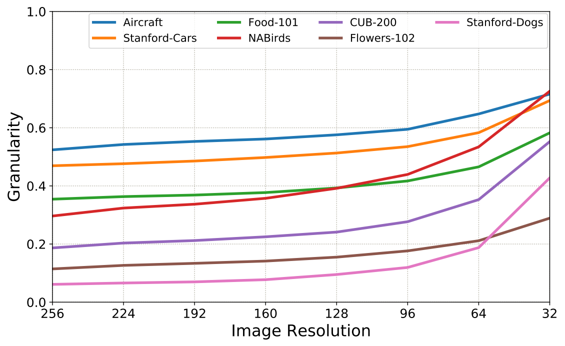
In this chapter, as a step toward a precise definition of fine-grained recognition, we present an axiomatic framework based on clustering theory for measuring dataset gran-



(a) Gaussian Noise



(b) Salt & Pepper Noise



(c) Reduced Resolution

Figure 6.12: Dataset granularity of CUB-200 with respect to nuisance factors including Gaussian noise, Salt & Pepper noise and reduced resolution. The datasets become more fine-grained when we increase the level of noise and reduce the image resolution.

ularity. We define a dataset granularity measure as a function that maps a labeled dataset and a distance function to a real number and then describe three desirable properties for the measure: scale invariance, isomorphism invariance, and granularity consistency. We then give four examples of granularity measures that satisfy all properties and choose the Ranking with Medoids index (RankM) based on assessment on simulated data and a hierarchical labeled CIFAR dataset. In our experiments, we show that dataset granularity correlates well with learning difficulty and transfer learning performance. In addition, dataset granularity is sensitive to the choice of pre-training data and nuisance factors, but insensitive to the choice of network.

Future directions include: (1) defining other axiomatic properties for granularity measures; (2) seeking more examples of granularity measures; (3) investigating the interplay between dataset granularity and other factors including label noise, number of samples or in different scenarios including low-shot learning and adversarial learning.

CHAPTER 7

CONCLUSIONS

In this dissertation, we studied several key problems associated with learning from fine-grained and long-tailed visual data. In particular, we focused on quantitatively modeling such data and designing deep networks that learn from such data.

One of the biggest challenge in fine-grained recognition is that the data annotation process often calls for specialized domain knowledge and thus is difficult to scale. The success of recently introduced iNaturalist dataset [160] opens up a promising direction to solve this issue. We feel that by collaborating with communities of domain experts, large amounts of annotated data with high quality could be available for building fine-grained visual recognition systems.

As the number of classes increase, how to deal with long-tailed data distribution and recognize novel classes that are not seen during training will be long-lasting challenges. We showed how to re-balance the loss during training based on the effective number of samples. It would be interesting to explore how to combine re-balancing strategies with few-shot or zero-shot learning for recognizing rare or unseen classes.

Another rather unexplored area is multi-modal learning. For example, two bird species that are visually similar may live in different habitats geographically or generate different sounds. How to learn a joint representation from multi-modal data for fine-grained recognition would be an intriguing direction.

BIBLIOGRAPHY

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

[2] Dimitris Achlioptas, Frank McSherry, and Bernhard Schölkopf. Sampling techniques for kernel methods. In *Advances in neural information processing systems*, 2002.

[3] Anelia Angelova and Shenghuo Zhu. Efficient object detection and segmentation for fine-grained recognition. In *CVPR*, 2013.

[4] Anelia Angelova, Shenghuo Zhu, and Yuanqing Lin. Image segmentation for large-scale subcategory flower recognition. In *WACV*, 2013.

[5] Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, and Trevor Darrell. Deep compositional captioning: Describing novel object categories without paired training data. In *CVPR*, 2016.

[6] Haim Avron, Huy Nguyen, and David Woodruff. Subspace embeddings for the polynomial kernel. In *Advances in neural information processing systems*, 2014.

[7] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *PAMI*, 2016.

[8] Frank B Baker and Lawrence J Hubert. Measuring the power of hierarchical cluster analysis. *Journal of the American Statistical Association*, 1975.

[9] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: Fine-grained image generation through asymmetric training. In *ICCV*, 2017.

[10] Sean Bell and Kavita Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. on Graphics*, 2015.

[11] Shai Ben-David and Margareta Ackerman. Measures of clustering quality: A working set of axioms for clustering. In *Advances in neural information processing systems*, 2009.

[12] Samy Bengio. Sharing representations for long tail computer vision problems. In *ICMI*, 2015.

[13] Thomas Berg and Peter N Belhumeur. Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In *CVPR*, 2013.

[14] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. Polynomial networks and factorization machines: New insights and efficient training algorithms. In *ICML*, 2016.

[15] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014.

[16] S Branson, G Van Horn, P Perona, and S Belongie. Improved bird species recognition using pose normalized deep convolutional nets. In *BMVC*, 2014.

[17] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *ECCV*, 2010.

[18] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 2018.

[19] Sijia Cai, Wangmeng Zuo, and Lei Zhang. Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization. In *ICCV*, 2017.

[20] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 1974.

[21] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012.

[22] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *ICCV*, 2013.

[23] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, 2002.

[24] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *JAIR*, 2002.

[25] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *JMLR*, 2010.

[26] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[27] Aruni Roy Chowdhury, Tsung-Yu Lin, Subhransu Maji, and Erik Learned-Miller. One-to-many face recognition with bilinear cnns. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[28] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015.

[29] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019.

[30] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 2018.

[31] Yin Cui, Yongzhou Xiang, Kun Rong, Rogerio Feris, and Liangliang Cao. A spatial-color layout feature for representing galaxy images. In *WACV*, 2014.

[32] Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *CVPR*, 2016.

[33] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017.

[34] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[35] David L Davies and Donald W Bouldin. A cluster separation measure. *PAMI*, 1979.

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[37] Jia Deng, Jonathan Krause, Michael Stark, and Li Fei-Fei. Leveraging the wisdom of the crowd for fine-grained recognition. *PAMI*, 2016.

[38] Bernard Desgraupes. Clustering indices. *University of Paris Ouest-Lab ModalX*, 2013.

[39] Sander Dieleman, Kyle W Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 2015.

[40] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[41] Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *ICCV*, 2017.

[42] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *ICML Workshop*, 2003.

[43] Charles Elkan. The foundations of cost-sensitive learning. In *IJCAI*, 2001.

[44] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[45] Edward B Fowlkes and Colin L Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 1983.

[46] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 1997.

[47] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, 2017.

[48] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*. ACL, 2016.

[49] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *CVPR*, 2016.

[50] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis and

the controlled generation of natural stimuli using convolutional neural networks. In *Advances in neural information processing systems*, 2015.

[51] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[52] Efstratios Gavves, Basura Fernando, Cees GM Snoek, Arnold WM Smeulders, and Tinne Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, 2013.

[53] Efstratios Gavves, Basura Fernando, Cees GM Snoek, Arnold WM Smeulders, and Tinne Tuytelaars. Local alignments for fine-grained categorization. *International Journal of Computer Vision*, 111(2):191–212, 2014.

[54] Weifeng Ge and Yizhou Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *CVPR*, 2017.

[55] Timnit Gebru, Judy Hoffman, and Li Fei-Fei. Fine-grained recognition in the wild: A multi-task domain adaptation approach. In *ICCV*, 2017.

[56] Yonatan Geifman and Ran El-Yaniv. Deep active learning over the long tail. *arXiv preprint arXiv:1711.00941*, 2017.

[57] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[58] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, 2004.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[60] Leo A Goodman and William H Kruskal. Measures of association for cross classifications. In *Measures of association for cross classifications*. Springer, 1979.

[61] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin. Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 2014.

[62] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large

minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[63] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[64] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks*, 2008.

[65] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, 2008.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[68] Xiangteng He and Yuxin Peng. Fine-graind image classification via combining vision and language. In *CVPR*, 2017.

[69] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[70] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015.

[71] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[72] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016.

[73] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[74] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 1985.

[75] Lawrence J Hubert and Joel R Levin. A general statistical framework for assessing categorical clustering in free recall. *Psychological bulletin*, 1976.

[76] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? In *Advances in neural information processing systems Workshop*, 2016.

[77] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[78] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, 2015.

[79] Svante Janson. Random coverings in several dimensions. *Acta Mathematica*, 1986.

[80] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 2002.

[81] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM international conference on Multimedia*. ACM, 2014.

[82] Edward Johns, Oisin Mac Aodha, and Gabriel J Brostow. Becoming the expert-interactive multi-class machine teaching. In *CVPR*, 2015.

[83] Lucas N Joppa, David L Roberts, and Stuart L Pimm. How many species of flowering plants are there? *Proceedings of the Royal Society B: Biological Sciences*, 2011.

[84] Herman Kahn and Andy W Marshall. Methods of reducing sample size in monte carlo computations. *Journal of the Operations Research Society of America*, 1953.

[85] Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In *AISTATS*, 2012.

[86] Maurice George Kendall et al. The advanced theory of statistics. *The advanced theory of statistics.*, 1946.

[87] Salman Khan, Munawar Hayat, Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *CVPR*, 2019.

[88] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 2018.

[89] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fgvc: Stanford dogs. In *CVPR Workshop*, 2011.

[90] Jon M Kleinberg. An impossibility theorem for clustering. In *Advances in neural information processing systems*, 2003.

[91] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[92] Shu Kong and Charless Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *CVPR*, 2017.

[93] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019.

[94] Jedrzej Kozerawski and Matthew Turk. Clear: Cumulative learning for one-shot one-class image recognition. In *CVPR*, 2018.

[95] Jan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop*, 2013.

[96] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *CVPR*, 2015.

[97] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *ECCV*, 2016.

[98] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[99] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[100] Neeraj Kumar, Peter N Belhumeur, Arijit Biswas, David W Jacobs, W John

Kress, Ida C Lopez, and João VB Soares. Leafsnap: A computer vision system for automatic plant species identification. In *ECCV*, 2012.

[101] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, 2015.

[102] Natalia Larios, Bilge Soran, Linda G Shapiro, Gonzalo Martínez-Muñoz, Junyuan Lin, and Thomas G Dietterich. Haar random forest features and svm spatial matching kernel for stonefly species identification. In *ICPR*, 2010.

[103] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[104] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *ICLR*, 2018.

[105] Tsung-Yi Lin, Yin Cui, Serge Belongie, and James Hays. Learning deep representations for ground-to-aerial geolocalization. In *CVPR*, 2015.

[106] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *PAMI*, 2018.

[107] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[108] Tsung-Yu Lin and Subhransu Maji. Visualizing and understanding deep texture representations. In *CVPR*, 2016.

[109] Tsung-Yu Lin and Subhransu Maji. Improved bilinear pooling with cnns. In *BMVC*, 2017.

[110] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

[111] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016.

[112] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational effi-

ciency of training neural networks. In *Advances in neural information processing systems*, 2014.

[113] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[114] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.

[115] Oisin Mac Aodha, Shihan Su, Yuxin Chen, Pietro Perona, and Yisong Yue. Teaching categories to human learners with visual explanations. In *CVPR*, 2018.

[116] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

[117] Subhransu Maji and Alexander C Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009.

[118] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[119] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[120] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems*, 2013.

[121] Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V Le, and Ruoming Pang. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*, 2018.

[122] Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006.

[123] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.

[124] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001.

[125] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[126] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *CVPR*, 2016.

[127] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

[128] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[129] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *KDD*, 2013.

[130] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[131] Qi Qian, Rong Jin, Shenghuo Zhu, and Yuanqing Lin. Fine-grained visual categorization via multi-stage metric learning. In *CVPR*, 2015.

[132] Svetlozar T Rachev. The monge–kantorovich mass transference problem and its stochastic applications. *Theory of Probability & Its Applications*, 1985.

[133] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, 2007.

[134] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *CVPR*, 2016.

[135] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

[136] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2012.

[137] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, 2007.

[138] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 1987.

[139] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *IJCV*, 2000.

[140] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[141] Nikolaos Sarafianos, Xiang Xu, and Ioannis A Kakadiaris. Deep imbalanced attribute classification using visual attention aggregation. In *ECCV*, 2018.

[142] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[143] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[144] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*, 2014.

[145] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016.

[146] Marcel Simon, Yang Gao, Trevor Darrell, Joachim Denzler, and Erik Rodner. Generalized orderless pooling performs implicit salient matching. In *ICCV*, 2017.

[147] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[148] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[149] Ming Sun, Yuchen Yuan, Feng Zhou, and Errui Ding. Multi-attention multi-class constraint for fine-grained image recognition. In *ECCV*, 2018.

[150] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[151] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[152] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[153] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[154] Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural computation*, 2000.

[155] Barbara Tillmann, Elena Rusconi, Caroline Traube, Brian Butterworth, Carlo Umilta, and Isabelle Peretz. Fine-grained pitch processing of music and speech in congenital amusia. *The Journal of the Acoustical Society of America*, 2011.

[156] Kai Ming Ting. A comparative study of cost-sensitive boosting algorithms. In *ICML*, 2000.

[157] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

[158] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *PAMI*, 2008.

[159] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *CVPR*, 2015.

[160] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018.

[161] Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.

[162] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[163] Andrea Vedaldi, Siddharth Mahendran, Stavros Tsogkas, Subhransu Maji, Ross Girshick, Juho Kannala, Esa Rahtu, Iasonas Kokkinos, Matthew B Blaschko, David Weiss, et al. Understanding objects in detail with fine-grained attributes. In *CVPR*, 2014.

[164] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *PAMI*, 2012.

[165] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *ICCV*, 2015.

[166] C. Vondrick, D. J. Patterson, and D. Ramanan. Efficiently scaling up crowd-sourced video annotation - A set of best practices for high quality, economical video labeling. *IJCV*, 101(1):184–204, 2013.

[167] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology*, 2011.

[168] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014.

[169] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Neural Information Processing Systems*, 2017.

[170] Jan D Wegner, Steven Branson, David Hall, Konrad Schindler, and Pietro Perona. Cataloging public objects using aerial and street-level images-urban trees. In *CVPR*, 2016.

[171] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 2009.

[172] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, 2001.

[173] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Data-driven 3d voxel patterns for object category recognition. In *CVPR*, 2015.

[174] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and

Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, 2015.

[175] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[176] Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, 2002.

[177] Zhe Xu, Shaoli Huang, Ya Zhang, and Dacheng Tao. Webly-supervised fine-grained visual categorization via deep domain adaptation. *PAMI*, 2016.

[178] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[179] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, 2015.

[180] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for deep face recognition with long-tail data. *arXiv preprint arXiv:1803.09014*, 2018.

[181] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 2014.

[182] Chong You, Chi Li, Daniel P Robinson, and René Vidal. A scalable exemplar-based subspace clustering algorithm for class-imbalanced data. In *European Conference on Computer Vision*, 2018.

[183] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *CVPR*, 2018.

[184] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[185] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[186] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[187] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, 2014.

[188] Ning Zhang, Evan Shelhamer, Yang Gao, and Trevor Darrell. Fine-grained pose prediction, normalization, and recognition. In *ICLR Workshop*, 2016.

[189] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *CVPR*, 2017.

[190] Xiaopeng Zhang, Hongkai Xiong, Wengang Zhou, Weiyao Lin, and Qi Tian. Picking deep filter responses for fine-grained image recognition. In *CVPR*, 2016.

[191] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *ICCV*, 2017.

[192] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *PAMI*, 2017.

[193] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 2006.

[194] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014.

[195] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.

[196] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018.