UNIVERSITY OF CALIFORNIA SAN DIEGO

To Overcome Limitations of Computer Vision Datasets

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Intelligent Systems, Robotics, & Control)

by

Bo Liu

Committee in charge:

Professor Nuno Vasconcelos, Chair
Professor Nikolay Atanasov
Professor Kenneth Kreutz-Delgado
Professor David Kriegman
Professor Truong Nguyen

2021

The dissertation of Bo Liu is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2021

DEDICATION

This thesis is dedicated to my wife, Bo Wang, whose unyielding love and support have inspired and encouraged me to pursue and complete my PhD degree.

EPIGRAPH

Truth is ever to be found in simplicity,

and not in the multiplicity and confusion of things.

—Sir Isaac Newton

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Finishing the PhD degree would be impossible for me, without the help, support, discussion, and company from many people. I would like to express my sincere gratitude to my family, advisor, thesis committee members, mentors, collaborators, colleagues, friends, etc.

At first, I would like to thank my Ph.D. advisor, Professor Nuno Vasconcelos. It was my great honor to work under his supervision. His critical thinking, research philosophy, technical writing/presentation, etc., have reshaped my research mindset. I would also like to thank Professors Kenneth Kreutz-Delgado, Professor David Kriegman, Professor Truong Nguyen, Professor Nikolay Atanasov for being my committee members and their insightful suggestions and comments.

I worked as a research intern in multiple industry research institutions. Those research experiences are insightful and unique. I would like to thank Mandar Dixit and Gang Hua for offering me my first intern experience in Microsoft and their support and help for my research. I would also like to thanks Haoxiang Li, Hao Kang and Gang Hua for being my mentors at Wormpex AI Research. I am also thankful to Roland Kwitt, who is co-author with me.

I also want to thank my colleagues of SVCL who have overlap with me, Mohammad Saberian, Jose Costa Pereira, Weixin Li, Mandar Dixit, Can Xu, Song Lu, Si Chen, Zhaowei Cai, Yingwei Li, Pedro Morgado, Yunsheng Li, Yi Li, Pei Wang, John Ho, Gina Wu, Brandon Leung, Xudong Wang, Gautam Nain, Xiangyun Zhao, Hongyuan Du, Linjun Li, etc. Without their company, the research and life during my Ph.D. study will be much more difficult. Among them, I am particularly thankful to Mandar Dixit who was very helpful when I started to work at SVCL, Xudong Wang, Hongyuan Du, Linjun Li, John Ho, Gina Wu, Brandon Leung and Yi Li who are co-authors with me, and Jose Costa Pereira, Yingwei Li and Pedro Morgado for maintaining the servers at the lab. I also thank Pedro Morgado for those workouts during spare time , and Yunsheng Li for relaxing talks and discussions.

Finally, and most importantly, I would like to thank my family. I am very grateful that my parents were fully supportive that I pursue my Ph.D. degree, and forgave me not going back for years. During me Ph.D. years, I met my wife and married her. I would like to greatly thank my wife for her unconditioned encourage and accompanying.

Chapter 2 is, in full, based on the material as it appears in the publication of “Feature Space Transfer for Data Augmentationn”, Bo Liu, Xudong Wang, Mandar Dixit, Roland Kwitt, and Nuno Vasconcelos, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2018. The dissertation author was the primary investigator and author of this material.

Chapter 3 is, in full, based on the material as it appears in the publication of “Sparse Pose Trajectory Completion”, Bo Liu, Mandar Dixit, Roland Kwitt, Gang Hua, Nuno Vasconcelos, in *arXiv preprint arXiv:2105.00125*. The dissertation author was the primary investigator and author of this material.

Chapter 4 is, in full, based on the material as it appears in the publication of “Few-Shot Open-Set Recognition using Meta-Learning”, Bo Liu, Hao Kang, Haoxiang Li, Gang Hua, Nuno Vasconcelos, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2020. The dissertation author was the primary investigator and author of this material.

Chapter 5 is, in full, based on the material as it appears in the submission of “GistNet: a Geometric Structure Transfer Network for Long-Tailed Recognition”, Bo Liu, Haoxiang Li, Hao Kang, Gang Hua, Nuno Vasconcelos, in *IEEE International Conference on Computer Vision* (ICCV), 2021. The dissertation author was the primary investigator and author of this material.

Chapter 6 is, in full, based on the material as it appears in the submission of “Breadcrumbs: Adversarial Class-Balanced Sampling for Long-tailed Recognition”, Bo Liu, Haoxiang Li, Hao Kang, Gang Hua, Nuno Vasconcelos, in *IEEE International Conference on Computer Vision* (ICCV), 2021. The dissertation author was the primary investigator and author of this material.

Chapter 7 is, in full, based on the material as it appears in the submission of “Semi-supervised Long-tailed Recognition using Alternate Sampling”, Bo Liu, Haoxiang Li, Hao Kang, Nuno Vasconcelos, Gang Hua, in *IEEE International Conference on Computer Vision* (ICCV), 2021. The dissertation author was the primary investigator and author of this material.

VITA

2013	B. S. in Electrical Engineering, Tsinghua University, Beijing, China
2016	M. S. in Electrical Engineering (Intelligent Systems, Robotics, & Control), University of California San Diego
2021	Ph. D. in Electrical Engineering (Intelligent Systems, Robotics, & Control), University of California San Diego

PUBLICATIONS

Hongyuan Du, Linjun Li, **Bo Liu**, Nuno Vasconcelos, “SPOT: Selective Point Cloud Voting for Better Proposal in Point Cloud Object Detection”, In *Proceedings of European Conference on Computer Vision* (ECCV), 2020.

Jordan A. Carlson, **Bo Liu**, James F. Sallis, J. Aaron Hipp, Vincent S. Staggs, Jacqueline Kerr, Amy Papa, Kelsey Dean, and Nuno M. Vasconcelos, “Automated High-Frequency Observations of Physical Activity using Computer Vision”, *Medicine and Science in Sports and Exercise* (MSSE), 2020.

Bo Liu, Hao Kang, Haoxiang Li, Gang Hua, Nuno Vasconcelos, “Few-Shot Open-Set Recognition using Meta-Learning”, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2020.

Chih-Hui Ho, **Bo Liu**, Tz-Ying Wu, Nuno Vasconcelos, “Exploit Clues from Views: Self-Supervised and Regularized Learning for Multiview Object Recognition”, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2020.

Bo Liu, Xudong Wang, Mandar Dixit, Roland Kwitt, Nuno Vasconcelos, “Feature Space Transfer for Data Augmentation”, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2018.

Jordan A. Carlson, **Bo Liu**, James F. Sallis, Jacqueline Kerr, J.Aaron Hipp, Vincent S. Staggs, Amy Papa, Kelsey Dean, Nuno M. Vasconcelos, “Automated ecological assessment of physical activity: Advancing direct observation”, *International Journal of Environmental Research and Public Health* (ECCV), 2017.

Bo Liu, Nuno Vasconcelos, “Bayesian Model Adaptation for Crowd Counts”, In *Proceedings of IEEE International Conference on Computer Vision* (ICCV), 2015.

ABSTRACT OF THE DISSERTATION

To Overcome Limitations of Computer Vision Datasets

by

Bo Liu

Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics, & Control)

University of California San Diego, 2021

Professor Nuno Vasconcelos, Chair

Large-scale datasets play a key role in the success of modern computer vision models. However, there are limitations of current datasets due to the data collection strategy. When an object has a canonical pose, pose bias is usually common. And models can only perform good on specific poses. We mitigate this by introducing two data augmentation methods, one in feature space with feature transfer, the other in image space with novel view synthesis. Both of them can improve the pose diversity of the current dataset. Imbalance class distribution is the nature of real world. To deal with this problem, we first discuss the few-shot open-set problem. By introducing meta-learning to open-set detection, PEELER rejects unseen samples with no cost of seen class accuracy. With a larger class label space, long-tailed recognition problem is then discussed. GistNet improve the long-tailed performance by geometry transfer, while Breadcrumb mitigates the over-fitting of few-shot samples by feature back-tracking. Last but not least, the task of semi-supervised long-tailed recognition is introduced. Alternate learning is presented to combine semi-supervised learning with long-tailed recognition. All of above mentioned methods are proved to be useful when facing certain limitations of computer vision datasets.

Chapter 1

Introduction

1.1 Computer Vision Datasets

Computer vision is a scientific field that deals how the computer can acquire and process the high-level understanding from digital images or videos. This understanding can be regarded as the extraction and perception of the real world to provide symbolic or numerical information, *e.g.* in the forms of decisions [53, 81, 47]. The goal is to understand and reproduce the human visual system automatically. There are several sub-domains of computer vision, including object recognition, detection, action recognition, image retrieval, pose estimation, tracking, segmentation, etc.

Most computer vision tasks are data-driven. Machine learning models rely on available datasets to tune parameters. Pioneers include MNIST [62], NORB [63], COIL [84], etc. Recently, deep learning has achieved great success and push the whole field moving forward a lot. This is main due to the availability of large-scale datasets, such as ImageNet [19].

Early computer datasets are usually collected and annotated "in the lab". This means all the images can be manually controlled and selected by experts to present the diverse natural world with limited source or meet any specific requirements. Recently, due to the development of the Internet and crowd-based annotation platforms, collecting large-scale datasets without specific supervision by experts, *i.e.* "in the wild", is possible. In result, data-hungry methods, such as deep learning is possible to apply and achieve significant success.

1.2 Limitations of Computer Vision Datasets

Modern datasets usually collect source images from the Internet. This makes it possible to collect plenty of data in a limited budget. However, it is not perfect. It introduces the *bias* of Internet images to computer vision datasets, and furthermore, harms the generalizability of computer vision models trained on those datasets.

One of the main bias is the pose coverage. Images on the Internet is uploaded by people, and people tend to upload photos that are visually pleasant. However, some objects are specifically more common in certain poses. For example, when one searches “dog” on the Internet, it is hard to find images from the back. Figure 1.1 shows the pose distribution of a ImageNet class “monitor”. For the object with canonical “frontal” pose, most images are from that pose, and a model trained with that data can only have good performance on that pose. However, in real world scenario, the model must have to deal with any of poses from an object. For example, a robot can encounter or go around an object from all different direction. It is unacceptable if it can only acquire the high-level information in certain views. It is also not the way how humans understand the world. Human beings can recognize any object in any views, and in most of the time, they can even hallucinate the missing part in the current view in their brains.

Another problem of computer vision datasets is from the imbalance nature of real world class distribution. The popularity of classes are different in the real world. And when a dataset is collected by randomly selecting images, the class distribution of the dataset should follow the one in real world, which is the natural distribution. In fact, the distribution is usually long-tailed. A small part of classes contributes large amount of data, while most of the classes only have few labelled examples. Unfortunately, regular computer vision models do not work well on this situation. They tend to over-fit to more populated classes and under-fit to less populated classes. People mitigate this problem by manually balancing the dataset before presenting to models. However, due to the large imbalance ratio, this lead to much more data collection effort and a waste of data in most populated classes.

1.3 Contributions of the Thesis

In the thesis, we present several works to deal with the aforementioned problems of computer vision datasets. Two of them deal with the problem by augmenting the data in feature space and image space respectively. One of them specifically studies the classes with insufficient

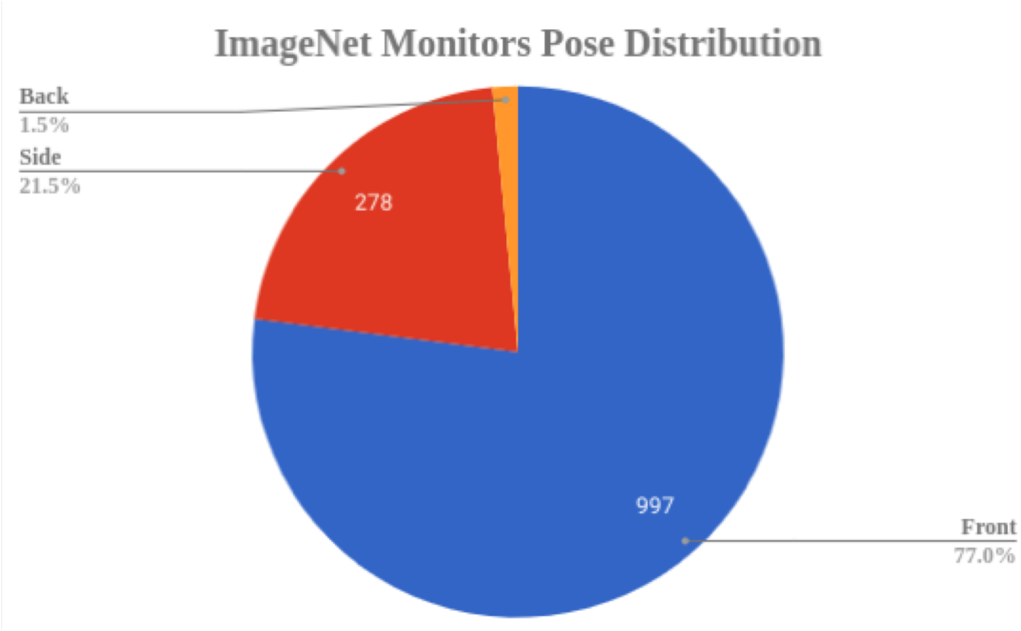


Figure 1.1: Pose Distribution of ImageNet Monitors.

labels in an open-set setting. Two of them investigates the long-tailed problem by adjusting the model and training strategy. Last but not least, we consider a problem that combines the data augmentation and model training.

1.3.1 Feature Space Augmentation

The problem of data augmentation in feature space is considered. A new architecture, denoted the FeATure TransfEr Network (FATTEN), is proposed for the modeling of feature trajectories induced by variations of object pose. This architecture exploits a parametrization of the pose manifold in terms of pose and appearance. This leads to a deep encoder/decoder network architecture, where the encoder factors into an appearance and a pose predictor. Unlike previous attempts at trajectory transfer, FATTEN can be efficiently trained end-to-end, with no need to train separate feature transfer functions. This is realized by supplying the decoder with information about a target pose and the use of a multi-task loss that penalizes category- and pose-mismatches. In result, FATTEN discourages discontinuous or non-smooth trajectories that fail to capture the structure of the pose manifold, and generalizes well on object recognition tasks involving large pose variation. For few-shot recognition, meta-learning is used to further stabilize the model when applied on unseen classes. Experimental results on the artificial ModelNet database show that it can successfully learn to map source features to target features of a desired pose, while preserving class identity. Most notably, by using feature space transfer for data augmentation (*w.r.t.* pose and depth) on SUN-RGBD objects, we demonstrate considerable performance improvements on one/few-shot object recognition in a transfer learning setup, compared to current state-of-the-art methods. The method is also applied on single-view reconstruction. By augmenting shape codes in terms of poses, it boosts the performance of the auto-encoder based reconstruction method.

1.3.2 Image Space Augmentation by View Synthesis

We propose a method to learn, even using a dataset where objects appear only in sparsely sampled views (*e.g.* Pix3D), the ability to synthesize a pose trajectory for an arbitrary reference image. This is achieved with a cross-modal pose trajectory transfer mechanism. First, a domain transfer function is trained to predict, from an RGB image of the object, its 2D depth map. Then, a set of image views is generated by learning to simulate object rotation in the depth space. Finally, the generated poses are mapped from this latent space into a set of corresponding RGB images using a learned identity preserving transform. This results in a dense pose trajectory of the object in image space. For each object type (*e.g.*, a specific Ikea chair model), a 3D CAD model is used to render a full pose trajectory of 2D depth maps. In the absence of dense pose sampling in image space, these latent space trajectories provide cross-modal guidance for learning. The learned pose trajectories can be *transferred* to unseen examples, effectively synthesizing all object views in image space. Our method is evaluated on the Pix3D and ShapeNet datasets, in the setting of novel view synthesis under sparse pose supervision, demonstrating substantial improvements over recent art.

1.3.3 Few-shot Open-set Recognition

The problem of open-set recognition is considered. While previous approaches only consider this problem in the context of large-scale classifier training, we seek a unified solution for this and the low-shot classification setting. It is argued that the classic softmax classifier is a poor solution for open-set recognition, since it tends to over-fit on the training classes. Randomization is then proposed as a solution to this problem. This suggests the use of meta-learning techniques, commonly used for few-shot classification, for the solution of open-set recognition. A new *oPen sEt mEta LEaRning* (PEELER) algorithm is then introduced. This combines the random selection of a set of novel classes per episode, a loss that maximizes the posterior entropy for examples of those classes, and a new metric learning formulation based on the Mahalanobis distance. Experimental results show that PEELER achieves state of the art open set recognition performance for both few-shot and large-scale recognition. On CIFAR and miniImageNet, it achieves substantial gains in seen/unseen class detection AUROC for a given seen-class classification accuracy.

1.3.4 Long-tailed Recognition with Geometry Transfer

The problem of long-tailed recognition, where the number of examples per class is highly unbalanced, is considered. It is hypothesized that the well known tendency of standard classifier training to over-fit to popular classes can be exploited for effective transfer learning. Rather than eliminating this over-fitting, *e.g.* by adopting popular class-balanced sampling methods, the learning algorithm should instead leverage this over-fitting to *transfer* geometric information from popular to low-shot classes. A new classifier architecture, GistNet, is proposed to support this goal, using constellations of classifier parameters to encode the class geometry. A new learning algorithm is then proposed for *GeometrIc Structure Transfer* (GIST), with resort to a combination of loss functions that combine class-balanced and random sampling to guarantee that, while over-fitting to the popular classes is restricted to geometric parameters, it is leveraged to transfer class geometry from popular to few-shot classes. This enables better generalization for few-shot classes without the need for the manual specification of class weights, or even the explicit grouping of classes into different types. Experiments on two popular long-tailed recognition datasets show that GistNet outperforms existing solutions to this problem.

1.3.5 Long-tailed Recognition with Feature Back-tracking

The problem of long-tailed recognition, where the number of examples per class is highly unbalanced, is considered. While training with class-balanced sampling has been shown effective for this problem, it is known to over-fit to few-shot classes. It is hypothesized that this is due to the repeated sampling of examples and can be addressed by feature space augmentation. A new feature augmentation strategy, EMANATE, based on back-tracking of features across epochs during training, is proposed. It is shown that, unlike class-balanced sampling, this is an adversarial augmentation strategy. A new sampling procedure, Breadcrumb, is then introduced to implement adversarial class-balanced sampling without extra computation. Experiments on three popular long-tailed recognition datasets show that Breadcrumb training produces classifiers that outperform existing solutions to the problem.

1.3.6 Semi-supervised Long-tailed Recognition

Main challenges in long-tailed recognition come from the imbalanced data distribution and sample scarcity in its tail classes. While techniques have been proposed to achieve a more balanced training loss and to improve tail classes data variations with synthesized samples, we resort to leverage readily available unlabeled data to boost recognition accuracy. The idea leads to a new recognition setting, namely semi-supervised long-tailed recognition. We argue this setting better resembles the real-world data collection and annotation process and hence can help close the gap to real-world scenarios. To address the semi-supervised long-tailed recognition problem, we present an alternate sampling framework combining the intuitions from successful methods in these two research areas. The classifier and feature embedding are learned separately and updated iteratively. The class-balanced sampling strategy has been implemented to train the classifier in a way not affected by the pseudo labels' quality on the unlabeled data. A consistency loss has been introduced to limit the impact from unlabeled data while leveraging them to update the feature embedding. We demonstrate significant accuracy improvements over other competitive methods on three datasets.

1.4 Organization of the Thesis

The thesis is organized as follows. In Chapter 2, we discuss feature transfer network, denoted as FATTEN, which augments the training data with pose completion in feature space. Chapter 3 describes a novel view synthesis method that generates real world images with unseen poses. Chapter 4 introduces and combines the open-set problem with few-shot learning. A meta-learner PEELER is implemented to deal with the few-shot open-set recognition problem. In Chapter 5, we present the long-tailed nature in real world datasets, and investigate the problem by recovering the natural class geometry with GistNet. Chapter 6 further discusses the long-tailed problem by introducing back-tracking to avoid feature space over-fitting during class-balanced sampling. Chapter 7 extends the long-tailed problem with semi-supervised learning. Alternate learning is implemented to better use both supervised and unsupervised data. Finally, we conclude the thesis in Chapter 8.

Chapter 2

Feature Space Transfer as Data Augmentation for Few-shot Classification and Single-View Reconstruction

2.1 Introduction

Convolutional neural networks (CNNs) trained on large datasets, such as ImageNet [19], have shown significant gains for computer vision problems like object recognition over the last few years. These models not only achieve human level performance in recognition challenges, but are also easily transferable to other data domains or tasks, by fine tuning. Many recent works have shown that ImageNet trained CNNs, like AlexNet [57], VGG [108], GoogLeNet [119], or ResNet [38] can be used as feature extractors for the solution of many other problems. Nevertheless, there are still challenges to CNN-based recognition. One limitation is that existing CNNs still have limited ability to handle pose variability. This is, in part, due to limitations of existing datasets, which are usually collected on the web and are biased towards a certain type of images. For example, objects that have a well defined "frontal view," such as "couch" or "clock," are rarely available from viewing angles that differ significantly from frontal.

This is problematic for applications like robotics, where a robot might have to navigate around or manipulate such objects. When implemented in real time, current CNNs tend to produce object labels that are unstable with respect to viewing angle. The resulting object recognition can vary from nearly perfect under some views to much weaker for neighboring, and very similar, views. One potential solution to the problem is to rely on larger datasets with a much more dense sampling of the viewing sphere. This, however, is not trivial, for a number of reasons. *First*, for many classes, such images are not easy to find on the web in large enough quantities. *Second*, because existing recognition methods are weakest at recognizing "off-view" images, the process cannot be easily automated. *Third*, the alternative of collecting these images in the lab is quite daunting. While this has been done in the past, *e.g.*, the COIL [84], NORB [63], or Yale face dataset, these datasets are too small by modern standards. The set-ups used to collect them, by either using a robotic table and several cameras, or building a camera dome, can also not be easily replicated and do not lend themselves to distributed dataset creation efforts, such as crowd

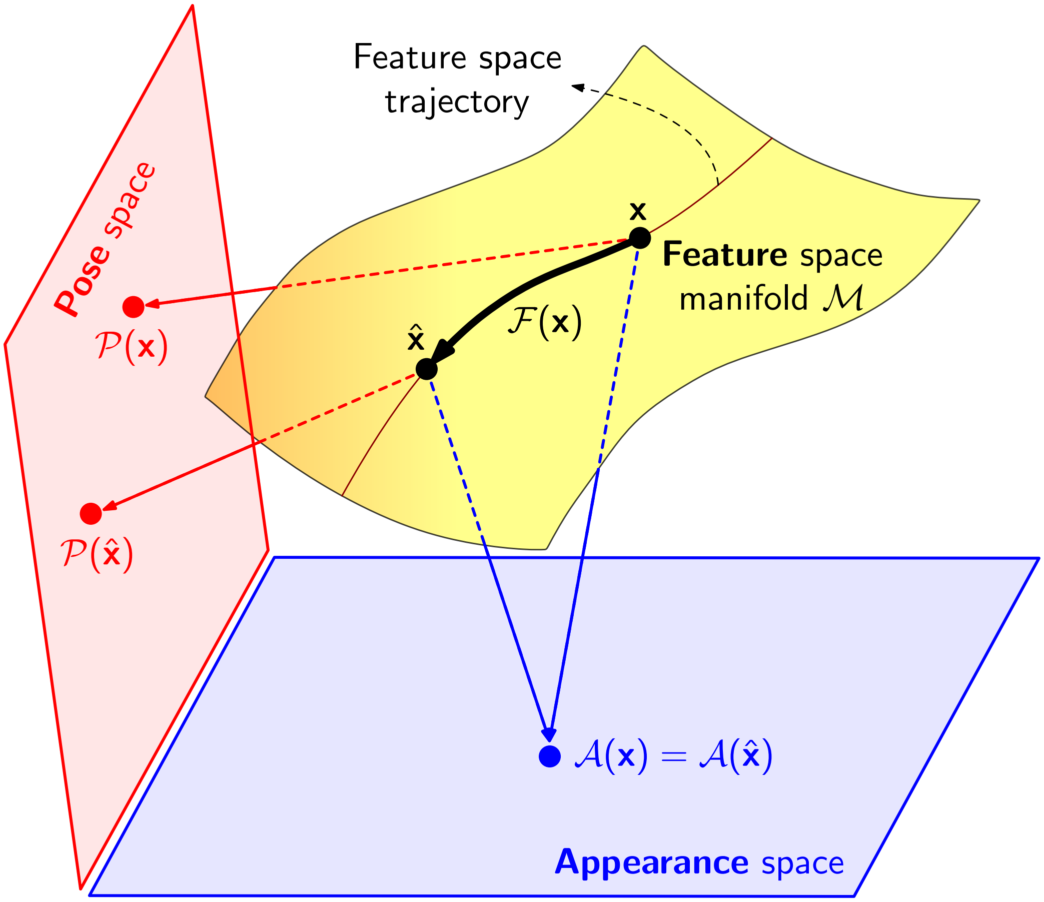

Feature space
trajectory
Pose space
$\mathbf{x}$
Feature space
manifold $\mathcal{M}$
$\mathcal{P}(\mathbf{x})$
$\hat{\mathbf{x}}$
$\mathcal{F}(\mathbf{x})$
$\mathcal{P}(\hat{\mathbf{x}})$
$\mathcal{A}(\mathbf{x}) = \mathcal{A}(\hat{\mathbf{x}})$
Appearance space

Figure 2.1: Schematic illustration of *feature space transfer* for variations in *pose*. The input feature $\mathbf{x}$ and transferred feature $\hat{\mathbf{x}}$ are projected to the same point in *appearance space*, but have different mapping points in *pose space*.

sourcing. Finally, even if feasible to assemble, such datasets would be massive and difficult to process. For example, the NORB recommendation of collecting 9 elevations, 36 azimuths, and 6 lighting conditions per object, results in 1944 images per object. Applying this standard to ImageNet would result in a dataset of close to 2 billion images!

Some of these problems can be addressed with computer generated images. This is indeed an established practice for problems that require multiple object views, such as shape recognition, where synthetic image datasets [90, 114] are routinely used. However, the application of networks trained on synthetic data to real images raises a problem of *domain adaptation*. Despite a vast literature on the topic [110, 61, 101, 120, 129, 105, 96], adaptation methods are usually not tailored for the transfer of object poses. In particular, they do not account the fact that, as illustrated in Fig. 2.1, objects subject to pose variation span low-dimensional manifolds of image space, or corresponding spaces of CNN features. This is because objects can be decomposed into appearance and pose components. While the appearance component is fixed, the pose component varies with viewing angle. Since the latter has few degrees of freedom, the *pose trajectories* spanned by the object for different angles define low-dimensional surfaces in image or feature space. The mapping between locations along a pose trajectory is denoted *pose transfer.* This has only been considered by a few works [20, 69], who have proposed models with explicit pose inputs to transfer objects along the pose manifold.

Besides explicit pose transfer, it is also possible to resort to few-shot learning methods. One popular solution is to rely on meta-learning [123]. This is a generic term for methods that try to train a "learner". Many recent works show that meta-learning is useful for the few-shot learning of a parameterized function that maps limited labeled data to a classifier [24, 109]. For pose transfer, this function can be meta-trained from data with many poses. However, all current meta-learning methods treat the training procedure as a black box, and rely on meta-training to learn all unknown structures implicitly. In particular, they don't have an explicit model for pose transfer. Moreover, most methods are only trained on original data, without augmentation. One possibility is to add an hallucination module [132], by using a generator to leverage prior visual knowledge and hallucinate additional training data for better classification. However, the generator of [132] fails to take the advantage of pose transfer to generate more effective data.

The benefits of data augmentation along pose trajectories are not limited to classification. For example, single-view reconstruction is another prime application target. This follows from the ill-defined nature of 3D shape recovery from a single object image, since part of the shape is never seen. Traditional augmentations are not effective for this problem and can even be misleading if combining information from different objects. We find that pose transfer is specifically suitable for this topic, because augmentations across pose trajectory leverage information from multiple views. This provides extra 3D structure while maintaining the semantic object information.

Contribution

In this work, we propose a universal framework, termed *FeATure TransfEr Network* (FATTEN), that addresses these problems. Essentially, FATTEN is an encoder-decoder architecture, inspired by Fig. 2.1. We parameterized pose trajectory transfer in terms of an *appearance map*, which captures properties invariant on pose, such as object texture and structure, and a *pose map*, which is pose dependent. Th *encoder* maps an input feature $\mathbf{x}$ into a pair of appearance $\mathcal{A}(\mathbf{x})$ and pose $\mathcal{P}(\mathbf{x})$ parameters. The *decoder* then takes the appearance parameter together with a target pose parameter $\mathbf{t} = \mathcal{P}(\hat{\mathbf{x}})$ and generate a feature vector $\hat{\mathbf{x}}$ with the new pose.

The FATTEN model is applied to few-shot recognition and single-view reconstruction. For recognition, a classifier is learned by meta-learning, using a parametric learner that takes the pose-generated data as labeled data to produce a classifier. To avoid mismatching, both pose transfer module and classifier are trained end-to-end, using a multi-task loss that accounts for both classification and feature transfer errors. For single-view reconstruction, FATTEN is used to augment shape codes. A 3D auto-encoder is used for 3D reconstruction, producing a shape code that is used as a bridge between a 2D image and the corresponding 3D shape. FATTEN is trained to augment shape codes along pose trajectories. The combination of these augmentations is shown to improve 3D reconstruction quality.

The performance of FATTEN is investigated in two stages. In the first stage, we examine the effectiveness of pose transfer. We utilize the model in a *multi-view retrieval* task, where generated features are used to retrieve features by category and pose. These experiments are carried out on a synthetic 3D dataset ModelNet [137]. Results show that our pose transfer module hallucinates features with good quality along both object category and pose dimensions, for applications involving computer graphics imagery. This could be of use for a now large body of 3D shape classification works [137, 113, 113, 91], where such datasets are predominant.

Second, FATTEN is embedded with other models to work on few-shot recognition and single-view reconstruction. For few-shot recognition, we combine the pose transfer module and a classifier and train the whole model end-to-end. We show that the proposed architecture outperforms both pure meta-learning methods and meta-learning methods with generic feature hallucinator. For single-view reconstruction, we combine FATTEN with a popular 3D reconstruction model, the BSP-Net, whose performance is also shown to benefit from the addition of FATTEN.

Organization

In Section 2.2, we review related work. Section 2.3 introduces the proposed FATTEN architecture. Section 2.4 and 2.5 discuss the application of FATTEN to few-shot recognition and single-view reconstruction. Section 2.6 presents experimental results on ModelNet, SUN-RGBD, and ShapeNet. Finally, Section 2.7 concludes the paper with a discussion of the main points and an outlook on open issues.

2.2 Related Work

Objects describe smooth trajectories in image space, where they span a 3D manifold, parameterized by the viewing angle. Hence, many of the manifold modeling methods proposed in the literature [102, 3, 127] could, in principle, be used to develop trajectory transfer algorithms. However, many of these methods are transductive, *i.e.*, they do not produce a function that can make predictions for images outside of the training set, and do not leverage recent advances in deep learning. While deep learning could be used to explicitly model pose manifolds, it is difficult to rely on CNNs pre-trained on ImageNet for this purpose. This is because these networks attempt to collapse the manifold into a space where class discrimination is linear. On the other hand, the feature trajectories in response to pose variability are readily available. These trajectories are also much easier to model. For example, if the CNN is successful in mapping the pose manifold of a given object into a single point, *i.e.*, exhibits total pose invariance for that object, the problem is already solved and trajectory leaning is trivial for that object.

On the other hand, trajectory transfer is popular for problems involving multi-view recognition. Due to the increased cost and difficulty of multi-view image collecting, such problems usually consider some level of learning from synthetic images. In fact, there is an established practice in shape recognition, where synthetic 3D datasets [90, 114] are widely used. A rich literature in shape recognition from synthetic datasets has produced many 3D representations [54, 137, 113, 113, 91, 48]. Nevertheless, it has also been shown that the 3D recognition problem can be efficiently solved as multi-view 2D recognition by using simple multi-view extensions of current CNNs [113]. However, it is not evident how these conclusions can generalize to real world datasets.

An important component of the proposed model is a pose transfer module that augments a dataset with feature responses of unseen object poses. In this sense, the problem is related to novel view synthesis [10, 89, 116, 122, 151, 135, 125]. [151] uses appearance flows to synthesize novel views of both objects and scenes. [116] combines information from multiple views. [135] and [125] mainly focus on scene images. All these works aim to generate realistic images, but do not discuss potential benefits for image classification. This mostly due to the difficulty of generating images, leading to models that only work well on limited categories. This is unlike the proposed model, which simplifies the task by only generating features. This generalizes to larger sets of categories and improves feature robustness.

With the introduction of large scale 3D datasets, such as ShapeNet [8], 3D modeling has been widely studied. One of the important tasks is single-view reconstruction, which recovers a 3D shape model from a single view of an object. Early works, such as 3D-R2N2 [15], generalize 2D convolution to 3D and model 3D shapes with voxels. This usually has low resolution, due to limits in computation and memory. Later on, methods such as [34], directly generate meshes or surfaces of the shape. This, however, could lead to over complicated surfaces, and sometimes produces open surfaces. Recently, there has been a trend of using implicit models [80, 13, 12], which model the shape with a 3D point classifier. In this work, we adopt one of the most recent and successful implicit models, the BSP-Net [12].

Instead, trajectory transfer is more closely related to the topic of transfer learning, where, there extensive work has been devoted to problems such as zero-shot [110, 61, 101] or few-shot learning [120, 129, 105]. Meta-learning has recently shown its effectiveness for few-shot recognition. Some methods, such as MAML and its variants [24, 25], or LEO [103], are gradient based. These methods take advantage of second derivatives to optimize the model from few-shot samples. Another group of methods, including the matching network [129], prototypical network [109], relation network [118], and category traversal [65], aims to learn robust metrics. Some few-shot methods have also proposed to augment training data by combining GANs with meta-learning [132], synthesizing features across object views [69] or using other forms of data hallucination [35]. Learning without forgetting [29] aims to transfer the existing classifier to novel classes without loss of performance on the base ones. [49] extends the few-shot problem from recognition to detection by feature re-weighting. Self-supervised learning or representation learning is a technique aiming for knowledge transfer. It has recently been studied for general classification tasks [30, 37, 9]. However, these methods tend to be of general purpose. None of them exploits specific properties of the pose manifold, such as the parametrization of Fig. 2.1. The introduction of networks that enforce such parametrizations is a form of regularization that improves on the transfer performance of generic procedures.

2.3 The Feature Transfer Meta-Learner Architecture

In this section, we describe the proposed architecture for *feature space transfer*.

2.3.1 Feature Transfer Motivation

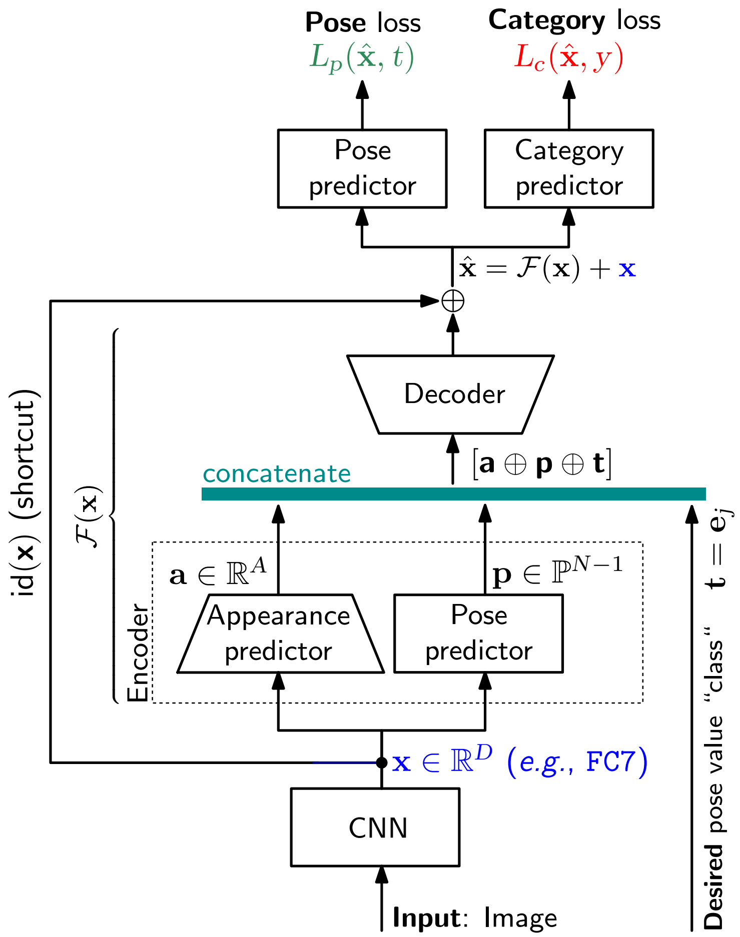
In this work, we assume the availability of a training set with pose annotations, *i.e.*, $\mathcal{S}_{train} = \{(\mathbf{x}_n, \mathbf{p}_n, y_n)\}_n$, where $\mathbf{x}_n \in \mathbb{R}^D$ is the feature vector (*e.g.*, a CNN activation at some layer) extracted from an image, $\mathbf{p}_n$ is the corresponding pose value and y_n a category label. The pose value could be a scalar p_n, *e.g.*, the azimuth angle on the viewing sphere, but is more generally a vector, *e.g.*, also encoding an elevation angle or even the distance to the object (object depth). The feature transfer problem is to learn the transfer function $\mathcal{F}(\mathbf{x}_n, \mathbf{p})$ that maps the source feature vector $\mathbf{x}_n$ to a target feature vector $\hat{\mathbf{x}}_n$ corresponding to a new pose $\mathbf{p}$.

2.3.2 The Feature Transfer Network Architecture

The FeATure TransfEr Network (FATTEN) architecture is illustrated by Fig. 2.1, which depicts a pose manifold spanned by an object under pose variation, and parameterized by two variables. One of them is an *appearance descriptor* $\mathbf{a} \in \mathbb{R}^A$ that represents pose invariant properties such as color or texture. This means that it has the same value for all points on the manifold. It can be thought of as an object code that distinguishes the manifold spanned by one object from those spanned by others. It is worth noting that appearance is not the same as category. Some objects of different categories may share more similar appearance than those of the same category. The other variable is a *pose descriptor* $\mathbf{p} \in \mathbb{R}^N$ that captures the point $\mathbf{x}$ that corresponds to a particular pose $\mathbf{p}$. In form, a feature point $\mathbf{x}$ on the manifold could be considered as the implementation of a mapping

$$\phi : \mathbb{R}^A \times \mathbb{R}^N \rightarrow \mathcal{M}, \quad \phi(\mathbf{a}, \mathbf{p}) \mapsto \mathbf{x} \ . \tag{2.1}$$

The FATTEN architecture, as a encoder-decoder architecture shown in Fig. 2.2, models the relationship between the feature vectors extracted from object images and its associated appearance and pose parameters. The encoder is designed to invert the mapping of (2.1), namely



Pose loss
$L_p(\hat{\mathbf{x}}, t)$
Category loss
$L_c(\hat{\mathbf{x}}, y)$
Pose predictor
Category predictor
$\hat{\mathbf{x}} = \mathcal{F}(\mathbf{x}) + \mathbf{x}$
Decoder
concatenate
$[\mathbf{a} \oplus \mathbf{p} \oplus \mathbf{t}]$
id(**x**) (shortcut)
$\mathcal{F}(\mathbf{x})$
Encoder
$\mathbf{a} \in \mathbb{R}^A$
$\mathbf{p} \in \mathbb{P}^{N-1}$
Appearance predictor
Pose predictor
$\mathbf{x} \in \mathbb{R}^D$ (*e.g.*, FC7)
CNN
Input: Image
Desired pose value "class" $\mathbf{t} = \mathbf{e}_j$

Figure 2.2: The FATTEN architecture. Here, id denotes the identity shortcut connection, D the dimensionality of the input feature space, A the dimensionality of the appearance space and $\mathbb{P}^{N-1}$ the $N-1$ probability simplex. Both pose predictors are pre-trained and share parameters.

implement ϕ^{-1}. Given a feature vector $\mathbf{x}$, it produces an estimate of the appearance $\mathbf{a}$ and pose $\mathbf{p}$ parameters. A *target* pose parameter $\mathbf{t}$ specifies the corresponding pose of the desired feature vector $\hat{\mathbf{x}}$, which will be then generated by a decoder that processes the concatenation of $\mathbf{a}$, $\mathbf{p}$ and $\mathbf{t}$. While, in principle, it would be sufficient to derive $\hat{\mathbf{x}}$ from $\phi(\mathbf{a},\mathbf{t})$, *i.e.*, to use the inverse of the encoder as a decoder, we have obtained the best results with the following modifications.

First, to prevent the encoder/decoder pair from learning a mapping that simply "matches" feature pairs, FATTEN implements the residual learning paradigm of [38]. In particular, the encoder-decoder is only used to learn the residual

$$\mathcal{F}(\mathbf{x}) = \hat{\mathbf{x}} - \mathbf{x} \tag{2.2}$$

between the target and source feature vectors. *Second*, two modules that explicitly predict the appearance **a** and pose **p** parameters are used instead of a single encoder. In our experience this decomposition facilitates learning, since the pose predictor can be learned with full supervision. *Third*, the source **p** and target **t** pose parameters are encoded as one-hot vectors instead of continuous scalars. This makes the dimensionality of the pose parameters closer to that of the appearance parameter, and triggers a more balanced training procedure. We have noted that, otherwise, the learning algorithm can have a tendency to ignore the pose parameters and produce a limited diversity of target feature vectors. *Finally*, the decoder can leverage the source pose **p**, in addition to **a** and **t** in (2.1). This again guarantees that the *intermediate* representation is higher dimensional and accelerates the learning of the decoder. We next discuss the details of the various network modules.

2.3.3 FATTEN Details

Encoder

The encoder consists of a pose and an appearance predictor. The pose predictor implements the mapping $\mathbf{p} = \mathcal{P}(\mathbf{x})$ from input feature vectors $\mathbf{x}$ to pose descriptors $\mathbf{p}$. The poses, as azimuth angle, are first internally regulated into a code vector $\mathbf{c} \in \mathbb{R}^N$ of dimensionality comparable to that of the appearance vector **a**. In the current implementation of FATTEN this is achieved in three steps. First, the full angle space is quantized into N cells with centroids $\mathbf{m}_i$. Second, Each pose is then assigned to the cell of the nearest representative $\mathbf{m}^*$ and converted by a N-dimensional one-hot vector that identifies $\mathbf{m}^*$. The pose mapping $\mathcal{P}$ is finally implemented with a N-way classifier that maps **x** into a vector of posterior probabilities

$$\mathbf{p} = [p(\mathbf{m}_1|\mathbf{x}), \ldots, p(\mathbf{m}_N|\mathbf{x})] \tag{2.3}$$

on the $N-1$ probability simplex $\mathbb{P}^{N-1}$. The classifier consists of a two-layer neural network, composed of a fully-connected layer, batch normalization, and a ReLU, followed by a softmax layer.

The appearance predictor implements the mapping $\mathbf{a} = \mathcal{A}(\mathbf{x})$ from input feature vectors **x** to appearance descriptors **a**. This is implemented with a two-layer network, where each layer consists of a fully-connected layer, batch normalization, and a ELU layer.

The outputs of the pose and appearance predictors are concatenated with a one-hot encoding of the *target* pose. The encoding is implemented in the same way as the pose code vector of the pose predictor.

Decoder

The decoder maps the vector of concatenated appearance and pose parameters

$$[\mathbf{a} \oplus \mathbf{p} \oplus \mathbf{t}] \tag{2.4}$$

where $\oplus$ denotes vector concatenation, into the residual $\hat{\mathbf{x}} - \mathbf{x}$ of (2.2). It is implemented with a two layer network, where the first layer contains a sequence of fully-connected layer, batch normalization, and ELU, and the second is a fully connected layer. The decoder output $\mathcal{F}(\mathbf{x})$ is then added to the input **x** to produce the target $\hat{\mathbf{x}}$.

Although conceptually similar, our architecture is different from AGA [20] and solves some key limitations of the latter. In particular, the feature synthesis function $f(\mathbf{x}, \mathbf{p}, \mathbf{t})$ of AGA is implemented as a series of encoder-decoder modules $f_{\mathbf{p},\mathbf{t}}(x)$, one for each pair of $(\mathbf{p}, \mathbf{t})$. The number of these functions grows exponentially and AGA needs to learn a different f for each

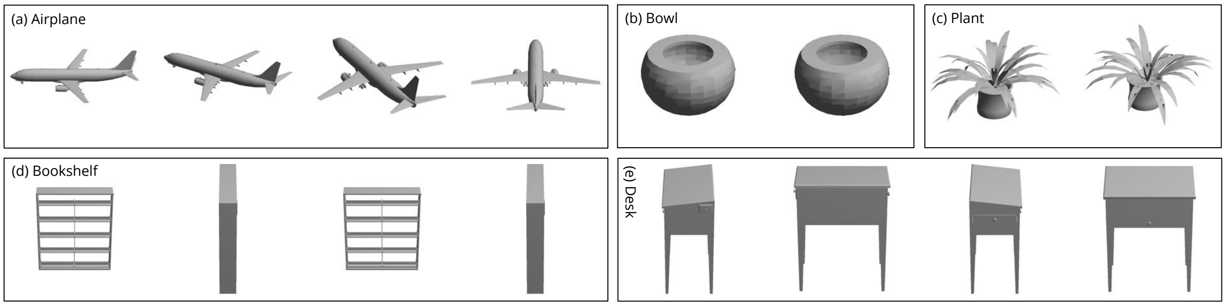


Figure 2.3: Exemplary ModelNet [137] views: (a) different views of one object (airplane); (b)-(c) different views of two *symmetric* objects (bowl, plant); (d)-(e) four views (bookshelf, desk) with 90 degrees difference.

$\mathbf{p} \to \mathbf{t}$ and $\mathbf{t} \to \mathbf{p}$; there is no provision to share information. Since FATTEN uses a *single* network to solve this task, (1) model complexity scales favorably with pose quantization and (2) due to weight sharing, pose translations are informed by each other. Also, AGA uses an L_2 regularizer in feature space, which may not preserve class identity; FATTEN uses a category loss to help address this problem.

2.3.4 FATTEN Training

FATTEN is trained as a data generator, independently of the downstream tasks to which it may be applied. A multi-task loss is designed to ensure *end-to-end* training that accomplishes two goals. The *first* is that the synthesized feature vector $\hat{\mathbf{x}}$ should correspond to the desired pose $\mathbf{t}$. This constraint is enforced by the pose loss, which is the cross-entropy loss commonly used for classification

$$L_p(\hat{\mathbf{x}}, \mathbf{t}) = -\mathbf{t}^T \log \rho(\mathcal{P}(\hat{\mathbf{x}})), \tag{2.5}$$

where ρ is the softmax function, i.e. $\rho_j(v) = \frac{e^{v_j}}{\sum_k e^{v_k}}$. Note that, as shown in Fig. 2.2, this requires feeding the target feature vector $\hat{\mathbf{x}}$ into a pose predictor $\mathcal{P}$. It is worth noting that, while this is only needed for training, the loss of (2.5) can also be computed during inference, since the target pose $\mathbf{t}$ is known. This can be used as a diagnostic measure for the performance of FATTEN.

The *second* goal is that the generated feature vector $\hat{\mathbf{x}}$ maintains all the semantic information of the source vector $\mathbf{x}$. This semantic information can vary from task to task. For classification, it is the category label y. Ideally, the synthesized feature vectors should achieve the same recognition results as the network used to extract the feature vectors, denoted as CNN in Fig. 2.2, in the original problem. To guarantee this, FATTEN uses the linear classifier obtained by training the CNN backbone as a category predictor. This predictor is fixed during the training of the remaining FATTEN blocks to avoid over-fitting. Its accuracy is measured by the cross-entropy loss

$$L_c(\hat{\mathbf{x}}, y) = -\log \rho_y(\hat{\mathbf{x}}), \tag{2.6}$$

where $\rho(v)$ is the softmax output of the category predictor. The *multi-task loss* is finally defined as

$$L_{\text{FATTEN}}(\hat{\mathbf{x}}, \mathbf{t}, y) = L_p(\hat{\mathbf{x}}, \mathbf{t}) + L_c(\hat{\mathbf{x}}, y). \tag{2.7}$$

This leads to three stages of training. In the first, the CNN backbone and class predictor are trained for category prediction, without pose information. This step can be skipped if a pre-trained CNN is available. In the second stage, the pose predictor $\mathcal{P}(\mathbf{x})$ is trained with pose supervision. This is then *embedded* into the encoder-decoder structure and pose loss structure. In the final training stage, both pose and category predictors are fixed, and the appearance predictor and decoder are trained end-to-end without further appearance supervision. We found this training strategy to be beneficial in two ways. First, embedding the pre-trained pose predictor reduces the number of degrees of freedom in the network, minimizing the ambiguity inherent to the fact that a given feature vector could be consistent with multiple pairs of pose and appearance parameters. Unlike pose parameters, FATTEN allows a certain flexibility on the appearance prediction for further robustness. For example, while all feature vectors $\mathbf{x}$ extracted from views of the same object should map into the same appearance parameter value $\mathbf{a}$, we do not enforce such constraint. This endows the network with invariance to small variations of the appearance descriptor, due to occlusions, variations in lighting, etc. Second, by pre-training the pose and category predictor, *only* weights of the encoder/decoder need to be learned end to end. The weights of the sub-networks used by the loss function(s) are fixed. This minimizes the chance that FATTEN will over-fit to specific poses or object categories.

2.4 Few-Shot Recognition

A one/few-shot classifier is a classifier trained with one or few examples per class. In this section we investigate the use of FATTEN as a data augmentation technique for this problem. We consider different classification strategies, from the traditional support vector machine (SVM) to modern meta-learners.

2.4.1 SVM

A classical solution to the few-shot problem is to rely on the good generalization ability of the SVM [16], which is typically used as a binary classifier. Given a set of training examples $\{\mathbf{x}_i, y_i\}$, the SVM training algorithm minimizes the hinge loss

$$L(\mathbf{x}_i, y_i) = \max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i - b)), \tag{2.8}$$

where $\mathbf{w}$ and b are the classifier parameters to be learned. This can be used to induce a soft margin by introduction of a regularization term, leading to the loss

$$L_{SVM} = \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i - b)) + \lambda||\mathbf{w}||^2. \tag{2.9}$$

A multi-class SVM is then built by learning binary classifiers that oppose one class to all others [43].

One way to implement few shot classification is to rely on a pre-trained CNN backbone, which is used a feature extractor. The softmax layer used for category prediction is then replaced by an SVM, which is trained with the features produced by the CNN for the few examples available from the target task. In our experiments, FATTEN is used as a feature augmentation technique for this second stage, where feature vectors of unavailable poses are synthesized to produce more data for SVM training. This allows us to increase the number of training examples, enabling more robust SVM training. While the simple use of an SVM is not a state of the art method for few-shot classification, the fact that it decouples feature synthesis from classification enables an effective evaluation of the benefits of data augmentation.

2.4.2 Meta-Learning Models

Recently, a number of model generalization approaches have been proposed or aggregated under the meta-learning term. These methods aim to produce a generalized classification method that can learn a model independently of the data domain, and adapt to new domains easily with limited or even no data. This property makes meta-learning models particularly suited for few-shot learning problems, where a training set S_{tr} of class space Y_{tr} and a testing set S_{te} of class space Y_{te} are given, with disjoint Y_{tr} and Y_{te}. While many samples are provided for S_{tr}, only a small number of samples are available in S_{te}. The model is evaluated by its classification accuracy in class space Y_{te}. More specifically, a *query set* Q with the same class space Y_{te} as S_{te}, which is also known is *support set*, is given. The model is evaluated on Q, with the help of samples and labels supported by S_{te}. When the support set S_{te} has N classes and K samples per class, the problem is called *N-way K-shot few-shot classification*.

Unlike traditional mini-batch training, meta-training methods employ episode training on S_{tr}. For each episode, a task is created by sampling a meta-training set M_{tr} from S_{tr} and a meta testing set M_{te} with overlapping classes from S_{te}. The meta-learning model F is then updated using all samples from M_{tr} before classifying samples in M_{te}. Given the updated model $F(\mathbf{x}, M_{tr})$ and example $(\mathbf{x}, y)$ in M_{te}, the model prediction $\hat{y} = F(\mathbf{x}, M_{tr})$ is then chosen to minimize a classification loss $L(\hat{y}, y)$.

The training of each episode mimics the scenario where a model is transferred to a new task. Because a new task is sampled and optimized per episode, the model is less over-fitted to the given training set S_{tr}, and will be more effective when transferred to a new task. During testing, given a new task of support set S_{te} and query set Q, the prediction for a sample $\mathbf{x} \in Q$ is made with $F(\mathbf{x}, S_{te})$.

Different meta-learning methods use different models and even different learning strategies. They can be broadly grouped into two types: gradient-based and metric-based. In gradient-based methods, M_{tr} is used to update the model F by directly back-propagating gradients. However, in metric-based learning, M_{tr} is used to provide class prototypes. Since FATTEN is a data generator, it is more suitable for use with metric-based methods. We consider the following two metric-learning approaches.

Prototypical Networks

Prototypical networks [109] leverages the distance from class center in an M-dimensional feature space. Given an embedding function f_ϕ, The meta-learning model is a classifier

$$F(\mathbf{x}) = \arg\max_{y \in Y_{te}} \frac{e^{-d(\mathbf{x}, \mathbf{c}_y)}}{\sum_{k \in Y_{te}} e^{-d(\mathbf{x}, \mathbf{c}_k)}} \tag{2.10}$$

where

$$d(\mathbf{x}, \mathbf{c}_k) = ||f_\phi(\mathbf{x}) - \mathbf{c}_k||_2 \tag{2.11}$$

is the Euclidean distance between example $\mathbf{x}$ and a prototype $\mathbf{c}_k$ of class k. During meta-training with M_{tr} or testing with S_{te}, F is updated by computing class prototypes as

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{\mathbf{x}_i \in S_k} f_\phi(\mathbf{x}_i), \tag{2.12}$$

where S_k is the set of all points with label $y = k$ in M_{tr} or S_{te}. During meta-training, given meta-training set M_{tr} and a sample ($\mathbf{x}$, y) from meta-testing set M_{te}, the classification loss is defined as

$$\begin{aligned} L_{\text{META}}(\mathbf{x}, y, M_{tr}) &= d(f_\phi(\mathbf{x}), \mathbf{c}_y) \\ &+ \log \sum_k \exp[-d(f_\phi(\mathbf{x}), \mathbf{c}_k)]. \end{aligned} \tag{2.13}$$

Relation Networks

Similar to prototypical networks, relation networks [118] leverage an embedding function f_ϕ to produce a feature map for example $\mathbf{x}$. However, the Euclidean distance is replaced by a relation module g_φ that produces a relation score in $[0, 1]$. The relation score of example $\mathbf{x}$ to class k is defined as

$$r(\mathbf{x}, k) = g_\varphi \left(\sum_{\mathbf{x}_i \in S_k} f_\phi(\mathbf{x}_i), f_\phi(\mathbf{x}) \right), \tag{2.14}$$

where S_k is a set contains all points with label $y = k$ in M_{tr} (during meta-training) or S_{te} (during testing). The loss function is the mean square error (MSE)

$$L_{\text{META}}(\mathbf{x}, y, M_{tr}) = \sum_k (r(\mathbf{x}, k) - \mathbf{1}_{k=y})^2, \tag{2.15}$$

where $\mathbf{1}_{k=y}$ is the indicator function of $k = y$, matched pairs receive a relation score of 1 and mismatched ones a score of 0.

2.4.3 Meta Training

Recent work [132] has shown that training a data augmentation module end-to-end with a meta-learning algorithm can improve the performance of the latter. For pose augmentation, this requires that FATTEN and the meta-learner be jointly trained. To accomplish this, a feature extractor and the FATTEN model are first pre-trained on training set S_{tr}, as discussed in

Section 2.3.4, and then embedded into the meta-training procedure. In a second stage, FATTEN and the classification model are trained together with meta-learning. For this, in each episode, meta-training M_{tr} and meta-testing M_{te} sets are first assembled from S_{tr}. Then, every example in M_{tr} and M_{te} is fed through FATTEN to generate features with a set of target poses $\{\mathbf{t} = 1, \ldots, N\}$. These generated features together with the original features from M_{tr} and M_{te} create augmented meta-training/-testing sets M_{tr}^{aug} and M_{te}^{aug}. The meta-learning model is learned with the meta-learning loss of (2.13) or (2.15), but the meta-learning sets are replaced by M_{tr}^{aug} and M_{te}^{aug}. FATTEN is updated according to the FATTEN loss of (2.7) with the classification term replaced by the meta-learning classification loss. The overall loss of each episode is

$$L(\hat{\mathbf{x}}, y, \mathbf{t}, M_{tr}^{aug}) = L_{\text{META}}(\hat{\mathbf{x}}, y, M_{tr}^{aug}) + L_p(\hat{\mathbf{x}}, \mathbf{t}). \tag{2.16}$$

for any $(\hat{\mathbf{x}}, y) \in M_{te}^{aug}$. Because the feature extractor is fine-tuned, the pose predictor should be changed accordingly. However, we find that fine-tuning the pose predictor does not improve classification performance. This implies that without explicit supervision, the meta-learning fine-tuning does not change the latent pose trajectory. In result, we fix the pose predictor in all settings.

2.5 Single-View Reconstruction

Single-view reconstruction aims to reconstruct the 3D shape of an object from a single image of the object. This is quite difficult because most of the 3D structure of the object is occluded and has to be hallucinated by the reconstruction model. Since FATTEN is designed to augment features along pose trajectories it can simplify this hallucination, by synthesizing a set of views from the single available image.

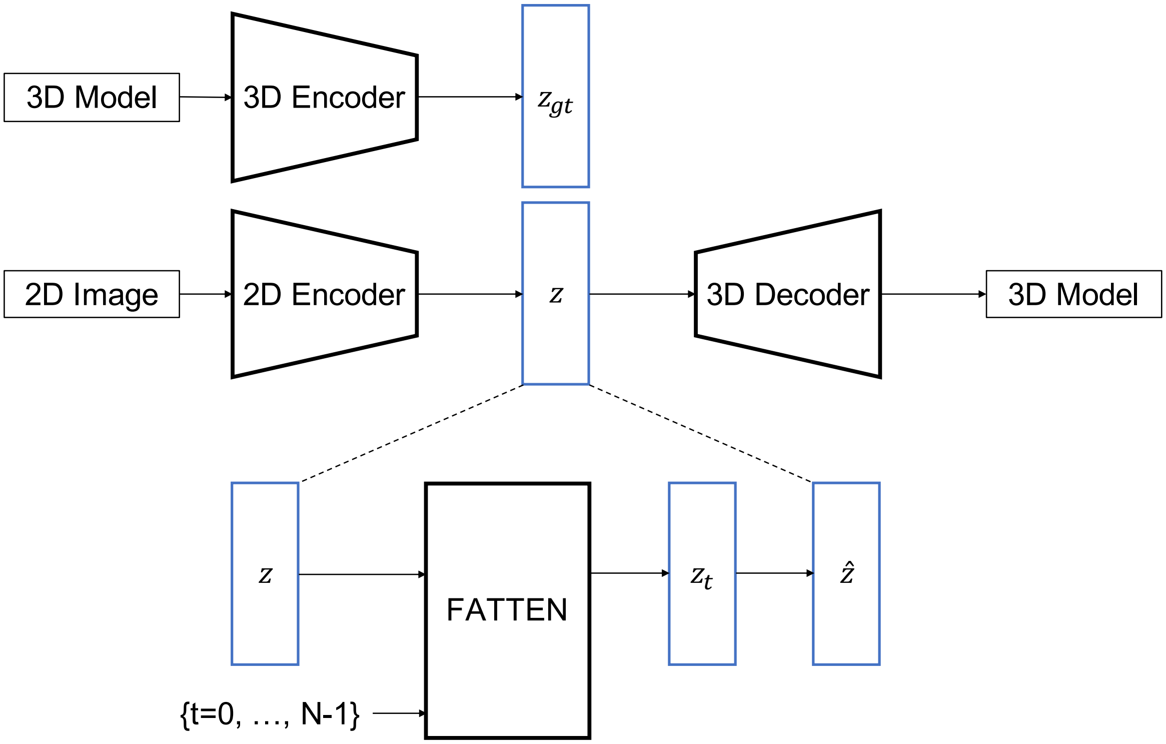


Figure 2.4: Single view reconstruction with the BSP-Net. A 3D encoder extracts a ground-truth shape code from a 3D object, a 2D encoder extracts a shape code from a 2D image, and a decoder reconstructs a 3D model. FATTEN is applied to the shape code, which it augments along pose trajectories. The resulting pose-trajectory codes are combined into a shape code that is input to the BSP-Net decoder.

2.5.1 Single-View Reconstruction Model

We consider the recent BSP-Net [12], one of the most recent and successful 3D shape generators. This is an implicit model that, given a shape feature vector and a point in 3D, classifies the point as being inside or outside the shape. The object surface is modelled as the combination of a set of *binary space partitions*, *i.e.* hyper-planes in 3D space.

For single-view reconstruction, the BSP-Net is implemented with the auto-encoder structure of Figure 2.4. A 2D convolutional neural network is used as an image encoder that maps the input image I into a shape code

$$\mathbf{z} = \mathcal{E}_{\text{image}}(I). \tag{2.17}$$

The decoder takes this code and a set of 3D point coordinates $\mathcal{X} = \{\mathbf{x}_i\}$, classifying each point as being inside or or outside the shape, producing a set of binary labels

$$\mathcal{Y} = \mathcal{D}_{\text{shape}}(\mathbf{z}, \{x_i\}) \tag{2.18}$$

that, together with $\mathcal{X}$, characterize the shape. For example, the sets $\mathcal{S}$ and $\mathcal{Y}$ can be used to create a surface mesh using the marching cubes algorithm [76].

2.5.2 Shape Code Augmentation

To apply FATTEN to the single view reconstruction problem, $\mathbf{z}$ can be seen as a feature vector that encodes shape information but is sensitive to pose information. FATTEN can then be used to augment $\mathbf{z}$ with feature vectors corresponding to other views of the object and these feature vectors then combined to achieve a more robust shape code. For this, the shape code predicted from the input image is first augmented along a pose trajectory of N pre-defined view angles $\mathbf{t}_k$, as shown in Figure 2.4. The shape code $\mathbf{z}$ and a one-hot encoded target pose $\mathbf{t}_k$ are input to the FATTEN module, which synthesizes a new shape code

$$\mathbf{z}_k = \text{FATTEN}(\mathbf{z}, \mathbf{t}_k) \tag{2.19}$$

corresponding to pose $\mathbf{t}_k$ but maintaining the semantic information of $\mathbf{z}$. In the second step, the set of feature vectors $\{\mathbf{z}_k, k = 1, \ldots, N\}$ produced by FATTEN is then mapped into a single shape code $\hat{\mathbf{z}}$, as required by the BSP-Net decoder $\mathcal{D}_{\text{shape}}$. In this work, this is implemented with a simple average pooling operation

$$\hat{\mathbf{z}} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{z}_k. \tag{2.20}$$

Other reduction methods are discussed and compared in the experimental section.

2.5.3 Training

The BSP-Net is pre-trained as in [12] and fixed. The 2D encoder is used to extract the shape code **z** from input images. The training of FATTEN follows the procedure of Section 2.3.4 with one exception. For reconstruction, the goal is not to classify the input images, i.e. the semantic information is not in the form of class labels. The shape code should encode the shape of the object, instead of just its class. In the reconstruction task, ground-truth shapes are also available during training. In fact, they are used to train the image encoder $\mathcal{E}_{\text{image}}$. The corresponding shape codes $\mathbf{z}_{\text{gt}}$ summarize the semantic information required for this task. Hence, the classification loss of Section 2.3.4 is replaced by a regression loss, consisting of the L_2 distance between the feature vector $\mathbf{z}_k$ synthesized by FATTEN and the ground-truth code

$$L_{\text{s}}(\mathbf{z}_k, \mathbf{z}_{\text{gt}}) = ||\mathbf{z}_k - \mathbf{z}_{\text{gt}}||^2. \tag{2.21}$$

The FATTEN loss is finally given by

$$L(\mathbf{z}_k, \mathbf{t}_k, \mathbf{z}_{\text{gt}}) = \lambda L_p(\mathbf{z}_k, \mathbf{t}_k) + L_{\text{s}}(\mathbf{z}_k, \mathbf{z}_{\text{gt}}). \tag{2.22}$$

2.6 Experiments

We first train and evaluate the FATTEN model on the artificial ModelNet [137] dataset (Sec. 2.6.1), and then assess its feature augmentation performance on the one/few-shot object recognition task of [20] (Sec. 2.6.2). This is done by both training an SVM on augmented data, to evaluate augmentation performance separately, and integrating classifier design and FATTEN training with meta-learning (Sec. 2.6.3). Finally, we evaluate the FATTEN model on the single-view reconstruction task, using ShapeNet [8] (Sec. 2.6.4).

2.6.1 Feature Quality Evaluation

Dataset

ModelNet [137] is a 3D synthetic data set of 3D voxel grids. It contains 4000 shapes from 40 object categories. Given a 3D shape, it is possible to render 2D images from any pose. In our experiments, we follow the rendering strategy of [113]. 12 virtual cameras are placed around the object, in increments of 30 degrees along the z-axis, and 30 degrees above the ground. Example rendered views are shown in Fig. 2.3. The training and testing splits are those proposed in the ModelNet benchmark, namely 80 objects per category for training and 20 for testing. However, the dataset contains some categories of symmetric objects, such as “bowl”, which produce identical images from all views (see Fig. 2.3(b)) and some that lack any distinctive information across views, such as “plant” (see Fig. 2.3(c)). These objects are eliminated and only the remaining 28 object categories are used.

Implementation

To verify the generality of FATTEN, both VGG16 [108] and ResNet-101 [38] are adopted as backbone architectures in feature transfer experiments. All feature vectors $\mathbf{x}$ are collected from activations of the last fully-connected layer of networks fine-tuned on the training set, namely `fc7` in VGG16 and `pool5` in ResNet101. The pose predictor is trained with a learning rate of 0.01 for 1000 epochs, and evaluated on the testing corpus. The complete FATTEN model is then trained for 1000 epochs with a learning rate of 0.01. The angle range of $[0°, 360°]$ is split into 12 non-overlapping intervals of size 30° each, labeled as 0-11. Each angle is then converted to a classification label based on the interval it belongs to.

Table 2.1: *Top:* Pose prediction error (in %); *Bottom:* Pose & category accuracy (in %) of generated features.

Degrees →	0	30	60	90	120	150	180
VGG16	72.3	2.2	1.1	4.0	0.9	1.0	18.5
ResNet-101	64.4	2.9	2.3	5.1	2.3	1.6	21.4

	Pose	**Object category**
VGG16	96.20	83.65
ResNet-101	99.95	84.13

Feature Transfer Results

The feature transfer performance of FATTEN is assessed in two steps. The accuracy of the pose predictor is evaluated *first*, with the results listed in Table. 2.1. The large majority of the errors have magnitude of 180°. This is not surprising, since ModelNet images have no texture. As as shown in Fig. 2.3(d)-(e), object views that differ by 180° can be similar or even identical for some objects. However, this is not a substantial problem for transfer. Since two feature vectors corresponding to the 180° difference are close to each other in feature space, to the point where the loss cannot distinguish them clearly, FATTEN will generate target features close to the source, which is the goal anyway. If these errors are disregarded, the pose prediction has accuracy 90.8% for VGG16 and 85.8% for ResNet-101.

The *second* evaluation step measures the feature transfer performance of the whole network, given the pre-trained pose predictor. During training, each feature in the training set is transferred to all 12 views (including identity). During testing, this is repeated for each test feature. The accuracy of the pose and category prediction of the features generated on the test corpus, is listed in Table 2.1. Note that, here, category refers to object category or class. It is clear that on a large synthetic dataset, such as ModelNet, FATTEN can generate features of good quality, as indicated by the pose prediction accuracy of 99.95% and the category prediction accuracy of 84.13% of the ResNet-101. Further, pose prediction error as well as pose and category accuracy of synthesized features is similar for the two backbones.

Table 2.2: Retrieval performance in mAP [%] of real and synthesized features, on the testing portion of ModelNet.

Feature type	(P)ose	(C)ategory	P + C
Real	54.58	32.71	23.65
Synthesized	77.62	28.89	11.07

Retrieval with Generated Features

A set of retrieval experiments is performed on ModelNet to further assess the effectiveness of FATTEN generated features. These experiments address the question of whether the latter can be used to retrieve instances of (1) the same class or (2) the same pose. Since all features are extracted from the VGG16 `fc7` layer, the Euclidean distance

$$d_1(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2 \tag{2.23}$$

is a sensible similarity measure for the purpose of retrieving images of the same *object category*. This is because the model is trained to map features with equal category labels to the same partitions of the feature space (enforced by the category loss L_c). However, d_1 is inadequate for *pose* retrieval. Instead, retrieval is based on the activation of the second fully-connected layer of the pose predictor $\mathcal{P}$, denoted by $\gamma(\mathbf{x})$. The *pose distance function* is then defined as

$$d_2(\mathbf{x},\mathbf{y}) = ||\gamma(\mathbf{x})-\gamma(\mathbf{y})||_2. \tag{2.24}$$

Finally, the performance of *joint* category & pose retrieval is measured with a combined distance, *i.e.*,

$$d_c(\mathbf{x},\mathbf{y}) = d_1(\mathbf{x},\mathbf{y}) + \lambda d_2(\mathbf{x},\mathbf{y}). \tag{2.25}$$

All queries and instances to be retrieved are based on features *synthesized* for the *testing* corpus of ModelNet. For each synthesized feature vector, three queries are performed: (1)

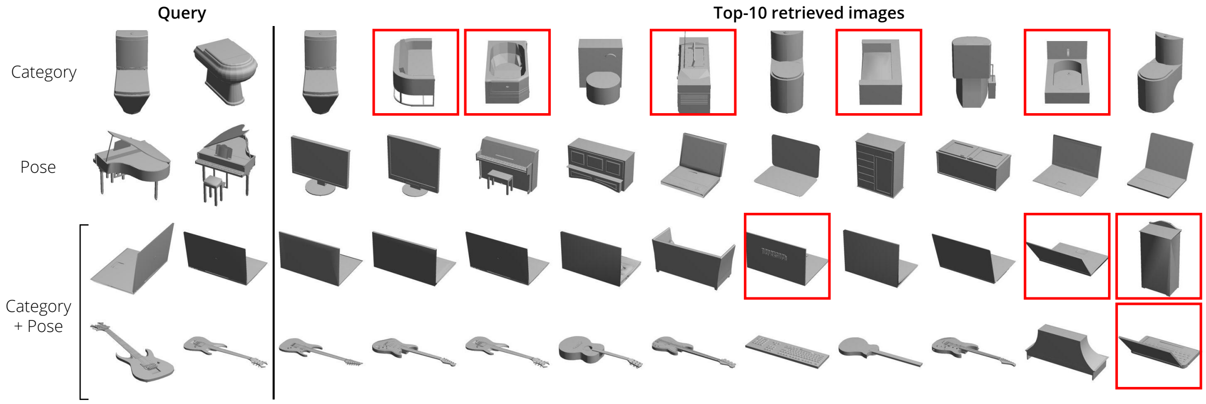


Figure 2.5: Exemplary retrieval results for the experiments of Sec. 2.6.1. Rows are annotated by the retrieval *type* and errors are highlighted in red. In the **query**, *(left)* shows the original image, *(right)* shows the original image corresponding to the pose of the generated feature.

Category, (2) *Pose*, and (3) *Category & Pose*. This is compared to the performance, on the same experiment, of the real features extracted from the testing corpus by the backbone CNN. Retrieval results are listed in Table 2.2 and some retrieval examples are shown in Fig. 2.5. The synthesized features enable a very high mAP for *pose retrieval*, even higher than the mAP of real features. This is strong evidence that FATTEN successfully encodes pose information in the transferred features. The mAP of the synthesized features is lower for *category retrieval* and the combination of both. However, the performance of the real features is also weak on these tasks. This could be due to a failure of mapping features from the same category into well defined neighborhoods, or to the distance metric used for retrieval. While retrieval performs a nearest neighbor search under these metrics, the network optimizes the cross-entropy loss on the softmax output(s) of both output branches of Fig. 2.2. The distance of (2.25) may be a particularly poor way to assess joint category and pose distances. In the following section, we will see that using a strong classifier (*e.g.*, a SVM) on the generated features produces significantly better results.

2.6.2 Data Augmentation on Few-shot Recognition

The experiments above provide no insight on whether FATTEN generates meaningful features for tasks involving real world data. In this section, we assess feature transfer performance on a one/few-shot object recognition problem. On this task, feature transfer is used for feature space "fattening" or *data augmentation*. The benchmark data is collected from SUN-RGBD [112], following the setup of [20].

Dataset

The whole SUN-RGBD dataset contains $10,335$ images and their corresponding depth maps. Additionally, 2D and 3D bounding boxes are available as ground truth for object detection. *Depth* (distance from the camera plane) and *Pose* (rotation around the vertical axis of the 3D coordinate system) are used as pose parameters in this task. The depth range of $[0,5)$ m is broken into non-overlapping intervals of size 0.5m. An additional interval $[5,+\infty)$ is included for larger depth values. For pose, the angular range of $[0^\circ,180^\circ]$ is divided into 12 non-overlapping intervals of size 15° each. These intervals are used for one-hot encoding and system training. However, to allow a fair comparison with AGA during testing, we restrict the desired pose t to take the values 45°, 75°, ..., 180°, prescribed in [20]. This is mainly to ensure that our system generates 11 synthetic points along the *Depth* trajectory and 7 along the *Pose* trajectory.

The first $5,335$ images of SUN-RGBD are used for training and the remaining 5000 images for testing. If only ground truth bounding boxes were used for object extraction, the instances would not be balanced *w.r.t.* categories, nor *w.r.t.* pose/depth values. To remedy this issue, a fast R-CNN [31] object detector is fine-tuned on the dataset and proposals with IoU > 0.5 (to ground truth boxes) and detection scores > 0.7 are used to extract object images for training. Since this strategy produces a large amount of data, the training set can be easily balanced per category, as well as pose and depth. In the testing set, only ground truth bounding boxes are used to exact objects. All source features are exacted from the penultimate (*i.e.*, `fc7`) layer of the fine-tuned fast R-CNN detector for all instances from both training and testing sets.

Evaluation is based on the source and target object classes in [20]. We define a source dataset $\mathcal{S}$ and two different (disjoint) target datasets, $\mathcal{T}_1$ and $\mathcal{T}_2$. A third target dataset is defined

Table 2.3: List of object categories in the source $\mathcal{S}$ training set and the two target/evaluation sets $\mathcal{T}_1$ and $\mathcal{T}_2$.

$\mathcal{S}$ (19, Source)			
bathtub	counter	lamp	sofa
bed	desk	monitor	table
bookshelf	door	night stand	tv
box	dresser	pillow	toilet
chair	garbage bin	sink	

$\mathcal{T}_1$ (10)		$\mathcal{T}_2$ (10)	
picture	stove	mug	microwave
whiteboard	cabinet	telephone	coffee table
fridge	printer	bowl	recycle bin
counter	computer	bottle	cart
books	ottoman	scanner	bench

Table 2.4: One-/Five-shot recognition accuracy for three recognition problems (from SUN-RGBD). Accuracies (in %) are averaged over 500 random runs. **Baseline** denotes the accuracy of a linear SVM, when trained on *single* instances of each class only.

		Baseline	**Hal.** [35]	**AGA** [20]	**FATTEN**
1–shot	$\mathcal{T}_1$ (10)	33.74	35.43	39.10	44.99
	$\mathcal{T}_2$ (10)	23.76	21.12	30.12	34.70
	$\mathcal{T}_3$ (20)	22.84	21.67	26.67	32.20
5–shot	$\mathcal{T}_1$ (10)	50.03	50.31	56.92	58.82
	$\mathcal{T}_2$ (10)	36.76	38.07	47.04	50.69
	$\mathcal{T}_3$ (20)	37.37	38.24	42.87	47.07

as the union of the first two, $\mathcal{T}_3 = \mathcal{T}_1 \cup \mathcal{T}_2$. Table 2.3 lists all the object categories in each set. The instances in $\mathcal{S}$ are collected from the training portion of SUN-RGBD only, while those in $\mathcal{T}_1$ and $\mathcal{T}_2$ are collected from the testing set. Further, $\mathcal{S}$ does not have class overlap with any $\mathcal{T}_i$, which ensures that FATTEN has no access to shared knowledge between training/testing images or classes.

Table 2.5: One-shot recognition accuracy under three meta-learning setups. Accuracies (in %) are averaged over 500 random runs. **SVM** denotes a linear SVM trained on *single* instances of each class. **G+SVM** denotes a linear SVM trained on *augmented* instances of each class.

		M	**G+M**	**GM+M**	**Imag.** [132]	**SVM**	**G+SVM**
$\mathcal{T}_1$ (10)	**Prototypical**	44.52	33.26	46.78	43.13	33.74	44.99
	Relation	41.25	36.75	44.45	43.86		
$\mathcal{T}_2$ (10)	**Prototypical**	34.22	23.69	37.08	31.64	23.76	34.70
	Relation	31.74	26.73	35.99	34.80		
$\mathcal{T}_3$ (20)	**Prototypical**	28.79	19.30	35.89	32.52	22.84	32.20
	Relation	34.02	27.71	33.74	29.01		

Table 2.6: One-shot recognition accuracy of the ”**GM+M**” setup for different embedding dimensions. Accuracies (in %) are averaged over 500 random runs.

Dimension	**64**	**256**	**1024**	**4096**
Prototypical	45.07	46.78	43.00	40.39
Relation	42.59	43.39	43.82	44.45

Implementation

Predictors of *pose* and *depth* are trained with a learning rate of 0.01 for 1000 epochs. The FATTEN network is fine-tuned, starting from the weights obtained from the ModelNet experiment of Sec. 2.6.1, with a learning rate of 0.001 for 2000 epochs. The classification problems on $\mathcal{T}_1$ and $\mathcal{T}_2$ are 10-class problems, whereas $\mathcal{T}_3$ is a 20-class problem. As a *baseline* for one-shot learning, we train a linear SVM using *only* a single instance per class. We then feed those same instances into the FATTEN network to generate artificial features for different values of depth and pose, in particular, 11 values for depth and 7 for pose. After feature synthesis, a linear SVM is trained with the same parameters on the *augmented* (“fattened”) feature set (source and target features).

Results

Table 2.4 lists the averaged one-shot (and five-shot) recognition accuracies (over 500 random runs) for all three evaluation sets $\mathcal{T}_i$. These are compared to the recognition accuracies of two data augmentation methods from the literature, feature hallucination [35] and AGA [20]. Table 2.4 supports the following conclusions. *First*, when compared to the SVM baseline, FATTEN achieves a remarkable and consistent improvement of around 10 percentage points on all evaluation sets. This indicates that FATTEN can actually embed the pose information into features and effectively "fatten" the data used to train the linear SVM. *Second*, and most notably, FATTEN achieves a significant improvement (about 5 percentage points) over AGA, and an even larger improvement over the feature hallucination approach of [35]. The improved performances of FATTEN over AGA and AGA over hallucination show that it is important (1) to exploit the structure of the pose manifold (which only FATTEN and AGA do), and (2) to rely on models that can capture defining properties of this manifold, such as continuity and smoothness of feature trajectories (which AGA does not).

While feature hallucination works well in the ImageNet1k low-shot setup of [35], Table 2.4 shows only marginal gains over the baseline (especially in the one-shot case). There may be several reasons as to why it fails in this setup. *First*, the number of examples per category (k in the notation of [35]) is a hyper-parameter set through cross-validation. To make the comparison fair, we chose to use the same value in all methods, which is $k = 19$. This may not be the optimal setting for [35]. *Second*, we adopt the same number of clusters as used by the authors when training the generator. However, the best value may depend on the dataset (ImageNet1k in [35] *vs.* SUN-RGBD here). Without clear guidelines of how to set this parameter, it seems challenging to adjust it appropriately. *Third*, all results of [35] list the top-5 accuracy, while we use top-1 accuracy. *Finally*, FATTEN takes advantage of pose and depth to generate features, while the hallucination feature generator is *non-parametric* and does not explicitly use this information.

The improvement of FATTEN over AGA can most likely be attributed to (1) the fact that AGA uses separate synthesis functions (trained independently) and (2) failure cases of the pose/depth predictor that determines *which* particular synthesis function is used. In case of the latter, generated features are likely to be less informative, or might even confound any subsequent classifier.

2.6.3 Augmentation with Meta-Learning

The experiments above show the effectiveness of pose guided data augmentation. However, the classifier is only trained a posteriori, on the augmented data. In this section, we evaluate the few-shot recognition problem with meta-learning, where the two are optimized jointly. Following the notations of Section 2.6.2, we use $\mathcal{S}$ during training, and $\mathcal{T}_i(i = 1,2,3)$ during testing.

Implementation

Meta-training involves the training of both feature extractor and classifier. We adopt the structure and fine-tune the parameters of the feature extractor used in Section 2.6.2 except the last fc layer, which is discussed in Section 2.6.3. During meta-training, per episode, we sample M_{tr} and M_{te} from $\mathcal{S}$, and calculate the meta loss. To thoroughly evaluate the effectiveness of meta-learning with data augmentation, we propose three different meta-learning setups. **M**: a meta-learner is trained on $\mathcal{S}$ without data augmentation. The meta loss is calculated on M_{tr} and M_{te} with (2.13) or (2.15). **G+M**: a meta-learner is trained on augmented data. All data points in M_{tr} are first augmented by FATTEN to assemble M_{tr}^{aug}. A meta loss is calculated on M_{tr}^{aug} and M_{te} with (2.13) or (2.15). Only the feature extractor and the meta-learning classifier are updated according to the loss. **GM+M**: A meta-learner is trained together with a data generator. All data points in M_{tr} and M_{te} are first augmented by FATTEN to assemble M_{tr}^{aug} and M_{te}^{aug}. The loss is calculated on M_{tr}^{aug} and M_{te}^{aug} with (2.16). Both the meta-learning classifier and the generator are updated according to the loss.

All meta-learning setups are tested with both the Prototypical [109] and the Relation

Network [118]. All meta-training parameters are those in the original papers with exception of the feature space dimension, which is discussed in Section 2.6.3. During testing, 1 instance per class is randomly sampled from $\mathcal{T}_i$ to form a 1-shot problem, and the remaining samples are used as testing set. The average accuracy over 500 episodes of different random samples is reported, to reduce the effects of randomness.

Results

Table 2.5 lists the averaged results for all three setups and two meta-learners. These are compared to the recent Imaginary [132] meta-learning based data augmentation method. Two baselines, SVM on 1-shot and SVM on augmentation, are also listed for comparison.

The Table 2.5 supports several conclusions. *First*, both the Prototypical and the Relation Network perform much better than the SVM, even without data augmentation. *Second*, when synthesized features are directly applied to the classifier, both meta-learning structures have a big performance drop. We believe this is due to mismatching. Because the data augmentation module is not optimized for the classifier, the synthesized features are not useful for meta-learning. *Third*, the performance of both meta-learners improves substantially, when data augmentation module and meta-learner are trained jointly. This again illustrates the benefits of data augmentation with FATTEN.

Surprisingly, the imaginary structure does not always outperform the meta-learner itself. This observation is not consistent with the results of [132]. We believe this is due to the difficulty of the dataset. Unlike ImageNet1k, our SUN-RGBD dataset only has 19 training classes. The may not create the data diversity needed by Imaginary to learn a good generator. This is less of a problem for FATTEN since, unlike unsupervised generators, it uses pose information as extra supervision. In result, it can generate more meaningful augmented data even for a limited meta-training set.

Ablation Study on Meta-learner Structure

Both prototypical and relation network use a feature embedding f_ϕ to map the high-dimensional input features into a low-dimension latent feature space. While the dimensions of this space are usually small (64) for meta-learning, this may not be ideal when the data augmentation module is introduced. Since the synthesized features are 4096 dimensional, we ablate the latent space dimension of the ”**GM+M**” configuration for the four dimensions reported in Table 2.6. All experiments are carried out on test set $\mathcal{T}_1$. These results show that the relation network benefits from a larger dimension, while the prototypical network prefers a lower dimensional embedding. It can also be seen that the prototypical network is much more sensitive to embedding dimension. For fair comparison, we use a 256-D prototypical and a 4096-D relation network in all experiments of Table 2.5.

2.6.4 Single-View Reconstruction

We next consider the task of single-view reconstruction.

Dataset

We use the 13 categories of ShapeNet [8] with more than $1,000$ shapes each, and the rendered views from 3D-R2N2 [15]. 80% of the objects from all categories are used for training, and the remaining for testing. Each object has 24 2D images rendered from random camera angles. During testing, the last generated view of each object is used as input. A symmetric Chamfer Distance is used for quantitative evaluation.

Implementation

The BSP-Net is first trained as in [12] and then frozen. A set of shape codes $\{\mathbf{z}_i\}$ is extracted from all 2D views in the training set and a set of ground-truth shape codes $\{\mathbf{z}_{i,\text{gt}}\}$ from

Table 2.7: Single-view reconstruction results per category and overall. Chamfer Distance is scaled by 1,000. BSP-Net* denotes results from original paper, BSP-Net those of our implementation. Best result in green, second best the best in red.

Method	**Atlas**	**OccNet**	**IM-Net**	**BSP-Net***	**BSP-Net**	**BSP-FAT**
airplane	0.587	1.534	2.211	0.759	0.713	0.636
bench	1.086	3.220	1.933	1.226	1.362	1.251
cabinet	1.231	1.099	1.902	1.188	1.184	1.132
car	0.799	0.870	1.390	0.841	0.864	0.831
chair	1.629	1.484	1.783	1.340	1.386	1.274
display	1.516	2.171	2.370	1.856	1.929	1.692
lamp	3.858	12.528	6.387	3.480	4.179	4.518
speaker	2.328	2.662	3.120	2.616	2.585	2.398
rifle	1.001	2.015	2.052	0.888	0.881	0.797
couch	1.471	1.246	2.344	1.645	1.627	1.464
table	1.996	3.734	2.778	1.643	1.641	1.499
phone	1.048	1.183	2.268	1.383	1.536	1.321
vessel	1.179	1.691	2.385	1.585	1.420	1.301
Overall	1.487	2.538	2.361	1.432	1.477	1.391

the corresponding 3D shapes. For each 2D view, the azimuth angular range of $[0°, 360°]$ is divided into 12 non-overlapping intervals of size 30° each. These intervals are used by the cross-entropy loss for pose training and the one-hot encoding of FATTEN.

A pose training set $\{(\mathbf{z}_i, p_i)\}$ is composed by shape codes and the corresponding discrete pose labels. The pose predictor is trained on $\{(\mathbf{z}_i, p_i)\}$ with a learning rate of 0.01 for 100 epochs. The training set $\{(\mathbf{z}_i, \mathbf{t}_p, \mathbf{z}_{i,\text{gt}}) | p = 0, \ldots, 11\}$ of FATTEN, where $\mathbf{t}_p$ is the one-hot encoding of pose p, is composed by all possible combinations of shape codes and target poses. FATTEN is then trained with a learning rate of 0.01 for 200 epochs, using $\lambda = 0.01$ in (2.22).

During inference, a shape code $\mathbf{z}$ is combined with all possible target poses $\{\mathbf{t}_p | p = 0, \ldots, 11\}$ to obtain a set of synthesized features $\{\mathbf{z}_p | p = 0, \ldots, 11\}$. The average of the features in this set is used by the BSP-Net decoder to reconstruct the 3D shape.

Table 2.8: Single-view reconstruction with different shape code reduction procedures.

Method	**BSP-Net**	**av pool**	**max pool**	**MLP**	**LSTM**
Chamfer	1.477	1.391	1.723	1.456	1.472
MSE	0.0092	0.0083	0.0102	0.0091	0.0092

Table 2.9: MSE loss of the shape code for different λ.

λ	1	0.5	0.1	0.05	0.01
MSE	0.0252	0.0234	0.0155	0.0126	0.0090

Results

The combination of BSP-Net and FATTEN, denoted BSP-FAT, is compared to Atlas [34], OccNet [80], IM-Net [13], and the baseline BSP-Net model [12]. The pytorch code provided by the authors of the BSP-Net is used to implement the model. This gives slightly different results from those reported in their paper, which are produced with tensor-flow code. We report the pytorch results for comparison, and also include the original results for reference.

The overall and category performance are shown in Table 2.7. BSP-FAT achieves the best overall performance and the best or the second best result in 10 of the 13 categories. When compared to BSP-Net, from which it differs only by addition of FATTEN feature augmentation, it has improved performance in 12 of the 13 categories. This shows that FATTEN can stably improve single-view reconstruction.

Ablations

We first compare different choices of shape code reduction. In addition to average pooling, we test max pooling, an MLP with two fully-connected layers, and an LSTM similar to that of [15]. In the cases of MLP and LSTM, the networks are included in the training. Table 2.8 shows that none of these choices outperforms average pooling. Further examination of the

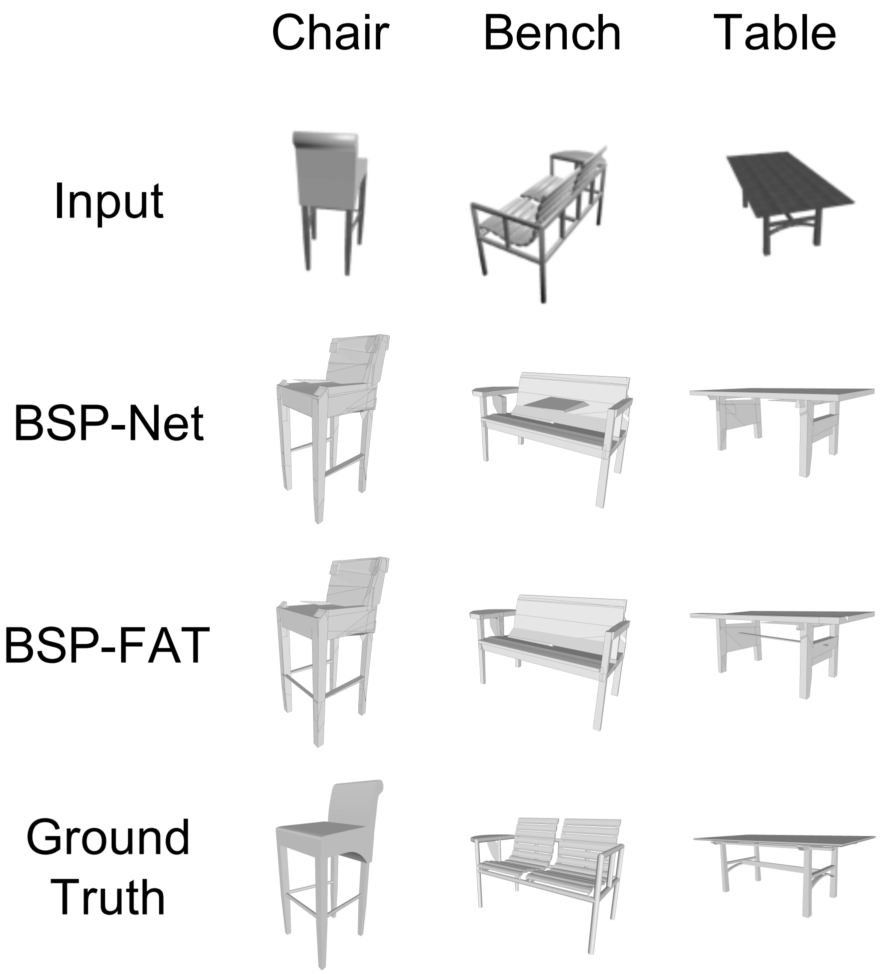


Figure 2.6: Examples of 3D reconstruction. A chair, a bench and a table object are shown. For all objects, the figure shows the input image together with the original BSP-Net results, results of BSP-FAT, and the ground truth.

MSE distance between predicted and ground truth shape code, shows that only average pooling decreases the MSE distance of the original BSP-Net. We next consider how the training MSE loss varies with λ in (2.22). Table 2.9 shows that when λ is too large, giving the pose prediction too much weight, the shape code cannot be trained to produce a low MSE error, which is important for good reconstruction performance.

Visualization

Figure 2.6 shows some typical examples of how pose transfer improves single-view reconstruction. All three objects are reconstructed from a view where part of the object is barely visible or ambiguous. For example, the chair has a footrest beam on the front, of which only a very small portion is visible in the image. The original BSP-Net reconstruction fails to recover this shape feature. This is not the case when FATTEN is used, since the beam can be recovered by pose transfer from other views. Similarly, the table has an underlying beam that is hard to identify in the image, but recovered by FATTEN. For the bench, the input view is such that the right part of the bench can be confused for a small table, which results in the wrong reconstruction by the BSP-Net. FATTEN corrects this problem by pose transfer, leveraging the fact that this ambiguity does not occur in other views. These examples show that the perceptual gains of FATTEN are even larger than the gains in Chamfer distance suggest. Frequently, the reconstructions improve by addition of shape features, such as the footrest above, that can occupy few voxels but are semantically significant.

2.7 Conclusion

The proposed architecture to data augmentation in feature space, FATTEN, aims to learn trajectories of feature responses, induced by variations in image properties (such as pose). These trajectories can then be easily traversed via *one* learned mapping function which, when applied to instances of novel classes, effectively enriches the feature space by additional samples corresponding to a desired change, *e.g.*, in pose. This "fattening" of the feature space is highly beneficial in situations where the collection of large amounts of adequate training data to cover these variations would be time-consuming, if not impossible. In principle, FATTEN can be used for any kind of desired (continuous) variation, so long as the trajectories can be learned from external data. By discretizing the space of variations, *e.g.*, the rotation angle in case of pose, we also effectively reduce the dimensionality of the learning problem and ensure that the approach scales favorably w.r.t. different resolutions of desired changes. Finally, it is worth pointing out that feature space transfer via FATTEN is not limited to object images; rather, it is a generic architecture in the sense that any variation could, in principle, be learned and transferred.

Chapter 2 is, in full, based on the material as it appears in the publication of “Feature Space Transfer for Data Augmentationn”, Bo Liu, Xudong Wang, Mandar Dixit, Roland Kwitt, and Nuno Vasconcelos, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2018. The dissertation author was the primary investigator and author of this material.

Chapter 3

Sparse Pose Trajectory Completion

3.1 Introduction

It is known that an object viewed from varying angles spans a manifold in the space of images. Characterization of these manifolds is important for many problems in computer vision, including 3D scene understanding and view invariant object recognition. However, these manifolds are quite difficult to learn. This is, in part, because most object datasets do not contain a dense sampling of object views.

Datasets such as ImageNet or COCO, favor diversity of instances per object class over diversity with respect to object views. The situation is different for synthetic datasets, such as ModelNet [137] or ShapeNet [8], which include large numbers of views per object instance. There is, however, a large domain gap between synthetic datasets and datasets of natural images. Models trained on synthetic images, therefore, cannot be deployed on natural images directly.

Obtaining natural image datasets that have a large number of instances per object class and a dense pose trajectory per instance, does not seem very feasible at this point. Therefore, a possible solution to modeling pose manifold could be devised using *novel view synthesis* techniques [151, 122] that leverage the synthetic ShapeNet dataset and a *domain transfer* mapping learned between synthetic and real images [46]. A view synthesis framework can generate a dense trajectory of object poses, given an input image, and a domain transfer function could map each generated pose, individually, into the image space. Such an approach, however, ignores the problem that the transfer has to preserve object identity across all views.

This problem is illustrated with an example in Fig. 3.1 (middle), which shows two objects of same shape but different appearance, and their trajectory in image space as a function of view angle. What is also shown is the trajectory spanned by synthetic images from the CAD model of the objects. Since the CAD model does not capture object appearance, the two image domain trajectories map into a single trajectory in the synthetic domain. Hence, methods that rely on synthetic view synthesis and individual view transfer are likely to oscillate between the synthesis

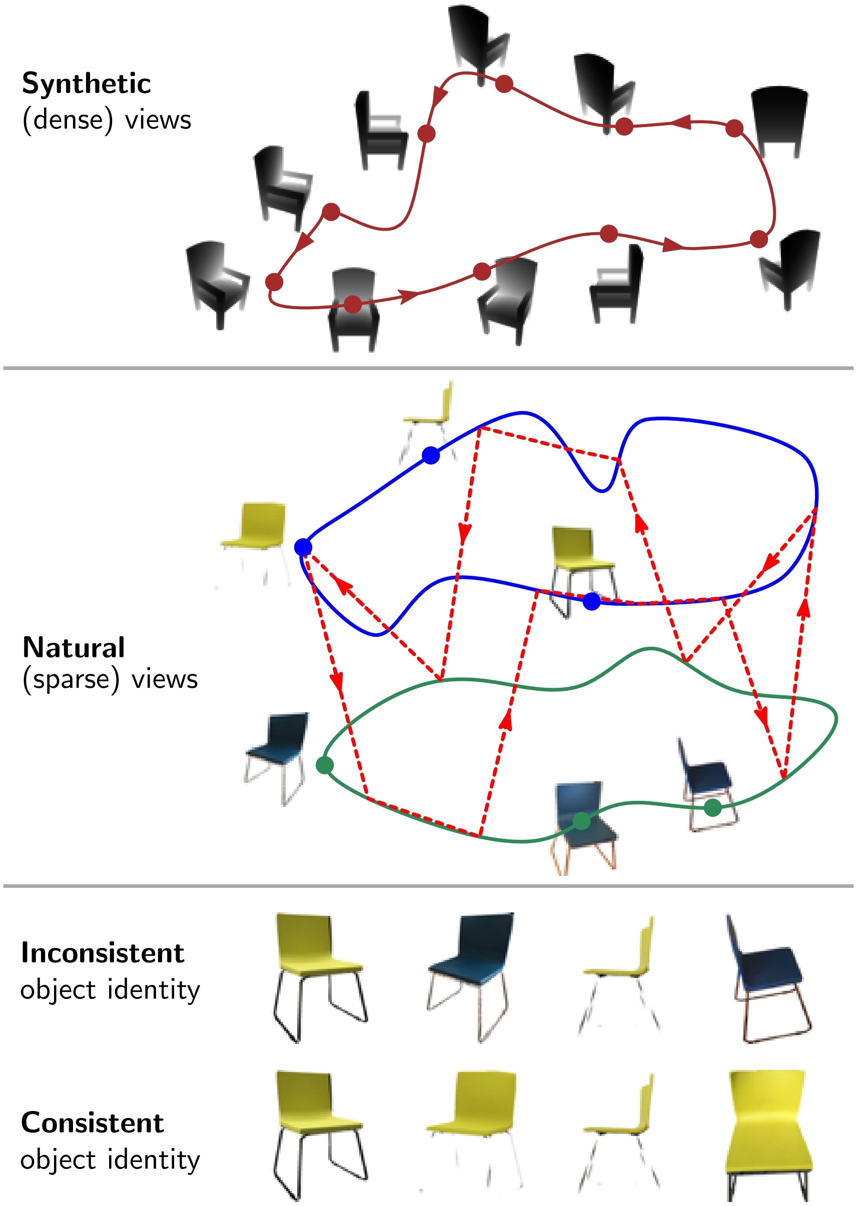


Figure 3.1: View synthesis via transfer from synthetic data. A densely sampled set of synthetic object views is transferred to the natural image domain, where only sparse views are available. Since synthetic images lack a rich characterization of appearance, the transfer can produce a trajectory (shown in red) that oscillates between objects of *similar shape* but *different appearance*. This is likely when synthetic views are transferred individually. Consistency of object identity requires pose trajectory transfer, *i.e.*, the transfer of the entire pose trajectory rather than independent views.

of images of the two objects (red-dashed trajectory). Image based domain transfer, therefore, does not suffice to solve the natural view synthesis problem. Instead, there is a need for *pose*

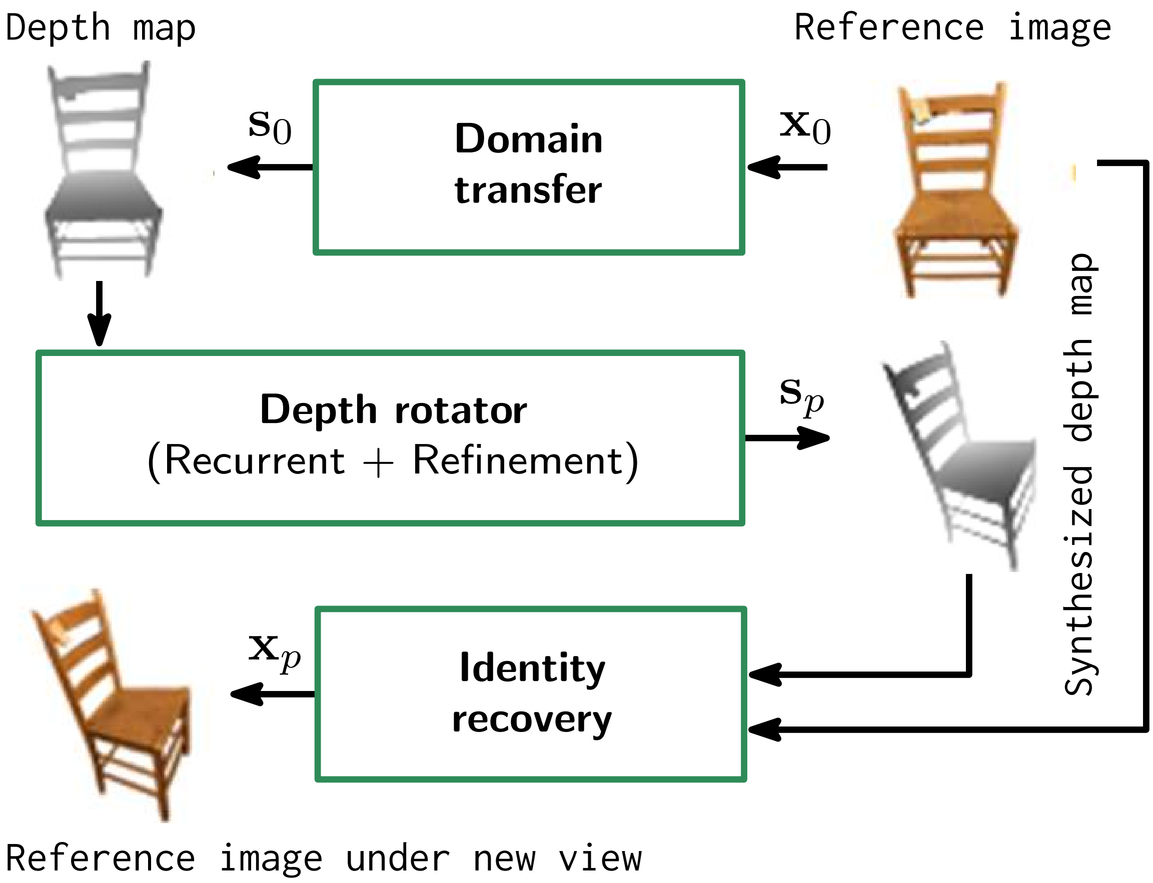


Figure 3.2: Illustration of the proposed **DRAW** approach. Arrows show the direction from inputs to outputs.

trajectory transfer approaches, *i.e.*, methods that transfer entire pose trajectories rather than one view at a time. This problem has some similarities to previous work on the hallucination of view changes on scene images [151]. However, such methods assume dense view supervision, which is only available in synthetic or video domains.

In this work, we address the more challenging problem of pose trajectory transfer with *sparse natural image data*. Unlike ShapeNet, most natural image datasets consist of object instances that are represented by a very few canonical views (*e.g.*, a chair instance captured in ∼1-5 poses). Such sparsity of poses, makes it impossible to directly uncover the underlying manifold by training or fine-tuning the existing view synthesis models [122, 143]. The pose trajectory must, therefore, be transferred from a densely represented latent space to the sparse image space.

We propose a formulation where an object's shape space serves as the latent space for view synthesis and transfer. Texture-less 3D CAD models are readily available online for many known object classes [117] and can be used to sample dense views of the object in the shape space. A trajectory in this auxiliary space can then be used to interpolate between the sparse poses of the object images. More formally, given a reference image $\mathbf{x}_0$, a depth map $\mathbf{s}_0$ is first synthesized. A complete pose trajectory, $\mathbf{s}_1, \ldots, \mathbf{s}_N$, is then generated in the latent space of CAD-based depth maps, and used to provide cross-modal guidance for the modeling of the pose trajectory in image space.

To guarantee *object identity* across the pose trajectory, we introduce the *Domain tRAnsferred vieW synthesis (DRAW)* architecture, as shown in Fig. 3.2. This consists of three modules. A *domain transfer* module is first used to translate the reference image $\mathbf{x}_0$ into the depth map $\mathbf{s}_0$. A *depth rotator* is then applied to synthesize a depth map $\mathbf{s}_p$ corresponding to a new view point. Finally, a *identity recovery network* is used to generate a new image $\mathbf{x}_p$ of the object under the new viewpoint, based on the rotated depth map and the original image.

While the domain transfer and depth rotator modules are variants of the previously studied problems of image translation and synthetic view synthesis, respectively, identity recovery poses a new challenge, critical to the generation of pose trajectories of consistent object identity. As shown in Fig. 3.2, it requires the disentanglement of the shape and appearance components of the reference image $\mathbf{x}_0$, and the combination of the appearance information with the synthetically generated new view $\mathbf{s}_p$ of the object shape.

To achieve this goal, we introduce a new *identity recovery network* that takes, as input, the reference image $\mathbf{x}_0$ and the rotated depth map $\mathbf{s}_p$ and predicts object views under all various combinations of domain (images *vs.* depth) and view angle (reference *vs.* p^{th} view). The requirement for these multiple predictions forces the network to more effectively disentangle shape and appearance information, enabling the synthesis of more realistic views of the reference object under new poses. This, in turn, enables DRAW to synthesize new view points of objects *without* requiring a training set of images with dense views on pose trajectory.

3.2 Related work

Image-to-image Transfer

Transferring images across domains has received substantial attention [46, 51, 71, 152]. Novel view synthesis can be considered as a special case of image-to-image transfer, where the source and target represent different views. There are, however, two key differences. First, view synthesis models need to explicitly infer shape from 2D image data. Second, while image transfer usually aims to synthesize style or texture, view transfer needs to “hallucinate” unseen shape information.

Domain Adaptation

Several methods [88, 27, 146, 104, 42, 126] have been proposed for domain adaptation of visual tasks. Generic domain adaptation aims to bridge the gap between domains by aligning their statistics. DRAW implements a much more complex form of unsupervised domain transfer, leveraging depth information to bridge the natural and synthetic domains and perform bi-directional transfer. Several domain adaptation approaches [41, 94] fuse color and depth features for pose estimation. Unlike these, DRAW relies on image-to-image transfer to leverage viewpoint supervision, which it then uses to decouple appearance and shape. This is critical to recover object identity.

Novel View Synthesis

Novel view synthesis addresses the generation of images of a given object under new views. One possibility is to simply generate pixels in the target view [122, 143], using autoencoders [122] or recurrent networks [143]. To eliminate some artifacts of these approaches, Zhou *et al.* [151] proposes an appearance flow based transfer module, which reconstructs the target view with pixels from the input and a dense flow map. This, however, cannot hallucinate pixels missing in the source view. Park *et al.* [89] use an image completion module, after flow based image reconstruction, to compensate for this and [116] designs independent modules to predict dense flow and pixel hallucination.

All these methods require training sets with dense pose trajectories, *i.e.*, large sets of views *of the same object instance*. For example, previous works in [116, 122, 143, 151] assume views under 16 or 18 fold azimuth rotation, and the method of Park *et al.* [89] requires additional 3D supervision. Hence, existing novel view synthesis methods are usually trained on and applied to ShapeNet [8]. Training or fine-tuning these models on the extremely sparse pose trajectories available in natural image datasets does not yield good results. This is shown in our experiments. DRAW makes up for the severe under-representation of image poses with the help of cross-modal supervision. A CAD based object model that generates dense views in the space of depth maps helps to transfer this trajectory to the space of natural images. The only other example of novel view synthesis on natural images is the work of [28] using the KITTI dataset. However, the generated views are restricted to a few frames and view point supervision is required. In contrast, DRAW generates a large set of views and can be deployed without pose supervision.

Human Pose Transfer

In this setting, the goal is to transfer a person across poses. Some recent works [78, 147, 85] have addressed this task, leveraging the availability of multi-pose datasets such as DeepFashion [73]. However, besides view points, these methods assume key point supervision [78] or leverage [85] pre-trained dense human pose estimation networks [1]. In summary, all these methods require additional supervision and are only applicable to human pose.

Single Image 3D Reconstruction

Many recent works have proposed to extract 3D shape from a single 2D image. With the availability of large-scale 3D CAD datasets, such as ShapeNet [8] and Pix3D [117], remarkable results have been achieved on this task [136, 15, 22]. Extraction of 3D shape can be useful for our purpose since an extracted 3D model can be trivially used to generate dense views of the object in the depth space. The depth rotation and refinement module (see section 3.4.1) seems to implement this implicitly as it simulates rotation of the object in depth space. An important distinction, however, between these methods and DRAW, is that the latter does not use any explicit 3D supervision for training.

3.3 DRAW

3.3.1 Architecture overview

The problem can be seen as one of domain adaptation, where data from a *source domain* (CAD-based depth maps), $\mathbb{S}$, for which view point annotations are available, is used to improve the performance of a task (view synthesis) in a *target domain* (images), $\mathbb{T}$, where this is inaccessible. As illustrated in Fig. 3.2, this allows the decomposition of the view generation problem into simpler tasks: a domain adaptation component, which maps images into depth maps and vice-versa, and a geometric component, implemented as a 3D rotation of the object. We propose three new modules to implement these tasks: a *domain transfer* module, a *depth rotator* and an *identity recovery* module.

The domain transfer module, $\mathcal{F}$, establishes a mapping from the target domain $\mathbb{T}$ of RGB images to the source domain $\mathbb{S}$ of depth maps. It implements

$$\mathcal{F} : \mathbb{T} \to \mathbb{S}, \quad \mathbf{x}_0 \mapsto \mathcal{F}(\mathbf{x}_0) = \mathbf{s}_0 \ , \tag{3.1}$$

where $\mathbf{x}_0$ and $\mathbf{s}_0$ are a reference image and a depth map of identical azimuth angle, respectively. The depth rotator module implements

$$\mathcal{G}(\mathbf{s}_0, p) = \mathbf{s}_p \tag{3.2}$$

for $p = 1, \dots, N-1$. It takes the depth map $\mathbf{s}_0$ associated with the reference view and synthesizes the depth maps associated with all other $N-1$ views. This is realized by two sub-modules. A recurrent rotator that generates novel depth map views and a refinement operator that leverages information from all synthesized depth maps to refine each of them. Finally, the identity recovery module implements

$$\mathcal{H} : \mathbb{T} \times \mathbb{S} \rightarrow \mathbb{T}, \quad (\mathbf{x}_0, \mathbf{s}_p) \mapsto \mathcal{H}(\mathbf{x}_0, \mathbf{s}_p) = \mathbf{x}_p \ \ , \tag{3.3}$$

taking, as input, the reference view $\mathbf{x}_0$ and the synthesized depth map $\mathbf{s}_p$ to produce the synthesized view $\mathbf{x}_p \in \mathbb{T}$. As the name suggests, this modules aims to recover the identity of $\mathbf{x}_0$ under the view of $\mathbf{s}_p$.

Domain Transfer (DT)

To learn the domain transfer model $\mathcal{F}$, we assume the existence of a dataset with paired images and depth maps, such as Pix3D [117] or RGB-D [59]. This makes the learning of this module a fairly standard domain transfer problem, where the input is a RGB image and the output is a depth map. We rely on an image style transfer model, similar to [46], to perform the transfer. Essentially, it is a fully convolutional network implemented with ResNet blocks, outputting a depth map and a foreground mask that identifies pixels associated with the object. The object depth map is then obtained by a combination of the two. Experimentally, we found that the use of the mask enables cleaner depth maps, which eventually lead to better depth rotation results. We refer the reader to the suppl. material for full architecture details.

The quality of the synthesized depth map, $\mathcal{F}(\mathbf{x}_0)$, is assessed by the L_1 loss, *i.e.*, $\|\mathbf{s}_0 - \mathcal{F}(\mathbf{x}_0)\|_1$. Under the framework of [46], this is complemented with an adversarial loss that discriminates between synthesized and real depth maps. This adversarial loss is implemented with a pair-wise discriminator D between the real ($\mathbf{s}_0$) and synthesized depth maps, conditioned on $\mathbf{x}_0$. The module is trained by iterating between learning of the discriminator/critic, with loss

$$\mathcal{L}_{\mathtt{DT}}^{\mathtt{critic}}(D) = \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_0}\left[(1-D(\mathbf{x}_0,\mathbf{s}_0))^2\right] + \mathbb{E}_{\mathbf{x}_0}\left[(D(\mathbf{x}_0,\mathcal{F}(\mathbf{x}_0)))^2\right] \ . \tag{3.4}$$

and learning of the mapping $\mathcal{F}$, with loss

$$\mathcal{L}_{\mathtt{DT}}(\mathcal{F}) = \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_0}\left[\|\mathbf{s}_0 - \mathcal{F}(\mathbf{x}_0)\|_1\right] + \lambda_{\mathcal{F}}\ \mathbb{E}_{\mathbf{x}_0}\left[(1-D(\mathbf{x}_0,\mathcal{F}(\mathbf{x}_0)))^2\right] \ , \tag{3.5}$$

where $\lambda_{\mathcal{F}}$ is a multiplier balancing the importance of the two loss components. We have found that the addition of the adversarial loss helps enforce both sharpness of the output and consistency between input and output; consequently, we use this approach for learning all modules of DRAW.

Depth Rotation & Refinement (DR)

The introduction of depth as an intermediate representation for image translation transforms view rotation into a geometric operation that can be learned from datasets of CAD models. Rather than reconstructing pixel depths from an appearance map, *e.g.*, using a dense appearance flow model [151], novel depth views are synthesized from a reference depth view $\mathbf{s}_0$. This leverages the fact that CAD datasets have many views per object and the view angles are known. The generation of novel depth views is implemented with the combination of (1) a depth map generator and (2) a 3D refinement module. The *depth map generator*, $\mathcal{G}_1$, is based on a recurrent network, which takes the reference depth map $\mathbf{s}_0$ as input and outputs a sequence of depths maps, *i.e.*,

$$\mathbf{s}_p = \mathcal{G}_1(\mathbf{s}_0, p), \quad p = 1,\ldots,N-1 \ , \tag{3.6}$$

where p is the azimuth angle. Our implementation of $\mathcal{G}_1$ is based on the ConvLSTM with skip connections of [116]. Given a set of depth maps $\{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{N-1}\}$ from N view points, the recurrent generator aims to minimize

$$\mathcal{L}_{\mathtt{RecGen}}(\mathcal{G}_1) = \sum_{p=0}^{N-1} \mathbb{E}_{\mathbf{s}_0}\left[\|\mathbf{s}_p - \mathcal{G}_1(\mathbf{s}_0, p)\|_1\right] \ . \tag{3.7}$$

The *refinement module* enforces consistency among neighboring views, via a 3D convolutional network that leverages the information of nearby synthesized views to refine each synthesized view. The N depth maps synthesized by the rotator are first stacked into a 3D volume[1] $\mathbf{s}' = [\mathbf{s}'_0 \oplus \mathbf{s}'_1 \oplus \ldots \oplus \mathbf{s}'_{N-1}]$. To ensure the refinement of the end views, *e.g.*, $\mathbf{s}_{N-1}$, cyclic padding is used on the third dimension. The volume $\mathbf{s}'$ is then processed by

$$\mathbf{s}'' = \mathcal{G}_2(\mathbf{s}') \ . \tag{3.8}$$

$\mathcal{G}_2$ is implemented by multiple layers of 3D convolutions with skip connections, to produce a 3D volume of concatenated refined depth maps $\mathbf{s}'' = [\mathbf{s}''_0 \oplus \mathbf{s}''_1 \oplus \ldots \oplus \mathbf{s}''_{N-1}]$. 3D refinement is supervised by a L_1 loss, *i.e.*,

$$\mathcal{L}_{\mathtt{3D}}(\mathcal{G}_2) = \sum_{p=0}^{N-1} \mathbb{E}_{\mathbf{s}''}\left[\|\mathbf{s}_p - \mathbf{s}''\|_1\right] \ . \tag{3.9}$$

This is complemented by an adversarial loss based on a pair-wise volume discriminator D_V, between the CAD-based depth map volume ($\mathbf{s}$) and the synthesized one ($\mathbf{s}''$), conditioned on $\mathbf{s}'$. The discriminator/critic loss is

$$\mathcal{L}_{\mathtt{V}}^{\mathtt{critic}}(D_V) = \mathbb{E}_{\mathbf{s}',\mathbf{s}}\left[(1 - D_V(\mathbf{s}', \mathbf{s}))^2\right] + \mathbb{E}_{\mathbf{s}',\mathbf{s}''}\left[(D_V(\mathbf{s}', \mathbf{s}'')^2\right] \ , \tag{3.10}$$

[1] $\oplus$ denotes concatenation along the third dimension.

while $\mathcal{G}_1$ and $\mathcal{G}_2$ are supervised by

$$\mathcal{L}_{\texttt{DR}}(\mathcal{G}_1,\mathcal{G}_2) = \mathcal{L}_{\texttt{RecGen}}(\mathcal{G}_1) \; + \; \lambda_{\texttt{3D}}\, \mathcal{L}_{\texttt{3D}}(\mathcal{G}_2) + \lambda_{\mathcal{G}}\, \mathbb{E}_{\mathbf{s}',\mathbf{s}''}\left[(1 - D_V(\mathbf{s}',\mathbf{s}''))^2\right] \; . \tag{3.11}$$

Identity Recovery (IR)

In standard domain transfer problems, the mapping between source and target domains is one-to-one. Each example in the source domain produces a different image in the target domain. This is not the case for the transfer between images and depth maps since, as illustrated in Fig. 3.1, objects of the same shape can have different appearance. Hence, the mapping between images and depth maps is not bijective. This poses no special problems to our domain transfer module, which implements a many-to-one mapping. On the other hand, it implies that it is impossible to recover the object identity uniquely from its depth map. It follows that, unlike the domain transfer module, the identity recovery model cannot be implemented with existing domain transfer networks. In addition to the depth map $\mathbf{s}_p$, this model must also have access to the reference image $\mathbf{x}_0$, *i.e.*, implement the mapping $\mathcal{H}$ of Eq. (3.3).

In a supervised regression setting, this mapping could be learned from triplets $(\mathbf{x}_0, \mathbf{s}_p, \mathbf{x}_p)$. However, we are aware of no datasets that could be used for this. Most image datasets do not contain depth information or view point labels. In general, it is difficult to even find multiple views of the same object with viewpoint annotations. In datasets such as Pix3D or RGB-D, there are at most a few views per object and these views are not aligned, *i.e.*, they change from object to object. Hence, $\mathcal{H}$ must be learned from unpaired data. This, however, is significantly more challenging than image-to-image transfer because $\mathcal{H}$ has to 1) *disentangle* the appearance and shape information of $\mathbf{x}_0$ and 2) *combine* the appearance information with the shape information of $\mathbf{s}_p$.

To enable this, we propose an encoder-decoder architecture. The encoder *disentangles* its input into a pair of shape and appearance parameters, via a combination of (1) a structure and

(2) an appearance predictor. The structure predictor implements the mapping $\mathbf{p} = \mathcal{P}(\mathbf{x})$ from the input image $\mathbf{x}$ to shape parameters $\mathbf{p}$, while the appearance predictor implements the mapping $\mathbf{a} = \mathcal{A}(\mathbf{x})$ from the input image $\mathbf{x}$ to appearance parameters $\mathbf{a}$. The decoder then *combines* these parameters into a reconstruction on its output, by taking a vector of concatenated appearance and shape parameters and decoding this representation into an image.

To force disentanglement, we exploit the fact that, while the object shape is captured by both its image and depth map, appearance is only captured by the image. It follows that (1) combining the shape information derived from domain A with the appearance information derived from domain B and (2) reconstructing should produce an image of the object in domain B under the view used in domain A. Hence, using both the image and shape domains as A and B, it should be possible to synthesize images with the four possible combinations of domain (image *vs.* depth map) and view (reference *vs.* target). By matching each of these four classes of synthesized images to true images of the four classes, we encourage the network to learn to disentangle and combine the shape and appearance representations.

In the multi-view setting, the four combinations are not available, since $\mathbf{x}_p$ is the target. Yet, the idea can be implemented with the remaining three combinations: reference image ($\mathbf{x}_0$), reference depth ($\mathbf{s}_0$) and target view depth ($\mathbf{s}_p$). This leads to the architecture of Fig. 3.3, which combines a pair of encoders and four decoders. The encoders are applied to the reference image $\mathbf{x}_0$ and the depth map $\mathbf{s}_p$. This results in two pairs of shape and appearance parameters

$$\mathbf{p}_r = \mathcal{P}(\mathbf{x}_0), \quad \mathbf{a}_r = \mathcal{A}(\mathbf{x}_0) \tag{3.12}$$

$$\mathbf{p}_s = \mathcal{P}(\mathbf{s}_p), \quad \mathbf{a}_s = \mathcal{A}(\mathbf{s}_p). \tag{3.13}$$

The decoders are then applied to the four possible *combinations* of these parameter vectors,

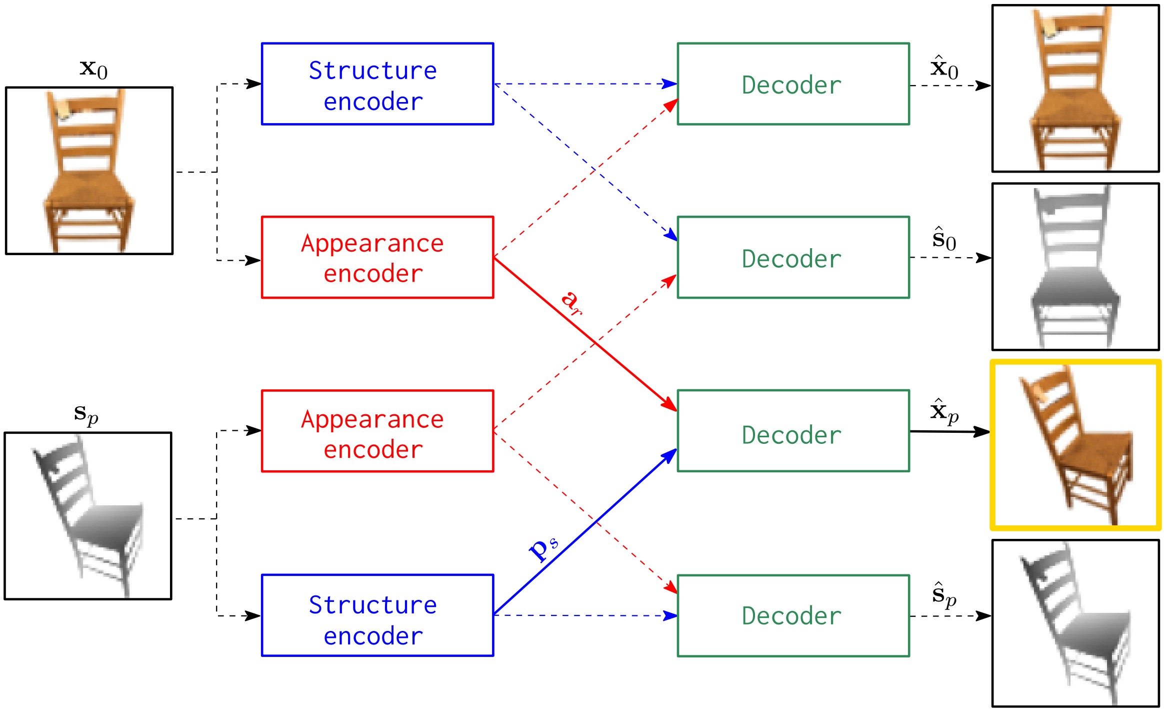


Figure 3.3: Data flow for the *identity recovery* module. Dashed-lines identify the training data flow, solid lines identify the data flow during inference. Blocks of equal color share parameters.

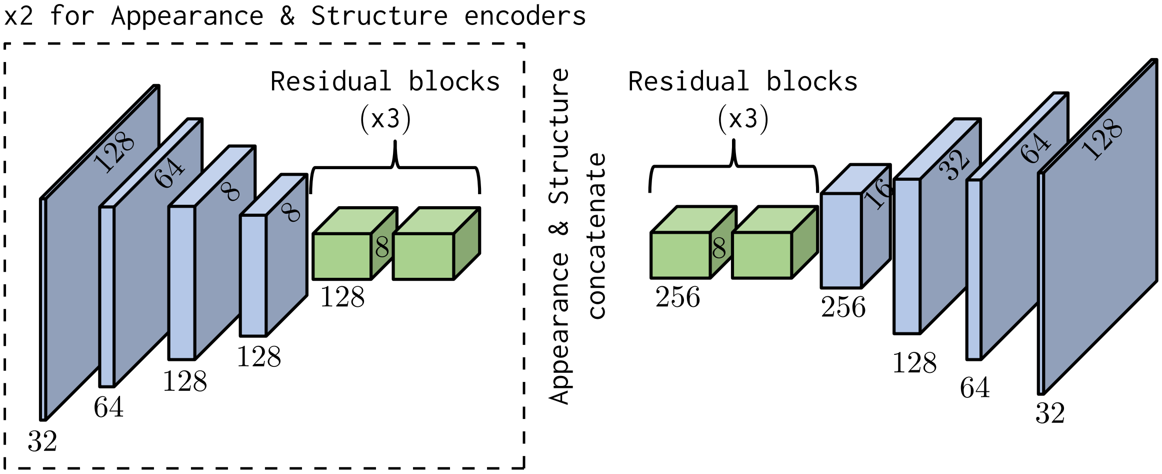


Figure 3.4: Architecture of the *identity recovery* module.

synthesizing four images,

$$\hat{\mathbf{x}}_0 = \mathrm{Dec}(\mathbf{p}_r, \mathbf{a}_r), \quad \hat{\mathbf{s}}_0 = \mathrm{Dec}(\mathbf{p}_r, \mathbf{a}_s), \tag{3.14}$$

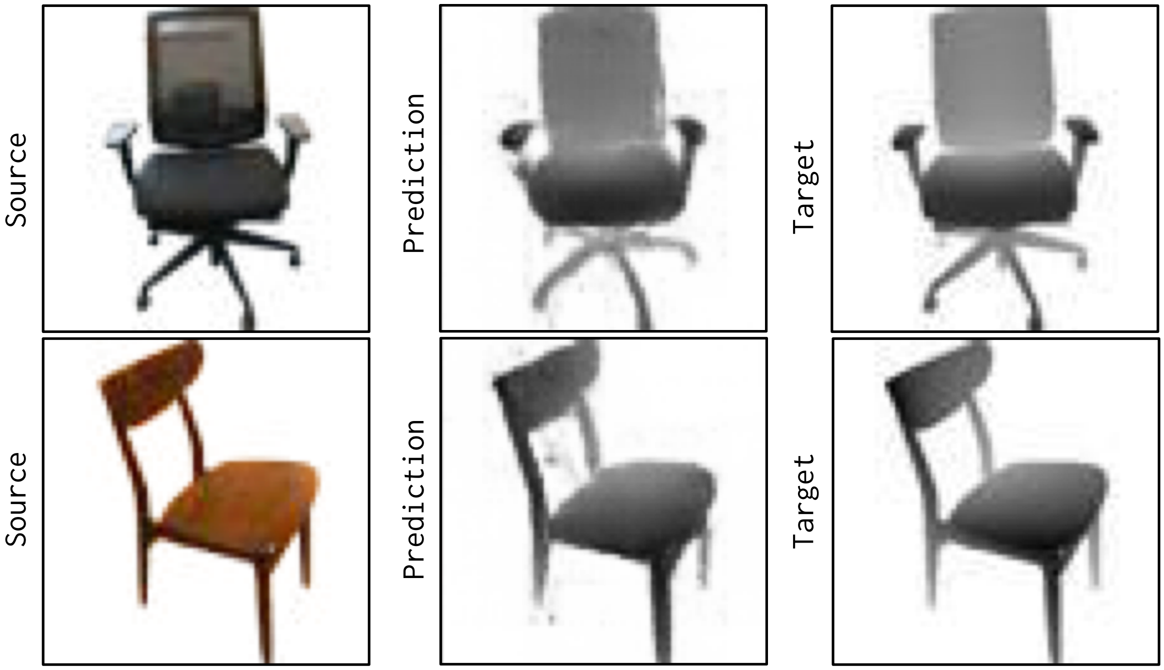


Figure 3.5: Example results for the *domain transfer* module.

$$\hat{\mathbf{x}}_p = \mathrm{Dec}(\mathbf{p}_s, \mathbf{a}_r), \quad \hat{\mathbf{s}}_p = \mathrm{Dec}(\mathbf{p}_s, \mathbf{a}_s). \tag{3.15}$$

As shown Fig. 3.3 (*right*), these are all possible combinations of shape and appearance from the real image with shape and appearance from the depth map “image.” To force the disentanglement into shape and appearance, the structure predictors, appearance predictors, and decoders share parameters. Note that this implies that only one encoder and one decoder are effectively learned. During inference, the target image $\mathbf{x}_p$ is obtained with

$$\hat{\mathbf{x}}_p = \mathrm{Dec}(\mathcal{P}(\mathbf{s}_p), \mathcal{A}(\mathbf{x}_0)) \ \ . \tag{3.16}$$

Training of the identity recovery model uses a mix of supervised and unsupervised learning. Since $\mathbf{x}_0, \mathbf{s}_0$, and $\mathbf{s}_p$ are available, they provide direct supervision for the synthesis of the combinations $\hat{\mathbf{x}}_0$, $\hat{\mathbf{s}}_0$, and $\hat{\mathbf{s}}_p$, respectively. Realized as a a supervised loss, we get

$$\mathcal{L}_{\mathtt{IR}}^{\mathtt{S}} = \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_0,\mathbf{s}_p}[||\mathbf{x}_0 - \hat{\mathbf{x}}_0||_1 + ||\mathbf{s}_0 - \hat{\mathbf{s}}_0||_1 + ||\mathbf{s}_p - \hat{\mathbf{s}}_p||_1], \tag{3.17}$$

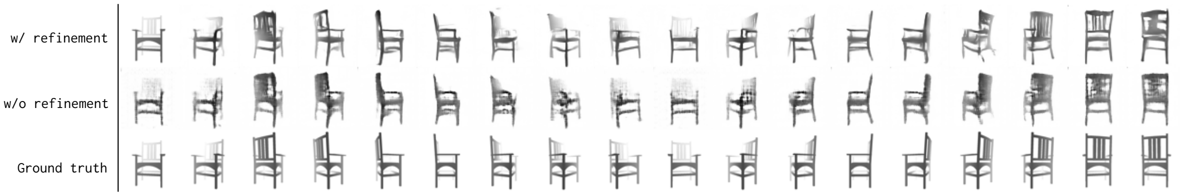


Figure 3.6: Qualitative depth rotation results w/ and w/o using 3D refinement, compared to the ground truth.

Table 3.1: Quantitative evaluation of the *depth rotator* module. For L_1, lower values are better; for SSIM, higher values are better.

View distance		1	2	3	4	5	6	7	8	9	All
w/o refinement	L_1	0.045	0.049	0.049	0.050	0.049	0.049	0.048	0.047	0.047	0.048
	SSIM	0.836	0.813	0.808	0.803	0.801	0.803	0.806	0.825	0.839	0.813
w/ refinement	L_1	0.031	0.036	0.037	0.038	0.038	0.037	0.035	0.031	0.026	0.035
	SSIM	0.945	0.913	0.907	0.898	0.897	0.902	0.915	0.940	0.975	0.918

ensuring the quality of *both* the disentanglement and the image synthesis. This is complemented by an adversarial loss where the combinations $(\mathbf{x}_0, \hat{\mathbf{s}}_0)$, $(\hat{\mathbf{x}}_0, \mathbf{s}_0)$ and $(\hat{\mathbf{x}}_p, \mathbf{s}_p)$ are all considered as fake pairs, to be indistinguishable from the real pair $(\mathbf{x}_0, \mathbf{s}_0)$. This pairwise discriminator/critic is trained with loss

$$\begin{aligned}\mathcal{L}_{\mathtt{IR}}^{\mathtt{critic}}(D) = \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_0}\left[(1 - D(\mathbf{x}_0, \mathbf{s}_0))^2\right] + \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_p}\left[(D(\mathbf{x}_0, \hat{\mathbf{s}}_0))^2\right] + \\ \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_0}\left[(D(\hat{\mathbf{x}}_0, \mathbf{s}_0))^2\right] + \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_p}\left[(D(\hat{\mathbf{x}}_p, \mathbf{s}_p))^2\right] \ ,\end{aligned} \tag{3.18}$$

while the encoder and decoder are trained with loss

$$\begin{aligned}\mathcal{L}_{\mathtt{IR}}(\mathcal{H}) = \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_p}[(1 - \mathcal{D}(\mathbf{x}_0, \hat{\mathbf{s}}_0))^2] + \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_p}[(1 - \mathcal{D}(\hat{\mathbf{x}}_0, \mathbf{s}_0))^2] + \\ \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_p}[(1 - \mathcal{D}(\hat{\mathbf{x}}_p, \mathbf{s}_p))^2] + \mathcal{L}_{\mathtt{IR}}^{\mathtt{S}} \ .\end{aligned} \tag{3.19}$$

Fig. 3.4 shows the structure of the identity recovery module.

Optimization Schedule

DRAW is trained in two stages to decouple domain transfer and view point synthesis. First, the depth rotation and refinement modules $(\mathcal{G}_1, \mathcal{G}_2)$, as well as its discriminator, D_V, are optimized with the losses of Eqs. (3.10) and (3.11). Once trained, these modules are frozen. The second stage then addresses end-to-end training of the domain transfer and identity recovery modules. The loss

$$\mathcal{L}(\mathcal{D}) = \mathcal{L}_{\texttt{DT}}^{\texttt{critic}}(D) + \lambda_2 \mathcal{L}_{\texttt{IR}}^{\texttt{critic}}(D) \tag{3.20}$$

is used to train discriminators/critics, and the loss

$$\mathcal{L}(\mathcal{F}, \mathcal{H}) = \mathcal{L}_{\texttt{DT}}(\mathcal{F}) + \lambda_1 \mathcal{L}_{\texttt{IR}}(\mathcal{H}) \tag{3.21}$$

supervises both parts.

3.4 Experiments

We evaluate DRAW using natural images from the Pix3D [117] dataset and synthetic images from the ShapeNet [8] dataset. To ensure enough diversity of instances in both datasets, we choose two categories: *chairs* and *tables*. First, we evaluate the three modules of DRAW separately on the *chair* class for: domain transfer between RGB images and depth maps, view synthesis or rotation simulation in the depth space, and identity recovery from depth to RGB. The L_1 difference norm and structural similarity measure (SSIM) are used as quantitative synthesis metrics. We then compare the performance of the full DRAW framework with three recent view synthesis methods [151, 122, 116] on *sparse pose completion* using the Pix3D dataset and a sub-sampled version of the ShapeNet dataset.

Datasets

On ShapeNet, 72 images of size 256 × 256 pixels are synthesized per CAD model, using 18 azimuth angles and elevations in $\{0^\circ, 10^\circ, 20^\circ, 30^\circ\}$. The dataset is split into 558 objects for training and 140 objects for testing [151] Pix3D combines 2D natural images with sparse views and 3D CAD models. Images and depth maps are cropped + resized to 256 × 256 pixel. The dataset includes multiple images aligned with each object. While DRAW does not require this, they are useful to evaluate identity recovery. Training and test sets are split based on objects to ensure that images with the same object do not appear in training *and* testing. This gives 758 training images from 150 objects and 140 test images from 26 objects.

3.4.1 Individual module evaluation

Domain Transfer (DT)

Since this is a fairly standard module, we do not carry out a detailed performance evaluation. Fig. 3.5 shows typical domain transfer results. In general, the predicted depth maps are fairly close to the ground-truth.

Depth Rotator (DR)

We compare the proposed combination of rotator + 3D refinement to a variant without the latter. Both models are trained on the depth maps from 18 ShapeNet views. Given a reference depth map, the task is to synthesize the remaining 17 depth maps. Fig. 3.6 shows some typical examples. Most depths maps are close to the ground truth, but refinement improves the rendering of fine details. Table 3.1 compares the L_1 and SSIM scores of the two methods, with refinement consistently improving the results for both scores.

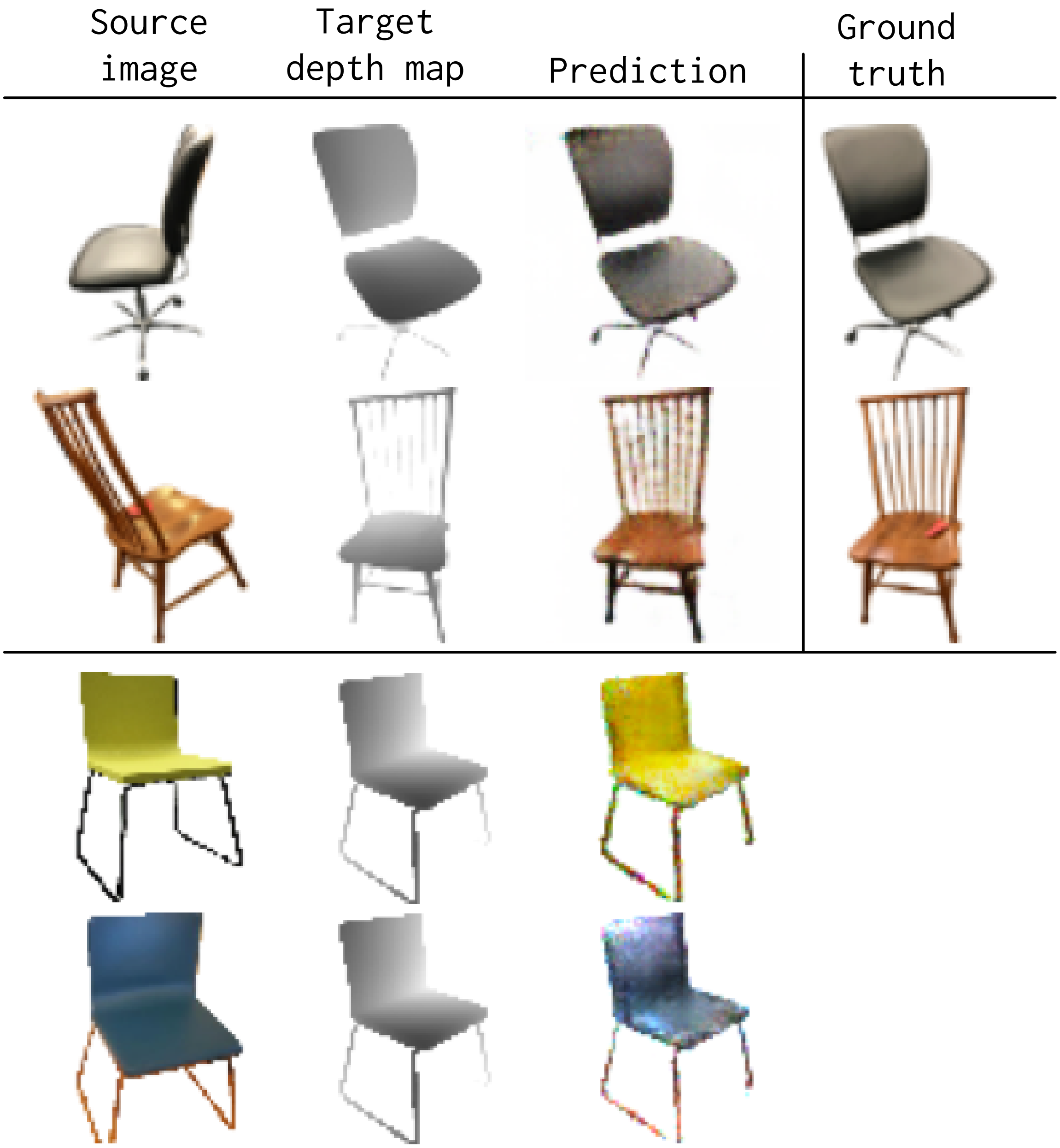
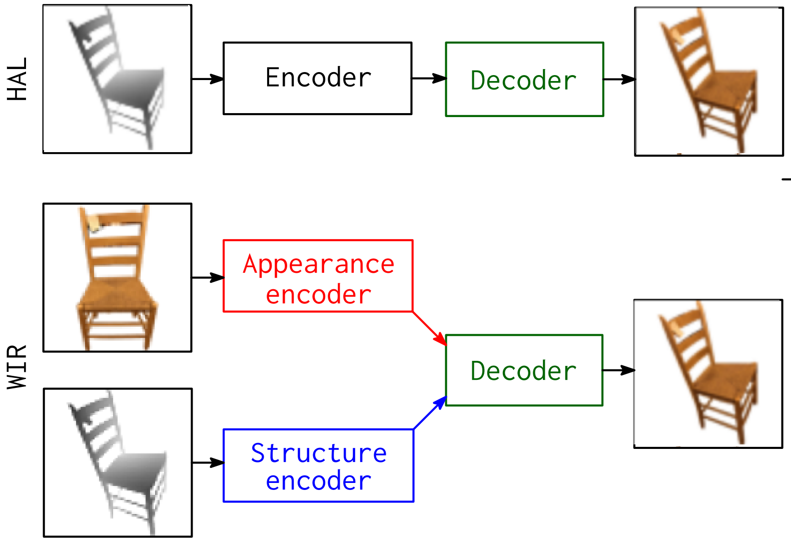


Figure 3.7: *Identity recovery*, using (1) different target depth maps (*top*) and (2) the same target depth map (due to the sparsity of pose trajectories, we don't have ground truth for the *bottom* part).

Identity Recovery (IR)

We consider two baselines and a loss function variation for comparison. The model in Fig. 3.8 (*top*) simply treats the problem as one of image to image translation. Since it



	L_1	SSIM
HAL	0.070	0.82
WIR	0.062	0.84
Ours	0.031	0.91
Ours + $\mathbf{x}_p$	0.029	0.92

Figure 3.8: Comparison of the identity recovery module of Fig. 3.3 to (1) a simple image-to-image translation model (HAL, *top*) and (2) a *weak identity recovery* module (WIR, *bottom*), using fewer disentanglement constraints.

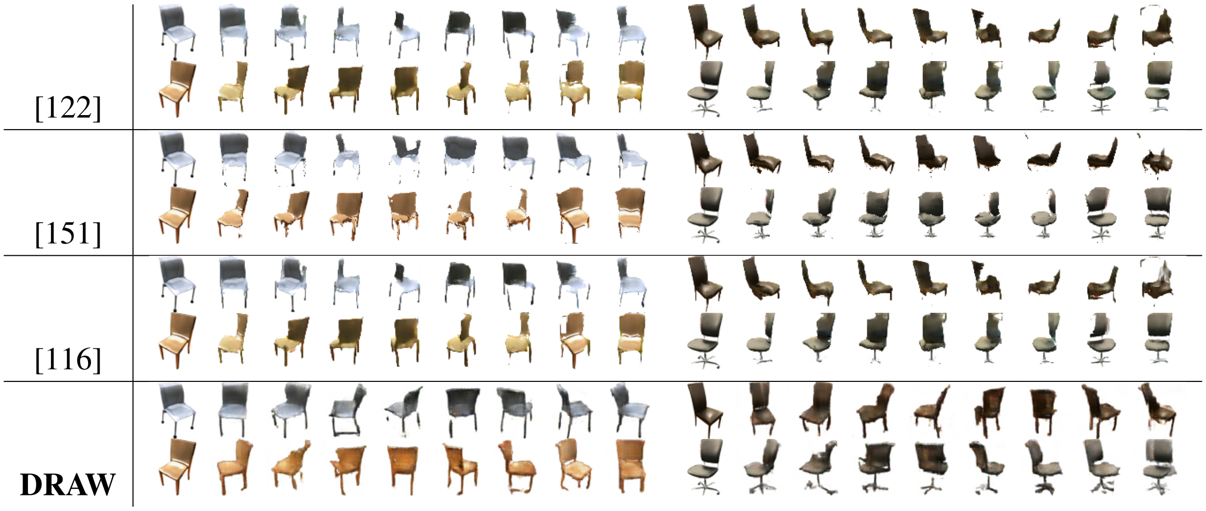


Figure 3.9: View synthesis comparisons on *chair* images from Pix3D. Only 9 out of 18 views are shown. Note that [116] is trained and tested with multiple views. DRAW generates the entire trajectory with a single image (best-viewed zoomed).

only has access to the depth map $\mathbf{s}_p$, it has to hallucinate the object appearance. We refer to it as the *hallucination* (HAL) model. The model of Fig. 3.8 (*bottom*) is a simpler variant of the identity recovery module of Fig. 3.3. It has access to both $\mathbf{x}_0$ and $\mathbf{s}_p$, but imposes much weaker disentanglement constraints because it does not require the synthesis of all combinations

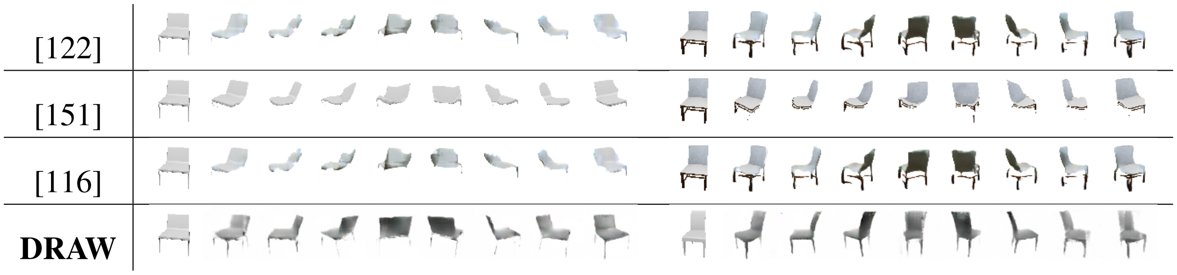


Figure 3.10: Comparison of view synthesis results on *chairs* from ShapeNet (best-viewed zoomed).

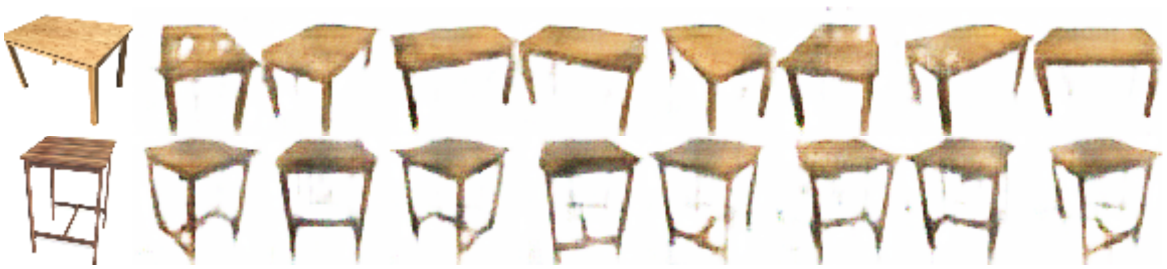


Figure 3.11: DRAW view synthesis on *table* images from Pix3D.

of shape and appearance. We refer to it as the *weak identity recovery* (`WIR`) module. A loss function variation is applied to our proposed model. For the training of our identity recovery model, we now include $\mathbf{x}_p$ in training and alter the supervised loss of Eq. (3.17) to $\mathcal{L}^{\texttt{S}}_{\texttt{IR}} = \mathbb{E}_{\mathbf{x}_0,\mathbf{s}_0,\mathbf{x}_p,\mathbf{s}_p}[||\mathbf{x}_0 - \hat{\mathbf{x}}_0||_1 + ||\mathbf{s}_0 - \hat{\mathbf{s}}_0||_1 + ||\mathbf{x}_p - \hat{\mathbf{x}}_p||_1 + ||\mathbf{s}_p - \hat{\mathbf{s}}_p||_1]$. This loss is not possible during training the full model end-to-end, but can be done when training the `IR` model separately. Our goal is to examine how much information we lose by removing the direct supervision on $\mathbf{x}_p$. From Fig. 3.8, we see that L_1 and SSIM do not change much by removing $\mathbf{x}_p$ from supervision.

All models are trained on pairs of RGB-D images corresponding to different views of the same object in Pix3D. During inference, a RGB image from view 1 and a depth map from view 2 are used to predict a RGB image from view 2. Due to the lack of supervision for target RGB images, `HAL` and `WIR` are optimized using the adversarial loss alone. Unsurprisingly, `HAL` has weak performance. It is also clear that the additional disentanglement constraints of `IR` lead to a performance improvement over `WIR`. Fig. 3.7 (*top*) shows examples of the views synthesized by the identity recovery module of DRAW. Note the quality of synthesis across very large view

Table 3.2: Cross-domain view synthesis comparison on Pix3D. For L_1 **lower** is better, for SSIM and inception, **higher** is better.

	[122]	[151]	[116]	**DRAW**
		Pix3D		
L_1	0.16	0.15	0.16	**0.12**
SSIM	0.45	0.46	0.45	**0.51**
		ShapeNet		
L_1	0.24	0.25	0.24	**0.20**
SSIM	0.49	0.46	0.47	**0.56**

angle transformations. Fig. 3.7 (*bottom*) further shows recovery results where two objects of identical shape but different appearance are presented. This example clearly demonstrates that the identity recovery module produces outputs that carry the input images' appearance upon receiving a common $\mathbf{s}_p$ but different $\mathbf{x}_p$.

3.4.2 Comparison to the state-of-the-art

We now evaluate the entire DRAW pipeline on the original task of *pose trajectory learning from sparse image data*. For this experiment we use the Pix3D dataset as well a sub-sampled version of the ShapeNet dataset, where, for each object instance, only ∼5 of its poses are retained in RGB, for training. We compare DRAW with the state-of-the-art novel view synthesis methods [122, 151, 116] in this sparse setting. For evaluation on Pix3D, the pre-trained models from [122, 151, 116] are fine-tuned to the Pix3D training data. For the ShapeNet experiment, these models are trained on the sub-sampled dataset from scratch.

Qualitative Analysis on Pix3D

Qualitative results of all methods are shown in Fig. 3.9. DRAW results for *tables* can be found in Fig. 3.11.

DRAW appears to preserve the appearance and the structure of the object across different poses. Other methods not do not seem to perform very well in comparison. This can be attributed to the severe deficiency of pose training data in Pix3D. Fine-tuning the models of [122, 151, 116] to limited object views leads to poor generalization. Notably, the results of [151] are the worst among all methods. This is perhaps due to the lighting variations and realistic texture in natural images, which results in poor appearance flow mapping compared to synthetic domain. It is also noteworthy that previous methods require view point labels for training. *DRAW doesn't need such supervision.*

Quantitative Analysis on Pix3D

We compiled a test set of valid trajectories from Pix3D, to evaluate the reconstructions of DRAW and it's competitors [122, 151, 116] trained with sparse pose supervision. When one image in the trajectory is used as input, all other images are used as targets to calculate the errors between the related generated view. Results for L_1 and SSIM measures, listed in Table 3.2, indicate that DRAW outperforms all other methods.

Analysis on Sparse ShapeNet

We next compare all methods on a pose-sparse subset of ShapeNet constructed by randomly sampling 5 RGB views per object instance. Models for [122, 151, 116] are trained from scratch on this sub-sampled data. Qualitative images are shown in Fig. 3.10 and quantitative results in Table 3.2 (bottom). It seems clear that in case of sparse pose training, DRAW performs better than other methods even on synthetic ShapeNet.

3.5 Conclusion

We introduce DRAW, a framework to learn pose trajectories from natural image datasets with very few canonical views available per object instance. To make up for such sparsity in the image space, we propose the use cross-modal guidance in the form of texture-less 3D CAD models available for objects. Dense pose trajectories of depth maps can be extracted from these models. Given an RGB image, DRAW operates by (1) mapping it into 2D depth maps, (2) simulating 3D object rotation in depth space and (3) re-mapping the generated views from depth to image space. DRAW can be trained with a set of images with *sparse views*, as in Pix3D. Pose trajectories are synthesized in the synthetic domain and transferred to the image domain in a manner that guarantees consistency of object identity. An identity recovery network that disentangles and recombines appearance and shape information was shown to be important for this purpose. Experiments show that state-of-the-art pose synthesis methods do not perform well if trained with sparse object views in image space. DRAW overcomes this issue by leveraging cross-modal structure priors and generates results of reasonable quality, structural integrity and instance identity.

Chapter 3 is, in full, based on the material as it appears in the publication of “Sparse Pose Trajectory Completion”, Bo Liu, Mandar Dixit, Roland Kwitt, Gang Hua, Nuno Vasconcelos, in *arXiv preprint arXiv:2105.00125*. The dissertation author was the primary investigator and author of this material.

Chapter 4

Few-Shot Open-Set Recognition using Meta-Learning

Table 4.1: Comparison between different recognition tasks.

Typical Recognition Paradigms	No. of Samples per Training Class	Supports unseen Class in Testing?
Closed-set [56]	Large	No
Few-shot [96, 24, 129, 109]	Small	No
Open-set [26, 4]	Large	Yes
Few-Shot Open-set	Small	Yes

4.1 Introduction

The introduction of deep convolutional neural networks (CNNs) has catalyzed large advances in computer vision. Most of these advances can be traced back to advances on object recognition, due to the introduction of large scale datasets, such as ImageNet [19], containing many classes and many examples per class. Many modern computer vision architectures are entirely or partially based on recognition networks. In the large scale setting, CNN-based classifiers trained by cross-entropy loss and mini-batch SGD have excellent recognition performance, achieving state of the art results on most recognition benchmarks.

However, variations of the recognition setting can lead to substantial performance degradation. Well known examples include few-shot learning [96, 24, 129, 109], where only a few training examples are available per class, domain adaptation [126], where a classifier trained on a source image domain (*e.g.* synthetic images) must be deployed to a target domain (*e.g.* natural images) whose statistics differ from those of the source, long tailed recognition [75], where the number of examples per class is highly unbalanced, or problems with label noise [82, 115]. All these *alternative recognition settings* stress the robustness, or ability to generalize, of the large-scale classifier.

More recently, there has been interest in endowing CNNs with self-awareness capabilities. At the core, self-awareness implies that, like humans, CNNs should identify what they can do and refuse what they cannot. Several variants of this problem have been proposed, including out-of-distribution detection [39], where the CNN rejects examples outside the training distribution (*e.g.* images from other domains or adversarial attacks [45]), realistic classification [77], where it rejects examples that it deems too hard to classify, or open-set recognition [4], where the examples to reject are from novel classes, unseen during training. While several techniques have been proposed, a popular approach is to force the CNN to produce high entropy posterior distributions in the rejection regions. Rejections can then be identified as examples that give rise to such distributions.

In this work, we consider open-set recognition. This has been mostly addressed in the large-scale setting, using solutions based on the large-scale classifier. These attempt to recognize the novel classes by post-processing the posterior class distribution [4], defining a "reject" class trained with examples that are either artificially generated [26] or sampled from the training classes [106], or a combination of the two. We seek to generalize open-set recognition beyond the large-scale setting, proposing that it should target a continuum of tasks that includes all the alternative recognition settings. Since training over many settings can be complex, we consider the two at the extremes of this continuum: large-scale and few-shot recognition.

There are three main reasons for this. First, open-set recognition is a challenge under all settings. A recognizer trained in the few-shot regime is not less likely to face unseen classes. An open-set recognition technique that also supports the few-shot setting is thus more useful than the one that does not. Second, few-shot open-set recognition is harder to solve than large-scale open-set recognition, due to the scarcity of labeled data. Hence, the few-shot setting poses a greater challenge to open-set recognition research. Third, like open-set recognition, the main challenge of few-shot recognition is to make accurate decisions for data unseen during training. Since this makes robustness the main trait of few-shot architectures, these architectures are likely to also excel at open-set recognition. Most notably, they are likely to beat the large-scale classifier, which tends not to score highly along the robustness dimension.

The main limitation of the large-scale classifier (trained by cross-entropy) is that it tends to over-fit the training classes. Because the ideal embedding for classification maps all examples of each class into a unique point in feature space, the large-scale classifier tends to learn embeddings that are only locally accurate. While the metric structure of the embedding is reflected by the semantic distances in the neighborhood of class representatives (the parameter vectors of the softmax layer), these distances are meaningless away from the latter. In result, large-scale classifier embeddings under-perform on tasks that require generalization beyond the training set, such as image retrieval [33], face identification [72], pose invariant recognition [40], person re-identification [148], or few-shot learning [129]. Embeddings derived from the metric learning literature [138] or explicitly designed for problems like few-shot recognition [132] tend to capture the metric structure of the data more broadly throughout the feature space and achieve better performance on these generalization critical tasks. It is thus expected that these embeddings will outperform the large-scale classifier on the open set recognition problem.

In this work, we investigate the open-set performance of a class of popular solutions to the few-shot problem, known as meta-learning (ML) [129]. ML approaches replace the traditional mini-batch CNN training by episodic training. Episodes are generated by randomly sampling a subset of the classes and a subset of examples per class, to produce a support and query set to which a classification loss is applied. This randomizes the classification task for which the embedding is optimized at each ML step, producing a more robust embedding, less over-fitted to any particular set of classes. We generalize the idea to open-set recognition, by randomly selecting a set of novel classes per episode, and introducing a loss that maximizes the posterior entropy for examples of these classes. This forces the embedding to better account for unseen classes, producing high-entropy posterior distributions beyond the regions of the target classes.

This solution has at least two strong benefits. First, because it draws on a state of the art approach to few shot learning, it immediately extends open set recognition to few-shot problems. Second, because ML embeddings are robust and open set recognition is about generalization, the performance of the latter improves *even in the large-scale setting*. This is shown by extensive experiments with various open-set recognition benchmarks, where we demonstrate significant improvements over the state of the art. We also investigate the role of the metric used in the feature space on open-set recognition performance, showing that a particular form of the Mahalanobis distance enables significant gains over the commonly used Euclidean distance.

Overall, the contributions of this work can be summarized as follows:

- A new ML-based formulation of open-set recognition. This generalizes open-set to the *few-shot* recognition setting.
- A new episodic training procedure, combining the cross-entropy loss and a novel *open-set loss* to improve open-set performance on both the large-scale and few-shot settings.
- A *Gaussian embedding* for ML-based open-set recognition.

4.2 Related Work

Open-Set Recognition

Open-set recognition addresses the classification setting where inference can face samples from classes unseen during training. The goal is to endow the open-set classifier with a mechanism to reject such samples. One of the first deep learning approaches was the work of Scheirer *et al.* [4], which proposed an extreme value parameter redistribution method for the logits generated by the classifier. Later works considered the problem in either discriminative or generative models. Schlachter *et al.* [106] proposed an intra-class splitting method, where a closed-set classifier is used to split data into typical and atypical subsets, reformulating open-set recognition as a traditional classification problem. G-OpenMax [26] utilizes a generator trained to synthesize examples from an extra class that represents all unknown classes. Neal *et al.* [83] introduced counterfactual image generation, which aims to generate samples that cannot be classified into any of the seen classes, producing an extra class for classifier training.

All these methods reduce the set of unseen classes to one extra class. While open samples can be drawn from different categories and have significant visual differences, they assume that a feature extractor can map them all into a single feature space cluster. While theoretically possible, this is difficult. Instead, we allow a cluster per seen class and label samples that do not fall into these clusters as unseen. We believe this is a more natural way to detect unseen classes.

Out-of-Distribution

A similar problem to open-set recognition is to detect out-of-distribution (OOD) examples. Typically [39], this is formulated as the detection of samples from a different dataset, *i.e.* a different distribution from that used to train the model. While [39] solves this problem by directly using the softmax score, later works [67, 130] improve results by enhancing reliability. This problem differs from open-set recognition in that the OOD samples are not necessarily from unseen classes. For examples, they could be perturbed versions of samples from seen classes, as is common in the adversarial attack literature. The only constraint is that they do not belong to the training distribution. Frequently, these samples are easier to detect than samples from unseen classes. In the literature, they tend to be classes from other datasets or even images of noise. This is unlike open-set recognition, where unseen classes tend to come from the same dataset.

Few-Shot Learning

Extensive research on few-shot learning [96, 24, 129, 109, 118, 97, 29, 87, 65, 18] has emerged in recent years. These methods can be broadly divided into two branches: optimization and metric based. Optimization based methods deal with the generalization problem by unrolling the back-propagation procedure. Specifically, Ravi *et al.* [96] proposed a learner module that is trained to update to novel tasks. MAML [24] and its variants [25] proposed a training procedure, where the parameters are updated based on the loss calculated by secondary gradients. Metric based approaches attempt to compare feature similarity between support and query samples.

Vinyals *et al.* [129] introduced the concept of episode training, where the training procedure is designed to mimic the test phase, which was used with a cosine distance to train recurrent networks. The prototypical network [109] introduced class prototypes constructed from support set features, by combining metric learning and cross-entropy loss. The relation network [118] explored the relationship between a pair of support and query features implicitly with a neural network, instead of building a metric directly on feature space.

Since a feature space metric is also useful for open-set recognition, we mainly focus on metric based approaches to few-shot learning. Although several methods have demonstrated promising results for the few-shot task, it is still unclear whether the resulting classifiers can successfully reject unseen samples. In this work, we explore this problem under both the large-scale and few-shot scenarios.

Learning without Forgetting

Several works [29, 92, 93, 100] extend traditional few-shot to learning without forgetting. The model is trained to work on an extra few-shot problem and maintains its performance on the original recognition problem. Our work will focus on the traditional few-shot problem, this direction is beyond our scope, and can be discussed in the future.

4.3 Classification tasks and meta Learning

In this section, we discuss different classification settings (Sec. 4.3.1) and meta learning (Sect. 4.3.2), which motivates and provides the foundation for the proposed solution to open-set recognition in both the large-scale and few-shot settings.

4.3.1 Classification settings

Softmax classification

The most popular deep learning architecture for object recognition and image classification is the softmax classifier. This consists of an embedding that maps images $\mathbf{x} \in \mathcal{X}$ into feature vectors $f_\phi(\mathbf{x}) \in \mathcal{F}$, implemented by multiple neural network layers, and a softmax layer with linear mapping that estimates class posterior probabilities with

$$p(y = k|\mathbf{x};\phi,\mathbf{w}_k) = \frac{\exp(\mathbf{w}_k^T f_\phi(\mathbf{x}))}{\sum_{k'} \exp(\mathbf{w}_{k'}^T f_\phi(\mathbf{x}))} \tag{4.1}$$

where ϕ denotes all the embedding parameters and $\mathbf{w}_k$ is a set of classifier weight vectors. Recent works [109] have combined metric learning with softmax classification, implementing the softmax layer with a distance function

$$p_\phi(y = k|\mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}),\mu_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}),\mu_{k'}))} \tag{4.2}$$

where $\mu_k = E[f_\phi(\mathbf{x})|y = k]$ [109]. The softmax classifier is learned with a training set $\mathbb{S} = \{(x_i^s, y_i^s)\}_{i=1}^{n^s}$, where $y_i^s \in \mathbb{C}^s, \forall i$, $\mathbb{C}^s$ is a set of training image classes, and n^s the number of training examples. This consists of finding the classifier and embedding parameters that minimize the cross entropy loss

$$\mathcal{L}_{CE} = \sum_{(x_i^s, y_i^s)} -\log p(y_i^s|\mathbf{x}_i^s) \tag{4.3}$$

Recognition performance is evaluated on a test set $\mathbb{T} = \{(x_i^t, y_i^t)\}_{i=1}^{n^t}$, where $y_i^t \in \mathbb{C}^t, \forall i$, $\mathbb{C}^t$ is a set of test classes, and n^s the number of test examples.

Closed- *vs.* **Open-set Classification**

Under the traditional definition of classification, the sets of training and test classes are identical, i.e. $\mathbb{C}^s = \mathbb{C}^t$. This is denoted as *closed set classification*. Recently, there has been interest in an alternative *open set* classification setting, where $\mathbb{C}^t = \mathbb{C}^s \cup \mathbb{C}^u$. In this case, the classes in $\mathbb{C}^s$ are denoted as *seen* (during training) classes and the classes in $\mathbb{C}^u$ as *unseen*.

Large-scale *vs.* **Few-shot Recognition**

In the large-scale recognition setting, the number n^s of training examples is quite high, reaching the millions for datasets like ImageNet. On the contrary, for few-shot recognition, this number is quite low, usually less than twenty examples per class. A few-shot problem with K training examples per class is commonly known as K-shot recognition. Note that the number n^t of test examples plays no role in differentiating these two settings. Since these examples are only used for performance evaluation, the test set should have the same cardinality under the two settings. As shown in Table 4.1, different combinations of properties in the scale and coverage of the training data define different classification paradigms. The few-shot open-set setting is largely unexplored in the literature and the focus of this paper.

4.3.2 Meta-learning

Meta-learning (ML) addresses the problem of "learning to learn". In this case, a meta-learner learns a learning algorithm by inspection of many learning problems. For this, the meta-learner relies on a meta training set $\mathbb{MS} = \{(\mathbb{S}^s_i, \mathbb{T}^s_i)\}_{i=1}^{N^s}$, where $(\mathbb{S}^s_i, \mathbb{T}^s_i)$ are the training and test set of the i^{th} learning problem and N^s the number of learning problems used for training; and a meta test set $\mathbb{MT} = \{(\mathbb{S}^t_i, \mathbb{T}^t_i)\}_{i=1}^{N^t}$, where $(\mathbb{S}^t_i, \mathbb{T}^t_i)$ are the training and test set of the i^{th} test learning problem and N^t is the number of learning problems used for testing. Given $\mathbb{MS}$, the meta-learner learns how to map a pair $(\mathbb{S}, \mathbb{T})$ into an algorithm that leverages $\mathbb{S}$ to optimally solve

$\mathbb{T}$.

The procedures are as follows. At meta iteration i, a meta-model h is initialized with the one produced by the previous meta-iteration. Two steps are then performed. First, the meta-learning algorithm performs the mapping

$$h' = \mathcal{M}(h, \mathbb{S}_i^s) \tag{4.4}$$

to produce an estimate h' of the optimal model for training set $\mathbb{S}_i^s$. The test set $\mathbb{T}_i^s$ is then used to find the model

$$h^* = \arg\min_h \sum_{(x_k, y_k) \in \mathbb{T}_i^s} L[y_k, h'(x_k)] \tag{4.5}$$

for a suitable loss function L, *e.g.* cross-entropy, using a suitable optimization procedure, *e.g.* back-propagation. The resulting h^* is finally returned as the optimal meta-model for meta-iteration i. During testing, the final meta-model h^* and a training set $\mathbb{S}_i^t$ from the meta test set $\mathbb{MT}$ are used by the meta-learner to produce a new model

$$h'' = \mathcal{M}(h^*, \mathbb{S}_i^t) \tag{4.6}$$

whose performance is evaluated with $\mathbb{T}_i^t$.

ML for Few-shot Recognition

While different approaches have been proposed to apply ML to few shot recognition, in this work we adopt the procedure introduced by the popular prototypical network architecture [109], which is the foundation of various other approaches. In this context, ML is mostly a randomization procedure. The pairs $(\mathbb{S}_i, \mathbb{T}_i)$ are denoted *episodes*, the training sets $\mathbb{S}_i$ are denoted *support sets* and the test sets $\mathbb{T}_i$ *query sets*. The meta-training set $\mathbb{MS}$ is generated by sampling train and test classes. In particular, the training set $\mathbb{S}_i^s$ of the i^{th} episode is obtained by sampling N classes from the set of classes of the low-shot problem, and K-examples per class. This defines a set of K-shot learning problems and is known as the *N-way K shot problem*. The model h is the softmax classifier of (4.2), the meta-learning mapping of (4.4) implements the Gaussian mean maximum likelihood estimator

$$\mu_k = \frac{1}{|\mathcal{P}_{i,k}|} \sum_{\mathbf{x}_j \in \mathcal{P}_{i,k}} f_\phi(\mathbf{x}_j) \tag{4.7}$$

where $\mathcal{P}_{i,k} = \{\mathbf{x}_j \in \mathbb{S}_i^s | y_j = k)$ is the set of support samples from class k, and back-propagation is used in (4.5) to update the embedding f_ϕ.

Benefits of ML for Open-set Classification

The ML procedure is quite similar to the standard mini-batch procedure for training softmax classifiers. Aside from the use of episode training instead of the more popular mini-batch training, there are two main differences. First, f_ϕ is updated by back-propagation from examples in $\mathbb{T}_i^s$ rather than $\mathbb{S}_i^s$. This is advantageous for few-shot learning due to the consistency with meta-testing, which is not the case in the large-scale regime. Second, mini-batches of random examples from all classes are replaced by examples from the subset of N classes contained in $\mathbb{S}_i^s$ and $\mathbb{T}_i^s$. Randomizing the classification task learned per episode forces the embedding f_ϕ to generalize better to unseen data. This property makes ML a good solution for few-shot learning, where the training data lacks a good coverage of intra-class variations. In this case, large subsets of test samples from the known classes are unseen during training. We propose that the same property makes ML a good candidate for open-set classification, where by definition the classifier must deal with unseen samples from unseen classes. This observation motivates us to design a unified ML-based solution to open-set classification that supports both large-scale and few-shot classification.

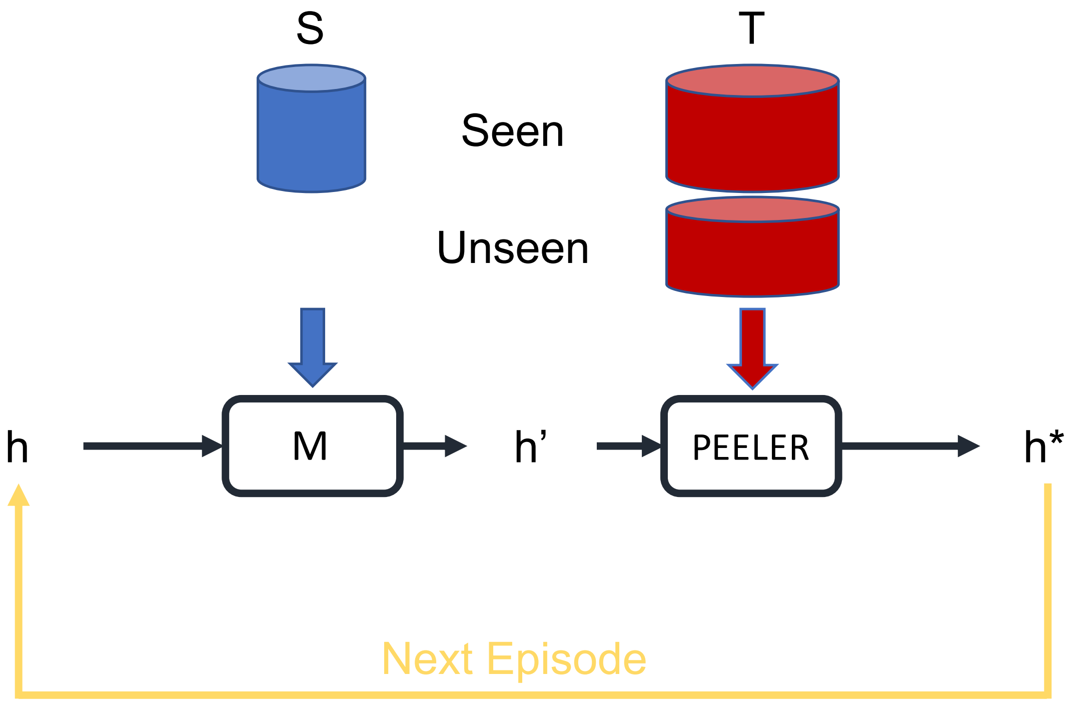


Figure 4.1: The proposed general framework for open-set meta-learning: a meta training set $\{\mathbb{S}, \mathbb{T}\}$ is sampled, where $\mathbb{T}$ contains classes not in $\mathbb{S}$ as "unseen" classes, and loss in (4.8) is minimized to obtain h^*.

4.4 Meta-learning-based Open-set Recognition

In this section, we introduce the proposed ML approach to open-set recognition. We first introduce the general procedure and then discuss the specific embedding metric used in our PEELER implementation.

4.4.1 Open-Set Meta-Learning

As shown in Figure 4.1, open-set meta-learning (PEELER) relies on a meta training set $\mathbb{MS} = \{(\mathbb{S}_i^s, \mathbb{T}_i^s)\}_{i=1}^{N^s}$, and a meta test set $\mathbb{MT} = \{(\mathbb{S}_i^t, \mathbb{T}_i^t)\}_{i=1}^{N^t}$. The only difference with respect to standard ML is that the episodes $(\mathbb{S}, \mathbb{T})$ are open set. While the training set $\mathbb{S}$ is identical to the one used in standard ML, the test set $\mathbb{T}$ is *augmented with unseen classes*. Hence, PEELER can be addressed with a solution similar to the ML procedure of Section 4.3.2. While the ML step remains as in (4.4), the optimization step of (4.5) becomes

$$h^* = \arg\min_h \{ \sum_{(x_k,y_k)\in\mathbb{T}_i^s | y_k \in \mathbb{C}_i^s} L_c[y_k, h'(x_k)] + \lambda \sum_{(x_k,y_k)\in\mathbb{T}_i^s | y_k \in \mathbb{C}_i^u} L_o[h'(x_k)] \} \tag{4.8}$$

where $\mathbb{C}_i^s$ ($\mathbb{C}_i^u$) is the set of seen (unseen) classes of $\mathbb{T}_i^s$, $L_c[.,.]$ is the classification loss to apply to the seen classes (typically the cross-entropy loss) and L_o is an open-set loss applied to those unseen in $\mathbb{S}_i^s$.

Few-shot Open-set Recognition

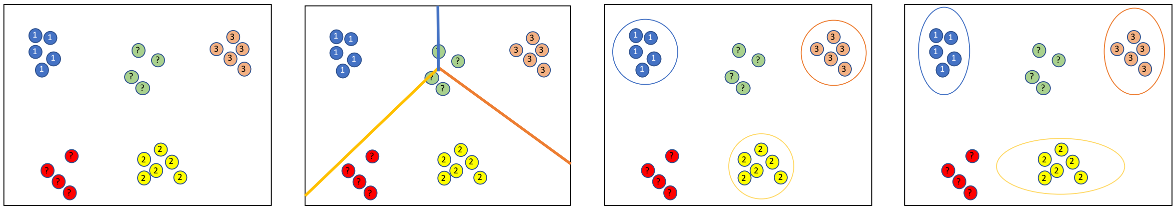
The few-shot setting requires the sampling of the classes that make up the support and query sets. Similarly to closed-set few-shot recognition, the support set $\mathbb{S}_i^s$ of the i^{th} episode is obtained by sampling N classes and K-examples per class. This defines the seen classes $\mathbb{C}_i^s$. However, the query set $\mathbb{T}_i^s$ is composed by the combination of these classes with M additional unseen classes $\mathbb{C}_i^u$. These support and query sets are used in (4.8).

Large-scale Open-set Recognition

In the large-scale setting, the seen classes have plenty of examples and are well trained without any need for re-sampling. However, re-sampling of the unseen classes is still advantageous, since it enables an embedding that generalizes better to these classes. In each episode, M classes are randomly sampled from the class label space to form the set of unseen classes $\mathbb{C}_i^u$, and the remaining classes are used as seen classes $\mathbb{C}_i^s$. Without a support set $\mathbb{S}_i^s$, the meta-learning step of (4.4) is no longer needed. Instead, we rely on the set of seen classes to adjust the model to only classify samples into those classes, *i.e.* a mapping

$$h' = \mathcal{M}(h, \mathbb{C}_i^s), \tag{4.9}$$

with the loss function of (4.8) is still applied.



(a) (b) (c) (d)

Figure 4.2: (a) Feature space of an open-set recognition problem with 3 seen and 2 unseen classes; (b) Optimal boundaries for closed-set classification; (c) Each class defines an euclidean distance of equal radius. While optimal for closed-set, this is sub-optimal for open set recognition; (d) a better set of open-set distances can be learned by allowing different Gaussian clusters of class-dependent variance along each dimension.

Open-set Loss

During inference, when faced with samples from unseen classes, the model should not assign a large probability to any class. In this case, a sample can be rejected if the largest class probability $\max_k p_\phi(y=k|\mathbf{x})$ among seen classes is small. To enable this, the learning algorithm should minimize the probabilities on seen classes for samples from $\mathbb{C}_i^u$. This can be implemented by maximizing the entropy of seen class probabilities, i.e using the negative entropy

$$L_o[\mathbf{x}] = \sum_{k \in \mathbb{C}_i^s} p(y=k|\mathbf{x}) \log p(y=k|\mathbf{x}) \tag{4.10}$$

as loss function.

4.4.2 Gaussian Embedding

While PEELER is a general framework, in this work we propose an implementation based on the prototypical network architecture [109]. A set of class prototypes is first defined and samples with a large distance to the closest class prototype are assigned to the set of unseen classes. For low-shot classification, the class prototypes are the class means of (4.7). For large-scale classification, we assume fixed prototypes that are embedded in the network and learned by back-propagation.

Although several distances are discussed in [109], the prototypical network is usually implemented with the Euclidean distance, *i.e.*,

$$d(f_\phi(\mathbf{x}),\mu_k) = (f_\phi(\mathbf{x}) - \mu_k)^T (f_\phi(\mathbf{x}) - \mu_k). \tag{4.11}$$

This implies that the features from each class follow a Gaussian distribution with mean $\mathbf{c}_k$ and diagonal covariance $\sigma^2 I$, where σ is shared by all classes. While sensible for closed-set few-shot learning, where the embedding is learned to produced such feature distributions, this can be very sub-optimal when open-set samples are introduced. Figure 4.2 illustrates the problem for a setting with three seen classes and two unseen classes. Even though, as shown in Figure 4.2 (a), the embedding is such that seen classes are spherically distributed and have the same covariance, the open-set samples, unseen during training, are still embedded onto the feature space at random locations. Hence, although the optimal boundaries of the closed-set classifier, shown in Figure 4.2 (b), match the contours of the distance of (4.11), shown in Figure 4.2 (c), they are not optimal for open set recognition. In fact, as shown in Figure 4.2 (d), the shape of the optimal boundary between seen and unseen classes can even vary from one class to another.

To account for this, we assume a Gaussian distribution of mean μ_k and covariance Σ_k for class k. The Euclidean distance of (4.11) is thus replaced by the Mahalanobis distance

$$d(f_\phi(\mathbf{x}),\mu_k) = [f_\phi(\mathbf{x}) - \mu_k]^T \Sigma_k^{-1} [f_\phi(\mathbf{x}) - \mu_k]. \tag{4.12}$$

To keep the number of parameters manageable, we assume that all covariance matrices are diagonal, *i.e.* $\Sigma_k = \text{diag}(\sigma_{k1}, \sigma_{k2}, \ldots, \sigma_{kM})$, and the precision matrix $A_k = \Sigma_k^{-1}$ is used to ease calculations. Similar to the class prototypes, the learning of precision matrices depends on the recognition setting. For large-scale open-set recognition, the A_k are network parameters learned directly by back-propagation. In the few-shot setting, where support samples are available, we introduce a new embedding function g_φ with learnable parameters φ, and define

$$A_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} g_\varphi(\mathbf{x}_i). \tag{4.13}$$

4.5 Experiments

The proposed method was compared to state-of-the-art (SOTA) approached to open-set recognition. Following Neal *et al.* [83], we evaluate both classification accuracy and open-set detection performance. To clarify the terminology, closed-set classes are the categories seen during training and open-set classes the novel categories only used for testing. All training is based on the training samples from closed-set classes. For testing, we use the testing samples from both the training and open-set classes. Classification *accuracy* is used to measure how well the model classifies closed-set samples, i.e., test samples from closed-set classes. The *AUROC* (Area Under ROC Curve) metric is used to measure how well the model detects open-set samples, i.e., test samples from open-set classes, within all test samples. To simplify the writing, we define the following acronyms: *our basic* represents prototypical networks with euclidean distance for open-set detection; *GaussianE* represents the Gaussian Embedding introduced in Sec. 4.4.2; *OpLoss* represents the proposed open-set loss.

4.5.1 Large-Scale Open-Set Recognition

Most prior works on open-set recognition are designed for the large-scale setting. Here we evaluate PEELER on CIFAR10 [56] and extended miniImageNet [29]. CIFAR10 consists of $60,000$ images of 10 classes. Following [83], 6 classes are first randomly selected as closed-set classes, and the other 4 kept as open-set classes. Results are averaged over 5 random partitions of closed/open-set classes. Extended miniImageNet is designed for few-shot learning. We use the 64 training categories and 600 images per category as closed set training data, while the 300

Table 4.2: Comparison to SOTAs on large-scale open-set recognition: PEELER outperforms all others in terms of open-set sample detection *AUROC*, for a comparable classification accuracy.

Model	Accuracy(%)	AUROC(%)
ConvNet on CIFAR10		
Softmax	80.1	67.7
OpenMax	80.1	69.5
G-OpenMax	81.6	67.9
Counter	82.1	69.9
Confidence	82.4	73.19
Our basic	82.4	74.62
Our basic + GaussianE	82.3	75.65
Our basic + GaussianE + OpLoss	82.3	**77.22**
ResNet-10 on CIFAR10		
Softmax	94.2	78.90
OpenMax	94.2	79.02
Confidence	94.0	80.90
Our basic	94.7	82.94
Our basic + GaussianE	94.3	83.12
Our basic + GaussianE + OpLoss	94.4	**83.99**
ResNet-10 on miniImageNet		
Softmax	76.1	76.65
OpenMax	76.1	77.80
Confidence	76.5	80.67
Our basic	76.4	80.59
Our basic + GaussianE	76.1	81.06
Our basic + GaussianE + OpLoss	76.3	**82.12**

extra images of each category are used for closed set testing. Images from 20 test categories are used for open set testing.

Training

On CIFAR10, we randomly sample 2 from the 6 closed-set classes per episode to apply the open-set loss, the remaining 4 classes are used to train classification. On miniImageNet, the closed/open-set partition is fixed. We use Adam [52] with an initial learning rate of 0.001, $\lambda = 0.5$ in (4.8) and $10,000$ training episodes. The learning rate is decayed by a factor of 0.1 after $6,000$ and $8,000$ episodes.

Results

We compare the proposed method to several open-set recognition SOTA methods, including OpenMax [4], G-OpenMax [26], and Counterfactual [83], and an out-of-distribution SOTA method, Confidence [64]. All models use the same CNN backbone for fair comparison. We tested both a CNN, denoted as ConvNet, proposed by [83], and ResNet-10 [38].

Table 4.2 shows that, for both backbones, PEELER outperforms all previous approaches by a significant margin. For a similar classification accuracy on seen classes, it achieves much higher AUROCs for the detection of unseen classes. The open-set detection performance is enhanced by both the proposed Gaussian embedding and open-set loss.

4.5.2 Few-Shot Open-Set Recognition

Dataset

Few-Shot Open-Set performance is evaluated on mini-Imagenet [129], using the splits of [96]. 64 classes are used for training, 16 for validation and another 20 for test.

Training

The open-set problem follows the 5-way few-shot recognition setting. During training, 10 classes are randomly selected from the training set per episode, 5 are used as closed-set classes and the other 5 as unseen. All support set samples are from the closed-set classes. The query set contains samples from closed-set classes, the closed-query set, and samples from open-set classes, the open-query set. The evaluation set is sampled from the test set with the same strategy. The evaluation is repeated 600 times to minimize uncertainty. The total number of training episodes is $30,000$. The learning rate is decreased by a factor of 0.1 after $10,000$ and $20,000$ episodes.

Table 4.3: Few-shot open-set recognition results. Comparison to several baselines and prior open-set methods.

Model	Accuracy(%)	AUROC(%)
5-way 1-shot		
GaussianE + OpenMax	57.89±0.59	58.92±0.59
GaussianE + Counterfactual	57.89±0.59	52.20±0.61
Our basic	56.31±0.57	58.94±0.60
Our basic + OpLoss	56.34±0.57	60.94±0.61
Our basic + GaussianE	57.89±0.59	58.66±0.60
Our basic + GaussianE + OpLoss	**58.31±0.58**	**61.66±0.62**
5-way 5-shot		
GaussianE + OpenMax	75.31±0.76	67.54±0.67
GaussianE + Counterfactual	75.31±0.76	53.25±0.59
Our basic	74.19±0.75	66.00±0.67
Our basic + OpLoss	74.14±0.74	67.92±0.68
Our basic + GaussianE	**75.31±0.76**	66.50±0.67
Our basic + GaussianE + OpLoss	75.08±0.72	**69.85±0.70**

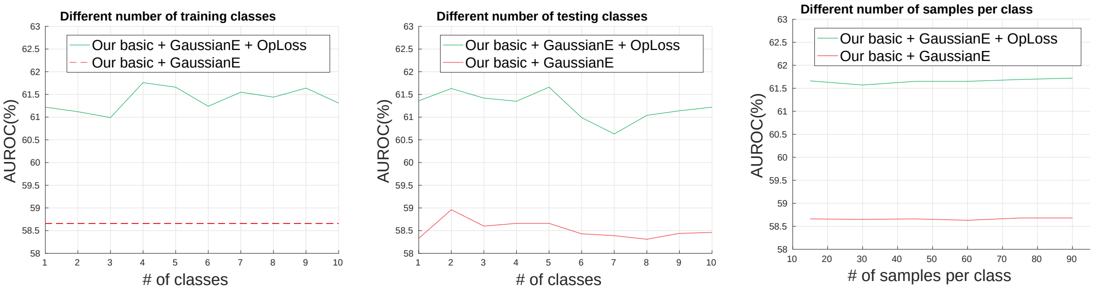
Table 4.4: 10-way 1-shot openset recognition results. The AUROC is higher than those in 5-way.

Model	Accuracy(%)	AUROC(%)
Our basic	39.61±0.40	71.32±0.70
Our basic + OpLoss	39.41±0.40	72.23±0.72
Our basic + GaussianE	40.18±0.40	71.31±0.70
Our basic + GaussianE + OpLoss	**41.90±0.39**	**74.97±0.74**

Results

Since not all prior open-set methods support the few-shot setting, some modifications were required. For example, generative methods [26, 83] do not support few-shot samples. Instead, we train the model on the pre-training set and fine-tune it on the support set. The closed-set classifier is as above, and a two-class classifier is further trained to detect open-set samples. OpenMax [4] is easier to apply under the few-shot setting. We apply OpenMax on the activations of the pre-softmax layer, *i.e.* the negative of the distance of our Gaussian setting. All methods are implemented with the ResNet-10 [38] for fair comparison.

As shown in Table 4.3, both OpenMax and Counterfactual have weaker performance than



(a) (b) (c)

Figure 4.3: Ablation Study: the proposed open-set loss produces consistent gain with different (a) number of training classes, (b) number of testing classes, or (c) number of testing samples per class.

the proposed approach. Note that a *AUROC* of 50% corresponds to chance performance. The proposed open-set loss, Gaussian embedding, and their combination all provide gains.

4.5.3 Ablation Studies

Sampling Strategy in Training

We study how the training sampling strategy affects open-set detection results. The number of unseen classes used to produce open-set samples by training episode is varied, for a fixed total number of open-set samples. The corresponding open-set detection results are shown in Figure 4.3 (a). The performance variation is minor, when compared to the average gain over the baseline method. We hypothesize that this robustness is due to task randomness. When the total number of training episodes is large, the model converges well no matter how many open-set classes are included in a single episode.

Factors in Open-set Testing

For large-scale open-set recognition, the number of open-set samples follows from the number of open-set classes. The difficulty of the open-set problem is determined by the latter. We try to discover factors that determine the difficulty of the few-shot open-set problem. A first

Table 4.5: The number of mis-classified frames (lower is better) by varying the number of annotated frames (I, II, III in the second row).

Category	VODC [131]			Ours		
	I	II	III	I	II	III
airplane	20	10	**0**	6	6	**0**
balloon	13	4	3	11	7	**2**
bear	3	3	**2**	**2**	**2**	**2**
cat	4	5	5	5	**3**	**3**
eagle	23	12	8	5	2	**0**
ferrari	11	7	6	11	**3**	**3**
figure skating	**0**	**0**	**0**	**0**	**0**	**0**
horse	5	**1**	**1**	5	4	3
parachute	14	10	2	10	2	**0**
single diving	18	13	5	4	2	**1**
Avg.	11.1	6.5	3.2	5.9	3.1	**1.4**

factor is the number of classes in the open-query set. We vary this number from 1 to 10, while keeping the training procedure and the total number of samples in the open-query set unchanged. Results are shown in Figure 4.3 (b). A second factor is the number of samples per class, which is changed while the number of classes remains 5. Results are shown in Figure 4.3 (c). The figures show that variations cause small changes in AUROC performance. This means that the factors have little impact in the difficulty of the problem.

Factors in Few-shot Testing

We compare 5-way to 10-way classification, for a fixed open-set component of both training and testing sets. Results are shown in Table 4.4. Although, as expected, 10-way underperforms 5-way classification, the open-set performance improves significantly. This shows that few-shot classification tasks with more class diversity are more robust to open-set samples.

4.5.4 Weakly Supervised Object Discovery

Finally, we investigate an application of few-shot open-set recognition. A weakly supervised object discovery problem is considered in [131]. A video of an object is given, but some frames are irrelevant for the presence of the object. The task is to find these irrelevant frames when the number of annotated frames (object presented or not) is limited. For the open-set problem, only relevant frame labels are given, *i.e.* frames containing the object. The irrelevant frames are detected as open-set samples.

The XJTU-Stevens Dataset [131] is adopted for evaluation. It has 101 videos from 10 categories with different instances, and some frames in those videos do not have the object. During training, a 5-way 1-shot few-shot open-set model is trained as described in Sec. 4.5.2. Instead of category level classification, we perform instance level classification of each frame. During testing, only one video is considered, using the frames annotated as relevant as the support set. This implies that only one Gaussian class center is provided. The unlabeled frames are labeled as seen or unseen by their Mahalanobis distance to the center. Frames with distance higher than a threshold are detected as irrelevant. The number of annotated relevant frames varies from 1 to 3. Results are shown as the number of mis-classified frames. The best VODC method in [131] is listed for comparison. Note that VODC requires the same number of annotated irrelevant frames, which PEELER does not. The proposed method largely outperforms VODC, halving the number of mis-classified frames in all three settings. This shows that its feature embedding is a better solution for open-set detection.

4.6 Conclusion

In this work, we have revisited the open-set recognition problem in the context of few-shot learning. We proposed an extension of meta-learning that includes an open-set loss, and a better metric learning design. The resulting classifiers provide a new stat-of-the-art for few-shot open-set recognition, on mini-Imagenet. It was shown that, with few modifications, the approach can also be applied to large-scale recognition, where it outperforms state of the art methods for open-set recognition. Finally, the XJTU-Stevens Dataset was used to demonstrate the effectiveness of the proposed model on a weakly supervised object discovery task.

Acknowledgement. Gang Hua was supported partly by National Key R&D Program of China Grant 2018AAA0101400 and NSFC Grant 61629301. Bo Liu and Nuno Vasconcelos were partially supported by NSF awards IIS-1637941, IIS-1924937, and NVIDIA GPU donations.

Chapter 4 is, in full, based on the material as it appears in the publication of “Few-Shot Open-Set Recognition using Meta-Learning”, Bo Liu, Hao Kang, Haoxiang Li, Gang Hua, Nuno Vasconcelos, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2020. The dissertation author was the primary investigator and author of this material.

Chapter 5

GistNet: a Geometric Structure Transfer Network for Long-Tailed Recognition

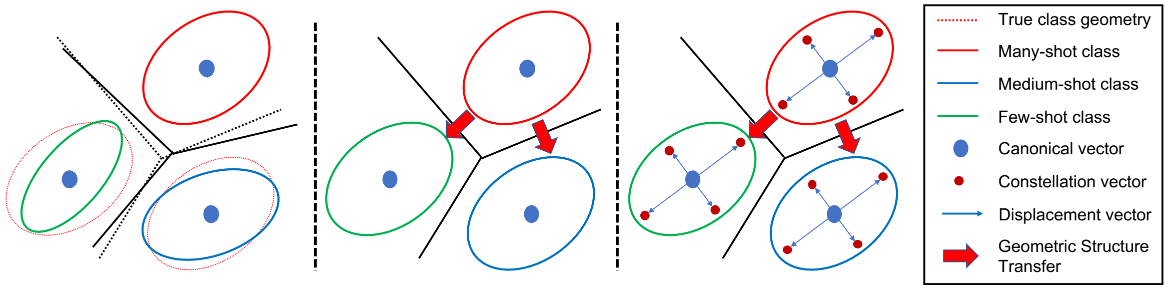


Figure 5.1: Left: in long-tailed recognition, the small number of samples from medium- and few-shot classes make it difficult to learn their geometry, leading to inaccurate class boundaries. This is unlike many-shot classes, whose natural geometry can usually be learned. Middle: the boundaries are corrected by transferring the geometric structure of the many-shot classes to the classes with few examples. Right: GistNet implements geometric structure transfer by implementing constellations of classification parameters. These consist of a class-specific center and a set of displacements shared by all classes. Under GIST training, these tend to follow the natural geometry of the many-shot classes, which is transferred to the medium- and few-shot classes.

5.1 Introduction

The availability of large-scale datasets, with large numbers of images per class [19], has been a major factor in the success of deep learning for tasks such as object recognition. However, these datasets are manually curated and artificially balanced. This is unlike most real world applications, where the frequencies of examples from different classes can be highly unbalanced, leading to skewed distributions with long tails.

This has motivated recent interest in the problem of long-tailed recognition [75], where the training data is highly unbalanced but the test set is kept balanced, so that equally good performance on all classes is crucial to achieve high overall accuracy.

Success in the long-tailed recognition setting requires specific handling of class imbalance during training, since a classifier trained with the standard cross-entropy loss will over-fit to highly populated classes and perform poorly on low-shot classes. This has motivated several works to fight class over-fitting with methods, like data re-sampling [145] or cost-sensitive losses [68], that place more training emphasis on the examples of lower populated classes.

It is, however, difficult to design augmentation or class weighting schemes that do not either under or over-emphasize the few-shot classes. In this work, we seek an approach that is fully data driven and *leverages* over-fitting to the popular classes, rather than combat it. The idea is to transfer some properties of these classes, which are well learned by the standard classifier, to the classes with insufficient data, where this is not possible.

For this, we leverage the interpretation of a deep classifier as the composition of an embedding, or feature extractor, implemented with several neural network layers and a parametric classifier, implemented with a logistic regression layer, at the top of the network. While the embedding is shared by all classes, the classifier parameters are class-specific, namely a weight-vector per class, as shown in Figure 5.1.

We exploit the fact that the configuration of these weight vectors determines the geometry of the embedding. This consists of the class-conditional distribution, and associated metric, of the feature vectors of each class, which define the class boundaries. For a well learned network, this geometry is identical for all classes. In the long-tailed setting, the geometry is usually well learned for many-shot classes, but not for classes with insufficient training samples, as shown in the left of Figure 5.1.

The goal is to transfer the geometric structure of the many-shot classes to the classes with few examples, as shown in the middle of the figure, to eliminate this problem. The challenge is to implement this transfer using only the available training data, *i.e.* without manual specification of class-weights or heuristic recipes, such as equating these weights to class frequency.

We address this challenge with a combination of contributions. First, we enforce a globally learned geometric structure, which is shared by all classes. To avoid the complexity of learning a full-blown distance function, which frequently requires a large covariance matrix, we propose a structure composed by a constellation of classifier parameters, as shown on the right of Figure 5.1. This consists of a class-specific center, which encodes the location of the class, and a set of displacements, which are shared by all classes and encode the class geometry.

Second, we rely on a mix of randomly sampled and class-balanced mini-batches to define two losses that are used to learn the different classifier parameters. Class-balanced sampling is used to learn the class-specific center parameters. This guarantees that the learning is based on the same number of examples for all classes, avoiding a bias towards larger classes. Random sampling is used to learn the shared geometry parameters (displacements). This leverages the tendency of the standard classifier to over-fit to the popular classes, making them dominant for the learning of class geometry, and thus allowing the transfer of geometric structure from these to the few-shot classes. In result, the few shot classes are learned equally to the popular classes with respect to location but inherit their geometric structure, which enables better generalization.

We propose a new learning algorithm, denoted *GeometrIc Structure Transfer* (GIST), that combines the two types of sampling, so as to naturally account for all the data in the training set, without the need for the manual specification of class weights, or even the explicit grouping of classes into different types. While we adopt the standard division into many-, medium-, and few-shot classes for evaluation, this is not necessary for training.

A deep network that implements the parameter constellations of Figure 5.1 and GIST training is then introduced and denoted as the GistNet. Experiments on two popular long-tailed recognition datasets show that it outperforms previous approaches to long-tailed recognition.

Overall, this work makes several contributions to long-tailed recognition. First, we point out that the tendency of the standard classifier to over-fit to popular classes can be advantageous for transfer learning. The goal should not be to eliminate this over-fitting, *e.g.* by uniquely adopting the now popular class-balanced sampling, but leverage it to transfer geometric information from the popular to the low-shot classes.

Second, we propose a new GistNet classifier architecture to support this goal, using constellations of classifier parameters to encode the class geometry.

Third, we introduce a new learning algorithm, GIST, that combines class-balanced and random sampling to leverage over-fitting to the popular classes and enable the transfer of class geometry from popular to few-shot classes.

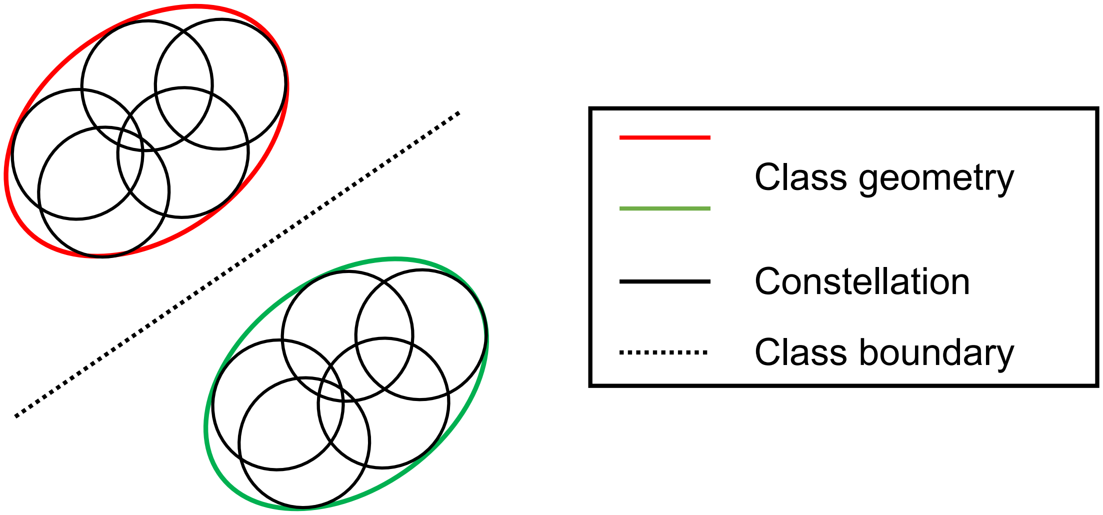


Figure 5.2: GistNet approximates the shared geometry by a constellation (mixture) of spherical Gaussians.

5.2 Related Work

Long-tailed Recognition

Long-tailed recognition has received increased attention in the recent past [134, 86, 68, 145, 75, 133]. Several approaches have been proposed, including metric learning [86, 145], hard negative mining [68], or meta-learning [133]. Some of these rely on novel loss functions, such as the lift loss [86], which introduces margins between many training samples, the range loss [145], which encourages data in the same class (different classes) to be close (far away), or the focal loss [68], which conducts online hard negative mining. These methods tend to improve performance on the few-shot end at the cost of many-shot accuracy.

Other methods, *e.g.* class-balanced experts [107] and knowledge distill [141], try to avoid this problem by manually dividing the training data into subsets, based on the number of examples, and training an expert per subset. However, experts learned from arbitrary data divisions can be sub-optimal, especially for few-shot classes.

Kang *et al.* [50] tackles the data-imbalance problem by decoupling the training feature embedding and classifier. Zhou *et al.* [150] also shows the effectiveness by using different training

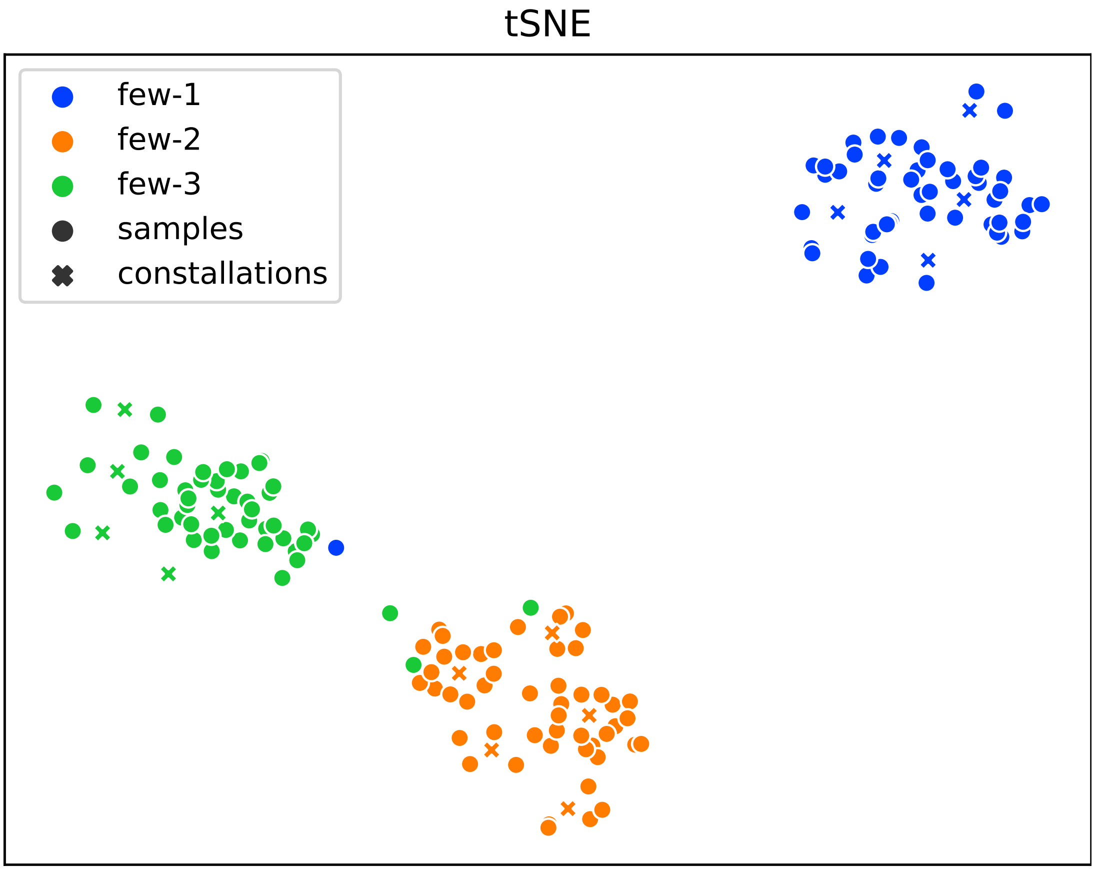


Figure 5.3: t-SNE visualization of 3 few-shot classes on ImageNet-LT test set, together with the constellations $\mathbf{w}_{kj}$.

strategies on feature embedding and classifier, and achieves this by cumulative learning. These methods, however, do not discuss the class geometry problem. In face recognition, Liu *et al.* [70] explores the long-tailed problem by knowledge transfer. The idea is similar to ours. But they achieve this by data synthesis, while we rely on model design and training strategy.

GistNet is closest to the OLTR approach of [75], which uses a visual memory and attention to propagate information between classes. This, however, is insufficient to guarantee the transfer of geometric class structure, as intended by GIST.

Few-shot Learning

Few-shot learning is a well-researched problem. A popular group of approaches is based on meta-learning, using gradient based methods such as MAML and its variants [24, 25], or LEO [103]. These methods take advantage of second derivatives to update the model from few-shot samples. Alternatively, the problem has been addressed with metric based solutions, such as the matching [129], prototypical [109], and relation [118] networks. These approaches learn metric embeddings that are transferable across classes.

There have also been proposals for feature augmentation, aimed to augment the data available for training, *e.g.* by combining GANs with meta-learning [132], synthesizing features across object views [69] or other forms of data hallucination [36]. All these methods are designed specifically for few-shot classes and often under-perform for many-shot classes.

Learning without Forgetting

Learning without forgetting aims to train a model sequentially on new tasks without forgetting the ones already learned. This problem has been recently considered in the few-shot setting [29], where the sequence of tasks includes a mix of many-shot and few-shot classes.

Proposed solutions [29, 99] attempt to deal with this by training on many-shots first, using the many-shot class weights to generate few-shot class weights, and combining them together. These techniques are difficult to generalize to long-tailed recognition, where the transition from many- to few- shot classes is continuous and includes a large number of medium-shot classes.

5.3 Geometric Structure Transfer

In this section, we introduce the proposed solution of the long-tailed recognition problem by geometric structure transfer and the GistNet architecture.

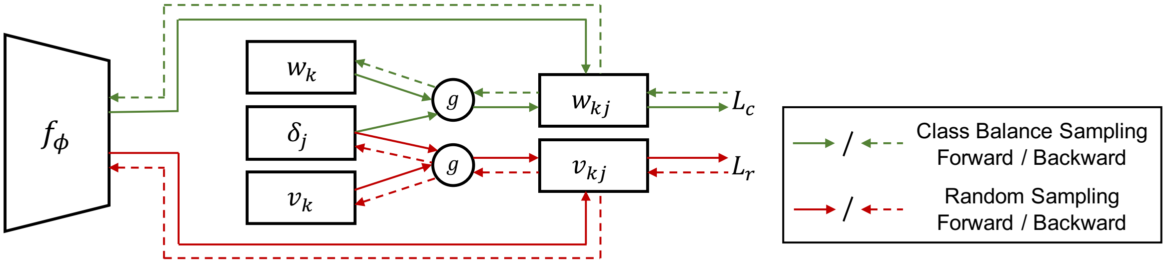

f_ϕ
w_k
δ_j
v_k
g
g
w_{kj}
v_{kj}
L_c
L_r
/ Class Balance Sampling Forward / Backward
/ Random Sampling Forward / Backward

Figure 5.4: GIST training. Solid arrows represent feed-forward and dashed ones back-propagation. Class-balanced mini-batches are used for the green connections, to guarantee that the parameters $\mathbf{w}_k$ are class-specific. Random sampling mini-batches are used for the red connections, enabling the displacements δ_j to be learned predominantly from many-shot classes. Note that the shape parameters δ_j receive no gradient from the class-balanced loss $\mathcal{L}_c$ and the constellation centers $\mathbf{w}_k$ receive no gradient from the random sampling loss $\mathcal{L}_r$.

5.3.1 Regularization by Geometric Structure Transfer

A popular architecture for classification is the softmax classifier. This consists of an embedding that maps images $\mathbf{x} \in \mathcal{X}$ into feature vectors $f_\phi(\mathbf{x}) \in \mathcal{F}$, implemented by multiple neural network layers, and a softmax layer that estimates class posterior probabilities according to

$$p(y = k|\mathbf{x}; \phi, \mathbf{w}_k) = \frac{\exp[\mathbf{w}_k^T f_\phi(\mathbf{x})]}{\sum_{k'} \exp[\mathbf{w}_{k'}^T f_\phi(\mathbf{x})]} \tag{5.1}$$

where ϕ denotes the embedding parameters and $\mathbf{w}_k$ is the weight vector of the k^{th} class.

The model is learned with a training set $\mathbb{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n^s}$ of n^s examples, by minimizing the cross entropy loss

$$\mathcal{L}_{CE} = \sum_{(\mathbf{x}_i, y_i) \in \mathbb{S}} -\log p(y_i|\mathbf{x}_i). \tag{5.2}$$

Recognition performance is evaluated on a test set $\mathbb{T} = \{(x_i, y_i)\}_{i=1}^{n^t}$, of n^t examples.

Learning with (5.2) produces a particular data-driven embedding geometry, which we denote the *natural* geometry for the training data. While parameters $\mathbf{w}_k$ of the classifier is class-specific and describes class centers, it is usually impossible to determine this geometry from the learned network parameters (shown in Sec. 5.4).

This is not a problem in regular large-scale recognition. In such a case, each class has enough training data and the natural geometry is successfully learned under cross-entropy loss without further regulations. For long-tailed recognition problems the situation is different. As in few-shot learning, the limited training data of few-shot classes leads to weakly defined class-conditional distributions and embedding geometry. However, this is not the case for classes with many samples, whose natural geometry can be learned from the data. In result, as illustrated in the left of Figure 5.1, the true class boundaries are usually not well learned for the few-shot classes.

In this work, we seek to leverage geometric regularization to improve the learning of the few-shot classes without sacrificing performance for the populated classes.

One possibility would be to enforce a pre-defined geometry for all classes, *e.g.* by adopting Mahalanobis distance $d(f_\phi(\mathbf{x}),\mu) = (f_\phi(\mathbf{x}) - \mu)^T \Sigma^{-1} (f_\phi(\mathbf{x}) - \mu)$ associated with Gaussian class conditionals of covariance Σ, or by assuming Gaussian class-conditionals and regularizing the covariance to be close to a pre-defined Σ.

This has several problems. First, it is not clear what the covariance Σ should be. Second, it ignores the natural geometry of the popular classes, which is well learned by the classifier of (5.1). Third, given the large dimensionality of $f_\phi(\mathbf{x})$, covariance regularization is difficult to implement, even for classes with many examples.

To avoid these problems, we seek a learning-based solution that does not require covariance estimation and leverages the natural geometry of the popular classes to regularize the geometry of the few-shot classes. Rather than forcing geometry through a distance function, which is hard to learn and implement, we pursue an alternative approach to guarantee that all classes have a *shared* geometric structure.

Ideally, this structure should be learned from data, so as to 1) follow the natural geometry of the highly populated classes, and 2) allow the transfer of that geometry to the classes of few examples. It should also be encoded in a relatively small number of parameters, which at most grows linearly with the dimension of $f_\phi(\mathbf{x})$.

To achieve these goals, we continue to rely on the softmax classifier of (5.1) and the cross-entropy loss of (5.2), but use an alternative implementation of the softmax layer

$$\begin{aligned} p_\phi(y=k|\mathbf{x}) &= \frac{\exp[\max_j \mathbf{w}_{kj}^T f_\phi(\mathbf{x})]}{\sum_{k'} \exp[\max_j \mathbf{w}_{k'j}^T f_\phi(\mathbf{x})]}, \\ \mathbf{w}_{kj} &= g(\mathbf{w}_k, \delta_j), \end{aligned} \tag{5.3}$$

where the canonical parameter vector $\mathbf{w}_k$ is replaced by a *constellation* of parameter vectors $\mathbf{w}_{kj}$, which are a function of $\mathbf{w}_k$ and a set of *structure parameters* δ_j shared by all classes. Under the simplest implementation of this idea, $g(\mathbf{w}_k, \delta_j) = \mathbf{w}_k + \delta_j$ and the structure parameters are a set of displacement vectors, as shown in the right of Figure 5.1.

Since these displacements are shared by all classes, the constellation is simply replicated around each $\mathbf{w}_k$, which is learned per class. Because, under the loss of (5.2), the highly populated classes tend to dominate the optimization of the shared parameters, the displacements δ_j tend to follow the natural geometry of these classes, which is thus transferred to the few-shot classes. This regularizes the learning of these classes, enabling the recovery of the true classification boundaries, as shown in the right of Figure 5.1.

The displacements δ_j are the parameters that contain geometry information. They transfer the geometry from highly populated classes to few-shot classes. With the help of geometry transfer, the model learns a better geometry for few-shot classes.

As shown in Figure 5.2, (5.3) is equivalent to replacing the natural geometry by several spherical Gaussians of means w_{kj} and choosing the one closest to the feature. This approximates the non-regulated geometry by a constellation of 5 spherical Gaussians, one per $\mathbf{w}_{kj}$. This geometry is visualized in Figure 5.3, where features from different classes are regulated by class specific constellations respectively. The constellation can be regarded as an umbrella. The model can learn the shape of the umbrella and where to place the umbrella for each class.

We denote the approach as *GeometrIc Structure Transfer* (GIST), to capture the fact that it transfers the essence, or gist, of the class geometry from popular to few-shot classes.

Note that the classifier in (5.3) is different from that in (5.1). There is an additional constraint: that the displacements δ_j are constant across classes. To avoid the model learns w_k to fit one of the constellations and ignore others. We first train the classifier from (5.1) to get a stable initialization of w_k, and then the whole classifier is trained to get the class-agnostic displacements. In such a case, the model will have to fit all available constellations to get lower loss instead of fitting one of them. Empirical examination in Section 5.5.3 shows the actual usage of $\{\delta_j\}$ is decent, and supports this assumption.

5.4 Geometry of the Cross-Entropy Classifier

A popular architecture for classification is the softmax classifier. This consists of an embedding that maps images $\mathbf{x} \in \mathcal{X}$ into feature vectors $f_\phi(\mathbf{x}) \in \mathcal{F}$, implemented by multiple neural network layers, and a softmax layer that estimates class posterior probabilities according to

$$p(y=k|\mathbf{x};\phi,\mathbf{w}_k) = \frac{\exp[\mathbf{w}_k^T f_\phi(\mathbf{x})]}{\sum_{k'} \exp[\mathbf{w}_{k'}^T f_\phi(\mathbf{x})]} \tag{5.4}$$

where ϕ denotes the embedding parameters and $\mathbf{w}_k$ is the weight vector of the k^{th} class. The model is learned with a training set $\mathbb{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n^s}$ of n^s examples, by minimizing the cross entropy loss

$$\mathcal{L}_{CE} = \sum_{(\mathbf{x}_i,y_i)\in\mathbb{S}} -\log p(y_i|\mathbf{x}_i). \tag{5.5}$$

Recognition performance is evaluated on a test set $\mathbb{T} = \{(x_i, y_i)\}_{i=1}^{n^t}$, of n^t examples.

From Bayes rule

$$p(y=k|\mathbf{x}) = \frac{p(\mathbf{x}|y=k)p(y=k)}{\sum_{k'} p(\mathbf{x}|y=k')p(y=k')}, \tag{5.6}$$

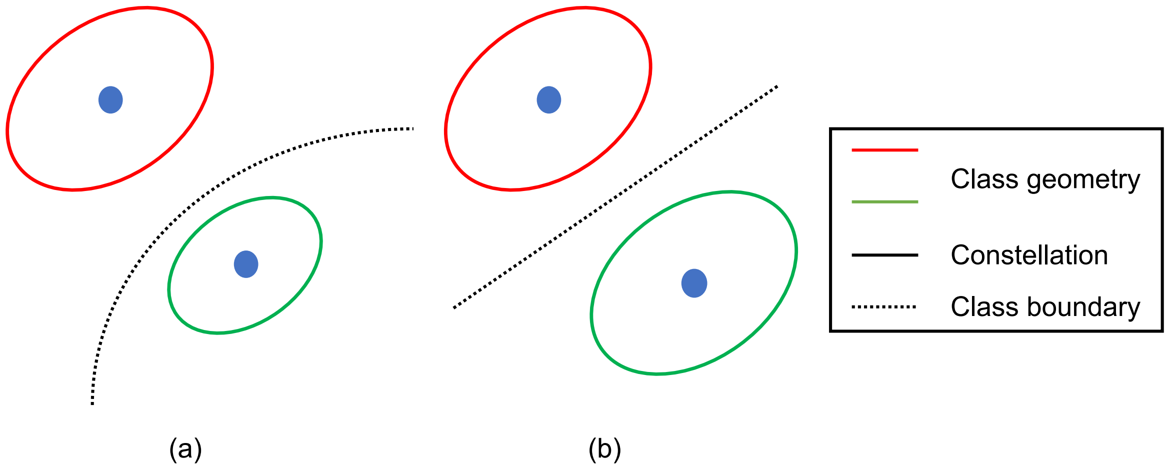


Figure 5.5: (a) The optimal classifier for two Gaussians of different covariance has quadratic boundary; (b) Linear boundary requires shared covariance.

the posterior distributions of (5.4) constrain the class-conditional distributions to the form

$$p(\mathbf{x}|y=k) \propto_{\mathbf{x}} \exp[\mathbf{w}_k^T f_\phi(\mathbf{x})], \tag{5.7}$$

where $\propto_{\mathbf{x}}$ means a proportional relationship that depends on $\mathbf{x}$. This implies that

$$p(\mathbf{x}|y=k) = h(f_\phi(\mathbf{x}))\exp[\mathbf{w}_k^T f_\phi(\mathbf{x}) - A(\mathbf{w}_k)] \tag{5.8}$$

where $h(.)$ is any non-negative function, and $A(.)$ a constant such that (5.8) integrates to 1. Hence, $p(\mathbf{x}|y=k)$ is an exponential family distribution of canonical parameter $\mathbf{w}_k$, sufficient statistic $f_\phi(\mathbf{x})$, cumulant function $A(\mathbf{w}_k)$ and underlying measure $h(.)$ [58]. This probability distribution can be uniquely expressed as [2]

$$p(\mathbf{x}|y=k) = u(f_\phi(\mathbf{x}))\exp[-d_\gamma(f_\phi(\mathbf{x}),\mu_k)], \tag{5.9}$$

where $\mu_k = \nabla A(\mathbf{w_k})$ is the mean of $p(\mathbf{x}|y=k)$, ∇A is the gradient of A, $u(.) = h(.)e^{\gamma(.)}$, and

$d_\gamma(.,.)$ is the Bregman divergence [6] with respect to the function

$$\gamma(\mu_k) = \mathbf{w}_k^T \mu_k - A(\mathbf{w}_k). \tag{5.10}$$

Since the cumulant $A(.)$ defines $\gamma(.)$, it determines the distance function $d_\gamma(.,.)$ and thus the geometry of the embedding.

While the discussion above applies to any exponential family distribution, the equalities above are particularly simple to verify for the case where the class-conditional distributions are spherical Gaussians (covariances $\Sigma_k = \sigma^2\mathbf{I}$). In this case

$$p(\mathbf{x}|y=k) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} e^{-\frac{1}{2\sigma^2}||f_\phi(\mathbf{x})-\mu_k||^2}, \tag{5.11}$$

which can be written as (5.9) with

$$d_\gamma(f_\phi(\mathbf{x}),\mu_k) = \frac{1}{2\sigma^2} e^{-||f_\phi(\mathbf{x})-\mu_k||^2}. \tag{5.12}$$

Similarly, they can be written in the form of (5.8), by expanding the 2-norm in the exponent and defining

$$\begin{aligned} h(f_\phi(\mathbf{x})) &= \frac{1}{\sqrt{(2\pi\sigma^2)^d}} e^{-\frac{1}{2\sigma^2}||f_\phi(\mathbf{x})||^2} && (5.13)\\ \mathbf{w} &= \frac{1}{\sigma^2}\mu && (5.14)\\ A(\mathbf{w}) &= \frac{\sigma^2}{2}||\mathbf{w}||^2. && (5.15) \end{aligned}$$

From (5.10) it follows that $\gamma(\mu) = \frac{1}{2\sigma^2}||\mu||^2$, which generates the Bregman divergence of (5.12), leading to an Euclidean geometry for all classes and spherically Gaussian class conditionals.

The point of the discussion above is that the softmax form of (5.4) constrains the geometry of the embedding, by defining the distance $d_\gamma(.,.)$. The fact that (5.4) is a linear classifier, i.e. has *linear class boundaries,* places further constraints on this geometry. Consider the case where the Gaussian class-conditionals have different covariances Σ_k. In this case, the optimal classifier is a "softmax" of the form $p(y=k|\mathbf{z}) = (e^{-\mathbf{z}^T\Sigma_k^{-1}\mathbf{z}+\mathbf{w}_k^T\mathbf{z}-\mathbf{b}_k})/(\sum_j e^{-\mathbf{z}^T\Sigma_j^{-1}\mathbf{z}+\mathbf{w}_j^T\mathbf{z}-\mathbf{b}_j})$ [21], where $\mathbf{z} = f_\phi(\mathbf{x})$, and has *quadratic* boundaries, as shown in the left of Figure 5.5. The problem is that this classifier would require a different divergence

$$d_{\gamma_k}(f_\phi(\mathbf{x}),\mu_k) = \frac{1}{2\sigma^2}e^{-(f_\phi(\mathbf{x})-\mu_k)^T\Sigma_k^{-1}(f_\phi(\mathbf{x})-\mu_k)} \tag{5.16}$$

per class, and this is not feasible under the discussion of the previous section. While the Gaussian of generic covariance can still be written in the exponential family form, this requires a quadratic transformation of $f_\phi(\mathbf{x})$.

It follows that the linear classifier is optimal *if and only if* the covariance is shared, i.e. $\Sigma_k = \Sigma, \forall k$, in which case all quadratic terms of $f_\phi(\mathbf{x})$ can be absorbed in $h(f_\phi(\mathbf{x}))$, as in (5.13), and thus cancel when (5.8) is inserted in (5.6). This is illustrated in the right of Figure 5.5. It follows that learning with the softmax model indirectly encourages the classes to have shared geometry. If the geometry were different, the CNN of (5.4) *could not* be optimal.

While learning with (5.5) produces a particular embedding geometry, which we denote the *natural* geometry for the training data, it is usually impossible to determine this geometry from the learned network parameters, because the cumulant $A(\mathbf{w}_k)$ is not observable in (5.4). On the other hand, it is possible to enforce a certain geometry through regularization. The simplest way to implement this *geometric regularization* is to implement the classifier with (5.9) and a pre-specified divergence $d_\gamma(.,.)$, e.g. the Euclidean distance. This encourages a desired geometry, e.g. Euclidean, and corresponding class-conditionals, e.g. Gaussian, even for a network trained with a few examples per class. Hence, it can improve generalization when training data is scarce. This type of regularization is leveraged by popular architectures for few-shot learning, e.g. the prototypical network [109].

5.4.1 Normalization

Recent works [29, 75] have shown that better few-shot or long-tailed classification accuracies are frequently obtained by performing the classification on the unit sphere, *i.e.* normalizing both embedding and classifier parameters to have unit norm. We follow this practice and adopt the weighted cosine classifier [29], replacing (5.3) with

$$\begin{aligned} p_\phi(y=k|\mathbf{x}) &= \frac{\exp[\max_j s_\tau(f_\phi(\mathbf{x}),\mathbf{w}_{kj})]}{\sum_{k'}\exp[\max_j s_\tau(f_\phi(\mathbf{x}),\mathbf{w}_{k'j}]}, \\ s_\tau(f_\phi(\mathbf{x}),\mathbf{w}) &= \tau\frac{\mathbf{w}^T f_\phi(\mathbf{x})}{||\mathbf{w}||||f_\phi(\mathbf{x})||} \end{aligned} \tag{5.17}$$

where τ is a parameter that controls the smoothness of the posterior distribution. This architecture is denoted as GistNet. In our implementation, τ is randomly initialized and learned end-to-end.

5.4.2 GIST Training

Deep networks are trained by stochastic gradient descent (SGD). This randomly samples mini-batches of b samples, and iterates across the training set. Due to the extreme class imbalance of long-tailed recognition, SGD tends to bias the model towards the classes with more samples.

In the literature, this problem is usually addressed by class-balanced sampling [145]. This first randomly samples b_c classes with equal probability, and draws b_n samples per class, producing a mini-batch of $b = b_c \times b_n$ samples. By iterating through all classes, the model is trained with an overall equal number of examples per class. For the classifier of (5.1), class-balanced sampling can significantly outperform regular sampling on few-shot classes. This also makes it a good solution for learning the class specific parameters $\{\mathbf{w}_k\}$ of GistNet.

However, the bias of regular sampling towards the highly populated classes is an *advantage* for the learning of the structure parameters $\{\delta_j\}$. After all, the point is exactly to learn these parameters from classes with substantial data and transfer them to the few-shot classes, where they cannot be learned accurately. Since the parameters are shared, both goals are accomplished

Table 5.1: Results on ImageNet-LT and Places-LT. ResNet-10/152 are used for all methods. For many-shot $t > 100$, for medium-shot $t \in (20, 100]$, and for few-shot $t \leq 20$, where t is the number of training samples.

	ImageNet-LT			
Method	Overall	Many-Shot	Medium-Shot	Few-Shot
Plain Model	23.5	41.1	14.9	3.6
Lifted Loss [86]	30.8	35.8	30.4	17.9
Focal Loss [68]	30.5	36.4	29.9	16.0
Range Loss [145]	30.7	35.8	30.3	17.6
FSLwF [29]	28.4	40.9	22.1	15.0
OLTR [75]	35.6	43.2	35.1	18.5
Decoupling [50]	41.4	51.8	38.8	21.5
Distill [141]	38.8	47.0	37.9	19.2
GistNet	**42.2**	**52.8**	**39.8**	**21.7**

	Places-LT			
Method	Overall	Many-Shot	Medium-Shot	Few-Shot
Plain Model	27.2	**45.9**	22.4	0.36
Lifted Loss [86]	35.2	41.1	35.4	24.0
Focal Loss [68]	34.6	41.1	34.8	22.4
Range Loss [145]	35.1	41.1	35.4	23.2
FSLwF [29]	34.9	43.9	29.9	29.5
OLTR [75]	35.9	44.7	37.0	25.3
Decoupling [50]	37.9	37.8	40.7	31.8
Distill [141]	36.2	39.3	39.6	24.2
GistNet	**39.6**	42.5	**40.8**	**32.1**

Table 5.2: Results on the iNaturalist 2018. All methods are implemented with ResNet-50.

Method	Accuracy
CB-Focal [17]	61.1
LDAM+DRW [7]	68.0
Decoupling [50]	69.5
GistNet	**70.8**

if the learning procedure emphasizes the highly populated classes, as is the case for regular sampling. This implies that GIST training should include a mix of regular sampling (for shared structure parameters) and class-balanced sampling (for class specific parameters).

We propose to implement this with the hybrid training scheme of Figure 5.4. In each iteration, two mini-batches $\mathbb{S}_c$, $\mathbb{S}_r$ are first sampled from the training set $\mathbb{S}$ by class-balanced sampling and random sampling, respectively. Two sets of class-specific parameters, $\{\mathbf{w}_k, \boldsymbol{\nu}_k\}$ are then learned, using the combination of (5.2), (5.3), and (5.17). The class-balanced mini-batch $\mathbb{S}_c$ is used with the resulting loss

$$\mathcal{L}_c = \sum_{(\mathbf{x}_i, y_i) \in \mathbb{S}_c} \{-\max_j s(f_\phi(\mathbf{x}_i), \mathbf{w}_{y_i j}) + \log \sum_k \exp[\max_j s(f_\phi(\mathbf{x}_i), \mathbf{w}_{kj})]\}, \quad \mathbf{w}_{kj} = g(\mathbf{w}_k, \delta_j) \tag{5.18}$$

to learn the parameters $\mathbf{w}_k$. The random mini-batch $\mathbb{S}_r$ is used with the loss

$$\mathcal{L}_r = \sum_{(\mathbf{x}_i, y_i) \in \mathbb{S}_r} \{-\max_j s(f_\phi(\mathbf{x}_i), \boldsymbol{\nu}_{y_i j}) + \log \sum_k \exp[\max_j s(f_\phi(\mathbf{x}_i), \boldsymbol{\nu}_{kj})]\}, \quad \boldsymbol{\nu}_{kj} = g(\boldsymbol{\nu}_k, \delta_j) \tag{5.19}$$

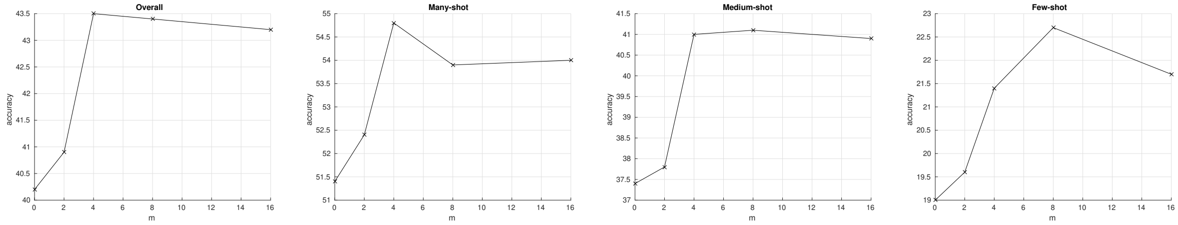
to learn the parameters $\boldsymbol{\nu}_k$. This results in the overall loss

$$\mathcal{L} = \mathcal{L}_r + \lambda \mathcal{L}_c. \tag{5.20}$$

The structure parameters δ_j are common to the two losses. However, as shown in Figure 5.4, during back-propagation only the gradient from $\mathcal{L}_r$ is used to update these parameters. This guarantees that the geometric structure is learned with random sampling. This structure is, however, propagated to the learning of the class specific parameters $\mathbf{w}_k$, which receive the gradient $\mathcal{L}_c$. In this way, the class specific parameters $\mathbf{w}_k$ are learned with class-balanced sampling, but this learning is informed by the structure parameters δ_j learned with random sampling. This leads to parameter constellations $\mathbf{w}_{kj}$ that, while shared across classes, are centered at class-specific locations.

Table 5.3: Ablation of GistNet components, on the ImageNet-LT *validation set*. For many-shot $t > 100$, for medium-shot $t \in (20, 100]$, and for few-shot $t \leq 20$, where t is the number of training samples.

Method	Overall	Many-Shot	Medium-Shot	Few-Shot
Plain Model	25.1	42.9	16.6	0.43
COS+CB	37.6	49.4	34.8	14.7
COS+CS+CB	39.5	52.6	36.3	14.5
COS+CS+GIST (GistNet)	**43.5**	**54.8**	**41.0**	**21.4**
COS+GIST	40.2	51.4	37.4	19.0
COS+CS+GIST ($\mathbf{w}_k$ and $\boldsymbol{\nu}_k$ combined)	40.9	**58.2**	34.6	14.8
COS+CS+GIST (g rotation)	**43.6**	55.1	40.8	**21.7**
COS+CS+GIST (g MLP)	43.4	54.2	**41.1**	21.5



Overall Many-shot Medium-shot Few-shot

Figure 5.6: Results on different size of structure parameters in few-shot, medium-shot, many-shot classes, and overall accuracy, searched on validation set.

Note that the displacements are forwarded together with $\mathbf{w}_k$ to calculate the class-balanced loss $\mathcal{L}_c$. This makes the two components $\{\mathbf{w}_j\}$ and $\{\delta_j\}$ of the classifier matching each other, although they are learned by different losses. The parameters $\boldsymbol{\nu}_k$ are only used at training time, to guarantee that the geometric parameters δ_j follow the natural geometry of the highly populated classes. They are discarded after training.

In GIST training, the class-specific weights $\mathbf{w}_k$ are trained with class-balanced sampling, while the structure parameters δ_j are trained with random sampling. This forces the latter to predominantly represent the structure of the popular classes and is what enables the geometric structure transfer of Figure 5.1.

5.5 Experiments

In this section, we discuss an evaluation of the long-tailed recognition performance of the GistNet.

5.5.1 Experimental Set-up

Datasets

We consider three long-tailed recognition datasets, ImageNet-LT [75], Places-LT [75] and iNatrualist18 [128]. ImageNet-LT is a long-tailed version of ImageNet [19] by sampling a subset following the Pareto distribution with power value $\alpha = 6$. It contains 115.8K images from 1000 categories, with class cardinality ranging from 5 to 1280. Places-LT is a long-tailed version of the Places dataset [149]. It contains 184.5K images from 365 categories with class cardinality ranging from 5 to 4980. iNatrualist18 is a long-tailed dataset, which contains 437.5K images from 8141 categories with class cardinality ranging from 2 to 1000.

Baselines

Following [75], we consider three metric-learning baselines, based on the lifted [86], focal [68], and range [145] losses, and one state-of-the-art method, FSLwF [29], for learning without forgetting. We also include state-of-the-art long-tailed recognition methods designed specifically for these two datasets: OLTR [75], Decoupling [50], and Distill [141]. The classifier of (5.1) with standard random sampling is denoted as the *Plain Model* for comparison.

Training Details

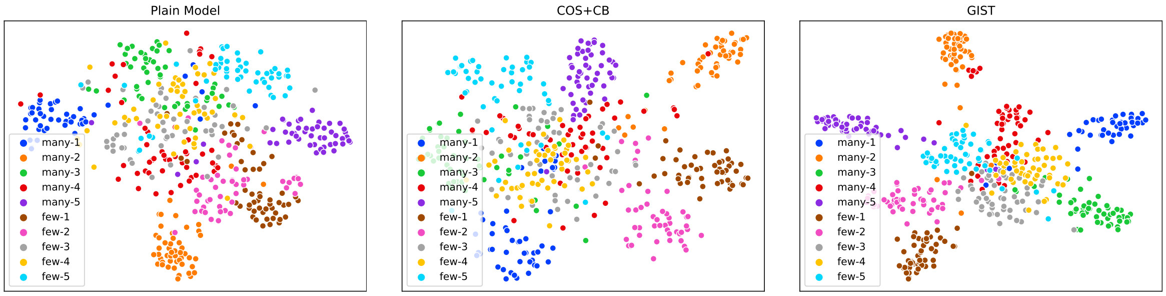
ResNet-10 and ResNet-152 [38] are used on ImageNet-LT and Places-LT respectively, and ResNet-50 is used on iNatrualist18. Unless otherwise noted, we use four vectors δ_j of structure parameters, each with the dimension of $f_\theta(\mathbf{x})$. The class center $\mathbf{w}_k$ completes a constellation of five vectors. The number of structure parameters is ablated in Section 5.5.3. In all experiments, $\lambda = 0.5$ is used in (5.20).

The model is first pre-trained without structure parameters, with 60 epochs of SGD, using momentum 0.9, weight decay 0.0005, and a learning rate of 0.1 that decays by 10% every 15 epochs. After this, the full model is subject to GIST training with momentum 0.9, weight decay 0.0005 for 60 epochs, and learning rate 0.1 that decays by 10% every 15 epochs. In this case, each iteration uses class-balanced and random sampling mini-batches of size 128, for an overall batch size of 256. One epoch is defined when the random sampling iterates over the entire training data. Codes are attached in supplementary.

5.5.2 Results

Table 5.1 present results on ImageNet-LT and Places-LT. GistNet outperforms all other methods on the two datasets. A further comparison is performed by splitting classes into *many shot* (more than 100 training samples), *medium shot* (between 20 and 100 training samples), and *few shot* (less that 20 training samples). GistNet achieves the best performance on 5 of the 6 partitions and is competitive on the remaining one.

While on Places-LT the largest gains are for the few-shot classes, in ImageNet-LT they hold for the medium- and many-shot classes. This suggests that, in this dataset, the remaining methods over-fit to the few-shot classes. The higher robustness of GistNet to this over-fitting can be explained by the predominance of the many-shot classes in the training of the structure parameters δ_j. Results on iNaturalist18 dataset are shown in Table 5.2, ours also outperforms all other methods.



Plain Model COS+CB GistNet

Figure 5.7: t-SNE visualizations of the embedding of *test set* images from 5 randomly chosen many- and few-shot ImageNet-LT classes, for three models.

5.5.3 Ablation Study

In this section, we discuss the effectiveness of the various components of GistNet, the choice of constellation function g, the number of structure parameters, and the actual usage of constellations. All models are trained and evaluated on the training and validation set of ImageNet-LT, respectively, using a ResNet-10 backbone.

Component Ablation

Starting from the plain model of (5.1), we incrementally add the cosine classifier (COS) used in (5.17), class-balanced sampling (CB), class structure parameters (CS), and GIST training (GIST). Table 5.3 shows that the combination of cosine classifier and class-balanced sampling (COS+CB) improves significantly on the plain model. The simple addition of the class structure parameters (COS+CS+CB) further improves the overall performance.

However, there is no significant improvement for few-shot classes. This can be explained by the fact that, with class-balanced sampling, the three class types are equally predominant for the learning of the structure parameters. Hence, there is no transfer of geometric structure from many- to few-shot classes. This is confirmed by the fact that, when GIST training is added (COS+CS+GIST), performance improves significantly for the few-shot classes. When compared to COS+CB, the GistNet model (COS+CS+GIST) has an overall gain of about 6 points and better performance for all class types. Among these, the gains are particularly large (around 6.5 points) for the few-shot classes.

The middle of the table investigates other possible configurations of the GistNet. Applying GIST training without class structure parameters (COS+GIST), *i.e.* using the combination of class balanced and random sampling only to learn the embedding $f_\phi(\mathbf{x})$, degrades performance for all class partitions. This shows the importance of enforcing a shared class structure among all classes.

Another variant is to remove the additional class centers $\{\mathbf{v}_k\}$ of Figure 5.4, using the centers $\{\mathbf{w}_k\}$ for both losses, *i.e.* replacing $\mathbf{v}_k$ with $\mathbf{w}_k$ in (5.19). This variant, denoted COS+CS+GIST ($\mathbf{w}_k$ and $\mathbf{v}_k$ combined), eliminates all the gains of GistNet for few-shot classes, while increasing the recognition accuracy for those in the many-shot partition. This is because the centers now receive gradient from the random sampling loss and are predominantly trained with many-shot data. The improved performance of GistNet over this variant shows that it is important to maintain the class-specificity of center training, while enforcing transfer of the geometric structure parameters, as done by GIST.

Different Choices of g

Beyond these variants, we have also considered different choices for the function g that defines the parameter constellations of (5.3). In addition to the default addition function implemented by GistNet, we considered two possibilities.

The first was a rotation. After the embedding and classifier parameters are normalized, we evaluate the distance between them on the d-dimensional unit sphere (where d is the dimension of $f_\phi(\mathbf{x})$). The structure parameters are then d-dimensional rotation matrices, which encourage all classes to have the same structure on the unit sphere. This is implement the rotation matrix by a transformation of d-dimensional displacement vector

$$\mathbf{R} = \mathbf{I} - \mathbf{u}\mathbf{u}^T - \mathbf{v}\mathbf{v}^T + [\mathbf{u},\mathbf{v}]\mathbf{R}_\theta[\mathbf{u},\mathbf{v}]^T, \tag{5.21}$$

where $\mathbf{u}$ is a unit vector, $\mathbf{v}$ is the normalized vector of a displacement vector δ_j, and $\mathbf{R}_\theta$ is the 2D rotation matrix between $\mathbf{u}$ and δ. Given a structure parameter vector δ_j, the parameter constellations are implemented as

$$\mathbf{w}_{kj} = g(\mathbf{w}_k, \delta_j) = \mathbf{R}\mathbf{w}_k \tag{5.22}$$

Details are discussed in supplementary.

The second was a learned function g, implemented by a two-layer MLP, and learned end-to-end.

Table 5.3 shows that the different implementations of g have little impact on the recognition performance. This suggests that the addition of global geometric constraints is much more important than the specific implementation details of these constraints.

Number of Structure Parameters

We next investigated the influence of the number m of structure parameters $\{\delta_j\}_{j=1}^m$. As shown in Figure 5.6, none of the alternatives tried ($m \in \{2,8,16\}$) outperformed the four parameters used in GistNet. For overall, many-, and medium-shot classes performance increases until $m = 4$ and then saturates. For few-shot classes, there was a one-point gain in using $m = 8$. This shows that this partition is the one that most benefits from geometry transfer.

Overall, these results confirm that while geometric transfer can produce significant gains, the GistNet architecture is quite robust to variations of detail.

Actual Usage of Constellations

Cross-entropy minimization encourages the use of more δ_j, since the coverage of the class distributions is better. It would be a waste not to use them all. In the test set of ImageNet-LT, the actual usage was $\{25\%, 23\%, 18\%, 17\%, 17\%\}$. 792 of 1000 classes chose each δ_j for at least 10% of test samples. This results further support that the model does not collapse to traditional classifier by fitting to only one constellation and ignoring others.

5.5.4 Visualization

Figure 5.7 shows a t-SNE [79] visualization of the embeddings learned by the Plain Model, the COS+CB baseline, and GistNet. For clarity, we randomly choose five classes from the many- and few-shot splits in ImageNet-LT. The figure shows the t-SNE projection of features of *test* samples from those classes. Compared to the two other models, GistNet produces classes that are better separated and have more consistent geometry. This is especially true for few-shot classes.

5.6 Conclusion

This work addressed the long-tailed recognition problem. A new architecture, GistNet, and training scheme, GIST, were proposed to enable transfer of geometric structure from highly populated to low-populated classes. This leverages the tendency of SGD training to over-fit to the populated classes, rather than simply fighting this tendency.

GistNet was shown to achieve state-of-the-art performance on two popular long-tailed datasets. Ablation studies have shown that, while geometric transfer enables significant recognition gains, the architecture is quite robust to variations of detail. This suggests that the addition of global geometric constraints to long-tailed recognition is more important than the specific implementation of these constraints.

Chapter 5 is, in full, based on the material as it appears in the submission of “GistNet:

a Geometric Structure Transfer Network for Long-Tailed Recognition”, Bo Liu, Haoxiang Li, Hao Kang, Gang Hua, Nuno Vasconcelos, in *IEEE International Conference on Computer Vision* (ICCV), 2021. The dissertation author was the primary investigator and author of this material.

Chapter 6

Breadcrumbs: Adversarial Class-Balanced Sampling for Long-tailed Recognition

6.1 Introduction

The availability of large-scale datasets, with many images per class [19], has been a major factor in the success of deep learning for computer vision. However, these datasets are manually curated and artificially balanced. This is unlike most real world applications, where the frequencies of examples from different classes can be highly unbalanced, leading to skewed distributions with long tails. These datasets are composed by a few popular classes and many rare classes. This class imbalance has been observed in image classification [133], face identification [44, 74], object detection [68, 153], and many other applications. Researchers have tackled it from various angles, including zero-shot learning [23, 139, 140], few-shot learning [129, 109, 24], and more recently long-tailed recognition [75].

In this work, we focus on the long-tailed recognition setting, where classes are grouped into three types that differ in training sample cardinality: many-shot ($>$ 100 samples), medium-shot (between 20 and 100 samples), and few-shot ($\leq$ 20 samples). Performance is evaluated over each group independently, in addition to the overall classification accuracy. While training data is highly unbalanced, the test set is kept balanced so that equally good performance on all classes is a requisite for high accuracy. One of the insights from the long-tailed recognition literature is that techniques targeting specific dataset limitations, *e.g.* few-shot learning by data augmentation [14, 132], predicting classifier weights [92], prototype-based non-parametric classifiers [109], and optimization with second derivatives [24], are frequently harmful to classes that do not suffer from those limitations, *e.g.* many-shot. Hence, it is important to address the problem holistically, considering all types of classes simultaneously.

Since long-tailed recognition datasets have a continuous coverage of the number of samples per class, they are best addressed by training a model on the entire dataset, in a way robust to data imbalance. Standard classifier training follows the *sample-balanced* sampling setting of Figure 6.1. This consists of sampling images uniformly to create batches for network

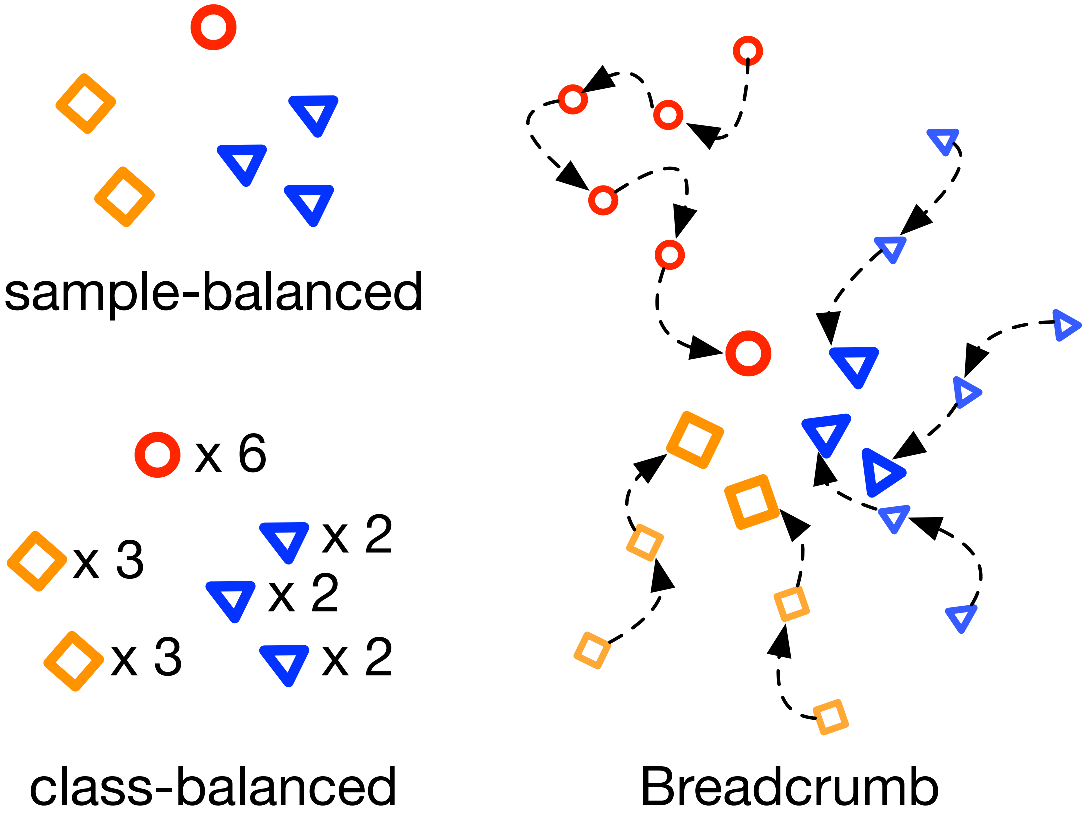


Figure 6.1: Upper-Left: Random sampling is sample-balanced. The number of examples per class has a long-tailed distribution. This leads to under-fitting in few-shot classes. Lower-Left: Class-balanced sampling duplicates few-shot samples in feature space and can leads to over-fitting for these classes. Right: Breadcrumb produces trails of features by back-tracking through training epochs. This is shown to be an adversarial augmentation technique, which mitigates the over-fitting problem.

training. In result, as shown in the figure, few-shot classes (red) are under-represented and many-shot classes (blue) are over-represented in each batch. Hence, learning typically *under-fits* less populated classes. This has motivated procedures to fight class imbalance with data re-sampling [145] or cost-sensitive losses [68] that place more training emphasis on examples of lower populated classes. One of the more successful approaches is to decouple the training of feature embedding and classifier [50]. While the embedding is learned with image-balanced training, the classifier is trained with *class-balanced* sampling. As illustrated in Figure 6.1, this consists of sampling classes uniformly and then sampling uniformly within the class. However, for few shot classes, this approach leads to repeated sampling of the same examples. In result, the classifier can easily *over-fit* on few-shot classes.

In this work, we adopt the decoupled training strategy but seek to avoid over-fitting in the classifier training stage. For this, we propose to enrich the training data in the feature space at the output of the embedding, without extra computation. The idea is to back-track features to access the large diversity of feature vectors that are available per training image in prior epochs. This can be exploited to generate more diverse training data than simply replicating existing features. We refer to this procedure as *feature back-tracking*. As shown in Figure 6.1, it allows the sampling of large numbers of feature vectors from the few-shot classes without duplication. Since the embedding changes across training epochs, an alignment is necessary to simplify network training. We show that a simple alignment of class means suffices to accomplish this goal and propose the *fEature augMentAtioN by bAck-tracking wiTh alignmEnt* (EMANATE) procedure. This consists of augmenting the feature vectors collected at an epoch with aligned replicas of the vectors that emanate from them by back-tracking.

A theoretical analysis shows that, unlike class-balanced sampling, EMANATE is an adversarial feature augmentation technique, in the sense that it is guaranteed to increase the training loss for any convergent training scheme. This places EMANATE in the realm of feature augmentation methods popular in the few-shot literature [14, 132]. However, these require extra computation to generate new examples and sometimes introduce convergence problems. EMANATE requires no extra computation and can be applied differently to each class, according to its number of samples. For classes with enough samples, only features from the last epoch are used, *i.e.* no re-sampling is performed. For those without, features are back-tracked over previous epochs, until there are enough features. This results in a new training feature set of higher variance for few-shot classes but forces no change on many-shot classes. In result, it is possible to improve classification accuracy for the former without degrading performance for the latter.

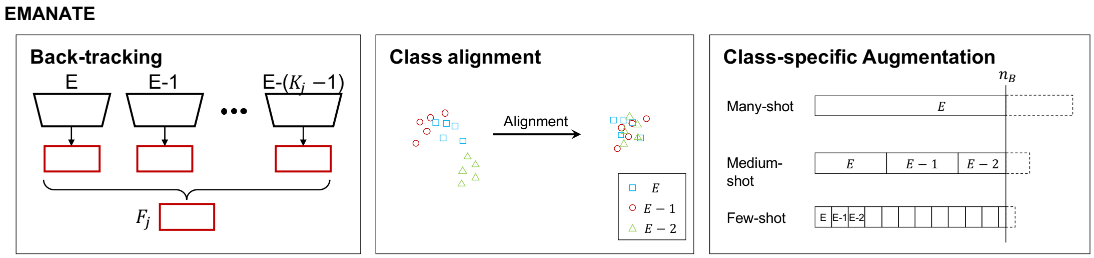


Figure 6.2: EMANATE. Left: features from embeddings learned in previous epochs are back-tracked to compose a class-balance training set. Middle: Class alignment aligns the means of features from different epochs. Right: different classes have different back-tracking lengths. Many-shot classes only collect features from the current epoch; medium-shot classes back-track for a few epochs; and few-shot classes from many. When the number of samples exceeds n_B, the earliest epoch is randomly sampled to meet this target.

A new sampling scheme, denoted *Breadcrumb Sampling* is then proposed to leverage the feature trails extracted by EMANATE, in the context of the two-stage training of class-balanced sampling. Breadcrumb Sampling relies on EMANATE to collect these feature trails in a first stage, when the embedding is trained with image-balanced sampling. In the second stage, the classifier is then learned with class-balanced training based on these trails. Two sampling variants are considered. Weak Breadcrumb Sampling only uses feature trails collected at the end of stage 1, *i.e.* once the embedding has converged. Strong Breadcrumb Sampling uses trails collected throughout stage 1 training, *i.e.* as the embedding evolves. This tends to create an even more adversarial training set.

Overall, this work makes several contributions. First, we point out that class-balanced sampling is not an adversarial augmentation technique, which limits its ability to combat overfitting in few-shot classes. Second, we propose EMANATE, a data augmentation technique that addresses this problem by feature back-tracking with alignment. Third, we show theoretically that, unlike class-balanced sampling, EMANATE is an adversarial technique. Fourth, we propose two variants of a new sampling scheme, Breadcrumbs, which leverage EMANATE to enable long-tailed recognition with state of the art performance. All of this is achieved with no extra computation and no performance degradation for classes with many examples.

6.2 Related Work

Long-tailed Recognition

Long-tailed recognition has recently received substantial attention [134, 86, 68, 145, 75, 133]. Several approaches have been proposed, including metric learning [86, 145], loss weighting [68], or meta-learning [133]. Some methods propose dedicated loss functions to mitigate the data imbalanced problem. For example, lift loss [86] introduces margins between many training samples. Range loss [145] encourages data from the same class to be close and different classes to be far away in the embedding space. The focal loss [68] dynamically balances weights of positive, hard negative, and easy negative samples. As reported by Liu *et al.* [75], when applied to long-tailed recognition, many of these methods improved accuracy of the few-shot group, but at the cost of lower accuracy for many-shot classes.

Other methods, *e.g.* class-balanced experts [107] and knowledge distill [141], try to mitigate this problem by artificially dividing the training data into subsets, based on number of examples, and training an expert per subset. However, experts learned from arbitrary data divisions can be sub-optimal, especially for few-shot classes, where training data is insufficient to learn the expert model.

More recent works [50, 150] achieve improved long-tailed recognition by training feature embedding and classifier with separate sampling strategies. The proposed Breadcrumbs approach follows this strategy, learning the embedding in a first stage with sample-balanced (random) sampling and the classifier in a second stage with class-balanced sampling. In fact, Breadcrumbs can be seen as a data augmentation method tailored for this strategy, improving its long-tailed recognition performance over all class groups.

Another related work is LEAP [70], a method mostly tested on person re-identification and face recognition problems, where datasets usually have long-tailed distributions. LEAP augments data samples from tail (few-shot) classes by transferring intra-class variations from head (many-shot) classes. This assumes a shared intra-class variation across classes, which can hold for person re-ID and face recognition but may not be applicable for general long-tailed recognition tasks. Besides, LEAP is technically orthogonal to Breadcrumbs and the two methods could potentially be combined for further improvement.

Few-shot Learning

Few-shot learning focus solely on the data scarcity problem. A large group of approaches is based on meta-learning, using gradient based methods such as MAML and its variants [24, 25], or LEO [103]. These methods take advantage of second derivatives to optimize the model from few-shot samples. Another group of methods, including matching network [129], prototypical network [109], and relation network [118],aims to learn robust metrics. Since these methods are designed specifically for few-shot classes, they often under-perform for many-shot classes, which makes them ineffective for long-tailed recognition.

Similarly to Breadcrumbs, some few-shot methods have proposed to augmenting training data by combining GANs with meta-learning [132], synthesizing features across object views [69] or using other forms of data hallucination [36]. All these method introduces non-negligible extra computation to generate the new data samples. The application of GAN-based methods to few-shot data without external large-scale datasets can also create convergence problem. In Breadcrumbs, data samples are augmented with saved feature vectors from prior epochs and no extra computation.

6.3 EMANATE

In this section, we introduce the data augmentation method that underlies Breadcrumbs.

6.3.1 Data Sampling and Decoupling Training

Consider an image recognition problem with training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i); i = 1, \ldots, N\}$, where x_i is an example and $y_i \in \{1, \ldots, C\}$ its label, where C is the number of classes. A CNN model combines a feature embedding $\mathbf{z} = f(\mathbf{x}; \theta) \in \mathbb{R}^d$, implemented by several convolutional layers of parameters θ, and a classifier $g(\mathbf{z}) \in [0,1]^C$ that operates on the embedding to produce a class prediction $\hat{y} = \arg\max_i g_i(\mathbf{z})$. Standard (image-balanced) CNN training relies on mini-batch SGD, where each batch is randomly sampled from $\mathcal{D}$. A class j of n_j training example has probability $\frac{n_j}{N}$ of being represented in the batch. Without loss of generality, we assume classes sorted by decreasing cardinality, *i.e.* $n_i \leq n_j, \forall i > j$.

In the long-tail setting, where $n_1 \gg n_C$, the model is not fully trained on classes of large index j (tail classes) and under-fits. This can be avoided with recourse to non-uniform sampling strategies, the most popular of which is class-balanced sampling. This samples each class with probability $\frac{1}{C}$, over-sampling tail classes, and is particularly successful when the training of embedding and classifier are decoupled [50], which is also simple to implement. The embedding is first trained with image-balanced sampling, and different sampling and structures can then be used for the classifier. In this work, we adopt the popular linear classifier $g(\mathbf{z}) = \nu(\mathbf{W}\mathbf{x} + \mathbf{b})$, where ν is the softmax function, and class-balanced sampling.

6.3.2 Augmentation by Feature back-tracking

Class-balanced sampling over-samples classes of few examples. For a class j with $n_j < N/C$ the over-sampling factor is $\rho = \frac{N}{Cn_j}$. In the long-tail setting, ρ is usually larger than 10. This heavily re-samples the few available samples and can lead to over-fitting, impairing generalization for tail classes. While over-fitting can be combated with data augmentation, traditional image-level methods, such as random cropping, horizontal flipping, or color jittering, make little difference in feature space, because the embedding is trained to be invariant to such transformations.

Feature-level augmentations have been investigated in the few-shot setting [132, 69, 36], but typically require training of additional models, which add complexity and sometimes have convergence problems. Ideally, the augmentation technique should be adversarial, *i.e.* increase training difficulty, and require little extra computation. One possibility is to rely on adversarial examples [32]. However, these require optimization at each training iteration and have large computational cost. In our experience, standard adversarial attacks are also not effective at improving generalization for tail classes, because they are too close to the few available examples.

In this work, we propose a different adversarial feature-level augmentation strategy, based on *feature backtracking.* The idea is that the embedding $f(\mathbf{x};\theta)$, obtained after training converges, is simply the final element in the family of embeddings $f(\mathbf{x};\theta^e)$ learned from epochs $e \in \{1,\ldots,E\}$, where E is the number of training epochs. It follows that a particular image $\mathbf{x}_i$ produces a sequence of feature vectors

$$\mathcal{B}_i = \{\mathbf{z}_i^e = f(\mathbf{x}_i;\theta^e)|e \in \{1,\ldots,E\}\} \tag{6.1}$$

during the optimization. We equate $\mathcal{B}_i$ to a trail of *bread crumbs* that can be backtracked, as illustrated in Figure 6.1(right). These bread crumbs can be used to perform data augmentation *without* added computation. It suffices to store, at epoch e the set of features

$$\mathcal{Z}^e = \{\mathbf{z}_i^e = f(\mathbf{x}_i;\theta^e)|\mathbf{x}_i \in \mathcal{D}\} \tag{6.2}$$

produced by the embedding learned at the end of the epoch. This is denoted as the training set *snapshot* at epoch e.

Since the embedding $f(\mathbf{x};\theta^e)$ changes with e, features from different epochs are usually not aligned in feature space. This may lead to bread crumb trails that are “all over the place,” *e.g.* because the space has been translated or rotated between epochs. Hence, when feature vectors collected at different epochs are to be used together, a *class alignment* is recommended to simplify the training. On the other hand, this alignment cannot be too strong, so as not to defeat the purpose of data-augmentation. In particular, the alignment operation should not jeopardize the adversarial nature of the latter. A simple operation, which is shown to satisfy this property in the following section, is to align the mean feature vectors synthesized per class during back-tracking. This consists of splitting $\mathcal{Z}^e$ into a set of class snapshots, where

$$\mathcal{Z}_y^e = \{\mathbf{z}_i^e \in \mathcal{Z}^e | y_i = y\} \tag{6.3}$$

is the snapshot of class y, compute the mean of each class

$$\bar{\mathbf{z}}_y^e = \frac{1}{n_j} \sum_{j=1}^{n_j} \mathbf{z}_{y,j}^e \tag{6.4}$$

where $\mathbf{z}_{y,j}^e$ is the j^{th} element of $\mathcal{Z}_y^e$, and apply

$$\mathbf{z}_{y,j}^{e' \to e} = \mathbf{z}_{y,i}^{e'} - \bar{\mathbf{z}}_j^{e'} + \bar{\mathbf{z}}_j^e, \tag{6.5}$$

where $\mathbf{z}_{y,j}^{e' \to e}$ is the alignment, with respect to snapshot e, of the j^{th} feature vector $\mathbf{z}_{y,j}^e$ of class y from epoch e'. This produces a snapshot *transferred from epoch e' to e*

$$\mathcal{Z}_y^{e' \to e} = \{\mathbf{z}_{y,j}^{e' \to e} | \mathbf{z}_{y,j}^{e'} \in \mathcal{Z}_y^{e'}\}. \tag{6.6}$$

This snapshot can then be combined with $\mathcal{Z}_y^e$ to produce an *augmented snapshot of class y for epoch e*

$$\mathcal{A}_y^e = \mathcal{Z}_y^e \bigcup \mathcal{Z}_y^{e' \to e}. \tag{6.7}$$

This process is denoted *fEature augMentAtioN by bAcktracking wiTh alignmEnt* (EMANATE), as $\mathcal{A}_y^e$ backtracks the breadcrumb trails that emanate from class y at epoch e.

6.3.3 Theoretical justification

In this section, we provide theoretical motivation for EMANATE as an adversarial data augmentation technique. Let $\nu(\mathbf{W}^e\mathbf{z}+\mathbf{b}^e)$ be the linear classifier learned at the end of epoch e, *i.e.* from the snapshots $\mathcal{Z}_y^e = \{\mathbf{z}_{y,i}^e\}$ of (6.3). The corresponding cross-entropy loss is

$$L(\mathcal{Z}^e,\mathbf{W}^e,\mathbf{b}^e) = \sum_y L_y(\mathcal{Z}_y^e,\mathbf{W}^e,\mathbf{b}^e) \tag{6.8}$$

where

$$L_y(\mathcal{Z}_y^e,\mathbf{W}^e,\mathbf{b}^e) = -\frac{1}{|\mathcal{Z}_y^e|}\Sigma_i \log \nu_y(\mathbf{W}^e\mathbf{z}_{y,i}^e+\mathbf{b}^e), \tag{6.9}$$

is the loss of class y and ν_y the y^{th} element of the softmax output. It is assumed that the classifier is optimal for the training data under this loss, *i.e.*

$$L_y(\mathcal{Z}_y^e,\mathbf{W}^e,\mathbf{b}^e) \leq L_y(\mathcal{Z}_y^e,\mathbf{W},\mathbf{b}), \quad \forall y,\mathbf{W},\mathbf{b}. \tag{6.10}$$

A feature augmentation procedure adds new features to $\mathcal{Z}_y^e$. It is denoted adversarial when the augmented training set is more challenging than the original.

Definition 1. *Consider the augmentation $\mathcal{A}_y^e$ of the training set snapshot $\mathcal{Z}_y^e$ from epoch e and class y. The augmentation is adversarial with respect to class y if*

$$L_y(\mathcal{A}_y^e,\mathbf{W}^e,\mathbf{b}^e) > L_y(\mathcal{Z}_y^e,\mathbf{W}^e,\mathbf{b}^e) \tag{6.11}$$

where $L_y(.)$ the loss of (6.9).

For low-shot classes y, class-balanced sampling replicates the features of $\mathcal{Z}_y^e$, creating the augmented feature set $\mathcal{A}_y^e = \mathcal{Z}_y^e \cup \mathcal{Z}_y^e$. Since, from (6.9)

$$\begin{aligned} L_y(\mathcal{A}_y^e, \mathbf{W}^e, \mathbf{b}^e) &= -\frac{1}{2|\mathcal{Z}_y^e|} 2\Sigma_i \log \nu_y(\mathbf{W}^e \mathbf{z}_{y,i}^e + \mathbf{b}^e), \\ &= L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e) \end{aligned} \tag{6.12}$$

we obtain the following corollary.

Corollary 1. *Class-balanced sampling is not an adversarial feature augmentation strategy.*

We next consider augmentation with EMANATE. The following lemma establishes a lower bound for the increase of the training loss under this augmentation technique.

Lemma 1. *Consider the augmentation of $\mathcal{Z}_y^e$ with the snapshot transferred from epoch $e' < e$ by EMANATE, i.e. $\mathcal{A}_y^e = \mathcal{Z}_y^e \cup \mathcal{Z}_y^{e' \to e}$, where $\mathcal{Z}_y^{e' \to e}$ is as defined in (6.6). Then*

$$L_y(\mathcal{A}_y^e, \mathbf{W}^e, \mathbf{b}^e) - L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e) \geq \frac{L_y(\mathcal{Z}_y^{e'}, \mathbf{W}^{e'}, \mathbf{b}^{e'}) - L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e)}{2}, \tag{6.13}$$

where ($\mathbf{W}^e, \mathbf{b}^e$) is the classifier of (6.10).

Proof. From (6.9),

$$L_y(\mathcal{A}_y^e, \mathbf{W}^e, \mathbf{b}^e) = \frac{1}{2} L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e) + \frac{1}{2} L_y(\mathcal{Z}_y^{e' \to e}, \mathbf{W}^e, \mathbf{b}^e) \tag{6.14}$$

and since

$$\begin{align} L_y(\mathcal{Z}_y^{e' \to e}, \mathbf{W}^e, \mathbf{b}^e) &= L_y(\{\mathbf{z}_i^{e'} - \bar{\mathbf{z}}^{e'} + \bar{\mathbf{z}}^e\}, \mathbf{W}^e, \mathbf{b}^e) \tag{6.15} \\ &= -\frac{1}{|\mathcal{Z}_y^{e'}|} \Sigma_i \log \nu_y(\mathbf{W}^e \mathbf{z}_i^{e'} - \mathbf{W}^e \bar{\mathbf{z}}^{e'} + \mathbf{W}^e \bar{\mathbf{z}}^{e'} + \mathbf{b}^e) \tag{6.16} \\ &= L(\mathcal{Z}_y^{e'}, \mathbf{W}^e, \mathbf{b}^e - \mathbf{W}^e \bar{\mathbf{z}}^{e'} + \mathbf{W}^e \bar{\mathbf{z}}^e) \tag{6.17} \end{align}$$

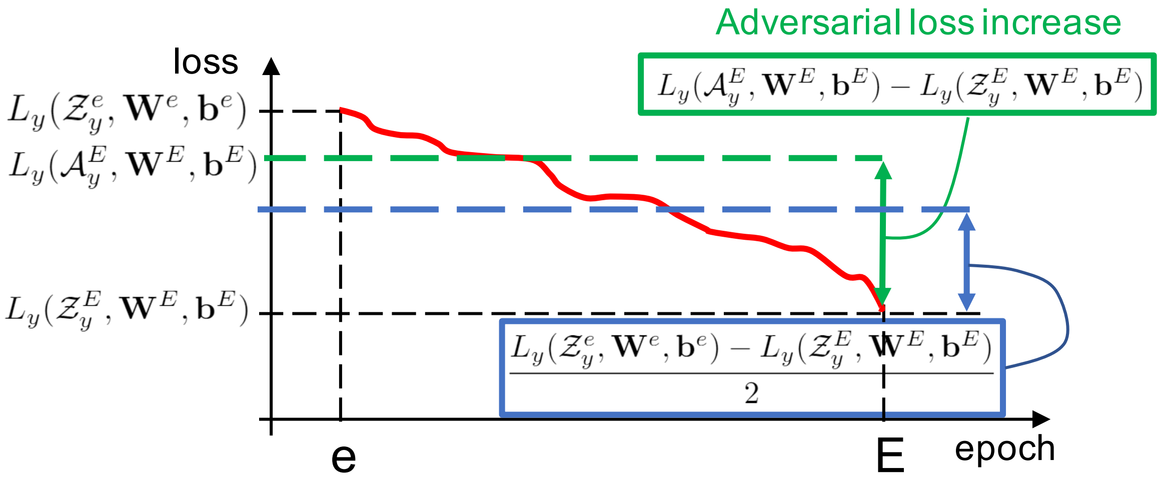

Adversarial loss increase
$L_y(\mathcal{A}_y^E, \mathbf{W}^E, \mathbf{b}^E) - L_y(\mathcal{Z}_y^E, \mathbf{W}^E, \mathbf{b}^E)$
loss
$L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e)$
$L_y(\mathcal{A}_y^E, \mathbf{W}^E, \mathbf{b}^E)$
$L_y(\mathcal{Z}_y^E, \mathbf{W}^E, \mathbf{b}^E)$
$\frac{L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e) - L_y(\mathcal{Z}_y^E, \mathbf{W}^E, \mathbf{b}^E)}{2}$
e
E
epoch

Figure 6.3: Adversarial nature of EMANATE. The loss increase, between two epochs, due to feature augmentation by EMANATE is never smaller than half of the training gain (loss decrease) between them.

it follows from (6.10) that

$$L_y(\mathcal{Z}_y^{e' \to e}, \mathbf{W}^e, \mathbf{b}^e) \geq L(\mathcal{Z}_y^{e'}, \mathbf{W}^{e'}, \mathbf{b}^{e'}) \tag{6.18}$$

and

$$L_y(\mathcal{A}_y^e, \mathbf{W}^e, \mathbf{b}^e) \geq \frac{L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e) + L(\mathcal{Z}_y^{e'}, \mathbf{W}^{e'}, \mathbf{b}^{e'})}{2} \tag{6.19}$$

from which the lemma follows. □

The lemma shows that the adversarial increase of the loss due to the augmentation ($L_y(\mathcal{A}_y^e, \mathbf{W}^e, \mathbf{b}^e) - L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e)$) is at least half of decrease in the loss of the trained classifier between epochs e' (loss $L(\mathcal{Z}_y^{e'}, \mathbf{W}^{e'}, \mathbf{b}^{e'})$) and e (loss $L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e)$), *i.e.* half of what has been gained by training the classifier from epochs e' to e. This is illustrated in Figure 6.3 and leads to the following theorem.

Theorem 1. *EMANATE is an adversarial feature augmentation strategy for any convergent training scheme, i.e. whenever* $L_y(\mathcal{Z}_y^{e'}, \mathbf{W}^{e'}, \mathbf{b}^{e'}) > L_y(\mathcal{Z}_y^e, \mathbf{W}^e, \mathbf{b}^e) \forall e' < e$.

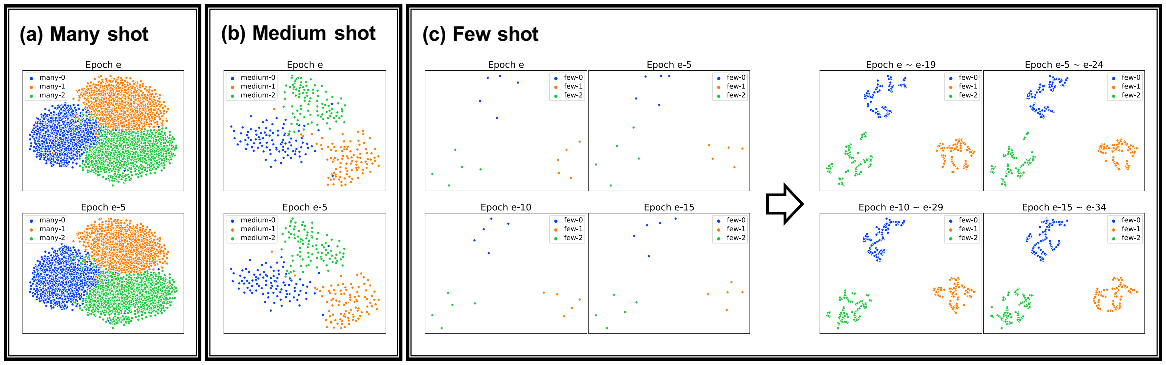


Figure 6.4: t-SNE visualizations of feature snapshots at different epochs. Many-shot (a) and medium-shot (b) features compose a well-defined geometry that does not change along epochs. Due to the scarcity of data, few-shot features (c, left) fail to hold a consistent geometry along epochs. After augmentation with EMANATE (c, right), the features have more variety and the geometry changes less among epochs.

Since successful training requires a convergent training scheme, EMANATE is an adversarial feature augmentation technique for most training procedures of practical interest.

6.3.4 Assembling feature trails

So far, we have considered the augmentation of $\mathcal{Z}_y^e$ with the transferred snapshot $\mathcal{Z}_y^{e'\to e}$. The augmentation can obviously be repeated for several transferred snapshots e', in order to meet any target number n_B of samples per class at epoch e. This is done as follows. Consider a class j with n_j samples. The features in $\mathcal{Z}_j^e$ are first selected. If there is a differential to n_B, the transferred snapshot $\mathcal{Z}_j^{e-1\to e}$ is selected next. The procedure is repeated until number of the feature vectors reaches n_B. If the addition of the final set places the feature cardinality above n_B, the necessary number of feature vectors is sampled randomly. The augmented set of features that emanate from class j at epoch e is then

$$\mathcal{A}_j^e = \bigcup_{k=0}^{K_j-2} \mathcal{Z}_j^{e-k\to e} \bigcup \tilde{\mathcal{Z}}_j^{e-(K_j-1)\to e}, \tag{6.20}$$

where $K_j = \left\lceil \frac{n_B}{n_j} \right\rceil$, and $\tilde{\mathcal{Z}}_j^{e-K_j-1\to e}$ is a random sample from $\mathcal{Z}_j^{e-K_j-1\to e}$ of size $n_B - K_j n_j$. The complete training set of epoch e is $\mathcal{A}^e = \cup_{j=1}^{C} \mathcal{A}_j^e$.

The number of snapshots in $\mathcal{A}_j^e$ depends heavily on the number of examples n_j of the class. As shown in Figure 6.2 (a, right), many-shot classes use a single snapshot, medium-shot classes require snapshots from a few epochs, and few-shot classes require many snapshots to assemble enough training features. However, in all cases, because all feature vectors are already computed during the optimization of the embedding, the only computation required is the mean alignment of (6.5). This is negligible when compared to the back-propagation computations, making EMANATE nearly computation free, if the necessary snapshots are kept in memory. In fact, it only necessary to keep in memory the snapshots of classes with $n_j < n_B$. Furthermore, the number K_j of snapshots to be stored adapts to n_j, as shown in (6.20). The larger the class, the fewer snapshots are required. In summary, EMANATE has no computational overhead and adapts the memory requirements to the class cardinalities, never requiring more than n_B examples per class. This is the complexity of class-balanced sampling.

Figure 6.4 shows a t-SNE [79] visualization of training set snapshots collected at different epochs. While the geometry of many- and medium shot classes (Figure 6.4(a,b)) is fairly stable across epochs, that of few-shot classes (Figure 6.4(c) left) can change significantly, due to data scarcity. EMANATE produces larger clusters with more stable geometry, enabling a more robust training set for the classifier.

6.4 Breadcrumbs

In this section, we investigate two sampling mechanisms based on EMANATE, which are denoted as Breadcrumb sampling. The two mechanisms differ in how the sets $\mathcal{A}_j^e$ are collected. In both cases, the two stage training procedure of [50] is adopted. In the first stage, the feature extractor $f(\mathbf{x};\theta)$ and the classifier $\nu(\mathbf{W}\mathbf{x}+b)$ are trained with image balanced sampling. The

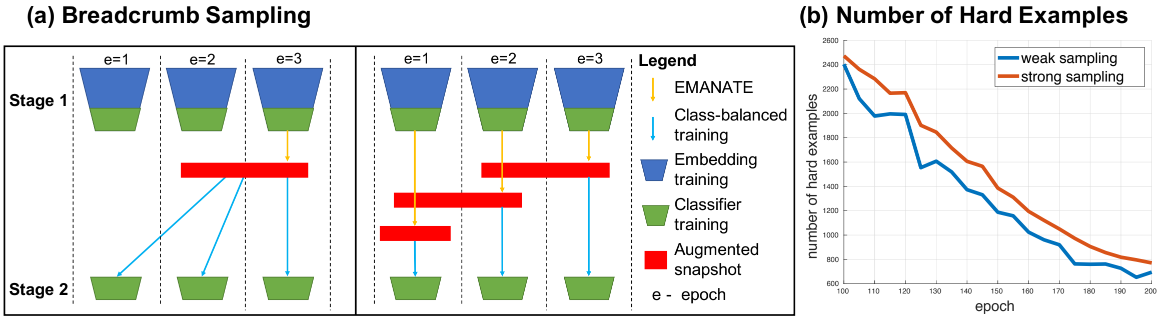


Figure 6.5: (a) Breadcrumb Sampling relies on EMANATE to collect augmented snapshots $\mathcal{A}_j^e$ (in red) in a first stage, when the embedding is trained with image-balanced sampling. In a second stage, the classifier is learned with class-balanced training based on these snapshots. In this example $E = 3$ and snapshots have length $K_j = 2$ (a single class is shown for simplicity). Left: Weak Breadcrumb Sampling only uses snapshots collected at the end of stage 1. Right: Strong Breadcrumb Sampling uses snapshots collected throughout stage 1 training. (b) Number of hard examples (loss larger than 5) in few-shot classes during training, for ResNet-10 on ImageNet-LT. Strong Breadcrumb sampling increases the number of hard examples during training, compared to the weak one. The plot starts at epoch 100 because early epochs have too many hard examples and dominate the scale.

sets $\mathcal{A}_j^e, e = \{1, \ldots, E\}$ of class snapshots are collected at each epoch of this stage. In the second stage, the feature extractor $f(\mathbf{x};\theta)$ is kept fixed and the classifier $\nu(\mathbf{W}\mathbf{x}+b)$ retrained using these sets. As shown in Figure 6.5, the two augmentation schemes differ in the classifier update step.

6.4.1 Weak Breadcrumb Sampling.

In the first approach, EMANATE is only applied *after convergence* of the first stage training. That is, only the sets $\mathcal{A}_j^E$ assembled in the *final* epoch E of the first stage are used to retrain the classifier in the second stage. This is illustrated in the left of Figure 6.5, for the case where $E = 3$ and augmentation sets span two epochs. We refer to this sampling technique as *Weak Breadcrumb Sampling*, since all snapshots emanate from the feature set produced by the optimal embedding $f(\mathbf{x}, \theta^E)$. While this creates some diversity, feature snapshots from neighboring epochs are likely to be similar. This makes the sampling technique less adversarial and therefore "weak".

Table 6.1: Ablation of Breadcrumb components, on the ImageNet-LT. For many-shot $t > 100$, for medium-shot $t \in (20, 100]$, and for few-shot $t \leq 20$, where t is the number of training samples.

Method	Overall	Many-Shot	Medium-Shot	Few-Shot
Decoupling [50]	41.4	51.8	38.8	21.5
+ back-tracking	41.2	50.4	38.5	23.8
+ class-specific	41.3	50.8	38.1	24.6
Weak Breadcrumb	43.2	53.6	39.8	25.1
Strong Breadcrumb	**44.0**	53.7	**41.0**	**26.4**
Breadcrumb	**44.0**	53.7	**41.0**	**26.4**
Breadcrumb(var.)	43.9	**53.8**	40.8	26.0
Breadcrumb(agn.)	38.5	47.3	35.6	24.0

6.4.2 Strong Breadcrumb Sampling.

Strong Breadcrumb Sampling aims to increase feature diversity, so as to create a more adversarial data augmentation. Rather than the augmentation $\mathcal{A}_j^E$ of the final epoch E of the first stage training, *all* augmentations $\mathcal{A}_j^e, e = \{1, \ldots, E\}$ are saved in that stage, as illustrated in the right of Figure 6.5. The classifier retraining of the second stage is then run for E epochs, using the the feature trail sets $\mathcal{A}_j^e$ collected at epoch e of the first stage to train the classifier in epoch e of the second stage. Since each epoch of classifier training contains new data, increasing the difficulty of the classification task, this sampling method is more adversarial and therefore ”strong”. Since the classifier is trained on an evolving feature set $\mathcal{A}^e$, this setting yields a natural selection of the target number n_B of samples per class. To keep the pace of classifier training the same as the embedding training, the size of the dataset should be approximately the same, *i.e.* $n_B = \left\lceil \frac{1}{C} \sum_{j=1}^{C} n_j \right\rceil$. Figure 6.2(b), shows that Strong Breadcrumb Sampling increases the number of hard examples in few-shot classes per epoch, when compared to Weak Breadcrumb Sampling. This confirms that it is a more adversarial data augmentation strategy.

6.5 Experiments

In this section, we discuss an evaluation of Breadcrumb sampling.

6.5.1 Experimental Set-up

Datasets

We consider three long-tailed recognition datasets, ImageNet-LT [75], Places-LT [75] and iNatrualist18 [128]. ImageNet-LT is a long-tailed version of ImageNet [19] by sampling a subset following the Pareto distribution with power value $\alpha = 6$. It contains 115.8K images from 1000 categories, with class cardinality ranging from 5 to 1280. Places-LT is a long-tailed version of the Places dataset [149]. It contains 184.5K images from 365 categories with class cardinality in $[5, 4980]$. iNatrualist18 is a long-tailed dataset, which contains 437.5K images from 8141 categories with class cardinality in $[2, 1000]$. Following [75], we present classification accuracies for both the entire dataset and three groups of classes: *many shot* (more than 100 training samples), *medium shot* (between 20 and 100), and *few shot* (less than 20 training samples).

Baselines

Following [75], we consider three metric-learning baselines, based on the lifted [86], focal [68], and range [145] losses, and one state-of-the-art method, FSLwF [29], for learning without forgetting. We also include long-tailed recognition methods designed specifically for these datasets, OLTR [75] and Distill [141], plus the recent state of the art Decoupling method [50]. The model with standard random sampling and end-to-end training is denoted as the *Plain Model* for comparison.

Table 6.2: Results on ImageNet-LT and Places-LT. ResNet-10/152 are used for all methods. For many-shot $t > 100$, for medium-shot $t \in (20, 100]$, and for few-shot $t \leq 20$, where t is the number of training samples.

	ImageNet-LT, ResNet-10			
Method	Overall	Many-Shot	Medium-Shot	Few-Shot
Plain Model	23.5	41.1	14.9	3.6
Lifted Loss [86]	30.8	35.8	30.4	17.9
Focal Loss [68]	30.5	36.4	29.9	16.0
Range Loss [145]	30.7	35.8	30.3	17.6
FSLwF [29]	28.4	40.9	22.1	15.0
OLTR [75]	35.6	43.2	35.1	18.5
Distill [141]	38.8	47.0	37.9	19.2
Decoupling(cRT) [50]	41.4	51.8	38.8	21.5
Breadcrumb	**44.0**	**53.7**	**41.0**	**26.4**

	Places-LT, ResNet-152			
Method	Overall	Many-Shot	Medium-Shot	Few-Shot
Plain Model	27.2	**45.9**	22.4	0.36
Lifted Loss [86]	35.2	41.1	35.4	24.0
Focal Loss [68]	34.6	41.1	34.8	22.4
Range Loss [145]	35.1	41.1	35.4	23.2
FSLwF [29]	34.9	43.9	29.9	29.5
OLTR [75]	35.9	44.7	37.0	25.3
Distill [141]	36.2	39.3	39.6	24.2
Decoupling(cRT) [50]	37.9	37.8	40.7	31.8
Breadcrumb	**39.3**	40.6	**41.0**	**33.4**

Training Details

ResNet-10 and ResNeXt-50 [38, 142] are used on ImageNet-LT; ResNet-152 is used on Places-LT; and ResNet-50 is used on iNatrualist18. The model is trained with SGD, using momentum 0.9, weight decay 0.0005, and a learning rate that cosine decays from 0.2 to 0. Each iteration uses class-balanced and random sampling mini-batches of size 512. One epoch is defined when the random sampling iterates over the entire training data. Under Strong Breadcrumb Sampling, class-balanced sampling is applied in the initial classifier training epochs, when there are not enough previous epochs to back-track. Codes are attached in supplementary.

Table 6.3: Results on ImageNet-LT, ResNeXt-50. For many-shot $t > 100$, for medium-shot $t \in (20, 100]$, and for few-shot $t \leq 20$, where t is the number of training samples.

Method	Overall	Many-Shot	Medium-Shot	Few-Shot
OLTR [75]	41.9	51.0	40.8	20.8
Decoupling(NCM) [50]	47.3	56.6	45.3	28.1
Decoupling(cRT) [50]	49.6	61.8	46.2	27.4
Decoupling(τ) [50]	49.4	59.1	46.9	30.7
Decoupling(LWS) [50]	49.9	60.2	47.2	30.3
Breadcrumb	**51.0**	**62.9**	**47.2**	**30.9**

6.5.2 Ablation Study

Several ablations were performed to study the effectiveness of the various components of Breadcrumb. In this study, all models are trained and evaluated on the training and test set of ImageNet-LT, respectively, using a ResNet-10 backbone.

Component Ablation

Starting from the baseline Decoupling (cRT) [50] method, we incrementally add feature back-tracking, class-specific augmentation, class alignment (leading to Weak Breadcrumb Sampling), and Strong Breadcrumb Sampling. Results are shown in Table 6.1. When only back-tracking is applied, all snapshots are collected from the last 10 epochs of image-balanced training (first stage), and the classifier trained (in the second stage) using this feature set and class-balanced sampling. No class alignment is applied. Compared to the baseline, back-tracking gives a reasonable gain on few-shot classes but harms many-shot performance. This can be explained by the fact that, for many-shot classes, features from the final epoch are replaced by those from prior epochs. Since the corresponding embeddings are sub-optimal, the augmented features are inferior to the final ones. This, however, is not the case in few-shot, where augmented features replace *duplicated* features.

The combination of back-tracking and class-specific augmentation, where different classes

Table 6.4: Results on the iNaturalist2018. All methods are implemented with ResNet-50.

Method	Accuracy
CB-Focal [17]	61.1
LDAM+DRW [7]	68.0
Decoupling(cRT) [50]	68.2
Decoupling(τ) [50]	69.3
Decoupling(LWS) [50]	69.5
Breadcrumb	**70.3**

have different back-tracking lengths, is denoted as "+ class-specific" in Table 6.1. Surprisingly, without class alignment, the performance on many-shot does not improve, even though no augmented features are introduced into those classes. We believe this is due to the fact that when few-shot features are augmented without alignment, those augmented features take up position in feature space that should not be assigned to them. This decreases the accuracy of many-shot classes. When class-alignment is applied (Weak Breadcrumb Sampling) we observe an improvement over all class partitions, with gains of 1.8% (Many), 1% (Medium), and $3,6\%$ (Few-Shot) and an overall improvement of 1.8% over the baseline. Finally, Strong Breadcrumb Sampling enables another 0.8% overall gain, for a total gain of 2.6% over the baseline.

Class Alignment Ablation

Since alignment makes a significant difference, we considered three different alignment choices. In Sec 6.3.2, only class-specific mean alignment is presented. It is also possible to align the feature variances. This is denoted as Breadcrumb(var.) in Table 6.1 and has a negligible difference. Hence, we only apply mean alignment unless otherwise noted. Another possibility is class-agnostic alignment, where only one mean is computed over all classes. This is listed as Breadcrumb(agn.) in Table 6.1. Its poor performance implies that class-agnostic alignment cannot fully eliminate the differences between epochs.

6.5.3 Comparison to the state of the art

Table 6.2 presents a final comparison to the methods in the literature on ImageNet-LT, using a ResNet-10, and Places-LT, using a ResNet-152. In these experiments we use Strong Breadcrumb Sampling, which is shown to outperform all other methods on both datasets. It achieves the best performance on 5 of the 6 partitions and is always better than the next overall best performer (Decoupling(cRT)). It is only outperformed by the Plain Model on the Many-Shot split of Places-LT, where this model severely over-fits to the Many-Shot classes, basically ignoring the Few-Shot ones, and achieving overall performance 12.1% weaker than Breadcrumb. Compared to the best models Breadcrumb also achieves significant gains on few-shot classes, especially on ImageNet-LT, where it beats the next best method by 4.9%. This suggests that previous methods over-fit for few-shot classes, a problem that is mitigated by the introduction of EMANATE and Strong Breadcrumb Sampling. Table 6.3 shows that these results are fairly insensitive to the backbone network. Breadcrumb achieves the best overall performance and the best performance on all partitions with a ResNeXt-50 backbone. Finally Table 6.4 shows that Breadcrumb again achieves the overall best results for a ResNet-50 on iNaturalist.

6.6 Conclusion

This work discussed the long-tailed recognition problem. A new augmentation framework, Breadcrumb, was proposed to increase feature variety and classifier robustness. Breadcrumb is based on EMANATE, a feature back-tracking procedure that aligns features vectors produced across several epochs of embedding training, to compose a class-balanced feature set for training the classifier at the top of the network. It is inspired by the the recent success of class-balanced training schemes. However, unlike previous schemes, it is shown to be an adversarial sampling scheme, a property that encourages better generalization. A comparison of two sampling schemes based on EMANATE confirmed this property, resulting in best performance for the Strong

Breadcrumb Sampling technique, where feature snapshots are collected while the embedding is evolving. Breadcrumb was shown to achieve state-of-the-art performance on three popular long-tailed datasets with different CNN backbones. Furthermore, Breadcrumb introduces no extra model, which means that it adds no computational overhead or convergence issues to the baseline model.

Chapter 6 is, in full, based on the material as it appears in the submission of “Breadcrumbs: Adversarial Class-Balanced Sampling for Long-tailed Recognition”, Bo Liu, Haoxiang Li, Hao Kang, Gang Hua, Nuno Vasconcelos, in *IEEE International Conference on Computer Vision* (ICCV), 2021. The dissertation author was the primary investigator and author of this material.

Chapter 7

Semi-supervised Long-tailed Recognition using Alternate Sampling

7.1 Introduction

Large-scale datasets, which contain sufficient data in each class, has been a major factor to the success of modern deep learning models for computer vision tasks, such as object recognition. These datasets are usually carefully curated and balanced to have an uniform data distribution over all classes. This balanced data distribution favors model training but could be impractical in many real world applications, where the frequency of samples from different classes can be imbalanced, leading to a long-tailed data distribution. As shown in Figure 7.1(b), several highly populated classes take up most of the labeled samples, and some of the classes only have very few samples during training.

The long-tailed recognition problem has been widely studied in the literature. One major challenge in this setting [75, 50, 150] to deep learning model training is the tendency of under-fitting in less-populated classes. The root causes of this under-fitting are the imbalanced training data distribution as well as the scarcity of data samples in the tail classes.

More specifically, with an imbalanced training data distribution, when several head classes take up most of the training samples, tail classes contribute little in the training loss. The model is such that biased towards head classes. Prior works [68, 7, 50, 150] tried to mitigate the issue by re-sampling the training data to be a balanced distribution or calibrating the sample weights in calculating the loss. However, still the scarcity of tail class data samples limits the intra-class variations and overall recognition accuracy. Methods focusing on few-shot learning have been introduced to address this problem through data augmentation and data synthesis [132, 36, 70].

In this work, we resort to a different path to leverage massive unlabeled real data in training to help improve the long-tailed recognition accuracy. Since data collection is much cheaper and accessible comparing to data annotation, additional unlabeled real data could readily be available in many real-world scenarios. This semi-supervised learning setting has been intensively studied in the literature [60, 95, 121, 5, 111]. However, as shown in Figure 7.1(a), when we carefully

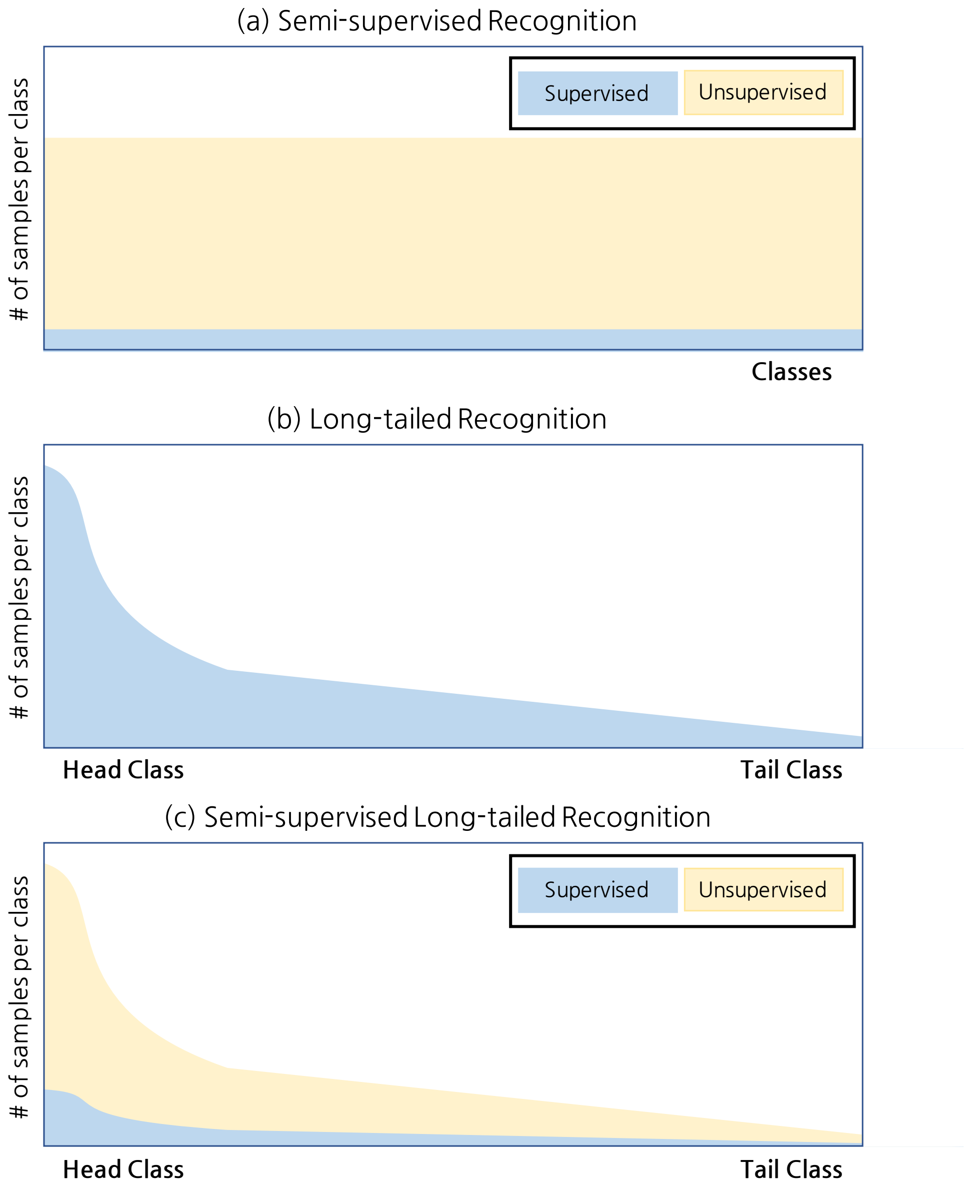


Figure 7.1: Comparison of different recognition paradigms: a) statistics of CIFAR-10 when used as a Semi-supervised Recognition benchmark; b) typical data distribution over classes in long-tailed recognition; c) the proposed Semi-supervised long-tail recognition setting, in which both labeled and unlabeled subsets follow the same underlying long-tailed data distribution.

look at the data distribution of the widely used benchmarks, we observe well-balanced labeled subset and unlabeled subset. As discussed above, the manually curated balanced distribution, can lead to a gap to real-world scenarios. This is especially true in unlabeled data. Without labels, people have no way to balance the data among classes.

In this paper, we propose a more realistic and challenging setting, namely semi-supervised long-tailed recognition. As shown in Figure 7.1(c), we assume a long-tailed data distribution of the overall dataset and both the labeled and unlabeled subsets of training data follow the same underlying long-tailed data distribution. This setting generally resembles a realistic data collection and annotation workflow. After collecting the raw data, one has no knowledge of its class distribution before annotation. As it is expensive to annotate the full corpus, a common practice is to randomly sample a subset for annotation under a given labeling budget. When the raw data follows a long-tailed class distribution, we should expect the same in the labeled subset.

While this new recognition paradigm shares the challenges in both semi-supervised learning and long-tailed recognition, there is no readily naive solution to it. Methods in long-tailed recognition rely on class labels to achieve balanced training, which are not available in the unlabeled portion in the semi-supervised long-tailed recognition. Prior semi-supervised methods without considering the long-tailed distribution could fail as well.

Taking one of the competitive baseline methods for example, Yang *et al.* [144] proposed to firstly train a recognition model with the labeled subset to generate pseudo labels for the unlabeled subset, then the model is fine-tuned with the full training dataset. However, when the labeled subset follows a long-tailed distribution, the pseudo labels are much less accurate for tail classes than head classes. As a result, the overall pseudo labels quality could be too bad to leverage (See Section 7.4.5 for results in CIFAR-10-SSLT and ImageNet-SSLT).

To address the semi-supervised long-tailed recognition problem, we present a method designed specifically for this setting. We bring the successful class-balanced sampling strategy and combined it with model decoupling in an alternate learning framework to overcome the difficulty of balancing unlabeled training data.

Inspired by [50], we decouple the recognition model into a feature embedding and a classifier, and train them with random sampling and class-balanced sampling respectively. As we are targeting at a semi-supervised setting, the classifier is only trained on labeled data to get around the difficulty of applying correctly class-balanced sampling on unlabeled data, aligning with the intuition that the classifier needs more robust supervision than the feature embedding.

After that, with the proposed alternative learning framework, we improve model by updating the feature embedding and the classifier iteratively. We assign pseudo labels with the up-to-date classifier and observed gradually improved accuracy of pseudo labels over iterations. The pseudo labels are then incorporated in fine-tuning the feature embedding with a regularization term to limit its potential negative impacts. Similar iterative design has been proposed in semi-supervised learning literature [60, 121] but important implementation details differ.

To summarize, in this paper, 1) we resort to semi-supervised learning to help improve long-tailed recognition accuracy and identify practical gap of current semi-supervised recognition datasets due to their well-balanced unlabeled subset; 2) we propose a new recognition paradigm named semi-supervised long-tailed recognition better resembling real-world data collection and annotation workflow; 3) we propose a new alternative sampling method to address the semi-supervised long-tailed recognition and demonstrate significant improvements on several benchmarks.

7.2 Related Work

Long-tailed Recognition

Long-tailed recognition has been recently studied a lot [134, 86, 68, 145, 75, 133]. Several approaches have been proposed, including metric learning [86, 145], loss weighting [68], and meta-learning [133]. Some methods design dedicated loss functions to mitigate the data imbalanced problem. For example, lift loss [86] introduces margins between many training samples. Range loss [145] encourages data from the same class to be close and different classes to be far away in the embedding space. The focal loss [68] dynamically balances weights of positive, hard negative, and easy negative samples. As reported by Liu *et al.* [75], when applied to long-tailed recognition, many of these methods improved accuracy of the few-shot group, but at the cost of lower accuracy over the many-shot classes.

Other methods, *e.g.* LDAM-DRW [7] replace cross-entropy loss with LDAM loss. This adds a calibration factor to the original cross-entropy loss. When combined with loss re-weighting, it improves the accuracy in all splits in long-tailed recognition. However, it can not be easily generalized to semi-supervised learning. Because both the calibration factor and the loss weight are calculated based on the number of samples of each class.

In face recognition and person re-identification, the datasets are mostly with long-tailed distribution. LEAP [70] augmented data samples from tail (few-shot) classes by transferring intra-class variations from head (many-shot) classes. Instead of data augmentation, we introduce unsupervised data to improve the performance of long-tailed recognition.

A recent work [144] rethinks the value of labels in imbalance learning. As part of the discussion, semi-supervised learning is included. However, only the basic pseudo label solution and simple datasets, such as CIFAR and SVHN, are discussed.

More recent works [50, 150] with improved long-tailed recognition share the observation that feature embedding and the classifier should be trained with different sampling strategies. In this work, we adopt our method on this observation to learn the feature embedding model with random sampling and train the classifier with class-balanced sampling. This design is further closely compatible with semi-supervised learning under alternate learning.

Semi-supervised Learning

Semi-supervised learning has been extensively discussed in recognition discipline [60, 95, 121]. One common observation is to optimize the traditional cross-entropy loss together with a regularization term that regulates the perturbation consistency of unlabelled data.

Ladder net [95] is introduced to minimise the reconstruction loss between the network outputs from a given sample and its perturbation. It is then simplified in [60] as two temporal modules: Π-Model and Temporal Ensembling. The Temporal Ensembling encourages the output of the network from unlabeled data to be similar to its counterpart from previous training epoch. More recently, Mean Teacher [121] extends it by assembling along training. Instead of storing previous predictions, they assemble a Teacher model by calculating the moving average of the training network, *i.e.* the Student. The Teacher is then used to provide the consistency of predictions to the Student.

In addition to that, MA-DNN [11] introduces a memory module to maintain the category prototypes and provide regularization for learning with unlabeled data. Label propagation [66] is also considered with the help of label graph. More recently, Mixmatch [5] and Fixmatch [111] improve the performance by introducing powerful data augmentations and perturbation consistencies.

All the semi-supervised methods above do not separate labeled data during semi-supervised training. In fact, it is beneficial to combine labeled data and unlabeled data in a certain proportion [60, 121]. However, without further knowledge, we have no insight how to deal with this combination when long-tailed distribution is included. Furthermore, long-tailed learning methods require calibration or re-sampling based on the class distribution. This combination of labeled and unlabeled data makes the distribution unstable. In result, this is not suitable for long-tailed recognition.

Recently, Salsa [98] proposes to decouple the supervised learning from semi-supervised training. Our method follows the alternate training scheme from it, because it is surprisingly compatible with long-tailed learning. In practice, our method differs from Salsa in the following aspects.

First, we adopt class-balanced sampling in supervised learning to deal with the long-tailed distribution. Second, we use supervised learning instead of self-supervised learning as initialization. We find that self-supervised learning results in inferior performance in long-tailed scenario. Third, the re-initialization is not needed. Because our initialization is already from

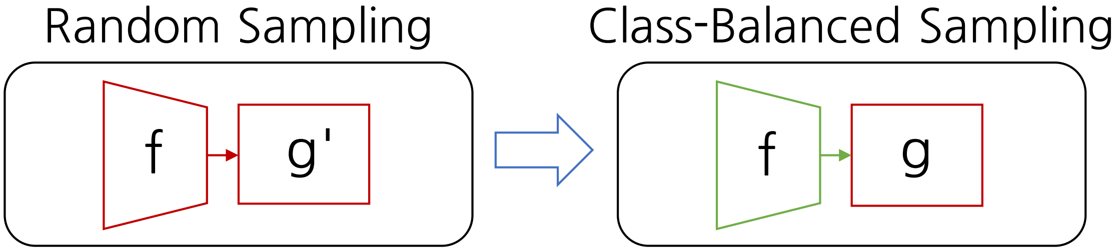


Figure 7.2: Initialization procedure. A recognition model is first trained with random sampling. After that the feature embedding is used to train a new classifier with class-balanced sampling. In the diagram, CNN components that are updated during training are highlighted in red.

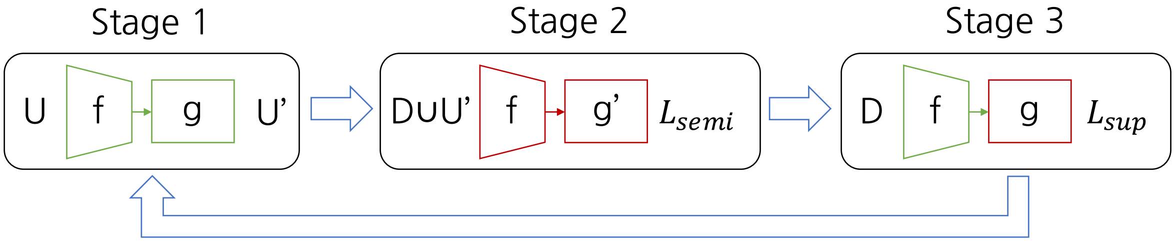


Figure 7.3: Diagram of alternate learning. CNN modules in green line is only used in forwarding. Those in red are fine-tuned with the corresponding loss. In Stage 1, samples from $\mathcal{U}$ are forwarded through f and g. $\mathcal{U}'$ consists of samples from $\mathcal{U}$, and pseudo labels acquired from g. In Stage 2, f and g' are trained on the combination of $\mathcal{D}$ and $\mathcal{U}'$. In Stage 3, only the classifier g is trained. f is fixed and only used in forwarding.

supervised learning, there is not a specific starting point to re-initialize the model. In fact, this enhances the soft constraint between the two stages in [98].

With the models continuously optimized along alternate learning, our method achieves superior performance while maintains the same amount of training epochs as fine-tuning simply on pseudo labels.

7.3 Method

In this section, we will introduce the proposed method to semi-supervised long-tailed recognition. The semi-supervised long-tailed recognition problem is first defined, and some notations are clarified. The decoupling strategy of long-tailed recognition is then discussed. This is also the initialization phase of our method. After that, the alternate learning scheme with 3 stages is fully discussed. The proposed method is outlined in Algorithm 1.

7.3.1 Semi-supervised Long-tailed Recognition

We start by defining the semi-supervised long-tailed recognition problem. Consider an image recognition problem with a labeled training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i); i = 1, \dots, N\}$, where x_i is an example and $y_i \in \{1, \dots, C\}$ its label, where C is the number of classes. For semi-supervised learning, there is also an unsupervised training subset $\mathcal{U} = \{\mathbf{x}_i; i = 1, \dots, M\}$.

Although the labels of data in $\mathcal{U}$ are not available, every sample has its label from $\{1, \dots, C\}$. For class j, we have n_j samples from $\mathcal{D}$ and m_j samples from $\mathcal{U}$. With the assumption that supervised and unsupervised data follow the same distribution, we have the fact

$$\frac{n_j}{N} = \frac{m_j}{M}, \quad \forall j. \tag{7.1}$$

The testing set, on the other hand, in order to evaluate the performance on every class without bias, is balanced sampled on all classes in $\{1, \dots, C\}$.

7.3.2 Model Decoupling and Data Sampling

A CNN model combines a feature embedding $\mathbf{z} = f(\mathbf{x}; \theta) \in \mathbb{R}^d$, and a classifier $g(\mathbf{z}) \in [0,1]^C$. Embedding $f(\mathbf{x}; \theta)$ is implemented by several convolutional layers of parameters θ. The classifier operates on the embedding to produce a class prediction $\hat{y} = \arg\max_i g_i(\mathbf{z})$. In this work, we adopt the popular linear classifier $g(\mathbf{z}) = \nu(\mathbf{W}\mathbf{x} + \mathbf{b})$, where ν is the softmax function.

Standard (random sampling) training of the CNN lies on mini-batch SGD, where each batch is randomly sampled from training data. A class j of n_j training examples has probability $\frac{n_j}{N}$ of being represented in the batch. Without loss of generality, we assume classes sorted by decreasing cardinality, *i.e.* $n_i \leq n_j, \forall i > j$. In the long-tailed setting, where $n_1 \gg n_C$, the model is not fully trained on classes of large index j (tail classes) and under-fits. This can be avoided with recourse to non-uniform sampling strategies, the most popular of which is class-balanced sampling. This samples each class with probability $\frac{1}{C}$, over-sampling tail classes.

[50, 150] shows that while classifier benefits from class-balanced sampling, feature embedding is more robust in random sampling. Practically, [50] achieves this by decoupling the training into two stages, and train the feature embedding with random sampling in the first stage, and classifier the second with class-balanced sampling.

7.3.3 Initialization

The initialization of the proposed method follows the decoupling from [50]. The two-stage initialization is illustrated in Figure 7.2. A CNN model is first trained with random sampling. A feature embedding $\mathbf{z} = f(\mathbf{x}; \theta) \in \mathbb{R}^d$, and a classifier $g'(\mathbf{z}) \in [0,1]^C$ are acquired. After convergence, the classifier is re-initialized and trained with class-balanced sampling, with the feature embedding fixed. This results in a class-balanced classifier $g(\mathbf{z}) \in [0,1]^C$. Both the feature embedding and the classifier are trained on the supervised training subset $\mathcal{D}$.

7.3.4 Alternate Learning

After obtaining an initialized model, most semi-supervised learning methods fine-tune the model on a combination of supervised and unsupervised samples. This is, however, incompatible with our long-tailed recognition model. When applied on unsupervised data, we have no ground truth for class-balanced sampling. One can make a sacrifice by relying on pseudo labels assigned by the initialized model. But the effectiveness will depend on the accuracy of pseudo labels.

It is even worse when considering the fact that long-tailed models usually have better performance on highly populated classes and worse on few-shot classes. Class-balanced sampling over-samples few-shot classes, while down-samples many-shot. This means, in general, the worse

Algorithm 1 Alternate Learning for Semi-supervised Long-tailed Recognition

```
Initialization:
        Input: supervised training subset $\mathcal{D}$
        Output: feature embedding $f$, random-trained classifier $g'$, class-balanced classifier $g$
        A CNN model is trained with random sampling. The feature embedding $f$ is then used to train a new classifier with class-balanced sampling.
Alternate Learning:
for $i = 1, \ldots, N$ do
    Stage 1: Label assignment
        $f$ and $g$ are used to assign labels to all samples in the unlabeled subset $\mathcal{U}$. The set U combined with assigned labels is $\mathcal{U}'$
    Stage 2: Semi-supervised training
        Fine-tune $f$ and $g'$ on the set $\mathcal{D} \cup \mathcal{U}'$. Random Sampling is used to minimize the semi-supervised loss $\mathcal{L}_{semi}$
    Stage 3: Supervised training
        Fine-tune $g$ on the set $\mathcal{D}$. Class-balanced Sampling is used to minimize the supervised loss $\mathcal{L}_{sup}$. $f$ is used to calculate features, but is not fine-tuned.
```

part of pseudo labels contributes more to the training loss than it should be, while the better part contributes less.

Another difficulty is the model compatibility when combining the long-tailed model to semi-supervised learning methods. Many semi-supervised learning methods evolve the model and pseudo labels at the same time. For example, Mean Teacher [121] assembles the teacher model by moving average and trains the student with consistency loss. When it comes to long-tailed model, it is not clear when we should update the feature embedding or classifier. And it is also difficult to incorporate both random and class-balanced sampling.

Inspired by [98], which separates supervised learning apart from semi-supervised learning, we propose an alternate learning scheme. The supervised training on data $\mathcal{D}$, and semi-supervised training on data $\mathcal{D} \cup \mathcal{U}$ are carried out in an alternate fashion together with model decoupling and different data sampling strategies.

In practice, after initialization, we have a feature embedding $\mathbf{z} = f(\mathbf{x};\theta)$, a classifier $g'(\mathbf{z})$ trained with random sampling, and a classifier $g(\mathbf{z})$ trained with class-balanced sampling. In [50], only $g(\mathbf{z})$ is used in testing. However, we keep the randomly trained classifier $g'(\mathbf{z})$ for further usage. The training scheme iterates among 3 stages for N loops, which are shown in Figure 7.3.

Stage 1: Label Assignment

In this stage, pseudo labels are assigned for the unsupervised subset $\mathcal{U}$. The feature embedding $f(\mathbf{x};\theta)$ and class-balanced classifier $g(\mathbf{z})$ are used. The choice of classifier is equivalent to the long-tailed model when tested for better overall accuracy. The unsupervised subset with pseudo labels is $\hat{\mathcal{U}} = \{(\mathbf{x}_i, \hat{\mathbf{y}}_i); i = 1, \ldots, M\}$, where $\hat{\mathbf{y}}_i$ are pseudo labels.

Stage 2: Semi-supervised Training

After label assignment, we have pseudo labels for all unsupervised data. The model is the fine-tuned on the combination of true and pseudo labels, *i.e.* on $\mathcal{D} \cup \hat{\mathcal{U}}$. In this stage, random sampling is used to update the feature embedding $f(\mathbf{x};\theta)$ and the randomly-trained classifier $g'(\mathbf{z})$. The classification is optimized by cross-entropy loss:

$$\mathcal{L}_{CE} = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D} \cup \hat{\mathcal{U}}} -\log g'_{y_i}(f(\mathbf{x}_i;\theta)), \tag{7.2}$$

where g'_{y_i} is the y_i-th element of g'.

In semi-supervised learning literature, a regularization loss is usually applied to maintain the consistency for unlabeled data. This consistency loss captures the fact that data points in the neighborhood usually share the same label. We adopt this idea and implement the temporal consistency from [60]. In practice, the class probabilities are acquired from g'. Given the class probability p^{e-1} from epoch $e-1$, and the class probability p^e from epoch e, the loss is KL-divergence between the two.

$$\mathcal{L}_{consist} = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D} \cup \hat{\mathcal{U}}} \sum_j p_j^{e-1} \log \frac{p_j^{e-1}}{p_j^e}, \tag{7.3}$$

where p_j^{e-1} and p_j^e are the j-th element of p^{e-1} and p^e respectively.

Overall, the semi-supervised learning loss is the combination of the two.

$$\mathcal{L}_{semi} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{consist}. \tag{7.4}$$

Stage 3: Supervised Training

We update the class-balanced classifier $g(\mathbf{z})$ based on the refined feature embedding, which is fine-tuned with semi-supervised learning in Stage 2. Specifically, the fine-tuning is applied with class-balanced sampling and only on the supervised subset $\mathcal{D}$. In this stage, only classifier is updated. The feature embedding is fixed and only used in forwarding. Given the class-balanced version of supervised subset $\mathcal{D}'$, the cross-entropy loss for classification is

$$\mathcal{L}_{sup} = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}'} - \log g_{y_i}(f(\mathbf{x}_i; \theta)), \tag{7.5}$$

where g_{y_i} is the y_i-th element of g.

7.3.5 Insight of the Design

Feature Embedding

Feature embedding is trained with random sampling and semi-supervised learning. This is consistent with long-tailed model in the sampling scheme. It also follows the fact that feature embedding is less prone to noisy labels. Actually, in self-supervised learning literature [30, 37, 9], the feature embedding can even be learned without labels.

Classifier

Classifier is learned with class-balance sampling and only supervised data. This is again the same as the supervised version. And by avoiding fitting the classifier on pseudo labels, we

Table 7.1: Results(Accuracy in %) on CIFAR-10-SSLT. ResNet-18 is used for all methods.

	Imbalance factor=100			
Method	Overall	Many-Shot	Medium-Shot	Few-Shot
LDAM-DRW (L) [7]	67.4	79.7	54.2	68.1
Pseudo-Label + L	69.6	69.7	55.1	80.2
Mean Teacher [121] + L	69.9	69.7	57.3	79.4
Decoupling (D) [50]	64.0	91.1	63.0	44.4
Pseudo-Label + D	68.9	92.7	70.8	49.8
Ours	71.3	89.5	67.7	60.2

	Imbalance factor=1000			
Method	Overall	Many-Shot	Medium-Shot	Few-Shot
LDAM-DRW (L) [7]	46.2	70.3	36.3	35.6
Pseudo-Label + L	48.4	74.0	39.3	36.0
Mean Teacher [121] + L	48.3	75.7	41.4	32.9
Decoupling (D) [50]	45.8	86.5	47.2	14.4
Pseudo-Label + D	46.5	89.0	47.0	14.2
Ours	66.7	84.4	69.4	51.4

prevent the wrong labels from propagating through the whole training process. Given the fact that the pseudo labels are provided by the classifier, if classifier is still optimize on those, wrong labels can be easily maintained in the fine-tuned version of the classifier.

Training the classifier only on labeled data also avoids the dilemma of class-balancing on unlabeled data. Without ground truth labels, class-balanced sampling can only rely on pseudo labels, which are not perfect. And the fact that pseudo labels have more errors on few-shot classes is specially not suitable for class-balanced sampling. Because when few-shot classes are over-sampled, those errors are also scaled up during training.

Table 7.2: Results(Accuracy in %) on ImageNet-SSLT. ResNet-18/50 are used for all methods. For many-shot $t > 100$, for medium-shot $t \in (10, 100]$, and for few-shot $t \leq 10$, where t is the number of labeled samples.

	ResNet-18			
Method	Overall	Many-Shot	Medium-Shot	Few-Shot
LDAM-DRW (L) [7]	21.3	42.6	27.0	8.6
Pseudo-Label + L	17.6	22.4	20.9	12.6
Mean Teacher [121] + L	21.3	41.8	28.1	7.6
Decoupling (D) [50]	24.8	53.9	31.1	8.7
Pseudo-Label + D	25.3	47.6	32.1	11.1
Ours	26.5	52.0	33.9	10.7
	ResNet-50			
Method	Overall	Many-Shot	Medium-Shot	Few-Shot
LDAM-DRW (L) [7]	24.9	51.2	31.1	9.9
Pseudo-Label + L	23.9	44.0	30.0	11.1
Mean Teacher [121] + L	25.6	49.1	31.8	11.7
Decoupling (D) [50]	27.2	58.5	34.2	9.8
Pseudo-Label + D	27.7	52.2	34.7	12.4
Ours	29.0	57.1	36.5	12.3

Table 7.3: Results(Accuracy in %) on iNaturalist2018-SSLT. ResNet-50 are used for all methods. For many-shot $t > 100$, for medium-shot $t \in (10, 100]$, and for few-shot $t \leq 10$, where t is the number of labeled samples.

Method	Overall	Many-Shot	Medium-Shot	Few-Shot
Decoupling [50]	27.9	54.1	41.7	24.8
Pseudo-Label + Decoupling	26.3	39.9	35.8	24.3
Ours	28.4	49.5	38.7	26.1

7.4 Experiments

7.4.1 Datasets

We manually curate two semi-supervised long-tailed recognition benchmarks.

CIFAR-10-SSLT

For easy comparison and ablation, we compose a lightweight semi-supervised long-tailed dataset based on CIFAR-10 [55]. Following [7], we randomly sample the training set of CIFAR-10 under an exponential function with imbalance ratios in $\{100, 1000\}$ (the ratio of most populated class to least populated). The unsupervised subset is collected from Tiny Images [124] following the strategy introduced in [144]. The class distribution of unlabeled data is always the same as the labeled one, with 5 times larger. For better description and comparison, we assign the 10 classes into 3 splits: many-shot, medium-shot, few-shot, with many-shot the most populated 3 classes, medium-shot the medium 3, and few-shot the least 4 classes.

ImageNet-SSLT

To evaluate the effectiveness of semi-supervised long-tailed recognition methods on large-scale datasets, we assemble a challenging dataset from ImageNet (ILSVRC-2012) [19]. The supervised subset is sampled with Lomax distribution with shape parameter $\alpha = 6$, scale parameter $\lambda = 1000$. It contains $41,134$ images from 1000 classes, with the maximum of 250 images per class and the minimum of 2 samples. The unsupervised subset is sampled under the same distribution with an unsupervised factor 4, *i.e.* $|\mathcal{U}| = 4|\mathcal{D}|$. The 1000 classes are divided into 3 splits based on the amount of labeled data n: many-shot ($n > 100$), medium-shot ($10 < n \leq 100$), few-shot ($n \leq 10$). In result, the dataset has 140 many-shot, 433 medium-shot, and 427 few-shot classes. Methods are evaluated under all classes and each class split.

iNaturalist2018-SSLT

We further curate a benchmark for semi-supervised long-tailed recognition based on iNaturalist 2018 [128]. iNaturalist 2018 is a long-tailed dataset sampled from natural distribution. We follow the distribution in both of the labeled and unlabeled subset. More specifically, Samples in each class is randomly down-sampled one-fifth of the total number as labeled data, and the remains are assigned as unsupervised subset. Classes with less than 2 labeled samples are eliminated. In result, iNaturalist2018-SSLT contains 8080 classes, with labeled samples from 200 to 2, and the unsupervised subset is 4 times larger. Classes are divided into three splits based on the number of labeled samples: many-shot ($[100, +\infty)$), medium-shot ($[10, 100)$), and few-shot ($[2, 10)$). It is a extremely long-tailed dataset, with 134 many-shot classes, 1220 medium-shot classes, and 7010 few-shot classes.

7.4.2 Network Architecture

ResNet-18 [38] is used on both CIFAR-10-SSLT and ImageNet-SSLT for fast experiments and comparison. ResNet-50 [38] is used on ImageNet-SSLT and iNaturalist2018-SSLT to show how methods scale up to larger networks.

7.4.3 Comparison Methods

To our best knowledge, there is no available method designated for semi-supervised long-tailed recognition. We explore typical long-tailed recognition methods and semi-supervised recognition methods, and combine them as baselines.

Long-tailed Recognition

We consider two long-tailed methods, one for loss calibration and the other for re-sampling. LDAM-DRW [7] converts cross-entropy loss to LDAM loss with calibration factors based on class counts. It further regulates the loss with a loss weight also from class counts. Decoupling [50] decouples the training of embedding and classifier with different sampling strategies. This is also the initialization in our method.

Semi-supervised Recognition

Pseudo-Label is a basic semi-supervised learning algorithm and can be easily combined with other models. It contains two phases. The first phase is initialization, the recognition model is trained on labeled data. Predictions of the initialized model are assigned on unlabeled data, *i.e.* pseudo labels. The initialized model is then trained or fine-tuned on the combination of labeled and unlabeled data. In practice, we combine Pseudo-Label method with the two long-tailed recognition models to create two semi-supervised long-tailed recognition baselines. Pseudo-Label combined with LDAM-DRW is the method used in [144].

Mean Teacher [121] is a well-known semi-supervised learning method. It contains a Student model that is trained with SGD and a Teacher model that is updated with moving average of the Student. It is, however, unclear how to train it with Decoupling. We only implement LDAM loss with Student training.

7.4.4 Training Detail

In initialization, the feature embedding is trained with 200 epochs, and classifier is learned in 10 epochs after that. Stage 2 contains 40 epochs of fine-tuning of the embedding on the whole dataset. In 5 loops of stages, it is in total 200 epochs of embedding fine-tuning. There are also 10 epochs of classifier fine-tuning in Stage 3 per loop. In semi-supervised learning loss (7.4), $\lambda = 1$ is used.

SGD optimizer with learning rate of 0.1 is used with cosine annealing during training in all stages. The momentum is 0.9, and weight decay is 0.0005.

All comparison methods are implemented with the hyper-parameters in their papers. The codes from authors are used when available.

7.4.5 Results

CIFAR-10-SSLT

Results are shown in Table 7.1 with imbalance ratio 100 and 1000. Our methods outperforms all other methods in overall accuracy.

Our initialized model is equivalent to Decoupling, which shows the worst performance among all methods. Alternate learning improves the overall performance more than 7% when imbalance factor is 100, and 20% with imbalance factor 1000. Most of the improvement is from medium and few-shot classes. The larger improvement on the more imbalanced distribution shows that our method is more effective with more skewed dataset.

When Pseudo-Label is added upon Decoupling, around 5% improvement is achieved with imbalance factor 100. But this improvement diminishes when the data is more imbalanced. This implies the fact that Pseudo-Label is more sensitive to bad tail class labelling.

With the improvement upon Pseudo-Label, our method has the same amount of training epochs on unsupervised data. The extra calculation in our methods compared to Pseudo-Label is from Stage 1 and 2. However, the classifier training is only on supervised data, and only the linear classifier is updated. And label assignment does not involve any back-propagation. The extra time on these two stages are trivial compared to the training of the whole model on the whole dataset.

LDAM-DRW provides very competitive results without any semi-supervised learning methods when imbalance factor is 100. However, it scales up bad when combined with semi-supervised techniques. By adding Pseudo-Label, it only improves 2% of overall accuracy. After looking at the splits results, we find that it improves the few-shot performance at the cost of many-shot. We believe this is because the wrong balancing factor introduced in LDAM loss. It does not match the true distribution, and skews the training process. Mean Teacher makes little difference from Pseudo-Label on LDAM-DRW.

ImageNet-SSLT

Results are shown in Table 7.2. Our methods outperforms all baseline methods with both ResNet-18 and -50 architectures. The ImageNet-SSLT setting is really challenging that all of the methods give below 30% overall accuracy. In fact, our method is the only one that improves the few-shot performance while maintains the many-shot accuracy.

On ImageNet-SSLT, Pseudo-Label based methods lose efficacy, because it improves few-shot performance with sacrifice on many-shot. This sacrifice is sometimes big, such as Pseudo-Label+LDAM-DRW with ResNet-18. This is not observed when Pseudo-Label is used on CIFAR-10-SSLT, where it improves the many-shot performance. This may be due to the bad many-shot pseudo-label quality on ImageNet-SSLT. Unlike CIFAR-10-SSLT, where the initialized model has 90% of accuracy on many-shot, many-shot performance on ImageNet-SSLT is only around 50%. These wrong labels can mislead the training and lower the performance of Pseudo-Label methods. Our method, on the other hand, updates the pseudo labels iteratively, and is less prone to this problem.

Specifically, adding Pseudo-Label on LDAM-DRW decreases the overall performance. This can be explained by the fact that the balancing factor in it does not match the true distribution. Mean Teacher improves upon LDAM-DRW when ResNet-50 is used. But it is still not as good as ours.

iNaturalist2018-SSLT

Results are shown in Table 7.3. Our method is the only one that improves the overall performance upon baseline. iNaturalist2018-SSLT is different from our other benchmarks in the amount of few-shot classes. It has a very long tail taking up 87% of the label space. This makes the dataset especially hard when combined with unsupervised data.

With the inferior quality of predictions, we see significant drop of Pseudo-Label method in many-shot split. In fact, Pseudo-Label decreases the accuracy of baselines in all splits. Our method mitigates this problem, and improve the few-shot performance. Given the fact that most classes are in few-shot split, our method is the only one that increase the overall performance.

Comparison among Benchmarks

From CIFAR-10-SSLT to ImageNet/iNaturalist2018-SSLT, the datasets have more and more classes and few-shot classes. In result, they are more and more challenging. This challenge makes Pseudo-Label method ineffective. From CIFAR-10-SSLT to ImageNet-SSLT, the shortcoming first appears in many-shot splits. On ImageNet-SSLT, Pseudo-Label improves the few-shot performance with a sacrifice of many-shot performance. Our method is more robust to this difficulty. It keeps the many-shot performance while improves the few-shot performance. On iNaturalist2018-SSLT, the Pseudo-Label improvement on few-shot split also disappears, and the drop on many-shot is big. Our method, however, can still improves the few-shot performance and control the drop of many-shot compared to the baseline.

All of these results show that semi-supervised long-tailed recognition is a challenging problem. Given the fact that this problem follows the natural workflow of data collecting, we believe it deserves more attention in the literature.

7.4.6 Ablations

We further study the training choices of alternate learning. This consists of two parts, *i.e.* the sampling choices and semi-supervised learning choices. Results on CIFAR-10-SSLT with imbalance factor 100 are listed in Table 7.4.

Sampling Choice

Currently, during alternate learning we use random sampling in Stage 2 and class-balance sampling in Stage 3. This is consistent with long-tailed recognition [50]. However, other combinations are possible. Results are listed in the first 3 lines of Table 7.4, with naming format:

$\{$sampling in Stage 2$\}$+$\{$sampling in Stage 3$\}$. In method names, "R" stands for random sampling and "C" stands for class-balanced.

None of the 3 alternatives can beat the initialized model (Decoupling). This is expected. When the classifier is randomly trained ("R+R" and "C+R"), the model performs bad on few-shot classes. This will in turn harm the training of embedding by pseudo labels on unsupervised subset. "C+C" trains the feature embedding with class-balanced sampling. However, it is balancing on pseudo labels, which can be wrong. The results show that this balancing yields inferior feature embedding.

Semi-supervised Learning Choice

We train feature embedding with the whole dataset, *i.e.* $\mathcal{D} \cup \mathcal{U}'$, and the classifier with labeled subset $\mathcal{D}$. Other combinations can also be investigated. The classifier can also be semi-supervise trained, *i.e.* on $\mathcal{D} \cup \mathcal{U}'$. At the same time, feature embedding is trained with or without $\mathcal{U}'$. We show the results in the last 2 lines of Table 7.4. In these two experiments, the classifier is always trained on $\mathcal{D} \cup \mathcal{U}'$. The difference is whether $\mathcal{U}'$ is used for embedding learning.

Compared to the regular setting, where the classifier is trained on $\mathcal{D}$, when we train it on $\mathcal{D} \cup \mathcal{U}'$, the performance is slightly lower. This can be explained by the fact that wrong pseudo labels in $\mathcal{U}'$ can be propagated through loops if the classifier is optimized on them. This is especially true for few-shot classes, where the accuracy is low. Because of class-balanced sampling, the impact of few-shot classes is amplified. When compared to Table 7.1, the main performance drop is from few-shot classes. This confirms our assumption.

However, when we further remove the unsupervised training of embedding, the performance drops a lot. It is even worse than the initialized model (Decoupling). In this case, the feature embedding should be equivalent to that of the initialization. The only difference is the classifier. This further proves the fact that fine-tuning classifier on pseudo-labels harms the performance.

Table 7.4: Ablation results(Accuracy in %) on CIFAR-10-SSLT, Imbalance factor 100 is used. Sampling methods are denoted as R for random, and C for class-balanced. The last two method names shows where the embedding is trained.

Method	Overall	Many-Shot	Medium-Shot	Few-Shot
R + R	50.9	93.0	57.8	14.1
C + R	61.2	91.3	62.6	37.6
C + C	63.3	91.2	64.4	41.6
$\mathcal{D} \cup \mathcal{U}'$	70.1	89.6	68.7	56.5
$\mathcal{D}$	63.3	91.6	61.9	43.2

Table 7.5: Pseudo label accuracy on unlabeled training subset. CIFAR-10-SSLT with imbalance ratio 100 is used. Compared to testing set, the unsupervised subset is not balanced. In result, the overall accuracy is higher than that on testing set, because of the domination of many-shot classes. The results in many/medium/few-shot splits are more useful.

Loop	Overall	Many-Shot	Medium-Shot	Few-Shot
0	87.7	92.3	63.0	41.8
1	87.9	92.3	64.0	48.1
2	87.8	92.1	64.7	52.2
3	87.8	91.8	65.3	55.8
4	87.7	91.6	65.8	57.8

Accuracy on Unsupervised Training Subset

In Stage 1, we assign pseudo labels for all samples in $\mathcal{U}$. Table 7.5 reveals how the accuracy changes along loops in all splits. Few-shot split performance improves much faster than others. This proves the effectiveness of our alternate learning scheme, and explains why our method outperforms the baselines by a large margin in few-shot classes.

The unsupervised subset has a long-tailed distribution, so the overall performance is dominated by many-shot. However, alternate learning still gets benefits from the improvement on few-shot split. Accuracy on different splits is more useful when we analyze how the model evolves during training.

7.5 Conclusion

This work introduces the semi-supervised long-tailed recognition problem. It extends the long-tailed problem with unsupervised data. With the property of labeled and unlabeled data obeying the same distribution, this problem setting follows the realistic data collection and annotation workflow.

A method based on alternate learning is proposed. By separating supervised training from semi-supervised and decoupling the sampling methods, it incorporates the decoupling training scheme in long-tailed recognition with semi-supervised learning.

Experiments show that the proposed method outperforms all current baselines. When results are split based on class cardinality, the method exhibits its robustness to defective pseudo labels. This is especially true for few-shot classes.

Chapter 7 is, in full, based on the material as it appears in the submission of “Semi-supervised Long-tailed Recognition using Alternate Sampling”, Bo Liu, Haoxiang Li, Hao Kang, Nuno Vasconcelos, Gang Hua, in *IEEE International Conference on Computer Vision* (ICCV), 2021. The dissertation author was the primary investigator and author of this material.

Chapter 8

Discussion and Conclusion

In the thesis, we have discussed limitations of computer vision datasets, and proposed several methods to deal with them. For the pose bias problem, we proposed to augment data along pose trajectories. First, FATTEN was introduced to synthesis data in feature space. Specifically, pose transfer was achieved by decoupling. A latent feature space was first decoupled into appearance space and pose space. With a new target pose, a decoder combined both spaces and produced a target feature in feature space. Experiments were presented on retrieval, recognition and 3D reconstruction tasks. Synthesized features, as queries, could retrieve real features with the same class and pose by a high mAP. With limited labelled samples, few-shot recognition was shown to be beneficial from synthesized features with novel poses. Performance on single-view 3D reconstruction, which only one view was available as input, was also improved by feature with new poses.

Second, we studied the pose bias problem by directly generate images with novel views. A domain transfer based framework, DRAW, was discussed. It combined the real world novel view synthesis with the well-studied view synthesis problem in synthetic domain by domain transfer. To deal with the mismatching problem when transferred from synthetic domain to natural domain, we introduced an identity recovery module that disentangled and combined structure information and appearance information, and was trained in a semi-supervised manner. Qualitative and quantitative evaluations showed that the proposed framework out-performed other methods when novel views were not seen during training.

The imbalance nature of real world class distribution has been considered and studied by improving classification model and its training strategy. First of all, for those less populated classes, few-shot problem was considered. Few-shot open-set recognition problem was proposed by considering the few-shot problem with open-set setting. A meta-learning based model, PEELER, was introduced. It was beneficial from the random sampling training strategy of meta-learning, and incorporated open-set loss and Gaussian embedding. Experiments showed that it could reject unseen classes successfully, while maintain the few-shot recognition accuracy.

With a larger label space with classes from both many- and few-shot, a long-tailed recognition problem has been discussed. Two methods, GistNet and Breadcrumb, were introduced. GistNet discussed the feature geometry collapsing problem in few-shot classes. The geometry was recovered by transferring the geometry from many-shot. The geometry recovery classifier was then trained with Gist Training, a hybrid training procedure that involved both random sampling and class-balanced sampling. Breadcrumb, on the other hand, discuss the over-fitting problem in feature space, when class-balanced sampling was used. Feature back-tracking was presented. It mitigated over-fitting on few-shot classes with no extra computation, and kept the many-shot performance. Experiments showed that both of the methods improved the recognition performance on long-tailed distribution training set with a large margin.

Finally, we revealed that the long-tailed problem could not solved only with improved models. It was also helpful to introduce more data. In result, a new task, semi-supervised long-tailed recognition, was introduced. It followed dataset collection procedure. And when datasets were partially annotated due to the budget, the supervised subset and the unsupervised subset should follow the same long-tailed distribution, *i.e.* the natural distribution. An alternate learning method was proposed to optimize the model on both supervised and unsupervised subset, and at the same time, balance the performance on all classes with long-tailed distribution. Its superior performance showed the success of introducing unsupervised data into long-tailed problem. By using all the data during data collection, it was also a more efficient way to use computer vision data.

Computer vision is a scientific field that studies real world visual information. Datasets are the finite representation of the infinite real world information. Limitations are the compromise of the finite data collection effort. With the development of techniques, the datasets will be more and more powerful, and the limitations can be different, but they always exist. However, we can discuss, study and find a way to overcome those limitations or even take use of them. This will motivate me to continue working on the topic and move computer vision datasets more useful despite the limitations.

Bibliography

[1] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *CVPR*, 2018.

[2] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749, 2005.

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[4] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[5] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

[6] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[7] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, pages 1565–1576, 2019.

[8] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[10] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016.

[11] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. Semi-supervised deep learning with memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 268–283, 2018.

[12] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 45–54, 2020.

[13] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[14] Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8680–8689, 2019.

[15] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[17] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.

[18] Debasmit Das and CS George Lee. A two-stage approach to few-shot learning for image recognition. *IEEE Transactions on Image Processing*, 29:3336–3350, 2019.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[20] Mandar Dixit, Roland Kwitt, Marc Niethammer, and Nuno Vasconcelos. Aga: Attribute-guided augmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7455–7463, 2017.

[21] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[22] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3D object reconstruction from a single image. In *CVPR*, 2017.

[23] Rafael Felix, Vijay BG Kumar, Ian Reid, and Gustavo Carneiro. Multi-modal cycle-consistent generalized zero-shot learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 21–37, 2018.

[24] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[25] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[26] ZongYuan Ge, Sergey Demyanov, Zetao Chen, and Rahil Garnavi. Generative openmax for multi-class open set classification. *arXiv preprint arXiv:1707.07418*, 2017.

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

[28] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[29] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[30] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[31] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[32] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[33] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *European conference on computer vision*, pages 241–257. Springer, 2016.

[34] T Groueix, M Fisher, VG Kim, BC Russell, and M Aubry. Atlasnet: a papier-mâché approach to learning 3d surface generation (2018). *arXiv preprint arXiv:1802.05384*, 11, 2018.

[35] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. *CoRR*, `arXiv:1606.02819v4`, 2016.

[36] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[37] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[39] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[40] Chih-Hui Ho, Pedro Morgado, Amir Persekian, and Nuno Vasconcelos. Pies: Pose invariant embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12377–12386, 2019.

[41] Judy Hoffman, Saurabh Gupta, Jian Leong, Sergio Guadarrama, and Trevor Darrell. Cross-modal adaptation for rgb-d detection. In *ICRA*, 2016.

[42] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[43] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[44] Chen Huang, Yining Li, Change Loy Chen, and Xiaoou Tang. Deep imbalanced learning for face recognition and attribute prediction. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[45] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[46] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[47] Bernd Jahne. *Computer vision and applications: a guide for students and practitioners*. Elsevier, 2000.

[48] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3779–3788, 2017.

[49] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8420–8429, 2019.

[50] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *Eighth International Conference on Learning Representations (ICLR)*, 2020.

[51] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017.

[52] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[53] Reinhard Klette. *Concise computer vision*. Springer, 2014.

[54] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3D SURF for robust three dimensional classification. In *ECCV*, 2010.

[55] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[56] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[58] Morton Kupperman et al. Probabilities of hypotheses and information-statistics in sampling from exponential-class populations. *The Annals of Mathematical Statistics*, 29(2):571–575, 1958.

[59] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *ICRA*, 2011.

[60] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[61] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):453–465, 2013.

[62] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[63] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104. IEEE, 2004.

[64] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

[65] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2019.

[66] Qimai Li, Xiao-Ming Wu, and Zhichao Guan. Generalized label propagation methods for semi-supervised learning. 2018.

[67] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[68] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[69] Bo Liu, Xudong Wang, Mandar Dixit, Roland Kwitt, and Nuno Vasconcelos. Feature space transfer for data augmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9090–9098, 2018.

[70] Jialun Liu, Yifan Sun, Chuchu Han, Zhaopeng Dou, and Wenhui Li. Deep representation learning on long-tailed data: A learnable embedding augmentation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2970–2979, 2020.

[71] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017.

[72] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

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

[74] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[75] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2537–2546, 2019.

[76] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[77] Jiajun Lu, Theerasit Issaranon, and David Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 446–454, 2017.

[78] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *NIPS*, 2017.

[79] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[80] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[81] S Nagabhushana. *Computer vision and image processing*. New Age International, 2005.

[82] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[83] Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng-Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628, 2018.

[84] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-100). 1996.

[85] Natalia Neverova, Riza Alp Guler, and Iasonas Kokkinos. Dense pose transfer. In *ECCV*, 2018.

[86] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012, 2016.

[87] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

[88] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *ICCV*, 2017.

[89] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 3500–3509, 2017.

[90] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1278–1286, 2015.

[91] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[92] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.

[93] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.

[94] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Domain transfer for 3D pose estimation from color images without manual annotations. *arXiv preprint arXiv:1810.03707*, 2018.

[95] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder networks. *arXiv preprint arXiv:1507.02672*, 2015.

[96] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[97] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 331–339, 2019.

[98] Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Semi-supervised learning with scarce annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 762–763, 2020.

[99] Mengye Ren, Renjie Liao, Ethan Fetaya, and Richard Zemel. Incremental few-shot learning with attention attractor networks. In *Advances in Neural Information Processing Systems*, pages 5276–5286, 2019.

[100] Mengye Ren, Renjie Liao, Ethan Fetaya, and Richard S Zemel. Incremental few-shot learning with attention attractor networks. *arXiv preprint arXiv:1810.07218*, 2018.

[101] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International conference on machine learning*, pages 2152–2161. PMLR, 2015.

[102] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[103] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[104] Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*, 2018.

[105] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[106] Patrick Schlachter, Yiwen Liao, and Bin Yang. Open-set recognition using intra-class splitting. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2019.

[107] Saurabh Sharma, Ning Yu, Mario Fritz, and Bernt Schiele. Long-tailed recognition using class-balanced experts. *arXiv preprint arXiv:2004.03706*, 2020.

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

[109] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[110] Richard Socher, Milind Ganjoo, Hamsa Sridhar, Osbert Bastani, Christopher D Manning, and Andrew Y Ng. Zero-shot learning through cross-modal transfer. *arXiv preprint arXiv:1301.3666*, 2013.

[111] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[112] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[113] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[114] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[115] Sainbayar Sukhbaatar and Rob Fergus. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, 2(3):4, 2014.

[116] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 155–171, 2018.

[117] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3D shape modeling. In *CVPR*, 2018.

[118] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[119] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[120] Kevin D Tang, Marshall F Tappen, Rahul Sukthankar, and Christoph H Lampert. Optimizing one-shot recognition with micro-set learning. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3027–3034. IEEE, 2010.

[121] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

[122] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*, pages 322–337. Springer, 2016.

[123] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[124] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[125] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 551–560, 2020.

[126] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[127] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[128] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[129] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[130] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

[131] Le Wang, Gang Hua, Rahul Sukthankar, Jianru Xue, Zhenxing Niu, and Nanning Zheng. Video object discovery and co-segmentation with extremely weak supervision. *IEEE transactions on pattern analysis and machine intelligence*, 39(10):2074–2088, 2016.

[132] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7278–7286, 2018.

[133] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*, pages 616–634. Springer, 2016.

[134] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pages 7029–7039, 2017.

[135] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020.

[136] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3D shape reconstruction via 2.5 d sketches. In *NIPS*, 2017.

[137] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[138] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

[139] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5542–5551, 2018.

[140] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. f-vaegan-d2: A feature generating framework for any-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10275–10284, 2019.

[141] Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *European Conference on Computer Vision*, pages 247–263. Springer, 2020.

[142] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[143] Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3D view synthesis. In *NIPS*, 2015.

[144] Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. *arXiv preprint arXiv:2006.07529*, 2020.

[145] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5409–5418, 2017.

[146] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *ICCV*, 2017.

[147] Bo Zhao, Xiao Wu, Zhi-Qi Cheng, Hao Liu, Zequn Jie, and Jiashi Feng. Multi-view image generation from a single-view. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 383–391. ACM, 2018.

[148] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. Re-ranking person re-identification with k-reciprocal encoding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1318–1327, 2017.

[149] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[150] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9719–9728, 2020.

[151] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer, 2016.

[152] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[153] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922, 2014.