

October 2021

## Learning from Limited Labeled Data for Visual Recognition

Jong-Chyi Su  
*University of Massachusetts Amherst*

Follow this and additional works at: [https://scholarworks.umass.edu/dissertations\\_2](https://scholarworks.umass.edu/dissertations_2)



Part of the [Artificial Intelligence and Robotics Commons](#)

---

### Recommended Citation

Su, Jong-Chyi, "Learning from Limited Labeled Data for Visual Recognition" (2021). *Doctoral Dissertations*. 2365.

<https://doi.org/10.7275/24431687> [https://scholarworks.umass.edu/dissertations\\_2/2365](https://scholarworks.umass.edu/dissertations_2/2365)

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact [scholarworks@library.umass.edu](mailto:scholarworks@library.umass.edu).

**LEARNING FROM LIMITED LABELED DATA  
FOR VISUAL RECOGNITION**

A Dissertation Presented

by

Jong-Chyi Su

Submitted to the Graduate School of the  
University of Massachusetts Amherst in partial fulfillment  
of the requirements for the degree of

**DOCTOR OF PHILOSOPHY**

September 2021

College of Information and Computer Science

© Copyright by Jong-Chyi Su 2021

All Rights Reserved

# LEARNING FROM LIMITED LABELED DATA FOR VISUAL RECOGNITION

A Dissertation Presented

by

Jong-Chyi Su

Approved as to style and content by:

---

Subhransu Maji, Chair

---

Erik Learned-Miller, Member

---

Rui Wang, Member

---

Bharath Hariharan, Member

---

James Allan, Chair of the Faculty  
College of Information and Computer Science

## ACKNOWLEDGMENTS

I would like to express my gratitude to everyone who made graduate school a remarkable journey for me. Thanks Subhransu Maji for giving me the opportunity to work with such an incredible mentor. I could not finish my thesis without all your advice and guidance. I also want to thank Erik Leaned-Miller, Bharath Hariharan, and Rui Wang for the thoughtful discussions and project ideas.

I would like to thank Omkar Parkhi, Tamara Berg, Yi-Hsuan Tsai, R. Manmatha, and Deva Ramanan for being my mentors during internships. Thanks Serge Belongie and Ravi Ramamoorthi for giving me research opportunities at UCSD, and Michael Tao for your mentorship. Thanks Yung-Yu Chuang and Shao-Yi Chien for leading me to the field of computer vision at NTU.

My PhD could not be fulfilled without all the support from the vision lab. Thanks Tsung-Yu Lin for your mentorship, you have always been a role model for me. Thanks Chenyun Wu and Zezhou Cheng for our collaborations in exciting projects. Thanks Aruni RoyChowdhury and Souyoung Jin for our adventures in Amherst. Thanks Matheus Gadelha, Huaizu Jiang, Gopal Sharma, Pia Bideau, Hang Su, and Ashish Singh for making an awesome vision lab.

I would also like to thank all my friends who have made my years in Amherst wonderful. Thanks Álex Santos and Ning-Hsuan Tseng for all the times hanging out together. Thanks Poyao Huang for your friendship. Thanks Mike Covault and Jon Covault, I will not forget your support and our moments together. Thanks Steve, Ming-Che, Yuan-Fen, Celia, Li-Yang, Peter, Jarrel, and Nigel in Amherst. Thanks to all my old friends: Eugene Wang, Yen-Han Chen, and Clark Chiu. Lastly, I am thankful to my family, who has given me all the support to pursue my dream without any worries.

## **ABSTRACT**

# **LEARNING FROM LIMITED LABELED DATA FOR VISUAL RECOGNITION**

SEPTEMBER 2021

Jong-Chyi Su

B.Sc., NATIONAL TAIWAN UNIVERSITY

M.Sc., UNIVERSITY OF CALIFORNIA SAN DIEGO

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Subhansu Maji

Recent advances in computer vision are in part due to the widespread use of deep neural networks. However, training deep networks require enormous amounts of labeled data which can be a bottleneck. In this thesis, we propose several approaches to mitigate this in the context of modern deep networks and computer vision tasks.

While transfer learning is an effective strategy for natural image tasks where large labeled datasets such as ImageNet are available, it is less effective for distant domains such as medical images and 3D shapes. Chapter 2 focuses on transfer learning from natural image representations to other modalities. In many cases, cross-modal data can be generated using computer graphics techniques. By forcing the agreement of predictions across modalities, we show that the models are more robust to image degradation, such as lower resolution, grayscale, or line drawings instead of color images in high-resolution. Similarly, we show that 3D shape classifiers learned from multi-view images can be transferred to the models of voxel or point cloud representations.

Another line of work has focused on techniques for few-shot learning. In particular, meta-learning approaches explicitly aim to generalize representations by emphasizing transferability to novel tasks. In Chapter 3, we analyze how to improve these techniques by exploiting unlabeled data from related tasks. We show that combining unsupervised objectives with meta-learning objectives can boost the performance of novel tasks. However, we find that small amounts of domain-specific data can be more beneficial than large amounts of generic data.

While transfer learning, unsupervised learning, and few-shot learning have been studied in isolation, in practice, one often finds that transfer learning from large labeled datasets is more effective than others. This is partly due to a lack of evaluation on benchmarks that contains challenges such as class imbalance and domain mismatch. In Chapter 4, we explore the role of expert models in the context of semi-supervised learning on a realistic benchmark. Unlike existing semi-supervised benchmarks, our dataset is designed to expose some of the challenges encountered in a realistic setting, such as the fine-grained similarity between classes, significant class imbalance, and domain mismatch between the labeled and unlabeled data. We show that current semi-supervised methods are negatively affected by out-of-class data, and their performance pales compared to a transfer learning baseline. Last, we leverage the coarse labels from a large collection of images to improve semi-supervised learning. In Chapter 5, we show that incorporating hierarchical labels in the taxonomy improves state-of-the-art semi-supervised methods.

# TABLE OF CONTENTS

	<b>Page</b>
<b>ACKNOWLEDGMENTS</b> .....	<b>iv</b>
<b>ABSTRACT</b> .....	<b>v</b>
<b>LIST OF TABLES</b> .....	<b>x</b>
<b>LIST OF FIGURES</b> .....	<b>xiii</b>
 <b>CHAPTER</b>	
<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1 Transfer learning .....	2
1.2 Few-shot learning .....	3
1.3 Self-supervised learning .....	4
1.4 Semi-supervised learning .....	5
1.5 Cross-modal learning and knowledge distillation .....	7
1.6 Contributions .....	7
<b>2. CROSS-MODAL LEARNING USING KNOWLEDGE DISTILLATION</b> .....	<b>9</b>
2.1 Knowledge Distillation .....	10
2.2 Experiments on Image Classification .....	11
2.2.1 Experimental Setup .....	12
2.2.1.1 Datasets .....	12
2.2.1.2 Models .....	12
2.2.1.3 Methods .....	13
2.3 Results on Image Classification .....	14
2.3.1 Simultaneous CQD and Model Compression .....	18
2.3.2 Understanding Why CQD Works .....	19
2.3.2.1 Relation to curriculum learning .....	19



2.3.2.2	Understanding CQD through gradient visualizations . . . . .	20
2.4	Experiments on 3D Shape Classification . . . . .	20
2.4.1	Experimental Setup . . . . .	22
2.4.1.1	Classification benchmark. . . . .	22
2.4.1.2	Input representations. . . . .	22
2.4.1.3	Data augmentation. . . . .	23
2.4.1.4	Classification Architectures. . . . .	24
2.5	Results on 3D Shape Classification . . . . .	25
2.5.1	Learning From a Few Examples . . . . .	25
2.5.2	Dissecting the MVCNN Architecture . . . . .	26
2.5.2.1	Effect of model architecture. . . . .	26
2.5.2.2	Effect of ImageNet pretraining. . . . .	27
2.5.2.3	Effect of shape rendering. . . . .	28
2.5.3	Cross-Modal Distillation. . . . .	28
2.5.4	Tradeoffs between Learned Representations . . . . .	30
2.5.5	Comparison to Prior Work . . . . .	31
<b>3.</b>	<b>IMPROVING FEW-SHOT LEARNING WITH SELF-SUPERVISED</b>	
	<b>LEARNING . . . . .</b>	<b>34</b>
3.1	Method . . . . .	36
3.1.1	Supervised Losses ( $\mathcal{L}_s$ ) . . . . .	37
3.1.2	Self-supervised Losses ( $\mathcal{L}_{ss}$ ) . . . . .	38
3.1.3	Stochastic Sampling and Training . . . . .	39
3.2	Experiments . . . . .	39
3.2.1	Experimental Setup . . . . .	39
3.2.2	Results on Few-shot Learning . . . . .	41
3.2.2.1	Self-supervised learning improves few-shot learning . . . . .	41
3.2.2.2	Gains are larger for harder tasks . . . . .	42
3.2.2.3	Improvements generalize to other meta-learners . . . . .	43
3.2.2.4	Self-supervision alone is not enough . . . . .	45
3.2.2.5	Few-shot learning as an evaluation for self-supervised tasks . . . . .	45
3.2.2.6	Comparison with prior works . . . . .	45
3.2.3	Analyzing the Effect of Domain Shift for Self-supervision . . . . .	45

3.2.4	Selecting Images for Self-supervision .....	48
<b>4.</b>	<b>A REALISTIC EVALUATION ON SEMI-SUPERVISED LEARNING .....</b>	<b>49</b>
4.1	A Realistic Benchmark .....	53
4.1.1	Semi-Aves .....	54
4.1.2	Semi-Fungi .....	54
4.1.3	Semi-iNat .....	55
4.2	Methods .....	56
4.3	Experiments .....	59
4.3.1	Implementation details .....	59
4.3.2	Results .....	64
4.3.2.1	Training from scratch. ....	65
4.3.2.2	Using expert models. ....	65
4.3.2.3	Effect of out-of-class unlabeled data. ....	66
4.3.2.4	Robustness to hyper-parameters and trends. ....	66
4.3.3	Comparison with related prior work on Semi-SL analysis .....	67
4.3.4	Analysis on out-of-class unlabeled data .....	67
<b>5.</b>	<b>SEMI-SUPERVISED LEARNING WITH TAXONOMIC LABELS .....</b>	<b>70</b>
5.1	Related Works .....	72
5.2	Method .....	73
5.2.1	Hierarchical supervised loss .....	73
5.2.2	Joint training with semi-supervised loss .....	74
5.3	Experiments .....	76
5.3.1	Experimental settings .....	76
5.3.2	Using phylum level supervision .....	78
5.3.3	Using different levels of supervision .....	79
5.3.4	Effect of domain shift .....	80
<b>6.</b>	<b>CONCLUSION .....</b>	<b>82</b>
	<b>BIBLIOGRAPHY .....</b>	<b>84</b>

## LIST OF TABLES

Table	Page
<p>2.1 <b>Cross quality distillation results.</b> Per-image accuracy on birds dataset (CUB) [179] and Stanford cars dataset (CARS) [91] for various methods and quality losses. All results are using <math>f = g = \text{vgg-m}</math> model. Training on <math>\mathcal{A}</math> and testing on <math>\mathcal{A}</math> is the upper bound of the performance in each setting (top row). Training on <math>\mathcal{A}</math> and testing on <math>\mathcal{B}</math> (no adaptation) often leads to a significant loss in performance. The proposed technique (CQD) outperforms fine-tuning (<math>\mathcal{B}</math>), data augmentation (<math>\mathcal{A} + \mathcal{B}</math>), and staged training (<math>\mathcal{A}, \mathcal{B}</math>) [120] on all datasets. ....</p>	15
<p>2.2 Accuracy of various techniques on the CUB localized/non-localized dataset. ....</p>	19
<p>2.3 Accuracy (%) of MVCNN with different CNN architectures. The VGG-11 architectures are on par with the residual network variants. ....</p>	27
<p>2.4 Effect of ImageNet pretraining on the accuracy (%) of MVCNN. The VGG-11 architecture is used and the full training/test split of the ModelNet40 dataset is used. ....</p>	28
<p>2.5 Accuracy (%) of MVCNN with different rendering methods. The number in the brackets are the number of views used. 12 views are used if not specified. ....</p>	29
<p>2.6 Accuracy, speed and memory comparison of different models. Memory usage during training which includes parameters, gradients, and layer activations, for a batch size 64 is shown. Forward-pass time is also calculated with batch size 64 using PyTorch with a single GTX Titan X for all the models. The input resolutions are <math>224 \times 224</math> for MVCNN, <math>32^3</math> for VoxNet, and 1024 points for PointNet. The accuracy numbers in brackets are for models trained with distillation as described in Chapter 2.5.3. ....</p>	31
<p>2.7 Accuracy (%) of state-of-the-art methods with different 3D representations. PC refers to point clouds. The order is grouped by input type and sorted by accuracy. ....</p>	33

3.1	<b>Example images and dataset statistics.</b> For few-shot learning experiments the classes are split into <i>base</i> , <i>val</i> , and <i>novel</i> set. Image representations learned on <i>base</i> set are evaluated on the <i>novel</i> set while <i>val</i> set is used for cross-validation. These datasets vary in the number of classes but are orders of magnitude smaller than ImageNet dataset. ....	40
3.2	<b>Performance on few-shot learning using different meta-learners.</b> Using jigsaw puzzle loss improves different meta-learners on most of the datasets. ProtoNet with jigsaw loss performs the best on all five datasets. ....	43
3.3	<b>Comparison with prior works on <i>mini-ImageNet</i>.</b> 5-shot 5-way classification accuracies on 600 test episodes are reported. The implementation details including image size, backbone model, and training are different in each paper. *validation classes are used for training. †dropblock [51], label smoothing, and weight decay are used. ....	44
4.1	A comparison of Semi-Aves, Semi-Fungi, and Semi-iNat datasets with existing Semi-SL benchmarks. Semi-Aves, Semi-Fungi, and Semi-iNat present a challenge due to the large number of classes, presence of novel images in the unlabeled set, long-tailed distribution of classes as indicated by the class imbalance ratio (maximum / minimum images per class) in the training set. ....	53
4.2	<b>The number of species in the taxonomy.</b> For each phylum, we select around one-third of the species for the in-class set $C_{in}$ and the rest for the out-of-class set $C_{out}$ . ....	55
4.3	<b>Results on Semi-Aves benchmark.</b> We experiment with six different Semi-SL methods as well as supervised baselines under different settings: (1) using $U_{in}$ or $U_{in} + U_{out}$ as the unlabeled dataset, (2) training from scratch, or using ImageNet or iNat pre-trained model. We show that when training from scratch with $U_{in}$ , MoCo + Self-Training performs the best. When having expert models, transfer learning is a strong baseline, and FixMatch and Self-Training can still give improvements. When adding unlabeled data from $U_{out}$ , the performance pales except for the self-supervised method when training from scratch. The best results and those within the variance are marked in teal. ....	60

4.4	<b>Results on Semi-Fungi benchmark.</b> We experiment on Semi-Fungi using the same hyper-parameters from Semi-Aves in Table 4.3. We can see similar conclusions: When training from scratch, MoCo + Self-Training performs the best and adding $U_{out}$ can give an extra performance boost. With expert models, FixMatch and Self-Training (with or without MoCo) is often the best performing one, but the latter is more robust to the out-of-class data. ....	61
4.5	<b>Results on Semi-iNat benchmark.</b> We experiment on Semi-iNat which is a larger dataset than Semi-Aves and Semi-Fungi. In this table, the validation set is for selecting best models during training, but not incorporated in the supervised loss. The conclusions are similar as Semi-Aves. When training from scratch, MoCo + Self-Training is the best. When initialized using expert models, FixMatch works the best when having in-class unlabeled data only, but Self-Training is more robust to out-of-class data. ....	62
5.1	<b>Statistics of the Semi-iNat dataset [155]. Left: Number of classes under each kingdom and phylum.</b> The species of Semi-iNat come from 3 kingdoms and 8 phyla. In each phylum, one-third of the species are used for in-class species $C_{in}$ and the rest are used for out-of-class species $C_{out}$ . <b>Right: Number of classes in each level of the taxonomy.</b> ....	77
5.2	<b>Results of adding hierarchical loss on the Semi-iNat dataset.</b> Adding hierarchical loss at the <i>phylum</i> level improves supervised training and all the semi-supervised methods when images come from $U_{in}$ , <i>i.e.</i> , images with species labels corresponding to those in the test set. The best methods are shown in <b>teal</b> . ....	78
5.3	<b>Results on the Semi-iNat dataset when having out-of-domain data.</b> The out-of-domain data $U_{out}$ degrades the performances of all the methods, except for the self-supervised method when training from scratch. However, adding hierarchical loss still gives improvements in most cases. The best methods are shown in <b>teal</b> . ....	80

## LIST OF FIGURES

Figure	Page
2.1	The CQD framework. . . . . 10
2.2	Examples of images from various cross-quality datasets used in our experiments. Images are from the birds [179] and cars dataset [91]. In each panel, the top row shows examples of the high-quality images and the bottom row shows examples of the corresponding low-quality images. These include (a) localized and non-localized images, (b) high- and low-resolution images, (c) color and edge images, and (d) regular and distorted images. . . . . 13
2.3	<b>Left:</b> Image and gradient of the image with respect to the true class label for the model trained on $\mathcal{B}$ (non-localized images) and CQD (from a model trained on localized images). Darker pixels represent higher gradient value. The gradients of the model trained using CQD are more focused on the foreground object. <b>Right:</b> The scatter plot of the fraction of total gradient within the bounding-box for 1000 training images for the two models. . . . . 21
2.4	Different representations of the input shape. From the left (a) the shapes in the database are represented as triangle meshes, (b) the shape converted to a $30^3$ voxel grid, (c) point cloud representation with 2048 points, and (d-f) the model rendered using Phong shading, and as depth and binary silhouette images respectively. . . . . 22
2.5	Classification accuracy as a function of training set size. MVCNN generalizes better than the other approaches. The two MVCNN curves correspond to variants with and without ImageNet pretraining. . . . . 27
2.6	Model distillation from MVCNN to VoxNet and PointNet. The accuracy is improved by 1.8% for VoxNet and 0.3% for PointNet with whole training set. . . . . 30
2.7	Accuracy obtained by ensembling models. Left shows results by averaging the predictions while right shows results by training a linear model on the concatenated features extracted from different models. . . . . 31

3.1	<b>Combining supervised and self-supervised losses for few-shot learning.</b> Self-supervised tasks such as jigsaw puzzle or rotation prediction act as a data-dependent regularizer for the shared feature backbone. Our work investigates how the performance on the <i>target task domain</i> ( $\mathcal{D}_s$ ) is impacted by the choice of the <i>domain used for self-supervision</i> ( $\mathcal{D}_{ss}$ ). . . . .	35
3.2	<b>Benefits of SSL for few-shot learning tasks.</b> We show the accuracy of the ProtoNet baseline of using different SSL tasks. The jigsaw task results in an improvement of the 5-way 5-shot classification accuracy across datasets. Combining SSL tasks can be beneficial for some datasets. Here SSL was performed on images within the base classes only. . . . .	42
3.3	<b>Benefits of SSL for <i>harder</i> few-shot learning tasks.</b> We show the accuracy of using the jigsaw puzzle task over ProtoNet baseline on harder versions of the datasets. We see that SSL is effective even on smaller datasets and the relative benefits are higher. . . . .	42
3.4	<b>Effect of size and domain of SSL on 5-way 5-shot classification accuracy.</b> (a) More unlabeled data from the same domain for SSL improves the performance of the meta-learner. (b) Replacing a fraction (x-axis) of the images with those from other domains makes SSL less effective. . . . .	46
3.5	<b>Overview of domain selection for self-supervision. Top:</b> We first train a domain classifier using $\mathcal{D}_s$ and (a subset of) $\mathcal{D}_p$ , then select images using the predictions from the domain classifier for self-supervision. <b>Bottom:</b> Selected images of each dataset using importance weights. . . . .	47
3.6	<b>Effectiveness of selected images for SSL.</b> With random selection, the extra unlabeled data often hurts the performance, while those sampled using the <i>importance weights</i> improve performance on all five datasets. . . . .	47
4.1	Accuracy of semi-supervised learning (Semi-SL) algorithms on the Semi-Aves and Semi-Fungi datasets (see Fig. 4.2) using (i) different pre-trained models, and (ii) in-class ( $U_{in}$ ) and out-of-class ( $U_{in} + U_{out}$ ) unlabeled data. The performances of the supervised baseline and supervised oracle are also shown. Transfer learning from experts is far more effective than Semi-SL from <i>scratch</i> , while in the transfer setting Semi-SL provides modest gains. Though out-of-class data ( $U_{out}$ ) is valuable when training from scratch, it is not the case when training from experts (details in Tab. 4.3 and 4.4). . . . .	51

4.2	<b>The proposed benchmark for semi-supervised learning.</b> The benchmark contains two datasets, with classes from the Aves and Fungi taxa respectively. Each represents a 200-way classification task and the training set contains (i) labeled images from these classes $L_{in}$ , (ii) unlabeled images from these classes $U_{in}$ , and (iii) unlabeled images from related classes $U_{out}$ , as seen on figures to the right. Moreover, the classes exhibit a long-tailed distribution with an imbalance ratio of 8 to 10. The benchmark captures conditions observed in some realistic applications that are not present in existing datasets used to evaluate semi-supervised learning. See § 4.1 and Tab. 4.1 for details. . . . .	52
4.3	<b>Examples from the Semi-iNat dataset.</b> The Semi-iNat dataset includes images from 8 different phyla. . . . .	55
4.4	<b>Relative gains of Semi-SL methods on Semi-Aves and Semi-Fungi.</b> <b>Left:</b> trained from scratch. <b>Right:</b> using expert models. For each Semi-SL method, we plot the relative gain, <i>i.e.</i> the difference between the supervised baseline in raw accuracy, from the results in both Tab. 4.3 and 4.4. This shows that (1) the presence of out-of-class data $U_{out}$ often hurts the performance, and (2) Self-Training is often the best method when using pre-trained models. . . . .	65
4.5	<b>Pseudo-label with different threshold <math>\tau</math>.</b> Pseudo-label is sensitive to the threshold hyperparameter. The negative impact of out-of-class unlabeled data is reduced by increasing the threshold, yet when the initial performance is low the scheme is not effective as seen by the performance of the ImageNet pre-trained model. . . . .	68
4.6	<b>Predictions of unlabeled data using a supervised model.</b> We plot the distribution of the predictions of data from $U_{in}$ and $U_{out}$ . Specifically, we plot the maximum probability of the class predictions (left), entropy of the predictions (middle), and the distillation loss between the teacher and student model before the training starts (right). Unlabeled data from the same distribution tend to have a higher maximum probability, a lower entropy, or a higher distillation loss. . . . .	68
5.1	<b>Adding supervision on coarse label in the taxonomy improves semi-supervised learning.</b> <b>Left:</b> The Semi-iNat dataset has <b>labeled data</b> from <b>in-class species</b> and <b>coarsely labeled data</b> from both <b>in-class</b> and <b>out-of-class species</b> . The coarse labels include information in the kingdom and phylum levels. <b>Right:</b> Having more fine-grained labels improves the performance on both supervised baseline and semi-supervised methods. . . . .	71



5.2 **Confusion matrices on the phylum level.** **Left:** Supervised baseline model without coarse-label supervision. **Right:** FixMatch + hierarchical loss on the phylum level. Combining semi-supervised methods and hierarchical supervision on coarsely labeled data reduces the confusion between phyla. .... 79

# CHAPTER 1

## INTRODUCTION

Image recognition has shown significant progress in the last decade, given the development of deep neural networks. However, training neural network models usually require a large amount of labeled data which are not always available. For example, in applications such as fine-grained classification of animal species, annotating images needs domain experts, and its associated cost is expensive. In this thesis, we explore different research directions to make models more robust given limited labeled data.

One common practice is *transfer learning*, where the model is first pre-trained on a large generic labeled dataset such as ImageNet [92], then fine-tuned on the target task with limited labels. This has been a standard technique for computer vision tasks [43, 129, 89]. Another line of work is *few-shot learning*, which focuses on improving the robustness using labeled data only. Recent works on few-shot learning include generating more training data and using meta-learning methods to make models adapt to target tasks quickly. The third type of method leverages unlabeled data and uses *unsupervised learning* to learn the representation. This type of method is also called self-supervised learning in recent works. The next category is *semi-supervised learning*, which considers both few labeled data and unlabeled data at the same time. Last, one can utilize different modalities of the data (*e.g.*, 3D shape and rendered images) by using *multi-modal learning* methods.

In the rest of this chapter, we will provide an overview and related works on each of the five research directions. We first discuss transfer learning and its limitations on deep neural networks in Chapter 1.1. We then introduce the task of few-shot transfer learning and state-of-the-art methods, including meta-learning in Chapter 1.2. In Chapter 1.3, we discuss

recent works on unsupervised representation learning (or self-supervised learning) and how to combine it with transfer learning. In Chapter 1.4, we provide the background and recent works of semi-supervised learning. We then discuss cross-modal transfer learning and the technique of knowledge distillation in Chapter 1.5. Last, we summarize the contributions of the thesis in Chapter 1.6.

## 1.1 Transfer learning

In the past years, the progress on computer vision has been driven by various benchmarks such as ImageNet [36, 134]. In particular, a transfer learning paradigm, where the model is first *pre-trained* on ImageNet then *fine-tuned* on target task, has shown state-of-the-art results on various downstream tasks, including object detection [56, 68], image segmentation [102], and image categorization [43, 100]. Using pre-trained models largely reduces the convergence time at the fine-tuning stage compared to training from a randomly initialized model (training from scratch) [67]. However, transfer learning by fine-tuning is not always optimal [89]. It is also sensitive to hyper-parameters when the target dataset is small. Furthermore, even though recent papers showed that model pre-trained on larger scale datasets [157] can further improve the performance [87], how to select the best pre-trained model for the target task is also crucial. For example, one can choose the model based on the similarity of datasets or use the task embedding to select the pre-trained model [35, 187, 1].

In this work, we focus on the target task of fine-grained classification. In real-world applications such as recognizing species, class distributions are usually long-tailed. The “head” classes have many images while the “tail” classes only have few images. This is particularly hard for transfer learning as stated above. Current state-of-the-art methods often use class-balancing loss functions [34], normalize the weights of the classifier [81], or incorporate few-shot learning methods [101] to improve the accuracy of the tail classes. In the next section, we describe the details of few-shot learning.

## 1.2 Few-shot learning

Few-shot learning aims to learn representations with few labeled images that generalize well [45, 108]. Recently, a few-shot transfer learning setting has been proposed to simulate and evaluate few-shot learning [171]. Given a set of *base* classes with abundant images, the model is trained to better transfer to the *novel* classes where only a few images are available. To this end, several *meta-learning* approaches propose to evaluate representations by sampling many few-shot tasks within the domain of a dataset. These include optimization-based meta-learners, such as model-agnostic meta-learner (MAML) [46], gradient unrolling [128], closed-form solvers [12], and convex learners [99]. The second class of methods relies on distance-based classifiers such as matching networks [171] and prototypical networks (ProtoNet) [146], or modeling higher-order relationships in data using a graph network [50]. Another class of methods [53, 124, 125] model the mapping between training data and classifier weights using a feed-forward network.

While the literature is rapidly growing, a recent study by Chen *et al.* [27] has shown that the differences between meta-learners are diminished when deeper networks are used. They show that fine-tuning based transfer learning is a strong baseline for few-shot learning and the performance of ProtoNet [146] matches or surpasses several recently proposed meta-learners. In later works, transfer learning can be combined with transductive learning [39] or unsupervised-learning [54, 156, 30] to achieve better performances. We will introduce unsupervised learning and its related works in the next section.

Yet another line of work for few-shot learning is to generate more data by data augmentation. While image-based data augmentation such as scaling and color jittering are widely used for training deep networks, recent approaches employ complex strategies to generalize representations to novel classes in the few-shot setting. Data hallucination [65, 177] generate novel examples based on estimated relationships between clusters of image features. Antoniou *et al.* [2] use generative adversarial networks (GANs) to generate face data, while Schwartz *et al.* [138] generate examples based on a model of intra-class vari-

ance learned from pairs of images within a class. These techniques are complementary to the aforementioned meta-learning based methods.

### 1.3 Self-supervised learning

While unsupervised learning is a general term for machine learning algorithms, one of the goals of unsupervised learning is to find underlying structures from the data itself. After training is done, the model can generate feature representations that are useful for downstream tasks. This type of unsupervised *representation* learning is often called *self-supervised learning* (Self-SL) in recent works. In the rest of the thesis, we will use the term “self-supervised learning”. Self-SL is particularly useful in computer vision since images are easy to get but human annotations are expensive to acquire.

One class of Self-SL methods remove part of the visual data and task the network with predicting what has been removed from the rest in a discriminative manner [96, 119, 165, 192, 193]. Some other self-supervised tasks include predicting rotation [54], relative patch location [40], clusters [20, 21], and the number of objects [114], *etc.* On top of them, combining different tasks can be beneficial [41].

Recent works have shown that methods based on contrastive learning objectives [64] outperform the above-mentioned methods based on pretext tasks [44, 73, 7, 181, 162, 66, 25, 116, 59]. These contrastive objectives [64] are often expressed in terms of noise-contrastive estimation (NCE) [62] or maximizing mutual information [116, 73] between different augmentations of an image. However, the focus of most prior works on self-supervised learning is to supplant traditional supervised representation learning with unsupervised learning on large unlabeled datasets for downstream tasks.

Comparing to pre-text tasks in which the self-supervised representations lag behind the fully-supervised ones [57, 88], contrastive-based methods are catching up with the supervised performances [66, 25, 59]. Self-supervised learning can also be used to improve semi-supervised learning [188, 26], which we will describe next.

## 1.4 Semi-supervised learning

Semi-supervised learning (Semi-SL) considers the setting where we have a large set of unlabeled data in which a subset is labeled. It is a more realistic setting since we often have a limited budget for annotation while the data itself is cheaper to acquire. Methods on semi-supervised learning usually rely on the smoothness assumption: if two data points are close in high-density regions, they should have the same label [23]. One way to leverage the smoothness assumption is entropy minimization [58]. By encouraging the conditional probability of the class predictions to be either 0 or 1 in high-density regions, the class boundaries will be more accurate in low-density regions.

Another type of method focuses on the transductive setting: instead of learning a discriminative model on the data, it directly estimates the labels. This can be naturally applied to semi-supervised learning via the smoothness assumption [78]. Graph-based methods can also be used to propagate labels on unlabeled data [8].

The third type of method is self-training, which has been proposed decades ago [106, 139]. The idea is to first train a model on labeled data, then use the model to automatically generate labels for the unlabeled data. “Pseudo-labeling” [97] includes only confident predictions, *i.e.*, those greater than a threshold for training. The pseudo-labels can be added iteratively to induce a “curriculum” [22, 9, 63]. Other self-training methods involve re-training a “student model” from a “teacher model” using its prediction computed in different ways [183, 184, 195, 26]. For example, adding noise and using a larger student model [183, 195], selecting k-most confident pseudo-labels [184], or using a distillation loss which softens the predictions [183, 26].

Consistency-based methods learn by encouraging the consistency of the model’s predictions on the unlabeled data. This is related to an old idea of “co-training” [14, 126]: the different “views” of the same object (*e.g.* image and its caption) should be classified with the same label. In recent papers, the views are often referred to as different augmentations of the image [6, 127, 95, 136], including adversarial versions [109]. Al-

ternatively, consistency can be enforced across time, *e.g.*, using moving average of the predictions (*temporal ensembling* [95]), using the moving average of model parameters (*mean teacher* [160]), or using a stochastic averaging of model parameters [3]. A number of methods for *data augmentation* have been proposed which has generally improved both supervised and semi-supervised learning. These include the variety of image augmentations proposed in RandAugment [33], the CutOut scheme [38], linear combinations of images used in MixUp [189], and even augmentations in feature space [93]. These have been incorporated in methods such as MixMatch [11], ReMixMatch [10], FixMatch [148], UDA [182], and ICT [170] in different ways for consistency-based learning. While consistency via data-augmentation is effective when a model is trained from scratch, it is unclear if this is effective when using a pre-trained model, as invariance to these transformations may have been acquired during supervised pre-training.

Lastly, one can use self-supervised learning objectives to improve semi-supervised learning. These include incorporating pre-text tasks such as predicting image rotations [54], the order of patches (jigsaw puzzle task) [113] during semi-supervised learning [188, 156, 130]. Alternatively, self-supervised learning can be used as an initialization before training with labels. Based on the recent success of contrastive-based methods [66, 161, 25], it has shown promising results on semi-supervised ImageNet benchmark [26].

Despite the vast literature on semi-supervised learning, current benchmarks on semi-supervised image classification such as CIFAR, SVHN, and ImageNet are not realistic. Those semi-supervised datasets are collected from curated datasets without class imbalance and contains no images from novel classes. Furthermore, the benchmarks do not compare with transfer learning, which has shown to be a strong baseline on semi-supervised learning [115]. To this end, we collected two semi-supervised fine-grained classification datasets obtained by sampling classes from the Aves and Fungi taxonomy. In Chapter 4, we investigate different state-of-the-art semi-supervised methods, including the recent ones using self-supervised learning. We also investigate the effect of transfer learning and the effect

of out-of-distribution unlabeled data. In Chapter 5, we further improve semi-supervised learning by incorporating coarse labels of the hierarchical taxonomy.

## 1.5 Cross-modal learning and knowledge distillation

Data can have different modalities, *e.g.* videos have images and audio across time. Methods on cross-modal learning often learn a joint embedding space for different modalities [47, 111, 147]. In some applications, one modality can have richer information than others, *e.g.* a color image and a silhouette image. In other cases, different modalities can contain complementary information, *e.g.* color and depth images. One way to transfer the information between different representations is to use *knowledge distillation* [72]. It improves the performance of a simple classifier  $g$ , *e.g.* a shallow neural network, by imitating the outputs of a complex classifier  $f$ , *e.g.* a deep neural network. Distillation was first proposed for “model compression” [18] where simple classifiers such as linear models were trained to match the predictions of a decision-tree ensemble, leading to compact models. Distillation has been applied to transferring rich representations across modalities. For example, a model trained with images can be used to guide the learning of a model for depth images [61], or to a model for sound waves [4]. Distillation has also been used on cross-modal self-supervised learning [161], on few-shot learning [163], and on semi-supervised learning [26]. In this work, we show that distillation is useful for cross-modal learning (Chapter 2), and as self-training in semi-supervised learning (Chapter 4).

## 1.6 Contributions

In this thesis, we present our BMVC paper on cross-modal distillation [153] and ECCV workshop paper on 3D shape recognition using distillation [152] in Chapter 2. Chapter 3 includes our paper on boosting semi-supervised learning using self-supervised learning [156]. To investigate the effectiveness of existing semi-supervised methods, we first collected Semi-Aves [154] and Semi-iNat [155] datasets. We use these two datasets for the



Kaggle challenges in FGVC7 and FGVC8 workshops, which were held in CVPR 2020 and 2021. Chapter 4 presents the benchmark and analysis using our collected datasets, which is included in our CVPR paper [151]. Last, we use the coarse taxonomy labels collected in the Semi-iNat dataset to further improve semi-supervised learning, as presented in Chapter 5.

## CHAPTER 2

# CROSS-MODAL LEARNING USING KNOWLEDGE DISTILLATION

Transfer learning has been a standard technique for computer vision tasks. A common strategy is to initialize the weights from an ImageNet pre-trained model, then fine-tune the model on the target task through back-propagation. However, transfer learning is sensitive to hyper-parameters such as regularization when the target dataset is small [89]. In this chapter, we aim to improve transfer learning by cross-modal learning when having small target datasets.

In Chapter 2.1 to Chapter 2.3, we aim to build image recognition models that are robust to various forms of degradation, such as loss in resolution, lower signal-to-noise ratio, poor alignment, etc. When the target dataset consists of degraded images, simply fine-tuning does not work well. For example, the performance of existing models for fine-grained recognition drops rapidly when the resolution of the input image is reduced (see Table 2.1). While it is natural to expect a drop in accuracy since it may be impossible to infer certain visual properties from the low-quality data, we are interested in models whose performance degrades more gracefully as the signal degrades. Such systems will have a significant impact in a wide range of applications including those where poor quality data poses critical challenges such as surveillance, recognition of satellite imagery, face recognition for biometrics, etc. To this end, we propose a technique called *Cross Quality Distillation* (CQD) which utilizes the fact that in many scenarios abundant high-quality labeled data is available from which it is easy to automatically generate low-quality labeled data. However, instead of directly training a model on the low-quality data, which does not generalize well, we use the model trained on the high-quality data to guide the learning of a

model on the low-quality by forcing agreement between their predictions on each example (Figure 2.1). This guidance helps the second model generalize better on the low-quality data even though it does not have a direct access to the high-quality data at training or testing time.

In Chapter 2.4 to Chapter 2.5, we use knowledge distillation to improve 3D shape classification. A 3D shape can be represented as multi-view images, voxels, or point clouds. We first analyze the performance of using one type of modalities. We then apply distillation across modalities to see if one model can guide the learning of other models. Last, we improve the performance by combining representations learned from different modalities.

## 2.1 Knowledge Distillation

Assume that we have data in the form of  $(\mathbf{x}_i, \mathbf{z}_i, y_i)$ ,  $i = 1, 2, \dots, n$  where  $\mathbf{x}_i \in \mathcal{A}$  is the high-quality data,  $\mathbf{z}_i \in \mathcal{B}$  is the corresponding low-quality data, and  $y_i \in \mathcal{Y}$  is the target label. In practice only the high-quality data  $\mathbf{x}_i$  is needed since  $\mathbf{z}_i$  can be generated from  $\mathbf{x}_i$ . The idea of CQD is to first train a model  $f$  to predict the labels on the high-quality data and train a second model  $g$  on the low-quality data by forcing an agreement between their corresponding predictions by minimizing the following objective (Fig. 2.1):

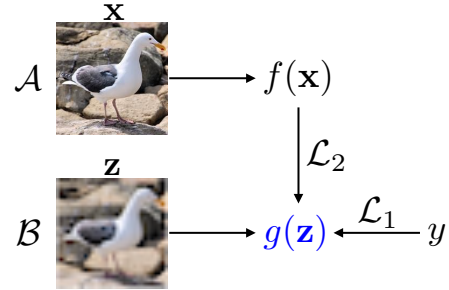


Figure 2.1: The CQD framework.

$$\sum_{i=1}^n \mathcal{L}_1(g(\mathbf{z}_i), y_i) + \lambda \sum_{i=1}^n \mathcal{L}_2(g(\mathbf{z}_i), f(\mathbf{x}_i)) + \mathcal{R}(g). \quad (2.1)$$

Here,  $\mathcal{L}_1$  and  $\mathcal{L}_2$  are loss functions,  $\lambda$  is a trade-off parameter, and  $\mathcal{R}(g)$  is a regularization term. The intuition for this objective is that by imitating the prediction of  $f$  on the high-quality data  $g$  can learn to generalize better on the low-quality data.

All our experiments are on multi-class classification datasets and we model both  $f$  and  $g$  using multi-layer CNNs, pre-trained on ImageNet dataset, with a final softmax layer to produce class probabilities  $\mathbf{p} = \sigma(\mathbf{z})$ , i.e.,  $p_k = e^{z_k} / \sum_j e^{z_j}$ . We use the cross-entropy loss  $\mathcal{L}_1(\mathbf{p}, \mathbf{q}) = \sum_i q_i \log p_i$ , and the cross-entropy of the predictions smoothed by a temperature parameter  $T$  for  $\mathcal{L}_2(\mathbf{p}, \mathbf{q}) = \mathcal{L}_1(\sigma(\log(\mathbf{p})/T), \sigma(\log(\mathbf{q})/T))$ . When  $T = 1$ , this reduces to the standard cross-entropy loss. We also found that squared-error between the logits ( $\mathbf{z}$ ) worked similarly. More details can be found in the experiments section.

## 2.2 Experiments on Image Classification

We show experiments on training Convolutional Neural Network (CNN) models for recognizing fine-grained categories of birds and cars in images of non-localized objects using localized ones (Figure 2.2a), low-resolution images using high-resolution images (Figure 2.2b), line drawings using color images (Figure 2.2c), and distorted images using non-distorted images (Figure 2.2d). This is a challenging task even on high-quality images, but the performance is often dramatically lower when applied to low-quality images. Our experiments show that CQD leads to significant improvements over a model trained on the low-quality data and other strong baselines for domain adaptation. The model works across a variety of tasks and domains without any task-specific customization. We also present preliminary experiments to address the case when the quality of the test data varies across examples, e.g. differing resolutions, using mixture models that lead to further gains.

We also relate CQD to existing approaches in the literature, spanning areas of domain adaptation, model compression [18, 72, 5], and learning with privileged information [168]. Our technique can be seen as a simple generalization of the model compression approach pioneered by Bucilă et al. [18], and revived recently in the context of CNNs by Hinton et al. [72] and Ba and Caruana [5]. In model compression, the goal is to learn a simple classifier that can imitate a more complex classifier on the same input, whereas in CQD the two functions operate on different inputs. However, the ideas are complementary and

can be combined. For example, we can use a deeper CNN trained on high-quality images to guide the learning of a shallower CNN on the low-quality images leading to even better generalization. Finally, we present insights into why the method works by relating it to the area of curriculum learning [9] and through visualizations of the learned models.

## 2.2.1 Experimental Setup

We begin by describing datasets, models, and training protocols used in our experiments. Chapter 2.3 describes the results of various experiments on CQD. Chapter 2.3.1 describes experiments for simultaneous quality distillation and model compression. Finally, Chapter 2.3.2 visualizes the distilled models to provide an intuition of why and how distillation works.

### 2.2.1.1 Datasets

We perform experiments on the CUB 200-2011 dataset [179] consisting of 11,788 images of 200 different bird species, and on the Stanford cars dataset [91] consisting of 16,185 images of 196 cars of different models and makes. Classification requires the ability to recognize fine-grained details which are impacted when the quality of the images is poor. Using the provided images and bounding-box annotations in these datasets, we create several cross-quality datasets which are described in detail in Chapter 2.3 and visualized in Figure 2.2. We use the training and test splits provided in the datasets.

### 2.2.1.2 Models

In our experiments, both  $f$  and  $g$  are based on CNNs pre-trained on the ImageNet dataset [36]. In particular we use `vgg-m` [24] and `vgg-vd` models [145] which obtain competitive performance on the ImageNet dataset. While there are better performing models for these tasks, *e.g.* those using novel model architectures [100, 32, 142, 77], and using additional annotations to train part and object detectors [15, 190, 191, 16], we perform ex-

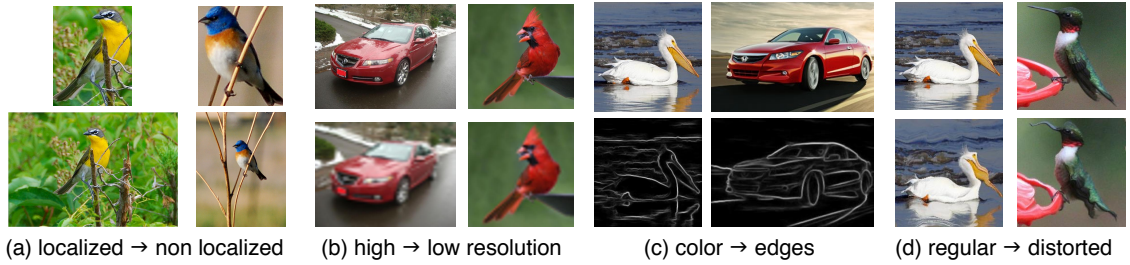


Figure 2.2: Examples of images from various cross-quality datasets used in our experiments. Images are from the birds [179] and cars dataset [91]. In each panel, the top row shows examples of the high-quality images and the bottom row shows examples of the corresponding low-quality images. These include (a) localized and non-localized images, (b) high- and low-resolution images, (c) color and edge images, and (d) regular and distorted images.

periments with simple models in the interest of a detailed analysis. However, we believe that our method is general and can be applied to other recognition architectures as well.

### 2.2.1.3 Methods

Below we describe various methods used in our experiments:

1. **Train on  $\mathcal{A}$ :** Starting from the ImageNet pre-trained model, we replace the 1000-way classifier (last layer) with a  $k$ -way classifier initialized randomly and then fine-tune the entire model with a small learning rate on domain  $\mathcal{A}$ . This is a standard way of transfer learning using deep models and has been successfully applied for a number of vision tasks including object detection, scene classification, semantic segmentation, texture recognition, and fine-grained classification [43, 129, 55, 31, 102, 110, 100].
2. **Train on  $\mathcal{B}$ :** Here we fine-tune the ImageNet pre-trained model on domain  $\mathcal{B}$ .
3. **Train on  $\mathcal{A} + \mathcal{B}$ :** Here we fine-tune the model on domain  $\mathcal{A}$  combined with domain  $\mathcal{B}$ . Data augmentation is commonly used while training CNNs to make them more robust.

4. **Train on  $\mathcal{A}$ , then train on  $\mathcal{B}$ :** This is a combination of  $\mathcal{A}$  and  $\mathcal{B}$  where the fine-tuning on domain  $\mathcal{B}$  is initialized from the model fine-tuned on domain  $\mathcal{A}$ . This “staged training” was recently proposed in [120] as a state-of-the-art technique for low-resolution image recognition. However, this method can only be applied when both  $f$  and  $g$  have the same structure. This is denoted by  $\mathcal{A}, \mathcal{B}$  in our experiments.
5. **Cross quality distillation (CQD):** Here we use a model  $f$  trained on domain  $\mathcal{A}$  (Method 1) to guide the learning of a second model  $g$  on domain  $\mathcal{B}$  using CQD (Equation 2.1). Like before, when  $f$  and  $g$  have an identical structure we can initialize  $g$  from  $f$  instead of the ImageNet model with random weights for the last layer.

**Optimization details** There are two parameters,  $T$  and  $\lambda$ , in the CQD model. The optimal value we found on validation set is  $T = 10$  for all experiments, and  $\lambda = 200$  for the CUB,  $\lambda = 50$  for the CARS dataset. The optimization in Equation 2.1 was solved using batch stochastic gradient descent, with learning rate starting from 0.0005 (0.0005 for CUB, 0.001 for CARS) changing linearly to 0.00005 after 30 epochs. Other parameters are as follows: momentum=0.9, weight decay=0.0005, batch size=128 (=32 when training vgg-vd). Instead of cross-entropy we also tried squared-distance on the logits  $\mathbf{z}$  as the loss function [5]. There was no significant difference between the two and we used cross-entropy for all our experiments. Our implementation is based on MatConvNet [169].

### 2.3 Results on Image Classification

We experiment with five different kinds of quality reduction to test the versatility of the approach. For each case, we report per-image accuracy on the test set provided in the dataset. Results using the vgg-m model for both function  $f$  and  $g$  are summarized in Table 2.1 and are described in detail below. The main conclusions are summarized at the end of this section.

Description	Method	Test	Local.	Resolution		Edge		Dist.	Local. + Res.
			CUB	CUB	CARS	CUB	CARS	CUB	CUB
Upper bound	$\mathcal{A}$	$\mathcal{A}$	67.0	67.0	59.3	67.0	59.3	67.0	67.0
No adaptation	$\mathcal{A}$	$\mathcal{B}$	57.4	39.4	7.6	1.9	4.2	49.7	24.9
Fine-tuning	$\mathcal{B}$	$\mathcal{B}$	60.8	61.0	41.6	29.2	45.5	58.4	46.2
Data augment.	$\mathcal{A} + \mathcal{B}$	$\mathcal{B}$	63.6	62.2	47.3	32.5	51.3	61.7	51.7
Staged training	$\mathcal{A}, \mathcal{B}$	$\mathcal{B}$	62.4	62.3	48.4	30.4	50.1	60.9	50.4
Proposed	CQD	$\mathcal{B}$	<b>64.4</b>	<b>64.4</b>	<b>48.8</b>	<b>34.1</b>	<b>51.5</b>	<b>63.0</b>	<b>52.7</b>

Table 2.1: **Cross quality distillation results.** Per-image accuracy on birds dataset (CUB) [179] and Stanford cars dataset (CARS) [91] for various methods and quality losses. All results are using  $f = g = \text{vgg-m}$  model. Training on  $\mathcal{A}$  and testing on  $\mathcal{A}$  is the upper bound of the performance in each setting (top row). Training on  $\mathcal{A}$  and testing on  $\mathcal{B}$  (no adaptation) often leads to a significant loss in performance. The proposed technique (CQD) outperforms fine-tuning ( $\mathcal{B}$ ), data augmentation ( $\mathcal{A} + \mathcal{B}$ ), and staged training ( $\mathcal{A}, \mathcal{B}$ ) [120] on all datasets.

**Localized to Non-localized Distillation.** To create the high-quality data, we use the provided bounding boxes in the CUB dataset to crop the object in each image. In this dataset, birds appear in various locations and scales and in clutter. Therefore,  $\text{vgg-m}$  trained and evaluated on the localized data obtains 67.0% accuracy, but when applied the non-localized data obtains only 57.4% accuracy (Table 2.1). When the model is trained on the non-localized data the performance improves to 60.8%. Staged training  $\mathcal{A}, \mathcal{B}$  improves the performance to 62.4%, but CQD improves further to 64.4%.

For this task, another baseline would be to train an object detector that first localizes the objects in images. For example, Krause *et al.* [90] report around 2.6% drop in accuracy (67.9%  $\rightarrow$  65.3%) when an R-CNN based object detector is used to estimate bounding-boxes of objects at test time instead of using true bounding-boxes (Table 2 in [90], CNN+GT BBox+ft vs. R-CNN+ft). Remarkably, using CQD we observe only 2.6% drop in performance (67.0%  $\rightarrow$  64.4%) without running any object detector. Moreover, our method only requires a single CNN evaluation and hence is faster. In Chap-



ter 2.3.2 we provide insights into why the distilled model performs better on non-localized images.

**High- to Low-Resolution Distillation.** Here we evaluate how models perform on images of various resolutions. For the CUB dataset, we use the localized images resized to  $224 \times 224$  for the high-resolution images, downsample to  $50 \times 50$ , and upsample to  $224 \times 224$  again for the low-resolution images. For the CARS dataset we do the same but for the entire image (bounding boxes are not used).

The domain shift leads to a large loss in performance here. On CUB the performance of the model trained on high-resolution data goes down from 67.0% to 39.4%, while the performance loss on CARS is even more dramatic going from 59.3% to a mere 7.6%. Man-made objects like cars contain high-frequency details such as brand logos, shapes of headlights, etc., which are hard to distinguish in low-resolution images. A model trained on the low-resolution images does much better, achieving 61.0% and 41.6% accuracy on birds and cars respectively. Color cues in the low-resolution are much more useful for distinguishing birds than cars which might explain the better performance on birds. Using CQD the performance improves further to 64.4% and 48.8% on the low-resolution data. On CARS the effect of both staged training and CQD is significant, leading to a more than 7% boost in performance.

**Color to Edges Distillation.** Recognizing line-drawings can be used for the retrieval of images and 3D shapes using sketches and has several applications in search and retrieval. As a proxy for line-drawings, we test the performance of various methods on edge images obtained by running the structured edge detector [42] on the color images. In contrast to low-resolution images, edge images contain no color information but preserve most of the high-frequency details. This is reflected in the better performance of the models on CARS than the CUB dataset (Table 2.1). Due to the larger domain shift, a model trained on color images performs poorly on edge images, obtaining 1.9% and 4.2% accuracy on CUB and CARS respectively.

Using CQD the performance improves significantly from 45.5% to 51.5% on CARS. On the CUB dataset, the performance also improves from 29.2% to 34.1%. The strong improvements on recognizing line drawings using distillation and staged training suggest that a better strategy to recognize line drawings of shapes used in various sketch-based retrieval applications [149, 175] is to first fine-tune the model on realistically rendered 3D models (*e.g.* with shading and texture) then distill the model to edge images.

**Non-distorted to Distorted Distillation.** Here the high-quality dataset is the localized bird images. To distort an image as seen in Figure 2.2d, we use the thin plate spline transformation with a uniform grid of  $14 \times 14$  control points. Each control point is mapped from a regular grid to a point randomly shifted by Gaussian distribution with zero mean and 4 pixels variance. Recognizing distorted images is challenging, and the performance of a model trained and evaluated on such images is 8.6% worse (67.0%  $\rightarrow$  58.4%). Using CQD the performance improves from 58.4% to 63.0%.

On this dataset, a baseline would be to remove the distortion by alignment methods such as congealing [75], or use a model that estimates deformations during learning, such as spatial transformer networks [77]. These methods are likely to work well but they require the knowledge of the space of transformations and are non-trivial to implement. On the other hand, CQD is able to nearly halve the drop in performance of the same CNN model without any knowledge of the nature of distortion and is easy to implement. Thus, CQD may be used whenever we can model the distortions algorithmically. For example, computer graphics techniques can be used to model the distortions from underwater imaging.

**Color to Non-localized and Low-Resolution Distillation.** Here the images have two different degradations at the same time: the low-quality data is low-resolution images with the object in clutter, where the high-quality data is high-resolution images cropped by the bounding boxes provided in the CUB dataset. Without adaptation, the performance drops to 24.9%, more than when only have one type of degradation (57.4% and 39.4% separately). We want to stress that the type of degradation in domain  $\mathcal{B}$  can be arbitrary, as long as we

have the instance-level correspondence between different domains which can be done by applying known transformations. As shown in the last column of Table 2.1, CQD improves 6.5% (46.2%  $\rightarrow$  52.7%) over fine-tuning.

**Summary.** In summary, we found that domain adaptation is critical since the performance of models trained on high-quality data is poor on the low-quality data. Data augmentation ( $\mathcal{A} + \mathcal{B}$ ) and staged training ( $\mathcal{A}, \mathcal{B}$ ) are quite effective, but CQD provides better improvements suggesting that adapting models on a per-example basis improves knowledge transfer across domains. CQD is robust and only requires setting a handful of parameters, such as  $T$  and  $\lambda$ , across a wide variety of quality losses. In most cases, CQD cuts the performance gap between the high- and low-quality data in half.

### 2.3.1 Simultaneous CQD and Model Compression

In this section, we experiment if a deeper CNN trained on high-quality data can be distilled to a shallow CNN trained on the low-quality data. This is the most general version of CQD where both the domains and functions  $f, g$  change. The formulation in Equation 2.1 does not require  $f$  and  $g$  to be identical. However,  $\mathcal{A} + \mathcal{B}$  and  $\mathcal{A}, \mathcal{B}$  baselines cannot be applied here.

We perform experiments on the CUB dataset using localized and non-localized images described earlier. The deeper CNN is the sixteen-layer “very deep” model (vgg-vd) and the shallow CNN is the five-layer vgg-m model used in the experiments so far. The optimal parameters obtained on the validation set for this setting were  $T = 10, \lambda = 50$ .

The results are shown in Table 2.2. The first row contains results using CQD for vgg-m model which is copied from Table 2.1 for ease of comparison. The third row shows the same results using the vgg-vd model. The accuracy is higher across all tasks. CQD leads to an improvement of 2.9% (69.5%  $\rightarrow$  72.4%) for the deeper model. The middle row shows results for training the vgg-m model on non-localized images from a vgg-vd model trained on the localized images. This leads to a further improvement of 0.9% (63.7%  $\rightarrow$

$f \rightarrow g$	training $\rightarrow$ testing		
	$\mathcal{A} \rightarrow \mathcal{A}$	$\mathcal{B} \rightarrow \mathcal{B}$	CQD $\rightarrow \mathcal{B}$
vgg-m $\rightarrow$ vgg-m	67.0	60.8	63.7
vgg-vd $\rightarrow$ vgg-m	-	-	64.6
vgg-vd $\rightarrow$ vgg-vd	74.9	69.5	72.4

Table 2.2: Accuracy of various techniques on the CUB localized/non-localized dataset.

64.6%) suggesting that model compression and cross quality distillation can be seamlessly combined.

## 2.3.2 Understanding Why CQD Works

### 2.3.2.1 Relation to curriculum learning

Curriculum learning is the idea that models generalize better when training examples are presented in the order of their difficulty. Bengio *et al.* [9] showed a variety of experiments where non-convex learners reach better optima when more difficult examples are introduced gradually. In one experiment a neural network was trained to recognize shapes. There were two kinds of shapes: `BasicShapes` which are canonical circles, squares, and triangles, and `GeomShapes` which are affine distortions of the `BasicShapes` on more complex backgrounds. When evaluated only on the test set of `GeomShapes`, the model first trained on `BasicShapes` then fine-tuned on `GeomShapes`, performed better than the model only trained on `GeomShapes`, or the one trained with a random ordering of both types of shapes.

We observe a similar phenomenon when training CNNs on low-quality data. For example, on the CARS dataset, staged training [120]  $\mathcal{A}, \mathcal{B}$  outperforms the model trained on low-resolution data  $\mathcal{B}$ , when evaluated on the low-resolution data  $\mathcal{B}$  (48.4% vs. 41.6%). Since low-quality data is more difficult to recognize, introducing them gradually might explain the better performance of the staged training and CQD techniques. Additional ben-

efits of CQD come from the fact that paired high- and low-quality images allowing better knowledge transfer through distillation.

### 2.3.2.2 Understanding CQD through gradient visualizations

Here we investigate how the knowledge transfer occurs between a model trained on localized images and non-localized images. Our intuition is that by trying to mimic the model trained on the localized images a model must learn to ignore the background clutter. In order to verify this, we compute the gradient of log-likelihood of the true label of an image with respect to the image using the CQD model and  $\mathcal{B}$  model, both are trained only on non-localized images. Figure 2.3-left shows the gradients for two different images. The darkness of each pixel  $i$  is proportional to the norm of the gradient vector at that pixel  $\|G_i\|_2$ ,  $G_i = [G_i^r, G_i^g, G_i^b]$  for  $r, g, b$  color channels. The gradients of the CQD model are more contained within the bounding box of the object, suggesting a better invariance to background clutter. As a further investigation we compute the fraction of gradients within the box:  $\tau = (\sum_{i \in \text{box}} \|G_i\|_2) / (\sum_{i \in \text{image}} \|G_i\|_2)$ . This ratio is a measure of how localized the relevant features are within the bounding box. A model based on a perfect object detector will have  $\tau = 1$ . We compute  $\tau$  for 1000 images for both CQD and  $\mathcal{B}$  models and visualize them on a scatter plot as seen in Figure 2.3-right. On average the CQD model has higher  $\tau$  than  $\mathcal{B}$  model, confirming our intuition that the CQD model is implicitly able to localize objects.

## 2.4 Experiments on 3D Shape Classification

We have shown that distillation can be used for training across images with different qualities. In the section, we extent the idea of cross-modal distillation to the 3D shape classification task. Techniques for analyzing 3D shapes are becoming increasingly important due to the vast number of sensors that are capturing 3D data, as well as numerous computer graphics applications. In recent years a variety of deep architectures have been approached

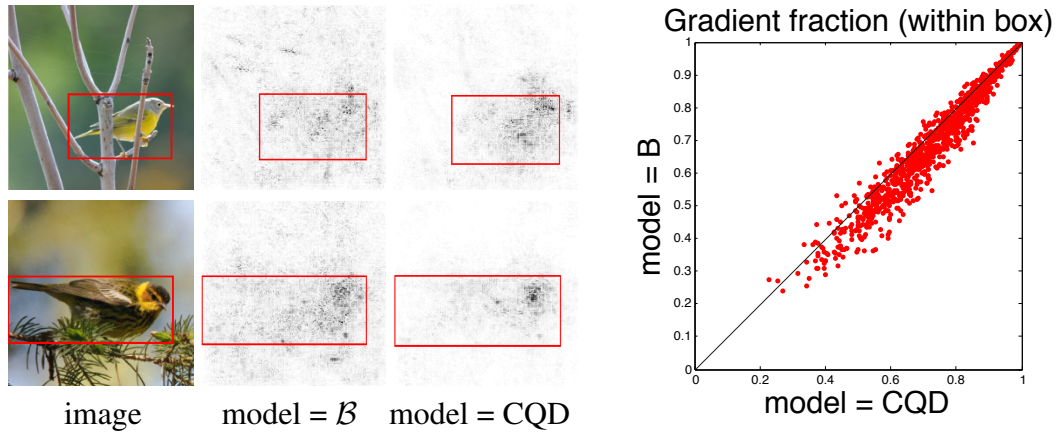


Figure 2.3: **Left:** Image and gradient of the image with respect to the true class label for the model trained on  $\mathcal{B}$  (non-localized images) and CQD (from a model trained on localized images). Darker pixels represent higher gradient value. The gradients of the model trained using CQD are more focused on the foreground object. **Right:** The scatter plot of the fraction of total gradient within the bounding-box for 1000 training images for the two models.

for classifying 3D shapes. These range from *multiview* approaches that render a shape from a set of views and deploy image-based classifiers, to *voxel*-based approaches that analyze shapes represented as a 3D occupancy grid, to *point*-based approaches that classify shapes represented as collection of points. However, there is relatively little work that studies the tradeoffs offered by these modalities and their associated techniques.

We first see the performance of using one type of the representations (modality), including multi-view images, voxels, and point clouds. We also experiment on few-shot learning setting where only few examples are available (Chapter 2.5.1). For image-based models, we further investigate different types of renderings and the effect of ImageNet pre-training (Chapter 2.5.2). Next, we apply distillation across modalities to see if one representation can guide the training for another. We also investigate if different representations are complementary to each other (Chapter 2.5.3).

We pick a representative technique for each modality. For multiview representation we choose the Multiview CNN (MVCNN) architecture [149]; For voxel-based representation

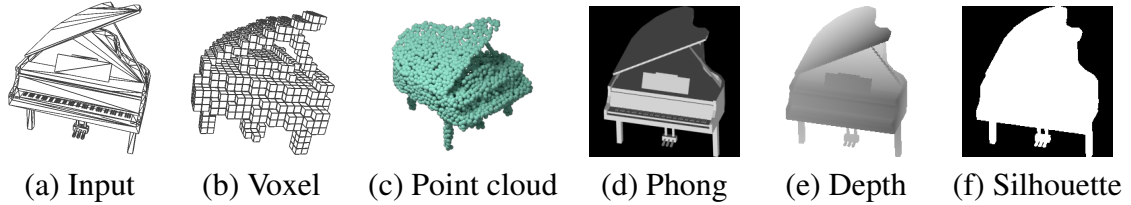


Figure 2.4: Different representations of the input shape. From the left (a) the shapes in the database are represented as triangle meshes, (b) the shape converted to a  $30^3$  voxel grid, (c) point cloud representation with 2048 points, and (d-f) the model rendered using Phong shading, and as depth and binary silhouette images respectively.

we choose the VoxNet [105, 180] constructed using convolutions and pooling operations on a 3D grid; For point-based representation we choose the PointNet architecture [150]. The analysis is done on the widely-used ModelNet40 shape classification benchmark [180].

## 2.4.1 Experimental Setup

### 2.4.1.1 Classification benchmark.

All our evaluation is done on the ModelNet40 shape classification benchmark [180] following the standard training and test splits provided in the dataset. There are 40 categories with 9483 training models and 2468 test models. The numbers of models are not equal across classes hence we report both the per-instance and per-class accuracy on the test set. While most of the literature report results by training on the entire training set, some earlier work, notably [149] reports results on training and evaluation on a subset consisting of 80 training and 20 test examples per category.

### 2.4.1.2 Input representations.

The dataset presents each shape as a collection of triangles, hence it is important to describe the exact way in which these are converted to point clouds, voxels, and images for input to different network architectures. These inputs are visualized in Figure 2.4 and described below:

- **Voxel representation.** To get voxel representations we follow the dataset from [122] where models are discretized to a  $30 \times 30 \times 30$  occupancy grid. The data is available from the author’s page.
- **Point cloud.** For point cloud representation we use the data from the PointNet approach [150] where 2048 points are uniformly sampled for each model.
- **Image representation.** To generate multiple views of the model we use a setup similar to [149]. Since the models are assumed to be upright oriented a set of virtual cameras are placed at 12 radially symmetric locations, i.e. every 30 degrees, facing the object center and at an elevation of 30 degrees. Comparing to [149], we render the images with black background and set the field-of-view of the camera such that the object is bounded by image canvas and rendered as an image of size  $224 \times 224$ . A similar scheme was used to generate views for semantic segmentation of shapes in the Shape PFCN approach [79]. This had a non-negligible impact on the performance of the downstream models as discussed in Chapter 2.5.1. Given the setup we considered three different ways to render the models described below:

1. Phong shading, where images are rendered with the Phong reflection model [121] using Blender software [13]. The light and material setup is similar to the approach in [149].
2. Depth rendering, where only the depth value is recorded.
3. Silhouette rendering, where images are rendered as binary images for pixels corresponding to foreground.

### 2.4.1.3 Data augmentation.

Models in the dataset are upright oriented, but not consistently oriented along the axis, *i.e.*, models could be rotated arbitrarily along the upright direction. Models that rely on voxel or point cloud input often benefit from rotation augmentation along the upright axis



during training and testing. Similar to the multiview setting we consider models rotated by 30 degree increments as additional data during training, and optionally aggregating votes across these instances at test time.

#### 2.4.1.4 Classification Architectures

We consider the following deep architectures for shape classification. **Multiview CNN (MVCNN)** The MVCNN architecture [149] uses rendered images of the model from different views as input. Each image is fed into a CNN with shared weights. A max-pooling layer across different views is used to perform an orderless aggregation of the individual representations followed by several non-linear layers for classification. While the original paper [149] used the VGG-M network [145] we also report results using:

- The VGG-11 network, which is the model with configuration A from [145]. The view-pooling layer is added before the first  $fc$  layer.
- Variants of residual networks proposed in [69] such as ResNet18, ResNet34, and ResNet50. The view-pooling layer is added before the final  $fc$  layer.

**Voxel network (VoxNet)** The VoxNet was first proposed in several early works [105, 180] that uses convolution and pooling layers defined on 3D voxel grids. The early VoxNet models [105] used two 3D convolutional layers and 2 fully-connected layers. In our initial experiments we found the capacity of this network is limited. We also experimented with the deeper VoxNet architecture proposed in [150] which has five blocks of (conv3d-batchnorm-LeakyReLU) and includes batch normalization [76]. All conv3d layers have kernel size 5, stride 1 and channel size 32. The LeakyReLU has slope 0.1. Two fully-connected layers ( $fc$ -batchnorm-ReLU- $fc$ ) are added on top to obtain class predictions.

**Point network (PointNet)** We follow the same architecture as PointNet [150] that operates on point cloud representations of a model. The architecture applies a series of non-linear mappings individually to each input point and performs orderless aggregations using

max-pooling operations. Thus the model is invariant to the order in which the points are presented and can directly operate on point clouds without additional preprocessing such as spatial partitioning, or graph construction. Additionally, some initial layers are used to perform spatial transformations (rotation, scaling, etc.) Despite its simplicity the model and its variants have been shown to be effective at shape classification and segmentation tasks [150, 123].

**Training details.** All the MVCNN models are trained in two stages as suggested in [149]. The model is first trained as a single-image classification task where the view-pooling layer is removed, then trained to jointly classify all the views with view-pooling layer in the second stage. We use the Adam optimizer [84] with learning rate  $5 \times 10^{-5}$  and  $1 \times 10^{-5}$  for first and second stage respectively and each stage is trained with 30 epochs. The batch size is set to 64 and 96 (eight models with twelve views) for each stage and the weight decay parameter is set to 0.001. The VoxNet is trained with Adam optimizer with learning rate  $1 \times 10^{-3}$  for 150 epochs. The batch size is set to 64 and weight decay parameter is 0.001.

## 2.5 Results on 3D Shape Classification

We begin by investigating the model generalization in Chapter 2.5.1. Chapter 2.5.2 analyzes the effect of different architectures and renderings for the MVCNN. Chapter 2.5.3 uses cross-modal distillation to improve the performance of VoxNet and PointNet. Chapter 2.5.4 compares the tradeoffs between different representations. Finally, Chapter 2.5.5 puts the results presented in this paper in the context of prior work.

### 2.5.1 Learning From a Few Examples

One of the most desirable properties of a classifier is its ability to generalize from a few examples. We test this ability by evaluating the accuracy of different models as a function of training set size. We select the first  $M_k$  models in the training set for each class, where

$$M_k = \min(N_k, \{10, 20, 40, 80, 160, 320, 889\}),$$

and  $N_k$  is the number of models in class  $k$ . The maximum number of models per-class in the training set of ModelNet40 is 889. Figure 2.5 shows the per-class and per-instance accuracy for three different models as a function of the training set size. The MVCNN with the VGG-11 architecture has better generalization than VoxNet and PointNet across all training set sizes. MVCNN obtains 77.8% accuracy using only 10 training models per class, while PointNet and VoxNet obtain 62.5% and 57.9% respectively. The performance of MVCNN is near optimal with 160 models per class, far fewer than PointNet and VoxNet. When using the whole dataset for training, MVCNN (95.0%) outperforms PointNet (89.1%) and VoxNet (85.6%) by a large margin.

Several improvements have been proposed for both point-based and voxel-based architectures. The best performing point-based models to the best of our knowledge is the Kd-Networks [85] which achieves 91.8% per-instance accuracy. For voxel-based models, O-CNN [176] uses sparse convolutions to handle higher resolution with Octave trees [107] and achieves 90.6% per-instance accuracy. However, all of them are far below the MVCNN approach. More details and comparison to the state-of-the-art are in Chapter 2.5.5.

## 2.5.2 Dissecting the MVCNN Architecture

Given the high performance of MVCNN we investigate what factors contribute to its performance as described next.

### 2.5.2.1 Effect of model architecture.

The MVCNN model in [149] used VGG-M architecture. However a number of different image networks have since been proposed. We used different CNN architectures for MVCNN and report the accuracies in Table 2.3. All models have similar performance suggesting that MVCNN is robust across different CNN architectures. In Table 2.5 we also

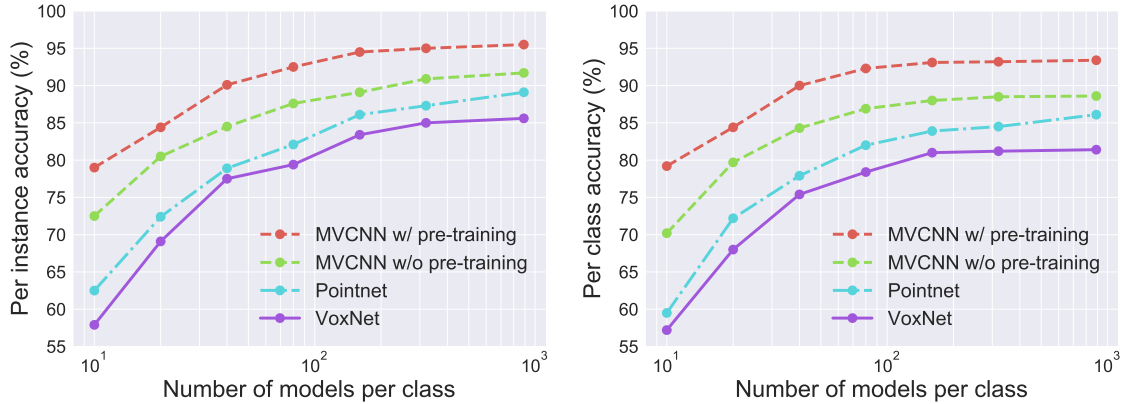


Figure 2.5: Classification accuracy as a function of training set size. MVCNN generalizes better than the other approaches. The two MVCNN curves correspond to variants with and without ImageNet pretraining.

compare with the results using VGG-M and AlexNet. With the same shaded images and training subset, VGG-11 achieves 89.1% and VGG-M has 89.9% accuracy.

Model	Per class acc.	Per instance acc.
VGG-11	92.4	95.0
ResNet 18	92.8	95.6
ResNet 34	93.4	95.9
ResNet 50	94.0	95.5

Table 2.3: Accuracy (%) of MVCNN with different CNN architectures. The VGG-11 architectures are on par with the residual network variants.

### 2.5.2.2 Effect of ImageNet pretraining.

MVCNN benefits from transfer learning from ImageNet classification task. However, even without ImageNet pretraining, the MVCNN achieves 91.3% per-instance accuracy (Table 2.4). This is higher than several point-based and voxel-based approaches. Figure 2.5 plots the performance of the MVCNN with VGG-11 network without ImageNet pretraining across training set sizes showing this trend is true throughout the training regime. In Chapter 2.5.3 we study if ImageNet pretraining can benefit such approaches using cross-modal transfer learning.

Model	Per class acc.	Per instance acc.
VGG-11 w/ ImageNet pretraining	92.4	95.0
VGG-11 w/o ImageNet pretraining	88.7	91.3

Table 2.4: Effect of ImageNet pretraining on the accuracy (%) of MVCNN. The VGG-11 architecture is used and the full training/test split of the ModelNet40 dataset is used.

### 2.5.2.3 Effect of shape rendering.

We analyze the effect of different rendering approaches for input to a MVCNN model in Table 2.5. Sphere rendering proposed in [122] refers to rendering each point as a sphere and was shown to improve performance with AlexNet MVCNN architectures. We first compared the tight field-of-view rendering with black background in this work to the rendering in [149]. Since [149] only reported results on the 80/20 training/test split, we first compared the performance of the VGG-11 networks using images from [149]. The performance difference was negligible. However with our shaded images the performance of the VGG-11 network improves by more than 2%.

Using depth images, the per instance accuracy is 3.4% lower than using shaded images, but concatenating shaded images with depth images gives 1.2% improvement. Furthermore, we found the shading information only provides 1.4% improvements over the binary silhouette images. This suggests that most of the discriminative shape information used by the MVCNN approaches lie in the boundary of the object.

### 2.5.3 Cross-Modal Distillation

Knowledge distillation [18, 72] was proposed for model compression tasks. They showed the performance of the model can be improved by training to imitate the output of a more accurate model. This technique has also been applied on transferring rich representations across modalities. For example, a model trained with images can be used to guide learning of a model for depth images [61], or to a model for sound waves [4]. We

Model	Rendering	Full training/test		80/20 training/test	
		Per class	Per instance	Per class	Per instance
VGG-M	Shaded from [149]	-	-	89.9	89.9
VGG-M	Shaded from [149] (80×)	-	-	90.1	90.1
VGG-11	Shaded from [149]	-	-	89.1	89.1
VGG-11	Shaded	<b>92.4</b>	<b>95.0</b>	<b>92.4</b>	<b>92.4</b>
VGG-11	Depth	89.8	91.6		
VGG-11	Shaded + Depth	94.7	96.2		
VGG-11	Silhouettes	90.7	93.6		
AlexNet	Sphere rendering (20×)	89.7	92.0		
AlexNet-MR	Sphere rendering (20×)	91.4	93.8		

Table 2.5: Accuracy (%) of MVCNN with different rendering methods. The number in the brackets are the number of views used. 12 views are used if not specified.

investigate such techniques for learning across different 3D representations; Specifically from MVCNN model to PointNet and VoxNet models.

To do this we first train the ImageNet initialized VGG-11 network on the full training set. The logits (the last layer before the softmax) are extracted on the training set. A PointNet (or VoxNet) model is then trained to minimize

$$\sum_{i=1}^n \mathcal{L}(\sigma(\mathbf{z}_i), y_i) + \lambda \sum_{i=1}^n \mathcal{L}\left(\sigma\left(\frac{\mathbf{z}_i}{T}\right), \sigma\left(\frac{\mathbf{x}_i}{T}\right)\right) \quad (2.2)$$

where  $\mathbf{x}_i$  and  $\mathbf{z}_i$  are the logits from the MVCNN model and from the model being trained respectively,  $y_i$  is the class label of the input  $s_i$ ,  $\sigma(x)$  is the softmax function, and  $\mathcal{L}$  is the cross-entropy loss  $\mathcal{L}(p, q) = -\sum p_i \log q_i$ .  $T$  is the temperature for smoothing the targets.  $\lambda, T$  are set by grid search for  $T \in [1, 20]$ ,  $\lambda \in [1, 100]$ . For example, in PointNet the best hyper-parameters are  $T = 20, \lambda = 50$  when training set is small, and  $T = 15, \lambda = 10$  when the training set is larger. In VoxNet we set  $T = 10, \lambda = 100$  in all cases. Figure 2.6 shows the result of training VoxNet and PointNet with distillation. For VoxNet the per instance accuracy is improved from 85.6% to 87.4% with whole training set; For PointNet the accuracy is improved from 89.1% to 89.4%. The improvement is slightly bigger when there is less training data.

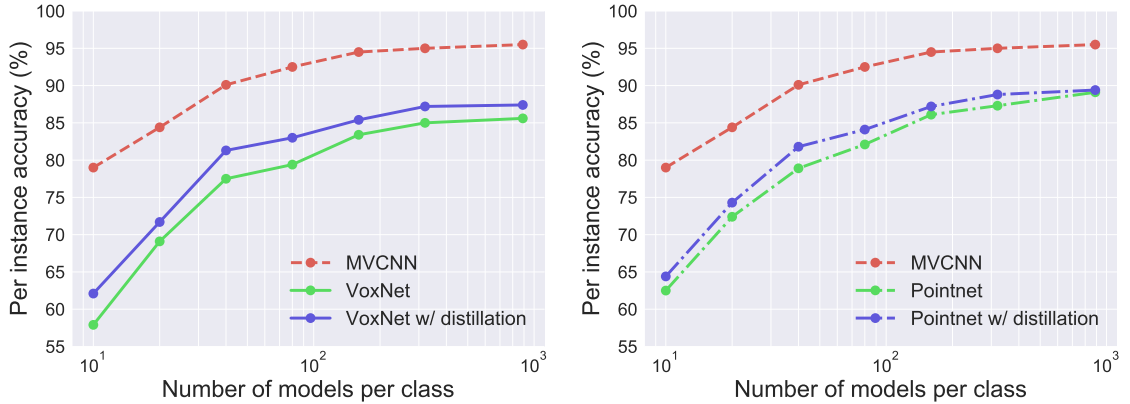


Figure 2.6: Model distillation from MVCNN to VoxNet and PointNet. The accuracy is improved by 1.8% for VoxNet and 0.3% for PointNet with whole training set.

### 2.5.4 Tradeoffs between Learned Representations

In this section, we analyze the tradeoffs between the different shape classifiers. Table 2.6 compares their speed, memory, and accuracy. The MVCNN model has more parameters and is slower, but the accuracy is 5.9% better than PointNet and 9.4% better than VoxNet. Even though the number of FLOPS are far higher for MVCNN the relative efficiency of 2D convolutions results in slightly longer evaluation time compared to VoxNet and PointNet.

We further use an ensemble model combining images, voxels, and point cloud representations. A simple way is to average the predictions from different models. As shown in Figure 2.7, the ensemble of VoxNet and PointNet has better performance than using single model. However, the predictions from MVCNN dominate VoxNet and PointNet and gives no benefit for combining the predictions from other models with MVCNN. A more complex scheme where we trained a linear model on top of features extracted from penultimate layers of these networks did not provide any improvements either.

Model	Forward time	# params	Memory (GB)	Per class acc. (%)	Per ins. acc. (%)
MVCNN	25.8 ms	128.9M	10.0	92.4	95.0
VoxNet	1.3 ms	1.4M	2.0	81.4 (82.5)	85.6 (87.4)
PointNet	3.1 ms	3.5M	4.4	86.1 (86.7)	89.1 (89.4)

Table 2.6: Accuracy, speed and memory comparison of different models. Memory usage during training which includes parameters, gradients, and layer activations, for a batch size 64 is shown. Forward-pass time is also calculated with batch size 64 using PyTorch with a single GTX Titan X for all the models. The input resolutions are  $224 \times 224$  for MVCNN,  $32^3$  for VoxNet, and 1024 points for PointNet. The accuracy numbers in brackets are for models trained with distillation as described in Chapter 2.5.3.

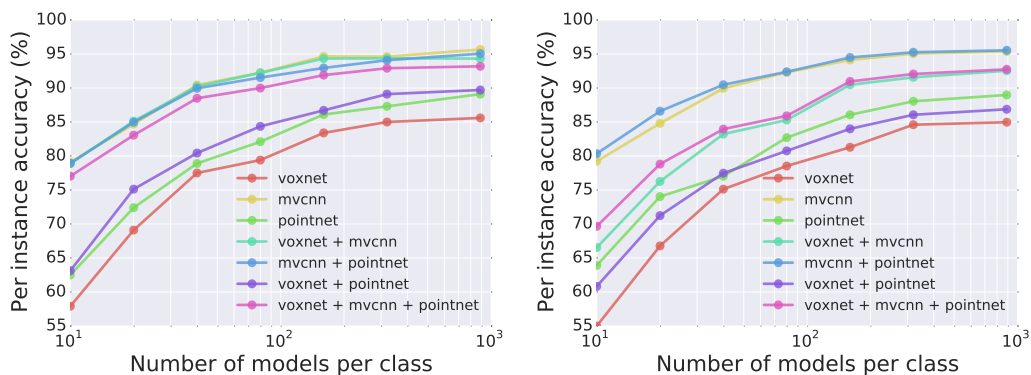


Figure 2.7: Accuracy obtained by ensembling models. Left shows results by averaging the predictions while right shows results by training a linear model on the concatenated features extracted from different models.

## 2.5.5 Comparison to Prior Work

We compare our MVCNN result with prior works in Table 2.7. The results are grouped by the input type. For multi-view image-based models, our MVCNN achieves 95.0% per instance accuracy, which is the best result between all competing approaches. Rotation Net [80], which predicts the object pose and class labels at the same time, is 0.2% worse than our MVCNN. Dominant Set Clustering [173] works by clustering image features across views and pooling within the clusters. Its performance is 1.0% lower than Rotation-Net. MVCNN-MultiRes [122] is the most related to our work. They showed that MVCNN with sphere rendering can achieve better accuracy than voxel-based network, suggesting



that there is room for improvement in VoxNet. Our VoxNet experiment corroborates to this conclusion. Furthermore, MVCNN-MultiRes uses images in multiple resolutions to boost its performance.

For point-based methods, PointNet [150] and DeepSets [186] use symmetric functions, i.e. max/mean pooling layers, to generate permutation invariant point cloud descriptions. DynamicGraph [178] builds upon PointNet by performing symmetric function aggregations on points within a neighborhood, instead of the whole set. Such neighborhood is computed dynamically by building nearest neighbor graph using distances defined in the feature space. Similarly, Kd-Networks [85] work by precomputing a graph induced by a binary spatial partitioning tree and use it to apply local linear operations. The best point-based method is 2.8% less accurate than our MVCNN.

For voxel-based methods, VoxNet [105] and 3DShapeNets [180] work by applying 3D convolutions on voxels. ORION [140] is based on VoxNet but predicts the orientation in addition to class labels. OctNet [132] and O-CNN [176] are able to process higher resolution grids by using an octree representation. FusionNet [71] combines the voxel and image representations to improve the performance to 90.8%. Our experiments in Chapter 2.5.4 suggests that since MVCNN already has 95.0% accuracy the benefit of combining different representations is not effective.

Model	Input	Per class acc.	Per ins. acc.
<b>Our MVCNN</b>		<b>92.4</b>	<b>95.0</b>
RotationNet [80]		-	94.8
Dominant Set Clustering [173]	Images	-	93.8
MVCNN-MultiRes [122]		91.4	93.8
PANORAMA-NN [143]			90.7
MVCNN [149]		90.1	90.1
DynamicGraph [178]			90.2
Kd-Networks [85]	PC	-	91.8
LocalFeatureNet [144]		-	90.8
PointNet++ [123]		-	90.7
DeepSets [186]		-	90.0
PointNet [150]		86.2	89.2
VRN Single [17]			-
O-CNN [176]	Voxels		90.6
ORION [140]		-	89.7
VoxNet [105]		-	83.0
3DShapeNets [180]		77.3	84.7
PointNet++ [123]		PC+Normal	-
FusionNet [71]	Voxels+Images	-	90.8

Table 2.7: Accuracy (%) of state-of-the-art methods with different 3D representations. PC refers to point clouds. The order is grouped by input type and sorted by accuracy.

## CHAPTER 3

# IMPROVING FEW-SHOT LEARNING WITH SELF-SUPERVISED LEARNING

We have shown that transfer learning on small datasets can be improved via cross-modal distillation; however, most of the time we only have images as training data. Current machine learning methods based on deep neural networks need enormous amounts of training data to learn new tasks. Even with transfer learning, the performances on few-shot classes are hard to improve. This is an issue for many practical problems across domains such as biology and medicine where labeled data is hard to come by. In contrast, we humans can quickly learn new concepts from limited training data by relying on our past “visual experience”. Recent work attempts to emulate this by training a feature representation to classify a training dataset of “base” classes with the hope that the resulting representation generalizes not just to unseen examples of the same classes but also to novel classes, which may have very few training examples (called few-shot learning). However, training for base class classification can force the network to only encode features that are useful for distinguishing between base classes. In the process, it might discard semantic information that is irrelevant for base classes but critical for novel classes. This might be especially true when the base dataset is small or when the class distinctions are challenging.

One way to recover this useful semantic information is to leverage representation learning techniques that do not use class labels, namely, *unsupervised* or *self-supervised learning*. The key idea is to learn about statistical regularities within images, such as the spatial relationship between patches, or its orientation, that might be a cue to semantics. Despite recent advances, these techniques have only been applied to a few domains (*e.g.*, entry-level

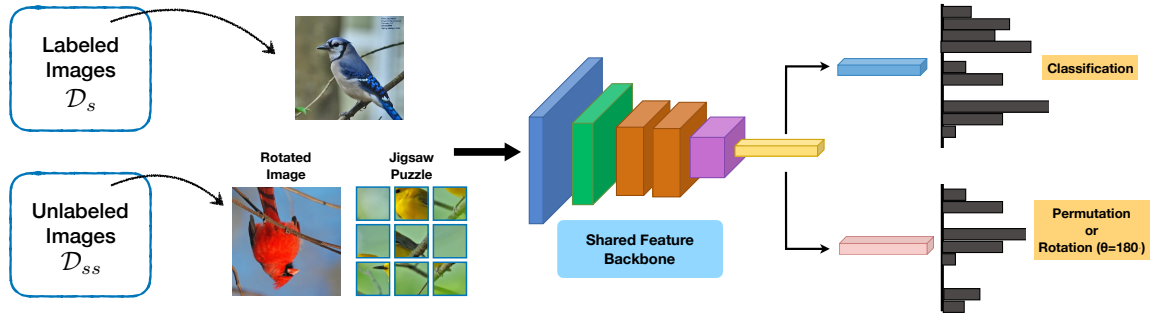


Figure 3.1: **Combining supervised and self-supervised losses for few-shot learning.** Self-supervised tasks such as jigsaw puzzle or rotation prediction act as a data-dependent regularizer for the shared feature backbone. Our work investigates how the performance on the *target task domain* ( $\mathcal{D}_s$ ) is impacted by the choice of the *domain used for self-supervision* ( $\mathcal{D}_{ss}$ ).

classes on internet imagery), and under the assumption that large amounts of unlabeled images are available. Their applicability to the general few-shot scenario is unclear. In particular, can these techniques prevent overfitting to base classes and improve performance on novel classes in the few-shot setting? If so, does the benefit generalize across domains and to more challenging tasks? Moreover, can self-supervision boost performance in domains where even unlabeled images are hard to get?

In this chapter we seek to answer these questions. We show that with *no additional training data*, adding a self-supervised task as an auxiliary task (Fig. 3.1) improves the performance of existing few-shot techniques on benchmarks across a multitude of domains (Fig. 3.2), in agreement with conclusions from similar recent work [52]. Intriguingly, we find that the benefits of self-supervision *increase* with the difficulty of the task, for example when training from a smaller base dataset, or with degraded inputs such as low resolution or grayscale images (Fig. 3.3).

One might surmise that as with traditional SSL, additional unlabeled images might improve performance further. But what unlabeled images should we use for novel problem domains where unlabeled data is not freely available? To answer this, we conduct a series of experiments with additional unlabeled data from different domains. We find that

adding more unlabeled images improves performance *only* when the images used for self-supervision are within the *same domain* as the base classes (Fig. 3.4a); otherwise, they can even *negatively* impact the performance of the few-shot learner (Fig. 3.4b). Based on this analysis, we present a simple approach that uses a domain classifier to pick similar-domain unlabeled images for self-supervision from a large and generic pool of images (Fig. 3.5). The resulting method improves over the performance of a model trained with self-supervised learning from images within the dataset (Fig. 3.6). Taken together, this results in a powerful, general, and practical approach for improving few-shot learning on small datasets in novel domains.

### 3.1 Method

We adopt the commonly used setup for few-shot learning where one is provided with labeled training data for a set of *base* classes  $\mathcal{D}_b$  and a much smaller training set (typically 1-5 examples per class) for *novel* classes  $\mathcal{D}_n$ . The goal of the few-shot learner is to learn representations on the base classes that lead to good generalization on novel classes. Although in theory the base classes are assumed to have a large number of labeled examples, in practice this number can be quite small for novel or fine-grained domains, *e.g.* less than 5000 images for the birds dataset [179], making it challenging to learn a generalizable representation.

Our framework, as seen in Fig. 3.1, combines *meta-learning* approaches for few-shot learning with *self-supervised learning*. Denote a labeled training dataset  $\mathcal{D}_s$  as  $\{(x_i, y_i)\}_{i=1}^n$  consisting of pairs of images  $x_i \in \mathcal{X}$  and labels  $y_i \in \mathcal{Y}$ . A feed-forward convolutional network  $f(x)$  maps the input to an embedding space which is then mapped to the label space using a classifier  $g$ . The overall mapping from the input to the label can be written as  $g \circ f(x) : \mathcal{X} \rightarrow \mathcal{Y}$ . Learning consists of estimating functions  $f$  and  $g$  that minimize an empirical loss  $\ell$  over the training data along with suitable regularization  $\mathcal{R}$  over the functions  $f$  and  $g$ . This can be written as:

$$\mathcal{L}_s := \sum_{(x_i, y_i) \in \mathcal{D}_s} \ell(g \circ f(x_i), y_i) + \mathcal{R}(f, g).$$

A commonly used loss is the cross-entropy loss and a regularizer is the  $\ell_2$  norm of the parameters of the functions. In a transfer learning setting  $g$  is discarded and relearned on training data for novel classes.

We also consider self-supervised losses  $\mathcal{L}_{ss}$  based on labeled data  $x \rightarrow (\hat{x}, \hat{y})$  that can be derived automatically without any human labeling. Fig. 3.1 shows two examples: the *jigsaw task* rearranges the input image and uses the index of the permutation as the target label, while the *rotation task* uses the angle of the rotated image as the target label. A separate function  $h$  is used to predict these labels from the shared feature backbone  $f$  with a self-supervised loss:

$$\mathcal{L}_{ss} := \sum_{x_i \in \mathcal{D}_{ss}} \ell(h \circ f(\hat{x}_i), \hat{y}_i).$$

Our final loss combines the two:  $\mathcal{L} := \mathcal{L}_s + \mathcal{L}_{ss}$  and thus the self-supervised losses act as a data-dependent regularizer for representation learning. The details of these losses are described in the next sections.

Note that the domain of images used for supervised  $\mathcal{D}_s$  and self-supervised  $\mathcal{D}_{ss}$  losses need not to be identical. In particular, we would like to use larger sets of images for self-supervised learning from related domains. The key questions we ask are: (1) How effective is SSL when  $\mathcal{D}_s = \mathcal{D}_{ss}$  especially when we have a small sample of  $\mathcal{D}_s$ ? (2) How do the domain shifts between  $\mathcal{D}_s$  and  $\mathcal{D}_{ss}$  affect generalization performance? and (3) How to select images from a large, generic pool to construct an effective  $\mathcal{D}_{ss}$  given a target domain  $\mathcal{D}_s$ ?

### 3.1.1 Supervised Losses ( $\mathcal{L}_s$ )

Most of our results are presented using a meta-learner based on prototypical networks [146] that perform episodic training and testing over sampled datasets in stages called meta-training and meta-testing. During meta-training, we randomly sample  $N$  classes from the

base set  $\mathcal{D}_b$ , then we select a support set  $\mathcal{S}_b$  with  $K$  images per class and another query set  $\mathcal{Q}_b$  with  $M$  images per class. We call this an  $N$ -way  $K$ -shot classification task. The embeddings are trained to predict the labels of the query set  $\mathcal{Q}_b$  conditioned on the support set  $\mathcal{S}_b$  using a nearest mean (prototype) classifier. The objective is to minimize the prediction loss on the query set. Once training is complete, given the novel dataset  $\mathcal{D}_n$ , class prototypes are recomputed for classification and query examples are classified based on the distances to the class prototypes.

Prototypical networks are related to distance-based learners such as matching networks [171] or metric-learning based on label similarity [86]. We also present few-shot classification results using a gradient-based meta-learner called MAML [46], and one trained with a standard cross-entropy loss on all the base classes.

### 3.1.2 Self-supervised Losses ( $\mathcal{L}_{ss}$ )

We consider two losses motivated by a recent large-scale comparison of the effectiveness of self-supervised learning tasks [57] described below:

- *Jigsaw puzzle task loss.* Here the input image  $x$  is tiled into  $3 \times 3$  regions and permuted randomly to obtain an input  $\hat{x}$ . The target label  $\hat{y}$  is the index of the permutation. The index (one of 9!) is reduced to one of 35 following the procedure outlined in [113], which grouped the possible permutations based on the hamming distance to control the difficulty of the task.
- *Rotation task loss.* We follow the method of [54] where the input image  $x$  is rotated by an angle  $\theta \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$  to obtain  $\hat{x}$  and the target label  $\hat{y}$  is the index of the angle.

In both cases we use the cross-entropy loss between the target and prediction.

### 3.1.3 Stochastic Sampling and Training

When the images used for SSL and meta-learning are identical, *i.e.*,  $\mathcal{D}_s = \mathcal{D}_{ss}$ , the same batch of images are used for computing both losses  $\mathcal{L}_s$  and  $\mathcal{L}_{ss}$ . For experiments investigating the effect of domain shifts described in Chapter 3.2.3 and 3.2.4, where SSL and meta-learner are trained on different domains, *i.e.*  $\mathcal{D}_s \neq \mathcal{D}_{ss}$ , a separate batch of size of 64 is used for computing  $\mathcal{L}_{ss}$ . After the two forward passes, one for the supervised task and one for the self-supervised task, the two losses are combined and gradient updates are performed. While other techniques exist [29, 82, 141], simply averaging the two losses performed well.

## 3.2 Experiments

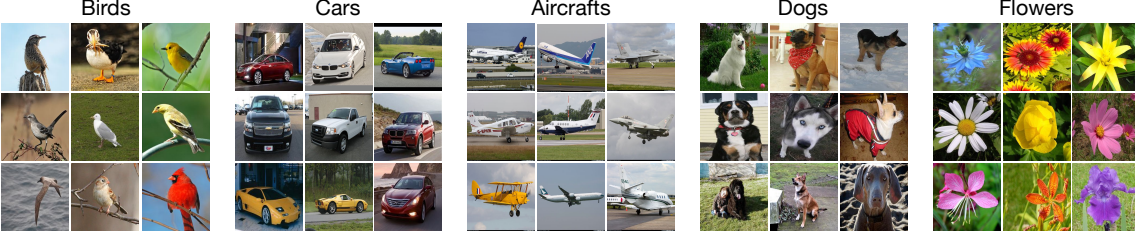
We first describe the datasets and experimental details. In Chapter 3.2.2, we present the results of using SSL to improve few-shot learning on various datasets. In Chapter 3.2.3, we show the effect of domain shift between labeled and unlabeled data for SSL. Last, we propose a way to select images from a pool for SSL to further improve the performance of few-shot learning in Chapter 3.2.4.

### 3.2.1 Experimental Setup

**Datasets and benchmarks.** We experiment with datasets across diverse domains: Caltech-UCSD birds [179], Stanford cars [91], FGVC aircrafts [104], Stanford dogs [83], and Oxford flowers [112]. Each dataset contains between 100 and 200 classes with a few thousands of images. We also experiment with the widely-used *mini*-ImageNet [171] and *tiered*-ImageNet [131] benchmarks for few-shot learning. In *mini*-ImageNet, each class has 600 images, wherein *tiered*-ImageNet each class has 732 to 1300 images.

We split classes within a dataset into three disjoint sets: *base*, *val*, and *novel*. For each class, all the images in the dataset are used in the corresponding set. A model is trained on the base set of categories, validated on the val set, and tested on the novel set of





Setting	Set	Stats	<i>mini-</i> ImageNet	<i>tiered-</i> ImageNet	Birds	Cars	Aircrafts	Dogs	Flowers
Few-shot transfer	Base	classes	64	351	100	98	50	60	51
		images	38,400	448,695	5885	8162	5000	10337	4129
	Val	classes	16	97	50	49	25	30	26
		images	9,600	124,261	2950	3993	2500	5128	2113
	Novel	classes	20	160	50	49	25	30	25
		images	12,000	206,209	2953	4030	2500	5115	1947

Table 3.1: **Example images and dataset statistics.** For few-shot learning experiments the classes are split into *base*, *val*, and *novel* set. Image representations learned on *base* set are evaluated on the *novel* set while *val* set is used for cross-validation. These datasets vary in the number of classes but are orders of magnitude smaller than ImageNet dataset.

categories given a few examples per class. For birds, we use the same split as [27], where  $\{base, val, novel\}$  sets have  $\{100, 50, 50\}$  classes respectively. The same ratio is used for the other four fine-grained datasets. We follow the original splits for *mini*-ImageNet and *tiered*-ImageNet. The statistics of various datasets used in our experiments are shown in Table 3.1. Notably, fine-grained datasets are significantly smaller.

We also present results on a setting where the base set is “degraded” either by (1) reducing the resolution, (2) removing color, or (3) reducing the number of training examples. This allows us to study the effectiveness of SSL on even smaller datasets and as a function of the difficulty of the task.

**Meta-learners and feature backbone.** We follow the best practices and use the code-base for few-shot learning described in [27]. In particular, we use ProtoNet [146] with a ResNet-18 [69] network as the feature backbone. Their experiments found this to be the

best performing. We also present experiments with other meta-learners such as MAML [46] and softmax classifiers in Chapter 3.2.2.

**Learning and optimization.** We use 5-way (classes) and 5-shot (examples per-class) with 16 query images for training. For experiments using 20% of labeled data, we use 5 query images for training since the minimum number of images per class is 10. The models are trained with ADAM [84] with a learning rate of 0.001 for 60,000 episodes. We report the mean accuracy and 95% confidence interval over 600 test experiments. In each test episode,  $N$  classes are selected from the novel set, and for each class 5 support images and 16 query images are selected. We report results for  $N = \{5, 20\}$  classes.

**Image sampling and data augmentation.** Data augmentation has a significant impact on few-shot learning performance. We follow the data augmentation procedure outlined in [27] which resulted in a strong baseline performance. For label and rotation predictions, images are first resized to 224 pixels for the shorter edge while maintaining the aspect ratio, from which a central crop of  $224 \times 224$  is obtained. For jigsaw puzzles, we first randomly crop  $255 \times 255$  region from the original image with random scaling between  $[0.5, 1.0]$ , then split into  $3 \times 3$  regions, from which a random crop of size  $64 \times 64$  is picked. While it might appear that with self-supervision the model effectively sees more images, SSL provides consistent improvements even after extensive data augmentation including cropping, flipping, and color jittering.

## 3.2.2 Results on Few-shot Learning

### 3.2.2.1 Self-supervised learning improves few-shot learning

Fig. 3.2 shows the accuracies of various models on few-shot learning benchmarks. Our ProtoNet baseline matches the results of the *mini*-ImageNet and birds datasets presented in [27] (in their Table A5). Our results show that jigsaw puzzle task improves the ProtoNet baseline on all seven datasets. Specifically, it reduces the *relative error rate* by 4.0%, 8.7%, 19.7%, 8.4%, 4.7%, 15.9%, and 27.8% on *mini*-ImageNet, *tiered*-ImageNet, birds,

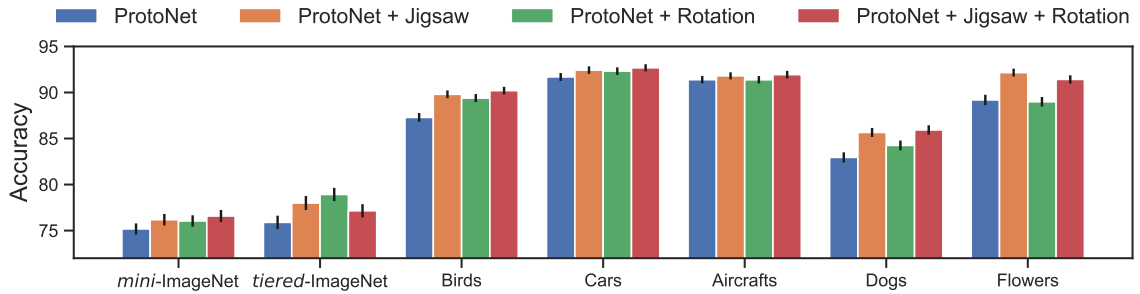


Figure 3.2: **Benefits of SSL for few-shot learning tasks.** We show the accuracy of the ProtoNet baseline of using different SSL tasks. The jigsaw task results in an improvement of the 5-way 5-shot classification accuracy across datasets. Combining SSL tasks can be beneficial for some datasets. Here SSL was performed on images within the base classes only.

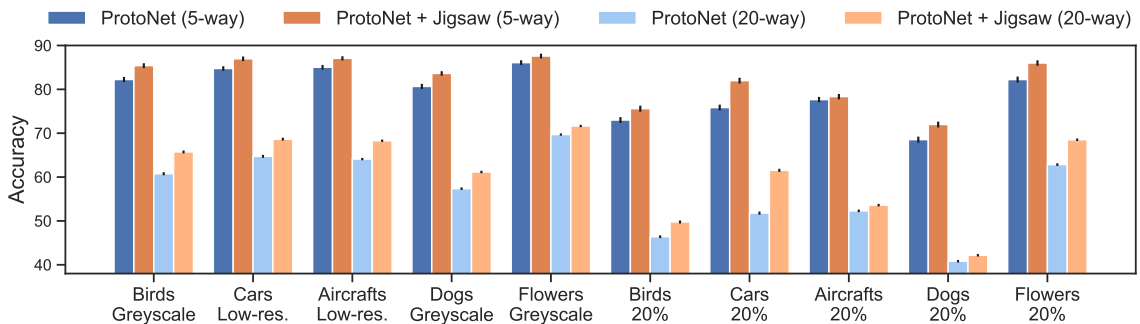


Figure 3.3: **Benefits of SSL for *harder* few-shot learning tasks.** We show the accuracy of using the jigsaw puzzle task over ProtoNet baseline on harder versions of the datasets. We see that SSL is effective even on smaller datasets and the relative benefits are higher.

cars, aircrafts, dogs, and flowers datasets respectively. Predicting rotations also improves the ProtoNet baseline on most of the datasets, except for aircrafts and flowers. We speculate this is because most flower images are symmetrical, and airplanes are usually horizontal, making the rotation task too hard or too trivial respectively to benefit the main task. In addition, combining these two SSL tasks can be beneficial sometimes.

### 3.2.2.2 Gains are larger for harder tasks

Fig. 3.3 shows the performance on the degraded version of the same datasets (first five groups). For cars and aircrafts we use low-resolution images where the images are down-

Loss	Birds	Cars	Aircrafts	Dogs	Flowers
	5-way 5-shot				
Softmax	81.5±0.5	87.7±0.5	89.2±0.4	77.6±0.6	91.0±0.5
Softmax + Jigsaw	83.9±0.5	90.6±0.5	89.6±0.4	77.8±0.6	91.1±0.5
MAML	81.2±0.7	86.9±0.6	88.8±0.5	77.3±0.7	79.0±0.9
MAML + Jigsaw	81.1±0.7	89.0±0.5	89.1±0.5	77.3±0.7	82.6±0.7
ProtoNet	87.3±0.5	91.7±0.4	91.4±0.4	83.0±0.6	89.2±0.6
ProtoNet + Jigsaw	<b>89.8±0.4</b>	<b>92.4±0.4</b>	<b>91.8±0.4</b>	<b>85.7±0.5</b>	<b>92.2±0.4</b>

Table 3.2: **Performance on few-shot learning using different meta-learners.** Using jigsaw puzzle loss improves different meta-learners on most of the datasets. ProtoNet with jigsaw loss performs the best on all five datasets.

sampled by a factor of *four* and up-sampled back to  $224 \times 224$  with bilinear interpolation. For natural categories we discard color. Low-resolution images are considerably harder to classify for man-made categories while color information is most useful for natural categories [153]. On birds and dogs datasets, the improvements using self-supervision (3.2% and 2.9% on 5-way 5-shot) are higher compared to color images (2.5% and 2.7%), similarly on the cars and aircrafts datasets with low-resolution images (2.2% and 2.1% vs. 0.7% and 0.4%). We also conduct an experiment where only 20% of the images in the base categories are used for both SSL and meta-learning (last five groups in Fig. 3.3). This results in a much smaller training set than standard few-shot benchmarks: 20% of the birds dataset amounts to only roughly 3% of the popular *mini*-ImageNet dataset. We find larger benefits from SSL in this setting. For example, the gain from the jigsaw puzzle loss for 5-way 5-shot car classification increases from 0.7% (original dataset) to 7.0% (20% training data).

### 3.2.2.3 Improvements generalize to other meta-learners

We combine SSL with other meta-learners and find the combination to be effective. In particular, we use MAML [46] and a standard feature extractor trained with cross-entropy loss (softmax) as in [27]. Table 3.2 compares meta-learners based on a ResNet-18 network

Model	Image Size	Backbone	SSL	Accuracy (%)
MAML [46]	84×84	Conv4-64	-	63.1
ProtoNet [146]		Conv4-64	-	68.2
RelationNet [158]		Conv4-64	-	65.3
LwoF [53]		Conv4-64	-	72.8
PFA [125]*		WRN-28-10	-	73.7
TADAM [117]		ResNet-12	-	76.7
LEO [135]*		WRN-28-10	-	77.6
MetaOptNet-SVM [99] <sup>†</sup>		ResNet-12	-	78.6
Chen <i>et al.</i> [27]	84×84	Conv4-64	-	64.2
(ProtoNet)	224×224	ResNet-18	-	73.7
Gidaris <i>et al.</i> [52] (ProtoNet)	84×84	Conv4-64	- Rotation	70.0 71.7
		Conv4-512	- Rotation	71.6 74.0
		WRN-28-10	- Rotation	68.7 72.1
<b>Ours</b> (ProtoNet)	224×224	ResNet-18	-	75.2
			Rotation	76.0
			Jigsaw	76.2
			Rot.+Jig.	76.6

Table 3.3: **Comparison with prior works on *mini-ImageNet*.** 5-shot 5-way classification accuracies on 600 test episodes are reported. The implementation details including image size, backbone model, and training are different in each paper. \*validation classes are used for training. <sup>†</sup>dropblock [51], label smoothing, and weight decay are used.

trained with and without *jigsaw puzzle loss*. We observe that the average 5-way 5-shot accuracies across five fine-grained datasets for softmax, MAML, and ProtoNet improve from 85.5%, 82.6%, and 88.5% to 86.6%, 83.8%, and 90.4% respectively when combined with the jigsaw puzzle task. Self-supervision improves performance across different meta-learners and different datasets; however, ProtoNet trained with self-supervision is the best model across all datasets.

#### 3.2.2.4 Self-supervision alone is not enough

SSL alone significantly lags behind supervised learning in our experiments. For example, a ResNet-18 trained with SSL alone achieve 32.9% (w/ jigsaw) and 33.7% (w/ rotation) 5-way 5-shot accuracy averaged across five fine-grained datasets. While this is better than a random initialization (29.5%), it is dramatically worse than one trained with a simple cross-entropy loss (85.5%) on the labels. Surprisingly, we also found that initialization with SSL followed by meta-learning did *not* yield improvements over meta-learning starting from random initialization, supporting the view that SSL acts as a feature regularizer.

#### 3.2.2.5 Few-shot learning as an evaluation for self-supervised tasks

The few-shot classification task provides a way of evaluating the effectiveness of self-supervised tasks. For example, on 5-way 5-shot aircrafts classification, training with only jigsaw and rotation task gives 38.8% and 29.5% respectively, suggesting that rotation is not an effective self-supervised task for airplanes. We speculate that it might be because the task is too easy as airplanes are usually horizontal.

#### 3.2.2.6 Comparison with prior works

Our results also echo those of [52] who find that the rotation task improves on *mini-* and *tiered-*ImageNet. In addition we show the improvement still holds when using deeper networks, higher resolution images, and in fine-grained domains. We provide a comparison with other few-shot learning methods in Table 3.3.

### 3.2.3 Analyzing the Effect of Domain Shift for Self-supervision

Scaling SSL to massive unlabeled datasets that are readily available for some domains is a promising avenue for improvement. *However, do more unlabeled data always help for a task in hand?* This question hasn't been sufficiently addressed in the literature as most prior works study the effectiveness of SSL on a curated set of images, such as ImageNet, and their transferability to a handful of tasks. We conduct a series of experiments to characterize

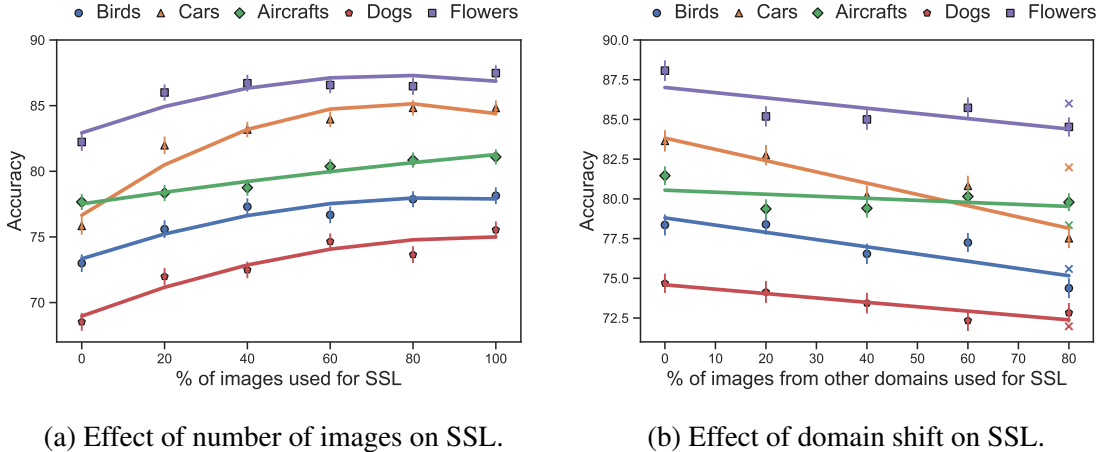


Figure 3.4: **Effect of size and domain of SSL on 5-way 5-shot classification accuracy.** (a) More unlabeled data from the same domain for SSL improves the performance of the meta-learner. (b) Replacing a fraction (x-axis) of the images with those from other domains makes SSL less effective.

the effect of size and distribution  $\mathcal{D}_{ss}$  of images used for SSL in the context of few-shot learning on domain  $\mathcal{D}_s$ .

First, we investigate if SSL on unlabeled data from the same domain improves the meta-learner. We use 20% of the images in the base categories for meta-learning identical to the setting in Fig. 3.3. The labels of the remaining 80% data are withheld and only the images are used for SSL. We systematically vary the number of images used by SSL from 20% to 100%. The results are presented in Fig. 3.4a. The accuracy improves with the size of the unlabeled set with diminishing returns. Note that 0% corresponds to no SSL and 20% corresponds to using only the labeled images for SSL ( $\mathcal{D}_s = \mathcal{D}_{ss}$ ).

Fig. 3.4b shows an experiment where a fraction of the unlabeled images are replaced with images from other four datasets. For example, 20% along the x-axis for birds indicate that 20% of the images in the base set are replaced by images drawn uniformly at random from other datasets. Since the numbers of images used for SSL is identical, the x-axis from left to right represents increasing amounts of domain shifts between  $\mathcal{D}_s$  and  $\mathcal{D}_{ss}$ . We observe that the effectiveness of SSL decreases as the fraction of out-of-domain images

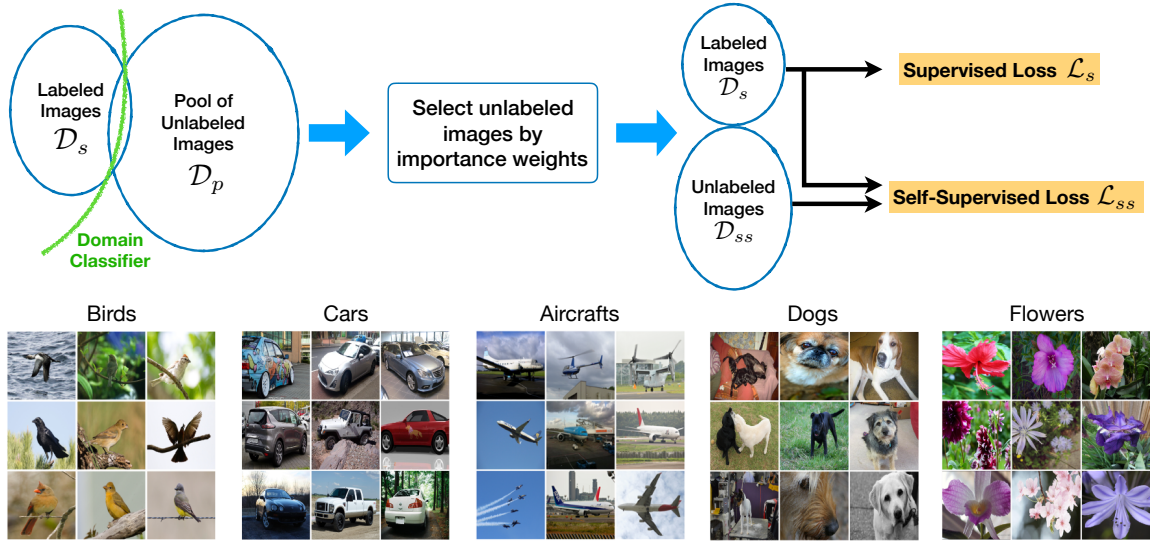


Figure 3.5: **Overview of domain selection for self-supervision.** **Top:** We first train a domain classifier using  $\mathcal{D}_s$  and (a subset of)  $\mathcal{D}_p$ , then select images using the predictions from the domain classifier for self-supervision. **Bottom:** Selected images of each dataset using importance weights.

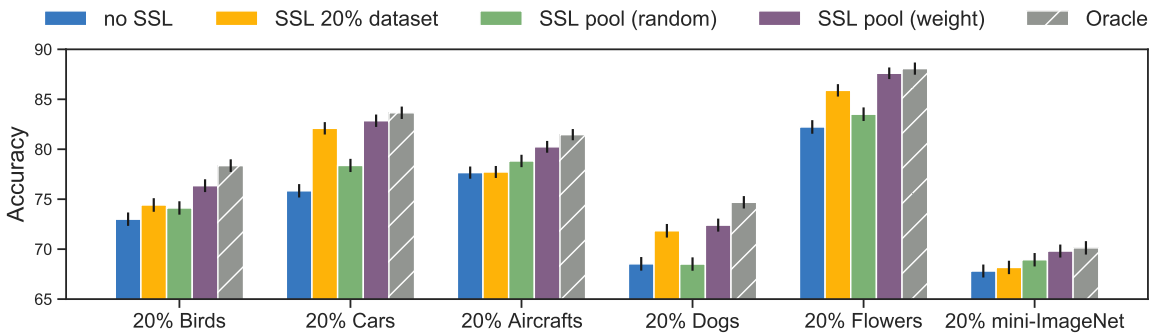


Figure 3.6: **Effectiveness of selected images for SSL.** With random selection, the extra unlabeled data often hurts the performance, while those sampled using the *importance weights* improve performance on all five datasets.

increases. Importantly, training with SSL on the available 20% within domain images (shown as crosses) is often (on 3 out of 5 datasets) better than increasing the set of images by five times to include out of domain images.



### 3.2.4 Selecting Images for Self-supervision

Based on the above analysis we propose a simple method to select images for SSL from a large, generic pool of unlabeled images in a dataset dependent manner. We use a “domain weighted” model to select the top images based on a domain classifier, in our case a binary logistic regression model trained with images from the source domain  $\mathcal{D}_s$  as the positive class and images from the pool  $\mathcal{D}_p$  as the negative class based on ResNet-101 image features. The top images are selected according to the ratio  $p(x \in \mathcal{D}_s)/p(x \in \mathcal{D}_p)$ . Note that these *importance weights* account for the domain shift. Fig. 3.5 shows an overview of the selection process.

We evaluate this approach using a pool of images  $\mathcal{D}_p$  consisting of (1) the training images of the “bounding box” subset of Open Images V5 [94] which has 1,743,042 images from 600 classes, and (2) iNaturalist 2018 dataset [166] which has 461,939 images from 8162 species. For each dataset, we use 20% of the labeled images as  $\mathcal{D}_s$ . The rest 80% of the data are only used as the “oracle” where the unlabeled data are drawn from the exact same distribution as  $\mathcal{D}_s$ . We show some of the selected images for self-supervision  $\mathcal{D}_{ss}$  in Fig. 3.5.

Fig. 3.6 shows the results of ProtoNet trained on 20% labeled examples with jigsaw puzzle as self-supervision. To have a fair comparison, for methods of selecting images from the pool, we select the same number (80% of the original labeled dataset size) of images as  $\mathcal{D}_{ss}$ . We report the mean accuracy of five runs. “SSL with 20% dataset” denotes a baseline of only using  $\mathcal{D}_s$  for self-supervision ( $\mathcal{D}_s = \mathcal{D}_{ss}$ ), which is our reference “lower bound”. SSL pool “(random)” and “(weight)” denote two approaches of selecting images for self-supervision. The former selects images uniformly at random, which is detrimental for cars, dogs, and flowers. The pool selected according to the *importance weights* provides significant improvements over “no SSL”, “SSL with 20% dataset”, and “random selection” baselines on all five datasets. The oracle is trained with the remaining 80% of the original dataset as  $\mathcal{D}_{ss}$ , which is a reference “upper bound”.

## CHAPTER 4

### A REALISTIC EVALUATION ON SEMI-SUPERVISED LEARNING

In Chapter 3, we have shown that self-supervised learning can improve the performance on few-shot classes, and an additional set of unlabeled data can further boost the performance if the data are from a similar distribution. However, when we have a large set of unlabeled data, it is natural to ask if semi-supervised learning (Semi-SL) methods can also be applied in addition to few-shot and unsupervised methods. Indeed, Zhai *et al.* [188] has combined semi-supervised and self-supervised methods to boost the performance, and Chen *et al.* [26] showed that self-training can achieve state-of-the-art results on semi-supervised benchmarks thanks to the recent advances in contrastive learning. Yet, the current literature on Semi-SL with deep networks for image classification has two main shortcomings. First, most methods are evaluated on curated datasets such as CIFAR, SVHN, or ImageNet, where class distribution is or is close to uniform and unlabeled data contains no novel classes. This is implicit in methods that rely on the assumption that the data is uniformly clustered, use a uniform instead of class-balanced loss, or categorize unlabeled data into one of the labeled classes. In practice, however, class distribution can be highly unbalanced or even unknown, and the unlabeled data may contain novel classes. How effective is Semi-SL in these situations?

Second, most literature has focused on training models from scratch. However, a practical approach for few-shot learning is to use expert models trained on large labeled datasets such as ImageNet [134] or iNaturalist [167]. What gains does Semi-SL provide in this setting, especially since many Semi-SL methods are based on learning invariances from data

based on transformations which might have already been learned by the experts during supervised training? Moreover, is out-of-domain data beneficial when experts are available?

In this chapter, we aim to answer these questions by conducting a systematic study of Semi-SL techniques (Fig. 4.1) on two fine-grained classification datasets that exhibit a long-tailed distribution of classes and contain a large number of out-of-class images (Fig. 4.2). These datasets are obtained by sampling classes under the Aves (birds) and Fungi taxonomy. The out-of-class images are other Aves (or Fungi) images not belonging to the classes within the labeled set. The first dataset was part of the semi-supervised challenge at FGVC7 workshop [154], while the second one is constructed from the FGVC fungi challenge [48] following a similar scheme, details of which are described in § 4.1.

On these datasets, we conduct a systematic study of existing deep-learning-based semi-supervised learning approaches for image classification. We perform experiments on Semi-SL methods including Pseudo-Label [97], Curriculum Pseudo-Label [22], FixMatch [148], self-training using distillation [183], self-supervised learning (MoCo [66]), as well as their combinations when effective. We investigate strategies for using unlabeled data when models are initialized from experts. We also evaluate the performance of methods that use unlabeled data from the same classes as the labeled dataset ( $U_{in}$ ) and a practical setting where the unlabeled data includes out-of-class images ( $U_{in} + U_{out}$ ). The high-level summary of the experiments reported in Fig. 4.1, Tab. 4.3, 4.4, and Fig. 4.4 are as follows:

- Some of the Semi-SL methods are effective when models are trained from scratch, especially those with self-supervised pre-training can significantly benefit from out-of-class data (long blue whiskers and longer orange whiskers above the *baseline* for *scratch* in Fig. 4.1). In this setting, self-supervised learning followed by distillation-based self-training performs the best (Tab. 4.3 and 4.4).
- The best Semi-SL approach significantly under-performs the supervised fine-tuning model trained on the labeled portion of the datasets (the *baseline* performance of *ImageNet* and *iNat* is higher than any Semi-SL model trained from scratch in Fig. 4.1).

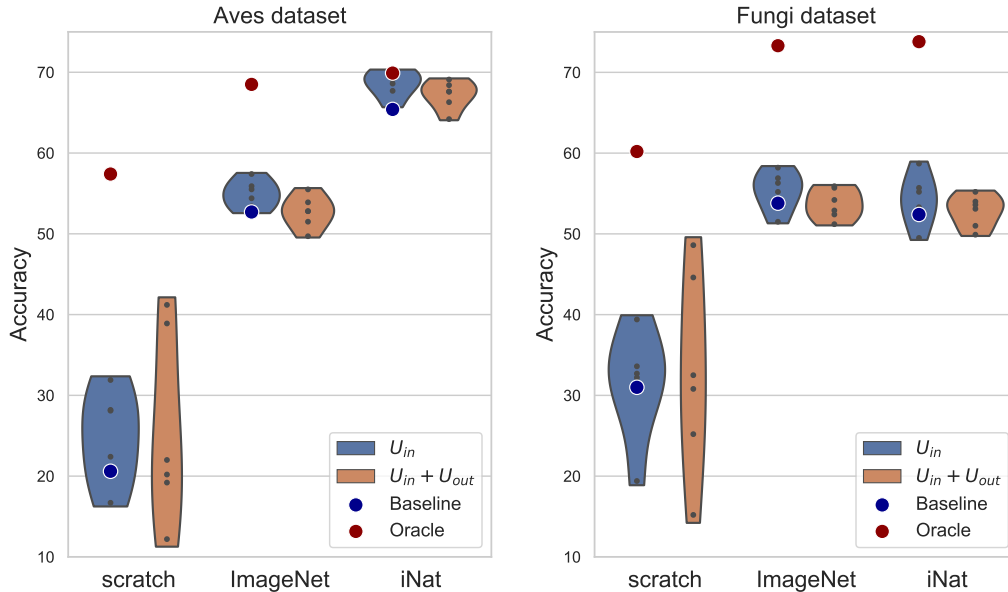


Figure 4.1: Accuracy of semi-supervised learning (Semi-SL) algorithms on the Semi-Aves and Semi-Fungi datasets (see Fig. 4.2) using (i) different pre-trained models, and (ii) in-class ( $U_{in}$ ) and out-of-class ( $U_{in} + U_{out}$ ) unlabeled data. The performances of the supervised baseline and supervised oracle are also shown. Transfer learning from experts is far more effective than Semi-SL from *scratch*, while in the transfer setting Semi-SL provides modest gains. Though out-of-class data ( $U_{out}$ ) is valuable when training from *scratch*, it is not the case when training from experts (details in Tab. 4.3 and 4.4).

- Picking the right expert provides further gains in this few-shot setting but not when training using the entire labeled dataset (*oracle* performance in Fig. 4.1).
- When training with experts, FixMatch gives the most improvements when having  $U_{in}$  only. However, the presence of out-of-class unlabeled data often hurts performance. Self-Training was the most robust to the presence of out-of-class data (Tab. 4.3, 4.4 and Fig. 4.4).
- Surprisingly, we found that no method was able to reliably use out-of-class data even though the domain shift is relatively small (the orange group is not higher than the blue groups for *ImageNet* and *iNat* unlike *scratch* in Fig. 4.1), echoing the experience of participants in the FGVC7 challenge [154].

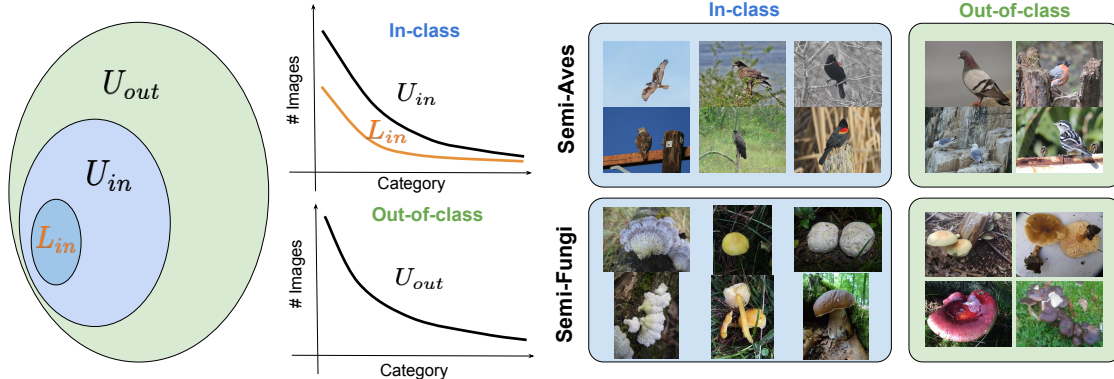


Figure 4.2: **The proposed benchmark for semi-supervised learning.** The benchmark contains two datasets, with classes from the Aves and Fungi taxa respectively. Each represents a 200-way classification task and the training set contains (i) labeled images from these classes  $L_{in}$ , (ii) unlabeled images from these classes  $U_{in}$ , and (iii) unlabeled images from related classes  $U_{out}$ , as seen on figures to the right. Moreover, the classes exhibit a long-tailed distribution with an imbalance ratio of 8 to 10. The benchmark captures conditions observed in some realistic applications that are not present in existing datasets used to evaluate semi-supervised learning. See § 4.1 and Tab. 4.1 for details.

- The performance of Semi-SL is far below the model trained using labels of the entire in-class data suggesting that there is significant room for improvement (*oracle* performance in Fig. 4.1).

In summary, we conduct a systematic evaluation of several recently proposed Semi-SL techniques on two challenging datasets representing a long-tailed distribution of fine-grained categories. We vary the initialization and the domain of the unlabeled data and analyze the robustness of various Semi-SL approaches. Our experiments indicate that Semi-SL does not work out-of-the-box in a transfer learning setting, especially in the presence of out-of-domain data. These results are in a similar vein to prior work on the evaluation of Semi-SL approaches that have analyzed the robustness of Semi-SL techniques to the choice of hyper-parameters [115], network architectures [26, 182], and domain shifts [115, 156, 172], *etc.* However, the evaluation in a transfer learning setting on the proposed benchmarks reveals additional insights. We hope these experiments inspire prac-

Dataset	Classes	Images	Unlabeled	Image	Class	Imba. Ratio
	$L_{in} / U_{in} / U_{out}$	$L_{in} / U_{in} / U_{out}$	Class Domain	Res.	Distri.	
CIFAR-10	10 / 10 / 0	4K / 40K / 0	$L = U$	$32 \times 32$	uniform	1
CIFAR-100	100 / 100 / 0	10K / 50K / 0	$L = U$	$32 \times 32$	uniform	1
SVHN	10 / 10 / 0	1K / 65K / 0	$L = U$	$32 \times 32$	uniform	1
STL-10	10 / 0 / -	5K / 0 / 100K	$L \neq U$	$96 \times 96$	uniform	1
ImageNet	1000 / 1000 / 0	140K / 1.26M / 0	$L = U$	$224 \times 224$	$\approx$ uniform	1.8
Semi-Aves	200 / 200 / 800	6K / 27K / 122K	$L = U_{in} \neq U_{out}$	$224 \times 224$	long-tailed	7.9
Semi-Fungi	200 / 200 / 1194	4K / 13K / 65K	$L = U_{in} \neq U_{out}$	$224 \times 224$	long-tailed	10.1
Semi-iNat	810 / (2439)	9K / (313K)	$L \neq (U_{in} \cup U_{out})$	$224 \times 224$	long-tailed	12.9

Table 4.1: A comparison of Semi-Aves, Semi-Fungi, and Semi-iNat datasets with existing Semi-SL benchmarks. Semi-Aves, Semi-Fungi, and Semi-iNat present a challenge due to the large number of classes, presence of novel images in the unlabeled set, long-tailed distribution of classes as indicated by the class imbalance ratio (maximum / minimum images per class) in the training set.

tical methods that combine the benefits of supervised learning and task-specific learning on partially labeled datasets.

## 4.1 A Realistic Benchmark

In Semi-SL, we are provided with labeled training data  $(x_i, y_i) \in L_{in}$  and unlabeled training data  $(u_i, \cdot) \in U$ . The unlabeled data can either belong to the same classes as the labeled data ( $U_{in}$ ), or to novel classes ( $U_{out}$ ). In a realistic setting, one may expect that the unlabeled data contains novel classes. In many applications it is easy to acquire images from related domains through coarse labeling, *e.g.*, it is easier to label an image as a bird than a “yellow bunting”. Such images could be potentially used to learn better representations. Thus we evaluate Semi-SL methods in two settings, one when the unlabeled data contains no novel images, and another when it does, *i.e.*,  $U_{in}$  and  $U_{in} + U_{out}$  respectively.

We use two datasets by sampling classes from the natural domains for our benchmark. As shown in Fig. 4.2, the classes belong to the Aves and Fungi taxonomy and contain a long-tailed distribution of classes, as commonly observed in fine-grained domains. Tab. 4.1

shows a comparison with other benchmarks. Larger image sizes, significant class imbalance, fine-grained categories, and a large number of out-of-class images allow a more realistic evaluation of Semi-SL techniques. Below we describe each dataset.

#### 4.1.1 Semi-Aves

We use the dataset from the semi-supervised challenge at the FGVC7 workshop at CVPR 2020 [154]. The dataset includes a subset of bird species from the Aves kingdom of iNaturalist 2018 dataset [167]. However, there are no overlapping images since the images were collected from recent years. There are 200 in-class and 800 out-of-class categories. The training and validation set has a total of 5959 labeled images, 26,640 in-class and 122,208 out-of-class unlabeled images, and 8000 test images. The training data in  $L_{in}$ ,  $U_{in}$ , and  $U_{out}$  is long-tail distributed, specifically  $L_{in}$  has 15 to 53 images and  $U_{in}$  has 16 to 229 images per class. The test data has a uniform distribution with 40 images per class.

#### 4.1.2 Semi-Fungi

We create a Semi-Fungi dataset following the similar strategy of the Semi-Aves dataset. We use the train-val set of images from the FGVCx Fungi challenge at the FGVC5 workshop at CVPR 2018 [48]. The dataset was collected from the “Svampe Atlas”<sup>1</sup> website, thus the image domain is different from iNaturalist. The original dataset has 1394 fungi species with a long-tailed distribution. We first sort the classes by frequency and randomly select 200 of the top 600 classes as in-class categories. We then select 20 images per class as the test set, and randomly select 4141 images as labeled data and the rest 13,166 images as in-class unlabeled data. The rest 1194 species are used as out-of-class unlabeled images, which has a total of 64,871 images. In Semi-Fungi, there are 6 to 78 images per class in  $L_{in}$ , and 16 to 276 images in  $U_{in}$ . The test set is uniformly distributed with 20 images per class.

---

<sup>1</sup><https://svampe.databasesen.org>



Figure 4.3: **Examples from the Semi-iNat dataset.** The Semi-iNat dataset includes images from 8 different phyla.

Kingdom	Phylum	$C_{in}$	$C_{out}$
Animalia (1294)	Mollusca	11	24
	Chordata	113	228
	Arthropoda	301	605
	Echinodermata	4	8
Plantae (1028)	Tracheophyta	336	674
	Bryophyta	6	12
Fungi (117)	Basidiomycota	29	58
	Ascomycota	10	20

Table 4.2: **The number of species in the taxonomy.** For each phylum, we select around one-third of the species for the in-class set  $C_{in}$  and the rest for the out-of-class set  $C_{out}$ .

### 4.1.3 Semi-iNat

Last, we use the dataset from the semi-supervised challenge at the FGVC8 workshop at CVPR 2021 [155]. Semi-iNat builds on the Semi-Aves while incorporating some new challenges. First, the dataset contains species from three kingdoms: Animal, Plants, and Fungi (Fig. 4.3 and Tab. 4.2), unlike Semi-Aves which contains only Aves (birds). There are



a total of 2439 animal species and  $\approx 330\text{k}$  images, which is more than two times larger than Semi-Aves. In addition, the domain labels are not provided for the challenge. We acquire them only for our analysis. Last, coarse taxonomy labels for unlabeled data are provided, which are easily obtained from non-experts and provide a weak supervisory signal that could be exploited by the learning methods.

## 4.2 Methods

In this section, we describe the details of the Semi-SL methods we compared in our benchmark.

**(1) Supervised baseline / oracle:** We train the model only using labeled data  $L_{in}$  with a cross-entropy loss. For the oracle, we include the ground-truth labels of  $U_{in}$  for training.

**(2) Pseudo-Labeling [97]:** The approach uses a base model’s confident predictions on unlabeled images as labels. Concretely, if the maximum probability of a class is greater than a threshold  $\tau$ , we then take the class as the target label. Following the implementation of Oliver *et al.* [115], we sample half of the batch from  $L_{in}$  and half from unlabeled data  $U$  during training. Denote  $(x_i, y_i)$  as a labeled sample, the predictions on unlabeled data  $u_i$  of the model  $f$  as  $q_i = f(u_i)$ , pseudo-label as  $\hat{q}_i = \operatorname{argmax}(q_i)$ , and cross-entropy function as  $H(p, q) = -\sum_r p(r) \log q(r)$ . Then, the objective for each batch is:

$$\mathcal{L} = \sum_{j=1}^n H(y_j, f(x_j)) + \sum_{i=1}^n \mathbf{1}[\max(q_i) \geq \tau] H(\hat{q}_i, q_i). \quad (4.1)$$

**(3) Curriculum Pseudo-Labeling [22]:** Unlike pseudo-labeling where labels are generated in an online manner, curriculum labeling generates pseudo-labels after the training is finished on the current labeled set before retraining. We first train a supervised model on labeled data  $L_{in}$ , then select images with the highest predictions from all the unlabeled data  $u \in U$ , and add them with their pseudo-labels to the labeled dataset. In the next iteration,

we retrain a model *from scratch* using the new set of labeled data. We repeat this process 5 times and select  $\{20, 40, 60, 80, 100\}\%$  of the unlabeled data from the original pool of unlabeled data  $U$ . The steps are as the following:

- (i) Initialize  $L = L_{in}, \beta = 20$ .
- (ii) Supervised training on  $L$ .
- (iii) Generate predictions  $q = f_{\theta}(x)$  for every  $u \in U$ .
- (iv) From  $U$  select  $\beta\%$  examples with highest prediction scores and their pseudo-labels as  $L_{top}$ .
- (v) Add selected unlabeled data with their pseudo-labels to the labeled dataset  $L = L_{in} \cup L_{top}$ .
- (vi) If  $\beta < 100, \beta = \beta + 20$  and repeat from step (ii).

**(4) FixMatch [148]:** FixMatch combines pseudo-labeling and consistency regularization. For each unlabeled image, it minimizes the cross-entropy between the pseudo-label (thresholded prediction) of the weakly-augmented image and the predictions of the strong-augmented image. For labeled data, only weak augmentations are applied. Specifically, let  $\alpha$  be a weak augmentation (image flipping in our case) and  $\mathcal{A}$  be a strong augmentation (RandAugment [33] in our case). Let the predictions under strong and weak augmentations are  $Q_i = f(\mathcal{A}(u_i)), q_i = f(\alpha(u_i))$ . The total loss for labeled and unlabeled data is

$$\mathcal{L} = \sum_{j=1}^m H(y_j, f(\alpha(x_j))) + \sum_{i=1}^{km} \mathbb{1}[\max(q_i) \geq \tau] H(\hat{q}_i, Q_i). \quad (4.2)$$

In the original implementation, each batch uses  $m$  labeled and  $km$  unlabeled data with a total batch size  $n = (k + 1)m$ , where the sampling ratio  $k$  is a hyper-parameter.

**(5) Self-Training:** While the term of ‘‘Self-Training’’ is general, we use this to refer to the following procedure using distillation [72]. We first train a supervised model  $f^t$  on the

labeled data which we call the teacher model, then train a student model  $f^s$  with scaled cross-entropy loss on the unlabeled data and cross-entropy loss on labeled data. Distillation was originally used for model compression [18], but has been shown to improve the performance when training the student model with the same architecture [49] or across different modalities [161, 61, 153]. Given unlabeled data  $(u, \cdot)$ , let the *logits* from teacher and student model as  $z^t$  and  $z^s$ , and the prediction of labeled data  $(x, y)$  from the student model is  $y^s$ . The objective includes the cross-entropy loss for labeled data  $(x, y)$ , and the distillation loss for unlabeled data:

$$\mathcal{L} = (1 - \lambda) \sum_{j=1}^n H(y_j, y_j^s) + \lambda \sum_{i=1}^n H\left(\sigma\left(\frac{z_i^t}{T}\right), \sigma\left(\frac{z_i^s}{T}\right)\right), \quad (4.3)$$

where  $\lambda$  is the weight between supervised and distillation losses,  $\sigma$  is the softmax function, and  $T$  is a temperature (scaling) parameter.

**(6) Self-Supervised Learning (MoCo [66]):** We use momentum contrastive (MoCo) learning as a strong baseline for self-supervised training. MoCo learns an image encoder  $f(x)$  that maps the image  $x$  to a representation  $q = f(x)$  and uses a contrastive objective that requires positive pairs to be closer than negative pairs in the representation space. The positive pairs are sampled from two geometric or photometric augmented views of a same images while negative images are augmentations from different images. MoCo adapts the InfoNCE [116] loss as the objective function. The loss for each encoded query  $q$  is:

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k^+ / T)}{\exp(q \cdot k^+ / T) + \sum_i^K \exp(q \cdot k_i^- / T)}, \quad (4.4)$$

where  $T$  is the temperature,  $k^+$  and  $k^-$  are the positive and negative sample of the query  $q$ . The number of negative samples  $K$  is limited by the mini-batch size. In order to stabilize the training, MoCo uses the memory bank [181] to store the negative samples and updates the encoder of the keys in the memory bank based on momentum. After the self-supervised

pre-training, we remove the MLP layers after the global average pooling layer, add a linear classifier (a fully convolutional layer followed by softmax), and train the entire network with supervised cross-entropy loss. We found that freezing the pre-trained backbone gives worse performance than fine-tuning the entire network.

**(7) MoCo + Self-Training:** Here we initialize the model using MoCo learning on the unlabeled data before semi-supervised learning using Self-Training. A recent work by Chen *et al.* [26] has shown this to be a strong semi-supervised learning baseline. The procedure is as follows:

- (i) Pre-train the model using MoCo on  $L_{in}$  and  $U$ .
- (ii) Fine-tune the model on  $L_{in}$  with a cross-entropy loss. Call this the teacher model  $f^t$ .
- (iii) Train a student model  $f^s$  initialized from step (i) with distillation loss (Eq. 4.3) using the teacher model  $f^t$ .

## 4.3 Experiments

### 4.3.1 Implementation details

**Network architecture and pre-training.** For a fair comparison, we use a ResNet-50 network [70] on  $224 \times 224$  images for all our experiments. For transfer learning, we use pre-trained models on ImageNet [134] and iNaturalist 2018 (iNat) [167] dataset, which contains 8142 species including 1248 Aves and 321 Fungi species. Note that there are *no overlapping images* between iNat’s training set and Semi-Aves, though there are overlapping categories. The images for Semi-Fungi images do not overlap with iNaturalist, but we do not know how many overlapping classes there are as species names were not provided in the original dataset [48] from which it was constructed. However, this is less of a concern as we find that iNat pre-trained model performs worse than an ImageNet pre-trained model on Semi-Fungi, suggesting the class overlap is likely small if any. To obtain an iNat pre-trained model, we train the model using SGD with momentum with a learning

Method	from scratch		from ImageNet		from iNat		
	Top1	Top5	Top1	Top5	Top1	Top5	
Supervised baseline	20.6±0.4	41.7±0.7	52.7±0.2	78.1±0.1	65.4±0.4	86.6±0.2	
Supervised oracle	57.4±0.3	79.2±0.1	68.5±1.4	88.5±0.4	69.9±0.5	89.8±0.7	
$U_{in}$	Pseudo-Label [97]	16.7±0.2	36.5±0.8	54.4±0.3	78.8±0.3	65.8±0.2	86.5±0.2
	Curriculum Pseudo-Label [22]	20.5±0.5	41.7±0.5	53.4±0.8	78.3±0.5	69.1±0.3	87.8±0.1
	FixMatch [148]	28.1±0.1	51.8±0.6	<b>57.4±0.8</b>	78.5±0.5	<b>70.2±0.6</b>	87.0±0.1
	Self-Training	22.4±0.4	44.1±0.1	55.5±0.1	79.8±0.1	67.7±0.2	87.5±0.2
	MoCo [66]	28.2±0.3	53.0±0.1	52.7±0.1	78.7±0.2	68.6±0.1	87.7±0.1
	MoCo + Self-Training	<b>31.9±0.1</b>	<b>56.8±0.1</b>	55.9±0.2	<b>80.3±0.1</b>	<b>70.1±0.2</b>	<b>88.1±0.1</b>
$U_{in} + U_{out}$	Pseudo-Label [97]	12.2±0.8	31.9±1.6	52.8±0.5	77.8±0.1	66.3±0.3	86.4±0.2
	Curriculum Pseudo-Label [22]	20.2±0.5	41.0±0.9	52.8±0.5	77.8±0.1	<b>69.1±0.1</b>	<b>87.6±0.1</b>
	FixMatch [148]	19.2±0.2	42.6±0.6	49.7±0.2	72.8±0.5	64.2±0.2	84.5±0.1
	Self-Training	22.0±0.5	43.3±0.2	<b>55.5±0.3</b>	<b>79.7±0.2</b>	67.6±0.2	<b>87.6±0.1</b>
	MoCo [66]	38.9±0.4	65.4±0.3	51.5±0.4	77.9±0.2	67.6±0.1	<b>87.3±0.2</b>
	MoCo + Self-Training	<b>41.2±0.2</b>	<b>65.9±0.3</b>	53.9±0.2	<b>79.4±0.3</b>	68.4±0.2	<b>87.6±0.2</b>

Table 4.3: **Results on Semi-Aves benchmark.** We experiment with six different Semi-SL methods as well as supervised baselines under different settings: (1) using  $U_{in}$  or  $U_{in} + U_{out}$  as the unlabeled dataset, (2) training from scratch, or using ImageNet or iNat pre-trained model. We show that when training from scratch with  $U_{in}$ , MoCo + Self-Training performs the best. When having expert models, transfer learning is a strong baseline, and FixMatch and Self-Training can still give improvements. When adding unlabeled data from  $U_{out}$ , the performance pales except for the self-supervised method when training from scratch. The best results and those within the variance are marked in **teal**.

Method	from scratch		from ImageNet		from iNat		
	Top1	Top5	Top1	Top5	Top1	Top5	
Supervised baseline	31.0±0.4	54.7±0.8	53.8±0.4	80.0±0.4	52.4±0.6	79.5±0.5	
Supervised oracle	60.2±0.8	83.3±0.9	73.3±0.1	92.5±0.3	73.8±0.3	92.4±0.3	
$U_{in}$	Pseudo-Label [97]	19.4±0.4	43.2±1.5	51.5±1.2	81.2±0.2	49.5±0.4	78.5±0.2
	Curriculum Pseudo-Label [22]	31.4±0.6	55.0±0.6	53.7±0.2	80.2±0.1	53.3±0.5	80.0±0.5
	FixMatch [148]	32.2±1.0	57.0±1.2	56.3±0.5	80.4±0.5	<b>58.7±0.7</b>	81.7±0.2
	Self-Training	32.7±0.2	56.9±0.2	56.9±0.3	81.7±0.2	55.7±0.3	82.3±0.2
	MoCo [66]	33.6±0.2	59.4±0.3	55.2±0.2	82.9±0.2	52.5±0.4	79.5±0.2
	MoCo + Self-Training	<b>39.4±0.3</b>	<b>64.4±0.5</b>	<b>58.2±0.5</b>	<b>84.4±0.2</b>	55.2±0.5	<b>82.9±0.2</b>
$U_{in} + U_{out}$	Pseudo-Label [97]	15.2±1.0	40.6±1.2	52.4±0.2	80.4±0.5	49.9±0.2	78.5±0.3
	Curriculum Pseudo-Label [22]	30.8±0.1	54.4±0.3	54.2±0.2	79.9±0.2	53.6±0.3	79.9±0.2
	FixMatch [148]	25.2±0.3	50.2±0.8	51.2±0.6	77.6±0.3	53.1±0.8	79.9±0.1
	Self-Training	32.5±0.5	56.3±0.3	<b>55.7±0.3</b>	81.0±0.2	<b>55.2±0.2</b>	<b>82.0±0.3</b>
	MoCo [66]	44.6±0.4	72.6±0.5	52.9±0.3	81.2±0.1	51.0±0.2	78.5±0.3
	MoCo + Self-Training	<b>48.6±0.3</b>	<b>74.7±0.2</b>	<b>55.9±0.1</b>	<b>82.9±0.2</b>	54.0±0.2	81.3±0.3

Table 4.4: **Results on Semi-Fungi benchmark.** We experiment on Semi-Fungi using the same hyper-parameters from Semi-Aves in Table 4.3. We can see similar conclusions: When training from scratch, MoCo + Self-Training performs the best and adding  $U_{out}$  can give an extra performance boost. With expert models, FixMatch and Self-Training (with or without MoCo) is often the best performing one, but the latter is more robust to the out-of-class data.

	Method	from scratch Top1	from ImageNet Top1	from iNat Top1
	Supervised baseline	18.5	40.4	47.7
	Supervised oracle	93.3	94.3	94.2
$U_{in}$	Pseudo-Label [97]	18.6	40.3	54.1
	Curriculum Pseudo-Label [22]	19.0	40.2	52.1
	FixMatch [148]	15.5	<b>44.1</b>	<b>59.8</b>
	Self-Training	20.3	42.4	50.4
	MoCo [66]	30.2	41.7	51.4
	MoCo + Self-Training	<b>32.0</b>	42.6	52.7
$U_{in} + U_{out}$	Pseudo-Label [97]	18.8	40.3	51.5
	Curriculum Pseudo-Label [22]	20.0	40.5	52.0
	FixMatch [148]	11.0	38.5	<b>53.3</b>
	Self-Training	19.7	<b>41.5</b>	49.3
	MoCo [66]	31.8	40.8	51.0
	MoCo + Self-Training	<b>32.9</b>	<b>41.5</b>	52.4

Table 4.5: **Results on Semi-iNat benchmark.** We experiment on Semi-iNat which is a larger dataset than Semi-Aves and Semi-Fungi. In this table, the validation set is for selecting best models during training, but not incorporated in the supervised loss. The conclusions are similar as Semi-Aves. When training from scratch, MoCo + Self-Training is the best. When initialized using expert models, FixMatch works the best when having in-class unlabeled data only, but Self-Training is more robust to out-of-class data.

rate of 0.0045 and a batch size of 64 for 75 epochs which matches the reported 60% Top-1 accuracy<sup>2</sup>. We use the ImageNet pre-trained model from torchvision [118].

**Data augmentation.** For the Semi-Fungi dataset, we first pre-process the images to have a maximum of 300 pixels for each side, while Semi-Aves has a maximum of 500 pixels and Semi-iNat has a maximum of 300 pixels. We use random-resize-crop to the size of  $224 \times 224$  and random-flipping for data augmentation, for all the methods except for MoCo and FixMatch. MoCo additionally uses Gaussian blur, color jittering, and random grayscale conversion, while FixMatch uses RandAugment [33].

**Hyperparameter search.** We found the Semi-SL methods to be sensitive to hyper-parameters such as learning rates, weight decay, etc. As noted in [115], a small validation set poses a risk of picking sub-optimal hyper-parameters. Moreover, labeled data is best used as a source of supervision. While k-fold cross-validation is an alternative, it is expensive. Hence, we use the combined training and validation set for training Semi-SL methods in our experiments and report performance on the test set which is sufficiently large. In particular, hyperparameters for all methods were based on the performance on the Semi-Aves dataset and kept fixed for the Semi-Fungi dataset (Tab. 4.4). Thus the results in Tab. 4.3 should be seen as a validation set performance, while those in Tab. 4.4 represent a novel test set. However, the high-level conclusions are identical across the two benchmarks.

**Semi-supervised training.** For Semi-SL methods except for FixMatch, we use SGD with a momentum of 0.9 and a cosine learning rate decay schedule [103] following [89, 148] for optimization. Learning rate and weight decay were picked from a range of [0.03, 0.0001]. We use a batch size of 64 during training. When there is unlabeled data, we select half of the batch from labeled and another half from unlabeled data (32 each). We train models for 10k and 50k iterations for training from expert models and from scratch. Other hyper-parameters include threshold  $\tau$  for Pseudo-Labeling, which we select from {0.80,

---

<sup>2</sup>[https://github.com/macaodha/inat\\_comp\\_2018](https://github.com/macaodha/inat_comp_2018)



0.85, 0.90, 0.95}. When training from scratch, we use  $\tau=0.85$  and 0.8 for with and without  $U_{out}$ ; when training from experts we use  $\tau=0.95$ . For Self-Training, we set  $T=1$  and  $\lambda=0.7$  for all the experiments. For FixMatch [148], we are able to train the model up to a batch size of 192 (32 labeled and 160 unlabeled images) on 4 GPUs. We find the performance drops significantly with small batch size (*e.g.* 48), however, we are unable to use the same batch size as original paper (*i.e.* 6144) due to limited resources. We use a learning rate of 0.01 and threshold  $\tau=0.80$  to train FixMatch for 500 epochs when training from scratch and 250 epochs with pre-trained models.

**Self-supervised training.** We adopt the default settings of MoCo-v2 [28], including MLP projector, 800 training epochs, *etc.*, but adapt the number of negative samples and learning rate to our task. We use a batch size of 256 and 2048 negative samples in all experiments. We find that using a large number of negative samples (*e.g.* 65,536) hurts the performance. When training the MoCo from scratch, we use the default learning rate of 0.03; when training MoCo from ImageNet or iNaturalist pre-trained model, we use a smaller learning rate (0.0003) and fewer training epochs (200) to avoid the potential forgetting problem. In the end, we train a classifier on the global average pooling features of ResNet-50 without freezing the backbone. We find that freezing the feature encoder always leads to worse performance than fine-tuning the entire network.

### 4.3.2 Results

Our experimental results on Semi-Aves, Semi-Fungi, and Semi-iNat are shown in Tab. 4.3, 4.4 and 4.5, respectively. To better visualize the results, we plot the relative gain of each Semi-SL method, *i.e.* the differences between supervised baseline in raw accuracy, on Semi-Aves and Semi-Fungi in Fig. 4.4. We discuss the results of each setting in the following.

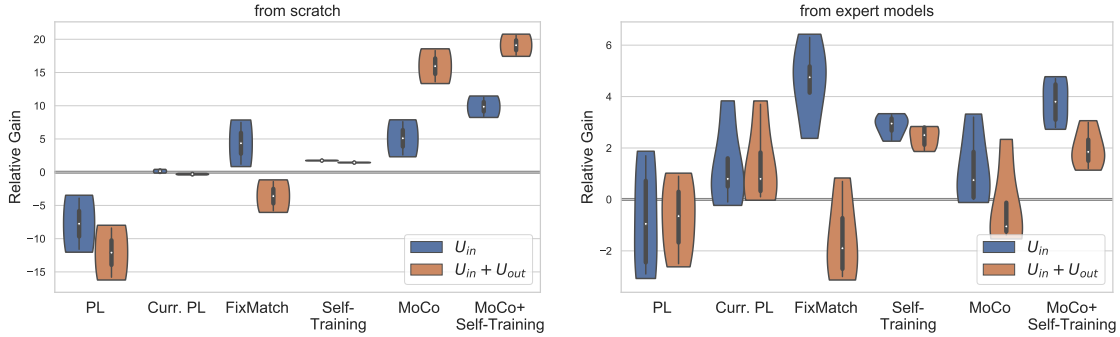


Figure 4.4: **Relative gains of Semi-SL methods on Semi-Aves and Semi-Fungi.** **Left:** trained from scratch. **Right:** using expert models. For each Semi-SL method, we plot the relative gain, *i.e.* the difference between the supervised baseline in raw accuracy, from the results in both Tab. 4.3 and 4.4. This shows that (1) the presence of out-of-class data  $U_{out}$  often hurts the performance, and (2) Self-Training is often the best method when using pre-trained models.

#### 4.3.2.1 Training from scratch.

We first discuss the results of training from scratch using only  $U_{in}$  on both datasets. Comparing to supervised baseline, Curriculum Pseudo-Label does not give improvements and Pseudo-Label even underperforms the baseline. This is possibly due to the low initial accuracy of the model which gets amplified during pseudo labeling. FixMatch and Self-Training both result in improvements. Self-supervised learning (MoCo) gives a good initialization and the improvements are similar or even more than using FixMatch. Finally, Self-Training using MoCo pre-trained model as the teacher model results in a further 2-3% improvement.

#### 4.3.2.2 Using expert models.

We then consider using an ImageNet or iNat pre-trained model for transfer learning with  $U_{in}$  only. The transfer learning baseline from either expert model outperforms the best Semi-SL method (MoCo + Self-Training) trained from scratch by a large margin, showing that transfer learning is more powerful in our realistic datasets. This observation echos Oliver *et al.* [115] who showed transferring from ImageNet to CIFAR10 performs

better than Semi-SL methods. Next, we can see that most of the Semi-SL methods, as well as MoCo pre-training, provide improvements over the baselines. The only exception is Pseudo-Label on Semi-Fungi. Among Semi-SL methods, FixMatch and MoCo + Self-Training perform the best.

#### 4.3.2.3 Effect of out-of-class unlabeled data.

Now we consider the setting where the unlabeled data contains both in-class and out-of-class data ( $U_{in} + U_{out}$ ). This is the trade-off between more unlabeled data at the cost of a distribution shift. This effect can be seen in the orange vs. blue plot in Fig. 4.4. When training from scratch, the performances of Pseudo-Label and FixMatch drop by 4-9%, while Curriculum Pseudo-Label and Self-Training only drop by less than 1%, showing that they are more robust to the domain shift of unlabeled data. On the other hand, self-supervised pre-training (MoCo) can benefit significantly from  $U_{out}$ , providing around 11% improvement over using  $U_{in}$  only on both Aves and Fungi datasets. Combining with Self-Training gives another 3-6% improvement, making the gap between transfer learning baseline smaller.

Finally, we consider having  $U_{in} + U_{out}$  with expert models. In Fig. 4.4 we can see the performance often drops in the presence of  $U_{out}$ . Curriculum Pseudo-Label and Self-Training are more robust and yield less than 1% decrease in most cases, while FixMatch is less robust whose performance drops by around 6%. The performances of MoCo also drops around 1-3% and are sometimes worse than the supervised baseline. Adding Self-Training however provides a 1-3% boost in performance. Overall, Self-Training from either a supervised or a self-supervised model is the most robust one.

#### 4.3.2.4 Robustness to hyper-parameters and trends.

We found Pseudo-Label to be sensitive to the threshold  $\tau$ . When using experts higher thresholds worked better. Increasing the threshold also increased the robustness in the presence of novel classes. Curriculum Pseudo-Label was found to be more robust in our benchmark, even when adding  $U_{out}$ . Self-Training was the most robust to hyper-parameters, we

chose the same temperature  $T$  and weight  $\lambda$  for all the experiments and it consistently improved results regardless of using an expert model or using out-of-domain data.

### 4.3.3 Comparison with related prior work on Semi-SL analysis

**On out-of-class unlabeled data.** Oliver *et al.* [115] showed that out-of-class unlabeled data negatively impacts performance, but analysis was done on CIFAR-10 with images from 6 labeled and 4 unlabeled classes. The classes are quite different making the problem of selecting in-domain images relatively easy in comparison to fine-grained domains — in our benchmarks the out-of-class data  $U_{out}$  are other species of birds or fungi. In fact, we show that more out-of-class data helps when using self-supervised and self-training methods trained from scratch. However, the additional data does not seem to help when initialized with experts.

**On transfer learning.** Oliver *et al.* showed a transfer learning accuracy of 87.9% on CIFAR-10 with 4k labels, outperforming many Semi-SL methods including PL [97] and VAT+EM [109]. Although recent results are better, the low resolution of CIFAR-10 ( $32 \times 32$  pixels) makes transfer learning from ImageNet less effective. On STL-10 that has a higher resolution ( $96 \times 96$  pixels), fine-tuning a ImageNet pre-trained ResNet-50 model on 5k labels provides 97.2% accuracy, while that trained on iNaturalist provides 95.0% accuracy. This beats 94.8% of FixMatch using 5k labeled examples when trained from scratch. Note that the iNaturalist dataset has no overlap with STL-10, yet transfer learning is effective.

### 4.3.4 Analysis on out-of-class unlabeled data

**The effect of threshold parameter for Pseudo-Label.** We found Pseudo-Label method is sensitive to the threshold parameter  $\tau$ . Fig. 4.5 plots the accuracy as a function of  $\tau$  with different unlabeled data and experts on Semi-Aves. A higher threshold performs better, especially in the presence of out-of-class data  $U_{out}$  as this excludes novel class images where the confidence of prediction is likely to be low. On the other hand, lower values work

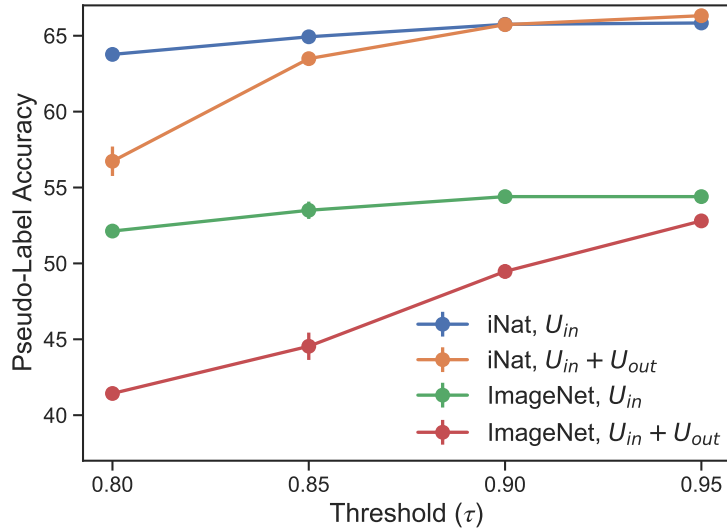


Figure 4.5: **Pseudo-label with different threshold  $\tau$** . Pseudo-label is sensitive to the threshold hyperparameter. The negative impact of out-of-class unlabeled data is reduced by increasing the threshold, yet when the initial performance is low the scheme is not effective as seen by the performance of the ImageNet pre-trained model.

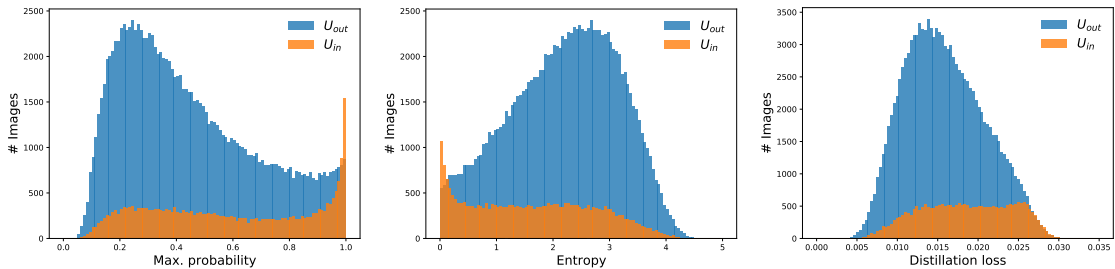


Figure 4.6: **Predictions of unlabeled data using a supervised model**. We plot the distribution of the predictions of data from  $U_{in}$  and  $U_{out}$ . Specifically, we plot the maximum probability of the class predictions (left), entropy of the predictions (middle), and the distillation loss between the teacher and student model before the training starts (right). Unlabeled data from the same distribution tend to have a higher maximum probability, a lower entropy, or a higher distillation loss.

just as well when unlabeled data is in-domain  $U_{in}$ . However, this scheme only appears to work when using strong experts (*e.g.*, iNat) whose confidence is likely calibrated, unlike random or ImageNet pre-trained model, where the presence of out-of-class data reduces

performance. This poses a practical problem for this method — increasing the threshold increases robustness but reduces the amount of unlabeled data that is used during training.

**The effect of out-of-class unlabeled data.** To see how the domain mismatch between  $U_{in}$  and  $U_{out}$  can affect Semi-SL methods, we analyze the predictions of the unlabeled data. We use the supervised model trained on  $L_{in}$  to compute the predictions of the unlabeled data on the Semi-Aves dataset. We plot the histogram of the maximum probability and the entropy of the predictions of  $U_{in}$  and  $U_{out}$  in Fig. 4.6 (left and middle). We also plot the distribution of the distillation loss, which is calculated between the supervised model (teacher) and the ImageNet pre-trained model (student), with a temperature  $T = 1$  (Fig. 4.6 right). This is in the beginning of the self-training process and the last layer of the student model is randomly initialized. Overall, the model is generally more uncertain about the out-of-class data, which often has a higher entropy or a smaller maximum probability. The distillation loss on  $U_{in}$  is also often higher than that of  $U_{out}$ , suggesting the model focuses more on those from  $U_{in}$  during training. However, there is still a good amount of data from  $U_{out}$  having a high maximum probability, which has a negative impact for pseudo-label methods.

## CHAPTER 5

### SEMI-SUPERVISED LEARNING WITH TAXONOMIC LABELS

In the previous chapter, we have shown that the performances of Semi-SL methods are still far below that of fully supervised training on Semi-Aves and Semi-iNat datasets. However, the label space of those datasets (biological taxonomy) is well-structured. In addition, one can obtain labels on the higher levels of the taxonomy at a lower cost. In this chapter, we investigate if those coarse labels can be incorporated in semi-supervised learning. In particular, can hierarchical labels improve the state-of-the-art semi-supervised learning algorithms?

We present an analysis on the Semi-iNat dataset [155], which was described in Chapter 4.1.3. Semi-iNat consists of images from 810 species, spanning three kingdoms and eight phyla. The dataset contains (1) a small set of images labeled at the species level, (2) a large set ( $9\times$ ) of coarsely-labeled images from the same species, and (3) an even larger set ( $32\times$ ) of coarsely-labeled images from the novel species within the same taxonomy. At test time, species classification accuracy is measured on novel images from the set of species within the labeled set. The dataset is fine-grained and naturally long-tailed posing challenges to existing approaches for semi-supervised learning.

We present results using a ResNet-50 network trained from scratch or on the ImageNet dataset using various semi- and self-supervised learning approaches. For supervised learning baseline with ImageNet pretraining, our experiments show that simply incorporating a hierarchical loss using the labels at the phylum level consisting of eight categories improves the top-1 species level classification accuracy from 40.4% to 46.6% (Table 5.2). This beats the gains using a state-of-the-art semi-supervised learning approach called FixMatch [148]

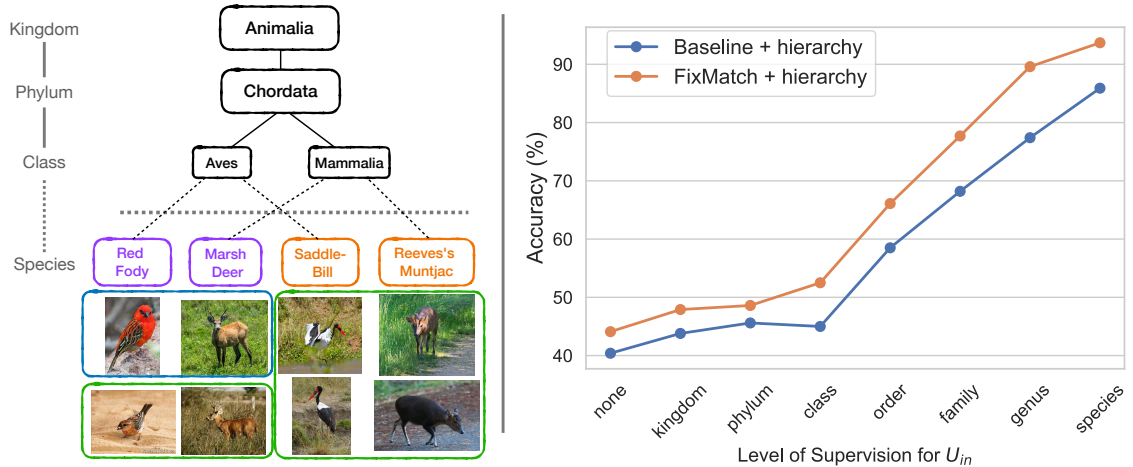


Figure 5.1: **Adding supervision on coarse label in the taxonomy improves semi-supervised learning.** **Left:** The Semi-iNat dataset has **labeled data** from **in-class species** and **coarsely labeled data** from both **in-class** and **out-of-class species**. The coarse labels include information in the kingdom and phylum levels. **Right:** Having more fine-grained labels improves the performance on both supervised baseline and semi-supervised methods.

which obtains 44.1%. However, the gains are complementary, and combining the two improves the performance further to 47.9%. Coarse taxonomic labels are also useful when models are trained from scratch, though in this setting self-supervised pretraining is also effective. We quantify the gains obtained directly using the hierarchical labels and when incorporating them within FixMatch across various levels of the hierarchy in the dataset from the Kingdom level (3 categories) to the Species level (810 categories, full supervision) (Figure 5.1 right).

The presence of out-of-domain data poses a challenge for learning. We find that the performance of nearly all approaches drops when the larger set of images are included during semi-supervised learning. For example, the performance of FixMatch drops from 47.9% to 41.1% (Table 5.3). This is problematic because labeling if an image contains one of the observed species is significantly more challenging, than say, obtaining a large pool of images with the same coarse labels as the species of interest. We present an approach



that selects relevant data from the set of coarsely labeled images guided by the hierarchy that improves the performance to 42.0%.

## 5.1 Related Works

**Semi-supervised learning.** Semi-supervised learning broadly aims to use weakly labeled data to improve the model generalization. Self-training approaches, proposed in some early work [106, 139] use the model’s own prediction to generate labels. Their modern incarnations include pseudo-labeling [97] which uses confident predictions as target labels for unlabeled data. Such labels can also be added gradually [9, 22] to reduce model drift. Another approach is to first train the model and then use its predictions to guide the training of a student model [183, 184, 195, 26] using distillation. Consistency-based approaches enforce the similarity of predictions between two augmentations of the same data as a form of supervision [6, 127, 95, 136, 109]. Recent examples of such techniques include MixMatch [11], ReMixMatch [10], FixMatch [148], and UDA [182], which combine geometric and photometric image augmentations with others such as MixUp [189]. Another line of works incorporates self-supervised learning for unlabeled data with supervised objectives. For example, one can combine self-supervised and supervised objectives to train a model jointly [188, 52, 156], or use self-supervised learning as pre-training [26] before fine-tuning with labeled data.

**Learning with hierarchical labels.** The hierarchical structure of the label space can be used to improve classification performance [133, 159, 60]. A common approach is to frame the problem as a structured prediction task to incorporate the structure. For example, Deng *et al.* [37] exploit the relations in the label spaces to improve classification using a probabilistic graphical model. Other works predict fine-grained labels by designing models that predict the labels [185, 194], or concatenating the features [174] learned from different levels in the hierarchy. The hierarchical label space can also be utilized to predict the label of the novel classes. For example, given coarse labels, Hsieh *et al.* [74] learn to assign fine-

grained pseudo-labels by meta-learning, assuming that the classifier can achieve the best performance when the missing labels are correctly recovered. Another application is zero-shot learning where attributes of the novel classes are provided, but there are no training images in novel classes. One can use graph convolution networks [137] or add novel nodes in the taxonomy tree [98] to leverage the hierarchical label space. Recently, coarse labels have also been incorporated in contrastive learning to improve image retrieval [164] and few-shot learning [19]. Unlike prior work, we investigate if hierarchical labels can be used in the context of semi-supervised learning, in particular to constrain the label space of techniques such as FixMatch or Pseudo-Labeling.

## 5.2 Method

**Notation and problem setting.** We focus on the structured prediction task where the label space  $y \in \mathcal{Y}$  has a hierarchical structure. It corresponds to a tree-structured biological taxonomy with 7 levels corresponding to the *kingdom*, *phylum*, *class*, *order*, *family*, *genus*, and *species*. Denote  $y^l$  as the label of an instance at the level  $l$ . Thus,  $y^1$  is the label at the kingdom level,  $y^2$  for the label in the phylum level, and the leaf nodes in the tree correspond to species-level labels denoted by  $y^7$ . Similarly, denote the sets of label space at a level  $l$  as  $\mathcal{C}^l$ . For example, the label space in the species level is  $y^7 \in \mathcal{C}^7$ , and in the phylum level  $y^2 \in \mathcal{C}^2$ . Given the label in a level, we can infer the labels in all the upper levels using the tree structure, *e.g.*, we can infer kingdom label given the phylum label, etc. The Semi-iNat [155] dataset provides species-level labels for a subset of images, but coarse labels (*e.g.*, kingdom and phylum) for a larger set of images (Table 5.1). Performance is measured as the accuracy at the species level on novel images.

### 5.2.1 Hierarchical supervised loss

We consider the supervised cross-entropy loss in each level of the hierarchy. For labeled data  $(x_i, y_i^7) \in \mathcal{L}$ , the model  $f$  first predicts the label space of the species  $\mathcal{C}^7$  with the

probabilities  $p_i^7 = f(x_i)$ . To use coarse labeled data, say at the phylum level  $(u_j, y_j^2) \in \mathcal{U}$  we apply a cross-entropy loss over the model’s prediction at the phylum level obtained by summing the probabilities of all the leaf nodes under each phylum. The marginalization can be done by  $p_i^2 = p_i^7 \cdot W_7^2$ , where the predefined matrix  $W_7^2$  represents the edges between the species and phylum level (the elements are 1 for the edges and 0 otherwise).

During training, we sample  $m$  labeled data and  $n$  coarsely labeled data in each batch for stochastic gradient descent. For labeled data, we can add supervised loss on all seven levels of the taxonomy. However, labels on the species level have strictly more supervision than on the other levels above, and empirically we also find that adding additional losses on higher levels does not help. Hence, we only add supervised loss on the lowest level possible. The complete hierarchical supervised loss is:

$$\mathcal{L}_{\text{hie}}^{7,2} = \sum_{i=1}^m H(y_i^7, p_i^7) + \sum_{j=1}^n H(y_j^2, q_j^2), \quad (5.1)$$

where  $H$  is the cross-entropy function  $H(u, v) = -\sum_w u(w) \log v(w)$ . The first term is the loss for labeled data on the species level, and the second term is the loss for coarsely labeled data on the phylum level. The superscript  $s, p$  on the loss  $\mathcal{L}_{\text{hie}}^{s,p}$  represents the level of supervision for labeled and coarsely labeled data. In the ablation studies, we will investigate the effect of different levels of supervision. Note that our method can be extended to general hierarchical graphs such as WordNet using marginalization methods [37, 137].

### 5.2.2 Joint training with semi-supervised loss

In addition to the hierarchical loss, we can add semi-supervised losses such as consistency regularization, entropy minimization, or pseudo-labeling on the species level for coarsely labeled data. We select representative semi-supervised methods including pseudo-label, FixMatch, Self-Training with distillation, and self-supervised training (MoCo) with distillation. We describe each method and how we incorporate hierarchical supervisions in the following.

**(1) Pseudo-Label [97].** Pseudo-label uses the model’s predictions as labels if the prediction is higher than a threshold  $\tau$ . Denote the pseudo-label in the species level as  $\hat{q}_i^7 = \operatorname{argmax}(q_i^7)$ , then the loss for pseudo-label training is:

$$\mathcal{L} = \mathcal{L}_{\text{hie}}^{7,2} + \sum_{j=1}^n \mathbb{1}[\max(r_i) \geq \tau] H(\hat{q}_i^7, q_i^7). \quad (5.2)$$

**(2) FixMatch [148].** FixMatch utilized two different augmentation functions for consistency training, one with weak augmentation  $\alpha(\cdot)$  and one with strong augmentation  $\mathcal{A}(\cdot)$ . For each coarsely labeled image  $u_j$ , the KL distance between the pseudo-label from weakly-augmented image  $q_j = f(\alpha(u_j))$  and the prediction of strongly-augmented image  $Q_j = f(\mathcal{A}(u_j))$  is minimized. To compute the supervised loss for labeled data, weak augmentation  $p_i = f(\alpha(x_i))$  is used. For unlabeled data, since the weakly-augmented data is only used for generating the pseudo-label without back-propagation, we add the supervised loss on strongly-augmented images  $Q_j = f(\mathcal{A}(u_j))$ . The final loss is:

$$\mathcal{L} = \underbrace{\sum_{i=1}^m H(y_i^7, p_i^7) + \sum_{j=1}^n H(y_j^2, Q_j^2)}_{\mathcal{L}_{\text{hie}}^{7,2}} + \sum_{j=1}^n \mathbb{1}[\max(q_i^7) \geq \tau] H(\hat{q}_i^7, Q_i^7). \quad (5.3)$$

**(3) Self-Training.** We use distillation [72] as the self-training method. Specifically, we first train a teacher model using labeled data for supervised learning  $\mathcal{L} = \sum_{i=1}^m H(y_i^7, p_i^7)$ . We then train a student model using distillation loss, which is the KL distance between the *logits* from the teacher and student models (denoted as  $z^t$  and  $z^s$ ). The final loss is:

$$\mathcal{L} = \mathcal{L}_{\text{hie}}^{7,2} + \sum_{i=1}^n H\left(\sigma\left(\frac{z_i^t}{T}\right), \sigma\left(\frac{z_i^s}{T}\right)\right), \quad (5.4)$$

where  $\sigma(\cdot)$  is the softmax function and  $T$  is the temperature parameter.

**(4) Self-Supervised Learning (MoCo) [66].** We use Momentum Contrastive (MoCo) [66] for self-supervised learning on the union of labeled and coarsely labeled data  $\mathcal{L} \cup \mathcal{U}$ . Specif-

ically, MoCo uses contrastive learning to minimize the representations of two different augmentations of an image. Denote  $q = f(x)$  as the representation of an image  $x$  and  $k^+$  as the positive sample, which is another augmentation of the same image  $x$ . The negative samples  $k_i^-$  are sampled from a memory bank. The InfoNCE [116] loss for the query  $q$  is:

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k^+/T)}{\exp(q \cdot k^+/T) + \sum_i^K \exp(q \cdot k_i^-/T)}, \quad (5.5)$$

where  $T$  is a temperature hyper-parameter. The encoder for the memory bank is updated based on momentum of the encoder  $f(\cdot)$  to stabilize the training. After the self-supervised pre-training is finished, we replace the MLP layers (after global pooling) with a linear projection layer and fine-tune the entire model using our supervised hierarchical loss (eq. 5.1).

**(5) MoCo + Self-Training.** This method combines the previous two methods, which is similar to the setting of Chen *et al.* [26]. We let the MoCo pre-trained model followed by supervised fine-tuning as the teacher model, then use the distillation to self-train a student model using the same loss in eq. 5.4.

## 5.3 Experiments

### 5.3.1 Experimental settings

**Dataset.** We use the Semi-iNat dataset [155] from the semi-supervised challenge at the FGVC8 workshop. The dataset has 810 in-class species and 1629 out-of-class species from 3 different kingdoms. The labeled data are from in-class species while the coarsely labeled data are drawn from in- and out-of-class species. Table 5.1 shows the statistics of the dataset, and Figure 5.1 left shows an example of the taxonomy. The competition version of the dataset provided the full taxonomies of the species-level data, while only coarse labels (kingdom and phylum level) are provided for the majority of the remaining data. However, we acquired species-level labels for all the images in the dataset from the

Kingdom	Phylum	$C_{in}$	$C_{out}$	Taxonomy	#Classes in $C_{in}$
Animalia (1294)	Mollusca	11	24	Kingdom	3
	Chordata	113	228	Phylum	8
	Arthropoda	301	605	Class	29
	Echinodermata	4	8	Order	123
Plantae (1028)	Tracheophyta	336	674	Family	339
	Bryophyta	6	12	Genus	729
Fungi (117)	Basidiomycota	29	58	Species	810
	Ascomycota	10	20		
Total #Classes		810	2439		

Table 5.1: **Statistics of the Semi-iNat dataset [155]. Left: Number of classes under each kingdom and phylum.** The species of Semi-iNat come from 3 kingdoms and 8 phyla. In each phylum, one-third of the species are used for in-class species  $C_{in}$  and the rest are used for out-of-class species  $C_{out}$ . **Right: Number of classes in each level of the taxonomy.**

competition organizers for a more detailed analysis of the utility of supervision at different levels in the hierarchy.

**Training details.** We use ResNet-50 [70] as our backbone model and an input size of  $224 \times 224$  for all the experiments. For all the methods except for MoCo and FixMatch, we use SGD with a momentum of 0.9 to train the model with 100k (from scratch) or 50k iterations (from expert models). The batch size is 60 for training supervised baselines. For semi-supervised methods, we sample 30 images each from labeled and coarsely labeled data with a total batch size of 60. The learning rate is searched within  $[0.001, 0.03]$ , and the weight decay is set for either 0.001 or 0.0001. The final hyper-parameters are set using the validation set of Semi-iNat, which is not included for the supervised loss but is used for training MoCo.

For MoCo, we follow the setting of MoCo-v2 [28] and use a batch size of 2048 negative samples. The training is done with a learning rate of 0.03 and 0.0003, and for 800 and 200 epochs, for training from scratch and from expert models respectively.

<b>Method</b>		<b>from scratch</b>		<b>from ImageNet</b>	
		w/o	w/	w/o	w/
$U_{in}$	Hierarchical Supervision (Phylum) $\rightarrow$				
	Supervised Learning Baseline	18.5	21.7 $\uparrow$	40.4	46.6 $\uparrow$
	Pseudo-Label [97]	18.6	22.7 $\uparrow$	40.3	44.9 $\uparrow$
	FixMatch [148]	15.5	25.7 $\uparrow$	<b>44.1</b>	<b>47.9<math>\uparrow</math></b>
	Self-Training	20.3	23.7 $\uparrow$	42.4	44.8 $\uparrow$
	MoCo [66]	30.2	33.5 $\uparrow$	41.7	41.9 $\uparrow$
MoCo + Self-Training [151]	<b>32.0</b>	<b>35.4<math>\uparrow</math></b>	42.6	45.8 $\uparrow$	

Table 5.2: **Results of adding hierarchical loss on the Semi-iNat dataset.** Adding hierarchical loss at the *phylum* level improves supervised training and all the semi-supervised methods when images come from  $U_{in}$ , *i.e.*, images with species labels corresponding to those in the test set. The best methods are shown in **teal**.

For FixMatch, we follow the original setting and use RandAugment [33] for augmentation. Due to the hardware constraints, we use a batch size of 32 for labeled data and 160 for coarsely labeled data for training using 4 GPUs. When training from scratch, we use a learning rate of 0.03 for 200k iterations; when training from expert models, we use a learning rate of 0.001 for 100k iterations. The threshold is set as 0.8 for all the settings.

### 5.3.2 Using phylum level supervision

We first consider the setting where all the images are within the set of labeled species. In § 5.3.4 we will analyze the effect and utility of adding novel species. Models are initialized randomly or from an ImageNet pre-trained model. For each setting, the baseline supervised learning and five semi-supervised learning methods are evaluated. We then analyze the effect of adding hierarchical loss.

Results are presented in Table 5.2. Adding a hierarchical loss gives almost a 10% improvement for FixMatch and 3% for all other methods when models are trained from scratch. When initialized with ImageNet pre-trained models, adding hierarchical loss provides 2-6% improvements in top-1 accuracy except for the self-supervised training method (MoCo). The confusion matrices at the phylum level for models trained with and with-

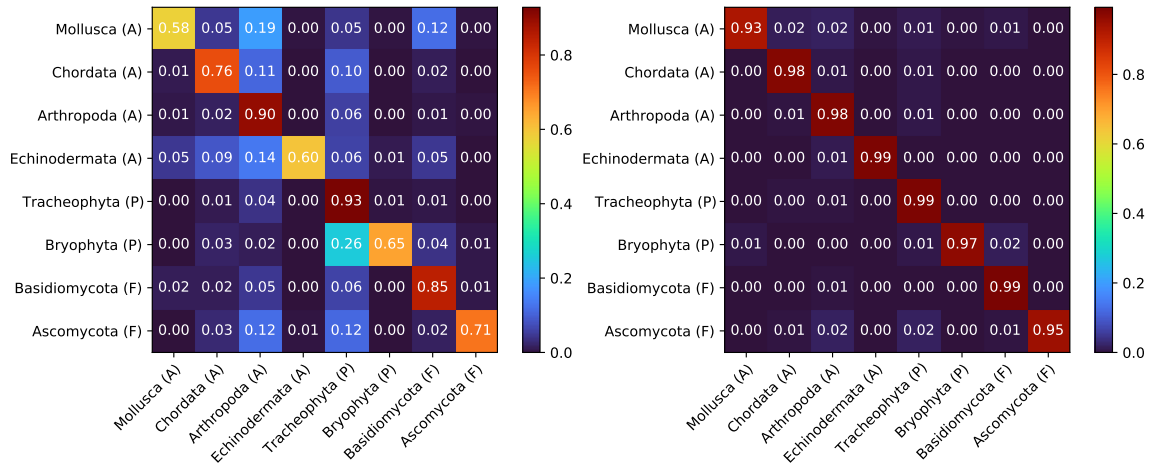


Figure 5.2: **Confusion matrices on the phylum level.** **Left:** Supervised baseline model without coarse-label supervision. **Right:** FixMatch + hierarchical loss on the phylum level. Combining semi-supervised methods and hierarchical supervision on coarsely labeled data reduces the confusion between phyla.

out hierarchical supervision are shown in Figure 5.2. Hierarchy supervision sensibly reduces confusion among the four phyla within the animal (A) kingdom (*e.g.*, Antropoda vs. Echinodermata), as well as at the plant (P) kingdom. The combined effect of hierarchical supervision and semi-supervised learning represents an overall improvement from 40.4% to 47.9%.

### 5.3.3 Using different levels of supervision

We consider supervision across levels of the taxonomy on top of FixMatch and supervised baseline. For example, if we have the labels in the order level for coarsely labeled data, *i.e.*  $(u_j, y_j^A) \in \mathcal{U}$ , then the hierarchical loss becomes  $\mathcal{L}_{\text{hie}}^{7,4}$ . As shown in Figure 5.1 right, we can see that finer-grained labels improve performance, but require more annotation effort. The number of classes on the taxonomy of Semi-iNat is shown in Table 5.1 right, which provides a rough proxy for the annotation effect. Even kingdom-level label (three categories) leads to small improvements, while class-level supervision (29 categories) improves the FixMatch performance (FixMatch + none) from 44.1% to 51.8% (FixMatch + class). Incorporating the hierarchical loss within FixMatch provides con-



<b>Method</b>		<b>from scratch</b>		<b>from ImageNet</b>	
		w/o	w/	w/o	w/
$U_{in} + U_{out}$	Hierarchical Supervision (Phylum) $\rightarrow$				
	Supervised Learning Baseline	18.5	20.5 $\uparrow$	40.4	<b>45.6</b> $\uparrow$
	Pseudo-Label [97]	18.8	21.2 $\uparrow$	40.3	44.0 $\uparrow$
	FixMatch [148]	11.0	21.1 $\uparrow$	38.5	41.1 $\uparrow$
	Self-Training	19.7	23.3 $\uparrow$	<b>41.5</b>	44.1 $\uparrow$
	MoCo [66]	31.8	29.4 $\downarrow$	40.8	39.3 $\downarrow$
	MoCo + Self-Training [151]	<b>32.9</b>	<b>35.4</b> $\uparrow$	<b>41.5</b>	42.6 $\uparrow$

Table 5.3: **Results on the Semi-iNat dataset when having out-of-domain data.** The out-of-domain data  $U_{out}$  degrades the performances of all the methods, except for the self-supervised method when training from scratch. However, adding hierarchical loss still gives improvements in most cases. The best methods are shown in **teal**.

sistent gains over the baseline hierarchical supervised across all levels in the taxonomy, suggesting that the benefits are complementary.

### 5.3.4 Effect of domain shift

Next, we consider the case when there exists domain shift, *i.e.* having  $U_{in} + U_{out}$ . The results are shown in Table 5.3. When training w/o coarse label supervision and initialized with ImageNet model, adding data from  $U_{out}$  degrades the performance by up to 4%, comparing to Table 5.2. In particular, FixMatch is less robust to the domain shift, echoing the findings in [151]. However, adding the hierarchical loss still gives up to 5% improvements except for MoCo. When training from scratch, we can see similar trends, but here MoCo + Self-Training performs the best.

To alleviate the effect of domain shift, we propose to filter the data by the confidence of the predicted labels before training semi-supervised methods. We first use the baseline model to generate predictions for images in  $U_{in} + U_{out}$ , then check if the maximum probability is greater than a threshold  $\tau = 0.8$ . We found that works well in practice, though more sophisticated out-of-domain detection techniques can be used. We additionally check if the coarse labels overlap with those in the  $U_{in}$  to filter out out-of-domain images. We

then train FixMatch using the selected coarse-labeled images. Using the supervision at the phylum level obtains an accuracy of 42.0% accuracy compared to 41.1% without any domain selection. This allows us to use uncurated data for improving performance, though the performance is lower than that of having only in-domain data (47.9%).

## CHAPTER 6

### CONCLUSION

Learning from few labeled data for visual recognition tasks is challenging. In Chapter 2, we used different modalities to improve image recognition through distillation. Yet, other modalities to explore include video and audio, image and text, and point clouds from LIDAR. Recently, many works are using contrastive learning for learning across modalities. Combining contrastive learning and distillation can further improve cross-modal learning.

Chapter 3 showed that self-supervised learning could improve few-shot learning, and extra unlabeled data can only help if the data is from a similar domain. We proposed to use a domain classifier for selecting images as the initial attempt. How to better incorporate the image selection process and self-supervised learning could be further investigated.

Chapter 4 created a new benchmark for semi-supervised learning, showing that existing methods are far from the performance of the oracle and are not effective as transfer learning. Moreover, our results show that the benefit of Semi-SL decreases when having out-of-class unlabeled data and initialized with expert models. We further leverage the coarse labels of the unlabeled data to improve semi-supervised learning in Chapter 5. However, semi-supervised learning is still far from being solved. One future direction is selecting unlabeled data during Semi-SL training, as simple thresholding the predictions (*e.g.* pseudo-label) does not improve. Another direction is to incorporate novel class predictions of the unlabeled data. Lastly, one of the standard Semi-SL methods is to use self-supervised pre-training followed by self-training. However, this method requires a separate training stage for self-supervised learning, and the representations may not be helpful later during

supervised fine-tuning. How to bridge self-supervised and semi-supervised learning needs further investigation.

## BIBLIOGRAPHY

- [1] Achille, Alessandro, Lam, Michael, Tewari, Rahul, Ravichandran, Avinash, Maji, Subhransu, Fowlkes, Charless, Soatto, Stefano, and Perona, Pietro. Task2Vec: Task embedding for meta-learning. In *International Conference on Computer Vision (ICCV)* (2019).
- [2] Antoniou, Antreas, Storkey, Amos, and Edwards, Harrison. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).
- [3] Athiwaratkun, Ben, Finzi, Marc, Izmailov, Pavel, and Wilson, Andrew Gordon. There are many consistent explanations of unlabeled data: Why you should average. *International Conference on Learning Representations (ICLR)* (2019).
- [4] Aytar, Yusuf, Vondrick, Carl, and Torralba, Antonio. Soundnet: Learning sound representations from unlabeled video. In *Neural Information Processing Systems (NeurIPS)* (2016).
- [5] Ba, Jimmy, and Caruana, Rich. Do deep nets really need to be deep? In *Neural Information Processing Systems (NeurIPS)* (2014).
- [6] Bachman, Philip, Alsharif, Ouais, and Precup, Doina. Learning with pseudo-ensembles. In *Neural Information Processing Systems (NeurIPS)* (2014).
- [7] Bachman, Philip, Hjelm, R Devon, and Buchwalter, William. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910* (2019).
- [8] Bengio, Yoshua, Delalleau, Olivier, and Le Roux, Nicolas. *Label Propagation and Quadratic Criterion*, semi-supervised learning ed. MIT Press, January 2006.
- [9] Bengio, Yoshua, Louradour, Jérôme, Collobert, Ronan, and Weston, Jason. Curriculum learning. In *International Conference on Machine Learning (ICML)* (2009).
- [10] Berthelot, David, Carlini, Nicholas, Cubuk, Ekin D, Kurakin, Alex, Sohn, Kihyuk, Zhang, Han, and Raffel, Colin. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *International Conference on Learning Representations (ICLR)* (2020).
- [11] Berthelot, David, Carlini, Nicholas, Goodfellow, Ian, Papernot, Nicolas, Oliver, Avital, and Raffel, Colin A. Mixmatch: A holistic approach to semi-supervised learning. In *Neural Information Processing Systems (NeurIPS)* (2019).

- [12] Bertinetto, Luca, Henriques, Joao F, Torr, Philip HS, and Vedaldi, Andrea. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)* (2019).
- [13] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam.
- [14] Blum, Avrim, and Mitchell, Tom. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory* (1998), pp. 92–100.
- [15] Bourdev, L., Maji, S., and Malik, J. Describing People: Poselet-Based Approach to Attribute Classification. In *International Conference on Computer Vision (ICCV)* (2011).
- [16] Branson, Steve, Horn, Grant Van, Belongie, Serge, and Perona, Pietro. Bird species categorization using pose normalized deep convolutional nets. In *British Machine Vision Conference (BMVC)* (2014).
- [17] Brock, André, Lim, Theodore, Ritchie, James M., and Weston, Nick. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236* (2016).
- [18] Buciluă, Cristian, Caruana, Rich, and Niculescu-Mizil, Alexandru. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), pp. 535–541.
- [19] Bukchin, Guy, Schwartz, Eli, Saenko, Kate, Shahar, Ori, Feris, Rogerio, Giryes, Raja, and Karlinsky, Leonid. Fine-grained angular contrastive learning with coarse labels. *arXiv preprint arXiv:2012.03515* (2020).
- [20] Caron, Mathilde, Bojanowski, Piotr, Joulin, Armand, and Douze, Matthijs. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)* (2018).
- [21] Caron, Mathilde, Bojanowski, Piotr, Mairal, Julien, and Joulin, Armand. Unsupervised pre-training of image features on non-curated data. In *International Conference on Computer Vision (ICCV)* (2019).
- [22] Cascante-Bonilla, Paola, Tan, Fuwen, Qi, Yanjun, and Ordonez, Vicente. Curriculum labeling: Self-paced pseudo-labeling for semi-supervised learning. *AAAI Conference on Artificial Intelligence (AAAI)* (2021).
- [23] Chapelle, Olivier, Scholkopf, Bernhard, and Zien, Alexander. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks* 20, 3 (2009), 542–542.

- [24] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)* (2014).
- [25] Chen, Ting, Kornblith, Simon, Norouzi, Mohammad, and Hinton, Geoffrey. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)* (2020).
- [26] Chen, Ting, Kornblith, Simon, Swersky, Kevin, Norouzi, Mohammad, and Hinton, Geoffrey. Big self-supervised models are strong semi-supervised learners. *Neural Information Processing Systems (NeurIPS)* (2020).
- [27] Chen, Wei-Yu, Liu, Yen-Cheng, Kira, Zsolt, Wang, Yu-Chiang, and Huang, Jia-Bin. A closer look at few-shot classification. In *International Conference on Learning Representations (ICLR)* (2019).
- [28] Chen, Xinlei, Fan, Haoqi, Girshick, Ross, and He, Kaiming. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297* (2020).
- [29] Chen, Zhao, Badrinarayanan, Vijay, Lee, Chen-Yu, and Rabinovich, Andrew. Grad-norm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)* (2018).
- [30] Chen, Zitian, Maji, Subhransu, and Learned-Miller, Erik. Shot in the dark: Few-shot learning with no base-class labels. *arXiv preprint arXiv:2010.02430* (2020).
- [31] Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., , and Vedaldi, A. Describing textures in the wild. In *Computer Vision and Pattern Recognition (CVPR)* (2014).
- [32] Cimpoi, M., Maji, S., and Vedaldi, A. Deep filter banks for texture recognition and description. In *Computer Vision and Pattern Recognition (CVPR)* (2015).
- [33] Cubuk, Ekin D, Zoph, Barret, Shlens, Jonathon, and Le, Quoc V. Randaugment: Practical automated data augmentation with a reduced search space. In *Computer Vision and Pattern Recognition (CVPR)* (2020).
- [34] Cui, Yin, Jia, Menglin, Lin, Tsung-Yi, Song, Yang, and Belongie, Serge. Class-balanced loss based on effective number of samples. In *Computer Vision and Pattern Recognition (CVPR)* (2019).
- [35] Cui, Yin, Song, Yang, Sun, Chen, Howard, Andrew, and Belongie, Serge. Large scale fine-grained categorization and domain-specific transfer learning. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [36] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition (CVPR)* (2009).

- [37] Deng, Jia, Ding, Nan, Jia, Yangqing, Frome, Andrea, Murphy, Kevin, Bengio, Samy, Li, Yuan, Neven, Hartmut, and Adam, Hartwig. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision (ECCV)* (2014), Springer.
- [38] DeVries, Terrance, and Taylor, Graham W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (2017).
- [39] Dhillon, Guneet S, Chaudhari, Pratik, Ravichandran, Avinash, and Soatto, Stefano. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729* (2019).
- [40] Doersch, Carl, Gupta, Abhinav, and Efros, Alexei A. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)* (2015).
- [41] Doersch, Carl, and Zisserman, Andrew. Multi-task self-supervised visual learning. In *International Conference on Computer Vision (ICCV)* (2017).
- [42] Dollár, Piotr, and Zitnick, C. Lawrence. Structured forests for fast edge detection. In *International Conference on Computer Vision (ICCV)* (2013).
- [43] Donahue, Jeff, Jia, Yangqing, Vinyals, Oriol, Hoffman, Judy, Zhang, Ning, Tzeng, Eric, and Darrell, Trevor. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning (ICML)* (2014).
- [44] Dosovitskiy, Alexey, Springenberg, Jost Tobias, Riedmiller, Martin, and Brox, Thomas. Discriminative unsupervised feature learning with convolutional neural networks. In *Neural Information Processing Systems (NeurIPS)* (2014).
- [45] Fei-Fei, Li, Fergus, Rob, and Perona, Pietro. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence* 28, 4 (2006), 594–611.
- [46] Finn, Chelsea, Abbeel, Pieter, and Levine, Sergey. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)* (2017).
- [47] Frome, Andrea, Corrado, Greg S, Shlens, Jon, Bengio, Samy, Dean, Jeff, Ranzato, Marc’Aurelio, and Mikolov, Tomas. Devise: A deep visual-semantic embedding model. In *Neural Information Processing Systems (NeurIPS)* (2013).
- [48] *2018 FGVCx Fungi Classification Challenge*. [https://github.com/visipedia/fgvcx\\_fungi\\_comp](https://github.com/visipedia/fgvcx_fungi_comp).
- [49] Furlanello, Tommaso, Lipton, Zachary C, Tschannen, Michael, Itti, Laurent, and Anandkumar, Anima. Born again neural networks. *International Conference on Machine Learning (ICML)* (2018).



- [50] Garcia, Victor, and Bruna, Joan. Few-shot learning with graph neural networks. In *International Conference on Learning Representations (ICLR)* (2018).
- [51] Ghiasi, Golnaz, Lin, Tsung-Yi, and Le, Quoc V. Dropblock: A regularization method for convolutional networks. In *Neural Information Processing Systems (NeurIPS)* (2018).
- [52] Gidaris, Spyros, Bursuc, Andrei, Komodakis, Nikos, Pérez, Patrick, and Cord, Matthieu. Boosting few-shot visual learning with self-supervision. In *International Conference on Computer Vision (ICCV)* (2019).
- [53] Gidaris, Spyros, and Komodakis, Nikos. Dynamic few-shot visual learning without forgetting. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [54] Gidaris, Spyros, Singh, Praveer, and Komodakis, Nikos. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)* (2018).
- [55] Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)* (2014).
- [56] Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)* (2014).
- [57] Goyal, Priya, Mahajan, Dhruv, Gupta, Abhinav, and Misra, Ishan. Scaling and benchmarking self-supervised visual representation learning. In *International Conference on Computer Vision (ICCV)* (2019).
- [58] Grandvalet, Yves, and Bengio, Yoshua. Semi-supervised learning by entropy minimization. In *Neural Information Processing Systems (NeurIPS)* (2005).
- [59] Grill, Jean-Bastien, Strub, Florian, Altché, Florent, Tallec, Corentin, Richemond, Pierre H, Buchatskaya, Elena, Doersch, Carl, Pires, Bernardo Avila, Guo, Zhao-han Daniel, Azar, Mohammad Gheshlaghi, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733* (2020).
- [60] Guo, Yanming, Liu, Yu, Bakker, Erwin M, Guo, Yuanhao, and Lew, Michael S. Cnn-rnn: a large-scale hierarchical image classification framework. *Multimedia tools and applications* 77, 8 (2018), 10251–10271.
- [61] Gupta, Saurabh, Hoffman, Judy, and Malik, Jitendra. Cross modal distillation for supervision transfer. In *Computer Vision and Pattern Recognition (CVPR)* (2016).
- [62] Gutmann, Michael, and Hyvärinen, Aapo. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), pp. 297–304.

- [63] Hacothen, Guy, and Weinshall, Daphna. On the power of curriculum learning in training deep networks. *International Conference on Machine Learning (ICML)* (2019).
- [64] Hadsell, Raia, Chopra, Sumit, and LeCun, Yann. Dimensionality reduction by learning an invariant mapping. In *Computer Vision and Pattern Recognition (CVPR)* (2006).
- [65] Hariharan, Bharath, and Girshick, Ross. Low-shot visual recognition by shrinking and hallucinating features. In *International Conference on Computer Vision (ICCV)* (2017).
- [66] He, Kaiming, Fan, Haoqi, Wu, Yuxin, Xie, Saining, and Girshick, Ross. Momentum contrast for unsupervised visual representation learning. In *CVPR* (2020).
- [67] He, Kaiming, Girshick, Ross, and Dollár, Piotr. Rethinking imagenet pre-training. In *International Conference on Computer Vision (ICCV)* (2019).
- [68] He, Kaiming, Gkioxari, Georgia, Dollár, Piotr, and Girshick, Ross. Mask r-cnn. In *International Conference on Computer Vision (ICCV)* (2017).
- [69] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)* (2016).
- [70] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. In *European conference on computer vision* (2016), Springer.
- [71] Hegde, Vishakh, and Zadeh, Reza. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695* (2016).
- [72] Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
- [73] Hjelm, R Devon, Fedorov, Alex, Lavoie-Marchildon, Samuel, Grewal, Karan, Bachman, Phil, Trischler, Adam, and Bengio, Yoshua. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations (ICLR)* (2019).
- [74] Hsieh, Cheng-Yu, Xu, Miao, Niu, Gang, Lin, Hsuan-Tien, and Sugiyama, Masashi. A pseudo-label method for coarse-to-fine multi-label learning with limited supervision. *ICLR LLD Workshop* (2019).
- [75] Huang, Gary, Mattar, Marwan, Lee, Honglak, and Learned-Miller, Erik G. Learning to align from scratch. In *Neural Information Processing Systems (NeurIPS)* (2012).
- [76] Ioffe, Sergey, and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)* (2015).

- [77] Jaderberg, Max, Simonyan, Karen, and Zisserman, Andrew. Spatial transformer networks. In *Neural Information Processing Systems (NeurIPS)* (2015).
- [78] Joachims, Thorsten. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning (ICML)* (1999).
- [79] Kalogerakis, Evangelos, Averkiou, Melinos, Maji, Subhransu, and Chaudhuri, Siddhartha. 3d shape segmentation with projective convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)* (2017).
- [80] Kanezaki, Asako, Matsushita, Yasuyuki, and Nishida, Yoshifumi. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. *arXiv preprint arXiv:1603.06208* (2016).
- [81] Kang, Bingyi, Xie, Saining, Rohrbach, Marcus, Yan, Zhicheng, Gordo, Albert, Feng, Jiashi, and Kalantidis, Yannis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217* (2019).
- [82] Kendall, Alex, Gal, Yarin, and Cipolla, Roberto. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [83] Khosla, Aditya, Jayadevaprakash, Nityananda, Yao, Bangpeng, and Fei-Fei, Li. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).
- [84] Kingma, Diederik P, and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [85] Klokov, Roman, and Lempitsky, Victor. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *International Conference on Computer Vision (ICCV)* (2017).
- [86] Koch, Gregory, Zemel, Richard, and Salakhutdinov, Ruslan. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop* (2015), vol. 2.
- [87] Kolesnikov, Alexander, Beyer, Lucas, Zhai, Xiaohua, Puigcerver, Joan, Yung, Jessica, Gelly, Sylvain, and Houlsby, Neil. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370* 6, 2 (2019), 8.
- [88] Kolesnikov, Alexander, Zhai, Xiaohua, and Beyer, Lucas. Revisiting self-supervised visual representation learning. In *Computer Vision and Pattern Recognition (CVPR)* (2019).
- [89] Kornblith, Simon, Shlens, Jonathon, and Le, Quoc V. Do better imagenet models transfer better? In *Computer Vision and Pattern Recognition (CVPR)* (2019).

- [90] Krause, Jonathan, Jin, Hailin, Yang, Jianchao, and Fei-Fei, Li. Fine-grained recognition without part annotations. In *International Conference on Computer Vision (ICCV)* (2015).
- [91] Krause, Jonathan, Stark, Michael, Deng, Jia, and Fei-Fei, Li. 3D object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3DRR)* (Sydney, Australia, 2013).
- [92] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NeurIPS)* (2012).
- [93] Kuo, Chia-Wen, Ma, Chih-Yao, Huang, Jia-Bin, and Kira, Zsolt. Featmatch: Feature-based augmentation for semi-supervised learning. *European Conference on Computer Vision (ECCV)* (2020).
- [94] Kuznetsova, Alina, Rom, Hassan, Alldrin, Neil, Uijlings, Jasper, Krasin, Ivan, Pont-Tuset, Jordi, Kamali, Shahab, Popov, Stefan, Mallocci, Matteo, Duerig, Tom, and Ferrari, Vittorio. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982* (2018).
- [95] Laine, Samuli, and Aila, Timo. Temporal ensembling for semi-supervised learning. *International Conference on Learning Representations (ICLR)* (2017).
- [96] Larsson, Gustav, Maire, Michael, and Shakhnarovich, Gregory. Learning representations for automatic colorization. In *European Conference on Computer Vision (ECCV)* (2016).
- [97] Lee, Dong-Hyun. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML* (2013).
- [98] Lee, Kibok, Lee, Kimin, Min, Kyle, Zhang, Yuting, Shin, Jinwoo, and Lee, Honglak. Hierarchical novelty detection for visual object recognition. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [99] Lee, Kwonjoon, Maji, Subhransu, Ravichandran, Avinash, and Soatto, Stefano. Meta-learning with differentiable convex optimization. In *Computer Vision and Pattern Recognition (CVPR)* (2019).
- [100] Lin, Tsung-Yu, RoyChowdhury, Aruni, and Maji, Subhransu. Bilinear CNN Models for Fine-grained Visual Recognition. *International Conference on Computer Vision (ICCV)* (2015).
- [101] Liu, Ziwei, Miao, Zhongqi, Zhan, Xiaohang, Wang, Jiayun, Gong, Boqing, and Yu, Stella X. Large-scale long-tailed recognition in an open world. In *Computer Vision and Pattern Recognition (CVPR)* (2019).

- [102] Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. *Computer Vision and Pattern Recognition (CVPR)* (2015).
- [103] Loshchilov, Ilya, and Hutter, Frank. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).
- [104] Maji, Subhransu, Rahtu, Esa, Kannala, Juho, Blaschko, Matthew, and Vedaldi, Andrea. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151* (2013).
- [105] Maturana, Daniel, and Scherer, Sebastian. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE International Conference on Intelligent Robots and Systems (IROS)* (2015).
- [106] McLachlan, Geoffrey J. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association* 70, 350 (1975), 365–369.
- [107] Meagher, Donald JR. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Electrical and Systems Engineering Department Rensselaer Polytechnic Institute Image Processing Laboratory, 1980.
- [108] Miller, Erik G, Matsakis, Nicholas E, and Viola, Paul A. Learning from one example through shared densities on transforms. In *Computer Vision and Pattern Recognition (CVPR)* (2000).
- [109] Miyato, Takeru, Maeda, Shin-ichi, Koyama, Masanori, and Ishii, Shin. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.
- [110] Mostajabi, Mohammadreza, Yadollahpour, Payman, and Shakhnarovich, Gregory. Feedforward semantic segmentation with zoom-out features. *Computer Vision and Pattern Recognition (CVPR)* (2015).
- [111] Ngiam, Jiquan, Khosla, Aditya, Kim, Mingyu, Nam, Juhan, Lee, Honglak, and Ng, Andrew Y. Multimodal deep learning. In *ICML* (2011).
- [112] Nilsback, M-E., and Zisserman, A. A visual vocabulary for flower classification. In *Computer Vision and Pattern Recognition (CVPR)* (2006).
- [113] Noroozi, Mehdi, and Favaro, Paolo. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)* (2016).
- [114] Noroozi, Mehdi, Pirsiavash, Hamed, and Favaro, Paolo. Representation learning by learning to count. In *International Conference on Computer Vision (ICCV)* (2017).

- [115] Oliver, Avital, Odena, Augustus, Raffel, Colin A, Cubuk, Ekin Dogus, and Goodfellow, Ian. Realistic evaluation of deep semi-supervised learning algorithms. In *Neural Information Processing Systems (NeurIPS)* (2018).
- [116] Oord, Aaron van den, Li, Yazhe, and Vinyals, Oriol. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).
- [117] Oreshkin, Boris, López, Pau Rodríguez, and Lacoste, Alexandre. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Neural Information Processing Systems (NeurIPS)* (2018).
- [118] Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, et al. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)* (2019).
- [119] Pathak, Deepak, Krähenbühl, Philipp, Donahue, Jeff, Darrell, Trevor, and Efros, Alexei A. Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition (CVPR)* (2016).
- [120] Peng, Xingchao, Hoffman, Judy, Yu, Stella X., and Saenko, Kate. Fine-to-coarse knowledge transfer for low-res image classification. In *IEEE International Conference on Image Processing (ICIP)* (2016).
- [121] Phong, Bui Tuong. Illumination for computer generated pictures. *Communications of the ACM* 18, 6 (1975), 311–317.
- [122] Qi, Charles, Su, Hao, Nießner, Matthias, Dai, Angela, Yan, Mengyuan, and Guibas, Leonidas. Volumetric and multi-view cnns for object classification on 3d data. In *Computer Vision and Pattern Recognition (CVPR)* (2016).
- [123] Qi, Charles R., Yi, Li, Su, Hao, and Guibas, Leonidas J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Neural Information Processing Systems (NeurIPS)* (2017).
- [124] Qi, Hang, Brown, Matthew, and Lowe, David G. Low-shot learning with imprinted weights. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [125] Qiao, Siyuan, Liu, Chenxi, Shen, Wei, and Yuille, Alan L. Few-shot image recognition by predicting parameters from activations. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [126] Qiao, Siyuan, Shen, Wei, Zhang, Zhishuai, Wang, Bo, and Yuille, Alan. Deep co-training for semi-supervised image recognition. In *European Conference on Computer Vision (ECCV)* (2018).
- [127] Rasmus, Antti, Berglund, Mathias, Honkala, Mikko, Valpola, Harri, and Raiko, Tapani. Semi-supervised learning with ladder networks. In *Neural Information Processing Systems (NeurIPS)* (2015).

- [128] Ravi, Sachin, and Larochelle, Hugo. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)* (2017).
- [129] Razavin, A. Sharif, Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. In *DeepVision workshop* (2014).
- [130] Rebuffi, Sylvestre-Alvise, Ehrhardt, Sebastien, Han, Kai, Vedaldi, Andrea, and Zisserman, Andrew. Semi-supervised learning with scarce annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2020), pp. 762–763.
- [131] Ren, Mengye, Triantafillou, Eleni, Ravi, Sachin, Snell, Jake, Swersky, Kevin, Tenenbaum, Joshua B, Larochelle, Hugo, and Zemel, Richard S. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations (ICLR)* (2018).
- [132] Riegler, Gernot, Ulusoy, Ali Osman, and Geiger, Andreas. Octnet: Learning deep 3d representations at high resolutions. In *Computer Vision and Pattern Recognition (CVPR)* (2017).
- [133] Ristin, Marko, Gall, Juergen, Guillaumin, Matthieu, and Van Gool, Luc. From categories to subcategories: large-scale image classification with partial class label refinement. In *Computer Vision and Pattern Recognition (CVPR)* (2015).
- [134] Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252.
- [135] Rusu, Andrei A, Rao, Dushyant, Sygnowski, Jakub, Vinyals, Oriol, Pascanu, Razvan, Osindero, Simon, and Hadsell, Raia. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960* (2018).
- [136] Sajjadi, Mehdi, Javanmardi, Mehran, and Tasdizen, Tolga. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Neural Information Processing Systems (NeurIPS)* (2016).
- [137] Samplawski, Colin, Learned-Miller, Erik, Kwon, Heesung, and Marlin, Benjamin M. Zero-shot learning in the presence of hierarchically coarsened labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2020), pp. 926–927.
- [138] Schwartz, Eli, Karlinsky, Leonid, Shtok, Joseph, Harary, Sivan, Marder, Mattias, Kumar, Abhishek, Feris, Rogerio, Giryes, Raja, and Bronstein, Alex. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *Neural Information Processing Systems (NeurIPS)* (2018).
- [139] Scudder, H. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory* 11, 3 (1965), 363–371.

- [140] Sedaghat, Nima, Zolfaghari, Mohammadreza, Amiri, Ehsan, and Brox, Thomas. Orientation-boosted voxel nets for 3d object recognition. *British Machine Vision Conference (BMVC)* (2017).
- [141] Sener, Ozan, and Koltun, Vladlen. Multi-task learning as multi-objective optimization. In *Neural Information Processing Systems (NeurIPS)* (2018).
- [142] Sermanet, Pierre, Frome, Andrea, and Real, Esteban. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054* (2014).
- [143] Sfikas, Konstantinos, Theoharis, Theoharis, and Pratikakis, Ioannis. Exploiting the panorama representation for convolutional neural network classification and retrieval. In *Eurographics Workshop on 3D Object Retrieval* (2017).
- [144] Shen, Yiru, Feng, Chen, Yang, Yaoqing, and Tian, Dong. Neighbors do help: Deeply exploiting local structures of point clouds. *arXiv preprint arXiv:1712.06760* (2017).
- [145] Simonyan, Karen, and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [146] Snell, Jake, Swersky, Kevin, and Zemel, Richard. Prototypical networks for few-shot learning. In *Neural Information Processing Systems (NeurIPS)* (2017).
- [147] Socher, Richard, Ganjoo, Milind, Manning, Christopher D, and Ng, Andrew. Zero-shot learning through cross-modal transfer. *Advances in neural information processing systems 26* (2013), 935–943.
- [148] Sohn, Kihyuk, Berthelot, David, Li, Chun-Liang, Zhang, Zizhao, Carlini, Nicholas, Cubuk, Ekin D, Kurakin, Alex, Zhang, Han, and Raffel, Colin. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Neural Information Processing Systems (NeurIPS)* (2020).
- [149] Su, Hang, Maji, Subhansu, Kalogerakis, Evangelos, and Learned-Miller, Erik G. Multi-view convolutional neural networks for 3d shape recognition. In *International Conference on Computer Vision (ICCV)* (2015).
- [150] Su, Hao, Qi, Charles, Mo, Kaichun, and Guibas, Leonidas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Computer Vision and Pattern Recognition (CVPR)* (2017).
- [151] Su, Jong-Chyi, Cheng, Zezhou, and Maji, Subhansu. A realistic evaluation of semi-supervised learning for fine-grained classification. In *Computer Vision and Pattern Recognition (CVPR)* (2021).
- [152] Su, Jong-Chyi, Gadelha, Matheus, Wang, Rui, and Maji, Subhansu. A deeper look at 3d shape classifiers. In *Second Workshop on 3D Reconstruction Meets Semantics, ECCV* (2018).



- [153] Su, Jong-Chyi, and Maji, Subhransu. Adapting models to signal degradation using distillation. In *British Machine Vision Conference (BMVC)* (2017).
- [154] Su, Jong-Chyi, and Maji, Subhransu. The semi-supervised inaturalist-aves challenge at fgvc7 workshop. *arXiv preprint arXiv:2103.06937* (2021).
- [155] Su, Jong-Chyi, and Maji, Subhransu. The semi-supervised inaturalist challenge at the fgvc8 workshop. *arXiv preprint arXiv:2106.01364* (2021).
- [156] Su, Jong-Chyi, Maji, Subhransu, and Hariharan, Bharath. When does self-supervision improve few-shot learning? In *ECCV* (2020).
- [157] Sun, Chen, Shrivastava, Abhinav, Singh, Saurabh, and Gupta, Abhinav. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision (ICCV)* (2017).
- [158] Sung, Flood, Yang, Yongxin, Zhang, Li, Xiang, Tao, Torr, Philip H.S., and Hospedales, Timothy M. Learning to compare: Relation network for few-shot learning. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [159] Taherkhani, Fariborz, Kazemi, Hadi, Dabouei, Ali, Dawson, Jeremy, and Nasrabadi, Nasser M. A weakly supervised fine label classifier enhanced by coarse supervision. In *International Conference on Computer Vision (ICCV)* (2019).
- [160] Tarvainen, Antti, and Valpola, Harri. Weight-averaged, consistency targets improve semi-supervised deep learning results. *CoRR*, vol. *abs/1703.2017* (1780).
- [161] Tian, Yonglong, Krishnan, Dilip, and Isola, Phillip. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699* (2019).
- [162] Tian, Yonglong, Krishnan, Dilip, and Isola, Phillip. Contrastive multiview coding. In *European Conference on Computer Vision (ECCV)* (2020).
- [163] Tian, Yonglong, Wang, Yue, Krishnan, Dilip, Tenenbaum, Joshua B, and Isola, Phillip. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539* (2020).
- [164] Touvron, Hugo, Sablayrolles, Alexandre, Douze, Matthijs, Cord, Matthieu, and Jégou, Hervé. Graft: Learning fine-grained image representations with coarse labels. *arXiv preprint arXiv:2011.12982* (2020).
- [165] Trinh, Trieu H, Luong, Minh-Thang, and Le, Quoc V. Selfie: Self-supervised pre-training for image embedding. *arXiv preprint arXiv:1906.02940* (2019).
- [166] Van Horn, Grant, Mac Aodha, Oisín, Song, Yang, Cui, Yin, Sun, Chen, Shepard, Alex, Adam, Hartwig, Perona, Pietro, and Belongie, Serge. The iNaturalist species classification and detection dataset. In *Computer Vision and Pattern Recognition (CVPR)* (2018).

- [167] Van Horn, Grant, Mac Aodha, Oisín, Song, Yang, Cui, Yin, Sun, Chen, Shepard, Alex, Adam, Hartwig, Perona, Pietro, and Belongie, Serge. The inaturalist species classification and detection dataset. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [168] Vapnik, Vladimir, and Vashist, Akshay. A new learning paradigm: Learning using privileged information. *Neural Networks* 22, 5 (2009), 544–557.
- [169] Vedaldi, Andrea, and Lenc, Karel. Matconvnet – convolutional neural networks for matlab. *Proceeding of the ACM International Conference on Multimedia* (2015).
- [170] Verma, Vikas, Lamb, Alex, Kannala, Juho, Bengio, Yoshua, and Lopez-Paz, David. Interpolation consistency training for semi-supervised learning. In *International Joint Conference on Artificial Intelligence* (2019).
- [171] Vinyals, Oriol, Blundell, Charles, Lillicrap, Timothy, Wierstra, Daan, et al. Matching networks for one shot learning. In *Neural Information Processing Systems (NeurIPS)* (2016).
- [172] Wallace, Bram, and Hariharan, Bharath. Extending and analyzing self-supervised learning across domains. In *ECCV* (2020).
- [173] Wang, Chu, Pelillo, Marcello, and Siddiqi, Kaleem. Dominant set clustering and pooling for multi-view 3d object recognition. In *British Machine Vision Conference (BMVC)* (2017).
- [174] Wang, Dequan, Shen, Zhiqiang, Shao, Jie, Zhang, Wei, Xue, Xiangyang, and Zhang, Zheng. Multiple granularity descriptors for fine-grained categorization. In *International Conference on Computer Vision (ICCV)* (2015).
- [175] Wang, Fang, Kang, Le, and Li, Yi. Sketch-based 3d shape retrieval using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)* (2015).
- [176] Wang, Peng-Shuai, Liu, Yang, Guo, Yu-Xiao, Sun, Chun-Yu, and Tong, Xin. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (SIGGRAPH)* 36, 4 (2017).
- [177] Wang, Yu-Xiong, Girshick, Ross, Herbert, Martial, and Hariharan, Bharath. Low-shot learning from imaginary data. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [178] Wang, Yue, Sun, Yongbin, Liu, Ziwei, Sarma, Sanjay E, Bronstein, Michael M, and Solomon, Justin M. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829* (2018).
- [179] Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. Caltech-UCSD Birds 200. Tech. Rep. CNS-TR-2010-001, California Institute of Technology, 2010.

- [180] Wu, Zhirong, Song, Shuran, Khosla, Aditya, Yu, Fisher, Zhang, Linguang, Tang, Xiaoou, and Xiao, Jianxiong. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition (CVPR)* (2015).
- [181] Wu, Zhirong, Xiong, Yuanjun, Yu, Stella X, and Lin, Dahua. Unsupervised feature learning via non-parametric instance discrimination. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [182] Xie, Qizhe, Dai, Zihang, Hovy, Eduard, Luong, Minh-Thang, and Le, Quoc V. Unsupervised data augmentation for consistency training. *Neural Information Processing Systems (NeurIPS)* (2020).
- [183] Xie, Qizhe, Luong, Minh-Thang, Hovy, Eduard, and Le, Quoc V. Self-training with noisy student improves imagenet classification. In *CVPR* (2020).
- [184] Yalniz, I Zeki, Jégou, Hervé, Chen, Kan, Paluri, Manohar, and Mahajan, Dhruv. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546* (2019).
- [185] Yan, Zhicheng, Zhang, Hao, Piramuthu, Robinson, Jagadeesh, Vignesh, DeCoste, Dennis, Di, Wei, and Yu, Yizhou. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *International Conference on Computer Vision (ICCV)* (2015).
- [186] Zaheer, Manzil, Kottur, Satwik, Ravanbakhsh, Siamak, Póczos, Barnabas, Salakhutdinov, Ruslan R, and Smola, Alexander J. Deep sets. In *Neural Information Processing Systems (NeurIPS)* (2017).
- [187] Zamir, Amir R, Sax, Alexander, Shen, William, Guibas, Leonidas J, Malik, Jitendra, and Savarese, Silvio. Taskonomy: Disentangling task transfer learning. In *Computer Vision and Pattern Recognition (CVPR)* (2018).
- [188] Zhai, Xiaohua, Oliver, Avital, Kolesnikov, Alexander, and Beyer, Lucas. S4L: Self-supervised semi-supervised learning. In *International Conference on Computer Vision (ICCV)* (2019).
- [189] Zhang, Hongyi, Cisse, Moustapha, Dauphin, Yann N, and Lopez-Paz, David. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)* (2018).
- [190] Zhang, N., Donahue, J., Girshick, R., and Darrell, T. Part-based R-CNNs for fine-grained category detection. In *European Conference on Computer Vision (ECCV)* (2014).
- [191] Zhang, Ning, Paluri, Manohar, Rantazo, Marc’Aurelio, Darrell, Trevor, and Bourdev, Lubomir. PANDA: Pose Aligned Networks for Deep Attribute Modeling. In *Computer Vision and Pattern Recognition (CVPR)* (2014).

- [192] Zhang, Richard, Isola, Phillip, and Efros, Alexei A. Colorful image colorization. In *European Conference on Computer Vision (ECCV)* (2016).
- [193] Zhang, Richard, Isola, Phillip, and Efros, Alexei A. Split-brain autoencoders: Un-supervised learning by cross-channel prediction. In *Computer Vision and Pattern Recognition (CVPR)* (2017).
- [194] Zhu, Xinqi, and Bain, Michael. B-cnn: branch convolutional neural network for hierarchical classification. *arXiv preprint arXiv:1709.09890* (2017).
- [195] Zoph, Barret, Ghiasi, Golnaz, Lin, Tsung-Yi, Cui, Yin, Liu, Hanxiao, Cubuk, Ekin D, and Le, Quoc V. Rethinking pre-training and self-training. *Neural Information Processing Systems (NeurIPS)* (2020).