Learning to Learn for Small Sample Visual Recognition

Yu-Xiong Wang

CMU-RI-TR-18-22

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

May, 2018

The Robotics Institute
Carnegie Mellon University

Thesis Committee:
Martial Hebert, Chair
Deva Ramanan
Ruslan Salakhutdinov
Andrew Zisserman, University of Oxford
Yann LeCun, Facebook AI Research & New York University

To my grandparents, my wife, and little wangwang.

And also, to infinite space and time.

Abstract

Understanding how humans and machines recognize novel visual concepts from few examples remains a fundamental challenge. Humans are remarkably able to grasp a new concept and make meaningful generalization from just few examples. By contrast, state-of-the-art machine learning techniques and visual recognition systems typically require thousands of training examples and often break down if the training sample set is too small.

This dissertation aims to endow visual recognition systems with low-shot learning ability, so that they learn consistently well on data of different sample sizes. Our key insight is that the visual world is well structured and highly predictable not only *in data and feature spaces* but also *in task and model spaces*. Such structures and regularities enable the systems to learn how to learn new recognition tasks rapidly by reusing previous *experiences*. This philosophy of *learning to learn*, or *meta-learning*, is one of the underlying tenets towards versatile agents that can continually learn a wide variety of tasks throughout their lifetimes. In this spirit, we address key technical challenges and explore complementary perspectives.

We begin by learning from extremely limited data (*e.g.*, one-shot learning). We cast the problem as *supervised knowledge distillation* and explore *structures within model pairs*. We introduce a meta-network that operates on the space of model parameters and encodes a *generic transformation* from "student" models learned from few samples to "teacher" models learned from large enough sample sets. By learning a series of transformations as more training data is gradually added, we further capture a notion of *model dynamics* to facilitate long-tail recognition with categories of different sample sizes. Moreover, by viewing the meta-network as an effective model adaptation strategy, we combine it with learning a generic model initialization and extend the use in few-shot human motion prediction tasks.

To further decouple a recognition model from ties to a specific set of categories, we introduce self-supervision using meta-data. We expose the model to a large amount of unlabeled real-world images through an *unsupervised meta-training* phase. By learning diverse sets of *low-density separators* across *auxiliary pseudo-classes*, we capture a more generic, richer description of the visual world. Since they are informative across different categories, we alternatively use the low-density separators to constitute an "off-the-shelf" library as *external memory*, enabling generation of new models on-the-fly for a variety of tasks, including object detection, hypothesis transfer learning, domain adaptation, and image retrieval. By doing so, we have essentially leveraged *structures within a large collection of models*.

We them move on to learning from a medium sized number of examples and explore *structures within an evolving model* when learning from continuously changing data streams and tasks. We rethink the dominant knowledge transfer paradigm that fine-tunes a fixed-size pre-trained model on new labeled target data. Inspired by developmental learning, we *progressively grow* a convolutional neural network with increased model capacity, which significantly outperforms classic fine-tuning approaches. Furthermore, we address *unsupervised fine-tuning* by transferring knowledge from *a discriminative* to *a generative* model on unlabeled target data. We thus make progress towards a lifelong learning process.

From a different perspective, humans can imagine what novel objects look like from different views. Incorporating this ability to hallucinate novel instances of new concepts and leveraging *joint structures in both data and task spaces* might help recognition systems perform better low-shot learning. We then combine a meta-learner with a "hallucinator" that produces additional training examples, and optimize both models jointly, leading to significant performance gains. Finally, combining these approaches, we suggest a broader picture of learning to learn predictive structures through exploration and exploitation.

Acknowledgments

An advisor is unique in one's life. At the very outset, I would like to extend my sincere gratitude to my advisor, Martial Hebert, for his unique influence over the past several years, as a mentor, collaborator, supporter, listener, advice giver, teacher, role model, and friend. When writing this dissertation, I remember every moment I have spent with Martial, especially those late afternoon discussions. I appreciate all that he has done for me, his remarkable energy in work, his enthusiasm for simple and solid research (of course that with elegant mathematics!), and his encouragement and consistent support especially in the face of failures and difficulties. All of these are the treasure for my future adventures.

I would also like to thank Deva Ramanan for being a fantastic mentor and a close friend in the last two years of my Ph.D. studies. I have enjoyed our time and the work we have done. I am sincerely grateful for his impact on my growth — teaching me how to think as a scientist, guiding me how to craft compelling stories, motiving me to insist on excellence, and exposing me to hands-on, real-world applications.

Thank you to my thesis committee members, Russ Salakhutdinov, Andrew Zisserman, and Yann LeCun, for their thoughtful advice, precious feedback, and flexibility throughout the entire process. I am thankful that I had the chance to work with Russ as well. He has been an invaluable source of insight, ideas, and practical guidance. I greatly appreciate Andrew and Yann for spending a precious amount of time and effort on my research, engaging in discussion of technical details, and providing useful ideas that drive me to think deeper. I am most influenced by Andrew for his devil-in-the-details spirit towards solid research and by Yann for his critical insight that breaks down complex problems into simple and manageable components and putting ideas in a broader research perspective.

Over the years, a number of people have also helped shape my research style and interests, and I have learned different useful skills from each one of them and relied on their advice. I especially thank Chris Atkeson, Drew Bagnell, Larry Davis, Fernando De la Torre, Alyosha Efros, David Forsyth, Bill Freeman, Abhinav Gupta, Daniel Huber, Takeo Kanade, Kris Kitani, Fei-Fei Li, Hongdong Li, David Lowe, Simon Lucey, Jitendra Malik, Matt Mason, Srinivas Narasimhan, Jean Ponce, Raj Reddy, Chuck Rosenberg, Yaser Sheikh, Leonid Sigal, Rahul Sukthankar, Rick Szeliski, Alan Yuille, Larry Zitnick, and Changshui Zhang for their guidance and encouragement. I would also like to thank members of Facebook AI Research for an excellent internship experience; the work I have done there becomes the last contribution of my dissertation. I am strongly indebted to Ross Girshick and Bharath Hariharan for their incredible mentorship. My grateful thanks are also extended to Dhruv Batra, Michael Cohen, Yann Dauphin, Piotr Dollár, Mohamed Elhoseiny, Georgia Gkioxari, Kaiming He, Kevin Matzen, Devi Parikh, Marcus Rohrbach, Mark Tygert, and Laurens van der Maaten for valuable and insightful discussions. I am also extremely grateful to my M.S. advisor, Yu-Jin Zhang, for introducing me to this fascinating world of vision and

learning.

I have also greatly benefited from colleagues and friends at the Smith Hall and Robotics Institute, for whom I have great respect and admiration. Thank to Aayush Bansal, Yang Cai, Xinlei Chen, Wen-Sheng Chu, Achal Dave, Tony Dear, Allison Del Giorno, Debadeepta Dey, Carl Doersch, David Fouhey, Rohit Girdhar, Liangke Gui, Hanzhang Hu, Peiyun Hu, Ed Hsiao, Hanbyul Joo, Hongwen Kang, Gunhee Kim, Chen Kong, Zhenzhong Lan, Yong-Jae Lee, Martin Li, Yin Li, Zhizhong Li, Minghuang Ma, Wei-Chiu Ma, Aravindh Mahendran, Kenneth Marino, Pyry Matikainen, Daniel Maturana, Ishan Misra, Ravi Teja Mullapudi, Dan Munoz, Luis Navarro-Serment, Ishan Nigam, Lerrel Pinto, Varun Ramakrishna, Nick Rhinehart, Olga Russakovsky, Scott Satkin, Kumar Shaurya Shankar, Abhinav Shrivastava, Gunnar Sigurdsson, Krishna Kumar Singh, Meng Song, Ekaterina Taralova, Yuandong Tian, Pavel Tokmakov, Yao-Hung Tsai, Jack Valmadre, Arun Venkatraman, Minh Vo, Jacob Walker, Xiaolong Wang, Fanyi Xiao, Pengtao Xie, Xuehan Xiong, Gengshan Yang, Shoou-I Yu, Zhiding Yu, Yin Zhang, Feng Zhou, Jiaji Zhou, Tinghui Zhou, and Jun-Yan Zhu, for discussions, feedback, and friendship. I am sincerely thankful to Pyry, Hongwen, Yuandong, David, Carl, Abhinav, and Ishan for their efforts in my research initialization, writing, and presentation.

Thank you to Suzanne Lyons Muth, Lynnetta Miller, Jess Butterbaugh, Christine Downey, Alan Guisewite, and Alison Day for all of their help during my time at the Robotics Institute. Thank you to Rocio Araujo for her help during my thesis proposal and defense.

The love, support and blessing showered on me by my family kept me motivated at my endeavour. I would specially like to thank my parents and my sister for their never-ending love, wisdom, and encouragement over the many years. I thank my parents-in-law for being the strongest support and a positive influence in my life.

Last, but far from least, I must thank my wife, Liangyan. *"Some of us get dipped in flat, some in satin, some in gloss. But every once in a while you find someone who's iridescent, and when you do, nothing will ever compare."* I appreciate her unconditional love, selfless sacrifice, unwavering support, and understanding through thick and thin as a wife and soul mate, and I appreciate her insights, comments, and contributions in proofreading and polishing my papers as a colleague. Special thanks to our little wangwang for the enjoyment and inspiration he has brought to us. Their smiles always warm my heart.

Support. This work was supported in part by ONR MURI N000141612007 and U.S. Army Research Laboratory (ARL) under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. We thank NVIDIA for donating GPUs and we also thank AWS Cloud Credits for Research Program and Facebook's Research and Academic Relations Program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Over the past decade, large-scale visual recognition has achieved high performance levels due to the integration of powerful machine learning techniques with big annotated training data sets [154, 209, 212, 226, 242, 321, 347, 357]. In practical applications, however, training examples are often expensive to acquire or otherwise scarce [112]. Visual phenomena follow an *intrinsic* long-tailed distribution, in which a few sub-categories are common while many are rare with limited training data even in the big-data setting [439, 440]. More crucially, current recognition systems assume a set of categories known a priori, despite the obviously dynamic and open nature of the visual world [30, 118, 263, 397].

Such scenarios of learning *novel categories from few examples* pose a multitude of open challenges for visual recognition in the wild. For instance, when operating in natural environments, robots are supposed to recognize unfamiliar objects after seeing only few examples [208]. Humans are remarkably able to grasp a new concept and make meaningful generalization to novel instances from just few examples [112, 326]. By contrast, typical machine learning techniques require tens, hundreds, or thousands of training examples and often break down if the training sample set is too small [21, 157].

Understanding the small-sample, or few/low-shot, learning mechanisms remains a fundamental challenge. In the classical learning framework, generating a recognition model from a single training example (or a scarce set of examples) is often infeasible due to overfitting effects. At a minimum, categorizing an object requires information about the category's mean and variance along each dimension in an appropriate feature space. The majority of the work can be cast as variations of this straightforward similarity-based approach, in which the mean represents the category prototype, and the inverse variances correspond to the dimensional weights in a category-specific similarity metric [326].

One-shot learning may seem impossible because a single example provides information about the mean or prototype of the category, but not about the variances or the similarity metric. Giving equal weight to every dimension in a high-dimensional feature space, or using the wrong similarity metric, is likely to be disastrous [326]. Hence, successful generalization from few training samples typically requires strong and appropriately tuned **inductive biases** using additional available information [25, 157]. Because of the highly predictive, structural patterns of the visual world, such information can be obtained by leveraging relationships between object categories.

This view has inspired a series of research works in the fields of one/few-shot learning [104], inductive transfer or transfer learning [288], multi-task learning [54], learning to learn [366], and meta-learning [328]. The existing work primarily focuses on learning structures in certain feature spaces between annotated object categories. Despite many notable successes, it is still unclear what kind of underlying structures and inductive biases are shared across a wide variety of categories and are useful for novel concepts.

Thesis Contributions. In this dissertation, we aim to endow visual recognition systems with low-shot learning ability, so that they learn consistently well on data of different sample sizes. Our key insight is that the visual world is *well structured and highly predictable* not only **in data and feature spaces** but also **in task and model spaces**. Such structures and regularities enable the systems to learn how to learn new recognition tasks rapidly by reusing previous *experience*, rather than considering each task in isolation.

This philosophy of **learning to learn**, or **meta-learning**, is one of the underlying tenets towards versatile agents that can continually learn a wide variety of tasks throughout their lifetimes. As defined in [370], given

- a family of tasks
- training experience for each of these tasks, and
- a family of performance measures (*e.g.*, one for each task),

an algorithm is said to *learn to learn* if its performance at each task improves with experience and with the number of tasks.

In this spirit, we address key technical challenges and leverage different and complementary perspectives, through knowledge distillation, unsupervised meta-learning, continual learning, and generative learning. We propose how these perspectives are equipped with powerful deep learning approaches to facilitate the recognition of novel visual concepts from few examples, producing state-of-the-art results for novel object and scene classification, fine-grained recognition, action recognition, domain adaptation, image retrieval, and human motion prediction.

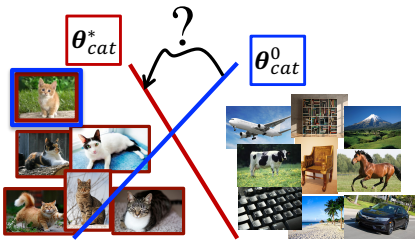
1.2 Organization

This dissertation is organized in the following four parts. The corresponding contributions were previously presented in [137, 138, 394–401].

1.2.1 Part I — Knowledge Distillation through Learning to Learn

We begin our journey by developing a series of conceptually simple but powerful approaches that can learn novel categories from extremely limited data, such as in one-shot learning. In this line of work, rather than generating a recognition model for a specific task in isolation, we cast small-sample recognition itself as a learning problem. We explore *structures within model pairs* that are shared among different recognition tasks on the model space. More precisely, we view a model learned from few annotated samples (as few as one and up to hundreds) as a "student" model and view its corresponding model learned from large enough sample sets (on the order of hundreds or thousands of) as a "teacher" model. We are then interested in inferring the relationship between these two types of models so that

it is possible to estimate the teacher model based on its student model. By viewing recognition tasks on existing categories as learning samples, we acquire the desired relationship through learning to learn. Such meta-knowledge is then distilled as useful inductive biases to facilitate the learning of novel classes.



Starting in Chapter 3 and [395], we initiate this learning to learn approach. Our main hypothesis is two-fold: (1) there exists a *generic transformation* from small-sample "student" models to the underlying large-sample "teacher" models *on the space of model parameters*, and (2) such a transformation could be effectively learned by high-capacity regressors, *e.g.*, deep neural networks. We empirically validate our hypothesis and introduce a *meta-network*, *i.e.*, a model regression network, to automatically learn the transformation on a large collection of model pairs. Experiments demonstrate that encoding this transformation as prior knowledge greatly facilitates the recognition in the small sample size regime on a broad range of tasks, including domain adaptation, fine-grained recognition, action recognition, and scene classification.

Instead of treating a recognition model being static, our approach analyzes how the model evolves for self-referential learning and self-improvement [335,336]. That is, we focus on how a recognition model changes during the learning process when gradually having access to more and more examples. By introducing this *task-level* meta-network, we *explicitly* guide the learning process of the apprentice — the small-sample model, to achieve similar levels of (generalization) performance as its teacher model. Such perspective is also broadly relevant to the recent work on learning to optimize [10, 236, 310], model distillation [18, 49, 159], and the analogy between recognition models and physical systems [331].

We further investigate the property of the model transformation. In particular, we assumed that the transformation is independent of the sample size whereas, in general, one would envision that the transformation would change when the number of samples increases dramatically all the way to identity for very large training sample sets. To tackle this limitation, we learn a sample size dependent transformation in Chapter 4 and [401]. The meta-network is instantiated as a residual network that learns the transformation graduality while explicitly maintaining its identity property. This thus allows our final meta-network to capture a notion of *model dynamics*, that predicts how model parameters are likely to change as more training data is gradually added.

Moreover, in Chapter 4 and [401], we naturally extend our approach to address a more challenging task: learn from long-tailed, imbalanced datasets that are prevalent in real-world settings. Here, the challenge is to learn accurate "few-shot" models for classes in the tail of the class distribution, for which little data is available. Often, the number of training examples varies considerably across different classes. We then transfer the meta-knowledge of model dynamics from the data-rich classes in the head of the distribution to the data-poor classes in the tail. Again, we train a meta-network to predict many-shot model parameters from few-shot model parameters. We transfer this meta-knowledge in a progressive manner, from classes in the head to the "body", and from the "body" to the tail. That is, we transfer knowledge in a gradual fashion, regularizing meta-networks for few-shot regression with those trained with more training data. We demonstrate results on image classification datasets tuned for the long-tailed setting, that significantly outperform

common heuristics, such as data resampling or reweighting.

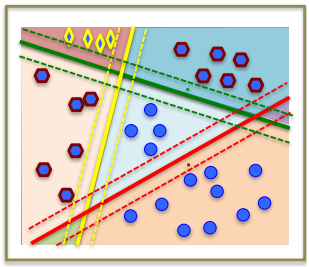
Our model regression meta-network can be also considered as an effective strategy for adapting a model to novel tasks. On the other hand, having a good generalization from few examples also relies on a generic initial model. To accomplish this, in Chapter 5 and [138], we propose *proactive and adaptive meta-learning* that introduces a novel combination of model-agnostic meta-learning and model regression networks and unifies them into an *integrated, end-to-end* framework. By doing so, on the one hand, our meta-learner produces a generic model initialization through aggregating contextual information from a variety of prediction tasks; on the other hand, this model is effectively adapted as a task-specific one by leveraging prior knowledge about how to transform few-shot model parameters to many-shot model parameters. Using this approach, we deal with a novel problem of *few-shot human motion prediction*. Human motion prediction, forecasting human motion in a few milliseconds conditioning on historical 3D skeleton sequence, is a long-standing problem in computer vision and robotic vision. Existing forecasting algorithms rely on extensive annotated motion capture data and are brittle for novel actions. Our predictor model significantly improves the prediction performance in the small-sample size regime.

1.2.2 Part II — Unsupervised Meta-Learning

The previous line of work is promising but still restrictive in the sense that the learned model structures and dynamics are tied to a specific set of categories due to its supervised nature. Even though current large-scale annotated datasets are comprehensive, they sample only a small fraction of the visual world biased to a selection of categories. It is still not clear how to take advantage of truly large sets of unlabeled real-world images, which constitute a much less biased sampling of the visual world.

In this part of the work, we develop a large-scale self-supervision approach to leveraging such unsupervised data sources as meta-data to improve the overall transferability of supervised convolutional neural networks (CNNs) and thus to facilitate the recognition of novel categories from few examples. We explore *structures within a large collection of models* and our approach is based on the informal intuition that, given a very large set of models, it is likely that some of the models would have good performance on a new recognition task, as stated in the infinite monkey theorem [1] and early research on random projection [43] and locality sensitive hashing [124].

Chapter 6 and [394] detail the proposed approach. Based on the transferability property of CNNs [427], conceptually, bottom and middle layers construct a feature space with high-density regions corresponding to potential latent categories. Top layer units in the pre-trained CNN, however, only have access to those regions associated with the original, observed categories. The units are then tuned to discriminate between these regions by separating the regions while pushing them further away from each other. To tackle this limitation, inspired by the intuition mentioned above, we introduce an additional *unsupervised meta-training* phase and expose multiple top layer units to a massive set of unlabeled images. We then encourage these units to generate diverse sets of low-density separations across the estimated pseudo-classes from the unlabeled data in activation spaces, which decouples them from ties to the original specific set of categories. We propose an unsupervised margin



maximization that jointly estimates compact high-density regions and infers low-density separators. The low-density separator modules can be plugged into any of the top layers of a standard CNN architecture. The resulting modified CNNs, *i.e.*, single-scale and multi-scale low-density separator networks, are fairly generic to a wide spectrum of novel categories, which significantly improve the performance in scene classification, fine-grained recognition, and action recognition with small training samples.

While these separations might not be meaningful in a semantic sense, the units have acquired certain experience by "playing with the data". Since *"a unit tries to discriminate the data manifold from its surroundings in all non-manifold directions"* [35], we capture a more generic, richer description of the visual world. Our low-density separators can be also viewed as an effective, discriminative compression of the unsupervised data. For a novel category, these separators *implicitly* connect its few examples with the corresponding latent pseudo-classes in the feature space, thus providing additional bits of information for the recognition task. While Chapter 6 provides a systematic way of integrating the low-density separators into a CNN architecture as a single unified model, Chapter 7 and Chapter 8 demonstrate that the separators could be, alternatively, used in an off-the-shelf fashion without fine-tuning. By doing so, we are able to produce a large collection of "off-line" models that are informative across different categories as *external memory*, and generate new models on-the-fly for a variety of tasks.

In Chapter 7 and [397], we leverage the unsupervised meta-training to generate object detectors in a way that is radically different from the conventional way of learning a detector from a large corpus of annotated positive and negative data samples. We assume that we have evaluated "off-line" a large library of detectors against a large set of detection tasks. Given a new target task, we evaluate a subset of the models on few samples from the new task and we use the matrix of models-tasks ratings to predict the performance of all the models in the library on the new task, enabling us to select a good set of detectors for the new task. This approach has three key advantages of great interest in practice: 1) generating a large collection of expressive models in an unsupervised manner is possible; 2) a far smaller set of annotated samples is needed compared to that required for training from scratch; and 3) recommending models is a very fast operation compared to the notoriously expensive training procedures of modern detectors. (1) will make the models informative across different categories; (2) will dramatically reduce the need for manually annotating vast datasets for training detectors; and (3) will enable rapid generation of new detectors.

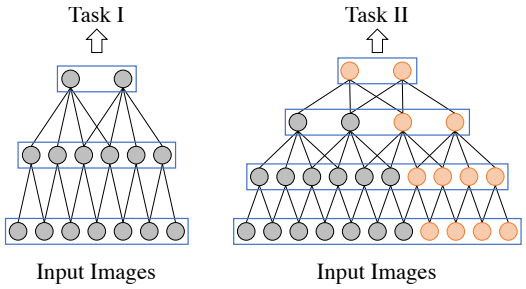
In Chapter 8 and [398], we consider a problem of hypothesis transfer learning. Category classifiers trained from a large corpus of annotated data are widely accepted as the sources for transfer or adaptation. Sources generated in this way are tied to a particular set of categories, limiting their transferability across a wide spectrum of target categories. We address this largely-overlooked yet fundamental source problem by introducing a systematic scheme for generating universal source hypotheses based on our unsupervised meta-training and proposing a principled, scalable approach to automatically tuning the transfer process. We demonstrate improvements over the state-of-the-art on domain adaptation and hypothesis transfer in the small sample size regime.

Moreover, in Chapter 8 and [399], we further address few-shot hash learning for image retrieval tasks. Current approaches to hash based semantic image retrieval assume a set of pre-defined categories and rely on supervised learning from a large number of annotated samples. The need for labeled samples limits their applicability in scenarios in which a user provides at query time a small set of training images defining a customized novel category. We then learn universal hash functions based on our unsupervised meta-learning and select

a task-specific combination of hash codes for a novel category from a few labeled samples. The resulting unsupervised generic hashing significantly outperforms current supervised and unsupervised hashing approaches on image retrieval tasks with small training samples.

1.2.3 Part III — Revisiting Fine-tuning in the Context of Continual Learning

We now consider learning novel categories from a medium sized number of examples and explore *structures within an evolving model* when learning from continuously changing data streams and tasks. For an autonomous agent, learning is a continual process. What it learns at one time-step while solving one task, it can use later, perhaps to solve a completely different task. Current computer vision systems are already moving more or less in this direction. This is usually accomplished through fine-tuning a *fixed-size* network on new *labeled* target data. Deep CNNs that are trained on a large enough, diverse "base" set of data (*e.g.*, ImageNet) exhibit certain attractive transferability properties for a broad range of tasks. Indeed, virtually every contemporary visual recognition system makes use of fine-tuning to transfer knowledge from ImageNet. However, an open question is how to best adapt a pre-trained CNN for novel categories and tasks. We address this issue here by rethinking the standard fine-tuning framework. We start in Chapter 9 by analyzing what components and parameters change during fine-tuning, and discover that increasing model capacity allows for more natural model adaption through fine-tuning.



More precisely, in Chapter 9 and [400], inspired by developmental learning, we explore *developmental neural networks* that grow in model capacity as new tasks as encountered. We demonstrate that the notion of growing a network, by adding additional units, helps facilitate knowledge transfer when encountering new tasks. We explore two mechanisms for adding units: either by widening existing layers or deepening the overall network, both of which significantly outperform classic fine-tuning approaches. But in order to properly grow a network, we show that newly added units must be appropriately normalized to allow for a pace of learning that is consistent with pre-existing units. Through visualizations and analysis, we demonstrate that developmental learning appears to regularize networks in a manner that encourages diversity of units and help guide the adaptation of pre-existing and new units.

In Chapter 10 and [137], we further consider tasks with different degree of supervision. In particular, we address "unsupervised fine-tuning" that transfers a pre-trained network to target tasks with unlabeled data such as image clustering tasks. To this end, we introduce group-sparse non-negative matrix factorization (NMF), a variant of NMF, to identify a rich set of high-level latent variables that are informative on the target task. The resulting *factorized convolutional network* can itself be seen as a feed-forward model that combines CNN and two-layer structured NMF. We empirically validate our approach and demonstrate state-of-the-art image clustering performance on challenging scene and fine-grained benchmarks. We further show that, when used as unsupervised initialization, our approach improves image classification performance as well.

Our observations thus support a developmental view of CNN optimization and recog-

nition model generation, in which model capacity is progressively grown throughout a lifelong learning process. While the specific approach was developed in the scenario of fine-tuning pre-trained CNNs, it suggests that such a principle might also benefit learning more generic CNN models from scratch: the network is self-growing and performing curriculum learning based on rich sets of *tasks*.

1.2.4 Part IV — Combining Generative Learning with Meta-Learning

In previous approaches, we have focused more on structures in task and model spaces. From a complementary perspective, humans can quickly learn new visual concepts, perhaps because they can easily visualize or imagine what novel objects look like in different poses or surroundings. In particular, many modes of variation (*e.g.*, camera pose, translation, lighting changes, and even articulation) are shared across categories. Incorporating this ability to hallucinate novel instances of new concepts and leveraging *joint structures in both data and task spaces* might help recognition systems perform better low-shot learning.

In Chapter 11 and [396], we present a novel approach to low-shot learning that uses this idea. While the images generated by current generative adversarial networks (GANs) are visually appealing, they tend to be biased to the few provided examples and do not follow the sample statistics in a desired feature space that are useful for recognition. The challenge then is to directly generate samples in the feature space with a good coverage of intra-class variation. Our approach combines a meta-learner with a "hallucinator" that produces additional training examples, and optimizes both models jointly. Our hallucinator can be incorporated into a variety of meta-learners, and provides significant gains irrespective of this choice.

Finally, in Chapter 12 we recap the dissertation by summarizing the contributions and discussing additional directions. In particular, by combining our approaches and perspectives, we suggest a broader picture of learning to learn predictive structures through exploration and exploitation. We also point out some other perspectives and potential applications beyond vision domains.

Chapter 2

Related Work

Understanding how a visual system recognizes novel categories from few examples and at a rapid pace is a fundamental challenge in the research of learning processes in both humans and machines [221]. While such an impressive competence has been demonstrated in humans [105, 365], *e.g.*, a six-year-old child has learned almost all of the $10 \sim 30$ thousand object categories in the world [42], the underlying mechanism remains a mystery. One widely used hypothesis is that humans make use of existing knowledge and experience acquired from previously learned categories when learning new ones [297].

In a similar spirit, successful generalization from small training sample sets, for artificial systems, requires strong and appropriately tuned "inductive biases" using additional available information [25, 157]. This has motivated a different yet relevant line of work in the fields of one/few-shot learning [104], inductive transfer or transfer learning [288], multi-task learning [54], learning to learn [366], and meta-learning [328]. Despite many notable successes, it is still unclear what kind of underlying structure and experience are shared across categories. In this thesis, we draw inspiration from the previous work but proposed new perspectives and leveraged state-of-the-art learning techniques through deep neural networks.

2.1 Data Manufacturing

For a learning task with limited training data, a natural source of information comes from additional data via "data manufacturing" [21]. This can be achieved in various ways. For instance, (1) obtain more examples of categories of interest from large amounts of unlabeled data as in semi-supervised learning [57, 305, 441] and active learning [293, 412], or from external annotated memory bank [417]; (2) augment the available examples by performing simple image transformations including jittering and noise injection as commonly used in deep learning [60, 95, 212] or complex category-independent example transformations [86, 144, 247]; (3) borrow examples from other relevant categories [239]; (4) introduce Universum examples (*i.e.*, unlabeled examples that do not belong to the concerned classes) for max-margin regularization [407]; and (5) synthesize new virtual examples, either rendered explicitly with computer graphics techniques or created implicitly through compositional representations [83, 94, 172, 267, 268, 289, 440]. These approaches can significantly improve recognition performance if a generative model that accounts for the underlying,

natural intra-class variability is known. Unfortunately, such a model is usually unavailable [21] and the generation of additional real or artificial examples often requires substantial effort.

Recently, a series of seminal work on generative adversarial networks (GANs) [23, 132, 257, 304, 314, 327, 433] casts new light on this problem. The main idea behind a GAN is to have two competing neural network models. The generator model takes noise as input and generates samples. The discriminator model receives samples from both the generator and the training data, and has to be able to distinguish between the two sources. Conceptually, this adversarial training procedure is useful for learning from few examples since it enables hallucinating additional examples. However, the current implementations of GANs, which primarily focus on directly generating natural images, bring limited performance boost in this case.

Often, these generative models have to be hand-designed for the domain, such as strokes [218,220] or parts [411] for handwritten characters. For more unconstrained domains, while there has been significant recent progress [132, 304, 316], modern generative models still cannot capture the entirety of the distribution [327]. Different classes might not share parts or strokes, but may still share modes of variation, since these often correspond to camera pose, articulation, *etc*. If one has a probability density on transformations, then one can generate additional examples for a novel class by applying sampled transformations to the provided examples [86, 144, 258].

Learning such a density is easier for handwritten characters that only undergo 2D transformations [258], but much harder for generic image categories. This problem is tackled by leveraging an additional dataset of images labeled with pose and attributes [86]; this allows to learn how images transform when the pose or the attributes are altered. To avoid annotation, transformations are transferred from a pair of examples from a known category to a "seed" example of a novel class [144]. However, learning to do this transfer requires a carefully designed pipeline with many heuristic steps. Our approach in Part IV follows this line of work, but learns to do such transformations by combing with meta-learning in an end-to-end manner, avoiding both brittle heuristics and expensive annotations. We present a unified view of meta-learning and show that our hallucination strategy can be adopted in any of the existing meta-learning methods.

2.2 Transfer Learning

In a broad sense, learning novel categories is addressed by exploiting and transferring knowledge gained from familiar categories [13,33,54,102,195,288,292,306,318,324,366,371, 372]. This is to imitate the human ability to adapt previously acquired experience when performing a new task [297].

2.2.1 Classic Transfer Learning

In the framework of classical machine learning, cross-generalization [21] and inter-class transfer [157] are typically achieved by discovering shared feature representations: (1) captured by linear or nonlinear feature transformations [6,54,116,198,258,369,390], (2) obtained by feature selection [108, 234, 235] or regularization [144], (3) described by similarities between novel classes and familiar classes [22], (4) encoded as a distance metric by metric learning [26, 29, 113, 303, 388, 409] or kernel learning [157], and (5) learned by boosting ap-

proaches [282, 374, 410]. In addition, contrastive loss functions [141, 202] and variants of the triplet loss [113, 338, 358] have been used for learning feature representations suitable for few-shot learning; the idea is to push examples from the same class closer together, and farther from other classes. Similarly, classifiers trained on small datasets are also encouraged to match those trained on large datasets by a carefully designed loss function [144].

Another type of knowledge transfer focuses on modeling (hyper-)parameters that are shared across domains, typically in the context of generative statistical modeling [50, 103, 232, 317]. A variational Bayesian framework is first developed by incorporating previously learned classes into the prior and combining with the likelihood to yield a new class posterior distribution [103, 104]. Gaussian processes [222, 317] and hierarchical Bayesian models [326] are also employed to allow transferring in a non-parametric Bayesian way. The recently proposed hierarchical Bayesian program learning utilizes the principles of compositionality and causality to build a probabilistic generative model of visual objects [218–220]. In addition, adaptive SVM and its variants present SVM-based model adaptation by combining classifiers learned on related categories [13, 14, 96, 196, 217, 373, 398, 418].

2.2.2 Transfer of Deep Neural Networks

In the era of deep learning, due to the demonstrated promising transferability of deep convolutional neural networks (CNNs) [226, 427], knowledge transfers in a more consistent manner [16, 33, 126, 283, 311]. It is common to pre-train a CNN on a large annotated dataset (*e.g.*, ImageNet [321] or Places [437]), and then use the CNN for target tasks in different ways: (1) as a fixed feature extractor [311]; (2) as an initialization for fine-tuning [126, 283]; and (3) as a guidance to transfer of knowledge across different modalities [17, 118, 140, 378, 381]. These transfer techniques are developed directly on particular tasks. By contrast, in this thesis we address a more general scenario that improves the overall transferability of CNNs. More importantly, in the case of limited target data such as one/few-shot learning, (2) and (3) are typically infeasible due to over-fitting effects, and (1) leads to degraded performance. Our approaches are developed to tackle these limitations.

A representative line of work addresses empirical evaluation of factors that affect the transfer performance in the conventional fine-tuning framework [15, 70, 173, 311, 434]. In Part III, however, we explore a better alternative to fine-tuning for the recognition of novel categories. While we investigate how to increase model capacity to improve learning on the target task, other contemporary work focuses on preserving the original capability on the source task during transfer [119, 238]. More relevant to our work, the progressive network [322] expands networks for reinforcement learning — a problem different from ours. Notably, [322, 363] expand all the layers, leading to a target network twice as wide as the source one, and they only train the new branch while freezing the original branch as off-the-shelf features. This strategy does not apply to our recognition tasks that typically do not have enough data to tackle such large growth in number of parameters. By contrast, we add a small fraction of new units and fine-tune the entire network. As ad-hoc approaches, [283] plug in a new adaptation layer to facilitate transfer for specific tasks, whereas their focus is not on the dynamic augmentation of model capacity as ours. In addition to unsupervised fine-tuning, we also explore unsupervised fine-tuning in Part III that transfers a pre-trained network to target tasks with unlabeled data.

Our approach in Part III is reminiscent of developmental learning [174, 273, 345], and is relevant to multi-task learning [54, 140, 262, 381] and lifelong learning [147, 265, 296, 368]. Different from the non-parametric shallow models (*e.g.*, nearest neighbors) that increase

capacity when memorizing new data [367, 370], our developmental network cumulatively grows its capacity *from novel tasks*. We learn predictive and recurrent structures in the model parameter space from multiple tasks [7], but with dynamically growing parameters.

2.3 Unsupervised and Semi-Supervised Learning

In Part II, we combine both supervised and unsupervised learning. There has been growing interest in learning CNNs in semi-supervised, self-supervised, or unsupervised fashions to extract generic features [90, 95, 134, 190, 202, 264, 393, 408]. Most existing unsupervised deep learning approaches focus on unsupervised learning of visual representations that are both sparse and allow image reconstruction [281], including unsupervised deep belief networks (DBN) [162], convolutional sparse coding [339], and multi-layer (denoising) auto-encoders (DAE) [36, 224]. Other types of supervisory information, such as clustering [73], surrogate classes [95], spatial context [90], temporal consistency [133, 134, 393], and image captions [190], have been explored to train CNNs in an unsupervised manner. Although showing initial promise, the performance of these unsupervised deep models is still not on par with that of their supervised counterparts. Instead of performing greedy layer-wise unsupervised pre-training in preparation for supervised training (*e.g.*, in DBN and DAE), our unsupervised meta-training in Part II "post-arranges" the supervised CNNs.

Another line of work trains deep multi-layer architectures in a semi-supervised fashion, by jointly learning an embedding task [408] or introducing additional entropy regularization [4, 229] with unlabeled data. The methods typically improve the model generalization for specific tasks, with both labeled and unlabeled data coming from the tasks of interest. Our scenario, however, bears some similarities to self-taught learning [306]. We improve the overall generality of CNNs for a wide spectrum of unseen categories by leveraging truly large sets of unlabeled real-world images.

A key assumption in many semi-supervised and unsupervised algorithms is the structure assumption: the decision boundary should not cross high-density regions, but instead lie in low-density regions [27, 56, 186, 408]. Given this assumption, our low-density separators in Part II aim to use unlabeled data to uncover this structure. Many algorithms begin by inferring a probability distribution from random samples, for example through density estimation [84], level set estimation [379], densest region detection [28], lowest-density hyperplane estimation [27], and clustering [46, 165]. All of these tasks, however, are notoriously difficult with respect to both sample complexity and computational complexity. To deal with this, we follow an alternative discriminative framework as in transductive SVM [56, 186], semi-supervised SVM [37], and predictable discriminative binary codes [309]. Given that most of these existing approaches [37, 186, 309] require certain labeled data, we then propose an *unsupervised margin maximization* to achieve low-density separation. Our work can be thus viewed as combining both neural networks and the max-margin principle into a unified framework.

2.4 Learning How to Learn

In Part I, we address the difference between a recognition model learned from few annotated samples and the underlying model that would be learned from a large set of samples. We meta-learn a generic, category agnostic transformation between small-sample and large-sample models by a regression neural network in a model-level big-data setting. Our ap-

proach could be seen as an alternative parametric way of doing model distillation that relies on the connection between different models (*i.e.*, a student and a teacher model) [18,49,159]. While model distillation focuses on models with different capacity but learned on the same data, our approach addresses models learned on different sized datasets and the acquired meta-knowledge is more explicitly represented.

In the context of learning to learn [335,366], we use the obtained inter-class *model structure* to modify a small-sample recognition model of a novel category. Our approach is relevant to the early and contemporary work on parameter prediction with one network that modifies the weights of another [41,332–334] and the recent work on inducing the weights of an image classifier from text [280, 352] or predicting network parameters from activations [300, 302]. From an optimization perspective, our approach is relevant to the recent research on learning to optimize which replaces hand-designed update rules with a learned update rule [10,236,310] and learning easily adaptable model parameters through gradient descent [114]. From a high-level perspective, this line of work casts a set of related learning tasks themselves as a learning problem.

Another different perspective is to capture knowledge common among a set of one-shot learning tasks during meta-training, and then to use the knowledge for a novel one-shot learning problem [41, 310, 388]. By virtue of deep learning, the recent approaches in this family typically learn a similarity distance metric via neural networks [351]. Flagship techniques include the early work of Siamese networks [48, 202] and their advanced version matching networks [388], which are augmented with attention and memory, as well as other modifications [259,351,354,376]. Due to the non-parametric nature (*i.e.*, using nearest neighbors as classifiers), these approaches are best effective for one-shot and extremely-few-shot learning and their performance significantly degenerates when the number of training samples increase. By contrast, due to its discriminative nature, our approach in Part I is consistently effective across datasets of different sizes, as shown in [144]. Using additional large-sample models as the explicit teacher also facilitates our meta-learning problem.

Part I

Learning to Learn: Knowledge Distillation through Model Regression Networks

The only person who is educated is the one who has learned how to learn and change.

Carl Rogers

Chapter 3

Learning Model Transformation for Easy Small Sample Learning

3.1 Motivation

Starting from this chapter, we explore a series of novel *learning to learn* approaches that leverage the knowledge gained when learning models in large sample sets to facilitate recognizing novel categories from extremely limited samples, such as in one-shot learning. From a discriminative machine learning perspective, object recognition is basically a process that learns an object category classifier to separate annotated positive and negative examples in a feature space. We assume a *fixed, discriminative* feature space, which is reasonable especially considering the recent learned feature representations via deep convolutional neural networks (CNNs). We now take the model such as SVM classifiers and make important modifications. The central issue can be reduced to the following: how to estimate a classifier that would be learned from a large set of samples (on the order of hundreds or thousands of) based on its corresponding classifier learned from few annotated samples (as few as one and up to hundreds)?

Our main hypothesis is that there exists **a generic nonlinear transformation** from small-sample (or few-shot) models to the underlying large-sample (or many-shot) models *for a variety of categories*. This hypothesis is validated empirically in Section 3.3. Intuitively, a model can be viewed as a separating hyperplane in the feature space.[1] Small training examples already constrain the search space by pointing to an initial hyperplane not far from the desired hyperplane produced by a large training set. When we gradually introduce additional examples, the initial hyperplane is progressively subject to a series of transformations until it converges, as illustrated in Figure 3.1.

We suspect that this transformation, or at least certain components of it, is fairly generic. In a machine learning context, a learner needs to be biased in some way for it to generalize well [25, 112, 157, 326]. Consequently, there might exist some systematic bias from a small-sample model to its large-sample version. In essence, this transformation potentially captures the natural intra-class variability in a discriminative manner and represents how sparse samples change to a category cluster. Hence, we view the model transformation as a form of shared structure and, when available, it can be re-purposed for novel categories.

[1] A kernel model can be viewed as a separating hyperplane in the lifted feature space.

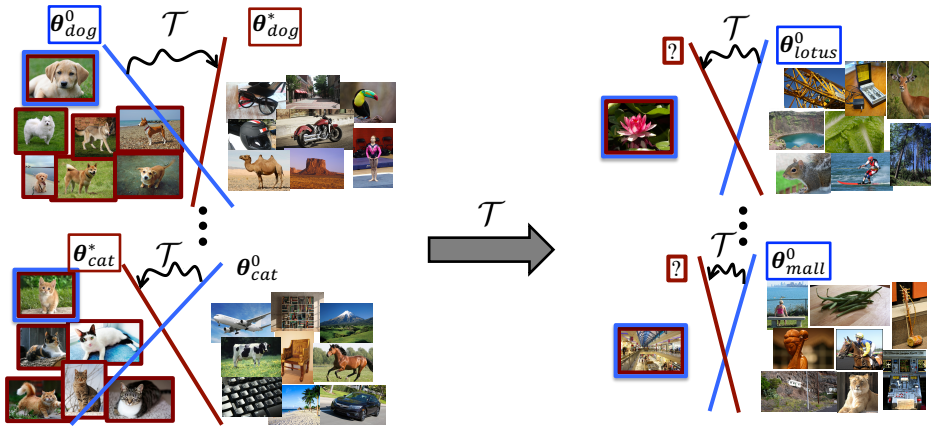


Figure 3.1: Our main hypothesis is that there exists a generic, category agnostic nonlinear transformation $\mathcal{T}$ from models $\boldsymbol{\theta}^0$ learned from few annotated samples (represented as blue) to the underlying models $\boldsymbol{\theta}^*$ learned from large sets of samples (represented as red). We estimate the transformation $\mathcal{T}$ by learning a meta-level regression network on a large collection of model pairs, *i.e.*, a model regression network. For a novel category/task (such as scene classification and fine-grained object recognition), we introduce the learned $\mathcal{T}$ to construct the task model and thus facilitate its generalization in the small sample size regime.

A desirable goal, then, is to find ways of automatically learning such a transformation. We achieve this by introducing a **meta-level network** that operates on the space of model parameters and learns on a large collection of model pairs, which we term as a *model regression network*. The meta-network explicitly regresses between the small-sample classifiers (as input) and their corresponding large-sample classifiers (as ground-truth) on a variety of known categories. The deep learning framework enables us to learn the transformation without imposing strong priors. Now, for a novel category/task, we *distill* the learned transformation to construct the task model and thus facilitate its generalization in the small sample size regime.

Recent progress in deep learning based object recognition shows that features extracted from deep CNNs that are trained on a large set of particular object categories exhibit attractive transferability [16, 91, 311, 427]. They could thus serve as universal feature extractors for novel categories/tasks. Our key insights then are that such generality would also hold on a model level and that it would be learnable in a similar fashion as on the feature level. This is also suggested by the duality perspective between the feature space and the classifier space [386]. Eventually, the transformation can be viewed to be imposed on features but parametrized in a model fashion.

From a complementary perspective, our approach could be seen as an alternative parametric way of doing model distillation that relies on the connection between different models [18, 49, 159]. In the terminology of model distillation [18, 159], we view a small-sample model as a "*student*" model and view its corresponding large-sample model as a "*teacher*" model. The student model is then transformed through our meta-network to mimic the behavior (*e.g.*, generalization ability) of its teacher model. Different from [18, 49, 159] that focus on the relationship between models of different complexity (*e.g.*, single and ensemble

models, and shallow and deep neural networks) on the same data, we deal with models of the same type but trained on data of different sample sizes. Moreover, our knowledge is *explicitly* represented by the meta-network, compared with using distillation loss as the *implicit knowledge* in [18, 49, 159].

Our contributions are three-fold. (1) We first show how to construct a training "model set" by generating a large collection of model pairs that are learned from small and large sample sets respectively on various categories. (2) We show how a model regression network, based on deep neural networks and this training model set, is learned and a generic transformation between these two types of models is identified by the regressor. (3) We finally show how our regression network is used to facilitate the recognition of novel categories from few samples, leading to significantly improved performance on a broad range of tasks, including domain adaptation, fine-grained recognition, action recognition, and scene classification.

3.2 Model Regression Networks

We are given a fixed, discriminative feature space $\mathcal{X}$ of dimensionality d, such as the current deep CNN features.[2] For an object category c of interest, we generate a model or classifier $h(\boldsymbol{x})$ that discriminates between its positive and negative instances $\boldsymbol{x} \in \mathcal{X}$. We consider, for example, the linear SVM classifier commonly used for object recognition tasks, which is a separating hyperplane in the feature space. The classifier $h(\cdot)$ can then be represented as a weight vector $\boldsymbol{\theta}$ belonging to the model parameter space Θ.

Let $\boldsymbol{\theta}^0$ indicate a classifier learned from few annotated samples *without any additional information*. Let $\boldsymbol{\theta}^*$ indicate the corresponding *underlying* classifier learned from a large set of annotated samples of the same category. Our goal is to generate $\boldsymbol{\theta}$ (or equivalently, $h(\cdot)$) that generalizes well from these few training examples, *i.e.*, to make $\boldsymbol{\theta}$ as close as to the desired $\boldsymbol{\theta}^*$. The key assumption is that there exists a generic, nonlinear transformation $\widetilde{\mathcal{T}} : \Theta \rightarrow \Theta$ for a broad range of categories, so that for $\boldsymbol{\theta}^0$ and $\boldsymbol{\theta}^*$ in any category c, we have $\boldsymbol{\theta}^* \approx \widetilde{\mathcal{T}}\left(\boldsymbol{\theta}^0\right)$. That is, there is a set of large-sample models and $\widetilde{\mathcal{T}}$ is the projection into that set (with $\boldsymbol{\theta}^*$ being a fixpoint of $\widetilde{\mathcal{T}}$). Once the transformation $\widetilde{\mathcal{T}}$ is available, we could easily improve the classifier generalization.

Inspired by recent progress in deep learning, it is possible to estimate this transformation $\widetilde{\mathcal{T}}$ from a large set of known categories. A straightforward approach then is to learn a regression function $\mathcal{T}$ parameterized by $\boldsymbol{w}$ based on a large collection of "annotated" model pairs $\left\{\left(\boldsymbol{\theta}_j^0, \boldsymbol{\theta}_j^*\right)\right\}_{j=1}^{J}$ from these categories. That is, $\boldsymbol{\theta}_j^* \approx \mathcal{T}\left(\boldsymbol{\theta}_j^0; \boldsymbol{w}\right)$ for any small-sample model $\boldsymbol{\theta}_j^0$ and its large-sample model $\boldsymbol{\theta}_j^*$ learned on the same category. We employ multi-layer neural networks as regressors, which are well-known to learn complex, nonlinear functions with minimal human design. By doing so, we avoid an explicit description of the space of transformations. We then use the obtained transformation in learning models for novel categories.

[2] Notation: Matrices are denoted by bold and italicized capital letters, vectors are denoted by bold and italicized lower-case letters, and transformation functions are denoted by italicized capital letters. For notational simplicity, $\boldsymbol{x}$ already includes a constant 1 as the last element and thus $\boldsymbol{\theta}$ includes the bias term.

3.2.1 Generation of Model Pairs

We start from large amounts of labeled data from a variety of categories, which are denoted as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{L}$. Here $\boldsymbol{x}_i \in \mathbb{R}^d$ is the ith data sample in the feature space $\mathcal{X}$, $y_i \in \{1, \ldots, C\}$ is the corresponding label, and C is the number of categories. Different from conventional recognition systems that directly learn from the data and label pairs, we learn on a model level. To this end, we produce a collection of model pairs $\left\{\left(\boldsymbol{\theta}_j^0, \boldsymbol{\theta}_j^*\right)\right\}_{j=1}^{J}$ as our *training model set* using the original training data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{L}$. Each model is generated as a binary classifier focused on separating a single category from all the remaining categories in a manner inspired by the one-vs.-all strategy in multi-class classification.

Specifically, for each category c, we first learn $\boldsymbol{\theta}^{c,*}$ from a large sample set. We treat $\boldsymbol{\theta}^{c,*}$ as the *ground-truth model*. Let the positive examples $\{\boldsymbol{x}_i^{c,pos}\}_{i=1}^{L_c}$ be all the data points of category c, where L_c is the total number of samples whose labels are c. We obtain negative examples $\{\boldsymbol{x}_i^{c,neg}\}_{i=1}^{M}$ by randomly sampling M data points from other categories not in category c. We train a binary SVM classifier $\boldsymbol{\theta}^{c,*}$ on the training set $\mathcal{P}^c = \{(\boldsymbol{x}_i^{c,pos}, +1)\}_{i=1}^{L_c} \cup \{(\boldsymbol{x}_i^{c,neg}, -1)\}_{i=1}^{M}$.

We now learn the small-sample model $\boldsymbol{\theta}^{c,0}$ for category c. Consistent with the few-shot scenario that consists of few positive examples, we randomly sample $N \ll L_c$ data points $\{\boldsymbol{x}_i^{c,pos}\}_{i=1}^{N}$ out of the L_c positive examples of category c. We train a binary SVM classifier $\boldsymbol{\theta}^{c,0}$ on the reduced training set $\mathcal{Q}^c = \{(\boldsymbol{x}_i^{c,pos}, +1)\}_{i=1}^{N} \cup \{(\boldsymbol{x}_i^{c,neg}, -1)\}_{i=1}^{M}$.

Note that we have many ways of choosing the small sample set for a given $\boldsymbol{\theta}^{c,*}$ to learn $\boldsymbol{\theta}^{c,0}$. This indicates that we could repeat the sampling procedure S times, leading to S small-sample models $\left\{\boldsymbol{\theta}_j^{c,0}\right\}_{j=1}^{S}$ learned from different small-sample sized training subset $\left\{\mathcal{Q}_j^c\right\}_{j=1}^{S}$ of $\mathcal{P}^c$. Since they correspond to the unique ground-truth model, we thus obtain a series of model pairs for category c as $\left\{\left(\boldsymbol{\theta}_j^{c,0}, \boldsymbol{\theta}^{c,*}\right)\right\}_{j=1}^{S}$. Including the learned model pairs from all the C categories, we generate the desired training model sets $\left\{\left(\boldsymbol{\theta}_j^0, \boldsymbol{\theta}_j^*\right)\right\}_{j=1}^{J}$, where $J = S \times C$. Due to subsampling, the size of the training model set could be potentially large, with many orders of magnitude larger than the number of categories.

3.2.2 Regression Network

Given the training model set $\left\{\left(\boldsymbol{\theta}_j^0, \boldsymbol{\theta}_j^*\right)\right\}_{j=1}^{J}$ with one to one model correspondence, we aim to learn a mapping: $\boldsymbol{\theta}^0 \rightarrow \boldsymbol{\theta}^*$. We parametrize the transformation as a regression function $\mathcal{T}\left(\boldsymbol{\theta}^0; \boldsymbol{w}\right)$, such that $\boldsymbol{\theta}^* \approx \mathcal{T}\left(\boldsymbol{\theta}^0; \boldsymbol{w}\right)$. We simply use the square of the Euclidean distance to quantify the quality of the approximation. For each model $\boldsymbol{\theta}_j^0$, we have the corresponding small sample set $\mathcal{Q}_j = \left\{\left(\boldsymbol{x}_i^j, y_i^j\right)\right\}_{i=1}^{M+N}$ used to learn the model as well. To make the regression more robust, we include the performance on these samples as an additional loss, which is standard in the transfer learning approaches with model parameter sharing [398, 418]. Our final loss function then is

$$L\left(\boldsymbol{w}\right) = \sum_{j=1}^{J} \left\{ \frac{1}{2} \left\| \boldsymbol{\theta}_j^* - \mathcal{T}\left(\boldsymbol{\theta}_j^0; \boldsymbol{w}\right) \right\|^2 + \lambda \sum_{i=1}^{M+N} \left[1 - y_i^j \left(\mathcal{T}\left(\boldsymbol{\theta}_j^0; \boldsymbol{w}\right)^T \boldsymbol{x}_i^j \right) \right]_+ \right\}. \tag{3.1}$$

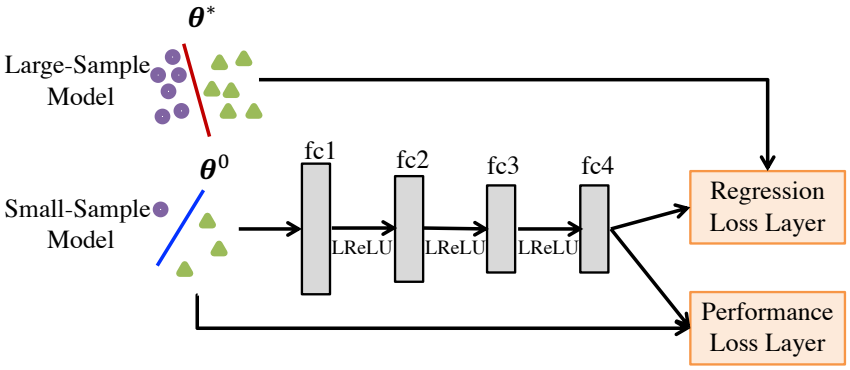


Figure 3.2: The architecture of our model regression network. Given a model $\boldsymbol{\theta}^0$ learned from few samples as input, it is passed though four fully-connected layers with leaky ReLU. On the loss layer, a model regression loss and a classification performance (*e.g.*, hinge) loss on the training data is minimized jointly.

The second term represents the data fitting on the training samples. Here, the performance loss is measured by a hinge loss, in which $[x]_+ = max(0, x)$, and it could be other types of losses such as logistic loss as well.

Consistent with recent work, we use a multi-layer feed-forward neural network as the regression function for its high capacity. As shown in Figure 3.2, our regression network consists of $F = 4$ fully-connected layers where the fth layer applies a nonlinear transformation G, which is an affine transformation followed by a nonlinear activation function. We use leaky ReLU. For the purpose of regression capacity, the number of units in the first two layers is larger than the dimensionality of the input classifier weight vectors. The desired transformation $\mathcal{T}$ is then represented as a series of transformations G layer by layer.

3.2.3 Implementation Details

For the feature space, consistent with recent work, we use the Caffe AlexNet CNN feature pre-trained on ILSVRC 2012 [91, 185, 212]. All the weights of the CNN are frozen to those learned on ILSVRC without fine-tuning on any other datasets. For each image, we extract the feature on the center 224×224 crop of the 256×256 resized image. It is a $d = 4{,}096$-dim feature vector $fc6$ taking from the penultimate hidden layer of the network, unless otherwise specified.

To generate the training model set, we use the ILSVRC 2012 training data set for purpose of reproducibility. There are $1{,}000$ object categories with 600 to $1{,}300$ images per category and 1.2 million images in total. We use Liblinear [101] to train linear SVM models $\boldsymbol{\theta}^0$ and $\boldsymbol{\theta}^*$. For each category, using all the positive images and randomly sampled negative images, we train $\boldsymbol{\theta}^*$ with the optimal SVM regularization parameter obtained by 10-fold cross-validation. We then randomly sample $N = 1, 2, \ldots, 9, 10, 15, 20, \ldots, 100$ positive images. For each N, we repeat random subsampling $S = 5$ times, and use different SVM regularization parameters from $10^{\{-2,-1,0,1,2\}}$ to train the SVM model $\boldsymbol{\theta}^0$ from few samples. These are essentially valid ways of doing "data augmentation" [212] for training the regression network, which mimic in practice how $\boldsymbol{\theta}^0$ changes. Hence, the number of the generated model pairs is 700 for each category, and the size of the training model set is

700,000. Finally, we randomly split the set with 685 model pairs as training and the remaining 15 pairs as validation per category.

We then use Caffe [185] to train our model regression network on the generated training model set and the corresponding training data set. The number of units from $fc1$ to $fc4$ are 6144, 5120, 4097, and 4097, respectively. We use 0.01 as the negative slope for leaky ReLU. λ is set to 1. We implement the loss function as two loss layers in Caffe, with one loss layer focusing on the model regression accuracy and the other focusing on the performance loss on the training data. We train the network using standard SGD and batch normalization [177].

3.2.4 Learning Target Models for Novel Categories

We now consider recognizing a novel category from a small labeled training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^K$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is a data sample and $y_i \in \{-1, 1\}$ is the corresponding label. As informative prior knowledge, we distill the obtained generic model transformation $\mathcal{T}$ into the target model $\boldsymbol{\theta}$, so that it has better generalization than the one produced only from the few training examples. We use a coarse-to-fine procedure that learns the target model in three steps: initialization, transformation, and refinement.

Initialization. In this first step, we directly learn the target model $\boldsymbol{\theta}^0$ on the small training sample set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^K$.

Transformation. Using $\boldsymbol{\theta}^0$ as input to our learned model regression network, after forward propagation, we obtain the output model $\mathcal{T}\left(\boldsymbol{\theta}^0; \boldsymbol{w}\right)$. This thus encodes the prior knowledge about $\boldsymbol{\theta}$ being preferable.

Refinement. We then introduce $\mathcal{T}\left(\boldsymbol{\theta}^0; \boldsymbol{w}\right)$ as biased regularization into the standard SVM max-margin formulation to retrain the model by minimizing

$$R\left(\boldsymbol{\theta}\right) = \frac{1}{2}\left\|\boldsymbol{\theta} - \mathcal{T}\left(\boldsymbol{\theta}^0; \boldsymbol{w}\right)\right\|^2 + \eta \sum_{i=1}^{K} \left[1 - y_i\left(\boldsymbol{\theta}^T \boldsymbol{x}_i\right)\right]_+. \tag{3.2}$$

Eqn. (3.2) is similar to the standard SVM formulation, with the only difference being the bias towards $\mathcal{T}\left(\boldsymbol{\theta}^0; \boldsymbol{w}\right)$ instead of 0. η is the regularization parameter used to control the trade-off between the regularization term and data fitting term. We thus obtain an intermediate solution with a decision boundary close to the regressed classifier while separating the labeled examples well.

3.3 Experimental Evaluation

In this section, we explore the use of our learned model regression network on a number of supervised learning tasks with limited data, including domain adaptation, fine-grained recognition, action recognition, and scene classification. We begin with a sanity check of the regression network for the 1,000 training categories on the ILSVRC validation data set. We then evaluate the network for one-shot domain adaptation and compare with state-of-the-art adaptation approaches. We further evaluate our approach for novel fine-grained, action, and scene categories. Finally, we present experimental results evaluating the impact of different feature spaces and model types.

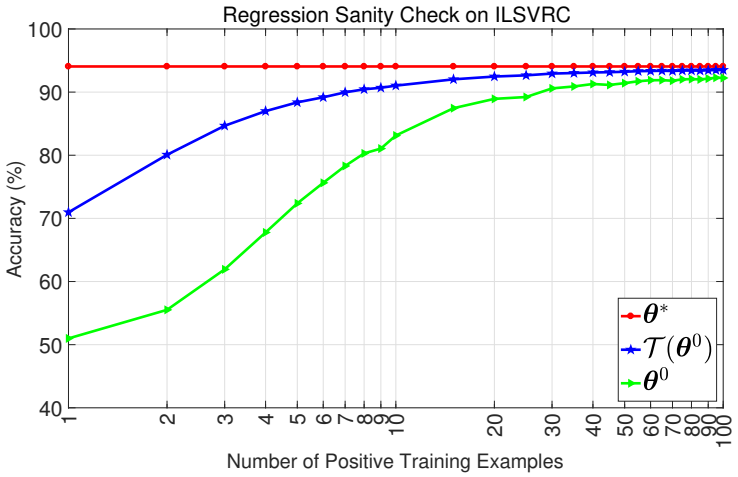


Figure 3.3: Performance sanity check of the model regression network by comparing small-sample models $\boldsymbol{\theta}^0$, large-sample models $\boldsymbol{\theta}^*$ (learned on thousands of examples), and regressed models $\mathcal{T}(\boldsymbol{\theta}^0)$ on the held-out ILSVRC validation set. X-axis: number of positive training examples. Y-axis: average binary classification accuracy. Our network effectively identifies a generic model transformation.

3.3.1 Sanity Check

Our model regression network is learned from the $1{,}000$ categories on the ILSVRC training data set. As a sanity check, the first question to answer is whether the learned transformation indeed improves generalization of the small-sample models for these categories. To answer this question, we evaluate the models on the held-out ILSVRC validation data set, which contains the same $1{,}000$ categories with 50 images per category and has no overlap with the ILSVRC training data.

Consistent with the way the models are generated, we evaluate them in a binary classification scenario. For each category, we construct a test set consisting of all these 50 positive images and 50 randomly sampled negative images from other categories. We compare the three types of models: small-sample models $\boldsymbol{\theta}^0$, large-sample models $\boldsymbol{\theta}^*$ (as ground-truth), and regressed models $\mathcal{T}(\boldsymbol{\theta}^0)$ (without the refinement step). We evaluate how performance varies with the number of positive training examples N when used to learn $\boldsymbol{\theta}^0$. We average the classification accuracy over the models corresponding to the same N but with different sampled training data and SVM regularization parameters. Figure 3.3 summarizes the average performance over the $1{,}000$ categories.

As expected, Figure 3.3 shows that $\mathcal{T}(\boldsymbol{\theta}^0)$ significantly improves the generalization of $\boldsymbol{\theta}^0$. In the one-shot learning case, there is a notable 20% performance improvement of $\mathcal{T}(\boldsymbol{\theta}^0)$ over $\boldsymbol{\theta}^0$, whose performance is only a little bit higher than chance (50% for binary classification). With increased number of training examples, the performance of $\mathcal{T}(\boldsymbol{\theta}^0)$ gradually converges to that of $\boldsymbol{\theta}^*$ trained on thousands of examples. This verifies the existence of a generic transformation from small-sample to large-sample models for these $1{,}000$ categories, which is effectively identified by our model regression network. In the following experiments, we will show that the learned transformation applies to other novel categories as well.

Source Prior Knowledge Type	Method	Acc (%)
NA	SVM (target only) [164]	62.28
Data	SVM (source only) [164]	53.51
	SVM (source and target) [164]	56.68
Feature	GFK [127]	65.16
	SA [110]	59.30
	Daumé III [81]	59.21
	MMDT [163]	59.21
Model Parameter	PMT [13]	66.30
	Late Fusion (Max) [164]	59.59
	Late Fusion (Lin. Int. Avg) [164]	60.64
Joint	Fine-tuning [164]	61.13
Model Transformation	Model Regression Network (Ours)	**68.47**

Table 3.1: Performance comparison between our model transformation with state-of-the-art approaches that adapt other types of prior knowledge gained on the ILSVRC source domain in manners of data, feature, model parameter, and joint fine-tuning for one-shot learning on the Webcam domain of the Office dataset.

3.3.2 One-Shot Adaptation

Our approach can be viewed as transferring certain prior knowledge gained from the source domain (ILSVRC) to new tasks. It is thus interesting to compare different types of prior knowledge, including those on data, feature, and model parameter levels. To this end, we provide a comprehensive evaluation in the scenario of domain adaptation, in which the target images come from the same set of source categories but are drawn from a different distribution. Due to the common categories between source and target domains, this experimental setup allows us to best identify the possible shared domain structure and compare with state-of-the-art adaptation approaches without learning additional category correspondence, which turns to be another difficult problem.

Datasets and tasks. We evaluate on the Office dataset [324], a standard domain adaptation benchmark for multi-class object recognition. The Office dataset is a collection of $4{,}652$ images from three distinct domains: Amazon, DSLR, and Webcam. We use Webcam as the target domain since it was shown to be the most challenging shifted domain [164]. Of the 31 categories in the dataset, 16 overlap with the categories presented in the $1{,}000$-category ILSVRC. We focus on these common classes as our target (*i.e.*, 16-way classification), as is customary in [164]. Following a similar experimental setup in [164], 1 labeled training and 10 test images per category are randomly selected on Webcam. We report average multi-class accuracy over 20 random train/test splits in Table 3.1.

Baselines. In addition to the SVM (target only) baseline that directly trains SVM classifiers on the target data, we compare against four other types of baselines that transfer prior knowledge on the ILSVRC source domain gained in manners of data, feature, model parameters, and joint fine-tuning.

- **(Type I) Data level.** SVM classifiers trained on only source data and both source and target data, respectively.
- **(Type II) Feature level.** Geodesic flow kernel (GFK) [127], Daumé III [81], subspace alignment (SA) [110], and max-margin domain transforms (MMDT) [163], which seek common feature spaces using learned feature embedding, augmentation, or transformation.
- **(Type III) Model parameter level.** Projective model transfer (PMT) [13] and late fusion [164], which adapt the parameters of the pre-trained source classifier to construct the target classifier.
- **(Type IV) Joint level.** Fine-tune the weights of the pre-trained CNN on the 16-way target classification task. These results are reported from [164].

Table 3.1 shows that our model transformation provides an alternative, competitive way to encode the shared structure and prior knowledge. It is on par with or outperforms other types of prior knowledge and adaption approaches. Notably, ours achieves significantly better performance than fine-tuning, the standard transfer strategy for CNNs, in this one-shot learning scenario. Fine-tuning requires a considerable amount of labeled target data and actually reduces performance in the very sparse label regime.

3.3.3 Learning Novel Categories

We now evaluate whether our learned model regression network facilitates the recognition of novel categories from few samples. For multi-class classification on the target datasets, we test how performance varies with the number of training samples per category. Following the standard practice, we train linear SVMs in a one-vs.-all fashion with default settings in Liblinear [101]. After obtaining the regressed models, we then incorporate them to retrain each one-vs.-all classifier.

Datasets and tasks. We evaluate on standard benchmark datasets for fine-grained recognition: Caltech-UCSD Birds (CUB) 200-2011 [389] and Oxford 102 Flowers [279], for action recognition (compositional semantic recognition): Stanford-40 actions [425], and for scene classification: MIT-67 [375]. These datasets are widely used for evaluating CNN representations [15], and we consider their diversity and coverage of novel categories. We follow the standard experimental setup (*e.g.*, the train/test splits) for these datasets. In our experiments, due to the lack of published protocols for small-sample learning, we randomly generate the small-sample version of training images as shown in Figure 3.4 and use all the same test images for testing.

Fine-grained recognition datasets. Caltech-UCSD Birds (CUB) $200-2011$ [389] contains $11{,}788$ images of 200 bird species (mostly North American). $5{,}994$ images are used for training (29 or 30 images per class) and $5{,}794$ images are used for testing. Bird bounding boxes, 15 part landmarks, 312 binary attributes, and boundary segmentation are available for this dataset. We only use the bounding box annotation during training and testing. Oxford 102 Flowers [279] contains 102 flower classes and each class consists of between 40 and 258 images. 10 images per class are used as training data and the rest are used as test data. We do not use the provided image segmentation for this dataset. The subtle difference across different subclasses requires a fine-detailed feature representation, which makes fine-grained recognition a good test of whether a generic representation can capture these subtle details.

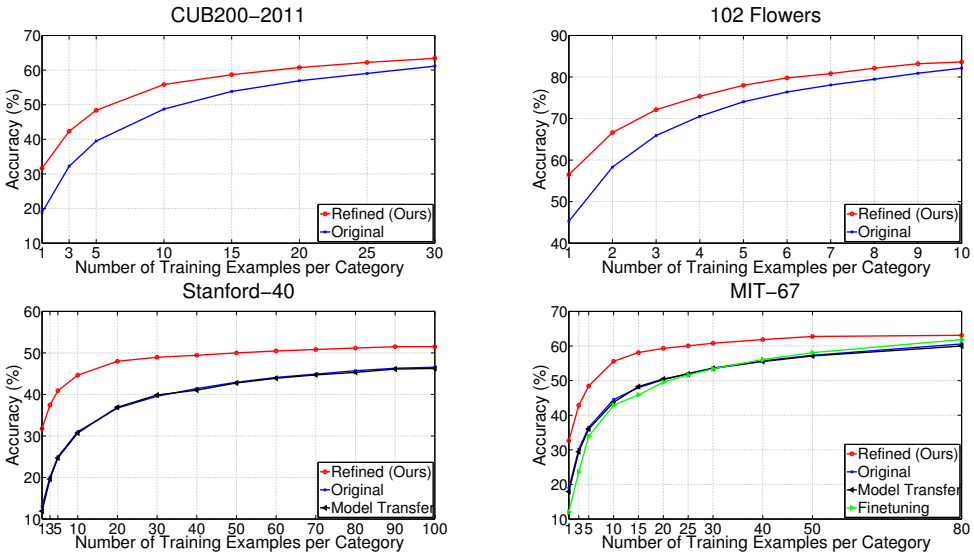


Figure 3.4: Performance comparisons between models learned from few samples and models refined by our model regression network for fine-grained recognition, action recognition, and scene classification on four benchmark datasets. For completeness, we also include additional baselines of transfer learning with model parameter sharing and CNN fine-tuning on certain datasets. The AlexNet CNN is used as the feature space. X-axis: number of training examples per class. Y-axis: average multi-class classification accuracy. Since they benefit from the learned generic model transformation, ours significantly outperform all the baselines for small sample learning.

Action recognition datasets. Stanford-40 [425] contains 9,532 images of humans performing 40 actions, with between 180 and 300 images per action class. We follow the standard split for this dataset: 100 images per class are used as training data and the rest are used as test data. The compositional recognition tasks are challenging category tasks, which include classes where the type of interactions between objects is the key indicator. Hence, these tasks require more sophistication to recognize than the other category recognition tasks [16].

Scene classification datasets. MIT-67 [375] contains 15,620 images spanning 67 indoor scene classes. The provided split for this dataset consists of 80 training and 20 test images per class. Having a significant statistics difference from the ILSVRC dataset, indoor scenes tend to vary a lot in term of composition and are better characterized by the objects which they contain. This makes MIT-67 an interesting and challenging test case for feature representation.

Baselines. Due to the CNN training procedure, the original models directly learned from target samples can be viewed as transfer learning with feature sharing. We also include the transfer learning baseline with model parameter sharing on Stanford-40 and MIT-67, which transfers the 1,000 ILSVRC category models using [373]. Moreover, we report an additional CNN fine-tuning baseline on MIT-67, which is the best fine-tuning result we have achieved following [155].

Figure 3.4 summarizes the average performance over 10 random splits on these datasets.

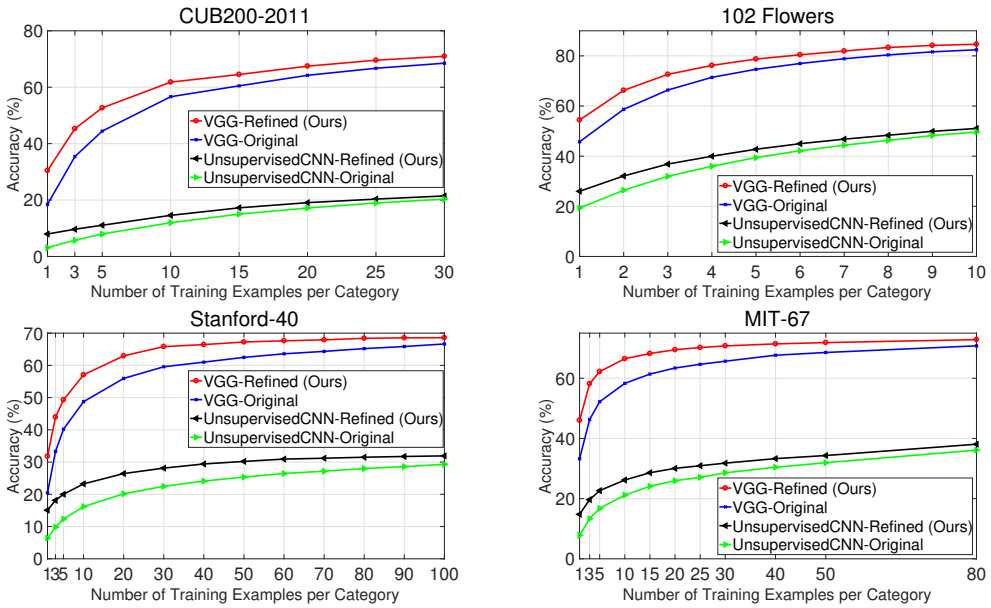


Figure 3.5: Feature space evaluation between models learned from few samples and models refined by our model regression network on these four benchmark datasets. The stronger VGG CNN [347], pre-trained on ILSVRC, and the unsupervised CNN [393], pre-trained on YouTube, are used as the feature space, respectively. Ours show consistent performance improvements over the original models for small sample learning in different feature spaces.

The performance of the model transfer is similar to the original models learned from few samples due to the dissimilarity between source and target tasks. In our case of limited target data, the standard fine-tuning approach leads to degraded performance due to overfitting. The models refined by our regression network, however, significantly outperform them for a broad range of novel categories. Our approach has particularly large performance boosts in one-shot learning scenarios. For example, there is a nearly 15% boost on MIT-67.

3.3.4 Evaluation of Different Feature Spaces

In the previous experiments, we used the AlexNet CNN as the feature. To test the robustness of our model regression network to the choice of the feature space, here we evaluate two additional features: the more powerful VGG19 CNN [347] $fc7$, pre-trained on ILSVRC 2012, and the unsupervised CNN [393] $fc6$, pre-trained on YouTube videos. We keep the other design choices the same (*e.g.*, the way of generating the training model set and the regression network structure). In a similar way as before, we train our network and evaluate the recognition performance on the target tasks with few samples. Figure 3.5 validates the benefit of our approach in different feature space settings. Importantly, it shows that the data used to estimate the model transformation (ILSVRC) is not necessarily the same as the data used to learn the feature representation (YouTube).

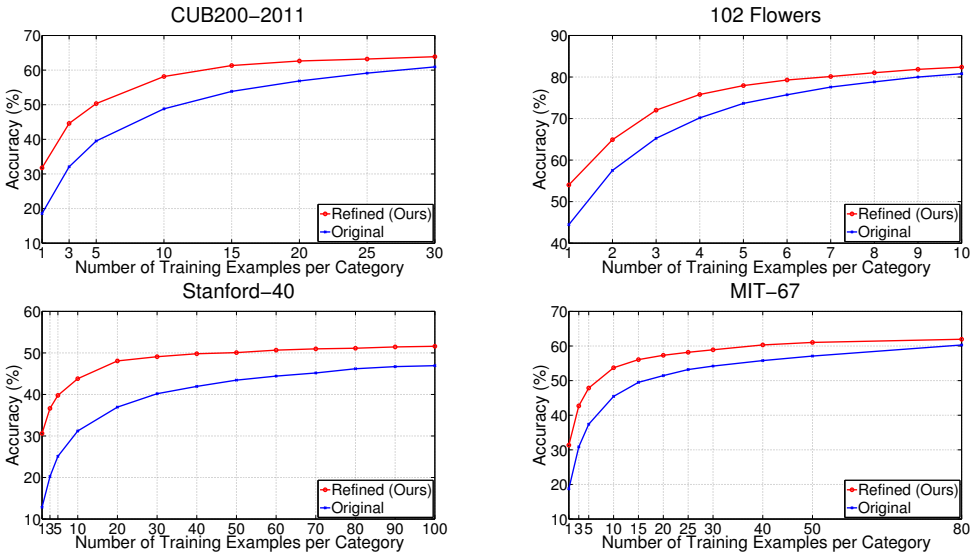


Figure 3.6: Model type evaluation between models learned from few samples and models refined by our model regression network on these four benchmark datasets. We evaluate the logistic regression as the model of interest. The robust performance shows generic transformations for different types of models.

3.3.5 Evaluation of Different Types of Classification Models

In the previous experiments, we focused on SVM classifiers. In fact, the models do not need to come from max-margin classifiers and could be other set of weights learned in different fashions. To verify this, we test a widely used alternative classifier, logistic regression, and keep the other design choices the same (*e.g.*, the way of generating the training model set and the regression network structure). Naturally, we change the hinge loss to the logistic loss. In a similar way as before, we train our network and evaluate the recognition performance on the target tasks with few samples as shown in Figure 3.6. Combining with Figure 3.4, the logistic regression demonstrates comparable performance to SVM, and the refined logistic regression classifiers generalize better as well.

3.4 Data-Level or Model-Level Transformation?: A Graphical Illustration

While our approach focuses on a model-level transformation, a contemporary line of work generates additional examples by transforming existing examples through a regression neural network and then learns recognition models using the augmented dataset [86, 144, 247]. One interesting question thus arises: what is the connection and distinction between a data-level transformation and a model-level transformation? Inspired by *category theory and commutative diagram* [12], this issue can be graphically understood by Figure 3.7.

In Figure 3.7, the top left node A represents a small sample set $\mathcal{Q}^0 = \{\boldsymbol{x}_i\}_{i=1}^M$, and the bottom left node B represents its corresponding large sample set $\mathcal{P}^* = \{\boldsymbol{x}_i\}_{i=1}^N$. The top

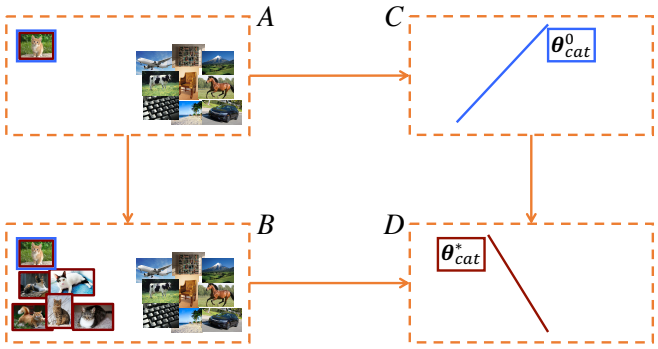


Figure 3.7: Graphical illustration of data-level and model-level transformations. To generate a large-sample model $\boldsymbol{\theta}^*$ (node D) from a small-sample set $\mathcal{Q}^0$ (node A), we have two commutative paths: the data-level transformation path $A \rightarrow B \rightarrow D$ and the model-level transformation path $A \rightarrow C \rightarrow D$.

right node C represents the small-sample model $\boldsymbol{\theta}^0$, and the bottom right node D represents the large-sample model $\boldsymbol{\theta}^*$. There are two paths to generate $\boldsymbol{\theta}^*$ from $\mathcal{Q}^0$. The conventional path $A \rightarrow B \rightarrow D$ first transforms the few examples in $\mathcal{Q}^0$ to produce more examples and obtain $\mathcal{P}^*$, and then learns $\boldsymbol{\theta}^*$ using $\mathcal{P}^*$. Our path $A \rightarrow C \rightarrow D$, however, first directly learns $\boldsymbol{\theta}^0$ using given $\mathcal{Q}^0$, and then transforms $\boldsymbol{\theta}^0$ to $\boldsymbol{\theta}^*$. Intuitively, these two paths should commute and lead to the same $\boldsymbol{\theta}^*$, as suggested by the duality perspective between the feature space and the classifier space [386]. In this view, our approach is essentially a valid way of doing data augmentation but parametrized in a model fashion.

From a practical perspective, however, each of these two paths has its own advantages and disadvantages, depending on specific tasks. On the one hand, our formulation leads to an easier learning problem, since it is better regularized. More precisely, for category c, different small-sample models $\boldsymbol{\theta}^0$ correspond to a unique large-sample model $\boldsymbol{\theta}^*$ (*i.e.*, learning a many-to-one transformation). By contrast, if directly learning a transformation on the data level, one input example might correspond to potentially many output examples (*i.e.*, learning a many-to-many transformation). In addition, our approach leads to test time efficiency, since it directly generates the desired recognition model instead of intermediate samples. On the other hand, learning a model-level transformation restricts the small-sample and large-sample models to belong to the same type of model, since they are supposed to reside in the same model parameter space. By contrast, learning a data-level transformation might allow us to learn a higher-capacity model with augmented data for potential better performance.

In a broad sense, the commutative diagram in Figure 3.7 not only applies to the small-sample learning scenario, but it also has implication for other recognition problems. And our model regression networks have been successfully used in [166, 168, 261]. For instance, one might morph between chair recognition models of two different viewpoints[3], or change a raw tomato recognition model to a cooked tomato recognition model [180], or compose classifiers of known visual concepts for unseen combinations of concepts [261], or map from free-hand sketch space to the space of photo classifiers [166], or predict a category's instance segmentation parameters as a function of its bounding box detection parameters [168], by

[3]Inspired by a conversation with Andrew Zisserman.

directly leveraging the model-level transformation.

3.5 Revisiting Model Transformation and Its Properties

The conventional few-shot learning approaches typically view a recognition model as static and they focus on discovering the relationship among models that are learned from large amounts of examples [103, 232, 317]. Our approach, however, has analyzed the difference between a small-sample recognition model and the underlying large-sample model. We have explored whether it is possible and how to estimate a large-sample model based on its corresponding small-sample model. Apparently, the vanilla regularization techniques used in statistical modeling are insufficient to tackle this challenging problem.

To make this point further, our approach suggests a way to address the small-sample learning problem through analyzing the *model dynamics*. While the existing work typically focuses on the final learning performance, the entire learning history implicitly contains abundant useful information for small-sample learning. That is, how does a recognition model $h\left(\boldsymbol{x};\boldsymbol{\theta}\right)$ change during the learning process when gradually having access to additional examples?

To address this issue, we make an analogy between the recognition model and a physical system as suggested in [331]. Let us consider training a continuous time physical system that performs potentially useful computations through its deterministic or stochastic dynamics. Given a state of sensory information (current and past inputs), the system is moving towards configurations that better explain the observed sensory data (equilibrium) [331]. We can think of the system's configuration as an explanation (or interpretation) for the observed sensory data. In the case of our recognition model, this means that the parameters of the model gradually move towards $\boldsymbol{\theta}^*$ that are more probable, given the sensory input (*i.e.*, additional examples) and according to the current "model of the world" associated with the parameters $\boldsymbol{\theta}^0$ of the model.

We cast estimation of the model dynamics as a learning problem, and exploit structures in the parameter space by leveraging supervised deep learning techniques. The model dynamics are explicitly represented by our meta-learner $\mathcal{T}$. For a novel task m, we predict the parameters of its recognition model from both few annotated examples and the meta-learner that encodes the parameter structures from previous recognition tasks, replacing the static model with a dynamic model $h_m\left(\boldsymbol{x};\mathcal{T}\left(\boldsymbol{\theta}_m^0,\boldsymbol{w}\right)\right)$. Hence, the learning occurs at two different time scales: rapid learning within tasks and more gradual, meta learning across many different tasks for self-referential learning and self-improvement [336]. The latter learning phase explicitly guides the learning process of the small-sample model.

In conventional statistical learning, we have a particular function of interest, whose behavior is constrained through a set of function evaluations using example images and their labels. In our setting, the examples are themselves recognition task instances, which means generalization corresponds to the ability to transfer knowledge between different tasks. Hence, our approach is different from the conventional approach of characterizing properties of tasks analytically and using these analytical insights to design learning algorithms by hand, *e.g.*, enforcing certain regularization for small-sample recognition.

Moreover, the model transformation has some interesting properties, which we discuss below.

Method	$\boldsymbol{\theta}^0$	MLP	ResNet1	ResNet2	ResNet3	ResNet4	$\boldsymbol{\theta}^*$
Acc (%)	77.64	94.85	90.62	95.88	96.62	97.68	98.60

Table 3.2: Performance comparisons of binary classification between different types of meta-learners (*i.e.*, model regression networks) for one-shot learning on 10 randomly sampled categories from ILSVRC. Our original meta-network is a 4-layer perceptron (MLP) and we now use ResNets with 1, 2, 3, and 4 residual blocks, respectively. The performance improvement shows the importance of identity regularization and residual learning.

Identity Regularization and Residual Learning. In our approach, we assumed that the transformation $\mathcal{T}$ was independent of the sample size whereas, in general, one would envision that $\mathcal{T}$ would change when the number of samples increases dramatically all the way to $\mathcal{T}$ = identity for very large training sample sets. This suggests the importance of the identity mapping in the model transformation. Also, we are actually more interested in estimating the change of the model parameter $\Delta\boldsymbol{\theta}$ instead of $\boldsymbol{\theta}$. Inspired by these insights and the recent work on residual learning [149,154], we instantiate the meta-learner as a residual network that learns the transformation graduality while explicitly maintaining its identity property.

Experimental analysis. We now use the state-of-the-art ResNet152 features [154]. We randomly sample 10 categories from ILSVRC. In a similar binary classification setup as before, we regress from 1-shot SVM $\boldsymbol{\theta}^0$ to $\boldsymbol{\theta}^*$ using meta-level ResNets [149, 154] with 1, 2, 3, and 4 residual blocks, respectively. Table 3.2 summarizes the results. The boosted recognition performance over our original version of regression networks (*i.e.*, a multi-layer perceptron) shows the importance of identity regularization and residual learning. In addition, the performance improves with the increased network depth.

Sample-Size Dependency. One limitation of the current few-shot learning approaches is that they typically cannot perform consistently well across datasets of different sample sizes [144]. A straightforward solution is to learn separate regression networks that address different small-sample models. We evaluate this simple approach as follows.

Method	1-shot		2-shot		4-shot	
	$\boldsymbol{\theta}_1^0$	$\mathcal{T}(\boldsymbol{\theta}_1^0)$	$\boldsymbol{\theta}_2^0$	$\mathcal{T}(\boldsymbol{\theta}_2^0)$	$\boldsymbol{\theta}_4^0$	$\mathcal{T}(\boldsymbol{\theta}_4^0)$
Acc (%)	18.87	27.72	30.01	36.44	39.85	45.62

Table 3.3: Performance comparisons of multi-class classification when learning separate regression networks that address different small-sample models on the SUN-397 dataset [414]. We regress 1-shot, 2-shot, and 4-shot softmax classifiers to the large-sample models using 3 different ResNets (each with 1 residual block), respectively. The performance improvement shows the effectiveness of a sample-size dependent transformation.

Experimental analysis. In a multi-class classification setup, we generate the 1-shot, 2-shot, and 4-shot softmax classifiers on the SUN-397 training dataset [414], respectively. We regress these 3 types of small-sample models to the large-sample models using 3 different ResNets (each with 1 residual block), respectively. Table 3.3 summarizes the results on the

validation dataset. In addition, when we test the learned transformation on the MIT-67 dataset, the classification performance of its 1-shot model improves from 39.91% to 46.04%. These results show the effectiveness of a sample size dependent transformation.

Based on these properties, in the next chapter we will show how to design and learn the model regression network in a more principled manner and how to apply it to address learning novel categories on data of different sample sizes.

Chapter 4

Learning to Model the Tail by Capturing Model Dynamics

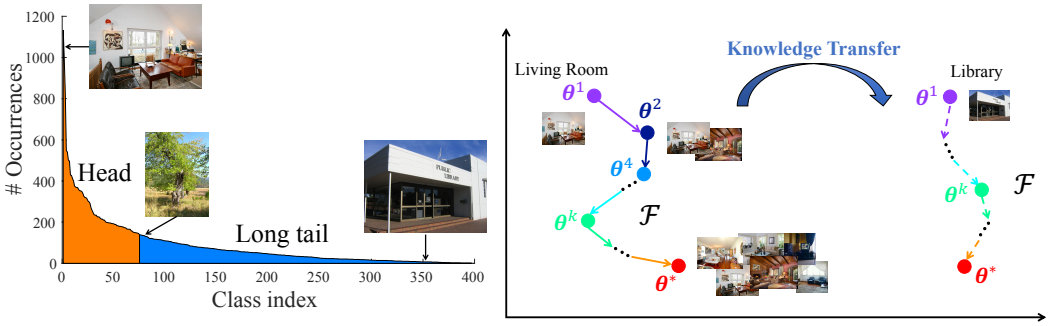
4.1 Motivation

In this chapter, we extend our learning to learn approach to address a more general, realistic long-tail recognition problem. State-of-the-art CNN models [154,212,347,357] are typically learned with *artificially balanced* datasets [242,321,436], in which objects of different classes have approximately evenly distributed, very large number of human-annotated images. In real-world applications, however, visual phenomena follow a long-tailed distribution as shown in Figure 4.1, in which the number of training examples per class varies significantly from hundreds or thousands for head classes to as few as one for tail classes [385,439,440].

Minimizing the skewed distribution by collecting more tail examples is a notoriously difficult task when constructing datasets [100,209,242,385]. Even those datasets that are balanced along one dimension still tend to be imbalanced in others [285]; *e.g.*, balanced scene datasets still contain long-tail sets of objects [414] or scene subclasses [439]. This *intrinsic* long-tail property poses a multitude of open challenges for recognition in the wild [31], since the models will be largely dominated by those few head classes while degraded for many other tail classes. Rebalancing training data [343,435] is the most widespread state-of-the-art solution, but this is *heuristic and suboptimal* — it merely generates redundant data through over-sampling or loses critical information through under-sampling.

An attractive alternative is to *transfer* knowledge from data-rich head classes to data-poor tail classes. While transfer learning from a source to target task is a well studied problem [288,427], by far the most common approach is to fine-tune a model pre-trained on the source task [162]. In the long-tailed setting, this fails to provide any noticeable improvement since pre-training on the head is quite similar to training on the unbalanced long-tailed dataset (which is dominated by the head) [385].

Inspired by the recent work on meta-learning [10,236,310,348,394,395], we instead transfer meta-level knowledge about *learning to learn* from the head classes. Specifically, we make use of the approach in Chapter 3, which describes a method for learning from small datasets (the "few-shot" learning problem) through estimating a generic model transformation. To do so, Chapter 3 learns a meta-level network that operates on the space of model parameters, which is specifically trained to *regress* many-shot model parameters (trained on large



(a) Long-tail distribution on the SUN-397 dataset.

(b) Knowledge transfer from head to tail classes.

Figure 4.1: Head-to-tail knowledge transfer in model space for long-tail recognition. Figure 4.1a shows the number of examples by scene class on SUN-397 [414], a representative dataset that follows an intrinsic long-tailed distribution. In Figure 4.1b, from the data-rich head classes (*e.g.*, living rooms), we introduce a meta-learner $\mathcal{F}$ to learn the model dynamics — a series of transformations (denoted as solid lines) that represents how few k-shot models $\boldsymbol{\theta}^k$ start from $\boldsymbol{\theta}^1$ and gradually evolve to the underlying many-shot models $\boldsymbol{\theta}^*$ trained from large sets of samples. The model parameters $\boldsymbol{\theta}$ are visualized as points in the "dual" model (parameter) space. We leverage the model dynamics as prior knowledge to facilitate recognizing tail classes (*e.g.*, libraries) by hallucinating their model evolution trajectories (denoted as dashed lines).

datasets) from few-shot model parameters (trained on small datasets). Our meta-level regressor, which we call *MetaModelNet*, is trained on classes from the head of the distribution and then applied to those from the tail. As an illustrative example in Figure 4.1, consider learning scene classifiers on a long-tailed dataset with many living-rooms but few outside libraries. We learn both many-shot and few-shot living-room models (by subsampling the training data as needed), and train a regressor that maps between the two. We can then apply the regressor on few-shot models of libraries learned from the tail.

The above description suggests that we need to split up a long-tailed training set into a distinct set of source classes (the head) and target classes (the tail). This is most naturally done by thresholding the number of training examples per class. But what is the correct threshold? A high threshold might result in a meta-network that simply acts as an identity function, returning the input set of model parameters. This certainly would not be useful to apply on few-shot models. Similarly, a low threshold may not be useful when regressing from many-shot models. Instead, we propose a "continuous" strategy that builds multiple regressors across a (logarithmic) *range* of thresholds (*e.g.*, 1-shot, 2-shot, 4-shot regressors, *etc.*), corresponding to different head-tail splits. Importantly, these regressors can be efficiently implemented with a *single, chained* MetaModelNet that is naturally regularized with residual connections, such that the 2-shot regressor need only predict model parameters that are fed into the 4-shot regressor, and so on (until the many-shot regressor that defaults to the identity). By doing so, MetaModelNet encodes a *trajectory* over the space of model parameters that captures their *evolution* with increasing sample sizes, as shown in Figure 4.1b. Interestingly, such a network is naturally trained in a *progressive* manner from the head towards the tail, effectively capturing the **gradual dynamics** of transferring meta-knowledge

from data-rich to data-poor regimes.

It is natural to ask what kind of dynamics are learned by MetaModelNet — how can one consistently predict how model parameters will change with more training data? We posit that the network learns to capture implicit **data augmentation** — for example, given a 1-shot model trained with a single image, the network may learn to implicitly add rotations of that single image. But rather than explicitly creating data, MetaModelNet predicts their impact on the learned model parameters. Interestingly, past work tends to apply the same augmentation strategies across all input classes. But perhaps different classes should be augmented in different ways — *e.g.*, churches maybe viewed from consistent viewpoints and should not be augmented with out-of-plane rotations. MetaModelNet learns *class-specific* transformations that are smooth across the space of models — *e.g.*, classes with similar model parameters tend to transform in similar ways (see Figure 4.1b and Figure 4.4 for more details).

Our contributions are three-fold. (1) We analyze the *dynamics of how model parameters evolve* when given access to more training examples. (2) We show that a single meta-network, based on deep residual learning, can learn to accurately predict such dynamics. (3) We train such a meta-network on long-tailed datasets through a recursive approach that gradually transfers meta-knowledge learned from the head to the tail, significantly improving long-tail recognition on a broad range of tasks.

4.2 Long-Tail Recognition

A widespread yet suboptimal strategy is to resample and rebalance training data in the presence of the long tail, either by sampling examples from the rare classes more frequently [343, 435], or reducing the number of examples from the common classes [148]. The former generates redundancy and quickly runs into the problem of over-fitting to the rare classes, whereas the latter loses critical information contained within the large sample sets. An alternative practice is to introduce additional weights for different classes, which, however, makes optimization of the models very difficult in the large-scale recognition scenarios [170].

Our underlying assumption that model parameters across different classes share similar dynamics is somewhat common in meta-learning [10,310,395]. While [10,310] consider the dynamics during stochastic gradient descent (SGD) optimization, we address the dynamics as more training data is gradually made available. In particular, the model regression network in Chapter 3 empirically shows a generic nonlinear transformation from small-sample to large-sample models for different types of feature spaces and classifier models. We extend [395] for long-tail recognition by introducing a single network that can model transformations across different sample sizes. To train such a network, we introduce recursive algorithms for head-to-tail transfer learning and architectural modifications based on deep residual networks (that ensure that transformations of large-sample models default to the identity).

Long-tail recognition is relevant to few-shot learning. However, the existing few-shot learning techniques [41, 310, 351, 388] are typically developed for a fixed set of few-shot tasks, in which each class has the same, fixed number of training samples. They appear difficult to generalize to novel tasks with a wide range of sample sizes, the hallmark of long-tail recognition.

4.3 Head-to-Tail Meta-Knowledge Transfer

Given a long-tail recognition task of interest and a base recognition model such as a deep CNN, our goal is to transfer knowledge from the data-rich head to the data-poor tail classes. As shown in Figure 4.1, knowledge is represented as trajectories in model space that capture the evolution of parameters with more and more training examples. We train a meta-learner (MetaModelNet) to learn such model dynamics from head classes, and then "hallucinate" the evolution of parameters for the tail classes. To simplify exposition, we first describe the approach for a fixed split of our training dataset into a head and tail. We then generalize the approach to multiple splits.

4.3.1 Fixed-Size Model Transformations

Let us write H_t for the "head" training set of $(\boldsymbol{x}, y)$ data-label pairs constructed by assembling those classes for which there exist more than t training examples. We will use H_t to learn a meta-network thats maps few-shot model parameters to many-shot parameters, and then apply this network on few-shot models from the tail classes. To do so, we closely follow the model regression framework in Chapter 3, but introduce notation that will be useful later.[1] Let us write a base learner as $g(\boldsymbol{x}; \boldsymbol{\theta})$ as a feedforward function $g(\cdot)$ that processes an input sample $\boldsymbol{x}$ given parameters $\boldsymbol{\theta}$. We first learn a set of "optimal" model parameters $\boldsymbol{\theta}^*$ by tuning g on H_t with a standard loss function. We also learn few-shot models by randomly sampling a smaller fixed number of examples per class from H_t. We then train a meta-network $\mathcal{F}(\cdot)$ to map or regress the few-shot parameters to $\boldsymbol{\theta}^*$.

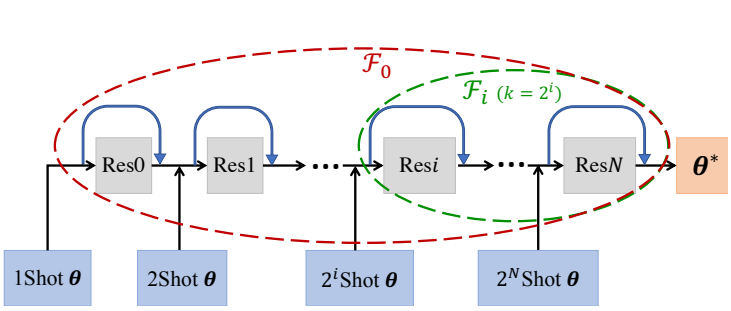
Parameters. In principle, $\mathcal{F}(\cdot)$ applies to model parameters from multiple CNN layers. Directly regressing parameters from all layers is, however, difficult to do because of the larger number of parameters. For example, recent similar methods for meta-learning tend to restrict themselves to smaller toy networks [10, 310]. For now, we focus on parameters from the last fully-connected layer for a single class — *e.g.*, $\boldsymbol{\theta} \in \mathbb{R}^{4096}$ for an AlexNet architecture. This allows us to learn regressors that are shared across classes (as in Chapter 3), and so can be applied to any individual test class. This is particularly helpful in the long-tailed setting, where the number of classes in the tail tends to outnumber the head. Later we will show that (nonlinear) fine-tuning of the "entire network" during head-to-tail transfer can further improve performance.

Loss function. The meta-network $\mathcal{F}(\cdot)$ is itself parameterized with weights $\boldsymbol{w}$. The objective function for each class is:

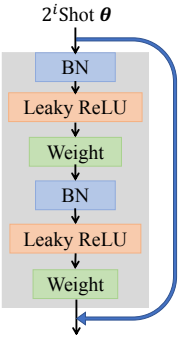
$$\sum_{\boldsymbol{\theta} \in k\text{Shot}(H_t)} \Big\{ ||\mathcal{F}(\boldsymbol{\theta}; \boldsymbol{w}) - \boldsymbol{\theta}^*||^2 + \lambda \sum_{(\boldsymbol{x}, y) \in H_t} \text{loss}\Big(g\big(\boldsymbol{x}; \mathcal{F}(\boldsymbol{\theta}; \boldsymbol{w})\big), y\Big) \Big\}. \tag{4.1}$$

The final loss is averaged over all the head classes and minimized with respect to $\boldsymbol{w}$. Here, $k\text{Shot}(H_t)$ is the set of few-shot models learned by subsampling k examples per class from H_t, and loss refers to the performance loss used to train the base network (*e.g.*, cross-entropy). $\lambda > 0$ is the regularization parameter used to control the trade-off between the two terms. In Chapter 3, we found that the performance loss was useful to learn regressors that maintained high accuracy on the base task. This formulation can be viewed as an extension to

[1] For notation simplicity, this chapter uses a slight modification of the notation in Chapter 3. For example, we use $\boldsymbol{\theta}$ to denote the few-shot model instead of $\boldsymbol{\theta}^0$ in Chapter 3.



(a) Learning a sample-size dependent transformation.



(b) Structure of residual blocks.

Figure 4.2: MetaModelNet architecture for learning model dynamics. We instantiate MetaModelNet as a deep residual network with residual blocks $i = 0,\ 1, \ldots, N$ in Figure 4.2a, which accepts few-shot model parameters $\boldsymbol{\theta}$ (trained on small datasets across a logarithmic range of sample sizes k, $k = 2^i$) as (multiple) inputs and regresses them to many-shot model parameters $\boldsymbol{\theta}^*$ (trained on large datasets) as output. The skip connections ensure the identity regularization. $\mathcal{F}_i$ denotes the meta-learner that transforms (regresses) k-shot $\boldsymbol{\theta}$ to $\boldsymbol{\theta}^*$. Figure 4.2b shows the structure of the residual blocks. Note that the meta-learners $\mathcal{F}_i$ for different k are derived from this *single, chained* meta-network, with nested circles (subnetworks) corresponding to $\mathcal{F}_i$.

those in Chapter 3 and [310]. With only the performance loss, Eqn. (4.1) reduces to the loss function in [310]. When the performance loss is evaluated on the subsampled set, Eqn. (4.1) reduces to the loss function in Eqn. (3.1) in Chapter 3.

Training. What should be the value of k, for the k-shot models being trained? One might be tempted to set $k = t$, but this implies that there will be some head classes near the cutoff that have only t training examples, implying $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^*$ will be identical. To ensure that a meaningful mapping is learned, we set

$$k = t/2.$$

In other terms, we intentionally learn *very*-few-shot models to ensure that target model parameters are sufficiently more general.

4.3.2 Recursive Residual Transformations

We wish to apply the above module on all possible head-tail splits of a long-tailed training set. To do so, we extend the above approach in three crucial ways:

- (Sample-size dependency) Generate a sequence of different meta-learners $\mathcal{F}_i$, each tuned for a specific k, where $k = k(i)$ is an increasing function of i (that will be specified shortly). Through a straightforward extension, prior work on model regression in Chapter 3 learns a single fixed meta-learner for all the k-shot regression tasks.

- (Identity regularization) Ensure that the meta-learner defaults to the identity function for large i: $\mathcal{F}_i \to \mathcal{I}$ as $i \to \infty$.

- (Compositionality) Compose meta-learners out of each other: $\forall i < j, \mathcal{F}_i(\boldsymbol{\theta}) = \mathcal{F}_j\Big(\mathcal{F}_{ij}(\boldsymbol{\theta})\Big)$ where $\mathcal{F}_{ij}$ is the regressor that maps between $k(i)$-shot and $k(j)$-shot models.

Here we dropped the explicit dependence of $\mathcal{F}(\cdot)$ on $\boldsymbol{w}$ for notational simplicity. These observations emphasize the importance of (1) the identity regularization and (2) sample-size dependent regressors for long-tailed model transfer. We operationalize these extensions with a recursive residual network:

$$\mathcal{F}_i(\boldsymbol{\theta}) = \mathcal{F}_{i+1}\Big(\boldsymbol{\theta} + f(\boldsymbol{\theta}; \boldsymbol{w}_i)\Big), \tag{4.2}$$

where f denotes a residual block parameterized by $\boldsymbol{w}_i$ and visualized in Figure 4.2b. Inspired by [149,395], f consists of batch normalization (BN) and leaky ReLU as pre-activation, followed by fully-connected weights. By construction, each residual block transforms an input $k(i)$-shot model to a $k(i+1)$-shot model. The final MetaModelNet can be efficiently implemented through a chained network of $N+1$ residual blocks, as shown in Figure 4.2a. By feeding in a few-shot model at a particular block, we can derive any meta-learner $\mathcal{F}_i$ from the central underlying chain.

4.3.3 Training

Given the network structure defined above, we now describe an efficient method for training based on two insights. (1) The recursive definition of MetaModelNet suggests a recursive strategy for training. We begin with the *last* block and train it with the *largest* threshold (*e.g.*, those few classes in the head with many examples). The associated k-shot regressor should be easy to learn because it is similar to an identity mapping. Given the learned parameters for the last block, we then train the next-to-last block, and so on. (2) Inspired by the general observation that recognition performance improves *on a logarithmic scale* as the number of training samples increases [353, 439, 440], we discretize blocks accordingly, to be tuned for 1-shot, 2-shot, 4-shot, ... recognition. In terms of notation, we write the recursive training procedure as follows. We iterate over blocks i from N to 0, and for each i:

- Using Eqn. (4.1), train parameters of the residual block $\boldsymbol{w}_i$ on the head split H_t with k-shot model regression, where $k = 2^i$ and $t = 2k = 2^{i+1}$.

The above "back-to-front" training procedure works because whenever block i is trained, all subsequent blocks $(i+1, \ldots, N)$ have already been trained. In practice, rather than holding all subsequent blocks fixed, it is natural to fine-tune them while training block i. One approach might be fine-tuning them on the current $k = 2^i$-shot regression task being considered at iteration i. But because MetaModelNet will be applied across a wide range of k, we fine-tune blocks in a *multi-task* manner across the current viable range of $k = (2^i, 2^{i+1}, \ldots, 2^N)$ at each iteration i.

4.3.4 Implementation Details

We learn the CNN models on the long-tailed recognition datasets in different scenarios: (1) using a CNN pre-trained on ILSVRC 2012 [91, 185, 212] as the off-the-shelf feature; (2) fine-tuning the pre-trained CNN; and (3) training a CNN from scratch. We use ResNet152 [154] for its state-of-the-art performance and use ResNet50 [154] and AlexNet [212] for their easy computation.

When training the residual block i, we use the corresponding threshold t and obtain C_t head classes. We generate the C_t-way many-shot classifiers on H_t. For few-shot models, we learn C_t-way k-shot classifiers on random subsets of H_t. Through random sampling, we generate S model mini-batches and each model mini-batch consists of C_t weight vector pairs. In addition, to minimize the loss function (4.1), we randomly sample 256 image-label pairs as a data mini-batch from H_t.

We then use Caffe [185] to train our MetaModelNet on the generated model and data mini-batches based on standard SGD. λ is cross-validated. We use 0.01 as the negative slope for leaky ReLU. Computation is naturally divided into two stages: (1) training a collection of few/many-shot models and (2) learning MetaModelNet from those models. (2) is equivalent to progressively learning a nonlinear regressor. (1) can be made efficient because it is naturally parallelizable across models, and moreover, many models make use of only small training sets.

4.4 Experimental Evaluation

In this section, we explore the use of our MetaModelNet on long-tail recognition tasks. We begin with extensive evaluation of our approach on scene classification of the SUN-397 dataset [414], and address the meta-network variations and different design choices. We then visualize and empirically analyze the learned model dynamics. Finally, we evaluate on the challenging large-scale, scene-centric Places [436] and object-centric ImageNet datasets [321] and show the generality of our approach.

4.4.1 Evaluation and Analysis on SUN-397

Dataset and task. We start our evaluation by fine-tuning a pre-trained CNN on SUN-397, a medium-scale, long-tailed dataset with 397 classes and 100–2,361 images per class [414]. To better analyze trends due to skewed distributions, we carve out a more extreme version of the dataset. Following the experimental setup in [3,173,400], we first randomly split the dataset into train, validation, and test parts using 50%, 10%, and 40% of the data, respectively. The distribution of classes is uniform across all the three parts. We then randomly discard 49 images per class for the train part, leading to a long-tailed training set with 1–1,132 images per class (median 47). Similarly, we generate a small long-tailed validation set with 1–227 images per class (median 10), which we use for learning hyper-parameters. We also randomly sample 40 images per class for the test part, leading to a balanced test set. We report 397-way multi-class classification accuracy averaged over all classes.

Comparison with State-of-the-Art Approaches

We first focus on fine-tuning the classifier module while freezing the representation module of a pre-trained ResNet152 CNN model [154,400] for its state-of-the-art performance. Using MetaModelNet, we learn the model dynamics of the classifier module, *i.e.*, how the classifier weight vectors change during fine-tuning. Following the design choices in Section 4.3.2, our MetaModelNet consists of 7 residual blocks. For few-shot models, we generate $S = 1000$ 1-shot, $S = 500$ 2-shot, and $S = 200$ 4-shot till 64-shot models from the head classes for learning MetaModelNet. At test time, given the weight vectors of all the classes learned through fine-tuning, we feed them as inputs to the different residual blocks according to their training sample size of the corresponding class. We then "hallucinate" the dynamics

Method	Plain [154]	Over-Sampling [343, 435]	Under-Sampling [148]	Cost-Sensitive [170]	MetaModelNet (Ours)
Acc (%)	48.03	52.61	51.72	52.37	**57.34**

Table 4.1: Performance comparison between our MetaModelNet and state-of-the-art approaches for long-tailed scene classification when fine-tuning the pre-trained ILSVRC ResNet152 on the SUN-397 dataset. We focus on learning the model dynamics of the classifier module while freezing the CNN representation module. By benefiting from the learned generic model dynamics from head classes, ours significantly outperforms all the baselines for the long-tail recognition.

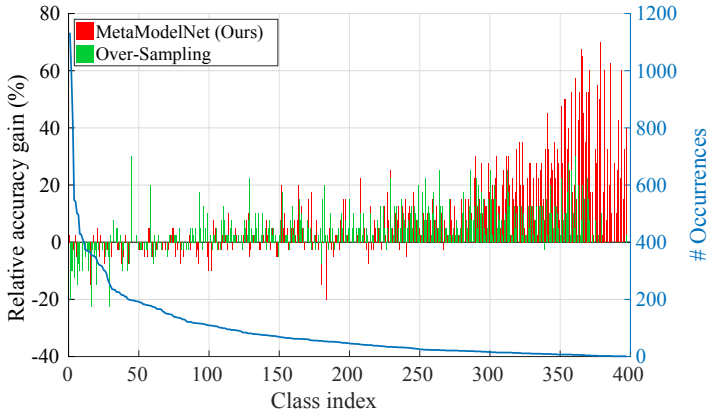


Figure 4.3: Detailed per class performance comparison between our MetaModelNet and the state-of-the-art over-sampling approach for long-tailed scene classification on the SUN-397 dataset. X-axis: class index. Y-axis (Left): per class classification accuracy improvement relative to the plain baseline. Y-axis (Right): number of training examples. Ours significantly improves for the few-shot tail classes.

of these weight vectors and use the outputs of MetaModelNet to modify the parameters of the final recognition model as in [395].

Baselines. In addition to the "plain" baseline that fine-tunes on the target data following the standard practice, we compare against three state-of-the-art baselines that are widely used to address the imbalanced distributions. (1) Over-sampling [343, 435], which uses the balanced sampling via label shuffling as in [343, 435]. (2) Under-sampling [148], which reduces the number of samples per class to 47 at most (the median value). (3) Cost-sensitive [170], which introduces additional weights in the loss function for each class with inverse class frequency. For a fair comparison, fine-tuning is performed for around 60 epochs using SGD with an initial learning rate of 0.01, which is reduced by a factor of 10 around every 30 epochs. All the other hyper-parameters are the same for all approaches.

Table 4.1 summarizes the performance comparison averaged over all classes and Figure 4.3 details the per class comparison. Table 4.1 shows that our MetaModelNet provides a promising way of encoding the shared structure across classes *in model space*. It outperforms existing approaches for long-tail recognition by a large margin. Figure 4.3 shows that our approach significantly improves accuracy in the tail.

Method	Model Regression [395]	MetaModelNet+Fix Split (Ours)	MetaModelNet+ Recur Split (Ours)
Acc (%)	54.68	56.86	**57.34**

Table 4.2: Ablation analysis of variations of our MetaModelNet. In a fixed head-tail split, ours outperforms the original approach in Chapter 3 and [395], showing the merit of learning a sample-size dependent transformation. By recursively partitioning the entire classes into different head-tail splits, our performance is further improved.

Ablation Analysis

We now evaluate variations of our approach and provide ablation analysis. Similar as in Section 4.4.1, we use ResNet152 in the first two sets of experiments and only fine-tune the classifier module. In the last set of experiments, we use ResNet50 [154] for easy computation and fine-tune through the entire network. Tables 4.2 and 4.3 summarize the results.

Sample-Size Dependent Transformation and Identity Regularization. We compare with the original approach in Chapter 3 and [395], which learns a single transformation for a variety of sample sizes and k-shot models, and importantly, learns a network without identity regularization. For a fair comparison, we consider a variant of MetaModelNet trained on a fixed head and tail split, selected by cross-validation. Table 4.2 shows that training for a fixed sample size and identity regularization provide a noticeable performance boost (2%).

Recursive Class Splitting. Adding multiple head-tail splits through recursion further improves accuracy by a small but noticeable amount (0.5% as shown in Table 4.2). We posit that progressive knowledge transfer outperforms the traditional approach because ordering classes by frequency is a natural form of curriculum learning.

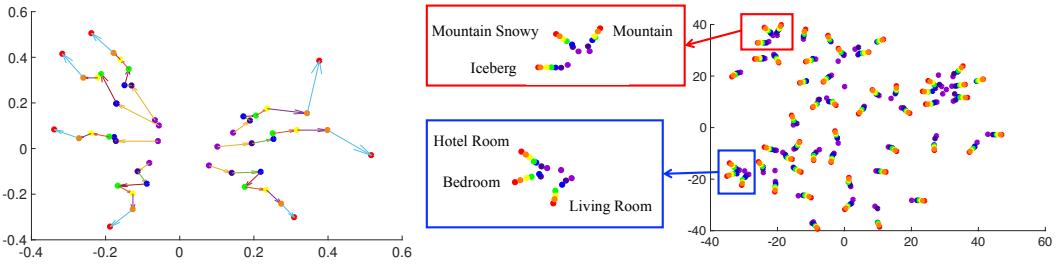
Joint Feature Fine-Tuning and Model Dynamics Learning. We also explore (nonlinear) fine-tuning of the "entire network" during head-to-tail transfer by jointly learning the classifier dynamics and the feature representation using ResNet50. We explore two approaches as follows. (1) We first fine-tune the whole CNN on the entire long-tailed training dataset, and then learn the classifier dynamics using the fixed, fine-tuned representation. (2) During the recursive head-tail splitting, we fine-tune the entire CNN on the current head classes in H_t (while learning the many-shot parameters $\boldsymbol{\theta}^*$), and then learn classifier dynamics using the fine-tuned features. Table 4.3 shows that progressively learning classifier dynamics while fine-tuning features performs the best.

4.4.2 Understanding Model Dynamics

Because model dynamics are highly nonlinear, a theoretical proof is rather challenging and outside the scope of this work. Here we provide some empirical analysis of model dynamics. When analyzing the "dual model (parameter) space", in which models parameters $\boldsymbol{\theta}$ can be viewed as points, Figure 4.4 shows that our MetaModelNet learns an approximately-smooth, nonlinear warping of this space that transforms (few-shot) input points to (many-shot) output points. For example, iceberg and mountain scene classes are more similar to

Scenario	Pre-Trained Features		Fine-Tuned Features (FT)		
Method	Plain [154]	MetaModelNet (Ours)	Plain [154]	Fix FT + MetaModelNet (Ours)	Recur FT + MetaModelNet (Ours)
Acc (%)	46.90	54.99	49.40	58.53	**58.74**

Table 4.3: Ablation analysis of joint feature fine-tuning and model dynamics learning on a ResNet50 base network. Though results with pre-trained features underperform those with a deeper base network (ResNet152, the default in our experiments), fine-tuning such features significantly improves results, even outperforming the deeper base network. By progressively fine-tuning the representation during the recursive training of MetaModelNet, performance significantly improves from 54.99% (changing only the classifier weights) to 58.74% (changing the entire CNN).



(a) PCA visualization.

(b) t-SNE visualization.

Figure 4.4: Visualizing model dynamics. Recall that $\boldsymbol{\theta}$ is a fixed-dimensional vector of model parameters — *e.g.*, $\boldsymbol{\theta} \in \mathbb{R}^{2048}$ when considering parameters from the last layer of ResNet. We visualize models as points in this "dual" space. Specifically, we examine the evolution of parameters predicted by MetaModelNet with dimensionality reduction — PCA (Figure 4.4a) or t-SNE [384] (Figure 4.4b). 1-shot models (purple) to many-shot models (red) are plotted in a rainbow order. These visualizations show that MetaModelNet learns an approximately-smooth, nonlinear warping of this space that transforms (few-shot) input points to (many-shot) output points. PCA suggests that many-shot models tend to have larger norms, while t-SNE (which nonlinearly maps nearby points to stay close) suggests that similar semantic classes tend to be close and transform in similar ways, *e.g.*, the blue rectangle encompasses "room" classes while the red rectangle encompasses "wintry outdoor" classes.

each other than to bedrooms. This implies that few-shot iceberg and mountain scene models lie near each other in parameter space, and moreover, they transform in similar ways (when compared to bedrooms). This *single* meta-network hence encodes *class-specific model transformations*. We posit that the transformation may capture some form of (*class-specific*) data augmentation. Finally, we find that some properties of the learned transformations are quite class-agnostic and apply in generality. Many-shot model parameters tend to have larger magnitudes and norms than few-shot ones (*e.g.*, on SUN-397, the average norm of 1-shot models is 0.53; after transformations through MetaModelNet, the average norm of the output models becomes 1.36). This is consistent with the common empirical observation that classifier weights tend to grow with the amount of training data, showing that they become more confident about their prediction.

Dataset	Places-205 [436]		ILSVRC-2012 [321]	
Method	Plain [212]	MetaModelNet (Ours)	Plain [212]	MetaModelNet (Ours)
Acc (%)	23.53	**30.71**	68.85	**73.46**

Table 4.4: Performance comparisons on long-tailed, large-scale scene-centric Places [436] and object-centric ImageNet [321] datasets. Our MetaModelNets facilitate the long-tail recognition with significantly diverse visual concepts and distributions.

4.4.3 Generalization to Other Tasks and Datasets

We now focus on the more challenging, large-scale scene-centric Places [436] and object-centric ImageNet [321] datasets. While we mainly addressed the model dynamics when fine-tuning a pre-trained CNN in the previous experiments, here we train AlexNet models [212] from scratch on the target tasks. Table 4.4 shows the generality of our approach and shows that MetaModelNets facilitate the recognition of other long-tailed datasets with significantly different visual concepts and distributions.

Scene Classification on the Places Dataset. Places-205 is a large-scale dataset which contains 2,448,873 training images approximately evenly distributed across 205 classes [436]. To generate its long-tailed version and better analyze trends due to skewed distributions, we distribute it according to the distribution of SUN and carve out a more extreme version (p^2, or $2\times$ the slope in log-log plot) out of the Places training portion, leading to a long-tailed training set with 5–9,900 images per class (median 73). We use the provided validation portion as our test set with 100 images per class.

Object Classification on the ImageNet Dataset. The ILSVRC 2012 classification dataset contains 1,000 classes with 1.2 million training images (approximately balanced between the classes) and 50K validation images [321]. There are 200 classes used for object detection which are defined as higher-level classes of the original 1,000 classes. Taking the ILSVRC 2012 classification dataset and merging the 1,000 classes into the 200 higher-level classes, we obtain a natural long-tailed distribution.

Chapter 5

Learning to Initialize and Adapt for Few-Shot Motion Prediction

5.1 Motivation

In this chapter, we exploit our learning to learn approach beyond recognition to a *prediction* task. One of the hallmarks of human intelligence is the ability to predict the future based on past observations. Through perceiving and forecasting how the environment evolves and how a fellow human acts, a human learns to interact with the world [387]. Remarkably, humans acquire such prediction ability from just few experiences, which is yet generalizable across different scenarios [337]. Similarly, to allow natural and effective interaction with humans, artificial agents should be able to do the same, *i.e.*, forecasting how a human moves or acts in the near future conditioning on a series of historical movements [204]. As a more concrete example illustrated in Figure 5.1, when deployed in natural environments, robots are supposed to predict unfamiliar actions after seeing only few examples. While *human motion prediction* has attracted increasing attention recently [20, 117, 123, 171, 183, 203, 207, 254, 286], the existing approaches rely on extensive annotated motion capture (mocap) data and are brittle for novel actions.

We believe that the significant gap between human and machine prediction arises from two issues. First, motion dynamics are difficult to model because they entangle physical constraints with goal-directed behaviors [254]. Second, there exists a lack of large-scale, annotated motion data. Current mocap datasets are constructed with dedicated sensored environments and so are *not scalable*. We think that this motivates the exploration of motion models learned from limited training data. Unfortunately, a substantial amount of annotated data is required for the state-of-the-art deep recurrent encoder-decoder network based models [20, 117, 123, 183, 254] to learn the desired motion dynamics. One stark evidence of this is that a constant pose predictor [254], as a naïve approach that does not produce interesting motion, sometimes achieves the best performance. One attractive solution is learning a "basis" of underlying knowledge that is shared across a wide variety of action classes, including never-before-seen actions. Much previous work learns a separate model for each action and restricts the training process on subsets of mocap data. More recently, one-hot vectors are introduced to incorporate the action class information and learn a single model for different action classes [254]. This, however, is still difficult to generalize to novel ac-

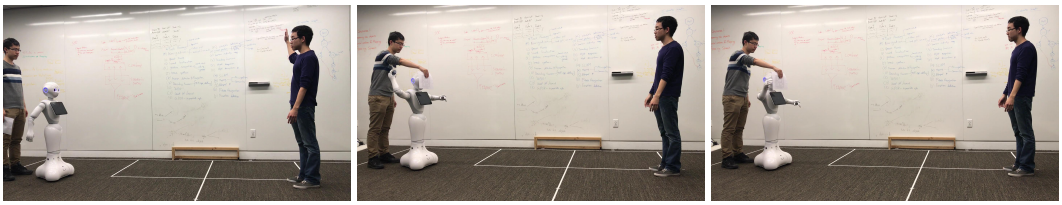


Figure 5.1: Illustration of the importance of *few-shot human motion prediction* for applications such as human-robot interaction and collaboration. When deployed in natural environments, a robot is supposed to predict unfamiliar actions after seeing only few examples through a camera inside its eyes, in the absence of large-scale, annotated motion capture data. Through forecasting how a human moves or acts in the near future conditioning on a series of historical movements, the robot could respond appropriately and expeditiously. **Left:** the robot starts learning an unfamiliar action, "waving", by observing a few "waving" examples from the human standing in front of itself. **Middle:** The eyes of the robot are blinded. **Right:** The robot is predicting "waving" action. Our meta-learning framework facilitates the few-shot motion prediction.

tions. Transfer learning [16, 54, 91, 283, 288, 311, 400] can be in principle applied to alleviate this problem by fine-tuning a pre-trained network from another task which has more labeled data; nevertheless, the benefit of pre-training decreases as the source task diverges from the target task [427].

Here we make the first attempt to address *few-shot human motion prediction*. Inspired by the recent progress on meta-learning and few-shot learning [114, 335, 366, 388, 395], we propose a novel meta-learning framework tailored for the task of human motion prediction. While there is increasing research on meta-learning and few-shot learning, most of the current approaches are developed for the simple domains like image classification with task-specific model architectures [259, 351, 388], which cannot be easily applied to the task we are interested in. *Our key insight* is that having a good generalization from few examples relies on both a generic initial model and an effective strategy for adapting this model to novel tasks. Based on this insight, our *proactive and adaptive meta-learning (PAML)* introduces a novel combination of the state-of-the-art model-agnostic meta-learning (MAML) [114] and model regression networks (MRN) in Chapter 3, and unifies them into an *integrated, end-to-end* framework. By doing so, MAML enables the meta-learner to aggregate contextual information from a variety of prediction tasks and thus produce a generic model initialization, and meanwhile MRN allows the meta-learner to transform a small-sample model and thus improve its generalization.

More concretely, what is needed in the first place is a systematic way to learn a beneficial common initialization that would serve as a good point to start training for the novel action being considered. This can be achieved by explicitly learning the initial parameters of a predictor model in a way that the model has maximal performance on a new task after the parameters have been updated with few training examples from that new task. To this end, we make use of the approach of MAML [114], which initializes the weights of a network such that standard SGD can make rapid progress on new task. We learn this initialization through a meta-learning procedure that learns from a large set of motion prediction tasks with small amounts of data. After obtaining the pre-trained model, MAML uses one or few

SGD updates to adapt it to novel tasks. Although the initial model is somewhat generic, plain SGD updates can only slightly modify its parameters [400] especially in the small-sample size regime, otherwise it would lead to severe over-fitting to the new data [155]. This is still far from satisfactory, because the obtained task-specific model is different from the one that would be learned from a large set of samples.

To address this limitation, we consider the meta-learning approaches that learn an update function or learning rule. Specifically, we leverage MRN in Chapter 3 as the adaptation strategy, which learns a meta-level network to regress many-shot model parameters from few-shot model parameters. While MRN was developed in the context of convolutional neural networks, we extend it to recurrent neural networks. More importantly, we unify MAML with MRN as PAML, an end-to-end framework for motion prediction. Our PAML model is not only directly initialized to produce the kinds of parameters that are useful for later adaptation, but it can also be effectively adapted to novel actions through exploiting the structure of model parameters shared across action classes.

Our contributions are three-fold. (1) To the best of our knowledge, this is the first time exploring the few-shot learning problem for human motion prediction. We show how meta-learning can be operationalized for predicting human motion from few examples. (2) We present a novel meta-learning approach, combining model-agnostic meta-learning with model regression networks, that learns both a generic model initialization and an effective model adaptation strategy jointly. Our approach is general and can be applied to other tasks as well. (3) We show how our approach facilitates the prediction of novel action classes from few examples, leading to significantly improved performance on the challenging motion capture dataset.

5.2 Human Motion Prediction

Human motion prediction has great application potential in a variety of scenarios in computer vision and robotic vision, including motion generation for computer graphics [207], proactive decision-making in autonomous driving systems [286], action anticipation [171, 203], and human-robot interaction and collaboration [204]. Traditional approaches mainly focus on state-space equations and latent-variable models, such as hidden Markov models [47], linear dynamic models [295], Gaussian process latent variable models [383,391], bilinear spatio-temporal basis models [5], and restricted Boltzmann machines [355,359–361]. In the deep learning era, the recurrent neural networks (RNNs) based approaches have attracted more attention and significantly pushed the state of the art in human motion prediction [117,123,183,201,254,423].

Flagship techniques include LSTM-3LR, ERD [117], SRNNs [183], and residual sup. [254]. LSTM-3LR (3 layers of long short-term memory cells) learns pose representation and temporal dynamics simultaneously via curriculum learning [117]. In additional to the concatenated LSTM units as in LSTM-3LR, ERD (encoder-recurrent-decoder) consists of nonlinear space encoders for data pre-processing [117]. SRNNs (structural RNNs) model human activity with a hand-designed spatio-temporal graph and introduce the encoded semantic knowledge into a recurrent network [183]. These approaches fail to consider the shared knowledge across action classes and they thus learn action-specific models and restrict the training process on the corresponding subsets of the motion capture (mocap) dataset. Residual sup. is a simple sequence-to-sequence architecture with a residual connection, which incorporates the action class information via one-hot vectors [254]. Despite their

promise, the existing methods directly learn on the target task with large amounts of training data, and cannot generalize well from few examples or to novel action classes. There has been little work on few-shot motion prediction as ours, which is crucial for robot learning in practice. Our task is also significantly different from few-shot imitation learning: while this line of work aims to learn and mimic human motion from demonstration [97, 115, 290, 442], our goal is to *predict unseen future* motions based on historical observations.

Our approach falls more into the learning to learn approaches. Often, such approaches cannot be easily re-purposed to handle different model architectures and different problem settings. Moreover, they aim to either obtain a better model initialization [114, 278] or learn an update function or learning rule [10, 32, 310, 332, 395], *but not the both*. By contrast, we present a unified view by taking these two aspects into consideration and show how they compensate and complement with each other in an integrated, end-to-end meta-learning framework. Our approach is also general and can be potentially applied to other tasks in addition to human motion prediction.

5.3 Proactive and Adaptive Meta-Learning

We now present our meta-learning framework for few-shot human motion prediction. The predictor (*i.e.*, learner) is a recurrent encoder-decoder network, which frames motion prediction as a sequence-to-sequence problem. The encoder takes as inputs a sequence of historical 3D skeletons and infers a latent representation. The decoder takes as inputs this latent representation and a seed motion frame and produces the predicted future sequence. The encoder and decoder are learned jointly to make the predicted sequence as close as to its ground-truth sequence.

To enable the predictor to rapidly produce satisfactory prediction from just few training sequences for a novel task (*i.e.*, action class), we introduce proactive and adaptive meta-learning (PAML). Through learning a large collection of few-shot prediction tasks on known action classes, the predictor jointly learns a generic model initialization and an effective model adaptation strategy. In the following sections, we first describe the meta-learning setup for motion prediction tasks and explain the corresponding predictor architecture and meta-learning procedure.

5.3.1 Meta-Learning Setup for Human Motion Prediction

Human motion is typically represented as sequential data. Given a historical motion sequence, we predict possible motion in the short-term or long-term future. Our goal here is *few-shot motion prediction* that aims to train a predictor model that can quickly adapt to a new task (*i.e.*, a novel action class) using only few training sequences. To achieve this, we introduce a *meta-learning mechanism* that treats entire prediction tasks as training examples. During meta-learning, the predictor or learner is trained on a set of prediction tasks guided by a *high-level meta-learner*, so that the trained predictor can accomplish the desired few-shot adaptation ability.

The predictor (*i.e.*, learner), represented by a parametrized function $\mathcal{P}_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$, maps an input sequence $\boldsymbol{X}$ to an output sequence $\widehat{\boldsymbol{Y}}$. We denote the input motion sequence of length n as $\boldsymbol{X} = \left\{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^n\right\}$, where $\boldsymbol{x}^i \in \mathbb{R}^d, i = 1, \ldots, n$ is a motion vector consisting of a set of 3D body joint angles using their exponential map representations [271], and d is the number of joint angles. The learner predicts motion vectors

$\widehat{\boldsymbol{Y}} = \left\{\widehat{\boldsymbol{x}}^{n+1}, \widehat{\boldsymbol{x}}^{n+2}, \ldots, \widehat{\boldsymbol{x}}^{n+m}\right\}$ in the next m time steps, where $\widehat{\boldsymbol{x}}^{j} \in \mathbb{R}^{d}, j = n+1, \ldots, n+m$ is the predicted motion vector at time j, and m is the output sequence length. The ground-truth of the future sequence is denoted as $\boldsymbol{Y}^{gt} = \left\{\boldsymbol{x}^{n+1}, \boldsymbol{x}^{n+2}, \ldots, \boldsymbol{x}^{n+m}\right\}$.

Meta-learning aims to train a *learning procedure* (*i.e.*, the meta-learner) that enables the predictor model to adapt to a large number of prediction tasks. For the k-shot prediction task, each task $T = \{L, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}\}$ aims to predict a certain action from few examples. It consists of a loss function L, a small training set $\mathcal{D}_{\text{train}} = \{(\boldsymbol{X}_u, \mathbf{Y}_u^{gt})\}, u = 1, \ldots, k$ with k action-specific past and future sequence pairs, and a test set $\mathcal{D}_{\text{test}}$ that has a set number of past and future sequence pairs for evaluation. The frame-wise Euclidean distance is commonly used as the loss function L for motion prediction. For each task, the meta-learner takes as input the training set $\mathcal{D}_{\text{train}}$ and produces a predictor (learner) that achieves high average prediction performance on its corresponding $\mathcal{D}_{\text{test}}$.

More precisely, we consider a distribution $p(T)$ over prediction tasks that we want our predictor to be able to adapt to. Meta-learning algorithms have two stages: meta-training and meta-testing. During the meta-training stage, a prediction task T_i is sampled from $p(T)$, and the predictor $\mathcal{P}$ is trained on its corresponding small training set $\mathcal{D}_{\text{train}}$ with the loss L_{T_i} from T_i. The predictor is then improved by considering how the test error on the corresponding test set $\mathcal{D}_{\text{test}}$ changes with respect to the parameters. The test error serves as the training error of the meta-learning process. During the meta-testing stage, a held-out set of prediction tasks drawn from $p(T)$ (*i.e.*, novel action classes) with its own small training set $\mathcal{D}_{\text{train}}$ and test set $\mathcal{D}_{\text{test}}$ are used to evaluate the performance of the predictor.

5.3.2 Learner: Encoder-Decoder Architecture

We use the state-of-the-art motion predictor in [254] as our learner $\mathcal{P}$. It is a recurrent encoder-decoder network that maps input to output sequences of skeletons. The encoder and decoder consist of gated recurrent unit (GRU) [66] cells as the building blocks, due to their superior performance. The input sequence is passed through the encoder to infer a latent representation. This latent representation and a seed motion frame are then fed into the decoder to output the first time step prediction. The decoder takes the output of itself as the next time step input and generates further prediction sequentially. Different from [254], to deal with novel action classes, we do not use one-hot vectors to indicate the action class of the current input.

5.3.3 Proactive Meta-Learner: Generic Model Initialization

Intuitively, if we have a *universal predictor* that is broadly applicable to a variety of tasks in $p(T)$ instead of a single individual task, it would serve as a good point to start training for a novel target task. We explicitly learn such general-purpose initial model by using the approach of model-agnostic meta-learning (MAML) [114]. MAML is developed for gradient-based learning rules (*e.g.*, SGD) and aims to learn a model in a way that a few SGD updates can make rapid progress on new tasks without over-fitting.

Concretely, when adapting to a new task T_i, the predictor's parameters $\boldsymbol{\theta}$ become $\boldsymbol{\theta}_i'$. In MAML, this is computed using one or more SGD updates on $\mathcal{D}_{\text{train}}$ of task T_i. For the sake of simplicity and without loss of generality, we consider one SGD update:

$$\boldsymbol{\theta}_i' = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L_{T_i}\left(\mathcal{P}_{\boldsymbol{\theta}}\right), \tag{5.1}$$

where α is the learning rate hyper-parameter. We aim to optimize the initial $\boldsymbol{\theta}$ such that the updated $\boldsymbol{\theta}_i'$ will produce maximal performance on the corresponding test set $\mathcal{D}_{\text{test}}$ of task T_i. When averaged across the tasks sampled from $p(T)$, we have the following meta-objective function:

$$\min_{\boldsymbol{\theta}} \sum_{T_i \sim p(T)} L_{T_i}\left(\mathcal{P}_{\boldsymbol{\theta}_i'}\right) = \sum_{T_i \sim p(T)} L_{T_i}\left(\mathcal{P}_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L_{T_i}(\mathcal{P}_{\boldsymbol{\theta}})}\right). \tag{5.2}$$

Note that the meta-optimization is performed over the predictor parameters $\boldsymbol{\theta}$, whereas the objective is computed using the updated model parameters $\boldsymbol{\theta}'$. This meta-optimization across tasks is performed via SGD in the form of

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{T_i \sim p(T)} L_{T_i}\left(f_{\boldsymbol{\theta}_i'}\right), \tag{5.3}$$

where β is the meta-learning rate hyper-parameter. During each iteration, we sample task mini-batch from $p(T)$ and perform the corresponding learner update in Eqn. (5.1) and meta-learner update in Eqn. (5.3).

5.3.4 Adaptive Meta-Learner: Model Adaptation Strategy

In MAML, the model parameters $\boldsymbol{\theta}_i'$ of a new task T_i are obtained by performing few plain SGD updates on top of the initial $\boldsymbol{\theta}$ using its small training set $\mathcal{D}_{\text{train}}$, following Eqn. (5.1). After meta-training, $\boldsymbol{\theta}$ tend to be generic. However, with limited training data from $\mathcal{D}_{\text{train}}$, SGD updates can only modify $\boldsymbol{\theta}$ slightly, which is still far away from the desired $\boldsymbol{\theta}_i^*$ that would be learned from a large set of target samples. Higher-level knowledge is thus necessary to guide the model adaptation to novel tasks. To this end, we consider model regression networks (MRN) in Chapter 3 as the adaptation strategy. MRN is developed in the context of image classification and learns a generic transformation from models learned from few samples to models learned from large enough sample sets.

More formally, let $\boldsymbol{\theta}_i^0$ denote the model parameters learned from small training set $\mathcal{D}_{\text{train}}$ by using SGD updates (*i.e.*, $\boldsymbol{\theta}_i'$ in Eqn. (5.1)). Let $\boldsymbol{\theta}_i^*$ denote the corresponding *underlying* model parameters learned from a large set of annotated samples. Our goal is to make the updated $\boldsymbol{\theta}_i'$ as close as to the desired $\boldsymbol{\theta}_i^*$. MRN assumes that there exists a generic non-linear transformation, represented by a regression function $\mathcal{H}(\cdot)$, parameterized with $\boldsymbol{w}$, in the model parameter space, such that $\boldsymbol{\theta}_i^* \approx \mathcal{H}\left(\boldsymbol{\theta}_i^0; \boldsymbol{w}\right)$ for a broad range of tasks T_i. The square of the Euclidean distance is used to quantify the quality of the approximation. We then estimate this transformation based on a large set of known tasks T_i drawn from $p(T)$, with model pairs $\left\{\left(\boldsymbol{\theta}_i^0, \boldsymbol{\theta}_i^*\right)\right\}$, during meta-training as follows:

$$\min_{\boldsymbol{w}} \sum_{T_i \sim p(T)} \left\| \mathcal{H}\left(\boldsymbol{\theta}_i^0; \boldsymbol{w}\right) - \boldsymbol{\theta}_i^* \right\|^2. \tag{5.4}$$

Consistent with Chapter 3, we use multi-layer feed-forward neural networks as the regression function $\mathcal{H}$.

5.3.5 An Integrated Framework

We introduce the adaptation strategy both in the meta-testing and meta-training phases. For task T_i, after performing few SGD updates on small training set $\mathcal{D}_{\text{train}}$, we then apply

the transformation $\mathcal{H}$ to obtain $\boldsymbol{\theta}_i'$. That is, Eqn. (5.1) is modified as

$$\boldsymbol{\theta}_i' = \mathcal{H}\left(\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L_{T_i}\left(\mathcal{P}_{\boldsymbol{\theta}}\right); \boldsymbol{w}\right). \tag{5.5}$$

During meta-training, for task T_i, we also have the underlying $\boldsymbol{\theta}_i^*$, which is obtained by performing SGD updates on the corresponding large sample set. Now, the meta-objective in Eqn. (5.2) becomes

$$\min_{\boldsymbol{\theta}, \boldsymbol{w}} \sum_{T_i \sim p(T)} L_{T_i}\left(\mathcal{P}_{\boldsymbol{\theta}_i'}\right) + \frac{1}{2}\lambda \left\|\boldsymbol{\theta}_i' - \boldsymbol{\theta}_i^*\right\|^2, \tag{5.6}$$

where λ is the trade-off hyper-parameter. Note that this is a *joint optimization* with respect to both $\boldsymbol{\theta}$ and $\boldsymbol{w}$, and we perform this meta-optimization across tasks using SGD. Hence, we integrate both model initialization and adaptation into an end-to-end meta-learning framework. The model is initialized to produce the parameters that are optimal for its adaptation; meanwhile, the model is effectively adapted by leveraging "learning to learn" knowledge about the relationship between small-sample and large-sample models.

During meta-testing, for a novel prediction task, with the learned generic model initialization $\boldsymbol{\theta}$ and model transformation $\mathcal{H}$, we use Eqn. (5.5) to obtain the task-specific predictor model.

5.4 Experimental Evaluation

We now present the experimental results of few-shot human motion prediction based on our proactive and adaptive meta-learning (PAML). Our approach is general and can be in principle applied to a broad range of few-shot learning tasks. For performance calibration, we begin with a sanity check of our approach on a standard few-shot image classification task and compare with existing meta-learning approaches. We then focus on our main task of human motion prediction. Through comparing with state-of-the-art motion prediction approaches, we show that our PAML significantly improves the prediction performance in the small sample size regime.

5.4.1 Sanity Check on Few-Shot Image Classification

While we propose a novel problem of few-shot human motion prediction, the majority of existing meta-learning and few-shot learning approaches are developed in the scenario of classification tasks. As a sanity check, the first question is how our meta-learning approach compares with these prior techniques. For a fair comparison, we evaluate on the standard few-shot image classification task. The most common setup is the N-way, k-shot classification that aims to classify data into N classes when we only have a small number (k) of labeled instances per class for training. The loss function is the cross-entropy error between the predicted and true labels. Following [114, 259, 310, 351, 388], we evaluate on the most widely used mini-ImageNet benchmark [310]. It consists of 64 meta-training and 24 meta-test classes, with 600 images of size 84×84 per class.

During meta-training, each task is sampled as an N-way, k-shot classification: we first randomly sample N classes from the meta-training classes; for each class, we randomly sample k and 1 examples to form the training and test set, respectively. During meta-testing,

Method	5-Way Acc (%)	
	1-shot	5-shot
Matching Networks [388]	43.56 ± 0.84	55.31 ± 0.73
MAML [114]	48.7 ± 1.84	63.1 ± 0.92
Meta-Learner LSTM [310]	43.4 ± 0.77	60.2 ± 0.71
Prototypical Networks [351]	46.61 ± 0.78	65.77 ± 0.70
Meta Networks [270]	49.21 ± 0.96	–
PAML (Ours)	$\mathbf{53.26 \pm 0.52}$	$\mathbf{68.19 \pm 0.61}$

Table 5.1: Performance sanity check of our approach by comparing with state-of-the-art meta-learning and few-shot learning approaches for few-shot image classification on the widely used mini-ImageNet dataset. Our PAML outperforms these baselines, showing its general effectiveness for few-shot learning.

we report performance on the unseen classes from the meta-test classes. We use the convolutional network in [114] as the classifier (learner). Our model adaptation network is a 2-layers fully-connected network with leaky ReLU nonlinearity

Table 5.1 summarizes the performance comparisons with the existing approaches on the standard 5-way, 1-/5-shot setting. Our PAML consistently outperforms all the baselines. In particular, there is a notable 5% performance improvement compared with MAML, showing the complementary benefits of our model adaptation strategy. This sanity check verifies the effectiveness of our meta-learning approach. Moreover, some of these existing methods, such as matching networks and prototypical networks, are designed with few-shot classification in mind, and are not readily applicable to domains such as human motion prediction. In the following experiments, we show how our approach can be used to facilitate the few-shot motion prediction.

5.4.2 Few-Shot Human Motion Prediction

We now focus on using our meta-learning approach for human motion prediction. To the best of our knowledge, this is the first time exploring the few-shot learning problem for human motion prediction. Due to the lack of published protocols, we propose our evaluation protocol for this task.

Dataset. We evaluate on Human 3.6M [178], a heavily benchmarked, large-scale motion capture (mocap) dataset that has been widely used in human motion analysis. Human 3.6M contains seven actors performing 15 varied activities. Following the standard experimental setup in [117, 183, 254], we down-sample the dataset by two, train on six subjects, and test on subject five. Each activity contains hours of video from these actors performing such activity. Sequence clips are randomly taken from the training and test videos to construct the corresponding training and test sequences [183]. Given the past 50 mocap frames (2 seconds in total), we forecast the future 10 frames (400 milliseconds in total) in short-term prediction and the future 40 frames (1,600 milliseconds in total) in long-term prediction.

Few-shot learning task and meta-learning setup. We use 11 classes of activities for meta-training: directions, greeting, phoning, posing, purchases, sitting, sitting down, taking photo, waiting, walking dog, and walking together. And we use the remaining 4 classes of activities for meta-testing: walking, eating, smoking, and discussion. These 4 activities

are commonly used to evaluate motion prediction algorithms in the previous work [117, 183, 254]. The *k-shot motion prediction task* which we address is: for a certain activity, given a small collection of k *action-specific* past and future sequence pairs, we aim to learn a predictor model so that it is able to predict the possible future motion for a new past sequence from that activity. Accordingly, the setup of k-shot prediction tasks in meta-learning is as follows. During meta-training, for each task, we randomly select one activity out of 11, and we sample k action-specific sequence pairs as $\mathcal{D}_{\text{train}}$. During meta-testing, for each of the 4 novel activities, we sample k sequence pairs from its training set to produce small-sample set $\mathcal{D}_{\text{train}}$. We then adapt our meta-learned predictor to the target action-specific predictor. We evaluate it on the corresponding test set. We run five trials for each activity and report the average performance.

Implementation details. In our experiments, the predictor is residual sup., the state-of-the-art human motion prediction approach based on a sequence-to-sequence encoder-decoder network [254]. For the encoder and decoder, we use a single GRU cell [66] with hidden size 1024, respectively. Following [254], we use tied weights between the encoder and decoder. We use spatial embedding for both the encoder and decoder. We use separate model adaptation networks for the embedding layers and the encoder/decoder, each of which is a 3-layers fully-connected network with leaky ReLU nonlinearity. In most cases, k is set as 5 and we also evaluate how performance changes when k varies. By cross-validation, the trade-off parameter λ is set as 0.1, the learning rate α is set as 0.001, and the meta-learning rate β is set as 0.005. For the predictor, we clip the gradient to a maximum ℓ_2-norm of 5. We run $10,000$ iterations during meta-training. We use PyTorch [291] to train our model.

Baselines. For a fair comparison, we compare with residual sup. [254], which is the same predictor as ours but is not meta-learned. In particular, we evaluate its variations in the small sample size regime and consider learning both action-specific and action-agnostic models in the following scenarios.

- **(Type I) Action-specific training from scratch.** For each of the 4 target activities, we directly learn an action-specific predictor using its k training sequence pairs.
- **(Type II) Action-agnostic training from scratch.** We learn a single predictor for the 4 target activities using all their training sequence pairs.
- **(Type III) Off-the-shelf transfer.** We learn a single predictor for the 11 meta-training activities using their large number of training sequence pairs, and directly use this predictor for the 4 target activities without modification.
- **(Type IV) Multi-task learning.** We learn a single predictor for all the 15 activities using a large number of training sequence pairs per activity for the 11 meta-training activities and k sequence pairs per activity for the 4 target activities.
- **(Type V) Fine-tuning transfer.** After learning a single predictor for the 11 meta-training activities using their large number of training sequence pairs, we fine-tune it to become an action-specific predictor for each of the 4 target activities, respectively, using its k training sequence pairs.

Evaluation metrics. We evaluate our approach both quantitatively and qualitatively. For the quantitative evaluation, we use the mean square error between the predicted motions and the ground-truth motions in the angle space, which is the standard metric in motion prediction [117, 183, 254]. We exclude the translation and rotation of the whole body,

since this information is independent of the actions themselves. We also qualitatively visualize the predictions frame by frame, following [117, 183, 254].

		Walking						Eating					
Type	millisecond	80	160	320	400	560	1000	80	160	320	400	560	1000
Residual sup. [254] $w/$ (Baselines)	$\text{Scratch}_{\text{spec}}$	1.90	1.95	2.16	2.18	1.99	2.00	2.33	2.31	2.30	2.30	2.31	2.34
	$\text{Scratch}_{\text{agn}}$	1.78	1.89	2.20	2.23	2.02	2.05	2.27	2.16	2.18	2.27	2.25	2.31
	$\text{Transfer}_{\text{ots}}$	0.60	0.75	0.88	0.93	1.03	1.26	0.57	0.70	0.91	1.04	1.19	1.58
	Multi-task	0.57	0.71	0.79	0.85	0.96	1.12	0.59	0.68	0.83	0.93	1.12	1.33
	$\text{Transfer}_{\text{ft}}$	0.44	0.55	0.85	0.95	0.74	1.03	0.61	0.65	0.74	0.78	0.86	1.19
Meta-learning (Ours)	PAML	**0.36**	**0.49**	**0.73**	**0.85**	**0.81**	**0.88**	**0.37**	**0.54**	**0.66**	**0.73**	**0.76**	**0.83**

		Smoking						Discussion					
Type	millisecond	80	160	320	400	560	1000	80	160	320	400	560	1000
Residual sup. [254] $w/$ (Baselines)	$\text{Scratch}_{\text{spec}}$	2.88	2.86	2.85	2.83	2.80	2.99	3.01	3.13	3.12	2.95	2.62	2.99
	$\text{Scratch}_{\text{agn}}$	2.53	2.61	2.67	2.65	2.71	2.73	2.77	2.79	2.82	2.73	2.82	2.76
	$\text{Transfer}_{\text{ots}}$	0.70	0.84	1.18	1.23	1.38	2.02	0.58	0.86	1.12	1.18	1.54	2.02
	Multi-task	0.71	0.79	1.09	1.20	1.25	1.23	0.53	0.82	1.02	1.17	1.33	1.97
	$\text{Transfer}_{\text{ft}}$	0.87	1.02	1.25	1.30	1.45	2.06	0.57	0.82	1.11	1.11	1.37	2.08
Meta-learning (Ours)	PAML	**0.41**	**0.70**	**0.82**	**1.05**	**1.00**	**1.03**	**0.43**	**0.73**	**1.01**	**1.06**	**1.15**	**1.17**

Table 5.2: Mean angle error comparisons between our PAML and variants of residual sup. [254] on the 4 target activities of the Human 3.6M dataset for $k = 5$-shot motion prediction. Residual sup. is the state-of-the-art motion prediction approach; we evaluate its variants in the small sample size regime and consider learning both action-specific and action-agnostic models in different scenarios. Our PAML consistently and significantly outperforms all the baselines. In particular, it is superior to the multi-task learning and transfer learning baselines on all the actions across different time horizons.

Comparison with the state-of-the-art motion prediction approaches. Table 5.2 shows the quantitative comparisons between our PAML and a variety of variants of residual sup. on the 4 target activities in the small sample size regime. While residual sup. has achieved impressive performance with a large amount of annotated motion sequences [254], its prediction significantly degrades when only a limited number of training sequences are available. As expected, directly training the predictor from few examples leads to poor performance (*i.e.*, with the angle error in range $2 \sim 3$), due to severe over-fitting. In such scenarios of training from scratch, learning an action-agnostic model is slightly better than learning an action-specific one (*e.g.*, decreasing the angle error by 0.1 at 80ms for walking), since the former allows the predictor to exploit some common motion regularities from multiple activities. By transferring knowledge from relevant activities with large sets of samples in a more principled manner, the prediction performance improves a little bit. This is achieved by multi-task learning, *e.g.*, training an action-agnostic predictor using both the 11 source and 4 target activities, or transfer learning, *e.g.*, first training an action-agnostic predictor using the source activities, and then using it either in the off-the-shelf manner or though fine-tuning.

However, modeling multiple actions is more challenging than modeling each action separately, due to the significant diversity of different activities. The performance improvement

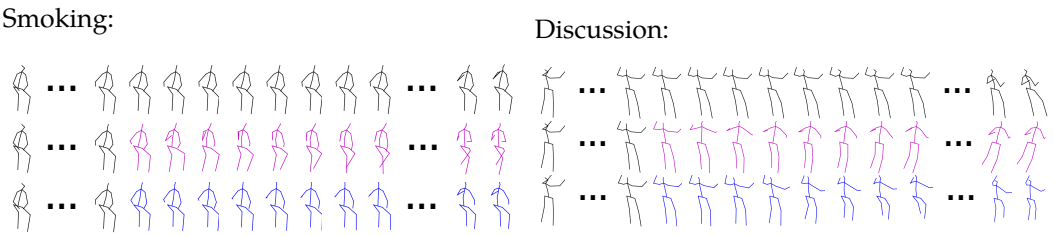


Figure 5.2: Visualizations for $k = 5$-shot motion prediction of smoking and discussion. Top: the conditioning sequence and the ground-truth of the prediction sequence. Middle: fine-tuning transfer of residual sup. [254], a top performing baseline. Bottom: our prediction results. The ground-truth and the conditioning sequences are shown in black, and the predictions are shown in color. Our PAML produces lower-error, smooth, and human-like prediction through meta-learning. **Best viewed in color with zoom.**

of these multi-task learning and transfer learning baselines is limited and their performance is also comparable. This thus demonstrates the general difficulty of our few-shot motion prediction task. By contrast, our PAML *consistently and significantly outperforms all the baselines on all the actions across different time horizons*, showing the effectiveness of our meta-learning mechanism. There is even a noticeable performance boost for the complicated motions (*e.g.*, decreasing the angle error by 0.5 at 80ms for smoking, compared with fine-tuning transfer). By explicitly learning from a large number of few-shot prediction tasks during meta-training, PAML is able to extract and leverage knowledge shared *both across different actions and across multiple few-shot prediction tasks*, thus improving the prediction of novel actions from few examples by a large margin.

Moreover, as mentioned before, most of the current meta-learning approaches, such as matching networks [388] and prototypical networks [351], are developed for the simple tasks like image classification with task-specific model architectures (*e.g.*, learning an embedding space that is useful for nearest neighbor or prototype classifiers), which are not readily applicable to our problem. Unlike them, our approach is general and can be effectively used across a broad range of tasks, as shown in Table 5.1 and Table 5.2. Figure 5.2 further visualizes our prediction and compares with a top performing baseline. From Figure 5.2, we can see that our PAML generates less-error, more smooth, and realistic prediction.

Ablation studies. In Table 5.3 and Table 5.4, we evaluate the contributions of different factors in our approach to the results.

Model initialization vs. model adaptation. Our meta-learning approach consists of two components: a generic model initialization and an effective model adaptation network. To understand the impact of each component, we conduct ablation analysis in Table 5.3. We can see that each component by itself is superior to the baselines reported in Table 5.2. This shows that meta-learning, in general, by leveraging shared knowledge across relevant tasks, enables us to deal with a novel task in a sample-efficient way. Moreover, our full PAML model consistently outperforms its variants, showing the effectiveness and complementarity of each component. This thus verifies the importance of simultaneously learning a generic initial model and an effective strategy for adapting this model to novel tasks.

Structure of $\mathcal{H}$. In Table 5.4 we compare different implementations of the model adaptation network $\mathcal{H}$: as a simple affine transformation, or as a neural network with $2 \sim 4$

		Walking						Eating					
Type	millisecond	80	160	320	400	560	1000	80	160	320	400	560	1000
Best baselines	$\text{Transfer}_{\text{ft}}$	0.44	0.55	0.85	0.95	0.74	1.03	0.61	0.65	0.74	0.78	0.86	1.19
Meta-learning (Ours)	PAML $w/$ init	0.41	0.52	0.79	0.88	0.89	0.93	0.51	0.56	0.70	0.72	0.79	0.95
	PAML $w/$ adapt	0.39	0.55	0.76	0.88	0.92	0.96	0.51	0.62	0.79	0.77	0.82	0.93
	full PAML	**0.36**	**0.49**	**0.73**	**0.85**	**0.81**	**0.88**	**0.37**	**0.54**	**0.66**	**0.73**	**0.76**	**0.83**
		Smoking						Discussion					
Type	millisecond	80	160	320	400	560	1000	80	160	320	400	560	1000
Best baselines	$\text{Transfer}_{\text{ft}}$	0.87	1.02	1.25	1.30	1.45	2.06	0.57	0.82	1.11	1.11	1.37	2.08
Meta-learning (Ours)	PAML $w/$ init	0.56	0.74	0.96	1.12	1.03	1.19	0.55	0.75	1.03	1.07	1.39	1.56
	PAML $w/$ adapt	0.59	0.77	0.87	1.09	1.12	1.13	0.49	0.81	1.15	1.16	1.19	1.32
	full PAML	**0.41**	**0.70**	**0.82**	**1.05**	**1.00**	**1.03**	**0.43**	**0.73**	**1.01**	**1.06**	**1.15**	**1.17**

Table 5.3: Ablation on model initialization vs. model adaptation. Each component by itself outperforms the baselines. Our full PAML consistently achieves the best performance, showing the importance of jointly learning a generic initial model and an effective strategy for adapting this model to novel tasks.

	Walking					
Method	80	160	320	400	560	1000
PAML $w/$ 1-layer, None	0.40	0.52	0.75	0.88	0.86	0.92
PAML $w/$ 2-layer, ReLU	0.40	0.52	0.76	0.87	0.84	0.93
PAML $w/$ 2-layer, Leaky ReLU	0.39	0.50	0.75	0.86	0.85	0.92
PAML $w/$ 3-layer, ReLU	**0.36**	0.51	0.74	0.89	0.83	0.90
PAML $w/$ 3-layer, Leaky ReLU	**0.36**	**0.49**	**0.73**	**0.85**	**0.81**	**0.88**
PAML $w/$ 4-layer, ReLU	0.38	0.53	0.74	0.88	0.84	0.91
PAML $w/$ 4-layer, Leaky ReLU	0.37	0.51	0.75	0.86	0.82	0.90

Table 5.4: Ablation on the structure of $\mathcal{H}$. We vary the number of fully-connected layers in the model adaptation network $\mathcal{H}$, and try both ReLU and leaky ReLU as activation function in the hidden layers. The results show that "3-layer, Leaky ReLU" works best, but in general $\mathcal{H}$ is robust to specific implementation choices.

layers. Since leaky ReLU is used in Chapter 3, we try both ReLU and leakyReLU as activation function in the hidden layers. The results show that a 3-layer fully-connected network with leaky ReLU gives the best prediction performance.

Impact of training sample sizes. In the previous experiments, we focused on a fixed $k = 5$-shot motion prediction task. To test how our meta-learning approach benefits from more training sequences, we evaluate the performance change with respect to the sample size k. Figure 5.3 summarizes the comparisons with fine-tuning transfer, a top performing baseline reported in Table 5.2, when k varies from 1 to 100 at 80ms. As a reference, we also include the *oracle* performance, which is the residual sup. baseline trained on the entire training set of the target activity (*i.e.*, with thousands of annotated sequence pairs). Figure 5.3 shows that our approach consistently outperforms fine-tuning and improves its performance with more and more training sequences. Interestingly, through our meta-learning mechanism,

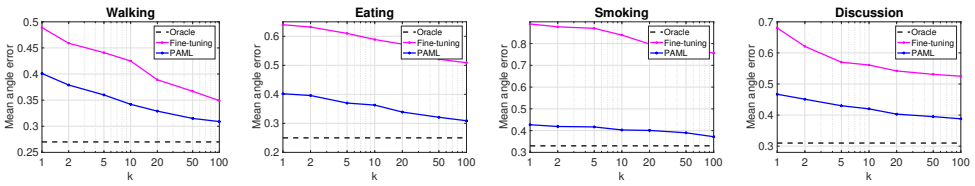


Figure 5.3: Impact of the training sample size k for k-shot motion prediction. We compare our PAML with fine-tuning transfer of residual sup. [254], a top performing baseline. As a reference, we also include the oracle performance, which is residual sup. trained *with thousands of annotated sequence pairs*. X-axis: number of training sequence pairs k per activity. Y-axis: mean angle error. Ours consistently outperform fine-tuning and *with only 100 sequences*, we achieve the performance slightly worse than the oracle.

with only 100 *sequences*, we achieve the performance that is slightly worse than the oracle trained with thousands of sequences.

Part II

Unsupervised Meta-Learning: Towards a Generic Recognition Model

Essentially, all models are wrong, but some are useful.

George Edward Pelham Box; Norman Draper

Chapter 6

Learning Low-Density Separators from Pseudo-Classes

6.1 Motivation

The previous line of work in Part I is promising but still restrictive in the sense that the learned model transformation and dynamics are tied to a specific set of categories due to its supervised nature. To decouple this tie, starting in this chapter, we develop a **large-scale self-supervision** approach to leveraging unsupervised real-world images as meta-data and learning a more generic recognition model.

Let us consider the generality of CNNs, which are typically trained on a particular set of categories (*e.g.*, ImageNet). Recent analysis shows that from bottom, middle, to top layers of the network, features make a transition from general to specific [15, 427]. While features in the bottom and middle layers are fairly generic to many categories (*i.e.*, low-level features of Gabor filters or color blobs and mid-level features of object parts), high-level features in the top layers eventually become specific and biased to best discriminate between this set of chosen categories. The generality of CNNs is thus limited by the *specialization of top layer units* to their original task. With limited samples from target tasks, fine-tuning cannot effectively adjust the units and would result in over-fitting, since it typically requires a significant amount of labeled data. Using off-the-shelf CNNs becomes the best strategy, despite the specialization and reduced performance.

In this chapter, we investigate how to improve pre-trained CNNs for learning novel categories from few examples. Our key insight is to expose multiple top layer units to *a massive set of unlabeled images*, as shown in Figure 6.1, which decouples these units from ties to the original specific set of categories. This additional stage is called **unsupervised meta-training** to distinguish this phase from the conventional unsupervised pre-training phase [130] and the training phase on the target tasks. Based on the above transferability analysis, intuitively, bottom and middle layers construct a feature space with *high-density regions* corresponding to potential latent categories. Top layer units in the pre-trained CNN, however, only have access to those regions associated with the original, observed categories. The units are then tuned to discriminate between these regions by separating the regions while pushing them further away from each other.

To tackle this limitation, our unsupervised meta-training provides a far larger pool of

Figure 6.1: We aim to improve the generality of pre-trained CNNs for the recognition of novel categories from few labeled examples. We perform a multi-stage training procedure: (1) we first pre-train a CNN that recognizes a specific set of categories on a large-scale labeled dataset (*e.g.*, ImageNet 1.2M), which provides fairly generic bottom and middle layer units; (2) we then meta-train the top layers as *low-density separators* on a far larger set of *unlabeled* data (*e.g.*, Flickr 100M), which further improves the generality of multiple top layer units; and (3) finally, we use our modified CNN on new categories/tasks (*e.g.*, scene classification, fine-grained recognition, and action recognition), either as off-the-shelf features or as initialization of fine-tuning that allows for end-to-end training.

unlabeled images as a much less biased sampling in the feature space. Now, instead of producing separations tied to the original categories, we generate diverse sets of separations across the unlabeled data. Since "*the unit tries to discriminate the data manifold from its surroundings in all non-manifold directions*" [35], we capture a more generic and richer description of the visual world [341].

How can we generate these separations in an unsupervised manner? Inspired by the structure/manifold assumption in shallow semi-supervised and unsupervised learning (*i.e.*, the decision boundary should not cross high-density regions, but instead lie in low-density regions) [27,56], we introduce a *low-density separator* (LDS) module that can be plugged into any (or all) top layers of a standard CNN architecture. More precisely, the vector of weights connecting a unit to its previous layer (together with the non-linearity) can be viewed as a separator or decision boundary in the activation space of the previous layer. LDS then generates connection weights (decision boundaries) between successive layers that traverse regions of as low density as possible and avoid intersecting high-density regions in the activation space. Many LDS methods typically infer a probability distribution, for example through densest region detection, lowest-density hyperplane estimation [27], and clustering [165]. However, exact clustering or density estimation is known to be notoriously difficult in high-dimensional spaces.

We instead adopt a discriminative paradigm [56,67,309,398] to circumvent the aforementioned difficulties. Using a max-margin framework, we propose an *unsupervised, scalable, coarse-to-fine* approach that jointly estimates compact, distinct high-density *pseudo-classes* (HDQC), *i.e.*, sets of data points sampled in high-density regions, as stand-ins for plausible high-density regions and infers low-density hyperplanes (separators). Our decoupled formulations generalize those in supervised binary code discovery [309] and semi-supervised learning [67], respectively; and more crucially, we propose a novel *combined* optimization to *jointly* estimate HDQC and learn LDS *in large-scale unsupervised* scenarios, from the labeled

ImageNet 1.2M [321] to the unlabeled Flickr 100M dataset [364].

Our approach of exploiting unsupervised learning on top of CNN transfer learning is unique as opposed to other recent work on unsupervised, weakly-supervised, and semi-supervised deep learning. Most existing unsupervised deep learning approaches focus on unsupervised learning of visual representations that are both sparse and allow image reconstruction [130], including deep belief networks (DBN), convolutional sparse coding, and (denoising) auto-encoders (DAE). Our unsupervised LDS meta-training is different from conventional unsupervised pre-training as in DBN and DAE in two important ways: (1) our meta-training "post-arranges" the network that has undergone supervised training on a labeled dataset and then serves as a kind of network "pre-conditioner" [130] for the target tasks; and (2) our meta-training phase is not necessarily followed by fine-tuning and the features obtained by meta-training could be used off the shelf.

Other types of supervisory information (by creating auxiliary tasks), such as clustering, surrogate classes [95, 155], spatial context, temporal consistency, web supervision, and image captions [190], have been explored to train CNNs in an unsupervised (or weakly-supervised) manner. Although showing initial promise, the performance of these unsupervised (or weakly-supervised) deep models is still not on par with that of their supervised counterparts, partially due to noisy or biased external information [190]. In addition, our LDS, if viewed as an auxiliary task, is directly related to discriminative classification, which results in more desirable and consistent features for the final novel-category recognition tasks. Unlike using a single image and its pre-defined transformations [95] or other labeled multi-view object [155] to simulate a surrogate class, our pseudo-classes capture a more natural representation of realistic images. Finally, while we boost the overall generality of CNNs for a wide spectrum of unseen categories, semi-supervised deep learning approaches typically improve the model generalization for specific tasks, with both labeled and unlabeled data coming from the tasks of interest [4, 408].

Our contributions are three-fold. (1) We first show how LDS, based on an unsupervised margin maximization, is generated without a bias to a particular set of categories. (2) We detail how to use LDS modules in CNNs by plugging them into any (or all) top layers of the architecture, leading to single-scale (or multi-scale) low-density separator networks. (3) We finally show how such modified CNNs, with enhanced generality, are used to facilitate the recognition of novel categories from few examples and significantly improve the performance in scene classification, fine-grained recognition, and action recognition. The general setup is depicted in Figure 6.1.

6.2 Pre-Trained Low-Density Separators from Unsupervised Data

Given a CNN architecture pre-trained on a specific set of categories, such as the ImageNet (ILSVRC) $1{,}000$ categories, we aim to improve the generality of one of its top layers, *e.g.*, the k-th layer. We fix the structures and weights of the layers from 1 to $k-1$, and view the activation of layer $k-1$ as a feature space. A unit s in layer k is fully connected to all the units in layer $k-1$ via a vector of weights $\boldsymbol{w}^s$. Each $\boldsymbol{w}^s$ corresponds to a particular decision boundary (partition) of the feature space. Intuitively, all the $\boldsymbol{w}^s$'s then jointly further discriminate between these $1{,}000$ categories, enforcing that the new activations in layer k are more similar within classes and more dissimilar between classes.

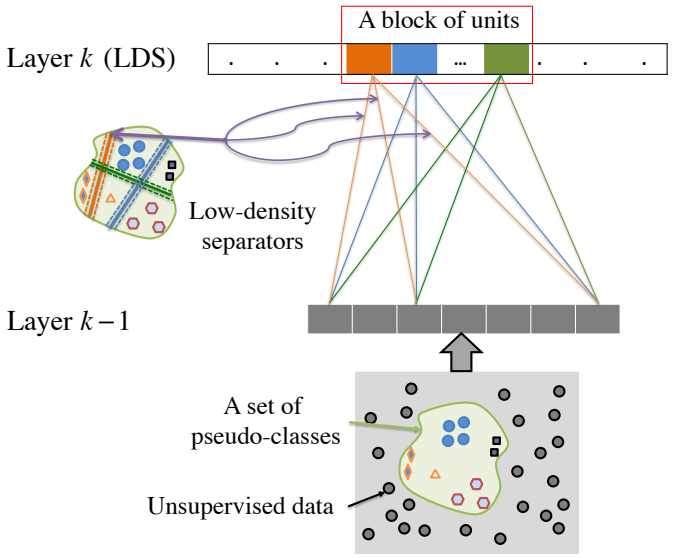


Figure 6.2: Illustration of learning low-density separators between successive layers on a large amount of unlabeled data. Note the color correspondence between the decision boundaries across the unlabeled data and the connection weights in the network.

To make $\boldsymbol{w}^s$'s and the associated units in layer k unspecific to the ImageNet 1,000 categories, we use a large amount of *unlabeled* images at the unsupervised meta-training stage. The layers from 1 to $k-1$ remain unchanged, which means that we still tackle the same feature space. The new unlabeled images now constitute a less biased sampling of the feature space in layer $k-1$. We introduce a new k-th layer with more units and encourage their unbiased exploration of the feature space. More precisely, we enforce that the units learn many diverse decision boundaries $\boldsymbol{w}^s$'s that traverse different low-density regions while avoiding intersecting high-density regions of the unsupervised data (untied to the original ImageNet categories). The set of possible arrangements of such decision boundaries is rich, meaning that we can potentially generalize to a broad range of categories.

6.2.1 Approach Overview

For each unlabeled image $\mathcal{I}_i$, where $i \in \{1, 2, \ldots, N\}$, let $\boldsymbol{x_i} \in \mathbb{R}^D$ and $\boldsymbol{\phi}_i \in \mathbb{R}^S$ be the vectorized activations in layers $k-1$ and k, respectively. Let $\boldsymbol{W}$ be the weights between the two layers, where $\boldsymbol{w}^s$ is the weight vector associated with the unit s in layer k. For notational simplicity, $\boldsymbol{x_i}$ already includes a constant 1 as the last element and $\boldsymbol{w^s}$ includes the bias term. We then have $\phi_i^s = f\left(\boldsymbol{w^s}^T \boldsymbol{x_i}\right)$, where $f(\cdot)$ is a non-linear function, such as sigmoid or ReLU. The resulting activation spaces of layers $k-1$ and k are denoted as $\mathcal{X}$ and $\mathcal{F}$, respectively.

To learn $\boldsymbol{w}^s$'s as low-density separators, we are supposed to have certain high-density regions which $\boldsymbol{w}^s$'s separate. However, accurate estimation of high-density regions is difficult. We instead generate pseudo-classes as stand-ins for plausible high-density regions. We want samples with the same pseudo-labels to be similar in activation spaces (constraint within pseudo-classes), while those with different pseudo-labels should be very dissimilar

in activation spaces (constraints between pseudo-classes). Note that in contrast to clustering, generating pseudo-classes does not require inferring membership for each data point. Formally, assuming that there are C desired pseudo-classes, we introduce a sample selection vector $\boldsymbol{T}_c \in \{0,1\}^N$ for each pseudo-class c. $T_{c,i} = 1$ if $\mathcal{I}_i$ is selected for assignment to pseudo-class c and zero otherwise. As illustrated in Figure 6.2, the optimization for seeking low-density separators (LDS) while identifying high-density pseudo-classes (HDPC) can be framed as

$$\begin{aligned} &\text{find } \boldsymbol{W} \in \text{LDS},\ \boldsymbol{T} \in \text{HDPC} \\ &\text{subject to } \boldsymbol{W} \text{ separate } \boldsymbol{T}. \end{aligned} \tag{6.1}$$

This optimization problem enforces that each unit s learns a partition $\boldsymbol{w}^s$ lying across the low-density region among certain salient high-density pseudo-classes discovered by $\boldsymbol{T}$. This leads to a difficult joint optimization problem in theory, because $\boldsymbol{W}$ and $\boldsymbol{T}$ are interdependent.

In practice, however, it may be unnecessary to find the global optimum. Reasonable local optima are sufficient in our case to describe the feature space, as shown by the empirical results in Section 6.4. We use an iterative approach that obtains salient high-density pseudo-classes from coarse to fine (Section 6.2.3) and produces promising discriminative low-density partitions among them (Section 6.2.2). We found that the optimization procedures converge in our experiments.

6.2.2 Learning Low-Density Separators

Assume that $\boldsymbol{T}$ is known, which means that we have already defined C high-density pseudo-classes by $\boldsymbol{T}_c$. We then use a max-margin formulation to learn $\boldsymbol{W}$. Each unit s in layer k corresponds to a low-density hyperplane $\boldsymbol{w}^s$ that separates positive and negative examples in a max-margin fashion. To train $\boldsymbol{w}^s$, we need to generate label variables $\boldsymbol{l}^s \in \{-1,1\}$ for each $\boldsymbol{w}^s$, which label the samples in the pseudo-classes either as positive (1) or negative (-1) training examples. We can stack all the labels for learning $\boldsymbol{w}^s$'s to form $\boldsymbol{L} = \left[\boldsymbol{l}^1, \ldots, \boldsymbol{l}^S\right]$. Moreover, in the activation space $\mathcal{F}$ of layer k, which is induced by the activation space $\mathcal{X}$ of layer $k-1$ and $\boldsymbol{w}^s$, it would be beneficial to further push for large inter-pseudo-class and small intra-pseudo-class distances. We achieve such properties by optimizing

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{L},\boldsymbol{\Phi}} & \sum_{s=1}^{S} \|\boldsymbol{w}^s\|^2 + \eta \sum_{i=1}^{N} \sum_{s=1}^{S} I_i \left[1 - l_i^s \left(\boldsymbol{w}^{s T} \boldsymbol{x}_i\right)\right]_+ \\ & + \frac{\lambda_1}{2} \sum_{c=1}^{C} \sum_{\substack{u=1 \\ v=1}}^{N} T_{c,u} T_{c,v} d\left(\boldsymbol{\phi}_u, \boldsymbol{\phi}_v\right) - \frac{\lambda_2}{2} \sum_{c'=1}^{C} \sum_{\substack{c''=1 \\ c'' \neq c'}}^{C} \sum_{\substack{p=1 \\ q=1}}^{N} T_{c',p} T_{c'',q} d\left(\boldsymbol{\phi}_p, \boldsymbol{\phi}_q\right), \end{aligned} \tag{6.2}$$

where d is a distance metric (*e.g.*, square of Euclidean distance) in the activation space $\mathcal{F}$ of layer k. $[x]_+ = max\,(0, x)$ represents the hinge loss. Here we introduce an additional indicator vector $\boldsymbol{I} \in \{0,1\}^N$ for all the pseudo-classes. $I_i = 0$ if $\mathcal{I}_i$ is not selected for assignment to any pseudo-class (*i.e.*, $\sum_{c=1}^{C} T_{c,i} = 0$) and one otherwise. Note that $\boldsymbol{I}$ is actually sparse, since only a portion of unlabeled samples are selected as pseudo-classes and only their memberships are estimated in $\boldsymbol{T}$.

The new objective is much easier to optimize compared to Eqn. (6.1), as it only requires producing the low-density separators $\boldsymbol{w}^s$ from known pseudo-classes given $\boldsymbol{T}_c$. We then

derive an algorithm to optimize problem (6.2) using block coordinate descent. Specifically, problem (6.2) can be viewed as a generalization of predictable discriminative binary codes in [309]: (1) compared with the fully labeled case in [309], Eqn. (6.2) introduces additional pseudo-class indicator variables to handle the unsupervised scenario; and (2) Eqn. (6.2) extends the specific binary-valued hash functions in [309] to general real-valued non-linear activation functions in neural networks.

We adopt a similar iterative optimization strategy as in [309]. To achieve a good local minimum, our insight is that there should be diversity in $\boldsymbol{w}^s$'s and we thus initializ $\boldsymbol{w}^s$'s as the top-S orthogonal directions of PCA on data points belonging to the pseudo-classes. We found that this initialization yields promising results that work better than random initialization and do not contaminate the pre-trained CNNs. For fixed $\boldsymbol{W}$, we update $\boldsymbol{\Phi}$ using stochastic gradient descent to achieve improved separation in the activation space $\mathcal{F}$ of layer k. This optimization is efficient if using ReLU as non-linearity. We use $\boldsymbol{\Phi}$ to update $\boldsymbol{L}$. $l_i^s = 1$ if $\phi_i^s > 0$ and zero otherwise. Using $\boldsymbol{L}$ as training labels, we then train S linear SVMs to update $\boldsymbol{W}$. We iterate this process a fixed number of times—$2 \sim 4$ in practice, and we thus obtain the low-density separator $\boldsymbol{w}^s$ for each unit and construct the activation space $\mathcal{F}$ of layer k.

6.2.3 Generating High-Density Pseudo-Classes

In the previous section, we assumed $\boldsymbol{T}$ known and learned low-density separators between high-density pseudo-classes. Now we explain how to find these pseudo-classes. Given the activation space $\mathcal{X}$ of layer $k-1$ and the activation space $\mathcal{F}$ of layer k (linked by the low-density separators $\boldsymbol{W}$ as weights), we need to generate C high-density pseudo-classes from the unlabeled data selected by $\boldsymbol{T}_c$. We hope that the pseudo-classes are distinct and compact in the activation spaces. That is, we want samples belonging to the same pseudo-classes to be close to each other in the activation spaces, while samples from different pseudo-classes should be far from each other in the activation spaces. To this end, we propose a coarse-to-fine procedure that combines the seeding heuristics of K-means++ [77] and a max-margin formulation [67] to gradually augment confident samples into the pseudo-classes. We suppose that each pseudo-class contains at least τ_0 images and at most τ images. Learning $\boldsymbol{T}$ includes the following steps:

Skeleton generation. We first choose a single seed point $T_{c,i_c} = 1$ for each pseudo-class using the K-means++ heuristics in the activation space $\mathcal{X}$ of layer $k-1$. All the seed points are now spread out as the skeleton of the pseudo-classes.

Pseudo-class initialization. We extend each single skeletal point to an initial pseudo-class by adding its nearest neighbors [75] in the activation space $\mathcal{X}$ of layer $k-1$. Each of the resulting pseudo-classes thus contains τ_0 images, which satisfies the constraint for the minimum number of selected samples.

Augmentation and refinement. In the above two steps, we select samples for pseudo-classes based on the similarity in the activation space of layer $k-1$. Given this initial estimate of pseudo-classes, we select additional samples using joint similarity in both activation spaces of layers $k-1$ and k by leveraging a max-margin formulation. For each pseudo-class c, we construct pseudo-class classifiers $\boldsymbol{h}_c^{\mathcal{X}}$ and $\boldsymbol{h}_c^{\mathcal{F}}$ in the two activation spaces. Note that $\boldsymbol{h}_c^{\mathcal{X}}$ and $\boldsymbol{h}_c^{\mathcal{F}}$ are different from the low-density separator $\boldsymbol{w}^s$. We use SVM responses to select

additional samples, leading to the following optimization:

$$\min_{\boldsymbol{T},\boldsymbol{h}_c^{\mathcal{X}},\boldsymbol{h}_c^{\mathcal{F}}} \alpha \sum_{c=1}^{C} \left(\left\| \boldsymbol{h}_c^{\mathcal{X}} \right\|_2^2 + \lambda_{\mathcal{X}} \sum_{i=1}^{N} I_i \left[1 - y_{c,i} \left(\boldsymbol{h}_c^{\mathcal{X}^T} \boldsymbol{x}_i \right) \right]_+ \right) + \sum_{c'=1}^{C} \sum_{\substack{c''=1 \\ c' \neq c''}}^{C} \sum_{j=1}^{N} T_{c',j} T_{c'',j}$$
$$+\beta \sum_{c=1}^{C} \left(\left\| \boldsymbol{h}_c^{\mathcal{F}} \right\|_2^2 + \lambda_{\mathcal{F}} \sum_{i=1}^{N} I_i \left[1 - y_{c,i} \left(\boldsymbol{h}_c^{\mathcal{F}^T} \boldsymbol{\phi}_i \right) \right]_+ - \sum_{j=1}^{N} T_{c,j} \left(\boldsymbol{h}_c^{\mathcal{F}^T} \boldsymbol{\phi}_j \right) \right)$$
$$s.t. \tau_0 \leq \sum_{i=1}^{N} T_{c,i} \leq \tau, \forall c \in \{1, \ldots, C\}, \tag{6.3}$$

where $y_{c,i}$ is the corresponding binary label used for one-vs.-all multi-pseudo-class classification: $y_{c,i} = 1$ if $T_{c,i} = 1$ and -1 otherwise. The first and second terms denote a max-margin classifier in the activation space $\mathcal{X}$, and the fourth and fifth terms denote a max-margin classifier in the activation space $\mathcal{F}$. The third term ensures that the same unlabeled sample is not shared by multiple pseudo-classes. The last term is a sample selection criterion that chooses those unlabeled samples with high classifier responses in the activation space $\mathcal{F}$.

This formulation is inspired by the approach to selecting unlabeled images using joint visual features and attributes [67]. We view our activation space $\mathcal{X}$ of layer k–1 as the feature space, and the activation space $\mathcal{F}$ of layer k as the learned attribute space. However, different from the semi-supervised scenario in [67], which provides an initially labeled training images, our problem (6.3) is entirely unsupervised. To solve it, we use initial $\boldsymbol{T}$ corresponding to the pseudo-classes obtained in the first two steps to train $\boldsymbol{h}_c^{\mathcal{X}}$ and $\boldsymbol{h}_c^{\mathcal{F}}$. After obtaining these two sets of SVMs in both activation spaces, we update $\boldsymbol{T}$. Following a similar block coordinate descent procedure as in [67], we iteratively re-train both $\boldsymbol{h}_c^{\mathcal{X}}$ and $\boldsymbol{h}_c^{\mathcal{F}}$ and update $\boldsymbol{T}$ until we obtain the desired τ number of samples.

6.3 Low-Density Separator Networks

6.3.1 Single-Scale Layer-Wise Training

We start from how to embed our LDS as a new top layer into a standard CNN structure, leading to single-scale network. To improve the generality of the learned units in layer k, we need to prevent co-adaptation and enforce diversity between these units [130, 427]. We adopt a simple random sampling strategy to train the entire LDS layer. We break the units in layer k into (disjoint) blocks, as shown in Figure 6.2. We encourage each block of units to explore different regions of the activation space described by a random subset of unlabeled samples. This sampling strategy also makes LDS learning scalable since direct LDS learning from the entire dataset is computationally infeasible.

Specifically, from an original selection matrix $\boldsymbol{T}_0 \in \{0,1\}^{N \times C}$ of all zeros, we first obtain a random sub-matrix $\boldsymbol{T} \in \{0,1\}^{M \times C}$. Using this subset of M samples, we then generate C high-density pseudo-classes by solving the problem (6.3) and learn S corresponding low-density separator weights by solving the problem (6.2), yielding a block of S units in layer k. We randomly produce J sub-matrices $\boldsymbol{T}$, repeat the procedure, and obtain $S \times J$ units (J blocks) in total. This thus constitutes layer k, the low-density separator layer. The entire single-scale structure is shown in Figure 6.3a.

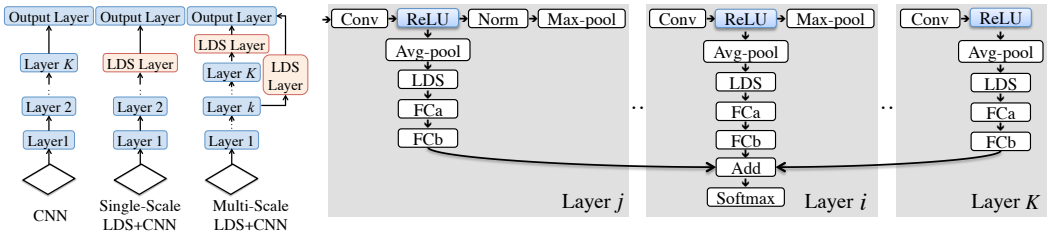


Figure 6.3: We use our LDS to revisit CNN architectures. In Figure 6.3a, we embed LDS learned from a large collection of unlabeled data as a new top layer into a standard CNN structure pre-trained on a specific set of categories (left), leading to single-scale LDS+CNN (middle). LDS could be also embedded into different layers, resulting multi-scale LDS+CNN (right). More specifically in Figure 6.3b, our multi-scale LDS+CNN architecture is constructed by introducing LDS layers into multi-scale DAG-CNN [422]. For each scale (level), we spatially (average) pool activations, learn and plug in LDS in this activation space, add fully-connected layers FCa and FCb (with K outputs), and finally add the scores across all layers as predictions for K output classes (that are finally soft-maxed together) on the target task. We show that the resulting LDS+CNNs can be either used as off-the-shelf features or discriminatively trained in an end-to-end fashion to facilitate novel category recognition.

6.3.2 Multi-Scale Structure

For a convolutional layer of size $H_1 \times H_2 \times F$, where H_1 is the height, H_2 is the width, and F is the number of filter channels, we first compute a $1 \times 1 \times F$ pooled feature by averaging across spatial dimensions as in [422], and then learn LDS in this activation space as before. Note that our approach applies to other types of pooling operation as well. Given the benefit of complementary features, LDS could also be operationalized on several different layers, leading to multi-scale/level representations. We thus modify the multi-scale DAG-CNN architecture [422] by introducing LDS on top of the ReLU layers, leading to multi-scale LDS+CNN, as shown in Figure 6.3b. We add two additional layers on top of LDS: FCa (with F outputs) that selects discriminative units for target tasks, and FCb (with K outputs) that learns K-way classifier for target tasks. The output of the LDS layers could be used as off-the-shelf multi-scale features. If using LDS weights as initialization, the entire structure in Figure 6.3b could also be fine-tuned in a similar fashion as DAG-CNN [422].

6.4 Experimental Evaluation

In this section, we explore the use of low-density separator networks (LDS+CNNs) on a number of supervised learning tasks with limited data, including scene classification, fine-grained recognition, and action recognition. We use two CNN models: AlexNet [212] and VGG19 [347] pre-trained on ILSVRC 2012 [321], as our reference networks. We implement the unsupervised meta-training on Yahoo! Flickr Creative Commons100M dataset (YFCC100M) [364], which is the largest single publicly available image and video database. We begin with plugging LDS into a single layer, and then introduce LDS into several top

layers, leading to a multi-scale model. We consider using LDS+CNNs as off-the-shelf features in the small sample size regime, as well as through fine-tuning when enough data is available in the target task.

Implementation details. During unsupervised meta-training, we use 99.2 million unlabeled images on YFCC100M [364]. After resizing the smallest side of each image to be 256, we generate the standard 10 crops (4 corners plus one center and their flips) of size 224×224 as implemented in Caffe [185]. For single-scale structures, we learn LDS in the $fc7$ activation space of dimension $4{,}096$. For multi-scale structures, following [422] we learn LDS in activation spaces of $Conv3$, $Conv4$, $Conv5$, $fc6$, and $fc7$ for AlexNet, and we learn LDS in activation spaces of $Conv43$, $Conv44$, $Conv51$, $Conv52$, and $fc6$ for VGG19. We use the same sets of parameters to learn LDS in these activation spaces without further tuning. In the LDS layer, each block has $S = 10$ units, which separate across $M = 20{,}000$ randomly sub-sampled data points. Repeating $J = 2{,}000$ sub-sampling, we then have $20{,}000$ units in total. Notably, each block of units in the LDS layer can be learned independently, making feasible for parallelization. For learning LDS in Eqn. (6.2), η and λ_1 are set to 1 and λ_2 is set to normalize for the size of pseudo-classes, which is the same setup and default parameters as in [309]. For generating high-density pseudo-classes in Eqn. (6.3), following [67,75], we set the minimum and maximum number of selected samples per pseudo-classes to be $\tau_0 = 6$ and $\tau = 56$, and produce $C = 30$ pseudo-classes in total. We use the same setup and parameters as in [67], where $\alpha = 1$, $\beta = 1$. While using only the center crops to infer pseudo-classes, we use all 10 crops to learn more accurate LDS.

Tasks and datasets. Similar as before, we evaluate on standard benchmark datasets for scene classification: SUN-397 [414] and MIT-67 [375], fine-grained recognition: Oxford 102 Flowers [279], and action recognition (compositional semantic recognition): Stanford-40 actions [425]. We follow the standard experimental setup (*e.g.*, the train/test splits) for these datasets as before. For example, on SUN-397, a subset of the dataset with 50 training and 50 test images per class are used for evaluation, averaging over 10 fixed and publicly available partitions.

6.4.1 Learning from Few Examples

The first question to answer is whether the LDS layers improve the transferability of the original pre-trained CNNs and facilitate the recognition of novel categories from few examples. To answer this question, we evaluate both LDS+CNN and CNN as off-the-shelf features without fine-tuning on the target datasets. This is the standard way to use pre-trained CNNs [311]. We test how performance varies with the number of training samples per category as in [395]. To make extensive comparisons with publicly available baselines, we use VGG19 in this set of experiments. Following the standard practice, we train simple linear SVMs in one-vs.-all fashion on $\mathcal{L}_2$-normalized features [311,422] in Liblinear [101].

Single-scale features. We begin by evaluating single-scale features on theses datasets. For a fair comparison, we first reduce the dimensionality of LDS+CNN from $20{,}000$ to $4{,}096$, the same dimensionality as CNN, followed by linear SVMs. This is achieved by selecting from LDS+CNN the $4{,}096$ most active features according to the standard criterion of multi-class recursive feature elimination (RFE) [38] using the target dataset. We also tested PCA. The performance drops, but it is still significantly better than the pre-trained CNN. Figure 6.4 summarizes the average performance over 10 random splits on these datasets. When used as off-the-shelf features for small-sample learning, our single-scale LDS+CNN significantly outperforms the vanilla pre-trained CNN, which is already a strong baseline. Our

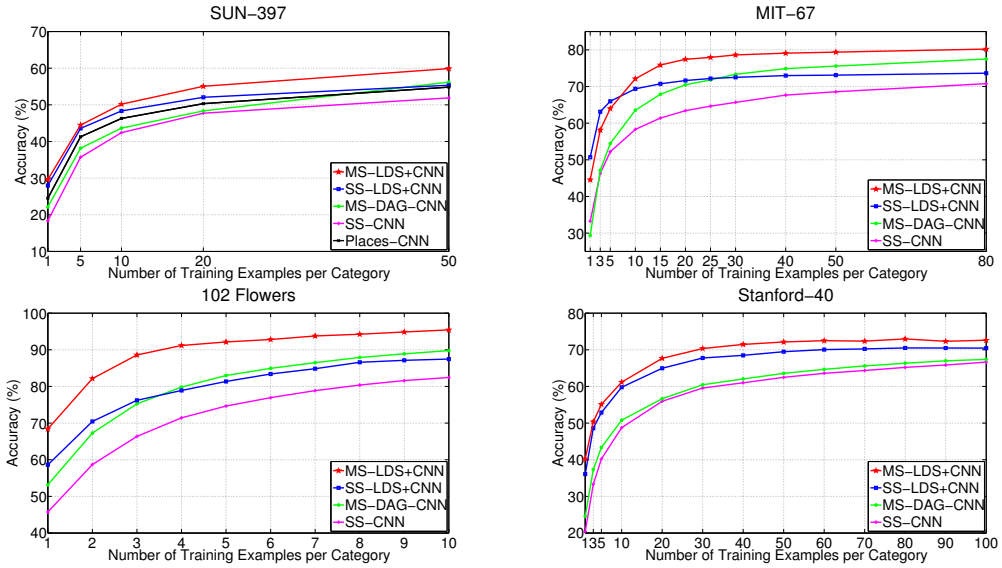


Figure 6.4: Performance comparisons between our single-scale LDS+CNN (SS-LDS+CNN), multi-scale LDS+CNN (MS-LDS+CNN) and the pre-trained single-scale CNN (SS-CNN), multi-scale DAG-CNN (MS-DAG-CNN) baselines for scene classification, fine-grained recognition, and action recognition from few labeled examples on four benchmark datasets. VGG19 [347] is used as the CNN model for its demonstrated superior performance. For SUN-397, we also include a publicly available strong baseline, Places-CNN, which trained a CNN (AlexNet architecture) from scratch using a scene-centric database with over 7 million annotated images from 400 scene categories, and which achieved state-of-the-art performance for scene classification [437]. X-axis: number of training examples per class. Y-axis: average multi-class classification accuracy. With improved transferability gained from a large set of unlabeled data, our LDS+CNNs with simple linear SVMs significantly outperform the vanilla pre-trained CNN and powerful DAG-CNN for small sample learning.

results are particularly impressive for the big performance boost, for example nearly 20% on MIT-67, in the one-shot learning scenario. This verifies the effectiveness of the layer-wise LDS, which leads to a more generic representation for a broad range of novel categories.

Multi-scale features. Given the promise of single-scale LDS+CNN, we now evaluate multi-scale off-the-shelf features. After learning LDS in each activation space separately, we reduce their dimensionality to that of the corresponding activation space via RFE for a fair comparison with DAG-CNN [422]. We train linear SVMs on these LDS+CNNs, and then average their predictions. Figure 6.4 summarizes the average performance over different splits for multi-scale features. Consistent with the single-scale results, our multi-scale LDS+CNN outperforms the powerful multi-scale DAG-CNN. LDS+CNN is especially beneficial to fine-grained recognition, since there is typically limited data per class for fine-grained categories. Figure 6.4 also validates that multi-scale LDS+CNN allows for transfer at different levels, thus leading to better generalization to novel recognition tasks compared to its single-scale counterpart. In addition, Table 6.1 further shows that our LDS+CNNs outperform weakly-supervised CNNs [190] that are directly trained on Flickr using external caption information.

Type	Approach	SUN-397	MIT-67	102 Flowers	Stanford-40
Weakly-supervised CNNs	Flickr-AlexNet	42.7	55.8	74.2	53.0
	Flickr-GoogLeNet	44.4	55.6	65.8	52.8
	Combined-AlexNet	47.3	58.8	83.3	56.4
	Combined-GoogLeNet	55.0	67.9	83.7	69.2
Ours	SS-LDS+CNN	55.4	73.6	87.5	70.5
	MS-LDS+CNN	**59.9**	**80.2**	**95.4**	**72.6**

Table 6.1: Performance comparisons of classification accuracy (%) between our LDS+CNNs and weakly-supervised CNNs [190] on the four datasets when using the entire training sets. In contrast to our approach that uses the Flickr dataset for unsupervised meta-training, Flickr-AlexNet/GoogLeNet train CNNs from scratch on the Flickr dataset by using associated captions as weak supervisory information. Combined-AlexNet/GoogLeNet concatenate features from supervised ImageNet CNNs and weakly-supervised Flickr CNNs. Despite the same amount of data used for pre-training, ours outperform the weakly-supervised CNNs by a significant margin due to their noisy captions and tags.

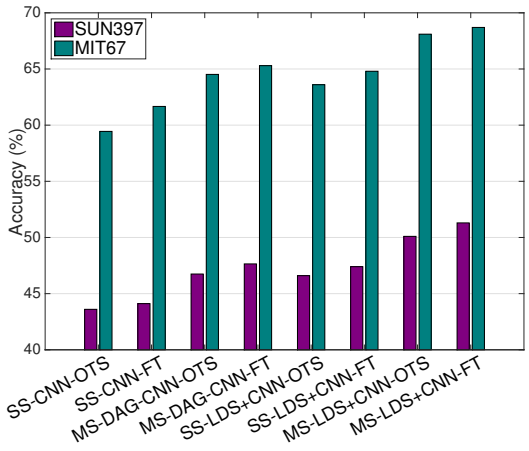


Figure 6.5: Effect of fine-tuning (FT) on SUN-397 (purple bars) and MIT-67 (blue bars). Fine-tuning LDS+CNNs (AlexNet) further improves the performance over the off-the-shelf (OTS) features for novel category recognition.

6.4.2 Fine-Tuning

With more training data available in the target task, our LDS+CNNs could be fine-tuned to further improve the performance. For efficient and easy fine-tuning, we use AlexNet in this set of experiments as in [422]. We evaluate the effect of fine-tuning of our single-scale and multi-scale LDS+CNNs in the scene classification tasks, due to their relatively large number of training samples. We compare against the fine-tuned single-scale CNN and multi-scale DAG-CNN [422], as shown in Figure 6.5. For completeness, we also include their off-the-shelf performance. As expected, fine-tuned models consistently outperform their off-the-

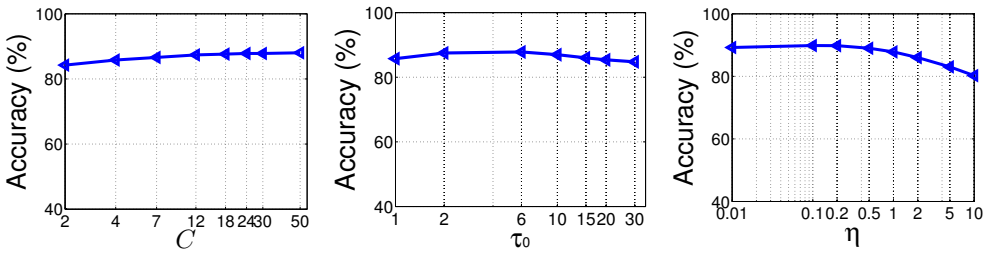


Figure 6.6: Representative hyper-parameter sensitivity experiment of single-scale LDS+CNN w.r.t. C pseudo-classes, τ_0 initial samples per pseudo-class, and η with other hyper-parameters fixed on 102 Flowers for the case of 10 training examples per class. There is a fairly smooth and flat region around Acc $= 87\%$.

shelf counterparts. Importantly, Figure 6.5 shows that our approach is not limited to small-sample learning and is still effective *even in the many training examples regime*.

6.5 Experimental Analysis and Visualization

In this section, we first provide hyper-parameter sensitivity analysis experiments, which show that the hyper-parameters have a wide range of reasonable values to choose from. The pseudo-class and feature visualizations further show the generality of our approach.

6.5.1 Hyper-Parameter Sensitivity Analysis

As mentioned in Section 6.4, we used the default hyper-parameters in [67,75,77,309]. More crucially, most of these hyper-parameters themselves in [67, 309] are set without tuning, where $\eta = 1$, $\alpha = 1$, $\beta = 1$, $\lambda_1 = 1$, for instance. Although we did not tune them either, our approach already significantly outperformed the baselines and could be even better with optimized hyper-parameters.

For the other hyper-parameters (*e.g.*, C, τ_0, τ), a series of sensitivity analysis experiments conducted in [67, 75, 77] (*e.g.*., Figure 6 in [77] and Figure 3 in [67]) show that "each of the hyper-parameters has a wide range of reasonable values to choose from" [77]).

In addition, the visualization of *randomly sampled* pseudo-classes in Section 6.5.2 and the recognition performance on the target tasks show that our approach is robust when generating *a large collection* of pseudo-classes and learning LDS. Such *large-scale ensemble behavior* substantially mitigates or even cancels out the randomization and deficiency of individual pseudo-classes. This is consistent with a similar observation in [77].

We also provide a representative sensitivity analysis experiment in Figure 6.6, which shows that there is a fairly smooth and flat region around Accuracy $= 87\%$.

6.5.2 Pseudo-Classes Visualization

To better understand LDS+CNNs, we provide qualitative visualization of the pseudo-classes in Figures 6.8, 6.9, and 6.10. In each figure, we show a set of pseudo-classes, which are sampled *randomly* from the entire pool of generated pseudo-classes. These figures show that

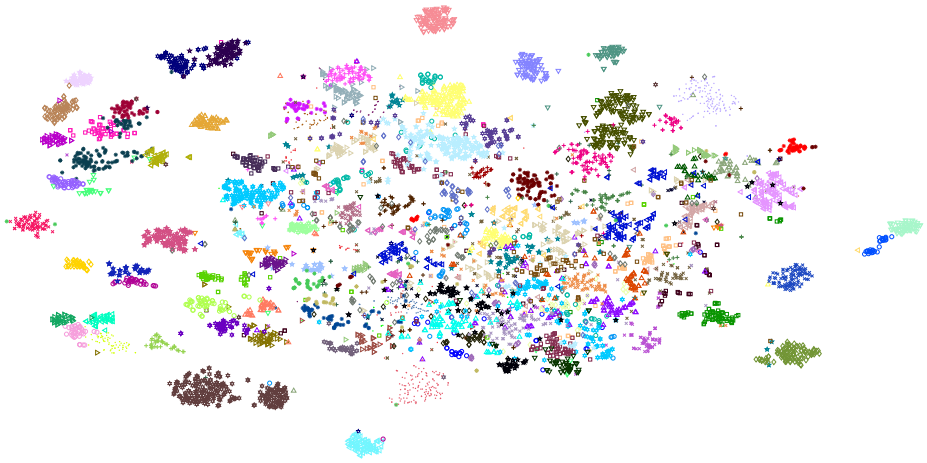


Figure 6.7: t-SNE feature visualization [384] of single-scale LDS+CNN on the 102 Flowers training and testing dataset. (We use both the training and testing data for better visualization due to the limited training data on 102 Flowers.) Although low-density separators are generated in an unsupervised fashion, it is interesting to note their generalization to novel categories when considering semantic groupings of labels. **Best viewed in color with zoom.**

our approach identifies diverse pseudo-classes with a wide coverage from large amounts of unlabeled data.

6.5.3 Feature Visualization

By using the t-SNE algorithm [384], we find a 2-dimensional embedding of the single-scale LDS+CNN features, and visualize them as points colored depending on their categories. Figures 6.7 shows the visualization on the 102 Flowers dataset. It again demonstrates the generality of our LDS+CNNs for novel categories.

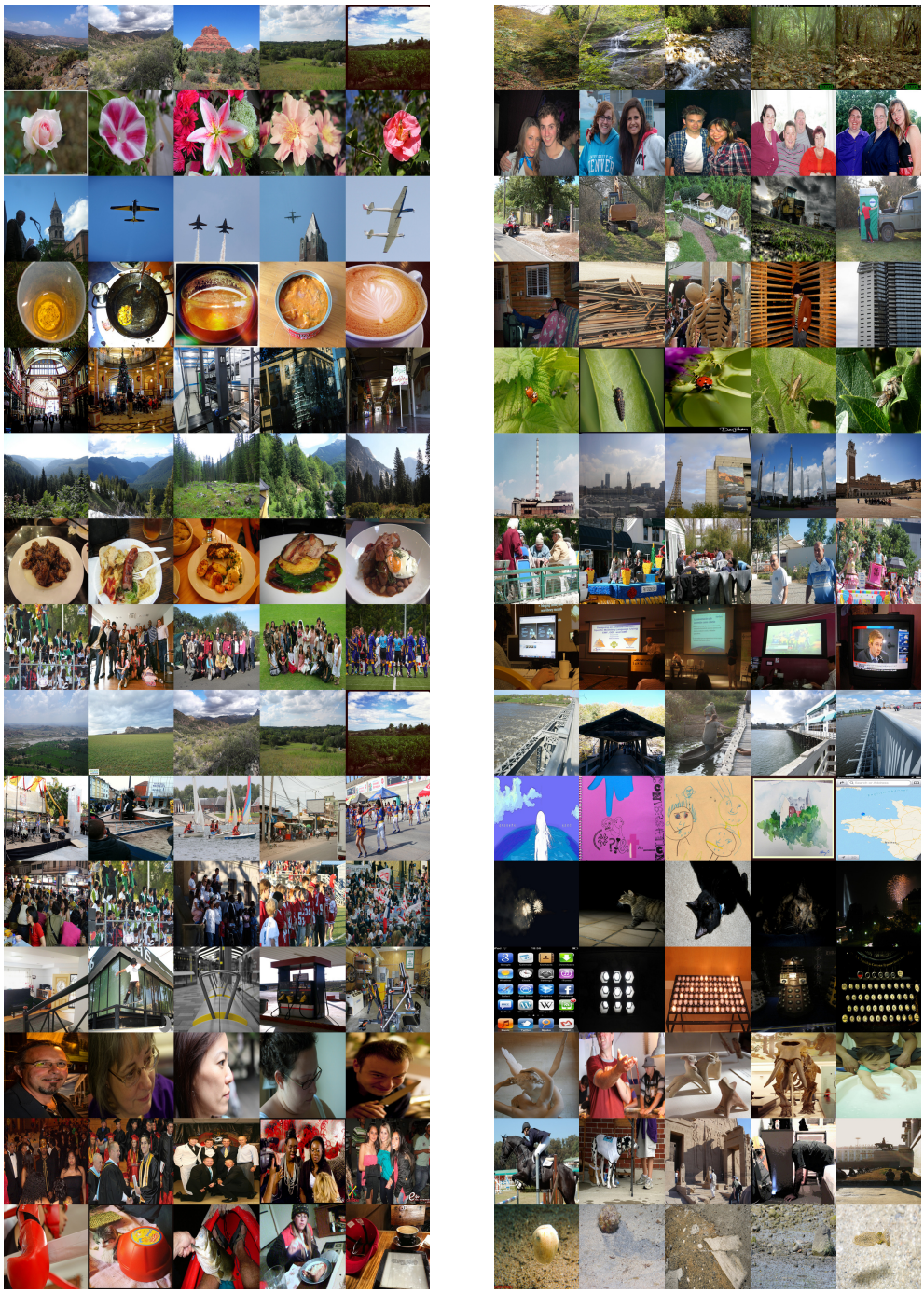


Figure 6.8: Example pseudo-classes visualization I. This set of pseudo-classes were sampled randomly from the entire pool of generated pseudo-classes (these are not curated). There are $C = 30$ pseudo-classes in total for each random sub-sampling. Each row contains representative images from two pseudo-classes, respectively. Viewing digitally with zoom is recommended.

Figure 6.9: Example pseudo-classes visualization II. This set of pseudo-classes were sampled randomly from the entire pool of generated pseudo-classes (these are not curated). There are $C = 30$ pseudo-classes in total for each random sub-sampling. Each row contains representative images from two pseudo-classes, respectively. Viewing digitally with zoom is recommended.

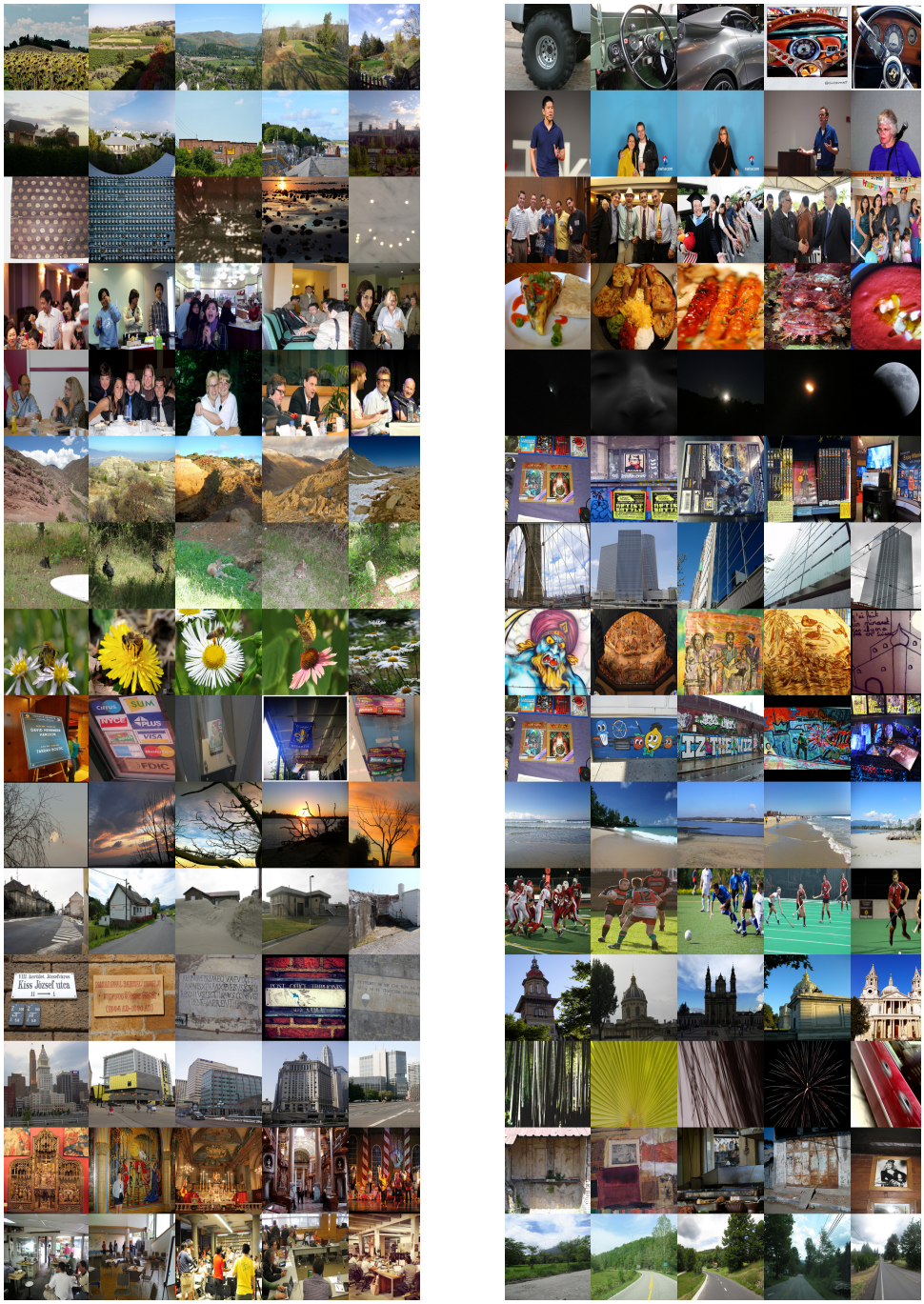


Figure 6.10: Example pseudo-classes visualization III. This set of pseudo-classes were sampled randomly from the entire pool of generated pseudo-classes (these are not curated). There are $C = 30$ pseudo-classes in total for each random sub-sampling. Each row contains representative images from two pseudo-classes, respectively. Viewing digitally with zoom is recommended.

Chapter 7

Extension to Object Detection via Model Recommendation

7.1 Motivation

Our unsupervised meta-learning has essentially produced a large collection of models (*i.e.*, low-density separators). Chapter 6 integrates this model library into a single network, through leveraging these separators to constitute a top layer of the network. An alternative way is to treat the library as external memory and use it in an "off-the-shelf" manner. By doing so, it allows us to generate new models on-the-fly for a variety of tasks. In this chapter, we show such use case by combining unsupervised meta-learning with model recommendation to address a more general, realistic object detection problem.

Over the past few decades, there has been much progress in designing effective object detectors, especially when enough data are available. For example, classic techniques, such as deformable part models (DPMs) [107] via sliding-window fashion, are best suitable for semi-rigid objects, while Exemplar-SVMs [252] with simple linear SVMs are highly category and instance specific; the recent top performing detection systems instead favor bottom-up region proposals [72,126,382] which tend to perform well for non-rigid objects. Along with these, more powerful appearance models, feature learning mechanisms beyond standard hand-engineered features such as R-CNN [126], fast R-CNN [125], faster R-CNN [315], SPP-net [152], YOLO [312,313], SSD [248], OHEM [344], R-FCN [79], FPN [243], RetinaNet [244], and Mask R-CNN [150] are emerging.

All of these modern detectors have in common the same supervised training framework in which a large annotated dataset is required and in which training from scratch is restarted for a new task, *e.g.*, a new category. In practice, however, it might be difficult to produce enough data for a new task. Moreover, in many applications it is desirable to *rapidly* train a new detector for a new task, something that is typically not possible in any of these current approaches which require expensive training iterations. Thus, our objective is to quickly generate good models for a detection task given a small number of labeled samples.

Again, similar to Chapter 6, our approach is based on the informal intuition that, given a very large set of models, it is likely that some of the models would have good performance on a new detection task, as stated in the *infinite monkey theorem* [1]. More formally, inspired by this intuition, we explore a new approach, which is based on the observation that, while

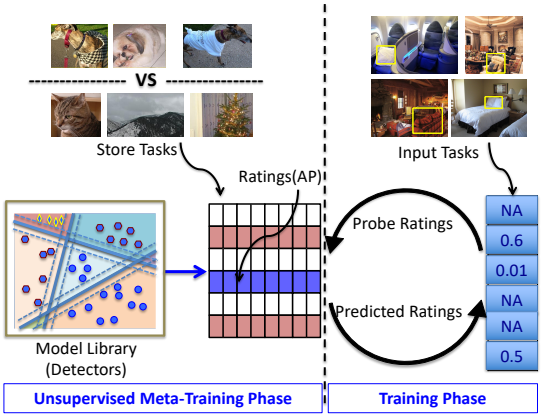


Figure 7.1: During unsupervised meta-training phase, a large library of object detectors informative across categories is generated. Their ratings on different detection tasks are recorded to form a ratings store. For a new target task or category and using ratings of a small probe set of detectors on its input task with limited samples, recommendations are made by collaborative filtering. A usable object detector for this new task is thus rapidly generated as single or ensemble of the recommended models.

it may be hard (impossible) to generate enough training data for a new task, it may be easy to generate a large library of models and evaluate them *off-line* on a large set of tasks. Given this data on the performance of many models on many tasks, it may be possible to guess which of these stored models are best suited for a novel target task by evaluating them on that new task. The hope of course is that we can do so with a smaller set of samples of the target task than would be necessary for training from scratch. In addition to (hopefully) requiring a much smaller sample set, this approach could be far more efficient because it merely *recommends* models, which have been generated off-line.

A natural question then is why should such an approach be possible? The first observation is that, while any *specific* detector cannot generalize well across tasks, in a large-scale library, however, it is likely that one of the library models happens to be tuned with the similar conditions as the new target task. Combining multiple such models into a single new model may perform well on the new task. This would be true especially when considering the shared properties across instances and categories [89, 191, 239, 287]. This is not sufficient, however, as we are still faced with the problem of selecting the right models out of the library. A naïve approach would be to evaluate each model from the library on the input task and select the one(s) that perform the best. Unfortunately, this direct approach typically performs poorly because of the limited data available in the input task. More precisely, the evaluation scores of the library model on the input task might be noisy enough that the ranking of the models is not reliable.

The second observation then is that, if the library and the set of tasks are large enough, there might be enough correlation between the models that it is possible to predict the performance of the models on the target tasks by using the *entire* combined experience with the model library and the tasks. This is similar to the approach taken in recommender systems [206] in which a matrix of ratings of items (the analog of our models) by users (the analog of our tasks) is used to predict the ratings that a new user (the target task)

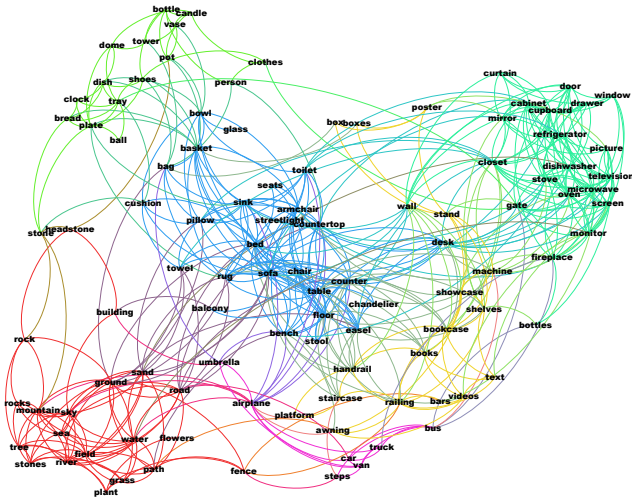


Figure 7.2: Continuous Category Space Discovery. We measure the similarity between the 107 categories on SUN by their number of shared PBC models within the top-200 ranked ones, and visualize them using ForceAtlas [24, 181]. The edge between two category nodes indicates that they have at least 10 models in common. Although the models are generated entirely unsupervisedly on PASCAL, it is interesting to note that visually or functionally similar categories are naturally grouped together: such as the green cluster of monitor, stove, microwave, oven, dishwasher; red cluster of sea, river, field, sky; and purple cluster of car, bus, van, truck, etc.

would generate, *i.e.*, predict the top ranked items (the analog of the best models for the target task). Inspired by prior work in the context of recommending action classifiers [255, 256], our goal is to explore how this approach can be used in *detection* tasks — a more general and challenging recognition scenario. While we share the same general idea of using recommender system for vision tasks, the specific technical approaches here are quite different from those in [255, 256].

The general setup is thus as follows. We first build a large library of object detectors and we record their performance or "ratings" on a large set of detection tasks, which we call the "ratings store". Consistent with Chapter 6, we call the process of generating and evaluating large library of models *off-line* as "meta-training". Given a new target task, recommendations are made by trying, or rating, a small subset, called the probe set, of detectors on the input task with few samples. That small set of ratings, along with the ratings of all the detectors in the ratings store, is used to predict the ratings of all the models on the target task. We then select models based on the recommendations and use them for the new task. This thus becomes our new formulation of object detection, which we term as **model recommendation for object detection** as shown in Figure 7.1.

Intuitively, in this approach we generate a new model by learning from experience, *i.e.*, from the matrix of evaluations of models on tasks, instead of learning from supervised data as is normally the case. The model recommendation system provides a succinct way to combine both large-scale models and big visual data into a joint task-model space. By directly manipulating in the new space, the experience of acquiring models becomes a procedure

similar to shopping for finished products in a store. The generated model library can be also viewed as a prior or regularization with respect to the common visual knowledge. In previous work, examples of transfer of prior experience to a new task include concept drift [380], domain adaption, transfer learning [288] (*e.g.*, sparse prototype representations [303], hypothesis transfer learning [215, 373], regularized SVM [14]), multi-task learning (*e.g.*, rank-reduced shared classification model [7, 298]), concept modeling in the field of multimedia (*e.g.*, LSCOM [272]), intermediate representation based on learning classifiers on related tasks (*e.g.*, Object Bank [237, 424], Classemes [38]), which address a different scenario and often require extensive supervised retraining on the new target task.

Our contributions are four-fold. (1) We show how such a recommendation setup can be operationalized using collaborative filtering techniques, including an analysis of its operation on a large-scale, controlled experiment. (2) We detail how tasks are defined and detectors are generated in the off-line stage. (3) We show how a universal detector library, predictable discriminative binary codes (PBCs) based on our LDS in Chapter 6, is generated without a bias to a particular set of categories, and we show how it is particularly effective for populating the ratings store in this setup. New detectors are thus obtained rapidly on novel classes with few samples *without* conventional supervised training involved. (4) Another interesting finding of this large-scale model system is a continuous category space discovered by model sharing, as shown in Figure 7.2 and explained in Section 7.5.1.

7.2 Terminology and Approach Overview

A *task* $T_j = \{(\boldsymbol{x}_{j1}, y_{j1}), (\boldsymbol{x}_{j2}, y_{j2}), ...\}$ is assumed to be a self-contained object detection problem, where $\{\boldsymbol{x}_{j1}, \boldsymbol{x}_{j2}, ...\}$ are input images, and $\{y_{j1}, y_{j2}, ...\}$ are the corresponding annotations indicating labels and bounding boxes for objects of interest. Note that annotations might have different interpretations across tasks. Moreover, the image samples associated with a task need not be unique; the same data sample might be shared across many tasks (*e.g.*, if one dataset is used for different tasks).

Models we are interested in here are object detectors. They can be pre-trained detectors with no free parameters, or they can be detectors trained on different data sources. They can be quite strong models highly tuned for certain tasks, or they can be generic weak detectors.

The *rating* of a model M_i on a task T_j is a number describing how well the model performs on that task, which is denoted by $R(M_i; T_j) = r_{ij}$, and is restricted to the range $[0, 1]$. For example, for the detection task, we follow the performance evaluation procedure of the PASCAL VOC Challenge [100]. A detection is considered correct when the area of overlap (measured by intersection-over-union, IoU) between the predicted bounding box and the ground-truth bounding box exceeds 50%, and the precision/recall curve is computed accordingly. Since a single number is needed as rating, the average precision (AP) is chosen to summarize the shape of the precision/recall curve.

The *ratings store* is a matrix $\boldsymbol{R}$, of the ratings of the models in the library on different tasks, where rows correspond to models, and columns correspond to tasks. The ratings store records the performance of models on tasks generated off-line during the meta-training phase, and encodes the performance distribution of models on different tasks. The matrix is $n \times m$, where n is the number of models in the library, and m is the number of store tasks. We assume that $\boldsymbol{R}$ is complete, *i.e.*, every model has been rated on every task in the store.

Finally, the task in which we are actually interested is a *target (or hidden) task*. The data in the *input task* is limited to a few samples from this target task. For instance, the target task

is to detect a certain type of cat; however, for its input task, we only have access to a limited number of training samples, *e.g.*, on the order of ten samples. We hope to recommend models that work well on the target task. The available ratings are the ones from the probe set on the input task, which are both limited and noisy. When this set of ratings on the input task is fed into a model recommendation system, the returned predictions are a full and de-noised version for the target task. Recommender systems relevant to our scenario are based on well-established collaborative filtering techniques [206].

Due to the complexity of real-world scenarios, recommending a single best model and hoping it will work well on the target task might not be realistic. One extension is to jointly recommend sets of models as in ensemble learning [438] and mixture of experts [189]. The simplest approach is to use the top-k recommendations, which simply selects the top-k models based on their predicted ratings from model recommendation. However, this obvious strategy can potentially recommend a highly redundant set. One alternative is to recommend a diverse set of top performing models [319]. Once a set of models is selected, they are combined by using a task-dependent fusion strategy, *e.g.*, by training weights for the models on the input task.

To sum up, we start with a large collection of models and images, from which we build a ratings store during meta-training. Then, during training given a new input task and a probe set on that task, we use collaborative filtering techniques to predict the ratings of all the models on the hidden task and to return a single or ensemble of models with the highest predicted ratings as the recommended models.

7.3 Collaborative Filtering

Based on the probe set ratings and the ratings store, collaborative filtering techniques predict the ratings of the entire library. We use collaborative filtering techniques based on matrix factorization, which assume a low-rank approximation to the ratings store that naturally embeds both tasks and models to a joint latent factor space of dimensionality d, such that task-model interactions are modeled as inner products in that space [205, 206]. Although the rating distribution of model recommendation might be different from that of the typical consumer recommender system, the ratings store still has exchangeability properties—arrays of random variables whose distributions are invariant to permutations of rows and of columns, which makes it statistically justified to use a factor model that implicitly encodes the Aldous-Hoover theorem for exchangeable matrices [193, 194].

In this approach, we associate each model M_i with a vector $\boldsymbol{u}_i \in \mathbb{R}^d$, and each task T_j with a vector $\boldsymbol{v}_j \in \mathbb{R}^d$. For a given model M_i, the elements of $\boldsymbol{u}_i$ measure the extent to which the model possesses the factors. For a given task T_j, the elements of $\boldsymbol{v}_j$ measure how well models with the corresponding factors will perform on the task. The interaction between M_i and T_j, *i.e.*, the overall performance of that model on the task, is then characterized by the dot product of $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$. The estimate of rating r_{ij} of model M_i on task T_j is approximated as

$$\widehat{r}_{ij} = \boldsymbol{u}_i^T \boldsymbol{v}_j. \tag{7.1}$$

After the recommender system infers $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ from rating patterns, it can easily predict the rating a model will give to any task by using Eqn. (7.1). The crucial issue is how to transform each model and task into vectors $\boldsymbol{u}_i, \boldsymbol{v}_j$.

Many matrix factorization techniques can be considered in this context, such as restricted Boltzmann machines [325], sparse coding [245, 246], and maximum margin matrix factorization optimizing directly for ranking scores [405]. Here we limit ourselves to direct factorization approaches which are sufficient for our purpose. Specifically, we consider approaches based on singular value decomposition (SVD) and non-negative matrix factorization (NMF), which we describe briefly in this section.

7.3.1 Factorization Techniques based on SVD

One simple way to identify the latent semantic model and task factors is by singular value decomposition (SVD) to decompose the rating matrix $\boldsymbol{R}$ into two factor matrices. In practice, the distribution of ratings may be significantly biased: some models may produce systematically higher ratings than others, and some tasks may be systematically easier than others. Hence, it is necessary to estimate the portion of rating values that individual model or task biases can explain [206]. A first-order approximation of the rating r_{ij} is introduced to identify the biases as $b_{ij} = \mu + q_i + p_j$, where μ denotes the global average rating, and the parameters q_i and p_j indicate the observed deviations of model M_i and task T_j, respectively, from the average. The biases can be easily estimated from the ratings store as in [206, 256] and used to modify Eqn. (7.1) as

$$\widehat{r}_{ij} = \mu + q_i + p_j + \boldsymbol{u}_i^T \boldsymbol{v}_j. \tag{7.2}$$

Now, the residual rating, defined as $\overline{r}_{ij} = r_{ij} - \mu - q_i - p_j$, does not remain positive. In this case, we can simply use conventional SVD to obtain the factor matrices. Formally, the residual rating matrix is decomposed as

$$\overline{R} = ESF \approx (E_d S_d)\, F_d = UV, \tag{7.3}$$

where d indicates the number of factors, S_d is the $d \times d$ upper left sub-matrix of S, and E_d, F_d are the first d columns of the left-singular vector matrix E, the first d rows of the right-singular vector matrix F, respectively. Hence, the model factor is $U = E_d S_d \in \mathbb{R}^{n \times d}$ with $\boldsymbol{u}_i^T$ as its ith row vector, and the task factor is $V = F_d \in \mathbb{R}^{d \times m}$ with $\boldsymbol{v}_j$ as its jth column vector.

7.3.2 Factorization Techniques based on NMF

Given that the ratings are all non-negative, an alternative approach is to use non-negative matrix factorization (NMF) [402], which incorporates the non-negativity constraint into the factored matrices. Formally, given the rating matrix $R \in \mathbb{R}_{\geq 0}^{n \times m}$ with non-negative elements, NMF seeks to decompose $\boldsymbol{R}$ into a non-negative $n \times d$ basis matrix $\boldsymbol{U}$ (model factor) and a non-negative $d \times m$ coefficient matrix $\boldsymbol{V}$ (task factor) , such that

$$R \approx \boldsymbol{U}\boldsymbol{V} = \sum_{l=1}^{d} \boldsymbol{u}_{\cdot l} \boldsymbol{v}_{l\cdot}, \tag{7.4}$$

where $\boldsymbol{u}_{\cdot l}$ is the lth column vector of $\boldsymbol{U}$ while $\boldsymbol{v}_{l\cdot}$ is the lth row vector of $\boldsymbol{V}$. To learn the two factor matrices, we use the prototypical multiplicative update rules [228].

Now, given the ratings of k probe models for an input task, denoted as r_k, the factor vector of the input task can be estimated by casting as a non-negative least-squares problem

with respect to v:

$$(\widetilde{\boldsymbol{u}}_k)\,\boldsymbol{v} = r_k, \tag{7.5}$$

where v is a $d \times 1$ vector, and $\widetilde{\boldsymbol{u}}_k$ is a $k \times d$ sub-matrix of U, whose rows are ratings of probe models on store tasks. v can be solved by fixing $\widetilde{\boldsymbol{u}}_k$ while updating v using the multiplicative update rules. After the factor v of the input task is learned, its ratings for all the models is predicted as

$$\widehat{r} = \boldsymbol{U}\boldsymbol{v}. \tag{7.6}$$

7.4 Recommender System Analysis

Naturally, the first and most important question to answer is whether collaborative filtering could successfully recommend *correct* models. A second question is to elucidate the role of the different design choices involved in this approach. To answer these questions, we designed a controlled experiment with real detection tasks, large-scale data (namely, PASCAL VOC 2007) and full-scale ratings store so that for any of the target tasks one or several of the n models in the library is the correct model to use or at least a reasonable enough approximation. Thus, this is an example of controlled recommender systems whose expected performance is known in advance. This is inspired by the M-closed framework [192] for statistical evaluation.

Of course, in order to support this controlled analysis, this setup is somewhat contrived. In particular, it uses a library of supervised models biased to a particular set of categories. We will describe the actual set of models that we propose to use in a real system in the next section, introducing a new way of generating object detectors in an *unsupervised* manner.

Specifically, we use all the $12{,}608$ ESVMs [252] pre-trained for 20 categories on the PASCAL VOC 2007 *trainval* dataset [100] as the model library. We treat these models independently here when using them to make detections. When our target task is defined as detection on the PASCAL VOC 2007 *test* dataset for the same 20 categories, there exists a subset of ESVMs from the corresponding category in the library that work well for the task, which are supposed to be identified by the recommender system.

Note that we use ESVMs as a convenient source for generating a large number of models to test our approach. We do not advocate that using ESVMs is in general the best tool for detection and, in fact, many other types of models could be used in this framework. For example, model recommendation can be naturally applied to other banks of detectors such as ELDAs [146] and Object Bank [237].

7.4.1 Task Generation

For the store tasks, generally speaking, we could design detection tasks based on specific categories, and group different object instances in the dataset accordingly to generate tasks, but this would bias the system strongly toward these specific categories and prevent it from generalizing to new input tasks. Hence, to demonstrate the performance of the proposed framework, we use purely random tasks instead. Specifically, each task is constructed by randomly selecting 10 images from the PASCAL trainval dataset. We generate $10{,}000$ different tasks in total. The final ratings store is of size $12{,}608 \times 10{,}000$. Since the images within a certain task are random and are therefore not tied to a specific category, a detection for a

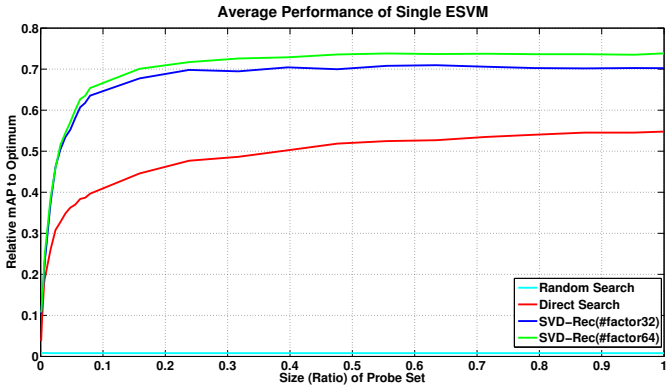


Figure 7.3: Effect of probe set size for *individual* ESVM across 20 categories on the PASCAL VOC 2007 detection test dataset. X-axis: probe set size (ratio to the size of the entire library). Y-axis: average performance of the recommended model using SVD with different factor numbers (green and blue curves) vs. random search (cyan-blue curve) and direct search (red curve) baselines, reported as relative mAP to optimum, which is the best achievable mAP for single ESVM (upper bound).

given task is counted as a positive detection if it intersects the bounding box of the ground-truth annotation of any category. This is distinct from the typical detection task setting, where one always focuses on sets of visually similar objects from the same category.

For the input tasks, we use the PASCAL VOC 2007 test dataset. We view the images containing objects from these 20 categories as 20 hidden tasks, respectively. Then, for each hidden task, we randomly select 10 images to create the corresponding input task. We randomly generate 50 input tasks per hidden task, and report the average performance.

7.4.2 Experimental Evaluation

There are several design choices in this approach, such as different input tasks, collaborative filtering techniques used, number of factors, size of probe sets, size of recommended models, etc. We evaluated the impact of all of these design choices on this ESVM-based setup in which we have predictable performance of the models. To this end, we only consider recommending the *single best* model. The model is selected given the input task, and then evaluated on the hidden task. The results are reported as relative mAP to optimum (the best achievable mAP for single ESVM on the hidden task, which is an upper bound on the performance).

Baselines. We compare against two natural baselines: 1) Random Search—randomly pick a model from the probe set; 2) Direct Search—pick the best model from the probe set based on their initial ratings on the input task.

Size of probe set. We randomly select a subset of models as the probe set. The average performance over 20 categories using SVD is shown in Figure 7.3 as function of the size of the probe set. The factor numbers are 32 and 64. Model recommendation works consistently better than direct search as the probe set increases, even when using all the models as the probe set. Note that the average model performance indicated by random search is quite poor.

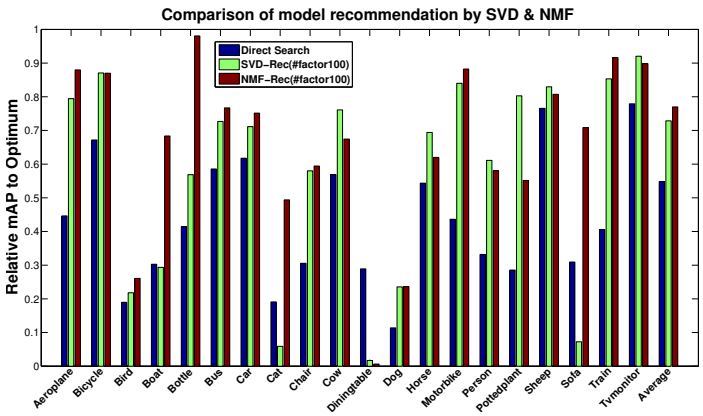


Figure 7.4: Comparison of different collaborative filtering techniques. X-axis: 20 categories on the PASCAL VOC 2007 dataset. Y-axis: recommendation performance using SVD (green bars) and NMF (red bars) both with factor number 100 vs. direct search baseline (blue bars), reported as relative mAP to optimum. NMF works better than SVD in the majority of cases.

Different collaborative filtering techniques. The relative performance of SVD and NMF when all the models are selected as the probe set is shown in Figure 7.4. The plot shows that they have comparable performance, with NMF better than SVD. Combining the two methods, *i.e.*, using the average output of these two systems as the final prediction would further improve the performance. Importantly, both collaborative filtering approaches perform significantly better than the naïve direct search.

7.5 Unsupervised Meta-Learning for Object Detection

The results of the last section validate our claim that model recommendation is able to select useful models. The other important issue remains how to generate a large collection of potentially "expressive" models. ESVMs are restricted in that each model is highly tuned for a specific instance, and thus constrained to one particular set of categories when building models. However, models informative across categories and datasets could be achieved via *unsupervised meta-training*. In this section, we give an example of meta-training that generates object detector models based on our modified LDS, which we term as predictable discriminative binary codes (PBCs) following [309].

We use CNNs as the feature space [212]. Different from Chapter 6, we use category-independent region proposals as the basic processing and decision units. Given a large corpus of unlabeled training images, we first generate region proposals using selective search [382]. For each region proposal, we extract a 4,096-D feature vector $fc7$ from the final hidden layer of the pre-trained CNNs structure on ILSVRC 2012 [321] (without fine-tuning on other datasets) [126, 185, 212]. Now we have constructed a feature space with unlabeled proposals.

Following the procedure in Chapter 6, we obtain several distinct and compact groups of labeled data by employing Max-Min sampling [75], which we use as initial pseudo-labeled data. We then produce the semi-supervised PBCs from these labeled datasets. Specifically, from a large proposal pool $\mathcal{P}$ we first draw a subset $\mathcal{A}_S$ by random subsampling. Within

$\mathcal{A}_{\mathcal{S}}$, we create prototype sets $\mathcal{B}_{\mathcal{PL}}$ by Max-Min sampling. We view $\mathcal{B}_{\mathcal{PL}}$ as pseudo-labeled data and the remaining ones in $\mathcal{A}_{\mathcal{S}}$ that are still unlabeled as $\mathcal{C}_{\mathcal{UL}}$. We use the same setup and parameters for the Max-Min sampling procedure reported in [75], resulting in $\mathcal{B}_{\mathcal{PL}}$ consisting of 30 categories with 6 samples per category. Based on only the pseudo-labeled data in $\mathcal{B}_{\mathcal{PL}}$, we learn a 10-D prototype PBC (*i.e.*, our modified LDS) [309]. We select and add 50 samples to each pseudo-category from the unlabeled proposals $\mathcal{C}_{\mathcal{UL}}$ using category specific attributes only to expand coverage as in [67]. Using this augmented dataset $\mathcal{D}_{\mathcal{AUG}}$, we retrain a new 10-D PBC, *i.e.*, 10 models. To ensure diversity, the subsampling procedure repeats for T times, and we learn $10T$ models in total. They thus build up our model library for widespread visual/attribute coverage.

7.5.1 Experimental Evaluation

To show that the PBC models generated by unsupervised meta-training are informative across categories, we consider large-scale detection tasks across different datasets.

We use the entire PASCAL VOC 2007 dataset to generate model library and ratings store. Store tasks are generated similarly as the previous ESVM-based system. The final ratings store is of size $10{,}000 \times 10{,}000$. For the hidden tasks, we consider detection of 107 object categories on the SUN 09 test dataset [68], which span from regions (*e.g.*, road, sky) to well defined objects (*e.g.*, car, sofa) and highly deformable objects (*e.g.*, river, curtain). The input tasks are generated by randomly sampling 10 images from the corresponding category on the SUN 09 training dataset. For those categories whose numbers of training samples are smaller than 10, we use all the training samples. We also randomly generate 10 input tasks per hidden task, and obtain the average performance. For each PBC model, we follow the typical detection pipeline of R-CNN as in [126].

This is a challenging problem given that feature learning is implemented on ImageNet and PBC models are hyper-trained on PASCAL, while the system is finally tested on SUN. They are very different domains. The farther away from these source domains, the more pertinent the test dataset will be for experimental evaluation. (Note that testing on other datasets, *e.g.*, PASCAL VOC 2012 and ImageNet, cannot serve this purpose due to shared data.) Compared with PASCAL, on the SUN dataset the number of objects and the amount of noise significantly increase with far more annotated object categories and typically 7 object classes per image [68]. Additional contextual information is thus usually necessary to boost the detection performance [68,71].

Ensemble Model Recommendation

To deal with the fact that different input tasks may require different factor numbers, we perform collaborative filtering using SVD and NMF with factor numbers d ranging from 100 to 1000. For each d, we evaluate their precision on the input task, and we then average the ratings across all these factor configurations. The final selected models are ranked according to this averaged prediction. Consistent with early work in consumer recommender system [413], we found that this method gives by far the best performance. After recommending the top-k desired models, we calibrate them by standard Platt scaling [252,299] on the input task to obtain comparable scores and then perform majority voting as the fused score for each proposal, followed by non-maximum suppression. Moreover, following the standard bounding box regression procedure [126], we also learn additional bounding box regressors on the input tasks, and rectify the region proposals at test time to mitigate mislo-

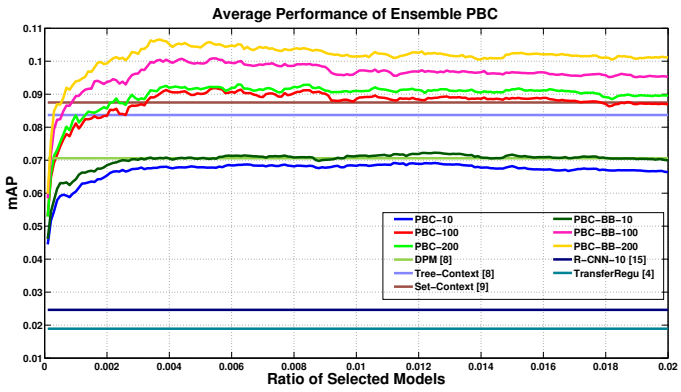
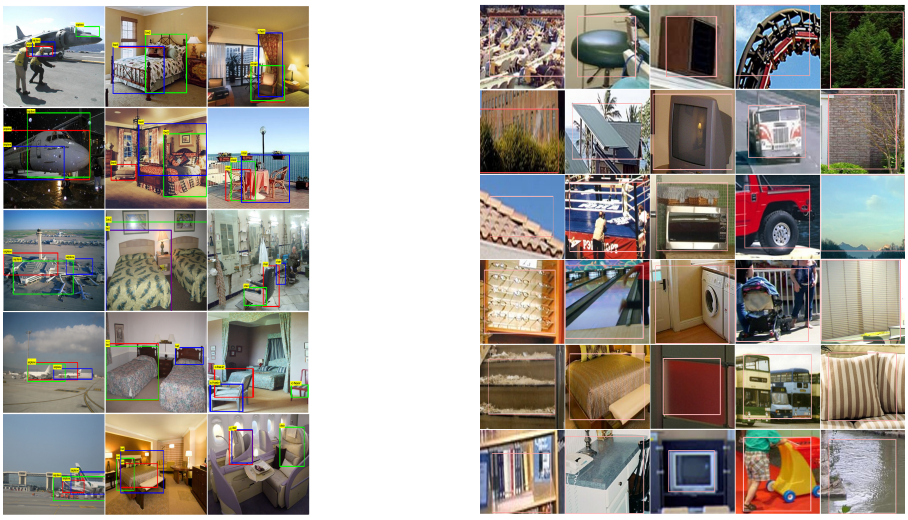


Figure 7.5: Average performance of ensemble PBC model recommendation with varied input task size over 107 categories on the SUN 09 dataset. X-axis: ratio of recommended models in the library. The ensemble model recommendation results for input tasks of size 10, 100, 200 *without* and *with* bounding box regression are shown in two sets of curves—PBC-10, PBC-100, PBC-200 and PBC-BB-10, PBC-BB-100, PBC-BB-200, respectively. We also show two sets of baselines to calibrate the results: the top three horizontal lines are the average performance of DPM using the entire training dataset and additionally annotated training data *without* and *with* introducing and encoding complex contextual relationships, respectively; the bottom two horizontal lines are the average performance of R-CNN directly trained and additionally transferred from our source models using 10 images from the input tasks, respectively. Note that accurate generic object detectors can be obtained based on input tasks that are from different categories and domains compared with where the PBC models are learned.

calizations induced by proposal based object detection. The average ensemble recommendation result is shown in Figure 7.5.

Baselines. We compare against five strong baselines: 1) DPM using the entire SUN training dataset and 26,000 additionally annotated objects [68]; 2) Tree-structured graphical model [68] and 3) Set-based representation [71] to encode complex contextual relationships; 4) Direct R-CNN training using 10 images from the input tasks [126]; 5) Transfer regularization [14] with R-CNN and our source models.

Figure 7.5 shows that ensemble of PBC models works consistently well as the size of the recommended model increases, as shown in the six curves. The result is quite significant if one notices the difference in numbers of training samples: Using only 10 images to select models generated from an out-of-domain dataset our approach is not only better than R-CNN directly trained from few samples, but it is also better than supervised DPM trained from *lots of* in-domain data (hundreds to thousands), comparable to DPM with additional contextual models. It is hard to make a direct comparison with Object Bank/Classemes since they use the output of all the models as new features to retrain a new classifier while we directly combine a subset of models. Experimentally, if we use Classemes features to retrain a detector with few samples, it is worse than R-CNN. For the SUN dataset, the mAP is low due to some hard categories. (The low mAP is of the same order of magnitude as in challenging datasets, *e.g.*, DPM on ImageNet.) Besides, in our case with large-scale



(a) (b)

Figure 7.6: (a) Collaborative detection of different models for the same task. For three representative tasks: airplane, bed, and chair (left to right), we show the top detection of the top-3 ranked models (red, green, and blue bounding boxes) on sample images (top to bottom). Note that they would detect the same, different parts of the same, or different objects. This complementary behavior explains the boosted performance of ensemble models in Figure 7.5. (b) The same model is informative across different tasks. For five representative models (left to right), we show detection on sample images (top to bottom). Note that the models learned by unsupervised meta-training can be to some extent interpreted as attribute detectors. For example, the first column corresponds to all staircase-like objects with vertical, horizontal, inclined, and curved orientations (bottom to top). This attributes-like behavior explains the generality of the models for new target tasks in Figure 7.5.

sources of non-categorical classifiers, the conventional transfer learning techniques developed with well-trained categorical source classifiers [14] perform poorly due to induced negative transfer. This indicates that cross-source relations estimated by the recommender system are crucial to identify the relevant sources here. That is, because we have only few target samples and because the generic sources are weak, direct transfer will be very noisy.

Similar to the object distribution in a large-scale scenario, the performance distribution of models in the library for a specific task also follows a *power law*, which is implicitly encoded by the recommender system. Moreover, as we increase the sample size in the input task from 10 to 100 and 200, it shows steadily increased performance due to more accurate mAP, rank, calibration, and regressors estimation, and outperforms sophisticated contextual models. Bounding box regression provides additional performance boost as expected. This demonstrates that the recommender system both successfully builds expressive models during meta-training and selects useful models based on input tasks.

Qualitative Visualization

Detection and model visualization. To better understand the PBC models, we provide two types of representative visualization: 1) Collaborative detection of different models for the same task in Figure 7.6a; 2) Attributes-like behavior of the same model across different tasks in Figure 7.6b.

Continuous category space discovery. By calculating the number of shared models, a similarity matrix is obtained between the 107 categories. We use ForceAtlas [24, 181] to visualize it in Figure 7.2. Although the models are generated without any label information, visually or functionally similar categories are naturally grouped together. It again shows the expressive power of the PBC models, which naturally identifies a continuous category space.

Chapter 8

Additional Applications

In this chapter, we demonstrate how our meta-learning approach and LDS can be applied to other problems and tasks with limited samples. We include two applications: unsupervised hypothesis transfer learning and few-shot hash learning for image retrieval.

8.1 Unsupervised Hypothesis Transfer Learning

8.1.1 Motivation

The first application is hypothesis transfer learning (HTL) [13, 14, 63, 96, 196, 214–217, 373, 418, 419, 421], a transfer learning (TL) approach that has recently attracted increased attention. Transfer learning benefits from transfer of prior knowledge from related tasks to new ones, in the majority of cases either on a data or instance level, or on a feature or parameter level [288]. Despite featuring well established theoretical guarantees, these approaches often suffer from great practical constraints and limitations [214, 288]: they require reusing data originating from the source domains and extensive supervised retraining on the target task, which is prohibitively expensive for large source data. On the contrary, HTL transfers directly on a model level by reusing source hypotheses — classifiers or models trained from source data. This framework is practically appealing, since it requires neither the availability of the source data nor any knowledge on how the source models relate to each other. HTL is also efficient especially with small target samples, in which source hypotheses are generated in advance and treated as black boxes without any consideration of their inner workings at transfer stages.

Much attention in HTL (and also TL) has been focused on integrating the source information into the target task in different ways. Unfortunately, very little work has addressed the generation of useful source models. In most cases, sources are simply category classifiers well-trained from large amounts of labeled samples. This however might be infeasible for real-world applications: we focus on learning from few samples for target categories, whereas we have to train good classifiers of related categories as sources from enough labeled data in advance. For instance, if we are interested in recognizing Père David's deer[1] from few samples, following the conventional HTL practice, we might need to first obtain

[1] A Père David's deer is a species of deer that has the neck of a camel, the hoofs of a cow, the tail of a donkey, and the antlers of a deer.

well-trained source classifiers of camels, cows, donkeys, and deer for transfer. Furthermore, sources generated in this way are tied to a specific set of categories due to its supervised nature, making it difficult to apply them across a wide spectrum of target categories.

It is thus unsurprising that the current HTL (and TL) algorithms are usually evaluated under well-controlled experimental setups: (1) use small-scale well-trained classifiers as sources, at most several hundred [13, 14, 419]; (2) split a dataset with a portion of categories as sources and the rest as targets, which implicitly reduces the impact of dataset bias, *e.g.*, leave-one-class-out [214,373]; (3) transfer between visually similar categories with ideal sources [164,239].

To address this largely-overlooked yet fundamental problem, our LDS can be viewed as a systematic scheme for generating *universal and expressive* source hypotheses in an *unsupervised* fashion, which frees the recognition from ties to a particular set of categories and which generalizes well for broad novel target classes. Now each hypothesis lies in a region of low density and the combined hypotheses constitute a joint partition of the feature space, leading to a library of unsupervised universal sources (UUS) with widespread visual coverage.

From the principle of Structural Risk Minimization, our UUS hypotheses provide an alternative mechanism to encode prior knowledge and control model capacity. This is related to the use of Universum (*i.e.*, unlabeled examples that do not belong to the concerned classes, sometimes called "non-examples") in addition to labeled data for capacity control, which proved to be helpful in various learning tasks [407]. When facing a large collection of non-examples, our UUS can be viewed as compressing the original source data while implicitly modeling a general distribution and preserving relevant information for classification.

Our UUS can be also considered as distinctive subdomains automatically discovered in a large source domain. Conventional discovery of latent domains for domain adaptation [128,165] is supervised, in which object category labels are used to constrain feasible subdomain separations on source datasets. However, our hypotheses are generated in an entirely unsupervised manner without requiring any labeled data. Moreover, [128, 165] need to explicitly model the distribution on different subdomains, and measure the distance between distributions. However, modeling the distribution of high-dimensional image features on large datasets is typically more difficult than classifying them. Hence, our approach is more flexible, scalable, and broadly applicable in practice.

This unprecedented large-scale source pool, with two orders of magnitude more hypotheses than the previous work, poses additional scalability challenges to the existing HTL approaches. These algorithms adopt a discriminative SVM framework (usually with a quadratic loss), in which a new target classifier is learned through adaptation by imposing closeness between the target classifier and a linear combination of the source hypotheses as regularizer. The weight associated to each source is either predefined for known transfer relationship [196], or determined by designing heuristic meta-level features [419], or estimated based on the conditional probability distribution of large amounts of unlabeled target data [63]. In other cases, the weight is obtained by minimizing empirical error with solely ℓ_2 norm [14,217,421] or sparsity-inducing (ℓ_0, ℓ_1) norm regularization [214,373]. These frameworks have been tested on problems with less than a few hundred sources, but have already showed some difficulty in selecting informative sources due to severe overfitting [214,373]. To resolve this issue, we propose a scalable model transfer SVM (MT-SVM) approach by combining an elastic net regularization and biased SVM with a hinge loss. The relatedness among the tasks is autonomously evaluated through a principled optimization

problem without extra validation, unlabeled samples, or a predefined ontology.

8.1.2 Model Transfer Support Vector Machine

Once we obtain the J source hypotheses $\{\boldsymbol{w}_j^{src}\}_{j=1}^J$ following the procedure of leaning LDS in Chapter 6, the original training samples used to build them are no longer used. We now consider a new target task with a small labeled training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^L$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ are the training samples and $y_i \in \{-1, 1\}$ are the corresponding labels. Hypothesis transfer learning (HTL) attempts to infer the target hypothesis $\boldsymbol{w}$ from both $\{\boldsymbol{w}_j^{src}\}_{j=1}^J$ and $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^L$ that generalizes better than the one produced only from $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^L$. HTL algorithms proposed so far are developed under a discriminative SVM framework modified by regularizing the distance between $\boldsymbol{w}$ and a linear combination of the sources $\boldsymbol{w}^{src}$. To identify useful sources, it is recast as a variable selection problem by constraining the combination weights with either ℓ_0, ℓ_1, or ℓ_2 norm [14, 214, 373]. However, a single type of norm has its own pros and cons. Especially, in our scenario with only few target samples and large-scale generic weak sources, these existing approaches would be very noisy due to severe over-fitting and would induce negative transfer.

As a well-known recipe, an elastic net regularization, combining a weighted mixture of ℓ_1 and squared ℓ_2 penalties, offers several desirable benefits: (1) ℓ_2 regularization is known to improve the generalization ability of empirical risk minimization [214]; (2) ℓ_1 norm, as a convex relaxation of ℓ_0 norm, always converges to a good solution in practice, avoiding potential bad local minima when using a greedy scheme [214] to directly solve ℓ_0 problems; (3) joint ℓ_1 and ℓ_2 enjoys a similar sparsity of representation and encourages a grouping effect [443]; (4) it is particularly useful in our case that the number of predictors is much bigger than the number of observations [443].

Formulation. By using the new regularization to rank the prior sources and introducing them as reference into SVM, we then obtain the objective function for our model transfer SVM (MT-SVM):

$$\min_{\boldsymbol{w},\boldsymbol{\beta}} \frac{1}{2}\left\|\boldsymbol{w} - \sum_{j=1}^{J} \beta_j \boldsymbol{w}_j^{src}\right\|^2 + \frac{\alpha}{2}\sum_{j=1}^{J}\beta_j^2 + \gamma\sum_{j=1}^{J}|\beta_j| + \lambda\sum_{i=1}^{L}\left[1 - y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i\right)\right]_+ . \tag{8.1}$$

The last term represents the data fit on the L training samples, measured by the hinge loss; it is the new information from the target domain. The first term is similar to the max-margin principle in standard SVMs, with the only difference being the bias towards the linear combination of the generic source hypotheses $\sum_{j=1}^J \beta_j \boldsymbol{w}_j^{src}$ instead of 0, in which β_j's are transfer weights; it is the prior information from the source domains. In order to automatically select the best subset of known hypotheses from which to transfer, the second and third terms are introduced as an elastic net regularization that favors sparse $\boldsymbol{\beta}$. Here, α, γ, and λ are the regularization parameters to control the trade-off between the error term and regularization terms.

Following the duality derivation analogous to standard SVMs, the optimal solution to Eqn. (8.1) satisfies

$$\boldsymbol{w} = \sum_{j=1}^{J} \beta_j \boldsymbol{w}_j^{src} + \sum_{i=1}^{L} \mu_i y_i \boldsymbol{x}_i, \tag{8.2}$$

where μ_i's are Lagrange multipliers. The final target model is then conceptually straightforward: it linearly combines the contribution from both the pre-trained generic models and target specific data, i.e., support vectors from both source and target domains. Combining Eqns. (8.1) and (8.2), as $\alpha \to \infty$ and $\gamma \to \infty$, β_j's will be forced to be zero and we will get back the standard SVM, *i.e.*, no transfer. As $\lambda \to 0$, $\boldsymbol{w}$ will be forced to be purely constructed as a weighted combination of $\{\boldsymbol{w}_j^{src}\}$'s, *i.e.*, maximum transfer. As $\alpha \to 0$, it becomes LASSO regression while ridge regression as $\gamma \to 0$. Hence by tweaking α, γ, and λ we obtain an intermediate solution with a decision boundary close to those of the auxiliary classifiers while separating the labeled examples well.

Optimization. The objective in Eqn. (8.1) can be optimized by alternating minimization of two subproblems:

- With fixed $\boldsymbol{w}$, the objective function of finding transfer weights $\boldsymbol{\beta}$ becomes an elastic net regularized least-squares minimization subproblem:

$$f(\boldsymbol{\beta}) = \frac{1}{2}\left\| \boldsymbol{w} - \sum_{j=1}^{J} \beta_j \boldsymbol{w}_j^{src} \right\|^2 + \frac{\alpha}{2}\sum_{j=1}^{J}\beta_j^2 + \gamma \sum_{j=1}^{J} |\beta_j|. \tag{8.3}$$

- With fixed $\boldsymbol{\beta}$, the objective function of learning target hypothesis $\boldsymbol{w}$ becomes a bias regularized SVM subproblem:

$$f(\boldsymbol{w}) = \frac{1}{2}\left\| \boldsymbol{w} - \sum_{j=1}^{J} \beta_j \boldsymbol{w}_j^{src} \right\|^2 + \lambda \sum_{i=1}^{L} \left[1 - y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i\right)\right]_+ . \tag{8.4}$$

Source selection by modified feature-sign search: We solve Eqn. (8.3) by extending the feature-sign search (FS) algorithm [230], one of the state-of-the-art techniques for efficient sparse coding (*i.e.*, ℓ_1 regularized least-squares) [245], to our case of elastic net regularization (*i.e.*, joint ℓ_1 and ℓ_2 regularized least-squares). FS searches and maintains an optimal active set of potentially nonzero coefficients and sets other coefficients zero. Although it was developed in the context of dictionary learning and sparse coding, FS still fits our scenario, in which we could view the source hypotheses $\{\boldsymbol{w}_j^{src}\}_{j=1}^{J}$ as known dictionary bases and rearrange them into the matrix form of dictionary $\boldsymbol{W}^{src}$. The equivalent optimization problem of Eqn. (8.3) is then

$$f(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{W}^{src}\boldsymbol{\beta}\|^2 + \frac{\alpha}{2}\|\boldsymbol{\beta}\|^2 + \gamma\|\boldsymbol{\beta}\|_1. \tag{8.5}$$

Since the only difference lies in the extra ℓ_2 regularization term that is differentiable, we modify the key update of $\widehat{\boldsymbol{\beta}}_{new}$ in a series of "feature-sign steps" with $\left(\widehat{\boldsymbol{W}}^{srcT}\widehat{\boldsymbol{W}}^{src} + \alpha \boldsymbol{I}\right)^{-1}$ instead of $\left(\widehat{\boldsymbol{W}}^{srcT}\widehat{\boldsymbol{W}}^{src}\right)^{-1}$ as in [230]. ($\widehat{\cdot}$ represents the active set.)

Model transfer via adaptive SVM: The optimal $\boldsymbol{w}$ in Eqn. (8.4) can be obtained by the Adaptive SVM algorithm [418, 419], which solves a quadratic program to maximize its Lagrange dual objective function. With small samples in our case, the problem can be efficiently solved by (modified) sequential minimal optimization [196, 419]. In addition, we initialize $\boldsymbol{w}$ using the standard SVM without bias on the given target training set. We then iteratively infer $\boldsymbol{\beta}$ and refine $\boldsymbol{w}$. Given the convexity of the problem, this block coordinate descent algorithm will converge to the global minimum.

Transfer Scenario	Method	Acc (%)
Non-Transfer	SVM (source only) [164]	59.15 ± 1.1
	SVM (target only) [164]	64.97 ± 1.8
	SVM (source and target) [164]	66.93 ± 1.3
Transfer with Source Data	GFK [127]	67.97 ± 1.4
	SA [110]	66.08 ± 1.4
	Daumé III [81]	71.39 ± 1.5
HTL with Supervised Sources	PMT [13]	69.81 ± 1.8
	MMDT [163]	67.75 ± 1.4
	Late Fusion (Max) [164]	68.86 ± 1.2
	Late Fusion (Lin. Int. Avg) [164]	66.45 ± 1.1
HTL with Unsupervised Sources	Clustering+MT-SVM	67.13 ± 1.2
	UUS+MT-SVM (Ours)	$\mathbf{74.83 \pm 1.2}$
HTL–Upper Bound	*Late Fusion (Lin. Int. Oracle) [164]*	$\mathit{76.76 \pm 1.3}$

Table 8.1: Performance comparisons between hypothesis transfer learning (HTL) with supervised (SS) and unsupervised (USS) source hypotheses generated from ILSVRC for one-shot learning in the Subset A (16 common classes) on the Webcam domain of the Office dataset. We also include for completeness the results of transfer learning with source data. Using a large library of unsupervised sources, ours yields performance superior to other state-of-the-art HTL methods with well-trained source category models, and even close to the oracle with an ideal source and the optimal transfer weight on the test set (performance upper bound).

8.1.3 Experimental Evaluation

In this section, we present experimental results evaluating our unsupervised sources (UUS) as well as our HTL approach (MT-SVM) on standard recognition benchmarks, comparing several state-of-the-art methods, and validating across tasks and categories the generality of our sources.

Implementation details. We use the modified $J = 20{,}000$ LDS [398] learned in $fc7$ feature space of the pre-trained AlexNet [91, 185, 212, 321] without fine-tuning as our UUS hypotheses. In term of MT-SVM, for λ, we use the default value 1 as in Adaptive SVM [418, 419]. For α and γ, in a preliminary experiment, we tested the ImageNet categories as targets and our UUSs for transfer. Empirically, we found that keeping the number of selected sources to be around $100 \sim 200$ yields good results. After searching α on a small grid $(0, 0.01, 0.1, 1, 10, 100)$ as suggested in [443], we found that $\alpha = 10$ roughly achieved the desired stable solution. For all our experiments, we then fixed $\alpha = 10$, and tuned γ to minimize the leave-one-out-error.

Comparisons with Supervised Sources. Naturally, the most critical question to answer is whether our UUS indeed facilitates generalization to novel categories with few samples, compared to their supervised counterparts (*i.e.*, category models, SS). To this end, we evaluate them on the Office dataset [324] in an experimental setup similar to Section 3.3.2.

Source hypotheses. We use our generated library of $20K$ UUSs as unsupervised sources.

Transfer Scenario	Method	Acc (%)
Non-Transfer	SVM (target only)	63.34 ± 2.1
HTL with SS	Multi-KT [373]	65.28 ± 1.3
	DAM [96]	66.13 ± 1.4
	GreedyTL [214]	68.72 ± 1.8
	SS+MT-SVM	70.30 ± 1.2
HTL with USS	Clustering+MT-SVM	64.92 ± 1.2
	UUS+MT-SVM (Ours)	$\mathbf{74.19 \pm 1.3}$

Table 8.2: Performance comparisons between HTL with SS and USS for one-shot learning in the Subset B (15 non-overlapping classes) on the Webcam domain of the Office dataset.

Moreover, for a fair comparison, we also generate another $20K$ sources by a naïve unsupervised approach denoted as clustering, which creates hypotheses by clustering the data and produces classifiers between clusters. For supervised sources (SSs), we use the labeled samples from the $1{,}000$ categories on ILSVRC as source data, with approximately $1{,}200$ examples per category. With the same CNN features, we then train source SVM classifiers in one-vs.-all fashion, leading to $1{,}000$ category models on these labeled samples.

Target tasks. To better understand the transfer process, we group the 31 target classes into two subsets. **Subset A:** we focus on the 16 common classes between Webcam and ILSVRC as our target categories as in [164]. 1 labeled training and 10 testing images per category are randomly selected on the Webcam domain, *i.e.*, one-shot transfer and a balanced test set across categories. Therefore, each test split has 160 examples. **Subset B:** we also test the other 15 non-overlapping classes as our target categories in the similar one-shot transfer scenario. For each subset, we evaluate our two types of sources, independently calculate the multiclass accuracy, and report the average performance and standard errors over 20 random train/test splits, as shown in Table 8.1 and Table 8.2.

Baselines. We compare against three types of baselines. **Type I non-transfer**: SVM (source only), SVM (target only), and SVM (source and target). They are category SVMs trained on only labeled source, only target, and both source and target data, respectively. For completeness we also include **Type II transfer learning based on (labeled or unlabeled) source data**: GFK [127], SA [110], and Daumé III [81]. For instance, Daumé III retrains SVMs on the augmented source and target data using tripled augmented feature, resulting in a relatively expensive procedure given the potentially large size of the source data and high feature dimensionality [164]. Note that some of these baselines are only available for Subset A, since they require that the source comes from the same category as the target.

Type III baselines of HTL with supervised sources. For Subset A, transfer becomes a domain adaptation problem: the transfer is largely dominated by the source category corresponding to the target as the single most relevant one, making other categories uninvolved in the transfer process. We then transfer the corresponding learned category models without a source selection step, including (1) PMT [13], which regularizes the angle between the target and source hyperplanes; (2) MMDT [163], which jointly optimizes over a feature

transformation mapping target points and classifier weights to the source feature space; (3) Late Fusion, which independently trains a source and a target category classifier, and sets the final score for each example by choosing the maximum (Max) or linear interpolation (Lin. Int.) of source and target classifier scores. We report the performance of linear interpolation both averaged across linear combination hyper-parameter settings (Avg) and with its best possible setting on the test set per experiment (Oracle). Importantly, the latter case is the best achievable performance for HTL (upper bound), which is equivalent to an ideal source (the same category as the target and with a large amount of training examples) transferred with the optimal transfer weight. These results are reported from [164]. For Subset B, without explicit category correspondence, we use all 1,000 supervised sources and transfer the relevant ones by state-of-the-art HTL approaches, including Multi-KT [373], DAM [96], and GreedyTL [214].

Table 8.1 and Table 8.2 show that our transfer with unsupervised source hypotheses outperforms non-transfer and other state-of-the-art techniques of transfer with source data and transfer with supervised source hypotheses. Notably, in Table 8.1 ours achieves significant performance *close to the oracle*. Moreover, the naïve unsupervised clustering approach works poorly here. This verifies our assumption that information across categories is actually intrinsic in the data even without any supervision and could be effectively identified by our UUS. With such unsupervised nature, our approach reduces the effort of collecting large amounts of labeled data and training accurate relevant source category models, as is normally the case in previous transfer learning works.

8.2 Few-Shot Hash Learning for Image Retrieval

Image retrieval is another important visual recognition problem and binary hashing, due to its computational and storage efficiency, has attracted considerable attention for representation and retrieval in large-scale image databases [87, 88, 129, 250, 269, 342, 431, 432]. While one/few-shot learning [104], as a fundamental problem, has been extensively discussed in the context of image recognition and classification [41, 388], very little work has addressed this issue for hash learning and image retrieval. In practical applications, however, a user might define customized query categories *on-the-fly* by supplying only a *small set* of specific examples, and requires the entire learning and retrieval procedure to be manageable in real time [38, 59, 61, 62, 111, 143, 213]. Such *few-shot hash learning* scenarios pose a significant challenge for the existing techniques, since they are usually category/dataset specific and cannot generalize well from few examples or generalize to novel categories.

Specifically, the state-of-the-art hashing approaches are data-dependent and directly learn hash functions from the target dataset in either an unsupervised or supervised manner. The unsupervised hashing [129, 179, 406] aims to propagate neighborhood relation of samples from a certain metric space into the Hamming space. However, distance metrics (*e.g.*, Euclidean distance or angular distance) typically cannot measure well the semantic similarity that is essential for image retrieval. By leveraging supervisory information in form of class labels, supervised hashing [250, 309, 342, 392] preserves semantic structure of the data. Unfortunately, with limited data, these approaches are prone to over-fitting, leading to degenerated performance.

Interestingly, the early research on data-independent hashing has learned generic binary codes that are independent of the data and categories. The flagship representatives, the locality sensitive hashing (LSH) and its variants [58, 124, 176, 184, 294], use simple random

projections to construct hash functions without exploring the data distribution. However, LSH usually requires long binary codes to achieve satisfactory retrieval accuracies, leading to large storage space and low recall performance [250]. Due to its pure data-independence, LSH still lacks the ability to preserve the desired semantic similarity.

In the spirit of learning classifier-based representations, some approaches, including Classemes [38], PiCoDes [40], Meta-Class [39], and predictable discriminative binary code (DBC) [309], leverage an *auxiliary labeled* dataset and generate codes either from pre-defined categories or learned super-classes of these categories. Unfortunately, to identify properties shared by many categories, these approaches rely on a large corpus of annotated auxiliary data samples and expensive training iterations. The generalization ability is still tied to this particular set of categories due to its supervised nature. In particular, the code length is usually constrained by the dimension of the original feature descriptors or number of categories [38].

To address these limitations, our LDS provides a basic framework for generating *unsupervised generic hashing* (UGH) in the sprite of LSH. Now a single code is informative by itself while the entire library of codes have a good coverage of the feature space [124, 184, 294]. Given a new target task with few samples, task/category-specific codes are selected from the large pool to form a compact representation of the novel category. The retrieval is accomplished by nearest neighbor search in the Hamming space. Distinguished from previous work, separating unsupervised code generation and task-specific code selection also makes the code length adaptive for different categories/tasks and makes the use of long codes feasible, which facilitates the practical usage of the hash codes [430].

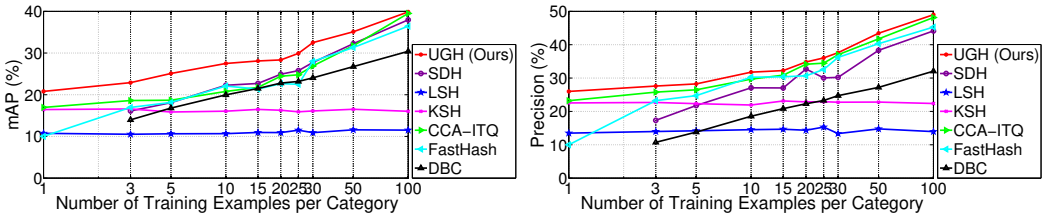
8.2.1 Code Selection and Usage for Novel Categories

This large library of hash codes can be viewed as an over-complete representation. Given a new target task, *e.g.*, a novel category, at query time with a small set of training images, we could simply use all of these codes as descriptors for image retrieval. Motivated by the power of sparse representations, an alternative is to select the most informative bits so as to infer what is shared with this specific input category. To achieve this, during the training phase, using all the codes as features and the small training samples from the target task, we first learn an ℓ_1-regularized model, *e.g.*, ℓ_1-regularized SVM, and pick the active bits according to the desired code length, which correspond to the weights of larger absolute value [309]. We thus obtain category specific codes as a compact representation. Using these codes, we then perform nearest neighbor search for retrieval purpose and evaluate the performance.

8.2.2 Experimental Evaluation

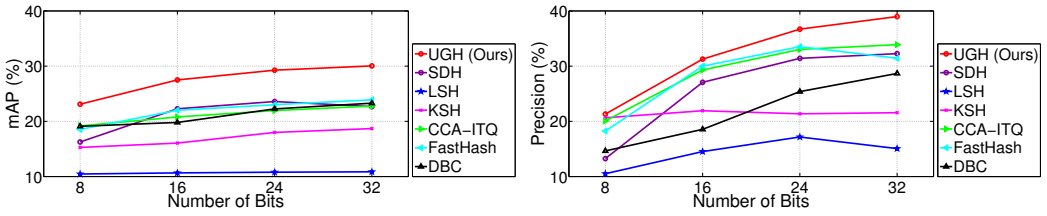
In this section, we present experimental results evaluating our UGH on multiple standard image retrieval benchmarks, comparing with several state-of-the-art supervised and unsupervised hashing methods for small-sample learning, and validating across tasks and categories the generality of UGH.

Implementation details. We use the modified $20{,}000$ LDS [399] learned in $fc6$ feature space of the pre-trained AlexNe [185,212] as our unsupervised generic hashing (UGH) functions. For fair comparisons, our UGH and all the following supervised and *unsupervised* baselines learn hash codes over the pre-trained AlexNet features. For all the baselines, we



(a) (b)

Figure 8.1: Performance comparisons between UGH and competing supervised hashing approaches for few-shot hash learning and image retrieval on the CIFAR10 dataset. X-axis: number of training examples per category. Y-axis: mean average precision (mAP) (Figure 8.1a) and precision@2 (Figure 8.1b). With the same code length 16, our UGH significantly outperforms these baselines for learning with few samples.



(a) (b)

Figure 8.2: Performance comparisons between UGH and competing supervised hashing approaches for few-shot hash learning and image retrieval on the CIFAR10 dataset. X-axis: code length. Y-axis: mean average precision (mAP) (Figure 8.2a) and precision@2 (Figure 8.2b). With the same 10 training examples per category, our UGH consistently outperforms these baselines by large margins across different code lengths.

ran codes provided by the authors and used the suggested or optimized parameters in all experiments.

Here similarity labels are defined by semantic-level labels. Images from the same category are considered semantically similar, and vice verse. Following the standard evaluation protocols [53, 129, 241, 250], each dataset is split into a large retrieval database and a small query set. In the previous work, a large amount of random samples from the retrieval database are used to train the hashing models [240, 250, 342, 432]. Since we are interested in the few-shot learning scenarios, we randomly sample small size training data to select (for our UGH) or learn (for baselines) hash codes. The number of training examples per category varies from 1 to 100 and the length of the hash codes varies from 8 to 32 bits. The retrieval performance on the query set is evaluated using mean average precision (mAP) and precision within Hamming radius 2 (Precision@2). To reduce the influence of random selection, all experiments are repeated ten times and the average mAP and precision are reported.

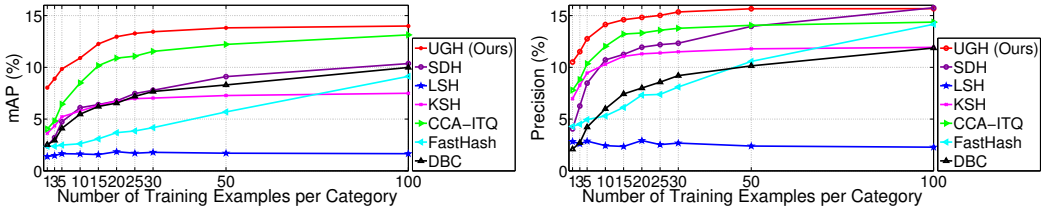
Comparisons with Supervised Hashing. Naturally, the first and most important question to answer is whether our UGH learned by unsupervised hyper-training indeed facilitates generalization to novel categories with few samples, compared to the state-of-the-art supervised hashing methods. We answer this question on the CIFAR10 benchmark [211]. This dataset consists of 60,000 images from 10 object classes, with 6,000 images per class. Following the standard practice [87], 50,000 images are used as the retrieval database and 10,000 images are used as the query set. This dataset is selected specifically for extensive evaluation and analysis. We include evaluation on more changeling datasets in the later sections.

Baselines. We compare against the state-of-the-art supervised hashing approaches, including CCA-ITQ [129], FastHash [240], SDH [342], KSH [250], and DBC [309]. DBC is the original supervised version of our approach. We also include the data-independent LSH as reference. These approaches can be viewed as *online binary codes*, as the hash functions are directly learned from the target dataset. In our preliminary experiments, we also tested the recent supervised deep hashing via training neural networks [431, 432]. These approaches typically require using the entire large-scale retrieval database for hash learning. With limited training data in our case, their performance is significantly inferior to that of other baselines which we reported. We thus did not include their results here.

Influence of training set size. First we evaluate the performance as a function of the number of training examples per category. The code length for all approaches is fixed as 16: our UGH selects 16 category-specific codes from the 20,000 hash library and the baselines directly learn the codes at length 16. Due to lack of public protocols for few-shot learning, we randomly sample $1, 3, 5, 10, 15, 20, 25, 30, 50$ and 100 images per category from the retrieval database as the training set. Figure 8.1 summarizes the average mAP and precision@2.

As shown in Figure 8.1, our UGH consistently outperforms all the other supervised hashing for small-sample learning. While the vanilla hashing approaches are over-fitting in this scenario, our universal binary representation, by leveraging large-scale unlabeled data, is effectively learned and transferable to novel categories. In addition to the unsupervised aspect, our code selection phase leads to both compact and discriminative codes for the target task, making it significantly different from LSH which typically requires long binary codes. To verify this, we tested random selection of the codes. While it is still better than the baselines, the performance drops. *e.g.*, in the one-shot case, the random selection achieved 18.42% mAP, which is better than 16.96% of CCA-ITQ (the best performing baselines) and is worse than 20.80% of our UGH with discriminative code selection.

An obvious advantage of UGH over its supervised counterpart DBC is that UGH makes it feasible to generate a large collection of hash codes based on unlabeled data. Another promising finding, based on Figure 8.1, is that UGH demonstrates more *expressive* and *universal* capability for novel categories with few samples compared to DBC. This verifies our assumption that information across categories is actually intrinsic in the data even without any supervision. One explanation is that by using another large-scale dataset (*e.g.*, Flickr-2M) apart from where the CNN feature is learned (ILSVRC), UGH would potentially prevent over-fitting and provide more generalization ability. This is similar to the case in which models are trained on the training dataset and their parameters are tuned on another validation dataset. More importantly, we introduce a series of sampling procedures in producing UGH to generate a diverse partition of the feature space in contrast to DBC (and other supervised hashing baselines). This leads to distributed representations, which is a crucial ingredient for generalization to new cases [34].



(a) (b)

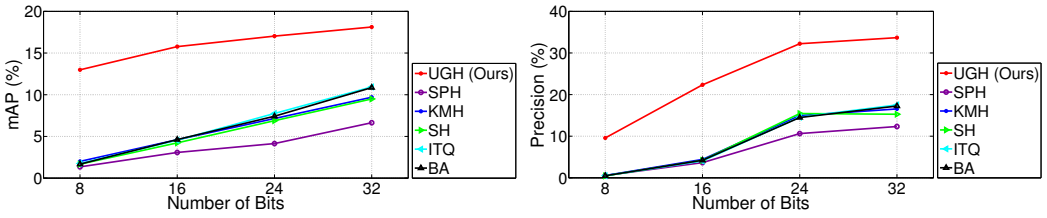
Figure 8.3: Performance comparisons between UGH and competing supervised hashing approaches for few-shot hash learning and image retrieval on the CIFAR100 dataset. X-axis: number of training examples per category. Y-axis: mean average precision (mAP) (Figure 8.3a) and precision@2 (Figure 8.3b). With the same code length 16, our UGH significantly outperforms these baselines for large-scale image retrieval tasks with few training samples.

Influence of code length. We also investigate the influence of the code length (the number of selected codes as the final descriptor) on the CIFAR10 dataset. 10 images per category are randomly selected as the training set; the retrieval database and query set remain as before. Figure 8.2 shows the mAP and precisioin@2 achieved by UGH and the supervised hashing baselines. These baselines directly learn the code at the desired length.

Figure 8.2 shows that UGH is more robust than other hashing competitors by maintaining very stable performance across increasing code lengths. This indicates the effectiveness of selecting category specific codes. CCA-ITQ tends to have good mAP performance; its precision@2, however, drops drastically with longer hash codes 32, which shows its inability to form compact clusters in the hash code space. A similar phenomenon is also observed for FastHash. Moreover, these conventional hashing approaches are restrictive in the sense that, for different code length, they need to re-learn the entire hash codes. On the contrary, the unsupervised, off-line and parallel aspects of our code generation mechanism makes it orders of magnitude faster and could be re-purposed for tasks with different desired code length. Even with additional code selection stage, training is efficient since the implementation of SVMs with binary codes could be greatly simplified and sped up by using a logical AND and a sparse summation for dot-products instead of floating-point calculations [39]. This favors such an approach to be used in ultra-large-scale scenarios.

Large-Scale Comparisons. We now move on to evaluate our UGH on the large-scale CIFAR100 dataset [211], which contains 100 categories with 600 images per category. Following the standard practice [269], we randomly select 100 images per category as the query set and use the remaining images as the retrieval database. Similar to the experimental setup as before, we focus on the influence of the training set size. Figure 8.3 summarizes the comparisons with the supervised baselines.

As shown in Figure 8.3, our UGH outperforms all the baselines by large margins in this large-scale scenario. The low performance of LSH suggests that CIFAR100 is more challenging than CIFAR10. In particular, Figure 8.3a shows that there is nearly 50% relative mAP performance boost in the one-shot learning case. Although SDH achieves comparable precision as our UGH when the number of samples is 100, it has a much lower mAP (3.63%



(a) (b)

Figure 8.4: Performance comparisons between UGH and competing unsupervised hashing approaches on the SUN-397 dataset. X-axis: code length. Y-axis: mean average precision (mAP) (Figure 8.4a) and precision@2 (Figure 8.4b). Both UGH and the baselines learn unsupervised hash codes over the pre-trained AlexNet features. The codes are learned on Flickr-2M and then tested on SUN-397. Our UGH consistently generalizes better than these baselines by large margins across different code lengths.

smaller) than ours. The improvements of UGH are more significant in the small sample size regime (*e.g.*, 1, 3 and 5), which is consistent with the observation on CIFAR10. While the state-of-the-art hash codes are learned separately for different target tasks, our UGH is inferred *once off-line* without knowing any target dataset, and generalizes well to the novel task without requiring additional, extensive hash training.

Comparisons with Unsupervised Hashing. Our approach estimates diverse pseudo-classes and learn hash functions that traverse across the low-density regions. Rather than simply due to additional unsupervised data, this hash learning scheme is crucial for its generalization across categories. To show this point, we further evaluate our UGH and unsupervised hashing on the SUN-397 dataset [414]. Following the standard practice [88], we use a subset which includes 42 categories with more than 500 images per category, leading to 35K images in total. The query set contains 4,200 images with 100 images per category randomly sampled from the dataset. The remaining images are used as the retrieval database. We focus on the influence of the code length.

Baselines. We compare against several state-of-the-art unsupervised hashing approaches, including PCA-ITQ [129], binary autoencoder (BA) [53], spectral hashing (SH) [406], spherical hashing (SPH) [156], and K-means hashing (KMH) [151], which are learned over the AlexNet features. Similar to our use case, they are now used in an *offline* manner, in which the codes are learned on Flickr and then tested on the target SUN-397.

Figure 8.4 shows that our UGH consistently achieves the best performance across different code lengths. This verifies that our generalization ability to novel tasks and categories comes not only from the generic CNN features, but also from the code generation mechanism. In addition to the initial Max-Min sampling that enforces diversity, our UGH introduces an additional expansion step to augment pseudo-categories with more data in a bootstrap manner, yielding more accurate sampling of the feature space structure. We further group the pseudo-categories into a set of abstract classes, leading to more generic hash codes. On the contrary, the existing unsupervised hashing approaches are proposed mainly for compression; with semantic information only coming from the input CNN features, their generalization ability is significantly limited.

Part III

Learning from Evolving Data Streams and Tasks: Rethinking Fine-Tuning

Each day learn something new, and just as important, relearn something old.

Robert Breault

Chapter 9

Developmental Learning: Fine-Tuning by Increasing Model Capacity

9.1 Motivation

We now consider learning novel categories from *a medium sized number of labeled examples*. The current *de facto standard* is to train a deep CNN on a large enough, diverse "base" set of data (*e.g.*, ImageNet), and then transfer this source CNN to target tasks [16, 226, 311, 427]. Fine-tuning is by far the dominant strategy under this paradigm [16, 126, 145, 283, 311, 422]. This approach was pioneered in [162] by transferring knowledge from a generative to a discriminative model, and has since been generalized with great success [126, 429]. The basic pipeline involves replacing the last "classifier" layer of a pre-trained network with a new randomly initialized layer for the target tasks of interest. The modified network is then fine-tuned with additional passes of appropriately tuned gradient descent on the target training set. Virtually *every* contemporary visual recognition system uses this pipeline. Even though its use is widespread, fine-tuning is still relatively poorly understood. For example, what fraction of the pre-trained weights actually change and how? More importantly, an open question remains how to best adapt a pre-trained CNN for novel categories/tasks.

To address these issues, in this chapter we explore **developmental neural networks** that grow in model capacity as new tasks as encountered. We demonstrate that growing a network, by adding additional units, facilitates knowledge transfer to new tasks. We explore two approaches to adding units as shown in Figure 7.1: going deeper (more layers) and wider (more channels per layer). Through visualizations, we demonstrate that these additional units help guide the adaptation of pre-existing units. Deeper units allow for new compositions of pre-existing units, while wider units allow for the discovery of complementary cues that address the target task. Due to their progressive nature, developmental networks still remain accurate on their source task, implying that they can learn without forgetting. Finally, we demonstrate that developmental networks particularly facilitate continual transfer across multiple tasks.

Our approach is loosely inspired by developmental learning in cognitive science. Humans, and in particular children, have the remarkable ability to continually transfer previously-

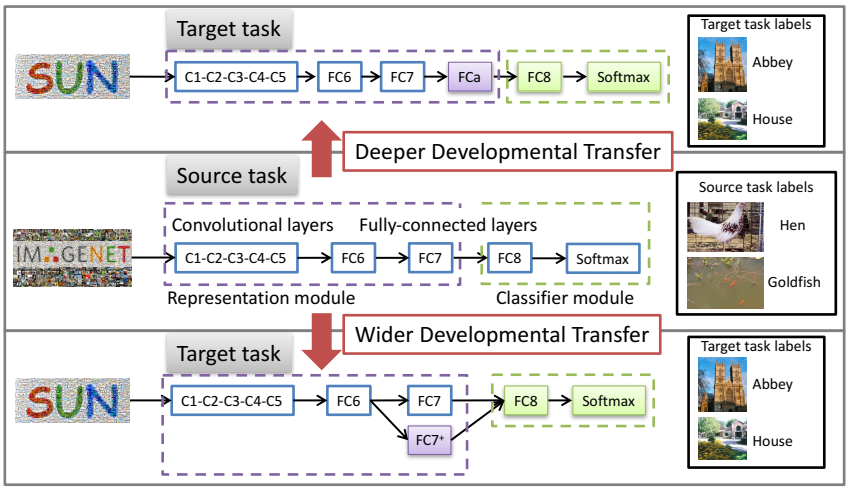


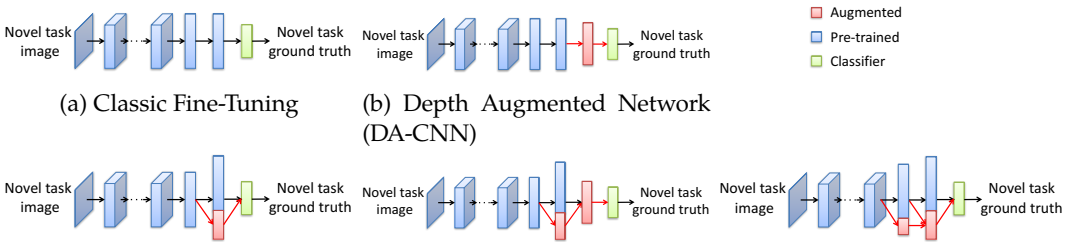
Figure 9.1: Transfer and developmental learning of pre-trained CNNs by increasing model capacity for the recognition of novel categories from few examples. The network (*e.g.*, AlexNet) is pre-trained on the source task (*e.g.*, ImageNet classification) with abundant data (middle row). Different from the dominant paradigm of fine-tuning a fixed-capacity model, we *grow this network* when adapting it to a novel target task (*e.g.*, SUN-397 scene classification) in two ways: (1) going deeper by adding more layers (top) and (2) going wider by adding more channels per layer (bottom).

acquired knowledge to novel scenarios. Much of the literature from both neuroscience [273] and psychology [174] suggests that such sequential knowledge acquisition is intimately tied with a child's growth and development.

Our contributions are three-fold. (1) We first demonstrate that the dominant paradigm of fine-tuning a fixed-capacity model is sub-optimal. (2) We explore several avenues for increasing model capacity, both in terms of going deeper (more layers) and wider (more channels per layer), and consistently find that increasing capacity helps, with a slight preference for widening. (3) We show that additional units must be normalized and scaled appropriately such that the "pace of learning" is balanced with existing units in the model. Finally, we use our analysis to build a relatively simple pipeline that "grows" a pre-trained model during fine-tuning, producing state-of-the-art results across a large number of standard and heavily benchmarked datasets (for scene classification, fine-grained recognition, and action recognition).

9.2 Approach Overview

Let us consider a CNN architecture pre-trained on a source domain with abundant data, *e.g.*, the vanilla AlexNet pre-trained on ImageNet (ILSVRC) with 1,000 categories [212,321]. We note in Figure 9.1 that the CNN is composed of a feature representation module $\mathcal{F}$ (*e.g.*, the five convolutional layers and two fully connected layers for AlexNet) and a classifier module $\mathcal{C}$ (*e.g.*, the final fully-connected layer with 1,000 units and the 1,000 -way softmax for ImageNet classification). Transferring this CNN to a novel task with limited training data



(a) Classic Fine-Tuning (b) Depth Augmented Network (DA-CNN)

(c) Width Augmented Network (WA-CNN) (d) Jointly Depth and Width Augmented Network (DWA-CNN) (e) Recursively Width Augmented Network (WWA-CNN)

Figure 9.2: Illustration of classic fine-tuning (a) and variations of our developmental networks with augmented model capacity (b–e).

(*e.g.*, scene classification of 397 categories from SUN-397 [414]) is typically done through fine-tuning [3, 15, 173].

In classic fine-tuning, the target CNN is instantiated and initialized as follows: (1) the representation module $\mathcal{F}_T$ is copied from $\mathcal{F}_S$ of the source CNN with the parameters $\Theta_T^{\mathcal{F}} = \Theta_S^{\mathcal{F}}$; and (2) a new classifier model $\mathcal{C}_T$ (*e.g.*, a new final fully-connected layer with 397 units and the 397 -way softmax for SUN-397 classification) is introduced with the parameters $\Theta_T^{\mathcal{C}}$ randomly initialized. All (or a portion of) the parameters $\Theta_T^{\mathcal{F}}$ and $\Theta_T^{\mathcal{C}}$ are fine-tuned by continuing the backpropagation, with a smaller learning rate for $\Theta_T^{\mathcal{F}}$. Because $\mathcal{F}_T$ and $\mathcal{F}_S$ have identical network structure, the representational capacity is fixed during transfer.

Our underlying thesis is that fine-tuning will be facilitated by *increasing representational capacity* during transfer learning. We do so by adding S new units $\{u_s\}_{s=1}^S$ into $\mathcal{F}_T$. As we will show later in our experiments, this significantly improves the ability to transfer knowledge to target tasks, particularly when fewer target examples are provided [373]. We call our architecture a *developmental network*, in which the new representation module $\mathcal{F}_T^* = \mathcal{F}_T \cup \{u_s\}_{s=1}^S$, and the classifier module remains $\mathcal{C}_T$.

Conceptually, new units can be added to an existing network in a variety of ways. A recent analysis, however, suggests that early network layers tend to encode generic features, while later layers tend to endode task-specific features [427]. Inspired from this observation, we choose to explore new units at later layers. Specifically, we either construct a completely new top layer, leading to a *depth augmented network* (DA-CNN) as shown in Figure 9.2b, or widen an existing top layer, leading to a *width augmented network* (WA-CNN) as shown in Figure 9.2c. We will explain these two types of network configurations in Section 9.3. Their combinations—a jointly depth and width augmented network (DWA-CNN) as shown in Figure 9.2d and a recursively width augmented network (WWA-CNN) as shown in Figure 9.2e—will also be discussed in Section 9.4.

9.3 Developmental Networks

For the target task, let us assume that the representation module $\mathcal{F}_T$ with fixed capacity consists of K layers $L_k, k = 1, \ldots, K$ with hidden activations $\boldsymbol{h}^k \in \mathcal{R}^{n_k}$, where n_k is the number of units at layer k . Let $\boldsymbol{W}^k$ be the weights between layer k and layer $k-1$. That is, $\boldsymbol{h}^k = f\left(\boldsymbol{W}^k \boldsymbol{h}^{k-1}\right)$, where $f(\cdot)$ is a non-linear function, such as ReLU. For notational

simplicity, $\boldsymbol{h}^k$ already includes a constant 1 as the last element and $\boldsymbol{W}^k$ includes the bias terms.

9.3.1 Depth Augmented Networks

A straightforward way to increase representational capacity is to construct a new top layer L_a of size S using $\{u_s\}_{s=1}^S$ on top of L_K , leading to the depth augmented representation module $\mathcal{F}_T^*$ as shown in Figure 9.2b. We view L_a as an adaptation layer that allows for novel compositions of pre-existing units, thus avoiding dramatic modifications to the pre-trained layers for their adaptation to the new task. The new activations $\boldsymbol{h}^a = f\left(\boldsymbol{W}^a \boldsymbol{h}^K\right)$ in layer L_a become the representation that is fed into the classifier module $\mathcal{C}_T$, where $\boldsymbol{W}^a$ denotes the weights between layers L_a and L_K .

9.3.2 Width Augmented Networks

An alternative way is to expand the network by adding $\{u_s\}_{s=1}^S$ to some existing layers while keeping the depth of the network fixed as shown in Figure 9.2c. Without loss of generality, we add all the units to the top layer L_K . Now the new top representation layer L_K^* consists of two blocks: the original L_K and the added L_K^+ with units $\{u_s\}_{s=1}^S$, leading to the width augmented representation module $\mathcal{F}_T^*$. The connection weights between L_K and the underneath layer L_{K-1} remains, *i.e.*, $\boldsymbol{h}^K{=}f\left(\boldsymbol{W}^K \boldsymbol{h}^{K-1}\right)$. We introduce additional lateral connection weights $\boldsymbol{W}^{K^+}$ between L_K^+ and L_{K-1} , which are randomly initialized, *i.e.*, $\boldsymbol{h}^{K^+}{=}f\left(\boldsymbol{W}^{K^+} \boldsymbol{h}^{K-1}\right)$. Finally, the concatenated activations $\left[\boldsymbol{h}^K, \boldsymbol{h}^{K^+}\right]$ of size $n_K{+}S$ from layer L_K^* are fed into the classifier module.

9.3.3 Learning at the Same Pace

Ideally, our hope is that the new and old units cooperate with each other to boost the target performance. For width augmented networks, however, the units start to learn at a different pace during fine-tuning: while the original units at layer L_k are already well learned on the source domain and only need a small modification for adaptation, the new set of units at layer L_k^+ are just set up through random initialization. They thus have disparate learning behaviors, in the sense that their activations generally have different scales. Naïvely concatenating these activations would restrict the corresponding units, leading to degraded performance and even causing collapsed networks, since the larger activations dominate the smaller ones [249]. Although the weights might adjust accordingly as fine-tuning processes, they require very careful initialization and tuning of parameters, which is dataset dependent and thus not robust. This is partially the reason that the previous work showed that network expansion was inferior to standard fine-tuning [238].

To reconcile the learning pace of the new and pre-existing units, we introduce an additional normalization and adaptive scaling scheme in width augmented networks, which is inspired by the recent work on combining multi-scale pre-trained CNN features from different layers [249]. More precisely, after weight initialization of $\mathcal{F}_T^*$, we first apply an $\mathcal{L}_2$ -norm normalization to the activations $\boldsymbol{h}^k$ and $\boldsymbol{h}^{k^+}$, respectively:

$$\widehat{\boldsymbol{h}}^k = \boldsymbol{h}^k / \left\|\boldsymbol{h}^k\right\|_2, \;\; \widehat{\boldsymbol{h}}^{k^+} = \boldsymbol{h}^{k^+} \Big/ \left\|\boldsymbol{h}^{k^+}\right\|_2 . \tag{9.1}$$

By normalizing these activations, their scales become homogeneous. Simply normalizing the norms to 1 slows down the learning and makes it hard to train the network, since the features become very small. Consistent with [249], we normalize them to a larger value (*e.g.*, 10 or 20), which encourages the network to learn well. We then introduce a scaling parameter γ for each channel to scale the normalized value as in [249]:

$$y_i^k = \gamma_i \widehat{h}_i^k, \;\; y_j^{k^+} = \gamma_j \widehat{h}_j^{k^+}. \tag{9.2}$$

which is motivated by batch normalization [177] and PReLU [153].

We found that for depth augmented networks, while this additional stage of normalization and scaling is not crucial, it is still beneficial. In addition, this stage only introduces negligible extra parameters, whose number is equal to the total number of channels. During fine-tuning, following [249], the derivatives with respect to the scaling factor γ and activations $\boldsymbol{h}$ are computed by backpropagation and chain rule. Let ℓ be the loss to minimize, *e.g.*, the softmax loss. We then have

$$\frac{\partial \ell}{\partial \widehat{\boldsymbol{h}}} = \frac{\partial \ell}{\partial \boldsymbol{y}} \cdot \gamma, \tag{9.3}$$

$$\frac{\partial \ell}{\partial \boldsymbol{h}} = \frac{\partial \ell}{\partial \widehat{\boldsymbol{h}}} \left(\frac{\boldsymbol{I}}{\|\boldsymbol{h}\|_2} - \frac{\boldsymbol{h}\boldsymbol{h}^T}{\|\boldsymbol{h}\|_2^3} \right), \tag{9.4}$$

$$\frac{\partial \ell}{\partial \gamma_i} = \sum_{y_i} \frac{\partial \ell}{\partial y_i} \widehat{h}_i. \tag{9.5}$$

The summation $\sum_{y_i}$ in Eqn. (9.5) runs over all positions of the feature map for channel i .

9.4 Experimental Evaluation

In this section, we explore the use of our developmental networks for transferring a pre-trained CNN to a number of supervised learning tasks with insufficient data, including scene classification, fine-grained recognition, and action recognition. We begin with extensive evaluation of our approach on scene classification of the SUN-397 dataset, focusing on the variations of our networks and different design choices. We also show that the network remains accurate on the source task. We then provide an in-depth analysis of fine-tuning procedures to qualitatively understand why fine-tuning with augmented network capacity outperforms classic fine-tuning. We further evaluate our approach on other novel categories and compare with state-of-the-art approaches. Finally, we investigate whether progressive augmenting outperforms fine-tuning a fixed large network and investigate how to cumulatively add new capacity into the network when it is gradually adapted to multiple tasks.

Implementation details. Following the standard practice, for computational efficiency and easy fine-tuning we use the Caffe [185] implementation of AlexNet [212], pre-trained on ILSVRC 2012 [321], as our reference network in most of our experiments. We found that our observations also held for other network architectures, such as VGG [347] and ResNet [154]. We also provide a set of experiment using VGG16 [347]. For the target tasks, we randomly initialize the classifier layers and our augmented layers. During fine-tuning, after resizing

the image to be 256×256 , we generate the standard augmented data including random crops and their flips as implemented in Caffe [185]. During testing, we only use the central crop, unless otherwise specified. For a fair comparison, fine-tuning is performed using stochastic gradient descent (SGD) with the "step" learning rate policy, which drops the learning rate in steps by a factor of 10 . The new layers are fine-tuned at a learning rate 10 times larger than that of the pre-trained layers (if they are fine-tuned). We use standard momentum 0.9 and weight decay 0.0005 without further tuning.

9.4.1 Evaluation and Analysis on SUN-397

We start our evaluation on scene classification of the SUN-397 dataset, a medium-scale dataset with around $108K$ images and 397 classes [414]. In contrast to other fairly small-scale target datasets, SUN-397 provides sufficient number of categories and examples while demonstrating apparent dissimilarity with the source ImageNet dataset. This greatly benefits our insight into fine-tuning procedures and leads to clean comparisons under controlled settings.

We follow the experimental setup in [3, 173], which uses a nonstandard train/test split since it is computationally expensive to run all of our experiments on the 10 standard subsets proposed by [414]. Specifically, we randomly split the dataset into train, validation, and test parts using 50% , 10% , and 40% of the data, respectively. The distribution of categories is uniform across all the three sets. We report 397 -way multi-class classification accuracy averaged over all categories, which is the standard metric for this dataset. We report the results using a single run due to computational constraints. Consistent with the results reported in [3, 173], the standard deviations of accuracy on SUN-397 classification are negligible, and thus having a single run should not affect the conclusions that we draw. For a fair comparison, fine-tuning is performed for around 60 epochs using SGD with an initial learning rate of 0.001 , which is reduced by a factor of 10 around every 25 epochs. All the other parameters are the same for all approaches.

Learning with Augmented Network Capacity. We first evaluate our developmental networks obtained by introducing a *single* new layer to deepen or expand the pre-trained AlexNet. For the depth augmented network (DA-CNN), we add a new fully connected layer FCa of size S^D on top of FC_7 whose size is $4{,}096$, where $S^D \in \{1{,}024, 2{,}048, 4{,}096, 6{,}144\}$. For the width augmented network (WA-CNN), we add a set of S^W new units as FC_7^+ to FC_7 , where $S^W \in \{1{,}024, 2{,}048\}$. After their structures are adapted to the target task, the networks then continue learning in four scenarios of gradually increasing the degree of fine-tuning: (1) "New": we only fine-tune the new layers, including the classifier layers and the augmented layers, while freezing the other pre-trained layers (*i.e.*, the off-the-shelf use case of CNNs); (2) " FC_7 –New": we fine-tune from the FC_7 layer; (3) " FC_6 –New": we fine-tune from the FC_6 layer; (4) "All": we fine-tune the entire network.

Table 9.1 summarizes the performance comparison with classic fine-tuning. The performance gap between our implementation of the fine-tuning baseline and that in [3, 173] is mainly due to different number of iterations: we used twice of the number of epochs in [3, 173] (30 epochs), leading to improved accuracy. Note that these numbers cannot be directly compared against other publicly reported results due to different data split. With relatively sufficient data, fine-tuning through the full network yields the best performance for all the approaches. Both our DA-CNN and WA-CNN significantly outperform the vanilla

Network	Type	Method	Acc (%)			
			New	FC_7–New	FC_6–New	All
AlexNet	Baselines	Finetuning-CNN	53.63	54.75	54.29	55.93
		[3,173]	48.4	—	51.6	52.2
	Single (Ours)	DA-CNN	54.24	56.48	57.42	58.54
		WA-CNN	**56.81**	56.99	57.84	58.95
	Combined (Ours)	DWA-CNN	56.07	56.41	56.97	57.75
		WWA-CNN	56.65	**57.10**	**58.16**	**59.05**
VGG16	Baselines	Finetuning-CNN	60.77	59.09	50.54	62.80
	Single (Ours)	DA-CNN	61.21	62.85	63.07	65.55
		WA-CNN	**63.61**	**64.00**	**64.15**	**66.54**

Table 9.1: Performance comparisons of classification accuracy (%) between the variations of our developmental networks *with augmented model capacity* and classic fine-tuning *with fixed model capacity* on scene classification of the SUN-397 dataset. The variations include: (1) for AlexNet, depth augmented network (DA-CNN), width augmented network (WA-CNN), jointly depth and width augmented network (DWA-CNN), and recursively width augmented network (WWA-CNN); and (2) for VGG16, DA-CNN and WA-CNN. Both our networks and the baselines are evaluated in four scenarios of gradually increasing the degree of fine-tuning, including fine-tuning only new layers, from FC_7 to new layers, from FC_6 to new layers, and the entire network. Ours *consistently and significantly* outperform the vanilla fine-tuned CNN for both AlexNet and VGG16 CNNs *in all these scenarios*. This shows the generality of our approach.

fine-tuned CNN *in all the different fine-tuning scenarios*. This verifies the effectiveness of increasing model capacity when adapting it to a novel task. While they have achieved comparable performance, WA-CNN slightly outperforms DA-CNN.

Increasing the Network Capacity through Combination or Recursion. Given the promise of DA-CNN and WA-CNN, we further augment the network by making it both deeper and wider or two-layer wider. For the jointly depth and width augmented network (DWA-CNN) (Figure **??**), we add FC_a of size S^{DW} on top of FC_7 while expanding FC_7 using FC_7^+ of size S^{DW} , where $S^{DW} \in \{1{,}024, 2{,}048\}$. For the recursively width augmented network (WWA-CNN) (Figure **??**), we both expand FC_7 using FC_7^+ of size S_7^{WW} and FC_6 using FC_6^+ of size S_6^{WW} , where $S_7^{WW} \in \{1{,}024, 2{,}048, 4{,}096\}$ and S_6^{WW} is half of S_7^{WW} .

We compare DWA-CNN and WWA-CNN with DA-CNN and WA-CNN in Table 9.1. The two-layer WWA-CNN generally achieves the best performance, indicating the importance of augmenting model capacity at different and complementary levels. The jointly DWA-CNN lags a little bit behind the purely WA-CNN. This implies different learning behaviors when we make the network deeper or wider. Their combination thus becomes a non-trivial task.

Diagnostic Analysis. While we summarize the best performance in Table 9.1, a diagnostic experiment in Table 9.2 on the number of augmented units S^D , S^W , S^{DW} , and S^{WW} shows that *all of these variations of network architectures significantly outperform classic fine-tuning*, in-

Method	Configuration	New	FC_7–New	FC_6-New	All
DA-CNN	FC_a–1,024	53.36	56.31	57.22	57.98
	FC_a–2,048	53.82	56.47	57.14	58.07
	FC_a–4,096	54.02	56.46	57.41	58.32
	FC_a–6,144	54.24	56.48	57.42	58.54
WA-CNN	FC_7^+–1,024	56.46	56.71	57.55	58.90
	FC_7^+–2,048	56.81	56.99	57.84	58.95
DWA-CNN	FC_7^+–1,024–FC_a–1,024	55.44	55.77	56.71	57.49
	FC_7^+–2,048-FC_a–2,048	56.07	56.41	56.97	57.75
WWA-CNN	FC_6^+–512–FC_7^+–1,024	56.13	57.10	57.65	58.80
	FC_6^+–1,024–FC_7^+–2,048	56.49	57.10	57.98	59.05
	FC_6^+–2,048–FC_7^+–4,096	56.65	57.03	58.16	58.98

Table 9.2: Diagnostic analysis of classification accuracy (%) for the variations of our developmental network (including DA-CNN, WA-CNN, DWA-CNN, and WWA-CNN) with different number of new units on SUN-397. Our networks demonstrate consistent improvements over the conventional fine-tuning, indicating the generality and robustness of augmenting model capacity for learning a novel task.

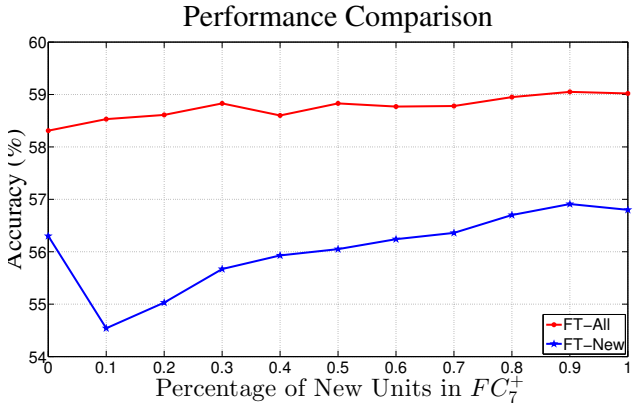


Figure 9.3: Analysis of unit allocation for two-layer width augmented networks that expand both FC_7 and FC_6 with a total of 2,000 new units on SUN-397. X-axis: percentage of the new units allocated to FC_7^+. Y-axis: multi-class classification accuracy. We evaluate in the fine-tuning "New" and "All" scenarios. The optimal pattern is a spread allocation where the higher layer FC_7^+ takes the majority of new units.

dicating the robustness of our approach. We found that this observation was also consistent with other datasets, which we evaluated in the later section. Overall, the performance increases with the augmented model capacity (represented by the size of augmented layers), although the performance gain diminishes with the increasing number of new units.

Method	Scaling	New	FC_7–New	FC_6–New	All
DA-CNN	*w/o*	**53.82**	**56.47**	56.25	57.21
FC_a–2,048	*w/*	53.51	56.15	**57.14**	**58.07**
WA-CNN	*w/o* (rand)	53.78	54.66	49.72	51.34
FC_7^+–2,048)	*w/o* (copy+rand)	53.62	54.35	53.70	55.31
	w/	**56.81**	**56.99**	**57.84**	**58.95**

Table 9.3: Performance comparisons of classification accuracy (%) for our depth (DA-CNN) or width (WA-CNN) augmented network *with and without* introducing normalization and scaling on SUN-397. The number of new units in FC_a for DA-CNN or in FC_7^+ for WA-CNN is generally $2{,}048$. Our normalization and scaling strategy reconciles the learning pace of new and old units, and thus greatly benefits both types of networks, *in particular WA-CNN*.

Allocation of Units under a Size Budget. An interesting question arises from the above analysis: given a budget of fixed number of new units, what is an optimal pattern of unit allocation to different layers? We analyze this issue in the scenario of two-layer width augmented network since it has achieved the best performance. Specifically, we expand both FC_7 and FC_6 using a total of $2{,}000$ new units, *i.e.*, $S_7^{WW} + S_6^{WW} = 2{,}000$. We change the size of the expanded layers FC_7^+ and FC_6^+ linearly in a step size of 200 while satisfying the constraint of the total number of units, resulting in 11 WWA-CNN architectures. We then fine-tune them in the "New" and "All" scenarios.

Figure 9.3 shows that all of these network variations are beneficial and outperform conventional fine-tuning, which is consistent with the results in Table 9.2. Importantly, diffusion of new units across both FC_7 and FC_6 leads to the best performance. This suggests that a better strategy when augmenting the capacity is to expand somehow at different layers rather than focus on a sole layer. In addition, the augmented structure that achieves the best performance demonstrates the shape of inverted triangle, in which the higher layer FC_7^+ takes the majority of new units. This is partially because FC_6 is more generic to different tasks while FC_7 is more specific to the original task, making it require more capacity (new units) to represent the novel target.

Importance of Reconciling the Learning Pace of New and Old Units. The previous work showed that network expansion did not introduce additional benefits [238]. We argue that its unsatisfactory performance is because of the failure of taking into account the different learning pace of new and old units. After exploration of different strategies, such as initialization, we found that the performance of a width augmented network significantly improves by a simple normalization and scaling scheme when concatenating the pre-trained and expanded layers. This issue is investigated for both types of model augmentation in Table 9.3. The number of new units is generally $2{,}048$; in the case of copying weights of the pre-trained FC_7 and then adding random noises as initialization for FC_7^+ , we use $4{,}096$ new units.

For WA-CNN, if we naïvely add new units without considering scaling, Table 9.3 shows that the performance is either only marginally better or even worse than classic fine-tuning (when fine-tuning more aggressively) in Table 9.1. This is consistent with the observation

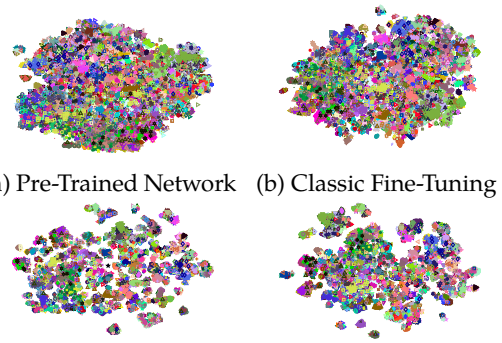
made in [238]. However, once the learning pace of the new and old units is re-balanced by scaling, WA-CNN exceeds the baseline by a large margin. For DA-CNN, directly adding new units without scaling already greatly outperforms the baseline, which is consistent with the observation in [283], although scaling provides additional performance gain. This suggests slightly different learning behaviors for depth and width augmented networks. When a set of new units are added to form a purely new layer, they have relatively more freedom to learn from scratch, making the additional scaling *beneficial yet inessential*. When the units are added to expand a pre-trained layer, however, the constraints from the synergy require them to learn to collaborate with the pre-existing units, which is explicitly achieved by the additional scaling.

Evaluation with the VGG16 Architecture. Table 9.1 also summarizes the performance of DA-CNN and WA-CNN using VGG16 [347] and shows the generality of our approach. Due to GPU memory and time constraints, we reduce the batch size and perform fine-tuning for around 30 epochs using SGD. All the other parameters are the same as before. Also, following the standard practice in Fast R-CNN [125], we fine-tune from the layer $Conv2\_1$ in the "All" scenario.

Learning without Forgetting. Conceptually, due to their developmental nature, our networks should remain accurate on their source task. Table 9.4 validates such ability of *learning without forgetting* by showing their classification performance on the source ImageNet dataset.

Type	Method	Acc (%)
Oracle	*ImageNet-AlexNet*	*56.9*
References	LwF [238]	55.9
	Joint [238]	56.4
Ours	DA-CNN	55.3
	WA-CNN	51.5

Table 9.4: Demonstration of the ability of *learning without forgetting* on the source (ImageNet) ILSVRC 2012 validation set. For our DA-CNN and WA-CNN that are fine-tuned on SUN-397, we re-fine-tune on the source ILSVRC 2012 training set, *i.e.*, re-training a new 1,000-way classifier layer and fine-tuning the augmented layers. We show the results of the oracle (*i.e.*, the original AlexNet) and the approaches that are *specifically designed* to preserve the performance on the source task during transfer [238] as references. While our approach focuses on improving the performance on the target task, it remains accurate on the source task. In addition, the existing approaches [238] can be naturally incorporated into our approach to further improve the performance on both source and target tasks.



(a) Pre-Trained Network (b) Classic Fine-Tuning

(c) Depth Augmented Network (DA-CNN) (d) Width Augmented Network (WA-CNN)

Figure 9.4: t-SNE visualizations of the top feature layers on the SUN-397 validation set. DA-CNN and WA-CNN show significantly better semantic separations. **Best viewed in color with zoom.**

9.4.2 Understanding of Fine-Tuning Procedures

We now analyze the fine-tuning procedures from various perspectives to gain insight into how fine-tuning modifies the pre-trained network and why it helps by increasing model capacity. We evaluate on the SUN-397 validation set. For a clear analysis and comparison, we focus on DA-CNN and WA-CNN, both with $2{,}048$ new units.

Feature Visualization

To roughly understand the topology of the feature spaces, we visualize the features using the standard t-SNE algorithm [384]. As shown in Figure 9.4, we embed the $4{,}096$ -dim FC_7 features of the pre-trained and fine-tuned networks, the $6{,}144$ -dim wider $FC_7 + FC_7^+$ features, and the $2{,}048$ -dim deeper FC_a features into a 2 -dim space, respectively, and plot them as points colored depending on their semantic category. While classic fine-tuning somehow improves the semantic separation of the pre-trained network, both of our networks demonstrate significantly clearer semantic clustering structures, which is compatible with their improved classification performance.

Maximally Activating Images

To further analyze how fine-tuning changes the feature spaces, we retrieve the top-5 images that maximally activate some unit as in [126]. We first focus on the *common units* in FC_7 of the pre-trained, fine-tuned, and width augmented networks. In addition to using the SUN-397 images, we also include the maximally activating images from the ILSVRC 2012 validation set for the pre-trained network as references. Figure 9.5 shows an interesting transition: while the pre-trained network learns certain concentrated concept specific to the source task (left), such concept spreads over as a mixture of concepts for the novel target task (middle left). Fine-tuning tries to re-centralize one of the concepts suitable to the target task, but with limited capability (middle right). Our width augmented network facilitates such re-centralization, leading to more discriminative patterns (right). Similarly, we illustrate the

Figure 9.5: Top 5 maximally activating images for eight FC_7 units. From left to right: ILSVRC 2012 validation images for the pre-trained network, and SUN-397 validation images for the pre-trained, fine-tuned, and width augmented (WA-CNN) networks. Each row of images corresponds to a common unit from these networks, indicating that our WA-CNN facilitates the specialization of the pre-existing units towards the novel target task. For example, the bottom row shows a transition from a parallel pattern in the pre-trained ImageNet network to several mixed concepts in the fine-tuned network, and finally to a pattern converging to vanishing points in our SUN-397 WA-CNN.

maximally activating images for units in FC_a of the depth augmented network in Figure 9.6, which shows quite different behaviors. Compared with the object-level concepts in the width augmented network, the depth augmented network appears to have the ability to model a large set of compositions of the pre-trained features and thus generates more scene-level, better clustered concepts.

Additional visualizations of maximally activating images are in Figure 9.7 and Figure 9.8 for the width augmented network and in Figure 9.9 and Figure 9.10 for the depth augmented network.

9.4.3 Generalization to Other Tasks and Datasets

We now evaluate whether our developmental networks facilitate the recognition of other novel categories. We compare with publicly available baselines and report multi-class classification accuracy. While the different variations of our networks outperform these baselines, we mainly focus on the width augmented networks (WA-CNN).

Tasks and datasets. We evaluate on standard benchmark datasets for scene classification: MIT-67 [375], for fine-grained recognition: Caltech-UCSD Birds (CUB) 200-2011 [389] and Oxford 102 Flowers [279], and for action recognition: Stanford-40 actions [425]. These datasets are widely used for evaluating the CNN transferability [15], and we consider their diversity and coverage of novel categories. We follow the standard experimental setup (*e.g.*, the train/test splits) for these datasets as detailed in Chapter 3.

Baselines. While comparing with classic fine-tuning is the fairest comparison, to show the superiority of our approach, we also compare against other baselines that are specifically designed for certain tasks. For a fair comparison, we focus on the approaches that use single

Figure 9.6: Top 5 maximally activating images from the SUN-397 validation set for fourteen FC_a units of the depth augmented network (DA-CNN). Each row of 5 images in the left and right columns corresponds to a unit, respectively, which is well aligned to a scene-level concept for the target task.

scale AlexNet CNNs. Importantly, our approach can be also combined with other CNN variations (*e.g.*, VGG-CNN [347], multi-scale CNN [145, 422]) for further improvement.

Table 9.5 shows that our approach achieves state-of-the-art performance on these challenging benchmark datasets and significantly outperforms classic fine-tuning by a large margin. In contrast to task customized CNNs that are only suitable for particular tasks and categories, the consistently superior performance of our approach suggests that it is generic for a wide spectrum of tasks.

9.5 A Single Universal Higher Capacity Model?

One interesting question is that our results could imply that standard models should have used higher capacity even for the source task (*e.g.*, ImageNet). To examine this, we explore progressive widening of AlexNet (WA-CNN). Specifically, in the source domain, Table 9.6 shows that progressive widening of a network outperforms a fixed wide network trained from scratch. More importantly, in the target domain, Table 9.7 shows that our progressive widening *significantly outperforms* fine-tuning a fixed wide network.

Type	MIT-67		102 Flowers		CUB200-2011		Stanford-40	
	Approach	Acc(%)	Approach	Acc(%)	Approach	Acc(%)	Approach	Acc(%)
ImageNet CNNs	Finetuning-CNN	61.2	Finetuning-CNN	75.3	Finetuning-CNN	62.9	Finetuning-CNN	57.7
	Caffe [422]	59.5	CNN-SVM [311]	74.7	CNN-SVM [311]	53.3	Deep Standard [16]	58.9
	—	—	CNNaug-SVM [311]	86.8	CNNaug-SVM [311]	61.8	—	—
Task Customized CNNs	Caffe-DAG [422]	64.6	LSVM [301]	87.1	LSVM [301]	61.4	Deep Optimized [16]	66.4
	—	—	MsML+ [301]	89.5	DeCaf+DPD [91]	65.0	—	—
	Places-CNN [437]	**68.2**	MPP [426]	91.3	MsML+ [301]	66.6	—	—
	—	—	Deep Optimized [16]	91.3	MsML+* [301]	67.9	—	—
Data Augmented CNNs	Combined-AlexNet [190]	58.8	Combined-AlexNet [190]	83.3	—	—	Combined-AlexNet [190]	56.4
Multi-Task CNNs	Joint [238]	63.9	—	—	Joint [238]	56.6	—	—
	LwF [238]	64.5	—	—	LwF [238]	57.7	—	—
Ours	WA-CNN	66.3	WA-CNN	**92.8**	WA-CNN	**69.0**	WA-CNN	**67.5**

Table 9.5: Performance comparisons of classification accuracy (%) between our developmental networks (WA-CNN) and the previous work for scene classification, fine-grained recognition, and action recognition. We roughly divide the baselines into four types: (1) ImageNet CNNs, which post-process the off-the-shelf CNN or fine-tune it in a standard manner; (2) task customized CNNs, which modify a standard CNN for a particular target task (*e.g.*, for MIT-67, Places-CNN trains a customized CNN on the Places dataset with 400 scene categories [437]); (3) data augmented CNNs, which concatenate features from the ImageNet AlexNet and an additional CNN trained on 100 million Flickr images in a weakly supervised manner [190]; (4) multi-task CNNs, which (approximately) train a CNN jointly from both the source and target tasks. Ours show consistently superior performance and generality for a wide spectrum of tasks.

Dataset	CNN	WA-CNN-scratch	WA-CNN-grow (Ours)
ImageNet	56.9	57.6	**57.8**

Table 9.6: Performance comparisons of classification accuracy (%) on the source dataset between a standard AlexNet (CNN), a wide AlexNet trained from scratch (WA-CNN-scratch), and a wide network trained progressively by fine-tuning on the source task itself (WA-CNN-grow). Progressive learning appears to help even on the source task.

Dataset	CNN-FT	WA-CNN-ori	WA-CNN-grow (Ours)
MIT-67	61.2	62.3	**66.3**
CUB200-2011	62.9	63.2	**69.0**

Table 9.7: Performance comparisons of classification accuracy (%) on the target datasets between standard fine-tuning of a standard AlexNet (CNN-FT), standard fine-tuning of a wide AlexNet (WA-CNN-ori), and fine-tuning by progressive widening of a standard AlexNet (WA-CNN-grow). With the same model capacity, WA-CNN-grow significantly outperforms WA-CNN-ori. See Figure 9.11 for a discussion.

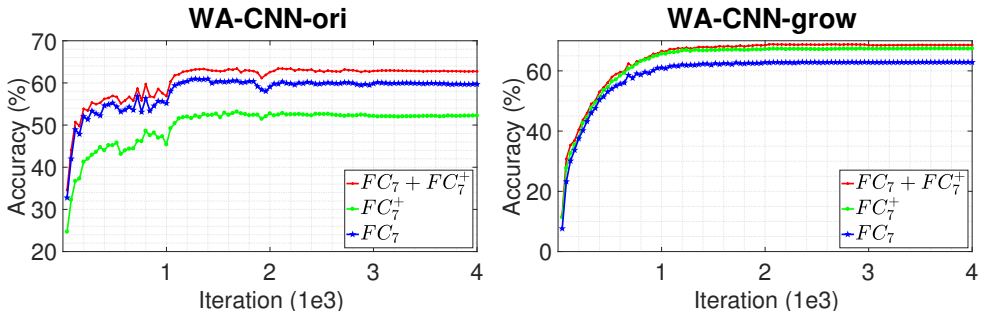


Figure 9.11: Learning curves of separate FC_7 and FC_7^+ and their combination for WA-CNN on the CUB200-2011 test set. Left and Right show different learning behaviors: the FC_7^+ curve is *below* the FC_7 curve for WA-CNN-ori, and *above* for WA-CNN-grow. Units in WA-CNN-ori appear to *overly-specialize* to the source, while the new units in WA-CNN-grow appear to be *diverse experts* better tuned for the novel target task. Interestingly, these experts allow for better adaptation of pre-existing and new units (Figure 9.12).

Figure 9.12: Top 5 maximally activating CUB200-2011 images for a representative FC_7 unit (1st row) and an FC_7^+ unit (2nd row). Each row of images corresponds to a common unit from two networks: WA-CNN-ori (left) and WA-CNN-grow (right). Compared to WA-CNN-ori, WA-CNN-grow facilitates the adaptation of pre-existing and new units towards the novel task by capturing discriminative patterns (top: birds in water; bottom: birds with yellow belly).

9.5.1 Cooperative Learning

Figure 9.11 and Figure 9.12 provide an in-depth analysis of the cooperative learning behavior between the pre-existing and new units and show that developmental learning appears to regularize networks in a manner that encourages diversity of units.

9.5.2 Continual Transfer across Multiple Tasks

Transfer across multiple tasks or datasets is of interest in real-world scenarios. Our approach is in particular suitable for such scenarios since we are able to cumulatively increase the model capacity and thus transfer knowledge from a series of previous tasks for the new one. In Table 9.8, we focus on MIT-67 as the target task. According to the similarity measure between the source and target datasets [16, 70], SUN-397 is more similar to the source ImageNet dataset than MIT-67. Our approach is suitable for such multi-task transfer since

	WA-CNN (Ours)		Baselines	
Scenario	ImageNet→MIT67	ImageNet→SUN→MIT67	Places [437]	ImageNet-VGG [238]
Acc(%)	66.3	**79.3**	68.2	74.0

Table 9.8: Through progressive growing via SUN-397, a widened AlexNet significantly improves the performance on MIT-67, and even outperforms fine-tuning a Places AlexNet that is directly trained on the Places dataset with 400 scene categories [437] and fine-tuning a fixed ImageNet VGG16 with higher capacity by a large margin.

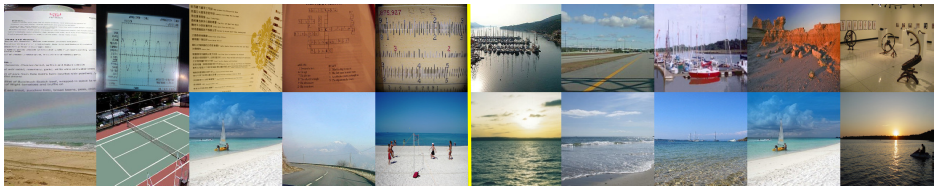
it facilitates *smooth transfer across tasks*, *i.e.*, gradual transfer starting from target tasks that are similar to the source task and then moving to target tasks that are dissimilar to the source task. Specifically, using our width augmented network (WA-CNN), from the pre-trained ImageNet AlexNet, we first add $2{,}048$ new units as FC_7^+ and transfer on SUN-397; we then further add $1{,}024$ new units as FC_7^{++} and transfer on MIT-67. Table 9.8 shows that by leveraging additional data from SUN-397 and shared knowledge across multiple tasks, the performance on the target MIT-67 significantly improves. In particular, the performance *even outperforms Places AlexNet that is directly trained using the Places dataset with* 400 *scene categories [437] and outperforms more powerful ImageNet VGG that is directly transferred on MIT-67 [238] by a large margin*. Our approach is in particular suitable for *continual, smooth* transfer across multiple tasks since we are able to cumulatively increase model capacity as demonstrated in Table 9.8.

Unit a

Unit b

Unit c

Unit d



Unit e

Figure 9.7: Top 5 maximally activating images for five FC_7 units (from top to bottom). For each unit, top left: 5 ILSVRC 2012 validation images for the pre-trained network as reference; top right: 5 SUN-397 validation images for the pre-trained network; bottom left: 5 SUN-397 validation images for the conventional fine-tuned network; and bottom right: 5 SUN-397 validation images for our width augmented network (WA-CNN). Each set of images in the sub-figure correspond to a common unit from these networks, indicating a transition that shows how fine-tuning with fixed and augmented capacity changes the pre-trained network. In particular, our WA-CNN *facilitates the specialization of the pre-existing units towards the novel target task*. *e.g.*, unit e shows a transition from a menu-like smooth surface in the pre-trained ImageNet network to several mixture of concepts in the fine-tuned network, and finally to a ocean-sky-view-like smooth surface in our SUN-397 WA-CNN.

Unit f

Unit g

Unit h

Unit i

Unit j

Figure 9.8: Top 5 maximally activating images for five FC_7 units (from top to bottom). For each unit, top left: 5 ILSVRC 2012 validation images for the pre-trained network as reference; top right: 5 SUN-397 validation images for the pre-trained network; bottom left: 5 SUN-397 validation images for the conventional fine-tuned network; and bottom right: 5 SUN-397 validation images for our width augmented network (WA-CNN). Each set of images in the sub-figure correspond to a common unit from these networks, indicating a transition that shows how fine-tuning with fixed and augmented capacity changes the pre-trained network. In particular, our WA-CNN *facilitates the specialization of the pre-existing units towards the novel target task*. *e.g.*, unit f shows a transition from a penguin-like vertically repeated pattern in the pre-trained ImageNet network to several mixture of concepts in the fine-tuned network, and finally to a wardrobe-like vertically repeated pattern in our SUN-397 WA-CNN.

Figure 9.9: Top 5 maximally activating images from the SUN-397 validation set for twenty-four FC_a units in our depth augmented network (DA-CNN). From the left to right column, each row of 5 images correspond to a unit. Different from the object-level concepts discovered in the width augmented network, DA-CNN appears to have the ability to *model a large set of new compositions of pre-existing units and thus generates more scene-level, better clustered concepts towards the novel target task*, *e.g.*, auditorium and veterinary room in the first row.

Figure 9.10: Top 5 maximally activating images from the SUN-397 validation set for twenty-four FC_a units in our depth augmented network (DA-CNN). From the left to right column, each row of 5 images correspond to a unit. Different from the object-level concepts discovered in the width augmented network, DA-CNN appears to have the ability to *model a large set of new compositions of pre-existing units and thus generates more scene-level, better clustered concepts towards the novel target task*, *e.g.*, bullring and castle in the first row.

Chapter 10

Factorized Convolutional Networks: Unsupervised Fine-Tuning for Image Clustering

10.1 Motivation

In the conventional paradigm and our modification in Chapter 9, fine-tuning requires *annotated* target data, and we use the term "supervised fine-tuning" to refer it. However, in scenarios where there are no labeled images for novel categories or tasks (*e.g.*, in image clustering applications), such supervised fine-tuning is inapplicable and how to best adapt a pre-trained CNN still remains an open challenge. Hence, we propose "unsupervised fine-tuning" *as a new paradigm* to address this issue.

To this end, we transfer knowledge *from a discriminative to a generative model* and explore "factorized convolutional networks" (FCNs) that *fine-tune the pre-trained CNN representations in an unsupervised manner*. Given unlabeled target images, a factorization of the CNN representations is learned using low-rank and group-sparsity constraints. Inspired by the success of non-negative matrix factorization (NMF) [227, 402] in clustering applications, we introduce a novel NMF based adaptation module with a generative loss that can be plugged into any standard CNN to facilitate the desired unsupervised transfer. As a classic multivariate analysis technique, the appeal of NMF is the ability to disentangle exploratory factors of variations underlying unlabeled, non-negative data samples as well as the inherent clustering property. Intuitively, the CNN activations of interest are those after the rectified linear units (ReLUs), which consistently show better recognition performance for various tasks and which are also non-negative. It is thus natural to investigate NMF techniques on top of CNN activations for image clustering, as shown in Figure 10.1.

More precisely, our key insight is to effectively adapt between the source and target tasks by both utilizing generic statistics learned from a large corpus of labeled source images through CNNs and separating out the current underlying factors of variation relevant to the observed, unlabeled target data via NMF. To better select a group of correlated CNN activations, we propose a variant of NMF — group-sparse NMF (GSNMF), which identifies a rich set of informative and discriminative latent variables across tasks. Given that NMF/GSNMF could also be interpreted as a two-layer neural network [227], our GSNMF

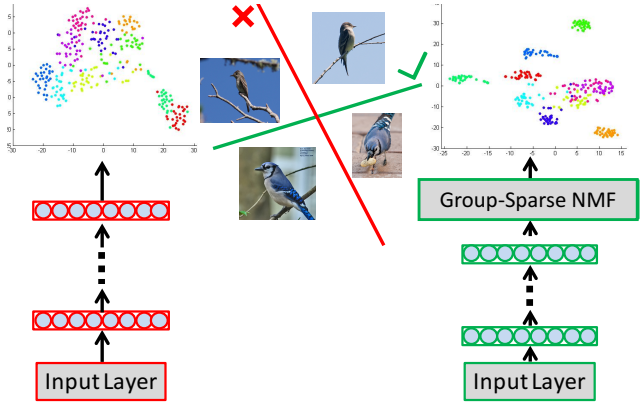


Figure 10.1: Unsupervised transfer of pre-trained CNN representations to novel target tasks with *unlabeled data* via a factorized convolutional network (FCN). Off-the-shelf features that are extracted from CNNs pre-trained on ImageNet are limited to describing subtle differences among novel fine-grained categories, as visualized by embedding the features in a 2-dim space via t-SNE [384] (**left**). By leveraging group-sparse non-negative matrix factorization (GSNMF), *unsupervised fine-tuning* is accomplished and features extracted from the resulting GSNMF-FCN model lead to more discriminative clusters (**right**). We thus learn a better representation with enhanced transferability for target tasks with unlabeled data, in which conventional supervised fine-tuning with back-propagation is inapplicable.

based FCN is then regarded as a principled feed-forward model. This allows to fine-tune the resulting augmented architecture (*i.e.*, modifying the CNN parameters as well) on the target task with respect to a NMF based objective using stochastic gradient descent and back-propagation.

Our contributions are four-fold. (1) Different from the conventional strategy that transfers knowledge from a generative to a discriminative model [162], we propose a novel way of CNN transfer — supervised, discriminative pre-training and then unsupervised, generative fine-tuning. (2) Based on this general principle, we show how factorized convolutional networks (FCNs), which combine NMF and pre-trained CNN, learn a more generic feature representation across tasks. (3) We show how to explicitly enforce group-sparsity on FCN to better leverage the correlation of CNN activations by introducing elastic net regularization into NMF. (4) Our unsupervised fine-tuning is general; it could be used in image clustering tasks and also used as unsupervised initialization to further improve classification tasks. Finally, to the best of our knowledge, we are the first to evaluate the performance of image clustering on challenging large-scale scene and fine-grained recognition datasets, producing state-of-the-art results.

10.2 Unsupervised Feature Learning and Image Clustering

Unsupervised feature learning. Unsupervised feature learning focuses on discovering low-dimensional features that capture some structure underlying the high-dimensional unlabeled data. Classic approaches include principal component analysis (PCA) [188], inde-

pendent component analysis (ICA) [175], and locally linear embedding (LLE) [320]. Inspired by the hierarchical architecture of the neural system, many new schemes that stack multiple layers of simple learning blocks, such as sparse coding [230], restricted Boltzmann machines (RBMs) [161], auto-encoders [131], and NMF [227], have been proposed to build deep representations [231, 403]. One similar work is the deep semi-NMF model, which stacks semi-NMF together to learn low-dimensional feature representations [377]. Another similar work is the deep linear discriminant analysis model, which projects high-dimensional observations to linearly separable representations [93].

Different from the previous work, we combine a NMF layer with a pre-trained deep CNN and use the reconstruction error as the objective function to fine-tune the network for unsupervised learning. We extend [139] in three important ways. (1) [139] is pipelined, while ours is end-to-end. [139] simply applies NMF on top of *off-the-shelf* CNN features, in which NMF reduces the dimension of *fixed* CNN features. In contrast, ours is more general and introduces NMF (and its variant) as a feature reconstruction (generative) loss for unsupervised CNN fine-tuning. Ours thus integrates NMF and CNN as a principled feed-forward network, and allows for fine-tuning the full network with back-propagation. As shown in the following sections, we not only learn the NMF adaptation layers, but also *modify (a portion of) the pre-trained CNN weights using the generative loss* towards the target task. Due to the end-to-end nature, ours is more flexible and achieves better performance. (2) [139] can only deal with image clustering. In contrast, due to the end-to-end nature, ours could be also used as *unsupervised initialization* and *improves image classification* on target tasks. (3) We have substantially extended experimental results, including more datasets, more baselines, different clustering techniques (k-means and spectral clustering), additional hyper-parameter analysis, and ablation analysis.

Domain adaptation and transfer learning. Another related line of work focuses on standard domain adaptation with the assumption that the data from source and target datasets share the same set of categories but have shifted distributions [120, 122]. Our work, however, does not have this assumption and addresses a more general, challenging task (*i.e.*, different but relevant source and target categories/tasks). The learning processes are different as well. [120, 122] explicitly use the source labels to infer the target labels. In contrast, we transfer a pre-trained (ImageNet) network to unsupervised target tasks and do not use the source labels in this process. Besides, [120, 122] are evaluated on image classification tasks while our work is mainly evaluated on image clustering tasks. The reconstruction loss used in [122] is also different: [122] simply reconstructs target raw images with the mean squared error loss, whereas we reconstruct the learned CNN features and leverage their non-negativity. More recently, a recurrent network with a single loss function is proposed to guide the agglomerative clustering [420]. While this work uses different network architectures for different datasets to train a dataset specific model, it fails to address the subtle difference among fine-grained categories. Different from this work, our model uses the same parameters and network architecture for all datasets, leading to a more universal feature representation.

In transfer learning, the target task is different from but related to the source task [288], such as transfer from object-centric source categories to scene-centric target categories or from coarse source categories to fine-grained target categories. The standard fine-tuning strategy [16] and its variants [238, 400] in supervised transfer are inapplicable here since they require a significant amount of labeled target data, which is simply not available. For novel categories, effective unsupervised transfer of CNN representations remains an open challenge [394].

Image clustering. Different from previous work [78, 109, 349], we use CNN features and evaluate our model on both standard image clustering datasets and large-scale image classification datasets. The latter datasets still remain challenging even for (supervised) image classification tasks. In related work, an ensemble of image prototype sets is sampled from the available data to represent a rich set of visual categories, and images are projected onto these prototypes as new feature representations [77]. Unlike [77], which takes advantage that the test data is used as unlabeled data for training (*i.e.*, transductive learning), we follow strict train/test splits for each dataset to ensure the generalization of our approach. Direct clustering in a pre-trained, fixed supervised CNN feature space [77] is simple but sub-optimal due to domain shift. Performing clustering through unsupervised deep feature learning provides an attractive option [415]. However, the performance of the unsupervised deep models is still not on par with that of their supervised counterparts. On the contrary, we leverage both supervised CNN feature learning and unsupervised transfer learning.

10.3 Factorized Convolutional Networks

Let us consider a CNN architecture pre-trained on a source domain with abundant data, for example the vanilla VGGNet [347] pre-trained on ImageNet (ILSVRC) $1{,}000$ categories [321]. The CNN is composed of a feature representation module $\mathcal{F}$ (*e.g.*, the 13 convolutional layers $C1$-$C13$ and two fully-connected layers $fc6$, $fc7$ for VGGNet) and a classifier module $\mathcal{C}$ (*e.g.*, the last fully-connected layer $fc8$ with $1{,}000$ units and the $1{,}000$-way softmax for ImageNet classification) [400].

We now transfer this CNN for representation learning on unlabeled target images in tasks such as image clustering. The transfer is accomplished through our unsupervised fine-tuning and the target CNN is instantiated and initialized in the following way, as shown in Figure 10.2: (1) the representation module $\mathcal{F}_T$ is copied from $\mathcal{F}_S$ of the source CNN with the parameters $\Theta^{\mathcal{F}}_T = \Theta^{\mathcal{F}}_S$, and (2) the classifier module $\mathcal{C}$ is removed and a new "adaptation module" $\mathcal{A}$ is introduced that consists of a group-sparse non-negative matrix factorization (GSNMF) on top of $fc7$ activations.

Note that the unsupervised fine-tuning is different from conventional supervised fine-tuning; in the latter case, a new classifier module $\mathcal{C}_T$ (*e.g.*, a new $fc8$ and softmax) is introduced with the parameters $\Theta^{\mathcal{C}}_T$ randomly initialized.

As a complex non-linear function of all input pixels, the $fc7$ representation may capture mid-level object parts as well as their high-level configurations [283]. Our GSNMF module then reduces feature dimension and enlarges sparsity among different groups simultaneously, thus identifying a rich set of informative latent variables useful for unsupervised adaptation. The GSNMF module is trained while a portion of the parameters $\Theta^{\mathcal{F}}_T$ are optionally fine-tuned (depending on the amount of available data) by continuing the back-propagation.

10.3.1 NMF/GSNMF Module

We consider an M dimensional random vector $\boldsymbol{x}$ with non-negative elements, *e.g.*, the CNN $fc7$ activations in our case. Its N observations are denoted as $\boldsymbol{x}_i, i = 1, 2, \ldots, N$. N is the batch size in our stochastic optimization. Let the data matrix be $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{M \times N}_{\geq 0}$. NMF seeks a non-negative basis matrix $\boldsymbol{W} \in \mathbb{R}^{M \times L}_{\geq 0}$ and a coefficient matrix $\boldsymbol{H} \in$

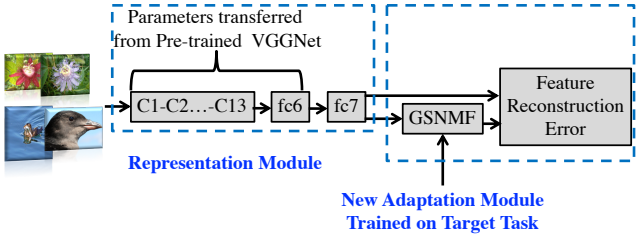


Figure 10.2: Illustration of unsupervised fine-tuning and factorized convolutional networks. A network (*e.g.*, VGGNet) is trained on the source task (*e.g.*, ImageNet classification) with a large amount of labeled images. The pre-trained parameters of its feature representation module ($C1$-$C13$ and $fc6$, $fc7$) are then transferred to the target task with unlabeled data (*e.g.*, image clustering). In such unsupervised scenario, we introduce a new "adaptation module" that consists of a group-sparse non-negative matrix factorization (GSNMF) on top of $fc7$ activations to compensate for the different image dataset statistics (*e.g.*, type of objects, typical viewpoints) between the source and target data. We then train the GSNMF module while fine-tuning the representation module based on the GSNMF reconstruction (generative) loss using the unlabeled target image data.

$\mathbb{R}_{\geq 0}^{L \times N}$ such that

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}. \tag{10.1}$$

Usually $L \ll min(M,N)$.

While classic NMF is able to identify informative latent variables, it is commonly known that large deep neural networks typically are comprised of many redundant and highly correlated units [167]. Hence, to better select a group of correlated CNN activations, we enforce additional group-sparsity constraints on NMF. The joint ℓ_1 and ℓ_2 norm penalty, *i.e.*, elastic net regularization, has been widely used as a group-sparse regularization technique [443]. The ℓ_1 part generates a sparse model while the ℓ_2 part encourages a smoothing, grouping effect [251]. Such group-sparsity property is beneficial when transferring a pre-trained CNN to a novel task, since it allows to select the correlated features suitable to the target data while discarding those uncorrelated ones. We thus impose a weighted mixture of ℓ_1 and squared ℓ_2 penalties on the coefficient matrix $\boldsymbol{H}$ to achieve the desired group-sparse representations. The resulting GSNMF objective function is defined as

$$\begin{aligned} f(\boldsymbol{W},\boldsymbol{H}) = \frac{1}{2} \parallel \boldsymbol{X} - \boldsymbol{W}\boldsymbol{H} \parallel_F^2 + \frac{\lambda_1}{2} \parallel \boldsymbol{H} \parallel_2^2 + \lambda_2 \parallel \boldsymbol{H} \parallel_1, \\ s.t. \boldsymbol{W},\boldsymbol{H} \geq 0. \end{aligned} \tag{10.2}$$

Here λ_1 and λ_2 are the hyper-parameters that control the importance of the ℓ_1 and ℓ_2 regularization terms.

10.3.2 Optimization

We use the alternating minimization procedure and multiplicative update rule to optimize Eqn. (10.2) following [228]. Since we impose group-sparsity on the coefficient matrix $\boldsymbol{H}$, the update rule of $\boldsymbol{W}$ remains the same as that in the standard NMF formulation [228]. We use gradient descent to optimize $\boldsymbol{H}$, and the first-order update rule of $\boldsymbol{H}$ should be generally

in the form of

$$\boldsymbol{H} \leftarrow \boldsymbol{H} - \eta * \frac{\partial f(\boldsymbol{H})}{\partial \boldsymbol{H}}, \tag{10.3}$$

where $*$ denotes the element-wise multiplication and the matrix η is the step size. We take the derivative of $f(\boldsymbol{H})$ in Eqn. (10.2) with respect to $\boldsymbol{H}$, leading to

$$\frac{\partial f}{\partial \boldsymbol{H}} = -\boldsymbol{W}^T\boldsymbol{X} + \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H} + \boldsymbol{\lambda_1}\boldsymbol{H} + \boldsymbol{\lambda_2}\boldsymbol{I}, \tag{10.4}$$

where $\boldsymbol{I}$ is an all-ones matrix of the same size as $\boldsymbol{H}$. Since the ℓ_1 norm is not differentiable at 0, Eqn. (10.4) is the subgradient at 0. Following a similar deriving procedure as in [228], we let the adaptive step size η to be

$$\eta = \frac{\boldsymbol{H}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H} + \boldsymbol{\lambda_1}\boldsymbol{H} + \boldsymbol{\lambda_2}\boldsymbol{I}}, \tag{10.5}$$

where the division is element-wise division, and we then have the following update rule

$$\begin{cases} \boldsymbol{W} \leftarrow \boldsymbol{W} * \frac{\boldsymbol{X}\boldsymbol{H}^T}{\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T}, \\ \boldsymbol{H} \leftarrow \boldsymbol{H} * \frac{\boldsymbol{W}^T\boldsymbol{X}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H} + \boldsymbol{\lambda_1}\boldsymbol{H} + \boldsymbol{\lambda_2}\boldsymbol{I}}. \end{cases} \tag{10.6}$$

Here the coefficient $\boldsymbol{H}$ is the new feature representation. Eqn. (10.6) is a straightforward modification to the multiplicative update rule in the standard NMF optimization [228]. Following a similar proof to that of [228] which uses an auxiliary function analogous to that used for proving convergence of the Expectation Maximization algorithm [82], we can show that the process converges. Since the update rules are multiplicative, when $\boldsymbol{W}$ and $\boldsymbol{H}$ are initialized as non-negative, they will remain non-negative during the optimization.

10.3.3 Unsupervised Fine-Tuning of the Network

As shown in Figure 10.2, we use the feature reconstruction (generative) loss in Eqn. (10.2) for our unsupervised fine-tuning, in contrast to the cross-entropy loss in conventional supervised fine-tuning. In the off-the-shelf (OTS) scenario, we only train the GSNMF module while freezing the pre-trained representation module. In the fine-tuning (FT) scenario, we train the GSNMF module while fine-tuning the representation module. Following the standard NMF practice, in our implementation, we introduce additional ℓ_2 normalization layers to $\boldsymbol{x}$ and the basis matrix $\boldsymbol{W}$ before the factorization layer. Regularizing the feature vector norm has been a staple of unsupervised learning approaches to prevent degenerate solutions and collapsed networks [308].

During each iteration, after forward propagation, we obtain the $fc7$ activations (*i.e.*, $\boldsymbol{x}$) on the mini-batch. The mini-batch size is $N = 256$. We learn $\boldsymbol{W}$ and $\boldsymbol{H}$ using the update rule in Eqn. (10.6) . We then fix $\boldsymbol{W}$ and $\boldsymbol{H}$, and the loss in Eqn. (10.2) reduces to the standard Euclidean loss $\| \boldsymbol{X} - \boldsymbol{W}\boldsymbol{H} \|_F^2$. We back-propagate the Euclidean error to update the parameters in the CNN representation module. This alternating fine-tuning strategy using generative connections could also be seen broadly relevant to the wake-sleep algorithm [160]. In our evaluation with limited target data, we froze the remaining layers underneath $fc7$ and did not fine-tune them due to over-fitting concerns. With more training data available, additional layers could be further fine-tuned.

Algorithm complexity. The time complexity of our approach is polynomial time $O(NLT)$, where N is the number of samples, L is the feature dimension, and T is the iteration number. In our experiments, a forward-backward pass took less than 0.5 second on a single Titan GPU.

10.4 Experimental Evaluation

In this section, we evaluate the representation transferability of our factorized convolutional networks (FCNs) on both standard image clustering datasets and multiple much more challenging benchmarks for image clustering, in which no labeled data is provided. We first introduce the datasets and the implementation details, and then present quantitative results by comparing with several state-of-the-art methods and validating across tasks the generality of FCN. In absolute terms, we achieve the best performance on all these benchmarks. We also show that FCN can be used as unsupervised initialization to further improve the performance of classification tasks. Our approach is general as it can be applied to different CNN architectures. Here we focus on VGGNet [347] and evaluate variants of our model:

NMF-FCN. All layers of VGGNet are frozen, and we feed the $fc7$ activations to an NMF module. The coefficient matrix $\boldsymbol{H}$ is used as the new feature representation.

GSNMF-FCN-OTS. All layers of VGGNet are frozen, and we feed the $fc7$ activations to a GSNMF module, in which the group-sparsity constraints are imposed on the standard NMF layer.

GSNMF-FCN-FT. All layers except the $fc7$ layer are frozen, and we combine VGGNet with a GSNMF module and fine-tune $fc7$ as well. The coefficient matrix $\boldsymbol{H}$ is used as the new feature representation.

10.4.1 Datasets

Our model is evaluated on diverse datasets including standard image clustering datasets and large-scale image classification datasets (used for the image clustering tasks):

MNIST [226]. MNIST consists of 28×28 gray scale images of handwritten digits ranging from 0 to 9. The dataset contains 50,000 training samples, 10,000 validation samples, and 10,000 test samples.

COIL-20 [274]. COIL-20 consists of 1440 32×32 gray scale images of 20 objects. The images of each object were taken 5 degree apart.

As there is no standard large-scale image clustering dataset, we evaluate our model on large-scale image classification datasets whose labels are not used during training: scene classification on **MIT-**67 [375], fine-grained recognition on **Caltech-UCSD Birds (CUB)** 200-2011 [389] and **Oxford** 102 **Flowers** [279]. We follow the standard experimental setup (*e.g.*, the train/test splits) for these datasets as detailed in Chapter 3.

These are very challenging tasks because of the following reasons. (1) There are strong domain shifts between the source and target datasets. Compared to the object-centric ILSVRC dataset where the CNN features are pre-trained, the target MIT-67 dataset is more scene-centric and consists of similar objects presented in different indoor scenes [375], and the target Birds-200 and Flowers-102 datasets involve very subtle differences between examples of a visual category [16]. Importantly, the transferability of a CNN decreases when the target task is far from the CNN source task [16]. (2) The datasets used for evaluation are standard classification benchmarks, and they are still very challenging even for supervised image classification. However, we tackle a more difficult scenario here by testing the representations for unsupervised image clustering, without having access to the label information on these datasets. We will show that with limited amount of unlabeled training data from distinct target tasks, our FCN model is capable of discovering informative and discriminative latent variables from CNN representations.

10.4.2 Baseline Models

In order to evaluate the performance of our FCN model, we compared it against not only the state-of-the-art algorithm, but also other linear and nonlinear dimension reduction algorithms that could be useful in learning effective feature representations. These baselines include:

CNN. All layers of VGGNet are frozen, and the $fc7$ activations are used as the feature representation with dimension $4{,}096$.

PCA-CNN. We perform PCA over the CNN representation and use the coefficient as the new feature representation. The number of principal components is set as $1{,}024$.

LLE-CNN [92]. Locally linear embedding uses an eigenvector based optimization technique to find the low-dimensional embedding of points, such that each point is still described with the same linear combination of its neighbors. The number of nearest neighbors is set as 12 and the feature dimension is set as $1{,}024$.

Autoencoder-CNN (AE-CNN). After careful preliminary experiments, we choose linear activations as the transition functions of the encoder and decoder. The autoencoder is trained in 200 epochs with a batch size of 128. To avoid over-fitting, we use 10% of training data as validation data.

Non-Negative AutoEncoder-CNN (NNAE-CNN). Due to limited training data, we use linear activations as the transition functions as above. During each iteration, we force the weights of the encoder and decoder to be non-negative.

EP-CNN [76, 77]. Ensemble projection samples from the available training data as an ensemble of image prototype sets and learns discriminative functions over these prototype sets. We follow the same parameter setting in [76, 77], and the feature dimension is $3{,}000$.

10.4.3 Implementation Details

Our FCN model includes two modules and is implemented in Keras [69]. For the CNN layers, we use the VGGNet pre-trained on ILSVRC where all the layers except $fc7$ are frozen to those learned on ILSVRC without fine-tuning [347]. In our preliminary experiment, we fine-tuned $fc6$ as well. Compared with only fine-tuning $fc7$, the performance dropped due to limited target data in our case. This is consistent with the observation in standard supervised fine-tuning. With more training data, fine-tuning more layers should further improve the performance. For each image, we resize the image to 224×224, and extract a $4{,}096$-dim feature vector from the entire image.

For the GSNMF module, to speed up the convergence rate of NMF, we use the nonnegative double singular value decomposition (NNDSVD) [45]. NNDSVD is a method based on two SVD processes: one approximates the initial data matrix, and the other approximates the positive components of the resulting partial SVD factors. We use the unlabeled training data on the target task to learn the bases and coefficients. L is set as $1{,}024$. The test images are then fed forward to the learned FCN model, producing a final $1{,}024$-dim feature representation.

Note that our main purpose is to validate whether the proposed approach is able to boost the transferability of CNN features for image clustering and is not to propose a better clustering approach. Hence, we use two standard clustering methods, which are spectral clustering (SC) [277] and k-means. Choosing the k number of clusters is typically difficult for clustering algorithms without any prior knowledge of the data. We then chose k as the number of classes for each target dataset. In our preliminary experiments, we found that

ours consistently outperformed baselines with different values of k, due to our improved feature representations. For a fair comparison, we perform ℓ_2 normalization on the feature representations for both our models and baselines. To reduce the influence of randomness introduced by different initializations of k-means, the k-means grouping stages in SC and k-means are repeated 10 times. The result with the minimum distortion is selected. Euclidean distance is used for both methods.

10.4.4 Methodology

Hyper-parameter settings. For the regularization parameters λ_1 and λ_2 in GSNMF, in a preliminary experiment, we tested image clustering on the Scene-15 dataset [106], which is a relatively small dataset. After searching λ_1 and λ_2 on a 2D grid $10^{[-4:1:1]} \times 10^{[-4:1:1]}$, we observed that the best performance was achieved when $\lambda_1 = 0.02$ and $\lambda_2 = 0.05$. In all our experiments, we then simply set λ_1 as 0.02 and λ_2 as 0.05.

Choosing the optimal representation dimension L remains challenging in dimensionality reduction. Similarly, in our preliminary experiment, we tested image clustering on Scene-15 and MIT-67, and found $L = 1{,}024$ usually achieved the best performance. In all our experiments, we then simply set $L = 1{,}024$. Even better performance could be obtained by further tuning these hyper-parameters. We also conduct hyper-parameter sensitivity analysis to test how λ_1, λ_2 and L affect the clustering accuracy.

Evaluation metrics. Consistent with the previous work, accuracy [416] and normalized mutual information (NMI) [51] are used as the evaluation criterion. We assume that the clustering algorithm is tested on N samples. For a sample $\boldsymbol{x}_i$, the cluster label is denoted as r_i, and its ground truth label is t_i. Accuracy is defined as

$$\text{accuracy} = \frac{\sum_{i=1}^{N} \delta(t_i, \text{map}(r_i))}{N}, \tag{10.7}$$

where $\delta(x, y)$ equals to 1 if x is equal to y, and 0 otherwise. The function $\text{map}(x)$ is the best permutation mapping function, which maps a cluster to its corresponding predicted label. Hence, a higher accuracy indicates that more samples are predicted correctly.

Now let C denote the cluster centers of the ground truth, and $C^{'}$ denote the cluster centers predicted by the clustering algorithm. NMI is then defined as

$$\text{NMI}(C, C^{'}) = \frac{\text{MI}(C, C^{'})}{\max(H(C), H(C^{'}))}, \tag{10.8}$$

where $H(C)$ and $H(C^{'})$ are the entropies of C and $C^{'}$, respectively. $\text{MI}(C, C^{'})$ is the mutual information of C and $C^{'}$. NMI measures the dependency of two distributions. A higher NMI means that two distributions are more similar.

10.5 Results and Discussion

In our FCN model, we aim to learn better high-level representations by introducing the GSNMF module on top of the original CNN representation. Furthermore, by fine-tuning the CNN representation module (the last fully-connected layer in our case), the additional degree of freedom allows the FCN model to represent the target data more effectively. To validate this, we evaluate our model on standard image clustering datasets and large-scale benchmarks.

	Method	k-means				spectral clustering			
		MNIST		COIL-20		MNIST		COIL-20	
		Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI
Baselines	CNN	46.7	38.4	74.0	89.2	51.1	41.3	84.3	92.0
	PCA-CNN	55.0	47.2	76.9	89.9	48.4	39.6	84.4	91.9
	LLE-CNN [92]	29.0	23.0	34.2	34.5	28.9	17.4	22.6	24.7
	AE-CNN	45.9	37.3	81.7	90.4	54.9	44.6	81.7	92.1
	NNAE-CNN	46.0	37.4	77.1	90.0	49.2	40.6	85.7	92.3
	EP-CNN [76,77]	60.0	58.0	80.5	92.2	53.2	47.6	86.7	93.8
Ours	NMF-FCN	61.7	57.8	82.3	93.0	56.4	47.6	88.6	93.9
	GSNMF-FCN-OTS	62.2	58.0	**84.3**	**94.4**	57.6	47.9	**89.2**	**94.0**
	GSNMF-FCN-FT	**63.3**	**58.6**	79.6	89.4	**58.4**	**48.0**	86.5	92.7

Table 10.1: Accuracy (%) and normalized mutual information (NMI) (%) of image clustering on two standard benchmark datasets. k-means and spectral clustering are used on top of the feature representations of our FCN model and baseline models. The best results are in bold.

	Method	k-means						spectral clustering					
		MIT-67		Birds-200		Flowers-102		MIT-67		Birds-200		Flowers-102	
		Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI
Baselines	CNN	45.0	63.2	32.5	61.6	45.7	63.7	37.3	57.6	26.5	57.0	44.7	63.6
	PCA-CNN	45.9	63.8	32.1	62.0	46.3	63.9	37.3	57.5	29.1	58.7	46.0	63.0
	LLE-CNN [92]	17.4	32.8	21.3	49.4	26.6	45.9	20.7	42.3	17.3	33.4	18.0	35.3
	AE-CNN	46.8	64.3	32.6	62.2	46.8	64.5	35.4	57.1	28.0	58.0	43.8	62.1
	NNAE-CNN	43.0	63.0	32.4	61.8	45.6	64.5	35.3	58.8	29.0	58.7	**46.6**	63.5
	EP-CNN [76,77]	47.2	64.6	31.0	61.1	43.9	60.0	43.7	61.9	29.0	59.6	43.0	61.3
Ours	NMF-FCN	46.8	64.0	34.1	62.2	50.2	65.7	42.8	61.4	28.7	58.9	45.3	**64.7**
	GSNMF-FCN-OTS	47.7	64.5	34.4	62.1	**51.0**	**65.9**	43.2	62.3	31.0	59.8	45.7	63.9
	GSNMF-FCN-FT	**48.2**	**64.9**	**35.1**	**62.5**	50.1	64.9	**44.2**	**62.5**	**32.2**	**60.0**	44.7	62.0

Table 10.2: Accuracy (%) and normalized mutual information (NMI) (%) of scene and fine-grained image clustering on three large-scale benchmark datasets. k-means and spectral clustering are used on top of the feature representations of our FCN model and baseline models. The best results are in bold.

Evaluation on standard image clustering datasets. Table 10.1 summarizes the comparison between our model and the baseline models. Our FCN model outperforms the original CNN representation by a large margin on these standard image clustering datasets. For example, in terms of accuracy, our FCN model outperforms the original CNN representation by 16.6% on MNIST and 10.3% on COIL-20.

Evaluation on large-scale benchmark datasets. Table 10.2 summarizes the k-means and spectral clustering (SC) performance of our FCN representation and related baseline features on the scene and fine-grained recognition datasets. We can see that our FCN representation outperforms the original CNN feature and other representations by a considerable margin using the same clustering method. For instance, in terms of clustering accuracy, FCN outperforms CNN by 3.2% on MIT-67, 2.6% on Birds-200 (where chance is 0.5%), and

	Method		
Dimension L	PCA-CNN	AE-CNN	GSNMF-FCN (Ours)
256	40.3	42.7	**43.2**
512	42.8	43.9	**45.1**
1024	45.9	46.8	**48.2**
2048	43.6	45.2	**46.0**

Table 10.3: Hyper-parameter sensitivity analysis on MIT-67: Accuracy (%) comparison between our FCN and PCA-CNN, AE-CNN as functions of the feature dimension L.

5.3% on Flowers-102 (where chance is 1%) when k-means is used.

Given that there are only 10 images available for each class on Flowers-102 to fine-tune our network, GSNMF-FCN-FT performs slightly worse than GSNMF-FCN-OTS, while GSNMF-FCN-OTS significantly outperforms CNN. Consistent with the standard supervised fine-tuning, for target tasks with medium sized data, GSNMF-FCN-FT consistently outperforms GSNMF-FCN-OTS (*e.g.*, by 1% on Birds-200 with 200 classes and $5{,}994$ images). With more data, GSNMF-FCN-FT will further improve the performance.

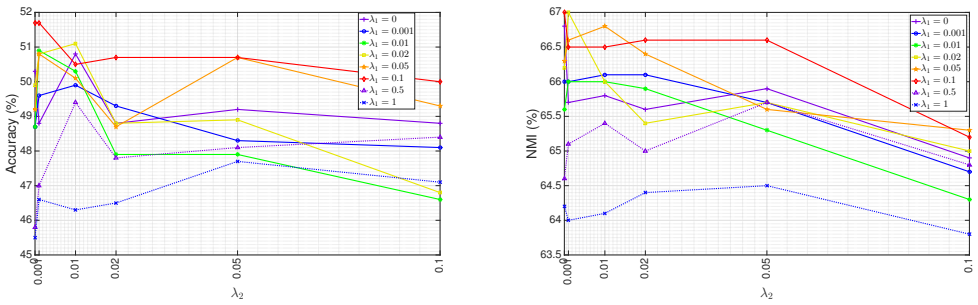
Moreover, EP-CNN reported improved performance over CNN in transductive learning, where the EP representation (ensemble of classifiers) was learned using both the training and test datasets [76]; however, in our case of learning representation on the training dataset and conducting clustering on the test dataset, EP-CNN shows inferior performance to CNN. This means that having access to the distribution of the test data is advantageous for EP-CNN.

The superior performance of our GSNMF-FCN reveals that it learns a more generic and transferable representation to capture the subtlety of differences across different categories and tasks. In particular, these results show that our approach, pre-trained on ILSVRC, is effective on a broad range of target domains, ranging from low source-target distance (*e.g.*, MIT-67), to medium distance (*e.g.*, Birds-200, Flowers-102), and to large distance (*e.g.*, MNIST) [16].

Hyper-parameter sensitivity analysis. We now examine the influence of the hyper-parameters of our model on its clustering performance. They are the regularization coefficients λ_1, λ_2 and the feature dimension L. We first evaluate L on MIT-67, and Table 10.3 shows that FCN consistently outperforms PCA-CNN and AE-CNN in different settings of L. We then evaluate λ_1, λ_2 on Flowers-102. Each time we change the value of one hyper-parameter with the others fixed to the values described in the experimental settings.

Figure 10.3 summarizes the hyper-parameter sensitivity analysis. The performance of our model increases with λ_2 at first and then stabilizes quickly given a certain λ_1. It validates that our model benefits from imposing the regularization terms. After λ_2 increases above some threshold (*e.g.*, 0.05), the accuracy and NMI become stable; as λ_2 increases further, the performance drops accordingly, implying that a larger ℓ_2 regularization coefficient will hurt the performance. Similar trend is observed for the ℓ_1 regularization (*e.g.*, for a fixed $\lambda_2 = 0.05$, our model achieves best when λ_1 lies in the range from 0.02 to 0.05).

Evaluation of group-sparsity formulations. Our GSNMF-FCN uses the joint ℓ_1 and ℓ_2 norm penalty to impose group-sparsity. The mixed $\ell_{2,1}$ norm penalty is an alternative [197]. In our preliminary experiment, we compared the two formulations, and found that intro-



(a) Accuracy (b) NMI

Figure 10.3: Hyper-parameter sensitivity analysis on Flowers-102: Accuracy (%) and NMI (%) of our FCN as functions of its regularization coefficients λ_1 and λ_2.

Method	MNITS	Birds-200
mixed $\ell_{2,1}$	56.5	29.1
joint ℓ_1 and ℓ_2 (Ours)	**57.6**	**31.0**

Table 10.4: Performance comparison of clustering accuracy (%) between different group-sparsity formulations. The joint ℓ_1 and ℓ_2 norm penalty outperforms the mixed $\ell_{2,1}$ norm.

Method	MIT-67	Birds-200	Flowers-102
CNN	70.78	68.51	82.44
GSNMF-FCN (Ours)	**72.75**	**70.89**	**84.70**

Table 10.5: Performance comparison of classification accuracy (%) between GSNMF-FCN and CNN. Learning SVM classifiers on top of the unsupervised GSNMF-FCN embedding outperforms training SVMs in the original CNN space.

ducing group sparsity helped and the joint ℓ_1 and ℓ_2 norm worked better, as shown in Table 10.4.

Unsupervised fine-tuning as initialization for image classification. Our FCN can be used to improve the classification performance as well. We evaluate this point by first learning GSNMF embedding on the target training data without using the labels and then training SVM classifiers on top of the learned embedding using the training labels. Table 10.5 shows that our approach outperforms SVM directly trained with the original CNN feature representation.

Part IV

Combining Generative Learning with Meta-Learning

What I cannot create, I do not understand.

Richard Phillips Feynman

Chapter 11

Few-Shot Learning from Imaginary Data

11.1 Motivation

Different from the previous chapters, in this part we explore from another perspective and focus on what is missing in the current meta-learning methods. Meta-learning methods train a *learner*, which is a parametrized function that maps labeled training sets to classifiers. Meta-learners are trained by *sampling* small training sets and test sets from a large universe of labeled examples, feeding the sampled training set to the learner to get a classifier, and then computing the loss of the classifier on the sampled test set. These methods directly frame few-shot learning as an optimization problem.

However, generic meta-learning methods treat images as black boxes, ignoring the structure of the visual world. In particular, many modes of variation (for example camera pose, translation, lighting changes, and even articulation) are shared across categories. As humans, our knowledge of these shared modes of variation may allow us to visualize what a novel object might look like in other poses or surroundings (Figure 11.1). If machine vision systems could do such "hallucination" or "imagination", then the hallucinated examples could be used as additional training data to build better classifiers.

Unfortunately, building models that can perform such hallucination is hard, except for simple domains like handwritten characters [258]. For general images, while considerable progress has been made recently in producing realistic samples, most current generative modeling approaches suffer from the problem of mode collapse [327]: they are only able to capture some modes of the data. This may be insufficient for few-shot learning since one needs to capture many modes of variation to be able to build good classifiers. Furthermore, the modes that are useful for classification may be different from those that are found by training an image generator. Prior work has tried to avoid this limitation by explicitly using pose annotations to generate samples in novel poses [86], or by using carefully designed, but brittle, heuristics to ensure diversity [144].

Our key insight is that the criterion that we should aim for when hallucinating additional examples is neither diversity nor realism. Instead, the aim should be to hallucinate examples that are *useful* for learning classifiers. Therefore, we propose a new method for few-shot learning that directly learns to hallucinate examples that are useful for classification by the

Figure 11.1: Given a single image of a novel visual concept, such as a blue heron, a person can visualize what the heron would look like in other poses and different surroundings. If computer recognition systems could do such hallucination, they might be able to learn novel visual concepts from less data.

end-to-end optimization of a classification objective that includes data hallucination in the model.

We achieve this goal by unifying meta-learning with hallucination. Our approach trains not just the meta-learner, but also a *hallucinator*: a model that maps real examples to hallucinated examples. The few-shot training set is first fed to the hallucinator; it produces an expanded training set, which is then used by the learner. Compared to plain meta-learning, our approach uses the rich structure of shared modes of variation in the visual world. We show empirically that such hallucination adds a significant performance boost to two different meta-learning methods [351, 388], providing up to a 6 point improvement when only a single training example is available. Our method is also agnostic to the choice of the meta-learning method, and provides significant gains irrespective of this choice. It is precisely the ability to leverage standard meta-learning approaches *without any modifications* that makes our model simple, general, and very easy to reproduce. Compared to prior work on hallucinating examples, we use no extra annotation and significantly outperform hallucination based on brittle heuristics [144]. We also present a novel meta-learning method and discover and fix flaws in previously proposed benchmarks.

11.2 Generative Models for Few-Shot Learning

One class of few-shot learning approaches builds generative models that can share priors across categories [104, 121, 326]. Often, these generative models have to be hand-designed for the domain, such as strokes [218, 220] or parts [411] for handwritten characters. For more unconstrained domains, while there has been significant recent progress [132, 304, 316], modern generative models still cannot capture the entirety of the distribution [327].

Different classes might not share parts or strokes, but may still share modes of variation, since these often correspond to camera pose, articulation, *etc*.. If one has a probability density on transformations, then one can generate additional examples for a novel class by applying sampled transformations to the provided examples [86, 144, 258]. Learning such

a density is easier for handwritten characters that only undergo 2D transformations [258], but much harder for generic image categories. This problem is tackled by leveraging an additional dataset of images labeled with pose and attributes [86]; this allows to learn how images transform when the pose or the attributes are altered. To avoid annotation, transformations are transferred from a pair of examples from a known category to a "seed" example of a novel class [144]. However, learning to do this transfer requires a carefully designed pipeline with many heuristic steps. Our approach follows this line of work, but learns to do such transformations in an end-to-end manner, avoiding both brittle heuristics and expensive annotations. We present a unified view of meta-learning and show that our hallucination strategy can be adopted in any of the existing meta-learning methods.

11.3 Meta-Learning

Let $\mathcal{X}$ be the space of inputs (*e.g.*, images) and $\mathcal{Y}$ be a discrete label space. Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$. Supervised machine learning typically aims to capture the conditional distribution $p(y|\boldsymbol{x})$ by applying a *learning algorithm* to a parameterized model and a training set $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i) \sim \mathcal{D}\}_{i=1}^{N}$. At inference time, the model is evaluated on test inputs $\boldsymbol{x}$ to estimate $p(y|\boldsymbol{x})$. The composition of the inference and learning algorithms can be written as a function h (a *classification algorithm*) that takes as input the training set and a test input $\boldsymbol{x}$, and outputs an estimated probability distribution $\hat{\boldsymbol{p}}$ over the labels:

$$\hat{\boldsymbol{p}}(\boldsymbol{x}) = h(\boldsymbol{x}, S_{\text{train}}). \tag{11.1}$$

In few-shot learning, we want functions h that have high classification accuracy even when S_{train} is small. Meta-learning is an umbrella term that covers a number of recently proposed empirical risk minimization approaches to this problem [114, 310, 351, 388, 395]. Concretely, they consider *parametrized* classification algorithms $h(\cdot, \cdot; \boldsymbol{w})$ and attempt to estimate a "good" parameter vector $\boldsymbol{w}$, namely one that corresponds to a classification algorithm that can learn well from small datasets. Thus, estimating this parameter vector can be construed as *meta-learning* [368].

Meta-learning algorithms have two stages. The first stage is *meta-training* in which the parameter vector $\boldsymbol{w}$ of the classification algorithm is estimated. During meta-training, the meta-learner has access to a large labeled dataset S_{meta} that typically contains thousands of images for a large number of classes C. In each iteration of meta-training, the meta-learner samples a *classification problem* out of S_{meta}. That is, the meta-learner first samples a subset of m classes from C. Then it samples a small "training" set S_{train} and a small "test" set S_{test}. It then uses its current weight vector $\boldsymbol{w}$ to compute conditional probabilities $h(\boldsymbol{x}, S_{\text{train}}; \boldsymbol{w})$ for every point $(\boldsymbol{x}, y)$ in the test set S_{test}. Note that in this process h may perform internal computations that amount to "training" on S_{train}. Based on these predictions, h incurs a loss $L(h(\boldsymbol{x}, S_{\text{train}}; \boldsymbol{w}), y)$ for each point in the current S_{test}. The meta-learner then backpropagates the gradient of the total loss $\sum_{(\boldsymbol{x},y)\in S_{\text{test}}} L(h(\boldsymbol{x}, S_{\text{train}}; \boldsymbol{w}), y)$. The number of classes in each iteration, m, and the maximum number of training examples per class, n, are hyperparameters.

The second stage is *meta-testing* in which the resulting classification algorithm is used to solve novel classification tasks: for each novel task, the labeled training set and unlabeled test examples are given to the classification algorithm and the algorithm outputs class probabilities.

Different meta-learning approaches differ in the form of h. The data hallucination method introduced in this paper is general and applies to any meta-learning algorithm of the form described above. Concretely, we will consider the following three meta-learning approaches:

Prototypical Networks. Snell *et al*. [351] propose an architecture for h that assigns class probabilities based on distances from class means $\boldsymbol{\mu}_k$ in a learned feature space:

$$h(\boldsymbol{x}, S_{\text{train}}; \boldsymbol{w}) = \hat{\boldsymbol{p}}(\boldsymbol{x}) \tag{11.2}$$

$$\hat{p}_k(\boldsymbol{x}) = \frac{e^{-d(\phi(\boldsymbol{x};\boldsymbol{w}_\phi),\boldsymbol{\mu}_k)}}{\sum_j e^{-d(\phi(\boldsymbol{x};\boldsymbol{w}_\phi),\boldsymbol{\mu}_j)}} \tag{11.3}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{(\boldsymbol{x}_i,y_i)\in S_{\text{train}}} \phi(\boldsymbol{x}_i;\boldsymbol{w}_\phi)\mathbf{I}[y_i = k]}{\sum_{(\boldsymbol{x}_i,y_i)\in S_{\text{train}}} \mathbf{I}[y_i = k]}. \tag{11.4}$$

Here $\hat{p}_k$ are the components of the probability vector $\hat{\boldsymbol{p}}$ and d is a distance metric (Euclidean distance in [351]). The only parameters to be learned here are the parameters of the feature extractor $\boldsymbol{w}_\phi$. The estimation of the class means $\boldsymbol{\mu}_k$ can be seen as a simple form of "learning" from S_{train} that takes place internal to h.

Matching Networks. Vinyals *et al*. [388] argue that when faced with a classification problem and an associated training set, one wants to focus on the features that are useful for *those particular class distinctions*. Therefore, after embedding all training and test points independently using a feature extractor, they propose to create a *contextual embedding* of the training and test examples using bi-directional long short-term memory networks (LSTMs) and attention LSTMs, respectively. These contextual embeddings can be seen as emphasizing features that are relevant for the particular classes in question. The final class probabilities are computed using a soft nearest-neighbor mechanism. More specifically,

$$h(\boldsymbol{x}, S_{\text{train}}; \boldsymbol{w}) = \hat{\boldsymbol{p}}(\boldsymbol{x}) \tag{11.5}$$

$$\hat{p}_k(\boldsymbol{x}) = \frac{\sum_{(\boldsymbol{x}_i,y_i)\in S_{\text{train}}} e^{-d(f(\boldsymbol{x}),g(\boldsymbol{x}_i))}\mathbf{I}[y_i = k]}{\sum_{(\boldsymbol{x}_i,y_i)\in S_{\text{train}}} e^{-d(f(\boldsymbol{x}),g(\boldsymbol{x}_i))}} \tag{11.6}$$

$$f(\boldsymbol{x}) = \text{AttLSTM}(\phi(\boldsymbol{x};\boldsymbol{w}_\phi), \{g(\boldsymbol{x}_i)\}_{i=1}^N; \boldsymbol{w}_f) \tag{11.7}$$

$$\{g(\boldsymbol{x}_i)\}_{i=1}^N = \text{BiLSTM}(\{\phi(\boldsymbol{x}_i;\boldsymbol{w}_\phi)\}_{i=1}^N; \boldsymbol{w}_g). \tag{11.8}$$

Here, again d is a distance metric. Vinyals *et al*.used the cosine distance. There are three sets of parameters to be learned: $\boldsymbol{w}_\phi, \boldsymbol{w}_g$, and $\boldsymbol{w}_f$.

Prototype Matching Networks. One issue with matching networks is that the attention LSTM might find it harder to "attend" to rare classes (they are swamped by examples of common classes), and therefore might introduce heavy bias against them. Prototypical networks do not have this problem since they collapse every class to a single class mean. We want to combine the benefits of the contextual embedding in matching networks with the resilience to class imbalance provided by prototypical networks.

To do so, we collapse every class to its class mean before creating the contextual embeddings of the test examples. Then, the final class probabilities are based on distances to the contextually embedded class means instead of individual examples:

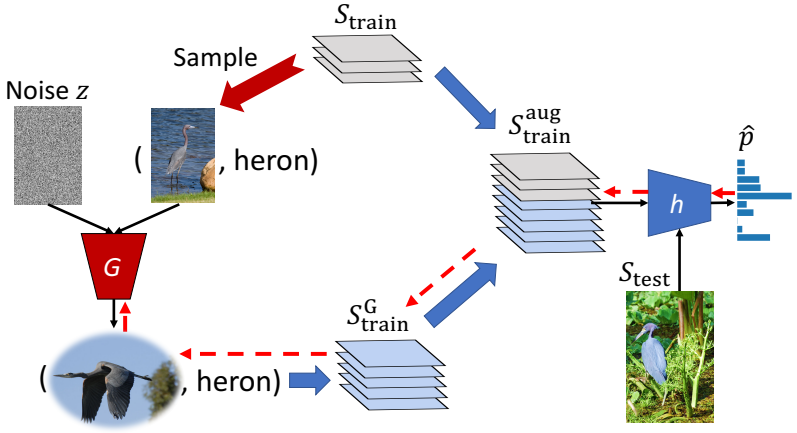


Figure 11.2: Meta-learning with hallucination. Given an initial training set S_{train}, we create an augmented training set $S^{\text{aug}}_{\text{train}}$ by adding a set of generated examples S^{G}_{train}. S^{G}_{train} is obtained by sampling real seed examples and noise vectors $\boldsymbol{z}$ and passing them to a parametric hallucinator G. The hallucinator is trained end-to-end along with the classification algorithm h. Dotted red arrows indicate the flow of gradients during back-propagation.

$$h(\boldsymbol{x}, S_{\text{train}}; \boldsymbol{w}) = \hat{\boldsymbol{p}}(\boldsymbol{x}) \tag{11.9}$$

$$\hat{p}_k(\boldsymbol{x}) = \frac{e^{-d(f(\boldsymbol{x}), \boldsymbol{\nu}_k)}}{\sum_j e^{-d(f(\boldsymbol{x}), \boldsymbol{\nu}_j)}} \tag{11.10}$$

$$f(\boldsymbol{x}) = \text{AttLSTM}(\phi(\boldsymbol{x}; \boldsymbol{w}_\phi), \{\boldsymbol{\nu}_k\}_{k=1}^{|\mathcal{Y}|}; \boldsymbol{w}_f) \tag{11.11}$$

$$\boldsymbol{\nu}_k = \frac{\sum_{(\boldsymbol{x}_i, y_i) \in S_{\text{train}}} g(\boldsymbol{x}_i) \mathbf{I}[y_i = k]}{\sum_{(\boldsymbol{x}_i, y_i) \in S_{\text{train}}} \mathbf{I}[y_i = k]} \tag{11.12}$$

$$\{g(\boldsymbol{x}_i)\}_{i=1}^{N} = \text{BiLSTM}(\{\phi(\boldsymbol{x}_i; \boldsymbol{w}_\phi)\}_{i=1}^{N}; \boldsymbol{w}_g). \tag{11.13}$$

The parameters to be learned are $\boldsymbol{w}_\phi, \boldsymbol{w}_g$, and $\boldsymbol{w}_f$. We call this novel modification to matching networks *prototype matching networks*.

11.4 Meta-Learning with Learned Hallucination

We now present our approach to few-shot learning by learning to hallucinate additional examples. Given an initial training set S_{train}, we want a way of *sampling* additional hallucinated examples. Following recent work on generative modeling [132, 200], we will model this stochastic process by way of a *deterministic* function operating on a *noise vector* as input. Intuitively, we want our hallucinator to take a single example of an object category and produce other examples in different poses or different surroundings. We therefore write this hallucinator as a function $G(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w}_G)$ that takes a seed example $\boldsymbol{x}$ and a noise vector $\boldsymbol{z}$ as input, and produces a hallucinated example as output. The parameters of this hallucinator are $\boldsymbol{w}_G$.

We first describe how this hallucinator is used in meta-testing, and then discuss how we train the hallucinator.

Hallucination During Meta-Testing. During meta-testing, we are given an initial training set S_{train}. We then hallucinate n_{gen} new examples using the hallucinator. Each hallucinated example is obtained by sampling a real example $(\boldsymbol{x}, y)$ from S_{train}, sampling a noise vector $\boldsymbol{z}$, and passing $\boldsymbol{x}$ and $\boldsymbol{z}$ to G to obtain a generated example $(\boldsymbol{x}', y)$ where $\boldsymbol{x}' = G(\boldsymbol{x}, z; \boldsymbol{w}_G)$. We take the set of generated examples S^G_{train} and add it to the set of real examples to produce an augmented training set $S^{\text{aug}}_{\text{train}} = S_{\text{train}} \cup S^G_{\text{train}}$. We can now simply use this augmented training set to produce conditional probability estimates using h. Note that the hallucinator parameters are kept fixed here; any learning that happens, happens within the classification algorithm h.

Meta-Training the Hallucinator. The goal of the hallucinator is to produce examples that help the classification algorithm learn a better classifier. This goal differs from realism: realistic examples might still fail to capture the many modes of variation of visual concepts, while unrealistic hallucinations can still lead to a good decision boundary [80]. We therefore propose to directly train the hallucinator to support the classification algorithm by using meta-learning.

As before, in each meta-training iteration, we sample m classes from the set of all classes, and *at most* n examples per class. Then, for each class, we use G to generate n_{gen} additional examples till there are exactly n_{aug} examples per class. Again, each hallucinated example is of the form $(\boldsymbol{x}', y)$, where $\boldsymbol{x}' = G(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w}_G)$, $(\boldsymbol{x}, y)$ is a sampled example from S_{train} and $\boldsymbol{z}$ is a sampled noise vector. These additional examples are added to the training set S_{train} to produce an augmented training set $S^{\text{aug}}_{\text{train}}$. Then this augmented training set is fed to the classification algorithm h, to produce the final loss $\sum_{(\boldsymbol{x},y)\in S_{\text{test}}} L(h(\boldsymbol{x}, S^{\text{aug}}_{\text{train}}), y)$, where $S^{\text{aug}}_{\text{train}} = S_{\text{train}} \cup S^G_{\text{train}}$ and $S^G_{\text{train}} = \{(G(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{w}_G), y_i)^{n_{\text{gen}}}_{i=1} : (\boldsymbol{x}_i, y_i) \in S_{\text{train}}\}$.

To train the hallucinator G, we require that the classification algorithm $h(\boldsymbol{x}, S^{\text{aug}}_{\text{train}}; \boldsymbol{w})$ is differentiable with respect to the elements in $S^{\text{aug}}_{\text{train}}$. This is true for many meta-learning algorithms. For example, in prototypical networks, h will pass every example in the training set through a feature extractor, compute the class means in this feature space, and use the distances between the test point and the class means to estimate class probabilities. If the feature extractor is differentiable, then the classification algorithm itself is differentiable with respect to the examples in the training set. This allows us to back-propagate the final loss and update not just the parameters of the classification algorithm h, but also the parameters $\boldsymbol{w}_G$ of the hallucinator. Figure 11.2 shows a schematic of the entire process.

Using meta-learning to train the hallucinator and the classification algorithm has two benefits. First, the hallucinator is directly trained to produce the kinds of hallucinations that are useful for class distinctions, removing the need to precisely tune realism or diversity, or the right modes of variation to hallucinate. Second, the classification algorithm is trained jointly with the hallucinator, which enables it to make allowances for any errors in the hallucination. Conversely, the hallucinator can spend its capacity on suppressing precisely those errors which throw the classification algorithm off.

Note that the training process is completely agnostic to the specific meta-learning algorithm used. We will show in our experiments that our hallucinator provides significant gains irrespective of the meta-learner.

11.5 Experimental Protocol

We use the benchmark proposed in [144]. This benchmark captures more realistic scenarios than others based on handwritten characters [218] or low-resolution images [388]. The benchmark is based on ImageNet images and subsets of ImageNet classes. First, in the *representation learning* phase, a convolutional neural network (ConvNet) based feature extractor is trained on one set of classes with thousands of examples per class; this set is called the "base" classes C_{base}. Then, in the *few-shot learning* phase, the recognition system encounters an additional set of "novel" classes C_{novel} with a small number of examples n per class. It also has access to the base class training set. The system has to now learn to recognize both the base and the novel classes. It is tested on a test set containing examples from both sets of classes, and it needs to output labels in the joint label space $C_{\text{base}} \cup C_{\text{novel}}$. The top-5 accuracy averaged over all classes, and also the top-5 accuracy averaged over just base-class examples, and the top-5 accuracy averaged over just novel-class examples are reported in [144].

Tradeoffs between Base and Novel Classes. We observed that in this kind of *joint* evaluation, different methods had very different performance tradeoffs between the novel and base class examples and yet achieved similar performance on average. This makes it hard to meaningfully compare the performance of different methods on just the novel or just the base classes. Further, we found that by changing hyperparameter values of some meta-learners it was possible to achieve substantially different tradeoff points without substantively changing average performance. This means that hyperparameters can be tweaked to make novel class performance look better at the expense of base class performance (or vice versa).

One way to concretize this tradeoff is by incorporating a prior over base and novel classes. Consider a classifier that gives a score $s_k(\boldsymbol{x})$ for every class k given an image $\boldsymbol{x}$. Typically, one would convert these into probabilities by applying a softmax function:

$$p_k(\boldsymbol{x}) = p(y = k|\boldsymbol{x}) = \frac{e^{s_k}}{\sum_j e^{s_j}}. \tag{11.14}$$

However, we may have some prior knowledge about the probability that an image belongs to the base classes C_{base} or the novel classes C_{novel}. Suppose that the prior probability that an image belongs to one of the novel classes is μ. Then, we can update Eqn. (11.14) as follows:

$$\begin{aligned} p_k(\boldsymbol{x}) &= p(y = k|\boldsymbol{x}) && (11.15)\\ &= p(y = k|y \in C_{\text{base}}, \boldsymbol{x})p(y \in C_{\text{base}}|\boldsymbol{x}) \\ &\quad + p(y = k|y \in C_{\text{novel}}, \boldsymbol{x})p(y \in C_{\text{novel}}|\boldsymbol{x}) && (11.16)\\ &= \frac{e^{s_k}\mathbf{I}[k \in C_{\text{base}}]}{\sum_j e^{s_j}\mathbf{I}[j \in C_{\text{base}}]}(1 - \mu) \\ &\quad + \frac{e^{s_k}\mathbf{I}[k \in C_{\text{novel}}]}{\sum_j e^{s_j}\mathbf{I}[j \in C_{\text{novel}}]}\mu. && (11.17) \end{aligned}$$

The prior probability μ might be known beforehand, but can also be cross-validated to correct for inherent biases in the scores s_k. However, note that in some practical settings, one may not have a held-out set of categories to cross-validate.Thus resilience to this prior is important.

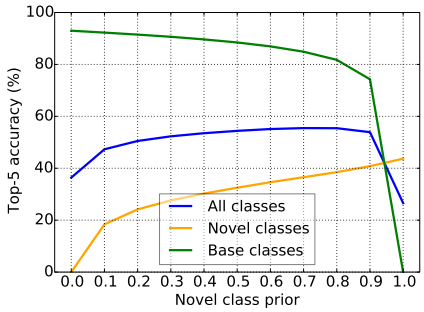


Figure 11.3: The variation of the overall, novel class, and base class accuracy for MN in the evaluation proposed in [144] as the novel class prior μ is varied.

Figure 11.3 shows the impact of this prior on matching networks in the evaluation proposed in [144]. Note that the overall accuracy remains fairly stable, even as novel class accuracy rises and base class accuracy falls. Such prior probabilities for calibration were proposed for the zero-shot learning setting [55].

A New Evaluation. The existence of this tunable tradeoff between base and novel classes makes it hard to make apples-to-apples comparisons of novel class performance if the model is tasked with making predictions in the joint label space. Instead, we use a new evaluation protocol that evaluates four sets of numbers:

1. The model is given test examples from the novel classes, and is only supposed to pick a label from the novel classes. That is, the label space is restricted to C_{novel} (note that doing so is equivalent to setting $\mu = 1$ for prototypical networks but not for matching networks and prototype matching networks because of the contextual embeddings). We report the top-5 accuracy on the novel classes in this setting.
2. Next, the model is given test examples from the base classes, and the label space is restricted to the base classes. We report the top-5 accuracy in this setting.
3. The model is given test examples from both the base and novel classes in equal proportion, and the model has to predict labels from the joint label space. We report the top-5 accuracy averaged across all examples. We present numbers both with and without a novel class prior μ; the former set cross-validates μ to achieve the highest average top-5 accuracy.

Note that, following [144], we use a disjoint set of classes for cross-validation and testing. This prevents hyperparameter choices for the hallucinator, meta-learner, and novel class prior from becoming overfit to the novel classes that are seen for the first time at test time.

11.6 Experimental Evaluation

11.6.1 Implementation Details

Unlike prior work on meta-learning which experiments with small images and few classes [114, 310, 351, 388], we use high resolution images and our benchmark involves hundreds

of classes. This leads to some implementation challenges. Each iteration of meta-learning at the very least has to compute features for the training set S_{train} and the test set S_{test}. If there are 100 classes with 10 examples each, then this amounts to 1000 images, which no longer fits in memory. Training a modern deep convolutional network with tens of layers from scratch on a meta-learning objective may also lead to a hard learning problem.

Instead, we first train a convolutional network based feature extractor on a simple classification objective on the base classes C_{base}. Then we extract and save these features to disk, and use these pre-computed features as inputs. For most experiments, consistent with [144], we use a small ResNet-10 architecture [154]. Later, we show some experiments using the deeper ResNet-50 architecture [154].

Meta-learner architectures. We focus on state-of-the-art meta-learning approaches, including prototypical networks (PN) [351], matching networks (MN) [388], and our improvement over MN — prototype matching networks (PMN). For PN, the embedding architecture consists of two MLP layers with ReLU as the activation function. We use Euclidean distance as in [351]. For MN, following [388], the embedding architecture consists of a one layer bi-directional LSTM that embeds training examples and attention LSTM that embeds test samples. We use cosine distance as in [388]. For our PMN, we collapse every class to its class mean before the contextual embeddings of the test examples, and we keep other design choices the same as those in MN.

Hallucinator architecture and initialization. For our hallucinator G, we use a three layer MLP with ReLU as the activation function. We add a ReLU at the end since the pre-trained features are known to be non-negative. All hidden layers have a dimensionality of 512 for ResNet-10 features and 2048 for ResNet-50 features. Inspired by [223], we initialize the weights of our hallucinator network as block diagonal identity matrices. This significantly outperformed standard initialization methods like random Gaussian, since the hallucinator can "copy" its seed examples to produce a reasonable generation immediately from initialization.

11.6.2 Results

As in [144], we run five trials for each setting of n (the number of examples per novel class) and present the average performance. Different approaches are comparably good for base classes, achieving 92% top-5 accuracy. We focus more on novel classes since they are more important in few-shot learning. Table 11.1 contains a summary of the top-5 accuracy for novel classes and for the joint space both with and without a cross-validated prior. Standard deviations for all numbers are of the order of 0.2%. We discuss specific results, baselines, and ablations below.

Impact of Hallucination. We first compare meta-learners with and without hallucination to judge the impact of hallucination. We look at prototypical networks (PN) and prototype matching networks (PMN) for this comparison. Figure 11.4 shows the improvement in top-5 accuracy we get from hallucination on top of the original meta-learner performance. The actual numbers are shown in Table 11.1.

We find that our hallucination strategy improves novel class accuracy significantly, by up to 6 points for prototypical networks and 2 points for prototype matching networks. This suggests that our approach is general and can work with different meta-learners. While the improvement drops when more novel category training examples become available, the

Method	Novel n=1	2	5	10	20	All n=1	2	5	10	20	All with prior n=1	2	5	10	20
ResNet-10															
PMN w/ G*	**45.8**	**57.8**	**69.0**	**74.3**	**77.4**	**57.6**	**64.7**	**71.9**	**75.2**	**77.5**	**56.4**	**63.3**	**70.6**	**74.0**	76.2
PMN*	43.3	55.7	*68.4*	*74.0*	*77.0*	55.8	63.1	*71.1*	*75.0*	*77.1*	54.7	62.0	*70.2*	*73.9*	75.9
PN w/ G*	*45.0*	*55.9*	67.3	73.0	76.5	*56.9*	*63.2*	70.6	74.5	76.5	*55.6*	*62.1*	69.3	73.1	75.4
PN [?]	39.3	54.4	66.3	71.2	73.9	49.5	61.0	69.7	72.9	74.6	53.6	61.4	68.8	72.0	73.8
MN [?]	43.6	54.0	66.0	72.5	76.9	54.4	61.0	69.0	73.7	76.5	54.5	60.7	68.2	72.6	75.6
LogReg	38.4	51.1	64.8	71.6	76.6	40.8	49.9	64.2	71.9	76.9	52.9	60.4	68.6	72.9	*76.3*
LogReg w/ Analogies [?]	40.7	50.8	62.0	69.3	76.5	52.2	59.4	67.6	72.8	76.9	53.2	59.1	66.8	71.7	**76.3**
ResNet-50															
PMN w/ G*	**54.7**	**66.8**	**77.4**	**81.4**	**83.8**	**65.7**	**73.5**	**80.2**	**82.8**	**84.5**	**64.4**	**71.8**	**78.7**	**81.5**	**83.3**
PMN*	53.3	*65.2*	*75.9*	80.1	82.6	64.8	*72.1*	78.8	81.7	*83.3*	63.4	*70.8*	*77.9*	*80.9*	*82.7*
PN w/ G*	*53.9*	65.2	75.7	*80.2*	*82.8*	*65.2*	72.0	*78.9*	*81.7*	83.1	*63.9*	70.5	77.5	80.6	82.4
PN [?]	49.6	64.0	74.4	78.1	80.0	61.4	71.4	78.0	80.0	81.1	62.9	70.5	77.1	79.5	80.8
MN [?]	53.5	63.5	72.7	77.4	81.2	64.9	71.0	77.0	80.2	82.7	63.8	69.9	75.9	79.3	81.9

Table 11.1: Top-5 accuracy on the novel classes and on all classes (with and without priors) for different values of n. *Our methods. PN: Prototypical networks, MN: Matching networks, PMN: Prototype matching networks, LogReg: Logistic regression. Methods with "w/ G" use a meta-learned hallucinator.

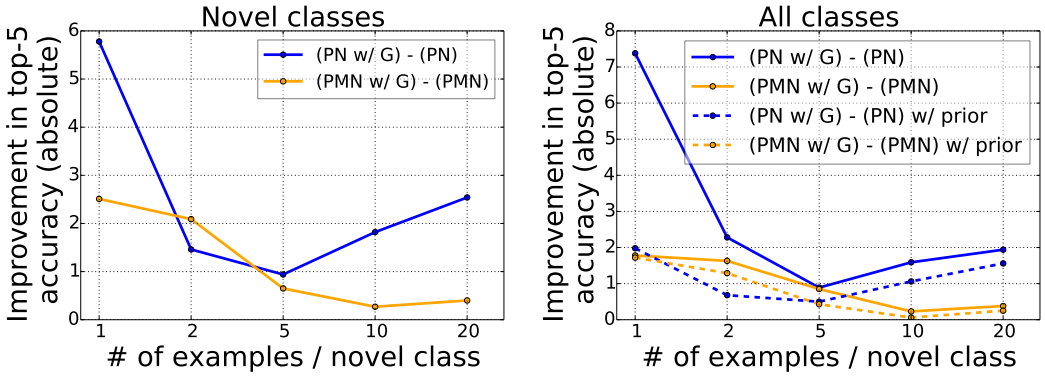


Figure 11.4: Improvement in accuracy by learned hallucination for different meta-learners as a function of the number of examples available per novel class.

gains remain significant until $n = 20$ for prototypical networks and $n = 5$ for prototype matching networks.

Accuracy in the joint label space (right half of Figure 11.4) shows the same trend. However, note that the gains from hallucination decrease significantly when we cross-validate for an appropriate novel-class prior $\boldsymbol{\mu}$ (shown in dotted lines). This suggests that part of the effect of hallucination is to provide resilience to mis-calibration. This is important in practice where it might not be possible to do extensive cross-validation; in this case, meta-learners with hallucination demonstrate significantly higher accuracy than their counterparts without hallucination.

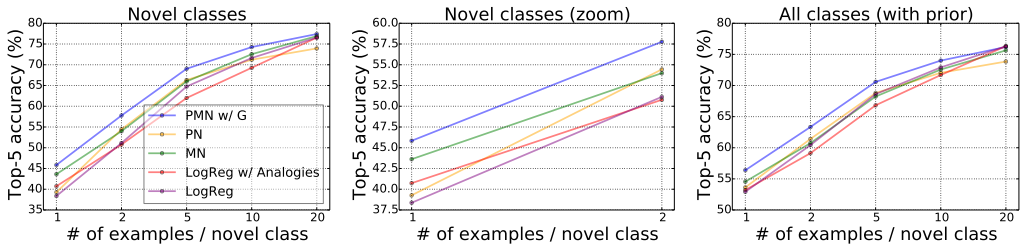


Figure 11.5: Our best approach compared to previously published methods. From left to right: just the novel classes, zoomed in performance for the case when the number of examples per novel class $n \leq 2$, performance on the joint label space with a cross-validated prior.

Comparison to Prior Work. Figure 11.5 and Table 11.1 compare our best approach (prototype matching networks with hallucination) with previously published approaches in few-shot learning. These include prototypical networks [351], matching networks [388], and the following baselines:

1. *Logistic regression*: This baseline simply trains a linear classifier on top of a pre-trained ConvNet-based feature extractor that was trained on the base classes.
2. *Logistic regression with analogies*: This baseline uses the procedure described in [144] to hallucinate additional examples. These additional examples are added to the training set and used to train the linear classifier.

Our approach easily outperforms all baselines, providing almost a 2 point improvement across the board on the novel classes, and similar improvements in the joint label space even after allowing for cross-validation of the novel category prior. Our approach is thus state-of-the-art.

Another intriguing finding is that our proposed prototype matching network outperforms matching networks on novel classes as more novel class examples become available (Table 11.1). On the joint label space, prototype matching networks are better across the board.

Interestingly, the method proposed in [144] *underperforms* the standard logistic regression baseline (although it does show gains when the novel class prior is not cross-validated, as shown in Table 11.1, indicating that its main impact is resilience to mis-calibration).

Unpacking the Performance Gain. To unpack where our performance gain is coming from, we perform a series of ablations to answer the following questions.

Are sophisticated hallucination architectures necessary?

In the semantic feature space learned by a convolutional network, a simple jittering of the training examples might be enough. We created several baseline hallucinators that did such jittering by: (a) adding Gaussian noise with a diagonal covariance matrix estimated from feature vectors from the base classes, (b) using dropout (PN/PMN w/ Dropout), and (c) generating new examples through a weighted average of real ones (PN/PMN w/ Weighted). For the Gaussian hallucinator, we evaluated both a covariance matrix shared across classes and class-specific covariances. We found that the shared covariance outperformed class-specific covariances by 0.7 point and reported the best results. We tried both

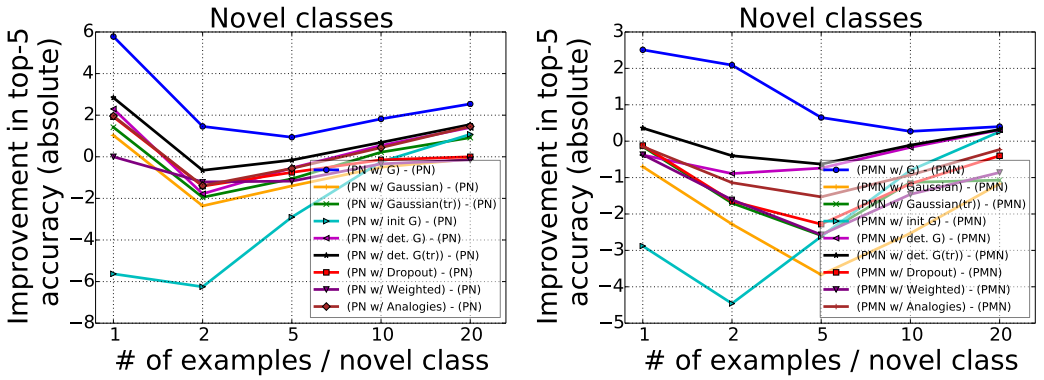


Figure 11.6: Comparison of our learned hallucination with several ablations for both PN (left) and PMN (right). Our approach significantly outperforms the baselines, showing that a meta-learned hallucinator is important. **Best viewed in color with zoom.**

retraining the meta-learner with this Gaussian hallucinator, and using a pre-trained meta-learner: PN/PMN w/ Gaussian uses a pre-trained meta-learner and PN/PMN w/ Gaussian(tr) retrains the meta-learner. As shown in Figure 11.6, while such hallucinations help a little, they often hurt significantly, and lag the accuracy of our approach by at least 3 points. This shows that generating useful hallucinations is not easy and requires sophisticated architectures.

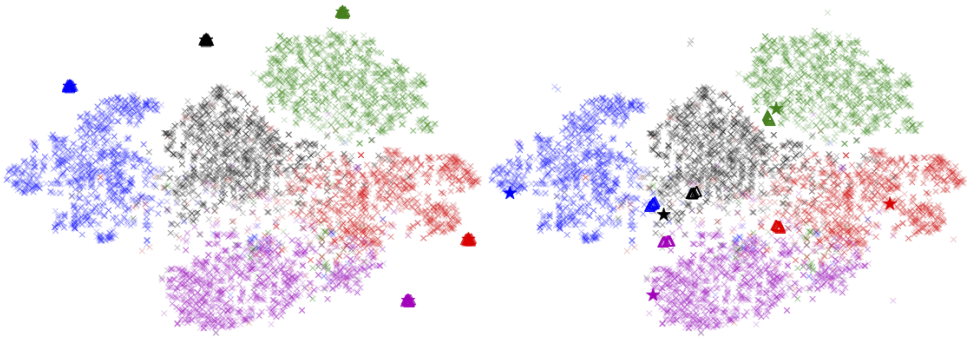
Is meta-learning the hallucinator necessary?

Simply passing Gaussian noise through an untrained convolutional network can produce complex distributions. In particular, ReLU activations might ensure the hallucinations are non-negative, like the real examples. We compared hallucinations with (a) an *untrained* G and (b) a *pre-trained and fixed* G based on analogies from [144] with our meta-trained version to see the impact of our training. Figure 11.6 shows the impact of these baseline hallucinators (labeled PN/PMN w/ init G and PN/PMN w/ Analogies, respectively). These baselines hurt accuracy significantly, suggesting that meta-training the hallucinator is important.

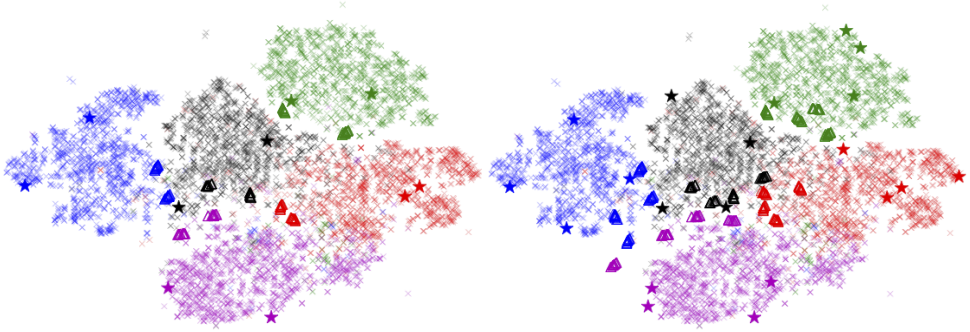
Does the hallucinator produce diverse outputs?

A persistent problem with generative models is that they fail to capture multiple modes [327]. If this is the case, then any one hallucination should look very much like the others, and simply replicating a single hallucination should be enough. We compared our approach with: (a) a deterministic baseline that uses our trained hallucinator, but simply uses a fixed noise vector $z = 0$ (PN/PMN w/ det. G) and (b) a baseline that uses replicated hallucinations during both training and testing (PN/PMN w/ det. G(tr)).These baselines had a very small, but negative effect. This suggests that our hallucinator produces *useful*, *diverse* samples.

Visualizing the Learned Hallucinations. Figure 11.7 shows t-SNE [384] visualizations of hallucinated examples for novel classes from our learned hallucinator and a baseline Gaussian hallucinator for prototypical networks. As before, we used statistics from the base class distribution for the Gaussian hallucinator. Note that t-SNE tends to expand out parts of the space where examples are heavily clustered together. Thus, the fact that the cloud of hallu-



(a) Gaussian baseline (b) G with 1 seed



(c) 2 seeds (d) 4 seeds

Figure 11.7: t-SNE visualizations of hallucinated examples. Seeds are shown as stars, real examples as crosses, hallucinations as triangles. (a) Gaussian, single seed. (b,c,d) Our approach, 1, 2, and 4 seeds respectively. **Best viewed in color with zoom.**

cinations for the Gaussian hallucinator is pulled away from the class distributions suggests that these hallucinations are very close to each other and far away from the rest of the class. In contrast, our hallucinator matches the class distributions more closely, and with different seed examples captures different parts of the space. Interestingly, our generated examples tend to cluster around the class boundaries. This might be an artifact of t-SNE, or perhaps a consequence of discriminative training of the hallucinator. However, our hallucinations are still fairly clustered; increasing the diversity of these hallucinations is an avenue for future work.

Representations from Deeper Models. All experiments till now used a feature representation trained using the ResNet-10 architecture [144]. The bottom half of Table 11.1 shows the results on features from a ResNet-50 architecture. As expected, all accuracies are higher, but our hallucination strategy still provides gains on top of both prototypical networks and prototype matching networks.

Chapter 12

Conclusions and Future Work

> Wir müssen wissen — wir werden wissen!
>
> — David Hilbert

This dissertation has made some progress towards endowing visual recognition systems with the ability of learning novel visual concepts from few examples. At first glance, this task seems daunting; current recognition systems are developed upon classic statistical machine learning, which is essentially guaranteed by the (weak) law of large numbers. However, to make low-shot learning possible, we need new ideas and to rethink the learning paradigm.

At the heart of the dissertation is the idea that we should leverage the structures and regularities of visual world not only in feature space but also in task and model space. By learning to learn such structures as appropriately tuned inductive biases, we are able to reuse previous experience, and address a new recognition task rapidly from limited training examples. In this spirit, we have addressed key technical challenges and explored different and complementary perspectives, including knowledge distillation, unsupervised meta-learning, continual learning, and generative learning.

In this final chapter, we take a step back from the work. We first discuss the observations, speculations, and lessons learned when developing our approaches, and address their limitations and possible future improvements. We then provide some other perspectives and potential future directions. Much work remains to be done in terms of understanding the low-shot learning mechanism and achieving applicable recognition performance.

12.1 Discussions: What Might be Wrong with Small Sample Learning?

Learning Process — Generating a Target Network without Extensive Data-Oriented Learning. Learning a recognition model is now generally cast as learning a *parametrized* neural network via gradient-based optimization. One partial reason why the model fails in the face of few labeled examples is that gradient-based optimization is slow and requires many weight updates using stochastic gradient descent (SGD) to gradually map training examples to model parameters [310, 388]. The variants of gradient-based optimization algorithms, such as momentum [275], Adagrad [98], Adadelta [428], and ADAM [199], were

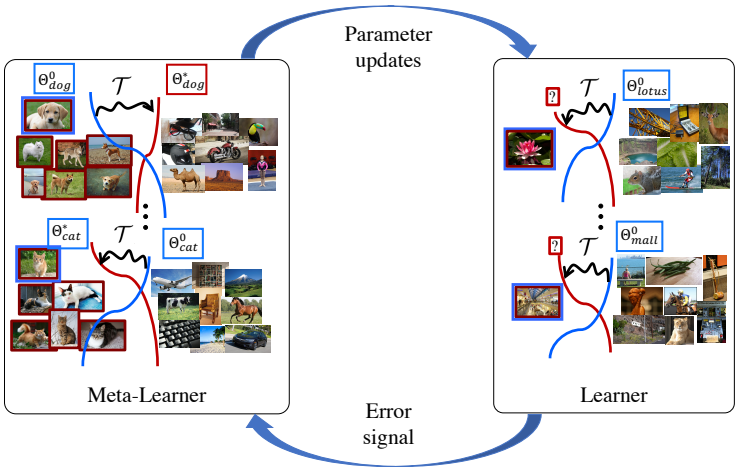


Figure 12.1: The meta-learner learns the model dynamics T from small-sample models Θ^0 (represented as blue) to the underlying large-small models Θ^* (represented as red) based on a large collection of model pairs from different recognition tasks. For a novel recognition task, the meta-learner proposes model updates to increase the learner's generalization in the small sample size regime.

not designed specifically to perform well under the constraint of a set number of updates. Specifically when applied to non-convex optimization problems, with a reasonable choice of hyper-parameters these algorithms do not have very strong guarantees of speed of convergence, beyond that they will eventually converge to a good solution after what could be many millions of iterations [310]. There is also recent exploration of training neural networks without back-propagation, such as [331].

In Part I, we provided a different perspective regarding model optimization by studying the relationship between models learned from few annotated samples and models learned from large enough sample sets. We then explicitly encoded their difference (*i.e.*, transformation) using a higher-level meta-model. Ideally, the meta-model is supposed to extract and contain *statistically sufficient* [404] information across a certain distribution of tasks. For a novel task from the same or relevant distribution, the meta-model allows to generate the corresponding model rapidly without extensive learning phases. The central issue then is, to make all this possible, what kind of information the meta-model should contain?

We explored "the direct change of model parameters" in Part I. Contemporary work also considered "estimation of gradient" [10, 182, 236, 310]. These approaches can be viewed as storing the *first-order information* in the space of model parameters. It is interesting to investigate *higher-order* parameter dynamics or other types of information. Moreover, we used a simple feed-forward network as the meta-model, which might not be enough to encode complex information. It is thus interesting to equip the meta-model with additional memory [135,329] through recurrent architectures in the form of recurrent neural networks (RNNs) or long short-term memories (LSTMs). In addition, our specific approach was developed in the context of simple linear classifiers, *i.e.*, the final layer of a CNN model. It is also interesting to explore the dynamics of other layers, including the fully-connected and convolutional layers, in a similar fashion as shown in Figure 12.1. This might be challenging due to the significant increase in the number of parameters and the mutual dependency of

parameters in hierarchical layers.

Loss Function in the Small Data Regime. Without the guarantee from the law of large numbers, it is likely that small-sample learning and large-sample learning follow different learning rules. Hence, the difficulty of small-sample learning might partially lie in the lack of an *appropriate objective function*, which might be significantly different from that in the large sample size regime. We tackled this slightly in this thesis. In Part IV, we directly hallucinated additional examples that were useful for the end-goal of learning classifiers, rather than aiming at the diversity or realism criterion as normally do. In Chapter 9, we observed that feature activations tended to become smaller when fine-tuning a pre-trained network on limited target data. A similar phenomenon has also been observed in [144], in which a novel loss function for representation learning is proposed to penalize the difference between classifiers learned on large and small datasets. More generally, what is an appropriate objective function when learning in the small sample size regime?

The recent seminal work of generative adversarial networks (GANs) [23,132,257,304,314, 327,433] might cast some interesting light on small-sample learning, owing to its promising learning mechanism [65]. Traditional machine learning needs to be given an explicit objective function, which evaluates how well the model is doing and which is typically effective with enough training samples. This objective function forms the basis of what the model learns and how well it learns. Typically, such an objective function is carefully, manually constructed. This is where the adversarial network shines. The adversarial network learns its own objective function — its own complex rules of what is correct and what is wrong — bypassing the need to carefully design and construct one [65].

Data Manufacturing via Unsupervised Regularization. A straightforward solution to low-shot learning is to generate additional data, as we did in Part IV or through generative adversarial networks (GANs). However, this line of work is still restrictive in the sense that the learned generator is tied to a specific set of categories due to its supervised nature. The challenge then is to enable the generator to hallucinate examples in the feature space with a good coverage of intra-class variation for a broad range of categories. Our low-density separators in Part II can be viewed as an effective, discriminative compression of the unsupervised data. For a novel category, these separators *implicitly* connect its few examples with the corresponding latent pseudo-classes in the feature space, thus providing additional bits of information for the recognition task.

Our approach suggests a way to hallucinate additional examples by leveraging the joint regularity from generative learning and a large amount of unlabeled real-world images, as shown in Figure 12.2. It is interesting to investigate different ways to combine a generator (*e.g.*, GAN) with our unsupervised meta-learning. For instance, (1) we first produce plausible pseudo-classes and further expand them via the generator; and (2) we directly generate examples while satisfying the statistics in the feature space, which is constrained by the unsupervised data (*e.g.*, using the maximum mean discrepancy (MMD) [136, 378] criterion). Other ways might be introducing tangent propagation which encourages invariant representations [346]. Moreover, the initial feature space could be fine-tuned as well during the sample generation process.

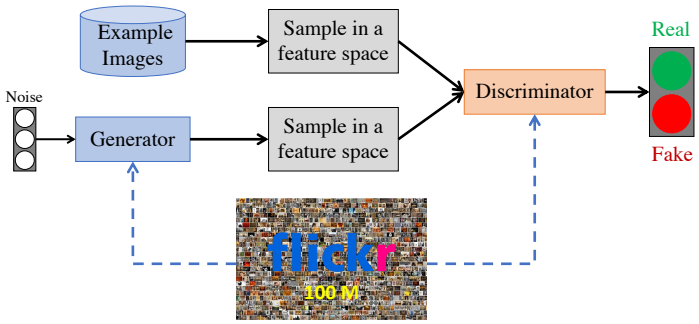


Figure 12.2: We generate additional examples by combing generative adversarial networks (GANs) with a large amount of unlabeled real-world images (*e.g.*, Flickr images) for learning from few examples. Plausible samples are directly generated in a feature space that is useful for recognition. The unsupervised data are exploited to satisfy the sample statistics in the desired feature space as well as to provide a good coverage of intra-class variation.

Developmental Learning of Generic Features. Assuming that we already have "the grand generic feature representation"[1], the small-sample learning problem might have been, at least partially, solved. In this case, a single example will suffice to discriminatively represent the entire class. The current approaches to learning CNNs, however, aim to learn a generic feature representation but they are based on a fixed set of visual concepts through representative images and a fixed model with constant capacity. This is not enough. In Part III, as improvements to the standard practice of fine-tuning a fixed-size network on labeled target data, we progressively grew a convolutional neural network with increased model capacity. Our approach thus suggests a developmental view of CNN optimization, in which model capacity is progressively grown throughout a lifelong learning process.

From a unified perspective, there is no clear boundary between learning an initial model and fine-tuning a pre-trained model, since both procedures are conducted using SGD. While our approach was developed in the scenario of fine-tuning pre-trained CNNs, it is interesting to explore if such a principle also benefits learning more generic CNN models from scratch, especially when learning from continuously evolving data streams and tasks. This strategy decouples the model from ties to the original specific set of categories and the limited representational capability.

In this framework, the resulting developmental neural network is a directed graph of pre-trained networks linked together with auxiliary, untrained units. It is interesting to investigate different configurations for constructing the network. In particular, inspired by the recent work on network modularization [8,9,169,210], we treat the already learned networks as modular components to compose a more advanced model. Rather than augment the network on unit levels, we now progressively grow the network by adding compositional modules or interleaved modules, as shown in Figure 12.3. Some specific architectures takes the forms of (1) introducing an entire new tower (*i.e.*, placing the pre-trained and new modules in parallel or in a stitch manner), which leads to a two-towers approach; (2) sequentially stack more modules on the top, which leads to a depth augmented modular network; (3) expanding certain existing modules, which leads to a width augmented modular network; and (4) their combinations.

[1]This term is inspired by the grand unified field theory in physics.

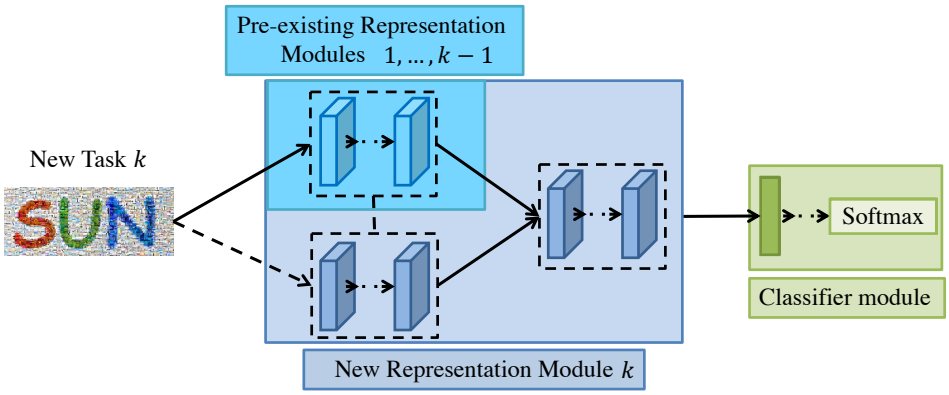


Figure 12.3: The developmental neural networks progressively grow in model capacity throughout a lifelong learning process, thus capturing a more generic and richer description of the visual world. We treat the pre-existing networks as modular components and introduce additional modules to compose a more advanced model via compositional or/and interleaved configurations as new tasks as encountered.

These additional modules help guide the adaptation of pre-existing modules. Compositional modules allow for new compositions of pre-existing modules, while interleaved modules allow for the discovery of complementary cues that address the new task. To improve the ability of learning without forgetting and make the network grow in a smooth manner, it is also interesting to investigate adding identity shortcuts and introducing weighting schemes for the pre-existing and new modules, training the new components while freezing the original modules or re-training the entire network.

During self-growing, the network is performing curriculum learning based on rich sets of tasks. It is interesting to investigate different types of tasks: (1) the same type of recognition tasks using different datasets, which consist of data sampled from easy to hard categories; (2) different recognition tasks using the same collection of sample images, such as from a coarse task of image classification to a fine-grained task of semantic segmentation; (3) tasks with different degree of supervision, from supervised tasks to self-supervised or unsupervised tasks. The corresponding learning objective functions of the network are constructed correspondingly and changed dynamically. For instance, in the unsupervised scenario, we generate a diverse set of pseudo-tasks as in Chapter 6. By leveraging the smoothness among tasks, the developmental network facilitates continual transfer across multiple tasks.

The Dynamics of a Large Collection of Models. In Part II, we showed the use of a large collection of models for the purpose of small-sample recognition. However, the analysis of the properties, behaviors, and applications of a large-scale model library goes beyond this thesis. In Part II, we mainly leveraged the intuition and observation that, in a large-scale library, it is likely that one of the library models happens to be tuned with the similar conditions as the new target task. In addition, we observed that the recognition performance of the library models on a new target task also followed a long-tail distribution. In Chapter 7, we further discovered a continuous category space based on the number of shared models

between different categories.

Traditionally, in machine learning, the use of multiple learning algorithms is discussed in ensemble learning [85, 438], which focuses on achieving better predictive performance than could be obtained from any of the constituent learning algorithms alone. The results in Part II suggest another promising direction — systematically analyzing the dynamics, equilibrium, synergetics, and self-organization of patterns and structures in such a large-scale model library system. By doing so, we might develop a corresponding theory similar to *thermodynamics* [52] and *synergetics* [142] in physics and have far-reaching applications.

Predictive Structure Learning: Exploration and Exploitation. Let us imagine that we are given a very large amount of unsupervised images as a much less biased sampling in a feature space. Within this universe of images, we have certain annotated image categories as well. On the one hand, the learning to learn approach discussed in Chapter 3 is essentially performing exploitation [233, 253, 284]: we probe a limited (but promising) region of the feature space (represented by the annotated categories) with the hope of improving a promising recognition model $\mathcal{M}$ that we already have at hand. During this process, we have acquired certain shared model structure. This operation amounts then to intensifying (refining) the search in the vicinity of $\mathcal{M}$, which is the learning of recognition models for relevant novel categories. On the other hand, the unsupervised meta-learning approach discussed in Chapter 6 is essentially performing exploration [233,253,284]: we probe a much larger portion of the feature space (but in a much coarse level due to the lack of supervision) with the hope of finding other promising recognition models that are yet to be refined. This operation amounts then to diversifying the search in order to have a good coverage of visual concepts and categories.

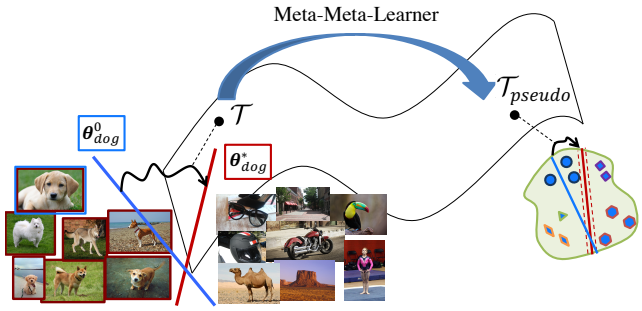


Figure 12.4: Illustration of exploration and exploitation for learning predictive model structure from annotated image categories and a very large amount of unsupervised images. After the meta-learners learn the model structure in the supervised region of the feature space, an additional meta-meta-learner is introduced to refine the meta-learners after exploring the unsupervised region of the feature space.

Such a perspective and the recent advance in processing large-scale datasets [187] thus suggest the combination of these two approaches. In addition to the meta-learners discussed in Chapter 3 (which learn model structures from small-sample models to large-sample models in the supervised region of the feature space), we introduce a higher-level meta-meta-learner that refines the meta-learner after exploring the unsupervised region of

the feature space through pseudo-classes and low-density separators discussed in Chapter 6. For instance, the meta-learners are instantiated as LSTMs and their hidden states are modified by the meta-meta-learner, which might be another LSTM. During the learning process, the model might progressively grow in a manner as in Chapter 9 and thus becomes more and more versatile and generic.

12.2 Other Perspectives and Future Directions

Exploratory Learning and Adversarial Interaction. In this thesis, we mainly focused on learning structures from collections of images, and paid little attention to the *interaction with the physical world*. Humans, especially babies and children, however, learn more in an exploratory and multi-modal way [350]. People live in a physical world, full of rich regularities that organize perception, action, and ultimately thought. Hence, people interact with the world through a vast array of sensory systems, including not only vision but also audition, touch, smell, proprioception, and balance. Importantly, people explore the world by moving and acting in highly variable and playful ways that are not goal-oriented. Through adversarial interactions, people accumulate and transfer previous experiences, and discover new problems and new solutions. In machine learning, such learning mechanism is studied in reinforcement learning [356] and more recently deep reinforcement learning [266]. It is interesting to address few-shot learning by combing exploratory learning with meta-learning and leveraging the state-of-the-art deep reinforcement learning [11] and multi-modal learning [19] techniques.

Reasoning using Symbolic, Hybrid Approaches. Another import aspect of human intelligence is the ability of reasoning [44, 99]. Machine reasoning is typically defined as "algebraically manipulating previously acquired knowledge in order to answer a new question" [44]. By transferring knowledge from common classes to rare classes, reasoning might benefit small-sample tail classes in long-tail recognition [64]. Earlier work in machine reasoning focuses on symbolic approaches [276], in which relations between abstract symbols are defined by the language of mathematics and logic and conclusions are generated by using logical techniques like deduction and induction. Recently, there is growing interest in integrating reasoning on top of deep representation learning [225,330], leading to hybrid approaches. It is interesting to explore how reasoning systems that consist of knowledge base and inference engine would effectively improve few-shot learning.

New Recognition Model Architectures. The backbone recognition model in this thesis was the state-of-the-art convolutional neural networks. At the heart of object recognition is equivalence and invariance [2,74]. Nonetheless, CNNs have some fundamental limitations, as pointed by the recent work [158, 323]. From the architecture design, the internal representation of CNNs does not take into account important orientational and relative spatial relationships between objects parts. Hence, CNNs require a large amount of labeled images from different view points to learn view-invariant representation. Some emerging model architectures, such as capsule networks [158, 323], explicitly encode relative relationships between objects and represent them as a 4D pose matrix. By combining our meta-learning approaches with such new architectures that have the built-in understanding of 3D space , we could further improve the recognition performance by only using a fraction of the data that a CNN would use.

High-Level Tasks. In this thesis, for evaluation purpose, we mainly focused on the task of image classification and we also considered other applications such as image clustering (Chapter 10), object detection (Chapter 7), image retrieval (Chapter 8), and human motion prediction (Chapter 5). Small sample learning is much broader concept and our learning to learn approaches are also general and can be potentially applied in other domains and tasks as well. Contemporary work also investigates other few-shot scenarios, such as semantic segmentation [307,340], robotic imitation learning [97,115], visual question answering [260, 362]. It is interesting to investigate how our approaches extend to these and other high-level few-shot learning tasks and generalize beyond vision domains.

Bibliography

[1] The infinite monkey theorem. http://en.wikipedia.org/wiki/Infinite_monkey_theorem.

[2] A. Achille and S. Soatto. Emergence of invariance and disentangling in deep representations. In *ICML Workshops*, 2017.

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

[4] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. P. Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *ECCV*, 2008.

[5] I. Akhter, T. Simon, S. Khan, I. Matthews, and Y. Sheikh. Bilinear spatiotemporal basis models. *ACM TOG*, 31(2):17, 2012.

[6] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *ICML*, 2007.

[7] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.

[8] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. In *NAACL*, 2016.

[9] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *CVPR*, 2016.

[10] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.

[11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[12] S. Awodey. *Category theory*. Oxford University Press, 2010.

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

[14] Y. Aytar and A. Zisserman. Enhancing exemplar SVMs using part level transfer regularization. In *BMVC*, 2012.

[15] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Factors of transferability for a generic ConvNet representation. *TPAMI*, 2015.

[16] H. Azizpour, A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. In *CVPR Workshops*, 2015.

[17] J. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *ICCV*, 2015.

[18] J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, 2014.

[19] T. Baltrušaitis, C. Ahuja, and L.-P. Morency. Multimodal machine learning: A survey and taxonomy. *IEEE TPAMI*, 2018.

[20] E. Barsoum, J. Kender, and Z. Liu. HP-GAN: Probabilistic 3D human motion prediction via GAN. *arXiv preprint arXiv:1711.09561*, 2017.

[21] E. Bart and S. Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *CVPR*, 2005.

[22] E. Bart and S. Ullman. Single-example learning of novel classes using representation by similarity. In *BMVC*, 2005.

[23] S. Bartunov and D. P. Vetrov. Fast adaptation in generative models with generative matching networks. *arXiv preprint arXiv:1612.02192*, 2016.

[24] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, 2009.

[25] J. Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[26] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. arXiv preprint arXiv:1306.6709, 2013.

[27] S. Ben-david, T. Lu, D. Pál, and M. Sotáková. Learning low density separators. In *AISTATS*, 2009.

[28] S. Ben-David, N. Eiron, and H. U. Simon. The computational complexity of densest region detection. *Journal of Computer and System Sciences*, 64(1):22–47, 2002.

[29] S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *COLT*, 2003.

[30] A. Bendale and T. Boult. Towards open world recognition. In *CVPR*, 2015.

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

[32] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei. On the optimization of a synaptic learning rule. In *Optimality in Artificial and Biological Neural Networks*, 1992.

[33] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. In *ICML Workshop on Unsupervised and Transfer Learning*, 2012.

[34] Y. Bengio. Deep learning: Progress in theory and attention mechanisms. In *CVPR Workshop on Deep Vision*, 2015.

[35] Y. Bengio. https://disqus.com/by/yoshuabengio/, 2016.

[36] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2007.

[37] K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *NIPS*, 1999.

[38] A. Bergamo and L. Torresani. Classemes and other classifier-based features for efficient object categorization. *IEEE TPAMI*, 36(10):1988–2001, 2014.

[39] A. Bergamo and L. Torresani. Meta-class features for large-scale object categorization on a budget. In *CVPR*, 2012.

[40] A. Bergamo, L. Torresani, and A. W. Fitzgibbon. Picodes: Learning a compact code for novel-category recognition. In *NIPS*, 2011.

[41] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, 2016.

[42] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[43] E. Bingham and H. Mannila. Random projection in dimensionality reduction: Applications to image and text data. In *SIGKDD*, 2001.

[44] L. Bottou. From machine learning to machine reasoning. *Machine learning*, 94(2):133–149, 2014.

[45] C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[46] P. Bradley, K. Bennett, and A. Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, 2000.

[47] M. Brand and A. Hertzmann. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.

[48] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688, 1993.

[49] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *KDD*, 2006.

[50] J. Burgess. *One-shot learning in discriminative neural networks*. PhD thesis, University of Cambridge, 2016.

[51] D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized nonnegative matrix factorization for data representation. *TPAMI*, 33(8):1548–1560, 2011.

[52] H. B. Callen. *Thermodynamics and an introduction to thermostatistics*. Wiley, 1985.

[53] M. A. Carreira-Perpinán and R. Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, 2015.

[54] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[55] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *ECCV*, 2016.

[56] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *AISTATS*, 2005.

[57] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. Adaptive Computation and Machine Learning. The MIT Press, 2006.

[58] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002.

[59] K. Chatfield, R. Arandjelović, O. Parkhi, and A. Zisserman. On-the-fly learning for visual search of large-scale image and video datasets. *International journal of multimedia information retrieval*, 4(2):75–93, 2015.

[60] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[61] K. Chatfield, K. Simonyan, and A. Zisserman. Efficient on-the-fly category retrieval using convnets and gpus. In *ACCV*, 2014.

[62] K. Chatfield and A. Zisserman. VISOR: Towards on-the-fly large-scale object category retrieval. In *ACCV*, 2012.

[63] R. Chattopadhyay, J. Ye, S. Panchanathan, W. Fan, and I. Davidson. Multisource domain adaptation and its application to early detection of fatigue. In *SIGKDD*, 2011.

[64] X. Chen, L.-J. Li, L. Fei-Fei, and A. Gupta. Iterative visual reasoning beyond convolutions. In *CVPR*, 2018.

[65] S. Chintala and Y. LeCun. A path to unsupervised learning through adversarial networks. https://code.facebook.com/posts/1587249151575490/a-path-to-unsupervised-learning-through-adversarial-networks/, 2016.

[66] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

[67] J. Choi, M. Rastegari, A. Farhadi, and L. Davis. Adding unlabeled samples to categories by learned attributes. In *CVPR*, 2013.

[68] M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 2010.

[69] F. Chollet et al. Keras. https://keras.io, 2015.

[70] B. Chu, V. Madhavan, O. Beijbom, J. Hoffman, and T. Darrell. Best practices for fine-tuning visual classifiers to new domains. In *ECCV Workshops*, 2016.

[71] R. G. Cinbis and S. Sclaroff. Contextual object detection using set-based classification. In *ECCV*, 2012.

[72] R. G. Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with Fisher vectors. In *ICCV*, 2013.

[73] A. Coates and A. Y. Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*. 2012.

[74] T. Cohen and M. Welling. Group equivariant convolutional networks. In *ICML*, 2016.

[75] D. Dai and L. V. Gool. Ensemble projection for semi-supervised image classification. In *ICCV*, 2013.

[76] D. Dai, M. Prasad, C. Leistner, and L. Van Gool. Ensemble partitioning for unsupervised image categorization. In *ECCV*. 2012.

[77] D. Dai and L. Van Gool. Unsupervised high-level feature learning by ensemble projection for semi-supervised image classification and image clustering. *arXiv preprint arXiv:1602.00955*, 2016.

[78] D. Dai, T. Wu, and S.-C. Zhu. Discovering scene categories by information projection and cluster sampling. In *CVPR*, 2010.

[79] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[80] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. Salakhutdinov. Good semi-supervised learning that requires a bad GAN. In *NIPS*, 2017.

[81] H. Daumé III. Frustratingly easy domain adaptation. In *ACL*, 2007.

[82] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, 39(1):1–38, 1977.

[83] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a Laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[84] L. Devroye and G. Lugosi. *Combinatorial methods in density estimation*. Springer Science & Business Media, 2012.

[85] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, 2000.

[86] M. Dixit, R. Kwitt, M. Niethammer, and N. Vasconcelos. AGA: Attribute-guided augmentation. In *CVPR*, 2017.

[87] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *ECCV*, 2016.

[88] T.-T. Do, A.-D. Doan, D.-T. Nguyen, and N.-M. Cheung. Binary hashing with semidefinite relaxation and augmented Lagrangian. In *ECCV*, 2016.

[89] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013.

[90] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[91] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[92] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *National Academy Sciences*, 100(10):5591–5596, 2003.

[93] M. Dorfer, R. Kelz, and G. Widmer. Deep linear discriminant analysis. In *ICLR*, 2016.

[94] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

[95] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, 2014.

[96] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, 2009.

[97] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In *NIPS*, 2017.

[98] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[99] J. S. B. Evans. *Bias in human reasoning: Causes and consequences.* Lawrence Erlbaum Associates, Inc, 1989.

[100] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[101] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[102] A. Farhadi, M. K. Tabrizi, I. Endres, and D. Forsyth. A latent model of discriminative aspect. In *CVPR*, 2009.

[103] L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *ICCV*, 2003.

[104] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE TPAMI*, 28(4):594–611, 2006.

[105] L. Fei-Fei. Knowledge transfer in learning to recognize visual objects classes. In *International Conference on Development and Learning*, 2006.

[106] L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

[107] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010.

[108] A. Ferencz, E. G. Learned-Miller, and J. Malik. Building a classification cascade for visual identification from one example. In *ICCV*, 2005.

[109] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[110] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, 2013.

[111] B. Fernando and T. Tuytelaars. Mining multiple queries for image retrieval: On-the-fly learning of an object-specific mid-level representation. In *ICCV*, 2013.

[112] M. Fink. *Acquiring a new class from a few examples: Learning recurrent domain structures in humans and machines*. PhD thesis, The Hebrew University of Jerusalem, 2011.

[113] M. Fink. Object classification from a single example utilizing class relevance metrics. In *NIPS*, 2005.

[114] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[115] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning (CoRL)*, 2017.

[116] F. Fleuret and G. Blanchard. Pattern recognition from one example by chopping. In *NIPS*, 2005.

[117] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *ICCV*, 2015.

[118] Y. Fu and L. Sigal. Semi-supervised vocabulary-informed learning. In *CVPR*, 2016.

[119] T. Furlanello, J. Zhao, A. M. Saxe, L. Itti, and B. S. Tjan. Active long term memory networks. arXiv preprint arXiv:1606.02355, 2016.

[120] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[121] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, and D. S. Phoenix. A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. *Science*, 2017.

[122] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*, 2016.

[123] P. Ghosh, J. Song, E. Aksan, and O. Hilliges. Learning human motion models for long-term predictions. In *3DV*, 2017.

[124] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[125] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[126] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[127] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

[128] B. Gong, K. Grauman, and F. Sha. Reshaping visual datasets for domain adaptation. In *NIPS*, 2013.

[129] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE TPAMI*, 35(12):2916–2929, 2013.

[130] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[131] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng. Measuring invariances in deep networks. In *NIPS*, 2009.

[132] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[133] R. Goroshin, M. Mathieu, and Y. LeCun. Learning to linearize under uncertainty. In *NIPS*, 2015.

[134] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *ICCV*, 2015.

[135] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[136] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *NIPS*, 2007.

[137] L.-Y. Gui, L. Gui, Y.-X. Wang, L.-P. Morency, and J. M. F. Moura. Factorized convolutional networks: Unsupervised fine-tuning for image clustering. In *WACV*, 2018.

[138] L.-Y. Gui, Y.-X. Wang, D. Ramanan, and J. M. F. Moura. Few-shot human motion prediction via meta-learning. In *ECCV Submission*, 2018.

[139] L. Gui and L.-P. Morency. Learning and transferring deep Convnet representations with group-sparse factorization. In *ICCV Workshops*, 2015.

[140] S. Gupta, J. Hoffman, and J. Malik. Cross modal distillation for supervision transfer. In *CVPR*, 2016.

[141] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[142] H. Haken. Synergetics. *Physics Bulletin*, 28(9):412, 1977.

[143] X. Han, B. Singh, V. I. Morariu, and L. S. Davis. VRFP: On-the-fly video retrieval using web images and fast fisher vector products. *IEEE TMM*, 19(7):1583–1595, 2017.

[144] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017.

[145] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[146] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012.

[147] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In *ICLR*, 2017.

[148] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE TKDE*, 21(9):1263–1284, 2009.

[149] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[150] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017.

[151] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013.

[152] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[153] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[154] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[155] D. Held, S. Thrun, and S. Savarese. Robust single-view instance recognition. In *ICRA*, 2016.

[156] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *CVPR*, 2012.

[157] T. Hertz, A. B. Hillel, and D. Weinshall. Learning a kernel function for classification with small training samples. In *ICML*, 2006.

[158] G. Hinton, N. Frosst, and S. Sabour. Matrix capsules with EM routing. In *ICLR*, 2018.

[159] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS workshops*, 2014.

[160] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158, 1995.

[161] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[162] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[163] J. Hoffman, E. Rodner, J. Donahue, T. Darrell, and K. Saenko. Efficient learning of domain-invariant image representations. In *ICLR*, 2013.

[164] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and T. Darrell. One-shot adaptation of supervised deep convolutional models. In *ICLR Workshops*, 2014.

[165] J. Hoffman, B. Kulis, T. Darrell, and K. Saenko. Discovering latent domains for multisource domain adaptation. In *ECCV*, 2012.

[166] C. Hu, D. Li, Y.-Z. Song, T. Xiang, and T. M. Hospedales. Sketch-a-classifier: Sketch-based photo classifier generation. In *CVPR*, 2018.

[167] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[168] R. Hu, P. Dollár, K. He, T. Darrell, and R. Girshick. Learning to segment every thing. In *CVPR*, 2018.

[169] R. Hu, M. Rohrbach, J. Andreas, T. Darrell, and K. Saenko. Modeling relationships in referential expressions with compositional modular networks. In *CVPR*, 2017.

[170] C. Huang, Y. Li, C. C. Loy, and X. Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016.

[171] D.-A. Huang and K. M. Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *ECCV*, 2014.

[172] S. Huang and D. Ramanan. Recognition in-the-tail: Training detectors for unusual pedestrians with synthetic imposters. In *CVPR*, 2007.

[173] M. Huh, P. Agrawal, and A. A. Efros. What makes ImageNet good for transfer learning? In *NIPS workshops*, 2016.

[174] W. Huitt and J. Hummel. Piaget's theory of cognitive development. *Educational psychology interactive*, 3(2):1–5, 2003.

[175] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[176] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998.

[177] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[178] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE TPAMI*, 36(7):1325–1339, 2014.

[179] G. Irie, Z. Li, X.-M. Wu, and S.-F. Chang. Locally linear hashing for extracting non-linear manifolds. In *CVPR*, 2014.

[180] P. Isola, J. J. Lim, and E. H. Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015.

[181] M. Jacomy, S. Heymann, T. Venturini, and M. Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization. *Medialab center of research*, 560, 2011.

[182] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. 2017.

[183] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-RNN: Deep learning on spatio-temporal graphs. In *CVPR*, 2016.

[184] H. Jégou, L. Amsaleg, C. Schmid, and P. Gros. Query adaptative locality sensitive hashing. In *ICASSP*, 2008.

[185] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

[186] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.

[187] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*, 2017.

[188] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[189] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2):181–214, 1994.

[190] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *ECCV*, 2016.

[191] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013.

[192] J. B. Kadane and N. A. Lazar. Methods and criteria for model selection. *Journal of the American Statistical Association*, 99(465):279–290, 2004.

[193] O. Kallenberg. *Probabilistic symmetries and invariance principles*. Springer Science & Business Media, 2006.

[194] A. Karatzoglou, M. Weimer, and A. J. Smola. Collaborative filtering on a budget. In *AISTATS*, 2010.

[195] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *ECCV*, 2012.

[196] W. Kienzle and K. Chellapilla. Personalized handwriting recognition via biased regularization. In *ICML*, 2006.

[197] J. Kim, R. D. Monteiro, and H. Park. Group sparsity in nonnegative matrix factorization. In *ICDM*, 2012.

[198] J. Kim and J. Collomosse. Incremental transfer learning for object recognition in streaming video. In *ICIP*, 2014.

[199] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[200] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

[201] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *NIPS*, 2015.

[202] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015.

[203] H. Koppula and A. Saxena. Learning spatio-temporal structure from RGB-D videos for human activity detection and anticipation. In *ICML*, 2013.

[204] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE TPAMI*, 38(1):14–29, 2016.

[205] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD*, 4(1):1–24, 2010.

[206] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[207] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM TOG*, 21(3):473–482, 2002.

[208] E. A. Krause, M. Zillich, T. E. Williams, and M. Scheutz. Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues. In *AAAI*, 2014.

[209] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalanditis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73, 2017.

[210] V. Krishnan and D. Ramanan. Tinkering under the hood: Interactive zero-shot learning with net surgery. *arXiv preprint arXiv:1612.04901*, 2016.

[211] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

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

[213] P. Kulkarni, G. Sharma, J. Zepeda, and L. Chevallier. Transfer learning via attributes for improved on-the-fly classification. In *WACV*, 2014.

[214] I. Kuzborskij, B. Caputo, and F. Orabona. Transfer learning through greedy subset selection. In *International Conference on Image Analysis and Processing (ICIAP)*, 2015.

[215] I. Kuzborskij and F. Orabona. Stability and hypothesis transfer learning. In *ICML*, 2013.

[216] I. Kuzborskij and F. Orabona. Fast rates by transferring from auxiliary hypotheses. *Machine Learning*, 106(2):171–195, 2017.

[217] I. Kuzborskij, F. Orabona, and B. Caputo. From N to N+1: Multiclass transfer incremental learning. In *CVPR*, 2013.

[218] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[219] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum. One shot learning of simple visual concepts. In *CogSci*, 2011.

[220] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. One-shot learning by inverting a compositional causal process. In *NIPS*, 2013.

[221] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 2016.

[222] N. D. Lawrence, J. C. Platt, and M. I. Jordan. Extensions of the informative vector machine. In *Deterministic and Statistical Methods in Machine Learning*. 2005.

[223] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[224] Q. V. Le, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[225] Y. LeCun. Power and the limits of deep learning. In *Workshop of AI and The Future of Work*, 2017.

[226] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[227] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[228] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

[229] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshops*, 2013.

[230] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006.

[231] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

[232] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML*, 2007.

[233] H. Lehtihet. Exploration and exploitation. https://www.researchgate.net/post/What_is_the_difference_between_exploration_vs_exploitation_intensification_vs_diversification_and_global_search_vs_local_search.

[234] K. Levi, M. Fink, and Y. Weiss. Learning from a small number of training examples by exploiting object categories. In *CVPR Workshops*, 2004.

[235] K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *CVPR*, 2004.

[236] K. Li and J. Malik. Learning to optimize. In *ICLR*, 2017.

[237] L.-J. Li, H. Su, E. P. Xing, and F.-F. Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.

[238] Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016.

[239] J. J. Lim, R. Salakhutdinov, and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *NIPS*, 2011.

[240] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 2014.

[241] G. Lin, C. Shen, D. Suter, and A. van den Hengel. A general two-step approach to learning-based hashing. In *ICCV*, 2013.

[242] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[243] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[244] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[245] B.-D. Liu, Y.-X. Wang, B. Shen, Y.-J. Zhang, and M. Hebert. Self-explanatory sparse representation for image classification. In *ECCV*, 2014.

[246] B.-D. Liu, Y.-X. Wang, Y.-J. Zhang, and B. Shen. Learning dictionary on manifolds for image classification. *Pattern Recognition*, 46(7):1879–1890, 2013.

[247] B. Liu, M. Dixit, R. Kwitt, and N. Vasconcelos. Feature space transfer for data augmentation. In *CVPR*, 2018.

[248] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[249] W. Liu, A. Rabinovich, and A. C. Berg. ParseNet: Looking wider to see better. In *ICLR workshop*, 2016.

[250] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernel. In *CVPR*, 2012.

[251] W. Liu, S. Zheng, S. Jia, L. Shen, and X. Fu. Sparse nonnegative matrix factorization with the elastic net. In *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2010.

[252] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *ICCV*, 2011.

[253] J. G. March. Exploration and exploitation in organizational learning. *Organization science*, 2(1):71–87, 1991.

[254] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *CVPR*, 2017.

[255] P. Matikainen, R. Sukthankar, and M. Hebert. Classifier ensemble recommendation. In *ECCV Workshop on Web-scale Vision and Social Media*, 2012.

[256] P. Matikainen, R. Sukthankar, and M. Hebert. Model recommendation for action recognition. In *CVPR*, 2012.

[257] A. Mehrotra and A. Dukkipati. Generative adversarial residual pairwise networks for one shot learning. *arXiv preprint arXiv:1703.08033*, 2017.

[258] E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from one example through shared densities on transforms. In *CVPR*, 2000.

[259] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learning. In *ICLR*, 2018.

[260] I. Misra, R. Girshick, R. Fergus, M. Hebert, A. Gupta, and L. van der Maaten. Learning by asking questions. In *CVPR*, 2018.

[261] I. Misra, A. Gupta, and M. Hebert. From red wine to red tomato: Composition with context. In *CVPR*, 2017.

[262] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016.

[263] I. Misra, Y.-X. Wang, and M. Hebert. Learning object models from few examples. In *SPIE Unmanned Systems Technology XVIII*, 2016.

[264] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016.

[265] T. M. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *AAAI*, 2015.

[266] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[267] Y. Movshovitz-Attias. *Dataset curation through renders and ontology matching*. PhD thesis, Carnegie Mellon University, 2015.

[268] Y. Movshovitz-Attias, Q. Yu, M. C. Stumpe, V. Shet, S. Arnoud, and L. Yatziv. Ontological supervision for fine grained classification of street view storefronts. In *CVPR*, 2015.

[269] L. Mukherjee, J. Peng, T. Sigmund, and V. Singh. Network flow formulations for learning binary hashing. In *ECCV*, 2016.

[270] T. Munkhdalai and H. Yu. Meta networks. In *ICML*, 2017.

[271] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[272] M. Naphade, J. R. Smith, J. Tesic, S.-F. Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. *IEEE MultiMedia*, 13(3):86–91, 2006.

[273] C. A. Nelson, M. L. Collins, and M. Luciana. *Handbook of developmental cognitive neuroscience*. MIT Press, 2001.

[274] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-20). Technical report, CUCS-005-96, 1996.

[275] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[276] A. Newell. Physical symbol systems. *Cognitive science*, 4(2):135–183, 1980.

[277] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *NIPS*, 2002.

[278] A. Nichol and J. Schulman. Reptile: A scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.

[279] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Sixth Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP)*, 2008.

[280] H. Noh, P. H. Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *CVPR*, 2016.

[281] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997.

[282] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR*, 2006.

[283] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[284] T. Osugi, D. Kim, and S. Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *ICDM*, 2005.

[285] W. Ouyang, X. Wang, C. Zhang, and X. Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *CVPR*, 2016.

[286] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[287] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009.

[288] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE TKDE*, 22(10):1345–1359, 2010.

[289] D. Park and D. Ramanan. Articulated pose estimation with tiny synthetic videos. In *CVPR*, 2015.

[290] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009.

[291] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS Workshops*, 2017.

[292] N. Patricia and B. Caputo. Learning to learn, from transfer learning to domain adaptation: A unifying perspective. In *CVPR*, 2014.

[293] G. Patterson, G. Van Horn, S. Belongie, P. Perona, and J. Hays. Tropel: Crowdsourcing detectors with minimal training. In *Human Computation and Crowdsourcing (HCOMP)*, 2015.

[294] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[295] V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *NIPS*, 2001.

[296] M. Pickett, R. Al-Rfou, L. Shao, and C. Tar. A growing long-term episodic & semantic memory. In *NIPS Workshops*, 2016.

[297] S. Pinker. How the mind works. *Annals of the New York Academy of Sciences*, 882(1):119–127, 1999.

[298] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Bilinear classifiers for visual recognition. In *NIPS*, 2009.

[299] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, 1999.

[300] H. Qi, M. Brown, and D. G. Lowe. Learning with imprinted weights. In *CVPR*, 2018.

[301] Q. Qian, R. Jin, S. Zhu, and Y. Lin. Fine-grained visual categorization via multi-stage metric learning. In *CVPR*, 2015.

[302] S. Qiao, C. Liu, W. Shen, and A. Yuille. Few-shot image recognition by predicting parameters from activations. In *CVPR*, 2018.

[303] A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008.

[304] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[305] I. Radosavovic, P. Dollár, R. Girshick, G. Gkioxari, and K. He. Data distillation: Towards omni-supervised learning. In *CVPR*, 2018.

[306] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, 2007.

[307] K. Rakelly, E. Shelhamer, T. Darrell, A. Efros, and S. Levine. Conditional networks for few-shot semantic segmentation. In *ICLR Workshop*, 2018.

[308] M. Ranzato. *Unsupervised learning of feature hierarchies*. PhD thesis, 2009.

[309] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*, 2012.

[310] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[311] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR Workshops*, 2014.

[312] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[313] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *CVPR*, 2017.

[314] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[315] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[316] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[317] E. Rodner and J. Denzler. One-shot learning of object categories using dependent Gaussian processes. In *Annual Symposium of the German Association for Pattern Recognition*, 2010.

[318] E. Rodner. Visual transfer learning: Informal introduction and literature overview. arXiv preprint arXiv:1211.1127, 2012.

[319] S. Ross, J. Zhou, Y. Yue, D. Dey, and J. A. Bagnell. Learning policies for contextual submodular prediction. In *ICML*, 2013.

[320] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[321] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[322] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[323] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *NIPS*, 2017.

[324] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.

[325] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, 2007.

[326] R. Salakhutdinov, J. Tenenbaum, and A. Torralba. One-shot learning with a hierarchical nonparametric Bayesian model. In *ICML Workshops*, 2012.

[327] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016.

[328] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. In *ICML*, 2016.

[329] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

[330] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017.

[331] B. Scellier and Y. Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[332] J. Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.

[333] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[334] J. Schmidhuber. A neural network that embeds its own meta-levels. In *IEEE International Conference on Neural Networks*, 1993.

[335] J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.

[336] J. Schmidhuber. Learning how to learn learning algorithms: Recursive self-improvement. https://uclmr.github.io/nampi/talk_slides/schmidhuber\discretionary{-}{}{}nampi.pdf, 2016.

[337] L. A. Schmidt. *Meaning and compositionality as statistical induction of categories and constraints*. PhD thesis, Massachusetts Institute of Technology, 2009.

[338] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[339] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*, 2013.

[340] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots. One-shot learning for semantic segmentation. In *BMVC*, 2017.

[341] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

[342] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *CVPR*, 2015.

[343] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016.

[344] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[345] O. Sigaud and A. Droniou. Towards deep developmental learning. *IEEE Transactions on Cognitive and Developmental Systems (TCDS)*, 8(2):90–114, 2016.

[346] P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition—tangent distance and tangent propagation. In *Neural networks: tricks of the trade*. 1998.

[347] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[348] A. Sinha, M. Sarkar, A. Mukherjee, and B. Krishnamurthy. Introspection: Accelerating neural network training by learning weight evolution. In *ICLR*, 2017.

[349] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *ICCV*, 2005.

[350] L. Smith and M. Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

[351] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[352] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013.

[353] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[354] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

[355] I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted Boltzmann machine. In *NIPS*, 2009.

[356] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[357] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[358] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Web-scale training for face identification. In *CVPR*, 2015.

[359] G. W. Taylor and G. E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *ICML*, 2009.

[360] G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In *NIPS*, 2007.

[361] G. W. Taylor, L. Sigal, D. J. Fleet, and G. E. Hinton. Dynamical binary latent variable models for 3D human pose tracking. In *CVPR*, 2010.

[362] D. Teney and A. v. d. Hengel. Visual question answering as a meta learning task. *arXiv preprint arXiv:1711.08105*, 2017.

[363] A. V. Terekhov, G. Montone, and J. K. O'Regan. Knowledge transfer in deep block-modular neural networks. In *Conference on Biomimetic and Biohybrid Systems*, 2015.

[364] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[365] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520, 1996.

[366] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[367] S. Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, 1996.

[368] S. Thrun. Lifelong learning algorithms. In *Learning to learn*. 1998.

[369] S. Thrun and T. M. Mitchell. Learning one more thing. In *IJCAI*, 1995.

[370] S. Thrun and J. O'Sullivan. Clustering learning tasks and the selective cross-task transfer of knowledge. In *Learning to learn*. 1998.

[371] T. Tommasi. *Learning to learn by exploiting prior knowledge*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2013.

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

[373] T. Tommasi, F. Orabona, and B. Caputo. Learning categories from few examples with multi model knowledge transfer. *IEEE TPAMI*, 36(5):928–941, 2014.

[374] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE TPAMI*, 29(5):854–869, 2007.

[375] A. Torralba and A. Quattoni. Recognizing indoor scenes. In *CVPR*, 2009.

[376] E. Triantafillou, R. Zemel, and R. Urtasun. Few-shot learning through an information retrieval lens. In *NIPS*, 2017.

[377] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller. A deep semi-NMF model for learning hidden representations. In *ICML*, 2014.

[378] Y.-H. H. Tsai, L.-K. Huang, and R. Salakhutdinov. Learning robust visual-semantic embeddings. In *ICCV*, 2017.

[379] A. B. Tsybakov et al. On nonparametric estimation of density level sets. *The Annals of Statistics*, 25(3):948–969, 1997.

[380] A. Tsymbal. The problem of concept drift: Definitions and related work. Technical report, Computer Science Department, Trinity College Dublin, 2004.

[381] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015.

[382] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[383] R. Urtasun, D. J. Fleet, A. Geiger, J. Popović, T. J. Darrell, and N. D. Lawrence. Topologically-constrained latent variable models. In *ICML*, 2008.

[384] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 9(Nov):2579–2605, 2008.

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

[386] V. N. Vapnik. *Statistical learning theory*. Wiley New York, 1998.

[387] D. Vernon, C. Von Hofsten, and L. Fadiga. *A roadmap for cognitive development in humanoid robots*. Springer Science & Business Media, 2011.

[388] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[389] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

[390] J. Wan, Q. Ruan, W. Li, and S. Deng. One-shot learning gesture recognition from RGB-D data using bag of features. *JMLR*, 14(1):2549–2582, 2013.

[391] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE TPAMI*, 30(2):283–298, 2008.

[392] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE TPAMI*, 34(12):2393–2406, 2012.

[393] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[394] Y.-X. Wang and M. Hebert. Learning from small sample sets by combining unsupervised meta-training with CNNs. In *NIPS*, 2016.

[395] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016.

[396] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan. Low-shot learning from imaginary data. In *CVPR*, 2018.

[397] Y.-X. Wang and M. Hebert. Model recommendation: Generating object detectors from few samples. In *CVPR*, 2015.

[398] Y.-X. Wang and M. Hebert. Learning by transferring from unsupervised universal sources. In *AAAI*, 2016.

[399] Y.-X. Wang and M. Hebert. Few-shot hash learning for image retrieval. In *ICCV Workshops*, 2017.

[400] Y.-X. Wang, D. Ramanan, and M. Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *CVPR*, 2017.

[401] Y.-X. Wang, D. Ramanan, and M. Hebert. Learning to model the tail. In *NIPS*, 2017.

[402] Y.-X. Wang and Y.-J. Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE TKDE*, 25(6):1336–1353, 2013.

[403] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang. Learning a task-specific deep architecture for clustering. In *ICDM*.

[404] L. Wasserman. *All of statistics: A concise course in statistical inference*. Springer Science & Business Media, 2013.

[405] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Maximum margin matrix factorization for collaborative ranking. In *NIPS*, 2007.

[406] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2009.

[407] J. Weston, R. Collobert, F. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *ICML*, 2006.

[408] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. In *ICML*, 2008.

[409] L. Wolf, T. Hassner, and Y. Taigman. The one-shot similarity kernel. In *ICCV*, 2009.

[410] L. Wolf and I. Martin. Robust boosting for learning from few examples. In *CVPR*, 2005.

[411] A. Wong and A. L. Yuille. One shot learning via compositions of meaningful patches. In *ICCV*, 2015.

[412] M. Woodward and C. Finn. Active one-shot learning. In *NIPS Deep Reinforcement Learning Workshop*, 2017.

[413] M. Wu. Collaborative filtering via ensembles of matrix factorizations. In *Proceedings of KDD Cup and Workshop*, 2007.

[414] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva. SUN database: Exploring a large collection of scene categories. *IJCV*, 119(1):3–22, 2016.

[415] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2015.

[416] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of International ACM SIGIR Conference on Research and Development in Informaion Retrieval (ACM SIGIR)*, 2003.

[417] Z. Xu, L. Zhu, and Y. Yang. Few-shot object recognition from machine-labeled web images. In *CVPR*, 2017.

[418] J. Yang, R. Yan, and A. Hauptmann. Adapting SVM classifiers to data with shifted distributions. In *ICDM Workshops*, 2007.

[419] J. Yang, R. Yan, and A. Hauptmann. Cross-domain video concept detection using adaptive svms. In *ACM MM*, 2007.

[420] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

[421] J. Yang and A. G. Hauptmann. A framework for classifier adaptation and its applications in concept detection. In *ACM international conference on Multimedia information retrieval (MIR)*, 2008.

[422] S. Yang and D. Ramanan. Multi-scale recognition with DAG-CNNs. In *ICCV*, 2015.

[423] Z. Yang, Y. Yuan, Y. Wu, W. W. Cohen, and R. R. Salakhutdinov. Review networks for caption generation. In *NIPS*, 2016.

[424] B. Yao, G. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, 2012.

[425] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *ICCV*, 2011.

[426] D. Yoo, S. Park, J.-Y. Lee, and S. Kweon. Multi-scale pyramid pooling for deep convolutional representation. In *CVPR Workshops*, 2015.

[427] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[428] M. D. Zeiler. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[429] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[430] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE TIP*, 24(12):4766–4779, 2015.

[431] Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In *CVPR*, 2016.

[432] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 2015.

[433] J. Zhao, M. Mathieu, and Y. Lecun. Energy-based generative adversarial networks. In *ICLR*, 2017.

[434] L. Zheng, Y. Zhao, S. Wang, J. Wang, and Q. Tian. Good practice in CNN feature transfer. arXiv preprint arXiv:1604.00133, 2016.

[435] Q. Zhong, C. Li, Y. Zhang, H. Sun, S. Yang, D. Xie, and S. Pu. Towards good practices for recognition & detection. In *CVPR workshops*, 2016.

[436] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE TPAMI*, 2017.

[437] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014.

[438] Z.-H. Zhou. *Ensemble methods: Foundations and algorithms*. CRC Press, 2012.

[439] X. Zhu, D. Anguelov, and D. Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014.

[440] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan. Do we need more training data? *IJCV*, 119(1):76–92, 2016.

[441] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, 2005.

[442] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[443] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, March 2005.