Open-World Machine Learning with Limited Labeled Data

by

Mengye Ren

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Department of Computer Science
University of Toronto

© Copyright 2022 by Mengye Ren

Open-World Machine Learning with Limited Labeled Data

Mengye Ren
Doctor of Philosophy

Department of Computer Science
University of Toronto
2022

Abstract

Over the past decades, machine learning has made great strides in enabling numerous artificial intelligence applications. Yet, most of its success relies on training models in a closed environment with a massive amount of curated data offline, and evaluating them in a similar test environment afterwards. This means most of the machine learning models cannot quickly adapt to new environments and learn new knowledge online with very few observations. In contrast, our human brain can learn new representations, concepts, and skills from an online stream of sensory inputs. This thesis aims to enable machines with several core abilities to learn new concepts in an open world without access to a massive amount of curated labeled data. Specifically, it addresses several key problems such as learning with limited labeled data, incremental data, unlabeled data, and imbalanced and noisy data. The algorithms proposed in this thesis can be naturally combined with any deep neural network and are agnostic to the network architecture. They can offer much greater flexibility and robustness for various open-world conditions, making learning-based approaches suitable to be deployed in general agent-based intelligent systems.

Acknowledgements

First and foremost, my gratitude is to Rich Zemel, my PhD advisor, for the superb guidance throughout my journey as a graduate student. It is always enjoyable to chat research with him. I am thankful for his open-mindedness and flexibility in my research directions, and in turn, I became more passionate about the topics that I ended up working on. I have so much to learn from him: his visionary taste, generalist approach, humor, leadership, as well as the balance between work and life. The house parties hosted by him and Toni were the most fun and cheerful times in my memories.

I have spent half of my time during my PhD working as a research scientist in the self-driving industry, and I cannot fully express my gratitude to Raquel Urtasun for taking me on this unique, exciting, and rewarding ride. She is my role model for her one-of-a-kind vision, leadership, and execution. Under her leadership, I had the opportunity to work with many amazing colleagues and learn how to navigate research projects. I am in debt to her countless iterations on my presentations and writings. I learned to become a more responsible and considerate researcher because of her guidance.

I am thankful to Roger Grosse, who kindly agreed to serve on my thesis committee and gave lots of insightful suggestions. I am always excited by his ingenious ideas and influenced by his integrity and principles. I thank Yee Whye Teh and Geoffery Hinton, who also served as my thesis examiner, and contributed many valuable comments. I am in debt to Sanja Fidler for her encouragement and constructive critiques on my presentations and writings. She helped me transition my perspective from a graduate student to an independent researcher. I am grateful to Mike Mozer for his articulated thoughts and vivid writings. I benefited a lot from our collaborations and he inspired me to be a better communicator.

I am gracious to my internship mentors throughout my journey: Jamie Kiros, Dumitru Erhan, and Kevin Swersky. Jamie took me on board with my very first research project and generously offered me novel ideas while I was still struggling to derive the backpropagation formulas of LSTMs. I thank Dumitru and Kevin for offering me internship opportunities at Google and Twitter at the time I barely knew how to do research. Dumitru taught me how to write industrial-grade research code and Kevin spent countless hours helping me set up the entire research workflow. Although doing research is often stressful, I had not experienced any pressure from working with these amazing mentors. They taught me that research is not always about publishing papers.

I was fortunate to start my graduate studies with three amazing colleagues in the same year: Eleni Triantafillou, Renjie Liao, and Tony Wu. We have spent many trips and left precious memories together in Houston, Washington DC, Montréal, Vancouver, Spain, and France, and I am lucky enough to have closely collaborated with them on several papers throughout my PhD years. Eleni and I share an interest in meta-learning and self-supervised learning and we have organized successful series of reading groups on both topics. Both Renjie and I split our time across Uber and the University, and we have had many common views and but also intellectual debates. I always enjoy my open and deep conversations with Tony. I also thank Bill Li, my awesome ex-roommate and colleague, for all the thought-provoking conversations, homemade cocktails, and jazz jam. I thank them for being amazing coworkers and friends.

This thesis would not have been possible without the joint effort of my collaborators of the papers that the thesis is based on. In addition to the people mentioned above, I thank Hugo Larochelle, Sachin Ravi, Jake Snell, Josh Tenenbaum, Ethan Fetaya, Michael Iuzzolino, Tyler Scott, Bin Yang

and Wenyuan Zeng for our amazing collaboration. It is my honor to have worked with them.

My gratitude is also to Matthias Bethege for generously offering me to visit his lab at the University of Tübingen in Germany for a month in 2018. I had such an enjoyable time there during research discussions, soccer game barbeques, and beer nights. His hospitality and cheerful attitude helped build a sociable side of my personality. Because of this exchange experience, I also had many thoughtful chats with Alex Ecker, Wieland Brendel, Mara Weis, and Claudio Michaelis.

I am thankful to Andreas Tolias, Xaq Pitkow, and Fabian Sinz for generously hosting many lab visits at Baylor College of Medicine in Houston, and I had truly benefited from the interdisciplinary collaboration with them in between the fields of neuroscience and machine learning. These visits helped shape my research vision towards developing more human-like machine learning algorithms.

I am fortunate to be part of the Z-Group with many friendly and enthusiastic colleagues: Jackson Wang, Alex Wang, James Lucas, Yujia Li, Jörn Jacobsen, Marc-Etienne Brunet, David Madras, Elliot Creager, Will Grathwohl, Kamyar Seyed Ghasemipour, Lisa Zhang, and others. I enjoyed our collaboration and had learned a lot in our weekly Z-Group meetings where I can feel the sense of community.

As a member of the University of Toronto and Vector Institute, I am grateful to many faculties for collaboration and conversations, in addition to the people mentioned: Toni Pitassi, David Duvenaud, Jimmy Ba, Marzyeh Ghassemi and others, and to student colleagues: Shengyang Sun, Guodong Zhang, Tingwu Wang, Ethan Wen, Huan Ling, Hang Chu, Aidan Gomez, Laleh Seyyed-Kalantari, and many others. I thank the CSC 411 teaching team: Matthew MacKay, Xiaohui Zeng, Jixuan Wang, Silviu Pitis, Shun Liao, and Rasa Hosseinzadeh. I had learned a lot from my first time being a course instructor.

I thank my ex-Uber and Waabi colleagues for this unique work and study experience in the past four years, to Yuwen Xiong, Wei-Chiu Ma, Quinlan Sykora, Chris Zhang, Jingkang Wang, Nicholas Vadivelu, Bob Wei, James Tu, Shuhan Tan, Lunjun Zhang, Ava Pun, Min Bai, Gallért Máttyus, Xinchen Yan, Eilyan Bitar, Yen-Chen Lin, Shenlong Wang, Siva Manivasagam, Abbas Sadat, Sergio Casas, Elaine Papa, Ming Liang, Kelvin Wong, Andrei Pokrovsky, Andrei Bârsan, Julieta Martinez, Luisa San Martin, Rui Hu and many others. In addition to collaborating on many projects, we also shared countless moments together in soccer games, ping-pong tournaments, ski and rock climbing trips, and road trips to Hawaii, Pittsburgh, and Muskoka. I am thankful to all my friends outside the workplace for spending warm and fun time together.

I thank Martin Min, Jakob Verbeek, Yixin Wang, Sarath Chandar, and Jim Fan for generously inviting me for a talk at their institutions, on topics related to this thesis. I acknowledge the NSERC Alexander Graham Bell Doctorate Scholarship and the IARPA MICrONS program for funding my PhD studies and I thank these national research programs for their support.

Lastly, my utmost gratitude goes to my parents, Yue and Lizhi, and my grandmothers Lanfen and Bingyu for raising and supporting me throughout the years, so that I can spend carefree days and nights pursuing my dream. I especially owe a huge thank you to my dear mother Yue. My progress and breakthroughs would have never been possible without her unwavering love and support and her brave and liberal mind.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's?

Alan Turing

Machine learning is one of the central topics in the field of artificial intelligence (AI). Since many intelligent behaviors cannot be easily defined by a standard program, instead of relying on manually designed rules, we use machine learning to obtain function approximations given many input and output observations. Today, with the help of machine learning, our computers can recognize our voice and handwriting, remember our faces, tag our photos, translate different languages, beat us in playing chess and go, and drive cars safely on the road. Just like what Alan Turing envisioned in the 1950s, computers today use machine learning to "simulate" the child's mind, a blank sheet that is gradually filled with a great variety of knowledge and representations.

The learning process of a machine, however, is still far from that of a child. Perhaps one of the most phenomenal differences between machine and human learning is the ability to learn tasks with scarce data in the natural world. Machine learning today often relies on training models in a *closed world* environment with a massive amount of curated data and evaluating them in a similar or identical test environment afterwards. This means, unlike humans, standard machine learning algorithms cannot quickly adapt to new environments and learn new knowledge online with very few observations. Throughout the thesis, we refer to this desired ability as *open-world* learning.

How can we bridge this apparent gap between humans and machines? My thesis aims to seek solutions that enable machines to learn new concepts in an *open world* without access to a large amount of curated labels. Specifically, it addresses several key problems under the umbrella of open-world learning, such as learning with limited labeled data, incremental data, unlabeled data, imbalanced and noisy data, and online and streaming data, all of which are not being considered in a typical machine learning pipeline today.

The ultimate solution to these problems will have profound implications for all of us. First, it will allow future intelligent agents to learn on the fly: your future home robot will adapt to your house, recognize new furniture, and learn to use new gadgets; your augmented reality eyeglasses will learn through your perspective of the world, fed with glimpses that have never been experienced in the

past; your personal AI assistant will adapt to your preferences and learn new skills in a conversation with you. Moreover, it will save millions of hours of engineering, labeling, and data curation effort in many industrial applications. Lastly, it will also be a milestone in the quest of understanding our human intelligence, by casting our learning process into a computational framework.

1.1 Overview of the Thesis

This thesis presents contributions that enable machines to acquire new concepts with very few labeled examples and make them more robust to many naturalistic and open-world conditions. In the past, there have been several machine learning paradigms, such as few-shot learning, continual learning, self-supervised learning, and so on, that are all motivated by this bigger picture of making machine learning more flexible and adaptive in an open world. Chapter 2 surveys the background literature of these topics. Specifically, it first discusses various learning paradigms that encourage learning at test time in a different environment from training, such as few-shot learning and continual learning, and then I discuss another thread of related research that aims to learn from unlabeled examples, e.g. self-supervised learning.

These learning paradigms, however, typically place narrow focus on one specific property such as the amount of domain shift or the number of labeled data points. Occasionally, the properties are orthogonal and their solutions can be composed together, but oftentimes the proposed solutions rely on some additional unrealistic assumptions. For example, standard semi-supervised learning leverage unlabeled data to improve the quality of the learned model; however, it assumes that the unlabeled data come from the same distribution as labeled ones and also belong to one of the pre-defined classes. In another example, standard few-shot learning aims to learn new classes with very few data points, but it assumes that the data points are equally distributed from a few new classes that are never seen during training. Or, the class imbalance problem often assumes that the class labels are correct and therefore a high training cost implies that the data point is from the minority class. In these examples, solutions that make assumptions of other properties of the learning environment can potentially break down when they are deployed in the open world when multiple problems co-exist.

Therefore, the core theme of this thesis is to seek new solutions that can address multiple open-world properties simultaneously, such as learning with limited labeled data, incremental growth of output space, unlabeled, imbalanced, and noisy data. To reach this goal requires us not only to develop new learning algorithms but also to rethink the learning paradigms that define the problems. Hence, a portion of the thesis, such as parts of Chapter 4 and 6, also aim to define new learning paradigms or benchmarks with additional naturalistic properties.

The literature of learning with limited labeled data is widely known as *few-shot learning*. Standard few-shot learning, however, only deals with a few new classes at test time. In Chapter 3, we focus on the problem of *incremental few-shot learning*, where the model needs to recognize both *old* classes that are seen many times during training and *new* classes that are just introduced at test time. Surprisingly, we find that many classical few-shot learning methods, which only focuses on solving new classes, actually suffers at dealing with this more realistic problem of combining old and new classes, likely because the representations of old and new classes are not compatible with each other. Unlike traditional solutions that directly use some feature vectors of the new class examples as classifier weights, our proposed method is based on continuous optimization which solves the weights

by balancing objectives brought by old and new classes, and reaches a better-optimized solution at test time.

Throughout the process of incrementally learning new classes, a real-world agent often encounters many more examples that are unlabeled than labeled. In Chapter 4, we make another step forward and introduce unlabeled data to the problem of few-shot learning. We propose a new learning paradigm of *semi-supervised few-shot learning* that considers unlabeled examples in addition to the constraint that very few data points are labeled in each learning episode. Our work is the first that addresses both semi-supervised learning and few-shot learning jointly. It not only reduces the reliance on the amount of labeled data in training and test tasks but also addresses the issue of *distractors*, classes that do not belong to any known classes, as it is not considered in classic semi-supervised learning. We propose new few-shot learning models that can circumvent being impacted by distractor classes, while still manage to leverage useful information from unlabeled data.

Despite the wide success of few-shot learning, episodes are typically sampled from a well-curated dataset instead of the noisy long-tailed distribution from the natural world. The distractor examples we introduced in Chapter 4 can also be thought of as a form of noisy training data. In Chapter 5, we study the problem of learning with *imbalanced and noisy class labels* in a standard machine learning context. Although both problems are commonly occurring in naturalistic learning environments, traditionally they have been separately studied with contradictory remedies. To address the conflict, we propose a data-driven example weighting mechanism that can be directly applied to both problems under a unified framework. Our algorithm leverages the clean and balanced validation set to calibrate the training example weights. Our model also highlights an efficient way for learning to learn that jointly updates inner and outer loop parameters at the same time.

Few-shot learning often comes with a rigid episodic setup, which makes it unnatural for modeling the continual incremental acquisition of new concepts. In Chapter 6, we propose a new learning paradigm of *online contextualized few-shot learning*. Although we have studied combining old and new classes in Chapter 3, the previous method focuses much on the notion of an episode but knowledge never grows sequentially and incrementally over time. While there have been some efforts in making these episodes more sequential just like the setup of incremental class learning, the separation of training and testing phases is still making it onerous to evaluate. Real-world agents do not rely on episodic stops but instead performs online continual learning that produces some output prediction every timestep in a sequence, modulated through a top-down stream of contextual information. Our new paradigm incorporates many naturalistic properties such as online, incremental, contextualized, few-shot, and semi-supervised, and we also develop a new benchmark based on indoor home imagery that mimics the visual input stream of a real-world agent. Lastly, we propose a new model, Contextual Prototypical Memory (CPM), that successfully tackles this problem of online contextualized class learning with limited labeled data.

Finally, in Chapter 7 we investigate learning of representations and categories on the fly through an online stream of visual inputs, without using any class labels. In previous chapters, learning is still largely driven by labeled examples: for instance, in Chapter 6, new category clusters are not created until the environment tells the agent that it is a new class. In this chapter, we introduce an algorithm that allows agents to instead simultaneously learn representations and categories from an unlabeled data stream. This can be seen as a preceding stage in the developmental process, as an agent can first explore the environment by learning representations and classes without labeled data,

and then go under supervision with a few examples. Our proposed model, *the online unsupervised prototypical network*, combines the prototypical network for concept learning with clustering-based self-supervision for representation learning and compares favorably to state-of-the-art self-supervised visual representation learning methods using only online data stream for training. Furthermore, it is also more robust against imbalanced distribution.

The chapters above describe the core techniques that enable us to build future intelligent system that learn new skills and concepts during deployment in an open world. Specifically, the open-worldness here admits training data with various non-ideal properties, such as limited labels (Ch. 3, 4, 6), incremental classes (Ch. 3, 6, 7), unlabeled data (Ch. 4, 6, 7), online streaming data (Ch. 6, 7), imbalanced data (Ch. 5, 7) and noisy labels and distractors (Ch. 4, 5). Towards the end, Chapter 8 concludes the thesis with an outlook to potential future research directions that extend beyond open-world machine learning.

1.2 Scope of the Thesis

Open-world machine learning is a broad topic by itself and it is useful to set up the scope of the thesis now. For most of the problem formulations discussed here, the input is often assumed to be a 2D image. However, almost all of the proposed algorithms, with the exception of the one from Chapter 7, can be easily adapted to non-visual inputs by simply changing the encoder network architecture for the target domain. The output of the model is typically assumed to be the class of the object in the image, represented by a categorical variable. Although this may be perceived as a major limitation, learning the concepts and categories is actually a core capability for high-level machine perception. Therefore, allowing machines to learn to perceive high-level concepts in an open world is a non-trivial effort. Success in this area could also potentially inspire future approaches that deal with more structured output spaces such as localization of objects and planning of motor movement by using a different decoder architecture, just like how standard deep neural networks that were originally designed for image classification have been successfully translated to other domains to deal with various input modalities and output spaces.

1.3 Publications Included in the Thesis

This thesis is composed of several conference papers of smaller scope:

- Chapter 3 (Ren et al., 2019): Ren, M., Liao, R., Fetaya, E., and Zemel, R. S. (2019). Incremental few-shot learning with attention attractor networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*

- Chapter 4 (Ren et al., 2018b): Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. (2018a). Meta-learning for semisupervised few-shot classification. In *6th International Conference on Learning Representations (ICLR)*

- Chapter 5 (Ren et al., 2018c): Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018b). Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

- Chapter 6 (Ren et al., 2021a): Ren, M., Iuzzolino, M. L., Mozer, M. C., and Zemel, R. S. (2021a). Wandering within a world: Online contextualized few-shot learning. In *9th International Conference on Learning Representations (ICLR)*.

- Chapter 7 (Ren et al., 2021b): Ren, M., Scott, T. R., Iuzzolino, M. L., Mozer, M. C., and Zemel, R. S. (2021b). Online unsupervised learning of visual representations and categories. *CoRR*, abs/2109.05675.

These chapters are composed of slight modifications of the papers above and have been reproduced here with the permission of the original authors.

1.4 Other Research During My PhD Study

The topic of open-world machine learning is the overture of a bigger dream of building a human-like, agent-based machine intelligence to continually learn, adapt, and reason in naturalistic environments. Throughout my PhD, I am grateful to have had the opportunity to work with many other collaborators on related topics aiming towards this more ambitious dream. In this section, I would like to mention a few different areas to highlight the effort we have made during my PhD study.

Few-shot and meta-learning: Besides the papers that I describe in this thesis, there are a few more works that fall within the area of few-shot learning and meta-learning. First of all, closely related to the problem introduced in Chapter 4, Wong et al. (2019) took another step forward and looked at directly recognizing unknown classes. SketchEmbedNet (Wang et al., 2021a) makes use of a sequential generative model to support few-shot learning. We also proposed a new few-shot learning task with ambiguous class definitions and found self-supervised learners significantly improve upon supervised baselines (Ren et al., 2020a). I have helped develop other self-supervised visual representation learning algorithms, such as LoCo (Xiong et al., 2020) and FlowE (Xiong et al., 2021).

Beyond few-shot learning, I have worked on meta-learning algorithms that can make general machine learning better and more efficient. Zhang et al. (2019) search for better neural architecture; Xiong et al. (2019) learn a meta-network that prevents forgetting during learning; and Wang et al. (2021c) efficiently optimize hyperparameters during training. Wu et al. (2018a); Lucas et al. (2021) also aim to understand the fundamental limitations of meta-learning.

Self-driving: As a part-time researcher at Uber ATG for the past four years, I got involved in many self-driving research projects, supervised by Prof. Raquel Urtasun. A self-driving car is an excellent example of an agent-based autonomy system deployed in the natural world, which is where my general research interest lies. First of all, I was involved in introducing machine learning components in the motion planning module of self-driving cars (Sadat et al., 2019, 2020) and this was made possible by imitating many human driving recordings. Secondly, I was interested in understanding the robustness against adversarial examples in the context of self-driving (Tu et al., 2020, 2021a,b; Wang et al., 2021b). Lastly, solving self-driving also requires understanding the coordination strategies of a group of agents, and we proposed new algorithms that can learn multi-agent communication for self-driving (Vadivelu et al., 2020; Sykora et al., 2020).

Brain- and cognitively inspired learning algorithms: One source of inspiration for building intelligent systems is the brain, and throughout my PhD, I also had the opportunity to work on learning algorithms that are brain- and cognitively inspired. The brain features the principle of *locality*: neurons send signals to their nearby neurons. We developed DivNorm (Ren et al., 2017), a mechanism that normalizes neural network activations in a local neighborhood. In LoCo (Xiong et al., 2020), we introduced local learning objectives, which enable distributed learning within a single neural network, just like biological neurons in our brain. Inspired by our brain's use of *visual attention* for efficient visual understanding, I have developed novel formulations of attention-based deep learning to 1) decompose complex visual scenes and form an object-centric representation (Ren and Zemel, 2017); 2) speed up computation using sparse convolution (Ren et al., 2018a); and 3) perform safer robotic motion planning (Wei et al., 2021).

Chapter 2

Background

In this chapter, we introduce the fundamental concepts as preliminaries for the subsequent chapters. We assume basic knowledge of neural networks and deep learning, and we focus on introducing various learning paradigms related to open-world machine learning, such as few-shot learning, continual learning, and self-supervised learning. We refer the readers to classic textbooks and survey papers on the building blocks of deep learning (LeCun et al., 2015; Goodfellow et al., 2016).

2.1 Classic Supervised Learning

To open up the dialogue of open-world machine learning, it may be useful to first lay down the setup of classic supervised machine learning. We denote $\mathbf{x}$ to be the input vector, and each input is associated with some label y. In the classification setting, y is a categorical variable $y \in [1, \ldots, K]$ indicating the class ID, where K is the total number of classes. Machine learning here typically means that we would like to learn a function f such that $f(\mathbf{x})$ well approximates y. To accomplish this task, the learning algorithm will need a dataset of examples, in this case, many pairs of $(\mathbf{x}, y)$ such that the function can be trained to jointly predict all the training labels. More concretely, in the classic framework of empirical risk minimization, we aim to seek for a function f parameterized by θ, such that the loss $\mathcal{L}$ over the training set $\{\mathbf{x}_i, y_i\}_{i=1}^N$ (empirical risk) is minimized:

$$\min_\theta \frac{1}{N} \sum_i \mathcal{L}(f(\mathbf{x}_i; \theta), y_i). \tag{2.1}$$

The seemingly simplistic supervised learning paradigm is actually powerful. When equipped with sufficient data points and model capacity, such as when f is a neural network with many hidden units, we can in fact model any learnable input-output relations by minimizing the empirical risk (Hornik et al., 1989). With the rise of deep learning in the past decade, classic supervised learning has revolutionized computer vision (Krizhevsky et al., 2012), machine translation (Sutskever et al., 2014), voice recognition (Graves et al., 2013), and many other fields. The classic supervised paradigm, however, relies heavily on a large amount of labeled data and focuses only on a fixed set of classes, which differs drastically from how humans acquire knowledge in the natural world. In this chapter, we focus on describing a few machine learning paradigms centered around the core problems of open-world machine learning.

First of all, humans have the remarkable ability to recognize new categories with a few examples, but such learning behavior is not well defined in classic supervised learning since y is always drawn from a fixed set of integers. Moreover, directly learning a neural network with few data points can result in poor generalization. Section 2.2 introduces *few-shot learning* that addresses the problem of learning new classes with a few examples.

Second, for classic supervised learning, $\mathbf{x}$ and y are typically assumed to be independently sampled from an identical distribution (the iid assumption). However, in the real world, when data is changing over time, a successful learning agent will have to constantly update its beliefs and incrementally build up its knowledge when new data come in. Section 2.3 describes *continual learning*, a learning paradigm that allows machines to incrementally learn a set of concepts or skills in a sequence and handle continuous distribution shifts.

Third, classic supervised learning relies on having class labels for each example, and collecting labels can be costly for most real-world applications. Furthermore, for learning sensorimotor tasks, humans and animals do not require many labels and most of the learning in these biological systems happened with only unlabeled data. These two reasons motivate us to study unsupervised learning, which are classic topics that aim to learn without labeled data. In Section 2.4, we focus on introducing one type of unsupervised learning that is called *self-supervised learning*, which has recently achieved superior performance on visual representation learning.

2.2 Few-Shot Learning

In the paradigm of few-shot learning (Li et al., 2007; Lake et al., 2011; Koch et al., 2015), models are asked to learn a new task with only a few labeled examples. In the recent episodic formulation of few-shot classification (Santoro et al., 2016; Vinyals et al., 2016), this means we would like the model to classify among a few novel classes that the model has never seen before, and "shot" here means the number of labeled examples. For example, "1-shot 5-way classification" means there is one example per class, and there are five novel classes to classify. Hence the task here contains five labeled examples in the training set for the model to learn, and we may use other test examples to evaluate the model's prediction accuracy.

How can a model rapidly learn a task with only a few examples, especially when the input is high dimensional and the model has tens of thousands of parameters? It almost sounds like a pipedream from the perspective of classic learning theory. In fact, many successful few-shot learning models leverage information from a large labeled training dataset to constrain the hypothesis space during test time. Consider a situation where we have a large labeled dataset for a set of classes $\mathcal{C}_{\text{train}}$. After training on examples from $\mathcal{C}_{\text{train}}$, our ultimate goal is to produce classifiers for a disjoint set of new classes $\mathcal{C}_{\text{test}}$, for which only a few labeled examples will be available. The idea behind the episodic paradigm is to simulate the types of few-shot problems happening at test time and to take advantage of the large amount of labeled data for $\mathcal{C}_{\text{train}}$.

Specifically, models are trained on K-shot, N-way episodes constructed by first sampling a small subset of N classes from $\mathcal{C}_{\text{train}}$ and then generating: 1) a training (support) set

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_{N\times K}, y_{N\times K})\} \tag{2.2}$$

Table 2.1: A summary of comparison between similarity-based few-shot classification methods.

Method	Comparison type	Similarity function
SiameseNet (Koch et al., 2015)	Pairwise	Neural network
MatchingNet (Vinyals et al., 2016)	Multi-way	Cosine
ProtoNet (Snell et al., 2017)	Multi-way	Negative squared Euclidean
RelationNet (Sung et al., 2018)	Multi-way	Neural network

containing K examples from each of the N classes and 2) a test (query) set

$$\mathcal{Q} = \{(\mathbf{x}_1^*, y_1^*), (\mathbf{x}_2^*, y_2^*), \ldots, (\mathbf{x}_T^*, y_T^*)\} \tag{2.3}$$

of different examples from the same N classes. Each $\mathbf{x}_i \in \mathbb{R}^D$ is an input vector of dimension D and $y_i \in \{1, 2, \ldots, N\}$ is a class label (similarly for $\mathbf{x}_i^*$ and y_i^*). Training on such episodes is done by feeding the support set $\mathcal{S}$ to the model and updating its parameters to minimize the loss of its predictions for the examples in the query set $\mathcal{Q}$.

One way to think of this approach is that it effectively trains models to be a good learning algorithm. Indeed, much like a learning algorithm, the model must take in a set of labeled examples and produce a predictor that can be applied to new examples. Moreover, training directly encourages the classifier produced by the model to have good generalization on the new examples of the query set. Due to this analogy, training under this paradigm is often referred to as *learning to learn* or meta-learning (Thrun, 1998).

On the other hand, referring to the content of episodes as training and test sets and to the process of learning on these episodes as meta-learning or meta-training (as is sometimes done in the literature) can be confusing. So for the sake of clarity, we will refer to the content of episodes as support and query sets, and to the process of iterating over the training episodes simply as training.

Common few-shot classification learning algorithms can be roughly categorized into several families: 1) Similarity learning approaches focus on learning a good similarity function that can tell whether two items are similar and belong to the same class; 2) recurrent networks learn parameters of an iterative recurrent process; 3) gradient-based meta-learning has a nested loop structure and uses gradient descent as an adaptation mechanism in the inner loop, while optimizing the meta-parameters in the outer loop; and 4) transfer learning approaches directly finetune the pretrained model towards the new test classes. We introduce each family of approaches in detail in the subsections below.

2.2.1 Similarity Learning

The process of few-shot classification has a close relation to the notion of similarity. If two images are really similar, then they are most likely of either the same instance or the same category. In fact, the human brain also extensively uses similarity to group objects into categories (Rosch and Mervis, 1975; Tversky, 1977), instead of a well-defined rule-based system. Traditionally, in the machine learning literature, similarity learning also relates to metric learning (Bellet et al., 2013), where a pairwise distance function is learned to measure how close two data points are. Below, we will introduce a few classic similarity learning approaches for few-shot classification.

Siamese Networks

Typical neural networks take a single image as an input, but a siamese network (Koch et al., 2015) instead takes two. The output of the network is a scalar that predicts whether the two inputs belong to the same class or not.

$$v = h(\mathbf{x}_1, \mathbf{x}_2; \theta), \tag{2.4}$$

where v is a real-valued scalar between 0 and 1. The siamese network first uses a shared encoder h_0 to process both images, and then concatenates the responses and passes the combined activation into another network h_1:

$$h(\mathbf{x}_1, \mathbf{x}_2) = h_1([h_0(\mathbf{x}_1), h_0(\mathbf{x}_2)]). \tag{2.5}$$

Such a network can be directly used for one-shot classification, by picking the maximum similarity to the image for each class. The potential drawback of this approach is that the training focuses on the pairwise similarity between two images, and it may be more efficient to look at more examples of the same class together. In addition, the concatenation operator is not commutative, and such inductive bias may be useful when comparing two inputs. These concerns will be addressed in the subsections below.

Matching Networks

Matching networks (Vinyals et al., 2016), also known as MatchingNets, extended siamese networks from one-shot learning to few-shot learning, i.e. to learn with a few examples per class. While both siamese networks and matching networks predict pairwise similarity between the two samples, the latter uses the idea of soft nearest neighbor matching among a set of examples during training. To predict the label of a query example $\mathbf{x}^*$ given a support set $\mathcal{S}$, it uses a form of soft attention over the support examples,

$$p(y^*|\mathbf{x}^*, \mathcal{S}) = \sum_{i=1}^{k} a(\mathbf{x}^*, \mathbf{x}_i) y_i, \tag{2.6}$$

where $a(\mathbf{x}^*, \mathbf{x}_i)$ is a normalized attention over the support examples, and is typically implemented as a softmax function over the cosine similarity between the supports and the query.

$$a(\mathbf{x}^*, \mathbf{x}_i) = \frac{\exp(c(h_s(\mathbf{x}^*), h_q(\mathbf{x}_i)))}{\sum_{i=1}^{T} \exp(c(h_s(\mathbf{x}^*), h_q(\mathbf{x}_i)))}, \tag{2.7}$$

where h_s and h_q are neural network encoders (potentially with $h_s = h_q$) to embed support and query examples, and c is a similarity function (e.g. cosine similarity). The entire network can be trained end-to-end using the cross-entropy loss by comparing to the query class labels. In the original paper, the author also proposed a full context embedding mechanism that introduces a recurrent network with attention in the encoder network to take account of the support set context. This is shown to have a small gain on benchmarks of natural images.

Matching networks are often thought to be the 1-nearest-neighbor classifier in the world of

few-shot learning. It is also related to neighborhood component analysis (NCA) (Goldberger et al., 2004), but NCA only aimed to learn a linear projection matrix for metric learning. In the human learning literature, there is also the exemplar theory (Medin and Schaffer, 1978), which predicts that categorization is done by matching the closest member in each category.

Prototypical Networks

Prototypical networks (Snell et al., 2017), or ProtoNets, have the virtue of simplicity and yet it often obtains impressive performance. Compared to matching networks, it simplifies the process of aggregating multiple examples from the same class. At a high level, it uses the support set $\mathcal{S}$ to extract a prototype vector from each class, and classifies the inputs in the query set based on their distance to the prototype of each class.

More precisely, prototypical networks learn an embedding function $h(\boldsymbol{x})$, parameterized as a neural network, that maps examples into a space where examples from the same class are close and those from different classes are far. Like matching networks, all parameters of prototypical networks lie in the embedding function. To compute the prototype $\boldsymbol{p}_c$ of each class c, a per-class average of the embedded examples is performed:

$$\mathbf{p}_c = \frac{\sum_i h(\mathbf{x}_i) z_{i,c}}{\sum_i z_{i,c}}, \quad \text{where} \quad z_{i,c} = \mathbb{1}[y_i = c]. \tag{2.8}$$

These prototypes define a predictor for the class of any new (query) example $\boldsymbol{x}^*$, which assigns a probability over any class c based on the distances between $\boldsymbol{x}^*$ and each prototype, as follows:

$$p(y^* = c|\mathbf{x}^*, \{\mathbf{p}_c\}) = \frac{\exp(-||h(\mathbf{x}^*) - \mathbf{p}_c||_2^2)}{\sum_{c'} \exp(-||h(\mathbf{x}^*) - \mathbf{p}_{c'}||_2^2)} \ . \tag{2.9}$$

The loss function used to update prototypical networks for a given training episode is then simply the average negative log probability of the correct class assignments, for all query examples:

$$-\frac{1}{T} \sum_i \log p(y^* = y_i^* | \boldsymbol{x}_i^*, \{\boldsymbol{p}_c\}) \ . \tag{2.10}$$

In fact, the idea of taking the mean of each class can be traced back in the early literature of machine learning, referred to as the nearest class mean classifier (Webb, 2003), and the distance-based cross-entropy objective was first described in a metric learning model proposed by Mensink et al. (2013). The difference is that the prototypical network is learned end-to-end in an episodic manner. Training of prototypical networks proceeds by minimizing the average loss, iterating over training episodes, and performing a gradient descent update for each. Empirically, ProtoNets are often found to perform better than MatchingNets. This suggests that averaging in the feature space is a more effective operator to rapidly generalize a new concept.

In the human concept learning literature, the prototypical network is similar to the prototype theory (Rosch and Mervis, 1975), which predicts that a category is represented by a prototype that is all of the common attributes shared by the members of a category. In contrast to the exemplar theory, prototype theory only needs to cache one prototype items per concept instead of all of the items. If each feature dimension is the log probability of the item containing a given attribute, then the arithmetic average computed by ProtoNets is approximating the product of the binary attributes.

The prototype theory is shown to support rapid learning due to its simplicity, but it loses information compared to the exemplar theory and therefore will have a hard time representing non-typical members such as a penguin being a member of birds (Murphy, 2004). In few-shot learning practices, prototypical networks often outperform matching networks, either with a pretrained embedding or an end-to-end learned one. This suggests that most of the few-shot learning experiments are indeed testing the rapid learning ability as opposed to trying to capture all the edge cases.

Relation Networks

Relation networks (Sung et al., 2018) further explore the design of similarity functions. In MatchingNets and ProtoNets, both cosine similarity and negative squared Euclidean distance functions are tested. They are simplistic, but they might lack expressivity in terms of modeling more complex relations. Towards this goal, relation networks leverage a learned network to compute the similarity function.

First of all, they compute the class representation by summing all the example features together:

$$\mathbf{x}_c = \sum_i h_\varphi(\mathbf{x}_i)\mathbb{1}[y_i = c]. \tag{2.11}$$

Then, they model the relations between examples and classes by using another network g:

$$r(\mathbf{x}^*, v_c) = g_\phi([h_\varphi(\mathbf{x}_c), h_\varphi(\mathbf{x}^*)]). \tag{2.12}$$

To train the networks h and g, their objective is a mean-squared error comparing r against the ground-truth one-hot relation vector. And to output a categorical output, one can simply choose the class with the strongest relation r. Empirical results show that relation networks outperform siamese networks, matching networks, and prototypical networks on few-shot classification benchmarks. Compared to prototypical networks, however, relation networks use more learnable parameters by using another dedicated relation network g_ϕ.

So far we have covered the similarity-based approaches. Table 2.1 provides a comparison of methods that we introduce in this subsection. All of them are designed for learning new classes by comparing them to similar items in the support set. Next, we introduce more general learners that can potentially be directed at learning other new tasks than few-shot classification.

2.2.2 Recurrent Network Learning

Many learning algorithms can be represented in the form of an iterative and recursive optimization process, and such a process can be further modeled with a recurrent neural network (RNN). The learnable parameters and hidden states in an RNN implicitly contain information about the learning process, the objective function, and the optimization steps. Such a flexible model has the potential benefit of scaling to many other tasks beyond few-shot classification, and it also resembles the human memory for implicitly storing recently learned patterns for recognition.

Santoro et al. (2016) proposed Memory Augmented Neural Networks (MANN) for one-shot classification. Instead of explicitly storing the examples or prototypes like in matching networks or prototypical networks, they featured a slot-based memory that can support reading and writing operations. When there is a new input, the memory network retrieves information by using soft

attention over all the stored slots. And after the reading operation is completed, it will use another set of attention mechanisms to write the hidden states into certain slots of the memory. Later, Graves et al. (2016) proposed an upgraded architecture called Neural Differentiable Computer (DNC), which is a more general-purpose memory-based recurrent neural network that can solve many reasoning tasks. The memory reading and writing operations introduce many degrees of freedom. These memory networks extensively use soft gatings to mimic algorithm-like discrete operations, but the gatings also introduce training difficulties, possibly due to gradient vanishing. Moreover, when benchmarked on few-shot learning, these generic memory-based RNNs do not perform as well as similarity-based approaches such as MatchingNets or ProtoNets.

Ravi and Larochelle (2017) proposed MetaLSTM that combines an RNN with an explicit optimization process. The update for the original LSTM network (Hochreiter and Schmidhuber, 1997) is:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{2.13}$$

where c_t is the LSTM cell state, f_t is the forget gate, i_t is the input gate, and $\tilde{c}_t$ is the input states. It is similar to the gradient descent algorithm if c_t represents the optimizee, f_t is 1, i_t is the learning rate, and $\tilde{c}_t$ is the gradient at the current step. In MetaLSTM, $\tilde{c}_t$ is indeed the gradient of the current loss on the support set, but i_t and f_t are learned functions and offers a lot more flexibility, and could potentially make the optimization process faster than standard gradient descent (Andrychowicz et al., 2016). Finally, the optimization process is unrolled just like an RNN, and all the parameters are learned to minimize the loss on the query set, just like other few-shot learning approaches.

Although MetaLSTM offers us a new perspective on using meta-learning to solve few-shot learning problems, and the recurrent process has a lot of flexibility, it is still challenging to apply such an algorithm, since the meta-network needs to scale with the complexity of the main network. To address this issue, Ravi and Larochelle (2017) proposed to share the LSTM parameters for each parameter coordinate and ignore the second-order gradients, making the number of meta-parameters to be constant. Empirical results on the mini-ImageNet benchmark (Vinyals et al., 2016) suggest that MetaLSTM is better compared to MatchingNets, but is worse than ProtoNets.

The general conclusion for RNN-based approaches is that they are very flexible models but the sheer complexity also prevents them from achieving high performance on few-shot classification. In the next section, we introduce another more constrained form of meta-learning that directly uses gradient descent for adaptation.

2.2.3 Gradient-Based Meta-Learning

Gradient descent is a universal optimization algorithm that is shown to be effective for optimizing deep neural networks. If we would like a network to solve a particular task, we can directly run gradient-based optimization on the network. But if we simply start with a random weight initialization, gradient descent will need hundreds of iterations and may still find solutions that fail to generalize to the query set. To this end, Finn et al. (2017) proposed Model-Agnostic Meta-Learning (MAML) to search for a good weight initialization that can lead to generalizable solutions in a few iterations. The algorithm features a nested loop structure. In the inner loop, the network receives task $\mathcal{T}_i$ and

performs a few gradient descent steps to solve the task:

$$\theta_i = \theta - \alpha \nabla_\theta \mathcal{L}_i(\mathcal{S}_i; \theta), \tag{2.14}$$

where $\mathcal{L}_i$ is the loss function of the task, and α is the inner learning rate. The reason that the inner loop is only a few gradient steps instead of solving until convergence is that it encourages the network to learn to quickly adapt to each task. In the outer loop, the network initialization parameters are learned:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_i(\mathcal{Q}_i; \theta_i), \tag{2.15}$$

where β is the outer learning rate, and θ_i represents the learned parameters through this nested loop learning process. The idea is to learn a universal initialization such that it can reach good performance for any tasks within a few gradient descent steps. Just like recurrent neural networks, MAML can be a universal approximator (Finn and Levine, 2018).

One caveat of MAML is that it needs to store a copy of the weights in each inner loop iteration, and this prevents efficient batching of different tasks since it needs to perform gradient descent for each task. Later development of gradient-based meta-learning found that it is not required to update the entire network to achieve sufficient adaptation. Rusu et al. (2019) proposed Latent Embedding Optimization (LEO), which instead optimizes a network embedding vector that can be decoded into the full parameters, but the gradients can be batched together. It can be thought of as a low dimensional projection of the full parameters into a smaller vector space. Zintgraf et al. (2019) proposed fast context adaptation which allocates an additional hidden state vector that is initialized from zero and adapts with the task. Raghu et al. (2020) found that only performing gradient descent on the last layer is often sufficient, suggesting that the success of MAML is mostly due to the effect of feature reusing, where the network can learn some general features from the training set, rather than rapid adaptation.

The findings of Raghu et al. (2020) invite the question of how well can feature reusing actually solve the problem of few-shot classification. Feature reusing is similar to the traditional way of doing transfer learning, where a model is first trained to recognize a set of training classes, and then finetuned to recognize the test classes. This question will soon be answered in the next section.

2.2.4 Transfer Learning Approaches

In standard transfer learning (Pan and Yang, 2010), a model is typically trained on task A, which has many labeled data, and then transferred to task B, which has fewer data. A deep neural network can reuse the output layer or reinitialize the output layer for the new task if it has a different output space. Additionally, we can finetune the network for more effective adaptation. There is no meta-learning required since you only need to do the finetuning once for each new task, and unlike MAML, the model does not need to iterate over nested loops during training.

In fact, such a transfer learning setup share a lot of similarities with few-shot learning: First, the model is trained on a source dataset with many training classes, and then it is transferred to a new target dataset to recognize a few new classes with limited labeled data. In the beginning, people were suspicious of whether such a simple approach would even succeed in few-shot learning, since the

number of data points for the new task is so scarce than any attempt of finetuning could lead to overfitting, and this was indeed the conclusion from the early literature on few-shot learning (Santoro et al., 2016; Vinyals et al., 2016); however, the authors of these papers did not try state-of-the-art deep networks such as ResNets (He et al., 2016). Moreover, most of the attention was on the Omniglot dataset, where standard deep classifiers are not able to learn very well due to the very large number of classes. A few years later, when people started to target the more challenging few-shot learning benchmarks made up of natural images with much fewer classes, Chen et al. (2019) found that a standard transfer learning type of approach, referred to as the "Baseline" approach, can surprisingly work well. The performance of "Baseline" is similar to or sometimes even better than that of state-of-the-art meta-learning or few-shot learning approaches.

There are two reasons why the simple transfer learning approach is favored in their experiments. First of all, in the benchmarks experimented by Chen et al. (2019), the number of classes is fewer and the number of images per class is more, compared to the original benchmarks tested by Lake et al. (2011); Santoro et al. (2016). In natural image datasets, each class contains over 500 images, whereas in the Omniglot dataset, each class only contains around 20 images. Note that this is not to be confused with the "K-shot N-way" episodes, but the overall dataset composition. Because of the changes in dataset composition, we are again hitting the sweet spot of classic supervised learning, and the conclusion here is that classic supervised learning can learn generalizable and transferable representations that support few-shot learning if the training set is fully labeled with a large number of images per class.

Second, Chen et al. (2019) also found out that the performance gets better when they upgrade the simple CNN with deeper and wider residual networks (He et al., 2016). This suggests that better network designs seem to already deliver good performance for few-shot learning out of the box, with only slight finetuning needed at test time. Similarly, Gidaris and Komodakis (2018) also found the benefit of standard pretraining of regular classification tasks, instead of using meta-learning.

The experiments of Chen et al. (2019) do provide a lot of insights for few-shot learning and since then the state-of-the-art few-shot approaches all use the suggested pretraining as an initial stage (Tian et al., 2020). The results also hint that the heavy-lifting in few-shot learning can be achieved through having richer and more informative representations. While Chen et al. (2019) emphasize the importance of representation learning as an initial step, it does not mean that all meta-learning approaches are now obsolete. First of all, meta-learning is still relevant when there is a vast number of tasks, but each task contains very few data points. Chen et al. (2019) were only able to show good performance on natural image benchmarks, where the image-class ratio is at the sweet spot for pretrained classifiers. Secondly, the baseline approach also relies on sampling iid from a large labeled classification dataset, whereas meta-learning approaches seem to be closer to the practical usage scenario of an online learning agent in the real world since it only needs to deal with one new task at a time. Nevertheless, standard meta-learning benchmarks have not yet developed a strong link to the learning process of an online agent, nor have them specifically addressed the online non-iid problem. The question of how to combine meta-learning and online continual learning will be further investigated in the later chapters of this thesis.

2.2.5 Other Few-Shot Learning Approaches

In addition to the types of few-shot learning approaches introduced above, there are other types of few-shot learners. Garcia and Bruna (2018) proposed to use graph neural networks for few-shot learning, where each node is an example in an episode, and the edges represent the affinity relations between examples, just like RelationNet. The reasoning process is modeled as graph propagation, and it is also suitable for transductive reasoning where the query examples are jointly classified.

A lot of the attention is focused on few-shot classification, but sometimes regression is also an interesting problem. While similarity-based approaches can only handle classification, gradient- and recurrent-based approaches can handle different output types. Conditional neural processes (Garnelo et al., 2018) is another type of general few-shot learner that can work with different output spaces and is particularly suitable for regression problems. It passes all the inputs through an encoder and sums them across to get the context encoding. Then the query inputs are conditioned with the context encoding to produce a context-dependent prediction. Experiments show that one can predict the missing data by providing the model with partial inputs, and the model can learn to interpolate. It also performs better than standard k-nearest neighbors and Gaussian processes, because of the learned representations.

Related to neural processes, Grant et al. (2018) proposed to cast gradient-based meta-learning as a form of hierarchical Bayesian learning, where the support set allows the meta-learners to infer task-specific parameters and the meta-learning process equates to the learning of the prior distribution of the parameters. The shortened gradient adaptation steps can be viewed as having a regularized least square problem and the prior term constrains the amount of deviation. Later, Rajeswaran et al. (2019) proposed to explicitly constrain MAML with a squared error regularizer, and instead of using a truncated number of gradient steps, it rolls out the optimization till convergence.

2.2.6 Few-Shot Learning Datasets

In this section, we introduce a few commonly used few-shoy learning datasets that will be later referred to in the thesis. All of the datasets here are image datasets and the main goal is to perform few-shot image classification.

Omniglot

Omniglot (Lake et al., 2011) is a dataset of 1,623 handwritten characters from 50 alphabets. Each character was drawn by 20 human subjects. We follow the few-shot setting proposed by Vinyals et al. (2016), in which the images are resized to 28×28 pixels and rotations in multiples of 90° are applied, yielding 6,492 classes in total. These are split into 4,112 training classes, 688 validation classes, and 1,692 testing classes.

mini-ImageNet

mini-ImageNet (Vinyals et al., 2016) is a modified version of the ILSVRC-12 dataset (Russakovsky et al., 2015), in which 600 images for each of 100 classes were randomly chosen to be part of the dataset. We rely on the class split used by Ravi and Larochelle (2017). These splits use 64 classes for training, 16 for validation, and 20 for testing. All images are of size 84×84 pixels.

tiered-ImageNet

Like *mini*-Imagenet, *tiered*-ImageNet (Ren et al., 2018b) is a subset of ILSVRC-12. However, it represents a larger subset of ILSVRC-12 (608 classes rather than 100 for *mini*-ImageNet). Analogous to Omniglot, in which characters are grouped into alphabets, *tiered*-ImageNet groups classes into broader categories corresponding to higher-level nodes in the ImageNet (Deng et al., 2009) hierarchy. There are 34 categories in total, with each category containing between 10 and 30 classes. These are split into 20 training, 6 validation, and 8 testing categories. This ensures that all of the training classes are sufficiently distinct from the testing classes, unlike *mini*-ImageNet and other alternatives such as *rand*-ImageNet proposed by Vinyals et al. (2016). For example, "pipe organ" is a training class, and "electric guitar" is a test class in the Ravi and Larochelle (2017) split of *mini*-Imagenet, even though they are both musical instruments. This scenario would not occur in *tiered*-ImageNet since "musical instrument" is a high-level category and as such is not split between training and test classes. This represents a more realistic few-shot learning scenario since in general, we cannot assume that test classes will be similar to those seen in training. Additionally, the tiered structure of *tiered*-ImageNet may be useful for few-shot learning approaches that can take advantage of hierarchical relationships between classes.

Other Few-Shot Learning Datasets

The benchmarks above, however, have not specifically considered domain shift between training and test episodes, since the datasets are homogeneously composed of either handwritten characters or natural images with a single object in the center, and one can argue that there might not be much need to perform test time adaptation except learning the classification layer. Guo et al. (2020) proposed the Cross-Domain Few-Shot Learning benchmark which contains one source domain to be *mini*-ImageNet and four different target domains: 1) CropDisease contains plant disease images; 2) EuroSAT contains satellite images of different objects; 3) ISIC contains colored medical images and 4) ChestX contains grayscale medical images. They claim that there is decreasing similarity to ImageNet from 1) to 4).

Since Guo et al. (2020) only use one source domain, it might be challenging for meta-learning methods to learn to generalize across domains. To address the meta-learning question, Triantafillou et al. (2020) proposed MetaDataset, which is a dataset of datasets. It contains 10 different datasets with domains range from general class images, class-specific images, handwriting images, and textures. These later developments of few-shot learning benchmarks represent an effort towards creating the need for test-time adaptation and meta-learning due to the added domain shifts.

Few-shot learning episodes will be much more realistic if they are embedded in an online continual learning process, rather than taken out the episodes sampled iid from a large pool of images. Antoniou et al. (2020) studied how to define a continual few-shot learning benchmark to make models more practical. It uses a downsampled version of the ImageNet dataset and each episode now becomes a sequence of several smaller classification problems. The classes can either be staying the same, growing over time, or getting redefined later. Models are then evaluated at the end of the sequence, to see whether they can learn these new concepts in a sequence. In our next section, we will review the related background in continual learning to allow us to further understand how to combine these two learning paradigms in our later chapters.

2.3 Continual Learning

The motivation of few-shot learning is closely related to that of continual learning (Parisi et al., 2019). We encounter new things with a few examples because we experience a lifelong sequential stream of data that keeps introducing new classes and new tasks to us. Unlike standard training of image classification tasks, where the machine learning model has the privilege of uniformly sampling examples and tasks to form a large batch of data, humans only get to experience one thing at a time: our visual glimpses represent a narrow visual range over a temporally correlated video stream, and our motor functions usually deal with one task simultaneously, either running, swimming, or skiing, but hardly never a combination of these.

One of the main difficulties of continual learning is the effect of catastrophic forgetting (McCloskey and Cohen, 1989; French, 1999), especially for models like deep neural networks. Essentially, the knowledge of older tasks is not being retained in the weight connections as the network is not experiencing inputs from these tasks again. On one hand, catrastrophic prevents us to learn efficiently in a world where observations are all sequential, but on the other hand, some level of forgetting makes sure that the capacity of the model is dedicated to the most recent events.

To address this challenge of catastrophic forgetting, there have been different forms of continual learning experiment setup (van de Ven and Tolias, 2018). The easiest type is continual distribution shifts, such as by using images with various rotation angles. People have also used classification problems to simulate different tasks. For example, in the first task the neural network needs to differentiate between images of "0" vs. "1", and the second "2" vs. "3", and so on. Each task is a binary classification problem, and in the end, you would expect to train a classifier that distinguishes even vs. odd digits. Sometimes, the task identifier is not given, and this can make it more challenging as the model probably needs to identify the task boundary if that is necessary, and this is referred to as the *task-agnostic* setting. Another type of continual learning is incremental class learning. Since the tasks are often based on image classification, therefore a more interesting or perhaps useful thing to do is to let the model recognize different image classes incrementally. For example, a dataset of a total of 100 classes will be equally divided into 10 tasks, each with 10 classes. And the model will be trained on a small subset of 10 classes at any given time, and at the end of each task, it will be evaluated on the combination of all the classes that have appeared in the past.

In the following subsections, we introduce several common types of methods designed for the continual learning challenge. 1) Regularization-based methods constrain the learning in the later stage to prevent it from deviating too much from earlier solutions; 2) model compression approaches try to allocate different parts of the model for solving different tasks and thus reducing the interference between tasks; and 3) rehearsal-based methods try to store some old tasks data and replay it with the new inputs to prevent forgetting. Lastly, we link back to few-shot learning by introducing a few recently proposed continual few-shot learning approaches.

2.3.1 Regularization-Based Learning

Regularization-based continual learning approaches typically have an additional regularizer introduced in the objective function to constrain the optimization of the current task. Evgeniou and Pontil (2004) proposed to add an L2 regularizer centered at the solution of the previous task. Elastic weight consolidation (EWC) (Kirkpatrick et al., 2017) has a different strength at each weight dimension

computed by the diagonal Fisher information matrix.

$$\mathcal{L}(\theta) = \mathcal{L}_n(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{n-1,i})^2, \tag{2.16}$$

where $\mathcal{L}_n$ is the current task loss, and θ^*_{n-1} is the final weight parameters for the previous task, and F_i is the i-th diagonal entry of the Fisher matrix. Synaptic Intelligence (SI) (Zenke et al., 2017) approximates the Fisher matrix online, instead of through batch processing at the end of the task.

$$\mathcal{L}(\theta) = \mathcal{L}_n(\theta) + c\sum_i \Omega^n_i(\theta_i - \theta^*_{n-1,i})^2, \tag{2.17}$$

$$\Omega^n_i = \sum_{m<n} \frac{\omega^m_i}{(\Delta^m_i)^2 + \xi}, \tag{2.18}$$

where Ω^n_i is the estimated regularization coefficient, and ω^n_i can be estimated as a running sum of the product of the gradients $\frac{\partial \mathcal{L}_n}{\partial \theta_i}$ and the actual update $\Delta\theta_i$:

$$\omega^n_i = \int_t \frac{\partial \mathcal{L}_n}{\partial \theta_i(t)} \frac{\partial \theta_i(t)}{\partial t} dt \tag{2.19}$$

$$\approx \sum_j \frac{\partial \mathcal{L}_n}{\partial \theta_i(j)} \Delta\theta_i(j). \tag{2.20}$$

Incremental Moment Matching (IMM) (Lee et al., 2017) aims to match the moments of posterior distributions for different tasks. Learning without Forgetting (LwF) (Li and Hoiem, 2018) uses the previous network's prediction to distill the prediction of the inputs on the new task. In variational continual learning (VCL) (Nguyen et al., 2018), the learned weights can be interpreted as a variational approximation to the model posterior distribution, and can also be later used as a prior distribution to regularize the next task. It is worth noting that although regularization-based methods typically do not store any raw data points, they do require the knowledge of task boundary so that they can checkpoint the weights from the previous task to use it to regularize the learning later. Hence, they cannot be directly ported to the task-agnostic setting.

2.3.2 Rehearsal-Based Learning

Rather than explicitly constraining the learning objective, rehearsal-based methods (Ratcliff, 1990) instead store old data to mitigate the effect of forgetting. Some replay mechanisms are designed for incremental class learning, while others are more general. For incremental class learning, many algorithms (Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2019; Wu et al., 2019) employs the "herding" mechanism (Welling, 2009) to select the most representative exemplars for each class, although Wu et al. (2019) found that random class-based sampling can be almost as strong. Hayes et al. (2019) proposed to perform online clustering for each class of the examples and replay based on the clusters. Hayes et al. (2020) later proposed to perform quantization in the latent space, and replay these hidden activations to train the top "plastic" layers only. This strategy is found to be the most memory-efficient and is also brain-inspired.

There are also more general types of rehearsal mechanisms that do not rely on the notion of a "class." Gradient episodic memory (GEM) (Lopez-Paz and Ranzato, 2017) stores the gradient update

directions and encourages new updates to align with older update directions. A-GEM (Chaudhry et al., 2019a) simplifies the constraint satisfaction by averaging the historical gradients, making the learning algorithm task-agnostic, and Aljundi et al. (2019) instead proposed to use gradient directions to pick more diverse samples. Another line of work uses reservoir sampling (Vitter, 1985), an online mechanism that decides the probability of retaining a sample in the buffer to simulate uniform sampling. Every time step it will generate a random integer j between 0 and the total number of items presented, and if j is smaller than the buffer size, then the item is chosen to be stored in the buffer at location j. Riemer et al. (2019) proposed Meta Experience Replay (MER), which combines a MAML-like (Finn et al., 2017) algorithm with reservoir sampling, where it updates the parameters in the outer loop after several gradient descent steps in each inner loop, to mitigate rapid changes. Buzzega et al. (2020) proposed Dark Experience Replay (DER) that leverages reservoir sampling but uses old prediction to distill the output, instead of directly storing the labels.

Lastly, as rehearsal methods are often criticized for storing the raw bits of input data, which is less brain-like, the generative replay mechanism instead uses a generative model to replay old data, instead of sampling from the data storage. Shin et al. (2017) explored the use of a generative adversarial network to mimic the cumulative distribution of all past data. FearNet (Kemker and Kanan, 2018) is another generative framework for continual learning which consists of three components: 1) a hippocampal complex (HC) network for storing recent memories; 2) a medial prefrontal cortex (mPFC) network for generating past data through a generative model; and 3) a basolateral amygdala network that acts as a controller to select between HC and mPFC networks. This setup nicely integrates short-term and long-term memory, and the encoder portion of the generative model is being used for image classification tasks. Lastly, van de Ven et al. (2020) proposed to have the lower network fixed and train the upper network through generative replay from the intermediate layer. They argued that although generative models at the image level show decent results, it is much harder to guarantee success in the task-agnostic setting, where the learner is oblivious of the task boundary, or in settings where the inputs are natural images. The authors made an analogy to the brain: The higher-level layers can be thought of as the hippocampus and lower-level layers the visual cortex, since the hippocampus is often thought to be at a higher level in the brain's processing hierarchy, and the visual cortex is often less plastic in adulthood. The authors also made the latent variables a Gaussian mixture conditioned on the classes, and added another gating mechanism that is conditioned on an internal context that allocates hidden neurons for each task. Overall, van de Ven et al. (2020) offered a comprehensive and biologically inspired solution for generative replay in a task-agnostic setting.

Nevertheless, it should be noted that these generative models also introduce data storage complexity since now the data is simply being stored in the weight connections instead of raw bits. Furthermore, generative models may also suffer from a much greater computational burden compared to rehearsal-based models. Lastly, it remains to be further understood 1) whether generative models also fundamentally suffer from catastrophic forgetting themselves, even when we replay the data that they learn to generate, and 2) what is the mixing rate for the generated data vs. current inputs when lacking the concept of "classes".

2.3.3 Model Compression

We introduce another type of continual learning approach that is based on model compression. The high-level idea is that only a part of the network is allocated for each task, and overall the effect of catastrophic forgetting can be mitigated since neurons are less likely to be overwritten once it is committed to a certain task.

PathNet (Fernando et al., 2017) allocates a unique pathway that connects different modules for each task, and the pathway is found through evolutionary search. Modules can be reused depending on whether the tasks are similar enough. Rajasegaran et al. (2019) later found that using a random selection strategy can also deliver good performance when combined with skip connections and residual connections. PackNet (Mallya and Lazebnik, 2018) learns such a computational mask through iterative pruning of connection weights. Weights that are not pruned are going to be kept fixed for the next task. The drawback is that the total capacity of the network is fixed and it cannot fit more tasks if there are more, since eventually all units will be recruited for a task and will stay frozen forever. Dynamically Expandable Networks (Yoon et al., 2018) allows the network to selectively retrain some of the units and to add additional units to the network if it fails to reach a certain loss level. Serrà et al. (2018) proposed to compute hard attention to neurons conditioned on a task embedding, and the unit will eventually saturate and will not be updating as much as other unused units that have not been assigned. Different from PathNet, here the pathways are allocated on the unit level, rather than on the module level. And instead of using evolutionary search, in this work, the pathways are controlled by the task embedding, which can be end-to-end learned through gradient descent.

It is worth noting that, according to a summary by Buzzega et al. (2020), these model compression approaches all require some form of knowledge of task boundary, during both training and testing, in order to know when to perform their predefined set of operations, either pathway search, pruning, or expansion. Moreover, they often freeze the learned units to prevent forgetting, and hence they do not promise a constant memory usage with regard to the number of tasks.

2.4 Self-Supervised Learning

In previous sections, we covered some aspects of open-world learning: few-shot learning uses less labeled data and continual learning allows the label space to change or grow. Both paradigms, however, rely on having labeled data to drive the learning process. In the real world, there is a vast amount of unlabeled data, and obtaining labels is usually a labor-intensive process. Humans and animals can learn most of their visual representations by watching and interacting with the world. One can argue that there is much more information in the inputs themselves, rather than in the class label space.

Traditionally, the literature on unsupervised learning studies the task of learning with unlabeled data. General paradigms are typically projecting the high dimensional data into lower dimensional manifold, potentially using a generative model that can capture the input data distribution. In the pre-deep learning era, common techniques include dimensionality reduction (Cunningham and Ghahramani, 2015), clustering (Lloyd, 1982), and probabilistic graphical models (Koller and Friedman, 2009). With the rise of deep learning in the past decade, a variety of deep generative models were proposed, including variational autoencoders (VAEs) (Kingma and Welling, 2014), generative adversarial

networks (GANs) (Goodfellow et al., 2014), and autoregressive models such as PixelCNNs (van den Oord et al., 2016). One of the major difficulties for these generative modeling-based approaches is the learning of the generative decoder, rather than the intermediate representations.

In the past few years, another line of work aims not to generate high dimensional data, but to learn meaningful representations in the latent space directly, through a general technique called "self-supervision." The idea is that instead of supervising the model with all the information from the input data, self-supervision asks the model to recover some parts of the inputs, and sometimes it is being hidden from the original inputs to make the task more challenging. Compared to generative modeling, the benefit of such an approach is that it no longer needs to deal with the challenging data generation problem, but instead, it becomes a more familiar style as standard supervised learning, except that the labels are not annotated by humans but directly coming from the input data.

Since the rise of deep learning, self-supervised learning has been a popular approach for modeling natural languages. Mikolov et al. (2013) proposed the skip-gram model that predicts the missing word given a window of words. Kiros et al. (2015) proposed the skip-thought model that tries to find the missing sentence given the context. Devlin et al. (2019) combined the task of masked word prediction and sentence prediction and has since become the standard model for natural language pretraining. Recently, self-supervised learning has gained a lot of attention in computer vision as well, as it can now be as good as classic supervised learning in image classification tasks. In this section, we introduce several common families of self-supervised learning of visual representations.

2.4.1 Pretext Tasks

Originally, self-supervised learning started with a general paradigm that the trainer hides a part of the inputs and then supervised the model to recover the missing part (Goyal et al., 2019). The task of recovering missing parts is referred to as the "pretext task," and there are various existing pretext tasks that are found to be effective. Doersch et al. (2015) proposed to ask the network to predict the spatial relations between two given image patches, and Noroozi and Favaro (2016) proposed to let the network solve a 3×3 puzzle given 9 randomly sampled image patches. These two tasks help the network to gain knowledge of part-whole relationships; however, as we shall see, many of the pretext tasks, including these two, rely on images with a single object centered in the middle in order to sample interesting tasks for the network to solve. Noroozi et al. (2017) proposed to ask the network to generate a counting variable so that the summation of each part is consistent with the whole image. Zhang et al. (2016) proposed to send in a grayscale image and ask the network to predict the original color. Gidaris et al. (2018) proposed to randomly rotate an input image to different angles and then ask the network to predict the transformation angle. These tasks all help the network learn general object concepts in order to successfully predict the orientation or the original color.

To evaluate the effectiveness of these self-supervised methods, people typically follow a linear readout protocol, where the main representation network is frozen, but a linear layer is then learned to classify the images using the entire labeled training set. Due to the restricted capacity of a single linear layer, better representations usually translate to higher classification scores. Alternatively, one can also use a k-nearest-neighbor classifier. The methods mentioned here showed increasing accuracies on linear readout; however, self-supervised learning still hasn't received much attention until the introduction of instance-based supervision, which will be covered next.

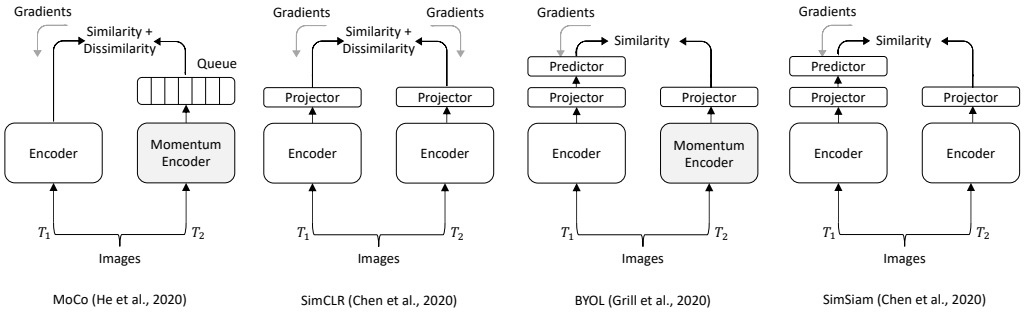


Figure 2.1: Several instance-based self-supervised representation learning algorithms. Figure style by Chen and He (2020). T_i represents random data augmentation transformation.

2.4.2 Instance-Based Objectives

Although the pretext tasks are generally interesting, the performance of the learned representations still lags supervised by a lot. It is later found out that a much simpler task can actually deliver quite amazing results. The task is to perform instance classification, i.e. to treat each image in the dataset with a different class label. Wu et al. (2018b) proposed a non-parametric classification layer that uses the image features from the previous epoch as the weights and the objective is to classify the image with their instance label using the standard cross-entropy loss, which is also referred to as the InfoNCE loss (van den Oord et al., 2018). Given an image query q and a list of images with different keys k, the loss function looks like the multi-class cross-entropy loss:

$$\mathcal{L}_{q,k^+,k^-} = -\log \frac{\exp(q \cdot k^+/\tau)}{\exp(q \cdot k^+/\tau) + \sum_{k^-} \exp(q \cdot k^-/\tau)}, \tag{2.21}$$

where k^+ is the positive match, and k^-'s are the negative matches. A similar loss also appeared before in the metric learning literature for learning with more negative samples (Sohn, 2016). van den Oord et al. (2018) found that generating two different cropped views of an image for keys and queries can make the self-supervision more challenging and as a result, it can improve the learned representations.

After Wu et al. (2018b) has found that the instance classification objective can achieve better performance compared to other self-supervised objectives we mentioned earlier, a series of important upgrades eventually led to matching performance compared to standard supervised learning. The original instance classification layer can be out-of-sync with the ever-changing network weights. To address this problem, in MoCo (He et al., 2020), the non-parametric classification layer is replaced by features from a "momentum encoder." Every iteration the momentum encoder will push the latest image features into the queue, while its weights will be updated with the main encoder by using an exponential moving average mechanism. SimCLR (Chen et al., 2020a) got rid of the momentum encoder and instead rely on a very large batch of images at each iteration. It is also equipped with a set of a stronger form of data augmentation including color jittering, and the inclusion of additional projection layers before the loss layer. The set of data augmentations, including color jittering, random cropping and random blurring, create a set of transformations that the model should be invariant of, and since the semantic content is not changed after the transformations, they

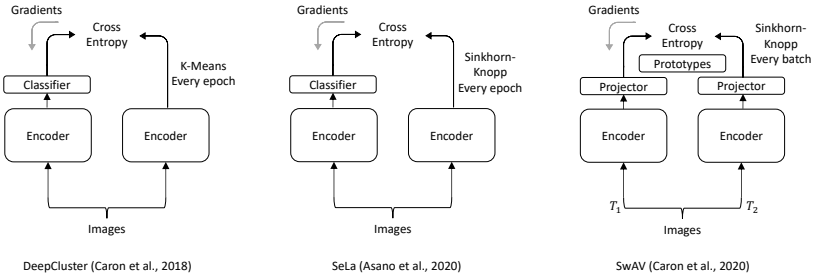


Figure 2.2: Several clustering-based self-supervised learning algorithms.

can provide additional model supervision that are not based on class labels. BYOL (Grill et al., 2020) and SimSiam (Chen and He, 2020) further reduce the reliance on large batch and negative samples by using a stop gradient mechanism to break the symmetry and avoid trivial solutions. By now, instance-based classification has become the go-to method for self-supervised representation learning, and it can achieve as good performance as supervised learning by using the linear readout protocol. Figure 2.1 summarizes the above instance-based self-supervised learning algorithms.

2.4.3 Clustering-Based Objectives

Instance-based objectives are based on an assumption that images are fairly different from one another and it would be possible to tell the image ID based on only a random crop. In other words, they ignore the cross-sample similarity by treating everything as a different instance. A natural question arises: can we group a few similar instances together and automatically form the notion of a class? In this section, we introduce an effort towards this direction using the idea of clustering. Clustering is a classic topic in machine learning (Lloyd, 1982) that aims to explain the data points by forming groups and clusters, and we shall see next how to integrate this classic idea with a powerful deep neural network.

DeepCluster (Caron et al., 2018) first pioneered the idea of using the cluster indices as pseudo labels to train a deep neural network. The idea is straightforward. We start with a random network. First, the k-means clustering procedure solves the following problem:

$$\min_{C\in\mathbb{R}^{d\times k}} \frac{1}{N}\sum_{n=1}^{N} \min_{y_n\in[0,1]^k} \|f(\mathbf{x}_n;\theta) - y_n C\|_2^2, \quad \text{s.t. } y_n^\top \mathbb{1}_k = 1, \tag{2.22}$$

where f is the encoder network, y is the cluster assignment, and C represents the cluster centroids. DeepCluster uses the k-means algorithm for solving y and C, by alterating the optimization of each variable. After the clustering procedure ended, it uses the pseudo labels y to train the encoder network parameters θ:

$$\min_{\theta} \frac{1}{N}\sum_{n}\sum_{k} -y_{n,k}\log(\mathbf{w}^\top f(\mathbf{x}_n;\theta) + b). \tag{2.23}$$

Caron et al. (2018) also mentioned that they needed to pay special attention to trivial solutions such as all examples collapsed to one big cluster, and potentially having empty clusters. First, they reinitialized empty clusters with perturbed centroids of non-empty clusters. Second, they resample

the images based on cluster indices to avoid extreme imbalance.

Since these hot fixes are mainly due to the erratic behavior of the k-means algorithm, Asano et al. (2020); Caron et al. (2020) proposed to use the Sinkhorn-Knopp algorithm (Cuturi, 2013) to perform clustering. We introduce this technique here following the formulation by Caron et al. (2020), which directly performs clustering online with the current mini-batch of examples. The assignment matrix Q solves the following optimization problem:

$$\max_{Q \in \mathcal{Q}} \operatorname{Tr}(Q^\top C^\top Z) + \epsilon H(Q), \tag{2.24}$$

where Z is the neural network features, C are the cluster centroids, $H(\cdot)$ is entropy function, and $\mathcal{Q}$ is the transportation polytope of the minibatch of size B:

$$\mathcal{Q} = \left\{ Q \in \mathbb{R}_+^{K \times B} | Q \mathbb{1}_B = \frac{1}{K} \mathbb{1}_B, Q^\top \mathbb{1}_K = \frac{1}{B} \mathbb{1}_K \right\}. \tag{2.25}$$

The solution Q^* can be computed as:

$$Q^* = \operatorname{diag}(\mathbf{u}) \exp(C^\top Z/\epsilon) \operatorname{diag}(\mathbf{v}), \tag{2.26}$$

where the renormalization vector $\mathbf{u}$ and $\mathbf{v}$ can be computed using the iterative Sinkhorn-Knopp algorithm.

Lastly, similar to other self-supervised methods, SwAV uses the cluster assignment of one view to supervise another view. Here, a view means one augmented version of the original image by performing random cropping and random color jittering. By performing this extra inference of clusters, SwAV is found to be consistently better compared to SimCLR. Figure 2.2 summarizes the above clustering-based self-supervised representation learning algorithms.

Although Sinkhorn-Knopp is much more robust than k-means by having the equal assignment as a constraint, it again assumes that the data follows a balanced distribution with each class having roughly the same number of elements. This could be easily violated with a long-tailed distribution, and we will revisit this assumption later in the chapters.

2.4.4 Self-Supervised Few-Shot Learning

Lastly, we will connect this section of self-supervised learning back to our first section of few-shot learning, and introduce some recent development on the topic of self-supervised few-shot learning. Standard few-shot learning is driven by labeled examples from a large labeled training set, and it would have been much better if it could also use less labeled data during training as well. Self-supervised learning on the other hand offers the promise of learning without using any labeled data, so there could be a natural combination between the two, especially if the few-shot learning paradigm can make self-supervised learning less reliant on training with a large batch, since the few-shot episodes are typically much smaller.

Huang et al. (2019) found that at test time, there is no need to provide the support set labels, and performing unsupervised clustering on the support set can provide similar performance compared to the standard few-shot learning setup. Other methods aimed to get rid of class labels during training. Hsu et al. (2019) proposed to perform clustering in a randomly drawn episode, and generate pseudo-

labels for the meta-learners, such as MAML, to learn from. This mechanism is similar to DeepCluster (Caron et al., 2018), but the learning is much noisier due to the small batch size and therefore it has to rely on some form of pretraining. Antoniou and Storkey (2019); Khodadadeh et al. (2019) augments the randomly drawn images to "create" the query set, similar to instance-based self-supervision Chen et al. (2020a). (Gidaris et al., 2019) found that using rotation prediction as an auxiliary task can also improve the performance of few-shot learning. Medina et al. (2020) proposed ProtoCLR, a *portmanteau* of ProtoNet Snell et al. (2017) and SimCLR (Chen et al., 2020a). Instead of having only two views, ProtoCLR creates multiple views for each support example and averages the features together, just like ProtoNet. They found that ProtoCLR can provide much better performance on few-shot learning episodes than SimCLR, because its training is more similar to standard few-shot learning.

In the following chapters, we will build on top of these machine learning paradigms, and we will start in Chapter 3 by introducing our work in incremental few-shot learning, which combines features from few-shot learning and incremental class learning.

Chapter 3

Combination of Old and New Categories

The availability of large-scale datasets played a significant role in the recent success of deep learning. The need for such a large dataset is however a limitation, since its collection requires intensive human labor. This is also strikingly different from human learning, where new concepts can be learned from very few examples. One line of work that attempts to bridge this gap is few-shot learning; however, as we mentioned in the previous chapter, few-shot models only focus on learning novel classes, ignoring the fact that many common classes are readily available in large datasets.

An approach that aims to enjoy the best of both worlds, the ability to learn from large datasets for common classes with the flexibility of few-shot learning for others, is incremental few-shot learning (Gidaris and Komodakis, 2018). This combines incremental learning where we want to add new classes without catastrophic forgetting (McCloskey and Cohen, 1989), with few-shot learning when the new classes, unlike the base classes, only have a small number of examples. One use case to illustrate the problem is a visual aid system. Most objects of interest are common to all users, e.g., cars, pedestrian signals; however, users would also like to augment the system with additional personalized items or important landmarks in their area. Such a system needs to be able to learn new classes from few examples, without harming the performance of the original classes and typically without access to the dataset used to train the original classes.

In this chapter, we present a method for incremental few-shot learning where during meta-learning we optimize a regularizer that reduces catastrophic forgetting from the incremental few-shot learning. Our proposed regularizer is inspired by attractor networks (Zemel and Mozer, 2001) and regularization-based continual learning (Kirkpatrick et al., 2017). We also show how this regularizer can be optimized, using recurrent back-propagation (Liao et al., 2018; Almeida, 1987; Pineda, 1987) to back-propagate through the few-shot optimization stage. Finally, we show empirically that our proposed method can produce competitive results in incremental few-shot learning on *mini*-ImageNet (Vinyals et al., 2016) and *tiered*-ImageNet (Ren et al., 2018b) tasks.

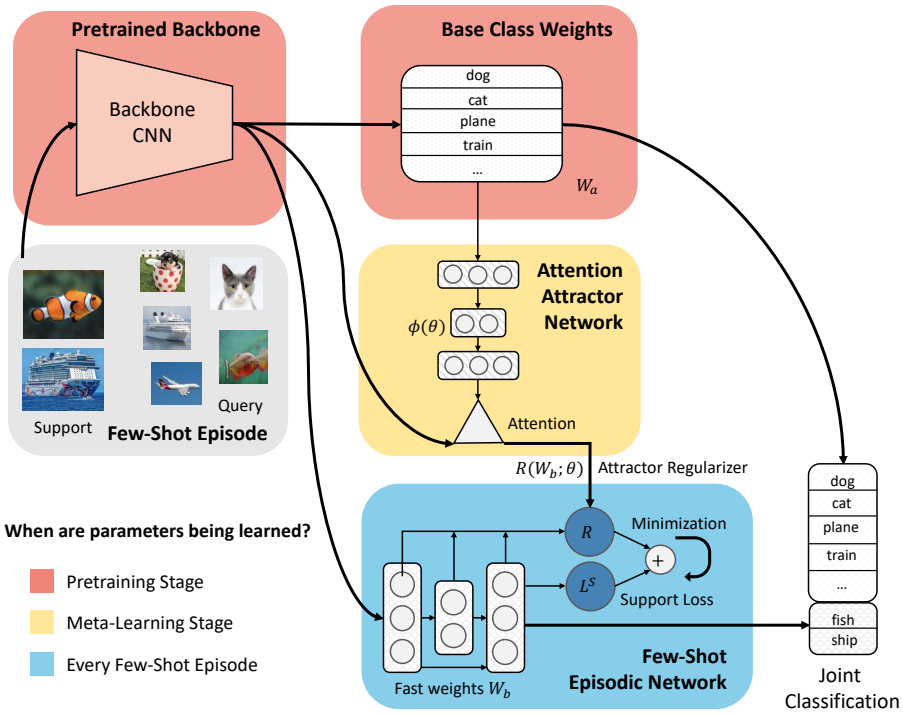


Figure 3.1: Our proposed attention attractor network for incremental few-shot learning. During pretraining, we learn the base class weights W_a and the feature extractor CNN backbone. In the meta-learning stage, a few-shot episode is presented. The support set only contains novel classes, whereas the query set contains both base and novel classes. We learn an episodic classifier network through an iterative solver, to minimize cross-entropy plus an additional regularization term predicted by the attention attractor network by attending to the base classes. The attention attractor network is meta-learned to minimize the expected query loss. During testing an episodic classifier is learned in the same way.

3.1 Model

In this section, we first define the setup of incremental few-shot learning, and then we introduce our new model, the Attention Attractor Network, which attends to the set of base classes according to the few-shot training data by using the attractor regularizing term. Figure 3.1 illustrates the high-level model diagram of our method.

3.1.1 Incremental Few-Shot Learning

The outline of our meta-learning approach to incremental few-shot learning is: (1) We learn a fixed feature representation and a classifier on a set of base classes; (2) In each training and testing episode we train a novel-class classifier with our meta-learned regularizer; (3) We optimize our meta-learned regularizer on combined novel and base classes classification, adapting it to perform well in conjunction with the base classifier. Details of these stages follow.

Pretraining Stage: We learn a base model for the regular supervised classification task on dataset $\{(x_{a,i}, y_{a,i})\}_{i=1}^{N_a}$ where $x_{a,i}$ is the i-th example from dataset $\mathcal{D}_a$ and its labeled class $y_{a,i} \in \{1, 2, ..., K\}$. The purpose of this stage is to learn both a good base classifier and a good representation. The

parameters of the base classifier are learned in this stage and will be fixed after pretraining. We denote the parameters of the top fully connected layer of the base classifier $W_a \in \mathbb{R}^{D\times K}$ where D is the dimension of our learned representation.

Incremental Few-Shot Episodes: A few-shot dataset $\mathcal{D}_b$ is presented, from which we can sample few-shot learning episodes $\mathcal{E}$. Note that this can be the same data source as the pretraining dataset $\mathcal{D}_a$, but sampled episodically. For each N-shot K'-way episode, there are K' novel classes disjoint from the base classes. Each novel class has N and M images from the support set S_b and the query set Q_b respectively. Therefore, we have $\mathcal{E} = (S_b, Q_b), S_b = (x^S_{b,i}, y^S_{b,i})^{N\times K'}_{i=1}, Q_b = (x^Q_{b,i}, y^Q_{b,i})^{M\times K'}_{i=1}$ where $y_{b,i} \in \{K+1, ..., K+K'\}$. S_b and Q_b can be regarded as this episodes training and validation sets. Each episode we learn a classifier on the support set S_b whose learnable parameters W_b are called the *fast weights* as they are only used during this episode. To evaluate the performance on a joint prediction of both base and novel classes, i.e., a $(K+K')$-way classification, a mini-batch $Q_a = \{(x_{a,i}, y_{a,i})\}^{M\times K}_{i=1}$ sampled from $\mathcal{D}_a$ is also added to Q_b to form $Q_{a+b} = Q_a \cup Q_b$. This means that the learning algorithm, which only has access to samples from the novel classes S_b, is evaluated on the *joint* query set Q_{a+b}.

Meta-Learning Stage: In meta-training, we iteratively sample few-shot episodes $\mathcal{E}$ and try to learn the meta-parameters in order to minimize the joint prediction loss on Q_{a+b}. In particular, we design a regularizer $R(\cdot, \theta)$ such that the *fast weights* are learned via minimizing the loss $\ell(W_b, S_b) + R(W_b, \theta)$ where $\ell(W_b, S_b)$ is typically cross-entropy loss for few-shot classification. The meta-learner tries to learn meta-parameters θ such that the optimal *fast weights* W^*_b w.r.t. the above loss function performs well on Q_{a+b}. In our model, meta-parameters θ are encapsulated in our attention attractor network, which produces regularizers for the fast weights in the few-shot learning objective.

Joint Prediction on Base and Novel Classes: We now introduce the details of our joint prediction framework performed in each few-shot episode. First, we construct an episodic classifier, e.g., a logistic regression (LR) model or a multi-layer perceptron (MLP), which takes the learned image features as inputs and classifies them according to the few-shot classes.

During training on the support set S_b, we learn the *fast weights* W_b via minimizing the following regularized cross-entropy objective, which we call the *episodic objective*:

$$L^S(W_b, \theta) = -\frac{1}{NK'} \sum_{i=1}^{NK'} \sum_{c=K+1}^{K+K'} y^S_{b,i,c} \log \hat{y}^S_{b,i,c} + R(W_b, \theta). \tag{3.1}$$

This is a general formulation and the specific functional form of the regularization term $R(W_b, \theta)$ will be specified later. The predicted output $\hat{y}^S_{b,i}$ is obtained via, $\hat{y}^S_{b,i} = \text{softmax}(\big[W_a^\top x_{b,i}, h(x_{b,i}; W^*_b)\big])$, where $h(x_{b,i})$ is our classification network and W_b is the fast weights in the network. In the case of LR, h is a linear model: $h(x_{b,i}; W_b) = W_b^\top x_{b,i}$. h can also be an MLP for more expressive power.

During testing on the query set Q_{a+b}, in order to predict both base and novel classes, we directly augment the softmax with the fixed base class weights W_a, $\hat{y}^Q_i = \text{softmax}(\big[W_a^\top x_i, h(x_i; W^*_b)\big])$, where W^*_b are the optimal parameters that minimize the regularized classification objective in Eq. (3.1).

3.1.2 Attention Attractor Networks

Directly learning the few-shot episode, e.g., by setting $R(W_b, \theta)$ to be zero or simple weight decay, can cause catastrophic forgetting on the base classes. This is because W_b which is trained to maximize the correct novel class probability can dominate the base classes in the joint prediction. In this section, we introduce the Attention Attractor Network to address this problem. The key feature of our attractor network is the regularization term $R(W_b, \theta)$:

$$R(W_b, \theta) = \sum_{k'=1}^{K'} (W_{b,k'} - u_{k'})^\top \mathrm{diag}(\exp(\gamma))(W_{b,k'} - u_{k'}), \tag{3.2}$$

where $u_{k'}$ is the so-called *attractor* and $W_{b,k'}$ is the k'-th column of W_b. This sum of squared Mahalanobis distances from the attractors adds a bias to the learning signal arriving solely from novel classes. Note that for a classifier such as an MLP, one can extend this regularization term in a layer-wise manner. Specifically, one can have separate attractors per layer, and the number of attractors equals the number of output dimension of that layer.

To ensure that the model performs well on base classes, the attractors $u_{k'}$ must contain some information about examples from base classes. Since we can not directly access these base examples, we propose to use the *slow weights* to encode such information. Specifically, each base class has a learned attractor vector U_k stored in the memory matrix $U = [U_1, ..., U_K]$. It is computed as, $U_k = f_\phi(W_{a,k})$, where f is a MLP of which the learnable parameters are ϕ. For each novel class k' its classifier is regularized towards its attractor $u_{k'}$ which is a weighted sum of U_k vectors. Intuitively the weighting is an attention mechanism where each novel class attends to the base classes according to the level of interference, i.e. how the prediction of new class k' causes the forgetting of base class k.

For each class in the support set, we compute the cosine similarity between the average representation of the class and base weights W_a then normalize using a softmax function

$$a_{k',k} = \frac{\exp\left(\tau A(\frac{1}{N}\sum_j h_j \mathbb{1}[y_{b,j} = k'], W_{a,k})\right)}{\sum_k \exp\left(\tau A(\frac{1}{N}\sum_j h_j \mathbb{1}[y_{b,j} = k'], W_{a,k})\right)}, \tag{3.3}$$

where A is the cosine similarity function, h_j are the representations of the inputs in the support set S_b and τ is a learnable temperature scalar. $a_{k',k}$ encodes a normalized pairwise attention matrix between the novel classes and the base classes. The attention vector is then used to compute a linear weighted sum of entries in the memory matrix U, $u_{k'} = \sum_k a_{k',k} U_k + U_0$, where U_0 is an embedding vector and serves as a bias for the attractor.

Algorithm 1 Meta Learning for Incremental Few-Shot Learning

Require: θ_0, $\mathcal{D}_a$, $\mathcal{D}_b$, h

Ensure: θ

$\theta \leftarrow \theta_0$;
for $t = 1$ ... T **do**
  $\{(x_b^S, y_b^S)\}, \{(x_b^Q, y_b^Q)\} \leftarrow \text{GetEpisode}(\mathcal{D}_b)$;
  $\{x_{a+b}^Q, y_{a+b}^Q\} \leftarrow \text{GetMiniBatch}(\mathcal{D}_a) \cup \{(x_b^Q, y_b^Q)\}$;

  repeat
    $L^S \leftarrow \frac{1}{NK'} \sum_i y_{b,i}^S \log \hat{y}_{b,i}^S + R(W_b; \theta)$;
    $W_b \leftarrow \text{OptimizerStep}(W_b, \nabla_{W_b} L^S)$;
  until W_b converges
  $\hat{y}_{a+b,j}^Q \leftarrow \text{softmax}([W_a^\top x_{a+b,j}^Q, h(x_{a+b,j}^Q; W_b)])$;
  $L^Q \leftarrow \frac{1}{2NK'} \sum_j y_{a+b,j}^Q \log \hat{y}_{a+b,j}^Q$;

  {Backprop through the above optimization via RBP}
  {A dummy gradient descent step}
  $W_b' \leftarrow W_b - \alpha \nabla_{W_b} L^S$;
  $J \leftarrow \frac{\partial W_b'}{\partial W_b}$; $v \leftarrow \frac{\partial L^Q}{\partial W_b}$; $g \leftarrow v$;
  repeat
    $v \leftarrow J^\top v - \epsilon v$; $g \leftarrow g + v$;
  until g converges

  $\theta \leftarrow \text{OptimizerStep}(\theta, g^\top \frac{\partial W_b'}{\partial \theta})$
end for

Our design takes inspiration from attractor networks Mozer (2009); Zemel and Mozer (2001), where for each base class one learns an “attractor" that stores the relevant memory regarding that class. We call our full model “dynamic attractors" as they may vary with each episode even after meta-learning. In contrast, if we only have the bias term U_0, i.e. a single attractor which is shared by all novel classes, it will not change after meta-learning from one episode to the other. We call this model variant the “static attractor".

In summary, our meta parameters θ include ϕ, U_0, γ and τ, which is on the same scale as the number of parameters in W_a. It is important to note that $R(W_b, \theta)$ is convex w.r.t. W_b. Therefore, if we use the LR model as the classifier, the overall training objective on episodes in Eq. (3.1) is convex which implies that the optimum $W_b^*(\theta, S_b)$ is guaranteed to be unique and achievable. Here we emphasize that the optimal parameters W_b^* are functions of parameters θ and few-shot samples S_b.

During meta-learning, θ are updated to minimize an expected loss of the query set Q_{a+b} which contains both base and novel classes, averaging over all few-shot learning episodes,

$$\min_{\theta} \quad \mathop{\mathbb{E}}_{\mathcal{E}} \left[L^Q(\theta, S_b) \right] = \mathop{\mathbb{E}}_{\mathcal{E}} \left[\sum_{j=1}^{M(K+K')} \sum_{c=1}^{K+K'} y_{j,c} \log \hat{y}_{j,c}(\theta, S_b) \right], \tag{3.4}$$

where the predicted class is $\hat{y}_j(\theta, S_b) = \text{softmax}\left(\left[W_a^\top x_j, h\left(x_j; W_b^*(\theta, S_b)\right) \right] \right)$.

3.1.3 Learning via Recurrent Back-Propagation

As there is no closed-form solution to the episodic objective (the optimization problem in Eq. 3.1), in each episode we need to minimize L^S to obtain W_b^* through an iterative optimizer. The question is how to efficiently compute $\frac{\partial W_b^*}{\partial \theta}$, i.e., back-propagating through the optimization. One option is to unroll the iterative optimization process in the computation graph and use back-propagation through time (BPTT) (Werbos, 1990). However, the number of iterations for a gradient-based optimizer to converge can be on the order of thousands, and BPTT can be computationally prohibitive. Another way is to use the truncated BPTT (Williams and Peng, 1990) (T-BPTT) which optimizes for T steps of gradient-based optimization, and is commonly used in meta-learning problems. However, when T is small the training objective could be significantly biased.

Alternatively, the recurrent back-propagation (RBP) algorithm (Liao et al., 2018; Almeida, 1987; Pineda, 1987) allows us to back-propagate through the fixed point efficiently without unrolling the computation graph and storing intermediate activations. Consider a vanilla gradient descent process on W_b with step size α. The difference between two steps Φ can be written as

$$\Phi(W_b^{(t)}) = W_b^{(t)} - F(W_b^{(t)}), \tag{3.5}$$

where

$$F(W_b^{(t)}) = W_b^{(t+1)} = W_b^{(t)} - \alpha \nabla L^S(W_b^{(t)}). \tag{3.6}$$

Since $\Phi(W_b^*(\theta))$ is identically zero as a function of θ,

$$\Phi(W_b^*) = W_b^* - F(W_b^*) = 0, \tag{3.7}$$

therefore using the implicit function theorem we have

$$\frac{\partial W_b^*}{\partial \theta} = \frac{\partial F(W_b^*)}{\partial \theta}, \tag{3.8}$$

$$\frac{\partial W_b^*}{\partial \theta} = J_{F,W_b^*}^\top \frac{\partial W_b^*}{\partial \theta} + \frac{\partial F}{\partial \theta}, \tag{3.9}$$

$$\frac{\partial F}{\partial \theta} = (I - J_{F,W_b^*}^\top) \frac{\partial W_b^*}{\partial \theta}, \tag{3.10}$$

$$\frac{\partial W_b^*}{\partial \theta} = (I - J_{F,W_b^*}^\top)^{-1} \frac{\partial F}{\partial \theta}, \tag{3.11}$$

where J_{F,W_b^*} denotes the Jacobian matrix of the mapping F evaluated at W_b^*. Algorithm 1 outlines the key steps for learning the episodic objective using RBP in the incremental few-shot learning setting. Note that the RBP algorithm implicitly inverts $(I - J^\top)$ by computing the matrix inverse vector product, and has the same time complexity compared to truncated BPTT given the same number of unrolled steps, but meanwhile RBP does not have to store intermediate activations.

Damped Neumann RBP To compute the matrix-inverse vector product $(I - J^\top)^{-1}v$, (Liao et al., 2018) propose to use the Neumann series:

$$(I - J^\top)^{-1}v = \sum_{n=0}^{\infty}(J^\top)^n v \equiv \sum_{n=0}^{\infty} v^{(n)}. \tag{3.12}$$

Note that $J^\top v$ can be computed by standard back-propagation. However, directly applying the Neumann RBP algorithm sometimes leads to numerical instability. Therefore, we propose to add a damping term $0 < \epsilon < 1$ to $I - J^\top$. This results in the following update:

$$\tilde{v}^{(n)} = (J^\top - \epsilon I)^n v. \tag{3.13}$$

In practice, we found the damping term with $\epsilon = 0.1$ helps alleviate the issue significantly.

3.2 Related Work

3.2.1 Gradient-Based Meta-Learning

Many recurrent network-based meta-learning methods typically learn the update policy yet lack an overall learning objective in the few-shot episodes. For gradient-based models like MAML (Finn et al., 2019), the performance could drop if at test time the model is trained for longer steps. To address this problem, Bertinetto et al. (2019) proposes to use fast convergent models like logistic regression (LR), which can be back-propagated via a closed-form update rule. Compared to Bertinetto et al. (2019), our proposed method using recurrent back-propagation (Liao et al., 2018; Almeida, 1987; Pineda, 1987) is more general as it does not require a closed-form update, and the inner loop solver can employ any existing continuous optimizers.

3.2.2 Incremental Few-Shot Learning

Incremental few-shot learning is also known as low-shot learning. To leverage a good representation, Hariharan and Girshick (2017); Wang et al. (2018); Gidaris and Komodakis (2018) starts off with a pre-trained network on a set of base classes, and tries to augment the classifier with a batch of new classes that have not been seen during training. Hariharan and Girshick (2017) proposes the squared gradient magnitude loss, which makes the learned classifier from the low-shot examples have a smaller gradient value when learning on all examples. Wang et al. (2018) propose the prototypical matching networks, a combination of prototypical network and matching network. The paper also adds hallucination, which generates new examples. Gidaris and Komodakis (2018) proposes an attention-based model which generates weights for novel categories. They also promote the use of cosine similarity between feature representations and weight vectors to classify images. In contrast, during each few-shot episode, we directly learn a classifier network that is randomly initialized and solved till convergence, unlike Gidaris and Komodakis (2018) which directly output the prediction. We found that using an iterative solver with the learned regularizer significantly improves the classifier model on the task of incremental few-shot learning.

Table 3.1: Comparison of our proposed model with other methods

Method	Few-shot learner	Episodic objective	Attention mechanism
Imprint (Qi et al., 2018)	Prototypes	N/A	N/A
LwoF (Gidaris and Komodakis, 2018)	Prototypes + base classes	N/A	Attention on base classes
Ours	A fully trained classifier	Cross entropy on novel classes	Attention on learned attractors

Table 3.2: *mini*-ImageNet 64+5-way results

Model	1-shot Acc. ↑	1-shot Δ ↓	5-shot Acc. ↑	5-shot Δ ↓
ProtoNet	42.73 ± 0.15	-20.21	57.05 ± 0.10	-31.72
Imprint	41.10 ± 0.20	-22.49	44.68 ± 0.23	-27.68
LwoF	52.37 ± 0.20	-13.65	59.90 ± 0.20	-14.18
Ours	**54.95** ± 0.30	-11.84	**63.04** ± 0.30	**-10.66**

Table 3.3: *tiered*-ImageNet 200+5-way results

Model	1-shot Acc. ↑	1-shot Δ ↓	5-shot Acc. ↑	5-shot Δ ↓
ProtoNet	30.04 ± 0.21	-29.54	41.38 ± 0.28	-26.39
Imprint	39.13 ± 0.15	-22.26	53.60 ± 0.18	-16.35
LwoF	52.40 ± 0.33	-8.27	62.63 ± 0.31	-6.72
Ours	**56.11** ± 0.33	**-6.11**	**65.52** ± 0.31	**-4.48**

Δ = average decrease in acc. caused by *joint* prediction within base and novel classes ($\Delta = \frac{1}{2}(\Delta_a + \Delta_b)$)
↑ (↓) represents higher (lower) is better.

3.3 Experiments

We experiment on two few-shot classification datasets, *mini*-ImageNet and *tiered*-ImageNet. Both are subsets of ImageNet (Russakovsky et al., 2015), with images sizes reduced to 84×84 pixels. We also modified the datasets to accommodate the incremental few-shot learning settings.

3.3.1 Datasets

- ***mini*-ImageNet** Proposed by Vinyals et al. (2016), *mini*-ImageNet contains 100 object classes and 60,000 images. We used the splits proposed by Ravi and Larochelle (2017), where training, validation, and testing have 64, 16 and 20 classes respectively.

- ***tiered*-ImageNet** Proposed by Ren et al. (2018b), *tiered*-ImageNet is a larger subset of ILSVRC-12. It features a categorical split among training, validation, and testing subsets. The categorical split means that classes that belong to the same high-level category, e.g. “working dog” and "terrier" or some other dog breed, are not split between training, validation, and test. This is a harder task, but one that more strictly evaluates generalization to new classes. It is also an order of magnitude larger than *mini*-ImageNet.

3.3.2 Experiment Setup

We use a standard ResNet backbone (He et al., 2016) to learn the feature representation through supervised training. For *mini*-ImageNet experiments, we follow Mishra et al. (2017) and use a modified version of ResNet-10. For *tiered*-ImageNet, we use the standard ResNet-18 (He et al., 2016), but replace all batch normalization (Ioffe and Szegedy, 2015) layers with group normalization (Wu and He, 2018), as there is a large distributional shift from training to testing in *tiered*-ImageNet due to categorical splits. We used standard data augmentation, with random crops and horizontal flips. We use the same pretrained checkpoint as the starting point for meta-learning.

In the meta-learning stage as well as the final evaluation, we sample a few-shot episode from the $\mathcal{D}_b$, together with a regular mini-batch from the $\mathcal{D}_a$. The base class images are added to the query set of the few-shot episode. The base and novel classes are maintained in equal proportion in our experiments. For all the experiments, we consider 5-way classification with 1 or 5 support examples (i.e. shots). In the experiments, we use a query set of size 25×2 =50.

We use L-BFGS (Zhu et al., 1997) to solve the inner loop of our models to make sure W_b converges. We use the ADAM optimizer (Kingma and Ba, 2015) for meta-learning with a learning rate of 1e-3, which decays by a factor of 10 after 4,000 steps, for a total of 8,000 steps. We fix recurrent backpropagation to 20 iterations and $\epsilon = 0.1$.

We study two variants of the classifier network. The first is a logistic regression model with a single weight matrix W_b. The second is a 2-layer fully connected MLP model with 40 hidden units in the middle and tanh non-linearity. To make training more efficient, we also add a shortcut connection in our MLP, which directly links the input to the output. In the second stage of training, we keep all backbone weights frozen and only train the meta-parameters θ.

3.3.3 Evaluation Metrics

We consider the following evaluation metrics: 1) overall accuracy on individual query sets and the joint query set ("Base", "Novel", and "Both"); and 2) decrease in performance caused by *joint* prediction within the base and novel classes, considered separately ("Δ_a" and "Δ_b"). Finally we take the average $\Delta = \frac{1}{2}(\Delta_a + \Delta_b)$ as a key measure of the overall decrease in accuracy.

3.3.4 Comparisons

We implemented and compared three methods. First, we adapted Prototypical Networks (Snell et al., 2017) to incremental few-shot settings. For each base class, we store a base representation, which is the average representation (prototype) over all images belonging to the base class. During the few-shot learning stage, we again average the representation of the few-shot classes and add them to the bank of base representations. Finally, we retrieve the nearest neighbor by comparing the representation of a test image with entries in the representation store. In summary, both W_a and W_b are stored as the average representation of all images seen so far that belong to a certain class. We also compare to the following methods:

- **Weights Imprinting ("Imprint")** (Qi et al., 2018): the base weights W_a are learned regularly through supervised pre-training, and W_b are computed using prototypical averaging.

- **Learning without Forgetting ("LwoF")** (Gidaris and Komodakis, 2018): Similar to Qi et al. (2018), W_b are computed using prototypical averaging. In addition, W_a is finetuned during episodic meta-learning. We implemented the most advanced variants proposed in the paper, which involves a class-wise attention mechanism. This model is the previous state-of-the-art method on incremental few-shot learning and has a better performance compared to other low-shot models (Wang et al., 2018; Hariharan and Girshick, 2017).

Table 3.4: Ablation studies on *mini*-ImageNet

	1-shot Acc. ↑	1-shot Δ ↓	5-shot Acc. ↑	5-shot Δ ↓
LR	52.74 ± 0.24	-13.95	60.34 ± 0.20	-13.60
LR +S	53.63 ± 0.30	-12.53	62.50 ± 0.30	-11.29
LR +A	**55.31** ± 0.32	**-11.72**	63.00 ± 0.29	-10.80
MLP	49.36 ± 0.29	-16.78	60.85 ± 0.29	-12.62
MLP +S	54.46 ± 0.31	-11.74	62.79 ± 0.31	-10.77
MLP +A	54.95 ± 0.30	-11.84	**63.04** ± 0.30	**-10.66**

Table 3.5: Ablation studies on *tiered*-ImageNet

	1-shot Acc. ↑	1-shot Δ ↓	5-shot Acc. ↑	5-shot Δ ↓
LR	48.84 ± 0.23	-10.44	62.08 ± 0.20	-8.00
LR +S	55.36 ± 0.32	-6.88	65.53 ± 0.30	-4.68
LR +A	55.98 ± 0.32	**-6.07**	65.58 ± 0.29	**-4.39**
MLP	41.22 ± 0.35	-10.61	62.70 ± 0.31	-7.44
MLP +S	**56.16** ± 0.32	-6.28	**65.80** ± 0.31	-4.58
MLP +A	56.11 ± 0.33	6.11	65.52 ± 0.31	-4.48

"+S" stands for static attractors, and "+A" for attention attractors.

3.3.5 Results

We first evaluate our vanilla approach on the standard few-shot classification benchmark where no base classes are present in the query set. Our vanilla model consists of a pretrained CNN and a single-layer logistic regression with weight decay learned from scratch; this model performs on-par with other competitive meta-learning approaches (1-shot 55.40 ± 0.51, 5-shot 70.17 ± 0.46). Note that our model uses the same backbone architecture as Mishra et al. (2017) and Gidaris and Komodakis (2018), and is directly comparable with their results. Similar findings of strong results using simple logistic regression on few-shot classification benchmarks are also recently reported in Chen et al. (2019). Our full model has similar performance as the vanilla model on pure few-shot benchmarks, and the full table is available in Supp. Materials.

Next, we compare our models to other methods on incremental few-shot learning benchmarks in Tables 3.2 and 3.3. On both benchmarks, our best performing model shows a significant margin over the prior works that predict the prototype representation without using an iterative optimization (Snell et al., 2017; Qi et al., 2018; Gidaris and Komodakis, 2018).

3.3.6 Ablation Studies

To understand the effectiveness of each part of the proposed model, we consider the following variants:

- **Vanilla ("LR, MLP")** optimizes a logistic regression or an MLP network at each few-shot episode, with a weight decay regularizer.
- **Static attractor ("+S")** learns a fixed attractor center u and attractor slope γ for all classes.
- **Attention attractor ("+A")** learns the full attention attractor model. For MLP models, the weights below the final layer are controlled by attractors predicted by the average representation across all the episodes. f_ϕ is an MLP with one hidden layer of 50 units.

Tables 3.4 and 3.5 shows the ablation experiment results. In all cases, the learned regularization function shows better performance than a manually set weight decay constant on the classifier network, in terms of both jointly predicting base and novel classes, as well as less degradation from individual prediction. On *mini*-ImageNet, our attention attractors have a clear advantage over static attractors.

Formulating the classifier as an MLP network is slightly better than the linear models in our experiments. Although the final performance is similar, our RBP-based algorithm has the flexibility

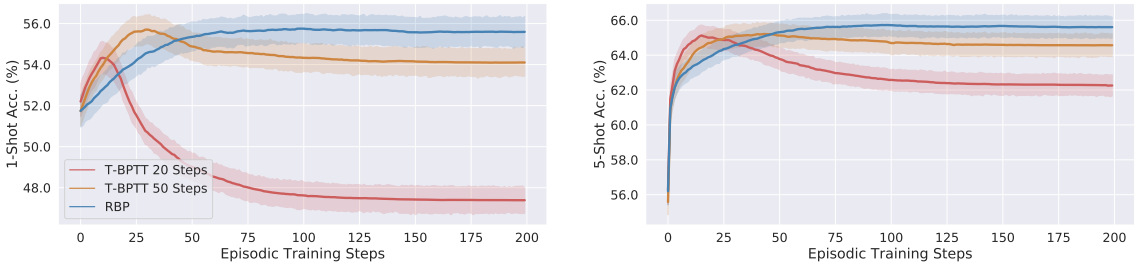
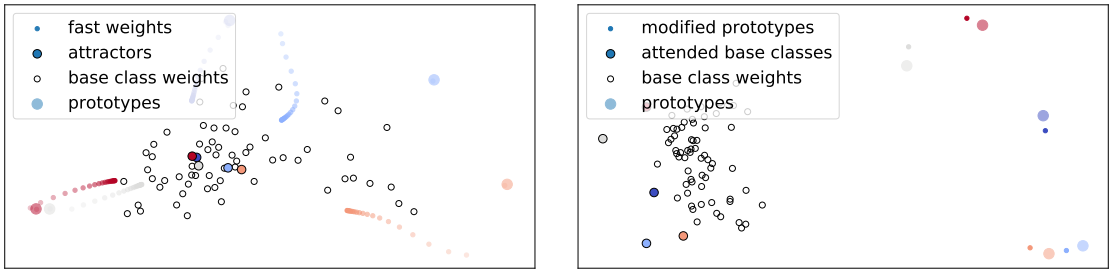


Figure 3.2: Learning the proposed model using truncated BPTT vs. RBP. Models are evaluated with 1-shot (left) and 5-shot (right) 64+5-way episodes, with various number of gradient descent steps.



(a) Ours (b) LwoF (Gidaris and Komodakis, 2018)

Figure 3.3: Visualization of a 5-shot 64+5-way episode using PCA. **Left:** Our attractor model learns to "pull" prototypes (large colored circles) towards base class weights (white circles). We visualize the trajectories during episodic training; **Right:** Dynamic few-shot learning without forgetting (Gidaris and Komodakis, 2018).

of adding the fast episodic model with more capacity. Unlike Bertinetto et al. (2019), we do not rely on an analytic form of the gradients of the optimization process.

Comparison to truncated BPTT (T-BPTT) An alternative way to learn the regularizer is to unroll the inner optimization for a fixed number of steps in a differentiable computation graph, and then back-propagate through time. Truncated BPTT is a popular learning algorithm in many recent meta-learning approaches (Andrychowicz et al., 2016; Ravi and Larochelle, 2017; Finn et al., 2017; Sprechmann et al., 2018; Balaji et al., 2018). Shown in Figure 3.2, the performance of T-BPTT learned models are comparable to ours; however, when solved to convergence at test time, the performance of T-BPTT models drops significantly. This is expected as they are only guaranteed to work well for a certain number of steps, and failed to learn a good regularizer. While an early-stopped T-BPTT model can do equally well, in practice it is hard to tell when to stop; whereas for the RBP model, doing the full episodic training is very fast since the number of support examples is small.

Visualization of attractor dynamics We visualize attractor dynamics in Figure 3.3. Our learned attractors pulled the fast weights close towards the base class weights. In comparison, Gidaris and Komodakis (2018) only modifies the prototypes slightly.

Varying the number of base classes While the framework proposed in this chapter cannot be directly applied on class-incremental continual learning, as there is no module for memory

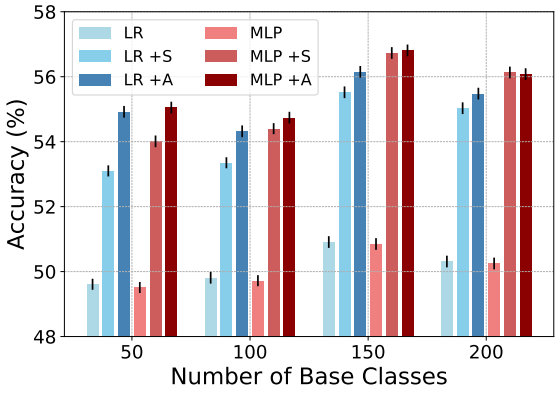


Figure 3.4: Results on *tiered*-ImageNet with {50, 100, 150, 200} base classes.

consolidation, we can simulate the continual learning process by varying the number of base classes, to see how the proposed models are affected by different stages of continual learning. Figure 3.4 shows that the learned regularizers consistently improve over baselines with weight decay only. The overall accuracy increases from 50 to 150 classes due to better representations on the backbone network and drops at 200 classes due to a more challenging classification task.

3.4 Discussion and Conclusion

In this chapter, we propose a new few-shot learning algorithm that can recognize new classes with a few examples and can also output base classes that have been learned before. This has extended the paradigm of classic few-shot learning, which only handles new classes at test time and interestingly, standard methods like ProtoNet (Snell et al., 2017) seem to not perform well on the joint classification task. we propose an attention attractor model, which regulates a per-episode training objective by attending to the set of base classes. We show that our iterative model that solves the few-shot objective till convergence is better than baselines that do one-step inference, and that recurrent back-propagation is an effective and modular tool for learning in a general meta-learning setting, whereas truncated back-propagation through time fails to learn functions that converge well.

There are still many limitations of this work, and since its publication, many of them have been addressed by subsequent literature. First of all, the metric we used here is in terms of the mean accuracy and the relative drop in performance. In addition to these two, Ye et al. (2021) propose to use harmonic mean accuracy, so that the model will be strongly penalized if it achieves a low score on either base or novel classes. They also proposed a new algorithm for synthesizing joint classifiers that achieves state-of-the-art performance.

Another limitation of our work is that we focus on the setting where there is no access to the base dataset. Although this approach is useful in the scenario where the base dataset is not available, and is very memory efficient, in the end, it might still be better to use a few images from the base dataset. This has been studied in the literature of low-shot learning, as well as many continual learning algorithms (Rebuffi et al., 2017; Chaudhry et al., 2019a; Kim et al., 2020; Buzzega et al., 2020; Mai et al., 2020). Besides data replay, we could also consider generative replay (van de Ven and Tolias, 2018; van de Ven et al., 2020), where a generative model is separately learned to generative

old data.

One line of subsequent works aims to extend the incremental few-shot learning setup beyond image classification. Pérez-Rúa et al. (2020) investigated incremental few-shot detection and Cermelli et al. (2020) proposed incremental few-shot segmentation. These works can achieve more fine-grained visual understanding by obtaining object localizations.

Ideally, the incremental learning procedure can be made sequentially, but in our work, this only contains the base and novel classes. A few subsequent works (Tao et al., 2020; Liu et al., 2020b; Zhang et al., 2021) addressed this extension. In Chapter 6, we will also present another perspective of incremental and continual few-shot learning by considering the problem to be task-agnostic. Instead of receiving explicit task boundaries, the model will be making a prediction after seeing each example in a sequence.

Chapter 4

Improving Few-Shot Learning with Unlabeled Data

There have been various meta-learning formulations that have led to significant progress recently in few-shot classification. However, this progress has been limited in the setup of each few-shot learning episode, which differs from how humans learn new concepts in many dimensions. In Chapter 3, we looked at learning both old and new classes, where new classes only contain a few examples, but each episode there still contains the support and query sets that are just like standard supervised learning. In this chapter, we aim to generalize the standard setup in two different ways.

First, we consider a scenario where the new classes are learned in the presence of additional *unlabeled* data. While there have been many successful applications of semi-supervised learning to the regular setting of a single classification task (Chapelle et al., 2010), where classes at training and test time are the same, such work has not addressed the challenge of performing transfer to new classes are never seen at training time, which we consider here.

Second, we consider the situation where the new classes to be learned are not viewed in isolation. Instead, many of the unlabeled examples are from different classes; the presence of such *distractor* classes introduces an additional and more realistic level of difficulty to the few-shot problem.

In this chapter, we define a new problem of semi-supervised few-shot learning and propose benchmarks for evaluation that are adapted from the Omniglot and *mini*-ImageNet benchmarks used in ordinary few-shot learning. We perform an extensive empirical investigation of the two settings mentioned above, with and without distractor classes. We then propose and study three novel extensions of Prototypical Networks (Snell et al., 2017), a state-of-the-art approach to few-shot learning, to the semi-supervised setting. Finally, we demonstrate in our experiments that our semi-supervised variants successfully learn to leverage unlabeled examples and outperform purely supervised Prototypical Networks.

4.1 Semi-Supervised Few-Shot Learning

We now define the semi-supervised setting considered in this work for few-shot learning.

The training set is denoted as a tuple of labeled and unlabeled examples: $(\mathcal{S}, \mathcal{R})$. The labeled portion is the usual support set $\mathcal{S}$ of the few-shot learning literature, containing a list of tuples

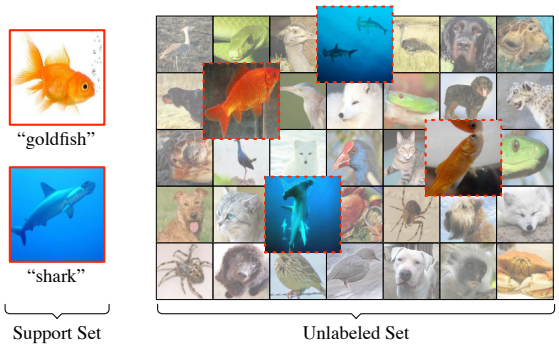


Figure 4.1: Consider a setup where the aim is to learn a classifier to distinguish between two previously unseen classes, goldfish, and shark, given not only labeled examples of these two classes, but also a larger pool of unlabeled examples, some of which may belong to one of these two classes of interest. In this work, we aim to move a step closer to this more natural learning framework by incorporating in our learning episodes unlabeled data from the classes we aim to learn representations for (shown with dashed red borders) as well as from *distractor* classes .

of inputs and targets. In addition to classic few-shot learning, we introduce an unlabeled set $\mathcal{R}$ containing only inputs: $\mathcal{R} = \{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_M\}$. As in the purely supervised setting, our models are trained to perform well when predicting the labels for the examples in the episode's query set $\mathcal{Q}$. Figure 4.2 shows a visualization of training and test episodes.

4.1.1 Semi-Supervised Prototypical Networks

In their original formulation, Prototypical Networks do not specify a way to leverage the unlabeled set $\mathcal{R}$. In what follows, we now propose various extensions that start from the basic definition of prototypes $\boldsymbol{p}_c$ and provide a procedure for producing refined prototypes $\tilde{\boldsymbol{p}}_c$ using the unlabeled examples in $\mathcal{R}$.

After the refined prototypes are obtained, each of these models is trained with the same loss function for ordinary Prototypical Networks of Equation 2.10, but replacing $\boldsymbol{p}_c$ with $\tilde{\boldsymbol{p}}_c$. That is, each query example is classified into one of the N classes based on the proximity of its embedded position with the corresponding *refined* prototypes, and the average negative log probability of the correct classification is used for training.

Prototypical Networks with Soft k-Means

We first consider a simple way of leveraging unlabeled examples for refining prototypes, by taking inspiration from semi-supervised clustering. Viewing each prototype as a cluster center, the refinement process could attempt to adjust the cluster locations to better fit the examples in both the support and unlabeled sets. Under this view, cluster assignments of the labeled examples in the support set are considered known and fixed to each example's label. The refinement process must instead estimate the cluster assignments of the unlabeled examples and adjust the cluster locations (the prototypes) accordingly.

One natural choice would be to borrow from the inference performed by soft k-means. We prefer this version of k-means over hard assignments since hard assignments would make the inference non-differentiable. We start with the regular Prototypical Network's prototypes $\boldsymbol{p}_c$ (as specified in

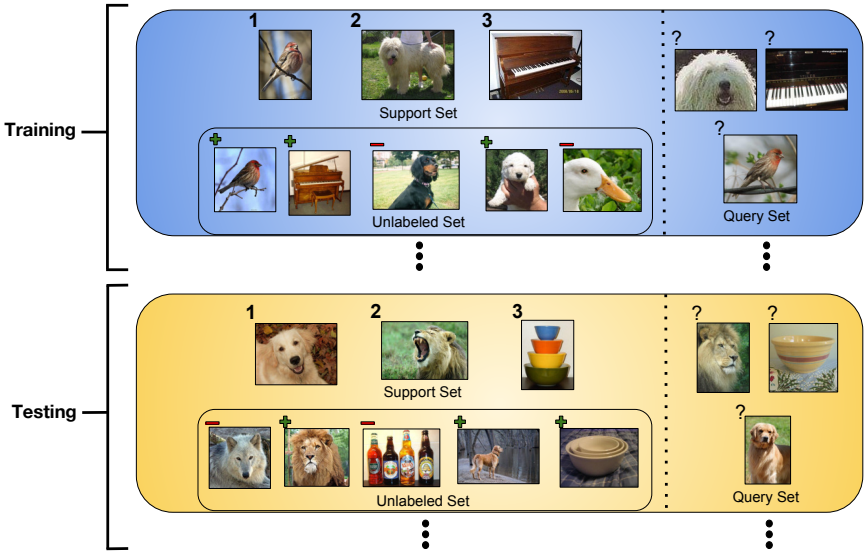


Figure 4.2: Example of the semi-supervised few-shot learning setup. Training involves iterating through training episodes, consisting of a support set $\mathcal{S}$, an unlabeled set $\mathcal{R}$, and a query set $\mathcal{Q}$. The goal is to use the labeled items (shown with their numeric class label) in $\mathcal{S}$ and the unlabeled items in $\mathcal{R}$ within each episode to generalize to good performance on the corresponding query set. The unlabeled items in $\mathcal{R}$ may either be pertinent to the classes we are considering (shown above with green plus signs) or they may be *distractor* items that belong to a class that is not relevant to the current episode (shown with red minus signs). However, note that the model does not actually have ground truth information as to whether each unlabeled example is a distractor or not; the plus/minus signs are shown only for illustrative purposes. At test time, we are given new episodes consisting of novel classes not seen during training that we use to evaluate the meta-learning method.

Equation 2.8) as the cluster locations. Then, the unlabeled examples get a partial assignment ($\tilde{z}_{j,c}$) to each cluster based on their Euclidean distance to the cluster locations. Finally, refined prototypes are obtained by incorporating these unlabeled examples.

This process can be summarized as follows:

$$\tilde{\boldsymbol{p}}_c = \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c} + \sum_j h(\tilde{\boldsymbol{x}}_j) \tilde{z}_{j,c}}{\sum_i z_{i,c} + \sum_j \tilde{z}_{j,c}}, \quad \text{where} \quad \tilde{z}_{j,c} = \frac{\exp\left(-||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_c||_2^2\right)}{\sum_{c'} \exp\left(-||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_{c'}||_2^2\right)} \tag{4.1}$$

Predictions of each query input's class is then modeled as in Equation 2.9, but using the refined prototypes $\tilde{\boldsymbol{p}}_c$.

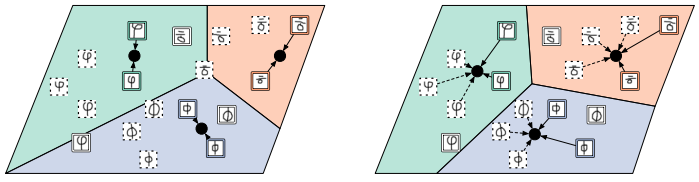


Figure 4.3: Left: The prototypes are initialized based on the mean location of the examples of the corresponding class, as in ordinary Prototypical Networks. Support, unlabeled, and query examples have solid, dashed, and white-colored borders respectively. Right: The refined prototypes obtained by incorporating the unlabeled examples, which classifies all query examples correctly.

We could perform several iterations of refinement, as is usual in k-means. However, we have experimented with various number of iterations and found results to not improve beyond a single refinement step. This is likely because the additional refinement steps make gradient backpropagation to the encoder network more difficult.

Prototypical Networks with Soft k-Means with a Distractor Cluster

The soft k-means approach described above implicitly assumes that each unlabeled example belongs to either one of the N classes in the episode. However, it would be much more general to not make that assumption and have a model robust to the existence of examples from other classes, which we refer to as distractor classes. For example, such a situation would arise if we wanted to distinguish between pictures of unicycles and scooters, and decided to add an unlabeled set by downloading images from the web. It then would not be realistic to assume that all these images are of unicycles or scooters. Even with a focused search, some may be from similar classes, such as bicycles.

Since soft k-means distributes its soft assignments across all classes, distractor items could be harmful and interfere with the refinement process, as prototypes would be adjusted to also partially account for these distractors. A simple way to address this is to add an additional cluster whose purpose is to capture the distractors, thus preventing them from polluting the clusters of the classes of interest:

$$\boldsymbol{p}_c = \begin{cases} \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c}}{\sum_i z_{i,c}} & \text{for} \quad c = 1...N \\ \mathbf{0} & \text{for} \quad c = N+1 \end{cases} \tag{4.2}$$

Here we take the simplifying assumption that the distractor cluster has a prototype centered at the origin. We also consider introducing length-scales r_c to represent variations in the within-cluster distances, specifically for the distractor cluster:

$$\tilde{z}_{j,c} = \frac{\exp\left(-\frac{1}{r_c^2}||\tilde{\boldsymbol{x}}_j - \boldsymbol{p}_c||_2^2 - A(r_c)\right)}{\sum_{c'} \exp\left(-\frac{1}{r_c^2}||\tilde{\boldsymbol{x}}_j - \boldsymbol{p}_{c'}||_2^2 - A(r_{c'})\right)}, \quad \text{where} \quad A(r) = \frac{1}{2}\log(2\pi) + \log(r) \tag{4.3}$$

For simplicity, we set $r_{1...N}$ to 1 in our experiments, and only learn the length-scale of the distractor cluster r_{N+1}.

Prototypical Networks with Soft k-Means and Masking

Modeling distractor unlabeled examples with a single cluster is likely too simplistic. Indeed, it is inconsistent with our assumption that each cluster corresponds to one class, since distractor examples may very well cover more than a single natural object category. Continuing with our unicycles and bicycles example, our web search for unlabeled images could accidentally include not only bicycles but other related objects such as tricycles or cars. This was also reflected in our experiments, where we constructed the episode generating process so that it would sample distractor examples from multiple classes.

To address this problem, we propose an improved variant: instead of capturing distractors with a high-variance catch-all cluster, we model distractors as examples that are not within some area of any of the legitimate class prototypes. This is done by incorporating a soft-masking mechanism on the contribution of unlabeled examples. At a high level, we want unlabeled examples that are closer

to a prototype to be masked less than those that are farther.

More specifically, we modify the soft k-means refinement as follows. We start by computing normalized distances $\tilde{d}_{j,c}$ between examples $\tilde{\boldsymbol{x}}_j$ and prototypes $\boldsymbol{p}_c$:

$$\tilde{d}_{j,c} = \frac{d_{j,c}}{\frac{1}{M}\sum_j d_{j,c}}, \text{ where } d_{j,c} = ||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_c||_2^2 \tag{4.4}$$

Then, soft thresholds β_c and slopes γ_c are predicted for each prototype, by feeding to a small neural network various statistics of the normalized distances for the prototype:

$$[\beta_c, \gamma_c] = \text{MLP}\left(\left[\min_j(\tilde{d}_{j,c}), \max_j(\tilde{d}_{j,c}), \text{var}_j(\tilde{d}_{j,c}), \text{skew}_j(\tilde{d}_{j,c}), \text{kurt}_j(\tilde{d}_{j,c})\right]\right) \tag{4.5}$$

This allows each threshold to use information on the amount of intra-cluster variation to determine how aggressively it should cut out unlabeled examples.

Then, soft masks $m_{j,c}$ for the contribution of each example to each prototype are computed, by comparing to the threshold the normalized distances, as follows:

$$\tilde{\boldsymbol{p}}_c = \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c} + \sum_j h(\tilde{\boldsymbol{x}}_j)\tilde{z}_{j,c} m_{j,c}}{\sum_i z_{i,c} + \sum_j \tilde{z}_{j,c} m_{j,c}}, \text{ where } m_{j,c} = \sigma\left(-\gamma_c\left(\tilde{d}_{j,c} - \beta_c\right)\right) \tag{4.6}$$

where $\sigma(\cdot)$ is the sigmoid function.

When training with this refinement process, the model can now use its MLP in Equation 4.5 to learn to include or ignore entirely certain unlabeled examples. The use of soft masks makes this process entirely differentiable[1]. Finally, much like for regular soft k-means (with or without a distractor cluster), while we could recursively repeat the refinement for multiple steps, we found a single step to perform well enough.

4.2 Related Work

The approach within which our work falls is that of similarity learning methods that are introduced in Chapter 2. Previous work in similarity learning for few-shot-classification includes Siamese Networks (Koch et al., 2015), Matching Networks (Vinyals et al., 2016), and Prototypical Networks (Snell et al., 2017), which is the model we extend to the semi-supervised setting in our work. Other meta-learning methods, such as gradient-based (Finn et al., 2017) and memory-based (Santoro et al., 2016; Ravi and Larochelle, 2017) meta-learning, are also competitive for standard few-shot learning, but we chose to extend Prototypical Networks in this work for its simplicity and efficiency.

Our proposed setup is also related to the problem of open-set recognition (Scheirer et al., 2013), which aims to add an "unknown" option in a regular classification task. Open-set classifiers typically leverage classic machine learning frameworks, e.g. SVMs (Scheirer et al., 2013), multi-class logistic regression (Bendale and Boult, 2016), and autoencoders (Oza and Patel, 2019). The difference between our setup and the classic open-set recognition set up is that our distractors also appear during training, and it is important to avoid interference from these unknown items during learning.

As for the literature on semi-supervised learning, while it is quite vast (Zhu, 2005; Chapelle et al., 2010), the most relevant category to our work is related to self-training (Yarowsky, 1995; Rosenberg

[1]We stop gradients of the statistics in Equation 4.5, to avoid potential numerical instabilities.

et al., 2005). Here, a classifier is first trained on the initial training set. The classifier is then used to classify unlabeled items, and the most confidently predicted unlabeled items are added to the training set with the prediction of the classifier as the assumed label. This is similar to our soft k-Means extension to Prototypical Networks. Indeed, since the soft assignments (Equation 4.1) match the regular Prototypical Network's classifier output for new inputs (Equation 2.9), then the refinement can be thought of re-feeding to a Prototypical Network a new support set augmented with (soft) self-labels from the unlabeled set.

Our algorithm is also related to transductive learning (Vapnik, 1998; Joachims, 1999; Fu et al., 2015), where the base classifier gets refined by seeing the unlabeled examples. In practice, one could use our method in a transductive setting where the unlabeled set is the same as the query set; however, here to avoid our model memorizing labels of the unlabeled set during the meta-learning procedure, we split out a separate unlabeled set that is different from the query set.

In addition to the original k-Means method (Lloyd, 1982), the most related work to our setup involving clustering algorithms considers applying k-Means in the presence of outliers (Hautamäki et al., 2005; Chawla and Gionis, 2013; Gupta et al., 2017). The goal here is to correctly discover and ignore the outliers so that they do not wrongly shift the cluster locations to form a bad partition of the true data. This objective is also important in our setup as not ignoring outliers (or distractors) will wrongly shift the prototypes and negatively influence classification performance. Our model is also related to Gaussian mixture models (McLachlan and Basford, 1988) on the more probabilistic side, but we drop the mixing coefficients since in few-shot learning we typically have an equal number of examples per class.

Our contribution to the semi-supervised learning and clustering literature is to go beyond the classical setting of training and evaluating within a single dataset, and consider the setting where we must learn to transfer from a set of training classes $\mathcal{C}_{\text{train}}$ to a new set of test classes $\mathcal{C}_{\text{test}}$.

4.3 Experiments

4.3.1 Adapting the Datasets for Semi-Supervised Learning

We evaluate the performance of our model on three datasets: Omniglot Lake et al. (2011), *mini*-ImageNet Vinyals et al. (2016) and *tiered*-ImageNet Ren et al. (2018b). For each dataset, we first create an additional split to separate the images of each class into disjoint labeled and unlabeled sets. For Omniglot and *tiered*-ImageNet we sampled 10% of the images of each class to form the labeled split. The remaining 90% can only be used in the unlabeled portion of episodes. For *mini*-ImageNet we instead used 40% of the data for the labeled split and the remaining 60% for the unlabeled, since we noticed that 10% was too small to achieve reasonable performance and avoid overfitting. We report the average classification scores over 10 random splits of labeled and unlabeled portions of the training set, with uncertainty computed in standard error (standard deviation divided by the square root of the total number of splits).

We would like to emphasize that due to this labeled/unlabeled split, we are using strictly less label information than in the previously published work on these datasets. Because of this, we do not expect our results to match the published numbers, which should instead be interpreted as an upper bound for the performance of the semi-supervised models defined in this work.

Table 4.1: Omniglot 1-shot classification results. "w/ D" denotes "with distractors" where the unlabeled images contain irrelevant classes.

Models	Acc.	Acc. w/ D
Supervised	94.62 ± 0.09	94.62 ± 0.09
Semi-Supervised Inference	97.45 ± 0.05	95.08 ± 0.09
Soft k-Means	97.25 ± 0.10	95.01 ± 0.09
Soft k-Means+Cluster	**97.68 ± 0.07**	97.17 ± 0.04
Masked Soft k-Means	97.52 ± 0.07	**97.30 ± 0.08**

Table 4.2: *mini*-ImageNet 1/5-shot classification results. "w/ D" denotes "with distractors" where the unlabeled images contain irrelevant classes.

Models	1-shot Acc.	5-shot Acc.	1-shot Acc w/ D	5-shot Acc. w/ D
Supervised	43.61 ± 0.27	59.08 ± 0.22	43.61 ± 0.27	59.08 ± 0.22
Semi-Supervised Inference	48.98 ± 0.34	63.77 ± 0.20	47.42 ± 0.33	62.62 ± 0.24
Soft k-Means	**50.09 ± 0.45**	**64.59 ± 0.28**	**48.70 ± 0.32**	**63.55 ± 0.28**
Soft k-Means+Cluster	49.03 ± 0.24	63.08 ± 0.18	**48.86 ± 0.32**	61.27 ± 0.24
Masked Soft k-Means	**50.41 ± 0.31**	**64.39 ± 0.24**	**49.04 ± 0.31**	62.96 ± 0.14

Episode construction then is performed as follows. For a given dataset, we create a training episode by first sampling N classes uniformly at random from the set of training classes $\mathcal{C}_{\text{train}}$. We then sample K images from the labeled split of each of these classes to form the support set, and M images from the unlabeled split of each of these classes to form the unlabeled set. Optionally, when including distractors, we additionally sample H other classes from the set of training classes and M images from the unlabeled split of each to act as the distractors. These distractor images are added to the unlabeled set along with the unlabeled images of the N classes of interest (for a total of $MN + MH$ images). The query portion of the episode is comprised of a fixed number of images from the labeled split of each of the N chosen classes. Test episodes are created analogously, but with the N classes (and optionally the H distractor classes) sampled from $\mathcal{C}_{\text{test}}$. Note that we used $M = 5$ for training and $M = 20$ for testing, thus also measuring the ability of the models to generalize to a larger unlabeled set size. We also used $H = N = 5$, i.e. used 5 classes for both the labeled classes and the distractor classes.

In each dataset, we compare our three semi-supervised models with two baselines. The first baseline, referred to as "Supervised" in our tables, is an ordinary Prototypical Network that is trained in a purely supervised way on the labeled split of each dataset. The second baseline, referred to as "Semi-Supervised Inference", uses the embedding function learned by this supervised Prototypical Network, but performs semi-supervised refinement of the prototypes at inference time using a step of Soft k-Means refinement. This is to be contrasted with our semi-supervised models that perform this refinement both at training time and at test time, therefore learning a different embedding function. We evaluate each model in two settings: one where all unlabeled examples belong to the classes of interest, and a more challenging one that includes distractors. Details of the model hyperparameters can be found in Appendix A.4.

Results for Omniglot, *mini*-ImageNet and *tiered*-ImageNet are given in Tables 4.1, 4.2 and 4.3, respectively, while Figure 4.4 shows the performance of our models on *tiered*-ImageNet (our largest

Table 4.3: *tiered*-ImageNet 1/5-shot classification results. "w/ D" denotes "with distractors" where the unlabeled images contain irrelevant classes.

Models	1-shot Acc.	5-shot Acc.	1-shot Acc. w/ D	5-shot Acc. w/ D
Supervised	46.52 ± 0.52	66.15 ± 0.22	46.52 ± 0.52	66.15 ± 0.22
Semi-Supervised Inference	50.74 ± 0.75	69.37 ± 0.26	48.67 ± 0.60	67.46 ± 0.24
Soft k-Means	51.52 ± 0.36	**70.25 ± 0.31**	49.88 ± 0.52	68.32 ± 0.22
Soft k-Means+Cluster	**51.85 ± 0.25**	69.42 ± 0.17	**51.36 ± 0.31**	67.56 ± 0.10
Masked Soft k-Means	**52.39 ± 0.44**	**69.88 ± 0.20**	**51.38 ± 0.38**	**69.08 ± 0.25**

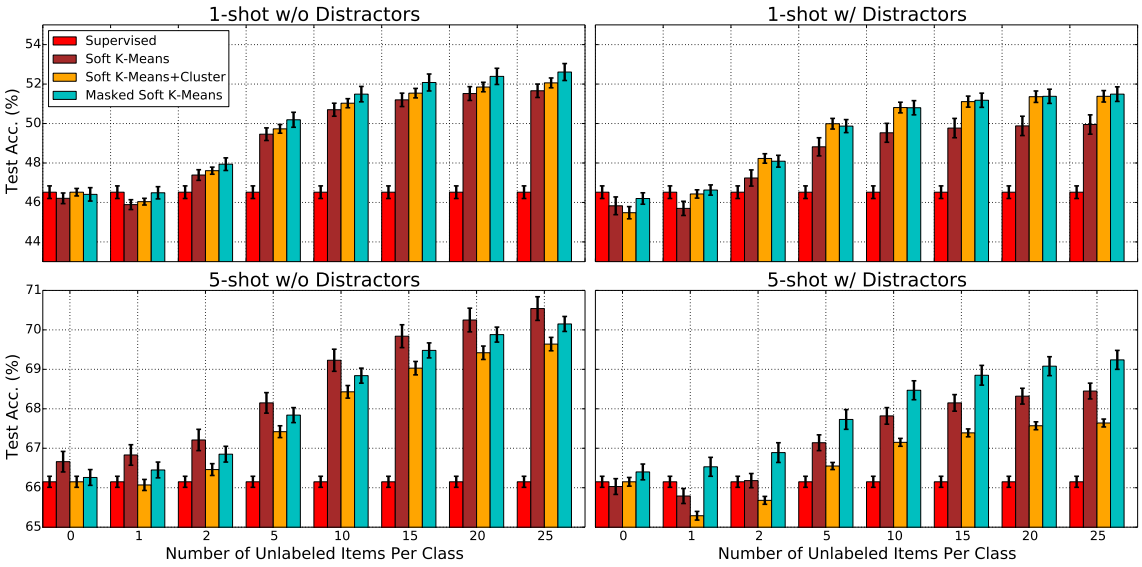


Figure 4.4: Model Performance on *tiered*-ImageNet with different number of unlabeled items during test time.

dataset) using different values for M (number of items in the unlabeled set per class).

Across all three benchmarks, at least one of our proposed models outperform the baselines, demonstrating the effectiveness of our semi-supervised meta-learning procedure. In particular, both Soft k-Means and Soft k-Means+Cluster perform well on 1-shot non-distractor settings, and Soft k-Means is better on 5-shot, as it considers the most unlabeled examples. With the presence of distractors, Masked Soft k-Means shows the most robust performance across all three datasets, reaching comparable performance compared to the upper bound of the best under non-distractor settings.

From Figure 4.4, we observe clear improvement in test accuracy when the number grows from 0 to 25. Note that our models were trained with $M = 5$ and thus are showing an ability to extrapolate in generalization. This confirms that, through meta-training, the models learned to acquire a better representation that will be more helpful after semi-supervised refinement.

Note that the wins obtained in our semi-supervised learning are super-additive. Consider the case of the simple k-Means model on 1-shot without Distractors. Training only on labeled examples while incorporating the unlabeled set during test time produces an advantage of 4.2% (50.7-46.5), and incorporating unlabeled examples during both training and test yields a win of 5.9% (52.4-46.5).

4.4 Discussion and Conclusion

In this work, we propose a novel semi-supervised few-shot learning paradigm, where an unlabeled set is added to each episode. We also extend the setup to more realistic situations where the unlabeled set has classes not belonging to the labeled classes. We propose several novel extensions of Prototypical Networks, and they show consistent improvements under semi-supervised settings compared to our baselines.

There have been a lot of follow-up literature on semi-supervised few-shot learning since the publication of our original paper (Ren et al., 2018b), which this chapter is based on. Our paradigm has also inspired transductive few-shot learning (Liu et al., 2019b), where the model is directly leveraging the query examples as unlabeled data. Liu et al. (2019a) also proposed a label propagation network for reasoning with unlabeled data with a few labeled examples. Self-training (Li et al., 2019b) uses model predicted labels, and transductive finetuning (Dhillon et al., 2020) uses the entropy loss to regularize the model prediction to be more confident. Boudiaf et al. (2020) proposes to use information maximization as the finetuning objective, which minimizes conditional entropy while maximizing marginal entropy. SIB (Hu et al., 2020) minimizes an empirical Bayes objective unrolled with synthetic gradients, and prototype rectification (Liu et al., 2020a) combines the support prototypes and pseudo-labeled unlabeled prototypes.

Our original paper also proposed *tiered*-ImageNet, a larger dataset for few-shot learning with a certain level of a categorical split. However, it is often found that pretrained representations typically perform competitively on few-shot learning by simply training a linear classifier on top (Chen et al., 2019; Tian et al., 2020). Since then, Meta-Dataset (Triantafillou et al., 2020) was proposed and it features multiple domains which encourages more recent few-shot meta-learners to generalize across multiple domains at test time (Requeima et al., 2019; Dvornik et al., 2020; Triantafillou et al., 2021; Liu et al., 2021).

This chapter investigates the ability to recognize distractors, which can be extended naturally to open-set concept learning. Wong et al. (2019) proposed to use a similar prototype-based clustering objective to perform clustering that helps identify unknown instances. Liu et al. (2019c) combined the solution of many-shot, few-shot, and open-set classification under a unified framework. Future visual class learning algorithms should no longer depend on the class composition of the dataset and should be able to recognize new classes at any time.

Looking forward, it seems that the fields of semi-supervised learning and few-shot learning are converging, since unlabeled data benefit not only few-shot inference but also representation learning. Semi-supervised learning using data augmentation can dramatically reduce the number of labels needed for training (Tarvainen and Valpola, 2017; Athiwaratkun et al., 2018; Berthelot et al., 2019, 2020; Xie et al., 2020a,b; Sohn et al., 2020), to an extent that is almost like few-shot learning with a lot of unlabeled data. In the meantime, unsupervised and self-supervised learning has also been applied to few-shot learning with improved performance (Gidaris et al., 2019; Chen et al., 2020c; Su et al., 2020; Medina et al., 2020). Perhaps soon there will be no separation between representation learning and few-shot learning, and we will see a unified framework that learns from a long-tailed distribution of data from the open world.

Chapter 5

Learning from Noisy and Imbalanced Data

Although few-shot learning allows us to recognize new classes, training is typically done with a class-balanced dataset. This means we can easily sample episodes or mini-batches without worrying about the long-tailed class distribution. In applications such as object detection in the context of self-driving, the vast majority of the training data is composed of standard vehicles but models also need to recognize rarely seen classes such as emergency vehicles or animals with very high accuracy. This will sometimes lead to biased training models that do not perform well in practice. Unlike standard few-shot learning datasets, real-world data distribution makes learning rarely seen classes with a few examples much more challenging.

Another popular type of problem in the natural world is label noise. To train a reasonable supervised deep model, we ideally need a large dataset with high-quality labels, which require many passes of expensive human quality assurance. Although coarse labels are cheap and of high availability, the presence of noise will hurt the model performance, e.g. Zhang et al. (2017) has shown that a standard CNN can fit any ratio of label flipping noise in the training set and eventually leads to poor generalization performance. Another related example of noisy data is the *distractor* examples that we introduced in Chapter 4, and in order to be robust from distractors, we need to suppress them when performing clustering for inference.

Training set biases and data noise can sometimes be addressed with dataset resampling (Chawla et al., 2002), i.e. choosing the correct proportion of labels to train a network on, or more generally by assigning a weight to each example and minimizing a weighted training loss. The example weights are typically calculated based on the training loss, as in many classical algorithms such as AdaBoost (Freund and Schapire, 1997), hard negative mining (Malisiewicz et al., 2011), self-paced learning (Kumar et al., 2010), and other more recent work (Chang et al., 2017b; Jiang et al., 2018).

However, there exist two contradicting ideas in training loss-based approaches. In noisy label problems, we prefer examples with smaller training losses as they are more likely to be clean images; yet in class imbalance problems, algorithms such as hard negative mining (Malisiewicz et al., 2011) prioritize examples with higher training loss since they are more likely to be the minority class. In cases when the training set is both imbalanced and noisy, these existing methods would have the wrong model assumptions. In fact, without a proper definition of an unbiased test set, solving the

training set bias problem is inherently ill-defined. As the model cannot distinguish right from wrong, stronger regularization can usually work surprisingly well in certain synthetic noise settings. Here we argue that in order to learn general forms of training set biases, it is necessary to have a small unbiased validation to guide training. It is actually not uncommon to construct a dataset with two parts—one relatively small but very accurately labeled, and another massive but coarsely labeled. Coarse labels can come from inexpensive crowdsourcing services or weakly supervised data (Cordts et al., 2016; Russakovsky et al., 2015; Chen and Gupta, 2015). Such a small validation set can also be thought of as the *support* set in a few-shot learning episode.

Different from existing loss-based approaches, in this chapter we follow a meta-learning paradigm and model the most basic assumption instead: *the best example weighting should minimize the loss of a set of unbiased clean validation examples that are consistent with the evaluation procedure*. Traditionally, validation is performed at the end of the training, which can be prohibitively expensive if we treat the example weights as some hyperparameters to optimize; to circumvent this, we perform validation at *every* training iteration to dynamically determine the example weights of the current batch. Towards this goal, we propose an online reweighting method that leverages an additional small validation set and adaptively assigns importance weights to examples in every iteration. We experiment with both class imbalance and corrupted label problems and find that our approach significantly increases the robustness to training set biases.

5.1 Learning to Reweight Examples

In this section, we derive our model from a meta-learning objective towards an online approximation that can fit into any regular supervised training. We also give a practical implementation suitable for any deep networks.

5.1.1 From a Meta-Learning Objective to an Online Approximation

Let (x, y) be an input-target pair, and $\{(x_i, y_i), 1 \leq i \leq N\}$ be the training set. We assume that there is a small unbiased and clean validation set $\{(x_i^v, y_i^v), 1 \leq i \leq M\}$, and $M \ll N$. Hereafter, we will use superscript v to denote validation set and subscript i to denote the i^{th} data. We also assume that the training set contains the validation set; otherwise, we can always add this small validation set into the training set and leverage more information during training.

Let $\Phi(x, \theta)$ be our neural network model, and θ be the model parameters. We consider a loss function $C(\hat{y}, y)$ to minimize during training, where $\hat{y} = \Phi(x, \theta)$.

In standard training, we aim to minimize the expected loss for the training set: $\frac{1}{N}\sum_{i=1}^{N} C(\hat{y}_i, y_i) = \frac{1}{N}\sum_{i=1}^{N} f_i(\theta)$, where each input example is weighted equally, and $f_i(\theta)$ stands for the loss function associating with data x_i. Here we aim to learn a reweighting of the inputs, where we minimize a weighted loss:

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta), \tag{5.1}$$

with w_i unknown upon beginning. Note that $\{w_i\}_{i=1}^{N}$ can be understood as training hyperparameters,

and the optimal selection of w is based on its validation performance:

$$w^* = \arg\min_{w, w \geq 0} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta^*(w)). \tag{5.2}$$

It is necessary that $w_i \geq 0$ for all i, since minimizing the negative training loss can usually result in unstable behavior.

Online approximation Calculating the optimal w_i requires two nested loops of optimization, and every single loop can be very expensive. The motivation of our approach is to adapt online w through a single optimization loop. For each training iteration, we inspect the descent direction of some training examples locally on the training loss surface and reweight them according to their similarity to the descent direction of the validation loss surface.

For most training of deep neural networks, SGD or its variants are used to optimize such loss functions. At every step t of training, a mini-batch of training examples $\{(x_i, y_i), 1 \leq i \leq n\}$ is sampled, where n is the mini-batch size, $n \ll N$. Then the parameters are adjusted according to the descent direction of the expected loss on the mini-batch. Let's consider vanilla SGD:

$$\theta_{t+1} = \theta_t - \alpha \nabla \left(\frac{1}{n} \sum_{i=1}^{n} f_i(\theta_t) \right), \tag{5.3}$$

where α is the step size.

We want to understand what would be the impact of training example i towards the performance of the validation set at training step t. Following a similar analysis to Koh and Liang (2017), we consider perturbing the weighting by ϵ_i for each training example in the mini- batch,

$$f_{i,\epsilon}(\theta) = \epsilon_i f_i(\theta), \tag{5.4}$$

$$\hat{\theta}_{t+1}(\epsilon) = \theta_t - \alpha \nabla \sum_{i=1}^{n} f_{i,\epsilon}(\theta) \Big|_{\theta=\theta_t}. \tag{5.5}$$

We can then look for the optimal ϵ^* that minimizes the validation loss f^v locally at step t:

$$\epsilon_t^* = \arg\min_{\epsilon} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta_{t+1}(\epsilon)). \tag{5.6}$$

Unfortunately, this can still be quite time-consuming. To get a cheap estimate of w_i at step t, we take a single gradient descent step on a mini-batch of validation samples wrt. ϵ_t, and then rectify the output to get a non-negative weighting:

$$u_{i,t} = -\eta \frac{\partial}{\partial \epsilon_{i,t}} \frac{1}{m} \sum_{j=1}^{m} f_j^v(\theta_{t+1}(\epsilon)) \Big|_{\epsilon_{i,t}=0}, \tag{5.7}$$

$$\tilde{w}_{i,t} = \max(u_{i,t}, 0). \tag{5.8}$$

where η is the descent step size on ϵ. The gradients is evaluated at $\epsilon = 0$ since we would like to make sure that the new weighting is at least better than zero weighting (i.e. not updating on the

mini-batch).

To match the original training step size, in practice, we can consider normalizing the weights of all examples in a training batch so that they sum up to one. In other words, we choose to have a hard constraint within the set $\{w : \|w\|_1 = 1\} \cup \{0\}$. Zero is included here in case we do not want to update on any examples from the mini-batch.

$$w_{i,t} = \frac{\tilde{w}_{i,t}}{(\sum_j \tilde{w}_{j,t}) + \delta(\sum_j \tilde{w}_{j,t})}, \tag{5.9}$$

where $\delta(\cdot)$ is to prevent the degenerate case when all w_i's in a mini-batch are zeros, i.e. $\delta(a) = 1$ if $a = 0$, and equals to 0 otherwise. Without the batch-normalization step, it is possible that the algorithm modifies its effective learning rate of the training progress, and our one-step look-ahead may be too conservative in terms of the choice of learning rate (Wu et al., 2018a). Moreover, by normalizing the learning rate, we effectively removed the meta-learning rate parameter η since the update will no longer depend on it.

5.1.2 Example: Learning to Reweight Examples in a Multi-Layer Perceptron Network

In this section, we study how to compute $w_{i,t}$ in a multi-layer perceptron (MLP) network. One of the core steps is to compute the gradients of the validation loss wrt. the local perturbation ϵ, We can consider a multi-layered network where we have parameters for each layer $\theta = \{\theta_l\}_{l=1}^L$, and at every layer, we first compute z_l the pre-activation, a weighted sum of inputs to the layer, and afterwards we apply a non-linear activation function σ to obtain $\tilde{z}_l$ the post-activation:

$$z_l = \theta_l^\top \tilde{z}_{l-1}, \tag{5.10}$$

$$\tilde{z}_l = \sigma(z_l). \tag{5.11}$$

During backpropagation, let g_l be the gradients of loss wrt. z_l, and the gradients wrt. θ_l is given by $\tilde{z}_{l-1} g_l^\top$. We can further express the gradients towards ϵ as a sum of local dot products.

$$\begin{aligned} &\frac{\partial}{\partial \epsilon_{i,t}} \mathbb{E}\left[f^v(\theta_{t+1}(\epsilon)) \Big|_{\epsilon_{i,t}=0} \right] \\ \propto &- \frac{1}{m} \sum_{j=1}^{m} \frac{\partial f_j^v(\theta)}{\partial \theta}\Big|_{\theta=\theta_t}^\top \frac{\partial f_i(\theta)}{\partial \theta}\Big|_{\theta=\theta_t} \\ = &- \frac{1}{m} \sum_{j=1}^{m} \sum_{l=1}^{L} ({\tilde{z}_{j,l-1}^v}^\top \tilde{z}_{i,l-1})({g_{j,l}^v}^\top g_{i,l}). \end{aligned} \tag{5.12}$$

Eq. 5.12 suggests that the meta-gradient on ϵ is composed of the sum of the products of two terms: $z^\top z^v$ and $g^\top g^v$. The first dot product computes the similarity between the training and validation inputs to the layer, while the second computes the similarity between the training and validation gradient directions. In other words, suppose that a pair of training and validation examples are very similar, and they also provide similar gradient directions, then this training example is helpful and should be up-weighted, and conversely, if they provide opposite gradient directions, this training

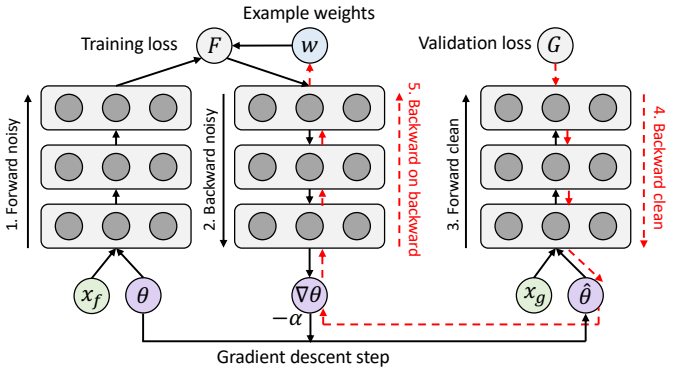


Figure 5.1: Computation graph of our algorithm in a deep neural network, which can be efficiently implemented using second order automatic differentiation.

example is harmful and should be downweighed.

5.1.3 Implementation using Automatic Differentiation

In an MLP and a CNN, the unnormalized weights can be calculated based on the sum of the correlations of layerwise activation gradients and input activations. In more general networks, we can leverage automatic differentiation techniques to compute the gradient of the validation loss wrt. the example weights of the current batch. As shown in Figure 5.1, to get the gradients of the example weights, one needs to first unroll the gradient graph of the training batch, and then use backward-on-backward automatic differentiation to take a second-order gradient pass (see Step 5 in Figure 5.1). We list detailed step-by-step pseudo-code in Algorithm 2. This implementation can be generalized to any deep learning architecture and can be very easily implemented using popular deep learning frameworks such as TensorFlow (Abadi et al., 2016).

Algorithm 2 Learning to Reweight Examples using Automatic Differentiation

Require: θ_0, $\mathcal{D}_f$, $\mathcal{D}_g$, n, m

Ensure: θ_T

for $t = 0$ ... $T - 1$ **do**
  $\{X_f, y_f\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_f, n)$
  $\{X_g, y_g\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_g, m)$
  $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
  $\epsilon \leftarrow 0$; $l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
  $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
  $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
  $\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$
  $l_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$
  $\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$
  $\tilde{w} \leftarrow \max(-\nabla\epsilon, 0)$; $w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$
  $\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$
  $\nabla\theta_t \leftarrow \text{BackwardAD}(\hat{l}_f, \theta_t)$
  $\theta_{t+1} \leftarrow \text{OptimizerStep}(\theta_t, \nabla\theta_t)$
end for

Training time Our automatic reweighting method will introduce a constant factor of overhead. First, it requires two full forward and backward passes of the network on training and validation respectively, and then another backward-on-backward pass (Step 5 in Figure 5.1), to get the gradients to the example weights, and finally, a backward pass to minimize the reweighted objective. In modern networks, a backward-on-backward pass usually takes about the same time as a forward pass, and therefore compared to regular training, our method needs approximately 3× training time; it is also possible to reduce the batch size of the validation pass for speedup. We expect that it is worthwhile to spend the extra time to avoid the irritation of choosing early stopping, finetuning schedules, and other hyperparameters.

5.2 Related Work

The idea of weighting each training example has been well studied in the literature. Boosting algorithms such as AdaBoost (Freund and Schapire, 1997), select harder examples to train subsequent classifiers. Similarly, hard example mining (Malisiewicz et al., 2011), downsamples the majority class and exploits the most difficult examples. Focal loss (Lin et al., 2017) adds a soft weighting scheme that emphasizes harder examples.

Hard examples are not always preferred in the presence of outliers and noise processes. Robust loss estimators typically downweigh examples with high loss. In self-paced learning (Kumar et al., 2010), example weights are obtained through optimizing the weighted training loss encouraging learning easier examples first. In each step, the learning algorithm jointly solves a mixed-integer program that iterates optimizing over model parameters and binary example weights. Various regularization terms on the example weights have since been proposed to prevent overfitting and trivial solutions

of assigning weights to be all zeros (Kumar et al., 2010; Ma et al., 2017; Jiang et al., 2015). Wang et al. (2017) proposed a Bayesian method that infers the example weights as latent variables. More recently, Jiang et al. (2018) proposed to use a meta-learning LSTM to output the weights of the examples based on the training loss. Reweighting examples is also related to curriculum learning (Bengio et al., 2009), where the model reweights among many available tasks. Similar to self-paced learning, typically it is beneficial to start with easier examples.

One crucial advantage of reweighting examples is robustness against training set bias. There has also been a multitude of prior studies on class imbalance problems, including using dataset resampling (Chawla et al., 2002; Dong et al., 2017), cost-sensitive weighting (Ting, 2000; Khan et al., 2015), and structured margin-based objectives (Huang et al., 2016). Meanwhile, the noisy label problem has been thoroughly studied by the learning theory community (Natarajan et al., 2013; Angluin and Laird, 1988) and practical methods have also been proposed (Reed et al., 2014; Sukhbaatar and Fergus, 2015; Xiao et al., 2015; Azadi et al., 2016; Goldberger and Ben-Reuven, 2017; Li et al., 2017; Jiang et al., 2018; Vahdat, 2017; Hendrycks et al., 2018). In addition to corrupted data, Koh and Liang (2017); Muñoz-González et al. (2017) demonstrate the possibility of a dataset adversarial attack (i.e. dataset poisoning).

Our algorithm also resembles MAML (Finn et al., 2017) by taking one gradient descent step on the meta-objective for each iteration. However, different from these meta-learning approaches, our reweighting method does not have any additional hyper-parameters and circumvents an expensive offline training stage. Hence, our method can work in an online fashion during regular training. Our method is also inspired by earlier literature on tuning learning rate online, such as stochastic meta descent (Schraudolph, 1999) and incremental delta-bar-delta (Sutton, 1992). Our clean validation set also shares some resemblance to the support set idea in standard few-shot learning. Similar to the setup introduce in the previous chapter, we are using the clean validation set (support set) to infer the labels of the noisy set (unlabeled set with distractors).

5.3 Experiments

To test the effectiveness of our reweighting algorithm, we designed both class imbalance and noisy label settings, and a combination of both, on standard MNIST and CIFAR benchmarks for image classification using deep CNNs.

5.3.1 MNIST Data Imbalance Experiments

We use the standard MNIST handwritten digit classification dataset and subsample the dataset to generate a class imbalance binary classification task. We select a total of 5,000 images of size 28×28 on class 4 and 9, where 9 dominates the training data distribution. We train a standard LeNet on this task and we compare our method with a suite of commonly used tricks for class imbalance: 1) Proportion weights each example by the inverse frequency 2) Resample samples a class-balanced mini-batch for each iteration 3) Hard Mining selects the highest loss examples from the majority class and 4) Random is a random example weight baseline that assigns weights based on a rectified

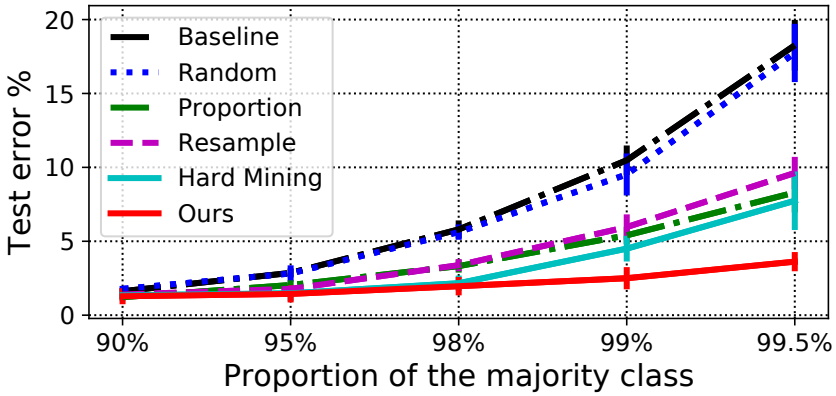


Figure 5.2: MNIST 4-9 binary classification error using a LeNet on imbalanced classes. Our method uses a small balanced validation split of 10 examples.

Gaussian distribution:

$$w_i^{\text{rnd}} = \frac{\max(z_i, 0)}{\sum_i \max(z_i, 0)}, \quad \text{where } z_i \sim \mathcal{N}(0, 1). \tag{5.13}$$

To make sure that our method does not have the privilege of training on more data, we split the balanced validation set of 10 images directly from the training set. The network is trained with SGD with a learning rate of 1e-3 and a mini-batch size of 100 for a total of 8,000 steps.

Figure 5.2 plots the test error rate across various imbalance ratios averaged from 10 runs with random splits. Note that our method significantly outperforms all the baselines. With a class imbalance ratio of 200:1, our method only reports a small increase of error rate around 2%, whereas other methods suffer terribly under this setting. Compared with resampling and hard negative mining baselines, our approach does not throw away samples based on their classes or training losses—as long as a sample is helpful towards the validation loss, it will be included as a part of the training loss.

5.3.2 CIFAR Noisy Label Experiments

Reweighting algorithms can also be useful on datasets where the labels are noisy. We study two settings of label noise here:

- UniformFlip: All label classes can uniformly flip to any other label classes, which is the most studied in the literature.
- BackgroundFlip: All label classes can flip to a single background class. This noise setting is very realistic. For instance, human annotators may not have recognized all the positive instances, while the rest remain in the background class. This is also a combination of label imbalance and label noise since the background class usually dominates the label distribution.

We compare our method with prior work on the noisy label problem.

- Reed, proposed by Reed et al. (2014), is a bootstrapping technique where the training target is a convex combination of the model prediction and the label.

Table 5.1: CIFAR UniformFlip under 40% noise ratio using a WideResNet-28-10 model. Test accuracy shown in percentage. Top rows use only noisy data, and bottom uses additional 1000 clean images. "FT" denotes finetuning on clean data.

Model	CIFAR-10	CIFAR-100
Baseline	67.97 ± 0.62	50.66 ± 0.24
Reed-Hard	69.66 ± 1.21	51.34 ± 0.17
S-Model	70.64 ± 3.09	49.10 ± 0.58
MentorNet	76.6	56.9
Random	86.06 ± 0.32.	58.01 ± 0.37
Using 1,000 clean images		
Clean Only	46.64 ± 3.90	9.94 ± 0.82
Baseline +FT	78.66 ± 0.44	54.52 ± 0.40
MentorNet +FT	78	59
Random +FT	86.55 ± 0.24	58.54 ± 0.52
Ours	**86.92 ± 0.19**	**61.34 ± 2.06**

- S-Model, proposed by Goldberger and Ben-Reuven (2017), adds a fully connected softmax layer after the regular classification output layer to model the noise transition matrix.
- MentorNet, proposed by Jiang et al. (2018), is an RNN-based meta-learning model that takes in a sequence of loss values and outputs the example weights. We compare numbers reported in their paper with a base model that achieves similar test accuracy under 0% noise.

In addition, we propose two simple baselines: 1) Random, which assigns weights according to a rectified Gaussian (see Eq. 5.13); 2) Weighted, designed for BackgroundFlip, where the model knows the oracle noise ratio for each class and reweights the training loss proportional to the percentage of clean images of that label class.

Clean validation set For UniformFlip, we use 1,000 clean images in the validation set; for BackgroundFlip, we use 10 clean images per label class. Since our method uses information from the clean validation, for a fair comparison, we conduct an additional finetuning on the clean data based on the pre-trained baselines. We also study the effect on the size of the clean validation set in an ablation study.

Hyper-validation set For monitoring training progress and tuning baseline hyperparameters, we split out another 5,000 hyper-validation set from the 50,000 training images. We also corrupt the hyper-validation set with the same noise type.

Experimental details For Reed model, we use the best β reported in Reed et al. (2014) ($\beta = 0.8$ for hard bootstrapping and $\beta = 0.95$ for soft bootstrapping). For the S-Model, we explore two versions to initialize the transition weights: 1) a smoothed identity matrix; 2) in background flip experiments we consider initializing the transition matrix with the confusion matrix of a pre-trained baseline model (S-Model +Conf). We find baselines can easily overfit the training noise, and therefore we also study early stopped versions of the baselines to provide a stronger comparison. In contrast, we find early stopping not necessary for our method.

Table 5.2: CIFAR BackgroundFlip under 40% noise ratio using a ResNet-32 model. Test accuracy shown in percentage. Top rows use only noisy data, and bottom rows use additional 10 clean images per class. "+ES" denotes early stopping; "FT" denotes finetuning.

Model	CIFAR-10	CIFAR-100
Baseline	59.54 ± 2.16	37.82 ± 0.69
Baseline +ES	64.96 ± 1.19	39.08 ± 0.65
Random	69.51 ± 1.36	36.56 ± 0.44
Weighted	79.17 ± 1.36	36.56 ± 0.44
Reed Soft +ES	63.47 ± 1.05	38.44 ± 0.90
Reed Hard +ES	65.22 ± 1.06	39.03 ± 0.55
S-Model	58.60 ± 2.33	37.02 ± 0.34
S-Model +Conf	68.93 ± 1.09	46.72 ± 1.87
S-Model +Conf +ES	79.24 ± 0.56	54.50 ± 2.51
Using 10 clean images per class		
Clean Only	15.90 ± 3.32	8.06 ± 0.76
Baseline +FT	82.82 ± 0.93	54.23 ± 1.75
Baseline +ES +FT	85.19 ± 0.46	55.22 ± 1.40
Weighted +FT	85.98 ± 0.47	53.99 ± 1.62
S-Model +Conf +FT	81.90 ± 0.85	53.11 ± 1.33
S-Model +Conf +ES +FT	85.86 ± 0.63	55.75 ± 1.26
Ours	**86.73 ± 0.48**	**59.30 ± 0.60**

To make our results comparable with the ones reported in MentorNet and to save computation time, we exchange their Wide ResNet-101-10 with a Wide ResNet-28-10 (WRN-28-10) (Zagoruyko and Komodakis, 2016) with dropout 0.3 as our base model in the UniformFlip experiments. We find that test accuracy differences between the two base models are within 0.5% on CIFAR datasets under 0% noise. In the BackgroundFlip experiments, we use a ResNet-32 (He et al., 2016) as our base model.

We train the models with SGD with momentum, at an initial learning rate 0.1 and a momentum 0.9 with mini-batch size 100. For ResNet-32 models, the learning rate decays $\times 0.1$ at 40K and 60K steps, for a total of 80K steps. For WRN and early stopped versions of ResNet-32 models, the learning rate decays at 40K and 50K steps, for a total of 60K steps. Under regular 0% noise settings, our base ResNet-32 gets 92.5% and 68.1% classification accuracy on CIFAR-10 and 100, and the WRN-28-10 gets 95.5% and 78.2%. For the finetuning stage, we run extra 5K steps of training on the limited clean data.

We report the average test accuracy for 5 different random splits of clean and noisy labels, with 95% confidence interval in Table 5.1 and 5.2. The background classes for the 5 trials are [0, 1, 3, 5, 7] (CIFAR-10) and [7, 12, 41, 62, 85] (CIFAR-100).

5.3.3 Results

The first result that draws our attention is that "Random" performs surprisingly well on the UniformFlip benchmark, outperforming all historical methods that we compared. Given that its performance is comparable with Baseline on BackgroundFlip and MNIST class imbalance, we hypothesize that random example weights act as a strong regularizer and under which the learning objective on UniformFlip is still consistent.

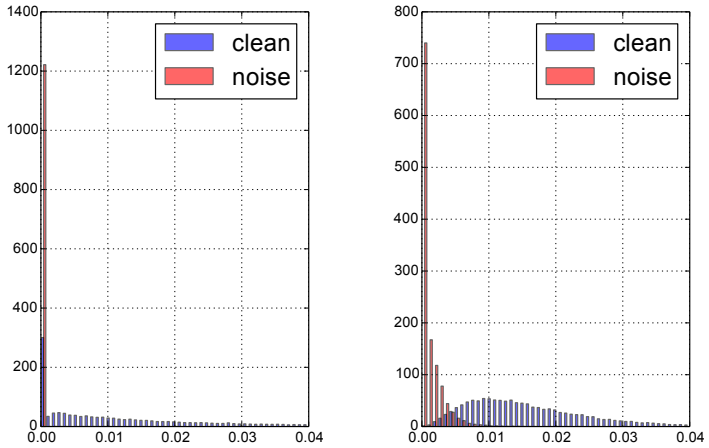


Figure 5.3: Example weights distribution on BackgroundFlip. Left: a hyper-validation batch, with randomly flipped background noises. Right: a hyper-validation batch containing only on a single label class, with flipped background noises, averaged across all non-background classes.

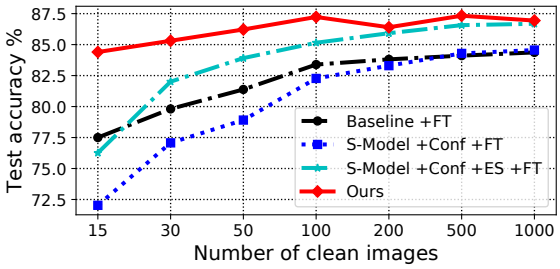


Figure 5.4: Effect of the number of clean imaged used, on CIFAR-10 with 40% of data flipped to label 3. "ES" denotes early stopping.

Regardless of the strong baseline, our method ranks the top on both UniformFlip and BackgroundFlip, showing our method is less affected by the changes in the noise type. On CIFAR-100, our method wins more than 3% compared to the state-of-the-art method.

Understanding the reweighting mechanism It is beneficial to understand how our reweighting algorithm contributes to learning more robust models during training. First, we use a pre-trained model (trained at half of the total iterations without learning rate decay) and measure the example weight distribution of a randomly sampled batch of validation images, which the model has never seen. As shown in the left figure of Figure 5.3, our model correctly pushes most noisy images to zero weights. Secondly, we conditioned the input mini-batch to be a single non-background class and randomly flip 40% of the images to the background, and we would like to see how well our model can distinguish clean and noisy images. As shown in Figure 5.3 right, the model is able to reliably detect images that are flipped to the background class.

Robustness to overfitting noise Throughout experimentation, we find baseline models can easily overfit to the noise in the training set. For example, shown in Table 5.2, applying early stopping

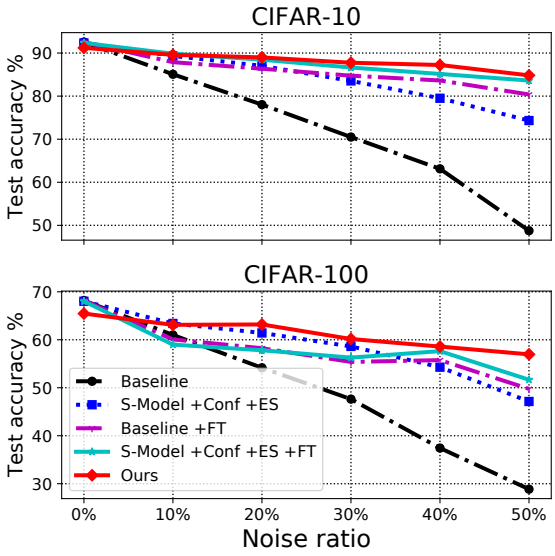


Figure 5.5: Model test accuracy on imbalanced noisy CIFAR experiments across various noise levels using a base ResNet-32 model. "ES" denotes early stopping, and "FT" denotes finetuning.

("ES") helps the classification performance of "S-Model" by over 10% on CIFAR-10. Figure 5.6 compares the final confusion matrices of the baseline and the proposed algorithm, where a large proportion of noise transition probability is cleared in the final prediction. Figure 5.7 shows training curves on the BackgroundFlip experiments. After the first learning rate decay, both "Baseline" and "S-Model" quickly degrade their validation performance due to overfitting, while our model remains the same validation accuracy until termination. Note that here "S-Model" knows the oracle noise ratio in each class, and this information is not available in our method.

Impact of the noise level We would like to investigate how strongly our method can perform on a variety of noise levels. As shown in Figure 5.5, our method only drops 6% accuracy when the noise ratio increased from 0% to 50%; whereas the baseline has dropped more than 40%. At 0% noise, our method only slightly underperforms baseline. This is reasonable since we are optimizing on the validation set, which is strictly a subset of the full training set, and therefore suffers from its own subsample bias.

Size of the clean validation set When the size of the clean validation set grows larger, finetuning on the validation set will be a reasonable approach. Here, we make an attempt to explore the tradeoff and understand when finetuning becomes beneficial. Figure 5.4 plots the classification performance when we varied the size of the clean validation on BackgroundFlip. Surprisingly, using 15 validation images for all classes only results in a 2% drop in performance, and the overall classification performance does not grow after having more than 100 validation images. In comparison, we observe a significant drop in performance when only finetuning on these 15 validation images for the baselines, and the performance catches up around using 1,000 validation images (100 per class). This phenomenon suggests that in our method the clean validation acts more like a regularizer rather than a data source for parameter finetuning, and potentially our method can be complementary with

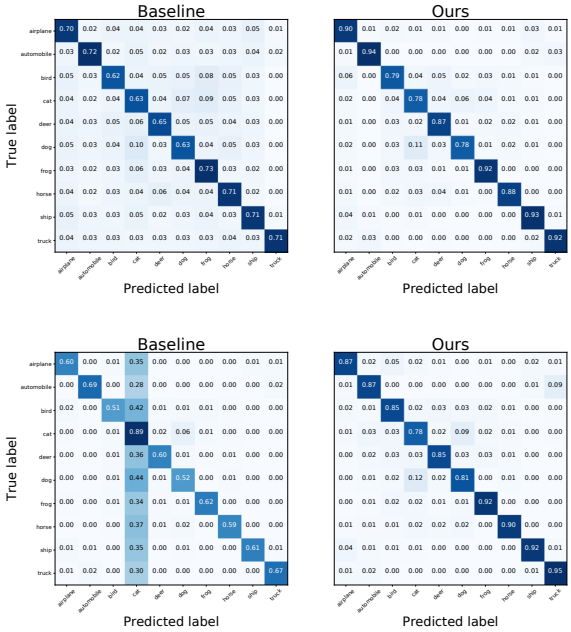


Figure 5.6: Confusion matrices on CIFAR-10 UniformFlip (top) and BackgroundFlip (bottom)

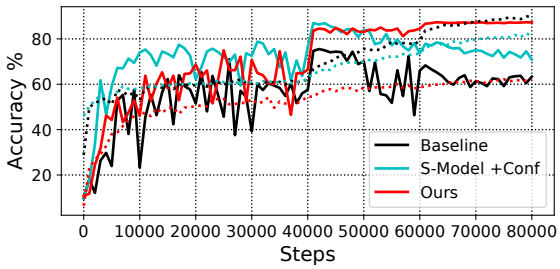


Figure 5.7: Training curve of a ResNet-32 on CIFAR-10 BackgroundFlip under 40% noise ratio. Solid lines denote validation accuracy and dotted lines denote training. Our method is less prone to label noise overfitting.

the finetuning-based method when the size of the clean set grows larger.

5.4 Discussion and Conclusion

In this chapter, we propose an online meta-learning algorithm for reweighting training examples and training more robust deep learning models. While various types of training set biases exist and manually designed reweighting objectives have their own bias, our automatic reweighting algorithm shows superior performance dealing with class imbalance, noisy labels, and both. Our method can be directly applied to any deep learning architecture and is expected to train end-to-end without any additional hyperparameter search. Validating on every training step is a novel setting and we show that it has links with model regularization, which can be a fruitful future research direction.

Since the publication of the original conference paper version of this chapter, there have been many related papers that improve upon our experiment results. On the data imbalance side, later works

have benchmarked their algorithms with larger datasets such as long-tailed CIFAR (Cui et al., 2019), long-tailed ImageNet (Liu et al., 2019c), long-tailed Places (Liu et al., 2019c), and iNaturalist (Horn et al., 2018). Cui et al. (2019) propose a class-balanced loss that weights the loss by $(1-\beta)/(1-\beta^{n_y})$ where β is a hyperparameter and n_y is the number of samples in the groundtruth class y. Cao et al. (2019) propose Label-Distribution-Aware Margin (LDAM) loss (for the y-th class):

$$\mathcal{L}_{\text{LDAM}} = -\log \frac{e^{z_y - \Delta_y}}{e^{z_y - \Delta_y} + \sum_{j \neq y} e^{z_j}}, \tag{5.14}$$

where Δ_j is the margin for the j-th class

$$\Delta_j = \frac{C}{n_j^{\frac{1}{4}}}, \tag{5.15}$$

and C is a hyperparamter and n_j is the number of examples in class j. In contrast to these analytical forms, Jamal et al. (2020) followed our data-driven weighting approach, but modified our framework in the following ways. First, it decouples the example weights into class-wise weights and example-wise weights. The class-wise portion follows Cui et al. (2019) and the example-wise portion follows our framework, except that they found removing zero-initialization, clipping and normalization can lead to better performance. Lastly, they optionally train with the focal loss (Lin et al., 2017) or the LDAM loss (Cao et al., 2019) in addition to the standard cross-entropy loss.

On the noisy label side, JointOptimization (Tanaka et al., 2018) iteratively updates the model parameters and the per-example pseudo labels by minimizing the KL divergence between the pseudo-label and the model prediction. Co-teaching (Han et al., 2018) maintains two networks and uses the output prediction of each network to supervise the other network. Li et al. (2019a) proposed a similar inner loop structure to ours. It uses a teacher network to slowly update the model parameters through an exponential moving average. At each iteration, it launches a batch of student networks that each perturbs the label and performs inner loop updates, and in the end, it minimizes the KL divergence between the students' predictions and the teacher's prediction. This is leveraging the fact that clean labels are often preferred common update directions and are more likely being learned by the slow teacher network.

Arazo et al. (2019) found that one can use a beta-mixture model to fit the loss into clean and noisy clusters, and weight model prediction by the probability of noisy label. They also use mix-up (Zhang et al., 2018) that makes a linear combination of two input and label pairs, which have been found to provide more robustness against label noise, since it is more likely that one of the labels among the two is still correct. P-correlation (Yi and Wu, 2019) learns a per-example pseudo label by jointly minimizing the entropy loss, the classification loss, and the compatibility loss. The entropy loss encourages the pseudo labels to be confident, and the classification loss asks the model to predict the pseudo labels, and lastly, the compatibility loss regularizes the pseudo labels to be compatible with the noisy labels. DivideMix (Li et al., 2020) combines ideas from Arazo et al. (2019) for obtaining clean vs. noisy clusters, and MixMatch (Berthelot et al., 2019) for semi-supervised learning, by treating noisy examples as unlabeled data.

As more and more recent progress is being made, we should be cautious about the additional assumptions of these proposed solutions. State-of-the-art methods for imbalanced data now use new weighting functions that are typically based on the number of samples per class. This might be

vulnerable to noisy labels that accidentally flip into a minority class. On the other hand, solutions for noisy labels are now increasingly relying on the assumption that the noisy data comes from the same closed set of classes as the clean data just like standard semi-supervised learning. This can make models prone to outliers and unseen classes like distractors. The pseudo labeling approach (Tanaka et al., 2018) can implicitly maximize the marginal entropy (Bridle et al., 1991), which encourages a balanced distribution in the pseudo label space. All of these come from the fact that the algorithms are benchmarked on standard image classification datasets with curated label space. In contrast, the original goal of this chapter is to bring new insights on the challenges of both imbalanced data and noisy labels, and to a large extent, it still remains a challenge today to jointly solve the two problems together. However, the reliance on a clean and balanced validation set in our model does seem quite onerous, and in fact, many of the subsequent noisy label approaches do not need a validation set. Therefore, in the future, there may appear new algorithms that jointly solve the two problems without using a clean validation set.

Lastly, our proposed method with an amortized one-step inner loop structure has also inspired other meta-learning algorithms for interleaving the updates of regular and meta parameters. Meta-Weight-Net (Shu et al., 2019) enhances our method by introducing a meta-network in replacement of per-example weights. Algorithms with a similar style also appear in neural architecture search (Liu et al., 2019a), multi-source domain adaptation (Li and Hospedales, 2020), and semi-supervised learning (Ren et al., 2020b). Undoubtedly, one major benefit of this type of online meta-learning is the massive saving of training time by allowing the learning of representations and fast adaptation to happen at the same time. Therefore, in the future, it will be exciting to anticipate new extensions that can make meta-learning more efficient and scalable.

Chapter 6

Online and Incremental Learning with Context

In Chapter 3 and 4, we examined variants of few-shot learning, where training is episodic. Within an isolated episode, a set of new classes is introduced with a limited number of labeled examples per class—the *support* set—followed by evaluation on an unlabeled *query* set. While this setup has inspired the development of a multitude of meta-learning algorithms which can be trained to rapidly learn novel classes with a few labeled examples, the classes learned are not carried over to future episodes. Although incremental learning and continual learning methods address the case where classes are carried over, the episodic construction of these frameworks seems artificial: in our daily lives, we do not learn new objects by grouping them with five other new objects, process them together, and then move on.

Therefore, these existing paradigms such as few-shot learning or continual learning do not well approximate the naturalistic conditions that humans and artificial agents encounter as they wander within a physical environment. Consider, for example, learning and remembering peoples' names in the course of daily life. We tend to see people in a given environment—work, home, gym, etc. We tend to repeatedly revisit those environments, with different environment base rates, nonuniform environment transition probabilities, and nonuniform base rates of encountering a given person in a given environment. We need to recognize when we do not know a person, and we need to learn to recognize them the next time we encounter them. We are not always provided with a name, but we can learn in a semi-supervised manner. And every training trial is itself an evaluation trial as we repeatedly use existing knowledge and acquire new knowledge.

To break the rigid, artificial structure of continual and few-shot learning, in this chapter we propose a new paradigm, *online contextualized few-shot learning*, where environments are revisited and the total number of novel object classes increases over time. Crucially, model evaluation happens on each trial, very much like the setup in online learning. When encountering a new class, the learning algorithm is expected to indicate that the class is "new," and it is then expected to recognize subsequent instances of the class once a label has been provided.

When learning continually in such a dynamic environment, contextual information can guide learning and remembering. Any structured sequence provides *temporal context*: the instances encountered recently are predictive of instances to be encountered next. In natural environments,

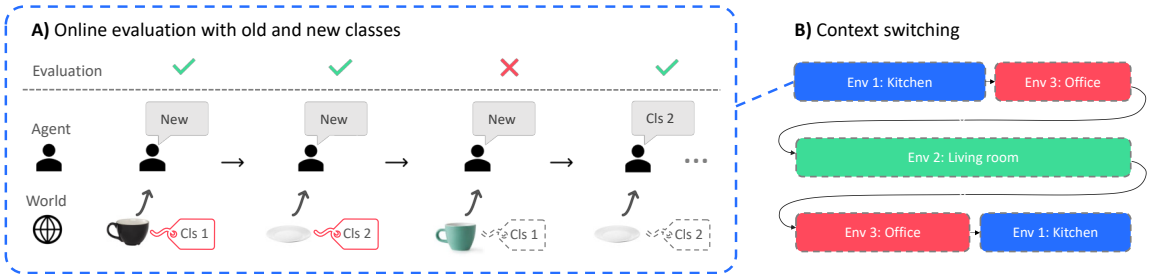


Figure 6.1: Online contextualized few-shot learning. A) Our setup is similar to online learning, where there is no separate testing phase; model training and evaluation happen *at the same time*. The input at each time step is an (image, class-label) pair. The number of classes grows *incrementally* and the agent is expected to answer "new" for items that have not yet been assigned labels. Sequences can be *semi-supervised*; here the label is not revealed for every input item (labeled/unlabeled shown by red solid/grey dotted boxes). The agent is evaluated on the correctness of all answers. The model obtains learning signals only on labeled instances, and is correct if it predicts the label of previously-seen classes, or 'new' for new ones. B) The overall sequence switches between different *learning environments*. While the environment ID is *hidden* from the agent, inferring the current environment can help solve the task.

spatial context—information in the current input weakly correlated with the occurrence of a particular class—can be beneficial for retrieval as well. For example, we tend to see our boss in an office setting, not in a bedroom setting. Human memory retrieval benefits from both spatial and temporal context (Howard, 2017; Kahana, 2012). In our online few-shot learning setting, we provide a spatial context in the presentation of each instance and temporal structure to sequences, enabling an agent to learn from both spatial and temporal context. Besides developing and experimenting on a toy benchmark using handwritten characters (Lake et al., 2011), we also propose a new large-scale benchmark for online contextualized few-shot learning derived from indoor panoramic imagery (Chang et al., 2017a). In the toy benchmark, temporal context can be defined by the co-occurrence of character classes. In the indoor environment, the context—temporal and spatial—is a natural by-product as the agent wandering in between different rooms.

In this chapter, we also propose a model that can exploit contextual information, called *contextual prototypical memory* (*CPM*), which incorporates an RNN to encode contextual information and a separate prototype memory to remember previously learned classes (see Figure 6.4). This model obtains significant gains on few-shot classification performance compared to models that do not retain a memory of the recent past. We compare to classic few-shot algorithms extended to an online setting, and CPM consistently achieves the best performance.

The main contributions of this chapter are as follows. First, we define an *online contextualized few-shot learning (OC-FSL)* setting to mimic naturalistic human learning. Second, we build three datasets: 1) *RoamingOmniglot* is based on handwritten characters from Omniglot (Lake et al., 2011); 2) *RoamingImageNet* is based on images from ImageNet (Russakovsky et al., 2015); and 3) *RoamingRooms* is our new few-shot learning dataset based on indoor imagery (Chang et al., 2017a), which resembles the visual experience of a wandering agent. Third, we benchmark classic FSL methods and also explore our CPM model, which combines the strengths of RNNs for modeling temporal context and Prototypical Networks (Snell et al., 2017) for memory consolidation and rapid learning.

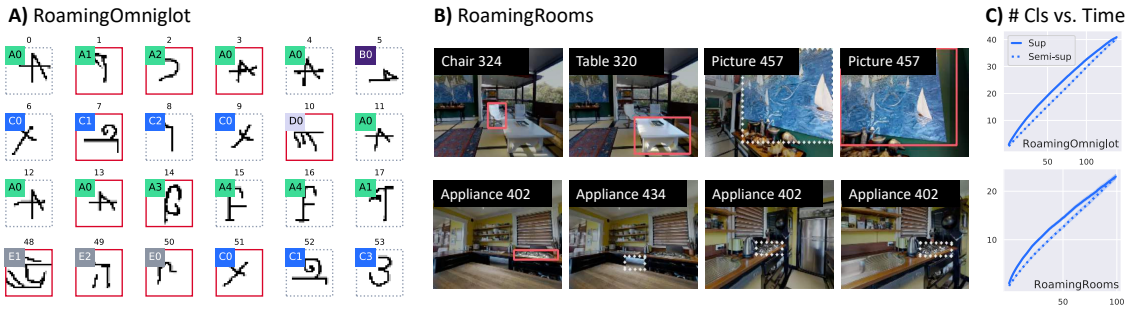


Figure 6.2: Sample online contextualized few-shot learning sequences. A) RoamingOmniglot. Red solid boxes denote labeled examples of Omniglot handwritten characters, and dotted boxes denote unlabeled ones. Environments are shown in colored labels in the top left corner. B) Image frame samples of a few-shot learning sequence in our RoamingRooms dataset collected from a random walking agent. The task here is to recognize and classify novel instance IDs in the home environment. Here the groundtruth instance masks/bounding boxes are provided. C) The growth of total number of labeled classes in a sequence for RoamingOmniglot (top) and RoamingRooms (bottom).

6.1 Online Contextualized Few-Shot Learning

In this section, we introduce our new online contextualized few-shot learning (OC-FSL) setup in the form of a sequential decision problem, and then introduce our new benchmark datasets.

Continual few-shot classification as a sequential decision problem: In traditional few-shot learning, an episode is constructed by a support set S and a query set Q. A few-shot learner f is expected to predict the class of each example in the query set $\mathbf{x}^Q$ based on the support set information: $\hat{y}^Q = f(\mathbf{x}^Q; (\mathbf{x}_1^S, y_1^S), \ldots, (\mathbf{x}_N^S, y_N^S))$, where $\mathbf{x} \in \mathbb{R}^D$, and $y \in [1 \ldots K]$. This setup is not a natural fit for continual learning, since it is unclear when to insert a query set into the sequence.

Inspired by the online learning literature, we can convert continual few-shot learning into a sequential decision problem, where every input example is also part of the evaluation: $\hat{y}_t = f(\mathbf{x}_t; (\mathbf{x}_1, \tilde{y}_1), \ldots, (\mathbf{x}_{t-1}, \tilde{y}_{t-1}))$, for $t = 1 \ldots T$, where $\tilde{y}$ here further allows that the sequence of inputs to be semi-supervised: $\tilde{y}$ equals y_t if labeled, or otherwise -1. The setup in Santoro et al. (2016) and Kaiser et al. (2017) is similar; they train RNNs using such a temporal sequence as input. However, their evaluation relies on a "query set" at the end. We instead evaluate online while learning.

Figure 6.1-A illustrates these features, using an example of an input sequence where an agent is learning about new objects in a kitchen. The model is rewarded when it correctly predicts a known class or when it indicates that the item has yet to be assigned a label.

Contextualized environments: Typical continual learning consists of a sequence of tasks, and models are trained sequentially for each task. This feature is also preserved in many incremental learning settings (Rebuffi et al., 2017). For instance, the split-CIFAR task divides CIFAR-100 into 10 learning environments, each with 10 classes, presented sequentially. In our formulation, the sequence returns to earlier environments (see Figure 6.1-B), which enables assessment of long-term durability of knowledge. Although the ground-truth environment identity is not provided, we structure the task such that the environment itself provides contextual cues which can constrain the correct class label. *Spatial* cues in the input help distinguish one environment from another. *Temporal* cues are

implicit because the sequence tends to switch environments infrequently, allowing recent inputs to be beneficial in guiding the interpretation of the current input.

RoamingOmniglot: The Omniglot dataset (Lake et al., 2011) contains 1623 handwritten characters from 50 different alphabets. We split the alphabets into 31 for training, 5 for validation, and 13 for testing. We augment classes by 90-degree rotations to create 6492 classes in total. Each contextualized few-shot learning image sequence contains 150 images, drawn from a random sample of 5-10 alphabets, for a total of 50 classes per sequence. These classes are randomly assigned to 5 different environments; within an environment, the characters are distributed according to a Chinese restaurant process (CRP) (Aldous, 1985) to mimic the distribution of naturally occurring objects with varying number of appearances. We chose CRP since there are only a few images per class, whereas an alternative would be the Pitman-Yor process (Pitman and Yor, 1997), which aims to model long-tailed distributions. Moreover, we switch between environments using a Markov switching process; i.e., at each step there is a constant probability of switching to another environment. An example sequence is shown in Figure 6.2-A.

RoamingRooms: As none of the current few-shot learning datasets provides the natural online learning experience that we would like to study, we created our own dataset using simulated indoor environments. We formulate this as a few-shot instance learning problem, which could be a use case for a home robot: it needs to quickly recognize and differentiate novel object instances, and large viewpoint variations can make this task challenging (see examples in Figure 6.2-B). There are over 7,000 unique instance classes in the dataset, making it suitable for meta-learning approaches.

Our dataset is derived from the Matterport3D dataset (Chang et al., 2017a) with 90 indoor worlds captured using panoramic depth cameras. We split these into 60 worlds for training, 10 for validation, and 20 for testing. MatterSim (Anderson et al., 2018) is used to load the simulated world and collect camera images and HabitatSim (Savva et al., 2019) is used to project instance segmentation labels onto 2D image space. We created a random walking agent to collect the virtual visual experience. For each viewpoint in the random walk, we randomly sample one object from the image sensor and highlight it with the available instance segmentation, forming an input *frame*. Each viewpoint provides environmental context—the other objects present in the room with the highlighted object.

Statistics of the RoamingRooms are included in Table 6.1, in comparison to other few-shot and continual learning datasets. Note that since RoamingRooms is collected from a simulated environment, with 90 indoor worlds consisting of 1.2K panorama images and 1.22M video frames. The dataset contains about 6.9K random walk sequences with a maximum of 200 frames per sequence. For training we randomly crop 100 frames to form a training sequence. There are 7.0K unique instance classes. We also include Table 6.2 to compare existing continual and few-shot learning paradigms.

Figure 6.3-A shows the object instance distribution. We see a strong temporal correlation, as 30% of the time the same instance appears in the next frame (Figure 6.3-B), but there is also a significant proportion of revisits. On average, there are three different viewpoints per 100-image sequence (Figure 6.3-C).

RoamingImageNet: To make the classification problem more challenging, we also report results on RoamingImageNet, which uses the same online sequence sampler as RoamingOmniglot but

Table 6.1: Continual & few-shot learning datasets

	Images	Sequences	Classes	Content
Permuted MNIST (Lecun et al., 1998)	60K	-	-	Hand written digits
Omniglot (Lake et al., 2011)	32.4K	-	1.6K	Hand written characters
CIFAR-100 (Krizhevsky, 2009)	50K	-	100	Common objects
mini-ImageNet (Vinyals et al., 2016)	50K	-	100	Common objects
tiered-ImageNet (Ren et al., 2018b)	779K	-	608	Common objects
OpenLORIS (She et al., 2019)	98K	-	69	Small table-top obj.
CORe50 (Lomonaco and Maltoni, 2017)	164.8K	11	50	Hand-held obj.
RoamingRooms (Ours)	1.22M	6.9K	7.0K	General indoor instances

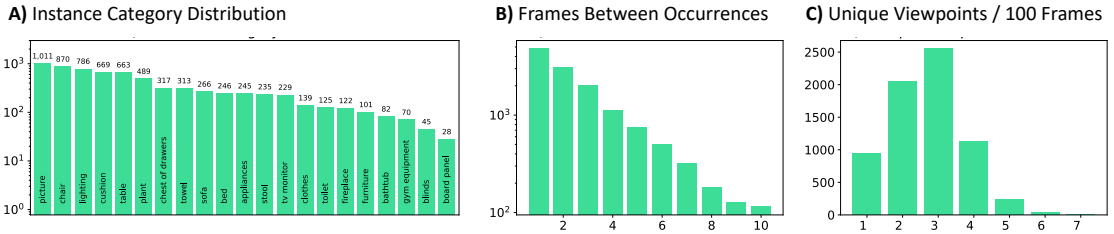


Figure 6.3: Statistics for our RoamingRooms dataset. Plots show a natural long tail distribution of instances grouped into categories. An average sequence has 3 different view points. Sequences are highly correlated in time but revisits are not uncommon.

applied on the *tiered*-ImageNet dataset (Ren et al., 2018b), a subset of the original ImageNet dataset (Russakovsky et al., 2015). It contains 608 classes with a total of 779K images of size 84×84. We use the same split of classes as Ren et al. (2018b). The high-level categories are used for sampling different "environments" just like the notion of alphabets in RoamingOmniglot.

6.2 Contextual Prototypical Memory Networks

In the online contextualized few-shot learning setup, the few-shot learner can potentially improve by modeling the temporal context. Metric learning approaches (Snell et al., 2017) typically ignore temporal contextual relations and directly compare the similarity between training and test samples. Gradient-based approaches (Javed and White, 2019), on the other hand, have the ability to adapt to new contexts, but they do not naturally handle new and unlabeled examples. Inspired by the contextual binding theory of human memory (Yonelinas et al., 2019), we propose a simple yet effective approach that uses an RNN to transmit spatiotemporal context and control signals to a prototype memory (Figure 6.4).

Prototype memory: We start describing our model with the prototype memory, which is an online version of the Prototypical Network (or *ProtoNet*) (Snell et al., 2017). ProtoNet can be viewed as a knowledge base memory, where each object class k is represented by a prototype vector $\mathbf{p}[k]$, computed as the mean vector of all the support instances of the class in a sequence. It can also be applied to our task of online few-shot learning naturally, with some modifications. Suppose that at time-step t we have already stored a few classes in the memory, each represented by their current

Table 6.2: Comparison of past FSL and CL paradigms vs. our online contextualized FSL (OC-FSL).

Tasks	Few Shot	Semi-sup. Supp. Set	Continual	Online Eval.	Predict New	Soft Context Switch
Incremental Learning (IL) (Rebuffi et al., 2017)	○	○	●	◒	○	○
Few-shot (FSL) (Vinyals et al., 2016)	●	○	○	○	○	○
Incremental FSL (Ren et al., 2019)	●	○	◒	○	○	○
Cls. Incremental FSL (Tao et al., 2020)	●	○	●	◒	○	○
Semi-supv. FSL (Ren et al., 2018b)	●	●	○	○	●	○
MOCA (Harrison et al., 2019)	●	○	●	○	○	◒
Online Mixture (Jerfel et al., 2019)	●	○	●	○	○	◒
Online Meta (Javed and White, 2019)	●	○	●	○	○	○
Continual FSL* (Antoniou and Storkey, 2019)	●	○	●	○	○	○
OSAKA* (Caccia et al., 2020)	●	○	●	●	◒	●
OC-FSL (Ours)	●	●	●	●	●	●

* denotes concurrent work.

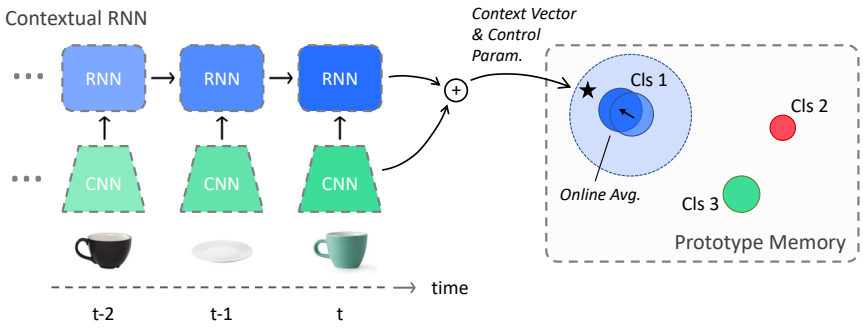


Figure 6.4: Contextual prototypical memory. Temporal contextual features are extracted from an RNN. The prototype memory stores one vector per class and does online averaging. Examples falling outside the radii of all prototypes are classified as "new."

prototype $\mathbf{p}_t[k]$, and we would like to query the memory using the input feature $\mathbf{h}_t$. We model our prototype memory as $\hat{y}_{t,k} = \text{softmax}(-d_{\mathbf{m}_t}(\mathbf{h}_t, \mathbf{p}_t[k]))$, (e.g. squared Euclidean distance or cosine dissimilarity) parameterized by a vector $\mathbf{m}_t$ that scales each dimension with a Hadamard product. To predict whether an example is of a new class, we can use a separate *novelty* output $\hat{u}_t^r$ with sigmoid activation, similar to the approach introduced in Ren et al. (2018b), where β_t^r and γ_t^r are yet-to-be-specified thresholding hyperparameters (the superscript r stands for read):

$$\hat{u}_t^r = \text{sigmoid}((\min_k d_{\mathbf{m}_t}(\mathbf{h}_t, \mathbf{p}_t[k]) - \beta_t^r)/\gamma_t^r). \tag{6.1}$$

Memory consolidation with online prototype averaging: Traditionally, ProtoNet uses the average representation of a class across all support examples. Here, we must be able to adapt the prototype memory incrementally at each step. Fortunately, we can recover the computation of a ProtoNet by performing a simple online averaging:

$$c_t = c_{t-1} + 1; \quad A(\mathbf{h}_t; \mathbf{p}_{t-1}, c_t) = \frac{1}{c_t}\left(\mathbf{p}_{t-1}c_{t-1} + \mathbf{h}_t\right), \tag{6.2}$$

where $\mathbf{h}$ is the input feature, and $\mathbf{p}$ is the prototype, and c is a scalar indicating the number of examples that have been added to this prototype up to time t. The online averaging function A can also be made more flexible to allow more plasiticity, modeled by a *gated averaging unit* (GAU):

$$A_{\text{GAU}}(\mathbf{h}_t; \mathbf{p}_{t-1}) = (1 - f_t) \cdot \mathbf{p}_{t-1} + f_t \cdot \mathbf{h}_t, \quad \text{where} \quad f_t = \text{sigmoid}(W_f[\mathbf{h}_t, \mathbf{p}_{t-1}] + b_f) \in \mathbb{R}. \tag{6.3}$$

When the current example is unlabeled, $\tilde{y}_t$ is encoded as -1, and the model's own prediction $\hat{y}_t$ will determine which prototype to update; in this case, the model must also determine a strength of belief, $\hat{u}_t^w$, that the current unlabeled example should be treated as a new class. Given $\hat{u}_t^w$ and $\hat{y}_t$, the model can then update a prototype:

$$\hat{u}_t^w = \text{sigmoid}((\min_k d_{\mathbf{m}_t}(\mathbf{h}_t, \mathbf{p}_t[k]) - \beta_t^w)/\gamma_t^w), \tag{6.4}$$

$$\Delta[k]_t = \underbrace{\mathbb{1}[\tilde{y}_t = k]}_{\text{Supervised}} + \underbrace{\hat{y}_{t,k}(1 - \hat{u}_t^w)\mathbb{1}[\tilde{y}_t = -1]}_{\text{Unsupervised}}, \tag{6.5}$$

$$c[k]_t = c[k]_{t-1} + \Delta[k]_t, \tag{6.6}$$

$$\mathbf{p}[k]_t = A(\mathbf{h}_t \Delta[k]_t; \mathbf{p}[k]_{t-1}, c[k]_t), \quad \text{or} \quad \mathbf{p}[k]_t = A_{\text{GAU}}(\mathbf{h}_t \Delta[k]_t; \mathbf{p}[k]_{t-1}). \tag{6.7}$$

As-yet-unspecified hyperparameters β_t^w and γ_t^w are required (the superscript w is for write). These parameters for the online-updating novelty output $\hat{u}_t^w$ are distinct from β_t^r and γ_t^r in Equation 6.1. The intuition is that for "self-teaching" to work, the model potentially needs to be more conservative in creating new classes (avoiding corruption of prototypes) than in predicting an input as being a new class.

Contextual RNN: Instead of directly using the features from the CNN $\mathbf{h}_t^{\text{CNN}}$ as input features to the prototype memory, we would also like to use contextual information from the recent past. Above we introduced threshold hyperparameters β_t^r, γ_t^r, β_t^w, γ_t^w as well as the metric parameter $\mathbf{M}_t$. We let the contextual RNN output these additional control parameters, so that the unknown thresholds and metric function can adapt based on the information in the context. The RNN produces the context vector $\mathbf{h}_t^{\text{RNN}}$ and other control parameters conditioned on $\mathbf{h}_t^{\text{CNN}}$:

$$\left[\mathbf{z}_t, \mathbf{h}_t^{\text{RNN}}, \mathbf{m}_t,\ \beta_t^r,\ \gamma_t^r,\ \beta_t^w,\ \gamma_t^w\right] = \text{RNN}(\mathbf{h}_t^{\text{CNN}}; \mathbf{z}_{t-1}), \tag{6.8}$$

where $\mathbf{z}_t$ is the recurrent state of the RNN, and $\mathbf{m}_t$ is the scaling factor in the dissimilarity score. The context, $\mathbf{h}_t^{\text{RNN}}$, serves as an additive bias on the state vector used for FSL: $\mathbf{h}_t = \mathbf{h}_t^{\text{CNN}} + \mathbf{h}_t^{\text{RNN}}$. This addition operation in the feature space can help contextualize prototypes based on temporal proximity, and is also similar to how the human brain leverages spatiotemporal context for memory storage (Yonelinas et al., 2019).

Loss function: The loss function is computed after an entire sequence ends and all network parameters are learned end-to-end. The loss is composed of two parts. The first is binary cross-entropy (BCE), for telling whether each example has been assigned a label or not, i.e., prediction of new classes, and u_t is the ground-truth binary label. Second we use a multi-class cross-entropy for

classifying among the known ones. We can write down the overall loss function as follows:

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} \lambda \underbrace{[-u_t \log(\hat{u}_t^r) - (1-u_t)\log(1-\hat{u}_t^r)]}_{\text{Binary cross entropy on old vs. new}} + \sum_{k=1}^{K} \underbrace{-\mathbb{1}[y_t = k](1-u_t)\log(\hat{y}_{t,k})}_{\text{Cross entropy on old classes}}. \tag{6.9}$$

Training and evaluation: During training, we sample learning sequences, and for each sequence, we perform one iterative update to minimize the loss function (Eq. 2.10). At the beginning of each sequence, the memory is reset. During training, the model learns from a set of training classes. During test time, the model recognizes new classes that have never been seen during training.

6.3 Related Work

Few-shot learning The model we propose, CPM, lies on the boundary between metric-based (Vinyals et al., 2016; Snell et al., 2017) and memory-based (Santoro et al., 2016) few-shot learning approaches, as we use an RNN to model the temporal context but we also use metric-learning mechanisms and objectives to train. (Triantafillou et al., 2020; Yu et al., 2020) also investigate the setup of having a variable number of support examples in a few-shot episode.

In terms of similar few-shot learning benchmarks, Antoniou et al. (2020) extend few-shot learning to a continual setting based on image sequences, each of which is divided into stages with a fixed number of examples per class followed by a query set. It focuses on more flexible and faster adaptation since the models are evaluated online, and the context is a soft constraint instead of a hard separation of tasks. Moreover, new classes need to be identified as part of the sequence, crucial to any learner's incremental acquisition of knowledge.

Continual learning: Traditionally, continual learning is studied with tasks such as permuted MNIST (Lecun et al., 1998) or split-CIFAR (Krizhevsky, 2009). Recent datasets aim to consider more realistic continual learning, such as CORe50 (Lomonaco and Maltoni, 2017) and OpenLORIS (She et al., 2019). We summarize core features of these continual learning datasets in Appendix B.1. First, both CORe50 and OpenLORIS have relatively few object classes, which makes meta-learning approaches inapplicable; second, both contain images of small objects with minimal occlusion and viewpoint changes; and third, OpenLORIS does not have the desired incremental class learning.

Caccia et al. (2020) proposes a setup to unify continual learning and meta-learning with a similar online evaluation procedure. However, there are several notable differences. First, their models focus on a general loss function without a specific design for predicting new classes; they predict new tasks by examining if the loss exceeds some threshold. Second, the sequences of inputs are fully supervised. Lastly, their benchmarks are based on synthetic task sequences such as Omniglot or tiered-ImageNet, which are less naturalistic than our RoamingRooms dataset.

Online meta-learning: Some existing work builds on early approaches (Thrun, 1998; Schmidhuber, 1987) that tackle continual learning from a meta-learning perspective. Finn et al. (2019) proposes storing all task data in a data buffer; by contrast, Javed and White (2019) proposes to instead learn a good representation that supports such online updates. In Jerfel et al. (2019), a hierarchical Bayesian mixture model is used to address the dynamic nature of continual learning.

Table 6.3: RoamingOmniglot OC-FSL results. Max 5 env, 150 images, 50 cls, with 8×8 occlusion.

Method	Supervised			Semi-supervised		
	AP	1-shot Acc.	3-shot Acc.	AP	1-shot Acc.	3-shot Acc.
LSTM	64.34	61.00 ± 0.22	81.85 ± 0.21	54.34	68.30 ± 0.20	76.38 ± 0.49
DNC	81.30	78.87 ± 0.19	91.01 ± 0.15	81.37	88.56 ± 0.12	93.81 ± 0.26
OML-U	77.38	70.98 ± 0.21	89.13 ± 0.16	66.70	74.65 ± 0.19	90.81 ± 0.34
OML-U++	86.85	88.43 ± 0.14	92.01 ± 0.14	81.39	71.64 ± 0.19	93.72 ± 0.27
O-MN	88.69	84.82 ± 0.15	95.55 ± 0.11	84.39	88.77 ± 0.13	97.28 ± 0.17
O-IMP	90.15	85.74 ± 0.15	96.66 ± 0.09	81.62	88.68 ± 0.13	97.09 ± 0.19
O-PN	90.49	85.68 ± 0.15	96.95 ± 0.09	84.61	88.71 ± 0.13	97.61 ± 0.17
CPM (Ours)	**94.17**	**91.99** ± 0.11	**97.74** ± 0.08	**90.42**	**93.18** ± 0.16	**97.89** ± 0.15

Table 6.4: RoamingRooms OC-FSL results. Max 100 images and 40 classes.

Method	Supervised			Semi-supervised		
	AP	1-shot Acc.	3-shot Acc.	AP	1-shot Acc.	3-shot Acc.
LSTM	45.67	59.90 ± 0.40	61.85 ± 0.45	33.32	52.71 ± 0.38	55.83 ± 0.76
DNC	80.86	82.15 ± 0.32	87.30 ± 0.30	73.49	80.27 ± 0.33	87.87 ± 0.49
OML-U	76.27	73.91 ± 0.37	83.99 ± 0.33	63.40	70.67 ± 0.38	85.25 ± 0.56
OML-U++	88.03	88.32 ± 0.27	89.61 ± 0.29	81.90	84.79 ± 0.31	89.80 ± 0.47
O-MN	85.91	82.82 ± 0.32	89.99 ± 0.26	78.99	80.08 ± 0.34	**92.43** ± 0.41
O-IMP	87.33	85.28 ± 0.31	90.83 ± 0.25	75.36	84.57 ± 0.31	91.17 ± 0.43
O-PN	86.01	84.89 ± 0.31	89.58 ± 0.28	76.36	80.67 ± 0.34	88.83 ± 0.49
CPM (Ours)	**89.14**	**88.39** ± 0.27	**91.31** ± 0.26	**84.12**	**86.17** ± 0.30	91.16 ± 0.44

Connections to the human brain: Our CPM model consists of multiple memory systems, consistent with claims of cognitive neuroscientists of multiple memory systems in the brain. The complementary learning systems (CLS) theory (McClelland et al., 1995) suggests that the hippocampus stores the recent experience and is likely where few-shot learning takes place. However, our model is more closely related to contextual binding theory (Yonelinas et al., 2019), which suggests that long-term encoding of information depends on binding spatiotemporal context, and without this context as a cue, forgetting occurs. Our proposed CPM contains parallels to human brain memory components (Cohen and Squire, 1980). Long-term statistical learning is captured in a CNN that produces a deep embedding. An RNN holds a type of working memory that can retain novel objects and spatiotemporal contexts. Lastly, the prototype memory represents the semantic memory, which consolidates multiple events into a single knowledge vector (Duff et al., 2020). Other deep learning researchers have proposed multiple memory systems for continual learning. In Parisi et al. (2018), the learning algorithm is heuristic and representations come from pretrained networks. In Kemker and Kanan (2018), a prototype memory is used for recalling recent examples, and rehearsal from a generative model allows this knowledge to be integrated and distilled into long-term memory.

6.4 Experiments

In this section, we show experimental results for our online contextualized few-shot learning paradigm, using benchmarks defined in Sec. 6.1, to evaluate our model CPM, and other state-of-the-art methods. For Omniglot, we apply an 8×8 CutOut (Devries and Taylor, 2017) to each image to make the task more challenging.

Table 6.5: RoamingImageNet OC-FSL results. Max 150 images and 50 classes. * denotes CNN pretrained using regular classification.

Method	Supervised			Semi-supervised		
	AP	1-shot Acc.	3-shot Acc.	AP	1-shot Acc.	3-shot Acc.
LSTM*	22.54	28.14 ± 0.20	52.07 ± 0.27	13.50	30.02 ± 0.20	46.95 ± 0.56
DNC*	26.80	33.45 ± 0.19	55.78 ± 0.27	16.50	39.53 ± 0.19	54.10 ± 0.54
OML-U	21.89	15.06 ± 0.14	52.52 ± 0.27	10.16	22.74 ± 0.17	55.81 ± 0.55
O-MN	13.05	20.61 ± 0.15	38.73 ± 0.24	9.32	25.96 ± 0.16	55.32 ± 0.51
O-IMP	14.25	22.92 ± 0.16	41.01 ± 0.25	4.55	20.70 ± 0.15	51.23 ± 0.53
O-PN*	23.10	32.82 ± 0.19	49.98 ± 0.25	15.76	36.69 ± 0.18	55.47 ± 0.53
CPM (Ours)	**34.43**	**40.40** ± 0.21	**60.29** ± 0.26	**24.75**	**44.58** ± 0.21	**58.72** ± 0.53

Implementation details: For RoamingOmniglot, we use the common 4-layer CNN for few-shot learning with 64 channels in each layer. For RoamingImageNet, we also use ResNet-12 with input resolution 84×84 (Oreshkin et al., 2018). For the RoamingRooms, we resize the input to 120×160 and use ResNet-12. To represent the feature of the input image with an attention mask, we concatenate the global average pooled feature with the attention ROI feature, resulting in a 512d feature vector. For the contextual RNN, in both experiments, we used an LSTM (Hochreiter and Schmidhuber, 1997) with a 256d hidden state. The best CPM model is equipped using GAU and cosine similarity for querying prototypes. Logits based on cosine similarity are multiplied with a learned scalar initialized at 10.0 (Oreshkin et al., 2018). We include additional training details in Appendix B.2.

Evaluation metrics: In order to compute a single number that characterizes the learning ability over sequences, we propose to use *average precision* (AP) to evaluate both with respect to old versus new and the specific class predictions. We also compute the "N-shot" accuracy; i.e., the average accuracy after seeing the label N times in the sequence. Note that these accuracy scores only reflect the performance on *known* class predictions.

- **Average precision:** We chose to use AP (average precision or area under the precision-recall curve) as a way of integrating two aspects of performance:

  1. the binary accuracy of whether an instance belongs to a known or unknown class (KU-Assign for short), and
  2. the accuracy of assigning an instance the correct class label given it is from a known class (Class-Assign for short).

  The procedure to calculate AP is as follows. We first sort all the KU-Assign, Class-Assign predictions across all sequences in descending order based on KU-Assign probability, where the high ranked predictions should be known (not novel) classes. For the N top ranked instances in the sorted list, we compute:

  1. precision@N = correct(Class-Assign)@N/N
  2. recall@N = correct(Class-Assign)@N/K,

  where K is the true number of known instances and correct(Class-Assign)@N is the count of the number of correct class assignments among the top N. (The class assignment for an

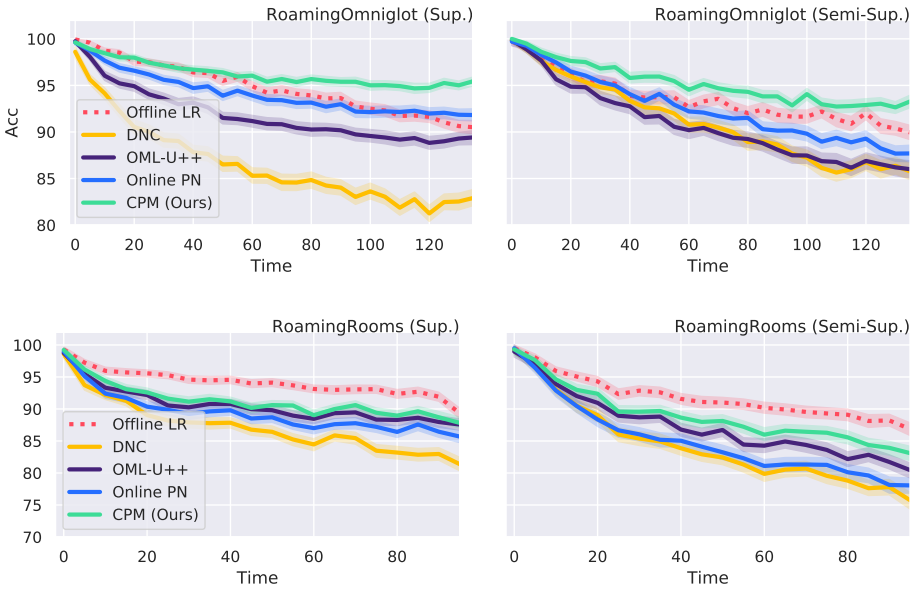


Figure 6.5: Few-shot classification accuracy over time. Top: RoamingOmniglot. Bottom: RoamingRooms. Left: Supervised. Right: Semi-supervised. An offline logistic regression (Offline LR) baseline is also included, using pretrained ProtoNet features. It is trained on all labeled examples except for the one at the current time step.

unknown instance is always incorrect.) To obtain the AP, we compute the integral of the function (y=precision@N, x=recall@N) across all N's.

- **N-shot accuracy:** We define N-shot accuracy as the number of times an instance that has been seen N times thus far in the sequence is classified correctly. We compute the mean and standard error of this over all sequences.

Comparisons: To evaluate the merits of our proposed model, we implement classic few-shot learning and online meta-learning methods. More implementation and training details of these baseline methods can be found in Appendix B.2.

- **OML** (Javed and White, 2019): This is an online version of MAML (Finn et al., 2017). It performs one gradient descent step for each labeled input image, and slow weights are learned via backpropagation through time. On top of OML, we added an unknown predictor $\hat{u}_t = 1 - \max_k \hat{y}_{t,k}$ [1] (**OML-U**). We also found that using cosine classifier without the last layer ReLU is usually better than using the original dot-product classifier, and this improvement is denoted as **OML-U++**.

- **LSTM** (Hochreiter and Schmidhuber, 1997) & **DNC** (Graves et al., 2016): We include RNN methods for comparison as well. Differentiable neural computer (DNC) is an improved version of memory augmented neural network (MANN) (Santoro et al., 2016).

- **Online MatchingNet (O-MN)** (Vinyals et al., 2016), **IMP (O-IMP)** (Allen et al., 2019) & **ProtoNet (O-PN)** (Snell et al., 2017): We used the same negative Euclidean distance as the

[1] We tried a few other ways and this is found to be the best.

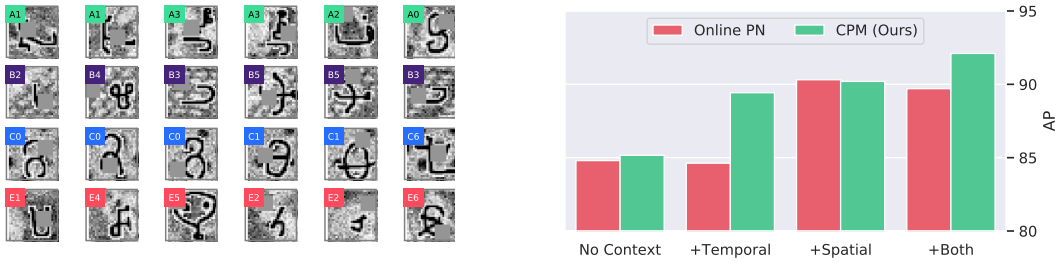


Figure 6.6: Effect of spatiotemporal context. Spatiotemporal context are added separately and together in RoamingOmniglot, by introducing texture background and temporal correlation. Left: Stimuli used for spatial cue of the background environment. Right: Our CPM model benefits from the presence of a temporal context ("+Temporal" and "+Both")

Table 6.6: Effect of forgetting over a time interval on RoamingOmniglot. Average accuracy vs. the number of time steps since the model has last seen the label of a particular class.

	Supervised						**Semi-Supervised**					
Interval	1 - 2	3 - 5	6 - 10	11 - 20	21 - 50	51 - 100	1 - 2	3 - 5	6 - 10	11 - 20	21 - 50	51 - 100
OPN 1-Shot	88.8	86.9	85.2	84.7	83.6	81.1	90.1	88.9	88.4	87.6	87.3	85.1
CPM 1-Shot	**96.1**	**94.0**	**93.0**	**91.6**	**88.2**	**84.6**	**95.9**	**93.8**	**92.8**	**91.8**	**89.4**	**85.7**
OPN 3-Shot	97.2	97.1	96.6	96.7	**96.5**	95.3	97.8	97.3	97.1	**97.8**	**97.7**	**96.8**
CPM 3-Shot	**98.5**	**98.2**	**97.5**	**97.2**	95.4	**95.5**	**98.7**	**97.5**	**97.5**	96.5	96.3	92.9

similarity function for these three metric learning based approaches. In particular, MatchingNet stores all examples and performs nearest neighbor matching, which can be memory inefficient. Note that Online ProtoNet is a variant of our method without the contextual RNN.

Main results: Our main results are shown in Table 6.3, 6.4 and 6.5, including both supervised and semi-supervised settings. Our approach achieves the best performance on AP consistently across all settings. Online ProtoNet is a direct comparison without our contextual RNN and it is clear that CPM is significantly better. Our method is slightly worse than Online MatchingNet in terms of 3-shot accuracy on the RoamingRooms semisupervised benchmark. This can be explained by the fact that MatchingNet stores all past seen examples, whereas CPM only stores one prototype per class. Per timestep accuracy is plotted in Figure 6.5, and the decaying accuracy is due to the increasing number of classes over time. In RoamingOmniglot, CPM is able to closely match or even sometimes surpass the offline classifier, which re-trains at each step and uses all images in a sequence except the current one. This is reasonable as our model is able to leverage information from the current context.

Effect of spatiotemporal context: To answer the question of whether the gain in performance is due to spatiotemporal reasoning, we conduct the following experiment comparing CPM with online ProtoNet. We allow the CNN to have the ability to recognize the context in RoamingOmniglot by adding a texture background image using the Kylberg texture dataset (Kylberg, 2011) (see Figure 6.6 left). As a control, we can also destroy the temporal context by shuffling all the images in a sequence. We train four different models on dataset controls with or without the presence of spatial or temporal context, and results are shown in Figure 6.6. First, both online ProtoNet and CPM benefit from the

Table 6.7: Effect of forgetting over a time interval on RoamingRooms. Average accuracy vs. the number of time steps since the model has last seen the label of a particular class.

	Supervised						**Semi-Supervised**					
	1 - 2	3 - 5	6 - 10	11 - 20	21 - 50	51 - 100	1 - 2	3 - 5	6 - 10	11 - 20	21 - 50	51 - 100
OPN 1-Shot	93.5	89.3	79.4	67.2	60.3	60.1	86.5	83.6	76.3	68.4	64.7	61.5
CPM 1-Shot	**95.7**	**92.2**	**85.7**	**75.2**	**70.0**	**66.4**	**91.0**	**88.7**	**82.9**	**77.0**	**72.2**	**66.5**
OPN 3-Shot	95.1	91.8	85.6	78.2	74.6	73.8	92.6	88.0	85.1	81.1	80.6	76.7
CPM 3-Shot	**96.1**	**93.8**	**87.7**	**81.4**	**79.1**	**78.2**	**94.8**	**91.0**	**86.9**	**83.1**	**82.7**	**79.2**

Table 6.8: Effect of forgetting over a time interval on RoamingImageNet. Average accuracy vs. the number of time steps since the model has last seen the label of a particular class.

	Supervised						**Semi-Supervised**					
	1 - 2	3 - 5	6 - 10	11 - 20	21 - 50	51 - 100	1 - 2	3 - 5	6 - 10	11 - 20	21 - 50	51 - 100
OPN 1-Shot	40.8	35.9	33.0	**30.7**	**27.0**	**21.4**	40.5	37.7	35.9	**33.7**	**31.5**	**28.4**
CPM 1-Shot	**67.5**	**52.9**	**35.5**	24.2	18.3	13.8	**60.4**	**51.3**	**39.5**	26.6	21.8	15.4
OPN 3-Shot	52.5	50.3	**48.8**	**47.2**	**44.4**	**42.3**	57.6	55.1	**54.6**	**52.3**	**52.1**	**49.5**
CPM 3-Shot	**77.8**	**64.5**	46.6	32.9	24.7	17.9	**76.1**	**61.8**	48.6	30.5	24.1	15.6

inclusion of a spatial context. This is understandable as the CNN has the ability to learn spatial cues, which re-confirms our main hypothesis that successful inference of the current context is beneficial to novel object recognition. Second, only our CPM model benefits from the presence of temporal context, and it receives distinct gains from spatial and temporal contexts.

Effect of forgetting: As the number of learned classes increases, we expect the average accuracy to drop. To further investigate this forgetting effect, we measure the average accuracy in terms of the number of time steps the model has last seen the label of a particular class. It is reported in Table 6.6, 6.7, 6.8, where we directly compare CPM and OPN to see the effect of temporal context. CPM is significantly better than OPN on 1-shot within a short interval, which suggests that the contextual RNN makes the recall of the recent past much easier. On RoamingImageNet, OPN eventually surpasses CPM on a longer horizon, and this can be explained by the fact that OPN has more stable prototypes, whereas prototypes in CPM could potentially be affected by the fluctuation of the contextual RNN over a longer horizon.

Ablation studies: We ablate each individual module we introduce. Results are shown in Tables 6.9 and 6.10. Table 6.9 studies different ways we use the RNN, including the context vector $\mathbf{h}^{\text{RNN}}$, the predicted threshold parameters β_t^*, γ_t^*, and the predicted metric scaling vector $_t$. Table 6.10 studies various ways to learn from unlabeled examples, where we separately disable the RNN update, prototype update, and distinct write-threshold parameters β_t^w, γ_t^w (vs. using read-threshold parameters), which makes it robust to potential mistakes made in semi-supervised learning. We verify that each component has a positive impact on the performance.

Embedding visualization: Figure 6.7 shows the learned embedding of each example in Online ProtoNet vs. our CPM model in RoamingOmniglot sequences, where colors indicate environment IDs. In Online ProtoNet, the example features does not reflect the temporal context, and as a result, colors are scattered across the space. By contrast, in the CPM embedding visualization, colors are

Table 6.9: Ablation of CPM architectural components on RoamingOmniglot

Method	$\mathbf{h}^{\text{RNN}}$	β_t^*, γ_t^*	Metric $\mathbf{m}_t$	GAU	Val AP
O-PN					91.22
No $\mathbf{h}^{\text{RNN}}$		✓	✓		92.52
$\mathbf{h}^{\text{RNN}}$ only	✓				93.48
No metric $\mathbf{m}_t$	✓	✓			93.61
No β_t^*, γ_t^*	✓		✓		93.98
$\mathbf{h}_t = \mathbf{h}_t^{\text{RNN}}$	✓	✓	✓		93.70
CPM Avg. Euc	✓	✓	✓		94.08
CPM Avg. Cos	✓	✓	✓		94.57
CPM GAU Euc	✓	✓	✓	✓	94.11
CPM GAU Cos	✓	✓	✓	✓	**94.65**

Table 6.10: Ablation of semi-supervised learning components on RoamingOmniglot

Method	RNN	Prototype	β_t^w, γ_t^w	GAU	Val AP
O-PN					90.83
O-PN		✓			89.10
O-PN		✓	✓		91.22
CPM					92.57
CPM	✓				93.16
CPM	✓	✓			93.20
CPM	✓	✓	✓		94.08
CPM	✓	✓	✓	✓	**94.65**

clustered together and we see a smoother transition of environments in the embedding space.

Control parameters vs. time: Finally we visualize the control parameter values predicted by the RNN in Figure 6.8. We verify that we indeed need two sets of β and γ for read and write operations separately as they learn different values. β^w is smaller than β^r which means that the network is more conservative when writing to prototypes. γ^w grows larger over time, which means that the network prefers a softer slope when writing to prototypes since in the later stage the prototype memory has already stored enough content and it can grow faster, whereas in the earlier stage, the prototype memory is more conservative to avoid embedding vectors to be assigned to wrong clusters.

6.5 Discussion and Conclusion

In this chapter, we proposed online contextualized few-shot learning, OC-FSL, a paradigm for machine learning that emulates a human or artificial agent interacting with a physical world. It combines multiple properties to create a challenging learning task: every input must be classified or flagged as novel, every input is also used for training, semi-supervised learning can potentially improve performance, and the temporal distribution of inputs is non-IID and comes from a generative model in which input and class distributions are conditional on a latent environment with Markovian

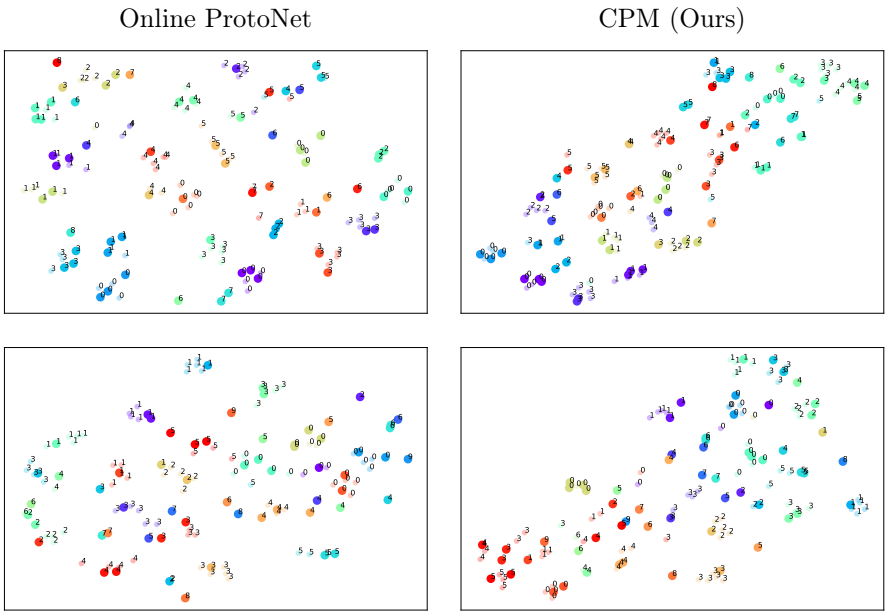


Figure 6.7: Embedding space visualization of RoamingOmniglot sequences using t-SNE (Maaten and Hinton, 2008). Different color denotes different environments. Text labels (relative to each environment) are annotated beside the scatter points. Unlabeled examples shown in smaller circles with lighter colors. Left: Online ProtoNet; Right: CPM. The embeddings learned CPM model shows a smoother transition of classes based on their temporal environments.

transition probabilities. We proposed the RoamingRooms dataset to simulate an agent wandering within a physical world. We also proposed a new model, CPM, which uses an RNN to extract spatiotemporal context from the input stream and to provide control settings to a prototype-based FSL model. In the context of naturalistic domains like RoamingRooms, CPM is able to leverage contextual information to attain performance unmatched by other state-of-the-art FSL methods.

Several interesting open questions can lead to future extensions of this work. First of all, our representation network does not get finetuned at test time to prevent interference. However, in the few-shot learning literature, people have found benefits of finetuning the representations, especially when there is a domain change at test time. Therefore, it remains a question to ask whether we can build finetuning or meta-learned adaptation mechanisms for OC-FSL.

Second, our proposed memory network is slot-based, but in order to fully mimic a biological memory system, the information is more likely to be stored in the connections rather than activations. It would be interesting to explore the possibility of replacing the slot-based memory with an attractor-based one using continuous updates, like in Chapter 3.

Third, the task addressed in this chapter is limited to object classification, and it would be interesting to perform few-shot continual learning of new skills, especially in these embodied environments we introduced. It would also be interesting to investigate the action space of the agent. The simplest form of action would be having an option to ask for the class label of the current input.

Lastly, the learning algorithm is still largely driven by labeled examples because new clusters are only created when there is a new class label, and the overall loss function only cares about labeled examples. How can a human or an animal learn visual representations and categories through a continual stream of glimpses of different objects? Fortunately, this last question will be addressed in our next chapter.

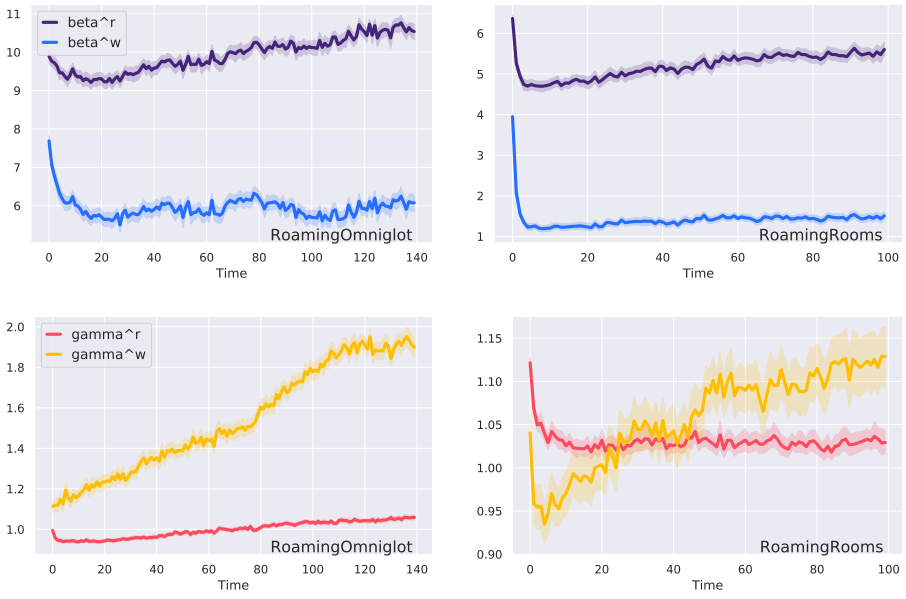


Figure 6.8: CPM control parameters $(\beta^{r,w}, \gamma^{r,w})$ vs. time. **Left:** RoamingOmniglot sequences; **Right:** RoamingRooms sequences; **Top:** $\beta^{r,w}$ the threshold parameter; **Bottom:** $\gamma^{r,w}$ the temperature parameter.

Chapter 7

Unsupervised Learning from an Online Visual Stream

With the introduction of online contextualized few-shot learning in the previous chapter, a natural follow-up question to ask is, can we still perform learning without any labeled examples? Humans learn from large amounts of unlabeled data through a continuous stream of experience that has strong temporal correlations and hierarchical structure. Can we equip machines with such learning capability?

The stream of online experience is determined by the fact that individuals operate in distinct physical environments, such as home and office. Transitions between environments occur on a coarse time scale relative to the rate of encounters with objects in the environment. The distribution over objects is strongly conditioned on the environment: frozen pizza and milk are in the supermarket, computer monitors, and desks at the office. The distribution may also be nonstationary; for example, a visitor may be infrequent, but when they are around, they're encountered frequently. And finally, the distribution can be highly unbalanced: an individual may interact with their co-workers daily but the boss only occasionally.

The goal of this chapter is to tackle the challenging problem of online unsupervised representation learning in the setting of environments with naturalistic structures. We desire a learning algorithm that will facilitate the categorization of objects encountered in the environment, with few- or zero-label supervision. In representation learning, methods often evaluate their ability to classify from the representation using either supervised linear readout or unsupervised clustering over the full dataset, both of which are typically done in a separate post-hoc evaluation phase. An important aim of our work is to produce object-category predictions throughout training and evaluation and to allow these predictions to guide subsequent categorization.

Unsurprisingly, the structure of natural environments contrasts dramatically with the standard scenario typically assumed by many machine learning algorithms: mini-batches of independent and identically distributed (iid) samples from a well-curated dataset. In unsupervised visual representation learning, the most successful methods rely on iid samples. Contrastive-based objectives (Chen et al., 2020a; He et al., 2020) typically assume that each instance in the mini-batch forms its own instance class, throwing away the potential similarity between instances. Clustering-based learning frameworks (Caron et al., 2018; Asano et al., 2020; Caron et al., 2020) often assume that the set of

cluster centroids remain relatively stable and that the clusters are balanced in size. Unfortunately, none of these assumptions necessarily hold true in a naturalistic online streaming setting. The performance of methods will suffer as a consequence. Contrastive approaches could eventually fail if examples of the same class are pushed apart; clustering approaches will behave erratically with nonstationary and imbalanced class distributions, two key facets of the natural online non-iid learning we focus on here.

To make progress on the challenge of unsupervised visual representation learning and categorization in a naturalistic setting, we propose the *online unsupervised prototypical network*, which performs learning of visual representations and object categories simultaneously in a single-stage process. Class prototypes are created via an online clustering procedure, and a contrastive loss (van den Oord et al., 2018) is used to encourage different views of the same image to be assigned to the same cluster. Notably, our online clustering procedure is more flexible relative to other clustering-based representation learning algorithms, such as DeepCluster (Caron et al., 2018) and SwAV (Caron et al., 2020): our model performs learning and inference as an online Gaussian mixture model, where clusters can be created online with only a single new example, and cluster assignments do not have to be balanced, which permits an approximation to natural imbalanced distributions from uncurated raw data.

We train and evaluate our algorithm on the RoamingRooms dataset proposed in Chapter 6, which uses imagery collected from a virtual agent walking through different rooms. Unlike the experiments in the previous chapter, our training is done without using any labeled data, and hence it is online and unsupervised. We compare to state-of-the-art unsupervised methods SimCLR (Chen et al., 2020a) and SwAV (Caron et al., 2020). Because they rely on sampling a large batch from an offline dataset, their performance drops significantly using smaller non-iid episodes. In contrast, our method can directly learn with online streaming data from small episodes, without requiring an example buffer, and surprisingly, we even outperform these strong methods trained offline with large batches of iid data. We also used RoamingOmniglot as a benchmark to investigate the effect of imbalanced classes and we find that our method is very robust to an imbalanced distribution of classes. For a version of ImageNet with non-iid structure, RoamingImageNet, we again outperform SimCLR and SwAV when using the same batch size. Finally, qualitative visualizations confirm that our online clustering procedure can automatically discover new concepts and categories by grouping a few similar instances together in the learned embedding space.

7.1 Online Unsupervised Prototypical Networks

In this section, we introduce our proposed model, *online unsupervised prototypical networks* (*OUPN*). We study the online categorization setting, where the model receives an input $\mathbf{x}_t$ at every time step t, and predicts both a categorical variable $\hat{y}_t$ to indicate the object class and also a binary variable $\hat{u}_t$ to indicate whether this is a known or new class. OUPN uses a network h to encode the input to obtain embedding $\mathbf{z}_t = h(\mathbf{x}_t; \theta)$, where θ represents the learnable parameters of the encoder network. $\mathbf{z}$ is then processed by a *prototype memory* which predicts $(\hat{y}_t, \hat{u}_t)$.

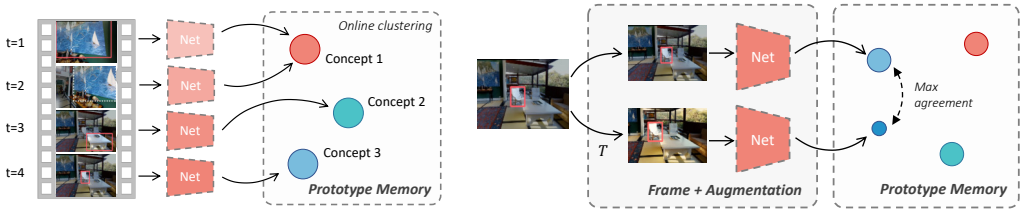


Figure 7.1: Our proposed online unsupervised prototypical network (OUPN). **Left:** OUPN learns directly from an online visual stream. Images are processed by a deep neural network to extract representations. Representations are stored and clustered in a prototype memory. Similar features are aggregated in a concept and new concepts can be dynamically created if the current feature vector is different from all existing concepts. **Right:** The network learning uses self-supervision that encourages different augmentations of the same frame to have consistent cluster assignments.

7.1.1 Prototype Memory

We formulate our prototype memory as a probabilistic mixture model, where each cluster corresponds to a Gaussian distribution $f(\mathbf{z}; \mathbf{p}, \sigma^2)$, with mean $\mathbf{p}$, a constant isotropic variance σ^2 shared across all clusters, and mixture weights w: $p(\mathbf{z}; P) = \sum_k w_k f(\mathbf{z}; \mathbf{p}_k, \sigma^2)$. Throughout a sequence, the number of components evolves as the model makes an online decision of when to create a new cluster or remove an old one. We assume that the prior distribution for the Bernoulli variable u is constant $u_0 \equiv \Pr(u = 1)$, and the prior for a new cluster is uniform over the entire space $z_0 \equiv \Pr(\mathbf{z}|u = 1)$. In the following, we formulate our prototype memory as an approximation to an online EM algorithm. The full derivation is included in the supplementary materials.

E-Step

Upon seeing the current input $\mathbf{z}_t$, the online clustering procedure needs to predict the cluster assignment or initiate a new cluster in the E-step.

Inferring cluster assignments. The categorical variable $\hat{y}$ infers the cluster assignment of the current input example with regard to the existing clusters.

$$\hat{y}_{t,k} = \Pr(y_t = k|\mathbf{z}_t, u = 0) = \frac{\Pr(\mathbf{z}_t|y_t = k, u = 0)\Pr(y_t = k)}{\Pr(\mathbf{z}_t, u = 0)} \tag{7.1}$$

$$= \frac{w_k f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2)}{\sum_{k'} w_{k'} f(\mathbf{z}_t; \mathbf{p}_{t,k'}, \sigma^2)} = \text{softmax}\left(\log w_k - \frac{1}{2\sigma^2} d(\mathbf{z}_t, \mathbf{p}_{t,k})\right), \tag{7.2}$$

where w_k is the mixing coefficient of cluster k and $d(\cdot, \cdot)$ is the distance function.

Inference on unknown classes. The binary variable $\hat{u}$ estimates the probability that the current input belongs to a new cluster:

$$\hat{u}_t = \Pr(u_t = 1|\mathbf{z}_t) = \frac{\Pr(\mathbf{z}_t|u_t = 1)\Pr(u_t = 1)}{\Pr(\mathbf{z}_t|u_t = 1)\Pr(u_t = 1) + \sum_k \Pr(\mathbf{z}_t|y_t = k, u = 0)\Pr(u = 0)} \tag{7.3}$$

$$= \frac{z_0 u_0}{z_0 u_0 + \sum_k w_k f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2)(1 - u_0)} \geq \frac{z_0 u_0}{z_0 u_0 + \max_k f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2)(1 - u_0)} \tag{7.4}$$

$$= \text{sigmoid}((\min_k d(\mathbf{z}_t, \mathbf{p}_{t,k}) - \beta)/\gamma), \tag{7.5}$$

where $\beta = -2\sigma^2(\log(z_0) + \log(u_0) - \log(1-u_0) + m\log(\sigma) + m\log(2\pi)/2)$, $\gamma = 2\sigma^2$.

M-Step

Here we infer the posterior distribution of the prototypes $\Pr(\mathbf{p}_{t,k}|\mathbf{z}_{1:t})$. We formulate an efficient recursive online update, similar to Kalman filtering, incorporating the evidence of the current input $\mathbf{z}_t$ and avoiding re-clustering the entire input history. We define $\hat{\mathbf{p}}_{t,k}$ as the mean posterior estimate of the k-th cluster at time step t, and $\hat{c}_{t,k}$ is the estimate of the inverse variance.

Updating prototypes. Suppose that in the E-step we have determined that $y_t = k$. Then the posterior distribution of the k-th cluster after observing $\mathbf{z}_t$ is:

$$\Pr(\mathbf{p}_{t,k}|\mathbf{z}_{1:t}, y_t = k) \propto \Pr(\mathbf{z}_t|\mathbf{p}_{t,k}, y_t = k)\Pr(\mathbf{p}_{t,k}|\mathbf{z}_{1:t-1}) \tag{7.6}$$

$$\approx f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2) \int_{\mathbf{p}'} f(\mathbf{p}_{t,k}; \mathbf{p}', \sigma^2_{t,d}) f(\mathbf{p}'; \hat{\mathbf{p}}_{t-1,k}, \hat{\sigma}^2_{t-1,k}) \tag{7.7}$$

$$= f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2) f(\mathbf{p}_{t,k}; \hat{\mathbf{p}}_{t-1,k}, \sigma^2_{t,d} + \hat{\sigma}^2_{t-1,k}). \tag{7.8}$$

If we assume that the transition probability distribution $\Pr(\mathbf{p}_{t,k}|\mathbf{p}_{t-1,k})$ is a zero-mean Gaussian with variance $\hat{\sigma}^2_{t,d} = (1/\rho - 1)\hat{\sigma}^2_{t-1,k}$, where $\rho \in (0, 1]$ is some constant that we defined to be the memory decay coefficient, and if $\sigma^2 = 1$, and $\hat{c}_{t,k} \equiv 1/\hat{\sigma}^2_{t,k}$, $\hat{c}_{t-1,k} \equiv 1/\hat{\sigma}^2_{t-1,k}$, it turns out we can formulate the update equation as follows, and $\hat{c}$ can be viewed as a count variable for the number of elements in each estimated cluster, subject to the decay factor ρ over time:

$$\hat{c}_{t,k} = \mathop{\mathbb{E}}_{y_t}[\hat{c}_{t,k}|_{y_t}] = \rho\hat{c}_{t-1,k} + \hat{y}_{t,k}(1 - \hat{u}_{t,k}), \tag{7.9}$$

$$\hat{\mathbf{p}}_{t,k} = \mathop{\mathbb{E}}_{y_t}[\hat{\mathbf{p}}_{t,k}|_{y_t}] = \mathbf{z}_t \frac{\hat{y}_{t,k}(1 - \hat{u}_{t,k})}{\rho\hat{c}_{t-1,k} + 1} + \hat{\mathbf{p}}_{t-1,k}\left(1 - \frac{\hat{y}_{t,k}(1 - \hat{u}_{t,k})}{\rho\hat{c}_{t-1,k} + 1}\right). \tag{7.10}$$

$$\hat{w}_{t,k} = \mathop{\mathbb{E}}_{y_t}[\hat{w}_{t,k}|_{y_t}] = \hat{c}_{t,k} / \sum_l \hat{c}_{t,l}. \tag{7.11}$$

Adding and removing prototypes. Our prototype memory is a collection of tuples $(\hat{\mathbf{p}}_k, \hat{c}_k)$. We convert the probability of whether an observation belongs to a new cluster into a decision: if $\hat{u}_t$ exceeds a threshold α, we create a new cluster in our prototype memory. Due to the decay factor ρ, our $\hat{c}$ estimate of a cluster can decay to zero over time, which is appropriate for modeling nonstationary environments. In practice, we keep a maximum number of K clusters, and if the memory is full and we try to add another cluster at time t, we simply pop out the least relevant prototype $\mathbf{p}_{k'}$, where $k' = \arg\min(\hat{w}_k)$: $P_t = P_{t-1} \setminus \{(\hat{\mathbf{p}}_{k'}, \hat{c}_{k'})\} \cup \{(\mathbf{z}_t, 1)\}$.

7.1.2 Representation learning

A primary goal of our learning algorithm is to learn good visual representations through this online categorization process. In the beginning, the encoder network is randomly initialized, and the prototype memory will not produce accurate class predictions since the representations are not informative. Our overall representation learning objective has three terms:

$$\mathcal{L} = \mathcal{L}_{\text{self}} + \lambda_{\text{ent}}\mathcal{L}_{\text{ent}} + \lambda_{\text{new}}\mathcal{L}_{\text{new}}. \tag{7.12}$$

This loss function drives the learning of the main network parameters θ, as well as other learnable control parameters β, γ, and τ. We explain each term in detail below.

1. **Self-supervised loss:** Inspired by recent self-supervised representation learning approaches, we apply augmentations on $\mathbf{x}_t$, and encourage the clustering assignments to match across different views. Self-supervision follows three steps: First, the model makes a prediction on the augmented view, and obtains $\hat{y}$ and $\hat{u}$ (E-step). Secondly, it updates the prototype memory according to the prediction (M-step). To create a learning target, we query the original view again, and obtain $\tilde{y}$ to supervise the cluster assignment of the augmented view, $\hat{y}'$, as in distillation (Hinton et al., 2015).

$$\mathcal{L}_{\text{self}} = \frac{1}{T} \sum_t -\tilde{y}_t \log \hat{y}'_t. \tag{7.13}$$

   Note that both $\tilde{y}_t$ and $\hat{y}'_t$ are produced after the M-step so we can exclude the "unknown" class in the representation learning objective. We here introduce a separate temperature parameter $\tilde{\tau}$ to control the entropy of the mixture assignment $\tilde{y}_t$.

2. **Entropy loss:** In order to encourage more confident predictions we introduce another loss function $\mathcal{L}_{\text{ent}}$ that controls the entropy of the original prediction $\hat{y}$, produced in the initial E-step.

$$\mathcal{L}_{\text{ent}} = \frac{1}{T} \sum_t -\hat{y}_t \log \hat{y}_t. \tag{7.14}$$

3. **New cluster loss:** Lastly, our learning formulation also includes a loss for initiating new clusters $\mathcal{L}_{\text{new}}$. We define it to be a Beta prior on the expected $\hat{u}$, and we introduce a hyperparameter μ to control the expected number of clusters.

$$\mathcal{L}_{\text{new}} = -\log \Pr(\mathbb{E}[\hat{u}]). \tag{7.15}$$

   This acts as a regularizer on the total number of prototypes: if the system is too aggressive in creating prototypes, then it does not learn to merge instances of the same class; while if it is too conservative, the representations can collapse to a trivial solution.

Relation to Online ProtoNet. The formulation of our probabilistic prototype memory is similar to Online ProtoNet, introduced in the Chapter 6. However, there are several main differences. First, we consider a decay term that can handle nonstationary mixtures, which is related to the variance of the transition probability. Second, our new cluster creation is unsupervised, whereas in Chapter 6 only labeled examples lead to new clusters. Most importantly, our representation learning objective is also entirely unsupervised, whereas Chapter 6 relies on a supervised loss.

Full algorithm. Let $\Theta = \{\theta, \beta, \gamma, \tau\}$ denote the union of the learnable parameters. Algorithm 3 outlines our proposed learning algorithm. The full list of hyperparameters are included in Appendix C.2.

Algorithm 3 Online Unsupervised Prototypical Learning

repeat
  $\mathcal{L}_{\text{self}} \leftarrow 0$, $p_{\text{new}} \leftarrow 0$.
  for $t \leftarrow 1 \dots T$ **do**
    Observe new input $\mathbf{x}_t$.
    Encode input, $\mathbf{z}_t \leftarrow h(\mathbf{x}_t; \theta)$.
    Compare to existing prototypes: $[\hat{u}_t, \hat{y}_t] \leftarrow \text{E-step}(\mathbf{z}_t, P; \beta, \gamma, \tau)$.
    if $\hat{u}_t^0 < \alpha$ **then**
      Assign $\mathbf{z}_t$ to existing prototypes: $P \leftarrow \text{M-step}(\mathbf{z}_t, P, \hat{u}_t, \hat{y}_t)$.
    else
      Recycle the least used prototype if P is full.
      Create a new prototype $P \leftarrow P \cup \{(\mathbf{z}_t, 1)\}$.
    end if
    Compute pseudo-labels: $[\_, \tilde{y}_t] \leftarrow \text{E-step}(\mathbf{z}_t, P; \beta, \gamma, \tilde{\tau})$.
    Augment a view: $\mathbf{x}'_t \leftarrow \text{augment}(\mathbf{x}_t)$.
    Encode the augmented view: $\mathbf{z}'_t \leftarrow h(\mathbf{x}'_t; \theta)$.
    Compare the augmented view to existing prototypes: $[\_, \hat{y}'_t] \leftarrow \text{E-step}(\mathbf{z}'_t, P; \beta, \gamma, \tau)$.
    Compute the self-supervision loss: $\mathcal{L}_{\text{self}} \leftarrow \mathcal{L}_{\text{self}} - \frac{1}{T} \tilde{y}_t \log \hat{y}'_t$.
    Compute the entropy loss: $\mathcal{L}_{\text{ent}} \leftarrow \mathcal{L}_{\text{ent}} - \frac{1}{T} \hat{y}_t \log \hat{y}_t$.
    Compute the average probability of creating new prototypes, $p_{\text{new}} \leftarrow p_{\text{new}} + \frac{1}{T} \hat{u}_t$.
  end for
  Compute the new cluster loss: $\mathcal{L}_{\text{new}} \leftarrow -\log \Pr(p_{\text{new}})$.
  Sum up losses: $\mathcal{L} \leftarrow \mathcal{L}_{\text{self}} + \lambda_{\text{ent}} \mathcal{L}_{\text{ent}} + \lambda_{\text{new}} \mathcal{L}_{\text{new}}$.
  Update parameters: $\Theta \leftarrow \text{optimize}(\mathcal{L}, \Theta)$.
until convergence
return Θ

It is worth noting that if we create a new prototype every time step, then OUPN is similar to a standard contrastive learning with an instance-based InfoNCE loss (Chen et al., 2020a; He et al., 2020); therefore it can be viewed as a generalization of this approach. Additionally, all the losses can be computed online without having to store any examples beyond the collection of prototypes.

7.2 Related Work

Self-supervised learning. Self-supervised learning methods discover rich and informative visual representations without using class labels. *Instance-based approaches* aim to learn invariant representations of each image under different transformations (van den Oord et al., 2018; Misra and van der Maaten, 2020; Tian et al., 2020; He et al., 2020; Chen et al., 2020a,b; Grill et al., 2020; Chen and He, 2020). They typically work well under iid learning with large batch sizes, which contrasts with realistic learning scenarios. Our method is also related to *clustering-based approaches*, which computes clusters on top of the learned embedding, and uses the cluster assignment to self-supervise the embedding network (Caron et al., 2018; Zhan et al., 2020; Asano et al., 2020; Caron et al., 2020; Li et al., 2021). To compute the cluster assignment, DeepCluster (Caron et al., 2018; Zhan

et al., 2020) and PCL (Li et al., 2021) use the k-means algorithm whereas SeLa (Asano et al., 2020) and SwAV (Caron et al., 2020) uses the Sinkhorn-Knopp algorithm (Cuturi, 2013). However, they typically assume a fixed number of clusters, and Sinkhorn-Knopp further assumes a balanced assignment as an explicit constraint. In contrast, our online clustering procedure is more flexible, as it can create new clusters on-the-fly with only a single new example and does not assume balanced cluster assignments.

Representation learning from video. There has also been a surge of interest in leveraging video data to learn visual representations (Wang and Gupta, 2015; Orhan et al., 2020; Pathak et al., 2017; Xiong et al., 2021). Wang and Gupta (2015) samples positive example pairs from a video track of the same object; Orhan et al. (2020) divides the video into equal chunks and trains a network to predict the chunk index. Pathak et al. (2017) learned visual representations using low-level motion-based grouping cues, and more recently, Xiong et al. (2021) uses optical flow to encourage regions that move together to have consistent representations across frames. These approaches all sample video subsequences uniformly over the entire dataset, whereas our model directly learns from an online stream of data. Our model also does not have the assumption that inputs must be adjacent frames in the video.

Online and incremental representation learning. Our work is also related to online and continual representation learning (Rebuffi et al., 2017; Castro et al., 2018; Rao et al., 2019; Jerfel et al., 2019; Javed and White, 2019; Hayes et al., 2020). Incremental learning (Rebuffi et al., 2017; Castro et al., 2018) studies learning a set of classes incrementally. Continual mixture models (Rao et al., 2019; Jerfel et al., 2019) designate a categorical latent variable that can be dynamically allocated for a new environment. Our model has a similar mixture latent variable setup but one major difference is that we operate on example-level rather than task-level. OML (Javed and White, 2019) is a meta-learning framework that learns the representation to support online learning at test time. Streaming learning (Hayes et al., 2020, 2019) aims to perform representation learning online. Most of the works here except (Rao et al., 2019) assume that the training data is fully supervised. Our prototype memory also resembles the replay buffer for experience replay methods (Buzzega et al., 2020; Kim et al., 2020), but we store the hidden feature prototypes instead of the input examples.

Latent variable modeling on sequential data. Our model also relates to a family of latent variable generative models for sequential data (Johnson et al., 2016; Krishnan et al., 2015; He et al., 2018; Denton and Fergus, 2018; Zhu et al., 2020). Both our model and these approaches aim to infer latent variables with temporal structure. However, instead of using input reconstruction as the main learning objective, our model leverages self-supervised contrastive learning to avoid learning difficulty in the generative decoder.

Online mixture models. Our clustering module is related to the literature on online mixture models, e.g., (Hughes and Sudderth, 2013; Pinto and Engel, 2015; Song and Wang, 2005). Typically, these are designed for fast and incremental learning of clusters without having to recompute clustering over the entire dataset. Despite presenting a similar online clustering algorithm, our goal is to jointly learn both online clusters and input representations that facilitate future online clustering episodes.

Few-shot learning. Our model is able to recognize a new class with only one or a few examples, which relates to the literature on few-shot learning (Li et al., 2007; Lake et al., 2015). Our prototype-based memory is also inspired by the Prototypical Network (Snell et al., 2017) and its variants, e.g., IMP (Allen et al., 2019) and Online ProtoNet (Ch. 6). Ren et al. (2018b); Huang et al. (2019) demonstrated improved performance by using unlabeled examples. More recent methods can entirely remove the reliance on class labels in the training pipeline, using self-supervised learning (Hsu et al., 2019; Gidaris et al., 2019; Antoniou and Storkey, 2019; Khodadadeh et al., 2019; Medina et al., 2020).

Classical few-shot learning, however, relies on episodes of an equal number of training and test examples from a fixed number of new classes. Incremental few-shot learning approaches (Gidaris and Komodakis, 2018; Ren et al., 2019) consider both old and new classes, while Meta-Dataset (Triantafillou et al., 2020) considers episodes with varying numbers of examples and classes. In Chapter 6, we defined a new setup that incrementally accumulates new classes and re-visits old classes over a sequence of inputs. We evaluate our algorithm on a similar setup; however, unlike in the previous chapter, our proposed algorithm here learns visual representations without relying on any class labels.

Human category learning. Our work is related to human concept learning models from cognitive science (Murphy, 2004). Carpenter and Grossberg (1987) and Anderson (1991) proposed different online clustering models on top of fixed representations to learn categories. SUSTAIN (Love et al., 2004) is another computational model for human category learning that can perform both supervised and unsupervised clustering of categories. Lake et al. (2009) proposed online mixture estimation and applied it on audio and visual data. In contrast to these human learning models, our model learns both representations and categories in an end-to-end fashion.

7.3 Experiments

In this section, we evaluate our proposed learning algorithm on a set of visual learning tasks and evaluate the quality of the output categories. Different from prior work on visual representation learning, we focus on online non-iid image sequences to highlight the merit of method, since that is the primary scenario of interest.

Implementation details. Throughout our experiments, we make two changes on the model inference procedure defined above. First, we use cosine similarity instead of negative squared Euclidean distance for computing the mixture logits, because cosine similarity is bounded and is found to be more stable to train. Second, when we perform cluster inference, we treat the mixing coefficients w_k as constant and uniform as otherwise we find that the representations may collapse into a single large cluster.

Evaluation setup. Although our training is entirely unsupervised, we can evaluate our model under both *unsupervised* and *supervised* settings. During evaluation, we present our model a sequence of new images (labeled or unlabeled) and we would like to see how well the model is able to output a successful grouping of the inputs. For the unsupervised setting, there is no class label associated with each image, and our model g directly predicts $\hat{y}_t = g(\mathbf{x}_{1:t})$. For the supervised setting,

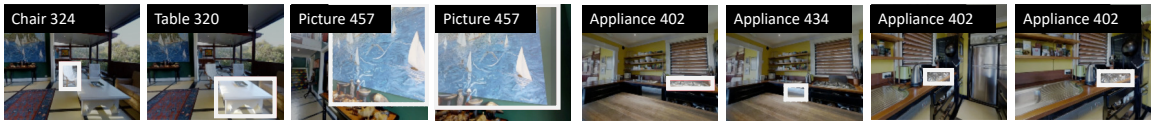


Figure 7.2: An example subsequence of the *RoamingRooms* dataset. Images represent consecutive glimpses of an online agent roaming in an indoor environment and the task is to recognize the object instances.

	AMI	AP
Supervised		
Online ProtoNet (Ch. 6)	79.02	89.94
Unsupervised		
Random Network	28.25	11.68
SimCLR (Chen et al., 2020a)	50.03	52.98
SwAV (Caron et al., 2020)	42.70	37.31
SwAV+Queue (Caron et al., 2020)	48.31	50.40
OUPN (Ours)	**78.16**	**84.86**

Table 7.1: Novel object recognition performance on *RoamingRooms*

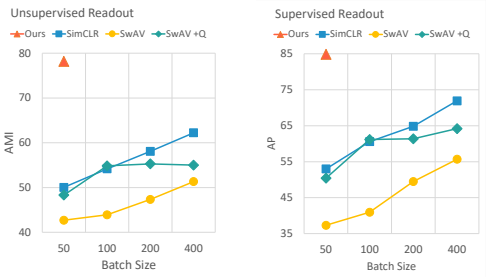


Figure 7.3: Comparison to SimCLR and SwAV with larger batch sizes on *RoamingRooms*

following our previous chapter, give the model the ground-truth label y_t after the prediction of $\hat{y}_t$, i.e. $\hat{y}_t = g(\mathbf{x}_{1:t}, y_{1:t-1})$.

Evaluation metrics. We use the following unsupervised and supervised metrics.

- **Adjusted mutual information (AMI):** In the *unsupervised* setting, we use the mutual information metric to evaluate the similarity between our prediction $\{\hat{y}_1, \ldots, \hat{y}_T\}$ the groundtruth class ID $\{y_1, \ldots, y_T\}$. Since the online clustering method admits a threshold parameter α to control the number of output clusters, therefore for each model we sweep the value of α to maximize the AMI score, to make the score threshold-invariant: $\text{AMI}_{\max} = \max_\alpha \text{AMI}(y, \hat{y}(\alpha))$.
- **Average precision (AP):** In the *supervised* setting, we followed the evaluation procedure in Chapter 6 and used average precision, which combines both accuracy for predicting known classes as well as unknown ones.

Competitive methods. We compare our proposed model with the following state-of-the-art self-supervised visual representation learning methods. Since none of these competitive methods are designed to output classes with a few examples, we use our online clustering procedure to read out from these learned representations.

- **SimCLR** (Chen et al., 2020a) is a contrastive learning method with an instance-based objective that tries to classify an image instance among other augmented views of the same batch of instances. It relies on a large batch size and is often trained on well-curated datasets such as ImageNet (Russakovsky et al., 2015).
- **SwAV** (Caron et al., 2020) is a contrastive learning method with a clustering-based objective. It has a stronger performance than SimCLR on ImageNet. The clustering is achieved through Sinkhorn-Knopp which assumes balanced assignment, and prototypes are learned by gradient descent.

Figure 7.4: Image retrieval results on *RoamingRooms*. In each row, the leftmost image is the query image, and top-9 retrieved images are shown to its right. For each retrieval its cosine similarity score is in the top left; a green border signifies a correct retrieval (matching the query instance), red is a false positive, yellow a miss. Recall is the proportion of instances retrieved within the top-9.

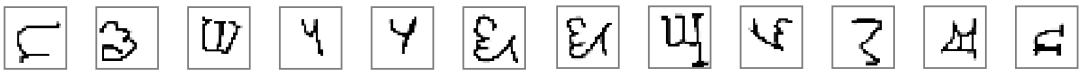


Figure 7.5: An example subsequence of an episode sampled from the *RoamingOmniglot* dataset

- **SwAV+Queue** (Caron et al., 2020) is a SwAV variant with an additional example queue. This setup is proposed in (Caron et al., 2020) to deal with small training batches. A feature queue that accumulates instances across batches allows the clustering process to access more data points. The queue size is set to 2000.

7.3.1 Indoor Home Environments

We first evaluate the algorithm using the *RoamingRooms* dataset, introduced in Chapter 6, where the images are collected from indoor environments (Chang et al., 2017a) using a random walking agent. The dataset contains 1.22M image frames and 7K instance classes from 6.9K random walk episodes collected using 1.2K panoramas. Each image is resized to $120 \times 160 \times 3$. We use a maximum of 50 frames for training due to memory constraints, and all methods are evaluated on the test set with a maximum of 100 frames per episode. Each frame has an object annotation with its segmentation mask and the task here is to recognize the object instance IDs. Since our task is classification, we also send the segmentation mask as an additional channel in the input. An example episode is shown in Fig. 7.2.

Implementation details. We use a ResNet-12 (Oreshkin et al., 2018) as the encoder network, and we train our method over 80k 50-frame episodes (4M image frames total), using a single NVIDIA 1080-Ti GPU. We follow the same procedure of image augmentation as SimCLR (Chen et al., 2020a). We use 150 prototypes with $\rho = 0.995$. More implementation details can be found in in Appendix C.2.

Results. Results are shown in Tab. 7.1. Although both SimCLR and SwAV have shown promising results on large batch learning on ImageNet, their performances here are relatively weak compared to the supervised baseline. Adding a queue slightly improves SwAV; however, since the examples in the

Table 7.2: Results on *RoamingOmniglot*

	AMI	AP
Supervised		
Pretrain-Supervised	84.48	93.83
Online ProtoNet (Ch. 6)	89.64	92.58
Unsupervised		
Random Network	17.66	17.01
SimCLR (Chen et al., 2020a)	59.06	73.50
SwAV (Caron et al., 2020)	62.09	75.93
SwAV+Queue (Caron et al., 2020)	67.25	81.96
OUPN (Ours)	**84.42**	**92.84**

Table 7.3: Results on *RoamingImageNet*

	AMI	AP
Supervised		
Pretrain-Supervised	29.44	24.39
Online ProtoNet (Ch. 6)	29.73	25.38
Unsupervised		
Random Network	4.55	2.65
SimCLR (Chen et al., 2020a)	6.87	12.25
SwAV (Caron et al., 2020)	9.87	5.23
SwAV+Queue (Caron et al., 2020)	10.61	4.83
OUPN (Ours)	**19.03**	**15.05**

Figure 7.6: Comparison to IID modes of SimCLR and SwAV on *RoamingRooms* with larger training batch sizes.

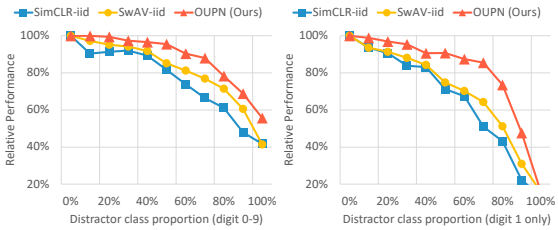


Figure 7.7: Robustness to imbalanced distributions by adding distractors (Omniglot mixed with MNIST images). Performance is relative to the original performance and a random baseline.

queue cannot be used to compute gradients, the nonstationary distribution still hampers gradient updates. In contrast, our method OUPN shows impressive performance on this benchmark: it almost matches the supervised learner in AMI, and reached almost 95% of the performance of the supervised learner in AP.

To illustrate the impact of our small batch episodes, we increase the batch size for SimCLR and SwAV, from 50 to 400, at the cost of using multiple GPUs training in parallel. The results are shown in Fig. 7.3. Results indicate that increasing the batch size can improve these baselines, which matches our expectation. Nevertheless, our method using a batch size of 50 is still able to outperform SimCLR and SwAV using a batch size of 400, which takes 8× computational resource compared to ours. Note that the large batch experiments are designed to provide the best setting for SimCLR and SwAV to succeed. We do not intend to run our model with larger batch size since our prototype memory is a sequential module. Moreover, keeping the batch size smaller allows quicker online adaptation and less memory consumption.

Comparison to iid modes of SimCLR and SwAV. The original SimCLR and SwAV were designed to train on iid data. To study the effects of this assumption, we implemented an approximation to an iid distribution by using a large random queue that shuffles the image frames. As in the study shown in Fig. 7.6, we again vary the batch size for these competitive methods. Both SimCLR and SwAV thrive with iid data; the gains of iid over non-iid can be seen by comparing Fig. 7.6 to Fig. 7.3. Larger batches help both methods again here. Interestingly, our method using a batch size of 50 non-iid data again outperforms both methods using a batch size of 400 of iid data.

Visualization on image retrieval. To verify the usefulness of the learned representation, we ran an image retrieval visualization over 100 test episodes in Fig. 7.4, and we also provided the cosine

similarity score for each retrieved image. The top retrieved images are all from the same instance of the query image, and our model sometimes achieves perfect recall. This confirms that our model can handle a certain level of view angle changes. We also investigated the missed examples and we found that these are taken from more distinct view angles. For example, the missed examples in bottom row are viewed from the back of the flower. However, note that we only computed the pairwise similarity score for retrieval, and therefore it is possible that the actual clusters are more inclusive as the model incrementally computes the average embedding as the prototype.

We visualize the clustering mechanism and the learned image embeddings on *RoamingRooms* in Fig. 7.8 and 7.9. The results suggest that our model can handle a certain level of view point changes by grouping different view points of the same object into a single cluster. It also shows that our model is instance-sensitive: for example, the headboard, pillows, and the blanket are successfully separated.

7.3.2 Handwritten Characters

We also evaluated our method on a different task: recognizing handwritten characters from Omniglot (Lake et al., 2015). In this experiment, images are not organized in a video-like sequence, and models have to reason more about conceptual similarity between images in order to learn grouping. Furthermore, since this is a more controllable setup, we can test our hypothesis concerning sensitivity to class imbalance by performing manipulations on the episode distribution.

Our episodes are sampled from the *RoamingOmniglot* dataset introduced in the previous chapter. An episode involves several different *contexts*, each consisting of a set of classes, and in each context, classes are sampled from a Chinese restaurant process. We use 150-frame episodes with 5 contexts.

Results. The results are reported in Table 7.2. Our model is able to significantly reduce the gap between supervised and unsupervised models, outperforming SimCLR and SwAV.

Effect of imbalanced distribution. We further study the effect of imbalanced cluster sizes by manipulating the class distribution in the training episodes. In the first setting, we randomly replace Omniglot images with MNIST digits, with probability from 0% to 100%. For example, at 50% rate, an MNIST digit is over 300 times more likely to appear compared to any Omniglot character class, so the episodes are composed of half frequent classes and half infrequent classes. In the second setting, we randomly replace Omniglot images with MNIST digit 1 images, which makes the imbalance even greater. We compared our method to SimCLR and SwAV in the iid setup, since this is the scenario they were designed for. Results of the two settings are shown in Fig. 7.7, and our method is shown to be more robust under imbalanced distribution than SimCLR and SwAV. Compared to clustering-based methods like SwAV, our prototypes can be dynamically created and updated with no constraints on the number of elements per cluster. Compared to instance-based methods like SimCLR, our prototypes also samples the contrastive pairs more equally during learning. We hypothesize that these model aspects contribute to the differences in robustness.

Visualization of learned categories. We visualize the learned categories in *RoamingOmniglot* using t-SNE (Maaten and Hinton, 2008), and the results are in Fig. 7.10 and 7.11. Different colors

Table 7.4: Effect of mem. size K

K	*RoamingOmniglot* AMI	*RoamingOmniglot* AP	*RoamingRooms* AMI	*RoamingRooms* AP
50	89.19	95.12	75.33	82.42
100	**90.54**	95.83	76.70	83.51
150	90.24	**95.92**	77.07	84.00
200	90.36	95.68	76.81	**84.45**
250	89.87	95.69	**77.83**	84.33

Table 7.5: Effect of decay rate ρ

ρ	*RoamingOmniglot* AMI	*RoamingOmniglot* AP	*RoamingRooms* AMI	*RoamingRooms* AP
0.9	51.12	64.19	65.07	75.50
0.95	79.78	89.30	74.33	81.92
0.99	89.43	95.54	76.97	84.05
0.995	**90.80**	**95.90**	**77.78**	**85.02**
0.999	86.27	93.69	38.89	39.37

Table 7.6: Effect of λ_{new}

λ_{new}	*RoamingOmniglot* AMI	*RoamingOmniglot* AP	*RoamingRooms* AMI	*RoamingRooms* AP
0.0	38.26	93.40	19.49	73.93
0.1	86.60	93.50	67.25	71.69
0.5	89.89	95.28	**78.04**	**84.85**
1.0	**90.06**	**95.81**	77.59	84.36
2.0	88.74	95.73	77.62	84.72

represent different ground-truth classes. Our method is able to learn meaningful embeddings and roughly group items of similar semantic meanings together.

Ablation studies. We study the effect of certain hyperparameters of our model in Tab. 7.4, 7.5 and 7.6, using the validation set of *RoamingRooms* and *RoamingOmniglot*. The memory size K does not affect the final performance, but interestingly the decay factor ρ is more important. As shown in Table 7.6, the new cluster loss is necessary since the model performance drops significantly when the value of λ_{new} is below 0.5. More studies on the effect of α, $\tilde{\tau}$, λ_{ent} and the Beta mean μ are included in Appendix C.3.2.

7.3.3 ImageNet Images

Lastly, we evaluate on episodes composed of ImageNet images, and using the *RoamingImageNet* dataset, which has a structure analogous to *RoamingOmniglot*. This dataset's scale resembles *RoamingRooms* (around 1.3 million images), but here we focus on semantic object classes instead of instance classes. We use this benchmark as a stress test: it does not fit the target scenario of an natural online exploration, but it still provides a test whether a method can learn semantic concepts with a large intra-class variance. We developed a form of non-iid episodes, consisting of 48 frames drawn with temporal dependence from 3 contexts. Results are shown in Table 7.3. Although ImageNet is a more challenging benchmark, our model is still considerably better than SimCLR and SwAV.

Results. Results are shown in Tab. 7.3. In this experiment, our model is worse than the iid baselines. This is understandable, as both SimCLR and SwAV are well-tuned methods on ImageNet and they use much more computational resources. However, it also suggests that our method has some limitations towards handling larger intra-class variance, and it is more challenging to assume the clustering latent structure on top of these embeddings. However, our model is still comparable to SimCLR using small non-iid batches and achieves the best performance on the AMI score.

7.4 Discussion and Conclusion

Taking steps towards the online and nonstationary learning process of real-world agents, in this chapter we develop an online unsupervised algorithm for learning visual representation and categories. Unlike standard self-supervised learning, our category learning is embedded in a probabilistic clustering module that is learned jointly with the representation encoder. Our clustering is more flexible and supports the learning of new categories with very few examples. Our method is evaluated in both synthetic and realistic image sequences and outperforms state-of-the-art self-supervised learning algorithms using online non-iid data inputs.

Beyond the datasets we benchmarked in this chapter, in the future, it will be exciting to try our algorithm on a more realistic first-person point-of-view video dataset, such as the SAYCam dataset (Sullivan et al., 2020), collected with a head mount camera of a child. This allows us to experiment the learning algorithm on the online imbalanced distribution of data stream that we had imagined.

The model we proposed in this chapter is a self-supervised extension of the online prototypical network introduced in Chapter 6. For simplicity, we omit the context modeling component since learning of a recurrent network with unsupervised data will bring additional challenges, and this can lead to new self-supervised extensions perhaps by combining ideas from the natural language learning literature (Devlin et al., 2019). For example, we could perform self-supervision by predicting based on the temporal context and predicting the future scenes.

In terms of the experiment results, the model performance on RoamingImageNet is not as competitive compared to the two other benchmarks. This suggests that representation learning for general natural images remains a difficult problem and how to learn representations with a small non-iid batch is very much still an unsolved challenge.

Lastly, another aspect that is not investigated here is the robustness to continual distribution shift. Traditional continual learning experiments are carried out in purely supervised settings, and combining distribution shift with our setup can lead to a more naturalistic learning environment. In order to tackle this challenge, we may have to incorporate some replay mechanisms (Chaudhry et al., 2019b) to avoid catastrophic forgetting.

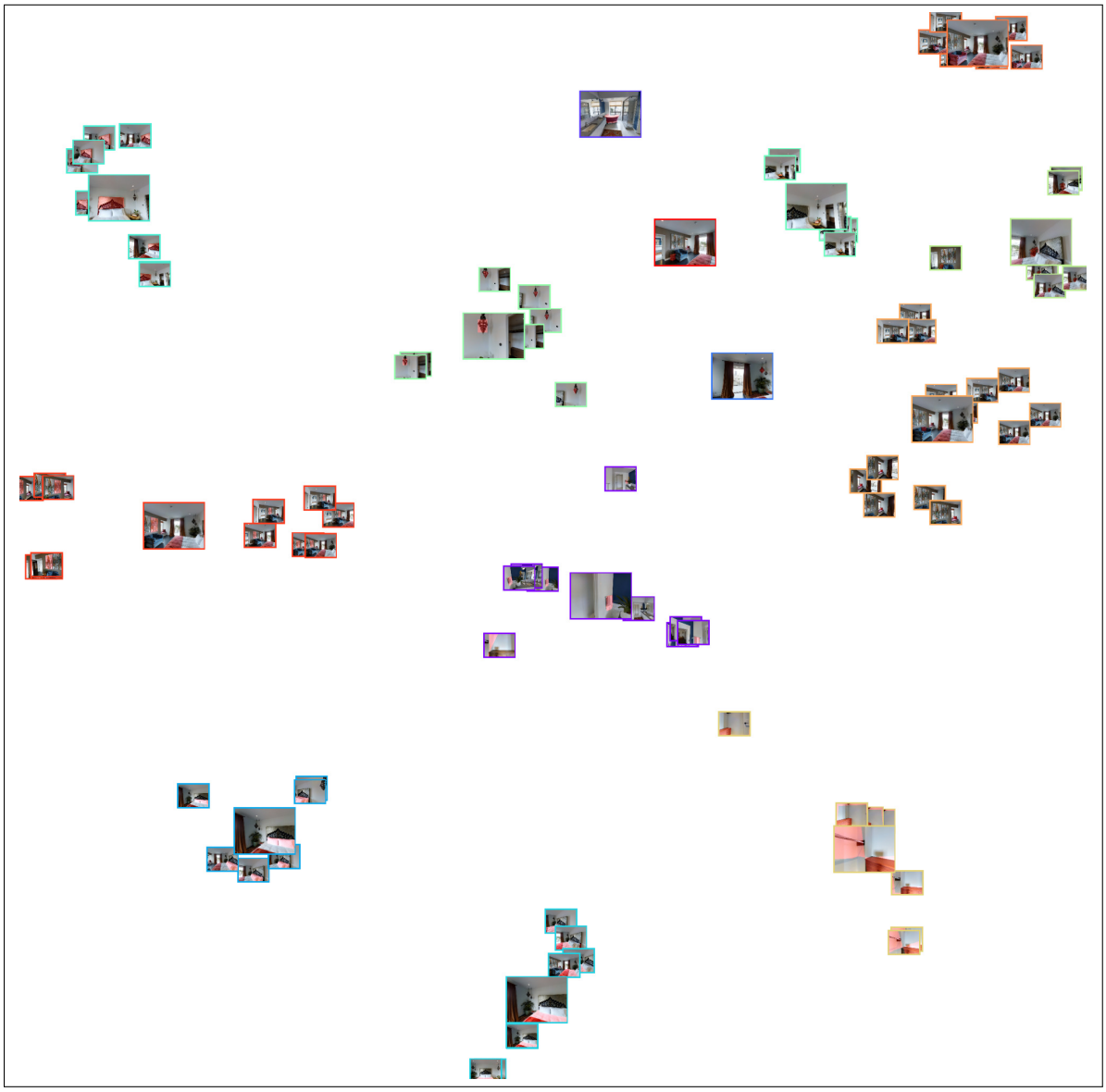


Figure 7.8: Embeddings and clustering outputs of an example episode (1). Embeddings are extracted from the trained CNN and projected to 2D space using t-SNE (Maaten and Hinton, 2008). The main object in each image is highlighted in a red mask. The nearest example to each cluster centroid is enlarged. Image border colors indicate the cluster assignment.

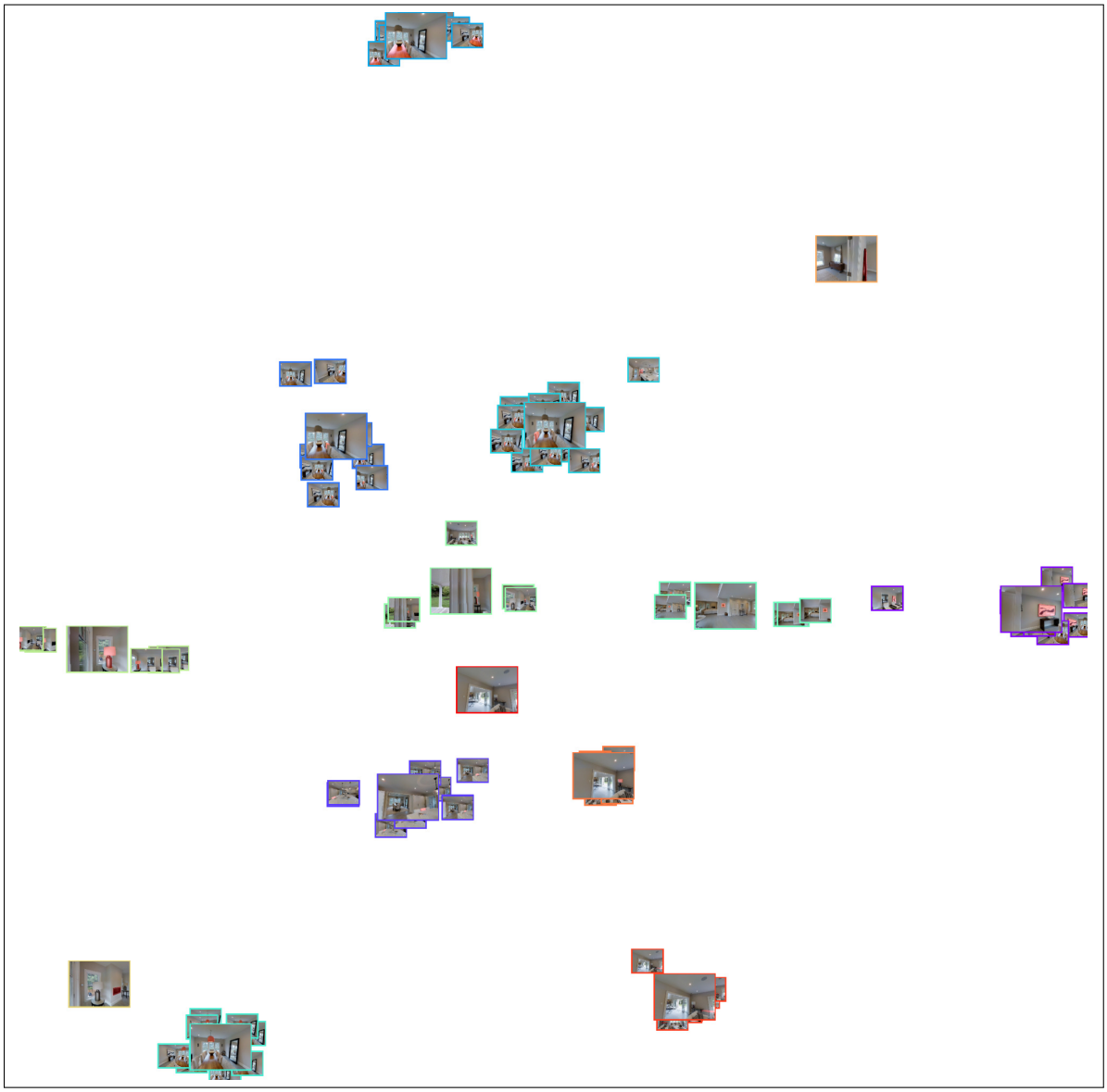


Figure 7.9: Embeddings and clustering outputs of another example episode (2).

Figure 7.10: Embedding visualization of an unsupervised training episode of *RoamingOmniglot*. Different colors denote the ground-truth class IDs.

Figure 7.11: Embedding visualization of an test episode of *RoamingOmniglot*.

Chapter 8

Conclusion and Future Directions

We have made machine learning much more flexible: examples can be either old or new, labeled or unlabeled, balanced or imbalanced, right or wrong, known or unknown.

Humans and animals have the remarkable ability to learn new concepts in the natural world with very few examples. Classic machine learning, on the other hand, requires many examples to recognize a fixed set of concepts in a closed environment. This thesis attempts to explore the question of how to let machines learn new concepts in an open world, with limited labeled data, incremental data, unlabeled data, imbalanced data, noisy data, and online and streaming data. By leveraging new meta-learning and representation learning algorithms, we have made machine learning much more flexible: examples can be either *old or new, labeled or unlabeled, balanced or imbalanced, right or wrong, known or unknown*. In the future, instead of training models in all possible environments and scenarios in advance, which is prohibitively expensive to scale up, we will build intelligent agents that can learn from their own perspectives and adapt to novel environments when needed.

The paradigms, algorithms, and techniques introduced in this thesis serve only as a start towards this bigger goal of building agent-based machine intelligence, and they potentially can lead to a wide range of future research possibilities. Immediate short-term extensions can make the output space much more expressive. We will be able to obtain much finer-grained visual understanding by localizing and segment the object with limited or no labels (Pérez-Rúa et al., 2020; Cermelli et al., 2020), and potentially using data from an online visual stream. This trend, combined with 3D vision techniques, will eventually lead to a learning algorithm that can capture a holistic 3D understanding of the world. Current approaches in this area are still mainly based on the classic paradigm of large-scale supervised machine learning (Liang et al., 2019; Aygün et al., 2021), and we will soon see exciting upgrades to open-world designs in this area.

Another important step forward beyond class learning is to acquire and exploit hierarchical and relational knowledge from an online and continual data stream. Learning scene graphs is a classic topic in computer vision (Choi et al., 2013; Zhao and Zhu, 2013); however, it is still a challenging problem due to the combinatorial nature of graphs, which makes the data imbalance problem more

severe. On the other hand, agents can become more efficient at solving new tasks by leveraging such a relational knowledge base. Many of the current approaches rely on modeling of explicit relations in terms of graphs (Krishna et al., 2017, 2019), but discrete graph connections could suffer from learning difficulties if connected to downstream tasks. Another possibility is to capture the relations in the latent embedding space that can be later recovered through readout (Wang et al., 2021a). In short, it is exciting to see many future opportunities to combine few-shot learning with an incremental, relational, and hierarchical knowledge base.

After learning new concepts and relations in an open world, how can we make the learned knowledge useful for solving new planning and reasoning tasks? Traditional planning algorithms employ heuristic search by using the available object information from perception. Now, with the perception network and concept storage combined in a unified framework, it will be a fruitful direction to integrate the planning module together (Levine et al., 2016; Zeng et al., 2019), such that new concepts can directly support the reasoning and decision making process in an end-to-end manner, and the goal-driven attention can in turn influence the perception process to make the overall process more efficient (Wei et al., 2021). Most of the explorations in end-to-end planning and reasoning are based on reinforcement learning and imitation learning, both of which require as many training examples as classic machine learning. On the other hand, we humans have been learning to learn over a lifetime to be more efficient at solving new challenges, and such a lifelong aspect of meta-learning can lead to various kinds of intelligent behaviors. Instead of only focusing on solving a single task, lifelong meta-learners will solve a multitude of planning and reasoning tasks and seek commonalities between tasks to allow a faster transfer.

As we looking for ways for agents to acquire general intelligence through multi-task learning, a more fundamental understanding of strong and systematic generalization (Bahdanau et al., 2019) will eventually be required. One way to approach this is through architectural innovations that can bring the inductive biases required for generalizing to new tasks with different input tokens and primitives. Another approach is to expose the agent to multiple tasks and environments and expect generalization to emerge when there are enough constraints from external environments. Finally, a more theoretical understanding of the limitation of inter-task generalization will also be a critical component of the full picture of open-world machine learning.

To conclude, this thesis presents our contribution towards open-world machine learning with limited labeled data, and the technical chapters explore learning under various naturalistic conditions. Combined together, they offer a holistic view on how to liberate machine learning from training only in a closed world environment of a fixed number of classes, towards learning in an open environment where new and rare-seen concepts appear endlessly. Looking forward, the journey of agent-based intelligence has just begun, and as envisioned by Alan Turing, one day we shall build machines that can learn in the natural world, just like a human child.

Appendix A

Appendix for Chapter 4

A.1 Omniglot Dataset Details

We used the following split details for experiments on Omniglot dataset. This is the same train/test split as (Vinyals et al., 2016), but we created our own validation split for selecting hyper-parameters. Models are trained on the train split only.

Train Alphabets: Alphabet_of_the_Magi, Angelic, Anglo-Saxon_Futhorc, Arcadian, Asomtavruli_(Georgian), Atemayar_Qelisayer, Atlantean, Aurek-Besh, Avesta, Balinese, Blackfoot_(Canadian_Aboriginal_Syllabics), Braille, Burmese_(Myanmar), Cyrillic, Futurama, Ge_ez, Glagolitic, Grantha, Greek, Gujarati, Gurmukhi (character 01-41), Inuktitut_(Canadian_Aboriginal_Syllabics), Japanese_
(hiragana), Japanese_(katakana), Korean, Latin, Malay_(Jawi_-_Arabic), N_Ko, Ojibwe_(Canadian_Aboriginal_Syllabics), Sanskrit, Syriac_(Estrangelo), Tagalog, Tifinagh

Validation Alphabets: Armenian, Bengali, Early_Aramaic, Hebrew, Mkhedruli_(Geogian)

Test Alphabets: Gurmukhi (character 42-45), Kannada, Keble, Malayalam, Manipuri, Mongolian, Old_Church_Slavonic_(Cyrillic), Oriya, Sylheti, Syriac_(Serto), Tengwar, Tibetan, ULOG

A.2 *tiered*-Imagenet Dataset Details

Each high-level category in *tiered*-ImageNet contains between 10 and 30 ILSVRC-12 classes (17.8 on average). In the ImageNet hierarchy, some classes have multiple parent nodes. Therefore, classes belonging to more than one category were removed from the dataset to ensure separation between training and test categories. Test categories were chosen to reflect various levels of separation between training and test classes. Some test categories (such as "working dog") are fairly similar to training categories, whereas others (such as "geological formation") are quite different. The list of categories is shown below and statistics of the dataset can be found in Table A.1. A visualization of the categories according to the ImageNet hierarchy is shown in Figure A.1.

Train Categories: `n02087551` (hound, hound dog), `n02092468` (terrier), `n02120997` (feline, felid), `n02370806` (ungulate, hoofed mammal), `n02469914` (primate), `n01726692` (snake, serpent, ophidian), `n01674216` (saurian), `n01524359` (passerine, passeriform bird), `n01844917` (aquatic bird), `n04081844` (restraint, constraint), `n03574816` (instrument), `n03800933` (musical instrument, in-

Table A.1: Statistics of the *tiered*-ImageNet dataset.

	Train	Val	Test	Total
Categories	20	6	8	34
Classes	351	97	160	608
Images	448,695	124,261	206,209	779,165

strument), n03125870 (craft), n04451818 (tool), n03414162 (game equipment), n03278248 (electronic equipment), n03419014 (garment), n03297735 (establishment), n02913152 (building, edifice), n04014297 (protective covering, protective cover, protection).

Validation Categories: n02098550 (sporting dog, gun dog), n03257877 (durables, durable goods, consumer durables), n03405265 (furnishing), n03699975 (machine), n03738472 (mechanism), n03791235 (motor vehicle, automotive vehicle),

Test Categories: n02103406 (working dog), n01473806 (aquatic vertebrate), n02159955 (insect), n04531098 (vessel), n03839993 (obstruction, obstructor, obstructer, impediment, impedimenta), n09287968 (geological formation, formation), n00020090 (substance), n15046900 (solid).

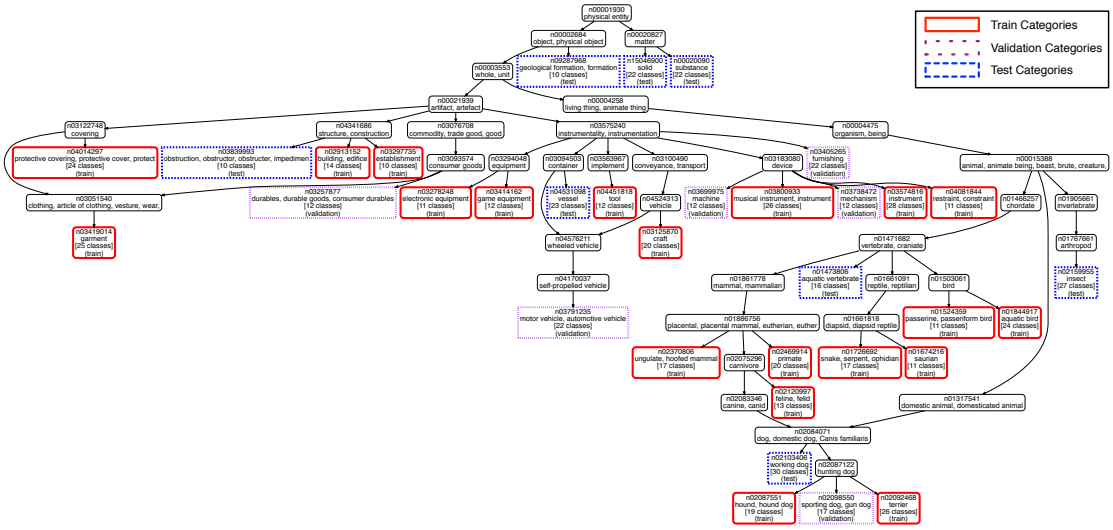


Figure A.1: Hierarchy of *tiered*-Imagenet categories. Training categories are highlighted in red, validation categories in pink, and test categories in blue. Each category indicates the number of associated classes from ILSVRC-12. Best viewed zoomed-in on electronic version.

A.3 Extra Experimental Results

A.3.1 Few-shot classification baselines

We provide baseline results on few-shot classification using 1-nearest neighbor and logistic regression with either pixel inputs or CNN features. Compared with the baselines, Regular ProtoNet performs significantly better on all three few-shot classification datasets.

	Omniglot	*mini*-ImageNet		*tiered*-ImageNet	
Models	1-shot	1-shot	5-shot	1-shot	5-shot
1-NN Pixel	40.39 ± 0.36	26.74 ± 0.48	31.43 ± 0.51	26.55 ± 0.50	30.79 ± 0.53
1-NN CNN rnd	59.55 ± 0.46	24.03 ± 0.38	27.54 ± 0.42	25.49 ± 0.45	30.01 ± 0.47
1-NN CNN pre	52.53 ± 0.51	32.90 ± 0.58	40.79 ± 0.76	32.76 ± 0.66	40.26 ± 0.67
LR Pixel	49.15 ± 0.39	24.50 ± 0.41	33.33 ± 0.68	25.70 ± 0.46	36.30 ± 0.62
LR CNN rnd	57.80 ± 0.45	24.10 ± 0.50	28.40 ± 0.42	26.55 ± 0.48	32.51 ± 0.52
LR CNN pre	48.49 ± 0.47	30.28 ± 0.54	40.27 ± 0.59	34.52 ± 0.68	43.58 ± 0.72
ProtoNet	**94.62 ± 0.09**	**43.61 ± 0.27**	**59.08 ± 0.22**	**46.52 ± 0.32**	**66.15 ± 0.34**

Table A.2: Few-shot learning baseline results using labeled/unlabeled splits. Baselines either takes inputs directly from the pixel space or use a CNN to extract features. "rnd" denotes using a randomly initialized CNN, and "pre" denotes using a CNN that is pretrained for supervised classification for all training classes.

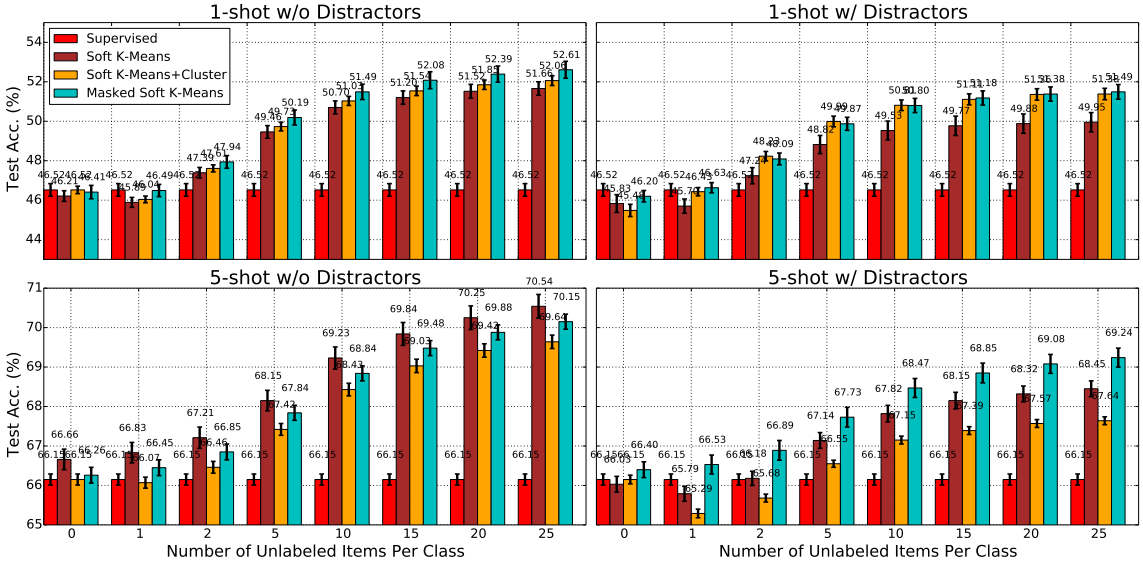


Figure A.2: Model Performance on *tiered*-ImageNet with different number of unlabeled items during test time. We include test accuracy numbers in this chart.

A.3.2 Number of unlabeled items

Figure A.2 shows test accuracy values with different number of unlabeled items during test time. Figure A.3 shows our mask output value distribution of the masked soft k-means model on Omniglot. The mask values have a bi-modal distribution, corresponding to distractor and non-distractor items.

A.4 Hyperparameter Details

For Omniglot, we adopted the best hyperparameter settings found for ordinary Prototypical Networks in Snell et al. (2017). In these settings, the learning rate was set to 1e-3, and cut in half every 2K updates starting at update 2K. We trained for a total of 20K updates. For *mini*-Imagenet and *tiered*-ImageNet, we trained with a starting learning rate of 1e-3, which we also decayed. We started the decay after 25K updates, and every 25K updates thereafter we cut it in half. We trained for a total of 200K updates. We used ADAM (Kingma and Ba, 2015) for the optimization of our models.

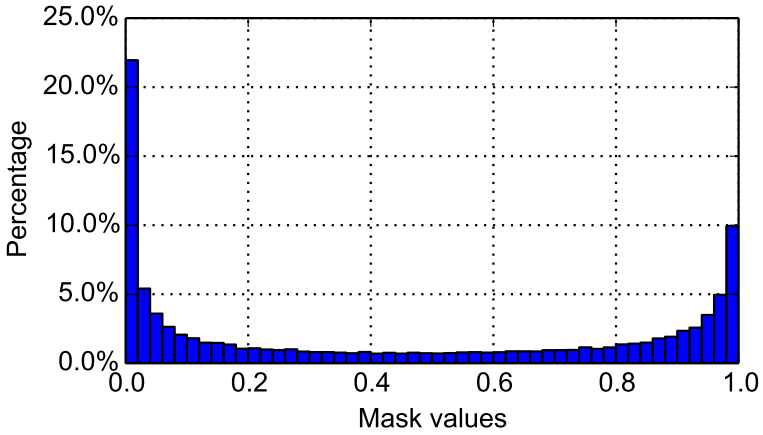


Figure A.3: Mask values predicted by masked soft k-means on Omniglot.

For the MLP used in the Masked Soft k-Means model, we use a single hidden layer with 20 hidden units with a tanh non-linearity for all 3 datasets. We did not tune the hyparameters of this MLP so better performance may be attained with a more rigorous hyperparameter search.

Appendix B

Appendix for Chapter 6

B.1 Dataset Details

B.1.1 RoamingOmniglot & RoamingImageNet Sampler Details

For the RoamingOmniglot and the RoamingImageNet experiments, we use sequences with maximum 150 images, from 5 environments. For individual environment, we use a Chinese restaurant process (Aldous, 1985) to sample the class distribution. In particular, the probability of sampling a new class is:

$$p_{\text{new}} = \frac{k\alpha + \theta}{m + \theta}, \tag{B.1}$$

where k is the number of classes that we have already sampled in the environment, and m is the total number of instances we have in the environment. α is set to 0.2 and θ is set to 1.0 in all experiments.

The environment switching is implemented by a Markov switching process. At each step in the sequence there is a constant probability p_{switch} that switches to another environment. For all experiments, we set p_{switch} to 0.2. We truncate the maximum number of appearances per class to 6. If the maximum appearance is reached, we will sample another class.

B.1.2 Additional RoamingRooms Statistics

Plots of additional statistics of RoamingRooms are shown in Figure B.1. In addition to the ones shown in the main paper, instances and viewpoints also follow long tail distributions. The number of objects in each frame follows an exponential distribution.

B.1.3 RoamingRooms Simulator Details

We generate our episodes with a two-stage process using two simulators – HabitatSim (Savva et al., 2019) and MatterSim (Anderson et al., 2018) – because HabitatSim is based on 3D meshes and using HabitatSim alone will result in poor image quality due to incorrect mesh reconstruction. Therefore we sacrificed the continuous movement of agents within HabitatSim and base our environment navigation on the discrete viewpoints in MatterSim, which is based on real panoramic images. The horizontal

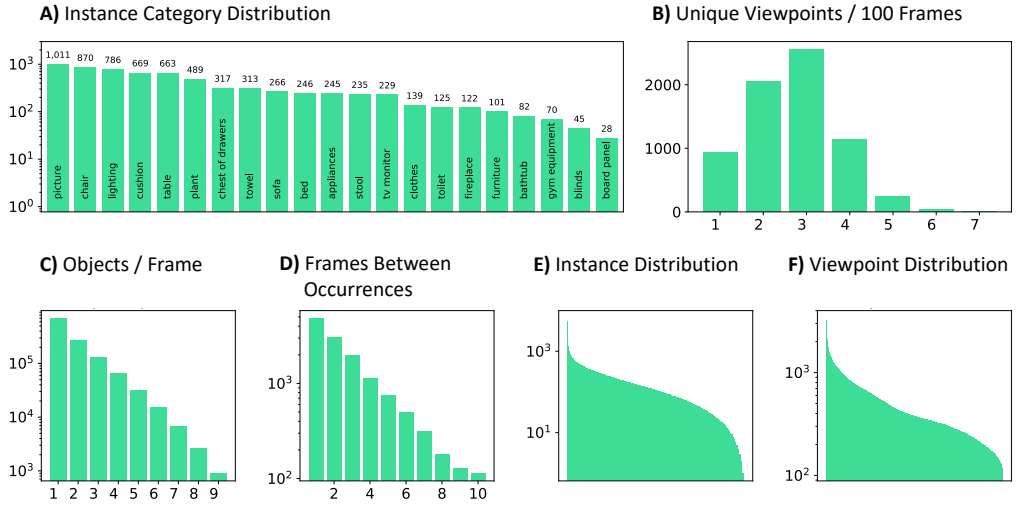


Figure B.1: Additional statistics about our RoamingRooms dataset.

field of view is set to 90 degrees for HabitatSim and 100 degrees for MatterSim, and we simulate with 800×600 resolution.

The first stage of generation involves randomly picking a sequence of viewpoints on the connectivity graph within MatterSim. For each viewpoint, the agent scans the environment along the yaw and pitch axes for a random period of time until a navigable viewpoint is within view. The time spent in a single viewpoint follows a Gaussian distribution with mean 5.0 and standard deviation 1.0. At the start of each new viewpoint, the agent randomly picks a direction to rotate and takes 12.5 degree steps along the yaw axis, and with 95% probability, a 5 degree rotation along the pitch axis is applied in a randomly chosen direction. When a navigable viewpoint is detected, the agent will navigate to the new viewpoint and reset the direction of scan. When multiple navigable viewpoints are present, the agent uniformly samples one.

In the second stage, an agent in HabitatSim retraces the viewpoint path and movements of the first stage generated by MatterSim, collecting mesh-rendered RGB and instance segmentation sensor data. The MatterSim RGB and HabitatSim RGB images are then aligned via FLANN-based feature matching (Muja and Lowe, 2009), resulting in an alignment matrix that is used to place the MatterSim RGB and HabitatSim instance segmentation maps into alignment. The sequence of these MatterSim RGB and HabitatSim instance segmentation maps constitute an episode.

We keep objects of the following categories: `picture, chair, lighting, cushion, table, plant, chest of drawers, towel, sofa, bed, appliances, stool, tv monitor, clothes, toilet, fireplace, furniture, bathtub, gym equipment, blinds, board panel`. We initially generate 600 frames per sequence and remove all the frames with no object. Then we store every 200 image frames into a separate file.

During training and evaluation, each video sequence is loaded, and for each image we go through each object present in the image. We create the attention map using the segmentation groundtruth of the selected object. The attention map and the image together form a *frame* in our model input. For training, we randomly crop 100 frames from the sequence, and for evaluation we use the first 100 frames for deterministic results.

B.1.4 Semi-Supervised Labels:

Here we describe how we sample the labeled vs. unlabeled flag for each example in the semi-supervised sequences in both RoamingOmniglot and RoamingRooms datasets. Due to the imbalance in our class distribution (from both the Chinese restaurant process and real data collection), directly masking the label may bias the model to ignore the rare seen classes. Ideally, we would like to preserve at least one labeled example for each class. Therefore, we designed the following procedure.

First, for each class k, suppose m_k is the number of examples in the sequence that belong to the class. Let α be the target label ratio. Then the class-specific label ratio α_k is:

$$\alpha_k = (1 - \alpha) \exp(-0.5(m_k - 1)) + \alpha. \tag{B.2}$$

We then for each class k, we sample a binary Bernoulli sequence based on $\text{Ber}(\alpha_k)$. If a class has all zeros in the Bernoulli sequence, we flip the flag of one of the instances to 1 to make sure there is at least one labeled instance for each class. For all experiments, we set $\alpha = 0.3$.

B.1.5 Dataset Splits

We include details about our dataset splits in Table B.1 and B.2.

B.2 Experiment Details

B.2.1 Network Architecture

For the RoamingOmniglot experiment we used the common 4-layer CNN for few-shot learning with 64 channels in each layer, resulting in a 64-d feature vector (Snell et al., 2017). For the RoamingRooms experiment we resize the input to 120×160 and we use the ResNet-12 architecture (Oreshkin et al., 2018) with {32,64,128,256} channels per block. To represent the feature of the input image with an attention mask, we concatenate the global average pooled feature with the attention ROI feature, resulting in a 512d feature vector. For the contextual RNN, in both experiments we used an LSTM (Hochreiter and Schmidhuber, 1997) with a 256d hidden state.

We use a linear layer to map from the output of the RNN to the features and control variables. We obtain $\gamma^{r,w}$ by adding 1.0 to the linear layer output and then applying the softplus activation. The bias units for $\beta^{r,w}$ are initialized to 10.0. We also apply the softplus activation to from the linear layer output.

B.2.2 Training Procedure

We use the Adam optimizer (Kingma and Ba, 2015) for all of our experiments, with a gradient cap of 5.0. For RoamingOmniglot we train the network for 40k steps with a batch size 32 and maximum sequence length 150 across 2 GPUs and an initial learning rate 2e-3 decayed by 0.1× at 20k and 30k steps. For RoamingRooms we train for 20k steps with a batch size 8 and maximum sequence length 100 across 4 GPUs and an initial learning rate 1e-3 decayed by 0.1× at 8k and 16k steps. We use the BCE coefficient $\lambda = 1$ for all experiments. In semi-supervised experiments, around 30% examples are labeled when the number of examples grows large ($\alpha = 0.3$, see Equation B.2). Early

Table B.1: Split information for *RoamingOmniglot*. Each column is an alphabet and we include all the characters in the alphabet in the split. Rows are continuation of lines.

Train	Angelic	Grantha	N Ko
	Aurek-Besh	Japanese (hiragana)	Malay
	Asomtavruli	Sanskrit	Ojibwe
	Korean	Arcadian	Greek
	Alphabet of the Magi	Blackfoot	Futurama
	Tagalog	Anglo-Saxon Futhorc	Braille
	Cyrillic	Burmese	Avesta
	Gujarati	Ge ez	Syriac (Estrangelo)
	Atlantean	Japanese (katakana)	Balinese
	Atemayar Qelisayer	Glagolitic	Tifinagh
	Latin	Inuktitut	
Val	Hebrew	Mkhedruli	Armenian
	Early Aramaic	Bengali	
Test	Gurmukhi	Kannada	Keble
	Malayalam	Manipuri	Mongolian
	Old Church Slavonic	Oriya	Syriac (Serto)
	Sylheti	Tengwar	Tibetan
	ULOG		

stopping is used in RoamingRooms experiments where the checkpoint with the highest validation AP score is chosen. For RoamingRooms, we sample Bernoulli sequences on unlabeled inputs to gradually allow semi-supervised writing to the prototype memory and we find it helps training stability. The probability starts with 0.0 and increase by 0.2 every 2000 training steps until reaching 1.0.

B.2.3 Data Augmentation

For RoamingOmniglot, we pad the 28×28 image to 32×32 and then apply random cropping.

For RoamingRooms, we apply random cropping in the time dimension to get a chunk of 100 frames per input example. We also apply random dropping of 5% of the frames. We pad the 120×160 images to 126 × 168 and apply random cropping in each image frame. We also randomly flip the order of the sequence (going forward or backward).

B.2.4 Spatiotemporal Context Experiment Details

We use the Kylberg texture dataset (Kylberg, 2011) without rotations. Texture classes are split into train, val, and test, defined in Table B.4. We resize all images first to 256×256. For each Omniglot image, a 28×28 patch is randomly cropped from a texture image to serve as background. Random Gaussian noises with mean zero and standard deviation 0.1 are added to the background images.

For spatial background experiments, we added an additional learnable network of the same size as the main network to take the background image as input, and output the same sized embedding vector. This embedding vector is further concatenated with the main embedding vector to form the final embedding of the input. We also found that using spatially overlayed images with a single CNN can achieve similar performance as well. The final numbers are reported using the concatenation approach since it is less prone to overlay noises and is more similar to the implementation we use in

Table B.2: Split information for *RoamingRooms*. Each column is the ID of an indoor world. Rows are continuation of the lines.

Train	r1Q1Z4BcV1o	JmbYfDe2QKZ	29hnd4uzFmX	ULsKaCPVFJR	E9uDoFAP3SH
	8WUmhLawc2A	Uxmj2M2itWa	mJXqzFtmKg4	V2XKFyX4ASd	EU6Fwq7SyZv
	gYvKGZ5eRqb	gxdoqLR6rwA	YFuZgdQ5vWj	gTV8FGcVJC9	sT4fr6TAbpF
	VVfe2KiqLaN	fzynW3qQPVF	WYY7iVyf5p8	VFuaQ6m2Qom	YmJkqBEsHnH
	2t7WUuJeko7	pLe4wQe7qrG	cV4RVeZvu5T	XcA2TqTSSAj	ur6pFq6Qu1A
	1pXnuDYAj8r	b8cTxDM8gDG	x8F5xyUWy9e	X7HyMhZNoso	aayBHfsNo7d
	TbHJrupSAjP	sKLMLpTHeUy	2azQ1b91cZZ	2n8kARJN3HM	Vvot9Ly1tCj
	S9hNv5qa7GM	EDJbREhghzL	qoiz87JEwZ2	q9vSo1VnCiC	Vt2qJdWjCF2
	VzqfbhrpDEA	D7G3Y4RVNrH	ZMojNkEp431	uNb9QFRL6hY	5LpN3gDmAk7
	rqfALeAoiTq	e9zR4mvMWw7	yqstnuAEVhm	zsNo4HB9uLZ	JF19kD82Mey
	759xd9YjKW5	wc2JMjhGNzB	rPc6DW4iMge	jh4fc5c5qoQ	HxpKQynjfin
	GdvgFV5R1Z5	kEZ7cmS4wCh	vyrNrziPKCB	D7N2EKCX4Sj	PX4nDJXEHrG
Val	s8pcmisQ38h	dhjEzFoUFzH	RPmz2sHmrrY	1LXtFkjw3qL	8194nk5LbLH
	jtcxE69GiFV	QUCTc6BB5sX	p5wJjkQkbXX	JeFG25nYj2p	82sE5b5pLXE
Test	oLBMNvg9in8	r47D5H71a5s	Z6MFQCViBuw	YVUC4YcDtcY	pRbA3pwrgk9
	SN83YJsR3w2	gZ6f7yhEvPG	ac26ZMwG7aT	7y3sRwLe3Va	B6ByNegPMKs
	UwV83HsGsw3	VLzqgDo317F	17DRP5sb8fy	pa4otMbVnkk	5ZKStnWn8Zo
	PuKPg4mmafe	Pm6F8kyY3z2	i5noydFURQK	ARNzJeq3xxb	5q7pvUzZiYa

Table B.3: *RoamingRooms* dataset split size

Split	Worlds	Sequences	Frames
Train	60	4,699	823,444
Val	20	725	125,823
Test	10	1,547	271,335
Total	90	6,971	1,220,602

the RoamingRooms experiments.

B.2.5 Baseline Implementation Details

Online meta-learning (OML): The OML model performs one gradient descent step for each input. In order for the model to predict unknown, we use the probability output from the softmax layer summing across the unused units. For example, if the softmax layer has 40 units and we have only seen 5 classes so far, then we sum the probability from the 6th to the last units. This summed probability is separately trained with a binary cross entropy, same as in Equation 6.9.

The inner learning rate is set to 1e-2 and we truncate the number of unrolled gradient descent steps to 5/20 (RoamingOmniglot/RoamingRooms), in order to make the computation feasible. For RoamingOmniglot, the network is trained with a batch size 32 across 2 GPUs, for a total of 20k steps, with an initial learning rate 2e-3 decayed by 0.1 at 10k and 16.7k steps. For RoamingRooms, the network is trained with a batch size 8 across 4 GPUs, for a total of 16k steps, with an initial learning rate 1e-3 decayed by 0.1 at 6.4k and 12.8k steps.

Table B.4: Split information for the Kylberg texture dataset. Each column is an texture type. Rows are continuation of lines.

Train	blanket2 pearlsugar1 sesameseeds1	ceiling2 rice2 stone1	floor2 scarf2 stoneslab1	grass1 screen1	linseeds1 seat2
Val	blanket1 rice1	canvas1 stone2	ceiling1	floor1	scarf1
Test	wall1 oatmeal1	lentils1 stone3	cushion1 seat1	rug1	sand1

Long short-term memory (LSTM): We apply a stacked two layer LSTM with 256 hidden dimensions. Inputs are $\mathbf{h}_t^{\text{CNN}}$ concatenated with the label one-hot vector. If an example is unlabeled, then the label vector is all-zero. We directly apply a linear layer on top of the LSTM to map the LSTM memory output into classification logits, and the last logit is the binary classification logit reserved for unknown. The training procedure is the same as our CPM model.

Differentiable neural computer (DNC): In order to make the DNC model work properly, we found that it is sometimes helpful to pretrain the CNN weights. Simply initializing from scratch and train CNN+DNC end-to-end sometimes results in poor performance. We hypothesize that the attention structure in the DNC model is detrimental to representation learning. Therefore, for RoamingOmniglot experiments, we use pretrained ProtoNet weights for solving 1-shot 5-way episodes to initialize the CNN, and we keep finetuning the CNN weights with 10% of the full learning rate. For RoamingRooms experiments, we train the full model end-to-end from scratch.

The DNC is also modified so that it is more effective using the label information from the input. In the original MANN paper (Santoro et al., 2016) for one-shot learning, the input features $\mathbf{h}_t^{\text{CNN}}$ and the label one-hot ID are simply concatenated to feed into the LSTM controller of MANN. We find that it is beneficial to directly add label one-hot vector as an input to the write head that generates the write attention and the write content. Similar to the LSTM model, the memory readout is also sent to a linear layer in order to get the final classification logits, and the last logit is the binary classification logit reserved for the unknowns. Finally we remove the linkage prediction part of the DNC due to training instability.

The controller LSTM has 256 hidden dimensions, and the memory has 64 slots each with 64 dimensions. There are 4 read heads and 4 write heads. The training procedure is the same as CPM.

Online ProtoNet: Online ProtoNet is our modification of the original ProtoNet (Snell et al., 2017). It is similar to our CPM model without the contextual RNN. The feature from the CNN is directly written to the prototype memory. In addition, we do not predict the control hyperparameters $\beta_t^{\{r,w\}}, \gamma_t^{\{r,w\}}$ from the RNN and they are learned as regular parameters. The training procedure is the same as CPM.

Online MatchingNet: Online MatchingNet is our modification of the original MatchingNet (Vinyals et al., 2016). We do not consider the context embedding in the MatchingNet paper since it was

originally designed for the entire episode using an attentional RNN encoder. It is similar to online ProtoNet but instead of doing online averaging, it directly stores each example and its class. Since it is an example-based storage, we did not extend it to learn from unlabeled examples, and all unlabeled examples are skipped. We use a similar decision rule to determine whether an example belongs to a known cluster by looking at the distance to the nearest exemplar stored in the memory, shifted by β and scaled by $1/\gamma$. Note that online MatchingNet is not efficient at memory storage since it scales with the number of steps in the sequence. In addition, we use the negative Euclidean distance as the similarity function. The training procedure is the same as CPM.

Online infinite mixture prototypes (IMP): Online IMP is proposed as a mix of prototype and example-based storage by allowing a class to have multiple clusters. If an example is classified as unknown or it is unlabeled, we will assign its cluster based on our prediction, which either assigns it to one of the existing clusters or creates a new cluster, depending on its distance to the nearest cluster. If a cluster with an unknown label later is assigned with an example with a known class, then the cluster label will also be updated. We use the same decision rule as online ProtoNet to determine whether an example belongs to a known cluster by looking at the distance to the nearest cluster, shifted by β and scaled by $1/\gamma$. As described above, online IMP has the capability of learning from unlabeled examples, unlike online MatchingNet. However similar to online MatchingNet, online IMP is also not efficient at memory storage since in the worst case it also scales with the number of steps in the sequence. Again, the training procedure is the same as CPM.

Appendix C

Appendix for Chapter 7

C.1 Method Derivation

C.1.1 E-Step

Inferring cluster assignments. The categorical variable $\hat{y}$ infers the cluster assignment of the current input example with regard to the existing clusters.

$$\hat{y}_{t,k} = \Pr(y_t = k|\mathbf{z}_t, u = 0) \tag{C.1}$$

$$= \frac{\Pr(\mathbf{z}_t|y_t = k, u = 0)\Pr(y_t = k)}{\Pr(\mathbf{z}_t, u = 0)} \tag{C.2}$$

$$= \frac{w_k f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2)}{\sum_{k'} w_{k'} f(\mathbf{z}_t; \mathbf{p}_{t,k'}, \sigma^2)} \tag{C.3}$$

$$= \frac{\exp(\log w_k - d(\mathbf{z}_t, \mathbf{p}_{t,k})/2\sigma^2)}{\sum_{k'} \exp(\log w'_k - d(\mathbf{z}_t, \mathbf{p}_{t,k'})/2\sigma^2)} \tag{C.4}$$

$$= \text{softmax}\left(\log w_k - d(\mathbf{z}_t, \mathbf{p}_{t,k})/2\sigma^2\right), \tag{C.5}$$

$$= \text{softmax}(v_{t,k}), \tag{C.6}$$

where w_k is the mixing coefficient of cluster k and $d(\cdot, \cdot)$ is the distance function, and $v_{t,k}$ is the logits. In our experiments, w_k's are kept as constant and τ is an independent learnable parameter.

Inference on unknown classes. The binary variable $\hat{u}$ estimates the probability that the current input belongs to a new cluster:

$$\hat{u}_t = \Pr(u_t = 1|\mathbf{z}_t) \tag{C.7}$$
$$= \frac{z_0 u_0}{z_0 u_0 + \sum_k w_k f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2)(1-u_0)} \tag{C.8}$$
$$= \frac{1}{1 + \frac{1}{z_0 u_0}\sum_k w_k f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2)(1-u_0)} \tag{C.9}$$
$$= \frac{1}{1 + \exp(\log(\frac{1}{z_0 u_0}\sum_k w_k f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2)(1-u_0)))} \tag{C.10}$$
$$= \frac{1}{1 + \exp(-\log(z_0) - \log(u_0) + \log(1-u_0) + \log(\sum_k w_k f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2))} \tag{C.11}$$
$$= \frac{1}{1 + \exp(-s + \log(\sum_k \exp(\log(w_k) - d(\mathbf{z}_t, \mathbf{p}_{t,k})/2\sigma^2)))} \tag{C.12}$$
$$= \text{sigmoid}(s - \log(\sum_k \exp(\log(w_k) - d(\mathbf{z}_t, \mathbf{p}_{t,k})/2\sigma^2))) \tag{C.13}$$
$$= \text{sigmoid}(s - \log(\sum_k \exp(v_{t,k}))), \tag{C.14}$$

where $\alpha = \log(z_0) + \log(u_0) - \log(1-u_0) + m\log(\sigma) + m\log(2\pi)/2$ and m is the input dimension. In our implementation, we use max here instead of logsumexp since we found max leads to better and more stable training performance empirically. It can be derived as a lower bound:

$$\hat{u}_t = \text{sigmoid}(s - \log(\sum_k \exp(\log(w_k) - d(\mathbf{z}_t, \mathbf{p}_{t,k})/2\sigma^2))) \tag{C.15}$$
$$\geq \text{sigmoid}(s - \log(\max_k \exp(-d(\mathbf{z}_t, \mathbf{p}_{t,k})/2\sigma^2))) \tag{C.16}$$
$$= \text{sigmoid}(s + \min_k d(\mathbf{z}_t, \mathbf{p}_{t,k})/2\sigma^2) \tag{C.17}$$
$$= \text{sigmoid}((\min_k d(\mathbf{z}_t, \mathbf{p}_{t,k}) - \beta)/\gamma), \tag{C.18}$$

where $\beta = -2s\sigma^2$, $\gamma = 2\sigma^2$. To make learning more flexible, we directly make β and γ as independent learnable parameters so that we can control the confidence level for predicting unknown classes.

C.1.2 M-Step

Here we infer the posterior distribution of the prototypes $\Pr(\mathbf{p}_{t,k}|\mathbf{z}_{1:t})$. We formulate an efficient recursive online update, similar to Kalman filtering, by incorporating the evidence of the current input $\mathbf{z}_t$ and avoiding re-clustering the entire input history. We define $\hat{\mathbf{p}}_{t,k}$ as the estimate of the posterior mean of the k-th cluster at time step t, and $\hat{\sigma}^2_{t,k}$ is the estimate of the posterior variance.

Updating prototypes. Suppose that in the E-step we have determined that $y_t = k$. Then the posterior distribution of the k-th cluster after observing $\mathbf{z}_t$ is:

$$\begin{aligned}
&\Pr(\mathbf{p}_{t,k}|\mathbf{z}_{1:t}, y_t = k) && \text{(C.19)}\\
\propto\ &\Pr(\mathbf{z}_t|\mathbf{p}_{t,k}, y_t = k)\Pr(\mathbf{p}_{t,k}|\mathbf{z}_{1:t-1}) && \text{(C.20)}\\
=\ &\Pr(\mathbf{z}_t|\mathbf{p}_{t,k}, y_t = k)\int_{\mathbf{p}'}\Pr(\mathbf{p}_{t,k}|\mathbf{p}_{t-1,k} = \mathbf{p}')\Pr(\mathbf{p}_{t-1,k} = \mathbf{p}'|\mathbf{z}_{1:t-1}) && \text{(C.21)}\\
\approx\ &f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2)\int_{\mathbf{p}'} f(\mathbf{p}_{t,k}; \mathbf{p}', \sigma^2_{t,d}) f(\mathbf{p}'; \hat{\mathbf{p}}_{t-1,k}, \hat{\sigma}^2_{t-1,k}) && \text{(C.22)}\\
=\ &f(\mathbf{z}_t; \mathbf{p}_{t,k}, \sigma^2) f(\mathbf{p}_{t,k}; \hat{\mathbf{p}}_{t-1,k}, \sigma^2_{t,d} + \hat{\sigma}^2_{t-1,k}). && \text{(C.23)}
\end{aligned}$$

If we assume that the transition probability distribution $\Pr(\mathbf{p}_{t,k}|\mathbf{p}_{t-1,k})$ is a zero-mean Gaussian with variance $\sigma^2_{t,d} = (1/\rho - 1)\hat{\sigma}^2_{t-1,k}$, where $\rho \in (0, 1]$ is some constant that we defined to be the memory decay coefficient, then the posterior estimates are:

$$\hat{\mathbf{p}}_{t,k}|_{y_t=k} = \frac{\mathbf{z}_t\hat{\sigma}^2_{t-1,k}/\rho + \hat{\mathbf{p}}_{t-1,k}\sigma^2}{\sigma^2 + \hat{\sigma}^2_{t-1,k}/\rho}, \quad \hat{\sigma}^2_{t,k}|_{y_t=k} = \frac{\sigma^2\hat{\sigma}^2_{t-1,k}/\rho}{\sigma^2 + \hat{\sigma}^2_{t-1,k}/\rho}. \tag{C.24}$$

If $\sigma^2 = 1$, and $\hat{c}_{t,k} \equiv 1/\hat{\sigma}^2_{t,k}$, $\hat{c}_{t-1,k} \equiv 1/\hat{\sigma}^2_{t-1,k}$, it turns out we can formulate the update equation as follows, and $\hat{c}_{t,k}$ can be viewed as a count variable for the number of elements in each estimated cluster, subject to the decay factor ρ over time:

$$\hat{c}_{t,k}|_{y_t=k} = \rho\hat{c}_{t-1,k} + 1, \tag{C.25}$$

$$\hat{\mathbf{p}}_{t,k}|_{y_t=k} = \mathbf{z}_t\frac{1}{\hat{c}_{t,k}|_{y_t=k}} + \hat{\mathbf{p}}_{t-1,k}\frac{\rho\hat{c}_{t-1,k}}{\hat{c}_{t,k}|_{y_t=k}}. \tag{C.26}$$

If $y_t \neq k$, then the prototype posterior distribution simply gets diffused at timestep t:

$$\Pr(\mathbf{p}_{t,k}|z_{1:t}, y_t \neq k) \approx f(\mathbf{p}_{t,k}; \hat{\mathbf{p}}_{t-1,k}, \hat{\sigma}^2_{t-1,k}/\rho) \tag{C.27}$$

$$\hat{c}_{t,k}|_{y_t\neq k} = \rho\hat{c}_{t-1,k}, \tag{C.28}$$

$$\hat{\mathbf{p}}_{t,k}|_{y_t\neq k} = \hat{\mathbf{p}}_{t-1,k}. \tag{C.29}$$

Finally, our posterior estimates at time t are computed by taking the expectation over y_t:

$$\hat{c}_{t,k} = \mathop{\mathbb{E}}_{y_t}[\hat{c}_{t,k}|_{y_t}] \tag{C.30}$$

$$= \hat{c}_{t,k}|_{y_t=k} \Pr(y_t = k|\mathbf{z}_t) + \hat{c}_{t,k}|_{y_t\neq k} \Pr(y_t \neq k|\mathbf{z}_t) \tag{C.31}$$

$$= (\rho\hat{c}_{t-1,k} + 1)\hat{y}_{t,k}(1 - \hat{u}_{t,k}) + \rho\hat{c}_{t-1,k}(1 - \hat{y}_{t,k}(1 - \hat{u}_{t,k})), \tag{C.32}$$

$$= \rho\hat{c}_{t-1,k} + \hat{y}_{t,k}(1 - \hat{u}_{t,k}), \tag{C.33}$$

$$\hat{\mathbf{p}}_{t,k} = \mathop{\mathbb{E}}_{y_t}[\hat{\mathbf{p}}_{t,k}|_{y_t}] \tag{C.34}$$

$$= \hat{\mathbf{p}}_{t,k}|_{y_t=k} \Pr(y_t = k|\mathbf{z}_t) + \hat{\mathbf{p}}_{t,k}|_{y_t\neq k} \Pr(y_t \neq k|\mathbf{z}_t) \tag{C.35}$$

$$= \left(\mathbf{z}_t \frac{1}{\hat{c}_{t,k}|_{y_t=k}} + \hat{\mathbf{p}}_{t-1,k} \frac{\rho\hat{c}_{t-1,k}}{\hat{c}_{t,k}|_{y_t=k}}\right) \hat{y}_{t,k}(1 - \hat{u}_{t,k}) + \hat{\mathbf{p}}_{t-1,k}(1 - \hat{y}_{t,k}(1 - \hat{u}_{t,k})) \tag{C.36}$$

$$= \mathbf{z}_t \frac{\hat{y}_{t,k}(1 - \hat{u}_{t,k})}{\rho\hat{c}_{t-1,k} + 1} + \hat{\mathbf{p}}_{t-1,k} \left(1 - \hat{y}_{t,k}(1 - \hat{u}_{t,k}) + \hat{y}_{t,k}(1 - \hat{u}_{t,k}) \frac{\rho\hat{c}_{t-1,k}}{\rho\hat{c}_{t-1,k} + 1}\right) \tag{C.37}$$

$$= \mathbf{z}_t \frac{\hat{y}_{t,k}(1 - \hat{u}_{t,k})}{\rho\hat{c}_{t-1,k} + 1} + \hat{\mathbf{p}}_{t-1,k} \left(1 - \frac{\hat{y}_{t,k}(1 - \hat{u}_{t,k})}{\rho\hat{c}_{t-1,k} + 1}\right). \tag{C.38}$$

Since $\hat{c}_{t,k}$ is also our estimate on the number of elements in each cluster, we can use it to estimate the mixture weights,

$$\hat{w}_{t,k} = \frac{\hat{c}_{t,k}}{\sum_{k'} \hat{c}_{t,k}}. \tag{C.39}$$

Note that in our experiments the mixture weights are not used and we assume that each cluster has an equal mixture probability.

C.2 Experiment Details

We provide additional implementation details in Tab. C.1, C.2 and C.3.

Table C.1: Hyperparameter settings for *RoamingRooms*

Hyperparameter	Values
Random crop area range	0.08 - 1.0
Random color strength	0.5
Backbone	ResNet-12 (Oreshkin et al., 2018)
Num channels	[32, 64, 128, 256]
τ init	0.1
β init	-12.0
γ init	1.0
Num prototypes K	150
Memory decay ρ	0.995
Sequence length / batch size	50 (100 eval)
Beta mean μ	0.5
Entropy loss λ_{ent}	0.0
New cluster loss λ_{new}	0.5
Threshold α	0.5
Pseudo label temperature ratio $\tilde{\tau}/\tau$	0.1
Learning rate schedule	[40k, 60k, 80k]
Learning rate	[1e-3, 1e-4, 1e-5]
Optimizer	Adam

Table C.2: Hyperparameter settings for *RoamingOmniglot*

Hyperparameter	Values
Random crop area range	0.08 - 1.0
Random color	None
Backbone	Conv-4 (Vinyals et al., 2016)
Num channels	[64, 64, 64, 64]
τ init	0.1
β init	-12.0
γ init	1.0
Num prototypes K	150
Memory decay ρ	0.995
Sequence length / batch size	150
Beta mean μ	0.5
Entropy loss λ_{ent}	1.0
New cluster loss λ_{new}	1.0
Threshold α	0.5
Pseudo label temperature ratio $\tilde{\tau}/\tau$	0.2
Learning rate schedule	[40k, 60k, 80k]
Learning rate	[1e-3, 1e-4, 1e-5]
Optimizer	Adam

Table C.4: Unsupervised iid learning on Omniglot using an MLP

Method	3-NN Error	5-NN Error	10-NN Error
VAE (Joo et al., 2020)	92.34±0.25	91.21±0.18	88.79±0.35
SBVAE (Joo et al., 2020)	86.90±0.82	85.10±0.89	82.96±0.64
DirVAE (Joo et al., 2020)	76.55±0.23	73.81±0.29	70.95±0.29
CURL (Rao et al., 2019)	78.18±0.47	75.41±0.34	72.51±0.46
SimCLR (Chen et al., 2020a)	44.35±0.55	42.99±0.55	44.93±0.55
SwAV (Caron et al., 2020)	**42.66±0.55**	**42.08±0.55**	44.78±0.55
OUPN (Ours)	43.75±0.55	**42.13±0.55**	**43.88±0.55**

Table C.3: Hyperparameter settings for *RoamingImageNet*

Hyperparameter	Values
Random crop area range	0.08 - 1.0
Random color strength	0.5
Backbone	ResNet-12 (Oreshkin et al., 2018)
Num channels	[32, 64, 128, 256]
τ init	0.1
β init	-12.0
γ init	1.0
Num prototypes K	600
Memory decay ρ	0.99
Sequence length / batch size	48
Beta mean μ	0.5
Entropy loss λ_{ent}	0.5
New cluster loss λ_{new}	0.5
Threshold α	0.5
Pseudo label temperature ratio $\tilde{\tau}/\tau$	0.0 (i.e. one-hot pseudo labels)
Learning rate schedule	[40k, 60k, 80k]
Learning rate	[1e-3, 1e-4, 1e-5]
Optimizer	Adam

C.2.1 Metric Details

For each method, we used the same nearest centroid algorithm for online clustering. For unsupervised readout, at each timestep, if the closest centroid is within threshold α, then we assign the new example to the cluster, otherwise we create a new cluster. For supervised readout, we assign examples based on the class label, and we create a new cluster if and only if the label is a new class. Both readout will provide us a sequence of class IDs, and we will use the following metrics to compare our predicted class IDs and groundtruth class IDs. Both metrics are designed to be threshold invariant.

For unsupervised evaluation, we consider the adjusted mutual information score. Suppose we have two clustering $U = \{U_i\}$ and $V = \{V_j\}$, and U_i and V_j are set of example IDs, and N is the total number of examples. U and V can be viewed as discrete probability distribution over cluster

Table C.5: Effect of threshold α

α	*RoamingOmniglot* AMI	*RoamingOmniglot* AP	*RoamingRooms* AMI	*RoamingRooms* AP
0.3	82.75	90.57	52.60	58.71
0.4	81.59	90.94	59.69	66.11
0.5	**89.65**	**95.22**	**77.96**	**84.34**
0.6	87.01	93.87	64.65	69.49
0.7	86.08	92.94	66.60	73.54

Table C.6: Effect of $\tilde{\tau}$

$\tilde{\tau}$ / τ	*RoamingOmniglot* AMI	*RoamingOmniglot* AP	*RoamingRooms* AMI	*RoamingRooms* AP
0.05	89.23	95.01	77.44	84.38
0.10	89.71	95.21	**77.89**	**84.99**
0.20	**89.78**	**95.31**	77.82	84.57
0.50	89.40	95.13	76.81	83.90
1.00	89.62	95.16	0.00	19.91

Table C.7: Effect of λ_{ent}

λ_{ent}	*RoamingOmniglot* AMI	*RoamingOmniglot* AP	*RoamingRooms* AMI	*RoamingRooms* AP
0.00	82.45	90.66	**76.64**	**84.11**
0.25	87.31	93.85	76.61	83.16
0.50	87.98	94.21	75.46	81.78
0.75	88.77	94.74	74.76	79.91
1.00	**89.70**	**95.14**	75.32	80.29

Table C.8: Effect of mean μ of the Beta prior

μ	*RoamingOmniglot* AMI	*RoamingOmniglot* AP	*RoamingRooms* AMI	*RoamingRooms* AP
0.3	84.14	93.19	68.75	72.58
0.4	86.59	93.10	69.19	73.86
0.5	**89.89**	**95.24**	**77.61**	**84.64**
0.6	85.93	93.81	64.21	73.23
0.7	26.22	92.08	48.58	64.28

IDs. Therefore, the mutual information score between U and V is:

$$\text{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right) \tag{C.40}$$

$$= \sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N} \log\left(\frac{N n_{ij}}{a_i b_j}\right). \tag{C.41}$$

The adjusted MI score[1] normalizes the range between 0 and 1, and subtracts the baseline from random chance:

$$\text{AMI}(U,V) = \frac{MI(U,V) - \mathbb{E}[MI(U,V)]}{\frac{1}{2}(H(U)+H(V)) - \mathbb{E}[MI(U,V)]}, \tag{C.42}$$

where $H(\cdot)$ denotes the entropy function, and $\mathbb{E}[MI(U,V)]$ is the expected mutual information by chance [2]. Finally, we sweep the threshold α to get a threshold invariant score:

$$\text{AMI}_{\max} = \max_{\alpha} \text{AMI}(y, \hat{y}(\alpha)). \tag{C.43}$$

C.3 Additional Experimental Results

C.3.1 Comparison to Reconstruction-Based Methods

We additionally provide Tab. C.4 to show a comparison with CURL (Rao et al., 2019) in the iid setting. We used the same MLP architecture and applied it on the Omniglot dataset using the same data split. Reconstruction-based methods lag far behind self-supervised learning methods. Our method is on par with SimCLR and SwAV.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html`
[2] `https://en.wikipedia.org/wiki/Adjusted_mutual_information`

C.3.2 Additional Studies on Hyperparameters

Tab. C.5, C.6, C.7, and C.8 show additional studies on the effect of the cluster creation threshold α, the temperature parameter $\tilde{\tau}$, the entropy loss parameter λ_{ent}, and the Beta prior mean. The Beta mean μ is computed as the following: Suppose a and b are the parameters of the Beta distribution, and μ is the mean. We fix $a = 4\mu$ and $b = 4 - a$.

C.3.3 Large Batch IID Results on *RoamingOmniglot* and *RoamingImageNet*

In Tab. C.9 and C.10, we provide results on training SimCLR and SwAV under the traditional setting of using large iid batches, whereas our model is still trained on non-iid batches. We are able to outperform these iid batch models on *RoamingOmniglot*, but there is still a gap on *RoamingImageNet*. It is worth mention that both SimCLR and SwAV are well tuned towards large batch settings on ImageNet, and in order to run these experiments on *RoamingImageNet*, both used 8 GPUs in parallel for a total batch size of 2048, which is much more computationally intensive than our model, which only used a single GPU. However, this also suggests that our method may have some limitation towards handling larger intra-class variance, and it is more challenging to assume the clustering latent structure on top of these embeddings.

Table C.9: Large batch iid results on *RoamingOmniglot*

	Batch size	AMI	AP
Random Network	-	17.66	17.01
SimCLR (iid)	512	58.63	78.46
SwAV (iid)	512	64.63	84.77
OUPN (Ours) (non-iid)	150	84.42	92.84

Table C.10: Large batch iid results on *RoamingImageNet*

	Batch size	AMI	AP
Random Network	-	4.55	2.65
SimCLR (iid)	2048	28.40	22.48
SwAV (iid)	2048	28.85	23.35
OUPN (Ours) (non-iid)	48	19.03	15.05

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P. A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

Aldous, D. J. (1985). Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII—1983*, pages 1–198. Springer.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019). Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS)*.

Allen, K. R., Shelhamer, E., Shin, H., and Tenenbaum, J. B. (2019). Infinite mixture prototypes for few-shot learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings of the 1st International Conference on Neural Networks*, volume 2, pages 609–618. IEEE.

Anderson, J. R. (1991). The adaptive nature of human categorization. *Psychological review*, 98(3):409.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. (2018). Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Angluin, D. and Laird, P. (1988). Learning from noisy examples. *Machine Learning*, 2(4):343–370.

Antoniou, A., Patacchiola, M., Ochal, M., and Storkey, A. J. (2020). Defining benchmarks for continual few-shot learning. *CoRR*, abs/2004.11967.

Antoniou, A. and Storkey, A. J. (2019). Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *CoRR*, abs/1902.09884.

Arazo, E., Ortego, D., Albert, P., O'Connor, N. E., and McGuinness, K. (2019). Unsupervised label noise modeling and loss correction. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Asano, Y. M., Rupprecht, C., and Vedaldi, A. (2020). Self-labelling via simultaneous clustering and representation learning. In *8th International Conference on Learning Representations (ICLR)*.

Athiwaratkun, B., Finzi, M., Izmailov, P., and Wilson, A. G. (2018). Improving consistency-based semi-supervised learning with weight averaging. *CoRR*, abs/1806.05594.

Aygün, M., Osep, A., Weber, M., Maximov, M., Stachniss, C., Behley, J., and Leal-Taixé, L. (2021). 4d panoptic lidar segmentation.

Azadi, S., Feng, J., Jegelka, S., and Darrell, T. (2016). Auxiliary image regularization for deep cnns with noisy labels. In *4th International Conference on Learning Representation (ICLR)*.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. C. (2019). Systematic generalization: What is required and can it be learned? In *7th International Conference on Learning Representations (ICLR)*.

Balaji, Y., Sankaranarayanan, S., and Chellappa, R. (2018). Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Bellet, A., Habrard, A., and Sebban, M. (2013). A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*.

Bendale, A. and Boult, T. E. (2016). Towards open set deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. (2020). Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *8th International Conference on Learning Representations (ICLR)*.

Berthelot, D., Carlini, N., Goodfellow, I. J., Papernot, N., Oliver, A., and Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Bertinetto, L., Henriques, J. F., Torr, P. H. S., and Vedaldi, A. (2019). Meta-learning with differentiable closed-form solvers. In *7th International Conference on Learning Representations (ICLR)*.

Boudiaf, M., Ziko, I. M., Rony, J., Dolz, J., Piantanida, P., and Ayed, I. B. (2020). Information maximization for few-shot learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, (NeurIPS)*.

Bridle, J. S., Heading, A. J. R., and MacKay, D. J. C. (1991). Unsupervised classifiers, mutual information and 'phantom targets'. In *Advances in Neural Information Processing Systems 4 (NIPS)*.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*.

Caccia, M., Rodríguez, P., Ostapenko, O., Normandin, F., Lin, M., Caccia, L., Laradji, I. H., Rish, I., Lacoste, A., Vázquez, D., and Charlin, L. (2020). Online fast adaptation and knowledge accumulation: a new approach to continual learning. *CoRR*, abs/2003.05856.

Cao, K., Wei, C., Gaidon, A., Aréchiga, N., and Ma, T. (2019). Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *15th European Conference on Computer Vision (ECCV)*.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Carpenter, G. A. and Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115.

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-end incremental learning. In *15th European Conference on Computer Vision (ECCV)*.

Cermelli, F., Mancini, M., Xian, Y., Akata, Z., and Caputo, B. (2020). A few guidelines for incremental few-shot segmentation. *CoRR*, abs/2012.01415.

Chang, A. X., Dai, A., Funkhouser, T. A., Halber, M., Nießner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017a). Matterport3d: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision (3DV)*.

Chang, H., Learned-Miller, E. G., and McCallum, A. (2017b). Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems (NIPS)*.

Chapelle, O., Schölkopf, B., and Zien, A. (2010). *Semi-Supervised Learning*. The MIT Press, 1st edition.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019a). Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations (ICLR)*.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. (2019b). Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357.

Chawla, S. and Gionis, A. (2013). k-means–: A unified approach to clustering and outlier detection. In *Proceedings of the 2013 SIAM International Conference on Data Mining (ICDM)*, pages 189–197. SIAM.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. (2020a). A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Chen, W., Liu, Y., Kira, Z., Wang, Y. F., and Huang, J. (2019). A closer look at few-shot classification. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Chen, X., Fan, H., Girshick, R. B., and He, K. (2020b). Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297.

Chen, X. and Gupta, A. (2015). Webly supervised learning of convolutional networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*.

Chen, X. and He, K. (2020). Exploring simple siamese representation learning. *CoRR*, abs/2011.10566.

Chen, Z., Maji, S., and Learned-Miller, E. G. (2020c). Shot in the dark: Few-shot learning with no base-class labels. *CoRR*, abs/2010.02430.

Choi, W., Chao, Y., Pantofaru, C., and Savarese, S. (2013). Understanding indoor scenes using 3d geometric phrases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cohen, N. and Squire, L. (1980). Preserved learning and retention of pattern-analyzing skill in amnesia: dissociation of knowing how and knowing that. *Science*, 210(4466):207–210.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cui, Y., Jia, M., Lin, T., Song, Y., and Belongie, S. J. (2019). Class-balanced loss based on effective number of samples. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cunningham, J. P. and Ghahramani, Z. (2015). Linear dimensionality reduction: survey, insights, and generalizations. *J. Mach. Learn. Res.*, 16:2859–2900.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems (NIPS)*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Denton, E. and Fergus, R. (2018). Stochastic video generation with a learned prior. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics.

Devries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552.

Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. (2020). A baseline for few-shot image classification. In *8th International Conference on Learning Representations (ICLR)*.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *2015 IEEE International Conference on Computer Vision (ICCV)*.

Dong, Q., Gong, S., and Zhu, X. (2017). Class rectification hard mining for imbalanced deep learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Duff, M. C., Covington, N. V., Hilverman, C., and Cohen, N. J. (2020). Semantic memory and the hippocampus: Revisiting, reaffirming, and extending the reach of their critical relationship. *Frontiers in Human Neuroscience*, 13:471.

Dvornik, N., Schmid, C., and Mairal, J. (2020). Selecting relevant features from a multi-domain representation for few-shot classification. In *16th European Conference on Computer Vision (ECCV)*.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. (2017). Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Finn, C. and Levine, S. (2018). Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *6th International Conference on Learning Representations (ICLR)*.

Finn, C., Rajeswaran, A., Kakade, S. M., and Levine, S. (2019). Online meta-learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139.

Fu, Y., Hospedales, T. M., Xiang, T., and Gong, S. (2015). Transductive multi-view zero-shot learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(11):2332–2345.

Garcia, V. and Bruna, J. (2018). Few-shot learning with graph neural networks. In *6th International Conference on Learning Representations (ICLR)*.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. M. A. (2018). Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. (2019). Boosting few-shot visual learning with self-supervision. In *ICCV*.

Gidaris, S. and Komodakis, N. (2018). Dynamic few-shot visual learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *6th International Conference on Learning Representations (ICLR)*.

Goldberger, J. and Ben-Reuven, E. (2017). Training deep neural-networks using a noise adaptation layer. In *5th International Conference on Learning Representation (ICLR)*.

Goldberger, J., Roweis, S. T., Hinton, G. E., and Salakhutdinov, R. (2004). Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems (NIPS)*.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems (NIPS)*.

Goyal, P., Mahajan, D., Gupta, A., and Misra, I. (2019). Scaling and benchmarking self-supervised visual representation learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. L. (2018). Recasting gradient-based meta-learning as hierarchical bayes. In *6th International Conference on Learning Representations (ICLR)*.

Graves, A., Mohamed, A., and Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP)*.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent - A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Guo, Y., Codella, N. C. F., Karlinsky, L., Smith, J. R., Rosing, T., and Feris, R. S. (2020). A broader study of cross-domain few-shot learning. In *14th European Conference on Computer Vision, ECCV*.

Gupta, S., Kumar, R., Lu, K., Moseley, B., and Vassilvitskii, S. (2017). Local search methods for k-means with outliers. *Proceedings of the VLDB Endowment*, 10(7):757–768.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I. W., and Sugiyama, M. (2018). Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, (NeurIPS)*.

Hariharan, B. and Girshick, R. B. (2017). Low-shot visual recognition by shrinking and hallucinating features. In *IEEE International Conference on Computer Vision (ICCV)*.

Harrison, J., Sharma, A., Finn, C., and Pavone, M. (2019). Continuous meta-learning without tasks. *CoRR*, abs/1912.08866.

Hautamäki, V., Cherednichenko, S., Kärkkäinen, I., Kinnunen, T., and Fränti, P. (2005). Improving k-means by outlier removal. In *Scandinavian Conference on Image Analysis*, pages 978–987. Springer.

Hayes, T. L., Cahill, N. D., and Kanan, C. (2019). Memory efficient experience replay for streaming learning. In *International Conference on Robotics and Automation (ICRA)*.

Hayes, T. L., Kafle, K., Shrestha, R., Acharya, M., and Kanan, C. (2020). REMIND your neural network to prevent catastrophic forgetting. In *16th European Conference on Computer Vision (ECCV)*.

He, J., Lehrmann, A. M., Marino, J., Mori, G., and Sigal, L. (2018). Probabilistic video generation using holistic attribute control. In *15th European Conference on Computer Vision (ECCV)*.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. (2020). Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. (2018). Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Hinton, G. E., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Horn, G. V., Aodha, O. M., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. J. (2018). The inaturalist species classification and detection dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hornik, K., Stinchcombe, M. B., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Howard, M. W. (2017). Temporal and spatial context in the mind and brain. *Current opinion in behavioral sciences*, 17:14–19.

Hsu, K., Levine, S., and Finn, C. (2019). Unsupervised learning via meta-learning. In *7th International Conference on Learning Representations (ICLR)*.

Hu, S. X., Moreno, P. G., Xiao, Y., Shen, X., Obozinski, G., Lawrence, N. D., and Damianou, A. C. (2020). Empirical bayes transductive meta-learning with synthetic gradients. In *8th International Conference on Learning Representations (ICLR)*.

Huang, C., Li, Y., Loy, C. C., and Tang, X. (2016). Learning deep representation for imbalanced classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Huang, G., Larochelle, H., and Lacoste-Julien, S. (2019). Centroid networks for few-shot clustering and unsupervised few-shot classification. *CoRR*, abs/1902.08605.

Hughes, M. C. and Sudderth, E. B. (2013). Memoized online variational inference for dirichlet process mixture models. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems (NIPS)*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Jamal, M. A., Brown, M., Yang, M., Wang, L., and Gong, B. (2020). Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Javed, K. and White, M. (2019). Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Jerfel, G., Grant, E., Griffiths, T., and Heller, K. A. (2019). Reconciling meta-learning and continual learning with online mixtures of tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Jiang, L., Meng, D., Zhao, Q., Shan, S., and Hauptmann, A. G. (2015). Self-paced curriculum learning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

Jiang, L., Zhou, Z., Leung, T., Li, L., and Fei-Fei, L. (2018). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*.

Johnson, M. J., Duvenaud, D., Wiltschko, A. B., Adams, R. P., and Datta, S. R. (2016). Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS)*.

Joo, W., Lee, W., Park, S., and Moon, I. (2020). Dirichlet variational autoencoder. *Pattern Recognit.*, 107:107514.

Kahana, M. J. (2012). *Foundations of human memory*. Oxford University Press, New York. OCLC: ocn744297060.

Kaiser, L., Nachum, O., Roy, A., and Bengio, S. (2017). Learning to remember rare events. In *5th International Conference on Learning Representations (ICLR)*.

Kemker, R. and Kanan, C. (2018). Fearnet: Brain-inspired model for incremental learning. In *6th International Conference on Learning Representations (ICLR)*.

Khan, S. H., Bennamoun, M., Sohel, F. A., and Togneri, R. (2015). Cost sensitive learning of deep feature representations from imbalanced data. *CoRR*, abs/1508.03422.

Khodadadeh, S., Bölöni, L., and Shah, M. (2019). Unsupervised meta-learning for few-shot image classification. In *NeurIPS*.

Kim, C. D., Jeong, J., and Kim, G. (2020). Imbalanced continual learning with partitioning reservoir sampling. In *16th European Conference on Computer Vision (ECCV)*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015 (NIPS)*.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Krishna, R., Chen, V. S., Varma, P., Bernstein, M., Ré, C., and Li, F. (2019). Scene graph prediction with limited labels. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L., Shamma, D. A., Bernstein, M. S., and Fei-Fei, L. (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vis.*, 123(1):32–73.

Krishnan, R. G., Shalit, U., and Sontag, D. A. (2015). Deep kalman filters. *CoRR*, abs/1511.05121.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems (NIPS)*.

Kumar, M. P., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems (NIPS)*.

Kylberg, G. (2011). *Kylberg Texture Dataset v. 1.0*. Centre for Image Analysis, Swedish University of Agricultural Sciences and ....

Lake, B. M., Salakhutdinov, R., Gross, J., and Tenenbaum, J. B. (2011). One shot learning of simple visual concepts. In *Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci 2011, Boston, Massachusetts, USA, July 20-23, 2011*.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lake, B. M., Vallabha, G. K., and McClelland, J. L. (2009). Modeling unsupervised perceptual category learning. *IEEE Trans. Auton. Ment. Dev.*, 1(1):35–43.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, S., Kim, J., Jun, J., Ha, J., and Zhang, B. (2017). Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40.

Li, D. and Hospedales, T. M. (2020). Online meta-learning for multi-source and semi-supervised domain adaptation. In *16th European Conference on Computer Vision (ECCV)*.

Li, F., Fergus, R., and Perona, P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.*, 106(1):59–70.

Li, J., Socher, R., and Hoi, S. C. H. (2020). Dividemix: Learning with noisy labels as semi-supervised learning. In *8th International Conference on Learning Representations (ICLR)*.

Li, J., Wong, Y., Zhao, Q., and Kankanhalli, M. S. (2019a). Learning to learn from noisy labeled data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, J., Zhou, P., Xiong, C., Socher, R., and Hoi, S. C. H. (2021). Prototypical contrastive learning of unsupervised representations. In *9th International Conference on Learning Representations (ICLR)*.

Li, X., Sun, Q., Liu, Y., Zhou, Q., Zheng, S., Chua, T., and Schiele, B. (2019b). Learning to self-train for semi-supervised few-shot classification. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, L. (2017). Learning from noisy labels with distillation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Li, Z. and Hoiem, D. (2018). Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947.

Liang, M., Yang, B., Chen, Y., Hu, R., and Urtasun, R. (2019). Multi-task multi-sensor fusion for 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liao, R., Xiong, Y., Fetaya, E., Zhang, L., Yoon, K., Pitkow, X., Urtasun, R., and Zemel, R. S. (2018). Reviving and improving recurrent back-propagation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Liu, H., Simonyan, K., and Yang, Y. (2019a). DARTS: differentiable architecture search. In *7th International Conference on Learning Representations ICLR*.

Liu, J., Song, L., and Qin, Y. (2020a). Prototype rectification for few-shot learning. In *16th European Conference on Computer Vision (ECCV)*.

Liu, L., Hamilton, W., Long, G., Jiang, J., and Larochelle, H. (2021). A universal representation transformer layer for few-shot image classification. In *9th International Conference on Learning Representations (ICLR)*.

Liu, Q., Majumder, O., Achille, A., Ravichandran, A., Bhotika, R., and Soatto, S. (2020b). Incremental few-shot meta-learning via indirect discriminant alignment. In *16th European Conference on Computer Vision (ECCV)*.

Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S. J., and Yang, Y. (2019b). Learning to propagate labels: Transductive propagation network for few-shot learning. In *7th International Conference on Learning Representations (ICLR)*.

Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., and Yu, S. X. (2019c). Large-scale long-tailed recognition in an open world. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Lomonaco, V. and Maltoni, D. (2017). Core50: a new dataset and benchmark for continuous object recognition. In *1st Annual Conference on Robot Learning (CoRL)*.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*.

Love, B. C., Medin, D. L., and Gureckis, T. M. (2004). Sustain: a network model of category learning. *Psychological review*, 111(2):309.

Lucas, J., Ren, M., Kameni, I., Pitassi, T., and Zemel, R. S. (2021). Theoretical bounds on estimation error for meta-learning. In *9th International Conference on Learning Representations (ICLR)*.

Ma, F., Meng, D., Xie, Q., Li, Z., and Dong, X. (2017). Self-paced co-training. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Mai, Z., Kim, H., Jeong, J., and Sanner, S. (2020). Batch-level experience replay with review for continual learning. *CoRR*, abs/2007.05683.

Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Mallya, A. and Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

McClelland, J., McNaughton, B., and O'Reilly, R. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102:419–57.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

McLachlan, G. J. and Basford, K. E. (1988). *Mixture models: Inference and applications to clustering*, volume 38. M. Dekker New York.

Medin, D. L. and Schaffer, M. M. (1978). Context theory of classification learning. *Psychological review*, 85(3):207.

Medina, C., Devos, A., and Grossglauser, M. (2020). Self-supervised prototypical transfer learning for few-shot classification. *CoRR*, abs/2006.11325.

Mensink, T., Verbeek, J. J., Perronnin, F., and Csurka, G. (2013). Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2624–2637.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations (ICLR)*.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2017). Meta-learning with temporal convolutions. *CoRR*, abs/1707.03141.

Misra, I. and van der Maaten, L. (2020). Self-supervised learning of pretext-invariant representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mozer, M. C. (2009). Attractor networks. *The Oxford companion to consciousness*, pages 86–89.

Muja, M. and Lowe, D. (2009). Flann-fast library for approximate nearest neighbors user manual. *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*.

Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. (2017). Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec@CCS)*.

Murphy, G. (2004). *The big book of concepts*. MIT press.

Natarajan, N., Dhillon, I. S., Ravikumar, P., and Tewari, A. (2013). Learning with noisy labels. In *Advances in Neural Information Processing Systems (NIPS)*.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *6th International Conference on Learning Representations (ICLR)*.

Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *14th European Conference on Computer Vision (ECCV)*.

Noroozi, M., Pirsiavash, H., and Favaro, P. (2017). Representation learning by learning to count. In *IEEE International Conference on Computer Vision (ICCV)*.

Oreshkin, B. N., López, P. R., and Lacoste, A. (2018). TADAM: task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Orhan, A. E., Gupta, V. V., and Lake, B. M. (2020). Self-supervised learning through the eyes of a child. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Oza, P. and Patel, V. M. (2019). C2AE: class conditioned auto-encoder for open-set recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

Parisi, G. I., Tani, J., Weber, C., and Wermter, S. (2018). Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. *Frontiers in neurorobotics*, 12:78–78.

Pathak, D., Girshick, R., Dollár, P., Darrell, T., and Hariharan, B. (2017). Learning features by watching objects move. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pérez-Rúa, J., Zhu, X., Hospedales, T. M., and Xiang, T. (2020). Incremental few-shot object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13843–13852. IEEE.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229.

Pinto, R. C. and Engel, P. M. (2015). A fast incremental gaussian mixture model. *CoRR*, abs/1506.04422.

Pitman, J. and Yor, M. (1997). The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pages 855–900.

Qi, H., Brown, M., and Lowe, D. G. (2018). Low-shot learning with imprinted weights. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. (2020). Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *8th International Conference on Learning Representations (ICLR)*.

Rajasegaran, J., Hayat, M., Khan, S. H., Khan, F. S., and Shao, L. (2019). Random path selection for continual learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS)*.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. (2019). Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Rao, D., Visin, F., Rusu, A. A., Pascanu, R., Teh, Y. W., and Hadsell, R. (2019). Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS)*.

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285.

Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations (ICLR)*.

Rebuffi, S., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Reed, S. E., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. *CoRR*, abs/1412.6596.

Ren, M., Iuzzolino, M. L., Mozer, M. C., and Zemel, R. S. (2021a). Wandering within a world: Online contextualized few-shot learning. In *9th International Conference on Learning Representations (ICLR)*.

Ren, M., Liao, R., Fetaya, E., and Zemel, R. S. (2019). Incremental few-shot learning with attention attractor networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Ren, M., Liao, R., Urtasun, R., Sinz, F. H., and Zemel, R. S. (2017). Normalizing the normalizers: Comparing and extending network normalization schemes. In *5th International Conference on Learning Representations (ICLR)*.

Ren, M., Pokrovsky, A., Yang, B., and Urtasun, R. (2018a). Sbnet: Sparse blocks network for fast inference. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ren, M., Scott, T. R., Iuzzolino, M. L., Mozer, M. C., and Zemel, R. S. (2021b). Online unsupervised learning of visual representations and categories. *CoRR*, abs/2109.05675.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. (2018b). Meta-learning for semi-supervised few-shot classification. In *6th International Conference on Learning Representations (ICLR)*.

Ren, M., Triantafillou, E., Wang, K., Lucas, J., Snell, J., Pitkow, X., Tolias, A. S., and Zemel, R. S. (2020a). Flexible few-shot learning of contextual similarity. *CoRR*, abs/2012.05895.

Ren, M. and Zemel, R. S. (2017). End-to-end instance segmentation with recurrent attention. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018c). Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Ren, Z., Yeh, R. A., and Schwing, A. G. (2020b). Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. (2019). Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, (NeurIPS)*.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. (2019). Learning to learn without forgetting by maximizing transfer and minimizing interference. In *7th International Conference on Learning Representations (ICLR)*.

Rosch, E. and Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7(4):573–605.

Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. (2015). Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2019). Meta-learning with latent embedding optimization. In *7th International Conference on Learning Representations (ICLR)*.

Sadat, A., Casas, S., Ren, M., Wu, X., Dhawan, P., and Urtasun, R. (2020). Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *16th European Conference on Computer Vision*.

Sadat, A., Ren, M., Pokrovsky, A., Lin, Y., Yumer, E., and Urtasun, R. (2019). Jointly learnable behavior and trajectory planning for self-driving vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. P. (2016). Meta-learning with memory-augmented neural networks. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D. (2019). Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision ICCV*.

Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., and Boult, T. E. (2013). Toward open set recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1757–1772.

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook*. Diplomarbeit, Technische Universität München, München.

Schraudolph, N. N. (1999). Local gain adaptation in stochastic gradient descent.

Serrà, J., Suris, D., Miron, M., and Karatzoglou, A. (2018). Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

She, Q., Feng, F., Hao, X., Yang, Q., Lan, C., Lomonaco, V., Shi, X., Wang, Z., Guo, Y., Zhang, Y., Qiao, F., and Chan, R. H. M. (2019). Openloris-object: A dataset and benchmark towards lifelong object recognition. *CoRR*, abs/1911.06487.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*.

Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., and Meng, D. (2019). Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*.

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS)*.

Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C., Cubuk, E. D., Kurakin, A., and Li, C. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Song, M. and Wang, H. (2005). Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In *Intelligent Computing: Theory and Applications III*, volume 5803, pages 174–183. International Society for Optics and Photonics.

Sprechmann, P., Jayakumar, S. M., Rae, J. W., Pritzel, A., Badia, A. P., Uria, B., Vinyals, O., Hassabis, D., Pascanu, R., and Blundell, C. (2018). Memory-based parameter adaptation. In *6th International Conference on Learning Representations (ICLR)*.

Su, J., Maji, S., and Hariharan, B. (2020). When does self-supervision improve few-shot learning? In *16th European Conference on Computer Vision (ECCV)*.

Sukhbaatar, S. and Fergus, R. (2015). Learning from noisy labels with deep neural networks. In *3rd International Conference on Learning Representations (ICLR)*.

Sullivan, J., Mei, M., Perfors, A., Wojcik, E. H., and Frank, M. C. (2020). Saycam: A large, longitudinal audiovisual dataset recorded from the infant's perspective.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., and Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014 (NIPS)*.

Sutton, R. (1992). Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*.

Sykora, Q., Ren, M., and Urtasun, R. (2020). Multi-agent routing value iteration network.

Tanaka, D., Ikami, D., Yamasaki, T., and Aizawa, K. (2018). Joint optimization framework for learning with noisy labels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tao, X., Hong, X., Chang, X., Dong, S., Wei, X., and Gong, Y. (2020). Few-shot class-incremental learning. *CoRR*, abs/2004.10956.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*.

Thrun, S. (1998). Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer.

Tian, Y., Krishnan, D., and Isola, P. (2020). Contrastive multiview coding. In *16th European Conference on Computer Vision (ECCV)*.

Ting, K. M. (2000). A comparative study of cost-sensitive boosting algorithms. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*.

Triantafillou, E., Larochelle, H., Zemel, R. S., and Dumoulin, V. (2021). Learning a universal template for few-shot dataset generalization. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P., and Larochelle, H. (2020). Meta-dataset: A dataset of datasets for learning to learn from few examples. In *8th International Conference on Learning Representations (ICLR)*.

Tu, J., Li, H., Yan, X., Ren, M., Chen, Y., Liang, M., Bitar, E., Yumer, E., and Urtasun, R. (2021a). Exploring adversarial robustness of multi-sensor perception systems in self driving. *CoRR*, abs/2101.06784.

Tu, J., Ren, M., Manivasagam, S., Liang, M., Yang, B., Du, R., Cheng, F., and Urtasun, R. (2020). Physically realizable adversarial examples for lidar object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tu, J., Wang, T., Wang, J., Manivasagam, S., Ren, M., and Urtasun, R. (2021b). Adversarial attacks on multi-agent communication. *CoRR*, abs/2101.06560.

Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

Vadivelu, N., Ren, M., Tu, J., Wang, J., and Urtasun, R. (2020). Learning to communicate and correct pose errors. In *4th Annual Conference on Robot Learning (CoRL)*.

Vahdat, A. (2017). Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

van de Ven, G. M., Siegelmann, H. T., and Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1):4069.

van de Ven, G. M. and Tolias, A. S. (2018). Generative replay with feedback connections as a general strategy for continual learning. *CoRR*, abs/1809.10635.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Kavukcuoglu, K., Vinyals, O., and Graves, A. (2016). Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems NIPS*.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS)*.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57.

Wang, A., Ren, M., and Zemel, R. S. (2021a). Sketchembednet: Learning novel concepts by imitating drawings. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Wang, J., Pun, A., Tu, J., Manivasagam, S., Sadat, A., Casas, S., Ren, M., and Urtasun, R. (2021b). Advsim: Generating safety-critical scenarios for self-driving vehicles. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, J., Ren, M., Bogunovic, I., Xiong, Y., and Urtasun, R. (2021c). Cost-efficient online hyperparameter optimization. *CoRR*, abs/2101.06590.

Wang, X. and Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *IEEE International Conference on Computer Vision (ICCV)*.

Wang, Y., Girshick, R. B., Hebert, M., and Hariharan, B. (2018). Low-shot learning from imaginary data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Y., Kucukelbir, A., and Blei, D. M. (2017). Robust probabilistic modeling with bayesian data reweighting. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Webb, A. R. (2003). *Statistical pattern recognition*. John Wiley & Sons.

Wei, B., Ren, M., Zeng, W., Liang, M., Yang, B., and Urtasun, R. (2021). Perceive, attend, and drive: Learning spatial attention for safe self-driving. In *International Conference on Robotics and Automation (ICRA)*.

Welling, M. (2009). Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Williams, R. J. and Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501.

Wong, K., Wang, S., Ren, M., Liang, M., and Urtasun, R. (2019). Identifying unknown instances for autonomous driving. In *3rd Annual Conference on Robot Learning (CoRL)*.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. (2019). Large scale incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, Y. and He, K. (2018). Group normalization. In *15th European Conference on Computer Vision (ECCV)*.

Wu, Y., Ren, M., Liao, R., and Grosse, R. B. (2018a). Understanding short-horizon bias in stochastic meta-optimization. In *6th International Conference on Learning Representations (ICLR)*.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018b). Unsupervised feature learning via non-parametric instance discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. (2015). Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xie, Q., Dai, Z., Hovy, E. H., Luong, T., and Le, Q. (2020a). Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Xie, Q., Luong, M., Hovy, E. H., and Le, Q. V. (2020b). Self-training with noisy student improves imagenet classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiong, Y., Ren, M., and Urtasun, R. (2019). Learning to remember from a multi-task teacher. *CoRR*, abs/1910.04650.

Xiong, Y., Ren, M., and Urtasun, R. (2020). Loco: Local contrastive representation learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020 (NeurIPS)*.

Xiong, Y., Ren, M., Zeng, W., and Urtasun, R. (2021). Self-supervised representation learning from flow equivariance. *CoRR*, abs/2101.06553.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics (ACL)*, pages 189–196. Association for Computational Linguistics.

Ye, H., Hu, H., and Zhan, D. (2021). Learning adaptive classifiers synthesis for generalized few-shot learning. *Int. J. Comput. Vis.*, 129(6):1930–1953.

Yi, K. and Wu, J. (2019). Probabilistic end-to-end noise correction for learning with noisy labels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yonelinas, A. P., Ranganath, C., Ekstrom, A. D., and Wiltgen, B. J. (2019). A contextual binding theory of episodic memory: systems consolidation reconsidered. *Nature reviews. Neuroscience*, 20(6):364–375.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. In *6th International Conference on Learning Representations (ICLR)*.

Yu, T., Geng, X., Finn, C., and Levine, S. (2020). Variable-shot adaptation for online meta-learning. *CoRR*, abs/2012.07769.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Zemel, R. S. and Mozer, M. C. (2001). Localist attractor networks. *Neural Computation*, 13(5):1045–1064.

Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., and Urtasun, R. (2019). End-to-end interpretable neural motion planner. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Zhan, X., Xie, J., Liu, Z., Ong, Y., and Loy, C. C. (2020). Online deep clustering for unsupervised representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations (ICLR)*.

Zhang, C., Ren, M., and Urtasun, R. (2019). Graph hypernetworks for neural architecture search. In *7th International Conference on Learning Representations (ICLR)*.

Zhang, C., Song, N., Lin, G., Zheng, Y., Pan, P., and Xu, Y. (2021). Few-shot incremental learning with continually evolved classifiers. *CoRR*, abs/2104.03047.

Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations (ICLR)*.

Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. In *14th European Conference on Computer Vision (ECCV)*.

Zhao, Y. and Zhu, S. (2013). Scene parsing by integrating function, geometry and appearance models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

Zhu, X. J. (2005). Semi-supervised learning literature survey.

Zhu, Y., Min, M. R., Kadav, A., and Graf, H. P. (2020). S3VAE: self-supervised sequential VAE for representation disentanglement and data generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. (2019). Fast context adaptation via meta-learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.