Robust Deep Learning in the Open World with Lifelong Learning and Representation Learning

by

Kibok Lee

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2020

Doctoral Committee:

Associate Professor Honglak Lee, Chair
Assistant Professor David Fouhey
Professor Alfred O. Hero, III
Assistant Professor Justin Johnson

Kibok Lee

kibok@umich.edu

ORCID iD: 0000-0001-6995-7327

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor, Professor Honglak Lee, for his guidance and support throughout my graduate studies. It was an honor and privilege to have him as a mentor. All my works presented in this dissertation would not have been possible without his insightful advice. I would also like to thank Professor Jinwoo Shin and Kimin Lee from KAIST for their continuous collaboration with me. More than half of my publications have been written through discussion with them.

I would like to extend my appreciation to my dissertation committee members, Professor David Fouhey, Professor Alfred O. Hero III, and Professor Justin Johnson. They provided valuable feedback on this dissertation.

I would like to thank all my former and current labmates for their valuable discussions and warm friendship: Scott Reed, Yuting Zhang, Wenling Shang, Junhyuk Oh, Ruben Villegas, Xinchen Yan, Ye Liu, Seunghoon Hong, Lajanugen Logeswaran, Rui Zhang, Sungryull Sohn, Jongwook Choi, Yijie Guo, Yunseok Yang, Wilka Carvalho, Anthony Liu, and Thomas Huang.

Many thanks to all my collaborators in academia and industry: Professor Anna Gilbert, Yi Zhang, Kyle Min, and Yian Zhu from UMich, Professor Bo Li from UIUC, Sukmin Yun from KAIST, Woojoo Sim from Samsung, Ersin Yumer, Raquel Urtasun, and Zhuoyuan Chen from Uber ATG, and Kihyuk Sohn and Chun-Liang Li from Google Cloud AI.

I am grateful to have had the opportunity to work on research projects at Uber ATG as a research intern, which allowed me to experience conducting research from an industrial perspective.

I would like to express my deepest gratitude to my family for their endless love and support. My parents and brother have always been a source of comfort and hope in difficult times. Your presence has been a great pleasure and encouragement to me.

Lastly, I gratefully acknowledge the financial support from Kwanjeong Educational Foundation Scholarship, Samsung Electronics, NSF CAREER IIS-1453651, ONR N00014-16-1-2928, and DARPA Explainable AI (XAI) program #313498.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Deep neural networks have shown a superior performance in many learning problems by learning hierarchical latent representations from a large amount of labeled data. However, the success of deep learning methods is under the closed-world assumption: no instances of new classes appear at test time. On the contrary, our world is open and dynamic, such that the closed-world assumption may not hold in many real applications. In other words, deep learning-based agents are not guaranteed to work in the open world, where instances of unknown and unseen classes are pervasive.

In this dissertation, we explore lifelong learning and representation learning to generalize deep learning methods to the open world. Lifelong learning involves identifying novel classes and incrementally learning them without training from scratch, and representation learning involves being robust to data distribution shifts. Specifically, we propose 1) hierarchical novelty detection for detecting and identifying novel classes, 2) continual learning with unlabeled data to overcome catastrophic forgetting when learning the novel classes, 3) network randomization for learning robust representations across visual domain shifts, and 4) domain-agnostic contrastive representation learning, which is robust to data distribution shifts.

The first part of this dissertation studies a cycle of lifelong learning. We divide it into two steps and present how we can achieve each step: first, we propose a new novelty detection and classification framework termed hierarchical novelty detection for detecting and identifying novel classes. Then, we show that unlabeled data easily obtainable in the open world are useful to avoid forgetting about the previously learned classes when learning novel classes. We propose a new knowledge distillation method and confidence-based sampling method to effectively leverage the unlabeled data.

The second part of this dissertation studies robust representation learning: first, we present a network randomization method to learn an invariant representation across visual changes, particularly effective in deep reinforcement learning. Then, we propose a domain-agnostic robust representation learning method by introducing vicinal risk minimization in contrastive representation learning, which consistently improves the quality of representation and transferability across data distribution shifts.

CHAPTER I

Introduction

1.1 Motivation

Deep learning is a class of machine learning algorithms that aims to learn hierarchical latent representations from high-dimensional raw data (Hinton and Salakhutdinov, 2006; LeCun et al., 2015). The power of deep learning lies not only in eliminating the need for hand-crafted features, but also in boosting the performance of machine learning algorithms. Deep learning has been considered a core method to make progress toward artificial intelligence. Specifically, deep learning approaches have been successfully applied in many learning problems, such as visual recognition (Girshick et al., 2014; Simonyan and Zisserman, 2015; He et al., 2016), speech recognition (Hinton et al., 2012; Graves et al., 2013), natural language processing (Mikolov et al., 2013; Pennington et al., 2014), and reinforcement learning (Mnih et al., 2015; Silver et al., 2017).

However, the success of deep learning methods is mostly under the closed-world assumption: training and test data are drawn from the same distribution, such that no instances of unknown or unseen classes appear at test time. On the contrary, many real-world problems do not satisfy this assumption as they lie in the open world. Therefore, the robustness of deep learning has been a serious concern regarding its practical applicability. To generalize deep learning methods to real-world applications, several research topics have been explored, including representation learning (Bengio et al., 2013; Chen et al., 2020a; He et al., 2020), domain adaptation (Ganin and Lempitsky, 2015; Ganin et al., 2016; Tobin et al., 2017), transfer learning (Donahue et al., 2014; Razavian et al., 2014; Yosinski et al., 2014), out-of-distribution detection (Hendrycks and Gimpel, 2016; Lee et al., 2018b,c), adversarial robustness (Szegedy et al., 2014; Goodfellow et al., 2015; Nguyen et al., 2015), lifelong learning (Li and Hoiem, 2016; Kirkpatrick et al., 2017; Parisi et al., 2019), zero-shot learning (Rohrbach et al., 2011; Xian et al., 2017), and few-shot learning (Koch et al.,

2015; Santoro et al., 2016; Vinyals et al., 2016). In this dissertation, we study lifelong learning and representation learning in the open-world setting for successful deployment of deep learning methods to real-world applications.

Lifelong learning aims to continually learn over time by acquiring new knowledge from a continuous stream of data while retaining previously learned knowledge (Thrun and Mitchell, 1995; Silver et al., 2013; Hassabis et al., 2017; Chen and Liu, 2018; Parisi et al., 2019). Humans are innately motivated to continually learn to improve their knowledge and develop their skills throughout their lives. Similarly, in real-world applications, intelligent agents are exposed to ever-changing environments, such that they are required to continually learn from their experiences to operate in new environments in their lifetime. For example, autonomous driving agents should interact with environments, such that they arrive at the destination without accident in any conditions. Lifelong learning enables the agents to deal with unexpected cases occurring in the real world. While most of the lifelong learning works have focused on classification tasks (Li and Hoiem, 2016; Kirkpatrick et al., 2017), lifelong learning has become successful with deep architectures in many different machine learning tasks, including representation learning (Rao et al., 2019), object detection (Liu et al., 2020), natural language processing (Sun et al., 2020), reinforcement learning (Tessler et al., 2017; Schwarz et al., 2018), and robotics (Lesort et al., 2020).

Lifelong learning can be divided into two steps: first, the agents should detect and identify "what to learn." To achieve this, novelty or out-of-distribution detection (Manevitz and Yousef, 2001; Pimentel et al., 2014; Hendrycks and Gimpel, 2016; Lee et al., 2018b,c) can be considered, as it is a task to detect objects never seen before. Novelty detection has several interchangeable terms with subtle differences: anomaly detection, outlier detection, and out-of-distribution detection. Anomaly or outlier detection implicitly assumes that detected novelties should be rejected as they are abnormal (Pimentel et al., 2014). However, novelty or out-of-distribution detection aims to distinguish known and novel objects, which could be from either unknown classes or unknown distribution. In lifelong learning, the goal of detection is to learn them, so novelty or out-of-distribution would be more appropriate. In particular, out-of-distribution (Hendrycks and Gimpel, 2016; Lee et al., 2018b,c) has recently gained increasing attention, as deep learning models are overconfident in their predictions, which are highly biased to in-distribution observed during training. However, novelty detection results are only the novelty of inputs, such that human annotation is required before learning them. In other words, the conventional novelty detection is not directly applicable to lifelong learning.

Our first contribution in Chapter II is proposing a new novelty detection and classifi-

cation framework termed hierarchical novelty detection to provide better identification of the novel objects to the agents. More specifically, in hierarchical novelty detection, novel classes are expected to be classified as semantically the most relevant label, i.e., the closest class in the hierarchical taxonomy built with known classes. For example, if an agent knows dogs, cats, and birds, it infers the general characteristics of dogs and cats, which are categorized as mammals. Then, when the agent observes a bear, hierarchical novelty detection enables the agent to predict its label as novel mammal. In this way, the knowledge of the agent is more organized based on the similarity of objects, such that the agent is able to efficiently learn on top of its simplified representation.

Next, the agents should continually update their knowledge without forgetting previously learned knowledge. Continual lifelong learning has been studied with different assumptions. Broadly speaking, there are three types of problems (van de Ven and Tolias, 2018): first, the goal of task-incremental learning (Li and Hoiem, 2016; Hou et al., 2018) is a kind of multitask learning, but tasks are sequentially given. In this problem, task boundaries are assumed to be clear and the agent receives information about the task at test time. Another is data- or domain-incremental learning (Kirkpatrick et al., 2017; Rusu et al., 2016b; Schwarz et al., 2018), where task identities are not available at test time. Instead, the output space is also fixed, such that the agent learns to generalize previously observed classes. Many robotics and reinforcement learning problems follow this assumption: for example, agents in video games (Bellemare et al., 2013; Cobbe et al., 2019) or 3D simulated environments (Beattie et al., 2016; Fan et al., 2018) have a predefined set of actions, but they can perform multiple tasks with the actions. Lastly, class-incremental learning (Rebuffi et al., 2017; Castro et al., 2018; Wu et al., 2018a; Lee et al., 2019a) does not assume explicit task boundaries, nor is the output space fixed.

In continual learning, deep learning agents easily forget about previously learned knowledge while acquiring new knowledge, which is often referred to as catastrophic forgetting (McCloskey and Cohen, 1989; French, 1999). There are two main streams of continual learning methods to overcome catastrophic forgetting: model-based and data-based. Model-based approaches (Rusu et al., 2016b; Kirkpatrick et al., 2017; Lee et al., 2017; Lopez-Paz et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Chaudhry et al., 2018; Jung et al., 2018; Mallya et al., 2018; Nguyen et al., 2018; Ritter et al., 2018; Schwarz et al., 2018; Serra et al., 2018; Yoon et al., 2018) keep the knowledge of previous tasks by penalizing the change of parameters crucial for previous tasks, such that measuring the importance of parameters on previous tasks is important in these approaches. Data-based approaches (Li and Hoiem, 2016; Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2018; Javed and Shafait, 2018; Wu et al., 2018a; Lee et al., 2019a) keep the knowledge of pre-

vious tasks by knowledge distillation (Hinton et al., 2015), which minimizes the distance between the manifold of the latent space in the previous and new models. In contrast to model-based approaches, they require data to be fed to get features on the latent space. As the similarity between the previously and newly learned tasks is important to prevent catastrophic forgetting, a small amount of memory is reserved to keep a coreset of data from previous tasks (Rebuffi et al., 2017; Castro et al., 2018; Nguyen et al., 2018; Lee et al., 2019a), or a generative model is trained to replay data of previous tasks (Rannen et al., 2017; Shin et al., 2017; Lesort et al., 2018; van de Ven and Tolias, 2018; Wu et al., 2018a) when training a new model.

Our second contribution in Chapter III is motivated by the fact that intelligent agents are exposed to streams of unlabeled data in the real world. To alleviate catastrophic forgetting, we propose to leverage unlabeled data easily obtainable in the open world. However, learning with infinite amounts of unlabeled data is inefficient, and the learning objectives proposed in prior works are not capable of fully leveraging unlabeled data, as they are studied under the closed-world assumption. To this end, we propose a confidence-based sampling method and a new learning objective for continual learning, termed global distillation. With hierarchical novelty detection and continual learning in the open world, intelligent agents grow closer to being successfully applied in real-world applications.

Representation learning is a fundamental problem of machine learning (Lee et al., 2007; Bengio et al., 2013): it aims to extract useful information from data for deployment to other machine learning applications. Indeed, the performance of machine learning methods heavily depends on the data representations. The success of deep learning is from the better quality of deeply learned features (Girshick et al., 2014; Razavian et al., 2014) compared with handcrafted ones, e.g., SIFT (Lowe, 1999) and HOG (Dalal and Triggs, 2005) in computer vision, and MFCC (Davis and Mermelstein, 1980) in speech processing. Deep representation learning has been studied in both supervised and unsupervised ways: a simple but effective way of supervised representation learning is to extract latent features from convolutional neural networks (Krizhevsky et al., 2012; He et al., 2015; Simonyan and Zisserman, 2015; Szegedy et al., 2015; He et al., 2016; Szegedy et al., 2016) trained on large-scale datasets (Deng et al., 2009) in classification tasks. In particular, deep metric learning (Yi et al., 2014; Hoffer and Ailon, 2015; Sohn, 2016; Song et al., 2016) focuses more on learning distance metrics in a supervised way. Unsupervised representation learning aims for learning representations from unlabeled data by solving pretext tasks that are different from downstream tasks. It is often referred to as self-supervised learning, as the learning problem of pretext tasks is based on self-supervision. Among many

self-supervised pretext tasks designed for representation learning, data reconstruction by autoencoding (Bengio et al., 2007; Vincent et al., 2010; Kingma and Welling, 2014), classifying individual instances (Dosovitskiy et al., 2014; Wu et al., 2018b; Chen et al., 2020a; He et al., 2020), and clustering (Xie et al., 2016; Caron et al., 2018, 2019; Ji et al., 2019; Asano et al., 2020; Yan et al., 2020) have been popular in recent literature.

However, deeply learned representations tend to overfit to training data distributions, such that the application of deep learning to the real world is challenging (Szegedy et al., 2014; Goodfellow et al., 2015; Hendrycks and Gimpel, 2016; Lee et al., 2018b). Specifically, domain generalization (Ganin and Lempitsky, 2015; Ganin et al., 2016; Tobin et al., 2017) and transfer learning (Donahue et al., 2014; Razavian et al., 2014; Yosinski et al., 2014) aim to generalize learned representations to different domains and tasks, and out-of-distribution detection (Hendrycks and Gimpel, 2016; Lee et al., 2018b,c) and adversarial robustness (Szegedy et al., 2014; Goodfellow et al., 2015; Nguyen et al., 2015) aim to avoid failures incurred by data from out-of-distribution or adversarially generated. The robustness and generalization ability of deep representation learning are a step towards its real-world application, directly related to the reliability of its operation and its wide applicability in the open world.

Our contributions are to improve the robustness and generalization ability of deep representation learning. First, we consider the problem of domain generalization in deep reinforcement learning. Although deep reinforcement learning has been applied to various applications, it has recently been evidenced that deep reinforcement learning agents often struggle to generalize to new environments, even when the new environments are semantically similar to the previously observed environments (Farebrother et al., 2018; Zhang et al., 2018b; Cobbe et al., 2019; Gamrian and Goldberg, 2019). To overcome this, domain randomization (Tobin et al., 2017) has been proposed to learn robust representations across certain characteristics randomized in a simulator, e.g., colors and textures of objects. However, while domain randomization successfully extrapolates the simulated characteristics to the real world, it requires a simulator, which may not always be available. Our first contribution to robust representation learning in Chapter IV is to propose network randomization, which has the effect of domain randomization without a simulator. When the domain of input observation is an image, by randomizing the parameters of certain layers of the policy network, the agents learn robust representations across visual pattern changes.

Next, we study domain-agnostic self-supervised representation learning. While contrastive learning has recently shown state-of-the-art performances (Chen et al., 2020a,b), it relies heavily on data augmentation techniques, which are carefully designed based on domain knowledge, e.g., in computer vision. This limits the applicability of contrastive

representation learning, as domain-specific knowledge is required to improve the quality of learned representations. Our second contribution to robust representation learning in Chapter V is to propose i-MixUp, which is a domain-agnostic contrastive learning method. We observe that i-MixUp improves the quality of learned representations in various cases, including when data distribution is shifted, when domain knowledge is not enough to design data augmentation techniques, and when the domain is not an image.

In summary, the goal of this dissertation is to improve the robustness of deep learning for a successful deployment of deep learning methods in real-world applications. To achieve this, we propose methods for lifelong learning and robust representation learning by answering the following research questions:

- How can novel classes be detected and identified for lifelong learning?
- How can models update continually without forgetting in the open-world setting?
- How can representations invariant across visual changes be learned?
- How can robust representations be learned in a domain-agnostic way?

1.2 Organization

In the first part, we study a cycle of lifelong learning: namely, a deep learning model first detects and identifies novel classes, and then extends its knowledge by learning to recognize novel classes without forgetting known classes. More specifically, in Chapter II, we propose a new novelty detection and classification framework termed hierarchical novelty detection. The intuition of hierarchical novelty detection comes from an empirical observation that hierarchical semantic relationships and the visual appearance of objects are highly correlated (Deng et al., 2010). Motivated by this, we build a taxonomy with the hypernym-hyponym relationships between known classes extracted from the natural language information (Miller, 1995). In this way, novel classes are expected to be classified as semantically the most relevant label, i.e., the closest class in the taxonomy. Next, in Chapter III, we address the problem of catastrophic forgetting (McCloskey and Cohen, 1989; French, 1999) in continual learning: deep learning models forget about the previously learned classes while learning to recognize new classes. To alleviate this, we propose to leverage unlabeled data, which are easily obtainable in the open world. To effectively leverage unlabeled data, we propose a global distillation learning objective and confidence-based sampling strategy.

In the second part, we study the problem of robust representation learning. More specifically, in Chapter IV, we consider generalization in deep reinforcement learning across visual changes. We observe that reinforcement learning agents fail to perform well in unseen environments even if there is only a small visual change from seen environments (Farebrother et al., 2018; Zhang et al., 2018b; Cobbe et al., 2019; Gamrian and Goldberg, 2019). To overcome this, we introduce random neural networks to generate randomized inputs. By re-initializing the parameters of random neural networks at every iteration, the agents learn robust and generalizable representations across tasks with various unseen visual patterns. Finally, in Chapter V, we focus on contrastive representation learning. While contrastive representation learning has recently shown state-of-the-art performances in many tasks in computer vision (Chen et al., 2020a,b), the majority of the improvement is from data augmentation techniques carefully designed based on the domain knowledge. To generalize the applicability of contrastive representation learning and improve its transferability to other domains, we propose domain-agnostic contrastive representation learning, motivated by the success of vicinal risk minimization in supervised learning (Zhang et al., 2018c). To achieve this, we first cast contrastive learning as learning a non-parametric classifier by assigning a unique virtual class to each data in a batch. Then, we linearly interpolate the inputs and virtual class labels in the data and label spaces, respectively.

1.3 List of Publications

Most of the contributions described in this dissertation have been published at various venues. The following list describes the publications corresponding to each chapter (* indicates equal contributions):

- Chapter II: **Kibok Lee**, Kimin Lee, Kyle Min, Yuting Zhang, Jinwoo Shin, and Honglak Lee. Hierarchical novelty detection for visual object recognition. In *CVPR*, 2018.
- Chapter III: **Kibok Lee**, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *ICCV*, 2019.
- Chapter IV: Kimin Lee*, **Kibok Lee***, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *ICLR*, 2020.
- Chapter V: **Kibok Lee**, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-MixUp: Vicinal risk minimization for contrastive representation learning. Preprint, 2020.

In addition, the following publications are also closely related to the problem of improving the robustness and generalization ability of deep learning in the open world:

- Yuting Zhang, **Kibok Lee**, and Honglak Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *ICML*, 2016.
- Anna C Gilbert, Yi Zhang, **Kibok Lee**, Yuting Zhang, and Honglak Lee. Towards understanding the invertibility of convolutional neural networks. In *IJCAI*, 2017.
- Kimin Lee, Honglak Lee, **Kibok Lee**, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *ICLR*, 2018.
- Kimin Lee, **Kibok Lee**, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018.
- Kimin Lee, Sukmin Yun, **Kibok Lee**, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. In *ICML*, 2019.
- **Kibok Lee**, Zhuoyuan Chen, Xinchen Yan, Raquel Urtasun, and Ersin Yumer. ShapeAdv: Generating shape-aware adversarial 3d point clouds. *arXiv preprint arXiv:2005.11626*, 2020.

CHAPTER II

Hierarchical Novelty Detection for Visual Object Recognition

Deep neural networks have achieved impressive success in large-scale visual object recognition tasks with a predefined set of classes. However, recognizing objects of novel classes unseen during training still remains challenging. The problem of detecting such novel classes has been addressed in the literature, but most prior works have focused on providing simple binary or regressive decisions, e.g., the output would be "known," "novel," or corresponding confidence intervals. In this chapter, we study more informative novelty detection schemes based on a hierarchical classification framework. For an object of a novel class, we aim for finding its closest super class in the hierarchical taxonomy of known classes. To this end, we propose two different approaches termed top-down and flatten methods, and their combination as well. The essential ingredients of our methods are confidence-calibrated classifiers, data relabeling, and the leave-one-out strategy for modeling novel classes under the hierarchical taxonomy. Furthermore, our method can generate a hierarchical embedding that leads to improved generalized zero-shot learning performance in combination with other commonly-used semantic embeddings.

2.1 Introduction

Object recognition in large-scale image datasets has achieved impressive performance with deep convolutional neural networks (CNNs) (He et al., 2015, 2016; Simonyan and Zisserman, 2015; Szegedy et al., 2015). The standard CNN architectures are learned to recognize a predefined set of classes seen during training. However, in practice, a new type of objects could emerge (e.g., a new kind of consumer product). Hence, it is desirable to extend the CNN architectures for detecting the novelty of an object (i.e., deciding if the object does

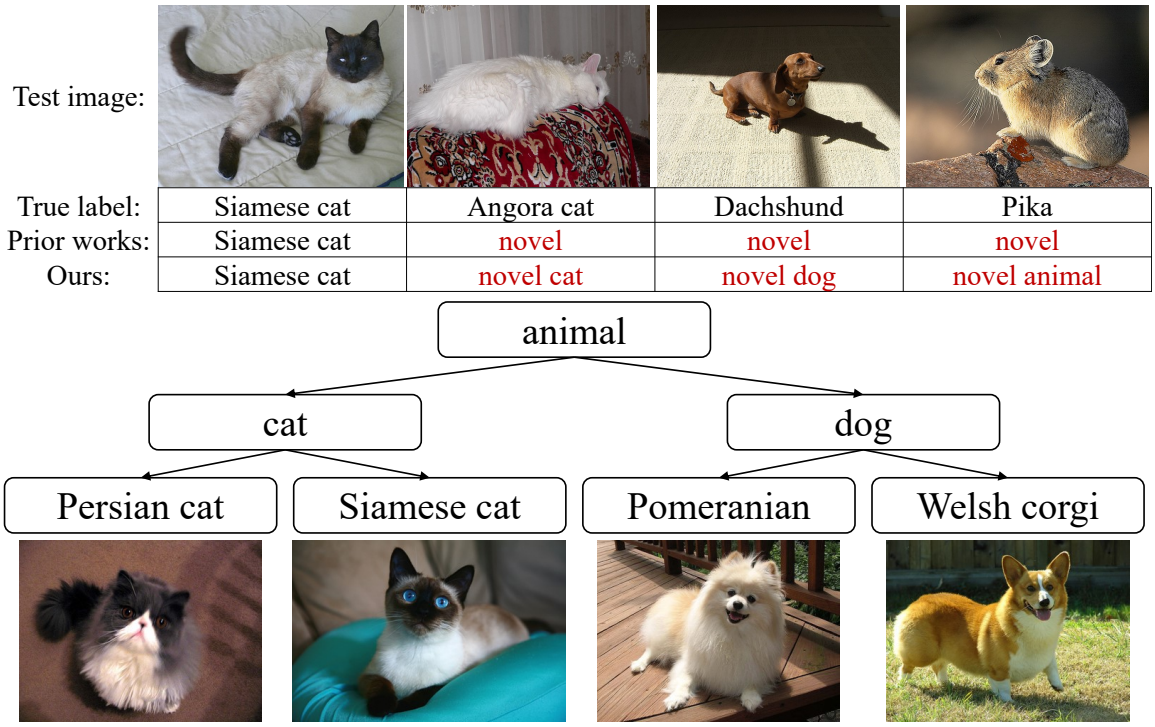


Figure 2.1: An illustration of our proposed hierarchical novelty detection task. In contrast to prior novelty detection works, we aim to find the most specific class label of a novel data on the taxonomy built with known classes.

not match any previously trained object classes). There have been recent efforts toward developing efficient novelty detection methods (Pimentel et al., 2014; Bendale and Boult, 2016; Hendrycks and Gimpel, 2016; Lakshminarayanan et al., 2017; Li and Gal, 2017), but most of the existing methods measure only the model uncertainty, i.e., confidence score, which is often too ambiguous for practical use. For example, suppose one trains a classifier on an animal image dataset as in Figure 2.1. A standard novelty detection method can be applied to a cat-like image to evaluate its novelty, but such a method would not tell whether the novel object is a new species of cat unseen in the training set or a new animal species.

To address this issue, we design a new classification framework for more informative novelty detection by utilizing a hierarchical taxonomy, where the taxonomy can be extracted from the natural language information, e.g., WordNet hierarchy (Miller, 1995). Our approach is also motivated by a strong empirical correlation between hierarchical semantic relationships and the visual appearance of objects (Deng et al., 2010). Under our scheme, a taxonomy is built with the hypernym-hyponym relationships between known classes such that objects from novel classes are expected to be classified into the most relevant label, i.e., the closest class in the taxonomy. For example, as illustrated in Figure 2.1, our goal is to distinguish "new cat," "new dog," and "new animal," which cannot be achieved in the

standard novelty detection tasks. We call this problem *hierarchical novelty detection* task.

In contrast to standard object recognition tasks with a *closed set* of classes, our proposed framework can be useful for extending the domain of classes to an *open set* with taxonomy information (i.e., dealing with any objects unseen in training). In practical application scenarios, our framework can be potentially useful for automatically or interactively organizing a customized taxonomy (e.g., company's product catalog, wildlife monitoring, personal photo library) by suggesting closest categories for an image from novel categories (e.g., new consumer products, unregistered animal species, untagged scenes or places).

We propose two different approaches for hierarchical novelty detection: top-down and flatten methods. In the top-down method, each super class has a confidence-calibrated classifier which detects a novel class if the posterior categorical distribution is close to a uniform distribution. Such a classifier was recently studied for a standard novelty detection task (Lee et al., 2018b), and we extend it for detecting novel classes under our hierarchical novelty detection framework. On the other hand, the flatten method computes a softmax probability distribution of all disjoint classes. Then, it predicts the most likely fine-grained label, either a known class or a novel class. Although the flatten method simplifies the full hierarchical structure, it outperforms the top-down method for datasets of a large hierarchical depth.

Furthermore, we combine two methods for utilizing their complementary benefits: top-down methods naturally leverage the hierarchical structure information, but the classification performance might be degraded due to the error aggregation. On the contrary, flatten methods have a single classification rule that avoids the error aggregation, but the classifier's flat structure does not utilize the full information of hierarchical taxonomy. We empirically show that combining the top-down and flatten models further improves hierarchical novelty detection performance.

Our method can also be useful for generalized zero-shot learning (GZSL) (Chao et al., 2016; Xian et al., 2017) tasks. GZSL is a classification task with classes both seen and unseen during training, given that semantic side information for all test classes is provided. We show that our method can generate a hierarchical embedding that leads to improved GZSL performance in combination with other commonly used semantic embeddings.

2.2 Related Work

Novelty detection. For robust prediction, it is desirable to detect a test sample if it looks unusual or significantly differs from the representative training data. Novelty detection is a task recognizing such abnormality of data (see Hodge and Austin (2004) and Pimentel

et al. (2014) for a survey). Recent novelty detection approaches leverage the output of deep neural network classification models. A confidence score about novelty can be measured by taking the maximum predicted probability (Hendrycks and Gimpel, 2016), ensembling such outputs from multiple models (Lakshminarayanan et al., 2017), or synthesizing a score based on the predicted categorical distribution (Bendale and Boult, 2016). There have also been recent efforts toward confidence-calibrated novelty detection, i.e., calibrating how much the model is certain with its novelty detection, by postprocessing (Liang et al., 2018) or learning with joint objective (Lee et al., 2018b).

Object recognition with taxonomy. Incorporating the hierarchical taxonomy for object classification has been investigated in the literature, either to improve classification performance (Deng et al., 2014; Yan et al., 2015), or to extend the classification tasks to obtain more informative results (Deng et al., 2012; Zhao et al., 2017). Specifically for the latter purpose, Deng et al. (2012) gave some reward to super class labels in a taxonomy and maximized the expected reward. Zhao et al. (2017) proposed an open set scene parsing framework, where the hierarchy of labels is used to estimate the similarity between the predicted label and the ground truth. In contemporary work, Simeone et al. (2017) proposed a hierarchical classification and novelty detection task for the music genre classification, but their settings are different from ours: in their task, novel classes do not belong to any node in the taxonomy. Thus, their method cannot distinguish the difference between novel classes similar to some known classes. To the best of our knowledge, our work is the first to propose a unified framework for hierarchical novelty detection and visual object recognition.

Generalized zero-shot learning (GZSL). We remark that GZSL (Chao et al., 2016; Xian et al., 2017) can be thought as addressing a similar task as ours. While the standard ZSL tasks test classes unseen during training only, GZSL tasks test both seen and unseen classes such that the novelty is automatically detected if the predicted label is not a seen class. However, ZSL and GZSL tasks are valid only under the assumption that specific semantic information of all test classes is given, e.g., attributes (Rohrbach et al., 2013; Lampert et al., 2014; Akata et al., 2015) or text description (Frome et al., 2013; Socher et al., 2013; Norouzi et al., 2014; Changpinyo et al., 2016; Fu and Sigal, 2016; Reed et al., 2016) of the objects. Therefore, GZSL cannot recognize a novel class if prior knowledge about the specific novel class is not provided, i.e., it is limited to classifying objects with prior knowledge, regardless of their novelty. Compared to GZSL, the advantages of the proposed hierarchical novelty detection are that 1) it does not require any prior knowledge on novel classes but only utilizes the taxonomy of known classes, 2) a reliable super class label

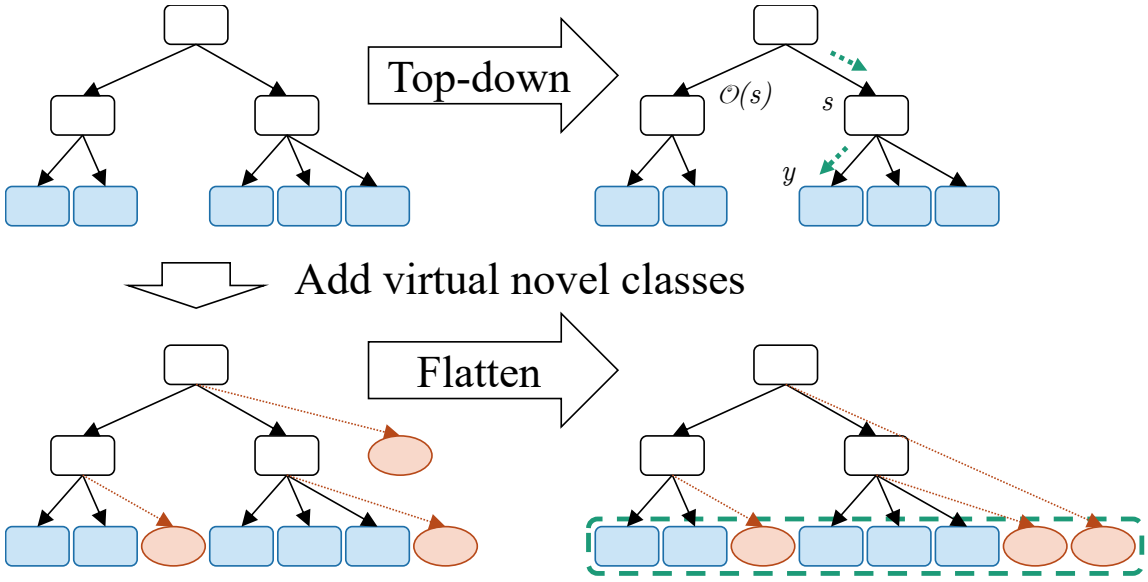


Figure 2.2: Illustration of two proposed approaches. In the top-down method, classification starts from the root class, and propagates to one of its children until the prediction arrives at a known leaf class (blue) or stops if the prediction is not confident, which means that the prediction is a novel class whose closest super class is the predicted class. In the flatten method, we add a virtual novel class (red) under each super class as a representative of all novel classes, and then flatten the structure for classification.

can be more useful and human-friendly than an error-prone prediction over excessively subdivided classes, and 3) high-quality taxonomies are available off-the-shelf and they are better interpretable than latent semantic embeddings. In Section 2.5, we also show that our models for hierarchical novelty detection can also generate a hierarchical embedding such that combination with other semantic embeddings improves the GZSL performance.

2.3 Approach

In this section, we define terminologies to describe hierarchical taxonomy and then propose models for our proposed hierarchical novelty detection task.

2.3.1 Taxonomy

A taxonomy represents a hierarchical relationship between classes, where each node in the taxonomy corresponds to a class or a set of indistinguishable classes.[1] We define three types of classes as follows: 1) *known leaf classes* are nodes with no child, which are known

[1]For example, if a class has only one known child class, these two classes are indistinguishable as they are trained with exactly the same data.

and seen during training, 2) *super classes* are ancestors of the leaf classes, which are also known, and 3) *novel classes* are unseen during training, so they do not explicitly appear in the taxonomy.[2] We note that all known leaf and novel classes have no child and are disjoint, i.e., they are neither ancestor nor descendant of each other. In the example in Figure 2.1, four species of cats and dogs are leaf classes, "cat," "dog," and "animal" are super classes, and any other classes unseen during training, e.g., "Angora cat," "Dachshund," and "Pika" are novel classes.

In the proposed hierarchical novelty detection framework, we first build a taxonomy with known leaf classes and their super classes. At test time, we aim to predict the most fine-grained label in the taxonomy. For instance, if an image is predicted as novel, we try to assign one of the super classes, implying that the input is in a novel class whose closest known class in the taxonomy is that super class.

To represent the hierarchical relationship, let $\mathcal{T}$ be the taxonomy of known classes, and for a class y, $\mathcal{P}(y)$ be the set of parents, $\mathcal{C}(y)$ be the set of children, $\mathcal{A}(y)$ be the set of ancestors including itself, and $\mathcal{N}(y)$ be the set of novel classes whose closest known class is y. Let $\mathcal{L}(\mathcal{T})$ be the set of all descendant known leaves under $\mathcal{T}$, such that $\mathcal{T} \backslash \mathcal{L}(\mathcal{T})$ is the set of all super classes in $\mathcal{T}$.

As no prior knowledge of $\mathcal{N}(y)$ is provided during training and testing, all classes in $\mathcal{N}(y)$ are indistinguishable in our hierarchical novelty detection framework. Thus, we treat $\mathcal{N}(y)$ as a single class in our analysis.

2.3.2 Top-Down Method

A natural way to perform classification using a hierarchical taxonomy is following *top-down* classification decisions starting from the root class, as shown in the top of Figure 2.2. Let $\mathcal{D}(s)$ be a subset of the training dataset $\mathcal{D}$ under a super class s, such that $(x, y) \in \mathcal{D}(s)$ is a pair of an image and its label sampled from $\mathcal{D}(s)$ satisfying $y \in \mathcal{C}(s) \cup \mathcal{N}(s)$. Then, the classification rule is defined as

$$\hat{y} = \begin{cases} \arg\max_{y'} p(y'|x, s; \theta_s) & \text{if confident,} \\ \mathcal{N}(s) & \text{otherwise,} \end{cases}$$

where θ_s is the model parameters of $\mathcal{C}(s)$ and $p(\cdot|x, s; \theta_s)$ is the posterior categorical distribution given an image x at a super class s. The top-down classification stops at s if the

[2] We note that "novel" in our task is similar but different from "unseen" commonly referred in ZSL works; while class-specific semantic information for unseen classes must be provided in ZSL, such information for novel classes is not required in our task.

prediction is a known leaf class or the classifier is not confident with the prediction (i.e., the predicted class is in $\mathcal{N}(s)$). We measure the prediction confidence using the KL divergence with respect to the uniform distribution: intuitively, a confidence-calibrated classifier generates near-uniform posterior probability vector if the classifier is not confident about its prediction. Hence, we interpret that the prediction is confident at a super class s if

$$\begin{aligned}\mathrm{KL}(\mathcal{U}(\cdot|s) \parallel p(\cdot|x,s;\theta_s)) &= \frac{1}{|\mathcal{D}(s)||\mathcal{C}(s)|}\sum_{x\in\mathcal{D}(s)}\sum_{y\in\mathcal{C}(s)}[-\log p(y|x,s;\theta_s)] - \log|\mathcal{C}(s)| \\ &\geq \lambda_s,\end{aligned}$$

where λ_s is a threshold, KL denotes the KL divergence, and $\mathcal{U}(\cdot|s)$ is the uniform distribution when the classification is made under a super class s, $|\mathcal{D}(s)|$ is the number of training data under s, and $|\mathcal{C}(s)|$ is the number of children of s. To train such confidence-calibrated classifiers, we leverage classes disjoint from the class s. Let $\mathcal{O}(s)$ be such a set of all known classes except for s and its descendents. Then, the objective function of our top-down classification model at a super class s is

$$\begin{aligned}\mathcal{L}_{\mathrm{TD}}(\theta_s;\mathcal{D}) =& \frac{1}{|\mathcal{D}(s)|}\sum_{(x,y)\in\mathcal{D}(s)}[-\log p(y|x,s;\theta_s)] \\ &+ \frac{1}{|\mathcal{D}(\mathcal{O}(s))||\mathcal{C}(s)|}\sum_{x\in\mathcal{D}(\mathcal{O}(s))}\sum_{y\in\mathcal{C}(s)}[-\log p(y|x,s;\theta_s)]\,, \qquad (2.1)\end{aligned}$$

where $\mathcal{D}(\mathcal{O}(s))$ denotes a subset of training data under $\mathcal{O}(s)$.

However, under the above top-down scheme, the classification error might aggregate as the hierarchy goes deeper. For example, if one of the classifiers has poor performance, then the overall classification performance of all descendent classes should be low. In addition, the taxonomy is not necessarily a tree but a directed-acyclic graph (DAG), i.e., a class could belong to multiple parents, which could lead to incorrect classification.[3] In the next section, we propose flatten approaches, which overcome the error aggregation issue. Nevertheless, the top-down method can be used for extracting good visual features for boosting the performance of the flatten method, as we show in Section 2.4.

[3]For example, if there are multiple paths to a class in a taxonomy, then the class may belong to (i.e., be a descendant of) multiple children at some super class s, which may lead to low KL divergence from the uniform distribution and the image could be incorrectly classified as $\mathcal{N}(s)$.

2.3.3 Flatten Method

We now propose to represent all probabilities of known leaf and novel classes in a single probability vector, i.e., we *flatten* the hierarchy, as described on the bottom of Figure 2.2. The key idea is that a probability of a super class s can be represented as $p(s|x) = \sum_{y \in \mathcal{C}(s)} p(y|x) + p(\mathcal{N}(s)|x)$, such that from the root node, we have $\sum_{l \in \mathcal{L}(\mathcal{T})} p(l|x) + \sum_{s \in \mathcal{T} \setminus \mathcal{L}(\mathcal{T})} p(\mathcal{N}(s)|x) = 1$, where l and s are summed over all known leaf classes and super classes, respectively. Note that $\mathcal{N}(s)$ is considered as a single novel class under the super class s, as discussed in Section 2.3.1. Thus, as described in Figure 2.2, one can virtually add an extra child for each super class to denote all novel classes under it. Let (x, y) be a pair of an image and its most fine-grained label. Then, the classification rule is

$$\hat{y} = \arg\max_{y'} p(y'|x; \theta),$$

where y' is either a known leaf or novel class. Here, a problem is that we have no training data from novel classes. To address this, we propose two approaches to model the score (i.e., posterior probability) of novel classes.

Data relabeling. A naive strategy is to relabel some training samples to its ancestors in hierarchy. Then, the images relabeled to a super class are considered as novel class images under the super class. This can be viewed as a supervised learning with both fine-grained and coarse-grained classes where they are considered to be disjoint, and one can optimize an objective function of a simple cross entropy function over all known leaf classes and novel classes:

$$\mathcal{L}_{\text{Relabel}}(\theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \left[-\log p(y|x; \theta_{\mathcal{T}}) \right]. \tag{2.2}$$

In our experiments, each training image is randomly relabeled recursively in a bottom-up manner with a probability of r, where $0 < r < 1$ is termed a relabeling rate. An example of relabeling is illustrated in Figure 2.3 (b).

Leave-one-out strategy. A more sophisticated way to model novel classes is to temporarily remove a portion of taxonomy during training: specifically, for a training label y, we recursively remove one of its ancestor $a \in \mathcal{A}(y)$ from the taxonomy $\mathcal{T}$ in a hierarchical manner. To represent a deficient taxonomy, we define $\mathcal{T} \setminus a$ as a taxonomy where a and its descendants are removed from the original taxonomy $\mathcal{T}$. At each stage of removal,

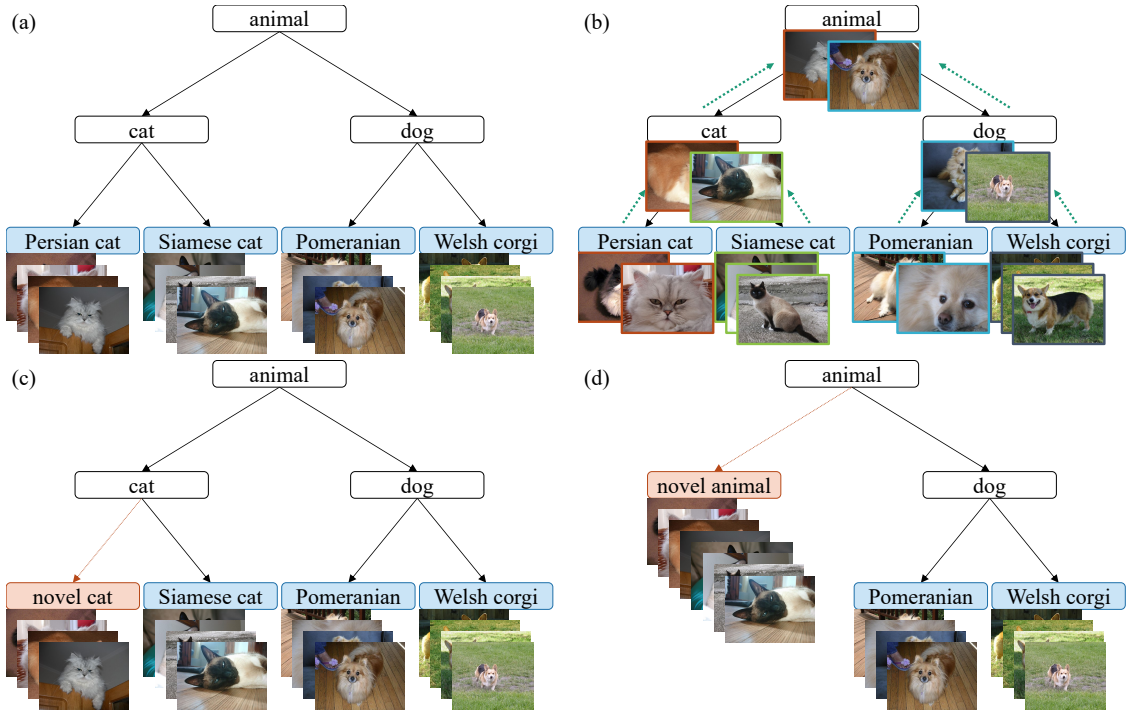


Figure 2.3: Illustration of strategies to train novel class scores in flatten methods. (a) shows the training images in the taxonomy. (b) shows relabeling strategy. Some training images are relabeled to super classes in a bottom-up manner. (c–d) shows leave-one-out (LOO) strategy. To learn a novel class score under a super class, one of its children is temporarily removed such that its descendant known leaf classes are treated as novel during training.

the training label y becomes a novel class of the parent of a in $\mathcal{T}\backslash a$, i.e., $\mathcal{N}(\mathcal{P}(a))$. Figure 2.3 (a, c–d) illustrates this idea with an example: in Figure 2.3 (a), when y is "Persian cat," the set of its ancestor is $\mathcal{A}(y) =\{$ "Persian cat," "cat," "animal" $\}$. In Figure 2.3 (c), images under $a =$"Persian cat" belong to $\mathcal{N}(\mathcal{P}(a)) =$"novel cat" in $\mathcal{T}\backslash a$. Similarly, in Figure 2.3 (d), images under $a =$"cat" belong to $\mathcal{N}(\mathcal{P}(a)) =$"novel animal" in $\mathcal{T}\backslash a$. As we leave a class out to learn a novel class, we call this *leave-one-out* (LOO) method. With some notation abuse for simplicity, the objective function of the LOO model is then

$$\mathcal{L}_{\text{LOO}}(\theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \left[-\log p(y|x; \theta_{\mathcal{L}(\mathcal{T})}) + \sum_{a\in\mathcal{A}(y)} -\log p(\mathcal{N}(\mathcal{P}(a))|x; \theta_{\mathcal{T}\backslash a}) \right], \tag{2.3}$$

where the first term is the standard cross entropy loss with the known leaf classes, and the second term is the summation of losses with $\mathcal{N}(\mathcal{P}(a))$ and the leaves under $\mathcal{T}\backslash a$. We provide further implementation details in Appendix A.1.1.

As we mentioned earlier, the flatten methods can be combined with the top-down

method in sequence: the top-down method first extracts multiple softmax probability vectors from visual features, and then the concatenation of all probabilities can be used as an input of the LOO model. We name the combined method TD+LOO for conciseness.

2.4 Evaluation: Hierarchical Novelty Detection

We present the hierarchical novelty detection performance of our proposed methods combined with CNNs on ImageNet (Deng et al., 2009), Animals with Attributes 2 (AwA2) (Lampert et al., 2014; Xian et al., 2017), and Caltech-UCSD Birds (CUB) (Welinder et al., 2010), where they represent visual object datasets with deep, coarse-grained, and fine-grained taxonomy, respectively. Experimental results on CIFAR-100 (Krizhevsky and Hinton, 2009) can be found in Appendix A.1.3, where the overall trends of results are similar to others.

2.4.1 Evaluation Setups

Compared algorithms. As a baseline, we modify the dual accuracy reward trade-off search (DARTS) algorithm (Deng et al., 2012) for our purpose. Note that DARTS gives some rewards to labels in hierarchy, where fine-grained prediction gets higher reward. Under this algorithm, for a novel class, its closest super class in the taxonomy would give the maximum reward. At test time, the modified DARTS generates expected rewards for all known leaf and novel classes, so prediction can be done in the same way as the flatten methods.

As our proposed methods, Relabel, LOO, and TD+LOO are compared. For a fair comparison in terms of the model capacity, deep Relabel and LOO models are also experimented, where a deep model is a stack of fully connected layers followed by rectified linear units (ReLU). We do not report the performance of the pure top-down method since 1) one can combine it with the LOO method for better performance as mentioned in Section 2.3.2, and 2) fair comparisons between the pure top-down method and others are not easy. Intuitively, the confidence threshold λ_s in Section 2.3.2 can be tuned: for example, the novel class score bias in the flatten method would improve the novel class detection accuracy, but large λ_s does not guarantee the best novel class performance in the top-down method because hierarchical classification results would tend to stop at the root class.

Datasets. ImageNet (Deng et al., 2009) consists of 22k object classes where the taxonomy of the classes is built with the hypernym-hyponym relationships in WordNet (Miller, 1995). We take 1k mutually exclusive classes in ILSVRC 2012 as known leaf classes, which are

a subset of the ImageNet.[4] Based on the hypernym-hyponym relationships in WordNet, we initially obtained 860 super classes of 1k known leaf classes, and then merged indistinguishable super classes. Specifically, if a super class has only one child or shares exactly the same descendant leaf classes, it is merged with classes connected to the class. After merging, the resultant taxonomy is a DAG and has 396 super classes where all super classes have at least two children and have different set of descendant leaf classes. On the other hand, the rest of 21k classes can be used as novel classes for testing. Among them, we discarded super classes, classes under 1k known leaf classes, and classes with less than 50 images for reliable performance measure. After filtering classes, we obtain about 16k novel classes. ILSVRC 2012 has about 1.3M training images and another 50k images in 1k known leaf classes. We put the 50k images aside from training and used for test, and we sampled another 50k images from 1.3M training images for validation. For novel classes, we sampled 50 images from each class. In summary, we have about 1.2M training images, 50k validation images, and 50k test images from known leaf classes, and 800k test images from novel classes.

AwA2 (Lampert et al., 2014; Xian et al., 2017) consists of 40 known leaf classes and 10 novel classes with 37k images, and CUB (Welinder et al., 2010) consists of 150 known leaf classes and 50 novel classes with 12k images. Similar to ImageNet, the taxonomy of each dataset is built with the hypernym-hyponym relationships in WordNet. The resultant taxonomy is a tree and has 21 and 43 super classes for AwA2 and CUB, respectively.

Training. We take ResNet-101 (He et al., 2016) as a visual feature extractor (i.e., the penultimate layer of the CNNs before the classification layer) for all compared methods. The CNNs are pretrained with ILSVRC 2012 1k classes, where they do not contain any novel classes of datasets experimented. Then, the final classification layer of the CNNs is replaced with our proposed models. Note that CNNs and our proposed models can be trained in an end-to-end manner, but we take and freeze the pretrained parameters in all layers except for the final layer for the sake of faster training.

For ImageNet, we use mini-batch SGD with 5k center-cropped data per batch. As a regularization, L2 norm weight decay with parameter 10^{-2} is applied. The initial learning rate is 10^{-2} and it decays at most two times when loss improvement is less than 2% compared to the last epoch. For AwA2 and CUB, the experiments are done in the same environment with the above except that the models are trained with the full-batch GD and Adam optimizer (Kingma and Ba, 2014).

[4] Except "teddy bear," all classes in ILSVRC 2012 are in ImageNet.

Table 2.1: Hierarchical novelty detection results on ImageNet, AwA2, and CUB. For a fair comparison, 50% of known class accuracy is guaranteed by adding a bias to all novel class scores (logits). The AUC is obtained by varying the bias. Known-novel class accuracy curve is shown in Figure 2.5. Values in bold indicate the best performance.

Method	ImageNet		AwA2		CUB	
	Novel	AUC	Novel	AUC	Novel	AUC
DARTS	10.89	8.83	36.75	35.14	40.42	30.07
Relabel	15.29	11.51	45.71	40.28	38.23	28.75
LOO	15.72	12.00	50.00	43.63	40.78	31.92
TD+LOO	**18.78**	**13.98**	**53.57**	**46.77**	**43.29**	**33.16**

Metrics. We first consider the top-1 accuracy by counting the number of predicted labels exactly matching the ground truth. Note that we have two types of classes in test datasets, i.e., known and novel classes. Performances on two types of classes are in trade-off relation, i.e., if one tunes model parameters for favoring novel classes, the accuracy of known classes would be decreased. Specifically, by adding some positive bias to the novel class scores (e.g., logits in the softmax), one can increase novel class accuracy while decreasing known class accuracy, or vice versa. Hence, for a fair comparison, we measure the novel class accuracy with respect to some fixed known class accuracy, e.g., 50%. As a more informative evaluation metric, we also measure the area under known-novel class accuracy curve (AUC). Varying the novel class score bias, a curve of known class accuracy versus novel class accuracy can be drawn, which depicts the relationship between the known class accuracy and the novel class accuracy. The AUC is the area under this curve, which is independent of the novel class score bias.

2.4.2 Experimental Results

We first compare the hierarchical novelty detection results of the baseline method and our proposed methods qualitatively with test images on ImageNet in Figure 2.4. We remark that our proposed methods can provide informative prediction results by utilizing the taxonomy of the dataset. In Figure 2.4 (a), LOO and TD+LOO find the ground truth label (the most fine-grained label in the taxonomy), while DARTS classifies it as "beagle," which is in fact visually similar to "American foxhound." In Figure 2.4 (b), none of the methods find the ground truth, but the prediction of TD+LOO is the most informative, as it is the closest label in the hierarchy. In Figure 2.4 (c–d), only the prediction of TD+LOO is correct, but the rest of the methods also give a reasonable amount of information. More qualitative results can be found in Appendix A.2 and A.3.

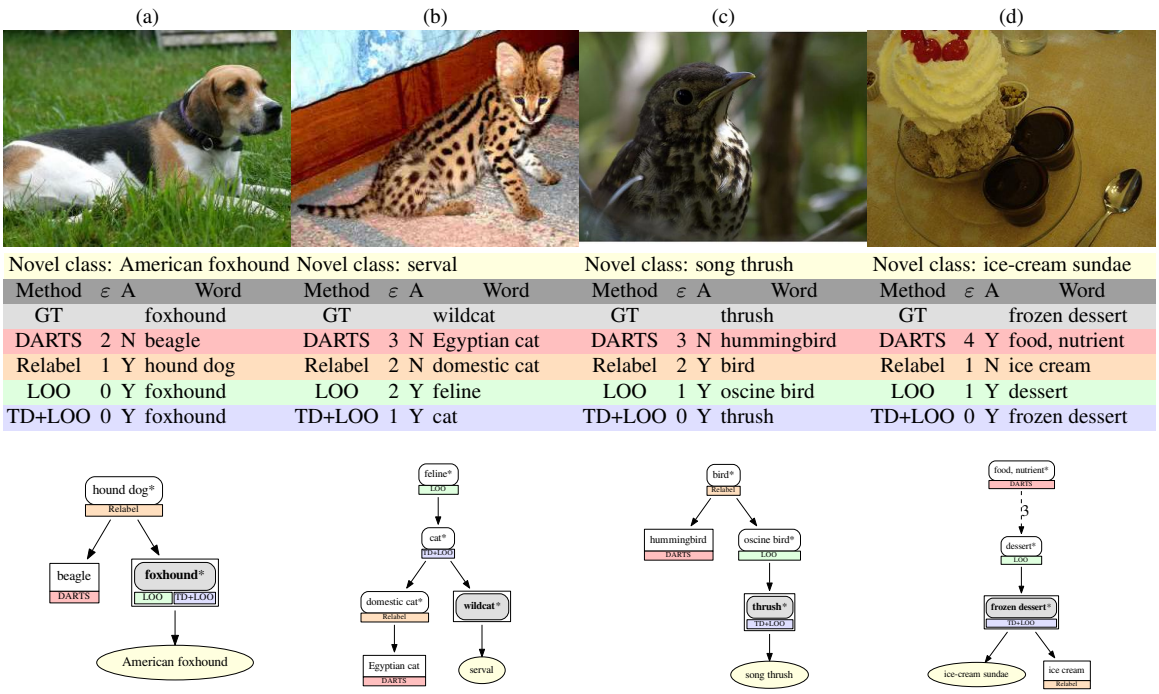
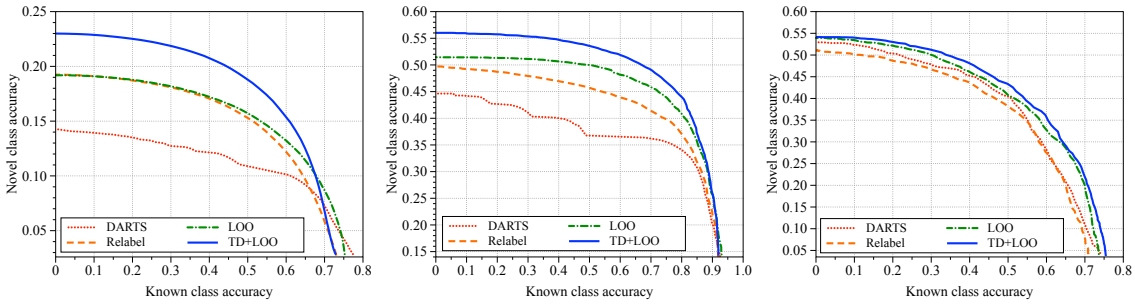


Figure 2.4: Qualitative results of hierarchical novelty detection on ImageNet. "GT" is the closest known ancestor of the novel class, which is the expected prediction, "DARTS" is the baseline method proposed in Deng et al. (2012) where we modify the method for our purpose, and the others are our proposed methods. "ε" is the distance between the prediction and GT, "A" indicates whether the prediction is an ancestor of GT, and "Word" is the English word of the predicted label. Dashed edges represent multi-hop connection, where the number indicates the number of edges between classes. If the prediction is on a super class (marked with * and rounded), then the test image is classified as a novel class whose closest class in the taxonomy is the super class.

Table 2.1 shows the hierarchical novelty detection performance on ImageNet, AwA2, and CUB. One can note that the proposed methods significantly outperform the baseline method in most cases, except the case of Relabel on CUB, because validation could not find the best relabeling rate for test. Also, we remark that LOO outperforms Relabel. The main difference between two methods is that Relabel gives a penalty to the original label if it is relabeled during training, which turns out to be harmful for the performance. Finally, TD+LOO exhibits the best performance, which implies that the multiple softmax probability vectors extracted from the top-down method are more useful than the vanilla visual features extracted from the state-of-the-art CNNs in the hierarchical novelty detection tasks. Figure 2.5 shows the known-novel class accuracy curves by varying the bias added to the novel class scores. Our proposed methods have higher novel class accuracy than the baseline in most regions.



(a) ImageNet (b) AwA2 (c) CUB

Figure 2.5: Known-novel class accuracy curves obtained by varying the novel class score bias on ImageNet, AwA2, and CUB. In most regions, our proposed methods outperform the baseline method.

2.5 Evaluation: Generalized Zero-Shot Learning

We present the GZSL performance of the combination of the hierarchical embedding obtained by the top-down method and other semantic embeddings on Animals with Attributes (AwA1 and AwA2)[5] (Lampert et al., 2014; Xian et al., 2017) and Caltech-UCSD Birds (CUB) (Welinder et al., 2010). In addition, experimental results on AwA1 can be found in Appendix A.4.2, where the overall trends of results are similar.

2.5.1 Evaluation Setups

Hierarchical embeddings for GZSL. GZSL requires an output semantic embedding built with side information, e.g., attributes labeled by human, or word embedding trained with a large text corpus. In addition to those two commonly used semantic embeddings, Akata et al. (2015) proposed to use hierarchical relationships between all classes, including classes unseen during training. Specifically, they measured the shortest path distance between classes in the taxonomy built with both known and novel classes, and took the vector of distance values as an output embedding. We refer to this embedding as Path.

On the other hand, motivated by the effectiveness of the features extracted from the top-down method shown in Section 2.4.2, we set the enumeration of the ideal multiple softmax probability vectors as the semantic embedding: let $\mathcal{C}(s)[i]$ be the i-th child of a super class s. Then, for a label y and a super class s, the i-th element of an ideal output probability

[5] AwA1 is similar to AwA2, but images in AwA1 are no longer available due to the public copyright license issue. We used precomputed CNN features for AwA1, which is available at http://datasets.d2.mpi-inf.mpg.de/xian/xlsa17.zip.

vector $t^{(y,s)} \in [0,1]^{|\mathcal{C}(s)|}$ is

$$t^{(y,s)}[i] = \begin{cases} 1 & \text{if } y \text{ belongs to } \mathcal{C}(s)[i], \\ 0 & \text{if } y \text{ belongs to } \mathcal{C}(s)[j] \text{ where } i \neq j, \\ \frac{1}{|\mathcal{C}(s)|} & \text{if } y \text{ is novel or does not belong to } s, \end{cases}$$

where $|\mathcal{C}(s)|$ is the number of known child classes under s. The resultant output embedding is the concatenation of them with respect to the super classes, i.e., the ground truth semantic vector of a class y is $t^y = [\dots, t^{(y,s)}, \dots]$, and we call this embedding TD. See Appendix A.4.1 for an example of the ideal output probability vector t^y.

Since classes sharing the same closest super class have exactly the same desired output probability vector, we made random guess for fine-grained classification in the experiments only with TD.

Datasets. AwA1 and AwA2 (Lampert et al., 2014; Xian et al., 2017) consists of 40 seen classes and 10 unseen classes with 37k images, and CUB (Welinder et al., 2010) consists of 150 seen classes and 50 unseen classes with 12k images,[6] where the taxonomy can be built in the same way with Section 2.4.

Training. We note that the performance of combined models is reported in Akata et al. (2015), but the numbers are outdated, due to the old CNNs and ZSL models. Thus, instead of making direct comparison with theirs, we construct the environment following the state-of-the-art setting and compare the performance gain obtained by combining different hierarchical embedding models with other semantic embeddings. We take ResNet-101 as a pretrained visual feature extractor, and we apply deep embedding model proposed in Zhang et al. (2016a) for training attribute and word embedding models, where it learns to map semantic embeddings to the visual feature embedding with two fully connected layers and ReLU between them. As a combination strategy, we calculate prediction scores of each model and then use their weighted sum for final decision, where the weights are cross-validated. See Akata et al. (2015) for more details about the combination strategy.

Metrics. The ZSL performance is measured by testing unseen classes only, and the GZSL performance is measured by the area under seen-unseen curve (AUC) following the idea in Chao et al. (2016). We measure the class-wise accuracy rather than the sample-wise accuracy to avoid the effect of imbalanced test dataset, as suggested in Xian et al. (2017).

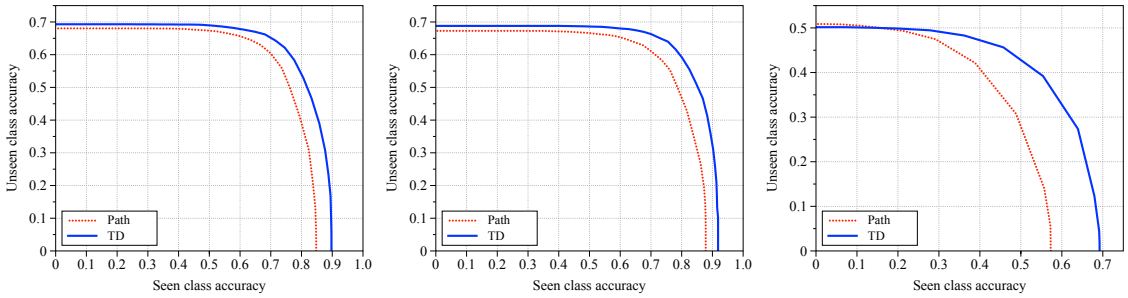
[6] In GZSL, we have semantic information of unseen classes. In this sense, although unseen classes are not used for training, they are *known* as such a class-specific semantic information is required.

Table 2.2: ZSL and GZSL performance of semantic embedding models and their combinations on AwA1, AwA2, and CUB. "Att" stands for continuous attributes labeled by human, "Word" stands for word embedding trained with the GloVe objective (Pennington et al., 2014), and "Hier" stands for the hierarchical embedding, where "Path" is proposed in Akata et al. (2015), and "TD" is output of the proposed top-down method. "Unseen" is the accuracy when only unseen classes are tested, and "AUC" is the area under the seen-unseen curve where the unseen class score bias is varied for computation. The curve used to obtain AUC is shown in Figure 2.6. Values in bold indicate the best performance among the combined models.

Embedding			AwA1		AwA2		CUB	
Att	Word	Hier	Unseen	AUC	Unseen	AUC	Unseen	AUC
✓	-	-	65.29	50.02	63.87	51.27	50.05	23.60
-	✓	-	51.87	39.67	54.77	42.21	27.28	11.47
✓	✓	-	67.80	52.84	65.76	53.18	49.83	24.13
-	-	Path	42.57	30.58	44.34	33.44	24.22	8.38
✓	-	Path	67.09	51.45	66.58	53.50	50.25	23.70
-	✓	Path	52.89	40.66	55.28	42.86	27.72	11.65
✓	✓	Path	68.04	53.21	67.28	54.31	**50.87**	24.20
-	-	TD	33.86	25.56	31.84	24.97	13.09	7.20
✓	-	TD	66.13	54.66	66.86	57.49	50.17	30.31
-	✓	TD	56.14	46.28	59.67	49.39	29.05	16.73
✓	✓	TD	**69.23**	**57.67**	**68.80**	**59.24**	50.17	**30.31**

2.5.2 Experimental Results

Table 2.2 shows the performance of the attribute and word embedding models, and two different hierarchical embedding models, i.e., Path and TD, and their combinations on AwA1, AwA2, and CUB. In Table 2.2, one can note that the standalone performance of TD is not better than Path, as it does not distinguish unseen classes sharing the same closest super class. In the same reason, the improvement on ZSL performance with the combined models is fairly small in the combination with TD. However, in the GZSL task, TD shows significantly better performance in the combined models, which means that our proposed top-down embedding is better when distinguishing both seen classes and unseen classes together. Compared to the best single semantic embedding model (with attributes), the combination with TD leads to absolute improvement of AUC by 7.65%, 7.97%, and 6.71% on AwA1, AwA2 and CUB, respectively, which is significantly better than that of Path.



(a) AwA1 (b) AwA2 (c) CUB

Figure 2.6: Seen-unseen class accuracy curves of the best combined models obtained by varying the unseen class score bias on AwA1, AwA2, and CUB. "Path" is the hierarchical embedding proposed in Akata et al. (2015), and "TD" is the embedding of the multiple softmax probability vector obtained from the proposed top-down method. In most regions, TD outperforms Path.

2.6 Summary

We propose a new hierarchical novelty detection framework, which performs object classification and hierarchical novelty detection by predicting the closest super class in a taxonomy. We propose several methods for the hierarchical novelty detection task and evaluate their performance. In addition, the hierarchical embedding learned with our model can be combined with other semantic embeddings such as attributes and words to improve generalized zero-shot learning performance. As future work, augmenting textual information about labels for hierarchical novelty detection would be an interesting extension of this work.

CHAPTER III

Overcoming Catastrophic Forgetting with Unlabeled Data in the Wild

Lifelong learning with deep neural networks is well-known to suffer from catastrophic forgetting: the performance on previous tasks drastically degrades when learning a new task. To alleviate this effect, we propose to leverage a large stream of unlabeled data easily obtainable in the wild. In particular, we design a novel class-incremental learning scheme with (a) a new distillation loss, termed global distillation, (b) a learning strategy to avoid overfitting to the most recent task, and (c) a confidence-based sampling method to effectively leverage unlabeled external data. Our experimental results on various datasets, including CIFAR and ImageNet, demonstrate the superiority of the proposed methods over prior methods, particularly when a stream of unlabeled data is accessible: our method shows up to 15.8% higher accuracy and 46.5% less forgetting compared to the state-of-the-art method.

3.1 Introduction

Deep neural networks (DNNs) have achieved remarkable success in many machine learning applications, e.g., classification (He et al., 2016), generation (Miyato et al., 2018a), object detection (He et al., 2017), and reinforcement learning (Silver et al., 2017). However, in the real world where the number of tasks continues to grow, the entire tasks cannot be given at once; rather, it may be given as a sequence of tasks. The goal of class-incremental learning (Rebuffi et al., 2017) is to enrich the ability of a model dealing with such a case, by aiming to perform both previous and new tasks well.[1] In particular, it has gained much attention

[1] In class-incremental learning, a set of classes is given in each task. In evaluation, it aims to classify data in any class learned so far without task boundaries.

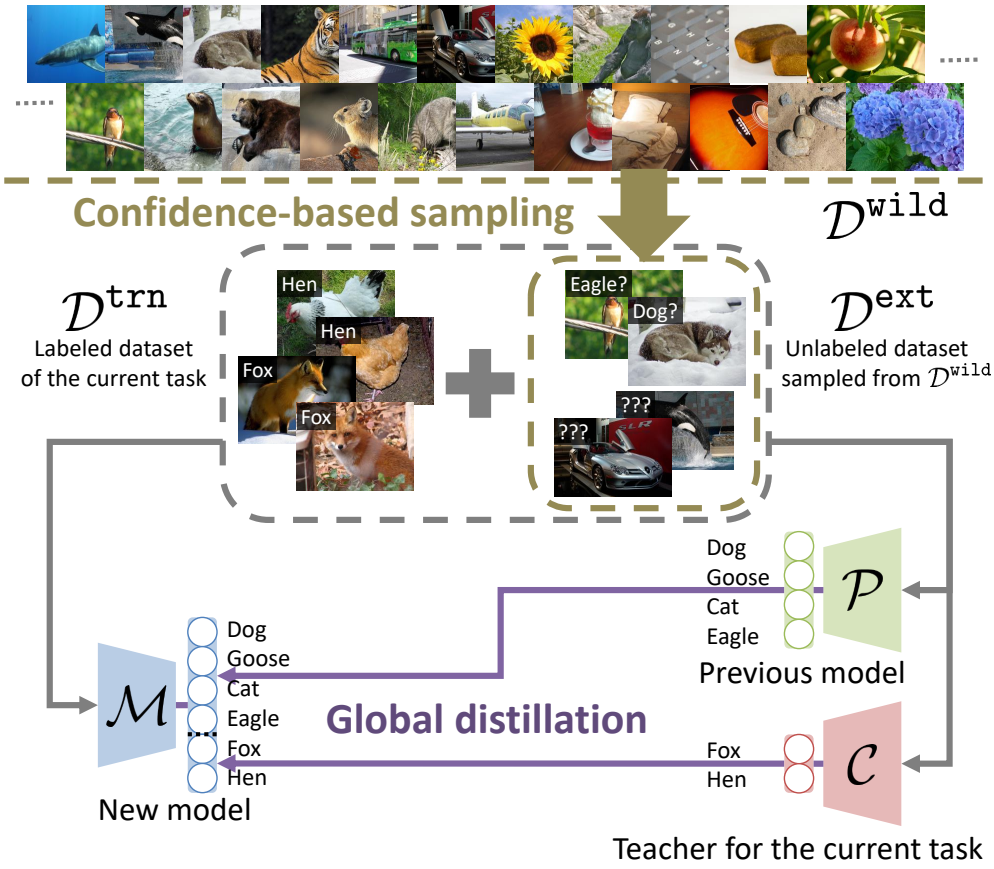


Figure 3.1: We propose to leverage a large stream of unlabeled data in the wild for class-incremental learning. At each stage, a confidence-based sampling strategy is applied to build an external dataset. Specifically, some of unlabeled data are sampled based on the prediction of the model learned in the previous stage $\mathcal{P}$ for alleviating catastrophic forgetting, and some of them are randomly sampled for confidence calibration. Under the combination of the labeled training dataset and the unlabeled external dataset, a teacher model $\mathcal{C}$ first learns the current task, and then the new model $\mathcal{M}$ learns both the previous and current tasks by distilling the knowledge of $\mathcal{P}$, $\mathcal{C}$, and their ensemble $\mathcal{Q}$.

recently as DNNs tend to forget previous tasks easily when learning new tasks, which is a phenomenon called catastrophic forgetting (McCloskey and Cohen, 1989; French, 1999).

The primary reason of catastrophic forgetting is the limited resources for scalability: all training data of previous tasks cannot be stored in a limited size of memory as the number of tasks increases. Prior works in class-incremental learning focused on learning in a closed environment, i.e., a model can only see the given labeled training dataset during training (Li and Hoiem, 2016; Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2018; Li et al., 2018). However, in the real world, we live with a continuous and large stream of unlabeled data easily obtainable on the fly or transiently, e.g., by data mining on social media (Mahajan et al., 2018) and web data (Krause et al., 2016). Motivated by this, we propose to leverage

such a large stream of unlabeled external data for overcoming catastrophic forgetting. We remark that our setup on unlabeled data is similar to self-taught learning (Raina et al., 2007) rather than semi-supervised learning, because we do not assume any correlation between unlabeled data and the labeled training dataset.

Contribution. Under the new class-incremental setup, our contribution is three-fold (see Figure 3.1 for an overview):

$\mathcal{A}$. We propose a new learning objective, termed global distillation, which utilizes data to distill the knowledge of reference models effectively.

$\mathcal{B}$. We design a 3-step learning scheme to improve the effectiveness of global distillation: (i) training a teacher specialized for the current task, (ii) training a model by distilling the knowledge of the previous model, the teacher learned in (i), and their ensemble, and (iii) fine-tuning to avoid overfitting to the current task.

$\mathcal{C}$. We propose a confidence-based sampling method to effectively leverage a large stream of unlabeled data.

In the contribution $\mathcal{A}$, global distillation encourages the model to learn knowledge over all previous tasks, while prior works only applied a task-wise local distillation (Li and Hoiem, 2016; Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2018). In particular, the proposed global distillation distills the knowledge of how to distinguish classes across different tasks, while local distillation does not. We show that the performance gain due to global distillation is particularly significant if some unlabeled external data are available.

In the contribution $\mathcal{B}$, the first two steps (i), (ii) of the proposed learning scheme are designed to keep the knowledge of the previous tasks, as well as to learn the current task. On the other hand, the purpose of the last step (iii) is to avoid overfitting to the current task: due to the scalability issue, only a small portion of data in the previous tasks are kept and replayed during training (Castro et al., 2018; Rebuffi et al., 2017; Nguyen et al., 2018). This inevitably incurs bias in the prediction of the learned model, being favorable for the current task. To mitigate the issue of imbalanced training, we fine-tune the model based on the statistics of data in the previous and current tasks.

Finally, the contribution $\mathcal{C}$ is motivated from the intuition that as the data distribution of unlabeled data is more similar to that of the previous tasks, it prevents the model from catastrophic forgetting more. Since unlabeled data in the wild is not necessarily related to the previous tasks, it is far from being clear whether they contain an useful information for alleviating catastrophic forgetting. Therefore, we propose to sample an external dataset

by a principled sampling strategy. To sample an effective external dataset from a large stream of unlabeled data, we propose to train a confidence-calibrated model (Lee et al., 2018a,b) by utilizing irrelevant data as out-of-distribution (OOD)[2] samples. We show that unlabeled data from OOD should also be sampled for maintaining the model to be more confidence-calibrated.

Our experimental results demonstrate the superiority of the proposed methods over prior methods. In particular, we show that the performance gain in the proposed methods is more significant when unlabeled external data are available. For example, under our experiment setup on ImageNet (Deng et al., 2009), our method with an external dataset achieves 15.8% higher accuracy and 46.5% less forgetting compared to the state-of-the-art method (E2E) (Castro et al., 2018) (4.8% higher accuracy and 6.0% less forgetting without an external dataset).

3.2 Approach

In this section, we propose a new learning method for class-incremental learning. In Section 3.2.1, we further describe the scenario and learning objectives. In Section 3.2.2, we propose a novel learning objective, termed global distillation. In Section 3.2.3, we propose a confidence-based sampling strategy to build an external dataset from a large stream of unlabeled data.

3.2.1 Preliminaries: Class-Incremental Learning

Formally, let $(x, y) \in \mathcal{D}$ be a data x and its label y in a dataset $\mathcal{D}$, and let $\mathcal{T}$ be a supervised task mapping x to y. We denote $y \in \mathcal{T}$ if y is in the range of $\mathcal{T}$ such that $|\mathcal{T}|$ is the number of class labels in $\mathcal{T}$. For the t-th task $\mathcal{T}_t$, let $\mathcal{D}_t$ be the corresponding training dataset, and $\mathcal{D}_{t-1}^{\texttt{cor}} \subseteq \mathcal{D}_{t-1} \cup \mathcal{D}_{t-2}^{\texttt{cor}}$ be a coreset[3] containing representative data of previous tasks $\mathcal{T}_{1:(t-1)} = \{\mathcal{T}_1, \ldots, \mathcal{T}_{t-1}\}$, such that $\mathcal{D}_t^{\texttt{trn}} = \mathcal{D}_t \cup \mathcal{D}_{t-1}^{\texttt{cor}}$ is the entire labeled training dataset available at the t-th stage. Let $\mathcal{M}_t = \{\theta, \phi_{1:t}\}$ be the set of learnable parameters of a model, where θ and $\phi_{1:t} = \{\phi_1, \ldots, \phi_t\}$ indicate shared and task-specific parameters, respectively.[4]

[2] Out-of-distribution refers to the data distribution being far from those of the tasks learned so far.

[3] Coreset is a small dataset kept in a limited amount of memory used to replay previous tasks. Initially, $\mathcal{D}_0^{\texttt{cor}} = \emptyset$.

[4] If multiple task-specific parameters are given, then logits of all classes are concatenated for prediction without task boundaries. Note that tasks do not have to be disjoint, such that a class can appear in multiple tasks.

The goal at the t-th stage is to train a model $\mathcal{M}_t$ to perform the current task $\mathcal{T}_t$ as well as the previous tasks $\mathcal{T}_{1:(t-1)}$ without task boundaries, i.e., all class labels in $\mathcal{T}_{1:t}$ are candidates at test time. To this end, a small coreset $\mathcal{D}_{t-1}^{\texttt{cor}}$ and the previous model $\mathcal{M}_{t-1}$ are transferred from the previous stage. We also assume that a large stream of unlabeled data is accessible, and an essential external dataset $\mathcal{D}_t^{\texttt{ext}}$ is sampled, where the sampling method is described in Section 3.2.3. Note that we do not assume any correlation between the stream of unlabeled data and the tasks. The outcome at the t-th stage is the model $\mathcal{M}_t$ that can perform all observed tasks $\mathcal{T}_{1:t}$, and the coreset $\mathcal{D}_t^{\texttt{cor}}$ for learning in subsequent stages.

Learning objectives. When a dataset $\mathcal{D}$ is labeled, the standard way of training a classification model $\mathcal{M} = \{\theta, \phi\}$ is to optimize the cross-entropy loss:

$$\mathcal{L}_{\texttt{cls}}(\theta, \phi; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} [-\log p(y|x; \theta, \phi)].$$

On the other hand, if we have a reference model $\mathcal{R} = \{\theta^{\mathcal{R}}, \phi^{\mathcal{R}}\}$, the dataset $\mathcal{D}$ does not require any label because the target label is given by $\mathcal{R}$:

$$\mathcal{L}_{\texttt{dst}}(\theta, \phi; \mathcal{R}, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{x\in\mathcal{D}} \sum_{y\in\mathcal{T}} [-p(y|x; \theta^{\mathcal{R}}, \phi^{\mathcal{R}}) \log p(y|x; \theta, \phi)],$$

where the probabilities can be smoothed for better distillation: let $z = \{z_y | y \in \mathcal{T}\} = \mathcal{M}(x; \theta, \phi)$ be the set of outputs (or logits). Then, with a temperature γ, the probabilities are computed as follows:

$$p(y = k|x; \theta, \phi) = \frac{\exp(z_k/\gamma)}{\sum_{y'\in\mathcal{T}} \exp(z_{y'}/\gamma)}.$$

Previous approaches. At the t-th stage, the standard approach to train a model $\mathcal{M}_t$ is to minimize the following classification loss:

$$\mathcal{L}_{\texttt{cls}}(\theta, \phi_{1:t}; \mathcal{D}_t^{\texttt{trn}}). \tag{3.1}$$

However, in class-incremental learning, the limited size of the coreset makes the learned model suffer from catastrophic forgetting. To overcome this, the previous model $\mathcal{P}_t = \{\theta^{\mathcal{P}}, \phi^{\mathcal{P}}_{1:(t-1)}\} \triangleq \mathcal{M}_{t-1}$ has been utilized to generate soft labels, which is the knowledge of $\mathcal{P}_t$ about the given data (Li and Hoiem, 2016; Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2018):

$$\sum_{s=1}^{t-1} \mathcal{L}_{\texttt{dst}}(\theta, \phi_s; \mathcal{P}_t, \mathcal{D}_t^{\texttt{trn}}), \tag{3.2}$$

where this objective is jointly optimized with Eq. (3.1). We call this task-wise knowledge distillation as local distillation (LD), which transfers the knowledge within each of the tasks. However, because they are defined in a task-wise manner, this objective misses the knowledge about discrimination between classes in different tasks.

3.2.2 Global Distillation

Motivated by the limitation of LD, we propose to distill the knowledge of reference models *globally*. With the reference model $\mathcal{P}_t$, the knowledge can be globally distilled by minimizing the following loss:

$$\mathcal{L}_{\texttt{dst}}(\theta, \phi_{1:(t-1)}; \mathcal{P}_t, \mathcal{D}_t^{\texttt{trn}} \cup \mathcal{D}_t^{\texttt{ext}}). \tag{3.3}$$

However, learning by minimizing Eq. (3.3) would cause a bias: since $\mathcal{P}_t$ did not learn to perform the current task $\mathcal{T}_t$, the knowledge about the current task would not be properly learned when only Eq. (3.1)+(3.3) are minimized, i.e., the performance on the current task would be unnecessarily sacrificed. To compensate for this, we introduce another teacher model $\mathcal{C}_t = \{\theta^{\mathcal{C}}, \phi_t^{\mathcal{C}}\}$ specialized for the current task $\mathcal{T}_t$:

$$\mathcal{L}_{\texttt{dst}}(\theta, \phi_t; \mathcal{C}_t, \mathcal{D}_t^{\texttt{trn}} \cup \mathcal{D}_t^{\texttt{ext}}). \tag{3.4}$$

This model can be trained by minimizing the standard cross-entropy loss:

$$\mathcal{L}_{\texttt{cls}}(\theta^{\mathcal{C}}, \phi_t^{\mathcal{C}}; \mathcal{D}_t). \tag{3.5}$$

Note that only the dataset of the current task $\mathcal{D}_t$ is used, because $\mathcal{C}_t$ is specialized for the current task only. We revise this loss in Section 3.2.3 for better external data sampling.

However, as $\mathcal{P}_t$ and $\mathcal{C}_t$ learned to perform only $\mathcal{T}_{1:(t-1)}$ and $\mathcal{T}_t$, respectively, discrimination between $\mathcal{T}_{1:(t-1)}$ and $\mathcal{T}_t$ is not possible with the knowledge distilled from these two reference models. To complete the missing knowledge, we define $\mathcal{Q}_t$ as an ensemble of $\mathcal{P}_t$ and $\mathcal{C}_t$: let

$$p_{\max} = \max_y p(y|x; \theta^{\mathcal{P}}, \phi_{1:(t-1)}^{\mathcal{P}}), \quad y_{\max} = \arg\max_y p(y|x; \theta^{\mathcal{P}}, \phi_{1:(t-1)}^{\mathcal{P}}).$$

Then, the output of $\mathcal{Q}_t$ can be defined as:

$$p(y|x;\theta^{\mathcal{Q}},\phi^{\mathcal{Q}}_{1:t}) = \begin{cases} p_{\max} & \text{if } y = y_{\max}, \\ \frac{1-p_{\max}-\varepsilon}{1-p_{\max}} p(y|x;\theta^{\mathcal{P}},\phi^{\mathcal{P}}_{1:(t-1)}) & \text{else if } y \in \mathcal{T}_{1:(t-1)}, \\ \varepsilon p(y|x;\theta^{\mathcal{C}},\phi^{\mathcal{C}}_{t}) & \text{else if } y \in \mathcal{T}_t, \end{cases} \tag{3.6}$$

such that $\sum_y p(y|x;\theta^{\mathcal{Q}},\phi^{\mathcal{Q}}_{1:t}) = 1$. Here, ε adjusts the confidence about whether the given data is in $\mathcal{T}_{1:(t-1)}$ or $\mathcal{T}_t$. This information is basically missing, however, can be computed with an assumption that the expected predicted probability is the same over all negative classes $\forall y \neq y_{\max}$, i.e., $\mathbb{E}_y\left[p_\varepsilon(y|x;\theta^{\mathcal{P}},\phi^{\mathcal{P}}_{1:(t-1)})\right] = \mathbb{E}_{y\neq y_{\max}}\left[p_\varepsilon(y|x;\theta^{\mathcal{C}},\phi^{\mathcal{C}}_{t})\right]$:

$$\varepsilon = \frac{(1-p_{\max})|\mathcal{T}_t|}{|\mathcal{T}_{1:t}|-1}. \tag{3.7}$$

Since the ensemble model $\mathcal{Q}_t$ is able to perform all tasks, all parameters can be updated:

$$\mathcal{L}_{\texttt{dst}}(\theta,\phi_{1:t};\mathcal{Q}_t,\mathcal{D}^{\texttt{ext}}_t). \tag{3.8}$$

Note that the labeled dataset $\mathcal{D}^{\texttt{trn}}_t$ is not used, because it is already used in Eq. (3.1) for the same range of classes.

Finally, our global distillation (GD) loss is Eq. (3.1)+(3.3)+(3.4)+(3.8):

$$\begin{aligned} &\mathcal{L}_{\texttt{cls}}(\theta,\phi_{1:t};\mathcal{D}^{\texttt{trn}}_t) + \mathcal{L}_{\texttt{dst}}(\theta,\phi_{1:(t-1)};\mathcal{P}_t,\mathcal{D}^{\texttt{trn}}_t\cup\mathcal{D}^{\texttt{ext}}_t) \\ &\qquad + \mathcal{L}_{\texttt{dst}}(\theta,\phi_t;\mathcal{C}_t,\mathcal{D}^{\texttt{trn}}_t\cup\mathcal{D}^{\texttt{ext}}_t) \\ &\qquad + \mathcal{L}_{\texttt{dst}}(\theta,\phi_{1:t};\mathcal{Q}_t,\mathcal{D}^{\texttt{ext}}_t). \end{aligned} \tag{3.9}$$

We study the contribution of each term in Table 3.2.

Balanced fine-tuning. The statistics of class labels in the training dataset is also an information learned during training. Since the number of data from the previous tasks is much smaller than that of the current task, the prediction of the model is biased to the current task. To remove the bias, we further fine-tune the model after training with the same learning objectives. When fine-tuning, for each loss with $\mathcal{D}$ and $\mathcal{T}$, we scale the gradient computed from a data with label $k \in \mathcal{T}$ by the following:

$$w^{(k)}_{\mathcal{D}} \propto \frac{1}{|\{(x,y)\in\mathcal{D}|y=k\}|}. \tag{3.10}$$

Since scaling a gradient is equivalent to feeding the same data multiple times, we call this method *data weighting*. We also normalize the weights by multiplying them with $|\mathcal{D}|/|\mathcal{T}|$, such that they are all one if $\mathcal{D}$ is balanced.

We only fine-tune the task-specific parameters $\phi_{1:t}$ with data weighting, because all training data would be equally useful for representation learning, i.e., shared parameters θ, while the bias in the data distribution of the training dataset should be removed when training a classifier, i.e., $\phi_{1:t}$. The effect of balanced fine-tuning can be found in Table 3.4.

Loss weight. We balance the contribution of each loss by the relative size of each task learned in the loss: for each loss for learning $\mathcal{T}$, the loss weight at the t-th stage is

$$w^L = \frac{|\mathcal{T}|}{|\mathcal{T}_{1:t}|}. \tag{3.11}$$

We note that the loss weight can be tuned as a hyperparameter, but we find that this loss weight performs better than other values in general, as it follows the statistics of the test dataset: all classes are equally likely to be appeared.

3-step learning algorithm. In summary, our learning strategy has three steps: training $\mathcal{C}_t$ specialized for the current task $\mathcal{T}_t$, training $\mathcal{M}_t$ by distilling the knowledge of the reference models $\mathcal{P}_t$, $\mathcal{C}_t$, and $\mathcal{Q}_t$, and fine-tuning the task-specific parameters $\phi_{1:t}$ with data weighting. Algorithm 3.1 describes the 3-step learning scheme.

For coreset management, we build a balanced coreset by randomly selecting data for each class. We note that other more sophisticated selection algorithms like herding (Rebuffi et al., 2017) do not perform significantly better than random selection, which is also reported in prior works (Castro et al., 2018; Wu et al., 2018a).

3.2.3 Sampling External Dataset

Although a large amount of unlabeled data would be easily obtainable, there are two issues when using them for knowledge distillation: (a) training on a large-scale external dataset is expensive, and (b) most of the data would not be helpful, because they would be irrelevant to the tasks the model learns. To overcome these issues, we propose to sample an external dataset useful for knowledge distillation from a large stream of unlabeled data. Note that the sampled external dataset does not require an additional permanent memory; it is discarded after learning.

Algorithm 3.1 3-step learning with GD.

$t = 1$
while true do
  Input: previous model $\mathcal{P}_t = \mathcal{M}_{t-1}$, coreset $\mathcal{D}_{t-1}^{\texttt{cor}}$, training dataset $\mathcal{D}_t$, unlabeled data stream $\mathcal{D}_t^{\texttt{wild}}$
  Output: new coreset $\mathcal{D}_t^{\texttt{cor}}$, model $\mathcal{M}_t = \{\theta, \phi_{1:t}\}$
  $\mathcal{D}_t^{\texttt{trn}} = \mathcal{D}_t \cup \mathcal{D}_{t-1}^{\texttt{cor}}$
  $N_C = |\mathcal{D}_{t-1}^{\texttt{cor}}|$, $N_D = |\mathcal{D}_t^{\texttt{trn}}|$
  Sample $\mathcal{D}_t^{\texttt{ext}}$ from $\mathcal{D}_t^{\texttt{wild}}$ using Algorithm 3.2
  Train $\mathcal{C}_t$ by minimizing Eq. (3.12)
  if $t > 1$ **then**
    Train $\mathcal{M}_t$ by minimizing Eq. (3.9)
    Fine-tune $\phi_{1:t}$ by minimizing Eq. (3.9), with data weighting in Eq. (3.10)
  else
    $\mathcal{M}_t = \mathcal{C}_t$
  end if
  Randomly sample $\mathcal{D}_t^{\texttt{cor}} \subseteq \mathcal{D}_t^{\texttt{trn}}$ such that $|\{(x, y) \in \mathcal{D}_t^{\texttt{cor}} | y = k\}| = N_C / |\mathcal{T}_{1:t}|$ for $k \in \mathcal{T}_{1:t}$
  $t = t + 1$
end while

Sampling for confidence calibration. In order to alleviate catastrophic forgetting caused by the imbalanced training dataset, sampling external data that are expected to be in the previous tasks is desirable. Since the previous model $\mathcal{P}$ is expected to produce an output with high confidence if the data is likely to be in the previous tasks, the output of $\mathcal{P}$ can be used for sampling. However, modern DNNs are highly overconfident (Guo et al., 2017; Lee et al., 2018b), thus a model learned with a discriminative loss would produce a prediction with high confidence even if the data is not from any of the previous tasks. Since most of the unlabeled data would not be relevant to any of the previous tasks, i.e., they are considered to be from out-of-distribution (OOD), it is important to avoid overconfident prediction on such irrelevant data. To achieve this, the model should learn to be confidence-calibrated by learning with a certain amount of OOD data as well as data of the previous tasks (Lee et al., 2018a,b). When sampling OOD data, we propose to randomly sample data rather than relying on the confidence of the previous model, as OOD is widely distributed over the data space. The effect of this sampling strategy can be found in Table 3.5. Algorithm 3.2 describes our sampling strategy. The ratio of OOD data ($N_{\text{prev}} : N_{\text{OOD}}$) is determined by validation; for more details, see Appendix B.2. This algorithm can take a long time, but we limit the number of retrieved unlabeled data in our experiment by 1M, i.e., $N_{\text{max}} = 1\text{M}$.

Algorithm 3.2 Sampling external dataset.

Input: previous model $\mathcal{P}_t = \{\theta^{\mathcal{P}}, \phi^{\mathcal{P}}_{1:(t-1)}\}$, unlabeled data stream $\mathcal{D}_t^{\texttt{wild}}$, sample size N_D, number of unlabeled data to be retrieved $N_{\max}$
Output: sampled external dataset $\mathcal{D}_t^{\texttt{ext}}$
$\mathcal{D}^{\text{prev}} = \emptyset$, $\mathcal{D}^{\text{OOD}} = \emptyset$
$N_{\text{prev}} = 0.3N_D$, $N_{\text{OOD}} = 0.7N_D$
$N(k) \triangleq |\{(x, y, p) \in \mathcal{D}^{\text{prev}} | y = k\}|$
while $|\mathcal{D}^{\text{OOD}}| < N_{\text{OOD}}$ **do**
  Get $x \in \mathcal{D}_t^{\texttt{wild}}$ and update $\mathcal{D}^{\text{OOD}} = \mathcal{D}^{\text{OOD}} \cup \{x\}$
end while
$N_{\text{ret}} = N_{\text{OOD}}$
while $N_{\text{ret}} < N_{\max}$ **do**
  Get $x \in \mathcal{D}_t^{\texttt{wild}}$ and compute the prediction of $\mathcal{P}$:
  $\hat{p} = \max_y p\left(y|x; \theta^{\mathcal{P}}, \phi^{\mathcal{P}}_{1:(t-1)}\right)$, $\hat{y} = \arg\max_y p\left(y|x; \theta^{\mathcal{P}}, \phi^{\mathcal{P}}_{1:(t-1)}\right)$
  if $N(\hat{y}) < N_{\text{prev}}/|\mathcal{T}_{1:(t-1)}|$ **then**
    $\mathcal{D}^{\text{prev}} = \mathcal{D}^{\text{prev}} \cup \{(x, \hat{y}, \hat{p})\}$
  else
    Replace the least probable data in class $\hat{y}$: $(x', \hat{y}, p') = \arg\min_{\{(x,y,p) \in \mathcal{D}^{\text{prev}} | y = \hat{y}\}} p$
    if $p' < \hat{p}$ **then**
      $\mathcal{D}^{\text{prev}} = (\mathcal{D}^{\text{prev}} \backslash \{(x', \hat{y}, p')\}) \cup \{(x, \hat{y}, \hat{p})\}$
    end if
  end if
  $N_{\text{ret}} = N_{\text{ret}} + 1$
end while
Return $\mathcal{D}_t^{\texttt{ext}} = \mathcal{D}^{\text{OOD}} \cup \{x | (x, y, p) \in \mathcal{D}^{\text{prev}}\}$

Confidence calibration for sampling. For confidence calibration, we consider the following confidence loss $\mathcal{L}_{\texttt{cnf}}$ to make the model produce confidence-calibrated outputs for data which are not relevant to the tasks the model learns:

$$\mathcal{L}_{\texttt{cnf}}(\theta, \phi; \mathcal{D}) = \frac{1}{|\mathcal{D}||\mathcal{T}|} \sum_{x \in \mathcal{D}} \sum_{y \in \mathcal{T}} [-\log p(y|x; \theta, \phi)].$$

During the 3-step learning, only the first step for training $\mathcal{C}_t$ has no reference model, so it should learn with the confidence loss. For $\mathcal{C}_t$, (x, y) is from OOD if $y \notin \mathcal{T}_t$. Namely, by optimizing the confidence loss under the coreset of the previous tasks $\mathcal{D}_{t-1}^{\texttt{cor}}$ and the external dataset $\mathcal{D}_t^{\texttt{ext}}$, the model learns to produce a prediction with low confidence for OOD data, i.e., uniformly distributed probabilities over class labels. Thus, $\mathcal{C}_t$ learns by optimizing the following:

$$\mathcal{L}_{\texttt{cls}}(\theta^{\mathcal{C}}, \phi_t^{\mathcal{C}}; \mathcal{D}_t) + \mathcal{L}_{\texttt{cnf}}(\theta^{\mathcal{C}}, \phi_t^{\mathcal{C}}; \mathcal{D}_{t-1}^{\texttt{cor}} \cup \mathcal{D}_t^{\texttt{ext}}). \tag{3.12}$$

Note that the model $\mathcal{M}_t$ does not require an additional confidence calibration, because the previous model $\mathcal{P}_t$ is expected to be confidence-calibrated in the previous stage. Therefore, the confidence-calibrated outputs of the reference models are distilled to the model $\mathcal{M}_t$. The effect of confidence loss can be found in Table 3.3.

3.3 Related Work

Continual lifelong learning. Many recent works have addressed catastrophic forgetting with different assumptions. Broadly speaking, there are three different types of works (van de Ven and Tolias, 2018): one is class-incremental learning (Rebuffi et al., 2017; Castro et al., 2018; Wu et al., 2018a), where the number of class labels keeps growing. Another is task-incremental learning (Li and Hoiem, 2016; Hou et al., 2018), where the boundaries among tasks are assumed to be clear and the information about the task under test is given.[5] The last can be seen as data-incremental learning, which is the case when the set of class labels or actions are the same for all tasks (Rusu et al., 2016b; Kirkpatrick et al., 2017; Schwarz et al., 2018).

These works can be summarized as continual learning, and recent works on continual learning have studied two types of approaches to overcome catastrophic forgetting: model-based and data-based. Model-based approaches (Rusu et al., 2016b; Kirkpatrick et al., 2017; Lee et al., 2017; Lopez-Paz et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Chaudhry et al., 2018; Jung et al., 2018; Mallya et al., 2018; Nguyen et al., 2018; Ritter et al., 2018; Schwarz et al., 2018; Serra et al., 2018; Yoon et al., 2018) keep the knowledge of previous tasks by penalizing the change of parameters crucial for previous tasks, i.e., the updated parameters are constrained to be around the original values, and the update is scaled down by the importance of parameters on previous tasks. However, since DNNs have many local optima, there would be better local optima for both the previous and new tasks, which cannot be found by model-based approaches. Data-based approaches (Li and Hoiem, 2016; Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2018; Javed and Shafait, 2018) keep the knowledge of previous tasks by knowledge distillation (Hinton et al., 2015), which minimizes the distance between the manifold of the latent space in the previous and new models. In contrast to model-based approaches, they require data to be fed to get features on the latent space. Therefore, the amount of knowledge kept by knowledge distillation depends on the degree of similarity between the data distribution used to learn the previous tasks in the previous stages and the one used to distill the knowledge in the

[5] The main difference between class- and task-incremental learning is that the model has single- and multi-head output layer, respectively.

later stages. To guarantee to have a certain amount of similar data, some prior works (Castro et al., 2018; Rebuffi et al., 2017; Nguyen et al., 2018) reserved a small amount of memory to keep a coreset, and others (Rannen et al., 2017; Shin et al., 2017; Lesort et al., 2018; van de Ven and Tolias, 2018; Wu et al., 2018a) trained a generative model and replay the generated data when training a new model. Note that the model-based and data-based approaches are orthogonal in most cases, thus they can be combined for better performance (Kim et al., 2018b).

Knowledge distillation in prior works. Our proposed method is a data-based approach, but it is different from prior works (Li and Hoiem, 2016; Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2018), because their model commonly learns with the task-wise local distillation loss in Eq. (3.2). We emphasize that local distillation only preserves the knowledge within each of the previous tasks, while global distillation does the knowledge over all tasks.

Similar to our 3-step learning, Schwarz et al. (2018) and Hou et al. (2018) utilized the idea of learning with two teachers. However, their strategy to keep the knowledge of the previous tasks is different: Schwarz et al. (2018) applied a model-based approach, and Hou et al. (2018) distilled the task-wise knowledge for task-incremental learning.

On the other hand, Castro et al. (2018) had a similar fine-tuning, but they built a balanced dataset by discarding most of the data of the current task and updated the whole networks. However, such undersampling sacrifices the diversity of the frequent classes, which decreases the performance. Oversampling may solve the issue, but it makes the training not scalable: the size of the oversampled dataset increases proportional to the number of tasks learned so far. Instead, we propose to apply data weighting.

Scalability. Early works on continual learning were not scalable since they kept all previous models (Li and Hoiem, 2016; Rusu et al., 2016b; Aljundi et al., 2017; Kirkpatrick et al., 2017; Yoon et al., 2018). However, recent works considered the scalability by minimizing the amount of task-specific parameters (Rebuffi et al., 2017; Schwarz et al., 2018). In addition, data-based methods require to keep either a coreset or a generative model to replay previous tasks. Our method is a data-based approach, but it does not suffer from the scalability issue since we utilize an external dataset sampled from a large stream of unlabeled data. We note that unlike coreset, our external dataset does not require a permanent memory; it is discarded after learning.

3.4 Experiments

3.4.1 Experimental Setup

Compared algorithms. To provide an upper bound of the performance, we compare an *oracle* method, which learns by optimizing Eq. (3.1) while storing all training data of previous tasks and replaying them during training. Also, as a *baseline*, we provide the performance of a model learned without knowledge distillation. Among prior works, three state-of-the-art methods are compared: *learning without forgetting (LwF)* (Li and Hoiem, 2016), *distillation and retrospection (DR)* (Hou et al., 2018), and *end-to-end incremental learning (E2E)* (Castro et al., 2018). For fair comparison, we adapt LwF and DR for class-incremental setting, which are originally evaluated in task-incremental learning setting: specifically, we extend the range of the classification loss, i.e., we optimize Eq. (3.1)+(3.2) and Eq. (3.1)+(3.2)+(3.4) for replication of them.

We do not compare model-based methods, because data-based methods are known to outperform them in class-incremental learning (Lesort et al., 2018; van de Ven and Tolias, 2018), and they are orthogonal to data-based methods, such that they can potentially be combined with our approaches for better performance (Kim et al., 2018b).

Datasets. We evaluate the compared methods on CIFAR-100 (Krizhevsky and Hinton, 2009) and ImageNet ILSVRC 2012 (Deng et al., 2009), where all images are downsampled to 32×32 (Chrabaszcz et al., 2017). For CIFAR-100, similar to prior works (Rebuffi et al., 2017; Castro et al., 2018), we shuffle the classes uniformly at random and split the classes to build a sequence of tasks. For ImageNet, we first sample 500 images per 100 randomly chosen classes for each trial, and then split the classes. To evaluate the compared methods under the environment with a large stream of unlabeled data, we take two large datasets: the TinyImages dataset (Torralba et al., 2008) with 80M images and the entire ImageNet 2011 dataset with 14M images. The classes appeared in CIFAR-100 and ILSVRC 2012 are excluded to avoid any potential advantage from them. At each stage, our sampling algorithm gets unlabeled data from them uniformly at random to form an external dataset, until the number of retrieved samples is 1M.

Following the prior works, we divide the classes into splits of 5, 10, and 20 classes, such that there are 20, 10, and 5 tasks, respectively. For each task size, we evaluate the compared methods ten times with different class orders (different set of classes in the case of ImageNet) and report the mean and standard deviation of the performance.

Evaluation metric. We report the performance of the compared methods in two metrics: the average incremental accuracy (ACC) and the average forgetting (FGT). For simplicity, we assume that the number of test data is the same over all classes. For a test data from the r-th task $(x, y) \in \mathcal{D}_r^{\texttt{test}}$, let $\hat{y}(x; \mathcal{M}_s)$ be the label predicted by the s-th model, such that

$$A_{r,s} = \frac{1}{|\mathcal{D}_r^{\texttt{test}}|} \sum_{(x,y) \in \mathcal{D}_r^{\texttt{test}}} \mathbb{I}(\hat{y}(x; \mathcal{M}_s) = y)$$

measures the accuracy of the s-th model at the r-th task, where $s \geq r$. Note that prediction is done without task boundaries: for example, at the t-th stage, the expected accuracy of random guess is $1/|\mathcal{T}_{1:t}|$, not $1/|\mathcal{T}_r|$. At the t-th stage, ACC is defined as:

$$\text{ACC} = \frac{1}{t-1} \sum_{s=2}^{t} \sum_{r=1}^{s} \frac{|\mathcal{T}_r|}{|\mathcal{T}_{1:s}|} A_{r,s}.$$

Note that the performance of the first stage is not considered, as it is not class-incremental learning. While ACC measures the overall performance directly, FGT measures the amount of catastrophic forgetting, by averaging the performance decay:

$$\text{FGT} = \frac{1}{t-1} \sum_{s=2}^{t} \sum_{r=1}^{s-1} \frac{|\mathcal{T}_r|}{|\mathcal{T}_{1:s}|} (A_{r,r} - A_{r,s}),$$

which is essentially the negative of the backward transfer (Lopez-Paz et al., 2017). Note that smaller FGT is better, which implies that the model less-forgets about the previous tasks.

Hyperparameters. The backbone architecture of all compared models is wide residual networks (Zagoruyko and Komodakis, 2016) with 16 layers, a widen factor of 2 (WRN-16-2), and a dropout rate of 0.3. Note that this has a comparable performance with ResNet-32 (He et al., 2016). The last fully connected layer is considered to be a task-specific layer, and whenever a task with new classes comes in, the layer is extended to produce a prediction for the new classes. The number of parameters in the task-specific layer is small compared to those in shared layers (about 2% in maximum in WRN-16-2). All methods use the same size of coreset, which is 2000. For scalability, the size of the sampled external dataset is set to the size of the labeled dataset, i.e., $N_D = |\mathcal{D}_t^{\texttt{trn}}|$ in Algorithm 3.2. For validation, one split of ImageNet is used, which is exclusive to the other nine trials. The temperature for smoothing softmax probabilities (Hinton et al., 2015) is set to 2 for distillation from $\mathcal{P}$ and $\mathcal{C}$, and 1 for $\mathcal{Q}$. For more details, see Appendix B.2.

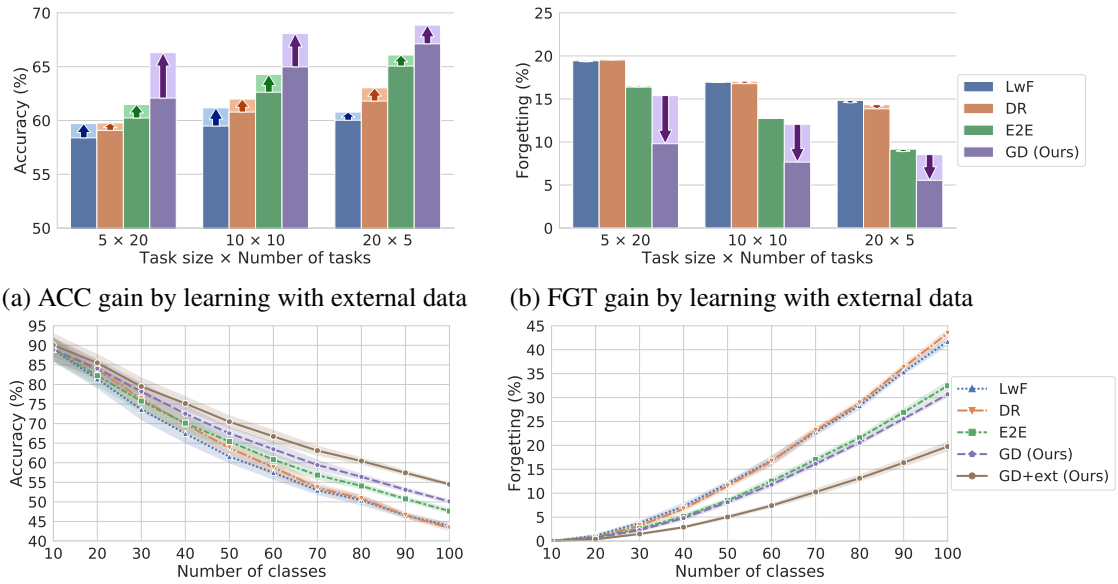
Table 3.1: Comparison of methods on CIFAR-100 and ImageNet. We report the mean and standard deviation of ten trials for CIFAR-100 and nine trials for ImageNet with different random seeds in %. ↑ (↓) indicates that the higher (lower) number is the better.

Dataset	CIFAR-100					
Task size	5		10		20	
Metric	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)
Oracle	78.6 ± 0.9	3.3 ± 0.2	77.6 ± 0.8	3.1 ± 0.2	75.7 ± 0.7	2.8 ± 0.2
Baseline	57.4 ± 1.2	21.0 ± 0.5	56.8 ± 1.1	19.7 ± 0.4	56.0 ± 1.0	18.0 ± 0.3
LwF	58.4 ± 1.3	19.3 ± 0.5	59.5 ± 1.2	16.9 ± 0.4	60.0 ± 1.0	14.5 ± 0.4
DR	59.1 ± 1.4	19.6 ± 0.5	60.8 ± 1.2	17.1 ± 0.4	61.8 ± 0.9	14.3 ± 0.4
E2E	60.2 ± 1.3	16.5 ± 0.5	62.6 ± 1.1	12.8 ± 0.4	65.1 ± 0.8	8.9 ± 0.2
GD (Ours)	62.1 ± 1.2	15.4 ± 0.4	65.0 ± 1.1	12.1 ± 0.3	67.1 ± 0.9	8.5 ± 0.3
+ `ext`	**66.3** ± **1.2**	**9.8** ± **0.3**	**68.1** ± **1.1**	**7.7** ± **0.3**	**68.9** ± **1.0**	**5.5** ± **0.4**
Dataset	ImageNet					
Task size	5		10		20	
Metric	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)
Oracle	68.0 ± 1.7	3.3 ± 0.2	66.9 ± 1.6	3.1 ± 0.3	65.1 ± 1.2	2.7 ± 0.2
Baseline	44.2 ± 1.7	23.6 ± 0.4	44.1 ± 1.6	21.5 ± 0.5	44.7 ± 1.2	18.4 ± 0.5
LwF	45.6 ± 1.9	21.5 ± 0.4	47.3 ± 1.5	18.5 ± 0.5	48.6 ± 1.2	15.3 ± 0.6
DR	46.5 ± 1.6	22.0 ± 0.5	48.7 ± 1.6	18.8 ± 0.5	50.7 ± 1.2	15.1 ± 0.5
E2E	47.7 ± 1.9	17.9 ± 0.4	50.8 ± 1.5	13.4 ± 0.4	53.9 ± 1.2	8.8 ± 0.3
GD (Ours)	50.0 ± 1.7	16.8 ± 0.4	53.7 ± 1.5	12.8 ± 0.5	56.5 ± 1.2	8.4 ± 0.4
+ `ext`	**55.2** ± **1.8**	**9.6** ± **0.4**	**57.7** ± **1.6**	**7.4** ± **0.3**	**58.7** ± **1.2**	**5.4** ± **0.3**

3.4.2 Evaluation

Comparison of methods. Table 3.1 and Figure 3.2 compare our proposed methods with the state-of-the-art methods. First, even when unlabeled data are not accessible, our method outperforms the state-of-the-art methods, which shows the effectiveness of the proposed 3-step learning scheme. Specifically, in addition to the difference in the loss function, DR does not have balanced fine-tuning, E2E lacks the teacher for the current task $\mathcal{C}_t$ and fine-tunes the whole networks with a small dataset, and LwF has neither $\mathcal{C}_t$ nor fine-tuning. Compared to E2E, which is the best state-of-the-art method, our method improves ACC by 4.8% and FGT by 6.0% on ImageNet with a task size of 5.

On the other hand, as shown in Figure 3.2(a)–3.2(b), learning with an unlabeled external dataset improves the performance of compared methods consistently, but the improvement is more significant in GD. For example, in the case of ImageNet with a task size of 5, by learning with the external dataset, E2E improves ACC by 3.2%, while GD does by 10.5%. Also, the relative performance gain in terms of FGT is more significant: E2E forgets 1.1%



(a) ACC gain by learning with external data

(b) FGT gain by learning with external data

(c) ACC w.r.t. the number of trained classes

(d) FGT w.r.t. the number of trained classes

Figure 3.2: Experimental results on CIFAR-100. (a,b) Arrows show the performance gain in the average incremental accuracy (ACC) and average forgetting (FGT) by learning with unlabeled data, respectively. (c,d) Curves show ACC and FGT with respect to the number of trained classes when the task size is 10. We report the average performance of ten trials.

Table 3.2: Comparison of models learned with different reference models on CIFAR-100 when the task size is 10. "$\mathcal{P}$," "$\mathcal{C}$," and "$\mathcal{Q}$" stand for the previous model, the teacher for the current task, and their ensemble model, respectively.

$\mathcal{P}$	$\mathcal{C}$	$\mathcal{Q}$	ACC (↑)	FGT (↓)
✓	-	-	62.9 ± 1.2	14.7 ± 0.4
✓	✓	-	67.0 ± 0.9	10.7 ± 0.3
-	-	✓	65.7 ± 0.9	11.2 ± 0.2
✓	✓	✓	**68.1** ± **1.1**	**7.7** ± **0.3**

less while GD does 43.1%. Overall, with our proposed learning scheme and the external dataset, GD improves its ACC by 15.8% and FGT by 46.5% over E2E.

Effect of the reference models. Table 3.2 shows an ablation study with different set of reference models. As discussed in Section 3.2.2, because the previous model $\mathcal{P}$ does not know about the current task, the compensation by introducing $\mathcal{C}$ improves the overall performance. On the other hand, $\mathcal{Q}$ does not show better ACC than the combination of $\mathcal{P}$ and $\mathcal{C}$. This would be because, when building the output of $\mathcal{Q}$, the ensemble of the output of $\mathcal{P}$ and $\mathcal{C}$ is made with an assumption, which would not always be true. However, the knowledge from $\mathcal{Q}$ is useful, such that the combination of all three reference models shows the best performance.

Table 3.3: Comparison of models learned with a different teacher for the current task $\mathcal{C}$ on CIFAR-100 when the task size is 10. For "cls," $\mathcal{C}$ is not trained but the model learns by optimizing the learning objective of $\mathcal{C}$ directly. The model learns with the proposed 3-step learning for "dst." The confidence loss is added to the learning objective for $\mathcal{C}$ for "cnf." We do not utilize $\mathcal{Q}$ for this experiment, because "cls" has no explicit $\mathcal{C}$.

$\mathcal{C}$	Confidence	ACC (↑)	FGT (↓)
-	-	62.9 ± 1.2	14.7 ± 0.4
cls	-	62.9 ± 1.3	14.5 ± 0.5
cls	cnf	65.3 ± 1.0	11.7 ± 0.3
dst	-	66.2 ± 1.0	11.2 ± 0.3
dst	cnf	**67.0 ± 0.9**	**10.7 ± 0.3**

Table 3.4: Comparison of different balanced learning strategies on CIFAR-100 when the task size is 10. "DW," "FT-DSet," and "FT-DW" stand for training with data weighting in Eq. (3.10) for the entire training, fine-tuning with a training dataset balanced by removing data of the current task, and fine-tuning with data weighting, respectively.

Balancing	ACC (↑)	FGT (↓)
-	67.1 ± 0.9	11.5 ± 0.3
DW	67.9 ± 0.9	9.6 ± 0.2
FT-DSet	67.2 ± 1.1	8.4 ± 0.2
FT-DW	**68.1 ± 1.1**	**7.7 ± 0.3**

Effect of the teacher for the current task $\mathcal{C}$. Table 3.3 compares the models learned with a different teacher for the current task $\mathcal{C}_t$. In addition to the baseline without $\mathcal{C}_t$, we also compare the model directly optimizes the learning objective of $\mathcal{C}_t$ in Eq. (3.5) or (3.12), i.e., the model learns with hard labels rather than soft labels when optimizing that loss. Note that introducing a separate model $\mathcal{C}$ for distillation is beneficial, because $\mathcal{C}$ learns better knowledge about the current task without interference from other classification tasks. Learning by optimizing the confidence loss improves the performance, because the confidence-calibrated model samples better external data as discussed in Section 3.2.3.

Effect of balanced fine-tuning. Table 3.4 shows the effect of balanced learning. First, balanced learning strategies improve FGT in general. If fine-tuning in 3-step learning is skipped but data weighting in Eq. (3.10) is applied in the main training (DW), the model shows higher FGT than having balanced fine-tuning on task-specific parameters (FT-DW), as discussed in Section 3.2.2. Note that data weighting (FT-DW) is better than removing the data of the current task to construct a small balanced dataset (FT-DSet) proposed in Castro et al. (2018), because all training data are useful.

Table 3.5: Comparison of different external data sampling strategies on CIFAR-100 when the task size is 10. "Prev" and "OOD" columns describe the sampling method for data of previous tasks and out-of-distribution data, where "Pred" and "Random" stand for sampling based on the prediction of the previous model $\mathcal{P}$ and random sampling, respectively. In particular, for when sampling OOD by "Pred," we sample data minimizing the confidence loss $\mathcal{L}_{\texttt{cnf}}$. When only Prev or OOD is sampled, the number of sampled data is doubled for fair comparison.

Prev	OOD	ACC (↑)	FGT (↓)
-	-	65.0 ± 1.1	12.1 ± 0.3
-	Random	67.6 ± 0.9	9.0 ± 0.3
Pred	-	66.0 ± 1.2	7.8 ± 0.3
Pred	Pred	65.7 ± 1.1	10.2 ± 0.2
Pred	Random	**68.1** ± **1.1**	**7.7** ± **0.3**

Effect of external data sampling. Table 3.5 compares different external data sampling strategies. Unlabeled data are beneficial in all cases, but the performance gain is different over sampling strategies. First, observe that randomly sampled data are useful, because their predictive distribution would be diverse such that it helps to learn the diverse knowledge of the reference models, which makes the model confidence-calibrated. However, while the random sampling strategy has higher ACC than sampling based on the prediction of the previous model $\mathcal{P}$, it also shows high FGT. This implies that the unlabeled data sampled based on the prediction of $\mathcal{P}$ prevents the model from catastrophic forgetting more. As discussed in Section 3.2.3, our proposed sampling strategy, the combination of the above two strategies shows the best performance. Finally, sampling OOD data based on the prediction of $\mathcal{P}$ is not beneficial, because "data most likely to be from OOD" would not be useful. OOD data sampled based on the prediction of $\mathcal{P}$ have almost uniform predictive distribution, which would be locally distributed. However, the concept of OOD is a kind of complement set of the data distribution the model learns. Thus, to learn to discriminate OOD well in our case, the model should learn with data widely distributed outside of the data distribution of the previous tasks.

3.5 Summary

We propose to leverage a large stream of unlabeled data in the wild for class-incremental learning. The proposed global distillation aims to keep the knowledge of the reference models without task boundaries, leading better knowledge distillation. Our 3-step learning scheme effectively leverages the external dataset sampled by the confidence-based sampling strategy from the stream of unlabeled data.

CHAPTER IV

Network Randomization: A Simple Technique for Generalization in Deep Reinforcement Learning

Deep reinforcement learning (RL) agents often fail to generalize to unseen environments (yet semantically similar to trained agents), particularly when they are trained on high-dimensional state spaces, such as images. In this chapter, we propose a simple technique to improve a generalization ability of deep RL agents by introducing a randomized (convolutional) neural network that randomly perturbs input observations. It enables trained agents to adapt to new domains by learning robust features invariant across varied and randomized environments. Furthermore, we consider an inference method based on the Monte Carlo approximation to reduce the variance induced by this randomization. We demonstrate the superiority of our method across 2D CoinRun, 3D DeepMind Lab exploration and 3D robotics control tasks: it significantly outperforms various regularization and data augmentation methods for the same purpose.

4.1 Introduction

Deep reinforcement learning (RL) has been applied to various applications, including board games (e.g., Go (Silver et al., 2017) and Chess (Silver et al., 2018)), video games (e.g., Atari games (Mnih et al., 2015) and StarCraft (Vinyals et al., 2017)), and complex robotics control tasks (Tobin et al., 2017; Ren et al., 2019). However, it has been evidenced in recent years that deep RL agents often struggle to generalize to new environments, even when semantically similar to trained agents (Farebrother et al., 2018; Zhang et al., 2018b; Cobbe et al., 2019; Gamrian and Goldberg, 2019). For example, RL agents that learned a near-optimal policy for training levels in a video game fail to perform accurately in unseen

(a) Outputs of random layer (b) 2D CoinRun (c) 3D DeepMind Lab (d) 3D Surreal robotics

Figure 4.1: (a) Examples of randomized inputs (color values in each channel are normalized for visualization) generated by re-initializing the parameters of a random layer. Examples of seen and unseen environments on (b) CoinRun, (c) DeepMind Lab, and (d) Surreal robotics control.

levels (Cobbe et al., 2019), while a human can seamlessly generalize across similar tasks. Namely, RL agents often overfit to training environments, thus the lack of generalization ability makes them unreliable in several applications, such as health care (Chakraborty and Murphy, 2014) and finance (Deng et al., 2016).

Generalization in RL can be characterized by visual changes (Cobbe et al., 2019; Gamrian and Goldberg, 2019), different dynamics (Packer et al., 2018), and various structures (Beattie et al., 2016; Wang et al., 2016). In this chapter, we focus on the generalization across tasks where the trained agents take various unseen visual patterns at test time, e.g., different styles of backgrounds, floors, and other objects (see Figure 4.1). We also found that RL agents completely fail due to small visual changes because it is challenging to learn generalizable representations from high-dimensional input observations, such as images.

To improve generalization, several strategies, such as regularization (Farebrother et al., 2018; Zhang et al., 2018b; Cobbe et al., 2019) and data augmentation (Tobin et al., 2017; Ren et al., 2019), have been proposed in the literature. In particular, Tobin et al. (2017) showed that training RL agents in various environments generated by randomizing rendering in a simulator improves the generalization performance, leading to a better performance in real environments. This implies that RL agents can learn invariant and robust representations if diverse input observations are provided during training. However, their method is limited by requiring a physics simulator, which may not always be available. This motivates our approach of developing a simple and plausible method for training RL agents.

The main contribution of this chapter is to develop a simple randomization technique for improving the generalization ability across tasks with various unseen visual patterns. Our main idea is to utilize random (convolutional) networks to generate randomized inputs

(see Figure 4.1(a)), and train RL agents by feeding them into the networks. Specifically, by re-initializing the parameters of random networks at every iteration, the agents are encouraged to be trained under a broad range of perturbed low-level features, e.g., various textures, colors, or shapes. We discover that the proposed idea guides RL agents to learn generalizable features that are more invariant in unseen environments (see Figure 4.3) than conventional regularization (Srivastava et al., 2014; Ioffe and Szegedy, 2015) and data augmentation (Cobbe et al., 2019; Cubuk et al., 2019a) techniques. Here, we also provide an inference technique based on the Monte Carlo approximation, which stabilizes the performance by reducing the variance incurred from our randomization method at test time.

We demonstrate the effectiveness of the proposed method on the 2D CoinRun (Cobbe et al., 2019) game, the 3D DeepMind Lab exploration task (Beattie et al., 2016), and the 3D robotics control task (Fan et al., 2018). For evaluation, the performance of the trained agents is measured in unseen environments with various visual and geometrical patterns (e.g., different styles of backgrounds, objects, and floors), guaranteeing that the trained agents encounter unseen inputs at test time. Note that learning invariant and robust representations against such changes is essential to generalize to unseen environments. In our experiments, the proposed method significantly reduces the generalization gap in unseen environments unlike conventional regularization and data augmentation techniques. For example, compared to the agents learned with the cutout (DeVries and Taylor, 2017b) data augmentation methods proposed by Cobbe et al. (2019), our method improves the success rates from 39.8% to 58.7% under 2D CoinRun, the total score from 55.4 to 358.2 for 3D DeepMind Lab, and the total score from 31.3 to 356.8 for the Surreal robotics control task. Our results can be influential to study other generalization domains, such as tasks with different dynamics (Packer et al., 2018), as well as solving real-world problems, such as sim-to-real transfer (Tobin et al., 2017).

4.2 Related Work

Generalization in deep RL. Recently, the generalization performance of RL agents has been investigated by splitting training and test environments using random seeds (Zhang et al., 2018a) and distinct sets of levels in video games (Machado et al., 2018; Cobbe et al., 2019). Regularization is one of the major directions to improve the generalization ability of deep RL algorithms. Farebrother et al. (2018) and Cobbe et al. (2019) showed that regularization methods can improve the generalization performance of RL agents using various game modes of Atari (Machado et al., 2018) and procedurally generated arcade environments called CoinRun, respectively. On the other hand, data augmentation techniques

have also been shown to improve generalization. Tobin et al. (2017) proposed a domain randomization method to generate simulated inputs by randomizing rendering in the simulator. Motivated by this, Cobbe et al. (2019) proposed a data augmentation method by modifying the cutout method (DeVries and Taylor, 2017b). Our method can be combined with the prior methods to further improve the generalization performance.

Random networks for deep RL. Random networks have been utilized in several approaches for different purposes in deep RL. Burda et al. (2019) utilized a randomly initialized neural network to define an intrinsic reward for visiting unexplored states in challenging exploration problems. By learning to predict the reward from the random network, the agent can recognize unexplored states. Osband et al. (2018) studied a method to improve ensemble-based approaches by adding a randomized network to each ensemble member to improve the uncertainty estimation and efficient exploration in deep RL. Our method is different from those prior works in that we introduce a random network to improve the generalization ability of RL agents.

Transfer learning. Generalization is also closely related to transfer learning (Parisotto et al., 2016; Rusu et al., 2016a,b), which is used to improve the performance on a target task by transferring the knowledge from a source task. However, unlike supervised learning, it has been observed that fine-tuning a model pre-trained on the source task for adapting to the target task is not beneficial in deep RL. Therefore, Gamrian and Goldberg (2019) proposed a domain transfer method using generative adversarial networks (Goodfellow et al., 2014) and Farebrother et al. (2018) utilized regularization techniques to improve the performance of fine-tuning methods. Higgins et al. (2017) proposed a multi-stage RL, which learns to extract disentangled representations from the input observation and then trains the agents on the representations. Alternatively, we focus on the zero-shot performance of each agent at test time without further fine-tuning of the agent's parameters.

4.3 Network Randomization Technique for Generalization

We consider a standard reinforcement learning (RL) framework where an agent interacts with an environment in discrete time. Formally, at each timestep t, the agent receives a state s_t from the environment[1] and chooses an action a_t based on its policy π. The environment returns a reward r_t and the agent transitions to the next state s_{t+1}. The return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the total accumulated rewards from timestep t with a discount factor $\gamma \in [0, 1)$. RL then maximizes the expected return from each state s_t.

[1]Throughout this chapter, we focus on high-dimensional state spaces, e.g., images.

4.3.1 Training Agents Using Randomized Input Observations

We introduce a random network f with its parameters ϕ initialized with a prior distribution, e.g., Xavier normal distribution (Glorot and Bengio, 2010). Instead of the original input s, we train an agent using a randomized input $\widehat{s} = f(s;\phi)$. For example, in the case of policy-based methods,[2] the parameters θ of the policy network π are optimized by minimizing the following policy gradient objective function:

$$\mathcal{L}^{\texttt{random}}_{\texttt{policy}} = \mathbb{E}_{(s_t,a_t,R_t)\in\mathcal{D}}\big[-\log\pi\left(a_t|f\left(s_t;\phi\right);\theta\right)R_t\big], \tag{4.1}$$

where $\mathcal{D} = \{(s_t, a_t, R_t)\}$ is a set of past transitions with cumulative rewards. By re-initializing the parameters ϕ of the random network per iteration, the agents are trained using varied and randomized input observations (see Figure 4.1(a)). Namely, environments are generated with various visual patterns, but with the same semantics by randomizing the networks. Our agents are expected to adapt to new environments by learning invariant representation (see Figure 4.3 for supporting experiments).

To learn more invariant features, the following feature matching (FM) loss between hidden features from clean and randomized observations is also considered:

$$\mathcal{L}^{\texttt{random}}_{\texttt{FM}} = \mathbb{E}_{s_t\in\mathcal{D}}\big[||h\left(f(s_t;\phi);\theta\right) - h\left(s_t;\theta\right)||^2\big], \tag{4.2}$$

where $h(\cdot)$ denotes the output of the penultimate layer of policy π. The hidden features from clean and randomized inputs are combined to learn more invariant features against the changes in the input observations.[3] Namely, the total loss is:

$$\mathcal{L}^{\texttt{random}} = \mathcal{L}^{\texttt{random}}_{\texttt{policy}} + \beta\mathcal{L}^{\texttt{random}}_{\texttt{FM}}, \tag{4.3}$$

where $\beta > 0$ is a hyper-parameter. The full procedure is summarized in Algorithm 4.1.

Details of the random networks. We propose to utilize a single-layer convolutional neural network (CNN) as a random network, where its output has the same dimension with the input (see Appendix C.4 for additional experimental results on the various types of random networks). To re-initialize the parameters of the random network, we utilize the following mixture of distributions: $P(\phi) = \alpha\mathbb{I}(\phi = \mathbf{I}) + (1-\alpha)\mathcal{N}\left(\mathbf{0}; \sqrt{\frac{2}{n_{\text{in}}+n_{\text{out}}}}\right)$, where $\mathbf{I}$ is an identity kernel, $\alpha \in [0, 1]$ is a positive constant, $\mathcal{N}$ denotes the normal distribution, and

[2] Our method is applicable to the value-based methods as well.

[3] FM loss has also been investigated for various purposes: semi-supervised learning (Miyato et al., 2018b) and unsupervised learning (Salimans et al., 2016; Xie et al., 2019).

Algorithm 4.1 PPO + random networks, Actor-Critic Style

for iteration$= 1, 2, \cdots$ **do**
  Sample the parameter ϕ of random networks from prior distribution $P(\phi)$
  for actor$= 1, 2, \cdots, N$ **do**
    Run policy $\pi\left(a|f\left(s;\phi\right);\theta\right)$ in the given environment for T timesteps
    Compute advantage estimates
  end for
  Optimize $\mathcal{L}^{\texttt{random}}$ in Eq. (4.3) with respect to θ
end for

Method	Classification Accuracy (%)	
	Train (seen)	Test (unseen)
ResNet-18	95.0 $\pm$ 2.4	40.3 $\pm$ 1.2
ResNet-18 + GR	96.4 $\pm$ 1.8	70.9 $\pm$ 1.7
ResNet-18 + CO	95.9 $\pm$ 2.3	41.2 $\pm$ 1.7
ResNet-18 + IV	91.0 $\pm$ 2.0	47.1 $\pm$ 15.1
ResNet-18 + CJ	95.2 $\pm$ 0.6	43.5 $\pm$ 0.3
ResNet-18 + ours	95.9 $\pm$ 1.6	**84.4** $\pm$ 4.5

Table 4.1: Classification accuracy (%) on dogs vs. cats dataset. The results show the mean and standard deviation averaged over three runs and the best result is indicated in bold.

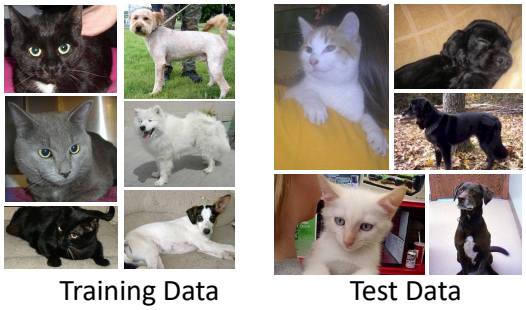


Figure 4.2: Samples of dogs vs. cats dataset. The training set consists of bright dogs and dark cats, whereas the test set consists of dark dogs and bright cats.

$n_{\texttt{in}}, n_{\texttt{out}}$ are the number of input and output channels, respectively. Here, clean inputs are used with the probability α because training only randomized inputs can complicate training. The Xavier normal distribution (Glorot and Bengio, 2010) is used for randomization because it maintains the variance of the input s and the randomized input $\widehat{s}$. We empirically observe that this distribution stabilizes training.

Removing visual bias. To confirm the desired effects of our method, we conduct an image classification experiment on the dogs and cats database from *Kaggle*.[4] Following the same setup as Kim et al. (2019), we construct datasets with an undesirable bias as follows: the training set consists of bright dogs and dark cats while the test set consists of dark dogs and bright cats (see Appendix C.8 for further details). A classifier is expected to make a decision based on the undesirable bias, (e.g., brightness and color) since CNNs are biased towards texture or color, rather than shape (Geirhos et al., 2019). Table 4.1 shows that ResNet-18 (He et al., 2016) does not generalize effectively due to overfitting to an undesirable bias in the training data. To address this issue, several image processing methods

[4] https://www.kaggle.com/c/dogs-vs-cats

(Cubuk et al., 2019a), such as grayout (GR), cutout (CO; DeVries and Taylor, 2017b), inversion (IV), and color jitter (CJ), can be applied (see Appendix C.3 for further details). However, they are not effective in improving the generalization ability, compared to our method. This confirms that our approach makes DNNs capture more desired and meaningful information such as the shape by changing the visual appearance of attributes and entities in images while effectively keeping the semantic information. Prior sophisticated methods (Ganin et al., 2016; Kim et al., 2019) require additional information to eliminate such an undesired bias, while our method does not.[5] Although we mainly focus on RL applications, our idea can also be explorable in this direction.

4.3.2 Inference Methods for Small Variance

Since the parameter of random networks is drawn from a prior distribution $P(\phi)$, our policy is modeled by a stochastic neural network: $\pi(a|s;\theta) = \mathbb{E}_{\phi}\left[\pi\left(a|f\left(s;\phi\right);\theta\right)\right]$. Based on this interpretation, our training procedure (i.e., randomizing the parameters) consists of training stochastic models using the Monte Carlo (MC) approximation (with one sample per iteration). Therefore, at the inference or test time, an action a is taken by approximating the expectations as follows: $\pi\left(a|s;\theta\right) \simeq \frac{1}{M}\sum_{m=1}^{M}\pi\left(a\middle|f\left(s;\phi^{(m)}\right);\theta\right)$, where $\phi^{(m)} \sim P\left(\phi\right)$ and M is the number of MC samples. In other words, we generate M random inputs for each observation and then aggregate their decisions. The results show that this estimator improves the performance of the trained agents by approximating the posterior distribution more accurately (see Figure 4.3(d)).

4.4 Experiments

In this section, we demonstrate the effectiveness of the proposed method on 2D CoinRun (Cobbe et al., 2019), 3D DeepMind Lab exploration (Beattie et al., 2016), and 3D robotics control task (Fan et al., 2018). To evaluate the generalization ability, we measure the performance of trained agents in unseen environments which consist of different styles of backgrounds, objects, and floors. We provide more detailed experimental setups and results in Appendix.

[5] Using the known bias information (i.e., {dark, bright}) and ImageNet pre-trained model, Kim et al. (2019) achieve 90.3%, while our method achieves 84.4% without using both inputs.

4.4.1 Baselines and Implementation Details

For CoinRun and DeepMind Lab experiments, similar to Cobbe et al. (2019), we take the CNN architecture used in IMPALA (Espeholt et al., 2018) as the policy network, and the Proximal Policy Optimization (PPO) (Schulman et al., 2017) method to train the agents.[6] At each timestep, agents are given an observation frame of size 64×64 as input (resized from the raw observation of size 320×240 as in the DeepMind Lab), and the trajectories are collected with the 256-step rollout for training. For Surreal robotics experiments, similar to Fan et al. (2018), the hybrid of CNN and long short-term memory (LSTM) architecture is taken as the policy network, and a distributed version of PPO (i.e., actors collect a massive amount of trajectories, and the centralized learner updates the model parameters using PPO) is used to train the agents.[7] We measure the performance in the unseen environment for every 10M timesteps and report the mean and standard deviation across three runs.

Our proposed method, which augments PPO with random networks and feature matching (FM) loss (denoted PPO + ours), is compared with several regularization and data augmentation methods. As regularization methods, we compare dropout (DO; Srivastava et al., 2014), L2 regularization (L2), and batch normalization (BN; Ioffe and Szegedy, 2015). For those methods, we use the hyperparameters suggested in Cobbe et al. (2019), which are empirically shown to be effective: a dropout probability of 0.1 and a coefficient of 10^{-4} for L2 regularization. We also consider various data augmentation methods: a variant of cutout (CO; DeVries and Taylor, 2017b) proposed in Cobbe et al. (2019), grayout (GR), inversion (IV), and color jitter (CJ) by adjusting brightness, contrast, and saturation (see Appendix C.3 for more details). As an upper bound, we report the performance of agents trained directly on unseen environments, denoted PPO (oracle). For our method, we use $\beta = 0.002$ for the weight of the FM loss, $\alpha = 0.1$ for the probability of skipping the random network, $M = 10$ for MC approximation, and a single-layer CNN with the kernel size of 3 as a random network.

4.4.2 Experiments on CoinRun

Task description. In this task, an agent is located at the leftmost side of the map and the goal is to collect the coin located at the rightmost side of the map within 1,000 timesteps. The agent observes its surrounding environment in the third-person point of view, where the agent is always located at the center of the observation. CoinRun contains an arbitrarily large number of levels which are generated deterministically from a given seed. In each

[6] We referred to https://github.com/openai/coinrun.

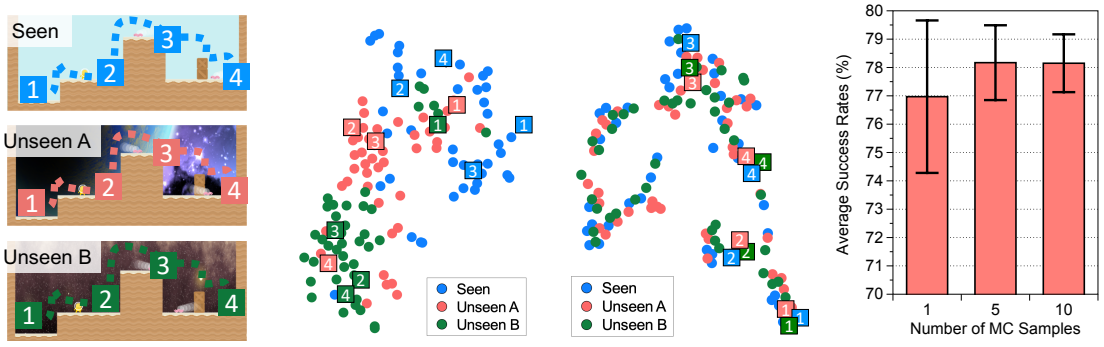
[7] We referred to the two actors setting in https://github.com/SurrealAI/surreal.

		PPO	PPO + DO	PPO + L2	PPO + BN	PPO + CO	PPO + IV	PPO + GR	PPO + CJ	PPO + ours	
										Rand	Rand + FM
Success rate	Seen	100 ± 0.0	100 ± 0.0	98.3 ± 2.9	93.3 ± 11.5	100 ± 0.0	95.0 ± 8.6	100 ± 0.0	100 ± 0.0	95.0 ± 7.1	**100** ± 0.0
	Unseen	34.6 ± 4.5	25.3 ± 12.0	34.1 ± 5.4	31.5 ± 13.1	41.9 ± 5.5	37.5 ± 0.8	26.9 ± 13.1	43.1 ± 1.4	76.7 ± 1.3	**78.1** ± 3.5
Cycle-consistency	2-way	18.9 ± 10.9	13.3 ± 2.2	24.4 ± 1.1	25.5 ± 6.6	27.8 ± 10.6	17.8 ± 15.6	17.7 ± 1.1	32.2 ± 3.1	64.7 ± 4.4	**67.8** ± 6.2
	3-way	4.4 ± 2.2	4.4 ± 2.2	8.9 ± 3.8	7.4 ± 1.2	9.6 ± 5.6	5.6 ± 4.7	2.2 ± 3.8	15.6 ± 3.1	39.3 ± 8.5	**43.3** ± 4.7

Table 4.2: Success rate (%) and cycle-consistency (%) after 100M timesteps in small-scale CoinRun. The results show the mean and standard deviation averaged over three runs and the best results are indicated in bold.

level, the style of background, floor, and obstacles is randomly selected from the available themes (34 backgrounds, 6 grounds, 5 agents, and 9 moving obstacles). Some obstacles and pitfalls are distributed between the agent and the coin, where a collision with them results in the agent's immediate death. We measure the success rates, which correspond to the number of collected coins divided by the number of played levels.

Ablation study on small-scale environments. First, we train agents on one level for 100M timesteps and measure the performance in unseen environments by only changing the style of the background, as shown in Figure 4.3(a). Note that these visual changes are not significant to the game's dynamics, but the agent should achieve a high success rate if it can generalize accurately. However, Table 4.2 shows that all baseline agents fail to generalize to unseen environments, while they achieve a near-optimal performance in the seen environment. This shows that regularization techniques have no significant impact on improving the generalization ability. Even though data augmentation techniques, such as cutout (CO) and color jitter (CJ), slightly improve the performance, our proposed method is most effective because it can produce a diverse novelty in attributes and entities. Training with randomized inputs can degrade the training performance, but the high expressive power of DNNs prevents from it. The performance in unseen environments can be further improved by optimizing the FM loss. To verify the effectiveness of MC approximation at test time, we measure the performance in unseen environments by varying the number of MC samples. Figure 4.3(d) shows the mean and standard deviation across 50 evaluations. The performance and its variance can be improved by increasing the number of MC samples, but the improvement is saturated around ten samples. Thus, we use ten samples for the following experiments.



(a) Trajectories (b) PPO (c) PPO + ours (d) Effects of MC samples

Figure 4.3: (a) We collect multiple episodes from various environments by human demonstrators and visualize the hidden representation of trained agents optimized by (b) PPO and (c) PPO + ours constructed by t-SNE, where the colors of points indicate the environments of the corresponding observations. (d) Average success rates for varying number of MC samples.

Embedding analysis. We analyze whether the hidden representation of trained RL agents exhibits meaningful abstraction in the unseen environments. The features on the penultimate layer of trained agents are visualized and reduced to two dimensions using t-SNE (Maaten and Hinton, 2008). Figure 4.3 shows the projection of trajectories taken by human demonstrators in seen and unseen environments (see Appendix C.13 for further results). Here, trajectories from both seen and unseen environments are aligned on the hidden space of our agents, while the baselines yield scattered and disjointed trajectories. This implies that our method makes RL agents capable of learning invariant and robust representations.

To evaluate the quality of hidden representation quantitatively, the cycle-consistency proposed in Aytar et al. (2018) is also measured. Given two trajectories V and U, $v_i \in V$ first locates its nearest neighbor in the other trajectory $u_j = \arg\min_{u \in U} \|h(v_i) - h(u)\|^2$, where $h(\cdot)$ denotes the output of the penultimate layer of trained agents. Then, the nearest neighbor of u_j in V is located, i.e., $v_k = \arg\min_{v \in V} \|h(v) - h(u_j)\|^2$, and v_i is defined as cycle-consistent if $|i - k| \leq 1$, i.e., it can return to the original point. Note that this cycle-consistency implies that two trajectories are accurately aligned in the hidden space. Similar to Aytar et al. (2018), we also evaluate the three-way cycle-consistency by measuring whether v_i remains cycle-consistent along both paths, $V \to U \to J \to V$ and $V \to J \to U \to V$, where J is the third trajectory. Using the trajectories shown in Figure 4.3(a), Table 4.2 reports the percentage of input observations in the seen environment (blue curve) that are cycle-consistent with unseen trajectories (red and green curves). Similar to the results shown in Figure 4.3(c), our method significantly improves the cycle-consistency compared to the vanilla PPO agent.

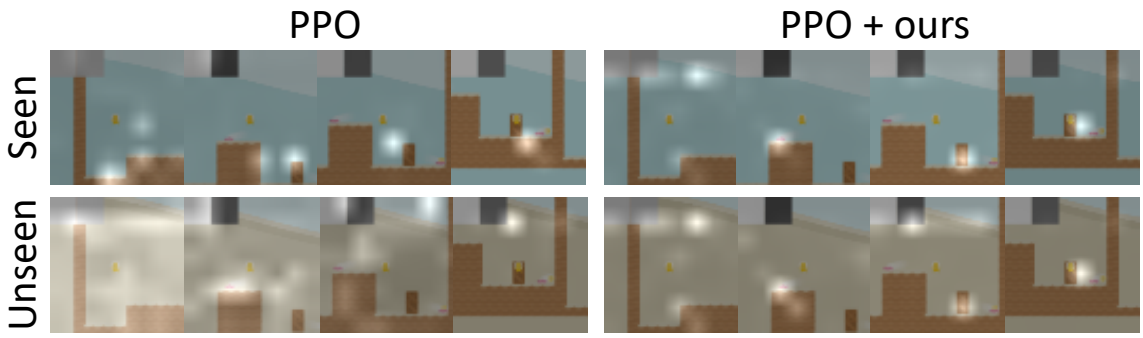
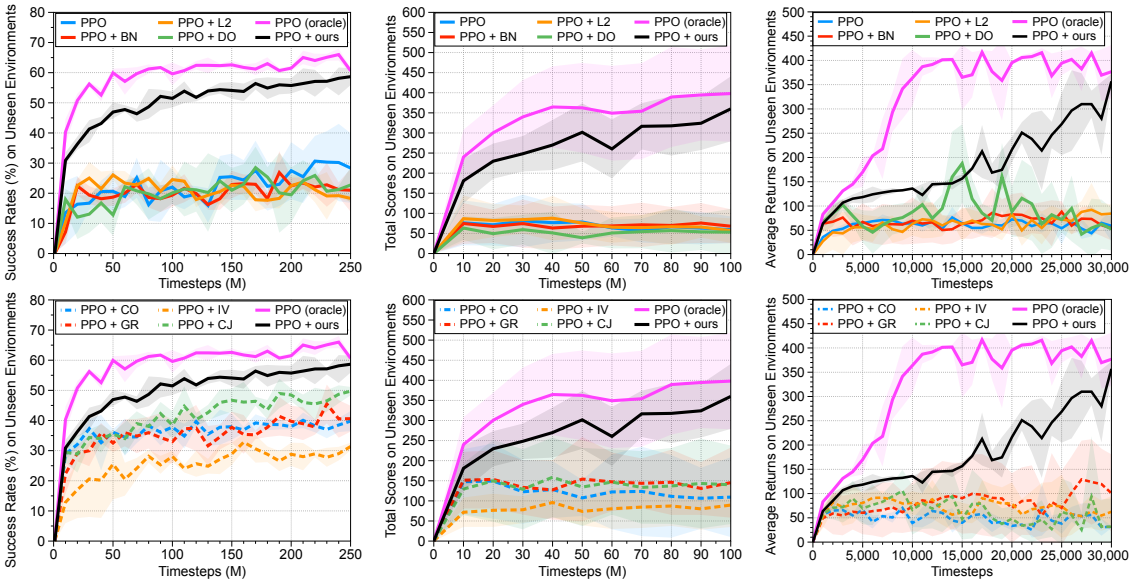


Figure 4.4: Visualization of activation maps via Grad-CAM in seen and unseen environments in the small-scale CoinRun. Images are aligned with similar states from various episodes for comparison.

Visual interpretation. To verify whether the trained agents can focus on meaningful and high-level information, the activation maps are visualized using Grad-CAM (Selvaraju et al., 2017) by averaging activations channel-wise in the last convolutional layer, weighted by their gradients. As shown in Figure 4.4, both vanilla PPO and our agents make a decision by focusing on essential objects, such as obstacles and coins in the seen environment. However, in the unseen environment, the vanilla PPO agent displays a widely distributed activation map in some cases, while our agent does not. As a quantitative metric, we measure the entropy of normalized activation maps. Specifically, we first normalize activations $\sigma_{t,h,w} \in [0,1]$, such that it represents a 2D discrete probability distribution at timestep t, i.e., $\sum_{h=1}^{H}\sum_{w=1}^{W}\sigma_{t,h,w} = 1$. Then, we measure the entropy averaged over the timesteps as follows: $-\frac{1}{T}\sum_{t=1}^{T}\sum_{h=1}^{H}\sum_{w=1}^{W}\sigma_{t,h,w}\log\sigma_{t,h,w}$. Note that the entropy of the activation map quantitatively measures the frequency an agent focuses on salient components in its observation. Results show that our agent produces a low entropy on both seen and unseen environments (i.e., 2.28 and 2.44 for seen and unseen, respectively), whereas the vanilla PPO agent produces a low entropy only in the seen environment (2.77 and 3.54 for seen and unseen, respectively).

Results on large-scale experiments. Similar to Cobbe et al. (2019), the generalization ability by training agents is evaluated on a fixed set of 500 levels of CoinRun. To explicitly separate seen and unseen environments, half of the available themes are utilized (i.e., style of backgrounds, floors, agents, and moving obstacles) for training, and the performances on 1,000 different levels consisting of unseen themes are measured.[8] As shown in Figure 4.5(a), our method outperforms all baseline methods by a large margin. In particular,

[8]Oracle agents are trained on same map layouts with unseen themes to measure the optimal generalization performances on unseen visual patterns.



(a) Large-scale CoinRun (b) DeepMind Lab (c) Surreal robotics

Figure 4.5: The performances of trained agents in unseen environments under (a) large-scale CoinRun, (b) DeepMind Lab and (c) Surreal robotics control. The solid/dashed lines and shaded regions represent the mean and standard deviation, respectively.

the success rates are improved from 39.8% to 58.7% compared to the PPO with cutout (CO) augmentation proposed in Cobbe et al. (2019), showing that our agent learns generalizable representations given a limited number of seen environments.

4.4.3 Experiments on DeepMind Lab and Surreal Robotics Control

Results on DeepMind Lab. We also demonstrate the effectiveness of our proposed method on DeepMind Lab (Beattie et al., 2016), which is a 3D game environment in the first-person point of view with rich visual inputs. The task is designed based on the standard exploration task, where a goal object is placed in one of the rooms in a 3D maze. In this task, agents aim to collect as many goal objects as possible within 90 seconds to maximize their rewards. Once the agent collects the goal object, it receives ten points and is relocated to a random place. Similar to the small-scale CoinRun experiment, agents are trained to collect the goal object in a fixed map layout and tested in unseen environments with only changing the style of the walls and floors. We report the mean and standard deviation of the average scores across ten different map layouts, which are randomly selected. Additional details are provided in Appendix C.7.

Note that a simple strategy of exploring the map actively and recognizing the goal object achieves high scores because the maze size is small in this experiment. Even though the

	PPO				PPO + ours	
	# of Seen Environments	Total Rewards	# of Seen Environments	Total Rewards	# of Seen Environments	Total Rewards
DeepMind Lab	1	55.4 $\pm$ 33.2	16	218.3 $\pm$ 99.2	1	**358.2** $\pm$ 81.5
Surreal Robotics	1	59.2 $\pm$ 31.9	25	168.8 $\pm$ 155.8	1	**356.8** $\pm$ 15.4

Table 4.3: Comparison with domain randomization. The results show the mean and standard deviation averaged over three runs and the best results are indicated in bold.

baseline agents achieve high scores by learning this simple strategy in the seen environment (see Figure C.1(c) in Appendix C.1 for learning curves), Figure 4.5(b) shows that they fail to adapt to the unseen environments. However, the agent trained by our proposed method achieves high scores in both seen and unseen environments. These results show that our method can learn generalizable representations from high-dimensional and complex input observations (i.e., 3D environment).

Results on Surreal robotics control. We evaluate our method in the Block Lifting task using the Surreal distributed RL framework (Fan et al., 2018): the Sawyer robot receives a reward if it succeeds to lift a block randomly placed on a table. We train agents on a single environment and test on five unseen environments with various styles of tables and blocks (see Appendix C.9 for further details). Figure 4.5(c) shows that our method achieves a significant performance gain compared to all baselines in unseen environments while maintaining its performance in the seen environment (see Figure C.8 in Appendix C.9), implying that our method can maintain essential properties, such as structural spatial features of the input observation.

Comparison with domain randomization. To further verify the effectiveness of our method, the vanilla PPO agents are trained by increasing the number of seen environments generated by randomizing rendering in a simulator, while our agent is still trained in a single environment (see Appendix C.7 and C.9 for further details). Table 4.3 shows that the performance of baseline agents can be improved with domain randomization (Tobin et al., 2017). However, our method still outperforms the baseline methods trained with more diverse environments than ours, implying that our method is more effective in learning generalizable representations than simply increasing the (finite) number of seen environments.

4.5 Summary

In this chapter, we explore generalization in RL where the agent is required to generalize to new environments in unseen visual patterns, but semantically similar. To improve the

generalization ability, we propose to randomize the first layer of CNN to perturb low-level features, e.g., various textures, colors, or shapes. Our method encourages agents to learn invariant and robust representations by producing diverse visual input observations. Such invariant features could be useful for several other related topics, like an adversarial defense in RL (see Appendix C.2 for further discussions), sim-to-real transfer (Tobin et al., 2017; Ren et al., 2019), transfer learning (Parisotto et al., 2016; Rusu et al., 2016a,b), and online adaptation (Nagabandi et al., 2019).

CHAPTER V

i-MixUp : Vicinal Risk Minimization for Contrastive Representation Learning

Contrastive learning has demonstrated state-of-the-art performance in self-supervised representation learning. However, a majority of the progress has been made by utilizing domain-specific data augmentation techniques, e.g., in computer vision. In this work, we propose *i-MixUp*, for domain-agnostic contrastive representation learning. First, we cast contrastive learning as learning a non-parametric classifier by assigning a unique virtual class to each data in a batch. Then, i-MixUp linearly interpolates the inputs in the data space and virtual class labels in the label space. In the experiments, we show that i-MixUp consistently improves the quality of image representation, resulting in a performance gain for downstream tasks. The learned representation also has better transferability under distribution shifts. Then, we demonstrate the domain-agnostic representation learning capability of i-MixUp by learning with a limited number of data augmentations. Finally, we evaluate i-MixUp on non-vision domains where data augmentation is not well established.

5.1 Introduction

Representation learning (Lee et al., 2007; Bengio et al., 2013; Razavian et al., 2014; Sermanet et al., 2014) is a fundamental task in machine learning since the success of machine learning relies on the quality of representation. Unsupervised or self-supervised representation learning (Self-SL) (Lee et al., 2007; Bengio et al., 2013) has been successfully applied in several domains, including image recognition (Wu et al., 2018b; Chen et al., 2020a; He et al., 2020), natural language processing (Mikolov et al., 2013), robotics (Ebert et al., 2018; Jang et al., 2018; Sermanet et al., 2018; Lee et al., 2019c), speech recognition (Ravanelli et al., 2020), and and video understanding (Korbar et al., 2018; Owens and Efros, 2018;

Wiles et al., 2018). Since no label is available in the unsupervised setting, pretext tasks are proposed to provide self-supervision: for example, training an autoencoder (Vincent et al., 2010; Kingma and Welling, 2014), clustering (Xie et al., 2016; Caron et al., 2018, 2019; Ji et al., 2019; Asano et al., 2020; Yan et al., 2020), solving jigsaw puzzles (Noroozi and Favaro, 2016; Kim et al., 2018a), and contrastive learning (Wu et al., 2018b; Bachman et al., 2019; Hjelm et al., 2019; Tian et al., 2019; Ye et al., 2019; Chen et al., 2020a,b; He et al., 2020; Misra and van der Maaten, 2020). Self-SL has also been used as an auxiliary task to improve the performance on the main task, such as generative model learning (Chen et al., 2019), semi-supervised learning (Zhai et al., 2019), and improving robustness and uncertainty (Hendrycks et al., 2019).

However, a majority of the pretext tasks rely on the domain-specific inductive bias. For example, solving jigsaw puzzles implicitly assumes that the data domain is a 2D image and the spatial consistency is predictable, e.g., the image is not a repeated pattern. In the case of contrastive representation learning, which has recently shown state-of-the-art performance, domain-specific data augmentation methods are the key of success: as anchors and positive samples are obtained from the same instance, data augmentation introduces semantically meaningful variance for better generalization. To achieve a strong, yet semantically meaningful data augmentation, domain knowledge is required, e.g., color jittering in 2D images or structural information in video understanding.

On the other hand, MixUp (Zhang et al., 2018c) has shown to be a successful domain-agnostic data augmentation method for supervised learning in various domains, including image classification (Zhang et al., 2018c), generative model learning (Lucas et al., 2018), and natural language processing (Guo et al., 2019; Guo, 2020). Inspired by MixUp, we propose i-*MixUp*, a domain-agnostic data augmentation method for contrastive representation learning. The key idea of i-MixUp is to introduce virtual labels in a batch and linearly interpolate instances and their corresponding virtual labels in the input and label spaces, respectively. We first introduce the N-pair loss (Sohn, 2016) adapted to memory-free contrastive learning, as it is simple and efficient when i-MixUp is applied. Then, we show the applicability of i-MixUp in the state-of-the-art memory-based contrastive learning method, MoCo (Chen et al., 2020b; He et al., 2020).

Through the experiments, we demonstrate the efficacy of i-MixUp in the variety of settings: first, we show the effectiveness of i-MixUp by evaluating the discriminative performance of learned representations on image classification tasks. Specifically, we adapt i-MixUp to the state-of-the-art contrastive learning methods, advancing the state-of-the-art performance on CIFAR-10 and 100 (Krizhevsky and Hinton, 2009). Also, we compare contrastive learning methods and other unsupervised representation learning methods, such

as autoencoding (Vincent et al., 2010) in a data-agnostic setting to simulate the case when domain knowledge is not available. Finally, we evaluate i-MixUp on other domains, including a speech dataset (Warden, 2018) and tabular datasets from the UCI machine learning repository (Asuncion and Newman, 2007), and demonstrate the improved contrastive representation learning performance.

Contribution. In summary, our contribution is three-fold:

- We propose i-MixUp as an effective domain-agnostic data augmentation method for contrastive representation learning, which extends MixUp (Zhang et al., 2018c) in supervised learning. We show how to apply i-MixUp in the state-of-the-art memory-free and memory-based contrastive representation learning methods.
- We show that representations learned with i-MixUp have better transferability under distribution shifts, as well as a better quality in the trained data distribution.
- Our extensive experimental results show that i-MixUp consistently improves the performance of contrastive learning in different conditions, including when 1) the domain knowledge for data augmentation is not available, 2) the model or training dataset size is small or large, and 3) even in non-vision domains.

5.2 Related Work

Self-supervised representation learning (Self-SL) aims at learning representations from unlabeled data by solving a pretext task that is different from the downstream task. The most studied pretext task may be data reconstruction by autoencoding (Bengio et al., 2007) and their variants such as denoising (Vincent et al., 2010), context prediction (Doersch et al., 2015), and inpainting (Pathak et al., 2016). Decoder-free Self-SL has made a huge progress in recent years. Exemplar CNN (Dosovitskiy et al., 2014) learns by classifying individual instances with data augmentation. Similarly, noise-as-target (Bojanowski and Joulin, 2017) learns representations by spreading them out on the hypersphere uniformly. Deep clustering (Xie et al., 2016; Caron et al., 2018, 2019; Ji et al., 2019; Asano et al., 2020; Yan et al., 2020) learns by enforcing a clustering assumption on representation space. Self-SL of visual representation, including colorization (Zhang et al., 2016b), solving jigsaw puzzles (Noroozi and Favaro, 2016), counting the number of objects (Noroozi et al., 2017), rotation prediction (Gidaris et al., 2018), next pixel prediction (Oord et al., 2018; Hénaff et al., 2019), and combinations of these (Doersch and Zisserman, 2017; Kim et al., 2018a; Noroozi et al., 2018) often leverages image-specific properties to design pretext tasks.

Contrastive representation learning has gained lots of attention for Self-SL (Wu et al., 2018b; Bachman et al., 2019; Hjelm et al., 2019; Tian et al., 2019; Ye et al., 2019; Chen et al., 2020a,b; He et al., 2020; Misra and van der Maaten, 2020). As opposed to early works on exemplar CNN (Dosovitskiy et al., 2014, 2015), contrastive learning pulls positive pairs and pushes negative pairs in the batch instead of training an instance classifier. Memory-based approaches (Wu et al., 2018b; Bachman et al., 2019; Hjelm et al., 2019; Tian et al., 2019; Misra and van der Maaten, 2020) maintain a memory bank of embedding vectors of instances where the memory is updated by the exponential moving average of input embedding vectors. Different from the prior works, MoCo (Chen et al., 2020b; He et al., 2020) maintained a queue of previously processed embedding vectors as a memory bank. They showed that differentiating the model for anchors and contrastive samples is effective, where the model for contrastive samples is updated by the exponential moving average of the model for anchors. On the other hand, recent works (Tian et al., 2019; Ye et al., 2019; Chen et al., 2020a; Misra and van der Maaten, 2020) showed that learning invariance to different views is important in contrastive representation learning. The views can be generated through data augmentation carefully designed based on domain knowledge (Ye et al., 2019; Chen et al., 2020a), splitting input channels (Tian et al., 2019), or borrowing the idea of other pretext tasks, such as creating a jigsaw or rotating (Misra and van der Maaten, 2020). In particular, SimCLR (Chen et al., 2020a) showed that memory-free approaches with a large batch size and strong data augmentation has a comparable performance to memory-based approaches.

Data augmentation. The purpose of data augmentation is to increase the diversity of data, especially when training data are not enough for generalization. Since the augmented data must be understood as the original data, data augmentation methods are carefully designed based on the domain knowledge on image (DeVries and Taylor, 2017b; Chen et al., 2019; Cubuk et al., 2019a,b; Zhong et al., 2020), speech (Amodei et al., 2016; Park et al., 2019), or natural language (Zhang et al., 2015; Wei and Zou, 2019).

On the other hand, some works have focused on domain-agnostic data augmentation methods: DeVries and Taylor (2017a) proposed to first encode the dataset and then augment data in the feature space. MixUp (Zhang et al., 2018c) is an effective domain-agnostic data augmentation method in supervised learning. Instead of empirical risk minimization, MixUp performs vicinal risk minimization by interpolating input data and their labels on the data and label spaces, respectively. MixUp has also shown its effectiveness in other applications, including generative adversarial networks (Lucas et al., 2018), natural language processing (Guo et al., 2019; Guo, 2020), and improving robustness and uncer-

tainty (Hendrycks et al., 2020). Other variations have also been investigated by interpolating in the feature space (Verma et al., 2019) or leveraging the domain knowledge (Yun et al., 2019).

5.3 Preliminary

5.3.1 Contrastive Representation Learning

Let $\mathcal{X}$ be the data space and $\mathbb{R}^D$ be the D-dimensional embedding space. Contrastive representation learning first constructs a batch of data pairs $\mathcal{B} = \{(x_i, \tilde{x}_i)\}_{i=1}^N$, where N is the batch size, $x_i, \tilde{x}_i \in \mathcal{X}$ are two views (e.g., augmentations) of the same data. Then, it learns a model $f : \mathcal{X} \to \mathbb{R}^D$ by pulling instances in the pair while pushing the rest in the embedding space. Cosine similarity $s(x, \tilde{x}) = (x^\top \tilde{x})/\|x\|\|\tilde{x}\|$ is used for the similarity score, which has shown to be effective (Wu et al., 2018a; Chen et al., 2020a; He et al., 2020). We review various contrastive loss formulations below. For conciseness, $f_i = f(x_i)$ and $\tilde{f}_j = f(\tilde{x}_j)$.

N-pair (Sohn, 2016). In the context of metric learning, Sohn (2016) introduced the N-pair loss. They formulated the negative log-likelihood loss by mining a pair of samples per class: the first samples in the pairs are considered as anchors, and the second samples as positive or negative samples. As we also have a pair of samples per instance, we adapt the N-pair loss for contrastive representation learning. The N-pair contrastive loss function is defined as:[1]

$$\ell_{\text{N-pair}}(x_i; \mathcal{B}) = -\log \frac{\exp\left(s(f_i, \tilde{f}_i)/\tau\right)}{\sum_{k=1}^N \exp\left(s(f_i, \tilde{f}_k)/\tau\right)}. \tag{5.1}$$

SimCLR (Chen et al., 2020a). This method has been proposed as a simple self-supervised contrastive representation learning method without a memory bank. In this formulation, each anchor has a positive sample and $2(N-1)$ negative samples. Let $x_{N+i} = \tilde{x}_i$ for conciseness. Then, the loss function is defined as:

$$\ell_{\text{SimCLR}}(x_i; \mathcal{B}) = -\log \frac{\exp\left(s(f_i, f_{(N+i) \bmod 2N})/\tau\right)}{\sum_{k=1, k\neq i}^{2N} \exp\left(s(f_i, f_k)/\tau\right)}. \tag{5.2}$$

Intuitively, Eq. (5.2) is similar to Eq. (5.1), but introduces more negative samples: while Eq. (5.1) has N negative samples in the batch, Eq. (5.2) has $2(N-1)$ negative samples.

[1] Prior works (Oord et al., 2018; Wu et al., 2018b) have proposed similar losses inspired by noise-contrastive estimation (Gutmann and Hyvärinen, 2010).

MoCo (Chen et al., 2020b; He et al., 2020). While it shares the same batch construction methods for $\mathcal{B}$, MoCo is different from previous methods since it retrieves negative embeddings from the memory bank. Specifically, MoCo maintains a queue of embeddings $\mathcal{M} = \{\mu_k\}_{k=1}^K$ as a memory bank that stores previously extracted embeddings, where K is the size of the memory bank. Another difference is the use of the exponential moving average (EMA) model for positive and negative embedding representations, whose parameters are updated as follows: $\theta_{f^{\text{EMA}}} \leftarrow m\theta_{f^{\text{EMA}}} + (1-m)\theta_f$, where $m \in [0, 1)$ is a momentum coefficient and θ is model parameters. The loss function is written as follows:

$$\ell_{\text{MoCo}}(x_i; \mathcal{B}, \mathcal{M}) = -\log \frac{\exp\left(s(f_i, \tilde{f}_i^{\text{EMA}})/\tau\right)}{\exp\left(s(f_i, \tilde{f}_i^{\text{EMA}})/\tau\right) + \sum_{k=1}^K \exp\left(s(f_i, \mu_k)/\tau\right)}, \tag{5.3}$$

and the memory bank $\mathcal{M}$ is updated with $\{\tilde{f}_i^{\text{EMA}}\}$ in the first-in first-out order.

5.3.2 MixUp in Supervised Learning

Suppose an one-hot label $y_i \in [0, 1]^C$ is assigned to a data x_i, where C is the number of classes. Let a linear classifier predicting the labels consists of weight vectors $\{w_1, \dots, w_C\}$, where $w_c \in \mathbb{R}^D$.[2] Then, the cross-entropy loss for supervised learning is defined as:

$$\ell_{\text{CE}}(x_i, y_i) = -\sum_{c=1}^C y_{i,c} \log \frac{\exp(w_c^\top f_i)}{\sum_{k=1}^C \exp(w_k^\top f_i)}. \tag{5.4}$$

While the cross-entropy loss is widely used for supervised training of deep neural networks, there are several challenges of training with the cross-entropy loss, such as preventing overfitting or networks being overconfident. Several regularization techniques have been proposed to alleviate these issues, including label smoothing (Szegedy et al., 2016), confidence calibration (Lee et al., 2018b), and adversarial training (Miyato et al., 2018b).

MixUp (Zhang et al., 2018c) is another simple and effective regularization method with a minimal computational overhead. It conducts a linear interpolation of two data instances in both input and label spaces and trains a model by minimizing the cross-entropy loss defined on the interpolated data and labels. Specifically, for two data instances (x_i, y_i), (x_j, y_j), the MixUp loss is defined as follows:

$$\ell_{\text{MixUp}}\big((x_i, y_i), (x_j, y_j); \lambda\big) = \ell_{\text{CE}}(\lambda x_i + (1-\lambda)x_j, \lambda y_i + (1-\lambda)y_j), \tag{5.5}$$

where $\lambda \sim \beta(\alpha, \alpha)$ is an interpolation coefficient. MixUp is a vicinal risk minimization

[2] We omit bias terms for presentation clarity.

method (Chapelle et al., 2001) that augments data both at input and output spaces in a data-driven manner. Not only improving the generalization on the supervised task, it is also known to improve adversarial robustness (Zhang et al., 2018c; Pang et al., 2019), confidence calibration or predictive uncertainty (Thulasidasan et al., 2019).

5.4 Vicinal Risk Minimization for Contrastive Representation Learning

As reviewed in Section 5.3, data augmentation is an essential part of contrastive learning (Chen et al., 2020a; He et al., 2020). In this section, we introduce *instance MixUp (i-MixUp)*, a data-driven data augmentation for contrastive representation learning that improves the generalization of self-supervisedly learned representations. Due to its simplicity, we first formulate i-MixUp in the N-pair contrastive loss (Sohn, 2016), which is a memory-free method, and then extend to MoCo (Chen et al., 2020b; He et al., 2020), which is a memory-based method.

5.4.1 i-MixUp for Memory-Free Contrastive Representation Learning

To apply MixUp to contrastive learning, we introduce a *virtual* label for each data instance in a batch. Let $v_i \in [0,1]^N$ be the virtual label of x_i in a batch $\mathcal{B}$, where $v_{i,i} = 1$ and $v_{i,j} = 0$ if $j \neq i$. With virtual labels, the N-pair contrastive loss in Eq. (5.1) can be written in the form of the cross-entropy loss:

$$\ell_{\text{N-pair}}(x_i, v_i; \mathcal{B}) = -\sum_{n=1}^{N} v_{i,n} \log \frac{\exp\left(s(f_i, \tilde{f}_n)/\tau\right)}{\sum_{k=1}^{N} \exp\left(s(f_i, \tilde{f}_k)/\tau\right)}. \tag{5.6}$$

Eq. (5.4) and (5.6) are similar in that both are defined on an input and its label. Their differences include 1) Eq. (5.4) takes a label y_i given in the dataset, while Eq. (5.6) takes a virtual label v_i in the batch, 2) the weight vector w_c in Eq. (5.4) becomes embedding vectors of positive and negative samples $\tilde{f}_n$ in Eq. (5.6), and 3) the inner product between two vectors is replaced with the temperature-scaled similarity. Note that the label space in contrastive learning is $[0,1]^N$, where N is the number of data pairs, or a batch size, instead of C, which is the number of semantic classes in supervised learning. This reformulation of contrastive loss provides an insight for manipulating the label space (as well as the input space) to better regularize the network training. Similarly to MixUp, i-MixUp interpolates two data instances, but in the input and virtual label spaces. Specifically, for two data instances (x_i, v_i), (x_j, v_j) and a batch of data pairs $\mathcal{B} = \{(x_i, \tilde{x}_i)\}_{i=1}^{N}$, the i-MixUp loss is

Algorithm 5.1 Loss for i-MixUp on N-pair contrastive learning in PyTorch-like style.

```
a, b = aug(x), aug(x) # different augmentations of x
lam = Beta(alpha, alpha).sample() # interpolator
randidx = randperm(N) # N = len(x)
a = lam * a + (1-lam) * a[randidx] # input
vlabels = lam * eye(N) + (1-lam) * eye(N)[randidx] # output
anchor, contrast = normalize(model(a)), normalize(model(b))
logits = matmul(anchor, contrast.T) / t # t: temperature
loss = KLDivLoss(log_softmax(logits), vlabels)
```

defined as follows:

$$\ell_{i\text{-MixUp}}\big((x_i, v_i), (x_j, v_j); \mathcal{B}, \lambda\big) = \ell_{\text{N-pair}}(\lambda x_i + (1-\lambda)x_j, \lambda v_i + (1-\lambda)v_j; \mathcal{B}). \tag{5.7}$$

Algorithm 5.1 provides the pseudocode of i-MixUp on the N-pair contrastive loss for one iteration. We present an application of i-MixUp to SimCLR in Appendix D.1.1.

Pair relations in contrastive loss. To make a sense of contrastive loss as a representation learning objective, one needs to properly define a pair relation $\{(x_i, \tilde{x}_i)\}_{i=1}^N$. For Self-SL (Chen et al., 2020a,b; He et al., 2020), where semantic class labels are not provided, the pair relation would be defined in that 1) a positive pair, x_i and $\tilde{x}_i$, are two augmented versions of the same data and 2) a negative pair, x_i and $\tilde{x}_{j\neq i}$, are simply different data instances. For supervised representation learning (Sohn, 2016; Khosla et al., 2020), x_i and $\tilde{x}_i$ are two data instances from the same class, while x_i and $\tilde{x}_{j\neq i}$ are from different classes. Note that two augmented versions of the same data also belong to the same class, so they can also be considered as a positive pair. i-MixUp is not limited to self-supervised contrastive learning, but it can also be used as a regularization method for supervised contrastive learning (Khosla et al., 2020) or deep metric learning (Sohn, 2016; Movshovitz-Attias et al., 2017; Teh et al., 2020).

5.4.2 i-MixUp for Memory-Based Contrastive Representation Learning

Unlike N-pair or SimCLR, MoCo is a memory-based contrastive learning method, such that negative samples are from the memory bank $\mathcal{M} = \{\mu_k\}_{k=1}^K$. Because embedding vectors from the memory bank are not paired with the anchors in the current batch, they are ignored in the virtual label. Instead, similarly to N-pair, for each anchor, we include the positive samples of other anchors as negatives. Let $\tilde{v}_i \in [0,1]^{N+K}$ be a virtual label indicating the positive sample of each anchor, where $\tilde{v}_{i,i} = 1$ and $\tilde{v}_{i,j} = 0$ for $j \neq i$. Then,

the extension of Eq. (5.3) is written in the form of the cross-entropy loss:

$$\ell_{\text{MoCo}}(x_i, \tilde{v}_i; \mathcal{B}, \mathcal{M}) = -\sum_{n=1}^{N} \tilde{v}_{i,n} \log \frac{\exp\left(s(f_i, \tilde{f}_n^{\text{EMA}})/\tau\right)}{\sum_{k=1}^{N} \exp\left(s(f_i, \tilde{f}_k^{\text{EMA}})/\tau\right) + \sum_{k=1}^{K} \exp\left(s(f_i, \mu_k)/\tau\right)}. \tag{5.8}$$

The application of i-MixUp to MoCo is straightforward: for two data $(x_i, \tilde{v}_i)$, $(x_j, \tilde{v}_j)$ and a batch of data pairs $\mathcal{B} = \{(x_i, \tilde{x}_i)\}_{i=1}^{N}$ and the memory bank $\mathcal{M}$, the i-MixUp loss is defined as follows:

$$\ell_{\text{MoCo}}(\lambda x_i + (1-\lambda)x_j, \lambda \tilde{v}_i + (1-\lambda)\tilde{v}_j; \mathcal{B}, \mathcal{M}). \tag{5.9}$$

Compared to the original MoCo in Eq. (5.3), Eq. (5.8) and Eq. (5.9) require clean samples to be fed to f^{EMA}. However, the additional computational cost is not significant, as they do not require gradient.

5.4.3 InputMix

Though i-MixUp is effective in contrastive representation learning, the contribution of other data augmentation methods to the quality of learned representations is also crucial. For the case when the domain knowledge about effective data augmentation methods is limited, we propose to apply InputMix together with i-MixUp, which mixes input data but not their labels. This method can be viewed as introducing structured noises driven by auxiliary data to the principal data with the largest λ, and the label of the principal data is assigned to the mixed data.

5.5 Experiments

In this section, we demonstrate the effectiveness of i-MixUp. In all experiments, we conduct self-supervised contrastive learning on the pretext dataset and evaluate the quality of learned representation via supervised classification on the downstream dataset. Specifically, in the first stage, a convolutional neural network followed by the two-layer multilayer perceptron (MLP) projection head is trained (Chen et al., 2020a). Then, the projection head is replaced with a linear classifier and only the linear classifier is trained with the labeled dataset. Unless otherwise stated, datasets for the pretext and downstream tasks are the same. For i-MixUp, we sample a single linear interpolator $\lambda \sim \beta(1, 1)$ for each training batch. Additional details for the experimental settings can be found in Appendix D.2.1.

Table 5.1: Comparison of memory-free and memory-based contrastive learning methods and i-MixUp on them with ResNet-50 on CIFAR-10 and 100. We report the mean and standard deviation of five trials with different random seeds in %. Equation number indicates the loss function of each method. i-MixUp improves the accuracy on the downstream task regardless of the data distribution shift between the pretext and downstream tasks.

Pretext	Downstream	Memory-Free Contrastive Learning		
		SimCLR (5.2)	N-pair (5.1)	i-MixUp (5.7)
CIFAR-10	CIFAR-10	92.5 $\pm$ 0.1	92.4 $\pm$ 0.1	**94.8** $\pm$ 0.2
	CIFAR-100	60.0 $\pm$ 0.2	60.2 $\pm$ 0.3	**63.3** $\pm$ 0.2
CIFAR-100	CIFAR-10	84.4 $\pm$ 0.2	84.4 $\pm$ 0.2	**86.2** $\pm$ 0.2
	CIFAR-100	68.7 $\pm$ 0.2	68.7 $\pm$ 0.2	**72.3** $\pm$ 0.2
Pretext	Downstream	Memory-Based Contrastive Learning		
		MoCo (5.3)	MoCo (5.8)	i-MixUp (5.9)
CIFAR-10	CIFAR-10	90.7 $\pm$ 0.3	90.5 $\pm$ 0.2	**92.7** $\pm$ 0.2
	CIFAR-100	60.8 $\pm$ 0.2	60.7 $\pm$ 0.5	**64.7** $\pm$ 0.4
CIFAR-100	CIFAR-10	83.1 $\pm$ 0.2	83.0 $\pm$ 0.2	**85.5** $\pm$ 0.3
	CIFAR-100	65.7 $\pm$ 0.1	66.0 $\pm$ 0.3	**70.0** $\pm$ 0.4

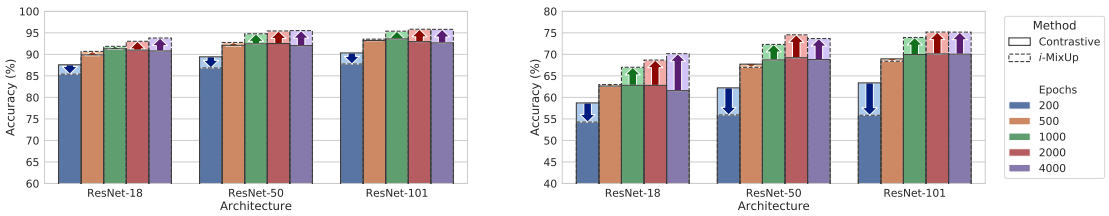
5.5.1 i-MixUp with Contrastive Learning Methods and Transferability

In this experiment, we show the applicability of i-MixUp to the state-of-the-art contrastive learning methods, and the transferability of the learned representations.

Setup. We experiment the following methods on CIFAR-10 and 100 (Krizhevsky and Hinton, 2009): 1) SimCLR (Chen et al., 2020a) and 2) N-pair (Sohn, 2016) contrastive loss in Eq. (5.1) as the memory-free method, 3) i-MixUp on N-pair contrastive learning in Eq. (5.7), 4) MoCo (Chen et al., 2020b; He et al., 2020)[3] and 5) our variant of MoCo in Eq. (5.8) for the memory-based method, and 6) i-MixUp on our variant of MoCo in Eq. (5.9). We apply a set of data augmentations randomly in sequence including resized crop (Szegedy et al., 2015), horizontal flip, color jittering, and gray scaling, as data augmentation plays an extremely crucial role in achieving high-quality self-supervised visual representations via contrastive learning (Chen et al., 2020a,b). We use ResNet-50 (He et al., 2016) as a backbone network and two-layer MLP (2048-128) as a projection head. The models are trained with a batch size of 512 (i.e., 256 pairs) for 1000 epochs.

Evaluation. The results are provided in Table 5.1. We test the classification accuracy using learned representations not only on the dataset for the pretext task, but also on the unseen

[3]We follow an improved version of MoCo in Chen et al. (2020b).



(a) CIFAR-10 (b) CIFAR-100

Figure 5.1: Comparison of performance gains by applying i-MixUp to N-pair contrastive learning with different model sizes and number of training epochs. The pretext and downstream tasks share the same dataset. While training more than 1000 epochs does not improve the performance of contrastive learning, i-MixUp benefits from longer training.

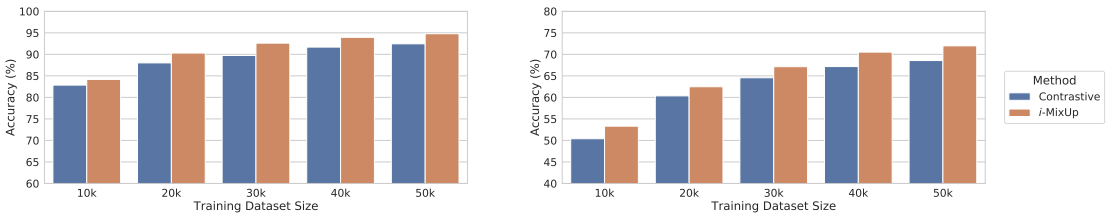
dataset at the pretext task (e.g., CIFAR-10 as pretext and CIFAR-100 as downstream tasks), to evaluate the transferability of the representation. We first verify that the N-pair formulation in Eq. (5.1) results in no worse performance than that of SimCLR in Eq. (5.2). This justifies to conduct subsequent experiments using the N-pair formulation instead of that of SimCLR, which is simpler and more efficient, especially when applying i-MixUp, while not losing the performance. When applying i-MixUp to the state-of-the-art contrastive learning methods, we observe consistent gains in the classification accuracy, e.g., 2.4%, 3.6% in N-pair, and 2.0%, 4.3% in MoCo, on CIFAR-10 and 100, respectively. Moreover, Similar performance gains are observed when learned representations from one dataset are evaluated for classification on another dataset.

5.5.2 Scaliability of i-MixUp

A better regularization method often benefits from longer training of deeper models, and may as well improve the performance when trained on a smaller dataset. To investigate the regularization effect of i-MixUp, we make a comparison between two models, N-pair contrastive loss and with i-MixUp, by training for different max epochs, model size, and training dataset sizes for the pretext task.

First, we test the efficacy of i-MixUp with respect to different model sizes and training epochs. We train ResNet-18, ResNet-50 and ResNet-101 models with the varying number of training epochs from 200 to 4000.[4] Figure 5.1 shows the performance of N-pair contrastive learning (solid box) and i-MixUp (dashed box). When models are trained for 200 epochs, i-MixUp underperforms the baseline. However, i-MixUp starts to outperform since training for 500 epochs, and improves by significant margins when trained longer. In addition, it also benefits from deeper models, achieving 95.8% on CIFAR-10 and 75.2% on

[4]Note that models with different number of epochs are trained independently, as we use the cosine learning rate schedule, such that the epoch number determines the learning rate.



(a) CIFAR-10 (b) CIFAR-100

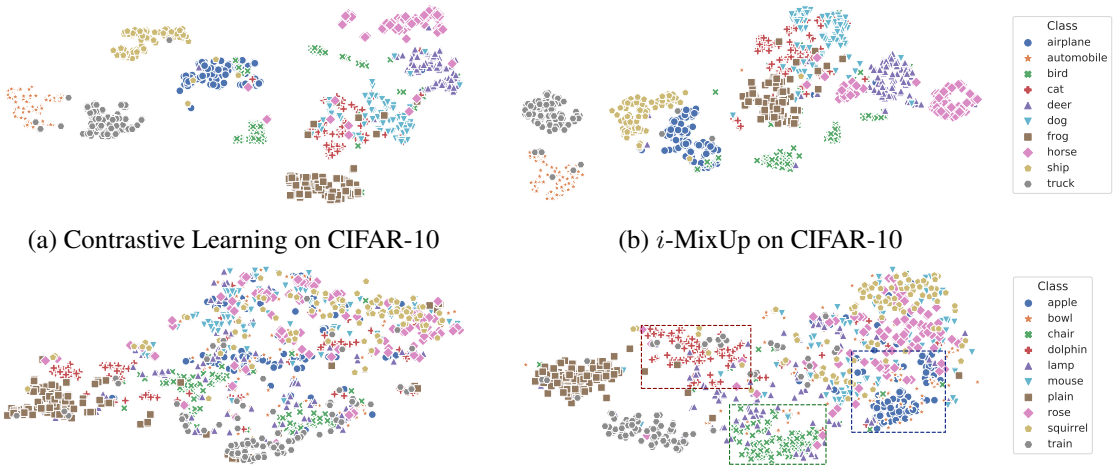
Figure 5.2: Comparison of contrastive learning and i-MixUp with ResNet-50 on CIFAR-10 and 100 with the different training dataset size. The pretext and downstream shares training dataset. The absolute performance gain by i-MixUp in contrastive learning does not decrease when the training dataset size increases.

CIFAR-100 using ResNet-101 after 4000 epochs of training, achieving state of the art. On the other hand, the models trained without i-MixUp starts to show overfitting to the pretext task when trained longer than 1000 epochs. The trend clearly shows that i-MixUp results in better representations via improved regularization.

In our second experiment, we evaluate the performance with varying dataset sizes from 20% (10k images) to 100% (50k images) of the entire pretext data. As in Figure 5.2, we observe consistent gain with i-MixUp using more data. One interesting finding is that the performance gain is larger when trained on the larger data. This may be due to the fact that, like MixUp (Zhang et al., 2018c), i-MixUp is a data-driven data augmentation whose augmentation complexity is determined by the amount of data. This is promising since unlabeled data is relatively cheap to obtain in practice. However, the effectiveness of i-MixUp over the baseline might be limited for applications where acquiring data is truly difficult.

5.5.3 Embedding Analysis

Figure 5.3 visualizes embedding spaces learned by N-pair contrastive learning and i-MixUp on CIFAR-10. When the downstream dataset is the same with the pretext task, both contrastive learning and i-MixUp cluster classes well, as shown in Figure 5.3(a) and 5.3(b). However, when the downstream task is transferred to CIFAR-100, i-MixUp in Figure 5.3(d) clusters classes better than contrastive learning in Figure 5.3(c). Specifically, clusters of "apple," "chair," and "dolphin," can be found in Figure 5.3(d) while they spread out in Figure 5.3(c). Also, "rose" and "squirrel" are more separated in Figure 5.3(d) than 5.3(c). This shows that the representation learned with i-MixUp is more generalizable than contrastive learning.



(a) Contrastive Learning on CIFAR-10

(b) i-MixUp on CIFAR-10

(c) Contrastive Learning on CIFAR-100

(d) i-MixUp on CIFAR-100

Figure 5.3: t-SNE visualization of embeddings trained by contrastive learning and i-MixUp with ResNet-50 on CIFAR-10. (a,b): Classes are well-clustered in both cases when applied to CIFAR-10. (c,d): When models are transferred to CIFAR-100, classes are more clustered for i-MixUp than contrastive learning, as highlighted in dashed boxes. We show 10 classes for a better visualization.

5.5.4 Contrastive Learning without Domain-Specific Data Augmentation

Data augmentation plays a key role in self-supervised contrastive learning, and therefore it raises a question when applying it to domains with a limited or no knowledge of such augmentations. In this section, we study the effectiveness of i-MixUp as a data-driven data augmentation for self-supervised contrastive learning.

Setup. As a domain-agnostic augmentation, we applied InputMix.[5] Specifically, for each data, we randomly choose two other auxiliary data in the batch, regardless of whether they are anchors or positive samples.[6] The linear interpolator is sampled from the Dirichlet distribution: $\lambda_1, \lambda_2, \lambda_3 \sim \text{Dir}(\alpha, \alpha, \alpha)$, where we choose $\alpha = 1$. For i-MixUp, we scale the sampled interpolators by half and add 0.5 to the principal data: i.e., $\lambda'_1 = 0.5 + 0.5\lambda_1, \lambda'_2 = 0.5\lambda_2, \lambda'_3 = 0.5\lambda_3$. However, when i-MixUp is not applied, simply assigning the largest interpolator to the principal data performs best. We also compare the denoising autoencoder (DAE) (Vincent et al., 2010) as a baseline. The autoencoder has ResNet-50 as its encoder and a three-layer transposed CNN followed by the sigmoid activation as its decoder. Similarly to models with contrastive learning, the encoder and decoder of DAE are first trained to denoise additive Gaussian noise with the standard deviation of 0.4. Then, we replace the

[5] Additive Gaussian noise and masking noise can be considered, but we found that applying InputMix performs best, and combinations of InputMix and other noises perform worse.

[6] Incorporating more or less auxiliary data does not improve the performance.

Table 5.2: Comparison of contrastive learning and i-MixUp with ResNet-50 on CIFAR-10 and 100 under different data augmentation methods: when 1) no data augmentation is available, 2) domain-agnostic data augmentation methods are applied, and 3) domain-specific data augmentation methods are applied. i-MixUp improves the performance of contrastive learning in all cases.

Pretext	Downstream	No Augmentation		Domain-Agnostic		Domain-Specific	
		Contrastive	i-MixUp	Contrastive	i-MixUp	Contrastive	i-MixUp
CIFAR-10	CIFAR-10	17.0	**49.7**	76.0	**79.7**	92.4	**94.8**
	CIFAR-100	2.8	**23.9**	41.8	**49.1**	60.2	**63.3**
CIFAR-100	CIFAR-10	18.4	**49.1**	69.6	**75.3**	84.4	**86.2**
	CIFAR-100	3.0	**23.8**	43.7	**50.7**	68.7	**72.3**

decoder with a linear classifier to evaluate the classification accuracy.

Evaluation. Table 5.2 shows the performance of compared methods with different set of data augmentation methods. First, compared to the accuracy of 92.4% when domain-specific data augmentations are applied, contrastive learning achieves only 17.0% when trained without any data augmentation. This suggests that data augmentation is an essential part for the success of self-supervised contrastive learning (e.g., SimCLR (Chen et al., 2020a)). However, i-MixUp is able to learn meaningful representations without *any* augmentation and achieves close to 50% accuracy on CIFAR-10. Next, InputMix is an effective domain-agnostic data augmentation method, such that contrastive learning outperforms DAE, where the best performance of DAE is 59.4% on CIFAR-10, and 31.7% on CIFAR-100. Also, i-MixUp further improves the performance of contrastive learning, achieving close to 80% on CIFAR-10.

5.5.5 i-MixUp on Other Domains

Finally, we apply i-MixUp on other domains beyond image, such as speech or tabular datasets, where data augmentation methods are less studied. For speech domain, we use the Speech Commands dataset (Warden, 2018), which contains 51k training, 7k validation, and 7k test data in 12 classes. [7] We apply a set of data augmentations randomly in sequence including changing amplitude, speed, and pitch in time domain, stretching, time shifting, and adding background noise in frequency domain. Data is then transformed to the mel spectogram in the size of 32×32. As a backbone network, we take VGGNet-19 (Simonyan and Zisserman, 2015) with batch normalization (Ioffe and Szegedy, 2015) and use $\tau = 0.2$. The details of experimental setting is in Appendix D.2.1.

[7] https://github.com/tugstugi/pytorch-speech-commands

Table 5.3: Comparison of contrastive learning and i-MixUp on Speech Commands (Warden, 2018) and tabular datasets from UCI machine learning repository (Asuncion and Newman, 2007). i-MixUp improves the performance of contrastive learning on those non-image domains.

Aug	Speech Commands		CovType		Higgs (100k)		Higgs (10M)	
	Contrastive	i-MixUp	Contrastive	i-MixUp	Contrastive	i-MixUp	Contrastive	i-MixUp
-	62.7	**68.1**	34.4	**69.1**	58.8	**72.6**	56.0	**74.6**
✓	77.8	**86.2**	65.2	**69.7**	73.3	**73.5**	**76.3**	75.4

Table 5.4: Comparison of MoCo v2 (Chen et al., 2020b) and i-MixUp with ResNet-50 on ImageNet.

Method	MoCo (5.3)	MoCo (5.8)	i-MixUp (5.9)
Accuracy	67.5	67.5	**68.6**

For tabular data, we use two datasets from UCI repository (Asuncion and Newman, 2007): Forest Cover Type (CovType) and Higgs Boson (Higgs). CovType contains 581k data instances for 7 forest cover type classification, where 15k are used for training and validation and the rest is used for test. Higgs contains 10M data instances for binary classification. We experiment with two protocols, where we used 100k or 10M data instances for training. 500k data instances are used for test. We train a 5-layer MLP with batch normalization. Since the prior knowledge on tabular data is very limited, only the masking noise with the probability 0.2 is considered as a data augmentation.

Table 5.3 summarizes the results. When no data augmentation is used, i-MixUp significantly improves the performance of contrastive learning. When data augmentations are used, i-MixUp shows further improvement, implying that i-MixUp has a potential as a domain-agnostic Self-SL method. However, when the number of data is too large, i-MixUp may over-regularize the network training.

5.5.6 i-MixUp on ImageNet

In this experiment, we aims to show that our proposed method can improve the state-of-the-art contrastive learning method in the large-scale setting as well. We follow the experimental settings on ImageNet (Deng et al., 2009) in MoCo v2 (Chen et al., 2020b). For the best performance, we apply the idea of CutMix (Yun et al., 2019) to i-MixUp, which is a variation of MixUp applicable in the image domain. As shown in Table 5.4, our method improves the performance of MoCo by 1.1%, which implies that our method is effective in the large-scale setting.

5.6 Summary

We propose i-MixUp, a data-driven data augmentation method for contrastive representation learning. The key idea of i-MixUp is to introduce a virtual label to each data instance, and interpolate both input and their corresponding output labels. We show that i-MixUp is applicable to the state-of-the-art memory-free and memory-based contrastive learning methods, which consistently improves the performance in variety of settings.

CHAPTER VI

Conclusion and Future Directions

Deep learning has become dominant in machine learning due to its superior performance. However, the success of deep learning methods is limited to the closed-world assumption, such that they might not be generalized to the open world. In this dissertation, we study lifelong learning and robust representation learning for a successful deployment of deep learning methods to real-world applications.

Lifelong learning. In the first part, we focus on a cycle of lifelong learning, which consists of novelty detection and continual learning. To provide a more information about novel objects, Chapter II introduces an informative novelty detection framework to detect and identify novel classes, termed hierarchical novelty detection, which essentially predicts the semantically closest class based on the hierarchy of known labels. Then, to alleviate catastrophic forgetting in the open-world setting, Chapter III proposes to leverage unlabeled data easily obtainable in the open world. To effectively leverage unlabeled data, we propose a global distillation learning objective and confidence-based sampling strategy.

Lifelong learning can be further improved in several directions. First, to provide more semantic details about novel classes, hierarchical novelty detection requires hierarchical taxonomy. In Chapter II, the hierarchical taxonomy is built based on the natural language information: specifically, the hypernym-hyponym relationships in WordNet. However, such information might not always be available in real-world applications. Therefore, an automatic and human-recognizable taxonomy construction and management would broaden the applicability of hierarchical novelty detection. Also, the performance of hierarchical novelty detection could be improved by more sophisticated methods. Next, although class-incremental learning in Chapter III has no task boundaries at test time, it assumes that tasks are sequentially given, with boundaries between them. This training strategy can be considered offline learning, as intelligent agents do not instantly update their knowledge based on their experiences, but wait until a sufficient amount of training data is collected.

However, as new tasks could emerge in the open world, seamless continual learning without task boundaries at training time is desirable. In a more challenging online learning scenario, training datasets are a stream of data, such that task boundaries are vague. Few-shot continual learning is desirable for quick adaptation to ever-changing environments. However, continual lifelong learning will fail when the model capacity is insufficient to memorize a new task together with all previously learned tasks. In that sense, an efficient continual learning model expansion over time is a promising topic.

Representation learning. In the second part, we propose methods to improve the robustness and generalization ability of deep representation learning. Focusing on domain generalization in deep reinforcement learning, Chapter IV proposes network randomization, which introduces random neural networks to make agents learn robust representations across visual changes. Also, to generalize the success of contrastive representation learning to any domain, Chapter V proposes i-MixUp, which is a domain-agnostic contrastive representation learning method. Specifically, we assign a unique virtual class to each data in a batch, and then interpolate the inputs and virtual class labels in the data and label spaces, respectively.

Our ideas can be extended to further improve the robustness of deep representation learning. First, Chapter IV focuses on learning representations robust across visual patterns, such as different styles of backgrounds, floors, and other objects. However, our preliminary results of dynamics generalization in Appendix C.10 implies that network randomization can be extended beyond visual changes. To apply network randomization to generalization problems in other domains, a clear problem statement (what to generalize) and the domain knowledge (what to randomize) are required. Next, in Chapter V, the results in Table 5.2 imply that using the same interpolator in both data and label spaces would not be optimal. Furthermore, the vicinal distribution estimated by interpolation may not follow the true manifold, such that the optimal interpolator would vary over different architectures, methods, and datasets. Therefore, learning to find the optimal interpolator would be an important step towards better contrastive representation learning.

APPENDICES

APPENDIX A

Supplementary Material of Chapter II

A.1 More on Hierarchical Novelty Detection

A.1.1 Details about Objectives

We present the exact objective functions and softmax probabilities we propose in the paper. Let $\tilde{p}(k) = \tilde{p}(y = k|x)$ be an unnormalized softmax score of the k-th class (which can be either known or novel), e.g., $\tilde{p}(k) = \exp\left(w_k^\top f(x) + b_k\right)$, where f is a visual feature extractor.

Top-down. The objective function of the top-down method is

$$\begin{aligned}\mathcal{L}_{\text{TD}}(\theta_s; \mathcal{D}) =& \frac{1}{|\mathcal{D}(s)|} \sum_{(x,y)\in\mathcal{D}(s)} \left[-\log p(y|x, s; \theta_s)\right] \\ &+ \frac{1}{|\mathcal{D}(\mathcal{O}(s))||\mathcal{C}(s)|} \sum_{x\in\mathcal{D}(\mathcal{O}(s))} \sum_{y\in\mathcal{C}(s)} \left[-\log p(y|x, s; \theta_s)\right]. \end{aligned} \tag{A.1}$$

The softmax probability used in this objective is

$$p(y|x, s; \theta_s) = \frac{\tilde{p}(y)}{\sum_{y'\in\mathcal{C}(s)} \tilde{p}(y')}.$$

Relabel. Since super classes in taxonomy have training data by data relabeling, the objective is a standard cross entropy loss over all super and leaf classes:

$$\mathcal{L}_{\text{Relabel}}(\theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \left[-\log p(y|x; \theta_{\mathcal{T}})\right]. \tag{A.2}$$

The softmax probability used in this objective is

$$p(y|x;\theta_{\mathcal{T}}) = \frac{\tilde{p}(y)}{\sum_{y'\in\mathcal{T}}\tilde{p}(y')} = \frac{\tilde{p}(y)}{\sum_{l\in\mathcal{L}(\mathcal{T})}\tilde{p}(l) + \sum_{s\in\mathcal{T}\setminus\mathcal{L}(\mathcal{T})}\tilde{p}(\mathcal{N}(s))}.$$

Here, $\mathcal{T}\setminus\mathcal{L}(\mathcal{T})$ represents all super classes in $\mathcal{T}$.

LOO. We note that there is a notation abuse in the second term of the objective function of LOO for simplicity. Without notation abuse, the exact objective is

$$\mathcal{L}_{\text{LOO}}(\theta;\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\left[-\log p(y|x;\theta_{\mathcal{L}(\mathcal{T})}) + \sum_{a\in\mathcal{A}(y)} -\log p(\mathcal{N}(\mathcal{P}(a))|x;\theta_{\mathcal{N}(\mathcal{P}(a))\cup\mathcal{L}(\mathcal{T}\setminus a)})\right]. \tag{A.3}$$

The softmax probabilities are defined as:

$$p(y|x;\theta_{\mathcal{L}(\mathcal{T})}) = \frac{\tilde{p}(y)}{\sum_{l\in\mathcal{L}(\mathcal{T})}\tilde{p}(l)},$$
$$p(\mathcal{N}(\mathcal{P}(a))|x;\theta_{\mathcal{N}(\mathcal{P}(a))\cup\mathcal{L}(\mathcal{T}\setminus a)}) = \frac{\tilde{p}(\mathcal{N}(\mathcal{P}(a)))}{\tilde{p}(\mathcal{N}(\mathcal{P}(a)) + \sum_{l\in\mathcal{L}(\mathcal{T}\setminus a)}\tilde{p}(l)}.$$

A.1.2 Hyperparameter Search

A difficulty in hierarchical novelty detection is that there are no validation data from novel classes for hyperparameter search. Similar to the training strategy, we leverage known class data for validation: specifically, for the top-down method, the novelty detection performance of each classifier is measured with $\mathcal{O}(s)$, i.e., for each classifier in a super class s, known leaf classes that do not belong to s are considered as novel classes.

$$\hat{y} = \begin{cases} \arg\max_{y'} p(y'|x,s;\theta_s) & \text{if KL}(\mathcal{U}(\cdot|s) \parallel p(\cdot|x,s;\theta_s)) \geq \lambda_s, \\ \mathcal{N}(s) & \text{otherwise,} \end{cases}$$

where λ_s is chosen to maximize the harmonic mean of the known class accuracy and the novelty detection accuracy. Note that λ_s can be tuned for each classifier.

For validating flatten methods, we discard logits of ancestors of the label of training data in a hierarchical manner. Mathematically, at the stage of removal of an ancestor $a \in \mathcal{A}(y)$, we do classification on $\theta_{\mathcal{T}\setminus a}$:

$$\hat{y} = \arg\max_{y'} p(y'|x; \theta_{\mathcal{T}\setminus a}),$$

where the ground truth is $\mathcal{N}(\mathcal{P}(a))$ at the stage. The hyperparameters with the best validation AUC are chosen.

Model-specific description. DARTS has an accuracy guarantee as a hyperparameter. We took the same candidates in the original paper, {0%, 10%, ..., 80%, 85%, 90%, 95%, 99%}, and found the best accuracy guarantee, which turned out to be 90% for ImageNet and CUB, and 99% for AwA2. Similarly, for Relabel, we evaluated relabeling rate from 5% to 95%, and found that 30%, 25%, and 15% are the best for ImageNet, AwA2, and CUB, respectively. For the top-down method and LOO, the ratio of two loss terms can be tuned, but the performance was less sensitive to the ratio, so we kept 1:1 ratio. For TD+LOO, we extracted the multiple softmax probability vectors from the top-down model and then trained the LOO.

There are some more strategies to improve the performance: the proposed losses can be computed in a class-wise manner, i.e., weighted by the number of descendant classes, which is helpful when the taxonomy is highly imbalanced, e.g., ImageNet. Also, the log of softmax and/or ReLU can be applied to the output of the top-down model. We note that stacking layers to increase model capacity improves the performance of Relabel, while it does not for LOO.

A.1.3 Experimental Results on CIFAR-100

We provide experimental results on CIFAR-100 (Krizhevsky and Hinton, 2009). The compared algorithms are the same with the other experiments, and we tune the hyperparameters following the same procedure used for the other datasets described in Section A.1.2.

Dataset. The CIFAR-100 dataset (Krizhevsky and Hinton, 2009) consists of 50k training and 10k test images. It has 20 super classes containing 5 leaf classes each, so one can naturally define the taxonomy of CIFAR-100 as the rooted tree of height two. We randomly split the classes into two known leaf classes and three novel classes at each super class, such that we have 40 known leaf classes and 60 novel classes. To build a validation set, we pick 50 images per known leaf class from the training set.

Preprocessing. CIFAR-100 images have smaller size than natural images in other datasets, so we first train a shallower network, ResNet-18 with 40 known leaf classes. Pretraining

Table A.1: Hierarchical novelty detection results on CIFAR-100. For a fair comparison, 50% of known class accuracy is guaranteed by adding a bias to all novel class scores (logits). The AUC is obtained by varying the bias.

Method	Novel	AUC
DARTS	22.38	17.84
Relabel	22.58	18.31
LOO	**23.68**	**18.93**
TD+LOO	22.79	18.54

is done with only training images, without any information about novel classes. And then, the last fully connected layer of the CNNs is replaced with our proposed methods. We use 100 training data per batch. As a regularization, L2 norm weight decay with parameter 10^{-2} is applied. The initial learning rate is 10^{-2} and it decays at most two times when loss improvement is less than 2% compared to the last epoch.

Experimental results. Table A.1 compares the baseline and our proposed methods. One can note that the proposed methods outperform the baseline in both novel class accuracy and AUC. However, unlike the results on the other datasets, TD+LOO does not outperform the vanilla LOO method, as one can expect that the vectors extracted from the top-down method might not be useful in the case of CIFAR-100 since its taxonomy is too simple and thus not informative.

A.2 Sample-Wise Qualitative Results

In this section, we show sample-wise qualitative results on ImageNet. We compared four different methods: DARTS (Deng et al., 2012) is the baseline method where we modify the method for our purpose, and the others, Relabel, LOO, and TD+LOO, are our proposed methods. In Figure A.1–A.8, we put each test image at the top, a table of the classification results in the middle, and a sub-taxonomy representing the hierarchical relationship between classes appeared in the classification results at the bottom. In tables, we provide the true label of the test image at the first row, which is either a novel class or a known leaf class. In the "Method" column in tables, "GT" is the ground truth label for hierarchical novelty detection: if the true label of the test image is a novel class, "GT" is the closest known ancestor of the novel class, which is the expected prediction; otherwise, "GT" is the true label of the test image. Each method has its own background color in both tables and sub-taxonomies. In sub-taxonomies, the novel class is shown in ellipse shape if exists, GT is double-lined, and the name of the methods is displayed below its prediction. Dashed edges represent multi-hop connection, where the number indicates the number of edges between classes: for example, a dashed edge labeled with 3 implies that two classes exist in the middle of the connection. Note that some novel classes have multiple ground truth labels if they have multiple paths to the taxonomy.

Figure A.1–A.2 show the hierarchical novelty detection results of known leaf classes, and Figure A.3–A.8 show that of novel classes. In general, while DARTS tends to produce a coarse-grained label, our proposed models try to find a fine-grained label. In most cases, our prediction is not too far from the ground truth.

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

Known class: stingray

Method	ε	A	Word
GT			stingray
DARTS	1	Y	ray
Relabel	4	N	tiger shark
LOO	2	Y	elasmobranch
TD+LOO	1	Y	ray

Known class: hen

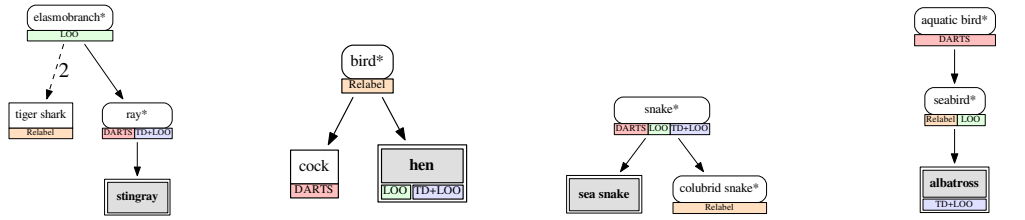
Method	ε	A	Word
GT			hen
DARTS	2	N	cock
Relabel	1	Y	bird
LOO	0	Y	hen
TD+LOO	0	Y	hen

Known class: sea snake

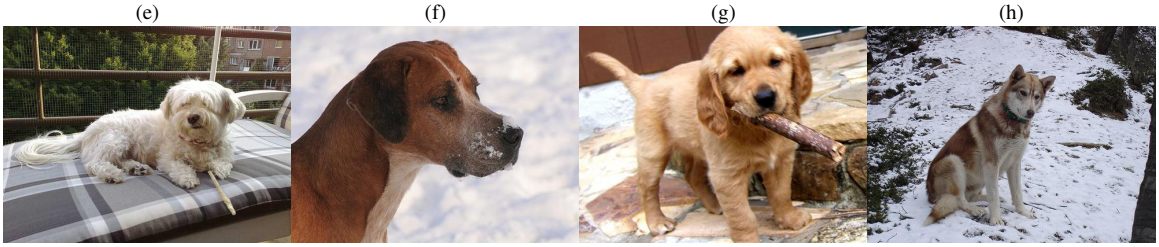
Method	ε	A	Word
GT			sea snake
DARTS	1	Y	snake
Relabel	2	N	colubrid snake
LOO	1	Y	snake
TD+LOO	1	Y	snake

Known class: albatross

Method	ε	A	Word
GT			albatross
DARTS	2	Y	aquatic bird
Relabel	1	Y	seabird
LOO	1	Y	seabird
TD+LOO	0	Y	albatross



(e) (f) (g) (h)



Known class: Maltese dog

Method	ε	A	Word
GT			Maltese dog
DARTS	5	N	Tibetan terrier
Relabel	4	N	terrier
LOO	0	Y	Maltese dog
TD+LOO	0	Y	Maltese dog

Known class: English foxhound

Method	ε	A	Word
GT			English foxhound
DARTS	4	N	Rhodesian ridgeback
Relabel	3	Y	hunting dog
LOO	1	Y	foxhound
TD+LOO	1	Y	foxhound

Known class: golden retriever

Method	ε	A	Word
GT			golden retriever
DARTS	2	Y	sporting dog
Relabel	1	Y	retriever
LOO	0	Y	golden retriever
TD+LOO	0	Y	golden retriever

Known class: Siberian husky

Method	ε	A	Word
GT			Siberian husky
DARTS	2	Y	working dog
Relabel	3	N	Eskimo dog
LOO	1	Y	sled dog
TD+LOO	1	Y	sled dog

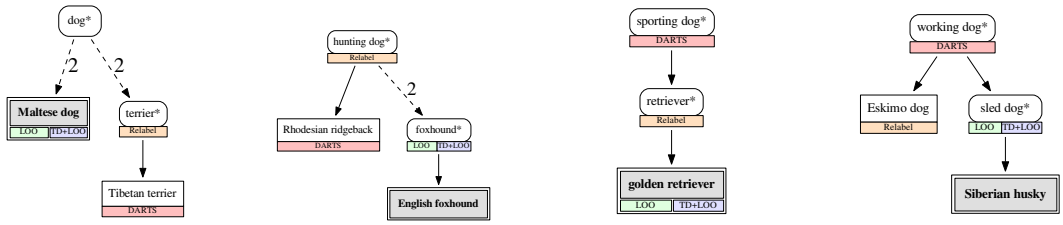
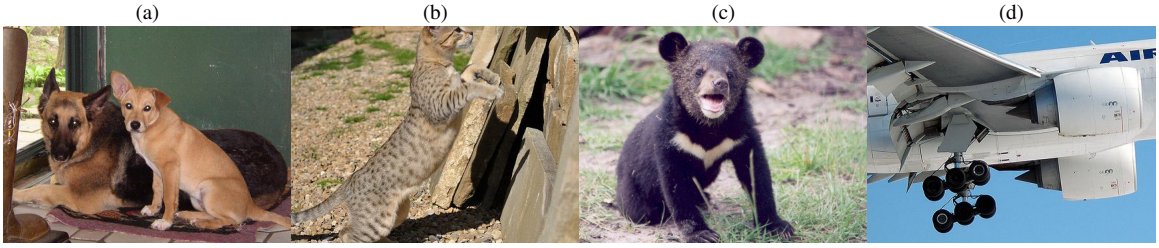
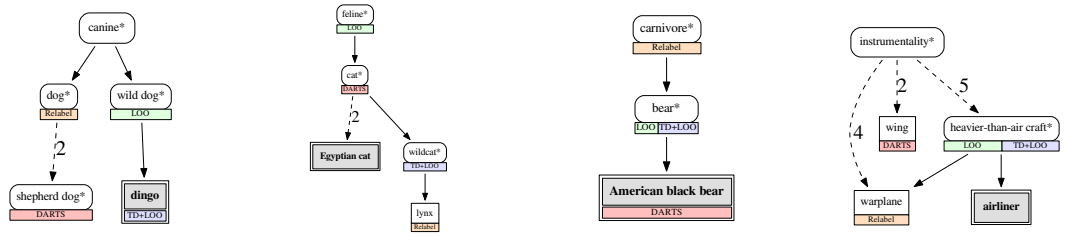


Figure A.1: Qualitative results of hierarchical novelty detection on ImageNet.

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



Known class: dingo				Known class: Egyptian cat				Known class: American black bear				Known class: airliner			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			dingo	GT			Egyptian cat	GT			American black bear	GT			airliner
DARTS	5	N	shepherd dog	DARTS	2	Y	cat	DARTS	0	Y	American black bear	DARTS	8	N	wing
Relabel	3	N	dog	Relabel	4	N	lynx	Relabel	2	Y	carnivore	Relabel	2	N	warplane
LOO	1	Y	wild dog	LOO	3	Y	feline	LOO	1	Y	bear	LOO	1	Y	heavier-than-air craft
TD+LOO	0	Y	dingo	TD+LOO	3	N	wildcat	TD+LOO	1	Y	bear	TD+LOO	1	Y	heavier-than-air craft



(e) (f) (g) (h)

Known class: digital clock				Known class: pitcher				Known class: soup bowl				Known class: toaster			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			digital clock	GT			pitcher	GT			soup bowl	GT			toaster
DARTS	3	N	digital watch	DARTS	7	N	drum	DARTS	1	Y	bowl	DARTS	9	N	furniture
Relabel	3	Y	measuring instrument	Relabel	1	Y	vessel	Relabel	1	Y	bowl	Relabel	7	N	instrumentality
LOO	2	Y	timepiece	LOO	6	N	percussion instrument	LOO	11	N	punch	LOO	1	Y	kitchen appliance
TD+LOO	0	Y	digital clock	TD+LOO	0	Y	pitcher	TD+LOO	11	N	punch	TD+LOO	0	Y	toaster

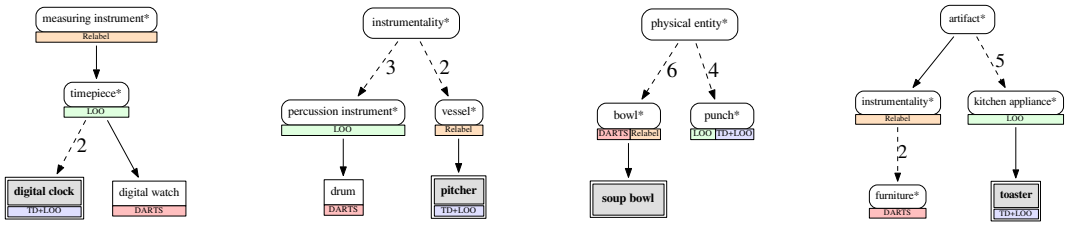
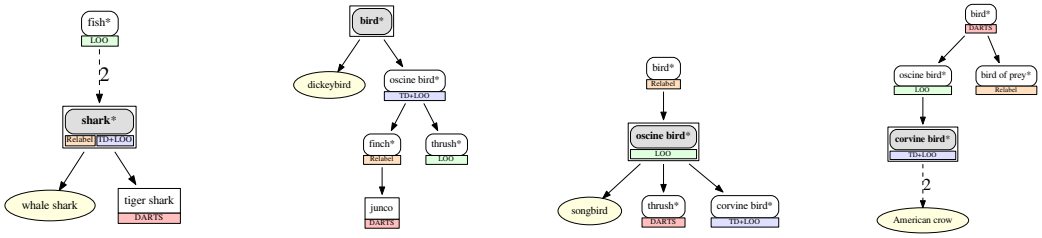


Figure A.2: Qualitative results of hierarchical novelty detection on ImageNet.

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

Novel class: whale shark				Novel class: dickeybird				Novel class: songbird				Novel class: American crow			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			shark	GT			bird	GT			oscine bird	GT			corvine bird
DARTS	1	N	tiger shark	DARTS	3	N	junco	DARTS	1	N	thrush	DARTS	2	Y	bird
Relabel	0	Y	shark	Relabel	2	N	finch	Relabel	1	Y	bird	Relabel	3	N	bird of prey
LOO	2	Y	fish	LOO	2	N	thrush	LOO	0	Y	oscine bird	LOO	1	Y	oscine bird
TD+LOO	0	Y	shark	TD+LOO	1	N	oscine bird	TD+LOO	1	N	corvine bird	TD+LOO	0	Y	corvine bird



(e) (f) (g) (h)

Novel class: raven				Novel class: swallow				Novel class: sheldrake				Novel class: scoter			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			corvine bird	GT			oscine bird	GT			duck	GT			duck
DARTS	0	Y	corvine bird	DARTS	0	Y	oscine bird	DARTS	4	N	American coot	DARTS	4	N	American coot
Relabel	2	Y	bird	Relabel	1	Y	bird	Relabel	2	Y	aquatic bird	Relabel	2	Y	aquatic bird
LOO	1	Y	oscine bird	LOO	1	N	finch	LOO	1	Y	anseriform bird	LOO	1	Y	anseriform bird
TD+LOO	2	N	thrush	TD+LOO	3	N	kite	TD+LOO	0	Y	duck	TD+LOO	0	Y	duck

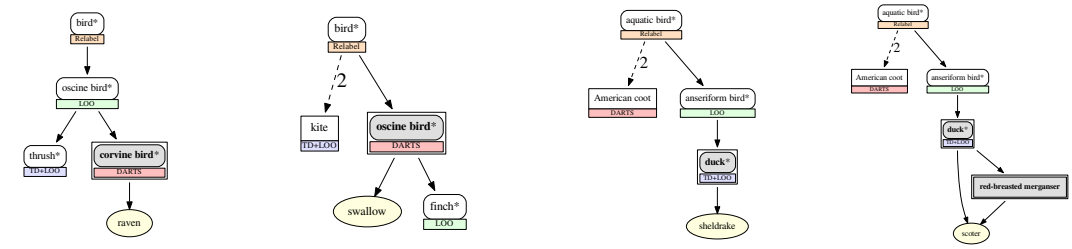
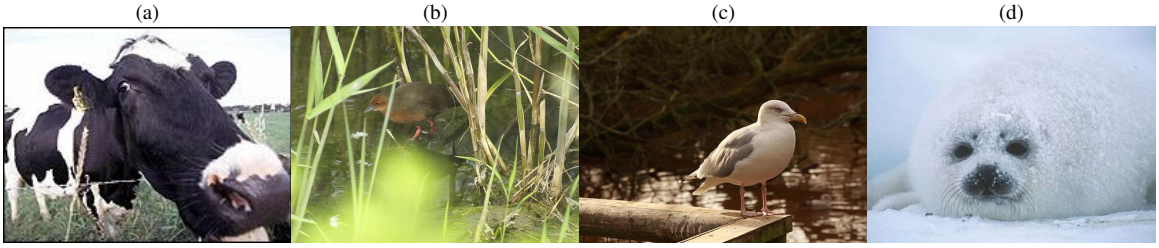
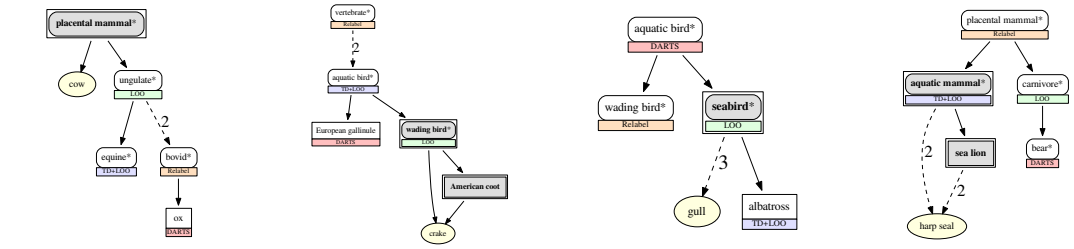


Figure A.3: Qualitative results of hierarchical novelty detection on ImageNet.

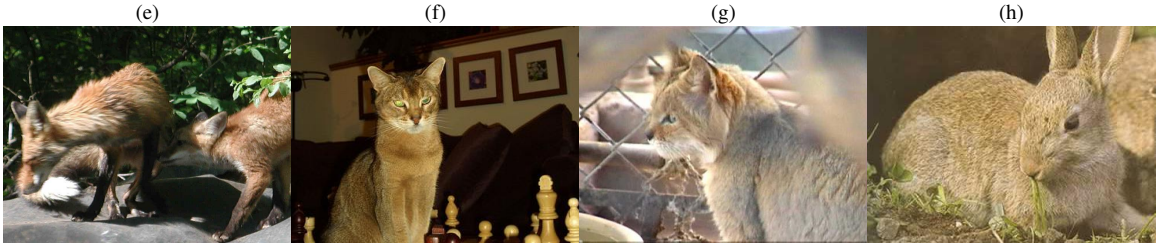
(a) (b) (c) (d)



Novel class: cow				Novel class: crake				Novel class: gull				Novel class: harp seal			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			placental mammal	GT			wading bird	GT			seabird	GT			aquatic mammal
DARTS	4	N	ox	DARTS	2	N	European gallinule	DARTS	1	Y	aquatic bird	DARTS	3	N	bear
Relabel	3	N	bovid	Relabel	3	Y	vertebrate	Relabel	2	N	wading bird	Relabel	1	Y	placental mammal
LOO	1	N	ungulate	LOO	0	Y	wading bird	LOO	0	Y	seabird	LOO	2	N	carnivore
TD+LOO	2	N	equine	TD+LOO	1	Y	aquatic bird	TD+LOO	1	N	albatross	TD+LOO	0	Y	aquatic mammal



(e) (f) (g) (h)



Novel class: red fox, Vulpes fulva				Novel class: Abyssinian cat				Novel class: sand cat				Novel class: European rabbit			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			fox	GT			domestic cat	GT			wildcat	GT			rabbit
DARTS	1	N	red fox, Vulpes vulpes	DARTS	1	N	Egyptian cat	DARTS	2	Y	feline	DARTS	1	Y	leporid mammal
Relabel	1	Y	canine	Relabel	0	Y	domestic cat	Relabel	2	N	domestic cat	Relabel	1	N	wood rabbit
LOO	0	Y	fox	LOO	1	Y	cat	LOO	1	Y	cat	LOO	0	Y	rabbit
TD+LOO	0	Y	fox	TD+LOO	0	Y	domestic cat	TD+LOO	0	Y	wildcat	TD+LOO	0	Y	rabbit

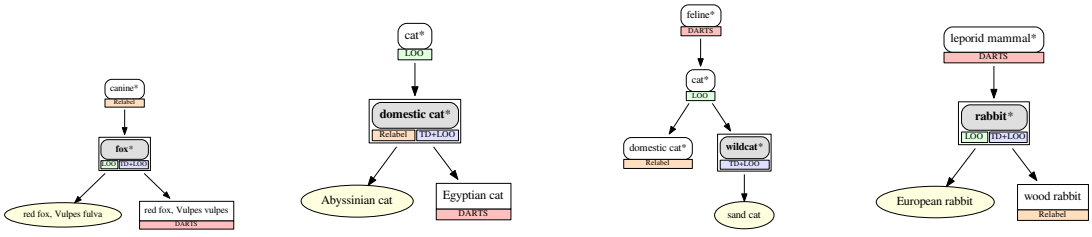
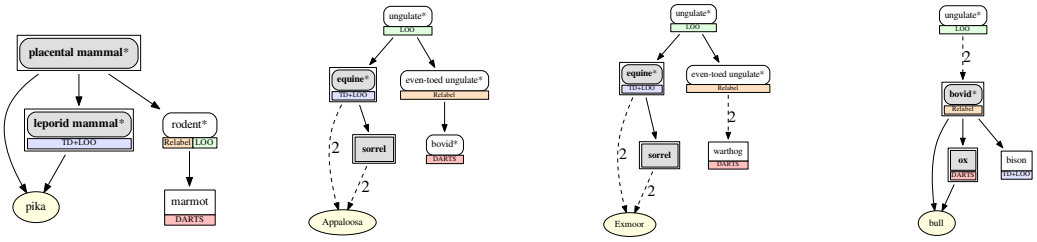


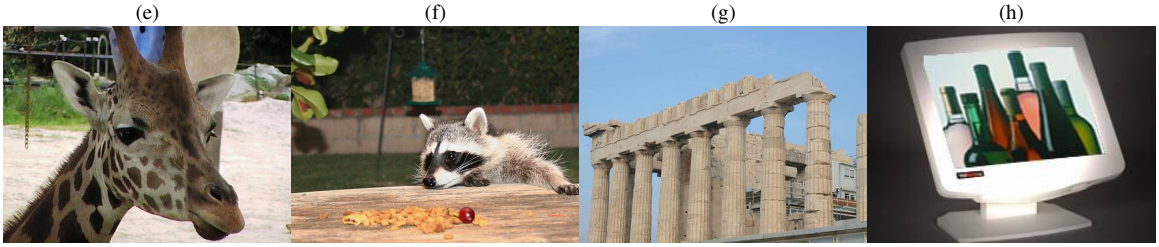
Figure A.4: Qualitative results of hierarchical novelty detection on ImageNet.

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

Novel class: pika				Novel class: Appaloosa				Novel class: Exmoor				Novel class: bull			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			placental mammal	GT			equine	GT			equine	GT			bovid
DARTS	2	N	marmot	DARTS	3	N	bovid	DARTS	4	N	warthog	DARTS	0	Y	ox
Relabel	1	N	rodent	Relabel	2	N	even-toed ungulate	Relabel	2	N	even-toed ungulate	Relabel	0	Y	bovid
LOO	1	N	rodent	LOO	1	Y	ungulate	LOO	1	Y	ungulate	LOO	2	Y	ungulate
TD+LOO	0	Y	leporid mammal	TD+LOO	0	Y	equine	TD+LOO	0	Y	equine	TD+LOO	1	N	bison



(e) (f) (g) (h)



Novel class: giraffe				Novel class: raccoon				Novel class: acropolis				Novel class: active matrix screen			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			even-toed ungulate	GT			procyonid	GT			castle	GT			electronic device
DARTS	1	N	antelope	DARTS	2	N	musteline mammal	DARTS	2	N	dam	DARTS	4	N	personal computer
Relabel	0	Y	even-toed ungulate	Relabel	1	Y	carnivore	Relabel	0	Y	structure, construction	Relabel	2	Y	instrumentality
LOO	1	Y	ungulate	LOO	1	Y	carnivore	LOO	2	N	residence	LOO	5	N	peripheral
TD+LOO	2	N	equine	TD+LOO	0	Y	procyonid	TD+LOO	1	N	triumphal arch	TD+LOO	0	Y	electronic device

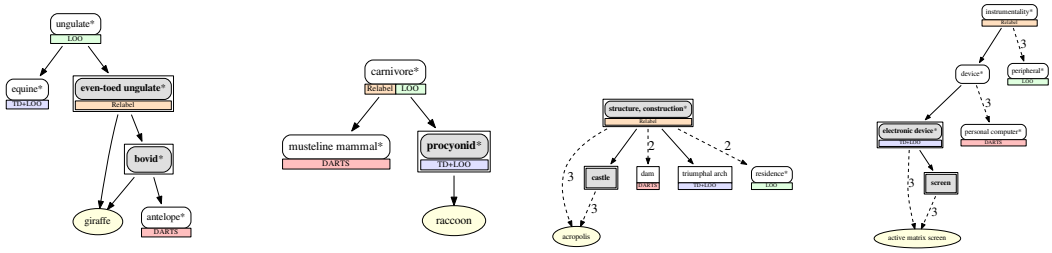
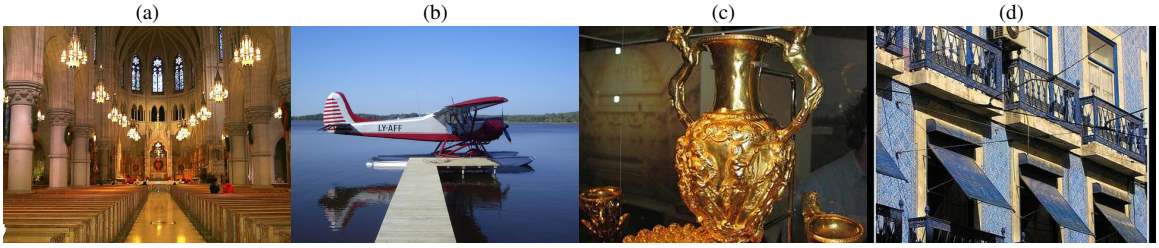
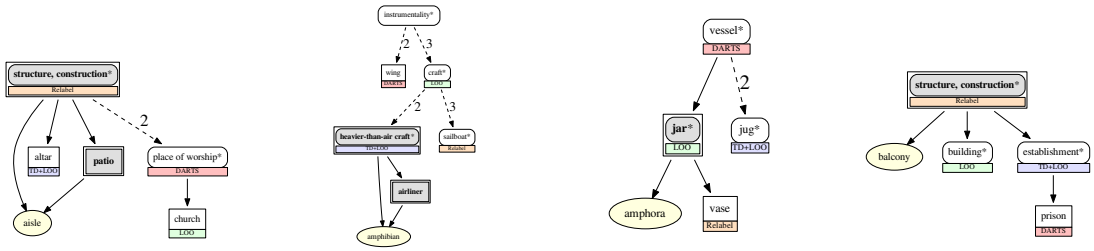


Figure A.5: Qualitative results of hierarchical novelty detection on ImageNet.

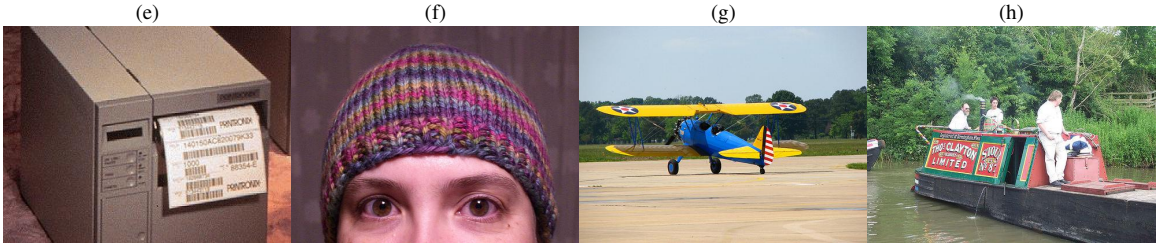
(a) (b) (c) (d)



Novel class: aisle				Novel class: amphibian				Novel class: amphora				Novel class: balcony			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			patio	GT			airliner	GT			jar	GT			structure, construction
DARTS	2	N	place of worship	DARTS	7	N	wing	DARTS	1	Y	vessel	DARTS	2	N	prison
Relabel	0	Y	structure, construction	Relabel	5	N	sailboat	Relabel	1	N	vase	Relabel	0	Y	structure, construction
LOO	3	N	church	LOO	2	Y	craft	LOO	0	Y	jar	LOO	1	N	building
TD+LOO	1	N	altar	TD+LOO	0	Y	heavier-than-air craft	TD+LOO	3	N	jug	TD+LOO	1	N	establishment



(e) (f) (g) (h)



Novel class: bar printer				Novel class: beanie				Novel class: biplane				Novel class: canal boat			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			machine	GT			cap	GT			airliner	GT			boat
DARTS	1	Y	peripheral	DARTS	6	N	wool	DARTS	7	N	wing	DARTS	3	Y	vehicle
Relabel	2	Y	electronic equipment	Relabel	2	N	hat	Relabel	7	N	parachute	Relabel	7	N	structure, construction
LOO	0	Y	machine	LOO	5	N	mask	LOO	1	Y	aircraft	LOO	9	N	shed
TD+LOO	0	Y	printer	TD+LOO	6	N	ski mask	TD+LOO	0	Y	heavier-than-air craft	TD+LOO	0	Y	boat

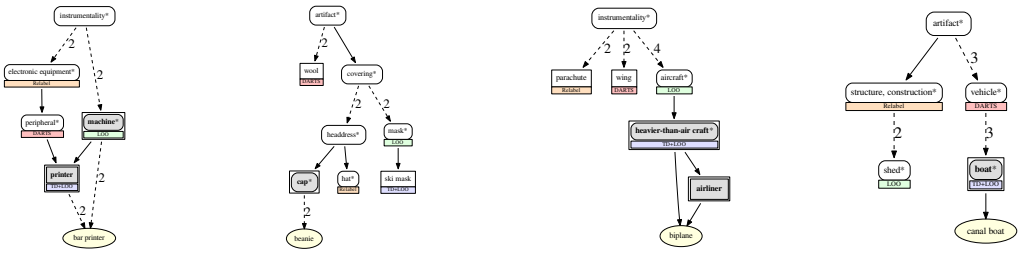
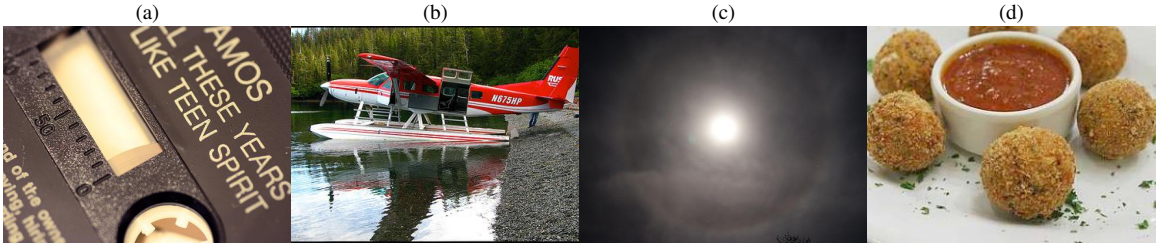
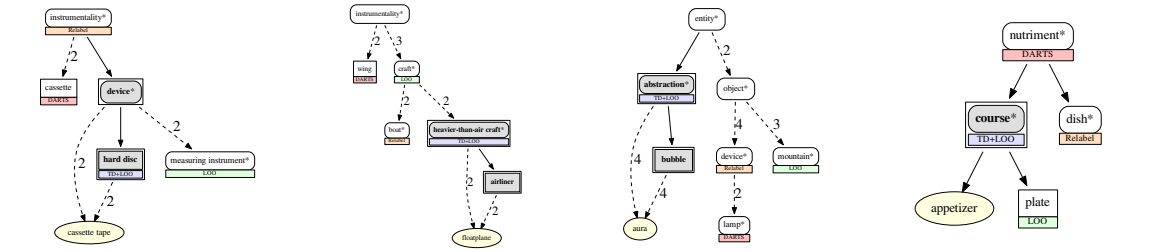


Figure A.6: Qualitative results of hierarchical novelty detection on ImageNet.

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



Novel class: cassette tape				Novel class: floatplane				Novel class: aura				Novel class: appetizer			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			device	GT			airliner	GT			abstraction	GT			course
DARTS	3	N	cassette	DARTS	7	N	wing	DARTS	9	N	lamp	DARTS	1	Y	nutriment
Relabel	1	Y	instrumentality	Relabel	4	N	boat	Relabel	7	N	device	Relabel	2	N	dish
LOO	2	N	measuring instrument	LOO	2	Y	craft	LOO	6	N	mountain	LOO	1	N	plate
TD+LOO	0	Y	hard disc	TD+LOO	0	Y	heavier-than-air craft	TD+LOO	0	Y	abstraction	TD+LOO	0	Y	course



(e) (f) (g) (h)

Novel class: hors d'oeuvre				Novel class: BLT sandwich				Novel class: kale				Novel class: cranberry			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			course	GT			sandwich	GT			cruciferous vegetable	GT			edible fruit
DARTS	1	N	plate	DARTS	2	Y	nutriment	DARTS	0	Y	cruciferous vegetable	DARTS	0	Y	fruit
Relabel	2	N	dish	Relabel	1	N	cheeseburger	Relabel	1	Y	vegetable	Relabel	0	Y	edible fruit
LOO	1	Y	nutriment	LOO	0	Y	sandwich	LOO	1	Y	vegetable	LOO	1	N	pomegranate
TD+LOO	0	Y	course	TD+LOO	0	Y	sandwich	TD+LOO	0	Y	head cabbage	TD+LOO	0	Y	strawberry

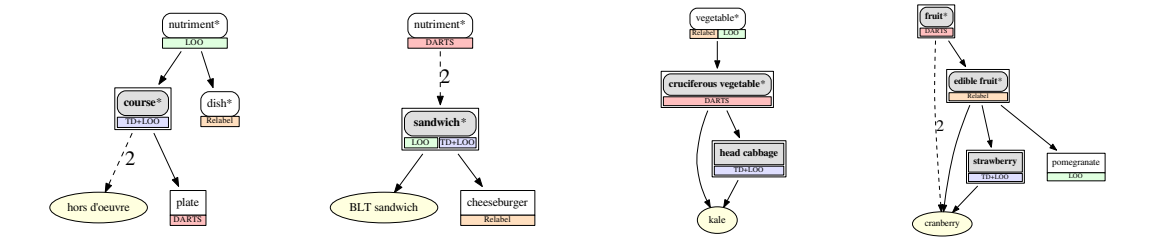
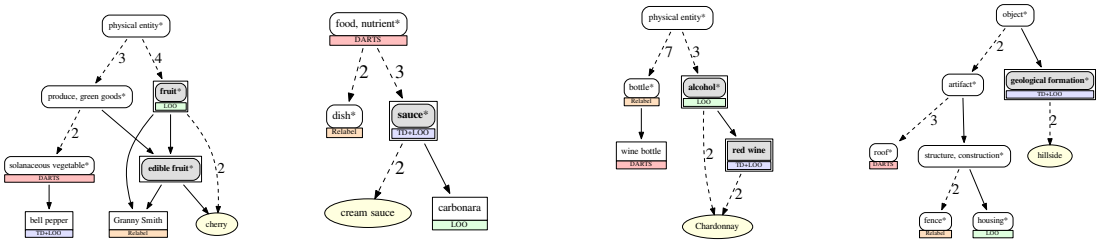


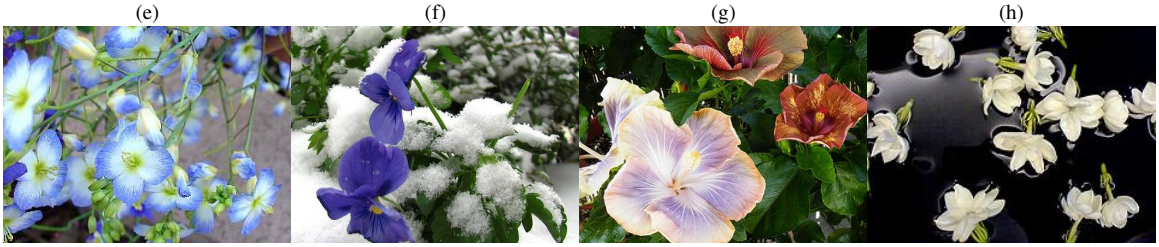
Figure A.7: Qualitative results of hierarchical novelty detection on ImageNet.

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

Novel class: cherry				Novel class: cream sauce				Novel class: Chardonnay				Novel class: hillside			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			edible fruit	GT			sauce	GT			alcohol	GT			geological formation
DARTS	3	N	solanaceous vegetable	DARTS	3	Y	food, nutrient	DARTS	11	N	wine bottle	DARTS	6	N	roof
Relabel	1	N	Granny Smith	Relabel	5	N	dish	Relabel	10	N	bottle	Relabel	6	N	fence
LOO	0	Y	fruit	LOO	1	N	carbonara	LOO	0	Y	alcohol	LOO	5	N	housing
TD+LOO	4	N	bell pepper	TD+LOO	0	Y	sauce	TD+LOO	0	Y	red wine	TD+LOO	0	Y	geological formation



(e) (f) (g) (h)



Novel class: heliophila				Novel class: tangle orchid				Novel class: rose mallow				Novel class: jasmine			
Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word	Method	ε	A	Word
GT			flower	GT			flower	GT			organism, being	GT			organism, being
DARTS	3	N	earthstar	DARTS	6	N	pot, flowerpot	DARTS	5	N	pot, flowerpot	DARTS	6	N	jar
Relabel	8	N	vegetable	Relabel	1	N	daisy	Relabel	7	N	vegetable	Relabel	1	N	daisy
LOO	1	Y	organism, being	LOO	8	N	vegetable	LOO	0	Y	organism, being	LOO	0	Y	organism, being
TD+LOO	0	Y	flower	TD+LOO	0	Y	flower	TD+LOO	0	Y	flower	TD+LOO	0	Y	flower

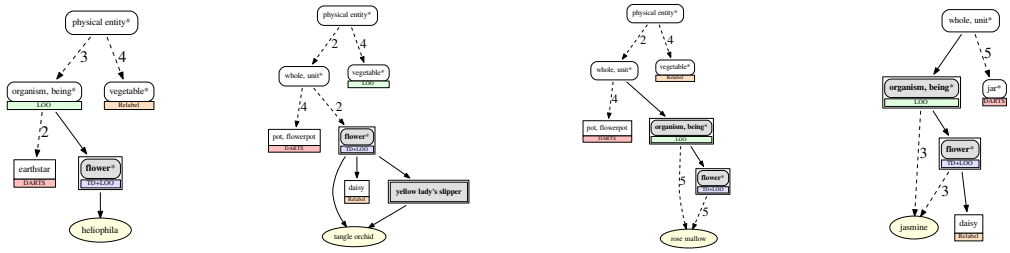


Figure A.8: Qualitative results of hierarchical novelty detection on ImageNet.

A.3 Class-Wise Qualitative Results

In this section, we show class-wise qualitative results on ImageNet. We compared four different methods: DARTS (Deng et al., 2012) is the baseline method where we modify the method for our purpose, and the others, Relabel, LOO, and TD+LOO, are our proposed methods. In a sub-taxonomy, for each test class and method, we show the statistics of the hierarchical novelty detection results of known leaf classes in Figure A.9–A.10, and that of novel classes in Figure A.11–A.14. Each sub-taxonomy is simplified by only showing test classes predicted with a probability greater than 0.03 in at least one method and their common ancestors. The probability is represented in colored nodes as well as the number below the English word of the class, where the color scale is displayed below. Note that the summation of the probabilities shown in each sub-taxonomy may be less than 1, since some classes with a probability less than 0.03 are omitted. In the graphs, known leaf classes are in rectangle, and super classes are rounded and starred. If the prediction is on a super class, then the test image is classified as a novel class whose closest class in the taxonomy is the super class. We remark that most of the incorrect prediction is in fact not very far from the ground truth, which means that the prediction still provides useful information. While our proposed methods tend to find fine-grained classes, DARTS gives more coarse-grained classes, where one can find the trend clearly in deep sub-taxonomies. Also, Relabel sometimes fails to predict the correct label but closer ones with a high probability which can be seen as the effect of relabeling.

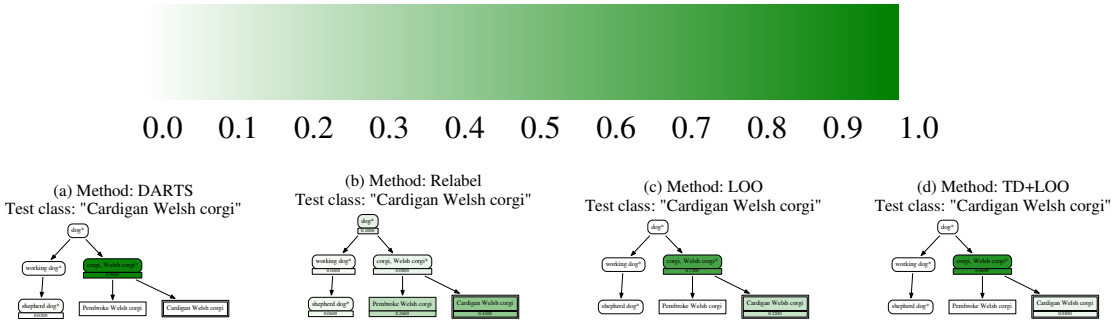


Figure A.9: Sub-taxonomies of the hierarchical novelty detection results of a known leaf class "Cardigan Welsh corgi." (Best viewed when zoomed in on a screen.)

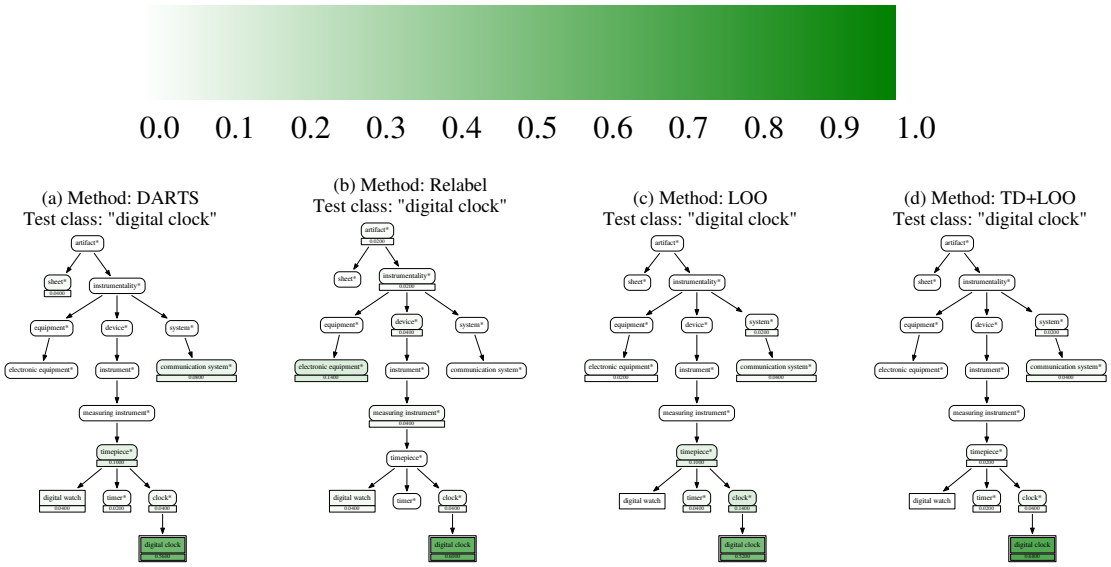


Figure A.10: Sub-taxonomies of the hierarchical novelty detection results of a known leaf class "digital clock." (Best viewed when zoomed in on a screen.)

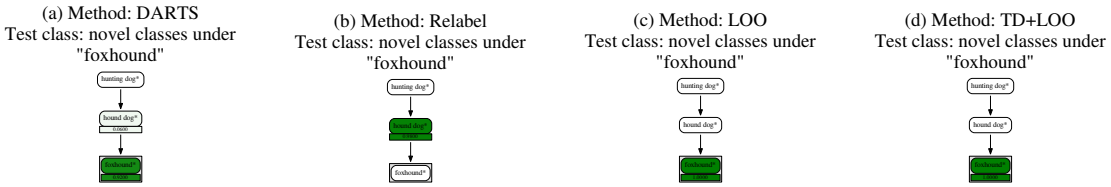


Figure A.11: Sub-taxonomies of the hierarchical novelty detection results of novel classes whose closest class in the taxonomy is "foxhound." (Best viewed when zoomed in on a screen.)

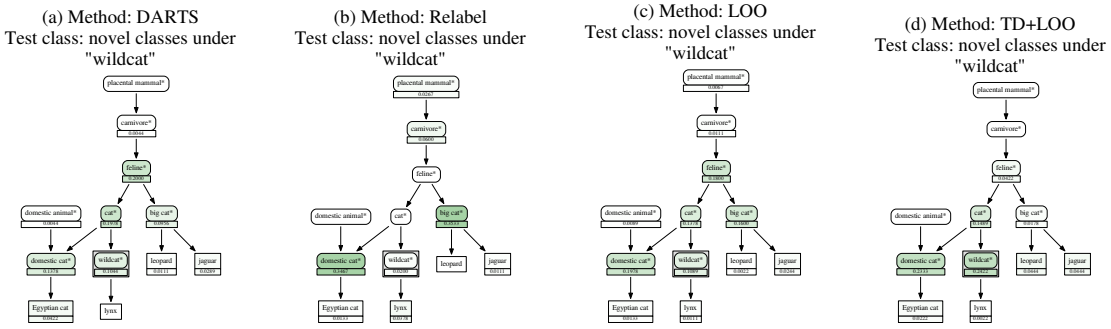


Figure A.12: Sub-taxonomies of the hierarchical novelty detection results of novel classes whose closest class in the taxonomy is "wildcat." (Best viewed when zoomed in on a screen.)

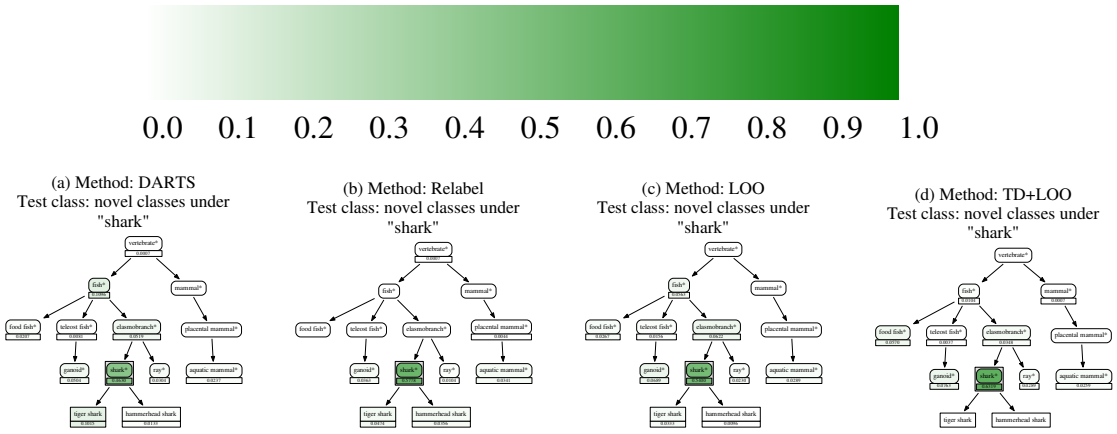


Figure A.13: Sub-taxonomies of the hierarchical novelty detection results of novel classes whose closest class in the taxonomy is "shark." (Best viewed when zoomed in on a screen.)

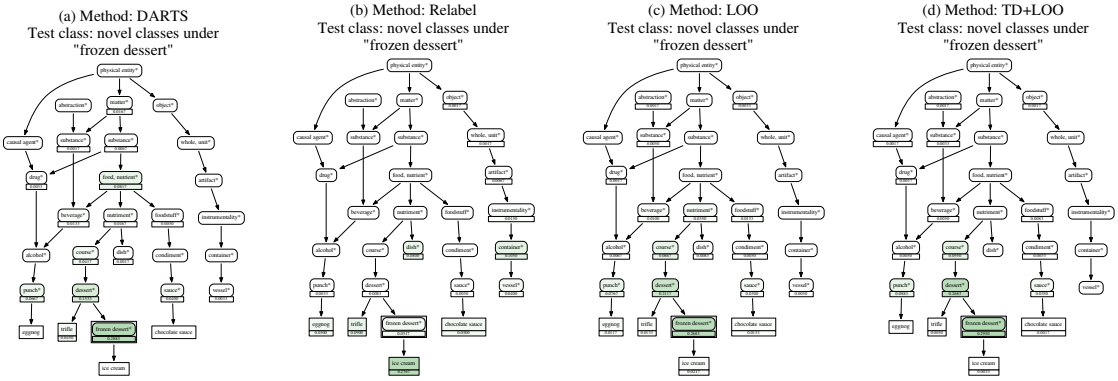
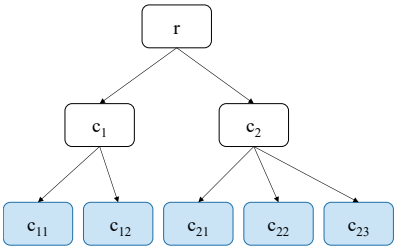


Figure A.14: Sub-taxonomies of the hierarchical novelty detection results of novel classes whose closest class in the taxonomy is "frozen dessert." (Best viewed when zoomed in on a screen.)

A.4 More on Generalized Zero-Shot Learning

A.4.1 Example of Top-Down Embedding

Figure A.15 illustrates an example of the ideal output probability vector t^y in a simple taxonomy, where t^y corresponds to the concatenation of the ideal output of the top-down method when the input label is y.



$$
\begin{array}{l}
t^y = [\ t^{(y,r)}\ ,\ t^{(y,c_1)}\ ,\ t^{(y,c_2)}\ ] \\
\hline
t^r = [1/2,\ 1/2,\ 1/2,\ 1/2,\ 1/3,\ 1/3,\ 1/3] \\
t^{c_1} = [\ 1\ ,\ 0\ ,\ 1/2,\ 1/2,\ 1/3,\ 1/3,\ 1/3] \\
t^{c_2} = [\ 0\ ,\ 1\ ,\ 1/2,\ 1/2,\ 1/3,\ 1/3,\ 1/3] \\
t^{c_{11}} = [\ 1\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 1/3,\ 1/3,\ 1/3] \\
t^{c_{12}} = [\ 1\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ 1/3,\ 1/3,\ 1/3] \\
t^{c_{21}} = [\ 0\ ,\ 1\ ,\ 1/2,\ 1/2,\ 1\ ,\ 0\ ,\ 0\ ] \\
t^{c_{22}} = [\ 0\ ,\ 1\ ,\ 1/2,\ 1/2,\ 0\ ,\ 1\ ,\ 0\ ] \\
t^{c_{23}} = [\ 0\ ,\ 1\ ,\ 1/2,\ 1/2,\ 0\ ,\ 0\ ,\ 1\ ]
\end{array}
$$

Figure A.15: An example of taxonomy and the corresponding t^y values.

A.4.2 Evaluation: Generalized Zero-Shot Learning on Different Data Splits

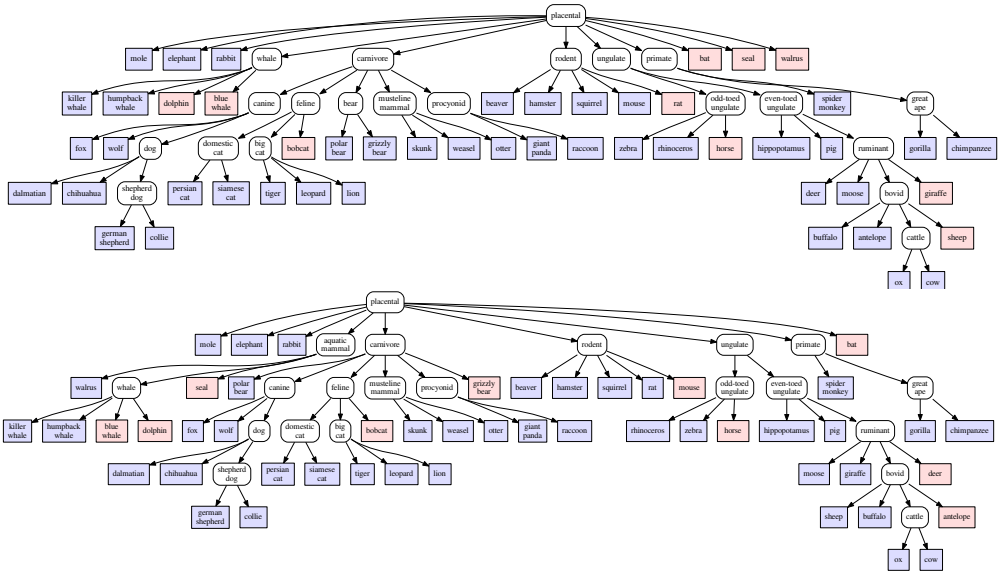


Figure A.16: Taxonomy of AwA built with the split proposed in (Xian et al., 2017) (top) and the split we propose for balanced taxonomy (bottom). Taxonomy is built with known leaf classes (blue) by finding their super classes (white), and then novel classes (red) are attached for visualization.

Table A.2: ZSL and GZSL performance of semantic embedding models and their combinations on AwA1 and AwA2 in the split of imbalanced taxonomy and that of balanced taxonomy. "Att" stands for continuous attributes labeled by human, "Word" stands for word embedding trained with the GloVe objective (Pennington et al., 2014), and "Hier" stands for the hierarchical embedding, where "Path" is proposed in (Akata et al., 2015), and "TD" is output of the proposed top-down method. "Unseen" is the accuracy when only unseen classes are tested, and "AUC" is the area under the seen-unseen curve where the unseen class score bias is varied for computation. The curve used to obtain AUC is shown in Figure A.17. Values in bold indicate the best performance among the combined models.

Embedding			AwA1				AwA2			
			Imbalanced		Balanced		Imbalanced		Balanced	
Att	Word	Hier	Unseen	AUC	Unseen	AUC	Unseen	AUC	Unseen	AUC
✓	-	-	65.29	50.02	65.86	54.18	63.87	51.27	71.21	59.51
-	✓	-	51.87	39.67	54.29	42.40	54.77	42.21	59.60	46.83
✓	✓	-	67.80	52.84	**67.32**	55.40	65.76	53.18	72.89	60.60
-	-	Path	42.57	30.58	53.40	41.63	44.34	33.44	60.45	48.13
✓	-	Path	67.09	51.45	65.86	54.18	66.58	53.50	71.87	60.08
-	✓	Path	52.89	40.66	58.49	45.62	55.28	42.86	66.83	53.05
✓	✓	Path	68.04	53.21	**67.32**	55.40	67.28	54.31	73.04	60.89
-	-	TD	33.86	25.56	40.38	31.39	31.84	24.97	45.33	36.76
✓	-	TD	66.13	54.66	65.86	54.18	66.86	57.49	72.75	62.79
-	✓	TD	56.14	46.28	57.88	47.63	59.67	49.39	65.29	53.40
✓	✓	TD	**69.23**	**57.67**	66.41	**55.84**	**68.80**	**59.24**	**75.09**	**64.36**

We present the quantitative results on a different split of AwA1 and AwA2 in this section. We note that the seen-unseen split of AwA proposed in (Xian et al., 2017) has an imbalanced taxonomy as shown in the top of Figure A.16. Specifically, three classes belong to the root class, and another two classes belong to the same super class. To show the importance of balanced taxonomy, we make another seen-unseen split for balancing taxonomy, while unseen classes are ensured not to be used for training the CNN feature extractor. The taxonomy of new split is shown in the bottom of Figure A.16.

Table A.2 shows the performance of the attribute and word embedding models, and two different hierarchical embedding models, i.e., Path and TD, and their combinations on AwA1 and AwA2 with the split of the imbalanced taxonomy and that of the balanced taxonomy. Compared to the imbalanced taxonomy case, in the balanced taxonomy, the standalone performance of hierarchical embeddings has similar tendency, but the overall performance is better in all cases. However, in the combined model, while Path does not improve the performance much, TD still shows improvement on both ZSL and GZSL tasks. Note that the combination with TD has lower ZSL performance than the combination with-

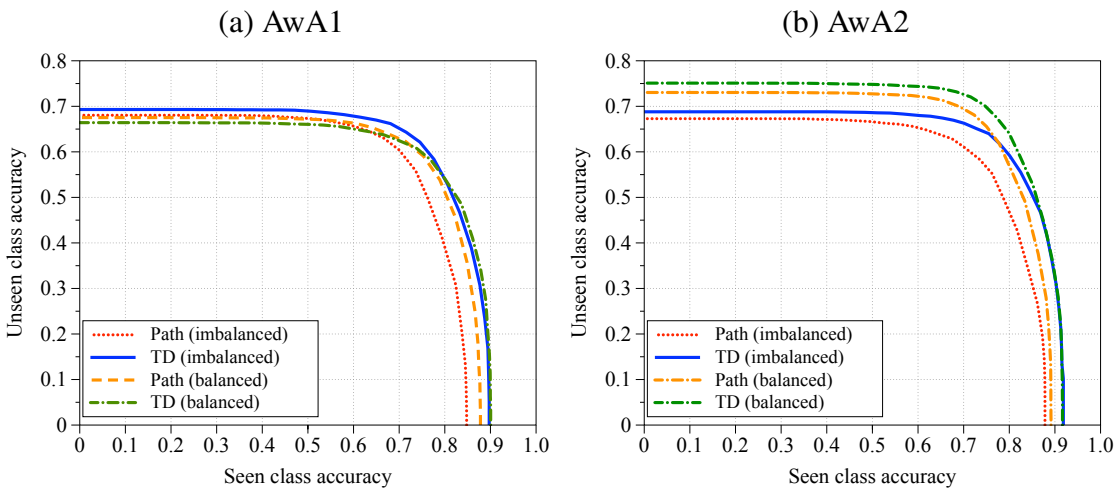


Figure A.17: Seen-unseen class accuracy curves of the best combined models obtained by varying the unseen class score bias on AwA1 and AwA2, with the split of imbalanced taxonomy and that of balanced taxonomy. "Path" is the hierarchical embedding proposed in (Akata et al., 2015), and "TD" is the embedding of the multiple softmax probability vector obtained from the proposed top-down method. We remark that if the dataset has a balanced taxonomy, the overall performance can be improved.

out TD in some cases, because only AUC is the criterion for optimization. Compared to the best single semantic embedding model (with attributes), the combination with TD leads to absolute improvement of AUC by 1.66% and 4.85% in the split we propose for balanced taxonomy on AwA1 and AwA2, respectively.

These results imply that with more balanced taxonomy, the hierarchy of labels can be implicitly learned without a hierarchical embedding such that the performance is generally better, but yet the combination of an explicit hierarchical embedding improves the performance.

APPENDIX B

Supplementary Material of Chapter III

B.1 Illustration of Global Distillation

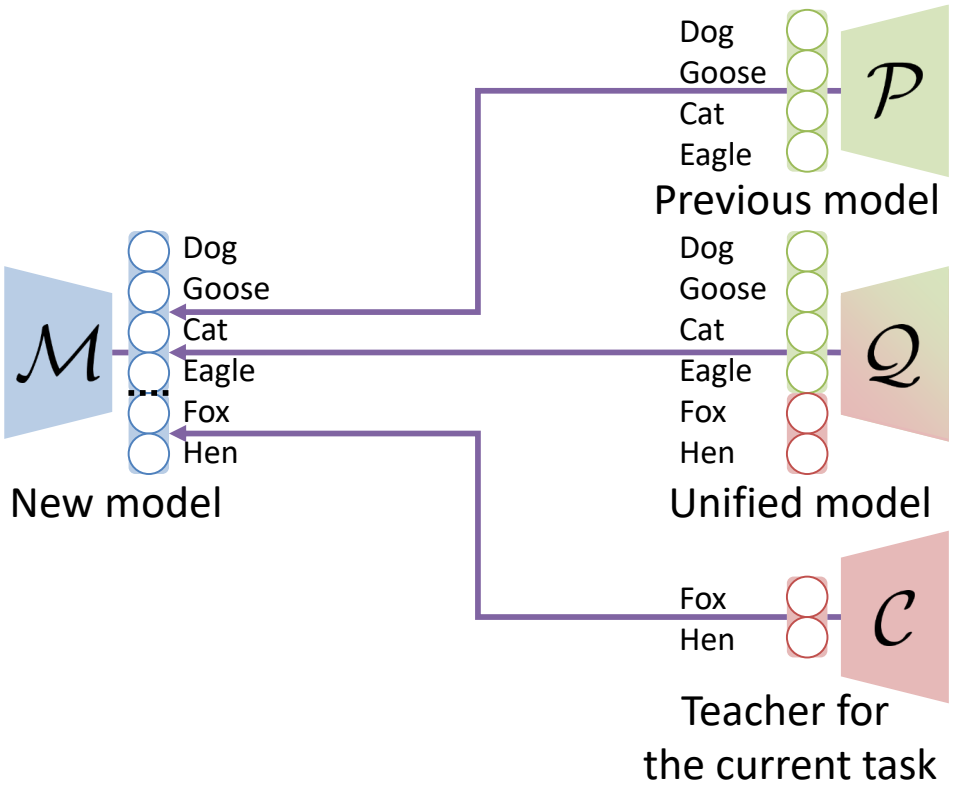


Figure B.1: An illustration of how a model $\mathcal{M}$ learns with global distillation (GD). For GD, three reference models are used: $\mathcal{P}$ is the previous model, $\mathcal{C}$ is the teacher for the current task, and $\mathcal{Q}$ is an ensemble of them.

B.2 Details on Experimental Setup

Hyperparameters. We use mini-batch training with a batch size of 128 over 200 epochs for each training to ensure convergence. The initial learning rate is 0.1 and decays by 0.1 after 120, 160, 180 epochs when there is no fine-tuning. When fine-tuning is applied, the model is first trained over 180 epochs where the learning rate decays after 120, 160, 170 epochs, and then fine-tuned over 20 epochs, where the learning rate starts at 0.01 and decays by 0.1 after 10, 15 epochs. We note that 20 epochs are enough for convergence even when fine-tuning the whole networks for some methods. We update the model parameters by stochastic gradient decent with a momentum 0.9 and an L2 weight decay of 0.0005. The size of the coreset is set to 2000. Due to the scalability issue, the size of the sampled external dataset is set to the size of the labeled dataset. The ratio of OOD data in sampling is determined by validation on a split of ImageNet, which is 0.7. For all experiments, the temperature for smoothing softmax probabilities γ is set to 2 for distillation from $\mathcal{P}$ and $\mathcal{C}$, and 1 for distillation from $\mathcal{Q}$.

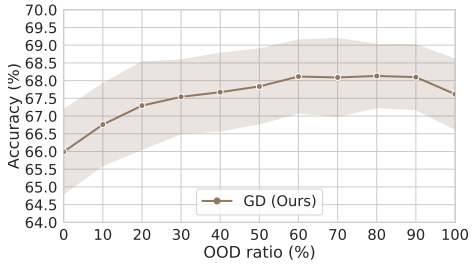
Scalability of methods. We note that all compared methods are scalable and they are compared in a fair condition. We do not compare generative replay methods with ours, because the coreset approach is known to outperform them in class-incremental learning in a scalable setting: in particular, it has been reported that continual learning for a generative model is a challenging problem on datasets of natural images like CIFAR-100 (Lesort et al., 2018; Wu et al., 2018a).

B.3 More Experimental Results

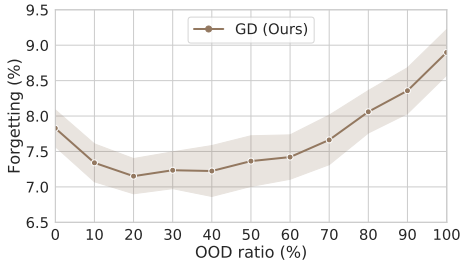
B.3.1 More Ablation Studies

Effect of the OOD ratio. We investigate the effect of the ratio between the sampled data likely to be in the previous tasks and OOD data. As shown in Figure B.2, the optimal OOD ratio varies over datasets, but it is higher than 0.5: specifically, the best ACC is achieved when the OOD ratio is 0.8 on CIFAR-100, and 0.7 on ImageNet. On the other hand, the optimal OOD ratio for FGT is different: specifically, the best FGT is achieved when the OOD ratio is 0.2 on CIFAR-100, and 0.5 on ImageNet.

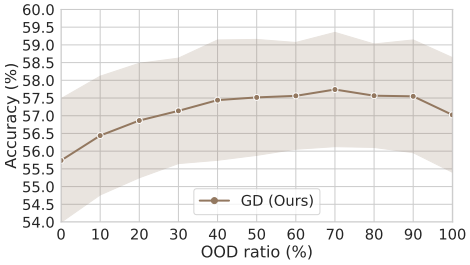
Effect of the correlation between the training data and unlabeled external data. So far, we do not assume any correlation between training data and external data. However, in this experiment, we control the correlation between them based on the hypernym-hyponym



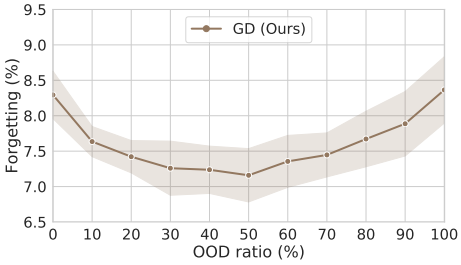
(a) CIFAR-100 ACC



(b) CIFAR-100 FGT

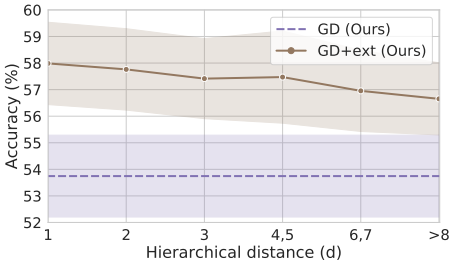


(c) ImageNet ACC

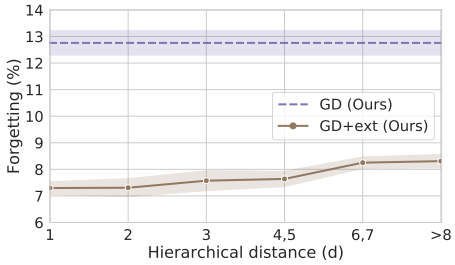


(d) ImageNet FGT

Figure B.2: Experimental results on CIFAR-100 and ImageNet when the task size is 10. We report ACC and FGT with respect to the OOD ratio averaged over ten trials for CIFAR-100 and nine trials for ImageNet.



(a) ImageNet ACC



(b) ImageNet FGT

Figure B.3: Experimental results on ImageNet when the task size is 10. We report ACC and FGT with respect to the hierarchical distance between the training dataset and unlabeled data stream averaged over nine trials.

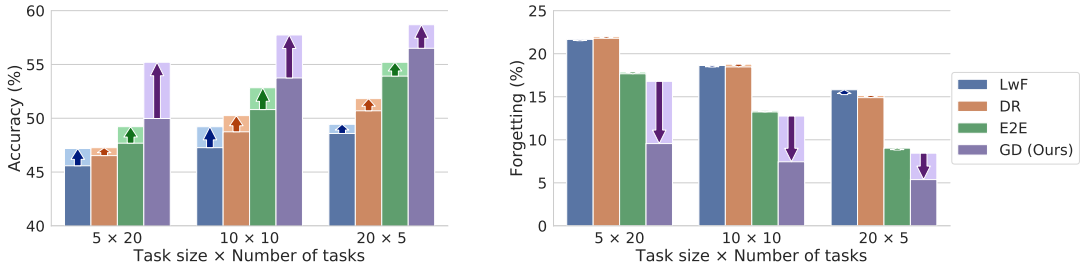
relationship between ImageNet class labels. Specifically, we first compute the hierarchical distance (the length of the shortest path between classes in hierarchy) between 1k classes in ImageNet ILSVRC 2012 training dataset and the other 21k classes in the entire ImageNet 2011 dataset. Note that the hierarchical distance can be thought as the semantic difference between classes. Then, we divide the 21k classes based on the hierarchical distance, such that each split has at least 1M images for simulating an unlabeled data stream. As shown in Figure B.3, the performance is proportional to the semantic similarity, which is inversely proportional the hierarchical distance. However, even in the worst case, unlabeled data are beneficial.

B.3.2 More Results

Table B.1: Comparison of methods on CIFAR-100 and ImageNet. We report the mean and standard deviation of ten trials for CIFAR-100 and nine trials for ImageNet with different random seeds in %. ↑ (↓) indicates that the higher (lower) number is the better.

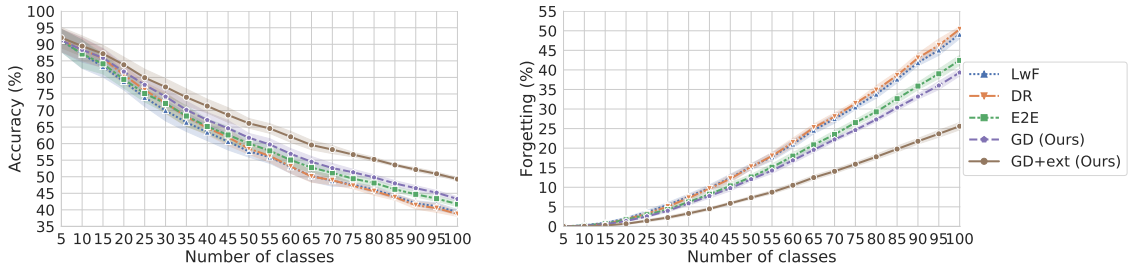
Dataset	CIFAR-100					
Task size	5		10		20	
Metric	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)
Oracle	78.6 ± 0.9	3.3 ± 0.2	77.6 ± 0.8	3.1 ± 0.2	75.7 ± 0.7	2.8 ± 0.2
Without an external dataset						
Baseline	57.4 ± 1.2	21.0 ± 0.5	56.8 ± 1.1	19.7 ± 0.4	56.0 ± 1.0	18.0 ± 0.3
LwF	58.4 ± 1.3	19.3 ± 0.5	59.5 ± 1.2	16.9 ± 0.4	60.0 ± 1.0	14.5 ± 0.4
DR	59.1 ± 1.4	19.6 ± 0.5	60.8 ± 1.2	17.1 ± 0.4	61.8 ± 0.9	14.3 ± 0.4
E2E	60.2 ± 1.3	16.5 ± 0.5	62.6 ± 1.1	12.8 ± 0.4	65.1 ± 0.8	8.9 ± 0.2
GD (Ours)	**62.1** ± **1.2**	**15.4** ± **0.4**	**65.0** ± **1.1**	**12.1** ± **0.3**	**67.1** ± **0.9**	**8.5** ± **0.3**
With an external dataset						
LwF	59.7 ± 1.2	19.4 ± 0.5	61.2 ± 1.1	17.0 ± 0.4	60.8 ± 0.9	14.8 ± 0.4
DR	59.8 ± 1.0	19.5 ± 0.5	62.0 ± 0.9	16.8 ± 0.4	63.0 ± 1.0	13.9 ± 0.4
E2E	61.5 ± 1.2	16.4 ± 0.5	64.3 ± 1.0	12.7 ± 0.4	66.1 ± 0.9	9.2 ± 0.4
GD (Ours)	**66.3** ± **1.2**	**9.8** ± **0.3**	**68.1** ± **1.1**	**7.7** ± **0.3**	**68.9** ± **1.0**	**5.5** ± **0.4**

Dataset	ImageNet					
Task size	5		10		20	
Metric	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)
Oracle	68.0 ± 1.7	3.3 ± 0.2	66.9 ± 1.6	3.1 ± 0.3	65.1 ± 1.2	2.7 ± 0.2
Without an external dataset						
Baseline	44.2 ± 1.7	23.6 ± 0.4	44.1 ± 1.6	21.5 ± 0.5	44.7 ± 1.2	18.4 ± 0.5
LwF	45.6 ± 1.9	21.5 ± 0.4	47.3 ± 1.5	18.5 ± 0.5	48.6 ± 1.2	15.3 ± 0.6
DR	46.5 ± 1.6	22.0 ± 0.5	48.7 ± 1.6	18.8 ± 0.5	50.7 ± 1.2	15.1 ± 0.5
E2E	47.7 ± 1.9	17.9 ± 0.4	50.8 ± 1.5	13.4 ± 0.4	53.9 ± 1.2	8.8 ± 0.3
GD (Ours)	**50.0** ± **1.7**	**16.8** ± **0.4**	**53.7** ± **1.5**	**12.8** ± **0.5**	**56.5** ± **1.2**	**8.4** ± **0.4**
With an external dataset						
LwF	47.2 ± 1.7	21.7 ± 0.5	49.2 ± 1.3	18.6 ± 0.4	49.4 ± 0.8	15.8 ± 0.4
DR	47.3 ± 1.7	21.8 ± 0.6	50.2 ± 1.5	18.5 ± 0.5	51.8 ± 0.9	14.9 ± 0.5
E2E	49.2 ± 1.7	17.7 ± 0.6	52.8 ± 1.4	13.2 ± 0.2	55.2 ± 0.9	9.0 ± 0.4
GD (Ours)	**55.2** ± **1.8**	**9.6** ± **0.4**	**57.7** ± **1.6**	**7.4** ± **0.3**	**58.7** ± **1.2**	**5.4** ± **0.3**



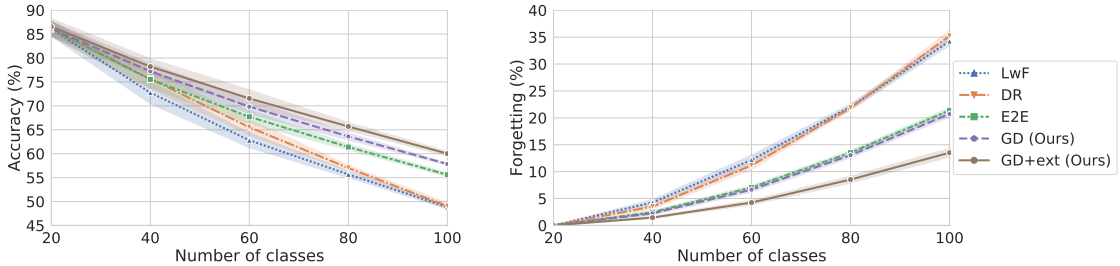
(a) ACC gain by learning with external data (b) FGT gain by learning with external data

Figure B.4: Experimental results on ImageNet. Arrows show the performance gain in ACC and FGT by learning with unlabeled data, respectively. We report the average performance of nine trials.



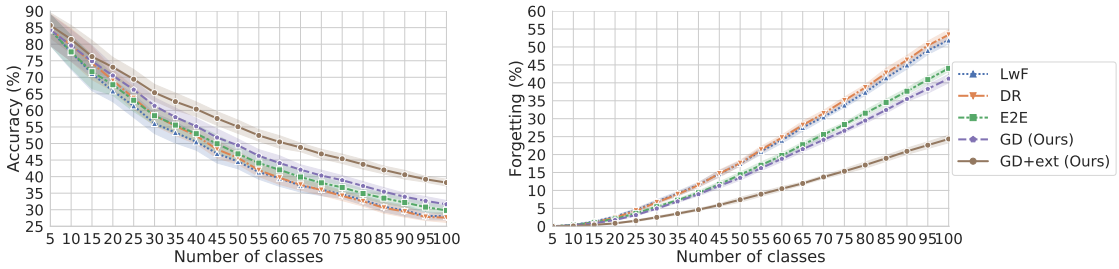
(a) ACC w.r.t the number of trained classes (b) FGT w.r.t the number of trained classes

Figure B.5: Experimental results on CIFAR-100 when the task size is 5. We report ACC and FGT with respect to the number of trained classes averaged over ten trials.



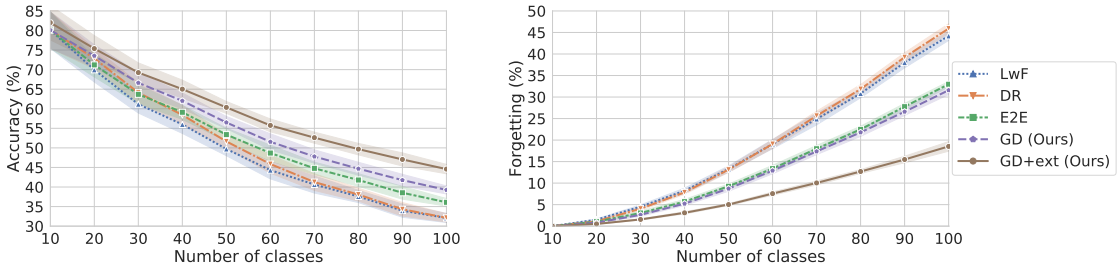
(a) ACC w.r.t the number of trained classes (b) FGT w.r.t the number of trained classes

Figure B.6: Experimental results on CIFAR-100 when the task size is 20. We report ACC and FGT with respect to the number of trained classes averaged over ten trials.



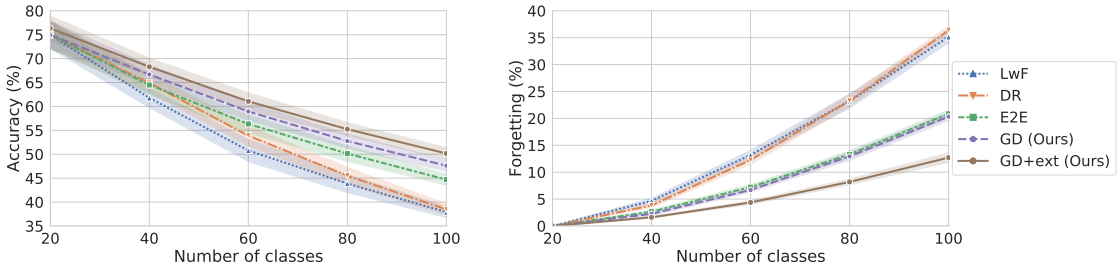
(a) ACC w.r.t the number of trained classes (b) FGT w.r.t the number of trained classes

Figure B.7: Experimental results on ImageNet when the task size is 5. We report ACC and FGT with respect to the number of trained classes averaged over nine trials.



(a) ACC w.r.t the number of trained classes (b) FGT w.r.t the number of trained classes

Figure B.8: Experimental results on ImageNet when the task size is 10. We report ACC and FGT with respect to the number of trained classes averaged over nine trials.



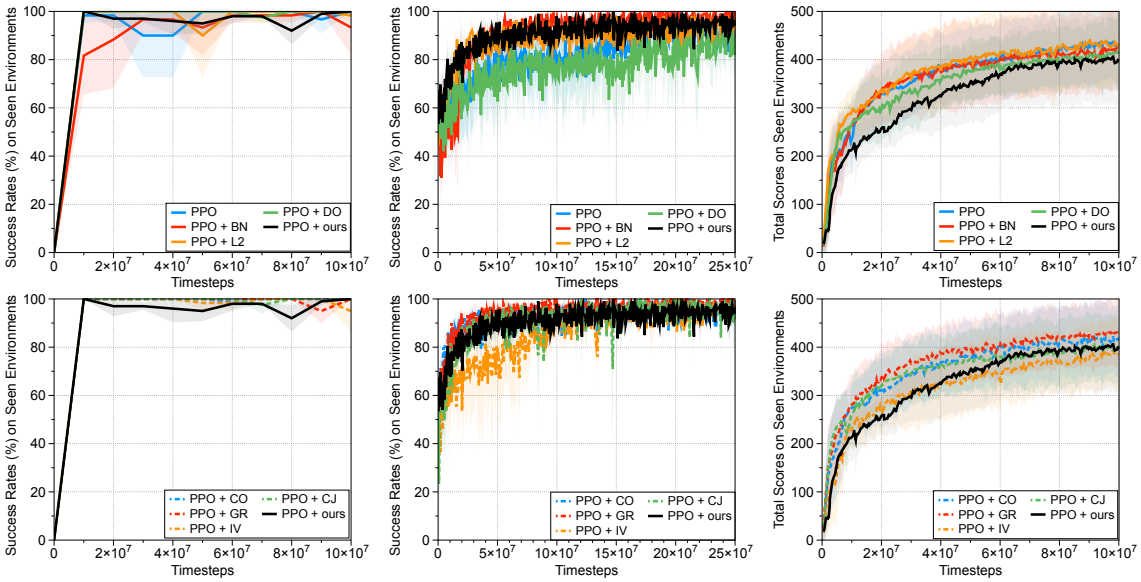
(a) ACC w.r.t the number of trained classes (b) FGT w.r.t the number of trained classes

Figure B.9: Experimental results on ImageNet when the task size is 20. We report ACC and FGT with respect to the number of trained classes averaged over nine trials.

APPENDIX C

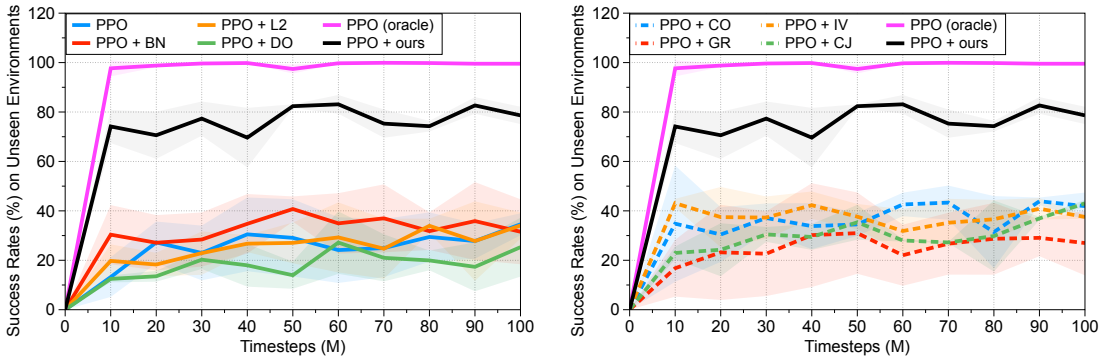
Supplementary Material of Chapter IV

C.1 Learning Curves



(a) Small-scale CoinRun (b) Large-scale CoinRun (c) DeepMind Lab

Figure C.1: Learning curves on (a) small-scale, (b) large-scale CoinRun and (c) DeepMind Lab. The solid line and shaded regions represent the mean and standard deviation, respectively, across three runs.



(a) Comparison with regularization techniques (b) Comparison with data augmentation techniques

Figure C.2: The performance in unseen environments in small-scale CoinRun. The solid/dashed line and shaded regions represent the mean and standard deviation, respectively, across three runs.

C.2 Robustness Against Adversarial Attacks

The adversarial (visually imperceptible) perturbation (Szegedy et al., 2014) to clean input observations can induce the DNN-based policies to generate an incorrect decision at test time (Huang et al., 2017; Lin et al., 2017). This undesirable property of DNNs has raised major security concerns. In this section, we evaluate if the proposed method can improve the robustness on adversarial attacks. Our method is expected to improve the robustness against such adversarial attacks because the agents are trained with randomly perturbed inputs. To verify that the proposed method can improve the robustness to adversarial attacks, the adversarial samples are generated using FGSM (Goodfellow et al., 2015) by perturbing inputs to the opposite direction to the most probable action initially predicted by the policy:

$$s_{\texttt{adv}} = s - \varepsilon \text{sign}\left(\nabla_s \log \pi(a^*|s;\theta)\right),$$

where ε is the magnitude of noise and $a^* = \arg\max_a \pi(a|s;\theta)$ is the action from the policy. Table C.1 shows that our proposed method can improve the robustness against FGSM attacks with $\varepsilon = 0.01$, which implies that hidden representations of trained agents are more robust.

C.3 Details for Training Agents Using PPO

Policy optimization. For all baselines and our methods, PPO is utilized to train the policies. Specifically, we use a discount factor $\gamma = 0.999$, a generalized advantage estimator (GAE) (Schulman et al., 2016) parameter $\lambda = 0.95$, and an entropy bonus (Williams and

	Small-Scale CoinRun		Large-Scale CoinRun		DeepMind Lab	
	Clean	FGSM	Clean	FGSM	Clean	FGSM
PPO	100	61.5 (-38.5)	96.2	77.4 (-19.5)	352.5	163.5 (-53.6)
PPO + ours	100	88.0 **(-12.0)**	99.6	84.4 **(-15.3)**	368.0	184.0 **(-50.0)**

Table C.1: Robustness against FGSM attacks on training environments. The values in parentheses represent the relative reductions from the clean samples.

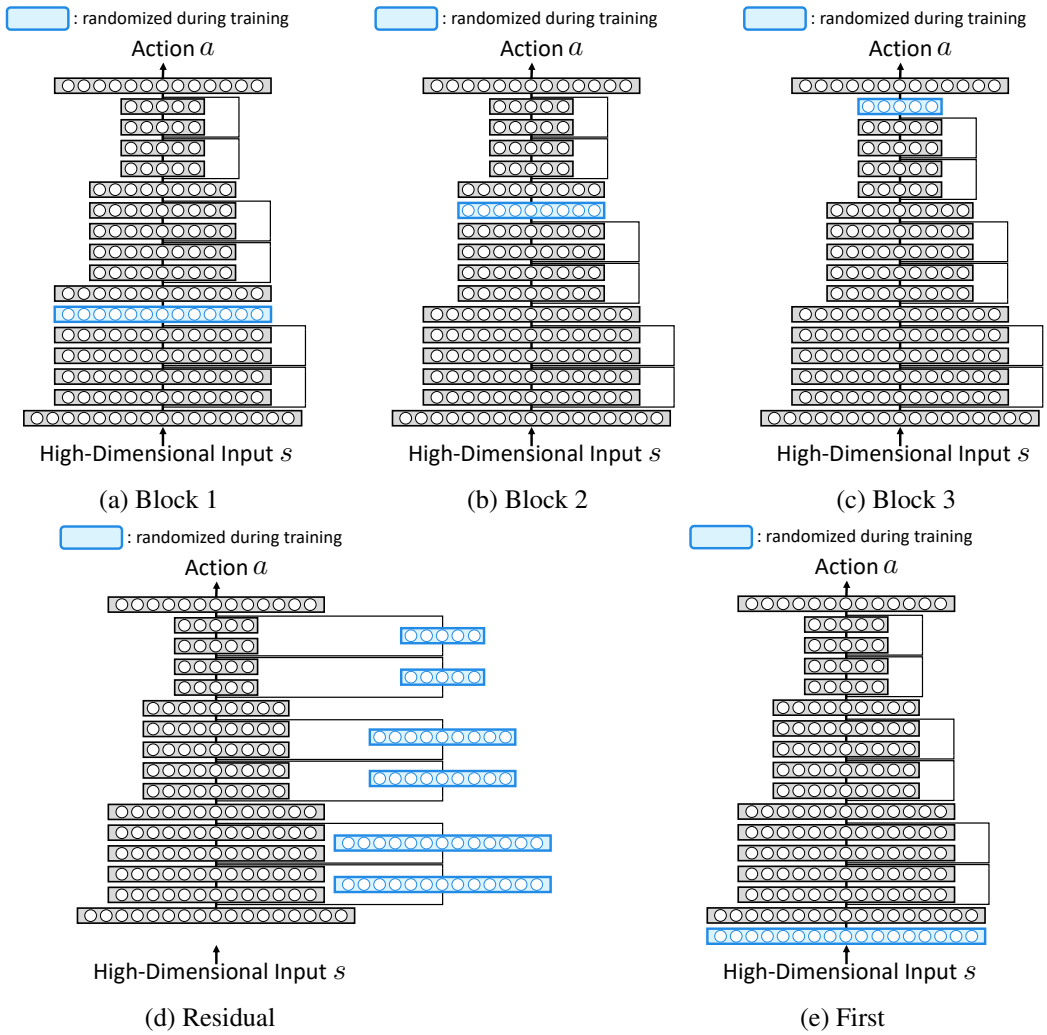
Peng, 1991) of 0.01 to ensure sufficient exploration. We extract 256 timesteps per rollout, and then train the agent for 3 epochs with 8 mini-batches. The Adam optimizer (Kingma and Ba, 2015) is used with the starting learning rate 0.0005. We run 32 environments simultaneously during training. As suggested in Cobbe et al. (2019), two boxes are painted in the upper-left corner, where their color represents the x- and y-axis velocity to help the agents quickly learn to act optimally. In this way, the agent does not need to memorize previous states, so a simple CNN-based policy without LSTM can effectively perform in our experimental settings.

Data augmentation methods. In this chapter, we compare a variant of cutout (DeVries and Taylor, 2017b) proposed in Cobbe et al. (2019), grayout, inversion, and color jitter (Cubuk et al., 2019a). Specifically, the cutout augmentation applies a random number of boxes in random size and color to the input, the grayout method averages all three channels of the input, the inversion method inverts pixel values by a 50% chance, and the color jitter changes the characteristics of images commonly used for data augmentation in computer vision tasks: brightness, contrast, and saturation. For every timestep in the cutout augmentation, we first randomly choose the number of boxes from zero to five, assign them a random color and size, and place them in the observation. For the color jitter, the parameters for brightness, contrast, and saturation are randomly chosen in [0.5,1.5].[1] For each episode, the parameters of these methods are randomized and fixed such that the same image pre-processing is applied within an episode.

C.4 Different Types of Random Networks

In this section, we apply random networks to various locations in the network architecture as illustrated in Figure C.3, and measure the performance in large-scale CoinRun without

[1]For additional details, see https://pytorch.org/docs/stable/_modules/torchvision/transforms/transforms.html#ColorJitter.

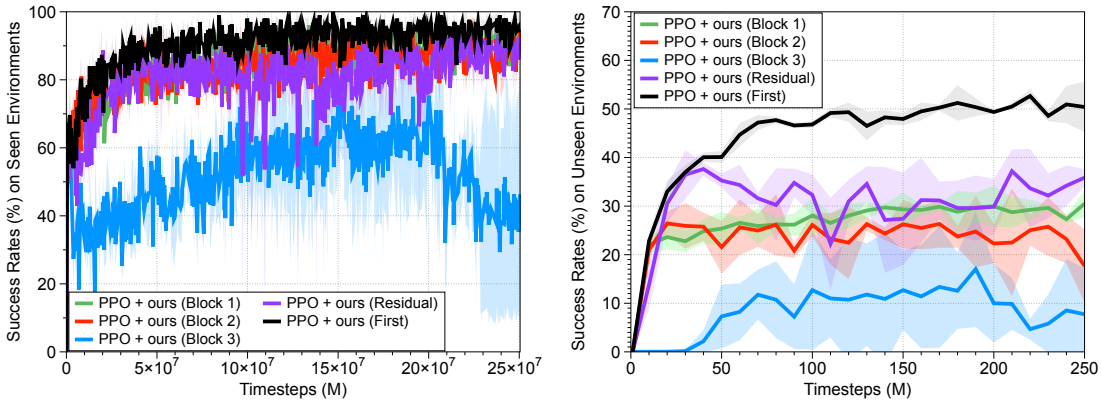


(a) Block 1 (b) Block 2 (c) Block 3

(d) Residual (e) First

Figure C.3: Network architectures with random networks in various locations. Only convolutional layers and the last fully connected layer are displayed for conciseness.

the feature matching loss. For all methods, a single-layer CNN is used with a kernel size of 3, and its output tensor is padded in order to be in the same dimension as the input tensor. As shown in Figure C.4, the performance of unseen environments decreases as the random network is placed in higher layers. On the other hand, the random network in residual connections improves the generalization performance, but it does not outperform the case when a random network is placed at the beginning of the network, meaning that randomizing only the local features of inputs can be effective for a better generalization performance.



(a) Performance in seen environments

(b) Performance in unseen environments

Figure C.4: The performance of random networks in various locations in the network architecture on (a) seen and (b) unseen environments in large-scale CoinRun. We show the mean performances averaged over three different runs, and shaded regions represent the standard deviation.

C.5 Environments in Small-Scale CoinRun

For small-scale CoinRun environments, we consider a fixed map layout with two moving obstacles and measure the performance of the trained agents by changing the style of the backgrounds.

(a) Seen environment

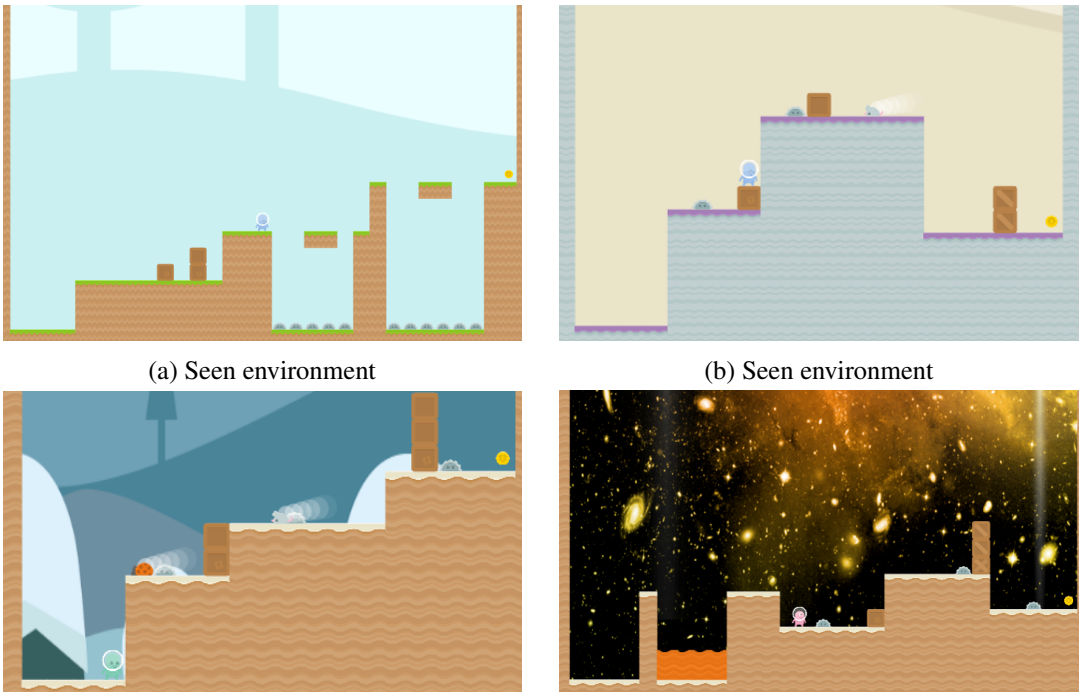
(b) Unseen environment

(c) Unseen environment

Figure C.5: Examples of seen and unseen environments in small-scale CoinRun.

C.6 Environments in Large-Scale CoinRun

In CoinRun, there are 34 themes for backgrounds, 6 for grounds, 5 for agents, and 9 for obstacles. For the large-scale CoinRun experiment, we train agents on a fixed set of 500 levels of CoinRun using half of the available themes and measure the performances on 1,000 different levels consisting of unseen themes.



(a) Seen environment (b) Seen environment

(c) Unseen environment (d) Unseen environment

Figure C.6: Examples of seen and unseen environments in large-scale CoinRun.

C.7 Environments on DeepMind Lab

Dataset. Among the styles (textures and colors) provided for the 3D maze in the DeepMind Lab, we take ten different styles of floors and walls, respectively (see the list below). We construct a training dataset by randomly choosing a map layout and assigning a theme among ten floors and walls, respectively. The domain randomization method compared in Table 4.3 uses four floors and four wall themes (16 combinations in total). Trained themes are randomly chosen before training and their combinations are considered to be seen environments. To evaluate the generalization ability, we measure the performance of trained agents on unseen environments by changing the styles of walls and floors. Domain randomization has more seen themes than the other methods, so all methods are compared with six floors and six walls (36 combinations in total), which are unseen for all methods. The mean and standard deviation of the average scores across ten different map layouts are reported in Figure C.7.

Action space. Similar to IMPALA (Espeholt et al., 2018), the agent can take eight actions from the DeepMind Lab native action samples: {Forward, Backward, Move Left, Move Right, Look Left, Look Right, Forward + Look Left, and Forward + Look Right}. Table C.2 describes the detailed mapping.

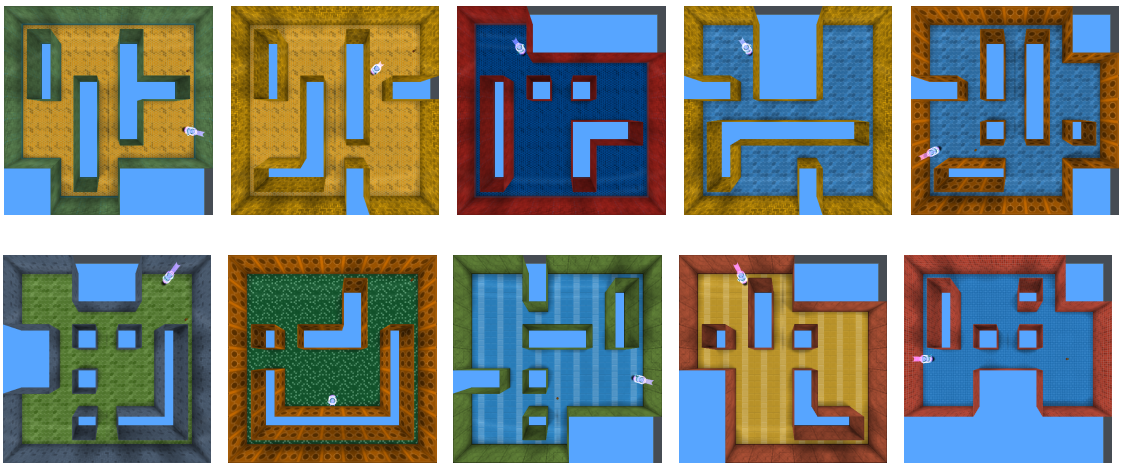


Figure C.7: The top-down view of the trained map layouts.

Action	DeepMind Lab Native Action
Forward	[0, 0, 0, 1, 0, 0, 0]
Backward	[0, 0, 0, -1, 0, 0, 0]
Move Left	[0, 0, -1, 0, 0, 0, 0]
Move Right	[0, 0, 1, 0, 0, 0, 0]
Look Left	[-20, 0, 0, 0, 0, 0, 0]
Look Right	[20, 0, 0, 0, 0, 0, 0]
Forward + Look Left	[-20, 0, 0, 1, 0, 0, 0]
Forward + Look Right	[20, 0, 0, 1, 0, 0, 0]

Table C.2: Action set used in the DeepMind Lab experiment. The DeepMind Lab native action set consists of seven discrete actions encoded in integers ([L,U] indicates the lower/upper bound of the possible values): 1) yaw (left/right) rotation by pixel [-512,512], 2) pitch (up/down) rotation by pixel [-512,512], 3) horizontal move [-1,1], 4) vertical move [-1,1], 5) fire [0,1], 6) jump [0,1], and 7) crouch [0,1].

C.8 Experiments on Dogs and Cats Database

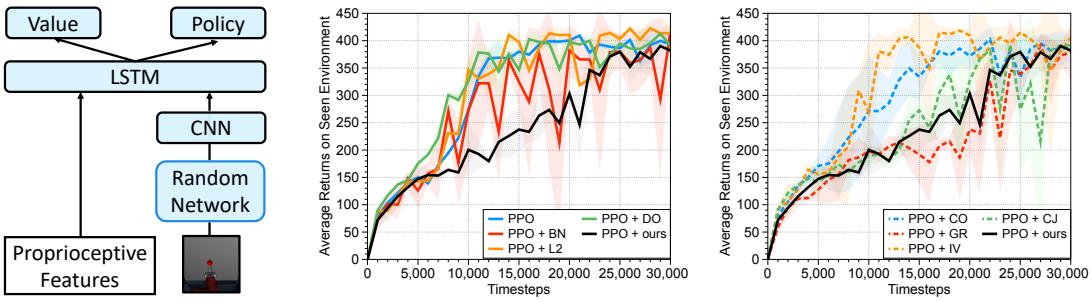
Dataset. The original database is a set of 25,000 images of dogs and cats for training and 12,500 images for testing. Similar to Kim et al. (2019), the data is manually categorized according to the color of the animal: bright or dark. Biased datasets are constructed such that the training set consists of bright dogs and dark cats, while the test and validation sets contain dark dogs and bright cats. Specifically, training, validation, and test sets consist of 10,047, 1,000, and 5,738 images, respectively.[2] ResNet-18 (He et al., 2016) is trained with an initial learning rate chosen from {0.05, 0.1} and then dropped by 0.1 at 50 epochs with

[2] https://github.com/feidfoe/learning-not-to-learn/tree/master/dataset/dogs_and_cats.

a total of 100 epochs. We use the Nesterov momentum of 0.9 for SGD, a mini-batch size chosen from {32, 64}, and the weight decay set to 0.0001. We report the training and test set accuracies with the hyperparameters chosen by validation. Unlike Kim et al. (2019), we do not use ResNet-18 pre-trained with ImageNet (Russakovsky et al., 2015) in order to avoid inductive bias from the pre-trained dataset.

C.9 Experimental Results on Surreal Robot Manipulation

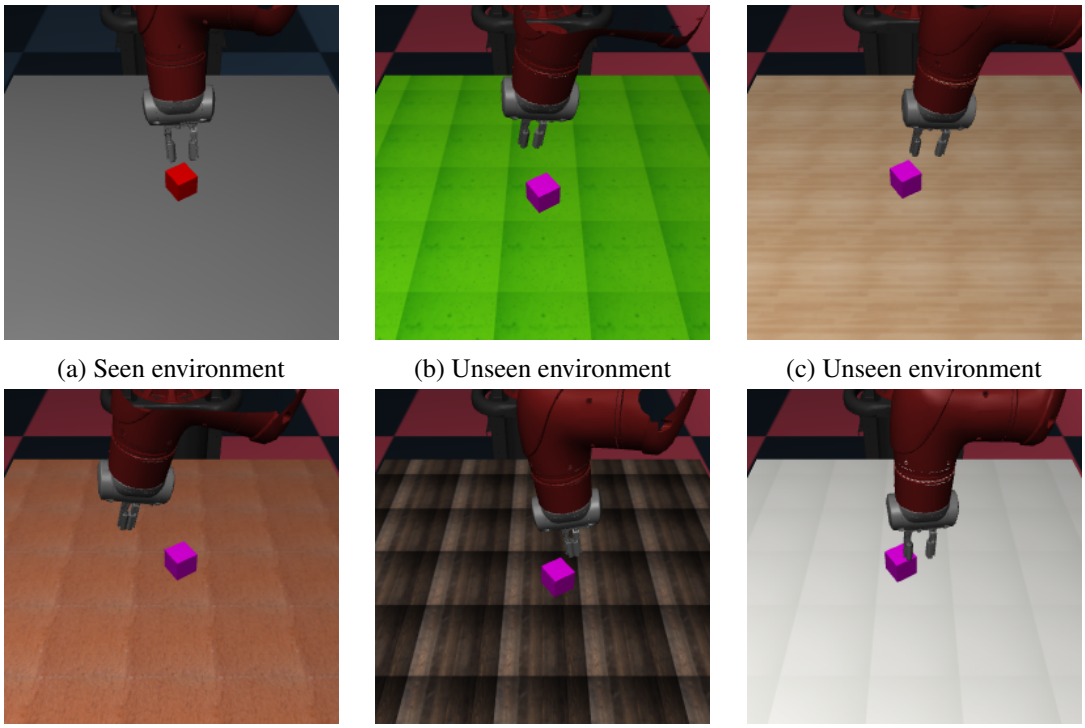
Our method is evaluated in the Block Lifting task using the Surreal distributed RL framework (Fan et al., 2018). In this task, the Sawyer robot receives a reward if it successfully lifts a block randomly placed on a table. Following the experimental setups in Fan et al. (2018), the hybrid CNN-LSTM architecture (see Figure C.8(a)) is chosen as the policy network and a distributed version of PPO (i.e., actors collect massive amount of trajectories and the centralized learner updates the model parameters using PPO) is used to train the agents.[3] Agents take 84×84 observation frames with proprioceptive features (e.g., robot joint positions and velocities) and output the mean and log of the standard deviation for each action dimension. The actions are then sampled from the Gaussian distribution parameterized by the output. Agents are trained on a single environment and tested on five unseen environments with various styles of table, floor, and block, as shown in Figure C.9. For the Surreal robot manipulation experiment, the vanilla PPO agent is trained on 25 environments generated by changing the styles of tables and boxes. Specifically, we use {blue, gray, orange, white, purple} and {red, blue, green, yellow, cyan} for table and box, respectively.



(a) Network architecture (b) Regularization (c) Data augmentation

Figure C.8: (a) An illustration of network architectures for the Surreal robotics control experiment, and learning curves with (b) regularization and (c) data augmentation techniques. The solid line and shaded regions represent the mean and standard deviation, respectively, across three runs.

[3] We referred to the two actors setting in https://github.com/SurrealAI/surreal.



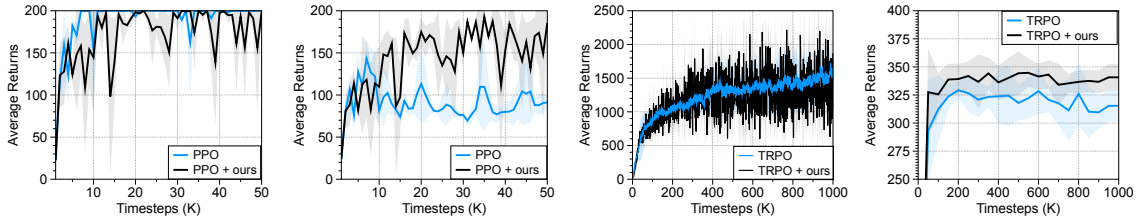
(a) Seen environment (b) Unseen environment (c) Unseen environment

(d) Unseen environment (e) Unseen environment (f) Unseen environment

Figure C.9: Examples of seen and unseen environments in the Surreal robot manipulation.

C.10 Extension to Domains With Different Dynamics

In this section, we consider an extension to the generalization on domains with different dynamics. Similar to dynamics randomization (Peng et al., 2018), one can expect that our idea can be useful for improving the dynamics generalization. To verify this, we conduct an experiment on CartPole and Hopper environments where an agent takes proprioceptive features (e.g., positions and velocities). The goal of CartPole is to prevent the pole from falling over, while that of Hopper is to make an one-legged robot hop forward as fast as possible, respectively. Similar to the randomization method we applied to visual inputs, we introduce a random layer between the input and the model. As a natural extension of the proposed method, we consider performing the convolution operation by multiplying a $d \times d$ diagonal matrix to d-dimensional input states. For every training iteration, the elements of the matrix are sampled from the standard uniform distribution $U(0.8, 1.2)$. One can note that this method can randomize the amplitude of input states while maintaining the intrinsic information (e.g., sign of inputs). Following Packer et al. (2018) and Zhou et al. (2019), we measure the performance of the trained agents on unseen environments with a different set of dynamics parameters, such as mass, length, and force. Specifically,



(a) CartPole (seen) (b) CartPole (unseen) (c) Hopper (seen) (d) Hopper (unseen)

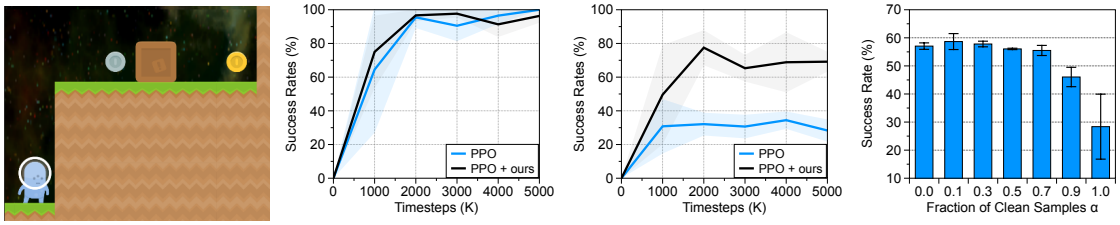
Figure C.10: Performances of trained agents in seen and unseen environments under (a/b) CartPole and (c/d) Hopper. The solid/dashed lines and shaded regions represent the mean and standard deviation, respectively.

for CartPole experiments, similar to Packer et al. (2018), the policy and value functions are multi-layer perceptrons (MLPs) with two hidden layers of 64 units each and hyperbolic tangent activation and the Proximal Policy Optimization (PPO) (Schulman et al., 2017) method is used to train the agents. The parameters of the training environment are fixed at the default values in the implementations from Gym, while force, length, and mass of environments are sampled from $[1, 5] \cup [15, 20]$, $[0.05, 0.25] \cup [0.75, 1.0]$, $[0.01, 0.05] \cup [0.5, 1.0]$ that the policy has never seen in any stage of training.[4] For Hopper experiments, similar to Zhou et al. (2019), the policy is a MLP with two hidden layers of 32 units each and ReLU activation and value function is a linear model. The trust region policy optimization (TRPO) (Schulman et al., 2015) method is used to train the agents. The mass of the training environment is sampled from $\{1.0, 2.0, 3.0, 4.0, 5.0\}$, while it is sampled from $\{6.0, 7.0, 8.0\}$ during testing.[5] Figure C.10 reports the mean and standard deviation across 3 runs. Our simple randomization improves the performance of the agents in unseen environments, while achieving performance comparable to seen environments. We believe that this evidences a wide applicability of our idea beyond visual changes.

C.11 Failure Case of Our Methods

In this section, we verify whether the proposed method can handle color (or texture)-conditioned RL tasks. One might expect that such RL tasks can be difficult for our methods to work because of the randomization. For example, our methods would fail if we consider an extreme seek-avoid object gathering setup, where the agent must learn to collect good objects and avoid bad objects which have the same shape but different color. However, we remark that our method would not always fail for such tasks if other environmental fac-

[4] We referred to https://bair.berkeley.edu/blog/2019/03/18/rl-generalization.
[5] We referred to the two actors setting in https://github.com/Wenxuan-Zhou/EPI.



(a) CoinRun with good and bad coins (b) Performance in seen environments (c) Performance in unseen environments (d) Performance in unseen environments

Figure C.11: (a) Modified CoinRun with good and bad coins. The performances on (b) seen and (c) unseen environments. The solid line and shaded regions represent the mean and standard deviation, respectively, across three runs. (d) Average success rates on large-scale CoinRun for varying the fraction of clean samples during training. Noe that $\alpha = 1$ corresponds to vanilla PPO agents.

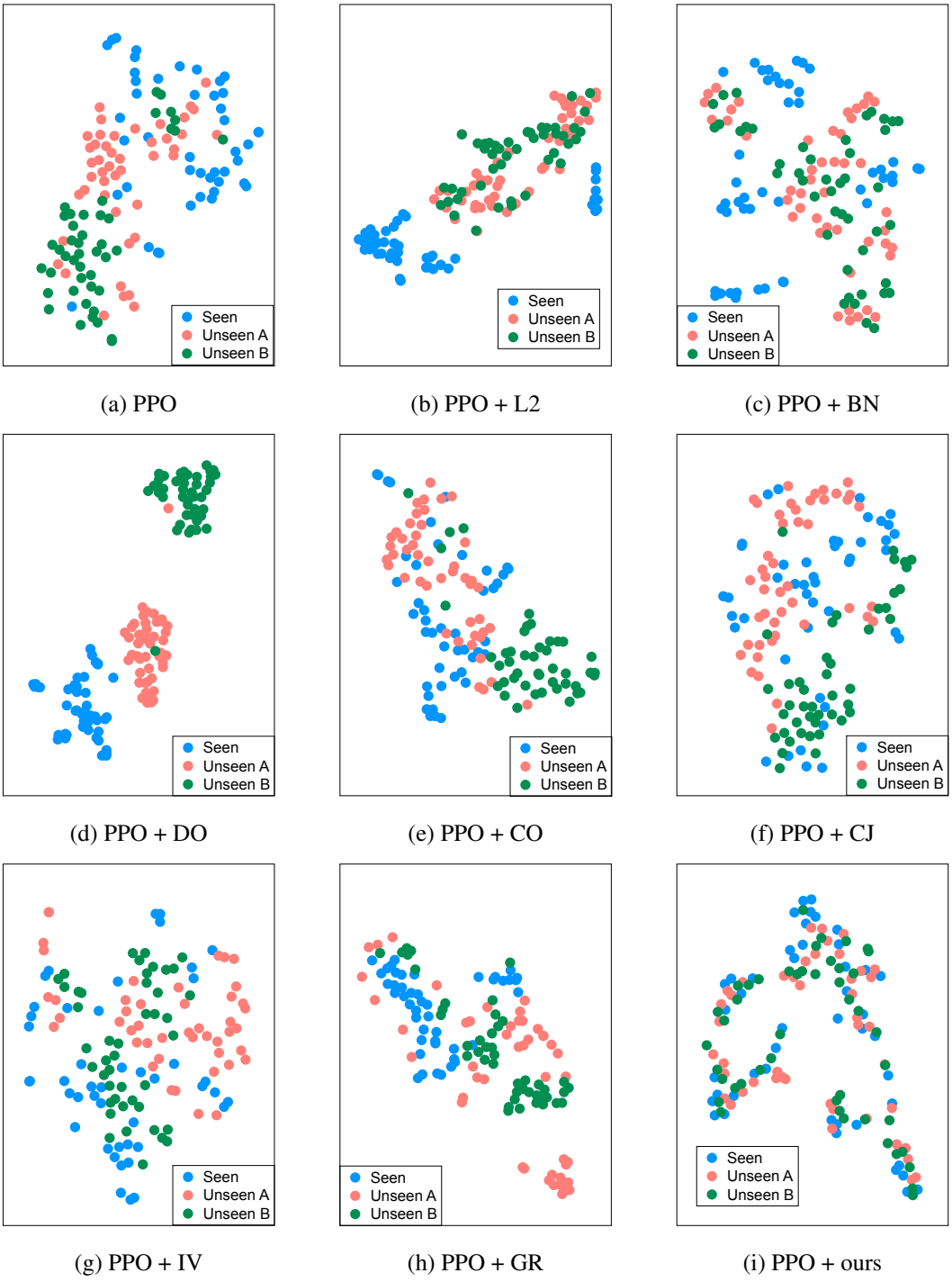
tors (e.g., the shape of objects in Collect Good Objects in DeepMind Lab (Beattie et al., 2016)) are available to distinguish them. To verify this, we consider a modified CoinRun environment where the agent must learn to collect good objects (e.g., gold coin) and avoid bad objects (e.g., silver coin). Similar to the small-scale CoinRun experiment, agents are trained to collect the goal object in a fixed map layout (see Figure C.11(a)) and tested in unseen environments with only changing the style of the background. Figure C.11(b) shows that our method can work well for such color-conditioned RL tasks because a trained agent can capture the other factors such as a location to perform this task. Besides, our method achieves a significant performance gain compared to vanilla PPO agent in unseen environments as shown in Figure C.11(c).

As another example, in color-matching tasks such as the keys doors puzzle in DeepMind Lab (Beattie et al., 2016), the agent must collect colored keys to open matching doors. Even though this task is color-conditioned, a policy trained with our method can perform well because the same colored objects will have the same color value even after randomization, i.e., our randomization method still maintains the structure of input observation. This evidences the wide applicability of our idea. We also remark that our method can handle more extreme corner cases by adjusting the fraction of clean samples during training. In summary, we believe that the proposed method covers a broad scope of generalization across low-level transformations in the observation space features.

C.12 Ablation Study for Fraction of Clean Samples

We investigate the effect of the fraction of clean samples. Figure C.11(d) shows that the best unseen performance is achieved when the fraction of clean samples is 0.1 on large-scale CoinRun.

C.13 Visualization of Hidden Features



(a) PPO (b) PPO + L2 (c) PPO + BN

(d) PPO + DO (e) PPO + CO (f) PPO + CJ

(g) PPO + IV (h) PPO + GR (i) PPO + ours

Figure C.12: Visualization of the hidden representation of trained agents optimized by (a) PPO, (b) PPO + L2, (c) PPO + BN, (d) PPO + DO, (e) PPO + CO, (f) PPO + GR, (g) PPO + IV, (h) PPO + CJ, and (I) PPO + ours using t-SNE. The point colors indicate the environments of the corresponding observations.

APPENDIX D

Supplementary Material of Chapter V

D.1 More Applications of i-MixUp

In this section, we introduce a few more variations of i-MixUp. For conciseness, we use v_i to denote virtual labels for different methods. We make the definition of v_i for each application clear.

D.1.1 i-MixUp for SimCLR

For each anchor, SimCLR takes other anchors as negative samples such that the virtual labels must be extended. Let $x_{N+i} = \tilde{x}_i$ for conciseness, and $v_i \in [0,1]^{2N}$ be the virtual label indicating the positive sample of each anchor, where $v_{i,N+i} = 1$ and $v_{i,j} = 0$ for $j \neq N+i$. Note that $v_{i,i} = 0$ because the anchor itself is not counted as a positive sample. Then, Eq. (5.2) can be represented in the form of the cross-entropy loss:

$$\ell_{\text{SimCLR}}(x_i, v_i; \mathcal{B}) = -\sum_{n=1}^{2N} v_{i,n} \log \frac{\exp\big(s(f_i, f_n)/\tau\big)}{\sum_{k=1,k\neq i}^{2N} \exp\big(s(f_i, f_k)/\tau\big)}. \tag{D.1}$$

The application of i-MixUp to SimCLR is straightforward: for two data instances (x_i, v_i), (x_j, v_j) and a batch of data $\mathcal{B} = \{x_i\}_{i=1}^{2N}$, the i-MixUp loss is defined as follows:[1]

$$\ell_{\text{SimCLR}}(\lambda x_i + (1-\lambda)x_j, \lambda v_i + (1-\lambda)v_j; \mathcal{B}). \tag{D.2}$$

[1] The j-th data can be excluded from the negative samples, but it does not result in a significant difference.

Note that only the input data of Eq. (D.2) is mixed, such that f_i in Eq. (D.1) is an embedding vector of the mixed data while the other f_n's are the ones of clean data. Because both clean data x_i's and their interpolations $\lambda x_i + (1-\lambda)x_j$'s need to be fed to the network f, i-MixUp for SimCLR requires twice more memory and training time compared to SimCLR when the same batch size is used.

D.1.2 i-MixUp for Supervised Contrastive Learning

Supervised contrastive learning has recently shown to be effective for supervised representation learning and it often outperforms the standard end-to-end supervised classifier learning (Khosla et al., 2020). Suppose an one-hot label $y_i \in [0,1]^C$ is assigned to a data x_i, where C is the number of classes. Let $x_{N+i} = \tilde{x}_i$ and $y_{N+i} = y_i$ for conciseness. For a batch of data pairs and their labels $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{2N}$, let $v_i \in [0,1]^{2N}$ be the virtual label indicating the positive samples of each anchor, where $v_{i,j} = 1$ if $y_j = y_i$ and $j \neq i$, and otherwise $v_{i,j} = 0$. Intuitively, $\sum_{j=1}^{2N} v_{i,j} = 2N_{y_i} - 1$ where N_{y_i} is the number of data with the label y_i. Then, the supervised version of the SimCLR (Sup-SimCLR) loss function is written as follows:

$$\ell_{\text{Sup-SimCLR}}(x_i, v_i; \mathcal{B}) = -\frac{1}{2N_{y_i} - 1} \sum_{n=1}^{2N} v_{i,n} \log \frac{\exp\left(s(f_i, f_n)/\tau\right)}{\sum_{k=1, k\neq i}^{2N} \exp\left(s(f_i, f_k)/\tau\right)}. \tag{D.3}$$

The application of i-MixUp to Sup-SimCLR is straightforward: for two data instances (x_i, v_i), (x_j, v_j) and a batch of data $\mathcal{B} = \{x_i\}_{i=1}^{2N}$, the i-MixUp loss is defined as follows:

$$\ell_{\text{Sup-SimCLR}}(\lambda x_i + (1-\lambda)x_j, \lambda v_i + (1-\lambda)v_j; \mathcal{B}). \tag{D.4}$$

D.1.3 i-MixUp for N-Pair Supervised Contrastive Learning

Note that i-MixUp in Eq. (D.4) is not as efficient as Sup-SimCLR in Eq. (D.3) due to the same reason in the case of SimCLR. To overcome this, we reformulate Sup-SimCLR to the N-pair loss (Sohn, 2016). Suppose an one-hot label $y_i \in [0,1]^C$ is assigned to a data x_i, where C is the number of classes. For a batch of data pairs and their labels $\mathcal{B} = \{(x_i, \tilde{x}_i, y_i)\}_{i=1}^{N}$, let $v_i \in [0,1]^N$ be the virtual label indicating the positive samples of each anchor, where $v_{i,j} = 1$ if $y_j = y_i$ and $j \neq i$, and otherwise $v_{i,j} = 0$. Then, the supervised version of the N-pair (Sup-N-pair) contrastive loss function is written as follows:

$$\ell_{\text{Sup-N-pair}}(x_i, v_i; \mathcal{B}) = -\frac{1}{N_{y_i}} \sum_{n=1}^{N} v_{i,n} \log \frac{\exp\left(s(f_i, \tilde{f}_n)/\tau\right)}{\sum_{k=1}^{N} \exp\left(s(f_i, \tilde{f}_k)/\tau\right)}. \tag{D.5}$$

Then, the i-MixUp loss for Sup-N-pair is defined as follows:

$$\ell_{\text{Sup-N-pair}}(\lambda x_i + (1-\lambda)x_j, \lambda v_i + (1-\lambda)v_j; \mathcal{B}). \tag{D.6}$$

D.1.4 i-CutMix

Under an assumption that substituting a continuous region in data to another one is semantically valid, we propose i-CutMix, a variation of i-MixUp applying the idea of CutMix (Yun et al., 2019) when mixing in the data space: instead of interpolating the entire input data, i-CutMix introduces a small box in one data and substitutes the region in the box with another data, where the ratio of area follows the interpolator λ.

D.2 More Experiments

D.2.1 Experimental Settings

CIFAR-10 and 100 (Krizhevsky and Hinton, 2009). The experiments are conducted in two stages: following Chen et al. (2020a), the convolutional neural network (CNN) part of ResNet-50 (He et al., 2016) followed by the two-layer multilayer perceptron (MLP) head (output dimensions are 2048 and 128, respectively) is trained with a batch size of 512 (i.e., 256 pairs) with the stochastic gradient descent (SGD) optimizer with a momentum of 0.9 over 1000 epochs. 10 epochs of warmup with a linear schedule to a initial learning rate of 0.125, followed by the cosine learning rate schedule (Loshchilov and Hutter, 2017) is used. We use the weight decay of 0.0001 at the first stage.

Then, the head of the CNN is replaced with a linear classifier, and only the linear classifier is trained with the labeled dataset. We use a batch size of 512 with the SGD optimizer with a momentum of 0.9 and a initial learning rate of 1 over 100 epochs, where the learning rate is decayed by 0.2 after 60, 75, 90 epochs. No weight decay is used at the second stage. The quality of representation is evaluated by the top-1 accuracy on the downstream task. We sample a single linear interpolator $\lambda \sim \beta(1,1)$ for each training batch. We use the temperature $\tau = 0.5$. Note that the optimal distribution of λ and the optimal value of τ varies over different architectures, methods, and datasets, but the choices above result in a reasonably good performance.

For MoCo, the memory bank size is 4096, and the momentum for the exponential moving average (EMA) update is 0.999.

For data augmentation, we follow Chen et al. (2020a) on CIFAR-10: We apply a set of data augmentations randomly in sequence including resized crop (Szegedy et al., 2015),

horizontal flip with a probability of 0.5, color jittering,[2] and gray scaling with a probability of 0.2.

For denoising autoencoder (DAE) (Vincent et al., 2010), the pretext task is to denoise the Gaussian noise with the zero mean and standard deviation of 0.4. The model is trained with the SGD optimizer with a momentum of 0.9 and a initial learning rate of 0.001 over 400 epochs, where the learning rate decays by 0.1 after 200 and 300 epochs with no weight decay. The decoder is replaced with a learnable linear classifier at the second stage. The classifier is computed by linear regression from the feature matrix obtained without data augmentation to the label matrix using the pseudoinverse, which results in a better performance than the SGD optimizer in this case.

Speech Commands (Warden, 2018). As a backbone network, we take VGGNet-19 (Simonyan and Zisserman, 2015) with batch normalization (Ioffe and Szegedy, 2015). We sample a single linear interpolator $\lambda \sim \beta(1, 1)$ for each training batch. We use the temperature $\tau = 0.2$. 10% of silence data (all zero) are added when training. At the first stage, the model is trained with the SGD optimizer with a momentum of 0.9 and a initial learning rate of 0.0001 over 500 epochs, where the learning rate decays by 0.1 after 300 and 400 epochs and the weight decay of 0.0001. The other settings are the same with the experiments on CIFAR-10 and 100. At the second stage, the MLP head is replaced with a linear classifier. The classifier is computed by linear regression from the feature matrix obtained without data augmentation to the label matrix using the pseudoinverse.

For data augmentation,[3] we apply a set of data augmentations randomly in sequence including changing amplitude, speed, and pitch in time domain, stretching, time shifting, and adding background noise in frequency domain. Each data augmentation is applied with a probability of 0.5. Augmented data are then transformed to the mel spectogram in the size of 32×32.

Tabular datasets (Asuncion and Newman, 2007). As a backbone network, we take a five-layer MLP with batch normalization. The output dimensions of layers are (2048-2048-4096-4096-8192), where all layers have batch normalization folowed by ReLU except for the last layer. The last layer activation is maxout (Goodfellow et al., 2013) with 4 sets, such that the output dimension is 2048. On top of this five-layer MLP, we attach two-layer MLP (2048-128) as a projection head. We sample a single linear interpolator $\lambda \sim \beta(2, 2)$ for each training batch. We use the temperature $\tau = 0.1$. At the first stage, the model is trained with

[2] Specifically, brightness, contrast, and saturation are scaled by a factor uniformly sampled from $[0.6, 1.4]$ at random, and hue is rotated in the HSV space by a factor uniformly sampled from $[-0.1, 0.1]$ at random.

[3] https://github.com/tugstugi/pytorch-speech-commands

Table D.1: Comparison of N-pair contrastive learning, SimCLR, MoCo, and i-MixUp and i-CutMix on them with ResNet-50 on CIFAR-10 and 100. We report the mean and standard deviation of five trials with different random seeds in %. i-MixUp improves the accuracy on the downstream task regardless of the data distribution shift between the pretext and downstream tasks. i-CutMix shows a comparable performance with i-MixUp when the pretext and downstream datasets are the same, but it does not when the data distribution shift occurs.

Pretext	Downstream	N-pair	i-MixUp	i-CutMix
CIFAR-10	CIFAR-10	92.4 $\pm$ 0.1	**94.8** $\pm$ 0.2	94.7 $\pm$ 0.1
	CIFAR-100	60.2 $\pm$ 0.3	**63.3** $\pm$ 0.2	61.5 $\pm$ 0.2
CIFAR-100	CIFAR-10	84.4 $\pm$ 0.2	**86.2** $\pm$ 0.2	85.1 $\pm$ 0.2
	CIFAR-100	68.7 $\pm$ 0.2	**72.3** $\pm$ 0.2	**72.3** $\pm$ 0.4
Pretext	Downstream	SimCLR	i-MixUp	i-CutMix
CIFAR-10	CIFAR-10	92.5 $\pm$ 0.1	**94.8** $\pm$ 0.2	**94.8** $\pm$ 0.2
	CIFAR-100	60.0 $\pm$ 0.2	**61.4** $\pm$ 1.0	57.1 $\pm$ 0.4
CIFAR-100	CIFAR-10	84.4 $\pm$ 0.2	**85.2** $\pm$ 0.3	83.7 $\pm$ 0.6
	CIFAR-100	68.7 $\pm$ 0.2	**72.3** $\pm$ 0.2	71.7 $\pm$ 0.2
Pretext	Downstream	MoCo	i-MixUp	i-CutMix
CIFAR-10	CIFAR-10	90.7 $\pm$ 0.3	92.7 $\pm$ 0.2	**93.1** $\pm$ 0.1
	CIFAR-100	60.8 $\pm$ 0.2	**64.7** $\pm$ 0.4	64.3 $\pm$ 0.1
CIFAR-100	CIFAR-10	83.1 $\pm$ 0.2	**85.5** $\pm$ 0.3	**85.5** $\pm$ 0.2
	CIFAR-100	65.7 $\pm$ 0.1	70.0 $\pm$ 0.4	**70.3** $\pm$ 0.1

a batch size of 1024 (i.e., 512 pairs) with the Adam optimizer with beta values of 0.9 and 0.999, a initial learning rate of 3×10^{-4} and 5×10^{-5} over 500 epochs for CovType and Higgs, respectively, and the weight decay of 0.0001. The other settings are the same with the experiments on CIFAR-10 and 100. At the second stage, the MLP head is replaced with a linear classifier. The classifier is computed by linear regression from the feature matrix obtained without data augmentation to the label matrix using the pseudoinverse. Since the prior knowledge on tabular data is very limited, only the masking noise with a probability of 0.2 is considered as a data augmentation.

D.2.2 Variations of i-MixUp

In this section, we additionally compare i-MixUp on SimCLR and i-CutMix on N-pair, SimCLR, and MoCo. To be fair with the memory usage in the pretext task stage, we reduce the batch size of i-MixUp and i-CutMix by half (256 to 128). Following the learning rate

Table D.2: Comparison of the N-pair self-supervised and supervised contrastive learning methods and i-MixUp on them with ResNet-50 on CIFAR-10 and 100. We also provide the performance of formulations proposed in prior works: SimCLR (Chen et al., 2020a) and supervised SimCLR (Khosla et al., 2020). i-MixUp improves the accuracy on the downstream task regardless of the data distribution shift between the pretext and downstream tasks, except the case that the pretest task has smaller number of classes than that of the downstream task. The quality of representation depends on the pretext task in terms of the performance of transfer learning: self-supervised learning is better on CIFAR-10, while supervised learning is better on CIFAR-100.

Pretext	Downstream	Self-Supervised Pretext			Supervised Pretext		
		SimCLR	N-pair	i-MixUp	SimCLR	N-pair	i-MixUp
CIFAR-10	CIFAR-10	92.5 $\pm$ 0.1	92.4 $\pm$ 0.1	**94.8** $\pm$ 0.2	95.6 $\pm$ 0.3	95.7 $\pm$ 0.1	**97.0** $\pm$ 0.1
	CIFAR-100	60.0 $\pm$ 0.2	60.2 $\pm$ 0.3	**63.3** $\pm$ 0.2	58.6 $\pm$ 0.2	**58.9** $\pm$ 0.5	57.8 $\pm$ 0.6
CIFAR-100	CIFAR-10	84.4 $\pm$ 0.2	84.4 $\pm$ 0.2	**86.2** $\pm$ 0.2	86.5 $\pm$ 0.4	86.7 $\pm$ 0.2	**88.7** $\pm$ 0.2
	CIFAR-100	68.7 $\pm$ 0.2	68.7 $\pm$ 0.2	**72.3** $\pm$ 0.2	74.3 $\pm$ 0.2	74.6 $\pm$ 0.3	**78.4** $\pm$ 0.2

adjustment strategy in Chen et al. (2020a), we also decrease the learning rate by half (0.125 to 0.0625) when the batch size is reduced. We note that i-MixUp and i-CutMix on SimCLR take approximately 2.5 times more training time to achieve the same number of training epochs. The results are provided in Table D.1. First, the performance of i-MixUp on N-pair is comparable to i-MixUp on SimCLR, while the N-pair version is computationally more efficient. When pretext and downstream tasks share the training dataset, i-CutMix often outperforms i-MixUp, though the margin is small. However, i-CutMix shows a worse performance in transfer learning.

Table D.2 compares the performance of SimCLR, N-pair, and the N-pair version of i-MixUp for both self-supervised and supervised contrastive learning. We confirm that the N-pair formulation results in no worse performance than that of SimCLR in supervised contrastive learning as well. i-MixUp improves the performance of supervised contrastive learning from 95.7% to 97.0% on CIFAR-10, similarly to MixUp for supervised learning where it improves the performance of supervised classifier learning from 95.3% to 96.4%. On the other hand, when the pretext dataset is CIFAR-100, the performance of supervised contrastive learning is not better than that of supervised learning: MixUp improves the performance of supervised classifier learning from 78.9% to 80.3%, and i-MixUp improves the performance of supervised contrastive learning from 74.6% to 78.4%.

While supervised i-MixUp improves the classification accuracy on CIFAR-10 when trained on CIFAR-10, the representation does not transfer well to CIFAR-100, possibly due to overfitting to 10 class classification. When pretext dataset is CIFAR-100, supervised

Table D.3: Comparison of contrastive learning and i-MixUp with ResNet-50 on CIFAR-10 and 100 in terms of the Fréchet embedding distance (FED) between training and test data distribution on the embedding space, and training and test accuracy. $\uparrow$ ($\downarrow$) indicates that the higher (lower) number is the better. i-MixUp improves contrastive learning in all metrics, which shows that i-MixUp is an effective regularization method for the pretext task, such that the learned representation is more generalized.

Pretext	Downstream	FED ($\times 10^{-4}$) ($\downarrow$)		Training Acc (%) ($\uparrow$)		Test Acc (%) ($\uparrow$)	
		Contrastive	i-MixUp	Contrastive	i-MixUp	Contrastive	i-MixUp
CIFAR-10	CIFAR-10	30.0	**16.7**	**96.1**	**96.1**	92.4	**94.8**
	CIFAR-100	13.8	**7.9**	**70.7**	69.5	60.2	**63.3**
CIFAR-100	CIFAR-10	15.2	**9.7**	88.1	**88.8**	84.4	**86.2**
	CIFAR-100	30.4	**13.3**	**85.6**	79.0	68.7	**72.3**

contrastive learning shows a better performance than self-supervised contrastive learning regardless of the distribution shift, as it learns sufficiently general representation for linear classifier to work well on CIFAR-10 as well.

D.2.3 Quantitative Embedding Analysis

To estimate the quality of representation by the similarity between training and test data distribution, we measure the Fréchet embedding distance (FED): similarly to the Fréchet inception distance (FID) introduced in Heusel et al. (2017), FED is the Fréchet distance (Fréchet, 1957; Vaserstein, 1969) between the set of training and test embedding vectors under the Gaussian distribution assumption. For conciseness, let $\bar{f}_i = f(x_i)/\|f(x_i)\|$ be an ℓ_2 normalized embedding vector; we normalize embedding vectors as we do when we measure the cosine similarity. Then, with the estimated mean $m = \frac{1}{N}\sum_{i=1}^{N} \bar{f}_i$ and the estimated covariance $S = \frac{1}{N}\sum_{i=1}^{N}(\bar{f}_i - m)(\bar{f}_i - m)^\top$, the FED can be defined as

$$d^2\big((m^{\mathtt{tr}}, S^{\mathtt{tr}}), (m^{\mathtt{te}}, S^{\mathtt{te}})\big) = \|m^{\mathtt{tr}} - m^{\mathtt{te}}\|^2 + \mathrm{Tr}\big(S^{\mathtt{tr}} + S^{\mathtt{te}} - 2(S^{\mathtt{tr}}S^{\mathtt{te}})^{\frac{1}{2}}\big). \quad \text{(D.7)}$$

As shown in Table D.3, i-MixUp improves FED over contrastive learning, regardless of the distribution shift. Note that the distance is large when the training dataset of the downstream task is the same with that of the pretext task. This is because the model is overfit to the training dataset, such that the distance from the test dataset, which is unseen during training, has to be large.

Table D.3 shows that i-MixUp reduces the gap between the training and test accuracy. This implies that i-MixUp is an effective regularization method for pretext tasks, such that the learned representation is more generalizable in downstream tasks.

BIBLIOGRAPHY

BIBLIOGRAPHY

Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, 2015.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 2017.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*, 2016.

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020.

Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *NeurIPS*, 2018.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019.

Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279, 2013.

Abhijit Bendale and Terrance Boult. Towards open set deep networks. In *CVPR*, 2016.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2007.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *PAMI*, 35(8):1798–1828, 2013.

Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In *ICML*, 2017.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *ICLR*, 2019.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018.

Bibhas Chakraborty and Susan A Murphy. Dynamic treatment regimes. *Annual review of statistics and its application*, 1:447–464, 2014.

Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *CVPR*, 2016.

Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *ECCV*, 2016.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In *NIPS*, 2001.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018.

Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. In *CVPR*, 2019.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020a.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *ICML*, 2019.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. In *CVPR*, 2019a.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019b.

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

Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

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

Jia Deng, Alexander C Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010.

Jia Deng, Jonathan Krause, Alexander C Berg, and Li Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *CVPR*, 2012.

Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *ECCV*, 2014.

Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

Terrance DeVries and Graham W Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017a.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017b.

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *ICCV*, 2017.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

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

Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NeurIPS*, 2014.

Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *PAMI*, 38(9):1734–1747, 2015.

Frederik Ebert, Sudeep Dasari, Alex X Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. In *CoRL*, 2018.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *ICML*, 2018.

Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *CoRL*, 2018.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Maurice Fréchet. Sur la distance de deux lois de probabilité. *COMPTES RENDUS HEBDOMADAIRES DES SEANCES DE L ACADEMIE DES SCIENCES*, 244(6):689–692, 1957.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. DeViSE: A deep visual-semantic embedding model. In *NeurIPS*, 2013.

Yanwei Fu and Leonid Sigal. Semi-supervised vocabulary-informed learning. In *CVPR*, 2016.

Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *ICML*, 2019.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, 2016.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

Anna C Gilbert, Yi Zhang, Kibok Lee, Yuting Zhang, and Honglak Lee. Towards understanding the invertibility of convolutional neural networks. In *IJCAI*, 2017.

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

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *ICML*, 2013.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

Hongyu Guo. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *AAAI*, 2020.

Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

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

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2016.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *NeurIPS*, 2019.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *ICLR*, 2020.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *ICML*, 2017.

Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

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

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *ECCV*, 2018.

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

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

Eric Jang, Coline Devin, Vincent Vanhoucke, and Sergey Levine. Grasp2vec: Learning object representations from self-supervised grasping. In *CoRL*, 2018.

Khurram Javed and Faisal Shafait. Revisiting distillation and incremental classifier learning. In *ACCV*, 2018.

Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019.

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetful learning for domain expansion in deep neural networks. In *AAAI*, 2018.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *CVPR*, 2019.

Dahun Kim, Donghyeon Cho, Donggeun Yoo, and In So Kweon. Learning image representations by completing damaged jigsaw puzzles. In *WACV*, 2018a.

Hyo-Eun Kim, Seungwook Kim, and Jaehwan Lee. Keep and learn: Continual learning by constraining the latent space for knowledge preservation in neural networks. In *MICCAI*, 2018b.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICML*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2015.

Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization. In *NeurIPS*, 2018.

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

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *PAMI*, 36(3):453–465, 2014.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *NeurIPS*, 2007.

Kibok Lee, Kimin Lee, Kyle Min, Yuting Zhang, Jinwoo Shin, and Honglak Lee. Hierarchical novelty detection for visual object recognition. In *CVPR*, 2018a.

Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *ICCV*, 2019a.

Kibok Lee, Zhuoyuan Chen, Xinchen Yan, Raquel Urtasun, and Ersin Yumer. ShapeAdv: Generating shape-aware adversarial 3d point clouds. *arXiv preprint arXiv:2005.11626*, 2020a.

Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-MixUp: Vicinal risk minimization for contrastive representation learning. Preprint, 2020b.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *ICLR*, 2018b.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018c.

Kimin Lee, Sukmin Yun, Kibok Lee, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. In *ICML*, 2019b.

Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *ICLR*, 2020c.

Michelle A Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *ICRA*, 2019c.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NeurIPS*, 2017.

Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. *arXiv preprint arXiv:1812.09111*, 2018.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 2020.

Yingzhen Li and Yarin Gal. Dropout inference in Bayesian neural networks with alpha-divergences. In *ICML*, 2017.

Yu Li, Zhongxiao Li, Lizhong Ding, Peng Yang, Yuhui Hu, Wei Chen, and Xin Gao. Supportnet: solving catastrophic forgetting in class incremental learning with support data. *arXiv preprint arXiv:1806.02942*, 2018.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, 2016.

Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *IJCAI*, 2017.

Xialei Liu, Hao Yang, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Continual universal object detection. *arXiv preprint arXiv:2002.05347*, 2020.

David Lopez-Paz et al. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

David G Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

Thomas Lucas, Corentin Tallec, Jakob Verbeek, and Yann Ollivier. Mixed batches and symmetric discriminators for gan training. In *ICML*, 2018.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov), 2008.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *JAIR*, 61:523–562, 2018.

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

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 2018.

Larry M Manevitz and Malik Yousef. One-class svms for document classification. *JMLR*, 2(Dec):139–154, 2001.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24: 109–165, 1989.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11), 1995.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018a.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *PAMI*, 41(8):1979–1993, 2018b.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *ICCV*, 2017.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. In *ICLR*, 2019.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *ICLR*, 2018.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *ICCV*, 2017.

Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *CVPR*, 2018.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *NeurIPS*, 2018.

Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *ECCV*, 2018.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. *arXiv preprint arXiv:1909.11515*, 2019.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*, 2016.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *ICRA*, 2018.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, 2007.

Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *ICCV*, 2017.

Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. In *NeurIPS*, 2019.

Mirco Ravanelli, Jianyuan Zhong, Santiago Pascual, Pawel Swietojanski, Joao Monteiro, Jan Trmal, and Yoshua Bengio. Multi-task self-supervised learning for robust speech recognition. In *ICASSP*, 2020.

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

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

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

Xinyi Ren, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Abhishek Gupta, Aviv Tamar, and Pieter Abbeel. Domain randomization for active pose estimation. In *ICRA*, 2019.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In *NeurIPS*, 2018.

Marcus Rohrbach, Michael Stark, and Bernt Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR*, 2011.

Marcus Rohrbach, Sandra Ebert, and Bernt Schiele. Transfer learning in a transductive setting. In *NeurIPS*, 2013.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR*, 2016a.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016b.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, 2018.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018.

Joan Serra, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NeurIPS*, 2017.

Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*, 2013.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Paolo Simeone, Raúl Santos-Rodríguez, Matt McVicar, Jefrey Lijffijt, and Tijl De Bie. Hierarchical novelty detection. In *International Symposium on Intelligent Data Analysis*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *NeurIPS*, 2013.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, 2016.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1): 1929–1958, 2014.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

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

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

Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. *arXiv preprint arXiv:2004.01113*, 2020.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, 2017.

Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46, 1995.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *PAMI*, 30(11):1958–1970, 2008.

Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11(Dec):3371–3408, 2010.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, 2016.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Jason W Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD Birds 200. Technical report, California Institute of Technology, 2010.

Olivia Wiles, A Koepke, and Andrew Zisserman. Self-supervised learning of a facial attribute embedding from video. In *BMVC*, 2018.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, and Yun Fu. Incremental classifier learning with generative adversarial networks. *arXiv preprint arXiv:1802.00853*, 2018a.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018b.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning: A comprehensive evaluation of the good, the bad and the ugly. *arXiv preprint arXiv:1707.00600*, 2017.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

Xueting Yan, Ishan Misra, Abhinav Gupta, Deepti Ghadiyaram, and Dhruv Mahajan. Clusterfit: Improving generalization of visual representations. In *CVPR*, 2020.

Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In *ICCV*, 2015.

Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *CVPR*, 2019.

Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep metric learning for person reidentification. In *ICPR*, 2014.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *ICLR*, 2018.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NeurIPS*, 2014.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

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

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *ICCV*, 2019.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018a.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018b.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018c.

Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *CVPR*, 2016a.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016b.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NeurIPS*, 2015.

Yuting Zhang, Kibok Lee, and Honglak Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *ICML*, 2016c.

Hang Zhao, Xavier Puig, Bolei Zhou, Sanja Fidler, and Antonio Torralba. Open vocabulary scene parsing. In *ICCV*, 2017.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020.

Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies. In *ICLR*, 2019.