

國立臺灣大學電機資訊學院電機工程學研究所

博士論文

Graduate Institute of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Doctoral Dissertation

有限或帶噪標籤資料學習於視覺分類

Learning from Limited or Noisily-Labeled Data  
for Visual Classification

林家慶

Chia-Ching Lin

指導教授: 雷欽隆 博士

共同指導: 王鈺強 博士

Advisor: Chin-Laung Lei, Ph.D.

Co-Advisor: Yu-Chiang Frank Wang, Ph.D.

中華民國 112 年 2 月

February, 2023



國立臺灣大學博士學位論文  
口試委員會審定書

PHD DISSERTATION ACCEPTANCE CERTIFICATE  
NATIONAL TAIWAN UNIVERSITY

有限與帶噪標籤資料學習於視覺分類

Learning from Limited and Noisily-Labeled Data for Visual Classification

本論文係林家慶（學號 D05921018）在國立臺灣大學電機工程學系完成之博士學位論文，於民國 111 年 12 月 28 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Electrical Engineering on 28 December 2022 have examined a PhD dissertation entitled above presented by Chia-Ching Lin (Studen ID D05921018) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

雷欽隆 璿

(指導教授 Advisor)

王昭 雲

王勝德

林彥宇

于 鈺

陳祝嵩

齊 勇 剛

系主任 Director:

李建模





## 誌謝

本論文在雷欽隆教授與王鈺強教授的指導下完成。雷教授給予學生極大的研究自由，並且總是能夠適時地提出獨到的見解，幫助學生突破自身的盲點；王教授為電腦視覺學界之翹楚，在研究問題發想、實驗設計、論文寫作等階段，皆給予許多關鍵的建議及實質的幫助。並且總是能夠在學生研究卡關時，以創新的想法及批判性的思考，將研究導向正軌。有雷教授與王教授的大力指導與支持，本論文才得以完成。

感謝口試委員顏嗣鈞教授、王勝德教授、林彥宇教授、陳祝嵩教授、于天立教授、王銘宏教授，針對本研究提出許多寶貴的建議，使本論文得以更加成熟、完備。

感謝網路計算暨安全實驗室的同學，尤其是博士班的夥伴許之凡博士、蔡孟翰、盧淑萍、彭議霆、陳憶賢、王雅平，他們的陪伴與勉勵，使我的博士生涯增添了許多溫暖。也感謝視覺與學習實驗室的同學，尤其是博士班的戰友陳尚甫、楊福恩、黃聖喻、林棋祥，以及專題生朱信塵，與他們在實驗細節及研究方法的諸多討論，總是使我獲益良多。

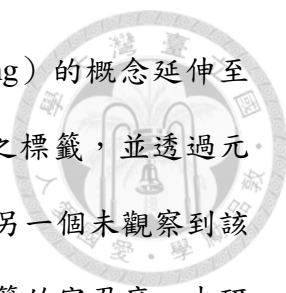
最後，特別感謝家人的諒解與支援，使我得以全力衝刺。也感謝大學同學與足球隊的好友、中研院及英業達的同仁，以及引領我進入資料科學世界的陳昇璋老師，有他們的支持與鼓勵，我才得以在漫長的博士班旅程當中堅持下去。





## 摘要

深度學習近年來於電腦視覺之多項任務中均取得突破性的成果，例如影像辨識、物件偵測、圖像語意分割等等。然而，深度學習模型的強大效能，往往來自於龐大且具有精確標註的訓練資料。受限於時間、金錢，與人力等因素，在現實世界的應用場景當中，研究人員往往必須面對不完美的訓練資料，例如數量有限或是帶有錯誤標籤的樣本。因此，如何在面對此類不完美的訓練資料時，仍然能夠訓練出可靠的深度學習模型，已然成為重要的研究主題。本研究分成兩個部分，第一部分探討有限的訓練資料，討論少樣本學習 (few-shot learning)；第二部分則是探討訓練資料數量充足，但是帶有錯誤標籤的情形，討論噪聲標籤學習 (noisy-label learning)。在少樣本學習的研究中，我們從資料擴增 (data augmentation) 的角度切入，模仿人類的學習模式：藉由觀察基礎類別 (base categories) 之大量訓練資料，學習舉一反三的能力，從而在面對新類別 (novel categories) 之少量學習樣本時，有能力幻想 (hallucinate) 出更多的新類別樣本。現有的資料幻想技術，並沒有充分利用人類學習新類別時可取得的各種先驗知識 (prior knowledge)，例如類別的語意資訊，或是基礎類別之大量訓練資料當中所隱含的豐富外觀資訊等等。此研究藉由將上述兩種資訊，藉由元學習 (meta learning) 的技巧，納入資料幻想模型的學習框架當中，並探討他們如何增進深度學習模型在少樣本學習上的表現。實驗結果顯示，我們的模型在提升少樣本類別之準確率，以及生成資料之可解釋性方面，均有比現有方法較佳的表現。在噪聲



標籤學習的研究中，我們將自監督學習 (self-supervised learning) 的概念延伸至集合的層次，刻意擾亂一個小批次 (mini-batch) 的訓練資料之標籤，並透過元學習的技巧，鼓勵模型在觀察到該標籤擾亂之後，仍然能夠與另一個未觀察到該標籤擾亂之穩定模型具有相似的表現，從而提高模型對錯誤標籤的容忍度。本研究更進一步將上述之集合層次的自監督學習框架進行延伸，發展出估計整體訓練資料當中不同類別之間的標記錯誤機率之演算法，使模型有機會針對特定標記錯誤進行補償；也發展出一種新的樣本重新加權 (sample reweighting) 演算法，自動調降標記錯誤之訓練樣本的權重，從而提升模型對於錯誤標記之訓練資料的穩健性。實驗結果顯示，我們提出的集合層次自監督學習 (set-level self-supervised learning, SLSSL) 可以有效提升模型在正確標籤之測試資料集的準確率。

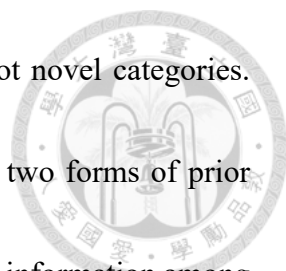
**關鍵字：**少樣本學習、噪聲標籤學習、元學習、自監督學習



# Abstract

Deep learning approaches have recently achieved remarkable progress in many computer vision tasks, such as visual classification, object detection, and semantic segmentation. However, training a deep neural network typically requires a large amount of accurately-labeled training data, which is time-consuming and labor-intensive to collect. In many real-world applications, researchers often have to build their networks based on *imperfect* training data, such as datasets with a limited number of samples or noisy labels. As a result, how to effectively learn from such imperfect datasets has become an important research topic. This study can be divided into two parts. In the first part, we focus on learning from limited training data, and discuss few-shot learning (FSL) tasks. In the second part, we focus on learning from noisily-labeled training data, which are assumed to have a sufficient amount of samples with noisy labels, and discuss noisy-label learning (NLL) tasks. In the study of FSL, we start with data augmentation approaches, in which a data *hallucinator* is built from base categories with a large number of training samples,





and then applied to generate additional training samples for few-shot novel categories. Particularly, we leverage the meta-learning technique to investigate two forms of prior knowledge to aid the construction of the hallucinator: 1) the semantic information among all categories; 2) the abundant appearance information extracted from base categories. Experiments show that, by leveraging the above prior knowledge, the proposed frameworks are able to make hallucinations that not only improve the performances of downstream FSL tasks, but also exhibit more explainable appearances or distributions than previous data hallucination works. In the study of NLL, we extend the idea of self-supervised learning to the set level, and propose the set-level self-supervised learning (SLSSL) technique. Specifically, we first corrupt the labels of a training set (i.e., mini-batch) and then obtain an *augmented* model by the meta-learning techniques. Next, we design our pretext task by imposing a consistency constraint on such augmented models to encourage the model's robustness against noisy labels. The proposed SLSSL technique can also be utilized to estimate the associated noise transition matrix to counteract the effect of noisy labels during model training, and also allows us to identify clean/noisy training samples via sample reweighting. Such SLSSL-based model training and sample reweighting algorithms can be further combined as an EM-like algorithm to boost the NLL performance. Experiments on synthetic and real-world noisily-labeled data confirm the effectiveness of our

framework.

**Keywords:** few-shot learning, noisy-label learning, meta learning, self-supervised learning







# Contents

	<b>Page</b>
口試委員會審定書	i
誌謝	iii
摘要	v
<b>Abstract</b>	<b>vii</b>
<b>Contents</b>	<b>xi</b>
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xvii</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
<b>Chapter 2 Preliminary</b>	<b>7</b>
2.1 Few-Shot Learning . . . . .	7
2.1.1 FSL Scenarios . . . . .	7
2.1.2 FSL Approaches . . . . .	10
2.1.2.1 Parameter-Level Approaches . . . . .	10
2.1.2.2 Data Hallucination Approaches . . . . .	11
2.1.2.3 Meta-Learned Hallucinator . . . . .	15
2.2 Noisy-Label Learning . . . . .	16
2.2.1 Noise Types . . . . .	16
2.2.2 NLL Approaches . . . . .	17

2.2.2.1	Parameter-Level Approaches . . . . .	18
2.2.2.2	Data-Level Approaches . . . . .	20



## **I Learning from Limited Data 21**

### **Chapter 3 Semantics-Guided Hallucination for Few-Shot Learning 23**

3.1	Overview . . . . .	23
3.2	Problem Definition . . . . .	24
3.3	Semantics-Guided Hallucination . . . . .	25
3.3.1	Generator as Data Hallucinator . . . . .	26
3.3.2	Semantics-Guided Hallucination Network . . . . .	27
3.3.3	Procedures of Training and Inference . . . . .	28
3.4	Experiments . . . . .	30
3.4.1	Datasets . . . . .	30
3.4.2	Implementation Details . . . . .	32
3.4.3	Results . . . . .	32
3.4.3.1	CIFAR-100 . . . . .	32
3.4.3.2	Animals with Attributes . . . . .	34
3.5	Summary . . . . .	35

### **Chapter 4 Feature Disentanglement based Hallucination for Few-Shot Learning 37**

4.1	Overview . . . . .	37
4.2	Problem Definition . . . . .	39
4.3	Appearance-Guided Hallucination . . . . .	41
4.3.1	Preserving Categorical Information . . . . .	42



4.3.2	Preserving Appearance Information . . . . .	45
4.3.3	Optimization Procedure . . . . .	47
4.3.4	Data Hallucination for Few-shot Learning . . . . .	48
4.4	Experiments . . . . .	49
4.4.1	Datasets . . . . .	49
4.4.2	FSL Settings and Implementation Details . . . . .	50
4.4.3	Evaluation . . . . .	55
4.4.4	Ablation Study . . . . .	59
4.5	Summary . . . . .	62

## **II Learning from Noisily-labeled Data 63**

### **Chapter 5 Set-Level Self-Supervised Learning from Noisily-Labeled Data 65**

5.1	Overview . . . . .	65
5.2	Problem Definition . . . . .	66
5.3	Set-Level Self-Supervised Learning . . . . .	67
5.3.1	SLSSL Operation . . . . .	67
5.3.2	SLSSL for Model Training . . . . .	68
5.3.3	SLSSL for Sample Reweighting . . . . .	74
5.3.4	SLSSL as an EM-like Algorithm . . . . .	77
5.4	Experiments . . . . .	78
5.4.1	Datasets . . . . .	78
5.4.2	Implementation details . . . . .	79
5.4.3	Quantitative Results . . . . .	82

5.4.4	Visualization and Analysis . . . . .	86
5.4.5	Limitations . . . . .	89
5.5	Summary . . . . .	90
<b>Chapter 6</b>	<b>Conclusion and Future Works</b>	<b>91</b>
<b>References</b>		<b>97</b>

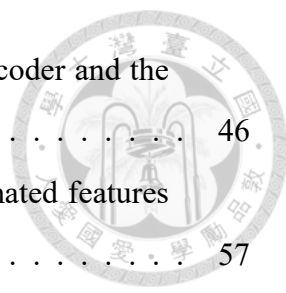




# List of Figures

2.1	Datasets considered in FSL tasks (modified from Figure 1 of [1]). . . . .	8
2.2	$N$ -way $K$ -shot scenario. . . . .	8
2.3	Multi-class few-shot classification scenario. . . . .	9
2.4	Illustration of data hallucination approaches for few-shot learning tasks (modified from Figure 1 of [2]). . . . .	11
2.5	Different types of hallucinators for few-shot learning tasks. . . . .	12
2.6	Train the hallucinator by learning to hallucinate more samples for $N$ -way $K$ -shot tasks. . . . .	15
2.7	Two kinds of instance-independent label noise. . . . .	17
2.8	Instance-dependent label noise. . . . .	18
2.9	Illustration of the loss correction method. . . . .	19
3.1	Illustration of unreasonable hallucination examples. . . . .	24
3.2	Training procedures for multi-class few-shot classification. . . . .	25
3.3	The proposed semantics-guided hallucination network (SGH-Net). . . . .	27
3.4	2D visualization of hallucinated features on selected categories of CIFAR- 100. . . . .	33
4.1	Illustration of category-specific and appearance-specific information con- tained in images. . . . .	38
4.2	Training and evaluation procedures for multi-class few-shot classification.	40
4.3	Illustration of our Feature Disentanglement and Hallucination Network (FDH-Net). . . . .	41





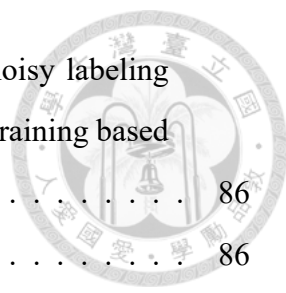
4.4	Illustration of the adversarial training of the appearance encoder and the discriminator. . . . .	46
4.5	t-SNE visualization and nearest real images of our hallucinated features on the CUB dataset. . . . .	57
4.6	Examples of hallucinated results on four benchmark datasets. . . . .	57
4.7	More data hallucination examples produced by our FDH-Net using the same seed image of school bus from <i>mini-ImageNet</i> . . . . .	58
4.8	t-SNE visualization of hallucinated features and appearance information generated by various hallucinators trained with various loss combinations of our FDH-Net. . . . .	61
5.1	Illustration of our set-level self-supervised learning (SLSSL). . . . .	67
5.2	SLSSL for model training. . . . .	70
5.3	The dirty-water analogy. . . . .	71
5.4	Example of SLSSL-based noise transition matrix estimation. . . . .	73
5.5	Good or bad sample weights. . . . .	74
5.6	SLSSL for sample reweighting. . . . .	75
5.7	The ground-truth noise transition matrix utilized to construct noisy datasets from CIFAR-10. . . . .	78
5.8	The estimated noise transition matrix for CIFAR-10 with 40% asymmetric noise. . . . .	86
5.9	The ground-truth and the estimated noise transition matrix for the original CIFAR-10 dataset without label noise. . . . .	88
5.10	Empirical distributions of sample weights derived at the end of three consecutive M-steps on CIFAR-10 with 40% asymmetric noise. . . . .	88
5.11	Learning curve on CIFAR-10 with 40% asymmetric label noise. . . . .	89



# List of Tables

2.1	Comparison of recent data hallucination approaches to FSL. . . . .	14
3.1	Results of top-5 accuracy (in percentage) on CIFAR-100 on novel classes only and on all classes. . . . .	32
3.2	Results of top-5 accuracy (in percentage) on AwA on novel classes and on all classes. . . . .	34
4.1	Performance comparison on fine-grained datasets (CUB and FLO). . . . .	55
4.2	Performance comparison on coarse-grained datasets ( <i>mini</i> -ImageMet and CIFAR-100). . . . .	55
4.3	The best set of loss weights found through validation for each dataset. . . . .	56
4.4	Ablation study for the loss functions of our proposed FDH-Net on CUB and <i>mini</i> -ImageNet datasets in the 1-shot case. . . . .	59
5.1	Comparison between K-means clustering and our EM-like iterative training strategy in SLSSL. . . . .	77
5.2	Performance comparison on CIFAR-10 across different noisy labeling schemes with NLL methods based on a single learning model. . . . .	82
5.3	Performance comparison on CIFAR-10 across different noisy labeling schemes with NLL methods adopting double models (i.e., co-training based approaches). . . . .	85
5.4	Performance comparison on CIFAR-10N with different human-labeling schemes. . . . .	85

5.5	Performance comparison on CIFAR-100 across different noisy labeling schemes with NLL methods adopting dual models (i.e., co-training based approaches). . . . .	86
5.6	Performance comparison on Clothing1M. . . . .	86





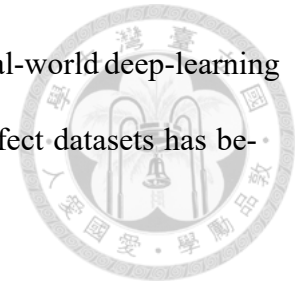
# Chapter 1

## Introduction

**Learning from imperfect data** Deep learning approaches have recently achieved remarkable progress in many computer vision tasks, such as visual classification [3–5], object detection [6–9], and semantic segmentation [10–12]. However, deep neural networks (DNNs) are notorious for being data-hungry [13, 14]; training a DNN typically requires a large amount of accurately-labeled data. As collecting a large-scale training dataset with high-quality annotations is often time-consuming and labor-intensive, deep learning practitioners may have to build their networks based on *limited* training data, making their models prone to overfitting to training samples and less generalizable at inference time [15].

Other researchers seek for alternative sources of large-scale labeled data that are less expensive, such as querying from search engines [16] or collecting social media images [17]. However, the label quality in such low-cost datasets is inevitably *low*; they often consist of samples with noisy labels. It has been shown in recent studies [18, 19] that DNNs can easily overfit to noisy labels and perform poorly on cleanly-labeled test data.

As limited and noisily-labeled datasets are ubiquitous in many real-world deep-learning based visual applications, how to effectively learn from such imperfect datasets has become an important research topic in recent years [15, 20–22].



**Human’s learning abilities – prior knowledge and advanced learning paradigms** We human beings, on the other hand, are naturally good at learning from limited or noisy supervision. For example, kids at an early age are generally able to recognize novel species of animal based on only few (sometimes even one) images. Such ability basically comes from various kinds of *prior knowledge* accumulated in one’s daily life [15], including a relatively large number of images from similar categories that one has ever seen, as well as the semantic relationship between seen and newly-encountered categories.

In addition to prior knowledge, humans can also learn from limited or noisily-labeled data through advanced learning paradigms beyond standard supervised approaches. For example, by inspecting previous learning experiences, a student can often learn novel skills more efficiently, even from partial or incomplete curricula. Such an ability of *learning to learn* motivates the research of *meta learning* [23], which aims at extracting *meta knowledge* across multiple related *base* tasks to guide the learning of *novel* tasks with limited or noisily-labeled data.

Another important human-like learning paradigm is to obtain a general understanding of the data by solving self-defined tasks. For example, by augmenting an image into two different views and defining them as a positive pair, the learner observing these two views must establish a basic perception of the image content in order to judge that they actually come from the same source image. Such an ability of *learning from intrinsic data structures* motivates the research of *self-supervised learning* [24], which aims at ex-

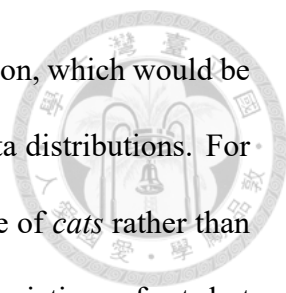
tracting *general-purposed knowledge* by defining *pretext* tasks with the supervisory signal generated from the data themselves without additional annotations.



In this dissertation, we aim at exploring the above learning paradigms exhibited in human learning scenarios and shift them to the field of deep learning for computer vision. Our goal is to design deep learning frameworks that build robust models from limited or noisily-labeled training data, with a special focus on visual classification tasks. This dissertation can be separated into two parts as follows.

**Part I: Learning from limited data – few-shot learning** In the first part of this dissertation, we focus on learning from limited training data and address few-shot learning (FSL) tasks [15]. Specifically, we focus on a practical yet challenging *multi-class few-shot classification* scenario [1], in which the training data are divided into two subsets with disjoint sets of categories: 1) a *base* dataset, with each base category containing a sufficient amount of training samples; and 2) a *novel* dataset, with only a limited amount of training samples for each novel category. The goal is to build a model from both base and novel datasets that can be used to predict the label of an unseen test sample from the joint label space.

Based on this setting, we focus on the data hallucination approach, which aims at learning a data hallucinator from the base dataset that can be used to generate additional training samples for novel categories. Particularly, we investigate two forms of prior knowledge to facilitate the construction of the hallucinator. First, we leverage the *semantic* information among all categories, such as semantic similarities based on word vectors of label names or class hierarchies. By exploiting such information in the data hallucination process and utilizing meta-learning techniques, the trained hallucinator would be encour-

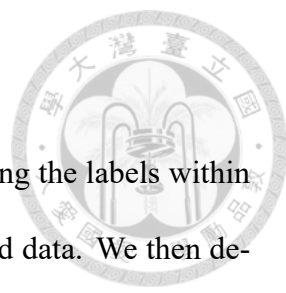


aged to generate data that exhibit *semantics-oriented* modes of variation, which would be more *reasonable* in the sense that similar categories share similar data distributions. For example, *dogs* would share similar poses and backgrounds with those of *cats* rather than *birds*, and hence the hallucinated dogs should refer to the intra-class variations of cats but not birds. Experiments show that our method quantitatively and qualitatively performed favorably against baseline and recent hallucination approaches.

Second, for FSL tasks without semantic information, we propose to leverage the abundant *appearance* information extracted from images of base categories, and apply a feature disentanglement technique to generate more diverse hallucinations for novel categories. The proposed framework, also optimized in a meta-learning manner, is able to fully exploit the base dataset to make hallucinations that are not only more *useful* for downstream FSL tasks, but also more *explainable* in the sense that every hallucinated feature of novel categories now has explicit appearance guidance from one of the base categories. Extensive experiments on both fine-grained and coarse-grained datasets confirm the effectiveness of the proposed framework.

**Part II: Learning from noisily-labeled data – noisy-label learning** In the second part of this dissertation, we focus on learning from noisily-labeled training data, and address noisy-label learning (NLL) tasks [22]. In NLL, we are given a sufficient amount of training data, with an unknown portion of samples being mislabeled based on noise processes that are also unknown. Our goal is to derive a robust model from the noisy training data that can still perform well on a *clean* test dataset. To achieve this goal, we extend the idea of self-supervised learning to the *set* level, and propose a novel set-level self-supervised learning (SLSSL) technique to enhance the robustness of the model against various label

noises.



Specifically, we first manipulate data at the set level by corrupting the labels within each mini-batch of training samples for mimicking the noisily-labeled data. We then derive a meta-learned model by applying single-step gradient descent to the learning model using the corrupted mini-batch as a unique way to *augment* the model. Next, we design the pretext tasks with self-supervisory guidance by enforcing the prediction consistency across the resulting meta-learned models, with an aim to encourage the model to exhibit sufficient robustness against label noises. Different from traditional *instance-based* self-supervised learning techniques that learn from intrinsic data structures without accessing their labels, the proposed SLSSL further takes *noisy supervision* into account when designing the pretext tasks. The proposed SLSSL can also be utilized as a building block to develop advanced algorithms for NLL tasks such as estimation of noise transition matrix and sample reweighting, which can be further combined as an Expectation-Maximization (EM)-like iterative training strategy. Experiments on synthetic and real-world noisily-labeled datasets confirm the effectiveness of the proposed framework.

The rest of this dissertation is organized as follows. In Chapter 2, we provide essential backgrounds and discuss important previous works related to the considered research topics of FSL and NLL. In Chapter 3 and Chapter 4, we propose two novel hallucination frameworks to address FSL tasks with and without side semantic information, respectively. In Chapter 5, we propose a novel set-level self-supervised learning technique to deal with NLL. Finally, we conclude our studies and suggest potential future research directions in Chapter 6.







# Chapter 2

## Preliminary

In this chapter, we provide essential backgrounds and discuss important previous works related to our studies. For clarity, the chapter is divided into two sections: few-shot learning (FSL) and noisy-label learning (NLL).

### 2.1 Few-Shot Learning

#### 2.1.1 FSL Scenarios

In FSL, one typically considers two datasets with disjoint sets of categories (as illustrated in Figure 2.1): 1) the *base* dataset  $\mathcal{D}_{base}$  of base categories  $\mathcal{C}_{base}$ , each containing a sufficient amount of samples; and 2) the *novel* dataset  $\mathcal{D}_{novel}$  of novel categories  $\mathcal{C}_{novel}$ , each containing a limited amount of samples. Based on this setting, two kinds of FSL scenarios are discussed in the literature: 1)  $N$ -way  $K$ -shot classification; and 2) multi-class few-shot classification, as will be detailed below.

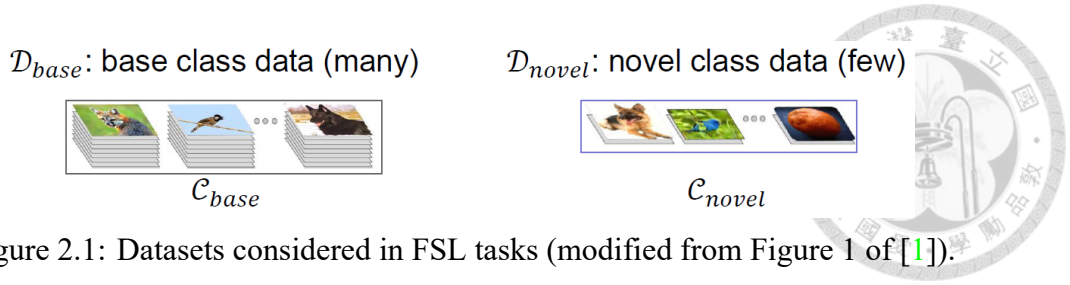


Figure 2.1: Datasets considered in FSL tasks (modified from Figure 1 of [1]).

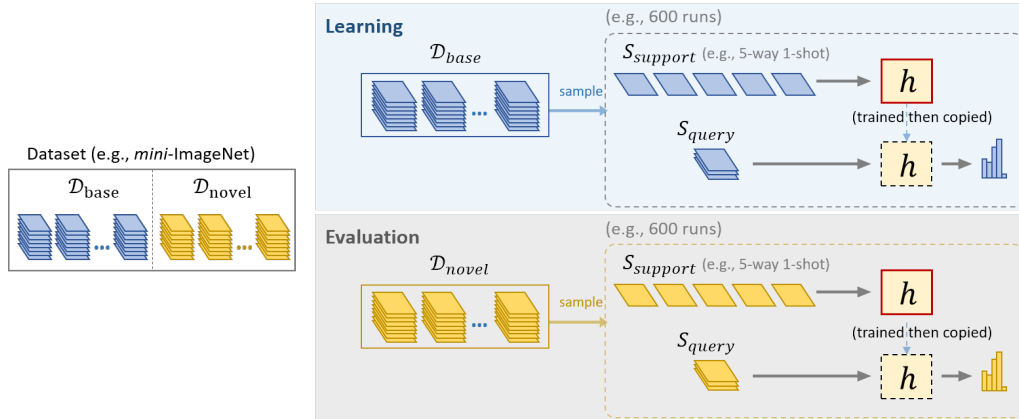


Figure 2.2:  $N$ -way  $K$ -shot scenario.

**$N$ -way  $K$ -shot classification** As illustrated in Figure 2.2, a typical  $N$ -way  $K$ -shot FSL scenario can be divided into a learning phase and an evaluation phase, with both phases involving a large number of small FSL tasks called *episodes* [25]. In each  $N$ -way  $K$ -shot episode, the learner is given a small training dataset (i.e., the *support* set  $S_{support}$ ), which contains  $K$  training samples (along with their labels) for each of the considered  $N$  categories. Based on  $S_{support}$ , the learner is then asked to build an FSL model  $h$  that can be used to classify the samples in a test dataset (i.e., the *query* set  $S_{query}$ ), which contains test samples for each of the considered  $N$  categories. Note that the ground-truth labels for samples in  $S_{query}$  are still available within each episode, but are used to evaluate the performance of  $h$  only. The shot number  $K$  in  $S_{support}$  is often limited (typically 1, 2, or 5), making each episode a challenging FSL task.

During the learning phase, the learner observes a large number of *base* episodes sampled from  $D_{base}$ , with the support set  $S_{support}$  and the query set  $S_{query}$  made by randomly sampling  $N$  categories from  $C_{base}$ . On the other hand, the learner needs to address *novel*

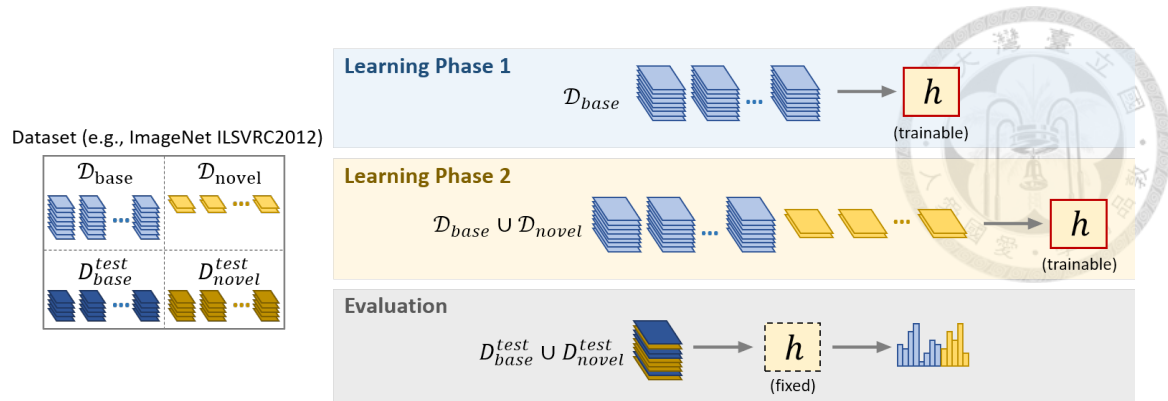


Figure 2.3: Multi-class few-shot classification scenario.

episodes with  $S_{support}$  and  $S_{query}$  made from  $\mathcal{D}_{novel}$  in a similar manner during the evaluation phase. The goal of the learner is to learn to tackle novel episodes by observing a large number of base episodes (i.e., *learning to learn* from limited data). Such a meta-learning paradigm [26, 27] plays an important role in a variety of FSL approaches, as will be detailed in Section 2.1.2 below.

**Multi-class few-shot classification** Different from the above  $N$ -way  $K$ -shot FSL scenario that focuses on small learning episodes, the goal in multi-class few-shot classification is to build a single model that can be used to predict the label of a test sample from the joint label space  $\mathcal{C}_{base} \cup \mathcal{C}_{novel}$ , as illustrated in Figure 2.3. One possible application, as noted in [1], is to deploy a recognition system that has been trained to recognize a fixed number of base categories. The recognition system deployed in the wild may then encounter novel categories with very little training data that also need to be recognized.

To achieve this goal, one typically considers two learning phases and one evaluation phase. In the first learning phase, the model  $h$  is trained on  $\mathcal{D}_{base}$  only, mimicking the scenario of a recognition system trained under fixed laboratory conditions. In the second learning phase,  $\mathcal{D}_{novel}$  with few training samples are also available, representing the novel data encountered in the wild. After learning,  $h$  is then expected to obtain the ability in

recognizing samples from both  $\mathcal{C}_{base}$  and  $\mathcal{C}_{novel}$  during the evaluation phase.



## 2.1.2 FSL Approaches

Recent approaches to FSL can be divided into two groups. The first group deals with FSL tasks at the parameter level, aiming at regularizing the model such that it is able to learn better from limited data. The second group addresses FSL at the data level, aiming at generating more training samples for those data-scarce categories.

### 2.1.2.1 Parameter-Level Approaches

**Transfer learning** One straightforward parameter-level approach to FSL is based on *transfer learning*, which focuses on learning a backbone from  $\mathcal{D}_{base}$ , followed by training or fine-tuning a linear classifier on  $\mathcal{D}_{novel}$  based on the learned representation [28–31]. Particularly, it has been shown in [30] that a deeper backbone with an increased model capacity significantly reduces the performance gaps among different parameter-level approaches to FSL, indicating the importance of good representations in dealing with FSL tasks.

**Meta learning** Recently, a more sophisticated learning paradigm called *meta learning* has gained much attention in the field of FSL. By advancing the idea of learning-to-learn [26], meta-learning approaches first learn a task-agnostic mechanism across a large number of base tasks, and then use the learned mechanism to guide a classifier to fast adapt to novel tasks. One common form of the mechanism is a proper metric space, on which non-parametric algorithms (e.g., nearest neighbor classifiers [25, 32]) or simple models (e.g., linear classifiers [33, 34]) can be readily adapted to classify novel instances. Other

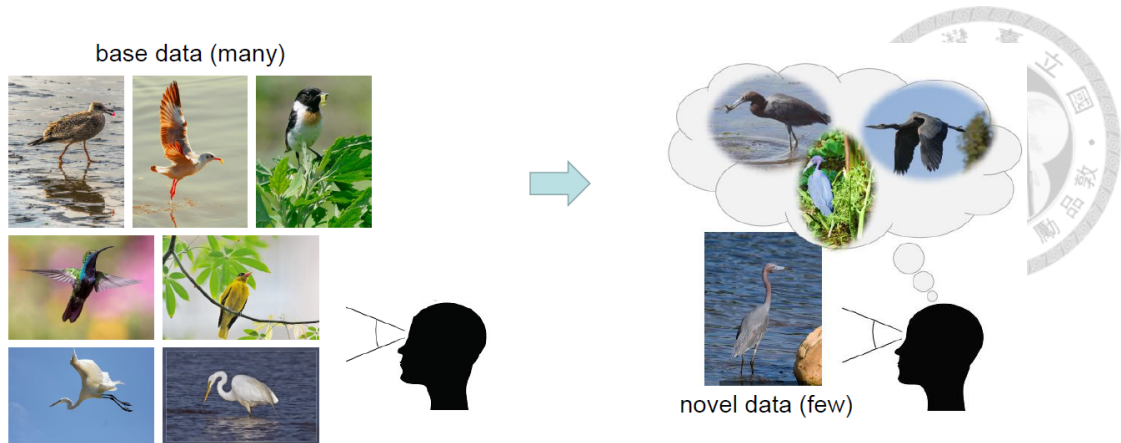


Figure 2.4: Illustration of data hallucination approaches for few-shot learning tasks (modified from Figure 1 of [2]).

forms of the mechanism include strategies for model optimization [35], model initialization [27], and memory-based operations [36].

**Remarks** Models in parameter-level approaches are typically trained and evaluated by the aforementioned  $N$ -way  $K$ -shot scheme, and are generally not applicable to the multi-class few-shot classification scenario, in which one wants to recognize both base and novel categories. Furthermore, most parameter-level FSL approaches consider a smaller number of ways (e.g.,  $N = 5$ ) compared to the entire set of novel categories of interest (e.g., 20 novel categories in *mini-ImageNet*). A recent work of Meta-Dataset [37] tackles this problem with a larger  $N$  of 50, which is still smaller than the number of the novel categories when one considers larger datasets such as ImageNet (ILSVRC-2012) [38] and Omniglot [39], which contain 1,000 and 1,623 categories, respectively.

### 2.1.1.2.2 Data Hallucination Approaches

The basic idea of data hallucination approaches is to first learn *intra-class variation* by observing a large number of samples from each of the base category in  $\mathcal{C}_{base}$ , and then apply the learned knowledge to imagine additional data for each of the novel category in

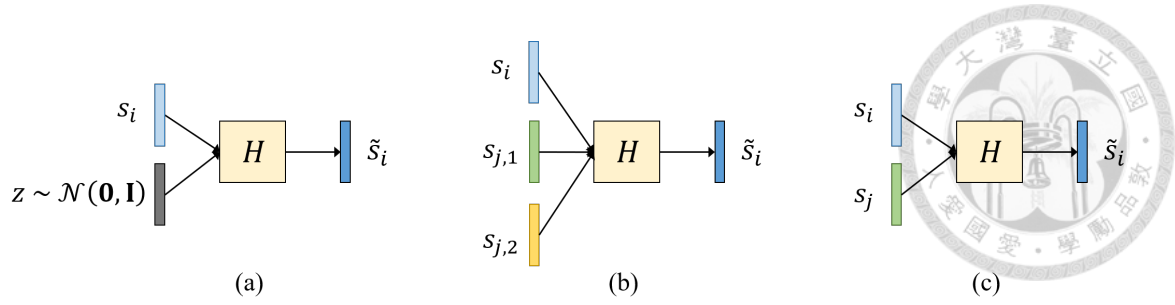


Figure 2.5: Different types of hallucinators for few-shot learning tasks.

$\mathcal{C}_{novel}$ . Such a strategy resembles human learning strategies based on prior knowledge, as illustrated in Figure 2.4.

Regarding the source of intra-class variation and input-output relationship of the generative model, existing hallucination techniques can be generally divided into three types, as illustrated in Figure 2.5 and summarized below.

**Noise-based hallucination** The first type implements the hallucinator as the generator in a conditional GAN [40–42], as shown in Figure 2.5(a), in which the hallucinator  $H$  takes a *seed* instance  $s_i$  from the  $i$ -th category as the condition, and makes hallucination  $\tilde{s}_i$  by *augmenting*  $s_i$ , with intra-class variation modeled and controlled by randomly-sampled Gaussian noise vectors  $z$ . In cGAN [2], a noise-based hallucinator is trained jointly with a meta-learning module (e.g., a prototypical network [32]) in an end-to-end manner for preserving the category of the hallucinated features. Based on cGAN, AFHN [43] proposes to add an adversarial loss to encourage the hallucinated outputs to be more realistic, and an anti-collapse regularizer to make them distributed more diverse. While also leveraging semantic information, dual-TriNet [44] proposes to build an autoencoder to map visual features into a semantic space, on which a large amount of noise-augmented semantic vectors can be sampled and then mapped back into the visual feature space. Despite these enhancements, it is still challenging for such noise-based hallucinators to capture sufficient

intra-class variation from base categories since they did not explicitly take any appearance guidance from the base data when hallucinating novel categories.



**Analogy-based hallucination** In order to effectively transfer intra-class variation from base categories to novel ones, the second type proposes an analogy-based hallucinator [1, 45]. As shown in Figure 2.5(b), analogy-based hallucinator aims at transferring the difference between a *pair* of training instances from a base category ( $s_{j,1}$  and  $s_{j,2}$ ) to the seed instance  $s_i$ , and generate a hallucinated sample  $\tilde{s}_i$  such that the difference between  $s_{j,1}$  and  $s_{j,2}$  would be preserved between  $s_i$  and  $\tilde{s}_i$ . For example, let  $s_i$  represent a *complete* apple, and let  $s_{j,1}$  and  $s_{j,2}$  represent a *complete* orange and a *sliced* orange, respectively. By taking  $s_i$ ,  $s_{j,1}$  and  $s_{j,2}$  as inputs, the trained hallucinator  $H$  would produce a hallucination  $\tilde{s}_i$  that represents a *sliced* apple. To achieve this goal, these methods often involve heavily heuristic design to collect feasible training data and are thus limited to finite modes of intra-class variation. More recently, [46] proposes to train an analogy-based hallucinator jointly with a meta-learning module in an end-to-end manner, without the need to indicate the specific hallucination outputs. However, it is not guaranteed that the difference sampled from arbitrary pair of base instances could be appropriately applied to the seed sample. For example, suppose that the sampled difference between  $s_{j,1}$  and  $s_{j,2}$  represents the transformation that makes a *perching* bird become a *flying* bird, while the seed instance  $s_i$  already represents a *flying* bird. In this case, it would be not clear what the hallucinated output  $\tilde{s}_i$  represents, as the concept of *a flying bird becomes flying* is not well-defined.

**Example-Guided hallucination** The third type directly takes examples from base categories and makes hallucination by combining the seed instance  $s_i$  with these base instances  $s_j$  in a structured way, as shown in Figure 2.5(c). Inspired by unpaired image-to-



Table 2.1: Comparison of recent data hallucination approaches to FSL.

	cGAN [2]	dual TriNet [44]	AFHN [43]	DTN [46]	IDeMe-Net [50]	SalNet [52]
Explicitly refer to base samples	-	-	-	✓	✓	✓
Free from semantic information	✓	-	✓	✓	✓	✓
Mode-collapse alleviation	-	-	✓	✓	✓	✓
Explainable hallucination	-	-	-	-	-	✓
Feature disentanglement	-	-	-	-	-	✓
Train from scratch	✓	✓	✓	✓	✓	-

image translation [47, 48], the authors of [49] propose to translate a base feature vector into the novel category of interest through adversarial training with cycle-consistency and covariance-preserving objectives. [50] proposes to fuse a seed image with other reference images in the patch level to make collage-style deformed images. Similarly, [51] utilizes an off-the-shelf image generator to reconstruct the seed image, and then fuses the seed image with its own reconstruction in the patch level. The most related work to ours is [52], which proposes to first segment the foreground and background components of two images using a pre-trained saliency network, and make hallucinations by swapping the backgrounds.

Compared to other types of hallucination techniques, these example-guided hallucinators are shown to produce more diverse outputs by explicitly taking appearance guidances from base categories, yet the drawback is that the referred instances are required to be visually or semantically similar to the seed instance in order to make feasible hallucinations. Furthermore, techniques based on patch-level combinations such as [50, 51] might fail to generate visually realistic hallucinations, making their improvements on FSL performances less explainable. As for the foreground/background swapping method [52], a pre-trained saliency network is needed, and the diversity of hallucination is limited to different foreground/background combinations. To sum up, existing example-guided methods either involve complex network architectures or optimization challenges, and are unable to produce realistic or explainable hallucinations.

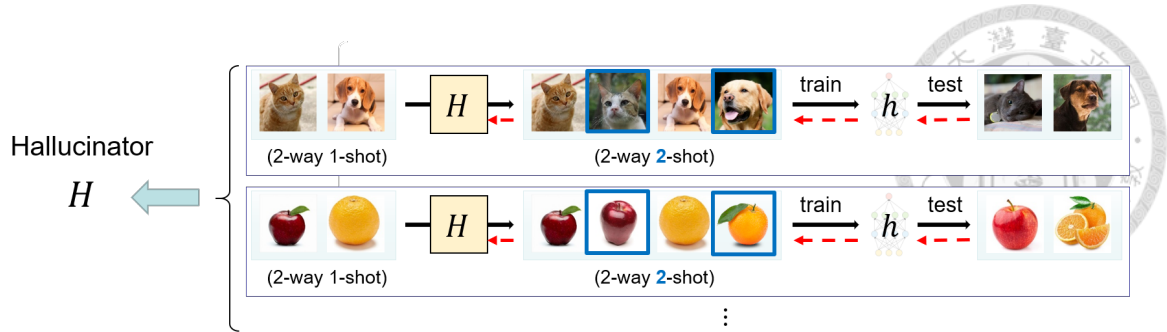


Figure 2.6: Train the hallucinator by learning to hallucinate more samples for  $N$ -way  $K$ -shot tasks.

In Table 2.1, we summarize and compare recent data hallucination works in different aspects, including claimed effectiveness and training requirements.

### 2.1.2.3 Meta-Learned Hallucinator

In this dissertation, we address FSL at the data level by hallucination techniques, and thus our modeling goal is different from parameter-level approaches. Specifically, all parameter-level approaches seek a model that *reduces* intra-class variation among base samples for better representations that can generalize to novel categories. Instead, data-level approaches seek a model that *learns* intra-class variation from base samples in order to hallucinate additional novel samples to alleviate the data scarcity problem. As a result, we follow most data-hallucination approaches to FSL [1, 2, 49, 50] and adopt the multi-class few-shot classification tasks to evaluate our framework.

It is worth noting that, for training feasibility, we still follow [1, 2, 49, 50] and train our hallucinators using a large number of  $N$ -way  $K$ -shot tasks made from  $\mathcal{D}_{base}$ . Specifically, in each  $N$ -way  $K$ -shot episode, we first have the hallucinator  $H$  *augment* the support set  $S_{support}$  by generating additional training data for each of the considered base categories. We then use the augmented support set to build the FSL model  $h$ , and evaluate the performance of  $h$  using the query set  $S_{query}$ . Intuitively, for  $h$  to perform well on

$S_{query}$ , the additional training data generated by  $H$  must be of high quality and *useful* for such classification tasks. We thus optimize  $H$  by minimizing the classification loss of  $h$ , as illustrated in Figure 2.6. The hallucinator  $H$  trained on  $\mathcal{D}_{base}$  in the above manner is used to generate additional training data for  $\mathcal{D}_{novel}$ , as will be detailed later in Chapter 3 and Chapter 4.

## 2.2 Noisy-Label Learning

### 2.2.1 Noise Types

For visual classification tasks, two types of label noise are commonly considered and discussed in the literature [22, 53]: 1) instance-independent noise (IIN); 2) instance-dependent noise (IDN). For datasets with IIN, the labels are assumed to be corrupted according to a *noise transition matrix*, with the  $(i, j)$ -th entry representing the probability of a sample with ground-truth label  $i$  being mislabelled to label  $j$ . Note that the probability values depend on the class pair only, regardless of the image *content* of the sample. Two kinds of IIN are commonly considered in existing NLL approaches: 1) class-independent (symmetric) noise; and 2) class-dependent (asymmetric) noise, as illustrated in Figure 2.7. In the figure, we start from the cleanly-labeled dataset on the left. In the middle, two kinds of IIN are injected based on different 3-by-3 matrices. The resulting noisily-labeled datasets are shown on the right, with red circles representing samples with wrong labels.

The noise transition matrix is assumed to be unknown. Nevertheless, various NLL approaches focused on estimating it to counteract the noise effect during training based on the above symmetric/asymmetric assumptions, as will be detailed in Chapter 5.

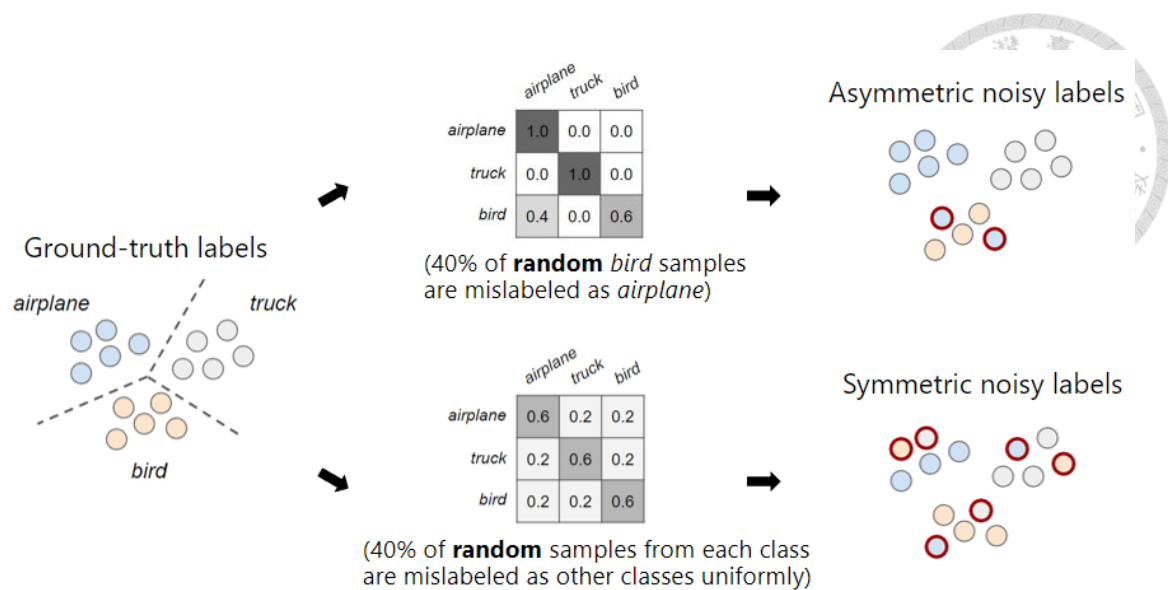


Figure 2.7: Two kinds of instance-independent label noise.

On the other hand, for a sample in datasets with IDN, the probability of being labeled as any category depends on its image content, as illustrated in Figure 2.8. In the figure, we again start from the cleanly-labeled dataset on the left, with a ground-truth decision boundary also shown between samples of *airplane* and *bird* as the dashed line. On the right, we show that samples near the ground-truth decision boundary are generally more ambiguous and difficult for an annotator to correctly annotate, resulting in samples with wrong labels in red circles.

**Remarks** In this dissertation, we focus on IIN synthesized from cleanly-labeled datasets. However, we also conduct experiments on real-world noisily-labeled datasets with more complicated noise patterns than either IIN or IDN, as will be detailed in Chapter 5.

## 2.2.2 NLL Approaches

Similar to the field of FSL, existing NLL methods can also be categorized into two groups [22]. The first group of parameter-level approaches focuses on regularizing the

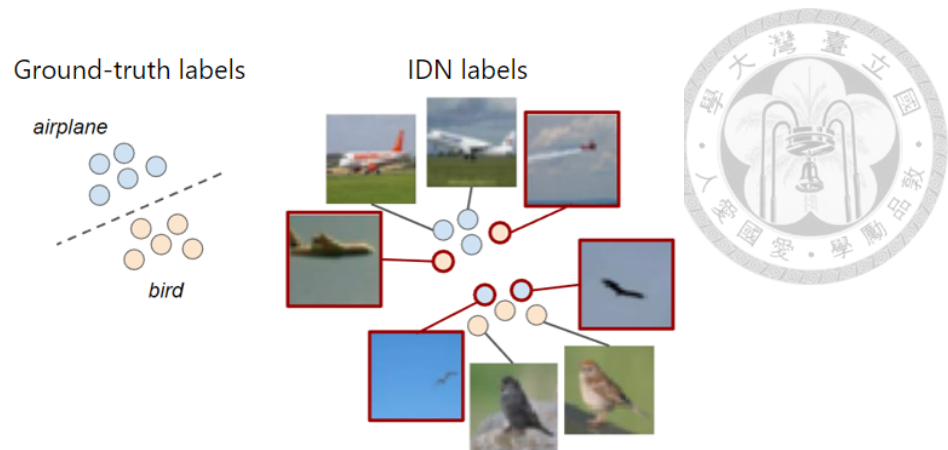


Figure 2.8: Instance-dependent label noise.

model to make it more robust against noisy labels in the training dataset. The second group deals with NLL at the data level by designing various sample selection or reweighting algorithms.

### 2.2.2.1 Parameter-Level Approaches

**Robust loss functions** To prevent DNNs from memorizing noisy labels, various robust loss functions [54–57] have been proposed to replace the widely-used categorical cross-entropy (CCE) loss. The basic idea is to design loss functions that are noise-tolerant during training under certain noise types. However, existing methods generally address simple tasks with a smaller number of categories or lower noise rates [22], making them less effective in practice.

**Loss correction** A number of NLL works [58–63] focus on estimating the class-wise noise transition matrix of noisy training data, which describes the relationships between noisy labels and their ground-truth ones, as described in Section 2.2.1. The estimated noise transition matrix is typically applied to refine the model output before computing the classification loss using the given label to counteract its noise effect. It has been shown

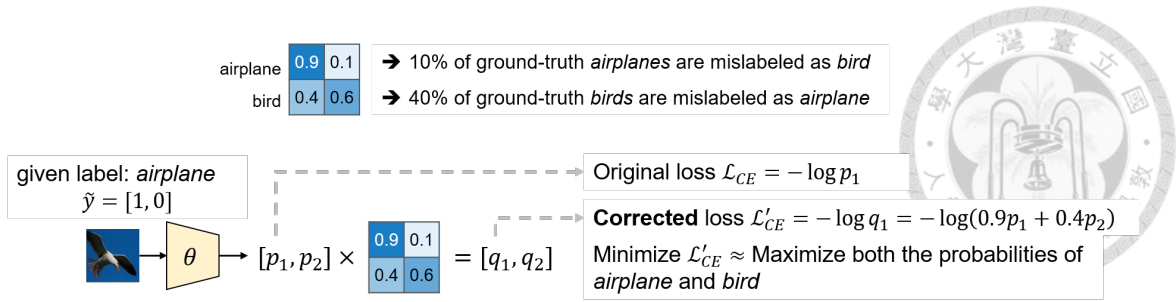


Figure 2.9: Illustration of the loss correction method.

in [59] and also illustrated in Figure 2.9 that minimizing such corrected loss toward noisy labels is equivalent to optimizing the DNN toward the ground-truth labels. Recently, Meta Loss Correction (MLC) [62] proposed to estimate the noise transition matrix of the noisy training dataset based on a small clean subset through a meta-learning technique. Despite their theoretical foundation, how to accurately estimate the noise transition matrix remains a challenging problem, especially when no clean training/validation sets are available.

**Self-supervised learning** Self-supervised learning (SSL) has been recently considered for approaching NLL tasks [64–67]. Most existing works adopt pretext tasks that are designed for SSL on *unlabeled* data, such as predicting image rotations [68] or contrastive-based instance discrimination [69–71], without utilizing the structural information contained in noisy labels for advanced self-supervisory signals for NLL. Recently, [72] propose Meta-Learning based Noise-Tolerant (MLNT) training, in which the noisy labels are further corrupted multiple times, and a consistency constraint is designed to regularize the model toward robustness under such label corruptions. This consistency constraint can be regarded as a self-supervisory signal that takes noisy labels into account. However, as MLNT chooses to corrupt labels *randomly*, the pretext task is limited to model regularization, not able to identify data with noisy labels as sample selection methods do. As introduced in the next section, our proposed strategy can be applied for improving model robustness and estimating label confidence (i.e., sample reweighting), and our experimen-

tal results would verify the effectiveness and applicability of our approach.



### 2.2.2.2 Data-Level Approaches

**Sample selection** Selecting clean training samples according to their label confidence scores is another alternative for solving NLL tasks. In co-training based approaches [73–77], two DNNs are trained in a collaborative manner, with each model being trained using clean samples selected by its peer model. Instead of maintaining multiple models, multi-round training approaches [78–80] propose to train a single model for multiple rounds and gradually increase the set of selected clean training samples.

**Sample reweighting** Instead of training the models on selected clean samples only, a number of works choose to conduct sample reweighting [81–87] before or during the model training process as a *soft* version of sample selection to fully utilize the training samples. More recently, [88–91] treat the selected clean and noisy samples as labeled and unlabeled data, respectively. They then leverage state-of-the-art semi-supervised learning techniques for training the learning models. While achieving promising results, most of the above methods perform sample selection based on the assumption that samples with small classification losses are with high probabilities for the assigned labels. This might not always hold for hard samples with correct samples to be recognized.



## **Part I**

# **Learning from Limited Data**







## Chapter 3

# Semantics-Guided Hallucination for Few-Shot Learning

### 3.1 Overview

In this chapter, we deal with FSL tasks with semantic information available. As we noted in Chapter 1 and 2, we focus on data hallucination approaches in this study. That is, we aim to build a hallucinator from base-category data that can be used to generate additional training samples for the data-scarce novel categories. We note that, most existing hallucination methods [1, 2] did not explicitly consider the relationship among different semantic concepts of categories during the hallucination process, which might make the hallucinated samples exhibiting unfeasible and inexplainable data distribution and thus limit the FSL performances. For example, hallucinating *dogs* by referring to *birds* may lead to dogs flying in the sky which would not make any sense and might not even exist in the test dataset, as illustrated in Figure 3.1.

To address the above concerns, we propose a novel Semantics-Guided Hallucination Network (SGH-Net) that exploits and incorporates semantic information into the halluci-

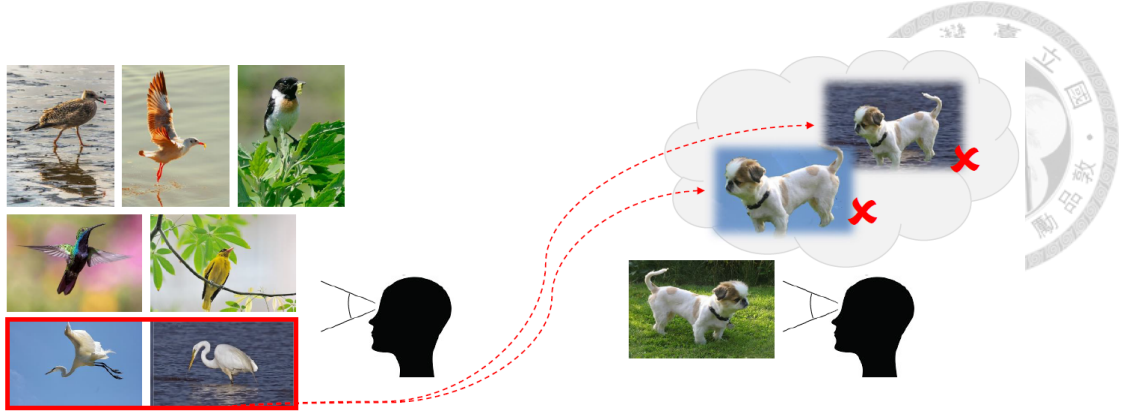


Figure 3.1: Illustration of unreasonable hallucination examples.

nation process, aiming at generating more reasonable and appropriate hallucinations for tackling FSL. The basic idea is, object categories with similar semantic concepts should possess and share similar data distributions, and thus their intra-class variations can be jointly utilized for FSL. For example, *dogs* would share similar poses and backgrounds with those of *cats* rather than *birds*. Thus, we present a framework to exploit the side semantic information and apply it to the deep hallucination process, so that the generated data samples can exhibit more semantics-oriented modes of variation. As a result, hallucination for a novel category can benefit more from the base categories with similar semantic concepts, via a single hallucinator trained on base categories only.

The main ideas of this chapter have been peer-reviewed and published in [92].

## 3.2 Problem Definition

We first define the settings and notations for the task of FSL with additional semantic information available. Suppose that we are given a training dataset  $\mathcal{D}_{base}$  of base categories  $\mathcal{C}_{base}$ , each with a sufficient amount of images, and another training dataset  $\mathcal{D}_{novel}$  of novel categories  $\mathcal{C}_{novel}$ , each with only few images available during training. Both  $\mathcal{D}_{base}$  and  $\mathcal{D}_{novel}$  consist of tuples  $\{(x_i, y_i, R_i)\}$  with  $y_i$  being the one-hot label vector of the  $i$ -th

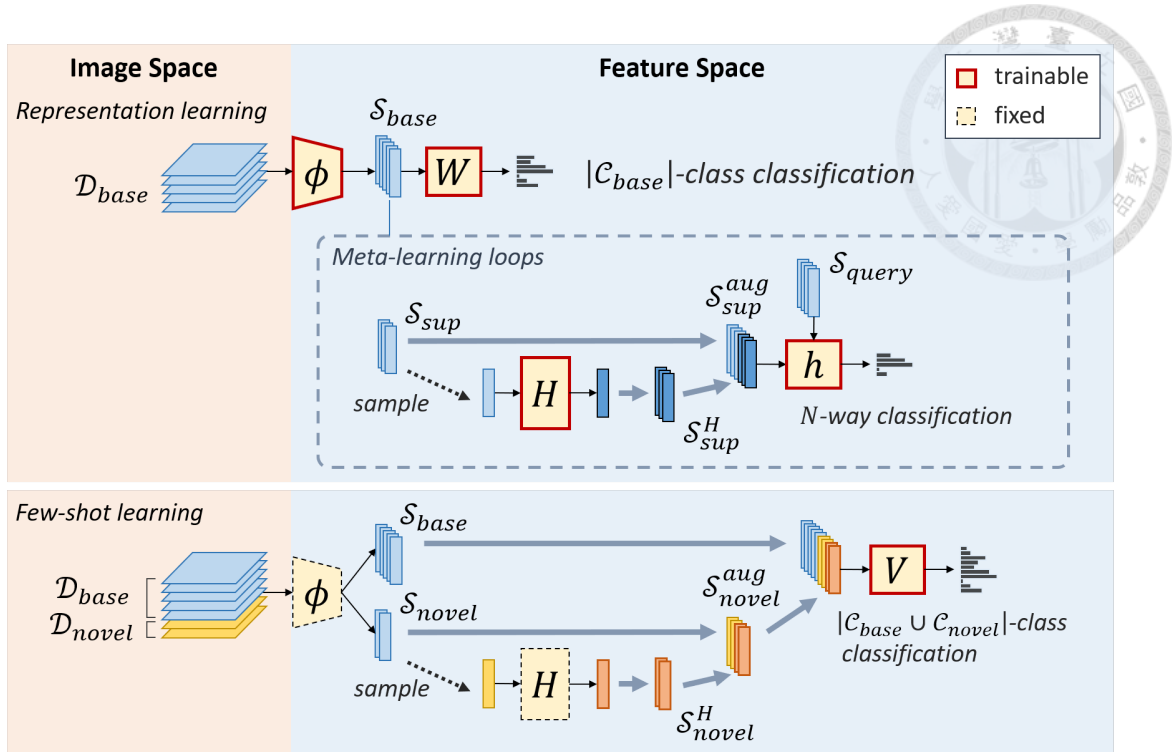
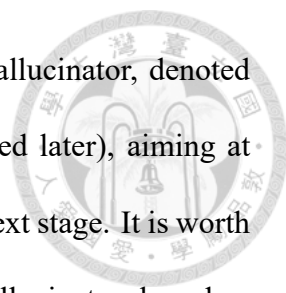


Figure 3.2: Training procedures for multi-class few-shot classification.

image  $x_i$ , and an additional vector  $R_i$  representing the semantic information associated with  $y_i$  (e.g., a word embedding vector of the associated label, or an attribute vector as those defined in the dataset of Animals with Attributes [93]). As discussed in Chapter 2, instead of applying the  $N$ -way  $K$ -shot scheme [25] as many FSL works do, we focus on a more practical yet challenging *multi-class few-shot classification* scenario introduced in [1] and adopted by [2, 49, 50, 94]. Our goal is to build a classifier that can be used to predict the label of an unseen test image  $x$  from  $\mathcal{C}_{novel} \cup \mathcal{C}_{base}$ .

### 3.3 Semantics-Guided Hallucination

Following [1, 2, 49, 50, 94], the training process of our FSL framework can be divided into two stages, as depicted in Figure 3.2. First, in the *representation learning* stage, only  $\mathcal{D}_{base}$  would be utilized. A convolutional neural network (CNN) based feature extractor, denoted as  $\phi$ , is trained on  $\mathcal{D}_{base}$  with a classifier  $W$ . A set  $\mathcal{S}_{base} = \{(s_i, y_i, R_i)\}$  is created,



with  $s_i = \phi(x_i)$  being the extracted feature of the  $i$ -th image. A hallucinator, denoted as  $H$ , is then trained on  $\mathcal{S}_{base}$  in a meta-learning manner (as detailed later), aiming at generating additional features to augment the training dataset in the next stage. It is worth noting that, the aforementioned training strategy allows us to design hallucinators based on representations derived by an off-the-shelf feature extractor, *without* the need to access the entire base image set or the infrastructure to re-train the backbone. As a result, we choose to follow previous works [1, 2, 49] and utilize such a non-end-to-end training strategy.

In the second training stage, we perform *few-shot learning* by observing both  $\mathcal{D}_{base}$  and  $\mathcal{D}_{novel}$ , with model parameters of  $\phi$  and  $H$  fixed from the previous training stage. We first create a set of novel features  $\mathcal{S}_{novel}$  from  $\mathcal{D}_{novel}$  using  $\phi$ . Based on the  $n$  feature vectors per novel category (we consider  $n \in \{1, 2, 5\}$  in this work), we have  $H$  generate additional feature vectors for each novel category (with the number set by cross-validation). Once the above data hallucination process is complete, we combine  $\mathcal{S}_{novel}$  with such generated feature data  $\mathcal{S}_{novel}^H$  for all novel categories, and use the resulting  $\mathcal{S}_{novel}^{aug} = \mathcal{S}_{novel} \cup \mathcal{S}_{novel}^H$  for training the FSL classifier  $V$ . Note that  $V$  is trained on  $\mathcal{S}_{novel}^{aug} \cup \mathcal{S}_{base}$ , and our final image classifier is obtained by concatenating  $\phi$  and  $V$ . In the following sections, we describe the design and training procedures of our data hallucinator  $H$  in detail.

### 3.3.1 Generator as Data Hallucinator

Inspired by [2], we propose a noise-based hallucinator  $H$  which can be viewed as the generator in conditional-GAN [40–42] (as depicted in Figure 2.5a), and implement it as a generative module  $G$  as depicted in Figure 3.3. The input of  $G$  consists of a noise vector  $z$  and a *seed* feature vector  $s_i$ , with  $z$  providing randomness to model the modes of variation and  $s_i$  serving as the condition to specify the category of the output  $\tilde{s}_i$ . The hallucination

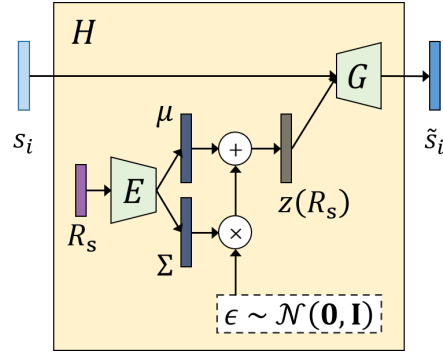


Figure 3.3: The proposed semantics-guided hallucination network (SGH-Net).

process can thus be expressed as:

$$\tilde{s}_i = G(z, s_i), \quad (3.1)$$

where the output  $\tilde{s}_i$  represents the hallucinated feature. It is worth noting that, in the framework presented in [2], the label information is not exploited in its hallucination process, and the label of  $\tilde{s}_i$  is simply assigned to be the same as  $s_i$ .

### 3.3.2 Semantics-Guided Hallucination Network

Note that in Equation (3.1), the distribution of the hallucinated output  $\tilde{s}_i$  is controlled by the noise vector  $z$ , which in [2] is simply sampled from a standard Gaussian distribution:  $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . Instead of modeling intra-class variations of different object categories by a fixed and pre-specified distribution as [2] did, we propose to incorporate semantic information into the hallucination process. This is realized by designing a *noise generator* that produces conditioned noise vectors as if they are sampled from a semantics-dependent distribution.

Specifically, suppose that the selected seed data point is  $(s_i, y_i, R_i)$ , we implement the noise generator as an encoder network  $E$  followed by a sampler function, as depicted in

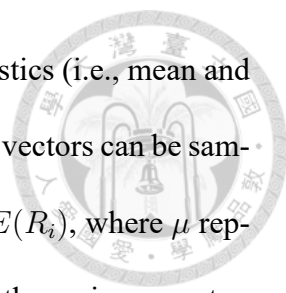


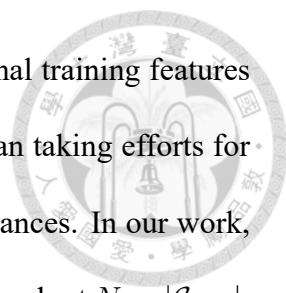
Figure 3.3. The encoder  $E$  takes  $R_i$  as its input and produces the statistics (i.e., mean and variance) needed to specify the desired distribution, from which noise vectors can be sampled. This process can be expressed as  $z \sim \mathcal{N}(\mu, \Sigma)$  with  $(\mu, \Sigma) = E(R_i)$ , where  $\mu$  represents the mean vector and  $\Sigma$  represents a diagonal matrix formed by the variance vector. Together with the generative model  $G$ , we obtain a unique data hallucination framework with our *semantics-guided hallucinator* specified as:

$$\tilde{s}_i = G(z(R_i), s_i), \quad (3.2)$$

where  $z$  is expressed explicitly as a function of the semantic input  $R_i$ . To make the whole hallucination process differentiable for gradient-based optimization, we utilize the *reparameterization* trick of variational autoencoders [95], as detailed in Figure 3.3.

### 3.3.3 Procedures of Training and Inference

**Representation learning stage** As noted at the beginning of Section 3.3, the hallucinator  $H$  is trained in a meta-learning manner in the representation learning stage. Specifically, as depicted in Figure 3.2,  $H$  is jointly trained with a meta-learning module  $h$  in a  $N$ -way  $K$ -shot fashion with a large number of episodes sampled from  $\mathcal{D}_{base}$ . Concretely, in each episode, we first randomly sample  $N$  categories from  $\mathcal{C}_{base}$  and  $K$  features per category to make a support set  $\mathcal{S}_{sup}$ . We then use  $H$  to augment  $\mathcal{S}_{sup}$ , making the hallucinated support set  $\mathcal{S}_{sup}^H$ , which is further combined with the real support set to form the *augmented* support set  $\mathcal{S}_{sup}^{aug} = \mathcal{S}_{sup} \cup \mathcal{S}_{sup}^H$ . In each episode, we also sample a set of test features from the selected  $N$  categories to make the query set  $\mathcal{S}_{query}$ . Both  $\mathcal{S}_{sup}^{aug}$  and  $\mathcal{S}_{query}$  are fed into  $h$  to produce the loss and gradient for adjusting the model parameters of  $H$  and



$h$ . By training in this way,  $H$  will be encouraged to generate additional training features that are more *useful* for  $N$ -way  $K$ -shot classification tasks, rather than taking efforts for more *realistic* features that may still fail to improve the FSL performances. In our work, we adopt the prototypical network [32] as  $h$  for its fast training speed, and set  $N = |\mathcal{C}_{base}|$ , for we observe that  $H$  trained with a larger  $N$  generally leads to better performances in the few-shot learning stage. Other hyperparameters such as the numbers of features for each base category in  $\mathcal{S}_{sup}$  and  $\mathcal{S}_{sup}^H$  are set by cross-validation.

**Few-shot learning stage** Next, in the few-shot learning stage, the trained hallucinator  $H$  is used to augment novel training features, as described in Section 3.2. Note that the semantics-dependent noise generator can also be interpreted as a strategy to regularize the manifold of the hallucinated features in the feature space, as will be shown later in the experiments. By incorporating such semantic information, proper intra-class variation would be introduced for the novel categories, and thus improved FSL performances can be expected.

**Evaluation stage** At inference time, the feature extractor  $\phi$  and FSL classifier  $V$  are concatenated to form our final image classifier, which is used to predict labels on a set of unseen test images from  $\mathcal{C}_{novel} \cup \mathcal{C}_{base}$ . Following [1] and [2], we evaluate the top-5 accuracy numbers over both  $\mathcal{C}_{novel}$  and  $\mathcal{C}_{novel} \cup \mathcal{C}_{base}$  to make sure that the performance gains on novel categories do not come at the expense of the overall performance.





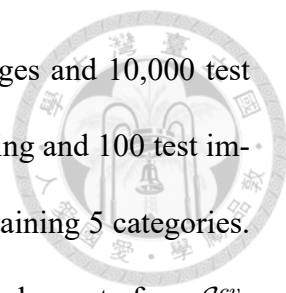
## 3.4 Experiments

### 3.4.1 Datasets

We evaluate the performance of the proposed SGH-Net following the same procedure in [1, 2] for multi-class few-shot classification Tasks. We consider CIFAR-100 [96] and Animals with Attributes (AwA) [93] datasets, with details of training and validation described below.

**General settings for training and validation** To alleviate possible overfitting problems due to data observed from novel categories, we follow [1, 2] and cross-validate the hyperparameters on a separate set of categories. Specifically, we first split all the categories of each considered dataset into two disjoint subsets, one denoted as  $\mathcal{C}^{cv}$  for cross-validation, and one denoted as  $\mathcal{C}^{fin}$  for final performance evaluation. Both subsets are further split into base and novel sets of categories to make  $\mathcal{C}^{cv} = \mathcal{C}_{base}^{cv} \cup \mathcal{C}_{novel}^{cv}$  and  $\mathcal{C}^{fin} = \mathcal{C}_{base}^{fin} \cup \mathcal{C}_{novel}^{fin}$ , respectively. For  $\mathcal{C}_{base}^{cv}$  and  $\mathcal{C}_{base}^{fin}$ , we use all training images per category to form  $D_{base}$ . As for  $\mathcal{C}_{novel}^{cv}$  and  $\mathcal{C}_{novel}^{fin}$ , we randomly sampled  $n \in \{1, 2, 5\}$  images for each category to form  $D_{novel}$ .

For each hyperparameter setting, we take training images from  $\mathcal{C}^{cv}$  into the training process to derive a classifier, with  $\mathcal{C}_{base}^{cv}$  and  $\mathcal{C}_{novel}^{cv}$  treated as  $\mathcal{C}_{base}$  and  $\mathcal{C}_{novel}$  in Section 3.2, and evaluate its performances on test images from  $\mathcal{C}_{novel}^{cv}$ . The hyperparameter setting that leads to the best performance is then used in the experiment on  $\mathcal{C}^{fin}$  for the final training process, with  $\mathcal{C}_{base}^{fin}$  and  $\mathcal{C}_{novel}^{fin}$  treated as  $\mathcal{C}_{base}$  and  $\mathcal{C}_{novel}$ . The results presented in this section are on  $\mathcal{C}^{fin}$ .



**CIFAR-100** The CIFAR-100 dataset contains 50,000 training images and 10,000 test images from 100 categories, with each category containing 500 training and 100 test images. The 100 categories are grouped into 20 superclasses, each containing 5 categories. We leverage this additional knowledge and randomly choose 10 superclasses to form  $\mathcal{C}^{cv}$ , and split each of its superclasses into two novel and three base categories, making  $\mathcal{C}_{novel}^{cv}$  and  $\mathcal{C}_{base}^{cv}$  to have sizes of 20 and 30, respectively. Similarly, the remaining 10 superclasses are used to form  $\mathcal{C}_{novel}^{fin}$  of size 20 and  $\mathcal{C}_{base}^{fin}$  of size 30. We follow the above procedure and use all 500 training images for each of the base categories in  $\mathcal{C}_{base}^{cv}$  (or  $\mathcal{C}_{base}^{fin}$ ), and randomly sample  $n \in \{1, 2, 5\}$  training images from each of the novel categories in  $\mathcal{C}_{novel}^{cv}$  (or  $\mathcal{C}_{novel}^{fin}$ ) for training our model. As for performance evaluation, we use all 100 testing images from each of both base and novel categories, and report the averaged top-5 accuracy over 20 runs, with each run observing a different samples of  $D_{novel}$ . We use spaCy [97] to obtain 300-dimensional word embedding vectors of label names as the side semantic information  $R_i$ .

**Animals with attributes** The AWA dataset [ ] contains 30,475 images from 50 species of animals. For training and validation, we randomly split all its 50 categories into four disjoint subsets to make  $\mathcal{C}_{novel}^{cv}$  of size 10,  $\mathcal{C}_{base}^{cv}$  of size 15,  $\mathcal{C}_{novel}^{fin}$  of size 10, and  $\mathcal{C}_{base}^{fin}$  of size 15. Images from each category in each of the above subsets are further split into training and testing datasets in a 4:1 ratio. Similar to CIFAR-100, we also report the averaged top-5 accuracy over 20 runs, with each run observing a different samples of  $D_{novel}$ . As for the side semantic information  $R_i$ , we directly use the 85-dimensional binary attribute vectors (e.g., color, stripe, furry, size, and habitat) provided in the dataset.

Table 3.1: Results of top-5 accuracy (in percentage) on CIFAR-100 on novel classes only and on all classes.

Test label space	$C_{novel}^{fin}$			$C_{novel}^{fin} \cup C_{base}^{fin}$		
	$n=1$	$n=2$	$n=5$	$n=1$	$n=2$	$n=5$
Baseline	28.96	46.20	60.85	67.36	73.59	80.35
cGAN [2]	47.65	59.65	70.82	70.88	77.07	81.86
cGAN* [2]	49.95	58.64	71.15	74.36	77.60	81.67
Analogy [1]	38.51	49.76	65.27	71.65	75.83	81.34
Analogy-Superclass (ours)	46.76	57.01	67.86	74.65	78.44	82.30
SGH-net (ours)	<b>55.84</b>	<b>66.28</b>	<b>74.14</b>	<b>76.68</b>	<b>80.18</b>	<b>82.95</b>

### 3.4.2 Implementation Details

We adopt a VGG-16 network as our main network, which consists of 13 convolutional layers as our feature extractor  $\phi$  (producing 512-dimensional feature vectors for each image) and a 3-layer multilayer perceptron (MLP) as our classifier  $W$ . The hallucinator  $H$  consists of an encoder network  $E$  and the generator network  $G$ . We implement  $E$  as a 2-layer MLP with double heads (one for  $\mu$  and one for  $\Sigma$ ), and  $G$  as a 3-layer MLP. The final classifier  $V$  is also implemented as a 3-layer MLP.

### 3.4.3 Results

#### 3.4.3.1 CIFAR-100

**Quantitative results** Table 3.1 lists the performance comparisons with different shot numbers  $n$ . In addition to the baseline (no hallucination) and our proposed SGH-Net, we also implement the conditional-GAN based hallucinator of [2], denoted as cGAN. For the sake of fair comparisons, we also implement the same method with one more hidden layer for the hallucinator network (denoted as cGAN\*), which utilizes roughly the same number of parameters as our proposed SGH-Net. It can be seen that our proposed SGH-Net outperforms all other methods. Most of the standard deviations are of the order of 3%

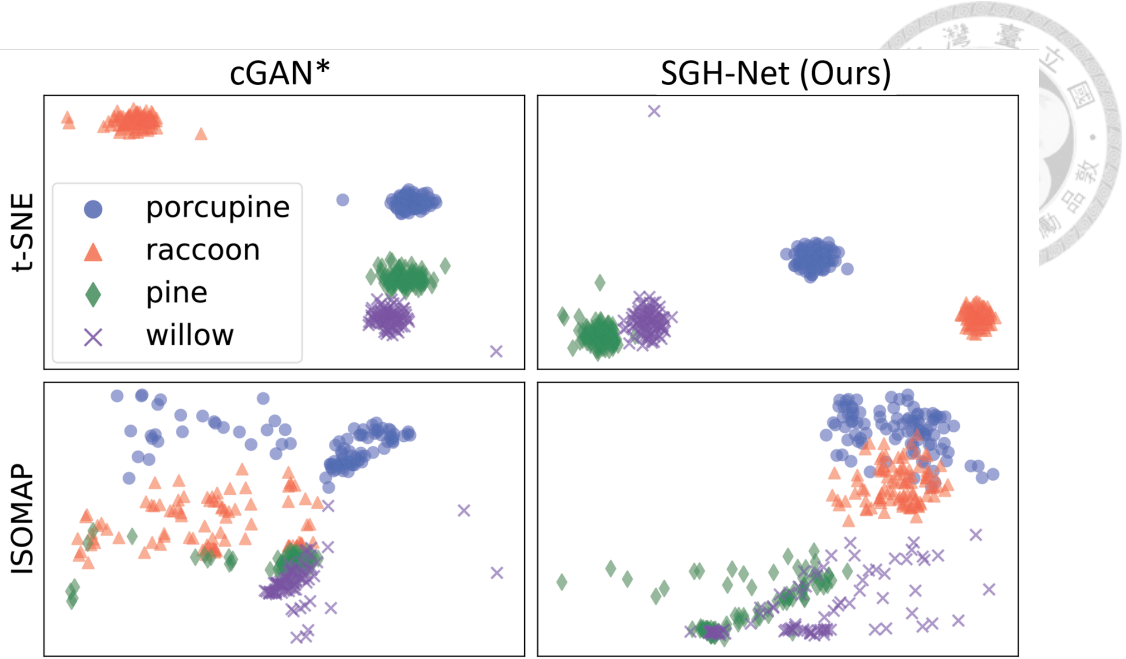


Figure 3.4: 2D visualization of hallucinated features on selected categories of CIFAR-100.

for numbers over  $\mathcal{C}_{novel}^{fin}$  and 1% for numbers over  $\mathcal{C}_{novel}^{fin} \cup \mathcal{C}_{base}^{fin}$ .

To make the comparisons more complete, we further consider recent FSL methods incorporating semantic information. We consider the analogy-based hallucinator proposed in [1] (denoted as Analogy), and further improve it by collecting each of the analogy quadruplets (on which  $a : b :: c : d$  is held) from the same superclass (denoted as Analogy-Superclass). As can be seen from the table, while the use of superclass information also exhibited satisfactory results, it would require prior/manual definitions of superclasses, which might not always be applicable. Nevertheless, while conditional-GAN based methods were promising, our model achieved the best results for all shot numbers.

**Visualization** Figure 3.4 shows 2D visualizations of hallucinated features for novel categories using our model and the conditional-GAN based hallucinator (cGAN\*) [2] (left) and our method (right) using t-SNE (top) and ISOMAP (bottom). We select four categories from two semantically different superclasses, and generate 100 hallucinated features based on a fixed seed  $s_i$  for each category. While both methods resulted in satisfactory data

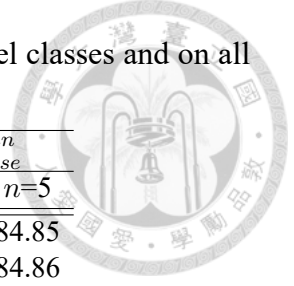


Table 3.2: Results of top-5 accuracy (in percentage) on AWA on novel classes and on all classes.

Test label space	$C_{novel}^{fin}$			$C_{novel}^{fin} \cup C_{base}^{fin}$		
	$n=1$	$n=2$	$n=5$	$n=1$	$n=2$	$n=5$
Baseline	42.03	56.93	72.78	71.03	77.70	84.85
cGAN [2]	59.04	67.14	76.61	77.28	80.98	84.86
cGAN* [2]	58.96	67.05	75.93	77.53	80.67	84.27
SGH-net (ours)	<b>63.59</b>	<b>70.55</b>	<b>78.20</b>	<b>79.78</b>	<b>82.50</b>	<b>85.40</b>

distributions via t-SNE [98] (which focuses more on capturing *local* structures of high-dimensional data), our model exhibited better results via ISOMAP visualization [99], in which *global* geometric structures of the data manifold would be preserved. For example, in the lower part of Figure 3.4, features generated from cGAN\* [2] tended to mix together without semantically meaningful shapes. On the other hand, features generated from our model exhibited more semantics-oriented modes of variation, as trees (pine and willow) distributed roughly along a line, and medium-sized mammals (porcupine and raccoon) distributed in a region more like an ellipse.

### 3.4.3.2 Animals with Attributes

**Quantitative results** Table 3.2 lists the performances on AWA using different methods. Again, our proposed SGH-Net performed favorably against all other methods. However, we observe a less significant improvement compared to the results on CIFAR-100, with larger standard deviations ranging from 1% to 4%. This may be due to the fact that the object of interest in AWA generally occupies a smaller portion of an image when compared to those in CIFAR-100. Moreover, the 50 categories in AWA are not balanced, and yet we do not leverage any external knowledge like superclass information to split those categories. These two facts may make the used semantic concepts (attribute vectors) less influential in the hallucination process.

### 3.5 Summary



In this study, we presented a novel semantics-guided hallucination network (SGH-Net) for few-shot image classification. Instead of modeling feature intra-class variation from a pre-specified distribution across different object categories, the proposed SGH-Net incorporates semantic information into the data hallucination process, and thus the augmented data would exhibit semantics-oriented modes of variation. As confirmed in our experiments on two benchmark datasets (CIFAR-100 and Animals with Attributes), our method quantitatively and qualitatively performed favorably against baseline and recent hallucination approaches.





## Chapter 4

# Feature Disentanglement based Hallucination for Few-Shot Learning

### 4.1 Overview

In Chapter 3, we addressed FSL tasks with semantic information available to mimic human learning procedures of leveraging prior knowledge. However, such semantic prior might not be always available in FSL problems with rarely-occurred categories or categories that require expert knowledge (e.g., medical imaging). In this chapter, we consider a more general FSL scenario in which semantic information is not available and only visual information from base-category data can be relied on. In this case, we propose to explore the abundant *appearance* information extracted from the base-category data as our prior knowledge to deal with FSL tasks.

The basic idea is, the total information contained in an image could be generally *disentangled* into two factors: one is category-specific, while the other is appearance-specific. As shown in Figure 4.1(a), images from a certain category *Western Gull* from the CUB dataset [100] may exhibit very diverse visual appearances. On the other hand,



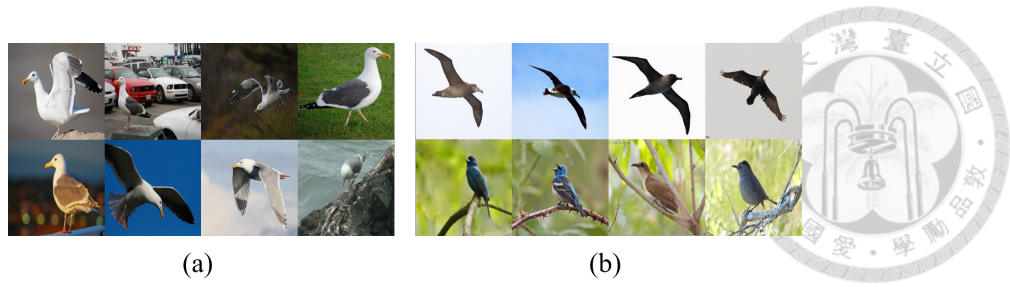
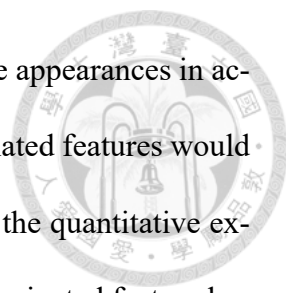


Figure 4.1: Illustration of category-specific and appearance-specific information contained in images.

images across different categories might still share highly similar appearance information, as illustrated in Figure 4.1(b), in which eight different species of birds from CUB exhibit only two kinds of appearance. Since the ultimate goal of data hallucination is to preserve the categorical information from the few-shot class and further introduce appearance variations to the hallucinated samples, it would be critical to be able to disentangle categorical and appearance information from visual contents. Note that we follow previous feature-disentanglement based works [101–104] and use the term *appearance* to describe *all* visual variability contained in an image apart from its categorical meaning, including but not limited to *pose*, *style*, and *background*.

Based on the above observation and motivation, we propose a Feature Disentanglement and Hallucination Network (FDH-Net) to jointly optimize both data hallucination and disentanglement. Our FDH-Net first disentangles visual features into category-specific and appearance-specific factors, and then hallucinate additional data by combining the categorical factor extracted from a novel sample with various appearance factors extracted from arbitrarily many base samples. To achieve this goal, our FDH-Net thus follows the example-guided hallucination approach as described in Section 2.1.2.2 and Figure 2.5(c), which takes a *seed* sample  $s_i$  and a *reference* samples  $s_j$  as its inputs to make each hallucination. By jointly optimizing the disentanglement module and the hallucination module, our FDH-Net is able to hallucinate features that not only preserve the



categorical information from its seed inputs, but also present desirable appearances in accordance with its reference inputs. With these properties, our hallucinated features would be not only more *useful* for downstream FSL tasks as later shown in the quantitative experiment results, but also more *explainable* in the sense that each hallucinated feature has an explicit appearance guidance as later demonstrated by visualizations. Furthermore, the reference inputs are not necessary to be visually similar to the seed inputs. As later will be shown in our experiments, our FDH-Net is able to extract proper appearance information and make feasible hallucination for improved FSL performance, even for coarse-grained datasets with diverse definitions of categories.

The main ideas of this chapter have been peer-reviewed and published in [105].

## 4.2 Problem Definition

The problem setting basically follows the standard multi-class few-shot classification scenario [1] described in Chapter 2 and adopted in Chapter 3. Suppose that we are given two training sets: 1) a base training set  $\mathcal{D}_{base} = \{(x, y) | x \in \mathcal{X}, y \in \mathcal{C}_{base}\}$ , where  $\mathcal{X}$  denotes the image space and  $\mathcal{C}_{base}$  denotes the set of base categories, each with a sufficient amount of samples; 2) a novel training set  $\mathcal{D}_{novel} = \{(x, y) | x \in \mathcal{X}, y \in \mathcal{C}_{novel}\}$ , where  $\mathcal{C}_{novel}$  denotes the set of novel categories, each with only few samples. Note that each sample in  $\mathcal{D}_{base}$  and  $\mathcal{D}_{novel}$  is just an image-label pair without additional semantic information. Based on  $\mathcal{D}_{base}$  and  $\mathcal{D}_{novel}$ , our goal is to build a classifier that can be used to predict the label of an unseen test sample from the joint label space  $\mathcal{C}_{base} \cup \mathcal{C}_{novel}$ . For the sake of clarity, we further denote the held-out testing sets of base and novel categories as  $\mathcal{D}_{base}^{test}$  and  $\mathcal{D}_{novel}^{test}$ , respectively.

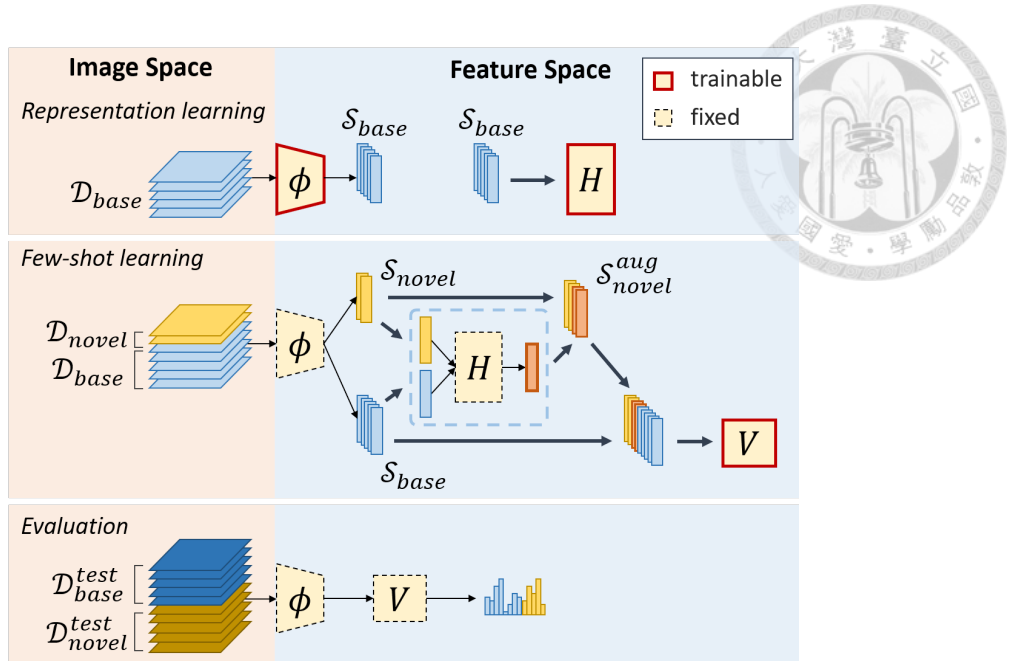


Figure 4.2: Training and evaluation procedures for multi-class few-shot classification.

To realize the above FSL goal, two learning stages are required for training (similar to Figure 3.2), and a single evaluation stage is needed, as depicted in Figure 4.2. The sets of categories involved during training are different in the two learning stages. In the first *representation learning* stage, only  $\mathcal{D}_{base}$  is utilized in a standard supervised-learning manner to derive a feature extractor  $\phi$ . Based on  $\phi$  (with model parameters fixed), a base feature set  $\mathcal{S}_{base} = \{(s, y) | s \in \mathbb{R}^d, y \in \mathcal{C}_{base}\}$  is created, where  $d$  denotes the dimension of the feature vector  $s = \phi(x)$ . Based on  $\mathcal{S}_{base}$ , a hallucinator  $H$  is then trained in a meta-learning manner as described in Chapter 2 (and detailed in Section 3.3.3) with an aim to generate additional training features for novel categories.

As for the second training stage, we perform *few-shot learning* with both  $\mathcal{D}_{base}$  and  $\mathcal{D}_{novel}$  available. We first create a novel feature set  $\mathcal{S}_{novel} = \{(s, y) | s \in \mathbb{R}^d, y \in \mathcal{C}_{novel}\}$  using  $\phi$  (with model parameters fixed). Based on  $\mathcal{S}_{base}$  and the  $n$  feature vectors per novel category (we consider  $n \in \{1, 5\}$  in this work), we use  $H$  to generate additional feature vectors for each novel category (with the number set by validation). Thus far, we obtain an *augmented* novel feature set  $\mathcal{S}_{novel}^{aug}$  made by few real and a large number of hallucinated

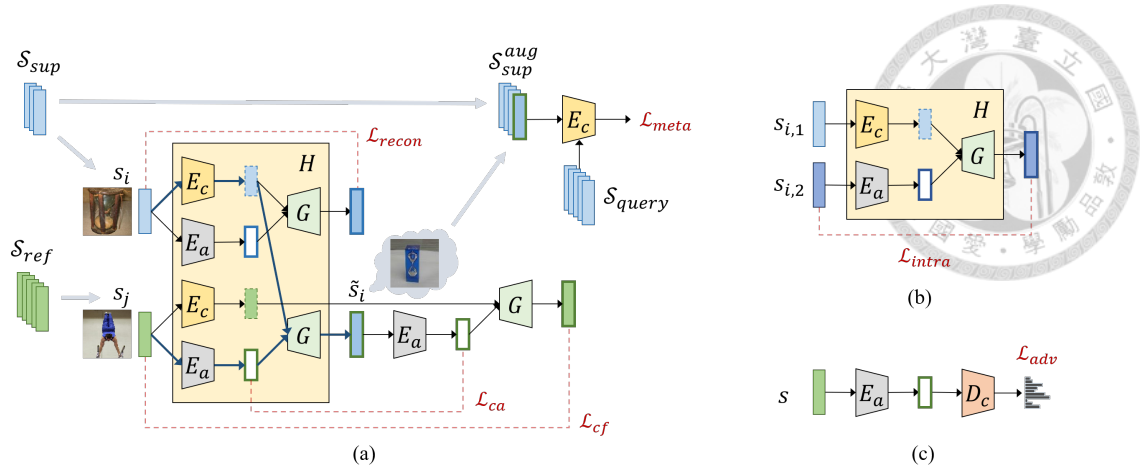


Figure 4.3: Illustration of our Feature Disentanglement and Hallucination Network (FDH-Net).

feature vectors for each novel category. Finally, a linear classifier  $V$  is trained on the joint feature set  $S_{base} \cup S_{novel}^{aug}$  in a supervised-learning manner.

As for the *evaluation* stage, the trained multi-class classifier (i.e., the composition of  $\phi$  and  $V$  in Figure 4.2) is directly applied to perform recognition test data from both base and novel categories, *without* the need to access additional training samples from both the base or novel categories. In the following sections, we describe the design and training procedures of our data hallucinator  $H$  in detail.

### 4.3 Appearance-Guided Hallucination

As stated in Section 4.1, our FDH-Net follows the example-guided hallucination approach, with the hallucinator  $H$  taking two feature vectors as inputs: 1) the *seed* feature vector  $s_i$ , which specifies the label of hallucinated features; 2) the *reference* feature vector  $s_j$ , which serves as the class-invariant appearance (object pose, background, illumination, etc.) guidance. Specifically,  $H$  consists of a class encoder  $E_c$ , an appearance encoder  $E_a$ , and a decoder  $G$ , as depicted in Figure 4.3. The class encoder  $E_c$  maps  $s_i$  into a class-specific space, on which classification tasks are performed. On the other hand, the

appearance encoder  $E_a$  maps  $s_j$  into another class-invariant appearance space, which aims at capturing shared modes of intra-class variation across categories. Finally, the decoder  $G$  takes the categorical and appearance information extracted by  $E_c$  and  $E_a$  as its inputs to produce the hallucinated feature  $\tilde{s}_i$ . The hallucination process can thus be expressed as:

$$\tilde{s}_i = G(E_c(s_i), E_a(s_j)). \quad (4.1)$$

Equation (4.1) can be interpreted in two ways. First, the hallucinated feature  $\tilde{s}_i$  can be seen as the seed feature  $s_i$  being transformed into another appearance, guided by the reference feature  $s_j$ . Second,  $\tilde{s}_i$  can also be seen as  $s_j$  being transformed into the same category as  $s_i$ . Either way, our hallucinator is able to generate additional training samples for the category of  $s_i$  to facilitate the downstream FSL tasks, with appearances referring to arbitrarily many  $\{s_j\}$  from other categories.

### 4.3.1 Preserving Categorical Information

**Meta-learning loss** For the hallucinated features  $\tilde{s}_i$  to preserve the categorical information of the seed  $s_i$ , we follow Chapter 3 and previous works [2, 43, 50] to train our hallucinator  $H$  jointly with a meta-learning module  $h$  with multiple  $N$ -way  $K$ -shot tasks made from  $\mathcal{D}_{base}$ . We again implement  $h$  as a prototypical network [32]. The basic idea is to have  $H$  generate additional training examples for each of the  $N$  categories, such that a better prototypical classifier  $h$  can be built to deal with each  $N$ -way  $K$ -shot task. Concretely, in each episode, we first sample  $N$  categories from  $\mathcal{C}_{base}$  and  $K$  features per category to make the *support* set  $S_{sup} = \{ \{ (s_{n,k}, y'_n) \}_{k=1}^K \}_{n=1}^N$ , where  $s_{n,k}$  denotes the  $k$ -th feature vector of the  $n$ -th category and  $y'_n$  denotes its  $N$ -way label within this episode

(typically,  $y'_n \in \{1, 2, \dots, N\}$  for all  $n$ ). To make hallucination, we further sample another  $N$  categories from  $\mathcal{C}_{base}$  and a few features per category to form the *reference* set  $S_{ref}$ . We then use Equation (4.1) to generate  $K_h$  additional features for each category in  $S_{sup}$ :

$$\tilde{s}_{n,k} = G(E_c(\bar{s}_n), E_a(s_r)) \quad (4.2)$$

for  $n = 1, 2, \dots, N$  and  $k = 1, 2, \dots, K_h$ , where  $\bar{s}_n = \frac{1}{K} \sum_{k=1}^K s_{n,k}$  is the *averaged* feature vector of the  $n$ -th category, and  $s_r$  is randomly sampled from  $S_{ref}$ . These additional features are added to  $S_{sup}$  to create an *augmented* support set  $S_{sup}^{aug} = \{(s_{n,k}, y'_n)\}_{k=1}^K \cup \{(\tilde{s}_{n,k}, y'_n)\}_{k=1}^{K_h}\}_{n=1}^N$ , from which  $h$  can be built. For simplicity, we directly compute the class prototypes and classification losses in the categorical feature space induced by the class encoder (i.e.,  $E_c$  in Figure 4.3), without explicitly implementing a prototypical network for the feature embedding purpose. Based on  $S_{sup}^{aug}$ , the class prototype of the  $n$ -th category within this episode can be derived by:

$$\mu_n = \frac{1}{K} \sum_{k=1}^K E_c(s_{n,k}) + \frac{1}{K_h} \sum_{k=1}^{K_h} E_c(\tilde{s}_{n,k}). \quad (4.3)$$

Let  $S_{query} = \{(s_q, y'_q)\}$  denotes the *query* set, the meta-learning loss for each episode can thus be expressed as:

$$\mathcal{L}_{meta} = \mathbb{E}_{(s_q, y'_q) \in S_{query}} \left[ - \sum_{n=1}^N \mathbb{I}[y'_q = n] \log P(n|s_q) \right], \quad (4.4)$$

where  $\mathbb{I}[\cdot]$  denotes the indicator function, which equals to 1 if the statement inside the brackets is true, and



$$P(n|s_q) = \frac{e^{-\|E_c(s_q) - \mu_n\|}}{\sum_{n'} e^{-\|E_c(s_q) - \mu_{n'}\|}} \quad (4.5)$$

computes the probability of  $s_q$  being assigned to the  $n$ -th category within this episode. By optimizing Equation (4.4), the hallucinator  $H$  is encouraged to generate data such that, when used to augment the support set, the classification performance for each episode can be improved. As a result, the hallucinated features  $\{\tilde{s}_i\}$  would be enforced to preserve categorical information from the corresponding seed inputs  $\{s_i\}$ .

**Reconstruction loss** Since our hallucinator  $H$  follows an autoencoder architecture [106], we further secure its data-recovery ability to facilitate training and ensure that most of the visual concepts (either categorical or appearance) will be properly embedded. Specifically, when the seed and reference input pairs are actually the same feature vector (i.e.,  $s_i = s_j$ ), the decoder  $G$  should be able to reconstruct it based on its categorical and appearance information. The reconstruction loss can be expressed as:

$$\mathcal{L}_{recon} = \mathbb{E} [\|s_i - G(E_c(s_i), E_a(s_i))\|]. \quad (4.6)$$

By training with Equation (4.4) and Equation (4.6), our hallucinator  $H$  is encouraged to generate additional features with categorical information being preserved from their seed inputs  $\{s_i\}$ . However, it is not guaranteed that the appearance information from  $\{s_j\}$  would be well referenced during the hallucination process.  $H$  may simply copy all the visual information (both categorical and appearance) from  $\{s_i\}$  without referring to  $\{s_j\}$ , and fail to generate useful hallucinations for downstream FSL tasks. To avoid this problem and improve the usefulness of the hallucinated features, we propose further loss functions

and training techniques as follows.



### 4.3.2 Preserving Appearance Information

**Consistency losses** For the hallucinated feature  $\tilde{s}_i$  to preserve the appearance information of the reference feature  $s_j$ , we leverage the feature disentanglement techniques from recent image-to-image translation works [107, 108] and propose the following consistency losses. Specifically, we extract the appearance information from the hallucinated output to obtain  $E_a(\tilde{s}_i)$ , which should represent the original appearance information  $E_a(s_j)$  used to generate  $\tilde{s}_i$ . The consistency loss in the *appearance* space can thus be expressed as:

$$\mathcal{L}_{ca} = \mathbb{E} [\|E_a(s_j) - E_a(\tilde{s}_i)\|]. \quad (4.7)$$

Similarly, we could also feed  $s_j$  into the class encoder  $E_c$  to extract its categorical information  $E_c(s_j)$ . By combining  $E_c(s_j)$  with the above  $E_a(\tilde{s}_i)$  (which we enforce to be close to  $E_a(s_j)$ ), the decoder  $G$  should be able to recover  $s_j$ . The consistency loss in the *feature* space can thus be expressed as:

$$\mathcal{L}_{cf} = \mathbb{E} [\|s_j - G(E_c(s_j), E_a(\tilde{s}_i))\|]. \quad (4.8)$$

Unfortunately, simply training our model with Equation (4.7) and Equation (4.8) still does not guarantee the appearance information will be well preserved in the hallucinated features. It is still possible that most of the categorical and appearance information gets embedded into the class space induced by  $E_c$ , leaving  $E_a$  degenerated with potentially all-zero outputs. In such a case,  $G$  can still achieve data recovery based on solely the output



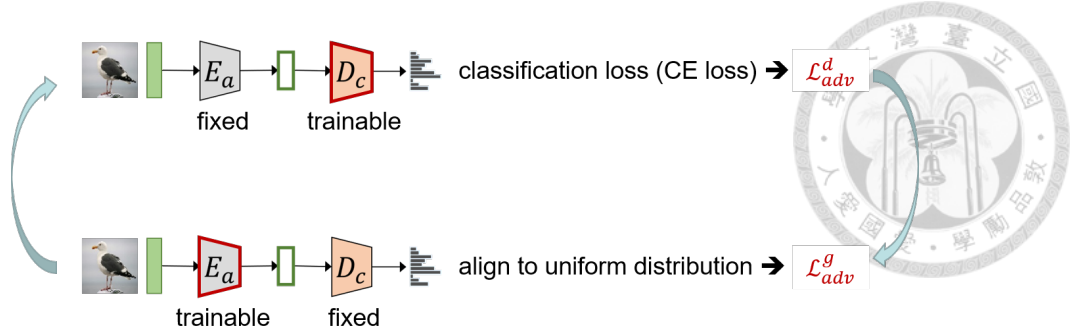


Figure 4.4: Illustration of the adversarial training of the appearance encoder and the discriminator.

of  $E_c$ , and make mediocre hallucination by modeling the intra-class variation using the noise-like output of  $E_a$  (just like the noise-based hallucinators [2, 43]).

To address the above issue and facilitate class-appearance disentanglement, we propose the intra-class training technique, as depicted in Figure 4.3(b). Specifically, let  $s_{i,1}$  and  $s_{i,2}$  denote two feature vectors sampled from the same category. If we take the categorical information from  $s_{i,1}$  and the appearance information from  $s_{i,2}$ , the decoder  $G$  should be able to reconstruct  $s_{i,2}$ , which shares the same categorical information with  $s_{i,1}$ . The intra-class consistency loss can thus be expressed as:

$$\mathcal{L}_{intra} = \mathbb{E} [\|s_{i,2} - G(E_c(s_{i,1}), E_a(s_{i,2}))\|]. \quad (4.9)$$

**Adversarial loss** By training with Equation (4.9), the appearance encoder  $E_a$  is enforced to extract non-trivial information from the reference input  $s_j$ , which might still contain categorical information. To further improve the quality of class-appearance disentanglement, we follow [107] to design a class discriminator  $D_c$ , which aims to correctly predict the labels of  $s_j$  from its appearance information  $E_a(s_j)$ . On the other hand, the appearance encoder  $E_a$  is trained in an adversarial manner by making the label *unpredictable* from its outputs. Specifically,  $E_a$  and  $D_c$  are trained iteratively, as depicted in Figure 4.4. When

optimizing  $D_c$ , we fixed  $E_a$  and impose a cross-entropy classification loss on the output of  $D_c$ ; When optimizing  $E_a$ , we fixed  $D_c$  and encourage its output to align to a uniform distribution. We note that, such training objectives require accessing to the original base class labeling ( $y \in \{1, 2, \dots, |\mathcal{C}_{base}|\}$ ) and are inherently less suitable in meta-learning frameworks, where only  $N$ -way labels ( $y' \in \{1, 2, \dots, N\}$ ) are available in each episode. Therefore, we resort to a mini-batch training scheme and propose the adversarial loss for  $D_c$  as:

$$\mathcal{L}_{adv}^d = \mathbb{E}_{(s,y) \in \mathcal{S}_{base}} \left[ \sum_{n=1}^{|\mathcal{C}_{base}|} \mathbb{I}[y = n] \log P_D(n|s) \right], \quad (4.10)$$

and the adversarial loss for  $E_a$  as

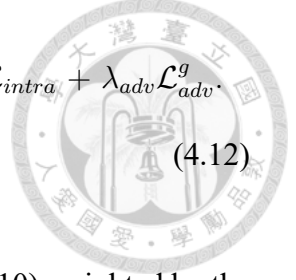
$$\mathcal{L}_{adv}^g = \mathbb{E}_{(s,y) \in \mathcal{S}_{base}} \left[ \frac{1}{|\mathcal{C}_{base}|} \sum_{n=1}^{|\mathcal{C}_{base}|} \log P_D(n|s) \right], \quad (4.11)$$

where  $P_D(n|s)$  denotes the  $n$ -th dimension of  $D_c(E_a(s))$ , i.e., the probability of  $E_a(s)$  being assigned to base class  $n \in \mathcal{C}_{base}$  by  $D_c$ .

### 4.3.3 Optimization Procedure

The total objective function of our FDH-Net is derived by collecting all the above losses from equations (4.4), (4.6), (4.7), (4.8), (4.9), (4.10), and (4.11), with each term weighted by a hyperparameter. For clarity, we express our FDH loss function for our FDH-Net (except for the discriminator  $D_c$ ) as:

$$\mathcal{L}_{fdh} = \lambda_{meta}\mathcal{L}_{meta} + \lambda_{recon}\mathcal{L}_{recon} + \lambda_{ca}\mathcal{L}_{ca} + \lambda_{cf}\mathcal{L}_{cf} + \lambda_{intra}\mathcal{L}_{intra} + \lambda_{adv}\mathcal{L}_{adv}^g. \quad (4.12)$$



As for the discriminator  $D_c$ , we simply optimize  $\mathcal{L}_{adv}^d$  in Equation (4.10) weighted by the same hyperparameter  $\lambda_{adv}$  as  $\mathcal{L}_{adv}^g$ . As described in previous sections, all loss components except  $\mathcal{L}_{adv}^d$  and  $\mathcal{L}_{adv}^g$  can be optimized within each  $N$ -way  $K$ -shot episode. To optimize  $\mathcal{L}_{adv}^d$  and  $\mathcal{L}_{adv}^g$ , we additionally sample a few mini-batches from the same  $N$  categories (with their base class labeling) for each episode, as will be detailed in the experiments.

### 4.3.4 Data Hallucination for Few-shot Learning

At the few-shot learning stage, we train the FSL classifier  $V$  with both the base and novel feature sets available (i.e.,  $\mathcal{S}_{base}$  and  $\mathcal{S}_{novel}$  in Figure 4.2). We first use the trained hallucinator  $H$  to augment  $\mathcal{S}_{novel}$  using Equation (4.1), with the seed feature  $s_i$  being the only shot of each novel category (or the average feature when multiple shots are available), and the reference features  $s_j$ 's randomly sampled from  $\mathcal{S}_{base}$ . After the hallucination process, we then take  $\mathcal{S}_{base}$  and the augmented novel feature set  $\mathcal{S}_{novel}^{aug}$  to train  $V$ .

As noted in Sec. 4.1, our FDH-Net allows feature hallucination using  $s_j$  which is not necessarily similar to  $s_i$  (both semantically or visually). However, we do observe an additional performance gain for coarse-grained datasets if we sample  $s_j$ 's from the *related* base categories defined for each  $s_i$ . To be more precise, we first pre-compute the class-wise feature average (i.e., prototype) for every base category in  $\mathcal{C}_{base}$ . For each novel sample  $s_i$ , we then define its related base categories using its top- $M$  nearest base prototypes in the feature space. We note that, instead of searching similar  $s_j$ 's in the *instance* level for every  $s_i$  as proposed in existing example-guided hallucination works [50, 52], we only make a

minimal effort to search related base categories in the *class* level, making the computational overhead almost negligible. Once the related base categories are specified, we then randomly sample  $s_j$ 's from those categories, without further similarity constraints. More details and a complete evaluation process will be presented in the following section.

## 4.4 Experiments

### 4.4.1 Datasets

We evaluate our proposed FDH-Net on two fine-grained datasets and two coarse-grained datasets, as described below.

**Caltech-UCSD Birds 200-2011 (CUB)** The CUB dataset [100] is a fine-grained dataset containing 11,788 images from 200 bird species. We follow the class split in [109] and use 100, 50, and 50 species as the base, validation, and novel categories, respectively.

**Oxford-Flowers (FLO)** The FLO dataset [110] is also a fine-grained dataset containing 8,189 images from 102 species of flowers. We follow the standard split and use 62, 20, and 20 species as the base, validation, and novel categories, respectively.

**mini-ImageNet** The *mini-ImageNet* dataset [25, 35] is a subset of the ImageNet dataset [38] and contains 60,000 images from 100 categories. We follow the class split in [35] and use 64, 16, and 20 categories as the base, validation, and novel categories, respectively.

**CIFAR-100** The CIFAR-100 dataset [96] also contains 60,000 images from 100 categories. We follow the class split in [44] and use 64, 16, and 20 categories as the base, validation, and novel categories, respectively.



Note that for all the above datasets, we take only images and categorical labels as specified in Sec. 4.2, without accessing any side information about classes such as class-level attributes or hierarchical structure like we did in Chapter 3.

#### 4.4.2 FSL Settings and Implementation Details

As discussed in Sec. 4.3, we follow [1, 2, 49, 50] to evaluate our proposed FDH-Net using multi-class few-shot classification settings. Unlike Chapter 3, where the dataset is divided into 8 subsets for training, validation, and evaluation (i.e., 4 disjoint class subsets  $\mathcal{C}_{base}^{cv}$ ,  $\mathcal{C}_{novel}^{cv}$ ,  $\mathcal{C}_{base}^{fin}$ ,  $\mathcal{C}_{novel}^{fin}$ , each further divided into training and testing subsets). Here, we follow the benchmark [1] and divide all images in each of the base, validation, and novel classes into training and testing subsets by a 4:1 ratio. As a result, we obtain a total of 6 subsets of images for each dataset: the *base training* subset, the *base testing* subset, the *validation training* subset, the *validation testing* subset, the *novel training* subset, and finally the *novel testing* subset. The hyperparameters are tuned on the base and validation subsets. The hyperparameter setting that leads to the best validation performance is then applied for the final evaluation on the base and novel subsets. Considering the fluctuating nature of the accuracy values in FSL tasks, we repeat the training and evaluation procedures depicted in Figure 4.2 for multiple trials, each with a particular few-shot training set  $\mathcal{D}_{novel}$  randomly sampled from either the validation or the novel training subset.

In addition to our FDH-Net, we also implement two baseline methods using the raw

features and the prototypical network features [32] (both without hallucination), as well as other state-of-the-art data hallucination approaches, including cGAN [2], IDeMe-Net [50], AFHN [43], and DTN [46]. Next, we present the general steps in a single trial of training and evaluation that are common for all the above methods.

- **Step 1:** We first train a CNN-based feature extractor  $\phi$  with the *base training* image subset, and then use the trained  $\phi$  to extract feature vectors for all the 6 image subsets. All the subsequent training and evaluation steps are conducted in the feature space.
- **Step 2:** We train a hallucinator  $H$  on the *base training* feature subset (i.e.,  $\mathcal{S}_{base}$ ) based on our FDH-Net or other hallucination approaches [2, 43, 46, 50]. We skip this step for the baselines.
- **Step 3:** We randomly sample  $n \in \{1, 5\}$  feature vectors per category from the *novel* (or *validation*) *training* feature subset to form  $\mathcal{S}_{novel}$ . Based on seed features  $s_i$ 's in  $\mathcal{S}_{novel}$  (and reference features  $s_j$ 's sampled from  $\mathcal{S}_{base}$  for IDeMe-Net [50], DTN [46], and our FDH-Net), we use the trained  $H$  to augment  $\mathcal{S}_{novel}$ . For the coarse-grained *mini-ImageNet* and *CIFAR-100* datasets, we add an additional constraint to our FDH-Net that  $s_j$ 's must be sampled from the related base categories for each  $s_i$ , as described in Sec. 4.3.4. This step is also skipped for the baselines.
- **Step 4:** After the hallucination process, we use  $\mathcal{S}_{base}$  and the augmented *novel* (or *validation*) *training* feature subset  $\mathcal{S}_{novel}^{aug}$  to train a multi-class classifier  $V$ .
- **Step 5:** We combine the trained feature extractor  $\phi$  and the multi-class classifier  $V$  as our final FSL model to predict the labels of the *base testing* image subset and the *novel* (or *validation*) *testing* image subset.

For each hyperparameter setting during validation, we repeat steps 2-5 for 5 trials. The hyperparameter setting and the corresponding hallucinator that lead to the highest top-5 accuracy in step 5 is then selected for the final evaluation, where we repeat 100 trials of steps 3-5 using the base and novel feature subsets. All reported numbers in this study are derived from this final evaluation.

Following [2, 43, 50], we use ResNet-18 [5] as our CNN-based feature extractor  $\phi$  in step 1. In step 2, we train all kinds of hallucinators  $H$ 's via the  $N$ -way  $K$ -shot scheme, with  $N$ ,  $K$ , learning rate, and the number of episodes being set as hyperparameters. In step 3, we serve the number of hallucinated features and the number of related base categories (i.e.,  $M$  in Sec. 4.3.4) for each seed feature as hyperparameters. In step 4, we implement  $V$  as a single-layer MLP and utilize the ADAM optimizer for 1000 iterations with a mini-batch size of 100 and a learning rate set as a hyperparameter.

**Baselines** We implement two baselines. For the first baseline (denoted as Baseline in all tables), we simply use the ResNet-18 raw features to run steps 4 and 5 without hallucination. For the second baseline (denoted as Baseline+ $E_c$  in all tables), we further utilize the *base training* feature subset ( $S_{base}$ ) to train an additional embedding network (implemented as a two-layer MLP with ReLU activation, just like the class encoder  $E_c$  of our FDH-Net) to map all feature vectors into another metric space, on which the prototypical loss is optimized. After training, we then use the embedded features to run steps 4 and 5 without hallucination.

**IDeMe-Net and DTN** For IDeMe-Net<sup>1</sup> [50] and DTN<sup>2</sup> [46], we use the codes provided by the authors to train the corresponding hallucinators in step 2.



**cGAN and AFHN** We use our own implementations of cGAN [2] and AFHN [43] to train the corresponding hallucinators in step 2. For cGAN, we follow the original settings in [2] to implement the hallucinator as a three-layer MLP with ReLU activation, and adopt a prototypical network as the meta-learning module with the embedding function implemented as a two-layer MLP with ReLU activation. The dimension of noise vectors and all hidden layers are set to 512, which is the same as the ResNet-18 feature dimension. As for AFHN, we follow the architecture described in [43] to implement the hallucinator as a two-layer MLP with LeakyReLU activation for the first layer and ReLU for the second one. The discriminator is also implemented as a two-layer MLP with LeakyReLU activation for the first layer. The dimension of all hidden layers is set to 1024 and the dimension of noise vectors is set to 512.

**Our FDH-Net** For our FDH-Net, the class encoder  $E_c$ , the appearance encoder  $E_a$ , the decoder  $G$ , and the class discriminator  $D_c$  are all implemented as two-layer MLPs with ReLU activation functions. All hidden layers have the same dimension as the ResNet-18 output dimension (i.e., 512).

As detailed in Sec. 4.3, our FDH-Net is trained with the objective  $\mathcal{L}_{fdh}$  in Equation (4.12), which includes the meta-loss  $\mathcal{L}_{meta}$ , reconstruction loss  $\mathcal{L}_{recon}$ , feature/appearance consistency losses  $\mathcal{L}_{ca}/\mathcal{L}_{cf}$ , intra-class consistency loss  $\mathcal{L}_{intra}$ , and the adversarial loss  $\mathcal{L}_{adv}^g$ . Instead of fine-tuning the weights for all loss terms in each experiment, we

---

<sup>1</sup><https://github.com/tankche1/IDeMe-Net>

<sup>2</sup><https://github.com/Yuxin-CV/DTN>






---

**Algorithm 1** FDH-Net
 

---

- 1: **Input:** Base image sets  $\{\mathcal{D}_b^{train}, \mathcal{D}_b^{test}\}$  and validation image sets  $\{\mathcal{D}_v^{train}, \mathcal{D}_v^{test}\}$
  - 2: **Output:** Feature extractor  $\phi$ , the best hallucinator  $H^*$
  - 3:  $\mathcal{D}_{base} \leftarrow \mathcal{D}_b^{train}, \mathcal{D}_{base}^{test} \leftarrow \mathcal{D}_b^{test}, \mathcal{D}_{novel}^{test} \leftarrow \mathcal{D}_v^{test}$
  - 4: Initialize  $\phi$
  - 5: Train  $\phi$  on  $\mathcal{D}_{base}$  with the standard cross-entropy loss
  - 6: Use the trained  $\phi$  to create feature sets  $\mathcal{S}_{base}$  from  $\mathcal{D}_{base}$ ,  $\mathcal{S}_{base}^{test}$  from  $\mathcal{D}_{base}^{test}$ , and  $\mathcal{S}_{novel}^{test}$  from  $\mathcal{D}_{novel}^{test}$
  - 7:  $\lambda_{meta} \leftarrow 1$
  - 8: **for**  $\lambda_{recon} = 1, 2, 5, 10, 20, 50$  **do**
  - 9:      $\lambda_{ca} \leftarrow 0.01 \times \lambda_{recon}, \lambda_{cf} \leftarrow 0.01 \times \lambda_{recon}, \lambda_{intra} \leftarrow \lambda_{recon}$
  - 10:    **for**  $\lambda_{adv} = 0.1, 0.5$  **do**
  - 11:     Initialize  $H$
  - 12:     Train  $H$  on  $\mathcal{S}_{base}$  with the loss defined in Equation (4.12)
  - 13:     **for**  $iteration = 1, 2, 3, 4, 5$  **do**
  - 14:          $\mathcal{D}_{novel} \leftarrow$  randomly sample  $n \in \{1, 5\}$  images per category from  $\mathcal{D}_v^{train}$
  - 15:         Use the trained  $\phi$  to create feature set  $\mathcal{S}_{novel}$  from  $\mathcal{D}_{novel}$
  - 16:         Use the trained  $H$  to augment  $\mathcal{S}_{novel}$  to create  $\mathcal{S}_{novel}^{aug}$
  - 17:         Train  $V$  on  $\mathcal{S}_{base} \cup \mathcal{S}_{novel}^{aug}$
  - 18:         Use the trained  $V$  to predict test features from  $\mathcal{S}_{base}^{test} \cup \mathcal{S}_{novel}^{test}$
  - 19:     **end for**
  - 20:    **end for**
  - 21:      $H^* \leftarrow H$  if achieve a higher average accuracy from lines 13-19
  - 22: **end for**
- 

fix  $\lambda_{meta} = 1$  and set  $\lambda_{ca} = \lambda_{cf} = 0.01 \times \lambda_{recon}$ , while  $\lambda_{recon}$  from  $\{1, 2, 5, 10, 20, 50\}$  is determined via validation. We also fixed  $\lambda_{intra} = \lambda_{recon}$  to enforce the appearance disentanglement ability. Finally, for the adversarial losses  $\mathcal{L}_{adv}^d$  and  $\mathcal{L}_{adv}^g$ , its weight  $\lambda_{adv}$  is searched from  $\{0.1, 0.5, 1.0, 5.0\}$  and empirically fixed in the range  $\lambda_{adv} \in \{0.1, 0.5\}$ . Based on the above guidelines, the pseudo code for training our FDH-Net is provided in Algorithm 1.

As we described in Sec. 4.3, we train the hallucinator on base categories by creating a large amount of  $N$ -way  $K$ -shot episodes. The seed vectors and reference vectors are simply randomly sampled in the training stage. Specifically, for each episode, we randomly sample  $N$  base categories and  $K$  features per class to form the support set  $S_{sup}$ , and the same random sample strategy is applied for the reference set  $S_{ref}$  (i.e., another  $N$

Table 4.1: Performance comparison on fine-grained datasets (CUB and FLO).

Method	CUB				FLO			
	1-shot		5-shot		1-shot		5-shot	
	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
Baseline	6.95±0.28	21.99±0.57	19.04±0.20	56.71±0.19	15.93±0.35	50.91±0.83	44.02±0.58	78.52±0.43
Baseline+ $E_c$	8.97±0.31	26.83±0.50	19.66±0.31	55.64±0.39	18.73±0.73	50.17±1.01	45.71±0.55	79.60±0.47
cGAN [2]	8.08±0.29	30.86±0.91	16.16±0.09	53.77±0.21	16.62±0.64	53.42±1.12	35.50±0.63	73.07±0.64
IDeMe-Net [50]	8.24±0.34	26.06±0.57	21.93±0.24	58.11±0.31	20.16±0.66	53.97±1.07	44.59±0.26	78.01±0.35
DTN [46]	8.30±0.36	30.91±0.69	14.42±0.33	51.99±0.59	14.98±0.63	58.19±1.01	44.67±0.56	85.16±0.44
AFHN [43]	9.68±0.29	38.47±0.55	20.25±0.26	62.07±0.32	22.88±0.66	63.67±0.90	40.24±0.66	82.66±0.40
Our FDH-Net	<b>12.64±0.31</b>	<b>45.51±0.62</b>	<b>23.09±0.35</b>	<b>67.14±0.28</b>	<b>26.81±0.91</b>	<b>70.06±1.07</b>	<b>46.04±0.56</b>	<b>86.93±0.39</b>

Table 4.2: Performance comparison on coarse-grained datasets (*mini*-ImageMet and CIFAR-100).

Method	<i>mini</i> -ImageMet				CIFAR-100			
	1-shot		5-shot		1-shot		5-shot	
	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
Baseline	6.13±0.21	21.68±0.42	22.95±0.29	60.38±0.33	6.64±0.19	26.03±0.56	17.72±0.29	59.49±0.34
Baseline+ $E_c$	8.31±0.30	26.72±0.52	19.75±0.29	57.23±0.40	8.16±0.32	29.02±0.60	14.88±0.27	55.87±0.33
cGAN [2]	6.27±0.32	32.40±0.76	17.17±0.36	57.08±0.34	7.60±0.14	41.52±0.98	17.18±0.15	62.52±0.22
IDeMe-Net [50]	14.61±0.40	45.21±0.77	22.89±0.35	65.29±0.41	12.31±0.64	46.24±0.82	19.18±0.41	64.05±0.37
DTN [46]	5.69±0.29	34.92±0.76	12.82±0.25	60.52±0.39	5.67±0.39	39.94±0.86	13.75±0.27	61.81±0.38
AFHN [43]	11.96±0.29	51.17±0.76	23.03±0.29	72.77±0.33	9.41±0.26	50.55±0.73	19.77±0.23	72.53±0.27
Our FDH-Net	11.85±0.31	54.01±0.78	20.63±0.31	72.06±0.34	9.07±0.27	56.94±0.85	17.19±0.31	<b>74.86±0.34</b>
Our FDH-Net*	<b>15.68±0.34</b>	<b>58.06±0.75</b>	<b>23.85±0.29</b>	<b>75.13±0.27</b>	<b>13.16±0.39</b>	<b>57.97±0.64</b>	<b>20.99±0.29</b>	74.20±0.30

base categories with  $K_h$  features per class). We then perform hallucination by taking seed vectors from  $S_{sup}$  and reference vectors from  $S_{ref}$  (as illustrated in Figure 4.3), and train our FDH-Net using the objective function defined in Equation (4.12).

### 4.4.3 Evaluation

**Quantitative results** Table 4.1 lists the performance comparisons on two considered fine-grained datasets, CUB and FLO, with each number representing the top-1 or top-5 accuracy (%) on  $\mathcal{D}_{novel}^{test}$ , followed by the 95% confidence interval. The best results are in **bold**. It can be seen that our FDH-Net performs favorably against all other methods, especially in the most challenging 1-shot case. With more training data available for novel categories (i.e.,  $n = 5$ ), the performance gaps among different hallucination approaches become smaller. Nevertheless, our framework still shows significant improvement.

As a realization of the example-guided hallucination approach, our proposed FDH-

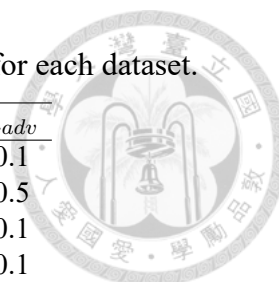


Table 4.3: The best set of loss weights found through validation for each dataset.

Dataset	$\lambda_{meta}$	$\lambda_{recon}$	$\lambda_{ca}$	$\lambda_{cf}$	$\lambda_{intra}$	$\lambda_{adv}$
CUB	1	2	0.02	0.02	2	0.1
FLO	1	50	0.5	0.5	50	0.5
<i>mini</i> -ImageNet	1	10	0.1	0.1	10	0.1
CIFAR-100	1	5	0.05	0.05	5	0.1

Net is intuitively more feasible for fine-grained datasets, in which different categories tend to share similar modes of intra-class variation such that the appearance information would be easier to be referred. However, we empirically find that even coarse-grained datasets can benefit from our disentanglement-based hallucination for FSL tasks. To see this, Table 4.2 lists the performance comparisons on the coarse-grained *mini*-ImageNet and CIFAR-100 datasets. In this case, our FDH-Net again performed favorably against all other methods in top-5 accuracy. By further constraining the reference features to be sampled from the related base categories for each seed feature (we set  $M = 20$  through validation), we observe a significant boost in performance in the last row of Table 4.2 (denoted as FDH-Net\*), especially in top-1 accuracy. The best set of hyperparameters for each dataset is listed in Table 4.3.

**Visualization** Figure 4.5 shows t-SNE visualizations of our hallucinated features for novel categories of the CUB dataset for the 1-shot case. Here, we randomly sample five novel categories from CUB and visualize the real and hallucinated features in the feature space. It can be seen that our FDH-Net is able to generate diverse yet discriminative hallucinations for categories unseen during training based on a single seed feature per novel category. For each category, we also selected two hallucination examples and show their nearest real images to demonstrate our ability of appearance transformation. We note that, since the effectiveness of our proposed model has been *quantitatively* confirmed through the aforementioned FSL performance comparisons in Table 4.1 and Table 4.2, we thus find

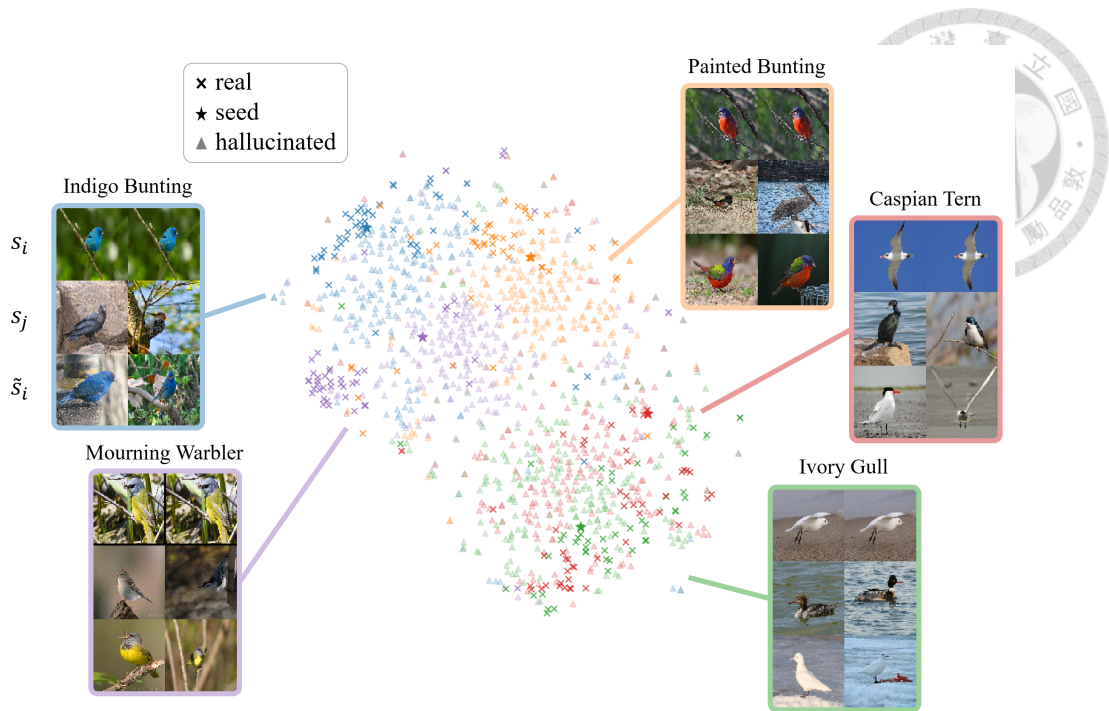


Figure 4.5: t-SNE visualization and nearest real images of our hallucinated features on the CUB dataset.



Figure 4.6: Examples of hallucinated results on four benchmark datasets.

the nearest real feature for each hallucinated feature from the features of the same label as the seed feature. For the image set of each category in Figure 4.5, the first row presents the image corresponding to the seed feature  $s_i$ , the second row presents images corresponding to the reference features  $s_j$  from base categories, and the last row presents images corresponding to the nearest real features of the hallucinated features  $\tilde{s}_i$ . As can be seen from the last row, our hallucinated features exhibit highly similar appearances according to the reference features presented in the second row. More examples from other datasets can be found in Figure 4.6, which again demonstrates our ability to produce desirable feature hallucinations, even for coarse-grained datasets, in which reference features might not be



Figure 4.7: More data hallucination examples produced by our FDH-Net using the same seed image of school bus from *mini-ImageNet*.

visually or semantically similar to the associated seed features. In Figure 4.6, the category labels are (from left to right): Le Conte Sparrow, Black Billed Cuckoo, Ivory Gull, and Western Gull from CUB; balloon flower, pink primrose, peruvian lily, and canterbury bells from FLO; hourglass, school bus, vase, and trifle from *mini-ImageNet*; turtle, plates, leopard, and maple from CIFAR-100.

To provide additional illustration examples, Figure 4.7 takes a single categorical seed from *mini-ImageNet* (i.e.,  $s_i$  in the first row) with different appearance references (i.e.,  $s_j$  in the second row). Each image in the second row is randomly sampled from a class (e.g., a trash can shown in the last column) different from the first row (school bus), and is expected to provide its appearance information (e.g., the presence of children) rather than its categorical one to make the hallucinated outputs shown in the third row. It can be seen that, even if the class information of  $s_i$  and  $s_j$  are quite different, our model is able to capture and leverage the appearance information for hallucinating the final outputs, while also preserving the category information of interest.

Table 4.4: Ablation study for the loss functions of our proposed FDH-Net on CUB and *mini*-ImageNet datasets in the 1-shot case.

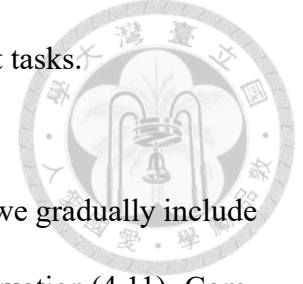
Item	Loss functions in FDH-Net						CUB		<i>mini</i> -ImageNet	
	$\mathcal{L}_{meta}$	$\mathcal{L}_{recon}$	$\mathcal{L}_{ca}$	$\mathcal{L}_{cf}$	$\mathcal{L}_{intra}$	$\mathcal{L}_{adv}^g$	top-1	top-5	top-1	top-5
(a) Baseline	-	-	-	-	-	-	6.95±0.28	21.99±0.57	6.13±0.21	21.68±0.42
(b)	✓	×	×	×	×	×	3.63±0.19	27.50±0.55	10.38±0.44	35.96±0.88
(c)	✓	✓	×	×	×	×	7.55±0.28	29.15±0.52	4.55±0.18	27.47±0.63
(d)	✓	✓	✓	×	×	×	6.34±0.27	26.29±0.55	4.94±0.20	28.66±0.64
(e)	✓	✓	✓	✓	×	×	7.17±0.29	28.37±0.53	5.37±0.22	29.75±0.68
(f)	✓	✓	✓	✓	✓	×	6.84±0.24	35.73±0.60	8.12±0.24	42.03±0.72
(g)	✓	✓	✓	✓	×	✓	6.65±0.27	24.37±0.48	4.86±0.21	28.01±0.62
(h)	✓	×	×	×	✓	✓	11.10±0.32	44.28±0.60	13.86±0.32	56.52±0.75
(i)	✓	×	✓	✓	✓	✓	11.87±0.34	45.07±0.63	15.11±0.35	57.37±0.65
(j)	×	✓	✓	✓	✓	✓	10.96±0.28	45.56±0.56	7.16±0.24	44.57±0.67
(k)	×	×	✓	✓	✓	✓	8.70±0.29	42.12±0.59	8.37±0.25	51.12±0.76
(l)	✓	✓	×	×	✓	✓	12.31±0.34	<b>46.06±0.63</b>	14.57±0.40	57.60±0.78
(m)	✓	✓	✓	✓	✓	✓	<b>12.64±0.31</b>	45.51±0.62	<b>15.68±0.34</b>	<b>58.06±0.75</b>

#### 4.4.4 Ablation Study

In this section, we provide quantitative and visualization results of the ablation study for the loss functions of our FDH-Net. Table 4.4 lists the performance comparisons on the *novel test* subset of CUB and *mini*-ImageNet in the 1-shot case, with each number representing the top-1 or top-5 accuracy (%) on  $\mathcal{D}_{novel}^{test}$ , followed by the 95% confidence interval. The best results are in **bold**. In addition to the Baseline (a) and our FDH-Net (m), we also conduct experiments for our framework with the hallucinator  $H$  trained using various loss combinations.

**The effectiveness of meta learning** First of all, the meta-learning loss  $\mathcal{L}_{meta}$  in Equation (4.4) is essential for the hallucinated features to preserve the categorical information of interest, and hence shall not be excluded; Otherwise, as shown in experiments (j) and (k), the downstream FSL performance will heavily degrade as the hallucinated features belong to arbitrary categories might be harmful to classification tasks. Interestingly, we also observe that  $\mathcal{L}_{meta}$  also prevents the degeneration problem in adversarial training between the appearance encoder  $E_a$  and the discriminator  $D_c$ , as all-zero hallucinated vectors

basically could not lead to improved performance for  $N$ -way  $K$ -shot tasks.



**The effectiveness of adversarial training** In experiments (b)-(f), we gradually include all the proposed loss components except the adversarial loss  $\mathcal{L}_{adv}^g$  in Equation (4.11). Compared to the Baseline, a marginal increment in top-5 accuracy can be observed from CUB and some cases of *mini*-ImageNet. However, there is often a large amount of performance degradation in terms of top-1 accuracy. The large performance gap between experiments (b)-(f) and the complete version (m) indicates that  $\mathcal{L}_{adv}^g$  plays a crucial role in our FDH-Net, which relies on good quality of class-appearance feature disentanglement.

**The effectiveness of consistency losses** In experiments (g)-(i) and (l), we include the adversarial loss  $\mathcal{L}_{adv}^g$ , and exclude other reconstruction and consistency loss components from our FDH-Net one type at a time to investigate their impacts on the performance. First of all, it can be seen that  $\mathcal{L}_{adv}^g$  helps the model to beat the Baseline in both top-1 and top-5 accuracy. While each consistency loss component has its own importance, intra-class consistency loss  $\mathcal{L}_{intra}$  in Equation (4.9) shows larger impacts on the performance, as can be seen from experiment (g). In experiments (l) and (m), we observe a negligible difference when training our framework with or without consistency losses  $\mathcal{L}_{ca}$  in Equation (4.7) and  $\mathcal{L}_{cf}$  in Equation (4.8). However, the full version of FDH-Net with both  $\mathcal{L}_{ca}$  and  $\mathcal{L}_{cf}$  included still shows a significant improvement in the coarse-grained *mini*-ImageNet dataset.

**Visualization** In Figure 4.8, we show the t-SNE visualization of hallucinated features and appearance information generated by the hallucinators trained with various loss combinations as those specified in Table 4.4. The column labels (a, e, f, g, h, and k) in Figure 4.8 indicate the experiment IDs used in Table 4.4. We select the same 5 novel

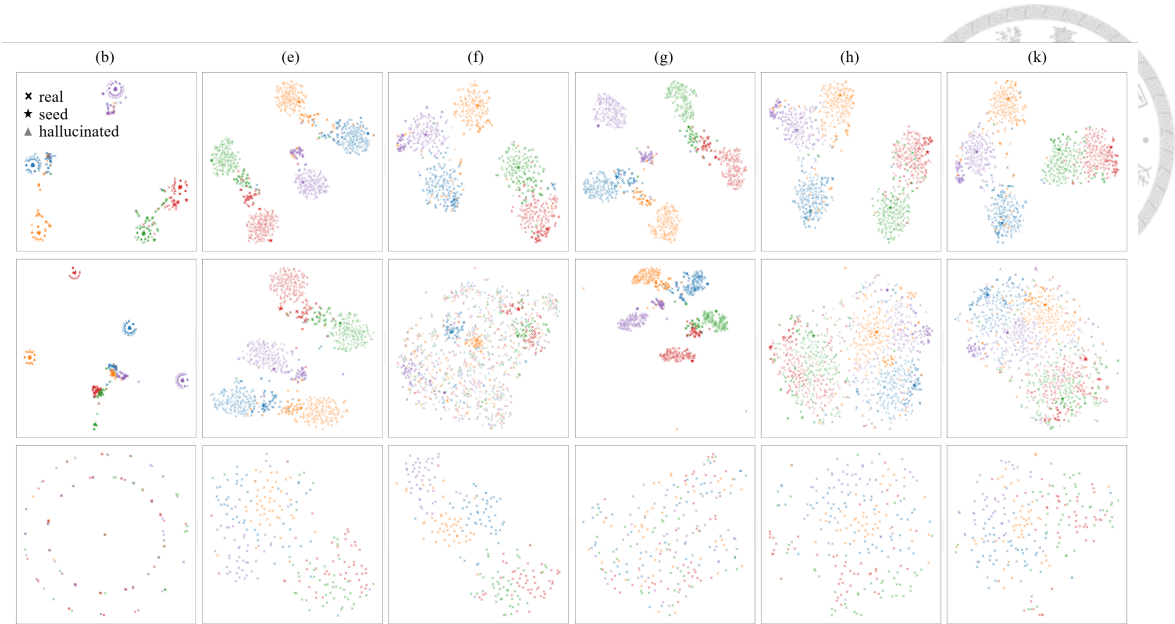


Figure 4.8: t-SNE visualization of hallucinated features and appearance information generated by various hallucinators trained with various loss combinations of our FDH-Net.

categories from CUB as those in Figure 4.5. The first row presents the real, seed, and hallucinated features in the class domain induced by the class encoder  $E_c$ , the second row presents the real, seed, and hallucinated features in the raw feature domain, and the third row presents the appearance information  $E_a(s)$  extracted from real features. The visualization in the middle panel of the last column is used in Figure 4.5.

It can be seen from the last column of Figure 4.8 that our FDH-Net is able to generate diverse yet discriminative hallucination not only in the class domain (the top panel) but also in the raw feature domain (the middle panel). Furthermore, our FDH-Net also leads to appearance information that is indistinguishable among the 5 categories (the bottom panel). This again confirms the effectiveness of our framework in class-appearance feature disentanglement. It is worth noting that the second to the last column (item (h): our FDH-Net without reconstruction loss  $\mathcal{L}_{recon}$  and consistency losses  $\mathcal{L}_{ca}$  and  $\mathcal{L}_{cf}$ ) shows similar visualization results, which further suggest that our FDH-Net is able to make decent hallucination and outperform previous most of the state-of-the-art hallucination ap-



proaches even without these three data-recovery constraints.



## 4.5 Summary

In this study, we propose a novel disentanglement-based data hallucination approaches to address the more general few-shot learning (FSL) tasks without semantic information among categories. Our proposed Feature Disentanglement and Hallucination Network (FDH-Net) disentangles input visual features into class-specific and appearance-specific factors, followed by feature hallucination for novel categories via combining the class information of interest with appearance information extracted from base categories. By leveraging meta-learning and feature disentanglement techniques, our model is able to fully exploit the whole base dataset to make hallucinations that are not only more *useful* for downstream FSL tasks, but also more *explainable* in the sense that every hallucinated feature has explicit appearance guidance. Extensive experiments on two fine-grained datasets (CUB and FLO) and two coarse-grained ones (*mini*-ImageNet and CIFAR-100) confirm the effectiveness of our model.



## **Part II**

# **Learning from Noisily-labeled Data**





## Chapter 5

# Set-Level Self-Supervised Learning from Noisily-Labeled Data

### 5.1 Overview

In this chapter, we deal with another kind of imperfect data: noisily-labeled training samples, and discuss noisy-label learning (NLL) tasks [22]. Noisy labels are ubiquitous in real-world imagery datasets due to human annotation errors or misleading/ambiguous visual contents. As discussed in Chapter 1, it has been shown in [18, 19] that deep neural networks (DNNs) can easily overfit to noisy labels and perform poorly on cleanly-labeled test data at inference time. It has also been shown in [22] that naive regularization techniques such as data augmentation [111], weight decay [112], dropout [113], and batch normalization [114] would not completely overcome the overfitting issue. Existing methods thus proposed more advanced NLL approaches such as estimating the noise transition matrix [58–63] or reweighting/selecting training samples [73–77, 88–91], aiming at counteracting the noise effect during training. On the other hand, self-supervised learning (SSL) techniques have been applied in [64–67] to deal with NLL tasks by designing

proper pretext tasks to pre-train the model without label supervision. However, existing SSL-based NLL approaches are performed at the *instance* level without accessing the (noisy) labels, and hence it is not clear whether such techniques would result in robust representations when we further take noisy supervision into account.

In this chapter, we extend the idea of SSL to the *set* level, and propose a novel set-level self-supervised learning (SLSSL) technique that fully utilizes the information contained in the noisy labels. Specifically, we *augment* a set (i.e., a mini-batch) of training samples by corrupting their (noisy) labels, and derive the *augmented* version of the learning models via a single-step model optimization. Next, we design our pretext task by enforcing multiple augmented models (derived from different label corruptions of the same mini-batch) to exhibit sufficient consistency, such that robustness against different types of label noise would be introduced to our learning model. The proposed data manipulation (i.e., label corruption of two classes at a time) allows us to design a novel estimation procedure for the noise transition matrix of the training dataset. Moreover, the proposed SLSSL can also be utilized to estimate the label quality of the training samples. As a result, the proposed learning scheme can be seen as an expectation-maximization (EM) algorithm, with the E-step focusing on model training and the M-step focusing on sample reweighting. Extensive experiments on synthetic and real-world noisily-labeled datasets confirm the effectiveness of our framework.

## 5.2 Problem Definition

In this study, we focus on image classification tasks. Suppose that we are given a training set  $\mathcal{D} = \{(x_n, \tilde{y}_n)\}$ , where  $x_n$  denotes the  $n$ -th image from the image space  $\mathcal{X}$ ,

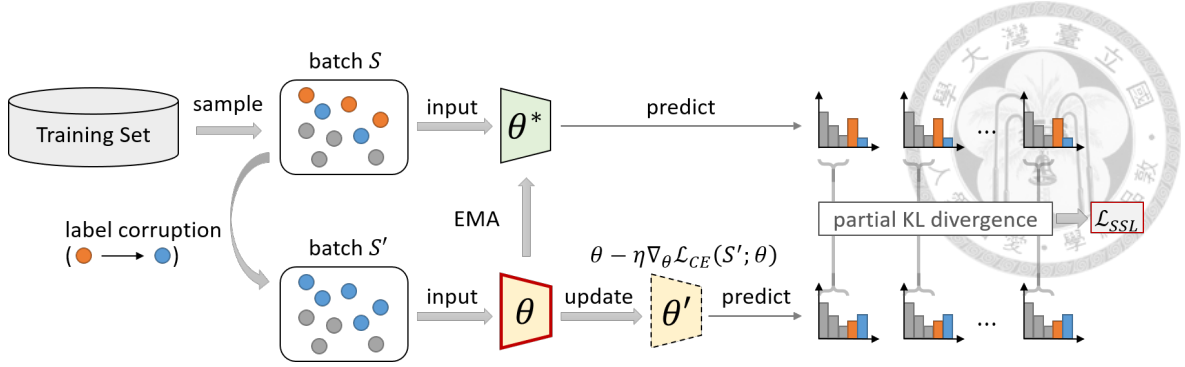


Figure 5.1: Illustration of our set-level self-supervised learning (SLSSL).

and  $\tilde{y}_n \in \{1, 2, \dots, C\}$  denotes the associated (noisy) label belonging to one of the considered  $C$  classes. Note that  $\tilde{y}_n$  is not necessarily consistent with the ground-truth label  $y_n$  of  $x_n$ . Let  $f(\cdot; \theta)$  denotes our learning model parameterized by  $\theta$ , which maps an image  $x_n$  to a  $C$ -dimensional probability space, and let  $f_c(\cdot; \theta)$  denotes the  $c$ -th dimension of  $f(\cdot; \theta)$ . That is, we have  $f_c(\cdot; \theta) \in [0, 1], \forall c$  and  $\sum_{c=1}^C f_c(\cdot; \theta) = 1$ . Our goal is to derive  $f(\cdot; \theta)$  from  $\mathcal{D}$  that performs well on a clean test set denoted by  $\mathcal{D}^t = \{(x_n^t, y_n^t)\}$ . As noted in Chapter 1, DNNs trained in a standard supervised-learning manner with classification objectives (e.g., categorical cross-entropy losses) can easily overfit to noisy labels and perform poorly on  $\mathcal{D}^t$ . To address this issue, we propose the following set-level self-supervised learning (SLSSL) technique as a building block to develop our NLL algorithms.

## 5.3 Set-Level Self-Supervised Learning

### 5.3.1 SLSSL Operation

Given a set of training samples (i.e., a mini-batch) denoted by  $S = \{(x_n, \tilde{y}_n)\}_{n=1}^N$  with size  $N$ , we propose to *augment* it by randomly selecting two classes  $i \neq j$  from  $\{1, 2, \dots, C\}$ , and then relabeling all the samples from label  $i$  using label  $j$ , as illustrated

in Figure 5.1. Note that we only change the labels from  $i$  to  $j$ , but not vice versa. The label-corrupted mini-batch can be thus denoted by  $S' = \{(x_n, \tilde{y}'_n)\}_{n=1}^N$ , where

$$\tilde{y}'_n = \begin{cases} j, & \text{if } \tilde{y}_n = i; \\ \tilde{y}_n, & \text{otherwise.} \end{cases} \quad (5.1)$$

We then use  $S'$  to derive the *augmented* model  $\theta'$  by applying single-step gradient descent to the learning model  $\theta$  with the cross-entropy classification loss computed using the corrupted labels. That is, we obtain  $\theta'$  by:

$$\theta' = \theta - \eta \nabla_{\theta} \sum_{n=1}^N \ell_{CE}(f(x_n; \theta), \tilde{y}'_n), \quad (5.2)$$

where  $\eta$  represents the learning rate used in such temporary single-step optimization, and  $\ell_{CE}$  denotes the cross entropy loss. Repeating the above procedure  $M$  times, we then obtain  $M$  augmented models from different label corrupted mini-batches of  $S$ , i.e., we observe  $\{\theta'_m\}_{m=1}^M$ , each corresponds to a specific version of label corruption (i.e, a specific choice of class pair  $(i, j)$ ). This set of *augmented* models  $\{\theta'_m\}_{m=1}^M$  would play an important role in guiding the learning of  $\theta$  in terms of both model training and sample reweighting, as will be detailed next in Section 5.3.2 and Section 5.3.3, respectively.

### 5.3.2 SLSSL for Model Training

**Enforcing model robustness against label corruptions** To make our learning model  $\theta$  more robust against various label *noises*, we first make it more robust against various label *corruptions* introduced in the above SLSSL operation. To achieve this, we enforce

all augmented models  $\{\theta'_m\}_{m=1}^M$  to give consistent prediction outputs on the same input  $x_n$ . Instead of pairwise comparing prediction outputs among all augmented models in  $\{\theta'_m\}_{m=1}^M$ , we utilize the exponential moving average (EMA) strategy to derive a *stable* model. Specifically, we follow MLNT [72] and maintain an EMA model  $\theta^*$  during training, with its parameters derived by self-ensemble of the learning model  $\theta$ :

$$\theta^* \leftarrow \alpha\theta^* + (1 - \alpha)\theta, \quad (5.3)$$

where  $\alpha$  is the hyperparameter controlling the update speed. The EMA model  $\theta^*$  is not affected by the label corruptions, and hence serves as a common target for all augmented models in  $\{\theta'_m\}_{m=1}^M$  to regularize their behaviors. That is, we enforce all  $\theta'_m(x_n)$ 's to be close to  $\theta^*(x_n)$  for all  $x_n$  from the mini-batch  $S$ .

Before formally defining the consistency constraints, we first provide an intuitive connection between label corruption and model robustness. As depicted in Figure 5.1, we derive  $\theta'_m$  by updating  $\theta$  on an augmented mini-batch  $S'$  with two classes  $i$  and  $j$  being randomly corrupted (say,  $i$  represents *bird* and  $j$  represents *airplane*), the predictions of  $f(\cdot; \theta'_m)$  for the remaining classes (e.g., *cars*) should *not* be altered by this label corruption. In other words, to exhibit the robustness of the learned model, its prediction outputs for classes other than  $i$  and  $j$  should be not sensitive to the above corruption. Thus, we modify the widely-used Kullback-Leibler (KL) divergence to define the SSL objective as:

$$\mathcal{L}_{SSL} = \frac{1}{MN} \sum_{m=1}^M \sum_{n=1}^N D_{KL}^p(f(x_n; \theta^*) \parallel f(x_n; \theta'_m)), \quad (5.4)$$

where  $D_{KL}^p$  represents the *partial* KL divergence computed by *excluding* the class dimen-



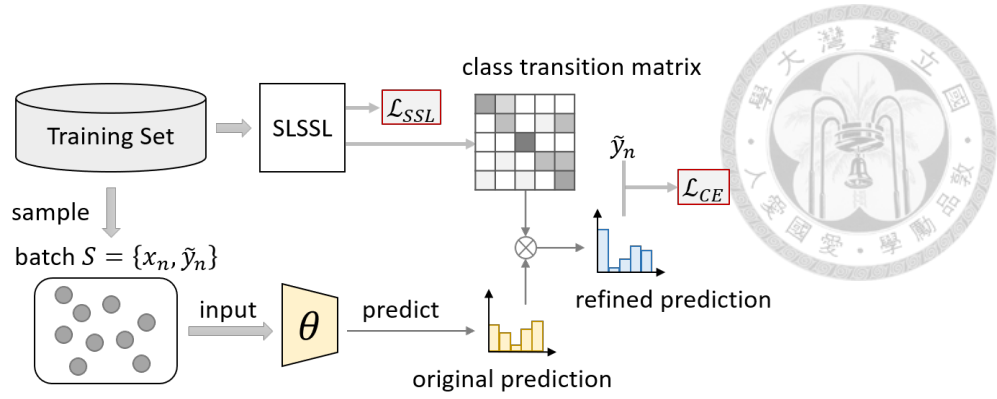


Figure 5.2: SLSSL for model training.

sions  $i$  and  $j$ , which are corrupted for obtaining  $\theta'_m$ . That is, we have:

$$D_{KL}^p(f(x_n; \theta^*) \parallel f(x_n; \theta'_m)) \triangleq \sum_{\substack{c=1 \\ c \neq i, j}}^C f_c(x_n; \theta^*) \log(f_c(x_n; \theta^*) / f_c(x_n; \theta'_m)). \quad (5.5)$$

It is worth noting that, this learning strategy can be viewed as a meta-learning technique introduced in MAML [27], which also imposes learning objectives on the single-step updated model (i.e.,  $\theta'_m$ ) for fast model adaptation. By optimizing the SSL loss defined in Equation (5.4), the learning model  $\theta$  (and also  $\theta^*$ ) would be encouraged to be more robust against various label corruptions, and hence would be expected to be more robust against various label noises.

**Estimating noise transition matrix for NLL** While the learning strategy proposed above can be viewed as a meta-learning technique, previous works like [58–63, 115, 116] further deal with NLL by estimating a class-wise  $C$ -by- $C$  noise transition matrix  $T$  for loss correction, as described in Section 2.2.2.1. Unfortunately, estimating  $T$  is a challenging task, especially when no clean training/validation sets are available (as the setting we considered here). We now explain how we solve this task using the above SLSSL

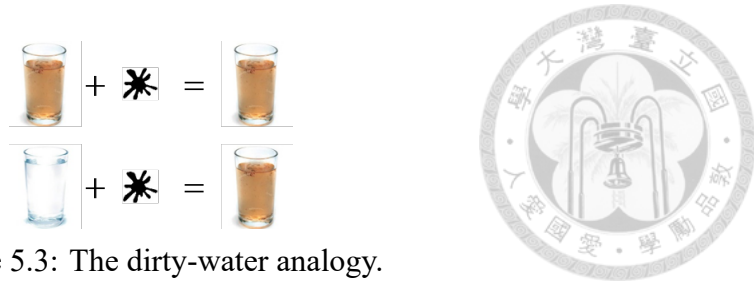


Figure 5.3: The dirty-water analogy.

technique.

Suppose that  $\theta'_m$  is derived by updating  $\theta$  on a mini-batch with classes  $(i, j)$  being corrupted, as described in Section 5.3.1. Intuitively, if the samples with true class  $i$  in the training dataset have a larger portion being mislabeled as  $j$  (say,  $i$  represents *bird* and  $j$  denotes *airplane*), the model  $\theta$  (and hence  $\theta^*$ ) trained on this dataset would have biased performance towards these two categories. Thus, the performance of  $\theta'_m$  would be *less* sensitive to the label corruption between these two classes. On the other hand, for the case where  $i$  and  $j$  are less likely to be confused (say,  $i$  represents *bird* and  $j$  represents *truck*), the associated  $\theta'_m$  would be expected to exhibit larger performance deviation.

As an analogy, consider adding a drop of ink into a glass of water, as illustrated in Figure 5.3. Intuitively, if the water is originally clean, a drop of ink will make the water become much dirtier. On the other hand, if the water is already dirty, the effect of adding a drop of ink might not be perceivable. Here, our learning model  $\theta$  (or its EMA version  $\theta^*$ ) is like a glass of water. By corrupting the labels in a mini-batch and applying single-step gradient descent (i.e., adding a drop of ink), we derived the *consequence* of such a label corruption in terms of the augmented model  $\theta'_m$  (i.e., another glass of water). The *impact* of such a label corruption can thus be measured by comparing  $\theta^*$  and  $\theta'_m$ , just like comparing the water before and after the ink pollution.

With the aforementioned observation and property, we propose to estimate  $T$  by measuring the deviation between  $\theta'_m$  and  $\theta^*$ . Specifically, we adapt the standard KL divergence

on each sample  $x_n$  in the mini-batch (across all  $C$  dimensions):



$$D_{KL}(f(x_n; \theta^*) \parallel f(x_n; \theta'_m)) = \sum_{c=1}^C f_c(x_n; \theta^*) \log(f_c(x_n; \theta^*)/f_c(x_n; \theta'_m)). \quad (5.6)$$

To estimate the  $(i, j)$ -th entry  $T_{ij}$ , we collect all  $\theta'_m$ 's that are derived by corrupting  $(i, j)$  across multiple mini-batches (e.g., within one training epoch), and compute the *inverse* of the averaged KL divergence as:

$$Q_{i,j} = \mathbb{E}_{(i,j)} [D_{KL}(f(x_n; \theta^*) \parallel f(x_n; \theta'_m))]^{-\tau}, \quad (5.7)$$

where  $\mathbb{E}_{(i,j)}$  indicates that the average is computed over all  $\theta'_m$ 's derived by corrupting  $(i, j)$ , and  $\tau$  is a sharpening parameter. Note that we also allow  $i = j$  in Equation (5.7) to cover the diagonal entries of  $T$ , with  $Q_{i,i}$  representing the average deviation between  $\theta^*$  and the model being updated using the *uncorrupted* mini-batch  $S$ . Since the sum of each row in  $T$  must equal to one (i.e.,  $\sum_{c=1}^C T_{ic} = 1, \forall i \in \{1, 2, \dots, C\}$ ), we simply normalize  $Q_{i,j}$  to obtain the  $(i, j)$ -th entry of  $\hat{T}$  as:

$$\hat{T}_{ij} = \frac{Q_{i,j}}{\sum_{c=1}^C Q_{i,c}}. \quad (5.8)$$

Figure 5.4 provides an example of the above procedure. As can be seen, corrupting *birds* and *airplanes* within each mini-batch produces a lower impact on the model (i.e., smaller  $D_{KL}$ ), reflecting the fact that a relatively larger portion of *birds* and *airplanes* are already highly mislabeled in the training dataset. This corresponds to the case of adding a

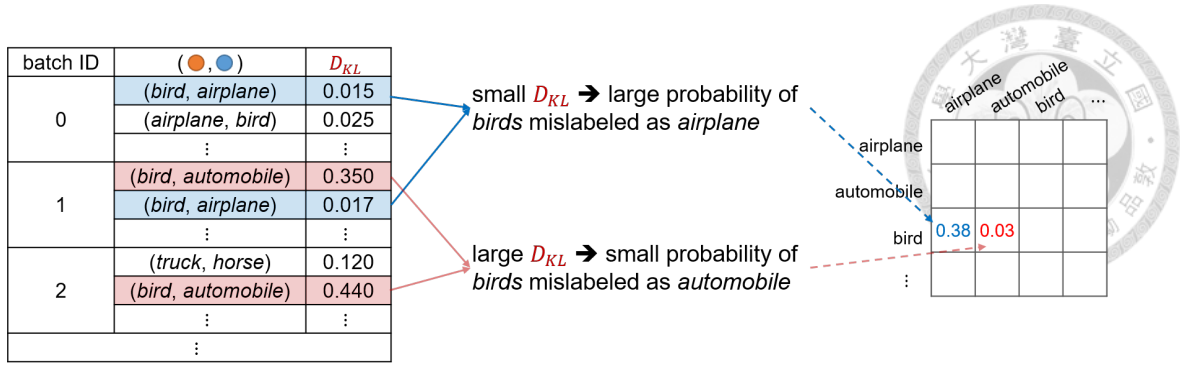


Figure 5.4: Example of SLSSL-based noise transition matrix estimation.

drop of ink to a glass of dirty water. On the other hand, corrupting *birds* and *automobiles* shows a higher impact (i.e., larger  $D_{KL}$ ), reflecting the fact that these two classes are less likely to be confused in the training dataset. This corresponds to the case of adding a drop of ink to a glass of clean water.

With the above derivation, we are able to estimate  $T$  on a per-epoch basis. To facilitate training stability, we further apply the EMA technique to accumulate the estimated matrices in different training epochs. Concretely, let  $\hat{T}_e$  denotes the matrix estimated within each epoch. The accumulated estimation of the noise transition matrix  $\hat{T}$  can be derived by  $\hat{T} = \beta\hat{T} + (1 - \beta)\hat{T}_e$ , with  $\beta \in [0, 1]$  controlling the update speed. Based on  $\hat{T}$ , we then follow forward loss correction methods [59, 62] and transform the model prediction output  $f(x_n; \theta)$  by multiplying it with  $\hat{T}$  before calculating the cross-entropy loss with respect to the noisy labels  $\{\tilde{y}_n\}$ , as illustrated in Figure 5.2. The *corrected* version of cross-entropy loss can then be expressed as:

$$\mathcal{L}_{CE} \triangleq -\frac{1}{N} \sum_{n=1}^N \tilde{y}_n \cdot \log(\hat{T}^t f(x_n; \theta)), \quad (5.9)$$

where the superscript  $t$  denotes matrix transpose. Finally, the total loss for model training in the proposed SLSSL is derived by combining  $\mathcal{L}_{SSL}$  in Equation (5.4) and  $\mathcal{L}_{CE}$  in

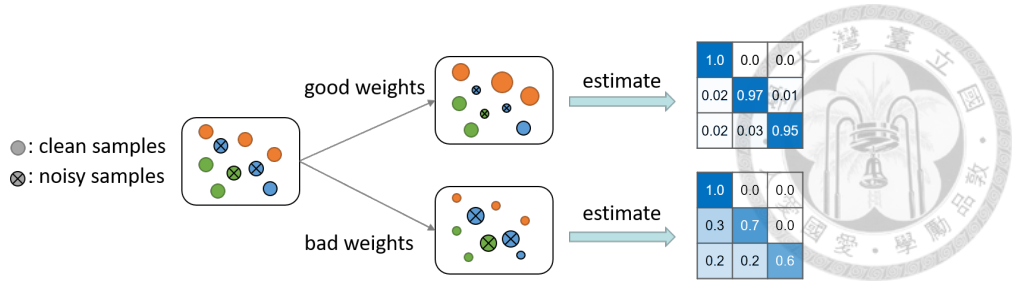


Figure 5.5: Good or bad sample weights.

Equation (5.9) as:

$$\mathcal{L}_E = \mathcal{L}_{CE} + \lambda_{SSL}\mathcal{L}_{SSL}, \quad (5.10)$$

where  $\lambda_{SSL}$  is a hyperparameter.

### 5.3.3 SLSSL for Sample Reweighting

As discussed in Chapter 2 and Section 5.1, another group of NLL works choose to reweight training samples based on their labeling confidence for training the learning model. We now explain how our SLSSL can also be utilized for sample reweighting as follows.

Before we delve into the SLSSL-based sample reweighting algorithm, we first discuss the relationship between sample weights and the estimated noise transition matrix, with the basic idea illustrated in Figure 5.5. By default, a noisily-labeled dataset is considered to have *good* weights if most of its samples with correct labels are assigned larger weights than those samples with incorrect labels. It has been shown in sample-reweighting approaches [81–87] that, by minimizing the *weighted* loss, a noisily-labeled dataset with such good weights would act like a cleanly-labeled dataset, as the incorrect labels contribute less than the correct labels during model training. Recall that the estimated noise

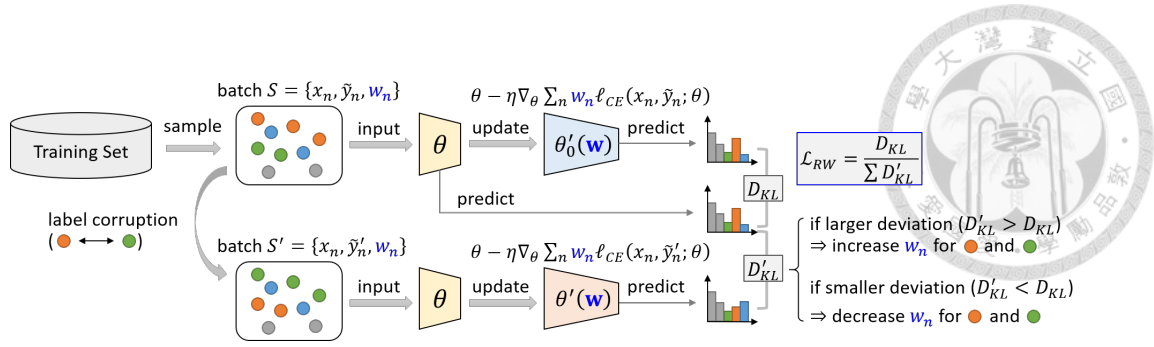


Figure 5.6: SLSSL for sample reweighting.

transition matrix  $\hat{T}$  can be applied for loss correction in NLL to counteract the noise effect in the training dataset. Intuitively,  $\hat{T}$  should be close to an *identity* matrix if most of the samples in the training dataset are correctly labeled, or equivalently, correctly-labeled samples are assigned with larger weights than those of noisily-labeled ones.

Formally, suppose that we are given a set (i.e., a mini-batch) of training samples  $S = \{(x_n, \tilde{y}_n, w_n)\}_{n=1}^N$ , where  $w_n$  denotes the weight of the  $n$ -th sample  $(x_n, \tilde{y}_n)$ . For sample reweighting, we follow a similar procedure of estimating the noise transition matrix as described in Section 5.3.2. However, we keep the model parameter  $\theta$  fixed and only update the *sample weights*  $\{w_n\}$  based on the estimated results, as illustrated in Figure 5.6. Based on the relationship between sample weights and the estimated noise transition matrix discussed above, the optimal weights in  $S$  should make  $\hat{T}_{ij}$  derived from Equation (5.6), (5.7), and (5.8) close to 0 for the case of  $i \neq j$  (i.e., the *off-diagonal* entries of  $\hat{T}$ ) and 1 for  $i = j$  (i.e., the *diagonal* entries of  $\hat{T}$ ).

To achieve this, we first use  $S$  to estimate the noise transition matrix as described in Sect. 5.3.2 with  $\{w_n\}$  included in the estimation process, and then optimize  $\{w_n\}$  based on the estimation results. Specifically, we follow the SLSSL procedure in Sect. 5.3.1 to create an augmented model set  $\{\theta'_m(\mathbf{w})\}_{m=1}^M$ , with each  $\theta'_m(\mathbf{w})$  derived by updating  $\theta$  using the *weighted* version of Equation (5.2):



$$\theta'_m(\mathbf{w}) = \theta - \eta \nabla_{\theta} \sum_{n=1}^N w_n \ell_{CE}(f(x_n; \theta), \tilde{y}'_n), \quad (5.11)$$

where  $\mathbf{w}$  denotes the collection of sample weights  $\{w_n\}$  within the mini-batch, and  $\tilde{y}'_n$  is obtained by corrupting a label pair  $i \neq j$  using Equation (5.1). To estimate  $\hat{T}$ , we follow Sect. 5.3.2 and compute the deviation between  $\theta'_m(\mathbf{w})$  and  $\theta$  by calculating the standard KL divergence  $D_{KL}(\theta \parallel \theta'_m(\mathbf{w}))$  using Equation (5.6). Since  $\theta'_m(\mathbf{w})$  is derived by corrupting a label pair  $i \neq j$ , this deviation thus corresponds to the off-diagonal entry  $\hat{T}_{ij}$ . We further let  $\theta'_0(\mathbf{w})$  denote the model derived by updating  $\theta$  through a single-step gradient descent based on the *original* mini-batch  $S$  (without label corruption), and we also compute its deviation from  $\theta$  (denoted by  $D_{KL}(\theta \parallel \theta'_0(\mathbf{w}))$ ) to represent all diagonal entries of  $\hat{T}$ . Since we estimate  $\hat{T}$  by taking inverse of the above deviations, for  $\hat{T}$  to be close to identity,  $D_{KL}(\theta \parallel \theta'_m(\mathbf{w}))$  should be much larger than  $D_{KL}(\theta \parallel \theta'_0(\mathbf{w}))$ .

Thus, our sample reweighting (RW) loss can be defined in the following contrastive form:

$$\mathcal{L}_{RW} = \frac{D_{KL}(\theta \parallel \theta'_0(\mathbf{w}))}{\sum_{m=1}^M D_{KL}(\theta \parallel \theta'_m(\mathbf{w}))}. \quad (5.12)$$

It can be seen that, the denominator in  $\mathcal{L}_{RW}$  describes the performance deviations between the original model and their augmented versions derived by multiple label corruptions. For a mini-batch with more clean samples, such label corruptions would lead to larger deviations, as compared to the case without label corruption (i.e., the numerator). We thus propose to minimize  $\mathcal{L}_{RW}$  for sample reweighting. In other words, samples leading to higher performance deviations from label corruptions are thus assigned with larger confi-

Table 5.1: Comparison between K-means clustering and our EM-like iterative training strategy in SLSSL.

	K-means	SLSSL
Initial E-step	Randomly initialize K centroids.	Randomly initialize model parameters, and train the model against noisy labels using samples (with equal weights).
M-step	Determine cluster membership for each sample based on the K centroids.	Determine the weights for all samples based on the trained model.
E-step	Re-compute K centroids based on the updated cluster membership.	Re-train the model based on the updated sample weights.

dences/weights. This serves as our sample reweighting strategy based on SLSSL.

### 5.3.4 SLSSL as an EM-like Algorithm

It is worth noting that, we can further integrate the proposed SLSSL-based model training (Sect. 5.3.2) and sample reweighting (Sect. 5.3.3) schemes as an EM-like algorithm for NLL. To be more specific, the E-step focuses on model training with sample weights being fixed, while the M-steps aims to reweight training samples with the derived model with parameters being fixed. In practice, we first randomly initialize  $\theta$  and train it from scratch by optimizing  $\mathcal{L}_E$  in Equation (5.10) on the unweighted training dataset ( $w_n = 1$  for all  $n$ ). After obtaining the best model through validation (usually the EMA model  $\theta^*$ ), we then fix its parameters and utilize it to optimize all  $\mathbf{w}$  by optimizing Equation (5.12). The updated sample weights are then applied in the model training again.

To give a more intuitive mathematical comparison, we take the K-means clustering as an example of EM algorithm, and compare it with our EM-like iterative learning strategy in Table 5.1. As confirmed by our experiment, this alternative optimization strategy would further boost the NLL performance.



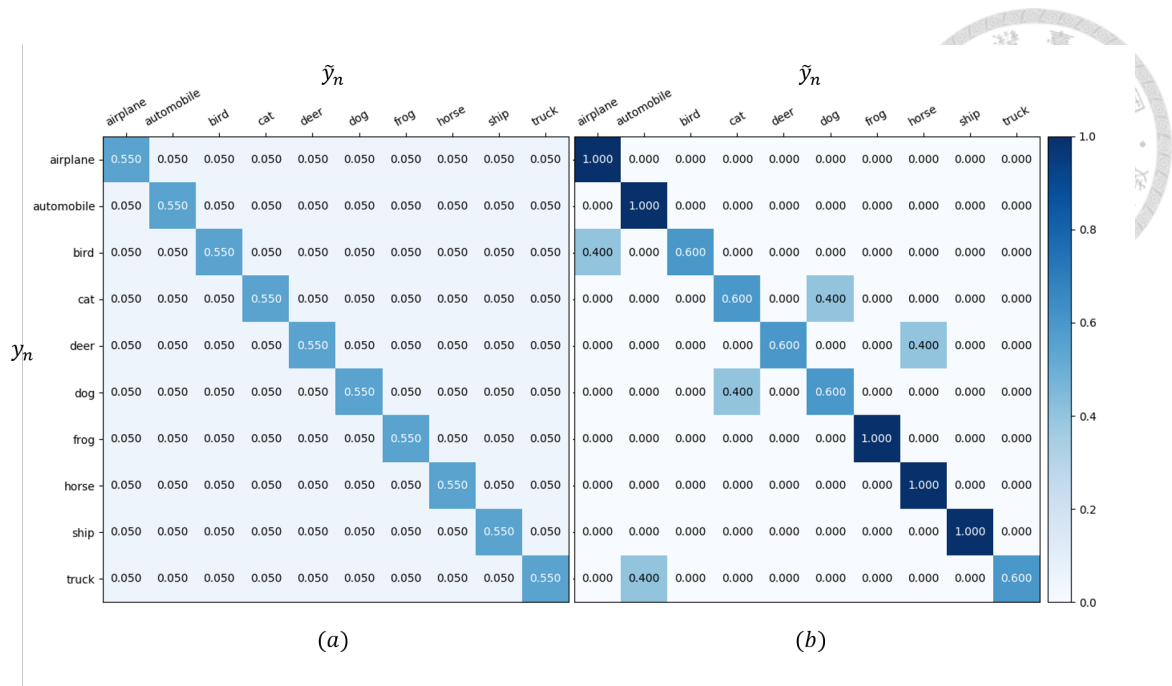


Figure 5.7: The ground-truth noise transition matrix utilized to construct noisy datasets from CIFAR-10.

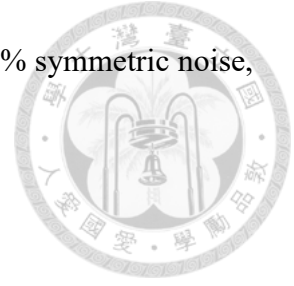
## 5.4 Experiments

### 5.4.1 Datasets

We follow previous NLL works [59, 72, 88, 89] to conduct experiments on two synthetic noisily-labeled datasets and two real-world noisily-labeled datasets, as described below.

**CIFAR-10** The CIFAR-10 dataset [96] contains 50K training images and 10K test images from 10 categories. We follow [72, 88] to corrupt the training labels by two types of noise: *symmetric* and *asymmetric*. For symmetric noise, each sample has a fixed probability to be mislabeled uniformly into the other 9 classes. As for asymmetric label noise, 5 noise transition patterns are proposed to simulate the class-dependent noise:  $bird \rightarrow airplane$ ,  $cat \leftrightarrow dog$ ,  $deer \rightarrow horse$ , and  $truck \rightarrow automobile$ . Figure 5.7(a) shows the noise

transition matrix used to construct a noisy CIFAR-10 dataset with 50% symmetric noise, and Figure 5.7(b) shows the one for 40% asymmetric noise.



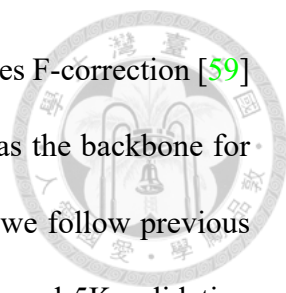
**CIFAR-100** The CIFAR-100 dataset [96] also consists of 50K training images and 10K test images. However, the number of categories (100) is much larger than CIFAR-10, and hence is considered a more challenging dataset in NLL. We follow DivideMix [88] and only consider symmetric noise patterns.

**CIFAR-10N** CIFAR-10N [117] is extended from CIFAR-10, with each training image containing three human-annotated labels (denoted as *Random- $i$* ,  $i \in \{1, 2, 3\}$ ). The three noisy labels for each image are further aggregated by majority voting (denoted as *Aggregate*) and randomly picking one wrong label if any (denoted as *Worst*).

**Clothing1M** Clothing1M [118] consists of 1M training images from 14 categories of clothes. The labels are extracted from the surrounding texts of images and are thus practically noisy. We follow DivideMix [88] and use the 14K clean validation set for hyperparameter tuning and report the prediction accuracy on the 10K clean test set.

## 5.4.2 Implementation details

We first conduct a preliminary experiment on CIFAR-10 using a single backbone to justify the proposed SLSSL. Since most state-of-the-art NLL approaches adapted double-model co-training techniques to boost performances [88, 89], we also integrate our SLSSL into such a co-training framework for fair comparisons on all the considered datasets.



**Single-model experiments** We first follow previous NLL approaches F-correction [59] and MLNT [72] and adapt a single Pre-Act ResNet-32 network [5] as the backbone for the synthetic noisy CIFAR-10 datasets. For training and validation, we follow previous works [72, 84] and split the 50K training samples into 45K training and 5K validation subsets. We then introduce symmetric or asymmetric label noise to the training subset as described above, and keep the labels of the 5K validation subset unchanged. All models are trained on the 45K noisy training subset, with hyperparameters being chosen based on the model performance on the 5K clean validation subset. After hyperparameter tuning, we then introduce label noises to all 50K training samples and rerun all experiments with the fixed hyperparameters for comparison.

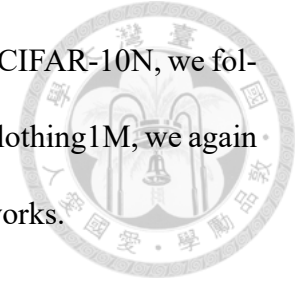
We develop our SLSSL algorithm based on the implementation of MLNT<sup>1</sup> [72], and follow most of its hyperparameters. Specifically, we train the model for 300 epochs using SGD with an initial learning rate 0.02 (divided by 2 after every 50 epochs), a momentum of 0.9, a weight decay of 0.0005, and a batch size of 128. The learning rate  $\eta$  used for obtaining the augmented models is set to 0.1 in Equation (5.2). For our SLSSL framework with both E and M steps, we set  $\eta = 0.001$  in Equation (5.11), and conduct the M-step for 5 iterations (3 M-step epochs per iteration) at every 50 E-step model training epochs: {50, 100, 150, 200, 250}.

As for the compared single-model methods, we train the model for 300 epochs using standard cross-entropy classification loss as the Baseline, and utilize the ground-truth noise transition matrix for F-correction [59]. The MLNT [72] results are simply reproduced by directly using their implementation.

---

<sup>1</sup><https://github.com/LiJunnan1992/MLNT>

**Double-model experiments** For both CIFAR-10, CIFAR-100, and CIFAR-10N, we follow DivideMix [88] to adapt two Pre-Act ResNet-18 networks. For Clothing1M, we again follow DivideMix and use two ImageNet-pretrained ResNet-50 networks.



To compare with current double-model NLL approaches, we integrate the proposed SLSSL into the current state-of-the-art method DivideMix. The original DivideMix algorithm trains two peer networks with a *warm-up* phase and a *co-training* phase. In the warm-up phase, the two networks are trained separately using standard classification objectives (i.e., cross-entropy loss). In the co-training phase, each network divides the training dataset into a labeled (clean) subset  $\mathcal{X}$  and an unlabeled (noisy) subset  $\mathcal{U}$  based on the small-loss principle (i.e., samples with smaller training losses are considered cleaner). The two networks are then trained collaboratively, with one network utilizing the data division  $\{\mathcal{X}, \mathcal{U}\}$  provided by the other network. This is called the *co-dividing* procedure, which can repeat for several training epochs to make the data division more reliable. We integrate SLSSL into DivideMix by applying the SLSSL-based sample reweighting algorithm described in Sec. 5.3.3 into the co-dividing procedure to enhance labeled/unlabeled data division.

We develop our SLSSL-based sample reweighting algorithm based on DivideMix’s official implementation<sup>2</sup>, and also follow most of their hyperparameters. For CIFAR-10 and CIFAR-10N, we first train two Pre-Act ResNet-18 networks separately for 10 epochs in the warm-up phase, and then enter the co-training phase for the remaining 290 epochs. At the 150 epoch, we replace DivideMix’s GMM-based reweighting procedure on the second network with our SLSSL-based sample reweighting algorithm (i.e., the M-step in Sect. 5.3.3), and conduct the M-step for 60 iterations (2 M-step epochs per iteration) at

---

<sup>2</sup><https://github.com/LiJunnan1992/DivideMix>

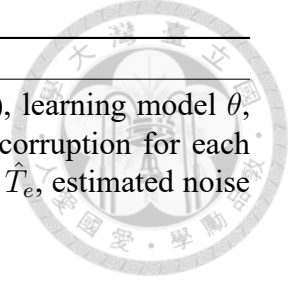
Table 5.2: Performance comparison on CIFAR-10 across different noisy labeling schemes with NLL methods based on a single learning model.

Method	No noise	Sym 50%	Sym 70%	Asym 30%	Asym 40%
Baseline (cross-entropy)	92.53±0.07	79.87±0.29	70.93±0.38	86.17±0.17	83.95±0.36
F-correction [59]	92.63±0.12	81.22±0.04	73.74±0.46	88.28±0.23	87.45±0.24
MLNT [72]	93.76±0.03	87.18±0.11	80.76±0.29	91.84±0.28	89.57±0.12
Our SLSSL (E-step only)	93.86±0.08	89.11±0.08	81.22±0.42	92.67±0.06	91.12±0.10
Our SLSSL (EM)	<b>94.17±0.02</b>	<b>90.13±0.16</b>	<b>83.91±0.30</b>	<b>92.94±0.16</b>	<b>91.78±0.12</b>

every 5 model training epochs:  $\{150, 155, \dots, 295\}$ . Based on the derived sample weights, we then follow standard DivideMix and apply dataset *co-dividing*, label *co-refinement*, and label *co-guessing* techniques to train the two networks. For CIFAR-100, we follow the above procedures except that the warm-up takes 30 epochs. As for Clothing1M, we also follow DivideMix and train our model on the 1M (noisy) training dataset, with hyperparameters being chosen based on the 14K (clean) validation subset. Similar to CIFAR-10/CIFAR-10N, we integrate our SLSSL algorithm into the co-training phase of DivideMix. We start our SLSSL-based sample reweighting at epoch 70. Since only 32K samples are randomly selected per training epoch, we conduct M-step for every epoch, and set the total epoch number in the co-training phase as 80. All the rest hyperparameters (e.g., learning rate scheduling) follow DivideMix.

**Algorithm** We provide the pseudo codes for our SLSSL in Algorithm 2 for model training (Section 5.3.2; E-step) and Algorithm 3 for sample reweighting (Section 5.3.3; M-step). The two algorithms can be combined as an EM-like iterative training strategy as described in Sect. 5.3.4.

### 5.4.3 Quantitative Results



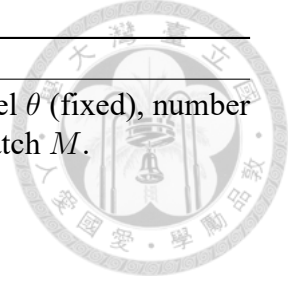
---

**Algorithm 2** SLSSL-E (model training in Sect. 5.3.2)

---

- 1: **Input:** Noisy training dataset  $\mathcal{D} = \{(x_n, \tilde{y}_n, w_n)\}$  (fixed  $\{w_n\}$ ), learning model  $\theta$ , EMA model  $\theta^*$ , number of E-step epochs  $P$ , number of label corruption for each mini-batch  $M$ , estimated noise transition matrix at the  $e^{th}$  epoch  $\hat{T}_e$ , estimated noise transition matrix  $\hat{T}$ .
  - 2: **if** first E-step **then**
  - 3:     Random initialize  $\theta$
  - 4:     Initialize EMA model  $\theta^* = \theta$
  - 5:     Initialize  $\hat{T}$  as identity
  - 6:     Initialize all sample weights  $w_n = 1$
  - 7: **end if**
  - 8: **for**  $e = 1$  to  $P$  **do**
  - 9:     **while** not done **do**
  - 10:         Sample mini-batch  $S = \{(x_n, \tilde{y}_n, w_n)\}_{n=1}^N$  from  $\mathcal{D}$
  - 11:         **for**  $m = 1$  to  $M$  **do**
  - 12:             Randomly sample a class pair  $(i, j)$
  - 13:             Obtain augmented set  $S' = \{(x_n, \tilde{y}'_n, w_n)\}_{n=1}^N$  by corrupting  $(i, j)$  in  $S$   
(Eq. 5.1)
  - 14:             Derive augmented model  $\theta'_m$  from  $\theta$  by single-step gradient descent on  $S'$   
(Eq. 5.2)
  - 15:             Compute the KL divergence  $D_{KL}$  between  $\theta'_m$  and  $\theta^*$  (Eq. 5.5)
  - 16:             **end for**
  - 17:             Compute set-level self-supervised loss  $L_{SSL}$  (Eq. 5.4)
  - 18:             Update  $\theta$  by minimizing  $L_{SSL}$
  - 19:             Compute classification loss  $L_{CE}$  and then correct it using  $\hat{T}$  (Eq. 5.9)
  - 20:             Update  $\theta$  by minimizing  $L_{CE}$
  - 21:             Aggregate  $D_{KL}$ 's derived by corrupting  $(i, j)$  to estimate the  $(i, j)^{th}$  entry of  $\hat{T}_e$  (Eq. 6)
  - 22:             **end while**
  - 23:             Update  $\hat{T}$  using EMA of  $\hat{T}_e$
  - 24:             Update  $\theta^*$  using EMA of  $\theta$
  - 25:     **end for**
- 

**Comparisons with single-model approaches** We first compare the proposed SLSSL with single-model NLL approaches using the same Pre-Act ResNet-32 backbone, including the simple baseline trained by standard supervised learning with categorical cross-entropy loss, F-correction [59] and MLNT [72]. The performances are summarized in Table 5.2. Here, we follow the same procedure in MLNT to generate symmetric and asymmetric noisy labels from CIFAR-10, and report the mean and standard error values of test accuracy across 3 runs. As can be seen from the table, with SLSSL-based model



---

**Algorithm 3** SLSSL-M (sample reweighting in Sect. 5.3.3)

---

- 1: **Input:** Noisy training dataset  $\mathcal{D} = \{(x_n, \tilde{y}_n, w_n)\}$ , learning model  $\theta$  (fixed), number of M-step epochs  $Q$ , number of label corruption for each mini-batch  $M$ .
  - 2: **for**  $e = 1$  to  $Q$  **do**
  - 3:     **while** not done **do**
  - 4:         Sample mini-batch  $S = \{(x_n, \tilde{y}_n, w_n)\}_{n=1}^N$  from  $\mathcal{D}$
  - 5:         **for**  $m = 1$  to  $M$  **do**
  - 6:             Randomly sample a class pair  $(i, j)$
  - 7:             Obtain augmented set  $S' = \{(x_n, \tilde{y}'_n, w_n)\}_{n=1}^N$  by corrupting  $(i, j)$  in  $S$   
(Eq. 5.1)
  - 8:             Derive  $\theta'_m(\mathbf{w})$  from  $\theta$  by weighted single-step gradient descent on  $S'$   
(Eq. 5.11)
  - 9:             Derive  $\theta'_0(\mathbf{w})$  from  $\theta$  by weighted single-step gradient descent on  $S$   
(Eq. 5.11)
  - 10:         **end for**
  - 11:         Compute the sample reweighting loss  $L_{RW}$  (Eq. 5.12)
  - 12:         Update  $\{w_i\}_{i=1}^N$  by minimizing  $L_{RW}$
  - 13:     **end while**
  - 14: **end for**
- 

training along (i.e., E-step in Sect. 5.3.2), our framework achieved significant performance improvement across all noise settings. By further applying sample reweighting (i.e., M-step in Sect. 5.3.3), the performances were further improved by a large margin, which confirms the effectiveness of our proposed SLSSL as a sample reweighting strategy.

**Comparisons with state-of-the-art double-model approaches** Next, we compare the performance of SLSSL with recent state-of-the-art methods utilizing two networks in the co-training fashion, including DivideMix [88], and DM-AugDesc [89]. Table 5.3 lists the performance comparisons on the synthetic noisy datasets generated from CIFAR-10. Following DivideMix and DM-AugDesc, we report both the *highest* test accuracy across all training epochs (denoted as *best*), and the average test accuracy over the *last* 10 epochs (denoted as *last*). As can be seen from the table, our frameworks performed favorably against state-of-the-art methods, including DivideMix on which we apply our SLSSL. It is worth noting that, DM-AugDesc also adapted DivideMix and focused on searching for

Table 5.3: Performance comparison on CIFAR-10 across different noisy labeling schemes with NLL methods adopting double models (i.e., co-training based approaches).

Noise type		Sym				Asym	Mean
Noise rate		20%	50%	80%	90%	40%	
DivideMix [88]	Best	96.1	94.6	93.2	76.0	93.4	90.66
	Last	95.7	94.4	92.9	75.4	92.1	90.10
DM-AugDesc [89]	Best	96.3	95.6	93.7	35.3	94.4	83.06
	Last	96.2	95.4	93.6	10.0	94.1	77.86
SLSSL (E-step only, single-model)	Best	93.5	90.3	76.4	57.3	92.3	81.96
	Last	93.1	90.1	76.1	57.0	91.8	81.62
SLSSL (EM single-model)	Best	95.8	94.4	93.2	67.0	93.1	88.70
	Last	95.2	94.0	92.8	61.5	92.4	87.18
SLSSL (EM double-model)	Best	96.3	95.0	93.4	77.5	94.2	<b>91.28</b>
	Last	96.2	94.8	93.2	76.7	93.9	<b>90.96</b>

Table 5.4: Performance comparison on CIFAR-10N with different human-labeling schemes.

Noise type		<i>Aggregate</i>	<i>Random-1</i>	<i>Random-2</i>	<i>Random-3</i>	<i>Worst</i>	Mean
DivideMix [88]	Best	95.2	95.5	95.5	95.5	93.0	94.94
	Last	95.0	95.1	95.2	95.2	92.6	94.62
Our SLSSL	Best	95.7	95.5	95.8	95.3	93.3	<b>95.12</b>
	Last	95.6	95.3	95.5	95.1	93.1	<b>94.92</b>

the *best* augmentation strategies for NLL. On the other hand, we only apply standard augmentation techniques such as random cropping and horizontal flipping in our experiments.

Table 5.4 and Table 5.5 list the performance comparisons between SLSSL and DivideMix [88] on the CIFAR-10N and CIFAR-100 datasets, respectively. Note that for each column in Table 5.5, we report the results from a single run using the same random seed for DivideMix [88] and our SLSSL to ensure the same noise setting. It can be seen that our method outperformed DivideMix on these challenging datasets, including the human-annotated one (CIFAR-10N) and the one that contains a much larger number of categories (CIFAR-100). Table 5.6 further presents results on the Clothing1M dataset, which confirms the effectiveness of our proposed SLSSL over state-of-the-art methods. Note that DM-AugDesc [89] is designed to search for best augmentation techniques for



Table 5.5: Performance comparison on CIFAR-100 across different noisy labeling schemes with NLL methods adopting dual models (i.e., co-training based approaches).

Method		Sym 20%	Sym 50%
DivideMix [88] (reproduced)	Best	77.50	74.20
	Last	77.00	73.80
Our SLSSL	Best	<b>78.08</b>	<b>74.34</b>
	Last	<b>77.83</b>	<b>73.95</b>

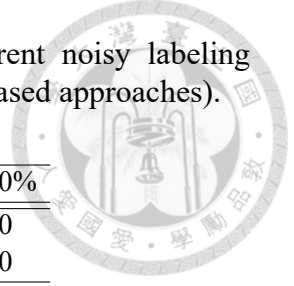


Table 5.6: Performance comparison on Clothing1M.

Method	Test Accuracy
Cross-Entropy	69.21
F-correction [59]	69.84
MLC [62]	71.06
MLNT [72]	73.47
T-Revision [61]	74.18
DivideMix [88]	74.76
DM-AugDesc [89]	75.11
Jo-SRC [66]	<b>75.93</b>
DivideMix [88] (reproduced)	74.34
Our SLSSL	<b>74.51</b>

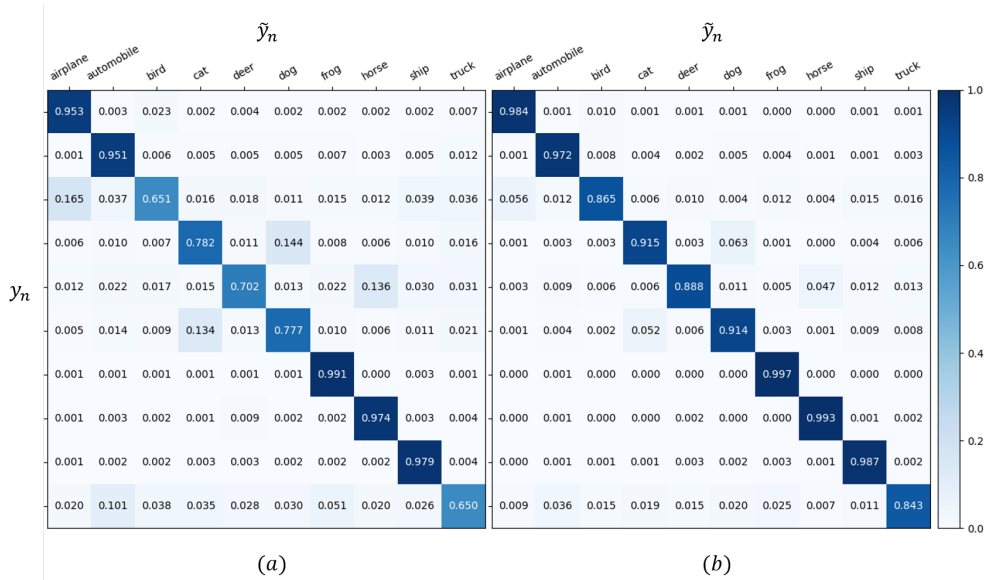


Figure 5.8: The estimated noise transition matrix for CIFAR-10 with 40% asymmetric noise.

NLL, while we simply perform random cropping and flipping in SLSSL.

#### 5.4.4 Visualization and Analysis

**Noise transition matrix for NLL** We show the noise transition matrix  $\hat{T}$  estimated based on Sec. 5.3.2 on the CIFAR-10 dataset with 40% asymmetric noise. We first trained a model by applying our E-step for 100 epochs, and then conducted the M-step for 20 epochs to assign weights for all training samples. Based on the sample weights, we then trained our model again using E-step for 100 epochs. As can be seen from Figure 5.8(a), the noise transition matrix estimated after the first E-step is fairly close to the ground-truth showed in Figure 5.8(b). In Figure 5.7(b), on the other hand, the estimated noise transition matrix becomes much closer to an identity matrix after a complete E-M-E cycle, indicating the effectiveness of our M-step in assigning sample weights. To quantitatively measure the *closeness* between noise transition matrices, we adopt the L2 distance:

$$d_2(A, B) = \sqrt{\sum_{i=1}^C \sum_{j=1}^C (a_{ij} - b_{ij})^2}, \quad (5.13)$$

where  $a_{ij}$  and  $b_{ij}$  are the  $(i, j)$ -th entries of  $C$ -by- $C$  square matrices  $A$  and  $B$ , respectively. Let  $\hat{T}_a$  denote the estimated matrix showed in Figure 5.8(a) and  $\hat{T}_b$  denote the one showed in Figure 5.8(b). We have  $d_2(\hat{T}_a, T_{asym0.4}) = 0.6759$  and  $d_2(\hat{T}_b, T_{asym0.4}) = 1.0127$ , where  $T_{asym0.4}$  denotes the ground-truth matrix used to introduce 40% asymmetric noise into the CIFAR-10 dataset. We also have  $d_2(\hat{T}_a, I) = 0.7424$  and  $d_2(\hat{T}_b, I) = 0.2981$ , where  $I$  denotes the  $C$ -by- $C$  identity matrix. From the above quantitative results, we see that the estimated transition matrix indeed becomes closer to  $I$  after sample reweighting.

To further validate the proposed SLSSL-based noise transition matrix estimation, we also show the estimated result for the original CIFAR-10 dataset without introducing any label noise. As can be seen from Figure 5.9, the estimated noise transition matrix shown in Figure 5.9(b) is indeed fairly close to an identity matrix shown in Figure 5.9(a).

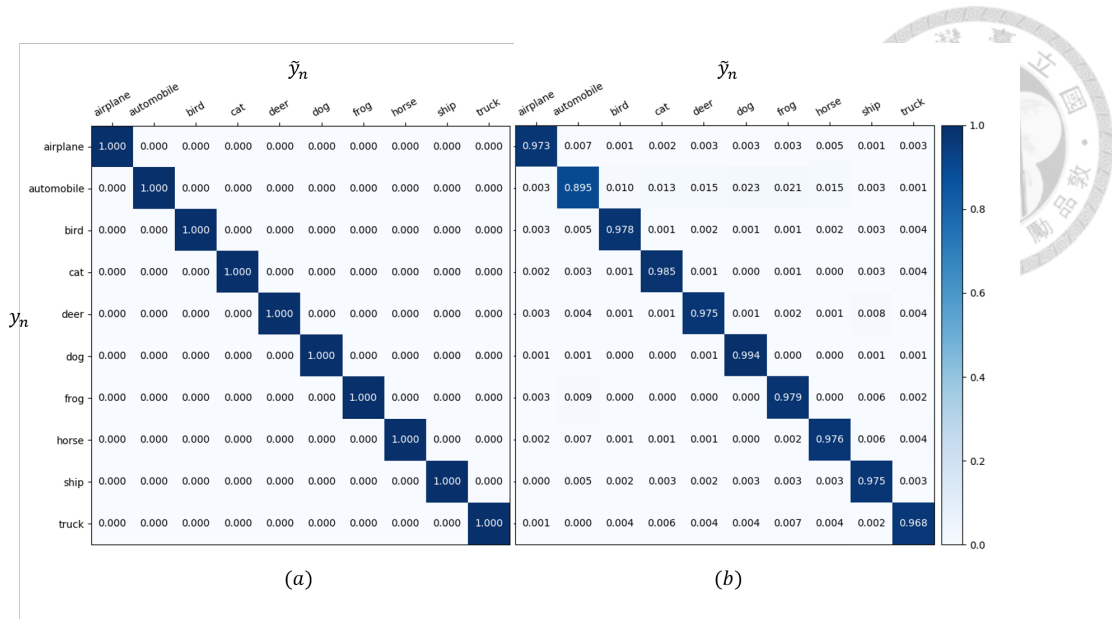


Figure 5.9: The ground-truth and the estimated noise transition matrix for the original CIFAR-10 dataset without label noise.

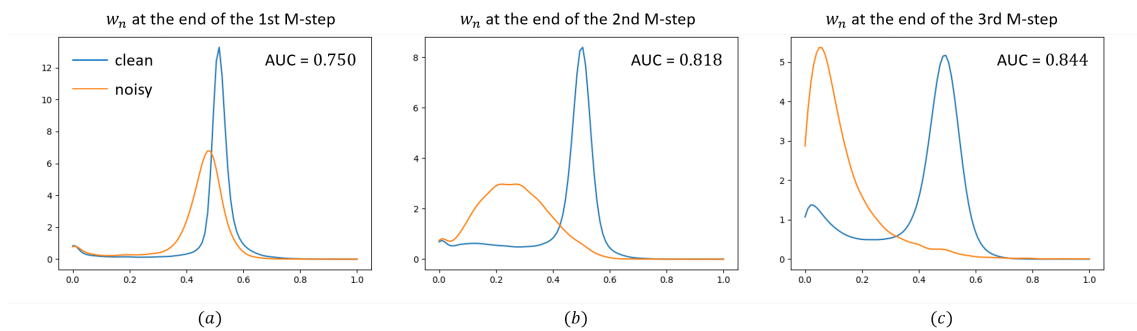


Figure 5.10: Empirical distributions of sample weights derived at the end of three consecutive M-steps on CIFAR-10 with 40% asymmetric noise.

**Rewighted samples** Here, we show the sample weights optimized based on our SLSSL at the end of three successive M-steps on CIFAR-10 with 40% asymmetric noise. As can be seen from Figure 5.10, as the optimization progresses (from (a)-(c)), the sample weights of clean and noisy samples become increasingly separable (with AUC scores reported in the figure). This further confirms the effectiveness of our SLSSL for sample reweighting in NLL.

**Validate EM-like iterative training strategy** We provide additional training statistics to further validate the proposed methods. In Figure 5.11(a), we show the Area Under the

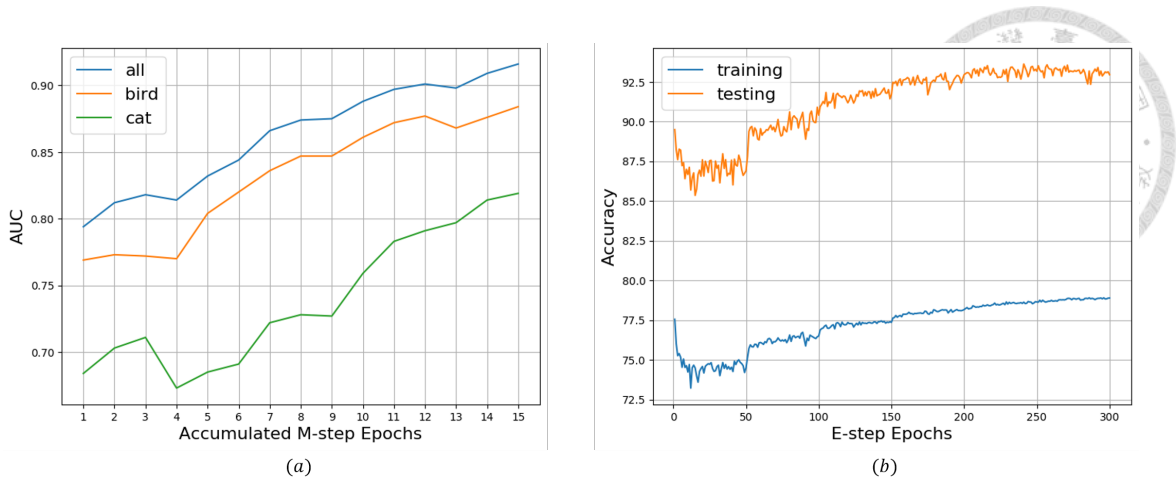
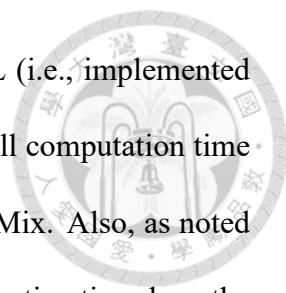


Figure 5.11: Learning curve on CIFAR-10 with 40% asymmetric label noise.

Curve (AUC) for clean/noisy sample classification based on the sample weights  $\{w_n\}$  obtained by our M-steps on CIFAR-10 with 40% asymmetric label noise in the single-model experiment. The X-axis indicates the *accumulated* M-step epochs over 5 iterations (3 M-step epochs per iteration) at E-step epochs from 50 up to 250. The AUC's are computed over all 50,000 training samples, 5000 bird samples, and 5000 cat samples, respectively. As can be seen, our M-step is able to effectively identify clean/noisy samples across successive M-steps, even for *bird* samples with 40% samples being mislabeled as *airplane*, and for *cat* samples which could be heavily confused with *dog* samples with 40% samples in each class being mislabeled as one another. In Figure 5.11(b), we show the training and testing accuracy values on the first network over all 300 training epochs. As can be seen, the accuracy value is increased after each iteration of M-step at epochs 50 and above, further validating the effectiveness of our SLSSL-based sample reweighting procedure.

### 5.4.5 Limitations

Since our proposed SLSSL can be viewed as a unique meta-learning scheme on existing NLL methods like DivideMix [88], we expect longer computation time during training. However, take CIFAR-10 dataset as an example, DivideMix took 15 hours to train using



a single Nvidia TITAN-V GPU, while the full version of our SLSSL (i.e., implemented as an EM algorithm) required 25 hours. It can be seen that, the overall computation time of our SLSSL is still in the same order as that of SOTAs like DivideMix. Also, as noted in [59–63], NLL methods based on class-wise noise transition matrix estimation share the limitation that the number of classes for NLL would be reasonable (e.g., 100 in CIFAR-100). This is to avoid an inaccurate estimation of a large noise transition matrix. Sharing the concern of the above works, this would also be among the current limitation of our work.

## 5.5 Summary

In this study, we proposed set-level self-supervised learning (SLSSL) to address noisy-label learning (NLL) tasks. SLSSL designs a pretext task at the set level by corrupting the labels within each training mini-batch and applying corresponding consistency constraints to enforce the model to exhibit sufficient robustness against label noise. By corrupting labels in a structured way, the proposed SLSSL can be utilized to estimate the noise transition matrix of the whole training dataset to counteract its noise effect during training. Moreover, the procedure of estimating the noise transition matrix can also be reversed for sample reweighting based on a novel high-impact principle. Finally, the proposed SLSSL-based model training and sample reweighting algorithms can be combined as an EM-like algorithm for improved NLL performance. With experiments conducted on synthetic (CIFAR-10 and CIFAR-100) and real-world (CIFAR-10N and Clothing1M) noisily-labeled datasets, the effectiveness of our proposed framework can be successfully confirmed.

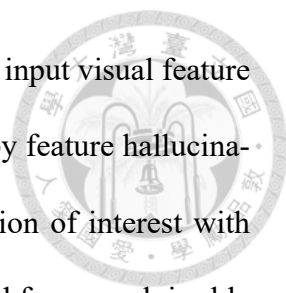


## Chapter 6

# Conclusion and Future Works

As stated in Chapter 1, the ultimate goal of the studies presented in this dissertation is to design robust and reliable deep-learning frameworks for visual classification tasks with imperfect training data by exploring and shifting advanced learning paradigms exhibited in human learning scenarios. Throughout the dissertation, we focused on learning from limited data and noisily-labeled data, and argued that the current deep-learning solutions did not fully utilize the side information or intrinsic data structures available during training. As imperfect data are inevitable and ubiquitous in nature, it is important to investigate other possibilities and opportunities for further performance improvement in this field.

To this end, in the first part of this dissertation (Chapter 3 and 4), we proposed novel data hallucination frameworks to deal with few-shot learning (FSL) tasks by leveraging prior knowledge and meta-learning techniques. Specifically, in the first study (Chapter 3), we proposed a novel data hallucination framework that incorporates semantic information to augment training data for few-shot categories. The augmented data would exhibit semantics-oriented modes of variation. Experiments show that our method quantitatively and qualitatively performed favorably against baseline and recent hallucination approaches. In the second study (Chapter 4), we proposed a novel disentanglement-based

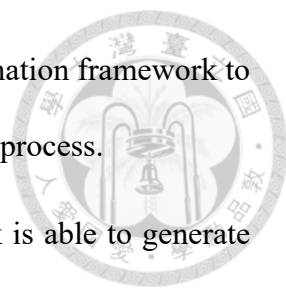


data hallucination framework. The proposed framework disentangles input visual feature into *categorical*-specific and *appearance*-specific factors, followed by feature hallucination for few-shot categories via combining the categorical information of interest with appearance information extracted from base categories. The proposed framework is able to fully exploit the base dataset to make hallucinations that are not only more *useful* for downstream FSL tasks, but also more *explainable* in the sense that every hallucinated feature has explicit appearance guidance.

In the second part of this dissertation (Chapter 5), we proposed a novel set-level self-supervised learning (SLSSL) framework to deal with noisy-label learning (NLL) tasks by advancing the self-supervised learning paradigm with intrinsic data structures with noisy supervision. The proposed SLSSL technique can be utilized as a building block to develop advanced algorithms for NLL. Specifically, we have applied SLSSL to train the model by estimating the noise transition matrix of the whole training dataset to counteract the noise effect. The procedure of the above matrix estimation can be further reversed to design a novel sample reweighting algorithm, which is free from the traditional small-loss assumption that may suffer from confirmation bias and favor early-encountered training samples. Furthermore, the proposed SLSSL-based model training and sample reweighting algorithms can be combined as an Expectation-Maximization (EM) like iterative learning framework for boosted NLL performance. Experiments on synthesized and real-world noisily-labeled datasets confirm the effectiveness of the proposed framework.

**Detailed contributions** We list the detailed contributions of this dissertation as follows.

- Part I: Learning from limited data – few-shot learning

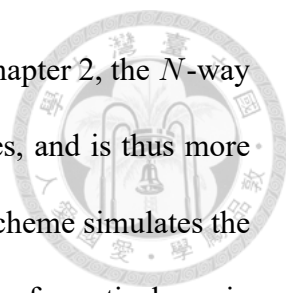
- 
1. We have proposed a novel semantics-guided data hallucination framework to leverage side semantic information into the hallucination process.
  2. The proposed semantics-guided hallucination framework is able to generate more reasonable training samples in the sense of hallucinated samples would exhibit semantics-oriented modes of variation.
  3. By utilizing the proposed semantics-guided hallucination framework, more reasonable and useful training samples can be generated for data-scarce categories for improved FSL performances.
  4. The effectiveness of the proposed semantics-guided hallucination framework can be successfully confirmed both quantitatively and qualitatively by our experiments on CIFAR-100 and Animals with Attributes datasets.
  5. We have proposed a novel feature-disentanglement based data hallucination framework named Feature Disentanglement and Hallucination Network (FDH-Net) by leveraging meta-learning and feature-disentanglement techniques.
  6. The proposed FDH-Net allows hallucination using appearance information extracted from base categories, while still preserving the categorical information from the associated data-scarce category.
  7. By sampling data pairs from arbitrary categories with no or a minimal similarity constraint, the proposed FDH-Net exploits the associated categorical and appearance information for data hallucination that achieves good classification performance.
  8. The effectiveness of the proposed FDH-Net can be successfully confirmed from both fine-grained (CUB and FLO) and coarse-grained (*mini*-ImageNet and CIFAR-100) FSL tasks.



• Part II: Learning from noisily-labeled data – noisy-label learning

1. We have proposed a novel set-level self-supervised learning (SLSSL) technique by extending the self-supervised learning technique to the set level.
2. The proposed SLSSL technique leverages intrinsic data structures with noisy supervision to design novel self-supervised pretext tasks by augmenting a set of training samples and imposing consistency constraints to improve the robustness of the learning models against noisy labels.
3. The proposed SLSSL technique allows us to estimate the associated noise transition matrix when training NLL models to counteract the label noise, without the need for strong statistical assumptions or collecting an extra set of clean samples.
4. The proposed SLSSL technique can also be utilized to identify the label quality of each training sample, and thus sample reweighting for NLL can be performed accordingly.
5. The proposed SLSSL-based model training and sample reweighting algorithms can be further combined and viewed as an EM-like algorithm, in which the E-step focuses on optimizing the model parameters with fixed sample weights, while the M-step focuses on sample reweighting based on the fixed model.
6. The effectiveness of the proposed SLSSL technique can be successfully confirmed from both synthesized (CIFAR-10/CIFAR-100) and real-world (CIFAR-10N/Clothing1M) NLL tasks.

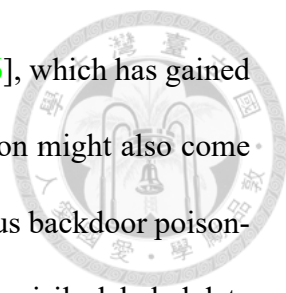
**Future works** As for possible future research opportunities, the research in this dissertation can be extended in the following dimensions.



For the research of learning from limited data, as we noted in Chapter 2, the  $N$ -way  $K$ -shot scheme focuses on a large number of small learning episodes, and is thus more demonstrated-oriented. On the other hand, the multi-class few-shot scheme simulates the scenario of deploying a recognition system in the wild, and is thus of practical use in real-world settings. Further exploiting the differences between both FSL schemes and leveraging their respective advantages would be an interesting direction for future work.

As we noted in Chapter 2, the proposed set-level self-supervised learning technique is a set-level extension of instance-discrimination based self-supervised learning [69–71]. Limited by the data structure (i.e., an *unordered set* of training samples along with their noisy labels), it would be more straightforward to augment the data by corrupting labels and imposing consistency constraints on the associated models. On the other hand, a much richer family of pretext tasks has been proposed at the instance level to learn general-purposed representations, such as predicting image rotations [68], solving patch-level jigsaw puzzles [119], and image colorization [120]. It would be an interesting and challenging task to extend such instance-level self-supervised learning techniques to the set level, and design more advanced data manipulations and pretext tasks at the set level for further opportunities.

In this dissertation, we focused on learning from limited data and noisily-labeled data, and discussed FSL and NLL for visual classification tasks only. However, other sources of data imperfection during training are also important research topics. For example, training data may be both limited in number and noisily labeled, leading to the challenging task of few-shot learning with noisy labels [121, 122]. In FSL, the sets of base and novel categories may be in different domains, leading to another challenging task of cross-domain few-shot learning [123, 124]. For NLL, more practical settings can

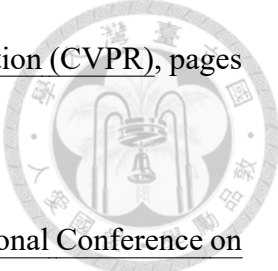


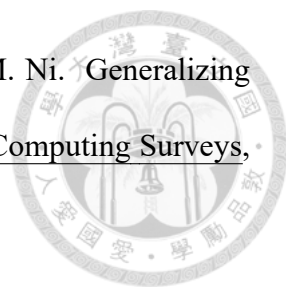
also be considered, such as instance-dependent noise (IDN) [125, 126], which has gained much attention in recent years. Besides label noises, data imperfection might also come from the image space, such as object occlusion [127] or even malicious backdoor poisoning [128]. Furthermore, other computer vision tasks with limited and noisily-labeled data are also worth paying attention to, such as object detection, semantic segmentation, and even multi-modality tasks that involve both visual and lingual information. We could also dig deeper into the *source* of the imperfect data, and further discuss the noise profiling (i.e., "what kinds of label noise do we have in the data?") or annotator modeling (i.e., "what makes an annotator make mistakes and how to avoid that?") problems. We believe that, despite the recent surge of success in the field of deep learning for computer vision, it is still a blooming research area for us to shrink the performance gap between machines and humans when addressing learning problems with various kinds of imperfect data.

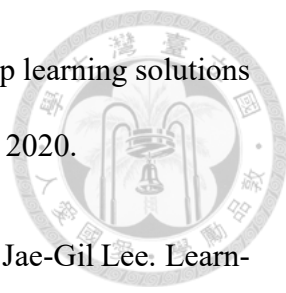


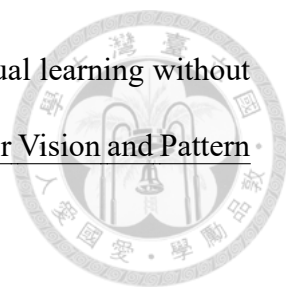
## References

- [1] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 3018–3027, 2017.
- [2] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7278–7286, 2018.
- [3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.
- [4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, 2015.
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.
- [6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of

- 
- the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 580–587, 2014.
- [7] Ross Girshick. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 1440–1448, 2015.
- [8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems 28 (NIPS 2015), pages 91–99, 2015.
- [9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 779–788, 2016.
- [10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3431–3440, 2015.
- [11] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05587, 2017.
- [12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 2961–2969, 2017.
- [13] Gary Marcus. Deep learning: A critical appraisal. arXiv:1801.00631, 2018.
- [14] Amina Adadi. A survey on data-efficient algorithms in big data era. Journal of Big Data, 8, 2021.

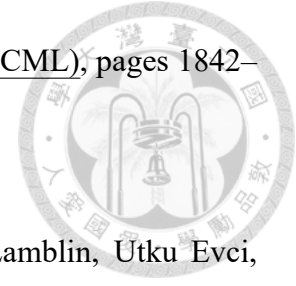
- 
- [15] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys, 53, 2020.
- [16] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. arXiv:1708.02862, 2017.
- [17] Dhruv Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In European Conference on Computer Vision (ECCV), 2018.
- [18] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In International Conference on Learning Representations (ICLR), 2017.
- [19] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In Proceedings of the 34th International Conference on Machine Learning (ICML), pages 233–242, 2017.
- [20] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 681–687, 2015.
- [21] Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N. Chiang, Zhihao Wu, and

- 
- Xiaowei Ding. Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. Medical Image Analysis, 63, 2020.
- [22] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. IEEE Transactions on Neural Networks and Learning Systems, pages 1–19, 2022.
- [23] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9):5149–5169, 2022.
- [24] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(11):4037–4058, 2021.
- [25] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In Advances In Neural Information Processing Systems 29 (NIPS 2016), pages 3630–3638, 2016.
- [26] Sebastian Thrun and Lorien Y. Pratt. Learning To Learn. Kluwer Academic Publishers, Boston, MA, 1998.
- [27] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning (ICML), pages 1126–1135, 2017.
- [28] Hang Qi, Matthew Brown, and David G. Lowe. Low-shot learning with imprinted weights. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5822–5830, 2018.


- 
- [29] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4367–4375, 2018.
- [30] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In International Conference on Learning Representations (ICLR), 2019.
- [31] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In European Conference on Computer Vision (ECCV), 2020.
- [32] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In Advances in Neural Information Processing Systems 30 (NIPS 2017), pages 4077–4087, 2017.
- [33] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In Proceedings of the 32nd International Conference on Machine Learning (ICML) Workshops, 2015.
- [34] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1199–1208, 2018.
- [35] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In International Conference on Learning Representations (ICLR), 2017.
- [36] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Proceedings

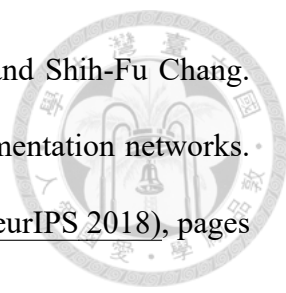


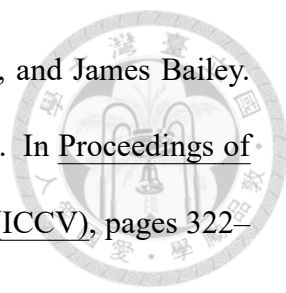
of The 33rd International Conference on Machine Learning (ICML), pages 1842–1850, 2016.

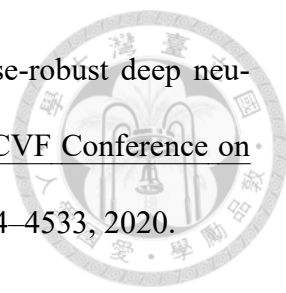


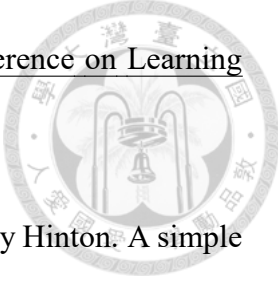
- [37] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In International Conference on Learning Representations (ICLR), 2020.
- [38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 248–255, 2009.
- [39] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. Science, 350(6266):1332–1338, 2015.
- [40] Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Ian J Goodfellow, and Jean Pouget-Abadie. Generative adversarial nets. In Advances in Neural Information Processing Systems 27 (NIPS 2014), pages 2672–2680, 2014.
- [41] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv:1411.1784, 2014.
- [42] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. Communications of the ACM, 63(11):139–144, 2020.


- 
- [43] Kai Li, Yulun Zhang, Kunpeng Li, and Yun Fu. Adversarial feature hallucination networks for few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 13470–13479, 2020.
- [44] Zitian Chen, Yanwei Fu, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. Multi-level semantic feature augmentation for one-shot learning. IEEE Transactions on Image Processing, 28(9):4594–4605, 2019.
- [45] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In Advances in Neural Information Processing Systems 31 (NeurIPS 2018), pages 2845–2855, 2018.
- [46] Mengting Chen, Yuxin Fang, Xinggong Wang, Heng Luo, Yifeng Geng, Xinyu Zhang, Chang Huang, Wenyu Liu, and Bo Wang. Diversity transfer network for few-shot learning. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 10559–10566, 2020.
- [47] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 2223–2232, 2017.
- [48] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In Advances in Neural Information Processing Systems 30 (NIPS 2017), pages 465–476, 2017.

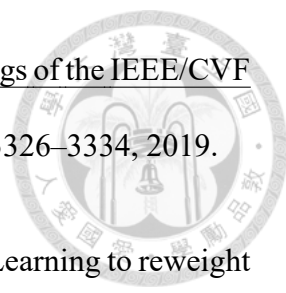
- 
- [49] Hang Gao, Zheng Shou, Alireza Zareian, Hanwang Zhang, and Shih-Fu Chang. Low-shot learning via covariance-preserving adversarial augmentation networks. In Advances in Neural Information Processing Systems 31 (NeurIPS 2018), pages 975–985, 2018.
- [50] Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8680–8689, 2019.
- [51] Satoshi Tsutsui, Yanwei Fu, and David Crandall. Meta-reinforced synthetic data for one-shot fine-grained visual recognition. In Advances in Neural Information Processing Systems 32 (NeurIPS 2019), pages 3063–3072, 2019.
- [52] Hongguang Zhang, Jing Zhang, and Piotr Koniusz. Few-shot learning via saliency-guided hallucination of samples. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2770–2779, 2019.
- [53] Benoit Frenay and Michel Verleysen. Classification in the presence of label noise: A survey. IEEE Transactions on Neural Networks and Learning Systems, 25(5):845–869, 2014.
- [54] Aritra Ghosh, Himanshu Kumar, and P.S. Sastry. Robust loss functions under label noise for deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, 2017.
- [55] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In Advances in Neural Information Processing Systems 31 (NeurIPS 2018), pages 8778–8788, 2018.

- 
- [56] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 322–330, 2019.
- [57] Yueming Lyu and Ivor W. Tsang. Curriculum loss: Robust learning and generalization against label corruption. In International Conference on Learning Representations (ICLR), 2020.
- [58] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In International Conference on Learning Representations (ICLR), 2017.
- [59] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1944–1952, 2017.
- [60] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In Advances in Neural Information Processing Systems 31 (NeurIPS 2018), pages 10456–10465, 2018.
- [61] Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? In Advances in Neural Information Processing Systems 32 (NeurIPS 2019), pages 6838–6849, 2019.

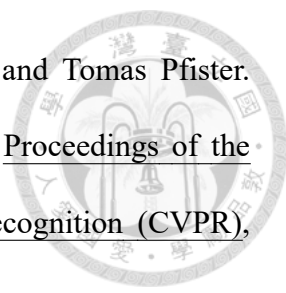
- 
- [62] Zhen Wang, Guosheng Hu, and Qinghua Hu. Training noise-robust deep neural networks via meta-learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4524–4533, 2020.
- [63] Yu Yao, Tongliang Liu, Bo Han, Mingming Gong, Jiankang Deng, Gang Niu, and Masashi Sugiyama. Dual t: Reducing estimation error for transition matrix in label-noise learning. In Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020.
- [64] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In Advances in Neural Information Processing Systems 32 (NeurIPS 2019), pages 15663–15674, 2019.
- [65] Aritra Ghosh and Andrew Lan. Contrastive learning improves model robustness under label noise. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pages 2703–2708, 2021.
- [66] Yazhou Yao, Zeren Sun, Chuanyi Zhang, Fumin Shen, Qi Wu, Jian Zhang, and Zhenmin Tang. Jo-src: A contrastive approach for combating noisy labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5192–5201, 2021.
- [67] Diego Ortego, Eric Arazo, Paul Albert, Noel E. O’Connor, and Kevin McGuinness. Multi-objective interpolation training for robustness to label noise. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 6606–6615, 2021.
- [68] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation


- 
- learning by predicting image rotations. In International Conference on Learning Representations (ICLR), 2018.
- [69] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Proceedings of the 37th International Conference on Machine Learning (ICML), pages 1597–1607, 2020.
- [70] Jean-Bastien Grill, Florian Strub, Florent Alché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In Advances in Neural Information Processing Systems 33 (NeurIPS 2020), pages 21271–21284, 2020.
- [71] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 15750–15758, 2021.
- [72] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S. Kankanhalli. Learning to learn from noisy labeled data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5051–5059, 2019.
- [73] Eran Malach and Shai Shalev-Shwartz. Decoupling ”when to update” from ”how to update”. In Advances in Neural Information Processing Systems 30 (NIPS 2017), pages 960–970, 2017.
- [74] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels.

- 
- In Proceedings of the 35th International Conference on Machine Learning (ICML), pages 2304–2313, 2018.
- [75] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In Advances in Neural Information Processing Systems 31 (NeurIPS 2018), pages 8527–8537, 2018.
- [76] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In Proceedings of the 36th International Conference on Machine Learning (ICML), pages 7164–7173, 2019.
- [77] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 13726–13735, 2020.
- [78] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In Proceedings of the 36th International Conference on Machine Learning (ICML), pages 5739–5748, 2019.
- [79] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In Proceedings of the 36th International Conference on Machine Learning (ICML), pages 1062–1070, 2019.
- [80] Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. O2u-net: A simple noisy

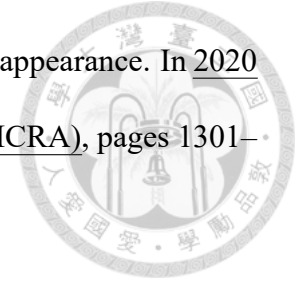
- 
- label detection approach for deep neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 3326–3334, 2019.
- [81] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In Proceedings of the 35th International Conference on Machine Learning (ICML), pages 4334–4343, 2018.
- [82] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In Advances in Neural Information Processing Systems 32 (NeurIPS 2019), pages 1919–1930, 2019.
- [83] Jiangfan Han, Ping Luo, and Xiaogang Wang. Deep self-learning from noisy labels. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 5138–5147, 2019.
- [84] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5552–5560, 2018.
- [85] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7017–7025, 2019.
- [86] Yikai Zhang, Songzhu Zheng, Pengxiang Wu, Mayank Goswami, and Chao Chen. Learning with feature-dependent label noise: A progressive approach. In International Conference on Learning Representations (ICLR), 2021.



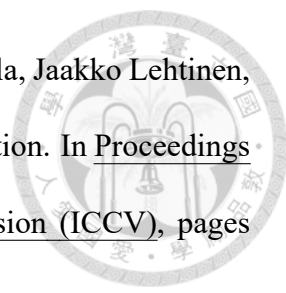
- 
- [87] Zizhao Zhang, Han Zhang, Sercan O. Arik, Honglak Lee, and Tomas Pfister. Distilling effective supervision from severe label noise. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 9294–9303, 2020.
- [88] Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In International Conference on Learning Representations (ICLR), 2020.
- [89] Kento Nishi, Yi Ding, Alex Rich, and Tobias Hollerer. Augmentation strategies for learning with noisy labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8022–8031, 2021.
- [90] Nazmul Karim, Mamshad Nayeem Rizve, Nazanin Rahnavard, Ajmal Mian, and Mubarak Shah. Unicon: Combating label noise through uniform selection and contrastive learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 9676–9686, 2022.
- [91] Haobo Wang, Ruixuan Xiao, Yiwen Dong, Lei Feng, and Junbo Zhao. Promix: Combating label noise via maximizing clean sample utility. arXiv:2007.08199v6, 2021.
- [92] Chia-Ching Lin, Yu-Chiang Frank Wang, Chin-Laung Lei, and Kuan-Ta Chen. Semantics-guided data hallucination for few-shot visual classification. In 2019 IEEE International Conference on Image Processing (ICIP), pages 3302–3306, 2019.
- [93] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly.

- 
- IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(9):2251–2265, 2019.
- [94] Frederik Pahde, Patrick Jähnichen, Tassilo Klein, and Moin Nabi. Cross-modal hallucination for few-shot fine-grained recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018.
- [95] Max Welling Diederik P Kingma. Auto-encoding variational bayes. In International Conference on Learning Representations (ICLR), 2014.
- [96] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Toronto, ON, Canada, 2009.
- [97] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1373–1378, 2015.
- [98] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(11):2579–2605, 2008.
- [99] Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. Science, 290(5500):2319–2323, 2000.
- [100] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.
- [101] Li Tang, Yue Wang, Qianhui Luo, Xiaqing Ding, and Rong Xiong. Adversarial

feature disentanglement for place recognition across changing appearance. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 1301–1307, 2020.



- [102] Dominik Lorenz, Leonard Bereska, Timo Milbich, and Bjorn Ommer. Unsupervised part-based disentangling of object shape and appearance. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10955–10964, 2019.
- [103] Bin Cheng, Tao Dai, Bin Chen, Shutao Xia, and Xiu Li. Efficient face manipulation via deep feature disentanglement and reintegration net. In ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1990–1994, 2021.
- [104] Liu Liu, Jiangtong Li, Li Niu, Ruicong Xu, and Liqing Zhang. Activity image-to-video retrieval by disentangling appearance and motion. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 2145–2153, 2021.
- [105] Chia-Ching Lin, Hsin-Li Chu, Yu-Chiang Frank Wang, and Chin-Laung Lei. Joint feature disentanglement and hallucination for few-shot image classification. IEEE Transactions on Image Processing, 30:9245–9258, 2021.
- [106] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006.
- [107] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In Proceedings of the European Conference on Computer Vision (ECCV), pages 35–51, 2018.

- 
- [108] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 10551–10560, 2019.
- [109] Nathan Hilliard, Lawrence Phillips, Scott Howland, Artëm Yankov, Courtney D. Corley, and Nathan O. Hodas. Few-shot learning with metric-agnostic conditional embeddings. arXiv:1802.04376, 2018.
- [110] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP), pages 722–729, 2008.
- [111] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of Big Data, 6(1):1–48, 2019.
- [112] Anders Krogh and John Hertz. A simple weight decay can improve generalization. In Advances in Neural Information Processing Systems 4 (NIPS 1991), pages 950–957, 1991.
- [113] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.
- [114] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML), pages 448–456, 2015.
- [115] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob

- 
- Fergus. Training convolutional networks with noisy labels. In International Conference on Learning Representations (ICLR) Workshops, 2015.
- [116] Alan Joseph Bekker and Jacob Goldberger. Training deep neural-networks based on unreliable labels. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2682–2686, 2016.
- [117] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with noisy labels revisited: A study using real-world human annotations. In International Conference on Learning Representations (ICLR), 2022.
- [118] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2691–2699, 2015.
- [119] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In European Conference on Computer Vision (ECCV), 2016.
- [120] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In European Conference on Computer Vision (ECCV), 2016.
- [121] Jiang Lu, Sheng Jin, Jian Liang, and Changshui Zhang. Robust few-shot learning for user-provided data. IEEE Transactions on Neural Networks and Learning Systems, 32(4):1433–1447, 2021.
- [122] Kevin J. Liang, Samrudhdhi B. Rangrej, Vladan Petrovic, and Tal Hassner. Few-shot learning with noisy labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 9089–9098, 2022.

- 
- [123] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. In International Conference on Learning Representations (ICLR), 2020.
- [124] Yunhui Guo, Noel C. Codella, Leonid Karlinsky, James V. Codella, John R. Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A broader study of cross-domain few-shot learning. In European Conference on Computer Vision (ECCV), 2020.
- [125] Xiaobo Xia, Tongliang Liu, Bo Han, Nannan Wang, Mingming Gong, Haifeng Liu, Gang Niu, Dacheng Tao, and Masashi Sugiyama. Part-dependent label noise: Towards instance-dependent label noise. In Advances in Neural Information Processing Systems 33 (NeurIPS 2020), pages 7597–7610, 2020.
- [126] Pengfei Chen, Junjie Ye, Guangyong Chen, Jingwei Zhao, and Pheng-Ann Heng. Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 11442–11450, 2021.
- [127] Feng Cen, Xiaoyu Zhao, Wuzhuang Li, and Guanghui Wang. Deep feature augmentation for occluded image classification. Pattern Recognition, 111:107737, 2021.
- [128] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. IEEE Transactions on Neural Networks and Learning Systems, pages 1–18, 2022.