박 사 학 위 논 문
Ph.D. Dissertation

심층 신경망을 활용한 라벨 노이즈에 강건한 학습법 연구

Robust Learning under Label Noise with Deep Neural Networks

2021

송 환 준 (宋 桓 準 Song, Hwanjun)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박 사 학 위 논 문

심층 신경망을 활용한 라벨 노이즈에 강건한 학습법 연구

2021

송 환 준

한 국 과 학 기 술 원

산업 및 시스템 공학과 (지식서비스공학대학원)

심층 신경망을 활용한 라벨 노이즈에 강건한 학습법 연구

송 환 준

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2020년 9월 3일

심사위원장	이 재 길	(인)
심 사 위 원	이 문 용	(인)
심 사 위 원	김 희 영	(인)
심 사 위 원	신 진 우 (AI 대학원)	(인)
심 사 위 원	윤 세 영 (AI 대학원)	(인)

Robust Learning under Label Noise with Deep Neural Networks

Hwanjun Song

Advisor: Jae-Gil Lee

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Industrial and Systems Engineering (Knowledge Service Engineering)

Daejeon, Korea
September 3, 2020

Approved by

Jae-Gil Lee
Professor of Industrial and Systems Engineering

The study was conducted in accordance with Code of Research Ethics*.

* Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DKSE 송환준. 심층 신경망을 활용한 라벨 노이즈에 강건한 학습법 연구. 산업 및 시스템 공학과 (지식서비스공학대학원) . 2021년. 82+v 쪽. 지도교수: 이재길. (영문 논문)
Hwanjun Song. Robust Learning under Label Noise with Deep Neural Networks. Department of Industrial and Systems Engineering (Graduate School of Knowledge Service Engineering) . 2021. 82+v pages. Advisor: Jae-Gil Lee. (Text in English)

초 록

빅 데이터의 출현으로 딥 러닝은 수많은 도메인에서 놀라운 성공을 달성하였다. 그러나 대규모 데이터에 대한 고품질 라벨 획득은 실제 시나리오에서 매우 어려우며 노이즈 라벨이있는 경우 심층 신경망의 성능은 그들에 과적합되어 일반화 성능이 크게 떨어진다. 이 과적합 문제는 드롭 아웃 및 배치 정규화과 같은 다양한 기존 정규화 기술을 사용하더라도 여전히 문제가되므로 강건한 딥러닝을 달성하는 것은 최근 기계학습 커뮤니티에서 가장 활발한 연구 주제 중 하나이다.

본 논문의 첫 번째 파트에서는 노이즈 라벨을 사용하는 지도학습에 대한 문제를 정의하고 이를 극복하기위한 최근 딥 러닝 기술의 발전에 대해 철저히 조사한다. 문헌조사는 최우수 국제 학술대회에서 발표된 논문들을 바탕으로하였다. 대부분의 연구는 (1) 손실값 조정: 손실 값을 조정하여 잘못된 라벨이 붙은 샘플의 부정적 영향을 최소화하는 방법 그리고 (2) 샘플 선택: 라벨 노이즈가있는 데이터에서 실제 라벨이 붙은 샘플을 식별하는 방법에 대한 두 가지 연구 방향에 초점을 맞추었다.

본 논문의 두 번째 파트에서는 앞서 언급 한 두 연구 방향의 장단점을 조사하고 두 방식의 장점을 모두 활용할 수 있는 하이브리드 방법 *SELFIE*를 제안한다. 이 하이브리드 접근법의 경우, 매우 높은 정확도로 손실을 조정할 수 있는 재처리 가능 샘플의 개념이 도입되었다. 재처리 가능으로 간주되는 샘플들의 경우, 그들을 손실값은 조정된 후 소 손실 기법(대표적인 샘플 선택 방법론)에 의해 선택된 샘플들의 손실과 결합된다. *SELFIE*의 우수성을 확인하기 위해 실제 또는 합성 노이즈 데이터세트를 사용하여 광범위한 실험을 수행하였고, 그 결과는 *SELFIE*가 최신 방법론들 보다 10.5 퍼센트 포인트까지 낮은테스트 오류를 달성한다는 것을 경험적으로 입증하였다.

본 논문의 세 번째 파트에서는 샘플 선택을 위해 *SELFIE* 방법론에서 사용된 소 손실 기법을 자세히 살펴본다. 현실적인 라벨 노이즈하에서 이 기법은 많은 잘못된 라벨이 붙은 샘플들을 깨끗한 샘플들로 잘못 분류하는 문제가 있었고 이를 극복하기 위해 조기 정지 기법을 기반으로 실제 라벨이 지정된 샘플을 선택하는 새로운 방법 *Prestopping*을 제안한다. *Prestopping*은 네트워크가 거짓 라벨이 붙은 샘플을 과적합하기 전에 학습 프로세스를 중지하여 초기 안전 샘플집합을 얻는다. 그 후 얻어진 안전 샘플집합을 사용하여 훈련을 재개하고 동시에 안전집합의 양과 질을 점진적으로 개선한다. *Prestopping*은 4개의 실제 또는 합성 노이즈 데이터세트에서 테스트 오류를 최대 18.1 퍼센트 포인트까지 개선한다.

*Prestopping*의 주요 기술적 과제는 학습 단계전환을 위한 최적의 중지 지점을 결정하는 것이다 (이를 최적 전환점이라고 부른다). *Prestopping*에서는 깨끗한 유효성 검사 세트 또는 실제 노이즈 비율을 최적 전환참을 찾기위한 감독으로 활용하지만 이는 실제로 획득하기가 어렵다. 본 논문의 마지막 파트에서는 이를 극복하기 위한 자가 전환 학습방법 *MORPH*를 제안한다. *MORPH*는 어떠한 감독없이 최적 전환점에서 학습 단계를 자동으로 전환한다. 5개의 벤치마크 데이터세트를 사용한 광범위한 실험은 여러 최신 방법들과 비교하여 오로지 *MORPH*만이 광범위한 노이즈 유형에서 실제 라벨이 지정된 샘플의 안전집합을 성공적으로 구성함을 보여준다. *MORPH*를 *SELFIE*의 샘플 선택기법인 소손실 기법을 대체하여 보다 높은 성능을 달성 할 수 있으나 이는 향후 연구로 남겨둔다.

핵 심 낱 말 강건한 딥 러닝, 노이즈 라벨, 라벨 노이즈, 강건한 학습, 지도 학습

Abstract

Deep learning has achieved remarkable success in numerous domains with help from large amounts of big data. However, the quality of data labels is a concern because of the lack of high-quality labels in many real-world scenarios. In the presence of noisy labels, the generalization performance of deep neural networks drastically falls down owing to their high capacity to overfit any noise labels. This overfitting issue still remains even with various conventional regularization techniques, such as dropout and batch normalization. Therefore, learning from noisy labels (robust training) has recently become one of the most active research topics in the machine learning community.

In the first part, we provide the problem statement for supervised learning with noisy labels, followed by a thorough survey on the advance in recent deep learning techniques for overcoming noisy labels; we surveyed recent studies by recursively tracking relevant bibliographies in papers published at premier research conferences. Throughout this survey, we note that the main research effort has been made to answer the two following questions: (1) how to minimize the negative influence of false-labeled samples by adjusting their loss values? and (2) how to identify true-labeled samples from noisy data?, both of which have been well-explored respectively by the two research directions, namely, *loss adjustment* and *sample selection*.

In the second part, we mainly focus on understanding the pros and cons of the aforementioned research directions and, subsequently, propose a *hybrid* learning approach called **SELFIE** that takes advantage of both loss adjustment and sample selection. For the hybrid approach, a new concept of a *refurbishable sample* is introduced to classify the sample whose loss can be correctly adjusted with high precision. The loss of refurbishable samples is adjusted first and then combined with that of the samples chosen by a representative sample selection criterion called *small-loss* trick. To validate the superiority of *SELFIE*, we conducted extensive experimentation using both real-world or synthetic noisy datasets. The results empirically verify that *SELFIE* significantly outperforms state-of-the-art methods in test error by up to 10.5 percentage point.

In the third part, we take a closer look at the *small-loss* trick adopted by *SELFIE* for sample selection. We argue that the trick misclassifies many false-labeled samples as clean samples in *realistic* noise. Hence, we present a new sample selection method called **Prestopping**, which derives a collection of true-labeled samples by using the *early stopping* mechanism. *Prestopping* obtains an initial safe set by stopping its learning process before the network begins to rapidly memorize false-labeled samples and, subsequently, resumes training to improve the quality and quantity of the set gradually. Compared with state-of-the-art methods including *SELFIE*, *Prestopping* further improves the test error by up to 18.1 percentage point on four real-world or synthetic noisy datasets.

The main technical challenge in *Prestopping* is determining the best stop point for its phase transition (we call it a best *transition point*). In *Prestopping*, a clean validation set or a known true noise rate is used for supervision, but they are usually hard to acquire in practice. In the last part, we introduce a novel self-transitional learning approach called **MORPH**, which automatically switches its learning phase at the best transition point *without* any supervision. Extensive experiments using five benchmark datasets demonstrate that only *MORPH* succeeds to construct a collection of almost true-labeled samples in a wide range of noise types. We leave the incorporation of *SELFIE* with *MORPH* as future work.

Keywords Robust Deep Learning, Noisy Label, Label Noise, Robust Learning, Supervised Learning

Contents

List of Tables

List of Figures

Chapter 1. Introduction

With the recent emergence of large-scale datasets, deep neural networks (DNNs) have recently gained significant attention in the machine learning community because they exhibit impressive performance in numerous machine learning tasks, such as computer vision [5, 6, 7], information retrieval [8, 9, 10], and language processing [11, 12, 13]. Their success is dependent on the availability of massive but carefully labeled data, which are expensive and time-consuming to obtain. Some non-expert sources, such as Amazon's Mechanical Turk and the surrounding tags of collected data, have been widely used to mitigate the high labeling cost; however, the use of these source often results in unreliable labels [14, 15, 16]. In addition, data labels can be extremely complex even for an inexperienced person [17]; they can also be adversarially manipulated by a label-flipping attack [18]. Such unreliable labels are called *noisy labels* because they may be *corrupted* from ground-truth labels. The ratio of corrupted labels in real-world datasets is reported to range from 8.0% to 38.5% [2, 19, 20, 21].

In the presence of noisy labels, training DNNs is known to be susceptible to noisy labels because of the significant number of model parameters that render DNNs overfit to even corrupted labels with the capability of learning any complex function [22]. Zhang et al. [23] demonstrated that DNNs can easily fit an entire training dataset with any ratio of corrupted labels, which eventually resulted in poor generalizability on a test dataset. Unfortunately, popular regularization techniques, such as data augmentation [24], weight decay [25], dropout [26], and batch normalization [27], do not completely overcome the overfitting issue. As shown in Figure 1.1, the gap in test accuracy between models trained on clean and noisy data remains significant even though all of the aforementioned regularization techniques are activated. Additionally, the accuracy drop with label noise is considered to be more harmful than with other noises, such as feature noise [28]. Hence, achieving a good generalization capability in the presence of noisy labels is a key challenge.

Numerous studies have been conducted to manage noisy labels in various directions[1] [29, 30]. Among them, *loss adjustment* and *sample selection* are the most active research directions being contributed by the machine learning community, where both are closely related to the following questions:

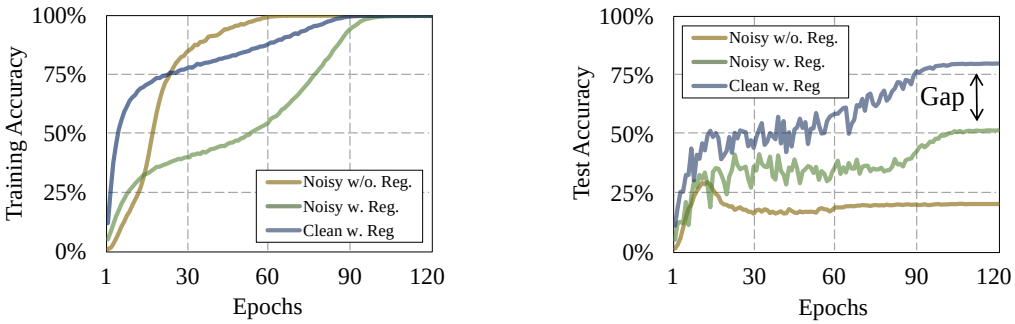
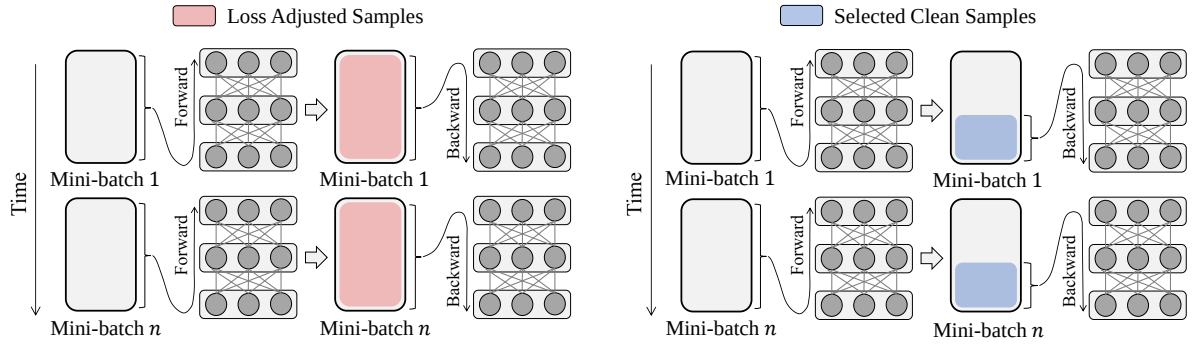


Figure 1.1: Convergence curves of training and test accuracy when training WideResNet-16-8 using a standard training method on the CIFAR-100 dataset with the symmetric noise of 40%: "Noisy w/o. Reg." and "Noisy w. Reg." are the models trained on noisy data without and with regularization, respectively, and "Clean w. Reg." is the model trained on clean data with regularization.

[1] A thorough survey on robust deep learning for overcoming noisy labels is provided in Chapter 3.



(a) Loss Adjustment. (b) Sample Selection.

Figure 1.2: Comparison of two different training procedures: (a) shows the training procedures of *loss adjustment*; (b) shows the training procedures of *sample selection*.

1. **How to minimize the negative influence of false-labeled samples by adjusting their loss values? (Figure 1.2(a))**
   The training process of a DNN is no longer noise-tolerant because of the loss computed by the noisy labels. *Loss adjustment* [31, 32, 33, 34, 35, 36, 37] is effective way for reducing the negative impact of noisy labels by adjusting the loss values of all training samples before updating the DNN. Either forward or backward losses of all training samples are adjusted and, subsequently, back-propagated to update the DNN, thereby alleviating the label noise to be accumulated.
2. **How to identify true-labeled samples from noisy data? (Figure 1.2(b))**
   Inspired by data cleaning, another effective way is filtering out false-labeled samples from the noisy data. *Sample selection* [1, 4, 38, 39, 40, 41, 42, 43, 44] identifies the samples whose labels are likely to be correct and, subsequently, uses them to update the DNN, while the rest samples are excluded from the update because they might accumulate label noise.

Here, a natural question arises: which direction is the best for robust deep learning against label noise? The answer is that **neither of these two direction is the optimal solution**. Briefly speaking, loss adjustment accumulates *severe* label noise incurred by the *false correction* because of the difficulty in adjusting the loss, especially when the number of classes or the number of false-labeled samples is large [39, 40]. On the other hand, sample selection effectively avoids such false correction by simply excluding unreliable samples, however, most of the methods in this direction ignore numerous useful samples because they are indistinguishable from false-labeled samples [34, 45].

In this thesis, our focus is developing highly robust training algorithms by overcoming the drawback of existing loss adjustment and sample selection approaches. Below, we provide an overview of the contributions of our work. Next, we outline the organization of the thesis and then summarize all the achievements during the Ph.D.

1.1 Contributions

We provide a brief summary of our contributions on each task considered in this thesis.

1.1.1 Part I: A Comprehensive Literature Survey

How can we improve the robustness of deep neural networks against noisy labels?
Which is the best research direction for overcoming noisy labels?
What are the main challenges that must be tackled for better practical use?

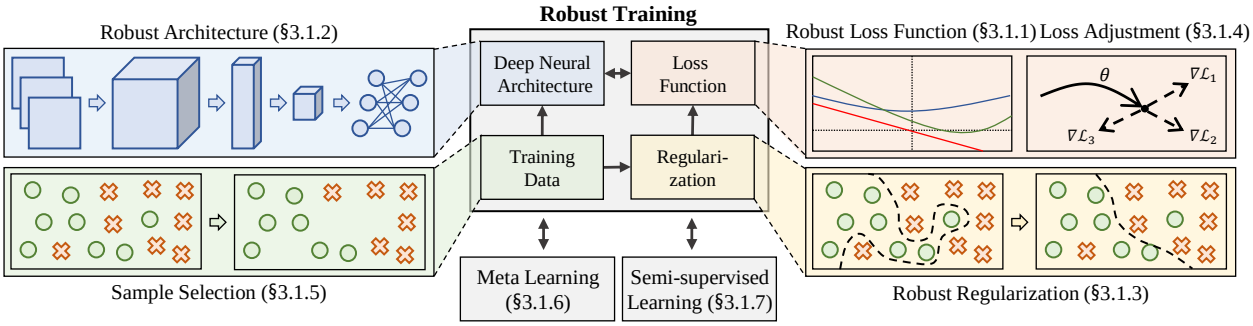


Figure 1.3: Categorization of recent deep learning methods for overcomming noisy labels.

The robustness of deep learning can be enhanced in numerous approaches [20, 26, 38, 46, 47, 48, 49]. Our survey [29] examines the advances in recent deep learning techniques for overcoming noisy labels by recursively tracking relevant bibliographies in papers published at premier research conferences, such as CVPR, ICCV, NeurIPS, ICML, and ICLR. In Chapter 3, we present a comprehensive review of 46 state-of-the-art robust training methods, followed by a systematic comparison of six properties used to evaluate their superiority. According to their methodological difference, all the methods are categorized into seven groups, as in Figure 1.3. Furthermore, several experimental guidelines are also discussed including publicly available datasets and evaluation metrics.

Contributions:

- Comprehensive Survey: This survey is the first to provide a systematic comparison of robust training methods; 46 recent robust learning methods are comprehensively investigated.
- Research Guildline: Several promising research directions is presented to deliver useful insights as a guideline for future studies.

Impact:

- This survey [29] was submitted to the *IEEE Transaction on Neural Networks and Learning Systems (TNNLS)* and is currently under review.

1.1.2 Part II: A Hybrid Learning Approach

Are there any drawbacks of using loss adjustment or sample selection methods?
Can we achieve the advantage of both loss adjustment and sample selection?

In Chapter 4, we present a hybrid learning approach, called **SELFIE** *(SELectively reFurbIsh unclEan samples)* [19], which achieves the advantages of both "loss adjustment" and "sample selection". As briefly stated in Chapter 1, loss adjustment allows for a full exploration of the training data by re-weighting all the losses; however, it suffers from the correction error when the noise is heavy. Conversely, sample selection effectively eliminates the noise accumulation by discarding all unclean samples but uses only the partial exploration

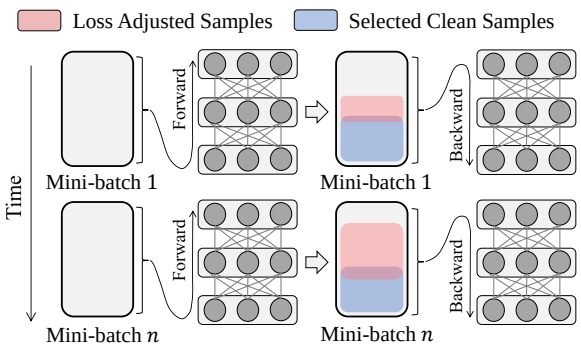


Figure 1.4: Training procedures of **SELFIE**.

of the training data. Thus, the key idea of *SELFIE* is to use *refurbishable* samples, which can be corrected with high precision, together with clean samples. As illustrated in Figure 1.4, we *selectively* correct the losses of the training samples classified as refurbishable and combine them with the losses of clean samples to propagate backward. Because the precision of the correction highly depends on the network performance, the proportion of refurbishable samples increases gradually as the training step progresses and eventually covers all samples in the training data. Overall, *SELFIE* reduces the possibility of the false correction while exploiting the full training data. We conducted extensive experiments to validate the superiority of *SELFIE* using four real-world or simulated noisy datasets. Compared with two state-of-the-art methods, *SELFIE* significantly improves the robustness to label noisy by up to 10.5*pp*[1] under any noise rate.

Contributions:

- Hybrid Approach: *SELFIE* is the first method to satisfy the two conflicting factors, namely (1) heavy noise and (2) full exploration.
- ANIMAL-10N Dataset: We build a benchmark dataset with realistic noises, which we call ANIMAL10N, and publicly release ANIMAL-10N data at https://dm.kaist.ac.kr/datasets/animal-10n

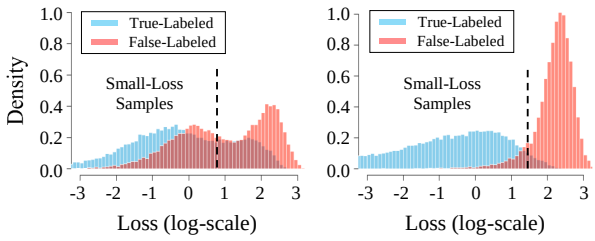
Impact:

- *SELFIE* [19] was published at the *International Conference on Machine Learning (ICML)* in 2019.
- *SELFIE* [19] was also used in production at *NAVER Corp.* to handle mislabeled data obtained from *NAVER SHOPPING* service in 2019 [LINK].

1.1.3 Part III: A Two-phase Learning Approach

Does a small-loss trick identify true-labeled samples even in realistic noise?
Can we better extract a collection of true-labeled samples than the small-loss trick?
What approach can we use to fully exploit the entire learning period without overfitting to noisy labels?

In Chapter 5, we take a closer look at a widely used *small-loss* trick, which is adopted by *SELFIE* to classify its clean samples from the noisy training samples. This trick selects a certain number of small-loss training samples as the clean ones based on the strong assumption supported by the *memorization effect* [22] that a DNN learns clean samples first and then gradually memorizes all noisy samples. However, we argue that the small-loss trick does not work well in *realistic* noise, such as asymmetric and real-world noises; many false-labeled samples are misclassified as the clean ones because the loss distributions between true-labeled and false-labeled samples are *overlapped* closely, as shown in Figure 1.5. Hence, we present a novel method called **Prestopping** [50] to derive a *maximal safe set* composed of true-labeled samples with high probability even in realistic noise. Two-phase learning is introduced: (Phase I) constructing an initial maximal safe set by stopping the DNN early and (Phase II) expanding and purifying the set during the remaining learning period. Extensive experiments using four real-world



(a) Asymmetric Noise. (b) Symmetry Noise.

Figure 1.5: Histogram of the distributions of losses at the training accuracy of 50% on a noisy CIFAR-100 with a noise rate of 40%.

[1] A *pp* is the abbreviation of a percentage point.

and simulated noisy datasets verifies that *Prestopping* significantly improves the test error by up to 18.1*pp* in a wide range of noise rates, compared with four state-of-the-art methods including *SELFIE*.

Contributions:

- Two-Phase Learning: *Prestopping* introduces a two-phase learning scheme to construct almost true-labeled samples based on the early stopping mechanism.
- Noise Type Robustness: We highlight the importance of dealing with noise type robustness for better practical use. *Prestopping* is highly robust to the noise type compared with state-of-the-art methods.

Impact:

- *Prestopping* [19] was published at the *International Conference on Machine Learning (ICML), Workshop on Uncertainty and Robustness in Deep Learning* in 2020.

1.1.4 Part IV: A Self-Transitional Learning Approach

Would it be possible to switch the learning phase of Prestopping without any supervision?
Can we identify true-labeled samples that indistinguishable from false-labeled samples?
Does the proposed method work well even for large-scale datasets with real-world label noise?

It is well known that memorization to noisy labels happens gradually, and a DNN trained only a few epochs would be mostly robust to noise. Based on this intuition, *Prestopping* early stops training before the DNN begins to rapidly memorize false-labeled samples. However, to determine the best stop point, it requires supervision from either a known true noise rate or a clean validation set, which are very hard to acquire in practice. In Chapter 6, we propose a novel self-transitional learning method called **MORPH** [51], which automatically estimates the best stop point and switch its learning phase from *seeding* to *evolution*. Briefly speaking, *MORPH* is an advanced version of *Prestopping* for sample selection; hence, each phase of *MORPH* respectively corresponds to that of *Prestopping*, but it does not require any supervision for its phase transition. After obtaining a initial safe set in Phase I, the phase is *fully automatically* transitioned into Phase II to expand the safe set gradually. Extensive experiments using five benchmark datasets demonstrate substantial improvement over state-of-the-art methods in terms of robustness (by up to 27.0*pp*) and efficiency (by up to 3.08 times).

Contributions:

- No Supervision for Transition: *MORPH* performs self-transitional learning without any supervision such as a true noise rate and a clean validation set, which are usually hard to acquire.
- Real-World Noise Robustness: *MORPH* attains high generalization performance even for two popular real-world datasets, namely WebVision-10 and FOOD-101N.
- Learning Efficiency: Differently from other methods, *MORPH* does not require any additional network or training round. Thus, it was significantly faster than other methods.

Impact:

- *MORPH* [19] was submitted at the *Advances in Neural Information Processing Systems (NeurIPS)*.

1.2 Thesis Organization

We now describe the organization of this thesis. In Chapter 2, the problem statement for supervised learning with noisy labels is provided along with the taxonomy of label noise. We also summarize the

notation frequently used throughout the thesis. In Chapter 3, we review 46 recent deep learning methods for overcoming noisy labels and compare them according to their methodological difference. In Chapter 4, we mainly focus on the two active research directions, namely loss adjustment and sample selection, and subsequently propose a hybrid approach to achieve the advantage of both directions. In Chapter 5, we revisit a small-loss trick, which is a popularly used sample selection method, and discuss its main limitation with a possible two-phase solution. In Chapter 6, we answer a longstanding issue how to automatically transition the two-phase solution without any supervision, which is a critical bottleneck to hinder the practical use of the proposed two-phase solution. In Chapter 7.1, we provide conclusions and discuss future directions. An overview of the thesis can be seen in Table 1.1.

Table 1.1: Overview of the thesis.

(§2) Preliminaries	(§2.1) Supervised Learning with Noisy Labels (§2.2) Taxonomy of Label Noise (§2.3) Non-Deep Learning Approaches
(§3) Literature Survey	(§3.1) Categorization of Robust Deep Learning Approaches (§3.2) Methodological Comparison (§3.3) Experimental Design
(§4) Hybrid Learning	(§4.1) Motivation and Overview (§4.2) Main Concept: Selective Loss Correction (§4.3) Algorithm Description (§4.4) Main Experiments (§4.5) Ablation Study
(§5) Two-Phase Learning	(§5.1) Motivation and Overview (§5.2) Main Concept: Early Stopping and Maximal Safe Set (§5.3) Algorithm Description (§5.4) Main Experiments (§5.5) Ablation Study (§5.6) A Case Study: Noisy Labels in CIFAR-100
(§6) Self-Transitional Learning	(§6.1) Motivation and Overview (§6.2) Main Concept: Seeding and Evolution (§6.3) Algorithm Description (§6.4) Main Experiments (§6.5) Ablation Study (§6.6) Noise Rate Estimation

1.3 Achievements

Our main research goals lie in improving the performance of machine learning techniques under *real-world* scenarios. Particularly, we are interested in designing more advanced approaches to handle *large-scale* and *noisy* data, which are two main real-world challenges to hinder the practical use of machine learning approaches.

Below, we summarize all the academical achievements we accomplished in chronological order:

- **Distributed Machine Learning at Scale (2017 – 2018)** [52, 53]
  [C.1] Song, H., Lee, J., and Han, W., "PAMAE: Parallel k-Medoids Clustering with High Accuracy and Efficiency," In *KDD*, 2017.
  [C.2] Song, H., and Lee, J., "RP-DBSCAN: A Superfast Parallel DBSCAN Algorithm Based on Random Partitioning," In *SIGMOD*, 2018.
- **Representation Learning (2018 – 2019)** [54, 55]
  [W.1] Park, D., Yoon, S., Song, H., and Lee, J., "MLAT: Metric Learning for kNN in Streaming Time Sereis," In *KDD, Workshop on Mining and Learning from Time Series*, 2019.
  [C.3] Park, D., Song, H., Kim, M., and Lee, J., "TRAP: Two-Level Regularized Autoencoder-based Embedding for Power-Law Distributed Data," In *The Web Conf (formerly WWW)*, 2020.
- **Training Acceleration in Deep Learning (2019 – 2020)** [56, 57]
  [J.1] Song, H., Kim, S., Kim, M., and Lee, J., "Ada-Boundary: Accelerating the DNN Training via Adaptive Boundary Batch Selection," *Machine Learning*, 2020.
  [C.4] Song, H., Kim, M., Kim, S., and Lee, J., "Carpe Diem, Seize the Samples Uncertain at the Moment for Adaptive Batch Selection," In *CIKM*, 2020.
- **Applied Data Science (2019 – Current)** [58, 59]
  [C.5] Kim, M., Song, H., Kim, D., Shin, K., and Lee, J., "PREMERE: Meta-reweighting via Self-ensembling for Point-of-Interest Recommendation," In *AAAI*, 2021.
  [C.6] Kim, S., Song, H., Kim, S., Kim, B., and Lee, J., "Revisit Prediction by Deep Survival Analysis," In *PAKDD*, 2020.
  [C.7] Kim, M., Kang, J., Kim, D., Song, H., Min, H., Nam, Y., Park, D., and Lee, J., "Hi-COVIDNet: Deep Learning Approach to Predict Inbound COVID-19 Patients and Case Study in South Korea," In *KDD (AI for COVID Track)*, 2020.
- **Uncertainty and Robustness in Deep Learning (2019 – Current)** [19, 29, 50, 51]
  [C.8] Song, H., Kim, M., and Lee, J., "SELFIE: Refurbishing Unclean Samples for Robust Deep Learning," In *ICML*, 2019.
  [W.2] Song, H., Kim, M., Park, D., and Lee, J., "How does Early Stopping Help Generalization against Label Noise?," In *ICML, Workshop on Uncertainty and Robustness in Deep Learning*, 2020.
  [J.2] Song, H., Kim, M., and Lee, J., "Learning from Noisy Labels with Deep Neural Networks: A Survey," *Transaction on Neural Networks and Learning Systems (Under Review)*, 2020.
  [C.9] Song, H., Kim, M., Park, D., and Lee, J., "MORPH: Robust Learning by Self-Transition for Handling Noisy Labels" In *ArXiv*, 2020.

This thesis focuses on the most recent topic, "Uncertainty and Robustness in Deep Learning". We believe that our research effort will significantly raise the practicality of machine learning in real-world scenarios.

Chapter 2. Preliminaries

In this section, the problem statement for supervised learning with noisy labels is provided along with the taxonomy of label noise. Managing noisy labels is a long-standing issue; therefore, we review the basic conventional approaches as well. Table 2.1 summarizes the notation frequently used in this study.

Figure 2.1: Summary of the notation.

Notation	Description
$\mathcal{X}$	the data feature space
$\mathcal{Y}$, $\tilde{\mathcal{Y}}$	the true and noisy label space
$\mathcal{D}$, $\tilde{\mathcal{D}}$	the clean and noisy training data
$P_{\mathcal{D}}$, $P_{\tilde{\mathcal{D}}}$	the joint distributions of clean and noisy data
$\mathcal{B}_t$	a set of mini-batch samples at time t
Θ_t	the parameter of a deep neural network at time t
$f(\cdot;\Theta_t)$	a deep neural network parameterized by Θ_t
$\mathcal{L}$	a specific loss function
$\mathcal{R}$	an empirical risk
$\mathbb{E}_{\mathcal{D}}$	an expectation over $\mathcal{D}$
x, x_i	a data sample of $\mathcal{X}$
y, y_i	a true label of $\mathcal{Y}$
$\tilde{y}$, $\tilde{y}_i$	a noisy label of $\tilde{\mathcal{Y}}$
η	a specific learning rate
τ	a true noise rate
b	the number of mini-batch samples in $\mathcal{B}_t$
c	the number of classes
T, $\hat{\mathrm{T}}$	the true and estimated noise transition matrix

2.1 Supervised Learning with Noisy Labels

Classification is a representative supervised learning task for learning a function that maps an input feature to a label [60]. In this paper, we consider a c-class classification problem using a DNN with a softmax output layer. Let $\mathcal{X} \subset \mathbb{R}^d$ be the feature space and $\mathcal{Y} = \{0,1\}^c$ be the ground-truth label space in a *one-hot* manner. In a typical classification problem, we are provided with a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ obtained from an unknown joint distribution $P_{\mathcal{D}}$ over $\mathcal{X} \times \mathcal{Y}$, where each (x_i, y_i) is *independent and identically distributed*. The goal of the task is to learn the mapping function $f(\cdot;\Theta) : \mathcal{X} \to [0,1]^c$ of the DNN parameterized by Θ such that the parameter Θ minimizes the empirical risk $\mathcal{R}_{\mathcal{L}}(f)$,

$$\mathcal{R}_{\mathcal{L}}(f) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}\big(f(x;\Theta), y\big)] = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}\big(f(x;\Theta), y\big), \tag{2.1}$$

where $\mathcal{L}$ is a certain loss function.

As data labels are corrupted in various real-world scenarios, we aim to train the DNN from noisy labels. Specifically, we are provided with a noisy training dataset $\tilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^N$ obtained from a noisy joint distribution $P_{\tilde{\mathcal{D}}}$ over $\mathcal{X} \times \tilde{\mathcal{Y}}$, where $\tilde{y}$ is a *noisy* label which may not be true. Hence, following the standard training procedure, a mini-batch $\mathcal{B}_t = \{(x_i, \tilde{y}_i)\}_{i=1}^b$ comprising b samples is obtained randomly from the noisy training dataset $\tilde{\mathcal{D}}$ at time t. Subsequently, the DNN parameter Θ_t at time t is updated along the descent direction of the empirical risk on mini-batch $\mathcal{B}_t$,

$$\Theta_{t+1} = \Theta_t - \eta \nabla \Big(\frac{1}{|\mathcal{B}_t|} \sum_{(x,\tilde{y})\in\mathcal{B}_t} \mathcal{L}\big(f(x;\Theta_t), \tilde{y}\big) \Big), \tag{2.2}$$

where η is a learning rate specified.

Here, the risk minimization process is no longer *noise-tolerant* because of the loss computed by the noisy labels. DNNs can easily memorize corrupted labels and correspondingly degenerate their generalizations on unseen data [17, 30, 61]. Hence, mitigating the adverse effects of noisy labels is essential to enable noise-tolerant training for deep learning.

2.2 Taxonomy of Label Noise

Even if data labels are corrupted from ground-truth labels without *any* prior assumption, in essence, the corruption probability is affected by the dependency between *data features* or *class labels*. A detailed analysis of the taxonomy of label noise was provided by Frénay and Verleysen [17].

A typical approach for modeling label noise assumes that the corruption process is conditionally *independent* of data features when the true label is given [23, 62]. That is, the true label is corrupted by a *label transition matrix* T, where $\mathrm{T}_{ij} \coloneqq p(\tilde{y} = j|y = i)$ is the probability of the true label i being flipped into a corrupted label j. In this approach, the noise is called a *symmetric* (or *uniform*) noise with a noise rate $\tau \in [0, 1]$ if $\forall_{i=j}\mathrm{T}_{ij} = 1 - \tau \wedge \forall_{i\neq j}\mathrm{T}_{ij} = \frac{\tau}{c-1}$, where a true label is flipped into other labels with equal probability. In contrast to symmetric noise, the noise is called an *asymmetric* (or *label-dependent*) noise if $\forall_{i=j}\mathrm{T}_{ij} = 1 - \tau \wedge \exists_{i\neq j, i\neq k, j\neq k}\mathrm{T}_{ij} > \mathrm{T}_{ik}$, where a true label is more likely to be mislabeled into a particular label. For example, a "dog" is more likely to be confused with a "cat" than with a "fish." In a stricter case when $\forall_{i=j}\mathrm{T}_{ij} = 1 - \tau \wedge \exists_{i\neq j}\mathrm{T}_{ij} = \tau$, the noise is called a *pair noise*, where a true label is flipped into only a certain label. However, this family of label noises is not realistic because wrong annotations are made regardless of data features.

For more realistic noise modeling, the corruption probability is assumed to be *dependent* on both the data features and class labels [20, 63]. Accordingly, the corruption probability is defined as $\rho_{ij}(x) = p(\tilde{y} = j|y = i, x)$. Unlike the aforementioned noises, because the data feature of a sample x also affects the chance of x being mislabeled, the noise is called an *instance-* and *label-dependent* noise. However, the modeling of this noise has not been investigated extensively yet owing to its complexity.

2.3 Non-Deep Learning Approaches

For decades, numerous methods have been proposed to manage noisy labels using conventional machine learning techniques. These methods can be categorized into *four* groups [17, 61], as follows:

- **Data Cleaning:** Training data are cleaned by excluding samples whose labels are likely to be corrupted. Bagging and boosting are used to filter out false-labeled samples to remove samples with higher weights because false-labeled samples tend to exhibit much higher weights than true-labeled samples [64, 65]. In addition, various methods, such as k-nearest neighbor, outlier detection, and anomaly detection, have been widely exploited to exclude false-labeled samples from noisy training data [66, 67, 68]. Nevertheless, this family of methods suffers from over-cleaning issue that overly removes even the true-labeled samples.

- **Surrogate Loss:** Motivated by the noise-tolerance of the 0-1 loss function [62], many researchers have attempted to resolve its inherent limitations, such as computational hardness and non-convexity that render gradient methods unusable. Hence, several convex surrogate loss functions, which approximate the 0-1 loss function, have been proposed to train a specified classifier under the binary classification setting [69, 70, 71, 72, 73]. However, these loss functions cannot support the multi-class classification task.

- **Probabilistic Method:** Under the assumption that the distribution of features is helpful in solving the problem of learning from noisy labels [74], the confidence of each label is estimated by clustering and then used for a weighted training scheme [75]. This confidence is also used to convert hard labels into soft labels to reflect the uncertainty of labels [76]. In addition to these clustering approaches, several Bayesian methods have been proposed for graphical models such that they can benefit from using any type of prior information in the learning process [77]. However, this family of methods may exacerbate the overfitting issue owing to the increased number of model parameters.

- **Model-based Method:** As conventional models, such as the SVM and decision tree, are not robust to noisy labels, significant effort has been expended to improve the robustness of these models. To develop a robust SVM model, misclassified samples during learning are penalized in the objective [78, 79]. In addition, several decision tree models are extended using new split criteria to solve the overfitting issue when the training data are not fully reliable [80, 81]. However, it is infeasible to apply the design principles in these models to deep learning.

Chapter 3. A Comprehensive Literature Survey

Summary: Chapter based on work being under review at TNNLS 2020 [29]

Deep learning has achieved remarkable success in numerous domains with help from large amounts of big data. However, the quality of data labels is a concern because of the lack of high-quality labels in many real-world scenarios. As noisy labels severely degrade the generalization performance of deep neural networks, learning from noisy labels (robust training) is becoming an important task in modern deep learning applications. In this chapter, we provide a comprehensive review of 46 state-of-the-art robust training methods, all of which are categorized into seven groups according to their methodological difference, followed by a systematic comparison of six properties used to evaluate their superiority. Subsequently, we summarize the typically used evaluation methodology, including public noisy datasets and evaluation metrics. Finally, we present several promising research directions that can serve as a guideline for future studies.

3.1 Categorization of Robust Deep Learning Approaches

According to our comprehensive survey, the robustness of deep learning can be enhanced in numerous approaches [20, 26, 38, 46, 47, 48, 49]. Figure 3.1 shows an overview of recent research directions conducted by the machine learning community. Most of them (i.e., §3.1.1–§3.1.5) focused on making a supervised learning process more robust to label noise. *Robust loss function* and *loss adjustment* aim to modify the loss function or its value; *robust architecture* aims to change an architecture to model a noise transition matrix of a noisy dataset; *robust regularization* aims to enforce a DNN to overfit less to false-labeled samples; *sample selection* aims to identify true-labeled samples from noisy training data. Beyond supervised learning, researchers have recently attempted to further improve noise robustness by adopting *meta learning* (§3.1.6) and *semi-supervised learning* (§3.1.7). In this section, we categorize all recent deep learning methods into *seven* groups corresponding to popular research directions, as shown in Figure 3.1.

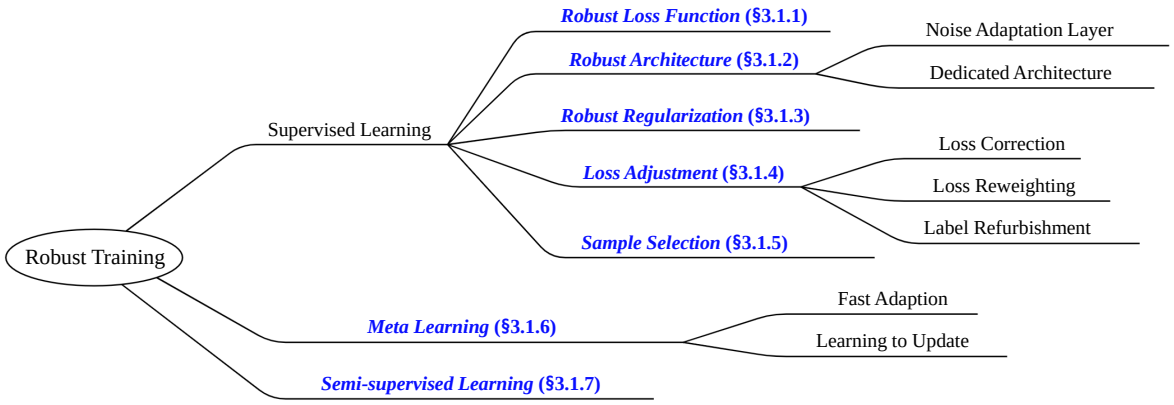


Figure 3.1: A high level research overview of robust deep learning for noisy labels. The research directions that are actively contributed by the machine learning community are categorized into seven groups in blue italic.

Table 3.1: Summary of existing deep learning methods according to the seven categories in Figure 1.3.

Category	Robust Deep Learning Method
Robust Loss Function	*Robust MAE* [46], *Generalized Cross Entropy* [82], *Symmetric Cross Entropy* [83], *Curriculum Loss* [44]
Robust Architecture	*Webly Learning* [84], *Noise Model* [85], *Dropout Noise Model* [86], *S–model* [87], *C–model* [87], *NLNN* [88], *Probabilistic Noise Model* [20], *Masking*[89], *Contrastive-Additive Network*[90]
Robust Regularization	*Adversarial Training* [91], *Label Smoothing* [92], *Mixup* [93], *Bilevel Learning* [94], *Annotator Confusion* [95], *Pre-training* [96]
Loss Adjustment	*Backward Correction* [31], *Forward Correction* [31], *Gold Loss Correction* [32], *Importance Reweighting* [33], *Active Bias* [34], *Bootstrapping* [35], *Dynamic Bootstrapping* [36], *D2L* [37]
Sample Selection	*Decouple* [38], *MentorNet* [39], *Co-teaching* [40], *Co-teaching+* [4], *Iterative Detection* [41], *ITLM* [42], *INCV* [1], *SELF* [43], *Curriculum Loss* [44]
Meta Learning	*Meta-Regressor* [48], *Knowledge Distillation* [97], *L2LWS* [98], *CWS* [99], *Automatic Reweighting* [100], *MLNT* [101], *Meta-Weight-Net* [102], *Data Coefficients* [103]
Semi-supervised Learning	*Label Aggregation* [49], *Two-Stage Framework* [104], *SELF* [43], *DivideMix* [105]

Table 3.1 summarizes 46 existing deep learning methods according to them. Some methods may belong to more than one categories if they combine multiple approaches. Below, we provide a detailed review for 46 state-of-the-art robust deep learning methods categorized into the seven groups.

3.1.1 Robust Loss Function

Considering the robustness of risk minimization schemes on the loss function, researchers have attempted to design robust loss functions [44, 46, 82, 83]. The goal is to provide a loss function that achieves a small risk for unseen clean data even when noisy labels exist in the training data.

Initially, Manwani and Sastry [70] theoretically proved a sufficient condition for the loss function such that risk minimization with that function becomes noise-tolerant for binary classification. Subsequently, the sufficient condition was extended for multi-class classification using deep learning [46]. Specifically, a loss function is defined to be *noise-tolerant* for a c-class classification under *symmetric* noise if the function satisfies the noise rate $\tau < \frac{c-1}{c}$ and

$$\sum_{j=1}^{c} \mathcal{L}\big(f(x;\Theta), y=j\big) = C, \ \forall x \in \mathcal{X}, \ \forall f, \tag{3.1}$$

where C is a constant. This condition guarantees that the classifier trained on noisy data has the same misclassification probability as that trained on noise-free data under the specified assumption. Moreover, if $\mathcal{R}_{\mathcal{L}}(f^*) = 0$, then the function is also noise-tolerant under an *asymmetric* noise, where f^* is a global risk minimizer of $\mathcal{R}_{\mathcal{L}}$.

For the classification task, the categorical cross entropy (CCE) loss is the most widely used loss function owing to its fast convergence and high generalization capability. However, in the presence of noisy

labels, the *robust MAE* [46] showed that the mean absolute error (MAE) loss achieves better generalization than the CCE loss because only the MAE loss satisfies the aforementioned condition. A limitation of the MAE loss is that its generalization performance degrades significantly when complicated data are involved. Hence, the *generalized cross entropy* (GCE) [82] was proposed to achieve the advantages of both MAE and CCE losses; the GCE loss is a more general class of noise-robust loss that encompasses both of them. Inspired by the symmetricity of the Kullback-Leibler divergence, the symmetric cross entropy (SCE) [83] was proposed by combining a noise tolerance term, namely reverse cross entropy loss, with the standard CCE loss. Meanwhile, the *curriculum loss* (CL) [44] is a surrogate loss of the 0-1 loss function; it provides a tight upper bound and can easily be extended to multi-class classification.

Nevertheless, it has been reported that performances with such losses are significantly affected by noisy labels [100]. Such implementations perform well only in simple cases, when learning is easy or the number of classes is small. Moreover, the modification of the loss function increases the training time for convergence [82].

3.1.2 Robust Architecture

In numerous studies, architectural changes have been made to model the label transition matrix of a noisy dataset. These changes include (1) adding a noise adaptation layer at the top of the softmax layer [84, 85, 87, 88] or (2) designing a new dedicated architecture [20, 89, 90]. The resulting architectures yielded improved generalization through the modification of the DNN output based on the estimated label transition probability.

Noise Adaptation Layer [84, 85, 87, 88]

The noise adaptation layer is intended to mimic the noise behavior in learning a DNN. Let $p(y|x;\Theta)$ be the output of the base DNN with a softmax output layer. Subsequently, the probability of a sample x being predicted as its annotated noisy label $\tilde{y}$ is parameterized by

$$\begin{aligned} p(\tilde{y}|x;\Theta,\mathcal{W}) &= \sum_{i=1}^{c} p(\tilde{y}, y=i|x;\Theta,\mathcal{W}) \\ &= \sum_{i=1}^{c} \underbrace{p(\tilde{y}|y=i;\mathcal{W})}_{\text{noise adaptation layer}} \underbrace{p(y=i|x;\Theta)}_{\text{base model}}. \end{aligned} \tag{3.2}$$

Here, the noisy label $\tilde{y}$ is assumed to be conditionally independent of the input feature x in general. Accordingly, as shown in Figure 3.2, the noisy adaptation layer is added at the top of the base DNN to model the label transition matrix parameterized by $\mathcal{W}$. This layer should be removed when a test dataset is to be predicted.

Webly learning [84] first trains the base DNN only for easy samples retrieved by search engines; subsequently, the confusion matrix for all training samples is used as the initial weight $\mathcal{W}$ of the noise adaptation layer. It fine-tunes the entire model in an end-to-end manner for hard training samples. Meanwhile, the *noise model* [85] initializes $\mathcal{W}$ to an identity matrix and adds a regularizer to force $\mathcal{W}$ to diffuse during DNN training. The *dropout*

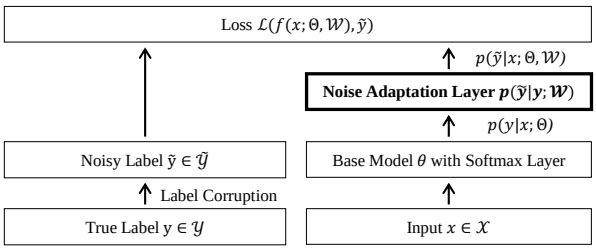


Figure 3.2: Noise modeling process using the noise adaptation layer.

noise model [26] applies dropout regularization to the adaptation layer, whose output is normalized by the softmax function to implicitly diffuse $\mathcal{W}$. The *s-model* [87] is similar to the *dropout noise model* but dropout is not applied. The *c-model* [87] is an extension of the s-model that models the instance- and label-dependent noise, which is more realistic than the symmetric and asymmetric noises. Meanwhile, *NLNN* [88] adopts the EM algorithm to iterate the E-step to estimate the label transition matrix and the M-step to back-propagate the DNN.

The drawback of this family is the strong assumption regarding the noise type, which hinders a model's generalization to complex label noise [20]. Meanwhile, for the EM-based method, becoming stuck in local optima is inevitable, and high computational costs are incurred [87].

Dedicated Architecture [20, 89, 90]

To overcome the aforementioned drawbacks of the noise adaptation layer, several studies have been conducted, where specific architectures have been designed. They typically aimed at increasing the reliability of estimating the label transition probability to handle more complex and realistic noise.

Probabilistic noise modeling [20] manages two independent networks, each of which is specialized to predict the noise type and label transition probability. Because an EM-based approach with random initialization is impractical for training the entire network, both networks are trained with massive noisy labeled data after the pre-training step with a small amount of clean data. Meanwhile, *masking* [89] is a human-assisted approach to convey the human cognition of invalid label transitions. Considering that noisy labels are mainly from the interaction between humans and tasks, the invalid transition investigated by humans was leveraged to constrain the noise modeling process. Owing to the difficulty in specifying the explicit constraint, a variant of generative adversarial networks (GANs) [106] was employed in this study. Most recently, the *contrastive-additive noise network* [90] was proposed to adjust incorrectly estimated label transition probabilities by introducing a new concept of quality embedding, which models the trustworthiness of noisy labels.

Compared with the noise adaptation layer, this family of methods significantly improves the robustness to more diverse types of label noise, but it cannot be easily extended to other architectures.

3.1.3 Robust Regularization

Regularization methods have been widely studied to improve the generalizability of a learned model in the machine learning community [24, 25, 26, 27]. By avoiding overfitting in model training, the robustness to label noise improves with widely-used regularization techniques such as *data augmentation* [24], *weight decay* [25], *dropout* [26], and *batch normalization* [27]. Additionally, *adversarial training* [91] enhances the noise tolerance by encouraging the DNN to correctly classify both original inputs and hostilely perturbed ones. *Label smoothing* [92] estimates the marginalized effect of label noise during training, thereby reducing overfitting by preventing the DNN from assigning a full probability to noisy training samples. These methods operate well on moderately noisy data. However, they are generic regularization techniques that are not specialized in handling label noise; hence, poor generalization could be obtained when the noise is heavy [95].

Recently, as learning from noisy labels has become a key challenge, more advanced regularization techniques have been proposed, which further improved robustness to label noise. *Mixup* [93] regularizes the DNN to favor simple linear behaviors in between training samples. First, the mini-batch is constructed using virtual training samples, each of which is formed by the linear interpolation of two noisy

training samples $(x_i, \tilde{y}_i)$ and $(x_j, \tilde{y}_j)$ obtained at random from noisy training data $\tilde{\mathcal{D}}$,

$$x_{mix} = \lambda x_i + (1-\lambda)x_j \quad \text{and} \quad y_{mix} = \lambda \tilde{y}_i + (1-\lambda)\tilde{y}_j, \tag{3.3}$$

where $\lambda \sim Beta(\alpha, \alpha)$ and $\alpha \in [0, \infty]$. *Mixup* extends the training distribution by updating the DNN for the constructed mini-batch.

Bilevel learning [94] uses a clean validation dataset to regularize the overfitting of a model by introducing a bilevel optimization approach, which differs from the conventional one in that its regularization constraint is also an optimization problem. Overfitting is controlled by adjusting the weights on each mini-batch and selecting their values such that they minimize the error on the validation dataset. Meanwhile, *annotator confusion* [95] assumes the existence of multiple annotators and introduces a regularized EM-based approach to model the label transition probability; its regularizer enables the estimated transition probability to converge to the true confusion matrix of the annotators. In addition, *pre-training* [96] empirically proves that fine-tuning on a pre-trained model provides a significant improvement in robustness compared with models trained from scratch. The universal representations of pre-training prevent the model parameters from being updated in the wrong direction by noisy labels.

The main advantage of this family of methods is its *flexibility* in collaborating with other directions because it only requires simple modifications during training. However, the performance improvement is relatively insignificant, and it tends to yield extra hyperparameters sensitive to both noise and data types.

3.1.4 Loss Adjustment

Loss adjustment is effective for reducing the negative impact of noisy labels by adjusting the loss of all training samples before updating the DNN. The methods associated with it can be categorized into three groups depending on their adjustment philosophy: (1) *loss correction* that estimates the label transition matrix to correct the forward or backward loss [31, 31, 32], (2) *loss reweighting* that imposes different importance to each sample for a weighted training scheme [33, 34], and (3) *label refurbishment* that adjusts the loss using the refurbished label obtained from a convex combination of noisy and predicted labels [35, 36, 37].

Generally, the family of methods in this category allows for a *full exploration* of the training data while adjusting the loss of every sample. However, the error incurred by *false* correction is accumulated, especially when the number of classes or the number of mislabeled samples is large [40, 50].

Loss Correction [31, 31, 32]

Similar to the noise adaptation layer presented in Section 3.1.2, this approach modifies the loss of each sample by multiplying the estimated label transition probability by the output of a specified DNN. The main difference is that the learning of the transition probability is decoupled from that of the model.

Backward correction [31] initially approximates the label transition matrix using the softmax output of the DNN trained without loss correction. Subsequently, it retrains the DNN while correcting the original loss based on the estimated matrix. The corrected loss of a sample $(x, \tilde{y})$ is computed by a linear combination of its loss values for observable labels, whose coefficient is the transition probability from each observable label $j \in \{1, \ldots, c\}$ to its target label $\tilde{y}$. Therefore, the backward correction $\overleftarrow{\mathcal{L}}$ is performed by multiplying the estimated transition probability with its corresponding loss value,

$$\overleftarrow{\mathcal{L}}\big(f(x;\Theta), \tilde{y}\big) = \sum_{j=1}^{c} \hat{p}(\tilde{y}|y{=}j)\mathcal{L}\big(f(x;\Theta), y{=}j\big) = \hat{\mathrm{T}}_{\cdot j}^{-1}\Big(\mathcal{L}\big(f(x;\Theta), y{=}1\big), \ldots, \mathcal{L}\big(f(x;\Theta), y{=}c\big)\Big)^{\top}, \tag{3.4}$$

where $\hat{\mathrm{T}}$ is the estimated label transition matrix.

Conversely, *forward correction* [31] uses a linear combination of a DNN's softmax outputs before applying the loss function. Hence, the forward correction $\vec{\mathcal{L}}$ is performed by multiplying the estimated transition probability with the softmax outputs during the forward propagation step,

$$\vec{\mathcal{L}}\big(f(x;\Theta),\tilde{y}\big) = \mathcal{L}\Big(\big(\hat{p}(\tilde{y}|y=1),\ldots,\hat{p}(\tilde{y}|y=c)\big)f(x;\Theta)^{\top},\tilde{y}\Big) = \mathcal{L}\big(\hat{\mathrm{T}}^{-1}f(x;\Theta)^{\top},\tilde{y}\big). \tag{3.5}$$

Furthermore, *gold loss correction* [32] was proposed to leverage available trusted labels for loss correction. To obtain a more accurate transition matrix, the confusion matrix for the trusted labels is utilized as additional information. Owing to the trusted labels, the noise robustness of the forward or backward correction method is further improved.

Loss Reweighting [33, 34]

Inspired by the concept of importance reweighting [107], loss reweighting aims to assign smaller weights to the samples with false labels and greater weights to those with true labels. Accordingly, the reweighted loss on the mini-batch $\mathcal{B}_t$ is used to update the DNN,

$$\Theta_{t+1} = \Theta_t - \eta\nabla\Big(\frac{1}{|\mathcal{B}_t|}\sum_{(x,\tilde{y})\in\mathcal{B}_t}\overbrace{w(x,\tilde{y})\mathcal{L}\big(f(x;\Theta_t),\tilde{y}\big)}^{\text{reweighted loss}}\Big), \tag{3.6}$$

where $w(x,\tilde{y})$ is the weight of a sample x with its noisy label $\tilde{y}$. Hence, the samples with smaller weights do not significantly affect the DNN learning.

In *importance reweighting* [33], the ratio of two joint data distributions $w(x,y) = P_{\mathcal{D}}(x,\tilde{y})/P_{\tilde{\mathcal{D}}}(x,\tilde{y})$ determines the contribution of the loss of each noisy sample. An approximate solution to estimate the ratio was developed because the two distributions are difficult to determine from noisy data. Meanwhile, *active bias* [34] emphasizes uncertain samples with inconsistent label predictions by assigning their prediction variances as the weights for training.

Label Refurbishment [35, 36, 37]

Refurbishing a noisy label $\tilde{y}$ effectively prevents overfitting to false labels. Let $\hat{y}$ be the current prediction of DNN $f(x;\Theta)$. Therefore, the refurbished label y^{refurb} can be obtained by a convex combination of the noisy label $\tilde{y}$ and the DNN prediction $\hat{y}$,

$$y^{refurb} = \alpha\tilde{y} + (1-\alpha)\hat{y}, \tag{3.7}$$

where $\alpha \in [0,1]$ is the label confidence of $\tilde{y}$. To mitigate the damage of incorrect labeling, this approach backpropagates the loss for the refurbished label instead of the noisy one, thereby yielding substantial robustness to noisy labels.

Bootstrapping [35] is the first method that proposes the concept of label refurbishment to update the target label of training samples. It develops a more coherent network that improves its ability to evaluate the consistency of noisy labels, with the label confidence α obtained via cross-validation. *Dynamic bootstrapping* [36] dynamically adjusts the confidence α of individual training samples. For each training epoch, it estimates the probability of a sample x being true-labeled by fitting a two-component and one-dimensional beta mixture model to the loss distribution of all training samples and then uses it as the confidence α. As true-labeled samples exhibit smaller losses than false-labeled ones,

the confidence α of a sample x is obtained through the posterior probability of the mixture model, where g is the beta component with a smaller mean.

D2L [37] trains a DNN using a dimensionality-driven learning strategy to avoid overfitting to false labels. A simple measure called *local intrinsic dimensionality* [108] is adopted to evaluate the confidence α in considering that the overfitting is exacerbated by dimensional expansion. Hence, refurbished labels are generated to prevent the dimensionality of the representation subspace from expanding at a later stage of training. Most recently, *SELFIE* [19] introduces a novel concept of *refurbishable samples* that can be corrected with high precision. The key idea is to consider the sample with consistent label predictions as refurbishable because such consistent predictions correspond to its true label with a high probability owing to the learner's perceptual consistency. Accordingly, the labels of only refurbishable samples are corrected to minimize the number of falsely corrected cases.

3.1.5 Sample Selection

To avoid any false corrections, many recent studies have adopted sample selection that involves selecting true-labeled samples from a noisy training dataset [38, 39, 40, 4, 42, 1, 41, 43, 44]). In this case, the update equation in Eq. (2.2) is modified to render a DNN more robust for noisy labels. Let $\mathcal{C}_t \subseteq \mathcal{B}_t$ be the selected *clean* samples at time t. Therefore, the DNN is updated only for the selected clean samples $\mathcal{C}_t$,

$$\Theta_{t+1} = \Theta_t - \eta \nabla \Big(\frac{1}{|\mathcal{C}_t|} \sum_{(x,\tilde{y}) \in \mathcal{C}_t} \mathcal{L}\big(f(x;\Theta_t), \tilde{y}\big) \Big). \tag{3.8}$$

Here, the remaining mini-batch samples, which are likely to be false-labeled, are excluded from the update to pursue robust learning. The key challenge is to design the *selection criteria* for sample selection.

Initially, *decouple* [38] proposes the decoupling of when to update from how to update. It updates the model for samples selected based on a disagreement between the two classifiers. Hence, two DNNs are maintained simultaneously and updated only using samples with different label predictions from these two DNNs. Subsequently, many researchers have adopted another selection criterion, called a *small-loss* trick, which treats a certain number of small-loss training samples as true-labeled samples. In particular, the small-loss trick successfully separates true-labeled samples from false-labeled samples because many true-labeled samples tend to exhibit smaller losses than false-labeled samples. This phenomenon is well justified by the *memorization effect* [22], i.e., DNNs are prone to learn clean samples first and then gradually learn other noisy samples.

MentorNet [39] introduces a collaborative learning paradigm in which a pre-trained mentor network guides the training of a student network. Based on the small-loss trick, the mentor network provides the student network with samples whose labels are liekly to be correct. *Co-teaching* [40] and *Co-teaching+* [4] also maintain two DNNs, but each DNN selects a certain number of small-loss samples and feeds them to its peer DNN for further training. Compared with *Co-teaching*, *Co-teaching+* further employs the disagreement strategy of *decouple*. *ITLM* [42] iteratively minimizes the trimmed loss by alternating between selecting true-labeled samples at the current moment and retraining the DNN using them. At each training round, only a fraction of small-loss samples obtained in the current round are used to retrain the DNN in the next round. *INCV* [1] randomly divides noisy training data and then employs cross-validation to classify true-labeled samples while removing large-loss samples at each training round. Here, *Co-teaching* is adopted to train the DNN on the identified samples in the final round of training. Other than the small-loss trick, *iterative detection* [41] detects false-labeled samples by employing the local outlier factor algorithm [109]. Furthermore, it uses similar or dissimilar sample pairs to learn deep

discriminative features; then, it gradually pulls away false-labeled samples from true-labeled samples in the deep feature space.

Recently, researchers have attempted to combine sample selection with other approaches. *SELFIE* [19] is a hybrid approach of sample selection and loss correction. The loss of refurbishable samples is corrected (i.e., loss correction) and then used together with that of small-loss samples (i.e., sample selection). Consequently, more training samples are considered for updating the DNN. Meanwhile, the *curriculum loss (CL)* [44] is combined with the robust loss function approach and used to extract the true-labeled samples from a noisy dataset based on a manually specified selection threshold. *SELF* [43] is combined with a semi-supervised learning approach to progressively filter out false-labeled samples. It exploits the *mean-teacher* model [110] to obtain a more stable supervisory signal than the noisy model snapshot.

This family of methods effectively avoids the risk of false correction by simply excluding unreliable samples. However, they may eliminate numerous useful samples, and their general philosophy of selecting small-loss samples is applicable to only some limited cases such as symmetric noise [19]. In addition, either the true noise rate or a clean validation dataset must be available to quantify the number of samples that should be selected as true-labeled ones by the model [40, 42].

3.1.6 Meta Learning

In recent years, meta learning has become an important topic in the machine learning community. The key concept is *learning to learn*, which performs learning at a level higher than conventional learning. Generally, meta learning is applied to improve noise robustness based on two approaches: (1) *fast adaption* that trains a model applicable to various learning tasks without overfitting to false labels [48, 101] and (2) *learning to update* that learns the loss adjustment rule to reduce the negative effects of the noisy labels [97, 98, 99, 100, 102, 103].

Fast Adaption [48, 101]

The *meta-regressor* [48] proposes to build a regressor that estimates the performances of robust training methods in a new noise learning task. It generates a meta-dataset composed of meta-features and meta-labels, where a meta-feature is the descriptor of a specified dataset, such as the number of classes and the number of features, and a meta-label is the F1-score of a specified robust training method obtained by learning the model for a synthetically corrupted dataset. Using the regressor trained on the meta-dataset, the meta-regressor recommends the most promising training method when applied to a newly given dataset. Meanwhile, *MLNT* [101] aims to obtain model parameters that can be easily fine-tuned or transferred to different label noises. To learn such model parameters, it generates multiple mini-batches with synthetically corrupted labels and then uses them to update the DNN such that the difference between the predictions obtained from the original and corrupted mini-batches is minimized.

These meta learning methods are general and model-agnostic. However, their drawback is the scalability caused by multiple inferences or updates prior to a conventional update for guidance.

Learning to Update [97, 98, 99, 100, 102, 103]

This approach is similar to loss adjustment, but the adjustment is automated in a meta-learning manner. *Knowledge distillation* [97] adopts the technique of transferring knowledge from one expert model to a target model. For the *label refurbishment* in Eq. (3.7), the prediction from the expert DNN trained on a small clean validation dataset is used instead of the prediction $\hat{y}$ from the target DNN.

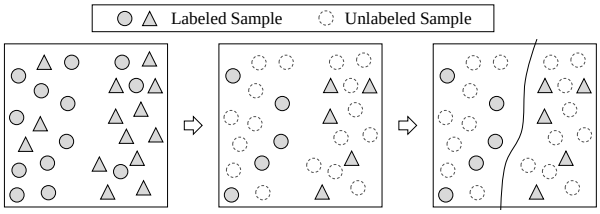
In addition, it leverages a knowledge graph, which encodes the structure of the label space, to further elaborate the expert DNN's prediction.

Meanwhile, other methods aim at learning the weight function $w(x, \tilde{y})$ for *loss reweighting* in Eq. (3.6). Specifically, *L2LWS* [98] and *CWS* [99] propose a unified neural architecture composed of a target DNN and a meta-DNN. The meta-DNN is trained on a small clean validation dataset; it then provides guidance to evaluate the weight score for the target DNN. Here, part of the two DNNs are shared and jointly trained to benefit from each other. *Automatic reweighting* [100] proposes a meta learning algorithm that learns the weights of training samples based on their gradient directions. It includes a small clean validation dataset into the training dataset and reweights the backward loss of the mini-batch samples such that the updated gradient minimizes the loss of this validation dataset. *Meta-Weight-Net* [102] parameterizes the weighting function as a multi-layer perceptron network with only one hidden layer. A meta-objective is defined to update its parameters such that they minimize the empirical risk of a small clean dataset. At each iteration, the parameter of the target network is guided by the weight function updated via the meta-objective. Likewise, *Data Coefficient* [103] proposes a meta re-weighting framework that optimizes the weights of samples by leveraging a small clean set, which is only 0.2% of the total training set, while refurbishing the label of samples probably mislabeled.

By learning the update rule via meta-learning, the trained network easily adapts to various types of label noise. Nevertheless, an unbiased clean validation dataset is essential to minimize the auxiliary objective for meta-learning, although it may not be available in real-world scenarios.

3.1.7 Semi-supervised Learning

Wemi-supervised learning is another method that reduces the annotation cost in the presence of both labeled and unlabeled data [111, 112]. To overcome noisy labels, several recent studies transformed the problem of learning from noisy labels into a semi-supervised learning task (e.g., *Label aggregation* [49], *two-stage framework* [104], *temporal ensembling* [113], *SELF* [43], *DivideMix* [105], and *MixMatch* [114]). In general, as shown in Figure 3.3, possibly false-labeled samples in noisy data are treated as *unlabeled*, whereas the remaining samples are treated as *labeled*. Subsequently, semi-supervised learning is performed using the transformed data.



(a) Noisy Data. (b) Transformed Data. (c) SSL.

Figure 3.3: Procedures for semi-supervised learning under label noise.

Label aggregation [49] adopts a simple label aggregation strategy used in semi-supervised learning. Several low-cost weak networks are trained on a clean validation dataset as multiple annotators. Subsequently, the true label of a noisy sample is obtained through a weighted label aggregation (i.e., ensemble) of the annotators' predictions. This method skips the process of extracting clean-labeled samples in Figure 3.3(b) by assuming the existence of the clean validation dataset. By contrast, the *two-stage framework* [104] pre-trains a DNN on noisy training data and then manages the labeled set by only maintaining samples for which the DNN's prediction probability to the annotated label is higher than a certain threshold, while the rest samples are included in the unlabeled set. Next, the transformed data are used to train another network using a semi-supervised method called *temporal ensembling* [113]. Most recently, *SELF* [43] adopted the concept of self-ensemble in semi-supervised learning to produce a more supervisory signal to filter out false-labeled samples during training. By maintaining the run-

ning average model as the backbone, it obtains the self-ensemble predictions of all training samples and then progressively removes samples whose ensemble predictions do not agree with their annotated labels. This method further leverages unsupervised loss from the samples not included in the selected set. Meanwhile, *DivideMix* [105] is an extension of the semi-supervised data augmentation technique called *MixMatch* [114]. A two-component and one-dimensional Gaussian mixture model is fitted to the training loss to obtain the confidence of an annotated label. By setting a confidence threshold, the training data is categorized into a labeled set and an unlabeled set. Subsequently, *MixMatch* is employed to train a DNN for the transformed data.

Noise robustness is significantly improved using several semi-supervised techniques. However, the hyperparameters introduced by these techniques render a DNN more susceptible to changes in data and noise types, and an increase in computational cost is inevitable.

3.2 Methodological Comparison

In this section, we compare the 46 deep learning methods for overcoming noisy labels introduced in Section 3.1 with respect to the following *six* properties. When selecting the properties, we refer to the properties that are typically used to compare the performance of robust deep learning methods [40, 19]. To the best of our knowledge, this survey is the first to provide a systematic comparison of robust training methods. This comprehensive comparison will provide useful insights for future studies.

- **(P1) Flexibility:** With the rapid evolution of deep learning research, a number of new network architectures are constantly emerging and becoming available. Hence, the ability to support any type of architecture is important. "Flexibility" ensures that the proposed method can quickly adapt to the state-of-the-art architecture.

- **(P2) No Pre-traing:** A typical approach to improve noise robustness is to use a pre-trained network; however, this incurs additional computational cost to the learning process. "No Pre-training" ensures that the proposed method can be trained from scratch without any pre-training.

- **(P3) Full Exploration:** Excluding unreliable samples from the update is an effective method for robust deep learning; however, it eliminates hard but useful training samples as well. "Full Exploration" ensures that the proposed methods can use *all* training samples without severe overfitting to false-labeled samples by adjusting their training losses.

- **(P4) No Supervision:** Learning with supervision, such as a clean validation set or a known noise rate, is often impractical because they are difficult to obtain. Hence, such supervision had better be avoided to increase practicality in real-world scenarios. "No Supervision" ensures that the proposed methods can be trained without any supervision.

- **(P5) Heavy Noise:** In real-world noisy data, the noise rate can vary from light to heavy. Hence, learning methods should achieve consistent noise robustness with respect to the noise rate. "Heavy Noise" ensures that the proposed methods can combat even the heavy noise.

- **(P6) Complex Noise:** The type of label noise significantly affects the performance of a learning method. To manage real-world noisy data, diverse types of label noise should be considered when designing a robust training method. "Complex Noise" ensures that the proposed method can combat even the complex label noise.

Table 3.2: Comparison of all proposed deep learning methods for overcoming noisy labels.

Category		Method	P1	P2	P3	P4	P5	P6	Implementation
Robust Loss Function (§3.1.1)		*Robust MAE*[46]	○	○	○	○	×	×	N/A
		Generalized Cross Entropy[82]	○	○	○	○	×	×	Unofficial (PyTorch)[2]
		Symmetric Cross Entropy[83]	○	○	○	○	×	×	Official (Keras)[3]
		Curriculum Learning[44]	○	○	○	×	○	△	N/A
Robust Architecture (§3.1.2)	Noisy Adaptation Layer	*Webly Learning*[84]	△	×	○	○	×	×	Official (Caffe)[4]
		Noise Model[85]	△	○	○	○	×	×	Unofficial (Keras)[5]
		Dropout Noise Model[86]	△	○	○	○	×	×	Official (MATLAB)[6]
		S-model[87]	△	○	○	○	×	×	Official (Keras)[7]
		C-model[87]	△	○	○	○	×	○	Official (Keras)[7]
		NLNN[88]	△	○	○	○	×	×	Unofficial (Chainer)[8]
	Dedicated Architecture	*Probablistic Noise Model*[20]	×	×	○	×	△	○	Official (Caffe)[9]
		Masking[89]	×	○	○	×	△	○	Official (TensorFlow)[10]
		Contrastive-Additive Network[90]	×	○	○	○	△	○	N/A
Robust Regularization (§3.1.3)		*Adversarial Training*[91]	○	○	○	○	△	△	Unofficial (PyTorch)[11]
		Label Smoothing[92]	○	○	○	○	△	△	Unofficial (PyTorch)[12]
		Mixup[93]	○	○	○	○	△	△	Official (PyTorch)[13]
		Bilevel Learning[94]	○	○	○	×	△	△	Official (TensorFlow)[14]
		Annotator Confusion[95]	○	×	○	○	△	△	Official (TensorFlow)[15]
		Pre-training[96]	○	×	○	○	△	△	Official (PyTorch)[16]
Loss Adjustment (§3.1.4)	Loss Correction	*Backward Correction*[31]	○	○	○	×	×	×	Official (Keras)[17]
		Forward Correction[31]	○	○	○	×	×	×	Official (Keras)[17]
		Gold Loss Correction[32]	○	×	○	×	×	×	Official (PyTorch)[18]
	Loss Reweigting	*Importance Reweighting*[33]	○	○	○	○	×	△	Unofficial (PyTorch)[19]
		Active Bias[34]	○	○	○	○	×	△	Unofficial (TensorFlow)[20]
	Label Refurbishment	*Bootstrapping*[35]	○	○	○	×	×	△	Unofficial (Keras)[21]
		Dynamic Bootstrapping[36]	○	○	○	○	×	△	Official (PyTorch)[22]
		D2L[37]	○	○	○	○	×	△	Official (Keras)[23]
Sample Selection (§3.1.5)		*Decouple*[38]	○	○	×	○	×	△	Official (TensorFlow)[24]
		MentorNet[39]	×	×	×	×	○	△	Official (TensorFlow)[25]
		Co-teaching[40]	○	○	×	×	○	△	Official (PyTorch)[26]
		Co-teaching+[4]	○	○	×	×	○	△	Official (PyTorch)[27]
		Iterative Detection[41]	○	○	×	○	○	△	Official (Keras)[28]
		ITLM[42]	○	○	×	×	○	△	Official (GluonCV)[29]
		INCV[1]	○	○	×	○	○	△	Official (Keras)[30]
Meta Learning (§3.1.6)	Fast Adaption	*Meta-Regressor*[48]	○	○	○	×	○	○	Official (R)[31]
		MLNT[101]	○	○	○	○	×	○	Official (PyTorch)[32]
	Learning to Update	*Knowledge Distillation*[97]	○	×	○	×	△	○	N/A
		L2LWS[98]	×	○	○	×	△	○	Unofficial (TensorFlow)[33]
		CWS[99]	×	○	○	×	△	○	N/A
		Automatic Reweighting[100]	○	○	○	×	△	○	Official (TensorFlow)[34]
		Meta-Weight-Net[102]	△	○	○	×	△	○	Official (PyTorch)[35]
		Data Coefficients[103]	○	○	○	×	○	○	Official (TensorFlow)[36]
Semi-supervised Learning (§3.1.7)		*Label Aggregation*[49]	○	×	○	×	×	△	N/A
		Two-Stage Framework[104]	○	×	○	○	○	△	N/A
		SELF[43]	○	○	○	○	○	△	N/A
		DivideMix[105]	○	○	○	○	○	△	Official (PyTorch)[37]

[2] https://github.com/AlanChou/Truncated-Loss
[3] https://github.com/YisenWang/symmetric_cross_entropy_for_noisy_labels
[4] https://github.com/endernewton/webly-supervised
[5] https://github.com/delchiaro/training-cnn-noisy-labels-keras
[6] https://github.com/ijindal/Noisy_Dropout_regularization
[7] https://github.com/udibr/noisy_labels
[8] https://github.com/Ryo-Ito/Noisy-Labels-Neural-Network
[9] https://github.com/Cysu/noisy_label
[10] https://github.com/bhanML/Masking
[11] https://https://github.com/sarathknv/adversarial-examples-pytorch

Table 3.3: Comparison of robust deep learning categories for overcoming noisy labels.

Category		P1	P2	P3	P4	P5	P6
Robust Loss Function (§3.1.1)		◯	◯	◯	◯	×	×
Robust Architecture (§3.1.2)	Noise Adaptation Layer	△	◯	◯	◯	×	×
	Dedicated Architecture	×	◯	◯	×	△	◯
Robust Regularization (§3.1.3)		◯	◯	◯	◯	△	△
Loss Adjustment (§3.1.4)	Loss Correction	◯	◯	◯	×	×	×
	Loss Reweighting	◯	◯	◯	◯	×	△
	Label Refurbishment	◯	◯	◯	△	×	△
Sample Selection (§3.1.5)		◯	◯	×	×	◯	△
Meta Learning (§3.1.6)	Fast Adaption	◯	◯	◯	△	△	◯
	Learning to Update	◯	◯	◯	×	△	◯
Semi-supervised Learning (§3.1.7)		◯	△	◯	◯	◯	△

Table 3.2 shows a comparison of all robust deep learning methods, which are grouped according to the most appropriate category. In the first row, the aforementioned six properties are labeled as P1–P6, and the availability of open source implementation is added in the last column. For each property, we assign "◯" if it is completely supported, "×" if it is not supported, and "△" if it is supported but not completely. More specifically, "△" is assigned to P1 if the method can be flexible but requires additional effort, to P5 if the method can combat only moderate label noise, and to P6 if the method does not make a strict assumption about the noise type but without explicitly modeling complex noise. The remaining properties (i.e., P2, P3, and P4) are only assigned "◯" or "×". Regarding the implementation, we assign "N/A" if a publicly available source code is not available.

No existing method supports all the properties. Each method achieves noise robustness by supporting a different combination of the properties. The supported properties are similar among the methods of the same (sub-)category because those methods share the same methodological philosophy; however, they differ significantly depending on the (sub-)category. Therefore, we investigate the properties generally supported in each (sub-)category and summarize them in Table 3.3. Here, the property of a

[12] https://github.com/CoinCheung/pytorch-loss
[13] https://github.com/facebookresearch/mixup-cifar10
[14] https://github.com/sjenni/DeepBilevel
[15] https://rt416.github.io/pdf/trace_codes.pdf
[16] https://github.com/hendrycks/pre-training
[17] https://github.com/giorgiop/loss-correction
[18] https://github.com/mmazeika/glc
[19] https://github.com/xiaoboxia/Classification-with-noisy-labels
[20] https://github.com/songhwanjun/ActiveBias
[21] https://github.com/dr-darryl-wright/Noisy-Labels-with-Bootstrapping
[22] https://github.com/PaulAlbert31/LabelNoiseCorrection
[23] https://github.com/xingjunm/dimensionality-driven-learning
[24] https://github.com/emalach/UpdateByDisagreement
[25] https://github.com/google/mentornet
[26] https://github.com/bhanML/Co-teaching
[27] https://github.com/bhanML/coteaching_plus
[28] https://github.com/YisenWang/Iterative_learning
[29] https://github.com/yanyao-shen/ITLM-simplecode
[30] https://github.com/chenpf1025/noisy_label_understanding_utilizing
[31] https://github.com/lpfgarcia/m2n
[32] https://github.com/LiJunnan1992/MLNT
[33] https://github.com/krayush07/learn-by-weak-supervision
[34] https://github.com/uber-research/learning-to-reweight-examples
[35] https://github.com/xjtushujun/meta-weight-net
[36] https://github.com/google-research/google-research/tree/master/ieg
[37] https://github.com/LiJunnan1992/DivideMix

Table 3.4: Summary of publicly available datasets used for studying label noise.

	Dataset	# Training	# Validation	# Testing	# Classes	Noise Rate (%)
Clean Data	MNIST [115][38]	60K	N/A	10K	10	≈ 0.0
	Fashion-MNIST [116][39]	60K	N/A	10K	10	≈ 0.0
	CIFAR-10 [117][40]	50K	N/A	10K	10	≈ 0.0
	CIFAR-100 [117][40]	50K	N/A	10K	100	≈ 0.0
	SVHN [118][41]	73K	N/A	26K	10	≈ 0.0
	Tiny-ImageNet [119][43]	100K	10K	10K	200	≈ 0.0
	ImageNet [5][42]	1.3M	50K	50K	1000	≈ 0.0
Real-world Noisy Data	ANIMAL-10N [19][44]	50K	N/A	5K	10	≈ 8.0
	Food-101N [2][45]	310K	5K	25K	101	≈ 18.4
	Clothing1M [20][46]	1M	14K	10K	14	≈ 38.5
	WebVision [21][47]	2.4M	50K	50K	1000	≈ 20.0

(sub-)category is marked as the majority of the belonging methods. If no clear trend is observed among those methods, then the property is marked "△".

3.3 Experimental Design

This section describes the typically used experimental design for comparing robust training methods in the presence of label noise. We introduce publicly available image datasets and then describe widely-used evaluation metrics.

3.3.1 Publicly Available Datasets

To validate the robustness of the proposed algorithms, an image classification task was widely conducted on numerous image benchmark datasets. Table 3.4 summarizes popularly-used public benchmark datasets, which are classified into two categories: (1) a "clean dataset" that consists of mostly true-labeled samples annotated by human experts and (2) a "real-world noisy dataset" that comprises real-world noisy samples with varying numbers of false labels.

Clean Datasets

According to the literature [19, 41, 105], *seven* clean datasets are widely used: MNIST[38], classification of handwritten digits [115]; Fashion-MNIST[39], classification of various clothing [116]; CIFAR-10[40] and CIFAR-100[40], classification of a subset of 80 million categorical images [117]; SVHN[41], classification of house numbers in Google Street view images [118]; ImageNet[42] and Tiny-ImageNet[43], image database organized according to the WordNet hierarchy and its small subset [5, 119]. Because the labels in these datasets are almost all true-labeled, their labels in the training data should be artificially corrupted for the evaluation of synthetic noises, namely *symmetric* noise and *asymmetric* noise.

[38] http://yann.lecun.com/exdb/mnist
[39] https://github.com/zalandoresearch/fashion-mnist
[40] https://www.cs.toronto.edu/ kriz/cifar.html
[41] http://ufldl.stanford.edu/housenumbers
[42] http://www.image-net.org
[43] https://www.kaggle.com/c/tiny-imagenet

Real-world Noisy Datasets

Unlike the clean datasets, real-world noisy datasets inherently contain many mislabeled samples annotated by non-experts. According to the literature [19, 20, 21, 2], *four* real-world noisy datasets are widely used: ANIMAL-10N[44], real-world noisy data of human-labeled online images for 10 confusing animals [19]; Food-101N[45], real-world noisy data of crawled food images annotated by their search keywords in the Food-101 taxonomy [2, 120]; Clothing1M[46], real-world noisy data of large-scale crawled clothing images from several online shopping websites [20]; WebVision[47], real-world noisy data of large-scale web images crawled from Flickr and Google Images search [21]. To support sophisticated evaluation, most real-world noisy datasets contain their own clean validation set and provide the estimated noise rate of their training set.

3.3.2 Evaluation Metrics

A typical metric to assess the robustness of a particular method is the prediction accuracy for unbiased and clean samples that are not used in training. The prediction accuracy degrades significantly if the DNN overfits to false-labeled samples [23]. Hence, *test accuracy* has generally been adopted for evaluation [17]. For a test set $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{T}|}$, let $\hat{y}_i$ be the predicted label of the i-th sample in $\mathcal{T}$. Subsequently, the test accuracy is formalized by

$$\text{Test Accuracy} = \frac{|\{(x_i, y_i) \in \mathcal{T} : \hat{y}_i = y_i\}|}{|\mathcal{T}|}. \tag{3.9}$$

If the test data are not available, *validation accuracy* can be used by replacing $\mathcal{T}$ in Eq. (3.9) with validation data $\mathcal{V} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{V}|}$ as an alternative,

$$\text{Validation Accuracy} = \frac{|\{(x_i, y_i) \in \mathcal{V} : \hat{y}_i = y_i\}|}{|\mathcal{V}|}. \tag{3.10}$$

Furthermore, if the specified method belongs to the "sample selection" category, *label precision* and *label recall* [40, 1] can be used as the metrics,

$$\begin{aligned} \text{Label Precision} &= \frac{|\{(x_i, \tilde{y}_i) \in \mathcal{S}_t : \tilde{y}_i = y_i\}|}{|\mathcal{S}_t|}, \\ \text{Label Recall} &= \frac{|\{(x_i, \tilde{y}_i) \in \mathcal{S}_t : \tilde{y}_i = y_i\}|}{|\{(x_i, \tilde{y}_i) \in \mathcal{B}_t : \tilde{y}_i = y_i\}|}, \end{aligned} \tag{3.11}$$

where $\mathcal{S}_t$ is the set of selected clean samples from a mini-batch $\mathcal{B}_t$. These two metrics are indicators of performance for the samples selected from the mini-batch as true-labeled ones [40].

Meanwhile, if the specified method belongs to the "label refurbishment" category, *correction error* [19] can be used as an indicator of how many samples are incorrectly refurbished,

$$\text{Correction Error} = \frac{|\{x_i \in \mathcal{R} : \text{argmax}(y_i^{refurb}) \neq y_i\}|}{|\mathcal{R}|}, \tag{3.12}$$

where $\mathcal{R}$ is the set of samples whose labels are refurbished by Eq. (3.7) and y_i^{refurb} is the refurbished label of the i-th samples in $\mathcal{R}$.

[44] https://dm.kaist.ac.kr/datasets/animal-10n
[45] https://kuanghuei.github.io/Food-101N
[46] https://www.floydhub.com/lukasmyth/datasets/clothing1m
[47] https://data.vision.ee.ethz.ch/cvl/webvision/download.html

3.4 Chapter Conclusions

Conclusion

In this chapter, we presented a comprehensive understanding of modern deep learning methods to address the negative consequences of learning from noisy labels. All the methods were grouped into seven categories according to their underlying strategies and described in chronological order along with their methodological weaknesses. Furthermore, a systematic comparison was conducted using six popular properties used for evaluation in the recent literature. According to the comparison results, there is no ideal method that supports all the required properties; the supported properties varied depending on the category to which each method belonged. Finally, several experimental guidelines were also discussed, including publicly available datasets and evaluation metrics.

Chapter 4. A Hybrid Learning Approach: SELFIE

Summary: Chapter based on work published at ICML 2019 [19]

Owing to the extremely high expressive power of deep neural networks, their side effect is to totally memorize training data even when the labels are extremely noisy. To overcome overfitting on the noisy labels, we propose a novel robust training method called **SELFIE**. Our key idea is to selectively refurbish and exploit unclean samples that can be corrected with high precision, thereby gradually increasing the number of available training samples. Taking advantage of this design, *SELFIE* effectively prevents the risk of noise accumulation from the false correction and fully exploits the training data. To validate the superiority of *SELFIE*, we conducted extensive experimentation using four real-world or synthetic datasets. The result showed that *SELFIE* remarkably improved absolute test error compared with two state-of-the-art methods.

4.1 Motivation and Overview

Motivation: Loss Adjustment VS. Sample Selection

Through the comprehensive review in Chapter 3, we found that there are *two* conflicting factors between "loss adjustment" and "sample selection" approaches, as shown in Table 4.1. To be specific, the family of method in loss adjustment operates well on moderately noisy data, but they fail to handle heavily noisy data owing to the accumulated noise from their false correction. On the other hand, the family of methods in sample selection achieves a much better performance on heavily noisy dataq by excluding all unclean samples containing many false-labeled instances. However, at the same time, they are not able to exploit training data fully; that is, their partial exploration rather eliminates numerous useful samples that might help robust training.

Table 4.1: Comparison of "loss adjustment" and "sample selection" approaches.

Category		Method	Full Exploration	Heavy Noise
Loss Adjustment (§3.1.4)	Loss Correction	*Backward Correction* [31]	○	×
		Forward Correction [31]	○	×
		Gold Loss Correction [32]	○	×
	Loss Reweigting	*Importance Reweighting* [33]	○	×
		Active Bias [34]	○	×
	Label Refurbishment	*Bootstrapping* [35]	○	×
		Dynamic Bootstrapping [36]	○	×
		D2L [37]	○	×
Sample Selection (§3.1.5)		*Decouple* [38]	×	○
		MentorNet [39]	×	○
		Co-teaching [40]	×	○
		Co-teaching+ [4]	×	○
		Iterative Detection [41]	×	○
		ITLM [42]	×	○
		INCV [1]	×	○
Hybrid Approach		**SELFIE [19]**	○	○

To the best of our knowledge, our proposed method *SELFIE* is the first method to satisfy the two conflicting factors: *heavy noise* and *full exploration*, by taking advantage of both loss adjustment and sample selection. Notably, the concept of the refurbishable samples in *SELFIE* enables to minimize the number of falsely corrected samples as well as to exploit full exploration of training data.

Overview: Hybrid Learning via SELFIE

A c-class classification problem requires the training data $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N$, where x_i is a sample and $y_i^* \in \{1, 2, \dots, c\}$ is its *true* label. Following the label noise scenario, let's consider the noisy training data $\tilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^N$, where $\tilde{y}_i \in \{1, 2, \dots, c\}$ is a *noisy* label which may not be true. Then, in conventional training, when a mini-batch $\mathcal{B}_t = \{(x_i, \tilde{y}_i)\}_{i=1}^b$ consists of b samples randomly drawn from the noisy training data $\tilde{\mathcal{D}}$ at time t, the network parameter Θ_t is updated in the descent direction of the expected loss on the mini-batch $\mathcal{B}_t$,

$$\Theta_{t+1} = \Theta_t - \eta \nabla \big(\frac{1}{|\mathcal{B}_t|} \sum_{x \in \mathcal{B}_t} \mathcal{L}(f(x; \Theta_t), \tilde{y}) \big), \tag{4.1}$$

where η is a learning rate and $\mathcal{L}$ is a loss function.

Following the hybrid principle of *SELFIE*, we modify the update equation to render the network more robust on noisy labels. Let $\mathcal{C}_t \subseteq \mathcal{B}_t$ be the *clean* samples and $\mathcal{R}_t \subseteq \mathcal{B}_t$ be the *refurbishable* samples. We correct the backward loss of the refurbishable sample $x \in \mathcal{R}_t$ by replacing its corrupted label $\tilde{y}$ with the *refurbished* label y^{refurb}. Subsequently, we back-propagate the losses for the refurbishable and clean samples to update the network,

$$\Theta_{t+1} = \Theta_t - \eta \nabla \Big(\frac{1}{|\mathcal{R}_t \cup \mathcal{C}_t|} \big(\sum_{x \in \mathcal{R}_t} \mathcal{L}(f(x; \Theta_t), y^{refurb}) \; + \sum_{x \in \mathcal{C}_t \cap \mathcal{R}_t^c} \mathcal{L}(f(x; \Theta_t), \tilde{y}) \big) \Big). \tag{4.2}$$

Here, it is *not* necessarily true that $\mathcal{R}_t \cap \mathcal{C}_t = \emptyset$. If a sample $x \in \mathcal{R}_t \cap \mathcal{C}$, being refurbishable precedes being clean because false-labeled instances could be included even in $\mathcal{C}_t$;[48] that is, the sample x needs to be refurbished.

To update the network by Eq. (4.2), the key challenge is how to construct $\mathcal{R}_t$ as well as correct the loss of $x \in \mathcal{R}_t$, which will be discussed in the next section. On the contrary, there have been extensive studies on how to construct $\mathcal{C}_t$. Thus, for $\mathcal{C}_t$, we simply adopt the widely used *small-loss* trick [39, 40] that selects $(1 - \tau) \times 100\%$ of small-loss instances as clean samples, where τ is the noise rate. If τ is unknown, τ can be inferred using cross-validation [121, 97].

4.2 Main Concept: Selective Loss Correction

4.2.1 Criterion of Refurbishable

Interestingly, before the network fully fits the noisy labels, the label prediction of false-labeled samples either (1) changes inconsistently or (2) corresponds to their *true* labels with high probability owing to the learner's perceptual consistency [47].

Hence, we aim to distinguish between the two cases to identify the samples that can be refurbished with high precision. Intuitively, the samples with consistent label predictions are regarded as refurbishable. The notion of being *refurbishable* is formalized as Definition 4.2.1 in which the predictive uncertainty uses the entropy to measure the consistency of label prediction.

[48] The evidence that $\mathcal{C}_t$, in fact, has quite many false-labeled samples is presented in Chapter 4.4.2.

Definition 4.2.1: Refurbishable Sample

A sample x is *refurbishable* if the predictive uncertainty in Eq. (4.3) $U(x,t;q) \leq \epsilon$ $(0 \leq \epsilon \leq 1)$.

$$U(x,t;q) = \big(1/\log(k)\big)\ \text{entropy}\big(p(y|x,t;q)\big) \quad \square \tag{4.3}$$

The detail of $U(x,t;q)$ is as follows. Let $\hat{y}_t = f(x,\Theta_t)$ be the predicted label of the sample x at time t and $\mathcal{H}_x^t(q) = \{\hat{y}_{t_1}, \hat{y}_{t_2}, \ldots, \hat{y}_{t_q}\}$ be the label history of the sample x that stores the predicted labels of the previous q epochs. Next, the probability of a label $y \in \{1,2,\ldots,c\}$ estimated as the label of the sample x is formulated by

$$p(y|x,t;q) = \frac{1}{|\mathcal{H}_x^t(q)|} \sum_{\hat{y} \in \mathcal{H}_x^t(q)} [\hat{y} = y], \qquad \text{where } [S] = \begin{cases} 1, & \text{if } S \text{ is true.} \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Then, we adopt the information entropy [122] to quantify uncertainty; hence, the predictive uncertainty $U(x;q)$ in is now completed by

$$\text{entropy}(p(y|x,t;q)) = -\sum_{j=1}^{c} p(j|x,t;q) \log p(j|x,t;q). \tag{4.5}$$

For c classes, to ensure $0 \leq U(x,t;q) \leq 1$, the empirical entropy is divided by $\log(k)$, which is the maximum value obtained when $\forall_j p(j|x,t;q) = 1/c$.

4.2.2 Loss Correction by Refurbishment

The loss of the refurbishable sample is corrected by replacing the corrupted label $\tilde{y}$ with the *refurbished* label y^{refurb} in Definition 4.2.2. Subsequently, it is combined with those of $(1-\tau) \times 100\%$ small-loss instances within the mini-batch, and back-propagated to update the network, as in Eq. (4.2). The leftover instances that might accumulate label noises are excluded from the update to pursue the robust learning.

Definition 4.2.2: Refurbished Label

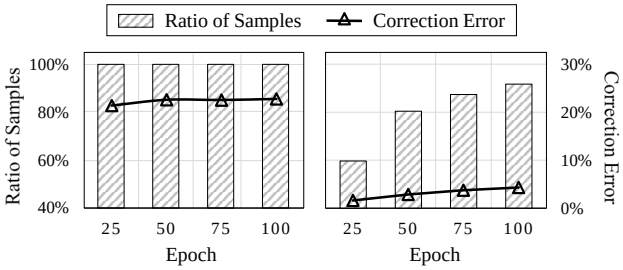
A *refurbished* label y^{refurb} of the refurbishable sample x is the most frequently predicted label for previous q times, as in Eq. (4.6), where the sample x satisfies the condition $U(x,t;q) \leq \epsilon$.

$$y^{refurb} = \text{argmax}_{1 \leq j \leq c}\ p(j|x,t;q) \quad \square \tag{4.6}$$

4.2.3 Quick Analysis

We peep at the experiment result to demonstrate the advantage of *selectively* correcting losses over *entirely* correcting losses. We trained DenseNet (L=25, k=12) using *SELFIE* on a noisy CIFAR-10 dataset. For the entire correction method, we set ϵ to be 1 such that all samples were considered to be refurbishable regardless of their predictive uncertainty. For the selective correction method, we set ϵ as 0.05 such that only samples with low predictive uncertainty were considered to be refurbishable. As shown in Figure 4.1(a), the entire correction method uses all training samples during training, but it suffers from the high correction error on training samples of over 20%. That is, the network consistently accumulates the noise from false-labeled samples, thus causing poor generalization on test data.

Conversely, as shown in Figure 4.1(b), the selective correction method reduces the noise by taking a part of training samples in the early stage of training (e.g., at the 25-th epoch), thereby achieving a significantly low correction error under 2%. Most importantly, even if only 60% of training samples are used for training initially, more training samples are added incrementally as the training epoch increases, while the low correction error is maintained at under 5%. Therefore, it is evident that the selective method guides the network to be trained more robustly on noisy data.



(a) Entire Correction. (b) Selective Correction (Ours).

Figure 4.1: Analysis on entire and selective loss correction methods using DenseNet (L=25, k=12) on CIFAR-10 with symmetry noise 40%. The ratio of samples used for training (hatched bar) is plotted with the correction error (line).

4.3 Algorithm Description

4.3.1 Main Algorithm: SELFIE

Algorithm 1 describes the overall procedure of *SELFIE*. First, during warm-up, by following the convention of the robust training, the network is trained on all training samples in the *default* manner, as shown in Eq. (4.1) (Lines 7-13). Even with the existence of noisy labels, deep networks learn clean and easy instances in the warm-up period without noise accumulation, which is known as *memorization* effect [22, 39]. Subsequently, after the warm-up period, $(1-\tau) \times 100\%$ of the small-loss samples are selected as clean samples $\mathcal{C}_t$ from the mini-batch $\mathcal{B}_t$ (Lines 14–16), and refurbishable samples $\mathcal{R}_t$ are identified and corrected by checking the condition for predictive uncertainty (Lines 17–23). Here, the refurbished samples $\mathcal{R}_t$ are aggregated for reuse (Lines 24–25). After that, the network is updated based on the clean samples $\mathcal{C}_t$ along with the refurbished samples $\mathcal{R}_t$, as shown in Eq. (4.2) (Lines 26–27). Then, the label history is updated at the end of every epoch (Line 28).

For more robust training, Algorithm 1 can be restarted multiple times with reusing the output $\mathbb{R}$ of the previous run (Line 1) as well as initializing the model parameter (Lines 2–4). This restart technique enables the network to be re-trained on *less-noisy* training data refurbished in the previous runs. In other words, a bunch of already refurbished samples are readily available from the very beginning of the current run. We demonstrate the effect of using the restart technique in Chapter 4.5.2.

4.3.2 Collaboration with Co-teaching: Co-SELFIE

An advantage of *SELFIE* is its *flexibility* with regard to collaboration with other orthogonal studies because it only needs a simple modification in the gradient descent step. Herein, for further improvement, we introduce *Co-SELFIE* combined with *Co-teaching* [40], which is a state-of-the-art robust training algorithm. *Co-SELFIE* maintains two networks simultaneously. In each mini-batch, each network identifies its own clean and refur-

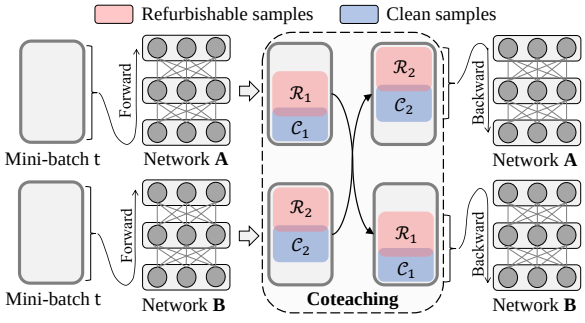


Figure 4.2: Training procedure of *Co-SELFIE*.

Algorithm 1 *SELFIE* Algorithm

Require: $\tilde{\mathcal{D}}$: noisy data, *epochs*, γ: warm-up, τ: noise rate, ϵ: uncertainty threshold, q: history length, *restart*: # restarts

Ensure: Θ_t: model parameter, $\mathbb{R}$: the entire refurbished samples

/* $\mathbb{R}$ is the entire refurbished samples in $\tilde{\mathcal{D}}$ */
$\mathbb{R} \leftarrow \emptyset$;
for $r = 0$ **to** *restart* **do**
  $t \leftarrow 1$;
  $\Theta_t \leftarrow$ Initialize the model parameter;
  /* **Learning from a noisy training dataset** $\tilde{\mathcal{D}}$ */
  for $i = 1$ **to** *epochs* **do**
    for $j = 1$ **to** $|\tilde{\mathcal{D}}|/|\mathcal{B}_t|$ **do**
      Draw a mini-batch $\mathcal{B}_t$ from $\tilde{\mathcal{D}}$;
      /* Warm-up periods */
      if $i \leq \gamma$ **then**
        /* Update by Eq. (4.1) */
        $\Theta_{t+1} = \Theta_t - \eta \nabla \big(\frac{1}{|\mathcal{B}_t|} \sum_{x \in \mathcal{B}_t} \mathcal{L}(f(x; \Theta_t), \tilde{y}) \big)$;
      else
        /* Clean sample selection */
        $\mathcal{C}_t \leftarrow (1 - \tau) \times 100\%$ of small-loss samples in $\mathcal{B}_t$;
        /* Selective loss correction in Sec. 4.2 */
        $\mathcal{R}_t \leftarrow \emptyset$; /* $\mathcal{R}_t$ is refurbished samples in $\mathcal{B}_t$ */;
        for each $x \in \mathcal{B}_t$ **do**
          /* By Eq. (4.3) */
          if $U(x, t; q) \leq \epsilon$ **or** $x \in \mathbb{R}$ **do**
            /* Refurbish */
            $\mathcal{R}_t \leftarrow \mathcal{R}_t \cup (x, y^{refurb})$;
        /* Aggregation */
        $\mathbb{R} \leftarrow \mathbb{R} \cup \mathcal{R}_t$;
        /* Update by Eq. (4.2) */
        $\Theta_{t+1} = \Theta_t - \eta \nabla \Big(\frac{1}{|\mathcal{R}_t \cup \mathcal{C}_t|} \big(\sum_{x \in \mathcal{R}_t} \mathcal{L}(f(x; \Theta_t), y^{refurb}) + \sum_{x \in \mathcal{C}_t \cap \mathcal{R}_t^c} \mathcal{L}(f(x; \Theta_t), \tilde{y}) \big) \Big)$;
      Update_Label_History(Get_Label($\mathcal{B}_t, \Theta_t$));
      $t \leftarrow t + 1$;
return $\Theta_t, \mathbb{R}$;

bishable samples and feeds such samples to its peer network for further training, as demonstrated in Figure 4.2. A mini-batch t is provided to the network **A** and the network **B**; $\mathcal{R}_t^A$ and $\mathcal{C}_t^A$ are obtained from the network **A**, and $\mathcal{R}_t^B$ and $\mathcal{C}_t^B$ are obtained from the network **B**; for backpropagation, $\mathcal{R}_t^A$ and $\mathcal{C}_t^A$ are fed to the network **B**, and $\mathcal{R}_t^B$ and $\mathcal{C}_t^B$ are fed to the network **A**. It is known that *Co-teaching* effectively removes the error incurred by the biased selection of training samples [40]. The advantage of *SELFIE* is boosted by *Co-teaching*. We also demonstrate the improvement of *Co-SELFIE* over *SELFIE* in Chapter 4.5.3.

4.4 Main Experiments

Detasets: To validate the superiority of *SELFIE*, we performed an image classification task on *four* benchmark datasets: CIFAR-10 (10 classes), CIFAR-100 (100 classes), Tiny-ImageNet (200 classes), and ANIMAL-10N (10 classes). ANIMAL-10N is our proprietary real-world noisy dataset of human-labeled online images for 10 confusing animals, with $50,000$ training and $5,000$ testing images. Please note that, in ANIMAL-10N, noisy labels were injected *naturally* by human mistakes, where its noise rate was estimated at 8%. It has been released on our site[49], and its details can be found in Chapter A. We did not apply any data augmentation or pre-processing procedures.

Noise Injection: Except ANIMAL-10N, since all datasets are clean, we artificially corrupted these datasets using a typical method for the evaluation of noisy labels [31, 40, 47]. As shown in Figure 4.9, for c classes, we applied the *label transition matrix* T: *(i) asymmetric noise* and *(ii) symmetric noise*. It is known that asymmetric noise is more realistic than symmetric noise because labelers may induce mistakes only within few classes [40, 100]. To evaluate the robustness on varying noise rates from light noise to heavy noise, we tested five noise rates $\tau \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$.

Network and Hyperparameters: For the classification task, we trained DenseNet (L=25, k=12) [123] and VGG-19 [124] with a momentum optimizer. Specifically, we used a momentum of 0.9, a batch size of 128, a dropout of 0.1 [26], and batch normalization [27]. For the training schedule, following the experimental setup of Huang et al. [123], we trained the network for 100 epochs and used an initial learning rate of 0.1, which was divided by 5 at 50% and 75% of the total number of epochs, respectively. Regarding the hyperparameters, we fixed *restart* to 2 (i.e., restarted Algorithm 1 *twice* after the first run) and used the best uncertainty threshold $\epsilon = 0.05$ and history length $q = 15$, which were obtained from a grid $\epsilon = \{0.05, 0.10, 0.15, 0.20\}$ and $q = \{10, 15, 20\}$ (please see Chapter 4.4.3 for details). The warm-up threshold γ was set to 25 for the initial learning.

Algorithms: We compared *SELFIE* with a baseline algorithm (denoted by *Default*) and two state-of-the-art robust training algorithms. *Default* trains the network without any processing for the noisy labels. The others are the representatives of loss correction and sample selection strategies, respectively. *Active Bias* [34] corrects the backward loss of training samples by the prediction variance of true label probabilities. *Co-teaching* [40] selects the clean samples by the loss-based trick and adopts the co-training [125] mechanism. All the algorithms were implemented using TensorFlow 1.8.0 and executed using a single NVIDIA Titan Volta GPU. For reproducibility, we provide the source code at https://github.com/kaist-dmlab/SELFIE.

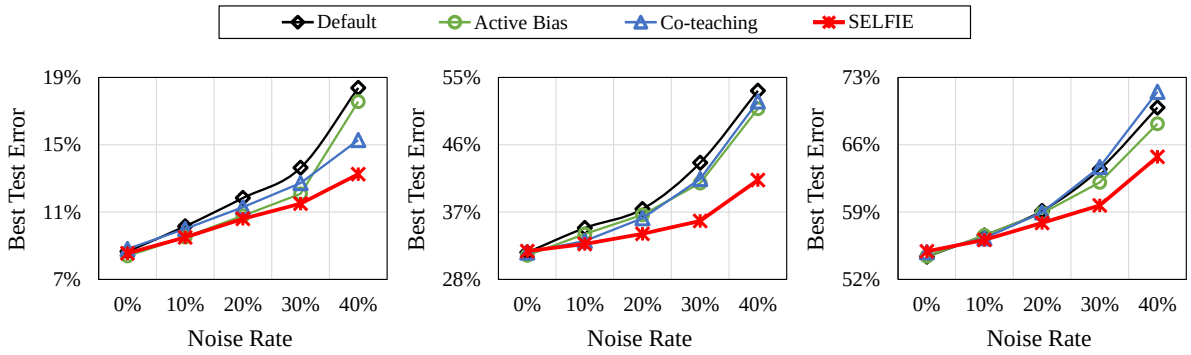
In support of reliable evaluation, we repeated every test *thrice* and reported the average and standard error of the best test errors. The best test error obtained during training is a common measure of robustness to noisy labels [38, 39].

4.4.1 Performance Comparison

Result with Synthetic Noise using DenseNet

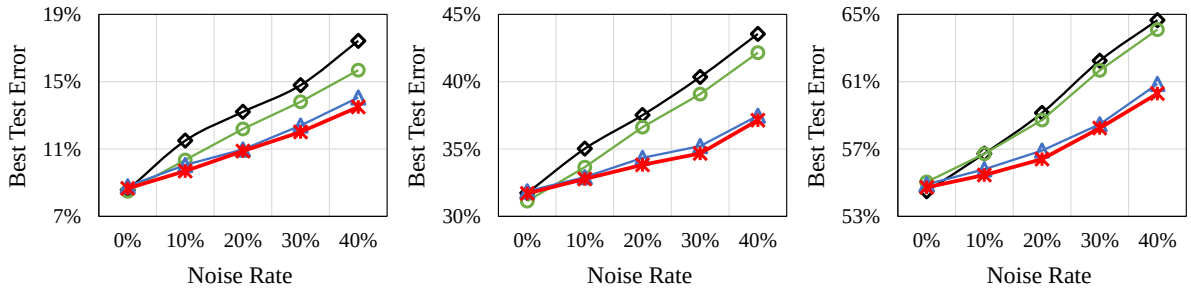
- **asymmetric Noise**: Figure 4.3 shows the test error of the four training methods using DenseNet (L=25, k=12) on three datasets with varying asymmetric noise rates. Generally, at any noise rate, *SELFIE* achieved the lowest test error on all datasets. The difference in the test errors between *SELFIE* and

[49] https://dm.kaist.ac.kr/datasets/animal-10n/



(a) CIFAR-10. (b) CIFAR-100. (c) Tiny-ImageNet.

Figure 4.3: The best test error of the four training methods using **DenseNet (L=25, k=12)** on three datasets with varying **asymmetric noise** rates.



(a) CIFAR-10. (b) CIFAR-100. (c) Tiny-ImageNet.

Figure 4.4: The best test error of the four training methods using **DenseNet (L=25, k=12)** on three datasets with varying **symmetric noise** rates.

Table 4.2: The best test error (%) on **asymmetric noise** 40% in Figure 4.3.

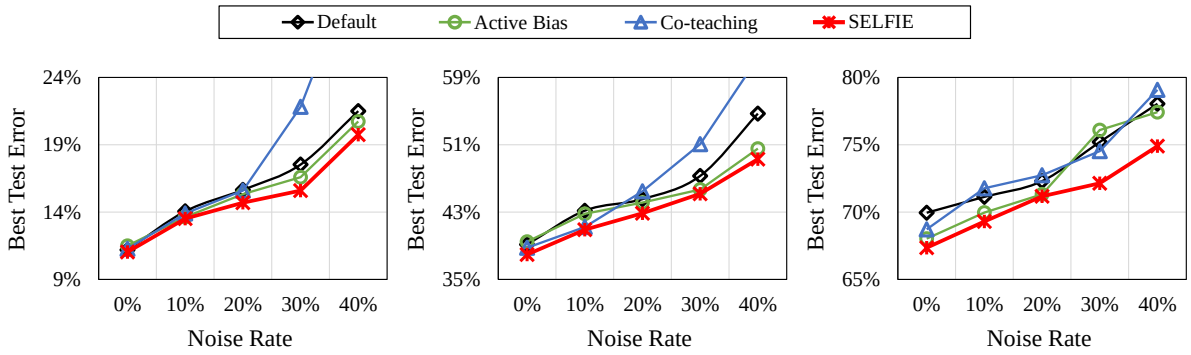
Method	CIFAR-10	CIFAR-100	Tiny-Image
Default	18.4±0.35	53.2±0.46	69.9±0.28
Active Bias	17.6±0.33	50.8±0.08	68.2±0.06
Co-teaching	15.3±0.30	51.8±0.75	71.5±0.19
SELFIE	**13.2±0.06**	**41.3±0.15**	**64.7±0.08**

Table 4.3: The best test error (%) on **symmetric noise** 40% in Figure 4.4.

Method	CIFAR-10	CIFAR-100	Tiny-Image
Default	17.4±0.25	43.6±0.21	64.7±0.21
Active Bias	15.7±0.39	42.2±0.29	64.1±0.35
Co-teaching	14.1±0.18	37.5±0.01	60.9±0.13
SELFIE	**13.5±0.04**	**37.1±0.02**	**60.3±0.08**

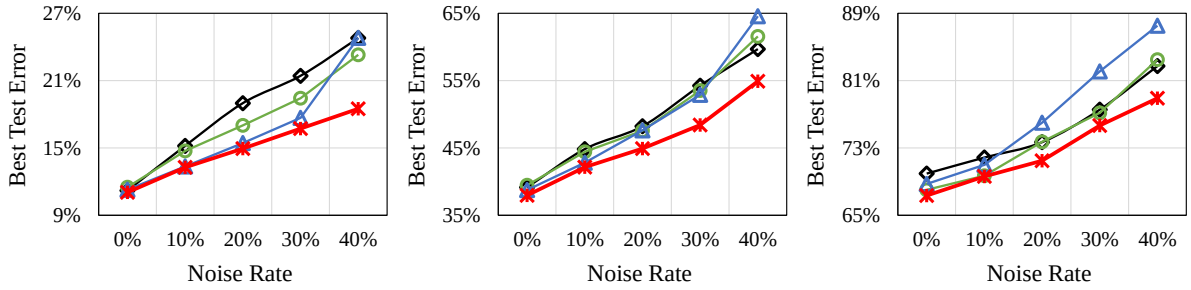
other methods increased as the noise rate increased. In particular, at the heavy noise rate of 40%, *SELFIE* significantly reduced the *absolute* test error by 5.2*pp*–11.9*pp* compared with *Default*, 3.5*pp*–9.5*pp* compared with *Active Bias*, and 2.1*pp*–10.5*pp* compared with *Co-teaching*. The test errors of all methods at the asymmetric noise rate of 40% are summarized in Table 4.2. Although the test errors of *Active Bias* and *Co-teaching* were lower than that of *Default*, they were not comparable to that of *SELFIE* except the light noise rates of 0%–20%. For a more robust training, this significant improvement in *SELFIE* empirically proves that it is essential to *(i)* selectively correct the unclean samples and *(ii)* exploit the full exploration of the training data.

- **Symmetric Noise**: Figure 4.4 shows the test error of the four training methods on three datasets using DenseNet (L=25, k=12) with varying symmetric noise rates. Similar to the asymmetric noise,



(a) CIFAR-10. (b) CIFAR-100. (c) Tiny-ImageNet.

Figure 4.5: The best test error of the four training methods using **VGG-19** on three datasets with varying **asymmetric noise** rates.



(a) CIFAR-10. (b) CIFAR-100. (c) Tiny-ImageNet.

Figure 4.6: The best test error of the four training methods using **VGG-19** on three datasets with varying **symmetric noise** rates.

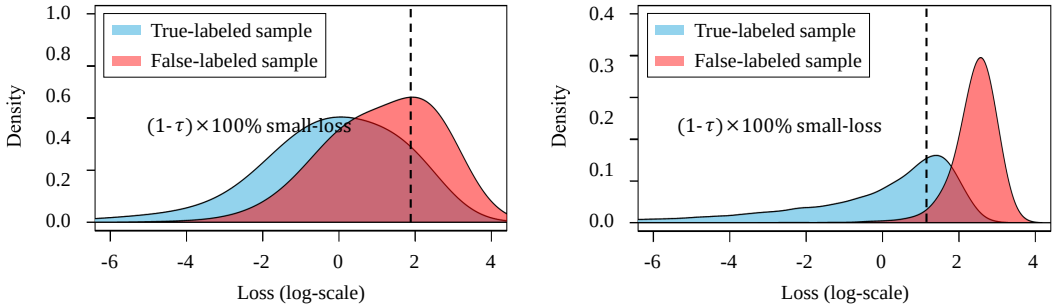
Table 4.4: The best test error (%) on **asymmetric noise** 40% in Figure 4.5.

Method	CIFAR-10	CIFAR-100	Tiny-Image
Default	21.5±1.37	54.7±0.13	78.0±0.26
Active Bias	20.7±0.09	50.5±0.57	77.4±0.66
Co-teaching	32.9±0.77	61.3±1.77	79.1±0.13
SELFIE	**19.7±0.18**	**49.3±0.10**	**74.9±0.11**

Table 4.5: The best test error (%) on **symmetric noise** 40% in Figure 4.6

Method	CIFAR-10	CIFAR-100	Tiny-Image
Default	24.8±0.19	59.7±0.17	82.7±0.60
Active Bias	23.3±0.31	61.6±1.65	83.5±1.04
Co-teaching	24.8±0.92	64.5±3.46	87.5±0.41
SELFIE	**18.5±0.13**	**54.9±0.38**	**78.9±0.33**

SELFIE generally outperformed other methods at any noise rate on all datasets. Quantitatively, at the heavy noise rate of 40%, *SELFIE* significantly reduced the *absolute* test error by 3.9*pp*–6.5*pp* compared with *Default*, 2.2*pp*–5.1*pp* compared with *Active Bias*, and 0.4*pp*–0.6*pp* compared with *Co-teaching*. The test errors of all methods at the symmetric noise rate of 40% are summarized in Table 4.3. Unlike the asymmetric noise, *Co-teaching* achieved a low test error comparable to *SELFIE*. In contrast, *Active Bias* showed a slightly better performance than *Default*, but was significantly worse than *SELFIE* and *Co-teaching*. Additionally, *Default* tended to show vulnerability even with the light noise rate of 10%.



(a) asymmetric Noise 40%. (b) Symmetry Noise 40%.

Figure 4.7: Loss distributions at the training accuracy of 50% on a noisy CIFAR-100 dataset.

Result with Synthetic Noise using VGG

Figures 4.5 and 4.6 show the test errors of the four training methods using VGG-19 with varying asymmetric and symmetric noise rates. The performance trends with VGG was similar to that with DenseNet. Again, *SELFIE* achieved the lowest test error at any noise rate of both noise types on all datasets. *SELFIE* outperformed the other methods by a larger margin at a higher noise rate. Especially when the noise rate was 40%, *SELFIE* reduced the *absolute* test error by 1.8*pp*–5.4*pp* compared with *Default*, 1.0*pp*–2.5*pp* compared with *Active Bias*, and 4.2*pp*–13.2*pp* compared with *Co-teaching* for the asymmetric noise; by 3.8*pp*–6.3*pp* compared with *Default*, 4.6*pp*–6.7*pp* compared with *Active Bias*, and 6.3*pp*–9.6*pp* compared with *Co-teaching* for the symmetric noise. On the other hand, *Active Bias* achieved the test error slightly lower than that of *Default*, and *Co-teaching* worked well only on CIFAR-10 with symmetric noise, i.e., Figure 4.6(a). Tables 4.4 and 4.5 summarize the test errors of all methods at the noise rate of 40% for the asymmetric and symmetric noises, respectively.

Result with Real-world Noise

Table 4.6 summarizes the best test errors of the four training methods using the two architectures on ANIMAL-10N. In both architectures, *SELFIE* achieved the lowest test error. Specifically, *SELFIE* improved the *absolute* test error by up to 0.9*pp* using DenseNet (L=25, k=12) and 2.4*pp* using VGG-19. *SELFIE* maintained its dominance over other methods on *real-world* noise, though the performance gain was not that huge because of a light noise rate (i.e., 8%).

Table 4.6: The best test errors (%) on ANIMAL-10N (8% noise).

Method	DenseNet	VGG-19
Default	17.9±0.02	20.6±0.14
Active Bias	17.6±0.17	19.5±0.26
Co-teaching	17.5±0.17	19.8±0.13
SELFIE	**17.0±0.10**	**18.2±0.09**

4.4.2 Anatomy of Small-Loss Trick

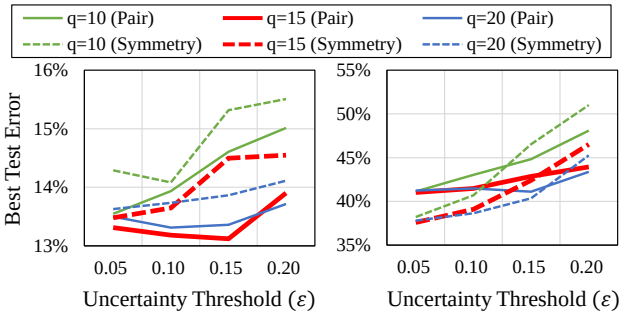
An interesting observation is the considerable performance difference in *Co-teaching* for asymmetric and symmetric noises, i.e., Figures 4.3 and 4.4. The difference was found to be due to the small-loss trick proposed by *Co-teaching*. As demonstrated in Figure 4.7(a), for the asymmetric noise, the distribution of false-labeled samples was overlapped closely with that of true-labeled samples. That is, clean (i.e., $(1-\tau)\times 100\%$ small-loss) samples contained a significant number of false-labeled samples, thereby causing the network to accumulate the label noise. In contrast, for the symmetric noise in Figure 4.7(b), the

two distributions were clearly separated. Most false-labeled samples exhibited a much higher loss than true-labeled samples.

SELFIE also adopts the small-loss trick to select clean samples, but achieved a remarkable performance for the asymmetric noise, as shown in Figure 4.3, because even clean samples are regarded as refurbishable if their label does not conform to the most frequently predicted label as in Eq. (4.2). Consequently, the labels of false-labeled samples are refurbished with high precision, even when they are classified as clean.

4.4.3 Hyperparmeter Selection

SELFIE receives the two hyperparameters: the uncertainty threshold ϵ and the history length q. To decide the best hyperparameters, we trained DenseNet (L=25, k=12) on CIFAR-10 and CIFAR-100, each of which was corrupted by asymmetric noise and symmetric noise at a rate of 40%. For the hyperparameter selection, the two hyperparameters were chosen in a grid $\epsilon \in \{0.05, 0.10, 0.15, 0.20\}$ and $q \in \{10, 15, 20\}$. Figure 4.8 shows the test error of *SELFIE* obtained by the grid search on the two noisy datasets. Regarding the uncertainty threshold ϵ, lower test error was generally achieved with a smaller ϵ, because it induces that the more consistent samples from label predictions become the refurbishable samples. As for the history length q, the smallest q tended to yield the worst performance in the two datasets, regardless of the noise type. Although there is no clear winner between $q = 15$ and $q = 20$, the q value of 15 achieved the smallest test error when the ϵ value was the smallest at 0.05. Therefore, in all experiments, we set the uncertainty threshold ϵ to 0.05 and the history length q to 15.



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

Figure 4.8: Grid search on CIFAR-10 and CIFAR-100 with a noise rate of 40%.

4.5 Ablation Study

4.5.1 Accuracy of Loss (Label) Correction

To verify the accuracy of the loss (label) correction, we show the confusion matrices before and after the correction in Figure 4.9. In the left column, the confusion matrices before correction correspond to the label transition matrices. Many entries other than the diagonal entries have non-negligible probability because of asymmetric or symmetric noises. In the right column, the confusion matrices after correction contain the refurbished labels determined by Definition 4.2.2. As opposed to the label transition matrices, only few entries other than the diagonal entries have non-negligible probability. Although the noise rate was very high (40%), most of the diagonal entries had probability over 0.95, thus proving very high correction accuracy. Therefore, a large portion of noises was successfully cleared by *SELFIE*.

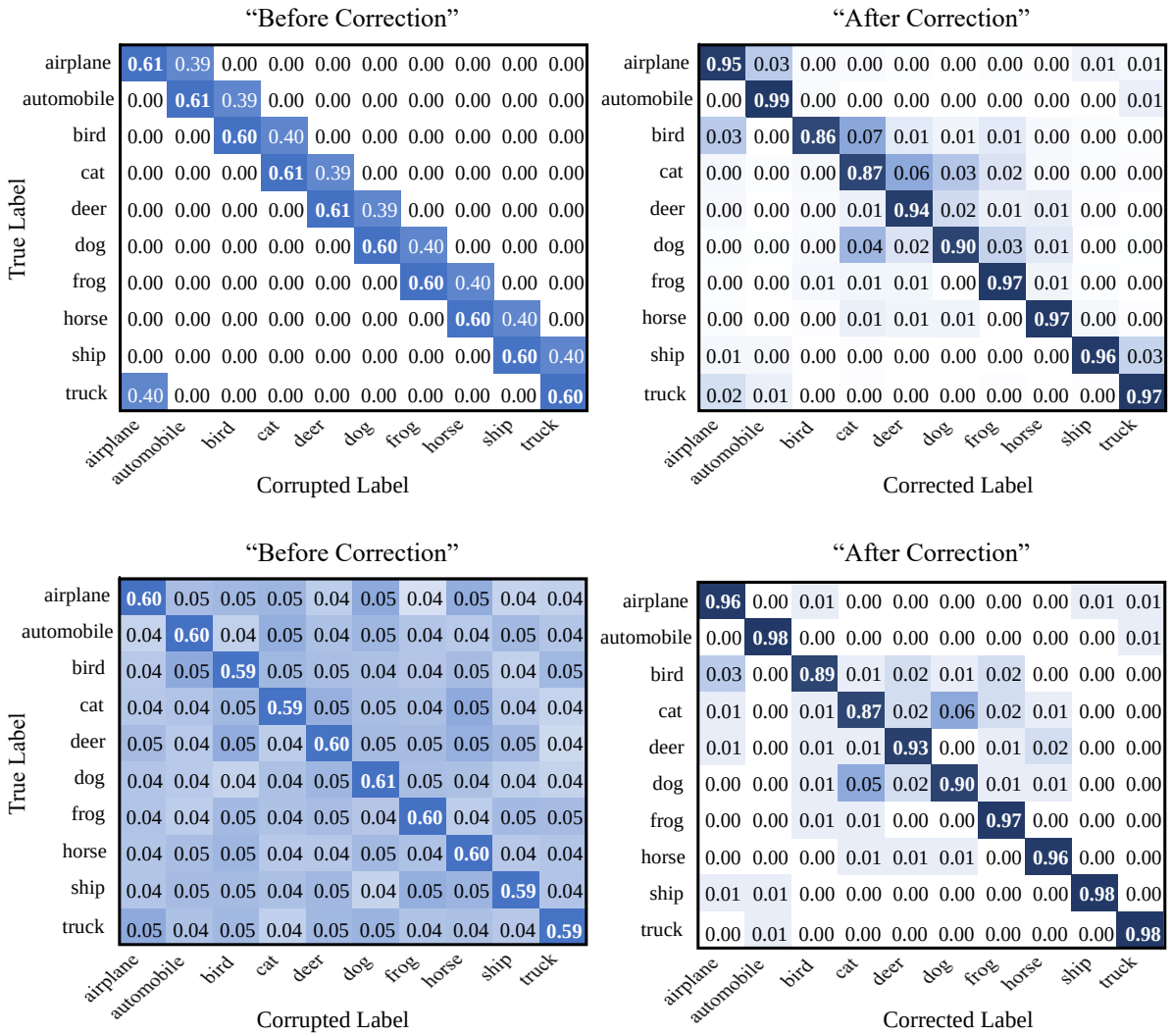
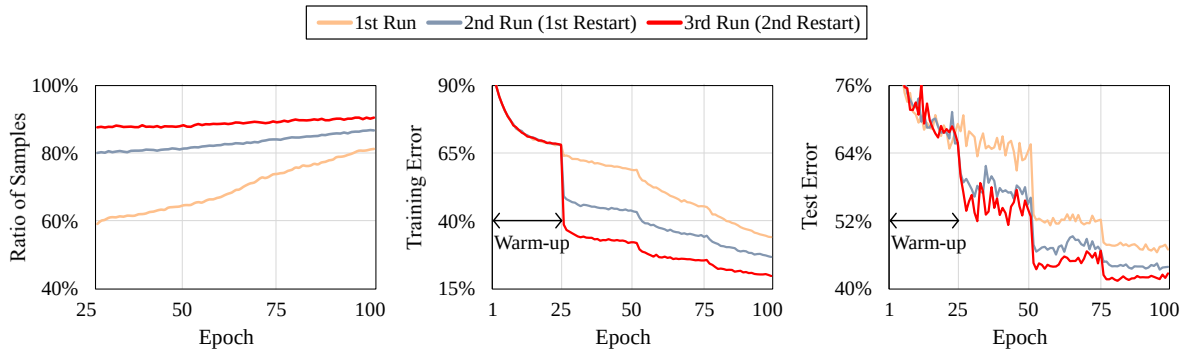


Figure 4.9: Confusion matrices on CIFAR-10 with (a) **asymmetric noise** 40% and (b) **symmetric noise** 40%.

4.5.2 Performance Improvement by Restarts

Figure 4.10 shows the effect of the restart technique on CIFAR-100 with a asymmetric noise of 40%. As shown in Figure 4.10(a), the number of samples available for training increased from 59.2% to 90.2% of the total training samples, as the number of runs increased. This effect encourages the network to be re-trained on a greater amount of training samples, which were refurbished in the previous runs. Therefore, the training and test errors were improved significantly in the next run, as shown in Figures 4.10(b) and 4.10(c). In detail, the training and test errors were 34.1% and 46.7% at the end of the first run, but were improved continuously through restart. They reached 26.8% and 43.4% at the end of the second run (i.e., first restart) and then 19.8% and 41.3% at the end of the third run (i.e., second restart). It is noteworthy that *SELFIE* achieved a significant reduction in *absolute* test error of 5.4*pp* using the restart technique. We expect that a larger number of restarts will further improve the performance of *SELFIE* at the expense of the training time.

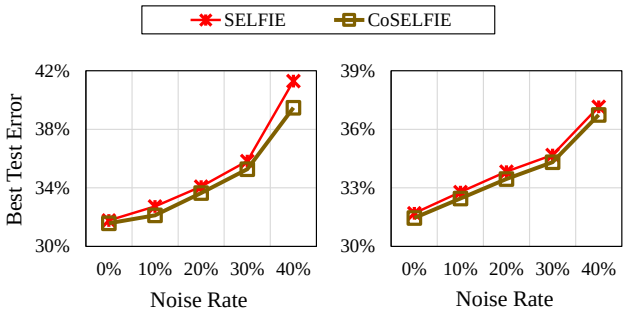


(a) Ratio of Samples. (b) Training Error. (c) Test Error.

Figure 4.10: Restart on CIFAR-100 with **asymmetric noise** 40%: (a) shows the ratio of samples used for training, (b) shows the reduction in training error, and (c) shows the reduction in test error.

4.5.3 Performance Improvement by Co-teaching

Please recall that *Co-SELFIE* (Section 4.3.2) is an extension of *SELFIE* based on *Co-teaching* [40]. Figure 4.11 shows the test errors of *SELFIE* and *Co-SELFIE* on CIFAR-100 with varying noise rates. Interestingly, by collaborating with *Co-teaching*, the test error of *SELFIE* was further improved in both noise types. In particular, in the asymmetric noise, the difference in the test errors between *SELFIE* and *Co-SELFIE* tended to be larger as the noise rate increased. Quantitatively, compared with *SELFIE*, *Co-SELFIE* reduced the *absolute* test error by 0.20*pp*–1.82*pp* in the asymmetric noise and 0.24*pp*–0.41*pp* in the symmetric noise.



(a) Asymmetric Noise. (b) Symmetry Noise.

Figure 4.11: Performance improvement of *Co-SELFIE* on CIFAR-100 with varying noise rates.

4.6 Chapter Conclusions

Conclusion

In this chaper, we proposed a novel hybrid method for robust training on noisy data, which we call *SELFIE*, that trains the network on precisely calibrated samples together with clean samples. Toward this goal, we introduced the concept of *selective loss correction* that identifies refurbishable samples and corrects their label with high precision. We conducted extensive experiments using two popular convolutional neural networks on four datasets with varying noise rates. Our experiment results showed that the robustness of a deep neural network on noisy data can be significantly improved by the selective loss correction on refurbishable samples. *SELFIE* guided the network to avoid noise accumulation from the false correction and allowed it to take advantage of the full exploration of training data. In addition, the results showed that the performance of *SELFIE* can be further improved by restarts and collaboration with other work.

Chapter 5. A Two-Phase Learning Approach: Prestopping

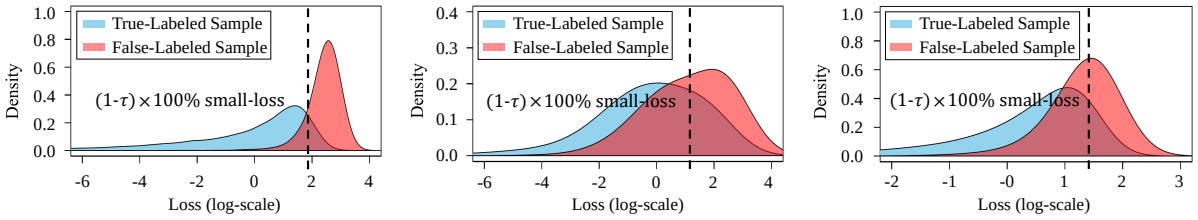
Summary: Chapter based on work being under review at ICLR 2020 [50]

In *SELFIE*, we adopted a widely used *small-loss* trick as its component for sample selection. In this chapter, we argue that selecting small-loss samples misclassifies many false-labeled samples as clean ones especially in *realistic* label noise and, accordingly, accumulates severe label noise from them. To handle this problem, we focus on the fact that such accumulated noise can be prevented by "early stopping" training a deep neural network before the noisy labels are severely memorized. Then, we present a *maximal safe set*, which is initially derived by the early stopped network, to resume the remaining learning process. Most importantly, since the early stop point, the maximal safe set maintains a collection of almost certainly true-labeled samples at each epoch under *any type* of label noise. Putting them all together, our novel *two-phase* learning method, called **Prestopping**, realizes noise-free training for practical use. Besides, we introduce **SELFIE+**, an improved version of *SELFIE*, by adopting the maximal safe set as its clean set. Extensive experiments using four image benchmark datasets verify that our methods significantly outperform four state-of-the-art methods.

5.1 Motivation and Overview

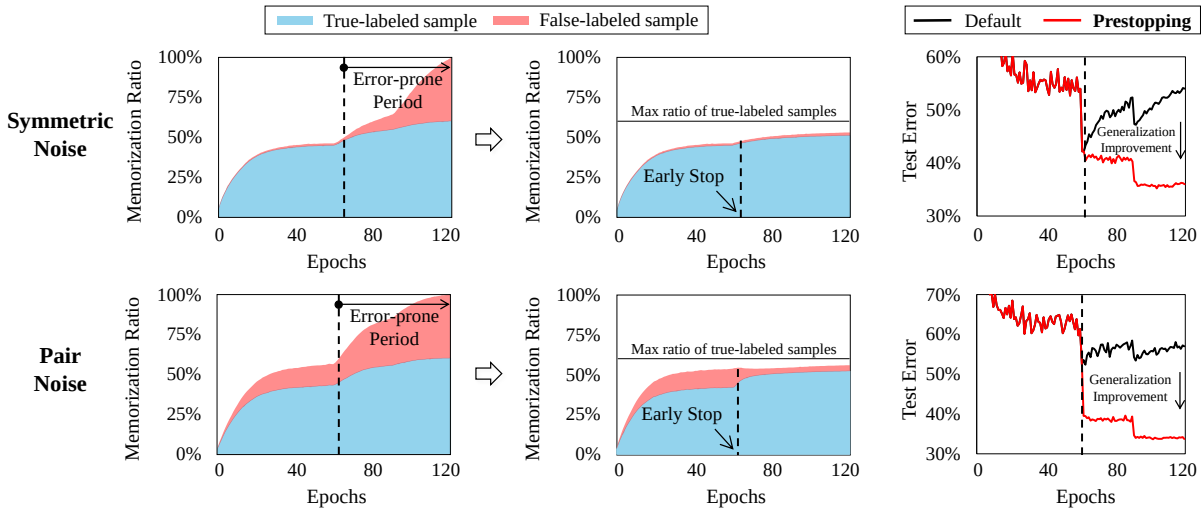
Motivation: Beyond the Small-Loss Trick

A popular approach to dealing with noisy labels is "sample selection" that filters out true-labeled samples from noisy training data [4, 19, 39, 40, 100]. Generally, $(1-\tau)\times 100\%$ of *small-loss* training samples are treated as true-labeled ones and then used to update a DNN robustly, where $\tau \in [0,1]$ is a noise rate. This *small-loss* trick is well known to be justified by the *memorization effect* [22] that DNNs tend to learn clean samples first and then gradually memorize all noisy ones. In practice, Han et al. [40] empirically proved that training on such small-loss samples yields a much better generalization performance under artificial noise scenarios.



(a) Symmetric Noise. (b) Asymmetric Noise. (c) Real-World Noise.

Figure 5.1: Loss distributions at a training accuracy of 50%: (a) and (b) show those on CIFAR-100 with two types of synthetic noises of 40%, where "symmetric noise" flips a true label into other labels with equal probability, and "asymmetric noise" flips a true label into a specific false label; (c) shows those on FOOD-101N† [2] with the real-world noise of 18.4%, where † indicates the subset in which correct labels are identified.



(a) Default. (b) *Prestopping*. (c) Test Error Convergence.

Figure 5.2: Key idea of *Prestopping*: (a) and (b) show how many true-labeled and false-labeled samples are memorized when training DenseNet (L=40, k=12)[50] on CIFAR-100 with two types of synthetic noises of 40%. "Default" is a standard training method, and "*Prestopping*" is our proposed one; (c) contrasts the convergence of test error between the two methods.

Despite its great success, we claim that the performance of the small-loss trick becomes considerably worse depending on the type of label noise. For instance, in *symmetric noise* (Figure 5.1(a)), the loss-based approach well separates true-labeled samples from false-labeled ones because many true-labeled ones exhibit smaller loss than false-labeled ones. On the other hand, in *asymmetric* and *real-world noises* (Figures 5.1(b) and 5.1(c)), many false-labeled samples are misclassified as true-labeled ones because the two distributions overlap closely; this overlap confirms that the small-loss trick still accumulates severe label noise from many misclassified cases, especially in real-world noise or asymmetric noise which is regarded as more *realistic* than symmetric noise [4, 100]. Nevertheless, supporting any type of label noise has been underexplored yet, though it is a main challenge for sample selection.

In this regard, as shown in Figure 5.2(a), we thoroughly investigated the memorization of a DNN on the noisy training samples, and, consesquently, observed the existence of *two* learning periods:

- **Noise-Robust Period:** The memorization of false-labeled samples is much less than that of true-labeled ones because the former are difficult to learn at an early stage. That is, the blue portion in Figure 5.2(a) grows much faster than the red portion before the dashed line.
- **Noise-Prone Period:** The memorization of false-labeled samples significantly increases because the network eventually begins to memorize all the noisy samples at a late stage of training. That is, the red portion in Figure 5.2(a) increases rapidly after the dashed line.

Based on these findings, we contend that differentiating the learning strategy over the aforementioned two learning periods should make a profound impact for sample selection. We propose a novel two-phase learning approach called *Prestopping*, which switches the criterion of sample selection when a DNN model enters the "noise-prone" period after the "noise-robust" period. Hence, as shown in Figure 5.2(b), the entire training process of *Prestopping* is differentiated into two learning phases by *early stopping*, thereby constructing a collection of almost true-labeled samples (we call it a *maximal safe set*). As a result, the generalization performance of a DNN improves remarkably even in *asymmetric* noise, as shown in Figure 5.2(c).

[50] The learning rate, as usual, was decayed at 50% and 75% of the total number of training epochs.

Overview: Two-Phase Learning via Prestopping

Prestopping divides the training process into two learning phases, each of which respectively corresponds to the aforementioned two learning periods:

1. **Early Stopping:** Because there is no benefit from the noise-prone period, *Prestopping* early stops training before that period begins. This early stopping effectively prevents a network from overfitting to false-labeled samples, and the samples memorized until that point are added to a *maximal safe set* because they are true-labeled with high precision.
2. **Learning from a Maximal Safe Set:** In support of noise-free training, *Prestopping* resumes training the early stopped network *only* using the maximal safe set, which is gradually increased as well as purified along with the network's learning progress.

Notably, our proposed merger of *early stopping* and *learning from the maximal safe set* indeed eliminates the noise-prone period from the training process.

In addition, we collaborate *Prestopping* with a hybrid of "loss correction" and "sample selection" approaches called *SELFIE*, which trains the network on selectively refurbished noisy samples together with small-loss samples. Here, the small-loss trick for sample selection in *SELFIE* is replaced with the maximal safe set in *Prestopping*. Note that their *synergistic* effect significantly improves the performance of *SELFIE* because the maximal safe set are true-labeled with high probability under *any* type of label noise.

5.2 Main Concept: Early Stopping and Maximal Safe Set

The key idea of *Prestopping* is learning from a maximal safe set with an early stopped network. Hence, the two phases of "early stopping" and "learning from the maximal safe set" respectively raise the questions about **(Q1)** *when is the best point to early stop the training process?* and **(Q2)** *what is the maximal safe set to enable noise-free training during the remaining period?* In this section, we first introduce the concept of network memorization and, subsequently, elaborate on the two main questions.

5.2.1 Network Memorization

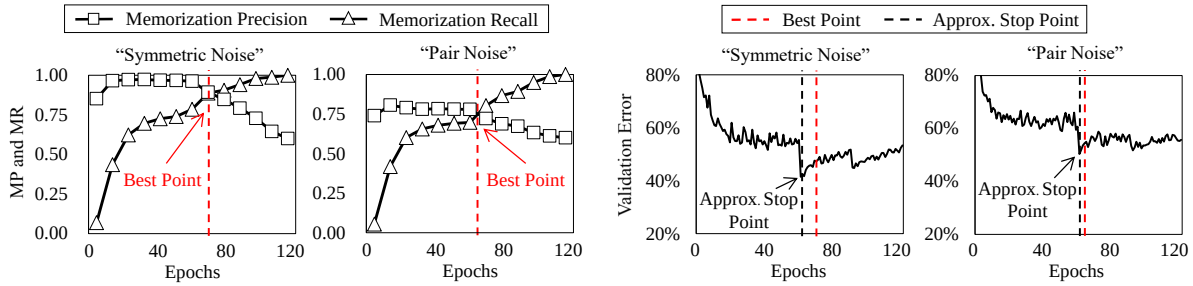
As for the notion of network memorization, a sample x is defined to be *memorized* by a network if the majority of its recent predictions at time t coincide with the given label, as in Definition 5.2.1.

Definition 5.2.1: Memorized Sample

Let $\hat{y}_t = f(x;\Theta_t)$ be the predicted label of a sample x at time t and $\mathcal{H}_x^t(q) = \{\hat{y}_{t_1}, \hat{y}_{t_2}, \ldots, \hat{y}_{t_q}\}$ be the history of the sample x that stores the predicted labels of the recent q epochs. Next, based on $\mathcal{H}_x^t(q)$, the probability of a label $y \in \{1, 2, \ldots c\}$ estimated as the label of the sample x is formulated by

$$p(y|x,t;q) = \frac{1}{|\mathcal{H}_x^t(q)|} \sum_{\hat{y} \in \mathcal{H}_x^t(q)} [\hat{y} = y], \qquad \text{where } [S] = \begin{cases} 1, & \text{if } S \text{ is true.} \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

Subsequently, the sample x with its noisy label $\tilde{y}$ is *memorized* by the network with the parameter Θ_t at time t if $\text{argmax}_y p(y|x,t;q) = \tilde{y}$ holds. □



(a) Ideal Method. (b) Validation Heuristic.

Figure 5.3: Early stop point estimated by ideal and heuristic methods when training DenseNet (L=40, k=12) on CIFAR-100 with two types of synthetic noises of 40%: (a) and (b) show the stop point derived by the ground-truth labels and the clean validation set, respectively.

5.2.2 Question 1: Best Point to Early Stop

It is desirable to stop the training process at the point when the network (1) not only accumulates *little* noise from the false-labeled samples, (2) but also acquires *sufficient* information from the true-labeled ones. Intuitively speaking, as indicated by the dashed line in Figure 5.3(a), the best stop point is the moment when *memorization precision* and *memorization recall* in Definition 5.2.2 cross with each other because it is the best trade-off between the two metrics. The period beyond this point is what we call the "noise-prone" period because MP starts decreasing rapidly, and other regions represent the "noise-robust" period.

If the ground truth y^* of a noisy label $\tilde{y}$ is known, the best stop point can be easily calculated by these metrics.

Definition 5.2.2: Memorization Metrics

Let $\mathcal{M}_t \subseteq \tilde{\mathcal{D}}$ be a set of memorized samples at time t according to Definition 5.2.1. Then, *memorization precision* and *recall* at time t are formulated by

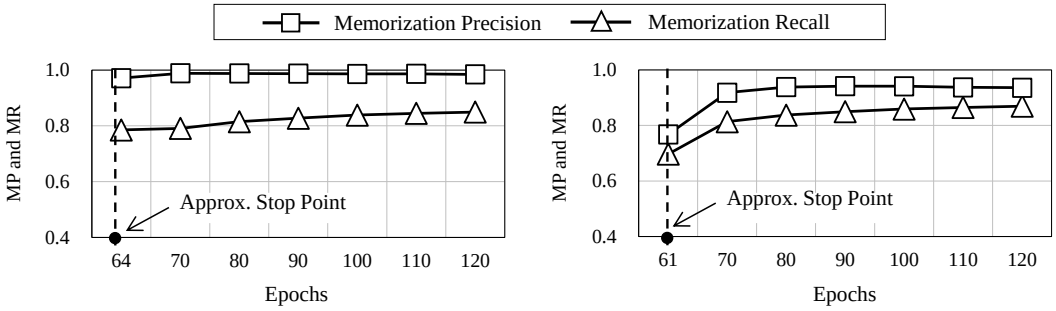
$$MP(t) = \frac{|\{(x,\tilde{y}) \in \mathcal{M}_t : \tilde{y} = y^*\}|}{|\mathcal{M}_t|} \quad \text{and} \quad MR(t) = \frac{|\{(x,\tilde{y}) \in \mathcal{M}_t : \tilde{y} = y^*\}|}{|\{(x,\tilde{y}) \in \tilde{\mathcal{D}} : \tilde{y} = y^*\}|}, \tag{5.2}$$

$$\text{where} \quad \mathcal{M}_t = \{(x,\tilde{y}) \in \tilde{\mathcal{D}} : \text{argmax}_y p(y|x,t;q) = \tilde{y}\}. \quad \square$$

However, it is not straightforward to find the exact best stop point *without* the ground-truth labels. Hence, we present two practical heuristics of approximating the best stop point with *minimal supervision*. These two heuristics require either a small clean validation set or a noise rate τ for the minimal supervision, where they are widely regarded as available in many studies [4, 19, 40, 100, 126]. We believe that other researchers will pursue this task in future work and surpass the performance of our baselines.

- **Validation Heuristic:** If a clean validation set is given, we stop training the network when the validation error is the *lowest*. It is reasonable to expect that the lowest validation error is achieved near the cross point; after that point (i.e., in the noise-prone period), the validation error likely increases because the network will be overfitted to the false-labeled samples. As shown in Figure 5.3(b), the estimated stop point is fairly close to the best stop point in both noise types.

- **Noise-Rate Heuristic:** If a noise rate τ is known, we stop training the network when the training error reaches $\tau \times 100\%$. If we assume that all true-labeled samples of $(1-\tau) \times 100\%$ are memorized



(a) Symmetric Noise. (b) Asymmetric Noise.

Figure 5.4: memorization precision and memorization recall of the maximal safe set during the remaining epochs when training DenseNet (L=40, k=12) on CIFAR-100 with two types of synthetic noises of 40%.

before any false-labeled samples of $\tau \times 100\%$ [22], this point indicates the cross point of memorization precision and memorization recall with their values to be all 1. This heuristic tends to perform worse than the validation heuristic because the assumption does not hold perfectly. We further discuss its quality in Chapter 5.4.2.

5.2.3 Question 2: Criterion of A Maximal Safe Set

Because the network is early stopped at the (estimated) best point, the set of memorized samples at that time is quantitatively sufficient and qualitatively less noisy. That is, it can be used as a safe and effective training set to resume the training of the early stopped network without accumulating the label noise. Based on this intuition, we define a *maximal safe set* in Definition 5.2.3, which is initially derived from the memorized samples at the early stop point and gradually increased as well as purified along with the network's learning progress. In each mini-batch $\mathcal{B}_t$, the network parameter Θ_t is updated using the current maximal safe set $\mathcal{S}_t$ as in Eq. (5.3), and subsequently a more refined maximal safe set $\mathcal{S}_{t+1}$ is derived by the updated network. These procedures are repeated until the end of training.

Definition 5.2.3: Maximal Safe Set

Let t_{stop} be the early stop point. A *maximal safe set* $\mathcal{S}_t$ at time t is defined to be the set of the memorized samples of the network $f(x;\Theta_t)$ when $t \geq t_{stop}$. The network $f(x;\Theta_t)$ at $t = t_{stop}$ is the early stopped network, i.e., $\mathcal{S}_{t_{stop}} = \mathcal{M}_{t_{stop}}$, and the network $f(x;\Theta_t)$ at $t > t_{stop}$ is obtained by

$$\Theta_{t+1} = \Theta_t - \eta \nabla \Big(\frac{1}{|\mathcal{B}_t'|} \sum_{x \in \mathcal{B}_t'} \mathcal{L}(f(x;\Theta_t), \tilde{y}) \Big), \quad \text{where} \quad \mathcal{B}_t' = \{x | x \in \mathcal{S}_t \cap \mathcal{B}_t\} \quad \square \tag{5.3}$$

Figure 5.4 shows the main advantage of learning from the maximal safe set during the remaining period. Even when the noise rate is quite high (e.g., 40%), this learning paradigm exploits most of true-labeled samples, in considering that the memorization recall of the maximal safe set was maintained over 0.81 in symmetric noise and over 0.84 in asymmetric noise after the 80th epoch. Further, by excluding unsafe samples that might accumulate the label noise, the memorization precision of the maximal safe set increases rapidly at the beginning of the remaining period; it was maintained over 0.99 in symmetric noise and over 0.94 even in asymmetric noise after the 80th epoch, which could *not* be realized by the small-loss trick [19, 40].

Algorithm 2 *Prestopping* with **Validation** Heuristic

Require: $\tilde{\mathcal{D}}$: data, $\mathcal{V}$: clean validation data, *epochs*: total number of epochs, q: history length

Ensure: Θ_t: network parameters

1: $t \leftarrow 1$; $\Theta_t \leftarrow$ Initialize the network parameter;
2: /* The parameter of the stopped network */
3: $\Theta_{t_{stop}} \leftarrow \emptyset$;
4: /* **Phase I: Learning from a noisy training dataset** */
5: **for** $i = 1$ **to** *epochs* **do**
6: **for** $j = 1$ **to** $|\tilde{\mathcal{D}}|/|\mathcal{B}_t|$ **do**
7: Draw a mini-batch $\mathcal{B}_t$ from $\tilde{\mathcal{D}}$;
8: /* Update by Eq. (2.2) */
9: $\Theta_{t+1} = \Theta_t - \alpha\nabla\big(\frac{1}{|\mathcal{B}_t|}\sum_{x\in\mathcal{B}_t}\mathcal{L}(f(x;\Theta_t),\tilde{y})\big)$;
10: /* A validation error at time t */
11: $val\_err \leftarrow$ Get_Validation_Error($\mathcal{V}$, Θ_t);
12: /* Save the network when val_error is the lowest */
13: **if** $isMin(val\_err)$ **then**
14: $\Theta_{t_{stop}} \leftarrow \Theta_t$;
15: $t \leftarrow t + 1$;
16: /* Load the network stopped at t_{stop} */
17: $\Theta_t \leftarrow \Theta_{t_{stop}}$;
18: /* **Phase II: Learning from a maximal safe set** */
19: **for** $i = stop\_epoch$ **to** *epochs* **do**
20: **for** $j = 1$ **to** $|\tilde{\mathcal{D}}|/|\mathcal{B}_t|$ **do**
21: Draw a mini-batch $\mathcal{B}_t$ from $\tilde{\mathcal{D}}$;
22: /* A maximal safe set in Definition 5.2.3 */
23: $\mathcal{S}_t \leftarrow \{x|\text{argmax}_y P(y|x,t;q) = \tilde{y}\}$;
24: /* Update by Eq. (5.3) */
25: $\Theta_{t+1} = \Theta_t - \alpha\nabla\big(\frac{1}{|\mathcal{S}_t\cap\mathcal{B}_t|}\sum_{x\in\mathcal{S}_t\cap\mathcal{B}_t}\mathcal{L}(f(x;\Theta_t),\tilde{y})\big)$;
26: $t \leftarrow t + 1$;
27: **return** Θ_t, $\mathcal{S}_t$;

5.3 Algorithm Description

5.3.1 Algorithm: Prestopping with Validation Heuristic

Algorithm 2 describes the overall procedure of *Prestopping* with the *validation* heuristic, which is self-explanatory. First, the network is trained on the noisy training data $\tilde{\mathcal{D}}$ in the *default* manner (Lines 5–9). During this first phase, the validation data $\mathcal{V}$ is used to evaluate the best point for the early stop, and the network parameter is saved at the time of the lowest validation error (Lines 10–14). Then, during the second phase, *Prestopping* continues to train the early stopped network for the remaining learning period (Lines 16–20). Here, the maximal safe set $\mathcal{S}_t$ at the current moment is retrieved, and each sample $x \in \mathcal{S}_t \cap \mathcal{B}_t$ is used to update the network parameter. The mini-batch samples not included in $\mathcal{S}_t$ are no longer used in pursuit of robust learning (Lines 21–25).

Algorithm 3 *Prestopping* with **Validation** Heuristic

Require: $\tilde{\mathcal{D}}$: data, $\mathcal{V}$: clean validation data, *epochs*: total number of epochs, q: history length
Ensure: Θ_t: network parameters

1: $t \leftarrow 1$; $\Theta_t \leftarrow$ Initialize the network parameter;
2: /* The parameter of the stopped network */
3: $\Theta_{t_{stop}} \leftarrow \emptyset$;
4: /* **Phase I: Learning from a noisy training dataset** */
5: **for** $i = 1$ **to** *epochs* **do**
6: **for** $j = 1$ **to** $|\tilde{\mathcal{D}}|/|\mathcal{B}_t|$ **do**
7: Draw a mini-batch $\mathcal{B}_t$ from $\tilde{\mathcal{D}}$;
8: /* Update by Eq. (2.2) */
9: $\Theta_{t+1} = \Theta_t - \eta\nabla\big(\frac{1}{|\mathcal{B}_t|}\sum_{x\in\mathcal{B}_t}\mathcal{L}(f(x;\Theta_t),\tilde{y})\big)$;
10: /* A training error at time t */
11: $train\_err \leftarrow$ Get_Training_Error($\tilde{\mathcal{D}}$, Θ_t)
12: /* Save the network when *val_error* is the lowest */;
13: **if** $train\_err \leq \tau$ **then**
14: $\Theta_{t_{stop}} \leftarrow \Theta_t$;
15: break;
16: $t \leftarrow t + 1$;
17: /* Load the network stopped at t_{stop} */
18: $\Theta_t \leftarrow \Theta_{t_{stop}}$;
19: /* **Phase II: Learning from a maximal safe set** */
20: **for** $i = stop\_epoch$ **to** *epochs* **do**
21: **for** $j = 1$ **to** $|\tilde{\mathcal{D}}|/|\mathcal{B}_t|$ **do**
22: Draw a mini-batch $\mathcal{B}_t$ from $\tilde{\mathcal{D}}$;
23: /* A maximal safe set in Definition 5.2.3 */
24: $\mathcal{S}_t \leftarrow \{x|\text{argmax}_y P(y|x,t;q) = \tilde{y}\}$;
25: /* Update by Eq. (5.3) */
26: $\Theta_{t+1} = \Theta_t - \eta\nabla\big(\frac{1}{|\mathcal{S}_t\cap\mathcal{B}_t|}\sum_{x\in\mathcal{S}_t\cap\mathcal{B}_t}\mathcal{L}(f(x;\Theta_t),\tilde{y})\big)$;
27: $t \leftarrow t + 1$;
28: **return** Θ_t, $\mathcal{S}_t$;

5.3.2 Algorithm: Prestopping with Noise-Rate Heuristic

Algorithm 3 describes the overall procedure of *Prestopping* with the *noise-rate* heuristic, which is also self-explanatory. Compared with Algorithm 2, only the way of determining the best stop point in Lines 11–15 has changed.

5.3.3 Collaboration with Sample Refurbishment: SELFIE+

To further improve the performance of *Prestopping*, we introduce *SELFIE+* that employs the concept of selectively refurbishing false-labeled samples in *SELFIE* [19]. In detail, the final maximal safe set $\mathcal{S}_{t_{end}}$ is retrieved from the first run of *Prestopping*, and then it is used as the clean set for the next run of *SELFIE* with initializing the network parameter. Following the update principle of *SELFIE*, the

network is trained based on the modified gradient update rule,

$$\Theta_{t+1} = \Theta_t - \eta \nabla \Big(\frac{1}{|\{x | x \in \mathcal{R}_t \cup \mathcal{S}_{t_{end}}\}|} \big(\sum_{x \in \mathcal{R}_t \cup \mathcal{S}^c_{t_{end}}} \mathcal{L}(f(x;\Theta_t), y^{refurb}) + \sum_{x \in \mathcal{S}_{t_{end}}} \mathcal{L}(f(x;\Theta_t), \tilde{y}) \big) \Big). \quad (5.4)$$

For each sample x in the mini-batch $\mathcal{B}_t$, the given label $\tilde{y}$ of x is selectively refurbished into y^{refurb} if it can be corrected with high precision. However, the mini-batch samples included in $\mathcal{S}_{t_{end}}$ are omitted from the refurbishment because their label is already highly credible. Then, the mini-batch samples in the refurbished set $\mathcal{R}_t$ are provided to update the network together with those in the final maximal safe set $\mathcal{S}_{t_{end}}$. Please see Chapter 4 for details on *SELFIE*.

5.4 Main Experiments

Datasets: To verify the superiority of *Prestopping* and *SELFIE+*, we performed an image classification task on *four* benchmark datasets: CIFAR-10 (10 classes), CIFAR-100 (100 classes), ANIMAL-10N (10 classes), FOOD-101N (101 classes)[51]. We did not apply any data augmentation.

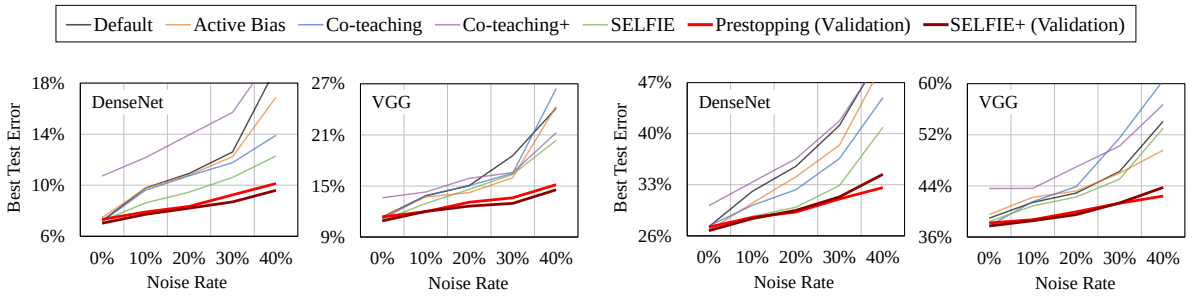
Noise Injection: As all labels in the CIFAR datasets are clean, we artificially corrupted the labels in these datasets using typical methods for the evaluation of synthetic noises [4, 40, 100]: *(i) symmetric noise* and *(ii) asymmetric noise*. For the asymmetric noise, the corrupted label was set to be the next label of the true label following the recent work [4, 19]. To evaluate the robustness on varying noise rates from light noise to heavy noise, we tested five noise rates $\tau \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$. In contrast, we did not inject any label noise into ANIMAL-10N and FOOD-101N because they contain real label noise estimated at 8.0% and 18.4% respectively [2, 19].

Clean Validation Data: Recall that a clean validation set is needed for the validation heuristic in Chapter 5.2.2. As for ANIMAL-10N and Food-101N, we did exploit their own clean validation data; $5,000$ and $3,824$ images, respectively, were included in their validation set. However, as no validation data exists for the CIFAR datasets, we constructed a small clean validation set by randomly selecting $1,000$ images from the *original* training data of $50,000$ images. Please note that the noise injection process was applied to only the rest $49,000$ training images.

Networks and Hyperparameters: For the classification task, we trained DenseNet (L=40, k=12) and VGG-19 with a momentum optimizer. Specifically, we used a momentum of 0.9, a batch size of 128, a dropout of 0.1, and batch normalization. *Prestopping* has only one unique hyperparameter, the history length q, and it was set to be 10, which was the best value found by the grid search (please see Chapter 5.4.4 for details). The hyperparameters used in the compared algorithms were favorably set to be the best values presented in the original papers. As for the training schedule, we trained the network for 120 epochs and used an initial learning rate 0.1, which was divided by 5 at 50% and 75% of the total number of epochs.

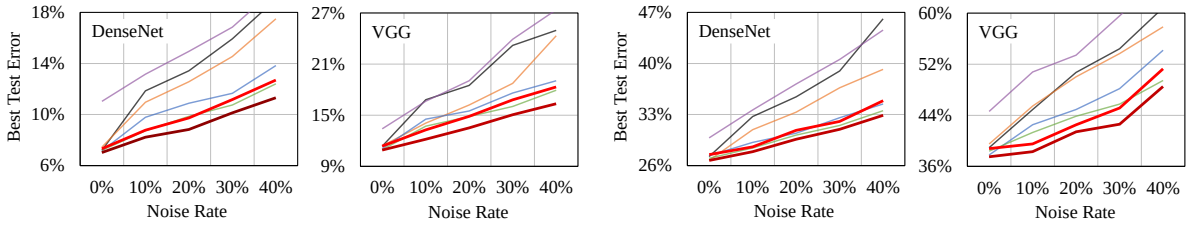
Algorithms: We compared *Prestopping* and *SELFIE+* with not only a baseline algorithm but also the *four* state-of-the-art robust training algorithms: *Default* trains the network without any processing for the label noise; *Active Bias* re-weights the backward loss of training samples based on their prediction variance; *Co-teaching* selects a certain number of small-loss samples to train the network based on the co-training mechanism [125]; *Co-teaching+* is similar to *Co-teaching*, but its small-loss samples are selected from the disagreement set; *SELFIE* selectively refurbishes noisy samples and exploits them together with

[51] In FOOD-101N, we used a subset of the entire training data marked with whether the label is correct or not.



(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 5.5: Best test errors using two CNNs on two datasets with varying **asymmetric** noise rates.



(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 5.6: Best test errors using two CNNs on two datasets with varying **symmetric** noise rates.

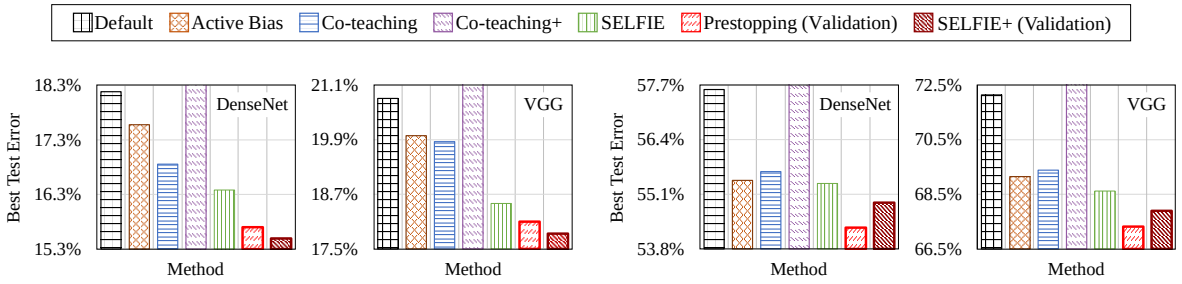
the small-loss samples. All the algorithms were implemented using TensorFlow 1.8.0 and executed using 16 NVIDIA Titan Volta GPUs. For reproducibility, we provide the source code at https://bit.ly/2l3g9Jx. In support of reliable evaluation, we repeated every task *thrice* and reported the average and standard error of the best test errors, which are the common measures of robustness to noisy labels [19, 34, 39, 100].

5.4.1 Performance Comparison (Validation Heuristic)

Figures 5.5 and 5.6 show the test error of the seven training methods using two CNNs on two datasets with varying *asymmetric* and *symmetric* noise rates, respectively. The results of *Prestopping* and *SELFIE+* were obtained using the *validation* heuristic. See Chapter 5.4.2 for the results of the *noise-rate* heuristic. In order to highlight the improvement of *Prestopping* and *SELFIE+* over the other methods, their lines are dark colored. Figure 5.7 shows the test error on two *real-world* noisy datasets with different noise rates. The best test errors with synthetic and real-world noises are summarized in Tables 5.1 and 5.2. In addition, more experimental results on Tiny-ImageNet and Clothing datasets are discussed in Chapter 5.4.3.

Result with Asymmetric Noise

The performance trend in the two network architectures were similar with each other. In general, either *Prestopping* or *SELFIE+* achieved the lowest test error in a wide range of noise rates on both CIFAR datasets. With help of the refurbished samples, *SELFIE+* achieved a slightly better performance than *Prestopping* in CIFAR-10. However, an opposite trend was observed in CIFAR-100; this phenomenon was due to a large number of falsely corrected labels under the asymmetric noise, especially when the number of classes is large (see Chapter 5.5.3 for details). Although *SELFIE* achieved relatively lower test error among the existing methods, the test error of *SELFIE* was still worse than that of *Prestopping*. *Co-teaching* did not work well because many false-labeled samples were misclassified as



(a) ANIMAL-10N ($\tau \approx 8.0\%$). (b) FOOD-101N ($\tau \approx 18.4\%$).

Figure 5.7: Best test errors using two CNNs on two datasets with **real-world** noises.

clean ones; *Co-teaching+* was shown to be even worse than *Co-teaching* despite it being an improvement of *Co-teaching* (see Chapter 5.5.2 for details). The test error of *Active Bias* was not comparable to that of *Prestopping*. The performance improvement of ours over the others increased as the label noise became heavier. In particular, at a heavy noise rate of 40%, *Prestopping* or *SELFIE+* significantly reduced the *absolute* test error by 2.2*pp*–18.1*pp* compared with the other robust methods.

Result with Symmetric Noise

Similar to the asymmetric noise, both *Prestopping* and *SELFIE+* generally outperformed the other methods. In particular, the performance of *SELFIE+* was the best at any noise rate on all datasets, because the synergistic effect was higher in symmetric noise than in asymmetric noise. Quantitatively, at a heavy noise rate of 40%, our methods showed significant reduction in the *absolute* test error by 0.3*pp*–17.5*pp* compared with the other robust methods. Unlike the asymmetric noise, *Co-teaching* and *SELFIE* achieved a low test error comparable to *Prestopping* because true-labeled samples could be well separated from false-labeled ones by their small-loss criteria; hence, the dominance between *Co-teaching* and *Active Bias* was reversed so that the former slightly outperformed the latter.

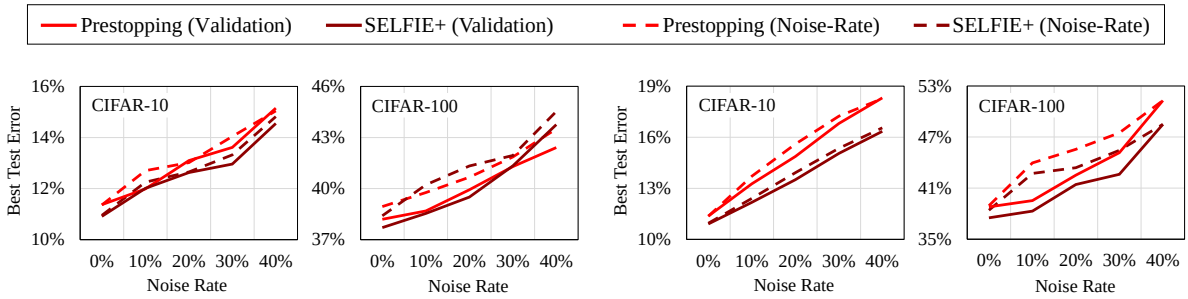
Result with Real-world Noise

Both *Prestopping* and *SELFIE+* maintained their dominance over the other methods under *real-world* label noise as well. *SELFIE+* achieved the lowest test error when the number of classes is small (e.g., ANIMAL-10N), while *Prestopping* was the best when the number of classes is large (e.g., FOOD-101N) owing to the difficulty in label correction of *SELFIE+*. (Thus, practitioners are recommended to choose between *Prestopping* and *SELFIE+* depending on the number of classes in hand.) Specifically, they improved the *absolute* test error by 0.4*pp*–4.6*pp* and 0.5*pp*–8.2*pp* in ANIMAL-10N and FOOD-101N, respectively. Therefore, we believe that the advantage of our methods is unquestionable even in the real-world scenarios.

5.4.2 Performance Comparison (Noise-Rate Heuristic)

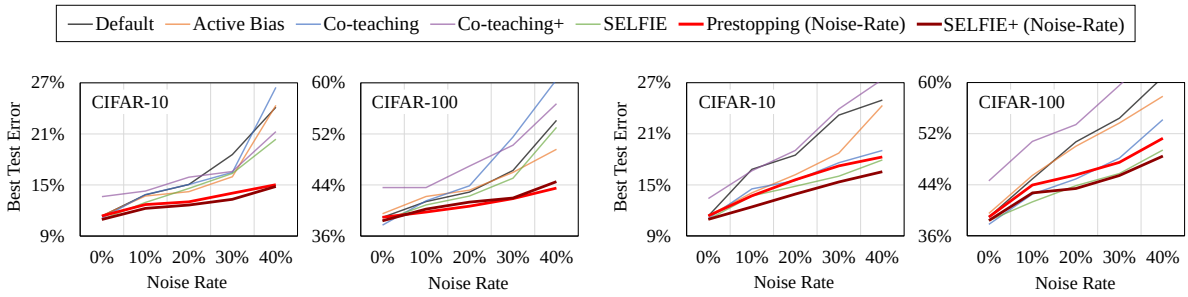
Result with Synthetic Noise

To verify the performance of *Prestopping* and *SELFIE+* with the *noise-rate* heuristic in Chapter 5.2.2, we trained a VGG-19 network on two simulated noisy datasets with the same configuration as in Chapter 5.4. Figure 5.9 shows the test error of our two methods using the noise-rate heuristic as well as those of the other five training methods. Again, the test error of either *Prestopping* or *SELFIE+*



(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 5.8: Best test errors using two CNNs on two datasets with varying symmetric noise rates.



(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 5.9: Best test errors using VGG-19 on two simulated noisy datasets with varying noise rates.

was the lowest at most error rates with any noise type. The trend of the noise-rate heuristic here was almost the same as that of the validation heuristic in Chapter 5.4.1. Especially when the noise rate was 40%, *Prestopping* and *SELFIE+* significantly improved the test error by 5.1*pp*–17.0*pp* in the asymmetric noise (Figure 5.9(a)) and 0.3*pp*–17.3*pp* in the symmetric noise (Figure 5.9(b)) compared with the other robust methods.

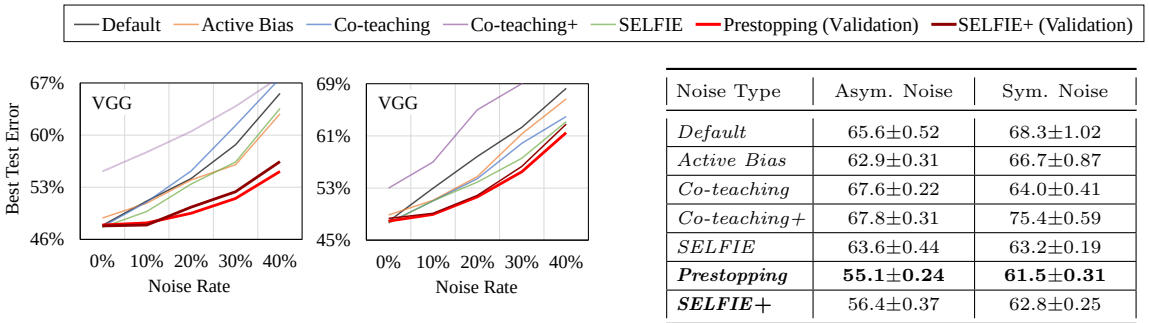
Comparison with Validation Heuristic

Figure 5.8 shows the difference in test error caused by the two heuristics. Overall, the performance with the noise-rate heuristic was worse than that with validation heuristic, even though the worse one outperformed the other training methods as shown in Figure 5.9. As the assumption of the noise-rate heuristic does not hold perfectly, a lower performance of the noise-rate heuristic looks reasonable. However, we expect that the performance with this heuristic can be improved by stopping a little earlier than the estimated point, and we leave this extension as the future work.

5.4.3 Two Challenging Datasets with Validation Heuristic

For a larger-scale experiment, we repeated the image classification task on Tiny-ImageNet (200 classes). Because no test set exists, randomly selected 9,000 images from the validation set were used as the test set, and the rest 1,000 validation images were used as the clean validation set for the validation heuristic. The experimental configurations were the same as those in Chapter 5.4.

Figure 5.10 shows the test error of the seven training methods using VGG-19 on Tiny-ImageNet. Similar to CIFAR datasets in Figures 5.5 and 5.6, both *Prestopping* and *SELFIE+* outperformed the other robust methods in both noise types. The only difference was that there was no synergistic effect

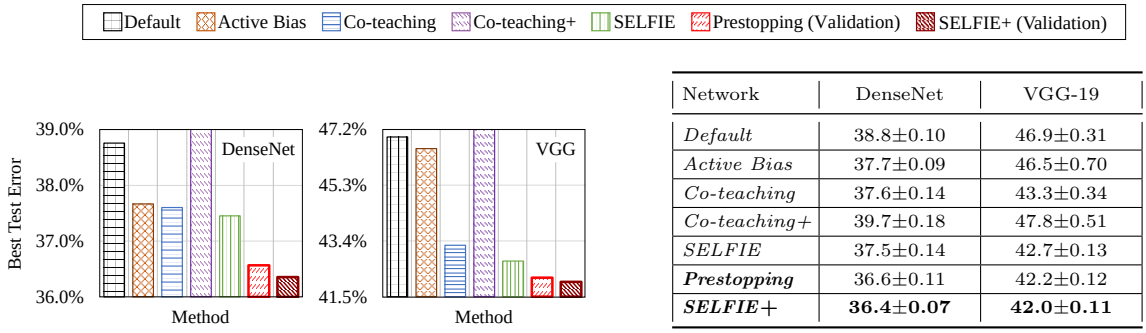


(a) Pair Noise. (b) Symmetric Noise.

Noise Type	Asym. Noise	Sym. Noise
Default	65.6±0.52	68.3±1.02
Active Bias	62.9±0.31	66.7±0.87
Co-teaching	67.6±0.22	64.0±0.41
Co-teaching+	67.8±0.31	75.4±0.59
SELFIE	63.6±0.44	63.2±0.19
Prestopping	**55.1±0.24**	**61.5±0.31**
SELFIE+	56.4±0.37	62.8±0.25

(c) Detailed Results at Noise Rate 40%.

Figure 5.10: Best test errors using VGG-19 on Tiny-ImageNet with varying noise rates.



(a) DenseNet. (b) VGG-19.

Network	DenseNet	VGG-19
Default	38.8±0.10	46.9±0.31
Active Bias	37.7±0.09	46.5±0.70
Co-teaching	37.6±0.14	43.3±0.34
Co-teaching+	39.7±0.18	47.8±0.51
SELFIE	37.5±0.14	42.7±0.13
Prestopping	36.6±0.11	42.2±0.12
SELFIE+	**36.4±0.07**	**42.0±0.11**

(c) Detailed Results.

Figure 5.11: Best test errors on Clothing70k ($\tau \approx 38.5\%$) along with the detailed result.

from collaboration with sample refurbishment because of the larger number of classes (i.e., 200). In particular, compared with other robust methods, *Prestopping* showed significant reduction in the *absolute* test error by 7.8*pp*–12.7*pp* at pair noise of 40% and 1.7*pp*–13.9*pp* at symmetric noise of 40%

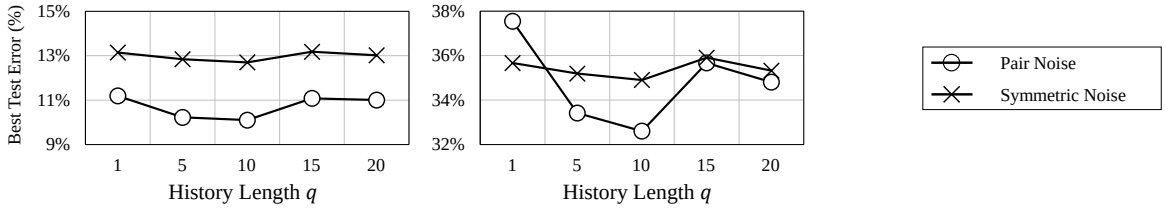
We additionally evaluated the superiority of *Prestopping* and *SELFIE+* on another challenging real-world noisy dataset Clothing70k (14 classes), a subset of Clothing1M [20], where its noise rate was estimated at 38.5%. For the Clothing70k dataset, we randomly selected 70,000 images from 1 million noisy training images in Clothing1M and used them as its noisy training set; we did exploit the original clean validation and test sets consisting of 14,313 and 10,526 images, respectively. The experimental configurations were the same as those in Chapter 5.4.

Figure 5.11 shows the test error of the seven training methods using two CNNs on Clothing70k with real-world noise of 38.5%. Overall, *Prestopping* and *SELFIE+* significantly outperformed the other robust methods. In particular, *SELFIE+* achieved the lowest test error because the number of classes was not that large (i.e., 14 classes). Compared with other state-of-the-art methods, it improved the *absolute* test error by 1.1*pp*–3.3*pp* using DenseNet and 0.7*pp*–5.8*pp* using VGG-19.

5.4.4 Hyperparameter Selection

Prestopping requires one additional hyperparameter, the history length q in Definition 5.2.1. For hyperparameter tuning, we trained DenseNet (L=40, k=12) on CIFAR-10 and CIFAR-100 with a noise rate of 40%, and the history length q was chosen in a grid $\in \{1, 5, 10, 15, 20\}$. Figure 5.12 shows the test

error of *Prestopping* obtained by the grid search on two noisy CIFAR datasets. Regardless of the noise type, the lowest test error was typically achieved when the value of q was 10 in both datasets. Therefore, the history length q was set to be 10 in all experiments.



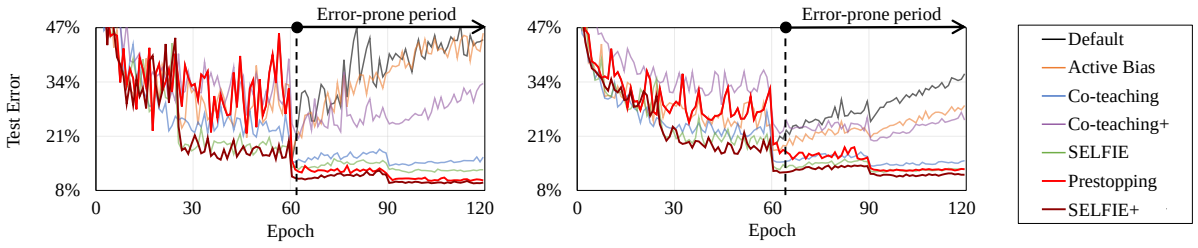
(a) CIFAR-10. (b) CIFAR-100.

Figure 5.12: Grid search on CIFAR-10 and CIFAR-100 with two types of noises of 40%.

5.5 Ablation Study

5.5.1 Convergence Analysis

To verify that the noise-prone period is eliminated, we plot the convergence of test error for the seven training methods. The test error of *Default*, *Active Bias*, and *Co-teaching+* first reached their lowest values before the noise-prone period, and then increased rapidly in the noise-prone period. On the other hand, the remaining methods kept reducing the test error even in that period. Notably, *SELFIE+* achieved the lowest test error at the end, owing to the successful merger of the advanced sample selection in *Prestopping* and the sample refurbishment in *SELFIE*. We observed that the test error of *SELFIE+* was much lower than those of other methods at an early stage of training (i.e., 25–60th epochs) because refurbishing false-labeled samples expedited the convergence of test error. In addition, the performance of *Prestopping* was shown to be almost the same as and comparable to that of *SELFIE+* in asymmetric and symmetric noises, respectively. The test error of *SELFIE* also converged faster than those of the other existing methods, but it was still inferior to *SELFIE+*. Thus, these results confirm that our two methods effectively overcome the noise-prone period.



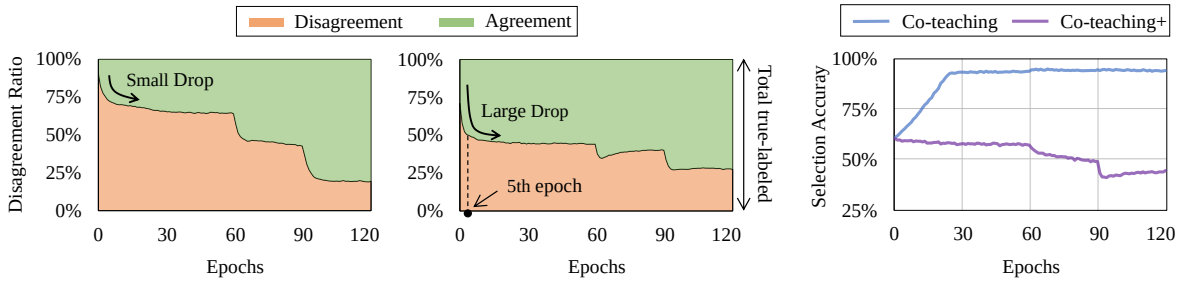
(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 5.13: Convergence curves of DenseNet (L=40, k=12) on CIFAR-10 with two types of synthetic noises of 40%.

5.5.2 Anatomy of Co-teaching+

Although *Co-teaching+* is the latest method, its performance was worse than expected, as shown in Chapter 5.4.1. Thus, we looked into *Co-teaching+* in more detail. A poor performance of *Co-teaching+* was attributed to the fast consensus of the label predictions for true-labeled samples, especially when

training a complex network. In other words, because two complex networks in *Co-teaching+* start making the same predictions for true-labeled samples too early, these samples are excluded too early. As shown in Figures 5.14(a) and 5.14(b), the disagreement ratio with regard to the true-labeled samples dropped faster with a complex network than with a simple network, in considering that the ratio drastically decreased to 49.8% during the first 5 epochs. Accordingly, it is evident that *Co-teaching+* with a complex network causes a *narrow exploration* of the true-labeled samples because *Co-teaching+* selects their small-loss samples from the disagreement set. and the selection accuracy of *Co-teaching+* naturally degraded from 60.4% to 44.8% for that reason, as shown in Figure 5.14(c). Therefore, we conclude that *Co-teaching+* may not suit a complex network.

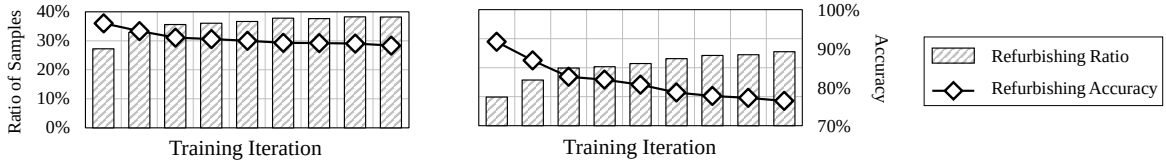


(a) Simple Network. (b) Complex Network. (c) Selection Accuracy.

Figure 5.14: Anatomy of *Co-teaching+* on CIFAR-100 with 40% symmetric noise: (a) and (b) show the change in disagreement ratio for all true-labeled samples, when using *Co-teaching+* to train two networks with different complexity, where "simple network" is a network with seven layers used by Yu et al. [4], and "complex network" is a DenseNet (L=40, k=12) used for our evaluation; (c) shows the accuracy of selecting true-labeled samples on the DenseNet.

5.5.3 Impact of Number of Classes on SELFIE+

We investigate the impact of the number of classes on *SELFIE+* under the asymmetric noise of 40%. Figure 5.15 shows the ratio of refurbished samples used for training (hatched bar) with the refurbishing accuracy (solid line) on two datasets with 10 and 100 classes. When the number of classes is small (e.g., 10), *SELFIE+* refurbished most of false-labeled samples very accurately; the refurbishing accuracy was consistently over 91.6% during the entire training period. On the other hand, when the number of classes is large (e.g., 100), the refurbishing accuracy drastically dropped from 88.3% to 68.8% as more samples were refurbished. Thus, as shown in Figure 5.15(b), collaboration with sample refurbishment is not always synergistic because such a significant drop in the refurbishing accuracy could induce many falsely corrected samples, especially when there are many classes.



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

Figure 5.15: Refurbishing ratio and accuracy of *SELFIE+* when training DenseNet (L=40, k=12) on CIFAR-10 and CIFAR-100 with asymmetric noise of 40%.

5.6 A Case Study: Noisy Labels in CIFAR-100

One interesting observation is a noticeable improvement of *SELFIE+* even when the noise rate was 0%, as shown in Table 5.1(a). It was turned out that *SELFIE+* sometimes refurbished the labels of the false-labeled samples which were *originally* contained in the CIFAR datasets. Figure 5.16 shows a few successful refurbishment cases. For example, an image falsely annotated as a "Boy" was refurbished as a "Baby" (Figure 5.16(a)), and an image falsely annotated as a "Mouse" was refurbished as a "Hamster" (Figure 5.16(c)). Thus, this sophisticated label correction of *SELFIE+* helps overcome the residual label noise in well-known benchmark datasets, which are misconceived to be clean, and ultimately further improves the generalization performance of a network.

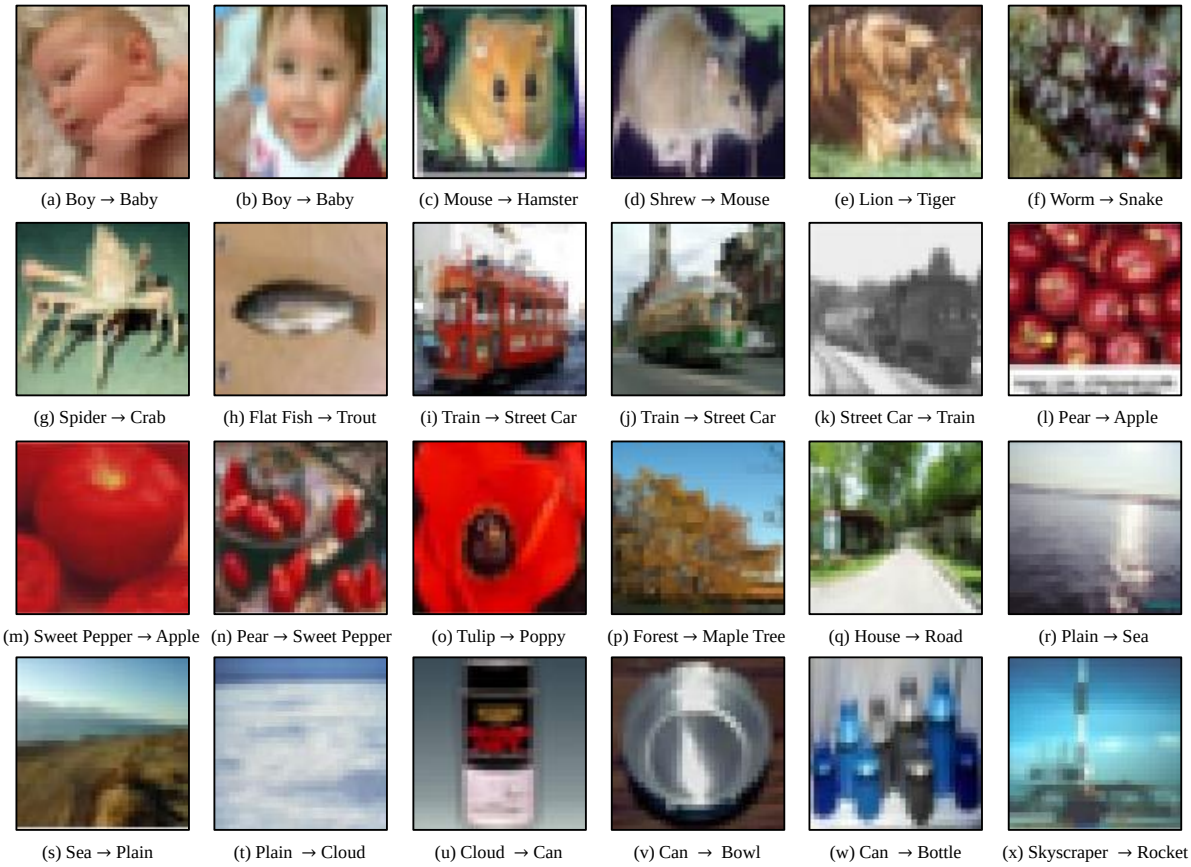


Figure 5.16: Refurbishing of false-labeled samples *originally* contained in CIFAR-100. The subcaption represents "original label" → "refurbished label" recognized by *SELFIE+*.

5.7 Complete Results on Best Test Error

Table 5.1 shows the test error of seven training methods using two CNNs on two *simulated* noisy data sets with varying noise rates. Table 5.2 shows the test error of seven training methods using two CNNs on two *real-world* noisy data sets with different noise rates.

Table 5.1: The best test errors (%) of seven training methods on two types of **synthetic** noises with varying noise rates (0%, 10%, 20%, 30%, and 40%) in Figures 5.5 and 5.6.

Noise Type	Asymmetric Noise in Figure 5.5				Symmetric Noise in Figure 5.6			
dataset	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Network	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG
Default	7.20±0.05	11.4±0.14	27.3±0.35	38.9±0.07	7.20±0.05	11.4±0.14	27.3±0.35	38.9±0.07
Active Bias	7.45±0.36	11.4±0.08	26.7±0.48	39.5±0.09	7.45±0.36	11.4±0.08	26.7±0.48	39.5±0.09
Co-teaching	7.16±0.01	11.1±0.14	27.4±0.87	37.7±0.04	7.16±0.01	11.1±0.14	27.4±0.87	37.7±0.04
Co-teaching+	10.7±0.27	13.6±0.28	30.1±0.24	43.6±0.09	10.7±0.27	13.6±0.28	30.1±0.24	43.6±0.09
SELFIE	7.20±0.06	11.0±0.07	27.0±0.41	38.4±0.12	7.20±0.06	11.0±0.07	27.0±0.41	38.4±0.12
Prestopping	7.32±0.06	11.4±0.15	27.2±0.11	38.2±0.17	7.32±0.06	11.4±0.15	27.2±0.11	38.2±0.17
SELFIE+	**7.02±0.11**	**10.9±0.10**	**26.7±0.18**	**37.7±0.16**	**7.02±0.11**	**10.9±0.10**	**26.7±0.18**	**37.5±0.16**

(a) The best test errors under asymmetric and symmetric noises of 0% ($\tau = 0.0$).

Noise Type	Asymmetric Noise in Figure 5.5				Symmetric Noise in Figure 5.6			
dataset	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Network	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG
Default	9.79±0.02	13.8±0.62	32.1±0.19	41.4±0.15	11.9±0.74	16.8±0.05	32.7±0.66	44.9±0.24
Active Bias	9.77±0.27	13.7±0.14	30.5±0.63	42.2±0.22	11.0±0.44	14.1±0.36	30.9±0.19	45.5±0.21
Co-teaching	9.62±0.09	13.9±0.13	30.2±0.38	41.5±0.20	9.79±0.18	14.5±0.12	29.2±0.15	42.5±0.30
Co-teaching+	12.2±0.14	14.3±0.07	33.4±0.14	43.6±0.52	13.2±0.34	16.7±0.09	33.6±0.41	50.8±0.24
SELFIE	8.62±0.36	13.0±0.07	28.7±0.20	40.9±0.14	8.73±0.12	13.7±0.23	28.4±0.09	41.3±0.32
Prestopping	7.91±0.19	**12.0±0.14**	28.5±0.14	38.7±0.20	8.77±0.44	13.3±0.03	28.6±0.84	39.5±0.21
SELFIE+	**7.72±0.01**	**12.0±0.10**	**28.3±0.12**	**38.5±0.24**	**8.23±0.25**	**12.2±0.13**	**27.9±0.03**	**38.3±0.16**

(b) The best test errors under asymmetric and symmetric noises of 10% ($\tau = 0.1$).

Noise Type	Asymmetric Noise in Figure 5.5				Symmetric Noise in Figure 5.6			
dataset	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Network	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG
Default	10.9±0.44	15.1±1.11	35.5±0.41	42.9±0.21	13.4±0.77	18.5±0.31	35.4±0.79	50.7±0.22
Active Bias	10.8±0.17	14.2±0.13	34.1±0.69	43.2±0.09	12.6±0.27	16.2±0.20	33.3±0.25	50.0±0.25
Co-teaching	10.7±0.13	15.1±0.16	32.3±0.95	43.9±0.31	10.9±0.27	15.5±0.08	30.4±0.20	44.9±0.17
Co-teaching+	13.9±0.38	15.9±0.31	36.6±0.34	47.0±0.34	15.0±0.34	19.0±0.11	37.2±0.18	53.4±0.14
SELFIE	9.46±0.10	14.6±0.06	29.9±0.44	42.3±0.17	9.86±0.25	14.8±0.25	30.2±0.37	43.9±0.09
Prestopping	8.34±0.18	13.1±0.08	**29.3±0.24**	39.9±0.42	9.73±0.28	14.9±0.23	30.8±0.70	42.5±0.20
SELFIE+	**8.20±0.17**	**12.6±0.03**	29.5±0.20	**39.5±0.25**	**8.83±0.30**	**13.5±0.05**	**29.6±0.40**	**41.6±0.08**

(c) The best test errors under asymmetric and symmetric noises of 20% ($\tau = 0.2$).

Noise Type	Asymmetric Noise in Figure 5.5				Symmetric Noise in Figure 5.6			
dataset	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Network	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG
Default	12.6±1.98	18.6±1.11	41.1±0.43	46.3±0.25	15.9±1.25	23.2±0.41	39.0±0.78	54.4±0.25
Active Bias	12.2±0.80	15.6±0.12	38.4±0.91	46.0±0.16	14.6±0.48	18.7±0.31	36.7±0.66	53.7±0.12
Co-teaching	11.8±0.54	16.5±0.11	36.6±1.08	51.5±0.36	11.7±0.35	17.6±0.02	32.6±0.33	48.2±0.34
Co-teaching+	15.7±0.19	16.6±0.12	41.8±0.79	50.2±0.21	16.8±0.12	24.0±0.17	40.6±0.24	59.6±0.23
SELFIE	10.6±0.34	16.4±0.17	32.9±0.13	45.1±0.16	10.8±0.09	16.0±0.24	31.5±0.33	45.8±0.60
Prestopping	9.24±0.24	13.6±0.14	**31.1±0.73**	**41.3±0.14**	11.2±0.97	16.8±0.23	32.1±0.94	45.2±0.31
SELFIE+	**8.67±0.29**	**13.0±0.09**	31.3±0.19	**41.3±0.05**	**10.1±0.70**	**15.1±0.05**	**31.0±0.88**	**42.6±0.06**

(d) The best test errors under asymmetric and symmetric noises of 30% ($\tau = 0.3$).

Noise Type	Asymmetric Noise in Figure 5.5				Symmetric Noise in Figure 5.6			
dataset	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Network	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG	DenseNet	VGG
Default	19.2±0.63	24.1±0.97	51.2±0.40	54.1±0.82	19.0±1.66	25.0±0.19	46.1±0.72	60.9±0.08
Active Bias	16.9±0.93	24.4±1.24	49.6±0.36	49.6±0.60	17.5±0.90	24.3±0.36	39.2±0.84	57.9±0.20
Co-teaching	13.9±1.11	26.5±2.47	45.0±1.06	60.5±0.69	13.8±0.41	19.1±0.07	34.5±0.13	54.2±0.45
Co-teaching+	20.6±0.74	21.3±0.20	50.7±0.65	56.7±0.16	19.9±0.28	27.4±0.24	44.6±1.04	66.0±0.31
SELFIE	12.3±0.80	20.4±0.24	40.9±1.00	53.0±0.45	12.4±0.07	18.0±0.14	33.5±0.80	49.5±0.07
Prestopping	10.1±0.20	15.2±0.54	**32.6±0.40**	**42.4±0.28**	12.7±0.33	18.3±0.07	34.9±0.63	51.3±0.44
SELFIE+	**9.60±0.13**	**14.6±0.17**	34.4±0.10	43.8±0.14	**11.3±0.03**	**16.4±0.08**	**32.9±0.20**	**48.5±0.05**

(e) The best test errors under asymmetric and symmetric noises of 40% ($\tau = 0.4$).

Table 5.2: The best test errors (%) on **real-world** noises in Figure 5.7.

dataset	ANIMAL-10N		FOOD-101N	
Network	DenseNet	VGG	DenseNet	VGG
Default	18.2±0.15	20.8±0.36	57.6±0.20	72.1±0.45
Active Bias	17.6±0.13	20.0±0.30	55.4±0.10	69.2±0.84
Co-teaching	16.9±0.14	19.9±0.01	55.6±0.09	69.4±0.27
Co-teaching+	19.8±0.11	22.4±0.61	58.6±0.24	75.5±0.53
SELFIE	16.4±0.21	18.5±0.14	55.4±0.17	68.6±0.24
Prestopping	15.7±0.18	18.1±0.14	**54.3±0.24**	**67.3±0.21**
SELFIE+	**15.5±0.12**	**17.8±0.07**	54.9±0.17	67.9±0.09

5.8 Chapter Conclusions

Conclusion

In this chapter, we proposed a novel two-phase training strategy for the noisy training data, which we call *Prestopping*. The first phase, "early stopping," retrieves an initial set of true-labeled samples as many as possible, and the second phase, "learning from a maximal safe set," completes the rest training process only using the true-labeled samples with high precision. *Prestopping* can be easily applied to many real-world cases because it additionally requires only either a small clean validation set or a noise rate. Furthermore, we combined this novel strategy with sample refurbishment to develop *Prestopping*. Through extensive experiments using various real-world and simulated noisy datasets, we verified that either *Prestopping* or *SELFIE+* achieved the lowest test error among the seven compared methods, thus significantly improving the robustness to diverse types of label noise. Overall, we believe that our work of dividing the training process into two phases by early stopping is a new direction for robust training and can trigger a lot of subsequent studies.

Chapter 6. A Self-Transitional Learning Approach: MORPH

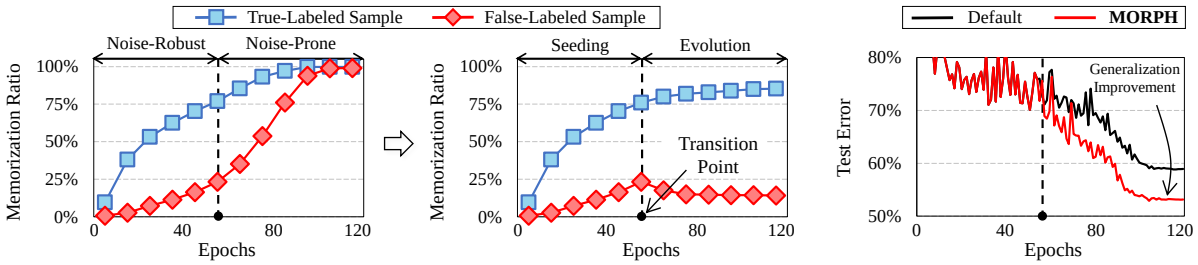
Summary: Chapter based on work being under review at NeurIPS 2020 [51]

The main limitation of *Prestopping* is the requirement of supervision, such as a clean validation set or a known noise rate, to evaluate the best early stopping point for its phase transition. However, they are hard to acquire in practice because they essentially need extra human effort for data labeling. To overcome this challenge, we introduce a novel self-transitional learning method called **MORPH**, which automatically switches its learning phase without *any* supervision. *MORPH* transitions its learning phase from *seeding* to *evolution* at the best transition point. In the seeding phase, the network is updated using all the samples to collect a seed of clean samples. Then, in the evolution phase, the network is updated only using the set of clean samples, which is continuously expanded and refined by the updated network. Extensive experiments using five benchmarks demonstrate substantial improvements over state-of-the-art methods even without any supervision.

6.1 Motivation and Overview

Prestopping empirically shows that a two-phase learning scheme helps effectively constructing a collection of clean samples which is called a maximal safe set. However, finding best transition point between the noise-robust period and the noise-prone period is very challenging. In *Prestopping*, two heuristics are proposed to fine the best point with minimal supervision from either (1) a clean validation set or (2) a known noise rate; however, it is less practical to use because they are hard to acquire in practice.

In this regard, we propose a novel *self-transitional learning* approach called *MORPH*, which automatically transitions its learning phase when a DNN enters the "noise-prone" period after the "noise-robust" period (i.e., the dashed line in Figure 6.1(b)). Thus, corresponding to these two periods, our key idea is to divide the training process into two learning phases:



(a) Default. (b) *MORPH*. (c) Test Error Convergence.

Figure 6.1: Key idea of *MORPH*: (a) and (b) show the memorization ratio when training a WideResNet-16-8 on FOOD-101N† with the real-world noise of 18.4%[51], where the memorization ratio is the number of memorized (see Definition 5.2.1) true- or false-labeled samples to the total number of true- or false-labeled training samples at each epoch. "Default" is a standard training method, and "*MORPH*" is our proposed one; (c) contrasts the convergence of their test error.

[51] The learning rate was decayed with cosine annealing.

1. **Seeding:** Owing to the negligible memorization of false-labeled samples, the network update is initiated using *all* the training samples in the noise-robust period. Because the samples memorized at this time are mostly true-labeled, they are exploited as a seed to derive a *maximal safe set* in the next phase. Note that the phase transition is fully automatic without *any* supervision.

2. **Evolution:** Without memorizing false-labeled samples, the network evolves by being updated *only* for the maximal safe set in the noise-prone period. Then, the updated network recognizes more true-labeled samples previously undistinguishable and filters out false-labeled samples incorrectly included. This alternating process repeats per iteration so that the maximal safe set is expanded and refined in the remaining noise-prone period.

Notably, *MORPH* effectively prevents the memorization of the false-labeled samples in the noise-prone period, as shown in Figure 6.1(b), by maintaining a collection of certainly true-labeled samples after exploiting the noise-robust period. As a result, as shown in Figure 6.1(c), the generalization performance of a DNN improves remarkably even in *real-world* noise.

6.2 Main Concept: Seeding and Evolution

In this section, we introduce a self-transitional learning approach called *MORPH* which comprises the following two phases. Differently from *Prestopping*, Phase I (Seeding) is transitioned without any supervision by Eq. (6.2) and Phase II (Evolution) keeps expanding the maximal safe set by alternating between two rules Eq. (6.4) and Eq. (6.5).

6.2.1 Phase I: Seeding during the Noise-Robust Period

Phase I initiates to update the network using *all* the training samples in a conventional way of Eq. (2.2) during the noise-robust period, where the memorization of false-labeled samples are suppressed. Concurrently, because most of the samples memorized until the transition point are true-labeled, *MORPH* collects them to form a seed to derive the maximal safe set in Phase II. The major technical challenge here is to estimate the best transition point.

The network predominantly learns true-labeled samples until the noise-prone period begins. That is, at the best phase transition point, the network *(i)* not only accumulates *little* noise from the false-labeled samples, *(ii)* but also acquires *sufficient* information from the true-labeled ones. In that sense, we revisit the two memorization metrics, namely, *memorization precision (MP)* and *memorization recall (MR)* in Definition 6.2.1,[52] which are indicators of evaluating the two aforementioned properties, respectively.

Definition 6.2.1: Memorization Metrics

Let $\mathcal{M}_t \subseteq \tilde{\mathcal{D}}$ be a set of memorized samples at time t according to Definition 5.2.1. Then, *memorization precision* and *recall* at time t are formulated by

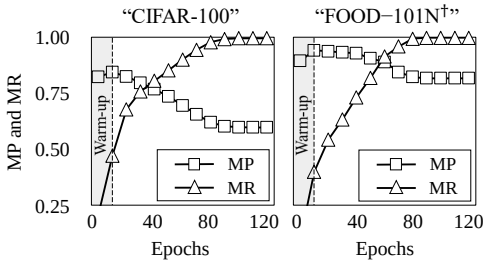
$$MP(t) = \frac{|\{(x,\tilde{y}) \in \mathcal{M}_t : \tilde{y} = y^*\}|}{|\mathcal{M}_t|} \quad \text{and} \quad MR(t) = \frac{|\{(x,\tilde{y}) \in \mathcal{M}_t : \tilde{y} = y^*\}|}{|\{(x,\tilde{y}) \in \tilde{\mathcal{D}} : \tilde{y} = y^*\}|}, \tag{6.1}$$

$$\text{where} \quad \mathcal{M}_t = \{(x,\tilde{y}) \in \tilde{\mathcal{D}} : \text{argmax}_y p(y|x,t;q) = \tilde{y}\}. \quad \square$$

[52] We repeat Definition 5.2.2 for ease of reading.

Figure 6.2 shows the change in MP and MR on the two *realistic* label noises over the training period. Owing to the memorization effect of a DNN, the two metrics were observed to naturally follow the *monotonicity* after a few warm-up epochs:

- *MP* monotonically decreases because a DNN tends to memorizes true-labeled samples first and then gradually memorizes all the false-labeled samples.
- *MR* monotonically increases because a DNN eventually memorizes all the true-labeled samples as the training progresses.



(a) Asymmetric Noise. (b) Real Noise.

Figure 6.2: MP and MR when training WideResNet-16-8 on CIFAR-100 with the asymmetric noise of 40% and FOOD-101N† with the real-world noise of 18.4%.

Under the monotonicity, the best transition point is the *only cross-point* of the two metrics, i.e., $MP(t) = MR(t)$, because it is the best trade-off between them. This approach is also supported by theoretical or empirical understanding of memorization in deep learning that a better generalization of a DNN is achieved when pure memorization (i.e., high MP) and its enough amount (i.e., high MR) are satisfied simultaneously [22, 127, 128, 129]. Then, by finding the answer of $MP(t) = MR(t)$ in Eq. (5.2), the cross-point satisfies

$$|\mathcal{M}_t| = |\{(x, \tilde{y}) \in \tilde{\mathcal{D}} : \tilde{y} = y^*\}| = (1-\tau)|\tilde{\mathcal{D}}| \quad \therefore |\mathcal{M}_t| = (1-\tau)|\tilde{\mathcal{D}}|, \tag{6.2}$$

where τ is the true noise rate. Because τ is typically unknown, *MORPH* uses an estimated noise rate $\hat{\tau}$ to check the condition in Eq. (6.2) for the phase transition.

Regarding the noise rate estimation, we fit a two-component Gaussian Mixture Model (GMM) to model the loss distribution of true-labeled and false-labeled samples because the distribution is bi-modal [36, 130]. At each epoch, *MORPH* accumulates the loss of all the training samples and fits the GMM to the accumulated loss by using the Expectation-Maximization (EM) algorithm. The probability of a sample x being false-labeled is obtained through its posterior probability. Accordingly, the noise rate τ is estimated by

$$\hat{\tau} = \mathbb{E}_{(x,\tilde{y})\in\tilde{\mathcal{D}}}[p\big(g|\mathcal{L}(f(x;\theta_t),\tilde{y})\big)], \tag{6.3}$$

where g is the Gaussian component with a larger mean (i.e., larger loss). A thorough analysis of estimating the noise rate is provided in Chapter 6.6.

Overall, *MORPH* transitions the learning phase at time t_{tr} when the number of memorized samples is greater than or equal to the estimated number of true-labeled ones (i.e., $|\mathcal{M}_{t_{tr}}| \geq (1-\hat{\tau})|\tilde{\mathcal{D}}|$). Please note that *MORPH* requires *neither* a true noise rate *nor* a clean validation set.

6.2.2 Phase II: Evolution during the Noise-Prone Period

Phase II robustly updates the network *only* using the selected clean samples called the *maximal safe set* in Definition 6.2.2. By the definition of the transition point, the initial maximal safe set is qualitatively clean and quantitatively sufficient (i.e., high MP and MR). Moreover, the set is further expanded and refined by iteratively updating the network with the current set $\mathcal{S}_t$ as in Eq. (6.4) and deriving a more refined set $\mathcal{S}_{t+1}$ as in Eq. (6.5). By the fact that false-labeled samples are easily forgotten [131], the network keeps more true-labeled samples $\mathcal{C}$ owing to the robust update, while forgetting the false-labeled samples $\mathcal{R}$ which were incorrectly memorized earlier. In this way, *MORPH* becomes better generalized to almost all true-labeled samples through the evolution of the maximal safe set.

Definition 6.2.2: Maximal Safe Set

A *maximal safe set* $\mathcal{S}_t$ is defined as the set of samples expected to be true-labeled at time $t(\geq t_{tr})$. It starts from $\mathcal{M}_{t_{tr}}$ at $t = t_{tr}$ and is managed as follows:

1. The refinement of the maximal safe set $\mathcal{S}_t$ accompanies the update of the network parameter θ_t. To be free from memorizing false-labeled samples in Phase II, the network parameter θ_{t+1} is robustly learned only using the samples in $\mathcal{S}_t$ out of the mini-batch $\mathcal{B}_t$ by

$$\theta_{t+1} = \theta_t - \alpha\nabla\big(\frac{1}{|\mathcal{B}_t \cap \mathcal{S}_t|}\sum_{(x,\tilde{y})\in(\mathcal{B}_t\cap\mathcal{S}_t)}\mathcal{L}(f(x;\theta_t),\tilde{y})\big). \tag{6.4}$$

2. Then, $\mathcal{S}_{t+1}$ reflects the changes by the network update at time $t+1$, i.e., newly memorized samples $\mathcal{C}$ and newly forgotten samples $\mathcal{R}$, as formulated by

$$\mathcal{S}_{t+1} = \mathcal{S}_t + \mathcal{C} - \mathcal{R}, \quad \text{where} \quad \begin{cases} \mathcal{C} &= \{(x,\tilde{y}) \in (\mathcal{B}_t \cap \mathcal{S}_t^c) : \text{argmax}_y p(y|x,t+1;q) = \tilde{y}\}, \\ \mathcal{R} &= \{(x,\tilde{y}) \in (\mathcal{B}_t \cap \mathcal{S}_t) : \text{argmax}_y p(y|x,t+1;q) \neq \tilde{y}\}. \end{cases} \tag{6.5}$$

□

To reduce the possibility of the overfitting to a small *initial* set, which is typically observed with a very high noise rate, *MORPH* combines the supervised loss term in Eq. (6.4) with an unsupervised loss term $\mathcal{J}(\theta_t)$ widely known as *consistency regularization* [114, 132, 133]. Without relying on possibly unreliable labels, this regularization effectively helps learn the *dark knowledge* [134] from *all* the training samples by penalizing the prediction difference between the original sample x and its augmented sample $\hat{x}$. (For the experiments in Chapter 6.4, $\hat{x}$'s were generated by random crops and horizontal flips.) Hence, the update rule is finally defined by

$$\theta_{t+1}=\theta_t-\alpha\nabla\big(\frac{1}{|\mathcal{B}_t \cap \mathcal{S}_t|}\sum_{x\in(\mathcal{B}_t\cap\mathcal{S}_t)}\mathcal{L}(f(x;\theta),\tilde{y})+w(t)\mathcal{J}(\theta_t)\big), \text{where } \mathcal{J}(\theta_t)=\frac{1}{\mathcal{B}_t}\sum_{x\in\mathcal{B}_t}||z(x;\theta_t)-z(\hat{x};\theta_t)||_2^2, \tag{6.6}$$

where $z(x;\theta_t)$ is the softmax vector of a sample x and $w(t)$ is a Gaussian ramp-up function to gradually increase the weight to the maximum value w_{max}. According to our ablation study in Chapter 6.5.1, the robustness of *MORPH* is considerably enhanced by using the consistency regularization.

6.3 Algorithm Description

Algorithm 4 describes the overall procedure of *MORPH*, which is self-explanatory. First, the network is trained on the noisy training data $\tilde{\mathcal{D}}$ in the *default* manner (Lines 5–8). During the first phase, the noise rate is estimated and subsequently used to find the moment when the noise-prone period begins. Here, Phase I transitions to Phase II if the transition condition holds (Lines 9–14). Subsequently, during the second phase, the mini-batch samples in the current maximal safe set are selected to update the network parameter with the consistency regularization (Lines 18–20). The rest mini-batch samples are excluded to pursue the robust learning. Finally, the maximal safe set is refined by reflecting the change in network memorization resulting from the update (Lines 21–22).

The main "additional" costs of *MORPH* are (1) the estimation of the noise rate (Line 10) and (2) the additional inference step for the consistency regularization (Line 20). Because the noise rate is estimated using the EM algorithm once per epoch, its cost is negligible compared with that of the forward and backward steps of a complex network. Thus, the additional inference in Phase II is the only part that

Algorithm 4 *MORPH*

Require: $\tilde{\mathcal{D}}$: data, *epochs*: total number of epochs, q: history length, w_{max}: maximum weight for $\mathcal{J}$

Ensure: θ_t: network parameters, $\mathcal{S}_t$: final maximal safe set

$t \leftarrow 1$; $\theta_t \leftarrow$ Initialize the network parameter; $\mathcal{S}_t \leftarrow \emptyset$;
/* **I. Seeding during Noise-Robust Period** */
for $i = 1$ **to** *epochs* **do**
  for $j = 1$ **to** $|\tilde{\mathcal{D}}|/|\mathcal{B}_t|$ **do**
    Draw a mini-batch $\mathcal{B}_t$ from $\tilde{\mathcal{D}}$;
    /* Standard update by Eq. (2.2) */
    $\theta_{t+1} = \theta_t - \alpha \nabla \big(\frac{1}{|\mathcal{B}_t|} \sum_{x \in \mathcal{B}_t} \mathcal{L}(f(x; \theta_t), \tilde{y}) \big)$;
    $t \leftarrow t + 1$;
  /* Noise rate estimation by Eq. (6.3) */
  $\hat{\tau} \leftarrow \mathbb{E}_{(x,\tilde{y}) \in \tilde{\mathcal{D}}} [p\big(g | \mathcal{L}(f(x; \theta_t), \tilde{y})\big)]$;
  /* Checking the phase transition condition */
  if $|\mathcal{M}_t| \geq (1 - \hat{\tau})|\tilde{\mathcal{D}}|$ **then**
    /* Assigning the initial maximal safe set */
    $t_{tr} \leftarrow t$; $\mathcal{S}_{t_{tr}} \leftarrow \mathcal{M}_{t_{tr}}$;
/* **II. Evolution during Noise-prone Period** */
for $i = t_{tr}|\mathcal{B}_t|/|\tilde{\mathcal{D}}| + 1$ **to** *epochs* **do**
  for $j = 1$ **to** $|\tilde{\mathcal{D}}|/|\mathcal{B}_t|$ **do**
    Draw a mini-batch $\mathcal{B}_t$ from $\tilde{\mathcal{D}}$;
    /* Robust update by Eq. (6.6) */
    $\theta_{t+1} = \theta_t - \alpha \nabla \big(\frac{1}{|\mathcal{B}_t \cap \mathcal{S}_t|} \sum_{x \in (\mathcal{B}_t \cap \mathcal{S}_t)} \mathcal{L}(f(x; \theta_t), \tilde{y}) + w(t) \mathcal{J}(\theta_t) \big)$;
    /* Updating the maximal safe set by Eq. (6.5) */
    $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t + \mathcal{C} - \mathcal{R}$;
    $t \leftarrow t + 1$;
return θ_t, $\mathcal{S}_t$;

increases the time complexity. Nevertheless, the additional cost is relatively *cheap* considering that other sample selection methods require either an additional network or multiple training rounds. In Chapter 6.4.2, we have empirically shown that *MORPH* is much faster than other methods.

6.4 Main Experiments

Detasets and Noise Injection: We performed an image classification task on *five* benchmark datasets: CIFAR-10, CIFAR-100, Tiny-ImageNet, WebVision 1.0, and FOOD-101N. As all the labels in CIFAR and Tiny-ImageNet are clean, we artificially corrupted the labels in these datasets following the previous work [40, 4]. We applied two synthetic label noises: (1) *symmetric noise* and (2) *asymmetric noise*. Random crops and horizontal flips were applied for data augmentation.

Networks and Hyperparameters: For CIFAR and Tiny-ImageNet, we trained a WideResNet-16-8 [135] from scratch using SGD with a momentum of 0.9, a batch size of 128, a dropout of 0.1, and a weight decay of 0.0005. The network was trained for 120 epochs with an initial learning rate of 0.1, which was decayed with cosine annealing [136]. As for the hyperparameters, we used the best history length

$q = 10$ and maximum weight $w_{max} = 5.0$, both of which were obtained via a grid search (see Chapter 6.4.4 for details). The w value gradually increased from 0 to w_{max} using a Gaussian ramp-up function because the network with a large w gets stuck in a degenerate solution at the beginning [132].

WebVision 1.0 and FOOD-101N are two large-scale datasets with real-world label noise [2, 21]. WebVision 1.0 contains 2.4M images crawled from websites using the 1,000 concepts in ImageNet ILSVRC12 [137], and FOOD-101N contains 310K food images crawled from websites with the FOOD-101 taxonomy [120]. We followed the same experimental configuration in the previous work. For WebVision 1.0, we trained an InceptionResNet-V2 [138] from scratch for the first 50 classes of the Google image subset [1]; it was trained for 120 epochs using SGD with a momentum of 0.9 and an initial learning rate of 0.1, which was divided by 10 after 40 and 80 epochs (refer to [1]). For FOOD-101N, we fine-tuned a ResNet-50 with the ImageNet pretrained weights for the entire training set [2, 3]; it was fine-tuned for 60 epochs using SGD with a momentum of 0.9 and an initial learning rate of 0.01, which was divided by 10 after 30 epochs (refer to [2]). Regardless of the dataset, we used a batch size of 64, a dropout of 0.4, and a weight decay of 0.001. Random crops and horizontal flips were applied for data augmentation.

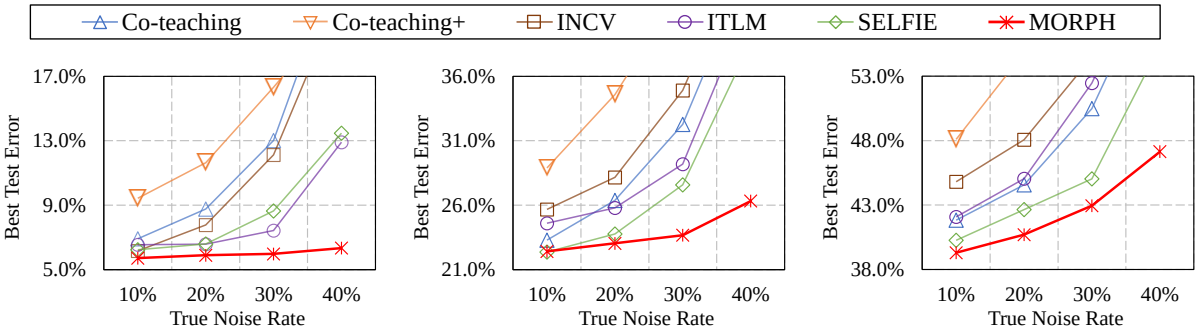
Algorithms: All the algorithms were implemented using TensorFlow 2.1.0 and executed using 16 NVIDIA Titan Volta GPUs. In support of reliable evaluation, we repeated every task *thrice* and reported the average test (or validation) error as well as the average training time. For reproducibility, we provide the full source code at https://bit.ly/2U7bMeR.

6.4.1 Performance Comparison

Robustness Comparison with Synthetic Noises

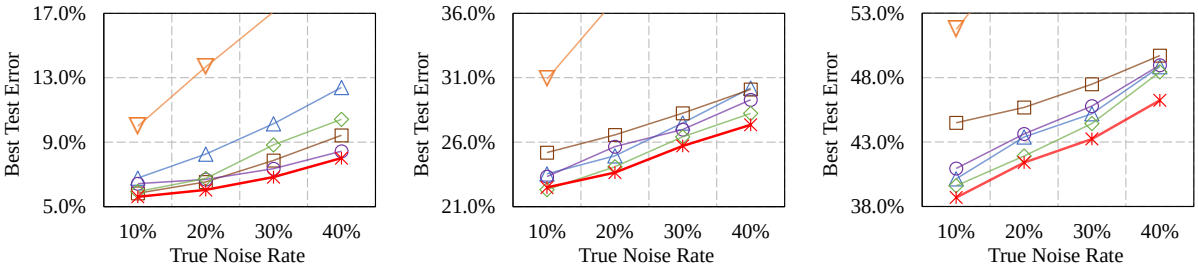
We compared *MORPH* with the *five* state-of-the-art sample selection methods: *Co-teaching* [40], *Co-teaching+* [4], *INCV* [1], *ITLM* [42], and *SELFIE* [19]. The other methods were re-implemented by the authors for fair comparison, and their hyperparameters were favorably configured to the best values:

- ***Co-teaching(+)*:** To decrease the number of selected samples gradually at the beginning of the training, the warm-up epoch is required as a hyperparameter; it was set to be 15, which is reported to achieve the best performance in the original paper [40].
- ***INCV*:** Following the original paper [1], the total number of training rounds was set to be 4; the network was trained for 50 epochs using the Adam optimizer; an initial learning rate was set to be 0.001, which was divided by 2 after 20 and 30 epochs and finally fixed to be 0.0001 after 40 epochs. Subsequently, all the training samples selected by *INCV* were used to retrain the network using *Co-teaching* with the same configuration described in Chapter 6.4.
- ***ITLM*:** Because it iterates the training process multiple times as well, the total number of training rounds was set to be 5. As mentioned in the original paper [42], the training process for the first 4 rounds was early stopped because it may help filter out false-labeled samples. Subsequently, without early stopping, the network was retrained using the samples selected from the 4th round during the last training round.
- ***SELFIE*:** Four hyperparameters are required for *SELFIE*. The warm-up epoch for sample selection was set to be 15 similar to *Co-teaching*. The uncertainty threshold and the history length for loss correction were set to be 0.05 and 15, respectively, which are the best values obtained from a grid search in the original paper [19]. The training process was restarted twice according to the authors' recommendation.



(a) CIFAR-10. (b) CIFAR-100. (c) Tiny-ImageNet.

Figure 6.3: Best test errors using WideResNet with varying **asymmetric noise** rates.



(a) CIFAR-10. (b) CIFAR-100. (c) Tiny-ImageNet.

Figure 6.4: Best test errors using WideResNet with varying **symmetric noise** rates.

Method	WebVision Val.	ILSVRC2012 Val.
F-correction [31]	38.88 (17.32)	42.64 (17.64)
Decouple [38]	37.46 (15.26)	41.74 (17.74)
Co-teaching [40]	36.42 (14.80)	38.52 (15.30)
MentorNet [39]	37.00 (18.60)	42.20 (20.08)
D2L [37]	37.32 (16.00)	42.20 (18.64)
INCV [1]	34.76 (14.66)	38.40 (15.02)
MORPH	**30.32 (11.33)**	**33.09 (12.62)**

Table 6.1: Comparison with state-of-the-art methods trained on WebVision V1. The value outside (inside) the parentheses denotes the top-1 (top-5) classification error (%) on the WebVision validation set and the ImageNet ILSVRC12 validation set. The results for baseline methods are borrowed from [1].

Method	FOOD-101 Val.
Cross-Entropy [2]	18.56
Weakly Supervised [139]	16.57
CleanNet $(w_{hard})^{\dagger}$ [2]	16.53
CleanNet $(w_{soft})^{\dagger}$ [2]	16.05
Guidance Learning† [3]	15.80
MORPH	**14.71**

Table 6.2: Comparison with state-of-the-art methods trained on FOOD-101N. The value denotes the top-1 classification error (%) on the FOOD-101 validation set. The results for baseline methods are borrowed from [2, 3]. † indicates that extra clean (or verification) labels were used for supervision.

Figures 6.3 and 6.4 show the test errors of the six sample selection methods on three datasets with varying *asymmetric* and *symmetric* noise rates, respectively. See Chapter 6.7.1 for the tabular reports.

Asymmetric Noise: *MORPH* generally achieved the lowest test errors with respect to a wide range of noise rates. The error reduction became larger as the noise rate increased, reaching 6.6*pp*–27.0*pp* at a heavy noise rate of 40%. In contrast, the performance of the other methods worsened rapidly, because the small-loss trick could not distinguish well true-labeled samples from false-labeled samples in asymmetric

and real-world noises, as illustrated in Figures 5.1(b) and (c).

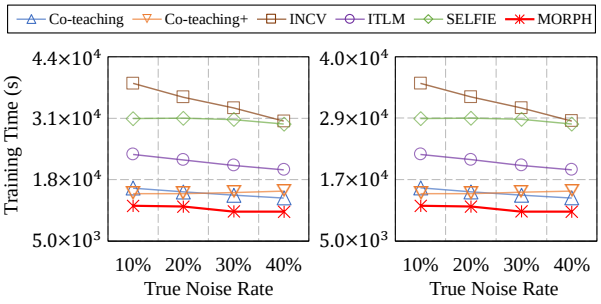
Symmetric Noise: *MORPH* generally outperformed the other methods again, though the error reduction was relatively small, i.e., 0.42*pp*–26.3*pp* at a heavy noise rate of 40%. The small-loss trick was turned out to be appropriate for symmetric noise, as illustrated in Figure 5.1(a).

Robustness Comparison with Real-World Noises

Tables 6.1 and 6.2 summarize the results on WebVision V1 and FOOD-101N, respectively. *MORPH* maintained its dominance over multiple state-of-the art methods for *real-world* label noise as well. It improved the top-1 validation error by 4.44*pp*–8.56*pp* and 1.09*pp*–3.85*pp* in WebVision V1 and FOOD-101N, respectively. This lowest error of *MORPH* in FOOD-101N was achieved even without extra supervision from the clean (or verification) labels.

6.4.2 Efficiency Comparison

Another advantage of *MORPH* is its efficiency in training the network. Figure 6.5 shows the training time of the six sample selection methods on two CIFAR datasets. *MORPH* was the fastest in each case because of its relatively cheap additional costs, as elaborated in Chapter 6.3. Overall, *MORPH* was 1.13–3.08 times faster than the other methods. The difference between the training time of the remaining methods tended to be determined by the total number of training rounds, which is represented by one of their hyperparameters. The general trend in training time remained the same for other datasets as well. See Table 6.7.2 in Chapter 6.7.2 for the result of all datasets.



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

Figure 6.5: Training time on CIFAR-10 and CIFAR-100 datasets with asymmetric noise.

6.4.3 Optimality of the Best Transition Point

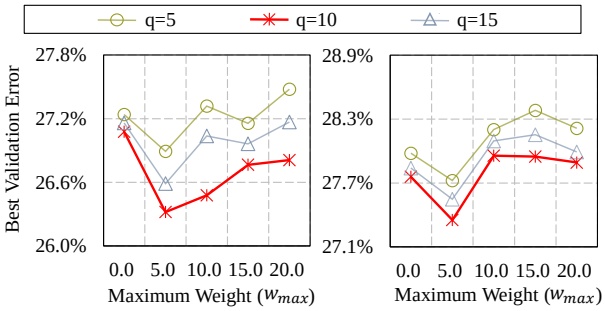
The optimality of the best transition point is a key issue. Hence, we enrich the empirical evidence for validating the optimality. As shown in Table 6.3 the best test error of *MORPH* was generally achieved at the estimated best point other than those around the best point. Overall, this empirical result confirms the optimality of the best point estimated by *MORPH*.

Table 6.3: Best test errors (%) of *MORPH* with "early" or "late" transition based on the best point, where α is added to the estimated noise rate in Eq. (4) to force early or late transition. CIFARs with two synthetic noises of 40% were used.

Transition Point	Early Trans.		Best Trans.	Late Trans.	
A value α	+10%	+5%	+0%	-5%	-10%
CIFAR-10 (Symmetric)	8.71	8.37	**8.01**	9.39	9.75
CIFAR-10 (Asymmetric)	7.53	**6.20**	6.34	7.43	8.08
CIFAR-100 (Symmetric.)	29.8	28.5	**27.4**	28.1	28.5
CIFAR-100 (Asymmetric)	28.3	27.2	**26.3**	27.0	28.9

6.4.4 Hyperparameter Selection

MORPH requires two additional hyperparameters: the history length q and the maximum regularization weight w_{max}. To ascertain the optimal values of these hyperparameters, we trained a WideResNet-16-8 on CIFAR-100 at a noise rate of 40%. Here, because no validation data exists for the CIFAR-100 dataset, we constructed a small clean validation set by randomly selecting $1,000$ images from the original training set of $50,000$ images. Then the noise injection process was applied to only the rest $49,000$ training images. Figure 6.6 shows the validation errors of *MORPH* obtained by grid search on the noisy CIFAR-100 dataset. The two hyperparameters were chosen in the grid $q \in \{5, 10, 15\}$ and $w_{max} \in \{0.0, 5.0, 10.0, 15.0\}$. Typically, the lowest validation error depending on the history length q was achieved when the value of q was 10 for both noise types. As for the maximum weight w_{max}, the validation error was observed to be the lowest when the value of w_{max} was 5.0. Therefore, we set the values of q and w_{max} to be 10 and 5.0, respectively, in all experiments.



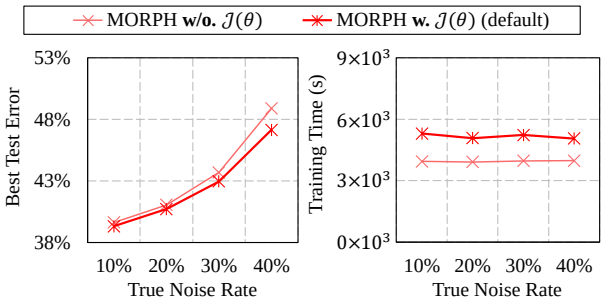
(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 6.6: Hyperparameter selection on the CIFAR-100 with two noise types of 40%.

6.5 Ablation Study

6.5.1 Effect of Consistency Regularization

Figure 6.7 shows the effect of the consistency regularization. Interestingly, as shown in Figure 6.7(a), the test error of *MORPH* was further improved by adding the consistency loss in Eq. (6.6). The higher the noise rate, the greater the benefit because the overfitting issue caused by a smaller size of the initial maximal safe set is mitigated. Regarding the training time, as shown in Figure 6.7(b), the regularization slightly slowed down the training speed owing to the extra inference steps. Because *MORPH* achieved lower test error than the other robust methods even *without* the regularization (see Chapter 6.7.1 for details), practitioners may skip employing the regularization if their time budgets are restricted.

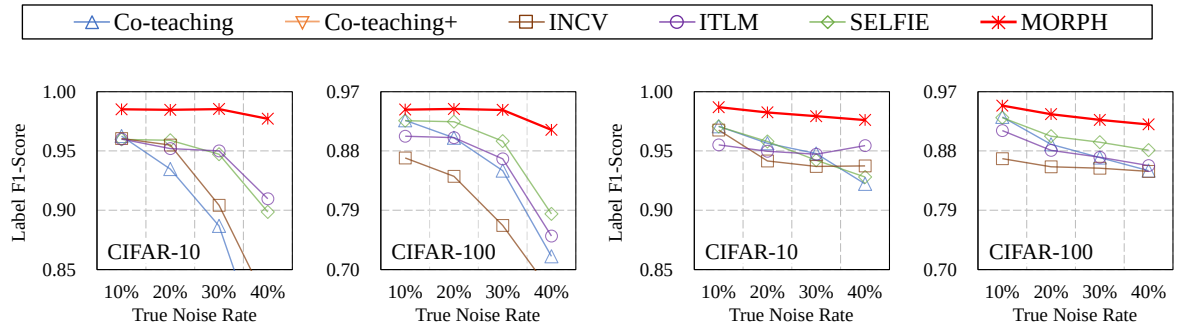


(a) Best Test Error. (b) Training Time.

Figure 6.7: Effect of the consistency loss $\mathcal{J}(\theta)$ on Tiny-ImageNet with asymmetric noise.

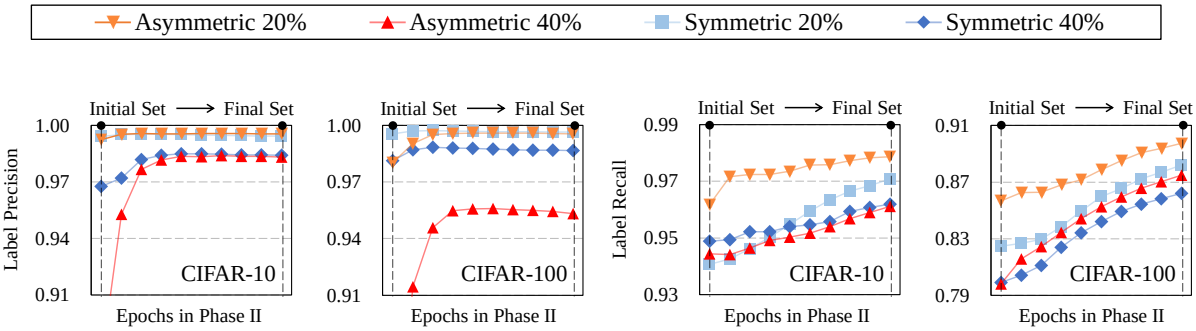
6.5.2 Analysis on Selected Clean Samples

The superior robustness of *MORPH* is attributed to its high *label precision* (LP) and *label recall* (LR), which are calculated by replacing $\mathcal{M}$ for MP and MR in Eq. (6.1) with the set of selected clean samples [40]. Hence, we compare *MORPH* with the other five sample selection methods in terms of the *label F1-score* $= (2 \cdot LP \cdot LR)/(LP + LR)$.



(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 6.8: Label F1-scores on two CIFAR datasets using WideResNet with varying noise rates.



(a) Label Precision. (b) Label Recall.

Figure 6.9: Evolution of the maximal safe set in Phase II using WideResNet on two CIFAR datasets.

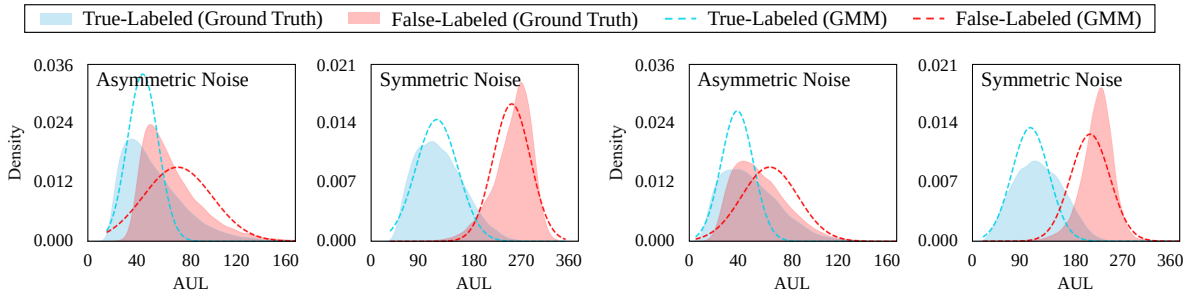
To evaluate the label F1-score, *MORPH* used its final maximal safe set, *Co-teaching(+)* and *SELFIE* used the samples selected during their last epoch, and *INCV* and *ITLM* used the samples selected for their final training round. Figure 6.8 shows their label F1-scores. Only *MORPH* achieved consistently high label F1-scores of over 0.91 in all the cases. This result corroborates that *MORPH* identifies true-labeled samples with high precision and recall regardless of the noise type and rate.

6.5.3 Evolution of the Maximal Safe Set

Figure 6.9 shows the LP and LR values on the maximal safe set obtained at each epoch since Phase II begins. At the beginning of Phase II, both LP and LR already exhibited fairly high values because the initial set is derived from the samples memorized at the transition point, which is best compromise between MP and MR. Moreover, they increased significantly along with further training by alternatively updating the network by Eq. (5.3) and refining the maximal safe set by Eq. (6.5). Notably, their improvement was consistently observed regardless of the noise type and rate, thereby achieving remarkably high LP and LR at the end of training in each case. The high F1-score of *MORPH* in Chapter 6.5.2 is well supported by this evolution of the maximal safe set in Phase II.

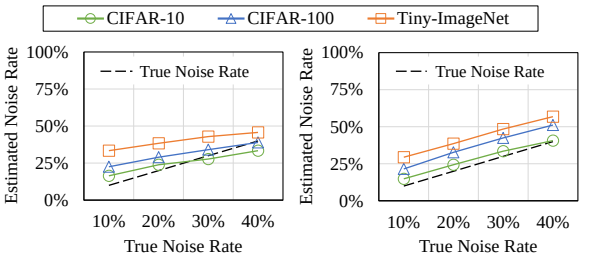
6.6 Noise Rate Estimation

Recently, numerous studies have provided practical algorithms for estimating the noise rate of the training data [36, 1, 140, 130]. Because the best transition point is found by estimating the noise rate, we explored *two* widely used methods in the literature, as follows:



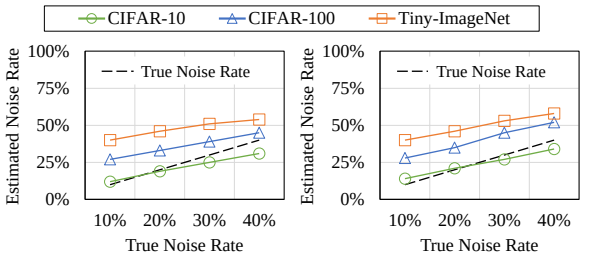
(a) CIFAR-10. (b) CIFAR-100.

Figure 6.10: AUL distributions of true-labeled and false-labeled samples using the ground-truth label and the GMM on two CIFAR datasets with two synthetic noises of 40%.



(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 6.11: Estimation with GMM.



(a) Asymmetric Noise. (b) Symmetric Noise.

Figure 6.12: Estimation with cross-validation.

1. **Gaussian Mixture Model (GMM):** The first method is exploiting a one-dimensional and two-component GMM to model the loss distribution of true-labeled and false-labeled samples [36, 130]. Because the loss distribution tends to be bimodal, the probability of a sample being a false-labeled sample is obtained through its posterior probability. Subsequently, the noise rate is estimated by computing the expectation of the posterior probability for all the training samples. However, in considering that the network eventually memorizes all the training samples, the training loss becomes less separable by the GMM as the training progresses. Thus, we computed the *Area Under the Loss curve* (AUL) [130], which is the sum of the samples' training losses obtained from all previous training epochs. The main benefit of the AUL is that its distribution remains separable even after the loss signal decays in later epochs. Therefore, as shown in Figure 6.10, the loss distributions of true-labeled and false-labeled samples are modeled by fitting the GMM to the AULs of all the training samples, and the noise rate at time t is estimated by

$$\hat{\tau} = \mathbb{E}_{(x,\tilde{y})\in\tilde{\mathcal{D}}}[p\big(g|AUL_t(x,\tilde{y})\big)], \quad \text{where } AUL_t(x,\tilde{y}) = \sum_{i=1}^{t} \mathcal{L}(f(x;\theta_t),\tilde{y}), \tag{6.7}$$

and g is the Gaussian component with a larger mean (i.e., larger AUL).

2. **Cross-Validation:** The second method is estimating the noise rate by applying cross-validation on two randomly divided noisy training datasets $\tilde{\mathcal{D}}_1$ and $\tilde{\mathcal{D}}_2$ [1]. Under the assumptions that these two datasets share exactly the same label transition matrix, the noise rate quantifies the test accuracy of DNNs, which are trained and tested on previously mentioned noisy datasets $\tilde{\mathcal{D}}_1$ and $\tilde{\mathcal{D}}_2$, respectively. In the case of synthetic noises, the test accuracy is approximated by a quadratic function of the noise rate. Therefore, the noise rate can be estimated from the test accuracy obtained by the cross-validation. Refer to [1] for details.

Figures 6.11 and 6.12 show the estimated noise rates using the GMM and cross-validation when training a WideResNet-16-8 on three benchmark datasets with varying noise rates. Generally, both methods performed well on the easy dataset (i.e., CIFAR-10), but their performance worsened as the training difficulty increased from CIFAR-10 to Tiny-ImageNet because the true-labeled but hard samples are *not* clearly distinguishable from the false-labeled samples. Nevertheless, the GMM showed considerably better performance than the cross-validation even in the two difficult datasets, CIFAR-100 and Tiny-ImageNet. Thus, we adopted the GMM for estimating the noise rate in *MORPH*.

6.7 Completed Results

6.7.1 Test Error with Synthetic Noises

Table 6.4 shows the *test error* of seven training methods using WideResNet-16-8 on three *simulated* noisy datasets with varying noise rates. Two variants of *MORPH* depending on the existence of the consistency regularization were included for comparison.

Table 6.4: Best and last test errors (%) of seven training methods on two types of **synthetic** noises.

Noise Type	Asymmetric Noise in Figure 6.3			Symmetric Noise in Figure 6.4		
Data Set	CIFAR-10	CIFAR-100	Tiny-Image	CIFAR-10	CIFAR-100	Tiny-Image
Co-teaching	6.92	23.3	41.8	6.77	23.5	40.1
Co-teaching+	9.43	28.9	48.1	10.0	30.9	51.8
INCV	6.16	25.7	44.8	5.82	25.2	44.5
ITLM	6.55	24.6	42.1	6.43	23.3	41.0
SELFIE	6.23	22.4	40.3	5.93	22.3	39.6
MORPH **w/o.** $\mathcal{J}(\theta)$	**5.59**	**22.1**	39.6	**5.28**	**21.8**	39.2
MORPH **w.** $\mathcal{J}(\theta)$	5.71	22.4	**39.3**	5.61	22.5	**38.7**

(a) Best and last test errors under asymmetric and symmetric noises of 10% ($\tau = 0.1$).

Noise Type	Asymmetric Noise in Figure 6.3			Symmetric Noise in Figure 6.4		
Data Set	CIFAR-10	CIFAR-100	Tiny-Image	CIFAR-10	CIFAR-100	Tiny-Image
Co-teaching	8.76	26.4	44.6	8.26	24.9	43.4
Co-teaching+	11.6	34.6	54.8	13.7	37.0	58.3
INCV	7.77	28.2	48.1	6.53	26.6	45.7
ITLM	6.58	25.8	45.1	6.69	25.7	43.6
SELFIE	6.59	23.8	42.7	6.74	24.1	41.9
MORPH **w/o.** $\mathcal{J}(\theta)$	**5.88**	**22.8**	41.1	**5.92**	**23.2**	**41.4**
MORPH **w.** $\mathcal{J}(\theta)$	5.89	23.0	**40.7**	6.03	23.6	**41.4**

(b) Best and last test errors under asymmetric and symmetric noises of 20% ($\tau = 0.2$).

Noise Type	Asymmetric Noise in Figure 6.3			Symmetric Noise in Figure 6.4		
Data Set	CIFAR-10	CIFAR-100	Tiny-Image	CIFAR-10	CIFAR-100	Tiny-Image
Co-teaching	13.0	32.3	50.5	10.2	27.5	45.2
Co-teaching+	16.3	42.9	60.9	17.1	45.0	62.7
INCV	12.1	34.9	54.5	7.87	28.2	47.5
ITLM	7.43	29.2	52.5	7.36	27.0	45.8
SELFIE	8.64	27.6	45.0	8.83	26.4	44.4
MORPH **w/o.** $\mathcal{J}(\theta)$	6.11	24.2	43.7	6.85	25.8	43.7
MORPH **w.** $\mathcal{J}(\theta)$	**5.97**	**23.7**	**43.0**	**6.83**	**25.7**	**43.2**

(c) Best and last test errors under asymmetric and symmetric noises of 30% ($\tau = 0.3$).

Noise Type	Asymmetric Noise in Figure 6.3			Symmetric Noise in Figure 6.4		
Data Set	CIFAR-10	CIFAR-100	Tiny-Image	CIFAR-10	CIFAR-100	Tiny-Image
Co-teaching	24.7	44.8	61.8	12.4	30.2	48.8
Co-teaching+	21.3	53.3	66.5	20.3	53.7	64.2
INCV	22.2	47.0	63.8	9.43	30.1	49.7
ITLM	12.9	41.9	63.9	8.43	29.3	49.0
SELFIE	13.5	38.6	55.4	10.4	28.2	48.4
MORPH **w/o.** $\mathcal{J}(\theta)$	7.25	27.1	48.9	8.27	27.8	47.0
MORPH **w.** $\mathcal{J}(\theta)$	**6.34**	**26.3**	**47.2**	**8.01**	**27.4**	**46.2**

(d) Best and last test errors under asymmetric and symmetric noises of 40% ($\tau = 0.4$).

6.7.2 Training Time with Synthetic Noises

Table 6.7.2 shows the *training time* of seven training methods using WideResNet-16-8 on three *simulated* noisy datasets with varying noise rates.

Table 6.5: Training time (sec) of seven training methods on two types of **synthetic** noises.

Noise Type	Asymmetric Noise in Figure 6.3			Symmetric Noise in Figure 6.4		
Data Set	CIFAR-10	CIFAR-100	Tiny-Image	CIFAR-10	CIFAR-100	Tiny-Image
Co-teaching	16,217	15,575	68,404	16,443	15,527	70,409
Co-teaching+	15,051	14,834	16,450	14,834	16,581	80,440
INCV	38,338	33,048	122,771	38,217	32,844	123,247
ITLM	23,341	22,162	79,246	23,701	22,384	83,371
SELFIE	30,894	31,107	121,787	31,319	31,107	123,663
MORPH **w/o.** $\mathcal{J}(\theta)$	**9,048**	**9,358**	**39,361**	**9,173**	**9,361**	**40,003**
MORPH **w.** $\mathcal{J}(\theta)$	12,442	12,091	53,009	12,229	12,182	50,495

(a) Training time under asymmetric and symmetric noises of 10% ($\tau = 0.1$).

Noise Type	Asymmetric Noise in Figure 6.3			Symmetric Noise in Figure 6.4		
Data Set	CIFAR-10	CIFAR-100	Tiny-Image	CIFAR-10	CIFAR-100	Tiny-Image
Co-teaching	15,382	14,563	64,710	15,351	14,103	65,121
Co-teaching+	15,019	16,936	82,391	15,229	17,169	82,639
INCV	35,456	30,520	113,722	35,707	30,261	113,208
ITLM	22,192	20,759	74,358	22,102	20,460	74,875
SELFIE	30,992	30,650	119,234	30,916	30,359	66,473
MORPH **w/o.** $\mathcal{J}(\theta)$	**9,258**	**9,497**	**39,102**	**9,363**	**9,412**	**39,677**
MORPH **w.** $\mathcal{J}(\theta)$	12,296	11,840	50,747	12,046	12,259	49,020

(b) Training time under asymmetric and symmetric noises of 20% ($\tau = 0.2$).

Noise Type	Asymmetric Noise in Figure 6.3			Symmetric Noise in Figure 6.4		
Data Set	CIFAR-10	CIFAR-100	Tiny-Image	CIFAR-10	CIFAR-100	Tiny-Image
Co-teaching	14,672	13,898	61,922	14,092	12,989	59,299
Co-teaching+	15,285	17,230	83,036	15,390	17,394	83,279
INCV	33,161	28,484	107,899	32,594	27,549	105,857
ITLM	21,018	19,477	70,823	19,972	18,381	66,473
SELFIE	30,728	30,120	118,978	30,619	29,573	117,016
MORPH **w/o.** $\mathcal{J}(\theta)$	**9,604**	**9,491**	**39,641**	**9,346**	**9,484**	**40,114**
MORPH **w.** $\mathcal{J}(\theta)$	11,220	11,775	52,242	11,875	12,175	49,446

(c) Training time under asymmetric and symmetric noises of 30% ($\tau = 0.3$).

Noise Type	Asymmetric Noise in Figure 6.3			Symmetric Noise in Figure 6.4		
Data Set	CIFAR-10	CIFAR-100	Tiny-Image	CIFAR-10	CIFAR-100	Tiny-Image
Co-teaching	14,107	13,386	60,262	13,171	11,886	53,660
Co-teaching+	15,527	17,244	82,982	15,365	17,459	80,038
INCV	30,376	26,628	104,597	28,915	24,668	99,314
ITLM	20,068	18,212	69,037	18,763	16,749	60,469
SELFIE	29,730	28,352	115,208	29,436	27,931	113,380
MORPH **w/o.** $\mathcal{J}(\theta)$	**9,646**	**9,477**	**39,728**	**9,598**	**9,488**	**40,154**
MORPH **w.** $\mathcal{J}(\theta)$	11,198	11,828	50,594	11,331	12,049	49,412

(d) Training time under asymmetric and symmetric noises of 40% ($\tau = 0.4$).

6.8 Chapter Conclusions

Conclusion

In this chapter, we proposed a novel self-transitional learning scheme called *MORPH* for noisy training data. The first phase exploits all training samples and estimates the most optimal transition point. The second phase completes the rest of the training process using only a maximal safe set with high label precision and recall. *MORPH* can be easily applied to many real-world cases because it requires neither a true noise rate nor a clean validation set. Through extensive experiments using various real-world and simulated noisy datasets, we verified that *MORPH* consistently exhibited significant improvement in both robustness and efficiency compared with state-of-the-art methods. Overall, we believe that the division of the training process into two phases, as in the proposed method, unveils a new approach to robust training and can inspire subsequent studies.

Chapter 7. Conclusions and Future Works

7.1 Conclusions

DNNs easily overfit to false labels owing to their high capacity in totally memorizing all noisy training samples. This overfitting issue still remains even with various conventional regularization techniques, such as dropout and batch normalization, thereby significantly decreasing their generalization performance. Even worse, in real-world applications, the difficulty in labeling renders the overfitting issue more severe. Therefore, learning from noisy labels has recently become one of the most active research topics in the machine learning community.

In this thesis, we first reviewed 46 modern deep learning approaches that address the negative consequences of learning from noisy labels. All the methods were grouped into *seven* categories according their underlying strategies and described in chronological order along with their methodological weaknesses. Furthermore, a systematic comparison was conducted using six popular properties used for evaluation in the literature. According to the comparison results, there is *no* ideal method that supports all the required properties; the supported properties varied depending on the category to which each method belonged.

Subsequently, we focused on the two conflicting properties between "loss adjustment" and "sample selection" approaches, namely (1) *heavy noise* and (2) *full exploration*. To support both properties, we proposed a hybrid learning approach called *SELFIE*, which selectively adjusts the losses of refurbishable samples and combines them with those of selected small-loss samples. The main concept of selective loss correction in *SELFIE* successfully reduced the possibility of false correction while exploiting full training data. Therefore, *SELFIE* outperformed the state-of-the-art robust training methods by up to 10.5*pp*.

Despite its great success, we argued that the performance of *SELFIE* becomes considerably worse depending on the type of label noise. The small-loss trick adopted by *SELFIE* for sample selection often misclassified false-labeled samples as true-labeled ones in realistic label noise because the loss distribution of them overlapped closely. Hence, we proposed a novel *two-phase* sample selection approach called *Prestopping*, which achieves noise type robustness by deriving a maximal safe set. Then, we introduced *SELFIE+* by incorporating *Prestopping* with *SELFIE*. According to our extensive evaluation, *Prestopping* and *SELFIE+* significantly outperformed multiple state-of-the-art robust methods including *SELFIE* by up to 18.1*pp*.

For the practical purpose, a limitation of *Prestopping* was the requirement of supervision such as a clean validation set or a known noise rate. In other words, finding the best transition point during the noise-robust period and selecting true-labeled samples during the noise-prone period is very challenging without any supervision. For these challenges, we developed a novel self-transitional learning approach called *MORPH*, which *automatically* finds the best transition point between the noise-robust period and the noise-prone period and, subsequently, *precisely* keeps expanding the maximal safe set during the noise-prone period. In *MORPH*, these two phases were tightly coupled through self-transitional learning, thereby achieving noise type robustness for better practical use. *MORPH* substantially improved robustness by up to 27.0*pp* and efficiency by up to 3.08 times.

Overall, we believe that our work will make a profound impact on robust optimization and significantly raise the practicality of deep learning in real-world scenarios.

7.2 Future Works

This section presents a few challenging but interesting future research directions:

- **Undistinguishable Sample:** Most studies have exploited the small-loss trick to select clean samples from noisy training data. However, difficult-to-learn samples with true labels are not distinguishable from samples with false labels because they exhibit large losses as well. Hence, they are easily misclassified as false-labeled samples and then excluded from the update, although they are informative for learning. This over-cleaning issue becomes more challenging, especially when many ambiguous classes exist in the training data. Distinguishing them from noisy data significantly affects robust deep learning.

- **Complex Label Noise:** Complex label noise, such as instance- and label-dependent label noise, has not been studied extensively. Most studies have mainly focused on two artificial label noises, i.e., symmetric and asymmetric, though these two noises are not very realistic in the real world. Hence, more diverse types of complex label noise should be studied to improve the practicality or generalizability of the algorithm.

- **Multi-label Data:** It is typically assumed that each data sample has only one true label. However, the data sample can be associated with a set of multiple true labels in modern applications, such as semantic scene classification [141] and music categorization [142]. In this setup, the annotator may omit some labels for a sample, thereby resulting in more diverse types of label noise. Therefore, managing label noise in multi-label data is a challenging future direction.

- **Learning Efficiency:** For robust deep learning, most studies have neglected the efficiency of the algorithm because their main goal is to improve the robustness to label noise. For example, maintaining multiple networks or training the DNN multiple rounds is frequently used, though it significantly degrades the learning efficiency of the algorithm. Owing to the rapid increase in the amount of available data, learning efficiency has become critical. Therefore, enhancing the efficiency of the method will significantly increase its usability in the big data era.

Chapter A. ANIMAL-10N Dataset

Because many researchers in robust optimization are facing a lack of real-world noisy data sets, we build a benchmark data set with realistic noises, which we call **ANIMAL-10N**, and publicly release it at https://dm.kaist.ac.kr/datasets/animal-10n in Figure A.1.

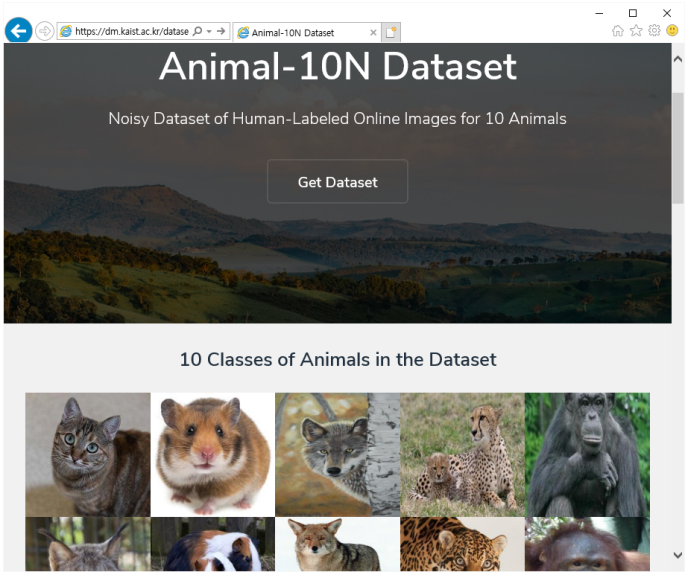


Figure A.1: ANIMAL-10N homepage.

Data Labeling

For human labeling, we recruited 15 participants, which were composed of ten undergraduate and five graduate students, on the KAIST online community. They were educated for one hour about the characteristics of each animal before the labeling process, and each of them was asked to annotate 4,000 images with the animal names in a week, where an equal number (i.e., 400) of images were given from each animal. More specifically, we combined the images for a pair of animals into a single set and provided each participant with *five* sets; hence, a participant categorized 800 images as either of two animals five times. After the labeling process was complete, we paid about US$150 to each participant. Finally, excluding irrelevant images, the labels for 55,000 images were generated by the participants. Please note that these labels may involve human mistakes because we intentionally mixed confusing animals.

Data Organization

We randomly selected 5,000 images for the test set and used the remaining 50,000 images for the training set. Because the test set should be free from noisy labels, only the images whose label matches the search keyword were considered for the test set. Besides, the images are almost evenly distributed to the ten classes (or animals) in both the training and test sets, as shown in Table A.1.

Table A.1: Number of the images for each class in the training and test sets of ANIMAL-10N.

Class	Cat	Lynx	Wolf	Coyote	Cheetah	Jaguar	Chimpanzee	Orangutan	Hamster	Guinea pig	Total
Training Set	5466	4608	5091	4841	4981	4913	5322	4999	4970	4809	50000
Test Set	557	485	423	410	509	524	620	557	440	475	5000

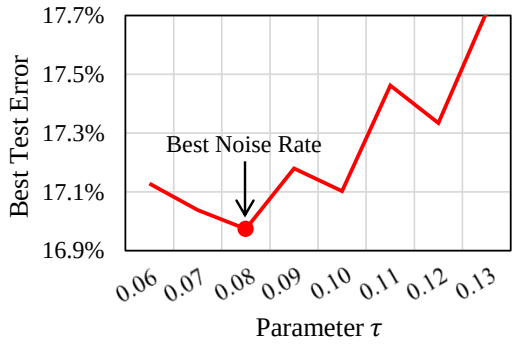


Figure A.2: Best noise rate of the ANIMAL-10N data set obtained by grid search.

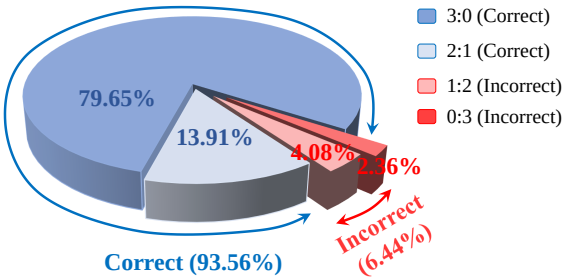


Figure A.3: Proportion of correct and incorrect labels by human inspection.

Noise Rate Estimation

Estimation by Accuracy

Because the ground-truth labels are unknown,[53] we estimated the noise rate τ by the cross-validation with grid search [121, 97]. Following the same configuration as in Section 4.4, we trained DenseNet (L=25, k=12) using *SELFIE* on the $50,000$ training images and evaluated the performance on the $5,000$ testing images. As shown in Figure A.2, the best noise rate was $\tau = 0.08$ from a grid $\tau \in [0.06, 0.13]$ when τ was incremented by 0.01. Therefore, we decided to set $\tau = 0.08$ for ANIMAL-10N.

Estimation by Human Inspection

We also estimated the noise rate τ by human inspection to verify the result based on the grid search. To this end, we randomly sampled 6,000 images and acquired two more labels for each of these images in the same way. Meanwhile, human experts different from the 15 participants carefully examined the 6,000 images to get the ground-truth labels. Comparing the human labels and the ground-truth labels in Figure A.3, the former in the legend represents the number of the votes for the *true* label, and the latter represents the number of the votes for the other label. Because three votes were ready for each image, for conservative estimation, the final human label was decided by majority. Thus, the two cases of 3:0 and 2:1 were regarded as *correct* labeling, and the other two cases of 1:2 and 0:3 were regarded as *incorrect* labeling. Overall, the proportion of incorrect human labels was $4.08 + 2.36 = 6.44\%$ in the sample, and it is fairly close to $\tau = 0.08$ obtained by the grid search.

[53] One might think that the search keyword could be used as the true label, but we found that the search results contained non-negligible erroneous images.

Bibliography

[1] P. Chen, B. Liao, G. Chen, and S. Zhang, "Understanding and utilizing deep neural networks trained with noisy labels," in *Proc. ICML*, 2019.

[2] K.-H. Lee, X. He, L. Zhang, and L. Yang, "CleanNet: Transfer learning for scalable image classifier training with label noise," in *Proc. CVPR*, 2018, pp. 5447–5456.

[3] Q. Li, X. Peng, L. Cao, W. Du, H. Xing, Y. Qiao, and Q. Peng, "Product image recognition with guidance learning and noisy supervision," *Computer Vision and Image Understanding*, p. 102963, 2020.

[4] X. Yu, B. Han, J. Yao, G. Niu, I. W. Tsang, and M. Sugiyama, "How does disagreement help generalization against label corruption?" in *Proc. ICML*, 2019.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NeurIPS*, 2012, pp. 1097–1105.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR*, 2016, pp. 779–788.

[7] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proc. CVPR*, 2017, pp. 3156–3164.

[8] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *Proc. ECIR*, 2016, pp. 45–57.

[9] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, and X. Cheng, "Deeprank: A new deep architecture for relevance ranking in information retrieval," in *Proc. CIKM*, 2017, pp. 257–266.

[10] K. D. Onal, Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. McNamara *et al.*, "Neural information retrieval: At the end of the early years," *Information Retrieval Journal*, vol. 21, no. 2-3, pp. 111–182, 2018.

[11] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. ACL*, 2018, pp. 328–339.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. ACL*, 2019, pp. 4171–4186.

[13] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proc. ACL*, 2015, pp. 959–962.

[14] G. Paolacci, J. Chandler, and P. G. Ipeirotis, "Running experiments on amazon mechanical turk," *Judgment and Decision Making*, vol. 5, no. 5, pp. 411–419, 2010.

[15] V. Cothey, "Web-crawling reliability," *Journal of the American Society for Information Science and Technology*, vol. 55, no. 14, pp. 1228–1238, 2004.

[16] W. Mason and S. Suri, "Conducting behavioral research on amazon's mechanical turk," *Behavior Research Methods*, vol. 44, no. 1, pp. 1–23, 2012.

[17] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Transaction on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2013.

[18] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines." in *Proc. ECAI*, 2012, pp. 870–875.

[19] H. Song, M. Kim, and J.-G. Lee, "SELFIE: Refurbishing unclean samples for robust deep learning," in *Proc. ICML*, 2019, pp. 5907–5915.

[20] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proc. CVPR*, 2015, pp. 2691–2699.

[21] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "Webvision database: Visual learning and understanding from web data," *arXiv preprint arXiv:1708.02862*, 2017.

[22] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *Proc. ICML*, 2017, pp. 233–242.

[23] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. ICLR*, 2017.

[24] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[25] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proc, NeurIPS*, 1992, pp. 950–957.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.

[28] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artificial Intelligence Review*, vol. 22, no. 3, pp. 177–210, 2004.

[29] H. Song, M. Kim, D. Park, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *arXiv preprint arXiv:2007.08199*, 2020.

[30] J. Zhang, X. Wu, and V. S. Sheng, "Learning from crowdsourced labeled data: A survey," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 543–576, 2016.

[31] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. CVPR*, 2017, pp. 1944–1952.

[32] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," in *Proc. NeurIPS*, 2018, pp. 10 456–10 465.

[33] R. Wang, T. Liu, and D. Tao, "Multiclass learning with partially corrupted labels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2568–2580, 2017.

[34] H.-S. Chang, E. Learned-Miller, and A. McCallum, "Active Bias: Training more accurate neural networks by emphasizing high variance samples," in *Proc. NeurIPS*, 2017, pp. 1002–1012.

[35] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *Proc. ICLR*, 2015.

[36] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," in *Proc. ICML*, 2019.

[37] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," in *Proc. ICML*, 2018.

[38] E. Malach and S. Shalev-Shwartz, "Decoupling" when to update" from" how to update"," in *Proc. NeurIPS*, 2017, pp. 960–970.

[39] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proc. ICML*, 2018.

[40] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. NeurIPS*, 2018, pp. 8527–8537.

[41] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia, "Iterative learning with open-set noisy labels," in *Proc. CVPR*, 2018, pp. 8688–8696.

[42] Y. Shen and S. Sanghavi, "Learning with bad training data via iterative trimmed loss minimization," in *Proc. ICML*, 2019.

[43] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox, "Self: Learning to filter noisy labels with self-ensembling," in *Proc. ICLR*, 2020.

[44] Y. Lyu and I. W. Tsang, "Curriculum loss: Robust learning and generalization against label corruption," in *Proc. ICLR*, 2020.

[45] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. CVPR*, 2016, pp. 761–769.

[46] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proc. AAAI*, 2017.

[47] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *Proc. ICLR*, 2015.

[48] L. P. Garcia, A. C. de Carvalho, and A. C. Lorena, "Noise detection in the meta-learning level," *Neurocomputing*, vol. 176, pp. 14–25, 2016.

[49] Y. Yan, Z. Xu, I. W. Tsang, G. Long, and Y. Yang, "Robust semi-supervised learning through label aggregation," in *Proc. AAAI*, 2016.

[50] H. Song, M. Kim, D. Park, and J.-G. Lee, "Prestopping: How does early stopping help generalization against label noise?" *arXiv preprint arXiv:1911.08059*, 2019.

[51] ——, "MORPH: Robust learning by self-transition for handling noisy labels," *arXiv preprint arXiv:2012.04337*, 2020.

[52] H. Song, J.-G. Lee, and W.-S. Han, "PAMAE: Parallel k-medoids clustering with high accuracy and efficiency," in *KDD*, 2017, pp. 1087–1096.

[53] H. Song and J.-G. Lee, "RP-DBSCAN: A superfast parallel dbscan algorithm based on random partitioning," in *SIGMOD*, 2018, pp. 1173–1187.

[54] D. Park, S. Yoon, H. Song, and J.-G. Lee, "MLAT: Metric learning for knn in streaming time series," *arXiv preprint arXiv:1910.10368*, 2019.

[55] D. Park, H. Song, M. Kim, and J.-G. Lee, "TRAP: Two-level regularized autoencoder-based embedding for power-law distributed data," in *The Web Conference*, 2020, pp. 1615–1624.

[56] H. Song, S. Kim, M. Kim, and J.-G. Lee, "Ada-Boundary: Accelerating the DNN training via adaptive boundary batch selection," *Machine Learning*, 2020.

[57] H. Song, M. Kim, S. Kim, and J.-G. Lee, "Carpe diem, seize the samples uncertain "at the moment" for adaptive batch selection," in *CIKM*, 2020.

[58] S. Kim, H. Song, S. Kim, B. Kim, and J.-G. Lee, "Revisit prediction by deep survival analysis," in *PAKDD*, 2020, pp. 514–526.

[59] M. Kim, J. Kang, D. Kim, H. Song, H. Min, Y. Nam, D. Park, and J.-G. Lee, "Hi-COVIDNet: Deep learning approach to predict inbound COVID-19 patients and case study in south korea," in *KDD*, 2020.

[60] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[61] N. Nigam, T. Dutta, and H. P. Gupta, "Impact of noisy labels in learning techniques: A survey," in *Proc. ICDIS*, 2020, pp. 403–411.

[62] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Proc. NeurIPS*, 2013, pp. 1196–1204.

[63] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *Proc. ICLR*, 2017.

[64] V. Wheway, "Using boosting to detect noisy data," in *Proc. PRICAI*, 2000, pp. 123–130.

[65] B. Sluban, D. Gamberger, and N. Lavrač, "Ensemble-based noise detection: Noise ranking and visual performance evaluation," *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 265–303, 2014.

[66] S. J. Delany, N. Segata, and B. Mac Namee, "Profiling instances in noise reduction," *Knowledge-Based Systems*, vol. 31, pp. 28–40, 2012.

[67] D. Gamberger, N. Lavrac, and S. Dzeroski, "Noise detection and elimination in data preprocessing: Experiments in medical domains," *Applied Artificial Intelligence*, vol. 14, no. 2, pp. 205–223, 2000.

[68] J. Thongkam, G. Xu, Y. Zhang, and F. Huang, "Support vector machine for outlier detection in breast cancer survivability prediction," in *Proc. APWeb*, 2008, pp. 99–109.

[69] V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *Proc. ICML*, 2012, pp. 567–574.

[70] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1146–1151, 2013.

[71] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, 2015.

[72] B. Van Rooyen, A. Menon, and R. C. Williamson, "Learning with symmetric label noise: The importance of being unhinged," in *Proc. NeurIPS*, 2015, pp. 10–18.

[73] G. Patrini, F. Nielsen, R. Nock, and M. Carioni, "Loss factorization, weakly supervised learning and label noise robustness," in *Proc. ICML*, 2016, pp. 708–717.

[74] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[75] U. Rebbapragada and C. E. Brodley, "Class noise mitigation through instance weighting," in *Proc. ECML*, 2007, pp. 708–715.

[76] T. Liu, K. Wang, B. Chang, and Z. Sui, "A soft-label method for noise-tolerant distantly supervised relation extraction," in *Proc. EMNLP*, 2017, pp. 1790–1795.

[77] F. O. Kaster, B. H. Menze, M.-A. Weber, and F. A. Hamprecht, "Comparative validation of graphical models for learning tumor segmentations from noisy manual annotations," in *Proc. MICCAI*, 2010, pp. 74–85.

[78] A. Ganapathiraju and J. Picone, "Support vector machines for automatic data cleanup," in *Proc. ICSLP*, 2000.

[79] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proc. ACML*, 2011, pp. 97–112.

[80] C. J. Mantas and J. Abellán, "Credal-C4. 5: Decision tree based on imprecise probabilities to classify noisy data," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4625–4637, 2014.

[81] A. Ghosh, N. Manwani, and P. Sastry, "On the robustness of decision tree learning under label noise," in *Proc. PAKDD*, 2017, pp. 685–697.

[82] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Proc. NeurIPS*, 2018, pp. 8778–8788.

[83] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. ICCV*, 2019, pp. 322–330.

[84] X. Chen and A. Gupta, "Webly supervised learning of convolutional networks," in *Proc. ICCV*, 2015, pp. 1431–1439.

[85] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," in *Proc. ICLRW*, 2015.

[86] I. Jindal, M. Nokleby, and X. Chen, "Learning deep networks from noisy labels with dropout regularization," in *Proc. ICDM*, 2016, pp. 967–972.

[87] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *Proc. ICLR*, 2017.

[88] A. J. Bekker and J. Goldberger, "Training deep neural-networks based on unreliable labels," in *Proc. ICASSP*, 2016, pp. 2682–2686.

[89] B. Han, J. Yao, G. Niu, M. Zhou, I. Tsang, Y. Zhang, and M. Sugiyama, "Masking: A new perspective of noisy supervision," in *Proc. NeurIPS*, 2018, pp. 5836–5846.

[90] J. Yao, J. Wang, I. W. Tsang, Y. Zhang, J. Sun, C. Zhang, and R. Zhang, "Deep learning from noisy image labels with quality embedding," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1909–1922, 2018.

[91] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, 2014.

[92] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," in *Proc. ICLRW*, 2017.

[93] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. ICLR*, 2018.

[94] S. Jenni and P. Favaro, "Deep bilevel learning," in *Proc. ECCV*, 2018, pp. 618–633.

[95] R. Tanno, A. Saeedi, S. Sankaranarayanan, D. C. Alexander, and N. Silberman, "Learning from noisy labels by regularized estimation of annotator confusion," in *Proc. CVPR*, 2019, pp. 11 244–11 253.

[96] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," in *Proc. ICML*, 2019.

[97] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li, "Learning from noisy labels with distillation," in *Proc. ICCV*, 2017, pp. 1910–1918.

[98] M. Dehghani, A. Severyn, S. Rothe, and J. Kamps, "Learning to learn from weak supervision by full supervision," in *Proc. NeurIPSW*, 2017.

[99] ——, "Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision," *arXiv preprint arXiv:1711.00313*, 2017.

[100] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *Proc. ICML*, 2018.

[101] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Learning to learn from noisy labeled data," in *Proc. CVPR*, 2019, pp. 5051–5059.

[102] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-Weight-Net: Learning an explicit mapping for sample weighting," in *Proc. NeurIPS*, 2019, pp. 1917–1928.

[103] Z. Zhang, H. Zhang, S. O. Arik, H. Lee, and T. Pfister, "Distilling effective supervision from severe label noise," in *CVPR*, 2020, pp. 9294–9303.

[104] Y. Ding, L. Wang, D. Fan, and B. Gong, "A semi-supervised two-stage approach to learning from noisy labels," in *Proc. WACV*, 2018, pp. 1215–1224.

[105] J. Li, R. Socher, and S. C. Hoi, "DivideMix: Learning with noisy labels as semi-supervised learning," in *Proc. ICLR*, 2020.

[106] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NeurIPS*, 2014, pp. 2672–2680.

[107] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 447–461, 2015.

[108] M. E. Houle, "Local intrinsic dimensionality I: An extreme-value-theoretic foundation for similarity applications," in *Proc. SISAP*, 2017, pp. 64–79.

[109] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.

[110] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. NeurIPS*, 2017, pp. 1195–1204.

[111] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[112] N. Grira, M. Crucianu, and N. Boujemaa, "Unsupervised and semi-supervised clustering: A brief survey," *A Review of Machine Learning Techniques for Processing Multimedia Content*, vol. 1, pp. 9–16, 2004.

[113] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *Proc. ICLR*, 2017.

[114] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "MixMatch: A holistic approach to semi-supervised learning," in *Proc. NeurIPS*, 2019, pp. 5050–5060.

[115] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits, 1998," vol. 10, p. 34, 1998, http://yann. lecun. com/exdb/mnist.

[116] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[117] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 and CIFAR-100 datasets," 2014, https://www.cs.toronto.edu/ kriz/cifar.html.

[118] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NeurIPSW*, 2011.

[119] A. Karpathy *et al.*, "Cs231n convolutional neural networks for visual recognition," *Neural Networks*, vol. 1, p. 1, 2016.

[120] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *Proc. ECCV*, 2014, pp. 446–461.

[121] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 447–461, 2016.

[122] D. Chandler, *Introduction to modern statistical mechanics*. Oxford University Press, 1987.

[123] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017, pp. 4700–4708.

[124] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[125] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *COLT*, 1998, pp. 92–100.

[126] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in *CVPR*, 2017, pp. 839–847.

[127] D. Krueger, N. Ballas, S. Jastrzebski, D. Arpit, M. S. Kanwal, T. Maharaj, E. Bengio, A. Fischer, and A. Courville, "Deep nets don't learn via memorization," in *ICLRW*, 2017.

[128] M. Li, M. Soltanolkotabi, and S. Oymak, "Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks," in *ICML*, 2020.

[129] C. Zhang, S. Bengio, M. Hardt, M. C. Mozer, and Y. Singer, "Identity Crisis: Memorization and generalization under extreme overparameterization," in *ICLR*, 2020.

[130] G. Pleiss, T. Zhang, E. R. Elenberg, and K. Q. Weinberger, "Detecting noisy training data with loss curves," 2020, https://openreview.net/forum?id=HyenUkrtDB.

[131] M. Toneva, A. Sordoni, R. T. d. Combes, A. Trischler, Y. Bengio, and G. J. Gordon, "An empirical study of example forgetting during deep neural network learning," in *ICLR*, 2019.

[132] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *ICLR*, 2017.

[133] H. Zhang, Z. Zhang, A. Odena, and H. Lee, "Consistency regularization for generative adversarial networks," in *ICLR*, 2020.

[134] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, 2015.

[135] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[136] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *ICLR*, 2016.

[137] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.

[138] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI*, 2017.

[139] B. Zhuang, L. Liu, Y. Li, C. Shen, and I. Reid, "Attend in groups: A weakly-supervised deep learning framework for learning from web data," in *CVPR*, 2017, pp. 1878–1887.

[140] J. Kremer, F. Sha, and C. Igel, "Robust active label correction," in *AISTATS*, 2018, pp. 308–316.

[141] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[142] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.