THE UNIVERSITY OF SYDNEY

Doctoral Thesis

Robust Representation Learning: Understanding the Role of Early Stopping amidst Noisy Labels

Author:
Yingbin Bai

Supervisor:
Senior Lecture Tongliang Liu

Co-Supervisor:
Research Fellow Baosheng Yu

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Sydney AI Centre
School of Computer Science, Faculty of Engineering
University of Sydney

April 28, 2024

Abstract of thesis entitled

Robust Representation Learning: Understanding the Role of Early Stopping amidst Noisy Labels

Submitted by

Yingbin Bai

for the degree of Doctor of Philosophy

at The University of Sydney

in April, 2024

As large models grow in popularity, the demand for data has skyrocketed. This surge in demand has surpassed the availability of high-quality annotated data and even outstripped the volume humans can realistically produce. Consequently, training now often relies on vast amounts of data sourced directly from the Internet, which inevitably includes a significant portion of inaccurately or incorrectly labeled data. It has been observed that deep neural networks have a tendency to overfit to such noisy data, resulting in compromised generalization ability. To mitigate this challenge, a variety of robust strategies have been explored and put forward. Among these, methods based on early stopping have stood out by consistently achieving substantial success.

Despite the pivotal role of early stopping in addressing label noise, a comprehensive understanding of its intricacies remains elusive. This thesis endeavors to bridge this gap, offering several novel contributions that deepen our insight into early stopping. First, it introduces an algorithm specifically designed to harness the strengths of early stopping, enabling the extraction of hard confident examples from noisy datasets. Next, it reveals that the adverse effects of mislabeled examples become more pronounced in the layers nearing the output and proposes a refined early stopping approach via a progressive stopping strategy. Additionally, this thesis

broadens the applicability of early stopping, applying it to self-supervised learning with the goal of overcoming semantic shift challenges. Finally, it provides an empirical examination the constraints of early stopping, seeking solutions for the issues posed by real-world label noise.

Robust Representation Learning: Understanding the Role of Early Stopping amidst Noisy Labels

by

Yingbin Bai
B.E. *University of Sydney*

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy

at

University of Sydney
April, 2024

Declaration

I, Yingbin Bai, declare that this thesis titled, "Robust Representation Learning: Understanding the Role of Early Stopping amidst Noisy Labels", which is submitted in fulfillment of the requirements for the Degree of Doctor of Philosophy, represents my own work except where due acknowledgement have been made. I further declared that it has not been previously included in a thesis, dissertation, or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signed: _____

Date: _____

Authorship attribution statement

This thesis contains materials from the following publications:

- Chapter 3 of this thesis is published as **Yingbin Bai**, and Tongliang Liu. "Me-Momentum: Extracting Hard Confident Examples from Noisily Labeled Data." *International Conference on Computer Vision*, 2021.
  I embarked on designing the study, crafting the algorithm from scratch, conducting the experiments, and drafting the manuscript.

- Chapter 4 of this thesis is published as **Yingbin Bai**, Erkun Yang, Bo Han, Yanhua Yang, Jiatong Li, Yinian Mao, Gang Niu, and Tongliang Liu. "Understanding and Improving Early Stopping for Learning with Noisy Labels." *Neural Information Processing Systems*, 2021.
  I pinpointed the research issue, introduced a groundbreaking approach to address it, carried out the necessary experiments, and crafted the preliminary versions of the manuscript.

- Chapter 5 of this thesis is published as **Yingbin Bai**, Erkun Yang, Zhaoqing Wang, Yuxuan Du, Bo Han, Cheng Deng, Dadong Wang, and Tongliang Liu. "RSA: Reducing Semantic Shift from Aggressive Augmentations for Self-supervised Learning." *Neural Information Processing Systems*, 2022.
  I designed the study, developed the algorithm, conducted the experiments, and composed the initial drafts of the manuscript.

- Chapter 6 of this thesis is published as **Yingbin Bai**, Zhongyi Han, Erkun Yang, Jun Yu, Bo Han, Dadong Wang, and Tongliang Liu. "Subclass-Dominant Label Noise: A Counterexample for the Success of Early Stopping." *Neural Information Processing Systems*, 2023.
  I identified the research problem, proposed a novel method to tackle it, executed the experiments, and wrote the drafts of the manuscript.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Student Name: Yingbin Bai

Signed: _____

Date: _____

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Supervisor Name: Tongliang Liu

Signed: _____

Date: _____

This thesis is dedicated to my loving family:
my parents, wife, and son.
I express my deepest gratitude for their unwavering love and support.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Tongliang Liu, for his invaluable advice, continuous support, and patience during my PhD study. His enormous knowledge and great experience have encouraged me in all the time of my academic research and daily life. I also extend my sincere gratitude to Dr. Dadong Wang for his generous support and thorough guidance. Without the Data61 scholarship, I cannot completely dedicate myself to research.

Additionally, I would like to convey my appreciation to my fellow lab students: Yu Yao, Songhua Wu, Xuefeng Li, Feiyu Wang, Huaxi Huang, Xiaobo Xia, Chaojian Yu, Zhaoqing Wang, and others. Their assistance and constructive discussions have immensely enriched my PhD journey.

Last but not least, I extend my heartfelt thanks to my family. Their unwavering love and support have been my strength and motivation throughout this journey.

Yingbin Bai
University of Sydney
April 28, 2024

Contents

Chapter 1

Introduction

As the emergence and rapid development of large-scale models, underscore a critical trend: an explosive increase in the demand for diverse and voluminous datasets[45, 100, 29, 110]. This hunger for data, voracious and unrelenting, has swiftly surpassed the pool of high-quality annotated data, traditionally curated and labeled with meticulous precision [101, 99]. In an effort to meet this overwhelming demand, researchers and developers are exploring beyond conventional data sources, tapping into the expansive reservoir of the Internet [15, 104]. This effort yields an abundance of cheap, raw data, yet it is fraught with challenges. The vast quantities of information sourced come with a plethora of inaccuracies and inconsistencies, especially in data labeling [130, 139, 103].

Deep neural networks (DNNs), with their impressive learning capabilities, are not immune to the pitfalls of label noise [2, 73, 41]. Numerous studies have revealed that DNNs have a tendency to overfit to the data they are presented with [38, 4, 33]. This overfitting becomes particularly problematic when noisy data is involved, which can significantly undermine the DNNs' ability to generalize effectively to new, unseen data [2, 77, 80]. Such deterioration in generalization capability poses another substantial challenge to developing real-world applications, escalating the requirement for datasets that are not merely large in scale but also superior in labeling quality [15, 67].

To combat the issue presented by label noise, the research community has engaged in an intensive quest for solutions, and a variety of methodologies have been carried out [56, 138, 80, 12, 134]. These can be broadly

categorized into two streams: model-based and model-free methods [135, 127]. Model-based methods combat the negative impacts of label noise by modeling the noise distribution [97, 128, 122]. With an understanding of this distribution, a transition matrix can be constructed to facilitate the transfer of knowledge from the noise distribution to the clean distribution [131, 135, 68]. The primary advantage of this approach is its robust theoretical foundation, which ensures network convergence provided certain assumptions are met [73, 131, 75]. In real-world scenarios, however, the noise distribution is typically unknown and challenging to estimate with precision. Consequently, when a model-based method relies on an inaccurate transition matrix, its effectiveness is substantially diminished [124, 138, 65].

In contrast, model-free methods tackle the challenge of label noise by identifying examples that are likely to be correctly labeled, and subsequently designate these as confident examples [126, 42, 108]. This kind of approaches leverages a phenomenon observed in DNNs known as memorization effect, wherein DNNs tend to first fit to the majority of patterns before overfitting to the minority [140, 4]. Given that clean samples exhibit common patterns while mislabeled examples typically deviate, the network will initially fit the clean examples, and then memorize mislabeled examples. By stopping the training process early, one can thus ensure a network that demonstrates strong generalization on the clean dataset [42, 91, 121]. This observation has inspired a plethora of approaches that utilize early stopping, which have shown consistent and significant success across noisy datasets [65, 57, 72].

Despite the pivotal role of early stopping in addressing label noise, a comprehensive understanding of its intricacies remains elusive. Key questions remain regarding the efficacy of early stopping in extracting hard examples from noisy datasets and determining the optimal stopping points to achieve a better initial network. Moreover, there is uncertainty about the applicability of early stopping beyond supervised learning and whether it can adequately address all types of label noise. To answer these questions, the ensuing chapters will dissect the subject through four distinct aspects: strengths, advancements, applicability, and limitations. Specifically, the

analysis will be organized as follows:

Chapter 2. Preliminaries. In this chapter, the problem of learning with noisy labels (LNL) is articulated. Subsequently, an overview of memorization effects and early stopping is provided.

Chapter 3. Strengths of Early Stopping. This chapter presents an novel algorithm that leverages the inherent strengths of early stopping, specifically devised to overcome the challenges associated with the extraction of hard confident examples from datasets contaminated by label noise. Such examples are of paramount importance; they hold essential information that, when accurately identified and utilized, can markedly enhance the efficacy of the training process.

- The contributions in this Chapter are included in:

> **Yingbin Bai**, and Tongliang Liu. "Me-Momentum: Extracting Hard Confident Examples from Noisily Labeled Data." *International Conference on Computer Vision*, 2021.

Chapter 4. Advancement of Early Stopping. In this chapter, we first present evidence demonstrating that the latter layers in a deep neural network exhibit much more sensitivity to label noise, while the former layers are quite robust. Building on these findings, we improve the early stopping technique by partitioning a DNN into different parts and applying progressive training strategies.

- The contributions in this Chapter are included in:

> **Yingbin Bai**, Erkun Yang, Bo Han, Yanhua Yang, Jiatong Li, Yinian Mao, Gang Niu, and Tongliang Liu. "Understanding and Improving Early Stopping for Learning with Noisy Labels." *Neural Information Processing Systems*, 2021.

Chapter 5. Applicability of Early Stopping. In this chapter, we investigate the applicability of early stopping techniques in the realm of self-supervised learning. Our approach offers a novel solution for addressing the issues of semantic shift induced by aggressive data augmentation.

- The contributions in this Chapter are included in:

> **Yingbin Bai**, Erkun Yang, Zhaoqing Wang, Yuxuan Du, Bo Han, Cheng Deng, Dadong Wang, and Tongliang Liu. "RSA: Reducing Semantic Shift from Aggressive Augmentations for Self-supervised Learning." *Neural Information Processing Systems*, 2022.

Chapter 6. Limitations of Early Stopping. This chapter conducts an empirical analysis of the limitations associated with early stopping, revealing its ineffectiveness against subclass-dominant label noise (SDN). To combat this specific type of label noise, we introduce NoiseCluster, a novel methodology devised to identify and correct SDN.

- The contributions in this Chapter are included in:

> **Yingbin Bai**, Zhongyi Han, Erkun Yang, Jun Yu, Bo Han, Dadong Wang, and Tongliang Liu. "Subclass-Dominant Label Noise: A Counterexample for the Success of Early Stopping." *Neural Information Processing Systems*, 2023.

Chapter 7. Conclusion. In this final chapter, we conclude our thesis by summarizing our key findings, reflecting on the implications of our research, and discussing potential directions for future research to build on our work.

Chapter 2

Preliminaries

In this chapter, we begin by delineating the research problem of learning with noisy labels (LNL). Following this, we present an overview of the memorization effects and early stopping method.

Problem Settings. Let D denote the joint probability distribution of a pair of random variables (X, Y), where X is the feature instances and Y represents the variable of correct labels. The label space is defined as $\{1, ..., C\}$, with C indicating the number of classes. In practical scenarios, data often experiences mislabeling due to a range of factors, such as the annotator's lack of expertise in specific data domains or inadvertent oversight of certain instances. Considering these factors, let $\widetilde{Y}$ represent the variable of noisy labels, which includes a mix of both correct and erroneous labels. Consequently, the noisy distribution can be denoted as $\widetilde{D}$, where pairs $(X, \widetilde{Y})$ are drawn.

In real-world scenarios, the complexity of studying and addressing data corruption arises from various factors, including inherent imperfections within models, instances of human error, and gaps in domain-specific knowledge. These multifaceted influences significantly contribute to the inherent challenges of investigating data corruption. Furthermore, for research purposes, label noise is typically categorized into three types, ranging from easy to hard: random classification noise (RCN), class-conditional label noise (CCN), and instance-dependent label noise (IDN).

RCN manifests as the flipping of clean labels with randomness and a consistent rate, underscoring its independence from the data and associated label information. This means that errors occur randomly and uniformly

across the dataset, with no correlation to the actual characteristics of the instances. In contrast, CCN's generation process is determined exclusively by class information without considering the specific details of individual instances. This implies that mislabeling occurs solely based on the class to which an instance belongs, irrespective of its unique features or context within instances. Finally, IDN encompasses the most comprehensive category, wherein noise generation is influenced by factors pertaining to both the class and the specific attributes of individual instances. In this scenario, mislabeling is not solely dictated by the class distribution but also by the intrinsic properties or characteristics of each instance. This leads to a more intricate and nuanced form of label noise, reflecting the diverse nature of real-world data.

Memorization Effects. Deep neural networks are widely recognized as universal approximators, endowed with the ability to represent functions of arbitrary complexity [38, 28, 49]. When provided with sufficient capacity, these networks can memorize all types of data, including mislabeled ones [4, 140]. However, the learning of correctly labeled and mislabeled data does not occur simultaneously. Typically, these networks display a tendency to initially grasp simple patterns, subsequently advancing to more complex ones [102, 66]. In datasets with label noise, correctly labeled data usually forms the majority, exhibiting common patterns, whereas mislabeled data, being in the minority, tends to have more varied patterns. As a result, deep neural networks tend to initially align with the patterns of correctly labeled data, gradually adapting to include the mislabeled data as they encounter increasingly complex patterns[66, 42, 108].

Early Stopping. To counteract the adverse effects of noisy labels, a variety of heuristic methods exploit the memorization effects of deep neural networks [56, 108, 138]. These methods involve pausing the training process before the network begins overfitting to mislabeled examples, a technique known as early stopping [106, 66, 72]. After halting training, examples that are likely to be correctly labeled are identified within the noisy datasets and then treated as confident examples for further training [93, 102, 124].

Among the various criteria for selecting confident examples, the small-loss trick stands out [42, 138, 102]. Han et al. [42] observed that correctly

labeled examples typically exhibit smaller losses compared to mislabeled ones. This approach becomes more effective when employed with multiple networks [65, 57]. Another effective criterion is the utilization of the network's own predictions [56, 108]. This process entails comparing these predictions with the original noisy labels, with aligned examples being classified as confident. Despite the absence of theoretical guarantees, early stopping based methods have proven to be effective in real-world scenarios, primarily due to their ability to operate without making specific assumptions about the noise distribution [72, 65, 57].

Chapter 3

Strengths of Early stopping

In the domain of learning with label noise, various state-of-the-art methods have employed the memorization effect, employing strategies like utilizing small loss to isolate confident examples. However, these methods often neglect the critical role of hard examples in shaping the decision boundary. Hard examples, which lie close to the boundary, are crucial for a robust model. Remarkably, existing methodologies fail to address the extraction of hard confident examples from noisy datasets.

In this chapter, we harness the inherent strength of early stopping and introduce a deep learning paradigm tailored precisely for addressing the label noise problem. To extract hard confident examples that contain non-simple patterns and are entangled with the inaccurately labeled examples, we borrow the idea of momentum from physics. Specifically, we alternately update the confident examples and refine the classifier. Note that the extracted confident examples in the previous round can be exploited to learn a better classifier and that the better classifier will help identify better (and hard) confident examples. We call the approach the "***Momentum** of **Me**morization*" (Me-Momentum). Empirical results on benchmark-simulated and real-world label-noise data illustrate the effectiveness of Me-Momentum for extracting hard confident examples, leading to better classification performance.

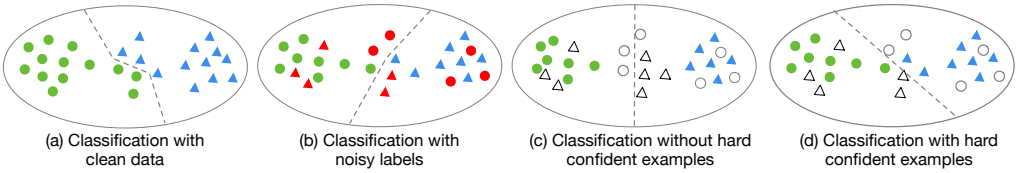


Figure 3.1: The illustration of the influence of hard (confident) examples in classification. Circles represent positive examples while triangles represent negative examples. Green and blue denote examples with accurate labels while red presents examples with incorrect labels. Blank circles and triangles represent unextracted data. (a) shows an example of classification with clean data. (b) shows noisy examples, especially those close to the decision boundary, will significantly degenerate the accuracy of the classifier. (c) shows confident examples help learn a fairly good classifier. (d) shows that hard confident examples are essential to train an accurate classifier.

3.1 Motivations and Contributions

A promising approach to tackling the issue of label noise involves the extraction of examples that possess clean labels—*confident examples*— [93, 111, 91, 30, 118, 102, 125, 126]. The idea is that compared with the original noisy training data, the extracted examples are less noisy and thus will lead to a classifier with better performance. Given only noisy data, state-of-the-art methods exploit the *memorization effect* [140, 5] to extract confident examples. The memorization effect will enable deep neural networks to first learn patterns that are shared by a majority of training examples. As clean labels are of majority in each noisy class [90, 130], deep neural networks would therefore first fit training data with clean labels, and then gradually fit the examples with incorrect labels [18]. Therefore, early stopping [66, 106] and the small loss trick [56, 42, 137] can be used to exploit confident examples.

Examples that are close to the decision boundary are called *hard examples*. As illustrated in Figure 3.1, hard (confident) examples play an important role in shaping the decision boundary. It has also been widely studied in the traditional classification problem that hard examples are essential to train accurate classifiers [116, 10, 51, 47]. Notwithstanding the importance of hard confident examples, none of the existing methods studies how to extract hard confident examples from noisy data. Note that

extracting hard confident examples is non-effortless. Since hard examples are often of a small proportion and contain less discriminative information compared with the easy ones (these that are far away from the decision boundary), they are often entangled with inaccurately labeled examples in the procedures of extraction.

In this chapter, by alternately updating the confident examples and refining the classifier, we propose a deep learning paradigm that is able to extract hard confident examples from the noisy training data, leading to better classification performance. Specifically, the idea is similar to the usage of momentum from physics. As stated in the statistical learning theory, with better training data, a better classifier can be obtained [88]. We can then think of the classifier as a particle traveling through the hypothesis space, getting acceleration from the confident data. Classifiers with better performance can be achieved by properly exploiting the previously extracted confident examples. This is similar to the momentum trick used in optimization that previous gradient information can be used to escape local minimum and achieve fast convergence rates [107][1]. At a high level, the proposed method is built on the memorization effect of deep neural networks and on the intuition that better confident examples will result in a better classifier and that a better classifier will identify better confident examples (and hard confident examples). The proposed method is therefore called the Momentum of Memorization (Me-Momentum).

We conduct experiments to show the effectiveness of the proposed Me-Momentum on noisy versions of *MNIST*, *CIFAR10*, *CIFAR100*, and a real-world label noise dataset *Clothing1M*. Specifically, on *MNIST* and *CIFAR*, we generate class-dependent and instance-dependent label noise and visualize the extracted hard confident examples, which justifies why Me-Momentum consistently outperforms the baseline methods.

[1]In optimisation, the parameter vector can be thought of as a particle traveling through the parameter space, getting acceleration from the gradient of the loss. The momentum trick demonstrated that the gradient in the previous update can help escape local minimum and achieve fast convergence rates.

3.2 Methodology

In this section, by specifying the proposed method of *momentum of memorization* (Me-Momentum; summarized in Algorithm 1), we would like to detail how to accomplish extracting hard confident examples and boosting the classification performance. At a high level, by alternately updating the confident examples and refining the classifier, Me-Momentum fulfills a positive cycle that better confident examples will result in a better classifier and that a better classifier will identify better confident examples. Specifically, Me-Momentum has two loops, i.e., an inner loop and an outer loop. In the inner loop, Me-Momentum alternates update of the confident examples and the classifier (Steps 2 and 3). However, the inner loop continually refines a classifier and thus depends heavily on the initialization of the classifier (Step 1). It may lead to the memorization of noisy labels and the inferiority of sample-selection bias. To handle this problem, the outer loop re-initializes the classifier (Step 5) while it maintains the previously extracted confident examples.

There are some points to be clarified for the proposed Algorithm 1:

Q1. How to initialize a good classifier in Step 1?

Q2. How to extract confident examples in Step 2?

Q3. How to validate the learned classifiers in Steps 3 and 5 without a clean validation set?

Q4. What are hard confident examples?

Q5. Why can hard confident examples be extracted?

Q6. Why the proposed method is called Me-Momentum?

To answer the *first question*, we would like to mention that the aim of the initialization in Step 1 is to initialize a good classifier for the positive cycle: a better classifier will identify better confident examples and better confident examples will result in a better classifier. A good candidate should have a fairly high classification accuracy, e.g., higher than random guessing. Otherwise, the positive cycle cannot be invoked. Fortunately, the initialization can be made by exploiting the memorization effect of deep neural

networks that would first fit clean data [5, 140]. Note that this memorization effect is independent of training optimization or network backbones [5]. Specifically, we use the early stopping trick. For easy understanding, we would like to introduce a definition of *high-peak*. A noisy validation accuracy at the i-th epoch is called an i-th high-peak if it achieves the highest accuracy in the epoch range $\{1, \ldots, i\}$. Assume the i-th and j-th high-peaks occur next to each other, having noisy validation accuracies of a and b, respectively. The training early stops if $(b-a)/(j-i) \leq \tau$, where τ is a hyper-parameter. In the experiments, we set $\tau = 0.1$, which empirically works well across all datasets. In Section 3.3.4, we compare the difference between the early stopping method and the traditional validation method. We also study the sensitivity of the hyper-parameter.

The answer to the *second question* is closely related to the memorization effect. Note that the classifier initialized in Step 1 would fit the clean data well but not the incorrectly labeled data because of the memorization effect and early stopping. Therefore, we can treat the training examples whose noisy labels are identical to the ones predicted by the classifier obtained in Step 1 as confident examples. This also applies for the classifiers in Step 3 to extract confident examples, which are iteratively trained by employing the updated confident data. Note that there are some other feasible methods to extract confident examples, e.g., extract those who have a large class posterior.

In Step 3, we aim to learn a better classifier compared with the one in the previous round. This can be achieved because of two reasons: (1) we initialize the network with the parameters of the classifier learned in the previous round; (2) we have a better set of confident examples as the training sample.

This starts the positive cycle that better confident examples will result in a better classifier and that a better classifier will identify better confident examples. The *third question* is essential for identifying the classifiers in the cycle. Note that the accurately labeled examples are always assumed to be dominant in each class in the community of learning with noisy labels [90, 74, 42]. Otherwise, the true class label cannot be identified by only exploiting the noisy data. This assumption implies that the performance

on the noisy validation set (split from the noisy training set) and these on test set are positively correlated. The noisy validation set could be used as a surrogate to validate the classifiers if no clean validation set is available. We therefore validate the classifiers in Steps 3 and 5 with the highest noisy validation accuracies during the training. Empirical results show that it works well.

Algorithm 1: Me-Momentum

Input: Noisy training data, noisy validation data, iteration number N_{inner} and N_{outer};
Output: Extracted confident examples and the classifier;
1: **Initialize** a classifier f_0 by using the noisy training data and early stopping; //memorization effect
for $i = 1, \ldots, N_{outer}$ **do**
  for $j = 1, \ldots, N_{inner}$ **do**
    2: **Update** the extracted confident examples;
      //i.e., the training examples whose noisy labels are identical to the ones predicted by f_{j-1}
    3: **Obtain** the classifier f_j;
      //initialize the network by using the parameters of f_{j-1} and train it by employing confident examples; the classifier f_j will be chosen with the highest noisy validation accuracy throughout the training procedure
    4: **Break** and output f_{j-1} if the highest validation accuracy is non-increasing in the loop;
  end
  5: **Re-initialize** a classifier f_0;
    //randomly initialize the network and train it by using confident examples; the classifier f_0 will be chosen with the highest noisy validation accuracy throughout the training procedure
  6: **Break** and output f_{j-1} if the highest validation accuracy is non-increasing in the loop;
end

To answer the *fourth question*, we define the hard examples by exploiting the memorization effect of deep neural networks, i.e., deep neural networks first fit the majority (or easy) patterns and then the minority (or hard) patterns. Specifically, hard examples are those which contain minority (or hard) patterns. Note that hard patterns are usually entangled with incorrect labels.

Following the previous question, we answer the *fifth question*. By simply exploiting the memorization effect, extracting confident examples with hard patterns is difficult. However, by using the proposed Me-Momentum method, we could extract hard examples. Due to the memorization effect of deep neural networks, the model first fits the simple patterns, i.e., examples with simple features, some of which also have hard features. Then deep neural networks can learn hard patterns from the fitted examples, which makes it possible for deep neural networks to extract hard confident examples from those examples entangled with incorrect labels. A visualization is shown in Figure 3.3.

To answer the *sixth question*, we would like to first mention that the proposed method heavily relies on the memorization effect of deep neural networks. Specifically, in Step 1, a classifier is initialized by exploiting the memorization effect via early stopping, which is used to identity confident examples. Later, the classifier and confident examples are iteratively refined and updated, respectively, which is the positive cycle we have mentioned before. Note that this cycle also depends on the memorization effect to update confident examples and refine classifiers. Our method is named as *momentum of memorization* (Me-Momentum) because it uses the trick of momentum to better exploit the memorization effect. Specifically, we can think of the classifier as a particle traveling through the hypothesis space, getting acceleration from the updated extracted confident data. We exploit the previously extracted confident examples to help learn a better classifier, training the network by using the confident examples extracted in the previous round. The impact of the confident examples will increase as we continue extracting more confident examples.

Relation to existing work: The strategy of alternatively optimizing the classifier and updating the training examples is not new for dealing with label noise. For example, Joint Optim [108], Co-teaching [42, 137], and SELF [91] are similar to ours. Specifically, Joint Optim and Co-teaching update the classifier with one step of stochastic gradient descent while SELF and the proposed method refine the classifier to be optimal with respect to the extracted confident examples. However, existing methods have not focused on extracting hard confident examples and thus are substantially

different from this chapter because they neglected the importance of avoiding the accumulated error caused by the single initialization of the classifier and the sample-selection bias. Experiments (e.g., Figures 3.2 and 3.3) show that Me-Momentum with the outer loop part (i.e., re-initialization of the classifier) contributes significantly to extracting hard confident examples and achieving high label precision.

Me-Momentum is similar to curriculum learning as it also learns from easy to difficult. However, curriculum learning needs a predefined curriculum (sample weighting scheme), e.g., assigning big/small weights for confident/noisy data. If the curriculum is not available, some clean data is required to learn a mentornet to provide a curriculum [56] or a latent variable could be introduced by self-paced learning [61] to learn a curriculum. Differently, Me-Momentum is only based on noisy data and does not explicitly learn a curriculum. Me-Momentum also has a similar flavor to active learning which tends to choose and label hard examples to learn from at each iteration. However, for active learning, no label information is available before the data is chosen while Me-Momentum has noisy labels and needs to consider the side-effect of label noise.

3.3 Experiments

Datasets: To verify the effectiveness of the proposed method, we do experiments on datasets with both synthetic and real-world label noise. Specifically, we manually corrupt *MNIST* [62], *CIFAR10*, and *CIFAR100* [59] with class-dependent label noise and instance-dependent label noise. We detail how to generate class-dependent and instance-dependent label noise in Appendix 1. We employ the real-world noisy dataset *Clothing1M* [130]. These datasets have been widely used in studies with noisy labels [42, 108, 128].

For *MNIST*, *CIFAR10*, and *CIFAR100*, we leave out 10% of the noisy training data as noisy validation data. *Clothing1M* contains one million noisy training images, which are crawled from shopping websites, labeling by surrounding text. Almost all existing work uses the $14k$ clean validation data in their experiments. To verify the robustness of the proposed method,

we also employ noisy validation data in our experiments. Specifically, $100k$ noisy data are randomly left as noisy validation data and the remaining $900k$ noisy data as the training data.

Baselines: Me-Momentum is compared against the following state-of-the-art approaches. (1) Statistically consistent methods: Forward [96], T-revision [128], and DMI [131]; (2) Statistically inconsistent methods: MentorNet [56], Co-teaching [42], Joint Optim [108], SELF [91], CDR [124], DivideMix [65], ELR+ [72] where MentorNet, Co-teaching, SELF, and DivideMix use the idea of extracting confident examples by employing the small loss trick. Note that DivideMix and ELR+ employ a semi-supervised approach for unconfident examples, which gives them an advantage for synthetic datasets whose numbers of confident examples are limited. Therefore, we only compare our method with them on real-world datasets.

Network structure and optimization: All the methods are implemented by PyTorch v1.5. For the experiments on *MNIST*, *CIFAR10*, and *CIFAR100*, we set $N_{\text{inner}} = 20$, $N_{\text{outer}} = 3$, 100 epochs for each inner loop and follow the settings of T-revision [128]. Specifically, LeNet-5, ResNet-18, and ResNet-34 networks are used for *MNIST*, *CIFAR10*, and *CIFAR100* respectively. We use SGD with momentum 0.9, weight decay 10^{-4}, batch size 128, and an initial learning rate of 10^{-2}, divided by 10 after the 40-th epoch and 80-th epoch respectively (we fix the learning rate of 10^{-2} for the early stopping method). Data augmentation is used with horizontal random flips and 32×32 random crops after padding 4 pixels on each side.

For *Clothing1M*, a ResNet-50 is used. To show the effectiveness of the proposed method, we do experiments by randomly initializing the network and pre-training it by employing *ImageNet*, respectively. As the noisy training sample contains a large number of examples, we set $N_{\text{inner}} = 6$ and $N_{\text{outer}} = 3$ and 5 epochs for each inner loop. We use SGD with momentum 0.9, weight decay 10^{-3}, batch size 32, with a learning rate of 5×10^{-3}, and divided it by 10 at the 3-rd and 5-th round in the inner loop. For each outer loop, the model will be randomly re-initialized (or replaced by a pre-trained one). The learning rate will be reset to 5×10^{-3}. For data augmentation, all images are resized to 256×256, horizontal random flipped, and 256×256 random cropped with padding 32 pixels on each side. Note that due to

the page limit, some complementary experiments to Sections 3.3.1 and 3.3.2 and the comparison with SELF are put in Appendix 2. The code is available at https://github.com/tmllab/Me-Momentum.

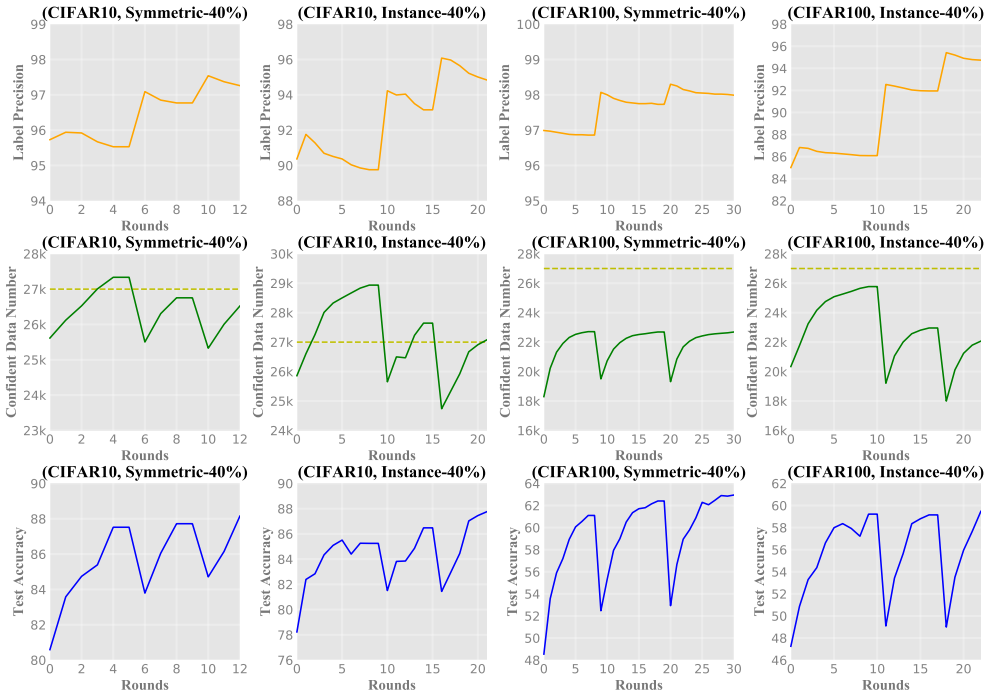


Figure 3.2: We call one update of the classifier and the extracted confident examples as one round. We illustrate how the label precision of the extracted confident examples, the number of the extracted confident examples, and the classification accuracy of the classifier trained by using the extracted confident examples change during the training of Me-Momentum. We have three distinct peaks in these figures because we have set $N_{\text{outer}} = 3$ and the classifiers are re-initialized in the outer loop. The dash lines in the second row indicate the number of clean labels in the noisy training data.

3.3.1 Verify Momentum of Memorization

In this section, we discussed that Me-Momentum is implemented by fulfilling the positive cycle that better confident examples will result in a better classifier and that a better classifier will identify better confident examples. In this subsection, we will empirically verify this positive cycle, which can be done on the synthetic datasets as we have their ground-truth labels.

In Figure 3.2, we can see that in the inner loops (e.g., rounds 0-5, rounds 6-9, and rounds 10-12 in the first column of figures represent three

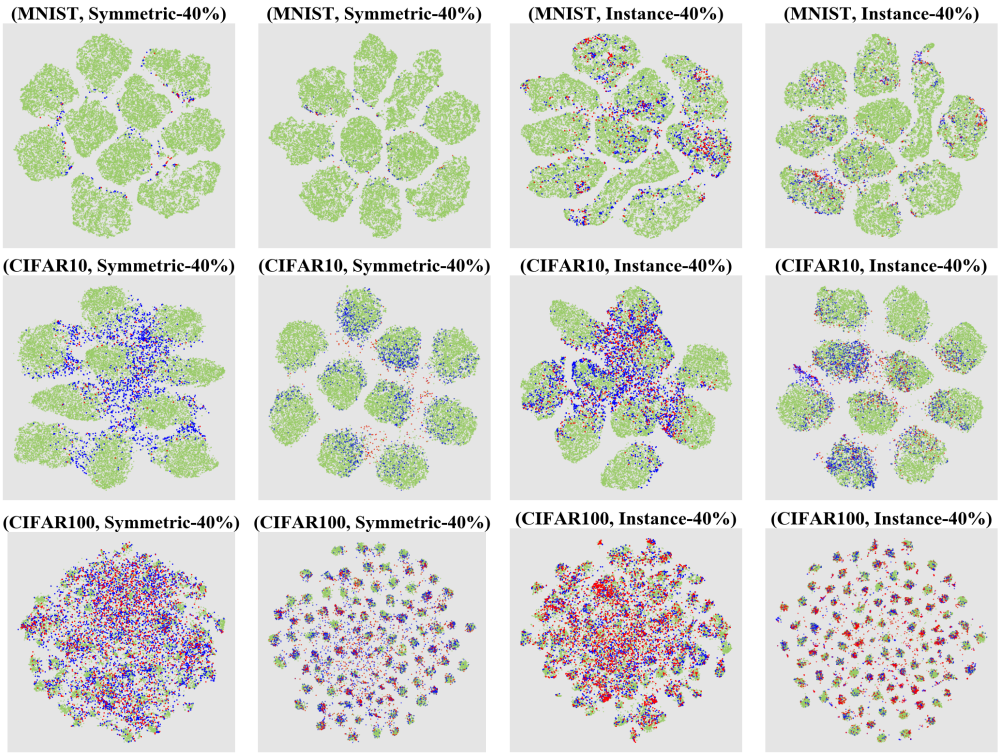


Figure 3.3: Visualization of the extracted confident examples. The first and third columns are about the confident data extracted in the first run of the inner loop; while the second and the fourth columns are about the confident data extracted in the outer loop. Specifically, green dots represent the data selected in the first round. Blue and red dots represent the newly extracted data in the middle and the end rounds respectively. Large figures for *CIFAR100* are provided in the supplementary material.

inner loops respectively), the classification accuracy generally increases (note that the figures are not smooth because the classifiers are tested on the unseen test data) and the number of extracted confident examples clearly increases (although their label precision decreases slightly). We can also see that in the outer loops (e.g., rounds 0, 6, and 10 in the first column of the figures consist of an outer loop), the classification accuracy clearly increases and the label precision of the extracted confident examples clearly increases (although the number of extracted confident examples decreases slightly). This implies that compared with previous classifiers and extracted confident examples, better ones are obtained, which empirically justifies the positive cycle. Note that the classification accuracy in the outer loop is low because the models are re-initialized.

Figure 3.2 also shows the importance of the outer loop of Me-Momentum. We can see that the label precision of the extracted confident examples slightly decreases in the inner loops. This is because the deep model gradually memorizes the noisy labels as we continually refine it. This issue can be handled by re-initializing the deep model in the outer loop. Specifically, we can see from Figure 3.2 that by re-initializing the model in the outer loop, the label precision of the extracted confident examples increases significantly. Although the number of the extracted confident examples decreases, the overall quality of the extracted confident examples is increasing as evidenced by the increase of the classification accuracy of the classifiers trained on the extracted confident data. Note that the low data quality in the first run of the inner loop also justifies that a single deep model initialization may lead to sample selection bias.

There are interesting observations of the proposed method that the number of extracted confident examples is close to the number of accurately labeled data in the training set and that the label precision of the extracted confident examples is quite high, i.e., almost all are above 90%. This empirically proves that Me-Momentum is powerful in extracting confident examples. In the next subsection, we will visualize that Me-Momentum is also good at extracting hard confident examples.

3.3.2 Visualize Hard Confident Examples

To justify that Me-Momentum is able to extract hard confident examples, we visualize the extracted confident examples by employing t-SNE [83]. Specifically, we show how the confident examples are progressively extracted in the inner and outer loops. The results are shown in Figure 3.3, where green, blue, and red dots represent confident examples extracted at the beginning, middle, and end rounds of the loops, respectively. On the datasets of *MNIST* and *CIFAR10*, we can clearly see that the blue and red dots are mostly located at the boundaries of the clusters of green dots. Although the figures of *CIFAR100* are small, we can also clearly see that there are lots of blue and red dots which are outside of the green clusters in the second and fourth figures. This supports and justifies our claim that

Table 3.1: Means and standard deviations of classification accuracy on *MNIST*

Methods	Sym-20	Sym-40	Inst-20	Inst-40
Cross-Entropy	97.88±0.27	97.41±0.18	97.61±0.28	92.93±0.81
MentorNet	96.57±0.18	96.16±0.49	94.66±0.35	88.51±0.36
Co-teaching	97.22±0.18	94.64±0.33	95.37±0.08	90.06±0.81
Forward	98.22±0.08	96.71±0.16	95.89±0.12	88.95±2.47
Joint Optim	98.58±0.15	98.12±0.06	98.10±0.14	92.00±1.39
DMI	98.92±0.11	98.63±0.11	98.75±0.11	97.58±0.82
T-revision	98.91±0.04	98.42±0.47	97.12±0.09	94.89±0.66
CDR	98.76±0.07	98.40 ±0.17	98.18±0.09	93.43±1.12
Ours	**98.94±0.13**	**98.66±0.07**	**98.96±0.06**	**98.11±0.35**

Me-Momentum is able to extract hard confident examples (those are close to the decision boundary).

Comparing the extracted results of the first run of the inner loop (the first and third columns) with those of the outer loop (the second and fourth columns), we can find that the cluster boundaries in the latter are more clear. This further justifies that why better classification performance can be obtained by re-initialization in the outer loop. Comparing the confident examples extracted on the class-dependent label noise datasets with those on the instance-dependent label noise datasets, we can observe that the proposed method is not sensitive to the type of label noise and can work well on the most general instance-dependent label noise cases.

3.3.3 Classification Accuracy

Synthetic data To evaluate the classification performance of Me-Momentum, we first conduct experiments on *MNIST*, *CIFAR10*, and *CIFAR100* with class-dependent and instance-dependent label noise. Each trial is repeated five times. The results are presented in Tables 3.1, 3.2, and 3.3, respectively. Me-Momentum consistently outperforms the baselines. Specifically, *CIFAR100* is the most challenging one among the three datasets. Me-Momentum outperforms the baselines by a clear margin across all the settings as shown in Table 3.3. Note that the performance gain in Me-Momentum is caused by the improvement of the quality of the extracted confident examples.

Table 3.2: Means and standard deviations of classification accuracy on *CIFAR10*

Methods	Sym-20	Sym-40	Inst-20	Inst-40
Cross-Entropy	85.00±0.43	79.59±1.31	85.92±1.09	79.91±1.41
MentorNet	80.49±0.11	77.48±3.45	79.12±0.42	70.27±1.52
Co-teaching	87.16±0.52	83.59±0.28	86.54±0.11	80.98±0.39
Forward	85.63±0.11	74.30±0.26	85.29±0.38	74.72±3.24
Joint Optim	89.70±0.36	87.79±0.20	89.69±0.42	82.62±0.57
DMI	88.18±0.13	83.98±0.48	89.14±0.36	84.78±1.97
T-revision	89.63±0.33	86.81±0.21	90.46±0.13	85.37±3.36
CDR	89.68±0.38	86.13±0.44	90.24±0.39	83.07±1.33
Ours	**91.44±0.33**	**88.39±0.34**	**90.86±0.21**	**86.66±0.91**

In the baselines, Co-teaching, Joint Optim, and T-revision are the representative methods that learn robust classifiers by extracting confident examples, refining the noisy labels, and exploiting the noise transition matrix, respectively. Note that Co-teaching keeps updating a constant number of confident examples from the mini-batches used in SGD. We therefore do not compare with its extracted confident examples in Section 3.3.2 as our method extracts confident examples from the whole training data at once. By comparing the classification performance, we can clearly see that the proposed method is much more powerful in extracting confident examples. Note that Joint Optim and T-revision employ all training data to train the classifiers; while our method only employs confident examples and discards the unconfident ones. The results further justify that Me-Momentum is able to extract high-quality confident examples. Note that the performance of Me-Momentum could be further improved by correcting the unconfident data with the idea of Joint Optim.

Real-world dataset We compare Me-Momentum with baseline methods on *Clothing1M* in Table 3.4, where “pre-trained" and “scratch" mean the network was pre-trained by employing *ImageNet* and initialized randomly, respectively;“clean" and “noisy" means the validation data is clean and noisy respectively. First, it is observed that Me-Momentum works well with noisy validation, even surpassing many baselines with clean validation. For a fair comparison, we also use clean validation to validate our method, which achieves the highest test accuracy of 75.18%, better than

Table 3.3: Means and standard deviations of classification accuracy on *CIFAR100*

Methods	Sym-20	Sym-40	Inst-20	Inst-40
Cross-Entropy	57.59±2.55	45.74±2.61	59.85±1.56	43.74±1.54
MentorNet	52.11±0.10	35.12±1.13	51.73±0.17	40.90±0.45
Co-teaching	59.28±0.47	51.60±0.49	57.24±0.69	45.69±0.99
Forward	57.75±0.37	38.59±1.62	58.76±0.66	44.50±0.72
Joint Optim	64.55±0.38	57.97±0.67	65.15±0.31	55.57±0.41
DMI	58.73±0.70	49.81±1.22	58.05±0.20	47.36±0.68
T-revision	65.40±1.07	57.71±0.84	60.71±0.73	51.54±0.91
CDR	66.52±0.24	60.18±0.22	67.06±0.50	56.86±0.62
Ours	**68.03±0.53**	**63.48±0.72**	**68.11±0.57**	**58.38±1.28**

Table 3.4: Classification accuracy on *Clothing1M*.

Method	Validation	Accuracy
Cross Entropy	Clean	69.54%
MentorNet	Clean	56.77%
Co-teaching	Clean	58.68%
Forward	Clean	69.84%
Joint Optim	Clean	72.23%
DMI	Clean	72.46%
T-revision	Clean	74.18%
DivideMix	Clean	74.76%
ELR+	Clean	74.81%
Ours (pre-trained)	Noisy	73.13%
Ours (scratch)	Clean	74.75%
Ours (pre-trained)	Clean	**75.18**%

T-revision by 1% and Joint Optim by 2.95%. Note that Forward and T-revision need the 50k clean data for estimating the transition matrix, while Me-Momentum does not need any clean data for training. In addition, to show the robustness of Me-Momentum, we conduct experiments with ResNet-50 from scratch, which achieves the second best accuracy.

3.3.4 Ablation Study

We discuss the early stopping trick used in Step 1 of Algorithm 1. The training early stops if $(b-a)/(j-i) \leq \tau$, where τ is a hyper-parameter. In the experiments, we set $\tau = 0.1$, which empirically works well across all

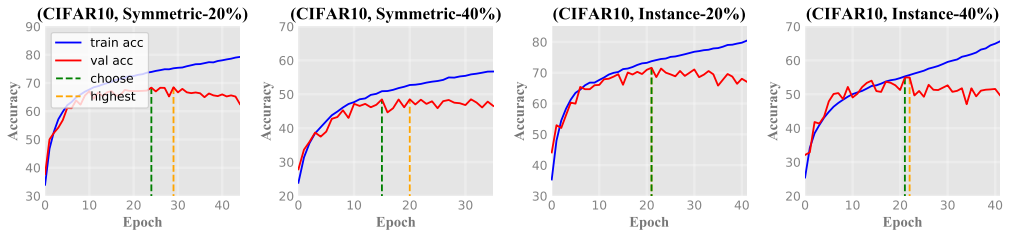


Figure 3.4: Comparing the difference between the early stopping method in Step 1 and the traditional validation method where the classifier with the highest validation accuracy during the whole training procedure will be output. The green dash line indicates the epoch at which early stopping happens; while the orange dash line indicates the epoch at which the highest validation accuracy is achieved during the whole training procedure. In the third plot, the two dash lines are identical to each other.

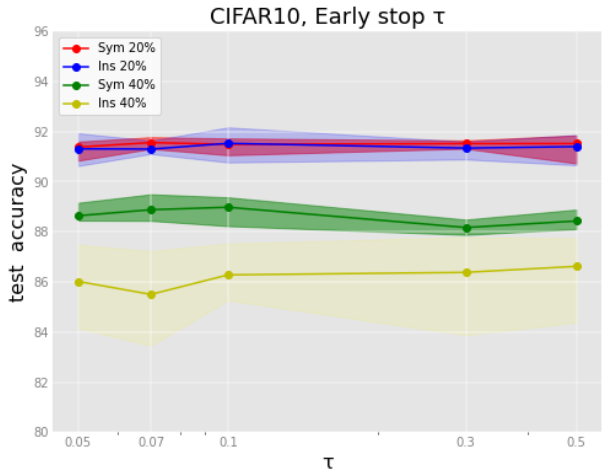


Figure 3.5: Illustrative of extracting hard confident examples.

datasets. In Figure 3.4, we compare the difference between the proposed early stopping method and the traditional validation method. Comparing the blue dash line with the yellow dash line in Figure 3.4, we can observe that the proposed early stopping strategy stops earlier and fits less noise, while the traditional methods would continue to fit more data and thus fit more noise.

We also study the sensitivity of the hyper-parameter. Specifically, we study its sensitivity on *CIFAR10* by setting τ to be the values in the range $\{0.05, 0.07, 0.1, 0.3, 0.5\}$. Other settings are the same as those in this chapter. The results are presented in Figure 3.5. We can see that the

classification performance of Me-Momentum is robust and not sensitive to the change of the value of τ.

3.4 Summary

In this chapter, we propose a method called Me-Momentum that is able to extract hard confident examples from noisily labeled data by exploiting the memorization effect of deep neural networks. At a high level, it fulfills a positive cycle that better confident examples will result in a better classifier and that a better classifier will identify better confident examples. We have empirically verified its effectiveness by analyzing the statistics of the extracted examples, visualizing the hard confident examples, and comparing its classification performance with state-of-the-art baselines. In the future, we will extend our work by utilizing and exploiting the unconfident examples, e.g., in a semi-supervised way to further boost the performance.

Chapter 4

Advancement of Early Stopping

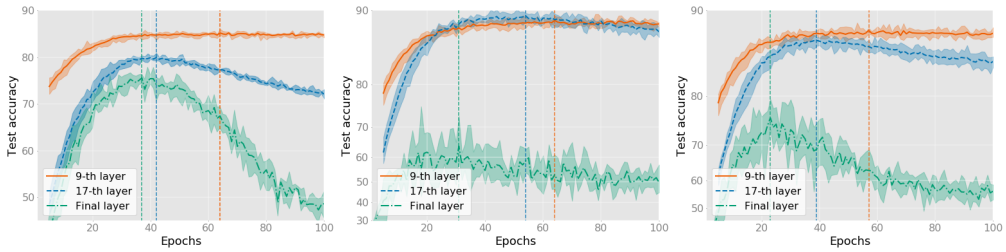
The memorization effect of deep neural network (DNN) plays a pivotal role in many state-of-the-art label-noise learning methods. To exploit this property, the early stopping trick, which stops the optimization at the early stage of training, is usually adopted. Current methods generally decide the early stopping point by considering a DNN as a whole. However, a DNN can be considered as a composition of a series of layers, and we find that the latter layers in a DNN are much more sensitive to label noise, while their former counterparts are quite robust. Therefore, selecting a stopping point for the whole network may make different DNN layers antagonistically affect each other, thus degrading the final performance. In this chapter, we propose to separate a DNN into different parts and progressively train them to address this problem. Instead of early stopping, which trains a whole DNN all at once, we initially train former DNN layers by optimizing the DNN with a relatively large number of epochs. During training, we progressively train the latter DNN layers by using a smaller number of epochs with the preceding layers fixed to counteract the impact of noisy labels. We term the proposed method as progressive early stopping (PES). Despite its simplicity, compared with the traditional early stopping, PES can help obtain more promising and stable results. Furthermore, by combining PES with existing approaches on noisy label training, we achieve state-of-the-art performance on image classification benchmarks.

4.1 Motivations and Contributions

To exploit the memorization effect, when the double descent phenomenon [9, 89, 55] cannot be guaranteed to occur, a core issue is to study when to stop the optimization of the network. While stopping the training for too few epochs can avoid overfitting to noisy labels, it can also make the network underfit to clean labels. Current methods [108, 91] usually adopt an early stopping strategy, which decides the stopping point by considering the network as a whole. However, since DNNs are usually optimized with stochastic gradient descent (SGD) with backpropagation, supervisory signals will gradually propagate through the whole network from latter layers (i.e., layers that are closer to output layers) to former layers (i.e., layers that are closer to input layers). Noting that the output layer is followed by the empirical risk in the optimization procedure. We hypothesize that noisy labels may have more severe impacts for the latter layers, which is different from current methods [42, 65] that usually stop the training of the whole network at once.

To empirically verify the above hypothesis, we analyze the impact of noisy labels on representations from different layers with different training epochs. To quantitatively measure the impact of noisy labels from intermediate layers, we first train the whole network on noisy data with different training epochs and fix the parameters for the selected layer and its previous layers. We then reinitialize and optimize the rest layers with clean data, and the final classification performance is adopted to evaluate the impact of noisy labels. For the final layer, we directly report the overall classification performance. As illustrated in Figure 4.1, we can see that latter layers always achieve the best performance at relatively smaller epoch numbers and then exhibit stronger performance drops with additional training epochs, which verifies the hypothesis that noisy data may have more severe impacts for latter layers. With this understanding, we can infer that the early stopping, which optimizes the network all at once, may fail to fully exploit the memorization effect and induce sub-optimal performance.

To address the above problem, we propose to optimize a DNN by considering it as a composition of several DNN parts and present a novel progressive early stopping (PES) method. Specifically, we initially train



(a) Symmetric 50% **(b)** Pairflip 45% **(c)** Instance 40%

Figure 4.1: We train a ResNet-18 model on CIFAR-10 with three types of noisy labels and evaluate the impact of noisy labels on the representations from the 9-th layer, the 17-th layer, and the final layer. The X-axis is the number of epochs for the first block of the network. The curves present the mean of five runs and the best performances are indicated with dotted vertical lines.

former DNN layers by optimizing them with a relatively large number of epochs. Then, to alleviate the impact of noisy labels for latter layers, we reinitialize and progressively train latter DNN layers by using smaller numbers of epochs with preceding DNN layers fixed. Since different layers are progressively trained with different early stopping epochs, we term the proposed method as progressive early stopping (PES). Despite its simplicity, compared with normal early stopping trick, PES can help to better exploit the memorization effect and obtain more promising and stable results. Moreover, since the model size and training epochs are gradually reduced during the optimization procedure, the training time of PES is only slightly greater than that of the normal early stopping. Finally, by combining PES with existing approaches on noisy label training tasks, we establish new state-of-the-art (SOTA) results on CIFAR-10 and CIFAR-100 with synthetic noise. We also achieve competitive results on one dataset with real-world noise: Clothing1M [130].

4.2 Related Work

Learning with noisy data has been well studied [73, 37, 82, 117, 81]. Current works can be mainly categorized into two groups: model-based and model-free methods. In this section, we briefly review some closely related works.

The first type models the relationship between clean labels and noisy

labels by estimating the noise transition matrix and build a loss function to correct the loss [97, 127, 135, 122]. [97] first combines algorithms for estimating the noise rates and loss correction techniques together and introduces two alternative procedures for loss correction. It also proves that both of the two procedures enjoy formal robustness guarantees *w.r.t.* the clean data distribution. DMI [131] proposes an information-theoretic loss function, which utilizes Shannon's mutual information and is robustness to different kinds of label noise. T-revision [128] estimates the noise transition matrix without anchor points by adding a fine-tuned slack variables. Although these methods have made certain progress, they are usually fragile to estimate the noise transition matrix for heavy noisy data and are also hard to handle a large number of classes. Therefore, in this chapter, we mainly focus on the model-free methods.

The second strand mainly counteracts noisy labels by exploiting the memorization effect that deep networks tend to first memorize and fit majority (clean) patterns and then overfit minority (noisy) patterns [5]. To exploit this property, Co-teaching [42] employs two networks with different initialization and uses *small loss* to select confident examples. M-correction [3] uses two Gaussian Mixture Models to identify confident examples, instead of using networks themselves. DivideMix [65] extends Co-teaching [42] and employs two Beta Mixture Model to select confident examples. MixMatch [11] is then adopted to leverage unconfident examples with a semi-supervised learning framework. All the above methods exploit the memorization effect by considering the adopted network as a whole. Recently, [64] shows that networks training with noisy labels can produce good representations, if the structure of networks suits the targeted tasks. Our method further explains that noisy labels have different impacts for different layers in a DNN. And latter layers will receive earlier and more severe impact than their former counterparts. Therefore, by considering a DNN as a composition of several layers and training different layers with different epochs, our method is able to better exploit the memorization effect and achieve superior performance.

4.3 Methodology

Let D be the distribution of a pair of random variables $(X, Y) \in \mathcal{X} \times \{1, ...C\}$, where X represents the feature instances, Y is the variable of correct labels, $\mathcal{X}$ denotes the feature space, and C is the number of classes. In many real-world problems, examples independently drawn from the distribution D are unavailable. Before being observed, the clean labels are usually randomly corrupted into noisy labels. Let $\tilde{D}$ be the noisy distribution from which pairs $(X, \tilde{Y})$ of features and noisy labels are drawn. For label noise learning, we can only access a sample set $\{x_i, \tilde{y}_i)\}_{i=1}^n$ independently drawn from $\tilde{D}$. The aim is to learn a robust classifier from the noisy sample set that can classify test instances accurately.

In the following, we first elaborate on the proposed progressive early stopping (PES). Then, based on PES, we provide a learning algorithm that learns with confident examples and semi-supervised learning techniques.

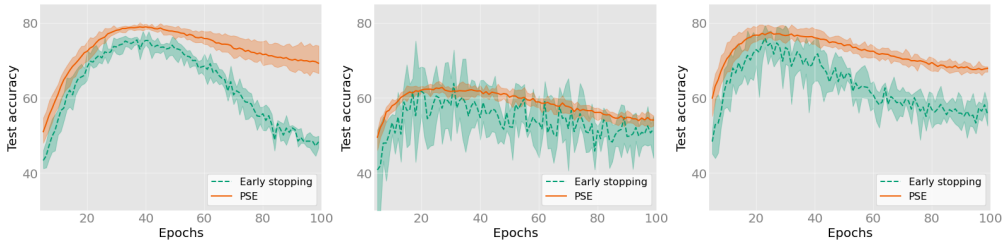
4.3.1 Progressive Early Stopping

When trained with noisy labels, if clean labels are of majority within each noisy class, deep networks tend to first fit clean labels during an early learning stage before eventually memorizing the wrong labels, which can be explained by the memorization effect.

Many current methods utilize this property to counteract the influence of noisy labels by stopping the optimization at an early learning phase. Specifically, a deep classifier can be obtained by optimizing the following objective function with a relatively small epoch number T:

$$\min_{\Theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i; \Theta), \tilde{y}_i), \tag{4.1}$$

where $f(\cdot; \Theta)$ is a deep classifier with model parameters Θ and $\mathcal{L}$ is the cross-entropy loss. When trained with noisy data, early learning regularization (ELR)[72] reveals that, for the most commonly used cross-entropy loss, the gradient is well correlated with the correct direction at the early learning phase. Therefore, with a properly defined small epoch number T, the classifier can have higher accuracy than at initialization. While, if



(a) Symmetric 50% **(b)** Pairflip 45% **(c)** Instance 40%

Figure 4.2: Performance of the traditional early stopping trick and the proposed PES on CIFAR-10 with different types of label noise. The lines present the mean of five runs.

we continue to optimize the deep model after T epochs, the classifier will be able to memorize more noise labels. Therefore, it is critical to select a proper epoch number T to utilize the memorization effect and alleviate the influence of noisy labels.

Current methods typically select the epoch number T by considering the network as a whole. However, as Figure 4.1 makes clear, the impact of noisy labels on different DNN layers are different, which implies that the traditional early stopping trick, which optimizes the whole network all at once, may make different DNN layers to be antagonistically affected by each other, thus degrading the final model performance.

To this end, we propose to separate a DNN into different parts and progressively train layers in different parts with different training epochs. Specifically, assume that the whole network $f(\cdot;\Theta)$ can be constituted with L DNN parts

$$\begin{aligned} z_1 &= f_1(x;\Theta_1), \\ z_l &= f_l(z_{l-1};\Theta_l), \quad l = 2,\ldots,L \end{aligned} \tag{4.2}$$

where $f_l(\cdot;\Theta_l)$ is the l-th DNN part and z_l is the corresponding output. The output of the last part z_L is the prediction. The network $f(\cdot;\Theta)$ can also be represented as $f(\cdot;\Theta_1,...\Theta_L)$. To counteract the impact of noisy labels, We initially optimize the parameter Θ_1 for the first part by training the whole network for T_1 epochs with the following objective,

$$\min_{\Theta_1\ldots\Theta_L} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f(x_i;\Theta_1,\ldots,\Theta_L),\tilde{y}_i). \tag{4.3}$$

Then, we keep the obtained parameter Θ_1^* fixed, reinitialize and progressively learn the l-th ($l = 2, \ldots, L$) DNN part with the parameters for preceding DNN parts fixed. The training procedure is conducted with T_l epochs by optimizing the following objective,

$$\min_{\Theta_l \ldots \Theta_L} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i; \Theta_1^*, \ldots, \Theta_{l-1}^*, \Theta_l, \ldots, \Theta_L), \tilde{y}_i), \quad l = 2 \ldots L \tag{4.4}$$

We gradually optimize the $(l+1)$-th DNN part with the obtained parameter Θ_l^* fixed, the optimization is continued until all the parameters have been optimized. As elaborated above, latter DNN parts are more sensitive to noisy labels than their former counterparts. Therefore, for the above initializing optimization in Eq. (4.3) and the following $L-1$ steps of optimization in Eq. (4.4), we gradually reduce the training epochs (i.e. $T_1 \geq T_2 \geq \cdots \geq T_L$) to better exploit the memorization effect. After optimization, we can obtain the final network as $f(\cdot, \Theta) = f(\cdot; \Theta_1^*, \ldots, \Theta_L^*)$. Since this model is obtained by progressively exploiting the early stopping strategies for different DNN parts, we term the proposed method as progressive early stopping (PES).

To explicitly verify the effectiveness of the proposed PES method, we conduct several pilot experiments, which compare the traditional early stopping and PES with label noise from different types and different levels. The results are illustrated in Figure 4.2, from which we can see that, compared with models trained with traditional early stopping, models trained with PES can achieve superior classification accuracy with smaller variations in all the cases. Current state-of-the-art methods [65] usually adopt models with the traditional early stopping as base models to distill confident examples and then utilize semi-supervised learning techniques by considering confident examples as labeled data and other noisy examples as unlabeled data to further improve the results. The final performance still heavily relies on the base model trained with noisy labels. By improving the performance of the base model, our method combined with semi-supervised learning techniques is able to establish new state-of-the-art results. In the following subsections, we will elaborate on how to utilize PES to distill confident examples and further combine it with semi-supervised learning

techniques. For all the experiments in this chapter, we separate networks by their modules. This approach is rooted in the observation that contemporary deep networks often integrate intricate designs within their modules, featuring elements like residual connections. Simply re-initializing parts of these modules can potentially have significant impacts on other interconnected areas. Therefore, based on our experience, we advocate for breaking down networks by their modules.

4.3.2 Learning with Confident Examples

Based on the deep network optimized with progressive early stopping, we can select confident examples to facilitate the model training. Here, confident examples refer to examples that have high probabilities with clean labels. In this chapter, we treat examples whose predictions are consistent with given labels as confident examples. In addition, to make the results more robust, we generate two different augmentations for any given input and use the average prediction to decide its predicted label. Formally, we can obtain the confident example set $\mathcal{D}_l$ as

$$\begin{aligned} \mathcal{D}_l &= \{(x_i, \tilde{y}_i) | \tilde{y}_i = \hat{y}_i, i = 1, \dots, n\}, \\ \hat{y}_i &= \underset{c \in \{1,\dots,C\}}{\arg\max} \frac{1}{2}[f^c(\text{Augment}(x_i); \Theta) + f^c(\text{Augment}(x_i); \Theta)], \end{aligned} \tag{4.5}$$

where Augment$(\cdot)$ indicates normal data augmentation operation including horizontal random flip and random crops, and $f^c(x; \Theta)$ is the predicted probability of x belonging to class c. Note that Augment$(\cdot)$ is a stochastic transformation, so the two terms in Eq (4.5) are not identical. The average prediction of augmented examples provides a more stable prediction and is found empirically to improve performance. After obtaining the confident example set, one can easily train a classifier by considering confident examples as clean data. However, since the number of confident examples for different classes can vary greatly, directly training the model with the obtained confident example set may introduce a severe class imbalance

problem. To this end, we adopt a weighted classification loss,

$$\mathcal{L}_w = \sum_{i=1}^{N} w_{\tilde{y}_i} \mathcal{L}(\tilde{y}_i, f(x_i; \Theta)), \tag{4.6}$$

where $w_{\tilde{y}_i}$ denotes the weight for the class of the i-th example. The set of confident examples for class c is denoted by σ_c, which indicates the number of confident examples belonging to class c within the labeled dataset $\mathcal{D}_l$. Therefore, the weight w_c for each class can be determined by the proportion of confident examples in that class relative to the total number of confident examples across all classes: $w_c = \sigma_c / (\sum_{j=1}^{C} \sigma_j)$.

4.3.3 Combining with Semi-supervised Learning

Training with only confident examples neglects the rest of the data and may suffer from insufficient training examples. To tackle this problem, we further resort to semi-supervised learning techniques by considering confident examples as labeled data and other noisy examples as unlabeled data. Specifically, the labeled data set and unlabeled data set can be obtained as

$$\begin{aligned} &\begin{cases} \mathcal{D}_l = \{(x_i, \tilde{y}_i) | \tilde{y}_i = \hat{y}_i, i = 1, ...n\} \\ \mathcal{D}_u = \{x_i | \tilde{y}_i \neq \hat{y}_i, i = 1, ...n\} \end{cases} \\ &\hat{y}_i = \arg\max_{c \in \{1, ...C\}} \frac{1}{2} [f^c(\text{Augment}(x_i); \Theta) + f^c(\text{Augment}(x_i); \Theta)], \end{aligned} \tag{4.7}$$

where the labeled data set $\mathcal{D}_l$ is the same as that in Eq (4.6), and $\mathcal{D}_u$ is the rest unlabeled data set. Similar to [65], we adopt MixMatch [11] as the semi-supervised learning framework to train the final classification models. For more details about semi-supervised learning, we refer to [11]. The whole learning algorithm is summarized in Algorithm 2.

4.4 Experiments

4.4.1 Datasets and Implementation Details

Datasets: We evaluate our method on two synthetic datasets, CIFAR-10 and CIFAR-100 [59] with different levels of symmetric, pairflip, and

Algorithm 2: Progressive Early Stopping with Semi-Supervised Learning

Input: Neural network with trainable parameters $\Theta = \{\Theta_1, \ldots, \Theta_L\}$, Noisy training dataset $\{x_i, \tilde{y}_i)\}_{i=1}^n$, Number of training epochs for different part: $T_1, \ldots, T_L$, and training epochs T_c for refining with confident examples.
for $i = 1, \ldots, T_1$ **do**
  Optimize network parameter Θ with Eq. (4.3);
end
for $l = 2, \ldots, L$ **do**
  Froze $\{\Theta_1, \ldots, \Theta_{l-1}\}$ and re-initialize $\{\Theta_l, \ldots, \Theta_L\}$;
  for $i = 1, \ldots, T_l$ **do**
    Optimize network parameter $\{\Theta_l, \ldots, \Theta_L\}$ with Eq. (4.4);
  end
end
Unfroze Θ;
for $i = 1, \ldots, T_c$ **do**
  Extract confident example set $\mathcal{D}_l$ and unlabeled set $\mathcal{D}_u$ with classifier $f(\cdot, \Theta)$ by Eq. (4.7);
  Training the classifier $f(\cdot, \Theta)$ with MixMatch loss on $\mathcal{D}_l$ and $\mathcal{D}_u$;
end
Evaluate the obtained classifier $f(\cdot, \Theta)$.

instance-dependent label noise (abbreviated as instance label noise) and a real-world dataset Clothing1M [130]. Both CIFAR-10 and CIFAR-100 contain 50k training images and 10k test images of size 32×32. Following previous works [42, 128, 72, 124], symmetric noise is generated by uniformly flipping labels for a percentage of the training dataset to all possible labels. Pairflip noise flips noisy labels into their adjacent class. And, instance noise is generated by image features. More details about the synthetic label noise are given in the *supplementary material*. For the flipping rate, it can include [42, 128] or ex-include [65, 72] true labels. We use the flipping rate including correct labels in Table 4.3 to compare with results in [65], and use without correct labels in the rest of the experiments. Clothing1M [130] is a large-scale dataset with real-world noisy labels, whose images are clawed from the online shopping websites, and labels are generated based on surrounding texts. It contains 1 million training images, and 15k validation images, and 10k test images with clean labels.

Table 4.1: Preliminary analysis of the performance and the quality of extracted confident examples on CIFAR-10. The mean and standard deviation are computed over five runs.

Metrics	Methods	Sym-20%	Sym-50%	Pair-45%	Inst-20%	Inst-40%
Test Accuracy	Early Stopping	82.55±2.46	70.76±1.24	60.62±5.59	84.41±0.90	74.73±2.65
	PES	**85.87±1.59**	**75.87±1.33**	**62.40±2.34**	**86.58±0.45**	**77.07±1.18**
Label Precision	Early Stopping	98.81±0.15	94.65±0.19	72.53±5.26	**98.70±0.43**	**90.77±1.87**
	PES	**98.96±0.09**	**95.46±0.14**	**72.99±2.27**	98.52±0.19	90.63±0.92
Label Recall	Early Stopping	88.51±2.26	75.18±1.00	67.84±5.06	90.37±1.01	82.15±3.17
	PES	**92.67±1.43**	**81.03±1.83**	**71.06±2.27**	**93.24±0.60**	**85.91±0.68**

Baselines: Semi-supervised learning may strongly boost the performance, we separately compare our method with approaches with or without semi-supervised learning. For the comparison with baselines with semi-supervised learning, we combine our proposed method with MixMatch used in [65] as indicated in Subsection 4.3.3. (1) Approaches without semi-supervised learning: Co-teaching [42], Forward [97], Joint Optim [108], T-revision [128], DMI [131], and CDR [124]. (2) Methods with semi-supervised learning: M-correction [3], DivideMix [65], and ELR+ [72]. We also adopt standard training with cross-entropy (CE) and MixUp [141] as baselines to show improvements.

Network structure and optimization: Our method is implemented by PyTorch v1.6. Baseline methods are implemented based on public codes with hyper-parameters set according to the original papers. For DivideMix and ELR+, we evaluate the test accuracy with the first network. To better demonstrate the robustness of our algorithm, we keep the hyper-parameters fixed for different types of label noise. More technique details are given in the *supplementary material*.

For experiments without semi-supervised learning, we follow [128], and use ResNet-18 [45] for CIFAR-10 and ResNet-34 for CIFAR-100. We split networks into three parts, the layers above block 4 as part 1, block 4 of ResNet as part 2, and the final layer as part 3. T_1 is defined as 25 for CIFAR-10 and 30 for CIFAR-100, T_2 as 7, and T_3 as 5. The network is trained for 200 epochs and SGD with 0.9 momentum is used. The initial learning rate is set to 0.1 and decayed with a factor of 10 at the 100th and 150th epoch respectively, and a weight decay is set to 10^{-4}. For T_2 and T_3, we employ an Adam optimizer with a learning rate of 10^{-4}.

For experiments with semi-supervised learning, we follow the setting of [65] with PreAct Resnet-18. We set the final layer as part 2, the rest as part 1. T_1 is defined as 20 for CIFAR-10 and 35 for CIFAR-100, and T_2 as 5. The network is trained for 300 epochs. For optimization, we use a single cycle of *cosine annealing* [78], and the learning rate begins from 2×10^{-2} and ends at 2×10^{-4}, with a weight decay of 5×10^{-4}. An Adam optimizer is adopted with a learning rate of 10^{-4} for T_2. For hyper-parameters from MixMatch, we set them according to the original paper [11].

For Clothing1M [130], we follow the previous work [108], and employ a ResNet-50 [45] pre-trained on ImageNet [60]. We set the final layer as part 2, the rest as part 1. T_1 and T_2 are defined as 20 and 7 respectively. The network is trained with CE loss for 50 epochs and SGD is used with 0.9 momentum and a weight decay of 10^{-3}. The learning rate is 5×10^{-3} and decayed by a factor of 10 at the 20th and 30th epoch respectively. We employ an Adam optimizer with a learning rate of 5×10^{-6} for T_2.

4.4.2 Preliminary Experiments

In Figure 4.2, we can observe that with the PES trick, the performance of classifiers is generally improved compared with that the traditional early stopping trick. In this section, we further carefully analyze the quality of extracted labels by examining them from three aspects, i.e., test accuracy, label precision, and label recall. Here, label precision indicates the ratio of the number of extracted confident examples with correct labels in the total confident example set, and label recall represents the ratio of the number of confident examples with correct labels among the total correctly labeled examples. Specifically, we train a neural network on CIFAR-10 with different kinds and levels of label noise for 25 epochs respectively and report the performance for each case before and after the proposed PES is applied.

Results in Table 4.1 clearly show that, compared with the traditional early stopping, PES can help to obtain higher accuracies, precisions, and recalls for most cases. For instance-dependent label noise, PES can achieve higher recall values with comparable label precision values. Note that models with high recall values can help to collect more confident examples,

Table 4.2: Comparison with state-of-the-art methods without semi-supervised learning on CIFAR-10 and CIFAR-100. The mean and standard deviation computed over five runs are presented.

Dataset	Method	Symmetric		Pairflip	Instance	
		20%	50%	45%	20%	40%
CIFAR10	CE	84.00±0.66	75.51±1.24	63.34±6.03	85.10±0.68	77.00±2.17
	Co-teaching	87.16±0.11	72.80±0.45	70.11±1.16	86.54±0.11	80.98±0.39
	Forward	85.63±0.52	77.92±0.66	60.15±1.97	85.29±0.38	74.72±3.24
	Joint Optim	89.70±0.11	85.00±0.17	82.63±1.38	89.69±0.42	82.62±0.57
	T-revision	89.63±0.13	83.40±0.65	77.06±6.47	90.46±0.13	85.37±3.36
	DMI	88.18±0.36	78.28±0.48	57.60±14.56	89.14±0.36	84.78±1.97
	CDR	89.72±0.38	82.64±0.89	73.67±0.54	90.41±0.34	83.07±1.33
	Ours	**92.38±0.40**	**87.45±0.35**	**88.43±1.08**	**92.69±0.44**	**89.73±0.51**
CIFAR100	CE	51.43±0.58	37.69±3.45	34.10±2.04	52.19±1.42	42.26±1.29
	Co-teaching	59.28±0.47	41.37±0.08	33.22±0.48	57.24±0.69	45.69±0.99
	Forward	57.75±0.37	44.66±1.01	27.88±0.80	58.76±0.66	44.50±0.72
	Joint Optim	64.55±0.38	50.22±0.41	42.61±0.61	65.15±0.31	55.57±0.41
	T-revision	65.40±1.07	50.24±1.45	41.10±1.95	60.71±0.73	51.54±0.91
	DMI	58.73±0.70	44.25±1.14	26.90±0.45	58.05±0.20	47.36±0.68
	CDR	66.52±0.24	55.30±0.96	43.87±1.35	67.33±0.67	55.94±0.56
	Ours	**68.89±0.45**	**58.90±2.72**	**57.18±1.44**	**70.49±0.79**	**65.68±1.41**

which is critical for learning with confident examples and semi-supervised learning. Therefore, by enhancing the performance of the initial model, PES can help to improve the final classification performance in all cases, which is also verified by the experiments in Section 4.4.3.

4.4.3 Classification Accuracy Evaluation

Synthetic datasets. We first verify the effectiveness of PES without semi-supervised learning techniques on two synthetic datasets: CIFAR-10 and CIFAR-100. For both of these two datasets, we leave 10% of data with noisy labels as noisy validation set. Results are presented in Table 4.2, which shows that our proposed method can consistently outperform all other baselines across various settings by a large margin.

Table 4.3 and Table 4.4 present the mean accuracy and standard deviation for our method and all baselines on CIFAR-10 and CIFAR-100, respectively. From the results, we can see that the proposed method can outperform all baselines in all cases. For pairflip label noise, the advantages of our proposed method become more apparent, and it significantly outperforms state-of-the-art methods by over 8% on both CIFAR-10 and

Table 4.3: Comparison with state-of-the-art methods with semi-supervised learning on CIFAR-10 and CIFAR-100 with symmetric label noise from different levels. Results with * are token from [65]. The mean and standard deviation are computed over three runs.

Dataset	CIFAR-10			CIFAR-100		
Methods / Noise	Sym-20%	Sym-50%	Sym-80%	Sym-20%	Sym-50%	Sym-80%
CE	86.5±0.6	80.6±0.2	63.7±0.8	57.9±0.4	47.3±0.2	22.3±1.2
MixUp	93.2±0.3	88.2±0.3	73.3±0.3	69.5±0.2	57.1±0.6	34.1±0.6
M-correction*	94.0	92.0	86.8	73.9	66.1	48.2
DivideMix*	95.2	94.2	93.0	75.2	72.8	58.3
DivideMix	95.6±0.1	94.6±0.1	92.9±0.3	75.3±0.1	72.7±0.6	56.4±0.3
ELR+	94.9±0.2	93.6±0.1	90.4±0.2	75.5±0.2	71.0±0.2	50.4±0.8
Ours (Semi)	**95.9±0.1**	**95.1±0.2**	**93.1±0.2**	**77.4±0.3**	**74.3±0.6**	**61.6±0.6**

Table 4.4: Comparison with state-of-the-art methods with semi-supervised learning on CIFAR-10 and CIFAR-100 with instance-dependent and pairflip label noise from different levels. The mean and standard deviation are computed over three runs.

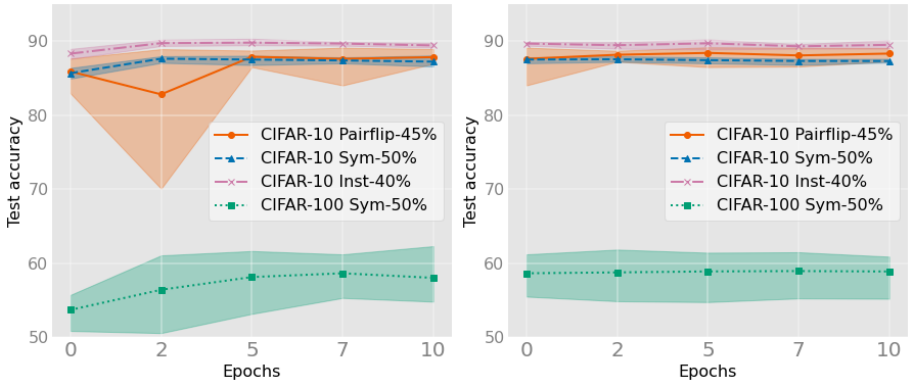
Dataset	CIFAR-10			CIFAR-100		
Methods / Noise	Inst-20%	Inst-40%	Pair-45%	Inst-20%	Inst-40%	Pair-45%
CE	87.5±0.5	78.9±0.7	74.9±1.7	56.8±0.4	48.2±0.5	38.5±0.6
MixUp	93.3±0.2	87.6±0.5	82.4±1.0	67.1±0.1	55.0±0.1	44.2±0.5
DivideMix	95.5±0.1	94.5±0.2	85.6±1.7	75.2±0.2	70.9±0.1	48.2±1.0
ELR+	94.9±0.1	94.3±0.2	86.1±1.2	75.8±0.1	74.3±0.3	65.3±1.3
Ours (Semi)	**95.9±0.1**	**95.3±0.1**	**94.5±0.3**	**77.6±0.3**	**76.1±0.4**	**73.6±1.7**

CIFAR-100. These empirical results support our proposal that PES can improve the quality of selected confident examples, which helps improve performance and reduce the variance of the final classifier.

Real-world dataset. We evaluate the performance of the proposed method on a real-world dataset with Clothing1M [130] and select methods such as CE, Forward, Joint-Optim, DMI, and T-revision, which use a single network, and also methods such as DivideMix and ELR+, which adopt an ensemble model with two different networks, as baselines. We also report the results for the proposed PES with a single network as *ours* and the results for PES, which ensembles two networks, as *ours**. The overall results are reported in Table 4.5, from which we can observe that the proposed PES with a single network can outperform all baselines using a single network. And with an ensemble model, which contains two different networks, our method can outperform all the adopted baselines. These results clearly

Table 4.5: Compassion with state-of-the-art methods on Clothing1M. Results of baseline methods are taken from the original papers. ours represent the results obtained by PES with a single network and ours* indicate the results obtained by PES with an ensemble model.

CE	Forward	Joint-Optim	DMI	T-revision	DivideMix*	ELR+*	Ours	Ours*
69.21	69.84	72.16	72.46	74.18	74.76	74.81	**74.64**	**74.99**



(a) T_2 **(b)** T_3

Figure 4.3: Sensitivity analysis for different training iteration numbers: T_2 and T_3.

demonstrate that, by enhancing the performance of the initial classification network, our method exhibits greater flexibility and robustness in handling such real-world noise problems.

4.4.4 Sensitivity Analysis

In this section, we investigate the hyper-parameter sensitivity for the training iteration number T_2 and T_3, respectively. We firstly analyze the training epoch number for the second DNN part by varying T_2 from the range of $[0, 10]$. The results are illustrated in Figure 4.3(a), from which can find that, with the increasing of T_2, the performance of PES first increase and then decrease in all the cases except for 45% Pairflip noise on the CIFAR-10 dataset. While the model achieves the best performance with T_2 as 7 for all types of noisy labels. Then we fix T_2 as 7, and analyze the impact of the third DNN part by varying T_3 from the range of $[0, 10]$. The results are shown in Figure 4.3(b). Although the performance variance for different T_3 is smaller than that for T_2, we can still observe that the best performance

Table 4.6: Training time comparison for baselines on CIFAR-10 with 50% Symmetric label noise.

CE	Co-teaching	CDR	T-revision	ELR+	DivideMix	Ours	Ours (Semi)
0.9h	1.5h	3.0h	3.5h	2.2h	5.5h	1.0h	3.1h

can be obtained when T_3 is set as 5. More importantly, from these two figures, we can get that both T_2 and T_3 are robust to the different types of noisy labels.

4.4.5 Training Time Comparison

In this section, we compare the training time of our method and other state-of-the-art baselines. All the experiments are conducted on a server with a single Nvidia V100 GPU. The training times for all the methods are reported in Table 4.6, from which we can get that our algorithm with cross-entropy loss achieves the fastest speed across all baselines, only about 1 hour. Our method combining with MixMatch [11] is also fast, only a little more than half of the training time of DivideMix. The time of ELR+ [72] shows superior, but ELR+ trains the network with fewer epochs, with 200 epochs compared with ours for 300 epochs.

4.5 Summary

In this chapter, we provide a progressive early stopping (PES) method to better exploit the memorization effect of deep neural networks (DNN) for noisy-label learning. We first find that the impact of noisy labels for former layers in a DNN is much less and later than that for latter DNN layers, and then build upon this insight to propose the PES method, which separates a DNN into different parts and progressively train each part to counteract the different impacts of noisy labels for different DNN layers. To show that PES can boost the performance of state-of-the-art methods, we conduct extensive experiments across multiple synthetic and real-world noisy datasets and demonstrate that the proposed PES can help to obtain substantial performance improvements compared to current state-of-the-art baselines. The main limitation of our method lies in that, by splitting a DNN into

different parts, PES introduces several additional hyper-parameters that need to be tuned carefully.

Chapter 5

Applicability of Early Stopping

This chapter focuses the applicability of early stopping techniques within the context of self-supervised learning. Most recent self-supervised learning methods learn visual representation by contrasting different augmented views of images. Compared with supervised learning, more aggressive augmentations have been introduced to further improve the diversity of training pairs. However, aggressive augmentations may distort images' structures, leading to a severe semantic shift problem where augmented views of the same image may not share the same semantics, thus degrading the transfer performance.

Furthermore, in self-supervised learning, the generated representations function similarly to labels in supervised learning. Consequently, semantic shift problem resulting from aggressive augmentations can be equated to a form of label noise. To address this problem, we propose a new SSL paradigm that counteracts the impact of semantic shifts by balancing the roles of weak and aggressively augmented pairs. Specifically, semantically inconsistent pairs are of minority, and we treat them as noisy pairs. Note that deep neural networks (DNNs) have a crucial memorization effect that DNNs tend to first memorize clean (majority) examples before overfitting to noisy (minority) examples. Therefore, we set a relatively large weight for aggressively augmented data pairs at the early learning stage. With the training going on, the model begins to overfit noisy pairs. Accordingly, we gradually reduce the weights of aggressively augmented pairs. In doing so, our method can better embrace aggressive augmentations and neutralize the semantic shift problem. Experiments show that our model achieves 73.1% top-1 accuracy on ImageNet-1K with ResNet-50 for 200

epochs, which is a 2.5% improvement over BYOL. Moreover, experiments also demonstrate that the learned representations can transfer well to various downstream tasks.

5.1 Motivations and Contributions

Representative contrastive methods are generally trained by maximizing agreement between differently augmented views of the same image (positive pairs), and increasing the distance between augmented views from different images (negative pairs) [123, 19, 43]. Compared with supervised learning, these works highlight the role of data augmentation for SSL and design more aggressive augmentation operations. Here, we refer aggressive augmentations as the operations that can possibly change the semantics of images, such as grayscale, color jitter, and Gaussian blur. Other used augmentations are referred as weak augmentations. It should be noted that the random cropping operation can also change the semantic information, however, this issue can be properly addressed by exploiting object detection techniques [105, 87]. Therefore, we do not include the random cropping operation as an aggressive augmentation. Although the aggressive augmentations can help to further improve the model performance, they also bring a severe semantic shift problem for training images. As illustrated in Figure 5.1, the first row shows original images from ImageNet [60] and CIFAR-100 [59] datasets. And the second row presents the corresponding augmented views with the widely used composition of augmentations [20, 19]. We can see that the augmented views can be hardly recognized as semantically consistent with their original versions. Pushing these images to have similar representations can adversely affect the model training, so we consider the semantically inconsistent pairs as noisy pairs. However, due to the diversity of training images and the randomness of augmentation, it is difficult to accurately measure the quality of augmented pairs. Attempting to roughly remove noisy pairs may result in a decrease in performance since they are mixed with clean (semantic consistent) pairs.

Fortunately, recent works [4, 140, 42, 56, 72] show that deep neural networks (DNNs) have a crucial memorization effect that DNNs tend

to first memorize clean (majority/semantically consistent) examples before overfitting noisy (minority/semantically inconsistent) examples. Motivated by this, we propose a method called Reducing Semantic shift from Aggressive augmentation (RSA) by dynamically adjusting the weights of clean and noisy pairs. Specifically, to counter the impact of the semantic shift problem, we introduce aggressive-weak augmented pairs, which are relatively cleaner than aggressive-aggressive pairs. At the beginning of the training process, we assign a high weight to aggressive-aggressive pairs and a lower weight to aggressive-weak pairs to maximize the utilization of all training examples. As training progresses, the model starts to overfit data that is semantically inconsistent. Since weak augmentations are less likely to induce semantic shifts and most semantically inconsistent data stems from aggressively augmented pairs, we progressively increase the weight of the aggressive-weak pairs. Simultaneously, we reduce the weight of the aggressive-aggressive pairs to reduce the impact of semantic inconsistencies.

Empirical results on multiple benchmark datasets show that our method can outperform state-of-the-art methods in various settings with a large margin. For instance, with 200 epochs of pre-training, our method achieves 73.1% Top-1 accuracy on ImagetNet-1K [60] linear evaluation protocol, which is 2.5% higher than BYOL [40]. Experiments on MS COCO [70] also show that our pre-trained model can continually improve the performance for multiple downstream tasks.

5.2 Related Work

Self-supervised learning (SSL) has attracted great attention to capture universal representations [54, 143, 94, 120, 17, 31, 145]. The core of SSL is designing agent tasks, which allow us to learn representations from large-scale unlabeled data via pseudo labels instead of using any human annotations. To this end, many proposals devise different solutions for constructing pseudo labels, including predicting the rotation of images [36], putting pieces of images together [92], or recovering color from grayscale images [142]. Particularly, Wu et al. [123] propose an instance-level classification, which regards images augmented from the same image as a positive

(a) Noisy samples from ImageNet-1K **(b)** Noisy samples from CIFAR-100

Figure 5.1: The first row is the original images (ImageNet-1K [60] images are resized to squares) and noisy samples from aggressive augmentation are in the second row. From the first three images of (a), we can observe that Color jitter operation makes the image too bright or too dark that covers the details of images; and in the fourth and fifth columns of (a), Grayscale and Gaussian blur operation leads images to be hardly distinguished from the background; and the same augmentation strategy from ImageNet-1K leads to more vague images for CIFAR-100 [59] shown in (b).

pair and others as negative examples. SimCLR [19] improves performance by inserting the projection network and introducing aggressive augmentation. He et al. [43, 20] store negative representations in a queue to reduce the memory requirement. BYOL [40] enhances the power of SSL by removing the dependence on negative examples, which also addresses the problem of false negative examples [24]. Despite these methods having proved their effectiveness based on aggressive augmentation, they ignore the semantic shift problem from it.

Noisy samples in aggressive augmentation. Recent studies [119, 105, 87, 98] have discovered that aggressive augmentation may generate noisy samples in the positive pairs. To alleviate this issue, ContraCAM [87] proposes a two-step approach to reduce the issue of random cropping, which seeks objects first and then crops images based on their locations. In addition, Gansbeke et al. [115] conduct experiments on scene-centric datasets (e.g., COCO) containing multiple objects in images and argue that SSL can overcome the issue of random cropping. Since the issue of random cropping has been well studied, our work focuses on other types of augmentations, which can be viewed as a complement to previous studies.

Learning with noisy labels. Reducing the semantic shift problem in SSL is similar to another well-studied topic in machine learning, learning

with noisy labels [73, 128, 135, 127, 134]. In this field, many state-of-the-art methods use the memorization effect to select confident examples in the early learning phase [93, 108, 65]. Specifically, Co-teaching [42] uses the early stopping trick and the small-loss strategy to choose confident examples. PES [7] finds examples with noisy labels have more detrimental effects on the latter layers, and improves the early stopping trick by progressively training each part of DNNs. However, directly using the early stopping trick is difficult in SSL because noisy pairs are randomly generated by aggressive augmentations over epochs while samples with noisy labels are fixed in learning with noisy labels. Besides, roughly removing noisy pairs is more likely to delete many clean pairs by mistake, especially in the low noise rate case.

5.3 Methodology

Our approach aims to minimize the detrimental impacts of semantically inconsistent (noisy) pairs from aggressive augmentations while taking advantage of aggressive augmentations. As such, we first revisit preliminaries on self-supervised learning. Then, we elaborate on the proposed learning algorithm that counterbalances the noise impacts by utilizing the memorization effect of DNNs.

5.3.1 Preliminaries on Self-supervised Learning

Self-supervised learning methods based on contrastive learning generally require learning an embedding space that can easily separate different examples. Let D be a set of images, and an image x_i is uniformly drawn from D. Denote t and t' as two different instances from the same distribution of image augmentation T. v and v' are two augmented views of the image x_i with $v = t(x_i)$ and $v' = t'(x_i)$, which are regarded as a pair of positive examples. Then, v and v' will separately feed through an encoder f_θ and a projector g_θ to embed z and z', which are required to be close to each other via a contrastive loss function, e.g., InfoNCE [95] can be expressed as,

$$\mathcal{L}_{NCE} = -\log \frac{\exp(z \cdot z'/\gamma)}{\exp(z \cdot z'/\gamma) + \sum_{n \in N} \exp(z \cdot n/\gamma)}, \tag{5.1}$$

where γ is a temperature parameter, and N is a set of negative example vectors. The embeddings of positive and negative examples are l_2-normalized. To stabilize the training process, some state-of-the-art methods [43, 145, 58] employ an asymmetric framework, including an online network and a target network. For the online and target networks, they have the same network structure but different weights of encoder f_ξ, and projector g_ξ, whose parameters ξ are updated by the online parameters θ with the exponential moving average method.

Recently, BYOL [40] found negative examples are not necessary and added a predictor q_θ in the online network to avoid collapsed solutions, e.g., all images have the same vector. And, the loss function can be simplified to,

$$\mathcal{L}_{mse} = 2 - 2 * \frac{\langle z, z' \rangle}{\|z\|_2 * \|z'\|_2}. \tag{5.2}$$

5.3.2 Description of RSA

As discussed in Introduction, noisy pairs from aggressive augmentations will lead to a severe semantic shift problem, which damages the generalization on downstream tasks. However, most of the sample pairs generated from aggressive augmentations are beneficial for the model performance, and what precise degree of distortion would cause bias remains unknown, so it is hard to distinguish between clean pairs and noisy pairs in SSL. Instead of separating noise from training data, we employ weak augmentation to generate sample pairs as relatively clean pairs.

To this end, we propose an efficient method to reduce semantic shift from aggressive augmentations, dubbed RSA. Envisioned by the memorization effect, DNNs will first fit clean pairs (majority/semantically consistent) of training data in the early learning phase and then overfit to noisy pairs (minority/semantically inconsistent). Noisy pairs account for the minority, so we assume that the noise impacts will be small and then increases as the training process. We give different weights to clean and noisy pairs, and gradually reduce the weights of noisy pairs against raised noise impacts.

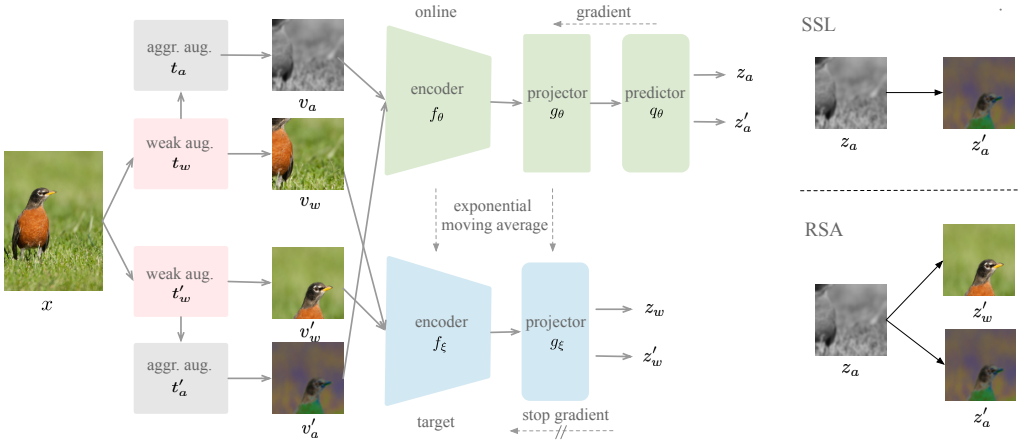


Figure 5.2: The illustration of our proposed method (RSA). We utilize an asymmetric-style framework, including an online network and a target network. The online network is optimized by gradients, and the target network is updated with the exponential moving average strategy. We first adopt the weak augmentation to generate two views (v_w, v'_w), then adopt the aggressive augmentations to further generate another two views (v_a, v'_a). Subsequently, we make aggressive-augmented views to keep consistent with their corresponding weak- and aggressive-augmented views in the embedding space. On the right of the image, we compare RSA with classical SSL methods. RSA forces learned representations to a balance between weak- and aggressive-augmented views. (Best viewed in color)

However, simply introducing clean pairs will significantly increase computation, and SSL has known that more computation will increases performance, which makes it difficult to fairly compare with other baselines. To limit computation, we meet two problems. First, generating more augmented instances, especially double instances, commonly means expensive computation, largely burdening CPU and hard disk, which leads to slow data processing down and low efficiency of GPU [19]. Second, more instances require more back-propagation times, which results in more processing time for GPU.

To address the first problem, we propose a novel data augmentation pipeline, called multi-stage augmentation, which can generate double instances with nearly the same resources. Specifically, data augmentation generates image variants by mixing different types of image transformation and arranging them into a queue. For a transformation, it receives an image from the result of the previous transformation and transforms the image

based on pre-defined probability, and then passes it to the next transformation, $t_{aug}(x) = t_a(t_w(x))$, where t_a and t_w are two subsets of t_{aug}. Based on this process, we can separate it into two child processes. Namely, t_w includes weak augmentation operations while t_a conducts aggressive augmentation operations based on t_w. This process can be described as,

$$\begin{aligned} v_w &= t_w(x) \quad v'_w = t'_w(x), \\ v_a &= t_a(v_w) \quad v'_a = t'_a(v'_w). \end{aligned} \tag{5.3}$$

For the second problem, we send aggressive augmented views to the online network and weak augmented views to the target network, which means each instance only forwards once. Accordingly, RSA requires back-propagation twice, which is equal to many state-of-the-art SSL methods with a symmetrized loss [19, 21, 40, 17].

$$\begin{aligned} z_a &= q_\theta(g_\theta(f_\theta(v_a))) \quad z_w = g_\xi(f_\xi(v_w)), \\ z'_a &= q_\theta(g_\theta(f_\theta(v'_a))) \quad z'_w = g_\xi(f_\xi(v'_w)). \end{aligned} \tag{5.4}$$

After obtaining four representations, we group them into four pairs and make each aggressive-augmented view to keep consistent with their corresponding weak- and aggressive-augmented views in the embedding space. For the corresponding aggressive-augmented views, rather than encoding them on the target network, we use the representations from the online network with a stop-gradient operation. The total loss is summarized as,

$$\begin{aligned} \mathcal{L} = &(1-\beta)\mathcal{L}_{mse}(z_a, z'_w) + \beta\mathcal{L}_{mse}(z_a, z'_a) \\ &+(1-\beta)\mathcal{L}_{mse}(z'_a, z_w) + \beta\mathcal{L}_{mse}(z'_a, z_a), \end{aligned} \tag{5.5}$$

where β is a calculated parameter to re-weight the losses from weak- and aggressive-augmented views. This loss function forces networks to learn representations that achieve a balance between weak- and aggressive-augmented views, which reduces overfitting to noisy pairs while avoiding a simple solution. As the training goes on, the noisy impacts will increase. We further offset it by using the memorization effect, which decreases β with a cosine decay equation,

Algorithm 3: RSA: Reducing semantic shift

Input: Neural network f_θ and f_ξ. Projector g_θ and g_ξ. Predictor q_θ. A batch of samples x. An image set D. Total number of training steps K. Weak augmentation function t_w. Aggressive augmentation function t_a. Hyper-parameter β_{base}.

for $k \leftarrow 1$ **to** K **do**

- $x \leftarrow D$ `// Sample a batch of images`
- $v_w \leftarrow t_w(x)$ and $v'_w \leftarrow t'_w(x)$ `// Multi-stage augmentation`
- $v_a \leftarrow t_a(v_w)$ and $v'_a \leftarrow t'_a(v'_w)$
- $z_a \leftarrow q_\theta(g_\theta(f_\theta(v_a)))$ and $z_w \leftarrow g_\xi(f_\xi(v_w))$ `// Embedding for four views`
- $z'_a \leftarrow q_\theta(g_\theta(f_\theta(v'_a)))$ and $z'_w \leftarrow g_\xi(f_\xi(v'_w))$
- $\mathcal{L} \leftarrow (1-\beta)\mathcal{L}_{mse}(z_a, z'_w) + \beta\mathcal{L}_{mse}(z_a, z'_a) + (1-\beta)\mathcal{L}_{mse}(z'_a, z_w) + \beta\mathcal{L}_{mse}(z'_a, z_a)$
- $\beta \leftarrow \beta_{base} \times \frac{1}{2}(cos(\pi\frac{k}{K}) + 1)$ `// Decreasing` β
- Update f_θ, g_θ and q_θ with $\mathcal{L}$
- Update f_ξ and g_ξ by slowly momentum updating with f_θ and g_θ

end

Output: The trained network f_θ

$$\beta = \beta_{base} \times \frac{1}{2}(cos(\pi\frac{k}{K}) + 1), \tag{5.6}$$

where β_{base} is a given number at the training beginning, and k and K are the current training steps and the total training steps.

5.4 Experiments

5.4.1 Datasets and Implementation Details

Datasets: We assess the proposed method on six image datasets, from small to large. We choose CIFAR-10/100 [59] for small datasets, and STL-10 [25] and Tiny ImageNet [1] for medium datasets, and ImageNet-100 [112] and ImageNet-1K [60] for large datasets. Note, ImageNet-100 contains 100 classes that are randomly selected from ImageNet-1K [60] and we choose the same classes with [112]. For STL-10, both 5k labeled and 100k unlabeled

Table 5.1: Analysis of the multi-stage augmentation and the memorization effect. We run methods on small and medium datasets for 200 epochs, and on ImageNet-100 for 100 epochs. The mean and standard deviation are computed over three trials. Note that two augmented instances with aggressive augmentation are referred to as AA, instances with one aggressive and one weak augmentation as AW, and instances with multi-stage augmentation as MA.

Method	Aug.	β	CIFAR-10	CIFAR-100	STL-10	Tiny ImageNet	ImageNet-100
BYOL	AA	–	90.2±0.0	63.9±0.5	91.3±0.1	49.5±0.1	80.9
BYOL	AW	–	90.5±0.2	66.1±0.3	90.7±0.0	51.1±0.3	81.2
RSA	MA	Fixed	91.3±0.2	66.5±0.0	92.8±0.2	53.9±0.2	83.2
RSA	MA	Decay	**92.1±0.0**	**67.6±0.3**	**93.0±0.2**	**54.7±0.3**	**83.5**

images are used for the pre-trained model, and only 5k labeled images are used for the linear evaluation.

Augmentation: In this chapter, we define "aggressive" augmentation including grayscale, color jitter, and Gaussian blur, while "weak" augmentation includes random crop and horizontal flip. The hyper-parameters of the augmentations are following MoCo v2 [20] except for the size of the cropped images for the small and medium datasets, and we resize images to 32×32 and 64×64 for the small and medium datasets, respectively.

Baselines: For the comparison, we re-implement the state-of-the-art methods, SimCLR [19], MoCo v2 [20], SimSiam [21], and BYOL [40] based on the public codes. We follow [21] that implements the MoCo v2 with a symmetrized loss function, and set the exponential moving average factor to 0.99 for all experiments. For BYOL [40] on the small and medium datasets, we follow [40], and set the channel inner layer of 1024 in the projection and prediction MLP and the output feature is 128. We initially set the exponential moving average factor as 0.99 and gradually enlarge it to 1.

Network structure and optimization: Our method and reproduced methods are implemented by PyTorch v1.8 and we conduct all experiments on Nvidia V100. Our method is based on our reproduced BYOL [40]. For the pre-train stage on the small and medium datasets, we adopt ResNet-18 [45] as a backbone. For optimization, we use SGD optimizer with a cosine-annealed learning rate of 0.1 [78], a momentum of 0.9, weight

decay of 5×10^{-4}, and a batch size of 256. We set $\beta_{base} = 0.3$ for CIFAR-10/100 and $\beta_{base} = 0.4$ for Tiny ImageNet and STL-10.

For the pre-train stage on large datasets, we adopt a standard ResNet-50 [45] as a backbone. For ImageNet-100, the network is trained using SGD optimizer with a single cycle of cosine annealing [78], and an initial learning rate of 0.2, a momentum of 0.9, weight decay of 10^{-4}, and a batch size of 256. We conduct ImageNet-1K experiments with 8× Nvidia V100 32G with Automatic Mixed Precision (AMP) package [85]. Specifically, we follow [40], and train a network with a LARS optimizer [136] with a single cycle of cosine annealing [78], a momentum of 0.9, weight decay of 10^{-6}, and a batch size of 2048. The base learning rate starts from 0.9 and 0.6 for 100 and 200 epochs respectively, linearly scaled by the times of batch size 256 [39]. We set $\beta_{base} = 0.4$ for ImageNet-100 and ImageNet-1K.

Evaluation: We evaluate the representations of the pre-trained model with the linear evaluation protocol, which freezes the encoder parameters and trains a linear classifier on top of the pre-trained model. For the small and medium datasets, we follow the setting in MoCo v2 [20] and train a linear classifier for 100 epochs with an initial learning rate of 30, no weight decay, and a momentum of 0.9. The learning rate will be multiplied by 0.1 at the 60 and 80 epochs. For the large datasets, we follow the evaluation setting in Mean Shift [58], which only requires 40 epochs and a batch size of 256. The linear classifier is trained with SGD and an initial learning rate of 0.01, weight decay of 10^{-4}, and a momentum of 0.9. The learning rate will be multiplied by 0.1 at 15, 30, and 40 epochs.

5.4.2 Preliminary Analysis

In Table 5.1, we investigate the effectiveness of the proposed multi-stage augmentation and the memorization effect. We first compare the linear classification of BYOL with two aggressive augmentations (AA) and with one aggressive and one weak augmentation (AW) and RSA with multi-stage augmentation (MA) without memorization effect (fixed β). From the second row, we can see that BYOL with AW can largely improve the performance on CIFAR-100 and Tiny ImageNet, and mildly improve the performance on CIFAR-10 and ImageNet-100, but the network suffers from

Table 5.2: Performance comparison with linear classification on small and medium datasets for 200 and 800 epochs. We adopt a ResNet-18 as a backbone for all experiments. The mean and standard deviation are computed over three trials.

Method	CIFAR-10		CIFAR-100		STL-10		Tiny ImageNet	
	200 ep	800 ep	200 ep	800 ep	200 ep	800 ep	200 ep	800 ep
SimCLR	88.3±0.2	91.1	59.5±0.3	63.8	87.9±0.4	91.0	46.7±0.1	48.4
MoCo v2	87.1±0.2	91.4	60.3±0.4	65.0	88.4±0.3	91.3	47.8±0.4	50.1
SimSiam	86.9±0.1	91.1	55.8±0.9	62.4	85.0±0.4	89.7	41.2±0.3	44.5
BYOL	90.2±0.0	92.7	63.9±0.5	68.2	91.3±0.1	93.4	49.5±0.1	53.0
RSA (Ours)	**92.3±0.1**	**93.7**	**68.1±0.5**	**70.4**	**93.1±0.2**	**94.0**	**54.8±0.4**	**55.5**

AW on STL-10. These inconsistent results suggest that although AW can help against the noise impacts from aggressive pairs, it reduces the diversity of augmented pairs compared with AA, which may result in a suboptimal result. In contrast, MA not only outperforms AA and AW but also generally improves performance across five datasets, suggesting that multi-stage augmentation (MA) can take advantage of AA and AW at the same time.

According to the memorization effect of DNNs, noise impacts vary at different stages of the training process and become more apparent at the end. In order to balance the increasing noise impact, we decay β during the training, which results in rising the weights of aggressive-weak augmented pairs and reducing the weights of aggressive-aggressive augmented pairs. Improved performance in the fourth row verifies that we can use the memorization effect of DNNs to further reduce noise impact. Notably, we set $\beta_{base} = 0.5$ in preliminary experiments, and the performance can be further improved with a turned β_{base} in Table 5.2 and 5.3.

5.4.3 Linear Classification

Small and medium datasets. We evaluate our method on small and medium datasets, with 200 and 800 epochs. We also run three trials for a short running time to evaluate the stability of the proposed method. Table 5.2 illustrates that the proposed method significantly improves the performance across the four datasets on short training time experiments, demonstrating that RSA can accelerate convergence and the small standard deviation also shows that the proposed method has good stability. For the

Table 5.3: Performance comparison with linear classification on ImageNet-100 for 100 and 200 epochs. All methods adopt ResNet-50 as a backbone.

Method	Batch size	100 ep	200 ep
SimCLR	256	79.1	82.4
MoCo v2	256	80.9	83.9
SimSiam	256	79.7	82.6
BYOL	256	80.9	83.6
RSA (Ours)	256	**83.7**	**85.5**

Table 5.4: Performance comparison with linear classification on ImageNet-1K. All methods use a standard ResNet-50 as a backbone without a multi-crop strategy.

Method	Neg. pairs	Batch Size	Epochs	Top-1
Supervised		256	100	76.2
InstDis [123]	✓	256	200	56.5
PIRL [86]	✓	256	200	63.6
SimCLR [19]	✓	4096	1000	69.3
MoCo v2 [20]	✓	256	200	67.5
JCL [16]	✓	256	200	68.7
ReSSL [145]	✓	256	200	69.6
InfoMin Aug. [113]	✓	256	200	70.1
W-MSE 4 [31]		4096	400	72.6
SimSiam [21]		256	200	70.0
SwAV [21]		4096	200	69.1
BYOL [21]		4096	100	66.5
BYOL [21]		4096	200	70.6
BYOL [40]		2048	300	72.4
RSA (Ours)		2048	100	71.4
RSA (Ours)		2048	200	**73.1**

long training time experiments, outstanding results show RSA can continue to improve the final performance.

Large datasets. We evaluate the performance of the proposed method on the large datasets, ImageNet-100 and ImageNet-1K. We reproduce all baselines on ImageNet-100 with batch size 256. The results on ImageNet-100 and ImageNet-1K are shown in Table 5.3 and Table 5.4 respectively. We

can see that RSA outperforms the state-of-the-art methods on ImageNet-100 with a relatively large margin across 100 and 200 epochs. For results on ImageNet-1K, RSA consistently surpasses baselines, e.g., the performance of RSA for 100 epochs has already surpassed BYOL training for 200 epochs. RSA achieves a new state-of-the-art result for 200 epochs, exhibiting a 2.5% improvement over BYOL. Overall, empirical results on linear evaluation verify that RSA can constantly improve the generalization in various settings.

5.4.4 Transfer Learning

We further verify the quality of representation learned by RSA on more downstream tasks. For object detection and instance segmentation, we follow [113], and adopt Mask R-CNN [44] with FPN [69] to fine-tune our pre-trained ResNet-50 model on COCO *train2017* with 1 × schedule and 2 × schedule, and evaluate performance on COCO *val2017*. We set the initial learning rate as 0.02 and 0.03 for the 1 × schedule and 2 × schedule experiments respectively.

Table 5.5 and Table 5.6 report the results of object detection and instance segmentation. We can observe that RSA outperforms all baselines on object detection and instance segmentation tasks, especially, showing superior over the strong baseline InfoMin Aug. [113] that adopts more advanced augmentation, RandAugment [26]. Strong performance on downstream tasks demonstrates that RSA can improve the general quality of learned representations.

5.4.5 Training Time Comparison

We compare the running time between our method and reproduced BYOL. We conduct experiments on CIFAR-100 and STL-10 for 800 epochs with a single Nvidia V100, and ImageNet-100 for 200 epochs with 4× Nvidia V100, respectively. For ImageNet-100, we use Automatic Mixed Precision package [85] to speed up the training process and save GPU memory. Note that since we do not have enough GPUs to run a standard BYOL with a

Table 5.5: Transfer learning on downstream tasks: object detection, instance segmentation. All models were pre-trained on ImageNet-1K for 200 epochs and fine-tuned on MS COCO with 1 × schedule. Object detection and instance segmentation results are from [113].

Method	Object detection			Instance segmentation		
	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	32.8	50.9	35.3	29.9	47.9	32.0
supervised	39.7	59.5	43.3	35.9	56.6	38.6
InstDis [123]	38.8	58.4	42.5	35.2	55.8	37.8
PIRL [86]	38.6	58.2	42.1	35.1	55.5	37.7
MoCo [43]	39.4	59.1	42.9	35.6	56.2	38.0
MoCo V2 [20]	40.1	59.8	44.1	36.3	56.9	39.1
InfoMin Aug. [113]	40.6	60.6	44.6	36.7	57.7	39.4
RSA (Ours)	**41.1**	**61.4**	**45.1**	**37.3**	**58.6**	**40.1**

Table 5.6: Transfer learning on downstream tasks: object detection, and instance segmentation. All models were pre-trained on ImageNet-1K for 200 epochs and fine-tuned on MS COCO with 2 × schedule. Baseline results are from [113].

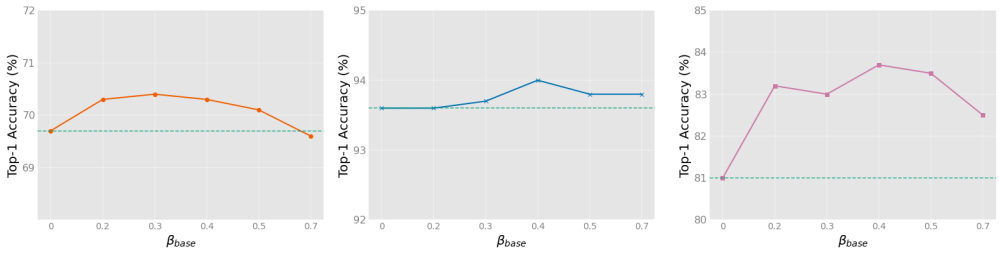
Method	Object detection			Instance segmentation		
	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
Random	38.4	57.5	42.0	34.7	54.8	37.2
Supervised	41.6	61.7	45.3	37.6	58.7	40.4
InstDis [123]	41.3	61.0	45.3	37.3	58.3	39.9
PIRL [86]	41.2	61.2	45.2	37.4	58.5	40.3
MoCo [43]	41.7	61.4	45.7	37.5	58.6	40.5
MoCo v2 [20]	41.7	61.6	45.6	37.6	58.7	40.5
InfoMin Aug. [113]	42.5	62.7	46.8	38.4	59.7	**41.4**
RSA (Ours)	**42.7**	**62.9**	**47.1**	**38.5**	**60.0**	**41.4**

Table 5.7: Training time comparison with BYOL on three datasets

Method	CIFAR-100	STL-10	ImageNet-100
BYOL	11.3h	77.6h	9.5h
RSA	11.5h	79.6h	10.4h

batch size of 4096 on ImageNet, we use ImageNet-100 instead of ImageNet-1K, which has similar processing efficiency but ten times the data.

Although our method uses double instances in the loss function 5.5, each instance in RSA only passes into the network one time. Therefore,



(a) CIFAR-100 [59] **(b)** STL-10 [25] **(c)** ImageNet-100 [60]

Figure 5.4: Sensitivity analysis for the hyper-parameter β_{base}. We conduct experiments on CIFAR-100 and STL-10 for 800 epochs, and ImageNet-100 for 100 epochs, respectively. Note that the green dash line is the result with $\beta = 0$.

the number of forward and backward passes keeps the same with BYOL. In addition, thanks to the efficient multi-stage augmentation, where we generate double instances using the nearly same resources. The main burden training time part of our proposed method may come from two more vector multiplication in our loss function 5.5. From Table 5.7, we can see that our method is as efficient as BYOL, and the differences between the two methods are less than 9% across three datasets.

5.4.6 Ablation Studies

In this section, we investigate the importance of the hyper-parameter β_{base} in Eq 5.6, which controls the initial weights of aggressive-aggressive and aggressive-weak augmented pairs in Eq 5.5. Figure 5.4 reports the results with linear classification on three datasets. First of all, we can observe that β_{base} makes remarkable contribution for the improvements compared with $\beta_{base} = 0$, with +0.4% (total increments +0.6%) for STL-10 and +0.7% (total increments +2.2%) for CIFAR-100. This suggests that removing aggressive-aggressive augmented pairs with $\beta_{base} = 0$ will limit the full exploration of the diversity of aggressive augmentation. However, if β_{base} is too large, aggressive-weak augmented pairs will make less contribution against noisy impacts.

Table 5.8: Performance comparison with linear classification for 200 epochs. The mean and standard deviation are computed over three trials.

Method	CIFAR-100	STL-10	ImageNet-100
SimSiam	55.8±0.9	85.0±0.4	82.6
Simsiam + RSA	**63.7±0.5**	**89.2±0.5**	**84.0**

5.4.7 Adaptability Studies

To evaluate the adaptability, we implement the proposed method based on SimSiam [21], termed SimSiam + RSA. Specifically, we change the same similarity function from mean square error to negative cosine similarity and adopt the same settings for all the hyper-parameters mentioned in Section 5.4.1. As illustrated in Table 5.8, we observe that the proposed Simsiam + RSA achieves better transfer performance than SimSiam on three datasets. For instance, RSA significantly raises the accuracy of linear probing from 55.8% to 63.7% (+7.9%) on the CIFAR-100 dataset. Therefore, the proposed method does not rely on the momentum network, demonstrating the adaptability of RSA.

5.5 Summary

In this chapter, we first empirically demonstrate that two positive instances generated by the aggressive augmentations can cause the semantic shift issue, which introduces noisy pairs and degrades the quality of learned representation. To alleviate this issue, we propose a novel method RSA, which dynamically adjusts the weights of clean and noisy pairs based on the memorization effects. Experimental results show that our proposed method achieves state-of-the-art results on various datasets and consistently improves the generalization on a series of downstream tasks.

The main limitation of this chapter is that, although we know there is noise from aggressive augmentations, the specific conditions under which noise occurs and how much DNNs will suffer from it are still unknown. We leave it as the future research direction.

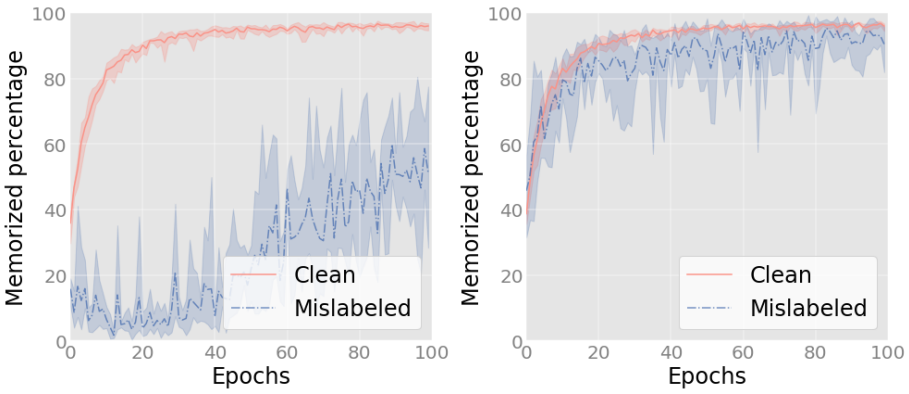
Chapter 6

Limitations of Early Stopping

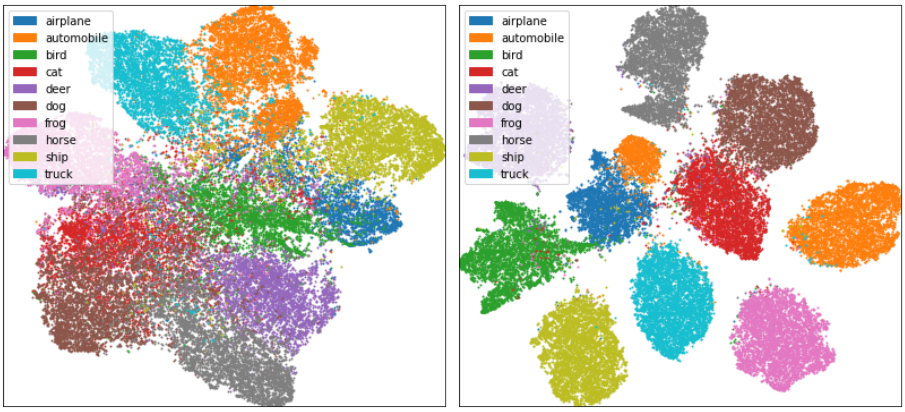
Currently, methods based on early stopping have become a mainstream research approach in the label noise learning community and are widely used in numerous real-world applications. However, the effectiveness of early stopping across all types of label noise remains in question. Our straightforward answer is no. In this chapter, we empirically investigate a previously overlooked and widespread type of label noise, subclass-dominant label noise (SDN). Our findings reveal that, during the early stages of training, deep neural networks can rapidly memorize mislabeled examples in SDN. This phenomenon poses challenges in effectively selecting confident examples using conventional early stopping techniques. To address this issue, we delve into the properties of SDN and observe that long-trained representations are superior at capturing the high-level semantics of mislabeled examples, leading to a clustering effect where similar examples are grouped together. Based on this observation, we propose a novel method called NoiseCluster that leverages the geometric structures of long-trained representations to identify and correct SDN. Our experiments demonstrate that NoiseCluster outperforms state-of-the-art baselines on both synthetic and real-world datasets, highlighting the importance of addressing SDN in learning with noisy labels.

6.1 Motivations and Contributions

This chapter presents a new type of label noise, subclass-dominant label noise (SDN), which is overlooked by existing studies. SDN refers to label noise in which mislabeled examples dominate at least one subclass, whose



(a) Sym **(b)** SDN



(c) SDN 10th epoch **(d)** SDN 100th epoch

Figure 6.1: (a, b) It is evident that early stopping can effectively separate symmetric label noise (Sym), but it proves ineffective for subclass-dominant label noise (SDN) as the network initially memorizes mislabeled examples.**(c, d)** Using the same settings as in (b), it becomes apparent that representations from the 100th epoch exhibit superior geometric structures compared to those from the 10th epoch. Specifically, the boundaries between classes become clearer, allowing for better distinguishability of correctly labeled and mislabeled examples.

examples exhibit distinct characteristics compared to others in the same class. Real-world datasets are prone to experiencing SDN due to several factors. The likelihood of human mislabeling is influenced by various object characteristics such as shape, context, and part [127, 71, 14, 114, 13]. Moreover, classification tasks usually define classes based on typical or common examples. However, in reality, numerous special sub-classes can exist, sharing common characteristics with other classes. For instance, while whiskers typically play a vital role in distinguishing between cats and dogs, Sphynx

cats, unlike other cat breeds, do not possess whiskers. An inexperienced annotator may mislabel the majority of Sphynx cats as dogs, leading to SDN in the dog class.

To evaluate the detrimental effects of SDN, we construct an SDN dataset and conduct an experiment on it. Specifically, we subdivide the aircraft class in CIFAR-10 [59] into two sub-classes based on their status: landed and flying. We generate the SDN dataset by flipping the labels of all landed airplane examples to the automobile class, leading to an overall noise rate of 2.7%. Our experiment involves training a ResNet-18 model [45] for 100 epochs using cross-entropy loss and extracting confident examples at each epoch. We identify confident examples based on the criteria described in [108]. To quantify the extent to which the network has memorized correctly labeled and mislabeled data, we compute the ratio of confident examples that include mislabeled data to the total number of mislabeled examples in the training dataset. As shown in Figure 6.1(a, b), early stopping fails to identify mislabeled examples because the network has already memorized the mislabeled examples in SDN. This outcome can be attributed to the fact that DNNs excel at learning from the majority of examples that share similar characteristics first. When the majority of examples in a clean subclass are mislabeled, these mislabeled examples often dominate the gradient and are learned first by DNNs, thereby posing challenges for methods that rely on early stopping.

To overcome this challenge, we explore a novel method named NoiseCluster, which is capable of distinguishing correctly labeled and mislabeled examples in SDN. The fundamental principle of NoiseCluster is rooted in the concept of later stopping. Contrary to the conventional belief that noisy data increasingly degrade representations over time, our findings suggest that the long-trained representations obtained from later stopping are more effective at capturing the high-level semantics of noisy examples. This leads to a clustering effect where the embeddings of noisy examples with similar characteristics are drawn closer together, as shown in Figure 6.1(c, d). Motivated by this observation, NoiseCluster utilizes the geometric structures of long-trained representations to identify and correct mislabeled examples of SDN. Specifically, we first extract features from the penultimate layer

of the network after halting the training progress with later stopping. We then group these features by class, based on feature density, and identify the largest group as clean due to its high likelihood of being correctly labeled, while treating the remaining groups as potentially mislabeled. To determine the label of a potentially mislabeled group, we assess its similarity to surrounding examples according to set distance and assign it the class with the most similar examples.

6.2 Related Work

Learning with noisy labels is a vibrant research area aimed at improving model robustness. Two main avenues of research have emerged: noise-modeling-based and memorization-effects-based methods.

Noise-modeling-based methods. From a noise modeling perspective, label noise can be categorized into three types: random classification noise (RCN), class-conditional label noise (CCN), and instance-dependent label noise (IDN). RCN maintains a static flip rate for binary classification [73]. CCN assumes that the noise generation is solely dependent on classes and independent of instances [97, 135, 122, 22]. The generation of IDN depends on both classes and instances, making IDN the most general type of label noise [127, 23, 146, 76]. The common philosophy behind noise modeling is to estimate the noise transition matrix. Xia et al. [127] assume annotators may make mistakes on the parts of instances rather than on whole instances, and replace the estimation of the instance-dependent transition matrix with the transition matrices for parts; Yang et al. [133] propose a method to generate Bayes optimal labels and estimate the transition matrix between Bayes optimal labels and noisy labels. Although noise-modeling-based methods do not theoretically require early stopping, many of them implicitly employ it to accurately estimate the transition matrix [97, 128, 131, 135, 133]. As a result, many existing methods struggle to handle SDN.

Memorization-effects-based methods. Arpit et al. [5] find that Deep Neural Networks (DNNs) tend to learn simple (clean) examples first,

then gradually memorize complex (noisy) examples. This observed phenomenon has inspired a subset of methodologies grounded in understanding and leveraging memorization effects [42, 6, 79, 52, 53]. JointOptim [108] introduces a pseudo-labeling technique known as hard-label, which regards the model predictions consistent with noisy labels as confident examples and uses them to train the model. DivideMix [65] utilizes Gaussian Mixture Model (GMM) to identify confident examples as labeled data and treat the remaining examples as unlabeled data for semi-supervised learning (SSL). Furthermore, some methods focus on the latent representations extracted from previous layers, as the final layer is particularly vulnerable to the influence of noisy examples [63, 121, 57, 7]. TopoFilter [121] utilizes k-NN to explore the topological information of the latent representations and then selects the largest component as clean data. Kim et al. [57] propose a noise detection method called FINE that utilizes Eigen decomposition and the principal component to select clean examples. PES [7] takes advantage of the fact that the effects of noisy examples tend to decrease as the layer number increases and improves early stopping by re-learning later layers based on the latent representations. NoiseCluster also exploits the geometric structures of the latent representations. However, unlike previous methods that rely on early stopping, NoiseCluster takes advantage of later stopping to extract long-trained representations, enabling the identification and correction of mislabeled examples in SDN.

6.3 SDN Analysis

In this section, we begin by defining subclass-dominant label noise (SDN). Next, we introduce a specially crafted dataset, CIFAR20-SDN, for exploring SDN, and compare it with two established real-world datasets. Finally, we delve into the characteristics of SDN from two distinct perspectives: its adverse effects on early stopping and its negative impact across various layers.

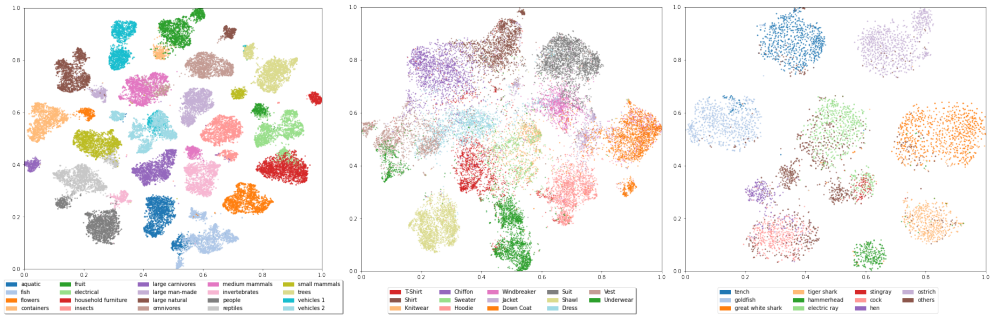
Definition of SDN. Let $\widetilde{D}$ denote the joint probability distribution of a pair of random variables $(X, \widetilde{Y})$, where X represents the feature instances and $\widetilde{Y}$ denotes the variable of noisy labels. The label space is defined as

$\{1, ..., C\}$, with C indicating the number of classes. Each x_i is associated with a latent (not provided) clean subclass label $b_i \in \{1, ..., B\}$, and B is the total number of sub-classes in this dataset. It is assumed that each clean subclass b corresponds to a single clean class c. Subclass-dominant noise (SDN) refers to a type of label noise in which mislabeled examples dominate at least one subclass. Specifically, within a clean subclass b, examples could be flipped into different classes, including its own clean class. If the class $j \in \{1, ..., C\}$ with the most flipped examples does not equal to the clean class c, then the subclass b is considered to be dominated by the examples flipped into noisy class j.

CIFAR20-SDN. To facilitate research on SDN, we introduce a representative SDN dataset, CIFAR20-SDN, built from CIFAR-100, which provides 20 class labels and 100 subclass labels. The generation of CIFAR20-SDN is similar to that of Pairflip label noise [42]. Specifically, for each class, a certain percentage of labels in the last subclass are flipped to the subsequent class. For example, the first class in CIFAR20 is termed "aquatic mammals", which comprises five sub-classes: "beaver", "dolphin", "otter", "seal", and "whale". Labels from the "whale" subclass are randomly flipped to the subsequent "fish" class. As a result, any "whale" labeled as "fish" is deemed an instance of SDN. Note that only the class labels are utilized for model training and evaluation.

To assess the similarity between CIFAR20-SDN and real-world noisy datasets, we compare the feature distribution of CIFAR20-SDN with two other real-world noisy datasets, Clothing1M [130] and WebVision [67]. Specifically, we directly visualize the features of CIFAR20-SDN, as it exclusively contains SDN, while for the two real-world datasets, we visualize the features of confident examples extracted using early stopping methods. As shown in Figure 6.2, it is clear that CIFAR20-SDN successfully emulates the properties of the two real-world datasets, with mislabeled examples clustering together and maintaining a distance from their noisy classes.

Negative effects of SDN on different layers. To thoroughly study the negative impacts of SDN on various layers, we train a PreAct ResNet-18 model on CIFAR20-SDN with four types of label noise. We report the highest test accuracy achieved during training as an indicator of the



(a) CIFAR20-SDN **(b)** Clothing1M **(c)** WebVision

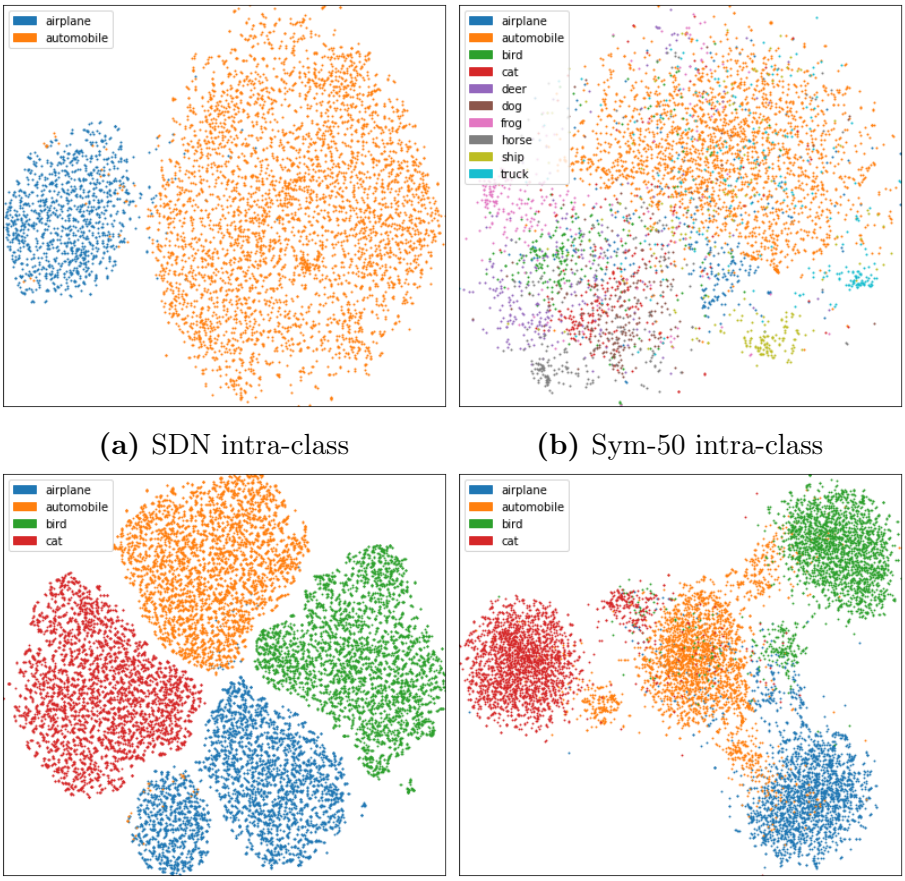
Figure 6.2: Comparison of feature distribution between CIFAR20-SDN and two real-world noisy datasets, with colors corresponding to clean labels. For Clothing1M, we utilize a provided evaluation set containing both clean and noisy labels for each example, employing the early stopping process described in Section 6.5.1. For WebVision, a DivideMix model is employed to generate features for the first ten classes; "clean" labels for these are inferred using models pretrained on ImageNet.

Table 6.1: Comparison of noisy impacts on different layers with different types of label noise. Training with clean labels terms as Clean; "Sym" is short for symmetric label noise; "IDN" is short for instance-dependant label noise; and "Pair" is short for pairflip label noise. "-XX" indicates the noise rate. The mean and standard deviation (percentage) computed over five runs are presented.

Noise type	Clean	Sym-20	IDN-20	Pair-20	SDN-20
Final	85.17±0.32	72.25±0.21	72.41±0.22	73.39±0.48	68.50±0.19
Penultimate	85.28±0.35	73.43±0.67	76.51±0.89	78.77±0.34	81.67±0.17

quality of the representations from the final layer. Additionally, to assess the representation quality from the penultimate layer, we freeze the other layers, re-initialize and retrain the final layer using clean labels.

Table 6.1 reveals that SDN significantly impacts the final layer more detrimentally than other types of label noise, resulting in the lowest test accuracy, which is roughly 80% of the accuracy achieved on the clean dataset. Conversely, when considering the penultimate layer, the outcomes achieved with SDN surpass those obtained with any other type of label noise, indicating that previous layers are less adversely impacted even after 300 epochs of training. These findings underscore the importance of utilizing the representations extracted from the penultimate layer in addressing



(a) SDN intra-class **(b)** Sym-50 intra-class

(c) SDN inter-class **(d)** Sym-50 inter-class

Figure 6.3: (a, b) illustrates the intra-class relationships between correctly labeled and mislabeled examples in a noisy class, while (c, d) demonstrates the inter-class relationships between mislabeled examples and their corresponding latent clean classes. To better visualize the intra-class feature relationships, we use the features from a noisy *automobile* class and visualize them with clean labels. Similarly, to effectively visualize inter-class feature relationships, we use features from four classes, as 2D images cannot adequately represent high-dimensional relationships.

SDN.

Advantages of long-trained representations. We extend the experiments depicted in Figure 6.1 and visualize the features extracted from the networks trained for 200 epochs. As shown in Figure 6.3(a, b), features within a noisy class are clustered based on their semantics, which suggests that networks can learn the semantics of mislabeled examples even when trained with noisy labels. Moreover, Figure 6.3(c, d) shows that features

of mislabeled examples tend to be close to the features with corresponding clean classes, implying that the geometric structures of the features can be utilized to identify their clean classes. Finally, a comparison of the four sub-figures reveals that the features extracted from SDN exhibit superior geometric structures compared to those extracted from symmetric label noise. This observation suggests that the properties of long-trained representations are particularly advantageous when dealing with SDN.

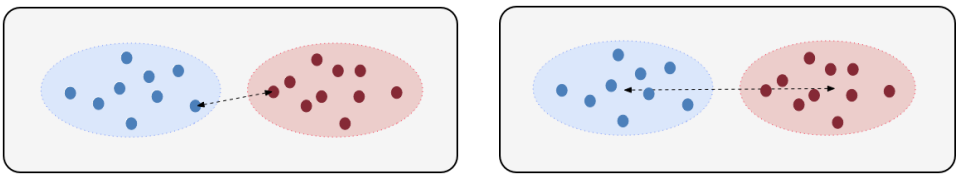
6.4 Methodology

To overcome the challenges posed by SDN, we introduce a novel method called NoiseCluster, which leverages the geometric structures of long-trained representations. NoiseCluster consists of two main steps: initially, it identifies potentially mislabeled examples by clustering similar instances into multiple sets. Subsequently, it performs label correction by computing set distance and reassigning the labels of potentially mislabeled examples.

6.4.1 Identify SDN

NoiseCluster begins with training on a noisy dataset, $(X, \widetilde{Y})$ and then halts the training with later stopping, whose stopping point is much later than that of early stopping, allowing for thorough exploitation of knowledge in noisy data. We then extract the latent feature Z of the input data X from the penultimate layer and reduce its dimensionality to two dimensions using t-SNE [84], resulting in a new feature denoted as $\hat{Z}$. To separate mislabeled data, we explore the geometric structures of $\hat{Z}_c$, where $c \in \{1, ..., C\}$ and C is the total number of classes. To handle the unknown number of clusters in each class, we employ DBSCAN [32], a feature density-based clustering algorithm, to cluster the feature $\hat{Z}_c$ into K groups U_K^c. The largest group, which is most likely to contain correctly labeled examples, is identified as the clean group, while the remaining groups are regarded as potentially mislabeled groups and denoted as U_{K-1}^c.

Stopping point selection. Although NoiseCluster can be applied to a fully trained model, we find that further training on the original noisy data does not constantly improve the representations after the training



(a) Set distance **(b)** Mean distance

Figure 6.4: The figure compares two methods for quantifying the similarity between two sets. Compared to mean distance shown in (b), set distance can more accurately measure the similarity between a potentially mislabeled group and a subset of similar examples within the class sets, particularly when the classes have high diversity.

becomes stable. Therefore, to reduce training time, we incorporate our method into the middle stage of the training process.

6.4.2 Label Correction

Simply removing all examples in U^c_{K-1} can result in the loss of many important examples. To address this challenge, we treat examples in a potentially mislabeled group U^c_k as a cohesive unit, where all examples share the same label. We then assess the similarity between this group and the surrounding examples by computing distances and assigning examples in the group to the class with the most similar examples. Specifically, due to the clustering effects of long-trained representations, we treat the examples in U^c_k as a set, and form multiple class sets O_b from the surrounding examples, where $b \in \{1, ..., C\}$. We measure the distance between U^c_k and O_b using set distance, which is defined as the minimum Euclidean distance between any feature z_q, z_p from sets U^c_k and O_b, as depicted in Figure 6.4. The formula for set distance is defined as follows:

$$d(U^c_k, O_b) = \inf_{p \in U^c_k, q \in O_b} \{\|z_q - z_p\|_2\}, \tag{6.1}$$

where $\|\cdot\|_2$ indicates $L2$ Norm. In practice, to mitigate the potential impacts of mislabeled examples close to the set U^c_k, we utilize a specified number of closest distances, denoted as $ClosePoint$, and compute their average. We then correct the labels in U^c_k with the category of the class set

with the closest distance, d_{kb}, which is defined as follows:

$$d_{kb} = \inf_{b \in \{1,...,C\}} \{d(U_k^c, O_b)\}. \tag{6.2}$$

After acquiring the corrected data, we use them to continue training the network for the rest of the epochs. The whole process of NoiseCluster is summarized in Algorithm 4.

Combine with SSL. Semi-supervised learning (SSL) has enabled numerous methods to attain impressive benchmarks in LNL [65, 72, 57]. NoiseCluster can also take advantage of SSL techniques to enhance its performance. Specifically, for each class c, DBSCAN divides features $\hat{Z}_c$ into K distinct groups U_K^c and assigns a group index g_i^c to each feature $\hat{z}_i^c$. Examples that cannot be clustered into any group are regarded as "noise". To integrate SSL, we treat the DBSCAN noise as the unlabeled data and the remaining examples as the labeled data. Accordingly, we can summarize the labeled and unlabeled data sets as follows:

$$\begin{cases} \mathcal{D}_l = \{(x_i, \widetilde{y}_i) \mid g_i^c = 1, ..., K_c\} \\ \mathcal{D}_u = \{x_i \mid g_i^c \neq 1, ..., K_c\} \end{cases}. \tag{6.3}$$

Combine with early stopping. Although NoiseCluster is primarily designed to address SDN, it can be seamlessly integrated with early stopping. To tackle a noisy dataset containing multiple types of label noise, we propose a two-stage training process. In the first stage, we employ early stopping to select confident data, thereby filtering out label noise that is sensitive to early stopping. In the second stage, we apply NoiseCluster to the confident data to identify and correct mislabeled examples in SDN. This two-stage strategy allows us to effectively address the challenges posed by a mixture of various types of label noise, resulting in enhanced performance on noisy datasets.

Algorithm 4: *NoiseCluster*

Input: Network f_θ; Final layer f_ξ; Noisy training dataset $(X, \widetilde{Y})$; Number of epochs for long-trained N; Class number C; DBSCAN Eps and $MinPts$.
for $i = 1, \ldots, N$ **do**
    Train f_θ and f_ξ on $(X, \widetilde{Y})$; `// standard training`
end
$\hat{Z} \leftarrow tSNE(f_\theta(X))$;
for $c \leftarrow 1$ **to** C **do**
    $U_K^c \leftarrow DBSCAN(\hat{Z}_c, Eps, MinPts)$; `// identify SDN`
    for U_k^c ***in*** U_K^c **do**
        if $U_k^c \neq largest$ **then**
            Compute set distance with Eq. (6.1);
            Update $\widetilde{Y}$ in U_k^c with Eq. (6.2) ;
        end
        $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup U_k^c$;
    end
end
Continually train f_θ and f_ξ on $\mathcal{D}_l$ for the rest of the epochs.

6.5 Experiments

6.5.1 Experimental Setting

Datasets. We evaluate the performance of NoiseCluster on a synthetic dataset, CIFAR20-SDN, and a real-world dataset, Clothing-1M [130]. To ensure a fair comparison with existing methods, we follow the settings of the previous methods [65, 72, 7] and conduct two sets of experiments on the synthetic dataset: one with semi-supervised learning (SSL) and one without SSL. In experiments without SSL, we reserve 10% of the training data as the validation set, while we utilize the entire training data for experiments with SSL. Clothing-1M contains over one million images gathered from the Internet. Additional details on implementation can be found in Appendix C.2.

Network structure and optimization. For CIFAR20-SDN, we employ ResNet-34 [45] for experiments without SSL and PreAct ResNet-18 [46] for experiments with SSL. During optimization, we train the model for 300

Table 6.2: Comparison with state-of-the-art methods without SSL on CIFAR20-SDN. "-XX" indicates the noise rate. The mean and standard deviation computed over five runs are presented.

Methods	SDN-12	SDN-16	SDN-18	SDN-20
CE [45]	70.42±0.32	67.38±1.31	66.28±0.51	65.23±0.60
JointOptim [108]	71.48±0.33	67.69±0.37	66.51±0.28	65.45±0.20
DMI [131]	70.67±0.64	66.95±0.61	65.86±0.32	64.35±0.52
TopoFilter [121]	69.04±0.47	64.78±0.49	63.52±0.36	62.97±0.62
PTD-R-V [127]	70.85±0.44	66.45±0.40	65.29±0.47	64.41±0.20
VolMinNet [68]	72.45±0.38	67.78±0.43	66.20±0.24	65.20±0.41
CDR [124]	72.61±0.36	68.87± 0.16	67.79±0.28	67.00±0.25
BLTM-V [133]	70.46±0.47	66.57±0.50	64.76±0.42	63.96±0.46
NoiseCluster	**80.20±0.80**	**78.77±1.01**	**77.01±0.34**	**72.67±0.68**

epochs, using a learning rate of 2×10^{-2}, a single cycle of cosine annealing [78], a momentum of 0.9, and a weight decay of 5×10^{-4}. We utilize a batch size of 128 and a stopping epoch of 80, with a ClosePoint value of 20. For DBSCAN hyperparameters, Eps and MinPts are set to 0.02 and 100, respectively.

For Clothing1M, we employ a ResNet-50 pre-trained on ImageNet[60], and train the network for only one epoch. Specifically, we first utilize 95% of the training data to train the network. Then, we use early stopping to select confident data from the remaining 5% of the training data and apply NoisyCluster to this data. For optimization, we use SGD with a learning rate of 0.01, a momentum of 0.9, a weight decay of 10^{-4}, and a batch size of 64. ClosePoint, Eps, and MinPts are set to 20, 0.04, and 100, respectively.

6.5.2 Comparison with State-of-the-art Methods

Approaches without SSL. To begin our comparison, we evaluate our method against state-of-the-art (SOTA) approaches that do not employ SSL. As shown in Table 6.2, it becomes evident that many of the methods struggle with SDN, with some even underperforming in comparison to standard training with Cross-Entropy (CE) loss. CDR [124] performs better than its counterparts, but it requires a known noise rate, which remains challenging to accurately estimate in SDN. Our method outperforms

Table 6.3: Comparison with state-of-the-art methods with SSL on CIFAR20-SDN. "-XX" indicates the noise rate. The mean and standard deviation computed over five runs are presented.

Methods	SDN-12	SDN-16	SDN-18	SDN-20
CE [46]	74.27±0.40	70.53±0.38	69.27±0.12	68.50±0.19
MixUp [141]	75.83±0.41	71.65±0.17	70.29±0.24	69.37±0.17
ELR+ [72]	75.22±0.37	71.09±0.30	70.18±0.23	69.55±0.15
DivideMix [65]	79.06±0.26	72.80±0.30	70.66±0.51	70.27±0.11
F-DivideMix [57]	79.71±0.32	74.00±0.83	71.69±0.22	69.59±0.43
PES semi [7]	80.27±0.30	73.95±0.24	72.78±0.22	71.72±0.45
NoiseCluster+	**82.20±0.93**	**80.35±1.23**	**77.59±0.86**	**73.54±0.72**

all baselines, delivering superior performance. Specifically, our proposed method surpasses CDR by 5.67% in classification accuracy when a noise rate is 20%, and the margin further extends to 7.59% for a noise rate of 12%. Furthermore, in comparison with noise-modeling-based methods, our approach outperforms the state-of-the-art VolMinNet by over 7% across four noise rates.

Approaches with SSL. We integrate our method with semi-supervised learning (SSL) techniques, denoted as NoiseCluster+ and compare it against SOTA approaches that also leverage SSL. To ensure a fair comparison, we include the results of MixUp [141], which is used by all baselines to boost performance. The results are summarized in Table 6.3. Interestingly, our observations indicate that SSL contributes marginally to performance enhancement on CIFAR20-SDN. Notably, DivideMix [65] shows only a modest advancement over MixUp. In contrast, methods based on representations, such as F-DivideMix [57] and PES semi [7], deliver better outcomes by avoiding the use of the final layer, which is prone to overfitting to SDN. Conclusively, our method, NoiseCluster+, consistently outperforms all baselines by a substantial margin, underscoring that the proposed techniques are the primary factors leading to the improvements.

To better understand why current SOTA methods face difficulties when dealing with SDN, we conduct experiments with DivideMix and compute the ratio of confident examples that include mislabeled data to the total

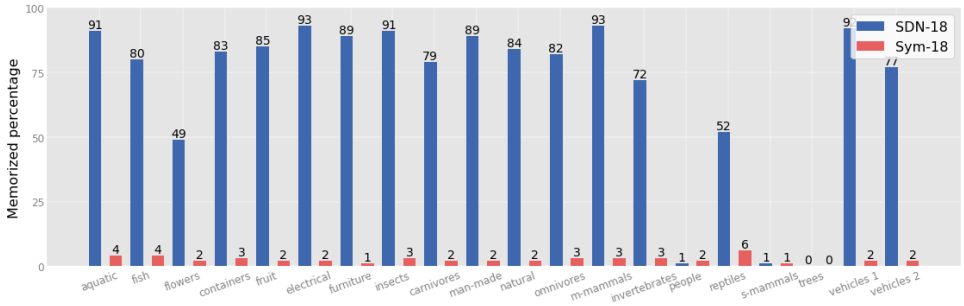


Figure 6.5: Evaluation memorized percentage per class obtained with DivideMix on CIFAR20-SDN (SDN) and symmetric label noise (Sym). "-XX" indicates the noise rate.

Table 6.4: Classification accuracy (%) on Clothing1M. Baseline results are taken from the original papers. * indicates the results obtained with an ensemble model.

CE	DMI	JointOptim	TopoFiler	PTD-R-V	VolMinNet
69.21	71.67	72.16	74.10	72.46	72.42
BLTM-V	F-DivideMix*	PES	DivideMix*	ELR+*	NoiseCluster
73.39	74.37	74.64	74.76	74.81	**75.51**

number of mislabeled examples in the training dataset at the end of training. As shown in Figure 6.5, DivideMix performs well on merely three classes. For the remaining classes, we can see that the confident data contains a substantial number of mislabeled examples, confirming that early stopping and other techniques utilized in DivideMix cannot effectively address SDN issues.

Real-world dataset. We perform experiments on a real-world dataset, Clothing1M, which is widely used as a benchmark due to its challenges. Many existing methods, though effective on noisy synthetic datasets, exhibit limited impact on Clothing1M. Our investigation suggests that this issue stems from the presence of SDN in Clothing1M, leading to suboptimal performance on the test data. As illustrated in Table 6.4, NoiseCluster significantly outperforms SOTA methods, achieving an accuracy of 75.51% without resorting to self/semi-supervised learning or ensemble network techniques.

(a) Sleep T-Shirt **(b)** Leather Jacket **(c)** Down Vest

Figure 6.6: SDN in Clothing1M. (a) The first row displays images from the "T-Shirt" class (X:0.45, Y:0.1). The second row shows images from the "Underwear" class. The third and fourth rows present images that are mislabeled as "Underwear". (b) The row order is as follows: "Jacket", "Windbreaker" (correct name:"trench coat"), "Leather Jacket" (X:0.75, Y:0.6). (c) The row order is as follows: "Down coat", "Vest" (correct name:"singlet" or "tank top"), "Down Vest" (X:0.9, Y:0.3). The coordinate values correspond to Figure 6.2 (b). It is worth noting that some label names within Clothing1M are inaccurate due to translation issues. After comparing images in the test set and Chinese label names, we put the correct names in parentheses.

To delve into the existence and underlying causes of SDN in real-world datasets, we manually generated t-SNE images, strictly adhering to the experimental process outlined in Section 6.5.1. We then showcase examples within clusters that are predominantly mislabeled. As Figure 6.6 illustrates, these mislabeled examples clearly belong to specific sub-classes, yet they share common characteristics with their associated noisy classes. These observations confirm the pervasive presence of SDN in real-world datasets. For additional details and further results, please refer to Appendix C.4.1.

6.5.3 Ablation Study

Sensitivity of hyper-parameters. We perform ablation studies using NoiseCluster on CIFAR20-SDN with noise rates of 12% and 20%. NoiseCluster employs DBSCAN [32], which involves two hyperparameters, Eps and MinPts. We find that the selection of these two hyperparameters is relatively straightforward when applied to feature representations

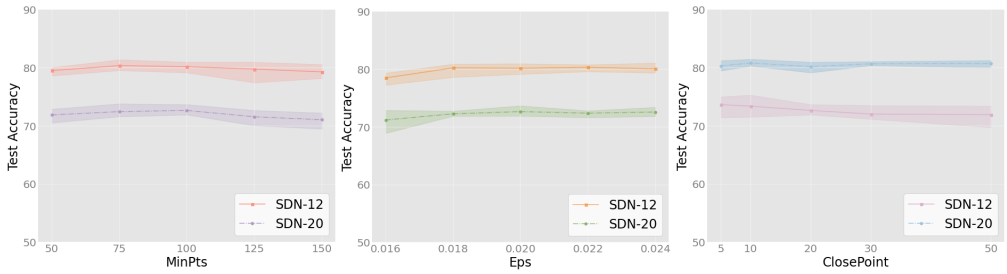
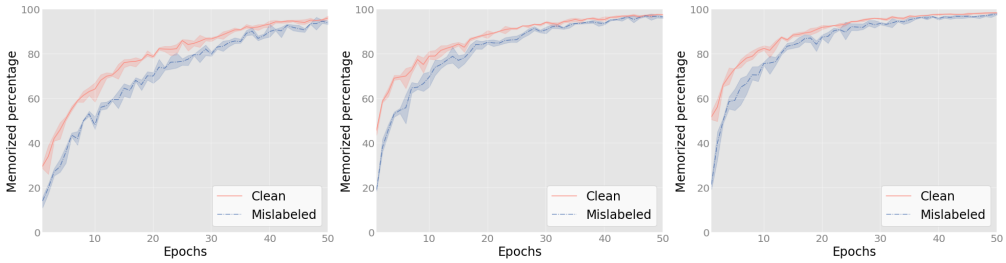


Figure 6.7: Ablation analyses of MinPts, Eps, and ClosePoint.

due to the substantial margin between classes. Additionally, to reduce the negative impacts of mislabeled examples, we introduce a hyperparameter named ClosePoint, which controls the number of points used to compute the set distance. The results shown in Figure 6.7 indicate that ClosePoint, along with Eps and MinPts, has a negligible impact on the final outcomes, indicating robustness against hyperparameter changes.

Self-supervised pretrained model. As self-supervised Learning methods continue to mature [43, 40, 145, 8], the utilization of pretrained models from this approach has emerged as a promising strategy for addressing noisy labels, particularly since they are unaffected by such labels during the pre-training stage [35, 132, 48, 144]. This raises the pivotal question: Does employing an self-supervised pretrained model simplify the addressing of SDN? While a thorough investigation will be reserved for future research work, the succinct answer is no. As shown in Figure 6.8, self-supervised pretrained models offer initial advantages by facilitating rapid learning of clean examples. However, this rapid learning also leads to the quick memorization of mislabeled examples. Furthermore, after only training for several epochs, the gaps between the memorization rates for clean and mislabeled examples become much smaller than those observed when training from scratch, suggesting that early stopping with self-supervised pretrained models is less effective. We hope that our research will inspire a re-thinking of utilizing self-supervised pretrained models in noisy datasets.

Importance of label correction. To mitigate the bias introduced by learning with SDN, we employ a label correction technique. Given that DNNs have already memorized many mislabeled examples during the early



(a) Scratch **(b)** MoCo V2[43] **(c)** RSA[8]

Figure 6.8: Examining memorization trends when training from scratch compared to training with self-supervised pretrained models on CIFAR20-SDN with an 18% noise rate. These models were pre-trained with MoCo V2 and RSA on the same dataset for 200 epochs, resulting in linear classification accuracies of 63.02 ± 0.31 and 77.67 ± 0.27, respectively. However, as the performance of pretrained models increases, the speed of overfitting to mislabeled data also becomes faster. The mean and standard deviation (percentage) computed over three runs are presented.

Table 6.5: Comparison with/without correcting labels on CIFAR20-SDN. "-XX" indicates the noise rate. The mean and standard deviation (percentage) computed over five runs are presented.

Methods	SDN-12	SDN-20
Ours	**80.20±0.80**	**72.67±0.68**
Ours w/o correction	78.68±0.61	71.42±0.54

learning phase, label correction becomes indispensable. In Table 6.5, we present a comparison of results with and without applying label correction and observe a performance improvement of over 1% for both low and high noise rates. These findings emphasize the importance of the label correction technique in addressing SDN.

6.6 Summary

In this chapter, we empirically examine subclass-dominant label noise (SDN), a new type of label noise that commonly occurs in real-world datasets. SDN presents a significant challenge to conventional methods that rely on early stopping. To overcome this issue, we reveal that long-trained representations are more effective at capturing the semantic relationships between

noisy examples. Building on this insight, we propose a novel approach, NoiseCluster, to identify and correct mislabeled examples in SDN. We hope our work will inspire the research community and pave the way for addressing label noise in real-world applications.

The primary limitation is that our proposed method is effective in addressing SDN cases where mislabeled examples have notably different semantics from correctly labeled examples. However, it could struggle in situations where correctly labeled and mislabeled examples exhibit close semantics, especially without any prior knowledge. We leave it as an open research topic for future investigation.

Chapter 7

Conclusion

In this thesis, we have conducted a meticulous examination of early stopping through four distinct lenses: strengths, advancements, applicability, and limitations. Our research has yielded several key insights and contributions. Firstly, we discovered that early stopping could be effectively used to extract hard confident examples. We also found that the impact of label noise varies across different layers, leading to the development of a progressive early stopping strategy. Additionally, we have shown that early stopping is versatile, addressing not only label noise issues but also related problems such as semantic shifts. Finally, our research uncovered the presence of subclass-dominant label noise, illustrating that early stopping is not universally effective against all forms of label noise. Building on these findings, we propose three methods based on early stopping and one incorporating later stopping, each tailored to overcome the identified challenges. Each method implemented has demonstrated promising outcomes, underscoring their potential effectiveness in their respective applications.

Future research could pursue two promising directions: First, the burgeoning field of self-supervised learning, with its increasing reliance on pseudo tasks, is likely to see a corresponding rise in the prevalence of task-generated noise. Investigating the application of early stopping across a broader array of pseudo tasks becomes essential. Second, given the inherent complexities associated with subclass-dominant label noise, developing and enhancing sophisticated methodologies to effectively surmount this challenge is crucial.

Appendix A

Supplementary for Chapter 3

A.1 Generating Label Noise

The label noise is generated according to symmetric class-dependent and instance-dependent noise transition matrices.

Noise transition matrix The *noise transition matrix* $T(x)$ was proposed to explicitly model the generation process of label noise, where $T_{ij}(x) = \Pr(\bar{Y} = j|Y = i, X = x)$, $\Pr(A)$ denotes as the probability of the event A, X as the random variable for the instance, $\bar{Y}$ as the noisy labels, and Y as the latent clean labels. At a high level, the ij-th entry of the transition matrix denotes the probability that the instance will flip from the clean class j to the noisy class i.

Symmetric class-dependent label noise If the flip rate is α, the diagonal entries of a symmetric transition matrix are $1 - \alpha$ and the off-diagonal entries are $\alpha/(c - 1)$ [42].

Instance-dependent label noise We generate instance-dependent label noise according to the following Algorithm 5 [127].

A.2 Comparison with SELF

The performance of our re-implemented SELF [91] is not as good as that in the original paper. For a fair comparison, we change our backbone consistently with SELF and compare with the results from the original paper directly (without Mean Teachers [109]). Specifically, the network is

Algorithm 5: Instance-dependent Label Noise Generation

Input: Clean samples $\{(x_i, y_i)\}_{i=1}^{n}$; Noise rate τ.
Sample instance flip rates $q \in \mathbb{R}^n$ from the truncated normal distribution $\tau\mathcal{N}(\tau, 0.1^2, [0, 1])$;
Independently sample $w_1, w_2, \ldots, w_c$ from the standard normal distribution $\mathcal{N}(0, 1^2)$;
For $i = 1, 2, \ldots, n$ do
  $p = x_i \times w_{y_i}$; // generate instance-dependent flip rates
  $p_{y_i} = -\infty$; // control the diagonal entry of the instance-dependent transition matrix
  $p = q_i \times softmax(p)$; // make the sum of the off-diagonal entries of the y_i-th row to be q_i
  $p_{y_i} = 1 - q_i$; // set the diagonal entry to be $1 - q_i$
  Randomly choose a label from the label space according to possibilities p as noisy label $\bar{y}_i$;
End for.
Output: Noisy samples $\{(x_i, \bar{y}_i)\}_{i=1}^{n}$

Table A.1: Means and standard deviations of classification accuracy compared with SELF on *CIFAR10*

	Sym-40	Sym-60
SELF	87.35%	75.47%
Ours	**92.31%**	**87.88%**

changed to ResNet26 with Shake-shake regularization [34]. The learning rate is set to 0.05 with weight decay of 2e-4.

Table A.1 and Table A.2 show that Me-Momentum outperforms SELF by a large margin in *CIFAR10* and *CIFAR100* with Symmetric 40% and Symmetric 60% noise. The gap between the performance of Me-Momentum and SELF becomes larger in Symmetric 60% in both datasets compared with Symmetric 40% because hard confident examples play a more important role in more noisy examples. Specifically, both of the methods are based on the same backbone with Cross-Entropy loss, so the improvement of Me-Momentum can only be as a result of the quality of extracted confident examples. Therefore, Me-Momentum is able to extract better hard confident examples than SELF.

Furthermore, the performance of Me-Momentum can be improved by

Table A.2: Means and standard deviations of classification accuracy compared with SELF on *CIFAR100*

	Sym-40	Sym-60
SELF	61.40	50.60
Ours	**68.25**	**59.51**

Table A.3: Means and standard deviations of classification accuracy on symmetric 50% label noise with different datasets

Methods	MNIST	CIFAR10	CIFAR100
Cross-Entropy	97.51 ± 0.28	77.11 ± 0.43	39.73 ± 2.74
MentorNet	90.13 ± 0.09	70.71 ± 0.24	38.45 ± 0.25
Co-teaching	91.68 ± 0.21	72.80 ± 0.45	51.60 ± 0.49
Forward	97.86 ± 0.22	77.92 ± 0.66	38.59 ± 1.62
Joint Optim	97.79 ± 0.13	85.00 ± 0.17	57.97 ± 0.67
DMI	97.04 ± 1.15	78.28 ± 0.48	49.81 ± 1.22
T-revision	98.38 ± 0.21	83.40 ± 0.65	57.71 ± 0.84
CDR	98.13 ± 0.17	82.64 ± 0.89	55.30 ± 0.96
Ours	**98.52 ± 0.09**	**86.40 ± 0.34**	**58.06 ± 0.59**

changing a better backbone. However, to show the effectiveness of the proposed method and avoid complexity, we choose the standard CNN network in the chapter.

A.3 Experiments on High Noise Rates

In Table A.3, we evaluate our method on Symmetric 50%. Note that we only use confident examples, discarding non-confident ones. If the noise rate is too high, e.g., noise rate 80%, the number of extracted examples might be insufficient to train a model adequately. This could be addressed by combining with semi-supervised learning techniques [91, 65]. However, our aim is to verify the method's effectiveness in extracting high-quality confident examples, not to boost classification performance.

Table A.4: Means and standard deviations of classification accuracy using noisy and clean validation sets on *CIFAR10*.

	noisy validation set	clean validation set
Sym-20	91.44 ± 0.33	91.60 ± 0.31
Sym-40	88.39 ± 0.34	89.14 ± 0.51
Sym-50	86.40 ± 0.34	86.88 ± 0.70
Inst-20	90.86 ± 0.21	91.34 ± 0.24
Inst-40	86.66 ± 0.91	87.80 ± 0.84

A.4 Me-Momentum with Clean Validation Sets

We conduct experiments with clean validation sets. In Table A.4, we can observe that a clean validation set aids in selecting a better model compared to a noisy validation set. This leads to improved final performance.

A.5 Comparison of Training Time

We compare the training time with representative baselines on *CIFAR10* with ResNet18 in Table A.5. The number of the inner loop varies because the inner loop stops by exploiting a noisy validation set. Note that we discard non-confident examples and only use confident examples to train the model. The training time in each round would be much less than the normal training.

Table A.5: Comparison of training time on *CIFAR10*.

Methods	Training time
CE	68 min
JointOptim	88 min
Co-teaching	91 min
T-revision	232 min
CDR	178 min
Ours Sym-50	169.4 ± 18.8 min
Ours Inst-40	160.0 ± 21.0 min

Note that CE stands for the normal neural network training by employing the cross-entropy loss function. The total number of epochs for CE

is set to 200. Since the number of the inner loop varies in the proposed method, we have reported the average training time of five runs and the standard deviation. Note that Me-momentum has a smaller training time on the instance-40% noise than that on the symmetric-50% noise. This may be caused by that the former setting is more difficult than the latter one and less confident examples are extracted. The conclusion is that the training time of the proposed method is shorter than T-revision and CDR but longer than Co-teaching, and JointOptim.

A.6 Experimental details for Clothing1M

Clothing1M contains 1 million examples with real-world noisy labels, the number of examples cross classes are significantly imbalanced. Moreover, the ratio of examples cross classes in the training dataset is also dramatically different from that of the validation and test dataset.

A common way to deal with the imbalance is to make the number of training examples in each class the same. That is to say, we need to reduce the number of training examples for different classes to the least number of the classes. Therefore, the method cannot fully use training examples, especially on an extremely imbalanced training dataset. We take two approaches to deal with it. 1) We introduce the importance reweighting technique to different examples [50]. Suppose the number of samples for class j on the training (validation) dataset is N^j_{train} (resp. N^j_{val}). The weight of samples in class j is calculated as:

$$W^j = \frac{\frac{1}{C}\sum_{j=1}^{C} N^j_{train}}{N^j_{train}} \times \frac{N^j_{val}}{\frac{1}{C}\sum_{j=1}^{C} N^j_{val}}, \tag{A.1}$$

where C is the total number of classes.

2) We extract the confident examples separately according to the class. Specifically, we select a classifier based on the highest score in each class, where the score S for class j is obtained with:

$$S^j = \beta \times Accuracy^j + (1-\beta) \times Precision^j. \tag{A.2}$$

We set β to 0.45 in experiments. Then, we use the classifier to extract the confident examples in this class.

A.7 More Results in Section 3.3.1

In Section 3.3.1, we discuss the statistics of the extracted confident examples. However, due to the space limit, in the chapter, we only provide parts of the empirical results.

A.8 More Results in Section 3.3.2

In Section 3.3.2, we have visualized the extracted confident examples by using t-SNE [83]. It verifies that the proposed Me-Momentum is effective to extract hard confident examples. However, due to the space limit, in the chapter, we only provide parts of the empirical results. In this supplementary material, we provide the visualization of all the employed datasets and settings.

Specifically, we show how the confident examples are progressively extracted in the inner and outer loops. In the figures, green, blue, and red dots represent confident examples extracted at the beginning, middle, and end rounds of the loops, respectively.

On the datasets of *MNIST* and *CIFAR10*, we can clearly see that the blue and red dots are mostly located at the boundaries of the clusters of green dots. On *CIFAR100*, we can also clearly see that there are lots of blue and red dots which are outside of the green clusters in the second and fourth figures. This supports and justifies our claim that Me-Momentum is able to extract hard confident examples (those are close to the decision boundary).

The figures are presented on the following pages.

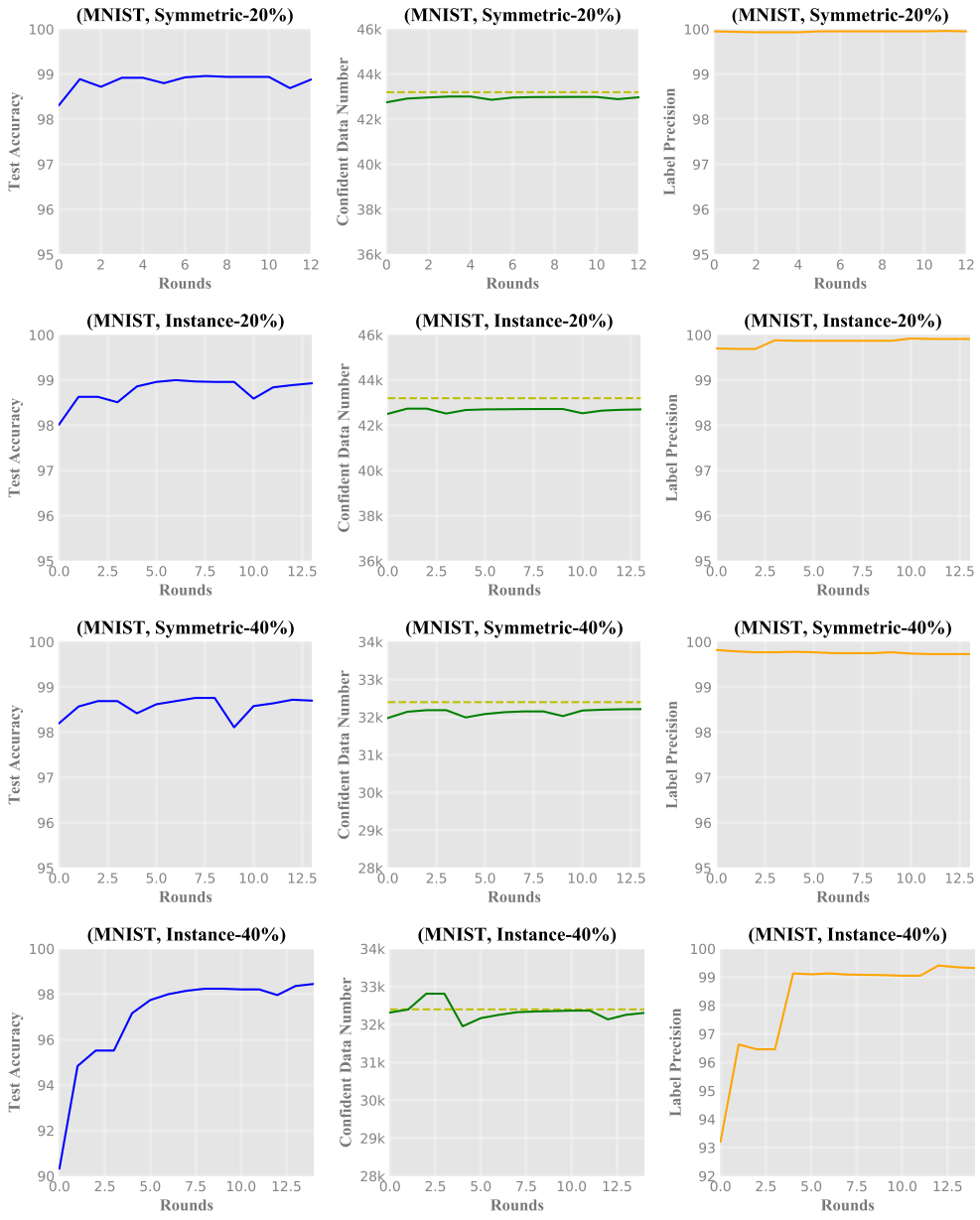


Figure A.1: Statistics of the extracted confident examples on *MNIST* by Me-Momentum. We call one update of the classifier and the extracted confident examples as one round. We illustrate how the label precision of the extracted confident examples, the number of the extracted confident examples, and the classification accuracy of the classifier trained by using the extracted confident examples change during the training of Me-Momentum. The dash lines in the middle column indicate the number of clean labels in the noisy training data.

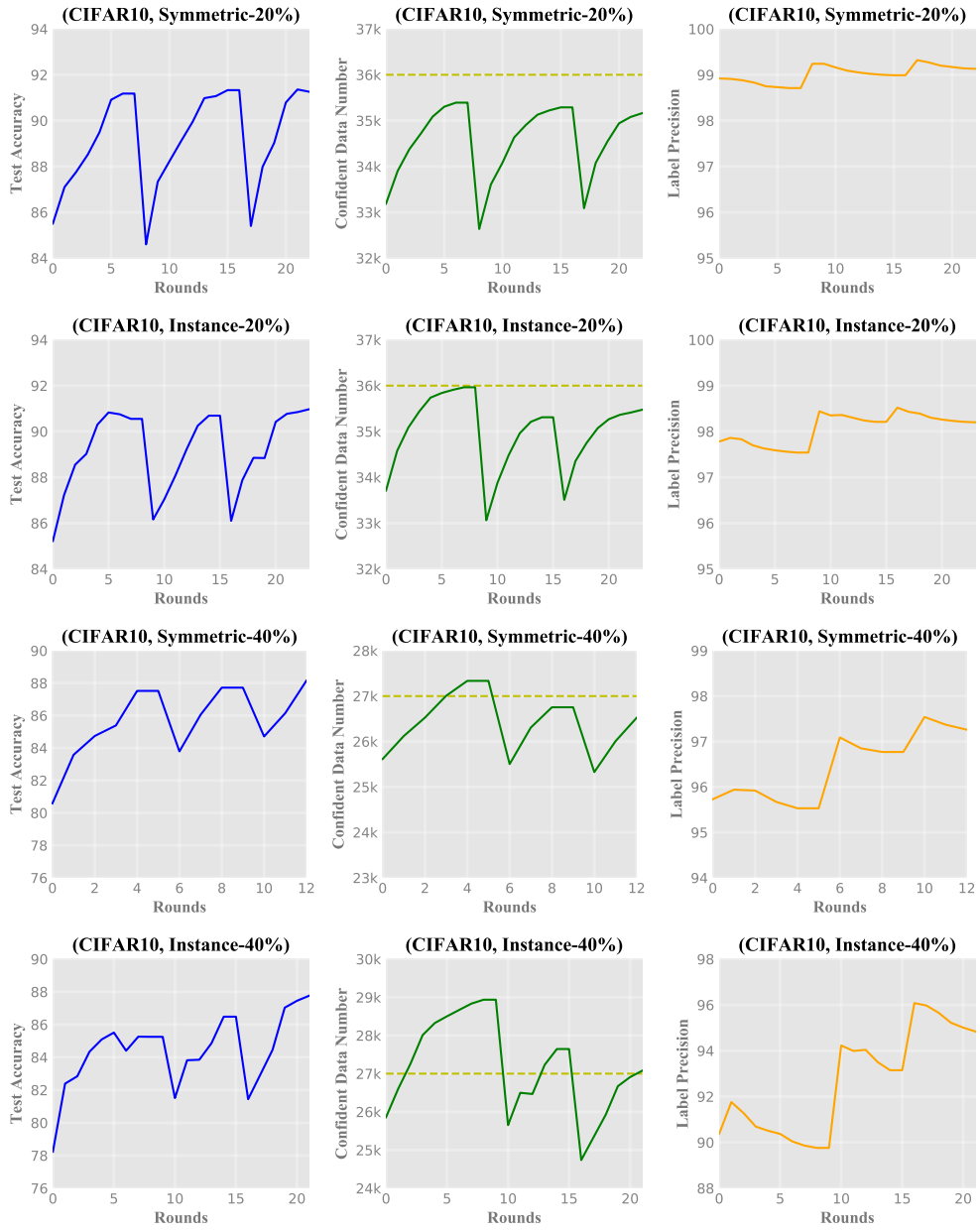


Figure A.2: Statistics of the extracted confident examples on *CIFAR10* by Me-Momentum. We call one update of the classifier and the extracted confident examples as one round. We illustrate how the label precision of the extracted confident examples, the number of the extracted confident examples, and the classification accuracy of the classifier trained by using the extracted confident examples change during the training of Me-Momentum. We have three distinct peaks in these figures because we have set $N_{\text{outer}} = 3$ and the classifiers are re-initialized in the outer loop. The dash lines in the middle column indicate the number of clean labels in the noisy training data.

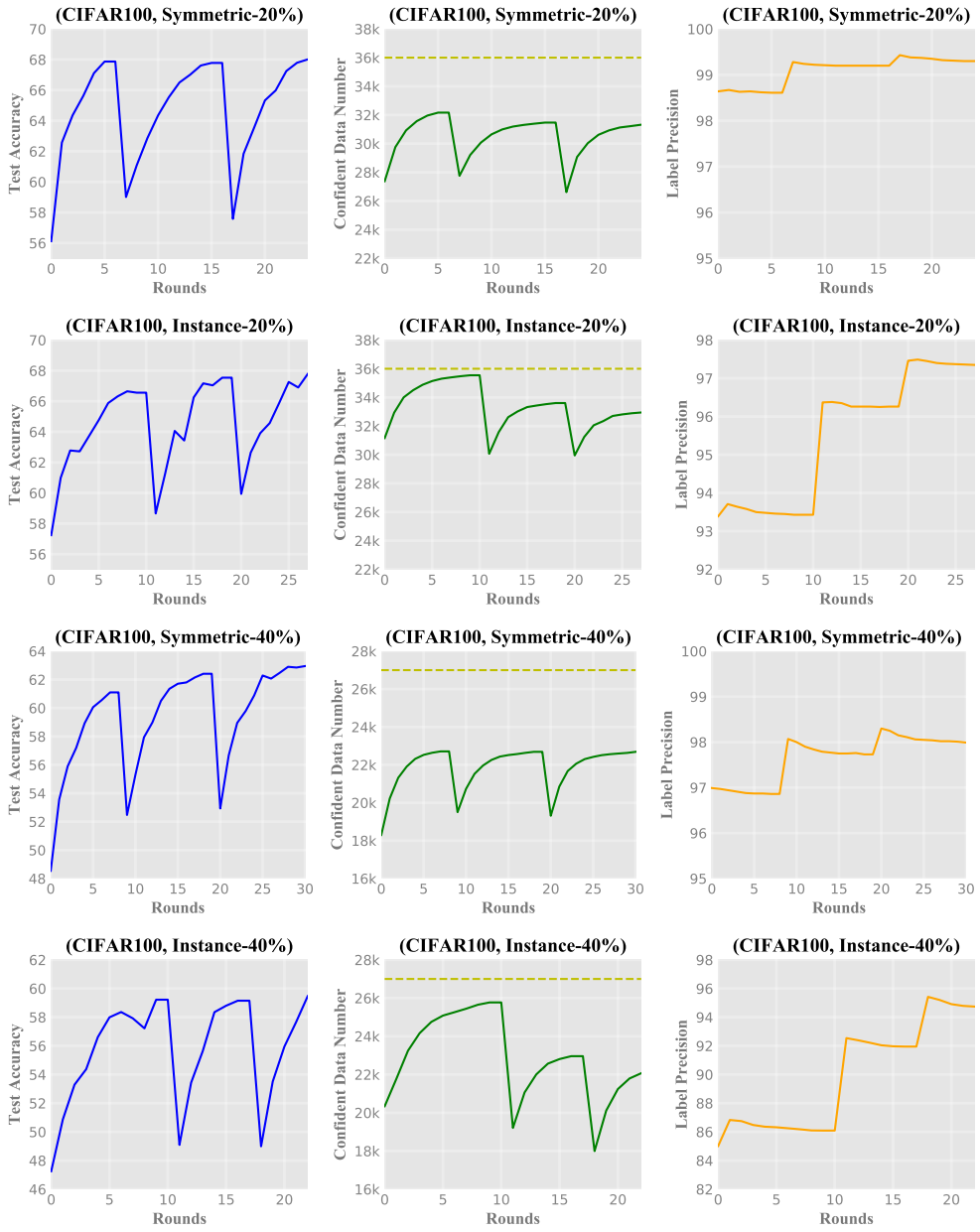


Figure A.3: Statistics of the extracted confident examples on *CIFAR100* by Me-Momentum. We call one update of the classifier and the extracted confident examples as one round. We illustrate how the label precision of the extracted confident examples, the number of the extracted confident examples, and the classification accuracy of the classifier trained by using the extracted confident examples change during the training of Me-Momentum. We have three distinct peaks in these figures because we have set $N_{\text{outer}} = 3$ and the classifiers are re-initialized in the outer loop. The dash lines in the middle column indicate the number of clean labels in the noisy training data.

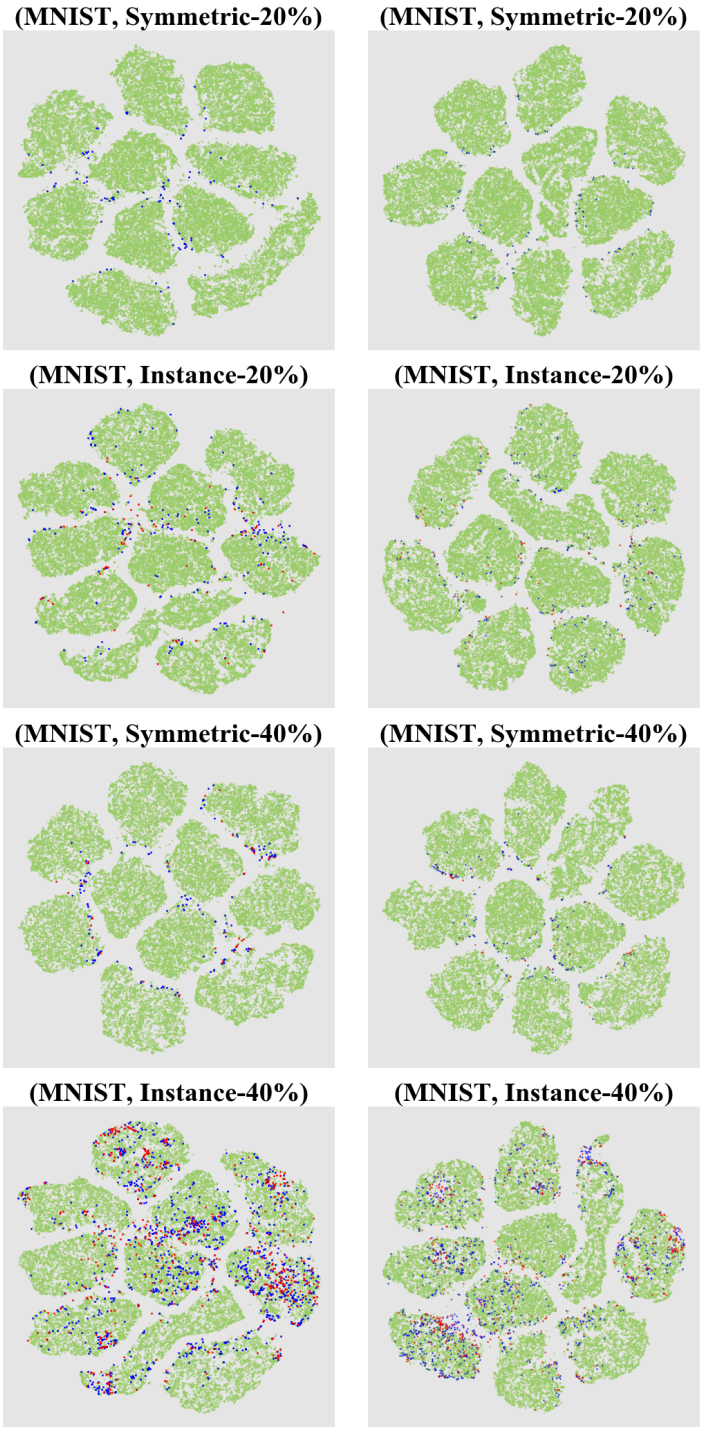


Figure A.4: Visualization of the extracted confident examples on *MNIST*. The first column is about the confident data extracted in the first run of the inner loop; while the second column is about the confident data extracted in the outer loop.

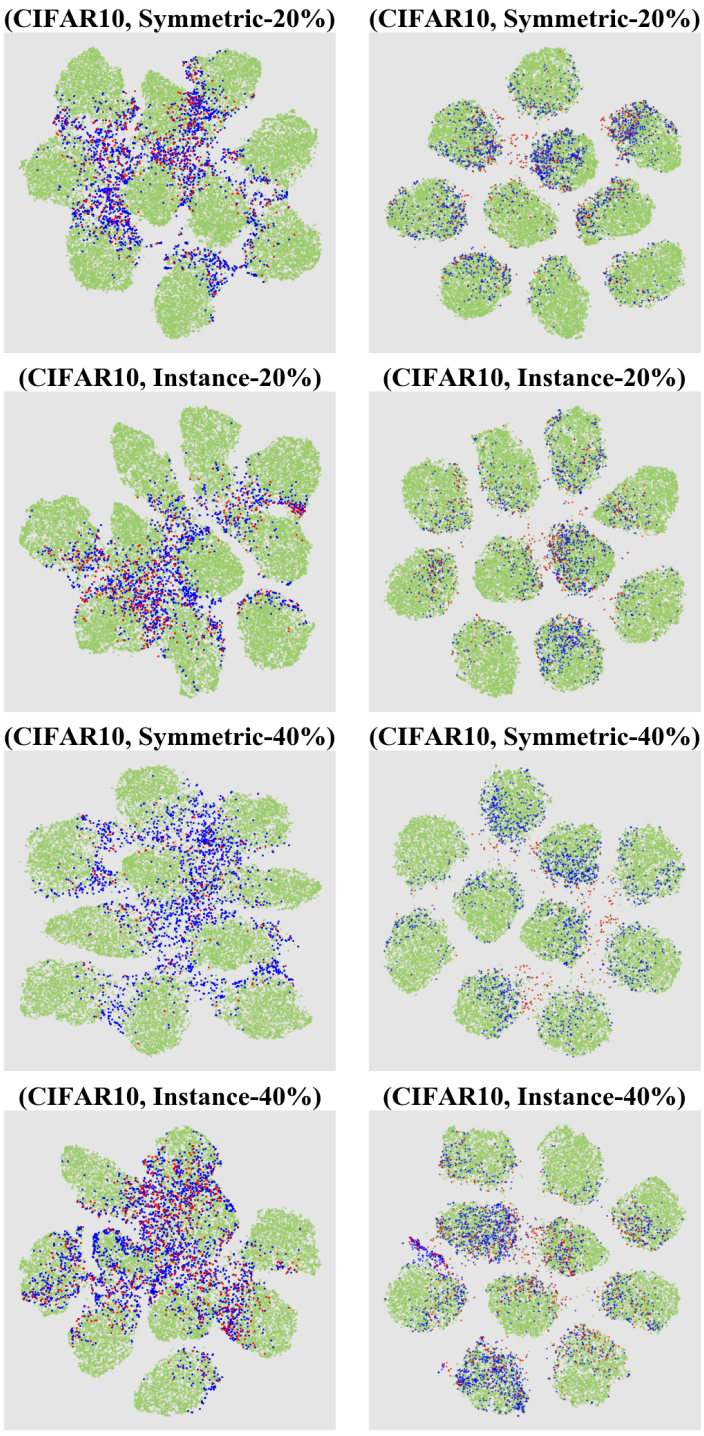


Figure A.5: Visualization of the extracted confident examples on *CIFAR10*. The first column is about the confident data extracted in the first run of the inner loop; while the second column is about the confident data extracted in the outer loop.

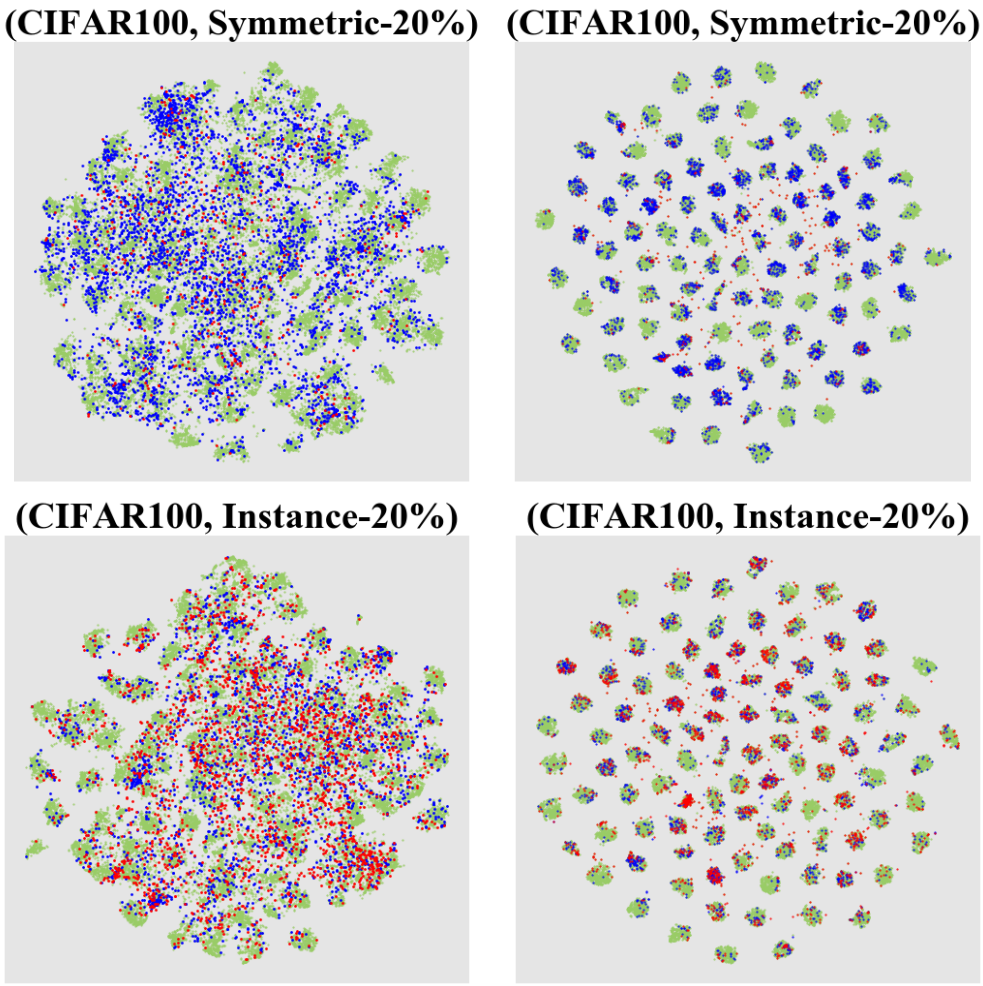


Figure A.6: Visualization of the extracted confident examples on *CIFAR100*. The first column is about the confident data extracted in the first run of the inner loop; while the second column is about the confident data extracted in the outer loop.

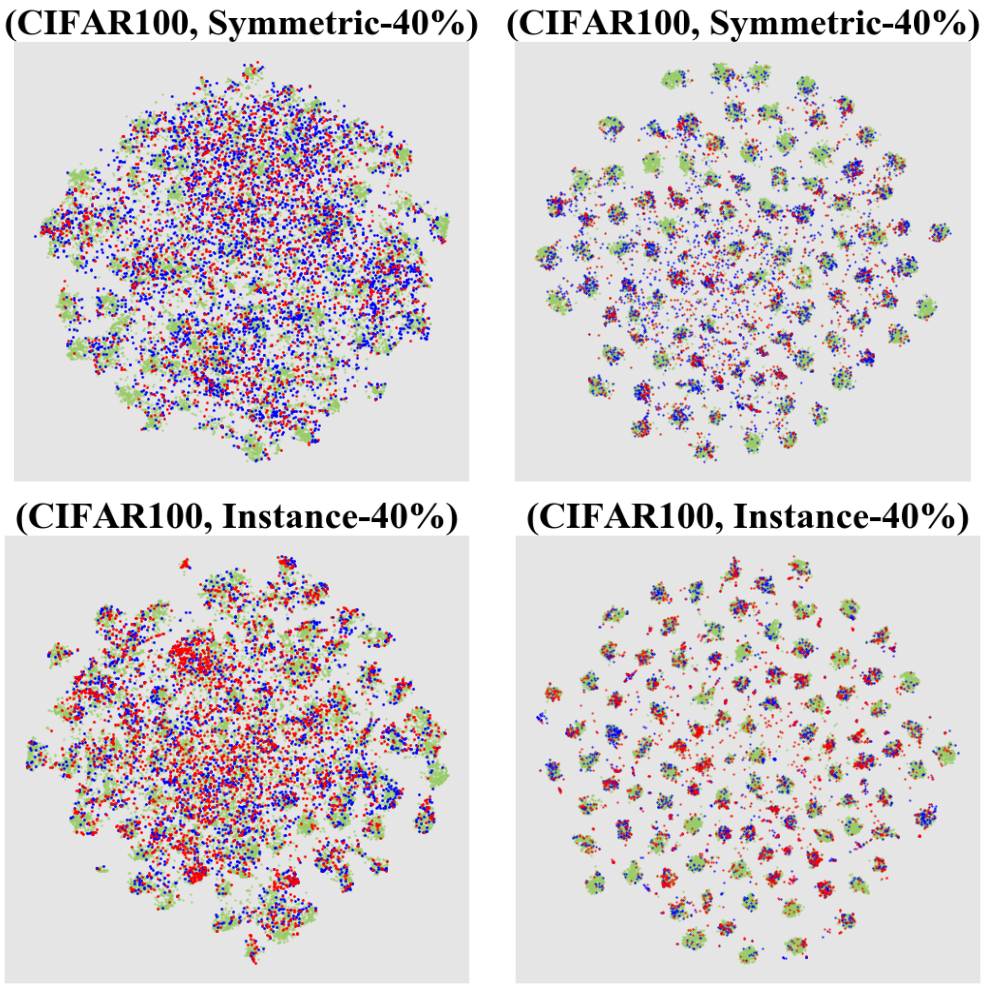


Figure A.7: Visualization of the extracted confident examples on *CIFAR100*. The first column is about the confident data extracted in the first run of the inner loop; while the second column is about the confident data extracted in the outer loop.

Appendix B

Supplementary for Chapter 4

B.1 Training details

In this section, we first provide details about the adopted three kinds of noisy labels. Then, we elaborate on the data preprocessing and the hyper-parameter settings in our experiments.

B.1.1 Definition of noise

According to different correlations between noisy labels and clean labels, there are three kinds of widely used label noise, namely symmetric class-dependent label noise, pairflip class-dependent label noise, and instance-dependent label noise [97, 42, 127]. In the following, we first introduce one basic concept: transition matrix [97], and then provide the details for all the three kinds of label noise, respectively.

Transition matrix: The *transition matrix* $T(\boldsymbol{x})$ is used to explicitly model the generation process of label noise, where $T_{ij}(\boldsymbol{x}) = \Pr(\bar{Y} = j|Y = i, X = \boldsymbol{x})$ is the flip rate between the true label and noisy label on given data $\boldsymbol{x}$. X is the variable of instances, Y is the variable of clean labels, and $\bar{Y}$ is the variable of noisy labels. $T_{ij}(\boldsymbol{x})$ is the ij-th entry of the transition matrix $T(\boldsymbol{x})$, which denotes the probability of the instance $\boldsymbol{x}$ with clean label i being observed with a noisy label j.

Symmetric label noise: Symmetric label noise is generated with symmetric class-dependent noise transition matrices. We set the flip rate α. Random flipping labels may change to true labels, so the flip rate

Algorithm 6: Instance-dependent Label Noise Generation

Input: Clean samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$; Noise rate τ.
Sample instance flip rates $q \in \mathbb{R}^n$ from the truncated normal distribution $\tau\mathcal{N}(\tau, 0.1^2, [0, 1])$;
Independently sample $w_1, w_2, \ldots, w_c$ from the standard normal distribution $\mathcal{N}(0, 1^2)$;
For $i = 1, 2, \ldots, n$ do
  $p = \boldsymbol{x}_i \times w_{y_i}$; // generate instance-dependent flip rates
  $p_{y_i} = -\infty$; // control the diagonal entry of the instance-dependent transition matrix
  $p = q_i \times softmax(p)$; // make the sum of the off-diagonal entries of the y_i-th row to be q_i
  $p_{y_i} = 1 - q_i$; // set the diagonal entry to be $1 - q_i$
  Randomly choose a label from the label space according to possibilities p as noisy label $\bar{y}_i$;
End for.
Output: Noisy samples $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^n$

may include or exclude true labels. For the flip rate excluding true labels, the diagonal entries of symmetric transition matrix are $1 - \alpha$ and the off-diagonal entries are $\alpha/(c-1)$. For the flip rate including true labels, the diagonal entries of symmetric transition matrix are $1 - (\alpha \times (c-1)/c)$ and the off-diagonal entries are α/c.

Pairflip class-dependent label noise: Pairflip noise is a simulation of fine-grained classification with noisy labels, where annotators may make mistakes only within very similar classes[138, 80]. The label noise is generated with pairflip class-dependent noise transition matrices, which is defined as follow. Let flip rate is α. The diagonal entries of a pairflip transition matrix are $1-\alpha$ and the entities for their adjacent classes, which the examples in a given class may be wrongly classified to, are α.

Instance-dependent label noise: Instance-dependent label noise is generated according to Algorithm 6. More details about this algorithm can be found in [127].

Table B.1: Training hyper-parameters for CIFAR-10/100 and Clothing1M

	CIFAR-10		CIFAR-100		Clothing1M
architecture	ResNet-18	PreAct ResNet-18	ResNet-34	PreAct ResNet-18	Resnet-50
loss function	CE	MixMatch loss	CE	MixMatch loss	CE
learning rate (lr)	0.1	0.02	0.1	0.02	5×10^{-3}
lr decay	100th & 150th	Cosine Annealing	100th & 150th	Cosine Annealing	20th & 30th
weight decay	10^{-4}	5×10^{-4}	10^{-4}	5×10^{-4}	10^{-3}
batch size	128	128	128	128	64
training examples	45,000	50,000	45,000	50,000	1,000,000
training epochs	200	300	200	300	50
PES lr	10^{-4}	10^{-4}	10^{-4}	10^{-4}	5×10^{-6}
T_1	25	20	30	35	20
T_2	7	5	7	5	7
T_3	5	-	5	-	-

B.1.2 Data preprocessing and experimental settings

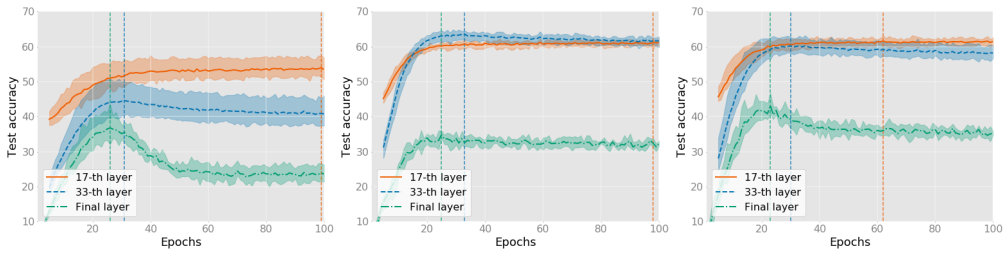
Data preprocessing: For experiments on CIFAR-10/100 [59] without semi-supervised learning, we use simple data augmentation techniques including random crop and horizontal flip. For experiments on CIFAR-10/100 with semi-supervised learning, except random cropping and horizontal flip, MixUp [141] is also employed, which is a critical component of MixMatch [11]. For Clothing1M [130], we first resize images to 256 $\times$ 256, and then random crop to 224 $\times$ 224, following a random horizontal flip.

Hyper-parameters of PES: We adopt an Adam optimizer for T_2 and T_3 for accelerating the model training and reducing the parameter turning, and T_2 and T_3 are chosen from $\{2, 5, 7\}$. Note that the number of total training epochs includes T_1, but excludes T_2 and T_3. To make PES work in large datasets, we regard training $100,000$ examples as an epoch in Clothing1M experiments.

Hyper-parameters of semi-supervised learning: We keep all the hyper-parameters fixed for different levels of noise, and only adjust λ_u for different noisy settings, since the ratio of confident examples (labeled data) and unconfident examples (unlabeled data) can vary greatly for different noisy settings. Specifically, we set $K = 2$, $T = 0.5$, and λ_u is chosen from $\{5, 15, 25, 50, 75, 100\}$. α begins with 4, and changes to 0.75 after 150th epoch. More details of hyper-parameters can be found in Table B.1 and Table B.2.

Table B.2: Semi-supervised loss weight λ_u for CIFAR-10/100

Datasets / Noise	Sym-20%	Sym-50%	Sym-80%	Pairflip-45%	Inst-20%	Inst-40%
CIFAR-10	5	15	25	5	5	15
CIFAR-100	50	75	100	50	50	50



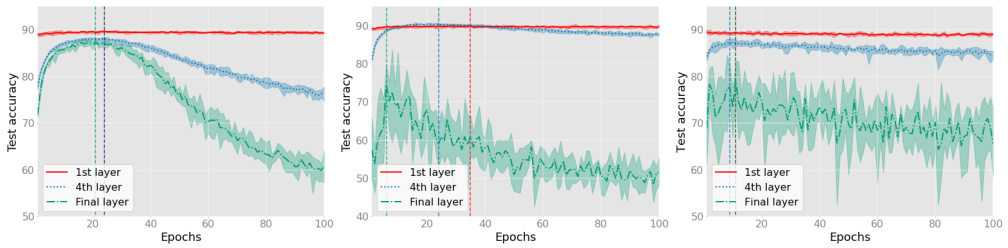
(a) Symmetric 50% **(b)** Pairflip 45% **(c)** Instance 40%

Figure B.1: We adopt ResNet-34 as the model on CIFAR-100 and evaluate the impact of noisy labels on the representations from the 17-th layer, the 33-th layer, and the final layer. The curves present the mean of five runs and the best performances highlight with dotted vertical lines.

B.2 Additional Experiments

In this section, we provide more experimental results on CIFAR-100 and Fashion-MNIST to further verify the hypothesize that noisy labels may have more severe impacts on the latter layers. We also provide additional comparisons with baselines, which exploit ensemble networks.

In the first experiment, we adopt a dataset with more classes: CIFAR-100 and a deeper network: ResNet-34 [45]. In addition, we adopt Fashion-MNIST [129], including 60,000 training images with 28x28 size and LeNet [27], which consists of two convolutional layers and three full-connected layers with ReLU activation. The learning procedure for CIFAR-100 and Fashion-MNIST is the same as that for CIFAR-10 in the chapter. Specifically, we first train the whole network on noisy data with different training epochs. For the final layer, we directly report the overall classification performance. For other selected layers, we frozen the parameters for the selected layer and previous layers, and then reinitialize and optimize the rest layers with clean data, and the final classification performance is adopted to evaluate the impact of noisy labels. We do not use image augmentation techniques for Fashion-MNIST dataset.



(a) Symmetric 50% **(b)** Pairflip 45% **(c)** Instance 40%

Figure B.2: We adopt LeNet as the model on Fashion-MNIST and evaluate the impact of noisy labels on the representations from the 1-st layer, the 4-th layer, and the final layer. The curves present the mean of five runs and the best performances highlight with dotted vertical lines. Note that vertical lines are merged together for the 1-st layer and 4-th layer on Symmetric 50%, and vertical lines of the 1-st layer and the final layer are merged together on Instance 40%.

Figure B.1 and Figure B.2 demonstrate the impacts of noisy labels on different layers on CIFAR-100 and Fashion-MNIST, respectively. From Figure B.1, we can see that the drop of the green line (the final layer) is the largest, the blue line (the 33-th layer) has a gradual decline, and the orange line (the 17-th layer) is relatively stable during the training process. These observations are similar to those for CIFAR-10. The performance of 17-th layer in ResNet-34 is affected by noisy labels later and less than that of the 9-th layer in ResNet-18. It is because there are more layers after the 17-th layer in ResNet-34 than the 9-th layer. Similar trends are observable in Figure B.2. The first layer is nearly unaffected by noisy labels, and the performance of the final layer has a larger decline compared with the 4-th layer. The learning speeds of different layers are unapparent in LeNet, since there are only three hidden layers in LeNet, and the gradient of losses transfers much easier compared with deeper networks. Another reason may be the simplicity of patterns in Fashion-MNIST without image augmentation techniques, which leads the convolutional layers to learn fast.

In the chapter, we compare our results with baselines evaluated with a single network. In this section, we compare our method with state-of-the-art methods with ensemble two networks taken from the original papers [65, 72]. We also adopt cross-entropy and MixUp [141] with a single network as baselines. From Table B.3, we can observe our results with a

Table B.3: Comparison with state-of-the-art methods using ensemble two networks and semi-supervised learning on CIFAR-10 and CIFAR-100 with symmetric label noise from different levels. Baseline results are taken from [65] and [72]. The highest results are reported for all the methods.

Dataset	CIFAR-10			CIFAR-100		
Methods / Noise	Sym-20%	Sym-50%	Sym-80%	Sym-20%	Sym-50%	Sym-80%
CE	87.2	80.9	65.8	58.1	47.5	23.6
MixUp	93.5	88.4	73.6	69.7	57.9	34.69
DivideMix*	**96.1**	94.6	93.2	77.3	74.6	60.2
ELR+	95.8	94.8	**93.3**	77.6	73.6	60.8
Ours (Semi)	**96.1**	**95.3**	**93.3**	**77.7**	**74.9**	**62.3**

single network are comparable to results of baselines with ensemble two networks. Specifically, on CIFAR-100, our method outperforms state-of-the-art methods across all settings.

Appendix C

Supplementary for Chapter 6

C.1 CIFAR20-SDN

C.1.1 Construction Details

Subclass-dominant label noise (SDN) is designed to mimic human labeling errors, which often emerge from distinct characteristics of examples. To identify such examples, we employ CIFAR-100, which provides two types of labels: class labels and sub-class labels. Samples belonging to the same subclass are considered to possess distinct characteristics. During the generation of the SDN dataset, we randomly flip a specified number of the last subclass labels within each class to the subsequent class. For instance, the labels of the "whale" subclass within the "aquatic mammals" class are probabilistically flipped to the "fish" class. Table C.1 illustrates the flipped sub-classes and the relationships between classes and sub-classes in CIFAR-100. For simplicity, we refer to this newly generated SDN dataset as CIFAR20-SDN.

C.1.2 Diverse Challenges across Different Classes

CIFAR20-SDN presents a multifaceted landscape with its array of subclasses, paving the way for a comprehensive study of SDN and a deeper insight into the intricacies of early stopping. In our experiments, we adjust the overall noise rate from 0% to 20%. This implies that within each specific subclass, the noise rate varies from 0% to 100%.

Table C.1: The table outlines the comprehensive structure of the classes within CIFAR20-SDN. Sub-classes marked in bold denote those from which examples are selectively flipped.

Class name	Abbreviation	Sub-classes name
aquatic mammals	aquatic	beaver, dolphin, otter, seal, **whale**
fish	fish	aquarium fish, flatfish, ray, shark,**trout**
flowers	flowers	orchids, poppies, roses, sunflowers, **tulips**
food containers	containers	bottles, bowls, cans, cups, **plates**
fruit and vegetables	fruit	apples, mushrooms, oranges, pears, **sweet peppers**
household electrical devices	electrical	clock, computer keyboard, lamp, telephone, **television**
household furniture	furniture	bed, chair, couch, table, **wardrobe**
insects	insects	bee, beetle, butterfly, caterpillar, **cockroach**
large carnivores	carnivores	bear, leopard, lion, tiger, **wolf**
large man-made outdoor things	man-made	bridge, castle, house, road, **skyscraper**
large natural outdoor scenes	natural	cloud, forest, mountain, plain, **sea**
large omnivores and herbivores	omnivores	camel, cattle, chimpanzee, elephant, **kangaroo**
medium-sized mammals	m-mammals	fox, porcupine, possum, raccoon, **skunk**
non-insect invertebrates	invertebrates	crab, lobster, snail, spider, **worm**
people	people	baby, boy, girl, man, **woman**
reptiles	reptiles	crocodile, dinosaur, lizard, snake, **turtle**
small mammals	s-mammals	hamster, mouse, rabbit, shrew, **squirrel**
trees	trees	maple, oak, palm, pine, **willow**
vehicles 1	vehicles 1	bicycle, bus, motorcycle, pickup truck, **train**
vehicles 2	vehicles 2	lawn-mower, rocket, streetcar, tank, **tractor**

As illustrated in Figure C.1, the effectiveness of early stopping in addressing SDN is notably inconsistent, heavily contingent on the specific subclass. Our analysis identifies two primary factors that significantly influence the success of early stopping. First, when the mislabeled subclass shares substantial semantic similarities with the clean class, distinguishing mislabeled examples from the clean class becomes challenging. For instance, mislabeling an instance from the "whale" subclass as a "fish" class is difficult to discern from other "fish" instances. Second, when classes have low diversity, the network tends to identify outliers, thus facilitating the detection of mislabeled examples, such as "trees" and "flowers". Conversely, distinguishing between correctly labeled and mislabeled examples becomes considerably more challenging in classes with higher diversity, such as "insects".

C.1.3 Clusters Effects

We train a ResNet-18 on CIFAR20-SDN for 200 epochs with an overall noise rate of 18% (SDN-18). Afterward, we utilized t-SNE to visualize all 45,000 training examples. By comparing Figure C.2 (a) and (b), we can

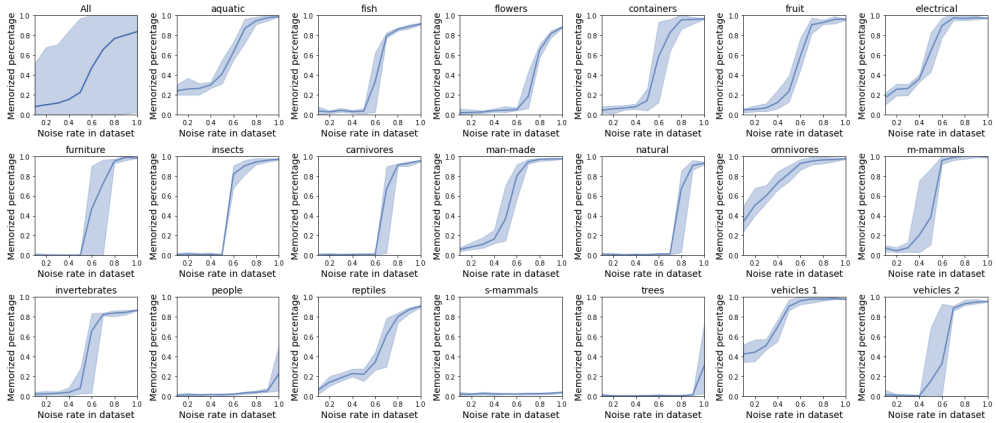


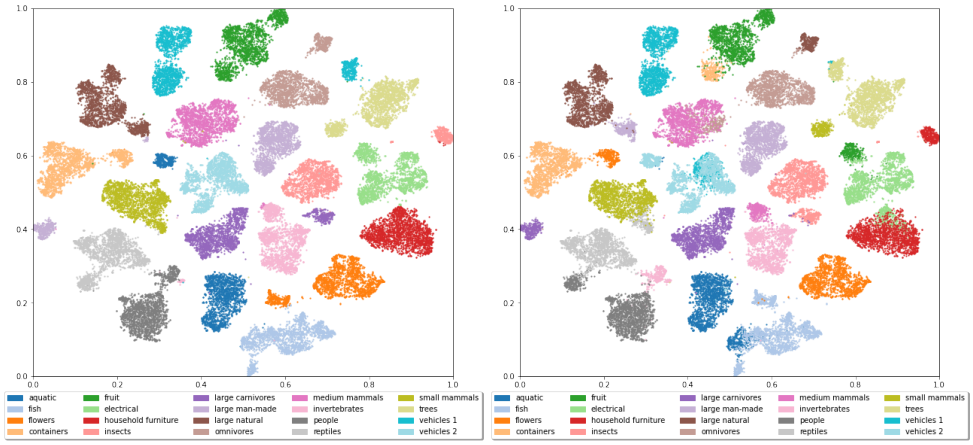
Figure C.1: The figure shows the effectiveness of early stopping across different classes and various noise rates. When the noise rate in the dataset is small, early stopping performs well in all classes. However, when it is over 50%, memorized mislabeled samples increases rapidly.

pinpoint the noise locations, specifically evident in the small clusters. Notably, the vast majority of the mislabeled examples tend to cluster together and usually maintain a distance from their noisy class. This pattern suggests that clustering algorithms, such as DBSCAN, can effectively identify them. However, when the semantics of mislabeled examples closely match those of the noisy class, these mislabeled examples' clusters may not be distinct from the main cluster of the noisy class, which is a limitation of our proposed method.

C.2 Experimental Settings

Data preprocessing. For all experiments, we employ simple data augmentation, including random crop and horizontal flip. For the Clothing-1M dataset, images are initially resized to a resolution of 256 x 256 pixels. Subsequently, a random crop is applied to produce images of 224 x 224 pixels, which are then followed by a random horizontal flip.

Hyper-parameters of semi-supervised learning. We follow [65, 7], and combine NoiseCluster with the semi-supervised learning method, MixMatch [11]. For experiments with SSL, we set $K = 2$, $T = 0.5$, and $\lambda_u = 0$. α begins at 4 and drops to 0.75 after half of the total number of epochs.



(a) CIFAR20-SDN with noisy labels **(b)** CIFAR20-SDN with clean labels

Figure C.2: The figure shows the distribution of training examples after 200 epochs on CIFAR20-SDN. We can find that the large clusters represent examples with correct labels, whereas the smaller clusters indicate those with incorrect labels.

Clothing1M. Due to the large size of the Clothing1M dataset [130], which consists of one million training examples, the computational time for t-SNE is significantly extended. This volume of data also results in changes to the hyper-parameters. To address these issues, we split the Clothing1M training data into two parts: a larger subset containing 950,000 examples and a smaller one with 50,000 examples. We initially employ the larger subset for training a model with early stopping. Subsequently, based on the trained model, we select and correct confident data from the smaller subset. This corrected data is then used for the remainder of the training process. Note that the model used in the Clothing1M experiments has been pre-trained with the ImageNet-1k dataset [60], leading to the generation of separable representations. As a result, late stopping is omitted in this case. When applying NoiseCluster to a model trained from scratch, it is advisable to employ late stopping.

(a) Noise rate 12% **(b)** Noise rate 20%

Figure C.3: Ablation analyses of stopping epoch. The mean and standard deviation (percentage) computed over five runs are presented.

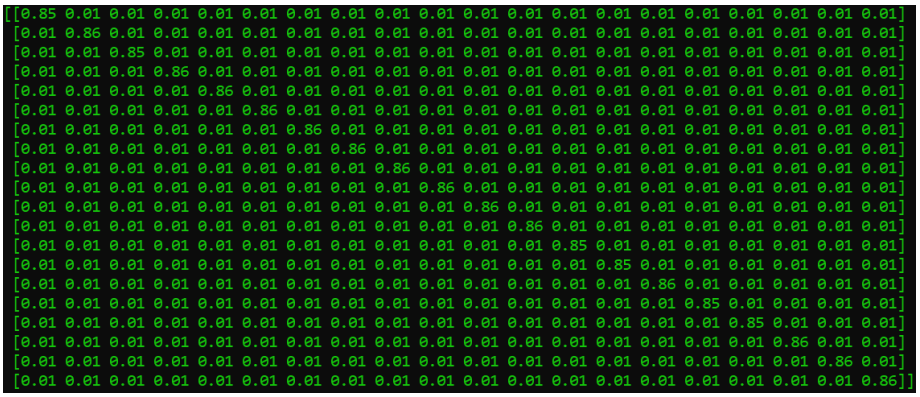
C.3 Additional Experiments

C.3.1 Ablation Study for Stopping Epoch

The effectiveness of NoiseCluster relies on long-trained representations derived from a model trained with later stopping, which stops the training process significantly later than what is common with early stopping. This raises two pertinent questions: How is the stopping epoch determined for later stopping? And does it display the same sensitivity as early stopping? To address these questions, we conduct a series of ablation experiments, adjusting the stopping epoch from 20th to 200th epochs.

As shown in Figure C.3, NoiseCluster does not perform well with representations obtained at the 20th epoch, indicating that the effectiveness of NoiseCluster hinges on long-trained representations. Additionally, we can observe that NoiseCluster maintains a stable performance between the 50th and 100th epochs. This span is considerably wider compared to early stopping, which typically require a precise stopping epoch or operate within a relatively narrow range [65, 72, 7]. To delve deeper into the impact of the stopping epoch, we expand the training period for later stopping, and stop the training up to the 200th epoch. Despite a slight drop in the performance of NoiseCluster, attributable to the degradation of representation quality, it remains superior to the best baseline, CDR[124]. These experiments suggest that the choice of the stopping epoch in NoiseCluster can

(a) $\hat{T} = -2$



(b) $\hat{T} = -4.5$

Figure C.4: (a, b) display the best estimated transition matrices during the training process with $\hat{T} = -2$ and $\hat{T} = -4.5$. The estimated noisy class-posterior probabilities are seen to be widespread across all classes, which is inconsistent with the generation of CIFAR20-SDN.

be made with considerable flexibility, displaying minimal sensitivity and negligible impact on the performance of NoiseCluster over a broad range.

C.3.2 Explore Noise-modeling-based Approaches

In this section, we further investigate into why noise-modeling-based approaches are ineffective for SDN. Methods based on noise modeling build a transition matrix between the clean distribution and noisy distribution [97, 128]. We utilize the state-of-the-art method, VolMinNet [68], on CIFAR20-SDN with a noise rate 18%. Following the original paper [68], we set $\hat{T}$ to -2 and -4.5, respectively.

As depicted in Figure C.4, it is evident that $\hat{T}$ holds the capability to

Table C.2: Training time comparison on CIFAR20-SDN with SDN-12. All methods run on four core CPU and a single Nvidia V100.

NoiseCluster	JointOptim	TopoFiler	CDR
1.7h	2.5h	1.5h	5.5h
NoiseCluster+	DivideMix	PES semi	F-DivideMix
2h	5.5h	3.1h	14.2h

modulate the extent of estimated noisy class-posterior probabilities within the transition matrix. For example, with $\hat{T} = -2$, VolMinNet predicts higher noisy class-posterior probabilities in the transition matrix compared to that with $\hat{T} = -4.5$. However, neither of these $\hat{T}$ settings successfully enable an accurate estimation of the transition matrix. Consequently, these inaccurately estimated transition matrices result in inferior performance on SDN.

C.3.3 Training Time Comparison

We evaluate the efficiency of our proposed method with state-of-the-art baselines. Table C.2 displays the total training time on CIFAR20-SDN. We can see that the training time of NoiseCluster is comparable to that of TopoFiler [121] and faster than JointOptim [108] and CDR[124]. Furthermore, NoiseCluster+ exhibits substantial efficiency compared with other baselines that employ SSL. For instance, it requires less than half the time needed by DivideMix [65].

C.4 Exploring SDN in Real-world Datasets

We presented only a limited number of SDN examples. In this subsection, we provide additional visual evidence to further demonstrate the presence of SDN in the Clothing1M and WebVision datasets.

C.4.1 SDN in Clothing1M

Figure C.5 presents a step-by-step exploration process for the Clothing1M dataset, demonstrating how features of mislabeled examples are clustered

Table C.3: Top-1 performance of our re-implemented DivideMix models alongside the pre-trained models, as assessed on the ImageNet and WebVision validation datasets. By employing an ensemble of pre-trained models, we generate pseudo clean labels to effectively identify mislabeled examples.

Methods	Classes	WebVision	ImageNet
DivideMix	50	77.34	75.2
DivideMix (reproduce)	50	77.04	74.24
Regnet_y_128gf (IMAGENET1K_SWAG_E2E_V1)	1000	86.64	91.12
ViT_H_14 (IMAGENET1K_SWAG_E2E_V1)	1000	86.08	91.2
Ensemble pretrained models	1000	86.96	91.6

together based on their semantics. Given the plethora of terms available for different clothing items, we can use accurate names to pinpoint these sub-classes. We highlight three sub-classes prominently impacted by label noise: "Sleep T-Shirt," "Leather Jacket," and "Down Vest," as shown in Figures C.6, C.7, and C.8.

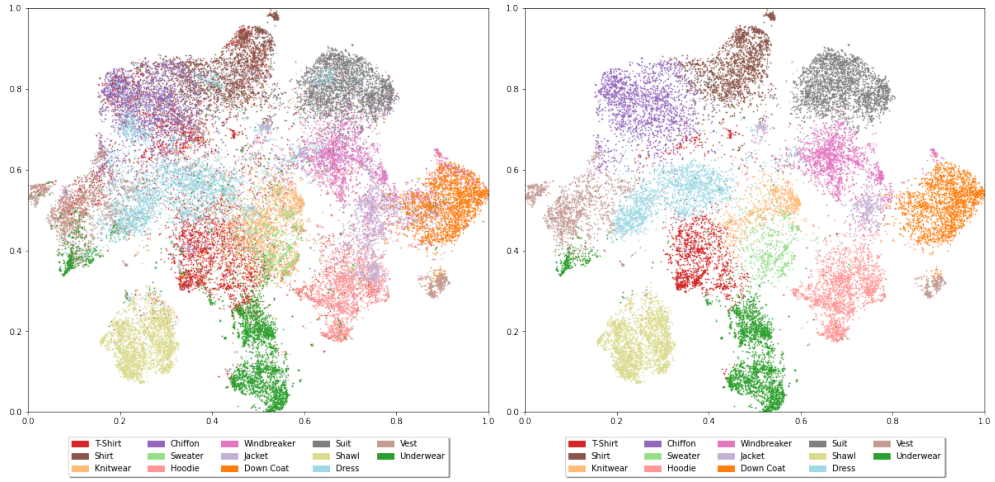
We can also observe that some mislabeled examples are not clustered together, but are instead evenly dispersed in some classes. It is important to understand that in the real world, clothing categories are not solely distinguished by visual features. For instance, the features of "Sweater" and "Knitwear" are intertwined, with the main distinction being the material, not the visual appearance. Similarly, the categories "Underwear" and "Vest" (correct name:"singlet" or "tank top") lead to confusion since some garments can serve multiple purposes.

Removing correctly labeled examples. Early stopping not only helps remove mislabeled examples but also inadvertently excludes those that are correctly labeled when SDN exists. A closer inspection of instances within the "Leather Jacket" cluster confirms the issue: out of the total examples, 313 are mislabeled, with only 23 correctly labeled. This finding indicates that more than 90% of the "Leather Jacket" examples are incorrectly labeled in the original dataset. Furthermore, the implementation of early stopping leads to the removal of the small fraction of correctly labeled examples from the confident set, as shown in Figure C.9, pushing the noise rate from over 90% to close to 100%.

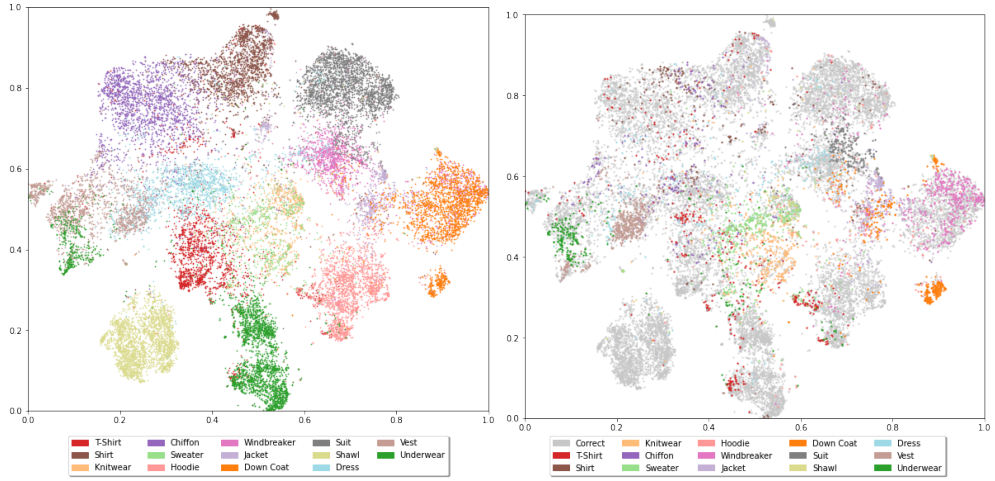
C.4.2 SDN in WebVision

To extend our evaluation of SDN prevalence, we engage in experiments using another real-world dataset: WebVision V1.0 [67]. Following the setting presented in [18], we utilize the first 50 classes of WebVision, known as "mini WebVision". To demonstrate the flexibility of integrating with existing early stopping methods, we employ DivideMix to select confident examples and extract long-trained representations from two networks, which have been trained for 80 epochs. The extracted features are presented in Figure C.10. We utilize an ensemble of pre-trained models to produce pseudo clean labels, and their performance is depicted in Table C.3.

In Figures C.11, C.12, and C.13, it is evident that, despite clear discrepancies between mislabeled and correctly labeled images, methods based on early stopping struggle to effectively identify these mislabeled examples. Moreover, these mislabeled samples do not fall into conventional categories such as "human" or "cars". Instead, they cluster according to different criteria, sharing similarities in attributes like shape and color with their correctly labeled counterparts. By considering out-of-domain examples as a single, unified class—termed "other", the mislabeled examples within each class can be interpreted as sub-classes of this "other" category.



(a) All examples, noisy labels **(b)** Confident examples, noisy labels



(c) Confident examples, clean labels **(d)** Mislabeled examples in confident set, clean labels

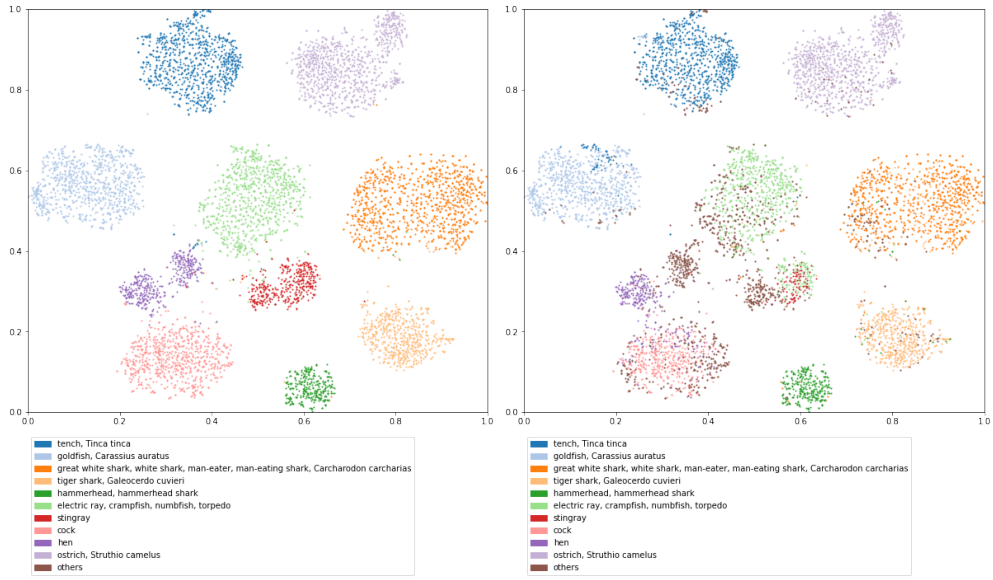
Figure C.5: (a) We visualize the features of all the examples in the evaluation set with noisy labels. (b) By removing unconfident examples, the features of confident examples are better separated. (c) We visualize the features of confident examples with clean labels. (d) We visualize the features of mislabeled examples in the confident set with clean labels. It is evident that the majority of mislabeled examples cluster together, confirming our observation of the clustering phenomenon in long-trained representations.

Figure C.6: Subclass: Sleep T-Shirt, positioned at coordinates (X:0.45, Y:0.1). The first row displays images from the "T-Shirt" class. The second row showcases those from the "Underwear" class, while the third and fourth rows present images mislabeled as "Underwear". For the mislabeled pictures, it's evident that these images depict a type of shirt meant for home use, Sleep T-Shirt. After comparing images in the test set and Chinese label names, we put the correct names in parentheses.

Figure C.7: Subclass: Leather Jacket, positioned at coordinates (X:0.75, Y:0.6). The first row images are from the "Jacket" class, and the second row presents images correctly labeled as the "Windbreaker" class (correct name:"trench coat"). Clearly, these mislabeled images belong to the "Leather Jacket" subclass within the "Jacket" category.

Figure C.8: Subclass: Down Vest, positioned at coordinates (X:0.9, Y:0.3). The first row displays images from the "Down coat" class, while the second row showcases those from the "Vest" class (correct name:"singlet" or "tank top"). When the "Vest" class is absent, these mislabeled images should be classified as "Down Vest" within the "Down coat" class.

Figure C.9: We discover 23 instances of "Leather Jacket" within the unconfident set. Despite being correctly labeled, these images are omitted from the confident set due to using early stopping. This process results in a purer cluster that predominantly contains mislabeled examples.



(a) Noisy labels **(b)** Pseudo clean labels

Figure C.10: Due to the absence of clean labels for the training data in WebVision, we employ two pre-trained models from PyTorch to generate pseudo-clean labels. While the pseudo-clean labels may not be accurate, they suffice to provide a general overview of the distribution of mislabeled examples. We then visualize confident features for the first ten classes using t-SNE, comparing noisy labels with the pseudo-clean ones. The category "other" represents examples where predictions from the pre-trained models are not in the first ten classes.

Figure C.11: Explore images categorized under the "Electric Ray" class. The first row showcases images sourced from the ImageNet validation set, while the second row features images from the WebVision validation set. Correctly labeled training images are in the third row, whereas the fourth and fifth rows illustrate instances of mislabeling in the training set. We can find majority of mislabeled images are in "black and white" pattern, which are close to some examples in the third row, so we call it "black and white" subclass.

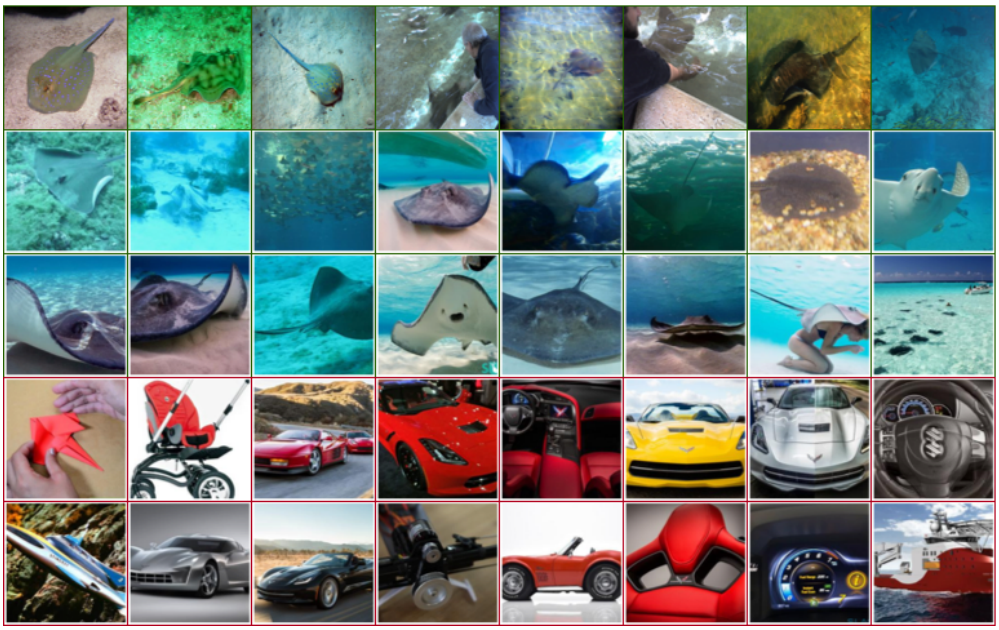


Figure C.12: The row structure remains consistent with the previous format. Notably, numerous cars are incorrectly classified in this category, alongside the presence of wheels and ships. The leading attribute contributing to this misclassification is likely the presence of "curves".

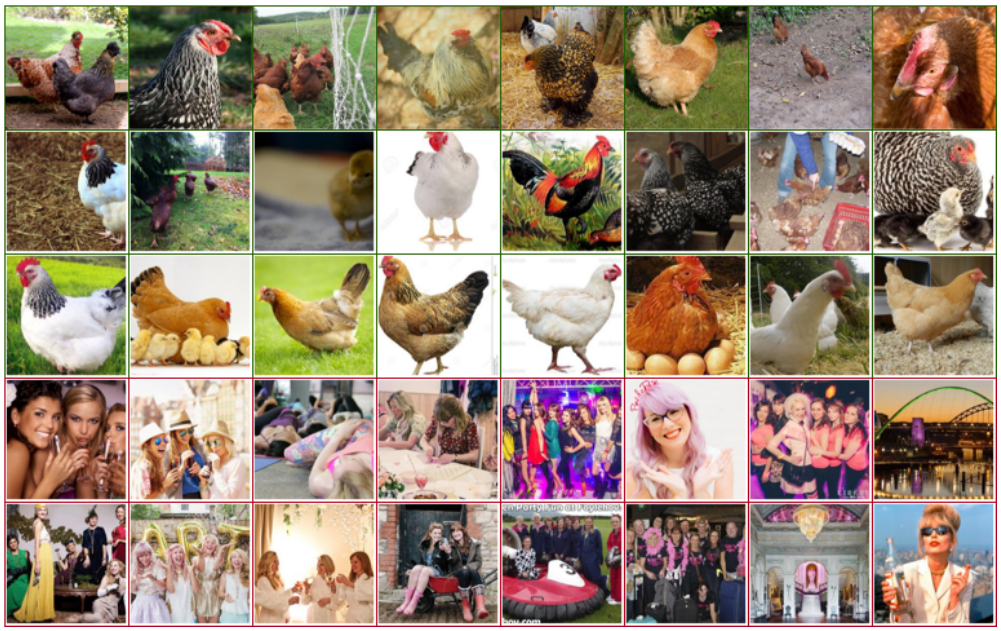


Figure C.13: The row structure remains consistent with the previous format. The mislabeled subclass may derive from color attributes, such as pink or white, which serve as the decisive classification features for the "hen" category.

Bibliography

[1] Z. Abai and N. Rajmalwar. "DenseNet Models for Tiny ImageNet Classification". In: *CoRR* abs/1904.10429 (2019).

[2] D. Angluin and P. D. Laird. "Learning From Noisy Examples". In: *Mach. Learn.* 2 (1987), pp. 343–370.

[3] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness. "Unsupervised Label Noise Modeling and Loss Correction". In: *ICML*. 2019, pp. 312–321.

[4] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-Julien. "A Closer Look at Memorization in Deep Networks". In: *ICML*. 2017, pp. 233–242.

[5] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-Julien. "A Closer Look at Memorization in Deep Networks". In: *ICML*. 2017, pp. 233–242.

[6] Y. Bai and T. Liu. "Me-Momentum: Extracting Hard Confident Examples from Noisily Labeled Data". In: *ICCV*. 2021.

[7] Y. Bai, E. Yang, B. Han, Y. Yang, J. Li, Y. Mao, G. Niu, and T. Liu. "Understanding and improving early stopping for learning with noisy labels". In: *NeurIPS*. 2021, pp. 24392–24403.

[8] Y. Bai, E. Yang, Z. Wang, Y. Du, B. Han, C. Deng, D. Wang, and T. Liu. "RSA: Reducing Semantic Shift from Aggressive Augmentations for Self-supervised Learning". In: *NeurIPS*. 2022, pp. 21128–21141.

[9] M. Belkin, D. Hsu, S. Ma, and S. Mandal. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854.

[10] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. "Curriculum learning". In: *ICML*. 2009, pp. 41–48.

[11] D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, and C. Raffel. "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *NeurIPS*. 2019, pp. 5050–5060.

[12] A. Berthon, B. Han, G. Niu, T. Liu, and M. Sugiyama. "Confidence Scores Make Instance-dependent Label-noise Learning Possible". In: *ICML*. Proceedings of Machine Learning Research. 2021, pp. 825–836.

[13] I. Biederman. "Recognition-by-components: a theory of human image understanding." In: *Psychological review* (1987), p. 115.

[14] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz. "Scene perception: Detecting and judging objects undergoing relational violations". In: *Cognitive psychology* (1982), pp. 143–177.

[15] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. "Language Models are Few-Shot Learners". In: *NeurIPS*. 2020, pp. 1877–1901.

[16] Q. Cai, Y. Wang, Y. Pan, T. Yao, and T. Mei. "Joint Contrastive Learning with Infinite Possibilities". In: *NeurIPS*. 2020.

[17] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". In: 2020, pp. 9912–9924.

[18] P. Chen, B. Liao, G. Chen, and S. Zhang. "Understanding and Utilizing Deep Neural Networks Trained with Noisy Labels". In: *ICML*. 2019, pp. 1062–1070.

[19] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *ICML*. 2020, pp. 1597–1607.

[20] X. Chen, H. Fan, R. B. Girshick, and K. He. "Improved Baselines with Momentum Contrastive Learning". In: *CoRR* abs/2003.04297 (2020).

[21] X. Chen and K. He. "Exploring Simple Siamese Representation Learning". In: *CVPR*. 2021, pp. 15750–15758.

[22] D. Cheng, Y. Ning, N. Wang, X. Gao, H. Yang, Y. Du, B. Han, and T. Liu. "Class-Dependent Label-Noise Learning with Cycle-Consistency Regularization". In: *NeurIPS*. 2022.

[23] H. Cheng, Z. Zhu, X. Li, Y. Gong, X. Sun, and Y. Liu. "Learning with Instance-Dependent Label Noise: A Sample Sieve Approach". In: *ICLR*. 2021.

[24] C.-Y. Chuang, J. Robinson, Y.-C. Lin, A. Torralba, and S. Jegelka. "Debiased contrastive learning". In: *NeurIPS* (2020), pp. 8765–8775.

[25] A. Coates, A. Y. Ng, and H. Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *AISTATS*. 2011, pp. 215–223.

[26] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. "RandAugment: Practical data augmentation with no separate search". In: *CoRR* abs/1909.13719 (2019).

[27] Y. L. Cun, L. D. Jackel, B. E. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. E. Hubbard. "Handwritten Digit Recognition: Applications of Neural Net Chips and Automatic Learning". In: *NATO*. 1989, pp. 303–318.

[28] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Math. Control. Signals Syst.* 2.4 (1989), pp. 303–314.

[29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *ICLR*. 2021.

[30] M. Dredze, K. Crammer, and F. Pereira. "Confidence-weighted linear classification". In: *ICML*. 2008, pp. 264–271.

[31] A. Ermolov, A. Siarohin, E. Sangineto, and N. Sebe. "Whitening for Self-Supervised Representation Learning". In: *ICML*. 2021, pp. 3015–3024.

[32] M. Ester, H. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on*

Knowledge Discovery and Data Mining (KDD-96). 1996, pp. 226–231.

[33] V. Feldman and C. Zhang. "What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation". In: *NeurIPS*. 2020, pp. 2881–2891.

[34] X. Gastaldi. "Shake-shake regularization". In: *arXiv preprint arXiv:1705.07485* (2017).

[35] A. Ghosh and A. S. Lan. "Contrastive Learning Improves Model Robustness Under Label Noise". In: *CVPR*. Computer Vision Foundation / IEEE, 2021, pp. 2703–2708.

[36] S. Gidaris, P. Singh, and N. Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *ICLR*. 2018.

[37] J. Goldberger and E. Ben-Reuven. "Training deep neural-networks using a noise adaptation layer". In: *ICLR*. 2017.

[38] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. The MIT Press, 2016.

[39] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour". In: *CoRR* abs/1706.02677 (2017).

[40] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning". In: *NeurIPS*. 2020, pp. 21271–21284.

[41] B. Han, Q. Yao, T. Liu, G. Niu, I. W. Tsang, J. T. Kwok, and M. Sugiyama. "A Survey of Label-noise Representation Learning: Past, Present and Future". In: *CoRR* abs/2011.04406 (2020).

[42] 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: *NeurIPS*. 2018, pp. 8527–8537.

[43] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *CVPR*. 2020, pp. 9726–9735.

[44] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. "Mask R-CNN". In: *ICCV*. 2017, pp. 2980–2988.

[45] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *CVPR*. 2016, pp. 770–778.

[46] K. He, X. Zhang, S. Ren, and J. Sun. "Identity Mappings in Deep Residual Networks". In: *ECCV*. 2016, pp. 630–645.

[47] W. He, B. Li, and D. Song. "Decision boundary analysis of adversarial examples". In: *ICLR*. 2018.

[48] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song. "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty". In: *NeurIPS*. 2019, pp. 15637–15648.

[49] K. Hornik, M. B. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[50] C. Huang, Y. Li, C. C. Loy, and X. Tang. "Learning Deep Representation for Imbalanced Classification". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 5375–5384.

[51] S.-J. Huang, R. Jin, and Z.-H. Zhou. "Active learning by querying informative and representative examples". In: *NeurIPS*. 2010, pp. 892–900.

[52] Z. Huang, X. Xia, L. Shen, B. Han, M. Gong, C. Gong, and T. Liu. "Harnessing out-of-distribution examples via augmenting content and style". In: *ICLR*. 2023.

[53] Z. Huang, M. Zhu, X. Xia, L. Shen, J. Yu, C. Gong, B. Han, B. Du, and T. Liu. "Robust Generalization against Photon-Limited Corruptions via Worst-Case Sharpness Minimization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16175–16185.

[54] J.-J. Hwang, S. X. Yu, J. Shi, M. D. Collins, T.-J. Yang, X. Zhang, and L.-C. Chen. "Segsort: Segmentation by discriminative sorting of segments". In: *ICCV*. 2019, pp. 7334–7344.

[55] T. Ishida, I. Yamane, T. Sakai, G. Niu, and M. Sugiyama. "Do We Need Zero Training Loss After Achieving Zero Training Error?" In: *ICML*. 2020, pp. 4604–4614.

[56] 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: *ICML*. 2018, pp. 2309–2318.

[57] T. Kim, J. Ko, S. Cho, J. Choi, and S. Yun. "FINE Samples for Learning with Noisy Labels". In: *NeurIPS*. 2021, pp. 24137–24149.

[58] S. A. Koohpayegani, A. Tejankar, and H. Pirsiavash. "Mean Shift for Self-Supervised Learning". In: *ICCV*. 2021, pp. 10306–10315.

[59] A. Krizhevsky, G. Hinton, et al. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *NeurIPS*. 2012, pp. 1097–1105.

[61] M. P. Kumar, B. Packer, and D. Koller. "Self-paced learning for latent variable models". In: *NeurIPS*. 2010, pp. 1189–1197.

[62] Y. LeCun, C. Cortes, and C. J. Burges. "The MNIST database of handwritten digits". In: *http://yann.lecun.com/exdb/mnist/* (1998).

[63] J. Li, M. Zhang, K. Xu, J. Dickerson, and J. Ba. "How does a Neural Network's Architecture Impact its Robustness to Noisy Labels?" In: *NeurIPS*. 2021, pp. 9788–9803.

[64] J. Li, M. Zhang, K. Xu, J. P. Dickerson, and J. Ba. "Noisy Labels Can Induce Good Representations". In: *arXiv preprint arXiv:2012.12896* (2020).

[65] J. Li, R. Socher, and S. C. H. Hoi. "DivideMix: Learning with Noisy Labels as Semi-supervised Learning". In: *ICLR*. 2020.

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

[67] W. Li, L. Wang, W. Li, E. Agustsson, and L. V. Gool. "WebVision Database: Visual Learning and Understanding from Web Data". In: *CoRR* abs/1708.02862 (2017).

[68] X. Li, T. Liu, B. Han, G. Niu, and M. Sugiyama. "Provably End-to-end Label-noise Learning without Anchor Points". In: *ICML*. 2021, pp. 6403–6413.

[69] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. "Feature Pyramid Networks for Object Detection". In: *CVPR*, pp. 936–944.

[70] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common Objects in Context". In: *ECCV*. 2014, pp. 740–755.

[71] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common Objects in Context". In: *ECCV*. 2014, pp. 740–755.

[72] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda. "Early-Learning Regularization Prevents Memorization of Noisy Labels". In: *NeurIPS*. 2020, pp. 20331–20342.

[73] T. Liu and D. Tao. "Classification with noisy labels by importance reweighting". In: *IEEE Transactions on pattern analysis and machine intelligence* 38.3 (2016), pp. 447–461.

[74] T. Liu and D. Tao. "Classification with noisy labels by importance reweighting". In: *IEEE Transactions on pattern analysis and machine intelligence* 38.3 (2016), pp. 447–461.

[75] Y. Liu. "Understanding Instance-Level Label Noise: Disparate Impacts and Treatments". In: *ICML*. 2021, pp. 6725–6735.

[76] Y. Liu. "Understanding Instance-Level Label Noise: Disparate Impacts and Treatments". In: *ICML*. 2021, pp. 6725–6735.

[77] P. M. Long and R. A. Servedio. "Random classification noise defeats all convex potential boosters". In: *ICML*. 2008, pp. 608–615.

[78] I. Loshchilov and F. Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *ICLR*. 2017.

[79] Y. Lu, Y. Bo, and W. He. "Noise Attention Learning: Enhancing Noise Robustness by Gradient Scaling". In: *NeurIPS*. 2022.

[80] Y. Lyu and I. W. Tsang. "Curriculum Loss: Robust Learning and Generalization against Label Corruption". In: *ICLR*. 2020.

[81] X. Ma, H. Huang, Y. Wang, S. Romano, S. M. Erfani, and J. Bailey. "Normalized Loss Functions for Deep Learning with Noisy Labels". In: *ICML*. 2020, pp. 6543–6553.

[82] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S. Xia, S. N. R. Wijewickrema, and J. Bailey. "Dimensionality-Driven Learning with Noisy Labels". In: *ICML*. 2018, pp. 3361–3370.

[83] L. v. d. Maaten and G. Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[84] L. v. d. Maaten and G. Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.11 (2008), pp. 2579–2605.

[85] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. "Mixed Precision Training". In: *ICLR*. 2018.

[86] I. Misra and L. van der Maaten. "Self-Supervised Learning of Pretext-Invariant Representations". In: *CVPR*. 2020, pp. 6706–6716.

[87] S. Mo, H. Kang, K. Sohn, C.-L. Li, and J. Shin. "Object-aware Contrastive Learning for Debiased Scene Representation". In: *Advances in Neural Information Processing Systems*. 2021, pp. 12251–12264.

[88] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2018.

[89] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. "Deep Double Descent: Where Bigger Models and More Data Hurt". In: *ICLR*. 2020.

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

[91] D. T. Nguyen, C. K. Mummadi, T. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox. "SELF: Learning to Filter Noisy Labels with Self-Ensembling". In: *ICLR*. 2020.

[92] M. Noroozi and P. Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: *ECCV*. 2016, pp. 69–84.

[93] C. G. Northcutt, T. Wu, and I. L. Chuang. "Learning with confident examples: Rank pruning for robust classification with noisy labels". In: *UAI*. 2017.

[94] P. O. O Pinheiro, A. Almahairi, R. Benmalek, F. Golemo, and A. C. Courville. "Unsupervised learning of dense visual representations". In: *NeurIPS* (2020), pp. 4489–4500.

[95] A. van den Oord, Y. Li, and O. Vinyals. "Representation Learning with Contrastive Predictive Coding". In: *CoRR* abs/1807.03748 (2018).

[96] 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: *CVPR*. 2017, pp. 1944–1952.

[97] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu. "Making Deep Neural Networks Robust to Label Noise: A Loss Correction Approach". In: *CVPR*. 2017, pp. 2233–2241.

[98] X. Peng, K. Wang, Z. Zhu, and Y. You. "Crafting Better Contrastive Views for Siamese Representation Learning". In: *CVPR*. 2022.

[99] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. "Learning transferable visual models from natural language supervision". In: *ICML*. 2021, pp. 8748–8763.

[100] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. "Improving language understanding by generative pre-training". In: (2018).

[101] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[102] M. Ren, W. Zeng, B. Yang, and R. Urtasun. "Learning to Reweight Examples for Robust Deep Learning". In: *ICML*. 2018, pp. 4331–4340.

[103] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. "Laion-5b: An open large-scale dataset for training next generation image-text models". In: (2022), pp. 25278–25294.

[104] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki. "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs". In: *arXiv preprint arXiv:2111.02114* (2021).

[105] R. R. Selvaraju, K. Desai, J. Johnson, and N. Naik. "Casting your model: Learning to localize improves self-supervised representations". In: *CVPR*. 2021, pp. 11058–11067.

[106] H. Song, M. Kim, D. Park, and J.-G. Lee. "Prestopping: How Does Early Stopping Help Generalization against Label Noise?" In: *arXiv preprint arXiv:1911.08059* (2019).

[107] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. "On the importance of initialization and momentum in deep learning". In: *ICML*. 2013, pp. 1139–1147.

[108] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. "Joint optimization framework for learning with noisy labels". In: *CVPR*. 2018, pp. 5552–5560.

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

[110] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C. Chang, I. Krivokon, W. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. H. Chi, and Q. Le. "LaMDA: Language Models for Dialog Applications". In: *CoRR* abs/2201.08239 (2022).

[111] S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, and J. Mohd-Yusof. "Combating Label Noise in Deep Learning using Abstention". In: *ICML*. 2019, pp. 6234–6243.

[112] Y. Tian, D. Krishnan, and P. Isola. "Contrastive Multiview Coding". In: *ECCV*. 2020, pp. 776–794.

[113] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. "What Makes for Good Views for Contrastive Learning?" In: *NeurIPS 2020*. 2020, pp. 6827–6839.

[114] A. M. Treisman and G. Gelade. "A feature-integration theory of attention". In: *Cognitive psychology* (1980), pp. 97–136.

[115] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, and L. V. Gool. "Revisiting Contrastive Methods for Unsupervised Learning of Visual Representations". In: (2021).

[116] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[117] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey. "Symmetric Cross Entropy for Robust Learning With Noisy Labels". In: *ICCV*. 2019, pp. 322–330.

[118] Y. Wang, A. Kucukelbir, and D. M. Blei. "Robust probabilistic modeling with bayesian data reweighting". In: *ICML*. JMLR. org. 2017, pp. 3646–3655.

[119] Y. Wang, J. Lin, J. Zou, Y. Pan, T. Yao, and T. Mei. "Improving Self-supervised Learning with Automated Unsupervised Outlier Arbitration". In: (2021).

[120] Z. Wang, Q. Li, G. Zhang, P. Wan, W. Zheng, N. Wang, M. Gong, and T. Liu. "Exploring set similarity for dense self-supervised representation learning". In: *CVPR*. 2022.

[121] P. Wu, S. Zheng, M. Goswami, D. N. Metaxas, and C. Chen. "A Topological Filter for Learning with Label Noise". In: *NeurIPS*. 2020, pp. 21382–21393.

[122] S. Wu, X. Xia, T. Liu, B. Han, M. Gong, N. Wang, H. Liu, and G. Niu. "Class2Simi: A Noise Reduction Perspective on Learning with Noisy Labels". In: *ICML*. 2021, pp. 11285–11295.

[123] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. "Unsupervised Feature Learning via Non-Parametric Instance Discrimination". In: *CVPR*. 2018, pp. 3733–3742.

[124] X. Xia, T. Liu, B. Han, C. Gong, N. Wang, Z. Ge, and Y. Chang. "Robust early-learning: Hindering the memorization of noisy labels". In: *ICLR*. 2021.

[125] X. Xia, T. Liu, B. Han, M. Gong, J. Yu, G. Niu, and M. Sugiyama. "Instance Correction for Learning with Open-set Noisy Labels". In: *arXiv preprint arXiv:2106.00455* (2021).

[126] X. Xia, T. Liu, B. Han, M. Gong, J. Yu, G. Niu, and M. Sugiyama. "Sample Selection with Uncertainty of Losses for Learning with Noisy Labels". In: *arXiv preprint arXiv:2106.00445* (2021).

[127] X. Xia, T. Liu, B. Han, N. Wang, M. Gong, H. Liu, G. Niu, D. Tao, and M. Sugiyama. "Part-dependent label noise: Towards instance-dependent label noise". In: *NeurIPS*. 2020, pp. 7597–7610.

[128] X. Xia, T. Liu, N. Wang, B. Han, C. Gong, G. Niu, and M. Sugiyama. "Are Anchor Points Really Indispensable in Label-Noise Learning?" In: *NeurIPS*. 2019, pp. 6835–6846.

[129] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *arXiv preprint arXiv:1708.07747* (2017).

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

[131] Y. Xu, P. Cao, Y. Kong, and Y. Wang. "L_DMI: A Novel Information-theoretic Loss Function for Training Deep Nets Robust to Label Noise". In: *NeurIPS*. 2019, pp. 6222–6233.

[132] Y. Xue, K. Whitecross, and B. Mirzasoleiman. "Investigating Why Contrastive Learning Benefits Robustness against Label Noise". In: *ICML*. Vol. 162. 2022, pp. 24851–24871.

[133] S. Yang, E. Yang, B. Han, Y. Liu, M. Xu, G. Niu, and T. Liu. "Estimating Instance-dependent Bayes-label Transition Matrix using a Deep Neural Network". In: *ICML*. 2022, pp. 25302–25312.

[134] Y. Yao, T. Liu, M. Gong, B. Han, G. Niu, and K. Zhang. "Instance-dependent label-noise learning under a structural causal model". In: *NeurIPS*. 2021, pp. 4409–4420.

[135] Y. Yao, T. Liu, B. Han, M. Gong, J. Deng, G. Niu, and M. Sugiyama. "Dual T: Reducing Estimation Error for Transition Matrix in Label-noise Learning". In: *NeurIPS*. 2020, pp. 7260–7271.

[136] Y. You, I. Gitman, and B. Ginsburg. "Scaling SGD Batch Size to 32K for ImageNet Training". In: *CoRR* abs/1708.03888 (2017).

[137] X. Yu, B. Han, J. Yao, G. Niu, I. W. Tsang, and M. Sugiyama. "How Does Disagreement Benefit Co-teaching?" In: *ICML*. 2019.

[138] X. Yu, B. Han, J. Yao, G. Niu, I. W. Tsang, and M. Sugiyama. "How does Disagreement Help Generalization against Label Corruption?" In: *ICML*. 2019, pp. 7164–7173.

[139] X. Yu, T. Liu, M. Gong, and D. Tao. "Learning with Biased Complementary Labels". In: *ECCV*. 2018, pp. 69–85.

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

[141] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *ICLR*. 2018.

[142] R. Zhang, P. Isola, and A. A. Efros. "Colorful Image Colorization". In: *ECCV*. 2016, pp. 649–666.

[143] X. Zhang and M. Maire. "Self-supervised visual representation learning from hierarchical grouping". In: *NeurIPS*. 2020, pp. 16579–16590.

[144] E. Zheltonozhskii, C. Baskin, A. Mendelson, A. M. Bronstein, and O. Litany. "Contrast to Divide: Self-Supervised Pre-Training for Learning with Noisy Labels". In: *WACV*. 2022, pp. 387–397.

[145] M. Zheng, S. You, F. Wang, C. Qian, C. Zhang, X. Wang, and C. Xu. "ReSSL: Relational Self-Supervised Learning with Weak Augmentation". In: (2021).

[146] Z. Zhu, T. Liu, and Y. Liu. "A Second-Order Approach to Learning With Instance-Dependent Label Noise". In: *CVPR*. 2021, pp. 10113–10123.

¬