Rethinking Safe Semi-supervised Learning: Transferring the Open-set Problem to A Close-set One

Qiankun Ma*1, Jiyao Gao*1, Bo Zhan1, Yunpeng Guo1, Jiliu Zhou1 and Yan Wang†1

1School of Computer Science, Sichuan University, Chengdu, China

Abstract

Conventional semi-supervised learning (SSL) lies in the close-set assumption that the labeled and unlabeled sets contain data with the same seen classes, called in-distribution (ID) data. In contrast, safe SSL investigates a more challenging open-set problem where unlabeled set may involve some out-of-distribution (OOD) data with unseen classes, which could harm the performance of SSL. When we are experimenting with the mainstream safe SSL methods, we have a surprising finding that all OOD data show a clear tendency to gather in the feature space. This inspires us to solve the safe SSL problem from a fresh perspective. Specifically, for a classification task with K seen classes, we utilize a prototype network not only to generate K prototypes of all seen classes, but also explicitly model an additional prototype for the OOD data, transferring the K-way classification on the open-set to the (K+1)-way on the close-set. In this way, the typical SSL techniques (e.g., consistency regularization and pseudo labeling) can be applied to tackle the safe SSL problem without additional consideration of OOD data processing like other safe SSL methods do. Particularly, considering the possible low-confidence pseudo labels, we further propose an iterative negative learning (INL) paradigm to enforce the network learning knowledge from complementary labels on wider classes, improving the network’s classification performance. Extensive experiments on four benchmark datasets show that our approach remarkably lifts the performance on safe SSL and outperforms the state-of-the-art methods.

1. Introduction

In the past decade, the development of deep learning brings prosperity to the field of computer vision, such as image classification, attributed to the growing scale of labeled data [18, 9, 23]. However, it is time-consuming and expensive to collect large amounts of labeled data. Numerous methods resort to semi-supervised learning (SSL) to relieve this restriction [33, 19, 32]. By exploring the massive valuable information from the abundant easy-to-acquire unlabeled data, SSL methods can effectively narrow the performance gap towards the fully-supervised models.

Although the SSL methods are proven to be effective in improving the classification performance in case of sparse labels, most of them follow a close-set assumption by default, that is, the labeled data and the unlabeled data share the same label space. See Figure 1(a) for example, both labeled data and unlabeled data are built on the classes of cat and dog, which are also called in-distribution (ID) data. Yet, in many real scenarios, such a simple and crude assumption often breaks down since the unlabeled data could be collected in the wild [40]. In this case, there will be many samples with unknown classes (or unseen classes) appearing in the unlabeled dataset, which we call out-of-distribution (OOD) data, resulting in an open-set problem. As shown in Figure 1, in the open-set, besides the cat and dog classes in the labeled set (or seen classes), there are also some unseen classes, e.g., butterfly, hen, rabbit. Without reliable labels for these unseen classes, the OOD data may lead to a sig-

*Equal contribution.
†Corresponding author: Yan Wang (wangyanscu@hotmail.com).
significant degradation of model performance [8, 4, 25], and even pose a significant safety risk to the accurate prediction since the model is forced to predict the unseen class as a seen one. This severely hinders the SSL methods from being deployed in real-world applications.

To handle this problem, a series of enhanced SSL methods have been emerged to improve the classification performance and guarantee the prediction safety in a more realistic scenario, namely safe semi-supervised learning (safe SSL) methods [8, 4, 10, 11, 41]. The safe here means that the model using extra unlabeled data will not be worse than a simple supervised one. Most current safe SSL methods adhere a two-step scheme: 1) identifying the OOD data, and 2) dealing with the OOD data. In stage one, these methods tend to identify the OOD data by classifying the OOD data and the ID data into two separate broad classes [8, 4, 10, 41]. After the OOD data is detected, some methods treat them as hazards and just discard them in the second stage [4, 41]. In contrast, He et al. [10] argued that the valuable information contained in the OOD data can be used to further improve the identification of OOD data. To this end, they endeavored to calibrate the seen-class probability distribution into a uniform distribution, thus suppressing the over-confidence problem to unseen class and eliminating the risk of hard OOD data being recognized as ID data.

Despite the effectiveness of the two-step pipeline, utilizing it in reality might still be cumbersome and requires consideration of some additional caveats. For instance, in stage one, clustering all ID data into one class is a coarse-grained binary classification task, which may conflict with the final fine-grained multi-class classification task in terms of feature space learning [14]. Furthermore, in stage two, it is not easy to come up with a feasible way to utilize OOD data to enhance model performance, either. When we are experimenting with these safe SSL methods, an interesting phenomenon strikes us as a surprise. We find that most safe SSL methods [8, 10, 14] show an intrinsic ability to gather the OOD features in the first stage like Figure 1(b), especially for [14] employing the self-supervised learning manner. This involves us to ask the following question: Why the OOD data which belong to different classes are gathered in the feature space?

![Figure 2](image)

**Figure 2.** CAM visualization of the results. For the ID data (Bird, Cat, Dog), the features are more class-discriminative.

This inspires us to solve the safe SSL problem from a new viewpoint, that is we can treat all the OOD data as an individual class peer to other seen classes during training. In this way, we can transfer a $K$-way classification task on the open-set to a simple $(K+1)$-way classification task on the close-set. Then we can tackle the complex safe SSL task in the same way as the trivial SSL task using all the methods that are applicable on the close-set assumption, greatly streamlining the safe SSL problem.

Following the above inspiration, in this paper, we propose a prototype-based safe SSL framework for image classification from a fresh perspective. Specifically, our framework includes two stages, including the first stage of unseen-class prototype generation and the second stage of semi-supervised image classification. Assuming that there are $K$ seen classes, the first stage starts by training a prototype network [31] on the labeled ID data so that it can produce $K$ prototypes of seen classes. Based on the prototypes, we can distinguish the feature distribution of all the OOD data according to the distance of the features from the prototypes, thereby explicitly modeling an additional prototype of all unseen classes. These $K+1$ prototypes allow the framework to make predictions for both the $K$ fine-grained seen classes and the remaining unseen class simultaneously, avoiding the conflict with coarse-grained classification. By casting the $K$-way classification to the $(K+1)$-way classification, we can directly use any common and effective SSL technique to carry out the semi-supervised classification task in the second stage.

A widely accepted solution for SSL is to filter high confidence pseudo labels for unlabeled data. Yet, removing all the unreliable predictions could lead to insufficient and categorically imbalanced training [37]. To make full use of unlabeled data, negative learning (NL) [37, 38, 16] is proposed to provide complementary labels which indicate the class to which the current sample is least likely to belong, namely negative class. However, the conventional NL may become less powerful when solving a classification task with a large number of classes. The reasons can be attributed to the following aspects: 1) Current NL methods don’t ensure the complementary label of the same sample keep unchanged during the training process. Wavering labels are unfavor-
able for stable training; 2) When there are excessive classes, current NL methods can only determine the complementary labels on one negative class at a time, which can solely supplement limited knowledge to the network and leave massive information on other classes unexplored. To alleviate these limitations, we propose an iterative negative learning (INL) paradigm in the second stage. Unlike the previous NL methods [16, 37], we harness a memory bank to preserve the complementary labels of unlabeled data at each training iteration. As training goes, the complementary labels will explore new negative classes iteratively based on the saved historical complementary labels. By this, the INL paradigm enables to fully unearth the knowledge of all unlabeled data in the whole feature space, helping the model classify the samples more confidently. We summarize our contributions as follows:

- We rethink the safe SSL problem from a fresh perspective and propose a prototype-based safe SSL framework to explicitly model the OOD data as a novel class peer to ID classes. In this manner, we transform the safe SSL from an open-set problem to a close-set one.
- We raise an INL paradigm to enhance the feature learning capability of our framework with regard to the unreliable predictions in SSL. By employing a memory bank to progressively update the complementary label until it covers most negative classes, our model can excavate the knowledge of the unlabeled data in the whole feature space, producing more confident classification results.
- We evaluate our framework on extensive benchmark datasets and the experimental results show that our method remarkably outperforms the existing state-of-the-art safe SSL methods.

2. Related Work

2.1. Semi-supervised Learning

Semi-supervised learning (SSL) has been a long-term research topic, and recently it has made a splash in computer vision (CV) attributed to the advancement of deep learning. The core idea of SSL is to make full use of extensive unlabeled data with as little labeled data as possible to achieve an approximate performance of fully supervised learning. There are a vast number of SSL methods proposed to resolve various CV tasks, which can be briefly categorized into pseudo-labeling, consistency regularization, and hybrid methods. The pseudo-labeling methods [3, 21, 15, 27] aim to produce pseudo labels for unlabeled data, and then supplement them into the labeled set to strengthen the model training. The consistency regularization methods [33, 29, 19, 24, 36, 35, 30, 13] assume that the decision boundary is supposed to cross the low-density region of samples, based on which they encourage the outputs of the unlabeled data and their perturbations to keep consistent in order to learn a discriminative decision boundary. Hybrid [1, 2, 32, 22] methods combine the two methods mentioned above and use some data augmentation approaches [5, 6, 39, 43] to enhance the SSL methods. However, these methods are based on the assumption that all labeled and unlabeled data share the same label space, which is often broken in real-world problems.

2.2. Safe Semi-Supervised Learning

In the open world, the unlabeled set is prone to contain some out-of-distribution (OOD) samples that do not belong to any seen class in the labeled dataset. It has been demonstrated that these OOD data will harm the in-distribution (ID) data classification on all seen classes [25]. Various solutions have been proposed to solve such a safety problem, called safe SSL methods [8, 4, 10, 11, 3, 41, 14]. A widely adopted way of safe SSL is to train a model on labeled data and use it to detect and remove the OOD data. Guo et al. [8] designed a weight function to subdue the weight of OOD data. Chen et al. [4] proposed a score function to filter out the OOD data based on their ensembled prediction results. However, by discarding the OOD data, these methods also abandon the valuable knowledge contained in them[10]. In contrast, Huang et al. [14] proposed a self-supervised warm-up training method to fully utilize the OOD data to further promote the OOD detection. Furthermore, He et al. [10] utilized the OOD data by calibrating their category distributions into uniform ones. Nevertheless, the application of OOD data is still dependent on the accuracy of the previous OOD data identification, which may be compromised by the limited labeled data. Different from the above methods, our work explicitly models all OOD data as a novel class peer to seen-classes, thus transferring the safe SSL from an open-set problem to close-set one. This brave attempt unifies the classification of OOD data and ID data into one feature space, enabling the full use of all OOD data and ID data and eliminating the possible negative impact of low performance when identifying OOD data separately.

2.3. Negative Learning

For SSL methods, incorrect supervisory information can cause catastrophic effects on network performance. The negative learning (NL) methods are therefore proposed to reduce the risk of incorrect information from the noisy labels in the labeled set [16], or the erroneously assigned pseudo labels in the unlabeled set [37, 38]. Generally, the routine NL is conducted by selecting a class randomly from all classes except for the given one as the complementary label per iteration, then optimizing the output probability corresponding to the complementary label to approach 0 [16]. Nevertheless, the existing NL methods only mine the useful
information on one complementary label, which has little impact on performance when the number of classes is large. Moreover, the complementary label could be variable as the training goes, bring much uncertainty to the model training. Our work addresses these problems by extending the vanilla NL to an iterative version, called INL. Our INL paradigm employs a memory bank to progressively update the complementary label until it covers most negative classes and remains one as the positive one. By employing the INL paradigm, our model can produce more confident classification results.

3. Methodology
3.1. Overview
Problem Formulation. Like the standard SSL problems, we assume that the training set of safe SSL contains two subsets: a labeled set $D^l = \{(x_i^l, y_i^l)\}_{i=1}^N$ with $N$ samples and an unlabeled dataset $D^u = \{(x_i^u)\}_{i=1}^M$ with $M$ samples, and $N \ll M$. Here, $x_i^l$ or $x_i^u$ represents the input image, $y_i^l \in \{1, 2, \cdots K\}$ denotes its corresponding label, and $K$ is the number of seen classes. Notably, the safe SSL setting indicates that in the unlabeled set, besides $x_i^u$ belonging to the $K$ seen classes, or called ID data, there may be $x_i^u$ that goes beyond the seen classes, called OOD data. The OOD data lead to a class distribution mismatch with ID data, which will further harm the performance of standard SSL methods. Our framework is proposed to ensure the safety and performance of the target classification task.

Framework Overview. As Figure 3 shows, our framework mainly contains two stages. The first stage aims to pre-train a prototype network on both labeled and unlabeled data, enabling the network to generate accurate prototypes of $K$ seen classes (Section 3.2). Then, we employ a distance-based function to distinguish the OOD data from the ID data, and model the prototype of all OOD data, thus turning the $K$-way classification to the $(K+1)$-way classification (Section 3.3). By regarding the prototype of OOD data as new class peer to seen classes, we solve the safe SSL problem in a simpler close-set way in the second stage by utilizing the common SSL techniques (pseudo labeling and consistency regularization) (Section 3.4). To cope with the erroneously assigned pseudo labels, we propose an iterative negative learning (INL) paradigm to relieve the incorrect guidance and improve the classification performance (Section 3.5).

3.2. Prototype Network Pre-training
In our framework, we adopt a prototype network [31] for the classification task by generating a set of prototypes $C = c_1, \cdots, c_K$ as the anchors of seen classes. A good prototype network contributes to generating accurate seen class prototypes, or ID prototypes, which can lay solid foundations for the following OOD data identification. To achieve this, we pre-train the prototype network on both $D^l$ and $D^u$ to improve its ability of feature representation.

Learning from Labeled Data. In each training iteration, we can first acquire the ID prototype of class $k (k = 1, \ldots, K)$ based on the labeled set $D^l$:

$$ c_k = \frac{1}{|D_k|} \sum_{(x_i, y_i) \in D_k} f_\theta(x_i^l), $$

where $D_k$ denotes the set of samples belonging to class $k$, $f_\theta$ is the prototype network parameterized by $\theta$. The prototype $c_k$ is actually the average of embedding features $f_\theta(x_i^l)$ belonging to class $k$. With these prototypes, we can obtain a $K$-dimensional vector that measures the similarity between $f_\theta(x_i^l)$ and the prototype set $C$:

$$ d_i = (f_\theta(x_i^l) \cdot c_1, \cdots, f_\theta(x_i^l) \cdot c_K)^T, $$

where “$\cdot$” denotes the inner product. Finally, the probability distribution $p_i^l$ of $x_i^l$ regarding the $K$ seen classes can be derived by applying a softmax function to the similarity vector $d_i$. To train the model, we use a standard cross-entropy loss on $p_i^l$ and its corresponding label $y_i^l$ as follows:

$$ \mathcal{L}_{ce} = -\frac{1}{b^l} \sum_{i=1}^{b^l} \ln (p_i^l \cdot [y_i^l]), $$

where $b^l$ denotes the number of labeled data in a mini-batch, and $p_i^l [k]$ represents the $k$-th element in $p_i^l$.

Learning from Unlabeled Data. To fully leverage the unlabeled data to enhance the representation capacity of the prototype network, we introduce an auxiliary self-supervised rotation prediction task with all unlabeled data. Specifically, given an unlabeled sample $x_i^u$, we rotate it by $0^\circ$, $90^\circ$, $180^\circ$, $270^\circ$, respectively, denoted as $x_{i,j}^u$, $j \in \{1, 2, 3, 4\}$. Through the prototype network $f_\theta$, we can obtain the embedding feature $f_\theta(x_{i,j}^u)$. Then the embedding feature is further sent to a 4-way rotation classification head $h_\varphi$ parameterized by $\varphi$, which consists of a linear layer and a softmax layer, to produce the prediction $q_{i,j}^u = h_\varphi(f_\theta(x_{i,j}^u))$. Accordingly, we additionally add a rotation prediction loss in the pre-training stage:

$$ \mathcal{L}_{rot} = -\frac{1}{4b^u} \sum_{i=1}^{b^u} \sum_{j=1}^{4} \ln (q_{i,j}^u [j]), $$

where $b^u$ denotes the number of unlabeled data in a mini-batch. We use the $\mathcal{L}_{ce}$ and $\mathcal{L}_{rot}$ to optimize our model in stage 1.

Prototype Update. At the first training iteration, all the prototypes are initialized randomly. At the end of each iteration, we update all the ID prototypes based on the
Stage I: Prototype Network Pre-training & OOD Prototype Generation

Figure 3. The overview of our framework, consisting of two stages. Stage I aims to pre-train the prototype network with labeled and unlabeled data by a cross-entropy loss $L_{ce}$ and a rotation prediction loss $L_{rot}$, respectively, then filter out the OOD data and model their prototype. Stage II shows the semi-supervised classification task on unlabeled data in a close-set way. The training losses include a consistency loss $L_{c}$, a positive learning loss $L_{p}$, and a negative learning loss $L_{n}$. The loss on labeled data is omitted for simplicity.

3.3. OOD Prototype Generation

After the pre-training, we have obtained a powerful prototype network and an ID prototype set $C$ with regard to seen classes. Based on this, we can distinguish all the OOD data from the ID data in the unlabeled set. The core idea is to use a distance-based function to filter out those data far away from all the ID prototypes. Specifically, for each unlabeled sample $x^u_i$, we calculate its distance from all ID prototypes as follows:

$$d^u_i = (\|f_\theta(x^u_i) - c_1\|_2, \ldots, \|f_\theta(x^u_i) - c_K\|_2)^T.$$ (5)

Then we classify the $x^u_i$ as ID or OOD data based on the minimal value in its distance vector $d^u_i$:

$$x^u_i \in \begin{cases} ID \text{ data}, & \min(d^u_i) \leq \lambda \\ OOD \text{ data}, & \min(d^u_i) > \lambda \end{cases},$$ (6)

where $\lambda$ is a threshold for OOD data filtering, which is determined by the OTSU algorithm [26]. For all the filtered OOD data, we can calculate their OOD prototype $c_o$ as Equation 1 does. Thus, we now have $K$ ID prototypes and a peer OOD prototype. The prototype set $C$ can be updated by adding $c_o$, namely $C = C \cup \{c_o\}$, where $|C| = K + 1$. In this way, we successfully turn the $K$-way classification task on the open-set to the $(K+1)$-way classification task on the close-set. Correspondingly, for any input $x_i$, we can attain its probability distribution $p_i$ over seen and unseen classes by computing the similarity between its embedding feature and the ID prototypes along with the OOD prototype, which just needs a small adjustment to Equation 2 by adding a term of $f_\theta(x_i) \cdot c_o$.

3.4. Close-set Semi-supervised Classification

Since we have $K$ ID prototypes and an OOD prototype, we can perform $(K+1)$-way classification like solving a standard SSL problem. For the labeled data, we also employ the entropy loss of Equation 3 to optimize our framework. For the unlabeled data, inspired by FixMatch [32], we combine consistency regularization and pseudo-labeling to make full use of them. Particularly, for an input unlabeled sample $x^u_i$, we apply weak and strong data augmentations to get its two augmented versions $x^{w^u}_i$ and $x^{s^u}_i$. Through the
prototype network and the softmax function, we can obtain two corresponding probability distributions \( p_i^w \) and \( p_i^s \).  

**Consistency Regularization.** The consistency regularization approach assumes that the predictions of a sample should remain consistent after different perturbations. Therefore, we impose a consistency loss on \( p_i^w \) and \( p_i^s \) to encourage them to be similar via KL divergence, which can be defined as follows:

\[
L_c = \frac{1}{b} \sum_{i=1}^{b^n} KL(p_i^w, p_i^s).
\]  

(7)

**Pseudo-Labeling.** Meanwhile, we regard the probability \( p_i^w \) from weak augmentation as pseudo label \( \hat{y}_i^w \) for \( x_i^u \). Considering the possible wrongly assigned pseudo labels, we propose to split the unlabeled data into high-confidence (HC) and low-confidence (LC) regions based on the maximum value in its pseudo label. For the HC data, we send them to positive learning, where pseudo labels are used to compute a positive loss with \( p_i^w \):

\[
L_p = -\frac{1}{b} \sum_{i=1}^{b^n} \mathbb{1} (p_i^w [m] > \tau_1) \ln (p_i^w [\hat{y}_i^w]),
\]  

(8)

where \( b \) denotes the number of HC data in a mini-batch, \( m \) is the index of the largest element in \( \hat{y}_i^w \), \( \tau_1 \) is a threshold to divide HF and LF data, and \( \mathbb{1} (\cdot) \) is an indication function (equaling 1 if the condition in the bracket is satisfied, otherwise 0).

As for the LF data with \( p_i^w [m] \leq \tau_1 \), the network is prone to assign wrong pseudo labels. To mitigate the accumulation of incorrect information provided by the erroneous pseudo labels during training, we resort to negative learning (NL) [16] and propose an iterative negative learning (INL) paradigm, which will be introduced in the next subsection in detail.

### 3.5. Iterative Negative Learning

Conventional NL [16, 37] believes that although a network cannot confidently tell which class an LF data belongs to, it can learn knowledge from the class the data does not belong to, or called negative class. Nevertheless, conventional NL only selects one negative class as the complementary label, which only provides limited information for learning. In addition, conventional NL tends to produce waiving complementary labels, hindering the stable training of the network. To relieve these problems, we improve NL to an iterative version, called INL. Different from NL, INL maintains a memory bank \( M \) with the size of \( M \times (K + 1) \) to save the historical complementary labels \( \{ \hat{y}_i^c \} \) for all the LF data after each iteration. Based on the historical complementary label, we can iteratively update it to cover more negative classes in the next iteration. With all the more comprehensive complementary labels, our framework can be enhanced to classify more confidently.

**Iterative Complementary Labels.** The complementary label \( \hat{y}_i^c \) in INL is updated in an iterative way. Specifically, in an iteration, given the probability distribution \( p_i^w \) from the sample \( x_i^u \), we pick the lowest probability score and update the \( \hat{y}_i^c \) in \( M_i \) (\( i = 1, \ldots, M \)) as follows:

\[
\hat{y}_i^c [l] = M_i [l] = \begin{cases} 1, & p_i^w [l] \leq \tau_2 \\ 0, & p_i^w [l] > \tau_2 \end{cases},
\]  

(9)

where \( l = \text{argmin} (p_i^w) \) denotes the index of class with the lowest probability score, and \( \tau_2 \) is the threshold to decide whether the class in \( \hat{y}_i^c \) could be regarded as a negative class confidently. Once the complementary label \( \hat{y}_i^c \) is updated, we save it in \( M_i \) and use it to optimize our framework via a negative learning loss. Then, the probability distribution \( p_i^w \) can be further calibrated in the next iteration. Accordingly, the complementary label \( \hat{y}_i^c \) is iteratively updated by including a new negative class. Please note that, during the iterative process, if the highest probability score is over \( \tau_1 \), the \( p_i^w \) will be utilized for positive learning as Equation 8 does. If the number of all negative classes reaches \( K \), i.e., only one class remains uncertain, the iterative process will stop. We summarize this iterative process in Algorithm 1.

Finally, the completely updated complementary label \( \hat{y}_i^c \) serves for the negative learning loss in the following network training.

**Negative Learning.** Considering that the complementary label indicates the negative classes that a sample should not belong to, we encourage the prediction probability scores on these negative classes to approach 0. Consequently, we can perform the negative learning by the following negative learning loss:

\[
L_n = -\frac{1}{b} \sum_{i=1}^{b^n} \sum_{j=1}^{K+1} \hat{y}_i^c [j] \ln (1 - p_i^w [j]),
\]  

(10)

where \( b^n \) denotes the number of LF data in a mini-batch.

In summary, our framework is trained in stage two by a total loss as below:

\[
L_2 = L_{ce} + L_n + \alpha (L_p + L_c),
\]  

(11)

where \( \alpha \) is a coefficient to balance these terms.

### 4. Experiments

#### 4.1. Datasets

To validate the effectiveness of our proposed framework, we carry out experiments on the same public datasets as [10] for semi-supervised image classification, including MNIST [20], CIFAR-10 [17], CIFAR-100 [17] and Tiny-ImageNet [7]. To achieve the safe SSL setting, we select various ratios of unlabeled data from seen-classes, namely...
Algorithm 1: Iterative Generation of Complementary Label

**Input:** The probability distribution $p_i^w \in R^{K+1}$ of sample $x_i^w$; the corresponding memory bank $\mathcal{M}_t \in R^{K+1}$.

**Parameters:** The number of negative classes $n^-$; the thresholds $\tau_1$ and $\tau_2$.

**Initialize:** $n^- = 0; \tau_1 = 0.95; \tau_2 = 0.05$; a complementary label $\hat{y}_i^c \in R^{K+1}$ initialized by 0.

while $n^- < K$ do
  Obtain the index $l = \text{argmin} (p_i^w)$ and $\mathcal{M}_t[l] = 0$;
  if $\max (p_i^w) > \tau_1$ then
    compute the positive learning loss $L_p$ based on Equation 8;
    break;
  end
  else if $p_i^w[l] \leq \tau_2$ then
    update $\hat{y}_i^c[l] = 1$ and save it in $\mathcal{M}_t[l]$;
    $n^-++$;
  end
  compute the negative learning loss $L_n$ based on Equation 10;
end

**Output:** The updated complementary label $\hat{y}_i^c$.

mismatch ratio. For instance, when the mismatch ratio is 0.5, it means that half of the unlabeled data is from seen classes and the remaining is from unseen classes. The detailed description of datasets can be found in the Supplementary Material.

### 4.2. Implementation details

When training with MNIST, we employ a two-layer CNN as the backbone network using stochastic gradient descent (SGD) with a learning rate of $1e^{-3}$ [8]. The network is trained for 500 epochs with a batch size of 100. As for CIFAR-10, CIFAR-100, and TinyImageNet, the WideResNet28-2 [42] is used as the backbone network. We use SGD to train the network with an initial learning rate of 0.03 which is adjusted via the cosine decay strategy, and a momentum of 0.9. The network in both stages is trained for 1024 epochs with a batch size of 64. The hyperparameter $\alpha$ in Equation 11 is dynamically updated as in [14]. In addition, we also investigate the influence of two thresholds $\tau_1$, $\tau_2$ in the Supplementary Material.

### 4.3. Comparison Methods

We compare our method with following state-of-the-art (SOTA) SSL methods: Pi-Model [29], Pseudo-Labeling (PL) [21], Temporal Ensembling [19], Mean Teacher [33], Virtual Adversarial Training (VAT) [24], FixMatch [32], Deep Safe Semi-Supervised Learning (DS3L) [8], Uncertainty Aware Self-Distillation (UASD) [4], Multi-Task Curriculum (MTC) [41], Curriculum Labeling (CL) [3], Safe Parameter Learning (SPL) [11], OpenMatch [28], Trash to Treasure (T2T) [14], and SAFE-STUDENT [10]. All the methods are compared on the testing set containing only ID samples. Note that, DS3L, UASD, MTC, CL, SPL, OpenMatch, T2T and SAFE-STUDENT are specially tailored for solving the safe SSL problem.

### 4.4. Experimental Results

#### 4.4.1 Comparison with SOTA Methods

Quantitative Analysis. Table 1 reports the results more comprehensively on all datasets with 0.3 or 0.6 mismatch ratio. Besides the comparison on MNIST and CIFAR-10, particularly, we list the quantitative comparison results on two larger datasets CIFAR-100 and TinyImageNet in the third and fourth columns. As can be seen, when the mismatch ratio is 0.3, our method achieves 72.5% accuracy in average on CIFAR-100, about 4.1% higher than SAFE-STUDENT and 2.7% higher than T2T which performs best in all other methods. Similarly, for the harder TinyImageNet dataset, our method can still outperform other safe SSL methods by at least 2.0% average accuracy even with a high-level ratio of 0.6. All these comparison results demonstrate the effectiveness and robustness of our method in resolving the safe SSL problem under open-set assumption. We also provide qualitative analysis in the Supplementary Material.

#### 4.4.2 Ablation Study

We validate the effectiveness of the proposed loss functions, i.e., $L_c$, $L_p$, $L_n$, by ablating them and measuring the performance on CIFAR-100 with mismatch ratio of 0.3. The results are reported in Table 3. By comparing the first and second rows, we can know the $L_c$ loss brings about 5.3% improvement. When integrating the $L_p$ loss in the third row or the $L_n$ loss in the fourth row, the performance gets further improved by 1.6% and 1.4%, respectively. With all the losses applied, our method can obtain the best accuracy of 72.5%. Through the ablation study, we can conclude that each proposed loss has a positive effect on our method.

#### 4.4.3 Identification of Unseen-Class

Since the unseen-class of OOD samples plays an important role in our method, we further evaluate the unseen-class
Table 1. Seen-class classification accuracy (%) of different methods on the four datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST ratio=0.3</th>
<th>MNIST ratio=0.6</th>
<th>CIFAR-10 ratio=0.3</th>
<th>CIFAR-10 ratio=0.6</th>
<th>CIFAR-100 ratio=0.3</th>
<th>CIFAR-100 ratio=0.6</th>
<th>TinyImagenet ratio=0.3</th>
<th>TinyImagenet ratio=0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pi-Model [29]</td>
<td>92.4±0.6</td>
<td>86.6±0.5</td>
<td>75.7±0.7</td>
<td>74.5±1.0</td>
<td>59.4±0.3</td>
<td>57.9±0.3</td>
<td>36.9±0.4</td>
<td>36.4±0.5</td>
</tr>
<tr>
<td>PL [21]</td>
<td>90.0±0.7</td>
<td>86.0±0.6</td>
<td>75.8±0.8</td>
<td>74.6±0.7</td>
<td>60.2±0.3</td>
<td>57.5±0.6</td>
<td>36.6±0.6</td>
<td>35.8±0.4</td>
</tr>
<tr>
<td>VAT [24]</td>
<td>94.5±0.3</td>
<td>90.4±0.3</td>
<td>76.9±0.6</td>
<td>75.0±0.5</td>
<td>61.8±0.4</td>
<td>59.6±0.6</td>
<td>36.7±0.5</td>
<td>36.3±0.6</td>
</tr>
<tr>
<td>FixMatch [32]</td>
<td>-</td>
<td>-</td>
<td>81.5±0.2</td>
<td>80.9±0.3</td>
<td>65.9±0.3</td>
<td>65.2±0.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DS3L [8]</td>
<td>96.8±0.3</td>
<td>94.5±0.4</td>
<td>78.1±0.4</td>
<td>76.9±0.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UASD [4]</td>
<td>96.2±0.6</td>
<td>94.3±0.8</td>
<td>77.6±0.4</td>
<td>76.0±0.4</td>
<td>61.8±0.4</td>
<td>58.4±0.5</td>
<td>37.1±0.7</td>
<td>36.9±0.6</td>
</tr>
<tr>
<td>MTC [41]</td>
<td>93.7±0.5</td>
<td>88.5±0.3</td>
<td>85.5±0.6</td>
<td>81.7±0.5</td>
<td>63.1±0.6</td>
<td>61.1±0.3</td>
<td>37.0±0.5</td>
<td>36.6±0.4</td>
</tr>
<tr>
<td>CL [3]</td>
<td>96.9±0.1</td>
<td>95.6±0.4</td>
<td>83.2±0.4</td>
<td>82.1±0.4</td>
<td>63.6±0.4</td>
<td>61.5±0.5</td>
<td>37.3±0.7</td>
<td>36.7±0.8</td>
</tr>
<tr>
<td>CL+SPL [10]</td>
<td>-</td>
<td>-</td>
<td>87.8±0.3</td>
<td>84.1±0.5</td>
<td>65.9±0.3</td>
<td>65.5±0.4</td>
<td>38.6±0.5</td>
<td>37.7±0.5</td>
</tr>
<tr>
<td>OpenMatch [28]</td>
<td>97.8±0.2</td>
<td>96.0±0.2</td>
<td>88.2±0.2</td>
<td>85.5±0.3</td>
<td>68.7±0.1</td>
<td>68.4±0.2</td>
<td>37.9±0.4</td>
<td>37.0±0.3</td>
</tr>
<tr>
<td>SAFE-STUDENT [10]</td>
<td>98.3±0.3</td>
<td>96.5±0.1</td>
<td>85.7±0.3</td>
<td>83.8±0.1</td>
<td>68.4±0.2</td>
<td>68.2±0.1</td>
<td>37.7±0.3</td>
<td>37.1±0.3</td>
</tr>
<tr>
<td>T2T [14]</td>
<td>98.4±0.1</td>
<td>96.2±0.2</td>
<td>89.0±0.4</td>
<td>86.9±0.2</td>
<td>69.8±0.2</td>
<td>68.0±0.2</td>
<td>39.1±0.3</td>
<td>37.3±0.3</td>
</tr>
<tr>
<td><strong>Our-Method</strong></td>
<td><strong>98.7±0.2</strong></td>
<td><strong>96.9±0.1</strong></td>
<td><strong>91.4±0.3</strong></td>
<td><strong>89.1±0.1</strong></td>
<td><strong>72.5±0.2</strong></td>
<td><strong>70.4±0.1</strong></td>
<td><strong>40.8±0.3</strong></td>
<td><strong>39.9±0.3</strong></td>
</tr>
</tbody>
</table>

Table 2. AUC (%) for unseen-class identification on MNIST.

<table>
<thead>
<tr>
<th>Method</th>
<th>ratio=0</th>
<th>ratio=0.1</th>
<th>ratio=0.2</th>
<th>ratio=0.3</th>
<th>ratio=0.4</th>
<th>ratio=0.5</th>
<th>ratio=0.6</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilities</td>
<td>84.3 ± 0.9</td>
<td>84.3 ± 0.9</td>
<td>84.3 ± 0.9</td>
<td>84.3 ± 0.9</td>
<td>84.3 ± 0.9</td>
<td>84.3 ± 0.9</td>
<td>84.3 ± 0.9</td>
<td>84.3</td>
</tr>
<tr>
<td>DS3L</td>
<td>95.9 ± 0.8</td>
<td>93.1 ± 0.4</td>
<td>91.7 ± 0.2</td>
<td>90.6 ± 0.1</td>
<td>90.5 ± 0.5</td>
<td>89.1 ± 0.2</td>
<td>85.1 ± 0.8</td>
<td>90.9</td>
</tr>
<tr>
<td>SAFE-STUDENT</td>
<td>98.1 ± 0.1</td>
<td>97.3 ± 0.2</td>
<td>96.5 ± 0.1</td>
<td>96.0 ± 0.9</td>
<td>94.6 ± 0.9</td>
<td>93.5 ± 0.3</td>
<td>91.4 ± 0.2</td>
<td>95.3</td>
</tr>
<tr>
<td>Proposed</td>
<td>98.4 ± 0.1</td>
<td>97.8 ± 0.1</td>
<td>97.0 ± 0.2</td>
<td>96.5 ± 0.3</td>
<td>95.2 ± 0.5</td>
<td>94.0 ± 0.2</td>
<td>92.1 ± 0.3</td>
<td>95.9</td>
</tr>
<tr>
<td><strong>Proposed(ID&amp;OOD)</strong></td>
<td><strong>98.7±0.3</strong></td>
<td><strong>98.0±0.2</strong></td>
<td><strong>97.5±0.1</strong></td>
<td><strong>97.0±0.2</strong></td>
<td><strong>96.1±0.4</strong></td>
<td><strong>94.8±0.2</strong></td>
<td><strong>93.0±0.2</strong></td>
<td><strong>96.4</strong></td>
</tr>
</tbody>
</table>

Table 3. Seen-class classification accuracy (%) of ablation study on CIFAR-100 with mismatch ratio of 0.3.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>our method w/o (\mathcal{L}_c, \mathcal{L}_p) and (\mathcal{L}_n)</td>
<td>64.7</td>
</tr>
<tr>
<td>our method w/o (\mathcal{L}_p, \mathcal{L}_n)</td>
<td>70.0</td>
</tr>
<tr>
<td>our method w/o (\mathcal{L}_n)</td>
<td>71.6</td>
</tr>
<tr>
<td>our method w/o (\mathcal{L}_p)</td>
<td>71.4</td>
</tr>
<tr>
<td><strong>our method</strong></td>
<td><strong>72.5</strong></td>
</tr>
</tbody>
</table>

Identification ability of our method by comparing it with probability method [12], DS3L [8], SAFE-STUDENT [10] on MNIST in this experiment. Similar to [16], we also use AUC to evaluate the identification ability, in which we regard the unseen class data as a negative class and the others as a positive one. Table 2 lists the comparison results under different mismatch ratios. It can be found that the recent DS3L and SAFE-STUDENT lag behind our method by 5% and 0.6% accuracy, respectively. To improve the identification accuracy, we further make some attempts to modulate the way of splitting ID and OOD samples. We denote the improved version as proposed (ID&OOD). It clearly rises the average accuracy from 95.9% to 96.4%, proving the excellent ability of our method.

4.4.4 Insight of Feature Distribution

To better support the effectiveness of our idea of turning N-way classification to \((K+1)\)-way classification, we visualize the feature distributions of our method with \((K+1)\)-way classification (denoted as Ours (K+1)), our method with \(K\)-way classification (denoted as Ours (K)), DS3L [8], and SAFE-STUDENT [10] by t-SNE [34] plotting. All the models are trained on CIFAR-10 (Fewer classes for bet-
ter presentation). As shown in Figure 4, although SAFE-STUDENT clusters the features of different classes in a more compact way than DS3L, our method can perform better by reducing the intra-class distance and increasing the inter-class distance. Without treating the OOD data as an individual class, the OOD features have already shown a tendency to gather. Once when we explicitly build the OOD prototype, all the classes, including the unseen-class marked in black, can be better distinguished, demonstrating the advancement of our powerful idea.

5. Conclusion

In this paper, we rethink the safe SSL problem from a novel aspect. Concretely, based on the experimental phenomenon that OOD data appear to cluster in the feature space, we propose a prototype-based framework to explicitly model the prototype of all OOD data. By treating the OOD prototype as equivalent to other ID prototypes, we transform the safe semi-supervised classification problem from open-set to close-set. Therefore, we can solve safe SSL using standard SSL techniques, including consistency regularization and pseudo-labeling. For the pseudo-labeling, we develop an INL paradigm to make full use of low-confidence pseudo labels by excavating more knowledge on wider classes. Extensive experiments on popular benchmarks demonstrate the effectiveness and superiority of our method.

6. Acknowledgement

This work is supported by National Natural Science Foundation of China (NSFC 62071314), Sichuan Science and Technology Program 2023YFG0263, 2023YFG0025, 2023NSFSC0497, and Opening Foundation of Agile and Intelligent Computing Key Laboratory of Sichuan Province.

References


