How Out-of-Distribution Data Hurts Semi-Supervised Learning

Xujiang Zhao*
NEC Laboratories America
xuzhao@nec-labs.edu

Killamsetty Krishnateja*, Rishabh Iyer, Feng Chen
The University of Texas at Dallas
{krishnateja.killamsetty,rishabh.iyer,feng.chen}@utdallas.edu

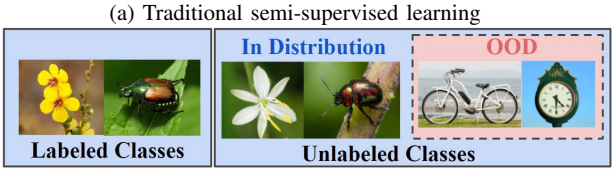
2022 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-5099-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDM54844.2022.00087

***Abstract*—Recent semi-supervised learning algorithms have demonstrated greater success with higher overall performance due to the use of better-unlabeled data representations. Nonetheless, recent research suggests that the performance of the SSL algorithm can be degraded when the unlabeled set contains out-of-distribution examples (OODs). This work addresses the following research question: *How do out-of-distribution (OOD) data adversely affect semi-supervised learning algorithms?* To answer this question, we investigate the critical causes of OOD's negative effect on SSL algorithms. In particular, we found that 1) certain kinds of OOD data instances that are close to the decision boundary have a more significant impact on performance than those that are further away, and 2) Batch Normalization (BN), a popular module, may degrade rather than improve performance when the unlabeled set contains OODs. To address these challenges, we developed a unified weighted robust SSL framework that can be easily extended to many existing SSL algorithms and improve their robustness against OODs. Having identified the limitations of low-order approximations in bi-level optimization, we developed an efficient bi-level optimization algorithm that could accommodate high-order approximations of the objective and could scale to a large number of inner optimization steps to learn a massive number of weight parameters. Furthermore, we conduct a theoretical analysis of the impact of faraway OODs in the BN step and propose a weighted batch normalization (WBN) procedure that uses the weights estimated by the bi-level optimization problem in the BN step. Additionally, we discuss the connection between our approach and low-order approximation techniques. Our extensive experiments on synthetic and real-world datasets demonstrate that our proposed approach significantly enhances the robustness of four representative SSL algorithms against OODs compared to four state-of-the-art robust SSL strategies.**

I. Introduction

Deep learning approaches have been shown to be successful on several supervised learning tasks, such as computer vision [6, 25], natural language processing [28], and speech recognition [31]. However, these deep learning models are data-hungry and often require massive amounts of labeled examples to obtain good performance. Obtaining high-quality labeled examples can be very time-consuming and expensive, particularly where specialized skills are required in labeling (for example, in cancer detection on X-ray or CT-scan images). As a result, semi-supervised learning (SSL) has emerged as a very promising direction, where the learning algorithms try to effectively utilize the large unlabeled set (in conjunction) with a relatively small labeled set. Several recent SSL algorithms have been proposed for deep learning and have shown great promise empirically. These include Entropy Minimization [7], pseudo-label based methods [15, 1, 3] and consistency based methods [20, 14, 24, 17] to name a few.

(a) Traditional semi-supervised learning



(b) Semi-supervised learning with OOD data

Fig. 1: (a) Traditional SSL. (b) SSL with OODs.

Despite the success of these SSL methods, they are designed with the assumption that labeled and unlabeled data have the same distribution. Fig 1 (a) shows an example of this. However, this assumption may not hold in many real-world applications, such as web classification and medical diagnosis, where some unlabeled examples are from novel classes unseen in the labeled data. For example, Fig 1 (b) illustrates an image classification scenario with out-of-distribution data, where the unlabeled dataset contains two novel classes (bicycle and clock) compared to the in-distribution classes (flower and beetle) in the labeled dataset. When the unlabeled set contains OOD examples (OODs), deep SSL performance can degrade substantially and is sometimes even worse than simple supervised learning (SL) approaches [18]. Moreover, it is unreasonable to expect a human to go through and clean a large and massive unlabeled set in such cases.

A typical approach to robust SSL against OODs is to assign a weight to each unlabeled example based on some criteria and minimize a weighted training or validation loss. In an ideal weighting scheme, positive weights should be assigned only to ID samples while zero weights should be assigned to OOD samples. Yan et al. [29] applied a set of weak annotators to approximate the ground-truth labels as pseudo-labels to learn a robust SSL model. [4] proposed a distributionally robust model that estimates a parametric weight function based on

* Equal Contribution. This work is done when Xujiang Zhao was with The University of Texas at Dallas.

978-1-6654-5099-7/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDM54844.2022.00087

both the discrepancy and the consistency between the labeled data and the unlabeled data. [5] (UASD) proposed to weigh the unlabeled examples based on an estimation of predictive uncertainty for each unlabeled example. The goal of UASD is to discard potentially irrelevant samples having low confidence scores and estimate the parameters by optimizing a regularized training loss.

A state-of-the-art method [9], called DS3L, considers a shallow neural network to predict the weights of unlabeled examples and estimate the parameters of the neural network based on a clean labeled set via bi-level optimization. It is common to obtain a dataset composed of two parts, including a relatively small but accurately labeled set and a large but coarsely labeled set from inexpensive crowd-sourcing services or other noisy sources.

There are three **main limitations** of DS3L and other methods as reviewed above. First, it lacks a study of potential causes about the impact of OODs on SSL, and as a result, the interpretation of robust SSL methods becomes difficult. Second, existing robust SSL methods did not consider the negative impact of OODs on the utilization of BN in neural networks, and as a result, their robustness against OODs degrades significantly when a neural network includes BN layers. The utilization of BNs for deep SSL has an implicit assumption that the labeled and unlabeled examples follow a single or similar distributions, which is problematic when the unlabeled examples include OODs [11]. Third, the bi-level learning algorithm developed in DS3L relies on low-order approximations of the objective in the inner loop due to vanishing gradients or memory constraints. As a result of not using the high-order loss information, the learning performance of DS3L could be significantly degraded in some applications, as demonstrated in our experiments. Our main technical contributions over existing methods are summarized as follows:

The effect of OOD data points. The first critical contribution of our work (Sec. III) is to analyze what kind of OOD unlabeled data points affect the performance of SSL algorithms. In particular, we observe that OOD samples lying close to the decision boundary have more influence on SSL performance than those far from the boundary. Furthermore, we observe that the OOD instances far from the decision-boundary (faraway OODs) can degrade SSL performance substantially if the model contains a batch normalization (BN) layer. The last observation makes sense logically as well since the batch normalization heavily depends on the mean and variance of each batch's data points, which can be significantly different for OOD points that came from very different distributions. We find these observations about OOD points consistent across experiments on several synthetic and real-world datasets.

Weighted Robust SSL Framework. Our second contribution is a unified, weighted robust SSL approach to improve many existing SSL algorithms' robustness by learning to assign weights to unlabeled examples based on a bi-level optimization approach. To address the limitation of low-order approximations in bi-level optimization (DS3L), we designed an implicit-differentiation based algorithm that considered high-order approximations of the objective and is scalable to a higher number of inner optimization steps to learn a massive amount of weight parameters. In addition to address the BN issue due to the faraway OODs, we propose *weighted batch normalization (WBN)* to carry the weights (learned from bi-level optimization) over in the BN step (Sec. IV-D).

Comprehensive experiments. We conduct extensive experiments on synthetic and real-world datasets. The results demonstrate that our weighted robust SSL approach significantly outperforms existing robust approaches (L2RW, MWN, Safe-SSL, and UASD) on four representative SSL algorithms. We also perform an ablation study to demonstrate which components of our approach are most important for its success.

II. Semi-Supervised Learning (SSL)

Given a training set with a labeled set of examples $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ and an unlabeled set of examples $\mathcal{U} = \{\mathbf{x}_j\}_{j=1}^m$. For any classifier model $f(\mathbf{x}, \theta)$ used in SSL, where $\mathbf{x} \in \mathbb{R}^C$ is the input data, and θ refers to the parameters of the classifier model. The loss functions of many existing methods can be formulated as the following general form:

$$\sum\nolimits_{(\mathbf{x}_i, y_i) \in \mathcal{D}} l(f(\mathbf{x}_i, \theta), y_i) + \sum\nolimits_{x_j \in \mathcal{U}} r(f(\mathbf{x}_j, \theta)), \qquad (1)$$

where $l(\cdot)$ is the loss function for labeled data (such as cross-entropy), and $r(\cdot)$ is the loss function (regularization function) on the unlabeled set. The goal of SSL methods is to design an efficient regularization function to leverage the model performance information on the unlabeled dataset for effective training. Pseudo-labeling [15] uses a standard supervised loss function on an unlabeled dataset using "pseudo-labels" as a target label as a regularizer. Π-Model [14, 20] designed a consistency-based regularization function that pushes the distance between the prediction for an unlabeled sample and its stochastic perturbation (e.g., data augmentation or dropout [23]) to a small value. Mean Teacher [24] proposed to obtain a more stable target output $f(x, \theta)$ for unlabeled set by setting the target via an exponential moving average of parameters from previous training steps. Instead of designing a stochastic $f(x, \theta)$, Virtual Adversarial Training (VAT) [17] proposed to approximate a tiny perturbation to unlabeled samples that affect the output of the prediction function most. MixMatch [3], UDA [27], and Fix-Match [22] choose the pseudo-labels based on predictions of augmented samples, such as shifts, cropping, image flipping, weak and strong augmentation, and mix-up [30] to design the regularization functions. However, the performance of most existing SSL can degrade substantially when the unlabeled dataset contains OOD examples [18].

III. Impact of OOD on SSL Performance

In this section, we provide a systematic analysis of the impact of OODs for many popular SSL algorithms, such as Pseudo-Label(PL) [15], Π-Model [14], Mean Teacher(MT) [24], and Virtual Adversarial Training (VAT) [17]. We illustrate the discoveries using the following

synthetic and real-world datasets. While we mainly focus on VAT as the choice of the SSL algorithm, the observations extend to other SSL algorithms as well.

Synthetic dataset. We considered two moons dataset (red points are labeled data, gray circle points are in-distribution (ID) unlabeled data) with OOD (yellow triangle points) points in three different scenarios that can exist in real-world, 1) Faraway OOD scenario where the OOD points exist far from decision boundary; 2) Boundary OOD scenario where the OOD points occur close to decision boundary; 3) Mixed OOD scenario where OOD points exist both far and close to the decision boundary, as shown in Fig 2.

Real-world dataset. We consider MNIST as ID data with three types of OODs to account for plausible real-world scenarios. 1)Faraway OOD: We used Fashion MNIST (F-MNIST) dataset, which contains fashion images as Faraway OOD dataset as it inherently has different patterns compared to MNIST dataset; 2) Boundary OOD: We used EMNIST dataset, which contains handwritten character digits as Boundary OOD dataset as it has similar patterns compared to MNIST dataset; In addition to EMNIST, we also considered Mean MNIST (M-MNIST) as a boundary OOD dataset, which was generated by averaging MNIST images from two different classes (usage of M-MNIST as boundary OOD is also considered in [8]); 4) Mixed OOD: For Mixed OOD dataset, we combined both Fashion MNIST and EMNIST together.

For all experiments in this section, we used a multilayer perceptron neural network (MLP) with three layers as a backbone architecture for the synthetic dataset and LeNet as a backbone for the real-world datasets. We consider the following models in the experiments: 1) *SSL-NBN*: MLP or LeNet model without Batch Normalization; 2) *SSL-BN*: MLP or LeNet model with Batch Normalization; 3) *SSL-FBN*: MLP or LeNet model where we freeze the batch normalization layers for the unlabeled instances. Freezing BN (FBN) [18] is a common trick to improve the SSL model robustness where we freeze batch normalization layers by not updating *running_mean* and *running_variance* in the training phase.

The following are the main observations. First, from Fig 2, we see that with BN (i.e., SSL-BN), there is a significant impact on model performance and learned decision boundaries in the presence of OOD. This performance degradation is even more pronounced in Faraway OOD since the BN statistics like the running mean/variance can be significantly changed by faraway OOD points. Secondly, when we do not use BN (i.e., SSL-NBN), the impact of the Faraway OOD and mixed OOD data is reduced. However, in the case of boundary OOD (Fig 2 (b) and EMNIST/M-MNIST case of Fig 2 (d)), we still see significant performance degradation compared to the skyline. However, BN is a crucial component in more complicated models (Eg: ResNet family), and we expect OOD instances to play a significant role there. Finally, when freezing the BN layers for the unlabeled data (i.e., SSL-FBN), we see that the Faraway and Mixed OODs' effect is alleviated; but SSL-FBN still performs worse than the SSL-NBN in Faraway and Mixed OODs (and there is big scope of improvement w.r.t the skyline). Finally, both SSL-NBN and SSL-FBN fail to efficiently mitigate the performance degradation caused by boundary OOD data points. We also show that similar observations made on the CIFAR-10 dataset (Fig 2 (e)).

Proposition 1. *Give in-distribution mini-batch $\mathcal{I} = \{\boldsymbol{x}_i\}_{i=1}^m$, OOD mini-batch $\mathcal{O} = \{\hat{\boldsymbol{x}}_i\}_{i=1}^m$, and the mixed mini-batch $\mathcal{IO} = \mathcal{I} \cup \mathcal{O}$. Denote $\mu_{\mathcal{M}}$ is the mini-batch mean of $\mathcal{M}$ (either $\mathcal{O}, \mathcal{I}$ or $\mathcal{IO}$). With faraway OOD: $\|\mu_{\mathcal{O}} - \mu_{\mathcal{I}}\|_2 > L$, where L is large ($L \gg 0$), we have:*

$$\|\mu_{\mathcal{IO}} - \mu_{\mathcal{I}}\|_2 > \frac{L}{2} \quad \textit{and} \quad BN_{\mathcal{I}}(\boldsymbol{x}_i) \neq BN_{\mathcal{IO}}(\boldsymbol{x}_i)$$

where $BN_{\mathcal{M}}(\boldsymbol{x}_i)$ is traditional batch normalizing transform based on mini-batch from $\mathcal{M}$ (either $\mathcal{I}$ or $\mathcal{IO}$).

Proof: The mini-batch mean of $\mathcal{I}$: $\mu_{\mathcal{I}} = \frac{1}{m}\sum_{i=1}^m \mathbf{x}_i$. The mixed mini-batch mean of $\mathcal{IO}$: $\mu_{\mathcal{IO}} = \frac{1}{2m}(\sum_{i=1}^m \mathbf{x}_i + \sum_{i=1}^m \hat{\mathbf{x}}_i) = \frac{1}{2}\mu_{\mathcal{I}} + \frac{1}{2}\mu_{\mathcal{O}}$. Then we have,

$$\|\mu_{\mathcal{IO}} - \mu_{\mathcal{I}}\|_2 = \|\frac{1}{2}\mu_{\mathcal{I}} + \frac{1}{2}\mu_{\mathcal{O}} - \mu_{\mathcal{I}}\|_2 > \frac{L}{2} \gg 0$$

The mini-batch variance of $\mathcal{I}$: $\sigma_{\mathcal{I}}^2 = \frac{1}{m}\sum_{i=1}^m (\mathbf{x}_i - \mu_{\mathcal{I}})^2$. The traditional batch normalizing transform based on mini-batch $\mathcal{I}$ for $\mathbf{x}_i$: $BN_{\mathcal{I}}(\mathbf{x}_i) = \gamma \frac{\mathbf{x}_i - \mu_{\mathcal{I}}}{\sqrt{\sigma_{\mathcal{I}}^2 + \epsilon}} + \beta$. The mini-batch variance of $\mathcal{IO}$,

$$\begin{aligned} \sigma_{\mathcal{IO}}^2 &= \frac{1}{2m}(\sum_{i=1}^m \mathbf{x}_i^2 + \sum_{i=1}^m \hat{\mathbf{x}}_i^2) - \mu_{\mathcal{IO}}^2 \\ &= \frac{1}{2}\sigma_{\mathcal{I}}^2 + \frac{1}{2}\sigma_{\mathcal{O}}^2 + \frac{1}{4}(\mu_{\mathcal{O}} - \mu_{\mathcal{I}})^2 \\ &\approx \frac{1}{4}(\mu_{\mathcal{O}} - \mu_{\mathcal{I}})^2, \end{aligned} \tag{2}$$

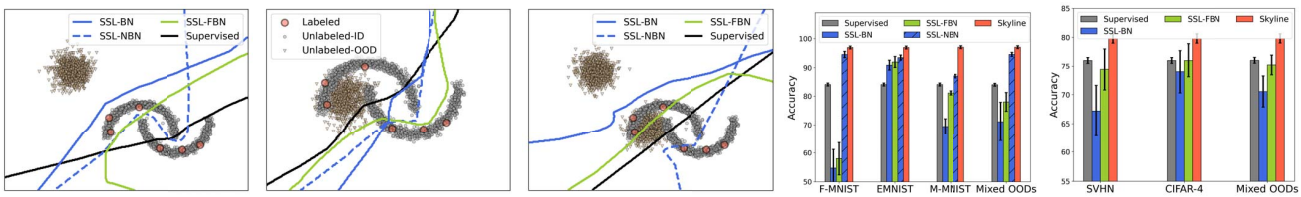
and the traditional batch normalizing transform based on mini-batch $\mathcal{IO}$ for $\mathbf{x}_i$,

$$\begin{aligned} BN_{\mathcal{IO}}(\mathbf{x}_i) &= \gamma \frac{\mathbf{x}_i - \mu_{\mathcal{B}}(IO)}{\sqrt{\sigma_{\mathcal{B}}^2(IO) + \epsilon}} + \beta \\ &= \gamma\Big(\frac{\mathbf{x}_i - \mu_O}{2\sqrt{\sigma_{\mathcal{B}}^2(IO) + \epsilon}} + \frac{\mathbf{x}_i - \mu_I}{2\sqrt{\sigma_{\mathcal{B}}^2(IO) + \epsilon}}\Big) + \beta \\ &\approx \gamma\Big(\frac{\mathbf{x}_i - \mu_O}{\|\mu_O - \mu_I\|_2} + \frac{\mathbf{x}_i - \mu_I}{\|\mu_O - \mu_I\|_2}\Big) + \beta \\ &\approx \gamma \frac{\mathbf{x}_i - \mu_O}{\|\mu_O - \mu_I\|_2} + \beta. \end{aligned} \tag{3}$$

The approximations in Eq. (2) and Eq. (3) hold when $\sigma_{\mathcal{I}}^2$ and $\sigma_{\mathcal{O}}^2$ have the same magnitude levels as μ_I, i.e., $\|\mu_{\mathcal{O}} - \sigma_{\mathcal{I}}^2\|_2 > L$ and $\|\mu_{\mathcal{O}} - \sigma_{\mathcal{O}}^2\|_2 > L$. Comparing Eq. (III) with Eq. (3), we prove that $BN_{\mathcal{I}}(\mathbf{x}_i) \neq BN_{\mathcal{IO}}(\mathbf{x}_i)$. ■

Proposition 1 shows that when unlabeled set contains OODs, the traditional BN behavior could be problematic, therefore resulting in incorrect statistics estimation, e.g., the output of BN for mixed mini-bath $BN_{\mathcal{IO}}(\mathbf{x}_i) \approx \gamma \frac{\mathbf{x}_i - \mu_{\mathcal{O}}}{\|\mu_{\mathcal{O}} - \mu_{\mathcal{I}}\|_2} + \beta$, which is not our expected result ($BN_{\mathcal{I}}(\mathbf{x}_i) = \gamma \frac{\mathbf{x}_i - \mu_{\mathcal{I}}}{\sqrt{\sigma_{\mathcal{I}}^2 + \epsilon}} + \beta$).

The main takeaways of the synthetic and real data experiments are as follows: 1) OOD instances close to decision



(a) Faraway OODs (b) Boundary OODs (c) Mixed OODs (d) OODs in MNIST (e) OODs in CIFAR10

Fig. 2: SSL performance with different type OODs in both synthetic and real-world datasets.

boundary (Boundary OODs) hurt SSL performance irrespective of the use of batch normalization; 2) OOD instances far from the decision boundary (Faraway OODs) hurt the SSL performance if the model involves BN. Freezing BN can reduce some impact of OOD to some extent but not entirely; 3) OOD instances far from the decision boundary will not hurt SSL performance if there is no BN in the model. To this end, we answered the question "*How out-of- distribution data hurt semi-supervised learning performance?*". In the next section, we propose a weighted robust SSL framework to address above mentioned issues caused by OOD data points.

IV. Methodology

In this section, we first proposed the weighted robust SSL (WR-SSL) framework in Sec. IV-A, and then introduce the implicit-differentiation (high-order approximation) based optimization algorithms to train the weighted robust SSL approach in Sec. IV-B. More importantly, we proposed weighted batch normalization to improve the robustness of our WR-SSL framework against OODs in Sec. IV-D.

A. Weighted Robust SSL Framework

Reweighting the unlabeled data. Consider the semi-supervised classification problem with training data (labeled $\mathcal{D}$ and unlabeled $\mathcal{U}$) and classifier $f(x;\theta)$. Generally, the optimal classifier parameter θ can be extracted by minimizing the SSL loss (Eq. (1)) calculated on the training set. In the presence of unlabeled OOD data, sample reweighting methods enhance the robustness of training by imposing weight w_j on the j-th unlabeled sample loss,

$$\mathcal{L}_T(\theta,\mathbf{w}) = \sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}} l(f(\mathbf{x}_i;\theta),y_i) + \sum_{x_j\in\mathcal{U}} w_j r(f(\mathbf{x}_j;\theta)),$$

where we denote $\mathcal{L}_U$ is the robust unlabeled loss, and we treat weight $\mathbf{w}$ as hyperparameter. Our goal is to learn a sample weight vector $\mathbf{w}$ such that $\mathbf{w}=0$ for OODs, $\mathbf{w}=1$ for In-distribution (ID) sample.

Denote $\mathcal{L}_V(\theta^*(\mathbf{w}),\mathbf{w}) = \mathcal{L}_V(\theta^*(\mathbf{w})) + \lambda\cdot\text{Reg}(\mathbf{w})$ as the validation loss with a regularization term over the validation dataset, where $\text{Reg}(\mathbf{w})$ is the regularization term, λ is the regularization coefficient, and $\mathcal{L}_V(\theta^*(\mathbf{w})) \triangleq \sum_{(\mathbf{x}_i,y_i)\in\mathcal{V}} l(f(\mathbf{x}_i,\theta^*(\mathbf{w})),y_i)$. This labeled set could either be a held out validation set, or the original labeled set $\mathcal{D}$. Intuitively, the problem given in Eq. (4) aims to choose weights of unlabeled samples $\mathbf{w}$ that minimizes the supervised loss evaluated on the validation set when the model parameters $\theta^*(\mathbf{w})$ are optimized by minimizing the weighted SSL loss $\mathcal{L}_T(\theta,\mathbf{w})$.

Bi-level optimization objective. Since manual tuning and grid-search for each w_i is intractable, we pose the weights optimization problem described above as a *bi-level* optimization problem.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathcal{L}_V(\theta^*(\mathbf{w}),\mathbf{w}), \\ \text{s.t.} \quad & \theta^*(\mathbf{w}) = \arg\min_{\theta} \mathcal{L}_T(\theta,\mathbf{w}). \end{aligned} \tag{4}$$

Calculating the optimal θ^* and $\mathbf{w}$ requires two nested loops of optimization, which is expensive and intractable to obtain the exact solution, especially when optimization involves deep learning model and large datasets. Since gradient-based methods like Stochastic Gradient Descent (SGD) have shown to be very effective for machine learning and deep learning problems, we adapt a high-order approximation strategy, as described in Sec IV-B.

B. high-order Optimization Approximation

In this section, we developed an efficient high-order optimization algorithms to train our weighted robust SSL framework.

Implicit Differentiation. Directly calculate the weight gradient $\frac{\partial\mathcal{L}_V(\theta^*(\mathbf{w}),\mathbf{w})}{\partial\mathbf{w}}$ by chain rule:

$$\frac{\partial\mathcal{L}_V(\theta^*(\mathbf{w}),\mathbf{w})}{\partial\mathbf{w}} = \underbrace{\frac{\partial\mathcal{L}_V}{\partial\mathbf{w}}}_{(a)} + \underbrace{\frac{\partial\mathcal{L}_V}{\partial\theta^*(\mathbf{w})}}_{(b)} \times \underbrace{\frac{\partial\theta^*(\mathbf{w})}{\partial\mathbf{w}}}_{(c)} \tag{5}$$

where (a) is the weight direct gradient (e.g., gradient from regularization term, $\text{Reg}(\mathbf{w})$), (b) is the parameter direct gradient, which are easy to compute. The difficult part is the term (c) (best-response Jacobian). We approximate (c) by using the Implicit function theorem [16],

$$\frac{\partial\theta^*(\mathbf{w})}{\mathbf{w}} = -\underbrace{\left[\frac{\partial\mathcal{L}_T}{\partial\theta\partial\theta^T}\right]^{-1}}_{(d)} \times \underbrace{\frac{\partial\mathcal{L}_T}{\partial\mathbf{w}\partial\theta^T}}_{(e)} \tag{6}$$

However, computing Eq. (6) is challenging when using deep nets because it requires to invert a high dimensional Hessian (term (d)), which often require $\mathcal{O}(m^3)$ operations. Therefore, we give the Neumann series approximations [16] of term (d) which we empirically found to be effective for SSL,

$$\left[\frac{\partial\mathcal{L}_T}{\partial\theta\partial\theta^T}\right]^{-1} \approx \lim_{P\to\infty}\sum_{p=0}^{P}\left[I - \frac{\partial\mathcal{L}_T}{\partial\theta\partial\theta^T}\right]^p \tag{7}$$

where I is the identity matrix.

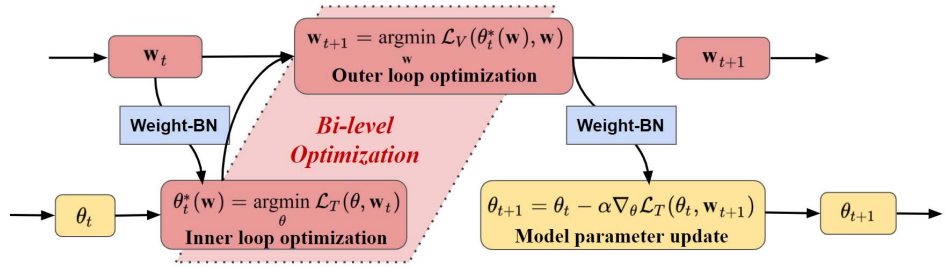


Fig. 3: Main flowchart of the proposed Weighted Robust SSL algorithm.

Since the algorithm mentioned in [16] utilizes the Neumann series approximation and efficient hessian vector product to compute the Hessian inverse product, it can efficiently compute the hessian-inverse product even when a larger number of weight hyperparameters are present. We should also note that the implicit function theorem's assumption $\frac{\partial \mathcal{L}_T}{\partial \theta} = 0$ needs to be satisfied to accurately calculate the Hessian inverse product. However in practice, we only approximate θ^*, and simultaneously train both $\mathbf{w}$ and θ by alternatively optimizing θ using $\mathcal{L}_T$ and $\mathbf{w}$ using $\mathcal{L}_V$.

C. Connections to low-order Approximation

Lemma 1. *Suppose that the Hessian inverse of training loss $\mathcal{L}_T$ with the model parameters θ is equal to the identity matrix $\frac{\partial^2 \mathcal{L}_T}{\partial \theta \partial \theta^T} = \mathbf{I}$ (i.e.,$P = 0$ for implicit differentiation approach). Suppose the model parameters are optimized using single-step gradient descent (i.e., $J = 1$ for the low-order approximation approach), and the model learning rate is equal to one. Then, the weight update step in both implicit differentiation and low-order approximation approach is equal.*

Proof:

Case 1: ***Our Approach:*** *In Eq. 7, we have* $\left[\frac{\partial \mathcal{L}_T}{\partial \theta \partial \theta^T}\right]^{-1} = \mathbf{I}$ *and substituting it in Eq.* (6)*, we have:*

$$\frac{\partial \theta^*(\boldsymbol{w})}{\boldsymbol{w}} = -\frac{\partial \mathcal{L}_T}{\partial \boldsymbol{w} \partial \theta^T} \tag{8}$$

Substituting the above equation in Eq. (5)*, we have:*

$$\frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}), \boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{\partial \mathcal{L}_V}{\partial \boldsymbol{w}} - \frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}))}{\partial \theta^*(\boldsymbol{w})} \times \frac{\partial^2 \mathcal{L}_T}{\partial \boldsymbol{w} \partial \theta^T} \tag{9}$$

Since, we are using unweighted validation loss $\mathcal{L}_V$, there is no dependence of validation loss on weights directly i.e., $\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{w}} = 0$. Hence, the weight gradient is as follows:

$$\frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}), \boldsymbol{w})}{\partial \boldsymbol{w}} = -\frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}))}{\partial^2 \theta^*(\boldsymbol{w})} \times \frac{\partial \mathcal{L}_T}{\partial \boldsymbol{w} \partial \theta^T} \tag{10}$$

Since we are using one-step gradient approximation, we have $\theta^*(\boldsymbol{w}) = \theta - \alpha \frac{\partial \mathcal{L}_T(\boldsymbol{w}, \theta)}{\partial \theta}$ where α is the model parameters learning rate.

The weight update step is as follows:

$$\boldsymbol{w}^* = \boldsymbol{w} + \beta \frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}))}{\partial \theta^*(\boldsymbol{w})} \times \frac{\partial^2 \mathcal{L}_T}{\partial \boldsymbol{w} \partial \theta^T} \tag{11}$$

where β is the weight learning rate.

Case 2: ***low-order approximation Approach***

In low-order approximation approach, we have $\theta^*(\boldsymbol{w}) = \theta - \alpha \frac{\partial \mathcal{L}_T(\boldsymbol{w}, \theta)}{\partial \theta}$ where α is the model parameters learning rate.

Using the value of θ^*, the gradient of validation loss with weight hyperparameters is as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}))}{\partial \boldsymbol{w}} &= \frac{\partial \mathcal{L}_V(\theta - \alpha \frac{\partial \mathcal{L}_T(\boldsymbol{w}, \theta)}{\partial \theta})}{\partial \boldsymbol{w}} \\ &= -\frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}))}{\partial \theta^*(\boldsymbol{w})} \times \alpha \frac{\partial^2 \mathcal{L}_T(\boldsymbol{w}, \theta)}{\partial \theta \partial \boldsymbol{w}^T} \end{aligned} \tag{12}$$

Assuming $\alpha = 1$, we have:

$$\frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}))}{\partial \boldsymbol{w}} = -\frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}))}{\partial \theta^*(\boldsymbol{w})} \times \frac{\partial^2 \mathcal{L}_T(\boldsymbol{w}, \theta)}{\partial \theta \partial \boldsymbol{w}^T} \tag{13}$$

Hence, the weight update step is as follows:

$$\boldsymbol{w}^* = \boldsymbol{w} + \beta \frac{\partial \mathcal{L}_V(\theta^*(\boldsymbol{w}))}{\partial \theta^*(\boldsymbol{w})} \times \frac{\partial^2 \mathcal{L}_T}{\partial \boldsymbol{w} \partial \theta^T} \tag{14}$$

where β is the weight learning rate

As shown in the cases above, we have a similar weight update. Hence, it inherently means that using the low-order approximation of $J = 1$ is equivalent to using an identity matrix as the Hessian inverse of training loss with θ. ■

Lemma 1 shows that the weight gradient from implicit differentiation is same as the weight gradient from the low-order approximation method (e.g., DS3L) when $P = 0, J = 1$.

D. Weighted Batch Normalization

In practice, most deep SSL model would use deep CNN. While BN usually serves as an essential component for many deep CNN models [10]. Specifically, BN normalizes input features by the mean and variance computed within each mini-batch. At the same time, OODs would indeed affect the SSL performance due to BN (discussed this issues in Sec. III). To address this issue, we proposed a *Weighted Batch Normalization* (WBN) that performs the normalization for each training mini-batch with sample weights $\mathbf{w}$. We present the WBN in Algorithm 1, where ϵ is a constant added to the mini-batch variance for numerical stability.

Proposition 2. *With faraway OOD: $\|\mu_{\mathcal{O}} - \mu_{\mathcal{I}}\|_2 > L$, where L is large ($L \gg 0$), given perfect weights $\boldsymbol{w} = \boldsymbol{w}_{\mathcal{I}} \cup \boldsymbol{w}_{\mathcal{O}}$, where $\boldsymbol{w}_{\mathcal{I}} = \mathbf{1}$ for mini-batch $\mathcal{I}$ and $\boldsymbol{w}_{\mathcal{O}} = \mathbf{0}$ for mini-batch $\mathcal{O}$, we have*

$$\mu_{\mathcal{I}} = \mu_{\mathcal{I}\mathcal{O}}^{\boldsymbol{w}} \quad \text{and} \quad BN_{\mathcal{I}}(\boldsymbol{x}_i) = WBN_{\mathcal{I}\mathcal{O}}(\boldsymbol{x}_i, \boldsymbol{w}) \tag{15}$$

where $\mu^{\mathbf{w}}_{\mathcal{IO}}$ is the weighted mini-batch mean of $\mathcal{IO}$, and $WBN_{\mathcal{IO}}(\boldsymbol{x}_i, \boldsymbol{w})$ is weighted batch normalizing transform based on the set $\mathcal{IO}$ with weights $\boldsymbol{w}$.

Proof: The symbols $\mathcal{I}, \mathcal{O}, \mathcal{IO}, \mu_{\mathcal{I}}, BN_{\mathcal{M}}(\mathbf{x}_i)$ are introduced in Proposition 1. The weighted mini-batch mean of $\mathcal{IO}$:

$$\begin{aligned}\mu^{\mathbf{w}}_{\mathcal{IO}} &= \frac{\sum_{i=1}^m w^i_{\mathcal{I}}\mathbf{x}_i + \sum_{i=1}^m w^i_{\mathcal{O}}\hat{\mathbf{x}}_i}{\sum_{i=1}^m w^i_{\mathcal{I}} + \sum_{i=1}^m w^i_{\mathcal{O}}} \\ &= \frac{\sum_{i=1}^m 1\cdot\mathbf{x}_i + \sum_{i=1}^m 0\cdot\hat{\mathbf{x}}_i}{\sum_{i=1}^m 1 + \sum_{i=1}^m 0} = \mu_I, \qquad (16)\end{aligned}$$

witch proves that $\mu_{\mathcal{I}} = \mu^{\mathbf{w}}_{\mathcal{IO}}$. The weighted mini-batch variance of $\mathcal{IO}$,

$$\begin{aligned}{\sigma^{\mathbf{w}}_{\mathcal{IO}}}^2 &= \frac{\sum_{i=1}^m w^i_{\mathcal{I}}(\mathbf{x}_i - \mu^{\mathbf{w}}_{\mathcal{IO}})^2}{\sum_{i=1}^m w^i_{\mathcal{I}} + \sum_{i=1}^m w^i_{\mathcal{O}}} + \frac{\sum_{i=1}^m w^i_{\mathcal{O}}(\hat{\mathbf{x}}_i - -\mu^{\mathbf{w}}_{\mathcal{IO}})}{\sum_{i=1}^m w^i_{\mathcal{I}} + \sum_{i=1}^m w^i_{\mathcal{O}}}{}^2 \\ &= \frac{\sum_{i=1}^m 1\cdot(\mathbf{x}_i - \mu_{\mathcal{IO}})^2}{m} = \sigma^2_{\mathcal{I}}.\end{aligned}$$

The weighted batch normalizing transform based on mini-batch $\mathcal{IO}$ for $\mathbf{x}_i$: $WBN_{\mathcal{IO}}(\mathbf{x}_i, \mathbf{w}) = \gamma\frac{\mathbf{x}_i - \mu^{\mathbf{w}}_{\mathcal{IO}}}{\sqrt{{\sigma^{\mathbf{w}}_{\mathcal{IO}}}^2 + \epsilon}} + \beta = \gamma\frac{\mathbf{x}_i - \mu_{\mathcal{I}}}{\sqrt{\sigma^2_{\mathcal{I}} + \epsilon}} + \beta$ which proves that $BN_{\mathcal{I}}(\mathbf{x}_i) = WBN_{\mathcal{IO}}(\mathbf{x}_i, \mathbf{w})$. ■

Proposition 2 shows that our proposed weighted batch normalization (WBN) can reduce the OOD effect and get the expected result. Therefore, our weighted robust SSL framework uses WBN instead of BN if model includes BN layer. Ablation studies in Sec. V demonstrates that such our approach with WBN can improve performance further. Finally, our weighted robust SSL framework is detailed in Algorithm 2.

Algorithm 1: Weighted Batch Normalization

Input: A mini-batch $\mathcal{M} = \{\mathbf{x}_i\}_{i=1}^m$ and sample weight $\mathbf{w} = \{w_i\}_{i=1}^m$; Parameters to be learned: γ, β
Output: $\{t_i = WBN_{\mathcal{M}}(\mathbf{x}_i, \mathbf{w})\}$

1 Weighted mini-batch mean: $\mu^{\mathbf{w}}_{\mathcal{M}} \leftarrow \frac{1}{\sum_{i=1}^m w_i}\sum_{i=1}^m w_i\mathbf{x}_i$
2 Weighted mini-batch variance: ${\sigma^{\mathbf{w}}_{\mathcal{M}}}^2 \leftarrow \frac{1}{\sum_{i=1}^m w_i}\sum_{i=1}^m w_i(\mathbf{x}_i - \mu^{\mathbf{w}}_{\mathcal{M}})^2$
3 Normalize: $\hat{x}_i \leftarrow \frac{\mathbf{x}_i - \mu^{\mathbf{w}}_{\mathcal{M}}}{\sqrt{{\sigma^{\mathbf{w}}_{\mathcal{M}}}^2 + \epsilon}}$
4 Scale and shift: $t_i \leftarrow \gamma\hat{x}_i + \beta \equiv WBN_{\mathcal{M}}(\mathbf{x}_i, \mathbf{w})$

E. Additional Implementation Details

In this subsection, we discuss additional implementational and practical tricks to make our weighted robust SSL scalable and efficient.

Last-layer gradients. Computing the gradients over deep models is time-consuming due to an enormous number of parameters in the model. To address this issue, we adopt a last-layer gradient approximation similar to [2, 13, 12] by only considering the last classification layer gradients of the classifier model in inner loop optimization (step 8 in algorithm 2). By simply using the last-layer gradients, we achieve significant speedups in weighted robust SSL.

Infrequent update w. We update the weight parameters every L iterations ($L \geq 2$). In our experiments, we see that we can set $L = 5$ without significant loss in accuracy. For MNIST experiments, we can be even more aggressive and set $L = 20$.

Algorithm 2: Weighted Robust SSL

Input: Labeled: $\mathcal{D}$, Unlabeled: $\mathcal{U}$, Reg param: λ
Output: Model params: θ, Instance/Cluster Weights: $\mathbf{w}$

1 Set $t = 0$; learning rate α, β;
2 Initialize model parameters θ and weight $\mathbf{w}$;
3 Apply K-means to $\mathcal{U}$ (if CRW);
4 Apply WBN instead of BN (if model includes BN);
5 **repeat**
6 /* Inner loop optimization, initial $\theta^0_t = \theta_t$ */
7 **for** $j = 1, \ldots, J$ **do**
8 $\theta^j_t(\mathbf{w}) = \theta^{j-1}_t - \alpha\nabla_\theta\mathcal{L}_T(\theta^{j-1}_t, \mathbf{w}_t)$
9 /* Outer loop optimization, set $\theta^*_t = \theta^J_t$ */
10 Approximate inverse Hessian via Eq. (7);
11 Calculate best-response Jacobian by Eq. (6);
12 Calculate weight gradient $\nabla_{\mathbf{w}}\mathcal{L}_V$ via Eq. (5);
13 update weight via $\mathbf{w}_{t+1} = \mathbf{w}_t - \beta\cdot\nabla_{\mathbf{w}}\mathcal{L}_V$;
14 /* Update net parameters */
15 $\theta_{t+1} = \theta_t - \alpha\nabla_\theta\mathcal{L}_T(\theta_t, \mathbf{w}_{t+1})$
16 $t = t + 1$
17 **until** *convergence*
18 **return** $\theta_{t+1}, \boldsymbol{w}_{t+1}$

Weight Sharing and Regularization. Considering the entire weight vector $\mathbf{w}$ (overall unlabeled points) is not practical for large datasets and easily overfits (see ablation study experiments), we propose two ways to fix this. The first is weight sharing via clustering, which we call Cluster Re-weight (CRW) method. Specifically, we use an unsupervised cluster algorithm (e.g., K-means algorithm) to embed unlabeled samples into K clusters and assign a weight to each cluster such that we can reduce the dimensionality of $\mathbf{w}$ from $|M|$ to $|K|$, where $|K| \ll |M|$. In practice, for high dimensional data, we may use a pre-trained model to calculate embedding for each point before applying the cluster method. In cases where we do not have an effective pre-trained model for embedding, we consider another variant that applies weights to every unlabeled point but considers an L1 regularization in Eq (4) for sparsity in $\mathbf{w}$. We show that both these tricks effectively improve the performance of reweighting and prevent overfitting on the validation set.

V. Experimental

To corroborate our algorithm, we conduct extensive experiments comparing our approach (WR-SSL) with some popular baseline methods. We aim to answer the following questions:

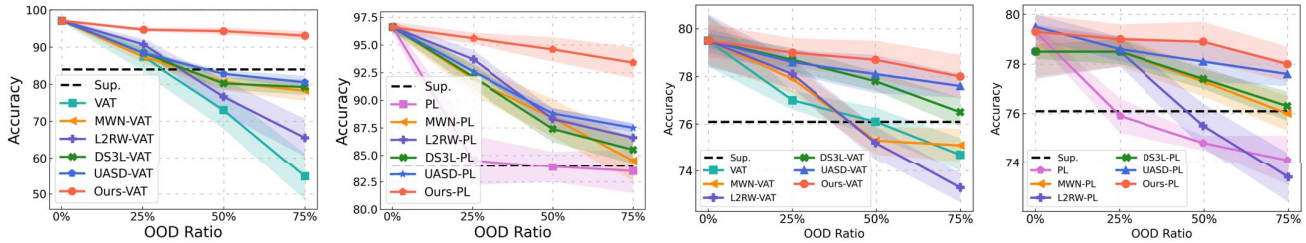
Question 1: Can our approach (WR-SSL) achieve better performance on both different types of OODs with varying OOD ratios compared with baseline methods?

Question 2: How do our approach compare in terms of running time comparing with baseline methods?

Question 3: What is the effect of each of the components of our approach (e.g., WBN, clustering/regularization, inverse Hessian approximation, inner loop gradient steps)?

A. Datasets

We consider four image classification benchmark datasets. (1) **MNIST**: a handwritten digit classification dataset, with



(a) Fashion MNIST. (b) Mean MNIST. (c) CIFAR10. (d) CIFAR10.

Fig. 4: (a-d) show test accuracy with varying OOD ratios of 0% to 75%. (a) shows results with MNIST as ID and F-MNIST as OOD, (b) shows results with MNIST as ID and M-MNIST as OOD, (c) (d) shows results with CIFAR10 dataset (6 classes ID, 4 classes OOD). (a) (c) (d) uses BN layers for all baselines except our methods that uses WBN layers. (b) shows all methods' results using a model w/o BN layers. (a) (c) use VAT, whereas (b) (d) use PL. Our methods consistently outperform all baselines across different datasets, base SSL algorithms, and models with or without BN layers.

50,000/ 10,000/ 10,000 training/validation/test samples, with training data, split into two groups – labeled and unlabeled in-distribution (ID) images (labeled data has ten images per class), and with two types of OODs: a) Fashion MNIST, b) Mean MNIST; (2) **CIFAR10**: a natural image dataset with 45,000/ 5,000/ 10,000 training/validation/test samples from 10 object classes, and following [18], we adapt CIFAR10 to a 6-class classification task, using 400 labels per class (from the 6 classes) and rest of the classes OOD (ID classes are "bird", "cat", "deer", "dog", "frog", "horse", and OOD data are from classes: "airline", "automobile", "ship", "truck"); (3) **CIFAR100**: another natural image dataset with 45,000/5000/10,000 training/validation/test images, similar to CIFAR10, we adapt CIFAR100 to a 50-class classification task, with 40 labels per class – the ID classes are the first 50 classes, and OOD data corresponds to the last 50 classes; (4) **SVHN-extra** : This is SVHN dataset with 531,131 additional digit images, and we adapt SVHN-extra to a 5-class classification task, using 400 labels per class. The ID classes are the first five classes, and OOD data corresponds to the last five classes.

B. Comparing Methods

To evaluate the effectiveness of our proposed weighted robust SSL approaches, we compare with five state-of-the-art robust SSL approaches, including UASD [5], DS3L (DS3L) [9], L2RW [19], and MWN [21]. The last two approaches L2RW and MWN, were originally designed for robust supervised learning (SL), and we adapted them to robust SSL by replacing the supervised learning loss function with an SSL loss function. We compare these robust approaches on four representative SSL methods, including Pseudo-Label (PL) [15], Π-Model (PI) [14, 20], Mean Teacher (MT) [24], and Virtual Adversarial Training (VAT) [17]. One additional baseline is the supervised learning method, named "Sup," which ignored all the unlabeled examples during training. Even Fixmatch [22] is the state-of-the-art SSL algorithm. We did not consider Fixmatch as the representative SSL method because Fixmatch is non-efficiency (100 hours for training in 1 2080Ti GPU) All the compared methods were built upon the open-source Pytorch implementation by [18].

C. Setup

In our experiments, we implement our approaches (Ours-SSL) for four representative SSL methods, including Pseudo-Label (PL), Π-Model (PI), Mean Teacher (MT), and Virtual Adversarial Training (VAT). The term "SSL" in Ours-SSL represent the SSL method, e.g., (Ours-VAT denotes our weighted robust SSL algorithm implemented based on VAT. We used the standard LeNet model as the backbone for the MNIST experiment and used *WRN-28-2* as the backbone for CIFAR10, CIFAR100, and SVHN experiments. For a comprehensive and fair comparison of the CIFAR10 experiment, we followed the same experiment setting of [18]. All the compared methods were built upon the Pytorch implementation by [18].

Hyperparameter setting. For our WR-SSL approach, we update the weights only using last layer for the inner optimization, we set $J = 3$ (for inner loop gradient steps), $P = 5$ (for inverse Hessian approximation) , $K = 20$ (for CRW), $\lambda = 10^{-7}$ (for L1), and $L = 5$ (for infrequent update) for all experiments. We trained all the networks for 2,000 updates with a batch size of 100 for MNIST experiments, and 500,000 updates with a batch size of 100 for CIFAR10, CIFAR100, and SVHN experiments. We did not use any form of early stopping but instead continuously monitored the validation set performance and report test error at the point of the lowest validation error. We show the specific hyperparameters used with four representative SSL methods on MNIST experiments in Table I. For CIFAR10, we used the same hyperparameters as [18]. For CIFAR100 and SVHN datasets, we used the same hyperparameters as CIFAR10[1].

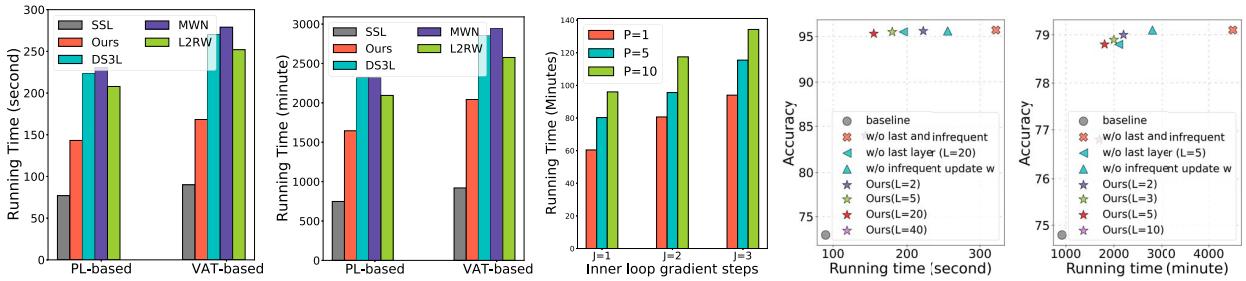
VI. Results and Discussion

A. Performance with different OOD datasets

In all experiments, we report the performance over five runs. Denote OOD ratio= $\mathcal{U}_{ood}/(\mathcal{U}_{ood}+\mathcal{U}_{in})$ where $\mathcal{U}_{in}$ is ID unlabeled set, $\mathcal{U}_{ood}$ is OOD unlabeled set, and $\mathcal{U} = \mathcal{U}_{in}+\mathcal{U}_{ood}$. The following experimental results on each dataset are to answer all **Questions** given in Sec. V.

In our experiments, we consider five different OOD datasets: Fashion MNIST, Mean MNIST, a subset of CIFAR-10, a

[1]The source code is accessible at https://github.com/zxj32/WR-SSL



(a) MNIST. (b) CIFAR10. (c) Running Time. (d) MNIST. (e) CIFAR10.

Fig. 5: Running time results. (a)-(b) show our proposed approaches are only $1.7\times$ to $1.8\times$ slower compared base SSL algorithms, while other robust SSL methods are $3\times$ slower. (c) shows that the running time of our method would increase with J (inner loop gradients steps) and P (inverse Hessian approximation) increase. (d)-(e) show running time of our strategies with different combinations of tricks viz; last layer updates and updating weights every L iterations. Note that by using only last layer updates, our strategies are around $2\times$ slower. With $L = 5$ and last layer updates, we are around $1.7\times$ to $1.8\times$ slower with comparable test accuracy.

TABLE I: Hyperparameter settings used in MNIST experiments for four representative SSL. All robust SSL methods (e.g., ours (WR-SSL), DS3L and UASD) are developed based on these representative SSL.

Shared	
Learning decayed by a factor of	0.2
at training iteration	1,000
coefficient = 1 (Do not use warmup)	
Supervised	
Initial learning rate	0.003
Π-Model	
Initial learning rate	0.003
Max consistency coefficient	20
Mean Teacher	
Initial learning rate	0.0004
Max consistency coefficient	8
Exponential moving average decay	0.95
VAT	
Initial learning rate	0.003
Max consistency coefficient	0.3
VAT ϵ	3.0
VAT ξ	10^{-6}
Pseudo-Label	
Initial learning rate	0.0003
Max consistency coefficient	1.0
Pseudo-label threshold	0.95

subset of CIFAR-100, and a subset of SVHN. The exact portions of OODs that belong to faraway OODs and boundary OODs, respectively, can not be shown explicitly as we did not find any existing methods can estimate the information. We roughly consider Fashion MNIST as a relatively faraway OODs since grayscale clothes images in Fashion MNIST are different from the digital number in MNIST. Moreover, we consider Mean MNIST (where the instances are mean images of two MNIST classes) as relative boundary OODs because the new fused images do not belong to MNIST class but contain very similar feature vectors (see Fig 6). For the subset OODs from CIFAR-10, CIFAR-100, and SVHN, we roughly consider them as relatively mixed type OODs. The subset may contain some classes close to in-distribution classes (e.g., seal v.s. whale) and some very different from in-distribution classes (e.g., flower v.s. fish).

We begin by running our algorithms on MNIST (i.e., MNIST as ID) with different types of OOD instances. First, we use Fashion MNIST [26] as OOD, which are grayscale article images. In this case, we compare all algorithms with batch normalization. As shown in Fig 4 (a), the performance of the existing SSL method decreases rapidly with an increase in OOD ratio. In contrast, our approach can still maintain clear performance improvement – e.g., our approach outperforms the base VAT SSL algorithm by almost 18% when OOD ratio = 50%. Compared with other robust SSL methods, our methods improve accuracy and suffer much less degradation under a high OOD ratio. Next, we use Mean MNIST (where the instances are mean images of two classes) as the OOD data. As shown in Fig 4 (b), we see a similar pattern that the accuracy of existing SSL methods decreases when the OOD ratio increases. Across different OOD ratios, our method significantly outperforms all baselines and the base SSL methods – e.g., 8% increase with our approach over DS3L when OOD ratio = 50%.

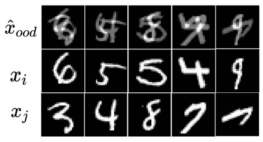


Fig. 6: Examples of Mean MNIST. $\hat{x}_{ood} = (x_i + x_j)/2$ where x_i, x_j are two samples from different MNIST classes.

We now study our algorithm's performance on other datasets, including CIFAR-10, CIFAR-100, and SVHN. Similar to [18], we freeze BN layers for all the methods. Recall for CIFAR-10, CIFAR-100, and SVHN datasets, we use a subset of the classes as ID and the rest of the classes (from the same dataset) as OOD. The average accuracy of all compared methods v.s. OOD ratio is plotted in Fig 4 (c)-(d) for CIFAR-10. Our approach consistently outperforms existing baselines (and also the PL algorithm) by almost 4.5% when the OOD ratio is 75%. In addition, we compared our approach with the DS3L and UASD (SOTA robust SSL methods) on CIFAR100 and SVHN datasets and got a similar pattern. The results are shown in Table II and III. Furthermore, in cases where we can not apply clustering (CRW), we also show the performance of our method without CRW. The result in Table II and III shows

that shows that our approach still performs better than DS3L and UASD, which demonstrate that our approach can achieve stable performance.

TABLE II: SVHN-Extra with different OOD ratio.

OOD ratio	*25%*	*50%*	*75%*
VAT	94.1± 0.5	93.6± 0.7	92.8± 0.9
L2RW-VAT	96.0± 0.6	93.5± 0.8	92.7± 0.8
MWN-VAT	96.2± 0.5	93.8± 1.1	93.0± 1.3
UASD-VAT	96.3± 0.5	94.2± 0.9	93.3± 1.3
DS3L-VAT	96.4± 0.7	93.9± 1.0	92.9± 1.2
Ours w/o CRW	96.6± 0.6	94.4± 0.8	93.2± 1.1
Ours-VAT	**96.8± 0.7**	**95.2± 0.9**	**94.9± 1.3**

TABLE III: CIFAR100 with different OOD ratio.

OOD ratio	*25%*	*50%*	*75%*
UASD-MT	60.5± 0.6	60.3± 0.7	58.5± 1.0
DS3L-MT	60.8± 0.5	60.1± 1.1	57.2± 1.2
Ours w/o CRW	61.5± 0.4	60.7± 0.6	59.0± 0.8
Ours-MT	**62.1± 0.5**	**61.0± 0.5**	**59.7± 0.9**
UASD-PI	61.1± 0.5	60.0± 0.9	58.4± 1.0
DS3L-PI	60.5± 0.6	60.1± 1.0	57.4± 1.3
Ours w/o CRW	61.2± 0.4	60.4± 0.4	58.9± 0.6
Ours-PI	**61.6± 0.4**	**60.7± 0.5**	**59.5± 0.7**

B. Efficiency Analysis

To evaluate the efficiency of our proposed approach, we first compare the running time among all methods. Fig 5 (a)-(b) shows the running time (relative to the original SSL algorithm) for MNIST and CIFAR-10. We see that our proposed approaches are only $1.7\times$ to $1.8\times$ slower than the original SSL algorithm, while other robust SSL methods (L2RW and DS3L) are almost $3\times$ slower. We note that our implementation tricks can also be applied to these other techniques (DS3L, L2RW, MWN), but this would possibly degrade performance since these approaches' performance is worse than ours even without these tricks. To further analyze our proposed speedup strategies, we plot the running time v.s. accuracy in Fig 5 (d)-(e) for different settings (with/without last and with/without infrequent updates). As expected, the results show that, without the only last layer updates and the infrequent updates (i.e. if $L = 1$), Algorithm 2 is $3\times$ slower than SSL baseline. Whereas with the last layer updates, it is around $2\times$ slower. We get the best trade-off between speed and accuracy considering both $L = 5$ and the last layer updates. In addition, we analyze the efficiency of our approach with varying inner loop gradients steps (J) and inverse Hessian approximation (P). The result shows that the running time of our method would increase with J and P increase, we choose best trade-off between speed and accuracy considering $J = 3$ and $P = 5$.

C. Additional Analysis

Analysis of weight variation. Fig 7 (a) shows the weight learning curve of our approach and DS3L on Fashion-MNIST OOD. The results show that our method learns better weights for unlabeled samples compared to DS3L. The weight distribution learned in other cases are also similar.

Size of the clean validation set. We explore the sensitivity of the clean validation set used in robust SSL approaches on Mean MNIST OOD. Fig 7 (b) plots the classification performance with varying the size of the clean validation set. Surprisingly, our methods are stable even when using only 25 validation images, and the overall classification performance does not grow after having more than 1000 validation images.

Ablation Studies. We conducted additional experiments on Fashion-MNIST OOD (see Fig 7 (c)-(e)) in order to demonstrate the contributions of the key technical components, including Cluster Re-weight (CRW) and weighted Batch Normalization (WBN). The key findings obtained from this experiment are *1)* WBN plays a vital role in our weighted robust SSL framework to improve the robustness of BN against OOD data; *2)* Removing CRW (or L1 regularization) results in performance decrease, especially for VAT based approach, which demonstrates that CRW (and L1 regularization) can further improve performance for our robust-SSL approach; *3)* Fig 7 (d)-(e) demonstrate that the performance of our approach would increase with (inner loop gradients steps) and inverse Hessian approximation) increase due to high-order approximation. Based on this, we choose best trade-off between speed and accuracy considering $J = 3$ and $P = 5$ when considering running time analysis in Fig 5 (c).

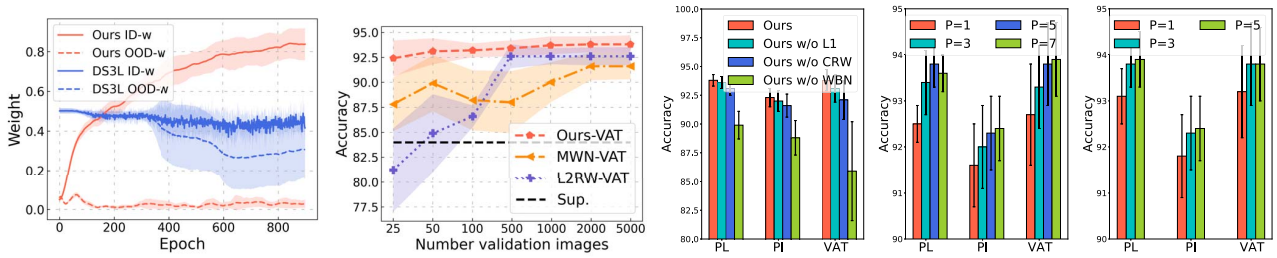
L1 vs CRW tricks. Next, we discuss the trade-offs between L1 and CRW. We first analyzed the sensitivity of our proposed CRW methods to the number of clusters used. Table IV demonstrates the test accuracies of our approach with varying numbers of clusters. The results indicate a low sensitivity of our proposed methods to the number of clusters. We also find that CRW generally can further improve the performance. Part of this success can be attributed to the good pretrained features. Our approach without CRW (only consider L1 regularization to reduce overfitting) performs well even without this additional information (Table II ,III, and Fig. 7 (c)).

TABLE IV: Test accuracies for different numbers of clusters K on the MNIST dataset with 50% Mean MNIST as OODs .

# Clusters	K=5	K=10	K=20	K =30
Ours-VAT	94.7± 0.7	95.3± 0.4	96.3± 0.5	95.6± 0.5
Ours-PL	95.2± 0.5	95.3± 0.5	96.2± 0.4	95.9± 0.5

VII. Conclusion

In this work, we first propose the research question: *How out-of-distribution data hurt semi-supervised learning performance?* To answer this question, we study the impact of OOD data on SSL algorithms and demonstrate empirically that the SSL algorithms' performance depends on how close the OOD instances are to the decision boundary (and the ID data instances). To address the above causes, we proposed a novel unified weighted robust SSL framework, which is designed to improve the robustness of BN against OODs. To address the limitation of low-order approximations in bi-level optimization (DS3L), we designed an implicit-differentiation based algorithm that considered high-order approximations of the objective and is scalable to a higher number of inner optimization steps to learn a massive amount of weight parameters. In addition, we conduct a theoretical analysis about the impact of faraway OODs in the BN step and discuss the connection between our approach (high-order approximation based on implicit differentiation) and low-order approximation



(a) Weight variation curves. (b) Effect of validation set. (c) Ablation Study. (d) Ablation Study (e) Ablation Study

Fig. 7: (a) shows that our method learns optimal weights for ID and OOD samples; (b) shows that our method is stable even for small validation set containing 25 images; (c) shows that WBN and CRW (or L1 regularization) are critical in retaining the performance gains of reweighting; (d)-(e) demonstrate that the performance of our approach would increase with (inner loop gradients steps) and inverse Hessian approximation) increase due to high-order approximation.

approaches. We show that our weighted robust SSL approach significantly outperforms existing robust approaches (L2RW, MWN, Safe-SSL, and UASD) on several real-world datasets.

Acknowledgments

The work of Xujiang Zhao and Feng Chen was supported by the National Science Foundation (NSF) under grant number 2147375, 2107449, and 1954376. Rishabh Iyer and Killamsetty Krishnateja would like to acknowledge support from NSF Grant Number IIS-2106937, a gift from Google Research, and the Adobe Data Science Research award.

References

[1] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," *arXiv*, 2019.

[2] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, "Deep batch active learning by diverse, uncertain gradient lower bounds," *arXiv preprint arXiv:1906.03671*, 2019.

[3] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *NeurIPS*, 2019.

[4] K. Chen, L. Yao, D. Zhang, X. Chang, G. Long, and S. Wang, "Distributionally robust semi-supervised learning for people-centric sensing," in *AAAI*, 2019.

[5] Y. Chen, X. Zhu, W. Li, and S. Gong, "Semi-supervised learning under class distribution mismatch." in *AAAI*, 2020.

[6] X. Dong, J. Guo, A. Li, W.-T. Ting, C. Liu, and H. Kung, "Neural mean discrepancy for efficient out-of-distribution detection," in *CVPR*, 2022, pp. 19 217–19 227.

[7] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *NIPS*, 2005.

[8] H. Guo, Y. Mao, and R. Zhang, "Mixup as locally linear out-of-manifold regularization," in *AAAI*, 2019.

[9] L.-Z. Guo, Z.-Y. Zhang, Y. Jiang, Y.-F. Li, and Z.-H. Zhou, "Safe deep semi-supervised learning for unseen-class unlabeled data," in *ICML*, 2020.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[12] K. Killamsetty, S. Durga, G. Ramakrishnan, A. De, and R. Iyer, "Grad-match: Gradient matching based data subset selection for efficient deep model training," in *ICML*. PMLR, 2021.

[13] K. Killamsetty, X. Zhao, F. Chen, and R. Iyer, "Retrieve: Coreset selection for efficient and robust semi-supervised learning," *NeurIPS*, 2021.

[14] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv*, 2016.

[15] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, 2013.

[16] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing millions of hyperparameters by implicit differentiation," in *AISTATS*, 2020.

[17] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *TPAMI*, 2018.

[18] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," in *NeurIPS*, 2018.

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

[20] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *NeurIPS*, 2016.

[21] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," in *NeurIPS*, 2019.

[22] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *arXiv preprint arXiv:2001.07685*, 2020.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR*, 2014.

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

[25] Z. Wang, Y. Chen, C. Zhao, and etc., "Clear: Contrastive-prototype learning with drift estimation for resource constrained stream mining," in *The Web Conference*, 2021, pp. 1351–1362.

[26] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv*, 2017.

[27] Q. Xie, Z. Dai, E. Hovy, and etc., "Unsupervised data augmentation for consistency training," *arXiv*, 2019.

[28] L. Xu, X. Zhang, X. Zhao, H. Chen, F. Chen, and J. D. Choi, "Boosting cross-lingual transfer via self-learning with uncertainty estimation," *arXiv preprint arXiv:2109.00194*, 2021.

[29] Y. Yan, Z. Xu, I. W. Tsang, and etc., "Robust semi-supervised learning through label aggregation," in *AAAI*, 2016.

[30] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv*, 2017.

[31] X. Zhao, X. Zhang, W. Cheng, W. Yu, Y. Chen, H. Chen, and F. Chen, "Seed: Sound event early detection via evidential uncertainty," in *ICASSP*. IEEE, 2022, pp. 3618–3622.