META-SEMI: A META-LEARNING APPROACH FOR SEMI-SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep learning based semi-supervised learning (SSL) algorithms have led to promising results in recent years. However, they tend to introduce multiple tunable hyper-parameters, making them less practical in real SSL scenarios where the labeled data is scarce for extensive hyper-parameter search. In this paper, we propose a novel meta-learning based SSL algorithm (*Meta-Semi*) that requires tuning only one additional hyper-parameter, compared with a standard supervised deep learning algorithm, to achieve competitive performance under various conditions of SSL. We start by defining a meta optimization problem that minimizes the loss on labeled data through dynamically reweighting the loss on unlabeled samples, which are associated with soft pseudo labels during training. As the meta problem is computationally intensive to solve directly, we propose an efficient algorithm to dynamically obtain the approximate solutions. We show theoretically that *Meta-Semi* converges to the stationary point of the loss function on labeled data under mild conditions. Empirically, *Meta-Semi* outperforms state-of-the-art SSL algorithms significantly on the challenging semi-supervised CIFAR-100 and STL-10 tasks, and achieves competitive performance on CIFAR-10 and SVHN.

1 INTRODUCTION

The recent success of deep learning in supervised tasks is fueled by abundant annotated training data (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Szegedy et al., 2015; LeCun et al., 2015; He et al., 2016; Huang et al., 2019). However, collecting precise labels in practice is usually very time-consuming and costly. In many real-world applications, only a small subset of all available training data are associated with labels (Oliver et al., 2018; Verma et al., 2019). Semi-supervised learning (SSL) is a learning paradigm that aims to improve model performances by simultaneously leveraging labeled and unlabeled data (Zhu et al., 2003; Chapelle et al., 2006; Turian et al., 2010).

In the context of deep learning, many successful SSL methods incorporate unlabeled data by performing unsupervised consistency regularization (Laine & Aila, 2016; Tarvainen & Valpola, 2017; Miyato et al., 2018; Verma et al., 2019; Berthelot et al., 2019). In specific, they first add small perturbations to the unlabeled samples, and then enforce the consistency between the model predictions on the original data and the perturbed data. Though impressive performance has been achieved, the state-of-the-art consistency based algorithms tend to introduce multiple tunable hyper-parameters. The final performance of the algorithms is usually conditioned on setting proper values for these hyper-parameters. However, in real semi-supervised learning scenarios, hyper-parameter searching is usually unreliable as the annotated data are scarce, leading to high variance when cross-validation is adopted (Oliver et al., 2018). This problem will become even more serious if the performance of the algorithm is sensitive to the hyper-parameter values. Furthermore, since the searching space grows exponentially with respect to the number of hyper-parameters (Bergstra & Bengio, 2012), the computational cost may become unaffordable for modern deep learning algorithms.

Another challenge to develop practical and robust deep SSL algorithms is how to exploit the *labeled data* more efficiently, as these data, although being scarce, have the precise and reliable annotations. Consistency based SSL algorithms (Laine & Aila, 2016; Tarvainen & Valpola, 2017; Miyato et al., 2018; Verma et al., 2019; Berthelot et al., 2019) usually model the labeled and unlabeled data in separate terms in the loss function, where the unlabeled data receives no supervision, at least explicitly, from the former, leading to an inefficient use of the labeled data.

In this paper, we propose a meta-learning based SSL algorithm, named *Meta-Semi*, to efficiently exploit the labeled data, while requiring tuning only one additional hyper-parameter to achieve impressive performance under various conditions. The proposed algorithm is based on a simple intuition: *if the network is trained with correctly "pseudo-labeled" unannotated samples, the final loss on labeled data should be minimized*. To be specific, we start by explicitly defining a meta reweighting objective: finding the optimal weights[1] for different pseudo-labeled samples to train a network, such that the final loss on labeled data is minimized. Note that the problem is computationally intensive to be directly solved via optimization algorithms. Therefore, we propose an approximated formulation, based on which a closed form solution can be obtained. We show theoretically that one meta gradient step is sufficient to obtain the approximate solutions at each iteration. Finally, we propose a dynamical weighting algorithm to reweight pseudo-labeled samples with 0-1 weights. Theoretical analysis shows that our method converges to the stationary point of the supervised loss function.

Our algorithm is empirically validated on widely used image classification benchmarks (CIFAR-10, CIFAR-100, SVHN and STL-10) with modern deep networks (e.g., CNN-13 and WRN-28). *Meta-Semi* outperforms state-of-the-art SSL algorithms, including ICT (Verma et al., 2019) and MixMatch (Berthelot et al., 2019), on the challenging CIFAR-100 and STL-10 SSL tasks significantly, while achieves slightly better performance than them on CIFAR-10. Besides, *Meta-Semi* is complementary to consistency based methods, i.e., performing consistency regularization in our algorithm further improves the performance. Moreover, sensitivity test on the only tunable hyper-parameter of *Meta-Semi* shows that the algorithm is quite robust to different hyper-parameter values.

2 Related Work

Consistency based semi-supervised learning has been extensively studied in the context of deep learning in recent years (Sajjadi et al., 2016; Laine & Aila, 2016; Tarvainen & Valpola, 2017; Miyato et al., 2018; Verma et al., 2019). These methods leverage unlabeled data by adding an unsupervised regularization term to the standard supervised loss: $\mathcal{L}_S + w\mathcal{L}_{US}$, where $\mathcal{L}_S$ is the conventional loss on labeled data, $\mathcal{L}_{US}$ is the loss contributed by unlabeled data which is usually defined as a measure of discrepancy between the model predictions on the original unlabeled samples and their perturbed counterparts, and w is a pre-defined coefficient. Existing approaches have proposed different ways to generate the perturbations for $\mathcal{L}_{US}$, including data augmentation (Bachman et al., 2014; Laine & Aila, 2016; Sajjadi et al., 2016), adversarial noise (Miyato et al., 2018), Dropout (Park et al., 2018), data interpolation (Verma et al., 2019), etc. To enhance the model stability, an exponential moving average (EMA) on parameters or predictions is often adopted (Laine & Aila, 2016; Tarvainen & Valpola, 2017). The effectiveness of these approaches is conditioned on the proper setting of the coefficient w. As recent methods (Berthelot et al., 2019; 2020) usually integrate multiple regularization techniques, finding the proper hyperparameter setting becomes a challenging problem in practice, especially in SSL scenarios where few samples are available for cross-validation.

Other semi-supervised learning algorithms. Early work on SSL can be categorized into cluster assumption based methods (Joachims, 2003; 1999) and graph assumption based methods (Zhu et al., 2003; Bengio et al., 2006). For deep learning based SSL, (Kingma et al., 2014; Odena, 2016) propose to train deep generators using both the labeled and unlabeled data to estimate the data distribution. Pseudo label based method (Lee, 2013) is also widely used in deep SSL. It progressively uses the highly confident model predictions to generate pseudo labels for unlabeled samples during training. Minimizing the entropy of the model prediction on unlabeled data is also proven effective for SSL (Grandvalet & Bengio, 2005; Miyato et al., 2018).

Meta learning. Since *Meta-Semi* follows a meta-learning paradigm, we briefly review the existing work on this topic. The idea of meta-learning is motivated by the goal of 'learning to learn better' (Lake et al., 2017; Andrychowicz et al., 2016). Meta-learning algorithms usually define a meta optimization problem to extract information from the learning process. For example, using the loss on a small amount of trustable data as the meta-objective is widely adopted in few-shot learning (Ravi & Larochelle, 2017; Ren et al., 2018a). MAML (Finn et al., 2017) proposes to minimize the meta loss directly via gradient descents. To address the challenge that naively minimizing the meta objective requires performing multiple meta update steps iteratively for every 'real' update step on model parameters, (Ren et al., 2018b) propose an online approximation method to make the meta training process more tractable. The proposed algorithm is similar to that in (Ren et al., 2018b), but

[1]Throughout the paper, the term "weights" always refer to the coefficients that we use to reweight each individual unlabeled sample, instead of referring to the parameters of neural networks.

our contributions lie in several important aspects. First, we exploit the labeled data more efficiently in SSL by leveraging the meta-reweighting method, which not only reduces the required number of tunable hyper-parameters, but also effectively improves the performance. As far as we know, this idea has not been explored in the literature. Second, we propose a novel dynamical re-weighting process that is tailored for SSL. This is non-trivial since directly applying the method in (Ren et al., 2018b) to SSL leads to inferior results (see: Table 1). Third, we provide a theoretical convergence analysis in the context of SSL, which utilizes different proof techniques from (Ren et al., 2018b).

3 METHOD

In this section, we introduce the details of our *Meta-Semi* algorithm. Different from most existing methods that leverage unsupervised consistency regularization, we propose to solve the SSL problem in a meta-learning paradigm. As an overview, we first compute the cross-entropy loss of unlabeled samples using their corresponding pseudo labels. Then we reweight the loss on each unlabeled sample by solving a meta optimization problem that minimizes the supervised loss of labeled samples. As directly solving the meta problem is computationally intractable, we propose an approximation method to dynamically obtain the 0-1 approximate solutions, which only requires one meta gradient descent step. In addition, theoretical guarantees are provided to show that our method converges to the stationary point of the supervised loss.

3.1 META OPTIMIZATION PROBLEM

We start by presenting the weighted loss function of our method, and defining a meta optimization problem to determine the value of the weight for each unlabeled sample.

Suppose that the networks are trained with stochastic gradient descent (SGD). At each iteration, we sample a mini-batch of labeled samples $\mathcal{X} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ together with a mini-batch of unlabeled samples $\mathcal{U} = \{(\boldsymbol{u}_j, \hat{\boldsymbol{y}}_j)\}$, where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ represent the i^{th} labeled sample and its associated ground truth label, respectively, and $\boldsymbol{u}_j$ and $\hat{\boldsymbol{y}}_j$ represent the j^{th} unlabeled sample and its pseudo label, respectively. Following earlier work (Verma et al., 2019; Berthelot et al., 2019), we use the MixUp augmentation (Zhang et al., 2018) to generate a mixed version of the inputs to improve the generalization performance, instead of directly using $\mathcal{X}$ and $\mathcal{U}$. The augmented mini-batch of training samples are denoted by $\tilde{\mathcal{X}} = \{(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i)\}$ and $\tilde{\mathcal{U}} = \{(\tilde{\boldsymbol{u}}_j, \hat{\boldsymbol{y}}_j)\}$. We defer the details on generating pseudo labels and obtaining $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{U}}$ to Section 3.3.

Consider training a deep network with parameters $\boldsymbol{\theta}$. We first feed an unlabeled sample $\tilde{\boldsymbol{u}}_j$ into the network, producing its prediction $p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta})$. Then we calculate the cross-entropy loss $L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}))$ using the corresponding soft pseudo label $\hat{\boldsymbol{y}}_j$. The loss of this sample is further reweighed by $w_j^* \in [0, 1]$ to construct the final loss function

$$\mathcal{L}_{meta} = \frac{1}{\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^*} \sum\nolimits_{j=1}^{|\tilde{\mathcal{U}}|} w_j^* L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta})). \tag{1}$$

Without loss of generality, we assume $\mathcal{L}_{meta} = 0$ when $\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^* = 0$. The weight scalar w_j^* is determined by minimizing the meta loss on the labeled data. To illustrate that, we first consider training the network with a similar weighted loss

$$\boldsymbol{\theta}^*(\boldsymbol{w}) = \arg\min_{\boldsymbol{\theta}} \sum\nolimits_{j=1}^{|\tilde{\mathcal{U}}|} w_j L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta})), \tag{2}$$

where $\boldsymbol{\theta}^*(\boldsymbol{w})$ is the optimal solution that minimizes the weighted loss. Obviously, it is a function of the weight vector $\boldsymbol{w} = [w_1, w_2, \ldots]^T$. Then the weights $\boldsymbol{w}^*$ is solved by minimizing the loss on labeled data $\tilde{\mathcal{X}}$ with $\boldsymbol{\theta}^*(\boldsymbol{w})$, namely

$$\boldsymbol{w}^* = \arg\min_{w_j \in [0,1], j=1,\ldots,|\tilde{\mathcal{U}}|} \sum\nolimits_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^*(\boldsymbol{w}))). \tag{3}$$

Intuitively, our aim is to find a subset of pseudo-labeled samples, which, if used for training, are the most beneficial in terms of the generalization performance. The labeled data are leveraged to determine if each pseudo-labeled sample should be used, instead of directly being used for train as most existing SSL algorithms do (Laine & Aila, 2016; Tarvainen & Valpola, 2017; Miyato et al., 2018; Verma et al., 2019; Berthelot et al., 2019). We argue that this is a more effective approach to exploit the supervision information.

3.2 APPROXIMATING THE META SOLUTION

To solve the meta optimization problem Eqs. (2) and (3) efficiently, we introduce a method to obtain an approximate solution. At t^{th} step in the training process, consider estimating $\boldsymbol{\theta}^*(\boldsymbol{w})$ by

performing M times of gradient descents starting from current values of network parameters $\boldsymbol{\theta}^t$:

$$\overline{\boldsymbol{\theta}}_M^t \approx \boldsymbol{\theta}^*(\boldsymbol{w}), \ \ \overline{\boldsymbol{\theta}}_0^t = \boldsymbol{\theta}^t, \tag{4}$$

$$\overline{\boldsymbol{\theta}}_{m+1}^t = \overline{\boldsymbol{\theta}}_m^t - \alpha^t \left[\frac{\partial \sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\overline{\boldsymbol{\theta}}_m^t))}{\partial \overline{\boldsymbol{\theta}}_m^t} \right], m = 0, 1, \ldots, M-1, \tag{5}$$

where α^t is the learning rate. As SGD has proven to be effective for optimizing deep networks, $\overline{\boldsymbol{\theta}}_M^t$ is a reliable alternate of $\boldsymbol{\theta}^*(\boldsymbol{w})$ as long as M is sufficiently large.

Given that $\boldsymbol{\theta}^*(\boldsymbol{w})$ can be estimated by $\overline{\boldsymbol{\theta}}_M^t$, a naive method of approximating $\boldsymbol{w}^*$ is to further estimate the gradient $\nabla_{\boldsymbol{w}} \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^*(\boldsymbol{w})))$ with $\overline{\boldsymbol{\theta}}_M^t$, and then repeatedly update $\boldsymbol{w}$ following similar gradient based optimization algorithms. However, it is computationally intensive to do that since updating $\boldsymbol{w}$ for N times requires MN steps of gradient descents on the network parameters. To get a efficient estimate of $\boldsymbol{w}^*$, we propose a dynamic approximation approach in the following.

First, to reduce the iterations of updating $\boldsymbol{w}$, we exploit a first order Taylor approximation of Eq. (3) at $\boldsymbol{w} = 0$:

$$\boldsymbol{w}^* \approx \underset{w_j \in [0,1], j=1,\ldots,|\tilde{\mathcal{U}}|}{\arg\min} \boldsymbol{w}^T \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial \boldsymbol{w}} \bigg|_{\boldsymbol{w}=0} \right]. \tag{6}$$

Notably, $\overline{\boldsymbol{\theta}}_M^t$ is obtained using the gradients of the weighted loss according to Eq. (5), and thus it is differentiable with respect to w_j. As the optimization objective in Eq. (6) is linear, it is straightforward to derive the solution:

$$w_j^* \approx w_j^t = \begin{cases} 1 & \frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial w_j} \Big|_{\boldsymbol{w}=0} \leq 0 \\ 0 & \frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial w_j} \Big|_{\boldsymbol{w}=0} > 0 \end{cases}, \tag{7}$$

where w_j^t denotes the approximate solution of w_j^*. The required steps of gradient descents are reduced to M from MN by leveraging Eq. (7). However, the algorithm is still inefficient since a large M is necessary to get a sufficiently accurate $\overline{\boldsymbol{\theta}}_M^t$. To further reduce the computational cost, an intriguing property can be leveraged. In the following proposition, we show that the results of Eq. (7) will remain the same if $\overline{\boldsymbol{\theta}}_M^t$ is replaced by $\overline{\boldsymbol{\theta}}_1^t$. In other words, Eq. (7) can be precisely solved using $\overline{\boldsymbol{\theta}}_1^t$ instead of $\overline{\boldsymbol{\theta}}_M^t$, and the former only needs one gradient descent step to obtain.

Proposition 1. *Suppose that $\overline{\boldsymbol{\theta}}_M^t$ is given by M steps of gradient descents starting from $\overline{\boldsymbol{\theta}}_0^t = \boldsymbol{\theta}^t$. Then we have*

$$\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial w_j} \bigg|_{\boldsymbol{w}=0} = M \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t))}{\partial w_j} \bigg|_{\boldsymbol{w}=0} \right], \ \forall \, 1 \leq j \leq |\tilde{\mathcal{U}}|. \tag{8}$$

Proof. See Appendix A. □

With Proposition 1, we are ready to present the final form of our dynamically reweighting formula:

$$w_j^t = \begin{cases} 1 & \frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t))}{\partial w_j} \Big|_{\boldsymbol{w}=0} \leq 0 \\ 0 & \frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t))}{\partial w_j} \Big|_{\boldsymbol{w}=0} > 0 \end{cases}. \tag{9}$$

As we leverage a meta learning approach to reweight different pseudo-labeled samples, we call our method *Meta-Semi*. The pseudo code of *Meta-Semi* is presented in Algorithm 1. In summary, after each standard forward step of the pseudo-labeled samples, we first update the parameters with the loss of all samples weighted by zero. Such a meta updating step does not change the values of parameters, but construct a differentiable computational graph. Then we calculate the supervised loss on labeled data, and exploit the computational graph to take the derivative of the supervised loss with respect to the zero weight, which is called "meta gradient". Finally, we only use the pseudo-labeled samples with negative meta gradients to train the network.

Interpretation of meta gradients. A straightforward way to interpret the meta gradients is that it can be viewed as the influence on the supervised loss when the weight of certain pseudo-labeled

Algorithm 1 The Meta-Semi Algorithm.

Initialize: $\boldsymbol{\theta}^0$
for $t = 1$ **to** T **do**
  Randomly sample $\mathcal{X}, \mathcal{U}$
  Generate $\tilde{\mathcal{X}}, \tilde{\mathcal{U}}$
  Compute $p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t)$, $\tilde{\boldsymbol{u}}_j \in \tilde{\mathcal{U}}$
  $\boldsymbol{w} \leftarrow 0$, $\overline{\boldsymbol{\theta}}_0^t \leftarrow \boldsymbol{\theta}^t$
  $\nabla_{\overline{\boldsymbol{\theta}}_0^t} \leftarrow \frac{\partial \sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}$
  $\overline{\boldsymbol{\theta}}_1^t \leftarrow \overline{\boldsymbol{\theta}}_0^t - \alpha^t \nabla_{\overline{\boldsymbol{\theta}}_0^t}$
  Compute $p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t)$, $\tilde{\boldsymbol{x}}_i \in \tilde{\mathcal{X}}$
  Meta Gradient: $\nabla_{\boldsymbol{w}}^t \leftarrow \frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t))}{\partial \boldsymbol{w}}$
  $\boldsymbol{w}^t \leftarrow sign(max(-\nabla_{\boldsymbol{w}}^t, 0))$ (Eq. (9))
  $\mathcal{L}_{meta} \leftarrow \frac{1}{\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t} \sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))$
  $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^t - \alpha^t \frac{\partial \mathcal{L}_{meta}}{\partial \boldsymbol{\theta}^t}$
end for

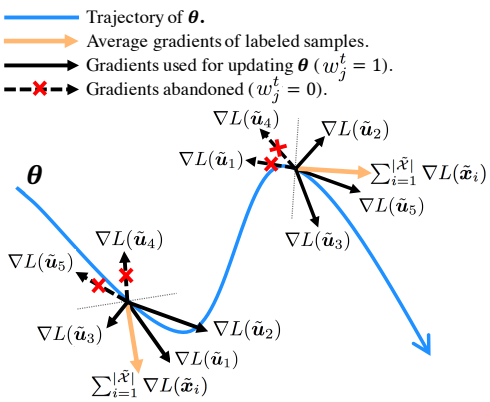

Figure 1: Illustration of *Meta-Semi*. Herein, $\nabla L(\tilde{\boldsymbol{u}}_j)$ and $\nabla L(\tilde{\boldsymbol{x}}_i)$ denote $\nabla_{\boldsymbol{\theta}^t} L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))$ and $\nabla_{\boldsymbol{\theta}^t} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t))$, respectively. Our method trains the networks using pseudo-labeled samples with similar gradient directions to labeled samples.

sample changes slightly around zero during training. In fact, there exists a more intriguing and interesting interpretation. The meta gradients given in Eq. (9) can be expressed as

$$\begin{aligned}\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t))}{\partial w_j}\bigg|_{\boldsymbol{w}=0} &= \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t))}{\partial \overline{\boldsymbol{\theta}}_1^t}\right]^T \left[\frac{\partial (\overline{\boldsymbol{\theta}}_0^t - \alpha^t \nabla_{\overline{\boldsymbol{\theta}}_0^t})}{\partial w_j}\right]\bigg|_{\boldsymbol{w}=0} \\ &= -\alpha^t \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right]^T \left[\frac{\partial L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right],\end{aligned} \tag{10}$$

which follows from $\nabla_{\overline{\boldsymbol{\theta}}_0^t} = \sum_{k=1}^{|\tilde{\mathcal{U}}|} w_k \frac{\partial L(\tilde{\boldsymbol{y}}_k, p(\tilde{\boldsymbol{u}}_k|\overline{\boldsymbol{\theta}}_0^t))}{\partial \overline{\boldsymbol{\theta}}_0^t}$ and $\overline{\boldsymbol{\theta}}_1^t = \overline{\boldsymbol{\theta}}_0^t = \boldsymbol{\theta}^t$. For the pseudo-unlabeled sample $(\tilde{\boldsymbol{u}}_j, \hat{\boldsymbol{y}}_j)$, its meta gradient is negatively proportional to the inner product of the average gradient of labeled samples and the gradient produced by itself. In other words, the sign of the meta gradient indicates whether the angle between the former and the later is larger than 90 degrees. Intuitively, if the pseudo label is correct, the corresponding gradient should guide the model towards a similar direction to the labeled samples, or at least should not be largely different from the supervised gradient in direction. In essence, *Meta-Semi* trains networks using pseudo-labeled samples whose gradient directions are similar to labeled samples. An illustration is shown in Figure 1.

3.3 Implementation Details

Pseudo labels. To obtain high quality pseudo labels for the original unlabeled mini-batch $\mathcal{U}$, we first apply an exponential moving average (EMA) on model parameters, which has proven to be effective in providing supervision on unlabeled data (Tarvainen & Valpola, 2017; Verma et al., 2019). Then we feed every unlabeled sample $\boldsymbol{u}_j$ in $\mathcal{U}$ into the EMA model, and take the corresponding softmax prediction as the soft pseudo label $\hat{\boldsymbol{y}}_j$.

MixUp augmentation is an important regularization technique used by state-of-the-art deep SSL algorithms (Verma et al., 2019; Berthelot et al., 2019). It improves the generalization performance of models by encouraging the 'convex' behavior between different samples. Given a pair of samples with corresponding annotations $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $(\boldsymbol{x}_2, \boldsymbol{y}_2)$, MixUp is performed to generate an augmented sample via linear interpolation:

$$\tilde{\boldsymbol{x}} = \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2, \quad \tilde{\boldsymbol{y}} = \lambda \boldsymbol{y}_1 + (1-\lambda)\boldsymbol{y}_2, \tag{11}$$

where λ is sampled from a pre-defined Beta distribution. In *Meta-Semi*, we leverage MixUp to generate the mixed training data $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{U}}$. Formally, $\tilde{\mathcal{X}}$ is obtained from only the labeled set $\mathcal{X}$:

$$\tilde{\mathcal{X}} = \text{MixUp}(\mathcal{X}, \text{Shuffle}(\mathcal{X}), \lambda_1), \quad \lambda_1 \sim \text{Beta}(\beta, \beta), \tag{12}$$

where β is the parameter of the Beta distribution, and it is the only tunable hyper-parameter (excluding the hyper-parameters of a supervised learning algorithm) in our algorithm. With regards to $\tilde{\mathcal{U}}$,

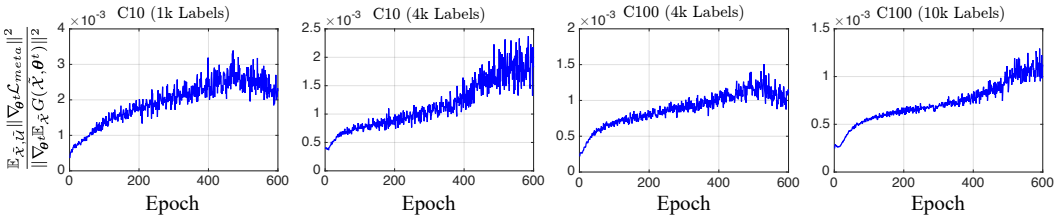

Figure 2: Empirical validation of Assumption 1. The value of $\frac{\mathbb{E}_{\tilde{\mathcal{X}},\tilde{\mathcal{U}}}\|\nabla_{\boldsymbol{\theta}^t}\mathcal{L}_{meta}\|^2}{\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|^2}$ is estimated at each training epoch using Monte-Carlo sampling with a sample size 500. Results on CIFAR-10 (C10) and CIFAR-100 (C100) with varying numbers of labeled samples are presented. It can be observed that the ratio generally increases before the 500^{th} epoch, but gradually becomes stable or even decreases in the last part of the training process when the learning rate approaches 0. Therefore, it is reasonable to assume that Assumption 1 holds.

we ideally want the unlabeled data to extract more information from the labeled samples. Therefore, we first concatenate $\mathcal{X}$ and $\mathcal{U}$ together, and then apply the MixUp procedure:

$$\tilde{\mathcal{U}} = \text{MixUp}(\mathcal{W}, \text{Shuffle}(\mathcal{W}), \lambda_2), \quad \mathcal{W} = \text{Concat}(\mathcal{X},\mathcal{U}), \quad \lambda_2 \sim \text{Beta}(\beta,\beta), \tag{13}$$

where the one-hot ground truth labels are used for $\mathcal{X}$ and the soft pseudo labels are used for $\mathcal{U}$.

Compatibility with consistency based methods. As a matter of fact, *Meta-Semi* is compatible with existing consistency based algorithms, and they can be integrated when necessary. To see this, the regularization term can be simply appended to the loss function with an addition coefficient w:

$$\mathcal{L} = \mathcal{L}_{meta} + w\mathcal{L}_{consistency}. \tag{14}$$

In experiments, we show that although *Meta-Semi* has already achieved state-of-the-art performance, its performance is still able to be significantly improved by integrating consistency regularization.

3.4 Convergence Analysis

In this section, we show theoretically that under some mild conditions, our method converges to the stationary point of the loss on labeled data. To make it clear, we first define the supervised loss on the labeled mini-batch $\tilde{\mathcal{X}}$:

$$G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t) = \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t)). \tag{15}$$

Thus, the expected loss on all the labeled data is $\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)$. Then we introduce the definition of Lipschitz-smooth and a mild assumption stating that the expected norm of gradients used for updating model parameters will not get too large compared with the gradient of the supervised loss.

Definition 1. *A function $f:\mathbb{R}^n\to\mathbb{R}$ is said to be Lipschitz-smooth with constant L if*

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x-y\|, \ \ \forall x, y \in \mathbb{R}^n.$$

Assumption 1. *For all $t \ge 0$, there exists a positive scalar σ, such that*

$$\mathbb{E}_{\tilde{\mathcal{X}},\tilde{\mathcal{U}}}\|\nabla_{\boldsymbol{\theta}^t}\mathcal{L}_{meta}\|^2 \le \sigma\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)\|^2.$$

In fact, the assumption is not very strong. Roughly, since $\mathcal{L}_{meta}$ is computed using the ground truth labels and the pseudo labels based on the prediction of the EMA model, it is usually very close to the minima of the loss function, especially when the networks tend to be stable with sufficiently large t. Empirically, we show that Assumption 1 holds in many cases of SSL, which is shown in Figure 2. Under this condition, the following proposition shows that our method converges to the stationary point of the loss on labeled data with proper learning rate schedules.

Proposition 2. *Assume that the loss function on labeled data $G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)$ is Lipschitz-smooth with regards to $\boldsymbol{\theta}^t$ for all $\tilde{\mathcal{X}}$, and that Assumption 1 holds. Suppose also that the learning rate $\alpha^t > 0$ satisfies:*

$$\lim_{t\to\infty} \alpha^t = 0, \quad \sum_{t=0}^{\infty} \alpha^t = \infty. \tag{16}$$

Then every limit point of the sequence $\{\boldsymbol{\theta}^t\}$ generated by Meta-Semi is a stationary point of $\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)$, namely,

$$\lim_{t\to\infty} \|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)\| = 0.$$

Proof. See Appendix B. □

Table 1: Performance of *Meta-Semi* and state-of-the-art SSL algorithms with the CNN-13 network. We report the average test errors and the standard deviations of 5 trials. In each setting, the best two results are **bold-faced**.

Dataset	CIFAR-10			CIFAR-100	
Number of Labeled Samples	1000	2000	4000	4000	10000
Supervised	$39.95 \pm 0.75\%$	$27.67 \pm 0.12\%$	$20.42 \pm 0.21\%$	$58.31 \pm 0.89\%$	$44.56 \pm 0.30\%$
Supervised + MixUp (Zhang et al., 2018)	$31.83 \pm 0.65\%$	$24.22 \pm 0.15\%$	$17.37 \pm 0.35\%$	$54.87 \pm 0.07\%$	$40.97 \pm 0.47\%$
Π-model (Laine & Aila, 2016)	$28.74 \pm 0.48\%$	$17.57 \pm 0.44\%$	$12.36 \pm 0.17\%$	$55.39 \pm 0.55\%$	$38.06 \pm 0.37\%$
Temp-ensemble (Laine & Aila, 2016)	$25.15 \pm 1.46\%$	$15.78 \pm 0.44\%$	$11.90 \pm 0.25\%$	-	$38.65 \pm 0.51\%$
Mean Teacher (Tarvainen & Valpola, 2017)	$18.27 \pm 0.53\%$	$13.45 \pm 0.30\%$	$10.73 \pm 0.14\%$	$45.36 \pm 0.49\%$	$35.96 \pm 0.77\%$
VAT (Miyato et al., 2018)	$18.12 \pm 0.82\%$	$13.93 \pm 0.33\%$	$11.10 \pm 0.24\%$	-	-
SNTG (Luo et al., 2018)	$18.41 \pm 0.52\%$	$13.64 \pm 0.32\%$	$10.93 \pm 0.14\%$	-	$37.97 \pm 0.29\%$
Learning to Reweight (Ren et al., 2018b)	$11.74 \pm 0.12\%$	-	$9.44 \pm 0.17\%$	$46.62 \pm 0.29\%$	$37.31 \pm 0.47\%$
MT + Fast SWA (Athiwaratkun et al., 2019)	15.58%	11.02%	9.05%	-	$33.62 \pm 0.54\%$
ICT (Verma et al., 2019)	$12.44 \pm 0.57\%$	$8.69 \pm 0.15\%$	$7.18 \pm 0.24\%$	$40.07 \pm 0.38\%$	$32.24 \pm 0.16\%$
Meta-Semi	$\mathbf{10.27 \pm 0.66\%}$	$\mathbf{8.42 \pm 0.30\%}$	$\mathbf{7.05 \pm 0.27\%}$	$\mathbf{37.61 \pm 0.56\%}$	$\mathbf{30.51 \pm 0.32\%}$
Meta-Semi + ICT	$\mathbf{9.29 \pm 0.62\%}$	$\mathbf{7.05 \pm 0.12\%}$	$\mathbf{6.42 \pm 0.18\%}$	$\mathbf{37.12 \pm 0.59\%}$	$\mathbf{29.68 \pm 0.05\%}$

Table 2: Performance of *Meta-Semi* with the WRN-28 network. Average test errors and standard deviations of 5 trials are reported. "#HPs" refers to the number of tunable hyper-parameters. The best results are **bold-faced**.

Methods	#HPs	CIFAR-10					CIFAR-100
		250 Labels	500 Labels	1000 Labels	2000 Labels	4000 Labels	10000 Labels
Mean Teacher (Tarvainen & Valpola, 2017)	2	$47.32 \pm 4.71\%$	$42.01 \pm 5.86\%$	$17.32 \pm 4.00\%$	$12.17 \pm 0.22\%$	$10.36 \pm 0.25\%$	-
VAT (Miyato et al., 2018)	2	$36.03 \pm 2.82\%$	$26.11 \pm 1.52\%$	$18.68 \pm 0.40\%$	$14.40 \pm 0.15\%$	11.05 v 0.31%	-
MixMatch (Berthelot et al., 2019)	4	$11.08 \pm 0.87\%$	$9.65 \pm 0.94\%$	$7.75 \pm 0.32\%$	$7.03 \pm 0.15\%$	$6.24 \pm 0.06\%$	$30.84 \pm 0.29\%$
Meta-Semi	**1**	$\mathbf{9.40 \pm 0.58\%}$	$\mathbf{8.18 \pm 0.43\%}$	$\mathbf{7.34 \pm 0.22\%}$	$\mathbf{6.58 \pm 0.07\%}$	$\mathbf{6.10 \pm 0.10\%}$	$\mathbf{29.69 \pm 0.18\%}$

4 EXPERIMENTS

In this section, we empirically evaluate the effectiveness of the proposed *Meta-Semi* method, analyze its time complexity experimentally, and give sensitivity tests. The ablation studies are deferred to Appendix C. All experiments are conducted using a single Nvidia Titan Xp GPU.

Table 3: Test errors on STL-10. We adopt the same experimental setups as (Berthelot et al., 2019). The best result is **bold-faced**.

Method	STL-10, 1000 labels
SWWAE (Zhao et al., 2015)	25.70%
CC-GAN (Denton et al., 2016)	22.20%
MixMatch (Berthelot et al., 2019)	$10.18 \pm 1.46\%$
Meta-Semi	$\mathbf{8.03 \pm 0.24\%}$

4.1 EXPERIMENTAL SETUP

Our experiments are based on four widely used image classification benchmarks, i.e., CIFAR-10/100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011) and STL-10 (Coates et al., 2011), and two modern deep networks, i.e., a 13-layer CNN (CNN-13) and the Wide-ResNet-28 (WRN-28). On CIFAR and SVHN, we randomly preserve the labels of certain numbers of samples (identical for each class), and remain all other samples unlabeled. On STL-10, we use pre-defined folds. Due to spatial limitation, details on data pre-processing, training/validation splitting, training configurations and baselines are deferred to Appendix D. These settings follow the common practice of SSL (Oliver et al., 2018; Berthelot et al., 2019; Verma et al., 2019; Tarvainen & Valpola, 2017; Athiwaratkun et al., 2019). The hyper-parameter β of *Meta-Semi* is selected among $[0.2, 1]$ on the validation set.

4.2 MAIN RESULTS

Results on CIFAR with various numbers of labeled samples are presented in Tables 1, 2. It can be observed that *Meta-Semi* consistently outperforms state-of-the-art SSL algorithms in terms of generalization performance, especially with relatively less labeled data and larger numbers of classes. For example, when using CNN-13, on CIFAR-10 with 4000 labels, *Meta-Semi* outperforms the competitive baseline, ICT, by 0.13% in absolute error, while with 1,000 labels on CIFAR-10 and with 4,000 labels on CIFAR-100, *Meta-Semi* yields more significant improvements of 2.17% and 2.46%, respectively. A plausible explanation for this phenomenon is that in these challenging cases, the large majority of training data are unannotated, and thus the consistency regularization terms in the loss function of consistency based methods dominant the training process. However, minimizing the consistency loss does not necessarily ensure high generalization performance since the unlabeled data receives no direct supervision from the labeled data. In contrast, our method exploits labeled data to perform supervision on unlabeled data, which is more robust for tasks with small labeled

Table 4: Test errors on SVHN with varying amount of labeled data. We report the average results and the standard deviations of 5 independent experiments. All results are based on CNN-13. The best results are **bold-faced**.

Methods	SVHN 500 labels	SVHN 1000 labels
VAT (Miyato et al., 2018)	-	5.42%
Π-model (Laine & Aila, 2016)	$6.65 \pm 0.53\%$	$4.82 \pm 0.17\%$
Temp-ensemble (Laine & Aila, 2016)	$5.12 \pm 0.13\%$	$4.42 \pm 0.16\%$
Mean Teacher (Tarvainen & Valpola, 2017)	$4.18 \pm 0.27\%$	$3.95 \pm 0.19\%$
ICT (Verma et al., 2019)	$4.23 \pm 0.15\%$	$3.89 \pm 0.04\%$
SNTG (Luo et al., 2018)	$3.99 \pm 0.24\%$	$3.86 \pm 0.27\%$
Meta-Semi	$4.12 \pm 0.21\%$	$3.92 \pm 0.11\%$
Meta-Semi + ICT	$\mathbf{3.98 \pm 0.09\%}$	$\mathbf{3.77 \pm 0.05\%}$

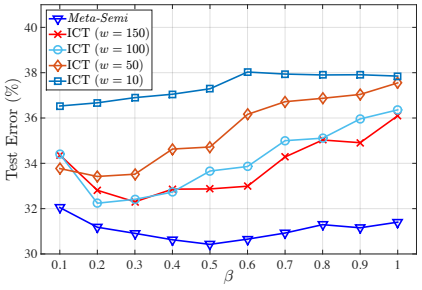

Figure 3: Test errors with varying β on CIFAR-100 using 10,000 labels. The CNN-13 network is used. We also report the results of ICT (Verma et al., 2019).

Table 5: Performance of *Meta-Semi* v.s. baselines with fixed amount of training time. We report the mean test errors of both networks on CIFAR-100 with 10,000 labels. The best results are **bold-faced**.

(a) CNN-13

Training Time	5.0h	7.5h	10.0h	12.6h
ICT (Verma et al., 2019)	33.43%	32.84%	32.61%	32.24%
Meta-Semi	**32.73%**	**31.81%**	**31.06%**	**30.84%**

(b) WRN-28

Training Time	13.7h	18.3h	22.8h	29.2h
MixMatch (Berthelot et al., 2019)	32.94%	31.91%	31.26%	30.84%
Meta-Semi	**31.74%**	**30.85%**	**30.50%**	**30.13%**

set. On the other hand, MixMatch utilizes MixUp augmentation to integrate labeled and unlabeled samples, showing robust performance with less labels as well, but *Meta-Semi* outperforms it in terms of test accuracy. Moreover, it is shown that the performance of *Meta-Semi* can be significantly improved by combining it with consistency generalization. On CIFAR-10 with 2000 labels, CNN-13 based *Meta-Semi* + ICT outperforms *Meta-Semi* by 1.37%.

Another important observation is that our method has only 1 tunable hyper-parameter. Provided that each hyper-parameter has M candidates, the cost of *Meta-Semi* for hyper-parameter search will be $1/M$ of VAT and Mean Teacher, and $1/M^3$ of MixMatch.

Results on STL-10 and SVHN are presented in Table 3 and Table 4, respectively. The results indicate that the test accuracy of *Meta-Semi* outperforms MixMatch by more than 2% on STL-10, and is comparable with state-of-the-art SSL algorithms on SVHN.

4.3 Hyper-parameter Sensitivity

The β parameter for the Beta distribution in MixUp augmentation is the only additional hyper-parameter that needs to be tuned when *Meta-Semi* is implemented in new SSL tasks. To study the sensitivity of our method to β, we vary the value of β, and present the test errors in Figure 3. For comparison, we also present the results of ICT (Verma et al., 2019) when its two additional hyper-parameters (β and the unsupervised regularization coefficient w) change among the recommended candidates provided by the original paper. One can observe that the performance of *Meta-Semi* is relatively stable when β ranges from 0.1 to 1. In contrast, ICT is sensitive to both the two hyper-parameters. It has been shown that hyper-parameter searching is difficult on realistic SSL tasks (Oliver et al., 2018). *Meta-Semi* can be more easily applied as it requires less effort for tuning hyper-parameters.

4.4 Efficiency of *Meta-Semi*

Our method generally requires more training time for each iteration as it includes bi-level optimization. However, we find that our algorithm converges fast and if we consider a fixed amount of training time, it still outperforms the others, as shown in Table 5.

5 Conclusion

In this paper, we have presented a novel semi-supervised classification algorithm under the meta-learning paradigm. The proposed *Meta-Semi* algorithm is capable of adapting to various SSL tasks with impressive performance via tuning only one additional hyper-parameter, and empirically we have observed that the model performance is robust to different settings of this hyper-parameter. Theoretically, we have provided the convergence analysis to show that *Meta-Semi* always converges to a stationary point under mild conditions. On four competitive datasets, *Meta-Semi* has achieved state-of-the-art performance compared to existing deep SSL algorithms.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, pp. 3981–3989, 2016.

Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. 2019.

Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *NeurIPS*, pp. 3365–3373, 2014.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 11 label propagation and quadratic criterion. 2006.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. In *ICLR*, 2020.

Dimitri P Bertsekas. Nonlinear programming. *Athena Scientific*, 1997.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA, September 2006.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pp. 215–223, 2011.

Emily Denton, Sam Gross, and Rob Fergus. Semi-supervised learning with context-conditional generative adversarial networks. *arXiv preprint arXiv:1611.06430*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pp. 1126–1135. JMLR. org, 2017.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NeurIPS*, pp. 529–536, 2005.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. In *ICLR*, 2017.

Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. Convolutional networks with dense connectivity. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pp. 200–209, 1999.

Thorsten Joachims. Transductive learning via spectral graph partitioning. In *AAAI*, pp. 290–297, 2003.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pp. 3581–3589, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pp. 1097–1105, 2012.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, volume 3, pp. 2, 2013.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. In *CVPR*, pp. 8896–8905, 2018.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NuerIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Augustus Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *NeurIPS*, pp. 3235–3246, 2018.

Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. In *AAAI*, 2018.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018a.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018b.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NeurIPS*, pp. 1163–1171, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pp. 1–9, 2015.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, pp. 1195–1204, 2017.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pp. 384–394. Association for Computational Linguistics, 2010.

Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. In *IJCAI*, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pp. 912–919, 2003.

APPENDIX: META-SEMI: A META-LEARNING APPROACH FOR SEMI-SUPERVISED LEARNING

A PROOF OF PROPOSITION 1

This section provides the proof of Proposition 1.

Proposition 1. *Suppose that $\overline{\boldsymbol{\theta}}_M^t$ is given by M times of gradient descents starting from $\overline{\boldsymbol{\theta}}_0^t = \boldsymbol{\theta}^t$. Then we have*

$$\left.\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial w_j}\right|_{\boldsymbol{w}=0} = M\left[\left.\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t))}{\partial w_j}\right|_{\boldsymbol{w}=0}\right],\ \forall\, 1\leq j\leq |\tilde{\mathcal{U}}|. \tag{17}$$

Proof. According to the updating rule $\overline{\boldsymbol{\theta}}_M^t = \overline{\boldsymbol{\theta}}_{M-1}^t - \alpha^t \nabla_{\overline{\boldsymbol{\theta}}_{M-1}^t} \sum_{k=1}^{|\tilde{\mathcal{U}}|} w_k L(\hat{\boldsymbol{y}}_k, p(\tilde{\boldsymbol{u}}_k|\overline{\boldsymbol{\theta}}_{M-1}^t))$, we obtain:

$$\left.\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial w_j}\right|_{\boldsymbol{w}=0} \tag{18}$$

$$= \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial \overline{\boldsymbol{\theta}}_M^t}\right]^T \left[\frac{\partial \overline{\boldsymbol{\theta}}_M^t}{\partial w_j}\right]\Bigg|_{\boldsymbol{w}=0} \tag{19}$$

$$= \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial \overline{\boldsymbol{\theta}}_M^t}\right]^T \left[\frac{\partial (\overline{\boldsymbol{\theta}}_{M-1}^t - \alpha^t \nabla_{\overline{\boldsymbol{\theta}}_{M-1}^t} \sum_{k=1}^{|\tilde{\mathcal{U}}|} w_k L(\hat{\boldsymbol{y}}_k, p(\tilde{\boldsymbol{u}}_k|\overline{\boldsymbol{\theta}}_{M-1}^t)))}{\partial w_j}\right]\Bigg|_{\boldsymbol{w}=0} \tag{20}$$

$$= \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial \overline{\boldsymbol{\theta}}_M^t}\right]^T \left[\frac{\partial \overline{\boldsymbol{\theta}}_{M-1}^t}{\partial w_j} - \alpha^t \sum_{k=1}^{|\tilde{\mathcal{U}}|} \left[\frac{\partial w_k \nabla_{\overline{\boldsymbol{\theta}}_{M-1}^t} L(\hat{\boldsymbol{y}}_k, p(\tilde{\boldsymbol{u}}_k|\overline{\boldsymbol{\theta}}_{M-1}^t))}{\partial w_k}\, \frac{\partial w_k}{\partial w_j}\right.\right.$$
$$\left.\left. + \frac{\partial w_k \nabla_{\overline{\boldsymbol{\theta}}_{M-1}^t} L(\hat{\boldsymbol{y}}_k, p(\tilde{\boldsymbol{u}}_k|\overline{\boldsymbol{\theta}}_{M-1}^t))}{\partial \nabla_{\overline{\boldsymbol{\theta}}_{M-1}^t} L(\hat{\boldsymbol{y}}_k, p(\tilde{\boldsymbol{u}}_k|\overline{\boldsymbol{\theta}}_{M-1}^t))}\, \frac{\partial \nabla_{\overline{\boldsymbol{\theta}}_{M-1}^t} L(\hat{\boldsymbol{y}}_k, p(\tilde{\boldsymbol{u}}_k|\overline{\boldsymbol{\theta}}_{M-1}^t))}{\partial w_j}\right]\right]\Bigg|_{\boldsymbol{w}=0} \tag{21}$$

$$= \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_M^t))}{\partial \overline{\boldsymbol{\theta}}_M^t}\right]^T \left[\frac{\partial \overline{\boldsymbol{\theta}}_{M-1}^t}{\partial w_j} - \alpha^t \left[\frac{\partial L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\overline{\boldsymbol{\theta}}_{M-1}^t))}{\partial \overline{\boldsymbol{\theta}}_{M-1}^t}\right.\right.$$
$$\left.\left. + \sum_{k=1}^{|\tilde{\mathcal{U}}|} w_k \frac{\partial \nabla_{\overline{\boldsymbol{\theta}}_{M-1}^t} L(\hat{\boldsymbol{y}}_k, p(\tilde{\boldsymbol{u}}_k|\overline{\boldsymbol{\theta}}_{M-1}^t))}{\partial w_j}\right]\right]\Bigg|_{\boldsymbol{w}=0} \tag{22}$$

$$= \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right]^T \left[\left.\frac{\partial \overline{\boldsymbol{\theta}}_{M-1}^t}{\partial w_j}\right|_{\boldsymbol{w}=0} - \alpha^t \frac{\partial L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right] \tag{23}$$

$$= \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right]^T \left[\left.\frac{\partial \overline{\boldsymbol{\theta}}_{M-2}^t}{\partial w_j}\right|_{\boldsymbol{w}=0} - 2\alpha^t \frac{\partial L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right] \tag{24}$$

$$= \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right]^T \left[\left.\frac{\partial \overline{\boldsymbol{\theta}}_0^t}{\partial w_j}\right|_{\boldsymbol{w}=0} - M\alpha^t \frac{\partial L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right] \tag{25}$$

$$= -M\alpha^t \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right]^T \left[\frac{\partial L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right]. \tag{26}$$

In the above, the Eq. (23) is obtained as we have $\overline{\boldsymbol{\theta}}_M^t = \overline{\boldsymbol{\theta}}_{M-1}^t = \ldots = \overline{\boldsymbol{\theta}}_0^t$ when $\boldsymbol{w} = 0$. The Eq. (25) follows repeatedly using Eqs. (18-23). Let $M = 1$, we have

$$\left.\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\overline{\boldsymbol{\theta}}_1^t))}{\partial w_j}\right|_{\boldsymbol{w}=0} = -\alpha^t \left[\frac{\partial \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right]^T \left[\frac{\partial L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t))}{\partial \boldsymbol{\theta}^t}\right]. \tag{27}$$

By combining Eq. (26) and Eq. (27), we prove the desired proposition. □

B PROOF OF PROPOSITION 2

This Section provides the proof of Proposition 2. In our proof, the MixUp augmentation is considered since it is an important part of our algorithm. We begin with a Lemma (Bertsekas, 1997) based on the definition of Lipschitz-smooth.

Definition 1. *A continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be Lipschitz-smooth with constant L if*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \forall x, y \in \mathbb{R}^n.$$

Lemma 1. *Assume that the continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz-smooth with the scalar L. Then*

$$f(x + y) \leq f(x) + y^T \nabla f(x) + \frac{L}{2}\|y\|^2, \forall x, y \in \mathbb{R}^n.$$

Then we introduce a mild assumption to restrict the expected norm of the gradients. The assumption is empirically shown to be generally held in semi-supervised learning.

To clearly present the assumption and the proof, we first define two new symbols. Suppose that the supervised loss on the labeled mini-batch $\tilde{\mathcal{X}}$ at t^{th} step is denoted by

$$G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t) = \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i|\boldsymbol{\theta}^t)). \tag{28}$$

In addition, we denote the dynamically weighted loss of pseudo-labeled samples by

$$F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t) = \mathcal{L}_{meta} = \frac{1}{\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t} \sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j|\boldsymbol{\theta}^t)), \tag{29}$$

Note that we assume $F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t) = 0$ if $\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t = 0$. Then we have the following assumption.

Assumption 1. *For all $t \geq 0$, there exists a positive scalar σ, such that*

$$\mathbb{E}_{\tilde{\mathcal{X}}, \tilde{\mathcal{U}}} \|\nabla_{\boldsymbol{\theta}^t} F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t)\|^2 \leq \sigma \|\nabla_{\boldsymbol{\theta}^t} \mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)\|^2.$$

Now we are ready to present the detailed proof. Our proof is partially inspired by the proof of convergence for gradient based methods with diminishing stepsize provided by (Bertsekas, 1997).

Proposition 2. *Assume that the loss function on labeled data $G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)$ is Lipschitz-smooth with regards to $\boldsymbol{\theta}^t$ for all $\tilde{\mathcal{X}}$, and that assumption 1 holds. Suppose also that the learning rate $\alpha^t > 0$ satisfies:*

$$\lim_{t \to \infty} \alpha^t = 0, \quad \sum_{t=0}^{\infty} \alpha^t = \infty. \tag{30}$$

Then every limit point of the sequence $\{\boldsymbol{\theta}^t\}$ generated by Meta-Semi is a stationary point of $\mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)$, namely,

$$\lim_{t \to \infty} \|\nabla_{\boldsymbol{\theta}^t} \mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)\| = 0.$$

Proof. The MixUp data augmentation needs to be considered because it is leveraged to generate $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{U}}$ from the original data. On the basis of the original labeled samples $\mathcal{X}$ and unlabeled samples $\mathcal{U}$ (associated with original pseudo labels), we have

$$\tilde{\mathcal{X}} = \text{MixUp}(\mathcal{X}, \text{Shuffle}(\mathcal{X}), \lambda_1), \lambda_1 \sim \text{Beta}(\alpha, \alpha), \tag{31}$$

$$\mathcal{W} = \text{Concat}(\mathcal{X}, \mathcal{U}), \tag{32}$$

$$\tilde{\mathcal{U}} = \text{MixUp}(\mathcal{W}, \text{Shuffle}(\mathcal{W}), \lambda_2), \lambda_2 \sim \text{Beta}(\alpha, \alpha). \tag{33}$$

Given that the MixUp augmentation is performed between the mini-batch and itself with certain random permutation, we define the expected loss over all possible permutations by

$$\begin{aligned} \overline{G}(\mathcal{X}, \boldsymbol{\theta}^t, \lambda_1) &= \mathop{\mathbb{E}}_{\tilde{\mathcal{X}} \in \text{MixUp}(\mathcal{X}, \text{Shuffle}(\mathcal{X}), \lambda_1)} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t) \\ &= \mathop{\mathbb{E}}_{\tilde{\mathcal{X}} \in \text{MixUp}(\mathcal{X}, \text{Shuffle}(\mathcal{X}), \lambda_1)} \sum_{i=1}^{|\tilde{\mathcal{X}}|} L(\tilde{\boldsymbol{y}}_i, p(\tilde{\boldsymbol{x}}_i | \boldsymbol{\theta}^t)), \end{aligned} \tag{34}$$

$$\begin{aligned} \overline{F}(\tilde{\mathcal{X}}, \mathcal{X}, \mathcal{U}, \boldsymbol{\theta}^t, \lambda_2) &= \mathop{\mathbb{E}}_{\tilde{\mathcal{U}} \in \text{MixUp}(\mathcal{W}, \text{Shuffle}(\mathcal{W}), \lambda_2)} \left[\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t \right] F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t) \\ &= \mathop{\mathbb{E}}_{\tilde{\mathcal{U}} \in \text{MixUp}(\mathcal{W}, \text{Shuffle}(\mathcal{W}), \lambda_2)} \sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j | \boldsymbol{\theta}^t)), \end{aligned} \tag{35}$$

where the first argument $\tilde{\mathcal{X}}$ of $\overline{F}(\cdot)$ is used for determining the dynamic weights of pseudo-labeled samples. Then we solve $\overline{G}$ and $\overline{F}$ in a closed form. Consider the following problem: N different items are paired to the same N items. Obviously, there are $N!$ modes of pairing in total. If we fix certain pair, we will have $(N-1)!$ modes of pairing left. Therefore, if we combine all $N!$ possible pairing modes together, we will find that any item is paired to every item (including itself) for $(N-1)!$ times. Similarly, in our problem, it is easy to obtain

$$\overline{G}(\mathcal{X}, \boldsymbol{\theta}^t, \lambda_1) = \frac{1}{|\mathcal{X}|!} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}} (|\mathcal{X}| - 1)! f_{ij}(\boldsymbol{\theta}^t, \lambda_1) = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}} f_{ij}(\boldsymbol{\theta}^t, \lambda_1), \tag{36}$$

where $\boldsymbol{x}_i, \boldsymbol{x}_j$ are the original samples that can be either labeled or unlabeled. The loss of the augmented sample generated by performing MixUp augmentation between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with λ_1 is denoted by $f_{ij}(\boldsymbol{\theta}^t, \lambda_1)$. In a similar way, we can obtain

$$\overline{F}(\tilde{\mathcal{X}}, \mathcal{X}, \mathcal{U}, \boldsymbol{\theta}^t, \lambda_2) = \frac{1}{|\tilde{\mathcal{U}}|} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X} \cup \mathcal{U}} w_{ij}^t(\lambda_2)\ f_{ij}(\boldsymbol{\theta}^t, \lambda_2), \tag{37}$$

where $w_{ij}^t(\lambda_2)$ is the dynamic weight determined by

$$w_{ij}^t(\lambda_2) = \begin{cases} 1 & [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} f_{ij}(\boldsymbol{\theta}^t, \lambda_2)] \geq 0 \\ 0 & [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} f_{ij}(\boldsymbol{\theta}^t, \lambda_2)] < 0 \end{cases}. \tag{38}$$

Now, consider the following inequation

$$[\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t)] = \frac{1}{\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t} [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T \left[\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t [\nabla_{\boldsymbol{\theta}^t} L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j | \boldsymbol{\theta}^t))] \right] \tag{39}$$

$$\geq \frac{1}{|\tilde{\mathcal{U}}|} [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T \left[\sum_{j=1}^{|\tilde{\mathcal{U}}|} w_j^t [\nabla_{\boldsymbol{\theta}^t} L(\hat{\boldsymbol{y}}_j, p(\tilde{\boldsymbol{u}}_j | \boldsymbol{\theta}^t))] \right]. \tag{40}$$

By taking the expectation over $\tilde{\mathcal{U}} \in \text{MixUp}(\mathcal{W}, \text{Shuffle}(\mathcal{W}), \lambda_2)$, we further obtain

$$\mathop{\mathbb{E}}_{\tilde{\mathcal{U}} \in \text{MixUp}(\mathcal{W}, \text{Shuffle}(\mathcal{W}), \lambda_2)} [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t)] \tag{41}$$

$$\geq \frac{1}{|\tilde{\mathcal{U}}|} [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} \overline{F}(\tilde{\mathcal{X}}, \mathcal{X}, \mathcal{U}, \boldsymbol{\theta}^t, \lambda_2)] \tag{42}$$

$$= \frac{1}{|\tilde{\mathcal{U}}|^2} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X} \cup \mathcal{U}} w_{ij}^t(\lambda_2) \, [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} f_{ij}(\boldsymbol{\theta}^t, \lambda_2)] \tag{43}$$

$$\geq \frac{1}{|\tilde{\mathcal{U}}|^2} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}} w_{ij}^t(\lambda_2) \, [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} f_{ij}(\boldsymbol{\theta}^t, \lambda_2)] \tag{44}$$

$$\geq \frac{1}{|\tilde{\mathcal{U}}|^2} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}} [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} f_{ij}(\boldsymbol{\theta}^t, \lambda_2)] \tag{45}$$

$$= \frac{|\tilde{\mathcal{X}}|}{|\tilde{\mathcal{U}}|^2} [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} \overline{G}(\mathcal{X}, \boldsymbol{\theta}^t, \lambda_2)]. \tag{46}$$

Then by taking the expectation over $\tilde{\mathcal{X}} \in \text{MixUp}(\mathcal{X}, \text{Shuffle}(\mathcal{X}), \lambda_1)$, we have

$$\mathop{\mathbb{E}}_{\tilde{\mathcal{X}} \in \text{MixUp}(\mathcal{X}, \text{Shuffle}(\mathcal{X}), \lambda_1)} \; \mathop{\mathbb{E}}_{\tilde{\mathcal{U}} \in \text{MixUp}(\mathcal{W}, \text{Shuffle}(\mathcal{W}), \lambda_2)} [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t)] \tag{47}$$

$$\geq \frac{|\tilde{\mathcal{X}}|}{|\tilde{\mathcal{U}}|^2} [\nabla_{\boldsymbol{\theta}^t} \overline{G}(\mathcal{X}, \boldsymbol{\theta}^t, \lambda_1)]^T [\nabla_{\boldsymbol{\theta}^t} \overline{G}(\mathcal{X}, \boldsymbol{\theta}^t, \lambda_2)]. \tag{48}$$

Finally, we take the expectation over λ_1, λ_2 and all possible batches $\mathcal{X}$,$\mathcal{U}$. Following from the convexity of $\|\cdot\|^2$, we have $\mathbb{E}(\|\cdot\|^2) \geq \|\mathbb{E}(\cdot)\|^2$. Therefore, we obtain

$$\mathbb{E}_{\tilde{\mathcal{X}}, \tilde{\mathcal{U}}} [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t)] \geq \frac{|\tilde{\mathcal{X}}|}{|\tilde{\mathcal{U}}|^2} \mathbb{E}_{\mathcal{X}, \mathcal{U}} \|\nabla_{\boldsymbol{\theta}^t} \mathbb{E}_{\lambda} \overline{G}(\mathcal{X}, \boldsymbol{\theta}^t, \lambda)\|^2 \tag{49}$$

$$\geq \frac{|\tilde{\mathcal{X}}|}{|\tilde{\mathcal{U}}|^2} \|\nabla_{\boldsymbol{\theta}^t} \mathbb{E}_{\mathcal{X}, \mathcal{U}} \mathbb{E}_{\lambda} \overline{G}(\mathcal{X}, \boldsymbol{\theta}^t, \lambda)\|^2 \tag{50}$$

$$\geq \frac{|\tilde{\mathcal{X}}|}{|\tilde{\mathcal{U}}|^2} \|\nabla_{\boldsymbol{\theta}^t} \mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)\|^2, \tag{51}$$

where Inequality (49) is obtained as λ_1, λ_2 are mutually independent. Then we consider the updating rule of the stochastic gradient descent (SGD) algorithm:

$$\Delta \boldsymbol{\theta} = \boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t = -\alpha^t \nabla_{\boldsymbol{\theta}^t} F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t). \tag{52}$$

Assume that the loss function on labeled data $G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)$ is Lipschitz-smooth with the constant L. Following Lemma 1, we have

$$G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^{t+1}) \leq G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t) + [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T \Delta \boldsymbol{\theta} + \frac{L}{2} \|\Delta \boldsymbol{\theta}\|^2 \tag{53}$$

$$= G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t) + \alpha^t \left[\frac{1}{2} \alpha^t L \|\nabla_{\boldsymbol{\theta}^t} F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t)\|^2 - [\nabla_{\boldsymbol{\theta}^t} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)]^T [\nabla_{\boldsymbol{\theta}^t} F(\tilde{\mathcal{X}}, \tilde{\mathcal{U}}, \boldsymbol{\theta}^t)] \right]. \tag{54}$$

Take the expectation over all possible $\tilde{\mathcal{X}}$,$\tilde{\mathcal{U}}$, and thus we obtain the following inequality using Assumption 1 and Inequality (51):

$$\mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^{t+1}) \leq \; \mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t) + \alpha^t \left[\frac{1}{2} \alpha^t \sigma L - \frac{|\tilde{\mathcal{X}}|}{|\tilde{\mathcal{U}}|^2} \right] \|\nabla_{\boldsymbol{\theta}^t} \mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)\|^2. \tag{55}$$

As $\alpha^t \to 0$, there exists some positive constant c such that for all t greater than some index $\bar{t}$, we have

$$\mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^{t+1}) \leq \; \mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t) - \alpha^t c \|\nabla_{\boldsymbol{\theta}^t} \mathbb{E}_{\tilde{\mathcal{X}}} G(\tilde{\mathcal{X}}, \boldsymbol{\theta}^t)\|^2, \;\; \forall t \geq \bar{t}. \tag{56}$$

We see that $\{\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\}$ is monotonically decreasing for all $t \geq \bar{t}$. As $G(\cdot)$ is computed using the cross-entropy loss over the predictions of networks, it follows $\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t) \geq 0$. Therefore, $\{\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\}$ converges to a finite value. By adding Inequality (56) over all $t > \bar{t}$, we obtain

$$c\sum_{t=\bar{t}}^{\infty}\alpha^t\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|^2 \leq \mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^{\bar{t}}) - \lim_{t\to\infty}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t) < \infty. \tag{57}$$

It cannot exist an $\epsilon > 0$ such that $\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|^2 > \epsilon$ for all t greater than some $\hat{t}$. If so, as $\sum_{t=0}^{\infty}\alpha^t = \infty$, the left side of Inequality (57) will come to infinity. Therefor, we must have:

$$\liminf_{t\to\infty}\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\| = 0. \tag{58}$$

In the following, we will show that $\limsup_{t\to\infty}\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\| = 0$. Firstly, assume the contaary, namely

$$\limsup_{t\to\infty}\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\| \geq \epsilon > 0. \tag{59}$$

Let $\{m_j\}$ and $\{n_j\}$ be the sequences of indexes such that

$$m_j < n_j < m_{j+1}, \tag{60}$$

$$\frac{\epsilon}{3} < \|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|, \quad m_j \leq t < n_j, \tag{61}$$

$$\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\| \leq \frac{\epsilon}{3}, \quad n_j \leq t < m_{j+1}. \tag{62}$$

Since $G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)$ is Lipschitz-smooth, it is easy to see that $\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)$ is also Lipschitz-smooth. Suppose that the corresponding Lipschitz constant is L'. Let $\bar{j}$ be a sufficiently large index such that

$$\sum_{t=m_{\bar{j}}}^{\infty}\alpha^t\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|^2 < \frac{\epsilon^2}{9\sqrt{\sigma}L'}. \tag{63}$$

For any $j \geq \bar{j}$ and any m with $m_j \leq m \leq n_j - 1$, we have

$$\|\nabla_{\boldsymbol{\theta}^{n_j}}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^{n_j}) - \nabla_{\boldsymbol{\theta}^m}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^m)\| \leq \sum_{t=m}^{n_j-1}\|\nabla_{\boldsymbol{\theta}^{t+1}}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^{t+1}) - \nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\| \tag{64}$$

$$\leq L'\sum_{t=m}^{n_j-1}\|\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t\| \tag{65}$$

$$= L'\sum_{t=m}^{n_j-1}\alpha^t\|\nabla_{\boldsymbol{\theta}^t}F(\tilde{\mathcal{X}},\tilde{\mathcal{U}},\boldsymbol{\theta}^t)\|. \tag{66}$$

By taking the expectation over $\tilde{\mathcal{X}},\tilde{\mathcal{U}}$, we have

$$\|\nabla_{\boldsymbol{\theta}^{n_j}}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^{n_j}) - \nabla_{\boldsymbol{\theta}^m}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^m)\| \leq L'\sum_{t=m}^{n_j-1}\alpha^t\mathbb{E}_{\tilde{\mathcal{X}},\tilde{\mathcal{U}}}\|\nabla_{\boldsymbol{\theta}^t}F(\tilde{\mathcal{X}},\tilde{\mathcal{U}},\boldsymbol{\theta}^t)\| \tag{67}$$

$$\leq \sqrt{\sigma}L'\sum_{t=m}^{n_j-1}\alpha^t\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\| \tag{68}$$

$$\leq \frac{3\sqrt{\sigma}L'}{\epsilon}\sum_{t=m}^{n_j-1}\alpha^t\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|^2 \tag{69}$$

$$\leq \frac{3\sqrt{\sigma}L'}{\epsilon}\frac{\epsilon^2}{9\sqrt{\sigma}L'} \tag{70}$$

$$= \frac{\epsilon}{3}, \tag{71}$$

where Inequality (69) follows from Inequality (61) and Inequality (68) follows from

$$\mathbb{E}_{\tilde{\mathcal{X}},\tilde{\mathcal{U}}}\|\nabla_{\boldsymbol{\theta}^t}F(\tilde{\mathcal{X}},\tilde{\mathcal{U}},\boldsymbol{\theta}^t)\|\leq\sqrt{\mathbb{E}_{\tilde{\mathcal{X}},\tilde{\mathcal{U}}}\|\nabla_{\boldsymbol{\theta}^t}F(\tilde{\mathcal{X}},\tilde{\mathcal{U}},\boldsymbol{\theta}^t)\|^2}\leq\sqrt{\sigma}\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|. \tag{72}$$

Thus, we have

$$\|\nabla_{\boldsymbol{\theta}^m}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^m)\|\leq\|\nabla_{\boldsymbol{\theta}^{n_j}}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^{n_j})\|+\frac{\epsilon}{3}\leq\frac{2\epsilon}{3},\quad\forall j\geq\bar{j},m_j\leq m\leq n_j-1. \tag{73}$$

As the inequality holds for all $m\geq m_{\bar{j}}$, we finally obtain

$$\|\nabla_{\boldsymbol{\theta}^m}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^m)\|\leq\frac{2\epsilon}{3},\quad\forall m\geq m_{\bar{j}}, \tag{74}$$

which contradicts Inequality (59), implying that

$$\liminf_{t\to\infty}\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|=\limsup_{t\to\infty}\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|=0. \tag{75}$$

Therefore, we prove that $\lim_{t\to\infty}\|\nabla_{\boldsymbol{\theta}^t}\mathbb{E}_{\tilde{\mathcal{X}}}G(\tilde{\mathcal{X}},\boldsymbol{\theta}^t)\|=0$. □

C Ablation Study

To provide additional insights into our method, we further conduct the ablation experiments by removing or altering the components of *Meta-Semi*. The results are shown in Table 6. It can be seen that parameter EMA and performing MixUp on unlabeled data are both important techniques to achieve high generalization performance. The observation is consistent with (Verma et al., 2019). In addition, if all pseudo-labeled samples are weighted by the constant 1, *Meta-Semi* is equivalent to a consistency based algorithm, which also shows effective performance.

Table 6: Ablation study results. We report the test errors on CIFAR-100 with 4,000 and 10,000 labels. The CNN-13 network is used.

Ablation	CIFAR-100 4000 labels	CIFAR-100 10000 labels
Without parameter EMA	47.68 ± 0.27%	37.15 ± 1.02%
One-hot pseudo labels	41.52 ± 0.51%	32.78 ± 0.41%
MixUp on unlabeled data only	37.69 ± 0.50%	30.56 ± 0.39%
MixUp on labeled data only	45.90 ± 0.15%	36.11 ± 0.21%
Without MixUp	46.71 ± 0.05%	35.98 ± 0.69%
Reweighting with the constant 1	40.26 ± 0.64%	32.17 ± 0.14%
Reweighting with -1 and 1	45.41 ± 0.38%	36.39 ± 0.44%
Meta-Semi	37.61 ± 0.56%	30.51 ± 0.32%
Meta-Semi + ICT	37.12 ± 0.59%	29.68 ± 0.05%

D Details of Experiments

Datasets. (1) The CIFAR-10 / CIFAR-100 datasets consist of 60,000 32x32 colored images of 10 / 100 classes, 50,000 for training and 10,000 for test. Following the common practice of SSL (Oliver et al., 2018; Berthelot et al., 2019; Verma et al., 2019; Tarvainen & Valpola, 2017; Athiwaratkun et al., 2019), we hold out 5k images from the training set as the validation set. Images are normalized with channel means and standard deviations for pre-processing. Then data augmentation is performed by 4x4 random translation followed by random horizontal flip (He et al., 2016; Huang et al., 2019). On CIFAR-10, we preserve 100, 200 and 400 labels per class respectively, corresponding to 1000, 2000, 4000 labeled samples in total. All other samples are unlabeled. We randomly split the dataset for 5 times to conduct multiple experiments, and report the mean test errors associated with standard deviations. Similarly, On CIFAR-100, evaluation is performed with 40 and 100 randomly preserved labeled samples per class. (2) SVHN consists of 32x32 colored images of digits. 73,257 images for training, 26,032 images for testing and 531,131 images for additional training are provided. Following (Luo et al., 2018; Tarvainen & Valpola, 2017), we merely perform random 2x2 translation to augment the training set, and hold out 1,000 images for validation. Similar to CIFAR, we randomly preserve 500 and 1,000 labels for experiments. (3) STL-10 (Coates et al., 2011) contains 5,000 training examples divided into 10 predefined folds with 1000 examples each, and 100,000 unlabeled images drawn from a similar—but not identical—data distribution. All the samples are 96x96 colored images. We use the same experimental protocol as (Berthelot et al., 2019).

Networks. Our experiments are based on a 13-layer CNN (CNN-13) and the Wide-RestNet-28-2 (WRN-28) network. The CNN-13 network has been adopted as the standard model for experiments

by state-of-the-art SSL algorithms (Verma et al., 2019; Tarvainen & Valpola, 2017; Athiwaratkun et al., 2019; Miyato et al., 2018; Luo et al., 2018; Park et al., 2018). Following (Verma et al., 2019), we remove the Gaussian noise layer and the dropout layer in the network. Other methods use these techniques if mentioned in their original papers, which provide stronger regularization. Some recent works adopt the WRN-28 network (Oliver et al., 2018; Berthelot et al., 2019) in their experiments. We also implement *Meta-Semi* with WRN-28 to present comparisons with them.

Large Validation Set. We note that the validation set we use may be relatively large in some settings (e.g. 5,000 for validation on CIFAR-10 with 1,000 labeled examples). However, since most prior SSL methods do so, we simply follow them to produce comparable results with them in the paper. On the other hand, as discussed in the sensitivity test, our method is less sensitive to the only tunable hyper-parameter β and thus requires less validation efforts. To further demonstrate this point, we perform a four-fold cross-validation on CNN-13 based *Meta-Semi* with 1,000 labeled samples on CIFAR-10 to search for the optimal β. Our method achieves a test error of $10.96 \pm 0.56\%$, which is slightly higher than the $10.27 \pm 0.66\%$ of using addition 5,000 labeled samples for validation, but still significantly outperforms baselines.

Training details. The CNN-13 network uses the SGD optimizer with a Nesterov momentum of 0.9. The L2 regularization coefficient is set to 1e-4, and the initial learning rate is set to 0.1. For all experiments with CNN-13, we train the network for 600 epochs using the cosine learning rate annealing technique (Loshchilov & Hutter, 2016; Huang et al., 2017; Verma et al., 2019). The batch size of labeled samples and unlabeled samples are set to 25 and 75 respectively. To generate pseudo labels for unlabeled samples, we use an exponential moving average on model parameters with a decay rate of 0.999. For WRN-28, we adopt exactly the same training details as (Berthelot et al., 2019) except for the batch size: we use 32 for labeled samples and 96 for unlabeled samples. The ratio of labeled/unlabeled samples in each mini-batch is always set to 1:3 in *Meta-Semi*, which consistently achieves excellent performance on the validation set, and does not need to be tuned for the specific SSL task.

Baselines. Our method is compared with several state-of-the-art baselines including SSL algorithms and a meta-reweighting method.

- Π-model (Laine & Aila, 2016) enforces the model predictions to remain the same when different augmentation and dropout modes are performed.
- Temp-ensemble (Laine & Aila, 2016) attaches a soft pseudo label for each unlabeled sample by performing a moving average on the historical predictions of networks.
- Mean Teacher (MT) (Tarvainen & Valpola, 2017) establishes a teacher network by performing exponential moving average on the parameters of the model, and leverages the teacher networks to produces supervision for unlabeled data.
- Virtual Adversarial Training (VAT) (Miyato et al., 2018) adds adversarial perturbations to the samples and enforce the model to have the same predictions on perturbed samples and the original samples.
- Smooth Neighbors on Teacher Graphs (SNTG)(Luo et al., 2018) constructs a teacher graph to regularize the feature distribution of unlabeled samples.
- Learning to Reweight (Ren et al., 2018b) proposes to reweight different training samples by solving a similar meta-learning problem to us. Since their original algorithm requires labels of all the training, we adopt a version modified for SSL in this paper. In specific, we retain our approach of generating pseudo-labeled samples, but use their reweighting strategy.
- MT + Fast SWA (Athiwaratkun et al., 2019) is an improved MT algorithm using a fast stochastic weight averaging optimizer.
- Interpolation Consistency Training (ICT) (Verma et al., 2019) encourages the prediction on an interpolation of unlabeled samples to be consistent with the interpolation of the predictions on those points. They first use MixUp augmentation in deep SSL.
- MixMatch (Berthelot et al., 2019) is a holistic deep SSL approach that integrates various dominant consistency regularization techniques.

We implement these methods in the same codebase, and search for the best hyper-parameters for them on the validation set according to the recommendations provided by their original papers. Notably, for MixMatch (Berthelot et al., 2019), we fix the sharpening temperature $T = 0.5$ and the number of unlabeled augmentations $K = 2$, and adjust the α parameter for Beta distribution and the unsupervised loss coefficient $\lambda_{\tilde{\mathcal{U}}}$, as suggested by the paper. We first reproduce the CIFAR-10 results of MixMatch reported by their paper, and then tune α and $\lambda_{\tilde{\mathcal{U}}}$ on the validation set of CIFAR-100.