Semi-Supervised Learning with Meta-Gradient

Xin-Yu Zhang, Taihong Xiao, Haolin Jia, Ming-Ming Cheng, Ming-Hsuan Yang

xinyuzhang@mail.nankai.edu.cn

Overview

Semi-Supervised Learning

Semi-supervised learning (SSL): labeled data + unlabeled data $\Longrightarrow$ better generalization ability.

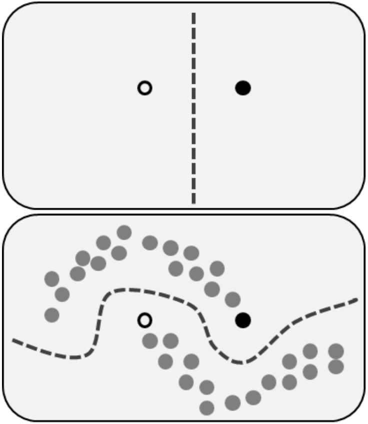

Figure: Illustration of the role of unlabeled data[1].

[1] https://en.wikipedia.org/wiki/Semi-supervised_learning

Consistency-Based SSL

Basic assumption: prediction consistency against perturbations of the input signals or model weights.

[2] Miyato *et al.* Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. In *TPAMI*, 2018

[3] Tarvainen & Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017

Consistency-Based SSL

Basic assumption: prediction consistency against perturbations of the input signals or model weights.

Two research directions for consistency-based SSL: **perturbing in the adversarial directions**[2] and **finding better "role models"**[3].

[2] Miyato *et al.* Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. In *TPAMI*, 2018

[3] Tarvainen & Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017

Consistency-Based SSL

Basic assumption: prediction consistency against perturbations of the input signals or model weights.

Two research directions for consistency-based SSL: **perturbing in the adversarial directions**[2] and **finding better "role models"**[3].

Problems: consistency loss is rather generic and does not exploit the label information (not designed for the specific task).

[2] Miyato *et al.* Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. In *TPAMI*, 2018

[3] Tarvainen & Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017

Consistency-Based SSL

Basic assumption: prediction consistency against perturbations of the input signals or model weights.

Two research directions for consistency-based SSL: **perturbing in the adversarial directions**[2] and **finding better "role models"**[3].

Problems: consistency loss is rather generic and does not exploit the label information (not designed for the specific task).

Our work is to bridge the gap between the consistency loss and the label information.

[2] Miyato *et al.* Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. In *TPAMI*, 2018

[3] Tarvainen & Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017

Overview

Challenge. The consistency loss, which is calculated from the unlabeled data, seems to have no relationship with the labeled data.

Overview

Challenge. The consistency loss, which is calculated from the unlabeled data, seems to have no relationship with the labeled data.

Solution. Magic of *meta-learning*: unfolding and differentiating through one SGD step.

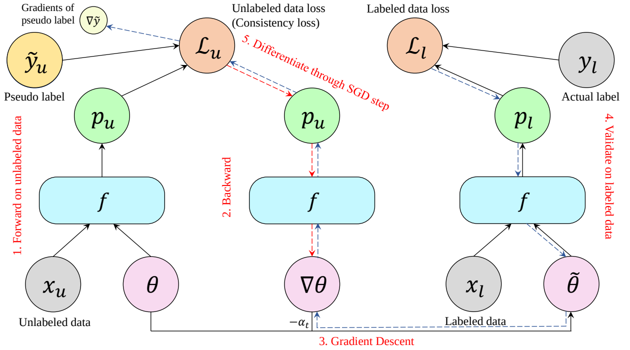

Figure: Illustration of the meta-learning philosophy.

Derivation

Formulation:

$$\begin{aligned} \min_{\mathcal{Y}} \quad & \sum_{k=1}^{N^l} \mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \boldsymbol{\theta}^*(\mathcal{Y})) \\ \text{s.t. } \ & \boldsymbol{\theta}^*(\mathcal{Y}) = \arg\min_{\theta} \sum_{i=1}^{N^u} \mathcal{L}(\boldsymbol{x}_i^u, \widehat{\boldsymbol{y}}_i; \boldsymbol{\theta}). \end{aligned} \tag{1}$$

Derivation

Formulation:

$$\begin{aligned} &\min_{\mathcal{Y}} \sum_{k=1}^{N^l} \mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \boldsymbol{\theta}^*(\mathcal{Y})) \\ &\text{s.t. } \boldsymbol{\theta}^*(\mathcal{Y}) = \arg\min_{\theta} \sum_{i=1}^{N^u} \mathcal{L}(\boldsymbol{x}_i^u, \widehat{\boldsymbol{y}}_i; \boldsymbol{\theta}). \end{aligned} \tag{1}$$

Solving Eq. (1) exactly is impossible, we adopt online approximation on the batch level.

Derivation

Initialize pseudo-labels:

$$\widetilde{\boldsymbol{y}}_i = f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t). \tag{2}$$

Derivation

Initialize pseudo-labels:

$$\widetilde{\boldsymbol{y}}_i = f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t). \tag{2}$$

Compute the unlabeled data loss and back-propagate the gradients:

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t) &= \Phi(f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t), \widetilde{\boldsymbol{y}}_i), \\ \nabla \boldsymbol{\theta}_t &= \frac{1}{B^u} \sum_{i=1}^{B^u} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t). \end{aligned} \tag{3}$$

Derivation

Initialize pseudo-labels:

$$\widetilde{\boldsymbol{y}}_i = f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t). \tag{2}$$

Compute the unlabeled data loss and back-propagate the gradients:

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t) &= \Phi(f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t), \widetilde{\boldsymbol{y}}_i), \\ \nabla \boldsymbol{\theta}_t &= \frac{1}{B^u} \sum_{i=1}^{B^u} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t). \end{aligned} \tag{3}$$

Apply one SGD step on the model parameters:

$$\widetilde{\boldsymbol{\theta}}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla \boldsymbol{\theta}_t, \tag{4}$$

where α_t is the learning rate of the inner loop.

Derivation

Evaluate on the labeled data and differentiate the labeled data loss:

$$\mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}) = \Phi(f(\boldsymbol{x}_k^l; \widetilde{\boldsymbol{\theta}}_{t+1}), \boldsymbol{y}_k),$$
$$\nabla \widetilde{\boldsymbol{y}}_i = \frac{1}{B^l} \sum_{k=1}^{B^l} \nabla_{\widetilde{\boldsymbol{y}}_i} \mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}). \tag{5}$$

Derivation

Evaluate on the labeled data and differentiate the labeled data loss:

$$\mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}) = \Phi(f(\boldsymbol{x}_k^l; \widetilde{\boldsymbol{\theta}}_{t+1}), \boldsymbol{y}_k),$$
$$\nabla \widetilde{\boldsymbol{y}}_i = \frac{1}{B^l} \sum_{k=1}^{B^l} \nabla_{\widetilde{\boldsymbol{y}}_i} \mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}). \tag{5}$$

Perform one SGD step on the pseudo-labels:

$$\widehat{\boldsymbol{y}}_i = \widetilde{\boldsymbol{y}}_i - \beta_t \nabla \widetilde{\boldsymbol{y}}_i, \tag{6}$$

where β_t is the meta learning rate,

Derivation

Evaluate on the labeled data and differentiate the labeled data loss:

$$\mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}) = \Phi(f(\boldsymbol{x}_k^l; \widetilde{\boldsymbol{\theta}}_{t+1}), \boldsymbol{y}_k),$$
$$\nabla \widetilde{\boldsymbol{y}}_i = \frac{1}{B^l} \sum_{k=1}^{B^l} \nabla_{\widetilde{\boldsymbol{y}}_i} \mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}). \tag{5}$$

Perform one SGD step on the pseudo-labels:

$$\widehat{\boldsymbol{y}}_i = \widetilde{\boldsymbol{y}}_i - \beta_t \nabla \widetilde{\boldsymbol{y}}_i, \tag{6}$$

where β_t is the meta learning rate,

Compute the consistency loss from the unlabeled data and the updated pseudo-labels.

Meta-Learning Algorithm

Algorithm 1 Meta-Learning Algorithm.

Input: regular learning rates $\{\alpha_t\}$,
meta learning rates $\{\beta_t\}$

for $t := 1$ *to #iters* **do**

$\{(\boldsymbol{x}_k^l, \boldsymbol{y}_k)\}_{k=1}^{B^l} \leftarrow \text{BatchSampler}(\mathcal{D}^l)$

$\{\boldsymbol{x}_i^u\}_{i=1}^{B^u} \leftarrow \text{BatchSampler}(\mathcal{D}^u)$

$\widetilde{\boldsymbol{y}}_i = f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t)$

$\mathcal{L}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t) = \Phi(f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t), \widetilde{\boldsymbol{y}}_i)$

$\nabla\boldsymbol{\theta}_t = \frac{1}{B^u}\sum_{i=1}^{B^u} \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t)$

$\widetilde{\boldsymbol{\theta}}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla\boldsymbol{\theta}_t$

$\mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}) = \Phi(f(\boldsymbol{x}_k^l; \widetilde{\boldsymbol{\theta}}_{t+1}), \boldsymbol{y}_k)$

$\nabla\widetilde{\boldsymbol{y}}_i = \frac{1}{B^l}\sum_{k=1}^{B^l} \nabla_{\widetilde{\boldsymbol{y}}_i}\mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1})$

$\widehat{\boldsymbol{y}}_i = \widetilde{\boldsymbol{y}}_i - \beta_t \nabla\widetilde{\boldsymbol{y}}_i$

$\mathcal{L}(\boldsymbol{x}_i^u, \widehat{\boldsymbol{y}}_i; \boldsymbol{\theta}_t) = \Phi(f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t), \widehat{\boldsymbol{y}}_i)$

$\nabla\widehat{\boldsymbol{\theta}}_t = \frac{1}{B^u}\sum_{i=1}^{B^u} \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{x}_i^u, \widehat{\boldsymbol{y}}_i; \boldsymbol{\theta}_t)$

$\boldsymbol{\theta}_{t+1} = \text{Optimizer}(\boldsymbol{\theta}_t, \nabla\widehat{\boldsymbol{\theta}}_t, \alpha_t)$

end

Convergence Theory — Convergence Guarantee

Theorem

Let

$$G(\boldsymbol{\theta}; \mathcal{D}^l) = \frac{1}{N^l} \sum_{k=1}^{N^l} \mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \boldsymbol{\theta}_t) \tag{7}$$

be the loss function of the labeled examples. Assume

(i) $\nabla_{\boldsymbol{\theta}} G$ is Lipschitz-continuous with a Lipschitz constant L_0; and

(ii) the norm of the Jacobian matrix of f *w.r.t.* $\boldsymbol{\theta}$ is upper-bounded, *i.e.*,

$$\exists M \in \mathbb{R}, \ s.t. \ \|J_{\boldsymbol{\theta}} f(\boldsymbol{x}_i^u; \boldsymbol{\theta})\| \leq M, \ \ \forall \ i \in \{1, \cdots, N^u\}. \tag{8}$$

If the regular learning rate α_t and meta learning rate β_t satisfy $\alpha_t^2 \beta_t < (4M^2 L_0)^{-1}$, then each SGD step of Alg. 1 will lead to the decrease of the validation loss $G(\boldsymbol{\theta})$, regardless of the selected unlabeled examples, *i.e.*,

$$G(\boldsymbol{\theta}_{t+1}) \leq G(\boldsymbol{\theta}_t), \quad \text{for each } t. \tag{9}$$

Convergence Theory — Convergence Rate

Theorem

Assume the same conditions as in Thm. 10, and

$$\inf_t \left(\beta_t - 4\alpha_t^2\beta_t^2 M^2 L_0\right) = D_1 > 0 \quad \text{and} \quad \inf_t \alpha_t = D_2 > 0. \tag{10}$$

We further assume that the unlabeled dataset contains the labeled dataset, *i.e.*, $\mathcal{D}^l \subseteq \mathcal{D}^u$. Then, Alg. 1 achieves $\mathbb{E}\left[\|\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}_t)\|^2\right] \leq \epsilon$ in $O(1/\epsilon^2)$ steps, *i.e.*,

$$\min_{1\leq t\leq T} \mathbb{E}\left[\|\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}_t)\|^2\right] \leq \frac{C}{\sqrt{T}}, \tag{11}$$

where C is a constant independent of the training process.

Trick 1 – First-Order Approximation

Differentiation through the SGD step in Eq. (5) is computationally expensive since the second-order derivative is involved. By applying the chain rule, we have

$$\frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}) = -\frac{\alpha_t}{B^u} \nabla_{\boldsymbol{\theta}}^{\top} \frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \boldsymbol{\theta}_t). \quad (12)$$

Trick 1 – First-Order Approximation

Differentiation through the SGD step in Eq. (5) is computationally expensive since the second-order derivative is involved. By applying the chain rule, we have

$$\frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1}) = -\frac{\alpha_t}{B^u} \nabla_{\boldsymbol{\theta}}^\top \frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \boldsymbol{\theta}_t). \tag{12}$$

Therefore, the gradients of the pseudo-labels can be formulated as

$$\nabla \widetilde{y}_{i,j} = \frac{1}{B^l} \sum_{k=1}^{B^l} \frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \boldsymbol{\theta}_t) = -\frac{\alpha_t}{B^u} \nabla_{\boldsymbol{\theta}}^\top \frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t) \cdot \nabla \boldsymbol{\theta}_t^l, \tag{13}$$

where

$$\nabla \boldsymbol{\theta}_t^l = \frac{1}{B^l} \sum_{i=1}^{B^l} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}_i^l, \boldsymbol{y}_i; \boldsymbol{\theta}_t), \tag{14}$$

Trick 1 – First-Order Approximation

By Taylor expansion,

$$\nabla \widetilde{y}_{i,j} \approx -\frac{\alpha_t}{2B^u\epsilon}\left(\frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t + \epsilon\nabla\boldsymbol{\theta}_t^l) - \frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t - \epsilon\nabla\boldsymbol{\theta}_t^l)\right), \tag{15}$$

where ϵ is a sufficiently small number.

Trick 1 – First-Order Approximation

By Taylor expansion,

$$\nabla \widetilde{y}_{i,j} \approx -\frac{\alpha_t}{2B^u\epsilon}\left(\frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t + \epsilon\nabla\boldsymbol{\theta}_t^l) - \frac{\partial \mathcal{L}}{\partial \widetilde{y}_{i,j}}(\boldsymbol{x}_i^u, \widetilde{\boldsymbol{y}}_i; \boldsymbol{\theta}_t - \epsilon\nabla\boldsymbol{\theta}_t^l)\right), \tag{15}$$

where ϵ is a sufficiently small number.

Furthermore, for the MSE loss $\Phi^{\mathrm{MSE}}(\boldsymbol{p}, \boldsymbol{y}) = \|\boldsymbol{p} - \boldsymbol{y}\|_2^2$, the gradients are given by:

$$\nabla \widetilde{\boldsymbol{y}}_i \approx \frac{\alpha_t}{B^u\epsilon}\left(f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t + \epsilon\nabla\boldsymbol{\theta}_t^l) - f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t - \epsilon\nabla\boldsymbol{\theta}_t^l)\right). \tag{16}$$

(Please refer to our paper for the technical details.)

Trick 2 – Mix-Up Augmentation

Due to the scarcity of labeled data, Mix-Up is a commonly-used technique to augment the training data and regularize the learned model.

Trick 2 – Mix-Up Augmentation

Due to the scarcity of labeled data, Mix-Up is a commonly-used technique to augment the training data and regularize the learned model.

Assuming the labeled data and unlabeled data are of equal batch size, *i.e.*, $B^l = B^u = B$, then we adopt the cross-domain mixup, namely,

$$\begin{aligned} \boldsymbol{x}_i^{\text{in}} &= \lambda_i \boldsymbol{x}_i^l + (1-\lambda_i)\boldsymbol{x}_i^u, \\ \boldsymbol{y}_i^{\text{in}} &= \lambda_i \boldsymbol{y}_i + (1-\lambda_i)\widehat{\boldsymbol{y}}_i, \end{aligned} \quad i = 1, 2, \cdots, B, \tag{17}$$

where $\lambda_1, \lambda_2, \cdots, \lambda_B \overset{\text{i.i.d.}}{\sim} \text{Beta}(\gamma, \gamma)$.

Trick 2 – Mix-Up Augmentation

Due to the scarcity of labeled data, Mix-Up is a commonly-used technique to augment the training data and regularize the learned model.

Assuming the labeled data and unlabeled data are of equal batch size, *i.e.*, $B^l = B^u = B$, then we adopt the cross-domain mixup, namely,

$$\begin{aligned} \boldsymbol{x}_i^{\text{in}} &= \lambda_i \boldsymbol{x}_i^l + (1-\lambda_i)\boldsymbol{x}_i^u, \\ \boldsymbol{y}_i^{\text{in}} &= \lambda_i \boldsymbol{y}_i + (1-\lambda_i)\widehat{\boldsymbol{y}}_i, \end{aligned} \quad i = 1, 2, \cdots, B, \tag{17}$$

where $\lambda_1, \lambda_2, \cdots, \lambda_B \overset{\text{i.i.d.}}{\sim} \text{Beta}(\gamma, \gamma)$.

Note that $\widehat{\boldsymbol{y}}_i$ here must be the *updated* pseudo label in Eq. (6), as the label information is encoded therein.

Overall Algorithm

Algorithm 2 Algorithm with Mix-Up Augmentation.

Input: regular learning rates $\{\alpha_t\}$,
meta learning rates $\{\beta_t\}$

for $t := 1$ *to #iters* **do**

$\{(\boldsymbol{x}_k^l, \boldsymbol{y}_k)\}_{k=1}^B \leftarrow \text{BatchSampler}(\mathcal{D}^l)$

$\{\boldsymbol{x}_i^u\}_{i=1}^B \leftarrow \text{BatchSampler}(\mathcal{D}^u)$

$\widetilde{\boldsymbol{y}}_i = f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t)$

$\mathcal{L}^{\text{KL}}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \boldsymbol{\theta}_t) = \Phi^{\text{KL}}(f(\boldsymbol{x}_k^l; \boldsymbol{\theta}_t), \boldsymbol{y}_k)$

$\nabla \boldsymbol{\theta}_t^l = \frac{1}{B} \sum_{k=1}^B \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{KL}}(\boldsymbol{x}_k^l, \boldsymbol{y}_k; \widetilde{\boldsymbol{\theta}}_{t+1})$

$\epsilon = 0.01 \|\nabla \boldsymbol{\theta}_t^l\|^{-1}$

$\nabla \widetilde{\boldsymbol{y}}_i = \epsilon^{-1} \left(f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t + \epsilon \nabla \boldsymbol{\theta}_t^l) - f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t - \epsilon \nabla \boldsymbol{\theta}_t^l) \right)$

$\widehat{\boldsymbol{y}}_i = \widetilde{\boldsymbol{y}}_i - \beta_t \nabla \widetilde{\boldsymbol{y}}_i$

$\lambda_i \leftarrow \text{Beta}(\gamma, \gamma)$

$\boldsymbol{x}_i^{\text{in}} = \lambda \boldsymbol{x}_i^l + (1 - \lambda) \boldsymbol{x}_i^u$

$\boldsymbol{y}_i^{\text{in}} = \lambda \boldsymbol{y}_i + (1 - \lambda) \widehat{\boldsymbol{y}}_i$

$\mathcal{L}_{\text{cls}}^{\text{KL}}(\boldsymbol{x}_i^{\text{in}}, \boldsymbol{y}_i^{\text{in}}; \boldsymbol{\theta}_t) = \Phi^{\text{KL}}(f(\boldsymbol{x}_i^{\text{in}}; \boldsymbol{\theta}_t), \boldsymbol{y}_i^{\text{in}}$

$\mathcal{L}_{\text{cons}}^{\text{MSE}}(\boldsymbol{x}_i^u, \widehat{\boldsymbol{y}}_i; \boldsymbol{\theta}_t) = \Phi^{\text{MSE}}(f(\boldsymbol{x}_i^u; \boldsymbol{\theta}_t), \widehat{\boldsymbol{y}}_i)$

$\nabla \widehat{\boldsymbol{\theta}}_t = \frac{1}{B} \sum_{i=1}^B \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{cls}}^{\text{KL}} + \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{cons}}^{\text{MSE}} \right)$

$\boldsymbol{\theta}_{t+1} = \text{Optimizer}(\boldsymbol{\theta}_t, \nabla \widehat{\boldsymbol{\theta}}_t, \alpha_t)$

end

Experiments on Small Datasets

Experiments on **SVHN**, **CIFAR-10**, and **CIFAR-100**.

Method	SVHN	CIFAR-10	CIFAR-100
Π-Model [7]	4.82%	12.36%	39.19%
TE [7]	4.42%	12.16%	38.65%
MT [9]	3.95%	12.31%	-
MT+SNTG [26]	3.86%	10.93%	-
VAT [8]	5.42%	11.36%	-
VAT+Ent [8]	3.86%	10.55%	-
VAT+Ent+SNTG [26]	3.83%	9.89%	-
VAT+VAdD [27]	3.55%	9.22%	-
MA-DNN [28]	4.21%	11.91%	34.51%
Co-training [29]	3.29%	8.35%	34.63%
MT+fastSWA [10]	-	9.05%	33.62%
TNAR-VAE [11]	3.74%	8.85%	-
ADA-Net [24]	4.62%	10.30%	-
ADA-Net+fastSWA [24]	-	8.72%	-
DualStudent [30]	-	8.89%	32.77%
Ours	**3.15%**	**7.78%**	**30.74%**
Fully-Supervised	2.67%	4.88%	22.10%

Figure: Semi-supervised classification results.

Experiments on ImageNet

Method	Top-1	Top-5
Labeled-Only	53.65%	31.01%
MT [9]	49.07%	23.59%
Co-training [29]	46.50%	22.73%
ADA-Net [24]	44.91%	21.18%
Ours	**44.87%**	**18.88%**
Fully-Supervised	29.15%	10.12%

Figure: Semi-supervised classification results on ImageNet.

Experiments on ImageNet

Method	Top-1	Top-5
Labeled-Only	53.65%	31.01%
MT [9]	49.07%	23.59%
Co-training [29]	46.50%	22.73%
ADA-Net [24]	44.91%	21.18%
Ours	**44.87%**	**18.88%**
Fully-Supervised	29.15%	10.12%

Figure: Semi-supervised classification results on ImageNet.

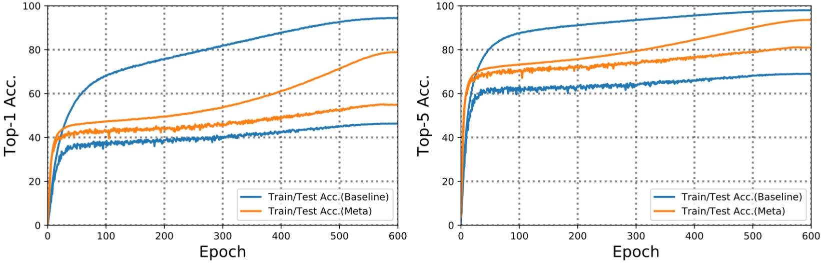

Figure: Accuracy curves of the baseline method and the meta-learning algorithm.

Ablation Studies and Visualization

No.	Meta	Mix-Up	SVHN	CIFAR-10	CIFAR-100
1			9.76%	15.43%	38.74%
2	✓		3.68%	11.63%	35.40%
3		✓	5.60%	11.10%	32.67%
4	✓	✓	**3.15%**	**7.78%**	**30.74%**

Figure: Ablation of each component.

Ablation Studies and Visualization

No.	Meta	Mix-Up	SVHN	CIFAR-10	CIFAR-100
1			9.76%	15.43%	38.74%
2	✓		3.68%	11.63%	35.40%
3		✓	5.60%	11.10%	32.67%
4	✓	✓	**3.15%**	**7.78%**	**30.74%**

Figure: Ablation of each component.

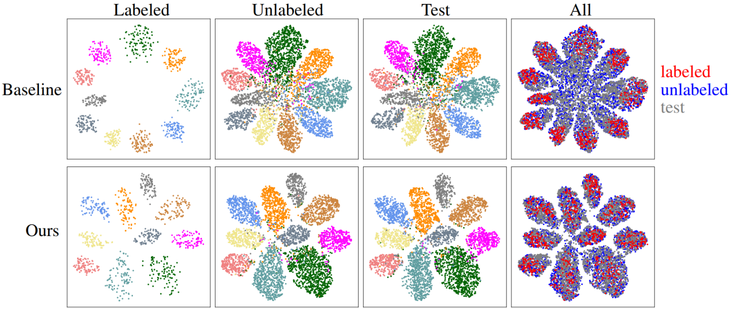

Figure: Feature Visualization.

Future Perspective

(i) Incorporation with the concurrent self-supervised SSL methods, *e.g.*, FixMatch[4];

(ii) Extension beyond semi-supervised classification, *e.g.*, weakly-supervised segmentation;

(iii) (May be too aggressive) Generalize the current bi-level optimization into multi-level optimization.

[4] Sohn *et al.* FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *NeurIPS*, 2020.

Thanks!