Two ways to improve FixMatch Performance

From Pseudo-Labels Balancing to Dynamic Threshold

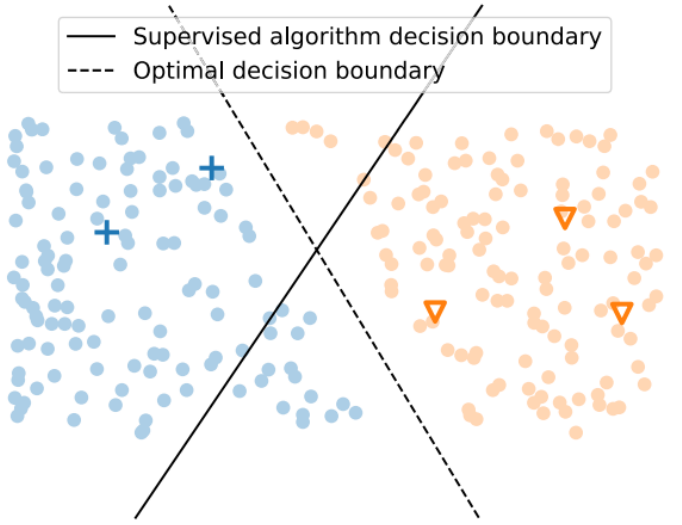
Background

Consistency Regularization(Smoothness)

The model's output should remain unchanged when the input is perturbed.

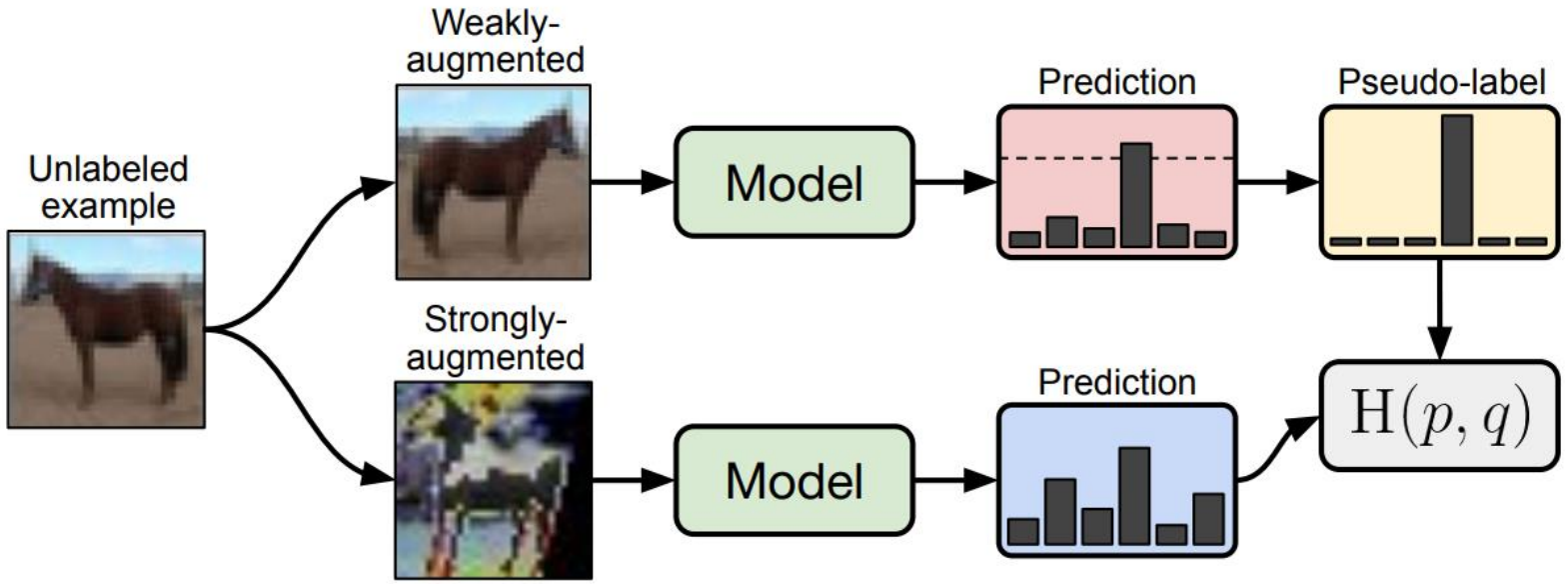
Entropy Minimization(Low-density)

The classifier's decision boundary should not pass through high-density regions of the marginal data distribution.



(a) Smoothness and low-density assumptions.

Background



$$\ell_s = \frac{1}{B}\sum_{b=1}^{B} \mathrm{H}(p_b, p_\mathrm{m}(y \mid \alpha(x_b)))$$

$$\ell_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau)\, \mathrm{H}(\hat{q}_b, p_\mathrm{m}(y \mid \mathcal{A}(u_b)))$$

Weakly augmented

Strongly augmented

模式分析与机器智能
工业和信息化部重点实验室
MIIT Key Laboratory of
Pattern Analysis & Machine Intelligence

ParNeC 模式识别与神经计算研究组
PAttern Recognition and NEural Computing

Debiased Learning from Naturally Imbalanced Pseudo-Labels

Xudong Wang[1] Zhirong Wu[2] Long Lian[1] Stella X. Yu[1]
[1]UC Berkeley / ICSI [2]Microsoft Research

CVPR 2022

Motivation

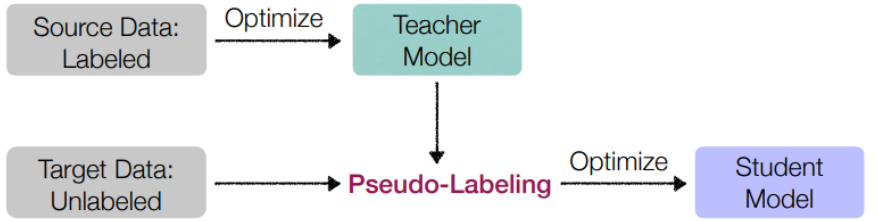
Even when networks are trained on **balanced** data, the pseudo-labels are still highly **class-imbalanced**.

Biases in Semi-supervised Learning

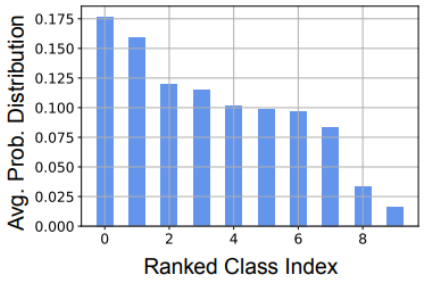
A student model will inherit the implicitly **imbalanced pseudo-labels** and, in turn, reinforces the teacher model's biases. Once confusing samples are wrongly pseudo-labeled, the mistake is almost impossible to be self-corrected.

But where exactly do the biases come from?

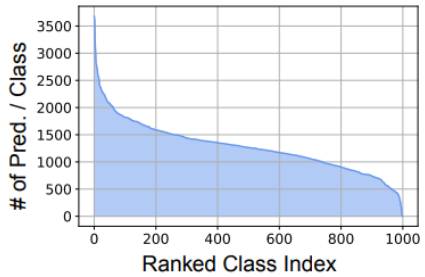
The blame for the pseudo-label bias can be largely attributed to inter-class confounding.



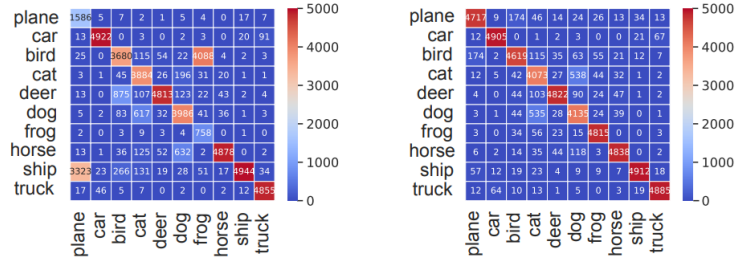
(a) Framework of pseudo-labeling



(b) FixMatch on CIFAR10 SSL



(c) CLIP on ImageNet ZSL



(a) FixMatch (b) DebiasPL

Figure 6. The cause for pseudo-label biases can be partially attributed to inter-class confounding. For example, FixMatch often misclassifies "ship" as "plane". The confusion matrix of FixMatch's and our DebiasPL's pseudo-labels are visualized.

Prerequisite Knowledge

NeurIPS 2020

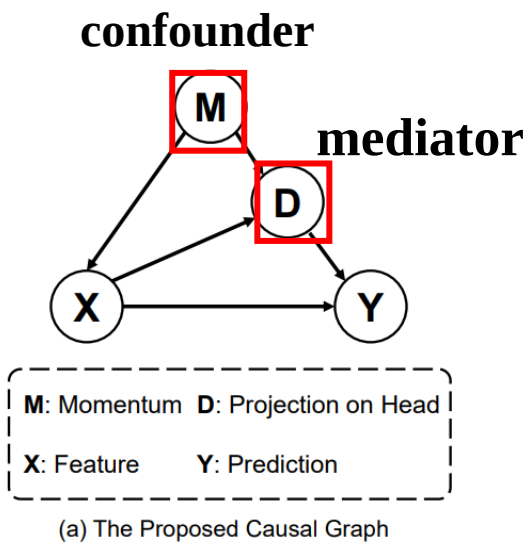
Long-Tailed Classification by Keeping the Good and Removing the Bad Momentum Causal Effect

Kaihua Tang[1], **Jianqiang Huang**[1,2], **Hanwang Zhang**[1]
[1]Nanyang Technological University, [2]Damo Academy, Alibaba Group
kaihua001@e.ntu.edu.sg, jianqiang.jqh@gmail.com, hanwangzhang@ntu.edu.sg

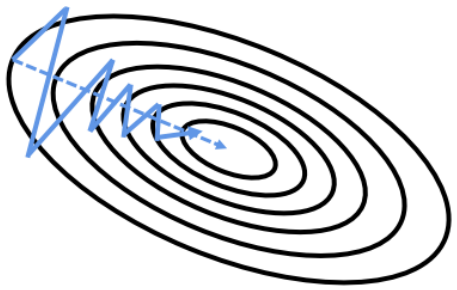
This paper establishes a **causal inference** framework and provides a fundamental theory for reweighting/re-sampling heuristics.

$X \leftarrow M \rightarrow D \rightarrow Y$ causes the spurious correlation even if X has nothing to do with the predicted Y.

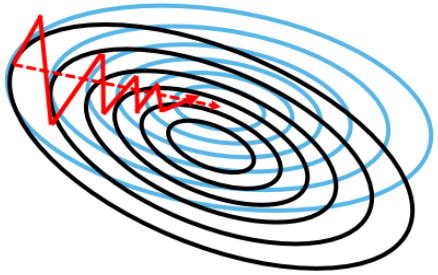
$X \rightarrow D \rightarrow Y$ respects the inter-relationships of the semantic concepts in classification.



(a) The Proposed Causal Graph



SGD Momentum in ***Balanced*** Dataset



SGD Momentum in ***Long-Tailed*** Dataset

Prerequisite Knowledge

We need the **direct** causal effect along path $X \rightarrow Y$.

Total Direct Effect (TDE)

$$\arg\max_{i \in C} \; TDE(Y_i) = [Y_{\boldsymbol{d}} = i | do(X = \boldsymbol{x})] - [Y_{\boldsymbol{d}} = i | do(X = \boldsymbol{x}_0)]$$

$P(Y = y|X = x)$ is the probability that $Y = y$ conditional on finding $X = x$, while $P(Y = y|do(X = x))$ is the probability that $Y = y$ when we intervene to make $X = x$.

Removes the "bad" **confounder** bias while keeps the "good" **mediator** bias

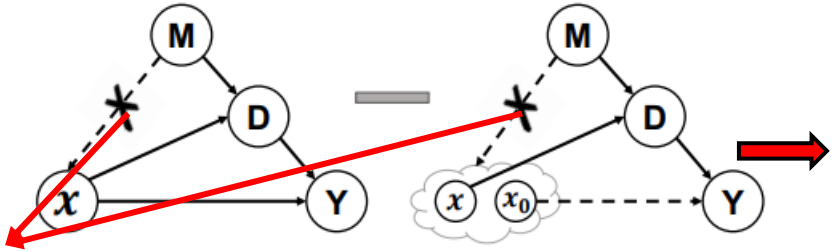


Figure 3: The TDE inference (Eq. (2)) for the long-tailed classification after de-confounded training. Subtracted left: $[Y_{\boldsymbol{d}} = i|do(X = \boldsymbol{x})]$, minus right: $[Y_{\boldsymbol{d}} = i|do(X = \boldsymbol{x}_0)]$.

drug → cure ⟹ 0

drug → placebo → cure

Controlled Direct Effect (CDE)

$$\text{CDE}(Y_i) = [Y_i|do(A_i), do(D)] - [Y_i|do(\hat{A}), do(D)]$$

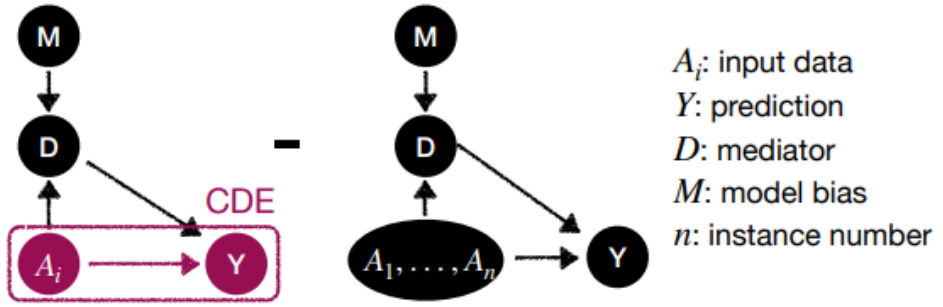


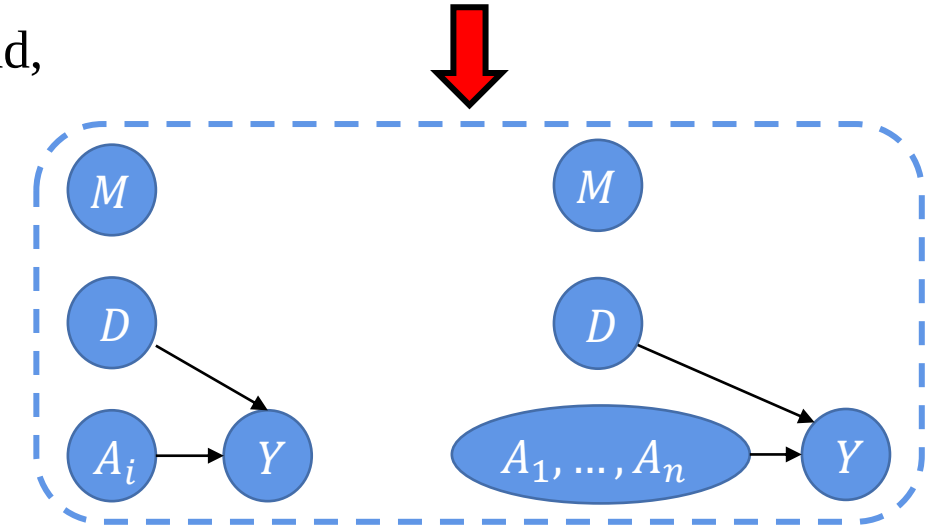
Figure 8. Causal graph of debiasing with counterfactual reasoning.

Measuring the counterfactual outcome via visiting all training samples is significantly **computational expensive**.

Use **Approximated Controlled Direct Effect** (ACDE) instead, which assumes that the model bias is not drastically changed.

$$[Y_i|do(\hat{A}), do(D)] \Rightarrow \hat{p} \leftarrow m\hat{p} + (1-m)\frac{1}{\mu B}\sum_{k=1}^{\mu B} p_k$$

Probability distribution for instance $\alpha(x_k)$ obtained via a softmax function



Method

Weakly augmented　　Strongly augmented

$$\mathcal{L}_{\mathrm{u}} = \frac{1}{\mu B}\sum_{i=1}^{\mu B} \mathbb{1}[\max(p(\alpha(x_i))) \geq \tau] \cdot \mathrm{H}(\hat{y}_i, p(\beta(x_i)))$$

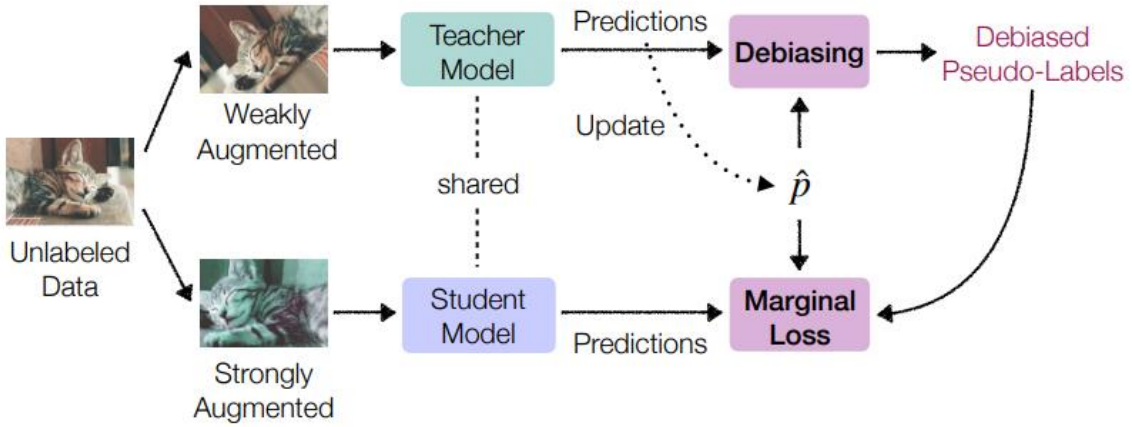
$$\tilde{f}_i = f(\alpha(x_i)) - \lambda \log \hat{p}$$

debias factor

$$\mathcal{L}_{\mathrm{AML}} = -\log \frac{e^{(z_{\hat{y}_i} - \Delta_{\hat{y}_i})}}{e^{(z_{\hat{y}_i} - \Delta_{\hat{y}_i})} + \sum_{k \neq \hat{y}_i}^{C} e^{(z_k - \Delta_k)}}$$

Demand a larger margin between hardly biased and highly biased classes

where $f(\alpha(\cdot))$ refers to logits of weakly-augmented unlabeled instance, $\Delta_j = \lambda \log(\frac{1}{\hat{p}_j})$ for $j \in \{1, ..., C\}$
$z = f(\beta(x_i))$



Method

Algorithm 1: PyTorch-style pseudocode for semi-supervised learning with DebiasPL

```
# initialize p_hat with 1/C, C is the number of classes
p_hat = torch.ones([1, C]) / C
# load a batch with unlabeled and labeled samples
# x:  labeled samples ; target:  labels for x ; u:  unlabeled samples
for (x, target), u in loader:
    # augment x with weak augmentation and get two versions of u with strong and weak augmentations
    x, u_s, u_w = weak(x), strong(u), weak(u)
    # model forward
    l_x, l_us, l_uw = model(x, u_s, u_w)

    # get debiased pseudo-labels
    p_uw = F.softmax(l_uw - tau * torch.log(p_hat), dim=1)
    max_probs, pseudo_label = torch.max(p_uw, dim=-1)

    # get mask for filtering instances with low confidence score
    mask = max_probs.ge(thresh).float()
    # update p_hat
    p_hat = momentum * p_hat + (1 - momentum) * p_uw.detach().mean(dim=0)

    # calculate loss_x for labeled instances
    loss_x = F.cross_entropy(l_x, target)
    # calculate marginal loss loss_u for unlabeled instances
    l_us = l_us + lambda * torch.log(p_hat)
    loss_u = (F.cross_entropy(l_us, pseudo_label, reduction='none') * mask).mean()
    # total loss
    loss = loss_x + lambda_u * loss_u

    # optimization step
    loss.backward()
    optimizer.step()
# update the ema model
model.momentum_update_ema()
```

Experiment

Method	CIFAR10-LT: # of labels (percentage)				CIFAR10: # of labels (percentage)		
	γ=100		γ=200				
	1244 (10%)	3726 (30%)	1125 (10%)	3365 (30%)	40 (0.08%)	80 (0.16%)	250 (2%)
UDA [68] §	-	-	-	-	71.0 ±6.0	-	91.2 ±1.1
MixMatch [5] §	60.4 ±2.2	-	54.5 ±1.9	-	51.9 ±11.8	80.8 ±1.3	89.0 ±0.9
CReST w/ DA [67]	75.9 ±0.6	77.6 ±0.9	64.1 ±0.22	67.7 ±0.8	-	-	-
CReST+ w/ DA [67]	78.1 ±0.8	79.2 ±0.2	67.7 ±1.4	70.5 ±0.6	-	-	-
CoMatch w/ SimCLR [12, 32]	-	-	-	-	92.6 ±1.0	94.0 ±0.3	95.1 ±0.3
FixMatch [57] §	67.3 ±1.2	73.1 ±0.6	59.7 ±0.6	67.7 ±0.8	86.1 ±3.5	92.1 ±0.9	94.9 ±0.7
FixMatch w/ DA w/ LA [4, 38, 57, 67] §	70.4 ±2.9	-	62.4 ±1.2	-	-	-	-
FixMatch w/ DA w/ SimCLR [4, 12, 57] §	-	-	-	-	89.7 ±4.6	93.3 ±0.5	94.9 ±0.7
DebiasPL (w/ FixMatch)	**79.2 ±1.0**	**80.6 ±0.5**	**71.4 ±2.0**	**74.1 ±0.6**	**94.6 ±1.3**	**95.2 ±0.1**	**95.4 ±0.1**
gains over the best FixMatch variant	**+8.8**	**+7.5**	**+9.0**	**+6.4**	**+4.9**	**+1.9**	**+0.5**

Table 2. Without any prior knowledge of the marginal class distribution of unlabeled/labeled data, the performance of DebiasPL on *both* **CIFAR *and* CIFAR-LT SSL benchmarks** surpasses previous SOTAs, which are *either designed for balanced data or meticulously tuned for long-tailed data*. DibasMatch is experimented with the same set of hyper-parameters across all benchmarks. § states the best-reported results of counterpart methods, copied from [32], [57] or [67]. γ: imbalance ratio. We report results averaged on 5 different folds.

Experiment

Method	B.S.	#epochs	Pre-train	1%		0.2%	
				top-1	top-5	top-1	top-5
FixMatch w/ DA [4, 57]	4096	400	✗	53.4	74.4	-	-
FixMatch w/ DA [4, 57]	4096	400	✓	59.9	79.8	-	-
FixMatch w/ EMAN [9, 57]	384	50	✓	60.9	82.5	43.6*	64.6*
DebiasPL w/ FixMatch	384	50	✓	**63.1** (+2.2)	**83.6** (+1.1)	**47.9** (+3.7)	**69.6** (+5.0)
DebiasPL (multi-views)	768	50	✓	**65.3** (+4.4)	**85.2** (+2.7)	**51.6** (+8.0)	**73.3** (+8.7)
DebiasPL (multi-views)	768	200	✓	**66.5** (+5.6)	**85.6** (+3.1)	**52.3** (+8.7)	**73.5** (+8.9)
DebiasPL (multi-views)	1536	300	✓	**67.1** (+6.2)	**85.8** (+3.3)	-	-
DebiasPL w/ CLIP [49]	384	50	✓	**69.1** (+8.2)	**89.1** (+6.6)	**68.2** (+24.6)	**88.2** (+23.6)
DebiasPL w/ CLIP (multi-views) [49]	768	50	✓	**70.9** (+10.0)	**89.3** (+6.8)	**69.6** (+26.0)	**88.4** (+23.8)
CLIP (few-shot) [49, 73]	256	50	✓	53.4	-	40.0	-
SwAV [11]	4096	50	✓	53.9	78.5	-	-
SimCLRv2 (+ Self-distillation) [13]	4096	400	✓	60.0	79.8	-	-
PAWS (multi-crops) † [2]	4096	50	✓	66.5	-	-	-
CoMatch (multi-views) [32]	1440	400	✓	67.1	87.1	-	-

Table 3. DebiasPL delivers state-of-the-arts results on **ImageNet-1K semi-supervised learning** with various fractions of labeling samples, especially for extremely low-shot settings. All results are produced with a backbone of ResNet-50. †: unsupervised pre-trained for 800 epochs, except for PAWS [2], which is pre-trained for 300 epochs with pseudo-labels generated non-parametrically. *: reproduced.

Experiment

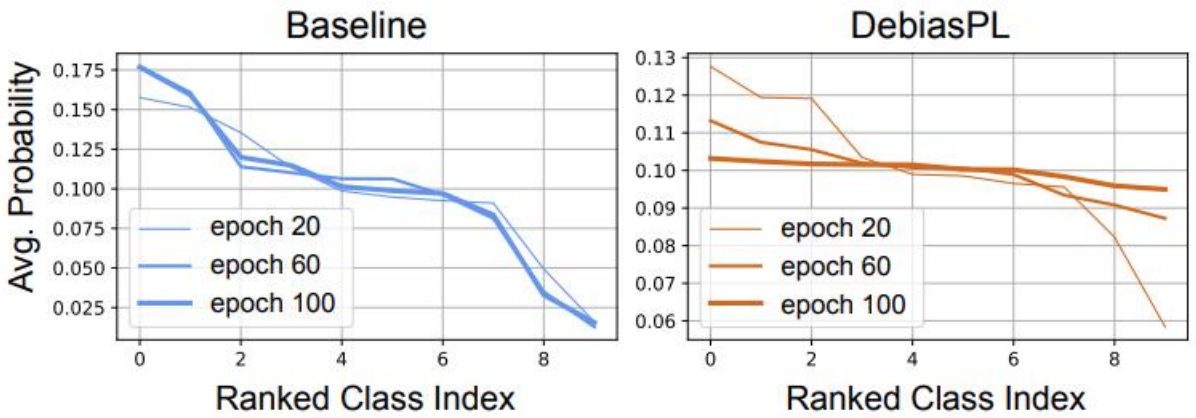


Figure 2. FixMatch's pseudo-labels are highly imbalanced across different training stages, even though the unlabeled and labeled data it trains on is class-balanced. In contrast, DebiasPL produces nearly balanced pseudo-labels at late stages. The probability distributions of FixMatch and DebiasPL are averaged over all unlabeled data. The class indices are sorted by average probability. We conduct experiments on CIFAR10 with 4 labeled instances per class.

Experiment

Method	Labeled: **LT**; 10% labeled, $\gamma = 200$	
	Unlabeled: **LT**	Unlabeled: **Balanced**
FixMatch [57]	62.3 ±1.6	72.1 ±2.3
DebiasPL	**71.4** ±**2.0** (+9.1)	**83.5** ±**2.4** (+11.4)

Table 4. DebiasPL consistently improves the performance of SSL when the unlabeled data is either the sames as labeled data, i.e., long-tailed distributed, or different with labeled data, i.e., balanced distributed across semantics. We report results averaged on 5 folds.

	FixMatch	MixMatch	UDA
Baseline	89.7 ± 4.6	47.5 ± 11.5	29.1 ± 5.9
+ DebiasPL	**94.6** ± **1.3**	**61.7** ± **6.1**	**43.2** ± **5.2**

Table 5. **DebiasPL is a universal add-on.** Top-1 accuracies of various SSL methods on CIFAR10, averaged on 5 folds, are compared. 4 instances per class are labeled.

Debiasing	Magirnal Loss	CIFAR10	CIFAR10-LT
		86.1	73.5
✓		93.3	79.6
✓	✓	94.6	80.6

Table 7. **Ablation study** on the **contribution of each component** of DebiasPL. Experimented on CIFAR10 and CIFAR10-LT ($\gamma =$ 100) SSL, in which 4 out of 5,000 samples are labeled per class for CIFAR10 and 30% instances are labeled for CIFAR10-LT. Results averaged over 5 different folds are reported.

by introducing the marginal loss is relatively smaller than the unbalanced benchmark.

λ	0.0	0.25	0.5	0.75	1.0	2.0
DebiasPL	73.5	79.5	80.6	80.5	80.5	77.7

Table 8. **Ablation study** on CIFAR10-LT ($\gamma = 100$) semi-supervised learning with DebiasPL under various **weight** λ of debiasing module and marginal loss. 30% samples are labeled. The model is identical to FixMatch when $\lambda = 0$. Results averaged over 5 different folds are reported.

模式分析与机器智能
工业和信息化部重点实验室
MIIT Key Laboratory of
Pattern Analysis & Machine Intelligence

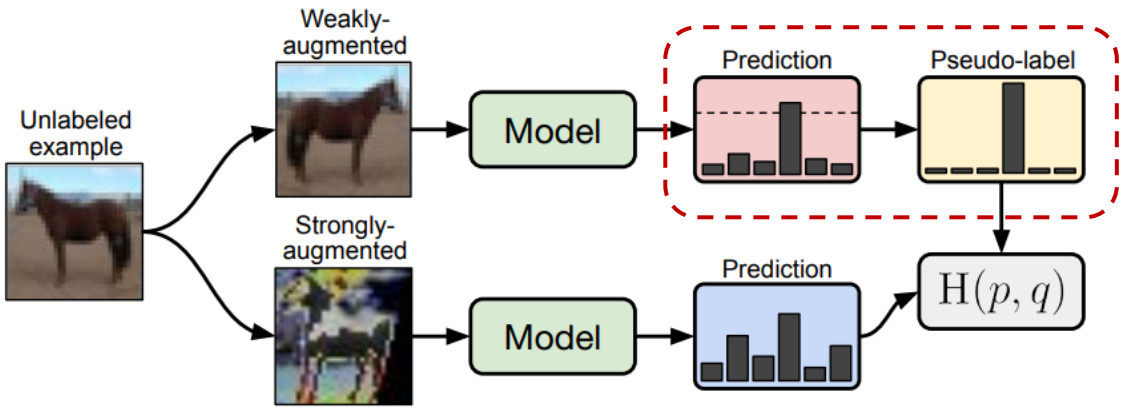
ParNeC 模式识别与神经计算研究组
PAttern Recognition and NEural Computing

Dash: Semi-Supervised Learning with Dynamic Thresholding

Yi Xu ✉[1] Lei Shang[1] Jinxing Ye[1] Qi Qian[1] Yu-Feng Li ✉[2] Baigui Sun[1] Hao Li[1] Rong Jin[1]

PMLR 2021

Motivation



$$\frac{1}{\mu B}\sum_{b=1}^{\mu B}\mathbb{1}(\max(q_b) \geq \tau)\,\mathrm{H}(\hat{q}_b, q_b)$$

τ is a fixed threshold(0.95)

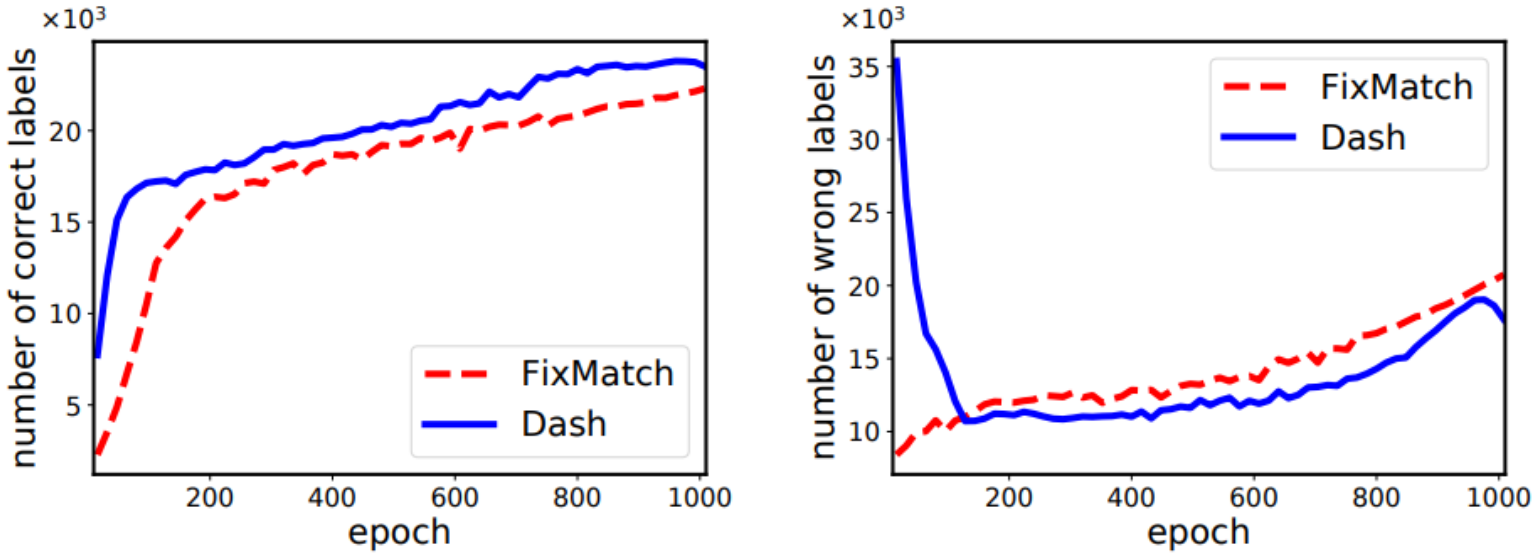
Figure 1: Diagram of FixMatch. A weakly-augmented image (top) is fed into the model to obtain predictions (red box). When the model assigns a probability to any class which is above a threshold (dotted line), the prediction is converted to a one-hot pseudo-label. Then, we compute the model's prediction for a strong augmentation of the same image (bottom). The model is trained to make its prediction on the strongly-augmented version match the pseudo-label via a cross-entropy loss.

Either use **all unlabeled examples** or the unlabeled examples with a **fixed high-confidence prediction** during the training progress.

→

There are too many **correct** pseudo labeled examples **eliminated**, and too many **wrong** pseudo labeled examples **selected**.

Motivation



(a) Number of selected unlabeled examples with correct pseudo labels

(b) Number of selected unlabeled examples with wrong pseudo labels

Figure 1. An example of experimental results on Wide ResNet-28-8 for CIFAR-100 with 400 labeled images illustrates the reason of dynamically selecting unlabeled data to train learning models. Pseudo labels are generated based on the prediction models. FixMatch selects unlabeled example if its confidence prediction is greater than 0.95, while the proposed Dash algorithm selects unlabeled example based on a dynamic threshold through optimization iterations. (a) The proposed Dash selects more examples with correct pseudo labels than that of FixMatch. (b) The proposed Dash maintains much more examples with wrong pseudo labels at the beginning but it will drop off more examples with wrong pseudo labels after several epochs, comparing to FixMatch.

Can we design a provable SSL algorithm
that selects unlabeled data with dynamic thresholding?

Method

Use **x** to denote the feature, **y** to denote the label. For simplicity, let ξ denote the input-label pair **(x, y)**

$$\xi := (\mathbf{x}, \mathbf{y})$$

We denote by P the underlying distribution of data pair ξ , then $\xi \sim \mathcal{P}$. The goal is to learn a model $\mathbf{w} \in \mathbb{R}^d$ via minimizing an optimization problem whose objective function $F(\mathbf{w})$ is the expectation of random loss function $f(\mathbf{w}; \xi)$

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) := \mathrm{E}_{\xi\sim\mathcal{P}}\left[f\left(\mathbf{w};\xi\right)\right] \quad f(\mathbf{w};\xi_i) = H(\mathbf{y}_i, \mathbf{p}(\mathbf{w};\mathbf{x}_i)) := \sum_{k=1}^{K} -y_{i,k} \log\left(\frac{\exp(p_k(\mathbf{w};\mathbf{x}_i))}{\sum_{j=1}^{K}\exp(p_j(\mathbf{w};\mathbf{x}_i))}\right)$$

Labeled examples $\mathbf{D}_l := \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, N_l\}$ Unlabeled training examples $\mathbf{D}_u := \{(\mathbf{x}_i^u, \widehat{\mathbf{y}}_i^u), i = 1, \ldots, N_u\}$

Supervised loss $F_s(\mathbf{w}) := \frac{1}{N_l}\sum_{i=1}^{N_l} f(\mathbf{w};\xi_i)$

Threshold on prediction or loss?

With supervised model $\boldsymbol{w}$ and weak augmentation α, we can get the class prediction for a weakly-augmented unlabeled sample $\boldsymbol{x_i^u}$

$$\mathbf{h}_i = \mathbf{p}(\mathbf{w}, \alpha(\mathbf{x}_i^u))$$

It creates a pseudo label by

$$\widehat{\mathbf{y}}_i^u = \arg\max(\mathbf{h}_i)$$

Since the new dynamic threshold is not fixed, we let it rely on the optimization iteration t and it is denoted by ρ_t. Then the unsupervised loss is given by

Loss function

$$F_u(\mathbf{w}) = \frac{1}{N_u}\sum_{i=1}^{N_u} I(f_u(\mathbf{w};\xi_i^u) \leq \rho_t) f_u(\mathbf{w};\xi_i^u)$$

$$H(\widehat{\mathbf{y}}_i^u, \mathbf{p}(\mathbf{w};\mathcal{T}(\mathbf{x}_i^u)))$$

Prediction

$$F_u(\mathbf{w}) = \frac{1}{N_u}\sum_{i=1}^{N_u} I(\max(\mathbf{h}_i) \geq \tau) H(\widehat{\mathbf{y}}_i^u, \mathbf{p}(\mathbf{w}, \mathcal{T}(\mathbf{x}_i^u)))$$

$\mathcal{T}(x)$ is the strongly-augmented version of x

Method

$$F_u(\mathbf{w}) = \frac{1}{N_u}\sum_{i=1}^{N_u} I(f_u(\mathbf{w};\xi_i^u) \le \rho_t) f_u(\mathbf{w};\xi_i^u)$$

Loss function

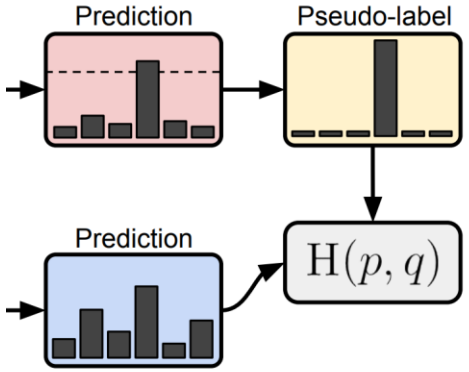
$$H(\widehat{\mathbf{y}}_i^u, \mathbf{p}(\mathbf{w};\mathcal{T}(\mathbf{x}_i^u)))$$

$$F_u(\mathbf{w}) = \frac{1}{N_u}\sum_{i=1}^{N_u} I(\max(\mathbf{h}_i) \ge \tau) H(\widehat{\mathbf{y}}_i^u, \mathbf{p}(\mathbf{w},\mathcal{T}(\mathbf{x}_i^u)))$$

Prediction

Why threshold on loss is better than on prediction?

Threshold on prediction **only** contains the information of **weakly** augmented samples, while threshold on loss function includes **both strongly** and **weakly** augmented samples.

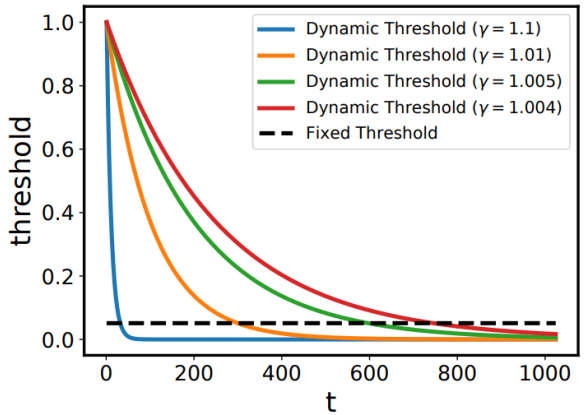


How to set dynamic threshold ρ_t?

$$\rho_t := C\gamma^{-(t-1)}\widehat{\rho}$$

Where $C > 1, \gamma > 1$ are two constants. $\widehat{\rho}$ can be determined by averaged loss of labeled examples during warm-up stage.

$$\widehat{\rho} \approx \frac{1}{|\mathbf{D}_l|}\sum_{\xi_i \in \mathbf{D}_l} f(\mathbf{w}_1;\xi_i)$$



Algorithm 1 Dash: Semi-Supervised Learning with **D**yna**m**ic Thre**sh**olding

Input: learning rate η_0 and mini-batch size m_0 for stage one, learning rate η and parameter m of mini-batch size for stage two, two parameters $C > 1$ and $\gamma > 1$ for computing threshold, and violation probability δ.

// Warm-up Stage: run SGD in T_0 iterations.

Initialization: $\mathbf{u}_0 = \mathbf{w}_0$

for $t = 0, 1, \ldots, T_0 - 1$ **do**

Sample m_0 examples $\xi_{t,i}$ $(i = 1, \ldots, m_0)$ from $\mathbf{D}_l$,

$\mathbf{u}_{t+1} = \mathbf{u}_t - \eta_0 \tilde{\mathbf{g}}_t$ where $\tilde{\mathbf{g}}_t = \frac{1}{m_0} \sum_{i=1}^{m_0} \nabla f_s(\mathbf{u}_t; \xi_{t,i})$

end for

// Selection Stage: run SGD in T iterations.

Initialization: $\mathbf{w}_1 = \mathbf{u}_{T_0}$.

Compute the value of $\widehat{\rho}$ as in (16). // In practice, $\widehat{\rho}$ can be obtained as in (17).

for $t = 1, \ldots, T$ **do**

1) Sample $n_t = m\gamma^{t-1}$ examples from $\mathbf{D}_u$, where the pseudo labels in $\mathbf{D}_u$ are generated by FixMatch

2) Set the threshold $\rho_t = C\gamma^{-(t-1)}\widehat{\rho}$.

3) Compute truncated stochastic gradient $\mathbf{g}_t$ as (18).

4) Update solution by SGD using stochastic gradient $\mathbf{g}_t$ and learning rate η: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$.

end for

Output: $\mathbf{w}_{T+1}$

Warm up stage and selection stage

Experiment

Table 1. Comparison of top-1 testing error rates for different methods using Wide ResNet-28-2 for CIFAR-10, Wide ResNet-28-8 for CIFAR-100 (in %, mean ± standard deviation).

	CIFAR-10			CIFAR-100		
Algorithm	40 labels	250 labels	4000 labels	400 labels	2500 labels	10000 labels
Π-Model	-	54.26±3.97	14.01±0.38	-	57.25±0.48	37.88±0.11
Pseudo-Labeling	-	49.78±0.43	16.09±0.28	-	57.38±0.46	36.21±0.19
Mean Teacher	-	32.32±2.30	9.19±0.19	-	53.91±0.57	35.83±0.24
MixMatch	47.54±11.50	11.05±0.86	6.42±0.10	67.61±1.32	39.94±0.37	28.31±0.33
UDA	29.05±5.93	8.82±1.08	4.88±0.18	59.28±0.88	33.13±0.22	24.50±0.25
ReMixMatch	19.10±9.64	5.44±0.05	4.72±0.13	**44.28**±2.06	27.43±0.31	23.03±0.56
RYS (UDA)	-	5.53±0.17	4.75±0.28	-	-	-
RYS (FixMatch)	-	5.05±0.12	4.35±0.06	-	-	-
FixMatch (CTA)	11.39±3.35	5.07±0.33	4.31±0.15	49.95±3.01	28.64±0.24	23.18±0.11
Dash (CTA, ours)	**9.16**±4.31	4.78±0.12	4.13±0.06	44.83±1.36	27.85±0.19	22.77±0.21
FixMatch (RA)	13.81±3.37	5.07±0.65	4.26±0.05	48.85±1.75	28.29±0.11	22.60±0.12
Dash (RA, ours)	13.22±3.75	**4.56**±0.13	**4.08**±0.06	44.76±0.96	**27.18**±0.21	**21.97**±0.14

Experiment

Table 2. Comparison of top-1 testing error rates for different methods using Wide ResNet-28-2 for SVHN and Wide ResNet-37-2 for STL-10 (in %, mean ± standard deviation).

	SVHN			STL-10
Algorithm	40 labels	250 labels	1000 labels	1000 labels
Π-Model	-	18.96±1.92	7.54±0.36	26.23±0.82
Pseudo-Labeling	-	20.21±1.09	9.94±0.61	27.99±0.83
Mean Teacher	-	3.57±0.11	3.42±0.07	21.43±2.39
MixMatch	42.55±14.53	3.98±0.23	3.50±0.28	10.41±0.61
UDA	52.63±20.51	5.69±2.76	2.46±0.24	7.66±0.56
ReMixMatch	3.34±0.20	2.92±0.48	2.65±0.08	5.23±0.45
RYS (UDA)	-	2.45±0.08	2.32±0.06	-
RYS (FixMatch)	-	2.63±0.23	2.34±0.15	-
FixMatch (CTA)	7.65±7.65	2.64±0.64	2.36±0.19	5.17±0.63
Dash (CTA, ours)	3.14±1.60	2.38±0.29	2.14±0.09	**3.96**±0.25
FixMatch (RA)	3.96±2.17	2.48±0.38	2.28±0.11	7.98±1.50
Dash (RA, ours)	**3.03**±1.59	**2.17**±0.10	**2.03**±0.06	7.26±0.40

Dash has large improvement when the labeled examples is small

Ablation

Table 3. Comparison of top-1 testing error rates for different values of γ on CIFAR-10 (in %).

γ	1.01	1.1	1.2	1.3
250 labels	4.85	**4.76**	4.99	4.82
4000 labels	4.39	4.28	**4.11**	4.31

Table 4. Comparison of top-1 testing error rates for PL and Dash with PL on CIFAR-10 (in %).

Algorithm	PL	Dash-PL
250 labels	49.78	**46.90**
4000 labels	16.09	**15.59**

Pseudo-Labeling

Thanks