SOTA deep CNN architectures and their principles:

Meta Pseudo Labels (CVPR2021)

Artificial Intelligence

Creating the Future

Dong-A University

Division of Computer Engineering &
Artificial Intelligence

9 Meta Pseudo Labels

Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, Quoc V. Le, “Meta Pseudo Labels,” arXiv:2003.10580, 2020, **CVPR2021** (**Google AI, Brain Team**)

Official (tensorflow) : https://github.com/google-research/google-research/tree/master/meta_pseudo_labels
pytorch : https://github.com/kekmodel/MPL-pytorch

Ref : https://kmhana.tistory.com/33

PR-336: Meta Pseudo Labels, https://www.youtube.com/watch?v=UZMDIHPVVjY

Abstract

- Present Meta Pseudo Labels, a **semi-supervised learning method** that achieves a new state-of-the-art **top-1 accuracy of 90.2% on ImageNet**, which is 1.6% better than the existing state-of-the-art [16].
- Like Pseudo Labels, Meta Pseudo Labels has **a teacher network to generate pseudo labels on unlabeled data to teach a student network.**
- However, unlike Pseudo Labels where the teacher is fixed, **the teacher in Meta Pseudo Labels is constantly adapted by the feedback of the student’s performance on the labeled dataset.**
- As a result, **the teacher generates better pseudo labels to teach the student.**

Pseudo Labels (Or Self-training)

1) Teacher generates pseudo labels on unlabeled images.
2) These pseudo labeled images are then combined with labeled images to train the student.
3) The student learns to become better than the teacher, thanks to the abundance of pseudo labeled data and the regularization such as data augmentation

- Main drawback : **Confirmation bias** in pseudo-labeling – If the pseudo labels are inaccurate, the student will learn from inaccurate data.

Opinion
- Student를 가르치는 Teacher, 그 Teacher을 발전시키는 Student
- Semi-Supervised Learning의 가능성을 여김 없이 보여준 논문
- 엄청난 모델 크기를 자랑하고 있는 Transformer 계열 모델 사이에서, 상대적으로 매우 작은 모델로 고성능 기록

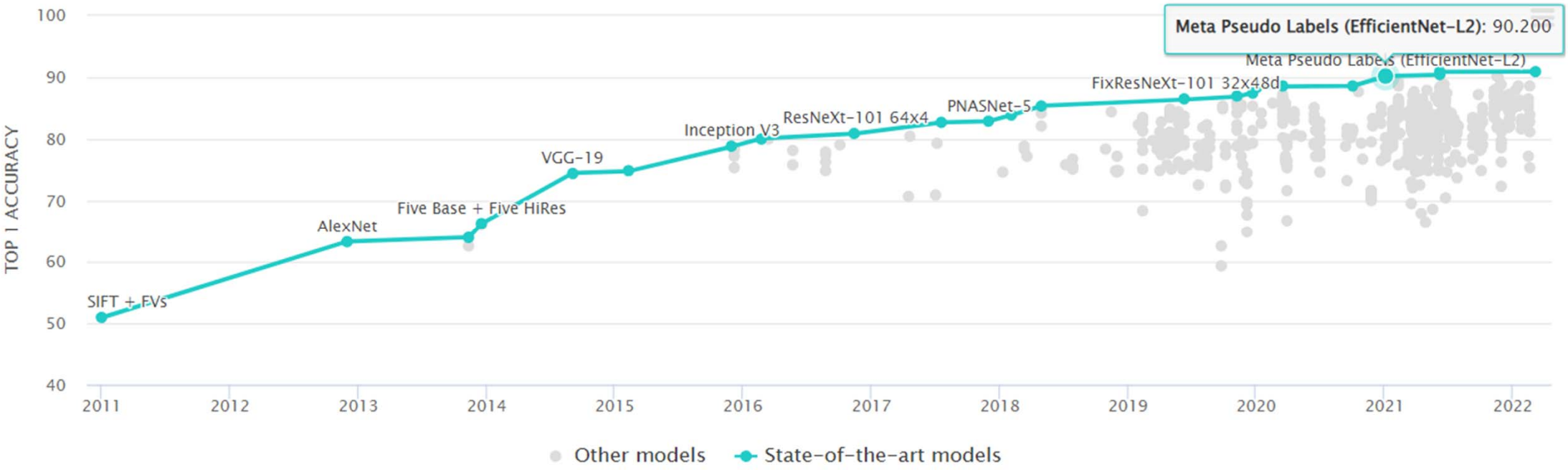
- [16] : SAM(Sharpness-Aware Minimization for Efficiently Improving Generalization) Optimizer, Google Research, 2020년 말 SOTA

9 Meta Pseudo Labels

Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, Quoc V. Le, “Meta Pseudo Labels,” arXiv:2003.10580, 2020, **CVPR2021** (**Google AI, Brain Team**)

https://paperswithcode.com/paper/meta-pseudo-labels

ImageNet에서 처음으로 top-1 accuracy 90% 넘김



Opinion
- 2020년 처음 ImageNet Top-1 Accuracy 90.0% 이상 달성
- 2019년 EfficientNet와 2020년대 ViT 이후로, SOTA를 위에 두 방법이 양분함.
- EfficientNet과 ViT 방식 이외에 또 다른 ImageNet Top-1 Accuracy에서 SOTA를 달성한 방법이 Semi-supervised Learning을 적용한 Noisy Student가 대표적임

Opinion
- "Meta Pseudo Labels"는 EfficientNet을 기본으로 사용하고 있지만, Noisy Student 방법처럼 Semi-supervised Learning 방식을 차용하고 있음

9 Meta Pseudo Labels

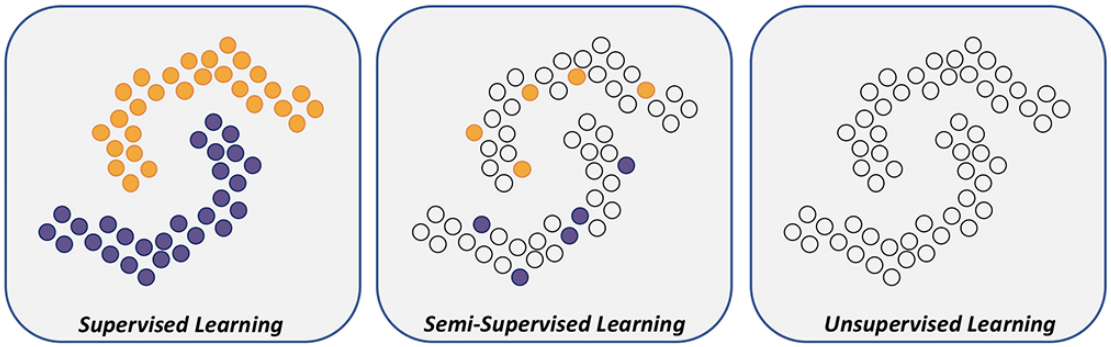
Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, Quoc V. Le, “Meta Pseudo Labels,” arXiv:2003.10580, 2020, **CVPR2021** (**Google AI, Brain Team**)

Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	Extra Training Data	Paper	Code	Result	Year	Tags
1	**CoAtNet-7**	90.88%		2440M	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer, JFT-3B
2	**ViT-G/14**	90.45%		1843M	✓	Scaling Vision Transformers			2021	Transformer, JFT-3B
3	**CoAtNet-6**	90.45%		1470M	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer, JFT-3B
4	**V-MoE-15B** (Every-2)	90.35%		14700M	✓	Scaling Vision with Sparse Mixture of Experts			2021	Transformer
5	**Meta Pseudo Labels** (EfficientNet-L2)	90.2%	98.8%	480M	✓	Meta Pseudo Labels			2021	EfficientNet, JFT-300M
6	**SwinV2-G**	90.17%			✓	Swin Transformer V2: Scaling Up Capacity and Resolution			2021	Transformer
7	**Florence-CoSwin-H**	90.05%	99.02%		✓	Florence: A New Foundation Model for Computer Vision			2021	Transformer

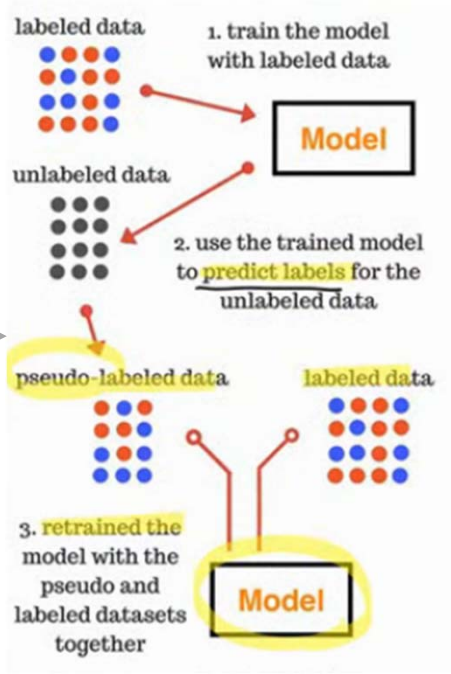
9 Meta Pseudo Labels

SSL (Semi-Supervised Learning) Methods

- **Consistency Regularization**
  - Ladder networks
- **Pseudo-label Methods (Entropy Minimization)**
  - Self Training
- Generative Models
  - VAE
  - GAN
- Graph-Based Methods
  - Label Propagation
  - Node Embedding → Few-shot relative
- **Holistic Approach (Consistency Regularization + Pseudo-label Methods)**
  - MixMatch, ReMixMatch, FixMatch



Self-training Example



Pseudo Labeled Strategy : 위에서 얻은 전체 데이터 쌍를 가지고 테스트시 사용할 새로운 모형을 만든다. 2번째에서 얻은 예측치 중에 Confidence 가 높은 데이터만을 가지고, final 모형을 만들고 이를 반복한다.
Unlabeled Data 학습시 Confidence 가 높은 데이터가 틀리면 Loss를 높게 증가하고, **Confidence 가 낮은 데이터가 틀리면, Loss를 적게 증가**시킨다.

9 Meta - Pseudo Labels

SSL (Semi-Supervised Learning) Methods

➢ Consistency Regularization

- Classifier should **output the same class distribution for an unlabeled example** even after it has been **augmented**

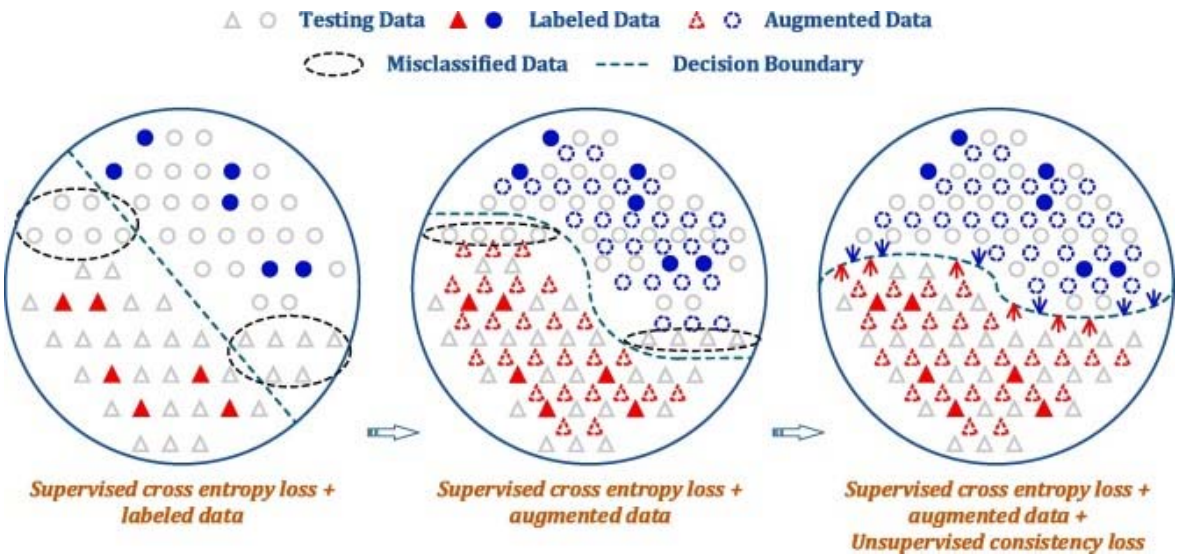
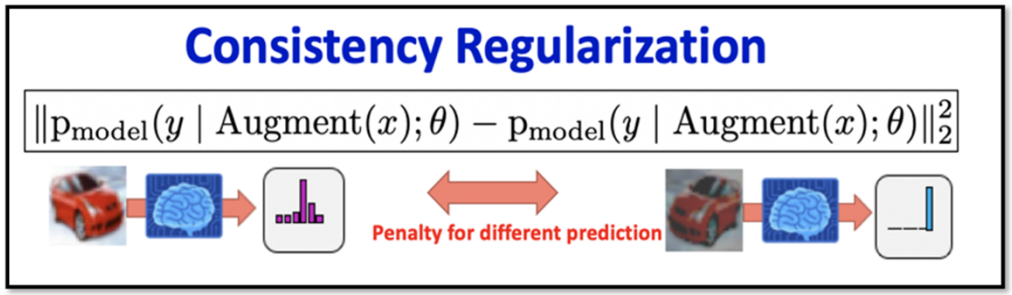


Fig source: A consistency regularization based semi-supervised learning approach for intelligent fault diagnosis of rolling bearing, Kun et al., 2020



Consistency Regularization : “데이터에 가해진 작은 변형은 라벨을 변형시키지 않는다 ” 이다.

왼쪽의 차에 오른쪽과 같이 변형을 가해도 ” 차 ” 라는 라벨은 변하지 않는다는 것. 그렇다면 모델 또한 동일하게 예측해야 되기 때문에 왼쪽 사진의 예측 확률 분포와 오른쪽 사진의 예측 확률 분포를 동일하게 만들도록 학습하는 것

$$\sum_{b=1}^{\mu B} \|p_m(y|\alpha(u_b)) - p_m(y|\alpha(u_b))\|_2^2$$

Formula of consistency regularization

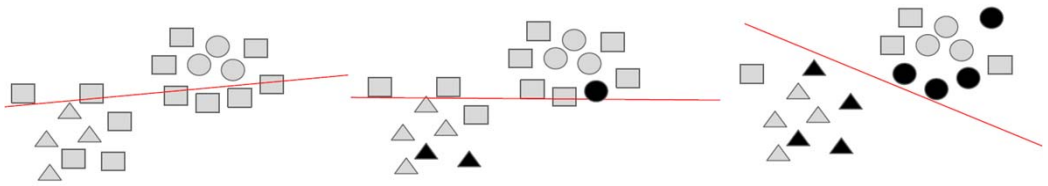
[참조] https://ainote.tistory.com/6

9 Meta - Pseudo Labels

SSL (Semi-Supervised Learning) Methods

➢ Entropy Minimization

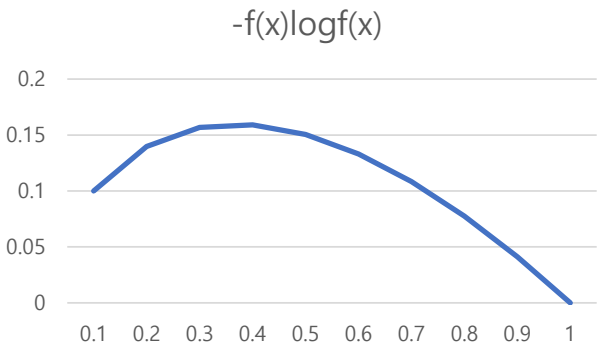
- Classifier's decision boundary should not pass through high-density regions of the marginal data distribution → **pseudo label (sharpening)**



- **Entropy minimization**는 **'decision boundary**는 데이터의 저밀도 지역에서 형성될 것'이라는 가정에 기초해서 **unlabeled data output**의 **entropy**를 **minimization**함. (애매하게 하지 말고 확실하게 함)
- Classifier A가 예측한 y에 대한 확률이 **[0.1, 0.8, 0.1]**이고,
- Classifier B가 예측한 y에 대한 확률이 **[0.1, 0.6, 0.3]** 이라고 하자.
- 이때, Classifier A가 더 confident하고 낮은 entropy를 갖는다. (class 1을 예측하는데 좀 더 확신을 갖고 있다.) 이와 같이 모델을 좀 더 confident하게 학습하는 것이 entropy minimization의 목표이다 (=decision boundary 근처에 있는 모호한 것을 확실하게 구별할 수 있도록)

- **labeled data** 학습할 때는 **objective function**에 **entropy minimization term**을 추가한 형태의 loss로 학습을 진행하고, **unlabeled data**에 대해서는 **entropy minimization term**에 대해서만 학습을 진행

[0.1, 0.8, 0.1] : entropy=0.277
[0.1, 0.6, 0.3] : entropy=0.390



A simple loss term which can be applied to unlabeled data is to encourage the network to make "confident" (low-entropy) predictions for all examples, regardless of the actual class predicted. Assuming a categorical output space with K possible classes (e.g. a K-dimensional softmax output), this gives rise to the "entropy minimization" term (Grandvalet & Bengio, 2005):

$$-\sum_{k=1}^{K} f_\theta(x)_k \log f_\theta(x)_k \qquad (4)$$

9 Meta - Pseudo Labels

SSL (Semi-Supervised Learning) Methods

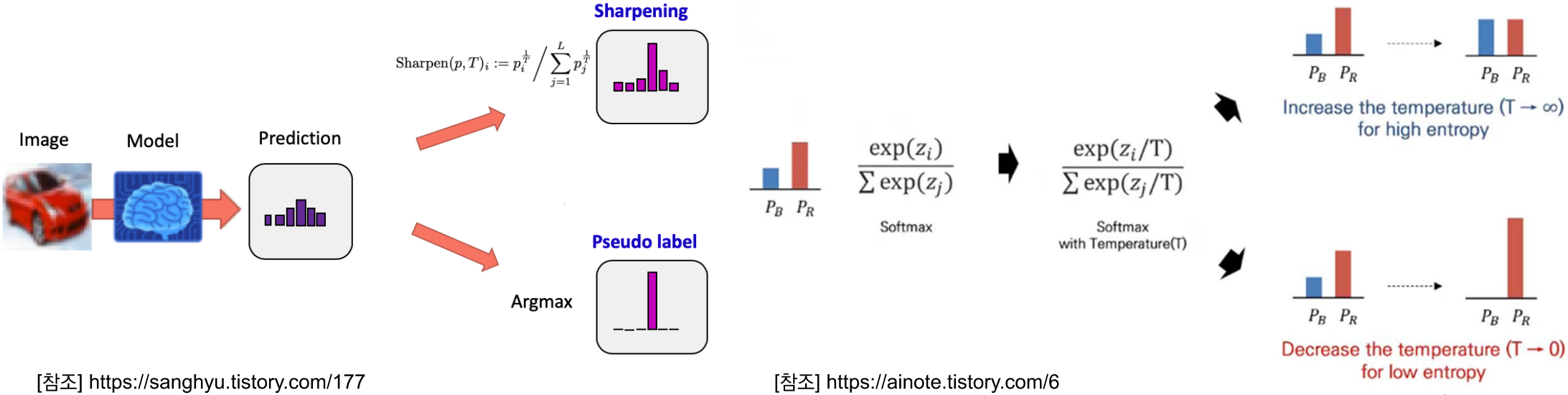
➢ Entropy Minimization

- 이 방법은 단일기법으로 성능이 좋지 못하나, 최신 방법들에 약간의 변형 사용시 효과 좋음.
- MixMatch, UDA, ReMixMatch : temperature sharpening을 통해 entropy minimization을 간접적으로 사용
- FixMatch : pseudo-label을 이용하므로, 간접적 entropy minimization임 (pseudo-label은 pseudo-label을 달 때, argmax()를 통해 one-hot vector로 mapping 과정이 들어가므로, entropy minimization라고 할 수 있음)

$$\text{Sharpen}(p, T)_i := p_i^{\frac{1}{T}} \Big/ \sum_{j=1}^{L} p_j^{\frac{1}{T}}$$

p	$p^{1/T}$	
T	0.2	
0.1	0.00001	0.000917
0.2	0.00032	0.029358
0.4	0.01024	0.93945
0.2	0.00032	0.029358
0.1	0.00001	0.000917
	0.0109	1

p	$p^{1/T}$	
T	2	
0.1	0.316228	0.146447
0.2	0.447214	0.207107
0.4	0.632456	0.292893
0.2	0.447214	0.207107
0.1	0.316228	0.146447
	2.159338	1



[참조] https://sanghyu.tistory.com/177

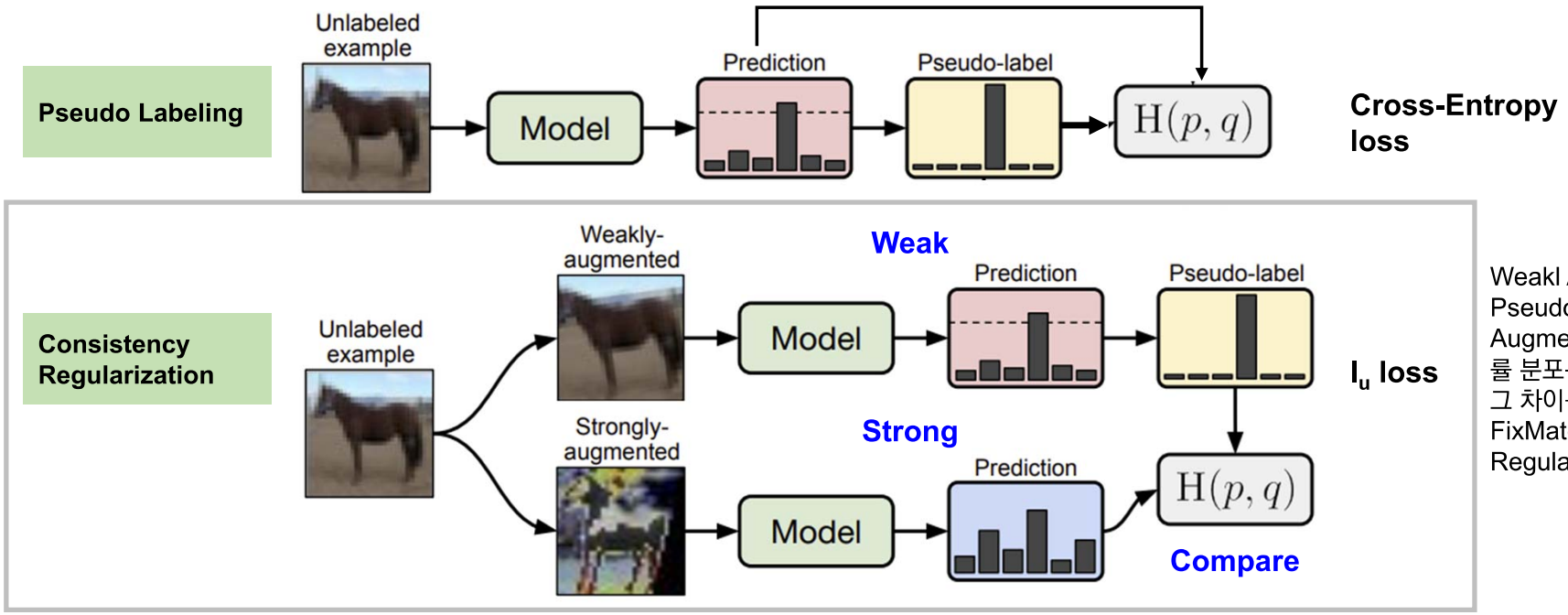
[참조] https://ainote.tistory.com/6

9 Meta - Pseudo Labels

SSL (Semi-Supervised Learning) Methods

➢ **FixMatch**

K. Sohn et al., "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence," Google Research, arxiv



WeakI Augmented된 이미지를 통해 Pseudo-label을 생성. Strong Augmented 이미지를 모델이 예측한 확률 분포는 정답 값과 차이가 나게 된다. 그 차이를 줄이며 학습하는 것이 FixMatch의 학습 방식 (Consistency Regularization의 변형)

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.**

9 Meta - Pseudo Labels

Self-training with Noisy Student

Require: Labeled images $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ and unlabeled images $\{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_m\}$.

1: Learn teacher model θ_*^t which minimizes the cross entropy loss on labeled images

$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f^{noised}(x_i, \theta^t))$$

2: Use a normal (i.e., not noised) teacher model to generate soft or hard pseudo labels for clean (i.e., not distorted) unlabeled images

$$\tilde{y}_i = f(\tilde{x}_i, \theta_*^t), \forall i = 1, \cdots, m$$

3: Learn an **equal-or-larger** student model θ_*^s which minimizes the cross entropy loss on labeled images and unlabeled images with **noise** added to the student model

$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f^{noised}(x_i, \theta^s)) + \frac{1}{m}\sum_{i=1}^{m} \ell(\tilde{y}_i, f^{noised}(\tilde{x}_i, \theta^s))$$

4: Iterative training: Use the student as a teacher and go back to step 2.

Algorithm 1: Noisy Student Training.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, Quoc V. Le, “Self-training with Noisy Student improves ImageNet classification.” arXiv:1911.04252, CVPR2020, Google Research Team / Carnegie Mellon Univ.

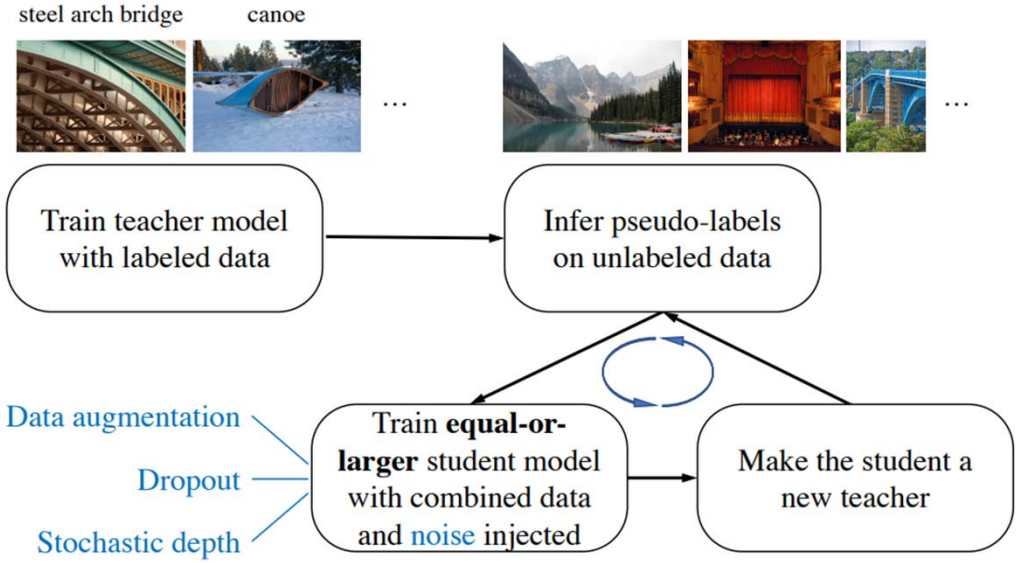


Figure 1: Illustration of the Noisy Student Training. (All shown images are from ImageNet.)

Noisy Student에는 labeled image로 teacher를 학습시키고, teacher로 unlabeled image에 대한 pseudo label를 생성함. teacher로 생성한 pseudo labeled image와 labeled image로 student를 학습함. 이 student를 teacher로 사용하여 다시 pseudo label을 생성함. 그리고 새로운 student은 새로운 pseudo labeled image와 labeled image로 학습함. 이 과정이 반복되면서 Student는 teacher보다 나은 일반화된 성능을 얻게 된다. 그리고 Noisy student의 핵심은 dropout, random augmentation, stochastic depth 를 활용하여 student에 noise를 추가한 것이다.

Noist Studet의 단점은 teacher가 생성한 pseudo label이 정확하지 않으면, student가 정확하지 않은 데이터로 학습됨. 따라서 teacher보다 더 좋은 성능을 얻을 수 없게 됨. 이 단점을 개선하기 위해서 Meta Pseudo Labels는 teacher와 student를 동시에 학습함

9 Meta - Pseudo Labels

Rethinking ImageNet Pre-training

Kaiming He, Ross Girshick, Piotr Dollár, “Rethinking ImageNet Pre-training,” ICCV2019
Facebook AI Research (FAIR)

- He et al. show a surprising result that such **ImageNet pre-training does not improve accuracy on the COCO dataset**

- How about more data?
- How about stronger data augmentation?
- How about self-training?

기존의 ImageNet기반의 pre-training 방법

pre-training없이 scratch (random으로 셋팅)로 학습

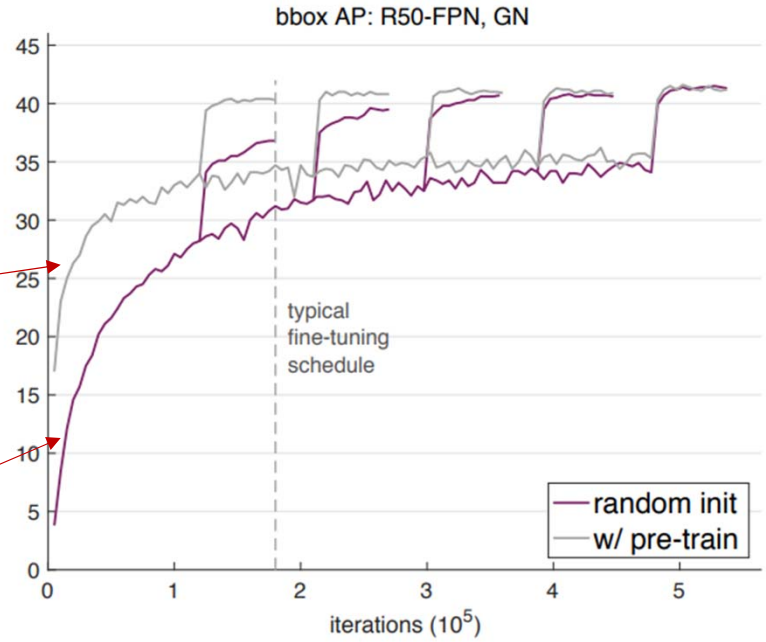


Figure 1. We train Mask R-CNN with a ResNet-50 FPN(Feature Pyramid Network) and GroupNorm backbone on the COCO train2017 set and evaluate bounding box AP on the val2017 set, initializing the model by **random weights** or **ImageNet pre-training**. We explore different training schedules by varying the iterations at which the learning rate is reduced (where the accuracy leaps). The model trained from random initialization needs more iterations to converge, but converges to a solution that is no worse than the finetuning counterpart. Table 1 shows the resulting AP numbers.

- Pre-Training은 빠른 학습을 돕긴 하지만 2가지 가정이 뒷받침될 때 Scratch(w/o Pre-Training)로 학습을 해도 충분히 좋은 성능을 낼 수 있다고 여러 실험을 통해 증명하였다. : 1) 적절한 Normalization, 2) 충분히 오래 학습하기

9 Meta - Pseudo Labels

Rethinking Pre-training and Self-Training

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D. Cubuk, Quoc V. Le , “Rethinking Pre-training and Self-training,” arXiv:2006.06882, Google Research Team

Setup	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Rand Init	39.2	41.5	43.9	44.3
ImageNet Init	(+0.3) 39.5	(-0.7) 40.7	(-0.8) 43.2	(-1.0) 43.3
Rand Init w/ ImageNet Self-training	(+1.7) 40.9	(+1.5) 43.0	(+1.5) 45.4	(+1.3) 45.6

Table 2: In regimes where pre-training hurts, self-training with the same data source helps. All models are trained on the full COCO dataset.

- Stronger data augmentation and more labeled data further diminish the value of pre-training.
- Unlike pre-training, **self-training** is always helpful when using stronger data augmentation, in both low data and high-data regimes.
- In the case that pre-training is helpful, self-training improves upon pre-training.

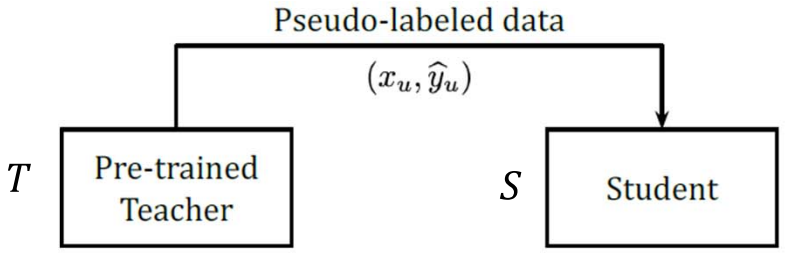
- Pre-training의 필요성을 부정하며, 오히려 Self-training을 하면 더 좋은 성능을 낼 수 있음을 강조한다. 또한, 강력한 Data Augmentation을 이용할 경우 오히려 Pre-training이 성능을 악화 시킨다고 실험을 통해 밝혔다.

9 Meta - Pseudo Labels

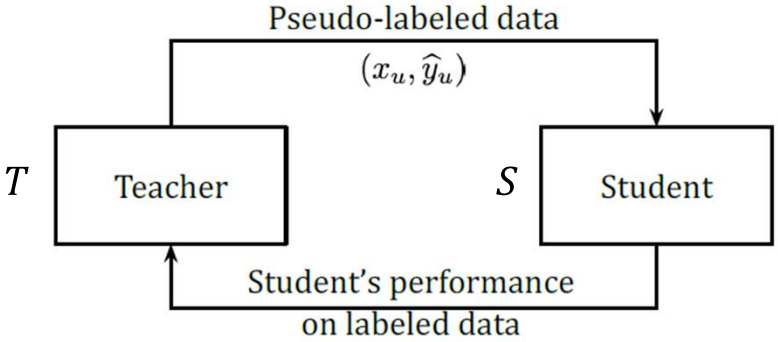
Meta-Pseudo Labels Methods

Notations

Pseudo Labels



Meta Pseudo Labels



Pseudo Labels, where a fixed pre-trained teacher (θ_T) generates pseudo labels for the student to learn from.

Meta Pseudo Labels, where the teacher is trained along with the student. The student is trained based on the pseudo labels generated by the teacher (top arrow). The teacher is trained based on the performance of the student on labeled data (bottom arrow).

Drawback : The problem of confirmation bias in pseudo-labeling
If the pseudo-labels are inaccurate, the student will learn from inaccurate.

→

Meta Pseudo Labels utilizes the feedback from the student to inform the teacher to generate better pseudo labels.

T, S : Teacher Network, Student Network

θ_T, θ_T : Parameters for T and S

(x_l, y_l) : A batch of images and labels

x_u : A batch of unlabeled images

$T(x_u; \theta_T)$: *Soft* predictions of Teacher network on the batch of unlabeled images x_u

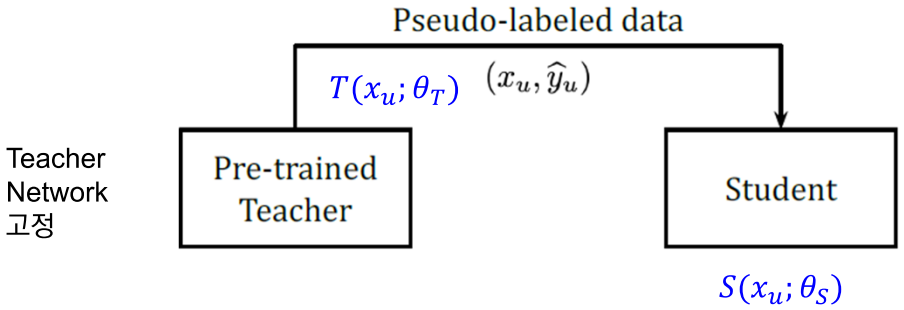
$S(x_u; \theta_S)$, $S(x_l; \theta_S)$

$CE(q, p)$: Cross-entropy loss between two distributions

9 Meta - Pseudo Labels

Meta-Pseudo Labels Methods

Pseudo Labels as an optimization problem



- **Pseudo Labels (PL) trains the student model to minimize the cross-entropy loss on unlabeled data.**
- *Pseudo target* $T(x_u;\theta_T)$ is produced well pre-trained teacher model with *fixed* parameter θ_T

$$\theta_S^{\mathrm{PL}} = \underset{\theta_S}{\operatorname{argmin}} \underbrace{\mathbb{E}_{x_u}\Big[\mathrm{CE}\big(T(x_u;\theta_T), S(x_u;\theta_S)\big)\Big]}_{:=\mathcal{L}_u(\theta_T,\theta_S)} \quad (1)$$

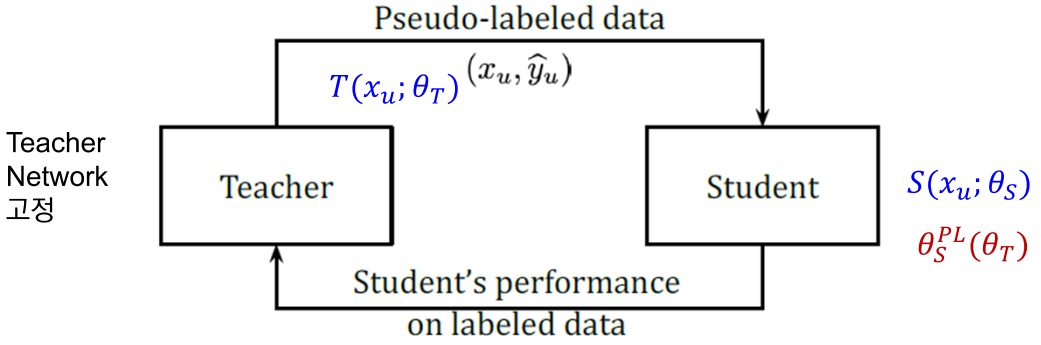
- Hope that the obtained θ_S^{PL} would ultimately achieve a low loss on labeled data,

$$\mathbb{E}_{x_l,y_l}\Big[\mathrm{CE}\big(y_l, S(x_l;\theta_S^{\mathrm{PL}})\big)\Big] := \mathcal{L}_l\big(\theta_S^{\mathrm{PL}}\big)$$

9 Meta - Pseudo Labels

Meta-Pseudo Labels Methods

Meta-Pseudo Pseudo Labels



- Notice that the optimal student parameter θ_S^{PL} always depends on the teacher parameter θ_T. Express the dependency as $\theta_S^{PL}(\theta_T)$

$$\theta_S^{\mathrm{PL}} = \underset{\theta_S}{\operatorname{argmin}} \underbrace{\mathbb{E}_{x_u}\left[\mathrm{CE}\left(T(x_u;\theta_T), S(x_u;\theta_S)\right)\right]}_{:=\mathcal{L}_u(\theta_T,\theta_S)} \quad (1)$$

$$\theta_S^{\mathrm{PL}}(\theta_T) = \underset{\theta_S}{\operatorname{argmin}}\, \mathcal{L}_u(\theta_T, \theta_S)$$

- Ultimate **student loss** $L_l(\theta_S^{PL}(\theta_T))$ on **labeled data** is also a **function of teacher model parameters** θ_T.

Optimize L_l with respect to θ_T

$$\min_{\theta_T} \mathcal{L}_l\left(\theta_S^{\mathrm{PL}}(\theta_T)\right), \quad \text{Dependency of student on teacher}$$
$$\text{where} \quad \theta_S^{\mathrm{PL}}(\theta_T) = \underset{\theta_S}{\operatorname{argmin}}\, \mathcal{L}_u(\theta_T, \theta_S). \quad (2)$$

- By optimizing the teacher's parameter according to **the performance of the student on labeled data**, the pseudo labels can be adjusted accordingly to further improve student's performance.

Meta Pseudo Labels 최적화 어려움
- Teacher와 Student의 서로 간의 종속성 때문에, $\nabla_{\theta_T}\theta_S^{PL}(\theta_T)$ 를 계산하기 어려움
- 예를 들어, $argmin_{\theta_S}$ 가 모두 구해져 있어야 함

- As trying to **optimize the teacher on a meta level**, name Meta Pseudo Labels

9 Meta - Pseudo Labels

Practical Approximation

- From meta learning, approximate multi-step $argmin_{\theta_S}$ with the one-step gradient update of θ_S

$$\theta_S^{\mathrm{PL}}(\theta_T) \approx \theta_S - \eta_S \cdot \nabla_{\theta_S} \mathcal{L}_u\left(\theta_T, \theta_S\right),$$

$$\begin{aligned} &\min_{\theta_T} \quad \mathcal{L}_l\left(\theta_S^{\mathrm{PL}}(\theta_T)\right), \\ &\text{where} \quad \theta_S^{\mathrm{PL}}(\theta_T) = \operatorname*{argmin}_{\theta_S} \mathcal{L}_u\left(\theta_T, \theta_S\right). \end{aligned} \tag{2}$$

- Plugging this approximation into the optimization problem in Equation (2) leads to the **practical teacher objective in Meta Pseudo Labels**

$$\min_{\theta_T} \quad \mathcal{L}_l\Big(\theta_S - \eta_S \cdot \nabla_{\theta_S} \mathcal{L}_u\left(\theta_T, \theta_S\right)\Big). \tag{3}$$

- Soft label : Fully differentiable, Large computation
- Hard label : Small computation, REINFORCE approximation

If ***soft pseudo labels*** are used, i.e. $T(x_u; \theta_T)$ is the full distribution predicted by teacher, the objective above is fully differentiable with respect to θ_T and we can perform standard back-propagation to get the gradient.

However, in this work, we sample the **hard pseudo labels** from the teacher distribution to train the student. We use hard pseudo labels because they result in smaller computational graphs which are necessary for our large-scale experiments.

9 Meta - Pseudo Labels

Practical Approximation

- An **alternating optimization procedure** between the student update and the teacher update:

- Student: draw a batch of unlabeled data x_u, then sample $T(x_u; \theta_T)$ from teacher's prediction, and optimize objective 1 with SGD: $\theta'_S = \theta_S - \eta_S \nabla_{\theta_S} \mathcal{L}_u(\theta_T, \theta_S)$,

  Self-training

- Teacher: draw a batch of labeled data (x_l, y_l), and "reuse" the student's update to optimize objective 3 with SGD: $\theta'_T = \theta_T - \eta_T \nabla_{\theta_T} \mathcal{L}_l\big(\underbrace{\theta_S - \nabla_{\theta_S} \mathcal{L}_u(\theta_T, \theta_S)}_{= \theta'_S \text{ reused from student's update}}\big)$.

- The student's training relies on the objective Equation (1), except that the teacher parameter is not fixed anymore.
- θ_T is constantly changing due to the teacher's optimization.

- Student's parameter update can be reused in the one-step approximation of the teacher's objective

Auxiliary trains

- Teacher model : Supervised learning objective + Semi-supervised objective (on unlabeled data using UDA(Unsupervised Data Augmentation))
- Student model : After training meta-pseudo labels, finetune it on labeled data to improve its accuracy

9 Meta - Pseudo Labels

$$\theta'_T = \theta_T - \eta_T \nabla_{\theta_T} \mathcal{L}_l\big(\underbrace{\theta_S - \nabla_{\theta_S}\mathcal{L}_u(\theta_T, \theta_S)}_{=\theta'_S \text{ reused from student's update}}\big).$$

➢ Appendix A : Derivation of the Teacher's Update Rule

- By the chain rule, we have:

$$\begin{aligned}
\underbrace{\frac{\partial R}{\partial \theta_T}}_{1\times|T|} &= \frac{\partial}{\partial \theta_T}\mathrm{CE}\left(y_l, S\left(x_l; \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \nabla_{\theta_S}\mathrm{CE}(\widehat{y}_u, S(x_u;\theta_S))\right]\right)\right) \\
&= \frac{\partial}{\partial \theta_T}\mathrm{CE}\left(y_l, S(x_l; \bar{\theta}'_S)\right) \\
&= \underbrace{\frac{\partial \mathrm{CE}\left(y_l, S(x_l;\bar{\theta}'_S)\right)}{\partial \theta_S}\bigg|_{\theta_S=\bar{\theta}'_S}}_{1\times|S|} \cdot \underbrace{\frac{\partial \bar{\theta}'_S}{\partial \theta_T}}_{|S|\times|T|}
\end{aligned} \tag{6}$$

- The first factor in Equation 6 can be simply computed via back-propagation. We now focus on the second term

$$\begin{aligned}
\underbrace{\frac{\partial \bar{\theta}'_S}{\partial \theta_T}}_{|S|\times|T|} &= \frac{\partial}{\partial \theta_T}\mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \nabla_{\theta_S}\mathrm{CE}(\widehat{y}_u, S(x_u;\theta_S))\right] \\
&= \frac{\partial}{\partial \theta_T}\mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \cdot \left(\frac{\partial \mathrm{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)}{\partial \theta_S}\bigg|_{\theta_S=\theta_S}\right)^\top\right]
\end{aligned} \tag{7}$$

- To simplify notations, $\underbrace{g_S(\widehat{y}_u)}_{|S|\times|1|} = \left(\frac{\partial \mathrm{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)}{\partial \theta_S}\bigg|_{\theta_S=\theta_S}\right)^\top$ (8)

- Then, Equation (7) becomes

$$\underbrace{\frac{\partial \bar{\theta}'_S}{\partial \theta_T}}_{|S|\times|T|} = -\eta_S \cdot \frac{\partial}{\partial \theta_T}\mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\Big[\underbrace{g_S(\widehat{y}_u)}_{|S|\times 1}\Big] \tag{9}$$

- Applying *REINFORCE* equation (Monte-Carlo Policy Gradient) and Monte-Carlo Approximation,

$$\begin{aligned}
\nabla_{\theta_T}\mathcal{L}_l &= \eta_S \cdot \underbrace{\frac{\partial \mathrm{CE}\left(y_l, S(x_l;\theta'_S)\right)}{\partial \theta_S}}_{1\times|S|} \cdot \underbrace{\left(\frac{\partial \mathrm{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)}{\partial \theta_S}\bigg|_{\theta_S=\theta_S}\right)^\top}_{|S|\times 1} \cdot \underbrace{\frac{\partial \mathrm{CE}\left(\widehat{y}_u, T(x_u;\theta_T)\right)}{\partial \theta_T}}_{1\times|T|} \\
&= \underbrace{\eta_S \cdot \left(\left(\nabla_{\theta'_S}\mathrm{CE}\left(y_l, S(x_l;\theta'_S)\right)\right)^\top \cdot \nabla_{\theta_S}\mathrm{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)\right)}_{\text{A scalar} := h} \cdot \nabla_{\theta_T}\mathrm{CE}\left(\widehat{y}_u, T(x_u;\theta_T)\right)
\end{aligned} \tag{12}$$

Gradient of Student with θ'_S on labeled data (x_l, y_l)

Gradient of Student with θ_S on pseudo label $(x_u, \hat{y}_u)$

Gradient of Teacher with θ_T on pseudo label $(x_u, \hat{y}_u)$

Student feedback coefficient

9 Meta - Pseudo Labels

Algorithm

- UDA(Unsupervised Data Augmentation) objective is applied on the *teacher*
- The *student* only learns from the pseudo labeled data given by the teacher

Algorithm 1 The Meta Pseudo Labels method, applied to a teacher trained with UDA [76].

Input: Labeled data x_l, y_l and unlabeled data x_u.

Initialize $\theta_T^{(0)}$ and $\theta_S^{(0)}$

for $t = 0$ **to** $N - 1$ **do**

Sample an unlabeled example x_u and a labeled example x_l, y_l

Sample a pseudo label $\widehat{y}_u \sim P(\cdot|x_u; \theta_T)$

Update the student using the pseudo label $\widehat{y}_u$:

$$\theta_S^{(t+1)} = \theta_S^{(t)} - \eta_S \nabla_{\theta_S} \mathrm{CE}(\widehat{y}_u, S(x_u; \theta_S))|_{\theta_S = \theta_S^{(t)}}$$

Compute Student's feedback coefficient as in Equation 12:

$$h = \eta_S \cdot \left(\left(\nabla_{\theta'_S} \mathrm{CE}\left(y_l, S(x_l; \theta_S^{(t+1)})\right) \right)^\top \cdot \nabla_{\theta_S} \mathrm{CE}\left(\widehat{y}_u, S(x_u; \theta_S^{(t)})\right) \right)$$

Compute the teacher's gradient from the student's feedback:

$$g_T^{(t)} = h \cdot \nabla_{\theta_T} \mathrm{CE}(\widehat{y}_u, T(x_u; \theta_T))|_{\theta_T = \theta_T^{(t)}}$$

Compute the teacher's gradient on labeled data:

$$g_{T,\text{supervised}}^{(t)} = \nabla_{\theta_T} \mathrm{CE}(y_l, T(x_l; \theta_T))|_{\theta_T = \theta_T^{(t)}}$$

Compute the teacher's gradient on the UDA loss with unlabeled data:

$$g_{T,\text{UDA}}^{(t)} = \nabla_{\theta_T} \mathrm{CE}\Big(\text{StopGradient}(T(x_l); \theta_T), T(\text{RandAugment}(x_l); \theta_T)\Big)\Big|_{\theta_T = \theta_T^{(t)}}$$

(x_u under $T(x_l)$, x_u under RandAugment)

Update the teacher:

$$\theta_T^{(t+1)} = \theta_T^{(t)} - \eta_T \cdot \left(g_T^{(t)} + g_{T,\text{supervised}}^{(t)} + g_{T,\text{UDA}}^{(t)} \right)$$

end

return $\theta_S^{(N)}$

➢ Only the student model is returned for prediction and evaluations

9 Meta - Pseudo Labels

Code

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

train_loop

```
def train_loop(args, labeled_loader, unlabeled_loader, test_loader, finetune_dataset,
               teacher_model, student_model, avg_student_model, criterion,
               t_optimizer, s_optimizer, t_scheduler, s_scheduler, t_scaler, s_scaler):
    logger.info("***** Running Training *****")
    logger.info(f"   Task = {args.dataset}@{args.num_labeled}")
    logger.info(f"   Total steps = {args.total_steps}")

    if args.world_size > 1:
        labeled_epoch = 0
        unlabeled_epoch = 0
        labeled_loader.sampler.set_epoch(labeled_epoch)
        unlabeled_loader.sampler.set_epoch(unlabeled_epoch)

    labeled_iter = iter(labeled_loader)
    unlabeled_iter = iter(unlabeled_loader)

    # for author's code formula
    # moving_dot_product = torch.empty(1).to(args.device)
    # limit = 3.0**(0.5)  # 3 = 6 / (f_in + f_out)
    # nn.init.uniform_(moving_dot_product, -limit, limit)
```

```
for step in range(args.start_step, args.total_steps):
    if step % args.eval_step == 0:
        pbar = tqdm(range(args.eval_step), disable=args.local_rank not in [-1, 0])
        batch_time = AverageMeter()
        data_time = AverageMeter()
        s_losses = AverageMeter()
        t_losses = AverageMeter()
        t_losses_l = AverageMeter()
        t_losses_u = AverageMeter()
        t_losses_mpl = AverageMeter()
        mean_mask = AverageMeter()

    teacher_model.train()
    student_model.train()
    end = time.time()

    try:
        # error occurs ↓
        # images_l, targets = labeled_iter.next()
        images_l, targets = next(labeled_iter)
    except:
        if args.world_size > 1:
            labeled_epoch += 1
            labeled_loader.sampler.set_epoch(labeled_epoch)
        labeled_iter = iter(labeled_loader)
        # error occurs ↓
        # images_l, targets = labeled_iter.next()
        images_l, targets = next(labeled_iter)
```

(x_l, y_l)

9 Meta - Pseudo Labels

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

```
for step
    try:
        # error occurs ↓
        # (images_uw, images_us), _ = unlabeled_iter.next()
        (images_uw, images_us), _ = next(unlabeled_iter)
    except:                                   (x_u, aug(x_u))        (x_u, y_u)
        if args.world_size > 1:
            unlabeled_epoch += 1
            unlabeled_loader.sampler.set_epoch(unlabeled_epoch)
        unlabeled_iter = iter(unlabeled_loader)
        # error occurs ↓
        # (images_uw, images_us), _ = unlabeled_iter.next()
        (images_uw, images_us), _ = next(unlabeled_iter)

    data_time.update(time.time() - end)

    images_l = images_l.to(args.device)
    images_uw = images_uw.to(args.device)
    images_us = images_us.to(args.device)
    targets = targets.to(args.device)
    with amp.autocast(enabled=args.amp):
        batch_size = images_l.shape[0]                    (x_l, x_u, aug(x_u))
        t_images = torch.cat((images_l, images_uw, images_us))
        t_logits = teacher_model(t_images)
T(x_l; θ_T)  t_logits_l = t_logits[:batch_size]
        t_logits_uw, t_logits_us = t_logits[batch_size:].chunk(2)
        del t_logits          T(x_u; θ_T)   T(aug(x_u); θ_T)

        t_loss_l = criterion(t_logits_l, targets)      CE(y_l, T(x_l; θ_T))
```

```
for step
    soft_pseudo_label = torch.softmax(t_logits_uw.detach() / args.temperature, dim=-1)
ŷ_u max_probs, hard_pseudo_label = torch.max(soft_pseudo_label, dim=-1)
    mask = max_probs.ge(args.threshold).float()
    t_loss_u = torch.mean(
        -(soft_pseudo_label * torch.log_softmax(t_logits_us, dim=-1)).sum(dim=-1) * mask
    )
    weight_u = args.lambda_u * min(1., (step + 1) / args.uda_steps)
    t_loss_uda = t_loss_l + weight_u * t_loss_u

    s_images = torch.cat((images_l, images_us))   (x_l, aug(x_u))
    s_logits = student_model(s_images)
    s_logits_l = s_logits[:batch_size]      S(x_l; θ_S)
    s_logits_us = s_logits[batch_size:]     S(aug(x_u); θ_S)
    del s_logits

    s_loss_l_old = F.cross_entropy(s_logits_l.detach(), targets)   CE(y_l, S(x_l; θ_S))
    s_loss = criterion(s_logits_us, hard_pseudo_label)             CE(ŷ_u, S(aug(x_u); θ_S))

s_scaler.scale(s_loss).backward()
if args.grad_clip > 0:
    s_scaler.unscale_(s_optimizer)
    nn.utils.clip_grad_norm_(student_model.parameters(), args.grad_clip)
s_scaler.step(s_optimizer)
s_scaler.update()
s_scheduler.step()
if args.ema > 0:
    avg_student_model.update_parameters(student_model)
```

Student Model Backpropagation

9 Meta - Pseudo Labels

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

- Teacher model loss function 이해 : Without Student model Feedback

$T(x_u; \theta_T)$

```
soft_pseudo_label = torch.softmax(t_logits_uw.detach() / args.temperature, dim=-1)
max_probs, hard_pseudo_label = torch.max(soft_pseudo_label, dim=-1)
mask = max_probs.ge(args.threshold).float()
t_loss_u = torch.mean(                    T(aug(x_u); θ_T)
    -(soft_pseudo_label * torch.log_softmax(t_logits_us, dim=-1)).sum(dim=-1) * mask
)
weight_u = args.lambda_u * min(1., (step + 1) / args.uda_steps)
t_loss_uda = t_loss_l + weight_u * t_loss_u
```

$$g_{T,\mathrm{UDA}}^{(t)} = \nabla_{\theta_T} \boxed{\mathrm{CE}\Big(\mathrm{StopGradient}(T(x_u);\theta_T), T(\mathrm{RandAugment}(x_u);\theta_T)\Big)}\Big|_{\theta_T=\theta_T^{(t)}}$$

$$\theta_T^{(t+1)} = \theta_T^{(t)} - \eta_T \cdot \Big(g_T^{(t)} + \boxed{g_{T,\mathrm{supervised}}^{(t)} + g_{T,\mathrm{UDA}}^{(t)}}\Big)$$

9 Meta - Pseudo Labels

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

- Unlabeled dataset 파악

```
unlabeled_loader = DataLoader(
    unlabeled_dataset,
    sampler=train_sampler(unlabeled_dataset),
    batch_size=args.batch_size * args.mu,
    num_workers=args.workers,
    drop_last=True)


labeled_iter = iter(labeled_loader)
unlabeled_iter = iter(unlabeled_loader)


try:
    # error occurs ↓
    # (images_uw, images_us), _ = unlabeled_iter.next()
    (images_uw, images_us), _ = next(unlabeled_iter)
```

```
train_unlabeled_dataset = CIFAR10SSL(
    args.data_path, train_unlabeled_idxs,
    train=True,
    transform=TransformMPL(args, mean=cifar10_mean, std=cifar10_std)
)
```

```
class TransformMPL(object):
    def __init__(self, args, mean, std):
        if args.randaug:
            n, m = args.randaug
        else:
            n, m = 2, 10  # default

        self.ori = transforms.Compose([
            transforms.RandomHorizontalFlip(),
            transforms.RandomCrop(size=args.resize,
                                  padding=int(args.resize * 0.125),
                                  fill=128,
                                  padding_mode='constant')])
        self.aug = transforms.Compose([
            transforms.RandomHorizontalFlip(),
            transforms.RandomCrop(size=args.resize,
                                  padding=int(args.resize * 0.125),
                                  fill=128,
                                  padding_mode='constant'),
            RandAugmentCIFAR(n=n, m=m)])
        self.normalize = transforms.Compose([
            transforms.ToTensor(),
            transforms.Normalize(mean=mean, std=std)])

    def __call__(self, x):
        ori = self.ori(x)
        aug = self.aug(x)
        return self.normalize(ori), self.normalize(aug)
```

9 Meta - Pseudo Labels

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

for step

➢ Teacher model loss function : Student model 성능 Feedback

```
with amp.autocast(enabled=args.amp):
    with torch.no_grad():
        s_logits_l = student_model(images_l)
    s_loss_l_new = F.cross_entropy(s_logits_l.detach(), targets)

    # theoretically correct formula (https://github.com/kekmodel/MPL-pytorch/issues/6)
    # dot_product = s_loss_l_old - s_loss_l_new

    # author's code formula
    dot_product = s_loss_l_new - s_loss_l_old
    # moving_dot_product = moving_dot_product * 0.99 + dot_product * 0.01
    # dot_product = dot_product - moving_dot_product
    _, hard_pseudo_label = torch.max(t_logits_us.detach(), dim=-1)
    t_loss_mpl = dot_product * F.cross_entropy(t_logits_us, hard_pseudo_label)
    # test
    # t_loss_mpl = torch.tensor(0.).to(args.device)
    t_loss = t_loss_uda + t_loss_mpl

t_scaler.scale(t_loss).backward()
if args.grad_clip > 0:
    t_scaler.unscale_(t_optimizer)
    nn.utils.clip_grad_norm_(teacher_model.parameters(), args.grad_clip)
t_scaler.step(t_optimizer)
t_scaler.update()
t_scheduler.step()
```

$\mathrm{CE}(y_l, S(x_l; \theta_S^{(t+1)}))$

$T(aug(x_u); \theta_T)$

$\mathrm{CE}(\hat{y}_u, T(aug(x_u); \theta_T))$

Teacher Model Backpropagation

for step

```
teacher_model.zero_grad()
student_model.zero_grad()

if args.world_size > 1:
    s_loss = reduce_tensor(s_loss.detach(), args.world_size)
    t_loss = reduce_tensor(t_loss.detach(), args.world_size)
    t_loss_l = reduce_tensor(t_loss_l.detach(), args.world_size)
    t_loss_u = reduce_tensor(t_loss_u.detach(), args.world_size)
    t_loss_mpl = reduce_tensor(t_loss_mpl.detach(), args.world_size)
    mask = reduce_tensor(mask, args.world_size)

s_losses.update(s_loss.item())
t_losses.update(t_loss.item())
t_losses_l.update(t_loss_l.item())
t_losses_u.update(t_loss_u.item())
t_losses_mpl.update(t_loss_mpl.item())
mean_mask.update(mask.mean().item())

batch_time.update(time.time() - end)
pbar.set_description(
    f"Train Iter: {step+1:3}/{args.total_steps:3}. "
    f"LR: {get_lr(s_optimizer):.4f}. Data: {data_time.avg:.2f}s. "
    f"Batch: {batch_time.avg:.2f}s. S_Loss: {s_losses.avg:.4f}. "
    f"T_Loss: {t_losses.avg:.4f}. Mask: {mean_mask.avg:.4f}. ")
pbar.update()
if args.local_rank in [-1, 0]:
    args.writer.add_scalar("lr", get_lr(s_optimizer), step)
      wandb.log({"lr": get_lr(s_optimizer)})
```

9 Meta - Pseudo Labels

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

- Meta gradient

```
th amp.autocast(enabled=args.amp):
  with torch.no_grad():
      s_logits_l = student_model(images_l)
  s_loss_l_new = F.cross_entropy(s_logits_l.detach(), targets)

  # theoretically correct formula (https://github.com/kekmodel/MPL-pytorch/issues/6)
  # dot_product = s_loss_l_old - s_loss_l_new

  # author's code formula
  dot_product = s_loss_l_new - s_loss_l_old
  # moving_dot_product = moving_dot_product * 0.99 + dot_product * 0.01
  # dot_product = dot_product - moving_dot_product
```

$$\nabla_{\theta_T}\mathcal{L}_l = \eta_S \cdot \underbrace{\frac{\partial \mathrm{CE}\left(y_l, S(x_l;\theta'_S)\right)}{\partial \theta_S}}_{1\times|S|} \cdot \underbrace{\left(\frac{\partial \mathrm{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)}{\partial \theta_S}\bigg|_{\theta_S=\theta_S}\right)^{\top}}_{|S|\times 1} \cdot \underbrace{\frac{\partial \mathrm{CE}\left(\widehat{y}_u, T(x_u;\theta_T)\right)}{\partial \theta_T}}_{1\times|T|}$$

$$= \underbrace{\eta_S \cdot \left(\left(\nabla_{\theta'_S}\mathrm{CE}\left(y_l, S(x_l;\theta'_S)\right)\right)^{\top} \cdot \nabla_{\theta_S}\mathrm{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)\right)}_{\text{A scalar} := h} \cdot \nabla_{\theta_T}\mathrm{CE}\left(\widehat{y}_u, T(x_u;\theta_T)\right)$$

Compute the teacher's feedback coefficient as in Equation 12:

$$h = \eta_S \cdot \left(\left(\nabla_{\theta'_S}\mathrm{CE}\left(y_l, S(x_l;\theta_S^{(t+1)})\right)\right)^{\top} \cdot \nabla_{\theta_S}\mathrm{CE}\left(\widehat{y}_u, S(x_u;\theta_S^{(t)})\right)\right)$$

Compute the teacher's gradient from the student's feedback:

$$g_T^{(t)} = h \cdot \nabla_{\theta_T}\mathrm{CE}(\widehat{y}_u, T(x_u;\theta_T))|_{\theta_T=\theta_T^{(t)}}$$

5 Meta gradient with first-order Taylor expansion

As mentioned in the Meta Pseudo paper,

$$h = \eta_s(\nabla_{\boldsymbol{\theta}_s^{i+1}} L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1}))^{\top} \cdot \nabla_{\boldsymbol{\theta}_s^i} L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^i))) \quad (39)$$

$$g_{META} = h\nabla_{\boldsymbol{\theta}_t^i} L(\hat{y}_u, S(\boldsymbol{x}_u;\boldsymbol{\theta}_t^i)) \quad (40)$$

Using the first order Taylor expansion

$$L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^i)) = L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1})) + (\boldsymbol{\theta}_s^i - \boldsymbol{\theta}_s^{i+1})\nabla_{\boldsymbol{\theta}_s^{i+1}} L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1})) + O(2) \quad (41)$$

$$\approx L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1})) + (\boldsymbol{\theta}_s^i - \boldsymbol{\theta}_s^{i+1})\nabla_{\boldsymbol{\theta}_s^{i+1}} L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1})) \quad (42)$$

$$= L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1})) + \eta_s \nabla_{\boldsymbol{\theta}_s^i} L(\hat{y}_u, S(\boldsymbol{x}_u;\boldsymbol{\theta}_s^i)\nabla_{\boldsymbol{\theta}_s^i} L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1}) \quad (43)$$

So that we can use the difference between the old and new student-label loss to compute h

$$h = \eta_s(\nabla_{\boldsymbol{\theta}_s^{i+1}} L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1}))^{\top} \cdot \nabla_{\boldsymbol{\theta}_s^i} L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^i))) \quad (44)$$

$$\approx L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^i)) - L(y_l, S(\boldsymbol{x}_l;\boldsymbol{\theta}_s^{i+1})) \quad (45)$$

9 Meta - Pseudo Labels

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

```
for step
    args.num_eval = step // args.eval_step
    if (step + 1) % args.eval_step == 0:
        pbar.close()
        if args.local_rank in [-1, 0]:
            args.writer.add_scalar("train/1.s_loss", s_losses.avg, args.num_eval)
            args.writer.add_scalar("train/2.t_loss", t_losses.avg, args.num_eval)
            args.writer.add_scalar("train/3.t_labeled", t_losses_l.avg, args.num_eval)
            args.writer.add_scalar("train/4.t_unlabeled", t_losses_u.avg, args.num_eval)
            args.writer.add_scalar("train/5.t_mpl", t_losses_mpl.avg, args.num_eval)
            args.writer.add_scalar("train/6.mask", mean_mask.avg, args.num_eval)
              wandb.log({"train/1.s_loss": s_losses.avg,
                         "train/2.t_loss": t_losses.avg,
                         "train/3.t_labeled": t_losses_l.avg,
                         "train/4.t_unlabeled": t_losses_u.avg,
                         "train/5.t_mpl": t_losses_mpl.avg,
                         "train/6.mask": mean_mask.avg})

            test_model = avg_student_model if avg_student_model is not None else student_model
            test_loss, top1, top5 = evaluate(args, test_loader, test_model, criterion)

            args.writer.add_scalar("test/loss", test_loss, args.num_eval)
            args.writer.add_scalar("test/acc@1", top1, args.num_eval)
            args.writer.add_scalar("test/acc@5", top5, args.num_eval)
              wandb.log({"test/loss": test_loss,
                         "test/acc@1": top1,
                         "test/acc@5": top5})
```

```
for step

    is_best = top1 > args.best_top1
    if is_best:
        args.best_top1 = top1
        args.best_top5 = top5

    logger.info(f"top-1 acc: {top1:.2f}")
    logger.info(f"Best top-1 acc: {args.best_top1:.2f}")

    save_checkpoint(args, {
        'step': step + 1,
        'teacher_state_dict': teacher_model.state_dict(),
        'student_state_dict': student_model.state_dict(),
        'avg_state_dict': avg_student_model.state_dict() :
                                    if avg_student_model is not None else None,
        'best_top1': args.best_top1,
        'best_top5': args.best_top5,
        'teacher_optimizer': t_optimizer.state_dict(),
        'student_optimizer': s_optimizer.state_dict(),
        'teacher_scheduler': t_scheduler.state_dict(),
        'student_scheduler': s_scheduler.state_dict(),
        'teacher_scaler': t_scaler.state_dict(),
        'student_scaler': s_scaler.state_dict(),
    }, is_best)
```

9 Meta - Pseudo Labels

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

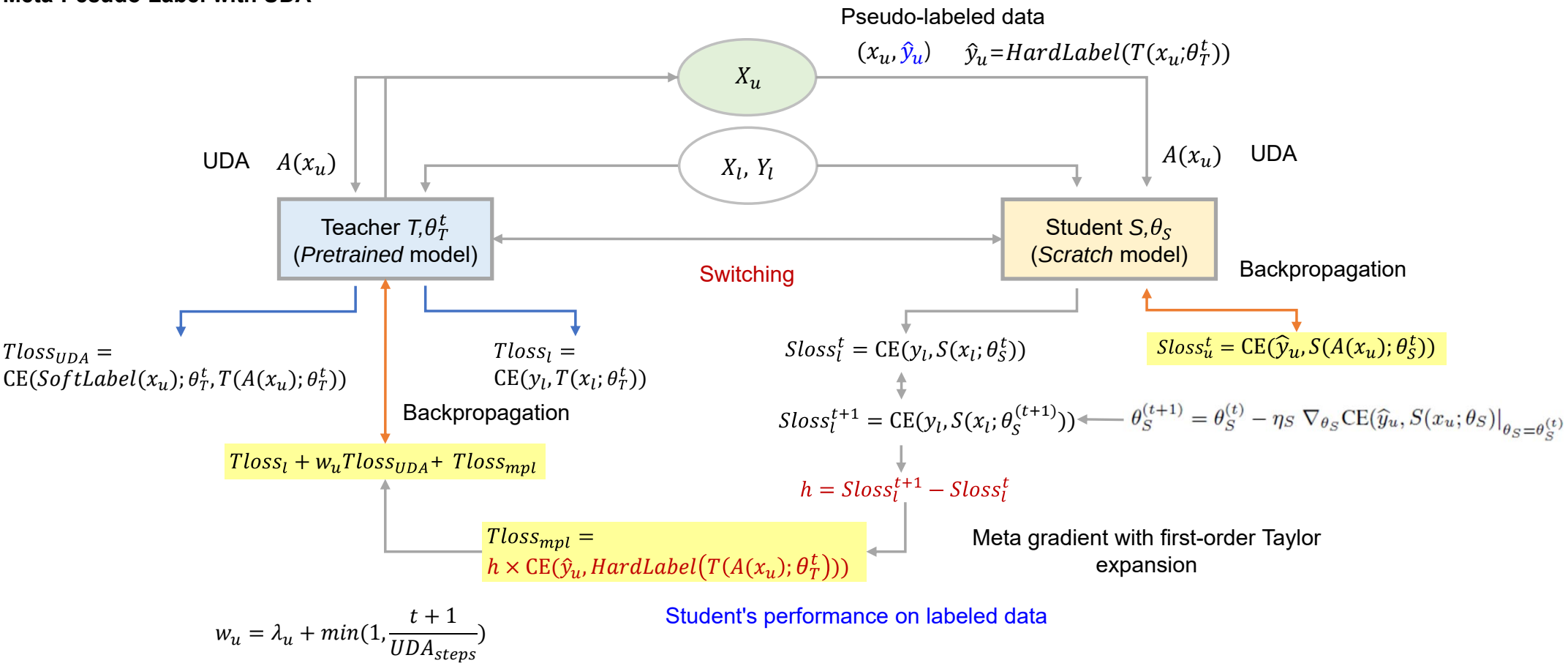
```
if args.local_rank in [-1, 0]:
    args.writer.add_scalar("result/test_acc@1", args.best_top1)
      wandb.log({"result/test_acc@1": args.best_top1})

# finetune
del t_scaler, t_scheduler, t_optimizer, teacher_model, labeled_loader, unlabeled_loader
del s_scaler, s_scheduler, s_optimizer
ckpt_name = f'{args.save_path}/{args.name}_best.pth.tar'
loc = f'cuda:{args.gpu}'
checkpoint = torch.load(ckpt_name, map_location=loc)
logger.info(f"=> loading checkpoint '{ckpt_name}'")
if checkpoint['avg_state_dict'] is not None:
    model_load_state_dict(student_model, checkpoint['avg_state_dict'])
else:
    model_load_state_dict(student_model, checkpoint['student_state_dict'])
finetune(args, finetune_dataset, test_loader, student_model, criterion)
return
```

9 Meta - Pseudo Labels

kekmodel/MPL-pytorch
https://github.com/kekmodel/MPL-pytorch/blob/main/main.py

Meta-Pesudo Label with UDA



9 Meta - Pseudo Labels

Experiments : Small Scale Experiments

- TwoMoon Dataset Experiment
  - TwoMoon dataset : 2 clusters each with 1,000 examples, 6 examples are labeled and others unlabeled
  - Teach / Student Model : Feed-forward fully-connected neural network with two hidden layers, each has 8 units

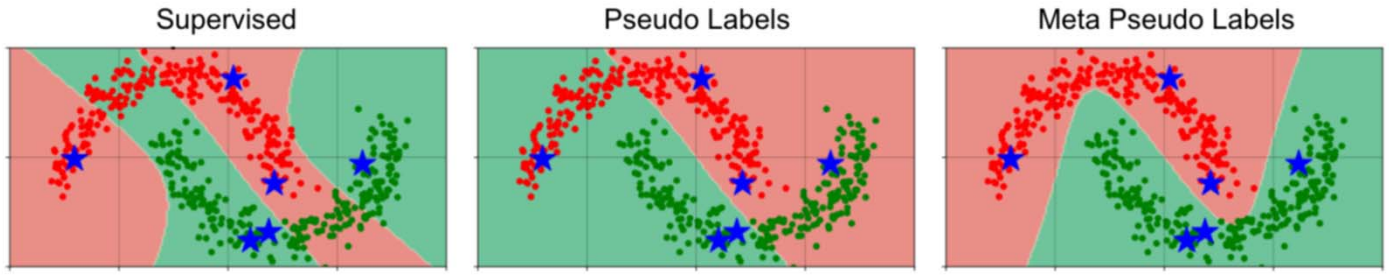


Figure 2: An illustration of the importance of feedback in Meta Pseudo Labels (right). In this example, Meta Pseudo Labels works better than Supervised Learning (left) and Pseudo Labels (middle) on the simple TwoMoon dataset. More details are in Section 3.1.

9 Meta - Pseudo Labels

Experiments : Small Scale Experiments

- CIFAR-10-4K, SVHN-1K, and ImageNet-10%
  - CIFAR and SVHN: WideResNet-28-2, Finetune 1000 steps with 512 batch
  - ImageNet: ResNet-50, Finetune 2000 steps with 512 batch

	Method	CIFAR-10-4K (mean ± std)	SVHN-1K (mean ± std)	ImageNet-10% Top-1	ImageNet-10% Top-5
Label Propagation Methods	Temporal Ensemble [35]	83.63 ± 0.63	92.81 ± 0.27	–	
	Mean Teacher [64]	84.13 ± 0.28	94.35 ± 0.47	–	
	VAT + EntMin [44]	86.87 ± 0.39	94.65 ± 0.19	–	83.39
	LGA + VAT [30]	87.94 ± 0.19	93.42 ± 0.36	–	
	ICT [71]	92.71 ± 0.02	96.11 ± 0.04	–	
	MixMatch [5]	93.76 ± 0.06	96.73 ± 0.31	–	
	ReMixMatch [4]	94.86 ± 0.04	97.17 ± 0.30	–	
	EnAET [72]	94.65	97.08	–	
	FixMatch [58]	95.74 ± 0.05	97.72 ± 0.38	71.5	89.1
	UDA* [76]	94.53 ± 0.18	97.11 ± 0.17	68.07	88.19
Self-Supervised Methods	SimCLR [8, 9]	–	–	71.7	90.4
	MOCOv2 [10]	–	–	71.1	–
	PCL [38]	–	–	–	85.6
	PIRL [43]	–	–	–	84.9
	BYOL [21]	–	–	68.8	89.0
	Meta Pseudo Labels	**96.11 ± 0.07**	**98.01 ± 0.07**	**73.89**	**91.38**
	Supervised Learning with full dataset*	94.92 ± 0.17	97.41 ± 0.16	76.89	93.27

Table 2: Image classification accuracy on CIFAR-10-4K, SVHN-1K, and ImageNet-10%. Higher is better. For CIFAR-10-4K and SVHN-1K, we report mean ± std over 10 runs, while for ImageNet-10%, we report Top-1=Top-5 accuracy of a single run. For fair comparison, we only include results that share the same model architecture: WideResNet-28-2 for CIFAR-10-4K and SVHN-1K, and ResNet-50 for ImageNet-10%. * indicates our implementation which uses the same experimental protocols. Except for UDA, results in the first two blocks are from representative important papers, and hence do not share the same controlled environment with ours.

9 Meta - Pseudo Labels

Experiments : ResNet50 + JFT

- Benchmark Meta pseudo labels on the labeled entire ImageNet dataset (25,000 examples) plus unlabeled images from JFT (12.8M images)

Method	Unlabeled Images	Accuracy (top-1/top-5)
Supervised [24]	None	76.9/93.3
AutoAugment [12]	None	77.6/93.8
DropBlock [18]	None	78.4/94.2
FixRes [68]	None	79.1/94.6
FixRes+CutMix [83]	None	79.8/94.9
NoisyStudent [77]	JFT	78.9/94.3
UDA [76]	JFT	79.0/94.5
Billion-scale SSL [68, 79]	YFCC	82.5/**96.6**
Meta Pseudo Labels	JFT	**83.2**/96.5

Table 3: Top-1 and Top-5 accuracy of Meta Pseudo Labels and other representative supervised and semi-supervised methods on ImageNet with ResNet-50.

9 Meta - Pseudo Labels

Experiments : Large Scale Experiment

- Architecture : EfficientNet-L2
- Datasets : Labeled entire ImageNet, Unlabeled JFT 130M (filtered down by Noisy student using confidence thresholds and up-sampling)

❖ Hybrid model parallelism

- A cluster of 2,048 TPUv3 cores – Divide into 128 identical replicas to run with standard data parallelism with synchronized gradients (2,048/128=16 cores)
- Two types of model parallelism : 1) Split 512x512 image into 16 patches, 2) Split each weight tensor into 16 parts

Table 4: Top-1 and Top-5 accuracy of Meta Pseudo Labels and previous state-of-the-art methods on ImageNet. With EfficientNet-L2 and EfficientNet-B6-Wide, Meta Pseudo Labels achieves an improvement of 1.6% on top of the state-of-the-art [16], despite the fact that the latter uses 300 million labeled training examples from JFT.

ResNet-50 [24]	26M	–	76.0	93.0	82.94
ResNet-152 [24]	60M	–	77.8	93.8	84.79
DenseNet-264 [28]	34M	–	77.9	93.9	–
Inception-v3 [62]	24M	–	78.8	94.4	83.58
Xception [11]	23M	–	79.0	94.5	–
Inception-v4 [61]	48M	–	80.0	95.0	–
Inception-resnet-v2 [61]	56M	–	80.1	95.1	–
ResNeXt-101 [78]	84M	–	80.9	95.6	85.18
PolyNet [87]	92M	–	81.3	95.8	–
SENet [27]	146M	–	82.7	96.2	–
NASNet-A [90]	89M	–	82.7	96.2	82.56
AmoebaNet-A [52]	87M	–	82.8	96.1	–
PNASNet [39]	86M	–	82.9	96.2	–
AmoebaNet-C + AutoAugment [12]	155M	–	83.5	96.5	–
GPipe [29]	557M	–	84.3	97.0	–
EfficientNet-B7 [63]	66M	–	85.0	97.2	–
EfficientNet-B7 + FixRes [70]	66M	–	85.3	97.4	–
EfficientNet-L2 [63]	480M	–	85.5	97.5	–
ResNet-50 Billion-scale SSL [79]	26M	3.5B labeled Instagram	81.2	96.0	–
ResNeXt-101 Billion-scale SSL [79]	193M	3.5B labeled Instagram	84.8	–	–
ResNeXt-101 WSL [42]	829M	3.5B labeled Instagram	85.4	97.6	88.19
FixRes ResNeXt-101 WSL [69]	829M	3.5B labeled Instagram	86.4	98.0	89.73
Big Transfer (BiT-L) [33]	928M	300M labeled JFT	87.5	98.5	90.54
Noisy Student (EfficientNet-L2) [77]	480M	300M unlabeled JFT	88.4	98.7	90.55
Noisy Student + FixRes [70]	480M	300M unlabeled JFT	88.5	98.7	–
Vision Transformer (ViT-H) [14]	632M	300M labeled JFT	88.55	–	90.72
EfficientNet-L2-NoisyStudent + SAM [16]	480M	300M unlabeled JFT	88.6	98.6	–
Meta Pseudo Labels (EfficientNet-B6-Wide)	390M	300M unlabeled JFT	90.0	98.7	**91.12**
Meta Pseudo Labels (EfficientNet-L2)	480M	300M unlabeled JFT	**90.2**	**98.8**	91.02

9 Meta - Pseudo Labels

Experiments : Ablations

- Meta Pseudo Labels with Different Amounts of Labeled Data

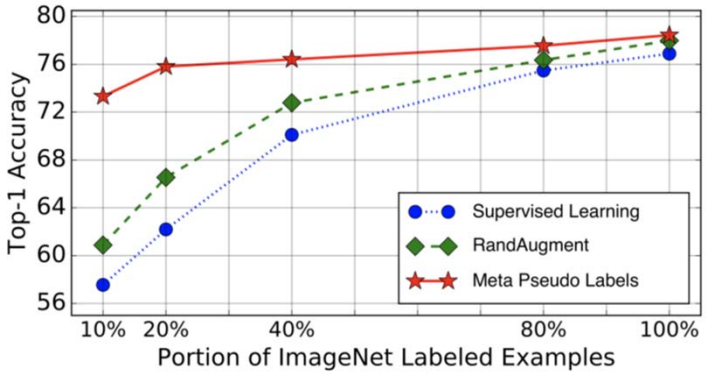


Figure 5: Performance of Supervised Learning, RandAugment, and Meta Pseudo Labels at different amounts of labeled examples.

- Meta Pseudo Labels with Different Training Techniques for the Teacher

Teacher	Pseudo-Labels	Mixup [85]	RandAugment
-Meta Pseudo Labels	83.79 ± 0.11	84.20 ± 0.15	85.53 ± 0.25
+Meta Pseudo Labels	$\mathbf{84.11 \pm 0.07}$	$\mathbf{84.81 \pm 0.19}$	$\mathbf{87.55 \pm 0.14}$

Table 10: Meta Pseudo Labels's accuracy for WideResNet-28-2 on CIFAR-10-4,000, where the teacher is trained with different techniques. All numbers are mean $\pm$ std over 10 runs

9 Meta - Pseudo Labels

Experiments : Ablations

- Meta Pseudo Labels Is a Mechanism to Address the Confirmation Bias of Pseudo Labels

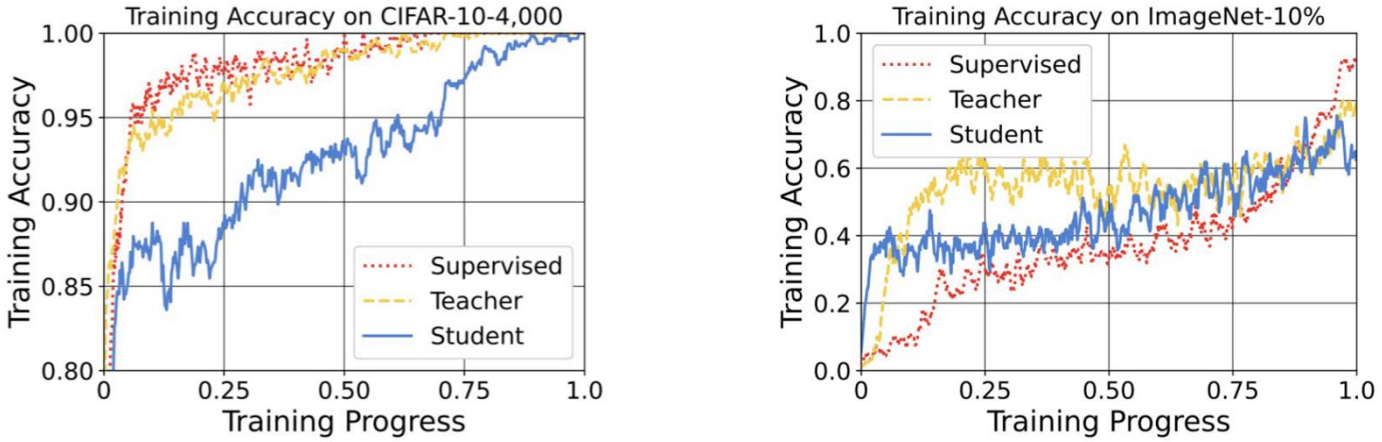


Figure 4: Training accuracy of Meta Pseudo Labels and of supervised learning on CIFAR-10-4,000 and ImageNet-10%. Both the teacher and the student in Meta Pseudo Labels have lower training accuracy, effectively avoiding overfitting

9 Meta Pseudo Labels

Experiments : Reduced Meta Pseudo Labels

- Compare Reduced Meta Pseudo Labels to NoisyStudent.
- In fact, the only difference between NoisyStudent and Reduced Meta Pseudo Labels is that Reduced Meta Pseudo Labels has a teacher that adapts to the student's learning state.

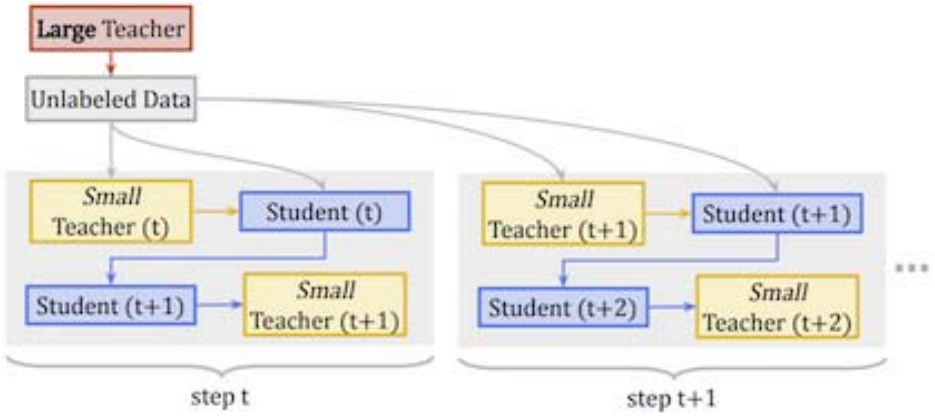


Figure 5: The ReducedMPL training procedure has 3 steps: (1) a large teacher q_{large} (red box) is pre-trained; (2) q_{large} assigns class distributions to the student's training data; (3) A small multi-layered perceptron q_Ψ calibrates the distributions computed by q_{large} to train the student. q_Ψ is trained along with the student, like the teacher in normal MPL.

Methods	CIFAR-10	SVHN	ImageNet
Supervised	97.18 ± 0.08	98.17 ± 0.03	84.49/97.18
NoisyStudent	98.22 ± 0.05	$\mathbf{98.71 \pm 0.11}$	85.81/97.53
Reduced Meta Pseudo Labels	$\mathbf{98.56 \pm 0.07}$	$\mathbf{98.78 \pm 0.07}$	**86.87/98.11**

Table 11: Image classification accuracy of EfficientNet-B0 on CIFAR-10 and SVHN, and EfficientNet-B7 on ImageNet. Higher is better. CIFAR-10 results are mean std over 5 runs, and ImageNet results are top-1/top-5 accuracy of a single run. All numbers are produced in our codebase and are controlled experiments.