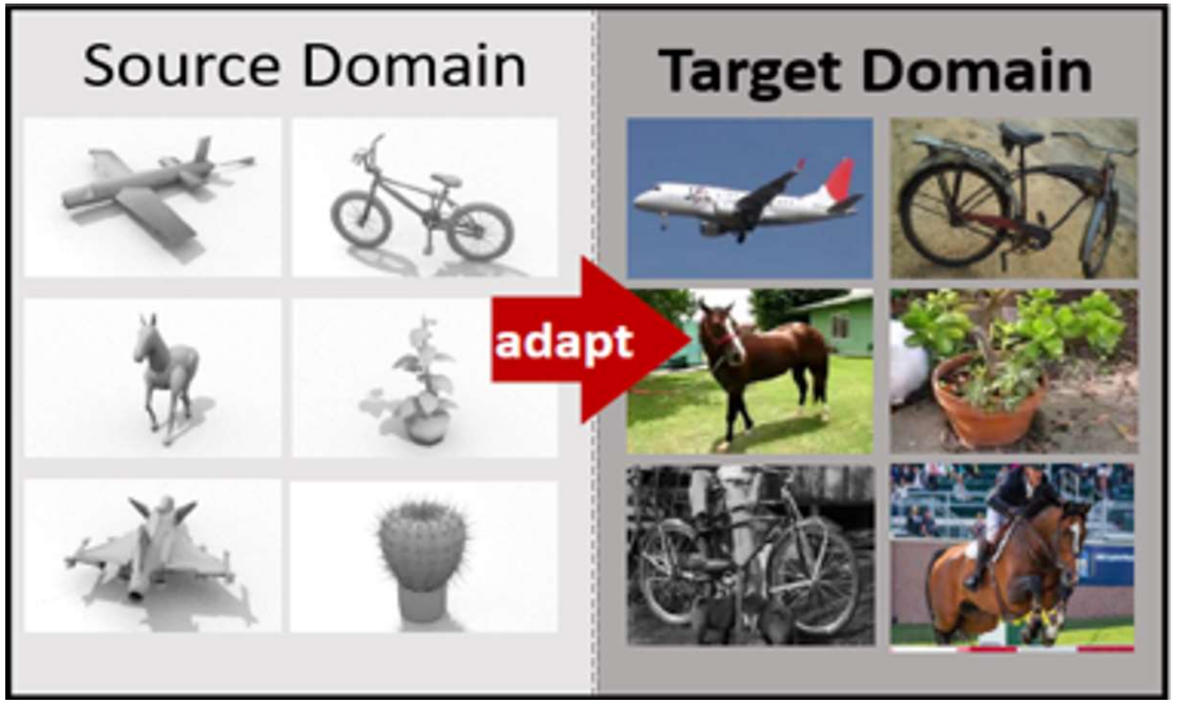
Semi-Supervised Domain Adaptation 연구 동향

소프트웨어학과 창경현

Contents

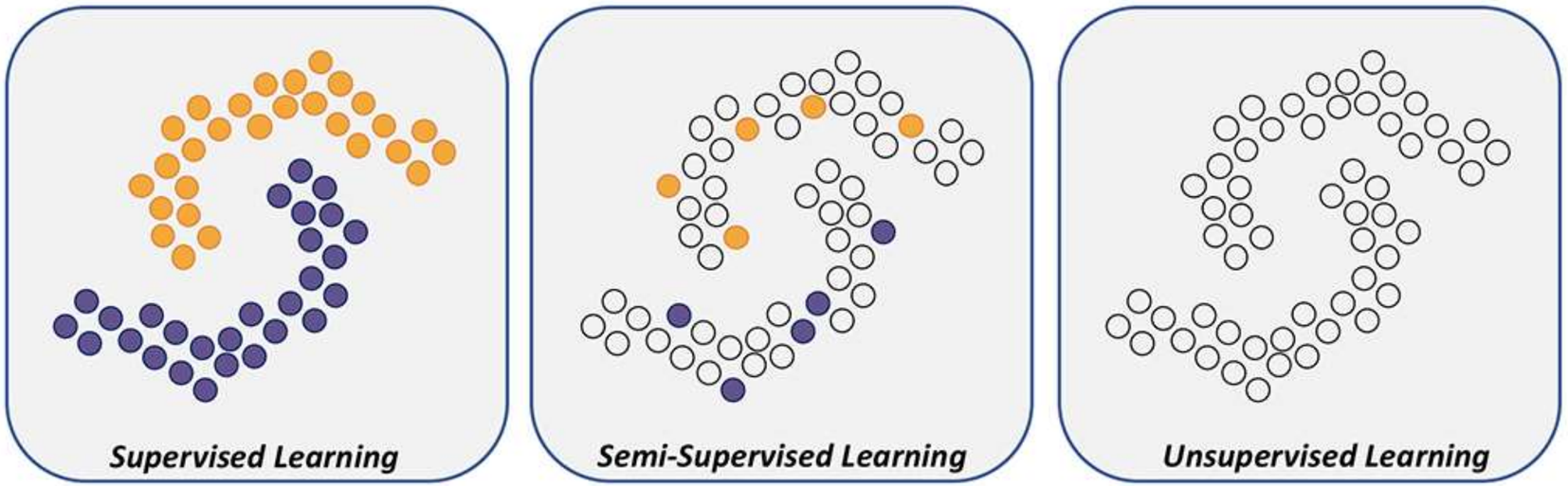
Domain Adaptation



레이블이 불충분하거나 존재하지 않는 목표 도메인에서 효과적으로 추론하는 모델을 학습하기 위해
레이블이 풍부하고 목표 도메인과 관련이 있는 소스 도메인을 이용하는 방법론

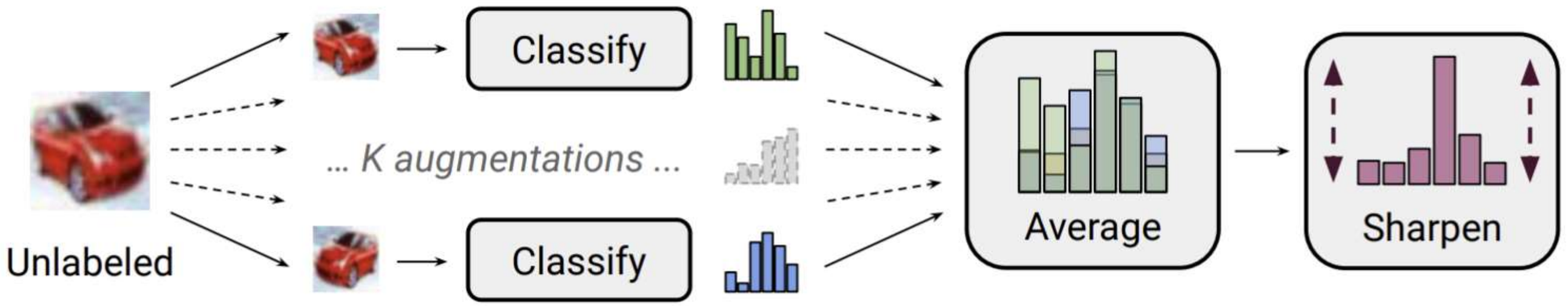
https://mcl.usc.edu/news/2018/12/16/mcl-research-on-domain-adaptation/

Semi-supervised learning



소량의 labeled data와 대량의 unlabeled data

MixMatch: A Holistic Approach to Semi-supervised Learning



Related works

Consistency Regularization: 데이터가 Augmentation되어도 예측 결과는 일관되게 유지해야 한다. (K개의 augmentation)

Sharpening: Soft Pseudo Labeling 라벨을 One-hot이 아닌 확률 분포로 생성한다.

MixUp: Augmented된 데이터들을 섞어서 새로운 샘플을 생성하고 예측에 활용한다.

MixMatch: A Holistic Approach to Semi-supervised Learning

Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (resp. unlabeled with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \text{Augment}(x_b)$ // *Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \text{Augment}(u_b)$ // *Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \text{p}_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$ // *Compute average predictions across all augmentations of* u_b
  $q_b = \text{Sharpen}(\bar{q}_b, T)$ // *Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ // *Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ // *Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ // *Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ // *Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ // *Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$

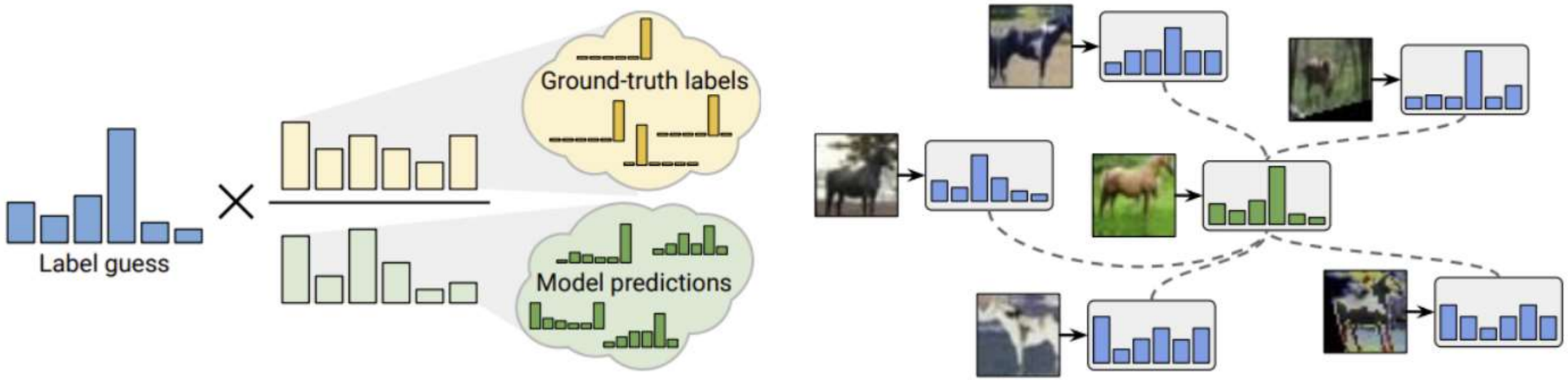
$$\mathcal{X}', \mathcal{U}' = \text{MixMatch}(\mathcal{X}, \mathcal{U}, T, K, \alpha)$$

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \text{H}(p, \text{p}_{\text{model}}(y \mid x; \theta))$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{L|\mathcal{U}'|} \sum_{u,q \in \mathcal{U}'} \|q - \text{p}_{\text{model}}(y \mid u; \theta)\|_2^2$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}}$$

ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring

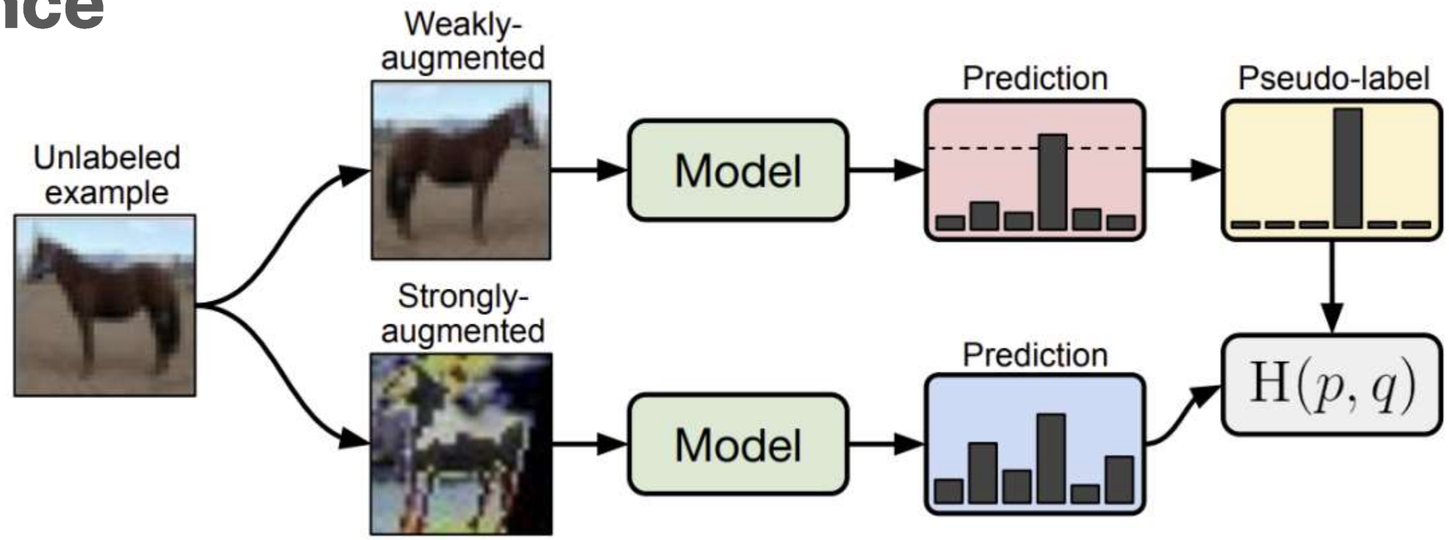


Weak & Strong Augmentation: K개의 Augmentation을 사용하지 않고, Weak와 Strong 두 가지 Augmentation을 도입. Weak Augmentation에서 레이블 데이터의 분포를 적용하여 의사 라벨링 진행.

Consistency Loss: Weak과 Strong Augmentation 사이의 일관성을 유지하기 위한 Consistency Regularization Loss 적용.

Distribution Alignment: Weak Augmentation에서 생성된 의사 라벨을 소스 도메인의 라벨 분포에 맞춰 정렬하여 모델의 예측 안정성 강화.

FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence



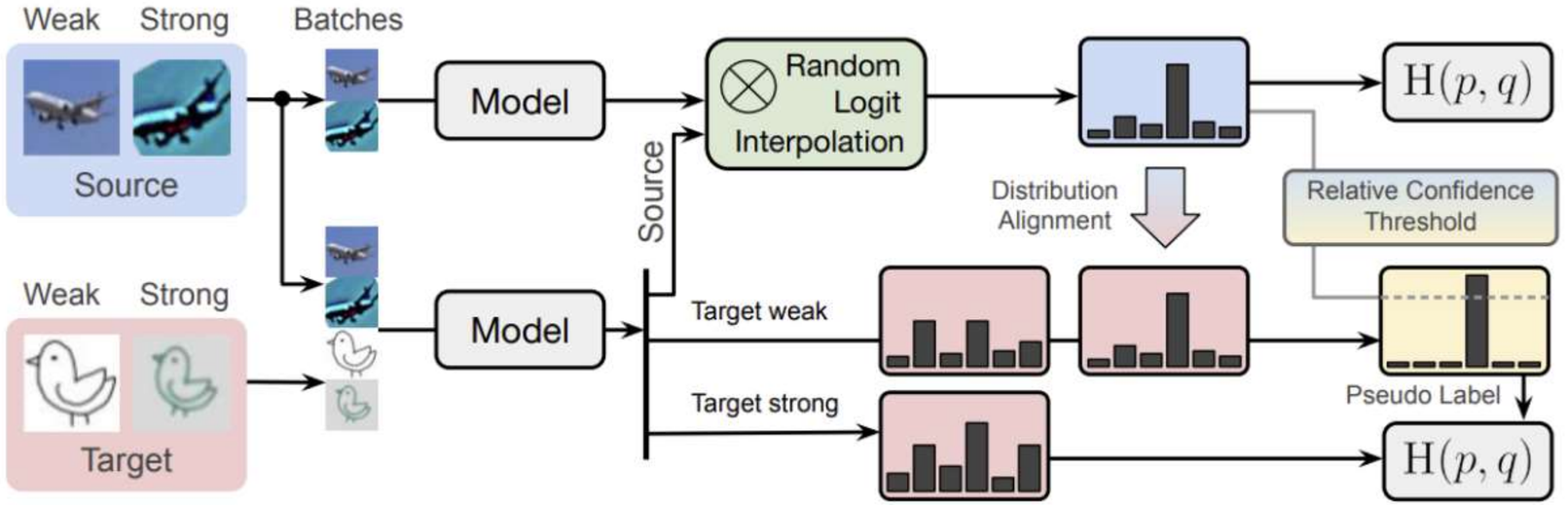
Weak Augmentation 데이터를 통해 One-hot Pseudo Labeling을 수행하고,

Strong Augmentation 데이터와의 일관성을 유지하기 위해 Cross Entropy Loss를 적용

Threshold가 존재하고, 이를 넘길 시에만 pseudo labeling 적용

Mixup을 적용하지 않고, labeled data와 unlabeled data가 별개로 작용

AdaMatch: A Unified Approach to Semi-Supervised Learning and Domain Adaptation



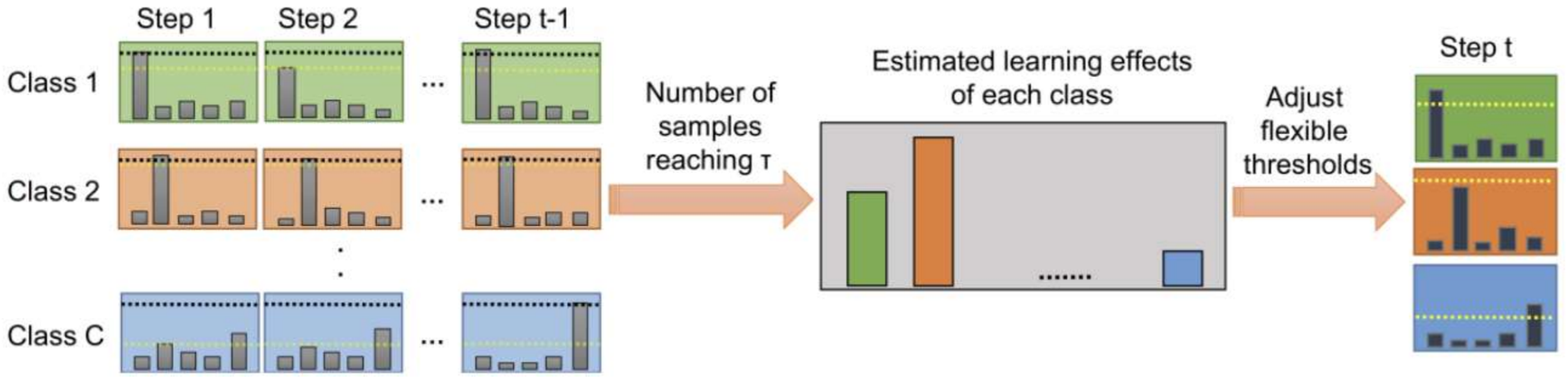
Weak & Strong Augmentation: Source와 Target 데이터 모두에 Weak 및 Strong Augmentation을 적용. (Input Batch)

Random Logit Interpolation: Source 데이터에 대해 서로 다른 배치를 모델에 통과시켜, 얻어진 logit 값들을 Random Logit Interpolation을 통해 조정.

Distribution Alignment: ReMixMatch와 유사하게 분포 정렬을 사용하지만, 실제 데이터 분포 대신 모델의 예측 분포를 사용.

Relative confidence threshold: 학습 초기의 낮은 예측 확률 문제를 해결하기 위해, 사용자 정의 임계값에 모델이 예측한 가장 높은 확률의 평균을 곱해 사용하는 상대적 임계값 도입. 학습이 진행됨에 따라 임계값이 점차 사용자 정의 값에 가까워짐.

FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling



기존 문제점: 기존 방법들은 각 클래스별 학습 난이도를 고려하지 않아, 학습이 쉬운 클래스는 과대 학습되고, 어려운 클래스는 학습에서 배제되는 문제가 발생. 이는 학습 데이터가 특정 클래스에 편향될 가능성을 높임.

클래스별로 학습 난이도를 구별하여 동적 Threshold을 도입. 각 클래스에 맞춰 다른 Threshold을 설정하여 학습의 균형을 맞춤.

Curriculum Pseudo Labeling: 학습 초기에는 각 클래스의 Threshold을 0에 가깝게 설정하고, 해당 클래스의 예측 신뢰도가 높아질수록 Threshold도 함께 증가시켜 학습 진행.

FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling

$$\sigma_t(c) = \sum_{n=1}^{N} \mathbb{1}(\max(p_{m,t}(y|u_n)) > \tau) \cdot \mathbb{1}(\arg\max(p_{m,t}(y|u_n) = c).$$

confidence값이 threshold값을 넘기면서
해당 클래스(c)로 예측된 개수

$$\beta_t(c) = \frac{\sigma_t(c)}{\max_c \sigma_t},$$

정규화 0-1 사이값으로
조정

$$\mathcal{T}_t(c) = \beta_t(c) \cdot \tau.$$

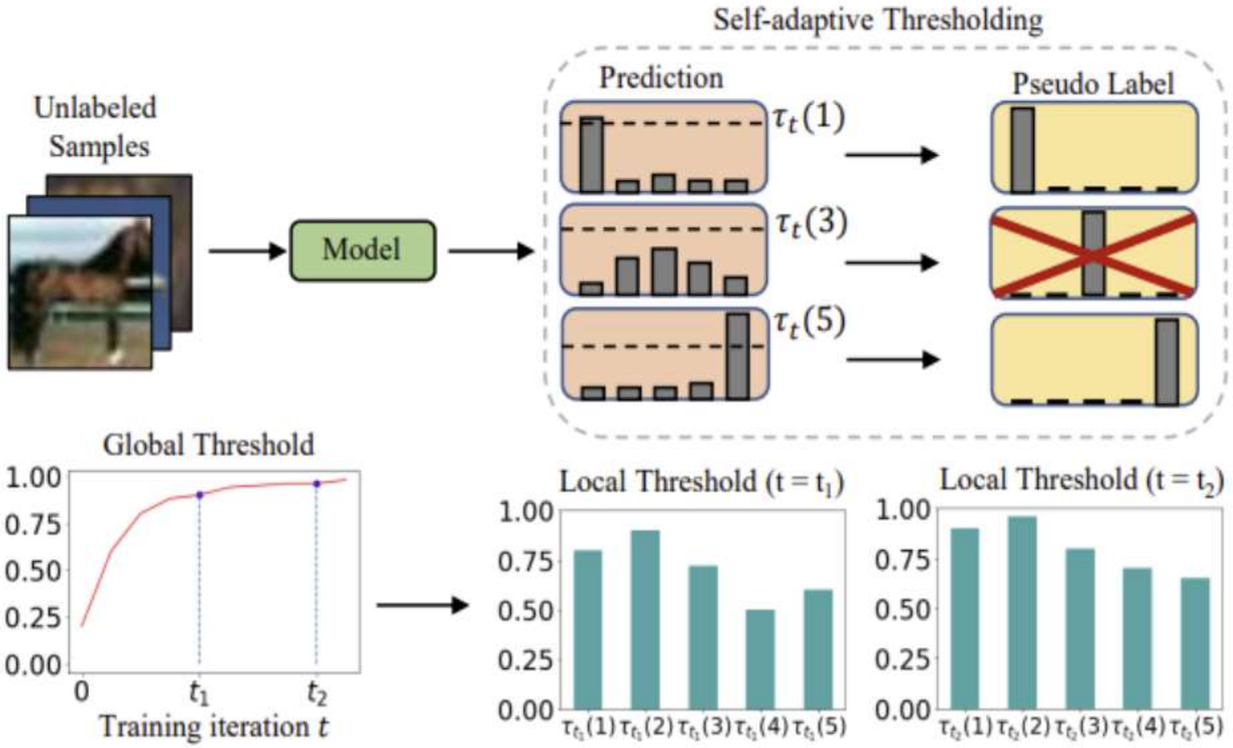
클래스별 treshold
조정.

$$\beta_t(c) = \frac{\sigma_t(c)}{\max\left\{\max_c \sigma_t, N - \sum_c \sigma_t\right\}},$$

학습에 활용되지 않는 unlabeled data 수

Threshold warm-up
모든 class의 threhold가 0부터 점진적
증가

FreeMatch: Self-Adaptive Thresholding for Semi-Supervised Learning



FlexMatch의 경우에는 고정한 global threshold와 adaptive local threshold를 사용함. 하지만 FreeMatch의 경우에는 global threshold 마저 adaptive하게 만듦

EMA를 통해 조금씩 treshold 업데이트

Self-Adaptive Fairness: Pseudo-label의 분포가 균형하지 않기 때문에 이를 균형하게 예측하도록 만들

FreeMatch: Self-Adaptive Thresholding for Semi-Supervised Learning

Unlabled data

	class 1	class 2	class 3	class 4	class 5	
	2000	2000	2000	2000	2000	
Pseudo label	998	999	1000	1001	1003	= 5001

$$\beta_t(c) = \frac{\sigma_t(c)}{\max\left\{\max_c \sigma_t, N - \sum_c \sigma_t\right\}},$$

5001 > 4999

$$\mathcal{T}_t(c) = \beta_t(c) \cdot \tau.$$

0.95

FreeMatch: Self-Adaptive Thresholding for Semi-Supervised Learning

$$\tau_t = \begin{cases} \frac{1}{C}, & \text{if } t = 0, \\ \lambda \tau_{t-1} + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B} \max(q_b), & \text{otherwise,} \end{cases}$$

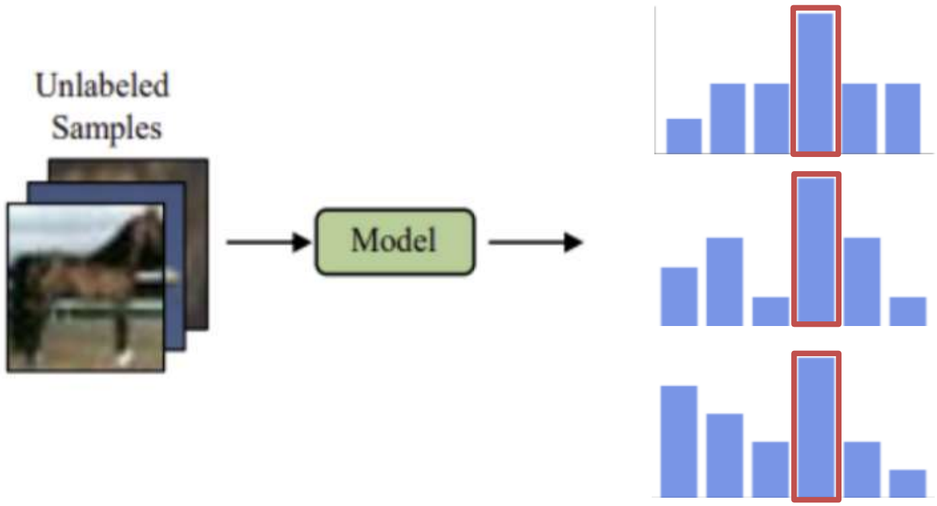
초기값은 1/전체 클래스의 개수

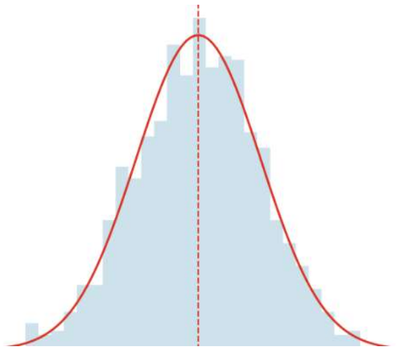
$$\tilde{p}_t(c) = \begin{cases} \frac{1}{C}, & \text{if } t = 0, \\ \lambda \tilde{p}_{t-1}(c) + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B} q_b(c), & \text{otherwise,} \end{cases}$$

$$\tau_t(c) = \text{MaxNorm}(\tilde{p}_t(c)) \cdot \tau_t = \frac{\tilde{p}_t(c)}{\max\{\tilde{p}_t(c) : c \in [C]\}} \cdot \tau_t,$$

이전 treshold 값과 현재 confidence의 값을 반영하여 treshold의 값을 업데이트 한다.

SoftMatch: Addressing the Quantity-Quality Trade-off in Semi-supervised Learning





예측 신뢰도에 대한 분포도

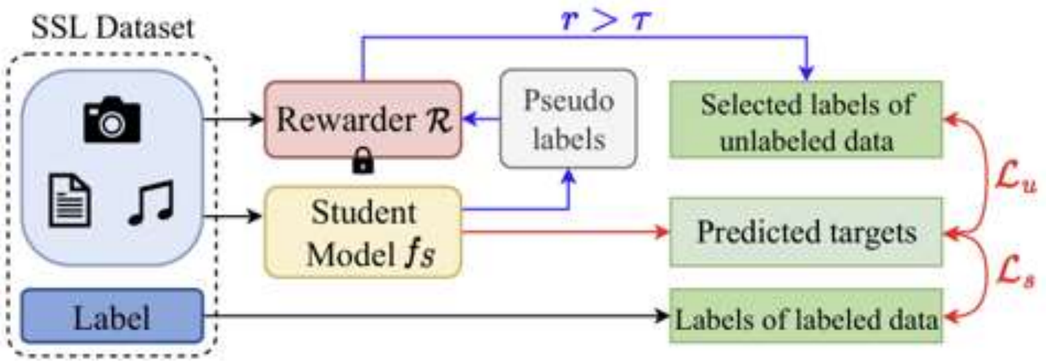
$$\lambda(\mathbf{p}) = \begin{cases} \lambda_{\max} \exp\left(-\frac{(\max(\mathbf{p})-\mu_t)^2}{2\sigma_t^2}\right), & \text{if } \max(\mathbf{p}) < \mu_t, \\ \lambda_{\max}, & \text{otherwise.} \end{cases}$$

$$L_s = \frac{1}{B_L}\sum_{i=1}^{B_L} H(y_i, p(y|x_{li}))$$

$$L_u = \frac{1}{B_U}\sum_{i=1}^{B_U} \lambda(p_i) H(\hat{p}_i, p(y|\Omega(x_{ui})))$$

$$L = L_s + L_s$$

SemiReward: A General Reward Model for Semi-supervised Learning



SemiReward: A General Reward Model for Semi-supervised Learning

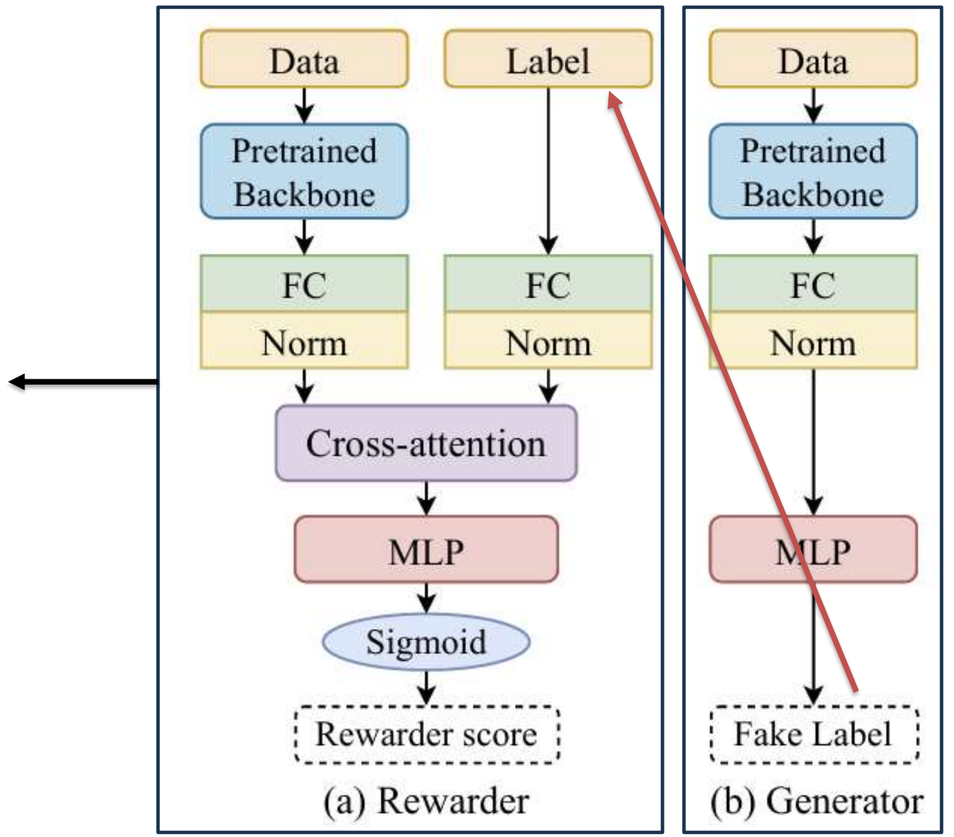
$$r(y^u, y^l) = \mathcal{S}(y^u, y^l) \simeq \mathcal{R}(x, y^u) \in [0, 1].$$

Ground truth label과 pseudo label(예측 값)은
유사해야 한다.
그 측정 값을 reward score로 정의

그리고 image와 해당 data로부터 생성된 pseudo label값으로 reward score를 대체하는 방식

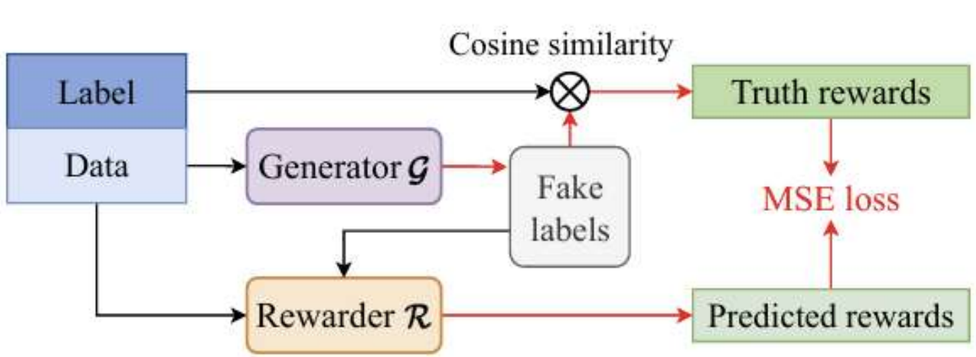
이렇게 reward를 훈련할 시, pseudo label과 image data의 유사도를 잘 검출해낼 수 있음

Pre-training Rewarder

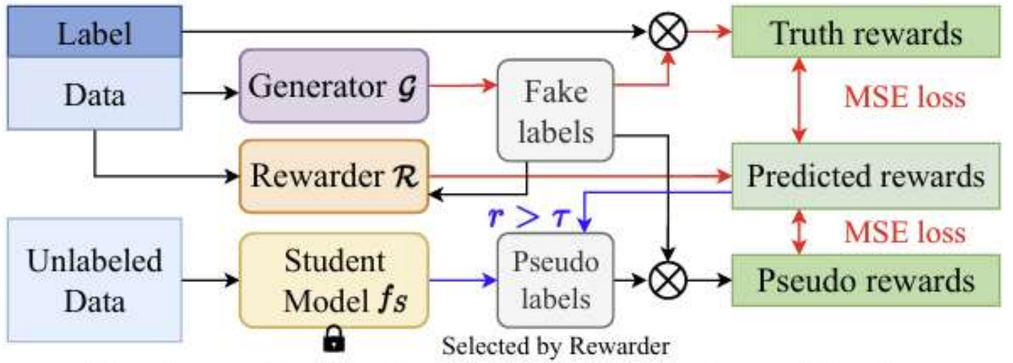


SemiReward: A General Reward Model for Semi-supervised Learning

Two-stage training paradigm of SemiReward



(a) Stage 1: Pre-training with labeled data



(b) Stage 2: Semi-supervised training with $\mathcal{D}_R$

Labeled data에서 rewarder 학습

unlabeled data에서 rewarder 학습

$$\mathcal{L} = \underbrace{\frac{1}{B_L}\sum_{i=1}^{B_L}\mathcal{H}\Big(y_i^l, f_S\big(\omega(x_i)\big)\Big)}_{\mathcal{L}_L} + \underbrace{\frac{1}{B_U}\sum_{j=1}^{B_U}\mathbb{I}(\mathcal{R}(x_j^u, y_j^u) > \tau)\mathcal{H}\Big(\hat{y}_j^u, f_S\big(\omega(x_j^u)\big)\Big)}_{\mathcal{L}_U} + \mathcal{L}_{\text{aux}},$$