

ADVERSARIAL TRAINING METHODS FOR SEMI-SUPERVISED TEXT CLASSIFICATION

Keywords : Adversarial training, virtual adversarial training

목차

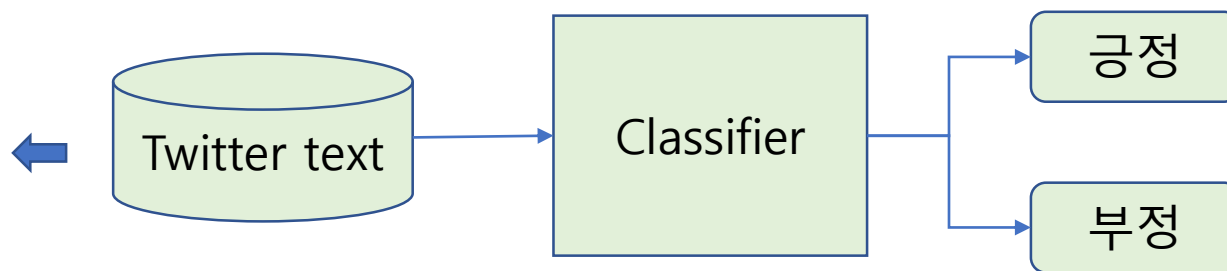
1. Class Imbalance Problem
2. Text Data Augmentation(...?)
3. Adversarial Training(논문)
4. Adversarial Training(연구)

1. Class Imbalance Problem !

Class Imbalance Problem???

- 태리짱 실물 개이쁨!! : 긍정
- 다이어트에 최고! ㅋㅋ : 긍정
- 게임 땀에 시험 망했네 : 부정
- 이러다 지각하겠다 ㅠㅠ : 부정

⋮



Twitter text 데이터의 비율이 [긍정 : 부정 = 50 : 50] 라면, 별 문제 없음.

Twitter text 데이터의 비율이 [긍정 : 부정 = 98 : 2] 이라면???

Class Imbalance Problem???

- 만약, 학습된 모델이 모든 데이터를 Positive 로 판단해도

맞춘 문제 = 980개 ,
틀린 문제 = 20개



평균 Accuracy : 98% !!!

반면, negative data 는 제대로 분류하지 못한다...



Recall : 100% , 0%

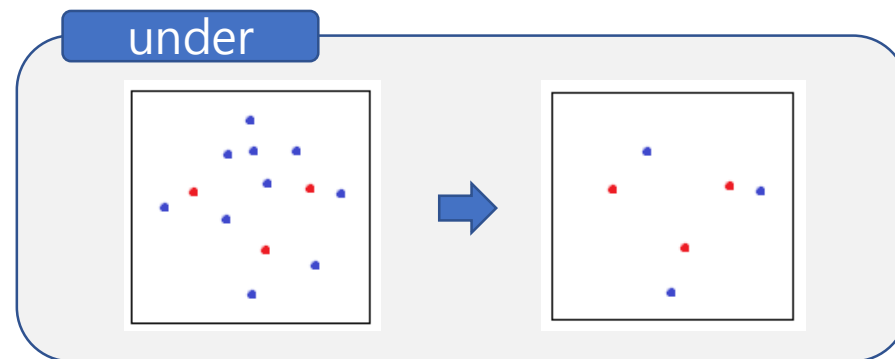
Precision : 98% , -%

- 실제로, 학습 데이터가 편향되어 있는 경우가 많아서 위와 같은 학습이 자주 발생한다 π...

기존의 Class Imbalance 보정 방법

Resampling Technique : 데이터 샘플링

1. Under-Sampling : Majority data 의 개수를 줄여 학습.



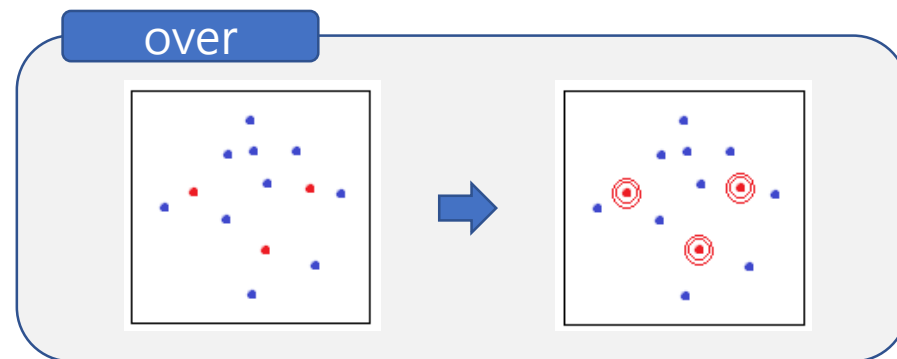
Ensemble Technique : 여러 학습 모델을 생성, 이용

기존의 Class Imbalance 보정 방법

Resampling Technique : 데이터 샘플링

1. Under-Sampling : Majority data 의 개수를 줄여 학습.
2. **Over-Sampling** : Minority data 를 여러 번 반복 학습.

Ensemble Technique : 여러 학습 모델을 생성, 이용



기존의 Class Imbalance 보정 방법

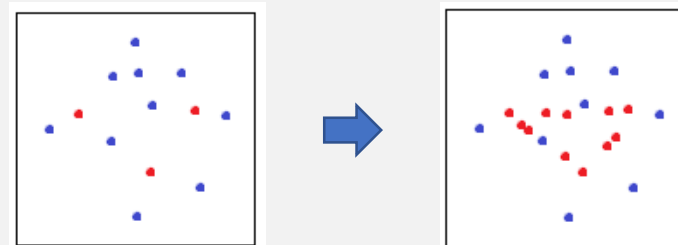
Resampling Technique : 데이터 샘플링

1. Under-Sampling : Majority data 의 개수를 줄여 학습.
2. **Over-Sampling** : Minority data 를 여러 번 반복 학습.
3. SMOTE : 서로 다른 Minority data 두 개를 이용하여, 새로운 synthetic instance 생성.

Ensemble Technique : 여러 학습 모델을 생성, 이용

Synthetic Minority Over-sampling Technique

SMOTE



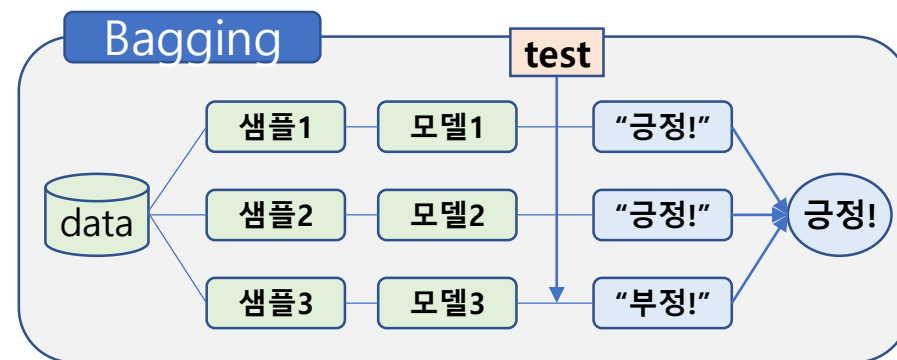
기존의 Class Imbalance 보정 방법

Resampling Technique : 데이터 샘플링

1. Under-Sampling : Majority data 의 개수를 줄여 학습.
2. **Over-Sampling** : Minority data 를 여러 번 반복 학습.
3. SMOTE : 서로 다른 Minority data 두 개를 이용하여, 새로운 synthetic instance 생성.
4. Both-Sampling(Under + Over), ROSE(SMOTE + bootstrapping) 등등...

Ensemble Technique : 여러 학습 모델을 생성, 이용

1. Bagging(Bootstrap Aggregating) : 샘플을 여러 번 뽑아, 샘플 별로 모델을 학습시킨 뒤, 다수 결과를 집계.



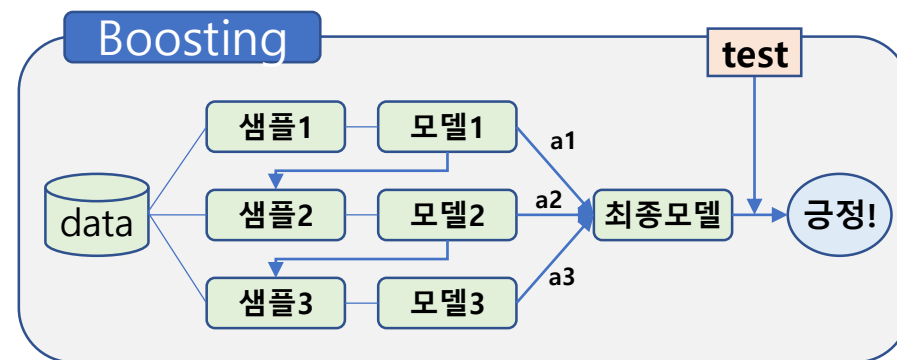
기존의 Class Imbalance 보정 방법

Resampling Technique : 데이터 샘플링

1. Under-Sampling : Majority data 의 개수를 줄여 학습.
2. **Over-Sampling** : Minority data 를 여러 번 반복 학습.
3. SMOTE : 서로 다른 Minority data 두 개를 이용하여, 새로운 synthetic instance 생성.
4. Both-Sampling(Under + Over), ROSE(SMOTE + bootstrapping) 등등...

Ensemble Technique : 여러 학습 모델을 생성, 이용

1. Bagging(Bootstrap Aggregating) : 샘플을 여러 번 뽑아, 샘플 별로 모델을 학습시킨 뒤, 다수 결과 집계.
2. Boosting : weak learner(오차율 50% 이하 모델) 들을 합쳐 하나의 strong learner 생성 및 결과 출력.



색다른 Class Imbalance 보정 방법??

- **Data Augmentation** : 기존 데이터 D 로부터, 새로운 데이터 D_{new} 생성
- **Adversarial Training** : 기존 데이터 D 에 noise를 가한 D' 를 학습 (데이터를 생성하지는 않는다!)

2. Text Data Augmentation (...?)

Text Data Augmentation

답장

전체답장

전달

🗑 삭제


스팸신고

안읽음

이동 ▼

...

번역

☆ 문제정의참조! 

▲

보낸사람

☆ Yong Suk Choi

>

받는사람

장두수

<

>, 조 수 필

<

>, 조 건희

<

아래 내용들을 모두 공유하고 필요한 부분에서 필요한 문제와 해결책을 제시하는데 참조하기 바란다.

- 1. Generative Adversarial Network를 이용한 진짜 같은 가짜 Data Augmentation

Text Data Augmentation

- **Data augmentation**

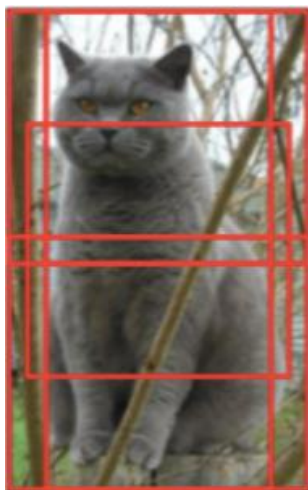
학습 데이터를 여러 방법으로 변형한 후,
변형된 데이터를 네트워크의 새로운 입력으로 사용
→ (와! 학습할 데이터가 많아졌어요!)



Original data



Horizontal flips



Random crops/scales



Color jittering



...



...

- 하지만... data augmentation... 이미지는 많이 하지만... 텍스트는 그다지...
- 왜?



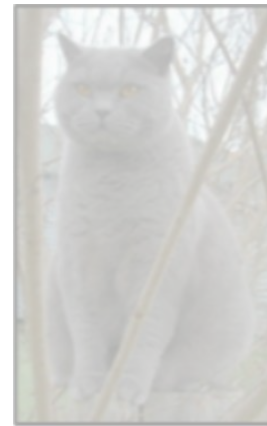
Original data



Horizontal flips



Random crops/scales



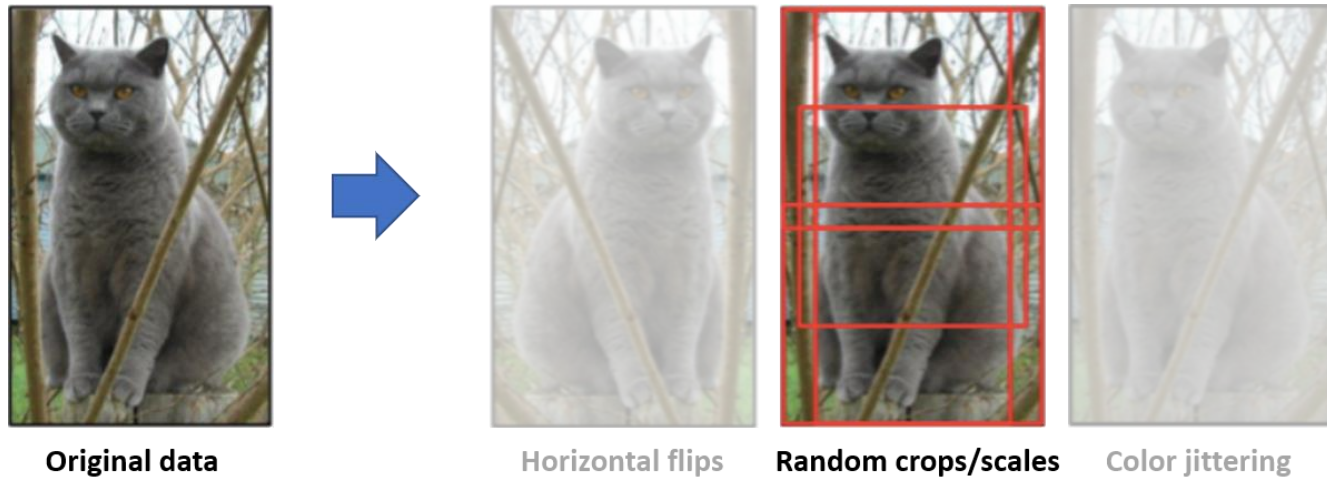
Color jittering

"I am a boy"



"boy a am I"

- 하지만... data augmentation... 이미지는 많이 하지만... 텍스트는 그다지...
- 왜?



"I am a boy" → "I am a" , "a boy" , ...

- 하지만... data augmentation... 이미지는 많이 하지만... 텍스트는 그다지...
- 왜?



“I am a boy” → “You are a girl” , “KimTaeri is my girlfriend” , ... (O)
 “I am three boy” , “I am a soccer”, ... (X)

text data augmentation 은 까다롭다...

ADVERSARIAL TRAINING

- 그렇다면,
 - **similar discrete word** 입력 대신,
(ex : I am a boy \rightarrow you are a girl \rightarrow [[0.124, -0.637, 0.394, ...] , [...] , [...] , [...]])
 - continuous word embedding**에 대한 **perturbation** 모델을 정의!
(ex : I am a boy \rightarrow [[0.157, -0.476, 0.354, ...] , ...] \rightarrow [[0.154, -0.469, 0.361, ...] , ...])

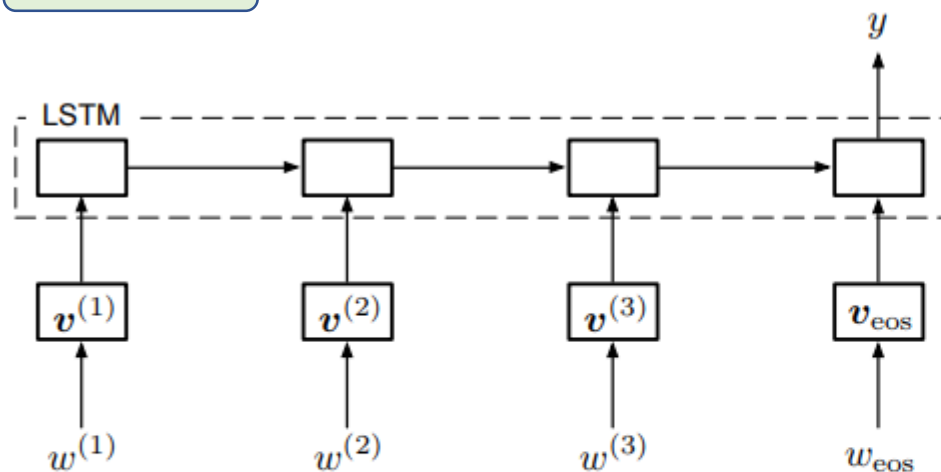
\rightarrow 이때, Adversarial Training Model의 핵심은 "적당한 perturb" 를 찾아내는 것!

ADVERSARIAL TRAINING (논문)

■ Adversarial Training

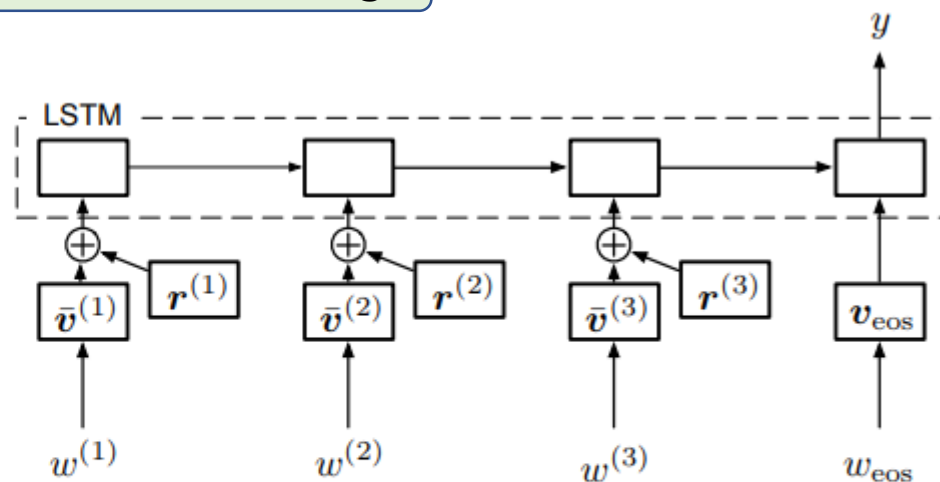
- Model

기존 LSTM



(a) LSTM-based text classification model.

Adversarial Training



(b) The model with perturbed embeddings.

Figure 1: Text classification models with clean embeddings (a) and with perturbed embeddings (b).

▪ Adversarial Training

- Loss Function

$$\text{Loss(cross entropy)} = -\log p(y \mid x; \theta)$$

↓ ↓ ↑
class data model

$$\text{Loss(adv)} = -\log p(y \mid \mathbf{x} + \mathbf{r}_{\text{adv}}; \theta) \text{ where } \mathbf{r}_{\text{adv}} = \arg \min_{\mathbf{r}, \|\mathbf{r}\| \leq \epsilon} \log p(y \mid \mathbf{x} + \mathbf{r}; \hat{\theta})$$

↓ ↓ ↓ ↑
class data noise model

유일한 차이점!

: 일정 범위(ϵ) 내의 “적절한” noise(= \mathbf{r}_{adv}) 를 찾아, 이를 input에 더해서 학습!

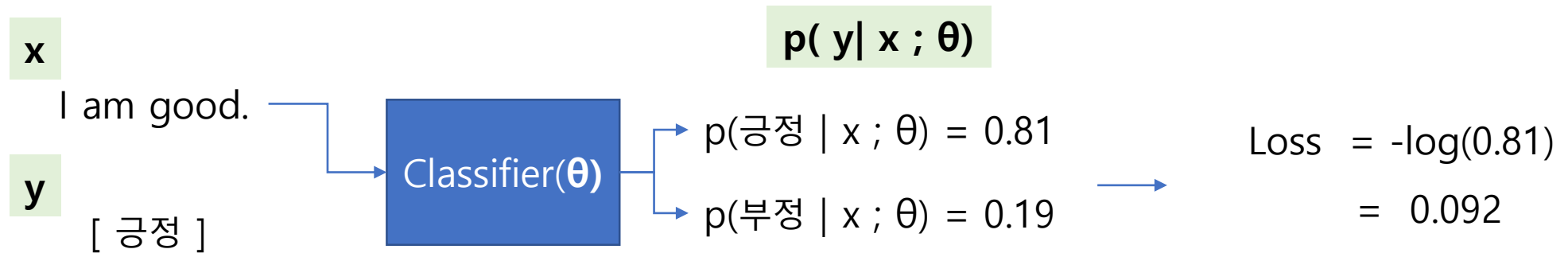
■ Adversarial Training

- Loss Function (1/2)

$$\text{Loss(cross entropy)} = -\log p(y \mid x; \theta)$$

↓ ↓ ↑
class data model

Ex) Emotion Classifier



요약 : "I am good" 을 넣으면, "I am good" 의 cross entropy 를 backpropagation!

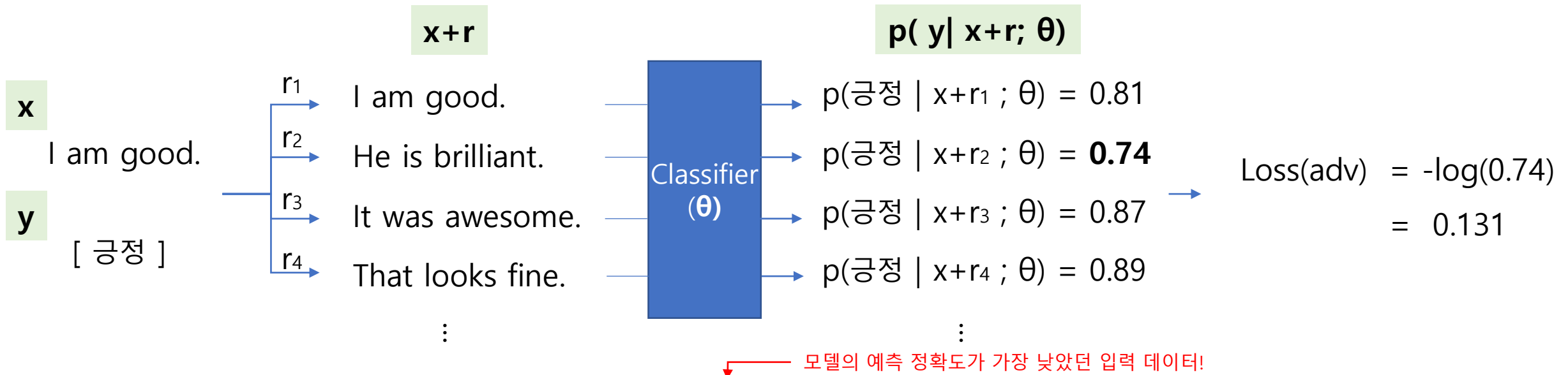
■ Adversarial Training

- Loss Function (2/2)

$$\text{Loss(adv)} = -\log p(y \mid \underset{\substack{\downarrow \\ \text{class}}}{x} + \underset{\substack{\downarrow \\ \text{data}}}{r_{\text{adv}}}; \underset{\substack{\downarrow \\ \text{noise}}}{\theta}) \text{ where } r_{\text{adv}} = \arg \min_{r, \|r\| \leq \epsilon} \log p(y \mid x + r; \hat{\theta})$$

model

Ex) Emotion Classifier



요약 : "I am good" 을 넣으면, "He is brilliant" 의 cross entropy 를 backpropagation!

■ Virtual Adversarial Training

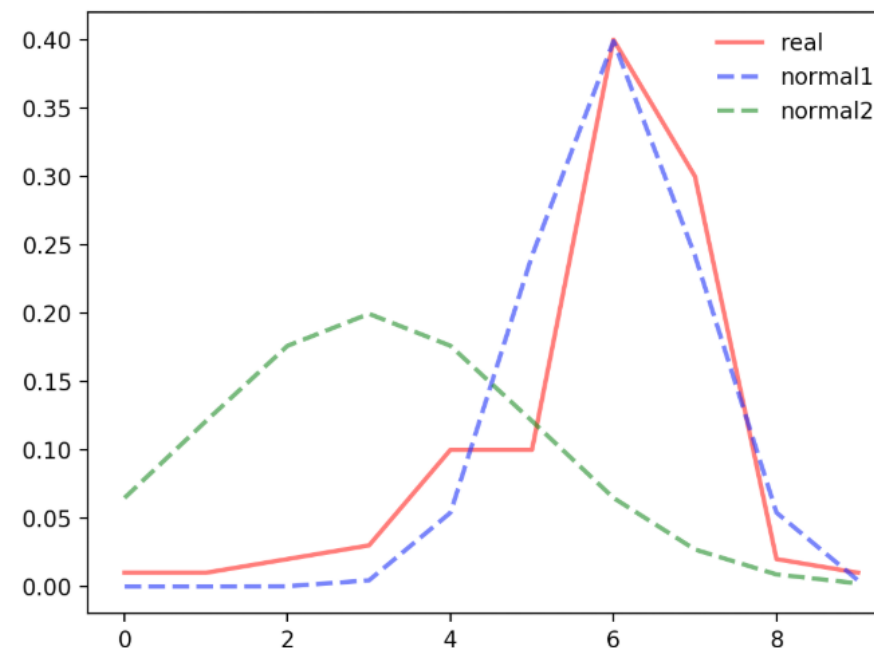
- Loss Function

$$\text{Loss}(\text{adv}) = \text{KL}[p(\cdot \mid \mathbf{x}; \hat{\boldsymbol{\theta}}) \parallel p(\cdot \mid \mathbf{x} + \mathbf{r}_{\text{v-adv}}; \boldsymbol{\theta})]$$

$$\text{where } \mathbf{r}_{\text{v-adv}} = \arg \max_{\mathbf{r}, \|\mathbf{r}\| \leq \epsilon} \text{KL}[p(\cdot \mid \mathbf{x}; \hat{\boldsymbol{\theta}}) \parallel p(\cdot \mid \mathbf{x} + \mathbf{r}; \boldsymbol{\theta})]$$

- KL (Kullback–Leibler divergence) 간단 소개

- 공식 : $D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$
- 의미 : 두 분포 간의 유사도. 유사할수록 값 작음.
- 예시 : $D(\text{초록} \parallel \text{빨강}) = 1.223$
 $D(\text{파랑} \parallel \text{빨강}) = 0.416$



요약 : 모델의 예측 정확도가 가장 낮았던 입력 데이터 의 KL Divergence 를 backpropagation!

- **Experiment & Result**

- 4가지 benchmark dataset에 대하여 Adversarial Training 시행.
- loss function = loss(cross entropy) + **loss(adv)** 로 정의!

- **Adversarial Training**

$$\text{loss(adv)} = -\log p(y \mid \mathbf{x} + \mathbf{r}_{\text{adv}}; \boldsymbol{\theta}) \text{ where } \mathbf{r}_{\text{adv}} = \arg \min_{\mathbf{r}, \|\mathbf{r}\| \leq \epsilon} \log p(y \mid \mathbf{x} + \mathbf{r}; \hat{\boldsymbol{\theta}})$$

- **Virtual Adversarial Training**

$$\text{loss(adv)} = \text{KL}[p(\cdot \mid \mathbf{x}; \hat{\boldsymbol{\theta}}) \parallel p(\cdot \mid \mathbf{x} + \mathbf{r}_{\text{v-adv}}; \boldsymbol{\theta})]$$
$$\text{where } \mathbf{r}_{\text{v-adv}} = \arg \max_{\mathbf{r}, \|\mathbf{r}\| \leq \epsilon} \text{KL}[p(\cdot \mid \mathbf{x}; \hat{\boldsymbol{\theta}}) \parallel p(\cdot \mid \mathbf{x} + \mathbf{r}; \hat{\boldsymbol{\theta}})]$$

- 결론 :
 1. 분류 성능 증가!
 2. V.A.T. 의 semi-supervised text classification 성능 준수 확인

ADVERSARIAL TRAINING (연구)

- 어...? 잠시만...?!

- “noise 포함 데이터”에 대한 loss를 학습하는 건 괜찮은 시도인 듯!

- 하지만, “원래 데이터”(noise X)에 대해서도 따로 학습을 진행해야 하는 것 아닌가?!

- (∵ 결국 중요한 건, 데이터 A를 넣었을 때 A를 잘 분류하는 모델을 만드는 것이므로!)

- > Adversarial Training을 적용하는 다양한 모델 학습 방식에 대한 탐구 진행!

- **연구 목표!**

(1) : 다양한 학습 방식에 따른 accuracy 확인

-> **Adversarial Training** 의 성능 검증


(2) : 학습 횟수에 따른 accuracy 확인


-> **Adversarial Training** 의
Data Augmentation 효과 검증

(3) : 다양한 classifier 에 대한 성능 검증

-> **Adversarial Training** 의 범용성 검증

- 분석을 위한 다양한 training 모델

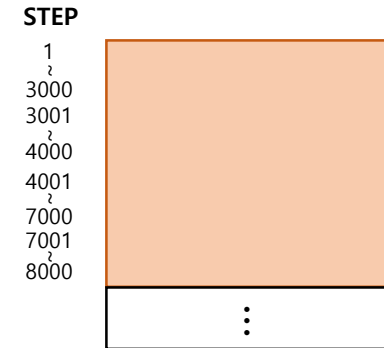
 : loss = loss(cross entropy) 사용.

 : loss = loss(cross entropy) + loss(adv) 사용.

- 모델 (1) : 일반적인 Classification 학습 모델.

$$\text{loss} = \text{loss}(\text{cross entropy})$$

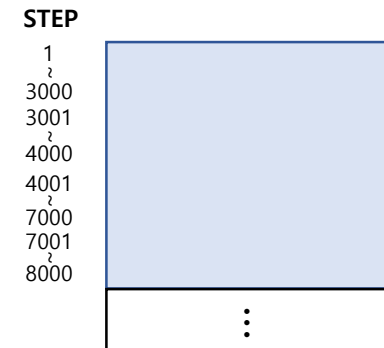
(A.T. 적용 X)



- 모델 (2) : 논문에서 제시한 Classification 학습 모델.

$$\text{loss} = \text{loss}(\text{cross entropy}) + \text{loss}(\text{adv})$$

(A.T. 적용 O)



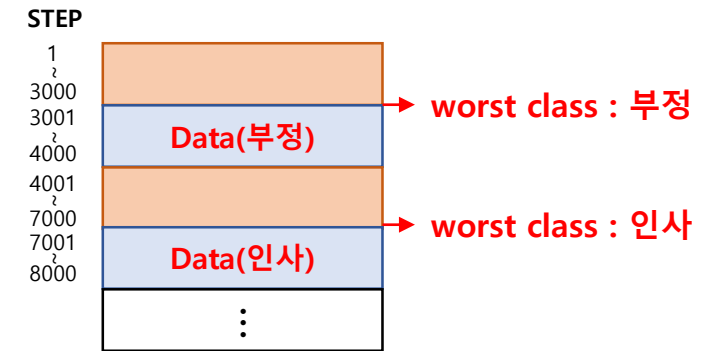
- 분석을 위한 다양한 training 모델

: loss = loss(cross entropy) 사용.

: loss = loss(cross entropy) + loss(adv) 사용.

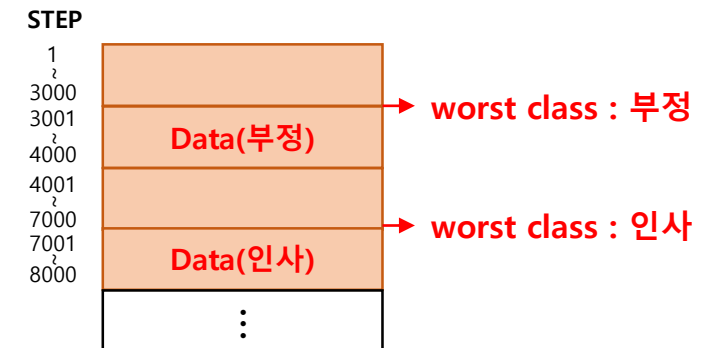
- 모델 (3) : "보충학습" 모델 (타겟 모델)

- 1~3000 step : loss = loss(cross entropy)
- 3001~4000 step : 가장 accuracy가 낮은 class에 대해서
loss = loss(cross entropy) + loss(adv)



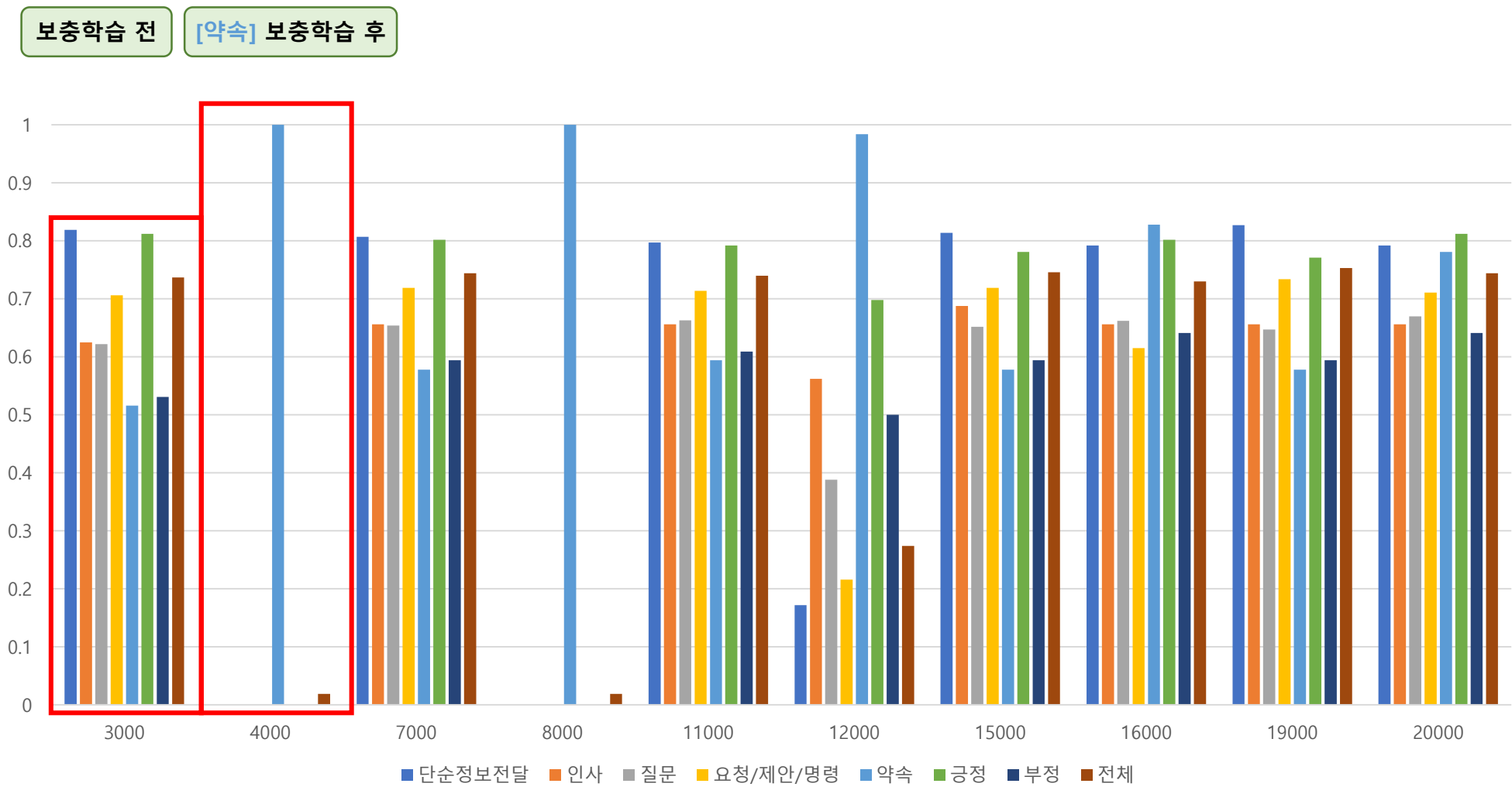
- 모델 (4) : 모델 (3)의 대조군. -> loss(adv) 의 검증을 위함.

- 1~3000 step : loss = loss(cross entropy)
- 3001~4000 step : 가장 accuracy가 낮은 class에 대해서
loss = loss(cross entropy) X



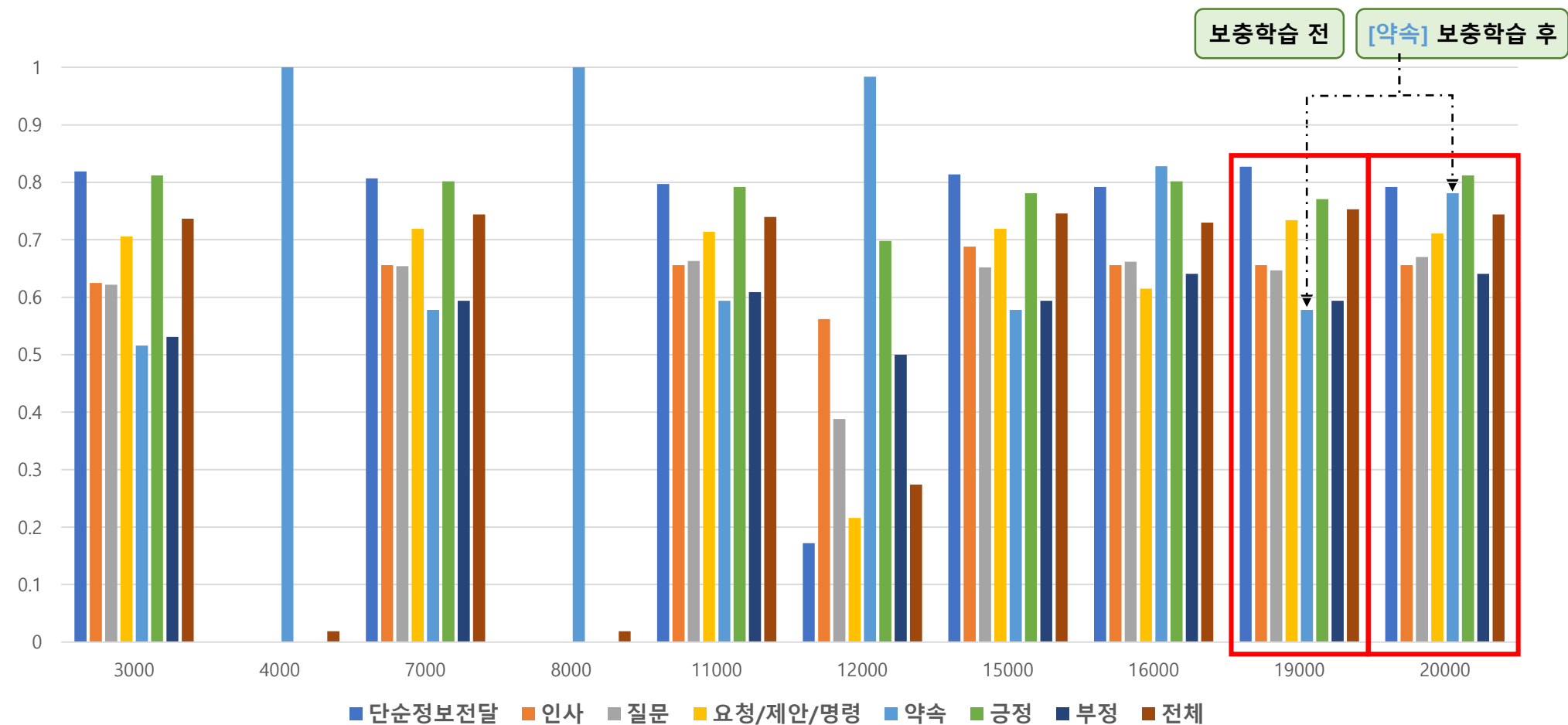
• 타겟 모델에 대한 훈련 결과

• 초반에는, 학습 효율 저하함.

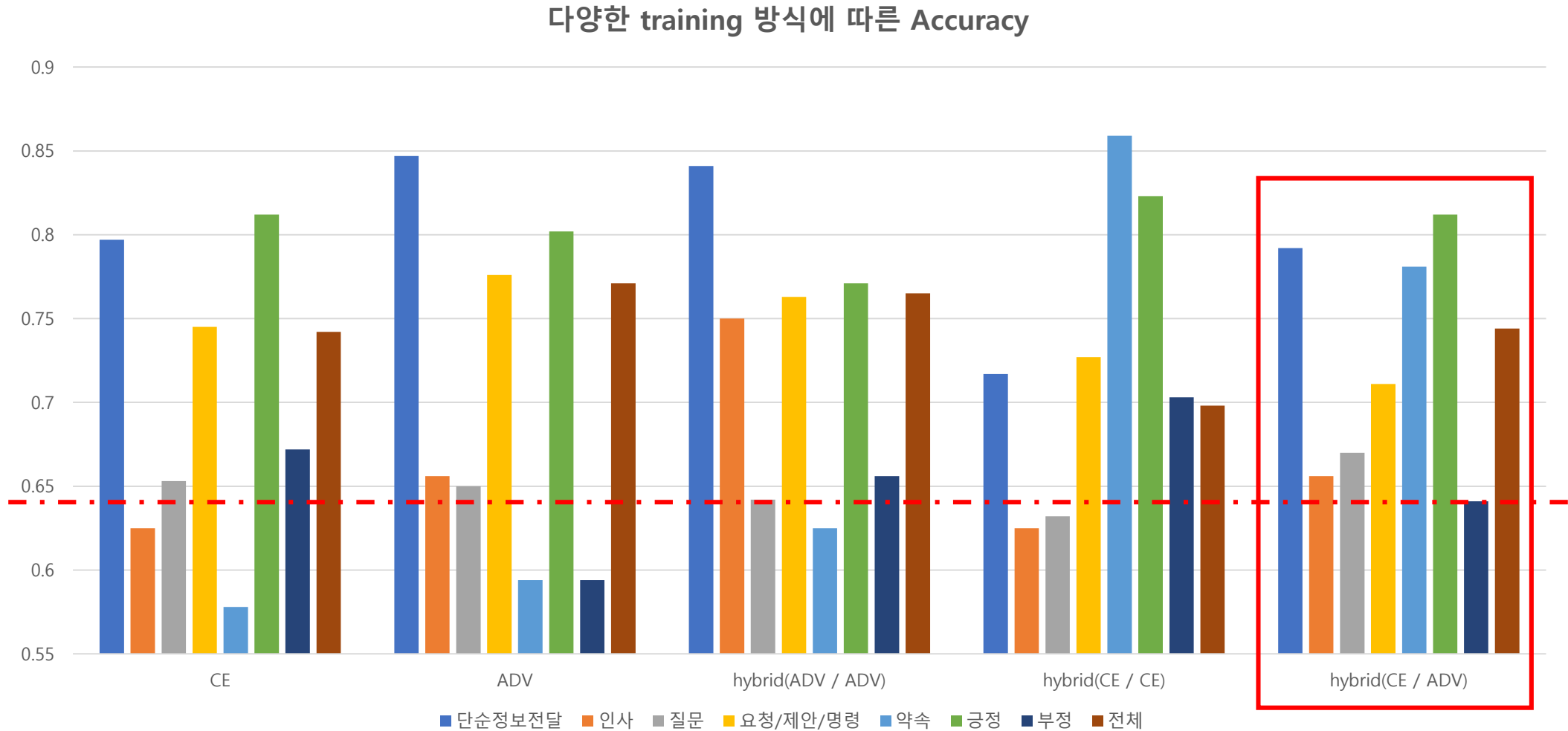


• 타겟 모델에 대한 훈련 결과

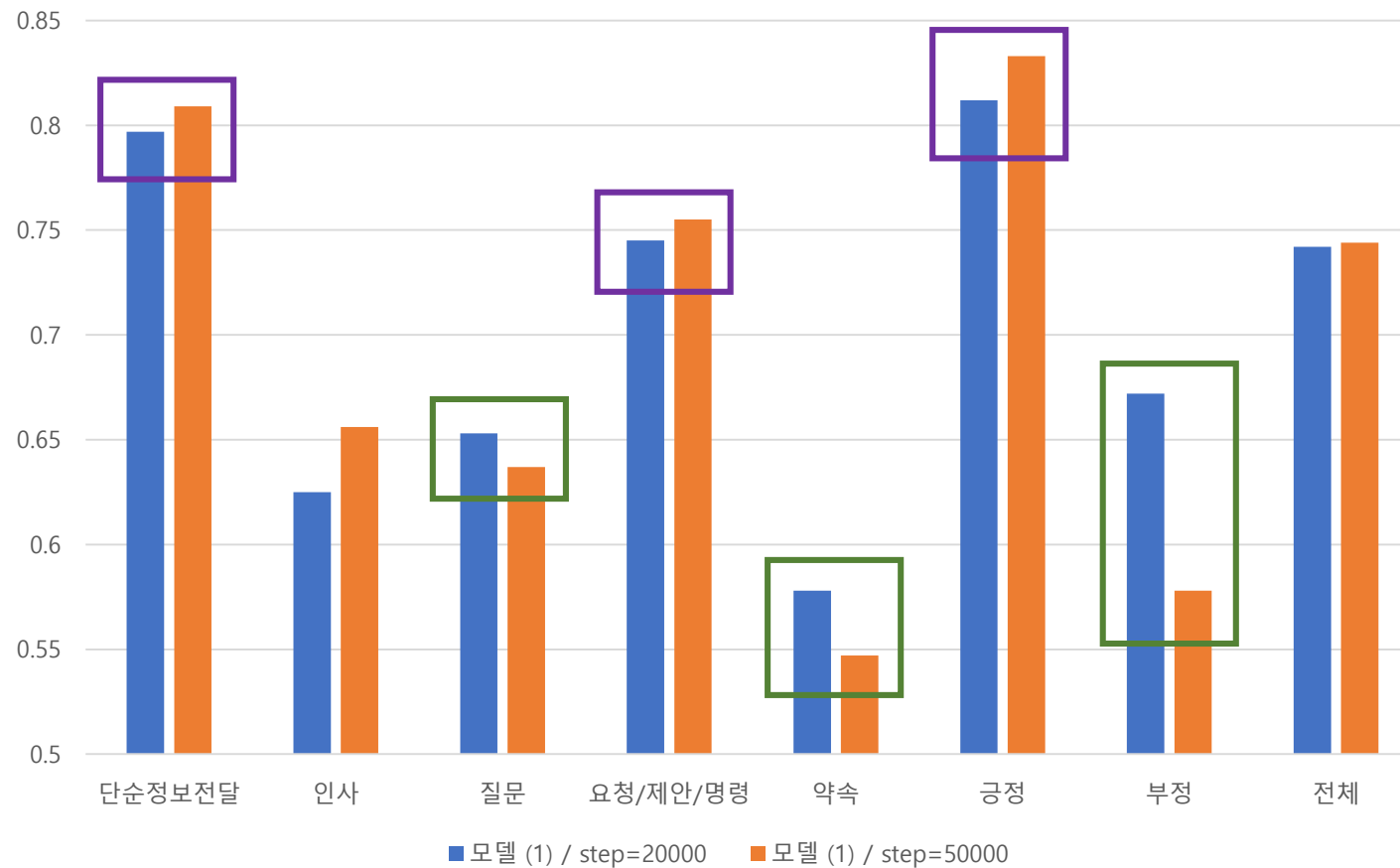
- 후반에는, "보충 학습" 한 Class에 대해 정확도를 높임.



- 다양한 training 방식에 따른 결과

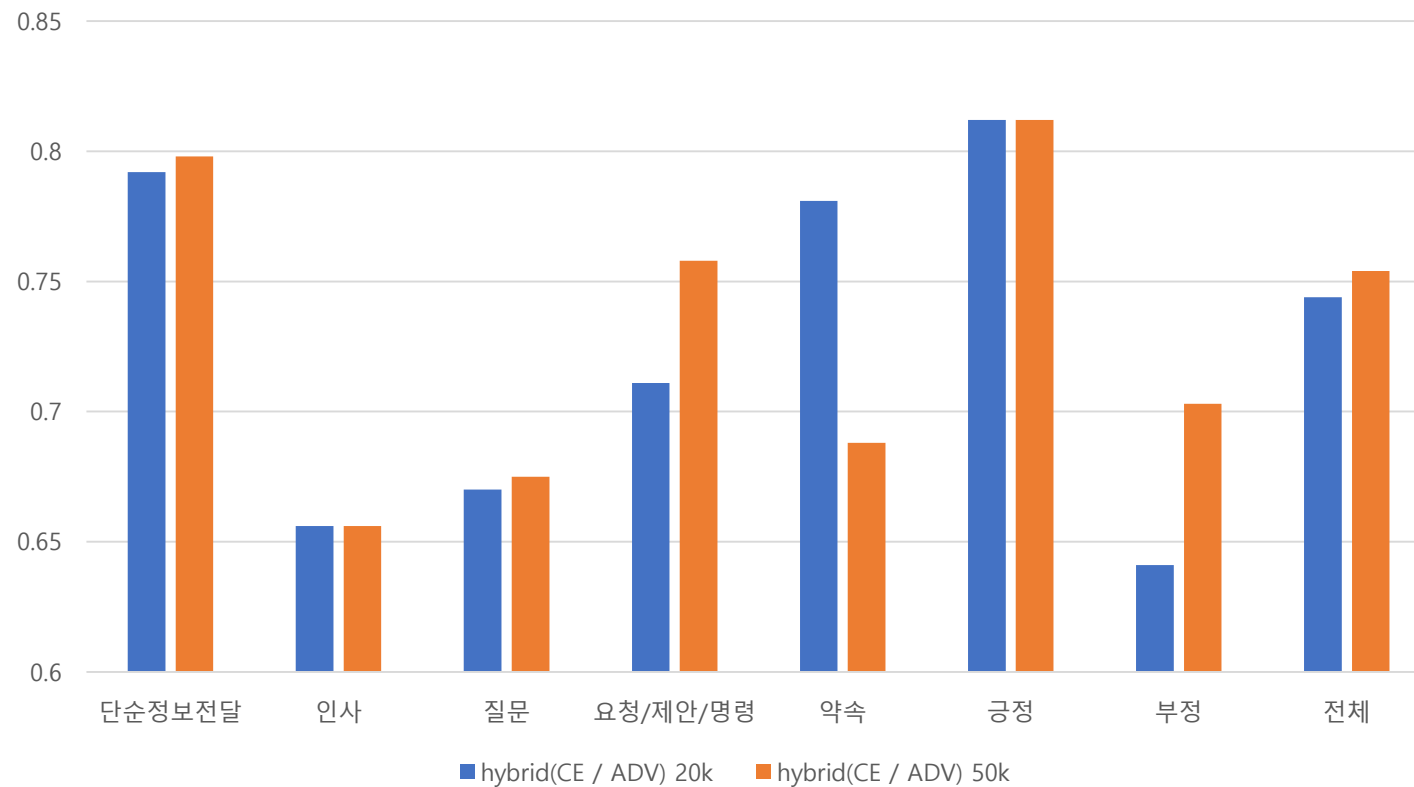


- 연구 목표 (2) : 학습 횟수에 따른 Adversarial Training의 성능 검증.
 - CE : 학습 횟수가 증가할 경우, 잘 인식하는 class는 더 잘 인식하고 잘 인식하지 못하는 class는 더욱 정확도가 떨어지는 현상 발생.

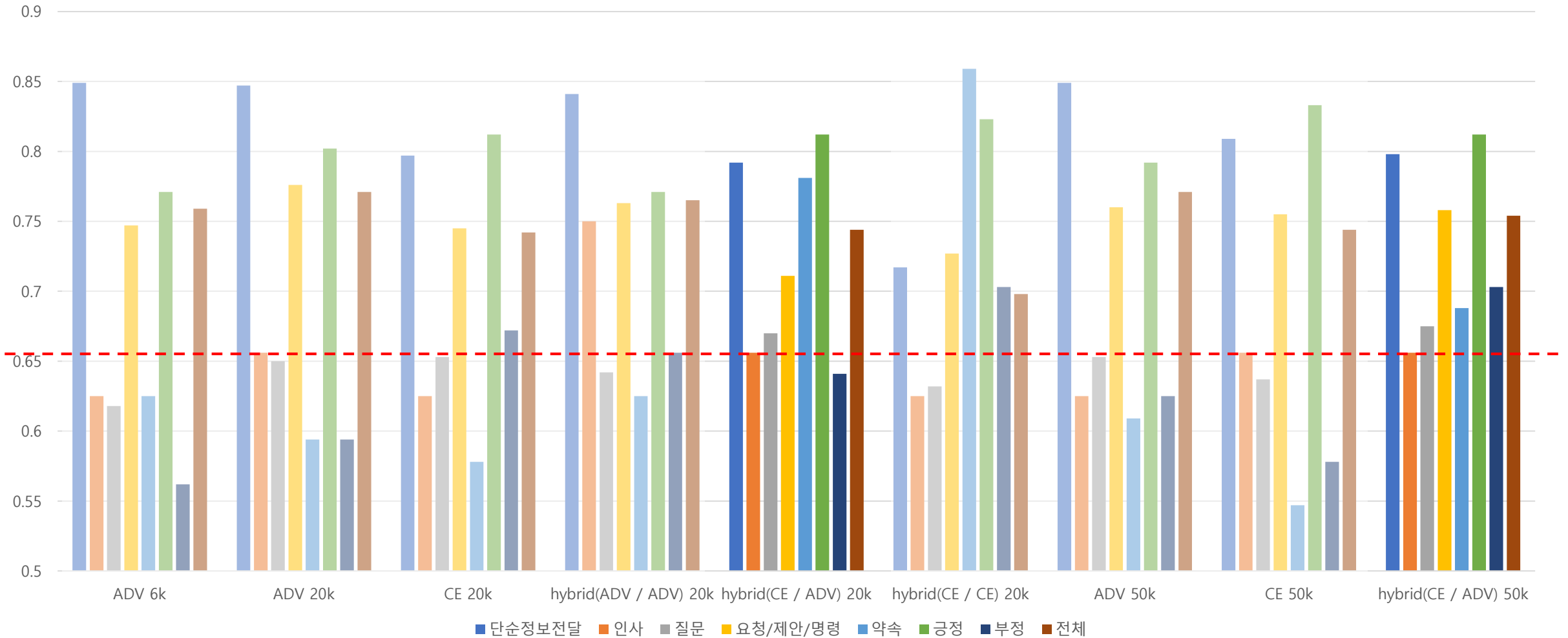


- 연구 목표 (2) : 학습 횟수에 따른 Adversarial Training의 성능 검증.

- hybrid : 모든 intent 의 classification accuracy 상승
([20k] 에서 과 보정된 "약속" 제외)



- 연구 목표 (2) : 학습 횟수에 따른 Adversarial Training의 성능 검증.



- 연구 결과!

(1) : 다양한 학습 방식에 따른 accuracy 확인

-> **Adversarial Training** 성능 준수함!

(2) : 학습 횟수에 따른 accuracy 확인

-> **Adversarial Training** 의
Data Augmentation 효과 검증됨!

(3) : 다양한 classifier 에 대한 성능 검증

-> **Future Work...**

Thank you!!