CV project 관련 여러 trials, 관련 기록들 최대한 자세하게
형식: 그냥 날짜랑 이름 적은다음에 자기가 적고싶은거 적기
내용은 자기가 한 실험 목적, 모델, hyperparameter 정보, … 이런거나 방향성 같은 부분들 넣으면 될듯

(동현 4/17)
요즘 하고있는것: 여러가지 최신 모델들 우리 데이타셋에서 사용가능하게 만들기.
MixMatch - 원래 지원됨.
Mean Teacher - 동현, 용민(4/10 - 4/12) 만들었으나 학습 과정에서 갑자기 에러나서 잘 안됨.
FixMatch - 동현
anEAT - 용민
simCLR - 수아

MixMatch 관련 실험들 -

FixMatch
현재 돌아가기는 하는데 실행속도 너무 느림. (아마 batchsize문제)
wide-resnet사용중인데 40-2 28-2 여러가지 크기 적용하면서 해보는중. (아마 너무오래걸려서 조금 잘나와도 이걸로 계속하기는 힘들거라고 생각함.)

생각하고 있는 방향성: 잘되는 모델 우선 찾아서 augmentation, pretrain, hyperparameter 여러가지 바꿔가면서 실험하기.

해보고싶은것 Mixmatch에다가 temperature sharpening 하지말고 threshold로 해보기(weight일정)
mixmatch에다가 다른 augmentation 테크닉 적용
mixmatch 하이퍼파라미터 바꾸는것.

(용민 4/17)
enaet 테스트

현재 Mixmatch가 사용하는 모델 : resnet18
현재 Mixmatch가 사용하는 pretrained 모델 : Imagenet(resnet 18)
RotNet -> resnet18로 pretrain시킨 후 Mixmatch 돌려보기

(동현 4/18)
Augmentation기술들 여러가지 해보려고 계획.
MixUp -> CutMix 로 바꿔서 해보는중
+ Resnet 18, 34, 50 세개 돌리고있음.

(동현 4/19)
Resnet 50 top1 67.4 top5 85.8(batch 50 lr 5e-4)
Resnet 34는 res18이랑 크게 달라진거 없음.

Resnet 50 돌릴때는 학습 좀 느려서 에폭 300까지도 키우는게 좋을듯함.

(용민 4/20)
session 177
trainset -> validation set : distribution 고르게 Resnet 50
lr: 5e-4, alpha: 0.75, lambda-u: 150 => 결과 : 68.617

(동현 4/20)
Trainset에서 label unlabel 비율 1:1로 들어가던거 2:5로 고침, 결과는 아직 확인 안함.
Rotnet Pretext task 어떻게 설계할지?

Data augmentation 기술: MixUP, CutMix, CutOut, randaugment 공부.
돌리고있는것: 30/60 아니면 40 단위로 cutout 사이즈 다르게 augment
fixmatch에서쓰는거?

(용민 4/20)
session 191
baseline (Mixmatch Resnet50) + Rotnet (1 rotated image per 1 unlabeled image)
lr : 5e-4, alpha: 0.75, labda-u : 150, lambda-rot: 1.0

session 192
[191 setting] + lambda-rot : 10.0

session 203
rotnet Res50으로 pretrain시키기 (lr: 5e-5)
rotlet loss cross entropy로 바꾸기

다음 할꺼
[session 191] + lambda-rot: 1.0 or 0.5, loss_rot: cross entropy

(동현 4/20)
지금까지한거(나중에레포트쓸떼)
새로운 모델 -> 용량부족해서 실패(Mean Teacher, Fixmatch, aneat)
기본모델 -> 새로운baseline(바뀐거 깃헙커밋에있음)
현재까지 정확도 거의 68.5나옴(이거 다시한번돌려봐야함)

여기서 지금
용민 -> rotnet추가
동현 -> augment더해보기

randaugment N,M = 2,9 2,5 두개 돌려보는중

(수아 4/21)
lr scheduling multistep 대신 exponential 써보기
main.py 303줄 scheduler = torch.optim.lr_scheduler.ExponentialLR(optimizer,gamma=0.1)

(동현 4/22)
FIxMatch에서 Threshold로 unlabeled data가 confident 하지않으면 사용 안하는거 해보고있는데, 0.9로 계속 하니까 그냥 비슷하게 나옴. unlabeled도 12프로 정도밖에 활용 안되고 학습이 멈춤에 따라 사용하는 unlabeled data의수도 늘어나지 않음.
threshold를 스케줄링 해야 할 듯.

동현
baseline model -> lr 스케줄링 있음.
loss, train acc계속 증가하지만 val acc는 수렴함. lambda-u 가 계속 늘어나니까 unlabeled를 활용 안하는건 아님.(69, 87)

FMM -> lr 스케줄링 있음, threshold 0.9
결과 자체는 비슷, 하지만 unlabeled data 활용도가 높지 않음(13/75), train acc하고 loss도 수렴하는 모습을 보임. 13/75개 정도 활용하고 50에폭에서 수렴하는 모습 보임(lr 스케줄 때문인거같음)

-> baseline에서 생각보다 unlabeled잘 활용못함. (13/75개밖에 안쓰는 FMM이랑 비슷하기 때문), 불확실한 unlabeled 도 억지로 가져와서 쓰기 때문?

-> FMM에서 threshold를 스케줄링하거나(0.9에서 점점 낮추기), lr스케줄링을 하지 말고 0.9로 해볼 필요가 있음. (사용하는 unlabel의 수를 점점 늘려서 쉬운 데이터 -> 어려운데이터 순서대로 점점 학습하게 만들기)

-> FMM에서 train에서도 수렴하는 이유가 뭘까? baseline에선 적어도 train은 계속 올라가면서 오버피팅 되는 모습을 보이는데..

(용민 4/23)
FIxmatch에서 threshold를 못넘는 unlabeled data에 대해 consistency regularization loss를 계산한 후 total loss로 합쳐보기. Pi model 자체의 성능이 Mixmatch보다 떨어지는것을 감안하면 여기서 추가적으로 무언가 더 해야할 듯 싶다. 아닌가

(용민 4/24)
- (100) : Fixmixmatch + Consistency loss
- (121) : Mixmatch + wide resnset 34 + shakeshake, lr = 5e-4
- (124) : lr = 5e-3
- Strong Augmentation + Consistency loss

1. 오버피팅 되는것.

< 스또리 >

모델 Res18
모델 Res50
label batchsize : unlabel batchsize = 3 : 7 (unlabel 데이터 양을 늘임)
Res50 + threshold 0.9
Res50 + theshold scheduling (0.9 → 0.5)

cross entropy 사용

(4) = 369
(5) = 371
(1) =372
(3) = 373
(2) = 374

optimizer : SGD로 고정

(79) Res50, (th: 0.9 → 0.5/느리게), (lr: 5e-4), (bs: [30, 75])
(348) efficient-b4, (th: 0.9 → 0.5/ 느리게), (lr: 5e-4 / [2,4,8], 0.5), (bs: [10,25])
(352) Res50, (th: 0.9 → 0.5/느리게), (lr: 5e-4), (bs: [30, 75]) + Augment

[model 비교]
(372) Res50, (th: TOP batchsize_u * acc / 100), (lr: 5e-4), (bs: [30,75])
(374) efficient-b4, (th: TOP batchsize_u * acc / 100), (lr: 5e-3 / [2,4,8], 0.5), (bs: [10, 25])
(373) efficient-b3, (th: TOP batchsize_u * acc / 100), (lr: 5e-4), (bs: [20,50])

[threshold policy 비교]
efficient-b4 environment: (lr: 5e-3 / [2,4,8], 0.5), (bs: [10,25]), (T: None)
(377) efficient-b4, (th: None, rampup)
(378) efficient-b4, (th: fixed 0.8)
(348) efficient-b4, (th: manual 0.9 → 0.5/ 느리게)
(387) efficient-b4, (th: manual 0.9 → 0.5/ 빠르게)
(374) efficient-b4, (%: train_acc)
(386) efficient-b4, (%: train_acc + th: 0.5)

efficient-b3 environment : (lr: 5e-4), (bs: [20,50]), (T: 0.5)
(405) efficient-b3, (th: None, rampup: lambda-u 0->150 in 20 epochs)
(407) efficient-b3, (th: fixed 0.9)
(408) efficient-b3, (th: fixed 0.8)
(430) efficient-b3, (th: fixed 0.8) + CE
(401) efficient-b3, (%: train_acc, th: 0.5)
(431) efficient-b3, (% train_acc, th: 0.5) + CE + last_acc_val update
(406) efficient-b3, (%: val_acc, th: 0.5)

(417) efficient-b3, (1~5 epoch → th: fixed 0.8,
6~300 epoch → %: train_acc, th: 0.5, ensure bs ratio)

(428) efficient-b3, (417) + last_acc_val update
(409) efficient-b3, (mixed loss => th: 0.9 + rampup)

(79) Res50, (th: 0.9 → 0.5/느리게), (lr: 5e-4), (bs: [30, 75])
(372) Res50, (th: TOP batchsize_u * acc / 100), (lr: 5e-4), (bs: [30,75])

[augmentation 비교]
(372) Res50, (th: TOP batchsize_u * acc / 100), (lr: 5e-4), (bs: [30,75])
(369) Res50, (th: TOP batchsize_u * acc / 100), (lr: 5e-4), (bs: [30,75]) + Rand(2,9)

(374) efficient-b4, (th: TOP batchsize_u * acc / 100), (lr: 5e-3 / [2,4,8], 0.5), (bs: [10, 25])
(371) efficient-b4, (th: TOP batchsize_u * acc / 100), (lr: 5e-3 / [2,4,8], 0.5), (bs: [10, 25]) + Rand(2,9)

(436) efficient-b3, (%: train_acc, th: 0.5), RA(2,3), lambda_u = 150
(437) efficient-b3, (%: train_acc, th: 0.5), RA(2,5), lambda_u = 150
(438) efficient-b3, (%: train_acc, th: 0.5), RA(2,5), lambda_u = 150

(448) efficient-b3, labeled + unlabeled (th: 0.8), AugMix
(449) efficient-b3, only supervised, AugMix

[실험 메모]
(378) fixed 0.8 => 80~90 epoch쯤에서 오버피팅 시작

일단 국룰은 efficientnet b4, SGD(decay 0.0004, momentum 0.9), lr (5e-3, [2,4,8] ½ ~ 5e-4), batchsize (10,25), lambda-u 150, MSE for unlabeled
이걸 썼다.

- 아예안하기(original mixmatch)
- Specific threshold value. (0.9, fixmatch)
- Schedule the threshold value along the training step => linear scheduling / depending on model's intermediate accuracy.
- Schedule the percentile of unlabeled data along the training step(curriculum labeling)
- Whether to add minimum or not

1. 아예안하기

2. 0.8?로처음부터끝까지 (unlabeled가 어느정도 많이 들어가는걸 보장해야하는데 어케함?)
3. Linear Scheduling 0.9 -> ? 미니멈 or not
4. 퍼센트스케줄링
5. 퍼센트스케줄링 + 미니멈

(*) 템포래쳐나 one hot 이거할껀지 말건지 이것도 … 나중에 해
(*) unlabeled이거 많으면(labeled적으면 어케될까..) 나중에 해

threshold X, unlabel weight rampup 할지 안할지, 단점 weight가 클때 train 방해 요소 일까?? ? ?
threshold O -> unlabel 데이터가 많으면 많을수록 좋지

- fixed
- moving
  - manual
  - adaptive
    1. percentage
       - train accuracy /val accuracy / mixed
       - try applying new hyper parameter (ex. percentage = acc * a)
       - applying mean
    2. threshold scheduling
    3. percentage + threshold schedulin지

1. 아예 안하기(semiloss 에서 lu scheduling 하는거 꼭 켜기) [377]
2. 0.8로 처음부터 끝까지(lu 150) [378]
3. 퍼센트인데 0.5보다는 큰것들로만(lu 150) [386]
4. 0.9->0.5 고정된 스케줄링 [387]

1 아예안하는거랑(걍 믹스매치, semiloss 바꿔야됨 16000step이니까 20에폭으로) we linearly ramp up λu to its maximum value over the first 16,000 steps of training as is common practice

2 adaptive 퍼센트로하는거랑 -> 여기서 더 좋은방법이 있을까? 더 발전시킬수 있는 방법

3 0.8 strict thresholding하는거

이렇게 세개가 좀 제일 다른거 세개인거같음.

*바꿀꺼***
labeled loss unlabeled loss 비교? 이거도 nsml.report로 찍기
step이 정수일때 그래프 찍지 말기(만약 튀는거 원인 못찾으면)

b3로 바꿀거면 20,50? 좀더 크게 테스트?

지금하고싶은게 약간
1번은 처음에 너무 안되는데 나중가면 약간 다 쓸수있으니까 오버피팅 줄일수있고
3번은 첨엔 잘돼도 한계 있으니까
2번으로 뭐 잘 조절하면 이게 최적이다 이건가?
아니면 초반에 3번 후반에 1번 뭐 할튼 둘이 잘 섞는게 중요하지 ㅇㅇ

근데 이게 좆도 없고 걍 다 똑같다 이게 문제인거지 지금 맞지
일단 로스를 찍어보자 일단 내일까지 봐야되나 그러면 내일 뭐할지라도 오늘 다 정해놔야겟네

0.9로바꿀까 약간 극단적으로 3번에 해당하는거 그걸로 하나 가고
1번에 해당되는건 lambda-u이거
0->300 으로되는건데 이거 0->20에폭으로 고쳐야함(만약배치20이면) 20에폭이후에는 lu = 150
이거 우리가 고친거라고? 원래 step에 따라 올랐어?
논문에그렇게옴 그건 모르겠 이상하게 되어있음 우리 코드에 근데 스텝으로 해야됨
rampup 이 16000스텝이래 논문말로는

2번을진짜 오지는 걸 하면 될수도 있고봐

1. train accuracy percentage에 해당되는 confidence 값들의 최소값을 threshold로 잡는다.
2. percentage 비율 조정
3. 다 똑같겠지. 뭔가 비교가 되면 그거에 이어서 더 할 수 있을텐데 지금 방향성을 못잡겠어

[내일 실험]
바꿔야하는것: 모델b3로, 그래프찍고싶은거 최대한많이찍기, good ulb튀는거없애기, 코드정리하기

(1) lambda-u scheduling 고쳐서 다시해보기 -> loss 두개 다 plot(lu곱한후값)
(2) fixed threshold 0.9 -> 이것도 loss 둘다 plot
(3) 이거진짜 기똥찬거 하나 없을까 1,2보다 훨 잘되는거 ㅜㅜ

+ 6000개로 두개 더 (1번, 3번)
+ ------5개------
+ fixed 0.9 / RA(2,5)
+ fixed 0.9 / RA(2,3)

나머지 두개는 augment? RA(2, min(9,epoch//10))

이거할거면 anchoring ? week augment로 Pseudo label 구하고 strong augment로 계산

우리 2,3에서도 T=0.5 하는거냐? 안하는거냐? 하는거냐? 안하는거냐? ㅇㅋ 3번에서 하지말아볼까 0.뭐가9에서 one hot으로 바꿔볼까 어차피 그게 그건가 0.9넘으면 그게 그거겠다
아 맞가

0.9, 0.1 → 0.81, 0.01 = 0.9878, 0.0122이거임?
0.8, 0.2 → 0.64, 0.04 = 0.9412, 0.0588

진짜엄청난 방법이 없을까 생각 퍼센트 진짜 괜찮아보였는데 이렇게되고보니까 걍 별로인듯 더 다듬어야지 지금 퍼센트는 이게 train accuracy가 80퍼라도 80개 안에 드는게 다 맞는게 아니자나
맞는다는거의 기준은뭘까
0.4 0.4 0. … 이렇게나오는 boundary case를 먹여주는건 학습에 도움이 될까 안될나

label 있는 데이타의 preciction이 label과 같을 때 confidence값들 > 0.5
confidence가정확히뭐야? 걍 one hot label이면 0.1 0.1 0.8 이런식으로 된다하면 이런거 그 값들 아
잠만 ㄱㄷ

일단 저거 하면서 글고 로스 스텝마다 찍는것도 하고, percentage 안에 드는 confidence값들의 최소값, 평균값도 찍어볼래

만약 저거 다 하고나서도 큰변화나 연관성 못찾으면 어쩔수없이 라벨 줄여가면서 해봐야되나
지금 라벨 16000개 ulb 4만개
만약 4천개 4만개면?
2천개 4만개면? 이거 적어도 distribution은 맞춰줄수있나?

max confidence, 바꿔야하는것: 모델b3로, 그래프찍고싶은거 최대한많이찍기, good ulb튀는거없애기, 코드정리하기, Temporature 제대로하기 **전부다 T=0.5**

b3 (20/50),
[1] MixMatch, Lu Scheduling 0->150 in 0->20 epochs. (405)
[2] 0.9 Fixed Threshold Scheduling, T = 0.5 (407) ====> 껐음 (430)
[3] 0.8 Fixed Threshold Scheduling, T=0.5 (408)
[4] Mixed Loss Function [1]+[2] (409)
[5] Adaptive Thresholding using train accuracy. Min 0.5 (401)
[6] Adaptive Thresholding using validation accuracy. Min 0.5 (406) => cross(423) => (431)
[7] Adaptive Thresholding using train accuracy. ensure ratio 2:5(417) => (428)
  train_accuracy < 20:

fixed threshold 0.9

else:

train accuracy percent + threshold 0.5

1. unlabeled loss가 labeled loss에 비해 너무 작음
   → scale을 키운다
   → cross entropy로 바꾼다

2. confidence값 100 step마다 평균내서 찍기
3. unlabeled loss값만 비교해보았을 때 adaptive thresholding이 가장 좋음
4. val accuracy는 초반엔 fixed th이 젤 좋은듯

We use this L_2 loss in eq. (4) (the multiclass Brier score [5]) because, unlike the cross-entropy, it is bounded and less sensitive to incorrect predictions. For this reason, it is often used as the unlabeled data loss in SSL [25, 44] as well as a measure of predictive uncertainty

믹스매치에선 이렇게 말하는데

내생각엔 1번은 냅두고 2나 3중에 하나 cross entropy로 바꾸고 567은 다 cross entropy로? 왜냐면 mse쓰는게 틀려도 좀 less sensitive하게하는건데 우리 threshold policy적용하면 그럴필요가 없는거 아닌가 그걸 보여주면 더 좋은거고?

2번: fixed scheduling 0.8-> with cross entropy loss (430)
6번: adaptive with train accuracy, min 0.5 with CE (423)

(406) → (423) cross entropy → (431) 마지막 val값들 넘겨주기

[threshold policy 비교]
efficient-b4 environment: (lr: 5e-3 / [2,4,8], 0.5), (bs: [10,25]), (T: None)
(377) efficient-b4, (th: None, rampup)
(378) efficient-b4, (th: fixed 0.8)
(348) efficient-b4, (th: manual 0.9 → 0.5/ 느리게)
(387) efficient-b4, (th: manual 0.9 → 0.5/ 빠르게)
(374) efficient-b4, (%: train_acc)
(386) efficient-b4, (%: train_acc + th: 0.5)

efficient-b3 environment : (lr: 5e-4), (bs: [20,50]), (T: 0.5)
(405) efficient-b3, (th: None, rampup: lambda-u 0->150 in 20 epochs)
(407) efficient-b3, (th: fixed 0.9)
(408) efficient-b3, (th: fixed 0.8)
(430) efficient-b3, (th: fixed 0.8) + CE
(373) efficient-b3, (%: train_acc)
(401) efficient-b3, (%: train_acc, th: 0.5)
(431) efficient-b3, (% train_acc, th: 0.5) + CE + last_acc_val update
(406) efficient-b3, (%: val_acc, th: 0.5)

(417) efficient-b3, (train_acc < 20 → th: fixed 0.8,
train_acc >= 20 → %: train_acc, th: 0.5, ensure bs ratio)
(428) efficient-b3, (417) + last_acc_val update

(409) efficient-b3, (mixed loss => th: 0.9 + rampup)

[메모]
(417), (428) loss_un 그래프 비교 → 이상함. 왜 안줄어드는지 모르겠음
(408), (430) cross entropy가 역전중

CE를 쓰니 loss_un의 스케일이 커지고, 더 많이 줄어드는 모습을 보이는데 loss_x도 더 많이 떨어지는 것을 보니 다행히 학습에 더 도움이 되는 듯 하다. 문제는 초반에 train accuracy가 높지도 않은데 loss_un의 값이 커서 학습에 어려움이 있고 따라서 accuracy가 처음에 잘 오르지 않는다는 것이다. 따라서 (409)번이나 (417)번과 같이 두 loss를 epoch에 따라 나눠주거나 섞어 주는 것이 필요하다고 생각한다.

예를 들어 (401)번과 (431)번을 비교해 보겠다. 처음에는 MSE를 사용한 (401)번의 good unlabeled 값이 더 높았는데, 15 epoch 이후에는 (431)번이 더 높다. (401)번의 train accuracy가 더 높은데도 불구하고 good unlabeled 값이 높다는 것은 confidence값이 전체적으로 높아졌다는 것. 이후에 train accuracy가 더 높아지며 (지금은 거의 비슷한 상태) good unlabeled 값이 훨씬 높아졌다. unlabeled 데이터의 confidence값은 확실히 높아졌지만 답을 못맞춘 채 confidence만 큰 빙구가 되었을 가능성도 있다. **(fixed thresholding의 경우라면?)** 그러나 validation accuracy가 더 빠른 속도로 올라가는 중이고 이제 거의 비슷해졌으므로 CE가 unlabeled 데이타를 더 많이, 더 정확하게 이용할 수 있는 좋은 방안이 될 수 있을 것이다. 그럼 MSE와 CE를 어떻게 하면 잘 섞을 수 있을까?

train accuracy 그래프를 비교해보면 MSE는 처음부터 잘 올라서 3 epoch에 13.52%가 되었는데 CE는 정신 못차리다가 4 epoch부터 오른다. 즉, 오르기 시작하는 포인트에서 이미 10퍼센트 이상 차이가 난다. 근데 50 epoch 쯤에 비슷해지는 것을 보면 CE의 학습 속도가 더 빠르다고도 볼 수 있다. 그럼 1~3 epoch까지는 MSE를 쓰고 그 다음부터는 CE를 쓰면 어떨까. 아니지. unlabelded loss의 이전 epoch값 (or 100 step avg)을 저장해서 그거보다 커지면 loss_un 적용하지 말기.
아닌가 이거 오르는건 똑같나. 3 epoch만큼 더 걸리는 것 뿐인가. 더 오래 loss가 떨어지고 더 오래 acc가 오르는건가 걍. 걍 이따 봐야하나

결론 : MSE랑 CE를 잘 섞어야함. 문제는 몇 epoch부터 섞을것인가

지금 둘다 똑같은 스케일의 CE

이것도 안나왔을때 해볼수있는것들

1. MSE lambda_u 스케일 키워가지고 loss_x랑 비슷하게 맞춰주기(fixed, adaptive에다가)
2. 라벨 적을때 우리꺼가 더 잘된다 이거 해보기
3. gg치고 차이 없는거 인정한다음에 augment하기

401 405 408 431 놔두기

*어차피 믹스매치는 절대 안바꿀거잖아 lambda u도 안고칠거잖아.

추후에 할때 CE로할건지 MSE로할건지
MSE로할거면 lambdau 1500으로 바꿀건지

국룰 : 4000 : 40000

1. mixmatch -150
2. adpative + 0.5 1500

이거두개 해보고
fixed threshold 에서 augment(2,x) 해서 더 잘되는값 찾자
(401) efficient-b3, (%: train_acc, th: 0.5) + lu 150 MSE
(436) 2,3 150
(437) 2,5 150

(438) 2,5 lu 1500

436 437 438 끈다고 하면..

labeled only augmix
fixed 0.8 + augmix

[지금 시도중인 것]
only supervised
(458) 5e-4
(460) 5e-5
(462) 5e-6
(464) 1e-6
(463) 5e-7 → 이것도 빨라보임 ;;
(465) 5e-8

[다음에 해볼 것]

(431)이 (401)보다 높자나. 그럼 그게 단순히 스케일이 커져서 그런걸까 아니면 cross entropy만의 뭔가 있는걸까.

그니까 지금 431이 딱 뭐야. last acc val 이게 정확히 뭐지? 엉 근데 그걸 바꿨는데도 처음에 val acc가 요동치긴 하는거야ㅇㅋㅇㅋ뭔지알겟음ㅓ ㅇㅇ

얘는 train accuracy 비율에 맞춰서 unlabeled 데이타를 쓰자나. 근데 초반에 train accuracy값이 매 epoch마다 average meter가 초기화되면서 batch idx가 작으면 얘 값이 이상해지지. 우리 그 그래프 이상해서 바꾼거. 걍 마지막 train_acc.val을 다음 epoch
efficient-b3, (% train_acc, th: 0.5) + CE + last_acc_val update
adaptive + th0.5

466은 그러면 mse로스케일만키워서 한거고..
467 468은 ce영향력을 좀더 키우면서 한거고

근데 지금 우리가 찾아낸거는?
fixed 0.9f랑 mixmatch를 그냥 mse로돌리면 비슷비슷하고
adaptive를 ce로돌리니까 훨씬 좋은 결과가 나왔다 여기까지잖아. 일단.든

mse로 했을때도 adaptive가 fixed 보다 잘됨 ㅇㅋㅇㅋ

experiment 1로 이렇게 그래프 세개 설명하면서 해도 되고. 근데 여기서 추가로 더 실험해서 발표 딱 잘 하려면? 한가지 정도 실험을 더 하거나(라벨 더 적을때?) /// confidence값등 다른 그래프를 내세우면서 우리 주장 뒷받침하는 느낌으로다가

우리는 mixmatch보다 unlabeled 데이타를 더 잘 활용한다는 걸 강점으로 내세워야해. >< ㅇㅇ confidence값도 우리께 훨씬 높자나. CE를 썼을 때

일단 그러면 봐바 저거 세개끄면 세자리 남잖아 네자리 남는구나 469도끄면
한자리는 근데 mixmatch+CE하고싶자나 그치

저거 왜꺼 잘될거같은데? 아 405랑 401이랑 431?
다 끝나면? 431엥간하면 안끝나려나 많이남앗나 아직

그러면 나 저녁먹고올테니까 저녁 먹고나서
0.발표때 어떤 그래프 뭐 넣고 뭐 더해야되고 여기서 부족한게
뭐있는지(발표가지금최우선이니까)
->일단이걸로 당장돌릴거 결정하고 요게 끝일거같긴한데.. 3번 더돌리는거 ㅋㅋ

보고서까지 다 생각했을때
1.앞으로 어떤 실험을 더 해야하는지

[1] (466) %: train_acc + th: 0.5 // MSE(lambda_u: 1200) 와 (431) 비교
그 담엔 CE로 여러 threshold policy를 비교하는게 필요함. 이틀 걸리니까 적어도 오늘 밤엔 해야하네 조졌네 ㅅㅂ
[2] CE %: train_acc + th: 0.4
[3] (467) CE lambda_u = 1.5
[4] (468) CE lambda_u = 2
[5] (**469**)CE 1까지 천천히 올리기 20epoch동안 → 요게 소용 없을것같거든

자. 지금 우리가 발표할 때 써먹을 수 있는게
401 <-> 405 <-> 408 <-> 431 일단 이거지. 아 그러면 ppt 를 만들면서 생각해볼까

낼 우리 만나지? 낼 너가 오냐? 나 지금 ppt살짝만들면서생각정리하려고

세세한 스토리정리나 알고리즘은 ppt만들면서 하고..이건 내가 이번에 거의 만들겡
[스토리 정리]

mixmatch는 threshold policy를 쓰지 않기 때문에 불확실한 guessed label을 많이 포함하고 있다. 때문에 lambda_u가 크면 잘못된 방향으로 학습될 가능성이 커 unlabeled loss term이 labeled lsos term에 비해 많이 작아야 하고, 이는 unlabeled 데이타를 많이 이용하지 못한다는 뜻이다.

우리는 threshold policy를 사용해 확실한 guessed label만을 사용했고, 따라서 unlabeled loss term이 커졌을 때 오히려 성능이 좋아짐을 확인했다. train accuracy는 조금 작아짐에 반해 validation accuracy는 커지는 것을 보아 unlabeled 데이타를 더 잘 활용하여 오버피팅이 줄어든 것으로 추측된다.

fixed보다 adaptive가 잘 된 이유?

[발표 개요]

1. brief purpose : find better thresholding policy

2. related models(algorithms)
3. experiments
    a. base environments : efficient b3, lr : 5e-4, …
    b. comparing threshold policy : different unlabeled loss graph, but same accuracy result → conclusion : unlabeled is not used well in mixmatch model
    c. MSE (small lambda u), MSE (small lambda u), CE comparison
    d. comparing threshold policy (with using CE loss term)
    e. using few labeled data
    f. extra : augmentation (randaugment, augmix)
4. conclusion
    a. Mixmatch use all unlabeled data without threshold policy. → include uncertain guessed labels → lambda_u should be small → unlabeled loss

지금 해야되는거.

183 72.5 85.56
405 73.36 84.417
401 73.96 84.96
408
431 74.69 84.61 / 74.85 85.11/ 74.71 85.46
위에다가 라벨 4000개만 해서 돌리기.

[1] (431) Adaptive Threshold + Res50 (lr: 5e-4), (bs: [30,75]) → (475)
[2] (405) Mixmatch_basic + label 4000 → (478)
[3] (408) Fixed Threshold + label 4000
[4] (431) Adaptive Threshold + label 4000 → (477) -> (485)

(372)도 Res50 씀

(488) : 405 이어서.

resnet50 왜안되는지 한번더.

할수있
다!!

발표 끝나고 해야 하는것.

보고서 작성하기 (CV Paper)
-> 구글독스에 작성, 포맷 맞게 옮기기, 표 및 그래프 잘 추가, 레퍼런스 추가 (형식 맞추는거 좀 오래 걸릴듯)

일단 논문에서는 밑에 이 귀찮은거 하는게 조금 시간 걸릴거같고 형식도 맞춰서 써야함.

좀 귀찮은거 -> 새로운 loss 식 그려야하고 슈도코드 써놔야함

그래프 두개(3.2에서)
표 세개(3.2, 3.3, 3.4) 이렇게 필요할거같음. 이것도 라텍스/엑셀 로 하면될듯

깃허브 정리 순서

1) 폴더 구조 깔끔하게 만들기 -> 딱 세개로만 mixmatch, fixed, adaptive이거 남기고 나머진 etc안에서 정리해놓기
2) 저 세개 코드 주석 다 달고 필요없는부분 지우기
3) 저 세개 코드에서 fixed -> threshold / adaptive -> min threshold랑 lambda u 값 파라미터로 가져올수있는지 확인하고 가져올수 있다면 나중에 readme 에 써놓기
4) Readme 작성 -> 총 4개(root, 저 세개 폴더), 내용은 안에 들어있는 파일들에 대한 설명하고, 실행하는 것 있으면 어케 실행하는지(3번에서 한것 포함해서) 작성하고 root는 좀더 많은 설명 필요할거같음.

깃허브 정리 아직안한것
root, 각 디렉토리 자세한 설명 (README)
experiment code 추가

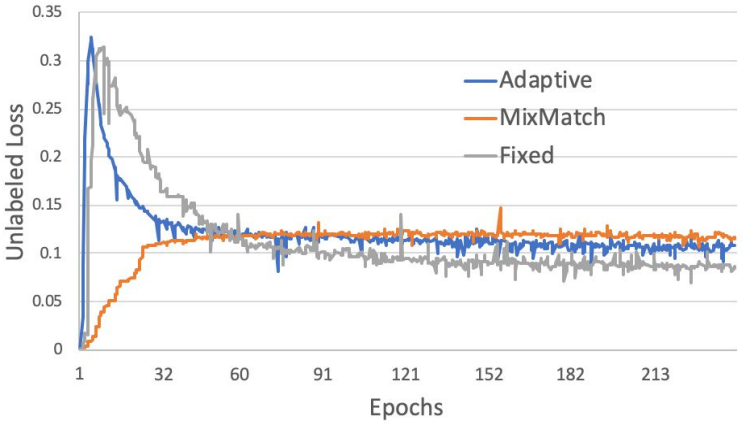
세션하고 모델 정리
우리 테스트에(논문,발표에) 사용된 모델 다운(?) 받아서 추가하기. 몇번 테스트 몇번 모델인지 따로 표시해놔야 할듯.

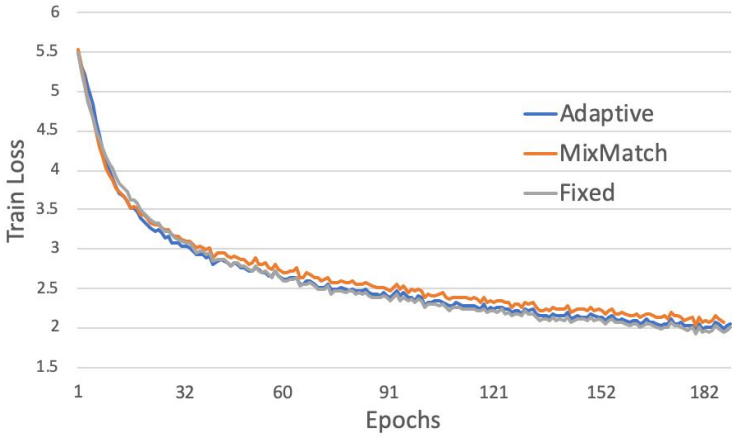
⇒ 2. nsml 내부에서 학습을 했다면, session과 model 이름, 그리고 pretraining에 사용한 code 및 실행 script를 첨부해주세요.

=> 깃허브에 readme에 추가하거나 pdf파일 하나 만들어서
논문 각 experiment 발표 각 experiment별로 몇번 세션인지 깔끔하게 정리해놓으면 될거같음.

-------여기까지중에 안한거 -------
논문에 자료들 넣기

제출!! (깃허브에도 보고서하고 이거 노트 올리기)





[word]
Pseudocode batchsize, 맨 윗에굵은선
링크
페이지 넘버 → 잘 안되서 걍 일일이 넣어줌
테이블 설명 텍스트 width 일정

[Github]
pretrained model upload