Experiment - Bag of Tricks

I. 실험 개요

"Bag of Tricks for Image Classification with CNN" 논문을 읽고 논문에서 소개된 다양한 기법들을 직접 적용해서 성능 변화를 확인 후 결과를 정리한다.

1. 실험은 https://github.com/bentrevett/pytorch-image-classification의 ResNet50(Base Model), ResNet152(Teacher Model) 모델을 기반으로 진행한다.
2. Cosine Learning Rate Decay, Label Smoothing, Knowledge Distillation, Mixup 네 가지 기법과 다양한 조합에 대한 성능 비교를 실시한다.

II. 실험 환경

Platform Google Colab, GPU NVIDIA A100-SXM4-40GB, CUDA 11.3, Pytorch 1.12.1, Torchvision 0.13

III. 실험 방법

1. **데이터 셋 준비**

   실험에 필요한 데이터 셋은 CUB200(2011년)을 사용한다. 총 11788장, 200개의 조류 클래스로 이루어진 데이터셋으로부터 8:2의 비율로 Train Set과 Test Set을 나누고, Train Set의 10%를 Valid Set으로 사용해 7.2:0.8:2의 비율로 실험을 진행한다. 추가로 이미지 데이터를 224x224의 해상도로 재조정하고 Pretrained Model에 맞도록 정규화하며 학습 데이터의 경우 랜덤하게 -5~5도의 회전, 수평 반전, Crop 기법으로 변형하여 사용한다.

2. **모델 선정**

Base 모델으로 23,917,832개의 파라미터를 학습할 수 있는 ResNet50을 사용하고, Base 모델에 Knowledge Distillation 기법을 적용하기 위해 사용될 Teacher Model으로는 58,553,608개의 파라미터를 학습할 수 있는 ResNet152로 선정했다. 두 모델 모두 기본적으로 BATCH_SIZE 64, EPOCH 10, Adam Optimizer와 OneCycle Learning Rate Scheduler를 채택하여 학습한다.

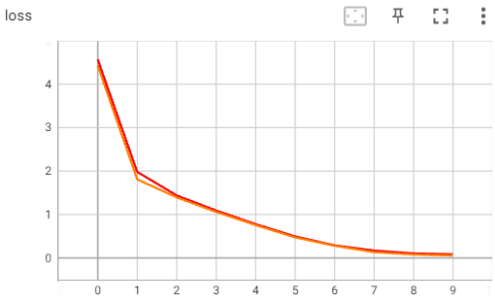
모델	#Params	Top-1	Top-5	Time/Epoch
ResNet50	23.9M	78.53	93.82	1.3 분
ResNet152	58.5M	**80.49**	**96.61**	1.58 분

Table 1. Base 모델과 Teacher Model의 비교

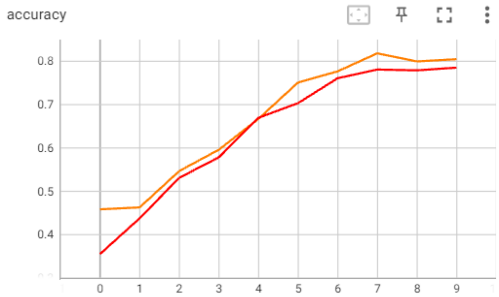
(a) OneCycle Learning Rate Schedule

그림 (b)~(d)는 두 모델의 비교 그래프를 나타낸 것이다. 모델 용량이 더 크고 깊은 ResNet152가 ResNet에 비해 모든 구간에서 Accuracy가 더 높은 것으로 확인되었다. 이 것을 이후 소개될 Knowledge Distillation 기법에서의 Teacher Model으로 사용할 것이다.

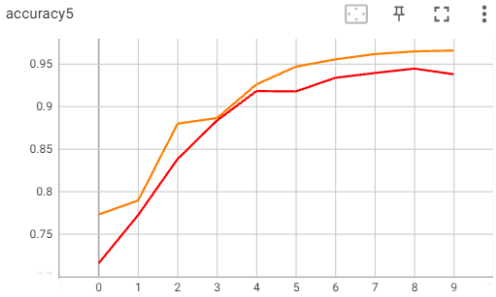
Teacher Model로써 사용할 것이며 Base Model의 그래프 (붉은색)를 각 기법에서 비교군으로 사용할 것이다.



(b) Train Loss



(c) Top-1 Validation Accuracy

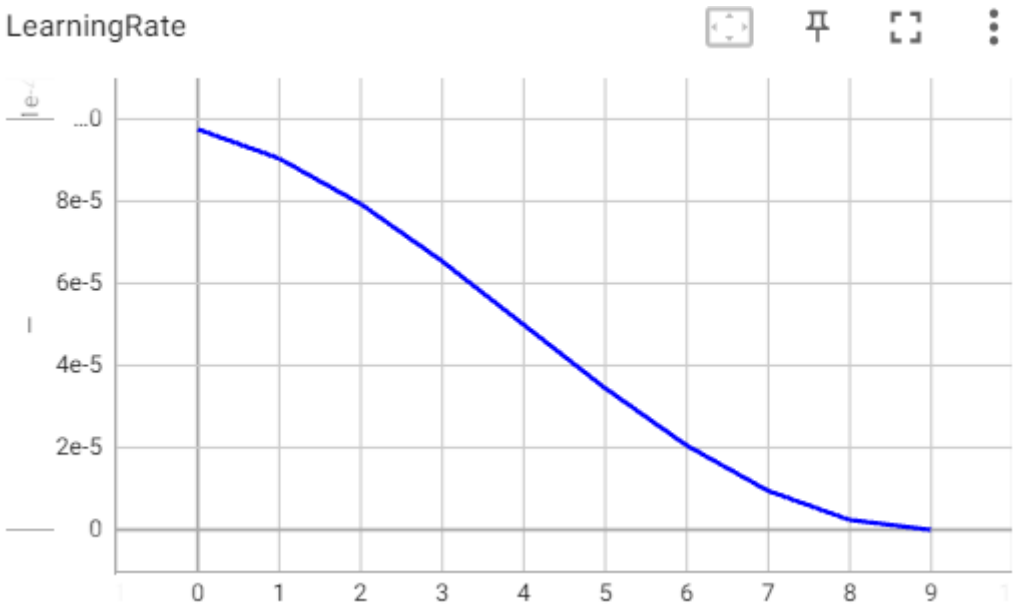


(d) Top-5 Validation Accuracy

3. 기법 적용

본 실험에 사용될 기법은 Cosine Learning Rate Decay, Label Smoothing, Knowledge Distillation, Mixup 네 가지이며 아래 a~e와 같은 조합을 적용하여 실험을 진행한다.

a. Cosine Learning Rate Decay



(a) Cosine Annealing Learning Rate Schedule

Learning Rate가 0.0001에서 0까지 Cosine 곡선에 따라 감소할 수 있도록 아래와 같은 Optimizer와 Scheduler를 정의하여 사용한다.

```
import torch.optim.lr_scheduler as lr_scheduler
import torch.optim as optim
FOUND_LR = 1e-3
params = [
        {'params': model.conv1.parameters(), 'lr': FOUND_LR / 10},
        {'params': model.bn1.parameters(), 'lr': FOUND_LR / 10},
        {'params': model.layer1.parameters(), 'lr': FOUND_LR / 8},
        {'params': model.layer2.parameters(), 'lr': FOUND_LR / 6},
        {'params': model.layer3.parameters(), 'lr': FOUND_LR / 4},
        {'params': model.layer4.parameters(), 'lr': FOUND_LR / 2},
        {'params': model.fc.parameters()},
        ]
optimizer = optim.Adam(params, lr = FOUND_LR)
if(CosineDecay):
    scheduler = lr_scheduler.CosineAnnealingLR(optimizer, T_max=TOTAL_STEPS,
 eta_min=0,last_epoch=-1)
```

b. Label Smoothing

One-Hot 라벨을 0과 1사이의 Soft Label로 스무딩하고 예측치와 함께 CrossEntropy를 계산하기 위해 아래와 같은 Loss Function을 정의한다. Smoothing 파라미터는 0.1으로 한다.

```
import torch.nn as nn
class LabelSmoothingCrossEntropy(nn.Module):
    def __init__(self):
        super(LabelSmoothingCrossEntropy, self).__init__()

    def forward(self, x, target, smoothing=0.1):
        confidence = 1.-smoothing
        logprobs = F.log_softmax(x, dim=-1)
        nll_loss = -logprobs.gather(dim=-1, index=target.unsqueeze(1))
        nll_loss = nll_loss.squeeze(1)
        smooth_loss = -logprobs.mean(dim=-1)
        loss = confidence * nll_loss + smoothing * smooth_loss
        return loss.mean()

if(LabelSmoothing):
    criterion = LabelSmoothingCrossEntropy()
```

c. Knowledge Distillation

   Teacher Model의 예측치로부터의 KLDivergence Loss와 Base Model의 Loss 둘을 더한 Total Loss를 얻기 위해 아래와 같은 Loss Function을 정의한다. 파라미터 α와 T는 각각 0.1, 10으로 한다. 실험에서 Knowledge Distillation과 Label Smoothing 기법을 같이 적용할 경우 Student Model에 단순히 Label Smoothing Loss function을 적용할 수 없으므로 Label Smoothing 기법으로 학습한 Teacher Model을 사용하도록 한다.

```
class knowledge_distillation_loss(nn.Module):
    def __init__(self):
        super(knowledge_distillation_loss, self).__init__()
        self.alpha = 0.1
        self.T = 10
    def forward(self, pred, labels, teacher_pred):
            student_loss = F.cross_entropy(input=pred, target=labels)
            distillation_loss = nn.KLDivLoss(reduction='batchmean')(F.log_sof
tmax(pred/self.T, dim=1), F.softmax(teacher_pred/self.T, dim=1)) * (self.T *
 self.T)
            total_loss =  self.alpha*student_loss + (1-self.alpha)*distillati
on_loss

            return total_loss

if(KnowledgeDistillation):
  criterion = knowledge_distillation_loss()
  teacher_model=ResNet(resnet152_config, OUTPUT_DIM)
  teacher_model.load_state_dict(torch.load('tut5-teacher-model.pt'))
```

d. Mixup (α=1, 0.2, 0.1, 0.01)

Train set 두 개를 랜덤하게 불러온 뒤, 베타 분포에 의해 얻은 0과 1사이 값에 따라 입력 x와 라벨 y를 새롭게 정의한다. 참고 문헌 [3]에서 α의 변화(베타 분포에서 α=β로 사용)에 따라 다른 학습 결과를 얻는다고 하므로 본 실험에서는 α가 1, 0.2, 0.1, 0.01인 네 경우에 대해서 결과를 비교하고자 한다.

```
import torch.utils.data as data
import numpy as np

train_iterator = data.DataLoader(train_data,
                                 shuffle = True,
                                 batch_size = BATCH_SIZE)
if(Mixup):
    mixup_iterator = data.DataLoader(train_data,
                                     shuffle = True,
                                     batch_size = BATCH_SIZE)

def train(model, iterator, optimizer, criterion, scheduler, device):
  if(Mixup):
          for (ox, oy), (mx, my) in zip(iterator, mixup_iterator):
              lam = np.random.beta(alpha, alpha)
              x = lam*ox+(1.-lam)*mx
              y = lam*oy+(1.-lam)*my
              y = y.to(torch.int64)
```

다음의 실험 방법 e~i는 a~d 기법의 조합으로 진행한다. 단, Label Smoothing과 Knowledge Distillation을 함께 사용하는 경우 c에서 언급한 방법을 적용한다.

e. Cosine Decay + Label Smoothing
f. Cosine Decay + Knowledge Distillation
g. Label Smoothing + Knowledge Distillation
h. Cosine Decay + Label Smoothing + Knowledge Distillation
i. Cosine Decay + Label Smoothing + Knowledge Distillation + Mixup (α=0.2, 0.01)

4. **결과 출력 및 성능 비교**

성능 비교를 위한 시각화 기법으로 TensorBoard를 연동하여 매 Epoch 마다 결과 데이터를 저장한다. Train과 Evaluation 각각의 과정에서 Train Loss, Top-1 Validation Accuracy, Top-5 Validation Accuracy, Learning Rate 데이터를 시각화한다. 이 때, TensorBoard의 UI에서 그래프 Smoothing 기능과 Outlier 제거 기능은 해제한다.

```
 import tensorflow as tf
import datetime
```

```
current_time = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
train_log_dir = 'logs/gradient_tape/' + current_time + '/train'
test_log_dir = 'logs/gradient_tape/' + current_time + '/test'
lr_log_dir = 'logs/gradient_tape/' + current_time + '/lr'
train_summary_writer = tf.summary.create_file_writer(train_log_dir)
test_summary_writer = tf.summary.create_file_writer(test_log_dir)
lr_summary_writer = tf.summary.create_file_writer(lr_log_dir)

for epoch in range(EPOCHS):
  #Train, Evaluate 이후
  with train_summary_writer.as_default():
        tf.summary.scalar('loss', train_loss, step=epoch)
        tf.summary.scalar('accuracy', train_acc_1, step=epoch)
        tf.summary.scalar('accuracy5', train_acc_5, step=epoch)
      with test_summary_writer.as_default():
        tf.summary.scalar('loss', valid_loss, step=epoch)
        tf.summary.scalar('accuracy', valid_acc_1, step=epoch)
        tf.summary.scalar('accuracy5',valid_acc_5, step=epoch)
      with lr_summary_writer.as_default():
        tf.summary.scalar('LearningRate', scheduler.get_last_lr()[0], step=epoch)
```

IV. 실험 결과

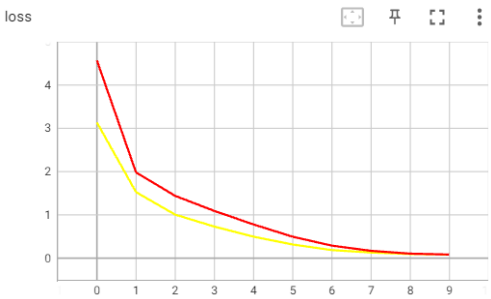
1. Cosine Learning Rate Decay

Base 모델에 Cosine Decay 기법 적용 시 결과는 Table 2와 같으며 (a)~(d)에서 노란색으로 표시했다.

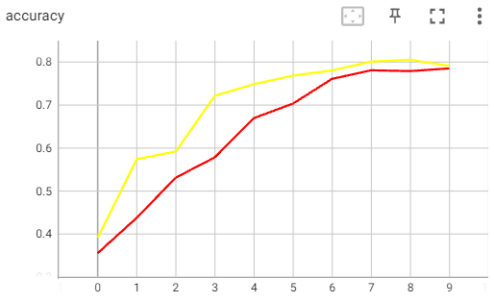
모델	기법	Top-1	Top-5	Time/Epoch
ResNet50	Base	78.53	93.82	1.3 분
	+Cosine Decay	**79.13(+0.6p)**	**95.77(+1.95p)**	1.3 분

Table 2. Cosine Decay

(a) Learning Rate Schedule

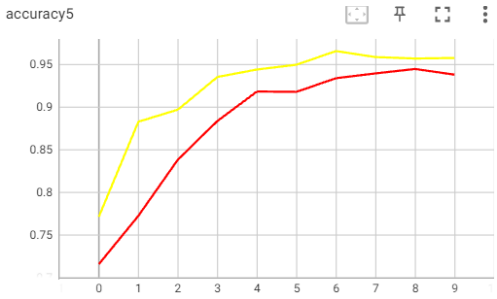


(b) Train Loss



(c) Top-1 Validation Accuracy

매 Epoch에서의 Learning Rate는 (a)와 같으며 Base Model의 결과에 비해 Loss, Accuracy가 더 나은 지표를 보여준다. 특히 초기 Learning Rate가 OneCycle 기법 보다 높은 데에도 불구하고 (b)와 같이 더 낮은 초기 Train Loss를 가지는 것이 관찰되었다.



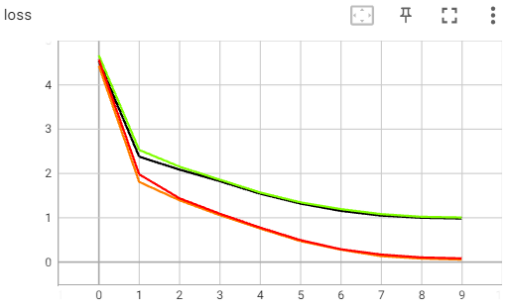
(d) Top-5 Validation Accuracy

2. **Label Smoothing**

Base Model과 Teacher Model에 Label Smoothing 기법 적용 시 결과는 Table 3과 같으며 (a)~(c)에서 Label Smoothing 기법으로 학습 한 Teacher Model은 검정색, Base Model의 경우 연녹색으로 표시했다.

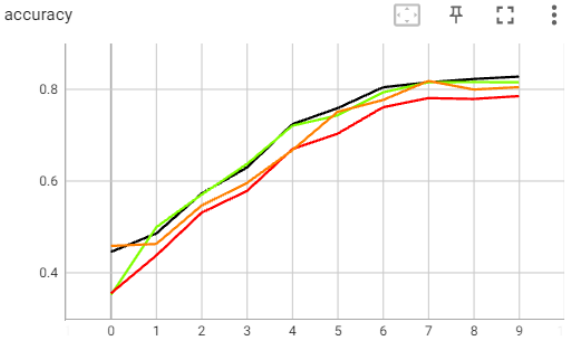
모델	기법	Top-1	Top-5	Time/Epoch
ResNet152	Teacher	80.49	96.61	1.4 분
	+Label Smoothing	82.78(2.28p)	96.34(-0.27p)	1.4 분
ResNet50	Base	78.53	93.82	1.3 분
	+Label Smoothing	**81.55(+3.02p)**	**96.17(+2.35p)**	1.3 분

Table 3. Label Smoothing

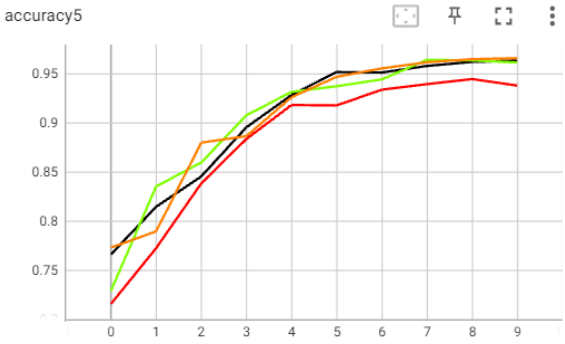


(a) Train Loss

Label Smoothing 기법이 적용 된 경우 모든 구간에서 Top-1과 Top-5 모두 Base 모델 보다 더 나은 지표를 보여주는 반면, Train Loss가 Base 모델보다 느리게 감소하는 모습을 볼 수 있다.



(b) Top-1 Validation Accuracy



(c) Top-5 Validation Accuracy

특이한 점은, Teacher Model의 경우 오히려 Top-5에서의 지표가 미약하게 감소한 결과가 나타났다.

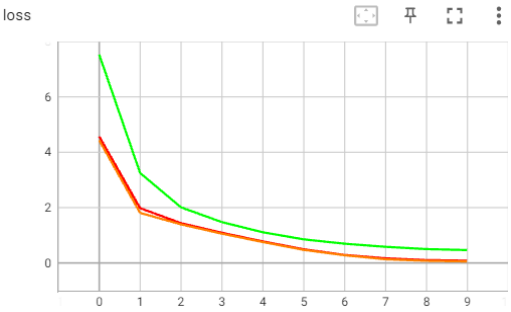
3. **Knowledge Distillation**

단, Accuracy의 경우 Top-1, Top-2 모두

Base 모델에 Label Smoothing 기법 적용 시 결과는 Table 3과 같으며 (a)~(c)에서 녹색으로 표시했다.

모델	기법	Top-1	Top-5	Time/Epoch
ResNet50	Base	78.53	93.82	**1.3 분**
	+Knowledge Distillation	**81.36(+2.83p)**	**96.90(+3.08p)**	1.4 분

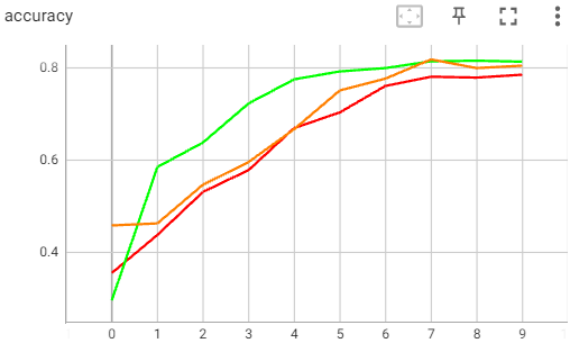
Table 4. Knowledge Distillation



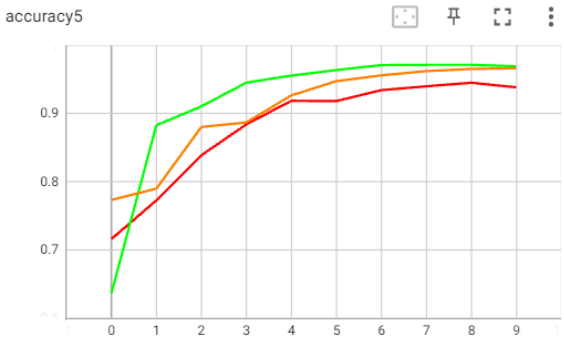
(a) Train Loss

모든 구간에서 Top-1과 Top-5 모두 Base 모델 보다 더 나은 지표를 보여주며, 모든 구간에서 Train Loss가 Base 모델보다 높은 bias를 가지며, 특히 실험 1과 비슷하게 초기 Epoch에서 더 높아진 bias를 가지는 것을 확인할 수 있다.

Teacher Model(주황)보다 모든 구간에서 더 좋은 지표를 나타냈다. 단, Base 모델을 학습할 때 보다 Epoch 당 학습에 소요되는 시간이 약 0.1 분 증가했다.



(b) Top-1 Validation Accuracy



(c) Top-5 Validation Accuracy

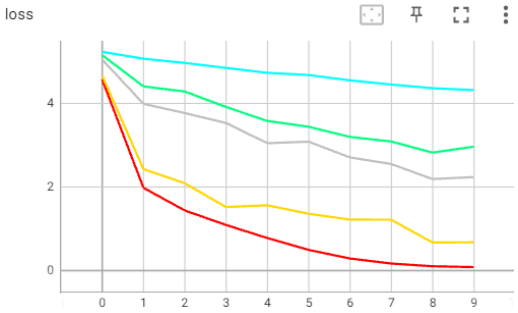
4. **Mixup (α=1, 0.2, 0.1, 0.01)**

   Base 모델에 Mixup 기법 적용 시 결과는 Table 5와 같으며 α가 1, 0.2, 0.1, 0.01에서의 결과를 (a)~(c)에서 각각 하늘색, 연두색, 은색, 금색으로 표시했다.

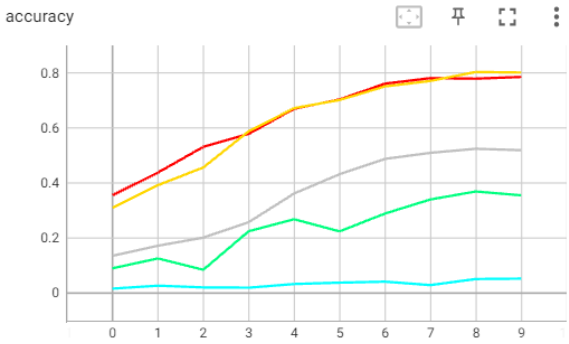
모델	기법	Top-1	Top-5	Time/Epoch
ResNet50	Base	78.53	93.82	**1.3 분**
	+Mixup α=1	5.27(-73.26p)	24.24(-69.58p)	2.5 분
	+Mixup α=0.2	35.53(-43.00p)	78.05(-15.77p)	2.5 분
	+Mixup α=0.1	51.92(-26.61p)	91.21(-2.61p)	2.5 분
	+Mixup α=0.01	**80.13(+1.6p)**	**96.38(+2.56p)**	2.5 분

Table 5. Mixup

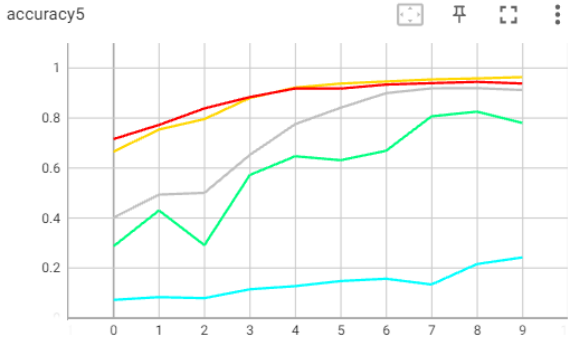
모델만 학습할 때 보다 약 1.2분 더 소요되었다. 단, 참고 문헌 [3]에서 RenNet 모델 최적의 α:α∈[0.1, 0.4]인 것과 본 실험에서 가장 우수한 성능을 보여준 α:α=0.01인 것을 고려해 실험 9에서 α=0.2과 0.01 두 가지 경우의 α과 함께 Mixup을 다시 한번 적용해보고자 한다.



(a) Train Loss



(b) Top-1 Validation Accuracy



(c) Top-5 Validation Accuracy

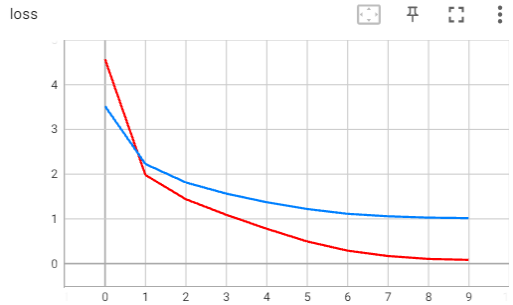
모든 α 경우에 대해 Train Loss가 감소는 하지만 Base 모델보다 Bias가 큰 것을 확인할 수 있다. 특히, α가 0.01일 때의 제외한 Mixup은 총 Epoch가 10인 본 실험에서 Base 모델 보다 더 낮은 지표를 보여주고 있다. 또, Epoch 당 학습에 소요되는 시간이 Base 모델보다 더 많은 시간이 소요되었다.

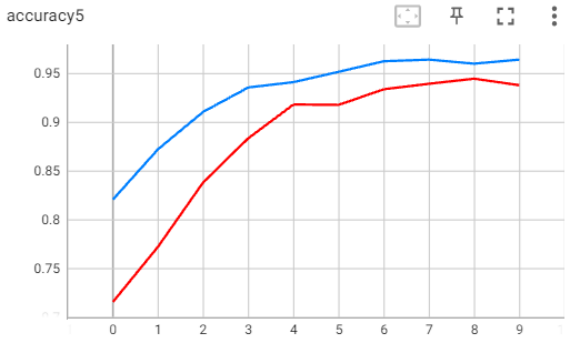
5. **Cosine Decay + Label Smoothing**

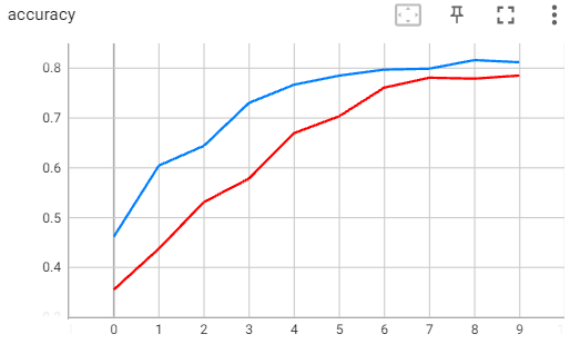
Base 모델에 Cosine Decay와 Label Smoothing 기법을 함께 적용 시 결과는 Table 6과 같으며 (a)~(c)에서 밝은 파란색으로 표시했다.

모델	기법	Top-1	Top-5	Time/Epoch
ResNet50	Base	78.53	93.82	**1.3 분**
	+Cosine Decay	79.13(+0.6p)	95.77(+1.95p)	1.3 분
	+Label Smoothing	**81.55(+3.02p)**	96.17(+2.35p)	1.3 분
	+Cosine Decay +Label Smoothing	81.20(+2.67p)	**96.44(+2.62p)**	1.4 분

Table 6. Cosine Decay, Label Smoothing







모든 구간에서 Top-1과 Top-5 모두 Base 모델 보다 더 나은 지표를 보며, 실험 1과 같이 초기 구간에서 Train Loss가 Base 보다 좋은 지표를 보이며, 실험 2와 같이 몇 Iteration 이후 Train Loss가 Base 모델보다 느리게 감소하는 모습을 볼 수 있다.

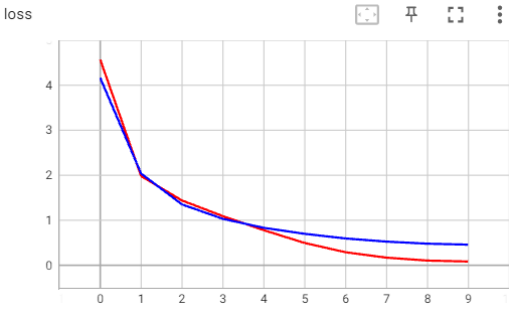
두 기법을 각각 단독 적용한 경우(실험 1, 2)보다 Epoch 당 학습 시간이 평균 0.1분 더 소요되었으며 Top-1에서는 실험 2의 결과보다 0.35p 낮은 지표가 나타났다. 반면 Top-5에서는 그 둘의 결과보다 최대 0.67p 더 높은 결과가 나타났다.

6. **Cosine Decay + Knowledge Distillation**

Base 모델에 Cosine Decay와 Knowledge Distillation 기법을 함께 적용 시 결과는 Table 7과 같으며 (a)~(c)에서 파란색으로 표시했다.

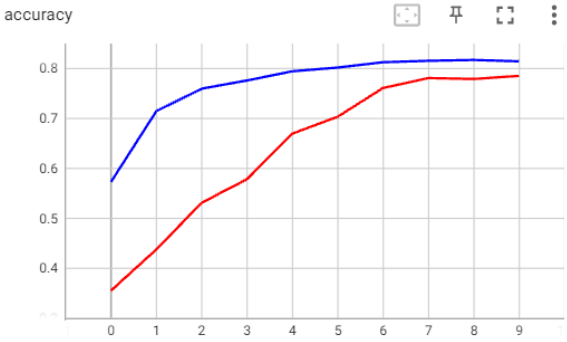
모델	기법	Top-1	Top-5	Time/Epoch
ResNet50	Base	78.53	93.82	**1.3 분**
	+Cosine Decay	79.13(+0.6p)	95.77(+1.95p)	1.3 분
	+Knowledge Distillation	81.36(+2.83p)	**96.90(+3.08p)**	1.4 분
	+Cosine Decay +Knowledge Distillation	**81.44(+2.91p)**	96.79(+2.97p)	1.5 분

Table 7. Cosine Decay, Knowledge Distillation

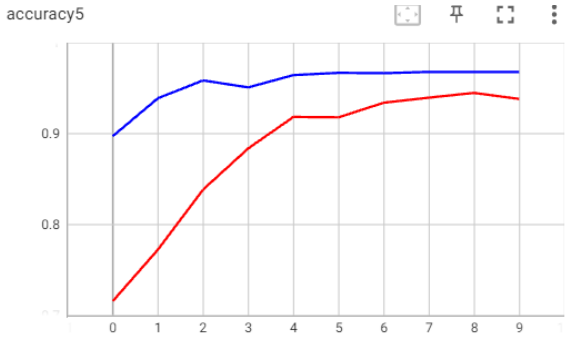


(a) Train Loss

모든 구간에서 Top-1과 Top-5 모두 Base 모델 보다 더 나은 지표를 보여주며, 실험 1보다는 더 높고 실험 3 보다는 더 낮지만 결국 Base 보다는 낮은 초기 Traing Loss를 갖는 것을 확인했다. 두 기법을 각각 단독 적용한 경우(실험 1, 3) 보다 Epoch 당 학습 시간이 평균 0.1분~0.2분 더 소요되었으다. Top-1에서는 실험 1, 3의 결과보다 최대 2.85p 높은 지표가 나타났다. 반면 Top-5에서는 실험 3의 결과가 0.11p 더 높은 지표를 보였다.



(b) Top-1 Validation Accuracy



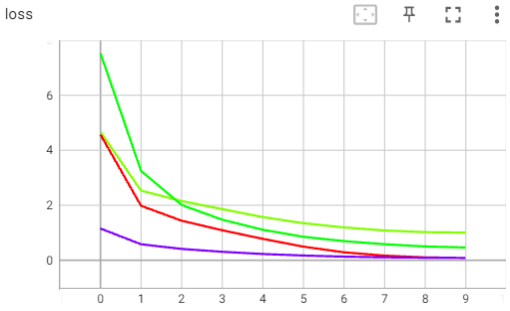
(c) Top-5 Validation Accuracy

7. **Label Smoothing + Knowledge Distillation**

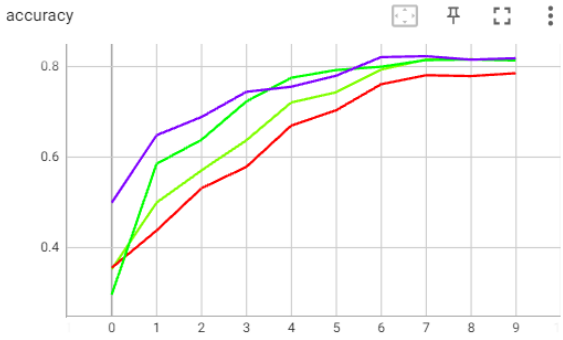
   Base 모델에 Label Smoothing과 Knowledge Distillation 기법을 함께 적용 시 III-3-(c)에서 언급한대로 Label Smoothing으로 학습한 Teacher 모델을 사용하도록 한다. 결과는 Table 8과 같으며 (a)~(c)에서 보라색으로 표시했다.

모델	기법	Top-1	Top-5	Time/Epoch
ResNet50	Base	78.53	93.82	1.3 분
	+Label Smoothing	81.55(+3.02p)	96.17(+2.35p)	1.3 분
	+Knowledge Distillation	81.36(+2.83p)	**96.90(+3.08p)**	1.4 분
	+Label Smoothing +Knowledge Distillation	**81.84(+3.31p)**	96.09(+2.27p)	1.5 분

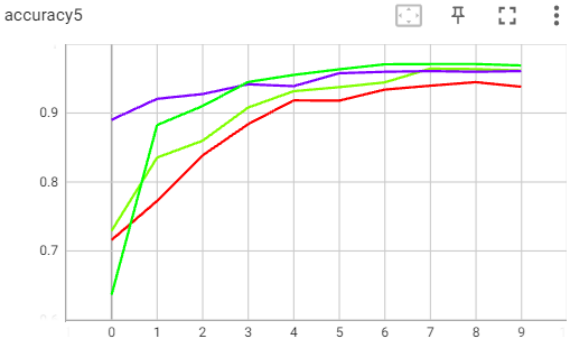
Table 8. Label Smoothing, Knowledge Distillation



(a) Train Loss



(b) Top-1 Validation Accuracy



(c) Top-5 Validation Accuracy

두 기법을 함께 적용한 경우, 실험 2(연녹색)와 3(녹색)에서 보인 것과는 달리 초기 Train Loss가 Base 모델보다 낮고, 몇 Iteration 이후 실험 2와 같이 느리게 감소하는 것을 확인할 수 있다.

Top-1에서 실험 2와 3의 경우 보다 최대 0.48p 더 높은 지표를 보였고, Top-5에선 Knowldege Distillation만을 적용한 실험 3이 본 실험 보다 오히려 더 0.81p 더 높은 지표를 보였다.

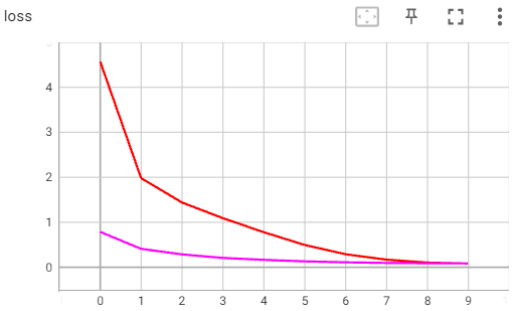
8. **Cosine Decay + Label Smoothing + Knowledge Distillation**

   Base 모델에 Cosine Decay, Label Smoothing, Knowledge Distillation 기법을 함께 적용 시 결과는 Table 9와

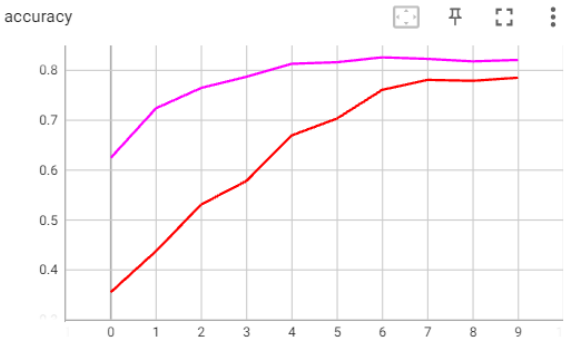
같으며 (a)~(c)에서 분홍색으로 표시했다.

모델	기법	Top-1	Top-5	Time/Epoch
ResNet50	Base	78.53	93.82	1.3 분
	+Cosine Decay +Label Smoothing +Knowledge Distillation	**82.09(+3.56p)**	**95.77(+1.95p)**	1.5 분

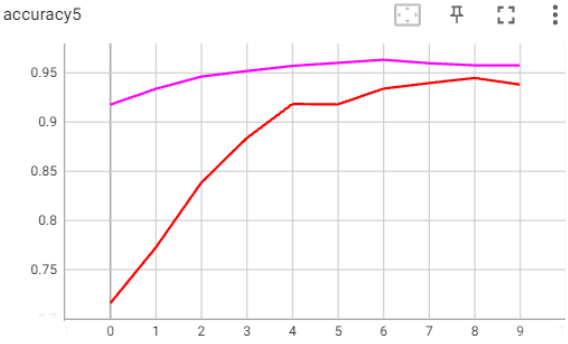
Table 9. Cosine Decay, Label Smoothing, Knowledge Distillation



(a) Train Loss



(b) Top-1 Validation Accuracy



(c) Top-5 Validation Accuracy

세 가지 기법을 함께 적용한 경우, 실험 7에서 확인한 것과 유사하게 (a)에서 모든 구간의 Train Loss가 Base 모델보다 낮고 완만하게 감소하는 것을 확인할 수 있다. 단, 실험 7의 경우 초기 Loss가 1.156이고 본 실험의 경우 0.787으로 더 낮은데, 낮은 초기 Loss를 보이는 Cosine Learning Rate Decay가 원인으로 보인다.

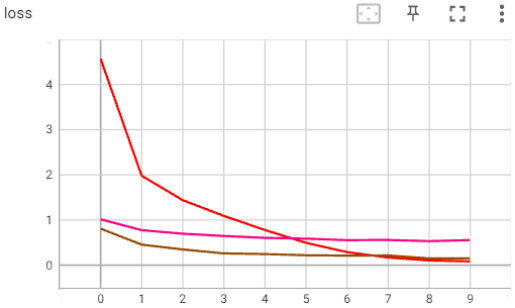
9. **Cosine Decay + Label Smoothing + Knowledge Distillation + Mixup (α=0.2, 0.01)**

   Base 모델에 Cosine Decay, Label Smoothing, Knowledge Distillation, Mixup (α=0.2 그리고 α=0.01) 기법을 함께 적용 시 결과는 Table 10과 같으며 (a)~(c)에서 각각 진분홍색과 갈색으로 표시했다.

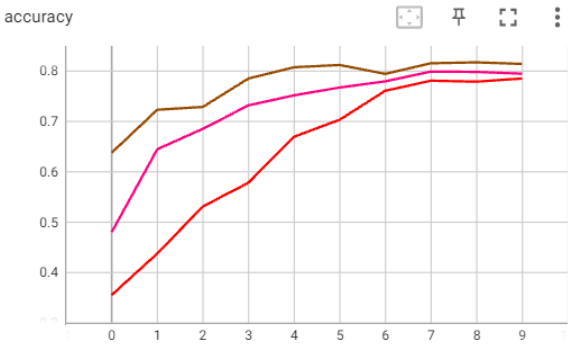
실험 7, 8과 같이 초기 Train Loss가 낮은 지점에서 시작하지만 감소 기울기가 Base보다 완만하여 Loss가 역전되는 상황이 관찰되었다. 단, 모든 구간에서 Top-1과 Top-5의 지표가 Base의 것보다 더 높았으며, 특히 α=0.01인 경우에서 더 높은 지표를 얻었다.

모델	기법	Top-1	Top-5	Time/Epoch
ResNet50	Base	78.53	93.82	1.3 분
	+Cosine Decay	79.49(+0.96p)	95.98(+2.16p)	2.5 분
	+Label Smoothing			
	+Knowledge Distillation			
	+Mixup α=0.2			
	+Mixup α=0.01	**81.43(+2.90p)**	**96.30(+2.48p)**	2.5 분

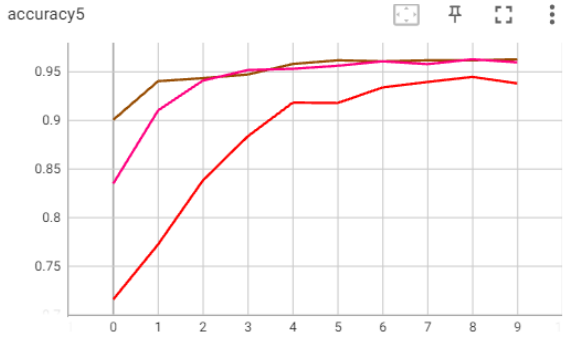
Table 10. Cosine Decay, Label Smoothing, Knowledge Distillation, Mixup (α=0.2 and 0.01)



(a) Train Loss



(b) Top-1 Validation Accuracy



(c) Top-5 Validation Accuracy

V. 결론

모델	기법	Top-1	Top-5	Time/Epoch
ResNet152	Teacher	80.49	96.61	1.4 분
	+Label Smoothing	82.78(2.28p)	96.34(-0.27p)	1.4 분
ResNet50	Base	78.53	93.82	1.3 분
	+Cosine Decay	79.13(+0.6p)	95.77(+1.95p)	1.3 분
	+Label Smoothing	81.55(+3.02p)	96.17(+2.35p)	1.3 분
	+Knowledge Distillation	81.36(+2.83p)	**96.90(+3.08p)**	1.4 분
	+Mixup α=1	5.27(-73.26p)	24.24(-69.58p)	2.5 분
	+Mixup α=0.2	35.53(-43.00p)	78.05(-15.77p)	2.5 분
	+Mixup α=0.1	51.92(-26.61p)	91.21(-2.61p)	2.5 분
	+Mixup α=0.01	80.13(+1.6p)	96.38(+2.56p)	2.5 분
	+Cosine Decay +Label Smoothing	81.20(+2.67p)	96.44(+2.62p)	1.4 분
	+Cosine Decay +Knowledge Distillation	81.44(+2.91p)	96.79(+2.97p)	1.5 분
	+Label Smoothing +Knowledge Distillation	81.84(+3.31p)	96.09(+2.27p)	1.5 분
	+Cosine Decay +Label Smoothing +Knowledge Distillation	**82.09(+3.56p)**	95.77(+1.95p)	1.5 분
	+Cosine Decay +Label Smoothing +Knowledge Distillation +Mixup α=0.2	79.49(+0.96p)	95.98(+2.16p)	2.5 분
	+Mixup α=0.01	81.43(+2.90p)	96.30(+2.48p)	2.5 분

Table 10. 실험 결과 정리 표.

실험 결과를 표 하나에 정리하여 나타내면 Table 10과 같다. Top-1 Validation Accuracy의 경우 실험 8에서의 기법 조합이 가장 우수한 지표를 보였고, Top-5 Validation Accuracy의 경우 실험 3의 기법만 적용한 경우 가장 우수한 지표를 보였다. Cosine Learning Rate Decay 기법은 적용하기 간편하면서도 성능을 쉽게 올릴 수 있던 것이 특징이었다. Label Smoothing 기법은 ResNet50에 학습했을 때와 달리 ResNet152에 학습했을 때의 결과에서 오히려 낮은 지표를 보이며 Knowledge Distillation 과 같은 기법과 같이 사용하기 어렵다는 특징이 있었으며 Train Loss 감소 기울기가 완만해 학습 시 많은 Epoch를 설정해야 할 것으로 예상된다. Knowledge Distillation 기법의 경우 성능을 높일 수 있

지만 Teacher Model을 준비해야 한다는 불편함이 있었다. Mixup 기법의 경우 학습에 적합한 파라미터 α를 잘 선정하지 않으면 모델의 성능이 오히려 크게 저하되는 특징이 있었다. 각 기법의 조합의 경우 각 기법을 단독으로 적용했을 때의 특징이 조합되어 나타나는 것이 보였다.

VI. 고찰

이번 실험은 ResNet에 논문에 소개된 여러가지 기법을 구현하고 적용해보며 결과를 비교하는 실험이었다. Cosine Learning Rate Schedule의 경우 참고 문헌 [1]에서 제시한 것과는 다르게 구현에 어려움을 겪어 Warmup 구간을 설정하지 않았으며 Mixup의 경우 파라미터 α에 따라 결과가 천차만별 이었고 다른 기법과는 다르게 많은 시간이 소요되었다. 단, 네 가지 기법 모두 모델의 성능을 향상시킬 수 있는 기법인 것은 확실하게 확인하게 되었다. 이후, 모델의 성능을 향상시켜야 할 상황이 생긴다면 Mixup 보단 다른 데이터 증강 기법을 이용할 것 같고, 더 많은 Epoch와 Label Smoothing으로 학습된 Teacher 모델과 함께 Knowledge Distillation과 Cosine Learning Rate Decay 기법을 적용하여 향상 시킬 것 같다.

VII. 참고 문헌

[1] He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 558-567).

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[3] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez- ´Paz. mixup: Beyond empirical risk minimization. CoRR, abs/1710.09412, 2017.

[4] SungJin Park, SeYoung Kim, and YoonYoung Lee. Model Optimization for Recycling Trash. https://github.com/boostcampaitech2/model-optimization-level3-cv-04, 2021.

[5] IBOK. [PyTorch] CosineAnnealingLR, CosineAnnealingWarmRestarts. https://bo-10000.tistory.com/88, 2021.

[6] 홍러닝. Label Smoothing. https://hongl.tistory.com/230, 2021.

[7] T. Ben, YongDuek Seo, and M. Aleksey. PyTorch Image Classification. https://github.com/bentrevett/pytorch-image-classification, 2021.