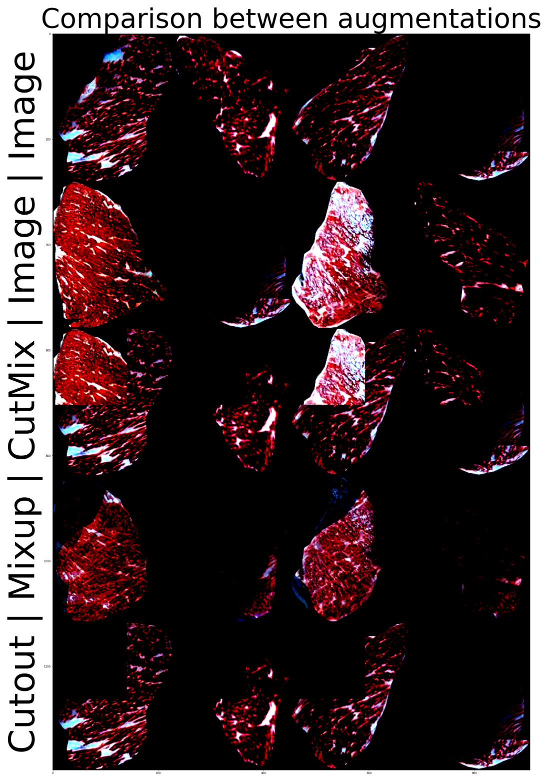
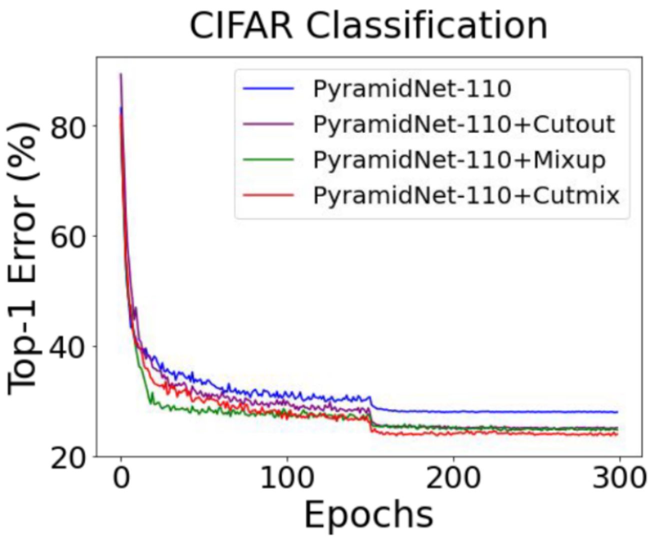
Cutmix Implementation

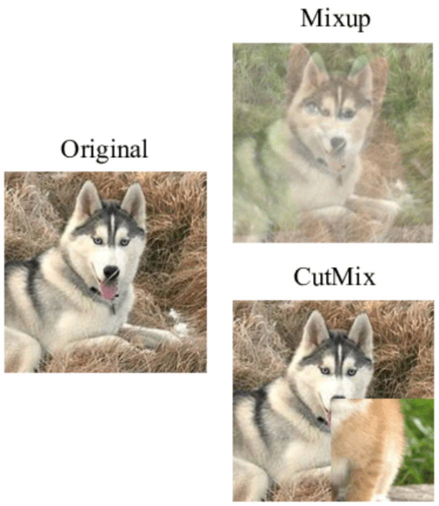
cutmix & mixup with resnet (on our data) code link:

https://colab.research.google.com/drive/1yJorYo1m_R8zAESdvlNnwArhPcOw2o02?authuser=1#scrollTo=Hhi37qcwVKye

Cutout, Mixup, Cutmix

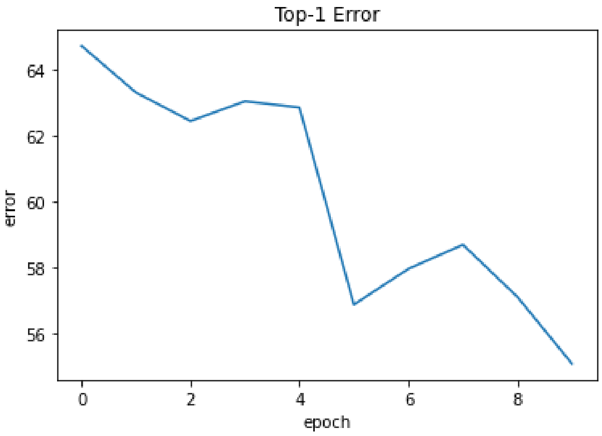

Comparison between augmentations
Cutout | Mixup | CutMix | Image | Image





🦋 Mix-up Implementation

[Cutmix]



Resnet으로 한우데이터셋에 적용한 error plot (epoch: 1~10)

Cutmix는 Randomness가 포함되고 이미지가 섞이는 것에 따라 손실을 계산할 때 필요한 람다값이 바뀌기 때문에 **Augmented 한 데이터셋**과 **변경하셔야 할 코드**로 첨부하겠습니다 🙂

Step 1: Augmented Dataset 불러오기 (train dataset3만 이미지 사이즈가 다릅니다!)

train dataset link1: https://drive.google.com/file/d/1-N54DtjJK4R0JhZO1dgUn9mAkewx4zVT/view?usp=sharing

train dataset link2: https://drive.google.com/file/d/1-0sVpudfhD922Wgwidlmt3RuZLX3WyCU/view?usp=sharing

train dataset link3: https://drive.google.com/file/d/1-3_9lkIhfMtgnKixwC1reBQUh4ASUXON/view?usp=sharing

train dataset link4: https://drive.google.com/file/d/1-6wo0tI4buUMKaeobO87WN28GIHtTdma/view?usp=sharing

(candidate train dataset link5) https://drive.google.com/file/d/1gKT7zqmfNoD965EU1xwtg-q6rMMsH44K/view?usp=sharing

(candidate train dataset link6) https://drive.google.com/file/d/1-NfobpSD6s9bSEUoddT2gKfuH_uTnwNP/view?usp=sharing

(candidate train dataset link7) https://drive.google.com/file/d/1-MEVLznocCb8chCK8wDRGWbiJ1iWv6UH/view?usp=sharing

Step 2: 다르게 augmented 된 train dataset을 원본 이미지 데이터와 함께 하나의 데이터셋 (=5만장)에 통일 (to use data = original image + train dataset1 +train dataset2 + train dataset3 +train dataset4)

Step 3: cut 함수 정의하기

```
def cut(W,H,lam):

    ######define the size of box######
    cut_rat = np.sqrt(1. - lam)
    cut_w = np.int64(W * cut_rat)
    cut_h = np.int64(H * cut_rat)
    ######define the size of box######

    #####randomly choose where to cut#####
    cx = np.random.randint(W) # uniform distribution
    cy = np.random.randint(H)
    #####randomly choose where to cut#####

    bbx1 = np.clip(cx - cut_w // 2, 0, W) # Cut, return coordinates of the box
    bby1 = np.clip(cy - cut_h // 2, 0, H)
    bbx2 = np.clip(cx + cut_w // 2, 0, W)
    bby2 = np.clip(cy + cut_h // 2, 0, H)

    return bbx1, bby1, bbx2, bby2
```

Step 4: Train function 안에 ####### 부분 코드 넣기

```
for images, labels in tqdm(train) :

        images, labels = images.to(device), labels.to(device)
        model = model.to(device)
        model.train()

        ##########여기서부터 복붙하시면 됩니다##########
        lam = np.random.beta(1.0, 1.0)
        rand_index = torch.randperm(images.size()[0])
        shuffled_labels = labels[rand_index]
        bbx1, bby1, bbx2, bby2 = cut(images.shape[2], images.shape[3], lam) # define a box to cut and mix
        images[:, :, bbx1:bbx2, bby1:bby2] = images[rand_index, :, bbx1:bbx2, bby1:bby2] #cut and mix
        lam = 1 - ((bbx2 - bbx1) * (bby2 - bby1) / (images.shape[-1] * images.shape[-2]))
        ########### 복붙 끝###########
```

Step 5: **Loss 계산 코드** 바꿔주기

```
loss = criterion(out, labels) * lam + criterion(out, shuffled_labels)*(1.0-lam)
```

(out은 ‘out = model(images)’로 model에서 나온 값입니다)

[*Appendix]*

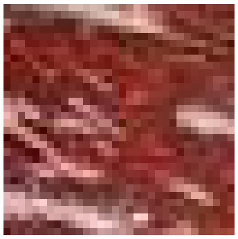
Augmentation method details:

1. Horizontal Flip + Random Rotation
2. Random Vertical Flip + Rotation
3. Random Vertical Flip + Center Crop

4. Random Vertical Flip + Color Jitter

augmentation method 1

augmentation method 2



augmentation method 3

augmentation method 4