- Anchor – 학습 데이터 (Anchor를 positive·negative로 분류) (pull together)
- positive – 긍정 데이터
- negative – 나머지 데이터

Positive samples P(i): Anchor의 긍정 데이터 & Anchor를 뺀 데이터

"under the umbrella of : ~의 범주 안에 있다.

hard negative : 실제로 negative 데이터 positive라고 잘못 예측하기 쉬운 데이터
(object detection에서 false negative 가 발생하는 이유는 positive 개체보다 비슷한
negative 개수가 많아서 (같은 공간 분포를 갖기 때문)

(hard negative mining: hard negative 데이터를 학습 데이터로 사용하기 위해 뽑는 (선택 (mining) 것)

- Representation learning (= Feature Learning)
  - 비록 각각 (classification / regression)를 수행한 각각 데이터를 효과적으로 설명한 점 (feature) 을
    기준으로 수행할 수 있도록 학습하는 과정
  - 충분한 Feature 요구됨. Softmax & cross entropy 에 깊음
    → supervised learning (충분 확보하는 classifier 부분 통해 이뤄짐.)

- Self-supervised Contrastive learning
  (Pre-training, Fine-training 으로 분리됨.)
  pre-training (사전학습) : 본 데이터를 (레이블 없는) 이용한 범용 모델 훈련학습.
  Fine-tuning (훈련조정) : pre-training된 모델 데이터에 분리된 후에 학습.

$$L = \sum_{i \in I} L_i^{self} = -\sum_{i \in I} \log \frac{\exp(z_i \cdot z_{j(i)}/\tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a/\tau)}.$$

i : anchor
j(i) : positive (하나)
k : negative (2(N-1) slices)
a : anchor 외 모든 데이터 (positive 1개 + negative 2N-2 개 = 2N-1개)

($z_i \cdot z_{j(i)}$) ($z_i \cdot z_a$): 각각 anchor 와 positive, anchor 와 positive, negatives 간 유사도

○ Supervised Contrastive learning
  - 같은 레이블 데이터 유사하도록 훈련학습 (→pull together)
  - 다른 레이블 데이터에 최대한 멀리 훈련학습 (→push apart)

1. Augmentation $Aug(x) = \tilde{x}$
2. Encoder $Enc(x,x) = r = own$ 영역 또 레벨에서 특징 추출. (feature extraction의 과정)
   └ feature extraction 에서 Global average pooling을 적용하여 통합되도록 수행 한다. $Enc(\cdot)$ 구성.
3. projection $Proj(\cdot) = z =$ DNN 영역 또 L_2 정규화된 수행 추출. (supervised Contrastive loss 과정)

$$\mathcal{L}_{out}^{sup} = \sum_{i \in I} \mathcal{L}_{out,i}^{sup} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p/\tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a/\tau)}$$

$P(i)$: positives (positive sample 데이터)
$|P(i)|$: cardinality (the number of positive samples)

$z_i \cdot z_p = z_p^T z_p$

같은 label의 positive 데이터

이것이 낮아지게 하려면
분자는 작아지고
분모는 ↑, 분자는 ↓
positive sample 데이터

negative sample 데이터
(커진다)
positive sample 데이터
(커짐)

Supervised Learning
- 일반적인 정답을 부여해 학습하는 딥러닝 기법
- labeling된 데이터 부족 시 본 성능을 기대하기 어려움

Self-supervised Contrastive Learning
- 데이터가 부족한 상황을 극복하기 위한 representation learning 기법
- Data augmentation을 통한 pretraining 및 Fine tuning

Supervised Contrastive Learning
- Label 활용 supervised learning
- Data augmentation을 활용한 pretraining + Fine tuning의
  contrastive learning

encoder와 projector를 통해서 Image representation된 vector를 생성) Stage 1
Back-propagation 과정에서 Contrastive loss function을 통해 vector 값을 업데이트)
이후 pretrain이 끝나면 해당 stage를 고정시킨 다음, Stage 2에서 classifier를 통한 이미지 분류 학습함) Stage 2

[SupContrast]

< main_supcon 이름 >

aug_1, aug_2
images = torch.cat ([aug_1, aug_2])
features = model (images)
loss = criterion (features, labels)

< Criterion >

{ features = [bsz, n_views,] # [256, 2, 128]
{ labels = [bsz]
{ mask = [bsz, bsz]

labels = labels.contiguous.view(-1,1)   # (256,1)
mask = torch.eq (labels, labels.T).float() # [256,256] → 각 생플쌍의 동일한 클래스 여부를 나타냄. → Mask[i,j] 가 동일한 클래스이면 1, 아니면 0

inputs로 들어온 features = # (256,2,128)
contrast_count = features.shape[1] #2.
contrast_feature = torch.cat (torch.unbind (features, dim=1), dim=0) # [512,128]

anchor_feature = contrast_feature
anchor_count = contrast_count

anchor_dot_contrast = $z_i \cdot z_p^T / \tau$ → anchor sample과 contrastive sample간의 유사도 측정 # (512,512)

logits_max, _ = torch.max (anchor_dot_contrast, dim=1) → anchor-dot-contrast에서 각 행의 최댓값을 구함, # (512,1)
                                                          각 anchor sample에 대한 최대 유사도 값을 구함

logits = anchor_dot_contrast - logits_max → 최댓값을 빼서 수치 안정성 보장   # (512,512)

mask = mask.repeat (anchor_count, contrast_count) # (512,512)
logits_mask = torch.scatter (torch.ones_like(mask), 1,
                torch.arange(batch_size * anchor_count).view(-1,1)), 0)
                                                    └ 0에서 511까지 정수 나열.

mask = mask × logits_mask → [i][j] 동일한 예인 1로 뽑되 대각 마스크 뽑혀 (자기 자신도 제외)
exp_logits = torch.exp(logits) * logits_mask → 각 앵커 샘플과 대조 샘플간의 유사도 계산
            └ 각 앵커 샘플에 대한 확률값의 합산
log_prob = logits - torch.log (exp_logits.sum(1)) → 앵커 샘플과 대조 샘플 간의 로그 확률 값 계산   # (512,512)

mean_log_prob_pos = (mask × log_prob).sum(1) /mask.sum(1) # (512)
                    └ 양성 샘플들의 로그 확률만 반영   └ 각 샘플의 양성 샘플수 인덱 → 양성 샘플들의 로그 확률 평균 양성 샘플 수로 나누어
                                                                            평균 로그 확률 계산 → 모델이 representation learning 학습하는데에 사용
                                                                            < 양성 샘플과 연관된 정보를 평균으로 묶어서 이
                                                                            학습하는 평가하는데에 사용 >

loss = - (self.temperature /self.base_temperature) × mean_log_prob_pos → 양자 평균 평균 로그 확률을 손실 값으로 변환하여
                                                                          더 높은 평균 로그 확률을 만든 방향으로 모델을 학습하기 위한
loss = loss.view (anchor_count, batch_size).mean()   음으로 변환

[우측 행렬 그림]
                512
[0 1 1 ···· 1]
[1 0 ···· (1]
[1 1 0 ···· 1] 512 → 자기 자신과 비교하는 유사도를
[1 1 1 0 ···· 1]   걸러 줄 수 있으므로 함.
[⋮]
[1 1 1 ···· 0]

$$\mathcal{L}_{out}^{sup} = \sum_{i \in I} \mathcal{L}_{out,i}^{sup} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp (z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp (z_i \cdot z_a / \tau)}$$

$P(i) = $ positives $\equiv \{ p \in A(i) : \tilde{y}_p = \tilde{y}_i \}$
$i \in I = \{1, \cdots 2N\}$ augmented samples
$\tilde{i} = $ anchor
$A(i) = I / \{i\}$