NUVILAB × boostcamp ai tech

프로젝트 제안서
(Project Development Proposal)

프로젝트명	Embedding space 분석과 음식 사전을 이용한 Food CLIP 성능 개선
작성일	2022.12.11
작성자	김기용, 김성수, 김주엽, 이 구, 이태희
팀 명	Boostcamp AI tech 4기 CV 7조 (BLACKBOX)

목차

1. 제안 요약

제 안 서 요 약			
과제명	Embedding space 분석과 음식 사전을 이용한 food CLIP 성능 개선		
개발목표	Food CLIP 모델 성능 개선		
개발내용 요약	Food CLIP에서의 hard-negative sampling 기법 개발		
연구기관	NUVILAB, Boostcamp AI Tech		
연구기간	1 개월	**총투입인력**	5 명

2. 개발 개요

2.1. 개발 범위 및 목표

구분	개발 범위 및 목표	개발주체	비고
SW개발	Food CLIP improvement with effective hard negative sampling - CLIP[1] 모델이 sample의 fine-grained feature간 차이를 잘 학습할 수 있게 하는 hard-negative sampling 기법 개발 - Synthetic hard negative 생성을 통한 food CLIP[1] 학습 설계	AI-tech CV 7조	

2.2. 배경 지식

2.2.1. CLIP[1]

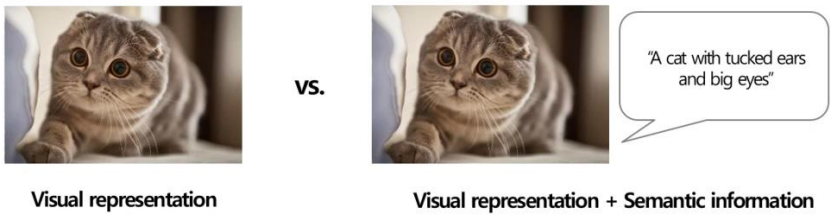


그림 1 기존 이미지 분류 방법과 CLIP[1]의 차이점

CLIP[1](Contrastive Language-Image Pre-Training)은 기존 Supervised learning 과 달리 Image-Text pair 를 기반으로 Visual representation 과 Semantic information 을 통해 이미지와 언어의 일반화된 특징을 학습한다. Image-Text pair 를 학습하는 기법은 Image captioning 과 같이 이전부터 사용이 되었지만, 모델의 크기로 인해 학습 시간이 길어 비효율적이라는 단점이 있다. CLIP[1]은 기존의 방식보다 효율적인 방법인 contrastive learning 을 적용하여

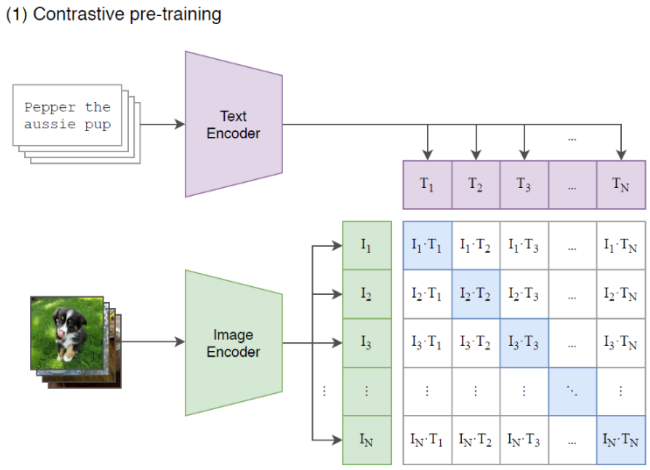


그림 2 Contrastive pre-training

pre-training 을 진행한다. Contrastive learning 을 통해 Image-Text pair 의 Positive sample 과 Negative sample 간의 유사도 관계를 학습한다.

CLIP[1]은 pre-training 을 위해 배치 단위로 이루어진 N 개의 Image-Text pair 에 대한 Embedding 벡터를 산출한다. 그림 2 와 같이 N 장의 이미지와 텍스트가 각각의 encoder 로 들어가게 되고 N 개의 벡터를 산출한다. 학습을 위해 자기를 제외한 이미지와 텍스트는 자신과 다르다는 관계를 이용한다. 그림 2 에서 파란색으로 표시된 행렬의 원소를 보게 되면 i번째 이미지는 i번째 텍스트만 유사도를 가지며, 따라서 해당 텍스트를 정답으로 취급한다. 즉, 총 N^2의 행렬 원소 중 N개는 Positive pair 로 취급하고, 그 외의 $N^2 - N$개의 원소는 Negative pair 로 취급한다. 이미지와 텍스트 벡터 간의 내적을 통해 Cosine 유사도를 계산하고, 각 유사도는 1~N 중에서 하나의 label 을 가진 것으로 해석할 수 있다.

CLIP[1]은 이미지를 pre-training 된 Image Encoder 에 통과시킨 후 텍스트와의 유사도를 계산한다. 이미지에 대한 유사도가 상대적으로 높은 텍스트를 선택하고 해당 텍스트를

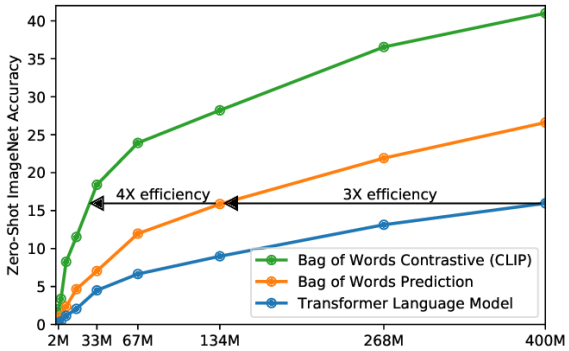


그림 3 CLIP[1]과 image captioning 간 zero shot prediction accuracy 비교

이미지에 대한 정답으로 예측한다. 즉, 예측하고자 하는 label 을 간단히 정의해 줌으로써

fine-tuning 과정을 거치지 않고 학습에서 한 번도 보지 못한 데이터셋에 대한 분류를 수행할 수 있게 된다.

그림 3 은 Image captioning 과 CLIP[1]을 이용한 Zero shot prediction 을 비교한 것이다. CLIP[1]이 image captioning 보다 이미지가 적은 데이터로 가장 높은 예측 정확도를 보이고 효율적인 pre-training 이 가능하다는 것을 확인할 수 있다. 또한 CLIP[1]은 보다 일반적인 데이터의 특성을 학습하여 generalization 측면에서 기존 ResNet 모델보다 더 robust 하다.

CLIP[1]은 음식 분류에 활용되기도 한다. 대표적인 예시로 LaunchPad의 Food CLIP[1] 연구가 있다. LaunchPad는 노이즈가 적은(Curated) 데이터를 사용하더라도 음식의 유형을 제대로 파악하기 어려울 수 있고 일부 음식에 대해 pre-training이 되지 않는 문제점을 해결하기 위해 자체적인 데이터셋을 구성하여 CLIP[1]의 성능을 개선 하였다.

	ResNet50 w/ Single Layer Head		CLIP	
	Frozen Model*	Fine-Tuned Model**	Zero-Shot	Linear Probe
Custom A: Industry	80.00%	85.00%	88.75%	97.22%
Custom B: Bing (curated)	64.22%	67.74%	82.98%	88.27%
Custom C: Tasteatlas	42.41%	60.04%	37.15%	74.56%
Custom D: Vegetable Cuts	73.93%	82.34%	67.04%	90.89%
Custom E: Cuts Mixed	75.59%	81.93%	39.92%	89.71%
Custom F: Exotic Foods	65.00%	80.56%	74.50%	90.55%

* ResNet50 model with standard ImageNet weights (all layers before model head are frozen); model head is a single-layer logistic regression output layer.
** ResNet50 model with fine-tuned ImageNet weights; model head is a single-layer logistic regression output layer.

	ResNet50 w/ Single Layer Head		CLIP	
	Frozen Model*	Fine-Tuned Model**	Zero-Shot	Linear Probe
Custom A: Industry	90.00%	91.25%	96.67%	100.00%
Custom B: Bing (curated)	81.23%	86.22%	94.53%	97.65%
Custom C: Tasteatlas	54.01%	72.88%	51.15%	87.39%
Custom D: Vegetable Cuts	87.17%	92.69%	85.21%	97.37%
Custom E: Cuts Mixed	91.45%	95.45%	56.15%	97.74%
Custom F: Exotic Foods	76.67%	87.22%	81.17%	96.66%

* ResNet50 model with standard ImageNet weights (all layers before model head are frozen); model head is a single-layer logistic regression output layer.
** ResNet50 model with fine-tuned ImageNet weights; model head is a single-layer logistic regression output layer).

그림 4 Top-1 Accuracy(좌)와 Top-2 Accuracy(우)

그림 4는 LaunchPad Food CLIP 모델의 zero-shot과 linear probing accuracy를 ResNet50과 비교한 것이다. 그림 4에서 확인할 수 있듯 각 데이터셋에서 Linear probe 기반 CLIP[1] 모델이 우수한 성능을 보인다. 또한 CLIP[1]의 Zero-Shot 기반 모델이 일부 데이터셋에 대해 ResNet50 baseline 모델보다 높은 성능을 보였다.

2.2.2. N-shot learning

최근 들어 Google Brain, Meta AI Research, OpenAI 등의 거대 연구집단에서는 강력한 컴퓨팅 파워를 기반으로 한 거대 모델들을 발표하고 있다. 특히, 최근에 발표된 Open AI 의 DALL-E 2[2], Google Brain 의 Imagen[3] 등의 text-to-image 생성 모델은 인간의 작업물과 유사한 품질의 이미지를 만들어낼 수 있는 수준에 도달하였음을 보여주었다.

Teddy bears swimming at the Olympics 400m Butterfly event. A cute corgi lives in a house made out of sushi. A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

그림 5 Imagen 의 text-to-image 생성 결과

이처럼 거대한 모델들의 뛰어난 성능은 막대한 양의 데이터에 기반한다. 하지만, 현실에는 labeling 되어있지 않은 데이터가 대부분이다. 데이터 labeling 을 위해 필요한 시간과 비용에 대한 제약이 존재하며, 분야에 따라 labeling 을 위해 전문 지식이 필요한 경우(ex. Biomedical)도 있다. 심지어, 비용과 시간 부담이 가능할지라도 충분한 양의 데이터를 얻기 어려운 경우(ex. 자연 재해, 희귀병)도 존재한다. 이러한 이유로, 적은 양의 데이터로 모델을 학습하는 방법에 대한 연구인 N-Shot learning 에 대한 관심이 점점 많아지고 있다.

N-Shot learning 이란, Few-Shot learning, One-Shot learning, Zero-Shot learning 을 모두 포함하는 개념이다. Few-Shot learning 은 학습을 위한 데이터인 support set 과 평가를 위한

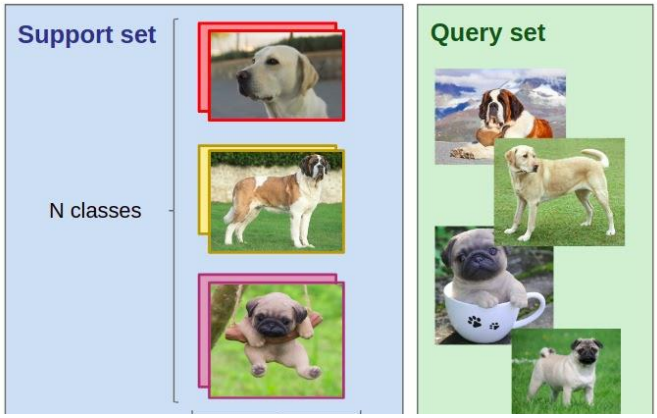


그림 6 Few-Shot learning 에서의 데이터 구성 예시

데이터인 query set 을 이용하여 진행된다. Support Set 이 포함하는 클래스 label 의 수를 N, 각 클래스에 대해 K 장의 이미지가 존재할 때, 이를 N-Way K-Shot Classification Task 라 부른다. Support Set 의 구성에 따라 task 의 난이도가 바뀌게 되는데, N 이 커질수록, K 가 작아질수록 어려운 task 가 된다. 보통 support set 의 각 클래스마다 2~5 장의 이미지를 사용하며, K 가 1 인 경우를 One-Shot learning 이라 부른다.

Few-Shot learning 의 대표적인 접근 방식으로는 data-driven approach(=data level approach)와 model-based approach(=parameter level approach)가 존재한다. Data-driven approach 는 support set (train set)으로 주어진 데이터에 transformation 을 적용하거나, GAN 등의 생성모델을 이용하여 모델을 학습시킬 수 있는 충분한 양의 학습 데이터를 생성하는 방법이다. Model-based approach 는 모델이 같은 클래스의 이미지와 서로 다른 클래스의 이미지를 구분할 수 있도록 feature vector 간의 similarity 를 학습하게 하거나, 적은 양의 data 에 model 이 overfitting 되지 않도록 regularization 등을 도입하는 방식이다. 이때, data-driven approach 의 경우 방법이 간단하고 직관적이지만, support set 전체 데이터의 모집단을 보장할 수 없다는 명확한 한계를 갖는다.

Zero-Shot learning 에서는 One-Shot, Few-Shot learning 과는 달리 단 한 장의 학습 이미지도 사용하지 않는다. 그 대신, Label 이 지정된 소수의 클래스 집합 데이터와 클래스에 대한 추가 정보만을 사용하여 한번도 본 적 없는 많은 클래스까지 잘 예측할 수 있도록 학습한다. 추가 정보로는 주로 attributes, hierarchy similarity measure 등을 이용한다. Attribute 의 경우, 클래스들의 특징을 사용하여 클래스 간의 유사도를 계산하며, 이를 통해 학습하지 않은 클래스의 label 을 예측하는데 사용한다. Hierarchy similarity measure 의 경우 각 클래스들의 계층적인 관계를 tree 구조로 표현한 후, 클래스들 사이의 similarity 정보를 사용한다.

이러한 Zero-Shot learning 의 목적은 이미 알고 있는 클래스의 데이터와 처음 보는 클래스의 데이터 모두를 올바른 클래스로 분류하는 것이다. 현실에서 label 정보가 존재하지 않는 데이터의 label 을 올바르게 예측하는 것이 중요한 task 의 경우 zero-shot learning 을 적용함으로써 이를 실현할 수 있을 것이다. 실제로 zero-shot learning 은 앞서 언급한 image classification 뿐만 아니라, semantic segmentation, image generation, object detection, image retrieval, natural language processing, action recognition 등의 다양한 분야에도 적용될 수 있다.

2.2.3. Self-supervised learning

세상의 대부분의 data 에는 label 이 존재하지 않기에 unlabeled dataset 을 가지고도 높은 성능의 모델을 만들기 위해 Self-supervised learning 이 고안되었다. Self-supervised learning 은 pretext task 를 정의하고, unlabeled dataset 을 사용하여 pretext task 에 대해 모델을 학습시킨다. 이렇게 학습시킨 모델을 downstream task 에 대해 transfer learning 하여 모델의 성능을 향상시킬 수 있는데, 대표적인 최근 연구 사례로 MAE[5][5](Masked Auto Encoder)와 DINO[4][4] (Self-distillation with no labels) 가 있다. 두 연구 사례를 통해, self- supervised learning 의 구체적인 과정을 살펴보고 개념을 이해한다.

① DINO[4] (Self-distillation with no labels)

DINO[4]는 unlabeled data 를 self-distillation 방식으로 학습시키는 방법론을 제안하였다. Idea 의 핵심은 같은 image 에 대해 다른 augmentation 을 적용시켜도 모델이 동일한 feature 를 출력하도록 유도하는 것이다. 기존의 knowledge distillation 방식이 teacher model 의 출력인 soft-label 로 student model 을 supervised 한 것과 달리, DINO[4]는 하나의 모델이 student, teacher 의 역할을 동시에 수행한다. 즉, 이미지에 서로 다른 두 가지의 augmentation 을 적용하여 모델에 통과시킨 두 개의 prediction probability 가 유사하도록 학습한다. 이를 통해, 이미지에 다양한 variation 이 발생하여도 이를 같은 이미지로 판별할 수 있는 분별력을 기르게 되어 model 은 label 의 도움 없이도 이미지 본연의 feature 를 학습하게 된다.

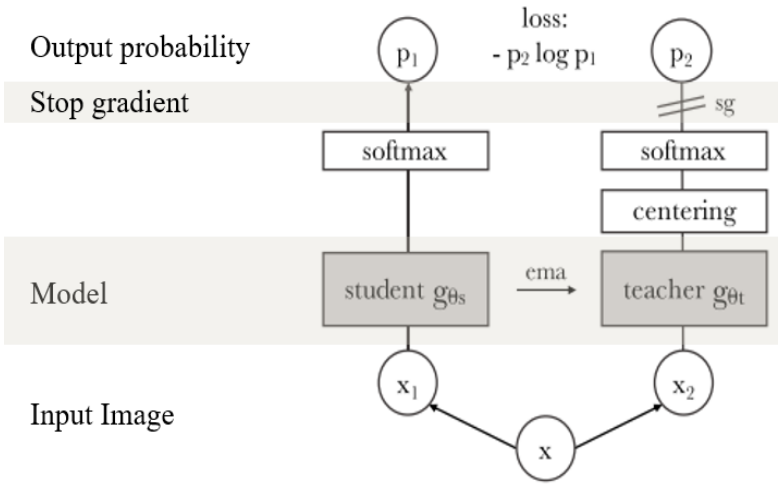


그림 7 DINO[4]의 self-supervised learning 구조

그림 7 은 DINO[4]의 학습 구조를 보여준다. 먼저, Input image x 는 서로 다른 두 개의 augmentation 을 거쳐 x1, x2 로 변환된 후 각각 student 와 teacher 모델로 입력된다. 이때 student 와 teacher 는 동일한 모델이나, student 모델은 학습 과정 속에서 weight 가

update 되는 반면 teacher 는 student model parameter 의 exponential moving average(ema)를 계산한 값으로 weight 를 update 한다. 즉, teacher 모델은 gradient update 가 중단된 상태로 student model 의 과거 학습 정보만을 distillation 받는 student 모델의 또 다른 모습이며, 이러한 이유로 self-distillation 이란 이름이 붙여졌다. DINO[4] 구조에선 x1, x2 에 대해 얻은 두 개의 확률 분포가 유사해지도록 cross-entropy loss 를 사용하는데, 두 출력에 공통적으로 포함된 이미지 본연의 feature 를 학습하는 데에 집중하게 된다.

② MAE[5] (Masked Auto Encoder)

NLP 분야에서 mask 를 활용한 self-supervised learning 기법은 GPT[6] 와 BERT[7]에서 큰 성공을 거두었다. ViT 의 등장 이후로 NLP 모델을 응용한 Vision task 의 연구 또한 활발히 진행되었는데, Masked Auto Encoder 는 이미지에 대해 Mask 를 적용하여 이미지를 효율적으로 학습하는 방법을 제안한다.

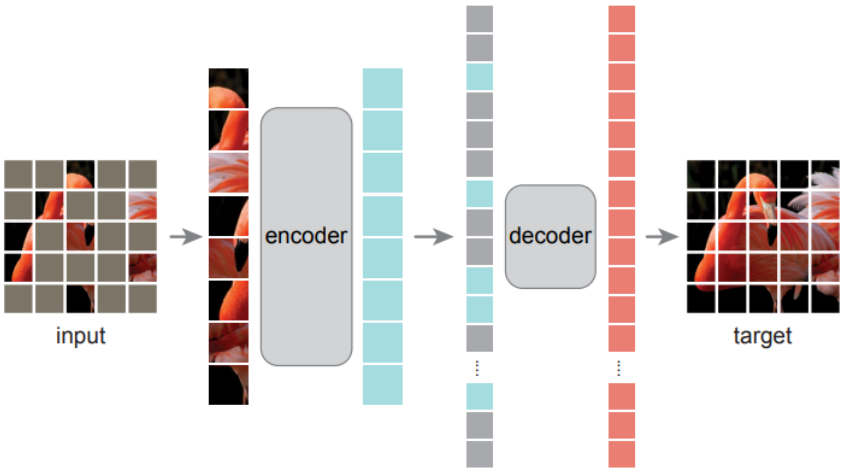


그림 8 MAE[5] architecture

MAE[5]는 Masked patches 를 원본 이미지로 재구성하는 self-supervised 방식을 통해 이미지에 대한 일반화된 특성을 학습하였다. 위 그림 8 은 MAE[5] Architecture 를 보여준다. Encoder 와 Decoder 는 비대칭 구조이며, 이미지를 Patch 로 분리하여 각 Patch 에 대한 Latent representation 을 학습한다. Encoder 는 Visible patches 에 대해서만 Encoding 하므로 시간과 메모리의 효율성이 대폭 향상하였다. Decoder 는 Linear projection 을 통해 평균과 표준편차를 고려한 Pixel 정보를 예측하여 이미지를 재구성한다. 또한, Masked patch 의 재구성만을 목적으로 하고, Encoder 와 독립적으로 설계할 수 있으므로 작고 가벼운 모델의 Decoder 로 구성한다. Masked patch 를 재구성하는 과정에서 MSE Loss 를 통해 Visible patch 와 비교함으로써 Latent representation 을 학습한다.

2.2.4. Contrastive learning

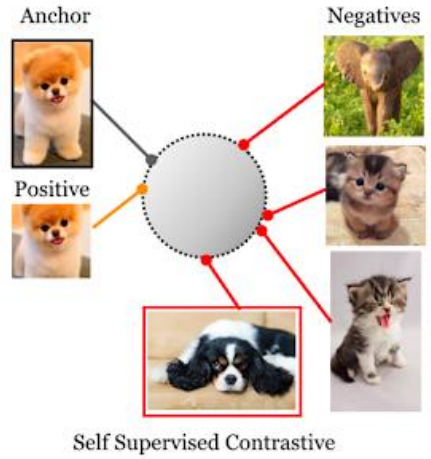


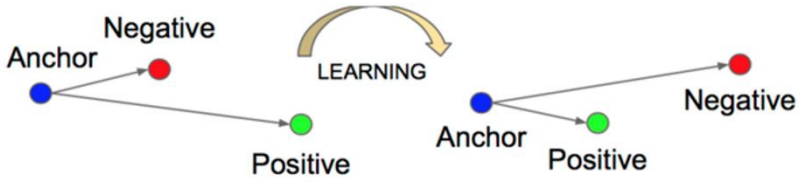
그림 9 Self-supervised contrastive

Contrastive learning 은 self-supervised learning 에서 latent feature 를 embedding space 상에서 학습하는 방법론이다. Contrastive loss 는 deep metric learning 초기 objective 로, pair 간 유사도를 Euclidean distance 또는 cosine similarity 로 측정한 뒤 anchor-positive pair 간의 거리가 가까워지는 방향으로, anchor-negative pair 간 거리가 margin 이상이 되도록 학습한다.

이러한 Contrastive learning 에서 negative batch 중 anchor 와 유사한 것을 학습할수록 representation 을 더 잘 학습할 수 있을 것이라는 아이디어와 함께 등장한 개념이 Hard negative sampling 이다. Anchor 와 다른 label 을 가지지만 유사한 특징을 가져 구별하기 어려운 example 들을 hard negative sample 로 정의한다.

Contrastive learning 을 구체적으로 이해하기 위해 metric learning 의 loss 중 하나인 triplet loss(supervised)를 예시로 embedding space 상의 feature vector 를 학습하는 방안을 소개한다. 동시에, 적절한 anchor, positive, negative sample 을 선정하여 mini-batch 를 구성하는 hard negative sampling[8]의 동작 방식을 살펴본다.

① Triplet loss



$$L(A,P,N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

A : anchor input
P : positive input
N : negative input
α : positive, negative pair 사이의 margin
f : embedding

그림 10 Triplet loss 의 objective function

Triplet loss 는 최종 feature vector 에 대해 embedding space 상에서 그림 10 과 같은 objective 를 가진다. (Positive 는 anchor 와 동일한 class 를 가지며, negative 는 anchor 와 다른 class 를 갖는다.) Anchor-negative 간의 거리가 anchor-positive 간의 거리와 margin 의 합보다 클 때 loss 는 0 이 된다.

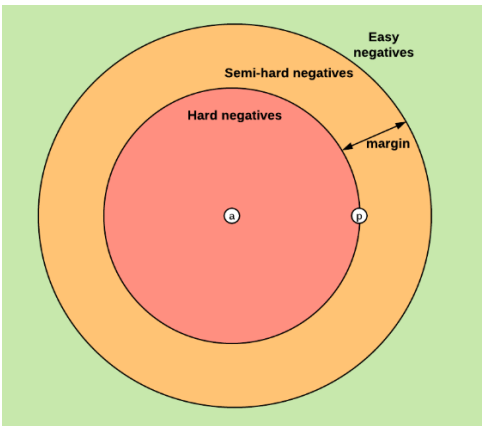


그림 11 Negative sample boundary

Triplet loss 에서 적절한 triplet (anchor, positive, negative)의 선정하는 것은 중요하다. 그림 11 은 anchor 와 positive 의 위치가 고정되어 있을 때 negative 의 종류를 distance 에 따라 구분한 것이다. 이 중 hard negative 는 embedding space 상에서 positive 보다 anchor 와 더 가깝게 projection 되는 sample 을 의미한다. Hard negative 는 anchor 와 다른 class 이지만 매우 유사한 특징을 가지기 때문에, model 은 anchor 와 거리를 벌리기 위해서 anchor-negative 간 미세한 차이점에 집중하여 학습하게 된다. 따라서, hard negative 를 잘 선정할수록 class 의 fine-grained feature 학습에 유리하다.

② Hard triplet mining[8]

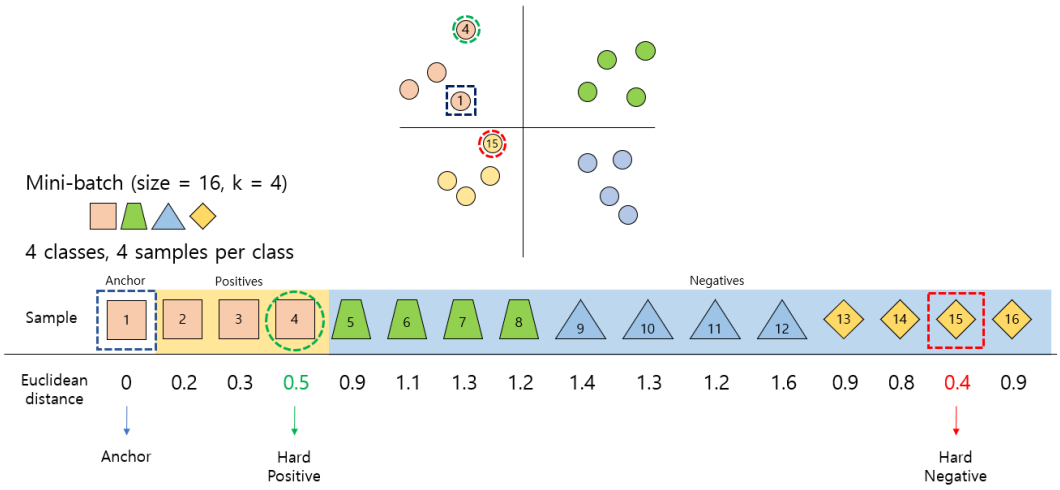


그림 12 Mini-batch hard triplet mining[8]

앞서 언급했듯, 적절한 triplet (anchor, positive, negative)를 선정하는 것은 triplet loss 의 수렴에 중요하다. 그림 12 는 mini batch 내에서 triplet 을 선정하는 대표적인 방법 중 하나인 online triplet mining 에 대한 그림이다. 예시는 batch size 가 16 일 때 4 개의 class 에 대해 각각 4 개의 sample 이 할당되어 있는 모습이며, 각 sample 을 1 ~ 16 번으로 표기하였다. Triplet 은 각 sample 에 대해 결정되므로, 하나의 mini-batch 에 대해 총 16 개의 triplet 을 구성하여 학습하게 된다. 그림 12 는 1 번 sample 이 anchor 인 경우를 나타내며, 나머지 15 개의 sample 각각에 embedding space 상에서 Euclidean distance 를 구한다. 이 후 1 번과 같은 class 를 갖는 sample 중 가장 거리가 먼 sample 을 hard positive 로, 다른 class 를 갖는 sample 중 가장 거리가 가까운 sample 을 hard negative 로 선정한다. 이렇게 triplet 을 구성하여 학습할 경우, 같은 class 내의 sample 들의 representation 에 대한 variance 를 최소화하면서 비슷한 특성을 지닌 negative 에 대한 분별력을 최대화할 수 있다. 이러한 hard triplet mining 방법은 metric-learning 의 분야 중 하나인 person re-identification 에도 적극 활용되고 있다.

Food CLIP 에서 비슷한 특성을 지닌 음식간(ex. 김치찜, 김치찌개) 차이점을 모델이 학습하고 이를 분별해내는 것은 중요하다. 따라서, 비슷한 특성을 가진 음식 sample 들이 학습 단계에서 같은 batch 내에 sampling 되어 이들 간 미세한 차이를 학습하도록 하는 것이 필수적이다.

2.2.5. Curriculum learning[9]

일반적인 딥러닝 프로세스에서는 전체 dataset 을 batch 단위로 나누어서 학습을 진행하며 각 batch 는 random 한 순서로 구성된다. 인간의 경우 쉬운 example 부터

학습하고 점차 어려운 example 까지 학습해 나간다. 이런 인간의 학습 프로세스를 모방한 것이 바로 Curriculum learning 이다.

Curriculum Learning[9]은 Continuation method 의 아이디어를 기반으로 등장했다. Continuation method 란 non-convex objective function 의 global optimum 에 근사한 해를 찾는 optimization 방법론이다. objective function 의 smooth version 을 찾아 해를 찾기 쉬운 convex function 으로 만들어준다. 점진적으로 highly smoothed version 부터 less smoothed version 까지 진행하며 최적해를 찾기 쉬운 objective 부터 어려운 objective function 까지 순차적으로 최적화하고 최종적으로 원래 objective 의 global optimum 에 근접한 해를 찾아낸다.

Curriculum learning[9]에서는 사전지식을 통해 data 를 난이도에 따라 정렬하고, 원, 정삼각형, 정사각형과 같이 쉬운 sample 부터 학습시켜 global picture 를 잡아 나간다. 이후 타원, 삼각형, 직사각형과 같은 어려운 sample 을 점차 추가하고 최종적인 target training distribution 을 학습하게 된다. 이러한 알고리즘이 model 이 local minimum 에 빠질 경향성을 감소시키며 수렴 속도도 향상시키고 결과적으로 모델의 성능도 향상시킬 수 있다고 한다.

하지만 대부분의 Real world data 에는 학습 순서를 정해주기 위해 필요한 학습 난이도의 pre-knowledge 가 없으므로 Curriculum learning[9]을 바로 적용할 수 없다는 것이 문제가 된다. 이 문제를 해결하기 위해 Self-paced learning[10]이 고안되었다. 일반적으로는 각 sample 에 대한 loss 를 계산하고 전체 data 에 대한 loss function 을 최소화하는 방향으로 parameter 를 학습한다. 이때 Self-paced learning[10]은 사전지식 대신 각 sample 의 loss 를 학습 난이도로써 사용하는 것이다. 임계값 이하의 loss 를 가지는 sample 들을 쉬운 example 로 보고 학습에 우선 사용하게 된다. 임계값을 단조적으로 증가시키며 학습을 진행하고 최종적으로는 전체 dataset 을 학습에 사용하게 된다. 아래와 같이 loss function 에 변수 vi 를 추가하여 각 data point 의 학습 데이터에 포함 시킬지의 여부를 결정한다. 이때, vi 는 임계값과의 비교를 통해 0 또는 1 의 값을 갖는다. 전체 학습 과정은 두 단계로, v_i 를 최적화하는 과정과 θ 를 최적화하는 과정으로 이루어진다.

$$mimimize\ L(\theta) = \frac{1}{N}\left(\sum_{i=1}^{N} v_i(L_i(\theta) - \lambda)\right) + r(\theta)$$

$$mimimize\ L(\theta) = \frac{1}{N}\sum_{i=1}^{N} v_i L_i(\theta) + f(v;\lambda) + r(\theta)\ \ s.t. v_i \in \Psi$$

만약 데이터에 대한 사전지식이 확보됐다면, Curriculum learning[9]을 Self-pace learning[10]과 결합하여 Self-paced Curriculum learning[11]을 적용할 수 있다. 위의 수식이

SPCL 의 loss function 이다. 먼저 Self-paced function f 를 추가로 구현하여 v_i 가 binary 값이 아닌 0 과 1 사이 실수를 출력하게 한다. 기존 binary scheme 이외의 linear scheme, logarithmic scheme, mixture scheme 등으로 f 를 구현할 수 있다. 또한 사전지식을 반영하기 위해 v_i 를 Curriculum region Ψ 안에 속하도록 제한해준다. 사전 지식을 통해 data example 에 순위를 매겨주고, 이렇게 매긴 순위 γ를 통해 아래와 같이 curriculum region Ψ를 도출한다.

$$\Psi = \{v | \gamma^{\mathrm{T}} v \leq c\} \quad (c \; is \; a \; constant)$$

$$f(v;\lambda) = \sum_{i=1}^{N} \zeta v_i - \frac{\zeta^{v_i}}{log\zeta'} \qquad v_i^* = \begin{cases} \frac{1}{log\zeta} \log(L_i(\theta) + \zeta), & L_i(\theta) < \lambda \\ 0, & L_i(\theta) \geq \lambda \end{cases} \quad (\zeta = 1 - \lambda, 0 < \lambda < 1)$$

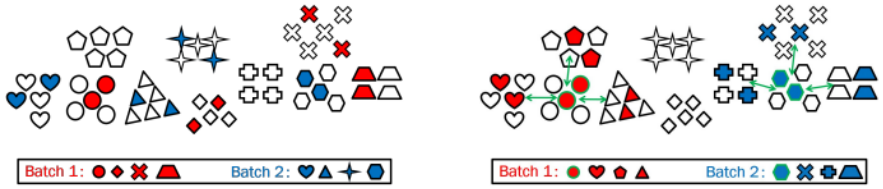
위 수식 중 f 는 logarithmic scheme 의 Self-paced function 이다. 이와 같은 f 를 포함한 L(θ)를 v_i 에 대해 편미분하여 optimal solution v_i^*를 찾을 수 있다. Loss 값 $L_i(\theta)$가 커질수록 해당 sample 에 더 작은 값의 가중치가 반영된다. 이와 같이 SPCL 에서는 사전지식을 바탕으로 한 학습 난이도와 각 sample 의 loss value 크기를 고려하며 학습을 진행한다.

최근 Curriculum learning 은 잘못 매겨진 label (Corrupted samples) 문제, Class imbalance, Reinforcement learning 등과 관련된 연구가 진행되고 있다. 또한 정제되지 않은 Real world dataset 에 인간의 학습 프로세스를 적용함으로써 모델 성능을 개선할 수 있을 것이다. 이번 Food CLIP 연구에서 역시 Curriculum learning 을 적용하여 모델의 성능을 향상시킬 수 있을 것이라 기대한다. NUVILAB 의 음식 자연어 사전의 카테고리 정보를 Curriculum learning 혹은 SPCL 의 사전지식으로 활용하여 학습 난이도에 따른 순차적 학습과 그에 따른 global minimum 에 가까운 해로의 수렴 효과를 기대해볼 수 있겠다.

3. 상세 제안 내용

3.1. Food tree based sampling을 통한 hard negative 선정

기존의 metric learning 방식은 2.2.4.2 절에서 설명한 PK sampler(P classes, K images)[8]를 바탕으로 mini-batch 를 구성하고, hard negative 를 선정한다. 그러나, P 개의 class 를 random 하게 선정하기 때문에 class 간 연관성이 고려되지 않으며, 모든 class 가 균일한 확률로 sampling 된다는 한계점이 존재한다. 이러한 한계점 때문에 informative 한 negative sample(anchor 와 비슷한 특성을 가져 구분이 어려운) 이 mini-batch 내에 포함되지 않을 경우, class 간 미세한 차이를 학습하지 못하게 된다.



(a) PK sampler (b) GS sampler

그림 13 PK sampler[8]와 GS sampler[12]의 mini-batch 구성 방식

이러한 random sampling 방식의 문제점을 해결하기 위해, 대표적인 metric learning 분야 중 하나인 person re-identification(ReID)에선 graph-sampling[12](GS sampler)이 제안되었다. GS ampler 의 기본 아이디어는 매 training epoch 마다 모델의 이전 학습 결과를 바탕으로 dataset 의 모든 class 간 nearest neighbor relationship(가장 유사한 class 끼리의 관계)를 graph 로 표현하고, 이를 mini-batch 구성에 반영하는 것이다. 그림 13 은 PK sampler[8]와 GS sampler[12]의 차이를 나타낸다. PK sampler[8]는 class 간의 관계를 고려하지 않기 때문에, cluster 간 거리와 상관없이 모든 class 에 대해 uniform 한 sampling 이 이루어진다. 반면, GS sampler 는 class 간 관계를 미리 계산한 후 embedding space 상에서 가장 가까운 cluster 로부터 negative sampling 을 하기 때문에 항상 fine-grained feature 학습에 효과적인 hard-negative sample 을 mini-batch 에 포함한다. GS sampler[12]는 PK sampler[8]의 한계점을 보완하여 3 개의 person Re-ID benchmark (CHUK03, Market1501, MSMT17)에 대해 rank 1 score 와 mAP 를 향상시켰다.

그림 14 Tree 구조의 음식 사전

Food CLIP 은 음식 사전을 기반으로 이미지와 text 간 matching 을 수행한다. 하나의 음식에서 다양한 음식이 파생될 수 있고 파생된 음식들이 비슷한 특성을 갖는다는 점을 고려할 때, 음식 class 간 관계는 계층적인 tree 구조로 표현이 가능하다. 그림 14 와 같은 tree 구조의 음식 사전을 통해 비슷한 특성의 음식(ex. 김치찜, 김치찌개, 김치 볶음밥)을 pre-define 한 후, mini-batch 구성 시 같은 계층의 음식을 sampling 하여 hard-negative pair 로 학습시킨다면 음식간 미세한 차이를 학습할 수 있을 것이다.

3.2. Feature mixing을 통한 synthetic hard negative 생성

MoCHi(**M**ixing **o**f **C**ontrastive **H**ard negat**i**ves)[13]는 embedding space 에서의 feature mixing 을 통해 hard negative sample 을 생성하는 방법이다. 아래의 그림 15 는 MoCHi[13]를 통해 생성한 synthetic negative sample 을 표현한 것이다. 생성된 negative sample 들은 기존의 negative 보다 query(anchor)에 더 가깝게 존재하여 informative 한 hard negative 로 작용한다.

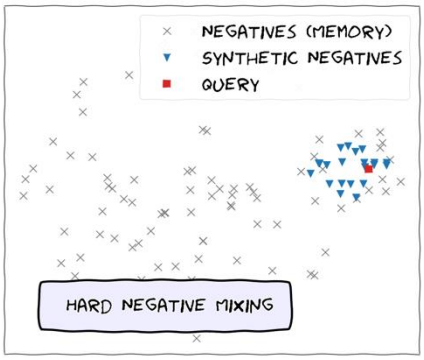


그림 15 MoCHi[13]를 통해 hard negative sample 을 생성한 결과

Mixing 은 feature 간 linear combination 을 기반으로 한 두 가지 방식을 사용한다. 첫 번째는 존재하는 negative sample 들 중 hardest negative sample 을 선정한 후 이들을 mixing 하는 것이다. 아래의 수식은 feature mixing 을 나타낸다. 이때, α_k 는 랜덤하게 정해지는 0 부터 1 사이의 값이며, n_i, n_j 는 hard negative sample 들 중 랜덤하게 선택되는 sample 이다.

$$h_k = \frac{\tilde{h}_k}{\|\tilde{h}_k\|_2}, where\ \tilde{h}_k = \alpha_k n_i + (1-\alpha_k) n_j$$

두 번째는 query(anchor)와 hardest negative sample 을 mixing 하는 것이다. Query 와 negative 간 linear combination 을 할 경우, 생성된 negative 는 embedding space 상에서 기존 보다 query 에 더 가깝게 위치하여 even harder negative sample 로 작용한다.

그림 16 은 MoCo-v2[14]에서 hard negative mixing 을 적용한 결과를 보여준다. 두 가지 방식의 mixing 모두 유의미한 성능 향상을 보여주었으며, 첫 번째 방식만을 사용했을 때(빨간색) 보다 두 가지 방식을 모두 사용하였을 때(보라색) 더 큰 성능 향상을 보여준다. 또, image classification, object detection, semantic segmentation 과 같은 downstream task 에서도 모두 유의미한 성능 향상을 확인하였다. 이는 synthetic negative sample 들이 useful representation 을 학습하는 데에 효과적임을 나타낸다.

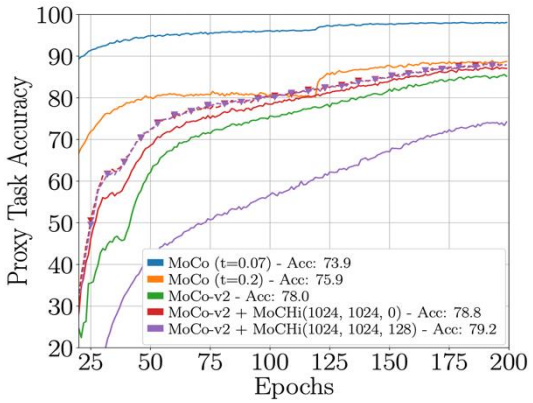


그림 16 Hard negative mixing 방식의 성능 비교

우리는 이러한 접근 방식을 토대로, 비슷한 visual feature 를 갖기 때문에 분리가 어려웠던 김치찜, 김치찌개와 같은 음식들에 대한 embedded feature 를 mixing 함으로써 기존보다 harder negative sample 들을 생성하여 Food CLIP 모델의 전체적인 성능 향상을 이루고자 한다.

4. 관련 경험

4.1. Mask Classification with hard negative sampling

4.1.1. 개요

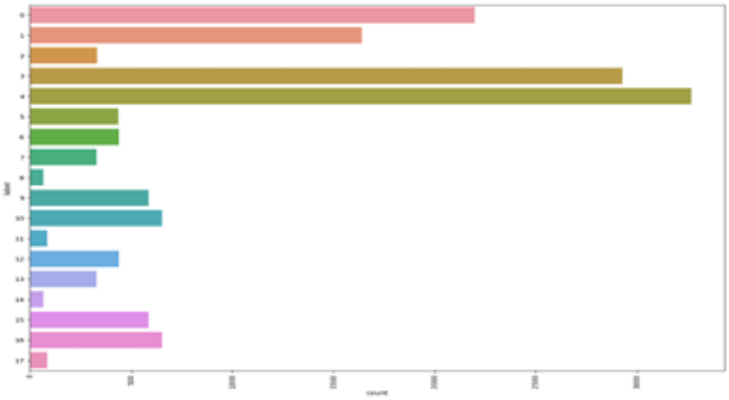


그림 17 Class imbalance in mask dataset

CV domain 에서 진행한 mask classification competition 은 인물의 얼굴 사진을 input 으로 받아 mask 착용 상태(wear, not wear, incorrect), 성별(male, female), 연령대(0~30, 30~59, 60~)를 classification 하는 것이 목적이다. Train set 은 총 18 개의 class 로 구성되는데, 그림 17 처럼 class imbalance 가 심한 상태이며 각 class 끼리 중복된 특성이 많다. (ex. 60 대 남성과 50 대 남성)

4.1.2. 문제 정의

그림 18 (좌) Wear, Female, 60 대 (우) Incorrect, Female, 60 대
: 두 image 는 유사한 특성을 갖지만 class 가 다르다.

그림 18 은 서로 다른 2 개의 class 에 대한 sample 을 각각 보여준다. 왼쪽은 마스크를 착용한 60 대 여성이며, 오른쪽은 마스크를 잘못 착용한(코까지 착용하지 않음) 60 대 여성이다. 두 이미지는 매우 비슷한 특성을 가지지만, 다른 class 로 labeling 되어 있기 때문에 모델이 이 두 class 를 예측하는 데에 오류가 발생하는 것을 파악하였다. 또한, class imbalance 에 의한 학습 불균형도 발생하여 유사한 class 간 미세한 차이를 학습하는데 어려움을 겪었다.

4.1.3. Triplet loss with custom mini batch sampling [Appendix: Code]

앞서 언급한 두 문제를 해결하기 위해선 이미지의 feature 를 비교하는 metric learning 이 필요하다고 생각하였고, triplet loss 를 적용하여 학습을 진행하였다. 특히, triplet loss 적용시 4.2.2 에서 언급한 두 가지 문제점을 고려하여 custom batch sampling 을 하였다.

첫번째로, 유사한 특성을 가지는 class 들을 pre-define 한 후, mini-batch 를 구성하는 단계에서 random sampling 이 아닌 conditional sampling 을 하여 class 간 미세한 차이를 학습하도록 유도하였다. 예를 들어, 그림 18 의 두 image 의 class index 는 각각 13,14 이며 mini batch 구성 단계에서 이 두 class 는 함께 sampling 되도록 설정하였다. 이렇게 sampling 할 경우, triplet 선정 단계에서 두 sample 이 서로 hard negative 로 작용하도록 하였다.

두번째로, data imbalance 를 해결하기 위해 oversampling 과 undersampling 기능을 구현하였다. Random sampling 을 할 경우 mini batch 의 구성은 전체 dataset 의 class distribution 을 따라가게 되므로, 균형 있는 학습이 어려워진다. 따라서 그림 17 의 class 분포를 확인하여, oversampling 과 undersampling 을 할 class 를 pre-define 한 후 이들이 mini-batch 구성 단계에서 sampling 될 확률을 보정하였다.

위 두 고려 사항이 적용된 mini-batch sampling 을 통해 모델을 학습 하였을 때, random sampling 을 적용했을 때 보다 4%의 f1 score 가 향상되었다. 그림 19 는 custom sampling 을 적용했을 때 각 class 에 대한 validation accuracy 를 random sampling 과 비교한 것이다. 실험 결과 거의 모든 class 에 대해 accuracy 가 향상되었는데, 다음 두 가지 관점에서 방법의 유효성을 확인할 수 있었다.

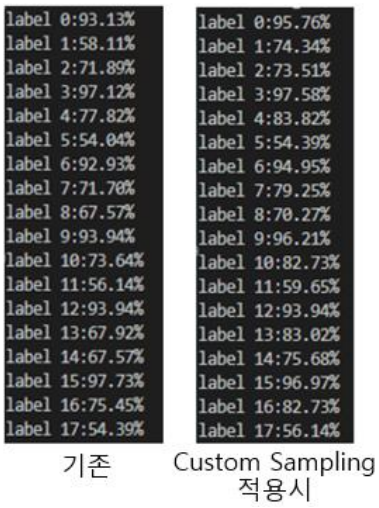


그림 19 Validation Accuracy 비교

- 비슷한 특성을 가지는 class pair 에 대한 분별력 증가 [Label: (1,2), (4,5), (7,8), (10,11), (13,14), (16,17)]
- Data 수가 적은 class 에 대한 성능 향상 [Label: 2, 8 ,11 ,14,17]

위 경험을 통해 metric learning 에서 각 class 특성에 대한 파악과 이에 기반한 mini-batch sampling 의 중요성을 알 수 있었으며 본 프로젝트 에서도 비슷한 특성의 음식을 분류하는데 적용할 수 있을 것이다.(Appendix 에 구현한 코드를 첨부)

4.2. Person re-identification과 Embedding space 분석

4.2.1. 개요

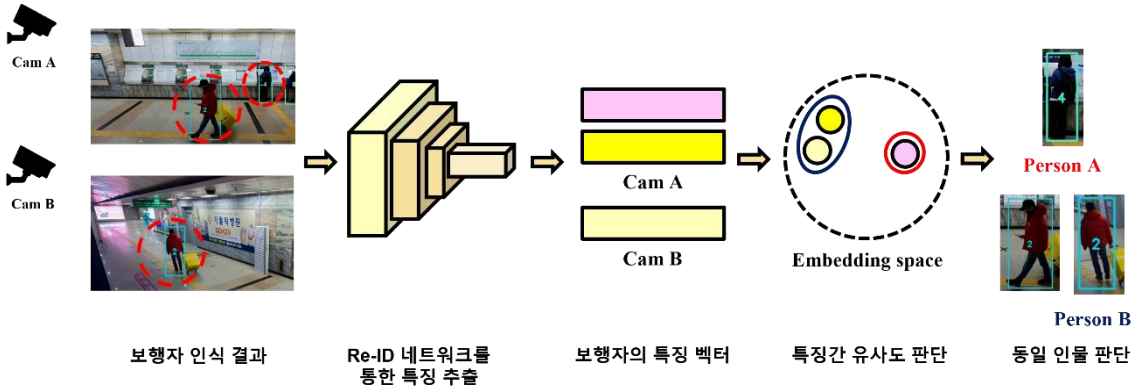


그림 20 Person re-identification 의 모식도

Person re-identification(이하 ReID)란 지능형 다중 CCTV 시스템에서 하나의 카메라에 국한되지 않고 여러 카메라를 넘나 들며 등장하는 동일인물을 식별하는 기술이다. 다양한 카메라 환경 속 동일 인물을 식별하는 것은 여러 현실적인 제약 (인물 간 occlusion, 촬영

각도/방향에 따른 variation)이 따르기 때문에, ReID 에선 보행자의 feature 간 세밀한 차이를 학습하는 것이 중요하고 전통적으로 metric learning 을 적용하는 분야로 알려져 있다.

4.2.2. ReID metric learning 분석 및 toy example 적용

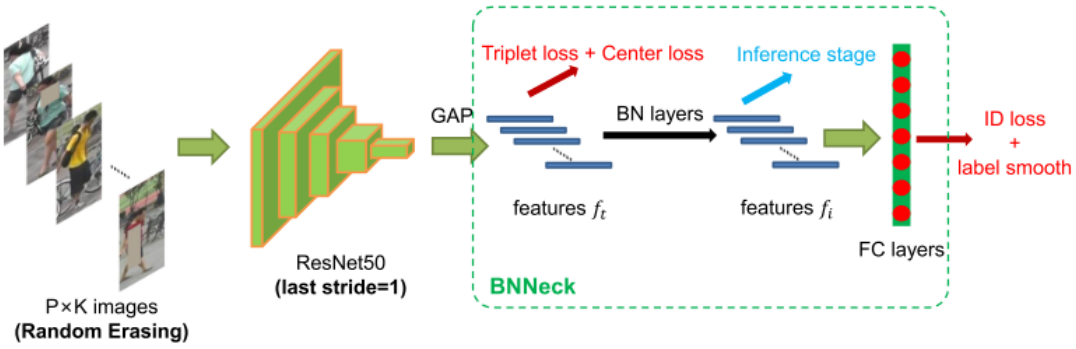


그림 21 ReID training pipeline

기존 ReID 연구들은 모델의 구조보단 모델의 학습 방법에 집중하였으며, 특히 cross-entropy loss 와 triplet loss 함수의 설계에 초점을 맞추어 왔다. 예시로, 그림 21 은 H.Luo 의 논문[15]에서 제시한 ResNet50 기반의 training pipeline 이다. 논문을 참고하여, metric learning 에 사용되는 각 loss 함수들이 embedding space 에서 어떤 영향을 주는지 알아보기 위해 MNIST 와 custom network 로 실험해보았다.

① Custom convolutional neural network

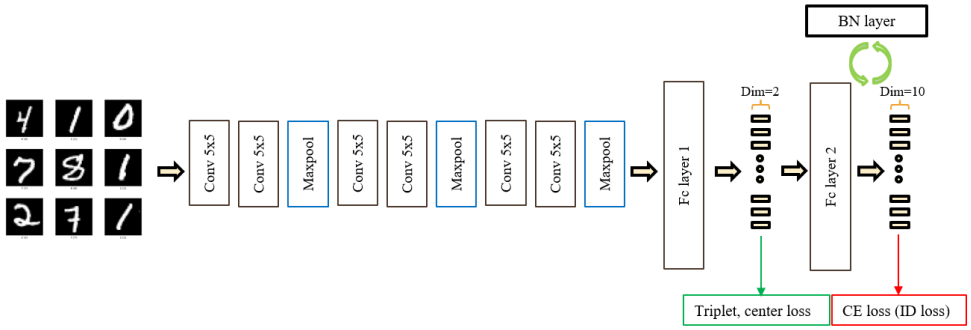


그림 22 실험을 위해 설계한 custom convolutional neural network

그림 22 는 custom network 의 학습 구조를 나타내며, loss 를 적용하는 대상과 순서는 그림 21 을 참고하였다. MNIST 를 Custom CNN 에 통과시켜 feature vector 를 생성한 후 이를 fully connected layer 에 통과시켜 dimension 을 2 로 축소한 뒤, visualize 하여 clustering 양상을 관찰하였다.

② Experimental results

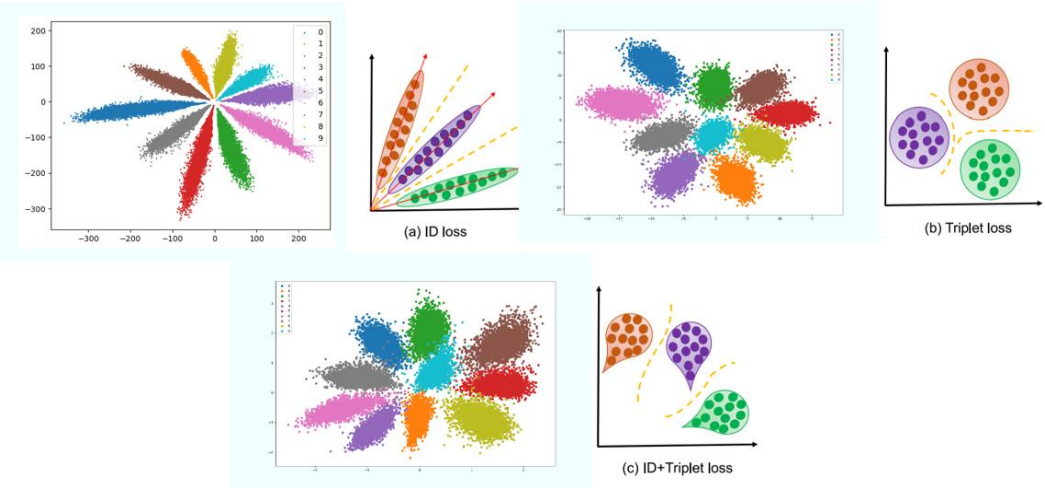


그림 23 CE loss 와 triplet loss 의 조합에 따른 embedding space 변화

그림 23 는 cross-entropy loss 와 triplet loss 가 학습 embedding space 상에 미치는 영향력을 case 별로 시각화 한 것이다. Cross entropy loss 만으로 모델을 학습했을 땐 class 간 angular distance 가 멀어지도록 cluster 가 형성된다. 반면, triplet loss 의 경우 같은 class 의 모든 sample 에 대해 비슷한 위치를 가지도록 cluster 가 형성되며 intra-class variation 이 cross entropy loss 보다 작다.

위와 같은 toy example 을 이용한 실험을 통해, Re-ID 분야에서 사용하는 metric learning 기법이 어떤 학습 효과를 갖는지 직관적으로 파악할 수 있었다. 특히, embedding space 상에서 loss 함수의 영향력을 직접 체험해 본 경험은 본 프로젝트의 구체적인 모델 설계 과정에서 도움이 될 것이다.

5. 개발 일정 및 역할 분담

단계	내용	일정(주차)											
		Pre-week		1/9-1/15		1/16-1/22		1/23-1/29		1/30-2/5		2/6-2/9	
프로젝트 준비	· 프로젝트 범위/환경 협의												
	· 관련 기술/논문 조사												
	· 코드 공통 템플릿 구성												
프로젝트 수행	· 모델 분석 및 선정												
	· EDA												
	· 프로토타입 제작												
	· 모델링 및 제안 기법 적용												
	· 학습 및 성능 개선												
마무리	· 발표자료 제작												
	· 1차 발표영상 제작 (~2/6)												
	· 최종 발표 영상 제작 (~2/9)												

표 1. 세부 일정 계획표

김기용	CLIP, Contrastive learning, Hard negative sampling
김성수(리더)	팀 경험 정리, 논문 조사 및 idea 제안
김주엽	Food CLIP , Self-supervised learning 및 MAE[5]
이 구	N-shot learning, 논문 조사 및 idea 제안
이태희	Contrastive learning, Curriculum learning, 문서 관리 및 통합

표 2. 제안서 작성 역할 분담

6. 기대 성과

Food CLIP의 성능 개선을 통해 최근 많이 사용되는 self-supervised contrastive learning 및 sampling 이론을 깊게 탐구하고 음식 분류 모델의 발전에 기여하고자 합니다.

7. Reference

Papers

[1]A. Radford et al., "Learning transferable visual models from natural language supervision," in International Conference on Machine Learning, 2021: PMLR, pp. 8748-8763.
[2]A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," arXiv preprint arXiv:2204.06125, 2022.
[3]C. Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," arXiv preprint arXiv:2205.11487, 2022.
[4]M. Caron et al., "Emerging properties in self-supervised vision transformers," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9650-9660.
[5]K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 16000-16009.
[6]A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.
[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
[8] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," arXiv preprint arXiv:1703.07737, 2017.
[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in Proceedings of the 26th annual international conference on machine learning, 2009, pp. 41-48.
[10]M. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," Advances in neural information processing systems, vol. 23, 2010.
[11]L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann, "Self-paced curriculum learning," in Twenty-ninth AAAI conference on artificial intelligence, 2015.
[12]S. Liao and L. Shao, "Graph sampling based deep metric learning for generalizable person re-identification," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 7359-7368.
[13]Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, "Hard negative mixing for contrastive learning," Advances in Neural Information Processing Systems, vol. 33, pp. 21798-21809, 2020.
[14]X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," arXiv preprint arXiv:2003.04297, 2020.
[15]H. Luo et al., "A strong baseline and batch normalization neck for deep person re-identification," IEEE Transactions on Multimedia, vol. 22, no. 10, pp. 2597-2609, 2019.

Articles

[16]Self-Supervised Learning(자기지도 학습 설명).
https://greeksharifa.github.io/self-supervised%20learning/2020/11/01/Self-Supervised-Learning/#self-supervised-learning.
[17]Extending Contrastive Learning to the Supervised Setting.
https://ai.googleblog.com/2021/06/extending-contrastive-learning-to.html.
[18]Contrastive Learning 이란. https://daebaq27.tistory.com/97.
[19]Contrastive Learning 이란? (Feat. Contrastive loss). https://89douner.tistory.com/334.
[20]Contrastive Representation Learning.
https://lilianweng.github.io/posts/2021-05-31-contrastive/.

[21]Understanding Few-Shot Learning in Computer Vision: What You Need to Know.
https://neptune.ai/blog/understanding-few-shot-learning-in-computer-vision.
[22][DMQA Open Seminar] Zero-shot learning.
https://www.youtube.com/watch?v=7uAszeiLE2w.
[23]The Essential Guide to Zero-Shot Learing [2022].
https://www.v7labs.com/blog/zero-shot-learning-guide.
[24]거대 AI 모델의 발전과 zero-shot 의 의미.
http://cloudinsight.net/ai/%EA%B1%B0%EB%8C%80-%EB%AA%A8%EB%8D%B8%EC%9D%98-%EB%B0%9C%EC%A0%84%EA%B3%BC-zero-shot%EC%9D%98-%EC%9D%98%EB%AF%B8/.

8. APPENDIX: Code

8.1. Mask classification 에서 직접 작성한 batch sampler

```python
class RandomIdentitySampler(Sampler):  # For Single GPU training
    """
    Randomly sample N identities, then for each identity,
    randomly sample K instances, therefore batch size is N*K.
    Args:
    - data_source (list): list of (img_path, pid).
    """
    def __init__(self, data_source,cfg):
        # self.data_source = data_source
        self.data_source = sorted(data_source,key=lambda x : x[1])

        self.batch_size = cfg.batch_size
        self.num_instances = cfg.num_instance
        self.num_pids_per_batch = self.batch_size // self.num_instances
        self.index_dic = defaultdict(list) # dict with list value

        self.undersample_id = cfg.undersample_id # undersampling할 class의 번호
        self.oversample_id = cfg.oversample_id # oversampling할 class의 번호
        self.undersample_rate = cfg.undersample_rate # undersampling 정도
        self.oversample_rate = cfg.oversample_rate # oversampling 정도


        for index, (_,pid) in enumerate(self.data_source):
            self.index_dic[pid].append(index) # pids : image index 형태의 dictionary로 저장

        self.id_cnt = list(map(lambda x : len(x[1]),self.index_dic.items()))
        self.pids = list(self.index_dic.keys())  # pids list

        # estimate number of examples in an epoch
        self.length = 0
        for pid in self.pids: # pid 들에 대해
            idxs = self.index_dic[pid] # pid에 해당하는 image index들
            num = len(idxs)
            if num < self.num_instances: # 만약 해당 id의 image 개수가 num_instances보다 작으면 전부 batch에 포함
                num = self.num_instances
            self.length += num - num % self.num_instances  # number of examples
```

```python
def __iter__(self):
    batch_idxs_dict = defaultdict(list)

    for pid in self.pids: # pid들에 대해
        idxs = copy.deepcopy(self.index_dic[pid]) # pid에 해당하는 image index들
        if len(idxs) < self.num_instances: # 만약 해당 id의 image 개수가 num_instances보다 작으면
            idxs = np.random.choice(idxs, size=self.num_instances, replace=True) # 중복을 허용하여 random choose한 idx list 반환
        random.shuffle(idxs)
        batch_idxs = []
        # 현재 pid에 대해
        for idx in idxs: # random shuffle 된 idx에 대해
            batch_idxs.append(idx) # batch_idxs list에 idx를 추가
            if len(batch_idxs) == self.num_instances: # batch에 pid에 해당하는 image를 num_instances만큼 추가했다면,
                batch_idxs_dict[pid].append(batch_idxs) # batch_idxs_dict에 pid : batch 형태로 추가
                batch_idxs = [] # batch 초기화
                # 예를 들면, 0번 pid에 대해 {0 :[[26,40,41,34],[21,6,10,44]...], }

    avai_pids = copy.deepcopy(self.pids) # pids

    final_idxs = []
    # while len(avai_pids) >= self.num_pids_per_batch + len(self.oversample_id): # self.num_instances = 4, self.num_pids_per_batch = 16

    while len(avai_pids) >= self.num_pids_per_batch + len(self.oversample_id): # self.num_instances = 4, self.num_pids_per_batch = 16
        selected_pids = random.sample(avai_pids, self.num_pids_per_batch)

        for pid in selected_pids: # selected_pids : random으로 뽑힌 self.num_pids_per_batch만큼의 pid
            if pid in self.oversample_id:
                temp = batch_idxs_dict[pid].pop(0)
                batch_idxs_dict[pid] += [temp]
                batch_idxs = temp
            elif pid in self.undersample_id:
                for _ in range(self.undersample_rate):
                    if batch_idxs_dict[pid]:
                        batch_idxs = batch_idxs_dict[pid].pop(0)
            else :
                batch_idxs = batch_idxs_dict[pid].pop(0) # self.num_instances개씩 뽑아놓은 batch들중 맨 앞을 pop
            final_idxs.extend(batch_idxs) # final_idxs에 batch_idxs elements들을 추가
            if len(batch_idxs_dict[pid]) == 0: # 해당 pid에 대해 더이상 추가할 sample들이 없으면
                avai_pids.remove(pid) # id list에서 pid를 제거

    return iter(final_idxs)

def __len__(self):
    return self.length
```