Learning Transferable Visual Models From Natural Language Supervision

Summary:

clip 모델 이전 vision은 predetermined object categories에서 예측하도록 훈련되어 해당 형태는 새로운 이미지가 들어왔을 때 분류하기 위해서는 새로운 labeling과 데이터가 필요하므로 유연하지 못함.

이에 해당 논문은 image에 대해 raw text를 직접학습하는 supervision 개념에 대해 소개하며 그런 형태로 학습된 모델의 장점을 기술함.

Abstract:

해당 논문은 크롤링해서 수집된 4억개 이미지, 텍스트 쌍을 사용하여 어떤 caption이 어떤 image와 연견되는지 예측하는 방식의 pretraining이 이미지 표현을 학습하는 데 효율적인 방법으로 언급.

pretraining 후 자연어를 사용해 학습된 시각적 개념을 참조하여 downstream task에 zero-shot 방법 사용.

30개 영역 computer vision dataset에서 resnet50과 비교하며 이때 zero-shot 방식을 사용

Approach:

1. natural language supervision
    a. clip 모델의 핵심은 natural language supervision을 이용해 이미지에 대한 인식을 학습하는 것.
    b. 자연어로 학습하는 것은 아래와 같은 장점 존재.
        i. natural language supervision은 데이터를 늘리기 편하며 인터넷 상에 존재하는 막대한 양의 텍스트 안에 존재하는 supervision을 가져와 학습하면 됨
        ii. 이미지에 대한 representation만 학습하지 않고 언어 데한 representation을 연결지어 zero-shot transfer도 가능하게 만듦.
2. creating a sufficiently large dataset
    a. clip 논문에서 언급된 핵심은 natural language supervision이며 해당 natural language supervision은 인터넷 상에 공개적으로 사용가능한(크롤링) 이미지, 텍

스트를 4억개 가까이 수집하여 새로운 데이터셋 구축.

b. 해당 데이터셋은 뭐리당 최대 2만개(이미지, 텍스트)를 포함하게끔 균형을 맞췄으며 해당 데이터셋을 WebimageText(WIT)로 부름

3. selecting an efficient pre-training method

a. 기존 SOTA cv 시스템은 연산량이 극도로 많음. ResNeXt101-32x48d의 경우 19개 GPU로 몇년이 걸리고 Noisy student EfficientNet-L2의 경우 33개 TPUv3로 몇 년이 걸림.

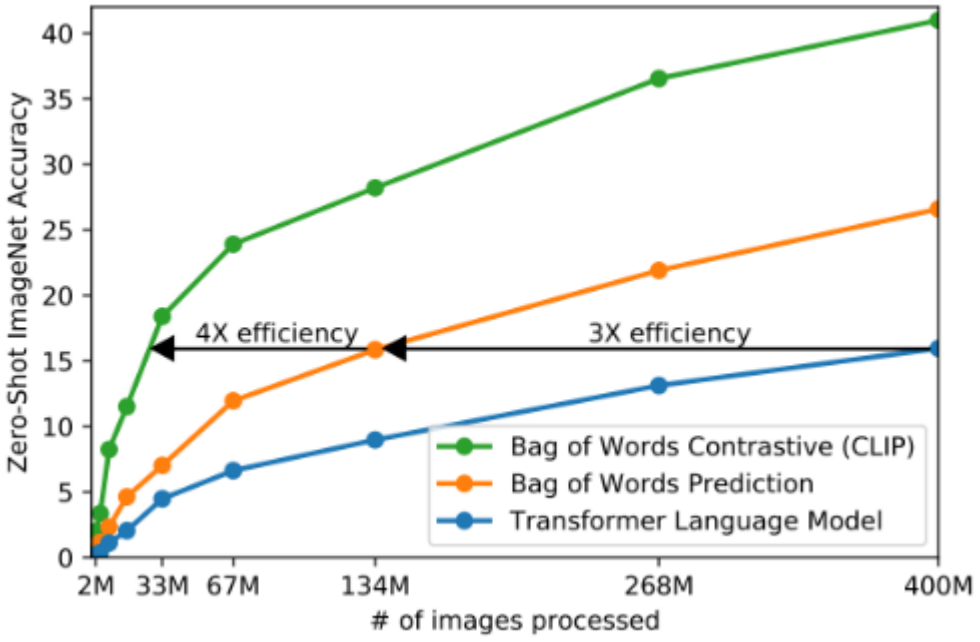
b. 이에 해당 논문에서 4억개의 데이터를 학습시키기 위한 방법은 아래와 같음.

i. virTEX처럼 이미지에 대한 캡션을 예측하기 위해 CNN과 text transformer를 한 번에 학습 (아래 그림의 파란선)

ii. 캡션의 정확한 단어들을 배열하는 예측보다 bag-of-words를 대상으로 예측하는 게 더 쉽게 바꿈(아래 그림의 주황선)

iii. prediction objective를 contrastive objective로 바꿈(초록색 선)

iv. 정확한 text를 예측하지 않고 bag-of-words로 예측하는 것과 contrastive objective로 학습함으로써 최종적으로 ImageNet으로 zero-shot transfer를 해볼 때 단순 transformer language model보다 12배 가까이 연산 효율 증가



c. CLIP 사전학습 방법은 아래와 같음

i. 입력으로 N개의 (이미지, 텍스트)를 batch 단위로 넣어준다.

ii. N개의 이미지와 텍스트가 각각 image encoder, text encoder를 통과해 features를 뽑아냄

iii. 두 features의 cosine similarity를 구함

iv. cosine similarity를 logit으로 보며 이 logit의 범위를 제어하기 위한 temperature parameter τ를 추가한다. 이때 τ는 hyperparameter가 아닌 학습가능한 값.

v. Cosine similarity를 logit으로보고 CE loss를 구함

vi. Cosine similarity의 대각 성분은 최대로 그 외의 성분들은 최소로 가는 방향으로 학습(contrastive learning)

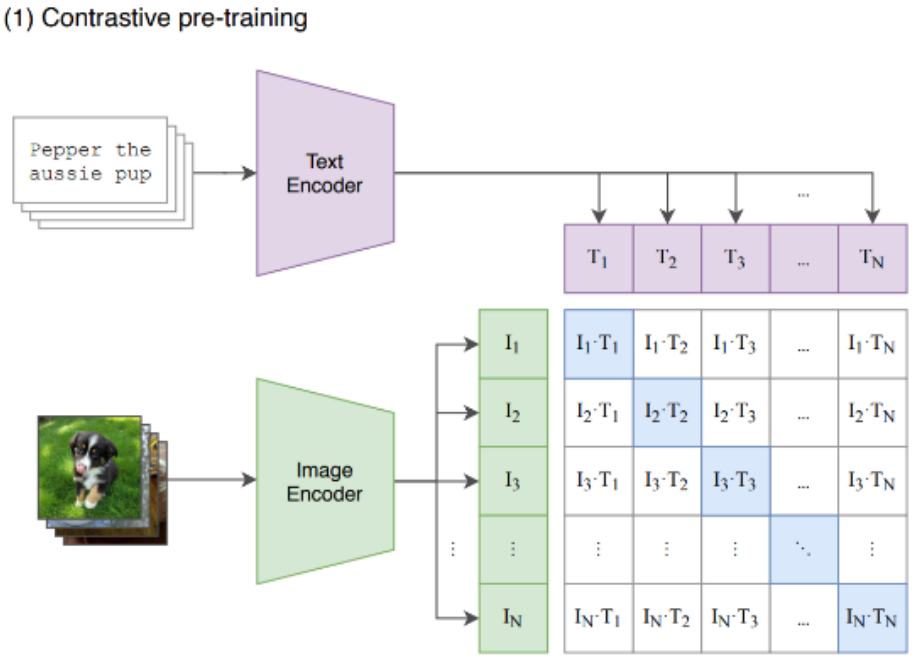


Figure 2. Pre-training method for CLIP

```
# OpenAI/CLIP 공식 코드 간략화

# image_input.shape = [batch_size, channels, height, width]
# text_input.shape = [batch_size, context_length]

# Image_encoder와 Text_encoder를 이용해 features 생성
image_features = image_encoder(image_input) # [batch_size, embeddings_dim]
text_features = text_encoder(text_input) # [batch_size, embeddings_dim]

# Cosine similarity 생성
# 각 features들을 미리 L2_norm으로 나눠준다.
image_features /= image_feautres.norm(dim=-1, keepdim=True)
text_features /= text_feautres.norm(dim=-1, keepdim=True)
```

```
# Dot product
similarity = image_featrues @ text_features.t() # [batch_size, batch_size]

# Scaling by temperature parameter
similarity = similarity * np.exp(t)
```

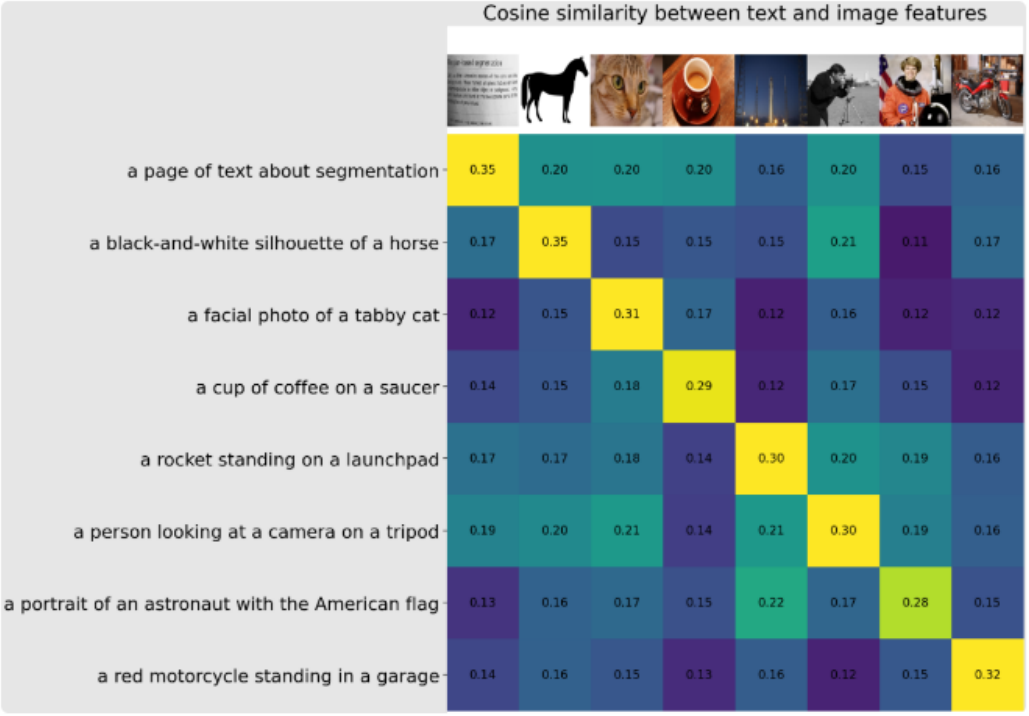


Figure 3. Example of cosine similarity between text and image features

d. choosing and scaling a model

   i. 해당 모델은 encoder에 image encoder, text encoder 존재.

   ii. image encoder의 경우 두가지가 있음

      1. `ResNet-50 base`로 ResNetD에서 사용된 기법과 rect-2 blur pooling을 적용했고, global average pooling layer를 attention pooling layer로 변경했다. 여기서 attention pooling은 transformer에서 사용되는 MSA를 이용하되 이 때 Query로 들어가는 값은 global average pooling의 결과이다.

      2. `vision transformer(ViT) base` 로 기존 ViT는 그대로 이용하지만 패치와 position embeddings를 더하는 부분에서 LayerNorm을 추가한 것과 약간 다른 initialization scheme을 이용.

   iii. text encoder는 transformer 사용. $L = 12,\ d_{model} = 512,\ h = 8$로 설정해 63M-parameter를 기본 사이즈로 사용. 연산효율을 고려해 스퀀스의 최대 길이를 76으로 설정.

   iv. scaling a model 설명

기존 CV 연구가 모델의 width, depth를 늘렸듯이 ResNet 기반의 image encoder도 width, depth, resolution을 늘려준다. 다른 연구들을 참고해 각각 하나만 올려주지 않고 골고루 동등한 비율로 늘려주는 옵션을 제공했다.

Text encoder의 경우 ResNet의 width가 늘어난만큼 text encoder의 width도 늘려주었다. 대신 다른 depth같은 것은 그대로 두었다. CLIP의 성능이 text encoder의 전체적인 부피에 딱히 민감하게 반응하지 않음을 확인했다.

실제 코드에서 모델을 불러올 때, RN50, RN50x4, RN50x16, RN50x64 로 불러올 수 있는데 이 표기가 모델의 scaling 정도를 의미한다.

Envioronment:

- AdamW optimizer 이용. Gain과 bias를 제외한 모든 weights에 적용.
- Cosine annealing learning rate scheduler 사용.
- ResNet-50을 baseline으로 둔 모델은 1 epoch 기준으로 hyperparameter tuning.
- 더 큰 모델들은 연산량이 너무 많아 heuristic하게 hyperparameter tuning.
- Temperature parameter인 τ는 0.07에서 시작하고, 이 τ에 의해 logits가 100배 이상 커지는 것은 학습 안정성을 위해 방지.
- Batch_size는 32768로 사용. (어마어마하다..)
- Mixed-precision, gradient checkpointing, half-precision Adam statistics, half-precision stochastically rounded text encoder weights 이용해 메모리 활용 극대화.
- Similarity를 구할 때 인접한 batch들에 한해서 GPU들끼리 연산을 분할해 진행.

→ 가장 큰 RN50x64 모델은 V100 GPU 592개로 18일동안 학습하고, 가장 큰 ViT는 V100 256개로 12일이 걸렸다. ViT-L/14의 경우 FixRes와 비슷하게 더 높은 해상도인 336 사이즈에 1번의 epoch를 추가 학습해줬다. 이걸 ViT-L/14@336px로 명명하고, 따로 언급되지 않은 CLIP은 모두 ViT-L/14@336px 버전의 CLIP을 의미한다. 이게 가장 성능이 좋다고 한다.

Experiments:

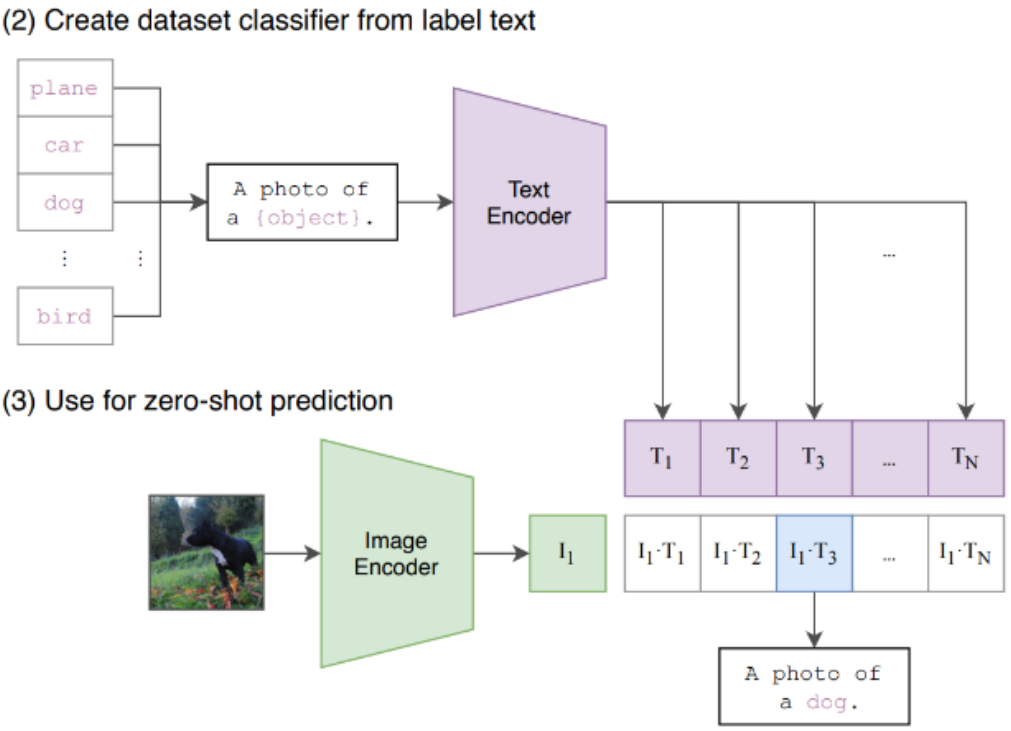
1. Zero-shot Transfer
    a. motivation
        i. CV에서 주로 사용되는 CIFAR-10, ImageNet 같은 데이터셋은 학습모델이 더 뛰어난 representation learning을 수행하는가를 확인하고자 만들어진 뉘앙스

라면 본 논문의 연구자들은 조금 더 데이터셋 자체가 어떤 task에 특화되어있는지에 집중하고자 했음. 이것은 `task learning` 관점이라 함

ii. 예시로 길가에서 바라본 도로번호가 데이터인 SVHN 데이터셋의 경우 해당 데이터셋에서 성능이 잘 나오면 도로번호를 해석하는 task를 잘 수행하는 것을 알 수있음.

iii. CLIP의 zero-shot transfer 성능 평가를 할 때에는 사용되는 데이터셋의 특성에 따라 domain generalization이 뛰어난지 task generalization이 뛰어난지를 다르게 해석할 수 있다. 이러한 관점으로 논문에서 다양한 데이터셋, task에 대한 clip의 zero-shot transfer 성능은 알아보는 실험을 진행함.

b. Using CLIP for Zero-Shot Transfer



1. CLIP을 zero-shot transfer할 때는 사전학습 방식과는 다름. 이미지가 주어진 데이터셋에서 뽑은 임의의 이미지 한 장이면 텍스트는 데이터셋에서 등장하는 모든 텍스트의 집합. 이 때 텍스트는 그대로 사용하지 않고 아래그림처럼 `A Photo of a {object}` 식으로 디폴트값에 더해준 다음 사용. 이렇게 적용하면 아래 그림처럼 이미지 feature는 1개, 텍스트의 feature는 클래스의 개수인 N개 만큼 나옴. 이후는 다음과 같음.

   1) 그림처럼 이미지와 텍스트 features를 encoder를 통해 뽑아냄 ($I_1, T_{(1,2,..,N)}$)

2) 둘의 cosine similarity를 구함
3) Cosine similarity를 logits로 보고 softmax 적용
4) 가장 확률이 높은 pair로 출력 결정
5) Zero-shot evaluation 동안 teat_features는 항상 동일하므로 계속 동일한 $T_{(1,2,\ldots,N)}$를 이용해 연산비용 낭비를 막음.

c. prompt engineering and ensembling

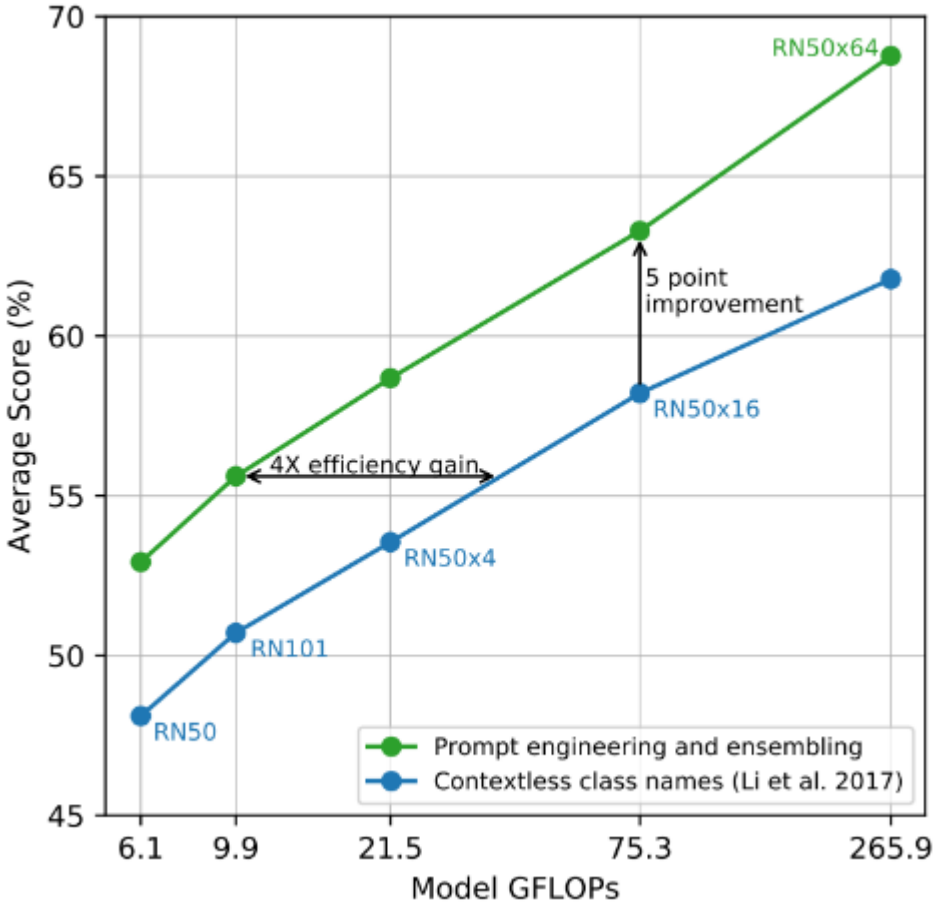
i. b에서 언급한 것처럼 클래스를 그대로 이용하지 않고 임의의 디폴트값에 더해준 다음 그걸 텍스트로 사용. 이걸 `prompt engineering` 이라 부름.

ii. 클래스를 바로 사용하지 않는 이유는 아래와 같음.

1. **클래스 중 동음이의어가 존재하는 경우 발생.** classification 용 데이터셋의 경우 이미지를 표현하는 언어적 정보는 겨우 클래스 하나이며 이것을 가지고 이미지를 구분하기는 불가능함.

2. **사전 학습된 사용된 데이터셋은 한 단어로만 이미지를 표현하는 경우가 드물다.**
대부분 이미지를 설명하는 텍스트가 문장의 형태이며 단어 하나로만 표현되는 경우가 매우 드묾. 이런 차이를 줄이기 위해 위의 경우처럼 `A Photo of a {object}` 식으로 변경하며 이런 prompt를 사용하게 되면 ImageNet의 경우 1.3%의 정확도가 향상되었음.

iii. 추가적으로 각 task마다 prompt를 알맞게 변형함으로써 zero-shot 성능을 확연히 늘릴 수 있음. 예를 들어 애완동물과 관련된 데이터셋의 경우 `A Photo of a {object}, a type of pet` , 위성사진 데이터셋의 경우 `A setellite photo of a {object}` 설정하면 성능이 올라갔으며 거기에 더해 단일 prompt만 이용하지 않고 더 다양한 prompt 설정하면 그 성능이 더 좋아짐.
ImageNet의 경우 80개의 context prompts를 사용할 경우 단일 prompt보다 3.5%만큼 성능이 향상되어서 전체적으로 보았을 때 prompt engineering and ensembling은 약 5%만큼의 정확도를 향상시켰다고 함.


70
65
60
55
50
45
Average Score (%)
6.1
9.9
21.5
75.3
265.9
Model GFLOPs
RN50x64
5 point
improvement
RN50x16
4X efficiency gain
RN50x4
RN101
RN50
Prompt engineering and ensembling
Contextless class names (Li et al. 2017)