

Learning Transferable Visual Models From Natural Language Supervision

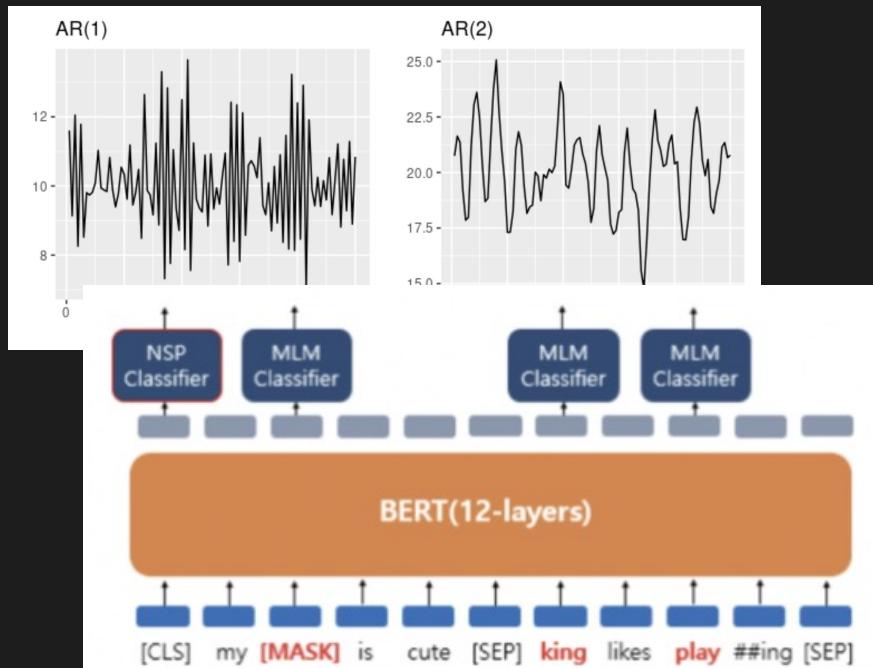
이윤영

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

introduction

In NLP...



Auto Regression model(ML), Masked Language Model

- > task agnostic : 작업에 구애받지 않는
- > 데이터셋 특유의 맞춤화가 필요 없음(by, zero-shot)
- > 특정 task 위한 특정한 데이터셋이 필요 없게 됨



GPT-3의 등장...

introduction

In NLP...

- web에서 가져온 dataset들이 고품질의 crowd-labeled NLP dataset들을 능가함
-> 그렇다면 Computer Vision 에서는??

introduction

In CV...

- ImageNet과 같은 crowd-labeled dataset으로 pre-train 하는 것이 표준 관행
- 여러가지 노력들이 있었음
 - 이러한 노력에도 불구하고 natural language supervision을 통한 image representation 활용은 드물었음 -> 정확도가 많이 낮았기 때문

introduction

In CV...

- 대신 weak supervision에서는 좋은 성능을 보여줬음!
(Instagram에서 ImageNet-related hash tag 예측)
 - weak supervision : low scale, well-targeting
 - 이 논문에서 평가하기를…
 - ‘제한된 양의 잘-라벨링된 데이터’와 ‘웹 상의 raw한 원시 데이터’ 사이의 현실적이고 실용적인 중간지대를 잘 찾아냄
 - 너희 실험 신중하게 잘 하긴 했어~근데~
 - static softmax classifiers, lack a mechanism for dynamic outputs
- > 유연성을 제한하고 ‘zero-shot’ 기능을 제한함

introduction

What is 'zero-shot'?

- model이 특정 task에 대해 training 없이 그 작업을 수행하는 것
- (나중에 나오겠지만) ‘a photo of cat’에 대한 특정 task를 학습시킨 적이 없지만 새로운 고양이 사진이 주어져도 ‘a photo of cat’과 대응할 수 있음

Zero-shot의 장점

- 유연성 : 새로운 task 또는 새로운 dataset에 대해 pre-training 없이 높은 성능 발휘 가능
- 효율성 : data labeling에 대한 시간과 비용을 절감할 수 있음
- 다양성 : 한 번 학습된 모델로 다양한 작업에 쉽게 적용할 수 있음

introduction

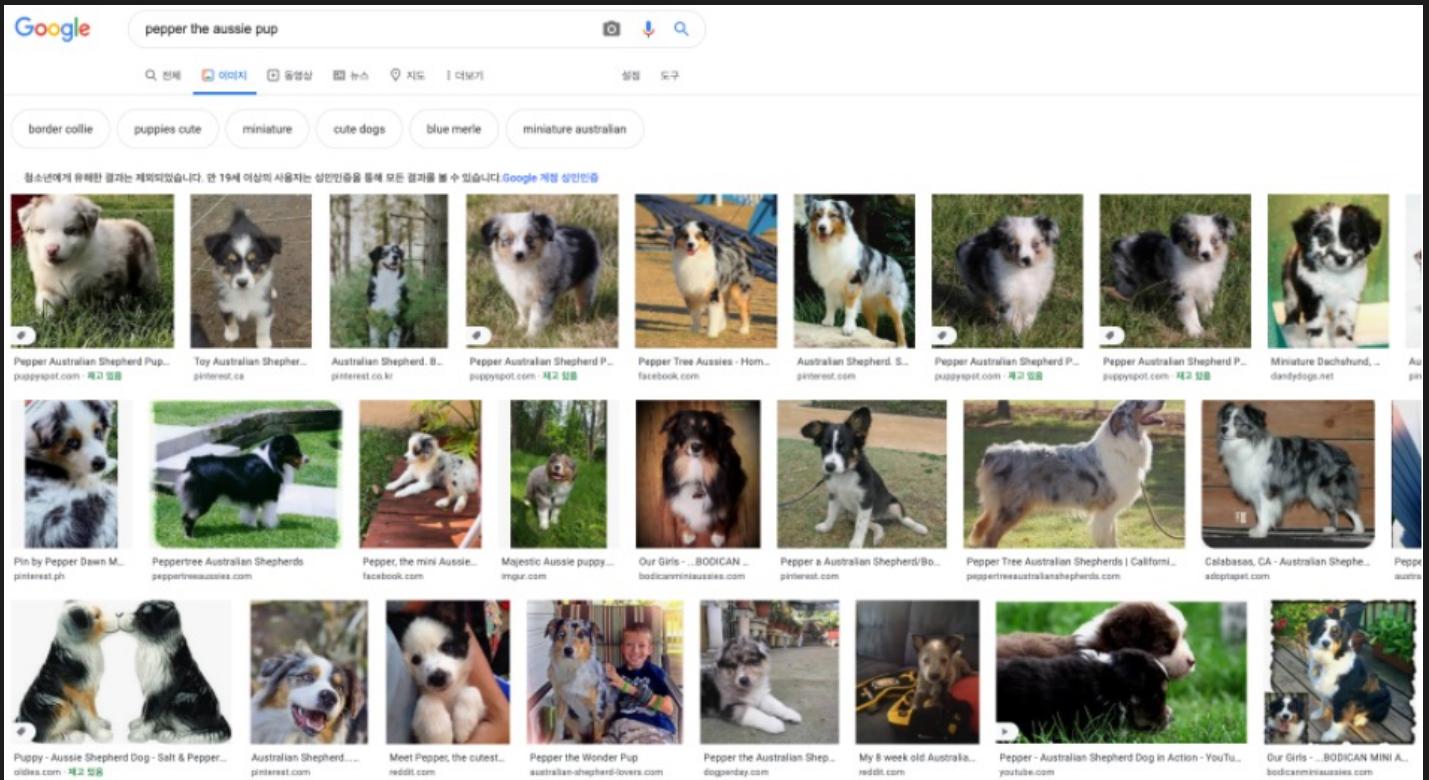
Weakly supervised model vs recent exploration

- 차이는 당연히 규모임(10만-20만 개의 data vs 수백만-수십억 개의 data)

introduction

그래서 이 논문에서의 dataset은…

- 인터넷의 4억개의 (이미지, 텍스트)쌍의 새로운 dataset 만듦.
 - ConVIRT 간소화 버전으로 training



introduction

CLIP
(Contrastive Language-Image Pre-training)

introduction

그래서 CLIP은..

- GPT계열과 유사하게 훈련 도중 OCR, geo-local, action recognition등 다양한 task를 학습한 것을 확인할 수 있었음
- 심지어 지도학습된 이전의 model들과 경쟁 가능한 것을 알 수 있었음
- ImageNet model들보다 성능 및 효율적인 측면에서 더 우수함
- zero-shot 평가 또한 더 우수함

Approach

2-1. Natural language supervision

논문의 core 접근법

- Natural language supervision (자연어를 통한 지시(또는 감독))
- 자연어를 이용한 방법론은 여러가지 이점이 있는데
 1. scale 확장이 쉬움(추가적인 data labeling이 필요없음)
 2. 방대한 양의 text data를 인터넷에서 수동적으로 학습이 가능함
 3. '그저' 표현법을 학습하는 것이 아닌, 언어와 연결시켜 유연한 **zero-shot transfer**가 가능함!

Approach

2-2. Creating a Sufficiently Large Dataset

이전 연구에 사용된 기존 dataset

- MS-COCO, Visual Genome : 고품질, crowded-labeled data but, 규모가 작음!(약 10만개)
- YFCC100M : 1억개의 사진 but, 품질 일정x, 의미없는 사진 제목과 카메라 설정 설명을 포함

WIT(Web Image Text)

- 기존 dataset들만을 고려하는 것은 이 연구의 잠재력을 과소평가하는 결과를 초래함
- 인터넷에서 수집한 4억개의 (image, text)쌍의 dataset 구축
- Query당 최대 2만개 * 50만개의 query
- GPT-2 training data(WebText) 수와 비슷함

Approach

2-3. Selecting an efficient Pre-Training Method

이전 연구에 사용된 기존 CV system

- 너무 많은 계산 자원이 필요함(ResNetXt : 19 GPU year 등..)
- 1000개의 ImageNet 학습이 이정도인데.. -> **효율적인 학습방법 필요!**

초기 접근법

- 이미지 CNN + 텍스트 transformer : 이미지에 대한 텍스트 설명을 예측하는 수행 학습
- but, 효과적인 scale 확장에 어려움을 겪었음 : 순서를 고려하는 transformer 모델이 순서를 고려하지 않은 텍스트 모음(bag of words) 인코딩 예측보다 3배 더 느림

Approach

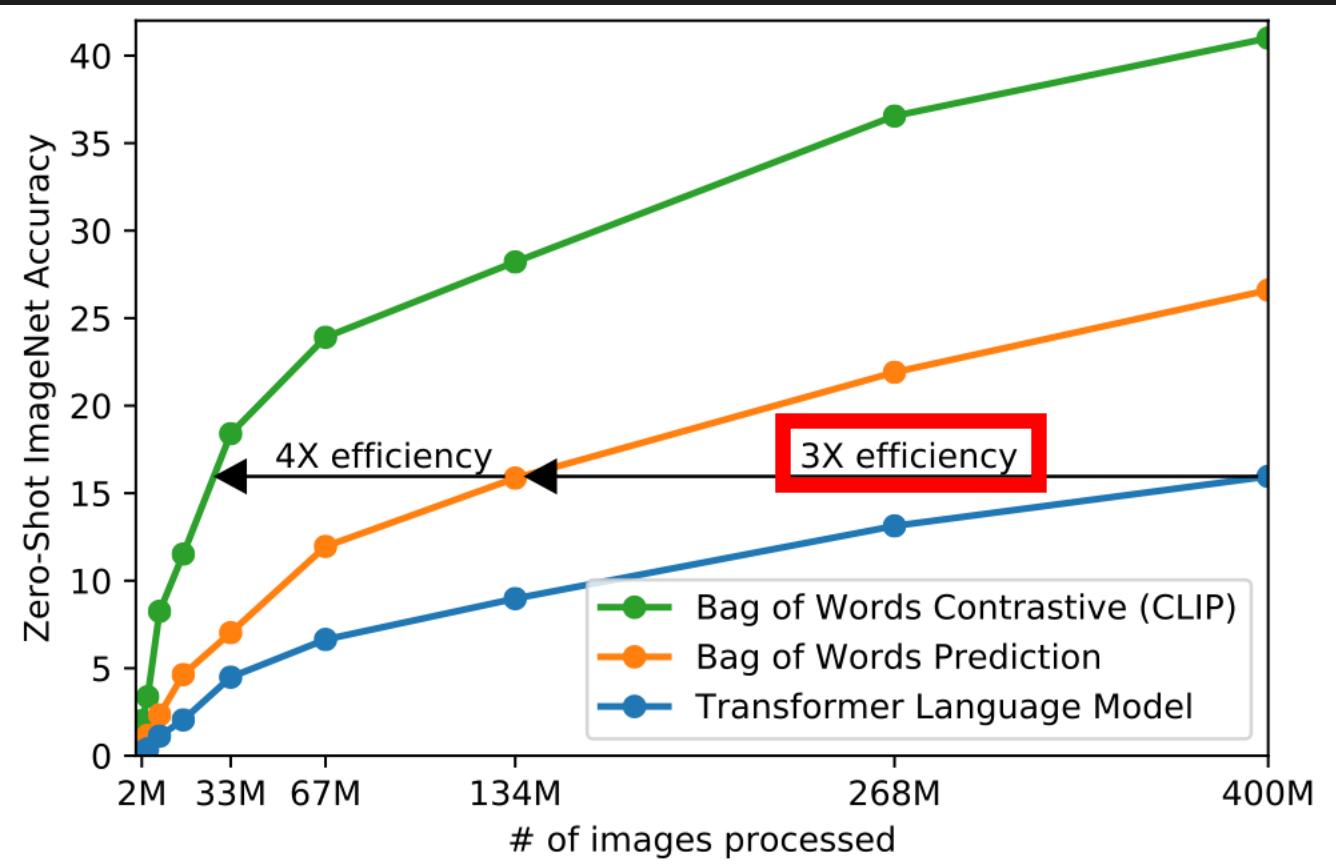
2-3. Selecting a

이전 연구에 사용된

- 너무 많은 계산
- 1000개의 Images

초기 접근법

- 이미지 CNN + 텍스트 CNN
- but, 효과적인 sequence를 고려하지 않은 텍스트



수행 학습
모델이 순서를 고려

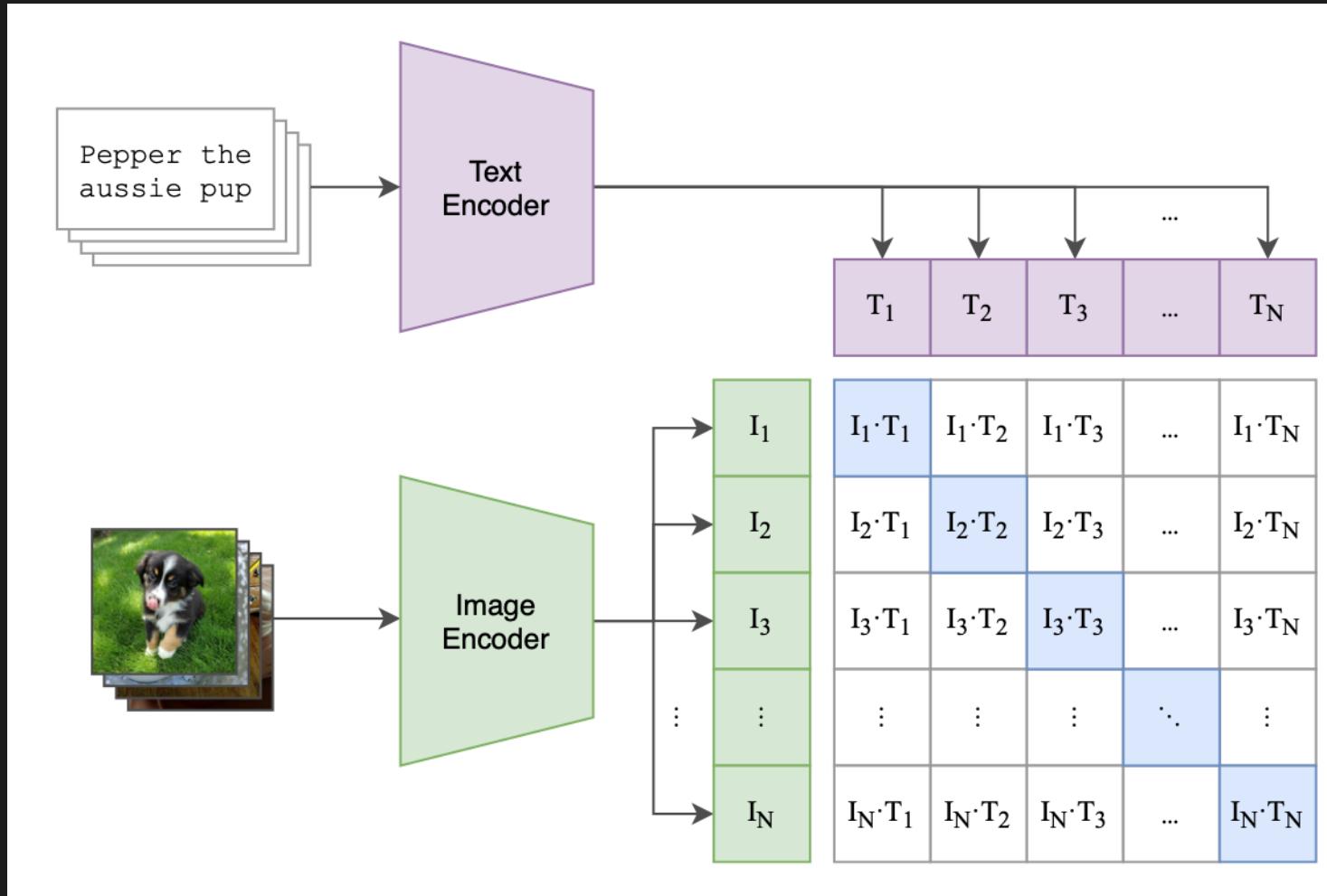
Approach

2-3. Selecting an efficient Pre-Training Method

새로운 접근법

- 텍스트의 정확한 다음 단어 예측은 어려운 작업임(transformer)
- 그래서 bag of words로 뮤어(순서 고려x) contrastive representation을 학습시키자
- contrastive representation : 주어진 두 이미지 텍스트 간의 유사도 학습

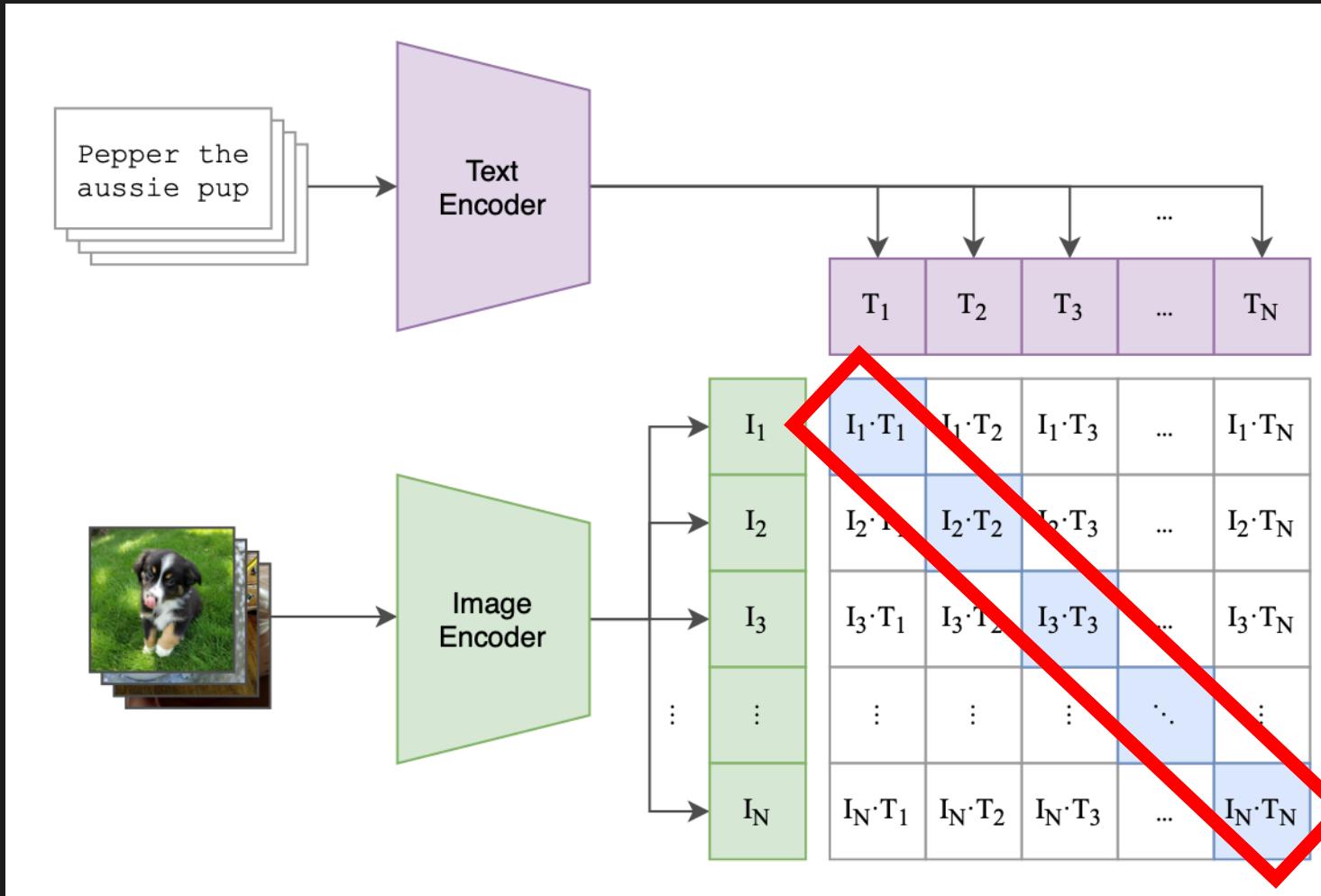
Approach



Batch size N : N개의 pair

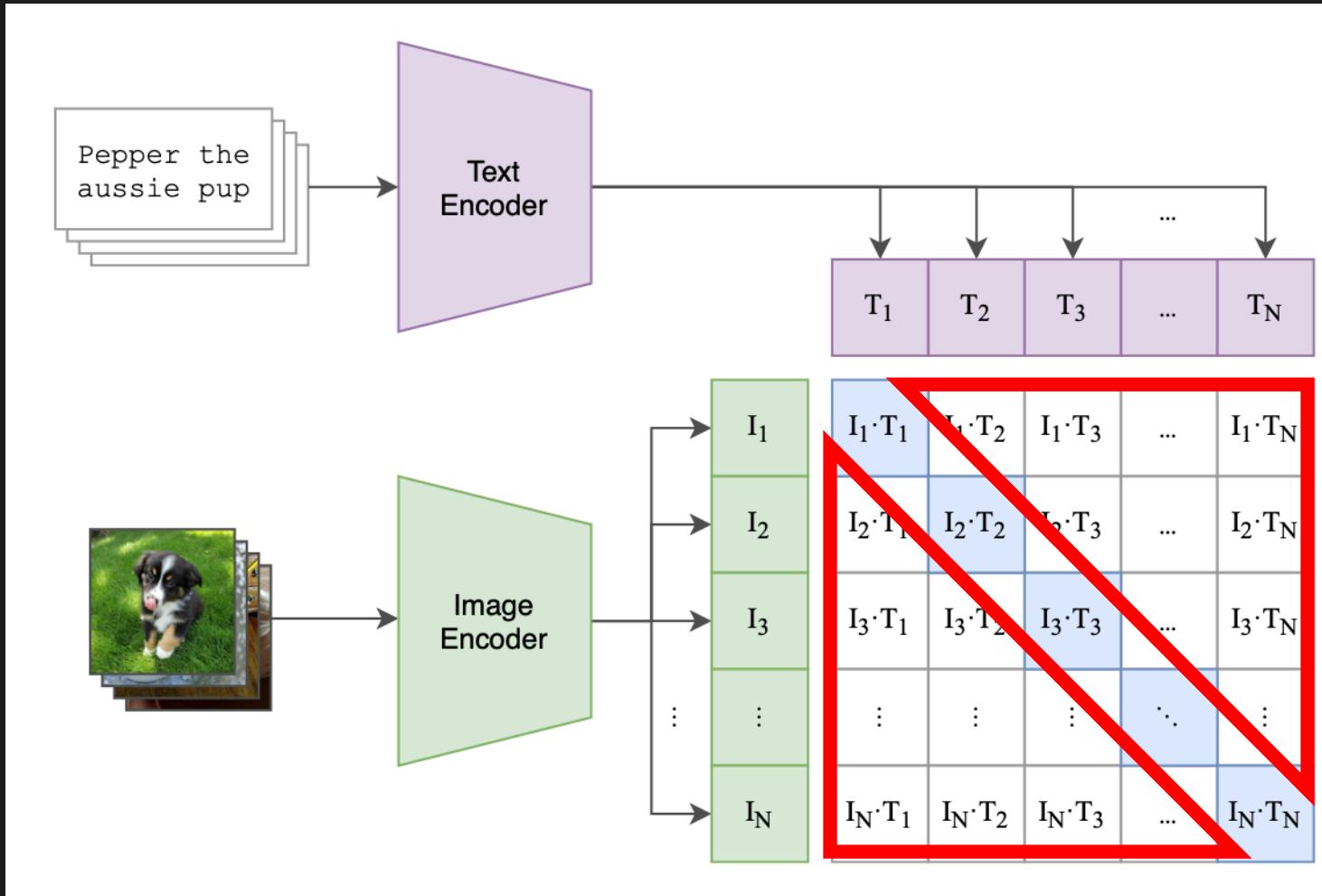
1. image, text 각각의 encoder
로 벡터로 변환
2. 내적을 통해 두 벡터간의 유사도
구함(이때, contrastive하게 구
함)
3. 총 N^2 개의 내적값이 나옴

Approach



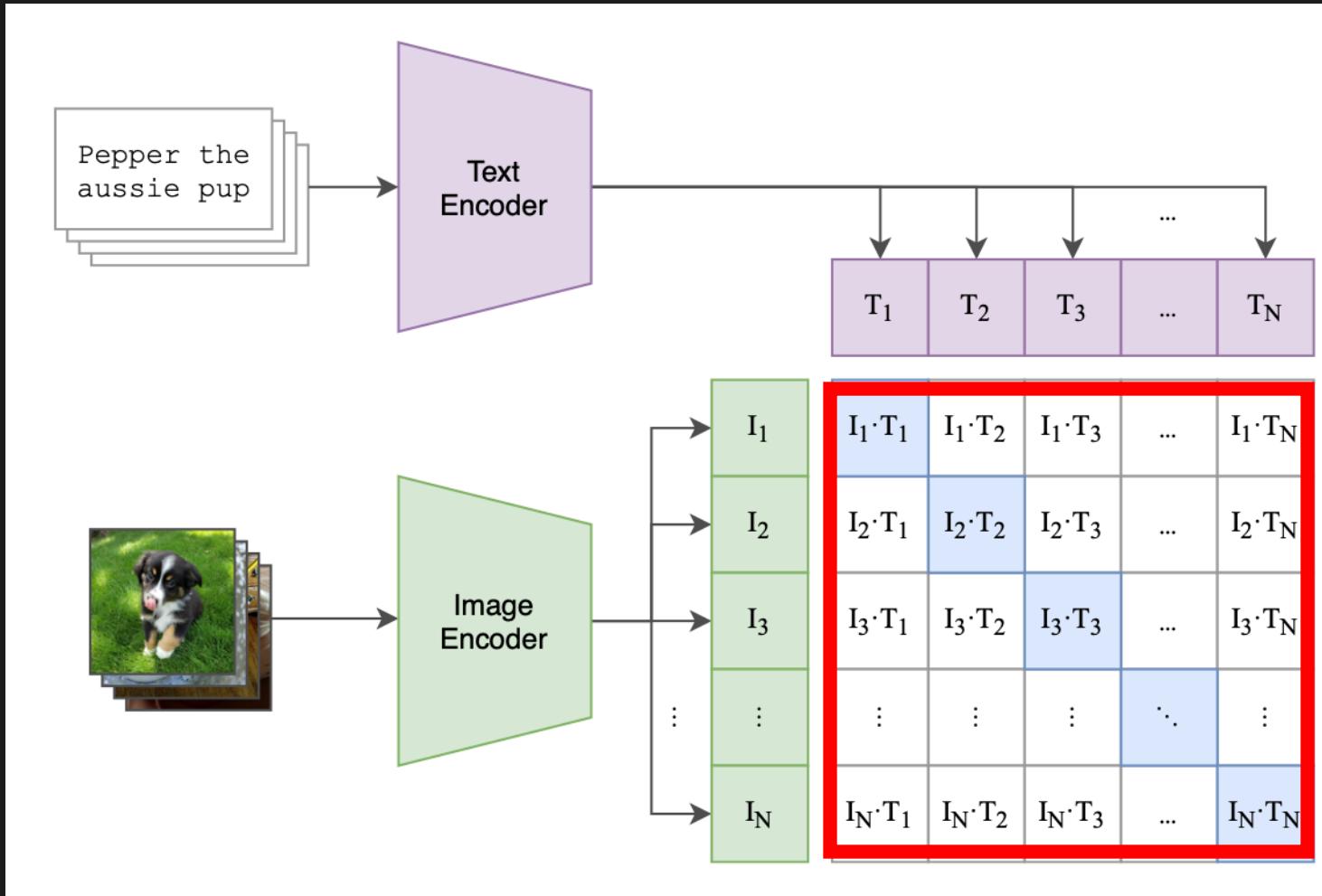
N개의 real pair에서 나온 값

Approach



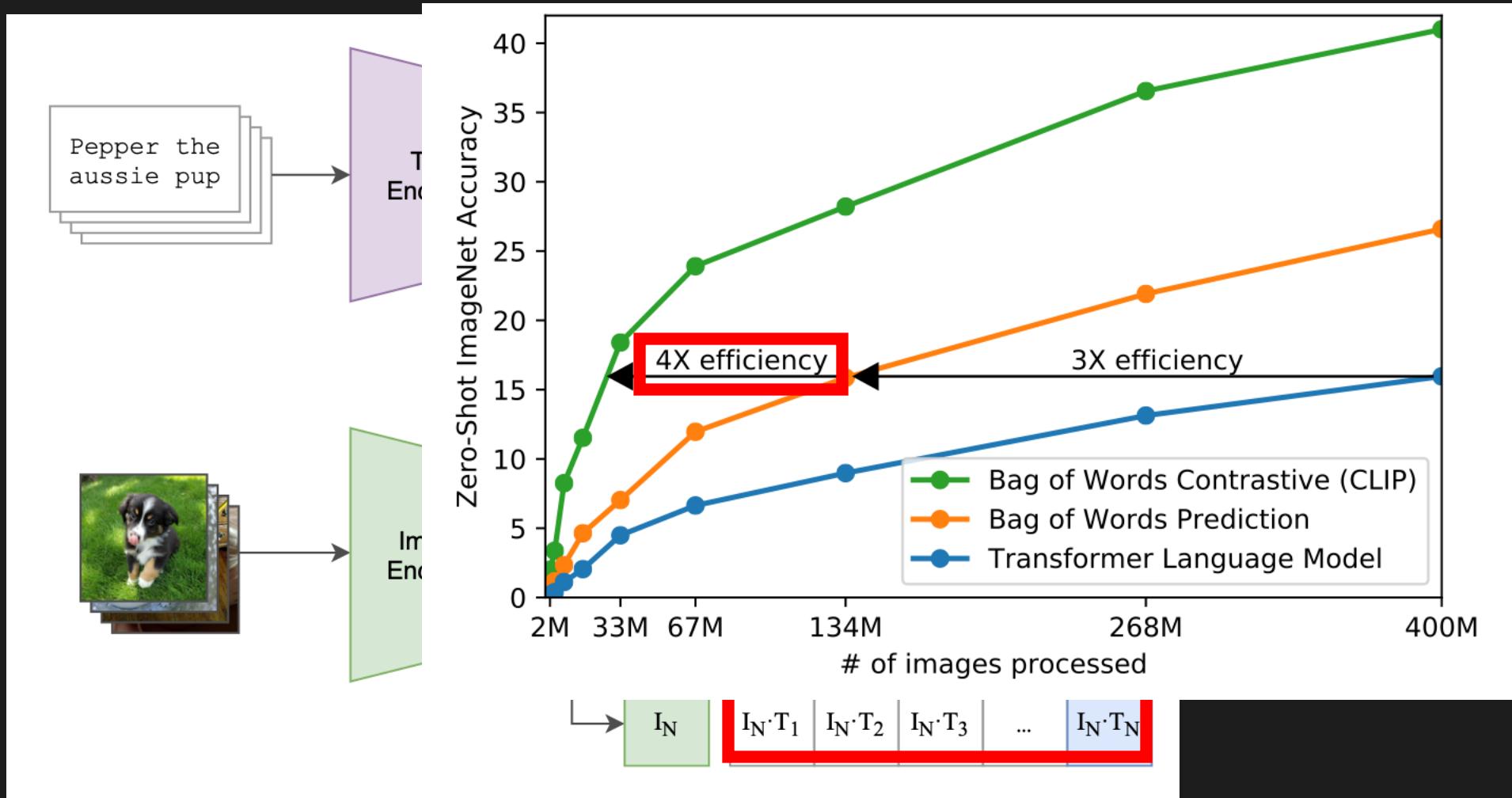
$N^2 - N$ 개의 incorrect pair에서 나온 값

Approach



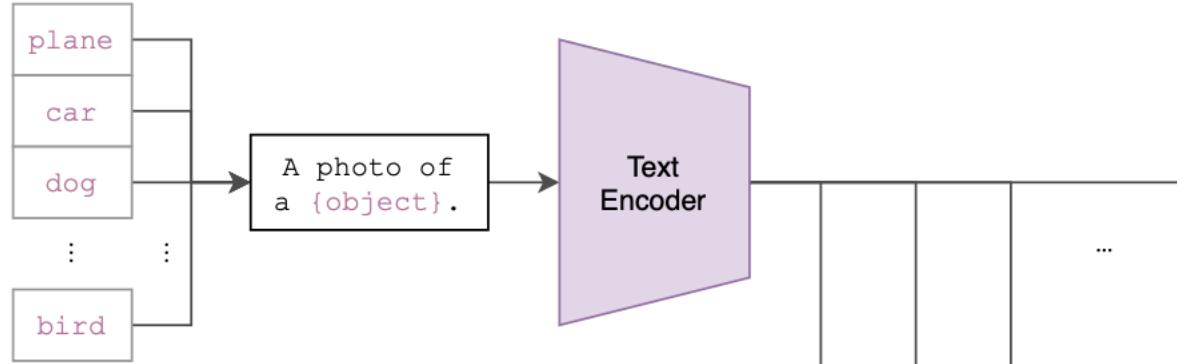
이 내적값들을
symmetric cross entropy loss값을 통해
최적화

Approach

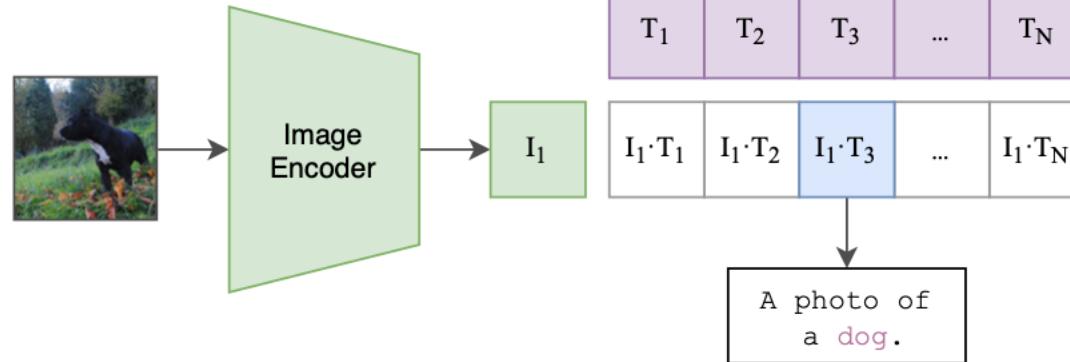


Approach

(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



Test 과정

1. dataset의 label로 prompt 생성
2. 생성된 프롬프트 embedding
3. 예측하고자 하는 이미지 벡터와 label 별로 생성된 프롬프트 벡터 유사도 계산 후 상대적으로 제일 높은 값 선택

Approach

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

1. 각 encoder를 통한 feature 추출

Approach

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

2. weight metrix 곱한다음, L2 norm

Approach

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

3. 각각의 Image, Text 벡터 간의 유사도(내적)계산
(t : softmax에 사용되는 temperature parameter)

Approach

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Approach

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```

4. [0, 1, 2, 3, ... ,n-1]의 정답 label 생성
-> 각 data가 동일한 index의 쌍으로 묶이는지 판별

Approach

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

5. 각 data 별로 정답 label과 cross entropy loss 계산
-> axis를 기준으로 분리

Approach

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

6. 두 개의 loss를 평균

Approach

- pre-train dataset이 매우 크기 때문에(약 4억개) overfitting은 고려하지 않음
- 그렇기 때문에 image encoder, text encoder를 추가적으로 초기화할 필요가 없음
- 기존처럼 non-linear projection이 아닌, linear projection을 사용(내적을 통한 유사도 계산) -> 별 차이가 없었기 때문

Approach

2-4. Choosing and Scaling model

- image encoder : ResNet-50 (attention pooling 대체) + ViT
- text encoder : transformer(12 layer, 512 wide, 8 head)
- 이전의 CV연구와 달리 ResNet의 경우 wide, depth, resolution 모두 확장(성능 우수)
- text encoder는 ResNet 너비에 맞췄으나 깊이를 증가시키지는 않음(CLIP 성능이 text encoder에 덜 민감하다는 것을 발견)

Approach

2-5. Training

- 5개의 ResNet(ResNet-50, ResNet-101, EfficientNet추가하여 4배, 16배, 64배 추가), 3개의 ViT(ViT-B/32, ViT-B/16, ViT-L/14) -> 총 32 epoch
- 학습을 위해 사용된 technique
 - use minibatch size of 32,768
 - gradient checkpointing
 - half-precision Adam statistics
 - half-precision stochastically rounded text encoder weights
- RN50x64로 학습시키는데 592대의 V100으로 GPU 18days, ViT 모델에서는 256대의 V100으로 GPU 12 days

Experiment

	aYahoo	ImageNet	SUN
Visual N-Grams	72.4	11.5	23.0
CLIP	98.4	76.2	58.5

CLIP vs Visual N-grams / zero-shot transfer comparison

- 각각의 dataset에 대해서 CLIP이 압도적으로 성능이 좋은 것을 확인할 수 있음

Experiment

Prompt engineering and ensembling

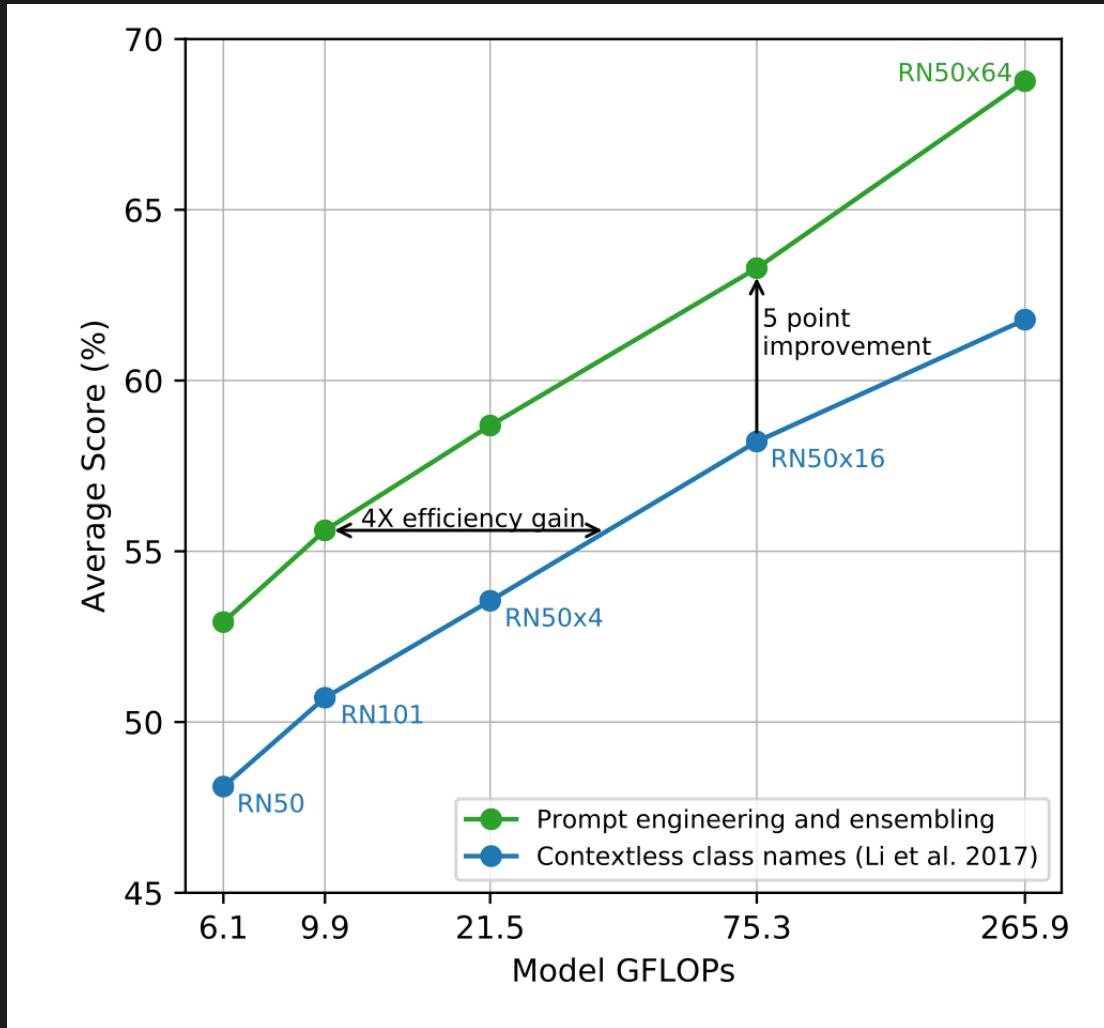
- 여러가지 문제점
 1. 다의어(Polysemy) : 클래스의 이름만으로는 이미지를 구분할 수 없음
(ex. 건설현장에서의 crane / 두루미의 crane)
 2. pre-train dataset에서 text가 하나의 단어로만 이루어진 경우는 거의 없음
-> 일반적으로 Image를 설명하는 문장의 형태

Experiment

Prompt engineering and ensembling

- 해결 방법
 1. 문장의 분포 차이를 해결하기 위해 ‘A photo of a {label}’이라는 프롬프트 사용
-> ImageNet에서 정확도가 1.3% 향상됨
 2. 카테고리 명시가 도움이 된다는 것을 알아 냄
-> ex. ‘A photo of a {label}, type of {category}’
->pets dataset에서는 pet명시를, food dataset에서는 음식종류 명시를, OCR에서는 인식대상에 따옴표 등을 명시하는 등
 3. ensemble 사용 : 여러 zero-shot classifier(각기 다른 프롬프트)를 사용한 후 결과를 종합
-> ‘A photo of~’, ‘A picture of~’, ‘A photo of small~’등
-> 확률 공간이 아닌 임베딩 공간에서 양상을 구성 후, 단일 평균화된 text 임베딩 세트에 저장
-> ImageNet에서 80개의 프롬프트 사용, 단일 기본 프롬프트보다 정확도 3.5% 향상됨

Experiment

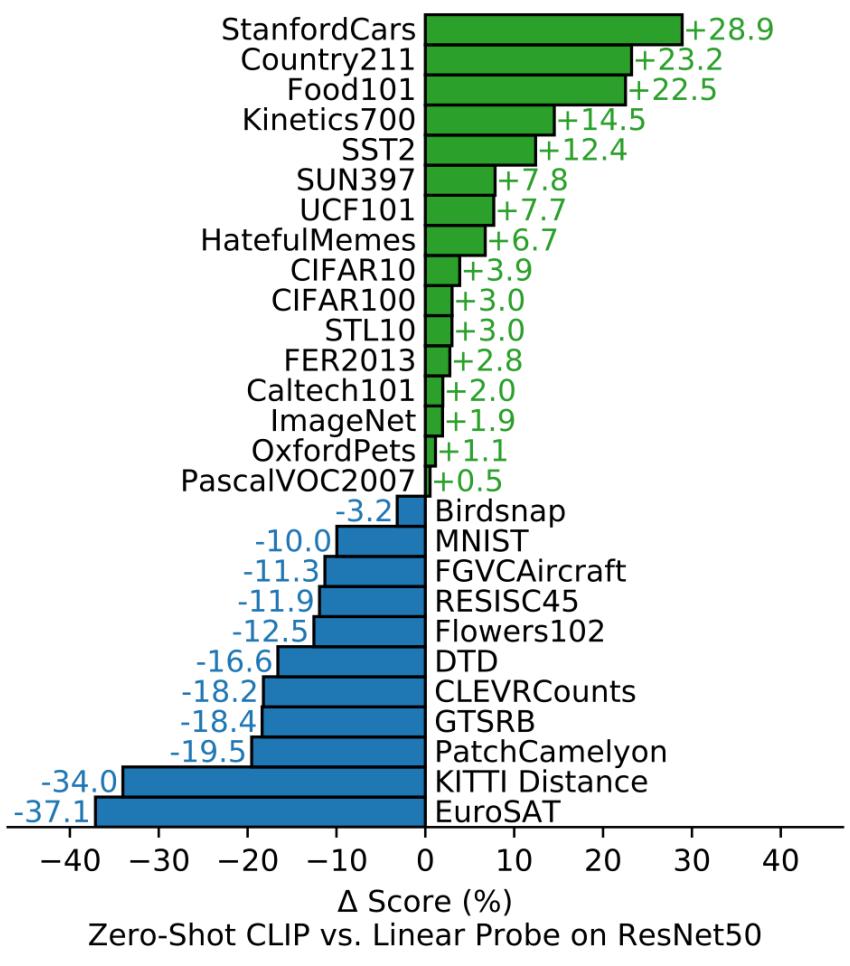


Prompt engineering and ensembling

- 결론적으로 프롬프트 + 양상률은 정확도 5%향상에 기여함

Experiment

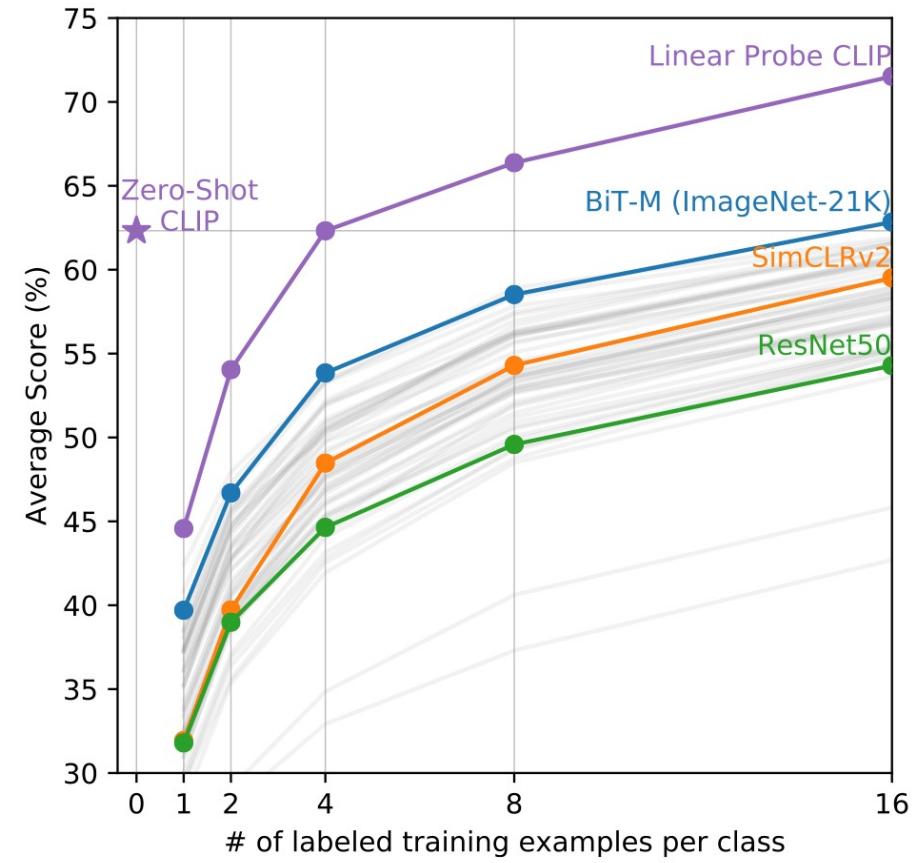
Zero-shot CLIP vs ResNet-50



- 27개 dataset에서 16개가 CLIP이 더 우세함을 보였음
- 특히 비디오 동작 인식 데이터에서 높은 성능을 보임
→ 자연어가 동사를 포함하는 시각적 개념에 대해 더 넓은 supervision을 제공
- 일반적인 객체 분류 dataset에서는 성능이 상대적으로 비슷함
- 위성 이미지 분류, 림프절 종양 탐지, 독일 교통판 표지 인식 등 전문적이고 복잡하고 추상적인 dataset에서는 CLIP의 성능이 저조하다는 것을 확인할 수 있음
- few-shot transfer를 통해 개선 가능성의 여지가 있음

Experiment

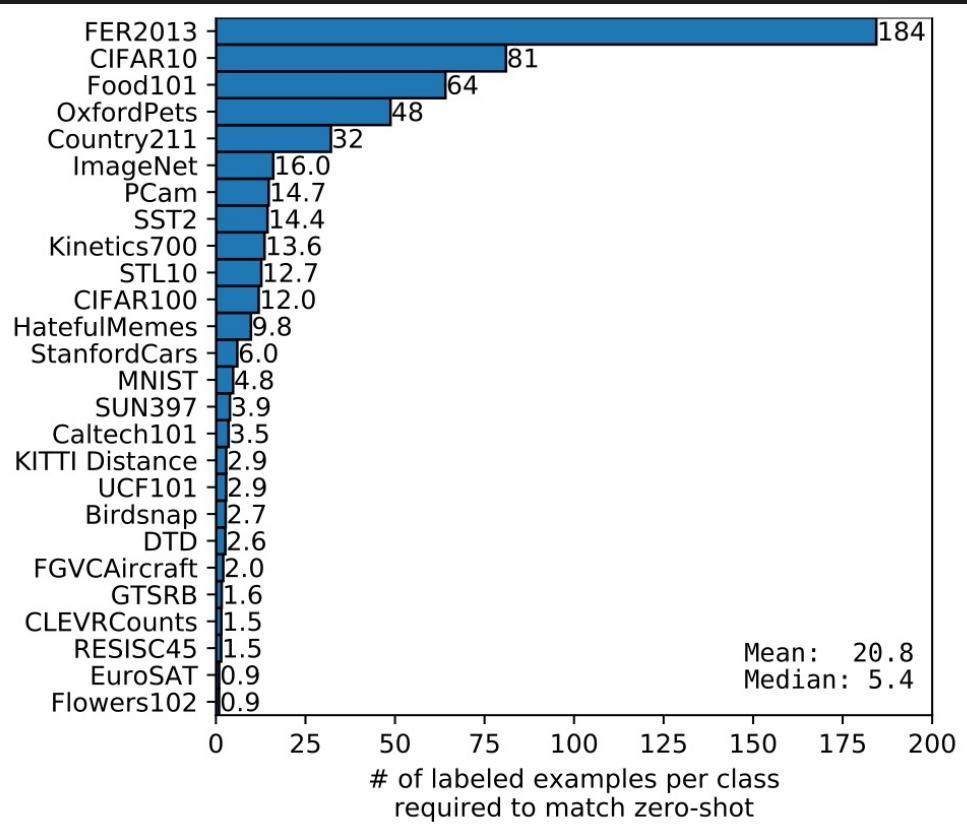
few-shot CLIP comparison



- zero-shot CLIP(보라색 별)이 다른 few-shot model들보다 이미 우세함을 보임
- few-shot CLIP(보라색 선)이 다른 few-shot model들보다 매우 우세함을 보이며, example수가 많이 주어질수록 성능이 더 좋아지는 것을 확인할 수 있음

Experiment

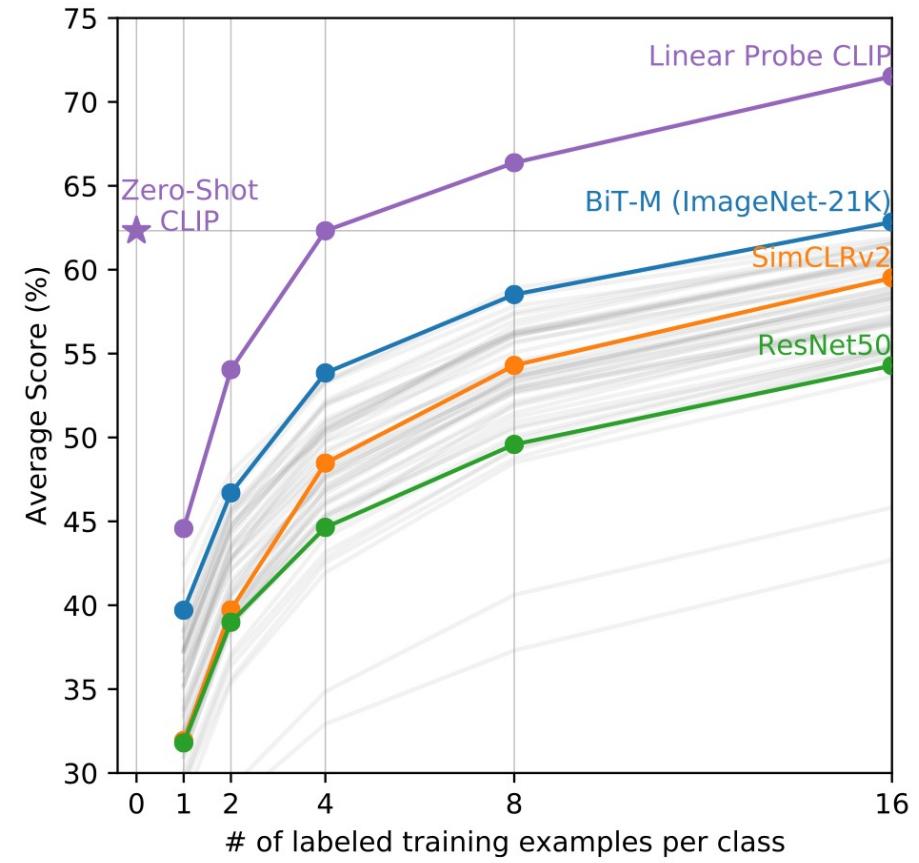
zero-shot vs few-shot



- 특정 dataset별로 zero-shot과 성능이 비슷해질 수 있도록 필요한 example의 수

Experiment

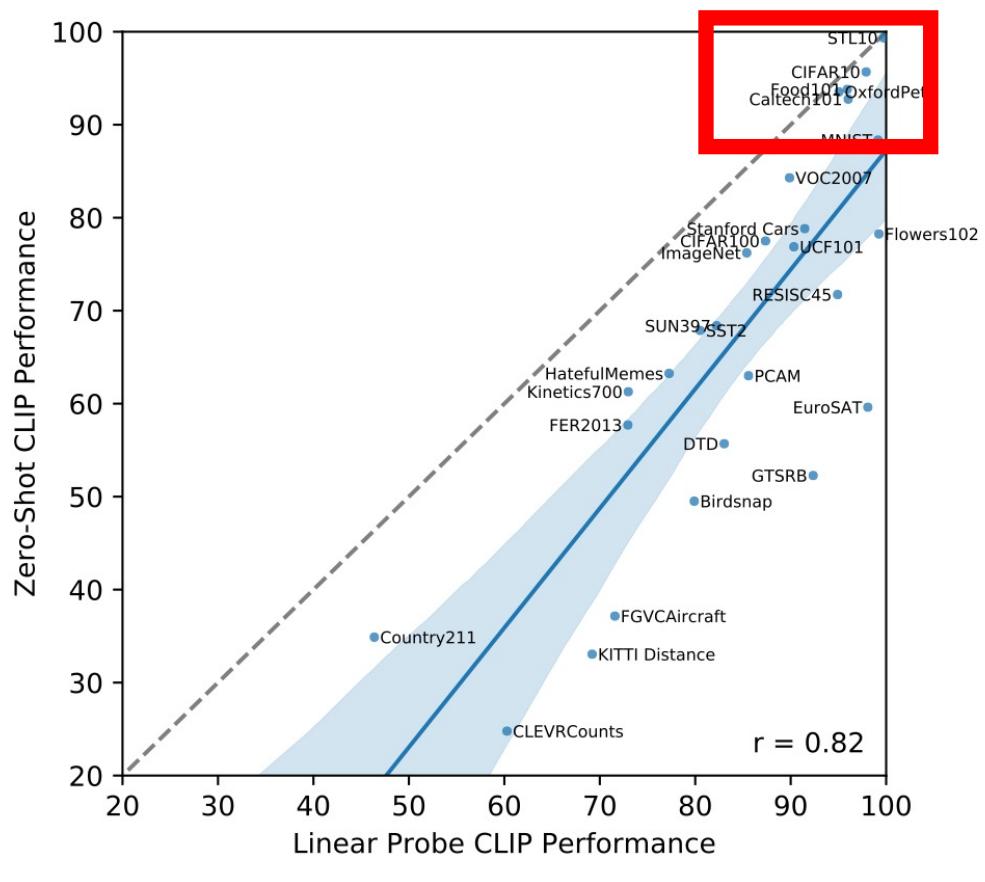
few-shot CLIP comparison



- zero-shot CLIP(보라색 별)이 다른 few-shot model들보다 이미 우세함을 보임
- few-shot CLIP(보라색 선)이 다른 few-shot model들보다 매우 우세함을 보이며, example수가 많이 주어질수록 성능이 더 좋아지는 것을 확인할 수 있음

Experiment

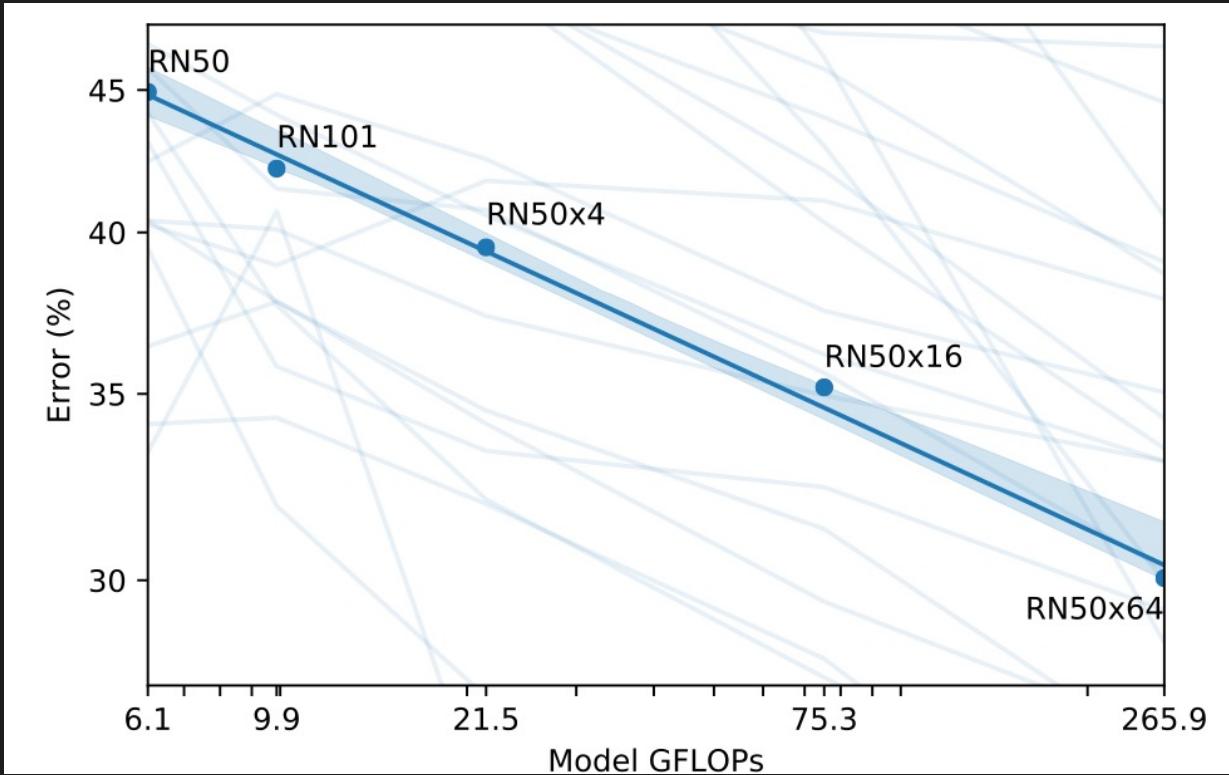
zero-shot vs linear prob



- zero-shot 과 마지막 layer에 classifier를 추가하여 classifier만 학습시킨 linear prob의 비교 graph
- 검은색 점선 : 이상적인 상황 / zero-shot과 linear prob의 성능이 같을 때
- 파란색 실선 : 실제 성능 차이 / zero-shot이 linear prob보다 10~25 포인트 성능이 낮은 것을 확인할 수 있음
- 일부 5개의 dataset(빨간 박스)에서는 이상적인 상황이 나타나지만, 대부분은 linear-prob가 우세함

Experiment

model 계산량과 성능과의 관계



- 계산량이 증가할 수록, error가 linear하게 감소하는 것을 확인할 수 있음
- 개별 dataset에서는 성능의 변동이 크다는 것을 확인할 수 있음
- 결론적으로 더 많은 연산자원을 사용할 수록 model의 성능이 더 좋아지는 것을 확인할 수 있음

Experiment

3-2. Representation Learning

보다 일반적이고 이상적인 표현의 기준이 무엇인가에 대한 연구가 필요함.

Experiment

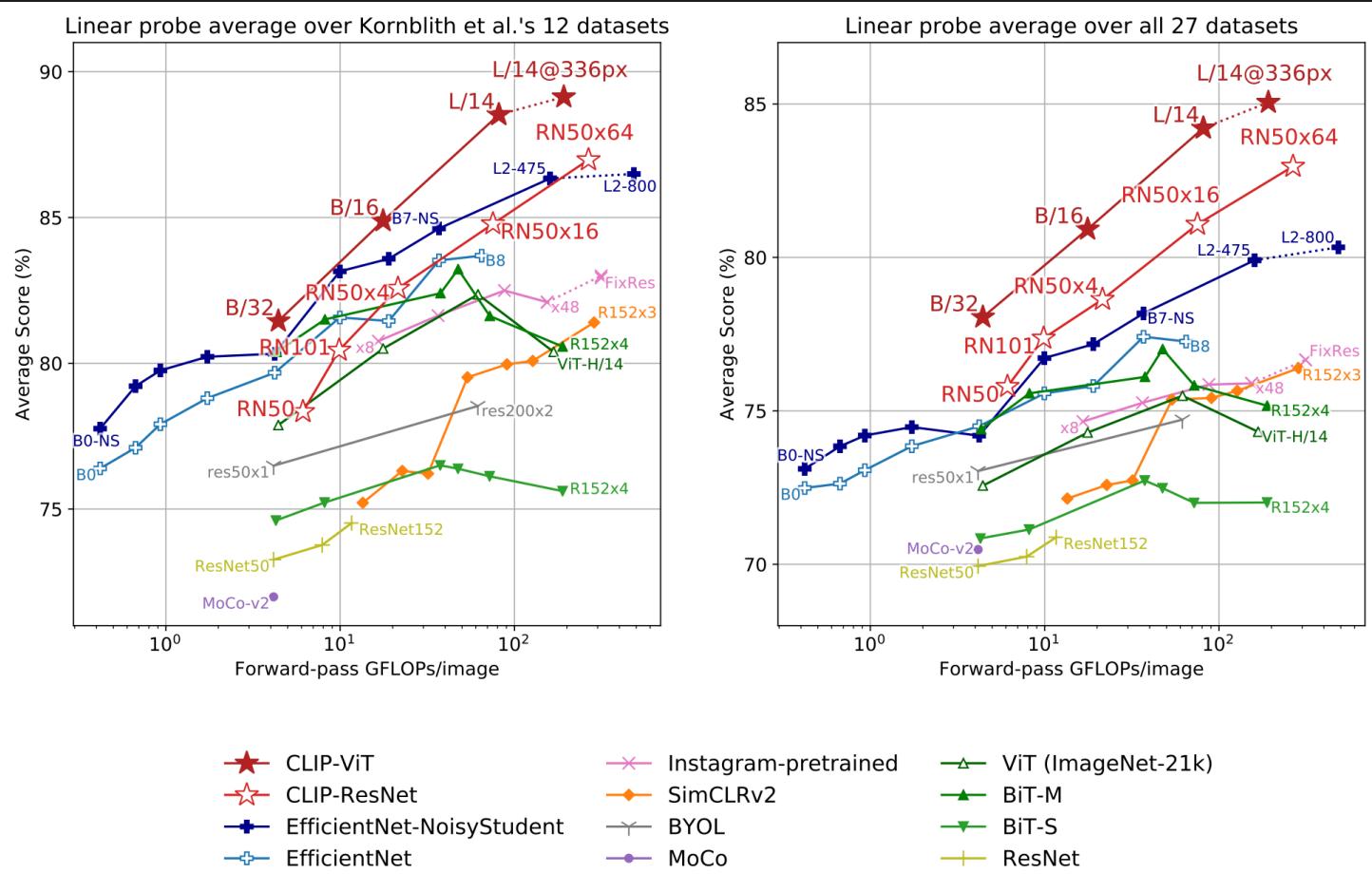
Representation Learning

- fine tuning vs linear classifier

fine tuning	linear classifier
representation을 dataset에 맞게 조정	일반적이고 강력한 representation 학습 가능
유연성 좋음	유연성 제한적 -> 실험에서 즉각적인 feedback
	zero-shot classifier와 접근방식이 유사함
larger design, 더 큰 hyoer-parameter space -> 공정한 비교가 어려움	최소한의 hyper-parameter만 조정 가능 -> 표준화된 representation
비교 비용이 큼	비교 비용이 작음

Experiment

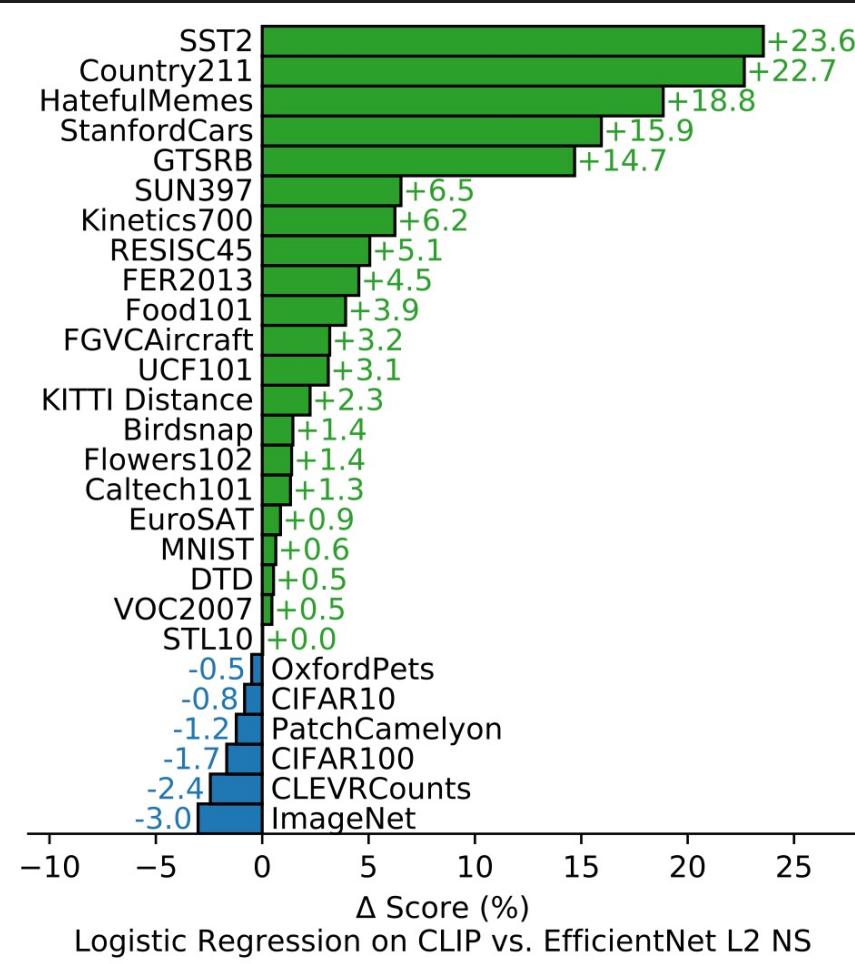
Linear probe comparison



model 별 성능차이
(12 datasets, 27 datasets)
CLIP이 다른 model들보다 성능이 더 좋은 것을 확인할 수 있음

Experiment

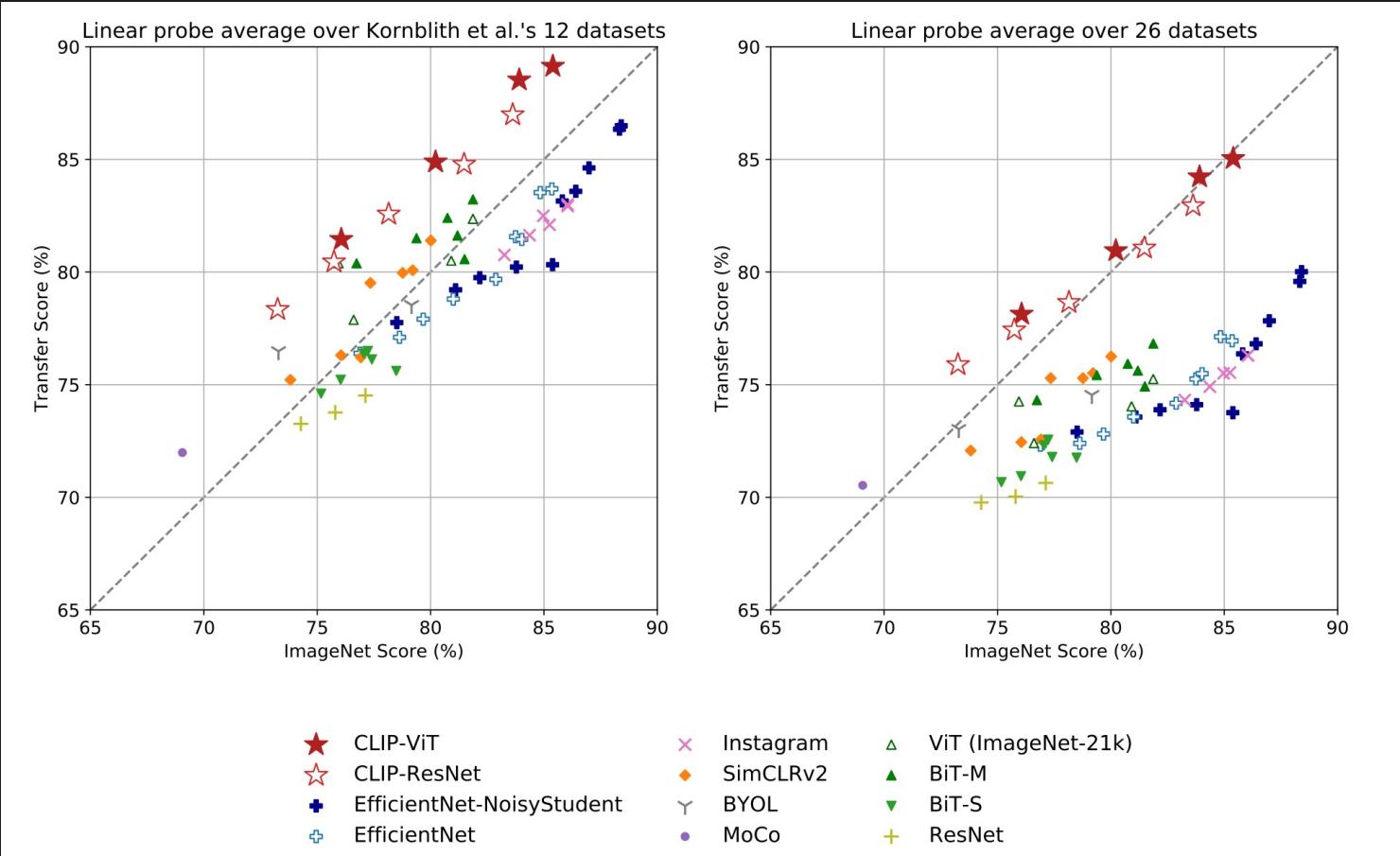
CLIP vs EfficientNet



- 대부분의 dataset에서 우수함을 보임
- 다만 단일 라벨, 저해상도 dataset에서는 EfficientNet이 다소 우수함
- EfficientNet을 학습하는데 쓰인 ImageNet에서도 EfficientNet이 다소 우수함

Experiment

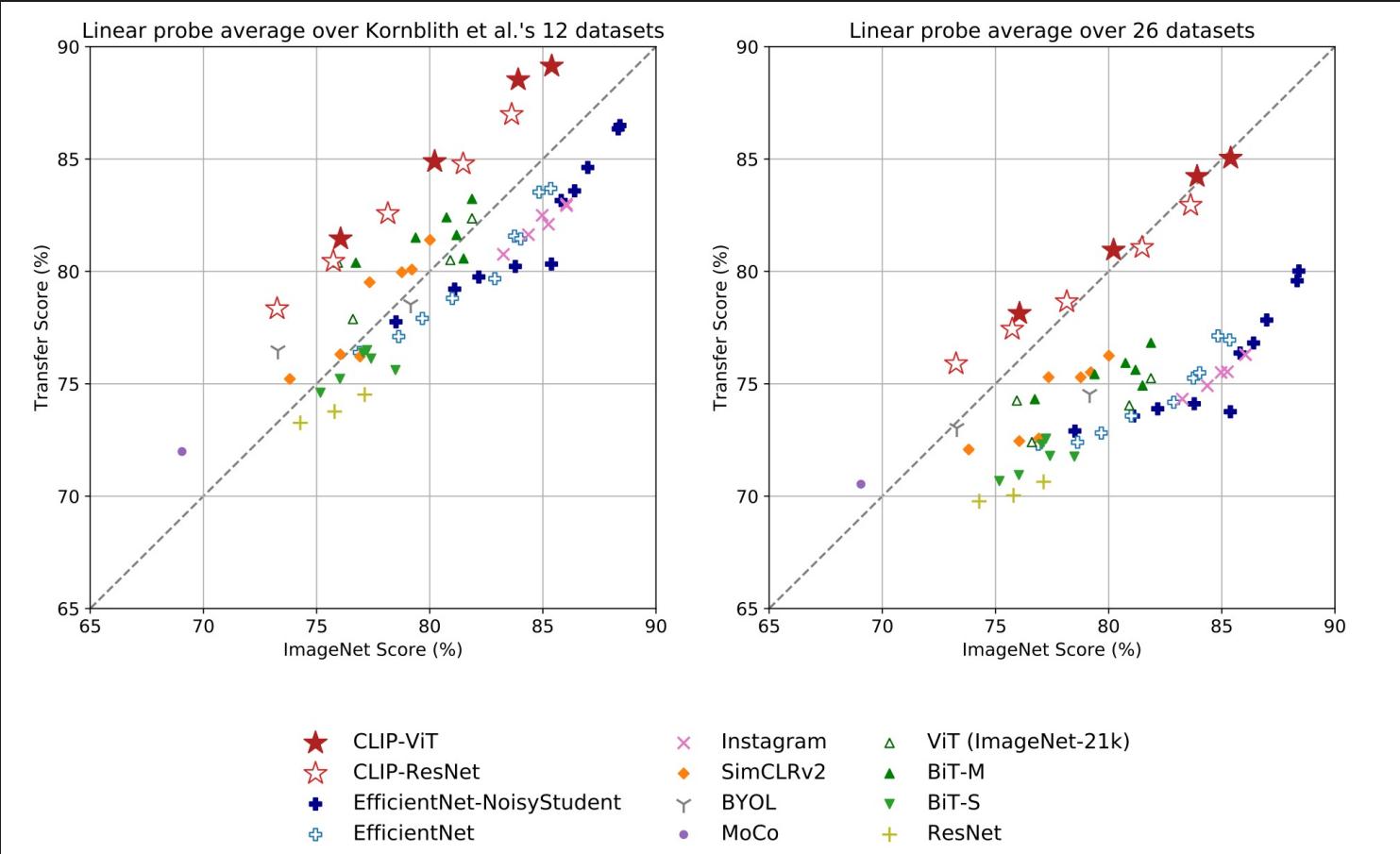
3-3. Robustness to Natural distribution shift



- ImageNet과 다른 dataset에서의 linear probe 성능 비교
- 많은 model들이 대각선보다 아래에 있음
→ ImageNet에 overfit!
- CLIP은 dataset에 상관없이 성능이 좋은 것을 확인할 수 있음

Experiment

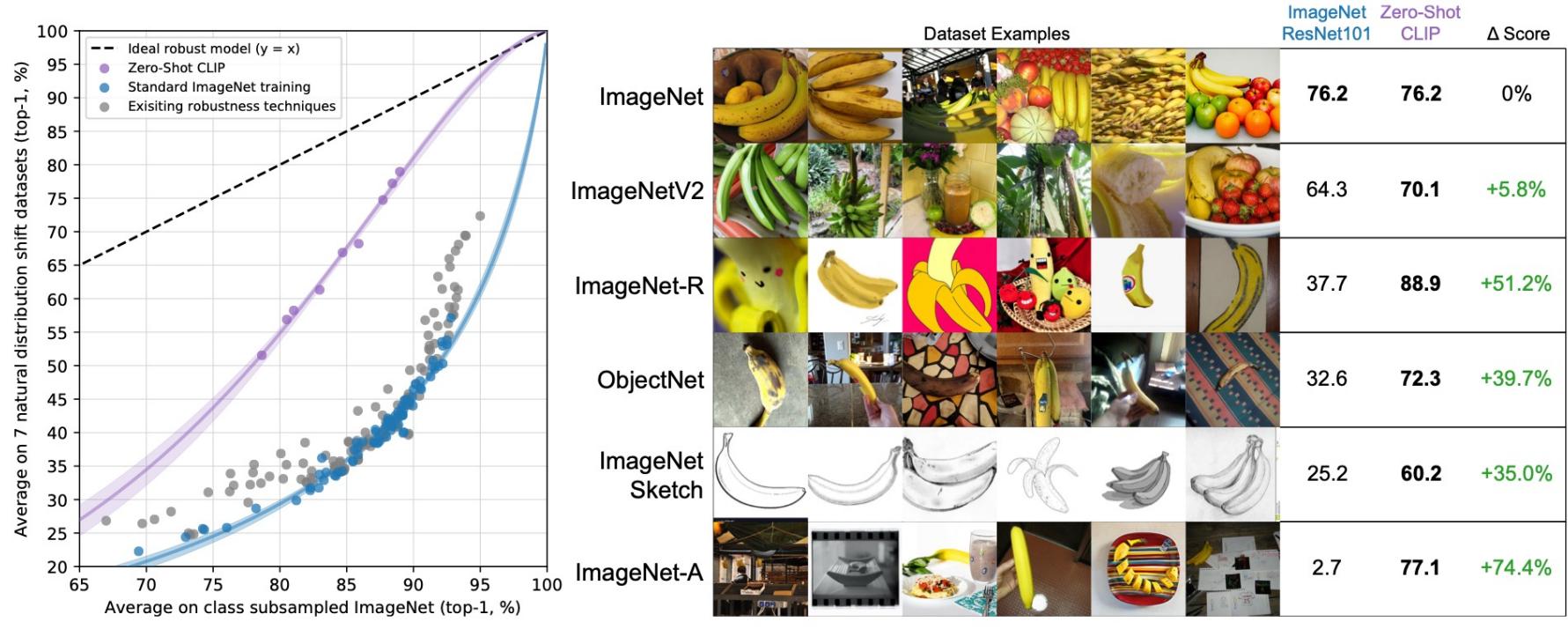
3-3. Robustness to Natural distribution shift



- ImageNet과 다른 dataset에서의 linear probe 성능 비교
- 많은 model들이 대각선보다 아래에 있음
→ ImageNet에 overfit!
- CLIP은 dataset에 상관없이 성능이 좋은 것을 확인할 수 있음

Experiment

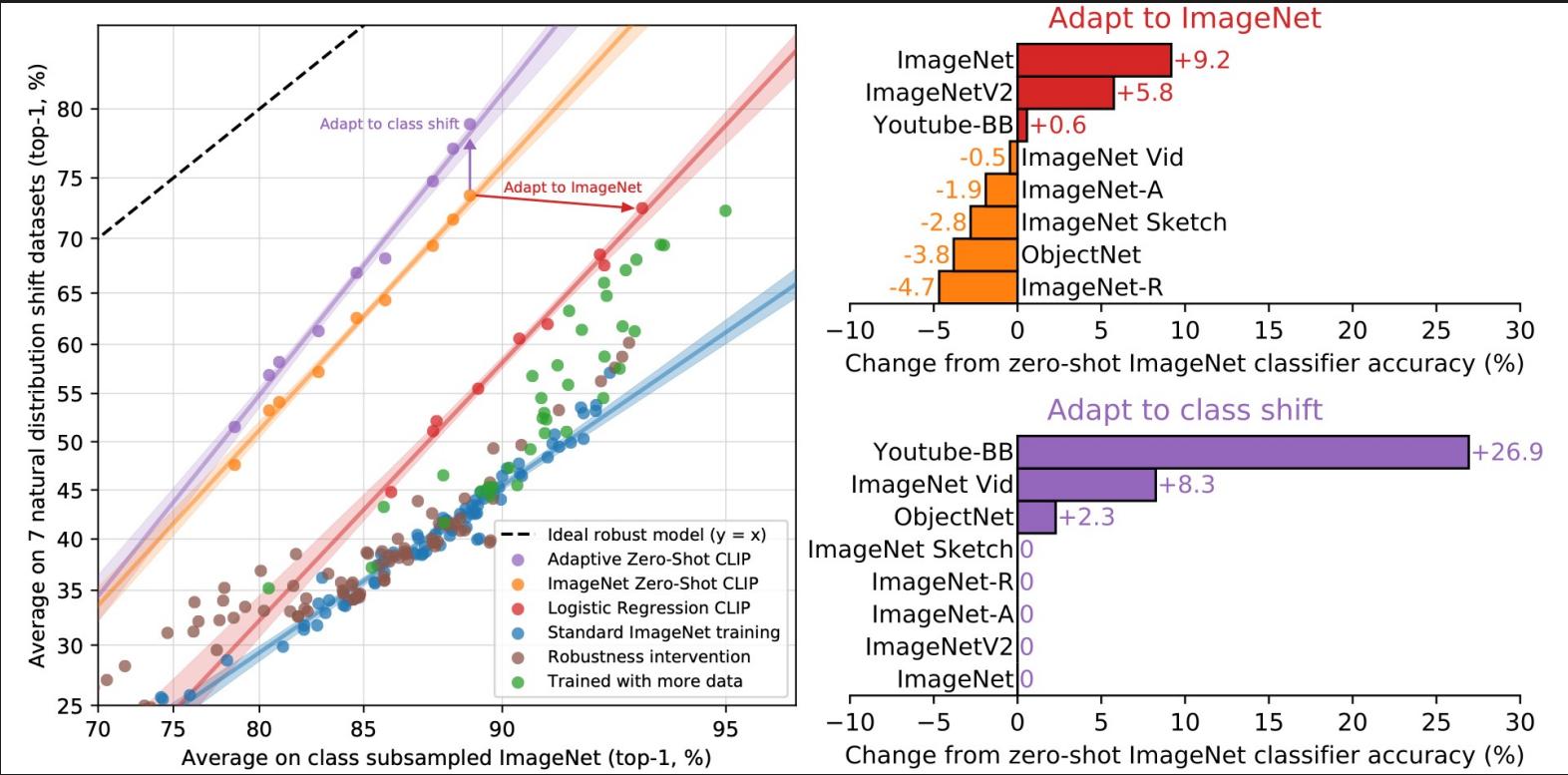
3-3. Robustness to Natural distribution shift



- ImageNet으로 학습한(+robustness techniques)model들에 비해 CLIP의 robustness 성능이 더 좋은 것을 확인할 수 있음

Experiment

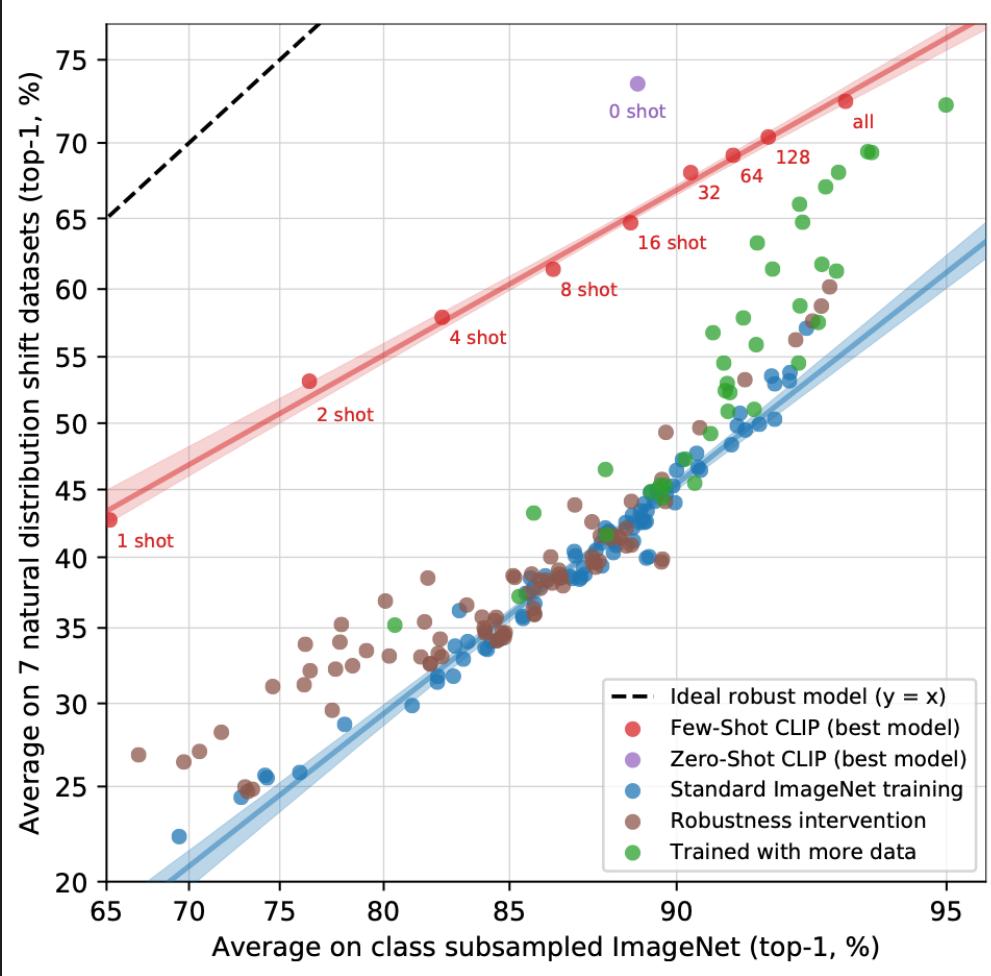
3-3. Robustness to Natural distribution shift



- zero-shot CLIP이 오히려 다른 dataset(ImageNet)에 supervised된 model들보다 robustness 성능이 더 좋음

Experiment

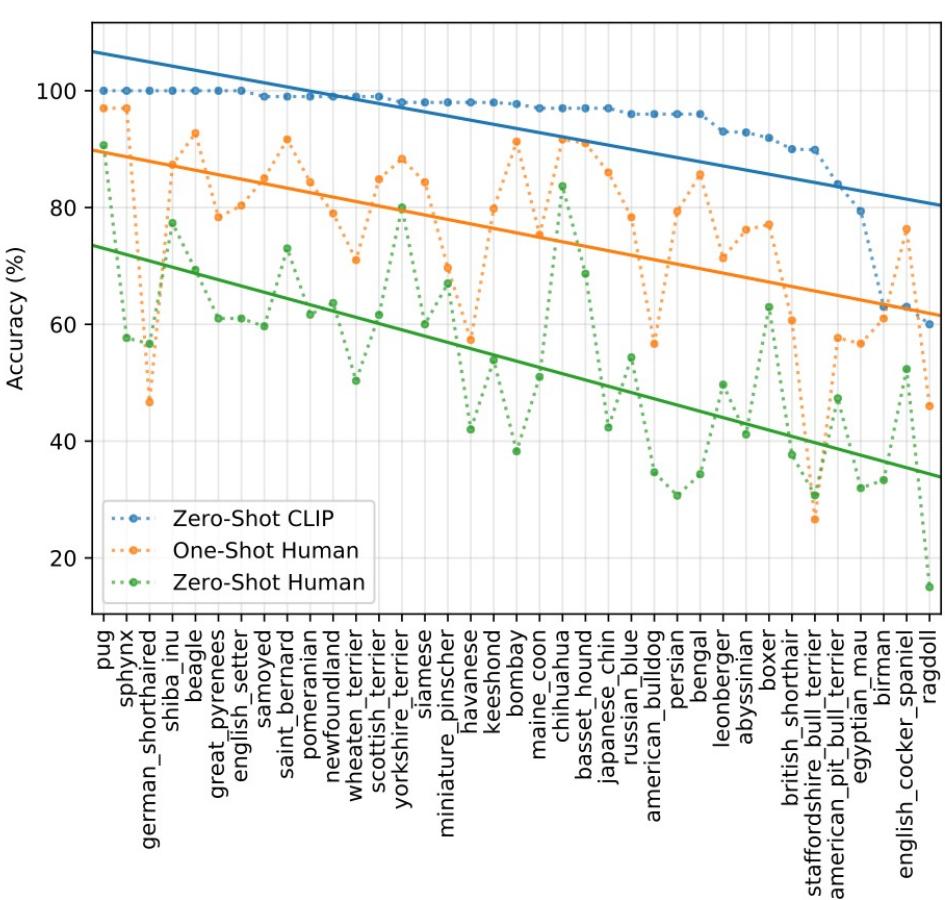
3-3. Robustness to Natural distribution shift



- CLIP이 다른 model들에 비해 robustness가 높은 것을 알 수 있음
- few-shot CLIP보다 zero-shot CLIP이 오히려 더 robustness가 높은 것을 확인할 수 있음

Experiment

4. Comparison to Human Performance

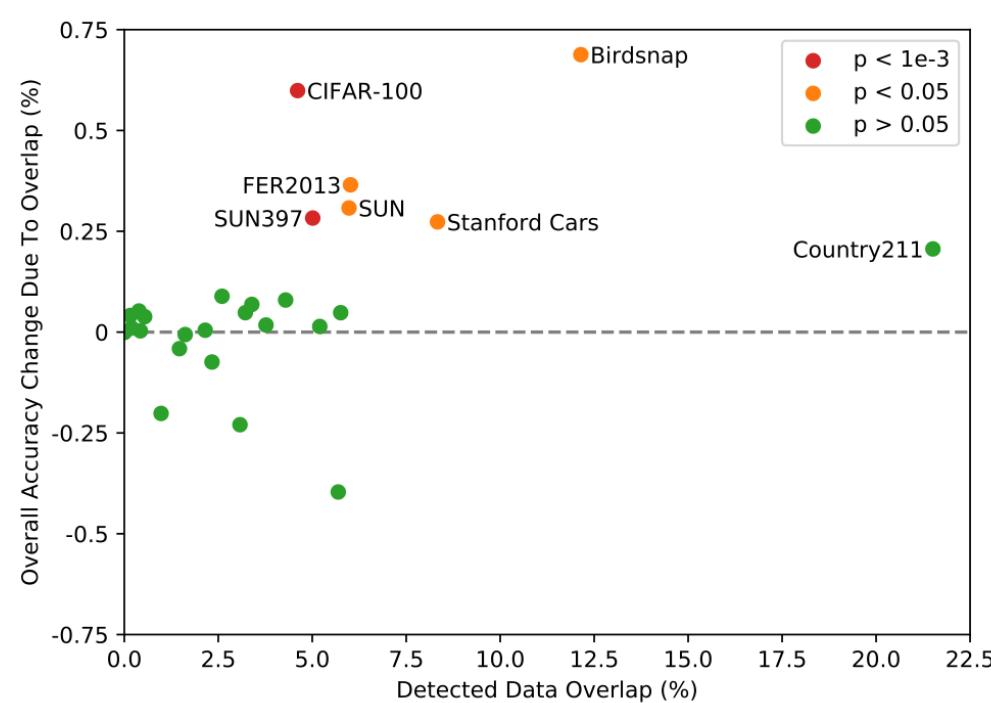
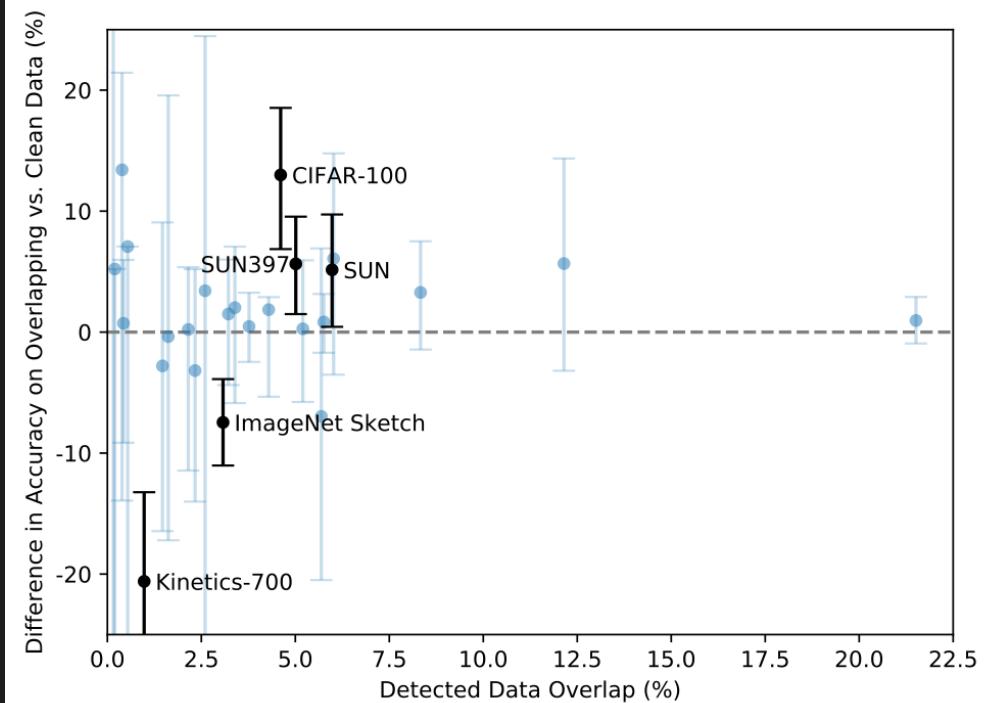


- CLIP이 인간의 성능(?)보다 더 좋은 것을 확인할 수 있음
- CLIP에게 어려운 문제는 인간에게도 어렵다!

	Accuracy	Majority Vote on Full Dataset	Accuracy on Guesses	Majority Vote Accuracy on Guesses
Zero-shot human	53.7	57.0	69.7	63.9
Zero-shot CLIP	93.5	93.5	93.5	93.5
One-shot human	75.7	80.3	78.5	81.2
Two-shot human	75.7	85.0	79.2	86.1

Experiment

5. Data overlap analysis



- data 중복이 모델 성능에 미치는 영향이 미미하다

Experiment

6. Limitation

1. 성능 부족(성능 개선을 위해서 많은 연구가 필요함)
2. computing resource
3. 특정 작업에서 약함(특정 카테고리에서 종류 구분, 물체 수 세기 등 세밀한 분류 작업에서 저조)
4. truly out-of-distribution에 약함(학습 중에 접하지 않은 data 분포)
5. few-shot에서의 성능이 zero-shot보다 떨어짐

Experiment

7. Broader Impacts

7-1. Bias

- CLIP이 학습하는 dataset과 class design으로 사회적 편향을 발생시키고 확대할 수도 있다
- FAIRFACE dataset(인종 별 사람 얼굴 데이터)을 분류해 보았을 때..
-> ‘BLACK(흑인)’이 비인간적인 카테고리(‘animal’, ‘gorilla’, ‘chimpanzee’ 등)에 잘못 분류된 비율이 높았다..
- 범죄관련 카테고리(‘thief’, ‘criminal’, ‘suspicious person’ 등)에 20대 남성 이미지를 여성 이미지보다 더 잘못 분류하는 비율이 높았다..

Experiment

7. Broader Impacts

7-2. Surveillance

- CCTV의 저해상도 이미지, celeb(셀럽) 이미지 학습으로 인해 유명인 식별 및 감시에 사용될 수도 있음
- 다만 well-tailored model들이 CLIP보다 감시에 더 잘.. 어울린다고 할 수 있음

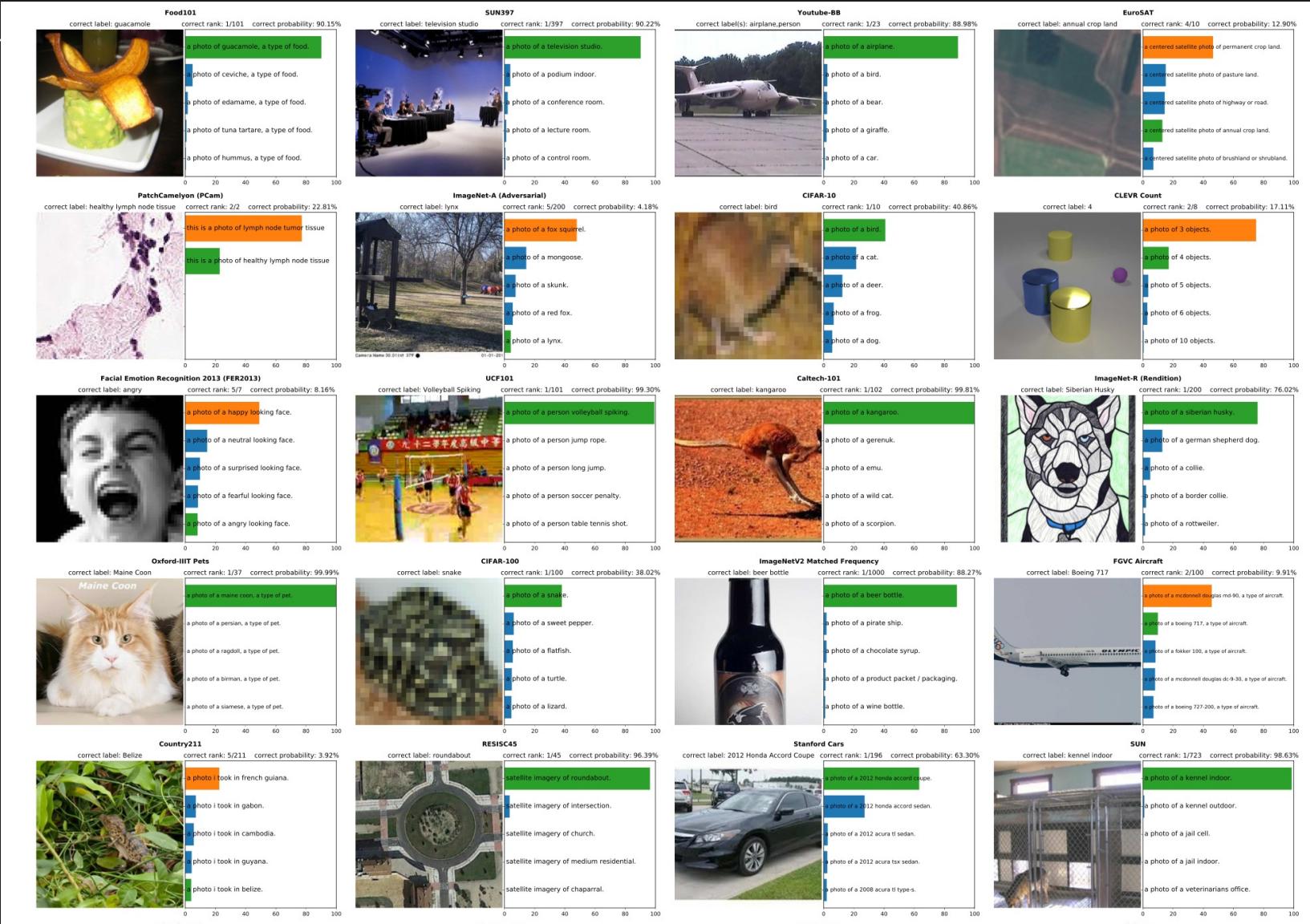
Experiment

7. Broader Impacts

7-3. Future Work

- 연구 과정 초기에 모델의 잠재적으로 유익한 다운스트림 사용 사례를 식별하여, 다른 연구자들이 응용 분야를 생각할 수 있도록 돋는다.
- 중요한 민감성과 많은 사회적 이해관계자가 관련된 작업을 표면화하여 정책 입안자들이 개입할 필요성을 제기할 수 있다.
- 모델의 편향성을 더 잘 특성화하여, 다른 연구자들에게 우려되는 영역과 개입이 필요한 영역을 경고한다.
- CLIP과 같은 시스템을 평가하기 위한 테스트 스위트를 만들어, 개발 주기 초기 단계에서 모델의 능력을 더 잘 특성화할 수 있도록 한다. 잠재적인 실패 모드와 추가 작업이 필요한 영역을 식별한다.

Experiment



Experiment

