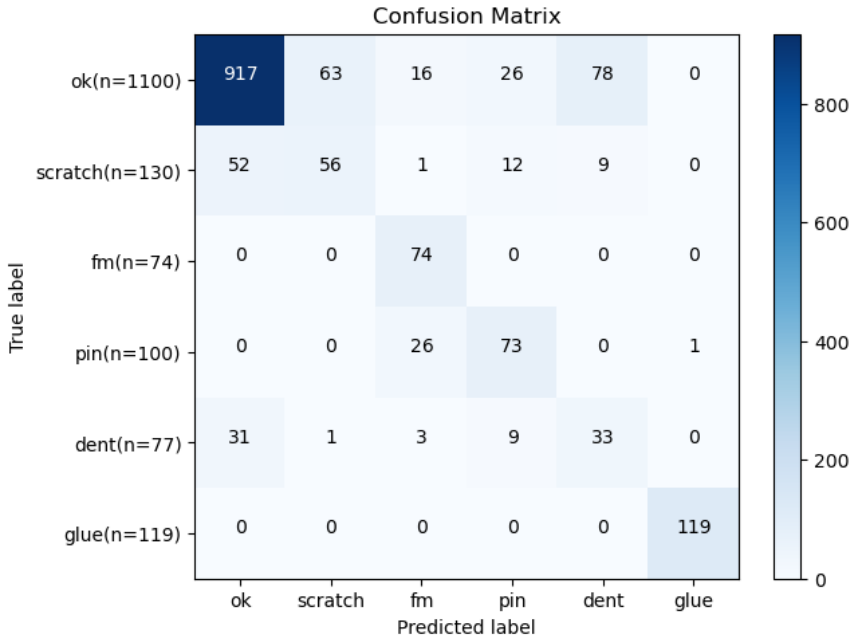
코그넥스 추가실험 & 제안3

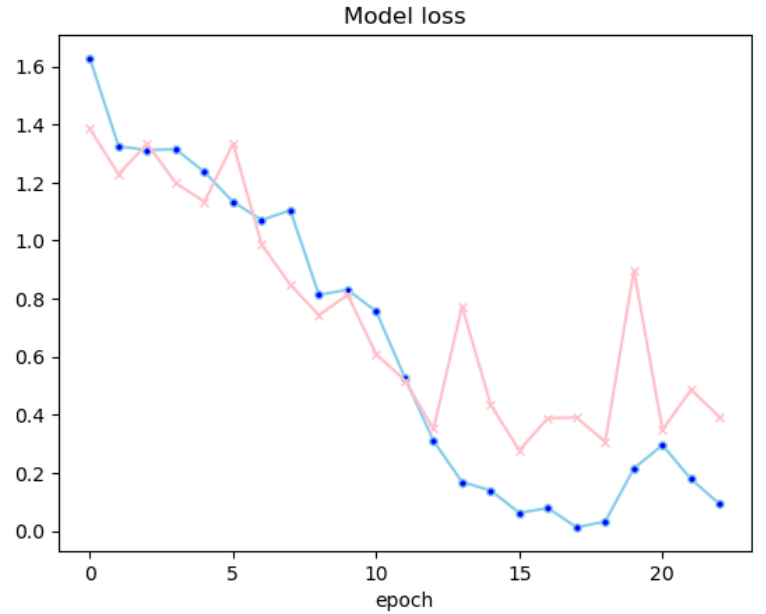
Baseline 실험 체계화

1-1) Train : No augmentation / Test : All of dataset / Model : resnet50

- accuracy of 1600 test images: 79.5% / macro f1 : 0.68 => weighted sampler 사용 전보다 0.3만큼 비약적 상승

	precision	recall	f1-score	support
0	0.92	0.83	0.87	1100
1	0.47	0.43	0.45	130
2	0.62	1.00	0.76	74
3	0.61	0.73	0.66	100
4	0.28	0.43	0.34	77
5	0.99	1.00	1.00	119
accuracy			0.80	1600
macro avg	0.65	0.74	0.68	1600
weighted avg	0.82	0.80	0.80	1600



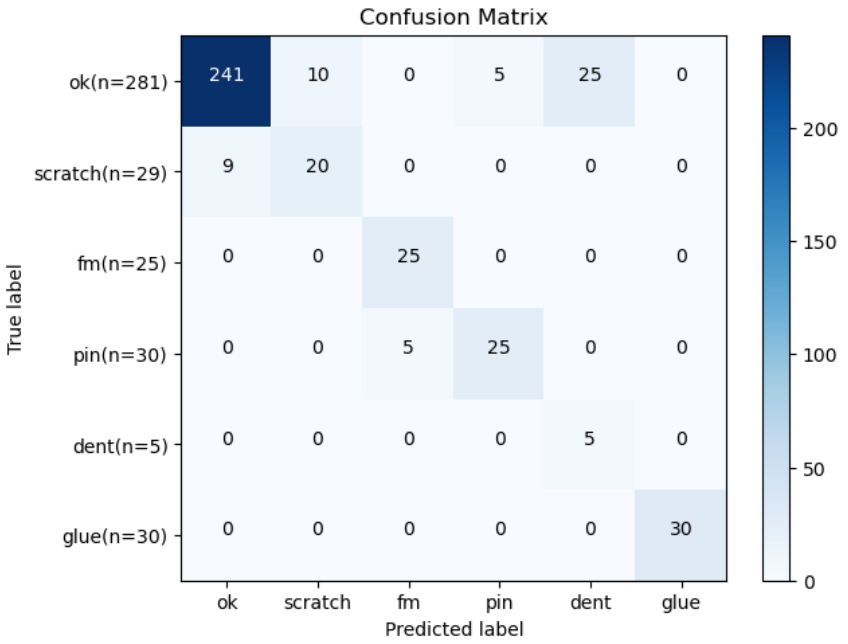


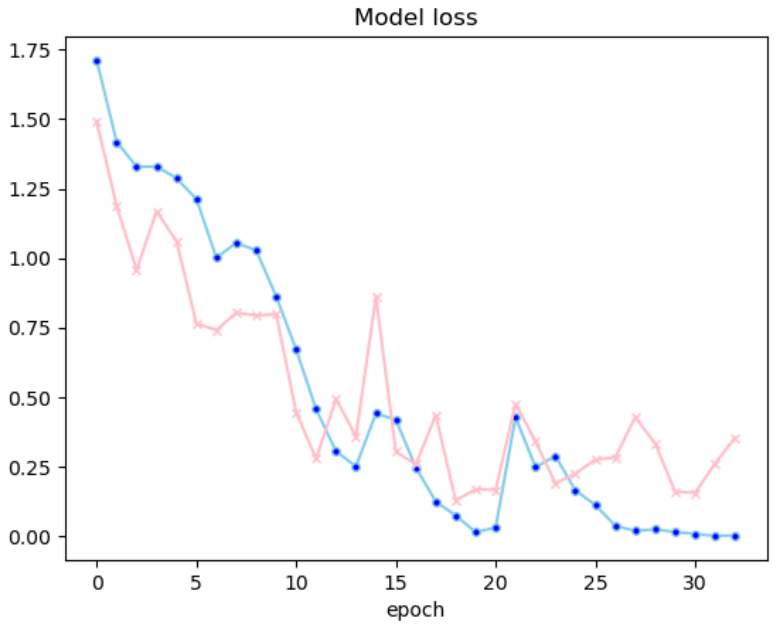
Baseline 실험 체계화

1-2) Train : No augmentation / Test : Only repeat / Model : resnet50

- accuracy of 400 test images: 86.5% / macro f1 : 0.77 => 생각보다 upper bound로써 높은 성능은 아니지 않나 판단이 됨

	precision	recall	f1-score	support
0	0.96	0.86	0.91	281
1	0.67	0.69	0.68	29
2	0.83	1.00	0.91	25
3	0.83	0.83	0.83	30
4	0.17	1.00	0.29	5
5	1.00	1.00	1.00	30
accuracy			0.86	400
macro avg	0.74	0.90	0.77	400
weighted avg	0.92	0.86	0.88	400



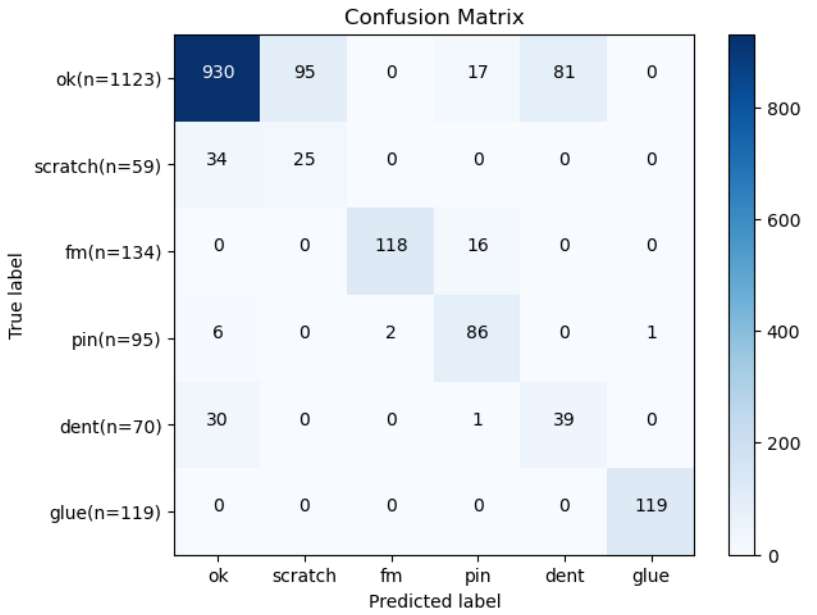


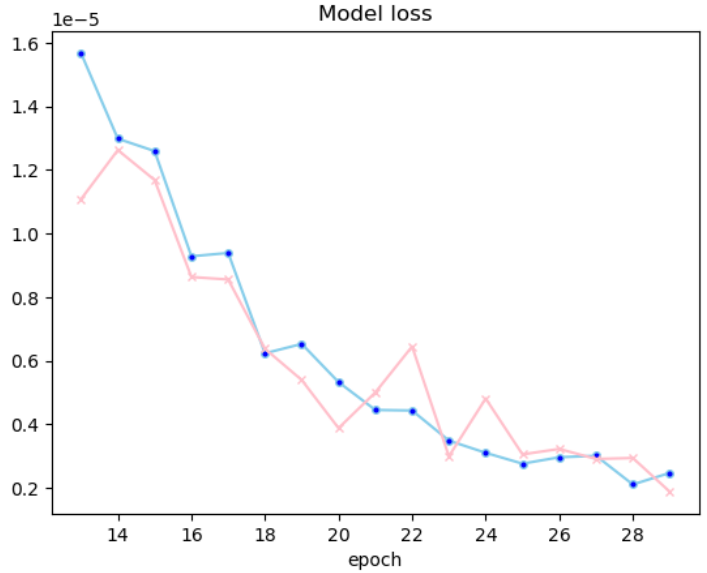
Baseline 실험 체계화

2-1) Train : Offline augmentation / Test : All of dataset / Model : resnet50

- accuracy of 1600 test images: 81.76% / macro f1 : 0.72

	precision	recall	f1-score	support
0	0.93	0.83	0.88	1123
1	0.21	0.42	0.28	59
2	0.98	0.88	0.93	134
3	0.72	0.91	0.80	95
4	0.33	0.56	0.41	70
5	0.99	1.00	1.00	119
accuracy			0.82	1600
macro avg	0.69	0.77	0.72	1600
weighted avg	0.87	0.82	0.84	1600



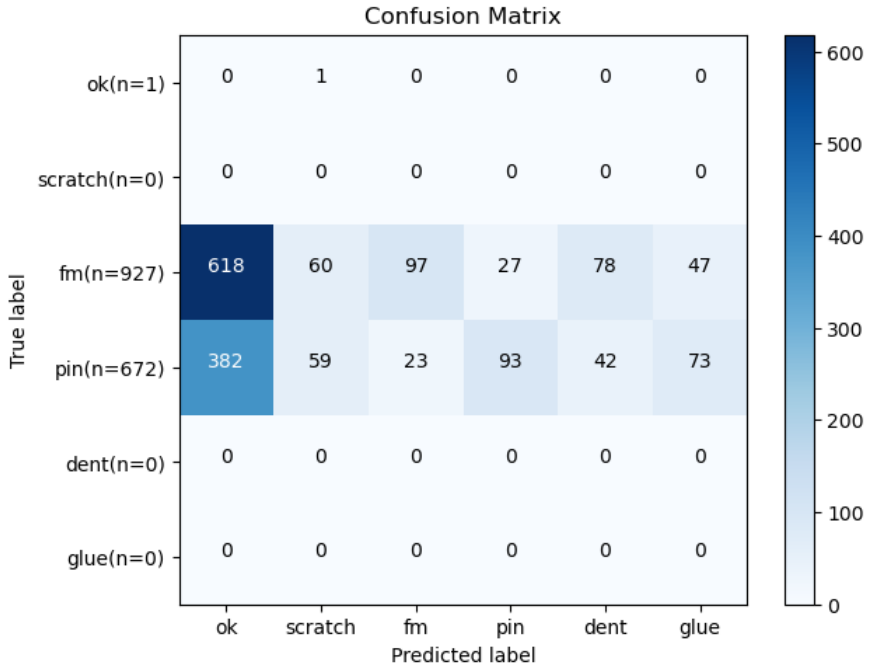


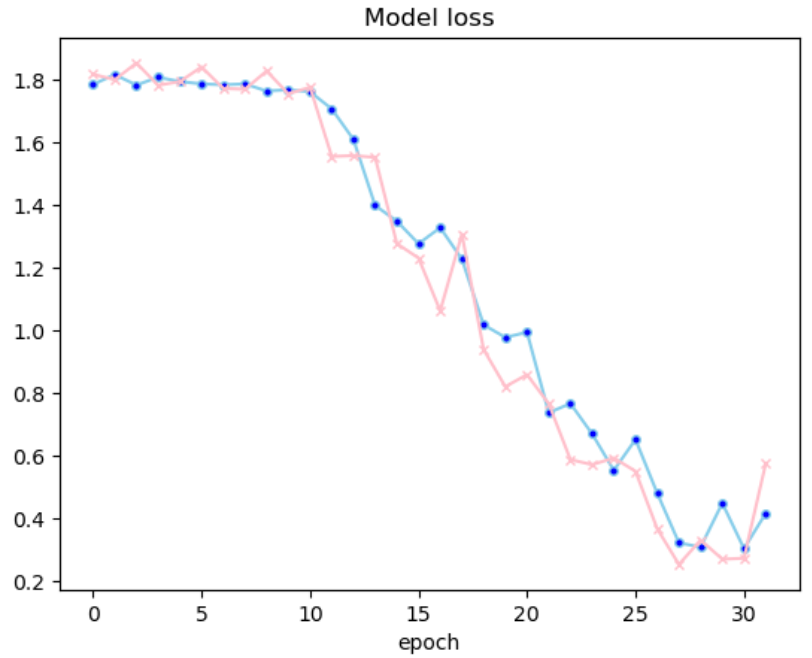
Baseline 실험 체계화

2-2) Train : Online augmentation / Test : All of dataset / Model : resnet50

- accuracy of 1600 test images: 11.875% / macro f1 : 0.07 => offline과 같은 방식을 전부 한꺼번에 도입했더니 원본 데이터에 큰 손상이 온것이라 판단됨

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.00	0.00	0.00	0
2	0.81	0.10	0.19	927
3	0.78	0.14	0.23	672
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
accuracy			0.12	1600
macro avg	0.26	0.04	0.07	1600
weighted avg	0.79	0.12	0.21	1600



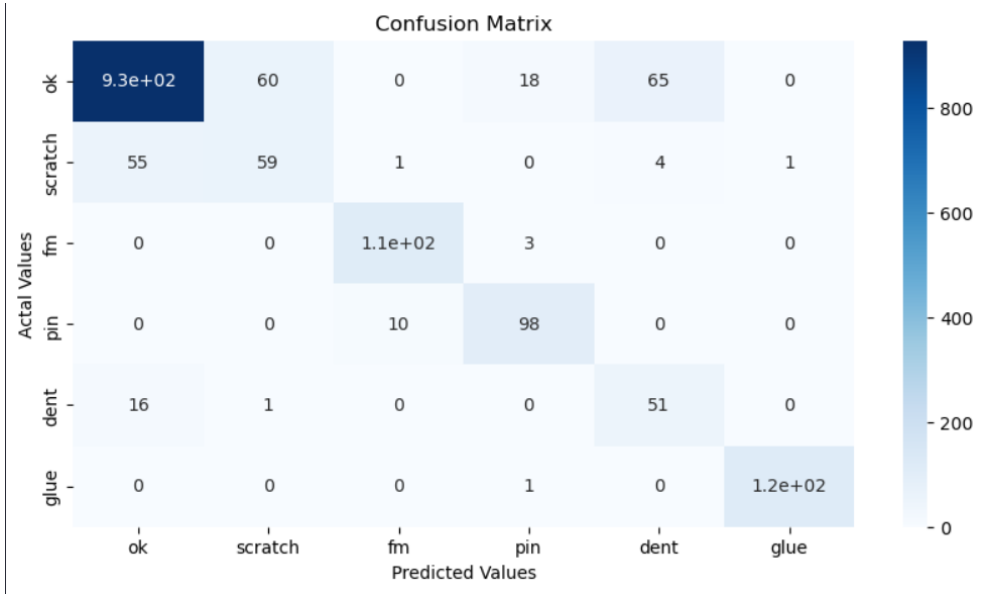


Baseline 실험 체계화

3) Train : Offline augmentation / Test : All of dataset / Model : DecAug

- accuracy of 1600 test images: 85.31% / macro f1 : 0.75

	precision	recall	f1-score	support
0	0.91	0.89	0.90	1017
1	0.60	0.65	0.63	110
2	0.66	0.96	0.78	82
3	0.84	0.69	0.76	146
4	0.45	0.44	0.44	123
5	1.00	0.98	0.99	122



Baseline 실험 체계화

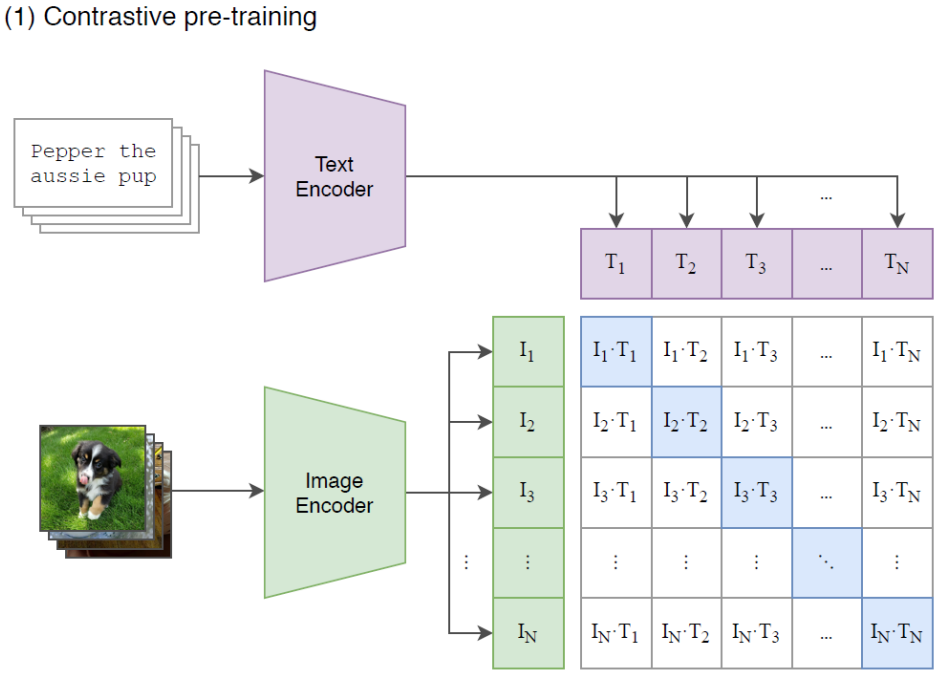
Result

Train / Test / Model	No aug / All condition / resnet50	No aug / Only repeat / resnet50	Offine Aug/ All condition / resnet50	Online aug / All Condition / resnet50	Offine Aug/ All condition / DecAug
Accuracy	79.5%	86.5%	81.76%	11.88%	85.31%
Macro F1 score	0.68	0.77	0.72	0.07	0.75

CLIP 활용 DG

CLIP : 텍스트에 포함된 supervision으로 학습하는 모델

- 1개의 batch는 N개의 (image, text) 쌍으로 구성
- image와 text를 하나의 공통된 space로 보냄
  - Backbone(resnet or VIT / bert) 사용해 feature vector를 뽑은 다음 ,
    모델의 weight(세타)를 곱해 벡터로 만든 다음 l2 norm을 사용해 벡터로 만듦
- Positive pair에서의 유사도(cosine similarity)는 최대화 &
  negative pair에서의 유사도는 최소화 하도록 CE loss 사용



CLIP 활용 DG

CLIP : 텍스트에 포함된 supervision으로 학습하는 모델

- 1개의 batch는 N개의 (image, text) 쌍으로 구성
- image와 text를 하나의 공통된 space로 보냄
  - Backbone(resnet or VIT / bert) 사용해 feature vector를 뽑은 다음 , 모델의 weight(세타)를 곱해 벡터로 만든 다음 l2 norm을 사용해 벡터로 만듦
- Positive pair에서의 유사도(cosine similarity)는 최대화 & negative pair에서의 유사도는 최소화 하도록 CE loss 사용

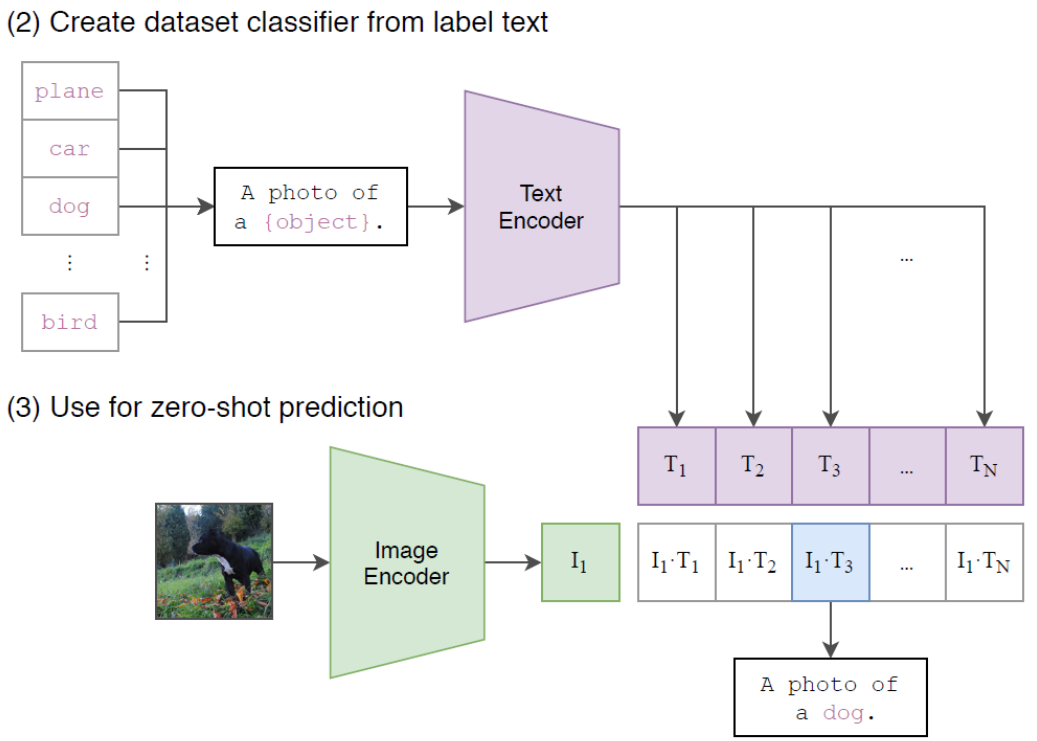
```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned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

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

CLIP 활용 DG

Zero-Shot Transfer CLIP

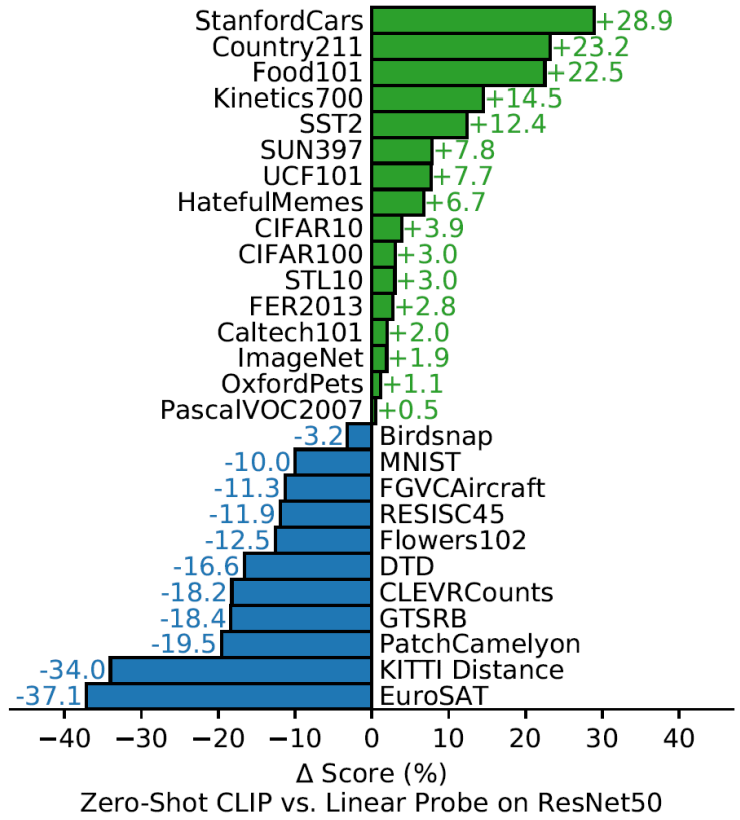
- 이미지가 주어지면 데이터셋의 모든 class와의 (image, text) 쌍에 대해 유사도를 측정하고 가장 그럴듯한(probable) 쌍을 출력
- 각 class name을 "A photo of a {class}." 형식의 문장으로 바꾼 뒤, 주어진 이미지와 유사도를 모든 class에 대해 측정
- 여기서 class는 처음 보는(학습 데이터에 존재하지 않는) class 또한 포함



CLIP 활용 DG

Zero-Shot Transfer CLIP

- 이미지 분류 문제에서 Visual N-grams 방식보다 실험한 3개의 데이터셋 모두에서 zero-shot 성능이 훨씬 뛰어남
- GPT-3에서 다룬 것과 비슷하게, 각 task에 맞는 prompt text를 적절히 선택해 주면 분류하는 데 더 효과적
- Stanford Cars, Food101과 같은 fine-grained task에서는 성능이 크게 앞선다
- 상당히 특수하거나 복잡한 경우 CLIP의 성능은 baseline보다 많이 낮은 경우도 존재(EuroSAT, RESISC45)



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

CLIP 활용 DG

Context-Aware Robust Fine-Tuning

- CLIP을 활용한 zero shot 기법을 사용하면 OOD에서는 좋은 성능을 보이지만 ID에서는 그저 그런 성능을 보임
- Fine-tuned CLIP(context-unaware) 에서는 이와 반대
- 두 모델을 동시에 사용해서 ID와 OOD의 성능을 향상시키자! =>CAR-FT(본 논문이 제안하는 모델)

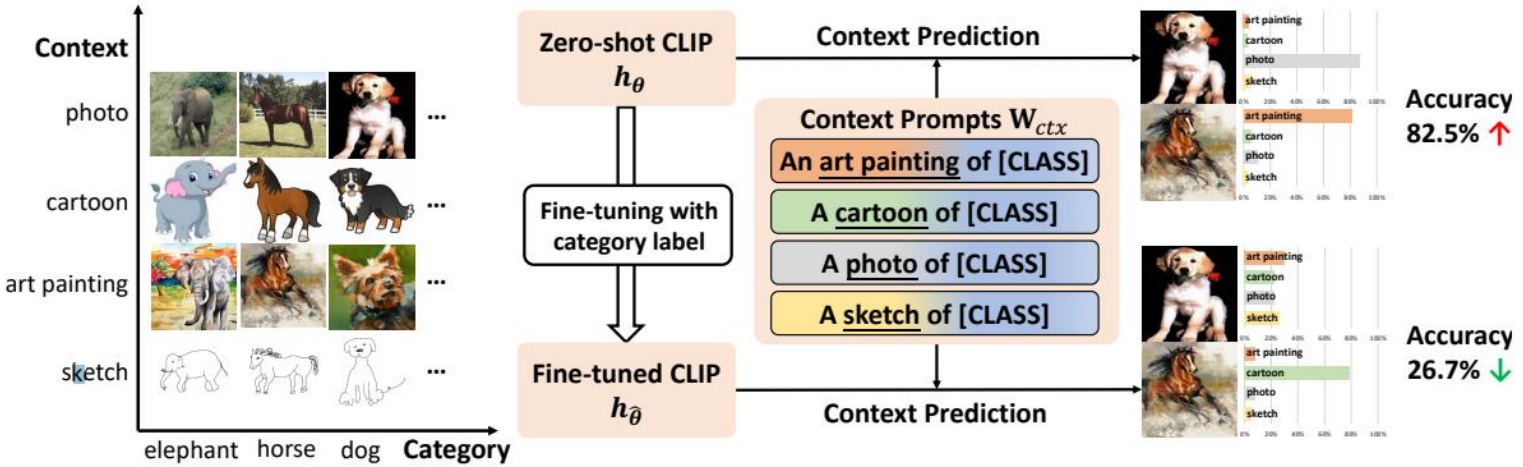
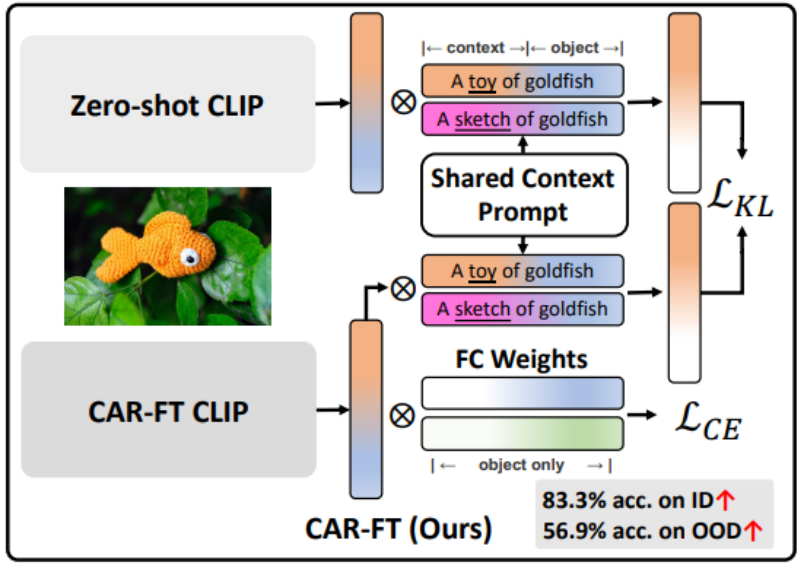


Fig. 1 We use PACS dataset to create a task of recognizing context from input images. After fine-tuning CLIP with category label, the context recognition accuracy dropped from 82.5% to 26.7%.

CLIP 활용 DG

Context-Aware Robust Fine-Tuning

- Train 1-1 단계(Zero-shot clip)
  - 기존 zero shot 모델과 동일하게 인풋 이미지에 대해 prompt text을 활용한 여러 class에 대한 확률 분포를 구함
  - 이때, prompt class는 기존과 다르게 “a {context} of {label}” 로 domain 정보 포함
  - 학습을 할 때 zero shot 모델은 Freeze

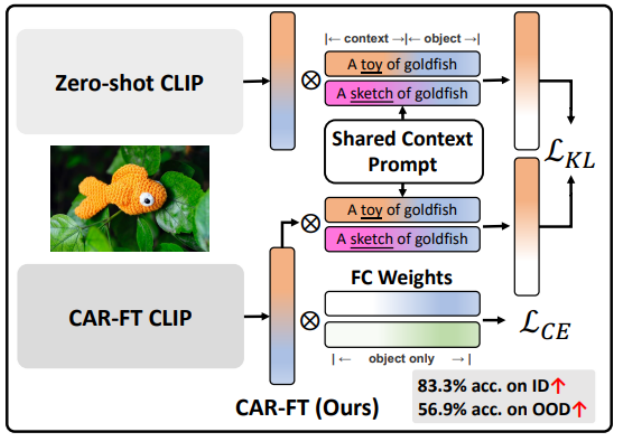


P. **(Bottom left)** Fine-tuned CLIP with linear classifier

CLIP 활용 DG

Context-Aware Robust Fine-Tuning

- Train 1-2 단계 (CAR-FT clip)
  - 기존 zero shot 모델과 동일하게 인풋 이미지에 대해 prompt text을 활용한 여러 class에 대한 확률 분포를 구함
  - CAR-FT모델은 weight를 fine tunning 진행
  - CAR-FT 모델은 2개의 output(prompt text을 활용한 여러 class에 대한 확률 분포)을 리턴
    - Output1의 prompt class는 zero-shot model과 동일하게 “a {context} of {label}”
    - Output2의 prompt class는 기존 class와 동일하게 “a photo of {label}”



P. **(Bottom left)** Fine-tuned CLIP with linear classifier

CLIP 활용 DG

Context-Aware Robust Fine-Tuning

- Train 2 단계 (Loss 계산)
  - context-class 형태의 prompt text를 활용한 확률 분포인 Zero-shot clip output과 CAR-FT clip output1 을 Kullback–Leibler divergence 계산
  - Class 형태의 prompt text를 활용한 확률 분포인 CAR-FT clip output2에 대해 정답 class vector와 CE 계산
  - 이 둘을 적당히 잘 합쳐 Loss 함수로 하여 CAR-FT CLIP 모델 재학습 진행

Algorithm 1 Pseudo code of CAR-FT

Input: Pre-trained image encoder h_θ and text encoder g_ϕ; Text prompts $\mathcal{T}$.

Output: Fine-tuned image encoder weights $\hat{\theta}$ and classification weights $\mathbf{W}_f$.

Compute $\mathbf{W}_{cls}$ based on $\mathcal{T}$ using Equation (1)
Compute $\mathbf{W}_{ctx}$ based on $\mathcal{T}$ using Equation (2)
Fix parameters θ, ϕ and $\mathbf{W}_{ctx}$
$\hat{\theta} \leftarrow \theta$, $\mathbf{W}_f \leftarrow \mathbf{W}_{cls}$
for each training steps do
    Sample a mini-batch images x with labels y
    Get reference context distribution of zero-shot model: $p_{ctx}(x;\ \theta) \leftarrow \text{Softmax}(\mathbf{W}_{ctx}^\top h_\theta(x))$
    Get predicted context distribution of fine-tuned model: $p_{ctx}(x;\ \hat{\theta}) \leftarrow \text{Softmax}(\mathbf{W}_{ctx}^\top h_{\hat{\theta}}(x))$
    Compute KL divergence loss $\mathcal{L}_{KL} \leftarrow \mathbf{KL}[p_{ctx}(x;\ \theta) \| p_{ctx}(x;\ \hat{\theta})]$
    Compute classification loss $\mathcal{L}_{CE}(\mathbf{W}_f^\top h_{\hat{\theta}}(x), y)$
    $\mathcal{L} \leftarrow \mathcal{L}_{CE} + \alpha \mathcal{L}_{KL}$
    Update parameters $\hat{\theta}$, $\mathbf{W}_f$ for minimizing $\mathcal{L}$

end for

CLIP 활용 DG

Context-Aware Robust Fine-Tuning

- Experiment
  - Imagenet 데이터에 대한 distribution shift과 관련된 여러 실험(clip zero shot, model soup, weight-space ensemble)을 했을 때 가장 좋은 성능을 보임

Table 1 Top@1 accuracy of compared methods on ImageNet and its derived distribution shifts. Avg. shifts presents the mean accuracy among five distribution shifts. We use ViT-B/16 as basic backbone.

		Distribution shifts						
	IN	IN-V2	IN-R	IN-Sketch	ObjectNet	IN-A	Avg. shifts	Avg. all
CLIP Zero-shot	68.3	61.9	77.6	48.3	29.8	50.1	53.5	56.0
Fine-tuning Only Methods								
FT	81.0	70.9	54.7	42.1	26.6	31.3	45.1	51.1
TP-FT	81.2	70.7	65.0	44.9	27.4	35.3	48.7	54.1
LP-FT	81.7	71.6	72.9	48.4	28.2	49.1	54.0	58.7
CAR-FT (Ours)	**83.3**	**74.0**	**75.4**	**53.0**	**32.6**	**49.5**	**56.9**	**61.3**
Combined with Weight-Space Ensemble								
WiSE-FT (opt. α)	81.7	72.7	78.8	53.2	33.4	52.2	58.1	62.0
+ CAR-FT	**82.1**	**73.3**	**79.2**	**54.5**	**33.8**	**53.6**	**58.9**	**62.8**
Greedy Model Soups	84.8	75.1	74.1	53.8	30.3	48.1	56.3	61.0
+ CAR-FT	**85.0**	**75.8**	**74.4**	**54.6**	**31.6**	**48.9**	**57.1**	**61.7**
Uniform Model Soups	83.4	74.7	76.8	54.6	31.5	50.0	57.5	61.8
+ CAR-FT	**83.9**	**75.1**	**77.3**	**55.5**	**32.0**	**51.1**	**58.2**	**62.5**

CLIP 활용 DG

Context-Aware Robust Fine-Tuning

- Experiment
  - CLIP을 활용한 DomainBed내의 기존 기법들, 혹은 RegNetY(EfficientNet 상위호완)를 활용한 기법들에 비해 더 좋은 성능을 보임

Table 2 Comparison with domain generalization methods on DomainBed. The accuracy reported are averaged on three trials.

	PACS	VLCS	OfficeHome	TerraInc	DomainNet	Avg.
Previous SOTA using RegNetY-16GF						
MIRO (Cha et al, 2022)	**97.4**	79.9	80.4	58.9	53.8	74.1
EoA (Arpit et al, 2021)	95.8	81.1	83.9	61.1	60.9	76.6
MIRO+SWAD (Cha et al, 2022, 2021)	96.8	81.7	83.3	**64.3**	60.7	77.3
ViT-B/16 Pre-trained by CLIP						
ERM	93.7	82.7	78.5	52.3	53.8	72.2
MIRO (Cha et al, 2022)	95.6	82.2	82.5	54.3	54.0	73.7
DPL (Zhang et al, 2022a)	97.3	84.3	84.2	52.6	56.7	75.0
CAR-FT (Ours)	96.8	**85.5**	**85.7**	61.9	**62.5**	**78.5**