InfoCSE:Information-aggregated Contrastive Learning

Xing Wu, Chaochen Gao ,Zijia Lin ,Jizhong Han ,Zhongyuan Wang ,Songlin Hu
EMNLP 2022

발제자: 김한성
23-04-07

Abstract

sentence embedding에 대한 Contrastive Learning에 대한 연구가 활발해졌다.

하지만 Contrastive Learning의 가정은 좋은 문장을 reconstruct하는 과정까지 포함해야한다.

reconstruct를 잘하기 위한 방법인 information-aggregated CSE, InfoCSE를 제안한다.

InfoCSE는 CLS의 representation과 denser sentence information (MLM)를 결합한 network구조를 갖고 있다.

이는 결국 STS에서 SOTA인 SimCSE대비 2.6%의 향상을 얻었다.

what is reconstruct

Sentence embedding is the process of converting a sentence into a numerical vector representation, which can be used in various natural language processing tasks such as sentiment analysis, text classification, and text clustering.

Reconstruction in sentence embedding refers to the process of reconstructing the original sentence from its vector representation. This is done by using a decoding function that maps the vector back to its original sentence. The quality of the reconstruction is an important factor in evaluating the performance of a sentence embedding model.

In practice, reconstruction is often used as a way to evaluate the quality of the sentence embedding model. The model is trained to minimize the difference between the original sentence and its reconstructed version, which helps to ensure that the embedding captures the most important information in the sentence.

by the MLM model, we can create new sentences that are similar to the original ones, but with some words replaced by their predicted values. This can help to increase the diversity of the training data and improve the performance of the MLM model.

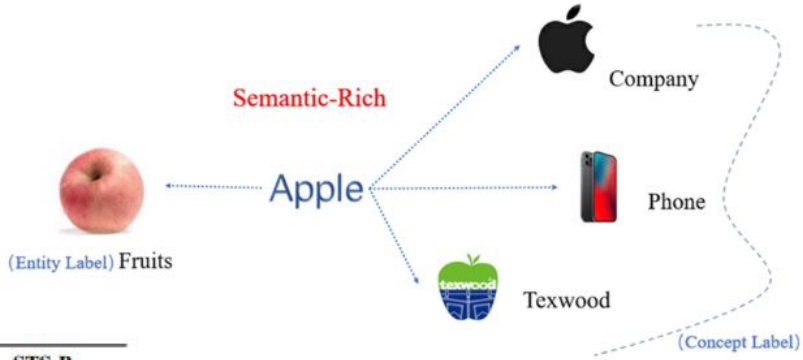
Introduction

- SimCSE의 기여점
  - Contrastive Learning을 통해 similar와 dissimilar에 대한 rich semantic information을 얻어냄

- SimCSE의 한계점
  - MLM 증강 학습에서 성능이 좋지 않음
  - 이는 reconstruct를 못한다고 볼 수 있다.

▼ rich semantic information

Rich semantic information refers to the depth and complexity of meaning contained in a piece of language, including its context, syntax, semantics, and pragmatics.



Model	STS-B
SimCSE-BERT$_{base}$	86.2
w/ MLM	
$\lambda = 0.01$	85.7
$\lambda = 0.1$	85.7
$\lambda = 1$	85.1

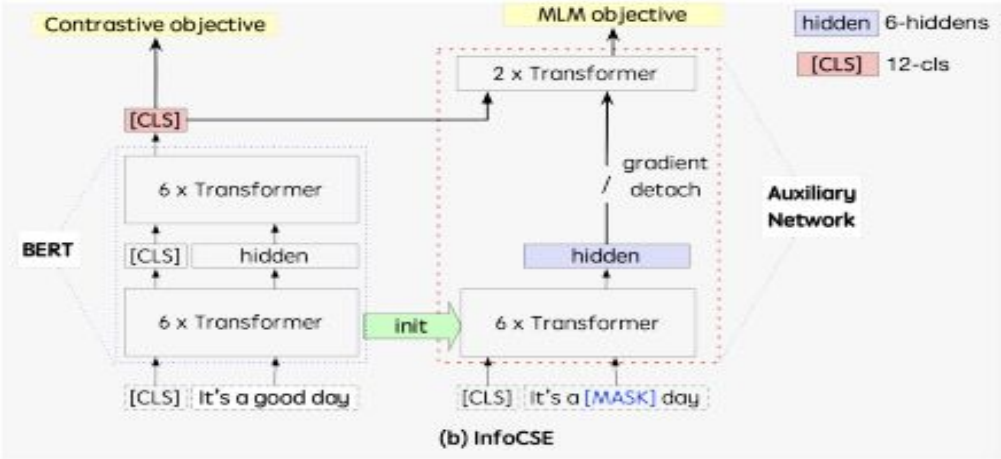
Table 1: Table from SimCSE (Gao et al., 2021). The masked language model (MLM) objective brings a consistent drop to the SimCSE model in semantic textual similarity tasks. "w/" means "with", λ is the balance hyperparameter for MLM loss.

https://www.researchgate.net/figure/Label-semantic-information-rich_fig1_355201848

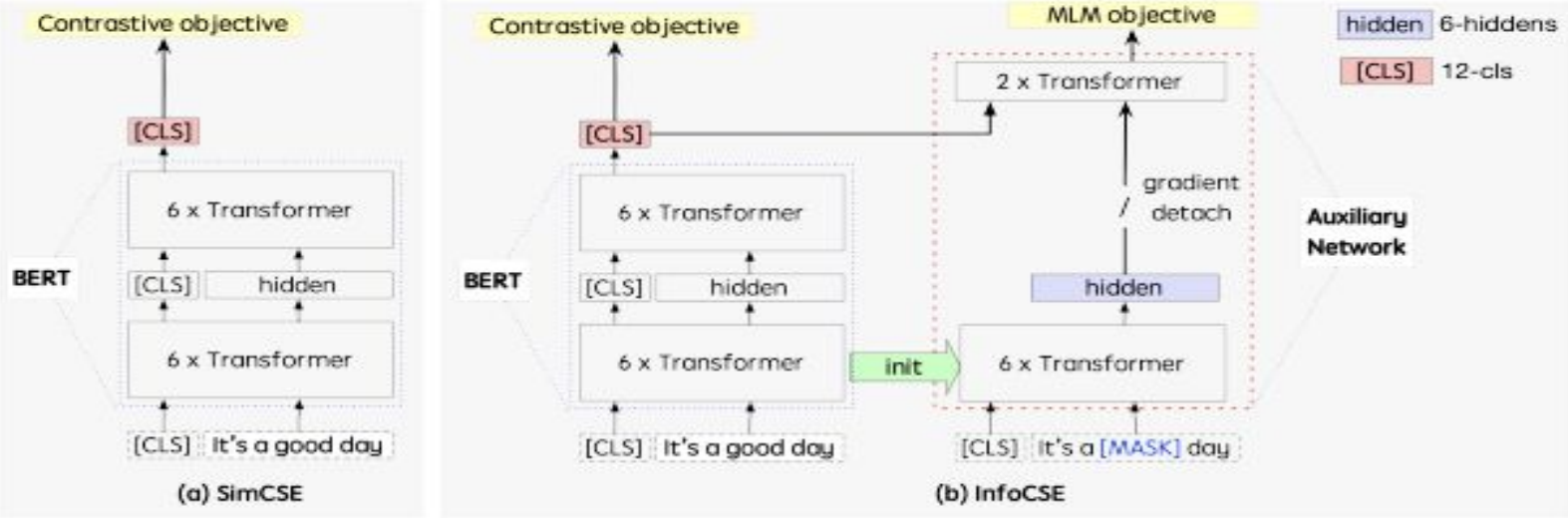
Introduction

- 저자는 여기서 MLM Objective에 대해 over-update하기 때문이라 주장
- MLM Objective와 CL Objective가 같이 update되는 것이 이유
- 이는 reconstruction task를 잘 다루지 못함

대조학습 기반 reconstruction task를 잘 다루는 information-aggregated contrastive learning framework를 제안



Introduction

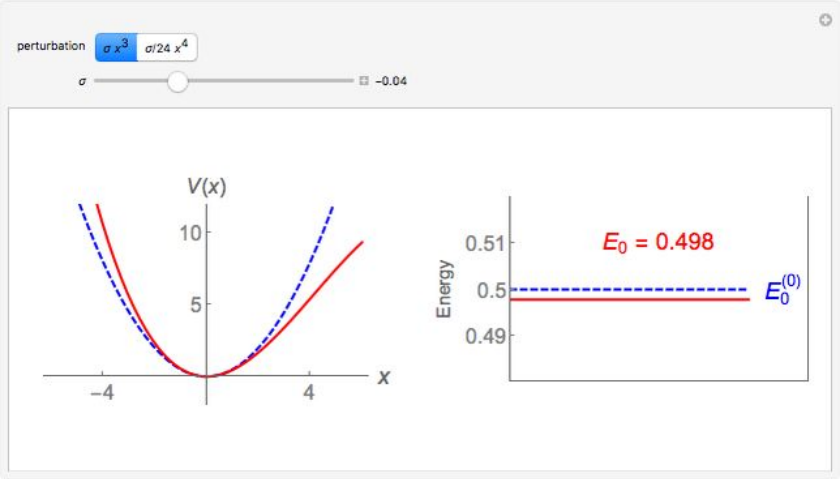


> So the MLM task is forced to rely more on the 12-cls embedding, encouraging the 12-cls embedding to encode richer semantic information

Introduction

What is perturbation?

수학과 물리학에서 섭동 이론 (perturbation theory)은 해석적으로 풀 수 없는 문제의 해를 매우 작다고 여길 수 있는 매개변수들의 테일러 급수로 나타내는 이론이다. 매개변수들이 매우 작으므로, 급수의 유한개의 항을 계산하여 근사적인 해를 얻을 수 있다



- The gradient update of the auxiliary network is only back-propagated to the BERT network through the *12-cls* embedding. Compared to perfTorming the MLM task directly on the BERT, the effect of gradient updates using the *12-cls* embedding will be much smaller and will not cause large perturbations to the contrastive learning task.

https://demonstrations.wolfram.com/PerturbationTheoryAppliedToTheQuantumHarmonicOscillator/

Backgrounds

BERT’s MLM Objective

- MLM은 랜덤하게 토큰들의 부분집합에 mask를 씌워 모델이 masked된 토큰을 예측

$$origin : x$$
$$masked : \hat{x}$$

$$L^{mlm} = \sum_{j \in masked} CE(H^j W, x^j)$$

H : hidden_state -> batch, tokens, hidden_state

W : projection -> hidden_state : vocab_size

x^j : j th tokens

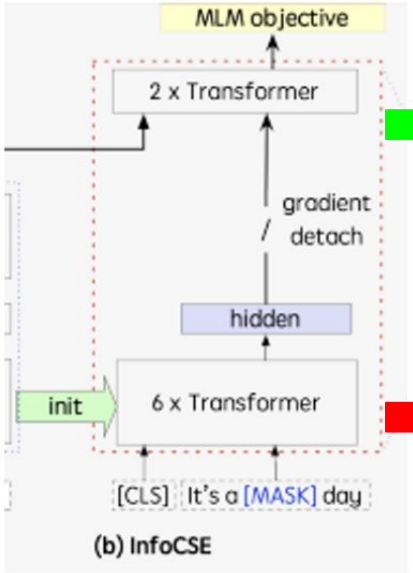
InfoCSE:Information-aggregated Contrastive Learning

Pre-training of The Auxiliary Network

$$L^{mlm} = \sum_{j \in masked} CE(H^j W, x^j)$$

$$L^{aux} = \sum_{j \in masked} CE(\tilde{H}^j W, x^j)$$

$$\mathcal{L}^{\text{pretrain}} = \mathcal{L}^{\text{aux}} + \mathcal{L}^{\text{mlm}}$$



train

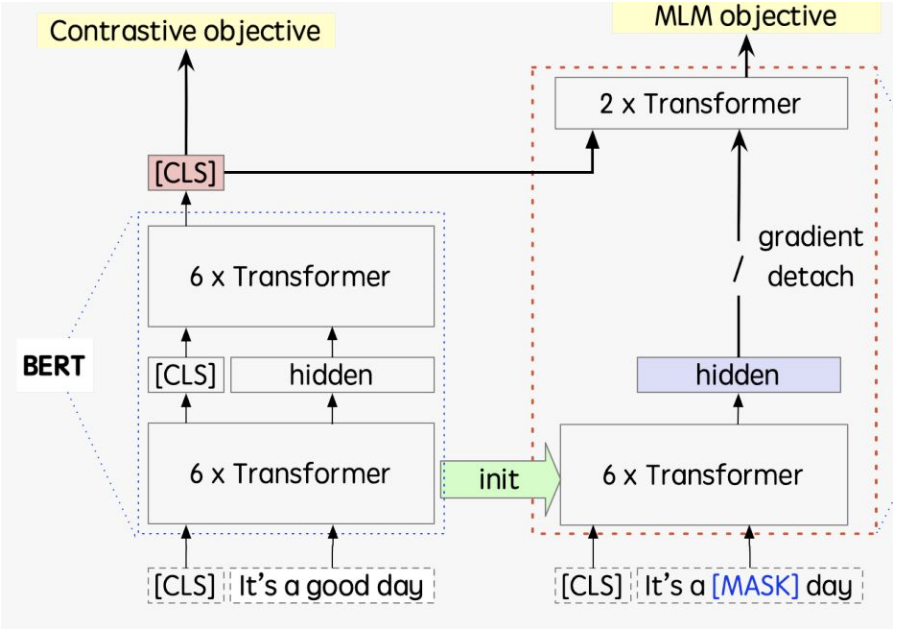
frozen

Pre-training of The Auxiliary Network

$$\tilde{H} = [h, M^{>0}]$$

The output projection matrix W is shared between the two MLM loss

InfoCSE:Information-aggregated Contrastive Learning



Joint Training of MLM and Contrastive Learning

$$\mathcal{L}^{\mathrm{cl}} = -\log \frac{\exp(\mathrm{sim}\,(h, h^{+})\,/\tau)}{\sum_{h' \in B} \exp(\mathrm{sim}\,(h, h'^{+})\,/\tau)}$$

$$L^{aux} = \sum_{j \in masked} CE(\tilde{H}^{j} W, x^{j})$$

$$\mathcal{L}^{\mathrm{joint}} = \mathcal{L}^{\mathrm{cl}} + \mathcal{L}^{\mathrm{aux}} * \lambda$$

Experiment

- STS
  - STS-B는 train과 dev셋이 존재하기 때문에 학습, 평가는 STS EVAL set에 진행
  - Spearman correlation coefficient
- Open-Domain Retriever
  - BEIR 18개 벤치마크에서 9개 선택
  - fact checking, citation prediction, duplicate question retrieval, parameter retrieval, news retrieval, question answering, tweet retrieval, biomedical IR, entity retrieval

- Pretraining Details
  - train 8 epochs
  - Adam optimizer with learning rate = 1e − 4
  - global batch size = 1024 on 8 Nvidia V100 GPUs
  - λ mlm = 1.0
- jointly training
  - Adam optimizer with learning rate = 3e − 5, batch size = 64 on a single Nvidia 3090 GPU
  - λ mlm = 0.005

Experiment result

Model	STS12	STS13	STS14	SICK15	STS16	STS-B	SICK-R	Avg.
GloVe embeddings(avg.)	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
$BERT_{base}$(first-last avg.)	39.70	59.38	49.67	66.03	66.19	53.87	62.06	56.70
$BERT_{base}$-flow	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
$BERT_{base}$-whitening	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-$BERT_{base}$ △	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CT-$BERT_{base}$ △	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
$ConSERT_{base}$ ♡	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
$BERT_{base}$-flow◇	63.48	72.14	68.42	73.77	75.37	70.72	63.11	69.57
SG-OPT-$BERT_{base}$ ♠	66.84	80.13	71.23	81.56	77.17	77.23	68.16	74.62
Mirror-$BERT_{base}$ ♯	69.10	81.10	73.00	81.90	75.70	78.00	69.10	75.40
SimCSE-$BERT_{base}$ ♣	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
ESimCSE-$BERT_{base}$ ⋆	**73.40**	83.27	**77.25**	82.66	78.81	80.17	**72.30**	78.27
DiffCSE-$BERT_{base}$ ¶	72.28	84.43	76.47	83.90	80.54	80.59	71.23	78.49
InfoCSE-$BERT_{base}$	70.53	**84.59**	76.40	**85.10**	**81.95**	**82.00**	71.37	**78.85**
$ConSERT_{large}$ ♡	70.69	82.96	74.13	82.78	76.66	77.53	70.37	76.45
$BERT_{large}$-flow◇	65.20	73.39	69.42	74.92	77.63	72.26	62.50	70.76
SG-OPT-$BERT_{large}$ ♠	67.02	79.42	70.38	81.72	76.35	76.16	70.20	74.46
SimCSE-$BERT_{large}$ ♣	70.88	84.16	76.43	84.50	79.76	79.26	73.88	78.41
ESimCSE-$BERT_{large}$ ⋆	**73.21**	85.37	**77.73**	84.30	78.92	80.73	**74.89**	79.31
DiffCSE-$BERT_{large}$ †	72.11	84.99	76.19	85.09	78.65	80.34	73.93	78.76
InfoCSE-$BERT_{large}$	71.89	**86.17**	77.72	**86.20**	**81.29**	**83.16**	74.84	**80.18**

STS

Dataset	SimCSE		ESimCSE		DiffCSE		InfoCSE	
	base	large	base	large	base	large	base	large
trec-covid	0.2750	0.2264	0.2291	0.2829	0.2368	0.2291	**0.3937**	0.3166
nfcorpus	0.1048	0.1356	0.1149	0.1483	0.1204	0.1470	0.1358	**0.1576**
nq	0.1628	0.1671	0.0935	0.1705	0.1188	0.1556	**0.2023**	0.1790
fiqa	0.0985	0.0975	0.0731	**0.1117**	0.0924	0.1027	0.0991	0.1000
arguana	0.2796	0.2078	0.3376	0.2604	0.2500	0.2572	0.3244	**0.4133**
webis-touche2020	**0.1342**	0.0878	0.0786	0.1057	0.0912	0.0781	0.0935	0.0920
quora	0.7375	0.7511	0.7411	0.7615	0.7491	0.7471	0.8241	**0.8268**
cqadupstack	0.1349	0.1082	0.1276	0.1196	0.1197	0.1160	**0.2097**	0.1881
dbpedia-entity	0.1662	0.1495	0.1260	0.1650	0.1537	0.1571	**0.2101**	0.1838
scidocs	0.0611	0.0688	0.0657	0.0796	0.0673	0.0699	0.0837	**0.0859**
climate-fever	**0.1420**	0.1065	0.0796	0.1302	0.1019	0.1087	0.0937	0.0840
scifact	0.2492	0.2541	0.3013	0.2875	0.2666	0.2811	0.3269	**0.3801**
hotpotqa	0.2382	0.1896	0.1213	0.1970	0.1730	0.2068	**0.3177**	0.2781
fever	**0.2916**	0.1776	0.0756	0.1689	0.1416	0.1849	**0.1978**	0.1252
average	0.2197	0.1948	0.1832	0.2135	0.1916	0.2030	**0.2509**	0.2436

Open-Domain Retriever

Ablation Study

The Impact of Auxilary Network

- pretrain없이 학습을 했을 때 table 4를 보면 joint training이 심각하게 떨어진다.
- 즉, pretraining이 joint training과정에서 large gradient oscillations를 avoid 해준다.
  - oscillations : 진동, 즉 over-update를 막아준다.

Model	STS-B
InfoCSE	**85.49**
w/o pre-training	83.73

Table 4: Development set results of STS-B for InfoCSE with or without auxiliary network pre-training. "w/o" denotes without.

Joint Training of MLM and Contrastive Learning

- table 5에 나와있듯이, MLM Loss가 없는 경우 SimCSE 보다 성능이 좋지 않았더라.

Model	STS-B
InfoCSE	**85.49**
w/o MLM loss	82.45
w/o Contrastive loss	40.00

Table 5: Development set results of STS-B for InfoCSE variants, where we vary the objective. "w/o" denotes without.

The Impact of Gradient Detach in Joint Training

Model	STS-B
InfoCSE	**85.49**
w/o gradient detach	84.41

Table 6: Development set results of STS-B for InfoCSE with or without gradient detach. "w/o" denotes without.

The Impact of Mask Rate

Mask Rate	*10%*	*15%*	*20%*	*25%*
STS-B	83.97	84.62	84.74	85.08
Mask Rate	*30%*	*35%*	*40%*	*45%*
STS-B	84.19	84.70	**85.49**	84.13

Table 7: Development set results of STS-B when we vary the mask rate.

Analysis

sentence embeddings. The difference is that RTD is a discriminative objective, while MLM is a generative objective. Therefore, we further explore whether these two different auxiliary objectives can coexist. Specifically, we simultaneously apply the

Model	STSB	Avg.
SimCSE	82.45	76.25
w/ RTD (DiffCSE)	84.56	78.27
w/ MLM (InfoCSE)	85.49	78.49
w/ RTD + MLM	**85.83**	**79.39**

Table 12: The comparison of the improvement brought by different auxiliary objectives to SimCSE. "w/" denotes without. "STS-B" denotes the best result on the STS-B development set. "Avg." denotes the corresponding average result on 7 semantic textual similarity (STS) test sets.

Momentum Contrast for Unsupervised Visual Representation Learning

Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick
Facebook AI Research (FAIR)
2020 CVPR

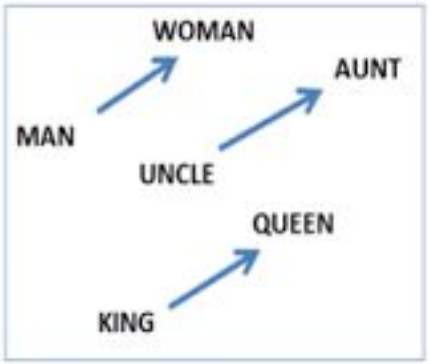
발제자: 김한성
23-04-07

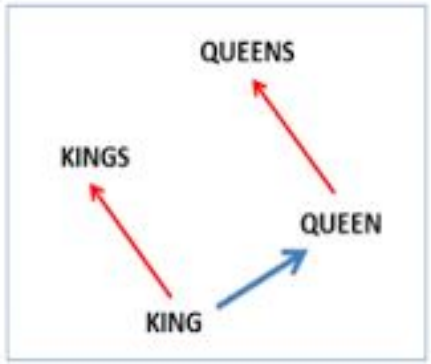
Abstract

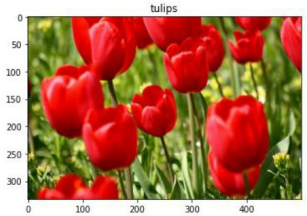
- 자연어 처리는 비지도 학습 대조학습이 좋은 성과를 보였으나 이미지 분야에서는 여전히 지도학습기반 대조학습이 지배적
- 본 저자는 이미지는 텍스트 대비 continous raw signal 이 존재한다고 주장
- 이를 위해서는 대용량 데이터를 학습 데이터로 활용할 수 있어야한다고 주장
- 이에 MoCo를 주장
- detection/segmentation task인 PASCAL VOC COCO에서 지도학습 기반 알고리즘 보다 좋은 성능을 보임

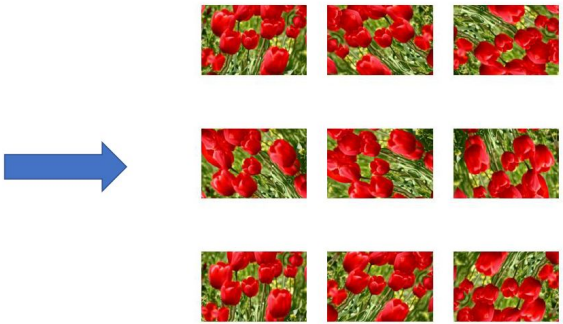
Introduction

continuous raw signal 이란?









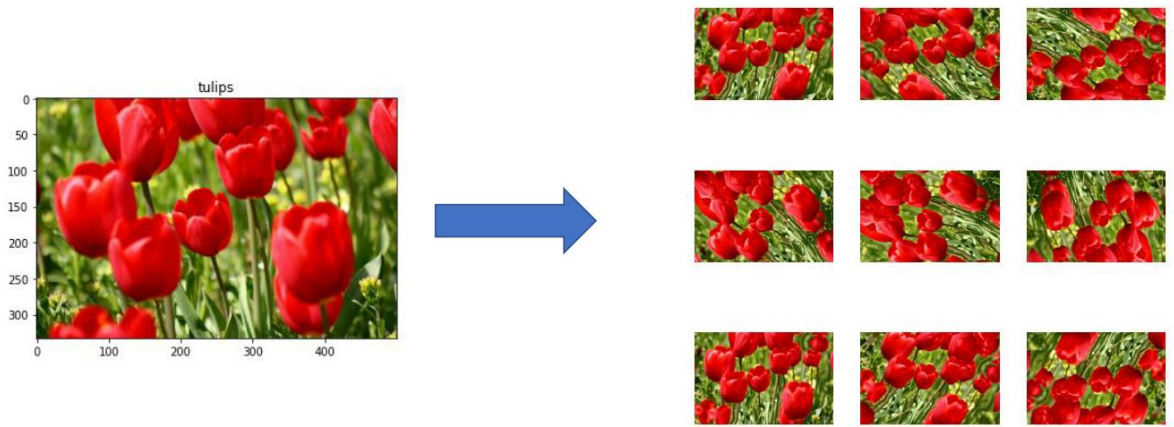
자연어 : 의미 단위로 word or sub-word가 인코딩의 대상이 되므로 이산적이라고 볼 수 있다고 주장

이미지 : 의미 단위가 아닌 이미지의 위치, 각도 등의 단위로 인코딩 되기 때문에 의미를 알기 위한 continous raw signal이 존재

https://hoya012.github.io/blog/Image-Data-Augmentation-Overview/
https://ratsgo.github.io/natural%20language%20processing/2017/03/08/word2vec/

Introduction

For good representation



결국 연속성을 파악하기 위해서는 기존 이미지에서 pos pair를 증강한다 하더라도 많은 양의 데이터 학습이 필요함.

https://hoya012.github.io/blog/Image-Data-Augmentation-Overview/

Introduction

Momentum Contrastive

1. 데이터의 양이 많아야 한다. (similar pair or dissimilar pair의 수)
2. consistent 해야 한다.

detection or segmentation인 7개 down-stream task에서 비지도 학습 기반이 지도학습을 능가하는 결과

Related Work

비지도 학습과 자기 지도 학습은 2가지 양상으로 발전하고 있고 본 논문에서는 loss를 구성하는 것으로 발전시켰다.

Contrastive loss

- similarities sample pair → pos ⬆️ neg ⬇️

Contrastive Learning

- 대조학습 정의 자체가 하나의 인스턴스를 더 정확히 표현하기 위한 방법론으로 유사한 것과 유사하지 않은 것에 대한 feature space를 변형해가며 학습하는 것이다.
- Unlabeled data에 대해서 더 좋은 representation을 줄 수 있는 것이다.

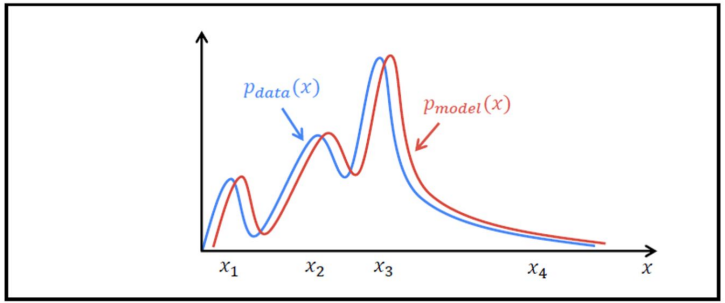
$$l_i = -log \frac{e^{sim(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} e^{sim(h_i, h_j^+)/\tau}}$$

Adversarial loss

- difference between 확률 분포
- representation Learning
  - GAN and noise-contrastive estimation(NCE)

The goal of the generative model is to find a $p_{model}(x)$ that approximates $p_{data}(x)$ well.

$p_{data}(x)$: *Distribution of actual images*



https://kjhov195.github.io/2020-03-09-generative_adversarial_network/

Method

InfoNCE

2.3 InfoNCE Loss and Mutual Information Estimation

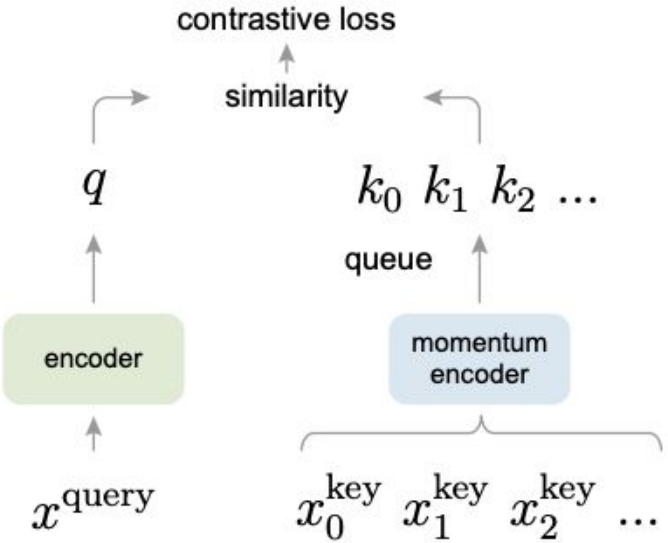
Both the encoder and autoregressive model are trained to jointly optimize a loss based on NCE, which we will call InfoNCE. Given a set $X = \{x_1, \dots x_N\}$ of N random samples containing one positive sample from $p(x_{t+k}|c_t)$ and $N-1$ negative samples from the 'proposal' distribution $p(x_{t+k})$, we optimize:

$$\mathcal{L}_{\mathrm{N}} = -\underset{X}{\mathbb{E}}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right] \tag{4}$$

$$l_i = -log\frac{e^{sim(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} e^{sim(h_i, h_j^+)/\tau}}$$

https://arxiv.org/pdf/1807.03748.pdf

Method

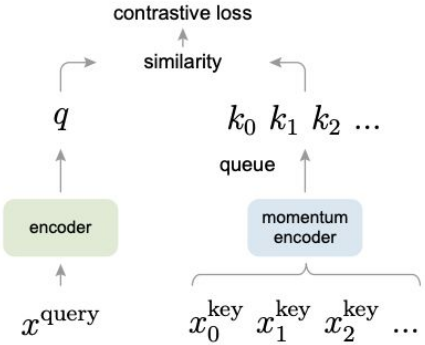


Formally, denoting the parameters of f_k as θ_k and those of f_q as θ_q, we update θ_k by:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q. \quad (2)$$

late-update 진행

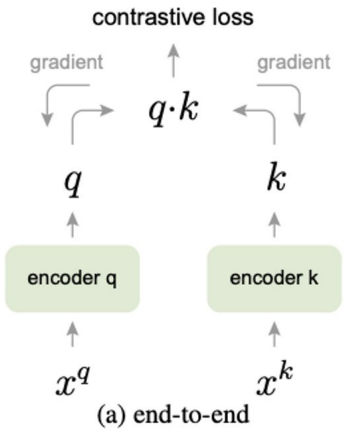
Method



the memory bank was updated when it was last seen, so the sampled keys are essentially about the encoders at multiple different steps all over the past epoch and thus are less consistent. A momentum update is adopted on the memory bank in [61]. Its momentum update is on the representations of the same sample, *not* the encoder. This momentum update is irrelevant to our method, because MoCo does not keep track of every sample. Moreover, our method is more memory-efficient and can be trained on billion-scale data, which can be intractable for a memory bank.

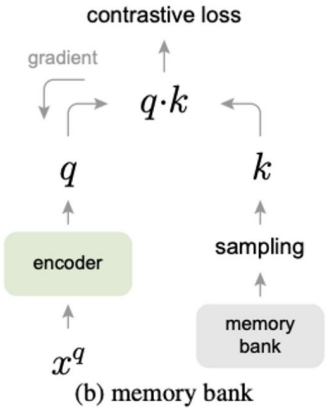
```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature
f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
  x_q = aug(x) # a randomly augmented version
  x_k = aug(x) # another randomly augmented version
  q = f_q.forward(x_q) # queries: NxC
  k = f_k.forward(x_k) # keys: NxC
  k = k.detach() # no gradient to keys

  # positive logits: Nx1
  l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

  # negative logits: NxK
  l_neg = mm(q.view(N,C), queue.view(C,K))

  # logits: Nx(1+K)
  logits = cat([l_pos, l_neg], dim=1)

  # contrastive loss, Eqn.(1)
  labels = zeros(N) # positives are the 0-th
  loss = CrossEntropyLoss(logits/t, labels)

  # SGD update: query network
  loss.backward()
  update(f_q.params)

  # momentum update: key network
  f_k.params = m*f_k.params+(1-m)*f_q.params

  # update dictionary
  enqueue(queue, k) # enqueue the current minibatch
  dequeue(queue) # dequeue the earliest minibatch
```

기존 알고리즘 비교



(a) end-to-end

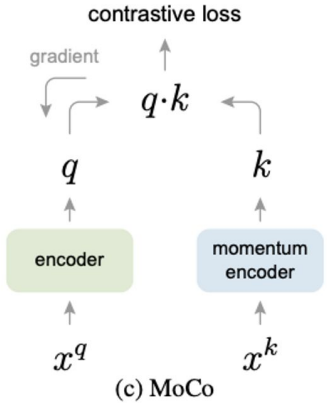
그레디언트를 같은 스텝에 진행. → GPU 메모리 이슈 존재 업데이트는 일관성있게 진행



(b) memory bank

랜덤하게 역전파 없이 추출하는 구조 단일 인코더로 진행

→ 하지만 sampling의 인코딩 값이 업데이트 되지 않음

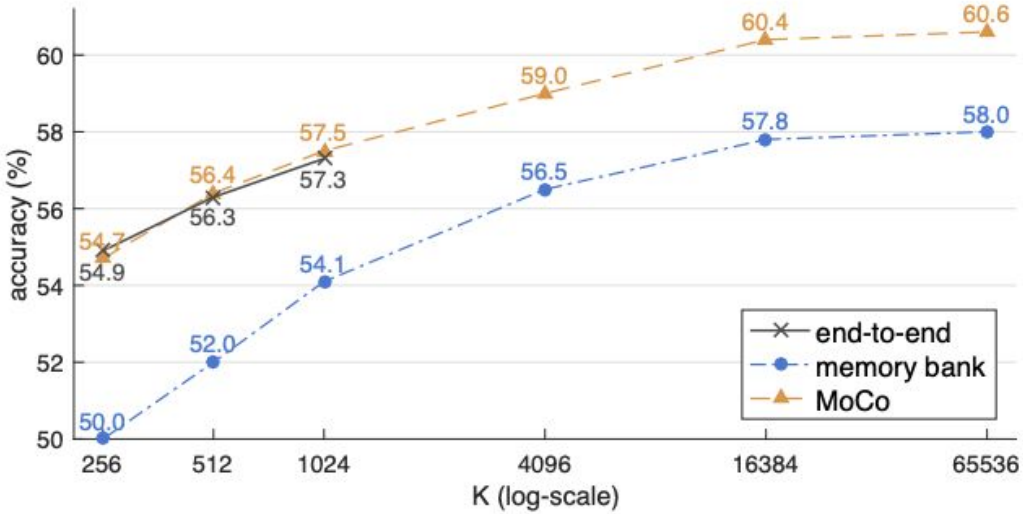


(c) MoCo

multiple different step에 역전파를 하는 것

메모리 효율적으로 대용량을 학습할 수 있게 됨

기존 알고리즘 비교



동일 메모리 공간상에서 end to end가 유사한 성능을 보이고 있는 것을 볼 수 있다. 하지만 end to end는 동일한 backward과정을 거치기 때문에 k사이즈가 버틸 수 있는 환경이 1024로 한정되어있다.

MoCo는 key와 query를 모두 업데이트 하면서도 대용량의 queue사이즈를 버틸 수 있는 좋은 알고리즘이다.

감사합니다.