ProSampler: Improving Contrastive Learning by Better Mini-batch Sampling

Tinglin Huang
tinglin.huang@yale.edu

Outline

- Negative Sampling for In-batch Contrastive Learning
- ProSampler: A Global Hard Negative Sampler
- Experiments on Three Modalities

Outline

- Negative Sampling for In-batch Contrastive Learning
- ProSampler: A Global Hard Negative Sampler
- Experiments on Three Modalities

Self-Supervised Learning (1)

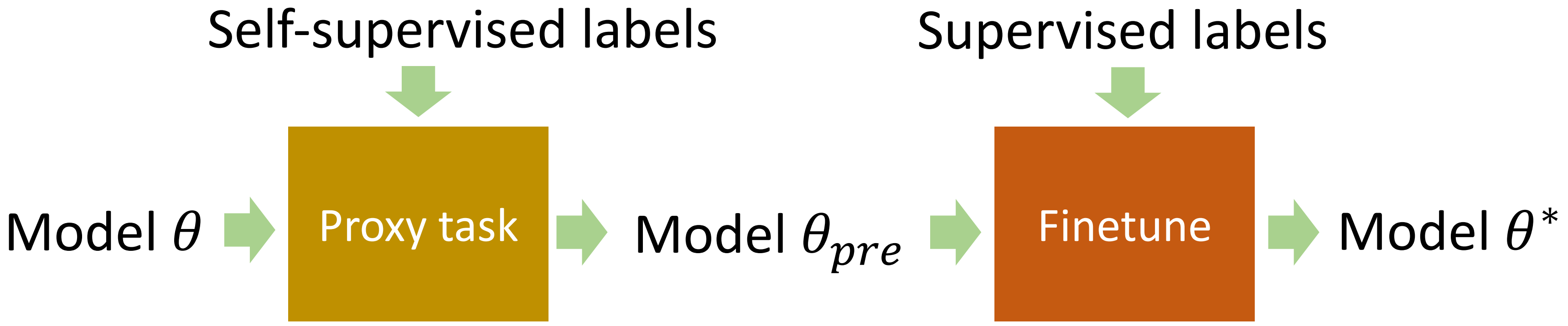
- Usually we train a model $f(\cdot)$ by minimizing the loss function
  - Dataset $\mathcal{D} = \{x_1, \dots, x_N\}$ with labels $\{y_i\}$, where x_i is a instance

$$\theta^* = \min_\theta \sum_i l(y_i, f_\theta(x_i))$$

  - Loss function depends on the task, e.g., cross entropy loss for classification task
- How to train a model **without** supervised signals?

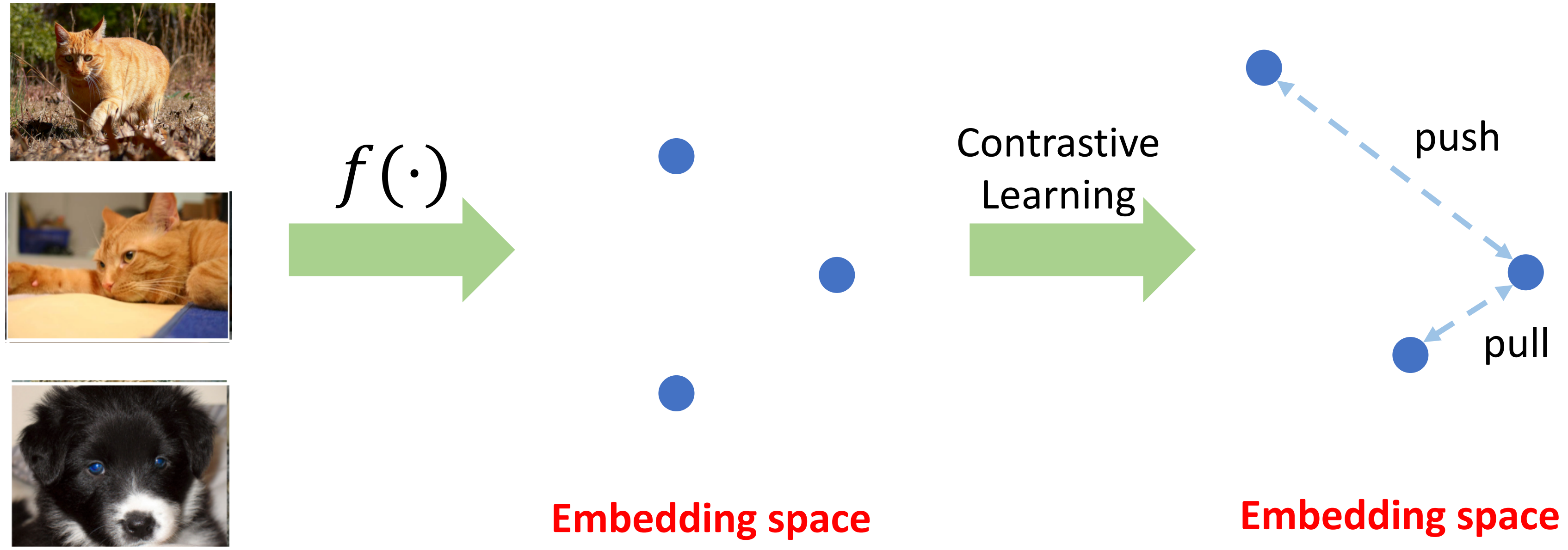
Self-Supervised Learning (2)

- How to train a model **without** supervised signals?
  - Design a proxy task!
  - Use the supervision signals from the data itself (**self-supervised** learning)
  - It can be inspired by some **domain insights**
    - For example, the representation of a cat should resemble other cats rather than a dogs



In-batch Contrastive Learning (1)

- Contrastive learning
  - One of the most successful self-supervised learning framework
  - Key idea: bringing **semantically similar** instances closer while pushing **dissimilar** instances



In-batch Contrastive Learning (2)

- Contrastive learning
  - Sample a mini-batch of instances $\{x_i\}_B$
    - B is the batchsize
  - Augment the instance x_i to generate **positive** pair (x_i, x_i^+)
    - E.g., image masking (CV), and word deletion (NLP)
  - For each positive pair, sample B^- other instances to generate **negative pairs**
    - We can have B^- negative pairs for each instance $\{(x_i, x_j)\}_{j\neq i}^{B^-}$ in the mini-batch
  - Decrease the distance between **positive** pairs, and increase the distance between **negative** pairs

In-batch Contrastive Learning (3)

- Apply InfoNCE loss to optimize:

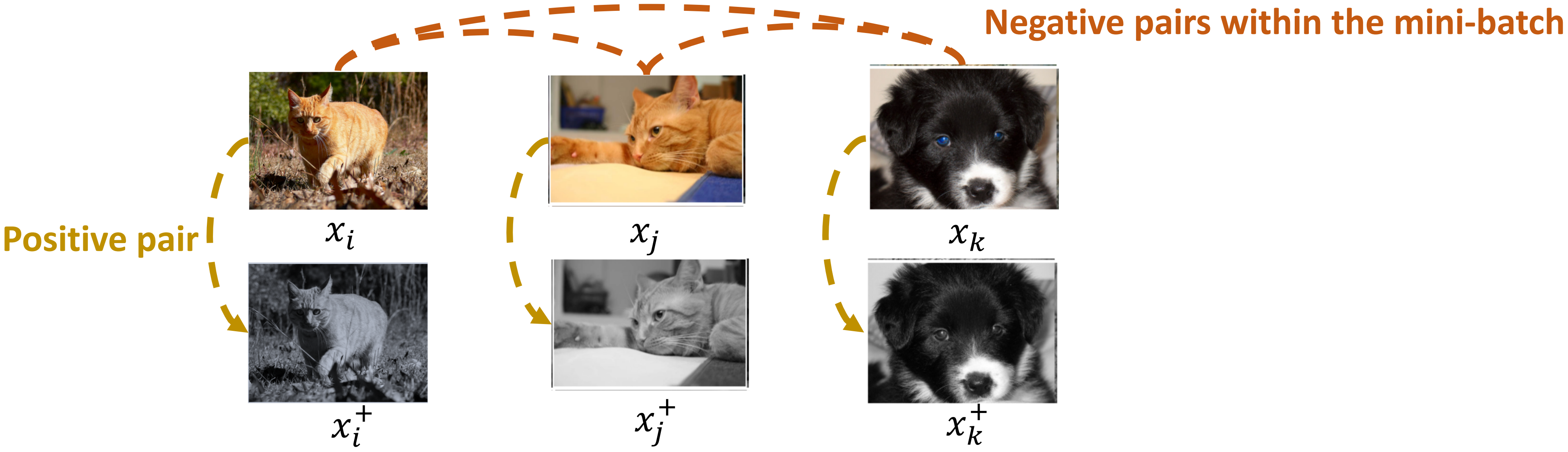
$$\min -\sum_{i=1}^{B} \log \frac{e^{f(x_i)^T f(x_i^+)}}{\sum_j e^{f(x_i)^T f(x_j)}}$$

$$= \min -\sum_{i=1}^{B} \Bigg(\underbrace{f(x_i)^T f(x_i^+)}_{\text{maximize}} - \underbrace{\log \sum_j e^{f(x_i)^T f(x_j)}}_{\text{minimize}} \Bigg)$$

- InfoNCE loss can be adapt to **various data modalities**
  - Data instance could be image, text or graph
- **How to sample negatives?**

In-batch Contrastive Learning (4)

- In-batch contrastive learning
  - We can directly treat **the other instances within a mini-batch** as negatives [1]!
    - We can have $B-1$ negative pairs for each instance $\{(x_i, x_j)\}_{j \neq i}^{B}$ in the mini-batch
  - It can simplify the training pipeline and is efficient
  - Increase the batchsize = increase the number of negatives



[1] Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *ICML*. 2020.

Negative Sampling (1)

- For in-batch contrastive learning, mini-batch sampling is equivalent to **negative sampling**
  - Every instances serve as negative to the other instances within the mini-batch
  - It is known as **in-batch negative sharing strategy**
- Negative sampling is really critical
  - MoCo[1] achieves promising results by storing the negatives in a memory bank and updating them using a momentum encoder.
  - SimCLR[2] shows that simply increasing the batch size to 8192 outperforms pervious carefully designed methods
- **What negatives contribute the most?**

[1] He, Kaiming, et al. "Momentum contrast for unsupervised visual representation learning." *CVPR*. 2020.
[2] Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *ICML*. 2020.

Negative Sampling (2)

- **Hard negative pair** contributes the most!
  - The hard-to-distinguish negative
  - Well-supported by many related studies on negative sampling, e.g., recommendation system[1] and dense retrieval[2]
  - Hard negative pairs provide meaningful gradient to the model
- Hard negative made great success in many real-world applications
  - 8% improvement of Facebook search recall[3]
  - 15% relative gains of Microsoft retrieval engine[2]
- How to sample hard negatives?

[1] Ying, Rex, et al. "Graph convolutional neural networks for web-scale recommender systems." *KDD*. 2018.
[2] Xiong, Lee, et al. "Approximate nearest neighbor negative contrastive learning for dense text retrieval." *ICLR*. 2020.
[3] Jui-Ting, Huang, et al. "Embedding-based retrieval in facebook search." KDD. 2020.

Negative Sampling (3)

- How to sample hard negatives?
  - Previous methods [1,2] apply **triplet loss** and **globally** pick the negative similar to the query one across the dataset
    - Triplet loss: sample one negative for each query instance

$$\min -\sum_{i=1}^{B} \log \frac{e^{f(x_i)^T f(x_i^+)}}{e^{f(x_i)^T f(x_i^+)} + \boxed{e^{f(x_i)^T f(x^-)}}}$$

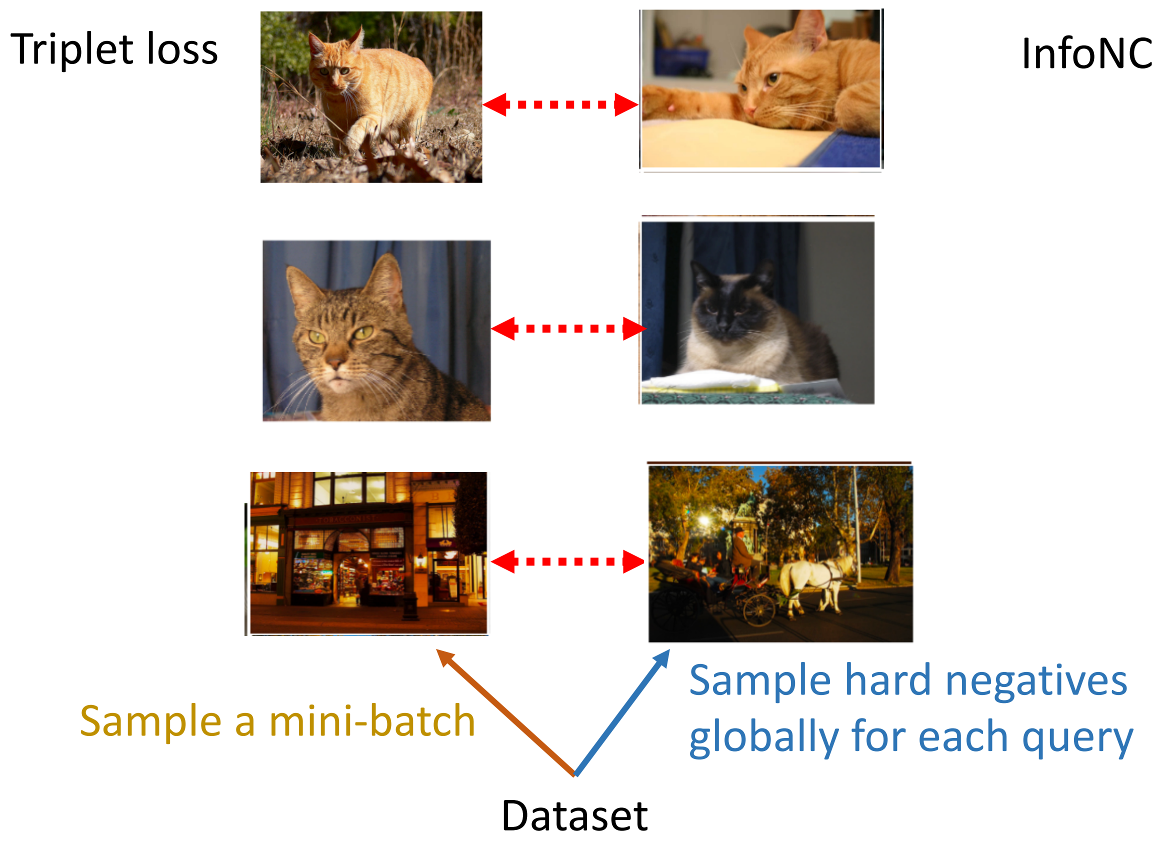
Only one negative pair

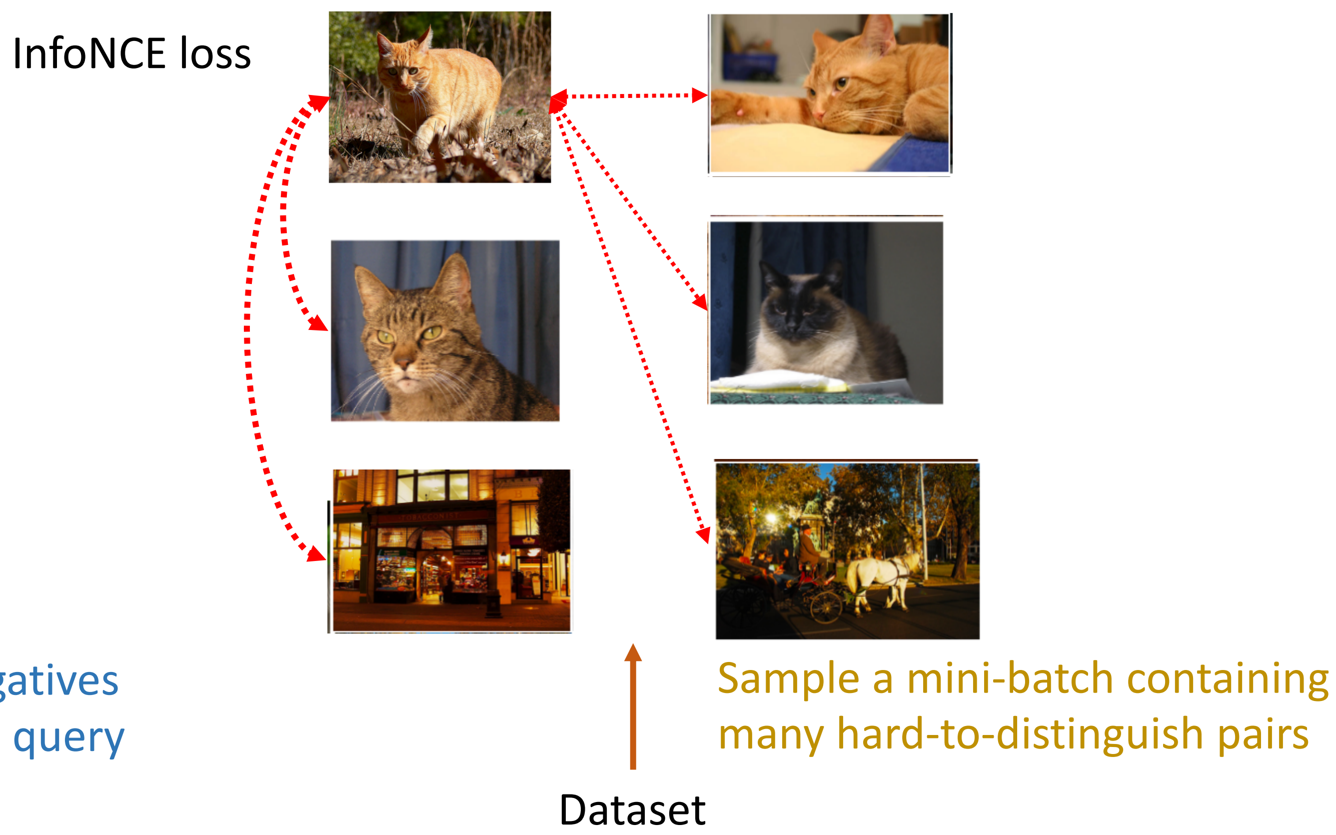
  - It is easy to recall the hard negative for the corresponding query
  - But it is **inapplicable** to in-batch contrastive learning, since it cannot guarantee the similarity between every instance pair within a mini-batch

[1] Ying, Rex, et al. "Graph convolutional neural networks for web-scale recommender systems." *KDD*. 2018.
[2] Xiong, Lee, et al. ”Approximate nearest neighbor negative contrastive learning for dense text retrieval." *ICLR*. 2020.

Negative Sampling (4)

- Global hard negative sampling in triplet loss and InfoNCE loss





Negative Sampling (5)

- How to sample hard negatives for in-batch contrastive learning?
  - Previous methods [1,2] perform negative sampling within the sampled mini-batch **locally**
  - Assign higher weights for hard negatives among the mini-batch

$$\min -\sum_{i=1}^{B} \log \frac{e^{f(x_i)^T f(x_i^+)}}{e^{f(x_i)^T f(x_i^+)} + \sum_{j \neq i} (\boxed{\lambda_{ij}} e^{f(x_i)^T f(x_j)} + \boxed{\lambda_{ij}^+})}$$

Assigned weight

Assigned weight related to the positive pair

  - But the batch size is **far smaller than** that dataset size, and sampling within the mini-batch **cannot effectively explore the hard negatives** from the whole dataset

[1] Chuang, Ching-Yao, et al. "Debiased contrastive learning." *NeuIPS.* 2020.
[2] Robinson, Joshua, et al. "Contrastive learning with hard negative samples." *ICLR*. 2021.

Problem Setup

- How to sample hard negatives for in-batch contrastive learning?
  - Previous methods in relative field show that **globally** sample hard negative can achieve promising results
    - Inapplicable to the in-batch contrastive learning framework
  - But existing methods for in-batch contrastive learning perform hard negative sampling **locally** within the mini-batch
    - Cannot effectively explore the hard negatives across the dataset, leading to a sub-optimal performance
- Target: design a **global hard** negative sampler for in-batch contrastive learning
  - Modality-independent
  - Can sample a mini-batch of instances where any instance pair are **hard to distinguish** across the dataset

Outline

- Negative Sampling for In-batch Contrastive Learning
- ProSampler: A Global Hard Negative Sampler
- Experiments on Three Modalities

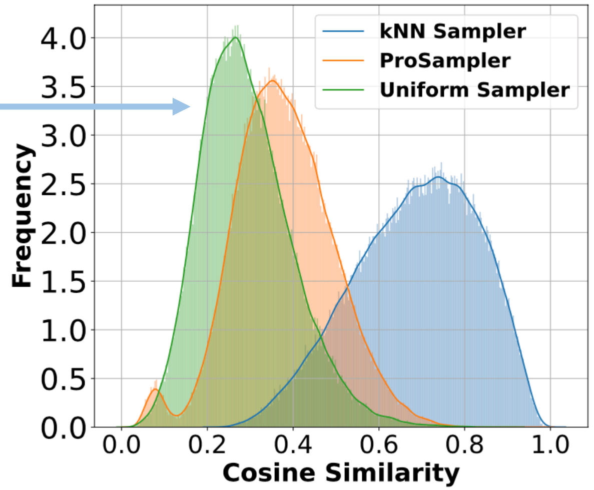
Two Extreme Strategies (1)

- Let's consider two **extreme** sampling strategies for in-batch contrastive learning
  - Represent extreme scenarios in terms of the **hardness** of a mini-batch they construct
- Uniform Sampler
  - Randomly sample a batch of instances from the dataset
- kNN Sampler
  - Pick an instance at random and retrieve a set of **nearest neighbors** to construct a batch
  - A naïve solution to globally sample a mini-batch with many hard negative examples

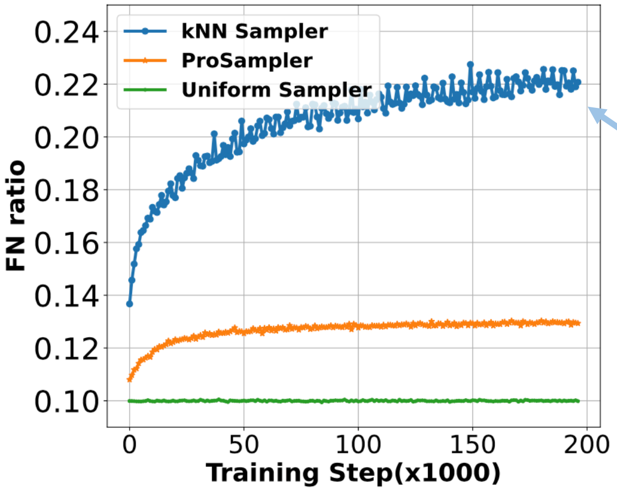
Two Extreme Strategies (2)

- But these methods suffer from the following limitations
  - Uniform Sampler neglects the effect of hard negatives
  - kNN Sampler will sample a lot of false negatives as the training epochs increase
    - False negative (FN): the negatives of the same class as query

Uniform sampler cannot sample hard negatives



Histogram of cosine similarity of negative pairs within a mini-batch



Percentage of FN within a mini-batch

kNN brings a lot of FN

Two Extreme Strategies (3)

- Uniform Sampler cannot leverage hard negatives to guide the optimization of the model
- kNN Sampler explicitly samples hard negatives but suffers from the false negative issue
- **A better global hard negative sampler** for in-batch contrastive learning should trade-off these two extreme sampling styles
  - Balance the exploitation of hard negatives and the FN issue

ProSampler

- ProSampler : Proximity Graph-based Sampler
  - Capture similarity relationships among instances by **proximity graph**
  - Perform negative sampling as a **walking** in the proximity graph
    - Collect the visited instances as sampling results
  - Smoothly interpolate between kNN Sampler and Uniform Sampler by **modulating two parameters**
- Why do we use proximity graph?
  - It can capture the similarity relationships among instances
  - It can be theoretically guaranteed that close instances will form a **local community** in the proximity graph
  - Sampling on the proximity graph can easily collect similar examples

Proximity Graph Construction (1)

- Definition
  - Proximity graph: $G = (\mathcal{V}, \mathcal{E})$
    - $\mathcal{V}$ is the node set and $\mathcal{E}$ is a collection of node pairs
  - $\mathcal{N}_i$ is the neighbor set of v_i in the G
  - N observation $\mathcal{V} = \{v_i | i = 1, \dots, N\}$ which is the node set in G
  - Representations $\{\mathbf{e}_i | i = 1, \dots, N\}$ generated by current encoder $f(\cdot)$
- Proximity graph construction
  - Randomly pick $M (M \ll N)$ **neighbor candidates** to form a candidate set $\mathcal{C}_i = \{v_m\}$ for each instance v_i
  - Select the K **nearest** ones from the candidate set to form the neighbor set

$$\mathcal{N}_i = \mathrm{TopK}_{v_m \in \mathcal{C}_i}(\mathbf{e}_i \cdot \mathbf{e}_m)$$

Proximity Graph Construction (2)

- Candidate set size M **controls the similarity** between center node and its neighbors
  - When $M = N$, proximity graph degenerates to kNN graph
  - When $M = 1$, each node will randomly connect with the other node
- Theoretical proof

Proposition 1. *Given an observation v_i with the corresponding representation $\mathbf{e}_i$, assume that there are at least S observations whose inner product similarity with v_i is larger than s, i.e.,*

$$\left|\left\{v_j \in \mathcal{V} \mid \mathbf{e}_i \cdot \mathbf{e}_j > s\right\}\right| \geq S. \tag{4}$$

Then in the proximity graph G, the similarity between v_i and its neighbors is larger than s with proximate probability at least:

Higher M indicates a greater probability that two adjacent nodes are similar

$$\mathbb{P}\left\{\mathbf{e}_i \cdot \mathbf{e}_k > s, \forall v_k \in \mathcal{N}_i\right\} \gtrapprox \left(1 - p^M\right)^K, \tag{5}$$

where $p = \frac{N-S}{N}$, and K is the number of neighbors.

Proximity Graph Sampling (1)

- Perform mini-batch sampling as graph sampling
- Two straightforward graph sampling methods
  - **Breadth-first Sampling(BFS)** collects all of the current node's immediate neighbors, then moves to its neighbors and repeats the procedure
  - **Depth-first Sampling(DFS)** randomly explores the node branch as far as possible
- We apply **Random Walk with Restart (RWR)** which exhibits a mixture of both
  - It can flexibly explore the negatives in proximity graph
  - Beginning at a node, the sampler iteratively teleports back to the start point with probability α or travels to a neighbor of the current position

Proximity Graph Sampling (2)

- Restart probability α can modulate the probability of sampling within a neighborhood
- Theoretical proof:

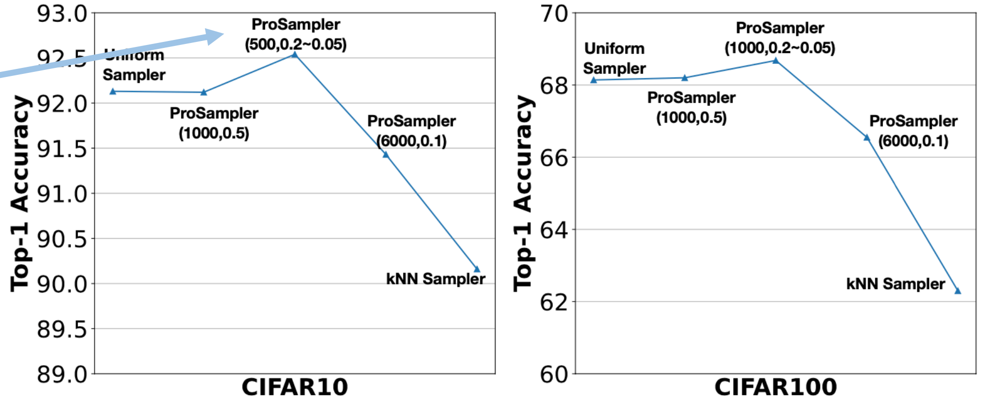
Proposition 2. *For all $0 < \alpha \leq 1$ and $\mathcal{S} \subset \mathcal{V}$, the probability that a Lazy Random Walk with Restart starting from a node $u \in \mathcal{S}$ escapes $\mathcal{S}$ satisfies $\sum_{v \in (\mathcal{V}-\mathcal{S})} \mathbf{p}_u(v) \leq \frac{1-\alpha}{2\alpha}\Phi(\mathcal{S})$, where $\mathbf{p}_u$ is the stationary distribution, and $\Phi(\mathcal{S})$ is the graph conductance of $\mathcal{S}$.*

  - The probability of RWR escaping from a local cluster can be bound by α
  - Higher α indicates that the walker will approximate BFS behavior and sample within a small locality
  - Lower α encourages the walker to visit the nodes which are further away from the center node.

ProSampler

- The number of candidates M and the restart probability α are the key to **flexibly control the hardness** of a sampled batch
  - When $M = N, \alpha = 1$, ProSampler behaves similarly to a kNN Sampler
    - Proximity graph is equivalent to kNN graph, and graph sampler will only collect the immediate neighbors around a center node
  - When $M = 1, \alpha = 0$, ProSampler performs as a Uniform Sampler
    - RWR degenerates into the DFS and chooses the neighbors that are linked at random

ProSampler(M, α)



(M, α) can find a balance between these two extreme samplers

Performance of different samplers on image classification task

ProSampler Pipeline

Algorithm 1: In-batch Contrastive Framework with ProSampler

Input: Dataset $\mathcal{D} = \{x_i | i = 1, \cdots, N\}$, Encoder $f(\cdot)$, Batchsize B, Graph update step t, Modality-specific augmentation functions $\mathcal{T}$.

for *iter* $\leftarrow 0, 1, \cdots$ **do**

Update proximity graph after t steps

// ProSampler

if $iter \% t == 0$ **then**

// Proximity Graph Construction

Build the proximity graph G by Algorithm 2.

end

RWR

// Proximity Graph Sampling

Randomly select a start node and get the mini-batch $\{x_i\}_B$ by Algorithm 3.

InfoNCE loss

Obtain positive pairs $\{(x_i, x_i^+)\}_B$ by augmentation functions $f_{aug}(\cdot) \sim \mathcal{T}$.

Generate representations $\{(\mathbf{e}_i, \mathbf{e}_i^+)\}_B$ by Encoder $f(\cdot)$.

Compute the loss by Eq. 1, where $\{(\mathbf{e}_i, \mathbf{e}_j)\}_{B(B-1)}^{i \neq j}$ are treated as negative pairs.

Update the parameters of $f(\cdot)$.

end

Outline

- Negative Sampling for In-batch Contrastive Learning
- ProSampler: A Global Hard Negative Sampler
- Experiments on Three Modalities

Experiments

- We evaluate the ProSampler on **four** representative in-batch contrastive learning framework on **three** data modalities
  - Image Modality: MoCo v3, SimCLR
  - Text Modality: SimCSE
  - Graph Modality: GraphCL
- We also equip two variants of InfoNCE objective with ProSampler to investigate its generality
  - DCL and HCL: locally negative sampling framework
- Training pipeline: self-supervised learning -> linear probing
  - linear probing: fix the pretrained representation and evaluate the performance on downstream task with a linear classifier

Image Modality

- Dataset: ImageNet
- Backbone: ResNet-50
- Baseline: SwAV and BYOL
  - SOTA self-supervised learning framework without negative sampling

Method	100 ep	400 ep
SwAV*	66.5	70.1
BYOL	66.5	73.2
SimCLR	64.0	68.1
w/ ProSampler	**64.7** (↑ 0.7)	**68.6** (↑ 0.5)
MoCo v3	68.9	73.3
w/ ProSampler	**69.5** (↑ 0.6)	**73.7** (↑ 0.4)

* without multi-crop augmentations.

Text Modality

- Dataset:7 semantic textual similarity tasks
- Backbone: BERT

Table 2: Overall performance comparison with different negative sampling methods on STS tasks.

Method	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
SimCSE-BERT$_{base}$	68.62	80.89	73.74	80.88	77.66	**77.79**	**69.64**	75.60
w/ ProSampler	**72.37**	**82.08**	**75.24**	**83.10**	**78.43**	77.54	68.05	**76.69**
DCL-BERT$_{base}$	65.22	77.89	68.94	79.88	**76.72**	73.89	**69.54**	73.15
w/ ProSampler	**69.55**	**82.66**	**73.37**	**80.40**	75.37	**75.43**	66.76	**74.79**
HCL-BERT$_{base}$	62.57	79.12	69.70	78.00	75.11	73.38	69.74	72.52
w/ ProSampler	**66.87**	**81.38**	**72.96**	**80.11**	**77.99**	**75.95**	**70.89**	**75.16**

Graph Modality

- Dataset: graph classification benchmark datasets
  - IMDB-B, IMDB-M, COLLAB, REDDIT-B
- Backbone: GIN

Table 3: Accuracy on graph classification task under LIBSVM (Chang and Lin, 2011) classifier.

Method	IMDB-B	IMDB-M	COLLAB	REDDIT-B
GraphCL	70.90±0.53	48.48±0.38	70.62±0.23	90.54±0.25
w/ ProSampler	**71.90±0.46**	**48.93±0.28**	**71.48±0.28**	**90.88±0.16**
DCL	71.07±0.36	48.93±0.32	**71.06±0.51**	90.66±0.29
w/ ProSampler	**71.32±0.17**	**48.96±0.25**	70.44±0.35	**90.73±0.34**
HCL	**71.24±0.36**	48.54±0.51	71.03±0.45	90.40±0.42
w/ ProSampler	71.20±0.38	**48.76±0.39**	**71.70±0.35**	**91.25±0.25**

Empirical Criterion of (M, α)

linearly decay α from 0.2 to 0.05 as the training epoch increases

Table 4: Impact of neighbor candidates M.

M	500	1000	2000	4000	6000
CIFAR10	**92.54**	92.49	91.83	91.72	91.43
CIFAR100	67.92	**68.68**	67.05	66.19	65.55
STL10	84.16	**84.38**	82.80	81.91	80.92
ImageNet-100	59.6	**60.8**	60.1	59.1	58.4
Wikipedia	71.36	**76.69**	76.09	75.76	75.11
COLLAB	70.47	**71.48**	70.93	70.46	70.24

Table 5: Impact of restart probability α.

α	0.1	0.3	0.5	0.7	0.2~0.05
CIFAR10	92.41	92.26	92.12	92.06	**92.54**
CIFAR100	68.31	67.98	68.20	68.00	**68.68**
STL10	83.01	80.69	83.93	82.56	**84.38**
ImageNet-100	60.8	59.6	58.1	57.7	**60.8**
Wikipedia	71.74	72.13	72.41	**76.69**	–
COLLAB	70.36	70.63	70.63	70.31	**71.48**

- The suggested M would be 500 for the small-scale dataset, and 1000 for the larger dataset
- The suggested α should be relatively high, e.g., 0.7, for the pre-trained language model-based method. Besides, dynamic decay α, e.g., 0.2 to 0.05, is the best strategy for the other algorithms.