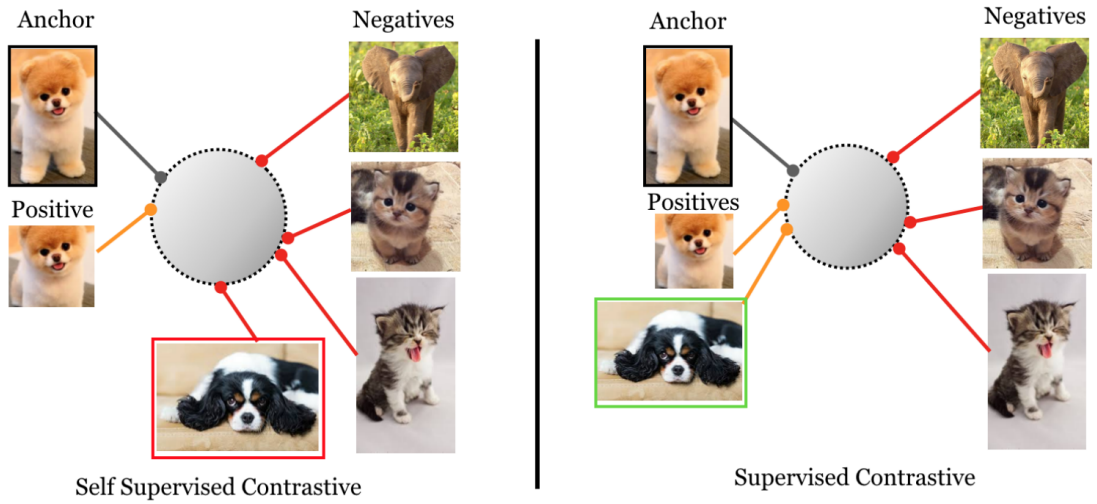
Contrastive learning in NLP

Contrastive learning is learning technique contrast semantically similar (positive) and dissimilar (negative) pairs of data points, encouraging the representations f of similar pairs (x, x+) to be close, and those of dissimilar pairs (x, x−) to be more orthogonal, involving elements:

- Training data point
- Positive points
- Negative points
- Objective function



Intuition

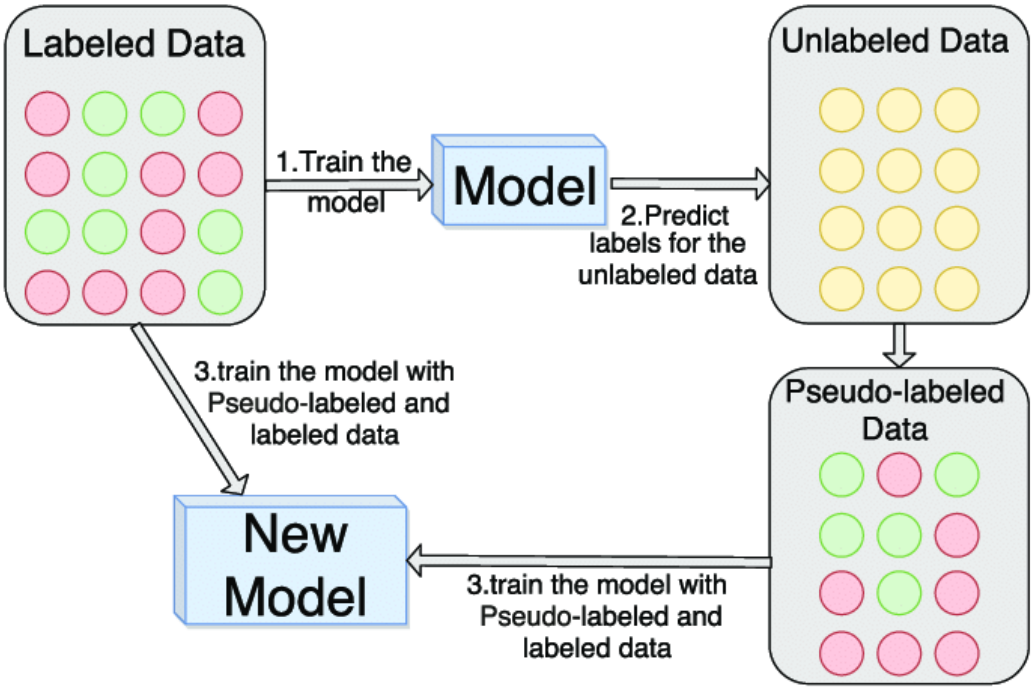
Current issues

- Self-supervised learning

  Data

  - Labeled data is costly
  - Noisy data is ubiquitous

- Semi supervised methods like pseudo-labeling methods require clean data for labeling model



→ Robust Self-supervised learning

- Spherical features

  Softmax Cross Entropy : robustness !!!

  - Noisy labels

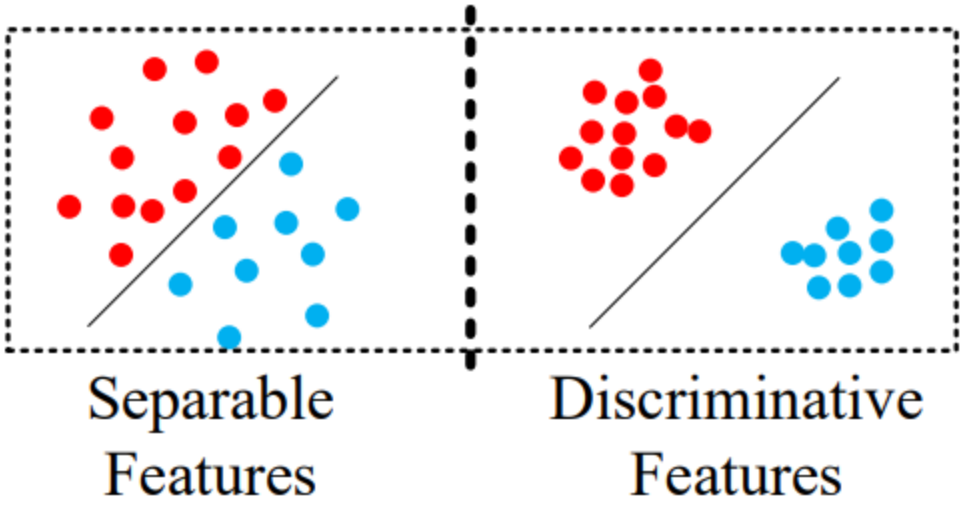
    1406.2080.pdf (arxiv.org)

    1805.07836.pdf (arxiv.org)

  - Poor margin

    eccv2016.pdf (kpzhang93.github.io)

    Separable not discriminative

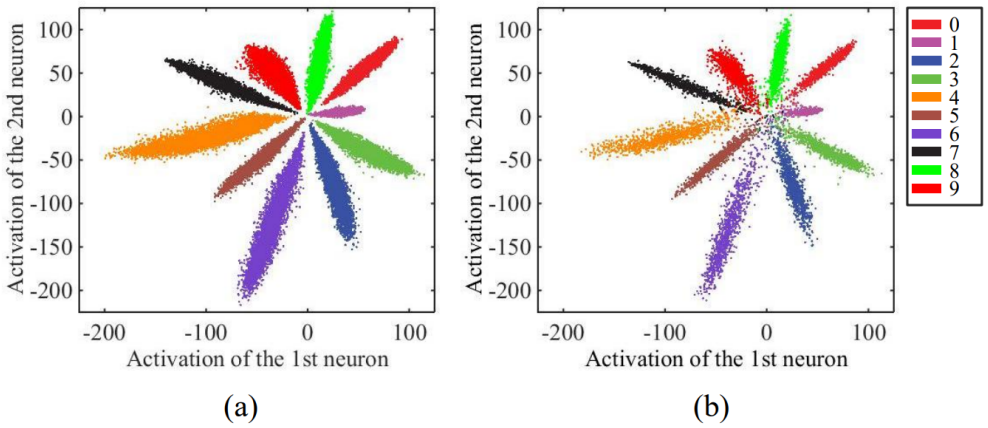


- Solution: Intra-class and inter-class enforcing:
  - Center loss

    eccv2016.pdf (kpzhang93.github.io)

    Squared Euclidean margin

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_C$$
$$= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$



- Angular losses

  1704.08063.pdf (arxiv.org)

  map original space to hypersphere space with l2 normalization with POVs:

  - Angular distribution of softmax
  - Squared Euclidean loss's incompatibility with softmax
  - Margin enforcing

$$L_{\text{ang}} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\cos(m\theta_{y_i,i})}}{e^{\|\boldsymbol{x}_i\|\cos(m\theta_{y_i,i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right)$$

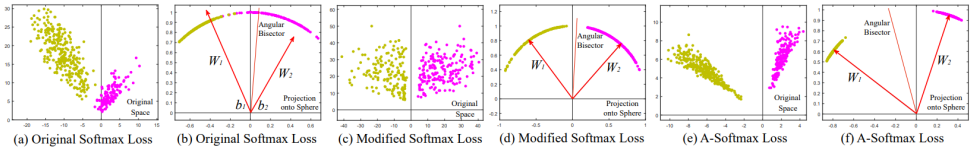
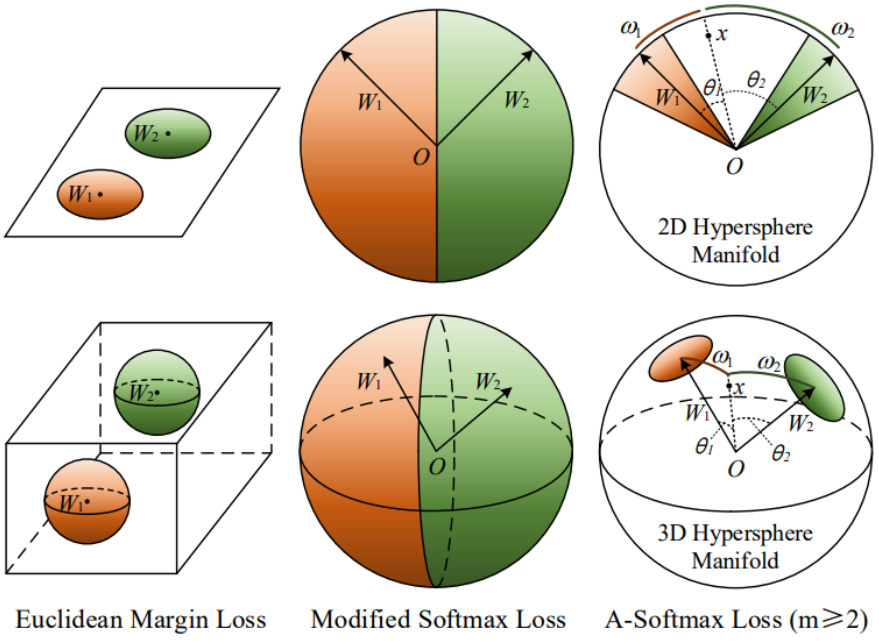


Figure 2: Comparison among softmax loss, modified softmax loss and A-Softmax loss. In this toy experiment, we construct a CNN to learn 2-D features on a subset of the CASIA face dataset. In specific, we set the output dimension of FC1 layer as 2 and visualize the learned features. Yellow dots represent the first class face features, while purple dots represent the second class face features. One can see that features learned by the original softmax loss can not be classified simply via angles, while modified softmax loss can. Our A-Softmax loss can further increase the angular margin of learned features.



→ Hyperspherical representation → L2 normalization feature embedding + Intra/Inter class enforcing?

In the same manner, Contrastive learning methods use l2 normalization in embedding space with desirable pros

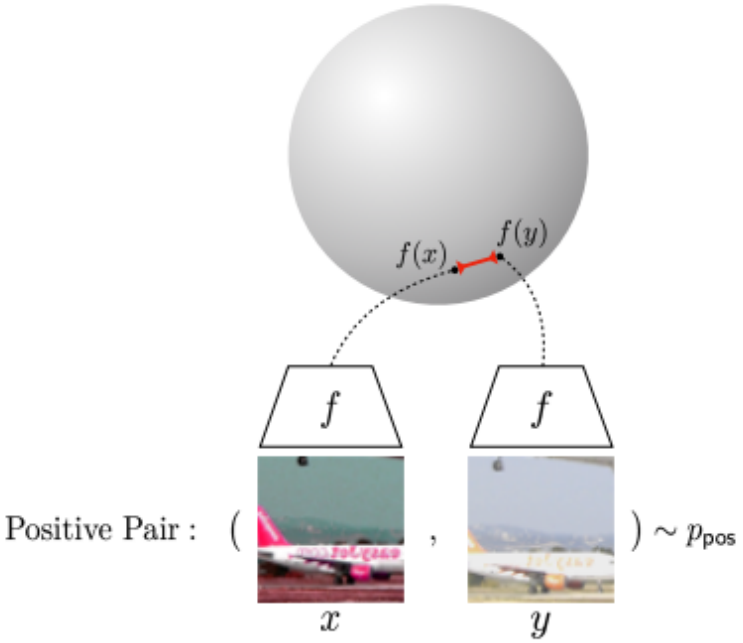
- Robust intra/inter class enforcing
- Stable training
- Well clustered classes are discriminative

Goal: learn features laying on unit sphere space (Sn-1) s.t:

Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere (mlr.press)

- Alignment

  Similar representation for similar data points

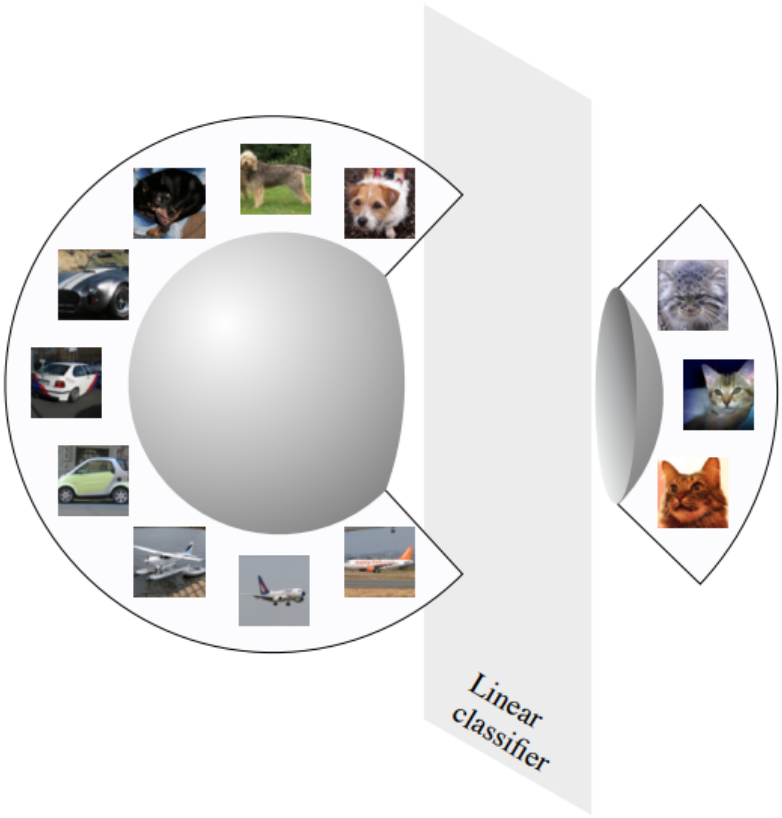


Alignment: Similar samples have similar features.
(Figure inspired by Tian et al. (2019).)

- Uniformity

  Representation preserve the orginal data information

Uniformity: Preserve maximal information.



Foundations:

Objective functions

- Contrastive Loss (Sci-Hub | | 10.1109/cvpr.2005.202)
  - Task: Face verification
  - Objective:
    - Energy function → Binary classification

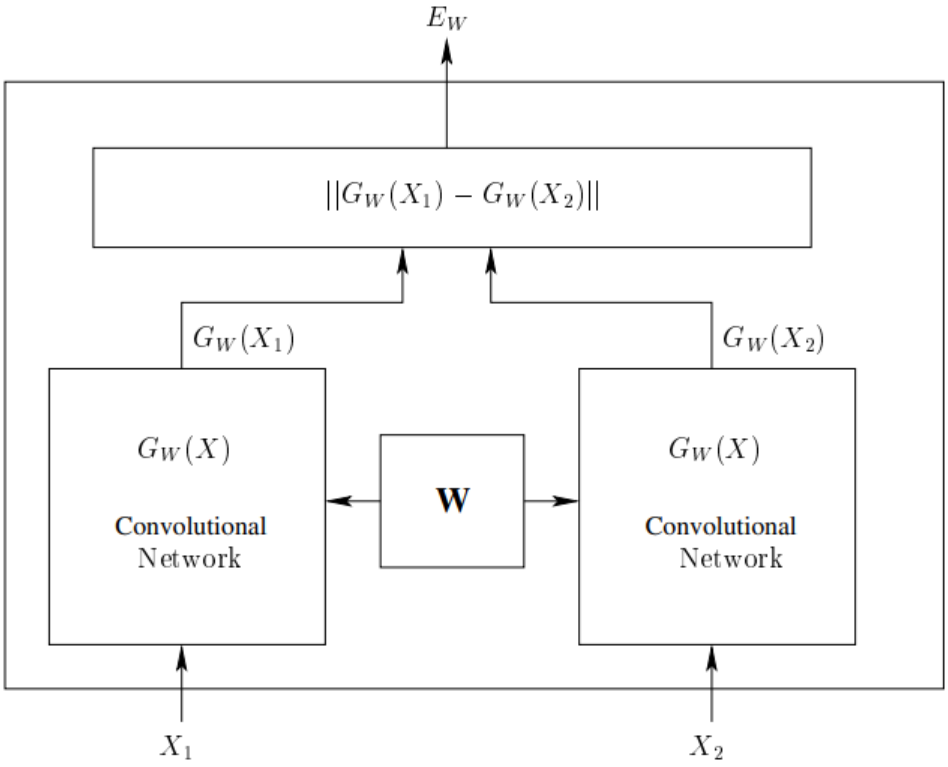


Figure 1. Siamese Architecture.

- Loss function:
  - Goal: Minimize the energy function for the CORRECT ANSWER !!!
  - Related:

    EBM generative Contrastive divergence

    Deep Generative Models | CFCS, CS Department, Peking Univeristy (deep-generative-models.github.io)

    cs236_lecture11.pdf (deepgenerativemodels.github.io)

$$
\begin{aligned}
& \nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta) \\
= \quad & \nabla_\theta f_\theta(x_{train}) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)} \\
= \quad & \nabla_\theta f_\theta(x_{train}) - \frac{1}{Z(\theta)} \int \nabla_\theta \exp\{f_\theta(x)\} \mathrm{d}x \\
= \quad & \nabla_\theta f_\theta(x_{train}) - \frac{1}{Z(\theta)} \int \exp\{f_\theta(x)\} \nabla_\theta f_\theta(x) \mathrm{d}x \\
= \quad & \nabla_\theta f_\theta(x_{train}) - \int \frac{\exp\{f_\theta(x)\}}{Z(\theta)} \nabla_\theta f_\theta(x) \mathrm{d}x \\
= \quad & \nabla_\theta f_\theta(x_{train}) - E_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})] \\
\approx \quad & \nabla_\theta f_\theta(x_{train}) - \nabla_\theta f_\theta(x_{sample}),
\end{aligned}
$$

- General

$$
\begin{aligned}
\mathcal{L}(W) &= \sum_{i=1}^{P} L(W, (Y, X_1, X_2)^i) \\
L(W, (Y, X_1, X_2)^i) &= (1 - Y) L_G \left(E_W(X_1, X_2)^i\right) \\
&\quad + Y L_I \left(E_W(X_1, X_2)^i\right)
\end{aligned}
$$

- Specific

$$
\begin{aligned}
L(W, Y, X_1, X_2) &= (1 - Y) L_G(E_W) + Y L_I(E_W) \quad (8) \\
&= (1 - Y) \frac{2}{Q} (E_W)^2 + (Y) 2Q \, e^{-\frac{2.77}{Q} E_W} \quad (9)
\end{aligned}
$$

- Design motivation

Condition 1 $\exists m > 0$, *such that* $E_W(X_1, X_2) + m < E_W(X_1, X_2')$,

- Triplet loss

  https://arxiv.org/pdf/1503.03832.pdf

- Task: Face feature representation → Prototype feature generation
- Objective:
  - Loss function:

$$\sum_{i}^{N}\left[\|f(x_i^a)-f(x_i^p)\|_2^2-\|f(x_i^a)-f(x_i^n)\|_2^2+\alpha\right]_+$$

  - Design motivation

    T is triplet set (cardinality N)

$$\|f(x_i^a)-f(x_i^p)\|_2^2+\alpha<\|f(x_i^a)-f(x_i^n)\|_2^2\,, \tag{1}$$

$$\forall\,(f(x_i^a),f(x_i^p),f(x_i^n))\in\mathcal{T}\,. \tag{2}$$

  - Sampling
    - hard cases for + and -

> In order to ensure fast convergence it is crucial to select triplets that violate the triplet constraint in Eq. (1). This means that, given x_i^a, we want to select an x_i^p (*hard positive*) such that $\text{argmax}_{x_i^p}\|f(x_i^a)-f(x_i^p)\|_2^2$ and similarly x_i^n (*hard negative*) such that $\text{argmin}_{x_i^n}\|f(x_i^a)-f(x_i^n)\|_2^2$.
>
> It is infeasible to compute the argmin and argmax across the whole training set. Additionally, it might lead to poor training, as mislabelled and poorly imaged faces would dominate the hard positives and negatives. There are two obvious choices that avoid this issue:

    - Whole set checkpoint review selection vs In-batch selection

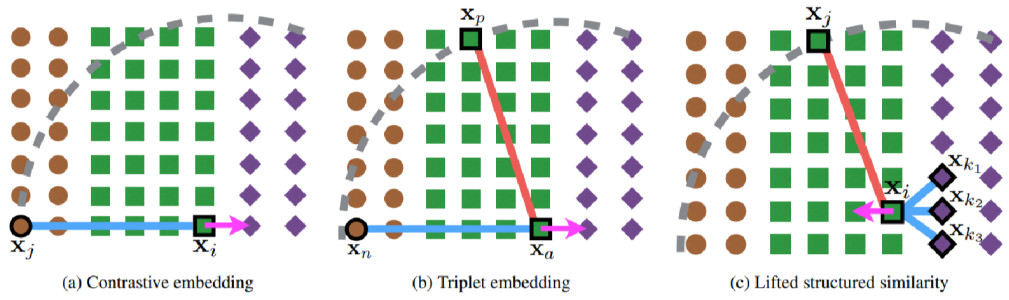
- Generate triplets offline every n steps, using the most recent network checkpoint and computing the argmin and argmax on a subset of the data.
- Generate triplets online. This can be done by selecting the hard positive/negative exemplars from within a mini-batch.

Selecting the hardest negatives can in practice lead to bad local minima early on in training, specifically it can result in a collapsed model (*i.e.* $f(x) = 0$). In order to mitigate this, it helps to select x_i^n such that

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2 \ . \qquad (4)$$

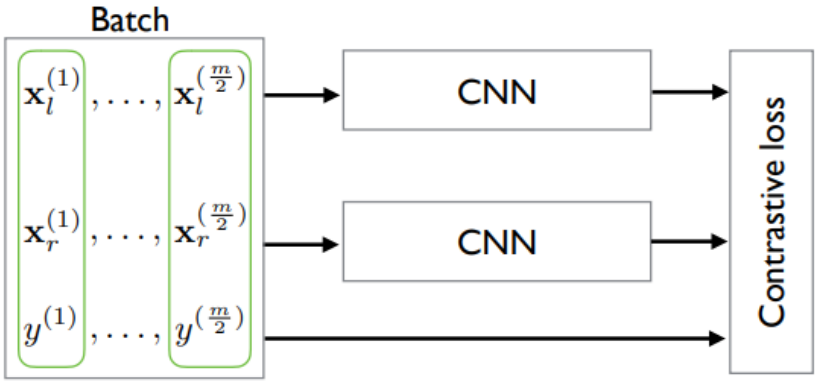
We call these negative exemplars *semi-hard*, as they are further away from the anchor than the positive exemplar, but still hard because the squared distance is close to the anchor-positive distance. Those negatives lie inside the margin α.

    - Harmonic embedding (Compatibility between versions)
- Lifted Structure loss:
    - Task: Deep metric learning (Meta learning method)
    - Objective:
        - Goal: use full batch value instead of triplet

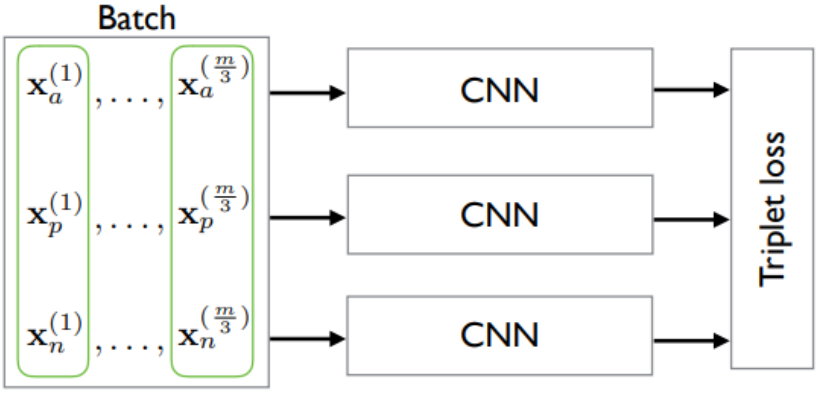


(a) Contrastive embedding (b) Triplet embedding (c) Lifted structured similarity

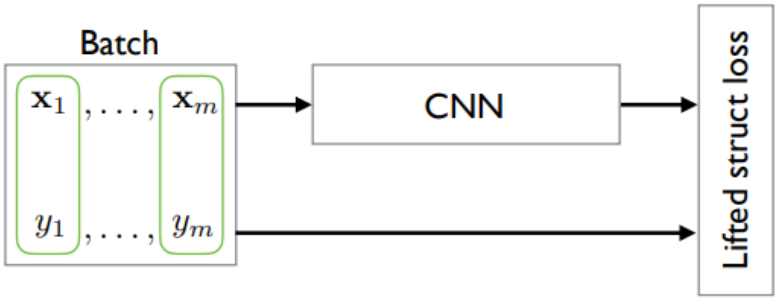
- Loss function:



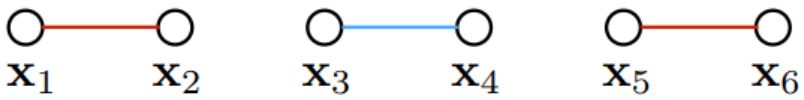
(a) Training network with contrastive embedding [14]



(b) Training network with triplet embedding [39, 31]

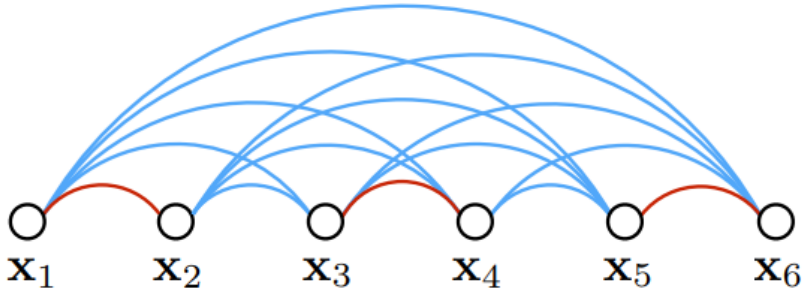


(c) Training network with lifted structure embedding



(a) Contrastive embedding

(b) Triplet embedding



(c) Lifted structured embedding

Use the upper bound : relaxed, avoid bad local optimum

$$J = \frac{1}{2|\widehat{\mathcal{P}}|} \sum_{(i,j)\in\widehat{\mathcal{P}}} \max\left(0,\ J_{i,j}\right)^2,$$

$$J_{i,j} = \max\left(\max_{(i,k)\in\widehat{\mathcal{N}}} \alpha - D_{i,k},\ \max_{(j,l)\in\widehat{\mathcal{N}}} \alpha - D_{j,l}\right) + D_{i,j}$$

$$\tilde{J}_{i,j} = \log\left(\sum_{(i,k)\in\mathcal{N}} \exp\{\alpha - D_{i,k}\} + \sum_{(j,l)\in\mathcal{N}} \exp\{\alpha - D_{j,l}\}\right) + D_{i,j}$$

$$\tilde{J} = \frac{1}{2|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \max\left(0,\ \tilde{J}_{i,j}\right)^2, \qquad (4)$$

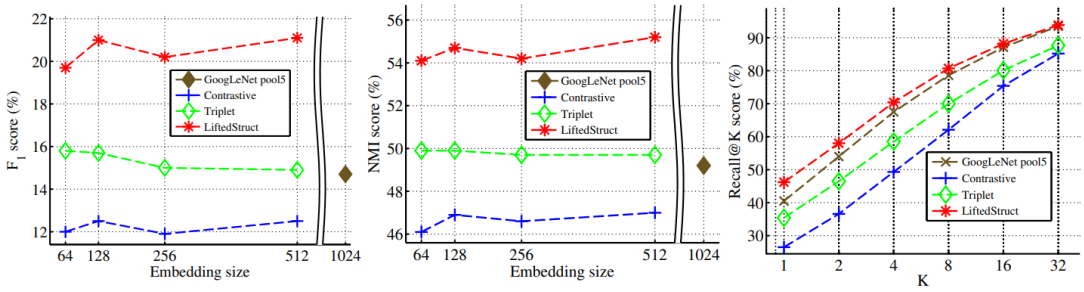
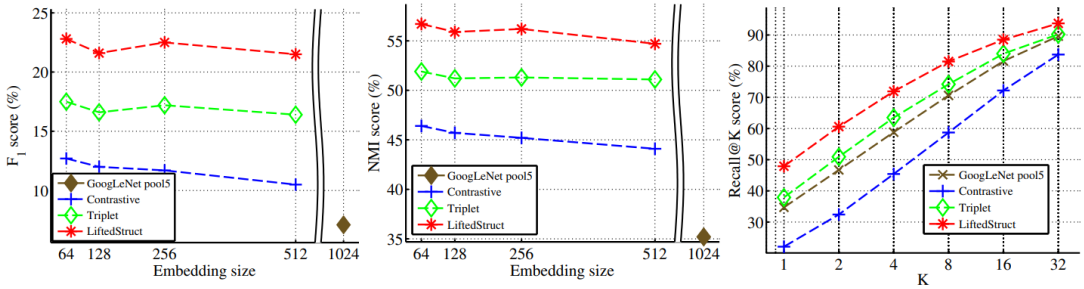


Figure 6: F_1, NMI, and Recall@K score metrics on the test split of CUB200-2011 with GoogLeNet [33].



- N-pair loss: (2016)

  Improved Deep Metric Learning with Multi-class N-pair Loss Objective (nips.cc)

  - Task: improve triplet loss, and CL loss

    > Compared to contrastive loss, triplet loss only requires the difference of (dis-)similarities between positive and negative examples to the query point to be larger than a margin m. Despite their wide use, both loss functions are known to suffer from slow convergence and they often require expensive data sampling method to provide nontrivial pairs or triplets to accelerate the training [2, 19, 17, 4].

    > enlarging the distances between that of negative examples. However, during one update, the triplet loss only compares an example with one negative example while ignoring negative examples from the rest of the classes. As a consequence, the embedding vector for an example is only guaranteed to be far from the selected negative class but not necessarily the others. Thus we can end up only differentiating an example from a limited selection of negative classes yet still maintain a small distance from many other classes. In practice, the hope is that, after looping over sufficiently many

  - Loss-function

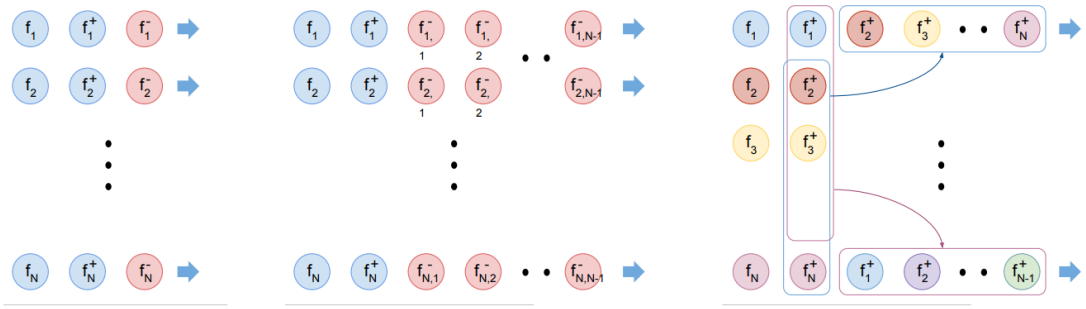
    N+1 tuple loss

$$\mathcal{L}(\{x, x^+, \{x_i\}_{i=1}^{N-1}\}; f) = \log\left(1 + \sum_{i=1}^{N-1} \exp(f^\top f_i - f^\top f^+)\right)$$

$$\mathcal{L}_{\text{N-pair}}(\mathbf{x}, \mathbf{x}^+, \{\mathbf{x}_i^-\}_{i=1}^{N-1}) = \log\left(1 + \sum_{i=1}^{N-1} \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-) - f(\mathbf{x})^\top f(\mathbf{x}^+))\right)$$
$$= -\log \frac{\exp(f(\mathbf{x})^\top f(\mathbf{x}^+))}{\exp(f(\mathbf{x})^\top f(\mathbf{x}^+)) + \sum_{i=1}^{N-1} \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-))}$$

- N-pair

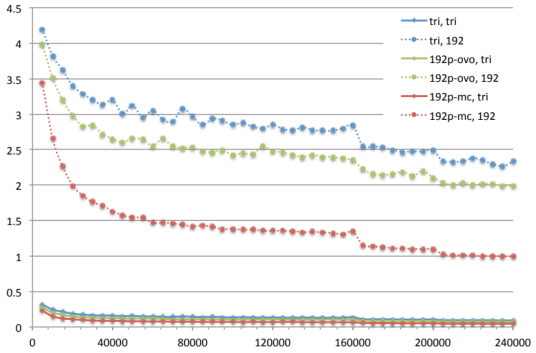
Now, we introduce an effective batch construction to avoid excessive computational burden. Let $\{(x_1, x_1^+), \cdots, (x_N, x_N^+)\}$ be N pairs of examples from N different classes, i.e., $y_i \neq y_j, \forall i \neq j$. We build N tuplets, denoted as $\{S_i\}_{i=1}^N$, from the N pairs, where $S_i = \{x_i, x_1^+, x_2^+, \cdots, x_N^+\}$. Here, x_i is the query for S_i, x_i^+ is the positive example and $x_j^+, j \neq i$ are the negative examples.



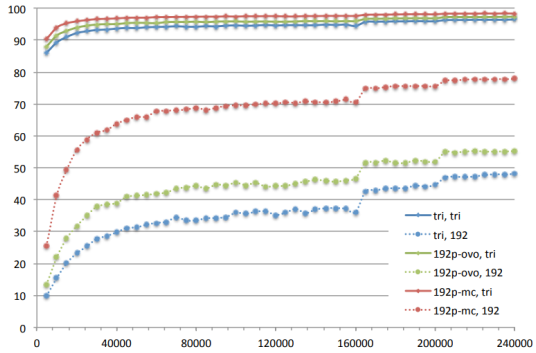
(a) Triplet loss (b) $(N+1)$-tuplet loss (c) N-pair-mc loss

- Mining hard cases

1. **Evaluate Embedding Vectors:** choose randomly a large number of output classes C; for each class, randomly pass a few (one or two) examples to extract their embedding vectors.
2. **Select Negative Classes:** select one class randomly from C classes from step 1. Next, greedily add a new class that violates triplet constraint the most w.r.t. the selected classes till we reach N classes. When a tie appears, we randomly pick one of tied classes [28].
3. **Finalize N-pair:** draw two examples from each selected class from step 2.



(a) Triplet and 192-pair loss



(b) Triplet and 192-way classification accuracy

- InfoNCE: (2019)

  Global NCE ?

  - **NCE: estimation methods {p(x|c)}, one real one fake, binary cls, logistic funtion**

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[\log\sigma(\ell_\theta(\mathbf{x}_i)) + \log(1-\sigma(\ell_\theta(\tilde{\mathbf{x}}_i)))\right]$$

$$\text{where } \sigma(\ell) = \frac{1}{1+\exp(-\ell)} = \frac{p_\theta}{p_\theta + q}$$

  InfoNCE: multi class

  - Mutual information maximization (Explanation of Contrastive Predictive Coding - Ruihong Qiu (proof of surrogate))

$$I(x;c) = \sum_{x,c} p(x,c)\log\frac{p(x|c)}{p(x)}.$$

- Predictive coding: preserve mutual information instead of likelihood function (

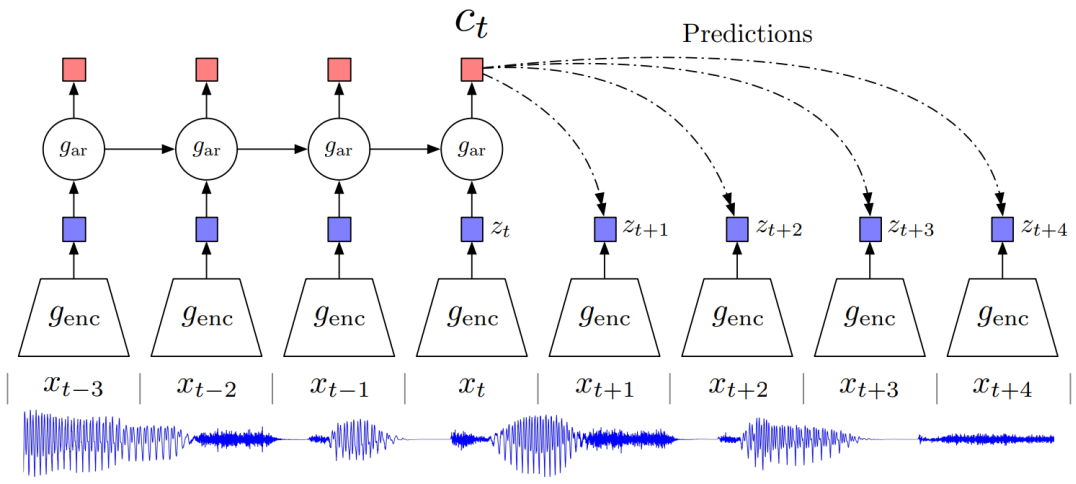
Explanation of Contrastive Predictive Coding - Ruihong Qiu NCE)

$$f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$$

- Loss function:

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

- Paper Modelling:



- Joint density function: (log-bilinear model)

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

- Word2vec and N tuple loss reminiscence

$$P(w^{(t+j)} \mid w^{(t)}) = P(D = 1 \mid w^{(t)}, w^{(t+j)}) \prod_{k=1,\, w_k \sim P(w)}^{K} P(D = 0 \mid w^{(t)}, w_k).$$

- Soft-Nearest Neighbors Loss (2019)

  https://arxiv.org/pdf/1902.01889.pdf

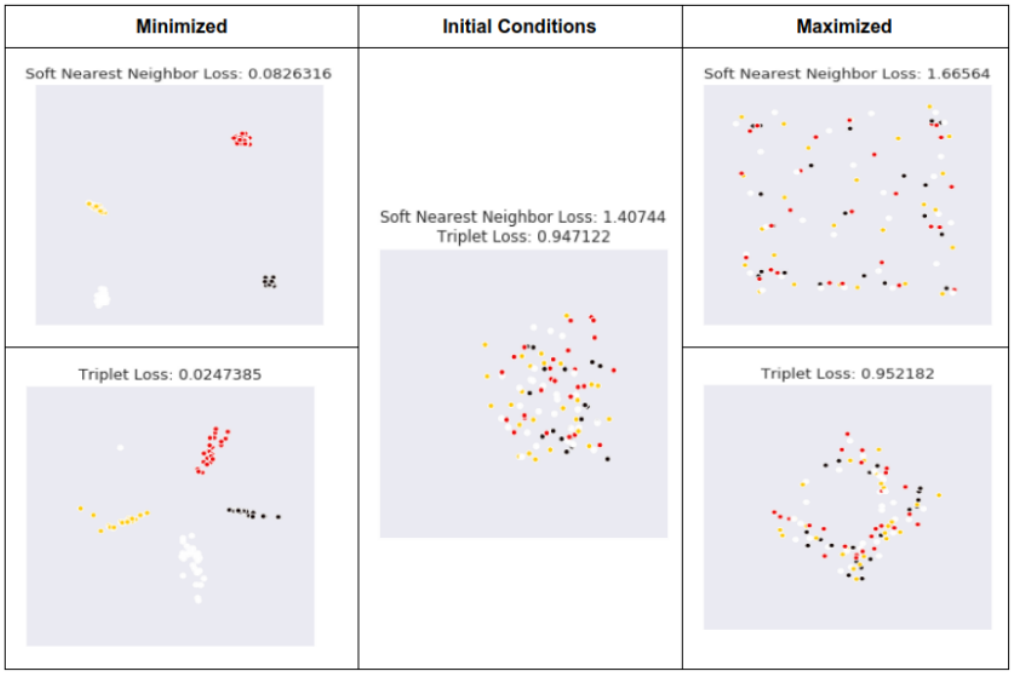
  - the entanglement boost KNN performance
  - temperature control the dominance of small distances

$$l_{sn}(x, y, T) = -\frac{1}{b} \sum_{i \in 1..b} \log \left(\frac{\sum\limits_{\substack{j \in 1..b \\ j \neq i \\ y_i = y_j}} e^{-\frac{||x_i - x_j||^2}{T}}}{\sum\limits_{\substack{k \in 1..b \\ k \neq i}} e^{-\frac{||x_i - x_k||^2}{T}}} \right)$$

$$\mathcal{L}_{\text{snn}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\sum_{i \neq j, y_i = y_j, j=1,\ldots,B} \exp(-f(\mathbf{x}_i, \mathbf{x}_j)/\tau)}{\sum_{i \neq k, k=1,\ldots,B} \exp(-f(\mathbf{x}_i, \mathbf{x}_k)/\tau)}$$

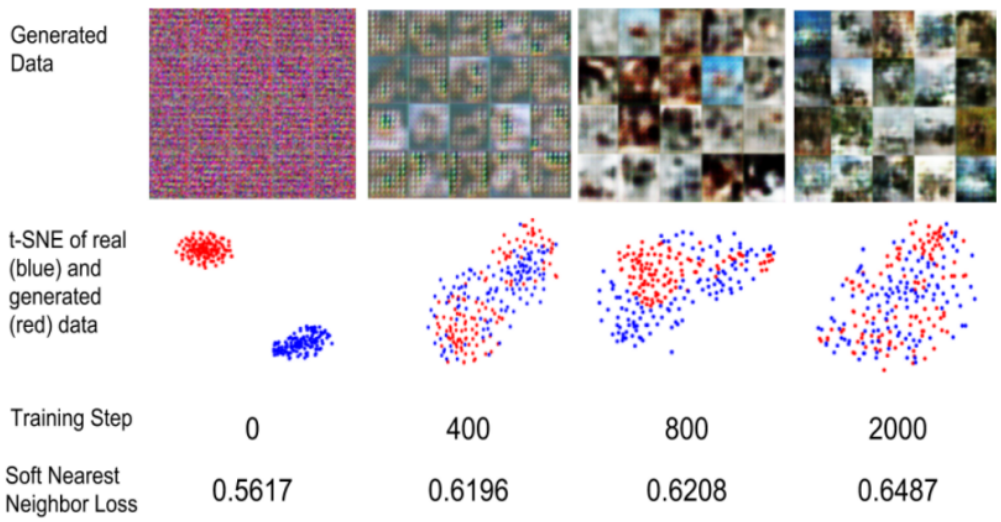
  - Analysis tools

- Compare with triplet:



- Resnet layers:

- GANs



- Regularizer for cross entropy:

CNN Model	Test Accuracy	Entangled	Baseline
MNIST	Best	**99.23%**	98.83%
	Average	**99.16%**	98.82%
Fashion-MNIST	Best	**91.48%**	90.42%
	Average	**91.06%**	90.25%
SVHN	Best	**88.81%**	87.63%
	Average	**89.90%**	89.71%
ResNet Model	**Test Accuracy**	**Entangled**	**Baseline**
CIFAR10	Best	**91.220%**	90.780%
	Average	**89.900%**	89.713%

Challenges for self-supervised

Contrastive learning has desirable properties:

- Robust
- Self-supervised training enabling

- Representation learning

yet, still face challenges:

- Non-trivial Data augmentation → What is proper way for NLP augmentation ?
- Sampling bias → Debias sampling strategies?

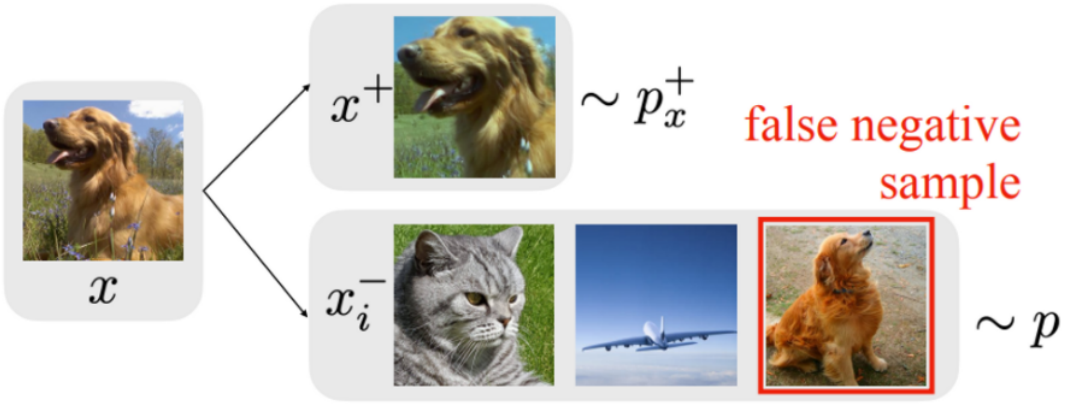


Figure 1: **"Sampling bias":** The common practice of drawing negative examples x_i^- from the data distribution $p(x)$ may result in x_i^- that are actually similar to x.

- Large batch-size

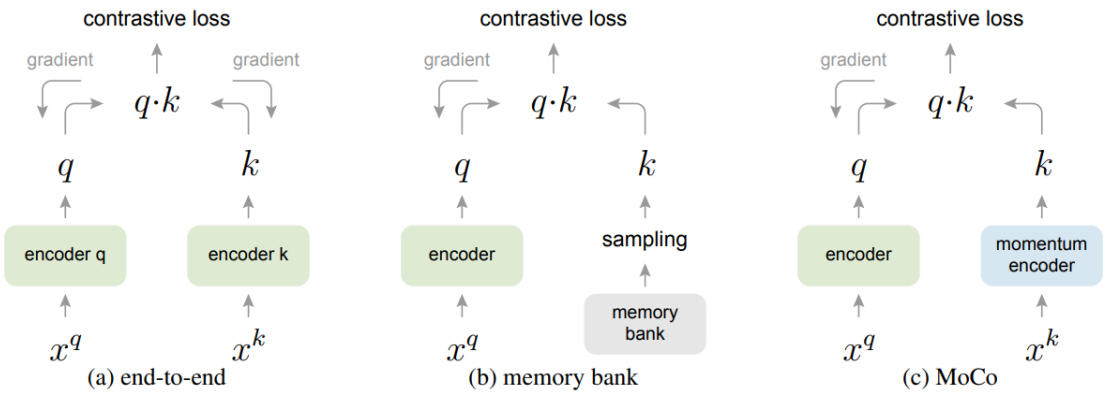
> *"We train with larger batch size (up to 32K) and longer (up to 3200 epochs)."*
>
> — Chen et al., SimCLR
>
> *"We use a very large minibatch size of 32,768."*
>
> — Radford et al., CLIP

- Negative mining → How to pick negative hard cases in self-supervised settings

Overall, we need to deduce:

- Computation-efficient methods for batching
- Unbiased, Efficient-negative-mining Sampling strategies
- Augmentation methods

Computation-efficient methods



Memory bank

1805.01978v1.pdf (arxiv.org)

- Formulation: NCE loss

  v is input image, v′ is image from other class

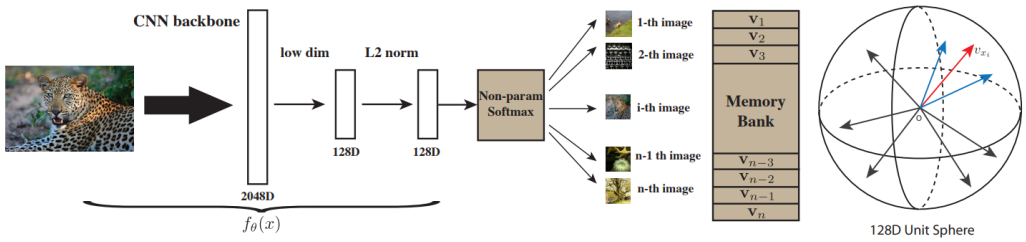
$$h(i, \mathbf{v}) := P(D = 1|i, \mathbf{v}) = \frac{P(i|\mathbf{v})}{P(i|\mathbf{v}) + mP_n(i)}. \quad (6)$$

Our approximated training objective is to minimize the negative log-posterior distribution of data and noise samples,

$$J_{NCE}(\boldsymbol{\theta}) = -E_{P_d}\left[\log h(i, \mathbf{v})\right] - m \cdot E_{P_n}\left[\log(1 - h(i, \mathbf{v}'))\right]. \quad (7)$$

- Methods
  - Memory bank

    Use bank to store the previous feature representation



  - Consistency training

    suppress the oscillation and fluctuation in training

$$J_{NCE}(\boldsymbol{\theta}) = -E_{P_d}\left[\log h(i, \mathbf{v}_i^{(t-1)}) - \lambda \|\mathbf{v}_i^{(t)} - \mathbf{v}_i^{(t-1)}\|_2^2\right]$$
$$-m \cdot E_{P_n}\left[\log(1 - h(i, \mathbf{v}'^{(t-1)}))\right]. \quad (10)$$

Momentum Contrast (MoCo)

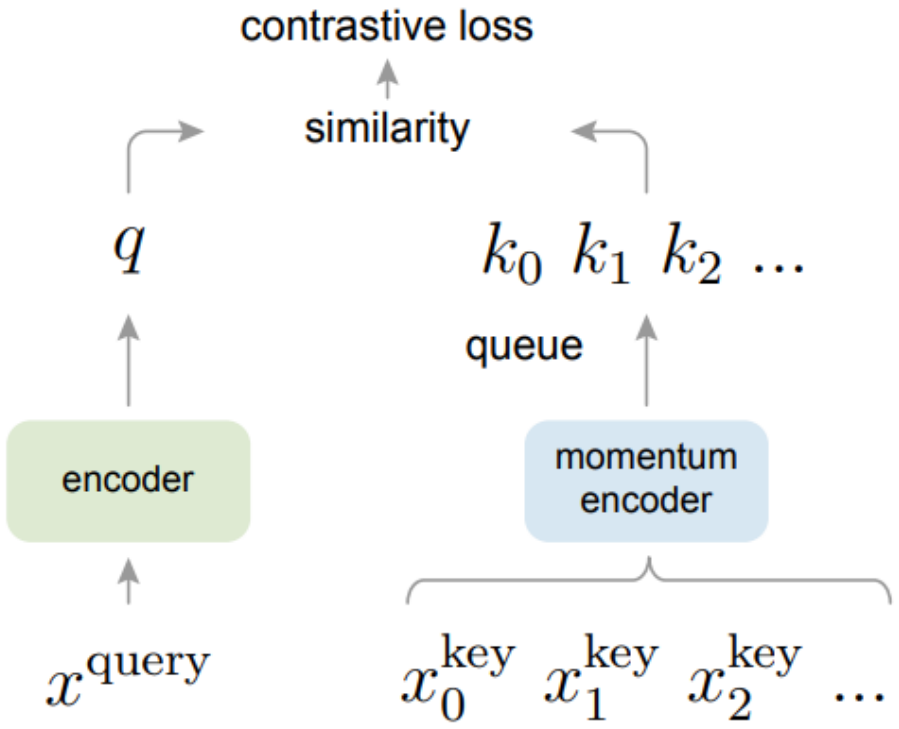
1911.05722.pdf (arxiv.org)

- Formulation: InfoNCE loss

  query q, positive key and k negative keys

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i/\tau)}$$

- Methods
  - Dynamic Dictionary look-up



  - Dictionary as a queue

> The samples in the dictionary are progressively replaced. The current mini-batch is enqueued to the dictionary, and the oldest mini-batch in the queue is removed. The dictionary always represents a sampled subset of all data, while the extra computation of maintaining this dictionary is manageable. Moreover, removing the oldest mini-batch can be beneficial, because its encoded keys are the most outdated and thus the least consistent with the newest ones.

  - Momentum update

    Stable training

Formally, denoting the parameters of f_{k} as θ_{k} and those of f_{q} as θ_{q}, we update θ_{k} by:

$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1-m)\theta_{\mathrm{q}}. \tag{2}$$

Sampling strategies

Principle

https://arxiv.org/pdf/2010.04592.pdf

In this section we describe our approach for hard negative sampling. We begin by asking *what makes a good negative sample?* To answer this question we adopt the following two guiding principles:

Principle 1. *q should only sample "true negatives" x_i^- whose labels differ from that of the anchor x.*

Principle 2. *The most useful negative samples are ones that the embedding currently believes to be similar to the anchor.*

Methods

SELF-SUPERVISED

- Vanilla setups (SimCLR, SimCSE)

  SimCLR https://arxiv.org/pdf/2002.05709.pdf

  SimCSE https://arxiv.org/pdf/2104.08821.pdf

$$\mathcal{L}^{self} = \sum_{i \in I} \mathcal{L}_i^{self} = -\sum_{i \in I} \log \frac{\exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_{j(i)}/\tau\right)}{\sum_{a \in A(i)} \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_a/\tau\right)}$$

  → Principles violation

- Debiased method 2007.00224.pdf (arxiv.org) (2020)
  - Motivation:

    Bias sampling in self-supervised: we use biased objective in unsupervised settings

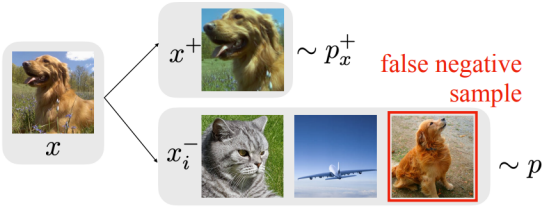


Figure 1: **"Sampling bias":** The common practice of drawing negative examples x_i^- from the data distribution $p(x)$ may result in x_i^- that are actually similar to x.

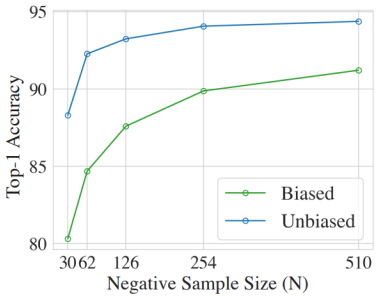


Figure 2: **Sampling bias leads to performance drop:** Results on CIFAR-10 for drawing x_i^- from $p(x)$ (biased) and from data with different labels, i.e., truly semantically different data (unbiased).

- Methods and analysis
  - Unbiased in supervised settings

$$L_{\text{Unbiased}}^N(f) = \mathbb{E}_{\substack{x\sim p, x^+\sim p_x^+ \\ x_i^-\sim p_x^-}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{N}\sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right]$$

  - Biased error
    - Widening the gap after training
    - Learning different representation

$$L_{\text{Biased}}^N(f) \geq L_{\text{Unbiased}}^N(f) + \mathbb{E}_{x\sim p}\left[0 \wedge \log \frac{\mathbb{E}_{x^+\sim p_x^+} \exp f(x)^\top f(x^+)}{\mathbb{E}_{x^-\sim p_x^-} \exp f(x)^\top f(x^-)} \right] - e^{3/2}\sqrt{\frac{\pi}{2N}}.$$

  - Problems

    we are sampling negatives from both pos and neg distribution

Let us assume the probability of anchor class c is uniform $\rho(c) = \eta^+$ and the probability of observing a different class is $\eta^- = 1 - \eta^+$.

- The probability of observing a positive example for $\mathbf{x}$ is $p_x^+(\mathbf{x}') = p(\mathbf{x}'|\mathbf{h}_{x'} = \mathbf{h}_x)$;
- The probability of getting a negative sample for $\mathbf{x}$ is $p_x^-(\mathbf{x}') = p(\mathbf{x}'|\mathbf{h}_{x'} \neq \mathbf{h}_x)$.

When we are sampling $\mathbf{x}^-$, we cannot access the true $p_x^-(\mathbf{x}^-)$ and thus $\mathbf{x}^-$ may be sampled from the (undesired) anchor class c with probability η^+. The actual sampling data distribution becomes:

$$p(\mathbf{x}') = \eta^+ p_x^+(\mathbf{x}') + \eta^- p_x^-(\mathbf{x}')$$

- Methods
  - Rejection Sampling directly from p and p+ → N positives required and Expensive calculation

An immediate approach would be to replace p_x^- in L_{Unbiased}^N with $p_x^-(x') = (p(x') - \tau^+ p_x^+(x'))/\tau^-$ and then use the empirical counterparts for p and p_x^+. The resulting objective can be estimated with samples from only p and p_x^+, but is computationally expensive for large N:

$$\frac{1}{(\tau^-)^N} \sum_{k=0}^{N} \binom{N}{k} (-\tau^+)^k \mathbb{E}_{\substack{x\sim p, x^+\sim p_x^+ \\ \{x_i^-\}_{i=1}^k \sim p_x^+ \\ \{x_i^-\}_{i=k+1}^N \sim p}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right], \tag{4}$$

  - Alternative asymptotic form

Lemma 2. *For fixed Q and $N \to \infty$, it holds that*

$$\mathbb{E}_{\substack{x\sim p, x^+\sim p_x^+ \\ \{x_i^-\}_{i=1}^N \sim {p_x^-}^N}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{N} \sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right] \tag{5}$$

$$\longrightarrow \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{\tau^-} \left(\mathbb{E}_{x^-\sim p}[e^{f(x)^T f(x^-)}] - \tau^+ \mathbb{E}_{v\sim p_x^+}[e^{f(x)^T f(v)}] \right)} \right]. \tag{6}$$

  - Empirical loss:

The empirical estimate of $\widetilde{L}^{Q}_{\text{Debiased}}$ is much easier to compute than the straightforward objective (5). With N samples $\{u_i\}_{i=1}^{N}$ from p and M samples $\{v_i\}_{i=1}^{M}$ from p_x^+, we estimate the expectation of the second term in the denominator as

$$g(x, \{u_i\}_{i=1}^{N}, \{v_i\}_{i=1}^{M}) = \max\left\{\frac{1}{\tau^-}\left(\frac{1}{N}\sum_{i=1}^{N} e^{f(x)^T f(u_i)} - \tau^+ \frac{1}{M}\sum_{i=1}^{M} e^{f(x)^T f(v_i)}\right),\ e^{-1/t}\right\}. \quad (7)$$

We constrain the estimator g to be greater than its theoretical minimum $e^{-1/t} \leq \mathbb{E}_{x^- \sim p_x^-} e^{f(x)^T f(x_i^-)}$ to prevent calculating the logarithm of a negative number. The resulting population loss with fixed N and M per data point is

$$L^{N,M}_{\text{Debiased}}(f) = \mathbb{E}_{\substack{x\sim p;\, x^+\sim p_x^+ \\ \{u_i\}_{i=1}^{N}\sim p^N \\ \{v_i\}_{i=1}^{N}\sim p_x^{+M}}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + N g\left(x, \{u_i\}_{i=1}^{N}, \{v_i\}_{i=1}^{M}\right)}\right], \quad (8)$$

- Results:

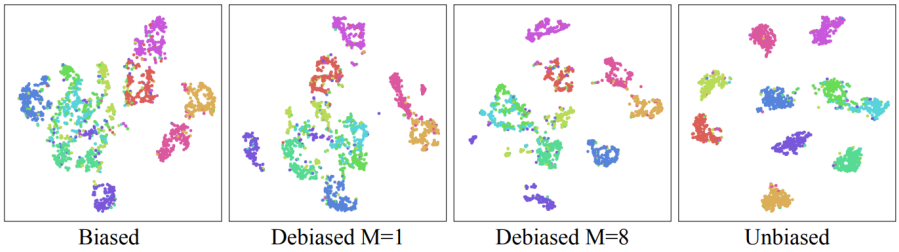


Figure 5: **t-SNE visualization of learned representations on CIFAR10.** Classes are indicated by colors. The debiased objective ($\tau^+ = 0.1$) leads to better data clustering than the (standard) biased loss; its effect is closer to the supervised unbiased objective.

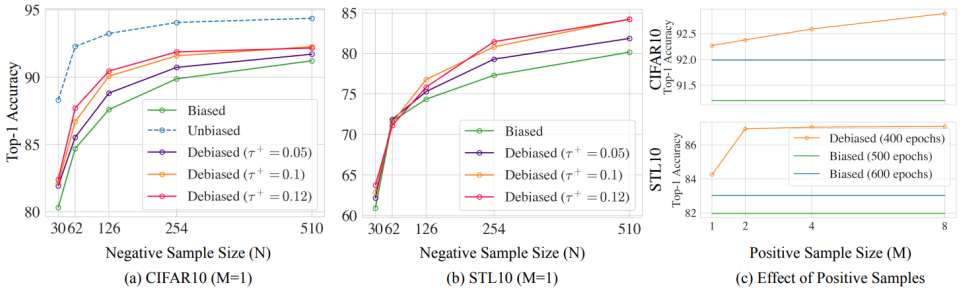


Figure 4: **Classification accuracy on CIFAR10 and STL10.** (a,b) Biased and Debiased ($M = 1$) SimCLR with different negative sample size N where $N = 2(BatchSize - 1)$. (c) Comparison with biased SimCLR with 50% more training epochs (600 epochs) while fixing the training epoch for Debiased ($M \geq 1$) SimCLR to 400 epochs.

- Hard mining 2010.04592.pdf (arxiv.org)
  - Motivation: Uphold the principle 2
  - Methods:

- Re-design the negative distribution:

conditioning on the event {h(x) ≠ h(x−)} which guarantees that (x, x−) correspond to different latent classes (Principle 1); 2) the concentration parameter β term controls the degree by which qβ up-weights points x− that have large inner product (similarity) to the anchor x (Principle 2). Since f lies on the surface of a hypersphere of radius 1/t, we have preferring points with large inner product equivalent to preferring points with small squared Euclidean distance.

$$q_\beta(x^-) = \tau^- q_\beta^-(x^-) + \tau^+ q_\beta^+(x^-),$$

$$\text{where} \quad q_\beta(x^-) \propto e^{\beta f(x)^\top f(x^-)} \cdot p(x^-)$$

$$q_\beta^-(x^-) := q_\beta(x^- | h(x) \neq h(x^-))$$

$$q_\beta^-(x^-) = \left(q_\beta(x^-) - \tau^+ q_\beta^+(x^-)\right)/\tau^-$$

- Loss function:

$$\mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{\tau^-}(\mathbb{E}_{x^-\sim q_\beta}[e^{f(x)^T f(x^-)}] - \tau^+ \mathbb{E}_{v\sim q_\beta^+}[e^{f(x)^T f(v)}])} \right].$$

- Results

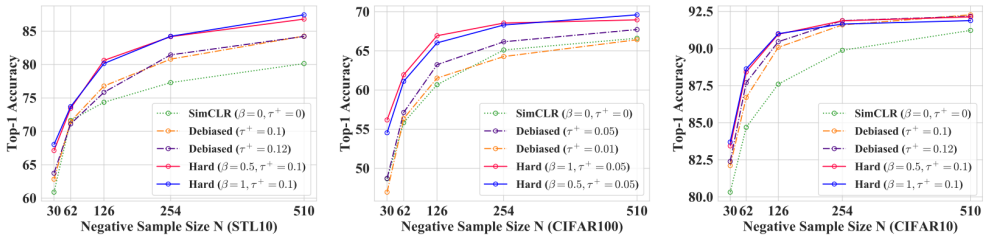


Figure 2: **Classification accuracy on downstream tasks.** Embeddings trained using hard, debiased, and standard ($\beta = 0, \tau^+ = 0$) versions of SimCLR, and evaluated using linear readout accuracy.

SUPERVISED

- Implicit hard mining (2020)

  2004.11362.pdf (arxiv.org)

  - Setup

    - *Data Augmentation* module, $Aug(\cdot)$. For each input sample, $\boldsymbol{x}$, we generate two random augmentations, $\tilde{\boldsymbol{x}} = Aug(\boldsymbol{x})$, each of which represents a different *view* of the data and contains some subset of the information in the original sample. Sec. 4 gives details of the augmentations.
    - *Encoder Network*, $Enc(\cdot)$, which maps $\boldsymbol{x}$ to a representation vector, $\boldsymbol{r} = Enc(\boldsymbol{x}) \in \mathcal{R}^{D_E}$. Both augmented samples are separately input to the same encoder, resulting in a pair of representation vectors. $\boldsymbol{r}$ is normalized to the unit hypersphere in $\mathcal{R}^{D_E}$ ($D_E = 2048$ in all our experiments in the paper). Consistent with the findings of [42, 52], our analysis and experiments show that this normalization improves top-1 accuracy.
    - *Projection Network*, $Proj(\cdot)$, which maps $\boldsymbol{r}$ to a vector $\boldsymbol{z} = Proj(\boldsymbol{r}) \in \mathcal{R}^{D_P}$. We instantiate $Proj(\cdot)$ as either a multi-layer perceptron [14] with a single hidden layer of size 2048 and output vector of size $D_P = 128$ or just a single linear layer of size $D_P = 128$; we leave to future work the investigation of optimal $Proj(\cdot)$ architectures. We again normalize the output of this network to lie on the unit hypersphere, which enables using an inner product to measure distances in the projection space. As in self-supervised contrastive learning [48, 3], we discard $Proj(\cdot)$ at the end of contrastive training. As a result, our inference-time models contain exactly the same number of parameters as a cross-entropy model using the same encoder, $Enc(\cdot)$.

  - Implicit hard-mining and loss-function design interpretability

    - Mining: Hard cases contribute more
    - Design:
      - Normalize remove bias in gradient in outside while inside doesn't
      - Outside gradient respect to z_i use mean representation of positives

$$\mathcal{L}_{out}^{sup} = \sum_{i \in I} \mathcal{L}_{out,i}^{sup} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_a / \tau\right)}$$

$$\mathcal{L}_{in}^{sup} = \sum_{i \in I} \mathcal{L}_{in,i}^{sup} = \sum_{i \in I} -\log \left\{ \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_a / \tau\right)} \right\}$$

$$\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{z}_i} = \frac{1}{\tau} \left\{ \sum_{p \in P(i)} \boldsymbol{z}_p (P_{ip} - X_{ip}) + \sum_{n \in N(i)} \boldsymbol{z}_n P_{in} \right\} \tag{4}$$

Here, $N(i) \equiv \{n \in A(i) : \tilde{\boldsymbol{y}}_n \neq \tilde{\boldsymbol{y}}_i\}$ is the set of indices of all negatives in the multiviewed batch, and $P_{ix} \equiv \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_x / \tau\right) / \sum_{a \in A(i)} \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_a / \tau\right)$. The difference between the gradients for the two losses is in X_{ip}.

$$X_{ip} = \begin{cases} \frac{\exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau)}{\sum_{p' \in P(i)} \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_{p'} / \tau\right)} & , \quad \text{if } \mathcal{L}_i^{sup} = \mathcal{L}_{in,i}^{sup} \\ \frac{1}{|P(i)|} & , \quad \text{if } \mathcal{L}_i^{sup} = \mathcal{L}_{out,i}^{sup} \end{cases} \tag{5}$$

Loss	Top-1
$\mathcal{L}_{out}^{sup}$	78.7%
$\mathcal{L}_{in}^{sup}$	67.4%

Table 1: ImageNet Top-1 classification accuracy for supervised contrastive losses on ResNet-50 for a batch size of 6144.

- Generalized
  - Generalized supervised settings

https://arxiv.org/pdf/2206.00384.pdf

Generalized for better augmentation

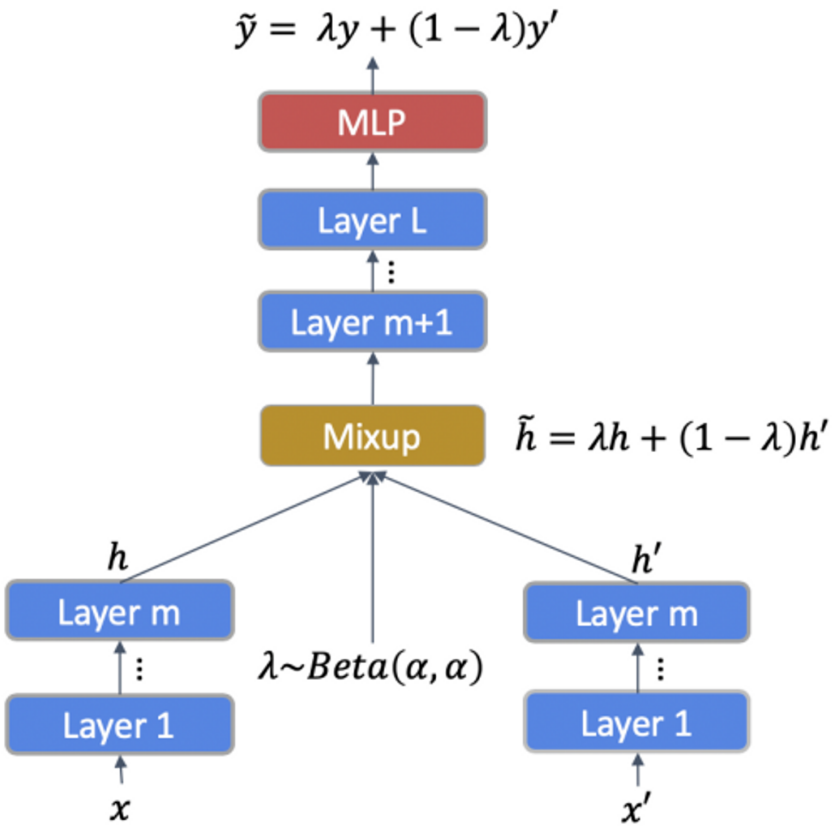


Figure 1: TMix takes in two text samples x and x' with labels y and y', mixes their hidden states h and h' at layer m with weight λ into $\tilde{h}$, and then continues forward passing to predict the mixed labels $\tilde{y}$.

z is feature of mixed version of x

Given this framework, we will briefly review the supervised contrastive loss [23] and its limitation. Then we show that our generalized supervised contrastive loss is a generalization of it and analyze how GenSCL is valid under generalized supervised contrastive loss. For the remainder of this paper, we denote a set of N randomly sampled batch data/label pairs as $\{\boldsymbol{x}_k, \boldsymbol{y}_k\}_{k=1...N}$ and corresponding multi-viewed batch as $\{\tilde{\boldsymbol{x}}_\ell, \tilde{\boldsymbol{y}}_\ell\}_{\ell=1...2N}$, where $\tilde{\boldsymbol{x}}_{2k}$ and $\tilde{\boldsymbol{x}}_{2k-1}$ are two different views of $\boldsymbol{x}_k$ $(k = 1...N)$. Within multi-viewed batch, let $i \in I \equiv \{1...2N\}$ be the index of an *anchor* and $A(i) \equiv I \setminus \{i\}$ is the set of *contrasts* w.r.t. an anchor i.

$$\mathcal{L}^{\text{gen}} = \sum_{i \in I} \mathcal{L}_i^{\text{gen}} = \sum_{i \in I} \frac{1}{|A(i)|} \text{CE}(Y(i), Z(i))$$
$$= \sum_{i \in I} \frac{-1}{|A(i)|} \sum_{j \in A(i)} \text{sim}(\mathbf{y}_i, \mathbf{y}_j) \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_j / \tau)}{\sum_{k \in A(i)} \exp(\mathbf{z}_i \cdot \mathbf{z}_k / \tau)}$$

- naïve *anchor* vs. naïve *contrasts*: the setting in original supervised contrastive learning, which is the special case in our loss when $\text{sim}(\boldsymbol{y}_i, \boldsymbol{y}_p) = 1$.
- mixed *anchor* vs. naïve *contrasts*: it can be represented by convex combination of supervised contrastive loss similar to [24].
- mixed *anchor* vs. mixed *contrasts*: supervised contrastive loss is not available to cover this scenario.

- Knowledge distillation

$$\mathcal{L} = \mathcal{L}_{\text{CE}}(\mathbf{p}^s, \mathbf{y}) + \alpha_{\text{kd}} \mathcal{L}_{\text{KL}}(\mathbf{p}^s, \mathbf{p}^t)$$

$$\mathcal{L} = \sum_{i \in I} \frac{1}{|A(i)|} \left\{ \text{CE}(Y(i), Z(i)) + \alpha_{\text{kd}} \text{CE}(P^t(i), Z(i)) \right\}$$
$$= \sum_{i \in I} \frac{-1}{|A(i)|} \sum_{j \in A(i)} \left(\text{sim}(\mathbf{y}_i, \mathbf{y}_j) + \alpha_{\text{kd}} \text{sim}(\mathbf{p}_i^t, \mathbf{p}_j^t) \right) \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_j / \tau)}{\sum_{k \in A(i)} \exp(\mathbf{z}_i \cdot \mathbf{z}_k / \tau)}$$

Augmentation methods

Principle

i) What are some label-preserving transformations, that can be applied to text, to compose useful augmented samples?

ii) Are these transformations complementary in nature, and can we find some strategies to consolidate them for producing more diverse augmented examples?

iii) How can we incorporate the obtained augmented samples into the training process in an effective and principled manner?

Methods

- Text space
  - Lexical Editing (token)
    - Common practice

Synonym Replacement

Random Insertion

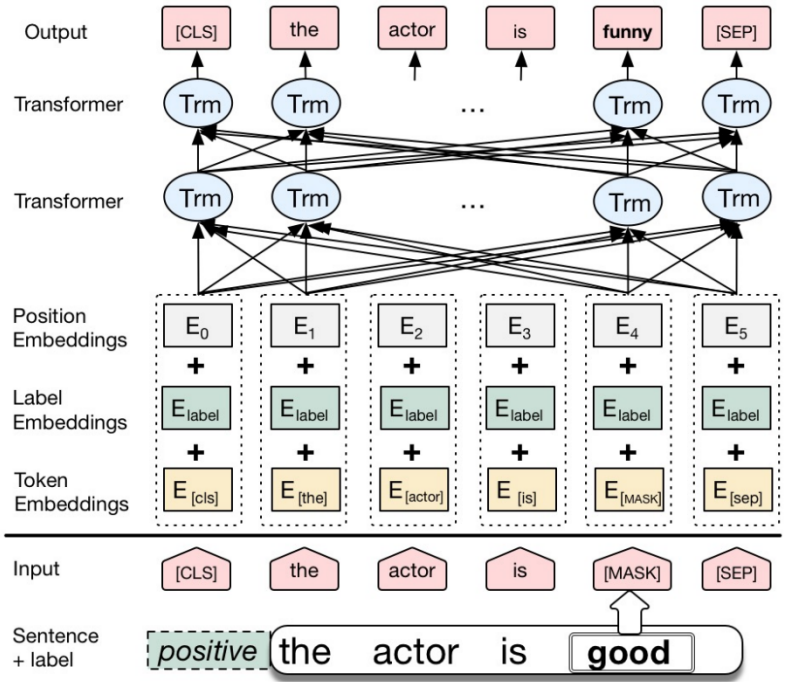
Random Swap

Random Deletion

Operation	Sentence
None	A sad, superior human comedy played out on the back roads of life.
SR	A ***lamentable***, superior human comedy played out on the ***backward*** road of life.
RI	A sad, superior human comedy played out on ***funniness*** the back roads of life.
RS	A sad, superior human comedy played out on ***roads*** back ***the*** of life.
RD	A sad, superior human out on the roads of life.

    - c-BERT reconstruction instead of BERT

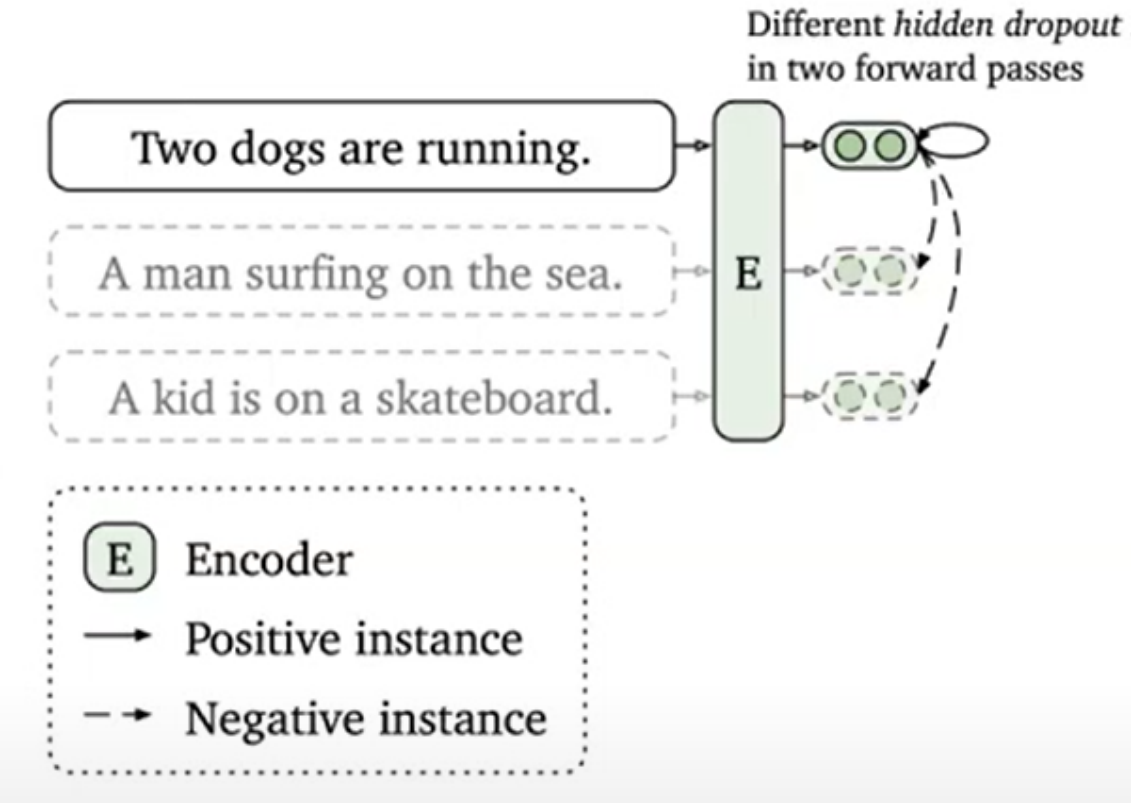
1812.06705.pdf (arxiv.org)



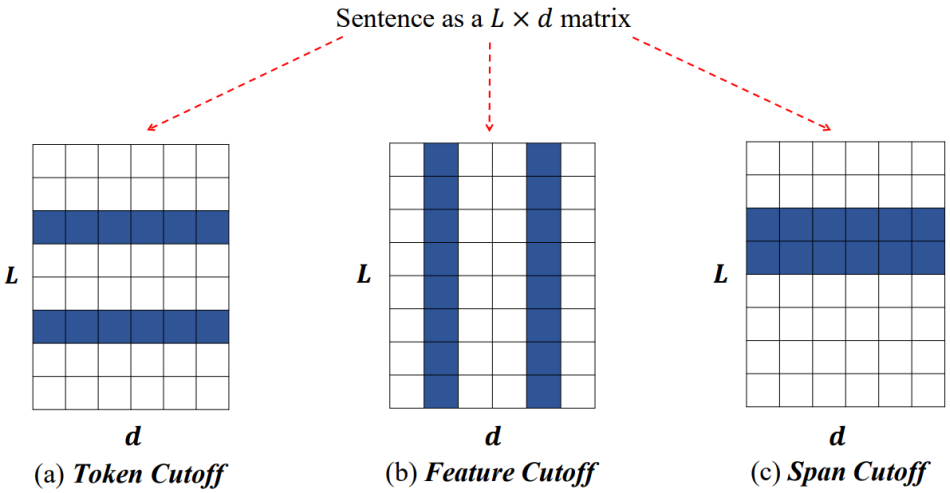
- Sentence:

  Back translation

- Embedding space
  - Dropout (SimCSE) 2104.08821.pdf (arxiv.org)



  - Cutoff 2009.13818.pdf (arxiv.org)



- Mixup 2004.12239.pdf (arxiv.org)

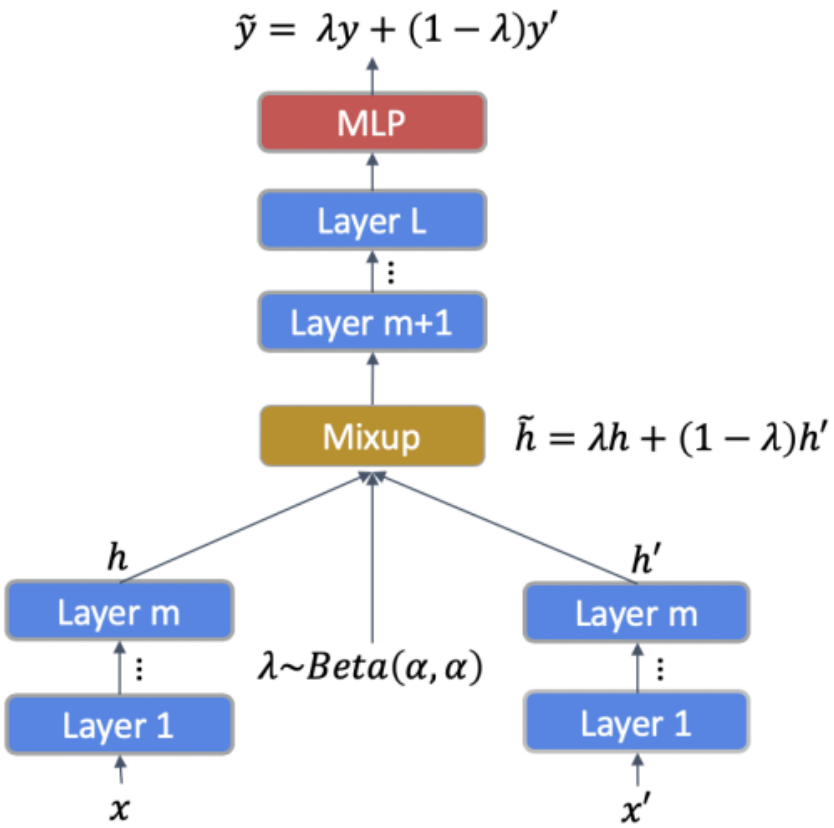
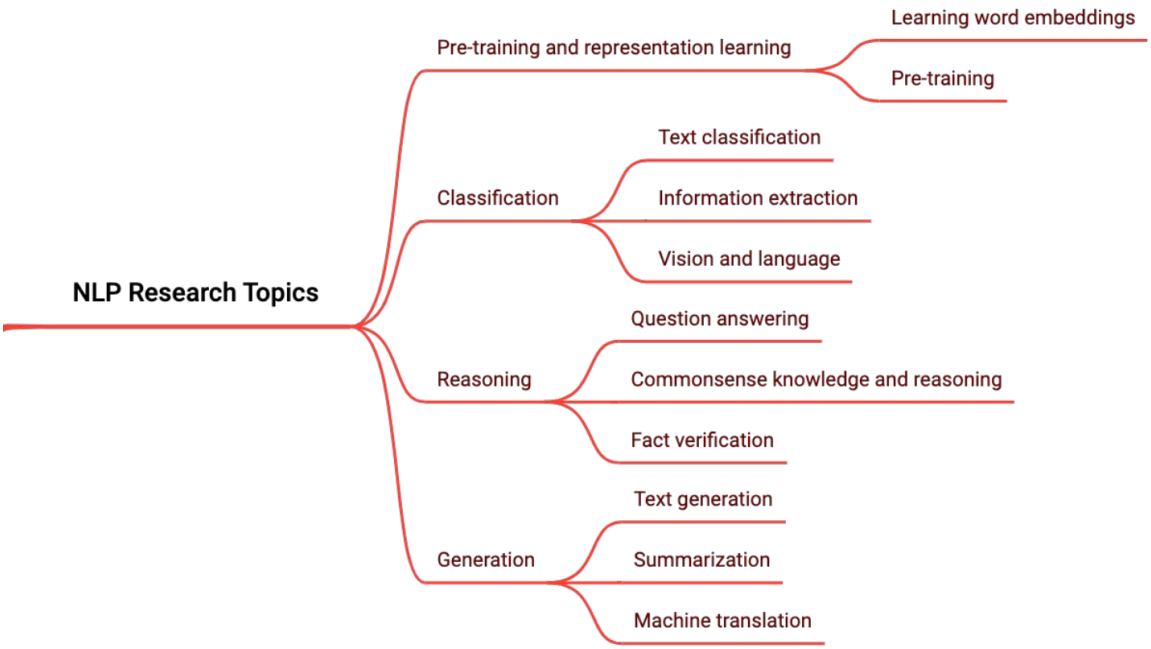


Figure 1: TMix takes in two text samples x and x' with labels y and y', mixes their hidden states h and h' at layer m with weight λ into $\tilde{h}$, and then continues forward passing to predict the mixed labels $\tilde{y}$.

Some applications in NLP

Overview


NLP Research Topics
Pre-training and representation learning
Learning word embeddings
Pre-training
Classification
Text classification
Information extraction
Vision and language
Reasoning
Question answering
Commonsense knowledge and reasoning
Fact verification
Generation
Text generation
Summarization
Machine translation

	supervised	self-supervised
end-task agnostic		COCO-LM: Meng, 2021 CLEAR: Wu, 2020 Electric: Clark, 2020 CLESS: Rethmeier, 2020 CoDA: Qu, 2020 MixText: Chen, 2020 DeCLUTR: Giorgi, 2020 OLFMLM: Aroca, 2020 CERT: Fang, 2020 CPC: Oord, 2018 QT: Logeswaran, 2018 Word2vec: Mikolov, 2013
end-task specific	👁 CLIP: Radford, 2020 CLESS: Rethmeier, 2020 CSS: Klein, 2020 👁 TCN: Jian, 2019 UST: Uehara, 2020 GILE: Pappas, 2019	CONPONO: Iter, 2020 👁 ALIGN: Jia, 2019 BIT: Duan, 2019

Text classification

- Cutoff

2009.13818.pdf (arxiv.org)

  - Properties
    - Multi-view learning
    - Consensus between perturbed samples
  - Loss function

Suppose there are N cutoff samples constructed from the same original input x (with a label of y), which are denoted by $x^1_{\text{cutoff}}, x^2_{\text{cutoff}}, ..., x^N_{\text{cutoff}}$, respectively. Since their semantic meanings are approximately preserved with the cutoff operation, we may incorporate them into the training objective by encouraging the model to make similar predictions across different samples. The training objective can be written as:

$$\mathcal{L} = \mathcal{L}_{\text{ce}}(x, y) + \alpha \sum_{i=1}^{N} \mathcal{L}_{\text{ce}}(x^i_{\text{cutoff}}, y) + \beta \mathcal{L}_{\text{divergence}}(x, x^1_{\text{cutoff}}, x^2_{\text{cutoff}}, ..., x^N_{\text{cutoff}}, y)$$

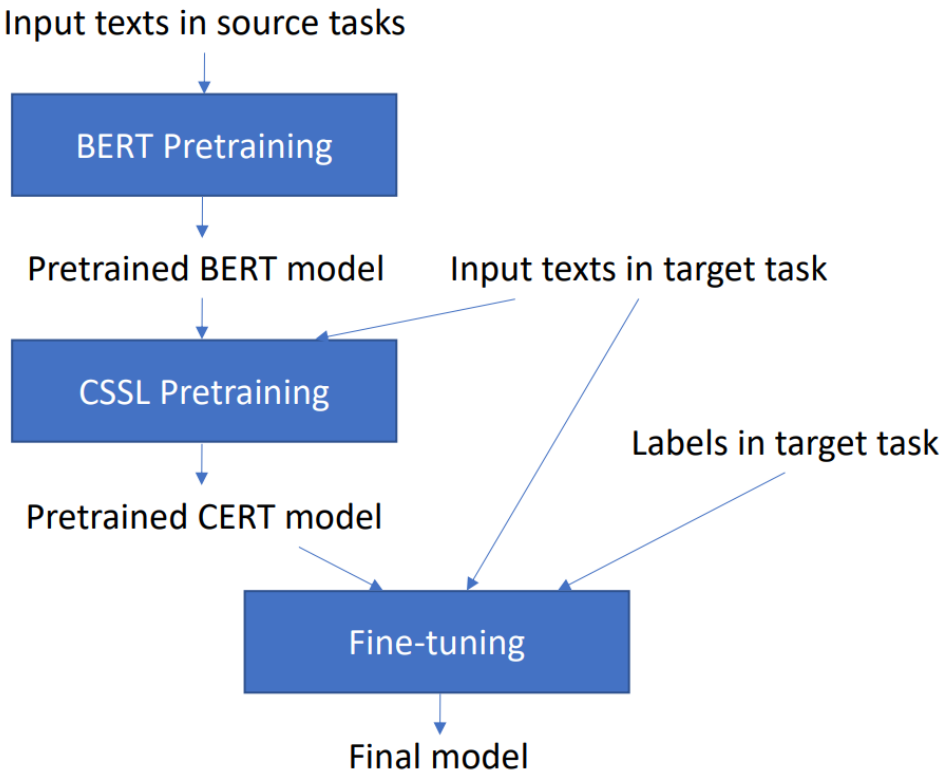
$$p_{avg} = \frac{1}{N+1} \sum_{i=0}^{N} p(y|x^i_{\text{cutoff}})$$

$$\mathcal{L}_{\text{divergence}} = \frac{1}{N+1} \sum_{i=0}^{N} \text{KL}[p(y|x^i_{\text{cutoff}})||p_{avg}]$$

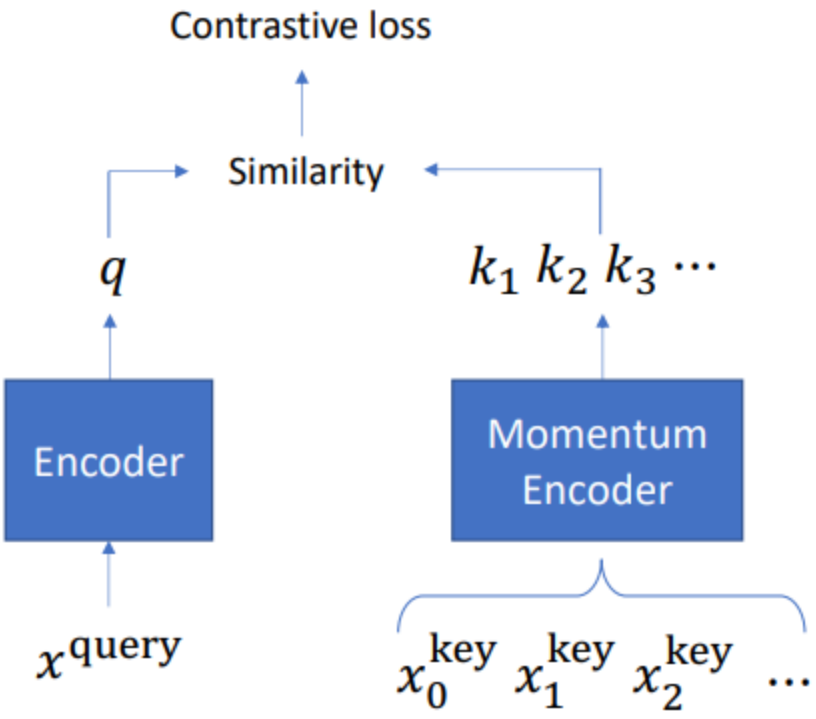
- CERT

2005.12766.pdf (arxiv.org)

- BERT + Backtranslation augmentation



- Loss function (maintain k augmented sentences, momentum update)



$$-\log \frac{\exp(\text{sim}(f_q(q;\theta_q), f_k(k_+;\theta_k))/\tau)}{\sum_{i=1}^{K} \exp(\text{sim}(f_q(q;\theta_q), f_k(k_i;\theta_k))/\tau)}$$

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q$$

- CoDA

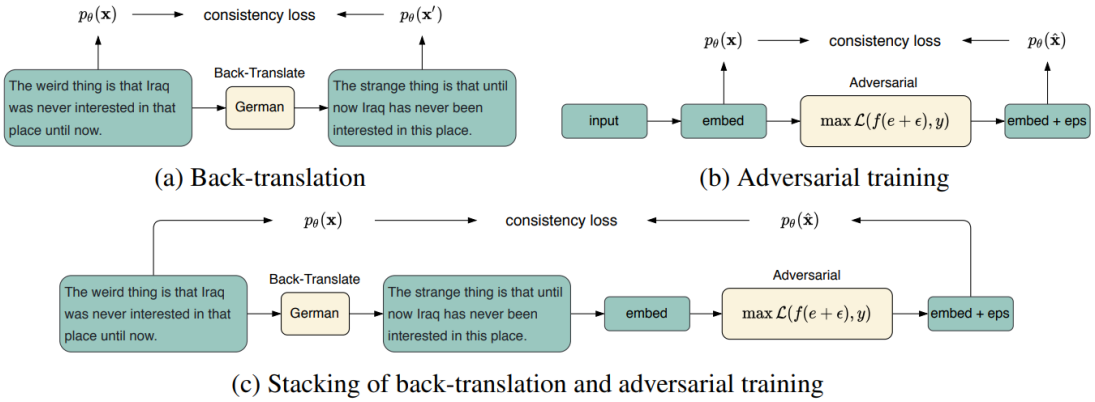
https://arxiv.org/pdf/2010.08670.pdf

- Properties

  Consistency training with:

  - Data augmentation
  - Adversarial training

<Adversarial self-supervised CL : 1f1baa5b8edac74eb4eaa329f14a0361-Paper.pdf (neurips.cc) >



(a) Back-translation (b) Adversarial training

(c) Stacking of back-translation and adversarial training

- Loss function
  - Consistency loss

    Local regularization

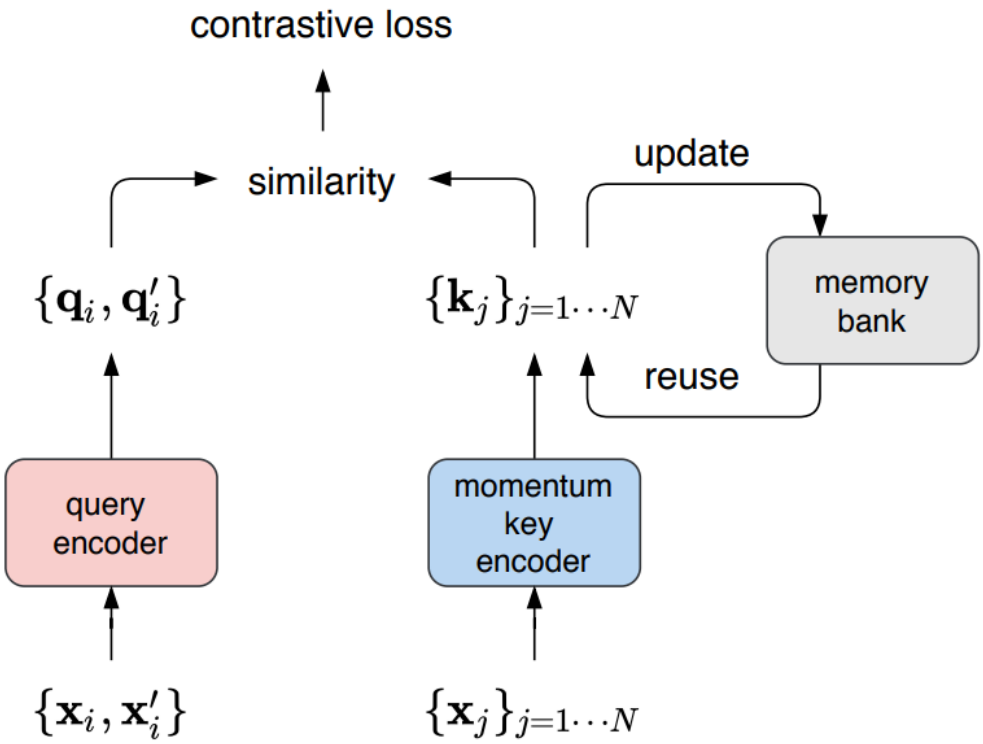
$$\boldsymbol{x}_i' = \text{BackTrans}(\boldsymbol{x}_i),\ \hat{\boldsymbol{x}}_i \approx \text{argmax}_{\tilde{\boldsymbol{x}}_i} \mathcal{R}_{\text{AT}}(\boldsymbol{x}_i', \tilde{\boldsymbol{x}}_i, y_i)\ ,$$
$$\mathcal{L}_{\text{consistency}}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i, y_i) = \mathcal{L}\big(p_\theta(\boldsymbol{x}_i), y_i\big) + \alpha \mathcal{L}(p_\theta(\hat{\boldsymbol{x}}_i), y_i) + \beta \mathcal{R}_{\text{CS}}(p_\theta(\boldsymbol{x}_i), p_\theta(\hat{\boldsymbol{x}}_i))$$

$$\mathcal{R}_{\text{CS}}(p_\theta(\boldsymbol{x}_i), p_\theta(\hat{\boldsymbol{x}}_i)) = \frac{1}{2}\big(\text{KL}(p_\theta(\boldsymbol{x}_i) \| M) + \text{KL}(p_\theta(\hat{\boldsymbol{x}}_i)) \| M)\big)$$

$$\text{where } M = (p_\theta(\boldsymbol{x}_i) + p_\theta(\hat{\boldsymbol{x}}_i))/2$$

  - Contrastive-Regularization loss

    Utilize the memory to store the previous representation of dictionary key



$$\mathcal{R}_{\text{contrast}}(\boldsymbol{x}_i, \boldsymbol{x}_i', \mathcal{M}) = \mathcal{R}_{\text{CT}}(\boldsymbol{q}_i, \boldsymbol{k}_i, \mathcal{M}) + \mathcal{R}_{\text{CT}}(\boldsymbol{q}_i', \boldsymbol{k}_i, \mathcal{M}),$$
$$\mathcal{R}_{\text{CT}}(\boldsymbol{q}_i, \boldsymbol{k}_i, \mathcal{M}) = -\log \frac{\exp(\text{sim}(\boldsymbol{q}_i, \boldsymbol{k}_i)/\tau)}{\sum_{\boldsymbol{k}_j \in \mathcal{M} \bigcup \{\boldsymbol{k}_i\}} \exp(\text{sim}(\boldsymbol{q}_i, \boldsymbol{k}_j)/\tau)},$$

- Loss function

$$\theta^* = \text{argmin}_\theta \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}} \mathcal{L}_{\text{consistency}}(\boldsymbol{x}_i, \boldsymbol{x}_i', y_i) + \lambda \mathcal{R}_{\text{contrast}}(\boldsymbol{x}_i, \boldsymbol{x}_i', \mathcal{M}) \ .$$

- Dual CL

  2201.08702.pdf (arxiv.org)

  - Properties:
    - Label aware augmentation
    - Dual losses

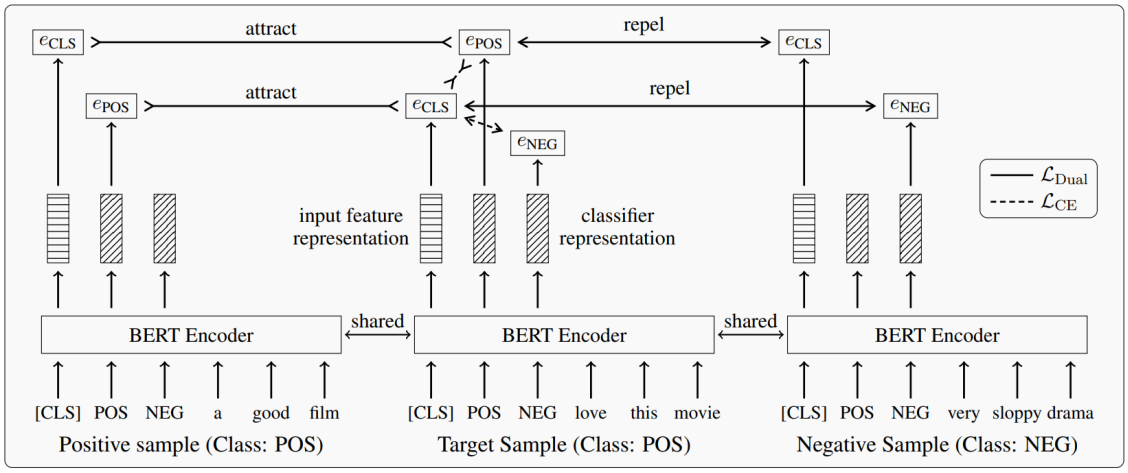


Figure 1: The framework of the proposed dual contrastive learning (DualCL).

- Loss function

$$\mathcal{L}_z = \frac{1}{N} \sum_{i \in \mathcal{I}} \frac{1}{|\mathcal{P}_i|} \sum_{p \in \mathcal{P}_i} -\log \frac{\exp(\boldsymbol{\theta}_p^* \cdot \boldsymbol{z}_i / \tau)}{\sum_{a \in \mathcal{A}_i} \exp(\boldsymbol{\theta}_a^* \cdot \boldsymbol{z}_i / \tau)}$$

$$\mathcal{L}_\theta = \frac{1}{N} \sum_{i \in \mathcal{I}} \frac{1}{|\mathcal{P}_i|} \sum_{p \in \mathcal{P}_i} -\log \frac{\exp(\boldsymbol{\theta}_i^* \cdot \boldsymbol{z}_p / \tau)}{\sum_{a \in \mathcal{A}_i} \exp(\boldsymbol{\theta}_i^* \cdot \boldsymbol{z}_a / \tau)}$$

$$\mathcal{L}_{\text{CE}} = \frac{1}{N} \sum_{i \in \mathcal{I}} -\log \frac{\exp(\boldsymbol{\theta}_i^* \cdot \boldsymbol{z}_i)}{\sum_{k \in \mathcal{K}} \exp(\boldsymbol{\theta}_i^k \cdot \boldsymbol{z}_i)}$$

$$\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{Dual}}$$

$$\hat{y}_i = \arg\max_k (\boldsymbol{\theta}_i^k \cdot \boldsymbol{z}_i)$$

- Results

  GLUE Benchmarks

  GLUE Explained: Understanding BERT Through Benchmarks · Chris McCormick (mccormickml.com)

  - CutOff

    Testbed: Roberta

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Avg
Testbed: ***RoBERTa-base***									
RoBERTa-base	87.6	92.8	91.9	78.7	94.8	-/90.2	63.6	91.2	-
ALUM	88.1	93.1	**92.0**	80.2	95.3	90.9/-	63.6	91.1	86.8
Token Cutoff	88.2	93.3	91.9	81.2	95.1	**91.1**/-	64.1	91.2	87.0
Feature Cutoff	88.2	93.2	**92.0**	81.6	95.3	90.7/-	63.6	91.2	87.0
Span Cutoff	**88.4**	**93.4**	**92.0**/-	**82.3**	**95.4**	**91.1**	**64.7**	91.2	**87.3**
Testbed: ***RoBERTa-large***									
RoBERTa-large	90.2	94.7	92.2	86.6	96.4	-/90.9	68.0	92.4	-
PGD	90.5	94.9	92.5	87.4	96.4	90.9/-	69.7	92.4	89.3
FreeAT	90.0	94.7	92.5	86.7	96.1	90.7/-	68.8	92.4	89.0
FreeLB	90.6	95.0	**92.6**	88.1	96.8	**91.4**/-	71.1	92.7	89.8
ALUM	90.9	95.1	92.2	87.3	96.6	91.1/-	68.2	92.1	89.2
SMART	91.1	**95.6**	92.4	**92.0**	96.9	89.2/92.1	70.6	92.8	90.0
Back-translation	91.1	95.3	92.0	91.7	97.1	90.9/93.5	69.4	92.8	90.1
Token Cutoff	91.0	95.3	92.3	90.6	96.9	90.9/93.2	70.0	92.5	90.0
Feature Cutoff	90.9	95.2	92.4	90.9	**97.1**	90.9/93.4	71.1	92.4	90.1
Span Cutoff	**91.1**	95.3	92.4	91.0	96.9	**91.4/93.8**	**71.5**	**92.8**	**90.3**

  - CERT

    Testbed: BERT

	CoLA	RTE	QNLI	STS-B	MRPC	WNLI	SST-2	MNLI (m/mm)	QQP
BERT (reported in Lan et al.2019 median)	60.6	70.4	92.3	90.0	-	-	93.2	86.6/-	91.3
BERT (our run, median)	59.8	71.4	91.9	89.8	91.1	56.3	93.4	86.3/86.2	91.2
CERT (ours, median)	62.1	72.2	92.3	89.8	91.0	56.3	93.6	86.3/86.2	91.4
BERT (our run, best)	60.9	72.1	92.1	90.1	92.5	56.3	94.4	86.4/86.3	91.4
CERT (ours, best)	62.9	74.0	92.5	90.6	92.5	56.3	93.9	86.6/86.5	91.7
XLNet	63.6	83.8	93.9	91.8	-	-	95.6	89.8/-	91.8
ERNIE2	65.4	85.2	94.3	92.3	-	-	96.0	89.1/-	92.5
RoBERTa	68.0	86.6	94.7	92.4	-	-	96.4	90.2/-	92.2
ALBERT	71.4	89.2	95.3	93.0	-	-	96.9	90.8/-	92.2

- CoDA

  Testbed: Roberta

Method	MNLI-m/ mm (Acc)	QQP (Acc/F1)	QNLI (Acc)	SST-2 (Acc)	MRPC (Acc/F1)	CoLA (Mcc)	RTE (Acc)	STS-B (P/S)	Avg
RoBERTa-large	90.2/-	92.2/-	94.7	96.4	-/90.9	68	86.6	92.4/-	88.9
Back-Trans	91.1/90.4	92/-	95.3	97.1	90.9/93.5	69.4	91.7	92.8/92.6	90.4
Cutoff	91.1/-	92.4/-	95.3	96.9	91.4/93.8	71.5	91.0	92.8/-	90.6
FreeLB	90.6/-	**92.6**/-	95	96.7	91.4/-	71.1	88.1	92.7/-	-
SMART	91.1/**91.3**	92.4/89.8	**95.6**	96.9	89.2/92.1	70.6	92	92.8/92.6	90.4
R3F	91.1/**91.3**	92.4/**89.9**	95.3	97.0	91.6/-	71.2	88.5	-	-
CoDA	**91.3**/90.8	92.5/**89.9**	95.3	**97.4**	**91.9/94**	**72.6**	**92.4**	**93/92.7**	**91.1**

- Dual CL

  Testbed: BERT-family

Model	Method	SST-2	SUBJ	TREC	PC	CR	Avg.
		10% training data					
BERT	CE	86.05±0.24	93.05±0.25	93.29±0.22	91.09±0.23	86.58±0.29	90.01±0.25
	CE+SCL [Gunel *et al.*, 2021]	86.64±0.17	93.20±0.16	93.70±0.24	91.46±0.23	88.14±0.26	90.63±0.21
	CE+CL	87.66±0.28	94.27±0.21	94.20±0.29	91.67±0.27	87.72±0.32	91.10±0.27
	DualCL w/o $\mathcal{L}_{\text{dual}}$	87.90±0.19	93.50±0.18	94.01±0.31	91.83±0.22	88.13±0.30	91.07±0.24
	DualCL	**88.40±0.20**	**94.50±0.21**	**94.93±0.23**	**92.36±0.16**	**89.01±0.28**	**91.84±0.22**
RoBERTa	CE	90.91±0.23	94.03±0.19	94.51±0.21	90.65±0.20	92.06±0.27	92.43±0.22
	CE+SCL [Gunel *et al.*, 2021]	91.00±0.29	94.37±0.30	94.85±0.24	90.82±0.20	92.32±0.25	92.67±0.26
	CE+CL	91.04±0.17	94.47±0.19	**95.68±0.26**	91.90±0.14	92.55±0.28	93.13±0.21
	DualCL w/o $\mathcal{L}_{\text{dual}}$	92.48±0.18	94.40±0.17	95.18±0.16	91.50±0.14	92.88±0.20	93.29±0.17
	DualCL	**92.67±0.21**	**94.78±0.19**	95.36±0.18	**92.17±0.20**	**93.24±0.24**	**93.64±0.20**
		full training data					
BERT	CE	91.19±0.23	96.40±0.19	97.21±0.20	95.06±0.14	92.09±0.24	94.39±0.20
	CE+SCL [Gunel *et al.*, 2021]	91.71±0.20	96.25±0.19	97.58±0.16	95.26±0.13	93.06±0.20	94.77±0.18
	CE+CL	91.95±0.22	96.72±0.15	97.80±0.14	95.21±0.11	93.19±0.19	94.97±0.16
	DualCL w/o $\mathcal{L}_{\text{dual}}$	91.99±0.15	96.78±0.13	97.70±0.19	95.30±0.15	93.14±0.19	94.97±0.16
	DualCL	**92.40±0.17**	**97.20±0.17**	**98.22±0.17**	**95.56±0.14**	**93.78±0.17**	**95.43±0.16**
RoBERTa	CE	94.09±0.24	96.60±0.21	97.10±0.20	95.10±0.19	93.41±0.24	95.26±0.22
	CE+SCL [Gunel *et al.*, 2021]	93.65±0.20	96.73±0.23	97.18±0.19	95.35±0.19	93.60±0.17	95.30±0.20
	CE+CL	94.33±0.21	97.04±0.17	**97.52±0.15**	95.32±0.10	93.49±0.25	95.54±0.18
	DualCL w/o $\mathcal{L}_{\text{dual}}$	94.41±0.23	96.79±0.24	97.10±0.25	95.30±0.12	94.01±0.25	95.52±0.22
	DualCL	**94.91±0.17**	**97.34±0.19**	97.40±0.17	**95.59±0.12**	**94.39±0.23**	**95.93±0.18**

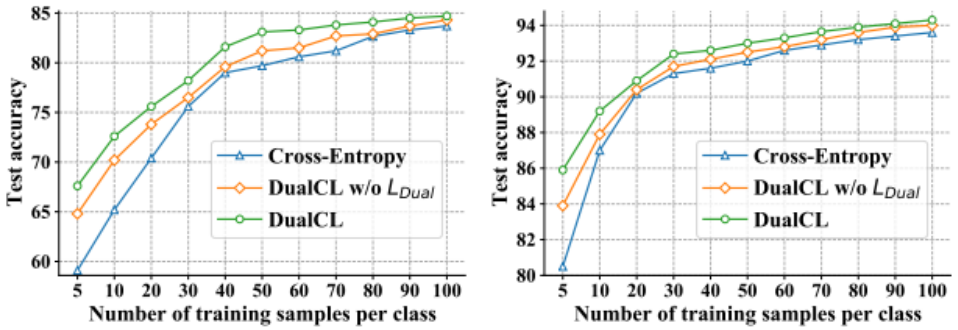


Figure 3: Test accuracy on the SST2 (left) and SUBJ (right) datasets in low-resource scenarios of DualCL with or without $\mathcal{L}_{\text{Dual}}$ compared with cross-entropy using BERT. The batch size here is set to 4.

Summary

- Pros
  - Robust learning technique
  - Representation learning
  - Self-supervised settings enabling

- Cons
  - Computation
  - Proper settings: Supervised ?

Case study

- Dataset: UIT-NLP student reviews

  Statistics:
  - train: 13000
  - test: 3166
- Method: Dual-loss
- Test-bed: phoBERT
- Advantages:
  - Innovative label-sentence augmentation
  - Robust learning methods requires less proportion of data
  - Not required for external classifier

- Results on UIT-NLP dataset
  - CE: 50 epochs

	precision	recall	f1-score	support
0	0.92	0.92	0.92	1590
1	0.89	0.92	0.90	1409
2	0.49	0.37	0.42	167
accuracy			0.89	3166
macro avg	0.77	0.73	0.75	3166
weighted avg	0.88	0.89	0.89	3166

- Dual-CL: 30 epochs

	precision	recall	f1-score	support
0	0.93	0.92	0.92	1590
1	0.88	0.96	0.92	1409
2	0.63	0.29	0.40	167
accuracy			0.90	3166
macro avg	0.82	0.72	0.75	3166
weighted avg	0.89	0.90	0.89	3166

Further

- Machine translation
- Question answering
- Sentence embedding

> https://github.com/ryanzhumich/Contrastive-Learning-NLP-Papers