Advanced Deep Learning

DATA.ML.230

Contrastive Learning

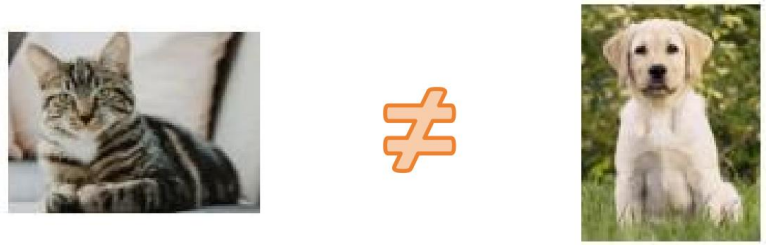
What is Contrastive Learning?

It involves training a model to differentiate between similar and dissimilar pairs of data points

What is Contrastive Learning?

It involves training a model to differentiate between similar and dissimilar pairs of data points by:

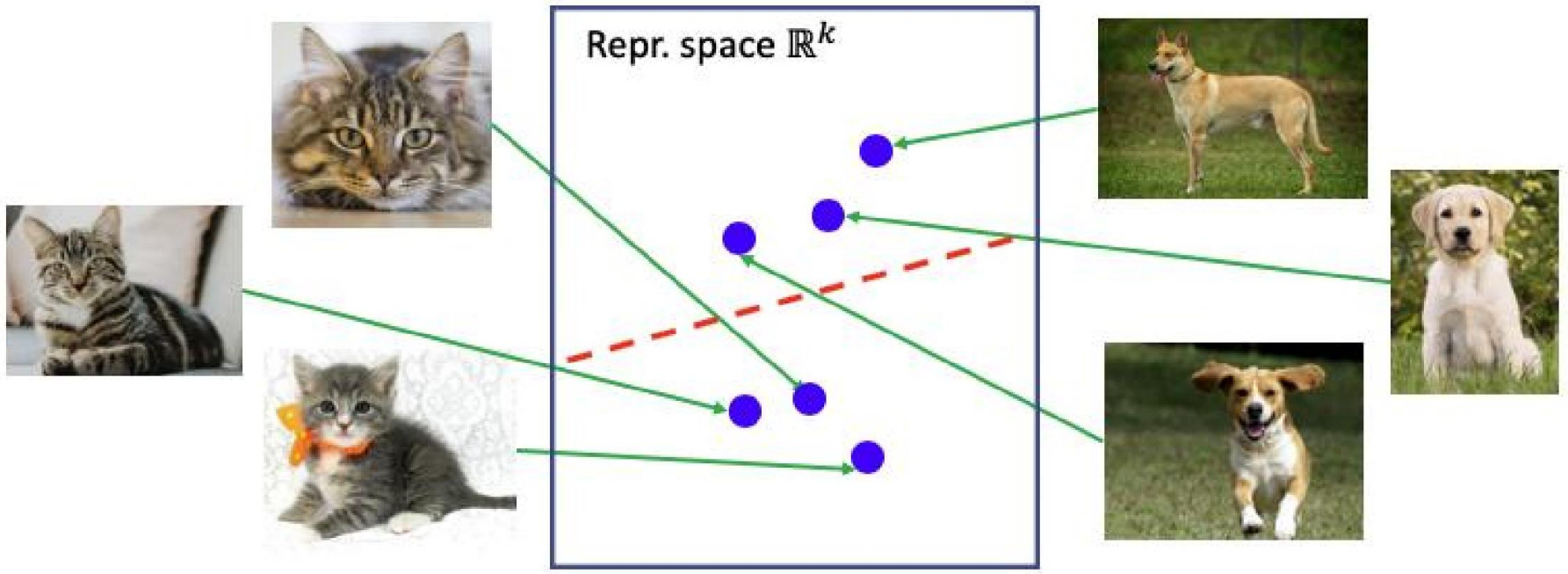
- Maximizing similarity between same-class (or similar) data points
- Minimizing similarity between different-class (or dissimilar) data points



What is Contrastive Learning?

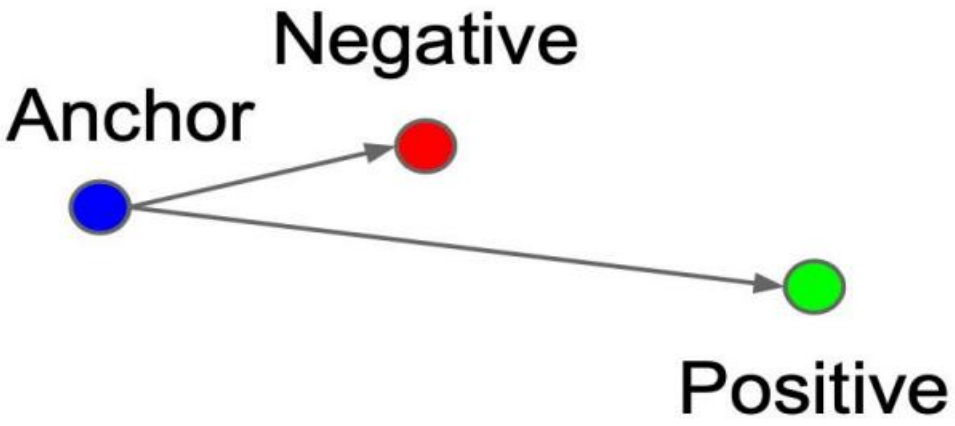
It involves training a model to differentiate between similar and dissimilar pairs of data points by:

- Maximizing similarity between same-class (or similar) data points
- Minimizing similarity between different-class (or dissimilar) data points



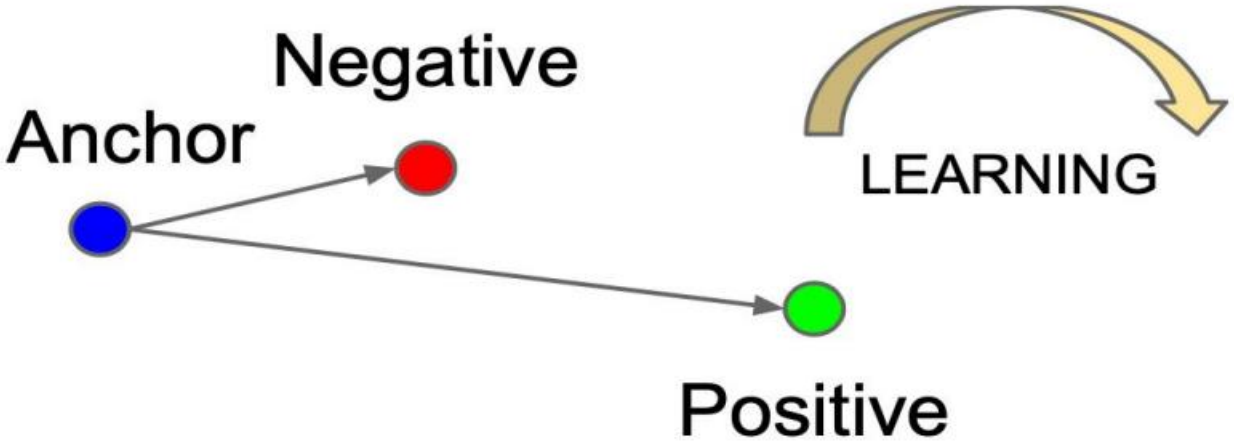
Goal

Contrastive Learning usually involves “anchors”, and “positive” and “negative” data points



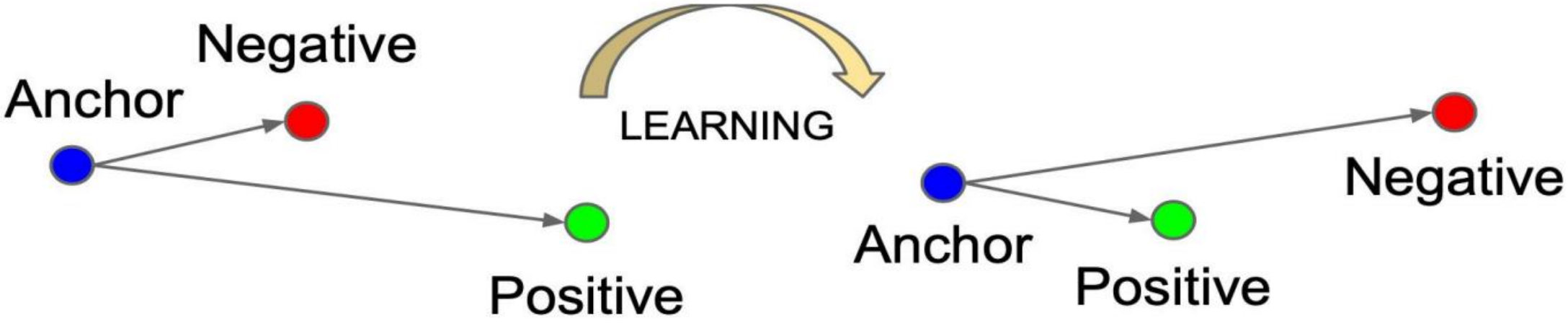
Goal

Contrastive Learning usually involves “anchors”, and “positive” and “negative” data points



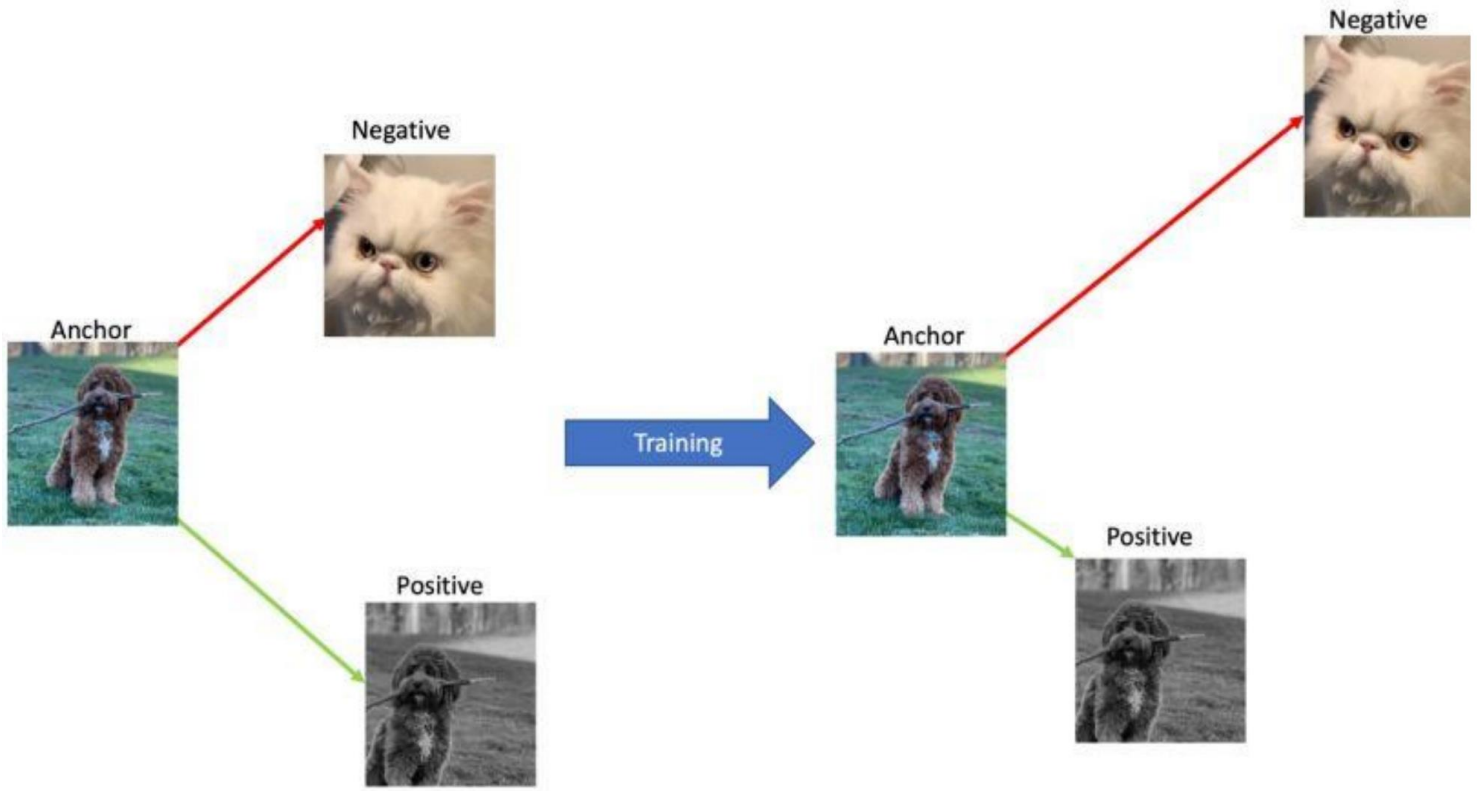
Goal

Contrastive Learning usually involves “anchors”, and “positive” and “negative” data points

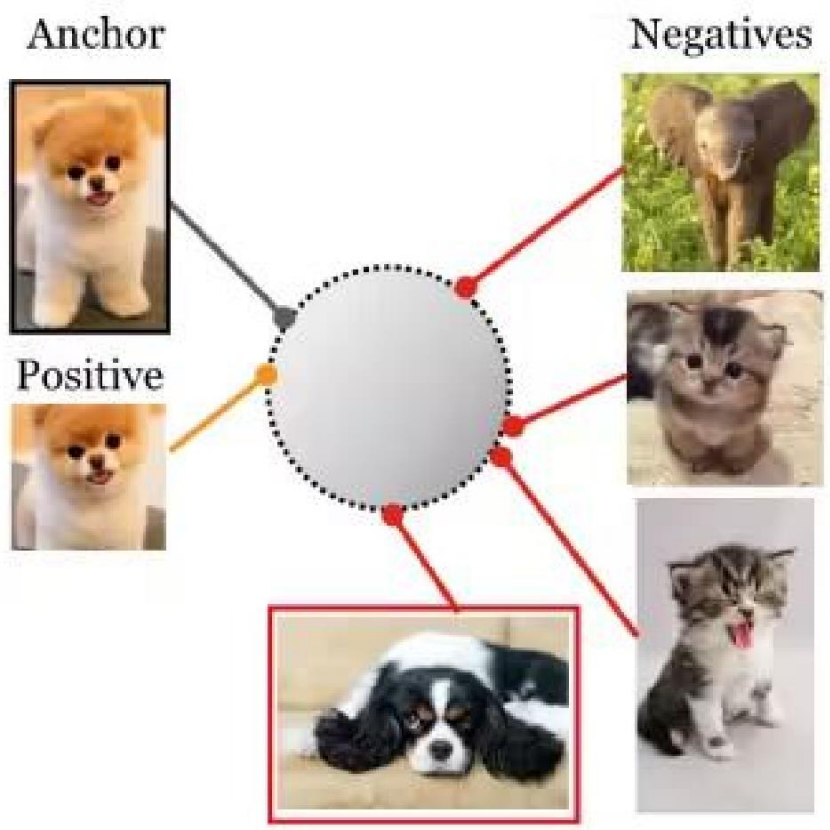


Goal

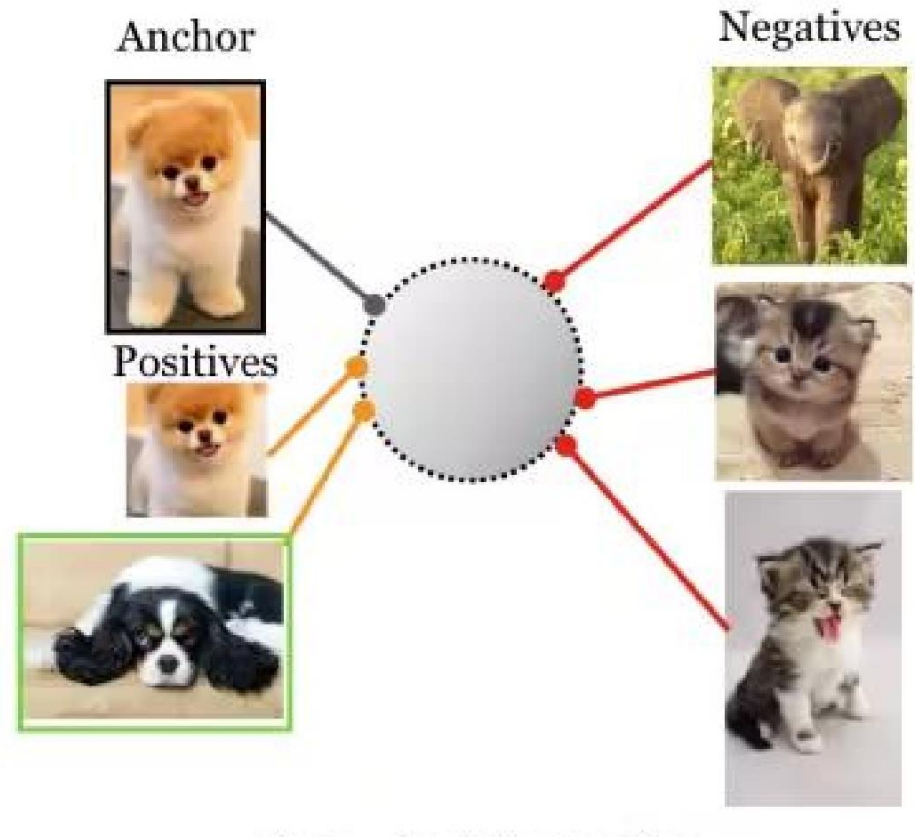
Contrastive Learning usually involves “anchors”, and “positive” and “negative” data points



Self-supervised vs. supervised CL



Self-supervised Contrastive Learning



Supervised Contrastive Learning

Why is it useful?

It enables us to train (deep) models to learn useful representations effectively without the need for explicit labeling.

Why is it useful?

It enables us to train (deep) models to learn useful representations effectively without the need for explicit labeling:

- There exist domains/applications where labeled data is scarce or/and expensive to obtain

Why is it useful?

It enables us to train (deep) models to learn useful representations effectively without the need for explicit labeling:

- There exist domains/applications where labeled data is scarce or/and expensive to obtain
- Labeling can lead to specialized representations that are not transferable to other tasks using the same data, e.g., face recognition vs. facial expression recognition tasks

Why is it useful?

It enables us to train (deep) models to learn useful representations effectively without the need for explicit labeling:

- There exist domains/applications where labeled data is scarce or/and expensive to obtain
- Labeling can lead to specialized representations that are not transferable to other tasks using the same data, e.g., face recognition vs. facial expression recognition tasks
- It allows for artificially populating the number of input examples/data used for training

Improved Representation Learning

Contrastive Learning targets learning representations that:

- Are invariant to changes in the input data that we want to preserve (e.g., class)
- Are sensitive to changes that alter the meaning or category of the data

Positive: I^+

Anchor: I^a

Negative: I^-

Improved Representation Learning

Contrastive Learning targets learning representations that:

- Are invariant to changes in the input data that we want to preserve (e.g., class)
- Are sensitive to changes that alter the meaning or category of the data

Leads to robust and generalizable models that perform better on a wide range of tasks (e.g., detection, classification, segmentation)

Positive: I^+

Anchor: I^a

Negative: I^-

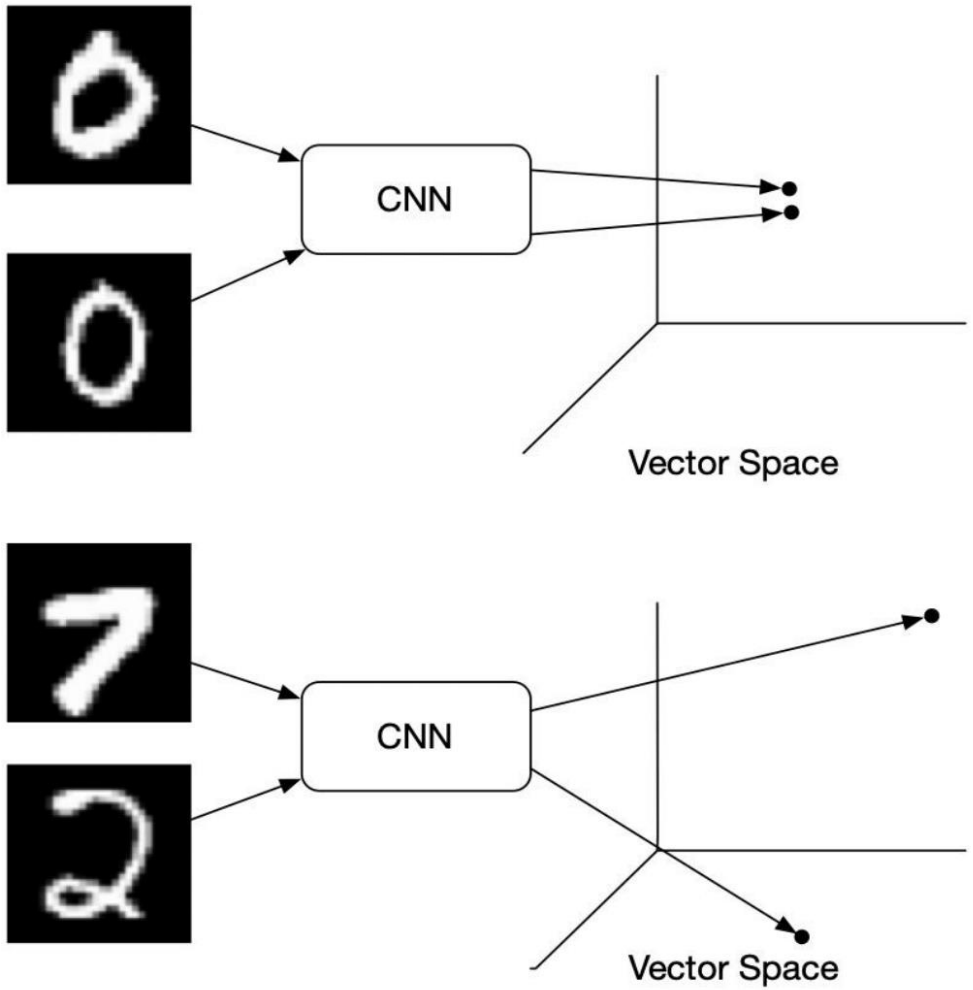
Contrastive Loss

Key idea:

- Positive pairs → Similar embeddings
- Negative pairs → Dissimilar embeddings

This is achieved by updating the parameters of the model to:

- Pull similar points together
- Push dissimilar points away



Contrastive Loss

2.3. The Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

Hadsell, Chopra, LeCun, “Dimensionality Reduction by Learning an Invariant Mapping”, CVPR 2006

Contrastive Loss

2.3. The Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

Hadsell, Chopra, LeCun, “Dimensionality Reduction by Learning an Invariant Mapping”, CVPR 2006

Contrastive Loss

2.3. The Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

Hadsell, Chopra, LeCun, “Dimensionality Reduction by Learning an Invariant Mapping”, CVPR 2006

Contrastive Loss

2.3. The Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

Hadsell, Chopra, LeCun, “Dimensionality Reduction by Learning an Invariant Mapping”, CVPR 2006

Contrastive Loss

2.3. The Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

Hadsell, Chopra, LeCun, “Dimensionality Reduction by Learning an Invariant Mapping”, CVPR 2006

Contrastive Loss

2.3. The Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

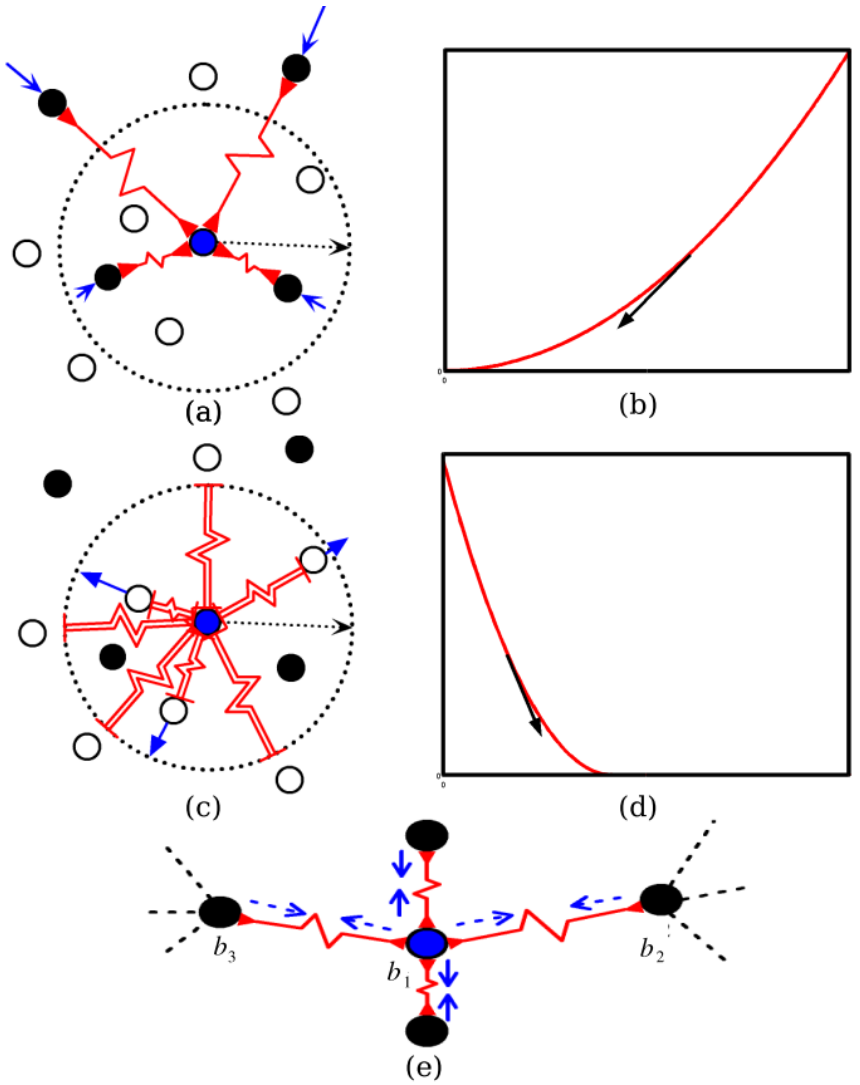
Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$



Hadsell, Chopra, LeCun, “Dimensionality Reduction by Learning an Invariant Mapping”, CVPR 2006

Contrastive Loss

Positive pair:

$$L(x_i, x_j) = \left\| M(x_i) - M(x_j) \right\|_2^2$$

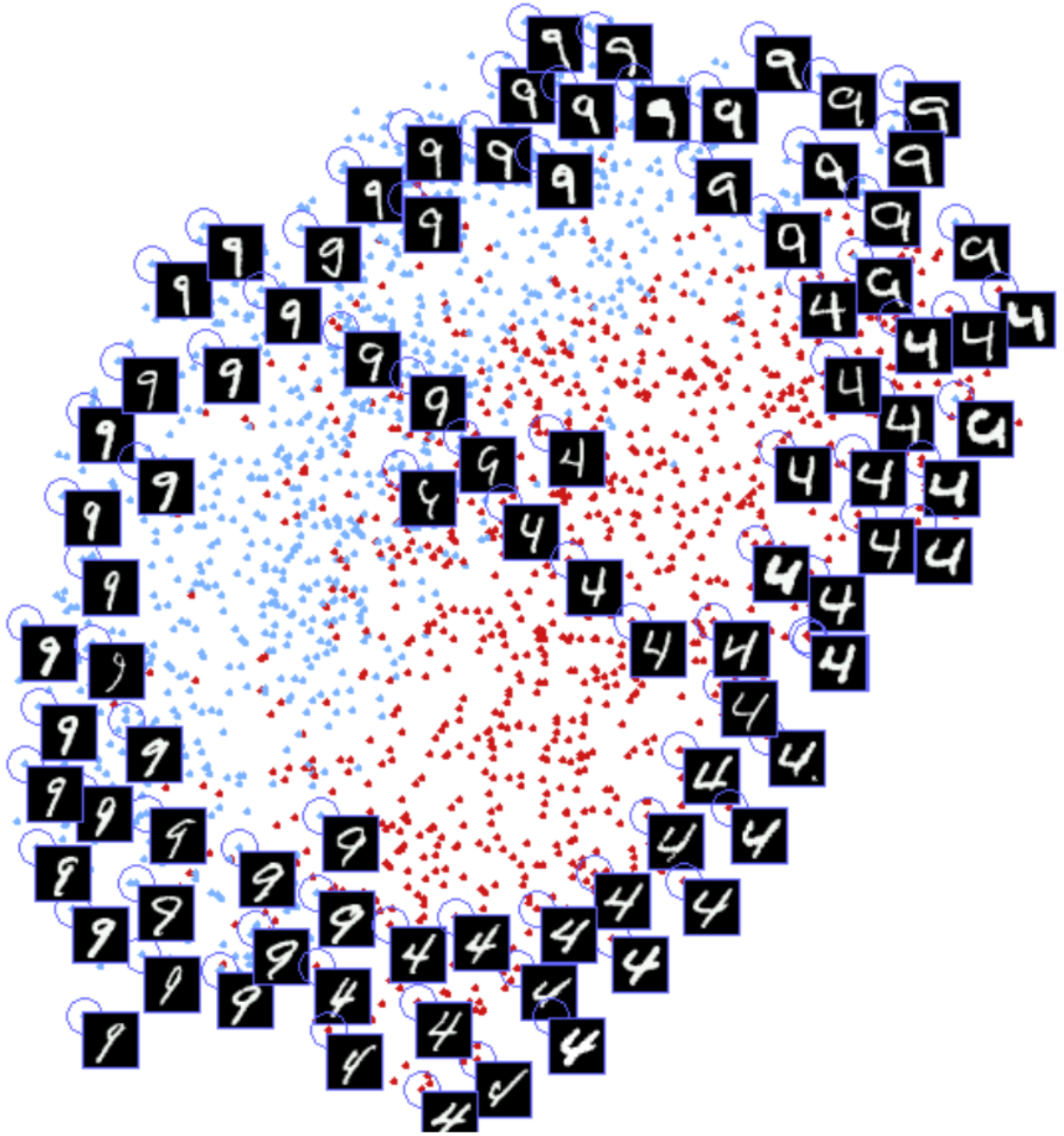
Embedding function M(•)
(Neural Network)

Negative pair:

$$L(x_i, x_j) = max(0, \varepsilon - \left\| M(x_i) - M(x_j) \right\|_2)^2$$

Margin
parameter

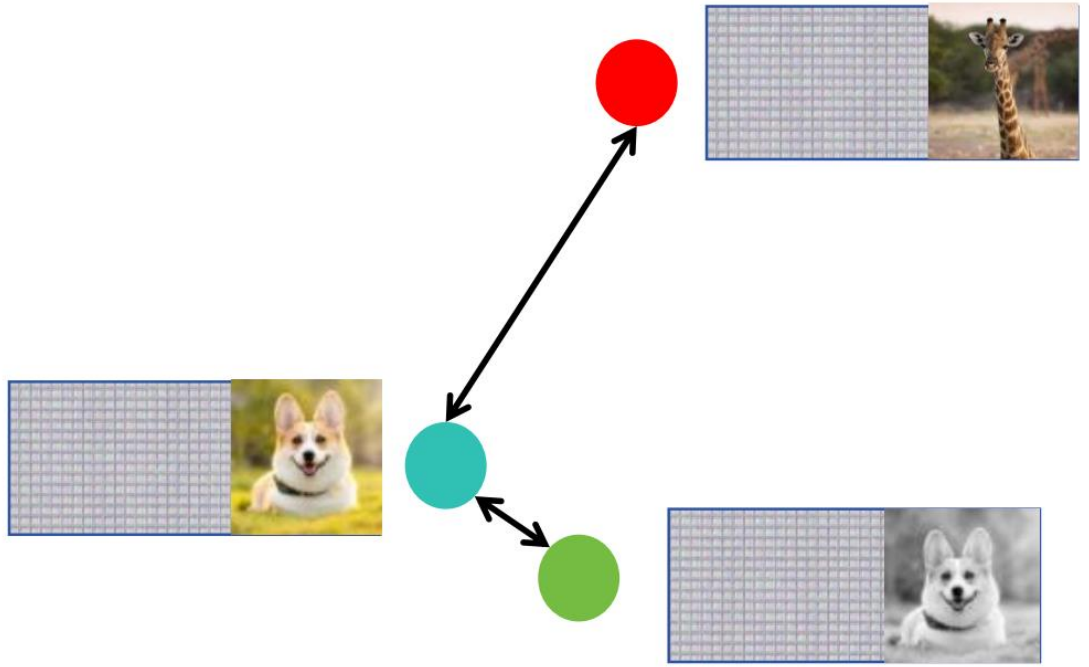
Contrastive Loss



Hadsell, Chopra, LeCun, “Dimensionality Reduction by Learning an Invariant Mapping”, CVPR 2006

Triplet Loss

$$L(x_i^a, x_i^p, x_i^n) = \left\| M(x_i^a) - M(x_i^p) \right\|_2^2 + \alpha - \left\| M(x_i^a) - M(x_i^n) \right\|_2^2$$



Chechik, Sharma, Shalit, Bengio, “Large Scale Online Learning of Image Similarity Through Ranking”, JMLR 2010

(N+1)-tuple Loss

Positive example

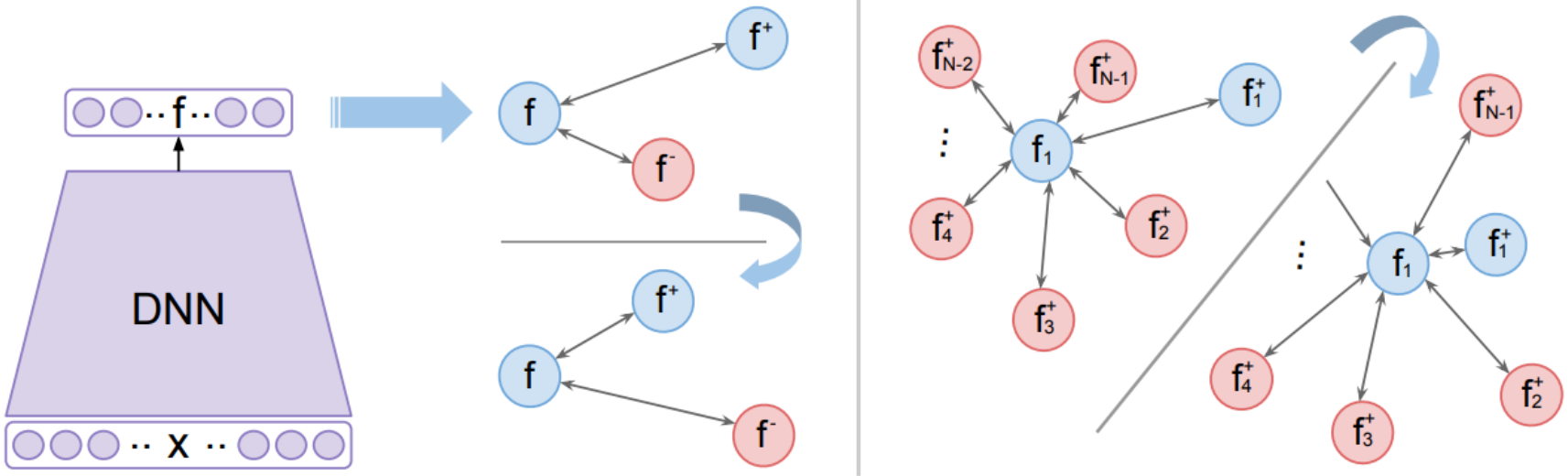
Negative examples

Consider an (N+1)-tuple of training examples $\{x, x^+, x_1, \cdots, x_{N-1}\}$

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

Triplet Loss

(N+1)-tuple Loss



Sohn, “Improved Deep Metric Learning with Multi-class N-pair Loss Objective”, NIPS 2016

N-pair loss

When the (N+1)-tuple loss is applied, for a mini-batch of M, Mx(N+1) examples need to be passed through f for each update.

Let $\{(x_1, x_1^+), \cdots, (x_N, x_N^+)\}$ be N pairs of examples from N different classes, i.e., $y_i \neq y_j, \forall i \neq j$:

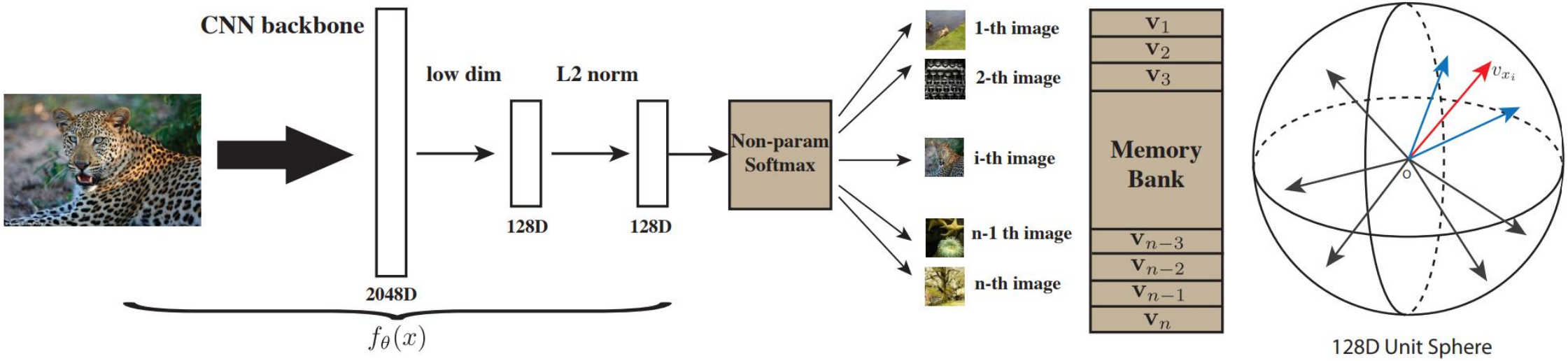
- Build N tuples $\{S_i\}_{i=1}^N$ from the N pairs $S_i = \{x_i, x_1^+, x_2^+, \cdots, x_N^+\}$
- The (N+1)-tuple loss is

$$\mathcal{L}_{N\text{-pair-mc}}(\{(x_i, x_i^+)\}_{i=1}^N; f) = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + \sum_{j \neq i} \exp(f_i^\top f_j^+ - f_i^\top f_i^+)\right)$$

Sohn, “Improved Deep Metric Learning with Multi-class N-pair Loss Objective”, NIPS 2016

Noise-Contrastive Estimation

Considers each input image as a unique class



Wu, Xiong, Yu, Lin, “Unsupervised Learning via Non-Parametric Instance Discrimination”, CVPR 2018

Noise-Contrastive Estimation

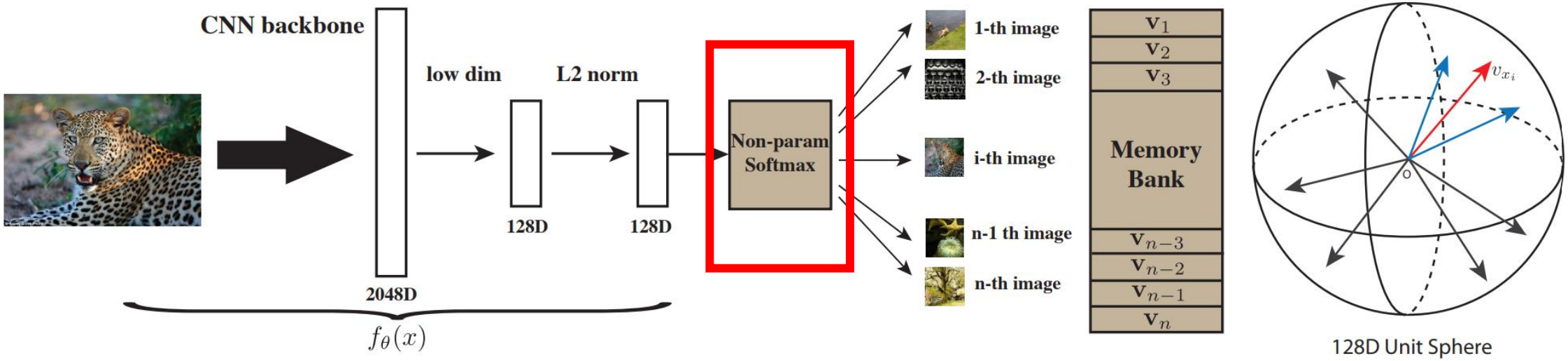
Considers each input image as a unique class.

Parametric softmax with parameters **w**

$$P(i|\mathbf{v}) = \frac{\exp\left(\mathbf{w}_i^T \mathbf{v}\right)}{\sum_{j=1}^{n} \exp\left(\mathbf{w}_j^T \mathbf{v}\right)}$$

Non-parametric softmax

$$P(i|\mathbf{v}) = \frac{\exp\left(\mathbf{v}_i^T \mathbf{v}/\tau\right)}{\sum_{j=1}^{n} \exp\left(\mathbf{v}_j^T \mathbf{v}/\tau\right)}$$



Wu, Xiong, Yu, Lin, “Unsupervised Learning via Non-Parametric Instance Discrimination”, CVPR 2018

Noise-Contrastive Estimation

Considers unit-nom embeddings → Normalization layer

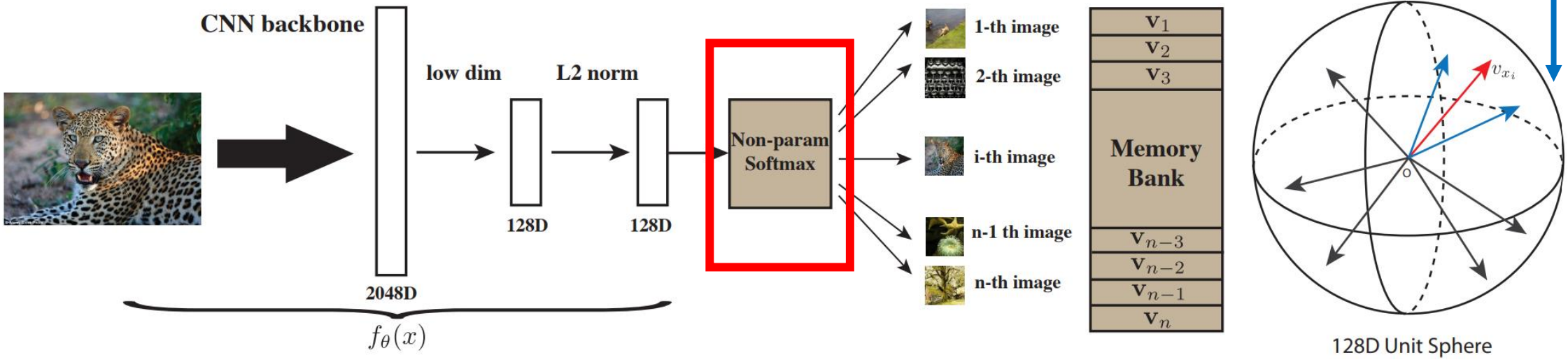
Considers each input image as a unique class.

Parametric softmax with parameters **w**

$$P(i|\mathbf{v}) = \frac{\exp\left(\mathbf{w}_i^T \mathbf{v}\right)}{\sum_{j=1}^{n} \exp\left(\mathbf{w}_j^T \mathbf{v}\right)}$$

Non-parametric softmax

$$P(i|\mathbf{v}) = \frac{\exp\left(\mathbf{v}_i^T \mathbf{v}/\tau\right)}{\sum_{j=1}^{n} \exp\left(\mathbf{v}_j^T \mathbf{v}/\tau\right)}$$



Wu, Xiong, Yu, Lin, "Unsupervised Learning via Non-Parametric Instance Discrimination", CVPR 2018

Noise-Contrastive Estimation

NCE loss introduces:

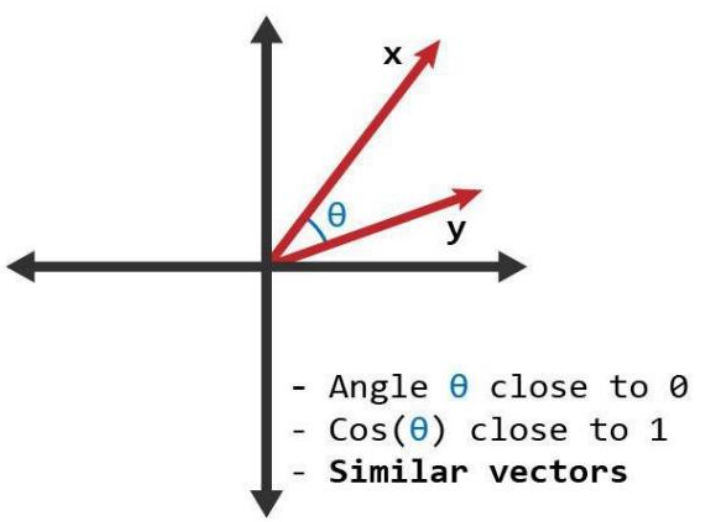
- explicit normalization
- a temperature parameter τ
- the idea of momentum encoder

$$\mathcal{L}_{\mathrm{NCE}} = -\sum_{n=1}^{N} \log \left(\frac{e^{\mathrm{CoSim}(\boldsymbol{z}_i, \boldsymbol{z}_i^{(t-1)})/\tau}}{\sum_{k=1}^{N} e^{\mathrm{CoSim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau}} \right) + \beta \| \boldsymbol{Z} - \boldsymbol{Z}^{(t-1)} \|_F^2$$

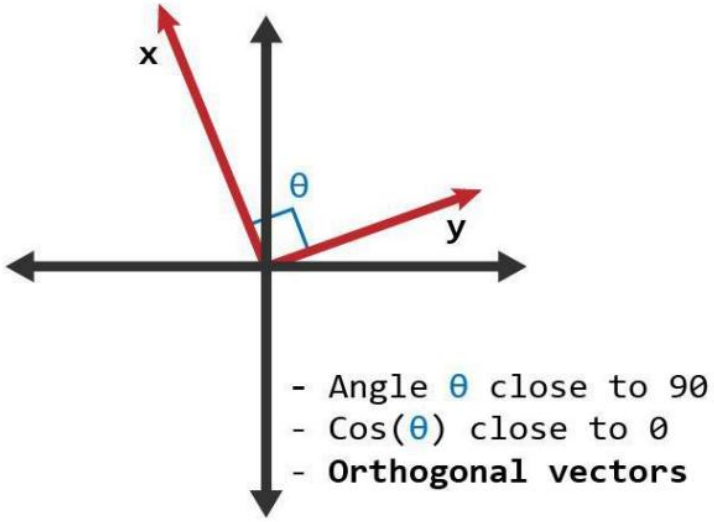
Wu, Xiong, Yu, Lin, “Unsupervised Learning via Non-Parametric Instance Discrimination”, CVPR 2018

Cosine similarity

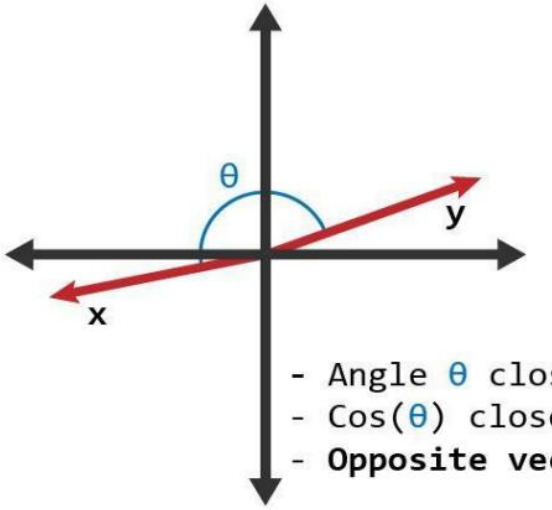
$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$



- Angle θ close to 0
- Cos(θ) close to 1
- **Similar vectors**



- Angle θ close to 90
- Cos(θ) close to 0
- **Orthogonal vectors**

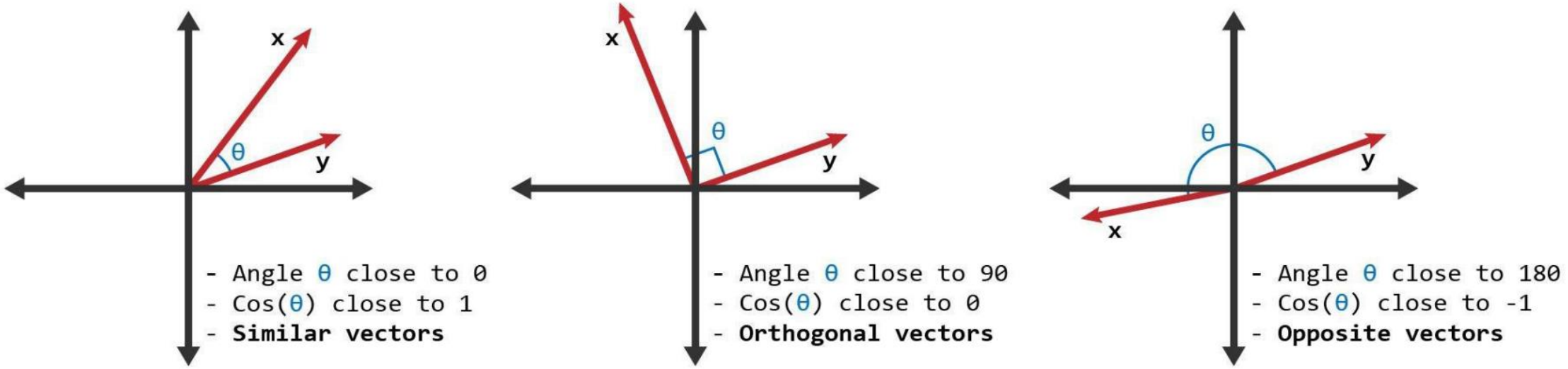


- Angle θ close to 180
- Cos(θ) close to -1
- **Opposite vectors**

Cosine similarity

Advantages:

- Even if two vectors are far apart based on Euclidean distance because of their magnitude, they can still have a small angle between them
- Captures orientation (angle) of points and not magnitude



InfoNCE

Considers each input image as a unique class.

$$\mathcal{L}_{\text{infoNCE}} = -\sum_{(i,j)\in\mathbb{P}} \log\left(\frac{e^{\text{CoSim}(\boldsymbol{z}_i,\boldsymbol{z}_j)/\tau}}{\sum_{k=1}^{N} e^{\text{CoSim}(\boldsymbol{z}_i,\boldsymbol{z}_k)/\tau}}\right)$$

Oord, Li, Vinyals, “Representation Learning with Contrastive Predictive Coding”, 1807.03748v2, 2018

SimCLR: A simple framework for CL Representations

Maximize the agreement of representations under data transformation, using contrastive loss in the latent/feature space

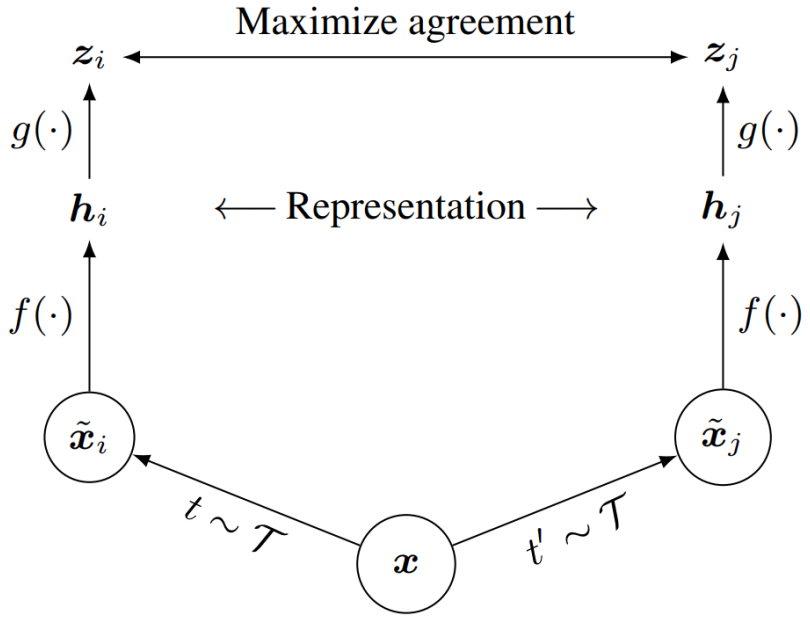
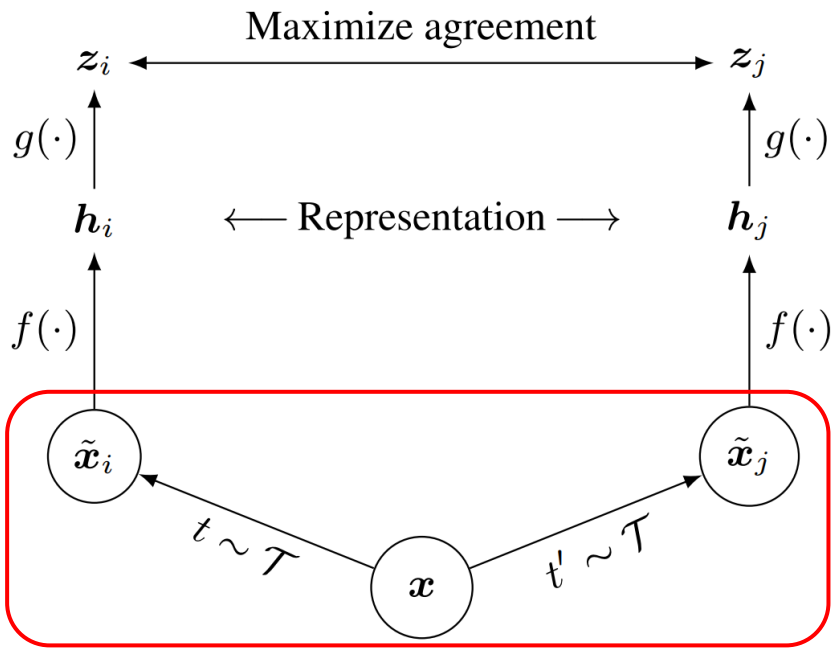


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

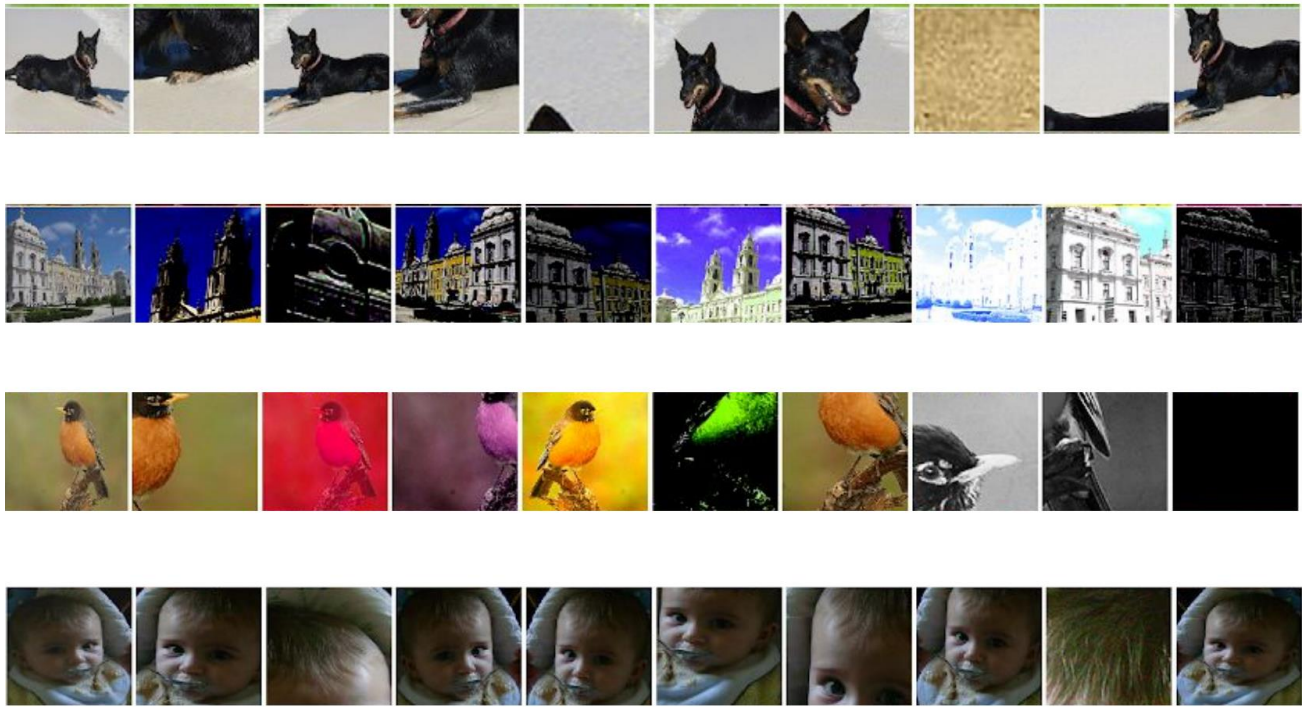
Chen et al., "A simple framework for contrastive learning of visual representations", ICML 2020

SimCLR: A simple framework for CL Representations



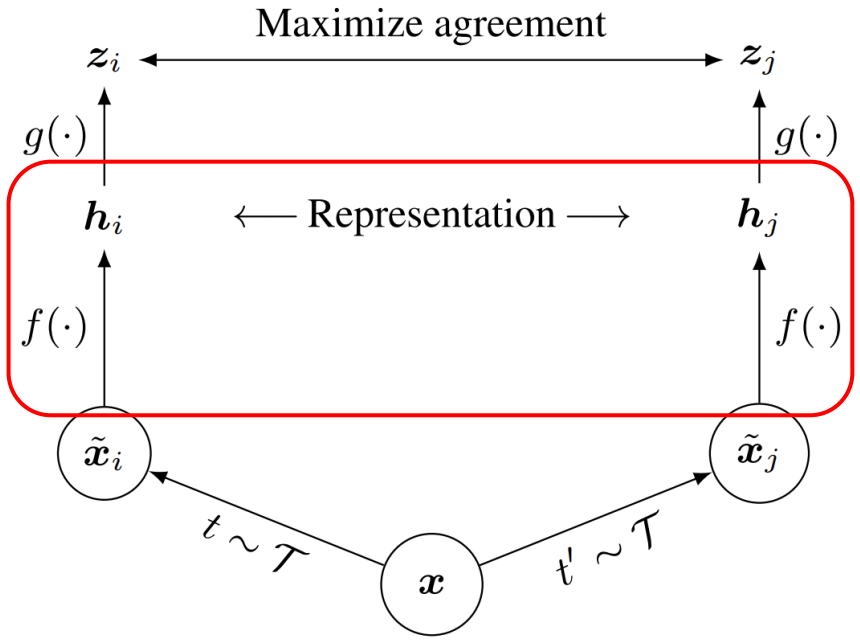
Use random crop and color distortion for augmentation

Example augmentations applied to the left-most images



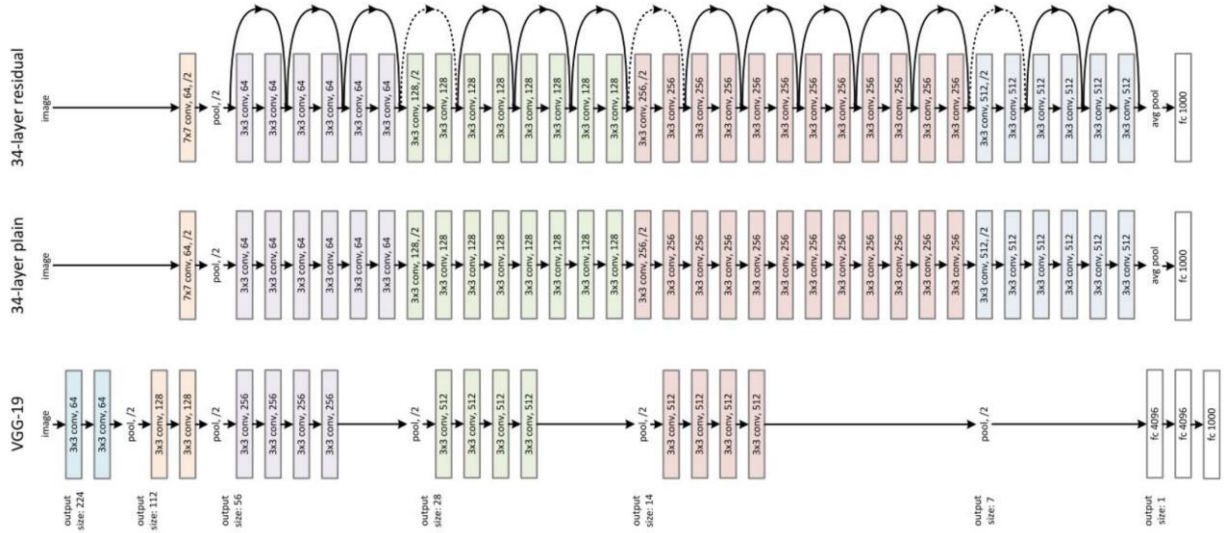
Chen et al., “A simple framework for contrastive learning of visual representations”, ICML 2020

SimCLR: A simple framework for CL Representations



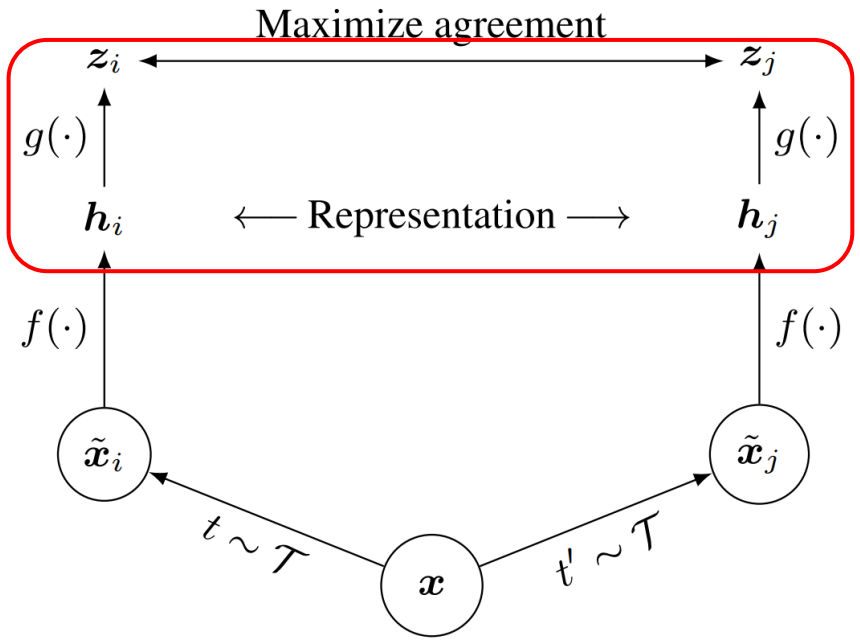
f(•) is the base network that computes internal representation

They use ResNet in the paper. However, it can be other networks.



Chen et al., “A simple framework for contrastive learning of visual representations”, ICML 2020

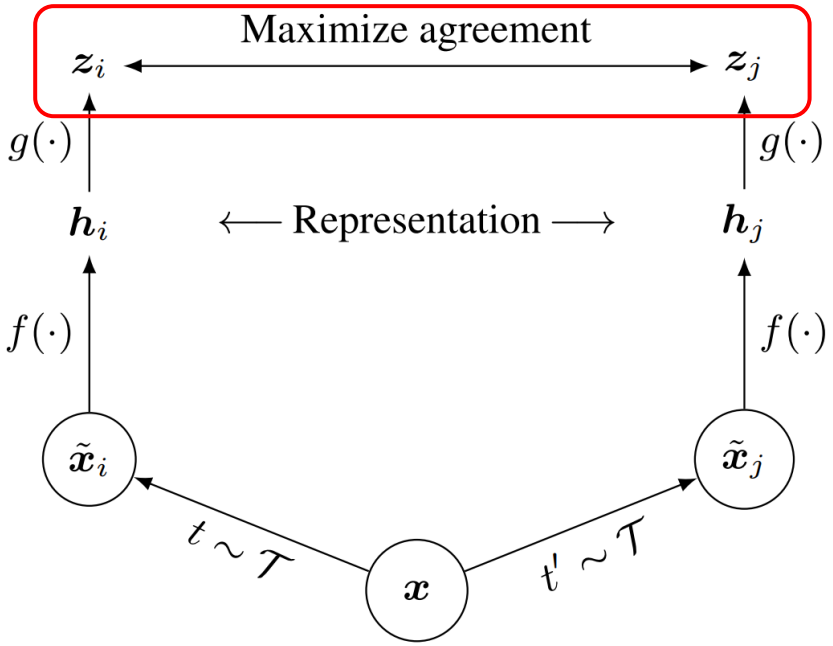
SimCLR: A simple framework for CL Representations



g(•) is a projection network (dense layers with non-linearities) that projects the representations to a latent space

Chen et al., “A simple framework for contrastive learning of visual representations”, ICML 2020

SimCLR: A simple framework for CL Representations



Maximize agreement using a contrastive task:

Let $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|$

Then the loss function for a positive pair of examples (i, j) is:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

This loss is computed across all positive pairs (both (i, j) and (j, i)) in a mini-batch

Chen et al., “A simple framework for contrastive learning of visual representations”, ICML 2020

SimCLR: A simple framework for CL Representations

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \dots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}, t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \dots, 2N\}$ and $j \in \{1, \dots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

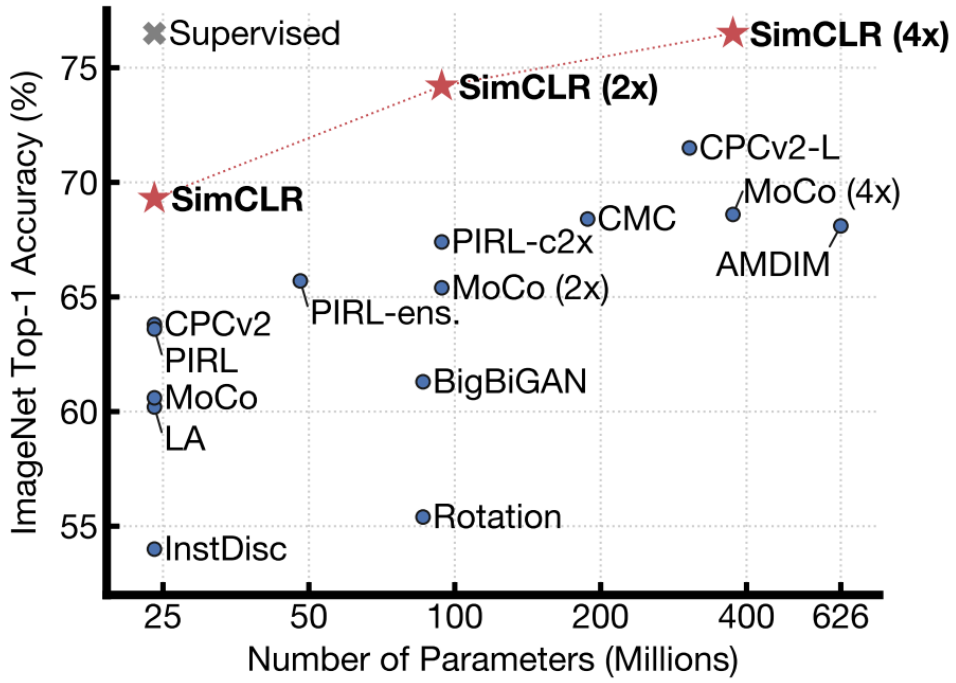


Figure 1. ImageNet Top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50. Our method, SimCLR, is shown in bold.

Chen et al., "A simple framework for contrastive learning of visual representations", ICML 2020

MoCo: Momentum Contrast for Unsupervised RL

MoCo decouples the training data (in the mini-batch) from the negative data:

- It considers an online-updated dictionary bank of negative data
- Uses two encoders:
  - One encoder for the training data (called queries) f_q with parameters θ_q which are updated by back-propagation
  - One encoder for the negative data (called keys forming the online dictionary) f_k with parameters θ_k which are updated by momentum update:

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

where $m \in [0, 1)$ is a momentum coefficient

He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

MoCo: Momentum Contrast for Unsupervised RL

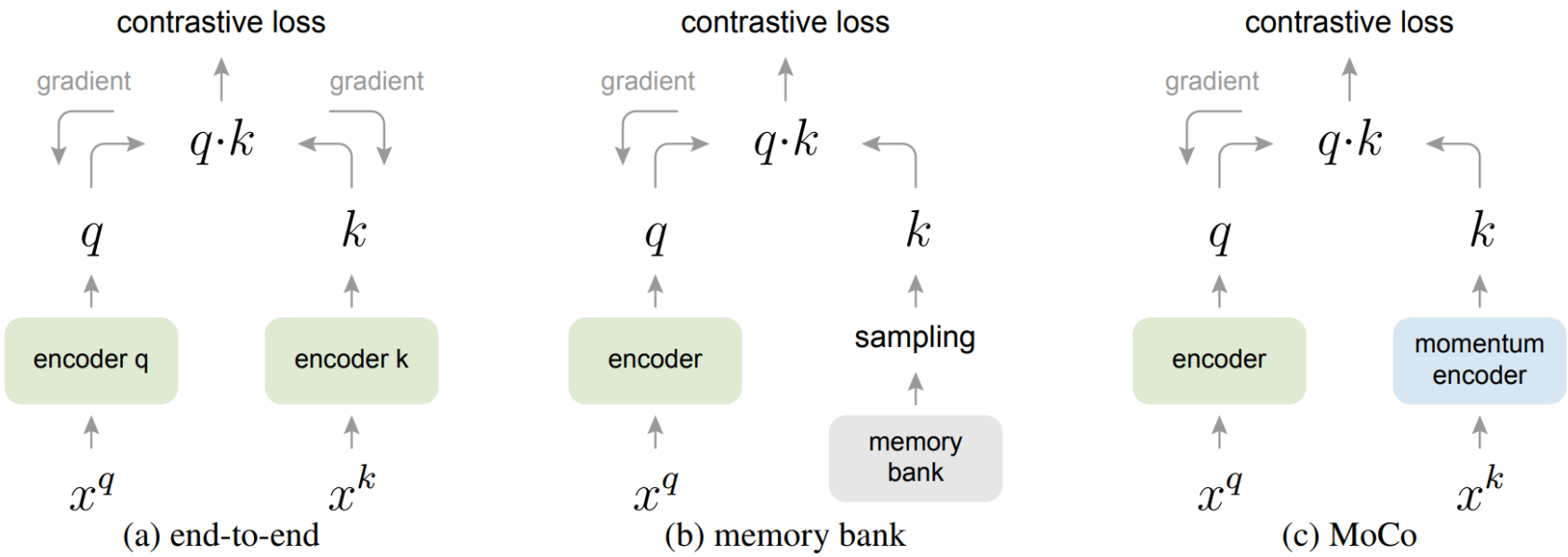


Figure 2. **Conceptual comparison of three contrastive loss mechanisms** (empirical comparisons are in Figure 3 and Table 3). Here we illustrate one pair of query and key. The three mechanisms differ in how the keys are maintained and how the key encoder is updated. **(a)**: The encoders for computing the query and key representations are updated ***end-to-end*** by back-propagation (the two encoders can be different). **(b)**: The key representations are sampled from a ***memory bank*** [61]. **(c)**: ***MoCo*** encodes the new keys on-the-fly by a momentum-updated encoder, and maintains a queue (not illustrated in this figure) of keys.

He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

MoCo: Momentum Contrast for Unsupervised RL

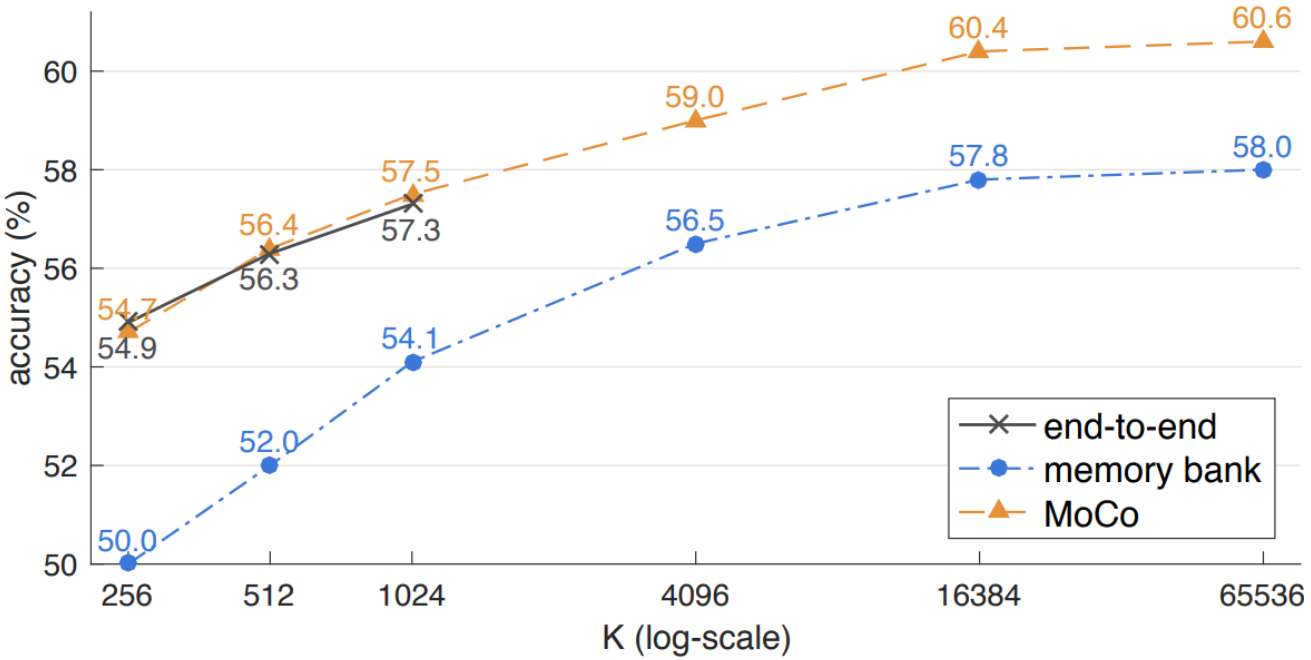


Figure 3. **Comparison of three contrastive loss mechanisms** under the ImageNet linear classification protocol. We adopt the same pretext task (Sec. 3.3) and only vary the contrastive loss mechanism (Figure 2). The number of negatives is K in memory bank and MoCo, and is $K-1$ in end-to-end (offset by one because the positive key is in the same mini-batch). The network is ResNet-50.

He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

BOYL: Bootstrap Your Own Latent

BOYL relies on two neural networks (online and target networks):

- The online network is trained to predict the target network representation of the same image under a different augmented view
- The target network is updated with a slow-moving average of the line network

It does not require the presence of negative pairs

Grill et al., “Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning”, NeurIPS 2020

BOYL: Bootstrap Your Own Latent

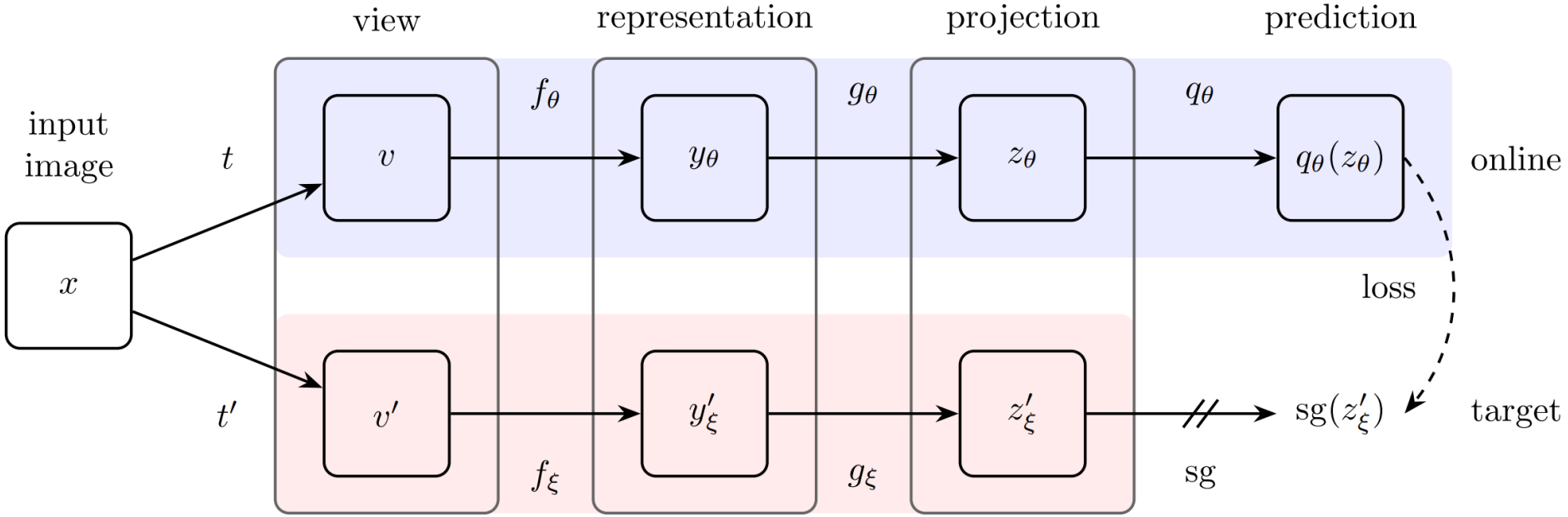


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the image representation.

$$\mathcal{L}_{\theta,\xi} \triangleq \left\| \overline{q_\theta}(z_\theta) - \overline{z}'_\xi \right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\left\| q_\theta(z_\theta) \right\|_2 \cdot \left\| z'_\xi \right\|_2}$$

$$\theta \leftarrow \mathrm{optimizer}\left(\theta, \nabla_\theta \mathcal{L}^{\mathtt{BYOL}}_{\theta,\xi}, \eta\right)$$

$$\xi \leftarrow \tau\xi + (1-\tau)\theta,$$

Grill et al., "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning", NeurIPS 2020

BOYL: Bootstrap Your Own Latent

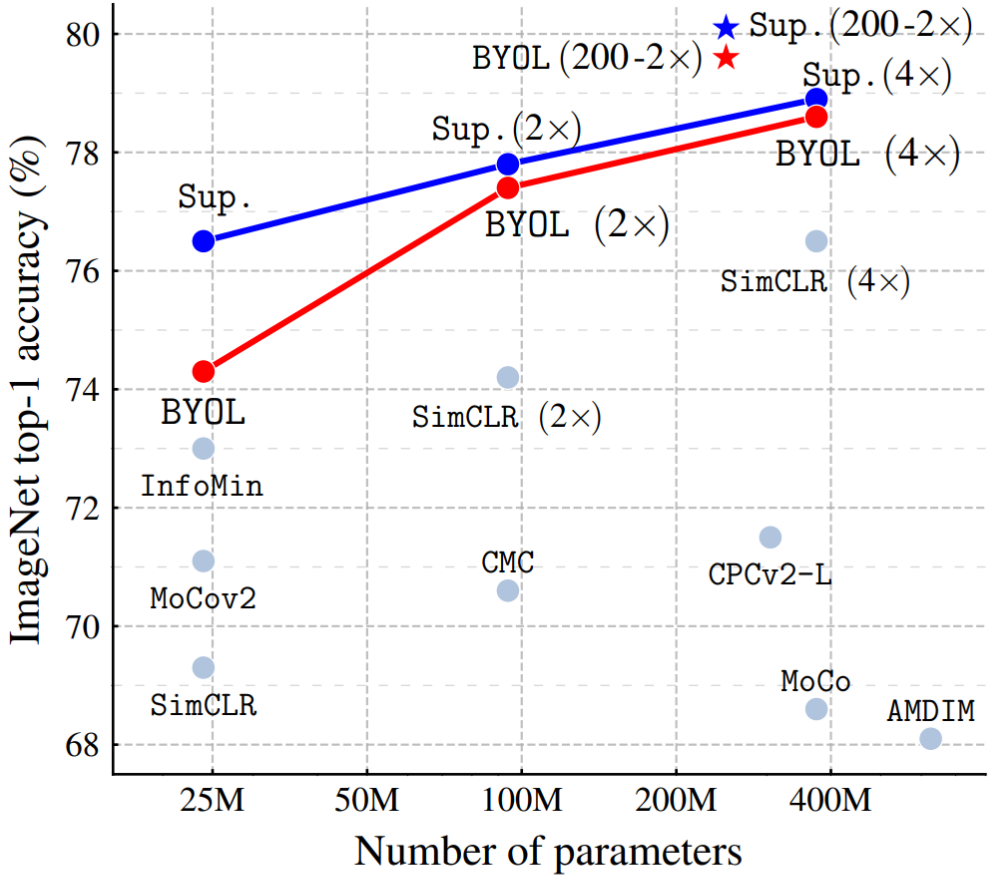


Figure 1: Performance of BYOL on ImageNet (linear evaluation) using ResNet-50 and our best architecture ResNet-200 (2×), compared to other unsupervised and supervised (Sup.) baselines [8].

Grill et al., "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning", NeurIPS 2020

SimSiam: Simple Siamese Representation Learning

Siamese networks can learn expressive representations without:

- Negative sample pairs
- Large batches
- Momentum encoders

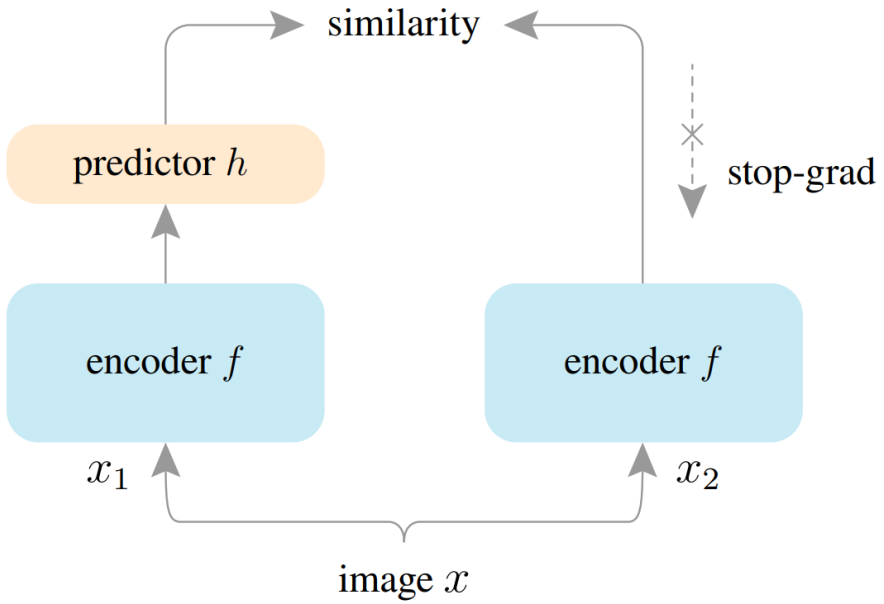


Figure 1. **SimSiam architecture**. Two augmented views of one image are processed by the same encoder network f (a backbone plus a projection MLP). Then a prediction MLP h is applied on one side, and a stop-gradient operation is applied on the other side. The model maximizes the similarity between both sides. It uses neither negative pairs nor a momentum encoder.

Chen, He, "Exploring Simple Siamese Representation Learning", CVPR 2021

SimSiam: Simple Siamese Representation Learning

Stop-grad

Define the negative cosine similarity: $\mathcal{D}(p_1, z_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{z_2}{\|z_2\|_2}$

The loss function is: $\mathcal{L} = \frac{1}{2}\mathcal{D}(p_1, z_2) + \frac{1}{2}\mathcal{D}(p_2, z_1)$

If we treat one representation (say z_2) as constant: $\mathcal{D}(p_1, \texttt{stopgrad}(z_2))$

The loss becomes: $\mathcal{L} = \frac{1}{2}\mathcal{D}(p_1, \texttt{stopgrad}(z_2)) + \frac{1}{2}\mathcal{D}(p_2, \texttt{stopgrad}(z_1))$

The encoder on x_2 receives no gradient from z_2 in the first term, but it receives gradients from p_2 in the second term (and vice versa for x_1)

SimSiam: Simple Siamese Representation Learning

Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# f: backbone + projection mlp
# h: prediction mlp

for x in loader: # load a minibatch x with n samples
   x1, x2 = aug(x), aug(x) # random augmentation
   z1, z2 = f(x1), f(x2) # projections, n-by-d
   p1, p2 = h(z1), h(z2) # predictions, n-by-d

   L = D(p1, z2)/2 + D(p2, z1)/2 # loss

   L.backward() # back-propagate
   update(f, h) # SGD update

def D(p, z): # negative cosine similarity
   z = z.detach() # stop gradient

   p = normalize(p, dim=1) # l2-normalize
   z = normalize(z, dim=1) # l2-normalize
   return -(p*z).sum(dim=1).mean()
```

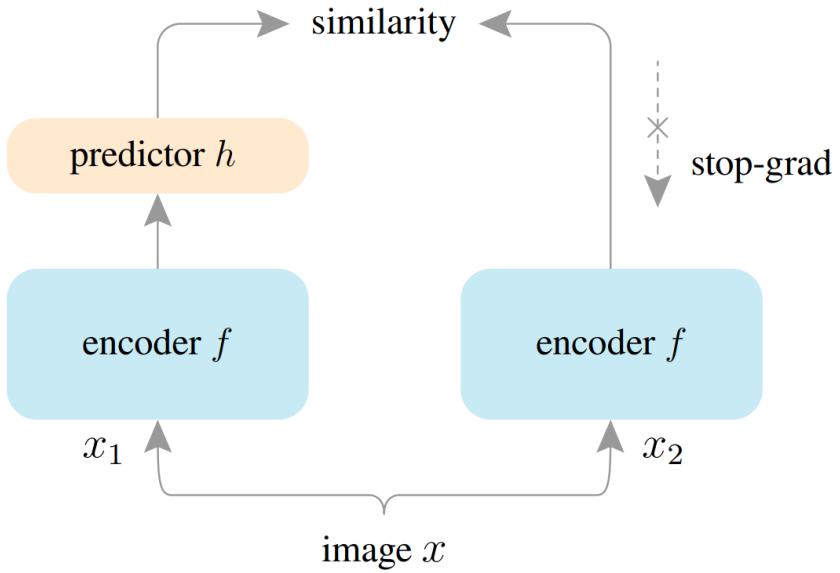


Figure 1. **SimSiam architecture**. Two augmented views of one image are processed by the same encoder network f (a backbone plus a projection MLP). Then a prediction MLP h is applied on one side, and a stop-gradient operation is applied on the other side. The model maximizes the similarity between both sides. It uses neither negative pairs nor a momentum encoder.

Chen, He, "Exploring Simple Siamese Representation Learning", CVPR 2021

Properties of Contrastive Losses

Data augmentations are crucial

ImageNet images are of different resolutions. To remove this effect:

- First random crop an image and resize to a standard resolution.
- Then apply a single or a pair of augmentations on one branch, while keeping the other as identity mapping.
- This is suboptimal compared to applying augmentations to both branches, but sufficient for ablation.

Chen, Luo, Li, “Intriguing Properties of Contrastive Losses”, NeurIPS 2021

Properties of Contrastive Losses

Data augmentations are crucial

Composition of crop and color stands out!

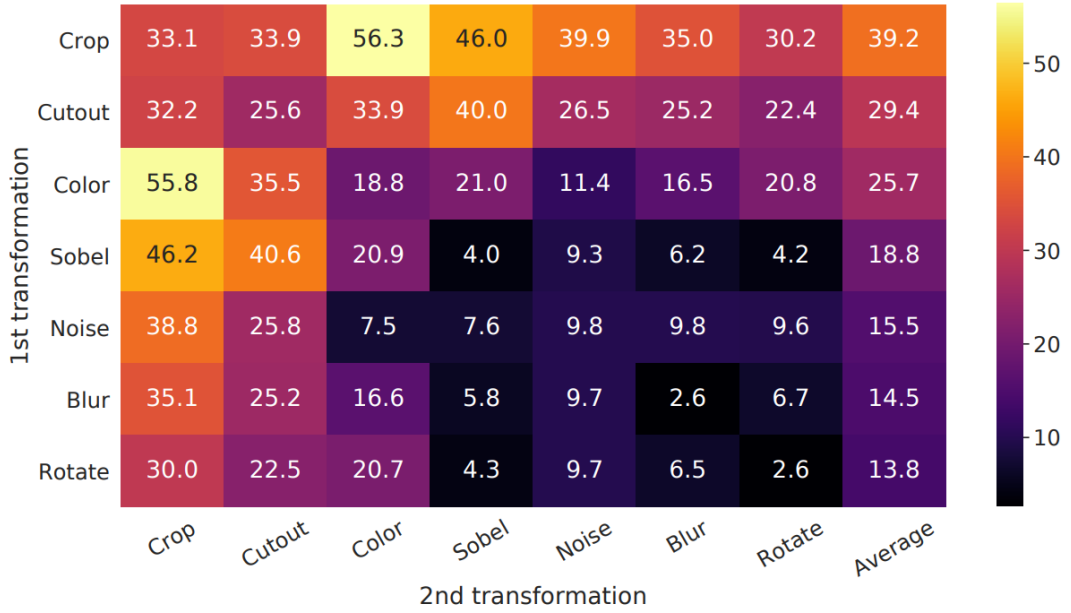
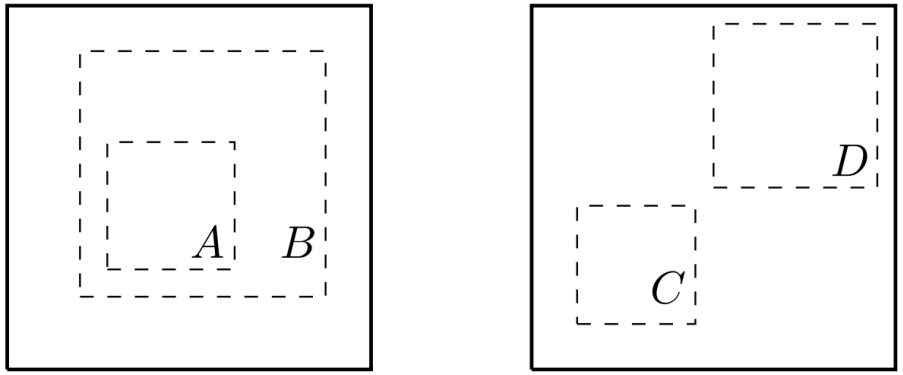


Figure 5. Linear evaluation (ImageNet top-1 accuracy) under individual or composition of data augmentations, applied only to one branch. For all columns but the last, diagonal entries correspond to single transformation, and off-diagonals correspond to composition of two transformations (applied sequentially). The last column reflects the average over the row.

Simply via Random Crop (with resize to standard size), we can mimic (1) global to local view prediction, and (2) neighboring view prediction. This simple transformation defines a family of predictive tasks.



(a) Global and local views. (b) Adjacent views.

Figure 3. Solid rectangles are images, dashed rectangles are random crops. By randomly cropping images, we sample contrastive prediction tasks that include global to local view ($B \rightarrow A$) or adjacent view ($D \rightarrow C$) prediction.

From SimCLR paper

Properties of Contrastive Losses

Projection head is important

A nonlinear projection head improves the representation quality of the layer before it

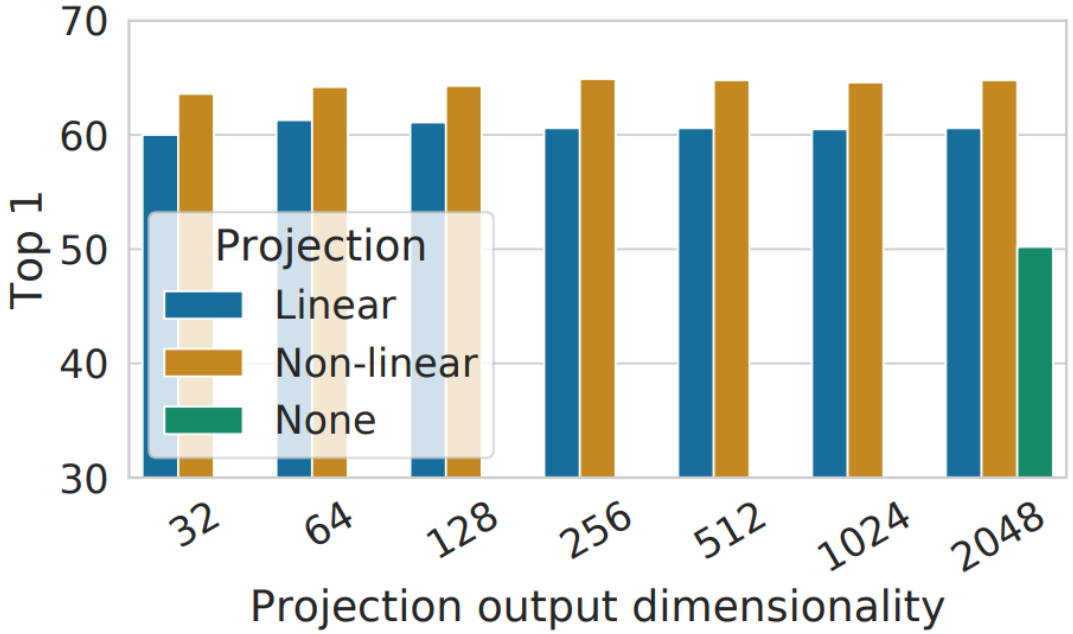


Figure 8. Linear evaluation of representations with different projection heads $g(\cdot)$ and various dimensions of $\boldsymbol{z} = g(\boldsymbol{h})$. The representation $\boldsymbol{h}$ (before projection) is 2048-dimensional here.

From SimCLR paper

Properties of Contrastive Losses

Model size

Unsupervised contrastive learning benefits more from bigger models (from SimCLR paper)

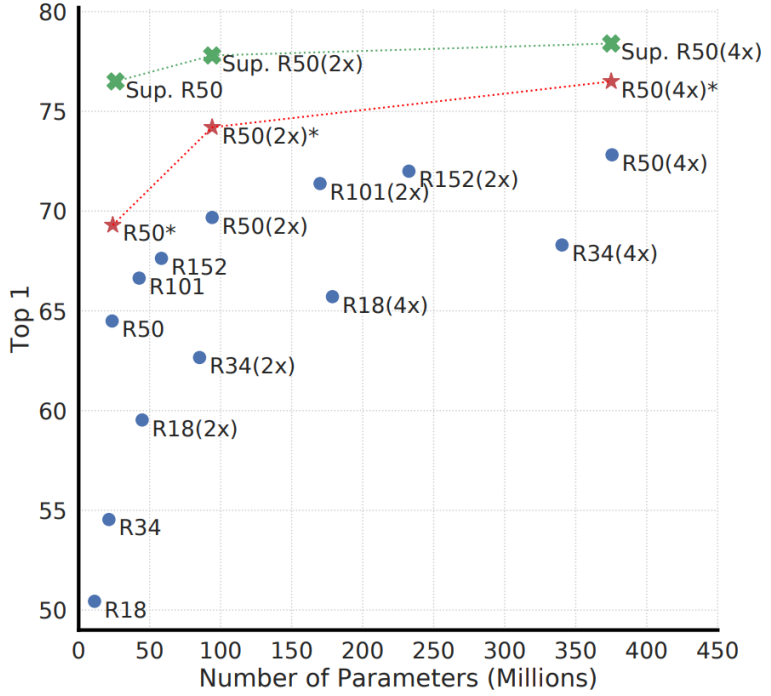
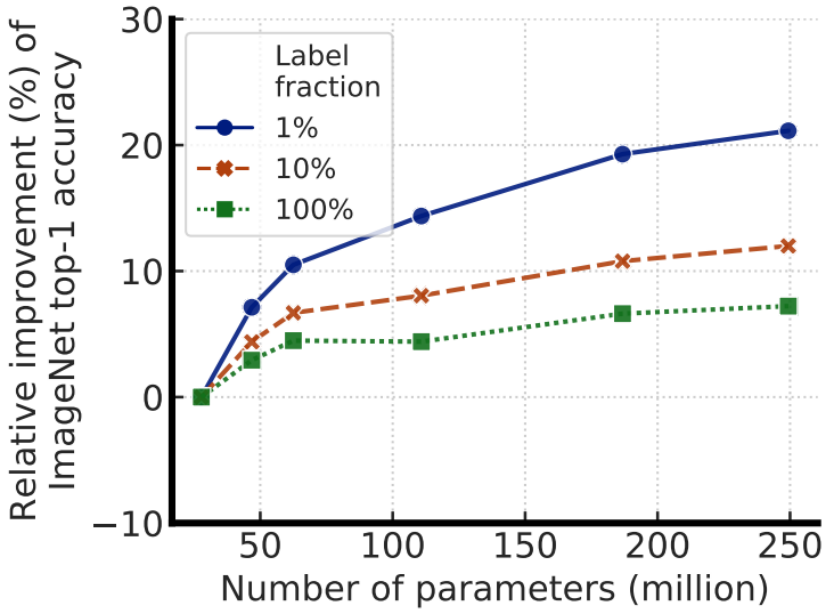
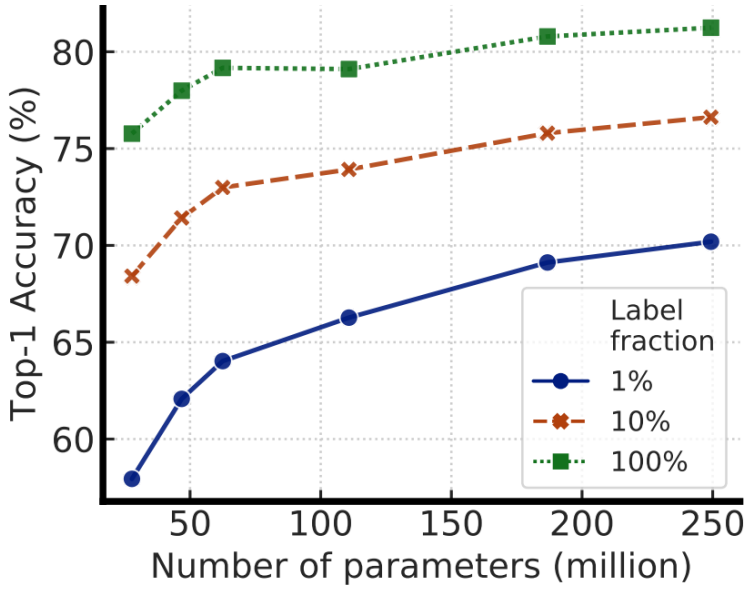


Figure 7. Linear evaluation of models with varied depth and width. Models in blue dots are ours trained for 100 epochs, models in red stars are ours trained for 1000 epochs, and models in green crosses are supervised ResNets trained for 90 epochs[7] (He et al., 2016).

Increasing the size of model size by 10X, it reduces required labels to achieve certain accuracy by 10X. (from SimCLRv2 paper)





Properties of Contrastive Losses

Effect of hyper-parameter values

Compare variants of contrastive loss in SimCLR:

- L2 normalization with temperature scaling makes a better loss
- Contrastive accuracy is not correlated with linear evaluation when L2 norm and/or temperature are changed

ℓ_2 norm?	τ	Entropy	Contrastive acc.	Top 1
Yes	0.05	1.0	90.5	59.7
	0.1	4.5	87.8	64.4
	0.5	8.2	68.2	60.7
	1	8.3	59.1	58.0
No	10	0.5	91.7	57.2
	100	0.5	92.1	57.0

Table 5. Linear evaluation for models trained with different choices of ℓ_2 norm and temperature τ for NT-Xent loss. The contrastive distribution is over 4096 examples.

Properties of Contrastive Losses

CL benefits from longer training

Compare epochs & batch size in SimCLR (hyper-parameter tuned with batch size of 4096)

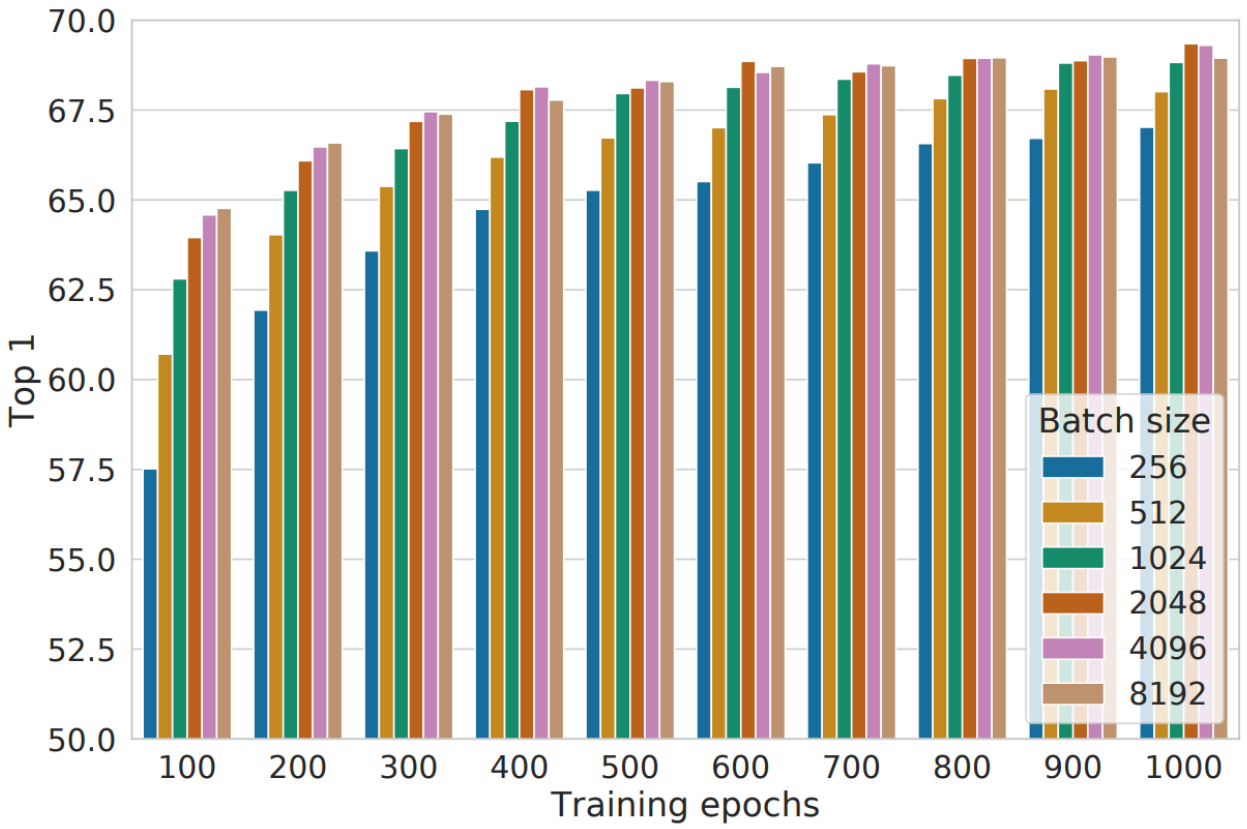


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

Properties of Contrastive Losses

Small batch sizes work well too with good hyper-parameter tuning

Original SimCLR was developed with large batch size, so the hyper-params were not optimized for smaller ones in the above batch size study.

With proper tuning on learning rate, temperature, and deeper projection head, the difference in batch sizes becomes smaller.

Table A.3: Linear evaluation accuracy (top-1) of ResNet-50 trained with different losses on ImageNet (with 3-layer projection head).

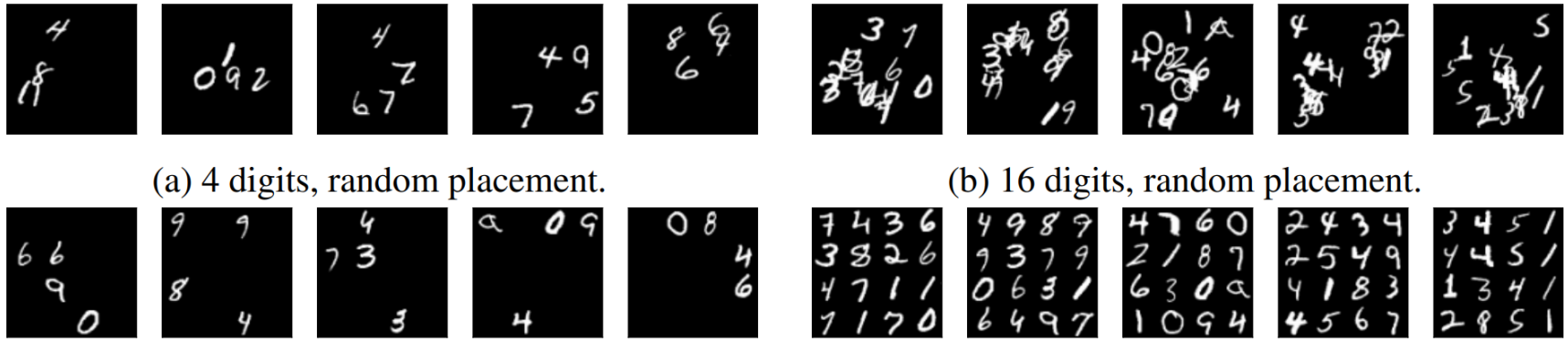
Loss	Epoch / Batch size	100	200	400	800
NT-Xent	512	66.6	68.4	70.0	71.0
	1024	66.8	68.9	70.1	70.9
	2048	66.8	69.1	70.4	71.3

Chen, Luo, Li, “Intriguing Properties of Contrastive Losses”, NeurIPS 2021

Properties of Contrastive Losses

Single object vs Multiple objects

- It has been conjectured that many existing contrastive learning methods take advantage of **dataset bias** (e.g. in ImageNet): there's a single/dominant object in the center, and random crops typically share object identity.
- Construct a dataset of multiple MNIST digits



(a) 4 digits, random placement.

(b) 16 digits, random placement.

(c) 4 digits, in-grid placement.

(d) 16 digits, in-grid placement.

Figure 2: MultiDigit dataset. More digits lead to more overlapping in random placement.

Chen, Luo, Li, "Intriguing Properties of Contrastive Losses", NeurIPS 2021

Properties of Contrastive Losses

Single object vs Multiple objects

- SimCLR is able to learn just fine even with multiple MNIST digits

Table 3: Top-1 linear evaluation accuracy (%) for pretrained ResNet-18 on the MultiDigits dataset. We vary the number of digits placed on the canvas during training from 1 to 16. During evaluation only 1 digit is present. As a baseline, a network with random weights gives 18% top-1 accuracy.

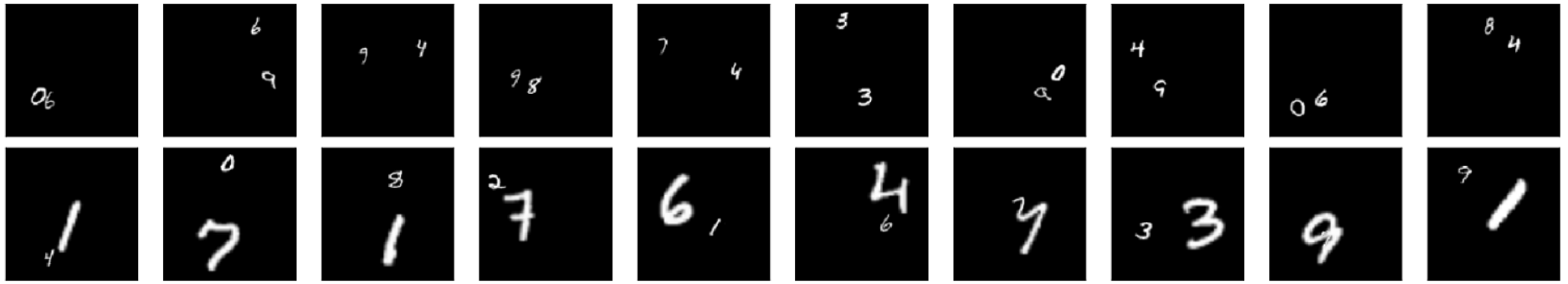
	Placing of digits	Number of digits (size 28×28)					
		1	2	4	8	12	16
Supervised	Random	99.5	99.5	99.3	99.4	98.9	98.3
	In-grid	99.5	99.6	99.5	99.3	98.6	92.4
SimCLR	Random	98.9	98.9	99.0	98.9	98.2	96.4
	In-grid	98.3	98.6	99.1	99.2	99.1	98.3

Chen, Luo, Li, “Intriguing Properties of Contrastive Losses”, NeurIPS 2021

Properties of Contrastive Losses

Larger objects suppress the learning of smaller objects

- Place two MNIST digits randomly on a canvas, and increase the size of one digit while keeping the other fixed.



(b) Two MNIST digits randomly placed on a shared canvas (of size 112×112). The two digits can have the same size (upper row) or different sizes (lower row), and digits of different sizes can be considered as competing features. We fix the size of one digit and vary the other.

Chen, Luo, Li, “Intriguing Properties of Contrastive Losses”, NeurIPS 2021

Properties of Contrastive Losses

Larger objects suppress the learning of smaller objects

- Place two MNIST digits randomly on a canvas, and increase the size of one digit while keeping the other fixed.

Table 4: Top-1 linear evaluation accuracy (%) for pretrained ResNet-18 on the MultiDigits dataset. We fix the size of 1st digit while increasing the size of the 2nd digit. For SimCLR, results are presented for two temperatures. Accuracies suffered from a significant drop when increasing 2nd digit size are red colored.

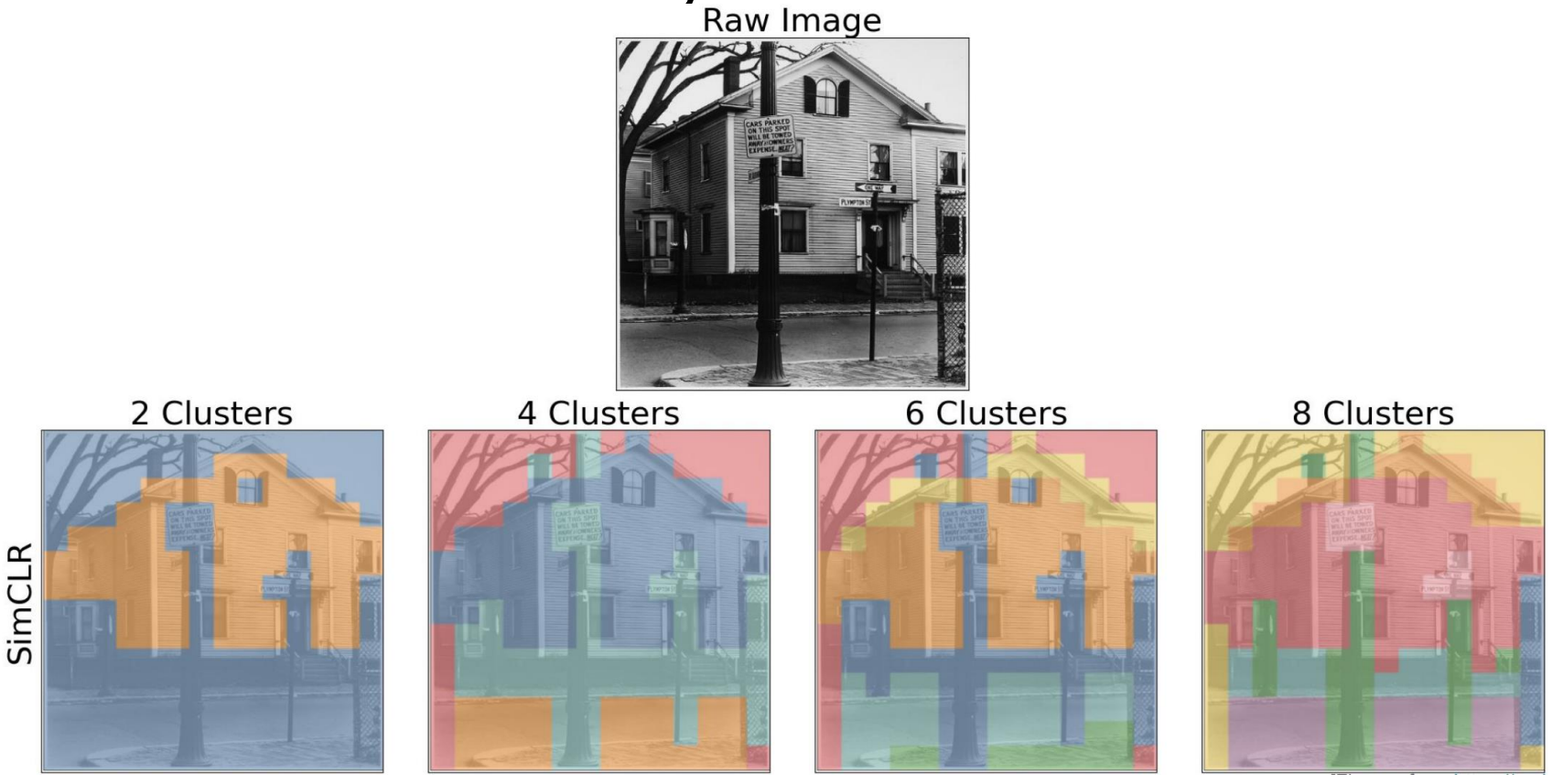
		2nd digit size (1st digit is kept the same size of 20×20)						
		20×20	30×30	40×40	50×50	60×60	70×70	80×80
Supervised	1st digit	99.1	99.2	99.2	99.2	99.1	99.1	99.0
	2nd digit	99.1	99.5	99.5	99.6	99.5	99.5	99.6
SimCLR ($\tau = 0.05$)	1st digit	97.8	97.6	96.2	96.5	88.5	74.5	39.9
	2nd digit	97.8	97.9	97.8	98.3	98.2	97.7	98.2
SimCLR ($\tau = 0.2$)	1st digit	98.7	98.8	98.3	87.5	24.9	19.8	20.3
	2nd digit	98.7	99.2	99.2	99.0	99.1	98.9	99.4
Random net (untrained)	1st digit	16.5	16.7	16.6	16.6	16.6	16.9	16.5
	2nd digit	16.5	19.1	21.9	24.1	26.5	28.1	29.0

Chen, Luo, Li, “Intriguing Properties of Contrastive Losses”, NeurIPS 2021

Properties of Contrastive Losses

Global features vs local features

- Is instance-based contrastive learning able to learn local features?
- Take a middle layer of SimCLR learned ResNet and perform clustering



Chen, Luo, Li, “Intriguing Properties of Contrastive Losses”, NeurIPS 2021