



# Self-Supervised Learning

Computer Vision

Sungjoon Choi, Korea University



# Content

- DrLIM
- Contrastive Predictive Coding
- SimCLR
- MoCo
- BYOL
- SimCLRv2
- Barlow Twins



# DrLIM

"Dimensionality Reduction by Learning an Invariant Mapping," 2006



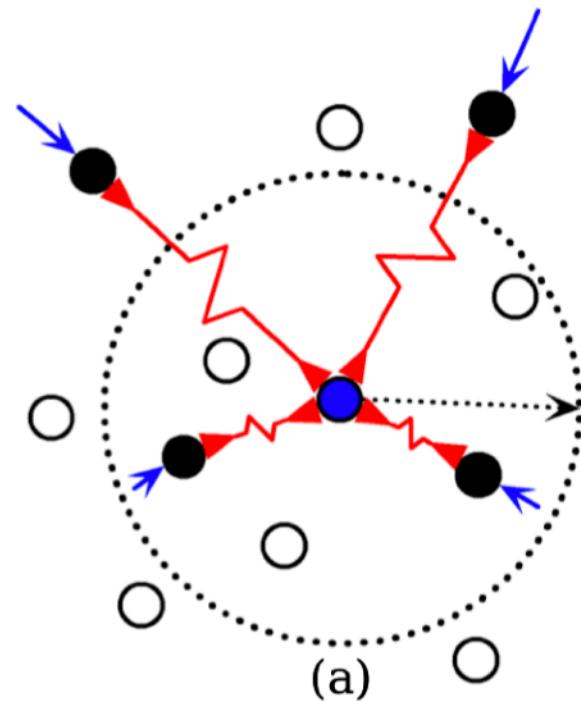
# Contrastive Learning

- This paper presents **DrLIM**, short for Dimensionality Reduction by Learning and Invariant Mapping.
- But, this paper is more famous for its **Contrastive Loss** function.
- Suppose that  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are a pair of samples and  $y$  is a binary label assigned to this pair where  $y = 0$  if they are similar and  $y = 1$  if they are dissimilar.

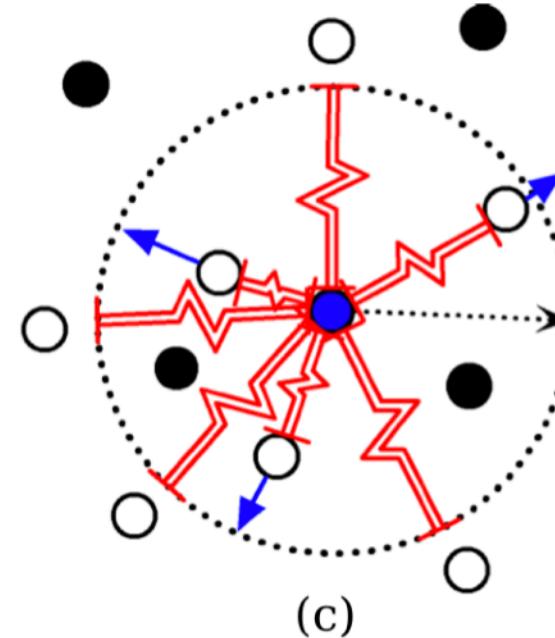
$$\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, y) = (1 - y) \frac{1}{2} (D_\theta)^2 + y \frac{1}{2} [\max(0, m - D_\theta)]^2$$

- where  $D_\theta(\mathbf{x}_1, \mathbf{x}_2) = \|f_\theta(\mathbf{x}_1) - f_\theta(\mathbf{x}_2)\|_2$

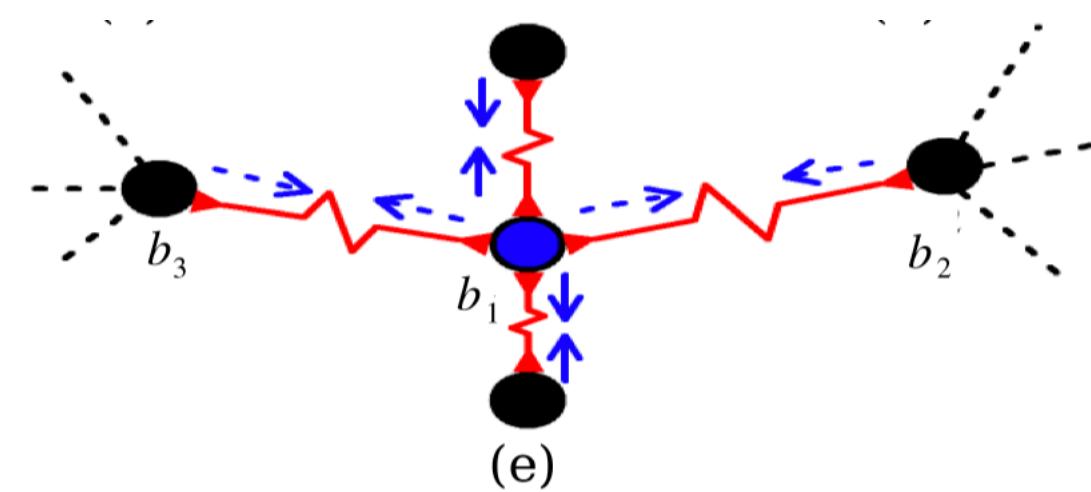
# Contrastive Learning



Attract similar pairs



Repulse dissimilar pairs  
(upto radius  $m$ )

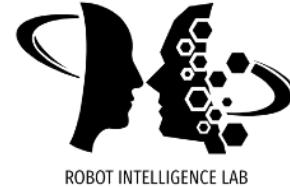


Equilibrium

# Algorithm

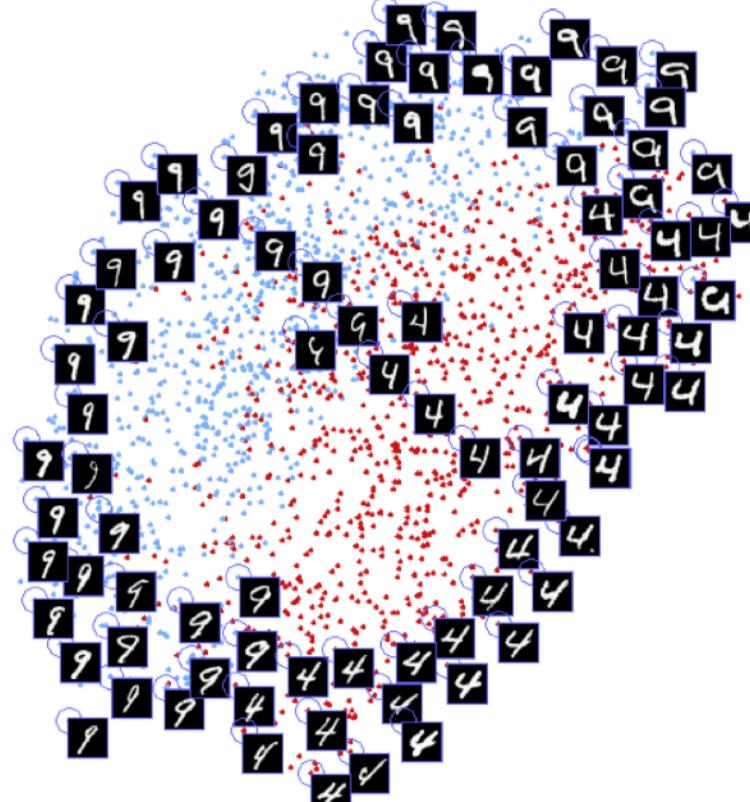
- **Step 1:** For each input sample  $\mathbf{x}_i$ , do the following:
  - Using prior knowledge, find the set of samples  $S_{\mathbf{x}_i} = \{\mathbf{x}_j\}_{j=1}^p$ , such that  $\mathbf{x}_j$  is deemed **similar** to  $\mathbf{x}_i$ .
  - Pair the sample  $\mathbf{x}_i$ , with **all the other** training samples and label the pairs so that:  $y_{ij} = 0$  if  $\mathbf{x}_j \in S_{\mathbf{x}_i}$ , and  $y_{ij} = 1$  otherwise.
- Combine all the pairs to form the labeled training set.
- **Step 2:** Repeat until convergence:
  - For each pair  $(\mathbf{x}_i, \mathbf{x}_j)$  in the training set, do
    - If  $y_{ij} = 0$ , the update  $\theta$  to **decrease**  $D_\theta = \|G_\theta(\mathbf{x}_i) - G_\theta(\mathbf{x}_j)\|_2$
    - If  $y_{ij} = 1$ , the update  $\theta$  to **increase**  $D_\theta = \|G_\theta(\mathbf{x}_i) - G_\theta(\mathbf{x}_j)\|_2$

# Caveat



How do we determine which input pairs should be **similar** and **dissimilar**?

# MNIST



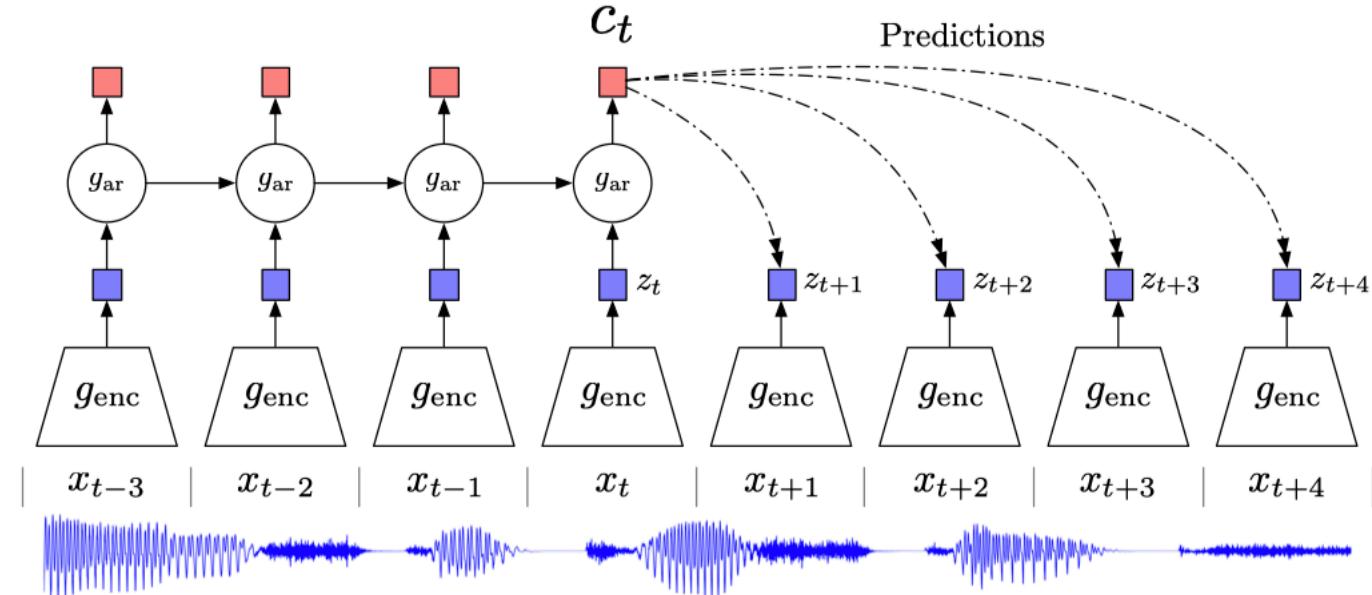
- 3,000 digit 4 images and 3,000 digit 9 images
- Each sample is paired with its 5 nearest neighbors, producing 30,000 similar pairs and  $2 \times 3,000 \times 3,000 = 18M$  negative pairs.



# Contrastive Predictive Coding

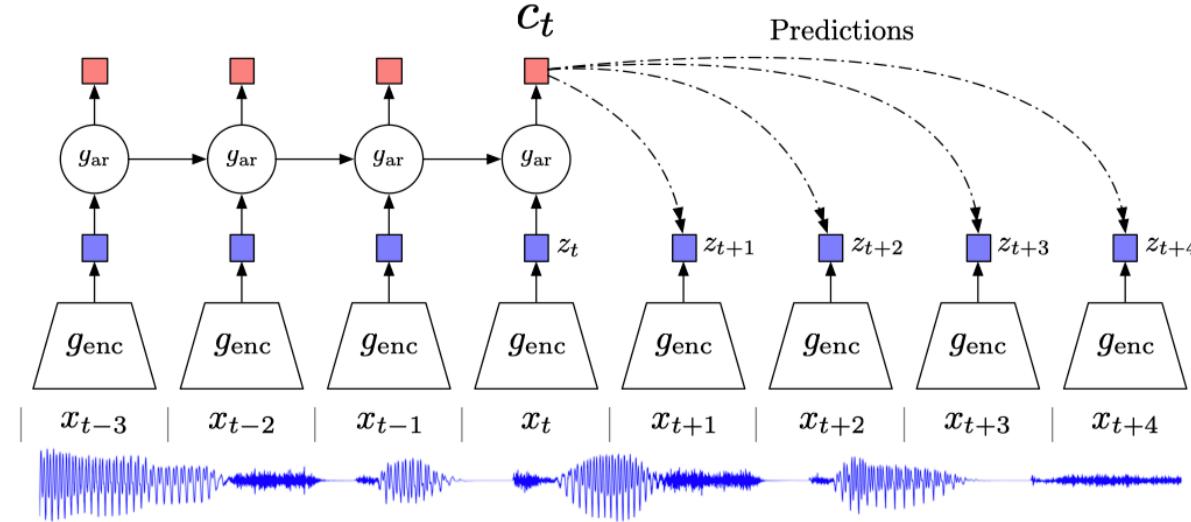
"Representation Learning with Contrastive Predictive Coding," 2019

# Contrastive Predictive Coding



The key insight of this model is to learn powerful representations by **predicting the future in latent space**.

# Contrastive Predictive Coding



- The objective of CPC is to produce a context latent representation  $c_t = g(z_{\leq t})$  that maximizes the mutual information between the future observation  $x_{t+k}$  and  $c_t$ .
- However, as computing the mutual information is intractable, following **density ratio** is maximized instead:

$$f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k} | c_t)}{p(x_{t+k})} = \exp(z_{t+k}^T W_k c_t)$$

# InfoNCE Loss

- How can properly maximize  $f_k(\mathbf{x}_i, \mathbf{c}_i; \theta)$  for a positive pair  $(\mathbf{x}_i, \mathbf{c}_i)$ ?
- How about this?

$$\hat{\theta} = \arg \max_{\theta} \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i, \mathbf{c}_i; \theta)$$

where  $D = \{\mathbf{x}_i, \mathbf{c}_i\}_{i=1}^N$  is the set of positive pairs.

# InfoNCE Loss

- How can properly maximize  $f_k(\mathbf{x}_i, \mathbf{c}_i; \theta)$  for a positive pair  $(\mathbf{x}_i, \mathbf{c}_i)$ ?
- How about this?

$$\hat{\theta} = \arg \max_{\theta} \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i, \mathbf{c}_i; \theta)$$

where  $D = \{\mathbf{x}_i, \mathbf{c}_i\}_{i=1}^N$  is the set of positive pairs.

- **InfoNCE loss** is proposed for this purpose:

$$\mathcal{L}_N(\theta) = \mathbb{E}_X \left[ \log \frac{f(\mathbf{x}_1, \mathbf{c}_t; \theta)}{\sum_{x_j \in X} f(x_j, \mathbf{c}_t; \theta)} \right]$$

where  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  is a set of  $N$  random samples containing one positive sample  $x_1$  and  $N - 1$  negative samples.

# InfoNCE Loss



This InfoNCE loss  $\mathcal{L}_N(\theta) = \mathbb{E}_X \left[ \log \frac{f(x_1, c_t; \theta)}{\sum_{x_j \in X} f(x_j, c_t; \theta)} \right]$  is a lower bound on **mutual information**:

$$I(x_1, c_t) \geq \log(N) - \mathcal{L}_N$$

hence minimizing the **infoNCE loss** will maximize the lower bound of **mutual information**.

# Speaker Identification

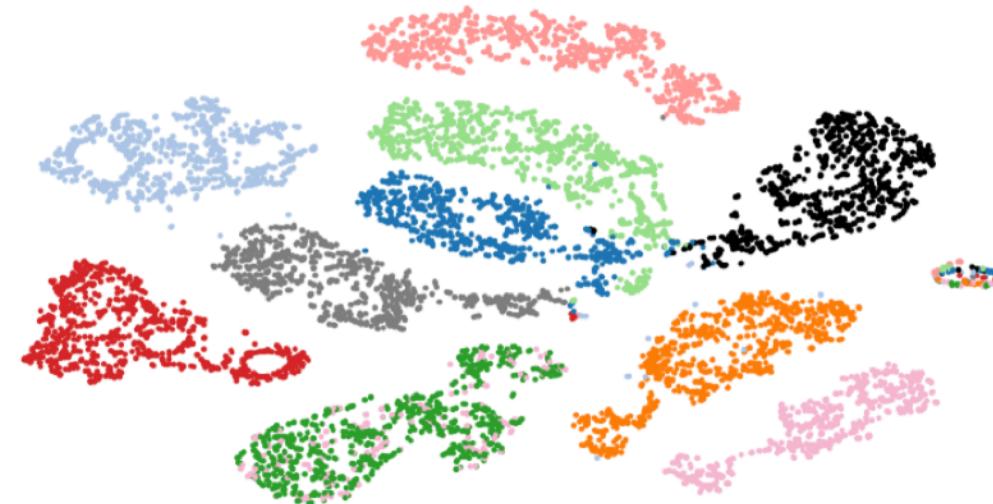
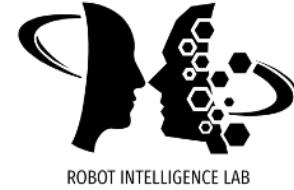


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.



# SimCLR

"A Simple Framework for Contrastive Learning of Visual Representations," 2020

# SimCLR



This is the famous 'SimCLR' paper.

# SimCLR

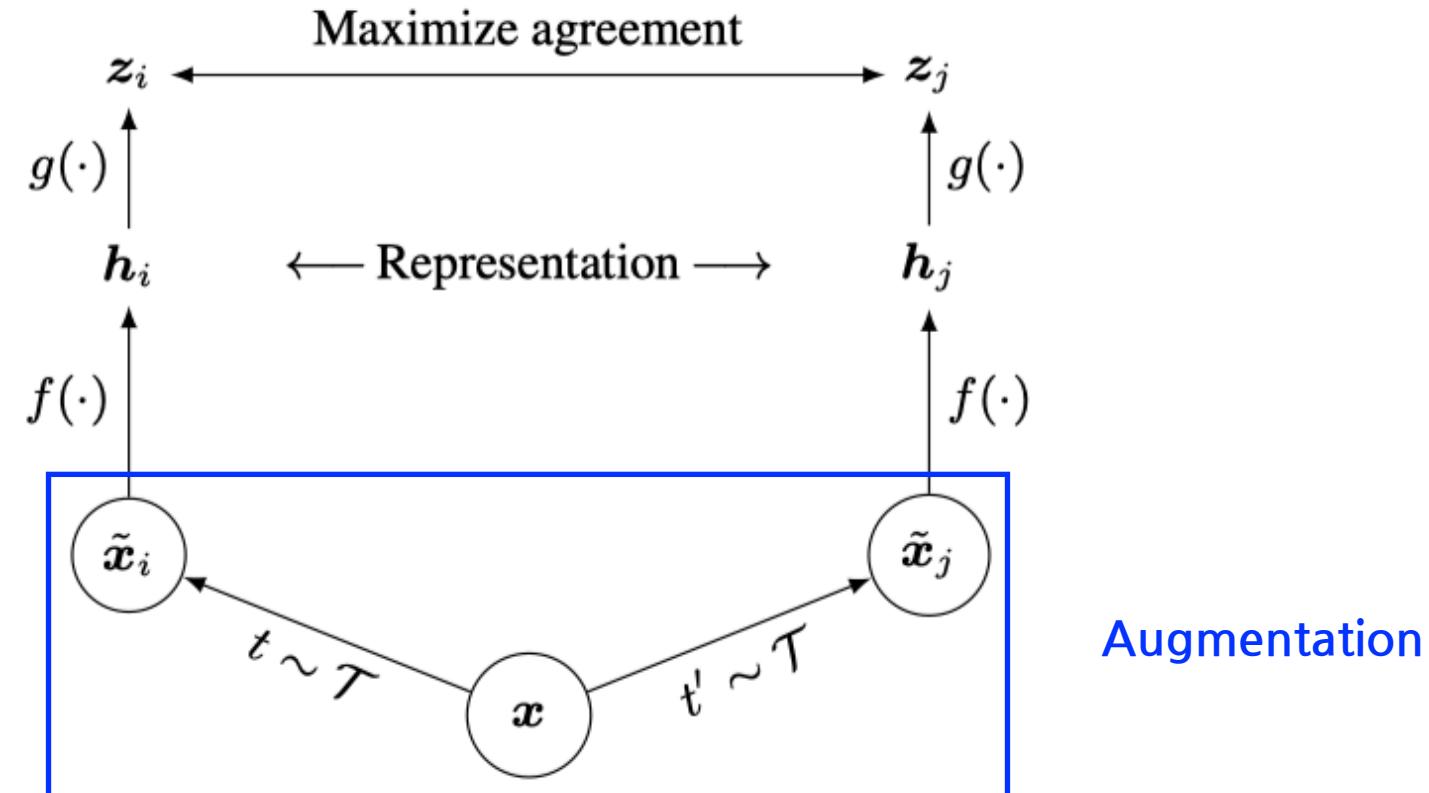
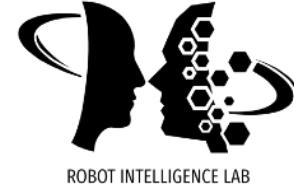


This paper presents a **Simple** framework for **Contrastive Learning** of visual **Representations**.

# Key Findings

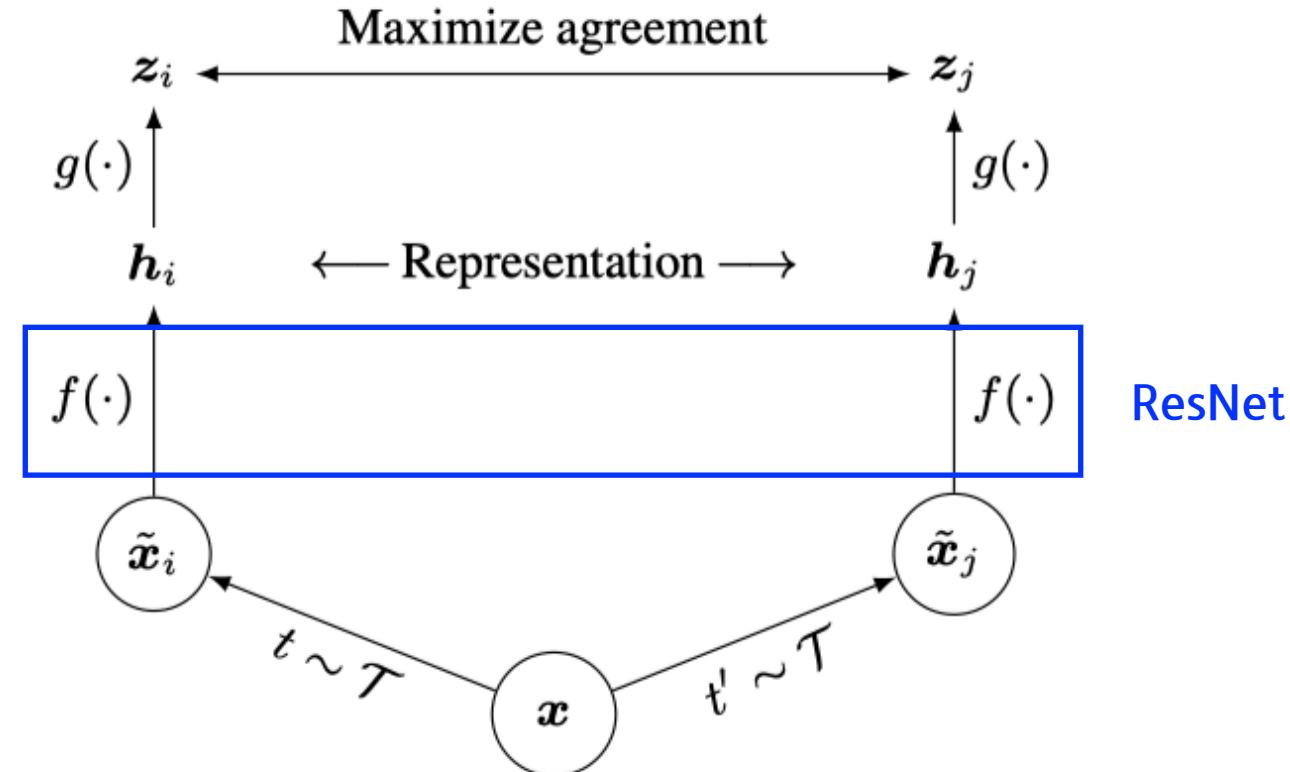
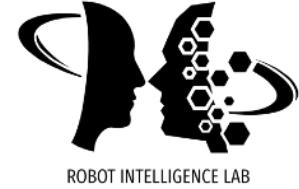
1. Composition of **data augmentations** plays a crucial role
2. Introduction a **learnable nonlinear transformation** between the representation and the contrastive loss substantially improves the quality of the representation
3. Contrastive learning benefits from **larger batch sizes**

# Contrastive Learning



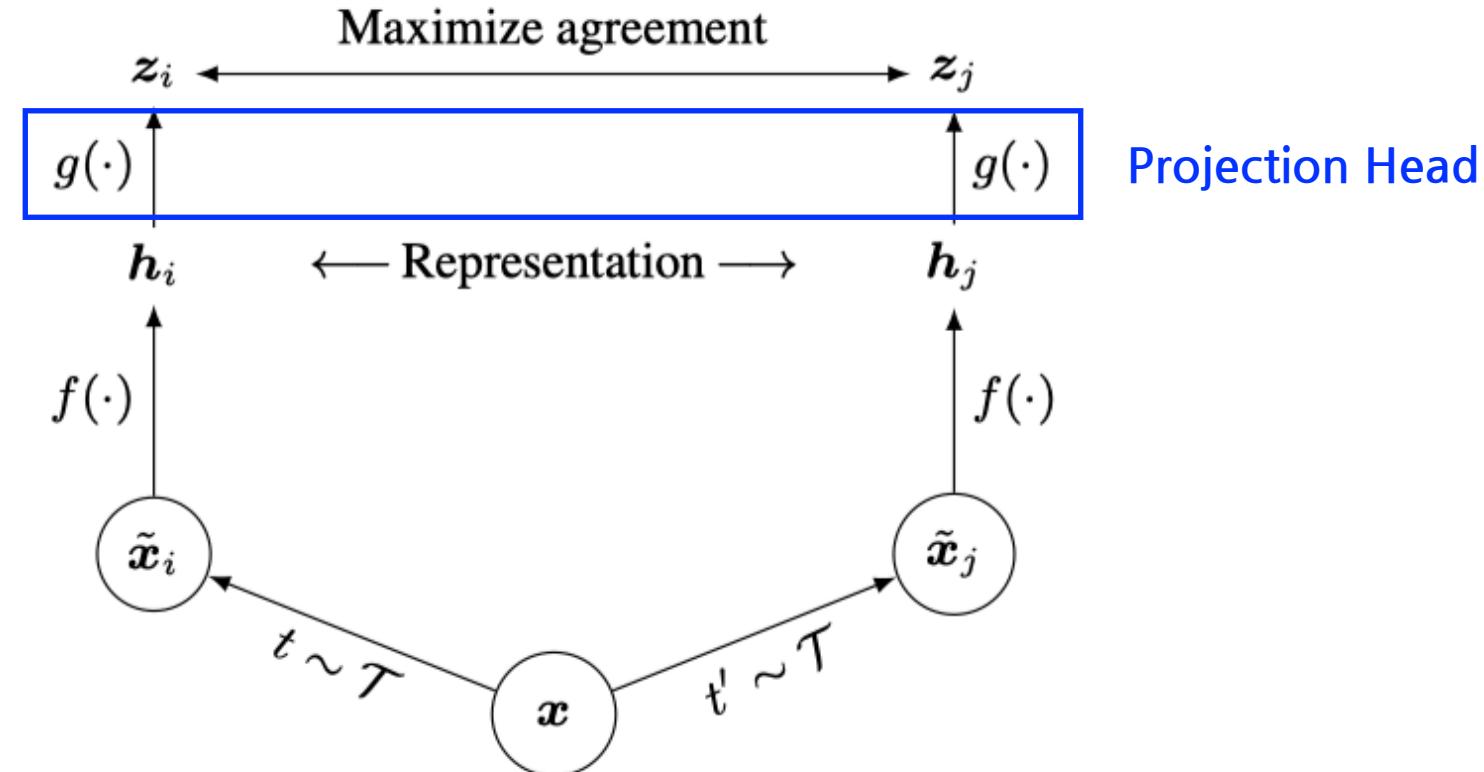
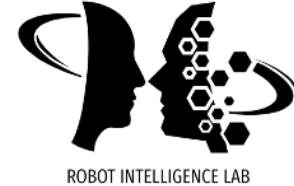
Given a single image  $x$ , two different images are augmented from **stochastic augmentations**:  
1) random cropping followed by resize, 2) random color distortions, 3) random Gaussian blur

# Contrastive Learning



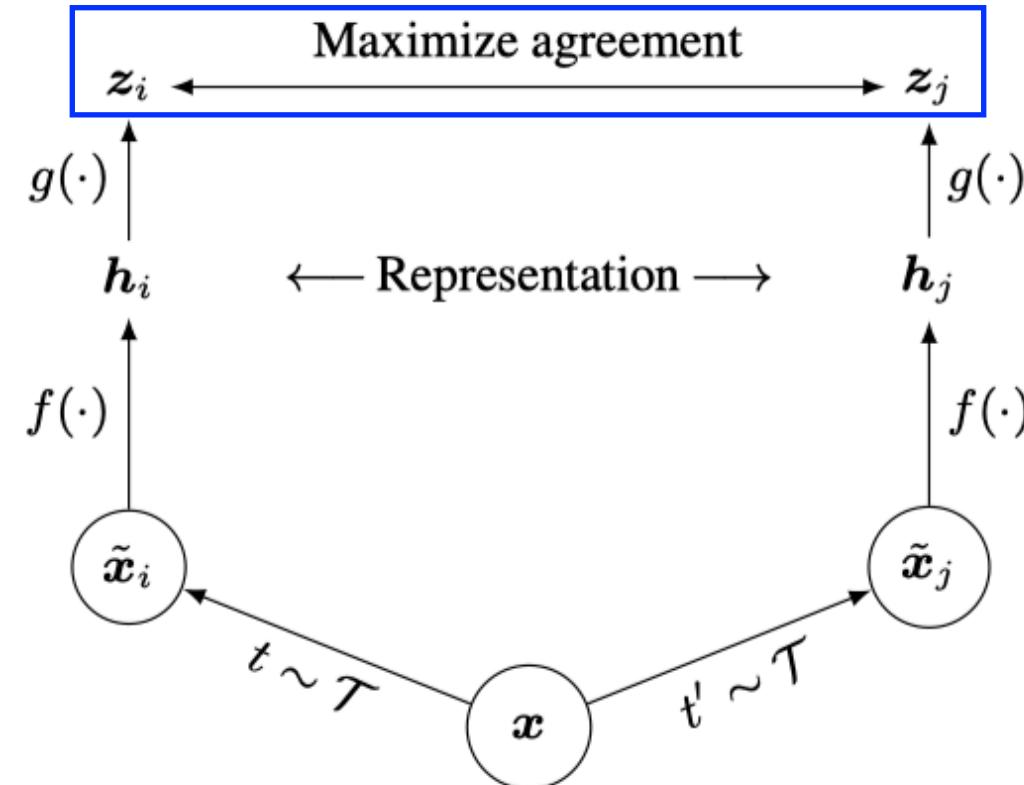
A neural network base encoder  $f(\cdot)$  using ResNet (output after the average pooling).

# Contrastive Learning



A small network (one hidden layer MLP) **projection head**  $g(\cdot)$  that maps representations to the space where **contrastive loss** is applied.

# Contrastive Learning



A contrastive loss function defined for a contrastive prediction task

# NT-Xent Loss



- From  $N$  minibatch examples,  $2N$  data points are derived from augmentation.
- Given a positive pair, the remaining  $2(N - 1)$  augmented examples are treated as negative examples
- The loss function for a positive pair  $(i, j)$  is

$$l_{\textcolor{blue}{i}, \textcolor{red}{j}} = - \log \frac{\exp(\text{sim}(\textcolor{blue}{z}_i, \textcolor{red}{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{I}_{[k \neq i]} \exp(\text{sim}(\textcolor{blue}{z}_i, z_k)/\tau)}$$

- where  $\mathbb{I}_{[k \neq i]} \in \{0,1\}$  is an indicator function.
- The final loss is computed across all pairs of  $(i, j)$  and  $(j, i)$  in a minibatch.
- It is called NT-Xent (the normalized temperature-scaled cross entropy loss).

# SimCLR



---

**Algorithm 1** SimCLR's main learning algorithm.

---

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .  
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do  
    for all  $k \in \{1, \dots, N\}$  do  
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$   
        # the first augmentation  
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$   
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation  
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection  
        # the second augmentation  
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$   
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation  
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection  
    end for  
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do  
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # 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  $g(\cdot)$ 
```

---

For all images in mini-batch

---

**Algorithm 1** SimCLR's main learning algorithm.
 

---

**input:** batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .

**for** sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  **do**

**for all**  $k \in \{1, \dots, N\}$  **do**

draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$

# the first augmentation

$\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$

$\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation

$\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection

# the second augmentation

$\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$

$\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation

$\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection

**end for**

**for all**  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  **do**

$s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # 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  $g(\cdot)$

---

Apply two different augmentations

---

**Algorithm 1** SimCLR's main learning algorithm.
 

---

**input:** batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .

**for** sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  **do**

**for all**  $k \in \{1, \dots, N\}$  **do**

draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$

# the first augmentation

$\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$

$\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation

$\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection

# the second augmentation

$\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$

$\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation

$\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection

**end for**

**for all**  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  **do**

$s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # 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  $g(\cdot)$

---

Symmetric Contrastive loss

---

**Algorithm 1** SimCLR's main learning algorithm.
 

---

**input:** batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .

**for** sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  **do**

**for all**  $k \in \{1, \dots, N\}$  **do**

draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$

# the first augmentation

$\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$

$\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation

$\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection

# the second augmentation

$\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$

$\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation

$\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection

**end for**

**for all**  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  **do**

$s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # 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  $\mathcal{L}$

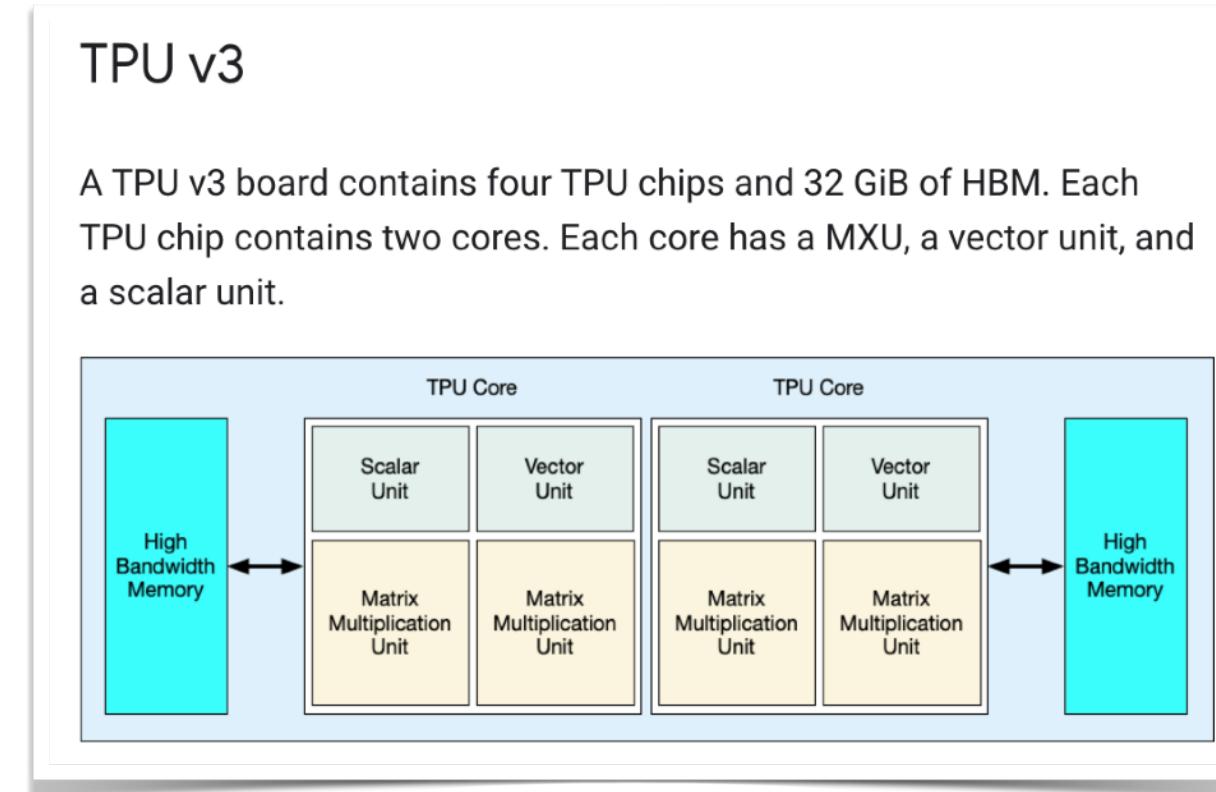
**end for**

**return** encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$

Once trained, use  $f(\cdot)$  only

# Larger Batch Size

- The batch size varies from 256 to 8,192 and when using 8,192 batch size, we get 16,382 negative examples.
- For 8,192 batch size, they used 128 TPU v3 cores..



# Data Augmentation



(a) Original



(b) Crop and resize



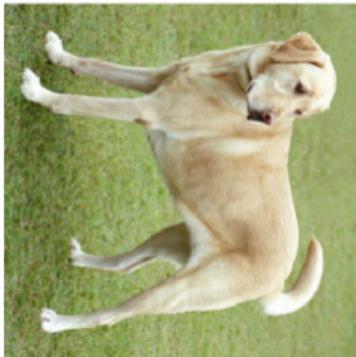
(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate  $\{90^\circ, 180^\circ, 270^\circ\}$



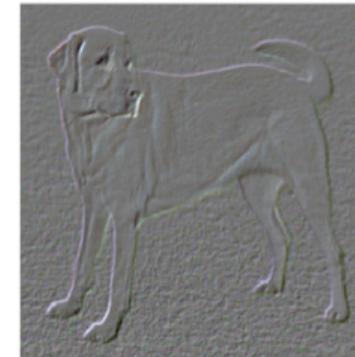
(g) Cutout



(h) Gaussian noise



(i) Gaussian blur



(j) Sobel filtering

Different data (image) augmentation operators

# Data Augmentation



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate  $\{90^\circ, 180^\circ, 270^\circ\}$



(g) Cutout



(h) Gaussian noise



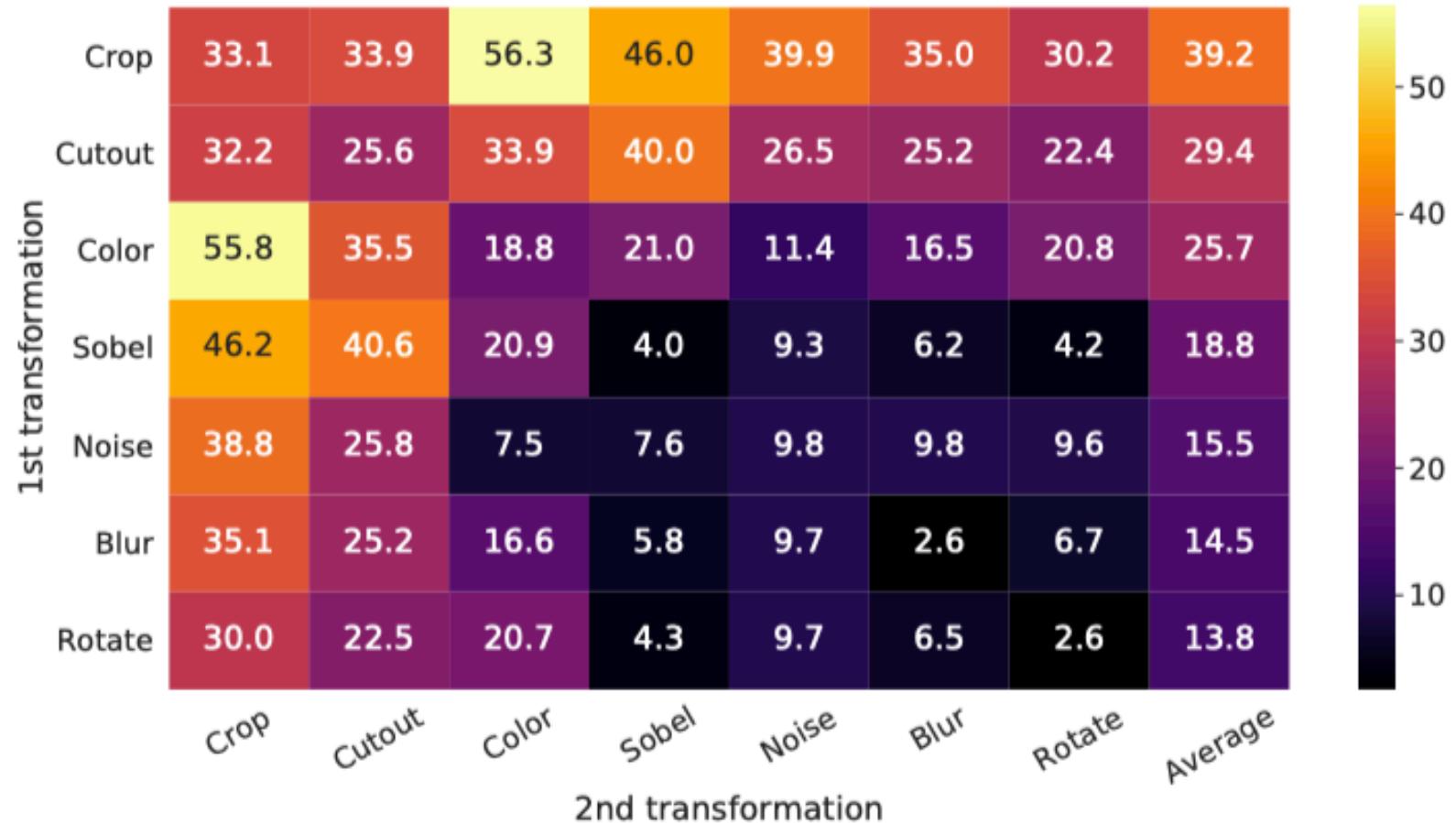
(i) Gaussian blur



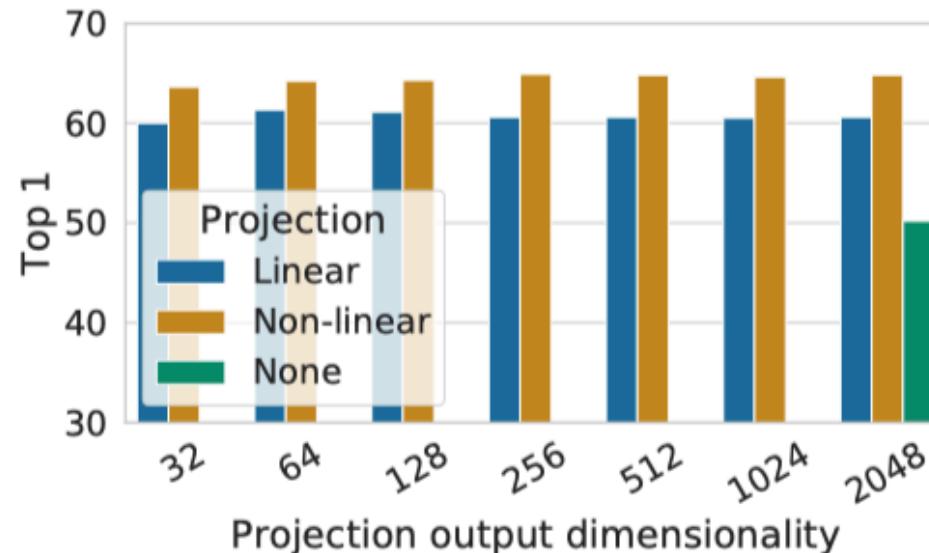
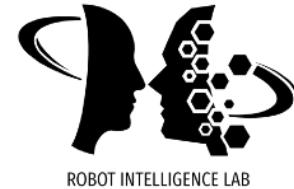
(j) Sobel filtering

The **ones** used in this paper

# ImageNet Top-1 Accuracy

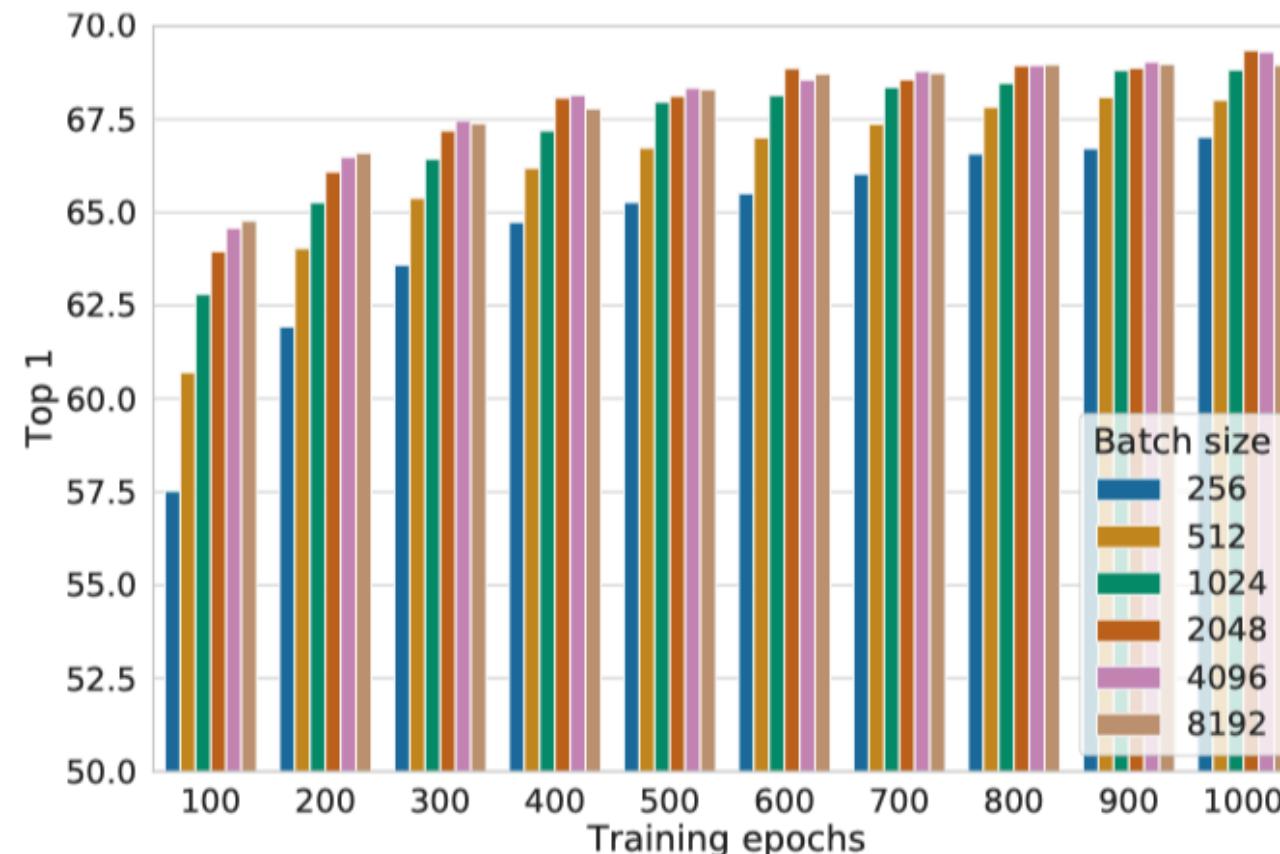


# Effect of Projection Head



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

# Effect of Batch Size



*Figure 9.* Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.<sup>10</sup>

How many GPUs does Google have...

# Linear Probing on ImageNet



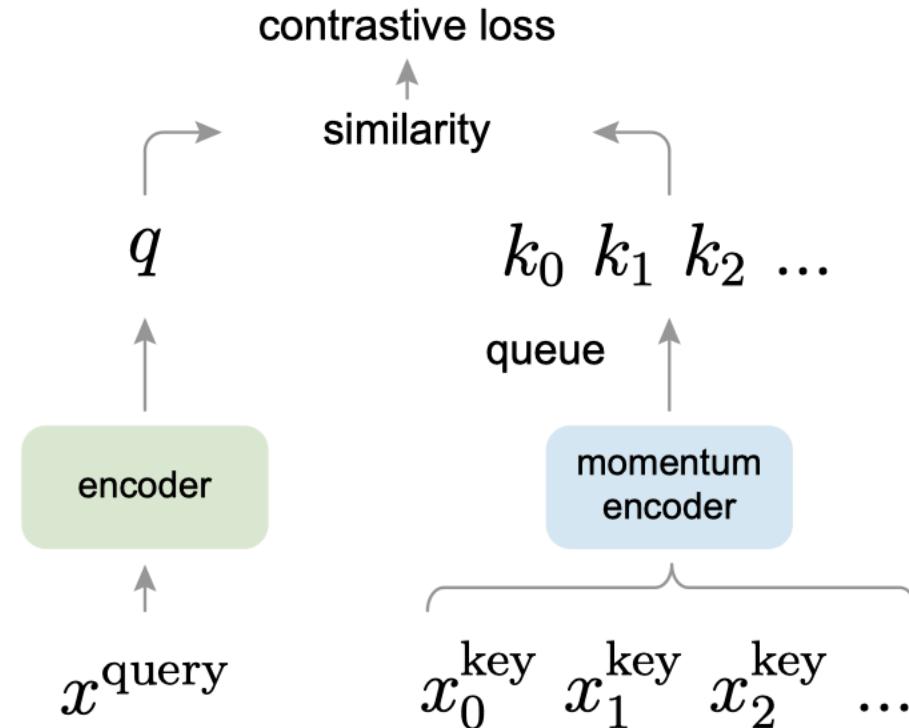
Method	Architecture	Param (M)	Top 1	Top 5
<i>Methods using ResNet-50:</i>				
Local Agg.	ResNet-50	24	60.2	-
MoCo	ResNet-50	24	60.6	-
PIRL	ResNet-50	24	63.6	-
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	<b>69.3</b>	<b>89.0</b>
<i>Methods using other architectures:</i>				
Rotation	RevNet-50 (4×)	86	55.4	-
BigBiGAN	RevNet-50 (4×)	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	-
CMC	ResNet-50 (2×)	188	68.4	88.2
MoCo	ResNet-50 (4×)	375	68.6	-
CPC v2	ResNet-161 (*)	305	71.5	90.1
SimCLR (ours)	ResNet-50 (2×)	94	74.2	92.0
SimCLR (ours)	ResNet-50 (4×)	375	<b>76.5</b>	<b>93.2</b>



# MOCO

"Momentum Contrast for Unsupervised Visual Representation Learning,"  
2020

# MoCo



- MoCo is a mechanism for building dynamic dictionaries for contrastive learning: a query matches a key if they are encoded views (e.g., data augmentation) of the same image.

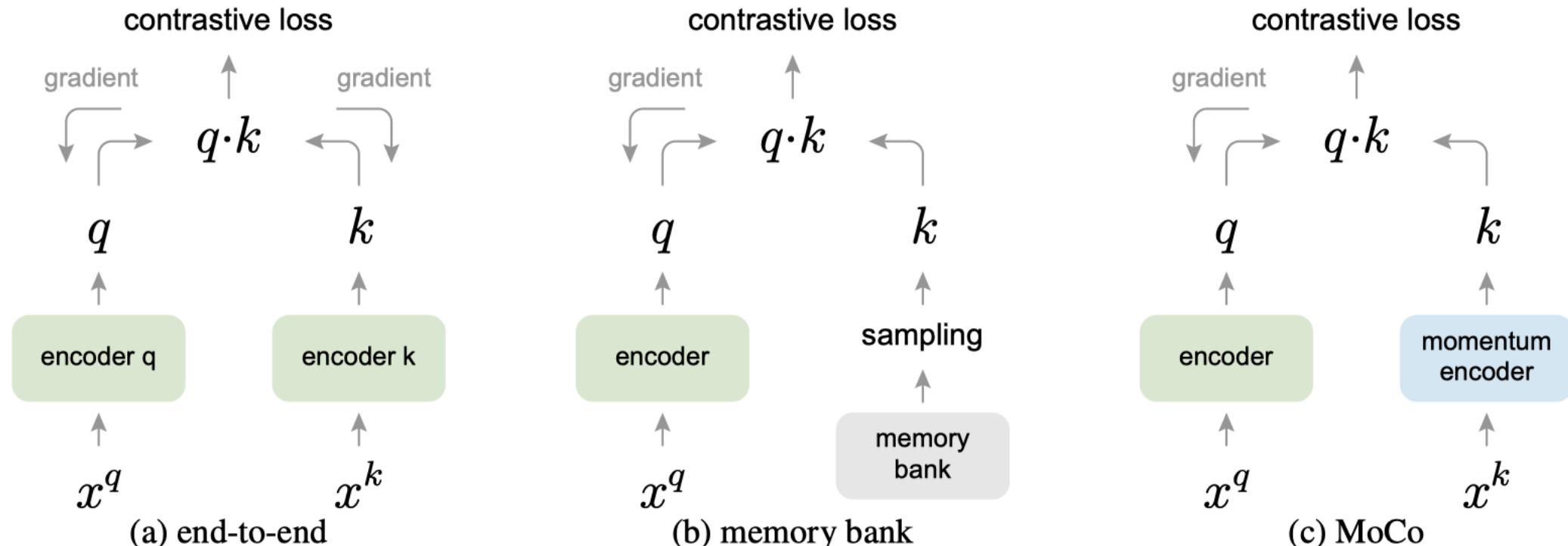
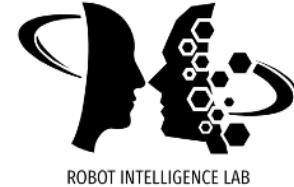
# InfoNCE Loss

- A contrastive loss is a function whose value is low when the query  $q$  is similar to its positive key  $k_+$  and dissimilar to all other keys (considered for  $q$ )

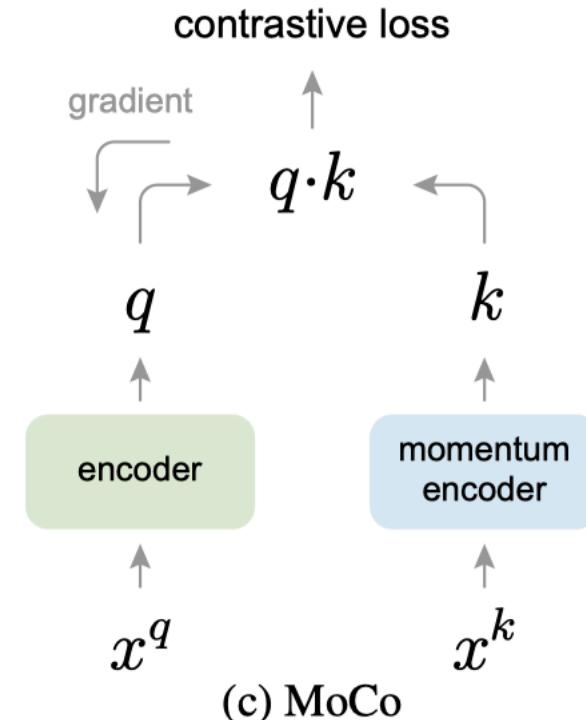
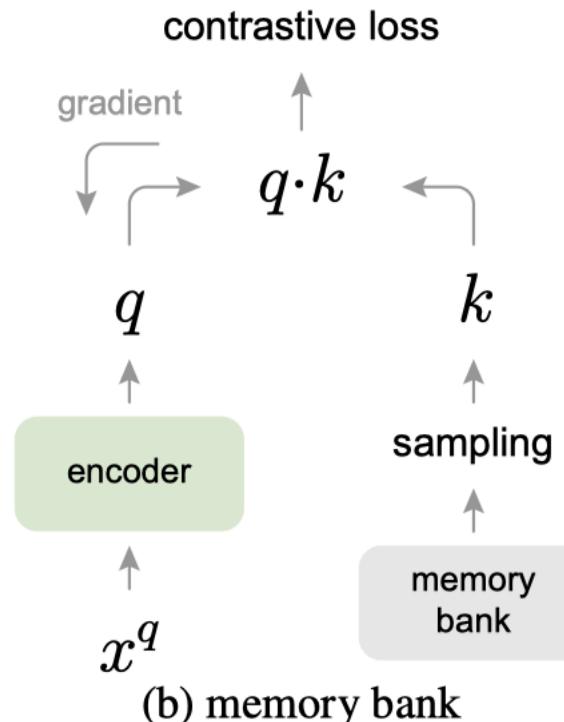
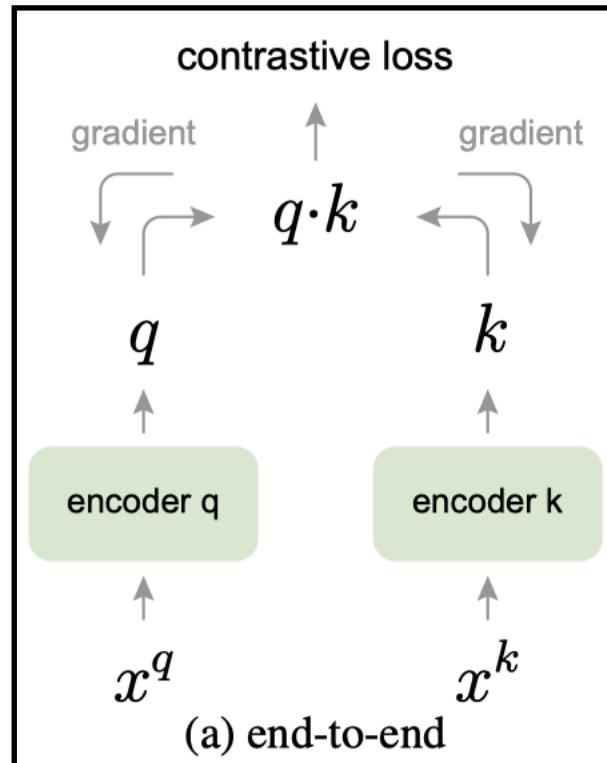
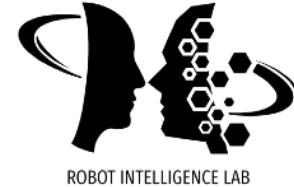
$$\mathcal{L}_q = \log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

where  $\tau$  is a temperature hyper-parameter. The sum is over one positive ( $i = 0$ ) and  $K$  negative samples. Intuitively, this loss is the log loss of a  $(K + 1)$ -way softmax-based classifier.

# Three Contrastive Losses

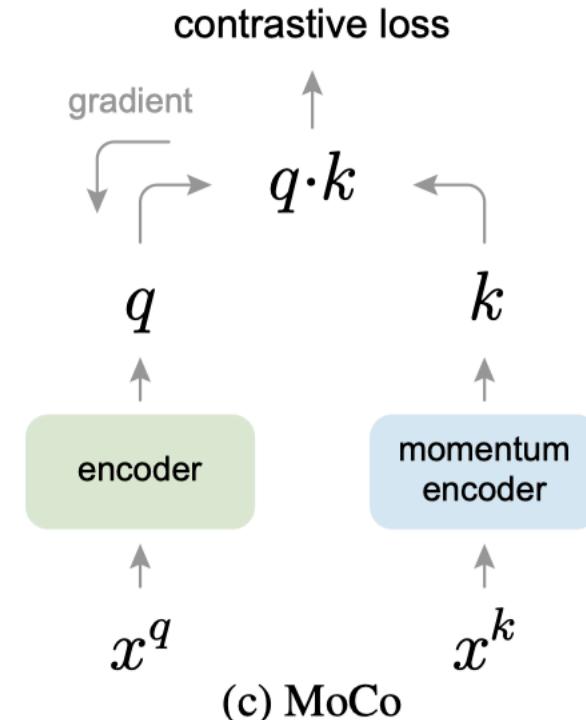
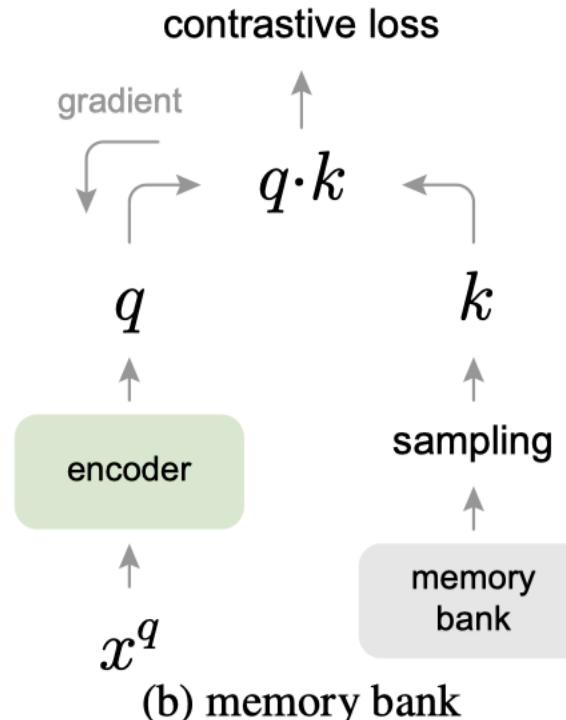
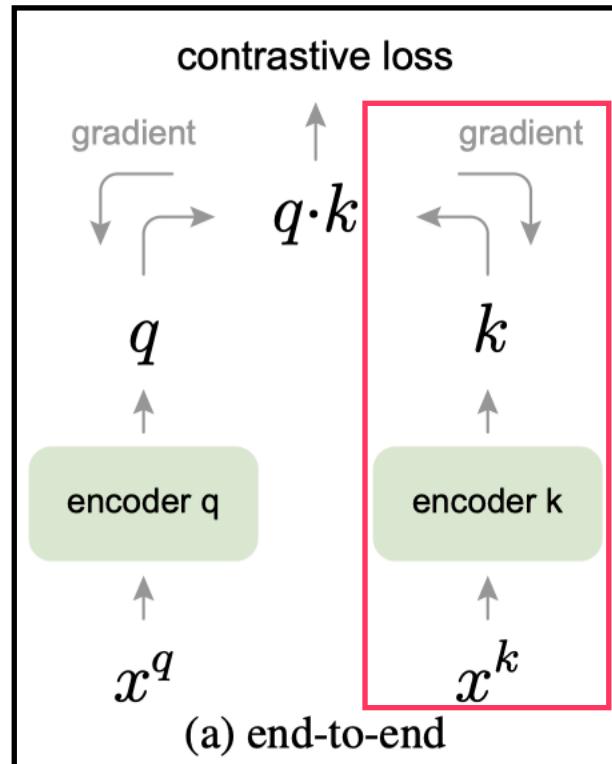
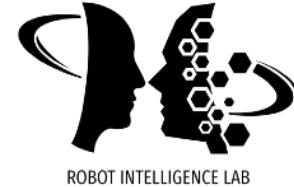


# Three Contrastive Losses



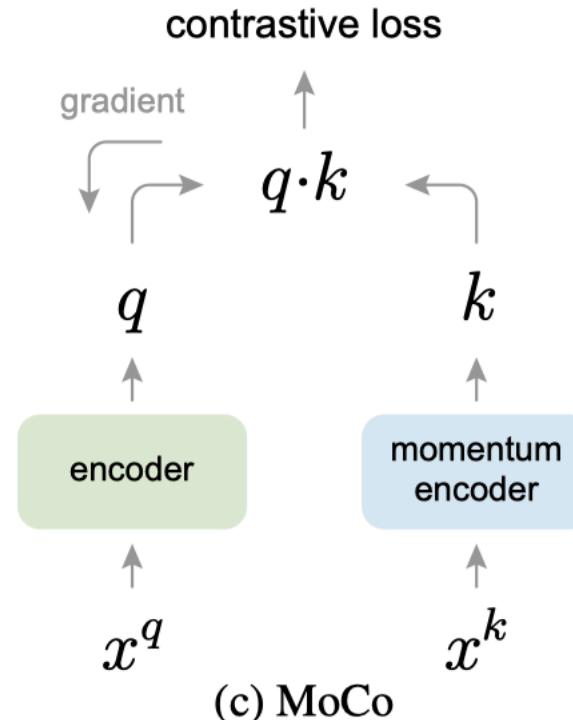
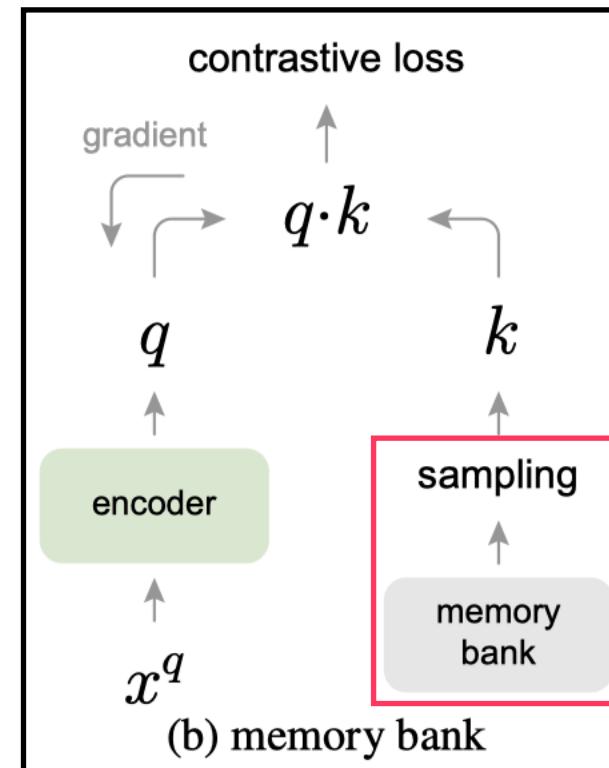
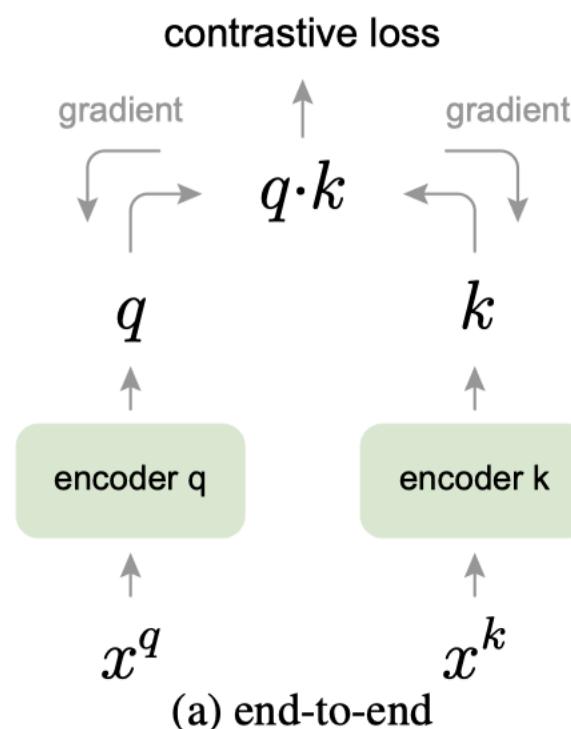
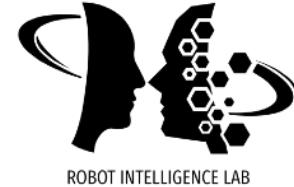
This one is similar to simCLR.

# Three Contrastive Losses



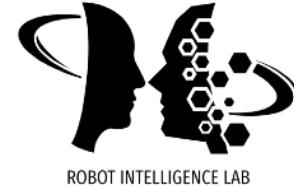
This one is similar to simCLR.  
It requires huge GPU memory!

# Three Contrastive Losses



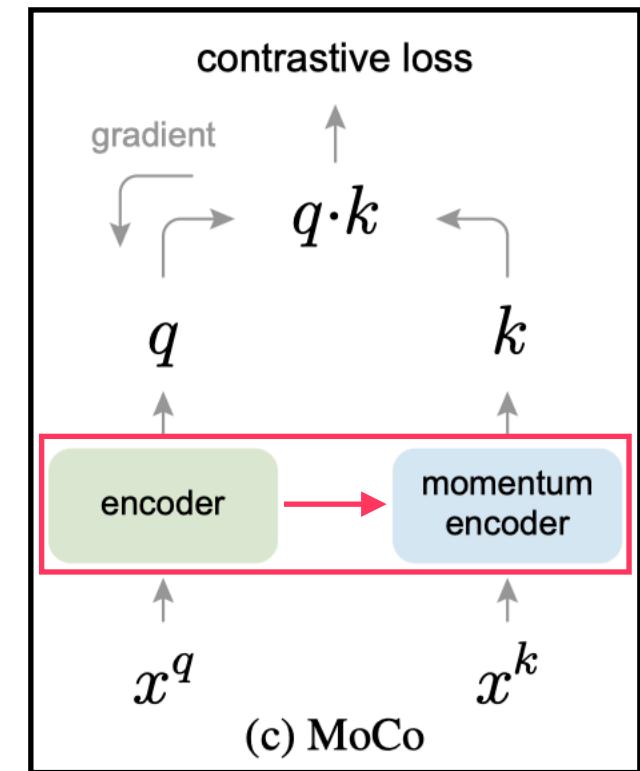
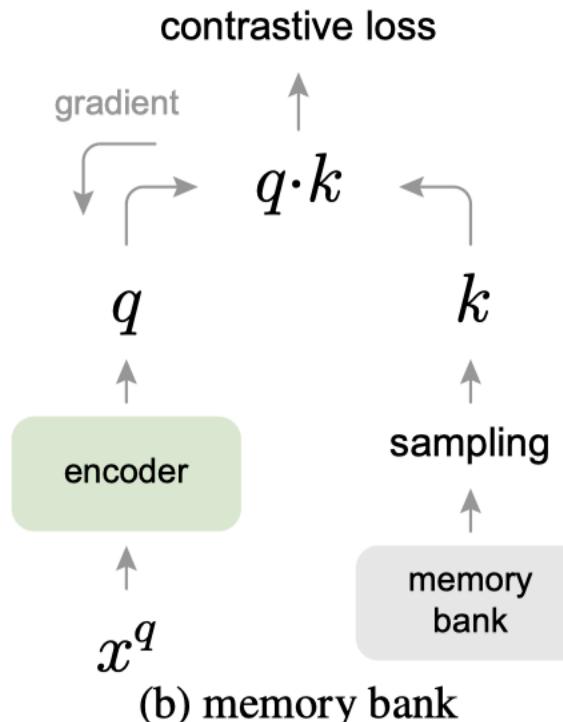
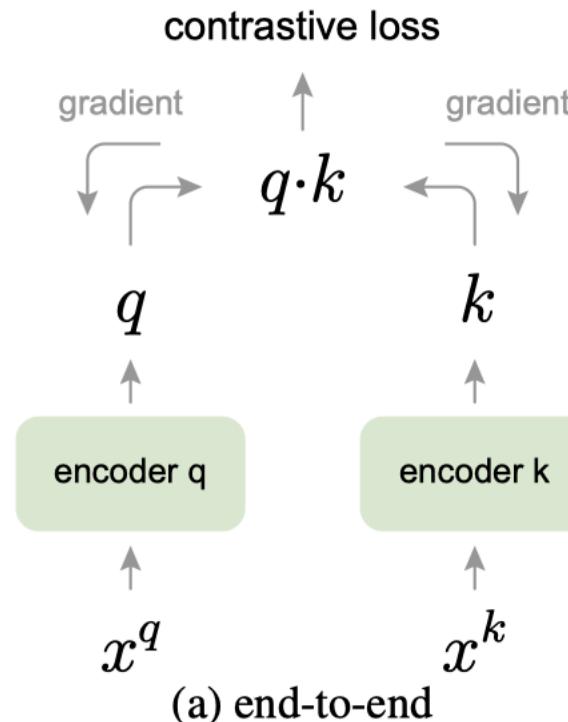
This one requires less memory in that gradient only flows with the positive encoder.  
However, it requires an additional memory bank.

# Good Dictionary?



"Our hypothesis is that good features can be learned by a large dictionary that covers a rich set of negative samples, while the encoder for the dictionary keys is kept as consistent as possible despite its evolution."

# MoCo



The key idea of MoCo is to use a queue (first-in, first-out) to make the dictionary large  
+ update the key encoder using a momentum update

---

## Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

---

```

# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
    x_q = aug(x) # a randomly augmented version
    x_k = aug(x) # another randomly augmented version

    q = f_q.forward(x_q) # queries: NxC
    k = f_k.forward(x_k) # keys: NxC
    k = k.detach() # no gradient to keys

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

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

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

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

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

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

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

```

---

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

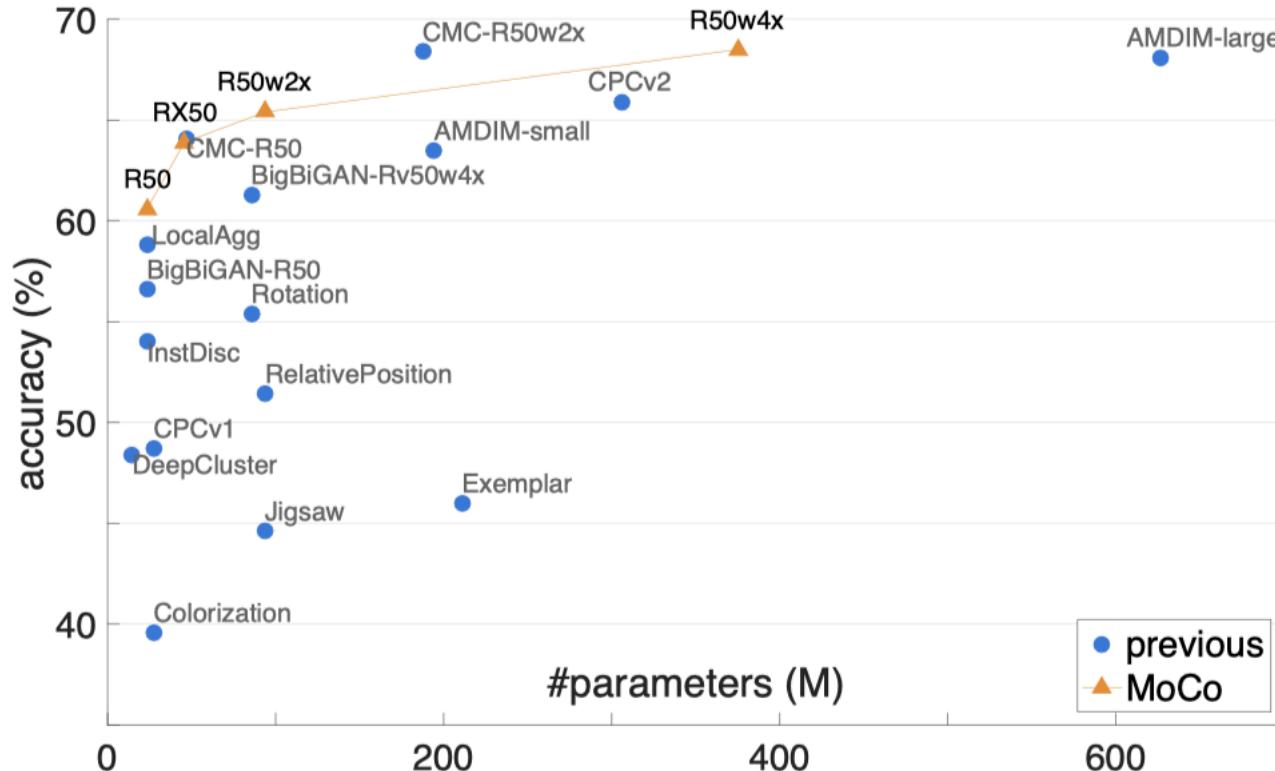
# Ablation: Momentum



**Ablation: momentum.** The table below shows ResNet-50 accuracy with different MoCo momentum values ( $m$  in Eqn.(2)) used in pre-training ( $K = 4096$  here) :

momentum $m$	0	0.9	0.99	0.999	0.9999
accuracy (%)	<i>fail</i>	55.2	57.8	59.0	58.9

# ImageNet Linear Probing



SimCLR paper

Method	Architecture	Param (M)	Top 1	Top 5
<i>Methods using ResNet-50:</i>				
Local Agg.	ResNet-50	24	60.2	-
MoCo	ResNet-50	24	60.6	-
PIRL	ResNet-50	24	63.6	-
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	<b>69.3</b>	<b>89.0</b>
<i>Methods using other architectures:</i>				
Rotation	RevNet-50 (4×)	86	55.4	-
BigBiGAN	RevNet-50 (4×)	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	-
CMC	ResNet-50 (2×)	188	68.4	88.2
MoCo	ResNet-50 (4×)	375	68.6	-
CPC v2	ResNet-161 (*)	305	71.5	90.1
SimCLR (ours)	ResNet-50 (2×)	94	74.2	92.0
SimCLR (ours)	ResNet-50 (4×)	375	<b>76.5</b>	<b>93.2</b>

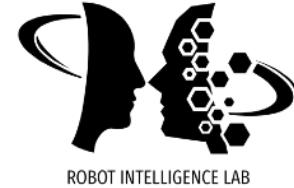


# BYOL

"Bootstrap Your Own Latent A New Approach to Self-Supervised Learning," 2020

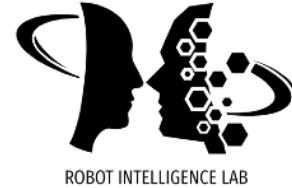


# Contrastive Learning



Previous SSL methods based on contrastive learning (e.g., DrLIM, SimCLR, MoCo, SimCLRv2) requires both positive (similar) and negative (dissimilar) images.

# BYOL



While state-of-the-art methods rely on **negative pairs**, BYOL achieves a new state-of-the-art **without them**.

# BYOL

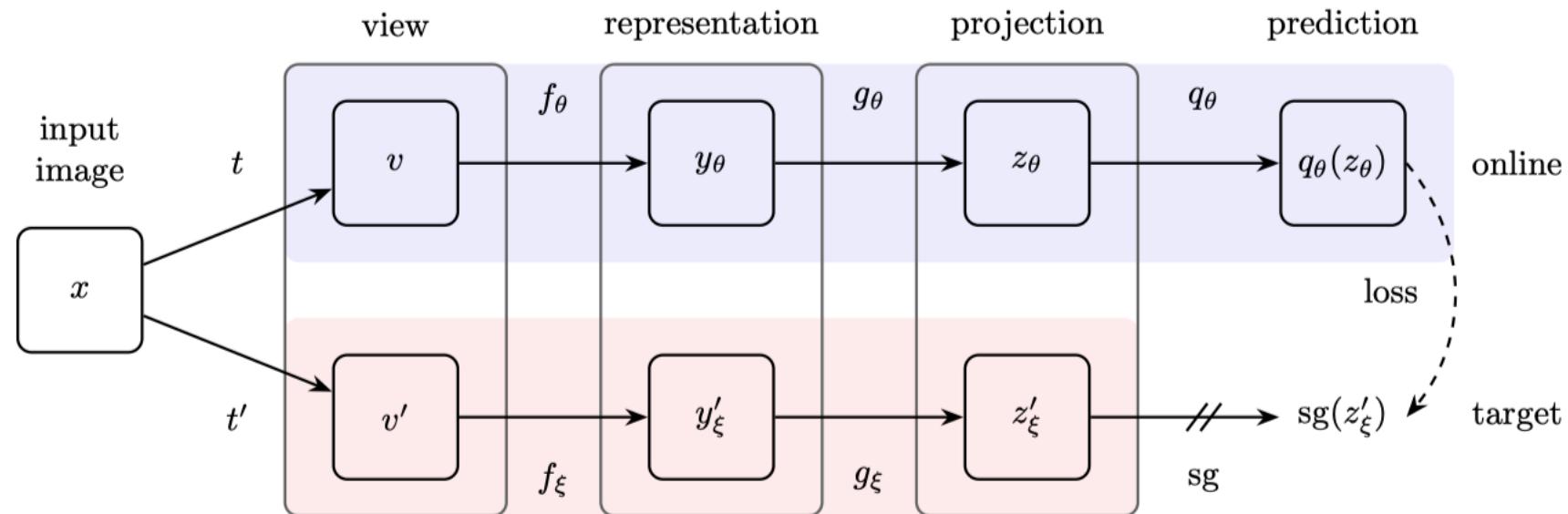


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between  $q_\theta(z_\theta)$  and  $sg(z'_\xi)$ , where  $\theta$  are the trained weights,  $\xi$  are an exponential moving average of  $\theta$  and sg means stop-gradient. At the end of training, everything but  $f_\theta$  is discarded,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

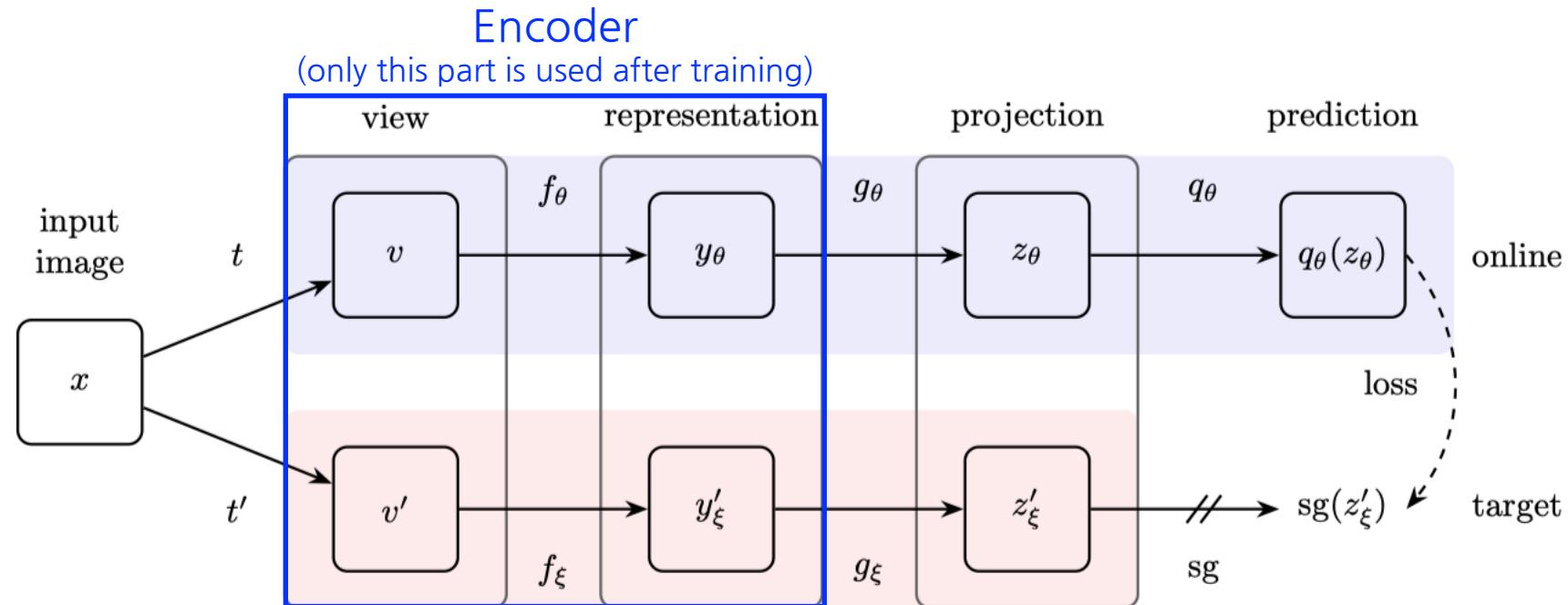


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between  $q_\theta(z_\theta)$  and  $sg(z'_\xi)$ , where  $\theta$  are the trained weights,  $\xi$  are an exponential moving average of  $\theta$  and sg means stop-gradient. At the end of training, everything but  $f_\theta$  is discarded,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

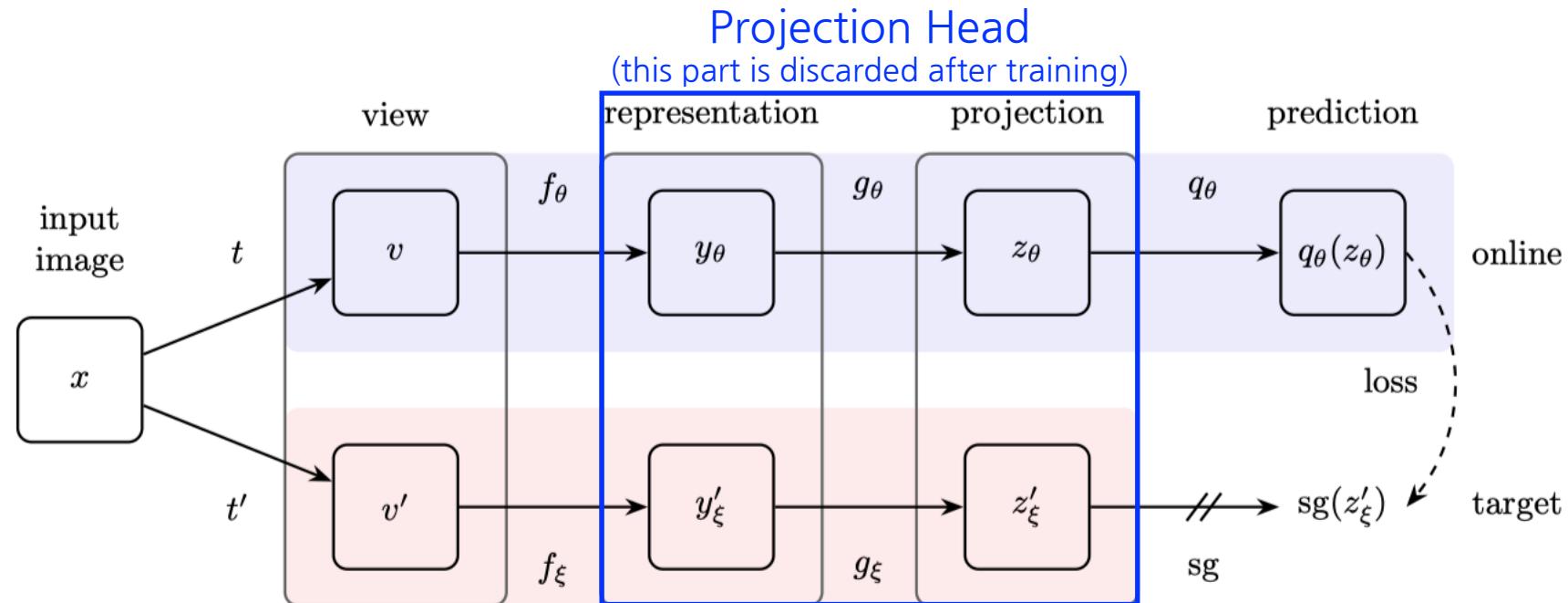


Figure 2: BYOL’s architecture. BYOL minimizes a similarity loss between  $q_\theta(z_\theta)$  and  $sg(z'_\xi)$ , where  $\theta$  are the trained weights,  $\xi$  are an exponential moving average of  $\theta$  and  $sg$  means stop-gradient. At the end of training, everything but  $f_\theta$  is discarded,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

# BYOL

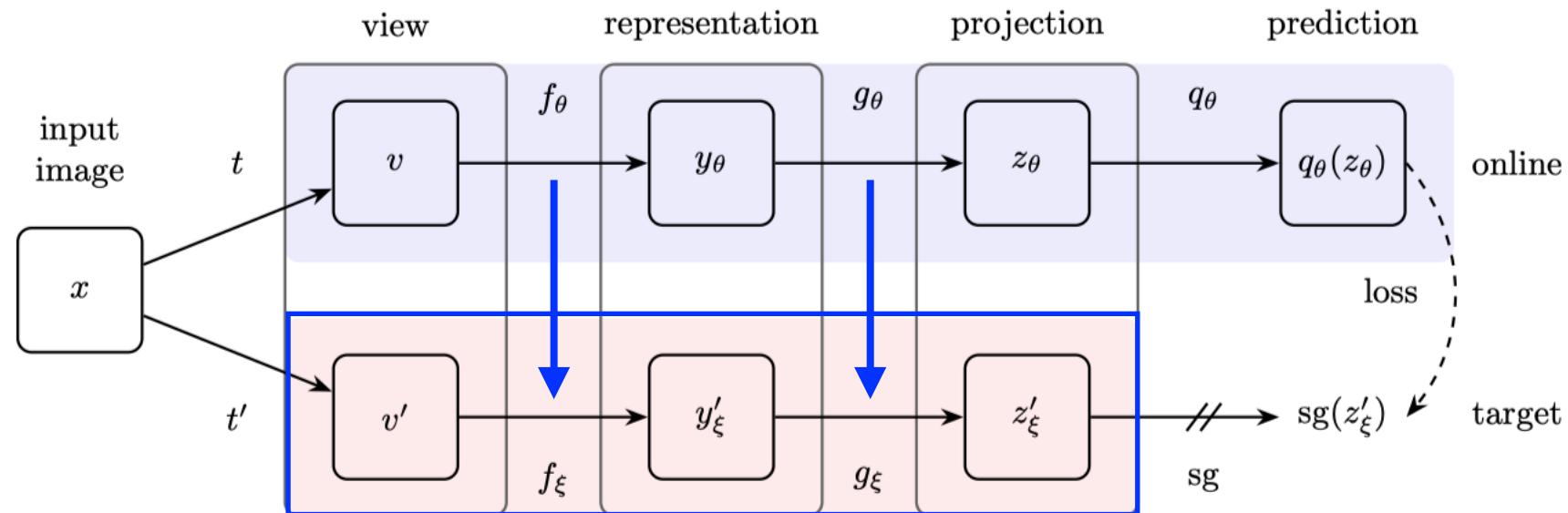


Figure 2: BYOL’s architecture. BYOL minimizes a similarity loss between  $q_\theta(z_\theta)$  and  $\text{sg}(z'_\xi)$ , where  $\theta$  are the trained weights,  $\xi$  are an exponential moving average of  $\theta$  and sg means stop-gradient. At the end of training, everything but  $f_\theta$  is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

Both target encoder and target projection head are updated via EMA.

# Update Rule

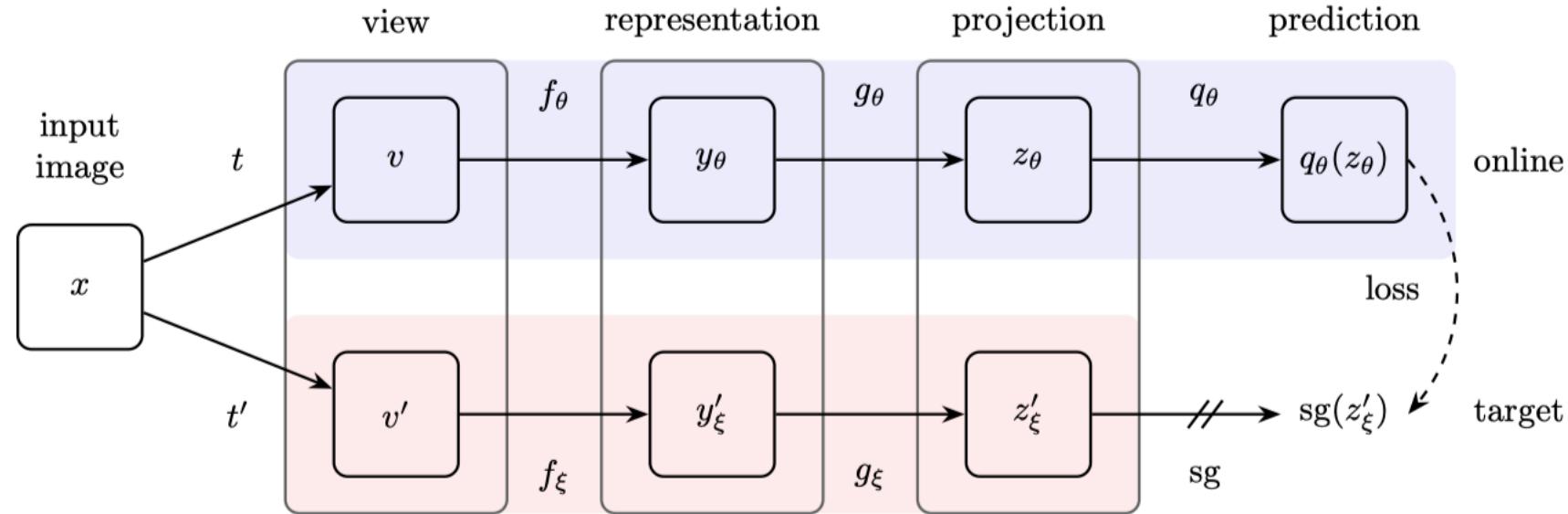


Figure 2: BYOL’s architecture. BYOL minimizes a similarity loss between  $q_\theta(z_\theta)$  and  $sg(z'_{\xi})$ , where  $\theta$  are the trained weights,  $\xi$  are an exponential moving average of  $\theta$  and sg means stop-gradient. At the end of training, everything but  $f_\theta$  is discarded,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

$$\mathcal{L}_{\theta,\xi}(v, v') = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_{\xi} \rangle}{\|q_\theta(z_\theta)\|_2 \cdot \|z'_{\xi}\|_2}, \quad \mathcal{L}_{\theta,\xi}^{\text{BYOL}} = \frac{1}{2} \left( \mathcal{L}_{\theta,\xi}(v, v') + \mathcal{L}_{\theta,\xi}(v', v) \right)$$

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}_{\theta,\xi}^{\text{BYOL}}$$

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

**Algorithm 1:** BYOL: Bootstrap Your Own Latent**Inputs :**

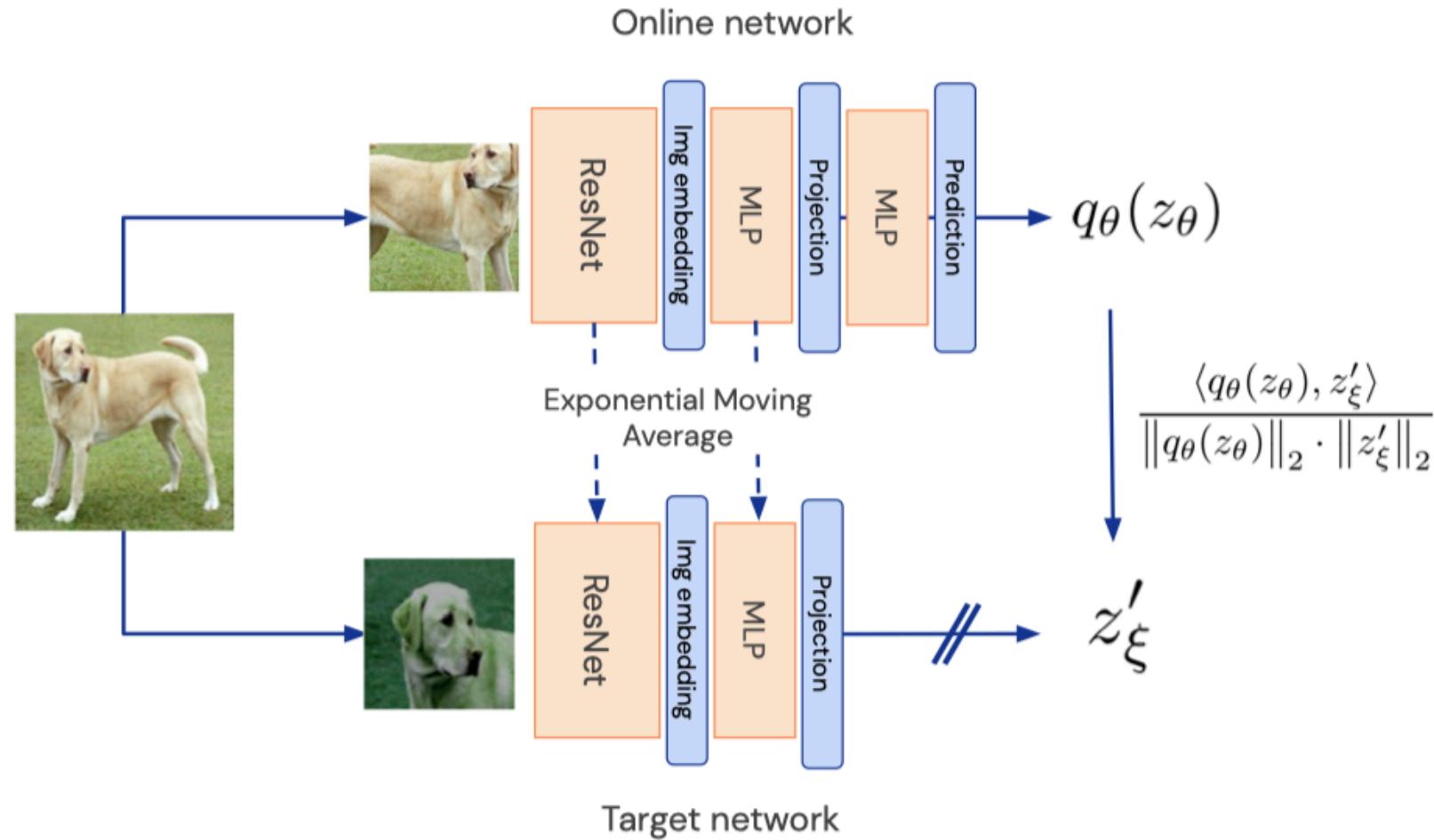
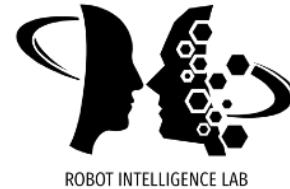
$\mathcal{D}$ , $\mathcal{T}$ , and $\mathcal{T}'$	set of images and distributions of transformations
$\theta$ , $f_\theta$ , $g_\theta$ , and $q_\theta$	initial online parameters, encoder, projector, and predictor
$\xi$ , $f_\xi$ , $g_\xi$	initial target parameters, target encoder, and target projector
optimizer	optimizer, updates online parameters using the loss gradient
$K$ and $N$	total number of optimization steps and batch size
$\{\tau_k\}_{k=1}^K$ and $\{\eta_k\}_{k=1}^K$	target network update schedule and learning rate schedule

```

1 for  $k = 1$  to  $K$  do
2    $\mathcal{B} \leftarrow \{x_i \sim \mathcal{D}\}_{i=1}^N$                                      // sample a batch of  $N$  images
3   for  $x_i \in \mathcal{B}$  do
4      $t \sim \mathcal{T}$  and  $t' \sim \mathcal{T}'$                                          // sample image transformations
5      $z_1 \leftarrow g_\theta(f_\theta(t(x_i)))$  and  $z_2 \leftarrow g_\theta(f_\theta(t'(x_i)))$  // compute projections
6      $z'_1 \leftarrow g_\xi(f_\xi(t'(x_i)))$  and  $z'_2 \leftarrow g_\xi(f_\xi(t(x_i)))$  // compute target projections
7      $l_i \leftarrow -2 \cdot \left( \frac{\langle q_\theta(z_1), z'_1 \rangle}{\|q_\theta(z_1)\|_2 \cdot \|z'_1\|_2} + \frac{\langle q_\theta(z_2), z'_2 \rangle}{\|q_\theta(z_2)\|_2 \cdot \|z'_2\|_2} \right)$  // compute the loss for  $x_i$ 
8   end
9    $\delta\theta \leftarrow \frac{1}{N} \sum_{i=1}^N \partial_\theta l_i$                                 // compute the total loss gradient w.r.t.  $\theta$ 
10   $\theta \leftarrow \text{optimizer}(\theta, \delta\theta, \eta_k)$                                     // update online parameters
11   $\xi \leftarrow \tau_k \xi + (1 - \tau_k) \theta$                                          // update target parameters
12 end
Output: encoder  $f_\theta$ 

```

# BYOL



# Linear Probing

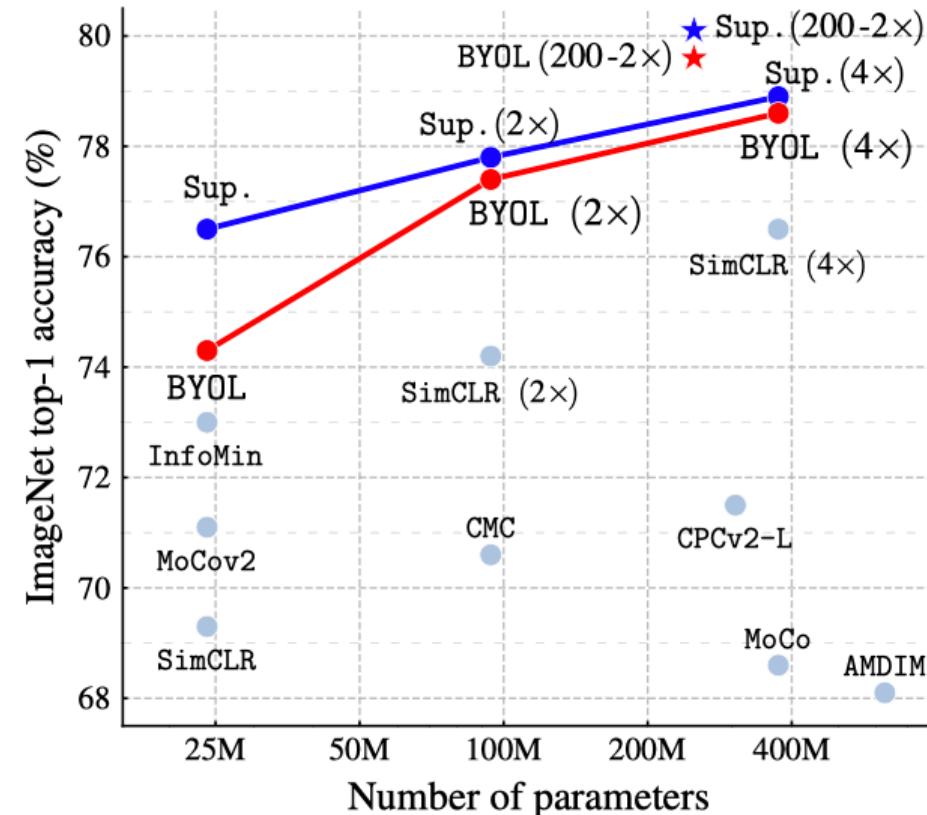
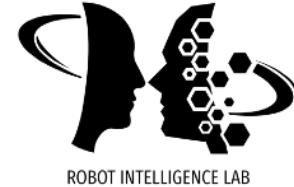
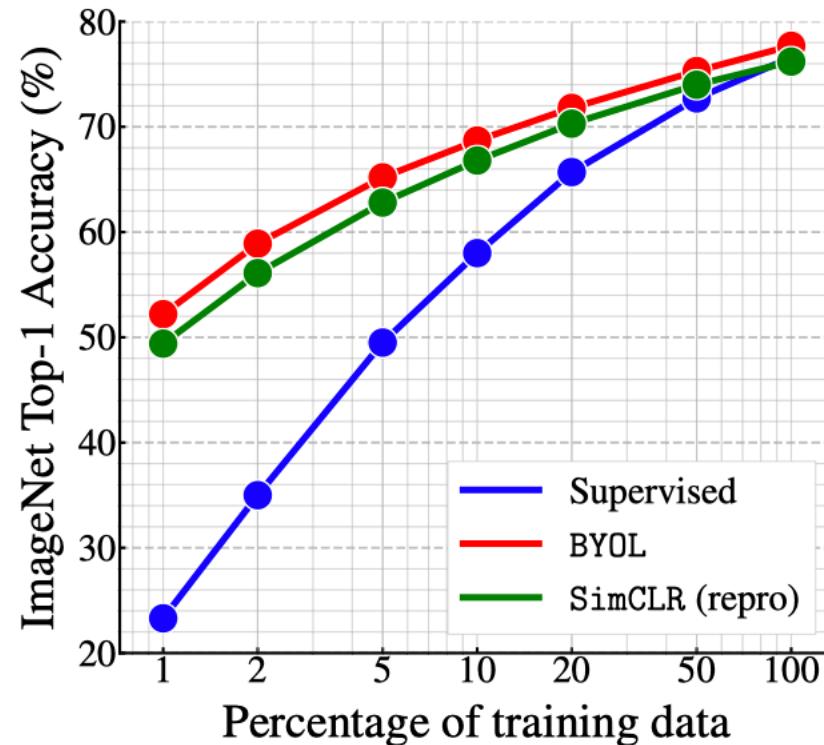
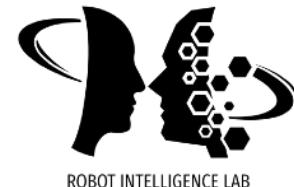
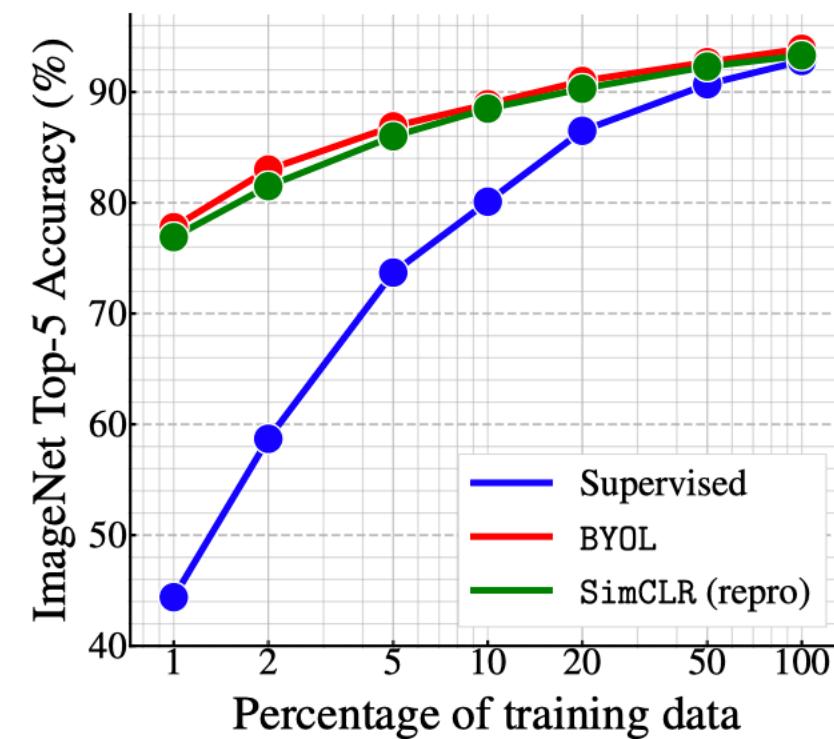


Figure 1: Performance of BYOL on ImageNet (linear evaluation) using ResNet-50 and our best architecture ResNet-200 ( $2\times$ ), compared to other unsupervised and supervised (Sup.) baselines [8].

# Fraction of Labels



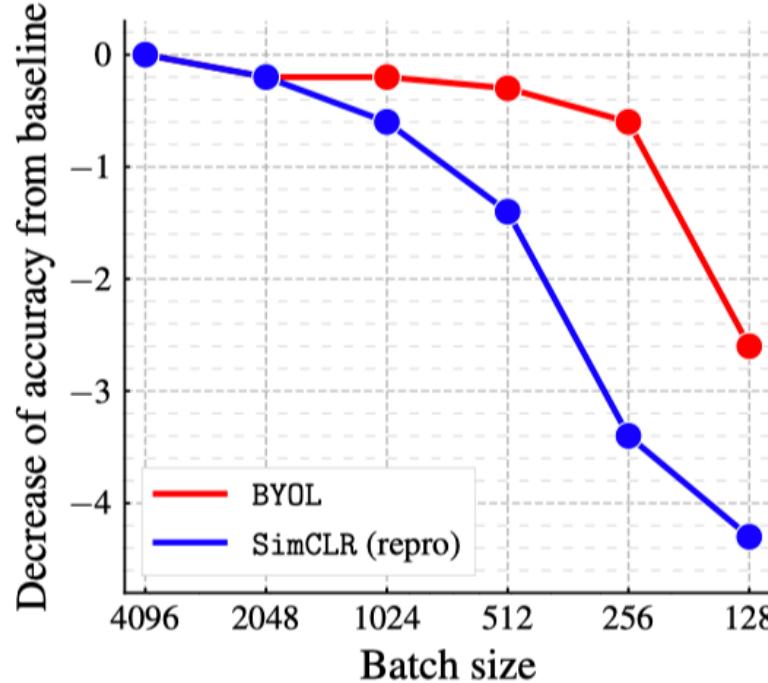
(a) Top-1 accuracy



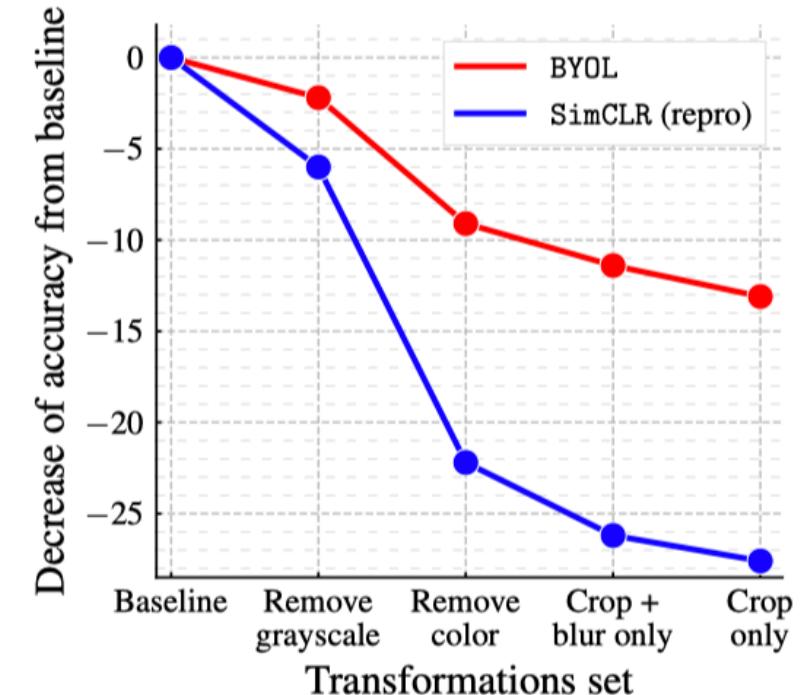
(b) Top-5 accuracy

Figure 4: Semi-supervised training with a fraction of ImageNet labels on a ResNet-50 ( $\times 1$ ).

# Ablation Study



(a) Impact of batch size



(b) Impact of progressively removing transformations

Figure 3: Decrease in top-1 accuracy (in % points) of BYOL and our own reproduction of SimCLR at 300 epochs, under linear evaluation on ImageNet.

BYOL is more robust to smaller batch sizes and less image augmentations.



# SimCLRv2

"Big Self-Supervised Models are Strong Semi-Supervised Learners," 2020

# Key Ingredients



Use big (deep and wide) networks during pre-training using **unsupervised data**

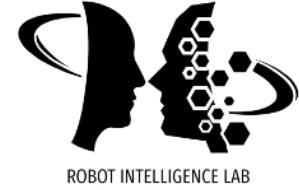


Use big (deep and wide) networks during fine-tuning using **supervised data**



Train smaller networks with **distillation** using **unsupervised data**

# Interesting Result



SimCLRv2 achieves 73.9% ImageNet top-1 accuracy with 1% of labels using ResNet-50.

# Interesting Result

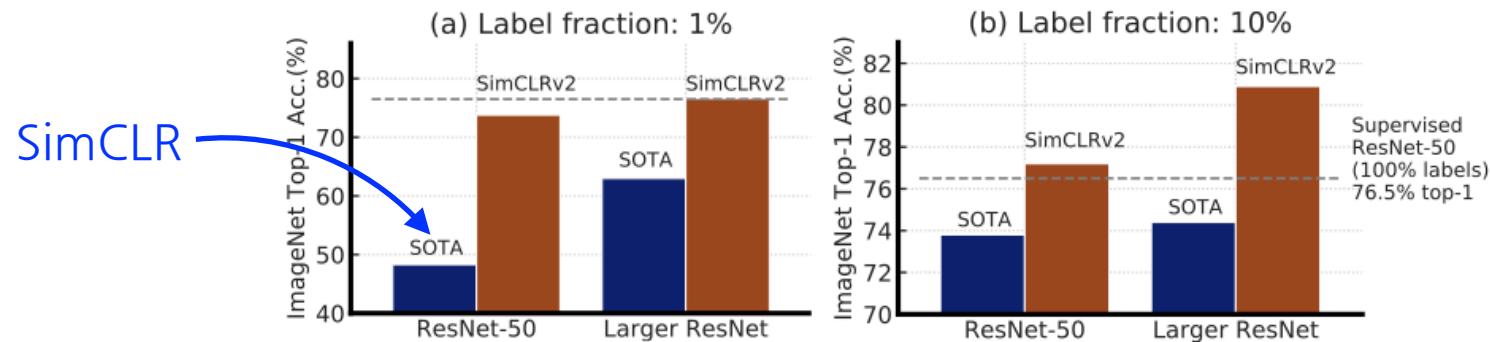
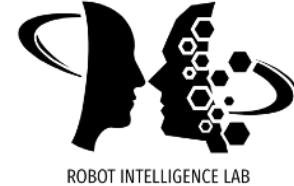
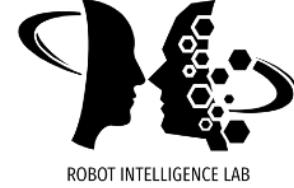


Figure 2: Top-1 accuracy of previous state-of-the-art (SOTA) methods [1, 2] and our method (SimCLRv2) on ImageNet using only 1% or 10% of the labels. Dashed line denotes fully supervised ResNet-50 trained with 100% of labels. Full comparisons in Table 3.

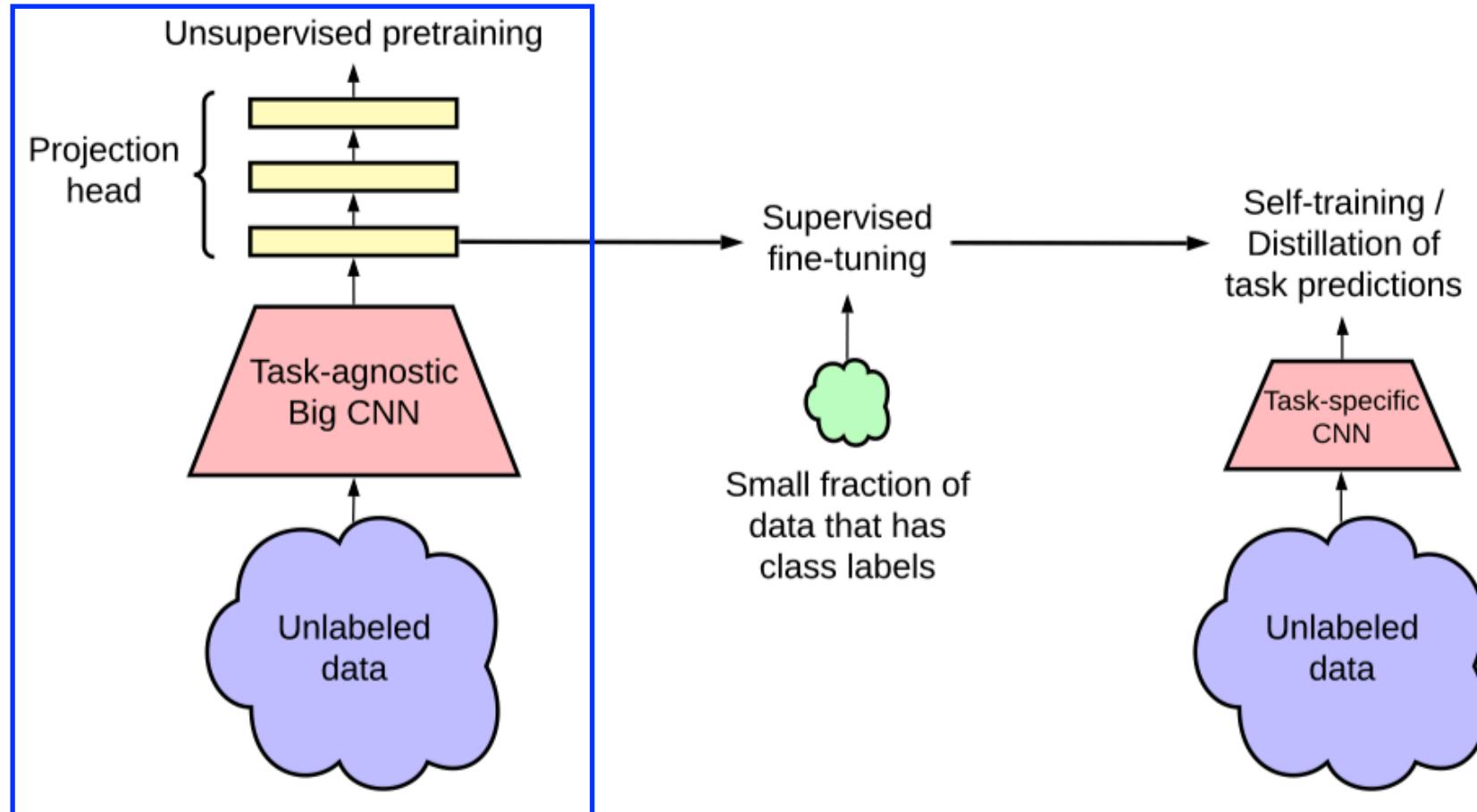
SimCLRv2 even surpasses **supervised learning results** on ImageNet!

# Key Findings



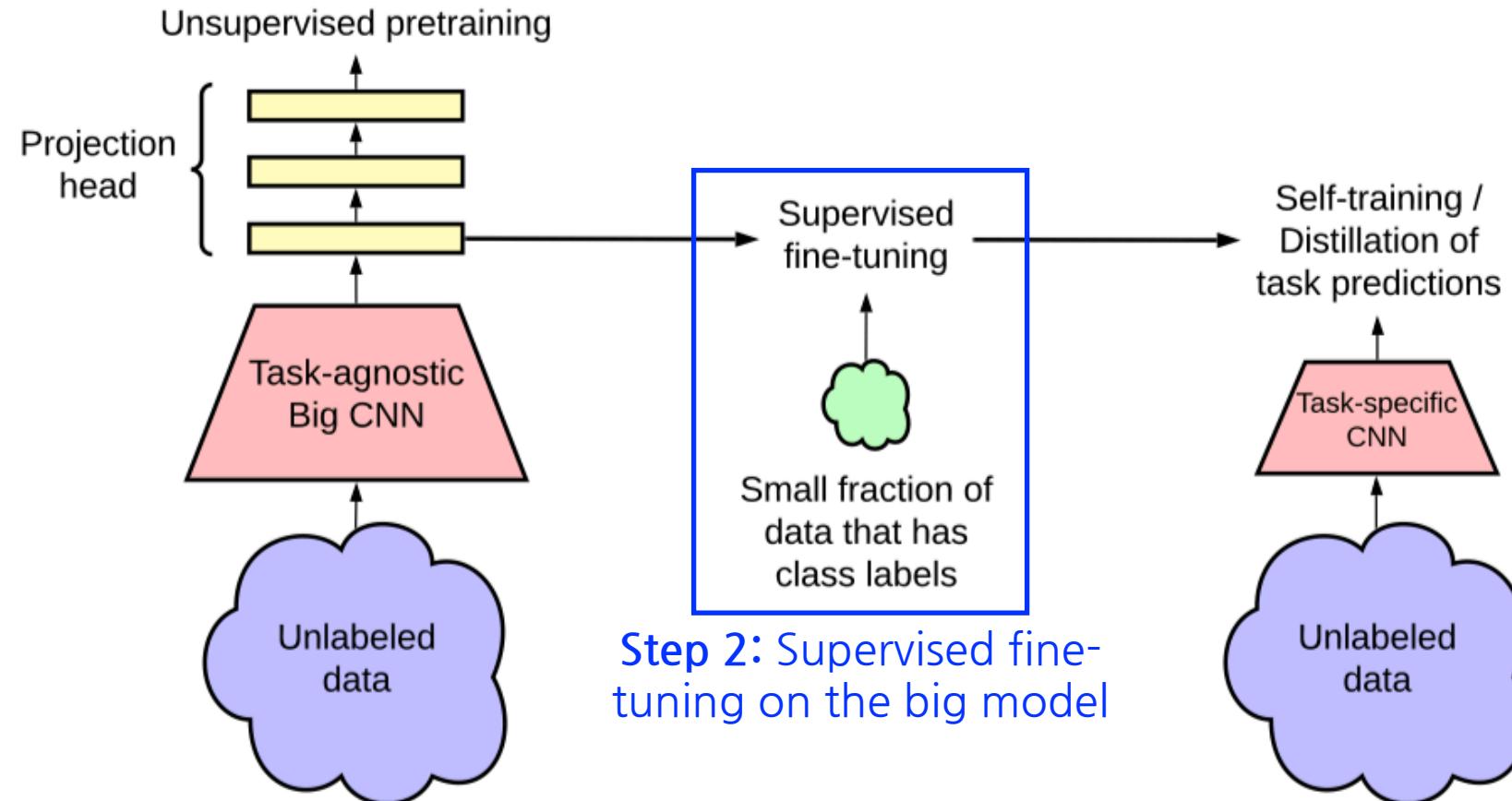
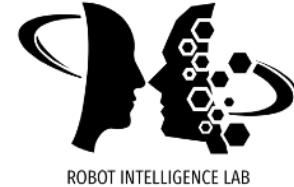
- Bigger self-supervised models are more **label efficient**, performing significantly better when fine-tuned on only a few labeled examples, even though they have more capacity to potentially overfit.
- Train a small model for a target task from the big model with unsupervised data (**distillation**) works well.
- **Deeper projection head** (compared to 1-hidden layer of SimCLR) improves the semi-supervised performance.

# SimCLRv2

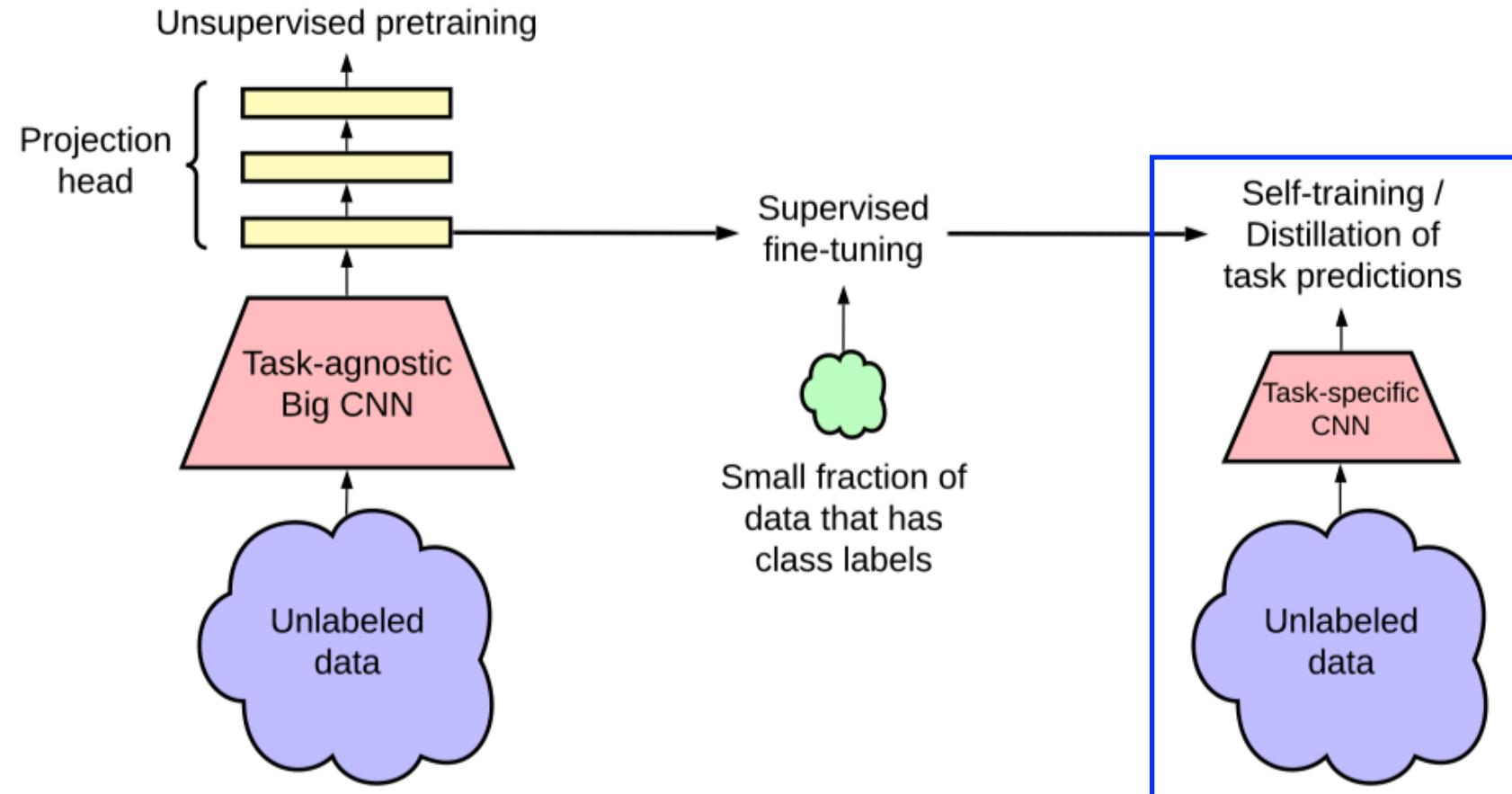


Step 1: SimCLR with a bigger model  
and deeper projection head

# SimCLRv2

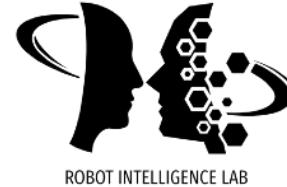


# SimCLRv2



Step 3: Distillation to a small model with unlabeled data

# Improvements over SimCLR



- Larger models for SimCLR (e.g., 152-layer ResNet with  $3 \times$  wider channels) obtains 29% relative improvement (in ImageNet top-1 accuracy when fine-tuned on 1% of labeled images).
- Instead of 2-layer projection head of SimCLR, SimCLRV2 uses a 3-layer projection head and fine-tuning from the 1st layer of the projection head, it achieves 14% relative improvements.
- Incorporating the memory mechanism from MoCo (buffering negative examples) yields an improvement of 1% for linear probing.
- Distillation works well.

# Knowledge Distillation

- Fine-tuned network (a big model) is used as a teacher to **impute (soft) labels** for training a student network (a small model) from the unlabeled dataset  $D$

$$\mathcal{L}^{\text{distill}} = - \sum_{\mathbf{x} \in D} \left[ \sum_y P^T(y | \mathbf{x}_i; \tau) \log P^S(y | \mathbf{x}_i; \tau) \right]$$

where  $P^T(\cdot)$  and  $P^S(\cdot)$  are teacher and student models, respectively, and only the student network  $P^S(y | \mathbf{x}; \tau)$  is trained.

# Bigger Models are More Label-Efficient



Table 1: Top-1 accuracy of fine-tuning SimCLRv2 models (on varied label fractions) or training a linear classifier on the representations. The supervised baselines are trained from scratch using all labels in 90 epochs. The parameter count only include ResNet up to final average pooling layer. For fine-tuning results with 1% and 10% labeled examples, the models include additional non-linear projection layers, which incurs additional parameter count (4M for 1× models, and 17M for 2× models). See Table H.1 for Top-5 accuracy.

Depth	Width	Use SK [28]	Param (M)	Fine-tuned on			Linear eval	Supervised
				1%	10%	100%		
50	1×	False	<b>24</b>	<b>57.9</b>	<b>68.4</b>	<b>76.3</b>	<b>71.7</b>	<b>76.6</b>
		True	35	64.5	72.1	78.7	74.6	78.5
	2×	False	94	66.3	73.9	79.1	75.6	77.8
		True	140	70.6	77.0	81.3	77.7	79.3
101	1×	False	43	62.1	71.4	78.2	73.6	78.0
		True	65	68.3	75.1	80.6	76.3	79.6
	2×	False	170	69.1	75.8	80.7	77.0	78.9
		True	257	73.2	78.8	82.4	79.0	80.1
152	1×	False	58	64.0	73.0	79.3	74.5	78.3
		True	89	70.0	76.5	81.3	77.2	79.9
	2×	False	233	70.2	76.6	81.1	77.4	79.1
		True	354	74.2	79.4	82.9	79.4	80.4
152	3×	True	<b>795</b>	<b>74.9</b>	<b>80.1</b>	<b>83.1</b>	<b>79.8</b>	<b>80.5</b>

# Bigger Models are More Label-Efficient

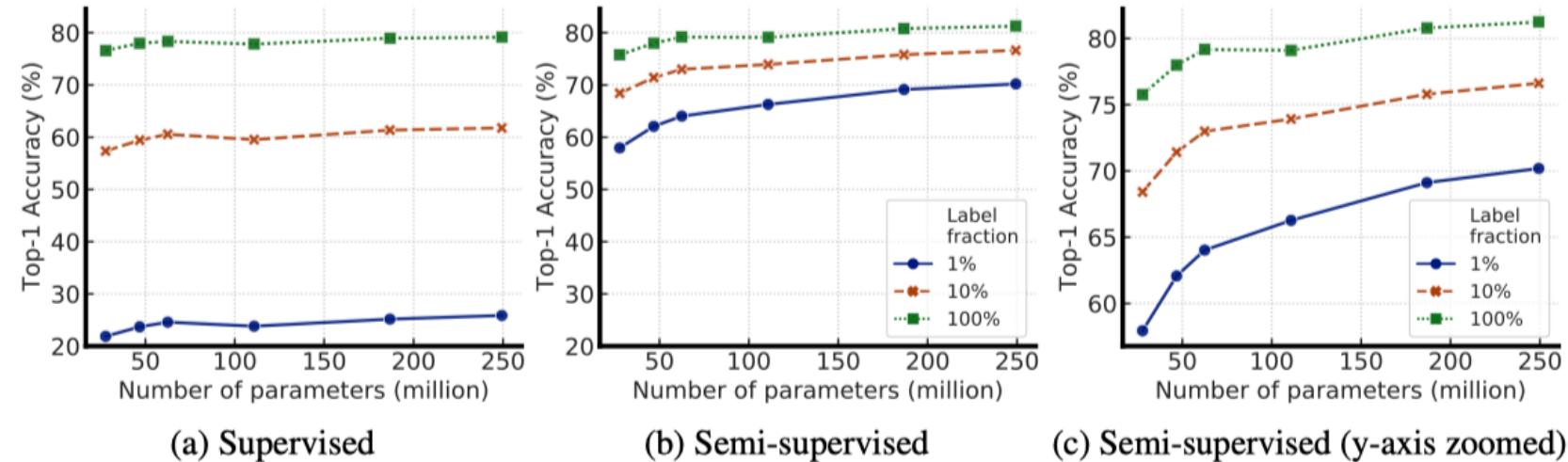
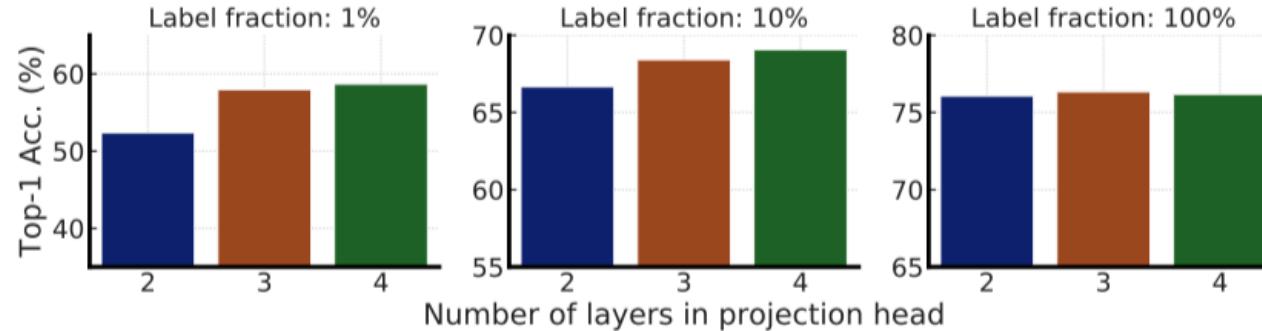
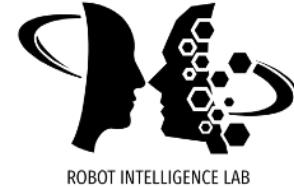
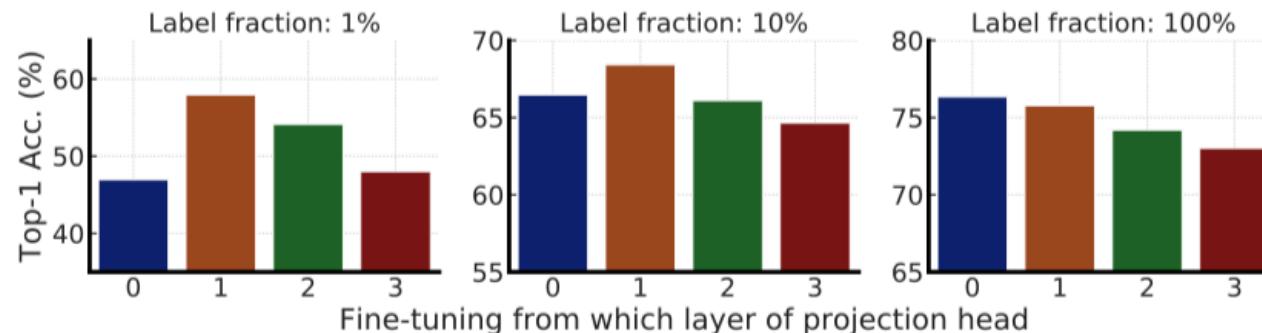


Figure 4: Top-1 accuracy for supervised vs semi-supervised (SimCLRV2 fine-tuned) models of varied sizes on different label fractions. ResNets with depths of 50, 101, 152, width multiplier of 1 $\times$ , 2 $\times$  (w/o SK) are presented here. For supervised models on 1%/10% labels, AutoAugment [35] and label smoothing [36] are used. Increasing the size of SimCLRV2 models by 10 $\times$ , from ResNet-50 to ResNet-152 (2 $\times$ ), improves label efficiency by 10 $\times$ .

# Projection Heads



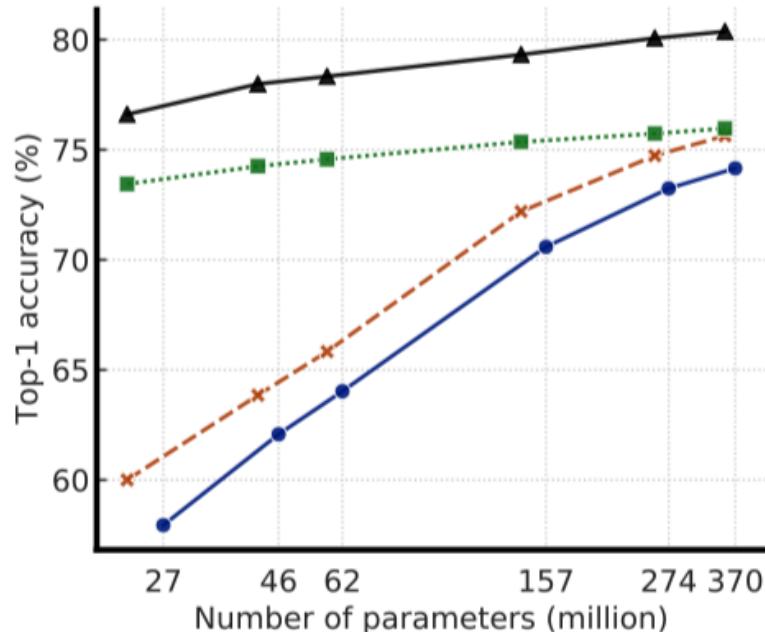
(a) Effect of projection head's depth when fine-tuning from optimal middle layer.



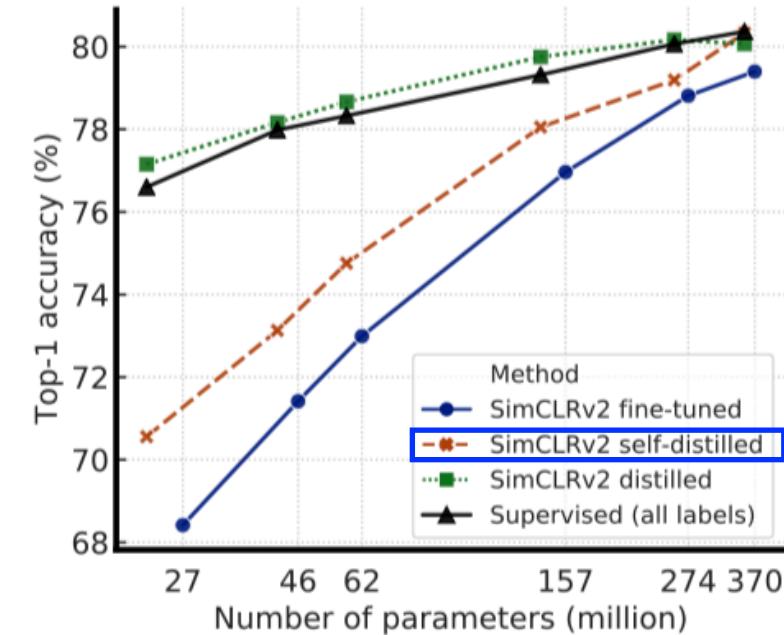
(b) Effect of fine-tuning from middle of a 3-layer projection head (0 is SimCLR).

Figure 5: Top-1 accuracy via fine-tuning under different projection head settings and label fractions (using ResNet-50).

# Distillation



(a) Label fraction 1%



(b) Label fraction 10%

Teacher and student has the same architecture!

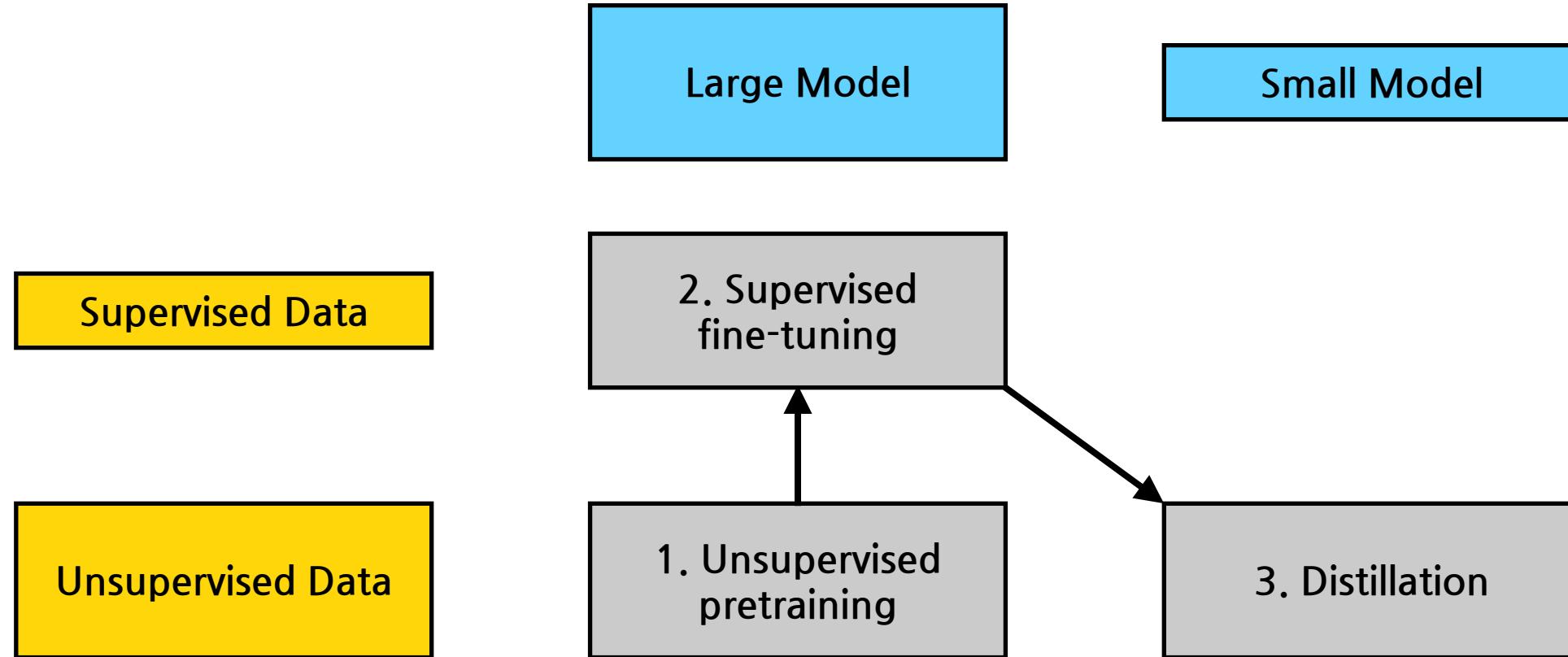
Figure 6: Top-1 accuracy of distilled SimCLRv2 models compared to the fine-tuned models as well as supervised learning with all labels. The self-distilled student has the same ResNet as the teacher (without MLP projection head). The distilled student is trained using the self-distilled ResNet-152 ( $2\times+SK$ ) model, which is the largest model included in this figure.

# ImageNet Accuracy

Table 3: ImageNet accuracy of models trained under semi-supervised settings. For our methods, we report results with distillation after fine-tuning. For our smaller models, we use self-distilled ResNet-152 ( $3\times+SK$ ) as the teacher.

Method	Architecture	Top-1		Top-5	
		Label fraction 1%	Label fraction 10%	Label fraction 1%	Label fraction 10%
Supervised baseline [30]	ResNet-50	25.4	56.4	48.4	80.4
<i>Methods using unlabeled data in a task-specific way:</i>					
Pseudo-label [11, 30]	ResNet-50	-	-	51.6	82.4
VAT+Entropy Min. [37, 38, 30]	ResNet-50	-	-	47.0	83.4
Mean teacher [39]	ResNeXt-152	-	-	-	90.9
UDA (w. RandAug) [14]	ResNet-50	-	68.8	-	88.5
FixMatch (w. RandAug) [15]	ResNet-50	-	71.5	-	89.1
S4L (Rot+VAT+Entropy Min.) [30]	ResNet-50 (4 $\times$ )	-	73.2	-	91.2
MPL (w. RandAug) [2]	ResNet-50	-	73.8	-	-
CowMix [40]	ResNet-152	-	73.9	-	91.2
<i>Methods using unlabeled data in a task-agnostic way:</i>					
InstDisc [17]	ResNet-50	-	-	39.2	77.4
BigBiGAN [41]	RevNet-50 (4 $\times$ )	-	-	55.2	78.8
PIRL [42]	ResNet-50	-	-	57.2	83.8
CPC v2 [19]	ResNet-161(*)	52.7	73.1	77.9	91.2
SimCLR [1]	ResNet-50	48.3	65.6	75.5	87.8
SimCLR [1]	ResNet-50 (2 $\times$ )	58.5	71.7	83.0	91.2
SimCLR [1]	ResNet-50 (4 $\times$ )	63.0	74.4	85.8	92.6
BYOL [43] (concurrent work)	ResNet-50	53.2	68.8	78.4	89.0
BYOL [43] (concurrent work)	ResNet-200 (2 $\times$ )	71.2	77.7	89.5	93.7
<i>Methods using unlabeled data in both ways:</i>					
SimCLRV2 distilled (ours)	ResNet-50	73.9	77.5	91.5	93.4
SimCLRV2 distilled (ours)	ResNet-50 (2 $\times$ +SK)	75.9	80.2	93.0	95.0
SimCLRV2 self-distilled (ours)	ResNet-152 (3 $\times$ +SK)	<b>76.6</b>	<b>80.9</b>	<b>93.4</b>	<b>95.5</b>

# SimCLRv2

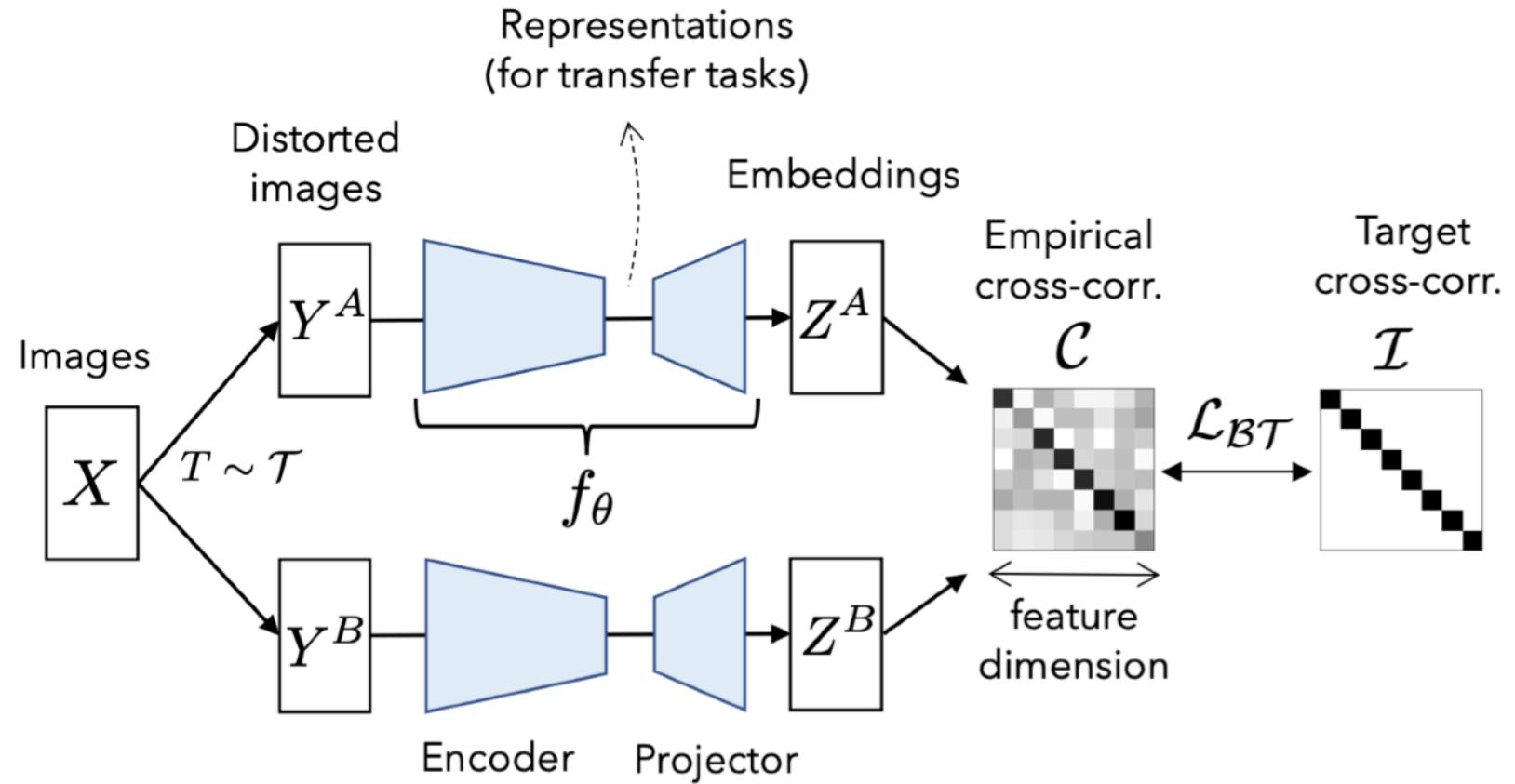




# Barlow Twins

"Barlow Twins: Self-Supervised Learning via Redundancy Reduction," 2021

# Barlow Twins



# Thank You



ROBOT INTELLIGENCE LAB