ROBOT INTELLIGENCE LAB

Self-Supervised Learning

Computer Vision

Sungjoon Choi, Korea University

kakaobrain

Content

ROBOT INTELLIGENCE LAB

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

ROBOT INTELLIGENCE LAB

DrLIM

"Dimensionality Reduction by Learning an Invariant Mapping," 2006

Contrastive Learning

ROBOT INTELLIGENCE LAB

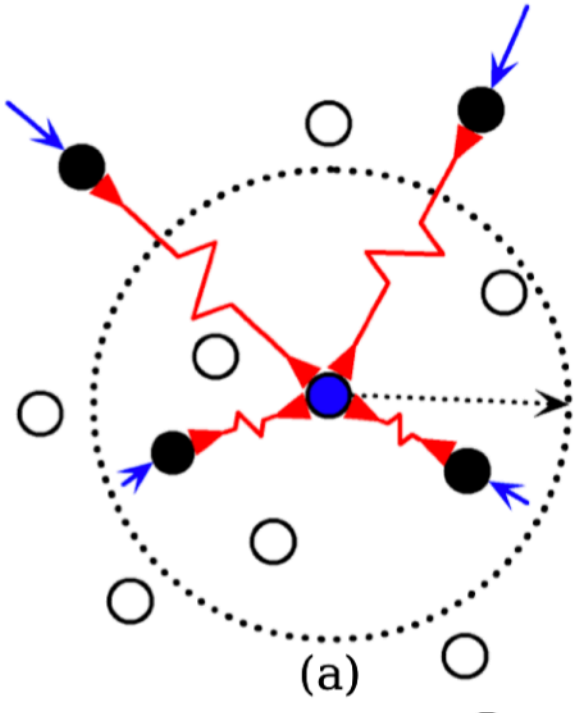
- This paper presents **DrLIM**, short for **D**imensionality **R**eduction by **L**earning and **I**nvariant **M**apping.
- But, this paper is more famous for its **Contrastive Loss** function.
- Suppose that $\mathbf{x}_1$ and $\mathbf{x}_2$ are a pair of samples and y is a binary label assigned to this pair where $y = 0$ if they are similar and $y = 1$ if the are dissimilar.

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

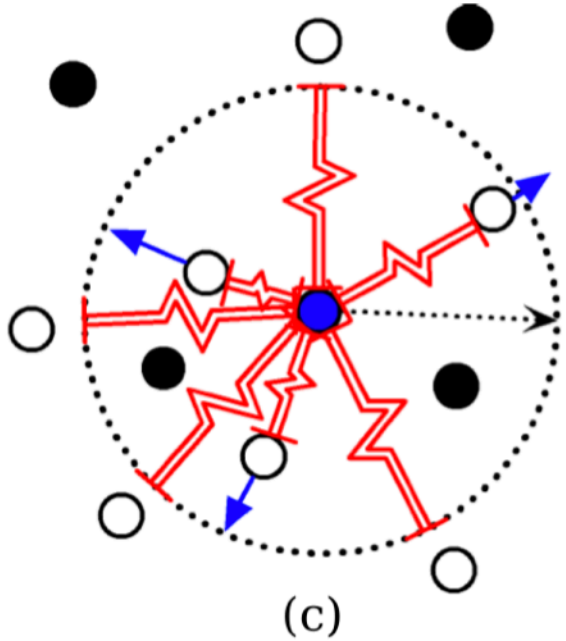
  - where $D_\theta(\mathbf{x}_1, \mathbf{x}_2) = \|f_\theta(\mathbf{x}_1) - f_\theta(\mathbf{x}_2)\|_2$

Contrastive Learning

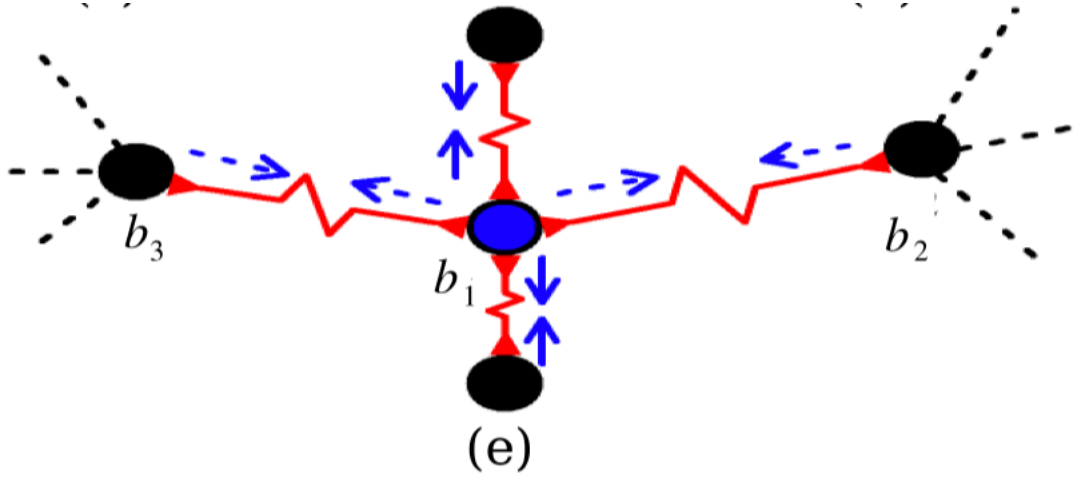
ROBOT INTELLIGENCE LAB



Attract similar pairs



Repulse dissimilar pairs
(up-to radius m)



Equilibrium

Algorithm

ROBOT INTELLIGENCE LAB

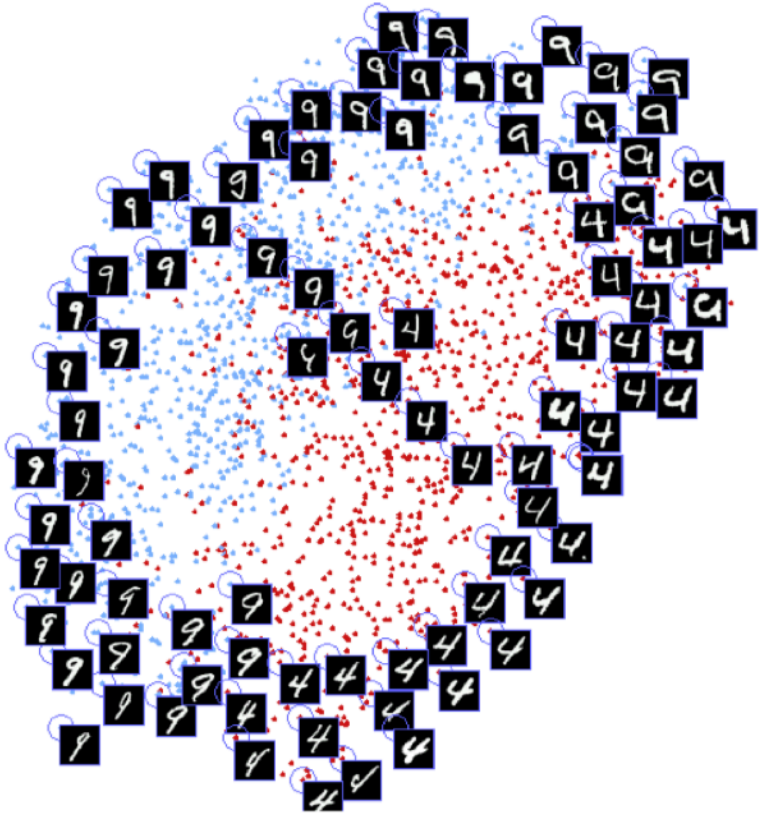
- **Step 1**: For each input sample $\mathbf{x}_i$, do the following:
  - Using prior knowledge, find the set of samples $S_{\mathbf{X}_i} = \{\mathbf{x}_j\}_{j=1}^{p}$, such that $\mathbf{x}_j$ is deemed similar to $\mathbf{x}_i$.
  - Pair the sample $\mathbf{x}_i$, with **all the other** training samples 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 θ to decrease $D_\theta = \|G_\theta(\mathbf{x}_i) - G_\theta(\mathbf{x}_j)\|_2$
    - If $y_{ij} = 1$, the update θ to increase $D_\theta = \|G_\theta(\mathbf{x}_i) - G_\theta(\mathbf{x}_j)\|_2$

Caveat

ROBOT INTELLIGENCE LAB

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.

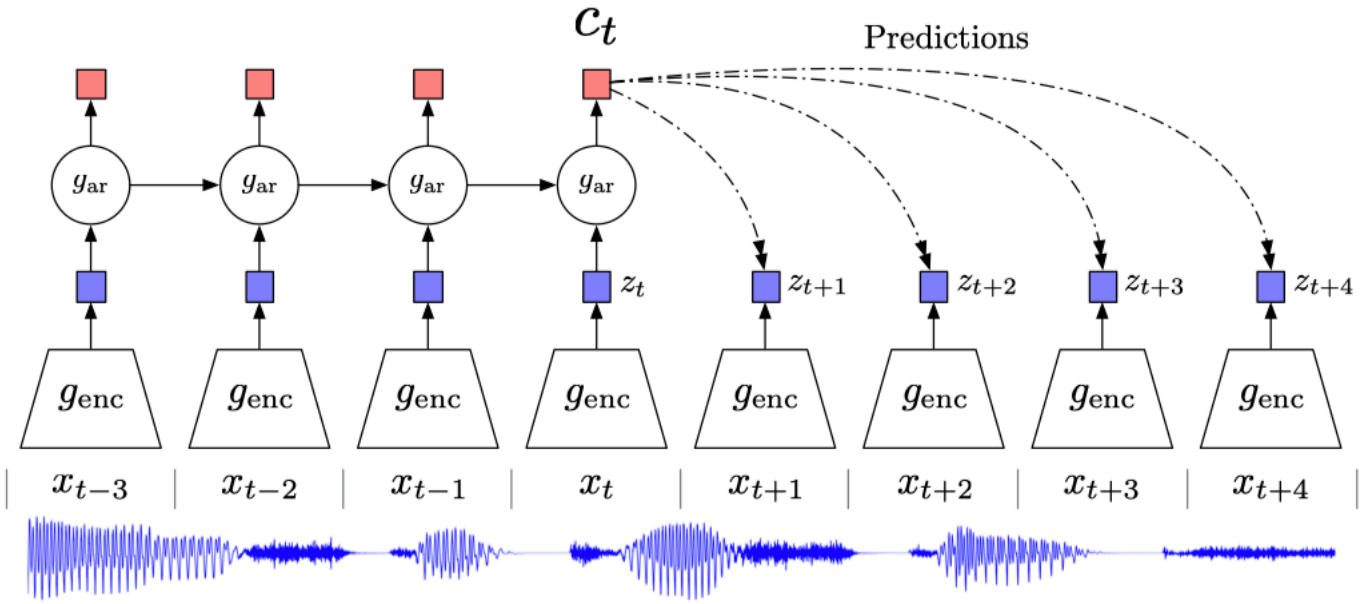
ROBOT INTELLIGENCE LAB

Contrastive Predictive Coding

"Representation Learning with Contrastive Predictive Coding," 2019

Contrastive Predictive Coding

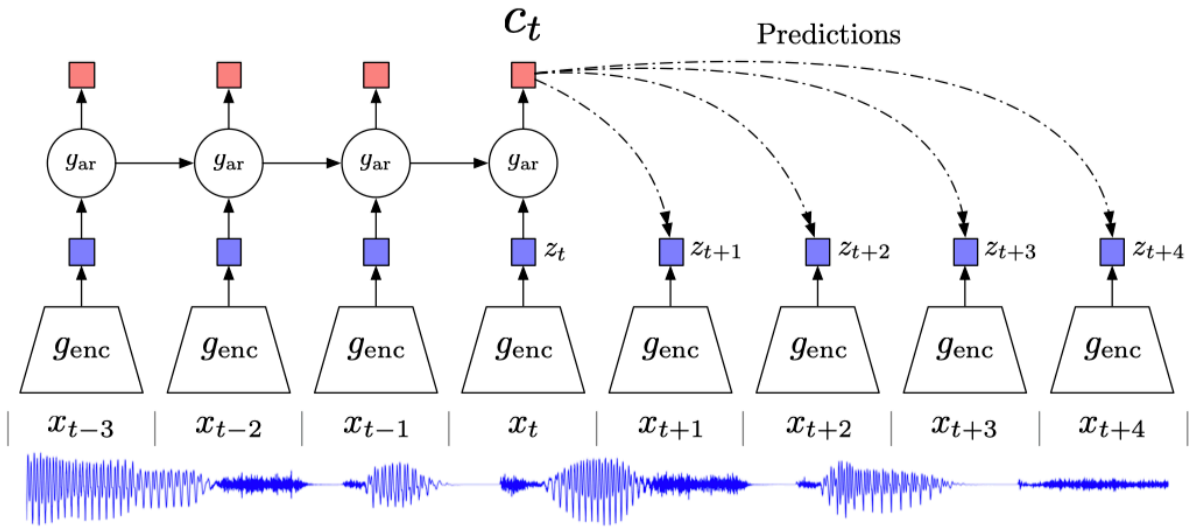
ROBOT INTELLIGENCE LAB



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

Contrastive Predictive Coding

ROBOT INTELLIGENCE LAB



- 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:

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

InfoNCE Loss

ROBOT INTELLIGENCE LAB

- How can properly maximize $f_k(x_i, c_i; \theta)$ for a positive pair (x_i, c_i)?
- How about this?

$$\hat{\theta} = \arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} f(x_i, c_i; \theta)$$

where $D = \{x_i, c_i\}_{i=1}^{N}$ is the set of positive pairs.

InfoNCE Loss

ROBOT INTELLIGENCE LAB

- How can properly maximize $f_k(x_i, c_i; \theta)$ for a positive pair (x_i, c_i)?
- How about this?

$$\hat{\theta} = \arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} f(x_i, c_i; \theta)$$

where $D = \{x_i, c_i\}_{i=1}^{N}$ is the set of positive pairs.

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

$$\mathcal{L}_N(\theta) = \mathbb{E}_X \left[\log \frac{f(x_1, c_t; \theta)}{\sum_{x_j \in X} f(x_j, c_t; \theta)} \right]$$

where $X = \{x_1, x_2, \ldots, x_N\}$ is a set of N random samples containing one positive sample x_1 and $N - 1$ negative samples.

InfoNCE Loss

ROBOT INTELLIGENCE LAB

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

ROBOT INTELLIGENCE LAB

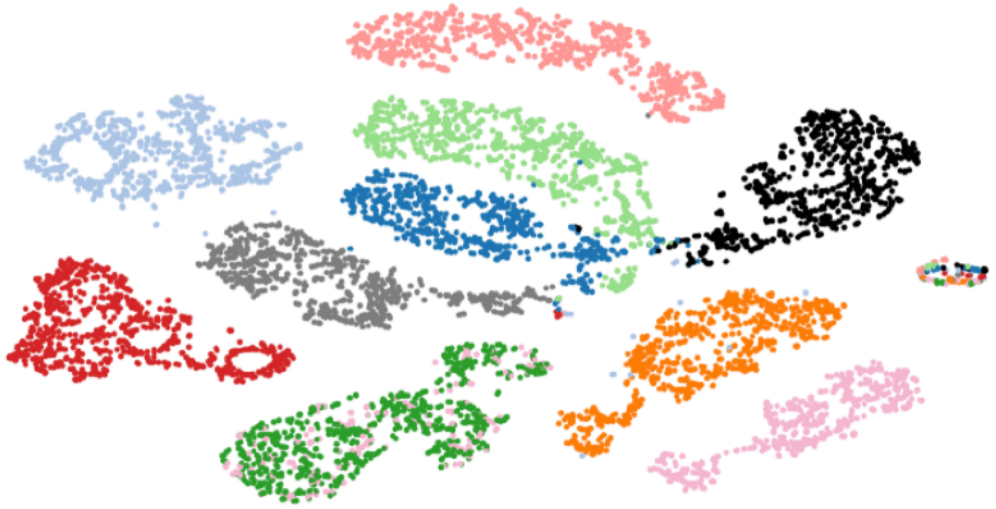


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

ROBOT INTELLIGENCE LAB

SimCLR

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

SimCLR

ROBOT INTELLIGENCE LAB

This is the famous '**SimCLR**' paper.

SimCLR

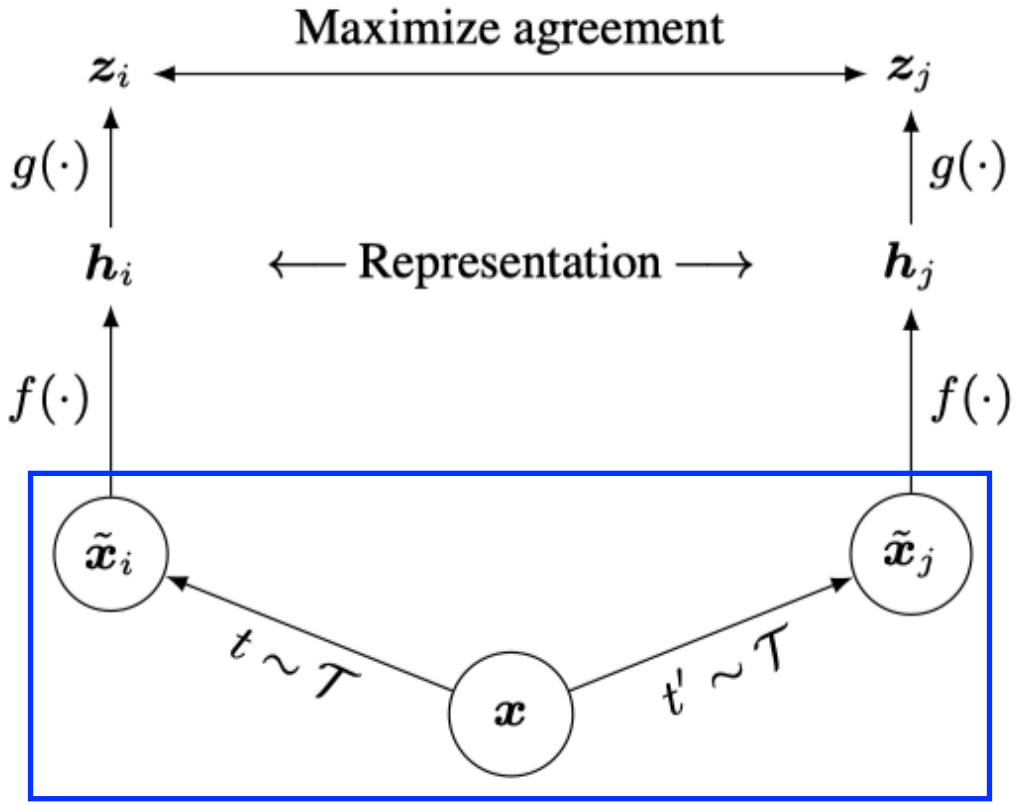
ROBOT INTELLIGENCE LAB

This paper presents a **Sim**ple framework for **C**ontrastive **L**earning of visual **R**epresentations.

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**

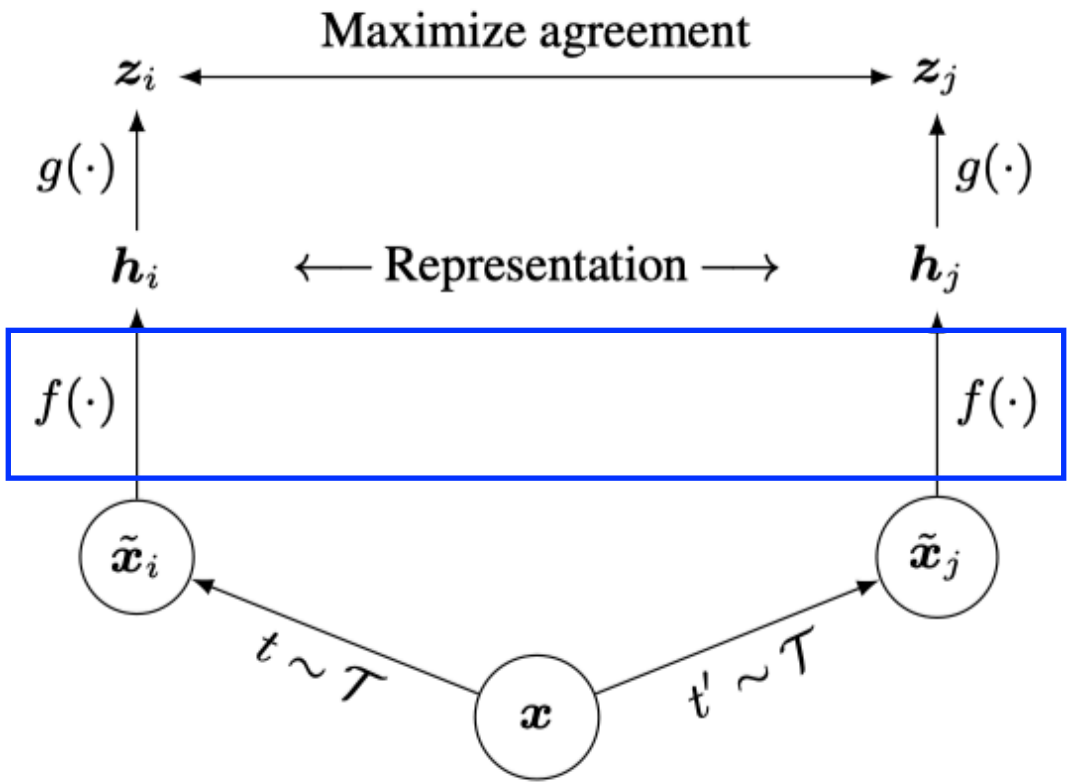
Contrastive Learning



Augmentation

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

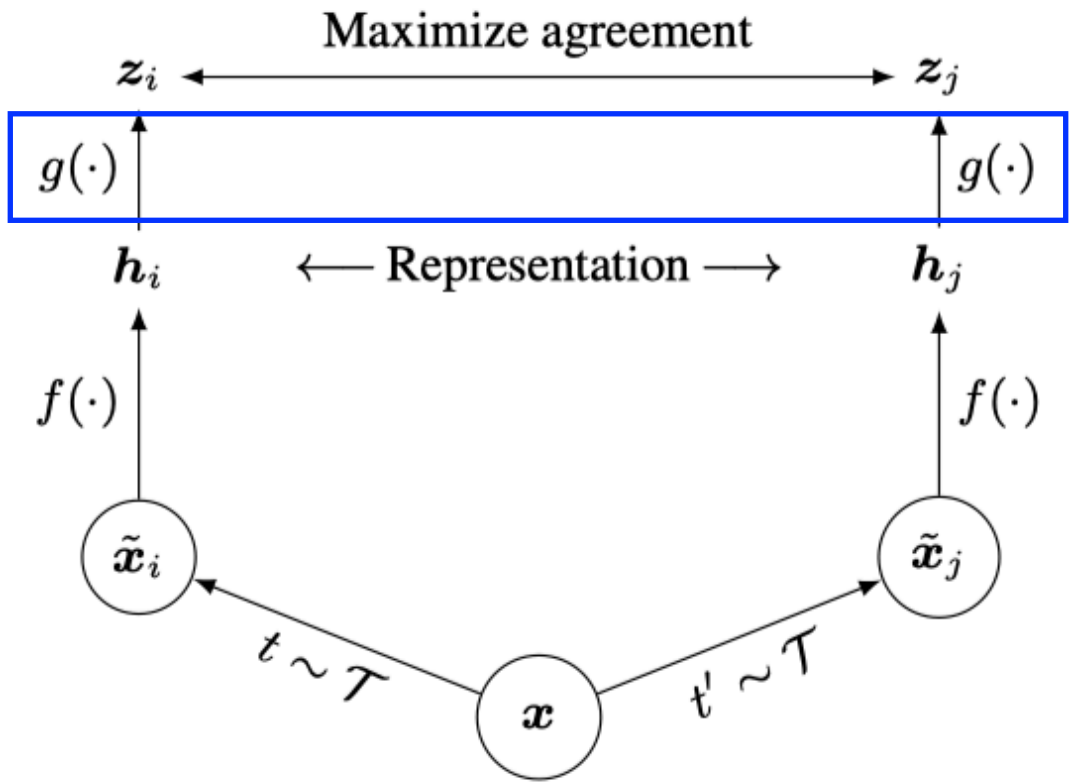
Contrastive Learning



ResNet

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

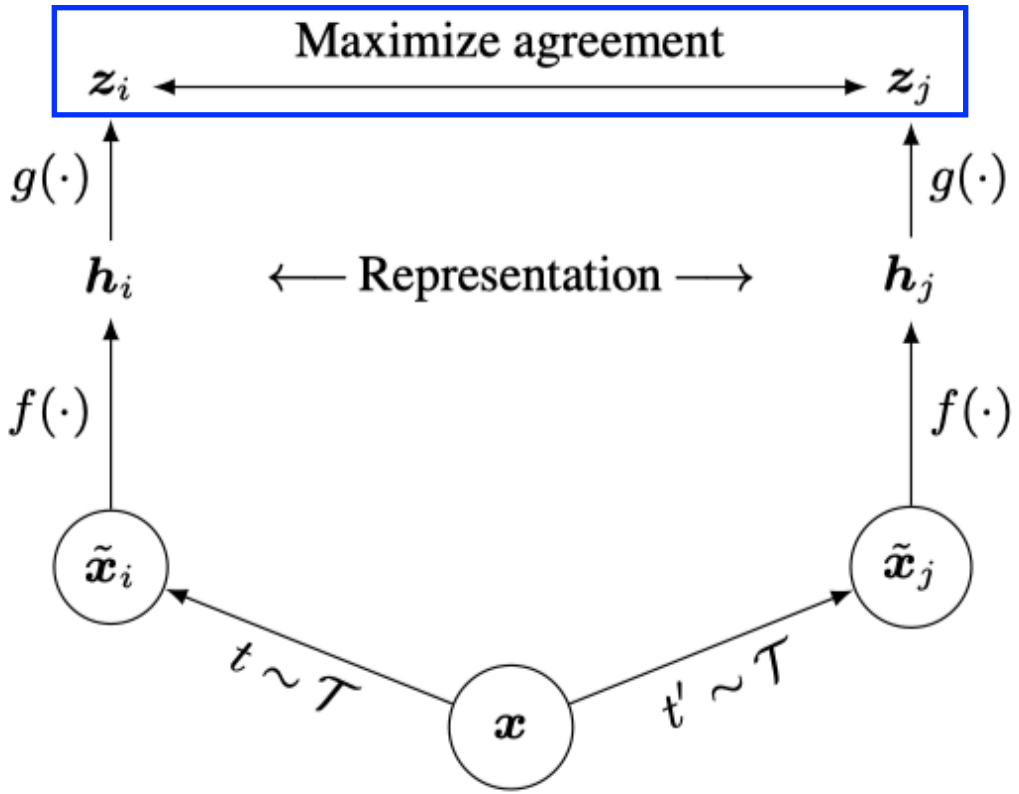
Contrastive Learning



Projection Head

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

ROBOT INTELLIGENCE LAB

- From N mini-batch 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

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

  - 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 mini-batch.
- It is called NT-Xent (the normalized temperature-scaled cross entropy loss).

SimCLR

ROBOT INTELLIGENCE LAB

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 $k \in \{1, \ldots, N\}$ **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 $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **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 $g(\cdot)$

For all images in mini-batch

SimCLR

ROBOT INTELLIGENCE LAB

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 $k \in \{1, \ldots, N\}$ **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 $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **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 $g(\cdot)$

Apply two different augmentations

SimCLR

ROBOT INTELLIGENCE LAB

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 $k \in \{1, \ldots, N\}$ **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 $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **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 $g(\cdot)$

Symmetric Contrastive loss

SimCLR

ROBOT INTELLIGENCE LAB

Algorithm 1 SimCLR's main learning algorithm.

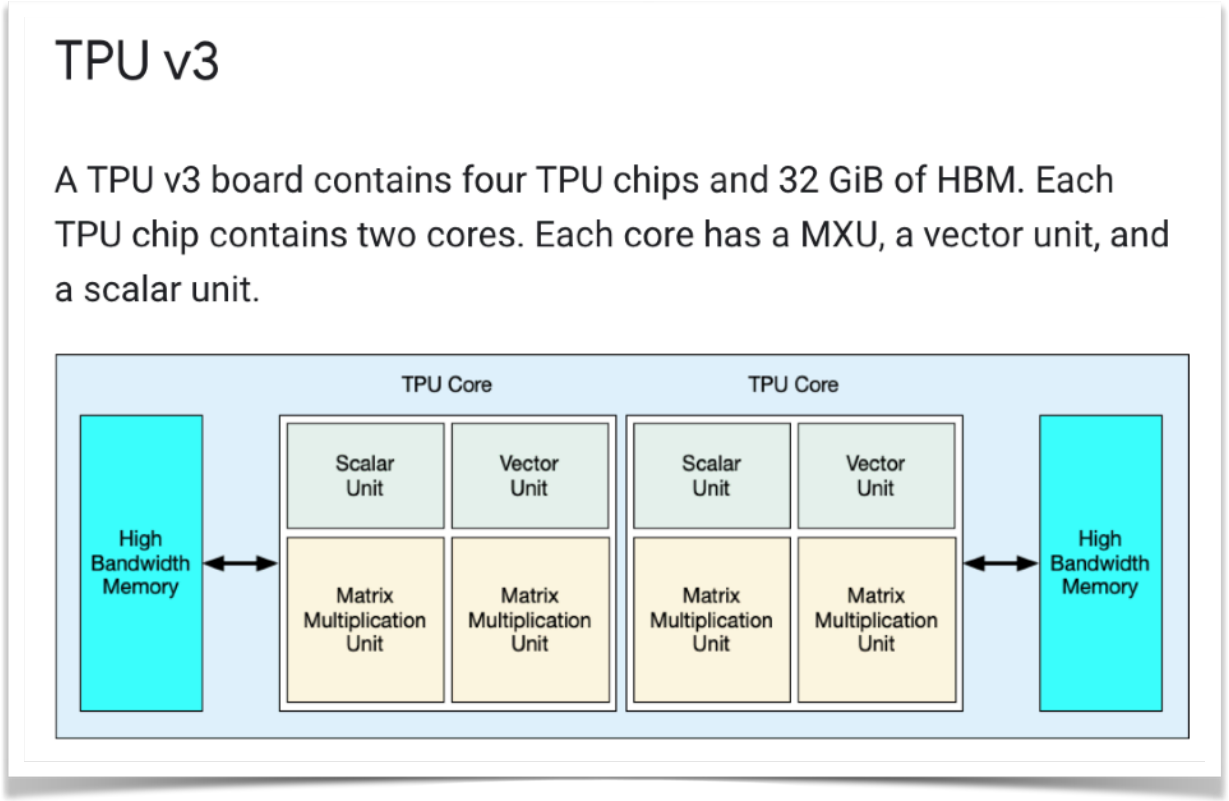
input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # 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 $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **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 $g(\cdot)$

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

Larger Batch Size

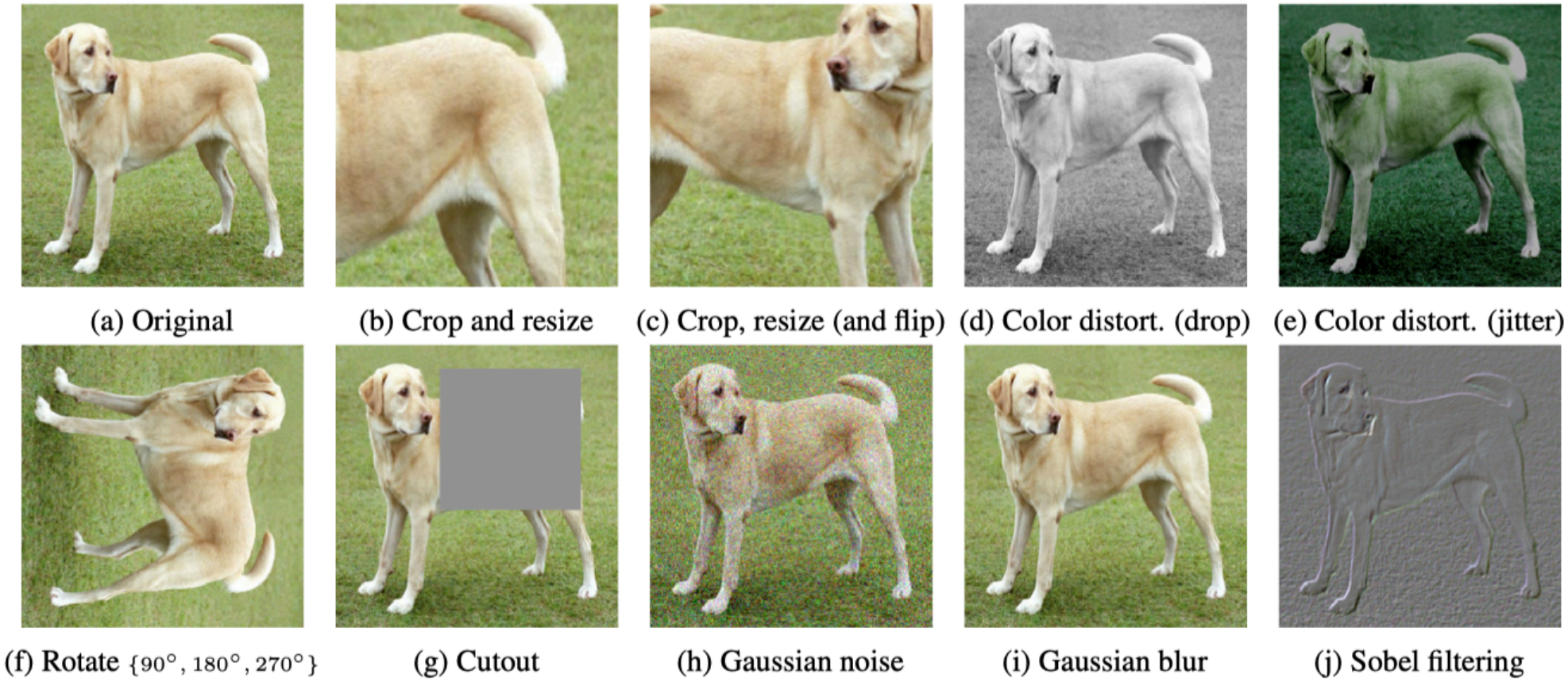
ROBOT INTELLIGENCE LAB

- 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..



https://cloud.google.com/tpu/docs/system-architecture-tpu-vm#tpu_v3

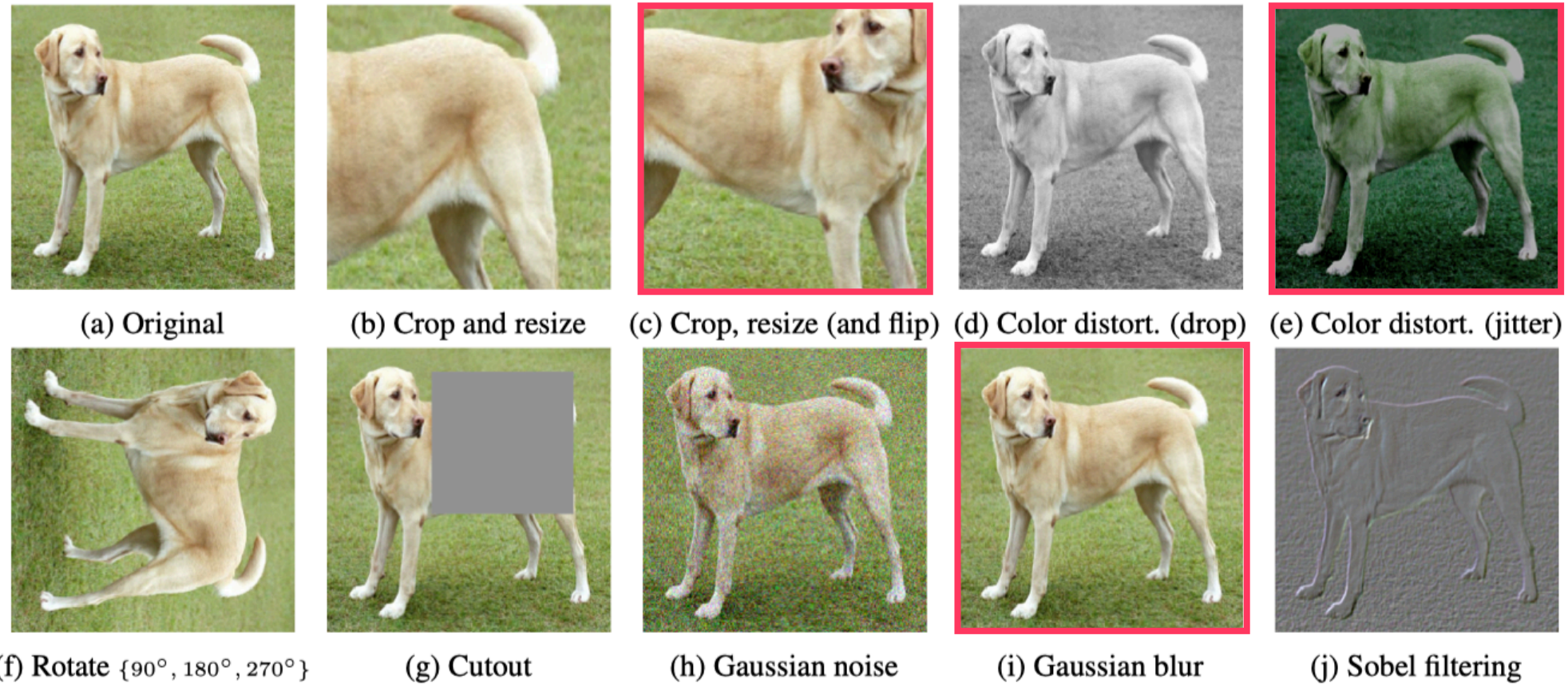
Data Augmentation



Different data (image) augmentation operators

Data Augmentation

ROBOT INTELLIGENCE LAB



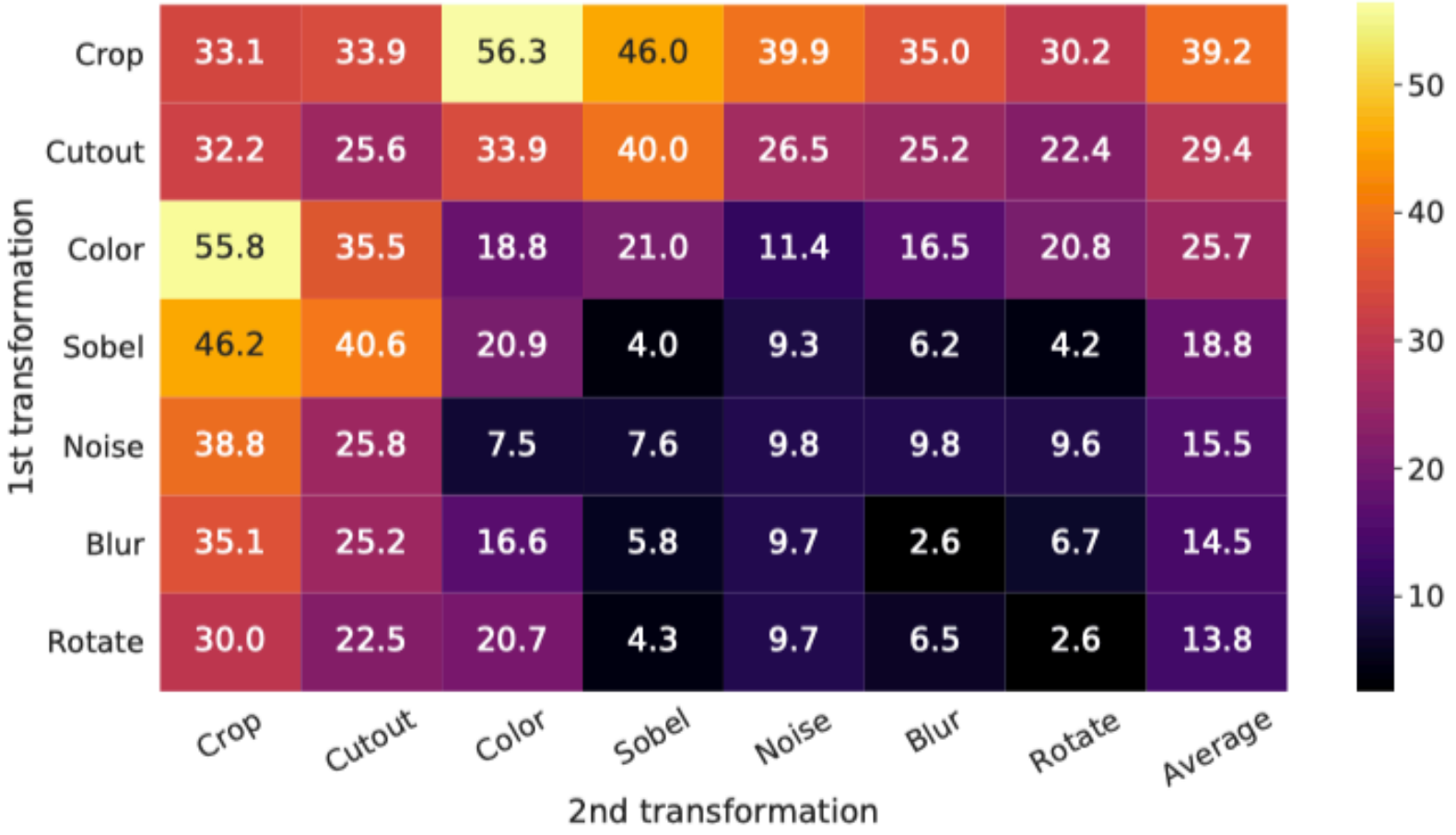
(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

The ones used in this paper

ImageNet Top-1 Accuracy

ROBOT INTELLIGENCE LAB



Effect of Projection Head

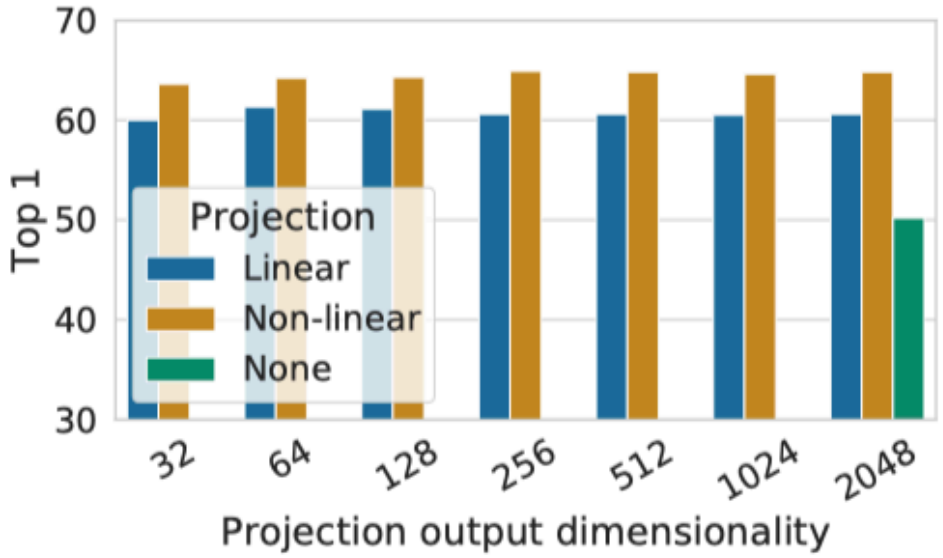


Figure 8. Linear evaluation of representations with different projection heads $g(\cdot)$ and various dimensions of $\boldsymbol{z} = g(\boldsymbol{h})$. The representation $\boldsymbol{h}$ (before projection) is 2048-dimensional here.

Effect of Batch Size

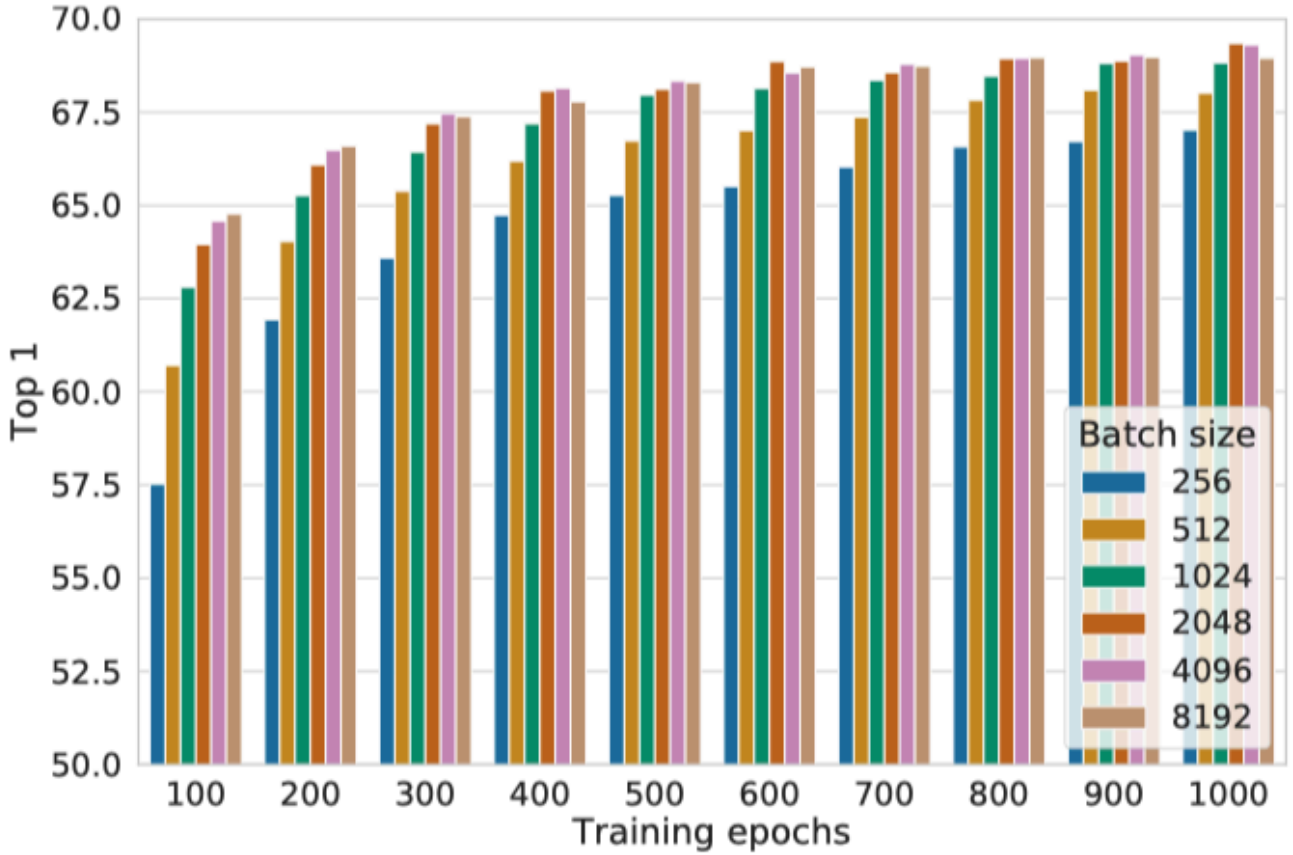


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

~~How many GPUs does Google has...~~

Linear Probing on ImageNet

ROBOT INTELLIGENCE LAB

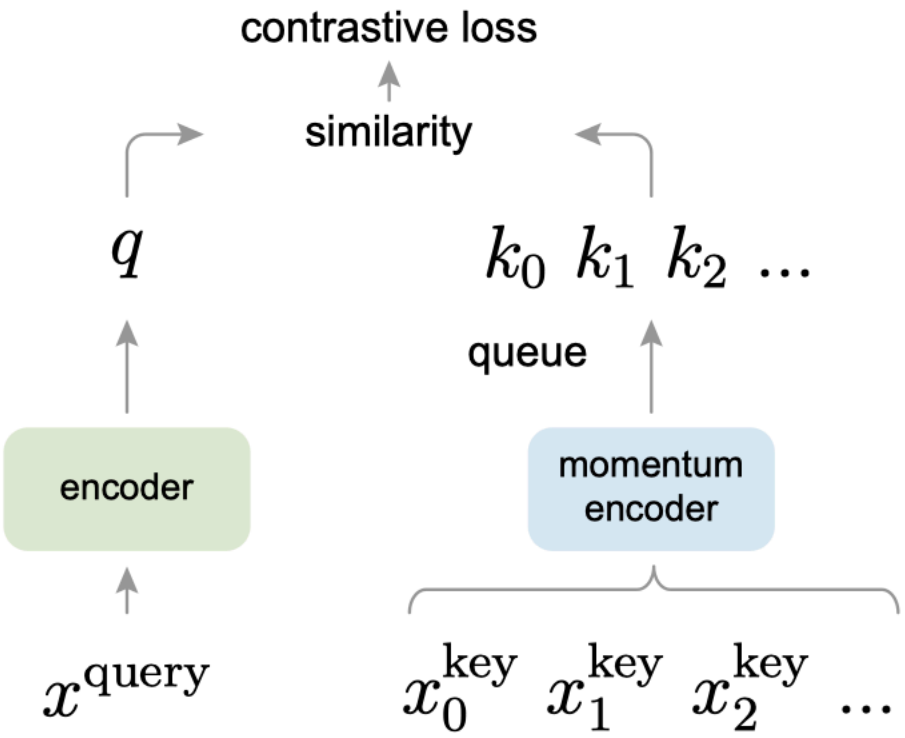
Method	Architecture	Param (M)	Top 1	Top 5
Methods using ResNet-50:				
Local Agg.	ResNet-50	24	60.2	-
MoCo	ResNet-50	24	60.6	-
PIRL	ResNet-50	24	63.6	-
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	**69.3**	**89.0**
Methods using other architectures:				
Rotation	RevNet-50 (4×)	86	55.4	-
BigBiGAN	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	**76.5**	**93.2**

ROBOT INTELLIGENCE LAB

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

ROBOT INTELLIGENCE LAB

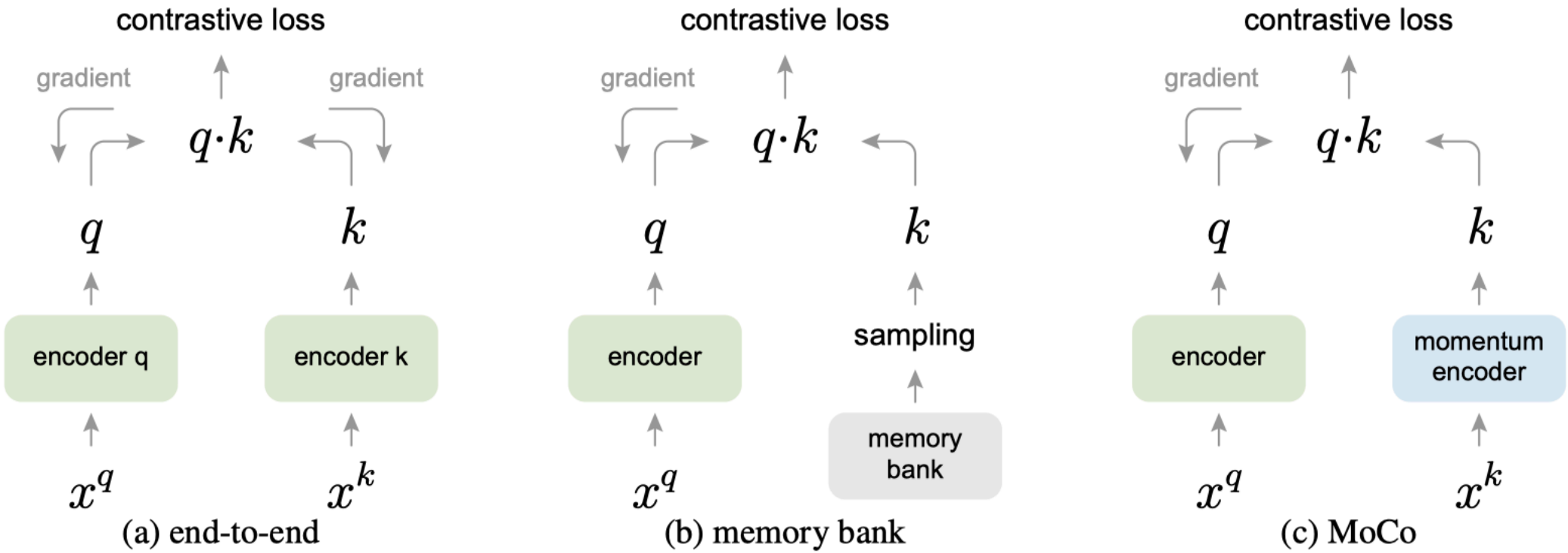
- A contrastive loss is a function whose value is low when the query q is similar to its positive key k_+ and dissimilar to all other keys (considered for q)

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

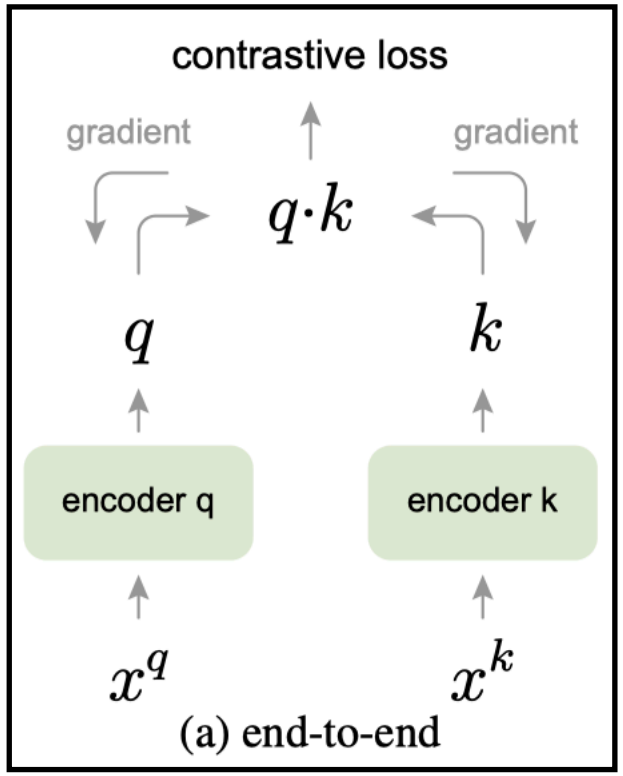
where τ is a temperature hyper-parameter. The sum is over one positive ($i = 0$) and K negative samples. Intuitively, this loss is the log loss of a $(K + 1)$-way softmax-based classifier.

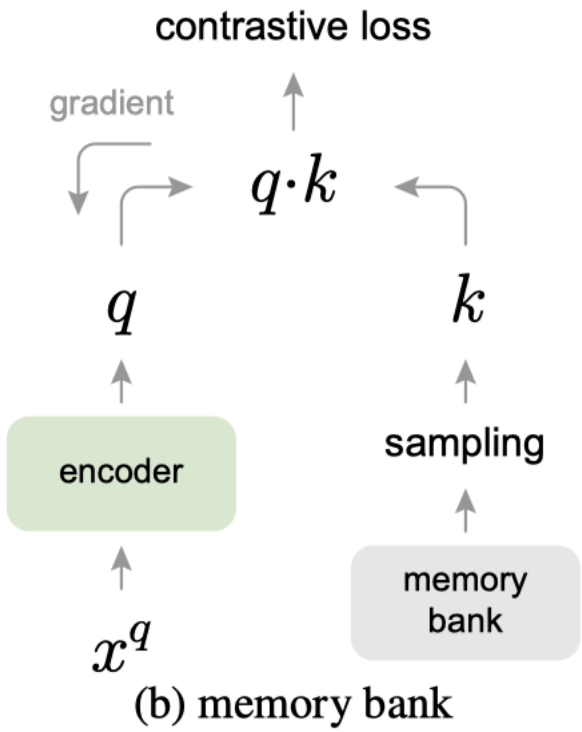
Three Contrastive Losses

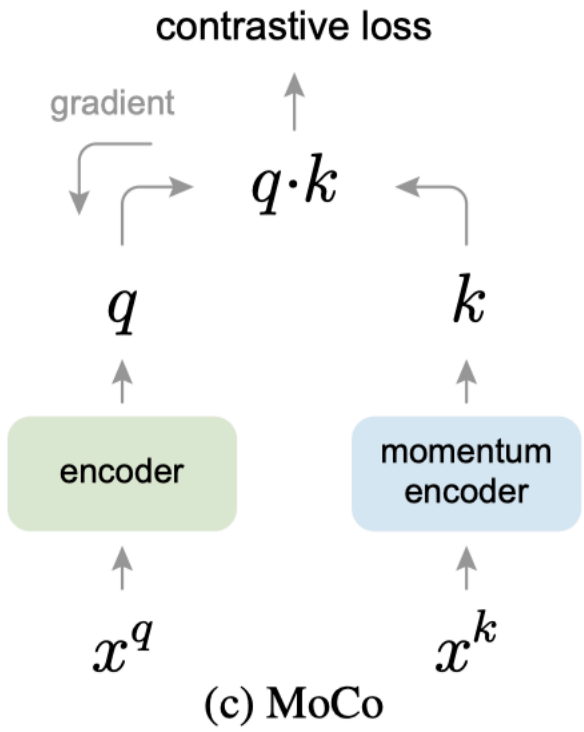
ROBOT INTELLIGENCE LAB



Three Contrastive Losses

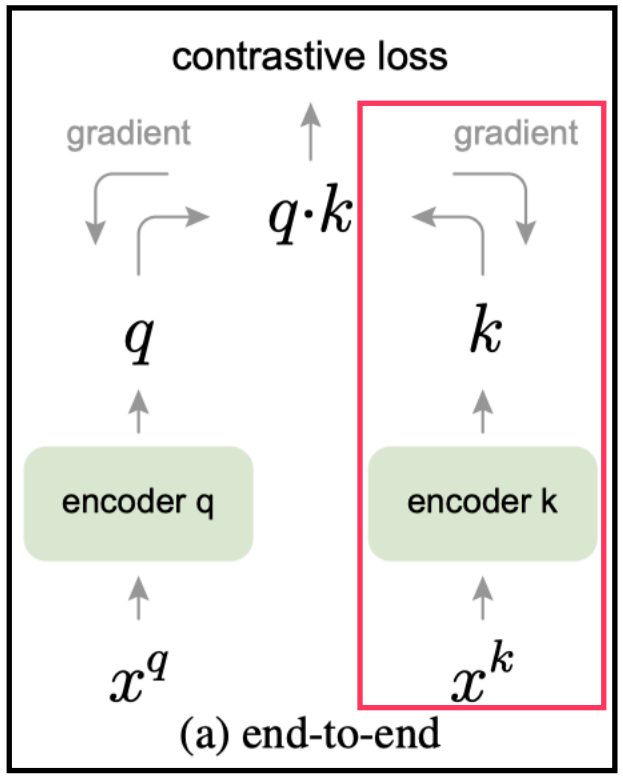


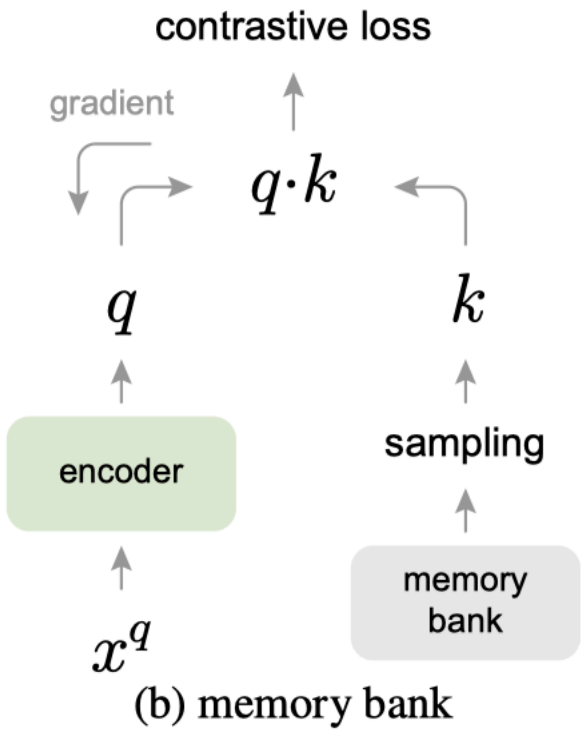


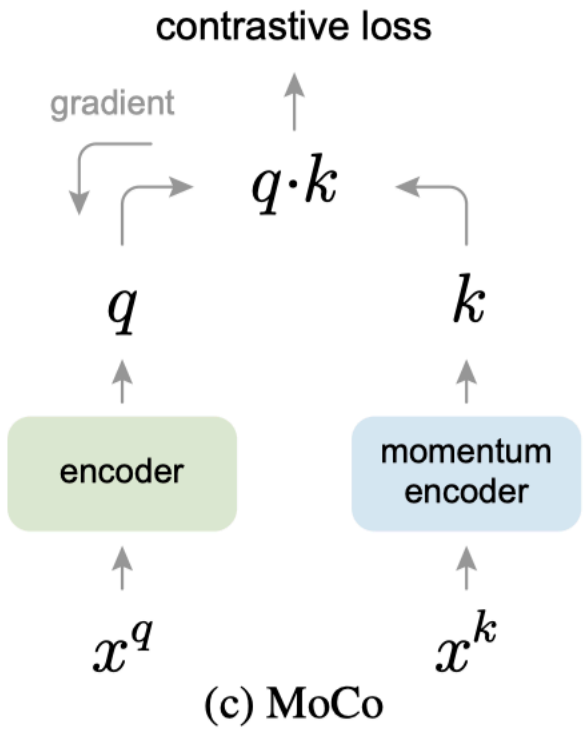


This one is similar to simCLR.

Three Contrastive Losses



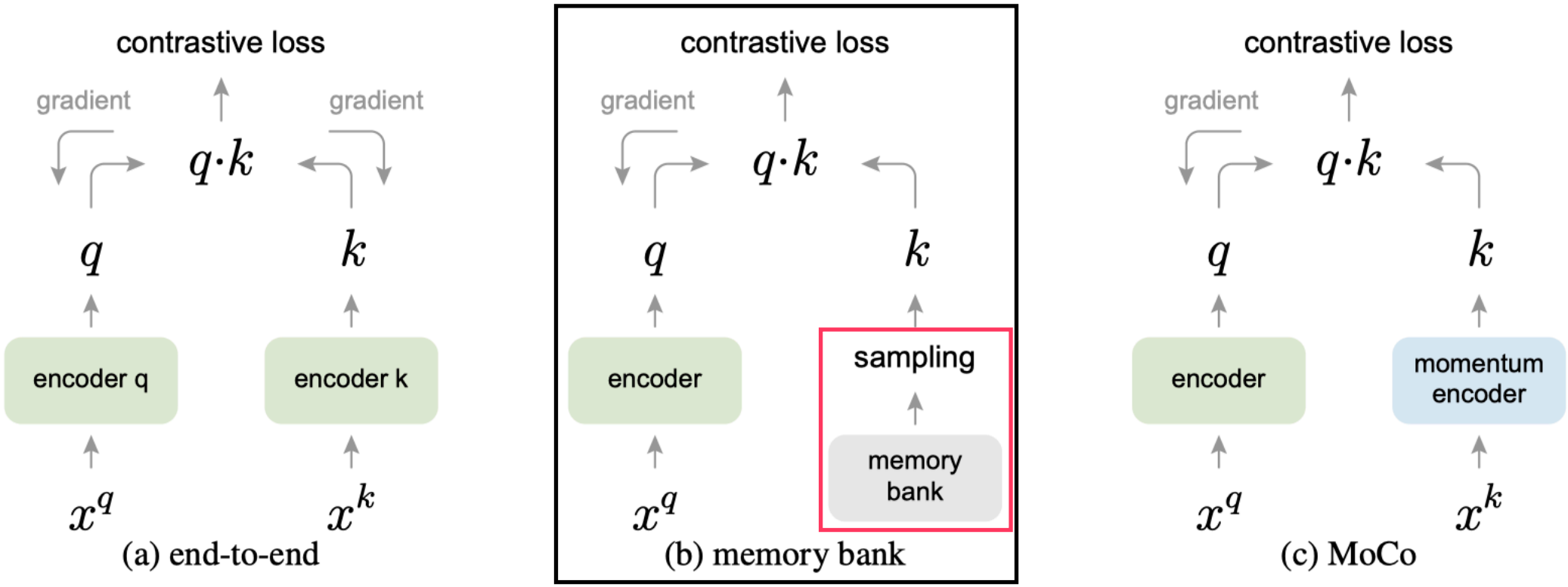




This one is similar to simCLR.

It requires **huge GPU memory**!

Three Contrastive Losses



This one requires less memory in that gradient only flows with the positive encoder.

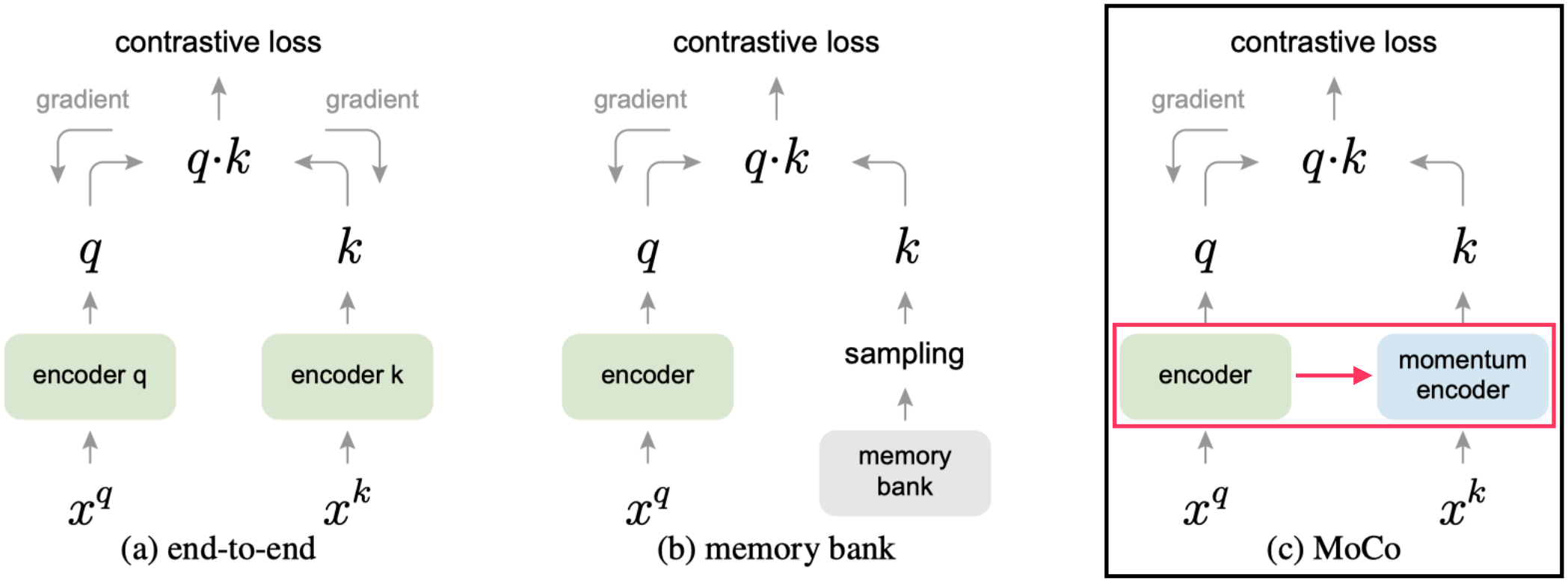
However, it requires an additional memory bank.

Good Dictionary?

ROBOT INTELLIGENCE LAB

"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

MoCo

ROBOT INTELLIGENCE LAB

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

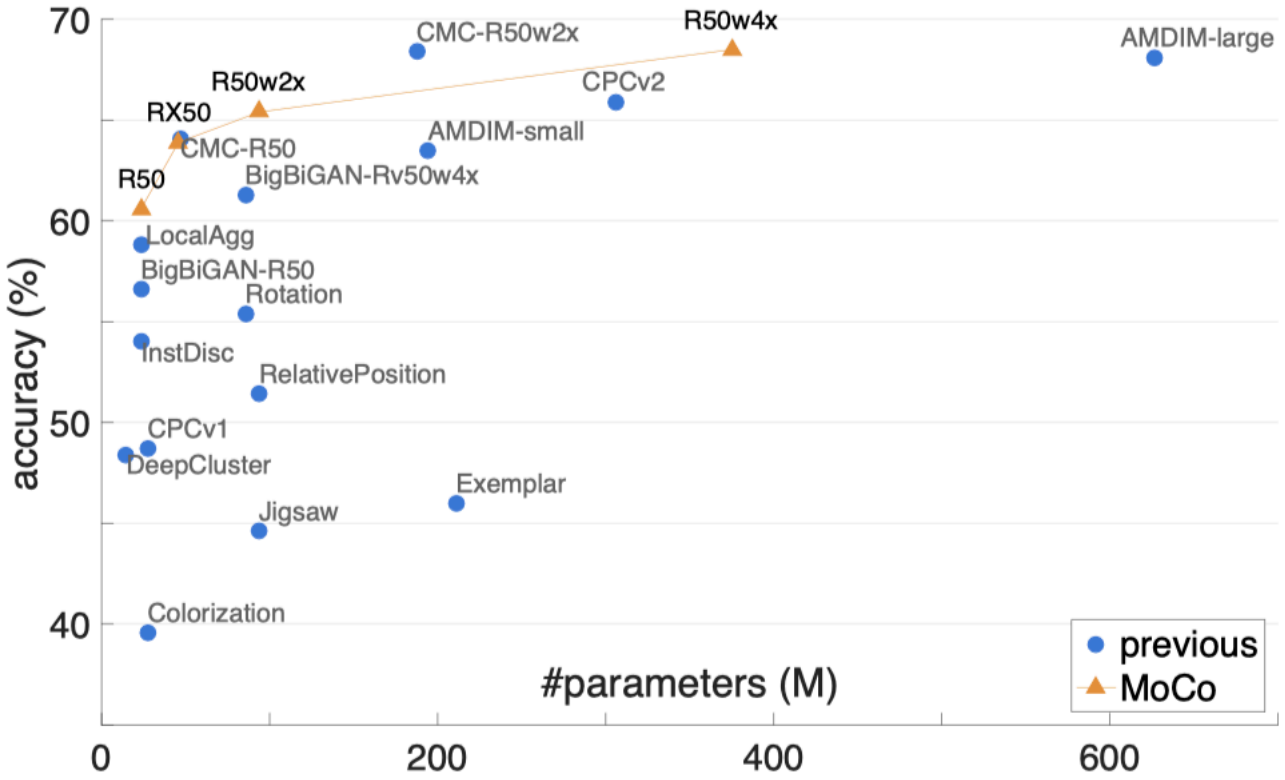
ROBOT INTELLIGENCE LAB

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 (%)	*fail*	55.2	57.8	59.0	58.9

ImageNet Linear Probing

ROBOT INTELLIGENCE LAB



SimCLR paper

Method	Architecture	Param (M)	Top 1	Top 5
Methods using ResNet-50:				
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	**69.3**	**89.0**
Methods using other architectures:				
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	**76.5**	**93.2**

ROBOT INTELLIGENCE LAB

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

ROBOT INTELLIGENCE LAB

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

BYOL

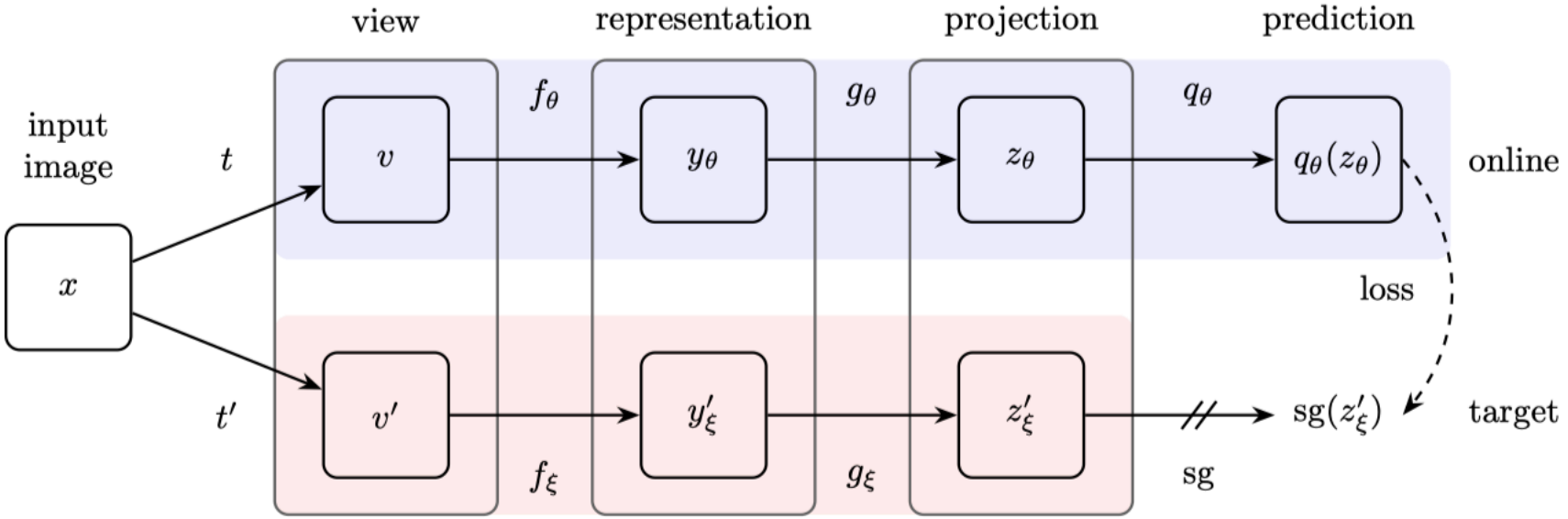


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the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

BYOL

ROBOT INTELLIGENCE LAB

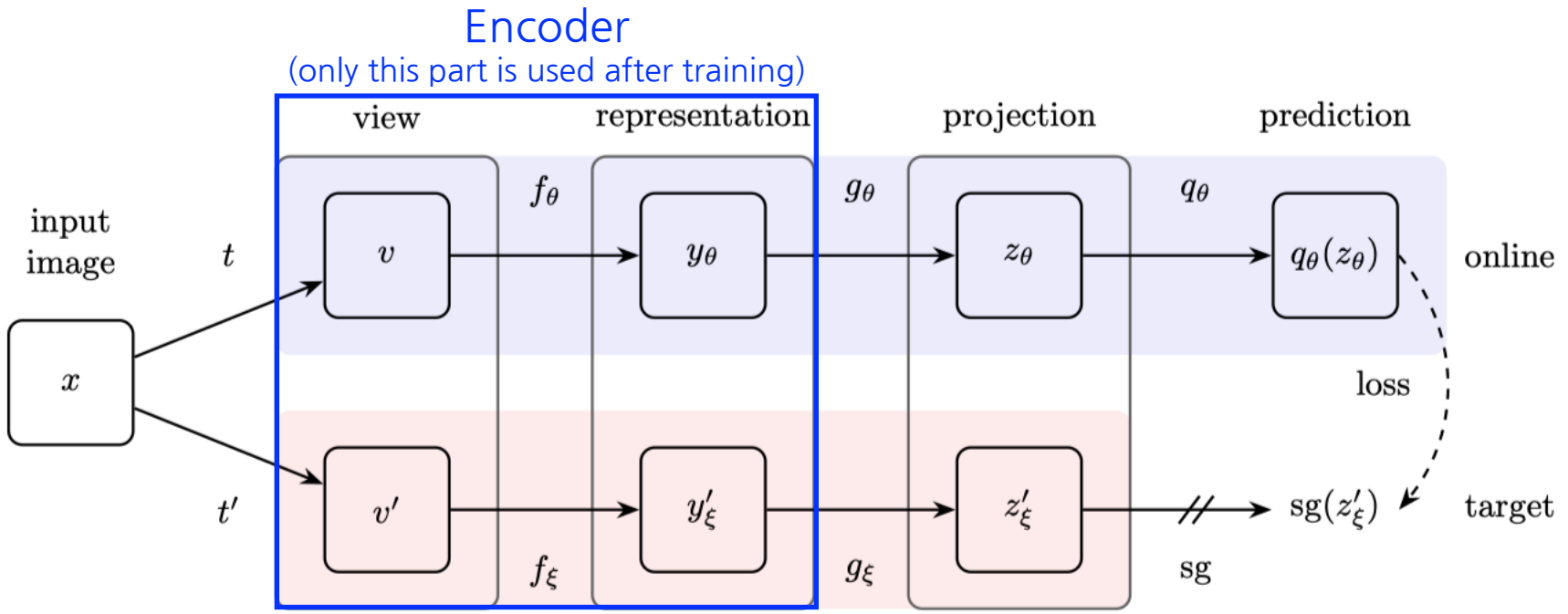


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the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

BYOL

ROBOT INTELLIGENCE LAB

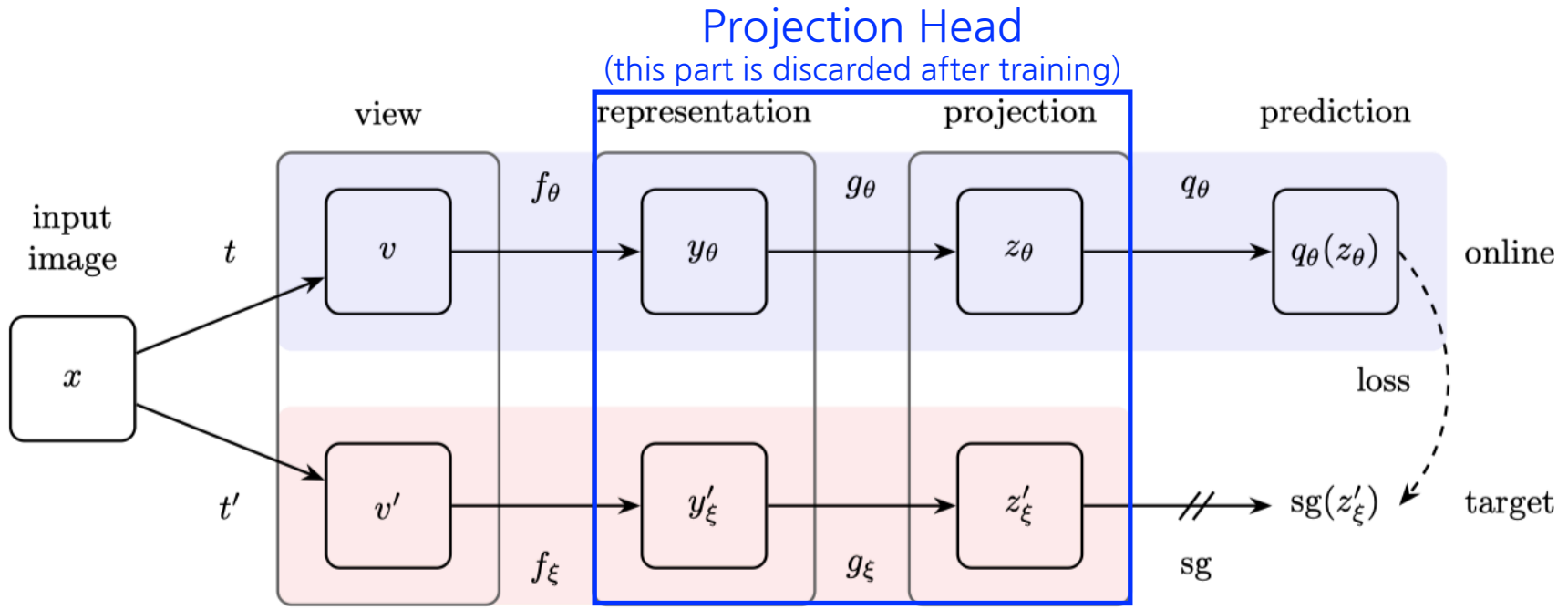


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the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

BYOL

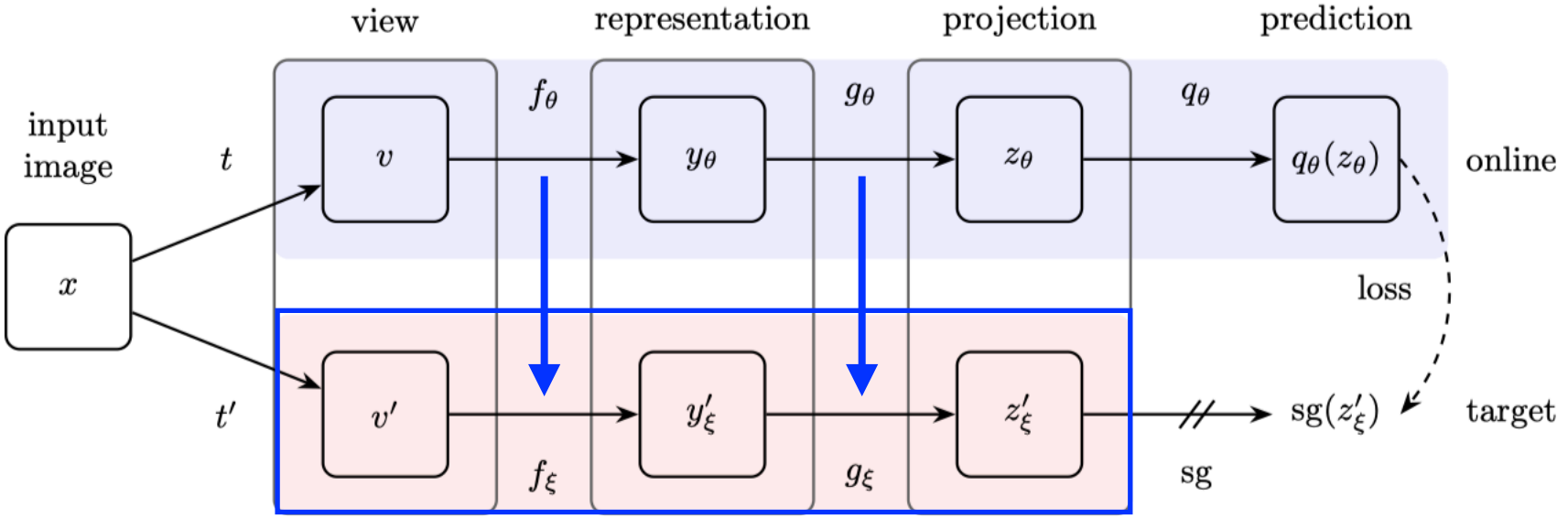


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the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

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

Update Rule

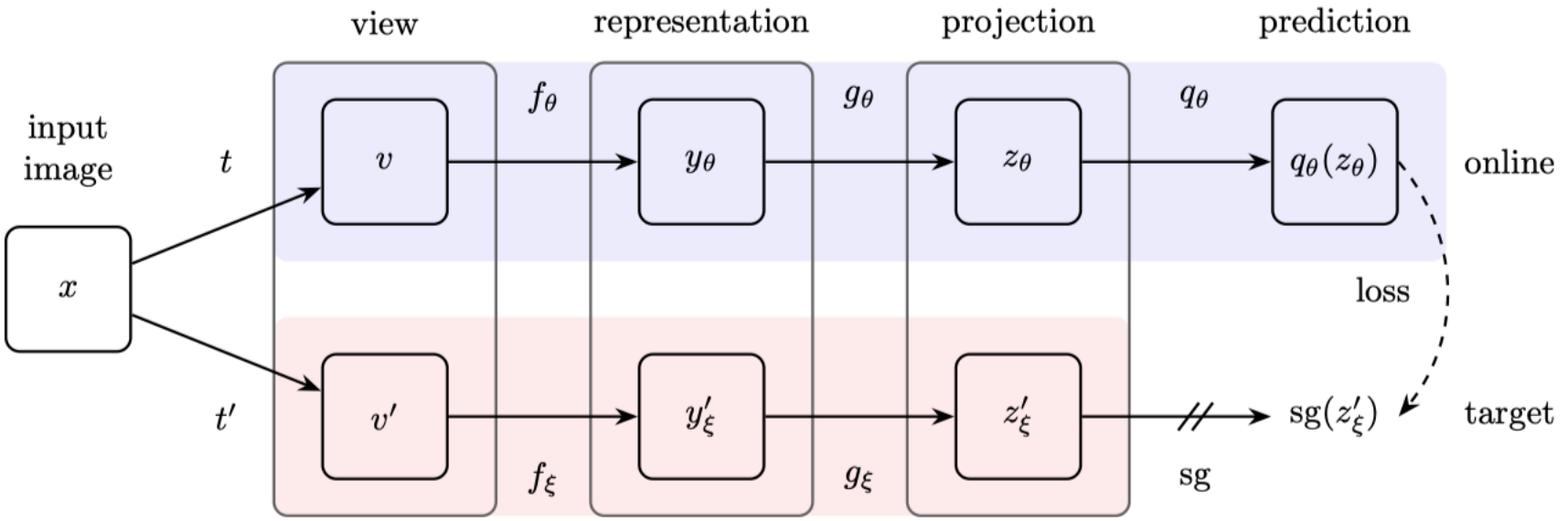


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the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

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

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

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

BYOL

ROBOT INTELLIGENCE LAB

Algorithm 1: BYOL: **B**ootstrap **Y**our **O**wn **L**atent

Inputs :

$\mathcal{D}$, $\mathcal{T}$, and $\mathcal{T}'$	set of images and distributions of transformations
θ, f_θ, g_θ, and q_θ	initial online parameters, encoder, projector, and predictor
ξ, f_ξ, g_ξ	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

for $k = 1$ **to** K **do**

  $\mathcal{B} \leftarrow \{x_i \sim \mathcal{D}\}_{i=1}^{N}$ // sample a batch of N images

  for $x_i \in \mathcal{B}$ **do**

    $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}'$ // sample image transformations

    $z_1 \leftarrow g_\theta(f_\theta(t(x_i)))$ and $z_2 \leftarrow g_\theta(f_\theta(t'(x_i)))$ // compute projections

    $z'_1 \leftarrow g_\xi(f_\xi(t'(x_i)))$ and $z'_2 \leftarrow g_\xi(f_\xi(t(x_i)))$ // compute target projections

    $l_i \leftarrow -2 \cdot \left(\frac{\langle q_\theta(z_1), z'_1 \rangle}{\|q_\theta(z_1)\|_2 \cdot \|z'_1\|_2} + \frac{\langle q_\theta(z_2), z'_2 \rangle}{\|q_\theta(z_2)\|_2 \cdot \|z'_2\|_2} \right)$ // compute the loss for x_i

  end

  $\delta\theta \leftarrow \frac{1}{N} \sum_{i=1}^{N} \partial_\theta l_i$ // compute the total loss gradient w.r.t. θ

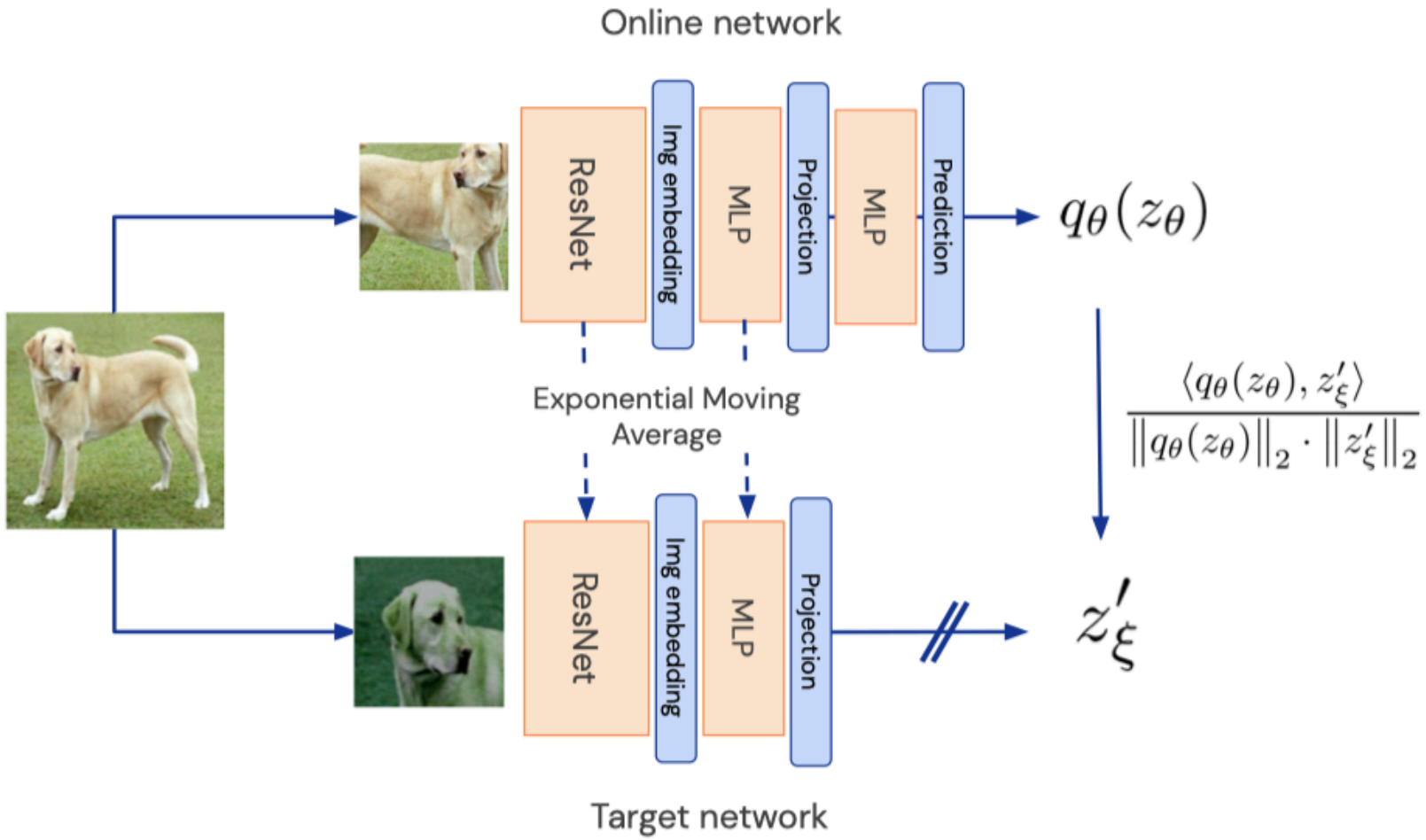
  $\theta \leftarrow \text{optimizer}(\theta, \delta\theta, \eta_k)$ // update online parameters

  $\xi \leftarrow \tau_k \xi + (1 - \tau_k)\theta$ // update target parameters

end

Output : encoder f_θ

BYOL


Online network
ResNet
Img embedding
MLP
Projection
MLP
Prediction
$q_\theta(z_\theta)$
Exponential Moving Average
$\frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\|q_\theta(z_\theta)\|_2 \cdot \|z'_\xi\|_2}$
ResNet
Img embedding
MLP
Projection
z'_ξ
Target network

Linear Probing

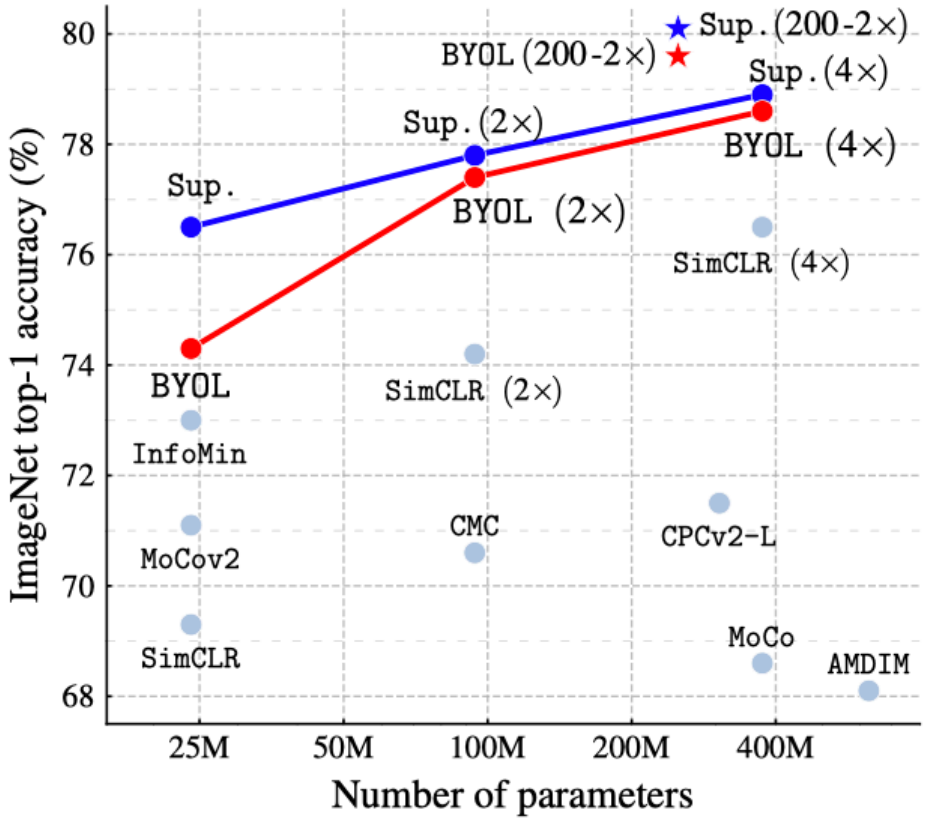
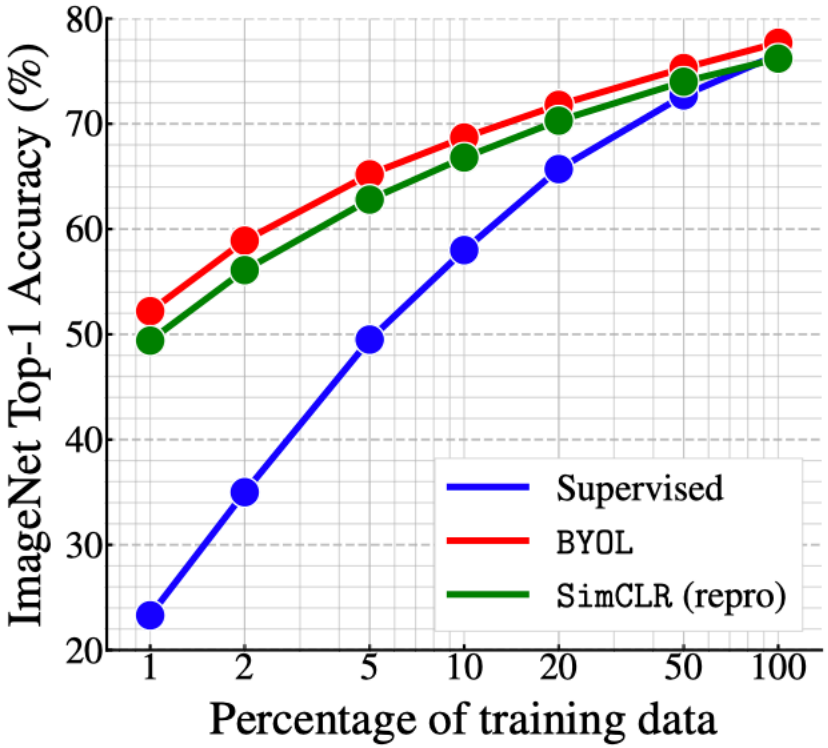


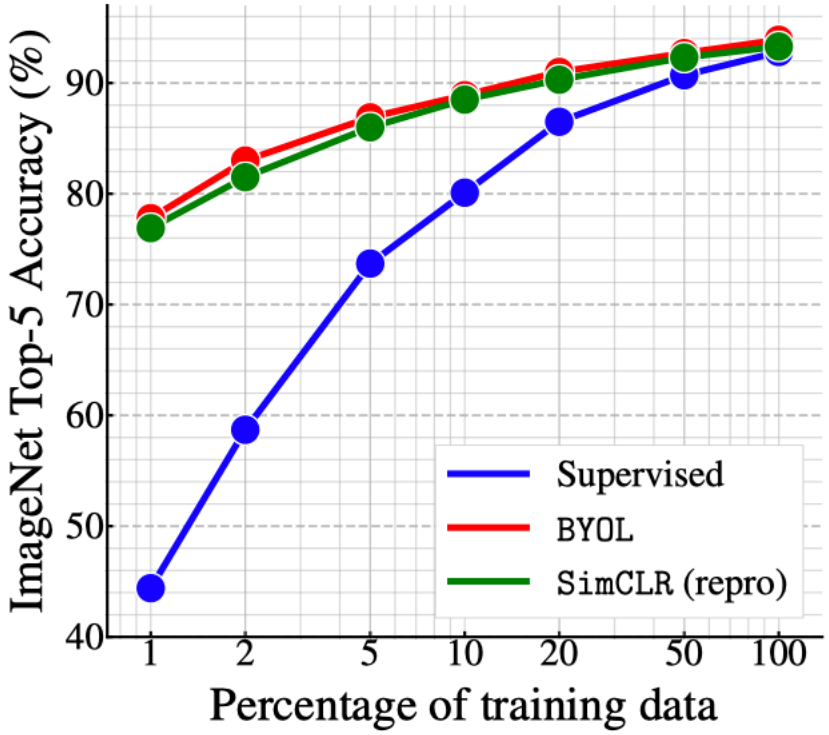
Figure 1: Performance of BYOL on ImageNet (linear evaluation) using ResNet-50 and our best architecture ResNet-200 (2×), compared to other unsupervised and supervised (Sup.) baselines [8].

Fraction of Labels

ROBOT INTELLIGENCE LAB



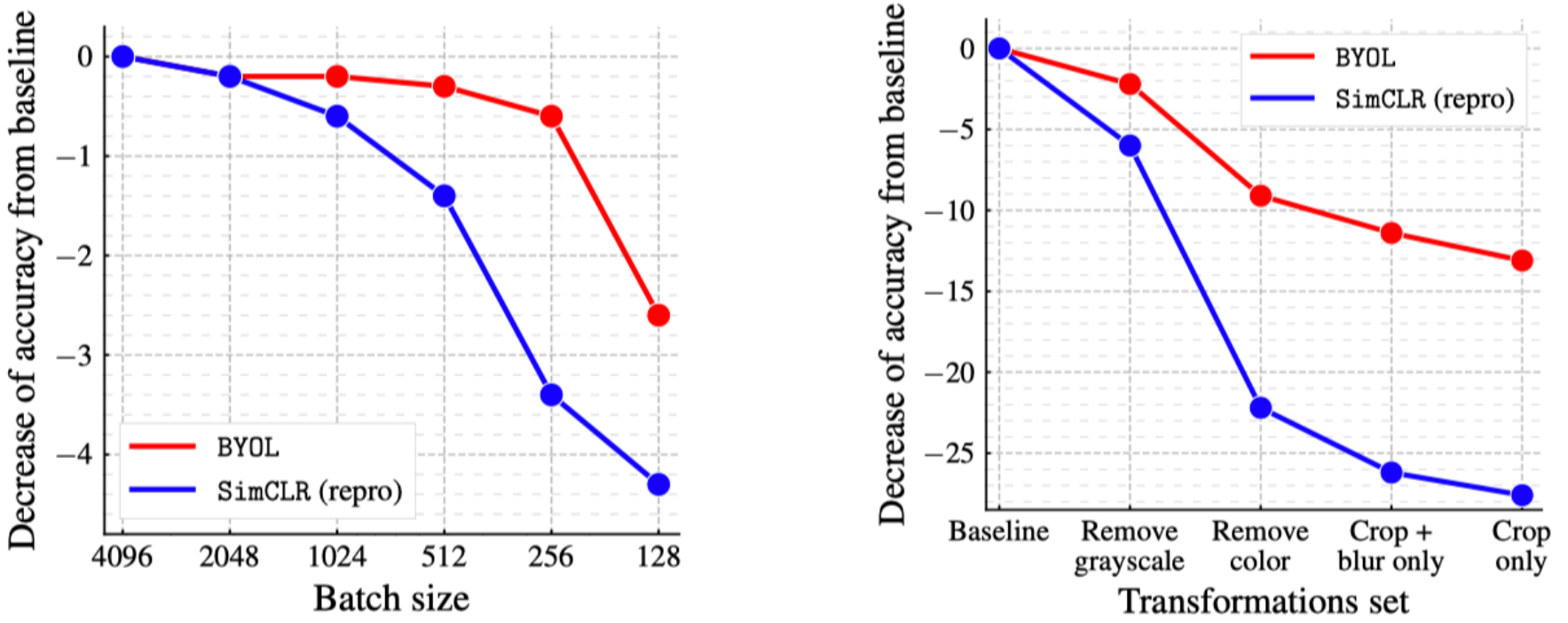
(a) Top-1 accuracy



(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**.

ROBOT INTELLIGENCE LAB

SimCLRv2

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

Key Ingredients

ROBOT INTELLIGENCE LAB

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

ROBOT INTELLIGENCE LAB

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

Interesting Result

ROBOT INTELLIGENCE LAB

SimCLR

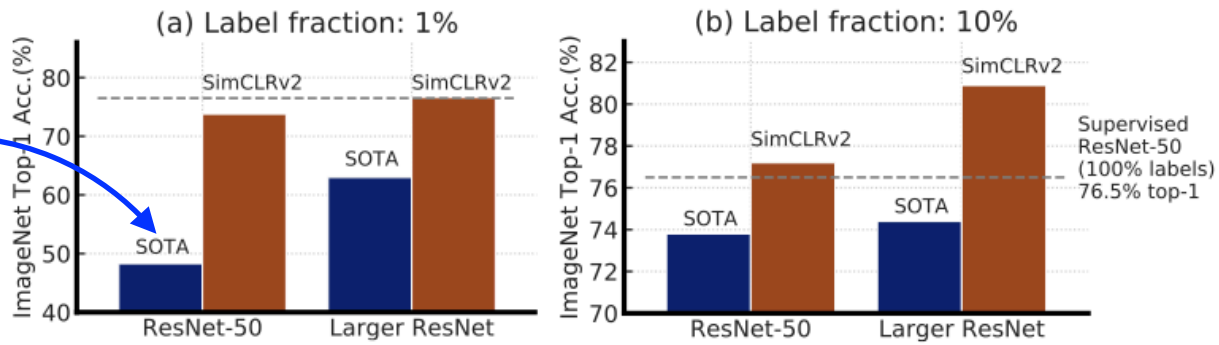


Figure 2: Top-1 accuracy of previous state-of-the-art (SOTA) methods [1, 2] and our method (SimCLRv2) on ImageNet using only 1% or 10% of the labels. Dashed line denotes fully supervised ResNet-50 trained with 100% of labels. Full comparisons in Table 3.

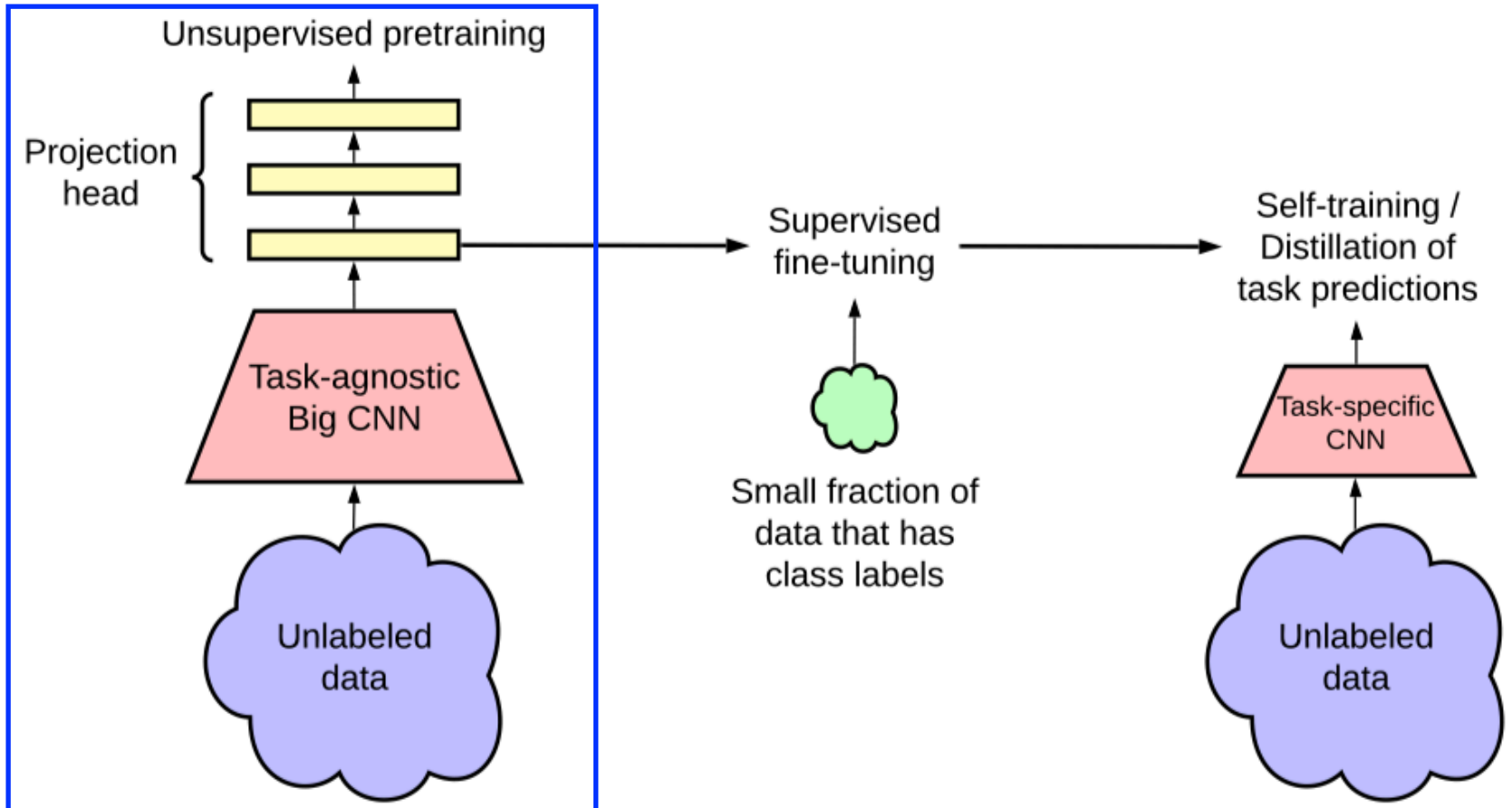
SimCLRv2 even surpasses supervised learning results on ImageNet!

Key Findings

ROBOT INTELLIGENCE LAB

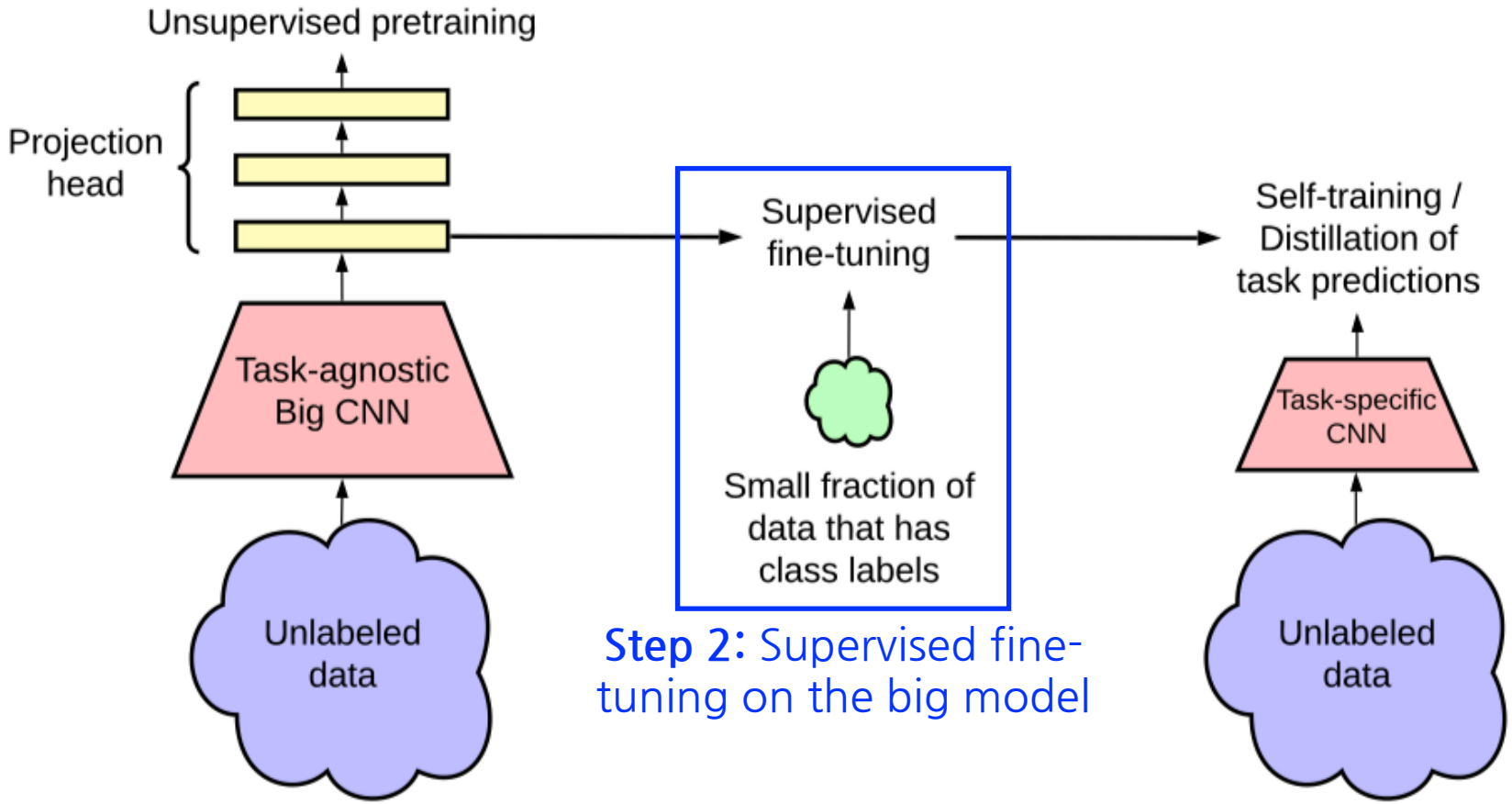
- Bigger self-supervised models are more **label efficient**, performing significantly better when fine-tuned on only a few labeled examples, even though they have more capacity to potentially overfit.
- Train a small model for a target task from the big model with unsupervised data (**distillation**)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

SimCLRv2



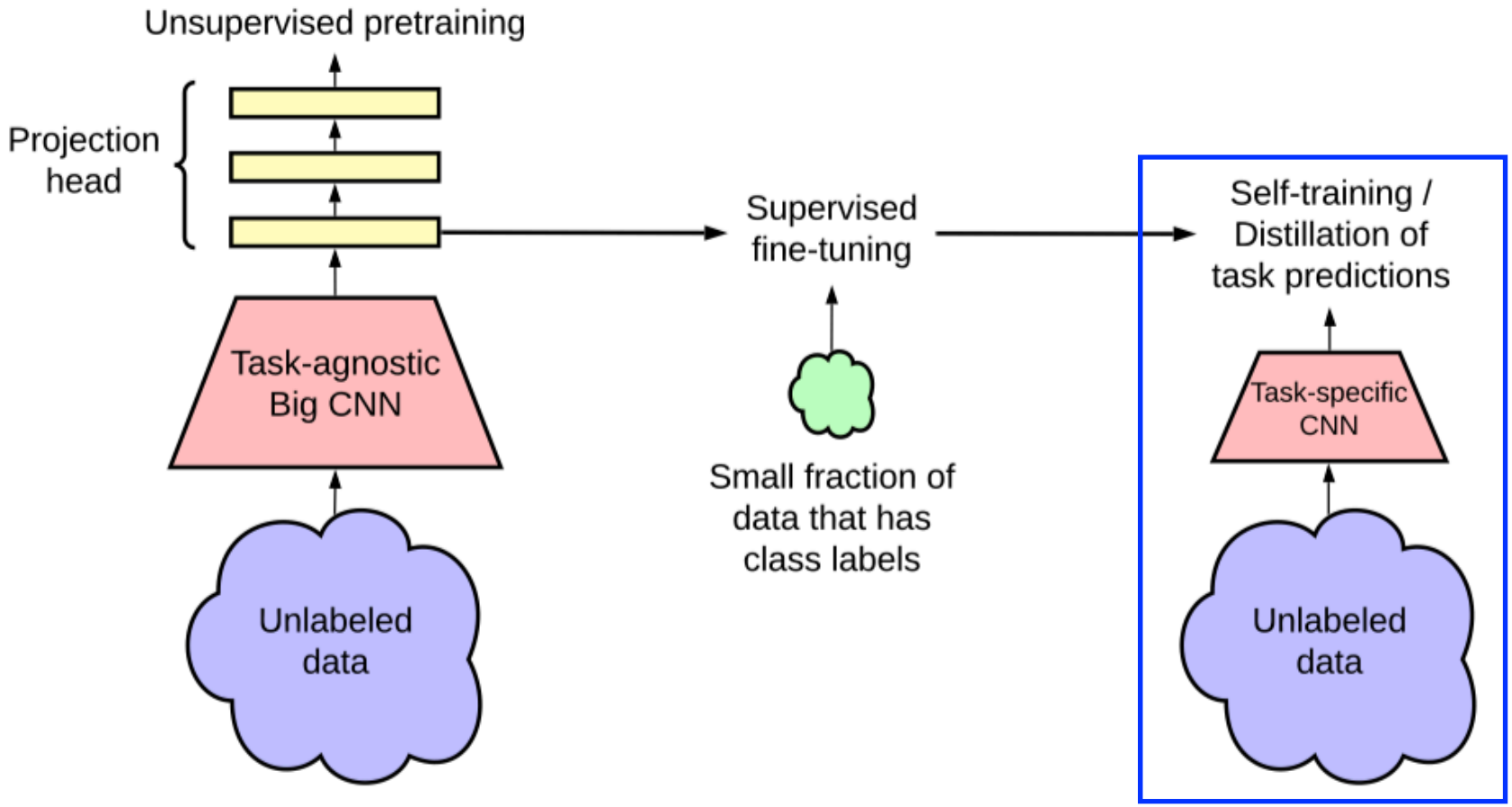
Step 1: SimCLR with a bigger model and deeper projection head

SimCLRv2



Step 2: Supervised fine-tuning on the big model

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 a 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 Distilation

ROBOT INTELLIGENCE LAB

- 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

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

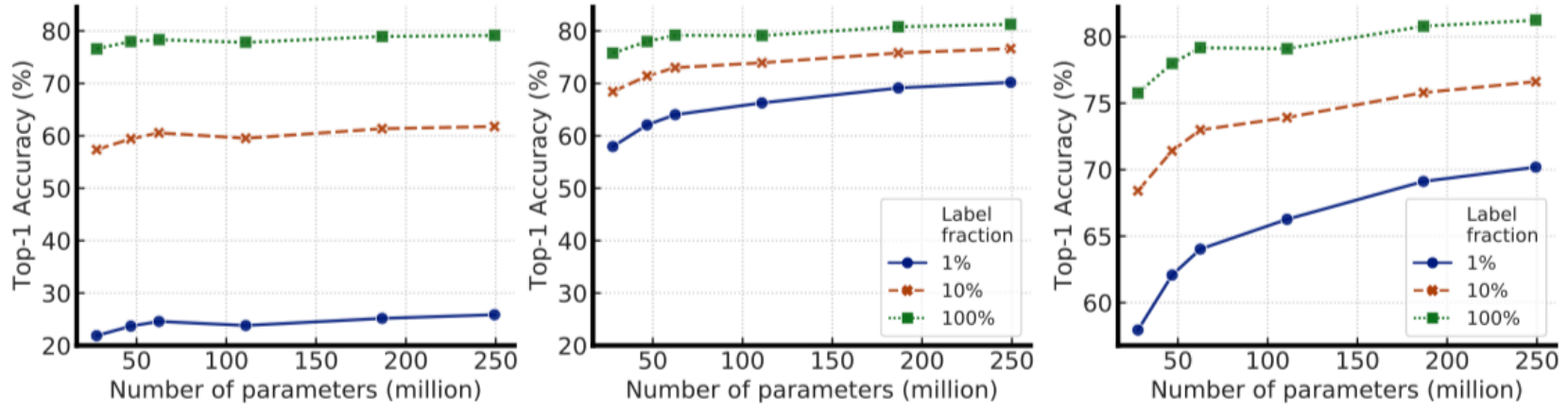
where $P^{T}(\,\cdot\,)$ and $P^{S}(\,\cdot\,)$ are teacher and student models, respectively, and only the student network $P^{S}(y\,|\,\mathbf{x};\tau)$ is trained.

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 1%	Fine-tuned on 10%	Fine-tuned on 100%	Linear eval	Supervised
50	1×	False	**24**	**57.9**	**68.4**	**76.3**	**71.7**	**76.6**
		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	**795**	**74.9**	**80.1**	**83.1**	**79.8**	**80.5**

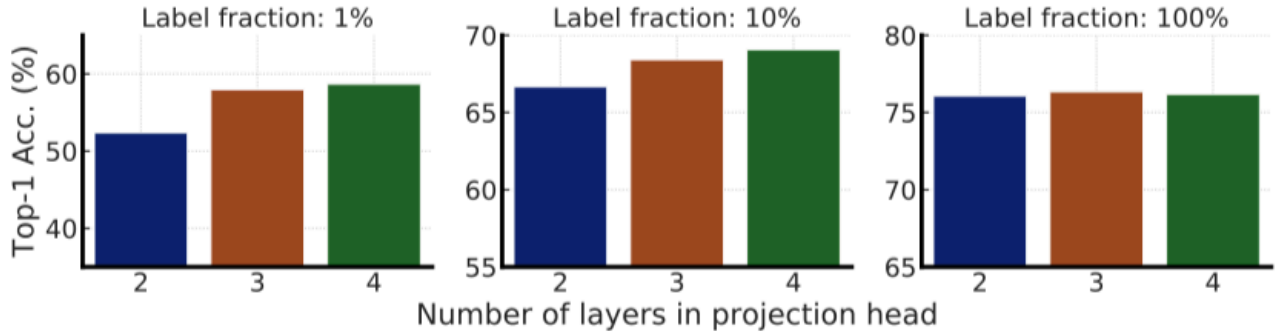
Bigger Models are More Label-Efficient



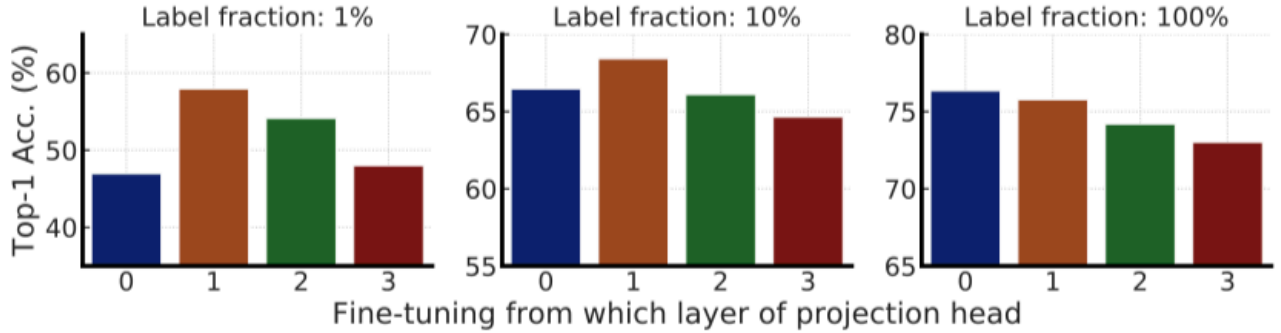
(a) Supervised (b) Semi-supervised (c) Semi-supervised (y-axis zoomed)

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×, 2× (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×, from ResNet-50 to ResNet-152 (2×), improves label efficiency by 10×.

Projection Heads



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

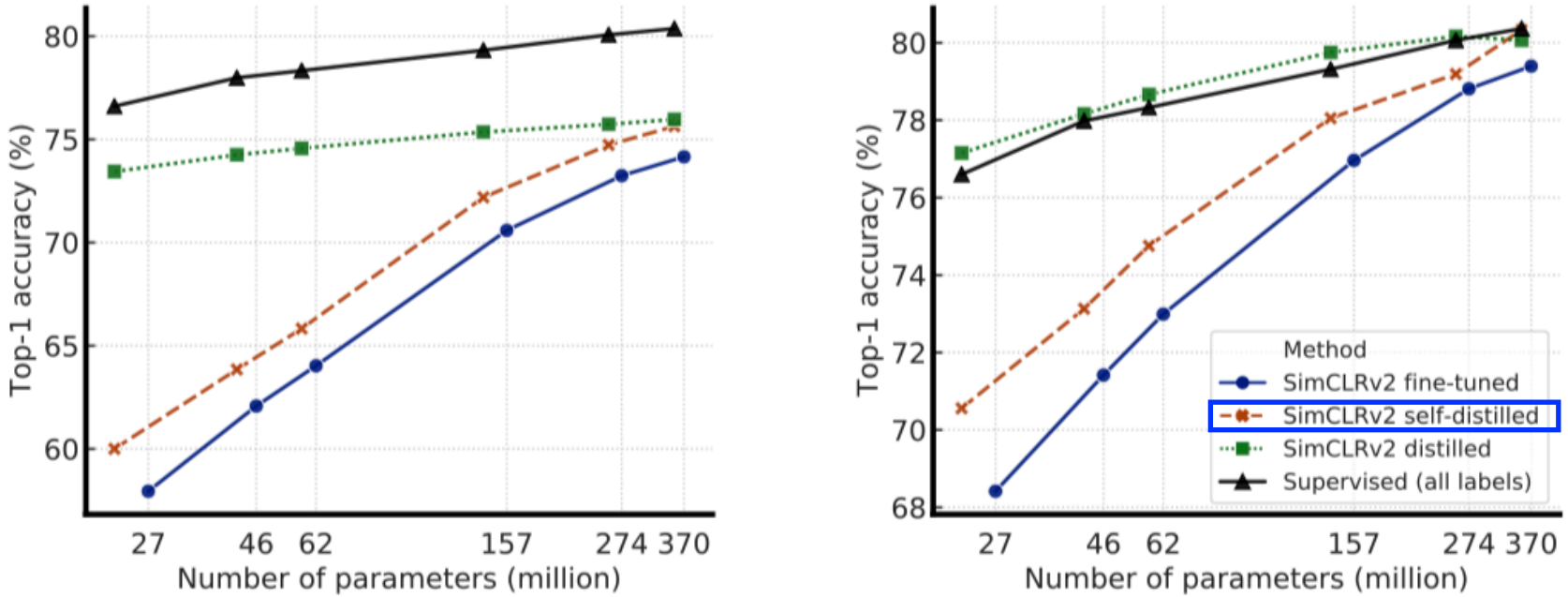


(b) Effect of fine-tuning from middle of a 3-layer projection head (0 is SimCLR).

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

Distillation

ROBOT INTELLIGENCE LAB



(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×+SK) model, which is the largest model included in this figure.

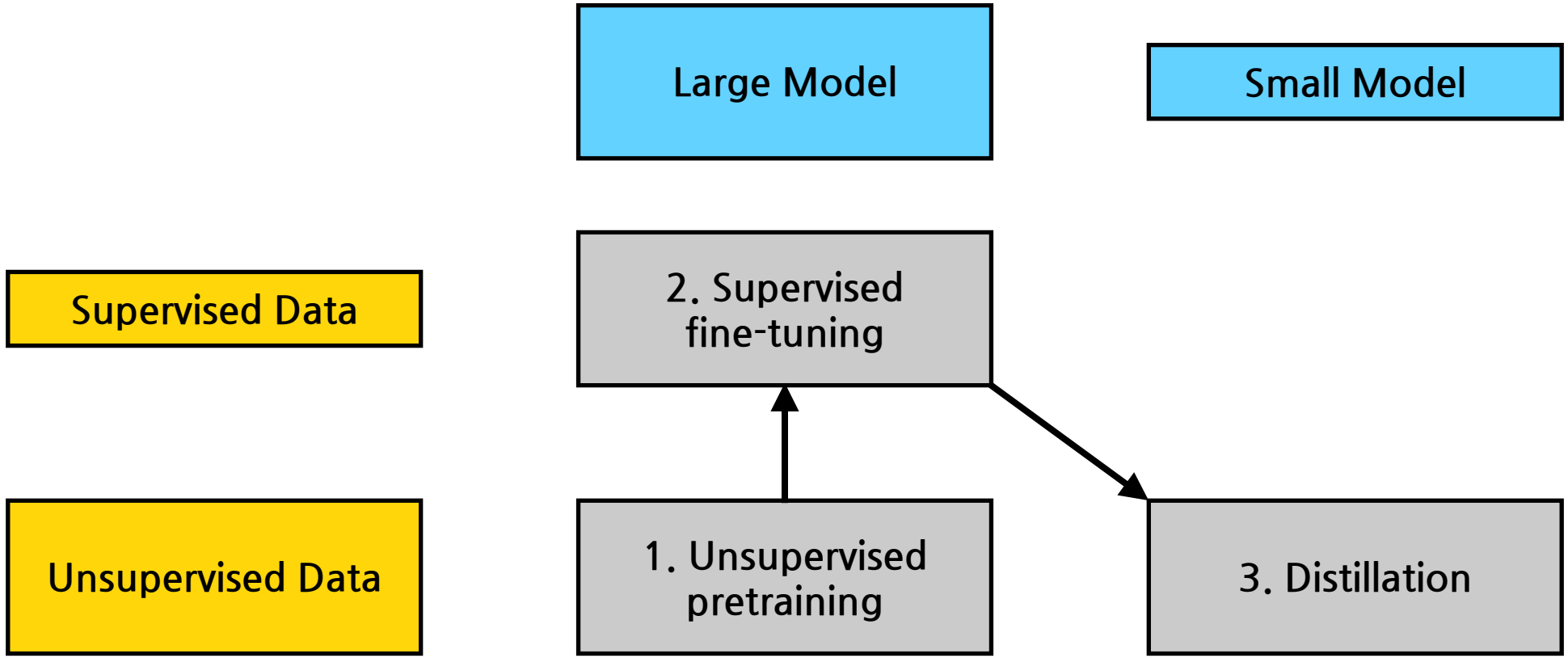
ImageNet Accuracy

ROBOT INTELLIGENCE LAB

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×+SK) as the teacher.

Method	Architecture	Top-1 Label fraction 1%	Top-1 Label fraction 10%	Top-5 Label fraction 1%	Top-5 Label fraction 10%
Supervised baseline [30]	ResNet-50	25.4	56.4	48.4	80.4
Methods using unlabeled data in a task-specific way:					
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×)	-	73.2	-	91.2
MPL (w. RandAug) [2]	ResNet-50	-	73.8	-	-
CowMix [40]	ResNet-152	-	73.9	-	91.2
Methods using unlabeled data in a task-agnostic way:					
InstDisc [17]	ResNet-50	-	-	39.2	77.4
BigBiGAN [41]	RevNet-50 (4×)	-	-	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×)	58.5	71.7	83.0	91.2
SimCLR [1]	ResNet-50 (4×)	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×)	71.2	77.7	89.5	93.7
Methods using unlabeled data in both ways:					
SimCLRv2 distilled (ours)	ResNet-50	73.9	77.5	91.5	93.4
SimCLRv2 distilled (ours)	ResNet-50 (2×+SK)	75.9	80.2	93.0	95.0
SimCLRv2 self-distilled (ours)	ResNet-152 (3×+SK)	**76.6**	**80.9**	**93.4**	**95.5**

SimCLRv2


Large Model
Small Model
Supervised Data
2. Supervised fine-tuning
Unsupervised Data
1. Unsupervised pretraining
3. Distillation

ROBOT INTELLIGENCE LAB

SwAV

"Unsupervised Learning of Visual Features by Contrasting Cluster Assignments," 2021

SwAV

ROBOT INTELLIGENCE LAB

Learn features by Swapping Assignments between multiple Views of the same image (SwAV)

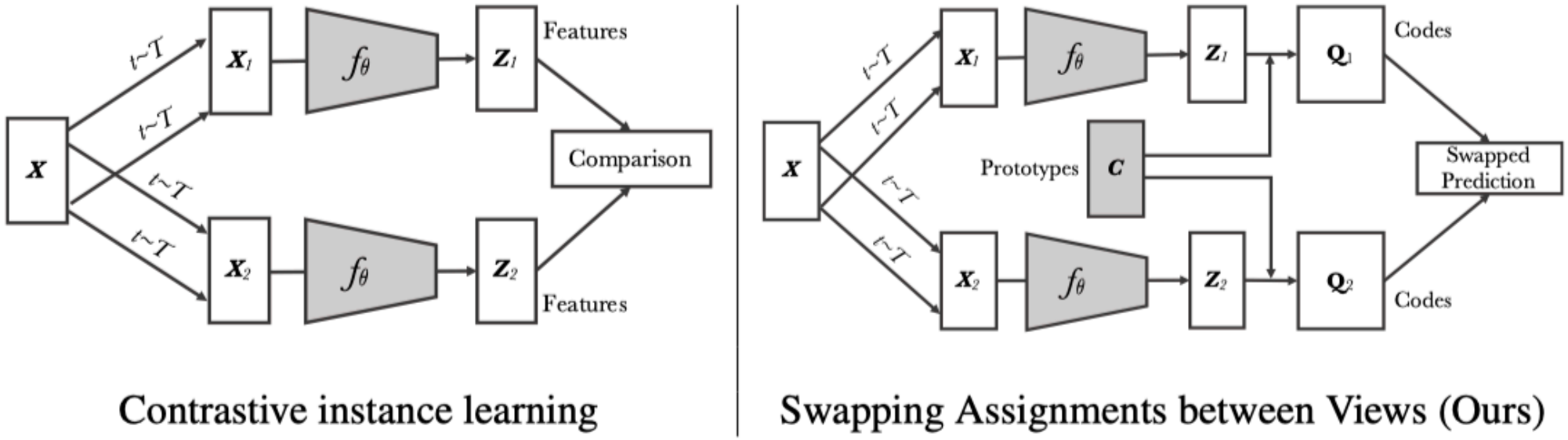
SwAV

ROBOT INTELLIGENCE LAB

- Recent SSL methods rely on two elements: (i) a contrastive loss and (ii) a set of image transformations.
- The proposed method use a **swapped prediction mechanism** where the code of a view is predicted from the representation of another view.
- SwAV can be trained with large and small batches and does not require a large memory bank or a momentum network.
- Multi-crop, a new data augmentation method is presented.

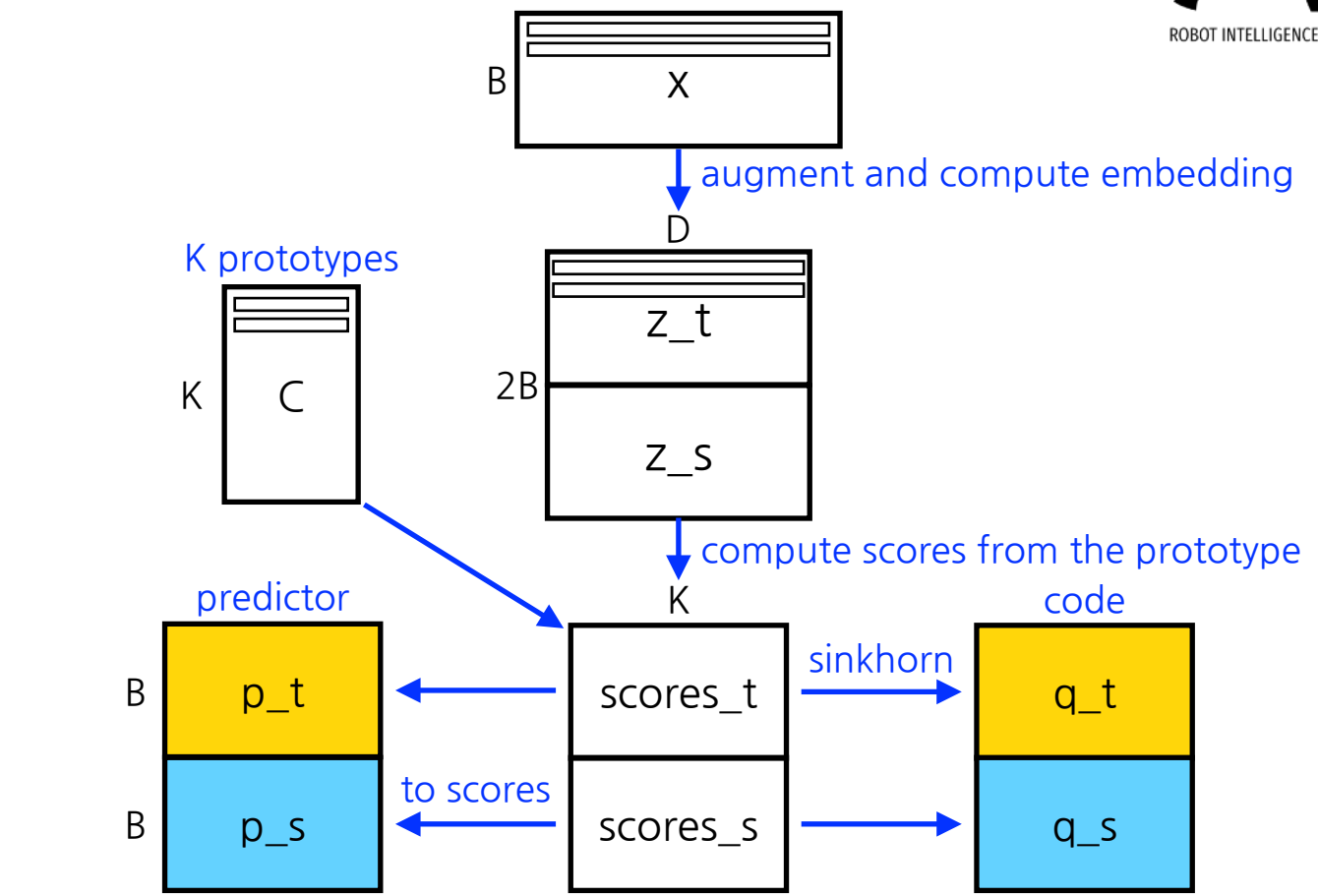
SwAV

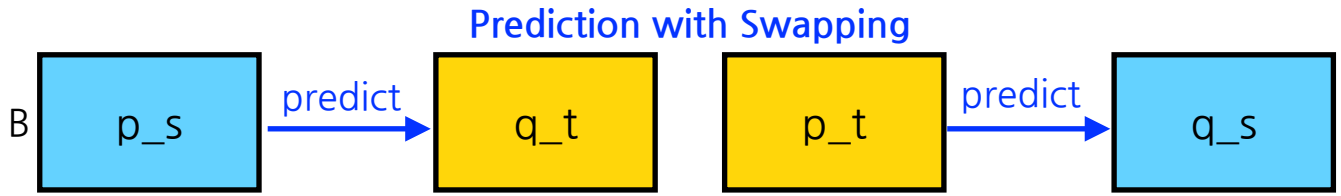
ROBOT INTELLIGENCE LAB



SwAV

```
for x in loader: # load a batch x with B samples
   x_t = t(x) # t is a random augmentation
   x_s = s(x) # s is a another random augmentation

   z = model(cat(x_t, x_s)) # embeddings:  2BxD

   scores = mm(z, C) # prototype scores:  2BxK
   scores_t = scores[:B]
   scores_s = scores[B:]

   # compute assignments
   with torch.no_grad():
      q_t = sinkhorn(scores_t)
      q_s = sinkhorn(scores_s)

   # convert scores to probabilities
   p_t = Softmax(scores_t / temp)
   p_s = Softmax(scores_s / temp)

   # swap prediction problem
   loss = - 0.5 * mean(q_t * log(p_s) + q_s * log(p_t))

   # SGD update:  network and prototypes
   loss.backward()
   update(model.params)
   update(C)

   # normalize prototypes
   with torch.no_grad():
      C = normalize(C, dim=0, p=2)
```





ImageNet LinearProbe

ROBOT INTELLIGENCE LAB

Method	Arch.	Param.	Top1
Supervised	R50	24	76.5
Colorization [65]	R50	24	39.6
Jigsaw [46]	R50	24	45.7
NPID [58]	R50	24	54.0
BigBiGAN [15]	R50	24	56.6
LA [68]	R50	24	58.8
NPID++ [44]	R50	24	59.0
MoCo [24]	R50	24	60.6
SeLa [2]	R50	24	61.5
PIRL [44]	R50	24	63.6
CPC v2 [28]	R50	24	63.8
PCL [37]	R50	24	65.9
SimCLR [10]	R50	24	70.0
MoCov2 [11]	R50	24	71.1
SwAV	R50	24	**75.3**

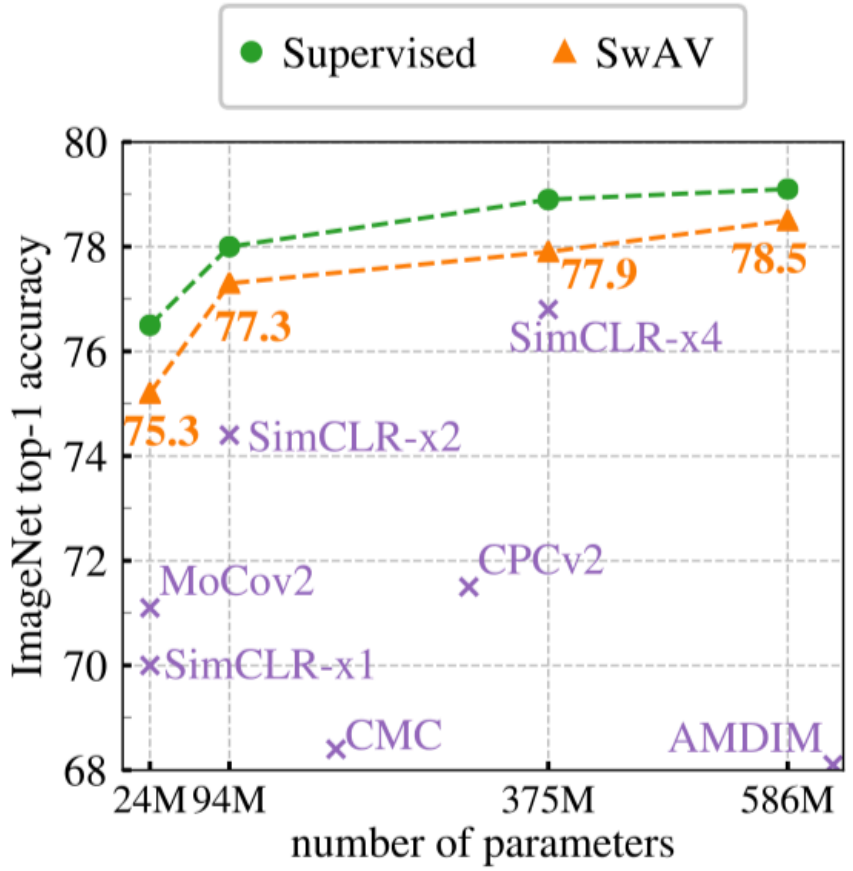


Figure 2: **Linear classification on ImageNet.** Top-1 accuracy for linear models trained on frozen features from different self-supervised methods. **(left)** Performance with a standard ResNet-50. **(right)** Performance as we multiply the width of a ResNet-50 by a factor $\times 2$, $\times 4$, and $\times 5$.

ROBOT INTELLIGENCE LAB

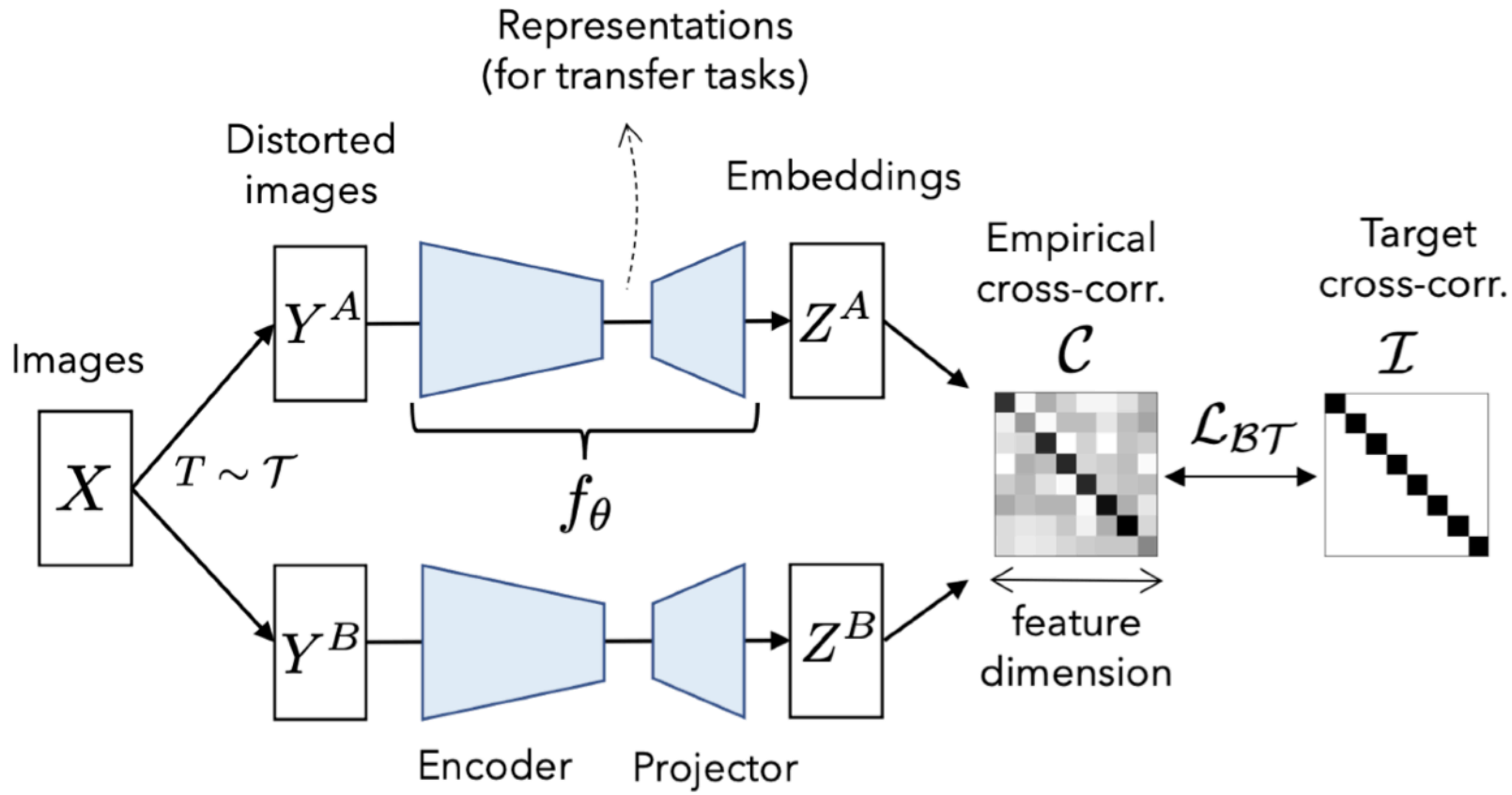
Barlow Twins

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

Self Supervised Learning

- A successful approach to SSL is to learn embeddings which are invariant to distortions of the input sample. However, a recurring issue with this approach is the existence of **trivial constant solutions**.
- This paper proposes an objective function that naturally avoids collapse by measuring the **cross-correlation matrix** between the outputs of **two identical networks** fed with distorted versions of a sample, and making it as close to the identity matrix as possible.
- The proposed **Barlow Twins** doe not require large batches (**SimCLR**) nor asymmetry between the network twins such as a predictor network (**BYOL**), gradient stopping (**BYOL**), or a moving average (**MoCo**) on the weight updates.

Barlow Twins



Barlow Twins

ROBOT INTELLIGENCE LAB

- Barlow Twins loss function

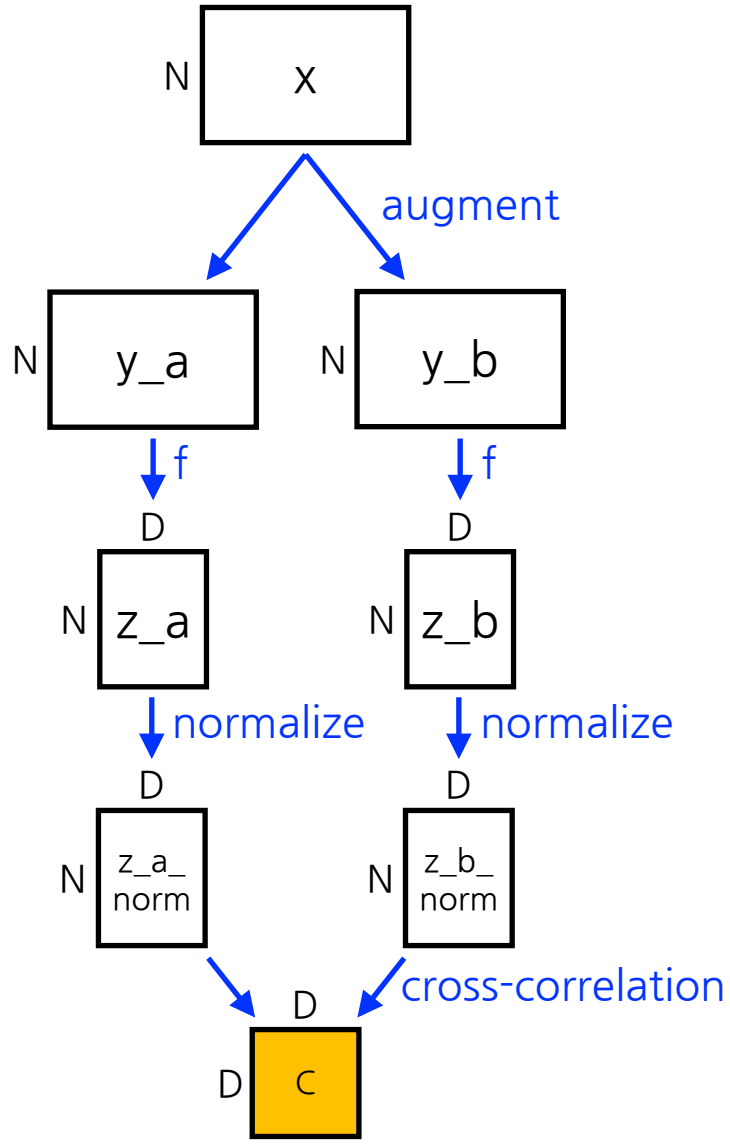
$$\mathcal{L}_{BT} = \underbrace{\boxed{\sum_i (1 - C_{ii})^2}}_{\text{invariance term}} + \lambda \underbrace{\boxed{\sum_i \sum_{j \neq i} C_{ij}^2}}_{\text{redundancy reduction term}}$$

where $C_{ij} = \dfrac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b (z^A_{b,i})^2} \sqrt{\sum_b (z^B_{b,j})^2}}$ is the cross-correlation matrix computed between the outputs of the two identical networks along the batch dimension and b indexes batch samples and i, j index the vector dimension of the networks' outputs.

Barlow Twins

Algorithm 1 PyTorch-style pseudocode for Barlow Twins.

```
# f: encoder network
# lambda: weight on the off-diagonal terms
# N: batch size
# D: dimensionality of the embeddings
#
# mm: matrix-matrix multiplication
# off_diagonal: off-diagonal elements of a matrix
# eye: identity matrix

for x in loader: # load a batch with N samples
    # two randomly augmented versions of x
    y_a, y_b = augment(x)

    # compute embeddings
    z_a = f(y_a) # NxD
    z_b = f(y_b) # NxD

    # normalize repr. along the batch dimension
    z_a_norm = (z_a - z_a.mean(0)) / z_a.std(0) # NxD
    z_b_norm = (z_b - z_b.mean(0)) / z_b.std(0) # NxD

    # cross-correlation matrix
    c = mm(z_a_norm.T, z_b_norm) / N # DxD

    # loss
    c_diff = (c - eye(D)).pow(2) # DxD
    # multiply off-diagonal elems of c_diff by lambda
    off_diagonal(c_diff).mul_(lambda)
    loss = c_diff.sum()

    # optimization step
    loss.backward()
    optimizer.step()
```

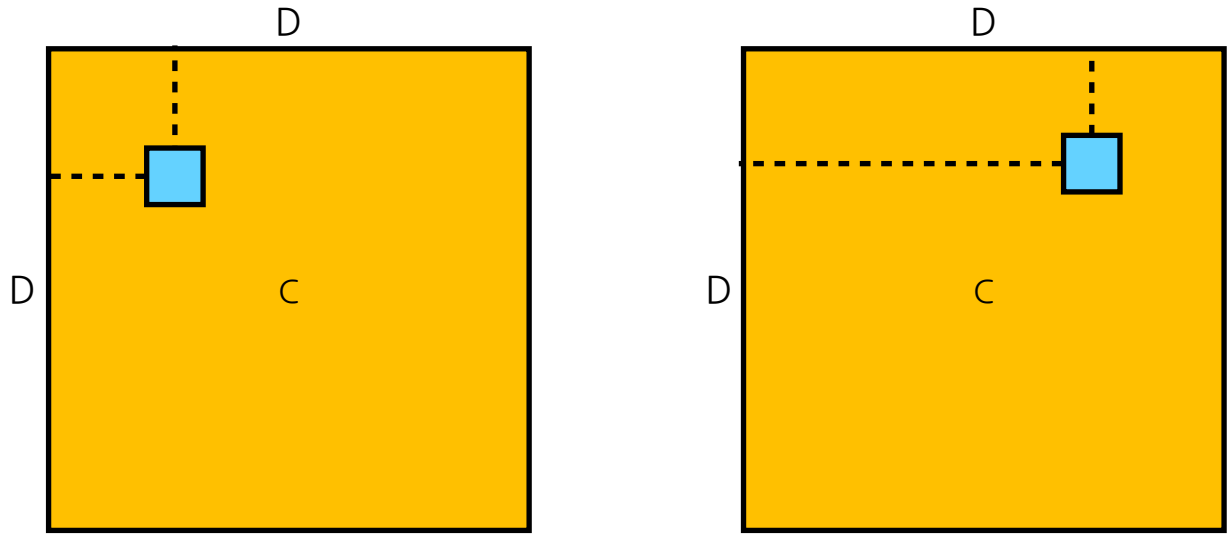


Barlow Twins

ROBOT INTELLIGENCE LAB

Algorithm 1 PyTorch-style pseudocode for Barlow Twins.

```
# f: encoder network
# lambda: weight on the off-diagonal terms
# N: batch size
# D: dimensionality of the embeddings
#
# mm: matrix-matrix multiplication
# off_diagonal: off-diagonal elements of a matrix
# eye: identity matrix

for x in loader: # load a batch with N samples
    # two randomly augmented versions of x
    y_a, y_b = augment(x)

    # compute embeddings
    z_a = f(y_a) # NxD
    z_b = f(y_b) # NxD

    # normalize repr. along the batch dimension
    z_a_norm = (z_a - z_a.mean(0)) / z_a.std(0) # NxD
    z_b_norm = (z_b - z_b.mean(0)) / z_b.std(0) # NxD

    # cross-correlation matrix
    c = mm(z_a_norm.T, z_b_norm) / N # DxD

    # loss
    c_diff = (c - eye(D)).pow(2) # DxD
    # multiply off-diagonal elems of c_diff by lambda
    off_diagonal(c_diff).mul_(lambda)
    loss = c_diff.sum()

    # optimization step
    loss.backward()
    optimizer.step()
```



ImageNet Linear Probing

ROBOT INTELLIGENCE LAB

Table 1. **Top-1 and top-5 accuracies (in %) under linear evaluation on ImageNet.** All models use a ResNet-50 encoder. Top-3 best self-supervised methods are underlined.

Method	Top-1	Top-5
Supervised	76.5	
MoCo	60.6	
PIRL	63.6	-
SimCLR	69.3	89.0
MoCo v2	71.1	90.1
SimSiam	71.3	-
SwAV (w/o multi-crop)	71.8	-
BYOL	74.3	91.6
SwAV	75.3	-
Barlow Twins (ours)	73.2	91.0

Batch size?

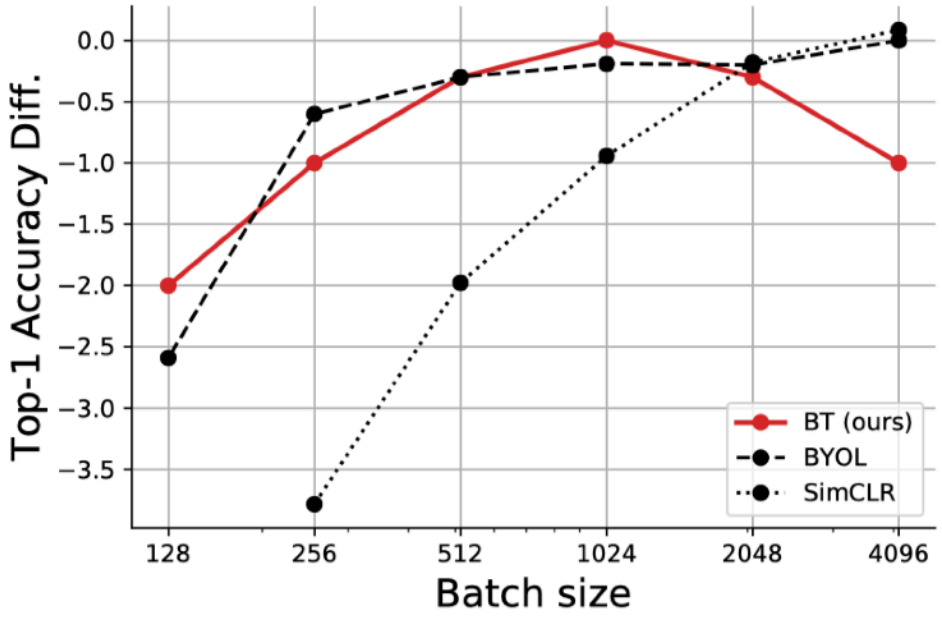


Figure 2. Effect of batch size. To compare the effect of the batch size across methods, for each method we report the difference between the top-1 accuracy at a given batch size and the best obtained accuracy among all batch size tested. BYOL: best accuracy is 72.5% for a batch size of 4096 (data from (Grill et al., 2020) fig. 3A). SIMCLR: best accuracy is 67.1% for a batch size of 4096 (data from (Chen et al., 2020a) fig. 9, model trained for 300 epochs). BARLOW TWINS: best accuracy is 71.7% for a batch size of 1024.

Latent Dimension?

ROBOT INTELLIGENCE LAB

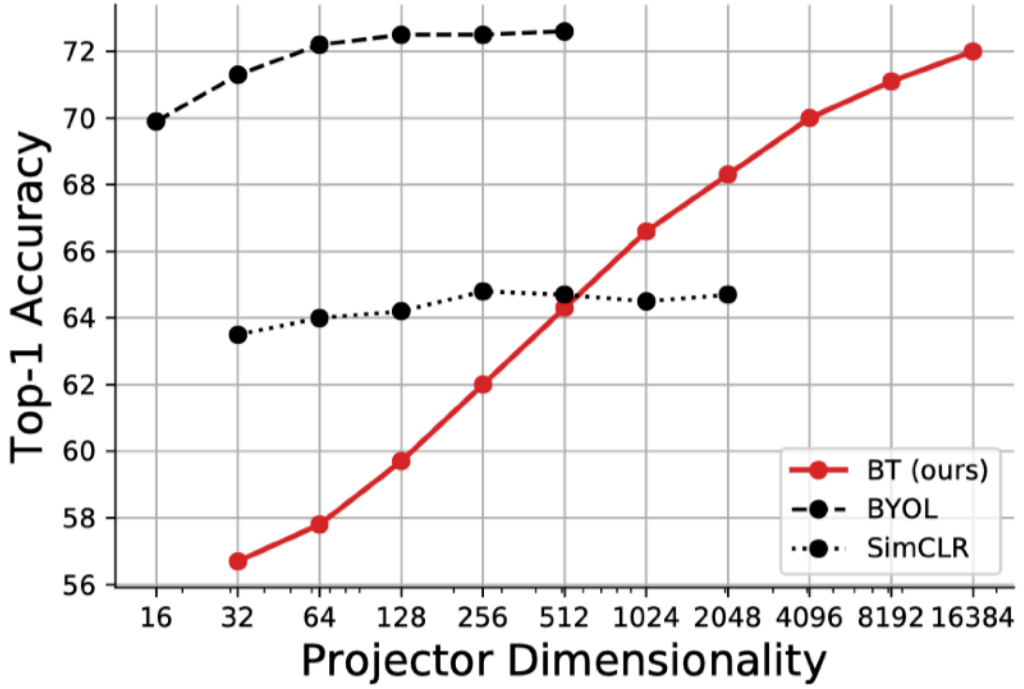


Figure 4. Effect of the dimensionality of the last layer of the projector network on performance. The parameter λ is kept fix for all dimensionalities tested. Data for SIMCLR is from (Chen et al., 2020a) fig 8; Data for BYOL is from (Grill et al., 2020) Table 14b.

Thank You