



# Computer Vision; Image Classification; Self-training & Contrastive Learning

---

**Maziar Raissi**

**Assistant Professor**

Department of Applied Mathematics

University of Colorado Boulder

[maziar.raissi@colorado.edu](mailto:maziar.raissi@colorado.edu)



Boulder

# Self-training with Noisy Student improves ImageNet classification

**Require:** Labeled images  $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$  and unlabeled images  $\{\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_m\}$ .

- 1: Learn teacher model  $\theta_*^t$  which minimizes the cross entropy loss on labeled images

$$\frac{1}{n} \sum_{i=1}^n \ell(y_i, f^{noised}(x_i, \theta^t))$$

- 2: Use an unnoised teacher model to generate soft or hard pseudo labels for unlabeled images

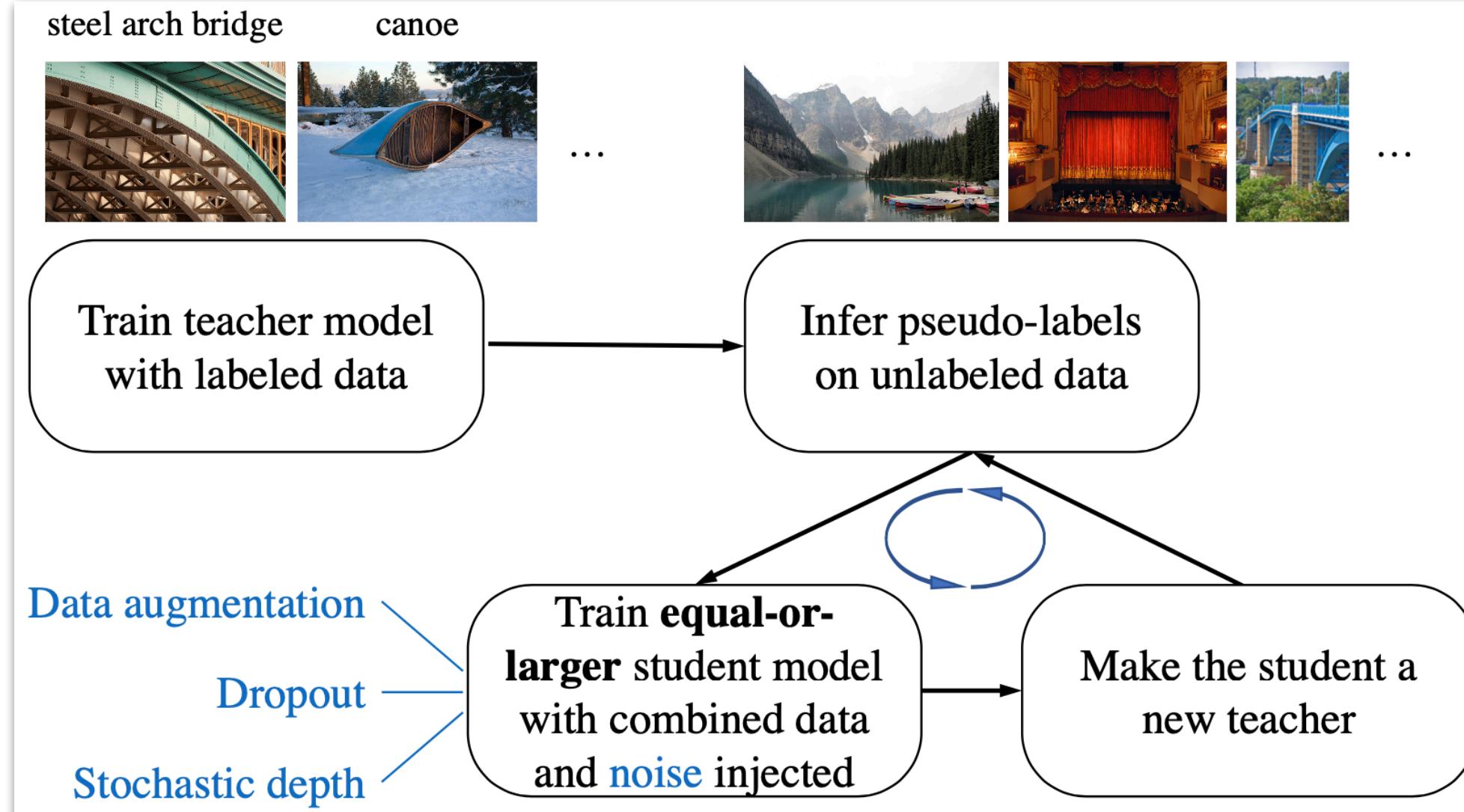
$$\tilde{y}_i = f(\tilde{x}_i, \theta_*^t), \forall i = 1, \dots, m$$

- 3: Learn an **equal-or-larger** student model  $\theta_*^s$  which minimizes the cross entropy loss on labeled images and unlabeled images with **noise** added to the student model

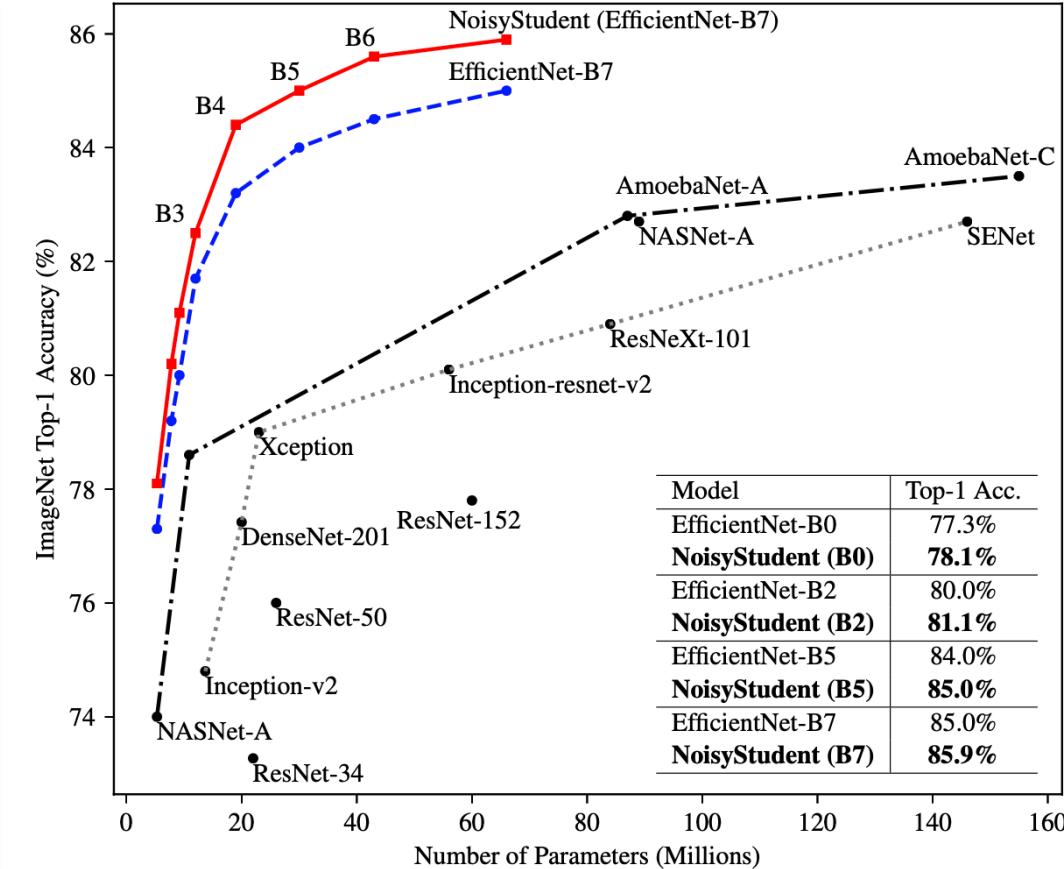
$$\frac{1}{n} \sum_{i=1}^n \ell(y_i, f^{noised}(x_i, \theta^s)) + \frac{1}{m} \sum_{i=1}^m \ell(\tilde{y}_i, f^{noised}(\tilde{x}_i, \theta^s))$$

- 4: Iterative training: Use the student as a teacher and go back to step 2.

	ImageNet top-1 acc.	ImageNet-A top-1 acc.	ImageNet-C mCE	ImageNet-P mFR
Prev. SOTA	86.4%	61.0%	45.7	27.8
NoisyStudent	<b>88.4%</b>	<b>83.7%</b>	<b>28.3</b>	<b>12.2</b>

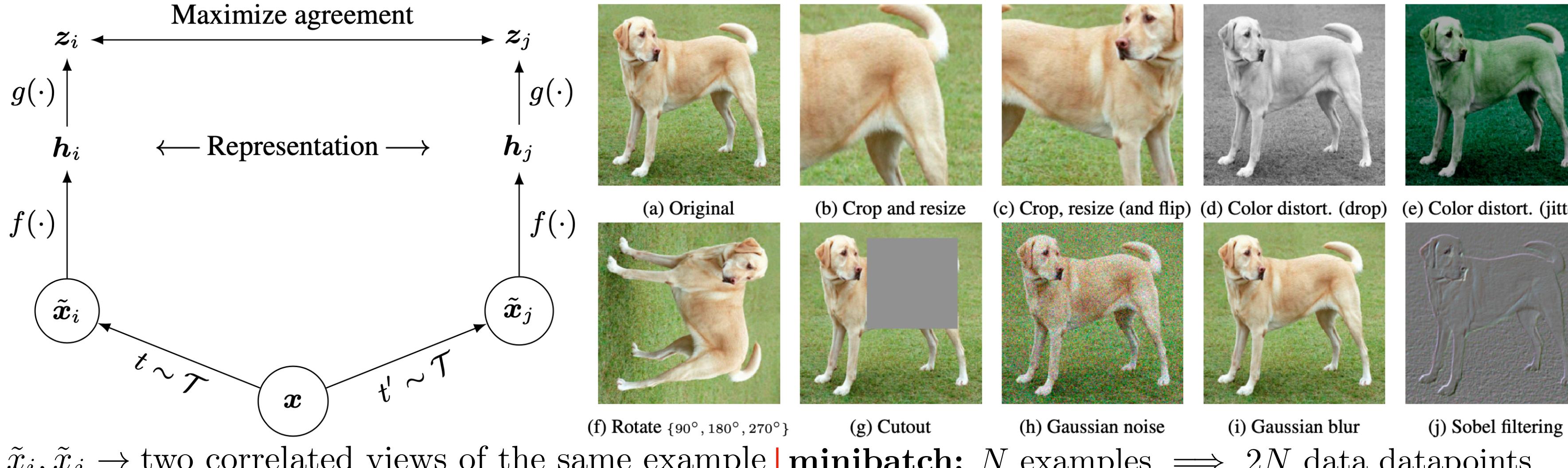


ImageNet-C and ImageNet-P test sets include images with common corruptions and perturbations such as blurring, fogging, rotation and scaling. ImageNet-A test set consists of difficult images that cause significant drops in accuracy to state-of-the-art models. These test sets are considered as “robustness” benchmarks.



mCE (mean corruption error) is the weighted average of error rate on different corruptions, with AlexNet’s error rate as a baseline (lower is better). mFR (mean flip rate) measures the model’s probability of flipping predictions under perturbations with AlexNet as a baseline (lower is better).

# A Simple Framework for Contrastive Learning of Visual Representations



## Data Augmentation

Random cropping and resizing to the original size  
 Random color distortions  
 Random Gaussian blur

## Base Encoder

$$h_i = f(\tilde{x}_i) = \text{ResNet}(\tilde{x}_i)$$

$h_i \in \mathbb{R}^d \rightarrow$  output after average pooling layer

## Projection Head

$$z_i = g(h_i) = W^2 \sigma(W^1 h_i)$$

ReLU

## Contrastive Loss

**Contrastive Prediction Task:** Given a set  $\{\tilde{x}_k\}$  including a positive pair of examples  $\tilde{x}_i$  and  $\tilde{x}_j$ , identify  $\tilde{x}_j$  in  $\{\tilde{x}_k\}_{k \neq i}$  for a given  $\tilde{x}_i$ .

**minibatch:**  $N$  examples  $\implies 2N$  data datapoints

Given a positive pair, treat the other  $2(N - 1)$  augmented examples as negatives

$$\text{sim}(u, v) = u^T v / \|u\| \|v\| \rightarrow \text{cosine similarity}$$

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

NT-Xent Loss (normalized temperature-scaled cross entropy loss)

$\tau \rightarrow$  temperature

The final loss is computed over all positive pairs  $(i, j)$  &  $(j, i)$ .

## Transfer learning performance

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
<i>Linear evaluation:</i>												
SimCLR (ours)	<b>76.9</b>	<b>95.3</b>	80.2	48.4	<b>65.9</b>	60.0	61.2	<b>84.2</b>	<b>78.9</b>	89.2	<b>93.9</b>	<b>95.0</b>
Supervised	75.2	95.7	81.2	56.4	64.9	<b>68.8</b>	<b>63.8</b>	83.8	<b>78.7</b>	92.3	94.1	94.2
<i>Fine-tuned:</i>												
SimCLR (ours)	<b>89.4</b>	<b>98.6</b>	<b>89.0</b>	<b>78.2</b>	<b>68.1</b>	<b>92.1</b>	<b>87.0</b>	<b>86.6</b>	<b>77.8</b>	92.1	<b>94.1</b>	97.6
Supervised	88.7	98.3	88.7	77.8	67.0	91.4	<b>88.0</b>	86.5	<b>78.8</b>	<b>93.2</b>	94.2	<b>98.0</b>
Random init	88.3	96.0	81.9	77.0	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5

## Self-supervised learning 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>

## 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	<b>76.5</b>	<b>93.2</b>

## Semi-supervised learning on ImageNet

Method	Architecture	Label fraction	
		1%	10%
Supervised baseline	ResNet-50	48.4	80.4
Top 5			

## Methods using other label-propagation:

Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2

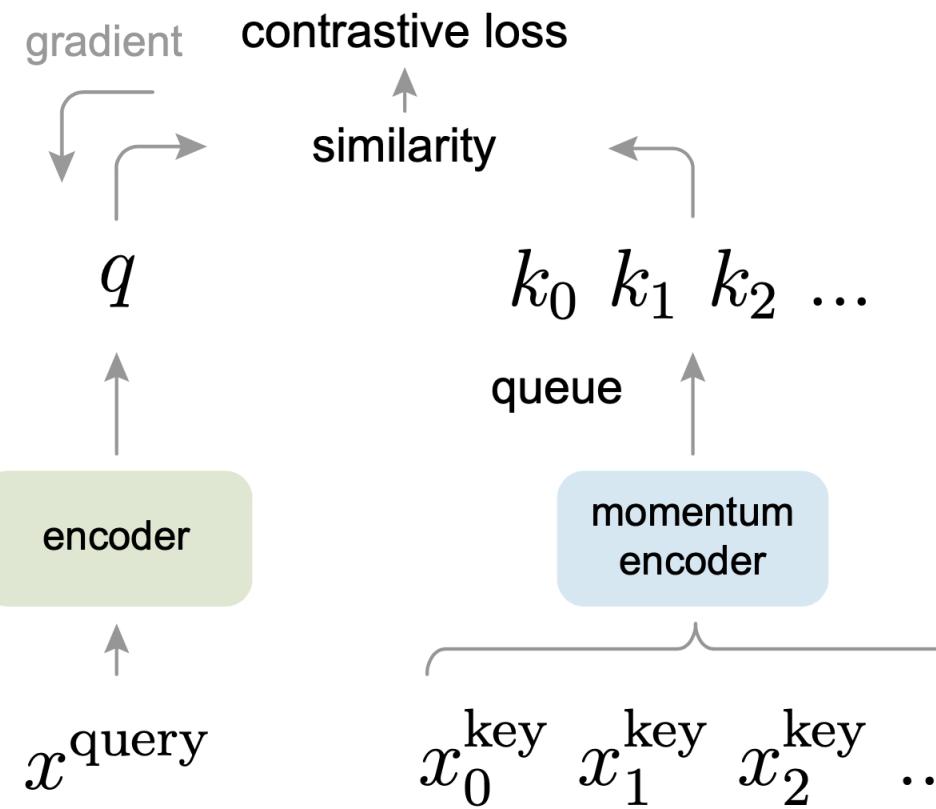
## Methods using representation learning only:

InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(*)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	<b>85.8</b>	<b>92.6</b>



Boulder

# Momentum Contrast for Unsupervised Visual Representation Learning



**Keys (tokens):** samples from the data (e.g., images or patches) represented by an encoder network

**Unsupervised Learning:**

dictionary look-up

An encoded "query" should be similar to its matching "key" and dissimilar to others

**Instance discrimination task**

A query matches a key if they are encoded views (e.g., different crops) of the same image

**Contrastive learning**

$q \rightarrow$  encoded query

$\{k_0, k_1, \dots\} \rightarrow$  set of encoded samples (keys of a dictionary)

$$k_+ \rightarrow \text{the single key in the dictionary that matches } q$$

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

InfoNCE (noise-contrastive estimation) loss

$\tau \rightarrow$  temperature hyper-parameter

$q = f_q(x^q)$  query sample

$k = f_k(x^k)$  encoder network

**Momentum Contrast (MoCo)**

- ① dynamic
  - ② large
  - ③ consistent dictionary
- dictionary as queue

The current mini-batch is enqueue to the dictionary, and the oldest mini-batch in the queue is removed

$\theta_k \rightarrow$  parameters of  $f_k$

$\theta_q \rightarrow$  parameters of  $f_q$

$\theta_k \leftarrow m\theta_k + (1-m)\theta_q, m \in [0, 1], m = 0.999$

$\theta_q \rightarrow$  updated by backprop

Object detection	fine-tuned on PASCAL VOC		
pre-train	AP <sub>50</sub>	AP	AP <sub>75</sub>
random init.	60.2	33.8	33.1
super. IN-1M	81.3	53.5	58.8
<b>MoCo</b> IN-1M	81.5 (+0.2)	55.9 ( <b>+2.4</b> )	62.6 ( <b>+3.8</b> )
<b>MoCo</b> IG-1B	82.2 ( <b>+0.9</b> )	57.2 ( <b>+3.7</b> )	63.7 ( <b>+4.9</b> )

Faster R-CNN

ImageNet-1M (IN-1M), Instagram-1B (IG-1B), super. (supervised)

## 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: Nx1
    k = f_k.forward(x_k) # keys: Nx1
    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.



Boulder

# Questions?

---