MI2RL Internal Paper Study

SimCLR v1, v2

Seungjoo Park
(School of Medicine, CHA University)

| Date. 2025.05.14 |

MI2RL

TABLE OF CONTENTS

SimCLR v1, v2

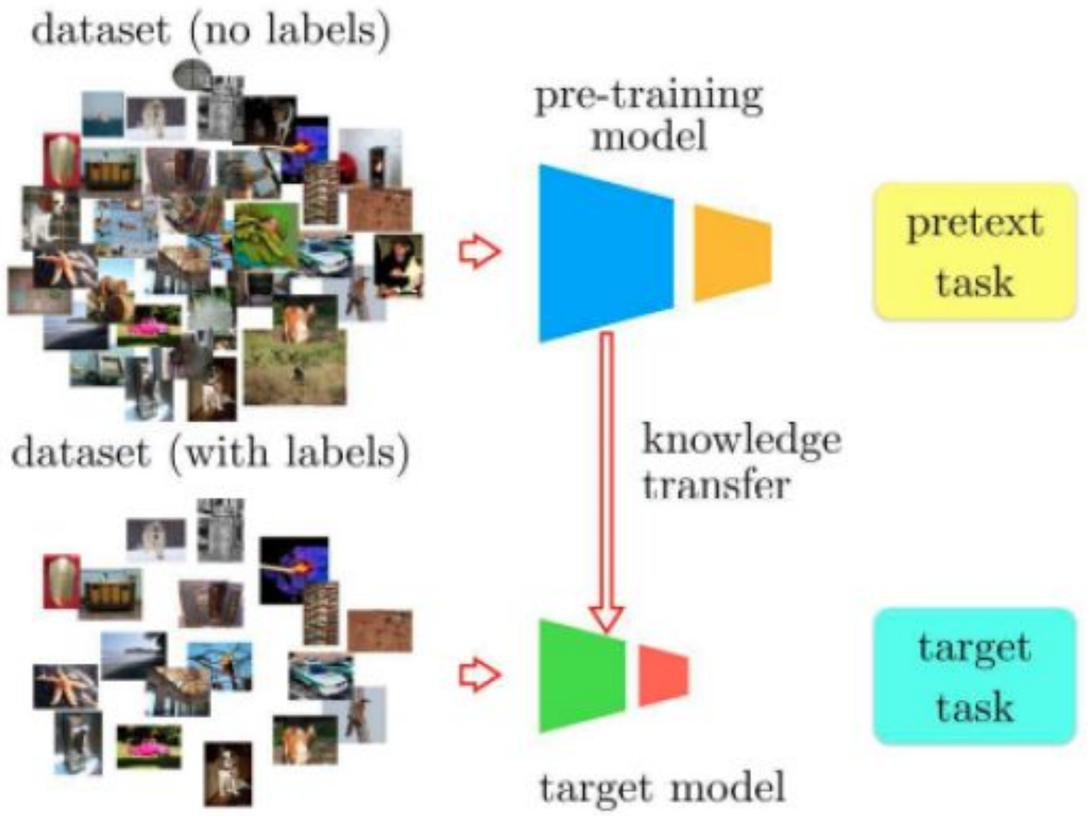
01 | Introduction
- Self-Supervised Learning
- Contrastive Learning

02 | SimCLR v1
- Method
- Results

03 | SimCLR v2
- Method
- Results

01 | Introduction

01 | Self-Supervised Learning

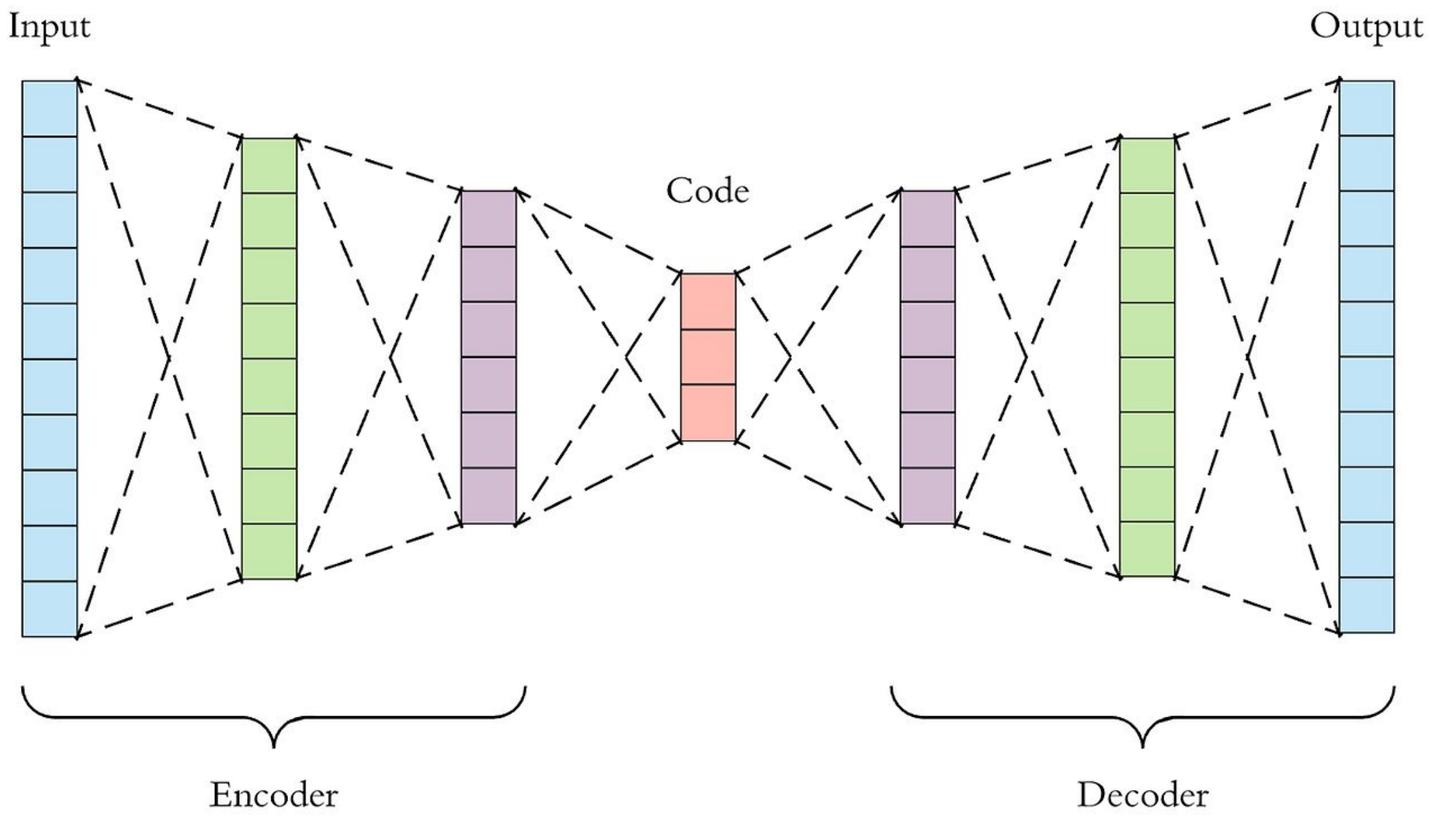


(1) Linear Evaluation
(2) Semi Supervised Learning
(3) Transfer Learning

01 | Introduction

01 | Self-Supervised Learning

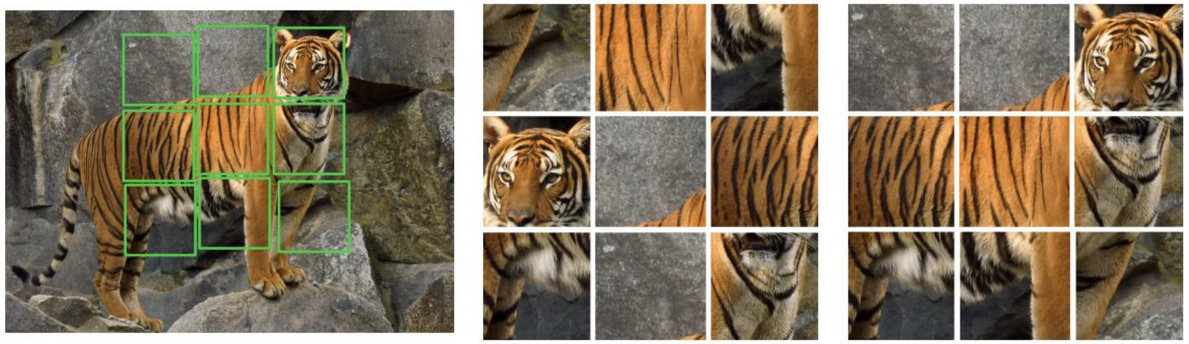
(1) Generative Learning -> computationally expensive



01 | Introduction

01 | Self-Supervised Learning

(2) Discriminative Approach



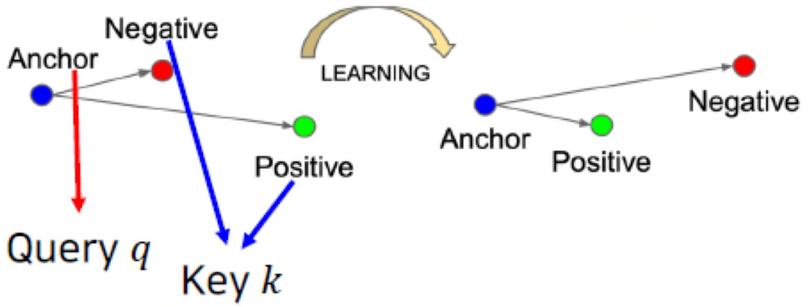
(a) (b) (c)

Fig. 1: Learning image representations by solving Jigsaw puzzles. (a) The image from which the tiles (marked with green lines) are extracted. (b) A puzzle obtained by shuffling the tiles. Some tiles might be directly identifiable as object parts, but others are ambiguous (*e.g.*, have similar patterns) and their identification is much more reliable when all tiles are jointly evaluated. In contrast, with reference to (c), determining the relative position between the central tile and the top two tiles from the left can be very challenging [10].

Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles
https://arxiv.org/pdf/1603.09246

01 | Introduction

02 | Contrastive Learning



$$score(f(x), \underline{f(x^+)}) >> score(f(x), \underline{f(x^-)})$$

positive samples negative samples

Cosine similarity

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

NT-Xent loss function (Normalized temperature scaled cross entropy loss)

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

Softmax

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

01 | Introduction

02 | Contrastive Learning

$$score(f(x), \underbrace{f(x^+)}_{\text{positive samples}}) >> score(f(x), \underbrace{f(x^-)}_{\text{negative samples}})$$

Cosine similarity

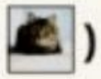
$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

NT-Xent loss function (Normalized temperature scaled cross entropy loss)

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

Softmax

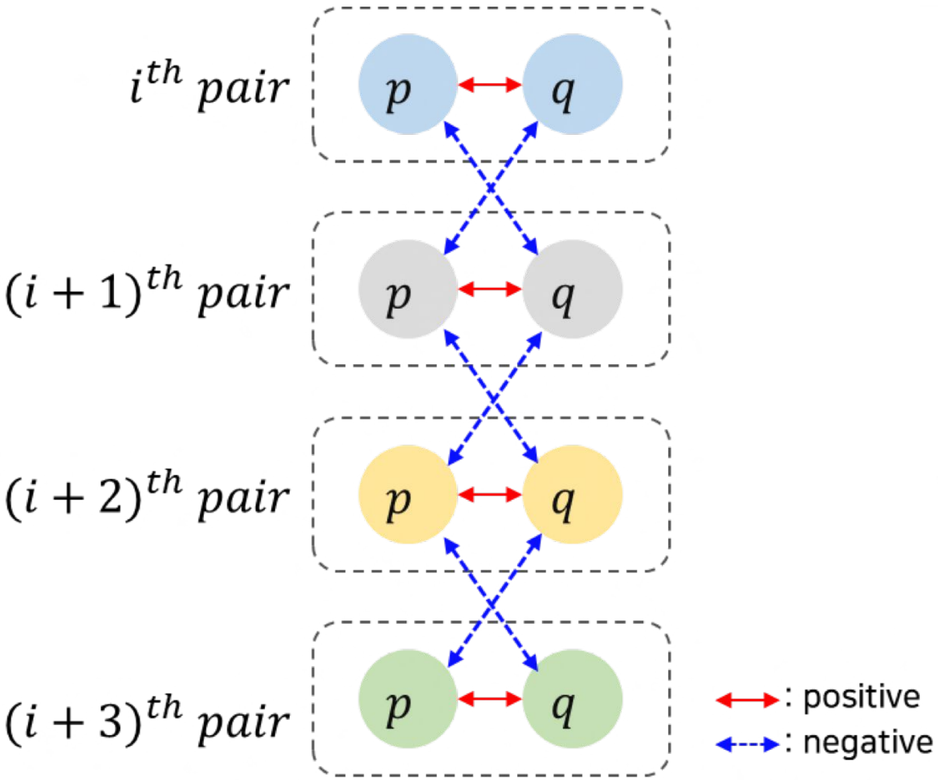
$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

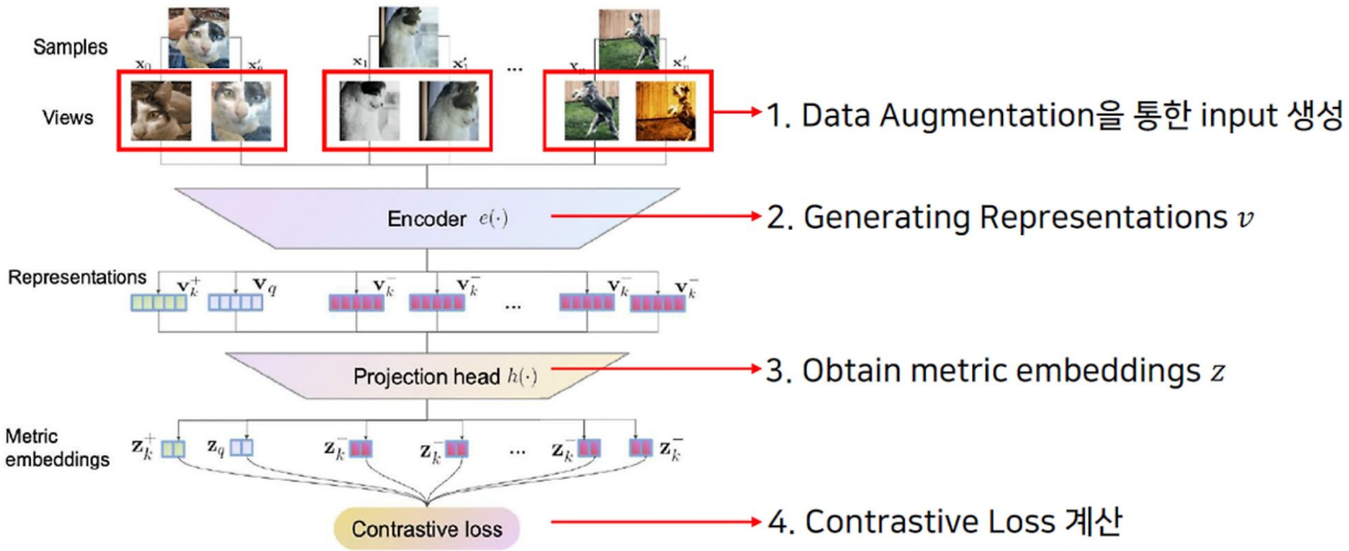
l(,) =-log($\frac{e^{\text{similarity}(\text{ }, \text{  })}}{e^{\text{similarity}(\text{ }, \text{ })} + e^{\text{similarity}(\text{ }, \text{ })} + e^{\text{similarity}(\text{ }, \text{ })}}$)

01 | Introduction

02 | Contrastive Learning

(1) Instance discrimination



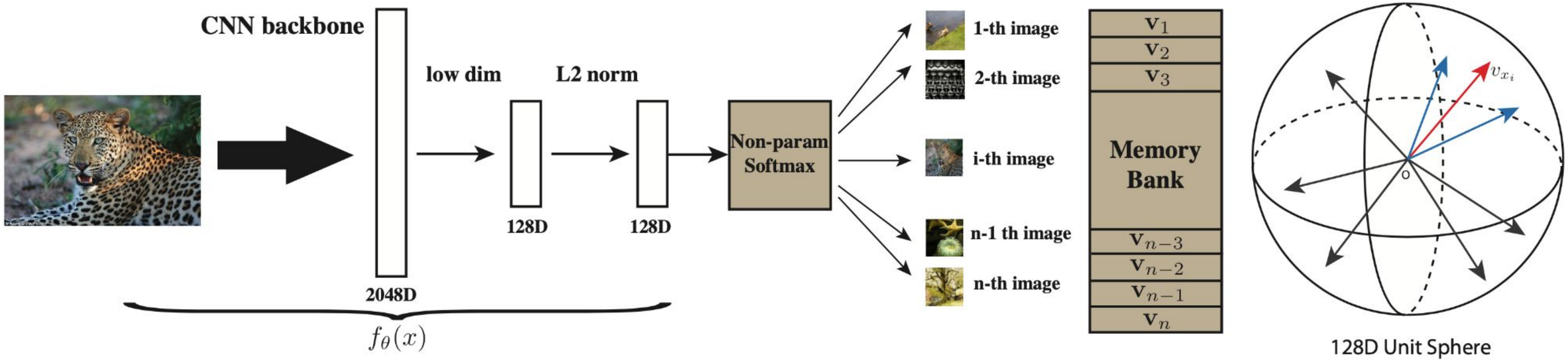


$$\mathcal{L} = -log\frac{exp(\mathbf{z}_i^T \mathbf{z}'_i/\tau)}{\sum_{j=0}^{K} exp(\mathbf{z}_i^T \mathbf{z}'_j/\tau)}$$

01 | Introduction

02 | Contrastive Learning

(2) Memory bank



02 | SimCLR v1

a SIMple framework for Contrastive Learning of visual Representations

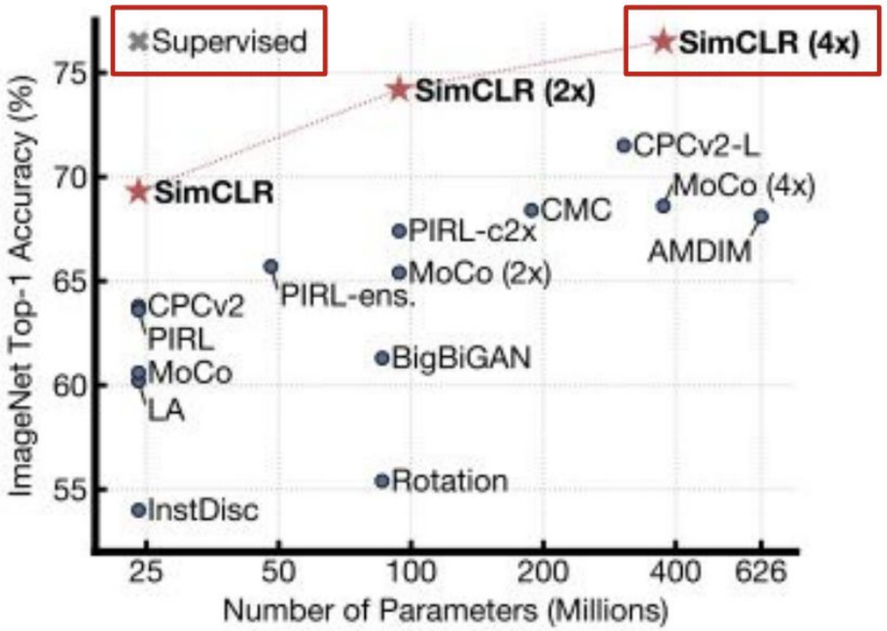


Figure 1. ImageNet Top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50. Our method, SimCLR, is shown in bold.

Major Components of SimCLR

1. composition of **data augmentation**
2. Introducing a learnable **nonlinear transformation**
3. benefits of **larger batch size** and more **training step**

02 | SimCLR v1

01 | Framework

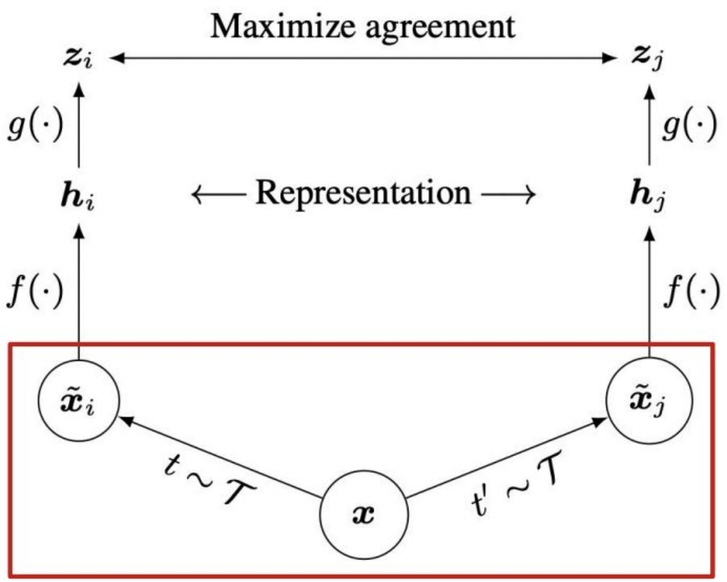


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

Positives : Same image with Difference augmentation → 2n

Negatives : Other images -> 2(n-1)

02 | SimCLR v1

01 | Framework

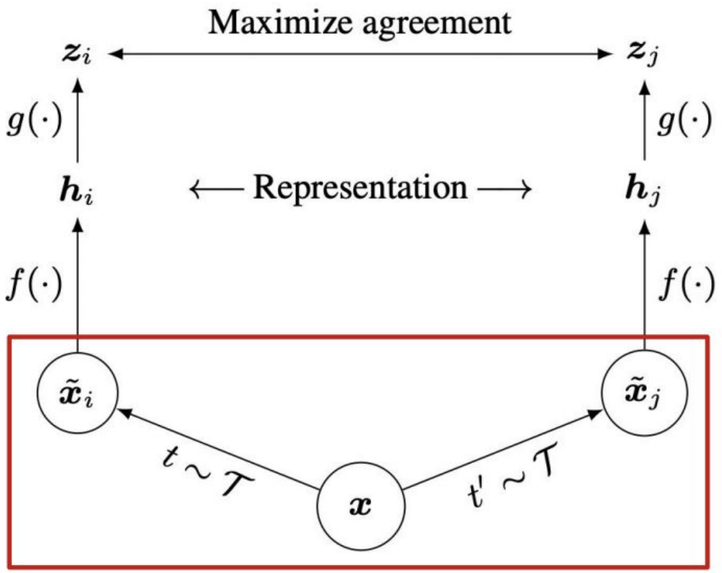


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

1. A **stochastic data augmentation module** that **transforms** any given **data example randomly** resulting in two correlated views of the same sample.

2. A neural network base encoder f(·) that extracts representation vectors from augmented data examples.

3. A NN projection head g(·) that maps representations to the space where contrastive loss is applied.

4. A contrastive loss function defined for a contrastive prediction task.

02 | SimCLR v1

01 | Framework

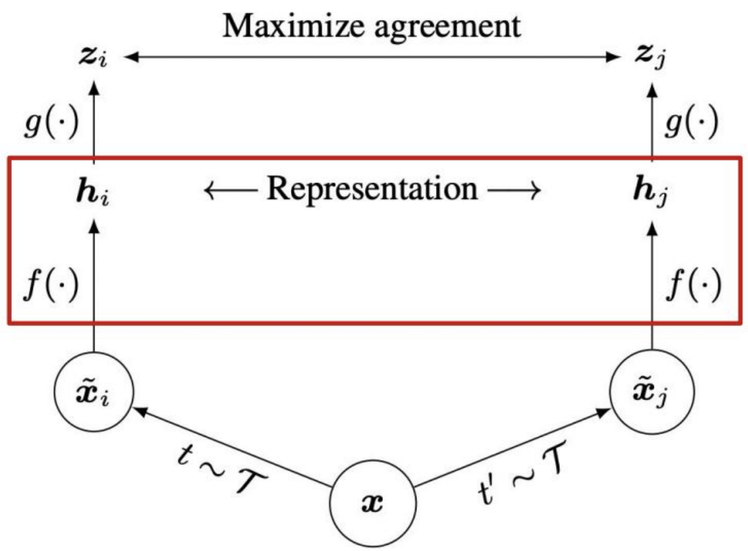


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

1. A stochastic data augmentation module that transforms any given data example randomly resulting in two correlated views of the same sample.

2. A neural network base **encoder f(·)** that **extracts representation vectors** from augmented data examples.

3. A NN projection head g(·) that maps representations to the space where contrastive loss is applied.

4. A contrastive loss function defined for a contrastive prediction task.

02 | SimCLR v1

01 | Framework

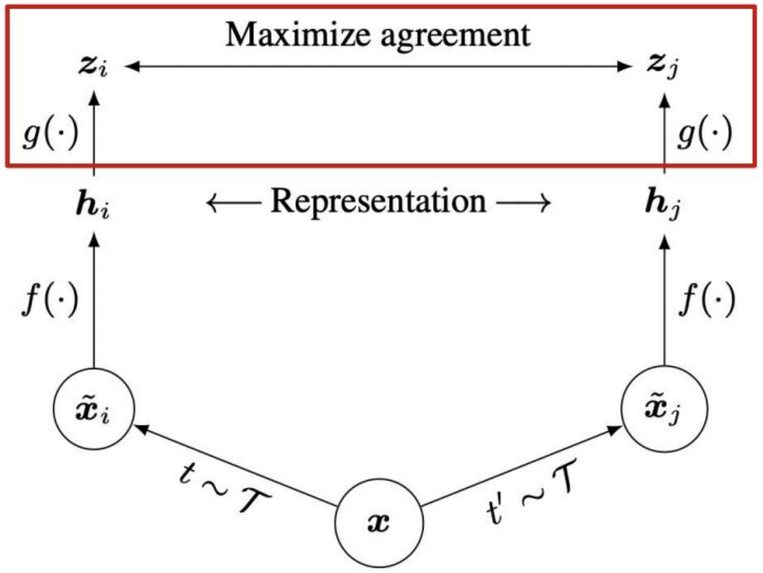


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

1. A stochastic data augmentation module that transforms any given data example randomly resulting in two correlated views of the same sample.

2. A neural network base encoder f(·) that extracts representation vectors from augmented data examples.

3. A NN **projection head g(·)** that **maps representations to the space** where contrastive loss is applied.

4. A contrastive loss function defined for a contrastive prediction task.

02 | SimCLR v1

01 | Framework

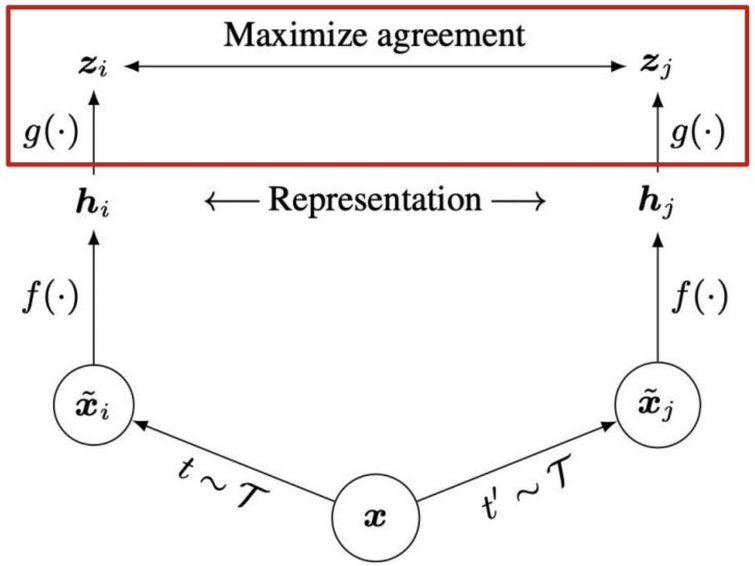


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

1. A stochastic data augmentation module that transforms any given data example randomly resulting in two correlated views of the same sample.

2. A neural network base encoder f(·) that extracts representation vectors from augmented data examples.

3. A NN projection head g(·) that maps representations to the space where contrastive loss is applied.

4. A **contrastive loss function** defined for a contrastive prediction task.

02 | SimCLR v1

01 | Framework

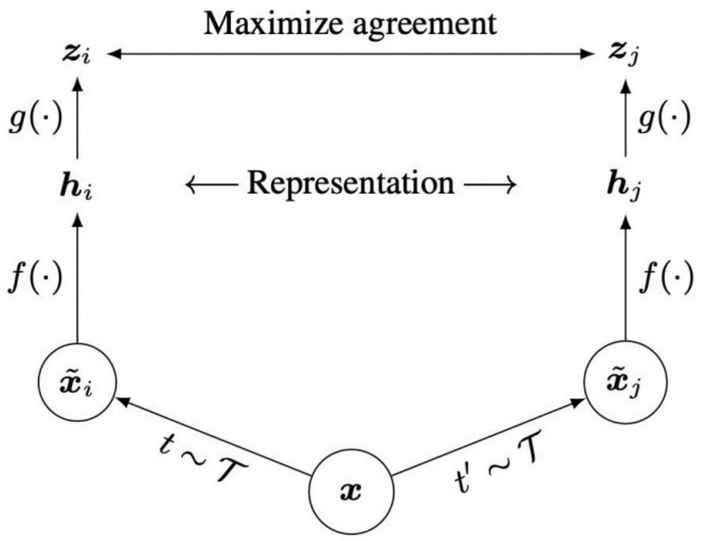


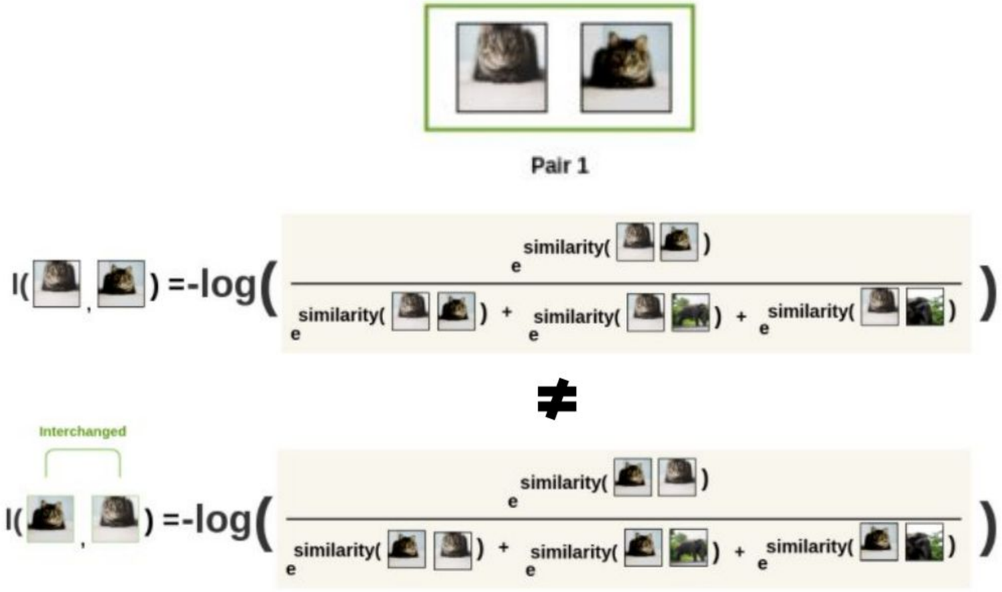
Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

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
1. $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
2. $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
3. $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # 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**
4. $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
5. **define** $\ell(i,j)$ **as** $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
6. $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

02 | SimCLR v1

01 | Framework



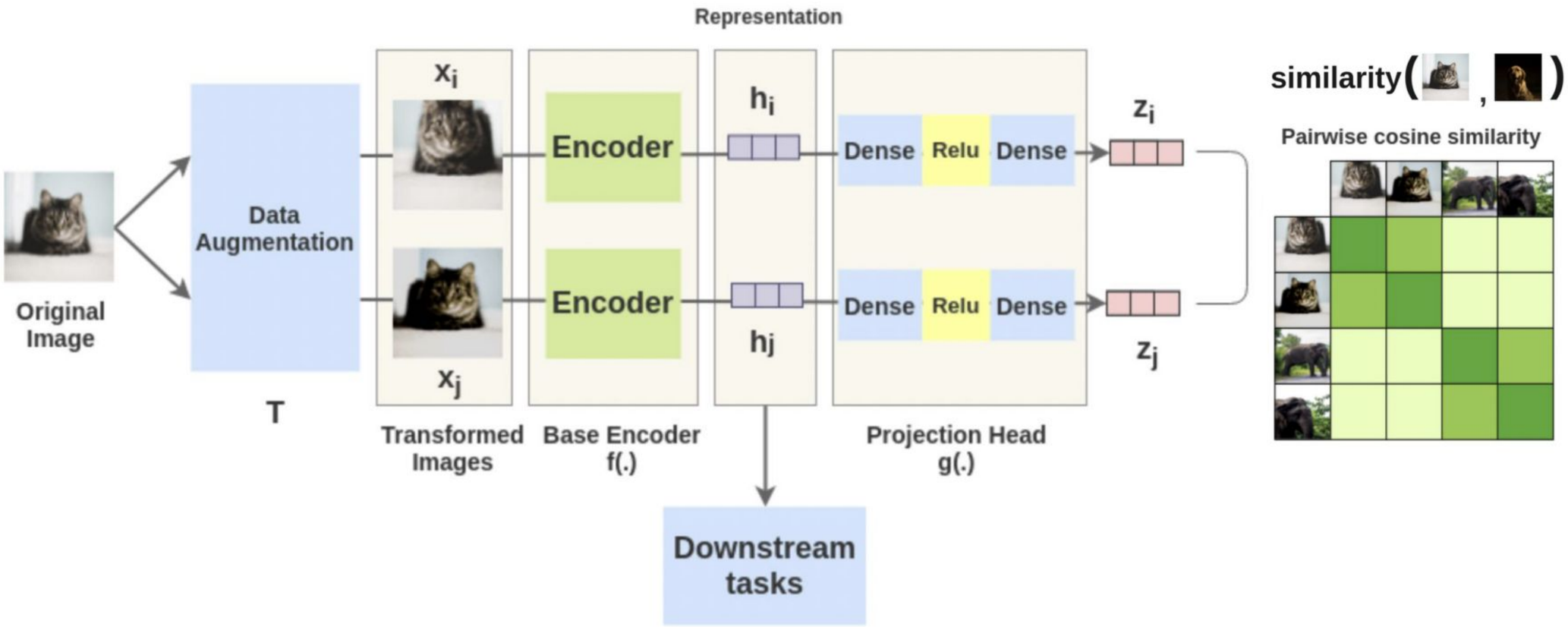
$$L = \frac{1}{2N} \sum_{k=1}^{N} [l(2k-1, 2k) + l(2k, 2k-1)]$$

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
1. $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
2. $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
3. $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # 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**
4. $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
5. **define** $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
6. $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

02 | SimCLR v1

01 | Framework



02 | SimCLR v1

01 | Framework

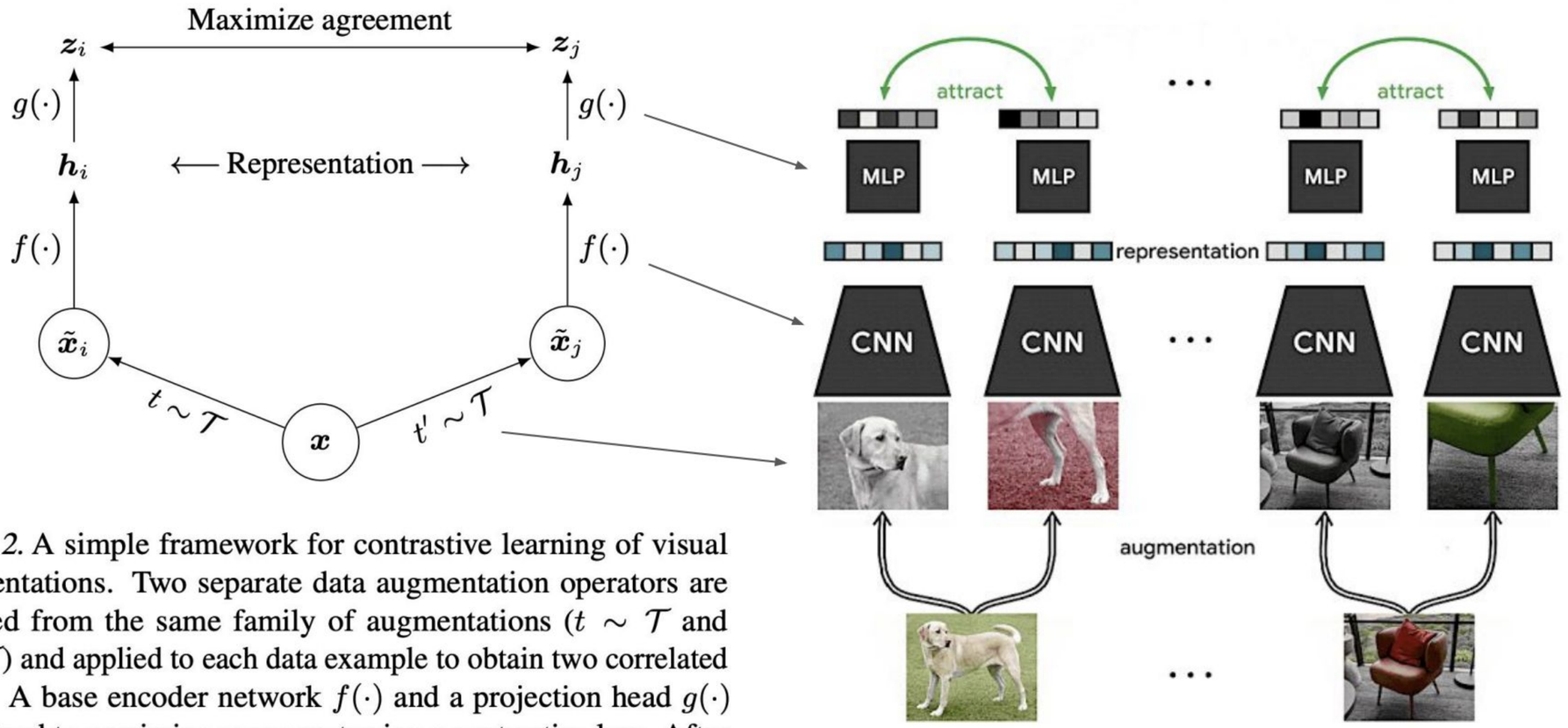
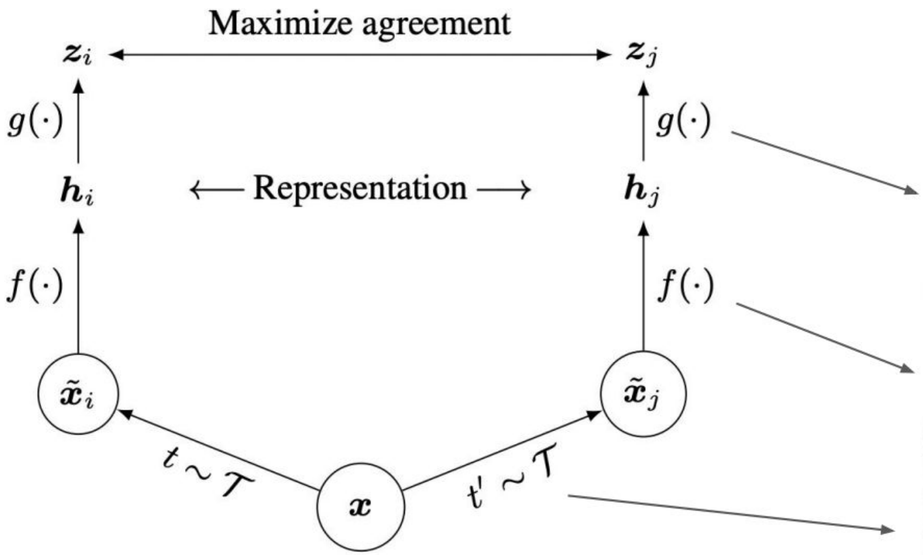


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

02 | SimCLR v1

01 | Framework



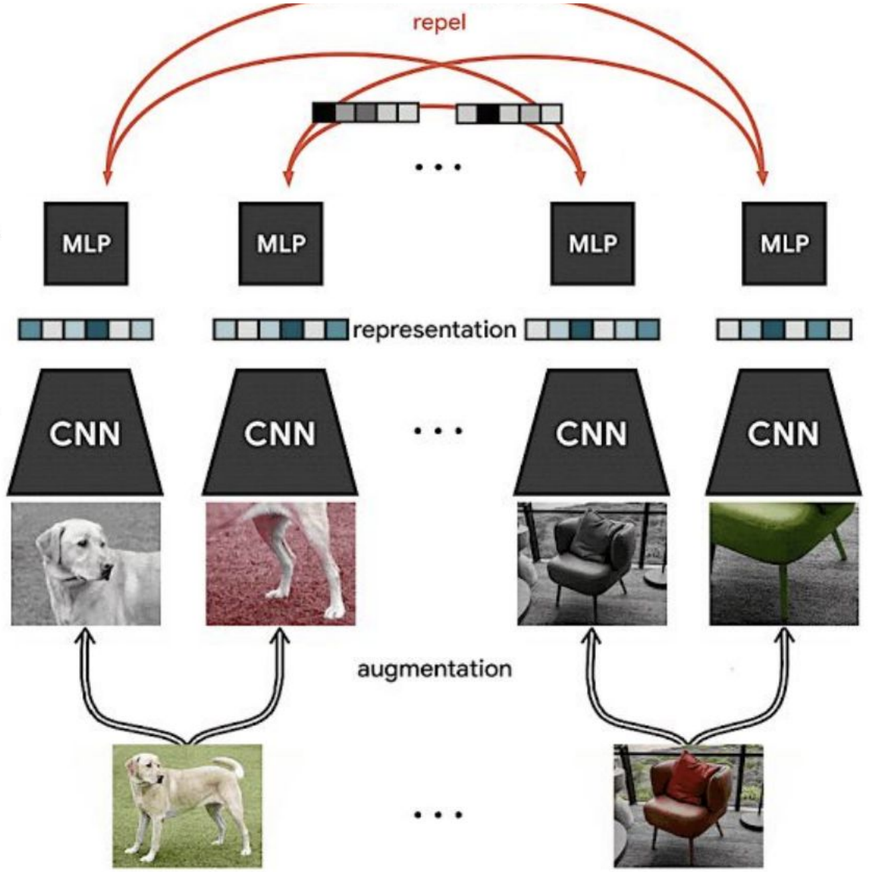


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

02 | SimCLR v1

01 | Framework

How to solve?
In case of MoCo: Memory Queue (First in First Out)
In case of SimCLR: Large Batch Size → Vary the training batch size N from 256 to 8,192

Dataset

- ImageNet 2012 dataset

Training

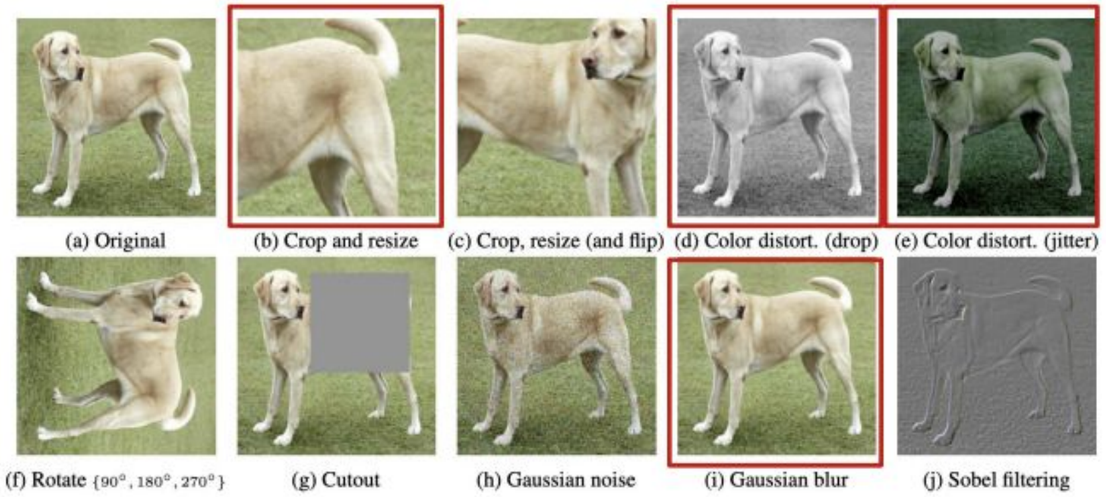
- Batch size: 256 ~ 8,192
  8,192 batch 기준 positive 하나를 제외한 나머지 → negative samples
  paired augmentation 이기 때문에 positive sample: 2 | negative sample: 16,382
- Optimizer : LARS (for all batch size)
- 32 to 128 TPU takes ~ 1.5 hrs to train ResNet50 with 4096 batch size for 100 epochs
- Aggregating BN means and variance over all device during training
- To evaluation the learned representations, linear evaluation protocol is used.

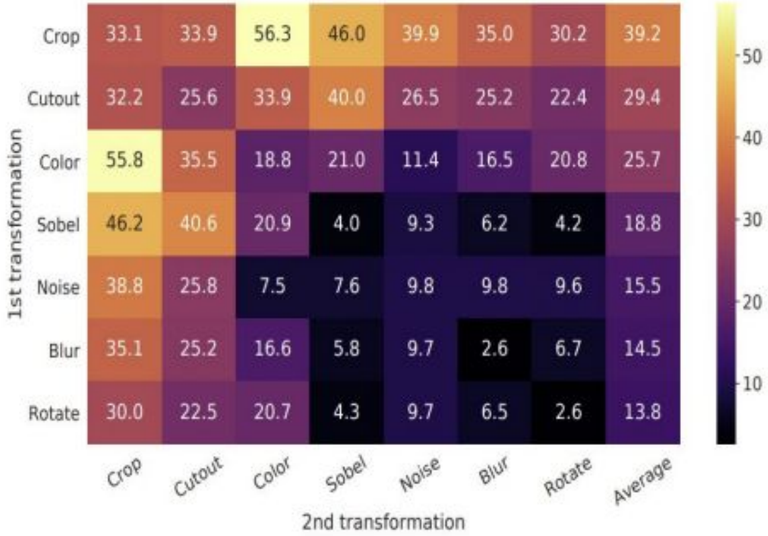
02 | SimCLR v1

02

| Augmentation

(1) Composition of multiple data augmentation operations is crucial





One composition of augmentations stands out:
random cropping and **random color distortion**

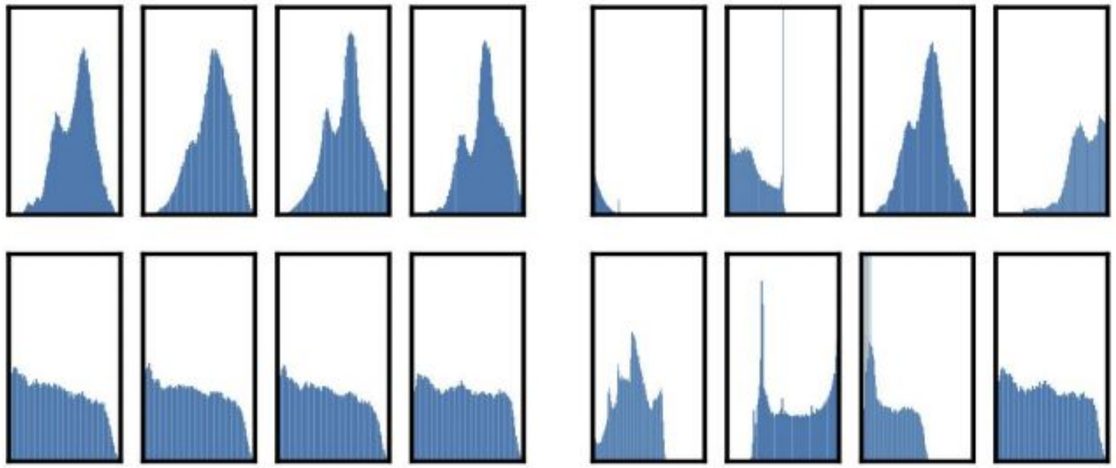
Observe that no single transformation suffices to learn good representations

02 | SimCLR v1

02

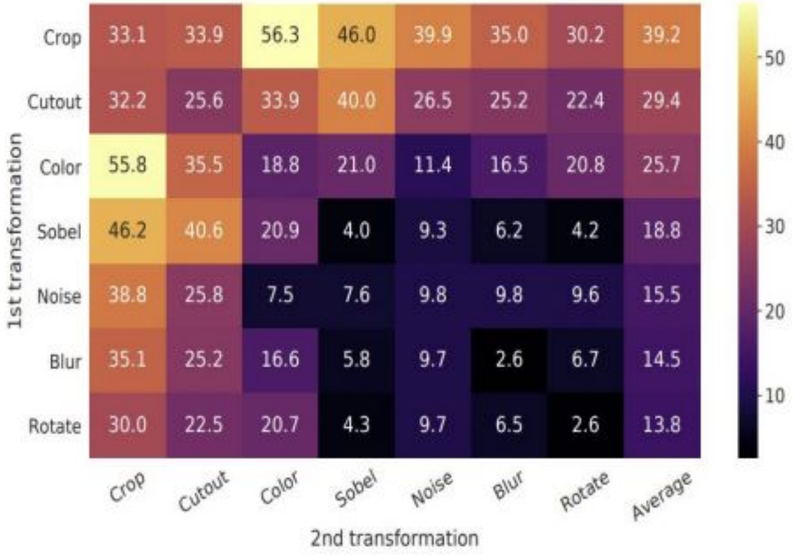
| Augmentation

(1) Composition of multiple data augmentation operations is crucial



(a) Without color distortion. (b) With color distortion.

One composition of augmentations stands out:
random cropping and **random color distortion**



Observe that no single transformation suffices to learn good representations

02 | SimCLR v1

02

| Augmentation

(1) Composition of multiple data augmentation operations is crucial

Methods	Color distortion strength 1/8	1/4	1/2	1	1 (+Blur)	AutoAug
SimCLR	59.6	61.0	62.6	63.2	64.5	61.1
Supervised	77.0	76.7	76.5	75.7	75.4	77.1

Unsupervised contrastive learning benefits from stronger **color data augmentation** than supervised learning

Observe that no single transformation suffices to learn good representations

02 | SimCLR v1

03 | Encoder head

(2) Introducing a learnable nonlinear transformation (Relu)

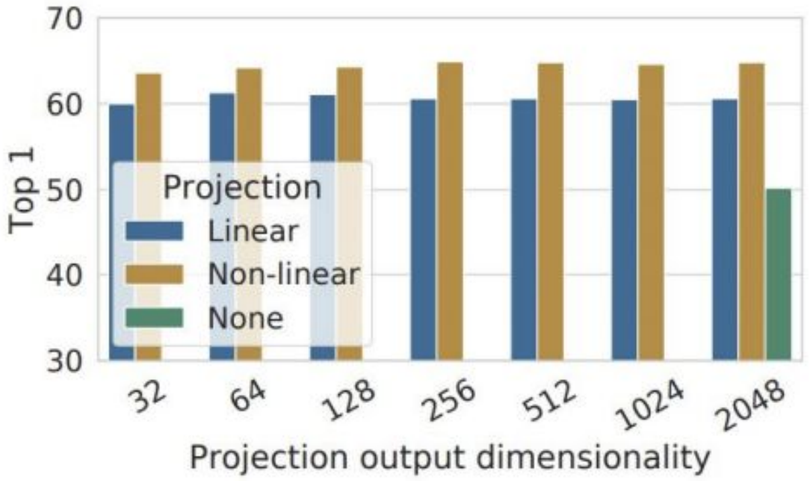
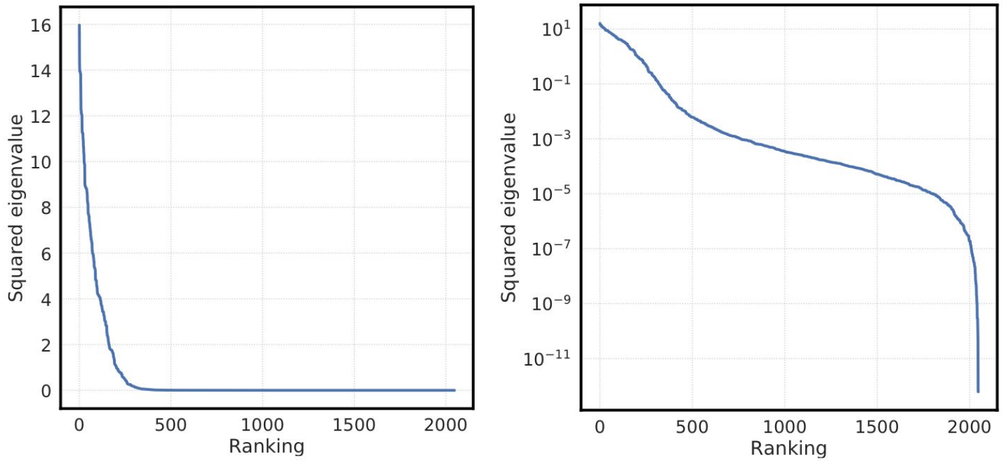


Figure 8. Linear evaluation of representations with different projection heads $g(\cdot)$ and various dimensions of $\boldsymbol{z} = g(\boldsymbol{h})$. The representation $\boldsymbol{h}$ (before projection) is 2048-dimensional here.



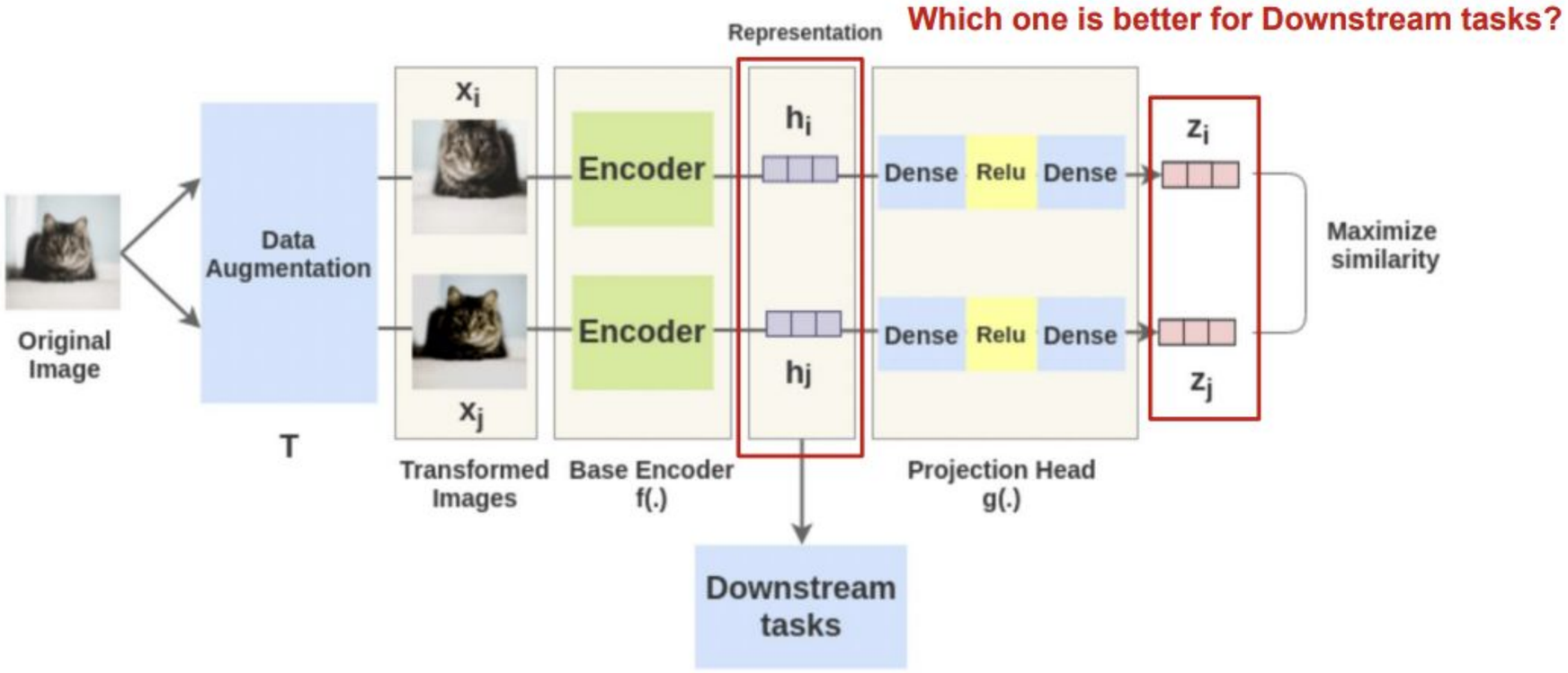
(a) Y-axis in uniform scale. (b) Y-axis in log scale.

Figure B.3. Squared real eigenvalue distribution of linear projection matrix $W \in R^{2048\times2048}$ used to compute $g(\boldsymbol{h}) = W\boldsymbol{h}$.

02 | SimCLR v1

03 | Encoder head

(2) Introducing a learnable nonlinear transformation (Relu)



02 | SimCLR v1

03 | Encoder head

(2) Introducing a learnable nonlinear transformation (Relu)

Shows that the hidden layer before the projection head is a better representation than the layer after.

Even when **nonlinear projection** is used, the layer before the projection head, h, is **still much better (>10%) than the layer after, z = g(h)**.

z = g(h) is trained to be invariant to data transformation. Thus, **g can remove information** that may be useful for the downstream task, such as the color or orientation of objects.

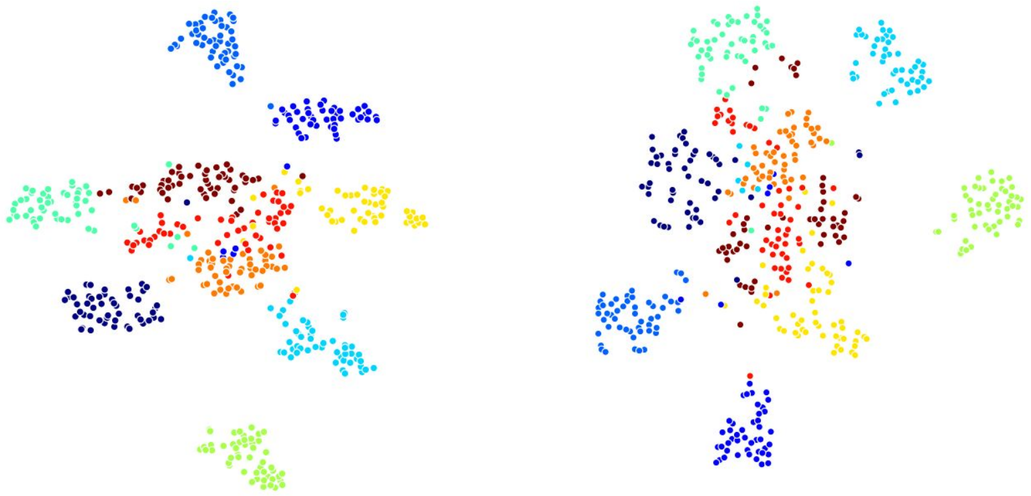
What to predict?	Random guess	Representation $\boldsymbol{h}$	Representation $g(\boldsymbol{h})$
Color vs grayscale	80	99.3	97.4
Rotation	25	67.6	25.6
Orig. vs corrupted	50	99.5	59.6
Orig. vs Sobel filtered	50	96.6	56.3

Accuracy of training additional MPs on different representations to predict the transformation applied.

02 | SimCLR v1

03 | Encoder head

(2) Introducing a learnable nonlinear transformation (Relu)



(a) $\boldsymbol{h}$ (b) $\boldsymbol{z} = g(\boldsymbol{h})$

Figure B.4. t-SNE visualizations of hidden vectors of images from a randomly selected 10 classes in the validation set.

02 | SimCLR v1

04 | Loss functions and Batch size

(3) Benefits of Representations learning with contrastive cross entropy and temperature

Name	Negative loss function	Gradient w.r.t. $\boldsymbol{u}$
NT-Xent	$\boldsymbol{u}^T\boldsymbol{v}^+/\tau - \log \sum_{\boldsymbol{v}\in\{\boldsymbol{v}^+,\boldsymbol{v}^-\}} \exp(\boldsymbol{u}^T\boldsymbol{v}/\tau)$	$(1 - \frac{\exp(\boldsymbol{u}^T\boldsymbol{v}^+/\tau)}{Z(\boldsymbol{u})})/\tau\boldsymbol{v}^+ - \sum_{\boldsymbol{v}^-} \frac{\exp(\boldsymbol{u}^T\boldsymbol{v}^-/\tau)}{Z(\boldsymbol{u})}/\tau\boldsymbol{v}^-$
NT-Logistic	$\log\sigma(\boldsymbol{u}^T\boldsymbol{v}^+/\tau) + \log\sigma(-\boldsymbol{u}^T\boldsymbol{v}^-/\tau)$	$(\sigma(-\boldsymbol{u}^T\boldsymbol{v}^+/\tau))/\tau\boldsymbol{v}^+ - \sigma(\boldsymbol{u}^T\boldsymbol{v}^-/\tau)/\tau\boldsymbol{v}^-$
Margin Triplet	$-\max(\boldsymbol{u}^T\boldsymbol{v}^- - \boldsymbol{u}^T\boldsymbol{v}^+ + m, 0)$	$\boldsymbol{v}^+ - \boldsymbol{v}^-$ if $\boldsymbol{u}^T\boldsymbol{v}^+ - \boldsymbol{u}^T\boldsymbol{v}^- < m$ else $\boldsymbol{0}$

Table 2. Negative loss functions and their gradients. All input vectors, i.e. $\boldsymbol{u}, \boldsymbol{v}^+, \boldsymbol{v}^-$, are ℓ_2 normalized. NT-Xent is an abbreviation for "Normalized Temperature-scaled Cross Entropy". Different loss functions impose different weightings of positive and negative examples.

To make the comparisons fair, we use the same `2 normalization for all loss functions, and we tune the hyperparameters, and report their best results.

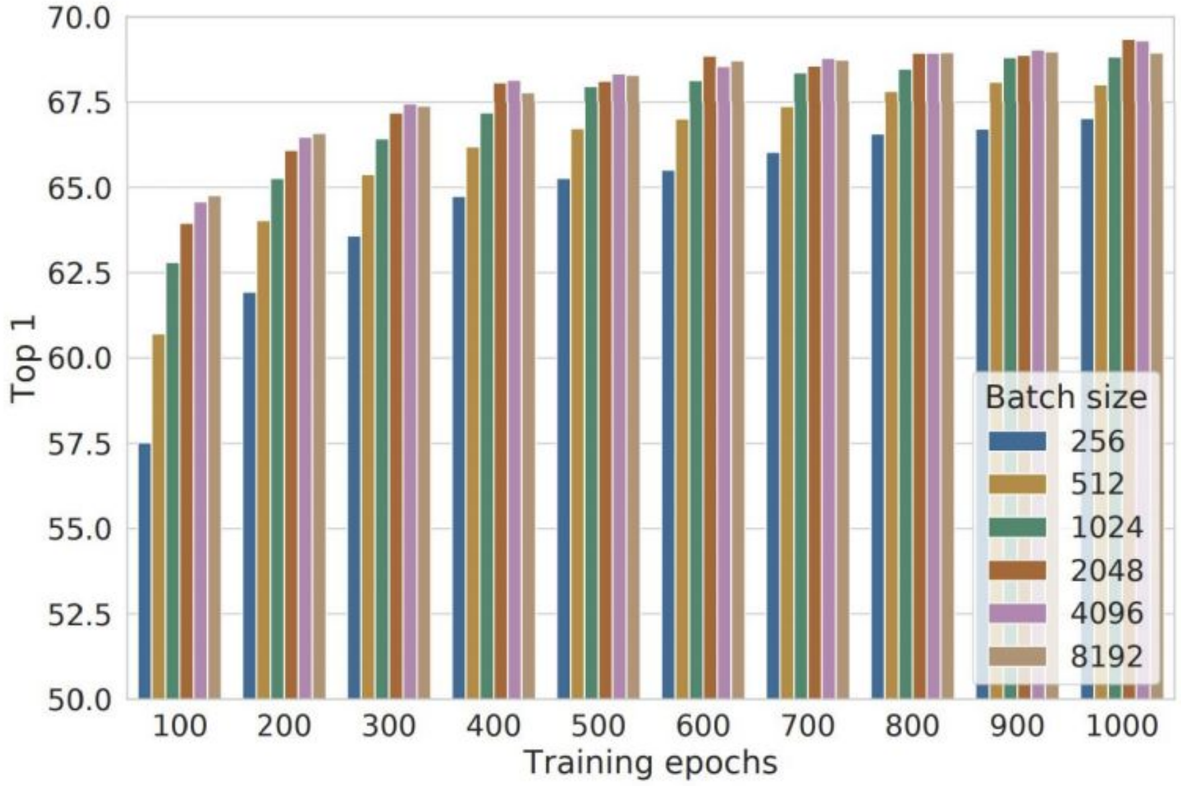
Margin	NT-Logi.	Margin (sh)	NT-Logi.(sh)	NT-Xent
50.9	51.6	57.5	57.9	63.9

While (semi-hard) negative mining helps, the **best result is still much worse than our default NT-Xent loss**.

02 | SimCLR v1

04 | Loss functions and Batch size

(4) Benefits of Larger Batch sizes and longer training



02 | SimCLR v1

05 | Results

A Simple Framework for Contrastive Learning of Visual Representations

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
SimCLR (ours)	**76.9**	**95.3**	80.2	48.4	**65.9**	60.0	61.2	**84.2**	**78.9**	89.2	**93.9**	**95.0**
Supervised	75.2	**95.7**	**81.2**	**56.4**	64.9	**68.8**	**63.8**	83.8	**78.7**	**92.3**	**94.1**	94.2
Fine-tuned:												
SimCLR (ours)	**89.4**	**98.6**	**89.0**	**78.2**	**68.1**	**92.1**	**87.0**	**86.6**	**77.8**	92.1	**94.1**	97.6
Supervised	88.7	98.3	**88.7**	**77.8**	67.0	91.4	**88.0**	86.5	**78.8**	**93.2**	**94.2**	**98.0**
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

Table 8. Comparison of transfer learning performance of our self-supervised approach with supervised baselines across 12 natural image classification datasets, for ResNet-50 ($4\times$) models pretrained on ImageNet. Results not significantly worse than the best ($p > 0.05$, permutation test) are shown in bold. See Appendix B.8 for experimental details and results with standard ResNet-50.

03 | SimCLR v2

Big Self-Supervised Models are Strong Semi-Supervised Learners

(1) Resnet + simCLR v2 → unsupervised learning

(2) few labeled data → supervised fine-tuning

(3) unlabeled data distillation

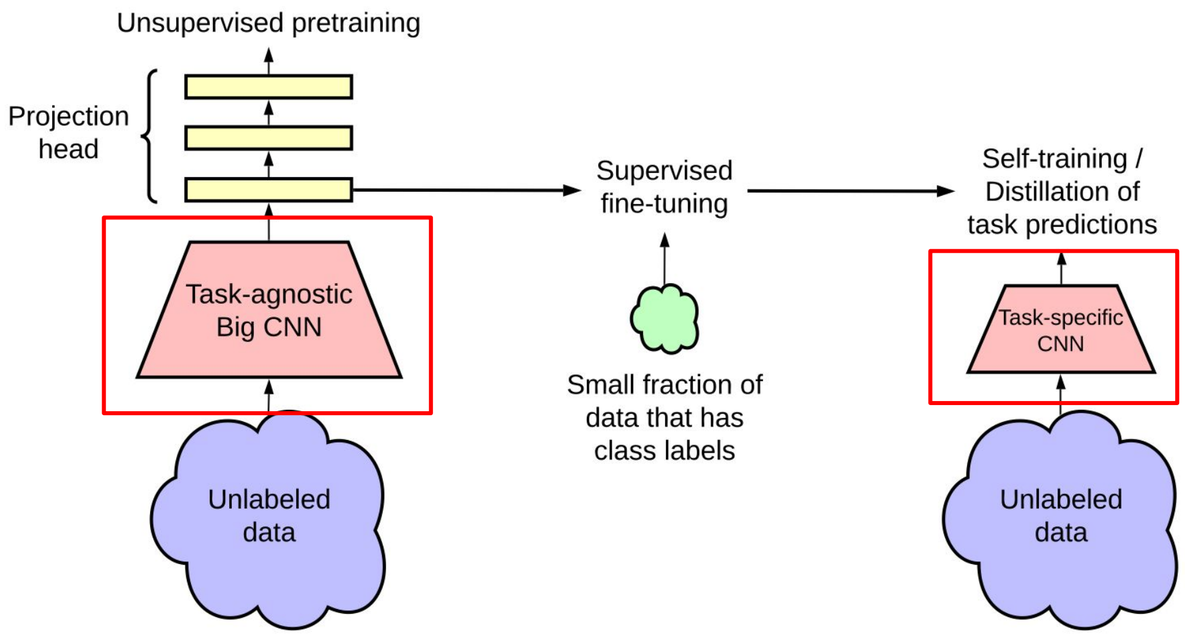
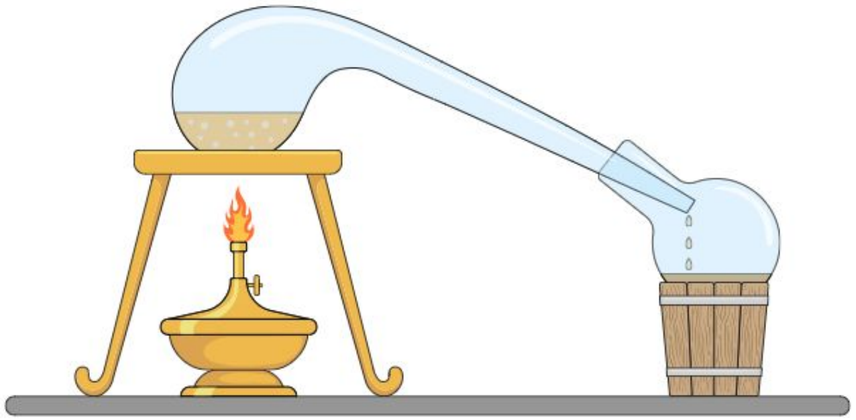
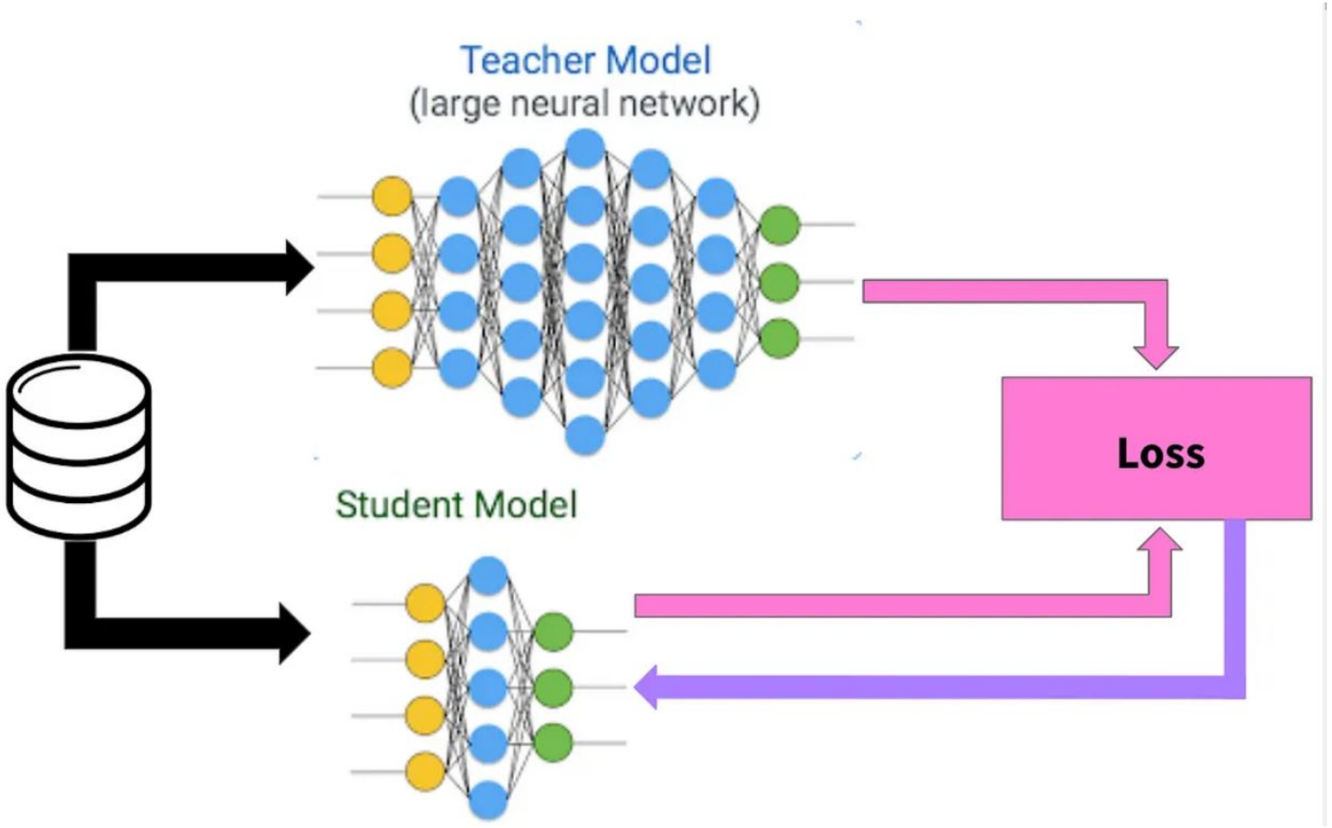


Figure 3: The proposed semi-supervised learning framework leverages unlabeled data in two ways: (1) task-agnostic use in unsupervised pretraining, and (2) task-specific use in self-training / distillation.

03 | SimCLR v2

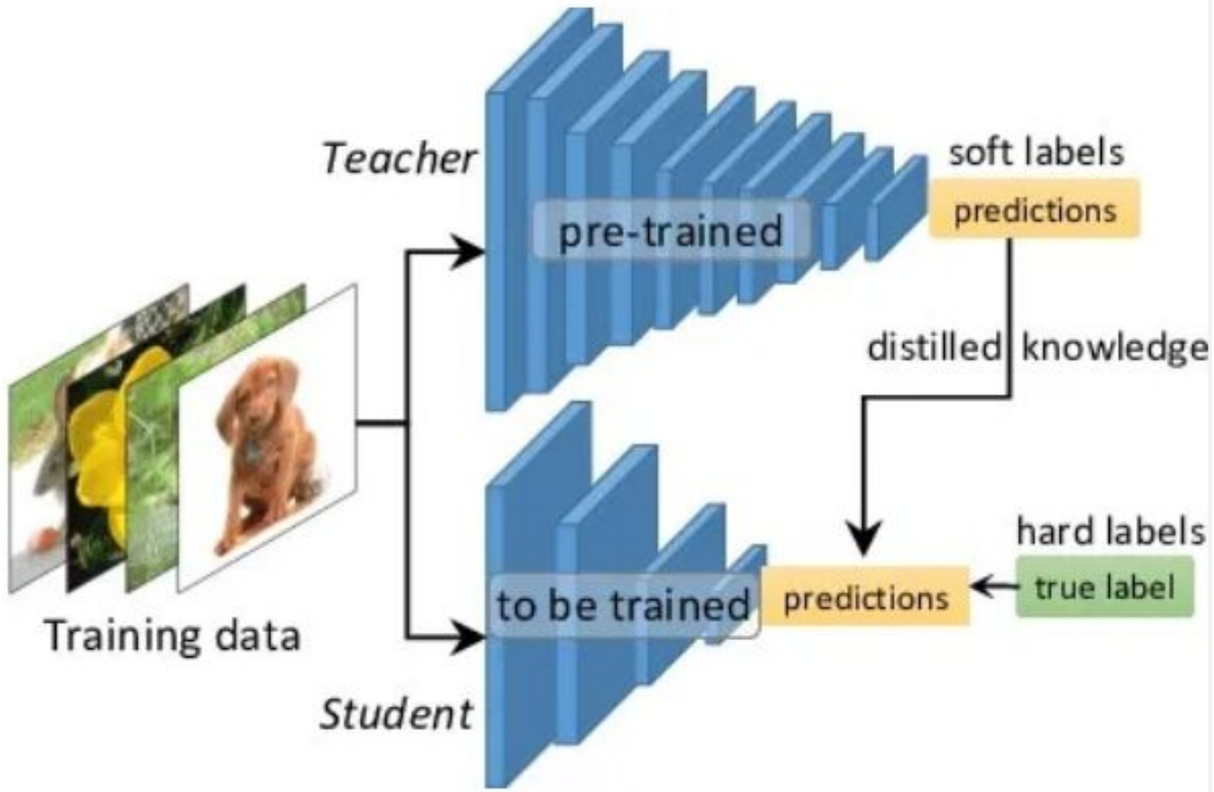
Big Self-Supervised Models are Strong Semi-Supervised Learners





03 | SimCLR v2

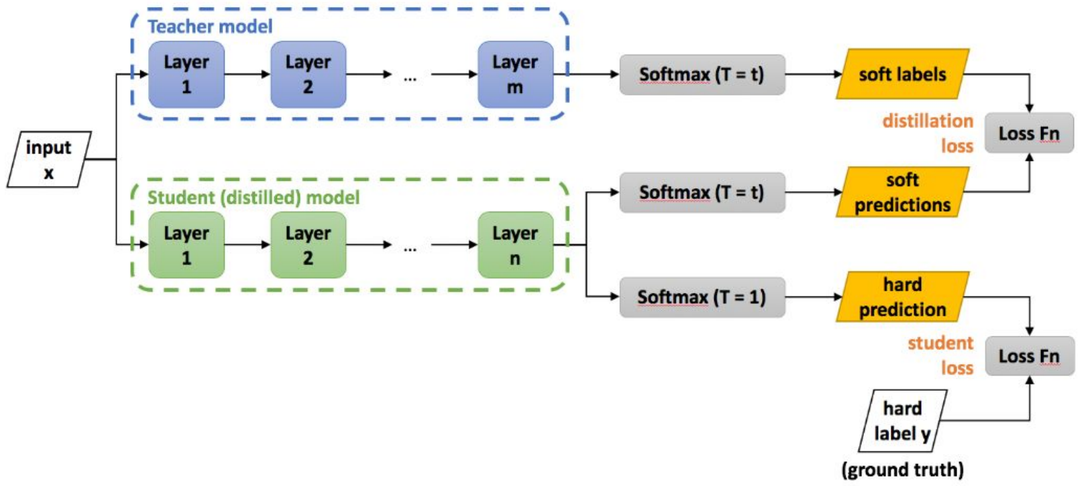
Big Self-Supervised Models are Strong Semi-Supervised Learners



cow	dog	cat	car
10^{-6}	.9	.1	10^{-9}

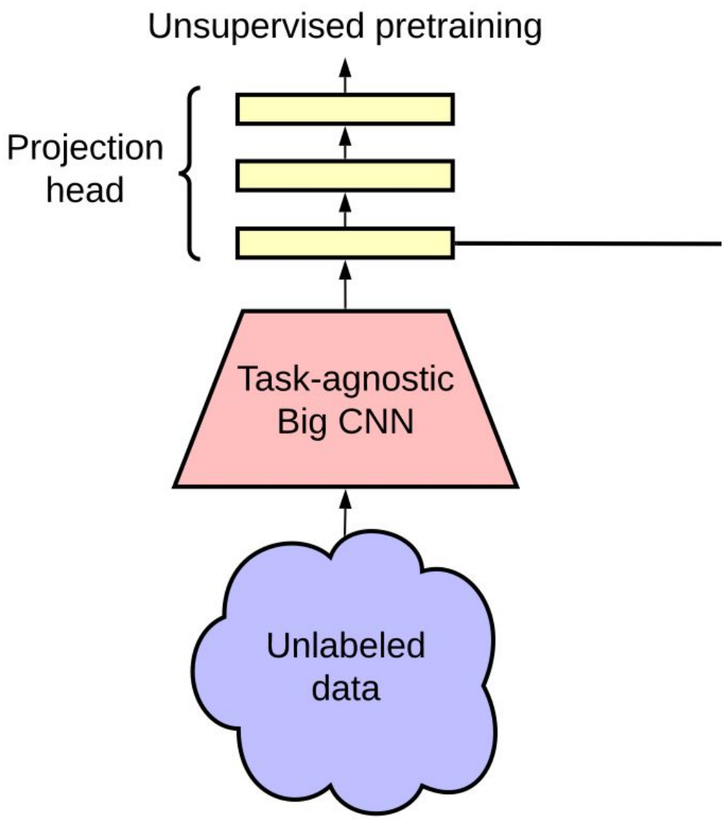
output of geometric ensemble

$$\text{Soft Label}(x)_i = \frac{e^{x_i/T}}{\sum_{j=1}^{N} e^{x_j/T}}$$



03 | SimCLR v2

01 | Framework



(1) Larger Resnet models → deeper but less wide

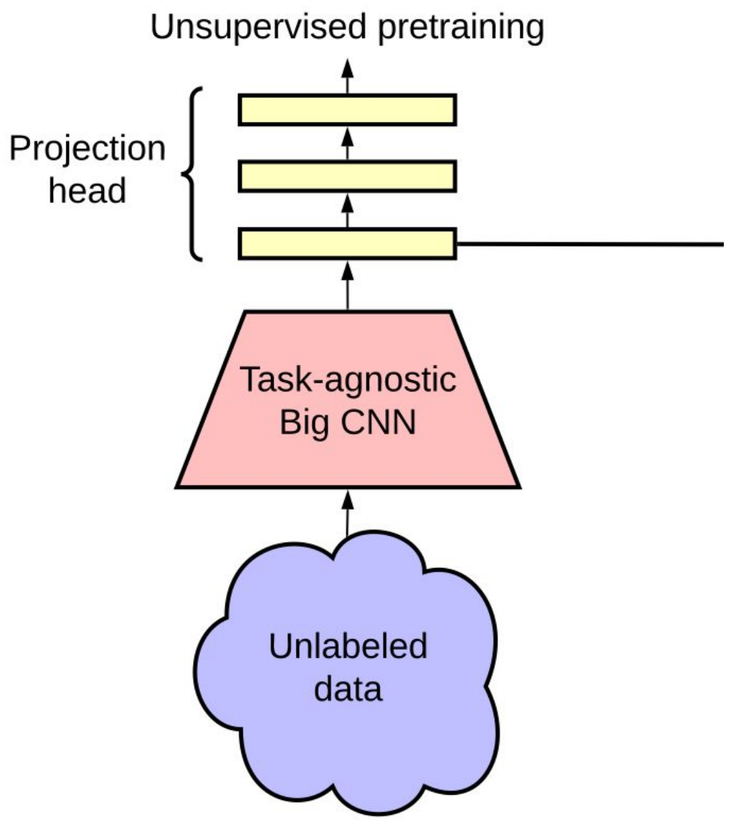
(2) Deeper non-linear network

(3) Moco memory mechanism

03 | SimCLR v2

01 | Framework

(1) Larger Resnet models → deeper but less wide

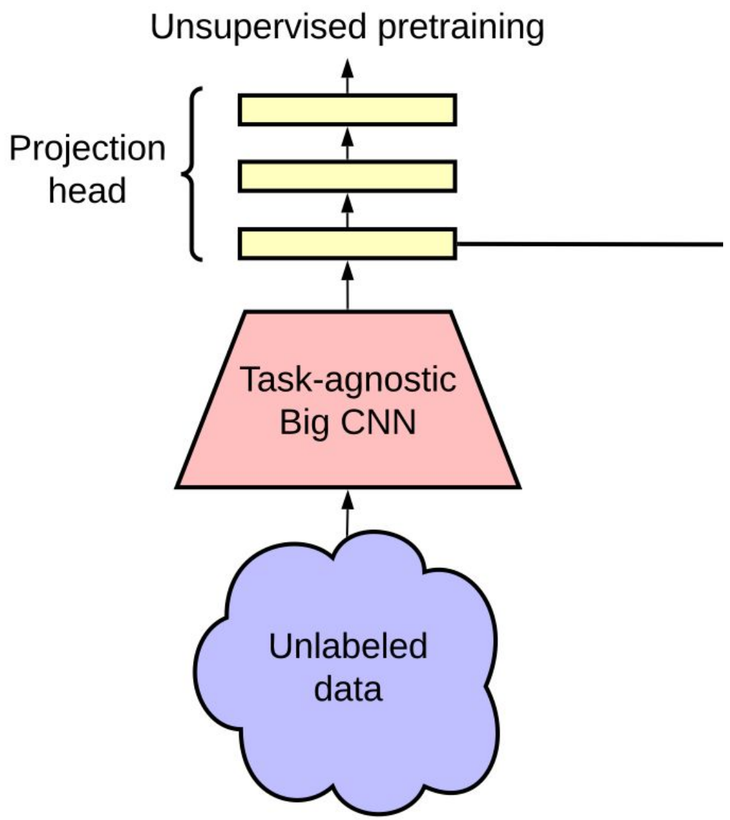


Depth	Width	Use SK [28]	Param (M)	Fine-tuned on 1%	Fine-tuned on 10%	Fine-tuned on 100%	Linear eval	Supervised
50	1×	False	**24**	**57.9**	**68.4**	**76.3**	**71.7**	**76.6**
		True	35	64.5	72.1	78.7	74.6	78.5
	2×	False	94	66.3	73.9	79.1	75.6	77.8
		True	140	70.6	77.0	81.3	77.7	79.3
101	1×	False	43	62.1	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**

03 | SimCLR v2

01 | Framework

(1) Larger Resnet models → deeper but less wide



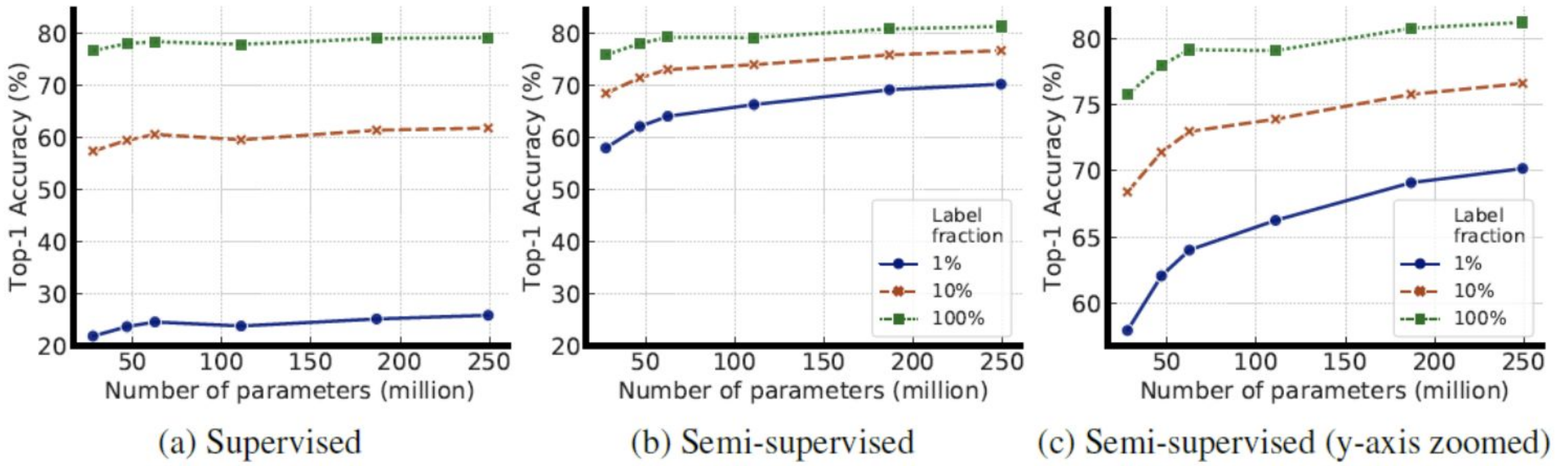
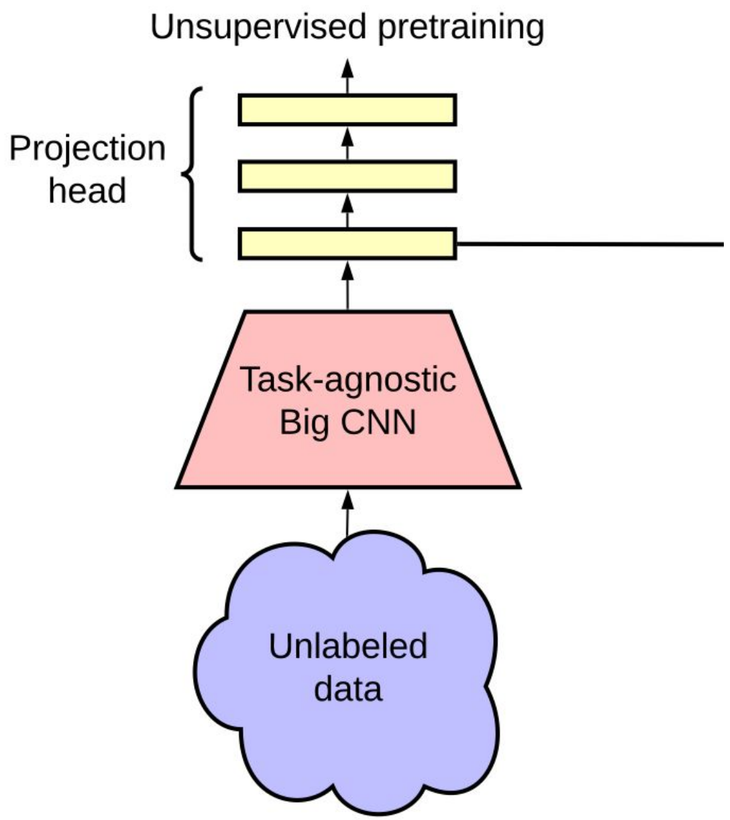


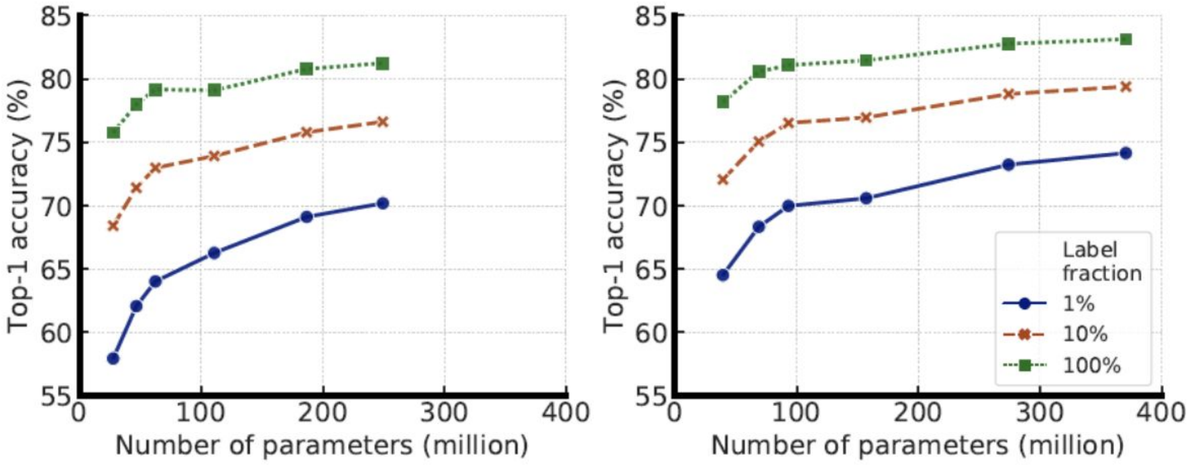
Figure 4: Top-1 accuracy for supervised vs semi-supervised (SimCLRv2 fine-tuned) models of varied sizes on different label fractions. ResNets with depths of 50, 101, 152, width multiplier of 1×, 2× (w/o SK) are presented here. For supervised models on 1%/10% labels, AutoAugment [35] and label smoothing [36] are used. Increasing the size of SimCLRv2 models by 10×, from ResNet-50 to ResNet-152 (2×), improves label efficiency by 10×.

03 | SimCLR v2

01 | Framework

(1) Larger Resnet models → deeper but less wide





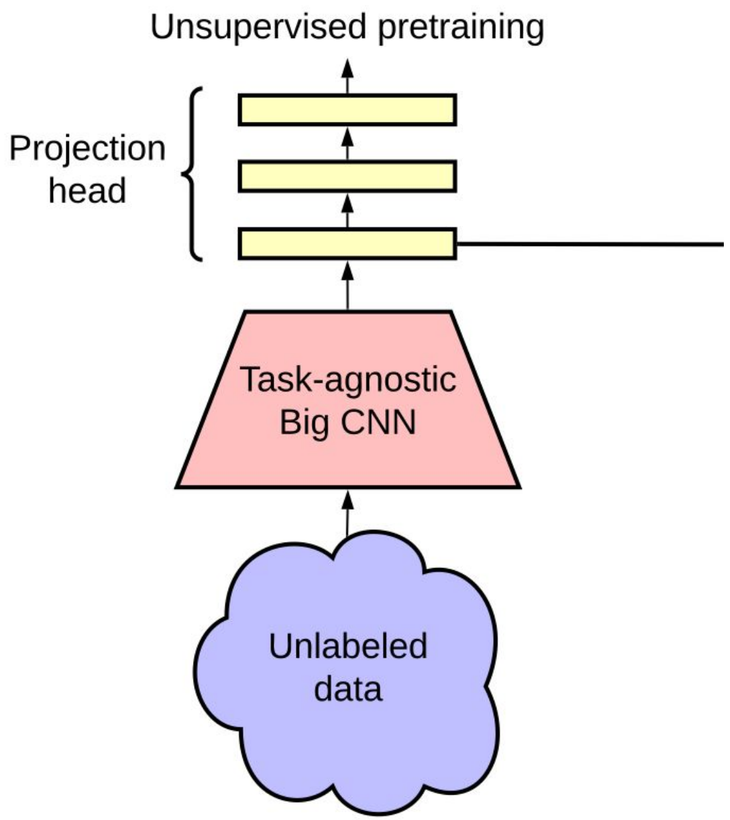
(a) Models without SK [28] (b) Models with SK [28]

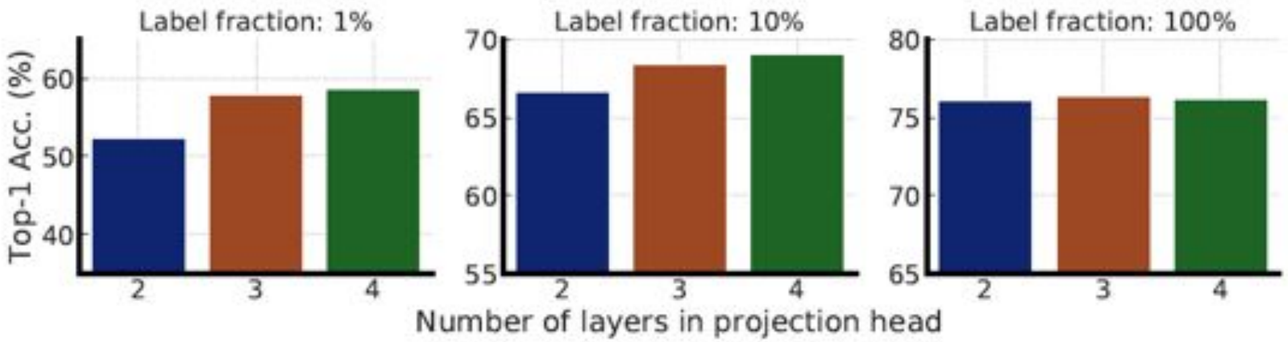
Figure B.1: Top-1 accuracy of fined-tuned SimCLRv2 models of different sizes on three label fractions. ResNets with depth in $\{50, 101, 152\}$, width in $\{1\times, 2\times\}$ are included here. Parameter efficiency also plays an important role. For fine-tuning on 1% of labels, SK is much more efficient.

03 | SimCLR v2

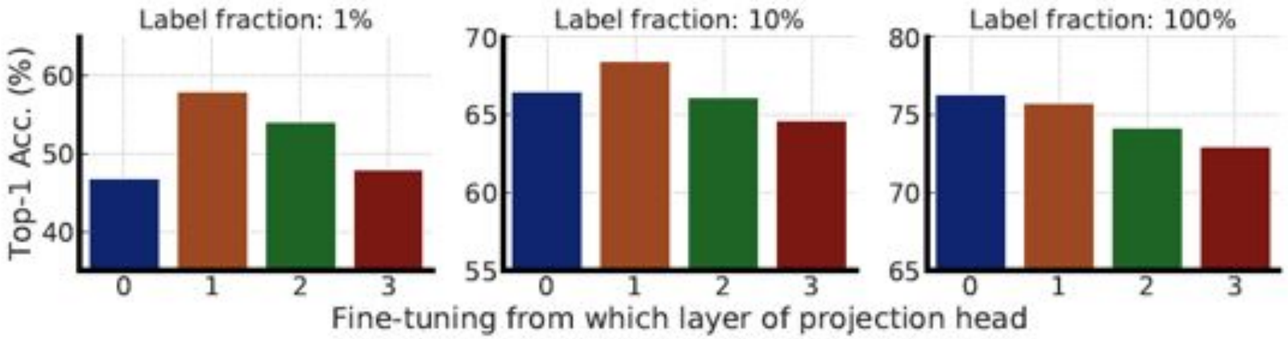
01 | Framework

(2) Deeper non-linear network





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

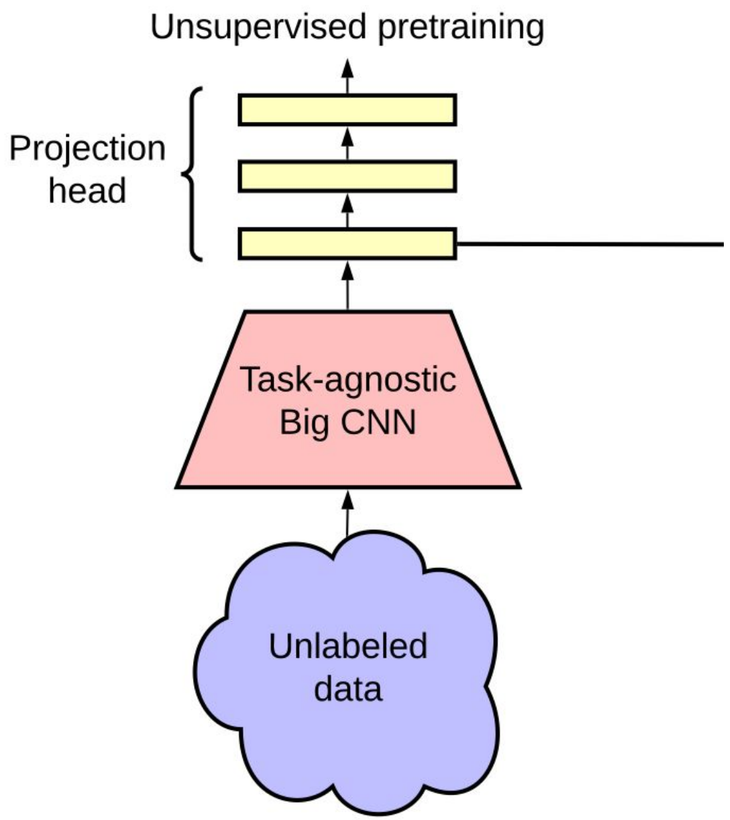


(b) Effect of fine-tuning from middle of a 3-layer projection head (0 is SimCLR).

03 | SimCLR v2

01 | Framework

(3) Moco memory mechanism



D The Impact of Memory

Figure D.1 shows the top-1 comparisons for SimCLRv2 models trained with or without memory (MoCo) [20]. Memory provides modest advantages in terms of linear evaluation and fine-tuning with 1% of the labels; the improvement is around 1%. We believe the reason that memory only provides marginal improvement is that we already use a big batch size (i.e. 4096).

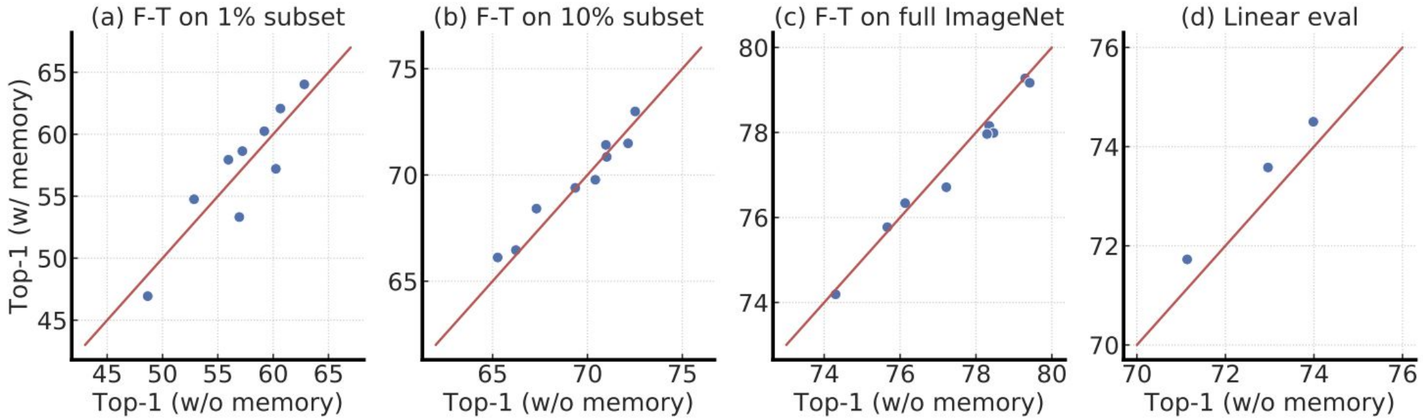
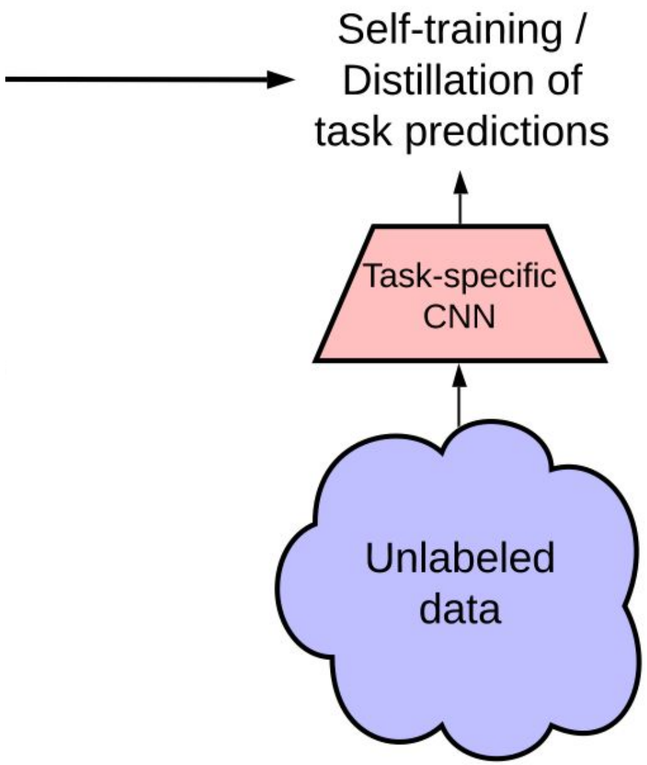


Figure D.1: Top-1 results of ResNet-50, ResNet-101, and ResNet-152 trained with or without memory.

03 | SimCLR v2

02 | Distillation



Self-training / knowledge distillation via unlabeled examples. To further improve the network for the target task, here we leverage the unlabeled data directly for the target task. Inspired by [23, 11, 22, 24, 12], we use the fine-tuned network as a teacher to impute labels for training a student network. Specifically, we minimize the following distillation loss where no real labels are used:

$$\mathcal{L}^{\text{distill}} = -\sum_{\boldsymbol{x}_i \in \mathcal{D}} \left[\sum_{y} P^T(y|\boldsymbol{x}_i;\tau) \log P^S(y|\boldsymbol{x}_i;\tau) \right] \tag{2}$$

where $P(y|\boldsymbol{x}_i) = \exp(f^{\text{task}}(\boldsymbol{x}_i)[y]/\tau) / \sum_{y'} \exp(f^{\text{task}}(\boldsymbol{x}_i)[y']/\tau)$, and τ is a scalar temperature parameter. The teacher network, which produces $P^T(y|\boldsymbol{x}_i)$, is fixed during the distillation; only the student network, which produces $P^S(y|\boldsymbol{x}_i)$, is trained.

While we focus on distillation using only unlabeled examples in this work, when the number of labeled examples is significant, one can also combine the distillation loss with ground-truth labeled examples using a weighted combination

$$\mathcal{L} = -(1-\alpha) \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}^L} \left[\log P^S(y_i|\boldsymbol{x}_i) \right] - \alpha \sum_{\boldsymbol{x}_i \in \mathcal{D}} \left[\sum_{y} P^T(y|\boldsymbol{x}_i;\tau) \log P^S(y|\boldsymbol{x}_i;\tau) \right]. \tag{3}$$

This procedure can be performed using students either with the same model architecture (self-distillation), which further improves the task-specific performance, or with a smaller model architecture, which leads to a compact model.

03 | SimCLR v2

02 | Distillation

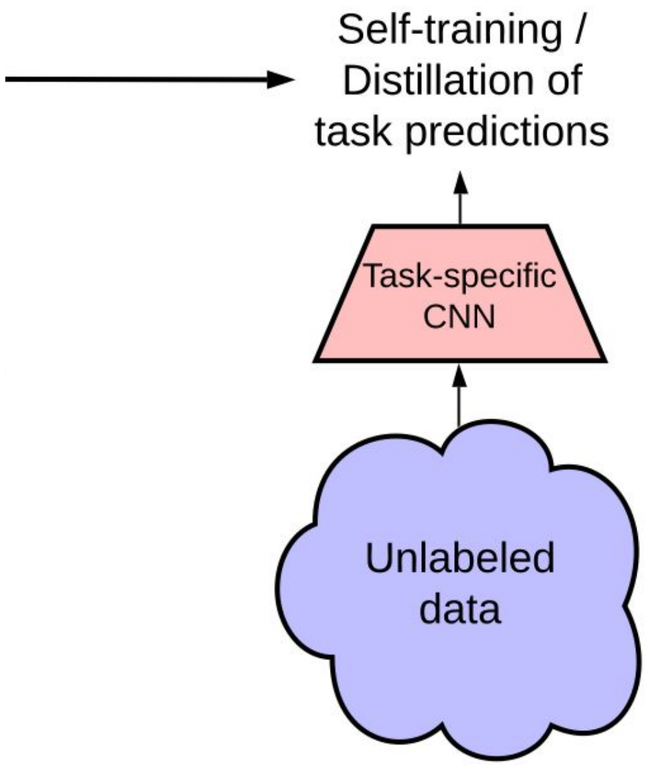
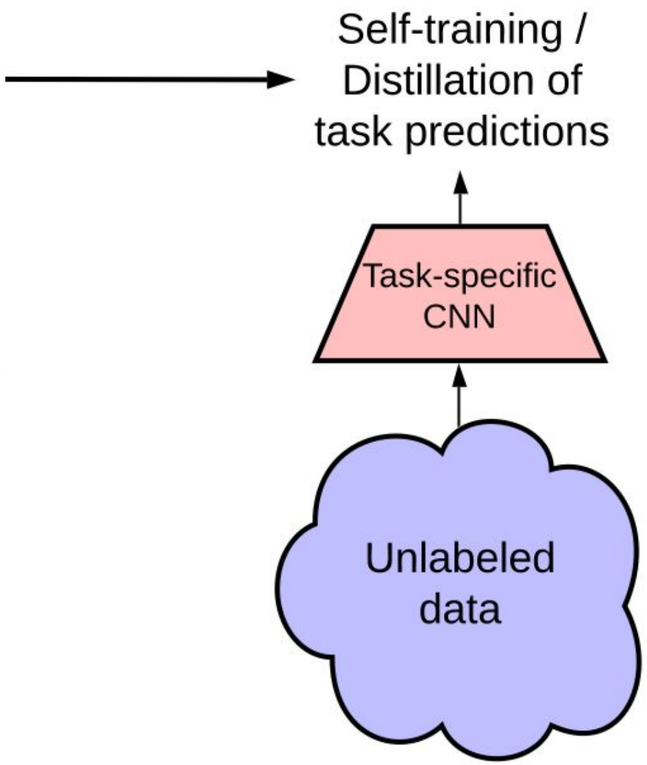


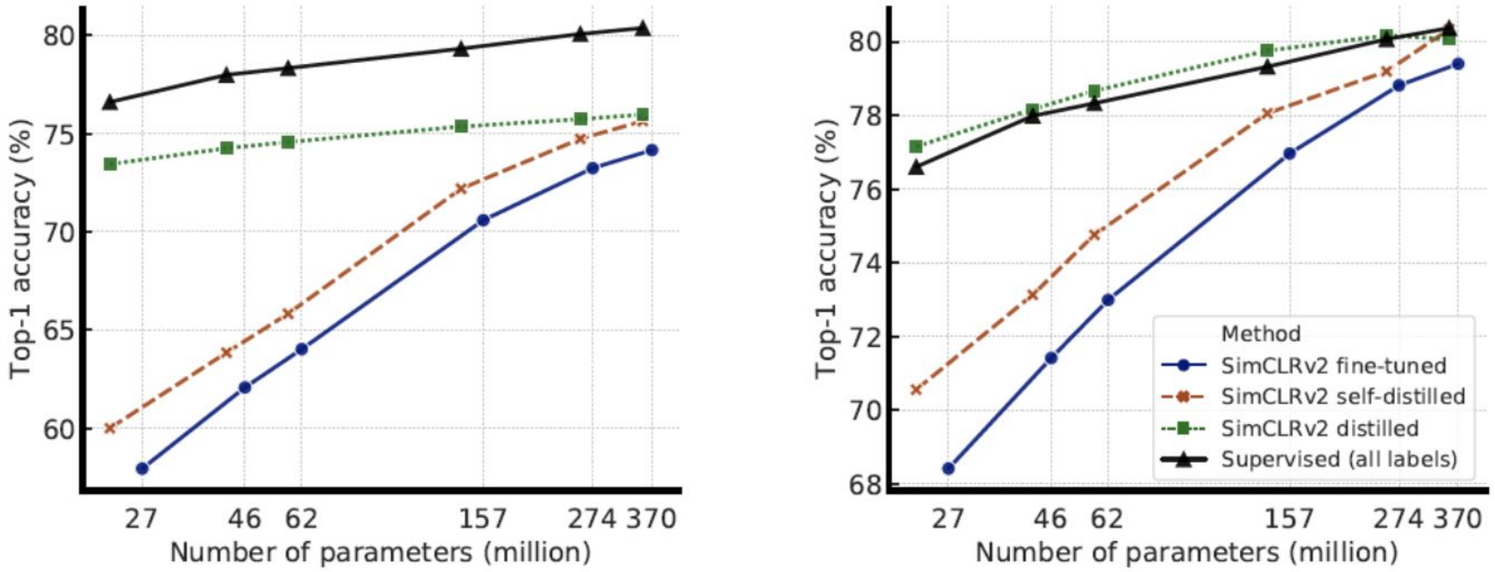
Table 2: Top-1 accuracy of a ResNet-50 trained on different types of targets. For distillation, the temperature is set to 1.0, and the teacher is ResNet-50 (2×+SK), which gets 70.6% with 1% of the labels and 77.0% with 10%, as shown in in Table 1. The distillation loss (Eq. 2) does not use label information. Neither strong augmentation nor extra regularization are used.

Method	Label fraction	
	1%	10%
Label only	12.3	52.0
Label + distillation loss (on labeled set)	23.6	66.2
Label + distillation loss (on labeled+unlabeled sets)	69.0	75.1
Distillation loss (on labeled+unlabeled sets; our default)	68.9	74.3

03 | SimCLR v2

02 | Distillation





(a) Label fraction 1%

(b) Label fraction 10%

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.

03 | SimCLR v2

03 | Results

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**