Self Supervised Learning & Contrastive Learning Methods

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

LMH
Lady Margaret Hall

July 25, 2025

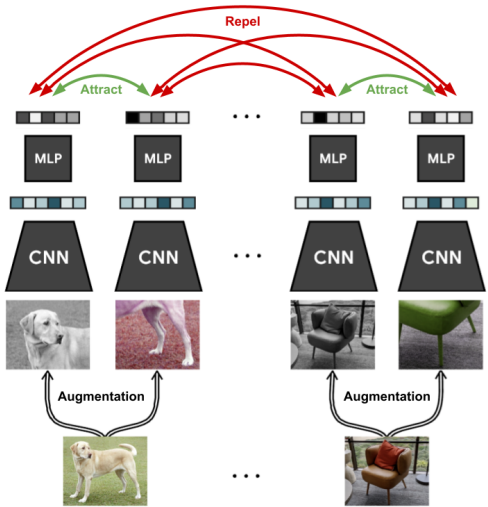

Repel
Attract
Attract
MLP
MLP
MLP
MLP
CNN
CNN
CNN
CNN
Augmentation
Augmentation

Table of Contents

Table of Contents (cont.)

Motivation for Self-Supervised Learning

1. **Imagine you have tons of data, but very few labels.**
   Annotating data is expensive and slow, but unlabeled data is everywhere!

Motivation for Self-Supervised Learning

1. **Imagine you have tons of data, but very few labels.**
   Annotating data is expensive and slow, but unlabeled data is everywhere!
2. **What if you could learn useful features from all that unlabeled data?**
   Self-supervised learning lets us pre-train models to extract general, reusable representations, reducing the need for labeled data later.

Motivation for Self-Supervised Learning

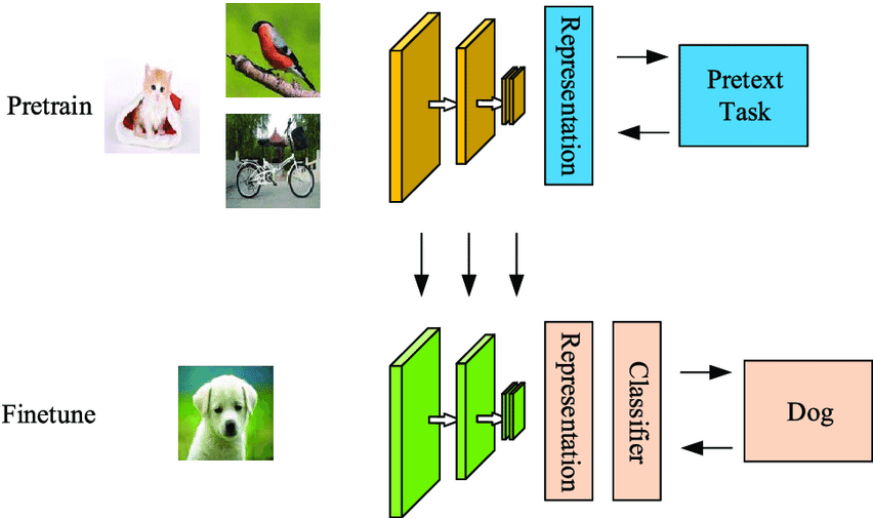
1. **Imagine you have tons of data, but very few labels.**
   Annotating data is expensive and slow, but unlabeled data is everywhere!
2. **What if you could learn useful features from all that unlabeled data?**
   Self-supervised learning lets us pre-train models to extract general, reusable representations, reducing the need for labeled data later.
3. **Think beyond images:**
   These techniques work not just for vision, but also for language, audio, and time-series data, enabling unified approaches across domains.

Motivation for Self-Supervised Learning

1. **Imagine you have tons of data, but very few labels.**
   Annotating data is expensive and slow, but unlabeled data is everywhere!
2. **What if you could learn useful features from all that unlabeled data?**
   Self-supervised learning lets us pre-train models to extract general, reusable representations, reducing the need for labeled data later.
3. **Think beyond images:**
   These techniques work not just for vision, but also for language, audio, and time-series data, enabling unified approaches across domains.
4. **Inspired by how humans learn:**
   We learn patterns from the world around us without explicit labels—self-supervised learning mimics this natural process.

Motivation for Self-Supervised Learning

1. **Imagine you have tons of data, but very few labels.**
   Annotating data is expensive and slow, but unlabeled data is everywhere!
2. **What if you could learn useful features from all that unlabeled data?**
   Self-supervised learning lets us pre-train models to extract general, reusable representations, reducing the need for labeled data later.
3. **Think beyond images:**
   These techniques work not just for vision, but also for language, audio, and time-series data, enabling unified approaches across domains.
4. **Inspired by how humans learn:**
   We learn patterns from the world around us without explicit labels—self-supervised learning mimics this natural process.
5. **A journey of methods:**
   The field has evolved from generative models (like VAEs and GANs), to contrastive and predictive coding, and now to hybrid approaches combining reconstruction and discrimination.


Pretrain
Representation
Pretext Task
Finetune
Representation
Classifier
Dog

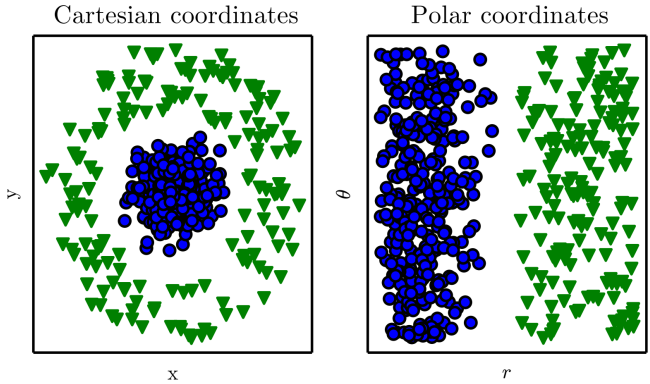
SSL: Learning Outcomes

By the end of this session, you should be able to:

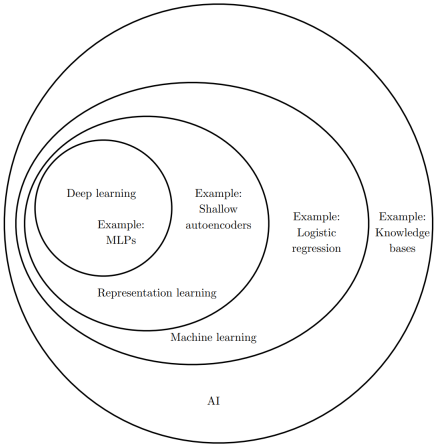
- **Explain** the data efficiency imperative and theoretical motivations behind self-supervised learning (SSL).
- **Identify** key cognitive principles that inspire self-supervision in machine learning.
- **Describe** Instance Discrimination and **implement** the Momentum Contrast (MoCo) framework.
- **Critically assess** pretext tasks such as rotation prediction, relative patch prediction, and view prediction.

SSL: **Introduction**

Representations Matter



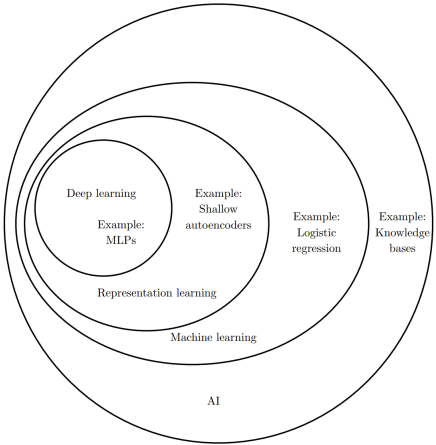
Introduction (cont.)



Deep Unsupervised Learning:

- Learns data representations without using labels.
- Is a subset of Deep Learning, which itself is a subset of Representation Learning, all within Machine Learning.

Introduction (cont.)



Deep Unsupervised Learning:

- Learns data representations without using labels.
- Is a subset of Deep Learning, which itself is a subset of Representation Learning, all within Machine Learning.

Self-Supervised Learning (SSL):

- A form of unsupervised learning that creates supervision from the data itself by designing *pretext tasks*.
- These tasks generate pseudo-labels, enabling the model to learn useful representations without manual annotation.
- Often used interchangeably with unsupervised learning, but SSL specifically focuses on leveraging intrinsic data structure.

Why Self-Supervised Learning?

- **High cost of dataset creation:** Each new task often requires building a new labeled dataset.
  - Involves preparing labeling manuals, defining categories, hiring annotators, building GUIs, and managing storage pipelines.
- **Expensive supervision:** High-quality labels can be costly or impractical to obtain (e.g., in medicine or legal domains).
- **Leverage unlabeled data:** The internet provides vast amounts of unlabeled images, videos, and text that can be utilized.
- **Cognitive motivation:** Animals and babies learn from their environment without explicit supervision, inspiring SSL approaches.

Why Self-Supervised Learning? (cont.)

"Give a robot a label and you feed it for a moment; teach a robot to label and you feed it for a lifetime."

— Pierre Sermanet

This quote highlights the core motivation behind self-supervised learning
→ enabling machines to generate their own supervision from data
→ thus reducing reliance on costly manual labeling and
→ fostering scalable, autonomous learning.

What is Self-Supervised Learning?

SSL is a type of unsupervised learning where the data itself provides supervision. Typically, part of the data is hidden, and a neural network is trained to predict it from the rest—this defines the *pretext task*. Well-chosen pretext tasks help models learn useful features without manual labels.

- **Pretext Task**: Examples include predicting masked patches, future representations (CPC), or distinguishing positive/negative pairs (contrastive).
- **Downstream Task**: After pre-training, the encoder is fine-tuned for tasks like classification or detection.
- **Key Trade-Offs**:
  - **Reconstructive** (e.g., autoencoders): capture low-level details, may miss semantics.
  - **Predictive/Contrastive** (e.g., CPC, MoCo): focus on high-level information.

What is Self-Supervised Learning? (cont.)

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

SSL: **Cognitive Principles**

Cognitive Principles

- **Predictive Coding**: The brain optimizes to predict future sensory inputs.
- **Error Correction**: Learning signals arise from reconstructive or contrastive errors.
  - *Reconstruction from Corrupted Inputs*:
    - Denoising autoencoders
    - Inpainting
    - Colorization, split-brain autoencoders
  - *Visual Common Sense Tasks*:
    - Relative patch prediction
    - Jigsaw puzzles
    - Rotation prediction
  - *Contrastive Learning*:
    - word2vec

Cognitive Principles (cont.)

- Contrastive Predictive Coding (CPC)
- Instance discrimination
- Recent state-of-the-art methods

- **Multi-View Perception**: Integrating different sensory views enhances representations.

SSL: **Pretext Task Taxonomy**

Pretext Task Taxonomy: Introduction

Pretext tasks are broadly categorized by their supervision signal and learning objective:

- **Reconstructive Methods**: Focus on reconstructing parts or properties of the input data.
- **Predictive Methods**: Involve predicting withheld or transformed aspects of the data.

Each category includes various techniques that have advanced SSL in computer vision.

Understanding this taxonomy helps in:

- Selecting suitable pretext tasks for specific applications.
- Gaining insight into how SSL methods learn robust and transferable representations.

SSL: **Instance Discrimination & Contrastive Learning**

Pretext Task Evolution:

- Earlier: handcrafted tasks (e.g., jigsaw, colorization).
- Now: contrastive learning directly optimizes for invariance and discrimination.
- Leads to better, more transferable features and narrows the gap with supervised learning.

Instance Discrimination & Contrastive Learning (cont.

Instance Discrimination:

- Treats each image as its own class.
- Positive pairs: augmented views of the same image.
- Negative pairs: views from different images.
- Promotes invariance to augmentations and discrimination between instances.

Contrastive Learning:

- Self-supervised approach for learning representations, especially in vision.
- Uses data structure instead of labels.
- Learns by pulling together similar (positive) pairs and pushing apart dissimilar (negative) pairs using a contrastive loss (e.g., InfoNCE).

Fundamentals of Instance Discrimination

Instance Discrimination is a self-supervised learning approach where each individual sample in the dataset is treated as a distinct class.

- The core idea is to learn representations by distinguishing between different instances, rather than relying on human-annotated labels.
- **Positive pairs** are generated by applying different augmentations (such as cropping, color jittering, or flipping) to the same image. These augmented views are considered to belong to the same class (i.e., the same instance).
- **Negative pairs** are formed by pairing augmented views of different images. These are treated as belonging to different classes (i.e., different instances).

Fundamentals of Instance Discrimination (cont.)



attract

repel

1. MoCo
2. SimCLR

Fundamentals of Instance Discrimination (cont.)

- The learning objective is to **maximize agreement** (similarity) between representations of positive pairs, while minimizing agreement between negative pairs.
- This is typically achieved using a contrastive loss function, such as InfoNCE, which encourages the model to bring positive pairs closer in the embedding space and push negative pairs apart.
- Instance discrimination forms the basis for many popular self-supervised learning methods, such as MoCo, SimCLR, and PIRL.
- By maximizing agreement across views of the same instance, the model learns to extract meaningful and invariant features, which can be transferred to downstream tasks.

Momentum Contrast for Unsupervised Visual Representation Learning

Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick

Facebook AI Research (FAIR)

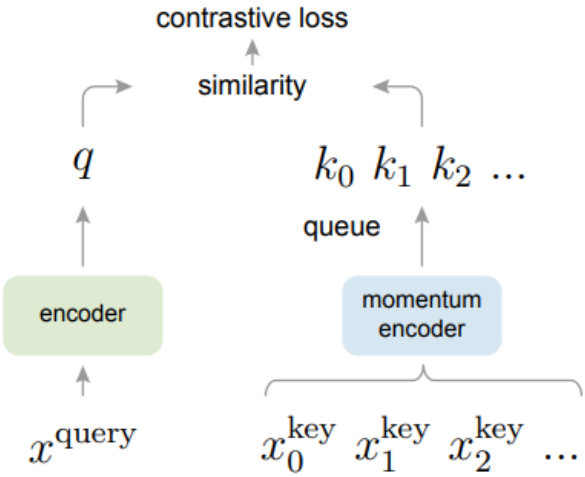
Momentum Contrast (MoCo) is a self-supervised learning framework designed for visual representation learning. Its key components are:

Query & Key Encoders: Two neural networks, f_q (query encoder) and f_k (key encoder), are used. The key encoder f_k is updated as an exponential moving average of the query encoder f_q:

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q$$

where θ_k and θ_q are the parameters of f_k and f_q, and m is the momentum coefficient (e.g., $m = 0.999$).

Momentum Contrast (MoCo) (cont.)



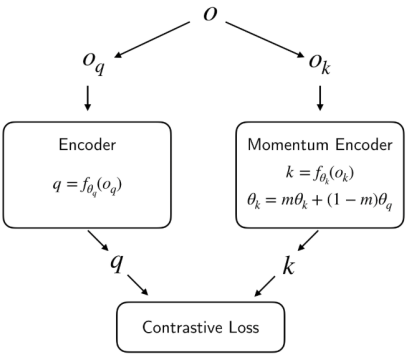
Dictionary Queue: MoCo maintains a large queue (dictionary) of encoded keys from previous batches. This enables the use of a large and consistent set of negative samples for contrastive learning, which is crucial for effective representation learning.

Contrastive Loss: The InfoNCE loss is used to train the encoders. For a given query q and its positive key k^+, along with a set of negative keys $\{k_0, k_1, ..., k_N\}$ from the dictionary, the loss is:

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

where τ is a temperature hyperparameter.

Momentum Contrast (MoCo) (cont.)



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

Momentum Contrast (MoCo) (cont.)

Key Steps in MoCo Training:

1. For each image, generate two augmentations: one for the query encoder (f_q), one for the key encoder (f_k).
2. Encode the query and key.
3. Compute the InfoNCE loss using the current positive key and the dictionary of negative keys.
4. Update f_q via backpropagation; update f_k via momentum.
5. Enqueue the new key and dequeue the oldest key to maintain the dictionary size.

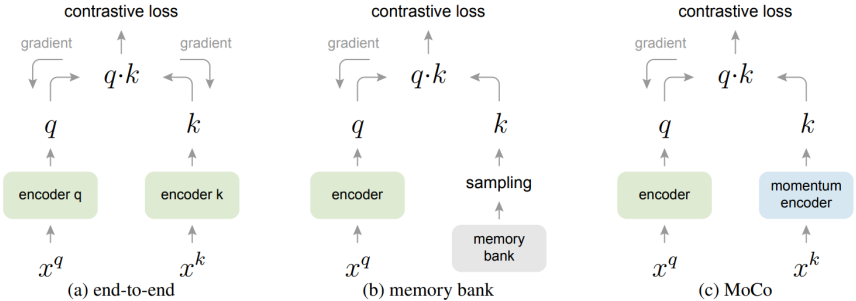
Momentum Contrast (MoCo) (cont.)

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

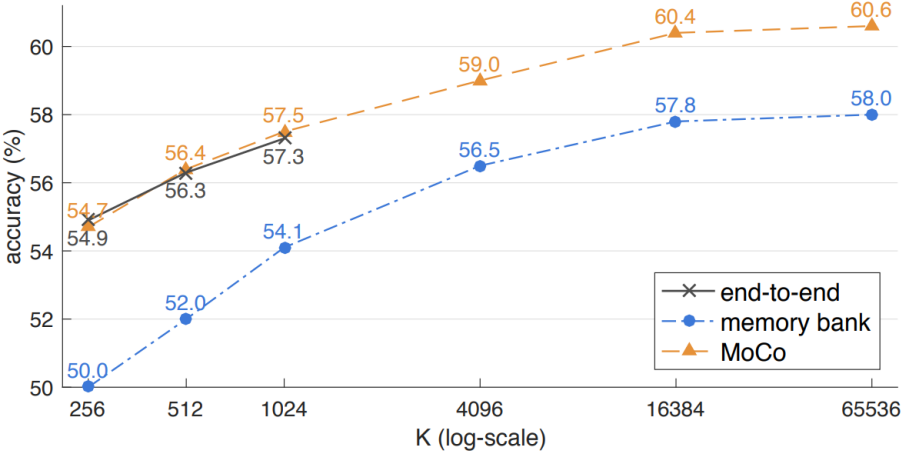
```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

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.

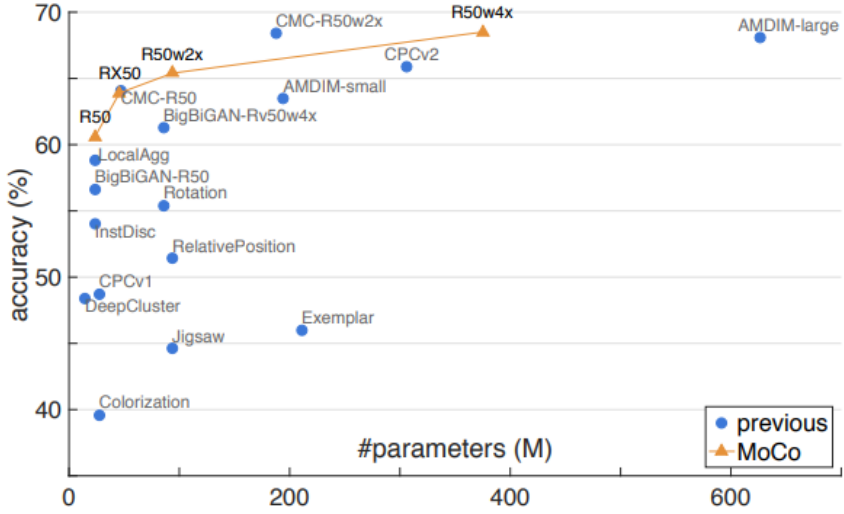
Momentum Contrast (MoCo) (cont.)



Momentum Contrast (MoCo) (cont.)



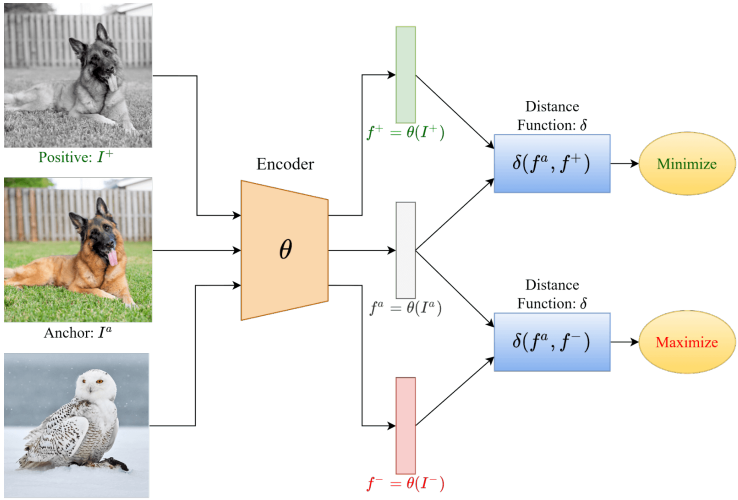
Momentum Contrast (MoCo) (cont.)



Momentum Contrast (MoCo) (cont.)

Key Advantages:

- Enables large and consistent negative sets for contrastive learning.
- Momentum update stabilizes the key encoder, improving training.
- Scalable to large datasets and high-dimensional representations.


Positive: I^+
Anchor: I^a
Negative: I^-
Encoder
θ
$f^+ = \theta(I^+)$
$f^a = \theta(I^a)$
$f^- = \theta(I^-)$
Distance Function: δ
$\delta(f^a, f^+)$
Minimize
Distance Function: δ
$\delta(f^a, f^-)$
Maximize

Recap & Contrastive Learning: Motivation

Where We Are

- Pretext tasks enable self-supervised learning by creating artificial labels.
- Instance discrimination treats each image as its own class, encouraging unique representations.
- MoCo framework uses a dynamic dictionary with a queue and momentum encoder for contrastive learning.
- **Limitations of MoCo:**
  - Need for large memory banks
  - Complexity of momentum encoders
  - Sensitivity to negative sampling

Why Evolve?

- Wouldn't it be great if we could get rid of all that extra memory and complicated tricks?
- Can we squeeze even more out of the positive pairs we already have?
- What if our models could focus on the big picture, not just tiny pixel details?

Contrastive Learning: Learning Outcomes

By the end of this lecture, you will be able to:

- Trace the exciting journey of contrastive learning, understanding how innovations evolved from MoCo $\rightarrow$ SimCLR $\rightarrow$ BYOL $\rightarrow$ DINO $\rightarrow$ iBOT.
- Spot the real-world trade-offs in model design—like balancing memory and compute, or choosing between negative sampling and avoiding Bayesian collapse.
- Derive and visualize how batch size and temperature (τ) shape the spread of learned embeddings.
- Explain in simple terms why we drop the projection head when moving to downstream tasks—and why that's a smart move!
- Analyze the clever tricks BYOL uses to avoid representational collapse, keeping its features meaningful.
- Illustrate why centering and sharpening the teacher's outputs are essential for effective learning.

Contrastive Learning: **Simple Framework for Contrastive Learning (SimCLR)**

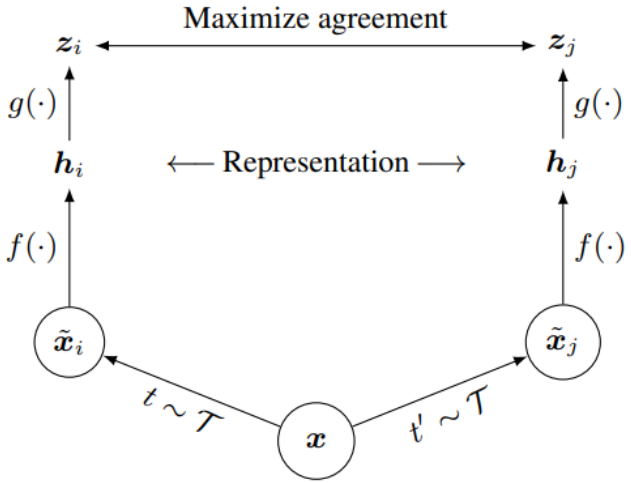
A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen [1] Simon Kornblith [1] Mohammad Norouzi [1] Geoffrey Hinton [1]

What is SimCLR?

- A simple and powerful method for learning image features without labels.
- Uses contrastive learning:
  - Pulls similar images (augmentations of the same image) closer together.
  - Pushes different images (from different sources) further apart.

SimCLR: Simple Framework for Contrastive Learning (cont.)



Why was SimCLR created?

- Earlier contrastive learning methods needed extra tricks:
  - Memory banks to store negative examples.
  - Special momentum encoders.
- SimCLR avoids these by:
  - Using very large batch sizes.
  - Always having plenty of negative examples in each batch.
- This makes SimCLR simpler and easier to train.

How does SimCLR work?

- **Augmentation pipeline:**
  - Each image is randomly cropped.
  - Color is changed.
  - Sometimes blurred.
  - This creates two different views of the same image.

- **Encoder + Projection Head:**
  - Both views go through a neural network encoder ($f(\cdot)$).
  - Then, they pass through a small MLP called the projection head ($g(\cdot)$).
  - This gives the final representations.

- **Contrastive Loss (NT-Xent):**
  - The model pulls together representations of two views of the same image.
  - It pushes apart representations of different images.
  - The loss function is:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k \neq i} \exp(\text{sim}(z_i, z_k)/\tau)}$$

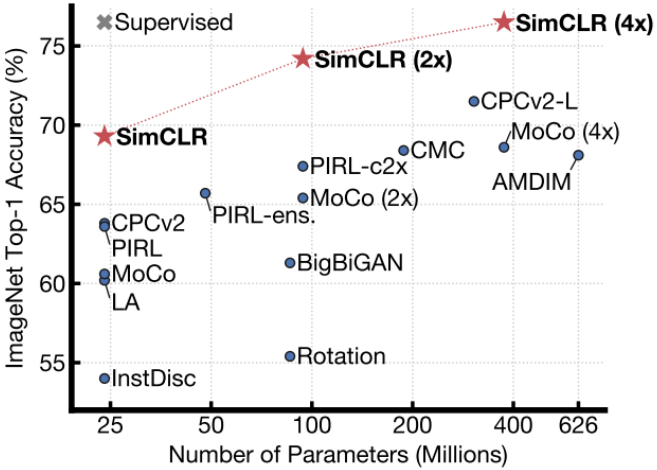
  - $\text{sim}(z_i, z_j)$ is the similarity between two representations.
  - τ is a temperature parameter.

SimCLR: Simple Framework for Contrastive Learning (cont.)

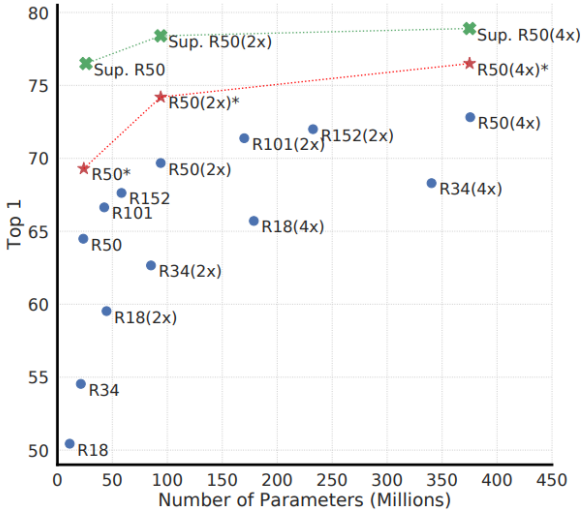
Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, temperature 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**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\tau \|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i,j) = -\log \frac{\exp(s_{i,j})}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k})}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network f

SimCLR: Simple Framework for Contrastive Learning (cont.)



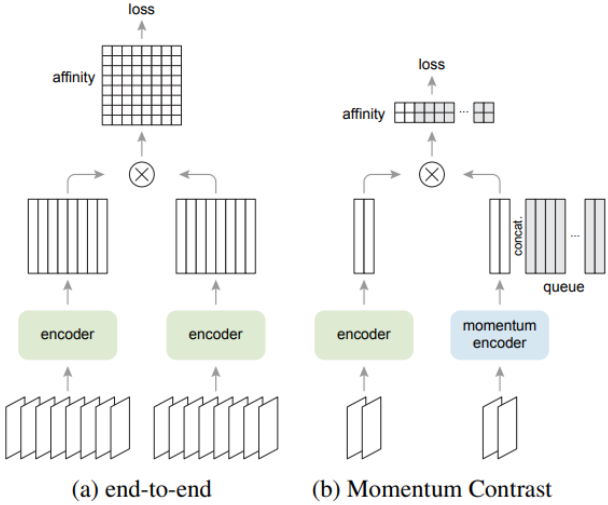
SimCLR: Simple Framework for Contrastive Learning (cont.)



What did we learn from SimCLR?

- Using strong data augmentations is really important for learning good features.
- Adding a nonlinear projection head (the MLP) helps the model learn better representations.
- SimCLR needs large batch sizes, which means it requires a lot of computing power.

MoCov2 vs. SimCLR



MoCov2 vs. SimCLR (cont.)

case	unsup. pre-train MLP	aug+	cos	epochs	ImageNet acc.	VOC detection AP_{50}	AP	AP_{75}
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	**82.5**	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	**800**	**71.1**	**82.5**	**57.4**	**64.0**

MoCov2 vs. SimCLR (cont.)

case	unsup. pre-train MLP	aug+	cos	epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	**67.5**
*results of **longer** unsupervised training follow:*						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	**71.1**

Contrastive Learning: **Bootstrap Your Own Latent (BYOL)**

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

Jean-Bastien Grill[*,1] **Florian Strub**[*,1] **Florent Altché**[*,1] **Corentin Tallec**[*,1] **Pierre H. Richemond**[*,1,2]
Elena Buchatskaya[1] **Carl Doersch**[1] **Bernardo Avila Pires**[1] **Zhaohan Daniel Guo**[1]
Mohammad Gheshlaghi Azar[1] **Bilal Piot**[1] **Koray Kavukcuoglu**[1] **Rémi Munos**[1] **Michal Valko**[1]

[1]DeepMind [2]Imperial College

[jbgrill,fstrub,altche,corentint,richemond]@google.com

Abstract

We introduce **B**ootstrap **Y**our **O**wn **L**atent (BYOL), a new approach to self-supervised image representation learning. BYOL relies on two neural networks, referred to as *online* and *target* networks, that interact and learn from each other. From an augmented view of an image, we train the online network to predict the target network representation of the same image under a different augmented view. At the same time, we update the target network with a slow-moving average of the online network. While state-of-the art methods intrinsically rely on negative pairs, BYOL achieves a new state of the art *without them*. BYOL reaches 74.3% top-1 classification accuracy on ImageNet using the standard linear evaluation protocol with a ResNet-50 architecture and 79.6% with a larger ResNet. We show that BYOL performs on par or better than the current state of the art on both transfer and semi-supervised benchmarks.

Why BYOL?

- Most contrastive methods use both positive and negative pairs.
- BYOL asks: Can we learn good features without negative pairs?
- BYOL shows it is possible to skip negatives and still learn useful representations.

BYOL (Bootstrap Your Own Latent) (cont.)

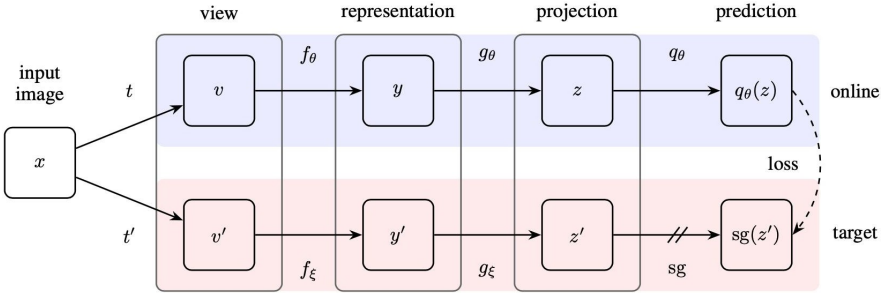
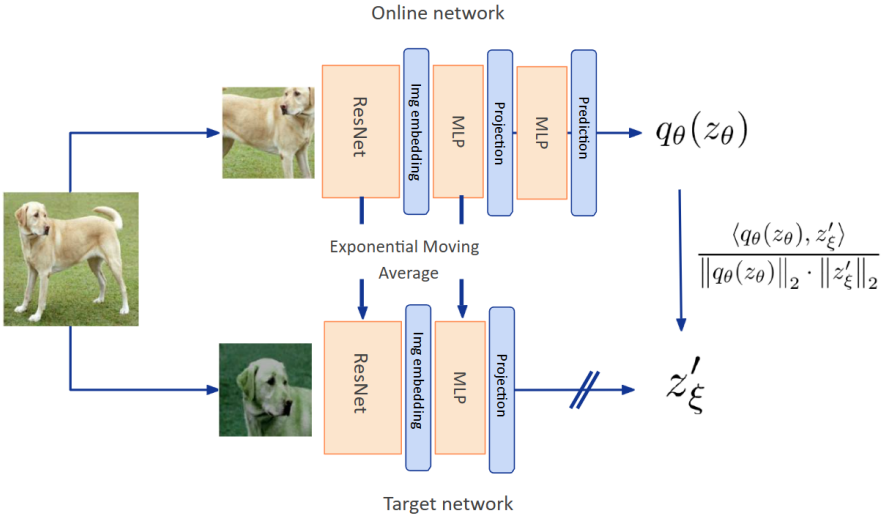


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z)$ and $\mathrm{sg}(z')$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded and y is used as the image representation.

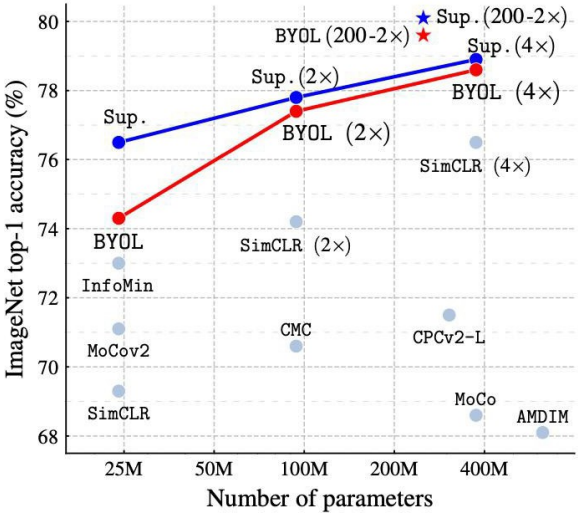
How does BYOL work?

- There are two networks: an **online network** and a **target network**. Both start out the same.
- Each network has an encoder (think of it as a feature extractor).
- The online network has an extra part called the **predictor head**.
- The target network's parameters are updated slowly using the online network's parameters (using something called an exponential moving average).
- The goal is simple: make the online network's prediction as close as possible to the target network's output, using mean squared error between their normalized outputs.

BYOL (Bootstrap Your Own Latent) (cont.)



BYOL (Bootstrap Your Own Latent) (cont.)



BYOL (Bootstrap Your Own Latent) (cont.)

Method	Top-1	Top-5
Local Agg.	60.2	-
PIRL [35]	63.6	-
CPC v2 [32]	63.8	85.3
CMC [11]	66.2	87.0
SimCLR [8]	69.3	89.0
MoCo v2 [37]	71.1	-
InfoMin Aug. [12]	73.0	91.1
BYOL (ours)	**74.3**	**91.6**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1	Top-5
SimCLR [8]	ResNet-50 (2→)	94M	74.2	92.0
CMC [11]	ResNet-50 (2→)	94M	70.6	89.7
BYOL (ours)	ResNet-50 (2→)	94M	77.4	93.6
CPC v2 [32]	ResNet-161	305M	71.5	90.1
MoCo [9]	ResNet-50 (4→)	375M	68.6	-
SimCLR [8]	ResNet-50 (4→)	375M	76.5	93.2
BYOL (ours)	ResNet-50 (4→)	375M	78.6	94.2
BYOL (ours)	ResNet-200 (2→)	250M	**79.6**	**94.8**

(b) Other ResNet encoder architectures.

Table 1: Top-1 and top-5 accuracies (in %) under linear evaluation on ImageNet.

BYOL (Bootstrap Your Own Latent) (cont.)

Method	Food101	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
BYOL (ours)	**75.3**	91.3	**78.4**	**57.2**	**62.2**	**67.8**	60.6	82.5	75.5	90.4	94.2	**96.1**
SimCLR (repro)	72.8	90.5	74.4	42.4	60.6	49.3	49.8	81.4	**75.7**	84.6	89.3	92.6
SimCLR [8]	68.4	90.6	71.6	37.4	58.8	50.3	50.3	80.5	74.5	83.6	90.3	91.2
Supervised-IN [8]	72.3	**93.6**	78.3	53.7	61.9	66.7	**61.0**	**82.8**	74.9	**91.5**	**94.5**	94.7
Fine-tuned:												
BYOL (ours)	**88.5**	**97.8**	86.1	**76.3**	63.7	91.6	**88.1**	**85.4**	**76.2**	91.7	**93.8**	97.0
SimCLR (repro)	87.5	97.4	85.3	75.0	63.9	91.4	87.6	84.5	75.4	89.4	91.7	96.6
SimCLR [8]	88.2	97.7	85.9	75.9	63.5	91.3	88.1	84.1	73.2	89.2	92.1	97.0
Supervised-IN [8]	88.3	97.5	**86.4**	75.8	**64.3**	**92.1**	86.0	85.0	74.6	**92.1**	93.3	**97.6**
Random init [8]	86.9	95.9	80.2	76.1	53.6	91.4	85.9	67.3	64.8	81.5	72.6	92.0

Table 3: Transfer learning results from ImageNet (IN) with the standard ResNet-50 architecture.

BYOL (Bootstrap Your Own Latent) (cont.)

What's special about BYOL?

- Even though there are no explicit negatives, the difference between the online and target networks acts like a source of "negative" information.
- This means we don't need huge batch sizes or to search for negative samples.
- BYOL makes self-supervised learning simpler and more efficient!

Contrastive Learning: **Distillation with No Labels (DINO)**

DINO (Distillation with No Labels)

Emerging Properties in Self-Supervised Vision Transformers

Mathilde Caron[1,2] Hugo Touvron[1,3] Ishan Misra[1] Hervé Jegou[1]
Julien Mairal[2] Piotr Bojanowski[1] Armand Joulin[1]

[1] Facebook AI Research [2] Inria* [3] Sorbonne University

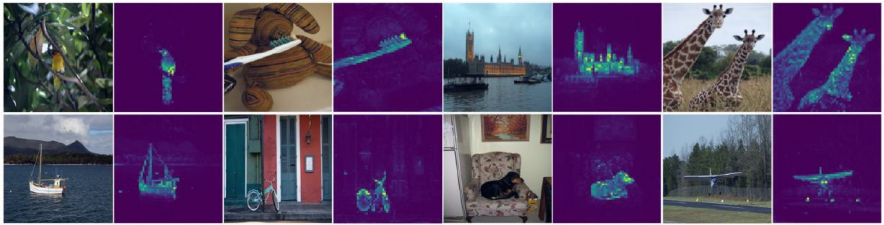


Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

DINO (Distillation with No Labels) (cont.)

Why DINO?

- Learns useful features from images without any labels.
- Helps the model understand both the big picture and small details.
- Combines the strengths of contrastive learning and clustering methods.

DINO (Distillation with No Labels) (cont.)

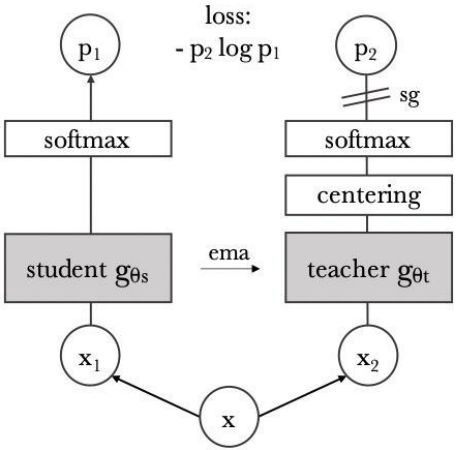


Figure 2: **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a K dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters.

DINO (Distillation with No Labels) (cont.)

How does DINO work?

- **Student and Teacher Networks:**
  There are two networks with the same design. The teacher is just a slowly updated version of the student (using EMA).
- **Multi-crop Strategy:**
  The model looks at the same image in different ways—two large views and several small crops—to learn features that work at different scales.
- **No Negatives Needed:**
  Instead of comparing with negative samples, DINO uses a cross-entropy loss on soft similarity scores between the student and teacher outputs.
- **Avoiding Collapse:**
  To make sure the model doesn't just output the same thing for every image, DINO centers and sharpens the teacher's outputs.

DINO (Distillation with No Labels) (cont.)

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

DINO (Distillation with No Labels) (cont.)

Table 2: **Linear and k-NN classification on ImageNet.** We report top-1 accuracy for linear and k-NN evaluations on the validation set of ImageNet for different self-supervised methods. We focus on ResNet-50 and ViT-small architectures, but also report the best results obtained across architectures. * are run by us. We run the k-NN evaluation for models with official released weights. The throughput (im/s) is calculated on a NVIDIA V100 GPU with 128 samples per forward. Parameters (M) are of the feature extractor.

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [11]	RN50	23	1237	69.1	60.7
MoCov2 [13]	RN50	23	1237	71.1	61.9
InfoMin [54]	RN50	23	1237	73.0	65.3
BarlowT [66]	RN50	23	1237	73.2	66.0
OBoW [21]	RN50	23	1237	73.8	61.9
BYOL [23]	RN50	23	1237	74.4	64.8
DCv2 [9]	RN50	23	1237	75.2	67.1
SwAV [9]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**
Supervised	ViT-S	21	1007	79.8	79.8
BYOL* [23]	ViT-S	21	1007	71.4	66.6
MoCov2* [13]	ViT-S	21	1007	72.7	64.4
SwAV* [9]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**
Comparison across architectures					
SCLR [11]	RN50w4	375	117	76.8	69.3
SwAV [9]	RN50w2	93	384	77.3	67.3
BYOL [23]	RN50w2	93	384	77.4	–
DINO	ViT-B/16	85	312	78.2	76.1
SwAV [9]	RN50w5	586	76	78.5	67.1
BYOL [23]	RN50w4	375	117	78.6	–
BYOL [23]	RN200w2	250	123	79.6	73.9
DINO	ViT-S/8	21	180	79.7	**78.3**
SCLRv2 [12]	RN152w3+SK	794	46	79.8	73.1
DINO	ViT-B/8	85	63	**80.1**	77.4

DINO (Distillation with No Labels) (cont.)

	Random	Supervised	DINO
ViT-S/16	22.0	27.3	45.9
ViT-S/8	21.8	23.7	44.7

Figure 4: **Segmentations from supervised versus DINO.** We visualize masks obtained by thresholding the self-attention maps to keep 60% of the mass. On top, we show the resulting masks for a ViT-S/8 trained with supervision and DINO. We show the best head for both models. The table at the bottom compares the Jaccard similarity between the ground truth and these masks on the validation images of PASCAL VOC12 dataset.

DINO (Distillation with No Labels) (cont.)

What makes DINO special?

- Learning from both big and small image parts helps the model understand local details.
- Using soft targets (from the teacher) gives the student more useful feedback than just using hard negatives.

Contrastive Learning: **DINOv2: Learning Robust Visual Features without Supervision**

DINOv2: Learning Robust Visual Features without Supervision

Maxime Oquab, Timothée Darcet**, Théo Moutakanni**,
Huy V. Vo*, Marc Szafraniec*, Vasil Khalidov*, Pierre Fernandez, Daniel Haziza,
Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba,
Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat,
Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal[1],
Patrick Labatut*, Armand Joulin*, Piotr Bojanowski***

Meta AI Research [1]*Inria*

DINOv2 (cont.)

	INet-1k k-NN	INet-1k linear
iBOT	72.9	82.3
+(our reproduction)	74.5 ↑ 1.6	83.2 ↑ 0.9
+LayerScale, Stochastic Depth	75.4 ↑ 0.9	82.0 ↓ 1.2
+128k prototypes	76.6 ↑ 1.2	81.9 ↓ 0.1
+KoLeo	78.9 ↑ 2.3	82.5 ↑ 0.6
+SwiGLU FFN	78.7 ↓ 0.2	83.1 ↑ 0.6
+Patch size 14	78.9 ↑ 0.2	83.5 ↑ 0.4
+Teacher momentum 0.994	79.4 ↑ 0.5	83.6 ↑ 0.1
+Tweak warmup schedules	80.5 ↑ 1.1	83.8 ↑ 0.2
+Batch size 3k	81.7 ↑ 1.2	84.7 ↑ 0.9
+Sinkhorn-Knopp	81.7 =	84.7 =
+Untying heads = DINOv2	82.0 ↑ 0.3	84.5 ↓ 0.2

DINOv2 (cont.)

Method	Arch.	Data	Text sup.	kNN val	linear val	linear ReaL	linear V2
Weakly supervised							
CLIP	ViT-L/14	WIT-400M	✓	79.8	84.3	88.1	75.3
CLIP	ViT-L/14_{336}	WIT-400M	✓	80.5	85.3	88.8	75.8
SWAG	ViT-H/14	IG3.6B	✓	82.6	85.7	88.7	77.6
OpenCLIP	ViT-H/14	LAION-2B	✓	81.7	84.4	88.4	75.5
OpenCLIP	ViT-G/14	LAION-2B	✓	83.2	86.2	89.4	77.2
EVA-CLIP	ViT-g/14	custom*	✓	**83.5**	86.4	89.3	77.4
Self-supervised							
MAE	ViT-H/14	INet-1k	×	49.4	76.6	83.3	64.8
DINO	ViT-S/8	INet-1k	×	78.6	79.2	85.5	68.2
SEERv2	RG10B	IG2B	×	–	79.8	–	–
MSN	ViT-L/7	INet-1k	×	79.2	80.7	86.0	69.7
EsViT	Swin-B/W=14	INet-1k	×	79.4	81.3	87.0	70.4
Mugs	ViT-L/16	INet-1k	×	80.2	82.1	86.9	70.8
iBOT	ViT-L/16	INet-22k	×	72.9	82.3	87.5	72.4
DINOv2	ViT-S/14	LVD-142M	×	79.0	81.1	86.6	70.9
	ViT-B/14	LVD-142M	×	82.1	84.5	88.3	75.1
	ViT-L/14	LVD-142M	×	**83.5**	86.3	89.5	78.0
	ViT-g/14	LVD-142M	×	**83.5**	**86.5**	**89.6**	**78.4**

DINOv2 (cont.)

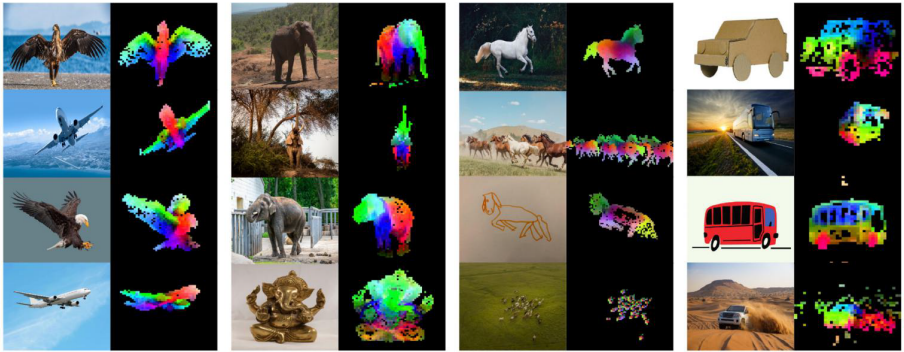


Figure 1: **Visualization of the first PCA components.** We compute a PCA between the patches of the images from the same column (a, b, c and d) and show their first 3 components. Each component is matched to a different color channel. Same parts are matched between related images despite changes of pose, style or even objects. Background is removed by thresholding the first PCA component.

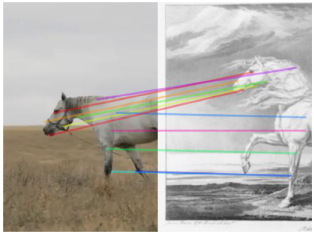


Figure 2: Feature matching

Figure 3: Image retrieval

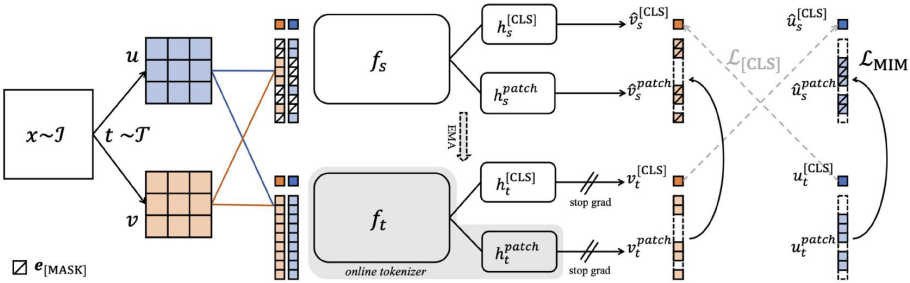
Figure 4: Segmentation

Figure 5: Depth prediction

Contrastive Learning: **Image BERT Pre-Training with Online Tokenizer (iBOT)**

iBOT 🤖: Image BERT Pre-Training with Online Tokenizer

Jinghao Zhou[1] **Chen Wei**[2] **Huiyu Wang**[2] **Wei Shen**[3] **Cihang Xie**[4] **Alan Yuille**[2] **Tao Kong**[1]
[1]ByteDance [2]Johns Hopkins University [3]Shanghai Jiao Tong University [4]UC Santa Cruz



Why iBOT?

- Improves self-supervised learning for vision transformers.
- Combines two main ideas:
  - Learning from missing pieces of images (like solving a puzzle).
  - Learning by copying a smart teacher model.

iBOT (cont.)

How does it work?

- **Masked Image Modeling (MIM):** Randomly hide some image patches, so the model has to guess what's missing.
- **Online Distillation:** There's a student model and a teacher model. The student tries to predict both the overall image and the hidden patches, using hints from the teacher.
- **Teacher as EMA:** The teacher isn't trained directly. Instead, it's a slow-moving average of the student, just like in DINO or BYOL.
- **Losses:**
  - **Image-level distillation:** The student learns to match the teacher's understanding of the whole image.
  - **Token-level distillation:** The student also learns to match the teacher's guesses for the hidden patches.

iBOT (cont.)

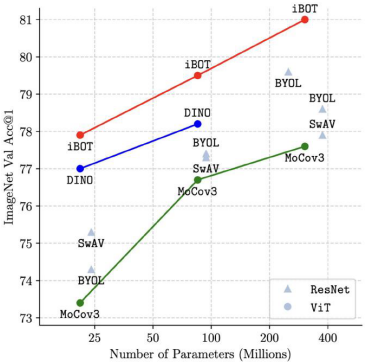


Table 6: **Object detection (Det.) & instance segmentation (ISeg.) on COCO and Semantic segmentation (Seg.) on ADE20K.** We report the results of ViT-S/16 (left) and ViT-B/16 (right). Seg.† denotes using a linear head for semantic segmentation.

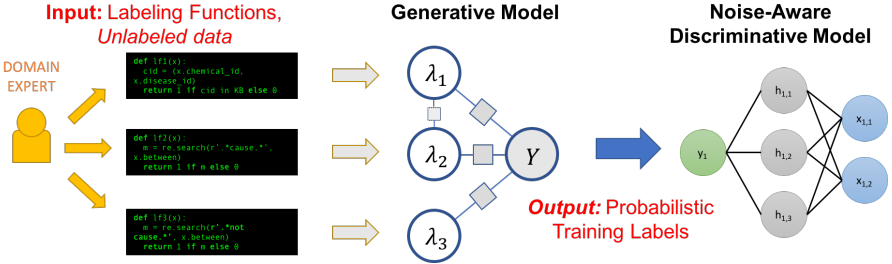
Method	Arch.	Param.	Det.	ISeg.	Seg.
			AP^b	AP^m	mIoU
Sup.	Swin-T	29	48.1	41.7	44.5
MoBY	Swin-T	29	48.1	41.5	44.1
Sup.	ViT-S/16	21	46.2	40.1	44.5
iBOT	ViT-S/16	21	**49.4**	**42.6**	**45.4**

Method	Det.	ISeg.	Seg.†	Seg.
	AP^b	AP^m	mIoU	mIoU
Sup.	49.8	43.2	35.4	46.6
BEiT	50.1	43.5	27.4	45.8
DINO	50.1	43.4	34.5	46.8
iBOT	**51.2**	**44.2**	**38.3**	**50.0**

Why is this cool?

- Mixing MIM and distillation helps the model learn both big-picture (global) and detailed (local) features.
- The model can do zero-shot classification—meaning it can recognize new things without extra training—by using prompts, just like CLIP!

Weak Supervised Learning (WSL)


Input: Labeling Functions,
Unlabeled data
DOMAIN EXPERT
def lf1(x):
cid = (x.chemical_id, x.disease_id)
return 1 if cid in KB else 0
def lf2(x):
m = re.search(r'.*cause.*', x.between)
return 1 if m else 0
def lf3(x):
m = re.search(r'.*not cause.*', x.between)
return 1 if m else 0
Generative Model
λ_1
λ_2
λ_3
Y
Output: Probabilistic Training Labels
Noise-Aware Discriminative Model
y_1
$h_{1,1}$
$h_{1,2}$
$h_{1,3}$
$x_{1,1}$
$x_{1,2}$

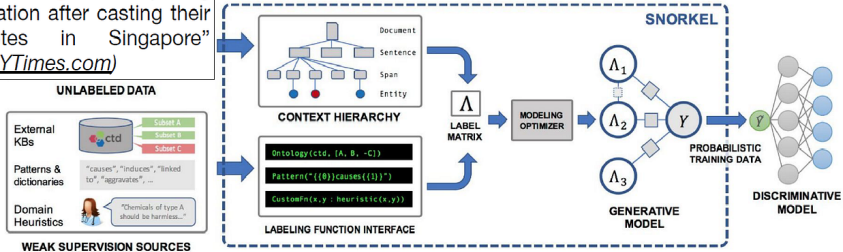
What is Weak Supervised Learning?

- Sometimes, we don't have perfect data or labels for training.
- **Weak Supervised Learning (WSL)** helps us learn even when our labels are not ideal.
- There are different ways labels can be "weak":
  - **Incomplete labels:** Only some data points have labels.
  - **Inexact labels:** Labels are not very detailed (e.g., we know what's in an image, but not where).
  - **Inaccurate labels:** Some labels might be wrong or noisy.

Snorkel: data programming

"Prime Minister Lee Hsien Loong and his wife Ho Ching leave a polling station after casting their votes in Singapore" (NYTimes.com)

UNLABELED DATA



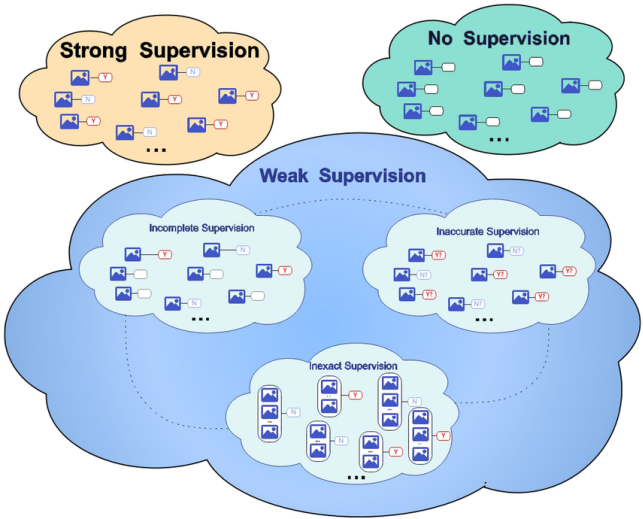
What is Weak Supervised Learning? (cont.)

Why do we use WSL?

- Getting perfect labels for all data is expensive and slow.
- Experts are needed to label some data, which costs a lot.
- WSL lets us use cheaper, less perfect labels to train our models.

When is WSL useful?

- When we have lots of data but not enough labels (like in medical images).
- When we can use rules or existing knowledge to help label data (like using a dictionary for text).


Strong Supervision
No Supervision
Weak Supervision
Incomplete Supervision
Inaccurate Supervision
Inexact Supervision

How can we do Weak Supervised Learning?

- **Programmatic Labeling (Data Programming)**
  - Write simple rules (labeling functions) to label data automatically.
  - Use a tool like Snorkel to combine these noisy labels and clean them up.
  - Steps: Unlabeled data $\rightarrow$ Rules $\rightarrow$ Cleaned labels $\rightarrow$ Train model.
  - (Snorkel: Ratner et al., 2016)

- **Multiple Instance Learning (MIL)**
  - Sometimes we only know if a group (bag) has something, not which item.
  - Example: We know an image has a cat, but not where.
  - The model learns to find the right items inside each group.
  - Used in medical images, object detection, etc.

- **Webly / Noisy Label Learning**
  - Get labels from the internet or crowdsourcing—they can be messy!
  - Use special tricks to handle wrong or noisy labels (like Co-teaching).

- **Hybrid Methods (Semi + Weak Supervision)**
  - Mix a small set of good labels with lots of weak ones.
  - Use tricks like pseudo-labeling or learning in steps.

Summary

- **SimCLR:**
  - Simple framework for contrastive learning.
  - Uses data augmentation and contrastive loss.
- **BYOL:**
  - Self-supervised approach without negative pairs.
  - Uses two networks: online and target.
- **DINO:**
  - Self-distillation method without labels.
  - Uses teacher-student networks and knowledge distillation.

Summary (cont.)

- **DINOv2:**
  - Improved version of DINO.
  - Enhanced training strategies and architectures.
  - Better performance and scalability.
- **iBOT:**
  - Combines contrastive learning with masked image modeling.
  - Uses BERT-style pretraining and vision transformers.

SSL: **References**

References

[1] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). *Extracting and Composing Robust Features with Denoising Autoencoders*. ICML.

[2] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A. A. (2016). *Context Encoders: Feature Learning by Inpainting*. CVPR.

[3] Noroozi, M., & Favaro, P. (2016). *Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles*. ECCV.

[4] Gidaris, S., Singh, P., & Komodakis, N. (2018). *Unsupervised Representation Learning by Predicting Image Rotations*. ICLR.

[5] van den Oord, A., Li, Y., & Vinyals, O. (2018). *Representation Learning with Contrastive Predictive Coding*. arXiv:1807.03748.

References (cont.)

[6] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv:1301.3781.

[7] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). *Momentum Contrast for Unsupervised Visual Representation Learning*. CVPR.

Contrastive Learning: **References**

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton. *A Simple Framework for Contrastive Learning of Visual Representations.* ICML 2020. arXiv:2002.05709.

[2] Jean-Bastien Grill et al. *Bootstrap Your Own Latent — A New Approach to Self-Supervised Learning.* NeurIPS 2020. arXiv:2006.07733.

[3] Mathilde Caron et al. *Emerging Properties in Self-Supervised Vision Transformers.* ICCV 2021. arXiv:2104.14294.

[4] Zheng Zhang et al. *iBOT: Image BERT Pre-Training with Online Token-Level Distillation.* ICML 2022. arXiv:2204.03615.

[5] Kristina Khvatova. *Weakly Supervised Learning: Introduction and Best Practices.* Data Science Milan, 2019.

[6] Frontiers in AI. *Taxonomy of Weakly Supervised Learning.* 2021.

References (cont.)

[7] Liangliang Cao, Zicheng Liu, and Thomas S. Huang. *Weakly Supervised Activity Classification.* Frontiers, 2012.

[8] Alex Ratner et al. *Snorkel: Rapid Training Data Creation with Weak Supervision.* ICML 2016. arXiv:1711.10160.

[9] **Survey:** Xinlei Chen, Kaiming He. *Exploring Simple Siamese Representation Learning.* CVPR 2021. arXiv:2011.10566.

[10] **Tutorial:** Yann LeCun's Lectures on Self-Supervised Learning (YouTube).

[11] **Code:** SimCLR (TensorFlow)

[12] **Code:** BYOL (PyTorch)

[13] **Code:** DINO (PyTorch)

References (cont.)

[14] **Code:** iBOT (PyTorch)

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.