Self-Supervised Contrastive Learning



Antonio Carta

2026-11-04

University of Pisa

antonio.carta@unipi.it

Outline

- **Self-Supervised Learning**: learning representations without labels
- **Contrastive learning and Augmentations**: compare and contrast positive/negative pairs, instance discrimination
- **SimCLR**: a simple and strong contrastive framework

Pretrain-then-Finetune

the standard recipe in computer vision / deep learning

1. **Pretrain** a backbone on a large labeled dataset (e.g. ImageNet classification)
2. **Remove** the classification head, keep the backbone as a feature extractor
3. **Finetune** on the downstream task (classification, detection, segmentation, ...)

The backbone learns a **representation** $h = f_\theta(x)$ that is reused across tasks.

Note: pre-training is a preliminary step that we use to learn the representations. The eng goal is the performance on **unseen downstream tasks**

Goal: fing a pretraining recipe that gives high performance on downstream vision benchmarks (after finetuning)

Labels are Expensive

- Representation quality scales with the number of samples: we need large datasets
- Annotations are expensive
  - cost grows with the task: image labels $<$ boxes $<$ segmentation masks
  - Some domains need experts (medical imaging, remote sensing, industrial inspection)
- Labels are noisy, biased, and tied to a fixed set of classes
- Unlabeled images are abundant and cheap (web, cameras, robots)

Definitions:

- **Self-supervised learning (SSL)**: train on a large unlabeled dataset with a task whose labels are **generated from the data itself**.
- **pretext task**: the self-supervised task that defines a supervised-like loss
- **downstream tasks**: the problems we actually want to solve after pretraining

Keep in mind: we don't care about the pretext task. It is only a way to train the representation.

Transferability

What makes a representation "useful"?

- **Transferable**: good performance on many downstream tasks, with little labeled data and little adaptation
  - we can finetune on small downstream tasks (few-shot learning)
- **Semantic**: semantically similar images are close, different images are far
  - Ideally classes are (linearly) separable and the representation is disentangled
- **Invariance**: the representation ignores nuisance factors (lighting, viewpoint, small crops, color shifts) while keeping semantic content
  - note that what is a nuisance depends on the downstream task

SSL methods differ in how they design pretext tasks and architecture to induce these properties

Finetuning

Finetuning

Full Finetuning:

- Replace the pretext head with a task-specific head
- Train the DNN on the downstream labeled data
  - usually with a smaller learning rate for the backbone
  - you can freeze some layers
  - rule of thumb: low layers are generic, high layers are task-specific. You need to finetune high layers, while low layers can be left frozen.

Benefits and Limitations:

- Full finetuning gives the best downstream accuracy, assuming you have enough data to avoid excessive overfitting
- expensive: one full training run per downstream task
- many hyperparameters: learning rate, schedule, augmentations, epochs

Linear probing

- **Freeze** the pretrained encoder f_θ
- Train only a **linear classifier** on top: $\hat{y} = \text{softmax}(W f_\theta(x) + b)$
- Standard benchmark: ImageNet top-1 accuracy of a linear probe on the frozen backbone

Intuition: we have a good representation if the ImageNet classes are linearly separable in feature space

- cheap: features can be precomputed once
- few hyperparameters

Linear probing is the standard evaluation method for SSL methods in classification tasks.

k-NN evaluation

- Embed all the training images with the frozen encoder (usually ℓ_2-normalized)
- For a test image, find its k nearest neighbors (cosine similarity)
- Predict the class by (weighted) majority vote of the neighbors

$$\hat{y} = \arg\max_c \sum_{i \in \mathcal{N}_k(x)} \exp(\text{sim}(z, z_i)/\tau) \mathbf{1}[y_i = c]$$

- **no training**: no hyperparameters except k (and τ)
- measures the **local structure** of the embedding space
- cheap to run during pretraining to monitor progress. Linear probing requires continuously retraining the classifier.

Prototype Learning

Training: one prototype per class, the mean embedding

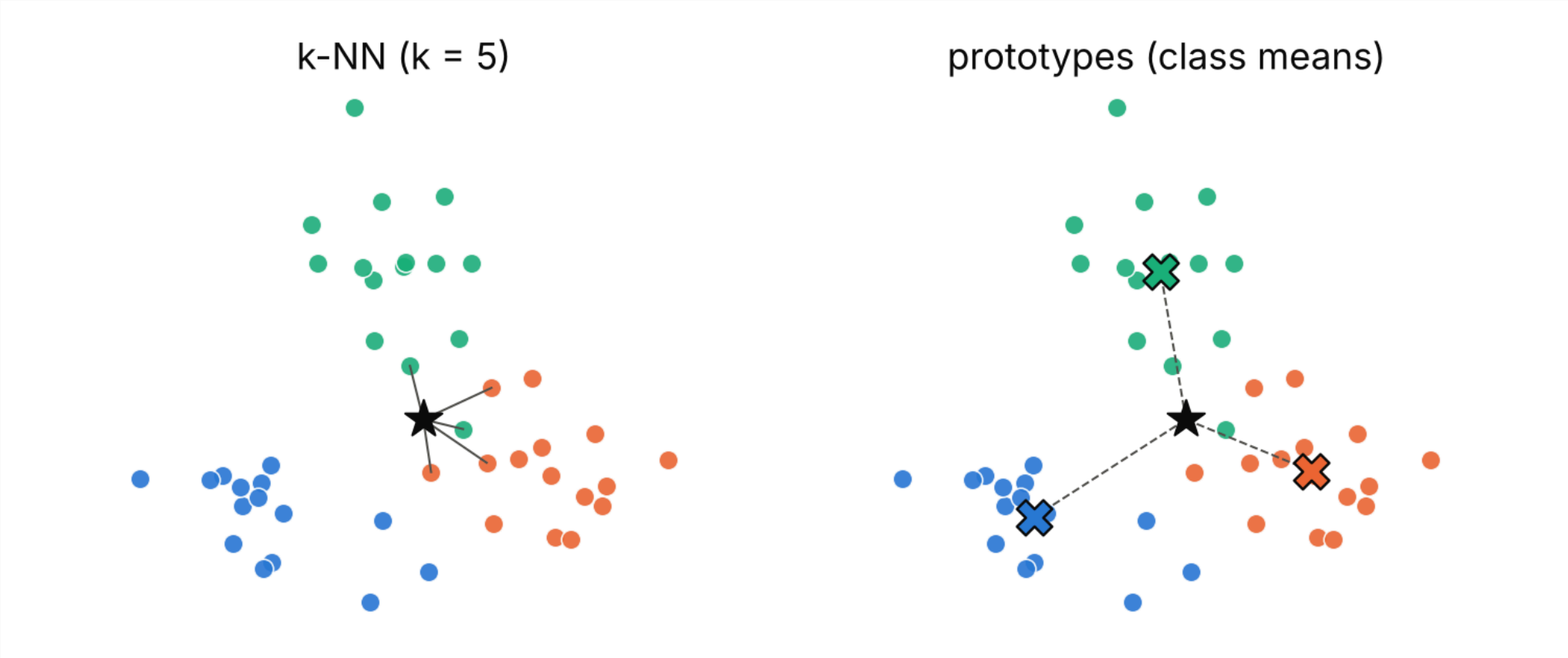
$$\mu_c = \frac{1}{|D_c|} \sum_{x \in D_c} f_\theta(x)$$

Inference: the class of the closest prototype

$$\hat{y} = \arg\min_c \| f_\theta(x) - \mu_c \|^2$$

- also called **nearest class mean** (NCM) classifier
- equivalent to a linear classifier
- works well when classes form compact clusters, and needs very few examples. Very common in few-shot learning.

Prototype Learning



(Early Historical) Examples of Pretext Tasks

Pretext Task

Pretext task: transform the unlabeled image and ask the network to predict a property of the transformation (or of the missing information).

- The label is free: we know the transformation we applied
- Train with a standard supervised loss (cross-entropy, regression)

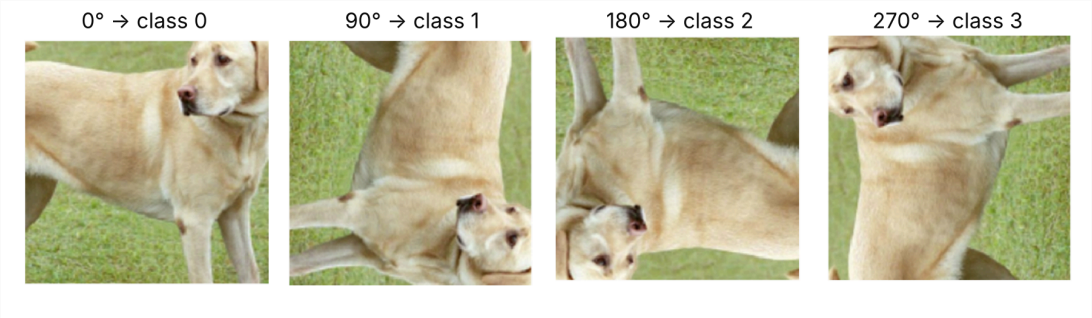
The task must be **solvable**, **not trivial** (no low-level cues such as edges, colors, artifacts), and **semantic** (solving it requires understanding the content).

Examples from other courses:

- masked language modeling in NLP
- autoencoding/reconstruction objectives

Rotation prediction

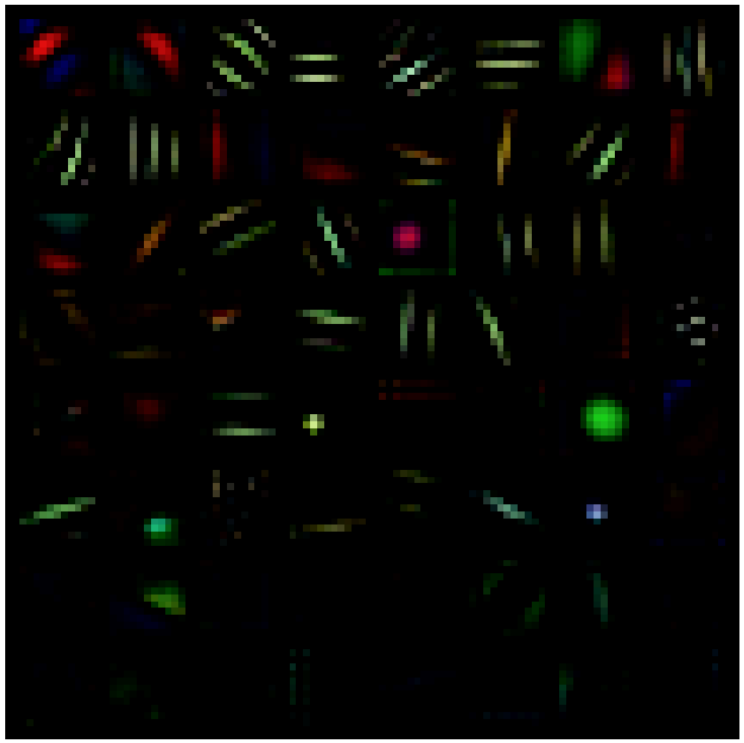
- Rotate each image by $r \in \{0°, 90°, 180°, 270°\}$ and train a CNN to predict r (4-way classification)
- To predict the rotation, the model must recognize objects and their canonical orientation (where is the head, where is the sky, …)



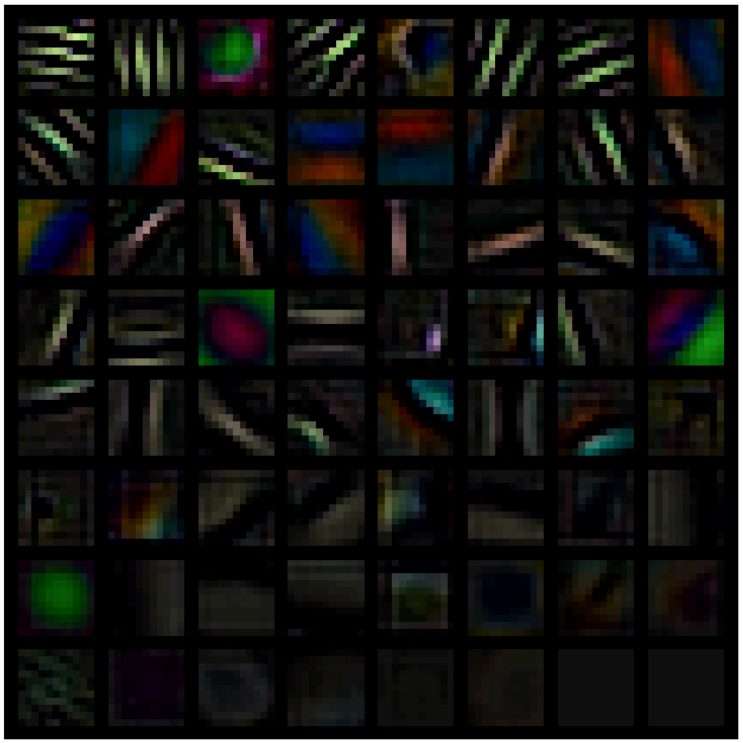
- no artifacts: rotations by multiples of 90° need no interpolation
- fails for images without a canonical orientation (textures, aerial images)

Gidaris et al. "Unsupervised Representation Learning by Predicting Image Rotations." ICLR 2018

Rotation Prediction: Learned Features



(a) **Supervised**



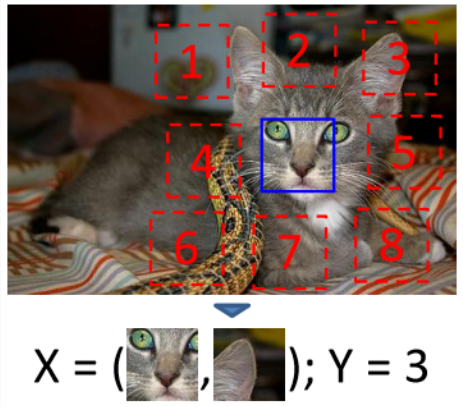
(b) **Self-supervised to recognize rotations**

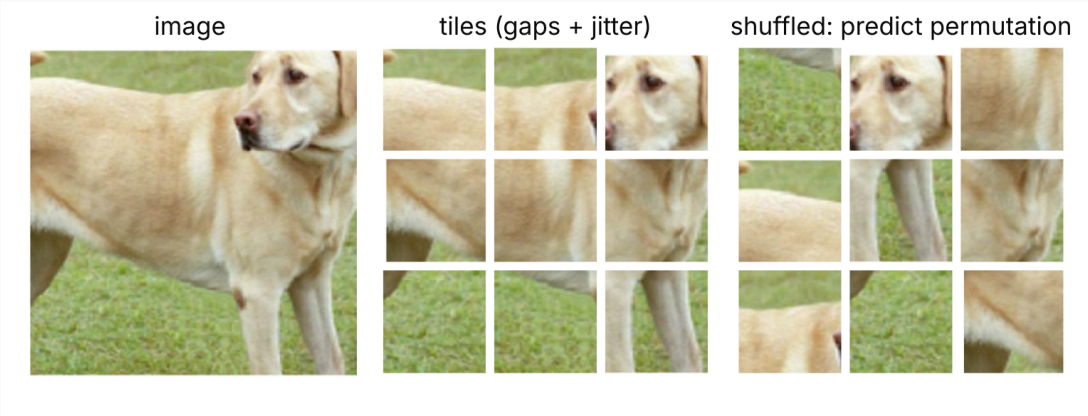
Figure 4: First layer filters learned by a AlexNet model trained on **(a)** the supervised object recognition task and **(b)** the self-supervised task of recognizing rotated images. We observe that the filters learned by the self-supervised task are mostly oriented edge filters on various frequencies and, remarkably, they seem to have more variety than those learned on the supervised task.

Gidaris et al. "Unsupervised Representation Learning by Predicting Image Rotations." ICLR 2018

Jigsaw puzzles

- **Precursor**: predict the relative position of two patches (8 possible neighbors in a 3×3 grid)
- **Jigsaw**: shuffle a 3×3 grid of tiles, predict the permutation
- Avoid trivial solutions: random gaps and jitter between tiles, color jitter, pixelation





Doersch et al. "Unsupervised Visual Representation Learning by Context Prediction." ICCV 2015
Noroozi & Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles." ECCV 2016

Colorization

- Convert the image to Lab color space
- Input: lightness channel L. Target: color channels ab
- Coloring requires semantics: grass is green, sky is blue, but a car can be any color
- **Classification** over quantized ab bins



- natural task with a dense output: learns local and global features
- input statistics differ from downstream (grayscale vs RGB images)

Zhang, Isola & Efros. "Colorful Image Colorization." ECCV 2016

Limitations and Challenges

- **Task-specific features**: the last layers specialize in the pretext task (rotation, permutation, color). Linear probe accuracy often peaks at **intermediate** layers
- **Shortcuts**: the network exploits low-level cues instead of semantics (e.g. chromatic aberration for patch position)
- **Hand-crafted**: each task requires careful engineering and encodes a specific (often arbitrary) inductive bias
- **Gap with supervised**: linear probe accuracy remained far from supervised pretraining

Next step: instead of predicting a property of a transformation, learn to be **invariant** to transformations by **comparing** images. We need both color and spatial transforms.

Shortcut learning

Shortcut learning: The model solves the pretext task with **low-level cues** instead of semantics.

- **Chromatic aberration**: lenses shift color channels toward the borders, revealing patch position
- **Edge continuity**: jigsaw tiles aligned by following lines
- **Color histograms**: crops of the same image share colors
- **Watermarks, text, borders, JPEG artifacts** in web images



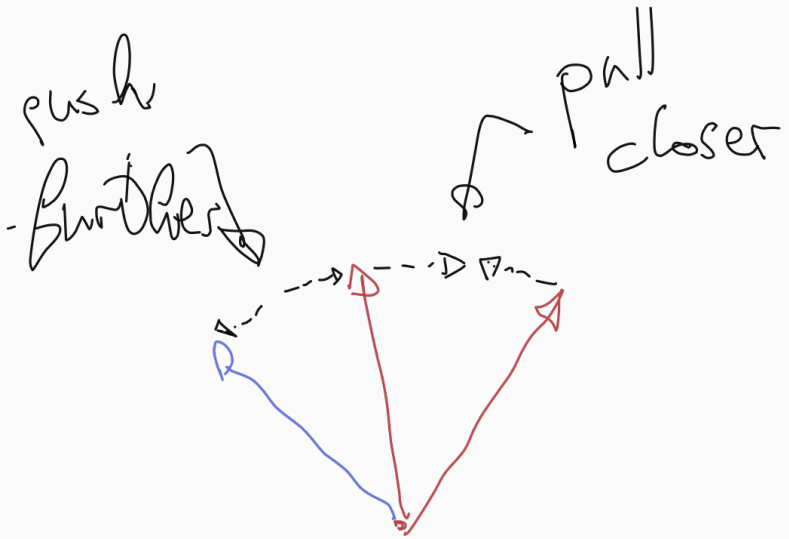
Self-Supervised Contrastive Learning

Contrastive Learning

Objective: learn a latent space that is semantically meaningful for downstream tasks.

Idea: in a semantic space, similar images are close to each other, different images are far.

- compare and contrast **pairs** of images during training
- **positive pairs** → pull them together
- **negative pairs** → push them apart



Design choices:

1. How do we define and sample positive/ negative pairs?
2. Which loss do we use?

Positive and negative pairs

Contrastive losses work in both supervised and self-supervised settings.

Supervised

- positive: samples from the **same class**
- negative: samples from **different classes**
- e.g. face recognition, person re-identification

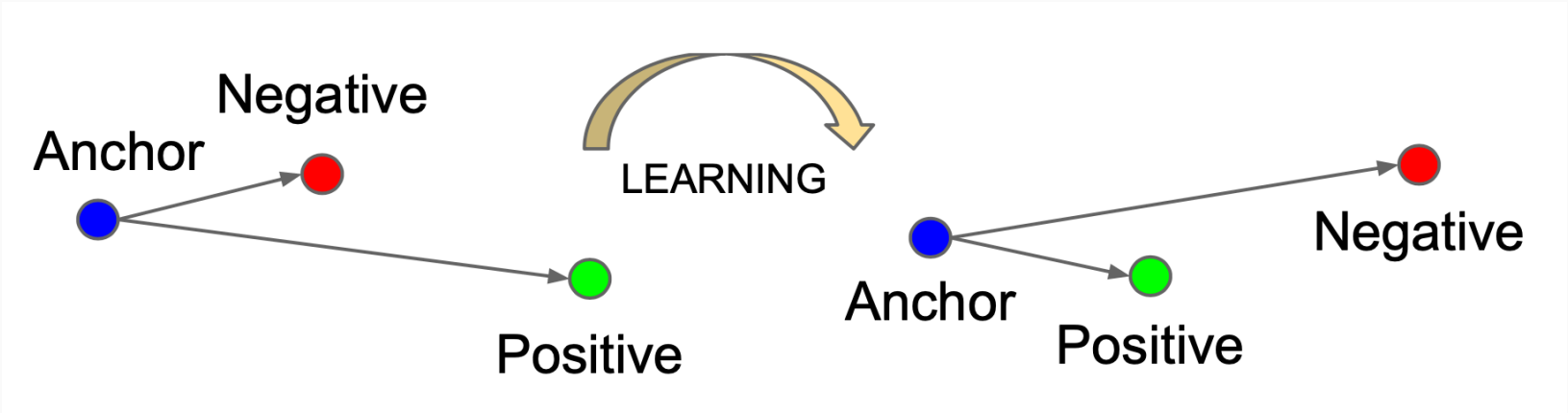
Self-supervised

- no labels: we must **define** the pairs
- positive: two augmented views of the **same image**
- negative: views of **different images**

Triplet Loss

Simplest example of contrastive learning: the **triplet loss** (anchor a, positive p, negative n, margin α)

$$\mathcal{L} = \max\left(0, \|z_a - z_p\|^2 - \|z_a - z_n\|^2 + \alpha\right)$$

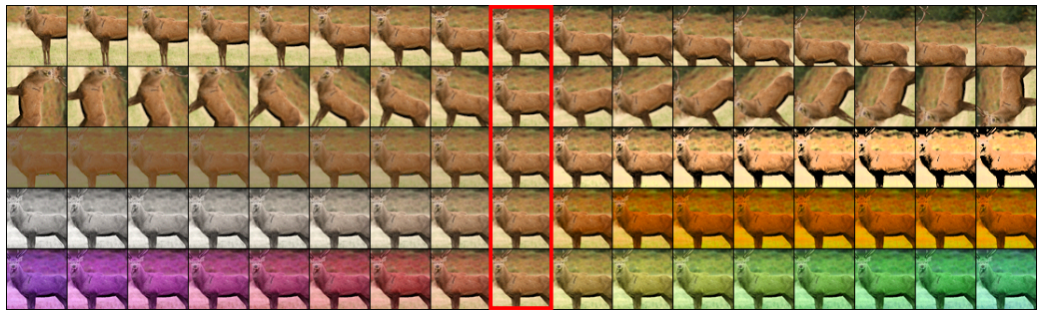


Schroff et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering." CVPR 2015

Instance Discrimination

Pretext task: each image is its own class (a **surrogate class**).

- Two random augmentations t, t' of an image x give two **views** $t(x)$, $t'(x)$: a **positive pair**. Views of other images are **negatives**
- The model must extract features that identify the image while being invariant to the augmentations
- ExemplarCNN: one class per image. Modern methods (SimCLR, MoCo) compare embeddings directly
- instance discrimination is the main SSL task used in modern contrastive methods

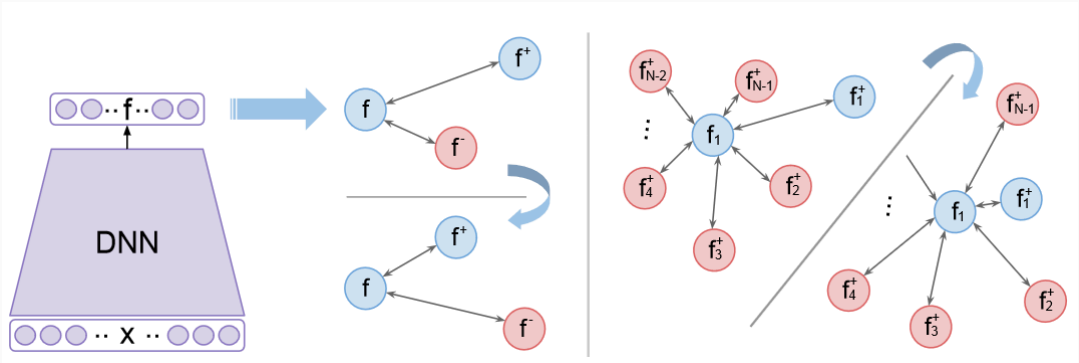


Dosovitskiy et al. "Discriminative Unsupervised Feature Learning with Exemplar CNNs." NeurIPS 2014
Wu et al. "Unsupervised Feature Learning via Non-Parametric Instance Discrimination." CVPR 2018

The Role of Negative Pairs

Why do we need to push negatives apart?

- With positives only, there is a **trivial solution**: map every image to the same point (**representation collapse**)
- Negatives provide **contrast**: the model must find features that distinguish an image from the others
- **Hard negatives** (similar embeddings but different images) give the most informative gradients
- One negative at a time (triplet) converges slowly → compare with **many** negatives at once (N-pair loss, InfoNCE)



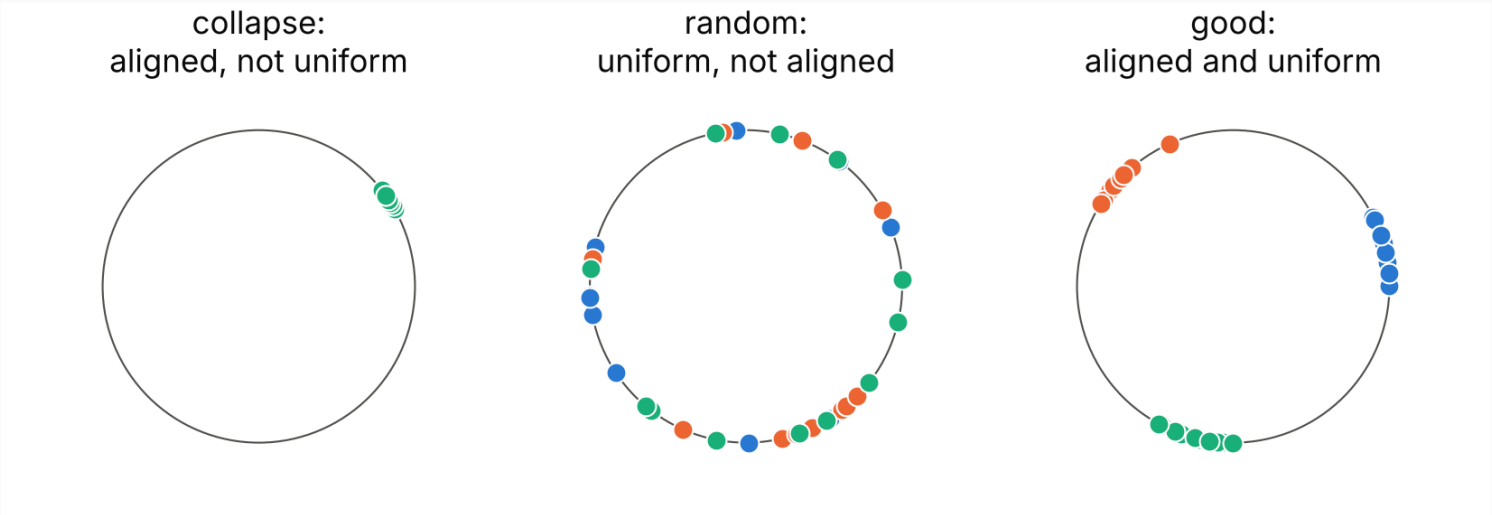
Sohn. "Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective." NeurIPS 2016

Embedding Space Geometry

Some geometric intuition about ℓ_2-normalized embeddings. Embeddings lie on the **unit hypersphere**. We use cosine similarity.

Ideal properties:

- **Alignment**: positive pairs are close
- **Uniformity**: embeddings spread over the sphere
- Positives give alignment, negatives give uniformity. Semantic classes become clusters on the sphere



Wang & Isola. "Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere." ICML 2020

Augmentations and Invariance in SSL

Augmentation and Invariance

With instance discrimination tasks, the contrastive objective asks for $f(t(x)) \approx f(t'(x))$ for all $t, t' \in \mathcal{T}$.

- In SSL: The augmentation set $\mathcal{T}$ **defines the invariances** of the representation. It is the main **inductive bias** of contrastive SSL
- In supervised DNNs: the labels tell the model what is semantic and what is nuisance
- In classic CV: hand-crafted features (e.g. SIFT) were designed to be invariant to scale, rotation, illumination. Here invariances are **learned from the augmentations**

Examples

Augmentation	Induced invariance
Crop + resize	scale, translation, occlusion, part → whole
Flip	left-right orientation
Color jitter	lighting and camera changes
Grayscale	color: forces shape and texture
Blur	high-frequency details, noise
Solarize	nonlinear intensity changes



Modern SSL methods use long augmentations pipelines to implicitly encode several invariances.

Challenges

Invariance is useful only if the downstream task does not need that information.

- **Color** invariance hurts fine-grained tasks where color matters (flowers, birds species)
- **Rotation** invariance hurts when orientation is semantic (digits 6 vs 9, scene layout)
- **Aggressive crops** may produce views with no shared object: the positive pair is not semantically related (false positives)
- Too weak augmentations: the task is trivial, the model learns shortcuts

InfoNCE Loss

InfoNCE loss: definition

Given an anchor z, a positive z^+ and negatives $z_1^-, \ldots, z_{N-1}^-$

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\text{sim}(z, z^+)/\tau)}{\exp(\text{sim}(z, z^+)/\tau) + \sum_{j=1}^{N-1} \exp\big(\text{sim}\big(z, z_j^-\big)/\tau\big)}$$

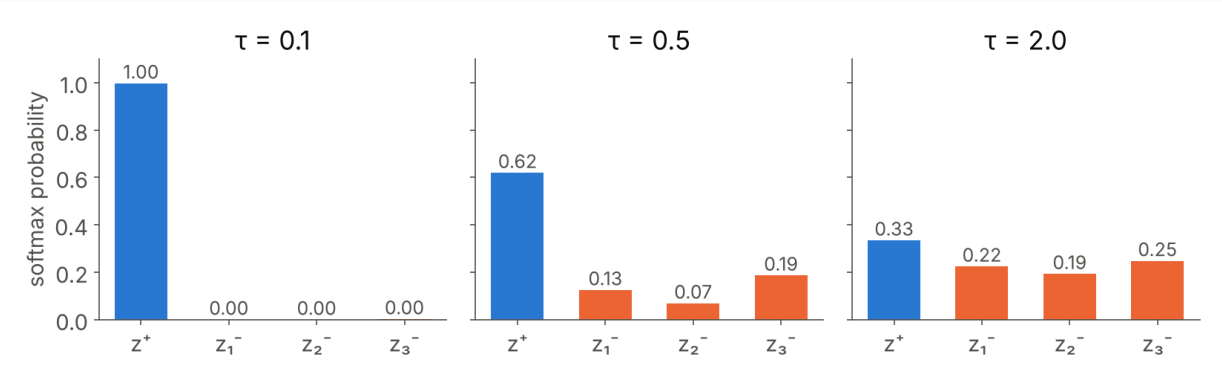
- $\text{sim}(u, v) = u^\top v/(\|u\| \, \|v\|)$ is often the cosine similarity
- τ is the temperature
- it is a **cross-entropy** loss over N "classes", where the correct class is the positive
- the gradient pulls z toward z^+ and pushes it away from each z_j^-, weighted by its softmax probability: hard negatives get larger weight

Intuition: similar to the standard softmax + crossentropy, but adapted to a context where we don't have supervised classes.

- the positive and negative instances take the role of the classes
- pairwise similarities take the role of the logits as input to the softmax

Temperature

as in the standard softmax, τ scales the logits before the softmax



- **Small τ**: sharp, focuses on the **hardest** negatives
- **Large τ**: flat, all negatives weigh the same, weak gradients

Wang & Liu. "Understanding the Behaviour of Contrastive Loss." CVPR 2021

Negatives Pairs

- SotA methods often use a very large number of negatives. SimCLR uses batches >4096.
- More negatives → higher chance of including **hard negatives**, more informative gradients
- Better coverage of the data distribution → more uniform embeddings

Challenge:the number of negatives is tied to the **batch size**. Memory grows linearly, the similarity matrix quadratically. Augmentation pipelines are expensive and CPU-bounded.

Representation collapse

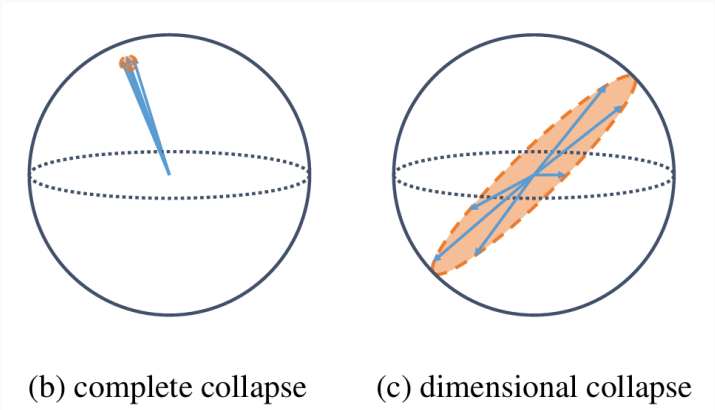
Complete collapse: $f(x) = c$ for every x.

- Positive pairs are perfectly aligned, the loss is zero
- The representation contains no information
- Symptoms: embedding std across the batch $\to 0$, k-NN accuracy at chance level

Dimensional collapse: embeddings span only a low-dimensional subspace.

- Not detected by the loss, wastes capacity
- Symptom: fast-decaying singular values of the embedding covariance

The main challenge of SSL: learn high-quality representations while **avoiding collapse**.



(b) complete collapse (c) dimensional collapse

Jing et al. "Understanding Dimensional Collapse in Contrastive Self-supervised Learning." ICLR 2022

SimCLR

Instance Discrimination: maximize the agreement between differently augmented views of the same image, via a contrastive loss in the latent space.

Main findings:

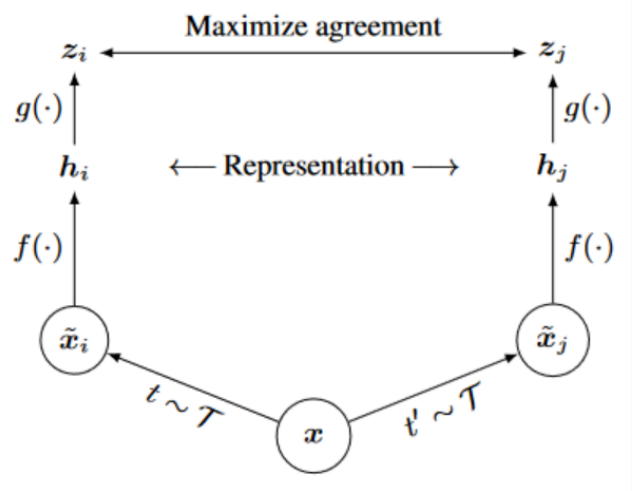
1. The **composition** of data augmentations plays a critical role in defining an effective pretext task
2. A learnable nonlinear **projection head** between the representation and the loss substantially improves the representation
3. Contrastive learning benefits from **larger batch sizes** and **longer training** than supervised learning

Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations." ICML 2020

SimCLR architecture

For each image x in a batch of size B:

1. two augmentations: $\tilde{x}_i = t(x)$, $\tilde{x}_j = t'(x)$
2. **encoder** f (e.g. ResNet-50): $h_i = f(\tilde{x}_i)$, used downstream
3. **projection head** g (MLP): $z_i = g(h_i)$, used only by the loss
4. **NT-Xent loss** over the $2B$ views



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

Each view has 1 positive and $2B - 2$ negatives (the other views in the batch).

Note that the NTXent is just a specific instantiation of InfoNCE.

Projection Head

$$x \xrightarrow{f} h \xrightarrow{g} z, \quad g(h) = W_2 \, \mathrm{ReLU}(W_1 h)$$

- The loss is computed on z, but downstream we use h (the projection head is discarded)
- A linear probe on h works better than on z. Intuition:
  - the contrastive loss makes z invariant to the augmentations, removing information (color, orientation, ...) that may be useful downstream
  - g is responsible for this loss of information while h keeps a richer representation
- the same "separate encoder and projector" design is used in almost all modern SSL methods

SimCLR data augmentation pipeline

simplified pipeline (applied independently to each view):

1. random crop with resize and horizontal flip
2. color distortion: color jitter and random grayscale
3. Gaussian blur

Empirical findings:

- no single augmentation is enough
- **crop + color distortion** is the most important pair
- without color distortion, crops of the same image can be matched by their color histograms (shortcut learning)
- contrastive learning needs stronger color augmentation than supervised learning

SimCLR training requirements

- **Large batches** are required to have a large number of negatives
- **Long training** schedules (hundreds of epochs)

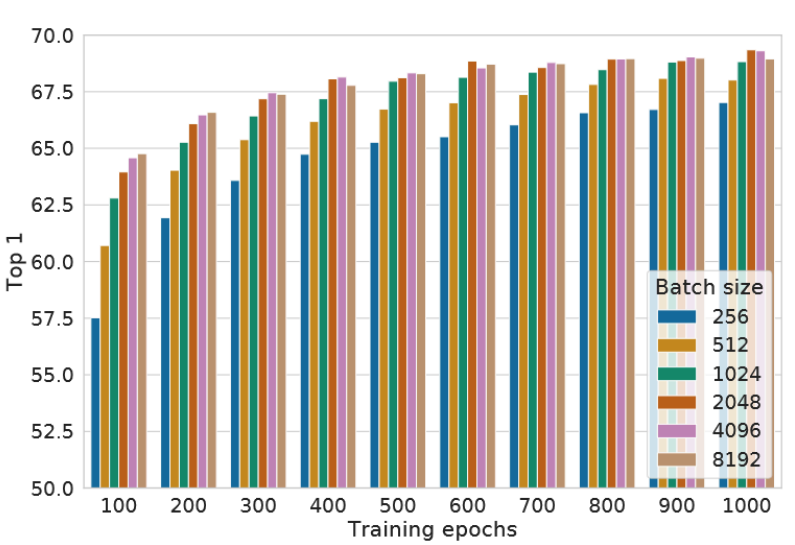


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

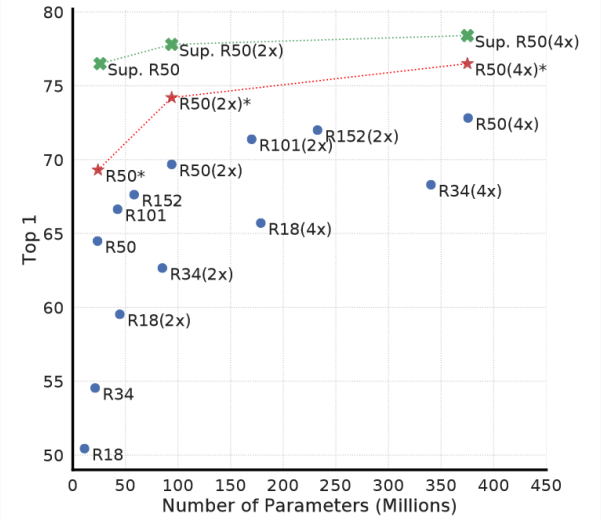


Figure 7. Linear evaluation of models with varied depth and width. Models in blue dots are ours trained for 100 epochs, models in red stars are ours trained for 1000 epochs, and models in green crosses are supervised ResNets trained for 90 epochs[7] (He et al., 2016).

SimSiam

- **SimSiam**: A simple contrastive SSL method
- **Loss**: negative cosine similarity (symmetrized)

$$\mathcal{D}(p_1, z_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{z_2}{\|z_2\|_2}$$

- **Stop-grad** blocks the gradient. The two encoders are the same but the learning signals flows only in one direction
- **Projector** only in the first augmentation

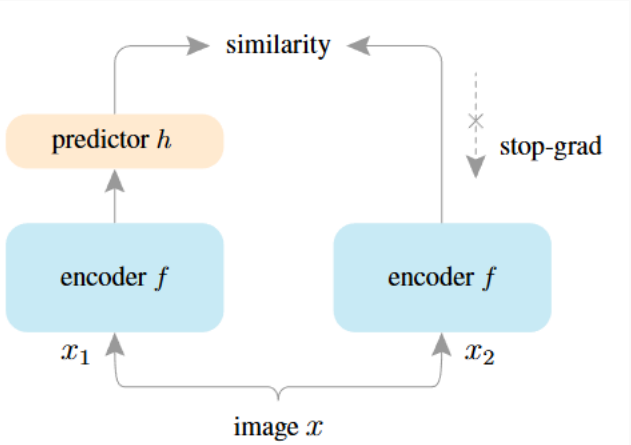


Figure 1: source: Xinlei Chen Kaiming He, 'Exploring Simple Siamese Representation Learning'.

SimSiam - Pseudocode

Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# f: backbone + projection mlp
# h: prediction mlp

for x in loader: # load a minibatch x with n samples
   x1, x2 = aug(x), aug(x) # random augmentation
   z1, z2 = f(x1), f(x2) # projections, n-by-d
   p1, p2 = h(z1), h(z2) # predictions, n-by-d

   L = D(p1, z2)/2 + D(p2, z1)/2 # loss

   L.backward() # back-propagate
   update(f, h) # SGD update

def D(p, z): # negative cosine similarity
   z = z.detach() # stop gradient

   p = normalize(p, dim=1) # l2-normalize
   z = normalize(z, dim=1) # l2-normalize
   return -(p*z).sum(dim=1).mean()
```

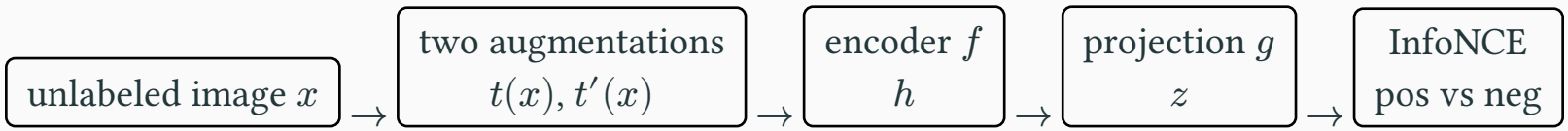
Figure 2: source: Xinlei Chen Kaiming He, 'Exploring Simple Siamese Representation Learning'.

Conclusions

Pretext Tasks in Contrastive Learning

	Pretext tasks	**Contrastive learning**
Objective	predict a property of the transformation	be invariant to the transformation
Signal	hand-designed labels (rotation, permutation, color)	comparisons between instances
Features	specialized to the pretext task	generic, semantic
Inductive bias	one specific task	the augmentation set
Performance	far from supervised	close to (or better than) supervised

Self-Supervised Contrastive Learning



- **Augmentations** define the invariances (positive pairs)
- **Negatives** prevent collapse
- The **encoder** f is kept for downstream tasks, the projection head g is discarded
- **Evaluation**: linear probing transfer to classification

Practical Tips

- **Batch size and negatives**: small batches mean few negatives and weak contrast. If you cannot afford large batches, use a queue (MoCo) or a method without negatives
- **Augmentations matched to the downstream task**: do not remove information you will need (color for species, orientation for text and medical images). Default ImageNet pipelines do not transfer to every domain
- **Linear probe results**: they depend on the probe hyperparameters and favor linearly separable features. k-NN, full finetuning and low-shot can have different results
- **Hyperparameters**: settings optimized for supervised training do not transfer to SSL (learning rate, epochs, weight decay)
- **Data loading**: heavy augmentations on CPU can be the bottleneck. Move them to the GPU or use optimized pipelines

Further Reading

- "A Cookbook of Self-Supervised Learning" https://arxiv.org/abs/2304.12210
  - more details about training contrastive SSL methods and different losses

Key Takeaways

- SSL learns representations from unlabeled data with self-generated labels: the pretext task is only a tool
- **Contrastive learning**: pull positive pairs together, push negative pairs apart. In SSL, positives are augmented views of the same image (instance discrimination)
- **Augmentations** are the main inductive bias: they define what the model becomes invariant to
- **InfoNCE**: a cross-entropy over one positive and many negatives, a lower bound on mutual information. Temperature controls the focus on hard negatives
- **SimCLR**: in-batch negatives, needs large batches.
- **Evaluation**: linear probing, k-NN, prototypes, finetuning, transfer to dense tasks
- Watch out for **collapse** and **shortcuts**

Next Lectures

- masked/distillation paradigms and SotA SSL methods
- lab on SSL