Self-Supervised Learning and Vision Foundation Models

Giacomo Cignoni - PhD student

giacomo.cignoni@phd.unipi.it

Part 1

Self-Supervised Learning

how to learn without labels

Definition

Self-Supervised Learning (SSL) is a training paradigm that aims to teach a network to extract meaningful **representations** from unlabeled data, with the learning signal derived from the data structure itself.

Unsupervised vs Self-Supervised learning

	Classic Unsupervised learning	Self-supervised learning
Main goal	Discover and expose structure in data	Train a network to learn transferable representations
Training signal	Structure-based criteria: distances, variance, likelihood	Prediction targets generated from the data itself
Typical output	Clusters, components, maps or density estimates	A pretrained encoder for downstream tasks
Examples	k-means, PCA, SOM, DBSCAN, …	*we will see them in this lesson*

> Classical methods organize the data based on their structure.
>
> Self-supervised methods use data structure to **train a reusable network**.

Why self-supervised learning?

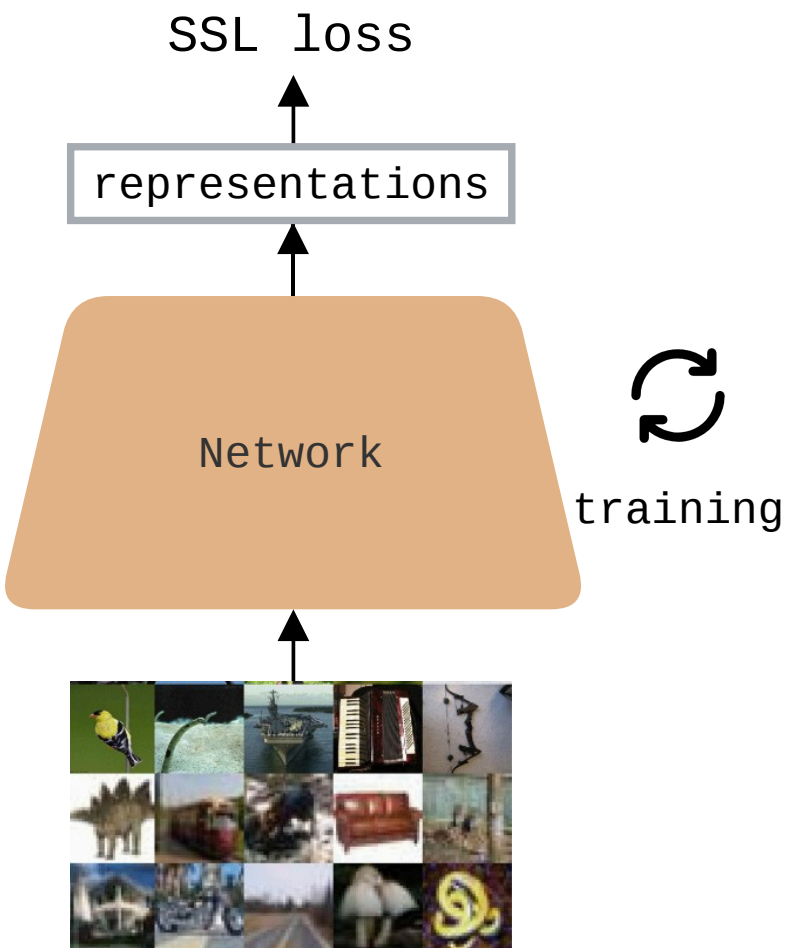
> **Pretrain a network on unlabeled data**, then adapt it to downstream tasks.

Pretraining provides:

- Reusable visual representations.
- A good initialization for faster and better convergence.
- Reduced need for downstream data and labels.

Good representations can also be useful **directly**: in **image retrieval**, feature similarity helps find related images.

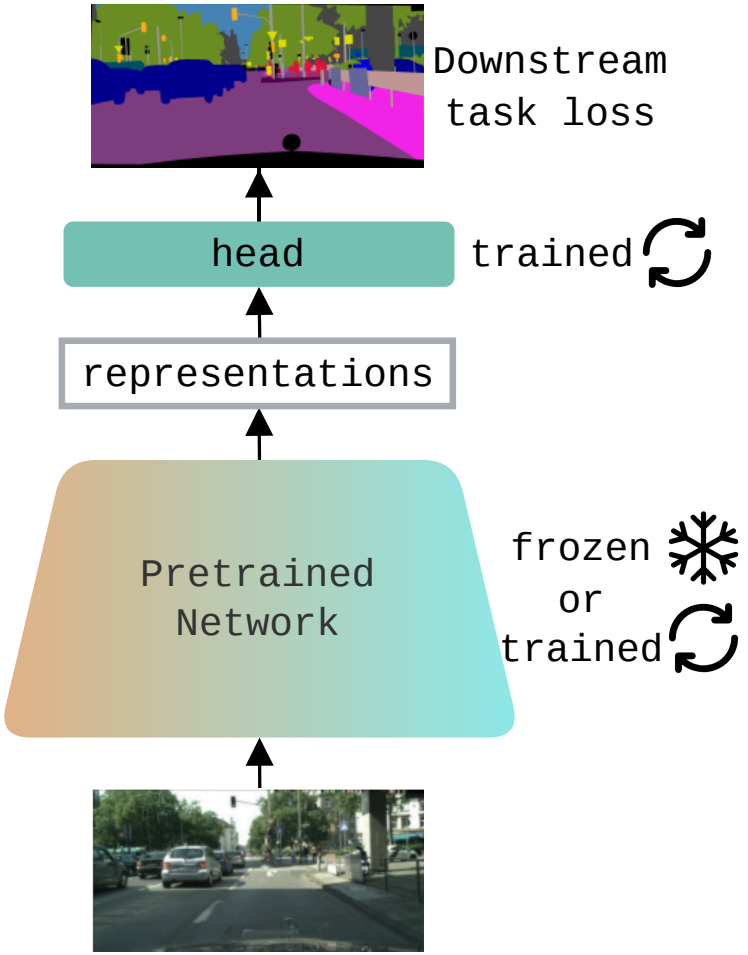
Stage 1: self-supervised pretraining



Train an encoder network on **unlabeled images** via Self-Supervised learning.

> The network learns **general**, **task-agnostic** representations from the pretraining data.
>
> Reuse the pretrained network later.

Stage 2: downstream tasks



Reuse the pretrained encoder and train a **task-specific head** using labeled data.

> The encoder provides **representations**.
>
> The head converts them into **predictions** for the downstream task.
>
> Keep the encoder **frozen** or **fine-tune** it.

Downstream examples: classification, segmentation, depth estimation, ...

How to evaluate representation quality?

Protocol	What is trained?	Criteria
k-NN	Nothing	On the downstream task, for each test sample, assign it to the majority label of the K nearest representations of downstream training samples.
Linear probe	Linear head only	Train a linear head only on downstream data.
Fine-tuning	Head and backbone	Train both linear head and backbone on downstream data.

Designing a useful pretext task

A task with automatically generated targets must still teach **useful visual representations**.

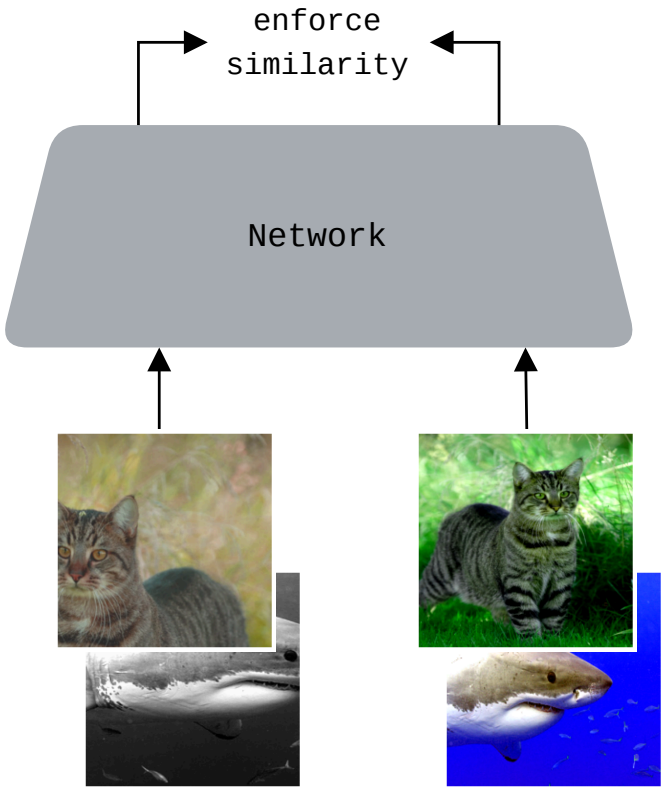
- **Choose the right difficulty**
  An easy task teaches little; an ambiguous task may be impossible.
- **Avoid shortcuts**
  The network may exploit borders, textures or camera artifacts instead of understanding objects.
- **Encourage transfer**
  Solving the pretext task should help with other visual tasks.

> Low pretext-task loss does not necessarily mean good downstream representations.

Part 1 · Self-Supervised Learning

Instance Discrimination Methods

Instance discrimination



Generate **two views of the same image** using random augmentations t_1 and t_2.

$$v_1 = t_1(x), \quad v_2 = t_2(x)$$

Encode both views with the **same network** f.

$$z_1 = f(v_1), \quad z_2 = f(v_2)$$

> Encourage **similar representations** for the two views, despite changes in appearance.

Agreement alone admits a trivial solution

two augmentations: $v_1 = t_1(x), \quad v_2 = t_2(x)$

$$L_{\text{agree}} = \|f(v_1) - f(v_2)\|^2$$

Complete collapse

$$f(x) = c \quad \forall x$$

Every image receives the same constant representation c.

Dimensional collapse

$$\text{rank}(\text{Cov}(z)) \ll d$$

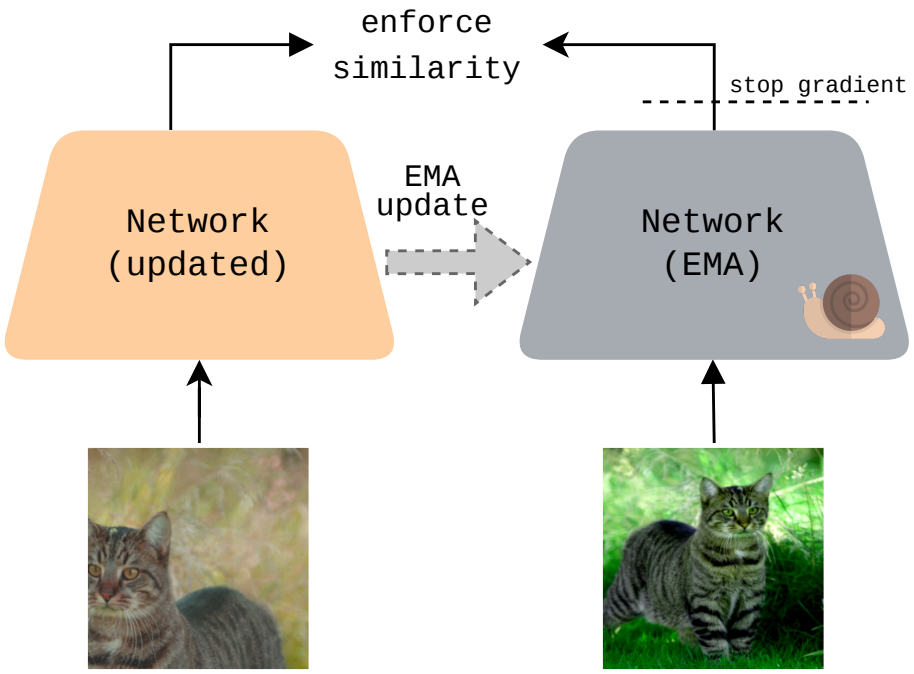
Features z with dimension d use only a small part of the available space.

Contrastive and siamese methods: recap

Method	Training signal	Key mechanism
SimCLR	make positive views closer, move away from negatives	Contrastive loss
SimSiam	Predict the other view	Predictor + stop-gradient

> Today: how teachers and explicit regularizers make label-free targets useful.

BYOL: learning from another network



BYOL: Bootstrap Your Own Latent

- Create two views of the same image.
- Train a **student** to predict a **teacher's** output.
- Update the teacher slowly from the student.

> Learn useful representations by predicting another view's features, **without negative pairs**.

Student and teacher: different roles

Role	Student: online network	Teacher: target network
Purpose	Predict the other view’s features	Provide the target features
Learning rule	Backpropagation and optimizer	Moving average of student weights
Output	A prediction to improve	A target for the current step

The teacher starts as a **copy of the student**.

> The teacher supplies supervision without human labels. Its targets evolve as training progresses.

BYOL loss

$$L = \|s - \texttt{stop_grad}(t)\|^2$$

s: student prediction t: teacher target

$$\frac{\partial L}{\partial s} = 2(s - t), \quad \frac{\partial L}{\partial t} = 0$$

- The teacher computes a target from the other view.
- The optimizer updates **only the student** through this loss.
- **Stop-gradient** blocks backpropagation into the teacher.

> How can change over time without receiving gradients from the prediction loss?

The teacher is a moving average

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

θ: student weights ξ: teacher weights τ: momentum

Example: one weight

Teacher: 2.00

Updated student: 3.00

Momentum: $\tau = 0.99$

$$\xi^{\text{new}} = 2.01$$

Why average?

- Smooth rapid student updates.
- Make targets change gradually.
- Retain information from earlier students.

> The student learns by gradients. The teacher follows through an exponential moving average.

How BYOL addresses collapse

Matching two views alone allows a trivial solution:

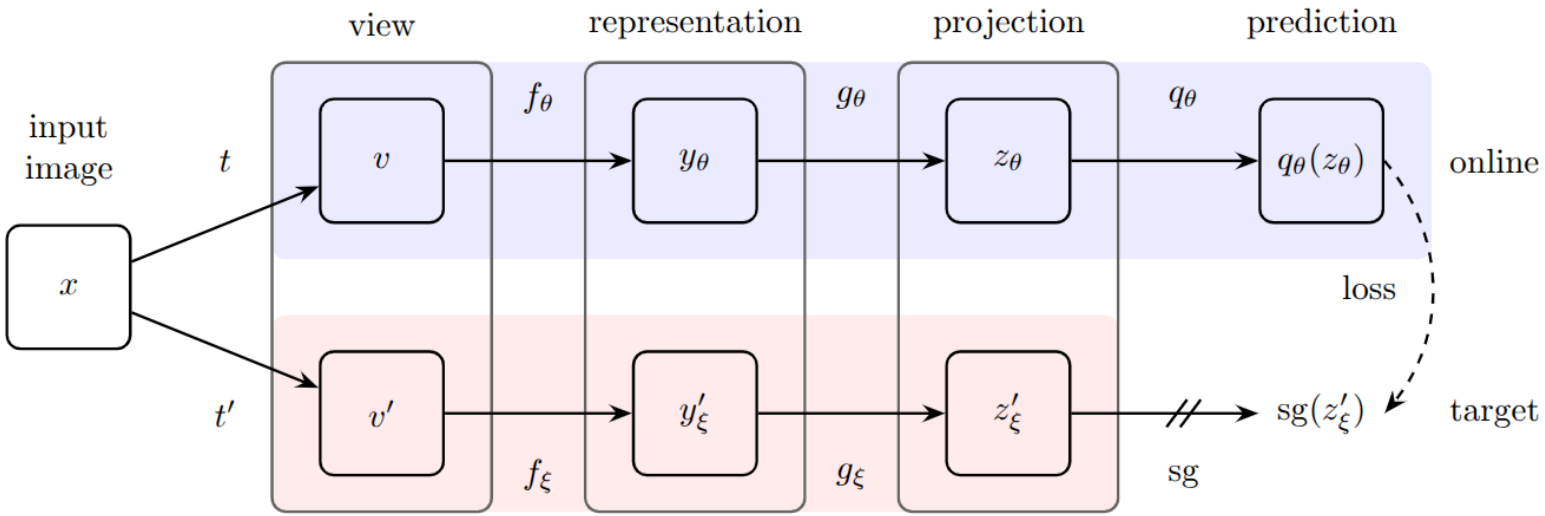
$$f(x) = c \quad \forall x$$

BYOL changes the learning dynamics through:

- **A student-only predictor head**
  The two branches perform different computations.
- **Stop-gradient**
  The teacher cannot directly adjust to reduce the current loss.
- **A slowly updated teacher**
  Targets follow the student's history rather than its current gradient.

> This complete recipe avoids collapse empirically.

BYOL: the complete architecture



Encoder f: extracts the representation y.
Projector g: maps it into the space used for SSL training.
Predictor q: predicts the teacher's projection; present only in the student.

After pretraining: keep the student encoder f_θ; discard the projector and predictor.

BYOL: do not forget normalization!

For two views v and v' of the same image:

$$p_\theta = q_\theta(g_\theta(f_\theta(v))), \quad z'_\xi = g_\xi\big(f_\xi(v')\big)$$

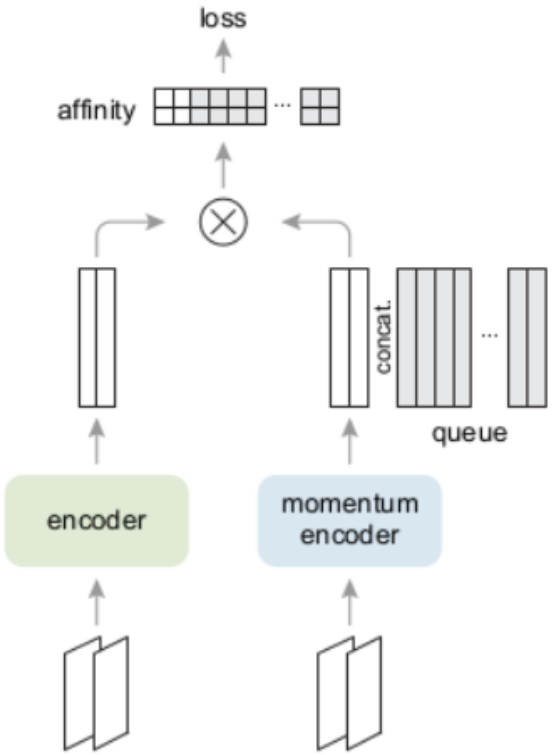
Compare the student prediction with the teacher target, **after L_2 normalization**:

$$L_{v,v'} = \left\| \frac{p_\theta}{\|p_\theta\|} - \mathrm{sg}\left(\frac{z'_\xi}{\|z'_\xi\|} \right) \right\|^2$$

Swap the views so that the student learns from both directions:

$$L_{\mathrm{BYOL}} = L_{v,v'} + L_{v',v}$$

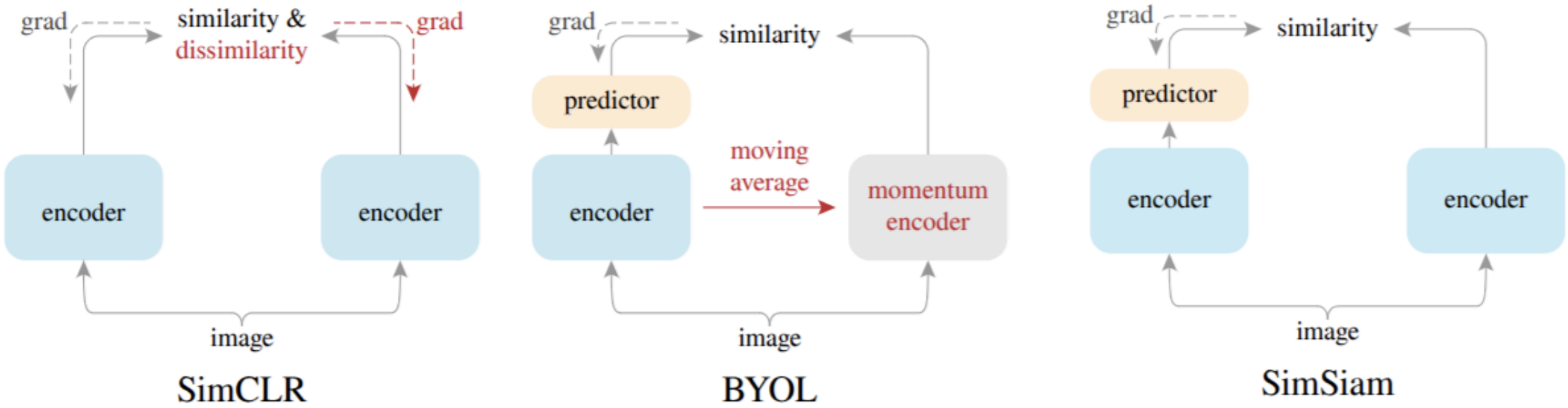
MoCo: Momentum Contrast



- **Momentum teacher:** its weights follow an EMA of the student like BYOL.
- **Contrastive loss:** like SimCLR, with negatives.
- **FIFO queue:** stores negative representations from previous minibatches. Add new ones and remove the oldest.

> The queue allows many negatives without requiring a large minibatch.

Summary: learning from two views



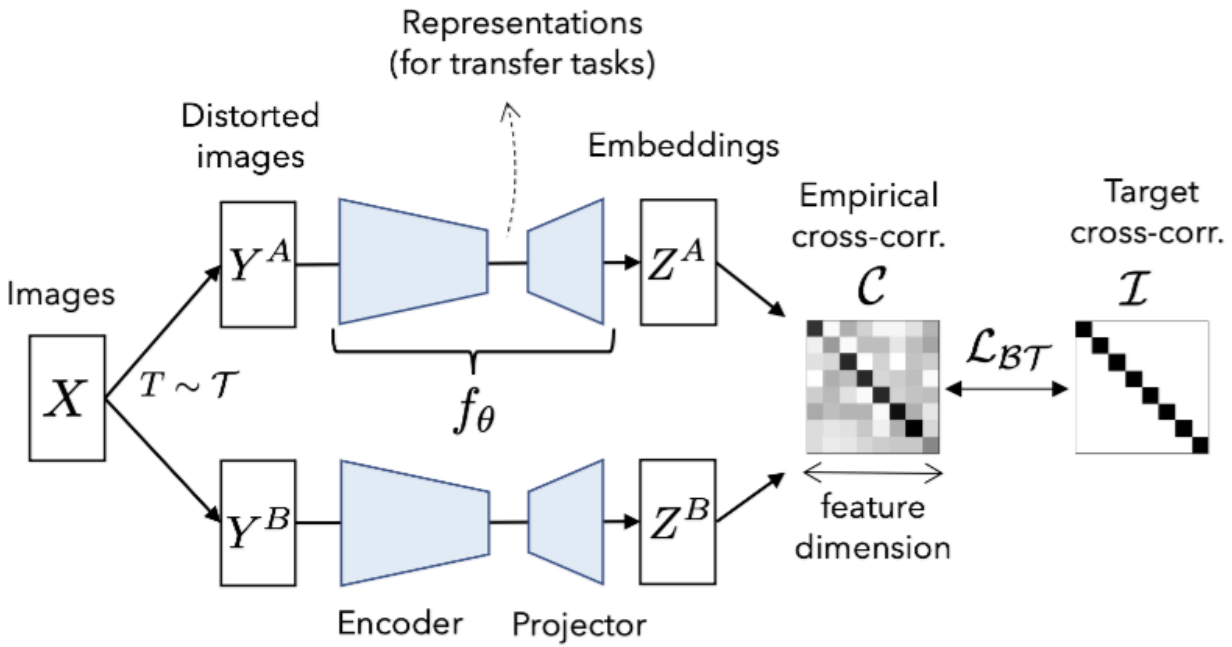
Preventing collapse with feature statistics

Another approach is to **directly constrain the feature statistics**.

- **Across views: agreement**
  Preserve the representation when the image is augmented.
- **Across images: variation**
  Avoid assigning the same representation to every image.
- **Across dimensions: less redundancy**
  Avoid copying one signal into every feature coordinate.

> **Barlow Twins** and **VICReg** encode these goals in the loss using minibatch statistics, without negative pairs or an EMA teacher.

Barlow Twins



For each batch of B images, encode two augmented views using the **same encoder and projector**.

The embeddings form two tensors $Z^A, Z^B \in \mathbb{R}^{B \times d}$, that are **normalized**.

Then compute **empirical cross-correlation** of the minibatch:

$$C = \left(Z^A\right)^\top \frac{Z^B}{B}$$

C has dimension $d \times d$.

Barlow Twins: the loss

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

Enforce C to be similar to the identity matrix I.

Diagonal: $C_{ii} \approx 1$

The same feature should be consistent to the 2 views.

> **Preserve information across augmentations.**

Off-diagonal: $C_{ij} \approx 0$

Different features should carry uncorrelated signals across images.

> **Avoid repeating information across dimensions.**

Two ways to lose information

1. All images look identical

Every image receives the same representation.

Features are constant across the batch: after centering, their values are zero.

$$C = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

The diagonal is **0 instead of 1**.

The **invariance term** penalizes the loss of variation across images.

2. Features repeat the same signal

Images still receive different representations, but every feature copies one varying signal.

All feature dimensions are perfectly correlated.

$$C = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

The off-diagonal is **1 instead of 0**.

The **redundancy term** penalizes the repeated information.

VICReg: three explicit constraints

$$L = \lambda s(Z, Z') + \mu(v(Z) + v(Z')) + \nu(c(Z) + c(Z'))$$

Term	Constraint	Effect / motivation
Invariance s	$\text{mean}_b \|z_b - z'_b\|^2$	Make paired views agree.
Variance v	$\text{mean}_j \max\left(0, \gamma - \sigma_{j(Z)}\right)$ σ_j: feature standard deviation across images. γ: minimum target.	Prevent constant features: each coordinate should vary between images.
Covariance c	**Within-view** covariance: $\frac{1}{d}\sum_{i \neq j} \text{Cov}(Z)_{ij}^2$	Reduce redundant signals across feature dimensions, separately in each view.

Part 1 · Self-Supervised Learning

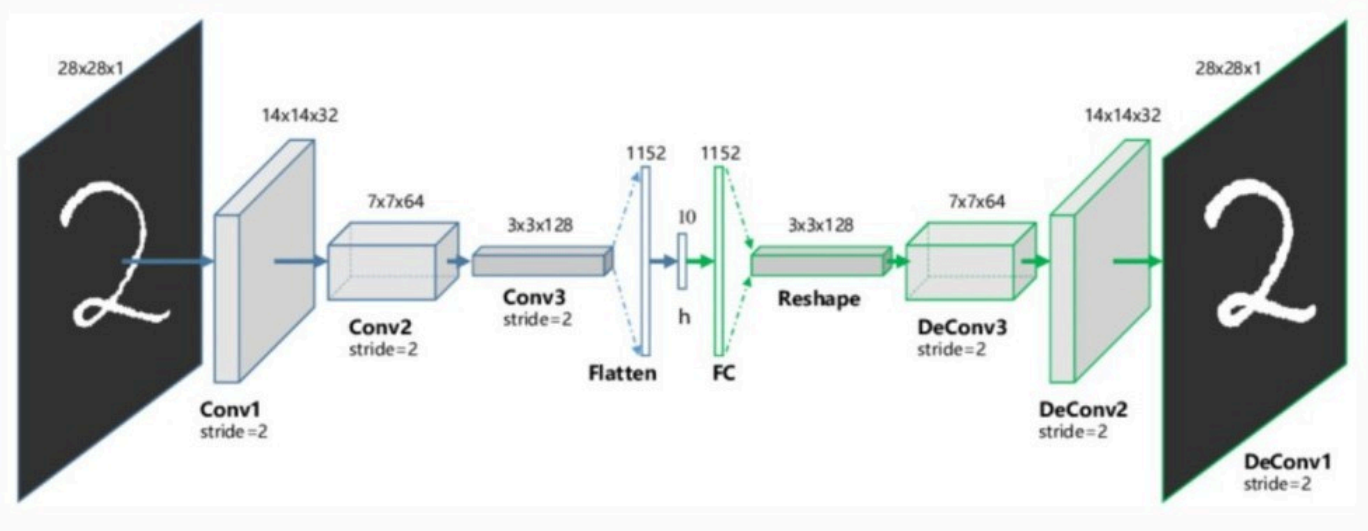
Reconstruction Models

Historical reconstruction: Autoencoders

$$\hat{x} = d_\varphi(f_\theta(x)), \quad L = \|\hat{x} - x\|^2$$

f_θ: encoder d_φ: decoder

- **Target:** reconstruct the original image.
- **Bottleneck:** compress information into a compact representation.



Masci et al., *Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction*, ICANN, 2011.

Masking: learning from missing information

“The cat sits on the ____.”

1. Hide a part of the observation.
2. Use the visible context to predict the missing content.
3. Learn from the original observation as the target.

BERT uses this idea for **masked language modeling**: predict masked tokens from surrounding text.
maybe you have encountered it in the NLP course?

> For **images**, the design question is what to predict: pixels, discrete codes or learned features.

Image patches become transformer tokens

Vision Transformers (**ViT**) split an image into **patches**, (often 16×16 pixels). Each patch is projected into a **token vector**.



Image patches



Masked patches

Image size: $H \times W$
Patch size: $P \times P$

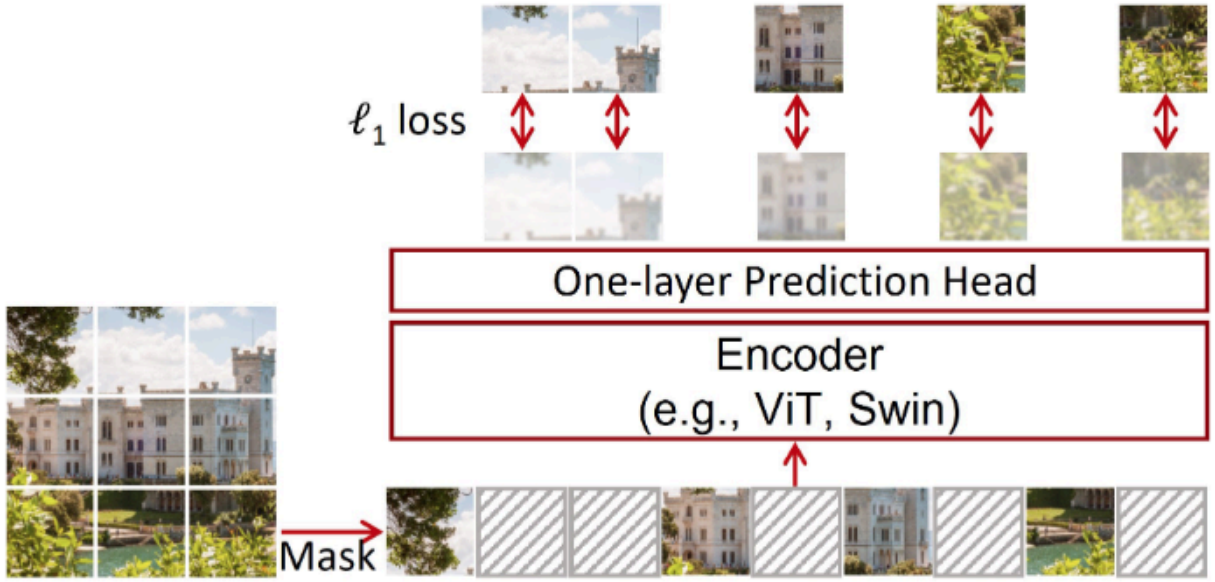
$$N = \frac{HW}{P^2}$$

Example: 224×224 image, 16×16 patches.

$$14 \times 14 = 196 \quad \text{tokens}$$

> Masked image modeling hides some patches and learns to predict information about the missing regions.

SimMIM: predicting masked pixels

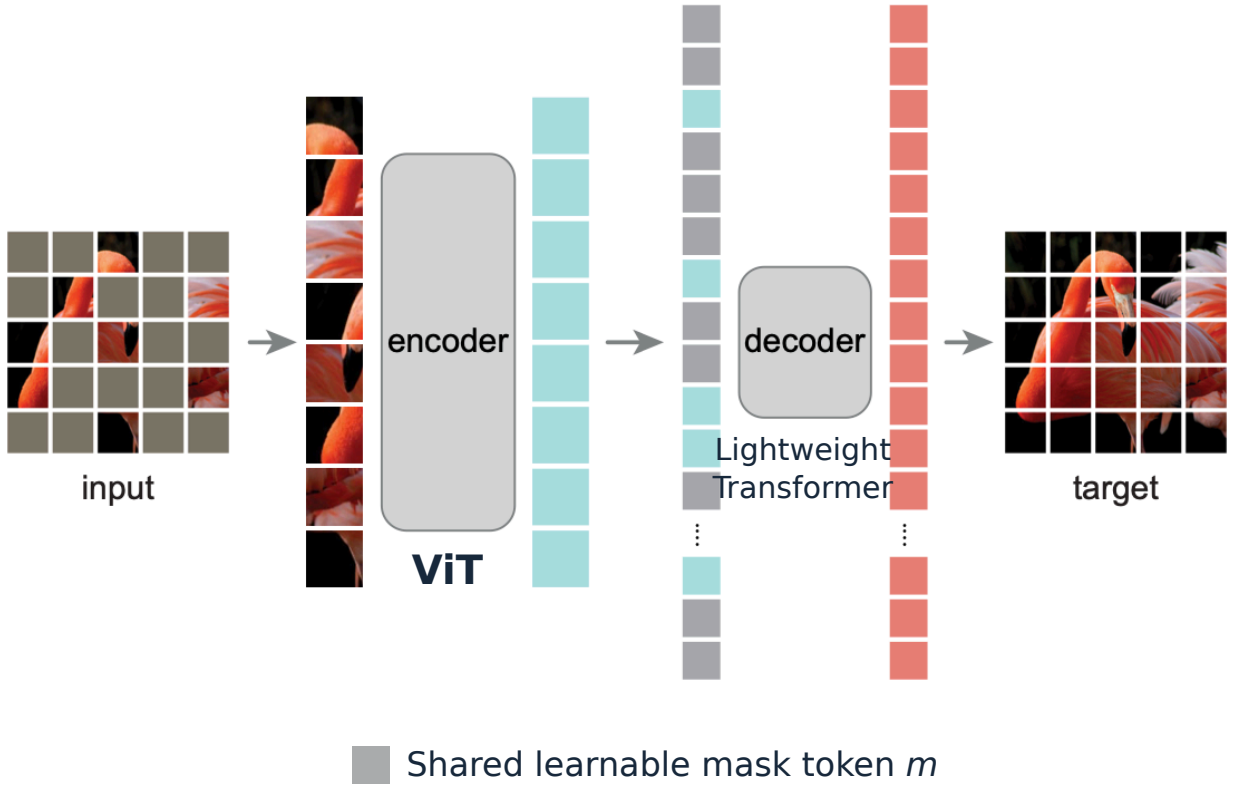


- Replace masked patch embeddings with the **same learnable token** m.
- Reconstruct RGB pixels with a **linear head**.

$$L = \frac{1}{\Omega(x_M)} \|y_M - x_M\|_1$$

x_M: original masked pixels.
y_M: predictions at masked positions.
$\Omega(x_M)$: number of masked pixels.

Masked Autoencoder (MAE)



Like classical autoencoders, MAE learns by **reconstructing the input**.

- Randomly **mask 75%** of patches.
- The ViT encoder processes **only visible patches**.
- Insert shared learnable tokens m **before the decoder**.

MAE: reconstruction and feature reuse

During pretraining

A **lightweight Transformer decoder** (shallower and narrower) reconstructs masked pixels.

For downstream tasks

Use features **before the decoder**. Keep the encoder and discard the reconstruction decoder.

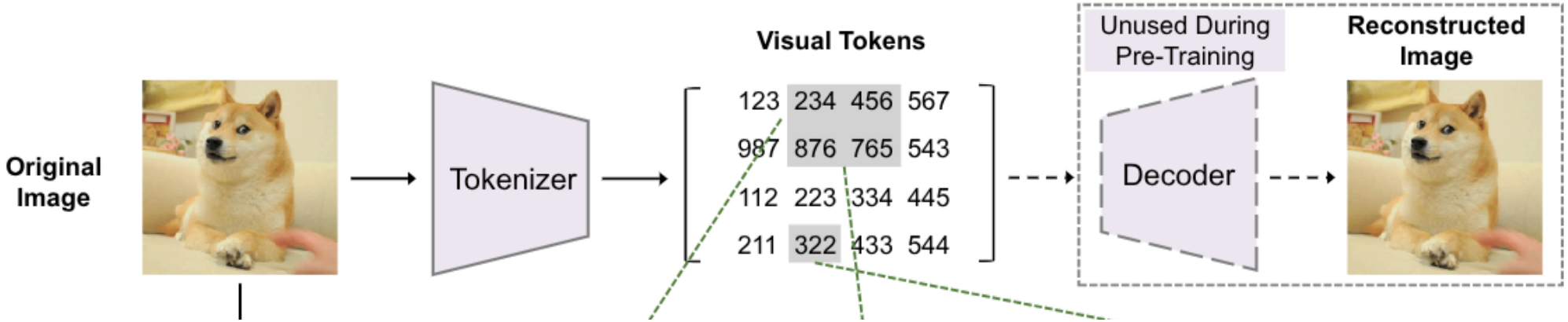
$$L_{\mathrm{MAE}} = \frac{1}{|M|} \sum_{i \in M} \mathrm{MSE}(\hat{x}_i, x_i)$$

M: masked patch positions.
$\hat{x}_i$ and x_i: predicted and original pixels of patch i.

Loss: mean squared error on the masked patches only.

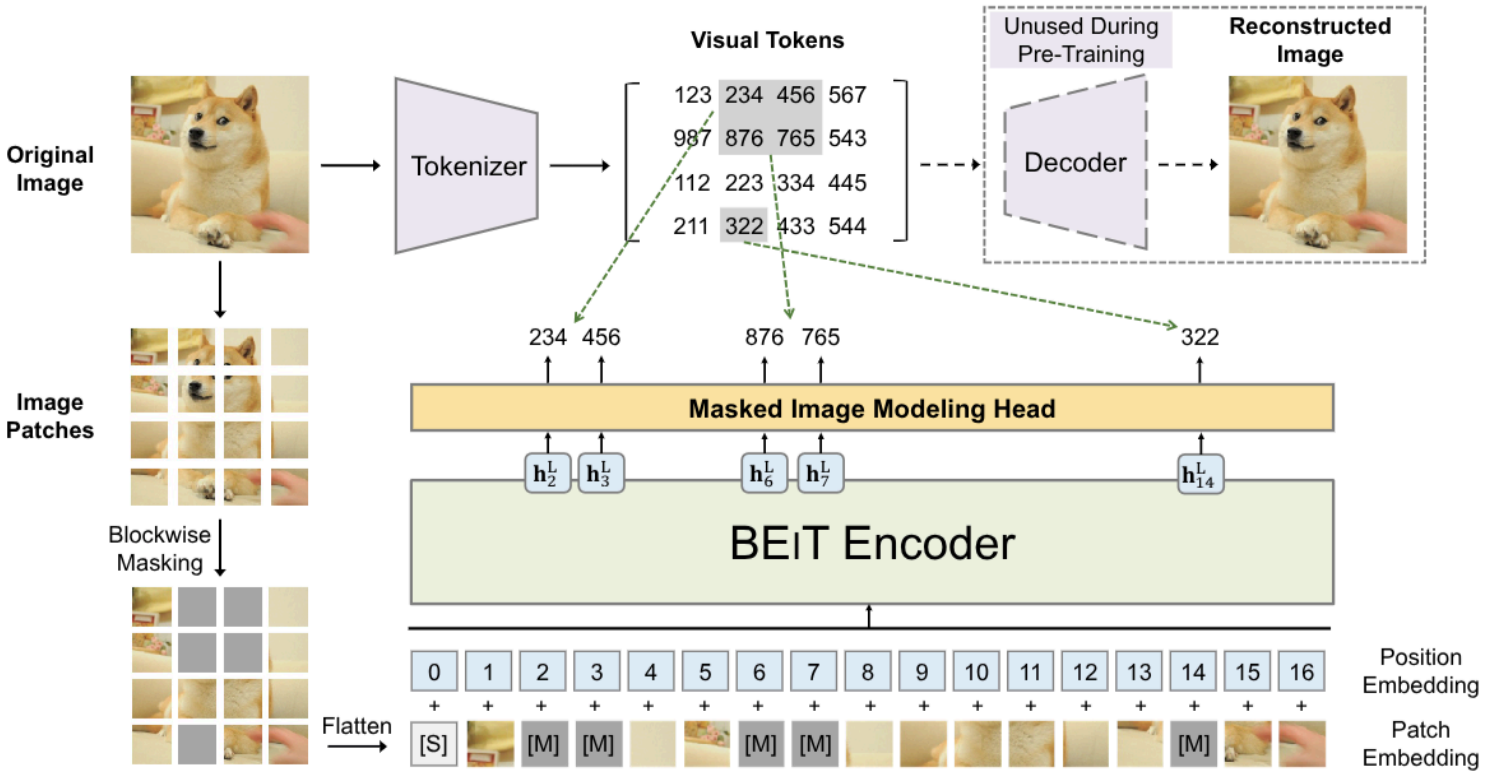
BEiT · Phase 1: learn a visual vocabulary

A **visual tokenizer** converts each image patch into a **discrete ID**.



- Train the **tokenizer and decoder** with variational autoencoder (with CNN).
- Each ID identifies a learned visual pattern.
- Then **freeze the tokenizer**. Its decoder is discarded.

BEiT · Phase 2: predict missing visual tokens



1. The **frozen tokenizer** reads the original image and generates target IDs.
2. Replace selected patch embeddings with a **learnable mask embedding**.
3. Train the **BEiT encoder and prediction head** to recover the IDs at masked positions.

Bao et al., *BEiT: BERT Pre-Training of Image Transformers*, ICLR, 2022.

BEiT predicts code IDs, not pixels

Prediction target	Discrete IDs generated by the frozen tokenizer

$$\mathcal{L}_{\mathrm{BEiT}} = \mathbb{E}_{M}\left[\sum_{i \in M} \log p_{\mathrm{MIM}}\left(z_i \mid x^{\mathcal{M}}\right)\right]$$

M: masked positions.

z_i: tokenizer's target ID at position i.

Cross-entropy classifies each masked position over the visual vocabulary.

> Phase 1 learns the visual vocabulary. Phase 2 learns to predict its codes from context.
>
> Reuse the **BEiT encoder** for downstream tasks.

Masked image modeling: recap

	SimMIM	MAE	BEiT
Encoder input	Full token grid	Visible tokens only	Full token grid
Mask tokens added	Before encoder	Before decoder	Before encoder
Prediction module	Linear head	Lightweight Transformer decoder	Classification head
Target / loss	Pixels / L_1	Pixels / MSE	Discrete code IDs / cross-entropy
Discrete tokenizer	Not needed	Not needed	Separately pretrained and frozen

> SimMIM and MAE predict **pixels**; BEiT predicts **discrete visual codes**.

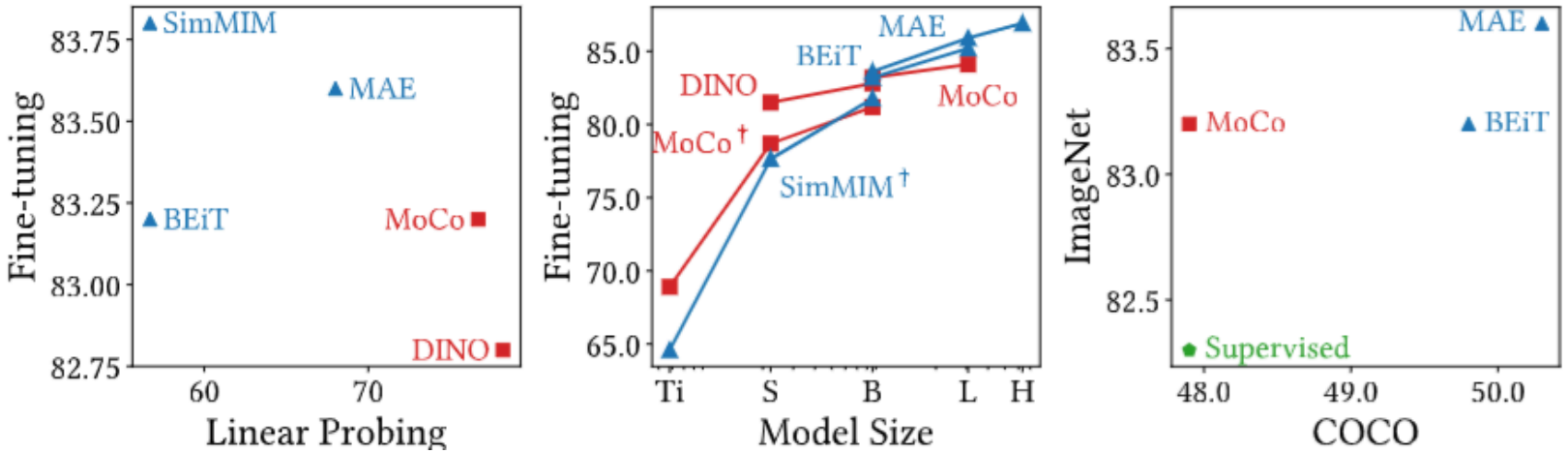
Instance Discrimination vs MIM

Design choice	Contrastive learning	Masked reconstruction
Training target	Match views of the same image against negatives	Predict missing pixels or visual codes
Views per image	Typically two or more augmented views	One masked view can suffice
Observed biases	More global structure and shape	More local detail and texture

Input pipeline cost: strong augmentations often run on the CPU. Generating multiple views can bottleneck data loading and leave the GPU waiting.

> Masked reconstruction instead does not require multiple augmented views.

Which representations transfer better?



- **Linear probing:** instance discrimination methods show stronger linear separability.
- **Fine-tuning and scaling:** masked models can outperform instance discrimination models, especially at larger sizes.
- **Dense task:** masked models do well on object detection, where spatial detail matters.

> The preferred representation depends on the downstream task.

Park et al., *What Do Self-Supervised Vision Transformers Learn?*, ICLR, 2023.

Part 2

Vision Foundation Models

scale and generalization matter

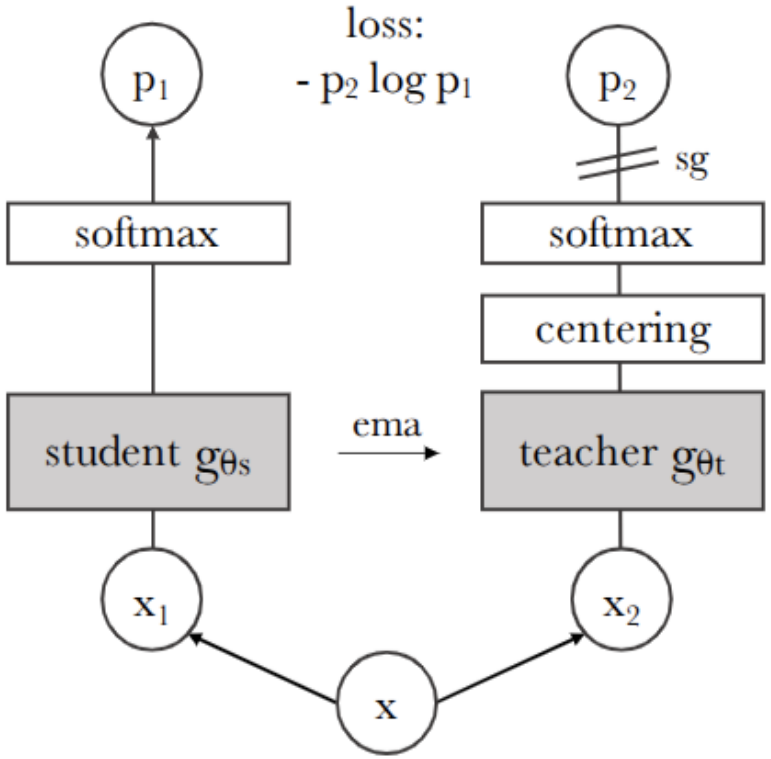
Vision foundation models

A large, broadly pretrained vision model that supports many downstream tasks.

- Scale and diversity of training data matter.
- Useful representations must transfer beyond one benchmark.
- A frozen backbone plus task heads is a practical deployment pattern.

> The objective is **broad reuse**: one visual backbone providing strong features across many **tasks and domains**

DINO: self-distillation on Vision Transformers



DINO matches teacher output distributions using a **ViT backbone and projection head**.

- **Student:** global and local crops; updated by gradients.
- **Teacher:** global crops; updated by EMA.

Instance discrimination objective: match distributions across different views of the same image.

DINO: centering and sharpening

$$p_t = \mathrm{softmax}\left(\frac{u_t - c}{\tau_t}\right)$$

u_t: teacher logits. c: running mean of teacher logits. τ_t: teacher temperature.

Operation	How it works	Failure it counters
Centering	Subtract the running mean of each teacher logit	The same output coordinate dominates for every image
Sharpening	Use a low temperature to produce peaked targets	Every image receives a flat, uniform distribution

> Centering balances output usage. Sharpening keeps individual targets informative. They work together with EMA and stop-gradient.

One DINO training step

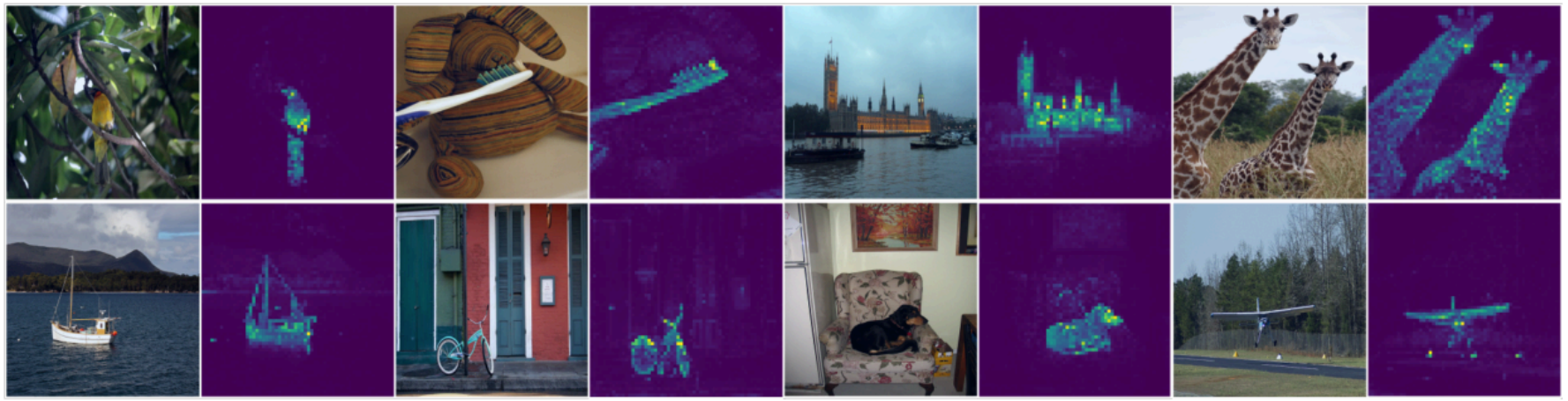
```
views = augment(images)                    # global + local crops
s_logits = student(views.all_crops)
with no_grad():
    t_logits = teacher(views.global_crops)
    p_t = softmax((t_logits - center) / tau_t, dim=-1)

log_p_s = log_softmax(s_logits / tau_s, dim=-1)
loss = cross_view_loss(p_t, log_p_s, skip_same_view=True)

optimizer.zero_grad()
loss.backward()
optimizer.step()                           # student only

ema_update(teacher, student, momentum)
center = rho * center + (1 - rho) * t_logits.mean(dim=0)
```

DINO: emergent visual structure



Images and last-layer CLS attention maps.

- Object structure: some attention heads highlight object regions without training on segmentation masks.
- Semantic similarity: image representations support strong k-NN classification using labeled reference images.

Caron et al., *Emerging Properties in Self-Supervised Vision Transformers*, ICCV, 2021.

DINOv2: curated data at scale

LVD-142M: 142 million images

Large web pool → remove duplicate → Visual retrieval → Curated dataset

> Self-supervision removes the need for training labels, but **which images we train on** still matters.

DINOv2: global + local objective

DINO teacher loss

Match student and teacher **image-level outputs** across different crops.

Encourages a useful **global representation**.

Calculated on CLS embedding token

Masked reconstruction loss

Mask student patches and predict the teacher's outputs at those positions.

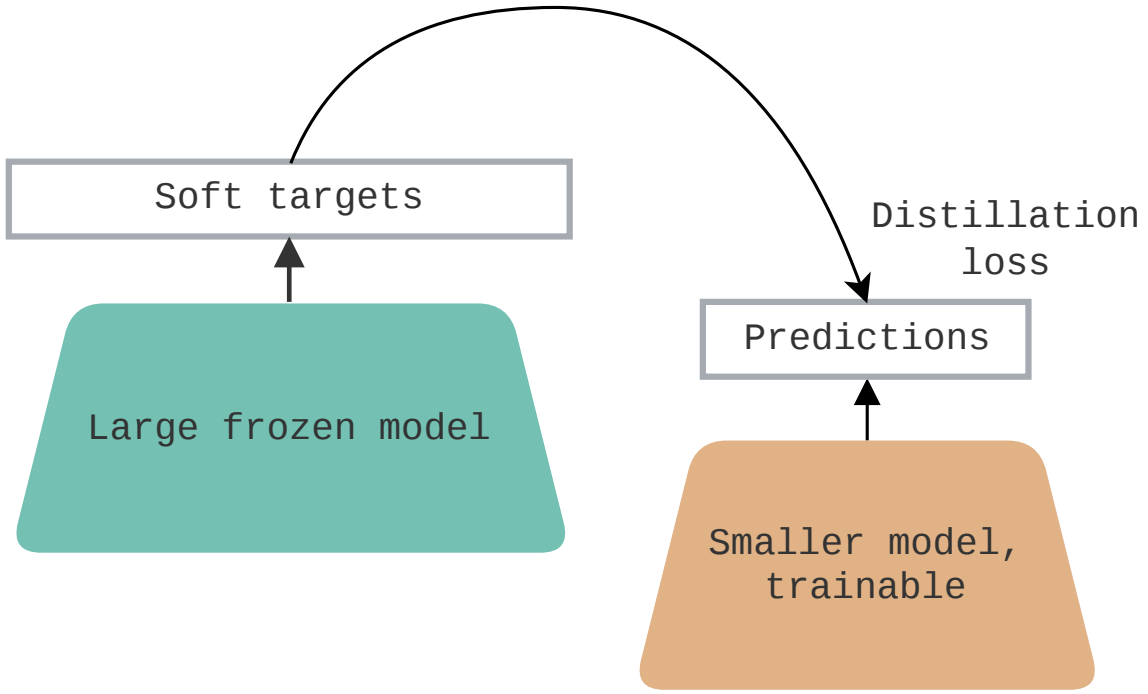
Encourages useful **local patch features**.

Calculated on patch embedding tokens

$$L_{\text{main}} = L_{\text{DINO}} + L_{\text{masked}}$$

> Combine global and local learning. Masked targets are **teacher outputs**.

DINOv2: distilling a large model



1. Train a **large SSL model**.
2. Freeze it as the **teacher**.
3. Train a **smaller student** to reproduce its outputs on unlabeled images.

Teacher outputs provide the supervision. Only the student receives gradient updates.

> Transfer knowledge into a cheaper backbone. Here the teacher is **fixed**, unlike an EMA teacher.

DINOv2: a family of reusable backbones

Backbone	Parameters	How it is obtained
ViT-S/14	21 million	Distilled from ViT-g
ViT-B/14	86 million	Distilled from ViT-g
ViT-L/14	300 million	Distilled from ViT-g
ViT-g/14	**1.1 billion**	Large-scale SSL training

> Train at large scale, then distill smaller models, so to have a backbone that fits the application’s requirement budget.

Application of DINOv2: Depth Anything

Estimate a **dense depth map from a single RGB image**.



DINOv2 encoder → Depth prediction decoder head → Depth map

Adapt the encoder and decoder using **depth supervision and pseudo-labels**.

> DINOv2 supplies the visual features. Task-specific training turns them into depth predictions.

Yang et al., *Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data*, CVPR, 2024.

DINOv3 scales both data and capacity

7B params

Parameters in the large ViT teacher.

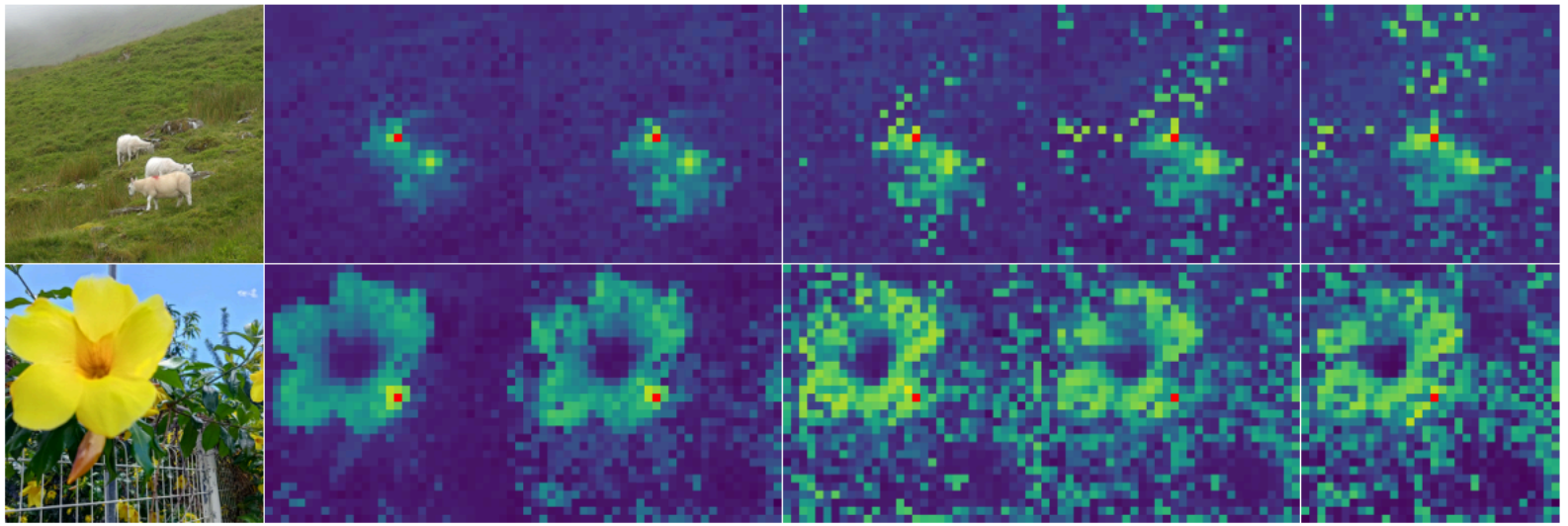
1.689B images

Images in the curated web training set.

Again a family of distilled models supports smaller inference budgets.

> At this scale, preserving dense feature quality becomes a distinct training problem.

Dense feature degradation



Cosine similarity to the red reference patch across training. Fig. 6, cropped.

> **Problem:** dense features degrade, even though the loss decreases, because global (CLS token) and local objective (patch tokens) compete

Gram anchoring for dense features

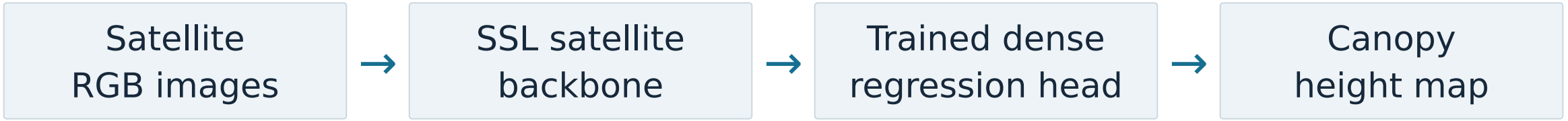
Keep **which patches look alike** fixed, even as the features themselves keep changing.

$$G = XX^\top \quad \left(G_{ij} = \text{similarity of patch } \ i \ \text{ and patch } \ j\right)$$

$$L_{\text{Gram}} = \sum_{i,j} \left(G_{ij}^{\text{student}} - G_{ij}^{\text{teacher}}\right)^2$$

- **What:** match the student's patch-similarity map to an early, good one.
- **Why:** keep.
- **Teacher:** an earlier checkpoint with sharp dense features, refreshed periodically (not the per-step EMA teacher).

DINOv3 in earth observation



- Domain pretraining: SAT-493M, **493 million images**.
- Head supervision: height targets from LiDAR data.
- Domain transfer depends on sensors, geography and task.

> SSL provides reusable spatial features. The supervised head assigns physical meaning and units.

JEPA: why predict representations?

Pixel reconstruction

Predict what a missing region **looks like**.

Recover colors, textures and fine details.

Representation prediction

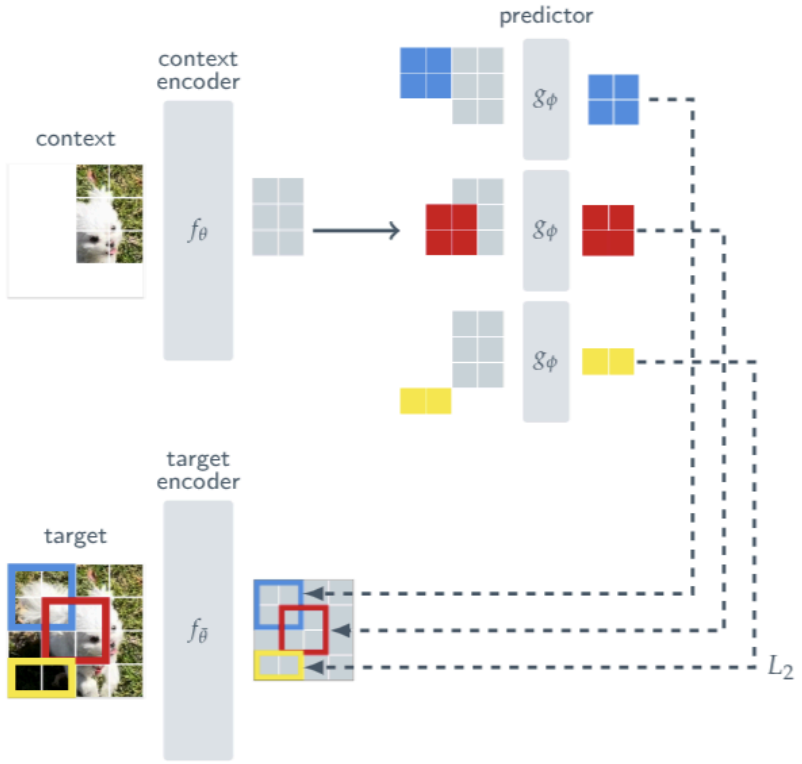
Predict the **features** of a missing region.

Match a learned description of its content.

- Exact pixel details may be ambiguous or irrelevant to downstream tasks.
- Predicting in feature space offers a way to learn visual structure without generating an image.

> Learn an encoder through **prediction in representation space**, then reuse its features for downstream tasks.

I-JEPA: predict missing image features



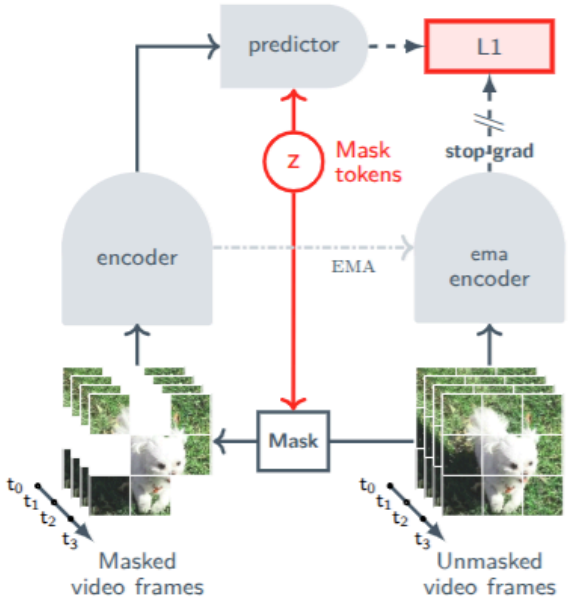
1. Encode the **visible context** with a ViT.
2. Predict features of **hidden target blocks**, using their positions.
3. Match features from an **EMA target encoder** that sees the full image.

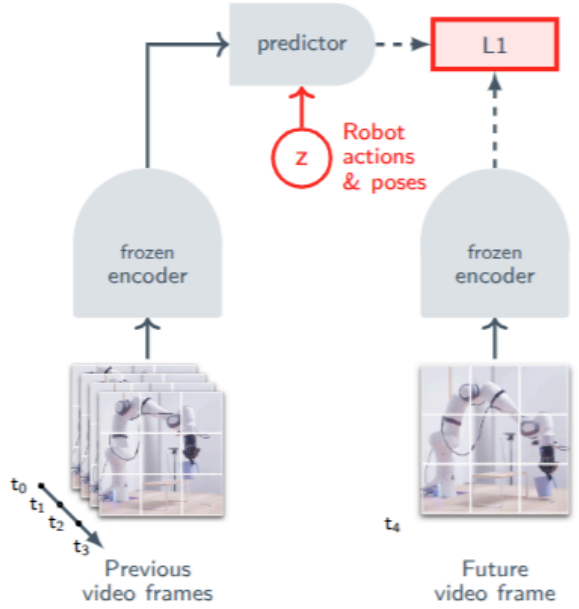
$$L \propto \sum_{i \in M} \left\| \hat{z}_i - \mathrm{sg}\left(z_i^t\right) \right\|^2$$

M: target positions.

The target is a **feature vector**, not RGB pixels. For downstream tasks, retain the encoder and train an appropriate head.

V-JEPA: learning from video





Video pretraining

Predict features of regions hidden **across space and time**.

Action-conditioned extension

Freeze the encoder and train a new head to predict future features from past observations and **robot actions**.

V-JEPA: Bardes et al., 2024 · V-JEPA 2: Assran et al., 2025, Fig. 2

Further reading

DINOv3: Self-supervised learning for vision at unprecedented scale Meta AI, 2025 · Blog post

An overview of DINOv3, scaling self-supervised learning, and applications of pretrained visual features.

A Cookbook of Self-Supervised Learning Balestriero et al., 2023 · Survey and practical guide

A reference on SSL method families, training recipes, and representation evaluation.