Week 7

Self-Supervised Learning

[ECEA0649/ECE40049] Deep Learning for Image Processing | Spring 2026

Kihyun Na (Research Professor)
BK21 AI Education and Research Group &
Institute for Information and Communication Technology,
Handong Global University

HANDONG GLOBAL UNIVERSITY

Curriculum

** Adjusted based on survey results*

Wk 1	OT + Introduction	**Wk 9**	Foundation Models (CLIP, SAM) + Paper #1
Wk 2	DL Fundamentals Review	**Wk 10**	Diffusion Models + Paper #2
Wk 3	CNN	**Wk 11**	Conditional Generation + Paper #3
Wk 4	From Sequence Modeling to Transformer	**Wk 12**	Vision-Language Models + Paper #4
Wk 5	Transformer in Vision	**Wk 13**	VLM Applications + Paper #5
Wk 6	Detection & Segmentation	**Wk 14**	Video Understanding + Paper #6
Wk 7	Self-Supervised Learning (Today)	**Wk 15**	Embodied AI & Robot Vision + Paper #7
Wk 8	Review Literacy + Role Explanation	**Wk 16**	Miniconference (Final Project)

Lecture Lecture + Paper Miniconference

Today's Agenda

01 **Why Self-Supervised Learning?** — 10 min

Motivation and the core idea of label-free representation learning

02 **Early Self-Supervision: Pretext Tasks and Reconstruction** — 25 min

From Rotation/Jigsaw/Inpainting/Colorization to Masked Autoencoders

03 **Contrastive and Non-Contrastive SSL** — 25 min

Contrastive: SimCLR → MoCo / Non-contrastive: BYOL, SimSiam / Collapse problem

04 **Self-Distillation in Vision Transformers** — 20 min

DINO → DINOv2

05 **Wrap-Up** — 10 min

Key takeaways, comparison summary, and Q&A

Part 1

Why Self-Supervised Learning?

Motivation and the core idea of label-free representation learning

The Annotation Bottleneck

Labeled data is expensive

* ImageNet (Deng et al. 2009):
  48,940 workers from 167 countries
  2.5 years of labeling (2008–2010)
  14 million images × 3 annotations each

* COCO: 328K images × 5 captions
  + bounding boxes + segmentation masks
* Medical imaging: $10–100 per image
(expert annotation required)

Unlabeled data is abundant

Instagram: tens of millions of photos/videos per day

YouTube: 500+ hours of video uploaded every minute (2023)

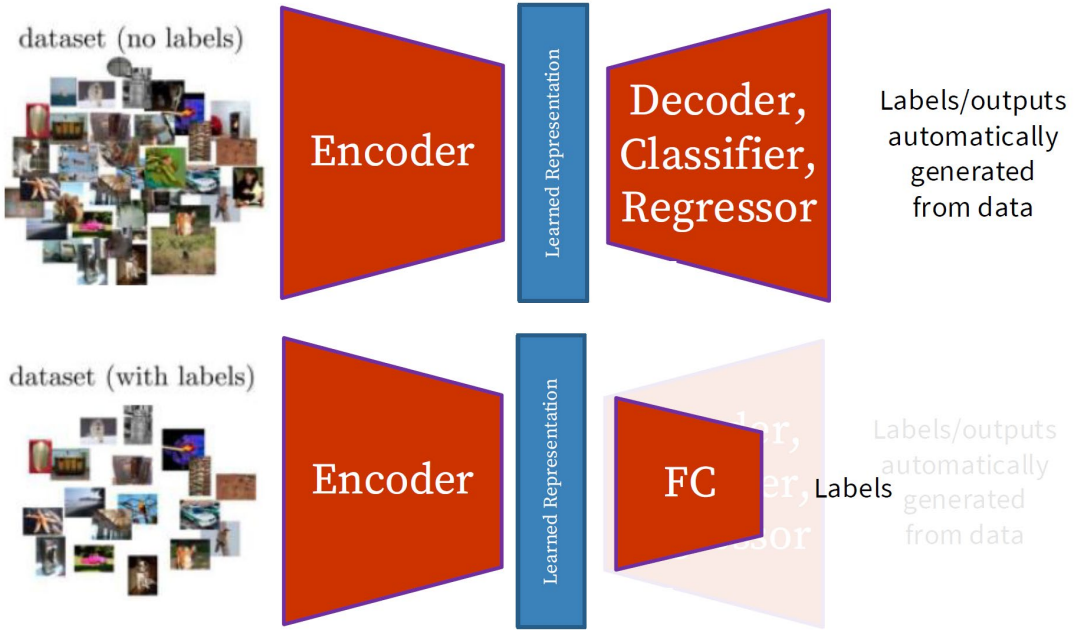
Web-scale crawls (e.g., Common Crawl / LAION): billions of image-text pairs

Can we learn useful visual representations from this ocean of unlabeled or weakly labeled web data

without costly human annotation?

→ Learn representations from unlabeled or naturally paired web data

Self-Supervised Learning Framework



Pretext Task
- Define a task based on the data itself
- No manual annotation
- Could be considered an **unsupervised** task;
- but we learn with supervised learning objectives, e.g., classification or regression.

Downstream Task
- The application you care about
- You do not have large datasets
- The dataset is labeled

Key Takeaway Learn representations from unlabeled data via pretext tasks, then transfer to downstream tasks with minimal labels.

Self-supervised pretext tasks

Example: learn to predict image transformations / complete corrupted images

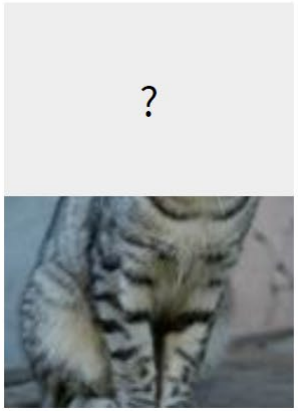
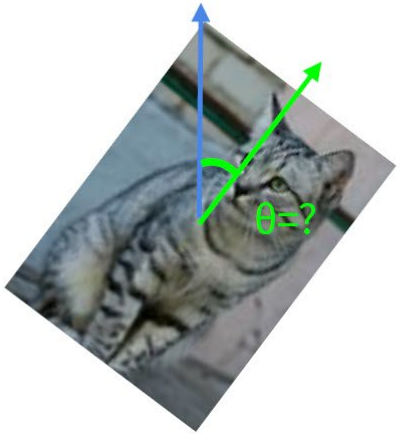


image completion



rotation prediction

“jigsaw puzzle”

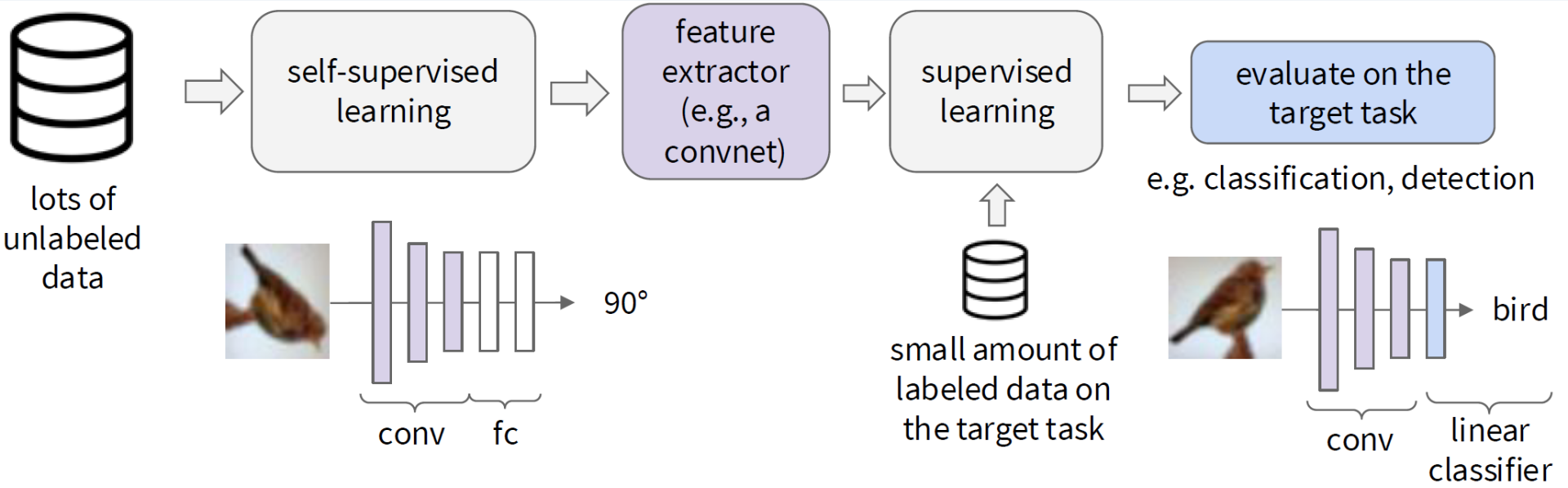
colorization

1. Solving the pretext tasks allow the model to learn good features.
2. We can automatically generate labels for the pretext tasks.

How to evaluate SSL learning method?

- **Pretext Task Performance**
  - Measure how well the model performs on the task it was trained on without labels.
- **Representation Quality**
  - Evaluate the quality of the learned representations
    - *Linear Evaluation Protocol:* Train a linear classifier on the leaerned representations;
    - *Clustering:* Measure clustering performance;
    - *t-SNE:* Visualize the representations to assess their separability.)
- **Robustness and Generalization**
  - Test how well the model generalizes to different datasets and is robust to variations.
- **Computational Efficiency**
  - Assess the efficiency of the method in terms of training time and resource requirements.
- **Transfer Learning and Downstream Task Performance**
  - Assess the utility of the learned representations by transferring them to a downstream supervised task.

How to evaluate SSL learning method?



1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

2. Attach a shallow network on the feature extractor; train the shallow network on the target task with small amount of labeled data

How to evaluate SSL learning method?

The ruler for every number in this lecture

Linear probe

Freeze the pretrained encoder
Train only a linear layer on top

Measures:
"How good are the frozen features by themselves?"

Example:
DINOv2 ViT-g → 86.5%
(just one linear layer!)

A standard evaluation protocol in SSL
Higher = better representation.

kNN classifier

No training at all

For each test image, find
K nearest neighbors in the training set (in feature space)
→ Vote on the label

Measures:
“How well-structured is the feature space?”

Example:
DINO ViT-S/8 → 78.3% kNN
(zero training, zero parameters)

Fine-tuning

Unfreeze the entire model
Train all parameters on downstream task + labels

Measures:
"How good is this as an initialization?"

Example:
MAE ViT-L → 84.9% fine-tune
(but only 75.3% linear probe
→ MAE needs fine-tuning)

*Note: Example numbers depend on backbone, dataset, and training setup

Part 2

Early Self-Supervision: Pretext Tasks and Reconstruction

From Rotation/Jigsaw/Inpainting/Colorization to Masked Autoencoders

Early Pretext Tasks (2016–2018)

Learn features by solving proxy tasks — no labels needed

Rotation (2018)

Predict rotation angle applied to image (0°, 90°, 180°, 270°)

4-way classification → Requires understanding object orientation

Gidaris et al., ICLR 2018

Jigsaw (2016)

Predict permutation of image patches

Solving the puzzle requires understanding spatial structure

Noroozi & Favaro
ECCV 2016

Colorization (2016)

Predict color channels from grayscale input

Requires understanding object semantics (sky=blue, grass=green)

Zhang et al.
ECCV 2016

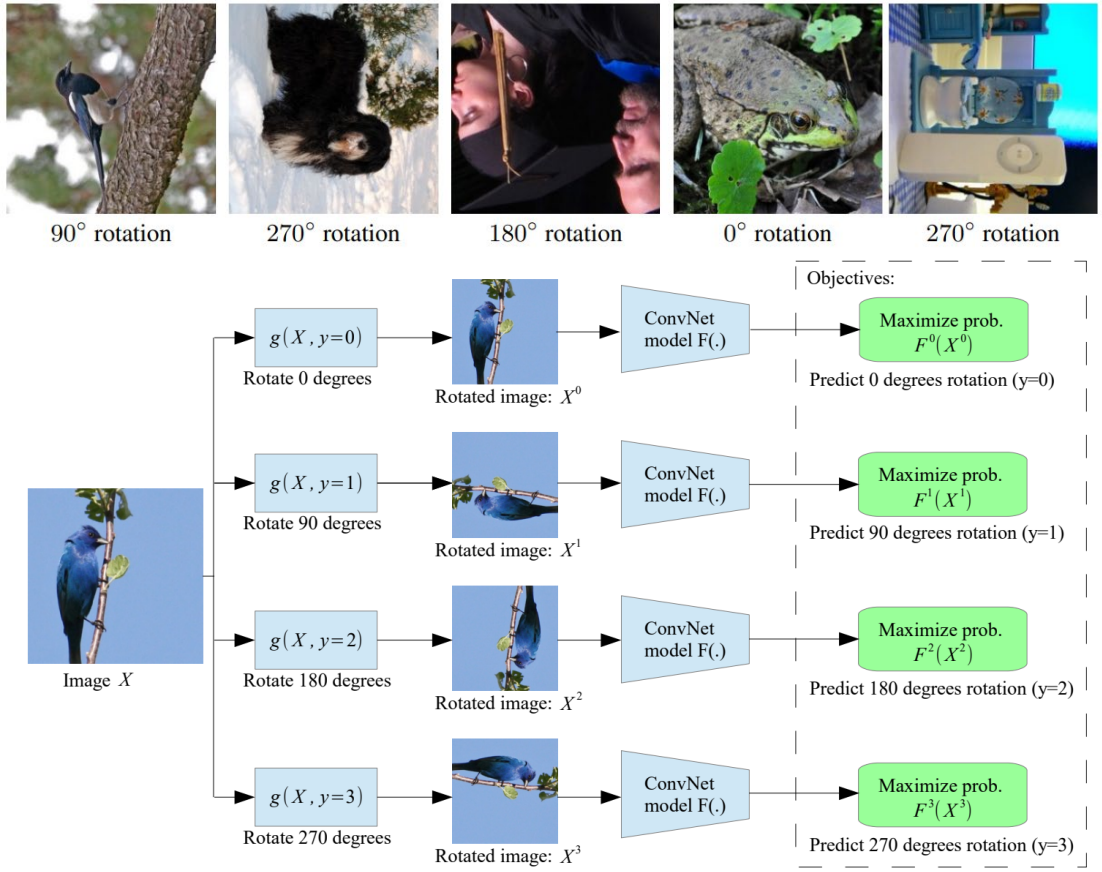
CPC (2018)

Contrastive Predictive Coding

Proposed InfoNCE loss → Foundation for SimCLR's NT-Xent

van den Oord et al.
arXiv 2018

Pretext task: predict rotations



Predicting Image Rotations

Hypothesis: a model could recognize the correct rotation of an object only if it has the “visual commonsense” of what the object should look like unperturbed.

Self-supervised learning by rotating the entire input images.

The model learns to predict which rotation is applied. (4-way cls.)

Pretext task: predict rotations

	Classification (%mAP)		Detection (%mAP)	Segmentation (%mIoU)
Trained layers	fc6-8	all	all	all
ImageNet labels	78.9	79.9	56.8	48.0
Random		53.3	43.4	19.8
Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6	32.6
Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9	
Context Encoders (Pathak et al., 2016b)	34.6	56.5	44.5	29.7
Tracking (Wang & Gupta, 2015)	55.6	63.1	47.4	
Context (Doersch et al., 2015)	55.1	65.3	51.1	
Colorization (Zhang et al., 2016a)	61.5	65.6	46.9	35.6
BIGAN (Donahue et al., 2016)	52.3	60.1	46.9	34.9
Jigsaw Puzzles (Noroozi & Favaro, 2016)	-	67.6	53.2	37.6
NAT (Bojanowski & Joulin, 2017)	56.7	65.3	49.4	
Split-Brain (Zhang et al., 2016b)	63.0	67.1	46.7	36.0
ColorProxy (Larsson et al., 2017)		65.9		38.4
Counting (Noroozi et al., 2017)	-	67.7	51.4	36.6
(Ours) RotNet	**70.87**	**72.97**	**54.4**	**39.1**

Pretrained with full ImageNet supervision

No pretraining

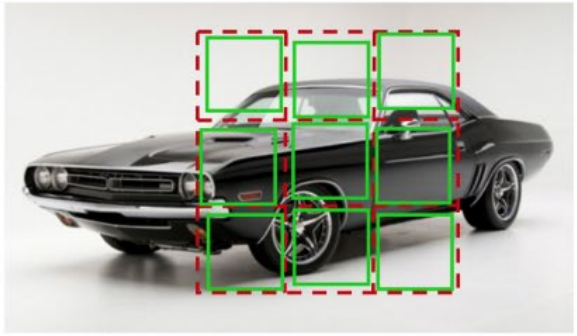
Self-supervised learning on ImageNet (entire training set) with AlexNet.

Finetune on labeled data from Pascal VOC 2007.

Self-supervised learning with rotation prediction

source: Gidaris et al. 2018

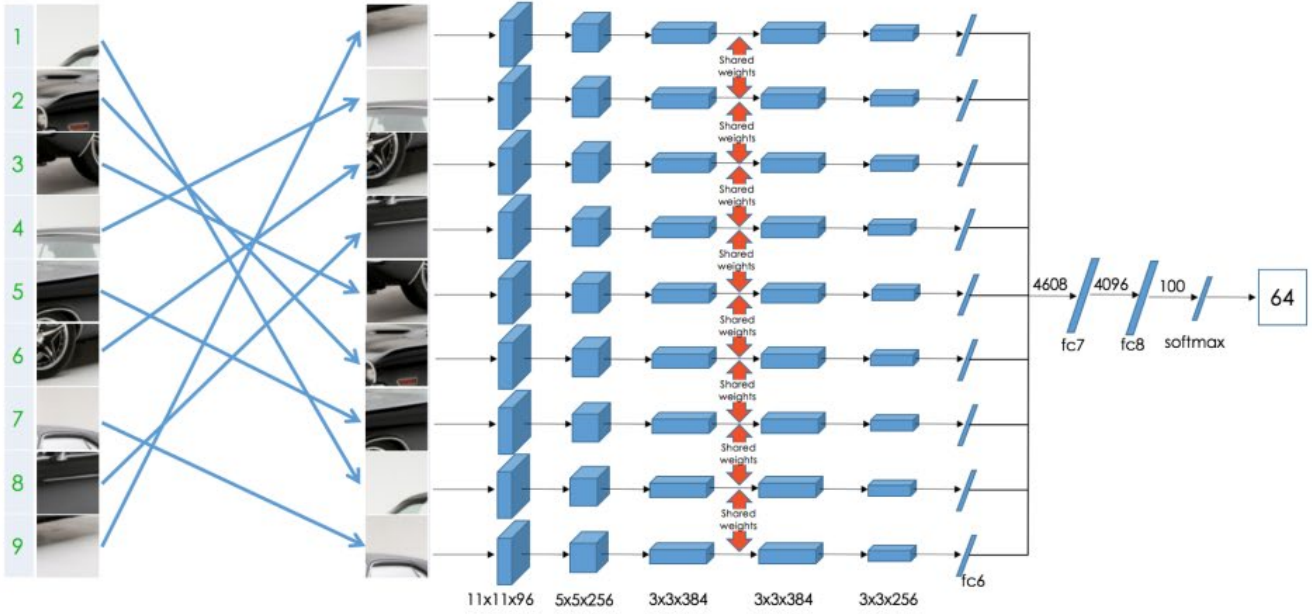
Pretext task: jigsaw puzzles



Permutation Set

index	permutation
64	9,4,6,8,3,2,5,1,7

Reorder patches according to the selected permutation



The model does not predict each patch location independently. Instead, it predicts which predefined permutation was applied to the whole set of patches.

Pretext task: jigsaw puzzles

Table 1: Results on PASCAL VOC 2007 Detection and Classification. The results of the other methods are taken from Pathak *et al.* [30].

Method	Pretraining time	Supervision	Classification	Detection	Segmentation
Krizhevsky*et al.* [25]	3 days	1000 class labels	**78.2%**	**56.8%**	**48.0%**
Wang and Gupta[39]	1 week	motion	58.4%	44.0%	-
Doersch *et al.* [10]	4 weeks	context	55.3%	46.6%	-
Pathak *et al.* [30]	14 hours	context	56.5%	44.5%	29.7%
Ours	2.5 days	context	**67.6%**	**53.2%**	**37.6%**

With only 2.5 days of Context-based SSL training, it closes the gap to within 10.6%p of supervised ImageNet (78.2%)

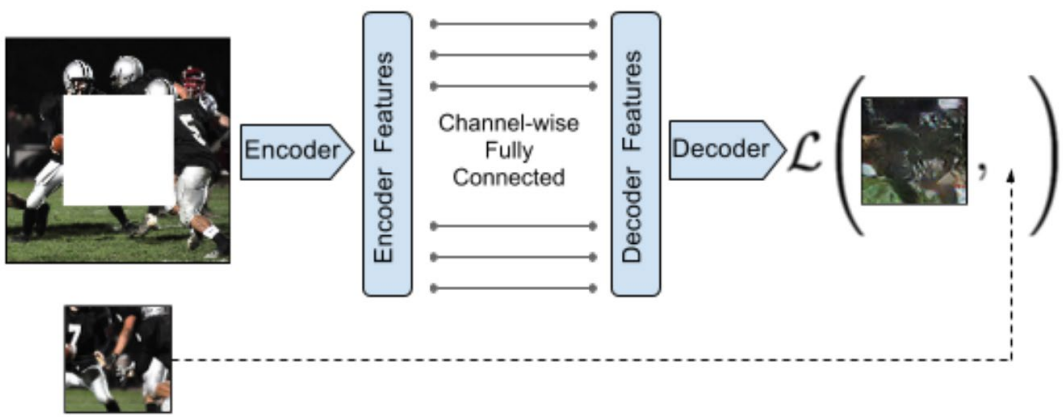
Pretext task: predict missing pixels (inpainting)

(a) Input context

(b) Human artist

(c) Context Encoder ($L2$ loss)

(d) Context Encoder ($L2$ + Adversarial loss)



Loss = reconstruction + adversarial learning

Element wise multiplication

$$L(x) = L_{recon}(x) + L_{adv}(x)$$

$$M = \begin{cases} 0 & \text{not masked} \\ 1 & \text{masked} \end{cases}$$

$$L_{recon}(x) = ||M * (x - F_\theta((1-M) * x))||_2^2$$

Encoder

$$L_{adv} = \max_D \mathbb{E}[\log(D(x))] + \log(1 - D(F((1-M) * x)))]$$

Adversarial loss between “real” images and inpainted images

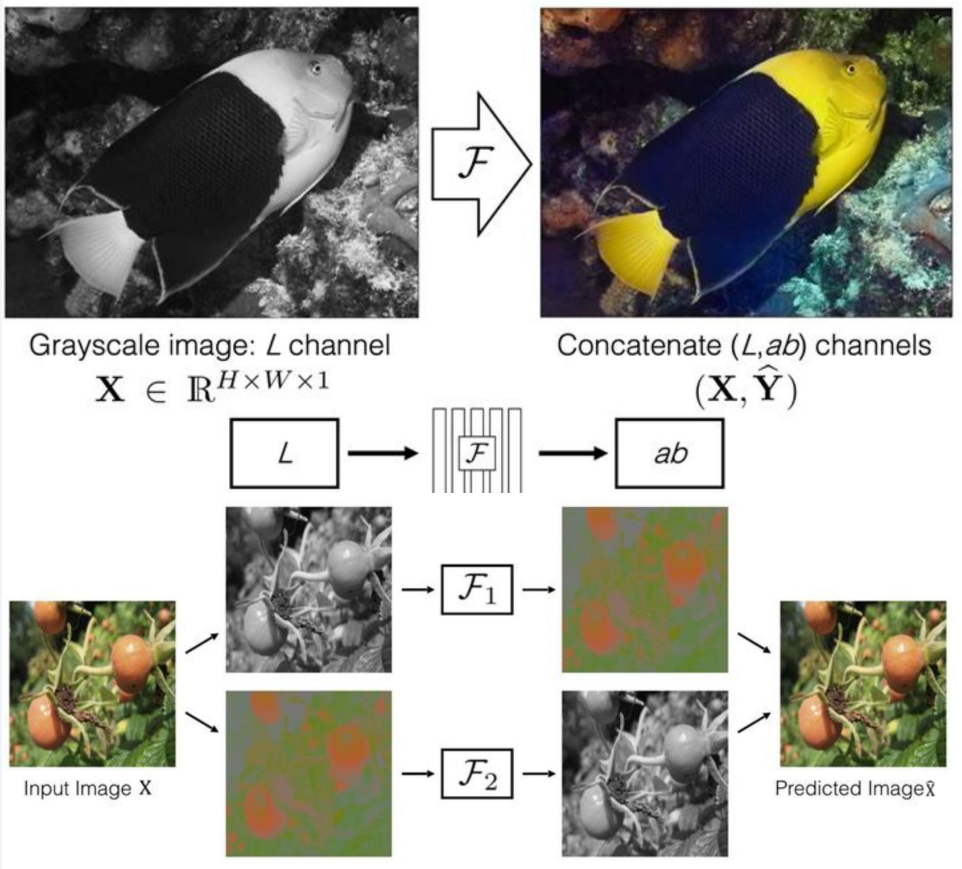
Pretext task: predict missing pixels (inpainting)

Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	78.2%	56.8%	48.0%
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal *et al.* [1]	egomotion	10 hours	52.9%	41.8%	-
Wang *et al.* [39]	motion	1 week	58.7%	47.4%	-
Doersch *et al.* [7]	relative context	4 weeks	55.3%	46.6%	-
Ours	context	14 hours	56.5%	44.5%	30.0%

Table 2: Quantitative comparison for classification, detection and semantic segmentation. Classification and Fast-RCNN Detection results are on the PASCAL VOC 2007 test set. Semantic segmentation results are on the PASCAL VOC 2012 validation set from the FCN evaluation described in Section 5.2.3, using the additional training data from [18], and removing overlapping images from the validation set [28].

Self-supervised learning on ImageNet training set, transfer to classification (Pascal VOC 2007), detection (Pascal VOC 2007), and semantic segmentation (Pascal VOC 2012)

Pretext task: colorization



Colorization as a pretext task

- Feed only the luminance channel (*L*)
- Predict the missing chrominance channels (*ab*)
- No manual labels: the original color image provides the target
- Encourages semantic understanding, since plausible colors depend on object identity and scene context

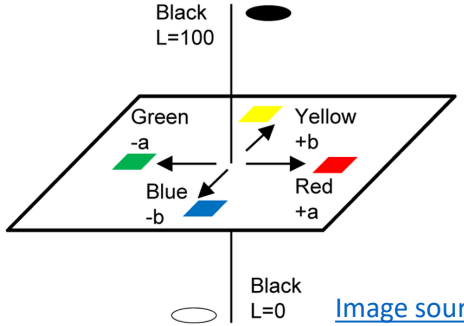
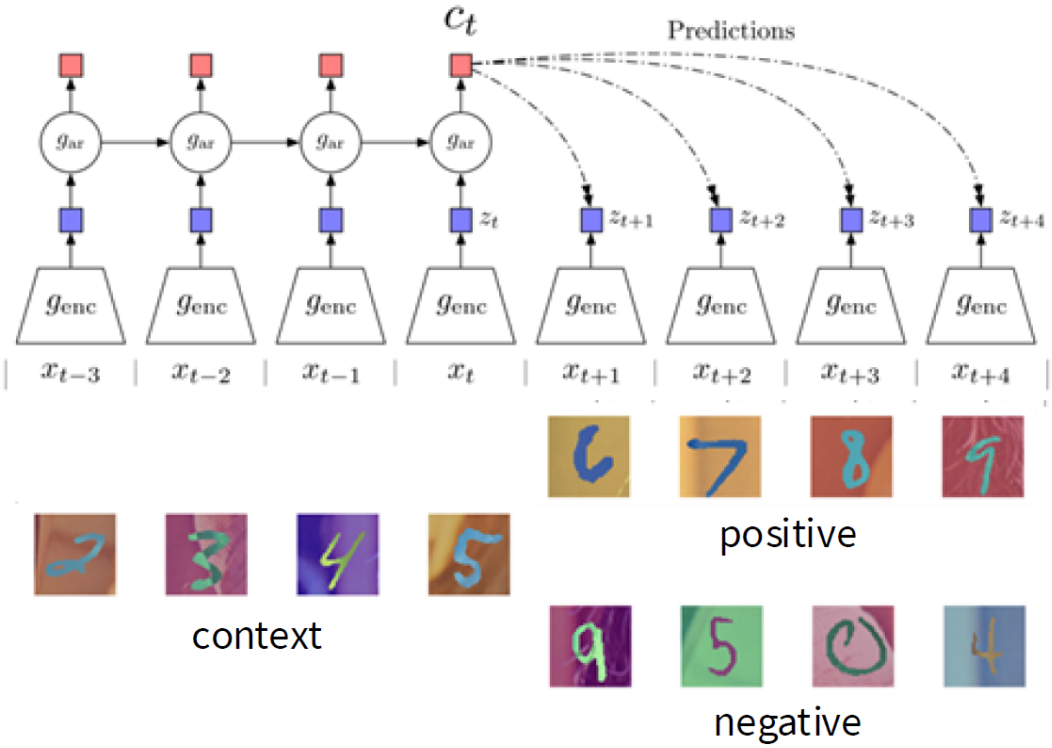


Image source: Liu et al., 2014

Pretext task: Contrastive Predictive Coding

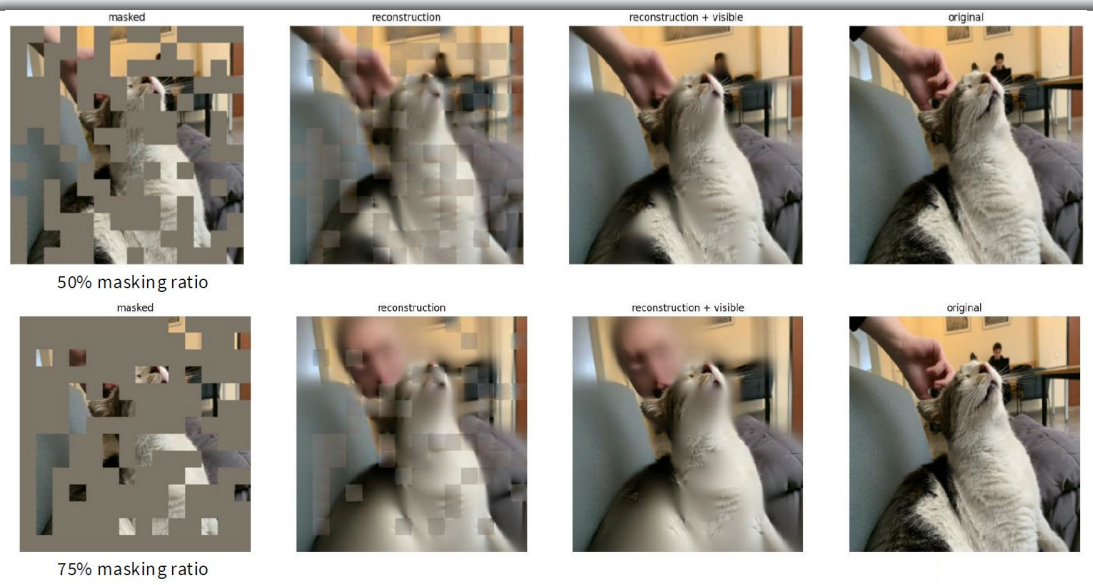


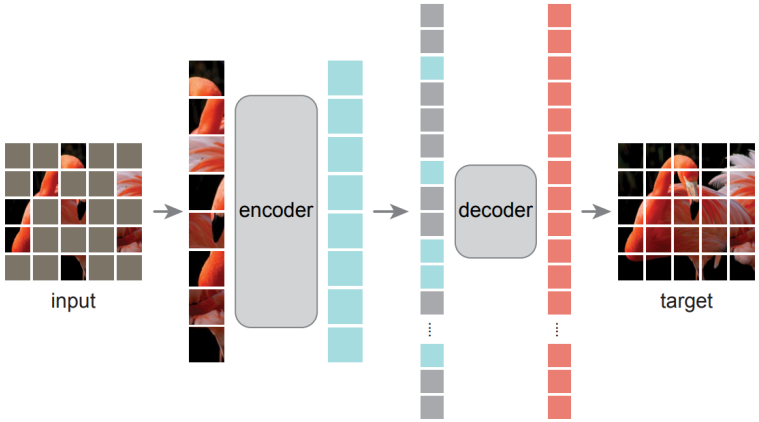
Contrastive: contrast between “right” and “wrong” sequences using contrastive learning.

Predictive: the model has to predict future patterns given the current context.

Coding: the model learns useful feature vectors, or “code”, for downstream tasks, similar to other self-supervised methods.

MAE: Masked Autoencoders





Key Takeaway MAE masks 75% of patches and feeds only visible patches to the encoder, making training both harder and faster.

MAE: Why 75% Masking?

He et al., CVPR 2022

The math

224×224 image
÷ 16×16 patches
= 196 patches total

75% masked
→ 49 visible patches
→ Encoder processes 49, not 196

→ ~4× less computation
→ roughly 3× faster

Why it works

High masking creates a hard task:

With 25% visible, the model cannot simply "fill in" from neighboring patches

→ Must learn semantic understanding to reconstruct missing regions

Low masking (e.g. 25%)
→ Too easy, poor features

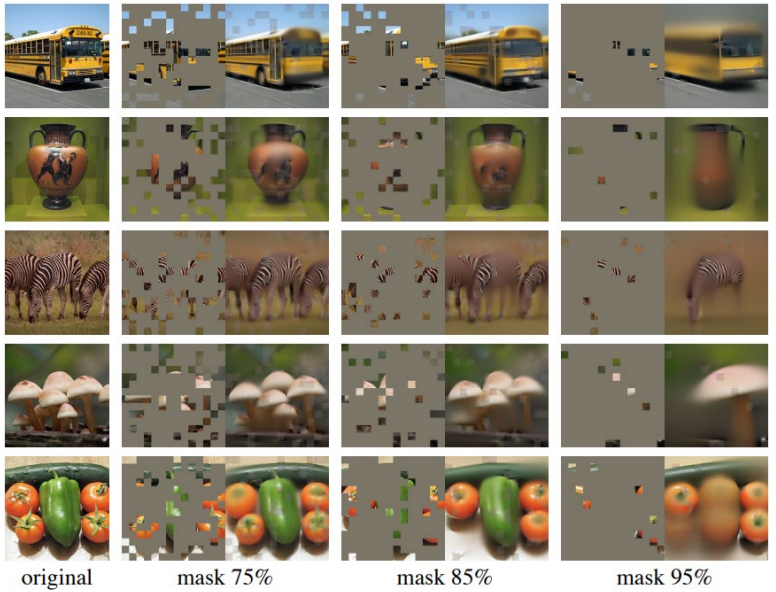


Figure 4. Reconstructions of ImageNet *validation* images using an MAE pre-trained with a masking ratio of 75% but applied on inputs with higher masking ratios. The predictions differ plausibly from the original images, showing that the method can generalize.

MAE: Ablation Studies

So many modeling/hyperparameter choices:

- Masking ratio
- Decoder depth
- Decoder width
- Mask token (used or not in encoder)
- Reconstruction target
- Data augmentation
- Mask sampling method
- Training schedule

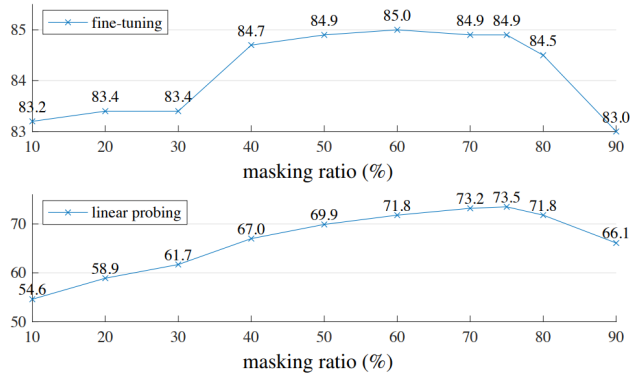
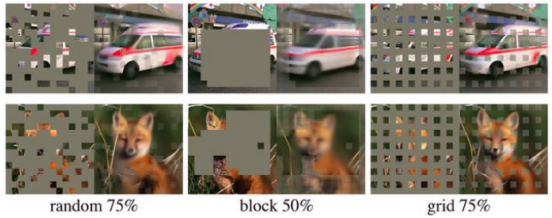


Figure 5. **Masking ratio**. A high masking ratio (75%) works well for both fine-tuning (top) and linear probing (bottom). The y-axes are ImageNet-1K validation accuracy (%) in all plots in this paper.

case	ratio	ft	lin
random	75	**84.9**	**73.5**
block	50	83.9	72.3
block	75	82.8	63.9
grid	75	84.0	66.0



Masked Image Modeling: MAE vs BEiT/iBOT

MAE / SimMIM — Pixel reconstruction

Reconstruct raw pixel values

SimMIM (Xie et al., CVPR 2022):
- Processes ALL patches
- (mask tokens in encoder)

MAE (He et al., CVPR 2022):
- Processes ONLY visible patches
- → Encoder sees 25%, not 100%
- → 3× faster training

★ MAE's key insight:
"skip masked patches entirely"

BEiT — Token prediction → iBOT

Predict discrete visual tokens
from a pre-trained dVAE

BEiT (Bao et al., ICLR 2022):
- 2-stage: train dVAE first
- More complex, similar results

iBOT (Zhou et al., ICLR 2022):
- Teacher = online tokenizer
- → No separate dVAE needed
- → Single-stage training
- → Used in DINOv2 (Part 4)

Pretext tasks from image transformations

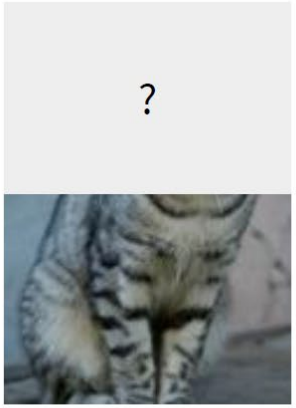
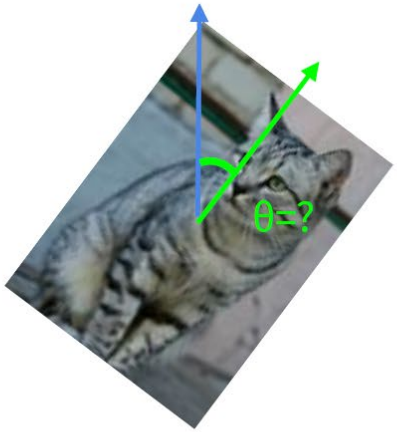


image completion



rotation prediction

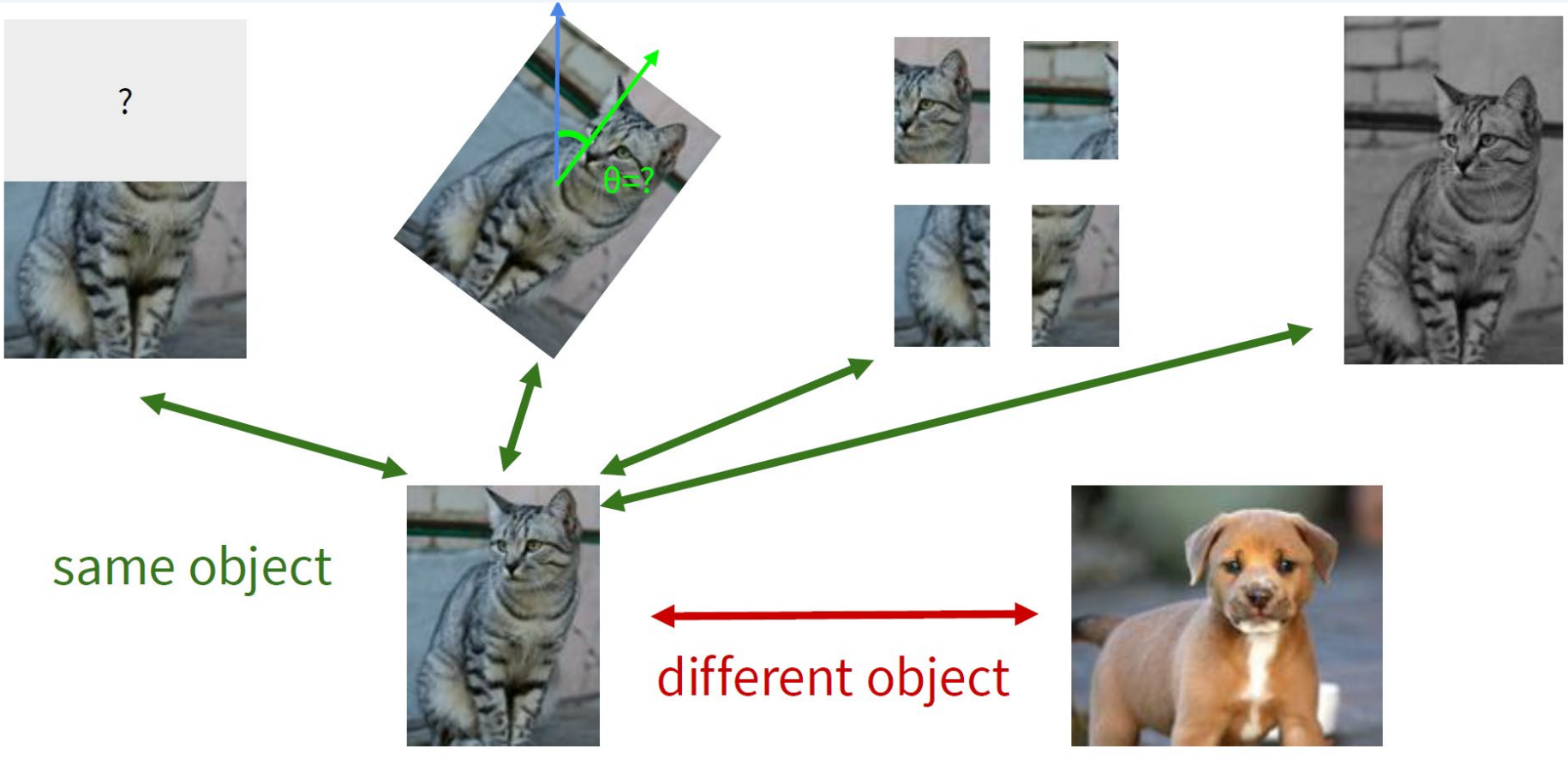
“jigsaw puzzle”

colorization

Learned representations may be tied to a specific pretext task!

Can we come up with a more general pretext task?

A more general pretext task?

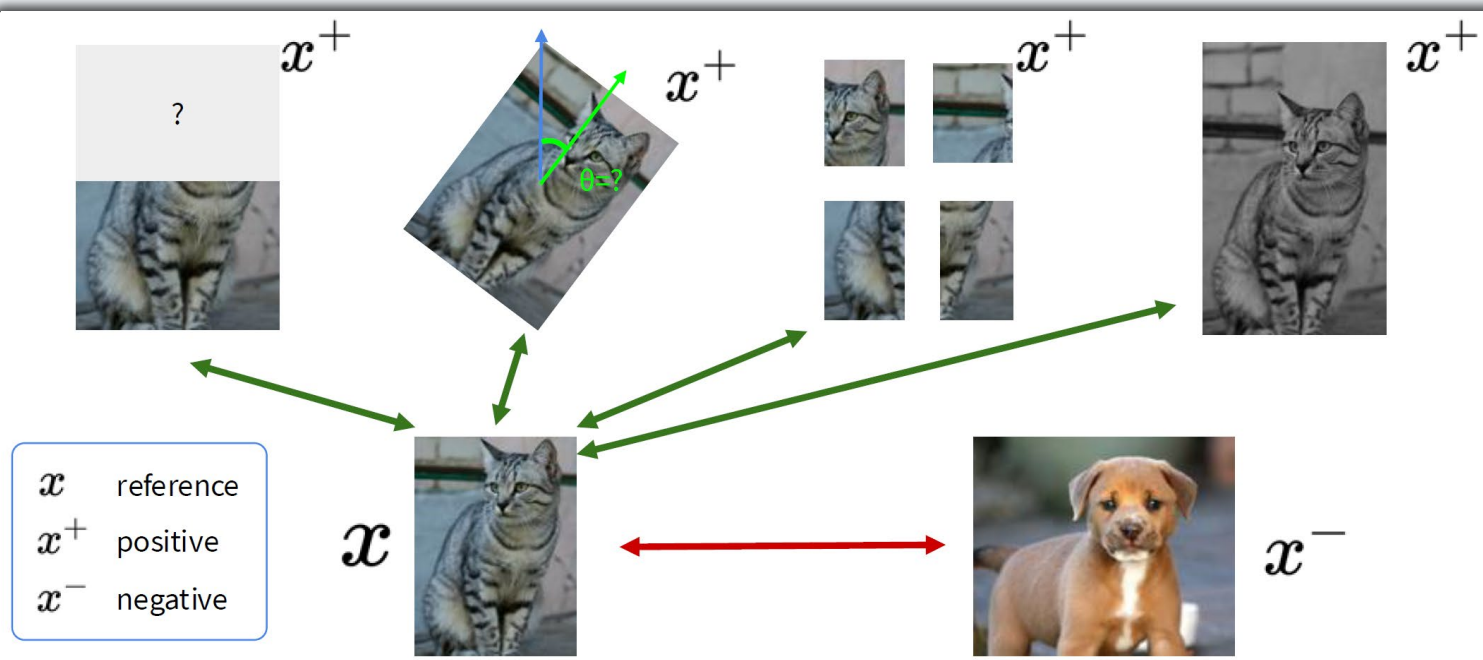


Part 3

Contrastive and Non-Contrastive SSL

Contrastive: SimCLR → MoCo / Non-contrastive: BYOL, SimSiam / Collapse problem

Contrastive Learning: Core Idea



Key Takeaway Pull together representations of the same image under different augmentations. Push apart representations of different images.

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

Formulation of contrastive learning

Loss function given 1 positive sample and N - 1 negative samples:

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-))} \right]$$

x

x^+

x

x_1^-

x_2^-

x_3^-

...

Formulation of contrastive learning

Loss function given 1 positive sample and N - 1 negative samples:

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-))} \right]$$

Commonly known as the InfoNCE loss (van den Oord et al., 2018)

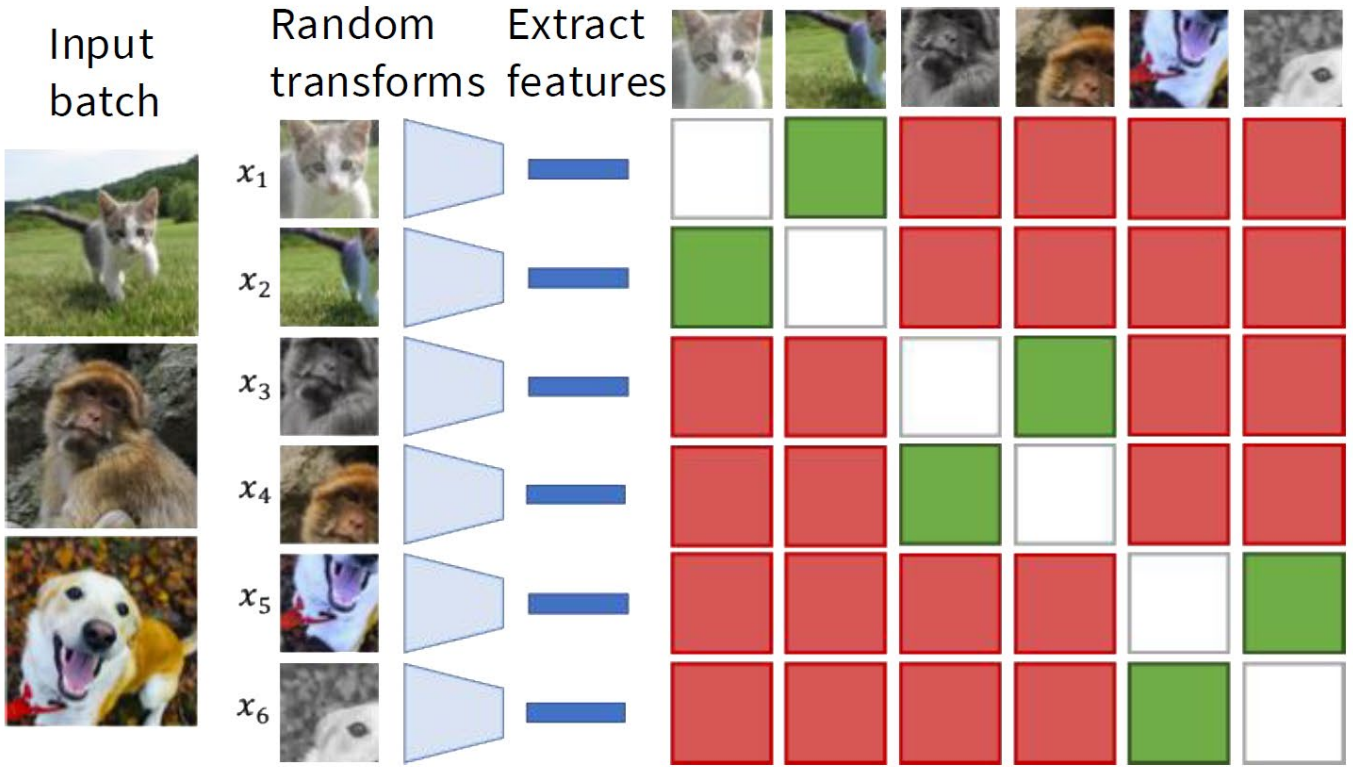
A lower bound on the mutual information between f(x) and $f(x^+)$

$$MI[f(x), f(x^+)] - \log(N) \geq -L$$

The larger the negative sample size (N), the tighter the bound

*Note: This bound is formally correct (CPC, van den Oord et al. 2018), but modern interpretations (Poole et al. 2019; Wang & Isola 2020) suggest alignment/uniformity may better explain contrastive learning's success than MI maximization.

SimCLR: A Simple Framework



Corresponding pairs should have similar features

Other pairs should have dissimilar features

Problem: Need large batch size with lots of negatives

SimCLR: A Simple Framework

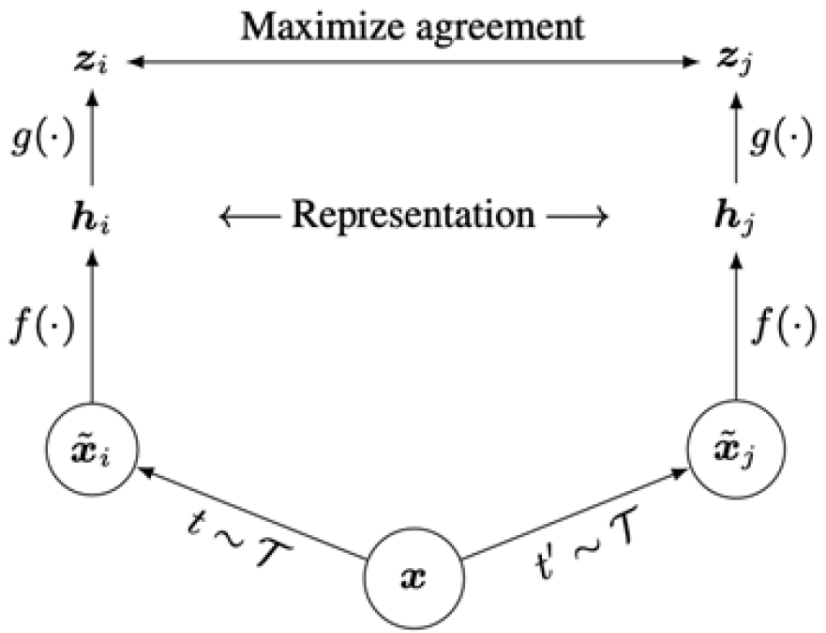
Cosine similarity as the score function:

$$s(u, v) = \frac{u^T v}{||u||||v||}$$

Use a projection network g(·) to project features to a space where contrastive learning is applied

Generate positive samples through data augmentation:

- random cropping, random color distortion, and random blur.



SimCLR: Why Augmentation Matters

The most important finding — not the architecture, but the data transforms

The shortcut problem

Random crop alone:
Two crops from the same image
share similar color distribution
→ Model learns a shortcut:
"same color histogram
= same image"
→ Never learns semantic features

Fix: Add color jittering
→ Forces model to learn
content, not color statistics

Ablation result

SimCLR ablation (Figure 5):

Crop only: ~33.1% (shortcut!)
Crop + Color: ~56.3% (+23.2%!)
Crop + Color + Blur: ~64.5% (Table 1)

Color jittering is the single
most important augmentation

Other useful transforms:
Gaussian blur
Random grayscale

Batch size = negatives

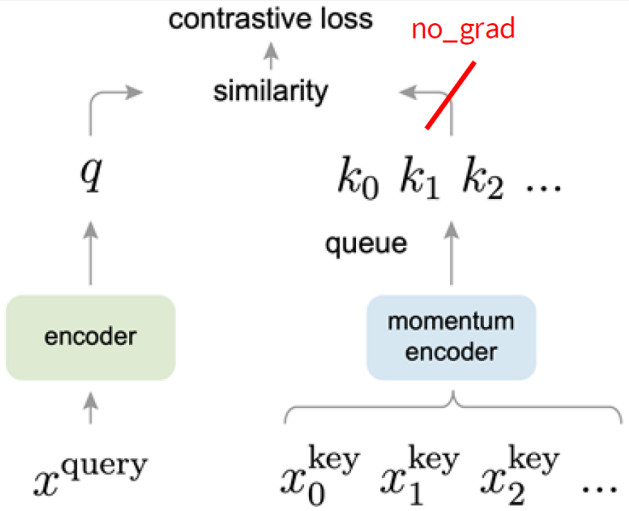
Batch size N=4096:
→ 2×4096 = 8192 views
→ Each anchor has
8192 − 2 = 8190 negatives

Larger batch = more negatives
= harder contrastive task
= better representations

But: requires 128 TPU v3 cores
→ ~$10K+ per training run

→ **MoCo** solves this

MoCo: Momentum Contrast



Key differences to SimCLR:

- Keep a running queue of keys (negative samples).
- Compute gradients and update the encoder only through the queries.
- Decouple min-batch size with the number of keys: can support a large number of negative samples.
- The key encoder is slowly progressing through the momentum update rules:

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

MoCo's solution

Decouple batch size from negatives:

1) Queue: store K=65,536 keys
from recent mini-batches
→ Many negatives, small batch OK

2) Momentum encoder:
$\theta_k \leftarrow 0.999 \times \theta_k + 0.001 \times \theta_q$
→ Keys stay consistent
(slowly updated, not stale)

MoCo v2: + SimCLR's MLP head
+ stronger augmentation → best of both

What is EMA?

Exponential Moving Average — a weighted running average that smoothly tracks the online encoder:

$\theta_{new} \leftarrow m \cdot \theta_{old} + (1-m) \cdot \theta_{current}$ where m ≈ 0.99–0.999

Target follows online slowly → stable, not stale. Same idea as SGD momentum & BN running stats.

The Collapse Problem

A trivial solution lurking in the "pull" objective

The Problem

The "pull" objective by itself:

$$minimize \;\; distance(f(x), f(x^+))$$

Trivial global minimum:

$$f(x) = c \quad for\ all\ x$$

→ distance = 0 ✓

→ but representation is useless ✗

This is representation collapse.

Every joint-embedding SSL method must solve it.

Not GAN mode collapse: the encoder maps everything to one point; not a few modes.

Two Paths Forward

Path 1: Contrastive

Add an explicit "push" term:

$$pull(x, x^+) + push(x, x^-)$$

Negatives block f → constant.

→ SimCLR, MoCo (just covered)

Path 2: Non-contrastive

Break the objective's symmetry

so $f = c$ is unreachable even without negatives.

→ BYOL, SimSiam, DINO (later)

Key Takeaway Collapse avoidance is the central design question of joint-embedding SSL. Reconstruction-based methods (MAE) escape this entirely; pixel targets have no trivial constant solution.

Beyond Negatives: BYOL & SimSiam

BYOL (Grill et al., NeurIPS 2020)

"Do we really need negatives?"

Student-teacher with EMA:

- Student: encoder + predictor
- Teacher: EMA of encoder (no grad)

$$L = \| p(f_\theta(x_1)) - sg(f_\xi(x_2)) \|^2$$

$$EMA\ update:\ \xi \leftarrow m \cdot \xi + (1-m) \cdot \theta$$

No negative pairs at all.

Collapse prevented by asymmetry: predictor on student + stop-grad on teacher.

Components:

Negatives ✗ · Momentum ✓ · Stop-grad ✓ · Predictor ✓

SimSiam (Chen & He, CVPR 2021)

"BYOL without momentum encoder"

Even simpler architecture:

- Shared encoder f for both views
- Predictor on one branch only

$$L = -\cos(\, p(f(x_1)), sg(f(x_2))\,)$$

$$symmetrized\ over\ x_1 \leftrightarrow x_2$$

No negatives, no momentum encoder.

★ Paper's Fig. 2: stop-grad alone prevents collapse.

EMA is a booster, not a requirement.

Components:

Negatives ✗ · Momentum ✗ · Stop-grad ✓ · Predictor ✓

Collapse Avoidance: A Taxonomy

The central design question of joint-embedding SSL

The Core Problem

Similarity maximization has a trivial minimum:

$$f(x) = constant \rightarrow loss = 0$$

Useless representation.

Every joint-embedding SSL method needs a **collapse-avoidance mechanism.**

(MAE is exempt: pixel reconstruction has no trivial constant solution.)

Five Mechanisms

Repulsion
negatives push apart
→ SimCLR, MoCo

Architectural asymmetry
predictor on one branch
→ BYOL, SimSiam

Temporal asymmetry
EMA teacher lags online
→ MoCo, BYOL, DINO

Gradient-flow asymmetry
stop-grad on target branch
→ BYOL, SimSiam, DINO

Output regularization
centering / decorrelation
→ DINO, Barlow Twins

The Deeper Question

Why does asymmetry work without negatives?

No complete theory yet.

SimSiam (Chen & He 2021):
EM-like alternating scheme.
Stop-grad = E-step target.
Predictor = M-step update.

Follow-ups (Tian 2021, Halvagal 2023): eigenspace dynamics interpretation.

Active research area!

Key Takeaway "Negatives vs. no-negatives" is the surface distinction. The deeper story is five mechanisms — repulsion, three asymmetries, and regularization — composed differently across methods.

One Lineage, Three Bottlenecks

Each method answers the bottleneck left by its predecessor

SimCLR
ICML 2020

Contrastive baseline: in-batch negatives (2N−2 per anchor) with NT-Xent loss.

↓ **Bottleneck:** *requires batch size 4096+ and ~128 TPU v3 cores*

MoCo
CVPR 2020

Queue (K=65K) decouples negatives from batch; momentum encoder keeps keys consistent.

↓ **Bottleneck:** *do we even need negatives?*

BYOL
NeurIPS 2020

Predictor + stop-grad + EMA teacher prevent collapse without any negatives.

↓ **Bottleneck:** *do we really need the EMA teacher?*

SimSiam
CVPR 2021

Shared encoder + stop-grad + predictor. Fig. 2: stop-grad alone prevents collapse; EMA is a booster.

Key Takeaway This is not a model gallery — it is a sequence of design questions, each narrowing what's essential.
Reading the arc backward: the minimum for non-contrastive SSL is stop-gradient + predictor.

Contrastive Methods: Comparison

SimCLR

Chen et al., ICML 2020

✓ Simple architecture
✓ Strong augmentation
✗ Large batch required (4096+)
✗ Many TPUs needed
Loss: NT-Xent (InfoNCE)
Negatives: in-batch (2N−2)

Components:
Negatives ✓ · Momentum ✗ · Stop-grad ✗ · Predictor ✗

MoCo

He et al., CVPR 2020

✓ Small batch OK
✓ Queue of 65K keys
✓ Momentum encoder
Loss: InfoNCE
Negatives: from queue

v2: + MLP head
+ stronger augmentation

Components:
Negatives ✓ · Momentum ✓ · Stop-grad ✓ · Predictor ✗

BYOL

Grill et al., NeurIPS 2020

✓ No negatives
✓ EMA teacher
✓ Predictor network
Loss: MSE (cosine sim)
Negatives: none!

Needs momentum encoder + predictor + stop-gradient

Components:
Negatives ✗ · Momentum ✓ · Stop-grad ✓ · Predictor ✓

SimSiam

Chen & He, CVPR 2021

✓ No negatives
✓ No momentum encoder
✓ Simplest architecture
Loss: cosine similarity
Negatives: none!

Needs predictor + stop- gradient

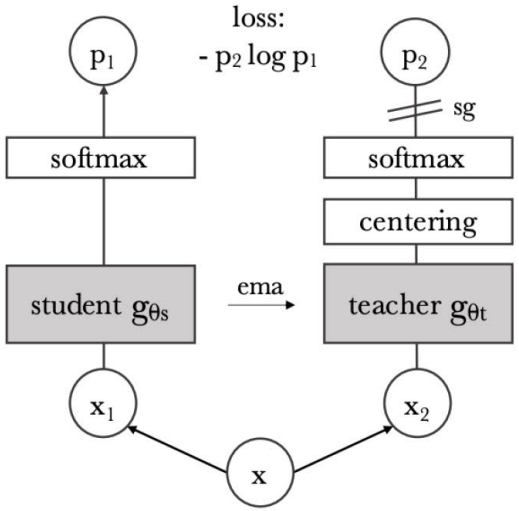
Components:
Negatives ✗ · Momentum ✗ · Stop-grad ✓ · Predictor ✓

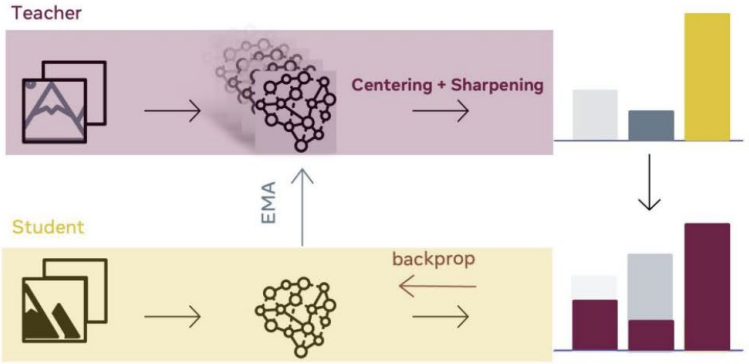
Part 4

Self-Distillation in Vision Transformers

DINO → DINOv2

DINO: Self-Distillation with No Labels





$$\theta_{\text{teacher}} \leftarrow \tau \cdot \theta_{\text{teacher}} + (1 - \tau) \cdot \theta_{\text{student}}$$

Key Takeaway Teacher-student self-distillation with ViT. Teacher = EMA of student. Cross-entropy loss (not dot product). Centering + sharpening prevent collapse.

DINO: Emerging Semantic Segmentation

"self-supervised ViT features contain explicit information about semantic segmentation" — paper abstract

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.

- Each attention head learns to focus on different semantic regions
- This does NOT emerge with supervised ViTs or CNNs
- kNN classifier: 78.3% top-1 on ImageNet (no fine-tuning!)
- Linear probe: 80.1% top-1 with ViT-Base/8
- **→ SSL + ViT produces features with built-in spatial understanding**

DINOv2: Why Two Losses?

Oquab et al., TMLR 2024 — learning principle focus

DINO loss (global)

Image-level self-distillation

Student & teacher see different crops of same image:

Teacher: global crop (224^2)

Student: global + local (96^2)

Student predicts teacher's output distribution (CLS token)

Cross-entropy loss + Centering + Sharpening

→ **Learns global semantics**

("what is this image?")

iBOT loss (local)

Patch-level masked prediction

Mask some patches in student view

Teacher sees all patches

Student predicts teacher's representation of masked patches

"Online tokenizer":

Teacher = tokenizer (EMA updated)

→ No separate dVAE (unlike BEiT)

→ Single-stage training

→ **Learns local structure**

("what's in this region?")

Why combine them?

Global only (DINO):

Strong cls, weak dense pred

Local only (iBOT/MAE):

Strong seg/depth, weaker cls

DINOv2 = DINO + iBOT:

Best of both worlds

ViT-g probe: 86.5% (> OpenCLIP)

Frozen features work across all tasks

Data, scale → Wk9 Foundation Models

DINOv2: Emergent Part Correspondences

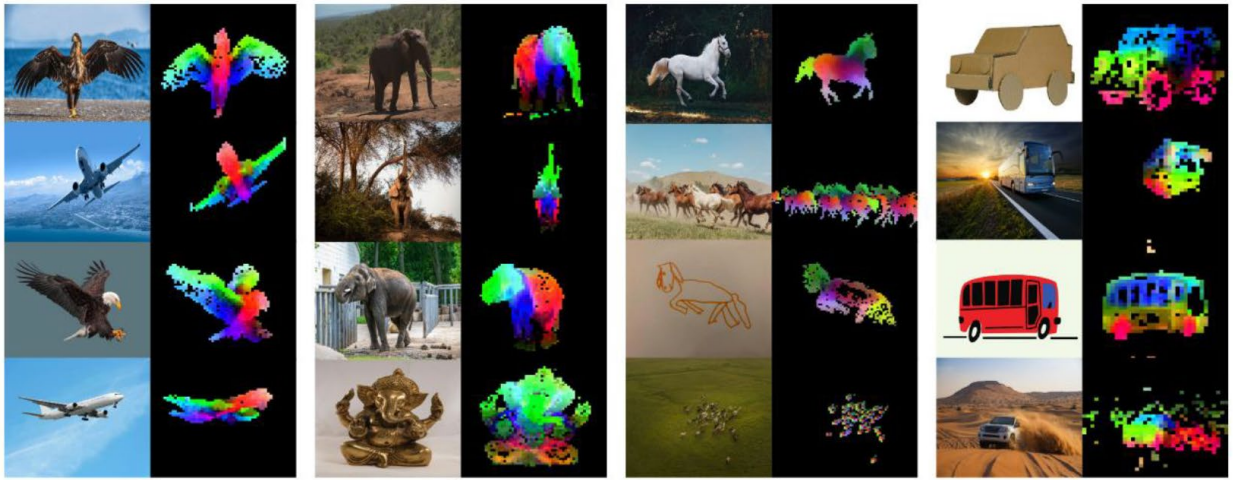


Figure 1: **Visualization of the first PCA components.** We compute a PCA between the patches of the images from the same column (a, b, c and d) and show their first 3 components. Each component is matched to a different color channel. Same parts are matched between related images despite changes of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

Key Takeaway Find the crossover: DINOv2 (no labels) matches or surpasses weakly-supervised methods trained with text supervision.

The Student-Teacher Framework

One pattern, many implementations — teacher architecture × target space

Model	Teacher architecture	Target space	Loss
MoCo	EMA encoder	feature (contrastive)	*InfoNCE*
BYOL	EMA encoder	feature (MSE)	$\|\|\cdot\|\|^2$
SimSiam	shared + stop-grad	feature (cosine)	$-cos$
BEiT	frozen dVAE	discrete tokens (8192-way)	*cross-entropy*
DINO	EMA ViT	distribution (CLS token)	*CE + centering*
iBOT	EMA ViT	patch distribution (masked)	*CE (patch-level)*
DINOv2	EMA ViT	both: DINO + iBOT targets	*CE (global + local)*

Two orthogonal axes

Teacher source: EMA (co-evolves) / frozen / shared+sg

Target space: feature / distribution / discrete tokens

★ "Online tokenizer" demystified

iBOT / DINOv2's "tokenizer" is an **EMA teacher ViT** producing **soft distributions** — not discrete tokens, despite the name. Contrast with BEiT's *frozen dVAE* over an 8192-class vocabulary.

Key Takeaway Student-teacher is one pattern, but "teacher" means different things across methods. The EMA family (MoCo → BYOL → DINO → DINOv2) shares a co-evolving target mechanism.

SSL → Detection: DINOv2 as Backbone

The connection to Week 6

Traditional pipeline

ImageNet supervised pretrain
→ Detection fine-tune

R-CNN (2014) to DETR (2020):
always used supervised ImageNet backbone

ResNet / Swin → Faster R-CNN
ResNet → DETR
Swin → DINO-DETR

The SSL pipeline (2025)

DINOv2 (SSL pretrain)
→ RF-DETR decoder
→ Real-time up to 60+ AP on COCO

SSL pretrained backbone
outperforms supervised backbone for detection!

RF-DETR (Robinson et al. ICLR 2026, Roboflow)

→ Paradigm shift:
At scale, SSL backbones can match or surpass supervised pretraining on several downstream vision tasks

What's next? → Wk9

DINOv3 (Meta, 2025):
7B params → distill to ViT-L/B

Not just "bigger SSL" — foundation model question:

How to curate 1.7B images?
How to stabilize at scale?
How to deploy efficiently?

This is Week 9's topic: Foundation Models

*Note: DINO (SSL) ≠ DINO-DETR (detection)

Part 5

Wrap-up & Forward

Three paradigms, one destination

Three SSL Paradigms

Reconstruction (MAE)

Reconstruct masked patches
Target: pixels or tokens

Strength:
- Simple, scalable, fast

Weakness:
- Frozen features are weak
- → Fine-tuning required

Best for:
- Large-scale pretraining
- when fine-tuning is planned

Key paper: MAE (CVPR 2022)

Contrastive (SimCLR/MoCo)

Match positive pairs,
repel negative pairs
Target: feature similarity

Strength:
- Strong linear probe
- Robust representations

Weakness:
- Large batch or queue needed
- Augmentation-sensitive

Best for:
- Transfer learning
- with limited labels

Key papers: SimCLR, MoCo

Self-Distillation (DINO)

Teacher-student
cross-entropy matching
Target: probability distribution

Strength:
- Best frozen features
- Semantic segmentation
- emerges without labels

Weakness:
- Complex training dynamics
- (centering, sharpening)

Best for:
- Universal backbone
- (DINOv2 → RF-DETR)

Key papers: DINO, DINOv2

The Emerging 4th Way: I-JEPA

The problem with existing approaches

MAE: predicts pixels
→ Wastes capacity on low-level detail (texture, edges) not semantics

SimCLR / DINO: need augmentations
→ Hand-designed invariances
→ Biased toward specific tasks

What if we could predict in representation space, without any augmentation?

I-JEPA: predict representations

Core idea:
- Mask context block → predict target
- In representation space (not pixels!)

No augmentation, no negatives, no pixel reconstruction

Efficiency:
ViT-H/14 in <72h (16 A100s)
→ 10× more efficient than MAE
→ 2.5× faster than iBOT

Strong on semantic + low-level tasks
V-JEPA: extended to video (2024)

SSL Timeline: 2018–2025

The arc of SSL Pretext tasks (hand-designed) → Contrastive (need negatives) → Self-distillation (no negatives) → Predict in latent space (I-JEPA, no augmentation) → Universal frozen features (DINOv2/v3).
SSL backbone now powers detection (RF-DETR), segmentation (SAM), and multimodal models (CLIP → Wk9).

Preview: Foundation Models (Wk9)

Path 1: SSL → Detection

DINOv2 (this week)
→ RF-DETR (Wk6)

SSL features as universal
backbone for vision tasks

No text, no labels
→ Pure visual learning

Path 2: SSL + Language

CLIP (Radford et al., ICML 2021)
→ Image-text contrastive
→ 400M image-text pairs

CLIP → Grounding DINO (Wk6)
→ Open-vocabulary detection

These two paths converge
in Wk9: Foundation Models
= SSL + scale + (optional) text
→ Universal visual intelligence

Summary & Resources

What We Covered

SSL learns representations from unlabeled data via pretext tasks

MAE: mask 75%, reconstruct pixels — simple but needs fine-tuning

SimCLR: contrastive + augmentation — color jittering is the key
MoCo: queue decouples batch size from negative count
BYOL/SimSiam: stop-gradient removes the need for negatives

DINO: self-distillation → semantic segmentation emerges without labels
DINOv2: universal frozen features → powers RF-DETR (Wk6)

Further Reading & Self-Study

Stanford CS231n L12, 13

Lilian Weng: Self-Supervised (2019) & Contrastive (2021)
lilianweng.github.io

Lightly AI - DINOv2 Deep Dive
https://www.lightly.ai/blog/dinov2

Balestriero et al. “A Cookbook of Self-Supervised Learning”

Next Week Week 8: Review Literacy + Role Explanation (for Miniconference!)

Acknowledgments

Some slides and figures in this course are adapted from or inspired by the following open resources.

- **Stanford CS231n: Deep Learning for Computer Vision (Spring 2025)**
  Fei-Fei Li, Ehsan Adeli, et al. | cs231n.stanford.edu
- **UMich EECS 498/598: Deep Learning for Computer Vision (Winter 2022)**
  Justin Johnson | web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/
- **MIT 6.7960: Deep Learning (Fall 2024)**
  Phillip Isola, Sara Beery, Jeremy Bernstein | ocw.mit.edu/courses/6-7960-deep-learning-fall-2024/
- **CMU 16-824: Visual Learning and Recognition (Fall 2025)**
  Jun-Yan Zhu | visual-learning.cs.cmu.edu
- **Key papers: CPC, SimCLR, MoCo, BYOL, SimSiam, MAE, SimMIM, BEiT, iBOT, DINO, DINOv2, I-JEPA, …**