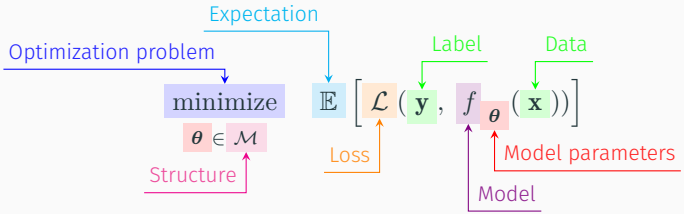
# Self-supervised Learning

RCP 209, 2025 – 2026

Javiera Castillo Navarro

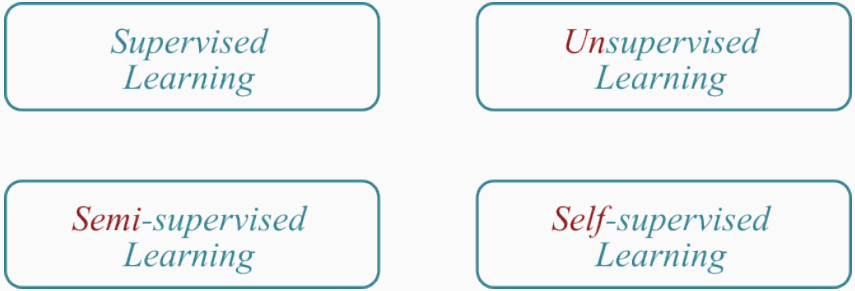
le cnam

# Quick recap




- **Design** prediction model $\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$ and loss $\mathcal{L}$
- **Learn** the model parameters approx. $\mathbb{E}$ with data at hand + solve the optimization problem
- **Apply** model to new data

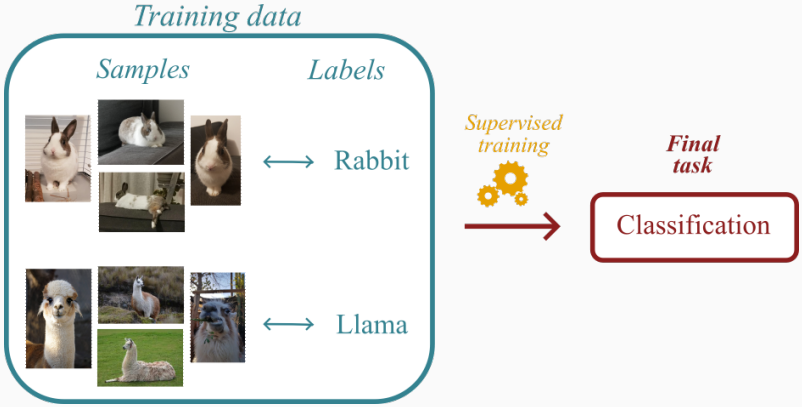
# Learning paradigms



# Supervised learning

- Given a **labeled dataset** of input-output pairs: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, with $x_i$: samples (*e.g.* images) and $y_i$: labels
- The **goal** is to learn a predictive function: $f_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by $\theta$ that maps inputs to their corresponding label.
- Training consists of **minimizing a supervised loss** over the dataset: $\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i)$

# Supervised learning




⇝ **Labels provide the supervisory signal:**
they **explicitely** tell the model what the **correct output** is.

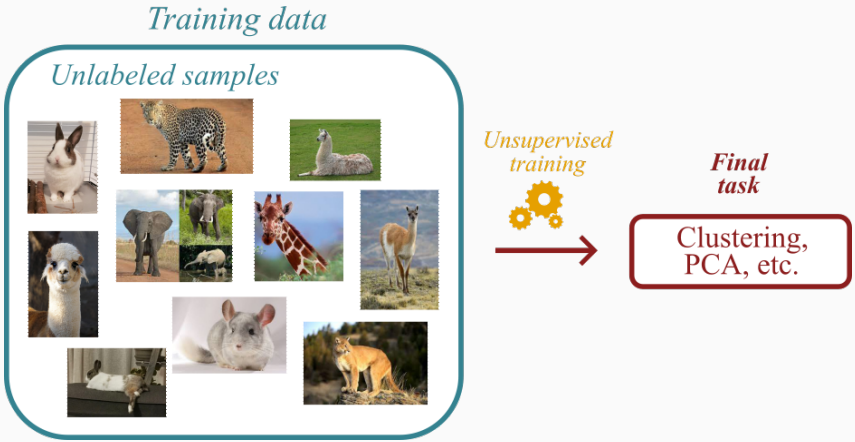
# Limitations of supervised learning

- **Data limitations**
  - → Still Requires labeled data ⇝ expensive and time-consuming
  - → Labels may be noisy, inconsistente or ambiguous
- **Scalability limitation**
  - → Annotation cost grows **linearly** with dataset size
  - → Many domains (medicine, rare events) have inherently limited labeled data
- **Generalization limitations**
  - → Poor performance in low data regimes or under distribution shifts
  - → Models learn **task-specific** representations ⇝ do not transfer to other tasks

# Unsupervised learning

- Given an **unlabeled dataset**: $\mathcal{D} = \{x_i\}_{i=1}^N$, with $x_i$: samples (*e.g.* images)
- The **goal** is to learn a function or representation: $f_\theta : \mathcal{X} \to \mathcal{Z}$, parameterized by $\theta$ that **captures structure or regularities** in the data: clusters, density, *etc.*
- Unsupervised learning aims to solve: $\min_\theta \mathcal{L}(f_\theta(x), x)$ or $\max_\theta \log p_\theta(x)$, depending on the model class.
- Unsupervised tasks:
  - Density estimation: model $p_\theta(x)$
  - Clustering: group samples by similarity
  - Dimensionality reduction: find low-dimensional latent representations

# Unsupervised learning




⇝ **No explicit supervisory signal,**
learning is guided by intrinsic properties of the data.

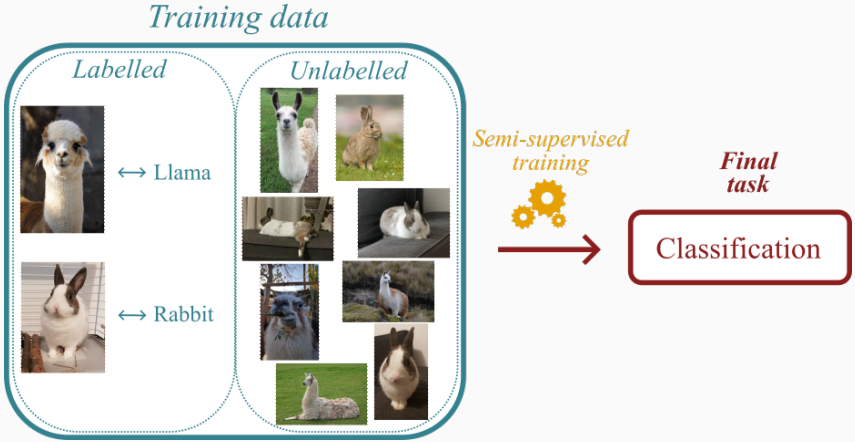
# Limitations of unsupervised learning

- **Ambiguity of objetives**
  - → Without labels, the model may learn structure that is irrelevant for some tasks
- **Weak supervisory signal**
  - → Reconstruction or density estimation do not guarantee discriminative features
- **Evauation limitations**
  - → No ground-truth ⇝ evaluation of the results is often subjective or indirect
- **Lack of semantic richness**
  - → Purely unsupervised features often **fail to capture semantic concepts** (e.g. object identity)

# Semi-supervised learning

- Given a dataset composed of a **small labeled subset**, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N_\ell}$, and a **large unlabeled subset**, $\mathcal{D} = \{x_i\}_{i=1}^{N_u}$, with $N_u >> N_\ell$
- The **goal** is to learn a predictive function: $f_\theta : \mathcal{X} \to \mathcal{Y}$, (same as supervised learning), **leveraging both types of data to improve performance**.
- Training usually combines:
  - **Supervised loss** on labeled samples:
    $$\mathcal{L}_{\text{sup}} = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \mathcal{L}(f_\theta(x_i), y_i)$$
  - **Unsupervised loss** (consistency/regularization) on unlabeled samples:
    $$\mathcal{L}_{\text{unsup}} = \frac{1}{N_u} \sum_{i=1}^{N_u} \mathcal{R}(f_\theta(x_i), x_i)$$

# Semi-supervised learning




⇝ **Assumption:** **Unlabeled data can reveal structure aligned with the supervised task**

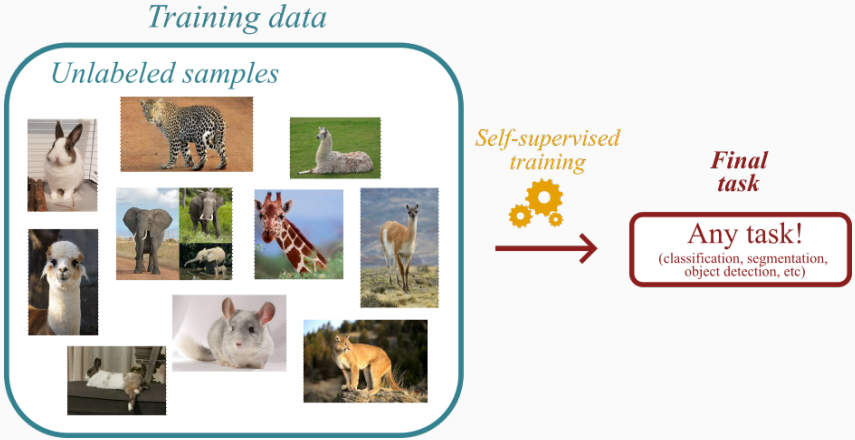
# Limitations of semi-supervised learning

- **Data limitations**
  - → Requires **large amounts** of labeled data ⇝ expensive and time-consuming
  - → Labels may be noisy, inconsistente or ambiguous
- **Quality of unlabeled data**
  - → Unlabeled data must follow the **same distribution** as the labeled data
  - → Unlabeled data can harm performance
- **Training instability**
  - → The unsupervised loss needs to be carefully chosen and tuned.
- **Limited semantic gains**
  - → They are not better than supervised learning when labeled data is abundant
  - → **Often surpassed by self-supervised strategies**

# Self-supervised learning

- Given an (usually big) **unlabeled dataset**: $\mathcal{D} = \{x_i\}_{i=1}^N$
- We define a **pretext task** and generate labels from the data itself $y_i^{\text{pretext}}$
- Now we have a **labeled dataset** $\mathcal{D} = \{(x_i, y_i^{\text{pretext}})\}_{i=1}^N$ to train $f_\theta$ in a **supervised way** on the pretext task.
- The **goal** is that the pretext task should help $f_\theta(x)$ to capture **useful representations** that transfer to downstream tasks.
- **Downstream task:** the real task of interest (*e.g.*, classification, segmentation, object detection, *etc.*)

# Self-supervised learning



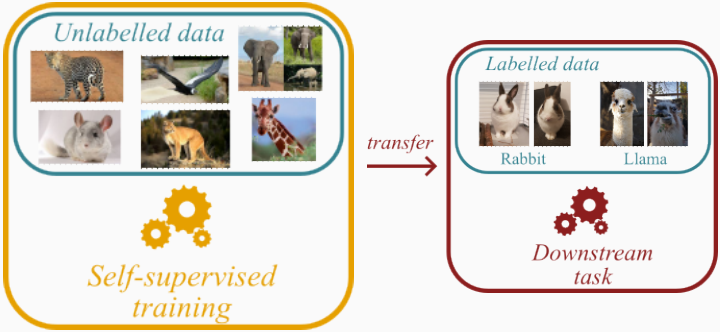

⇝ Use **intrinsic structure** in the data to create supervision **without human labels**.

# Why self-supervised learning?

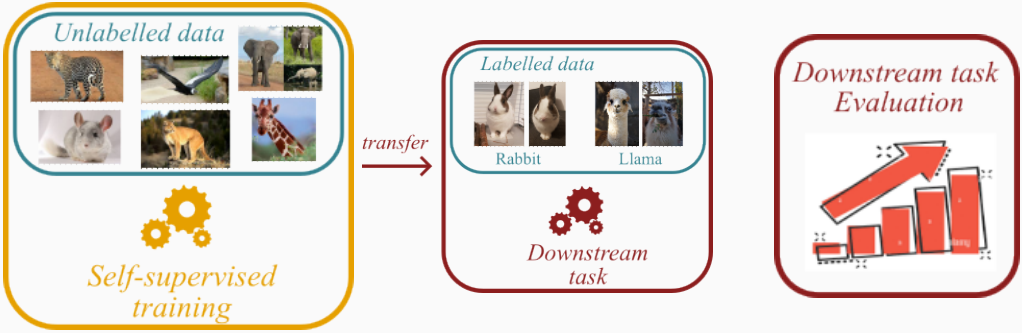
- **Labels are expensive**
  - → Unlabeled data is abundant at (almost) no cost
  - → Labels require expert knowledge and time
- **Supervised learning is limited**
  - → A single label contains very few bits of information, discarding all other information contained in images
  - → Does not transfer to other tasks/datasets
- **Unsupervised learning does not guarantee semantic features**
  - → Traditional unsupervised methods may capture irrelevant structure
- **SSL scales naturally**
  - → SSL objectives operate on **raw data at massive scale**
  - → Enables training very large models
  - → Learns general-purpose, **transferable representations**

# Evaluation of self-supervised learning

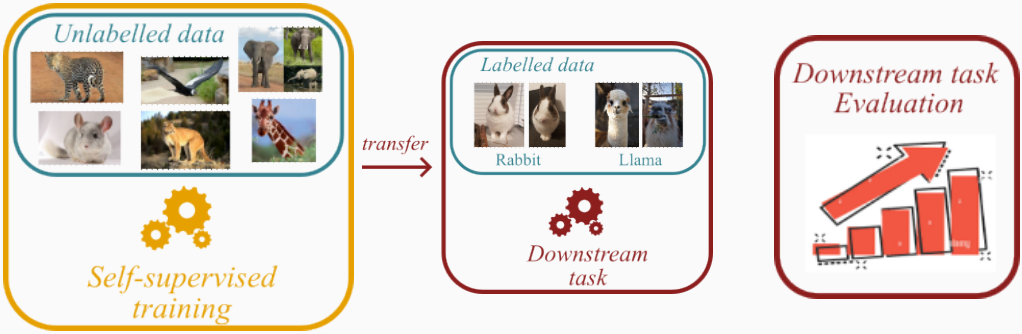
# Evaluation of self-supervised learning



# Evaluation of self-supervised learning

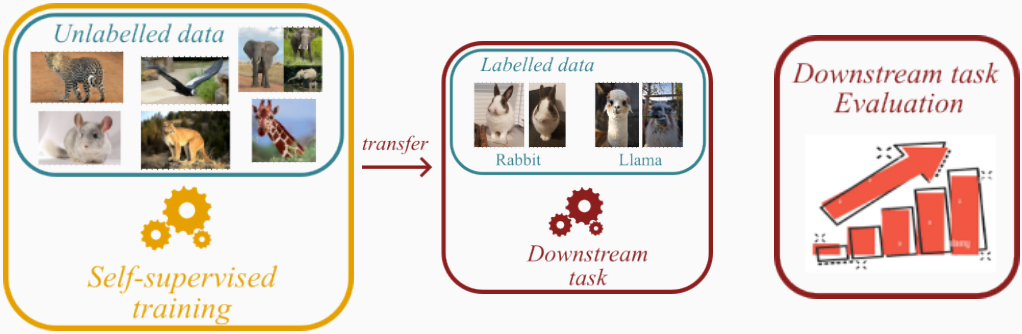


# Evaluation of self-supervised learning




→ Pretext task evaluation

# Evaluation of self-supervised learning



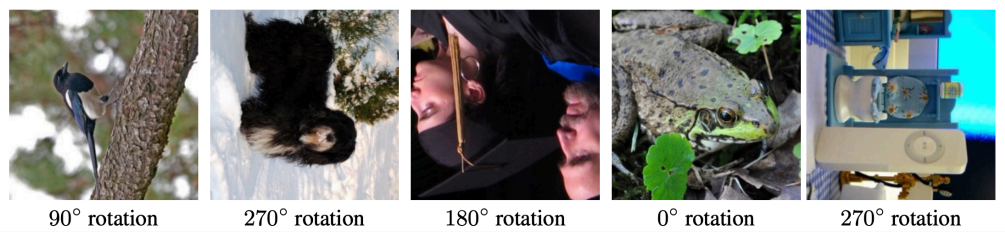

→ Pretext task evaluation

→ Evaluate the representations

- Fine-tuning
- Linear probing
- $k$-NN evaluation

# Prior self-supervised learning

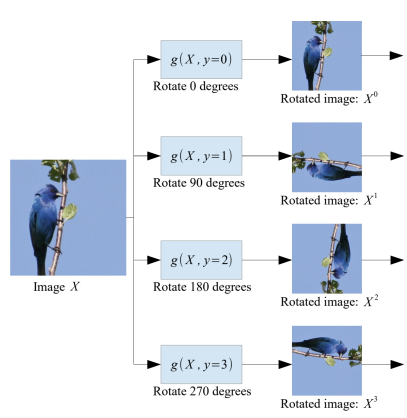
# Pretext task: Predict Rotations




**Hypothesis:** A model would be able to recognize the correct rotation of an object only if it has the *"visual commonsense"* of what the object should look like unperturbed.

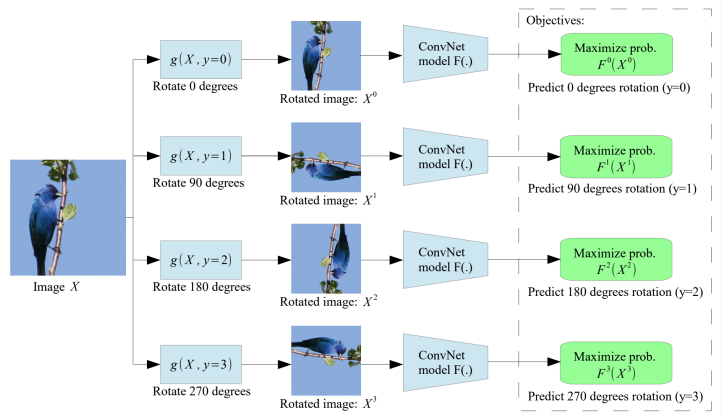
Image source: Gidaris et al. 2018

# Pretext task: Predict rotations



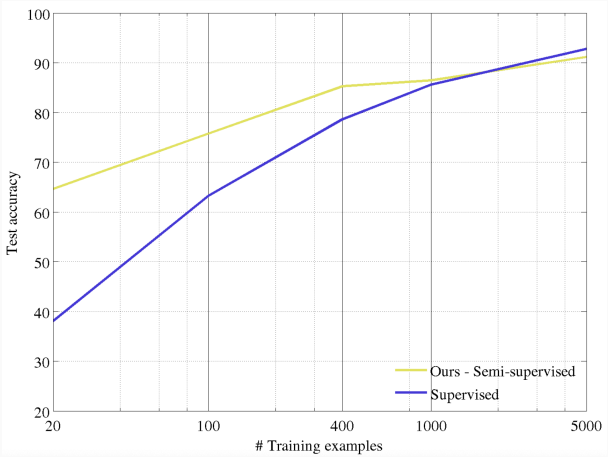

- SSL by rotating the inputs
- The model learns to predict the rotation angle
- 4-class classification

# Pretext task: Predict rotations




- SSL by rotating the inputs
- The model learns to predict the rotation angle
- 4-class classification

# Evaluation on semi-supervised learning




- Self-supervised learning on CIFAR10 (entire training set)
- Conv1 & Conv2 frozen
- Learn Conv3 & linear layers with subsets of labeled CIFAR10 data

# Transfer learned features to supervised learning

- Self-supervised learning on ImageNet (entire training set) with AlexNet
- Finetuning on labeled data from Pascal VOC 2007.

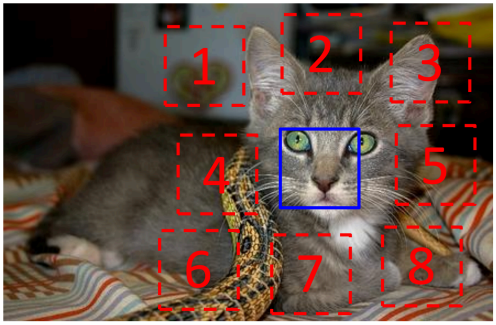
| | Classification (%mAP) | | Detection (%mAP) | Segmentation (%mIoU) |
|---|---|---|---|---|
| Trained layers | fc6-8 | all | all | all |
| ImageNet labels | 78.9 | 79.9 | 56.8 | 48.0 |
| Random | | 53.3 | 43.4 | 19.8 |
| Random rescaled Krähenbühl et al. (2015) | 39.2 | 56.6 | 45.6 | 32.6 |
| Egomotion (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 on ImageNet (supervised)

No pretraining

Self-supervised (rotation prediction)

# Pretext task: Predict relative patch locations



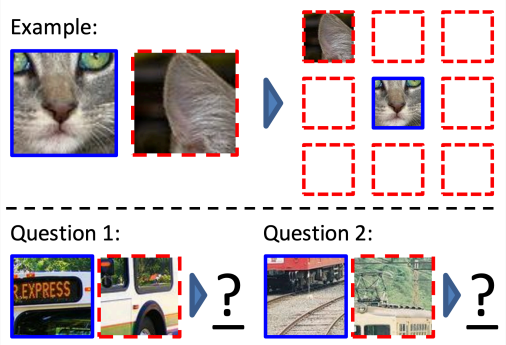
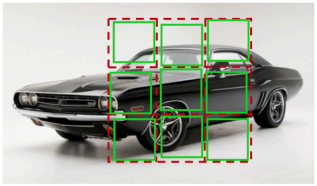




Image source: Doersch et al. 2015

# Pretext task: Solving *"jigsaw puzzles"*



Permutation Set

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

Reorder patches according to the selected permutation

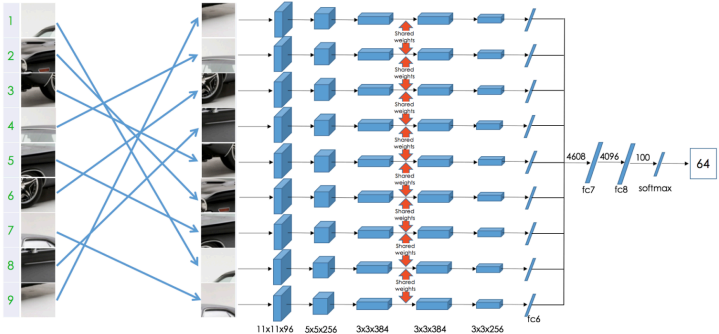



Image source: Noroozi & Favaro, 2016.

# Transfer learned features

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%** |

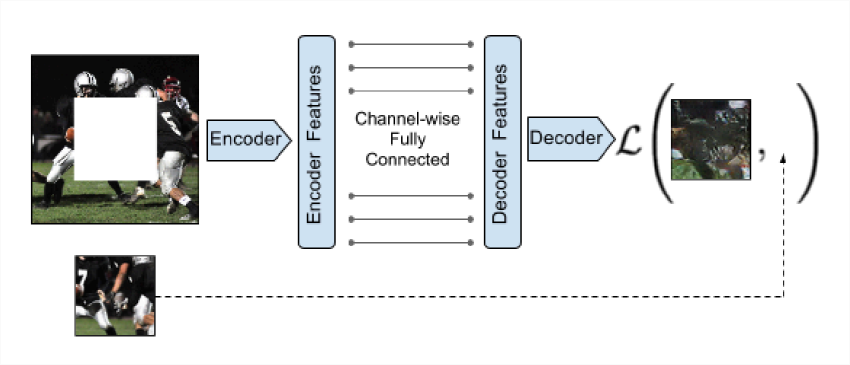
Relative patch location

Solving Jigsaw Puzzles

# Pretext task: Predict missing pixels (inpainting)


Pathak et al., “Context encoders: learning by inpainting”, *CVPR* 2016.

# Learning *inpainting* by reconstruction




Learning to **reconstruct** the missing pixels using **autoencoders**.

# Learning to inpaint by reconstruction

$$\mathcal{L}(x) = \mathcal{L}_{\text{reconstruction}}(x) + \mathcal{L}_{\text{adversarial}}(x)$$

# Learning to inpaint by reconstruction

$$\mathcal{L}(x) = \mathcal{L}_{\text{reconstruction}}(x) + \mathcal{L}_{\text{adversarial}}(x)$$

$$\mathcal{L}_{\text{reconstruction}}(x) = ||M \odot (x - F_\theta(1 - M) \odot x)||_2^2$$

with $\odot$: element-wise multiplication and $M$ a masked matrix with 1 if element is masked (else, 0)

# Learning to inpaint by reconstruction

$$\mathcal{L}(x) = \mathcal{L}_{\text{reconstruction}}(x) + \mathcal{L}_{\text{adversarial}}(x)$$

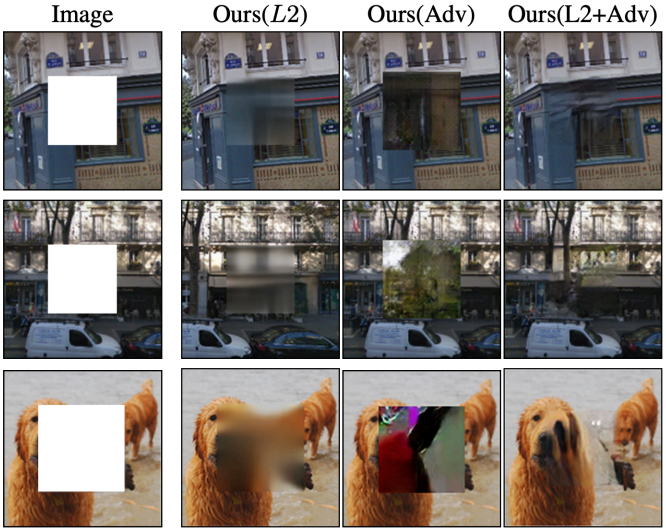
$$\mathcal{L}_{\text{reconstruction}}(x) = ||M \odot (x - F_\theta(1 - M) \odot x)||_2^2$$

with $\odot$: element-wise multiplication and $M$ a masked matrix with 1 if element is masked (else, 0)

$$\mathcal{L}_{\text{adversarial}}(x) = \max_D \mathbb{E}[\log(D(x)) + \log(1 - D(F((1 - M) \odot x)))]$$

adversarial loss between “real” images and inpainted ones.

# Inpainting evaluation

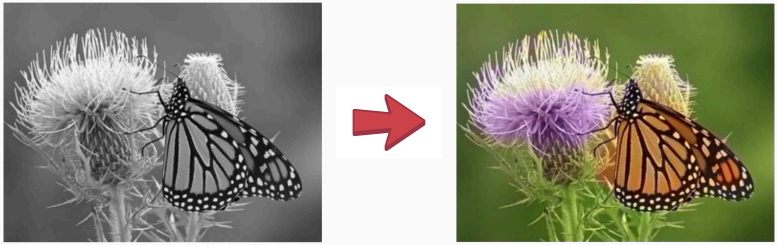


# Transfer learned features

| 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% |

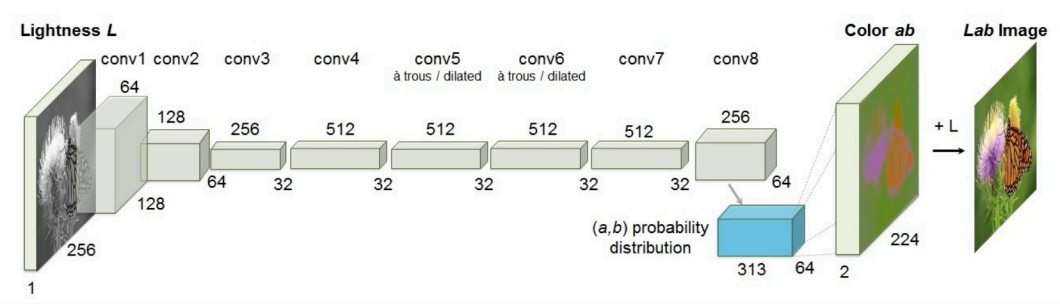
- Self-supervised training on ImageNet training set
- Transfer to classification (PASCAL VOC 2007), detection (PASCAL VOC 2007) and semantic segmentation (PASCAL VOC 2012)

# Pretext task: Colorization

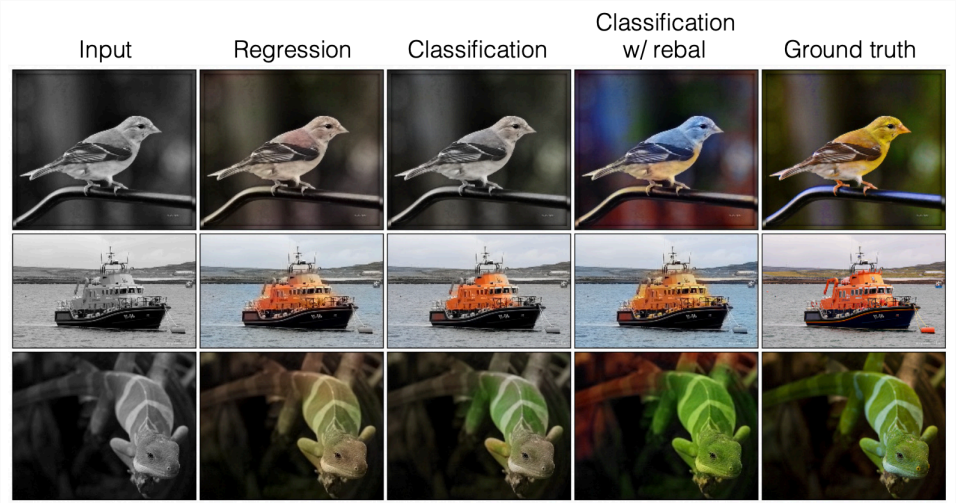


Zhang et al., “Colorful Image Colorization”, *ECCV* 2016.

# Pretext task: Colorization



# Pretext task: Colorization



# Limitations

- **Learn task-specific, non-generalizable features**
  - → Many early tasks encourage learning low-level cues (edges, textures, artifacts) rather than semantic features.
  - → Representations often do not transfer well to downstream vision tasks.
- **Easy to solve via shortcuts**
  - → Models frequently exploit trivial correlations (e.g., border artifacts for jigsaw, chromatic aberration for colorization).
  - → The pretext task can be solved without true understanding of objects or scenes.
- **Based on intuitions, with no theoretical guarantees**
  - → Pretext tasks (rotation, inpainting, colorization...) were designed based on heuristics, with no guarantee that solving the task yields useful representations.

# Families of self-supervised learning models today

*Contrastive Learning*

SimCLR, MoCo

*Masked Image Modeling*

MAE, SimMIM, MultiMAE

*Self-distillation*

BYOL, SiamSiam, DINO

*Canonical correlation Analysis*

VICReg, Barlow Twins, SWAV

Balestriero et al., “A cookbook of self-supervised learning”, 2023.

# Families of self-supervised learning models today

*Contrastive Learning*

SimCLR, MoCo

*Masked Image Modeling*

MAE, SimMIM, MultiMAE

*Self-distillation*

BYOL, SiamSiam, DINO

*Canonical correlation Analysis*

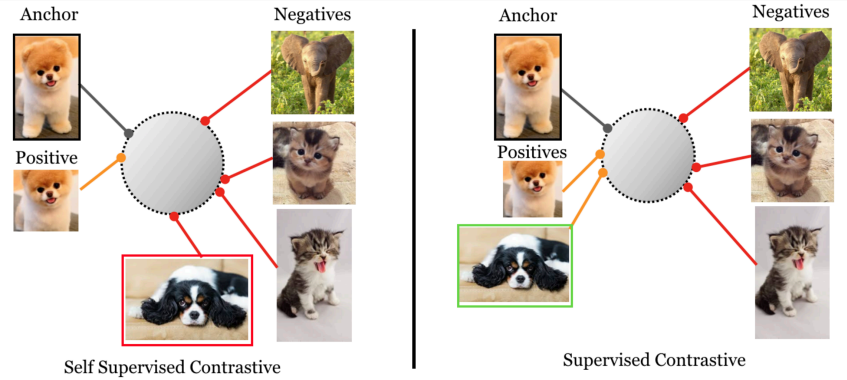
VICReg, Barlow Twins, SWAV

Balestriero et al., “A cookbook of self-supervised learning”, 2023.

# Contrastive Learning

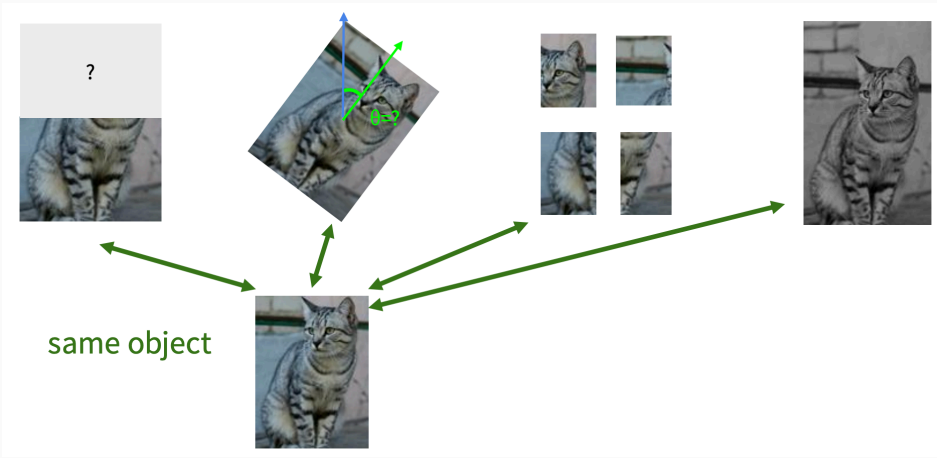
# Contrastive representation learning

**Goal:** To learn an **embedding space** in which **similar samples are close** to each other while **dissimilar ones are far apart**.

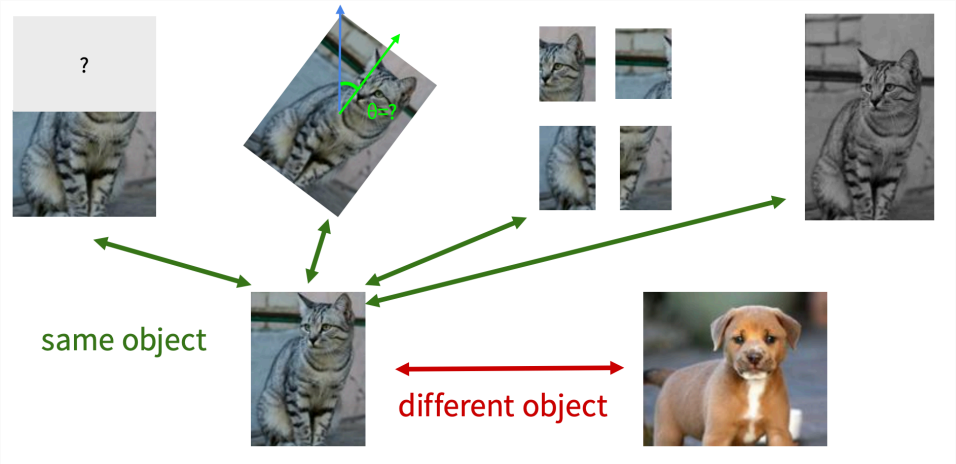



Khosla et al., "Supervised Contrastive Learning, *NeurIPS* 2020.

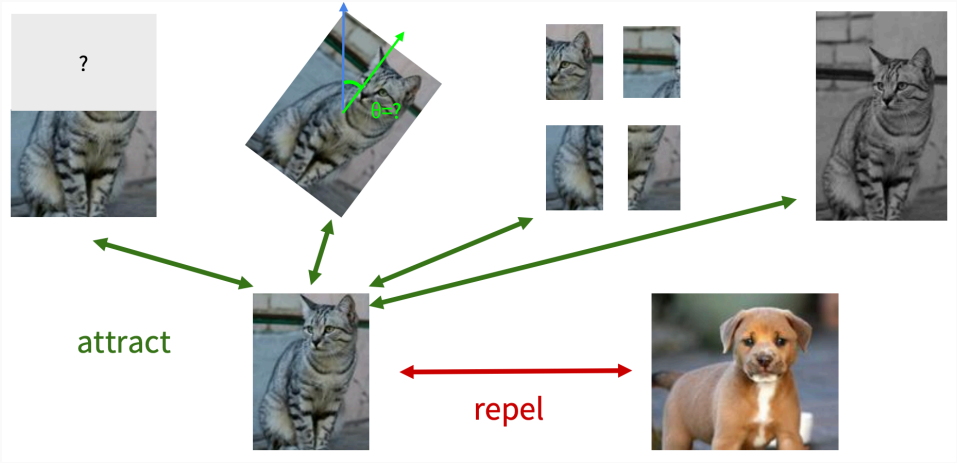
# A more general pretext task?



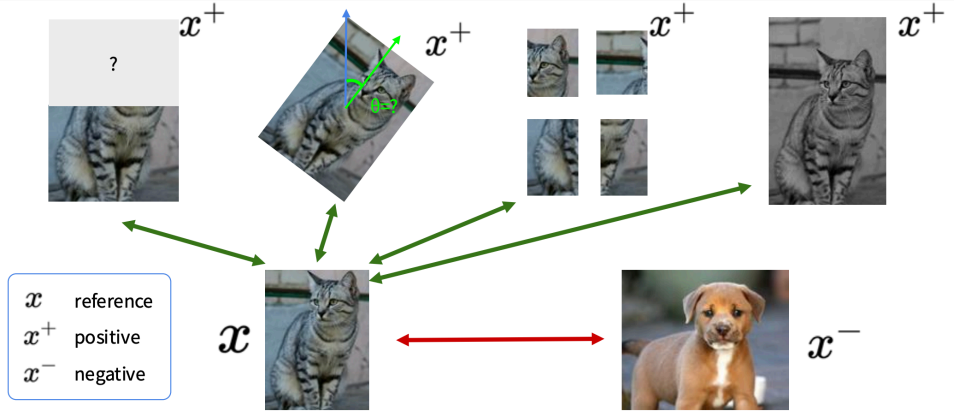
# A more general pretext task?



# Contrastive representation learning



# Contrastive representation learning



# A formulation of contrastive learning

We want:

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

with $x$: reference sample; $x^+$: positive sample; $x^-$:negative sample.

# A formulation of contrastive learning

We want:

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

with $x$: reference sample; $x^+$: positive sample; $x^-$:negative sample.

Given a **score function** we aim to learn a function $f$ that yields **high score for positive pairs** $(x, x^+)$ and low scores for negative pairs $(x, x^-)$

# A formulation of contrastive learning

Loss function given 1 positive sample and (N-1) negative samples

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

# A formulation of contrastive learning

Loss function given 1 positive sample and (N-1) negative samples

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

# A formulation of contrastive learning

Loss function given 1 positive sample and (N-1) negative samples

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

Does this look familiar?

# A formulation of contrastive learning

Loss function given 1 positive sample and (N-1) negative samples

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

Does this look familiar?

$\rightsquigarrow$ **Cross-entropy loss for a N-way softmax classifier!**

$\rightarrow$ learn to find de positive sample among $N$ samples

# A formulation of contrastive learning

Loss function given 1 positive sample and (N-1) negative samples

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

- Commonly known as **InfoNCE loss**
- It is a **lower bound of the mutual information** between $f(x)$ and $f(x^+)$

$$MI(f(x); f(x^+)) - \log N \geq -\mathcal{L}_{\text{InfoNCE}}$$

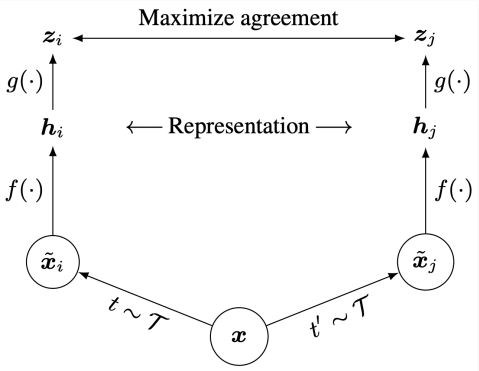
van den Oord et al., "Representation learning with contrastive predictive coding", 2018.

# SimCLR: A simple framework for contrastive learning

- **Cosine similarity** as the score function:

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

- Use a **projection head** $g(\cdot)$ to project features into the space where contrastive learning is applied
- **Generate positive samples through data augmentation**
  $\rightarrow$ random cropping, color distortion, flip, rotations, gaussian blur, etc.




Chen et al., "A simple framework for contrastive learning of visual representations", *ICML* 2020.

# SimCLR: Generating positive samples from data augmentations

(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

# SimCLR: algorithm

Generate a positive pair by sampling data augmentations

**Algorithm 1** SimCLR's main learning algorithm.

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

# SimCLR: algorithm

Generate a positive pair by sampling data augmentations

**Algorithm 1** SimCLR's main learning algorithm.

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

InfoNCE loss: Using all non-positive samples in the batch as x-

# SimCLR: algorithm

Generate a positive pair by sampling data augmentations

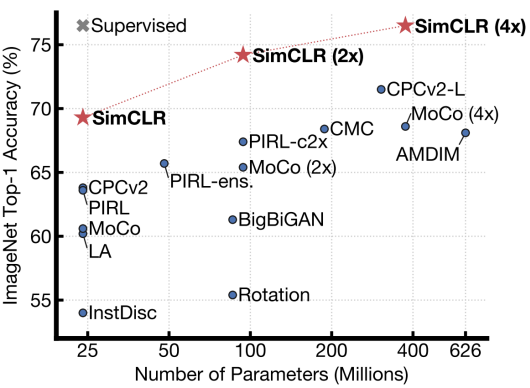
Iterate. Use each of the 2N samples as reference and compute average

InfoNCE loss: Using all non-positive samples in the batch as x-

**Algorithm 1** SimCLR's main learning algorithm.

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

# Evaluation: Training linear classifier on SimCLR features




- SSL training on ImageNet (entire training set)
- Freeze feature encoder, train a linear classifier on top with labeled data.

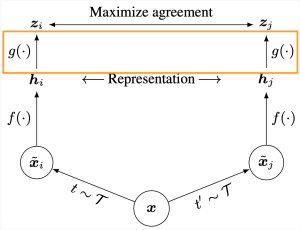
# Evaluation: Semi-supervised learning on SimCLR features

| Method | Architecture | Label fraction (Top 5) 1% | 10% |
|---|---|---|---|
| Supervised baseline | ResNet-50 | 48.4 | 80.4 |
| *Methods using other label-propagation:* | | | |
| Pseudo-label | ResNet-50 | 51.6 | 82.4 |
| VAT+Entropy Min. | ResNet-50 | 47.0 | 83.4 |
| UDA (w. RandAug) | ResNet-50 | - | 88.5 |
| FixMatch (w. RandAug) | ResNet-50 | - | 89.1 |
| S4L (Rot+VAT+En. M.) | ResNet-50 (4×) | - | 91.2 |
| *Methods using representation learning only:* | | | |
| InstDisc | ResNet-50 | 39.2 | 77.4 |
| BigBiGAN | RevNet-50 (4×) | 55.2 | 78.8 |
| PIRL | ResNet-50 | 57.2 | 83.8 |
| CPC v2 | ResNet-161(∗) | 77.9 | 91.2 |
| SimCLR (ours) | ResNet-50 | 75.5 | 87.8 |
| SimCLR (ours) | ResNet-50 (2×) | 83.0 | 91.2 |
| SimCLR (ours) | ResNet-50 (4×) | **85.8** | **92.6** |

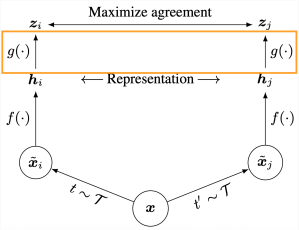
*Table 7.* ImageNet accuracy of models trained with few labels.

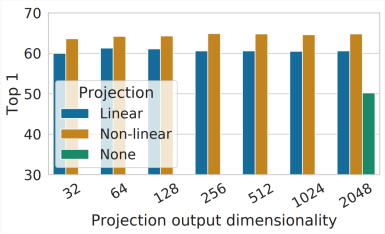
- SSL training on ImageNet (entire training set)
- Fine-tuning with small subset of the training labels

# Some design choices: the projection head



# Some design choices: the projection head




**Projections heads improve the representations**

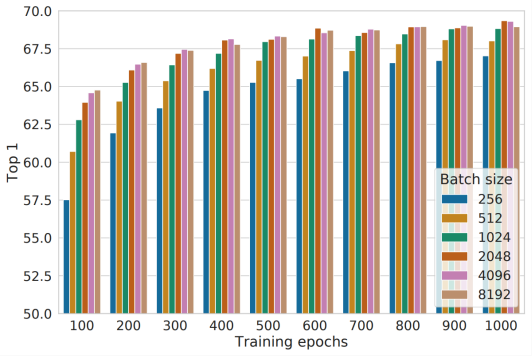
Possible explanation:

→ Contrastive learning discards useful information for downstream tasks in the last layers

→ Representation space $z$ is trained to be invariant to data transformation

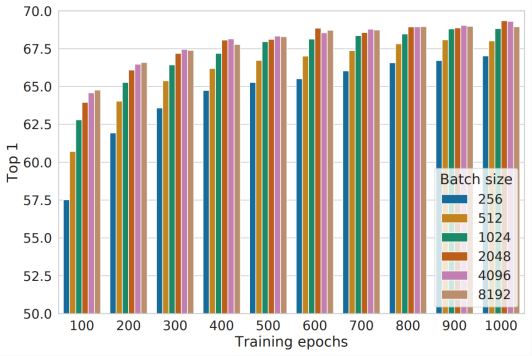
→ The projection head $g$ allows to preserve more information in $h$

# Some design choices: large batch size




**Large training batch size is crucial for SimCLR!**

# Some design choices: large batch size




**Large training batch size is crucial for SimCLR!**

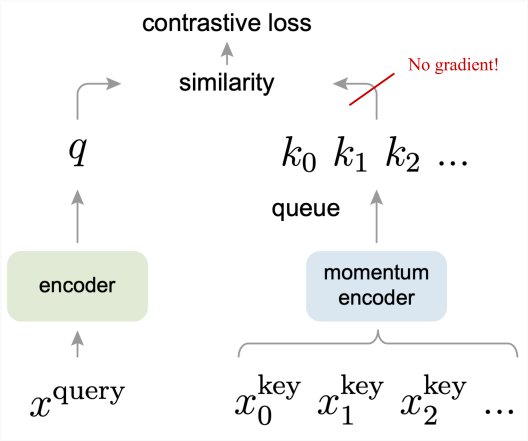
However:

→ Large batch sizes causes large memory usage

→ Requires distributed training on GPUS

$$MI(f(x); f(x^+)) - \log N \geq -\mathcal{L}_{\text{InfoNCE}}$$

# Momentum Contrastive Learning (MoCo)




Key differences to SimCLR:

- Keep a running **queue** of keys (negative samples)
- Compute gradients and update the encoder **only through the queries**
- Decouple mini-batch size with the number of keys → can support **large number of negative samples**
- The encoder is slowly progressing updated with EMA:

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

He et al., "Momentum contrast for unsupervised visual representation learning", *CVPR* 2020.

# MoCo algorithm

**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
```

Generate positive pair with data augmentations

No gradient through the key

Use the running queue of keys as the negative samples

InfoNCE loss

Update f_k with EMA

Update the FIFO negative sample queue

# MoCov2

**Improved Baselines with Momentum Contrastive Learning**

Xinlei Chen Haoqi Fan Ross Girshick Kaiming He
Facebook AI Research (FAIR)

A mix of ideas from SimCLR and MoCo:

→ **From SimCLR:** non-linear projection head and strong data augmentation

→ **From MoCo:** momentum-updated queues that allow training on a large number of negative samples

# MoCo vs. SimCLR vs. MoCov2

| 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** |

Table 1. **Ablation of MoCo baselines**, evaluated by ResNet-50 for (i) ImageNet linear classification, and (ii) fine-tuning VOC object detection (mean of 5 trials). "**MLP**": with an MLP head; "**aug+**": with extra blur augmentation; "**cos**": cosine learning rate schedule.

- Non-linear projection head and strong data augmentation are crucial for contrastive learning

# MoCo vs. SimCLR vs. MoCov2

| 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** |

Table 2. **MoCo *vs.* SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsupervised pre-training. "aug+" in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

- Non-linear projection head and strong data augmentation are crucial for contrastive learning
- Decoupling batch size with negative sample size allows MoCoV2 to outperform simCLR with smaller batches

# MoCo vs. SimCLR vs. MoCov2

| mechanism | batch | memory / GPU | time / 200-ep. |
|---|---|---|---|
| MoCo | 256 | **5.0G** | **53 hrs** |
| end-to-end | 256 | 7.4G | 65 hrs |
| end-to-end | 4096 | 93.0G† | n/a |

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

- Non-linear projection head and strong data augmentation are crucial for contrastive learning
- Decoupling batch size with negative sample size allows MoCoV2 to outperform simCLR with smaller batches
- Much smaller memory footprint!

# Contrastive learning: takeaways

→ A general formulation of contrastive learning:

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

→ **InfoNCE loss:** N-way classification among positive and negative samples

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

→ **Lower bound of the mutual information** between $f(x)$ and $f(x^+)$

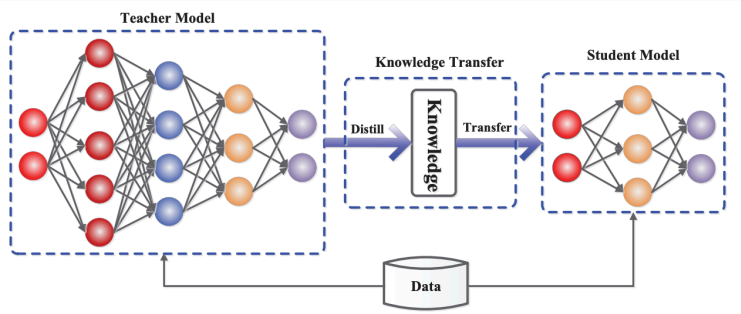
$$MI(f(x); f(x^+)) - \log N \geq -\mathcal{L}_{\text{InfoNCE}}$$

# Self-distillation

# Self-distillation

**What is distillation?**

In machine learning, knowledge distillation or model distillation is the process of **transferring knowledge from a large model to a smaller one**.



# DINO: Self-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

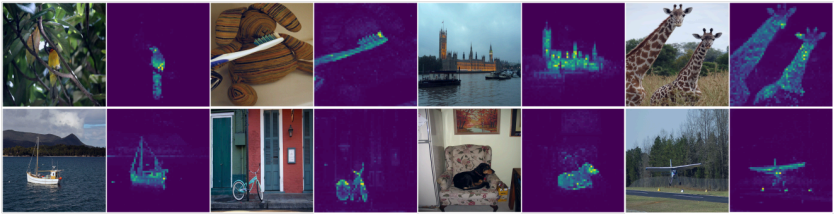
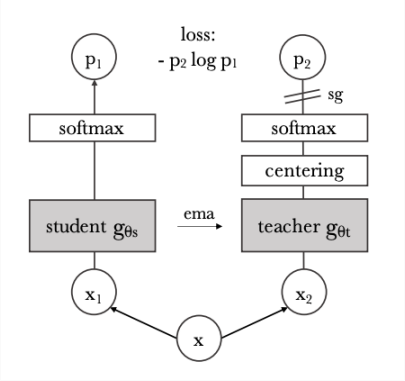


Figure 1: **Self-attention from a Vision Transformer with $8 \times 8$ patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns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

# DINO




- **Teacher-student framework**
  - → Student: trained by gradient descent
  - → Teacher: updated as an EMA of the student (no backprop)
- **View-based learning**
  - → Different augmentations of the same image are fed to student and teacher
- **Student trained to match teacher's predictions**
  - → Outputs are probability distributions
  - → Loss: cross-entropy between student and teacher predictions

Caron et al., "Emerging properties in self-supervised vision transformers", *ICCV* 2021.

# DINO: the algorithm

**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()
```

Positive pair (no negatives!)

The student should match the teacher assignments

Student is trained with backprop

EMA update of teacher weights

# DINOv2

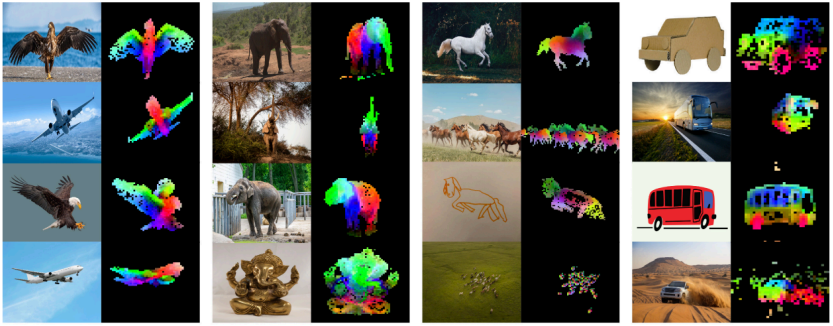
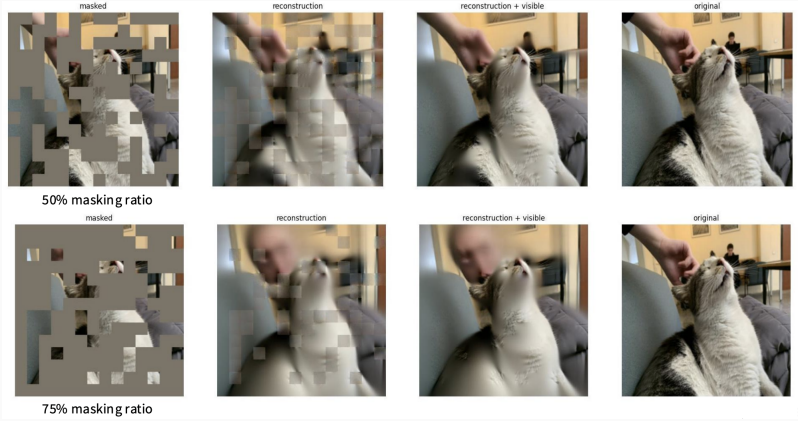



Figure 1: **Visualization of the first PCA components.** We compute a PCA between the patches of the images from the same column (a, b, c and d) and show their first 3 components. Each component is matched to a different color channel. Same parts are matched between related images despite changes of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

# Masked Image Modeling

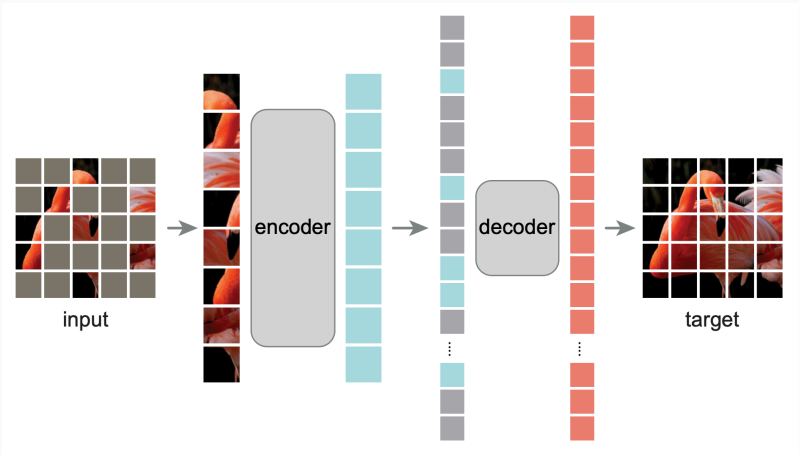
# Masked AutoEncoders (MAE)

Learning to reconstruct the image from a **very corrupted** input



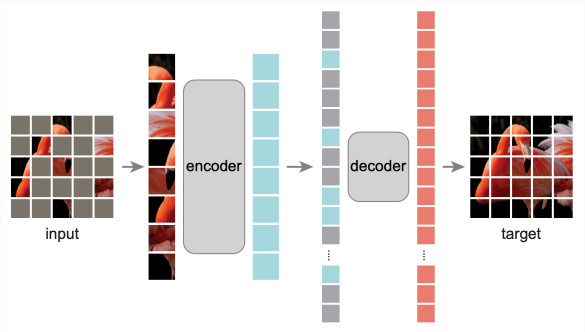

He et al., “Masked Autoencoders Are Scalable Vision Learners”, *CVPR* 2022.

# MAE: architecture




⇝ **ViT-based encoder-decoder architecture**

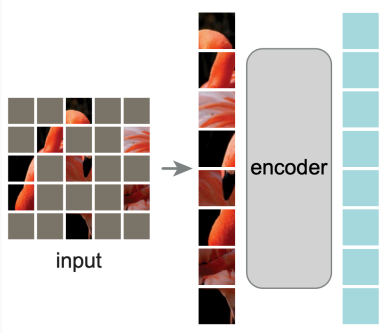
# MAE: how does it work?




- Split image into patches (like in ViT)
- Mask a **very large proportion** (*e.g.* 75%) of these patches
  → Hight masking ratio makes de task challenging and meaningful
- A ViT encodes only unmasked patches
  → less data allows for bigger encoder
- Masked & unmasked patches are decoded using a smaller ViT
- Reconstruction error is measured on masked patches only

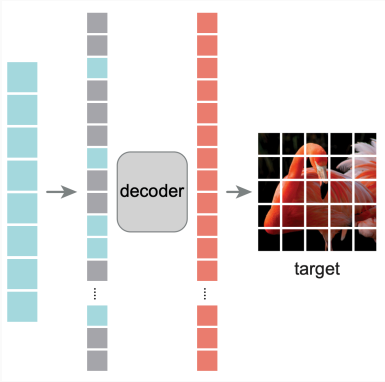
# MAE encoder

- Receives only the visible (unmasked) patches, typically $\sim 25\%$ of the input
- Maps patches to tokens using a linear embedding and positional encodings
- Applies a stack of transformer blocks to produce latent representations
- The reduced sequence length allows using a **large encoder** with limited computational cost



# MAE decoder

- Concatenates the encoder outputs with learned mask tokens
- Applies transformer blocks followed by a linear projection to reconstruct pixel (or patch) values
- Smaller and lighter than the encoder
- Discarded after pretraining; only the encoder is used for downstream tasks



# Reconstruction

- **MSE (Mean Squared Error) in the pixel space**, between input image and reconstructed image.
- This loss is computed on **masked patches only**:

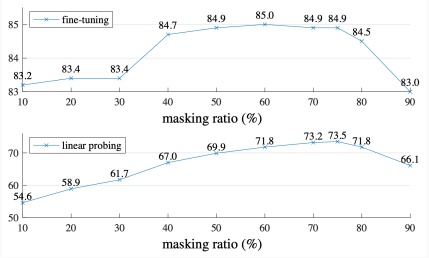
$$\mathcal{L}_{\text{MAE}} = \frac{1}{M} \sum_{i \in \mathcal{M}} ||x_i - \hat{x}_i||^2,$$

with $x_i$ the $i$-ème patch, $\hat{x}_i$ its reconstructed version, $\mathbf{M}$ the set of patches and $M = |\mathcal{M}|$

# Design choices and ablation studies

**So many modeling and hyperparameters choices!**

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



# Design choices and ablation studies

**So many modeling and hyperparameters choices!**

| blocks | ft | lin |
|---|---|---|
| 1 | 84.8 | 65.5 |
| 2 | **84.9** | 70.0 |
| 4 | **84.9** | 71.9 |
| 8 | **84.9** | **73.5** |
| 12 | 84.4 | 73.3 |

(a) **Decoder depth**. A deep decoder can improve linear probing accuracy.

| dim | ft | lin |
|---|---|---|
| 128 | **84.9** | 69.1 |
| 256 | 84.8 | 71.3 |
| 512 | **84.9** | **73.5** |
| 768 | 84.4 | 73.1 |
| 1024 | 84.3 | 73.1 |

(b) **Decoder width**. The decoder can be narrower than the encoder (1024-d).

| case | ft | lin | FLOPs |
|---|---|---|---|
| encoder w/ `[M]` | 84.2 | 59.6 | 3.3× |
| encoder w/o `[M]` | **84.9** | **73.5** | **1**× |

(c) **Mask token**. An encoder without mask tokens is more accurate and faster (Table 2).

| case | ft | lin |
|---|---|---|
| pixel (w/o norm) | 84.9 | 73.5 |
| pixel (w/ norm) | **85.4** | **73.9** |
| PCA | 84.6 | 72.3 |
| dVAE token | 85.3 | 71.6 |

(d) **Reconstruction target**. Pixels as reconstruction targets are effective.

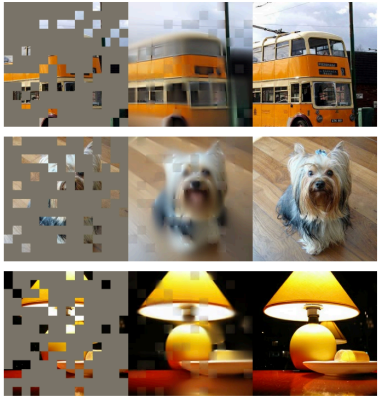
| case | ft | lin |
|---|---|---|
| none | 84.0 | 65.7 |
| crop, fixed size | 84.7 | 73.1 |
| crop, rand size | **84.9** | **73.5** |
| crop + color jit | 84.3 | 71.9 |

(e) **Data augmentation**. Our MAE works with minimal or no augmentation.

| 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 |

(f) **Mask sampling**. Random sampling works the best. See Figure 6 for visualizations.

# Results



Reconstruction results

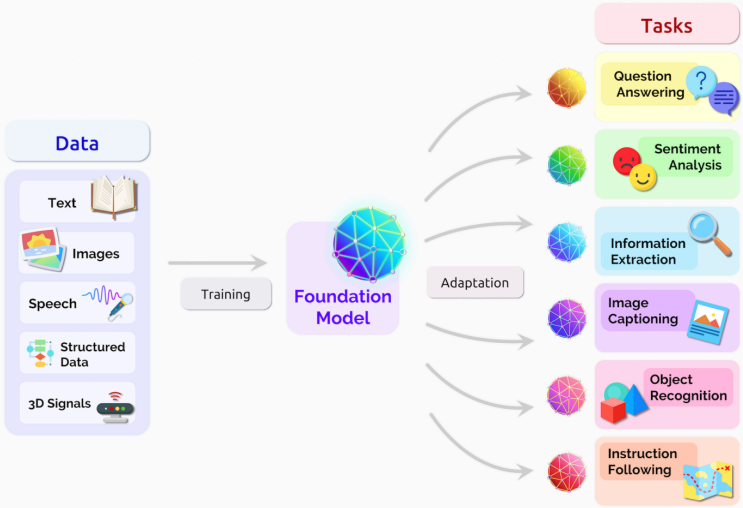
| method | pre-train data | ViT-B | ViT-L | ViT-H | ViT-$H_{448}$ |
|---|---|---|---|---|---|
| scratch, our impl. | - | 82.3 | 82.6 | 83.1 | - |
| DINO [5] | IN1K | 82.8 | - | - | - |
| MoCo v3 [9] | IN1K | 83.2 | 84.1 | - | - |
| BEiT [2] | IN1K+DALLE | 83.2 | 85.2 | - | - |
| MAE | IN1K | 83.6 | 85.9 | 86.9 | **87.8** |

Table 3. **Comparisons with previous results on ImageNet-**

Comparisons with other SSL methods

# Foundation Models

# What are *Foundation Models*?




Bommasani et al., “On the opportunities and risks of foundation models”, 2021.

# Foundation Models

**Goal:** **One model to rule them all!**

# Foundation Models

**Goal: One model to rule them all!**

A **foundation model** is any model that is trained on **broad data** (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a **wide range of downstream tasks**.

# Foundation Models

**Goal: One model to rule them all!**

A **foundation model** is any model that is trained on **broad data** (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a **wide range of downstream tasks**.

**Key ingredients:**

- → Large scale training data
- → Self-supervised pre-training
- → Transfer learning
- → (Ideally) can integrate **multiple modalities**

# Foundation Models

**Goal: One model to rule them all!**

A **foundation model** is any model that is trained on **broad data** (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a **wide range of downstream tasks**.

**Key ingredients:**

- → Large scale training data
- → Self-supervised pre-training
- → Transfer learning
- → (Ideally) can integrate **multiple modalities** (*e.g.* images & text)

# What are Vision & Language models?

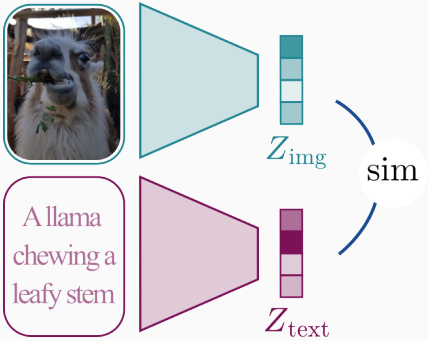

⇝ **Models that jointly learn from images and text**

# CLIP

**Goal:** Learn image and text embeddings

# CLIP

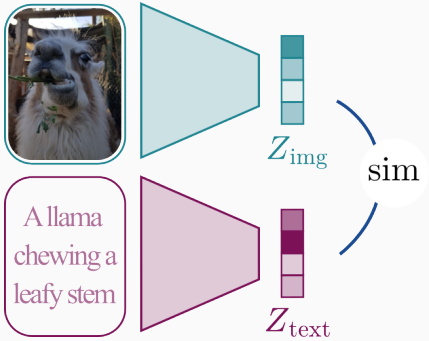
**Goal: Learn image and text embeddings**




Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP

**Goal: Learn image and text embeddings**

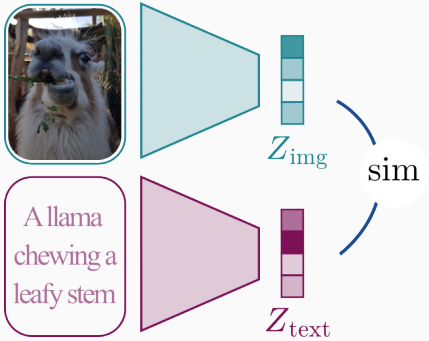



- Matching images and texts are close, and non-matching pairs are pushed apart.

Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP

**Goal: Learn image and text embeddings**



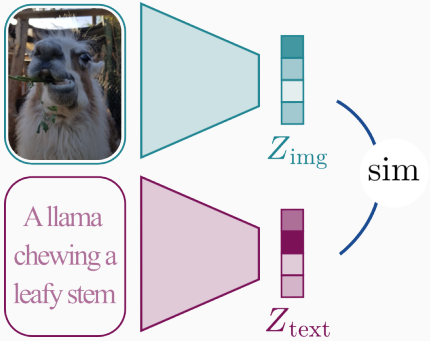

- Matching images and texts are close, and non-matching pairs are pushed apart.
- Uses *contrastive learning* (InfoNCE loss)

$$\mathcal{L}_{\text{InfoNCE}} = \mathbb{E}_{\substack{z, z'_{\text{pos}} \sim p(Z, Z') \\ z'_{\text{neg}} \sim p(Z')}} \left[ \log \frac{\exp \text{sim}(z, z'_{\text{pos}})}{\sum_{z'_{\text{neg}}} \exp \text{sim}(z, z'_{\text{neg}})} \right]$$

Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP

**Goal: Learn image and text embeddings**




- Matching images and texts are close, and non-matching pairs are pushed apart.
- Uses *contrastive learning* (InfoNCE loss)

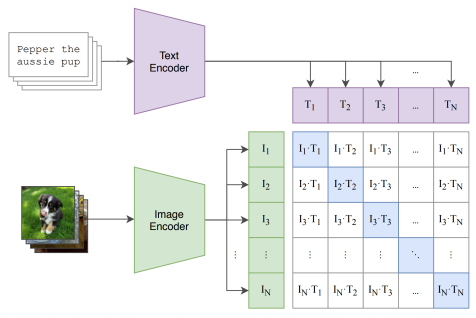
$$\mathcal{L}_{\text{InfoNCE}} = \mathbb{E}_{\substack{z, z'_{\text{pos}} \sim p(Z, Z') \\ z'_{\text{neg}} \sim p(Z')}} \left[ \log \frac{\exp \text{sim}(z, z'_{\text{pos}})}{\sum_{z'_{\text{neg}}} \exp \text{sim}(z, z'_{\text{neg}})} \right]$$

⇝ CLIP has the ability to match images with natural language concepts.

Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP

## Goal: Learn image and text embeddings

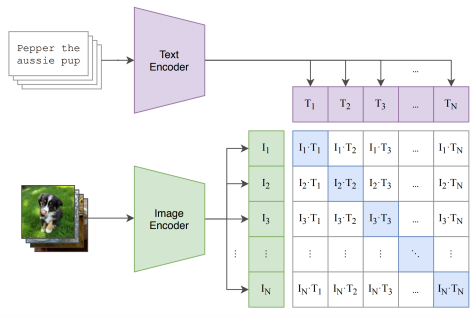



How it works?

Radford et al., “Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP

## Goal: Learn image and text embeddings



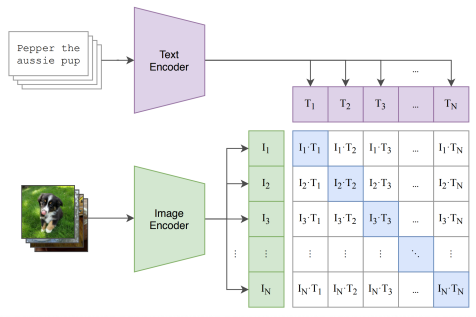

How it works?

- Take a batch of image–text pairs and encode them

Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP

## Goal: Learn image and text embeddings



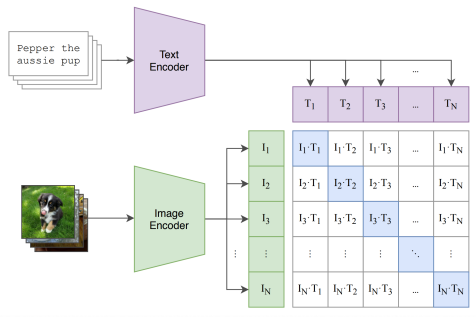

How it works?

- Take a batch of image–text pairs and encode them
- Compute a similarity matrix between all image–text combinations in the batch.

Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP

## Goal: Learn image and text embeddings



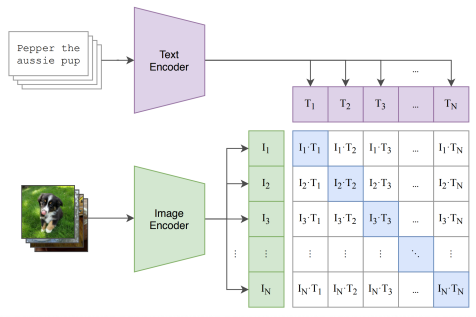

How it works?

- Take a batch of image–text pairs and encode them
- Compute a similarity matrix between all image–text combinations in the batch.
- The true pair should have the highest similarity.

Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP

## Goal: Learn image and text embeddings




How it works?

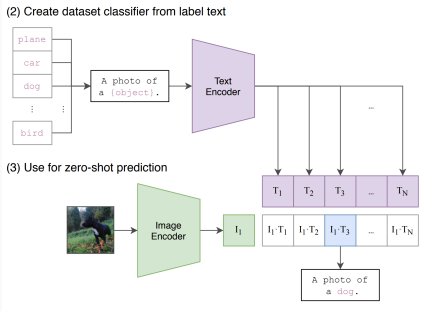
- Take a batch of image–text pairs and encode them
- Compute a similarity matrix between all image–text combinations in the batch.
- The true pair should have the highest similarity.
- Optimize a symmetric $\mathcal{L}_{\text{InfoNCE}}$

Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP was a change of paradigm

CLIP enabled:

- **Zero-shot classification** on *any* label set
  → By using prompts like "A photo of a `[class]`"
- Open-vocabulary recognition
  → Detect or segment categories never seen during training.
- Text-image and image-text retrieval

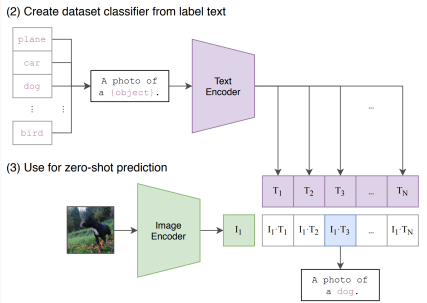



Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP was a change of paradigm

CLIP enabled:

- **Zero-shot classification** on *any* label set
  → By using prompts like "A photo of a `[class]`"
- Open-vocabulary recognition
  → Detect or segment categories never seen during training.
- Text-image and image-text retrieval




⇝ **CLIP enabled flexible, language-driven image understanding**

Radford et al., "Learning transferable visual models from natural language supervision, *ICML* 2021.

# CLIP results

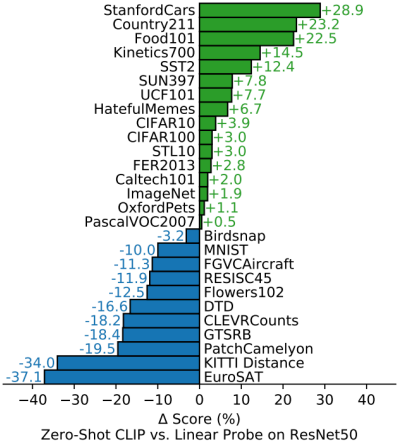



*Figure 5.* **Zero-shot CLIP is competitive with a fully supervised baseline.** Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

# CLIP results

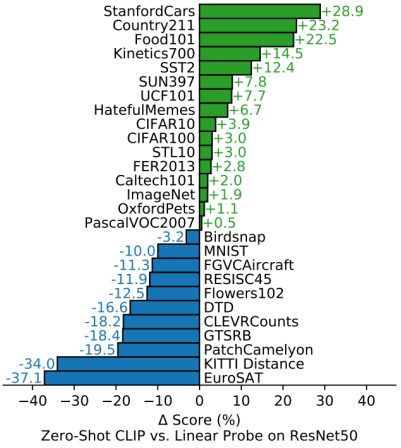



*Figure 5.* **Zero-shot CLIP is competitive with a fully supervised baseline.** Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

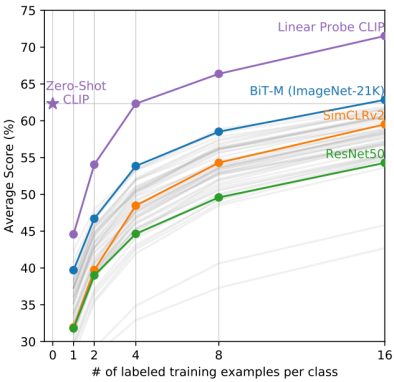
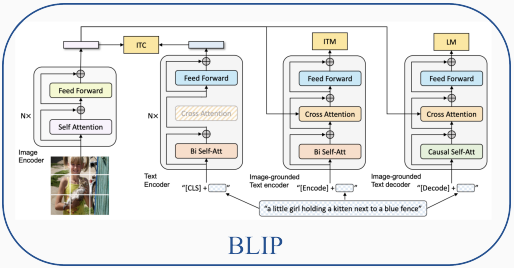
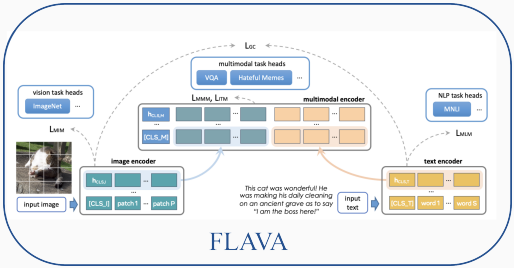



*Figure 6.* **Zero-shot CLIP outperforms few-shot linear probes.** Zero-shot CLIP matches the average performance of a 4-shot linear classifier trained on the same feature space and nearly matches the best results of a 16-shot linear classifier across publicly available models. For both BiT-M and SimCLRv2, the best performing model is highlighted. Light gray lines are other models in the eval suite. The 20 datasets with at least 16 examples per class were used in this analysis.

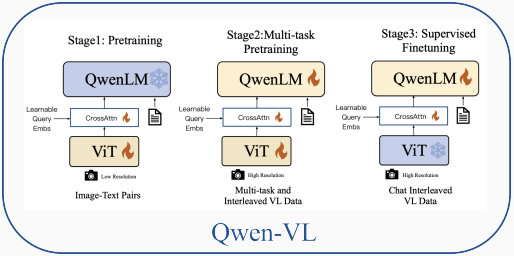
# Many flavors of VLMs




BLIP




FLAVA




Qwen-VL

- Different training strategies:
  - → Contrastive
  - → Masked modeling
  - → Generative
- Many applications!!

# Final words

- **Self-supervised learning changed how we build models**

# Final words

- **Self-supervised learning changed how we build models**
  - → We no longer train models for a single task, but to learn representations from data itself
  - → Pretraining becomes the central step; downstream tasks become adaptations
- **Foundation models are the natural next step**

# Final words

- **Self-supervised learning changed how we build models**
  - → We no longer train models for a single task, but to learn representations from data itself
  - → Pretraining becomes the central step; downstream tasks become adaptations
- **Foundation models are the natural next step**
  - → Large-scale self-supervised training on diverse data
  - → One model → many tasks, domains, and modalities
  - → Vision, language, and multimodal models now share a common paradigm

# Final words

- **Self-supervised learning changed how we build models**
  - → We no longer train models for a single task, but to learn representations from data itself
  - → Pretraining becomes the central step; downstream tasks become adaptations
- **Foundation models are the natural next step**
  - → Large-scale self-supervised training on diverse data
  - → One model → many tasks, domains, and modalities
  - → Vision, language, and multimodal models now share a common paradigm

⇝ **The goal is no longer to solve one task, but to learn a reusable foundation.**