

# Lecture 13: Self-Supervised Learning

# Last Week: Lots of Computer Vision Tasks

## Classification



CAT

No spatial extent

## Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

## Object Detection



DOG, DOG, CAT

Multiple Object

## Instance Segmentation

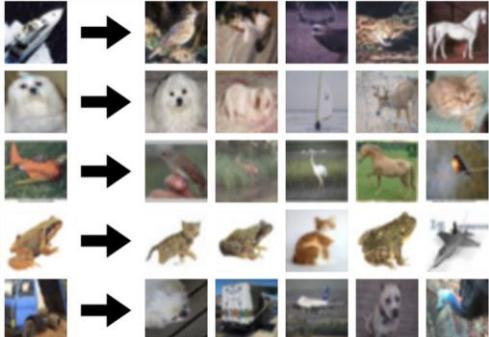


DOG, DOG, CAT

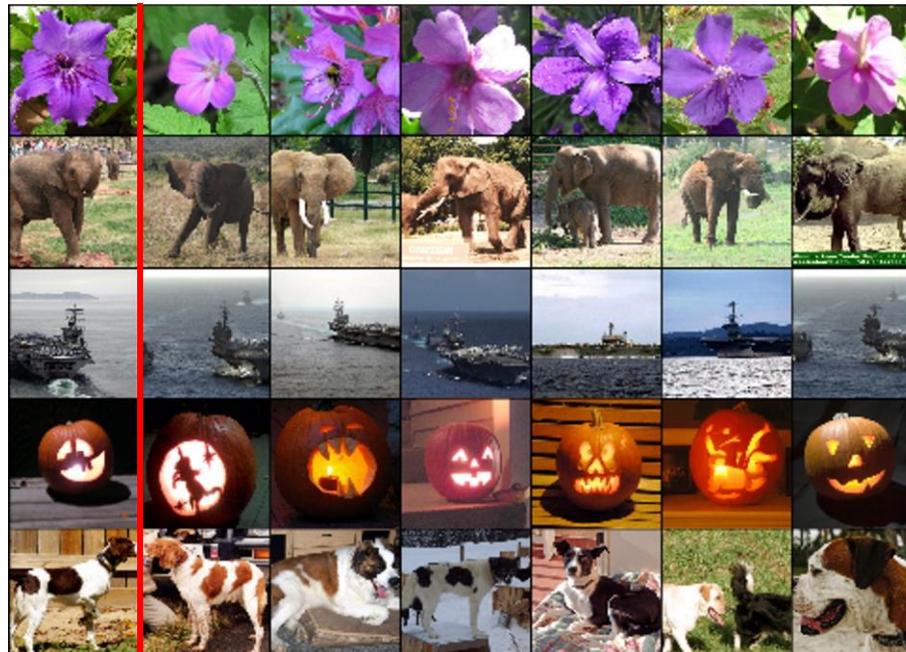
This image is CC0 public domain

# Last Week: Visualizing and Understanding

**Recall:** Nearest neighbors  
in pixel space



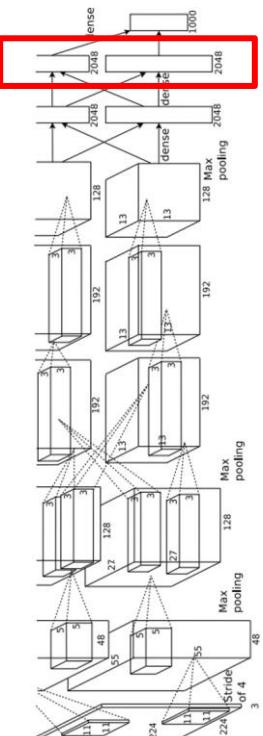
Test image    L2 Nearest neighbors in feature space



Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

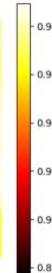
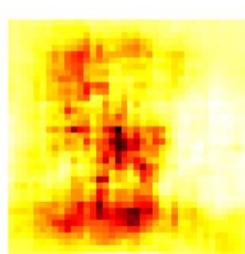
Figures reproduced with permission.

4096-dim vector

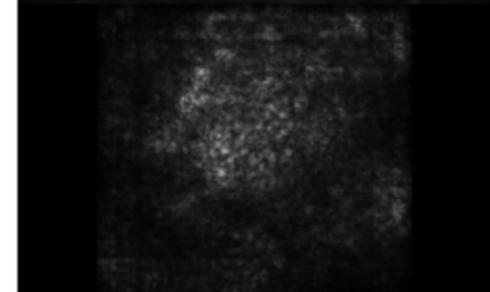
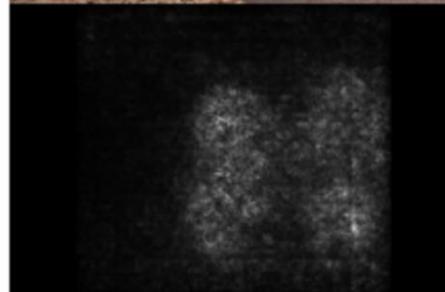
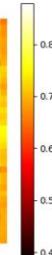
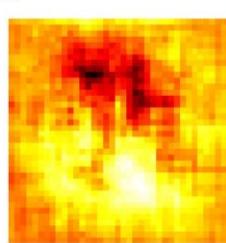


# Last Week: Visualizing and Understanding

schooner



African elephant, *Loxodonta africana*



# Last Week: Visualizing and Understanding

Content Image



+

Style Image



[Starry Night](#) by Van Gogh is in the public domain

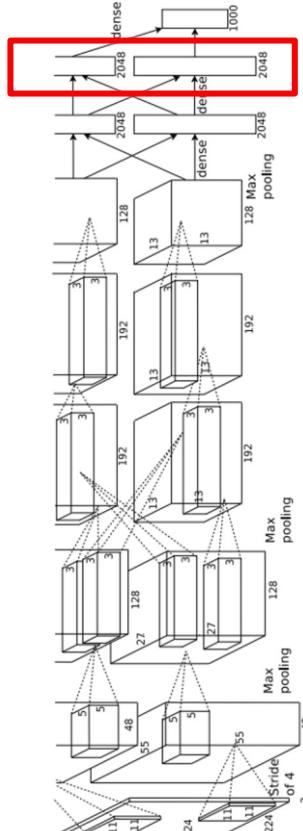
=



[This image](#) copyright Justin Johnson, 2015. Reproduced with permission.

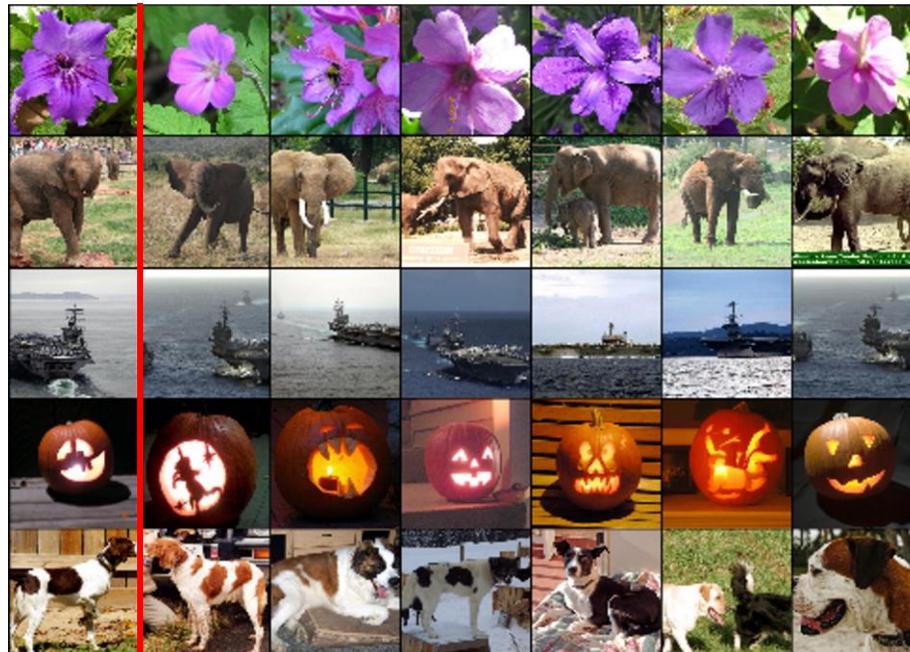
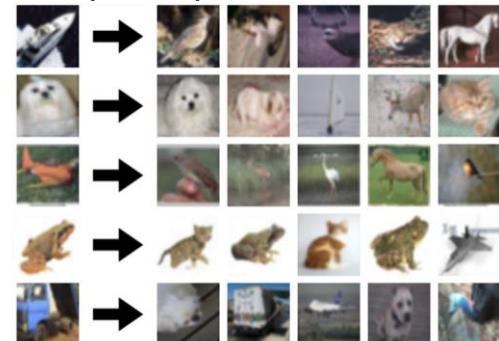
# Learned Representations

4096-dim vector



## Test image L2 Nearest neighbors in feature space

**Recall:** Nearest neighbors  
in pixel space



Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks". NIPS 2012.

Figures reproduced with permission.

# Self-supervised Learning

- Both aim to learn from data without manual label annotation.
- Self-supervised learning methods solve “pretext” tasks that produce **good features** for downstream tasks.
  - Learn with supervised learning objectives, e.g., classification, regression.
  - Labels of these pretext tasks are generated *automatically*

# Self-supervised pretext tasks

Example: learn to predict image transformations / complete corrupted images

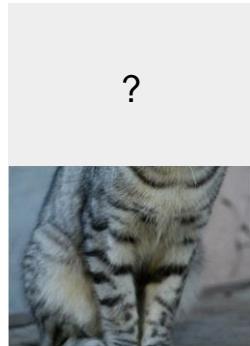
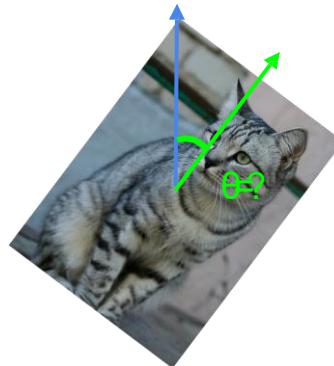


image completion



rotation prediction



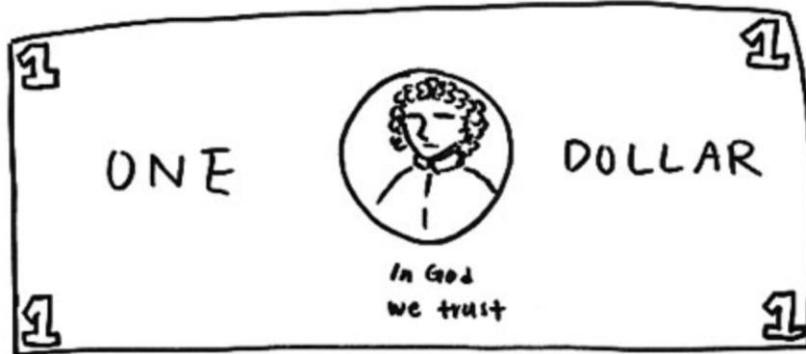
“jigsaw puzzle”



colorization

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

# Generative vs. Self-supervised Learning



Left: Drawing of a dollar bill from memory. Right: Drawing subsequently made with a dollar bill present. Image source: [Epstein, 2016](#)

Learning to generate pixel-level details is often unnecessary; learn high-level semantic features with pretext tasks instead

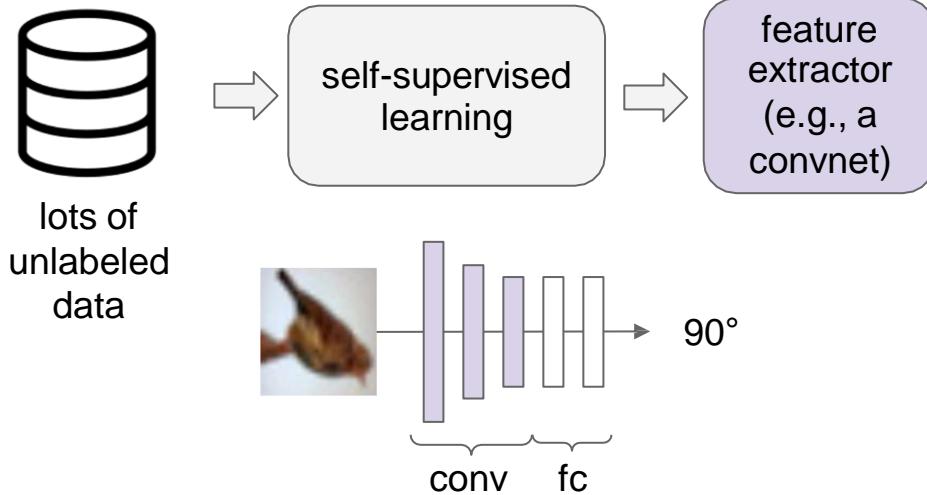
Source: [Anand, 2020](#)

# How to evaluate a self-supervised learning method?

We usually don't care about the performance of the self-supervised learning task, e.g., we don't care if the model learns to predict image rotation perfectly.

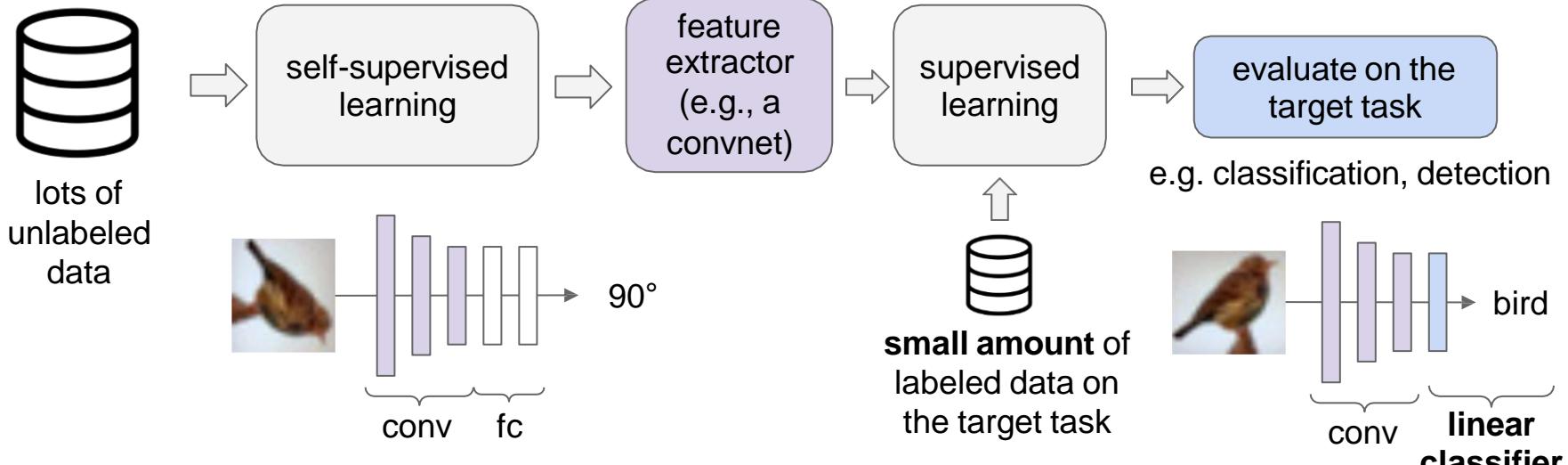
Evaluate the learned feature encoders on downstream *target tasks*

# How to evaluate a self-supervised learning method?



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

# How to evaluate a self-supervised learning method?



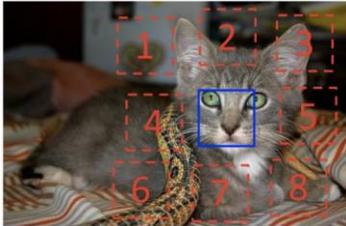
1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

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

# Broader picture

Today's lecture

computer vision



Doersch et al., 2015

robot / reinforcement learning



Dense Object Net (Florence  
and Manuelli et al., 2018)

language modeling

GPT-4 Technical Report

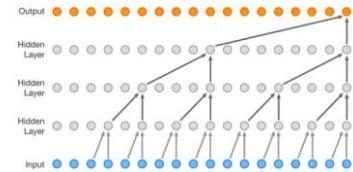
OpenAI\*

Abstract

We report the development of GPT-4, a large-scale, multimodal model which can accept image and text inputs and produce text outputs. While less capable than humans in many real-world scenarios, GPT-4 exhibits human-level performance on various professional and academic benchmarks, including passing a simulated bar exam with a score around the top 10% of test takers. GPT-4 is a Transformer-based model pre-trained to predict the next token in a document. The post-training alignment process results in improved performance on measures of factuality and adherence to desired behavior. A core component of this project was developing infrastructure and optimization methods that behave predictably across a wide range of scales. This allowed us to accurately predict some aspects of GPT-4's performance based on models trained with no more than 1/1,000th the compute of GPT-4.

GPT-4 (OpenAI 2023)

speech synthesis



Wavenet (van den Oord et  
al., 2016)

• • •

# Today's Agenda

## **Pretext tasks from image transformations**

- Rotation, inpainting, rearrangement, coloring

## **Contrastive representation learning**

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

# Today's Agenda

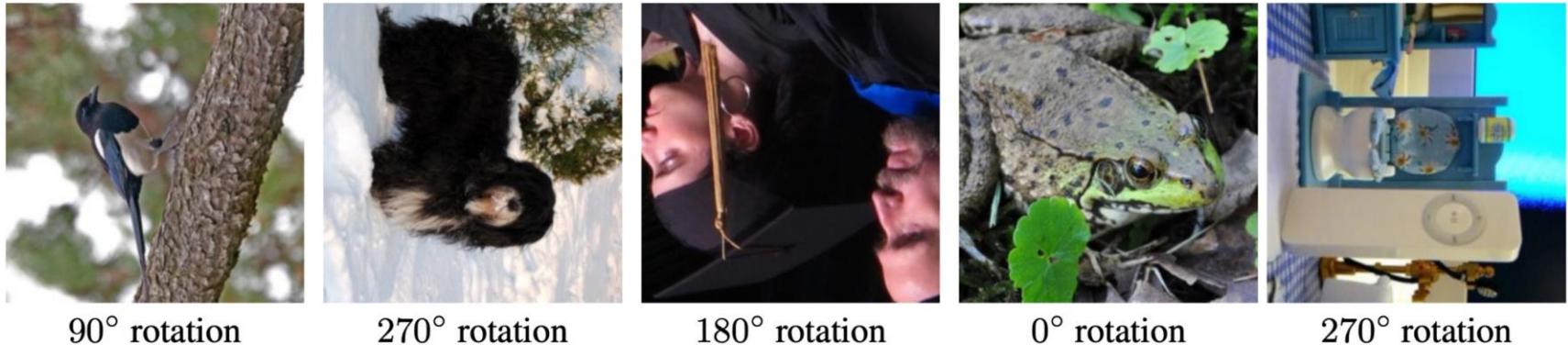
## **Pretext tasks from image transformations**

- Rotation, inpainting, rearrangement, coloring

## **Contrastive representation learning**

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

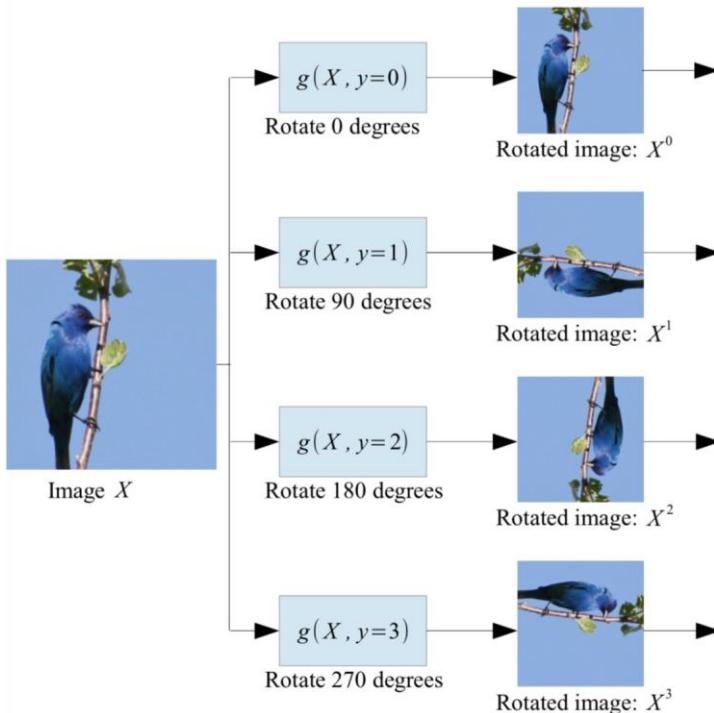
# Pretext task: predict rotations



**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.

(Image source: [Gidaris et al. 2018](#))

# Pretext task: predict rotations

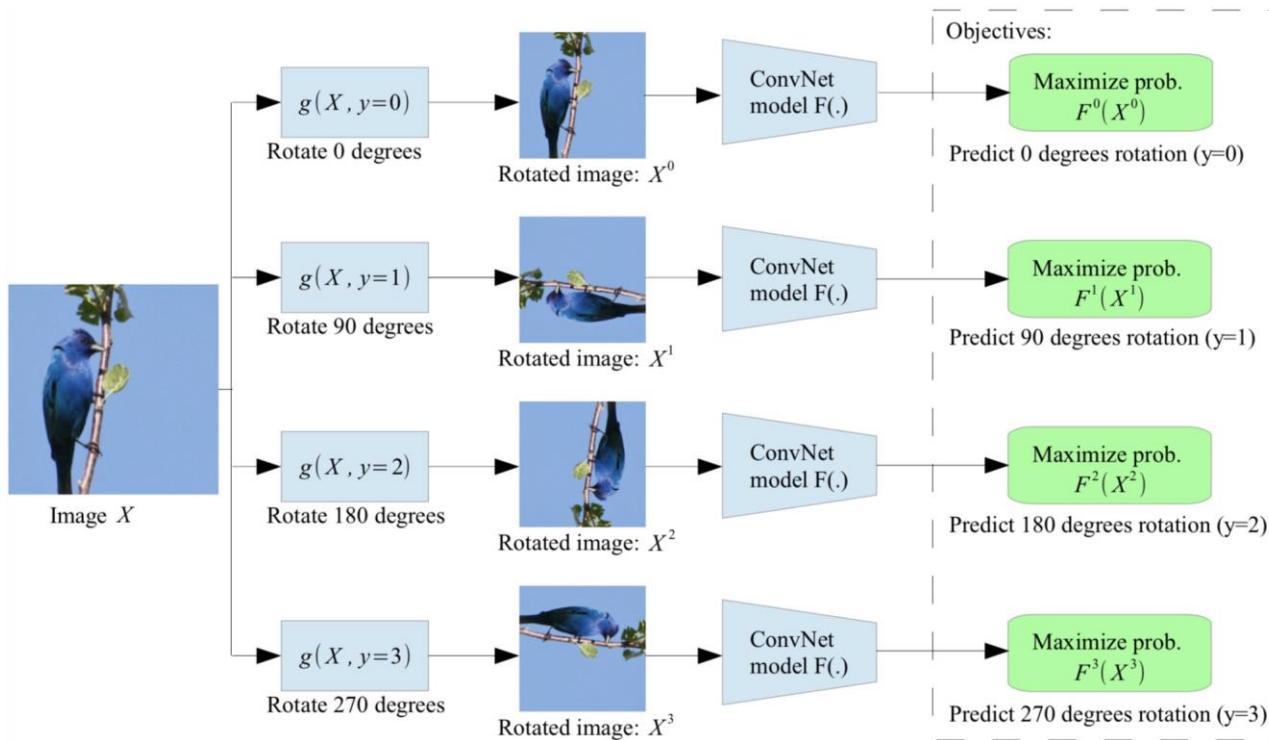


Self-supervised learning by rotating the entire input images.

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

(Image source: [Gidaris et al. 2018](#))

# Pretext task: predict rotations

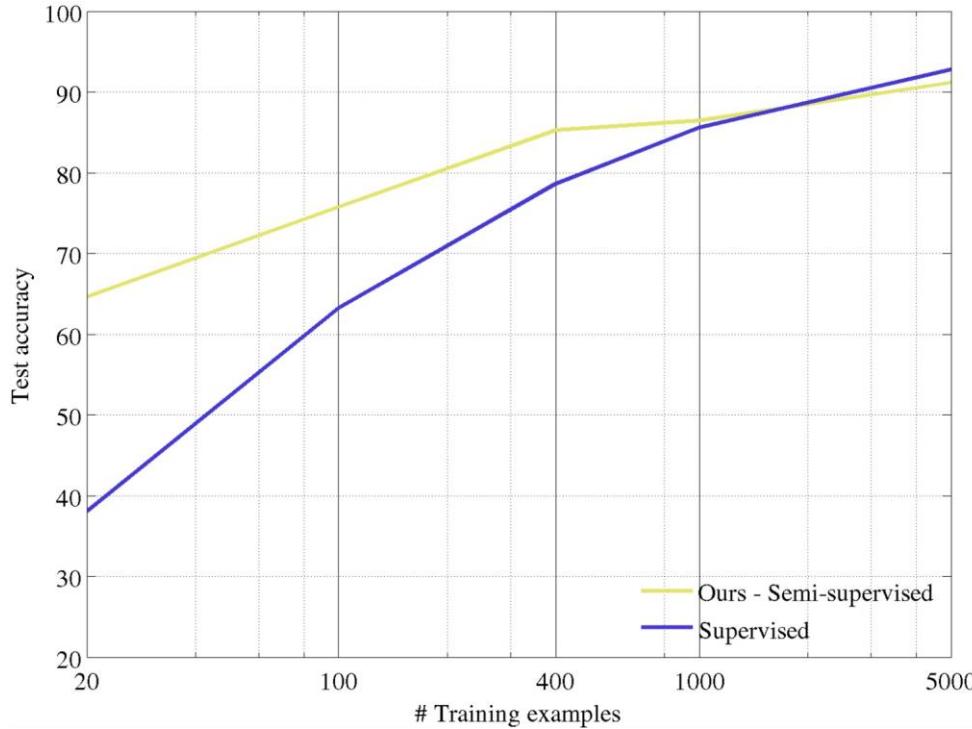


Self-supervised learning by rotating the entire input images.

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

(Image source: [Gidaris et al. 2018](#))

# Evaluation on semi-supervised learning



Self-supervised learning on  
**CIFAR10** (entire training set).

Freeze conv1 + conv2  
Learn **conv3 + linear** layers  
with subset of labeled  
CIFAR10 data (classification).

(Image source: [Gidaris et al. 2018](#))

# Transfer learned features to supervised learning

	Classification (%mAP)	Detection (%mAP)	Segmentation (%mIoU)
Trained layers	fc6-8	all	all
ImageNet labels	78.9	79.9	56.8
Random		53.3	43.4
Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6
Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9
Context Encoders (Pathak et al., 2016b)	34.6	56.5	44.5
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
BIGAN (Donahue et al., 2016)	52.3	60.1	46.9
Jigsaw Puzzles (Noroozi & Favaro, 2016)	-	67.6	53.2
NAT (Bojanowski & Joulin, 2017)	56.7	65.3	49.4
Split-Brain (Zhang et al., 2016b)	63.0	67.1	46.7
ColorProxy (Larsson et al., 2017)		65.9	
Counting (Noroozi et al., 2017)	-	67.7	51.4
(Ours) RotNet	<b>70.87</b>	<b>72.97</b>	<b>54.4</b>
			<b>39.1</b>

Self-supervised learning with rotation prediction

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**.

source: [Gidaris et al. 2018](#)

# Visualize learned visual attentions



Conv1  $27 \times 27$    Conv3  $13 \times 13$    Conv5  $6 \times 6$

(a) Attention maps of supervised model

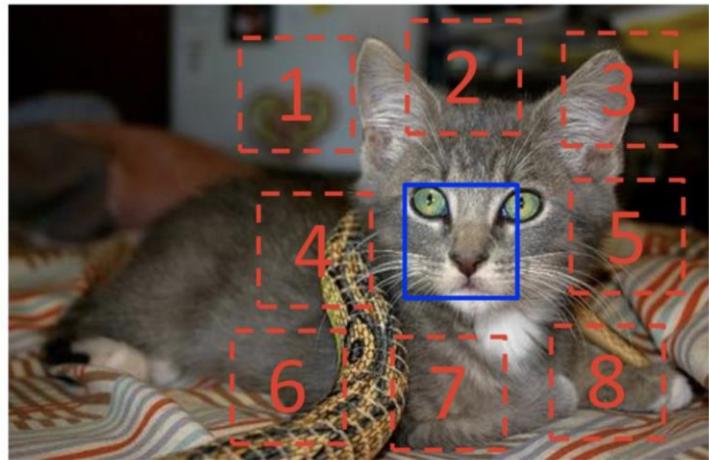


Conv1  $27 \times 27$    Conv3  $13 \times 13$    Conv5  $6 \times 6$

(b) Attention maps of our self-supervised model

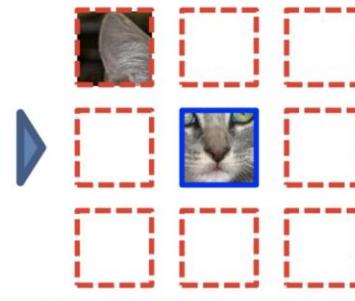
(Image source: [Gidaris et al. 2018](#))

# Pretext task: predict relative patch locations



$$X = (\text{[cat eye]}, \text{[cat ear]}); Y = 3$$

Example:



Question 1:

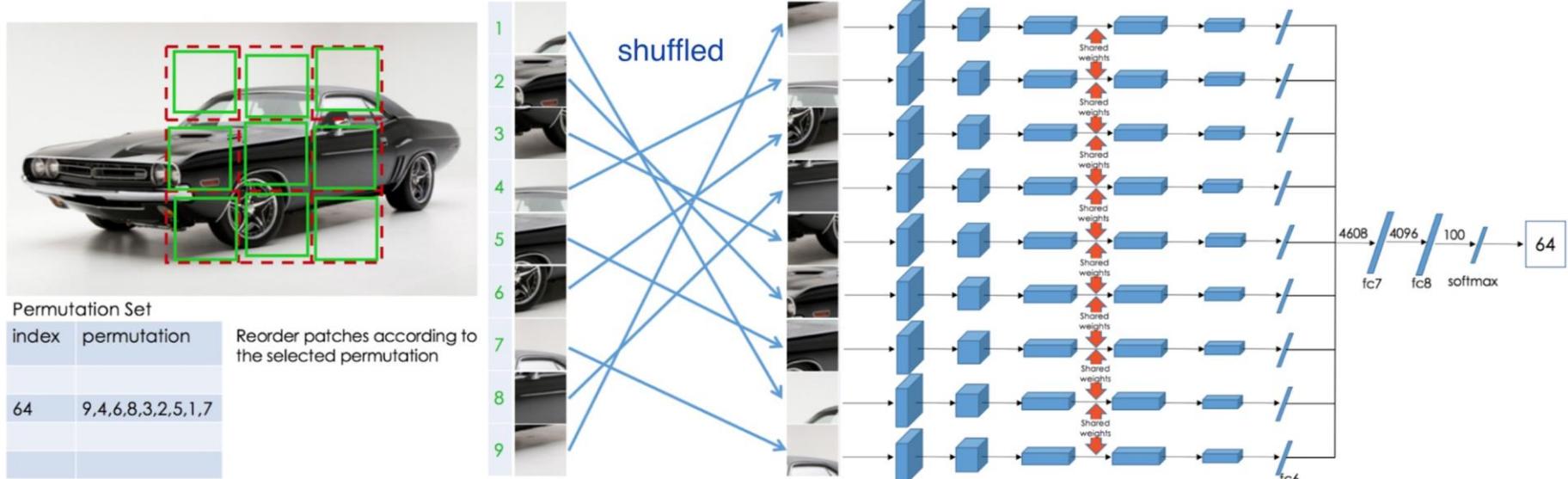


Question 2:



(Image source: [Doersch et al., 2015](#))

# Pretext task: solving “jigsaw puzzles”



(Image source: [Noroozi & Favaro, 2016](#))

# Transfer learned features to supervised learning

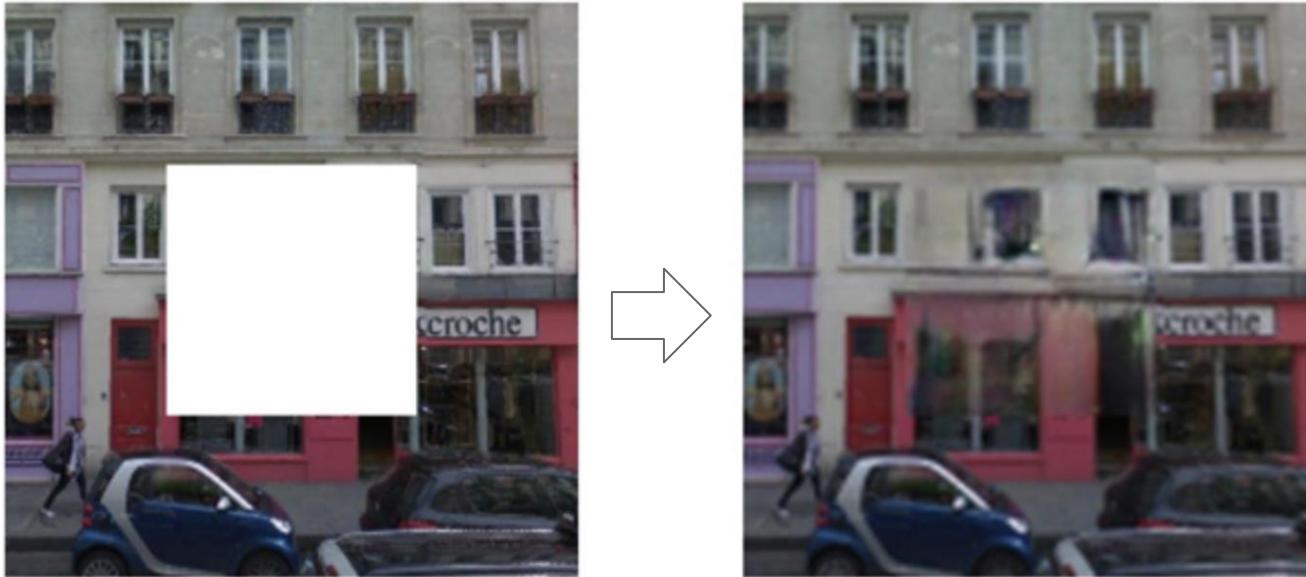
Table 1: Results on PASCAL VOC 2007 Detection and Classification. The results of the other methods are taken from Pathak *et al.* [30].

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

“Ours” is feature learned from solving image Jigsaw puzzles (Noroozi & Favaro, 2016). Doersch et al. is the method with relative patch location

(source: [Noroozi & Favaro, 2016](#))

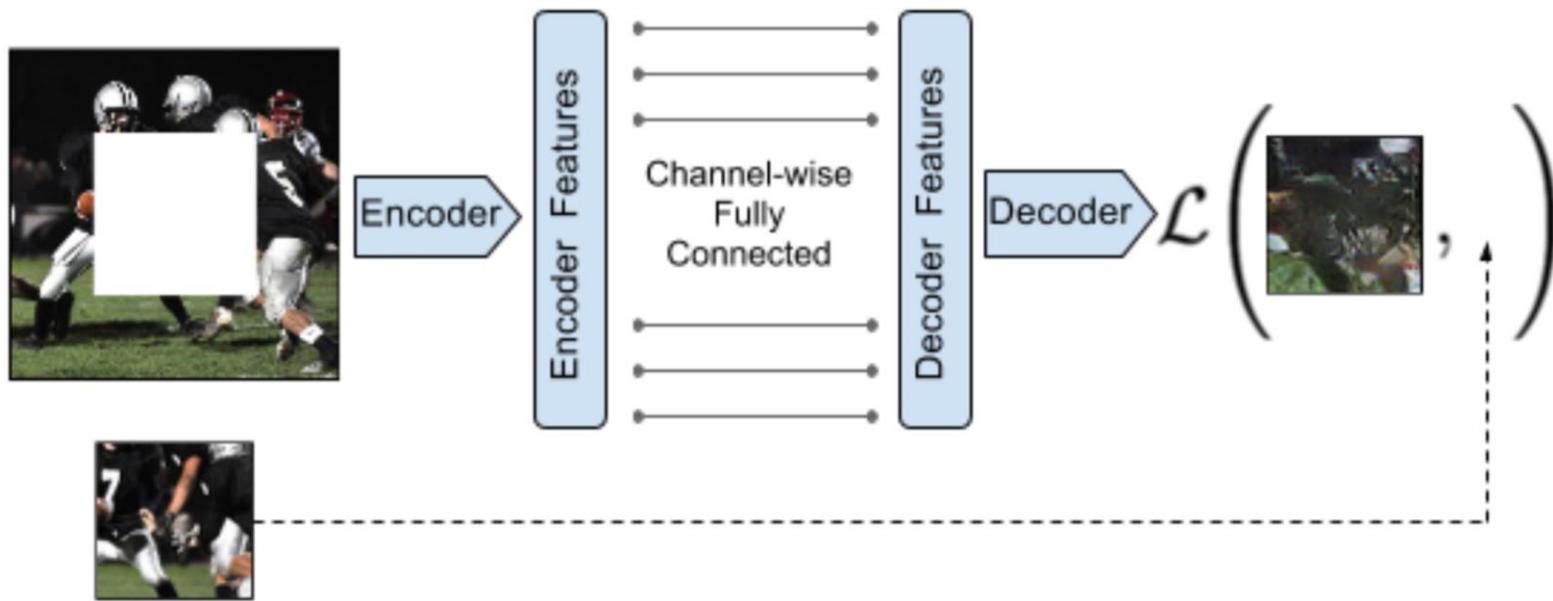
# Pretext task: predict missing pixels (inpainting)



*Context Encoders: Feature Learning by Inpainting* (Pathak et al., 2016)

Source: [Pathak et al., 2016](#)

# Learning to inpaint by reconstruction



Learning to reconstruct the missing pixels

Source: [Pathak et al., 2016](#)

# Inpainting evaluation



Input (context)



reconstruction

Source: [Pathak et al., 2016](#)

# Learning to inpaint by reconstruction

Loss = reconstruction + adversarial learning

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

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

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

Adversarial loss between “real” images and *inpainted images*

# Inpainting evaluation



Input (context)

reconstruction

adversarial

recon + adv

Source: [Pathak et al., 2016](#)

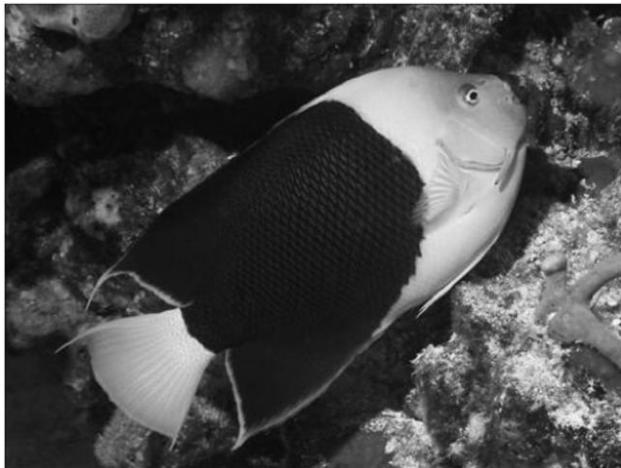
# Transfer learned features to supervised learning

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 <i>et al.</i> [1]	egomotion	10 hours	52.9%	41.8%	-
Wang <i>et al.</i> [39]	motion	1 week	58.7%	47.4%	-
Doersch <i>et al.</i> [7]	relative context	4 weeks	55.3%	46.6%	-
Ours	context	14 hours	56.5%	44.5%	30.0%

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

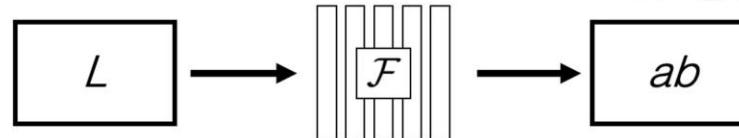
Source: [Pathak et al., 2016](#)

# Pretext task: image coloring



Grayscale image:  $L$  channel

$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$

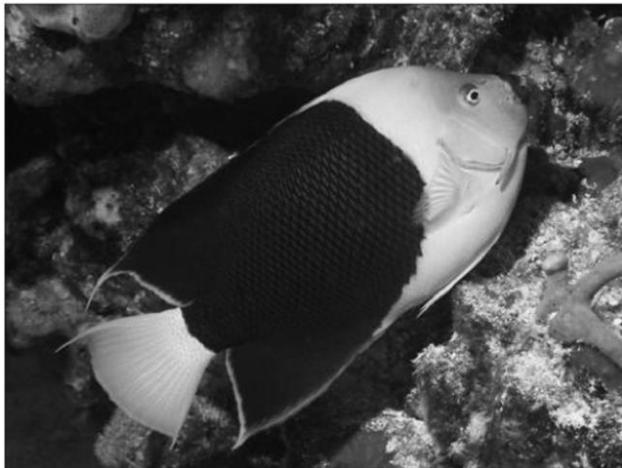


Color information:  $ab$  channels

$$\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 2}$$

Source: Richard Zhang / Phillip Isola

# Pretext task: image coloring

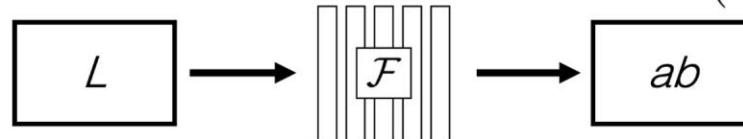


Grayscale image:  $L$  channel

$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$



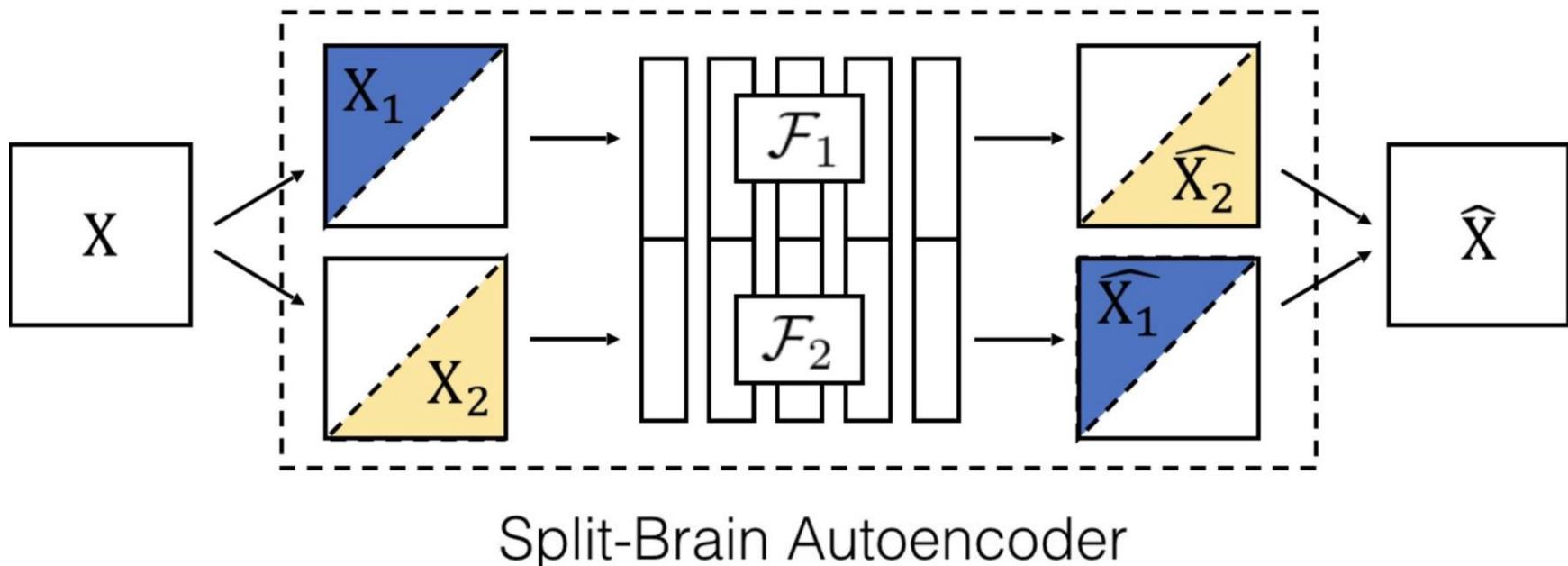
Concatenate  $(L, ab)$  channels  
 $(\mathbf{X}, \hat{\mathbf{Y}})$



Source: Richard Zhang / Phillip Isola

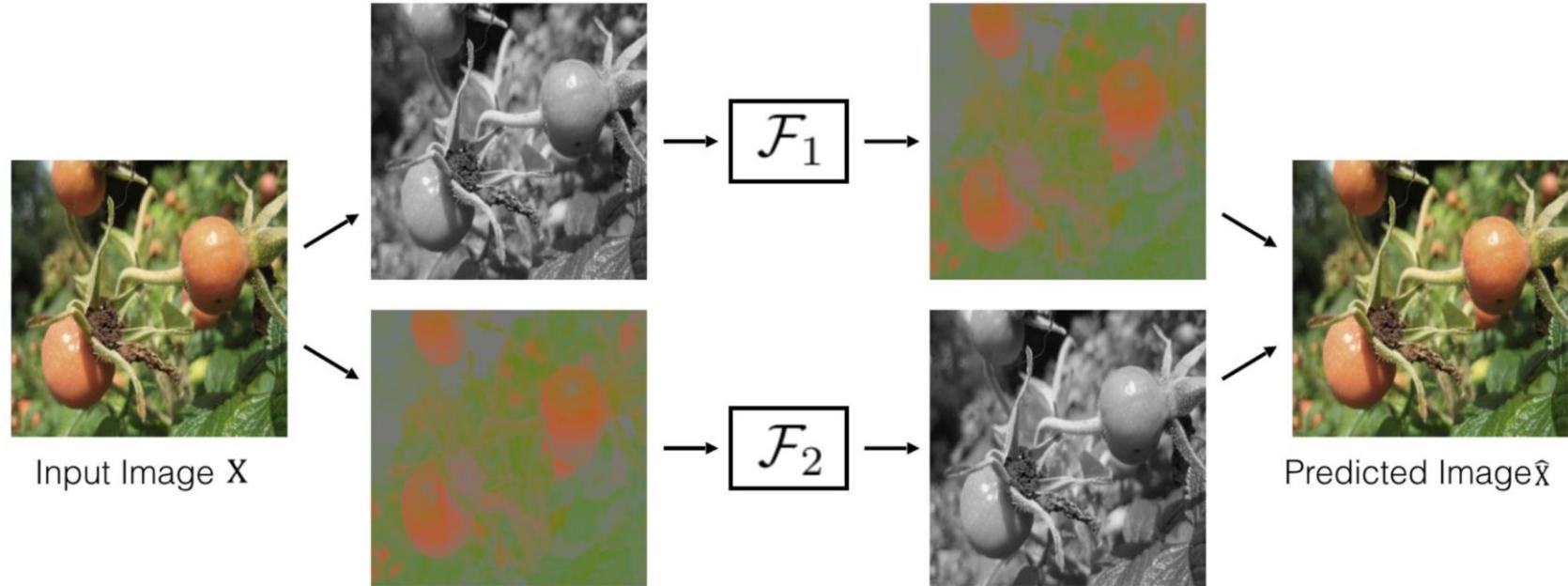
# Learning features from colorization: Split-brain Autoencoder

**Idea:** cross-channel predictions



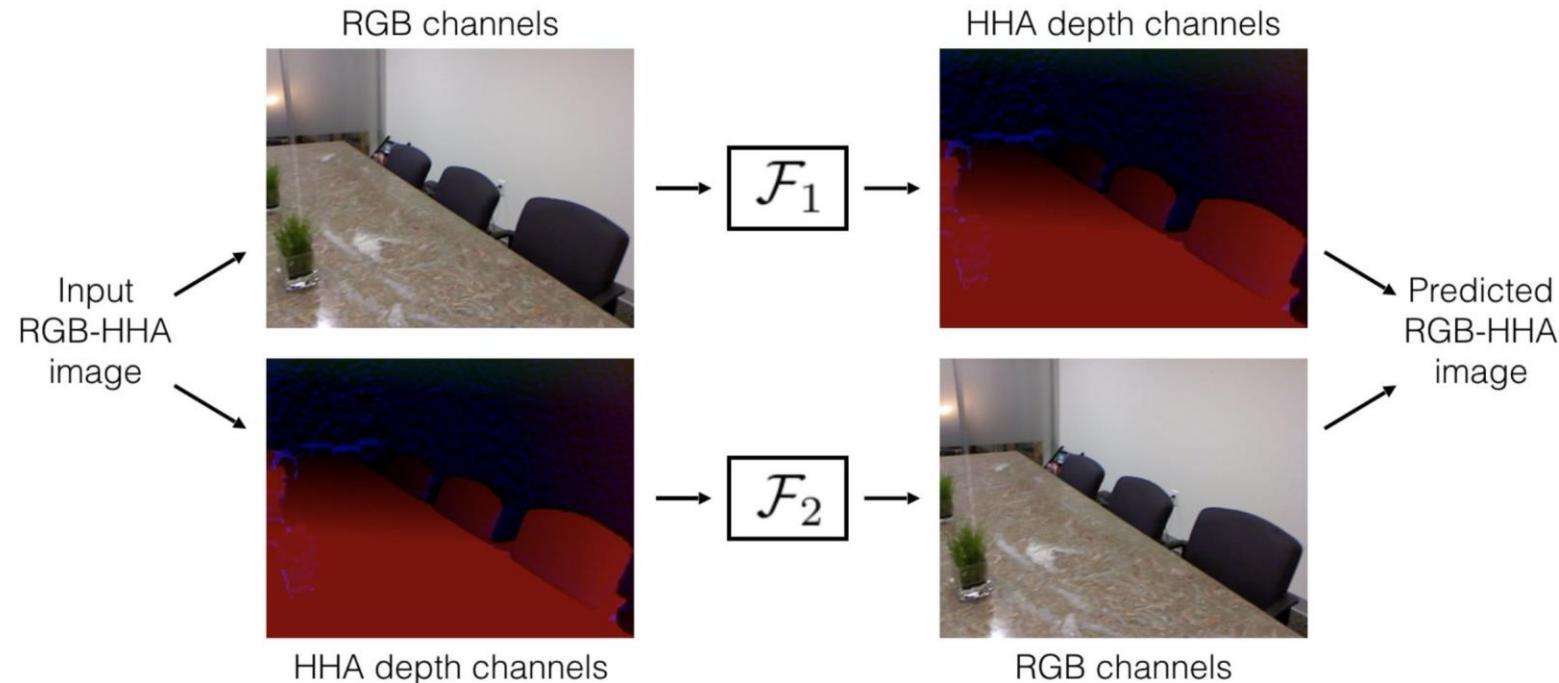
Source: Richard Zhang / Phillip Isola

# Learning features from colorization: Split-brain Autoencoder



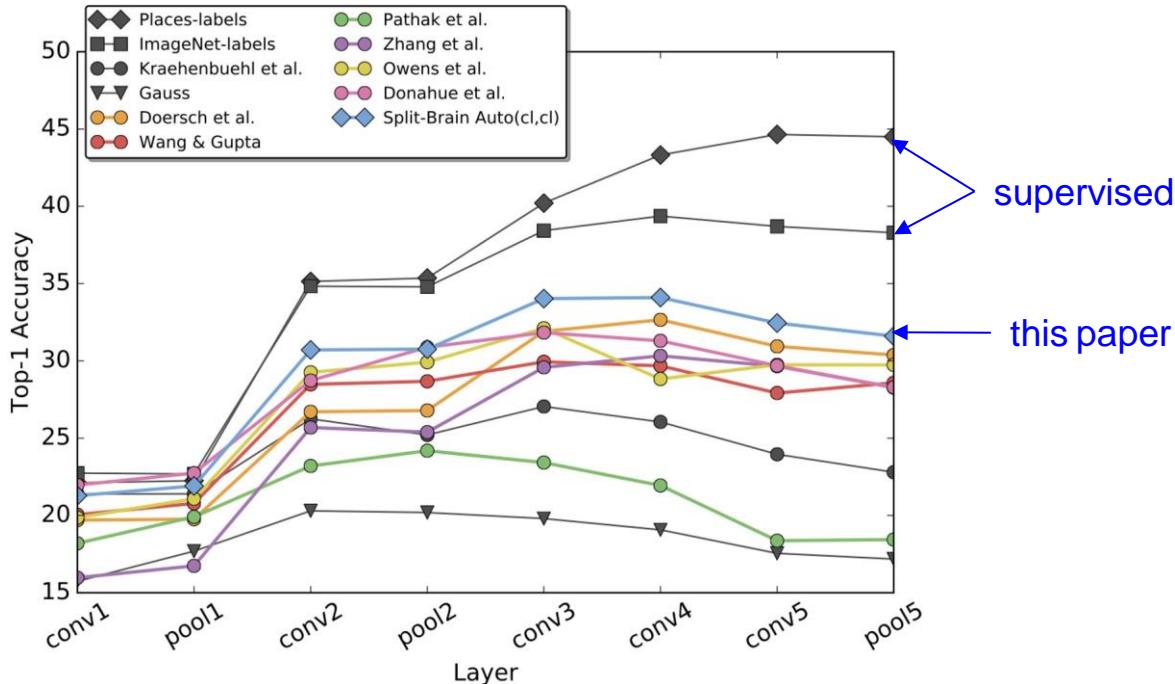
Source: Richard Zhang / Phillip Isola

# Learning features from colorization: Split-brain Autoencoder



Source: Richard Zhang / Phillip Isola

# Transfer learned features to supervised learning



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

Use *concatenated features* from  $F_1$  and  $F_2$

Labeled data is from the **Places** (Zhou 2016).

Source: [Zhang et al., 2017](#)

# Pretext task: image coloring



Source: Richard Zhang / Phillip Isola

# Pretext task: image coloring



Source: Richard Zhang / Phillip Isola

# Pretext task: video coloring

**Idea:** model the *temporal coherence* of colors in videos

reference frame



$t = 0$

how should I color these frames?



$t = 1$



$t = 2$



$t = 3$

...

Source: [Vondrick et al., 2018](#)

# Pretext task: video coloring

**Idea:** model the *temporal coherence* of colors in videos

reference frame



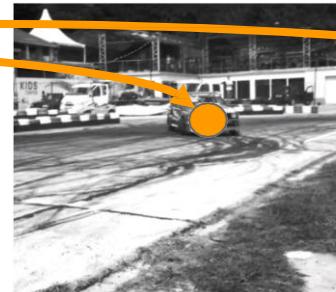
$t = 0$

how should I color these frames?

**Should be the same color!**



$t = 1$



$t = 2$



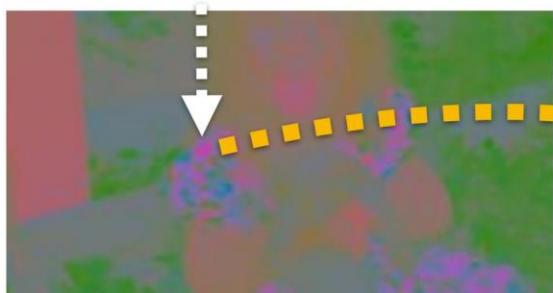
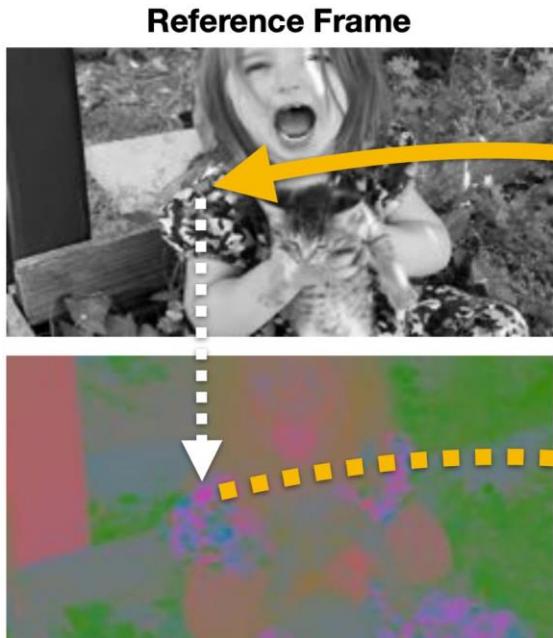
$t = 3$

...

**Hypothesis:** learning to color video frames should allow model to learn to track regions or objects without labels!

Source: [Vondrick et al., 2018](#)

# Learning to color videos



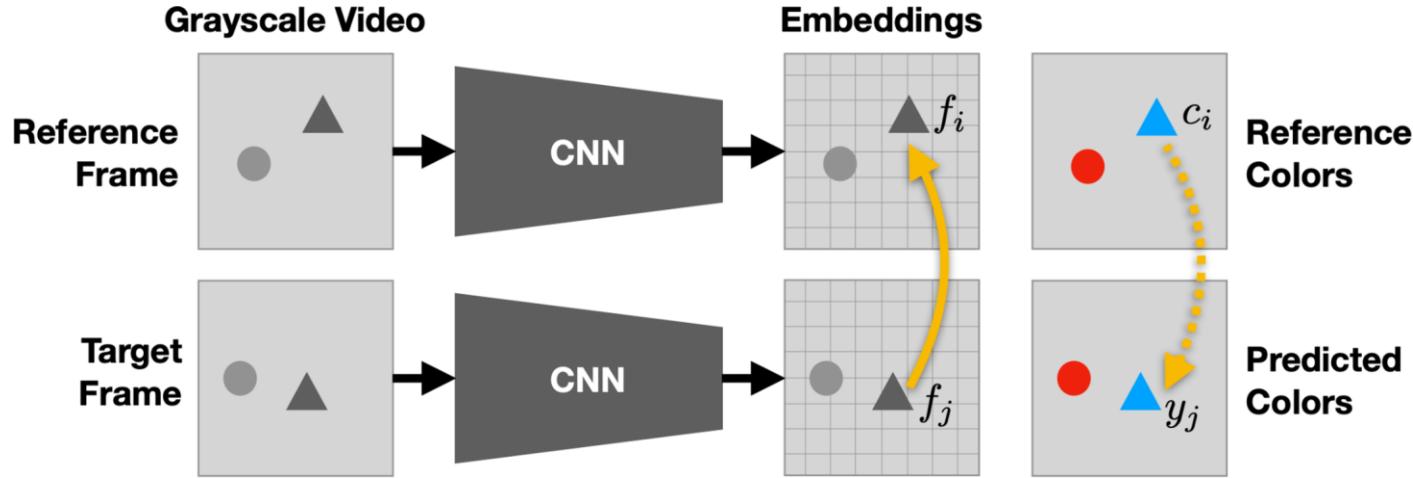
**Learning objective:**

Establish mappings between reference and target frames in a learned feature space.

Use the mapping as “pointers” to copy the correct color (LAB).

Source: [Vondrick et al., 2018](#)

# Learning to color videos

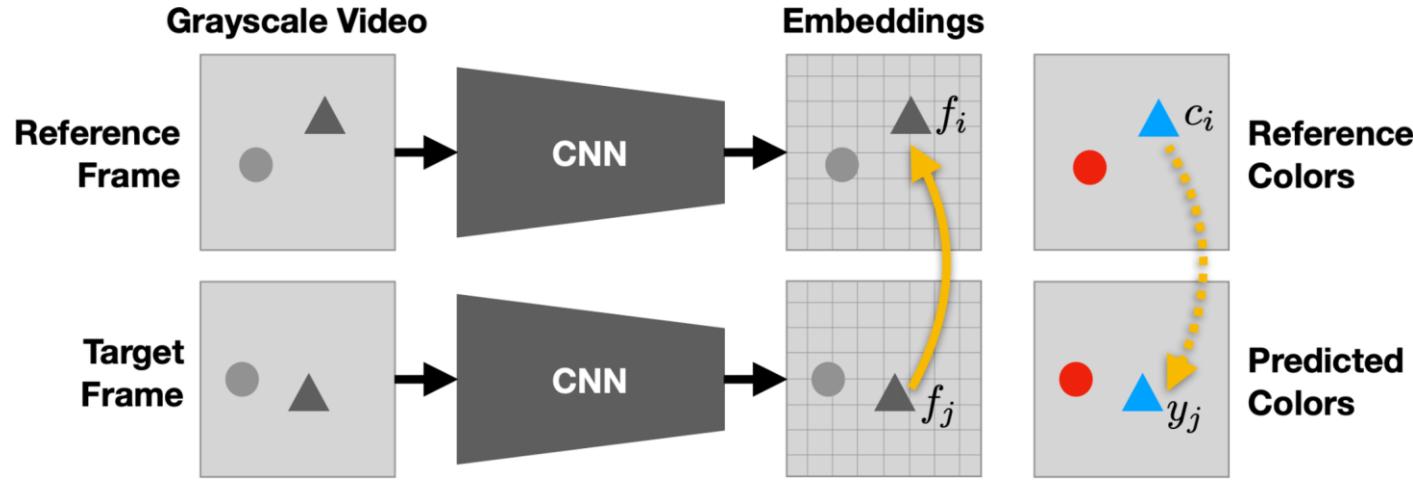


attention map on the  
reference frame

$$A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)}$$

Source: [Vondrick et al., 2018](#)

# Learning to color videos



attention map on the  
reference frame

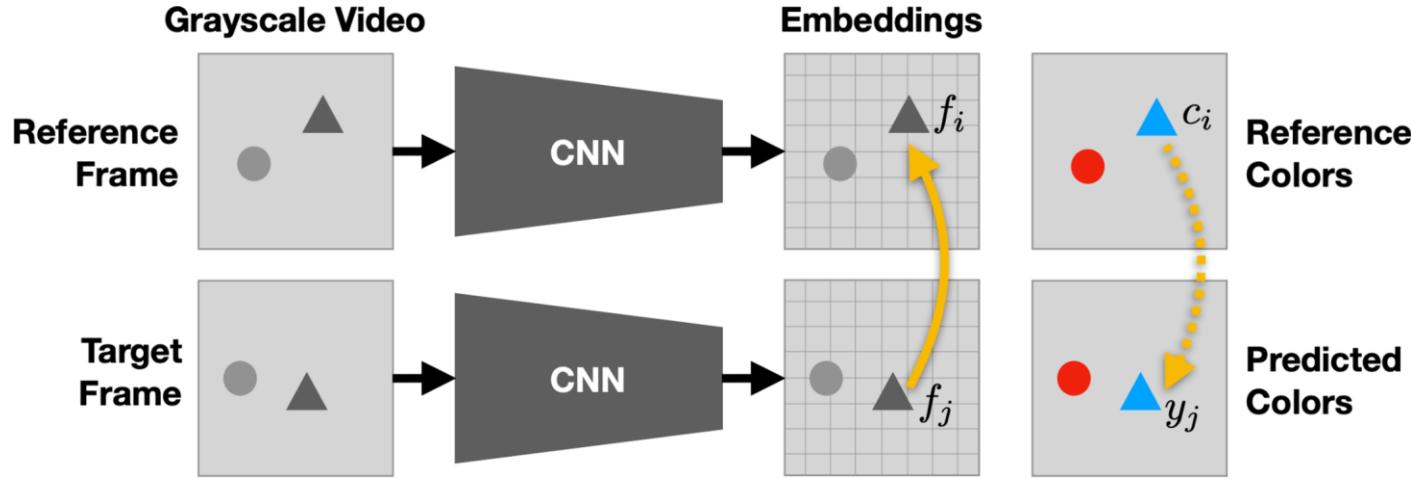
predicted color = weighted  
sum of the reference color

$$A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)}$$

$$y_j = \sum_i A_{ij} c_i$$

Source: [Vondrick et al., 2018](#)

# Learning to color videos



attention map on the  
reference frame

$$A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)}$$

predicted color = weighted  
sum of the reference color

$$y_j = \sum_i A_{ij} c_i$$

loss between predicted color  
and ground truth color

$$\min_{\theta} \sum_j \mathcal{L}(y_j, c_j)$$

Source: [Vondrick et al., 2018](#)

# Colorizing videos (qualitative)

reference frame



target frames (gray)



predicted color



Source: [Google AI blog post](#)

# Colorizing videos (qualitative)

reference frame



target frames (gray)



predicted color



Source: [Google AI blog post](#)

# Tracking emerges from colorization

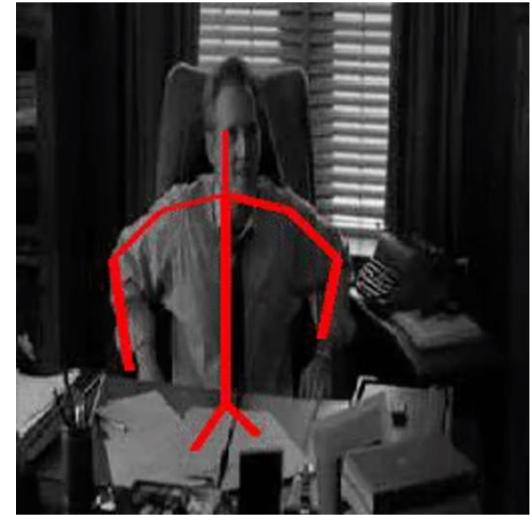
Propagate segmentation masks using learned attention



Source: [Google AI blog post](#)

# Tracking emerges from colorization

Propagate pose keypoints using learned attention



Source: [Google AI blog post](#)

# Summary: pretext tasks from image transformations

- Pretext tasks focus on “visual common sense”, e.g., predict rotations, inpainting, rearrangement, and colorization.
- The models are forced learn good features about natural images, e.g., semantic representation of an object category, in order to solve the pretext tasks.
- We don’t care about the performance of these pretext tasks, but rather how useful the learned features are for downstream tasks (classification, detection, segmentation).

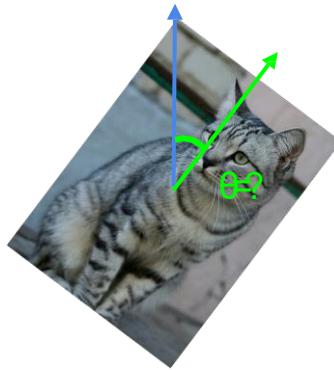
# Summary: pretext tasks from image transformations

- Pretext tasks focus on “visual common sense”, e.g., predict rotations, inpainting, rearrangement, and colorization.
- The models are forced learn good features about natural images, e.g., semantic representation of an object category, in order to solve the pretext tasks.
- We don’t care about the performance of these pretext tasks, but rather how useful the learned features are for downstream tasks (classification, detection, segmentation).
- Problems: 1) coming up with individual pretext tasks is tedious, and 2) the learned representations may not be general.

# Pretext tasks from image transformations



image completion



rotation prediction



“jigsaw puzzle”

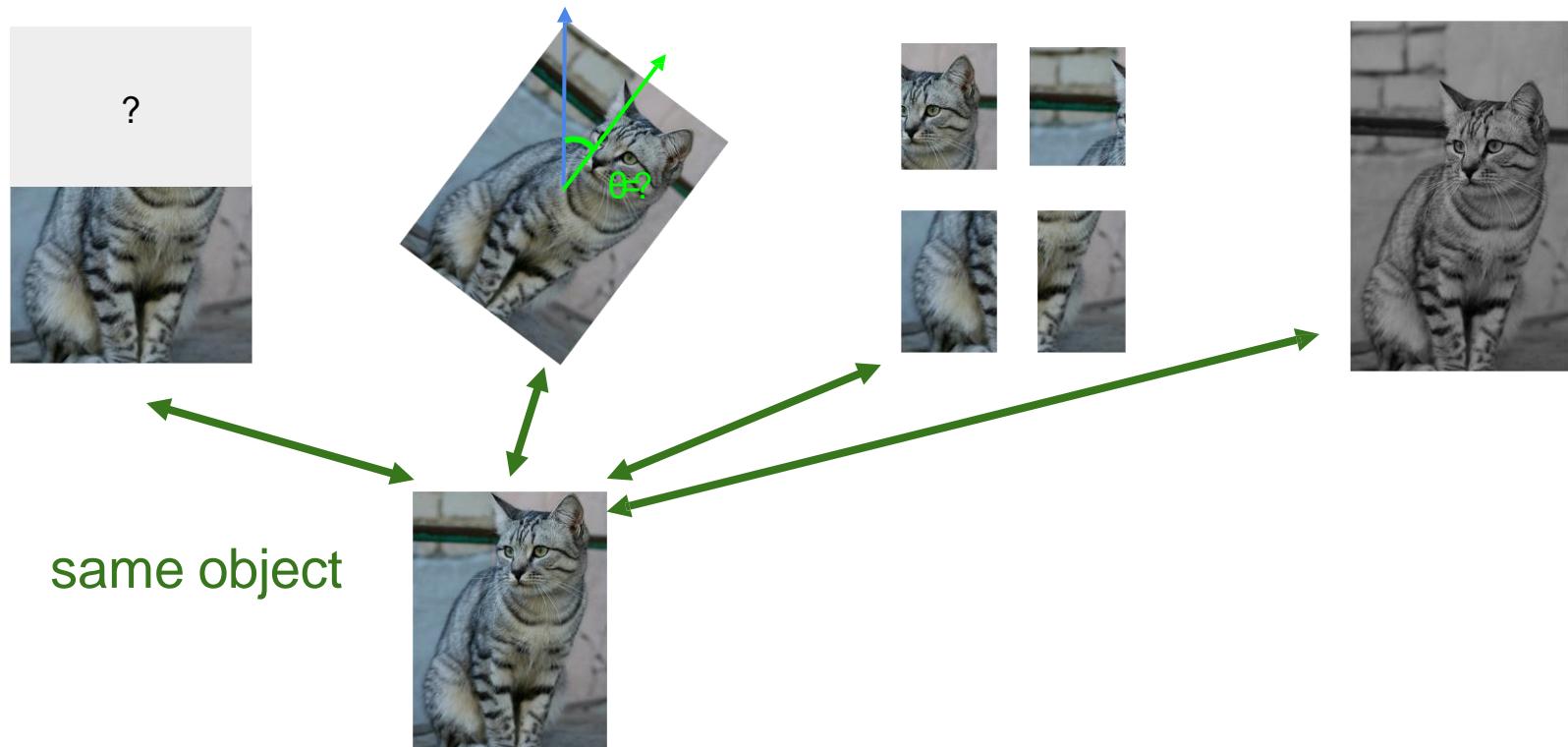


colorization

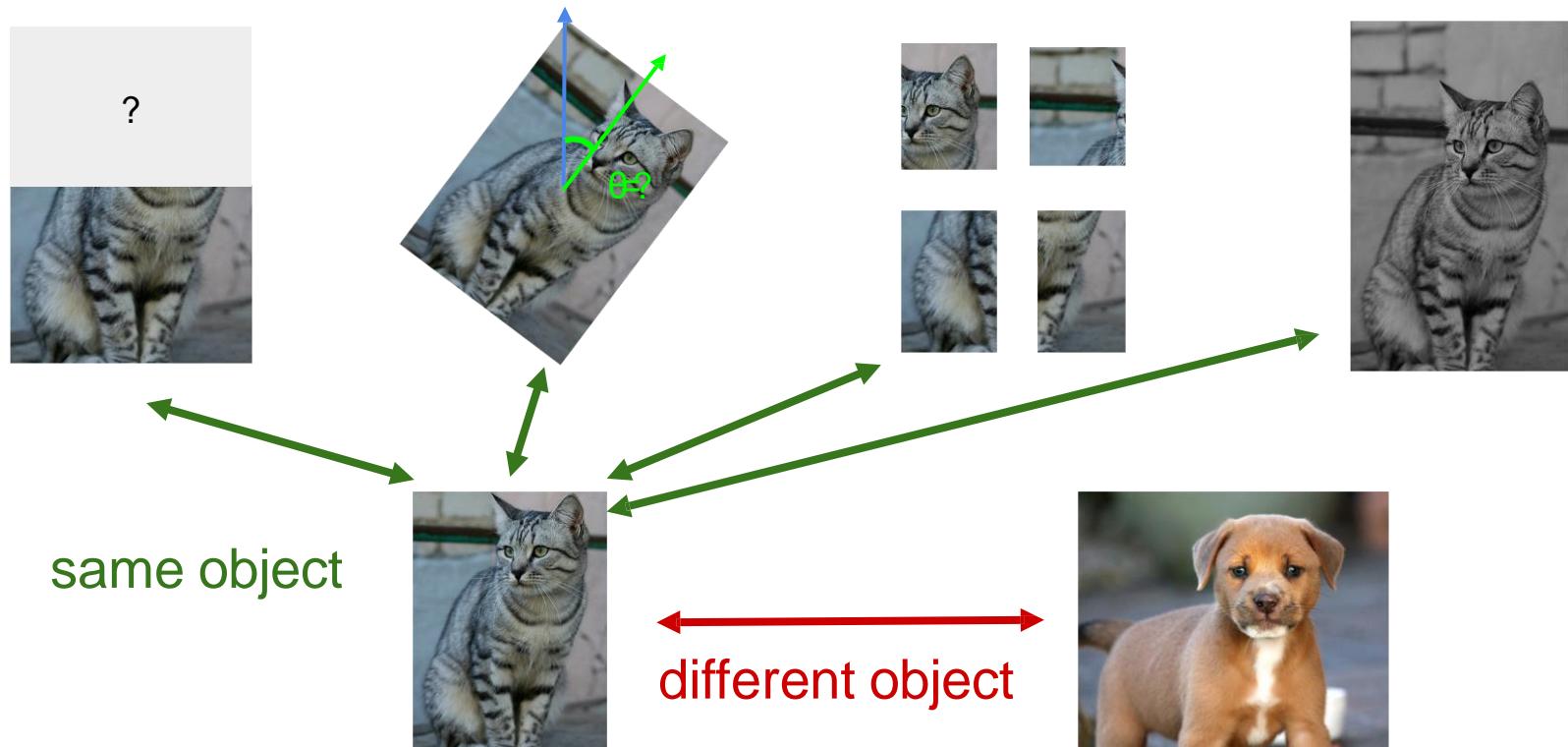
Learned representations may be tied to a specific pretext task!

Can we come up with a more general pretext task?

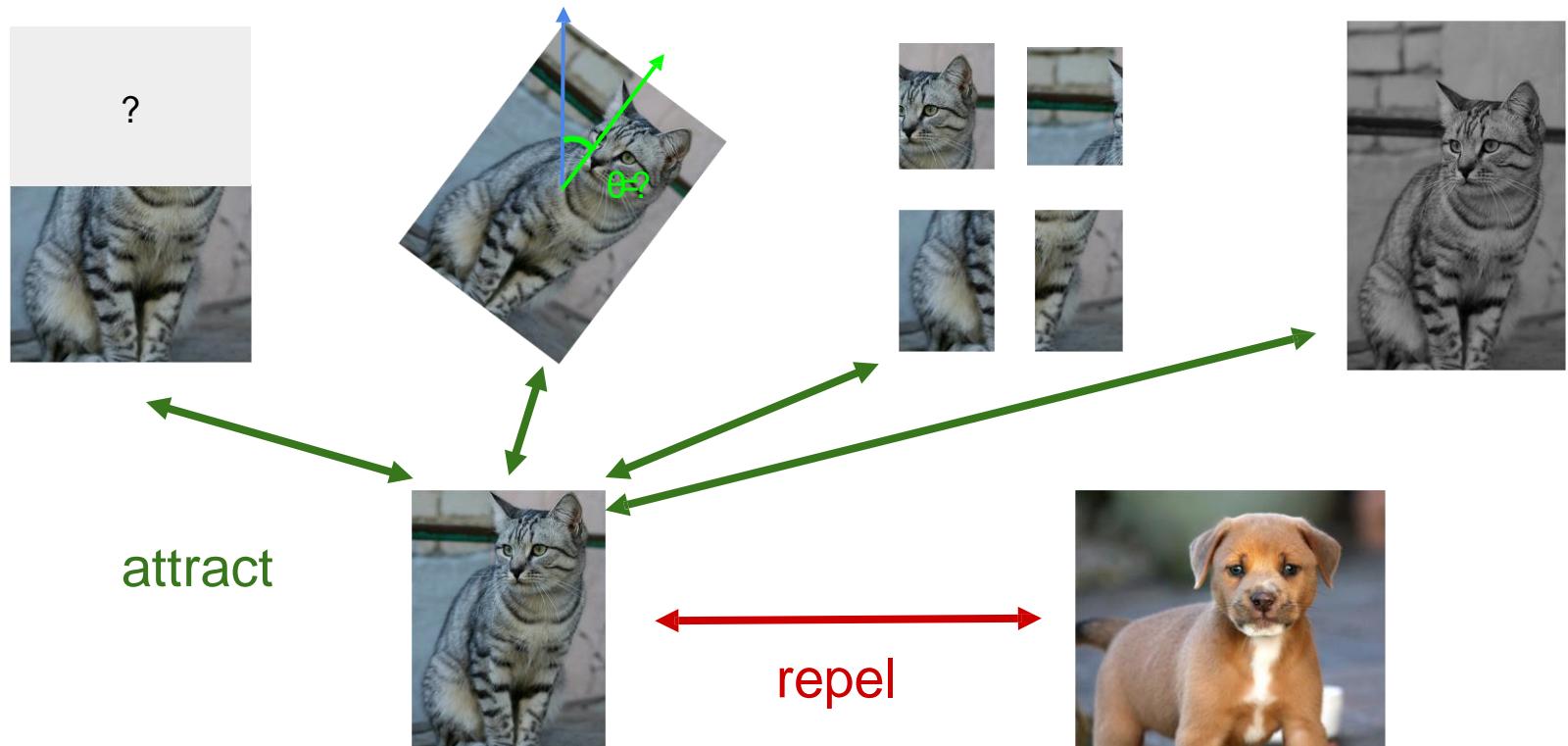
# A more general pretext task?



# A more general pretext task?



# Contrastive Representation Learning



# Today's Agenda

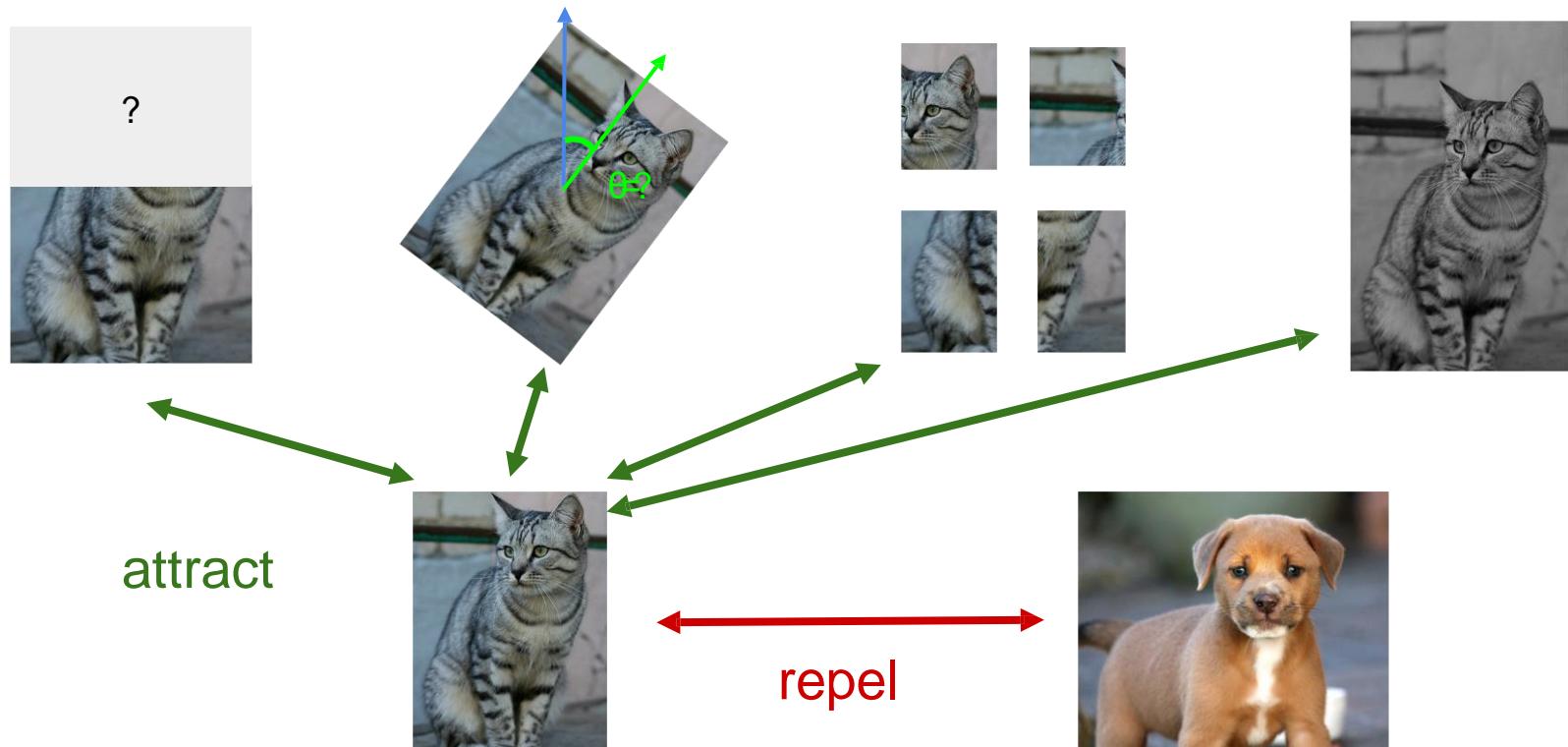
## Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring

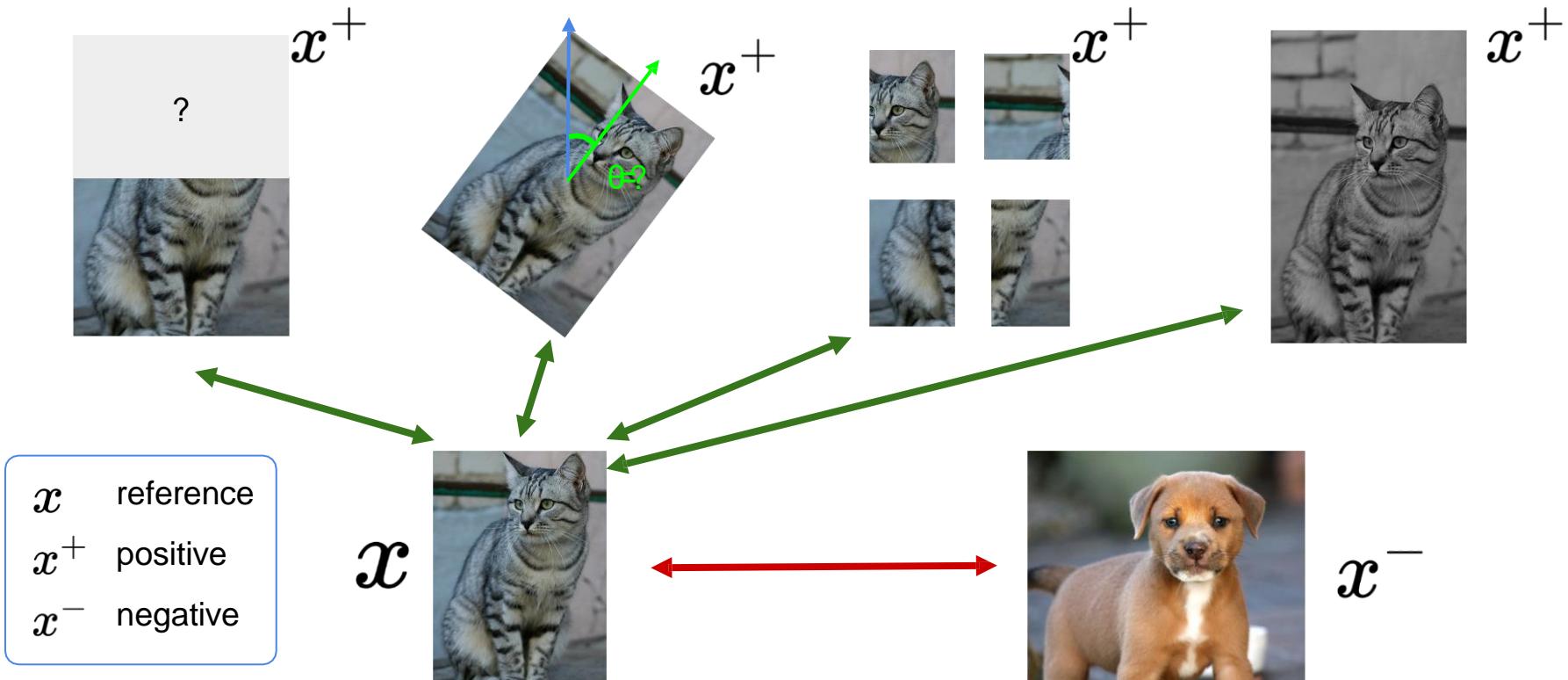
## Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

# Contrastive Representation Learning



# Contrastive Representation Learning



# A formulation of contrastive learning

What we want:

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

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

Given a chosen score function, we aim to learn an **encoder 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:

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

# A formulation of contrastive learning

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

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

 $x$  $x^+$  $x$  $x_1^-$  $x_2^-$  $x_3^-$  $\dots$

# A formulation of contrastive learning

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

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

score for the positive pair
score for the N-1 negative pairs

## This seems familiar ...

# A formulation of contrastive learning

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

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

score for the positive pair
score for the N-1 negative pairs

This seems familiar ...

# Cross entropy loss for a N-way softmax classifier!

I.e., learn to find the positive sample from the N samples

# A 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

Detailed derivation: [Poole et al., 2019](#)

# SimCLR: A Simple Framework for Contrastive Learning

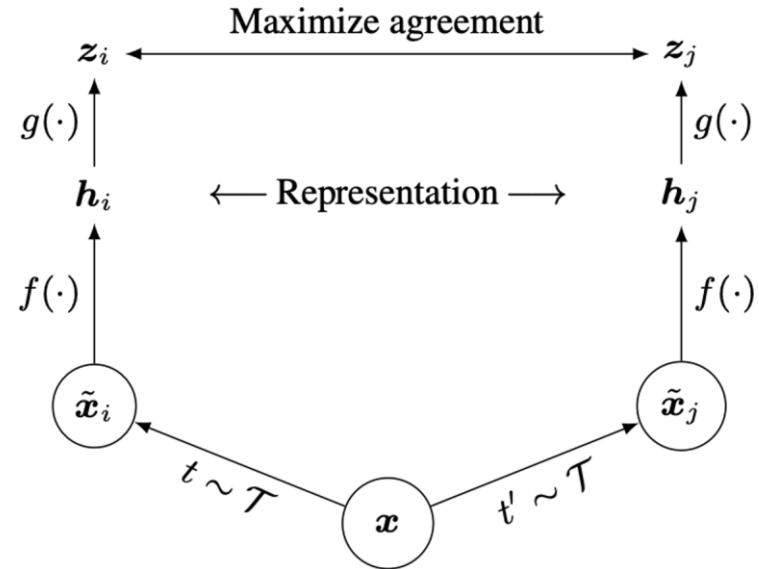
Cosine similarity as the score function:

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

Use a projection network  $\mathbf{g}(\cdot)$  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.



Source: [Chen et al., 2020](#)

# SimCLR: generating positive samples from data augmentation



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate  $\{90^\circ, 180^\circ, 270^\circ\}$



(g) Cutout



(h) Gaussian noise



(i) Gaussian blur



(j) Sobel filtering

Source: [Chen et al., 2020](#)

# SimCLR

Generate a positive pair  
by sampling data  
augmentation functions

---

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

---

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .  
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do  
    for all  $k \in \{1, \dots, N\}$  do  
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$   
        # the first augmentation  
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$   
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation  
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection  
        # the second augmentation  
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$   
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation  
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection  
    end for  
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do  
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # 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  $g(\cdot)$ 
```

---

\*We use a slightly different formulation in the assignment.  
You should follow the assignment instructions.

Source: [Chen et al., 2020](#)

# SimCLR

Generate a positive pair  
by sampling data  
augmentation functions

---

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

---

```

input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do
    for all  $k \in \{1, \dots, N\}$  do
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$ 
        # the first augmentation
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection
        # the second augmentation
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection
    end for
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # 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  $g(\cdot)$ 

```

---

\*We use a slightly different formulation in the assignment.  
You should follow the assignment instructions.

InfoNCE loss:  
Use all non-positive samples in the batch as  $x^-$

Source: [Chen et al., 2020](#)

# SimCLR

Generate a positive pair  
by sampling data  
augmentation functions

Iterate through and  
use each of the  $2N$   
sample as reference,  
compute average loss

---

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

---

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .  
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do  
    for all  $k \in \{1, \dots, N\}$  do  
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$   
        # the first augmentation  
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$   
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation  
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection  
        # the second augmentation  
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$   
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation  
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection  
    end for  
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do  
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # 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  $g(\cdot)$ 
```

---

\*We use a slightly different formulation in the assignment.  
You should follow the assignment instructions.

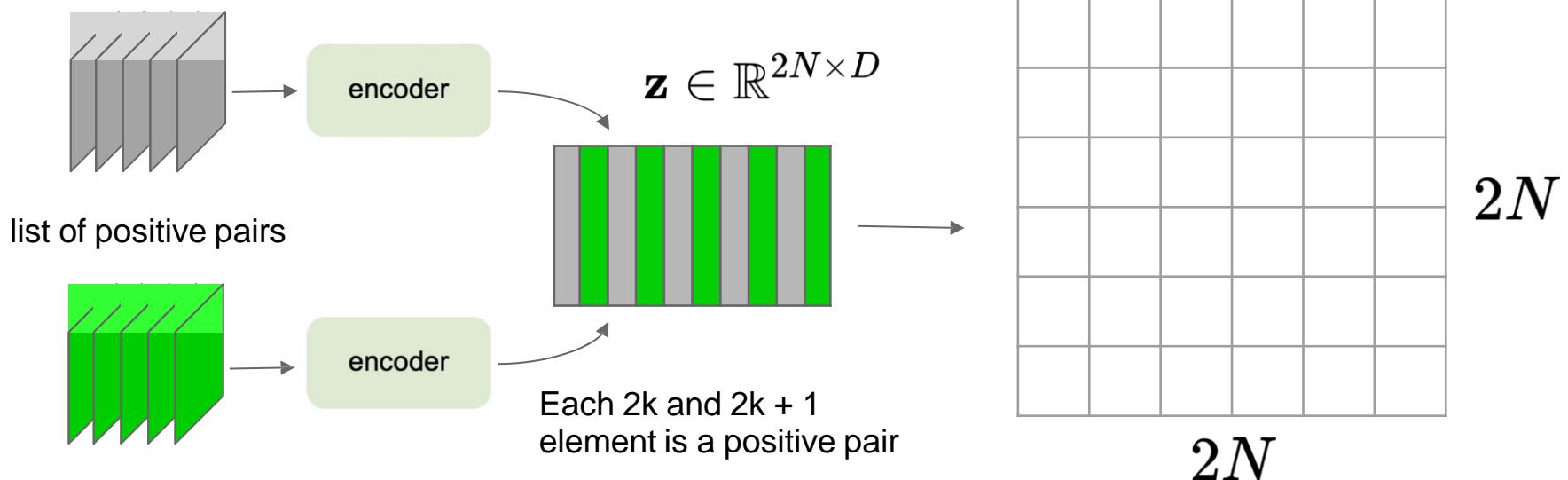
InfoNCE loss:  
Use all non-positive samples in the batch as  $x^-$

Source: [Chen et al., 2020](#)

# SimCLR: mini-batch training

$$s_{i,j} = \frac{z_i^T z_j}{\|z_i\| \|z_j\|}$$

“Affinity matrix”

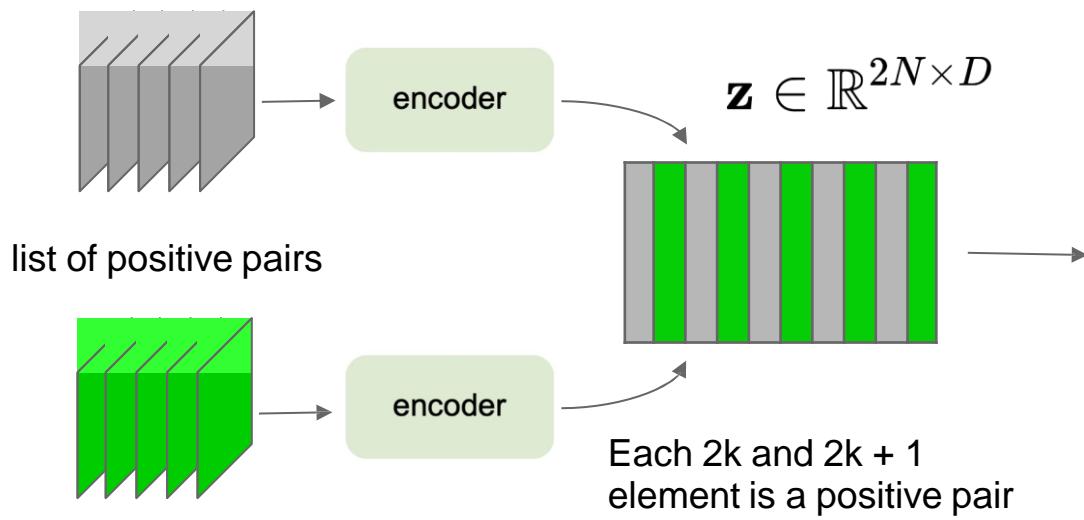


\*We use a slightly different formulation in the assignment.  
You should follow the assignment instructions.

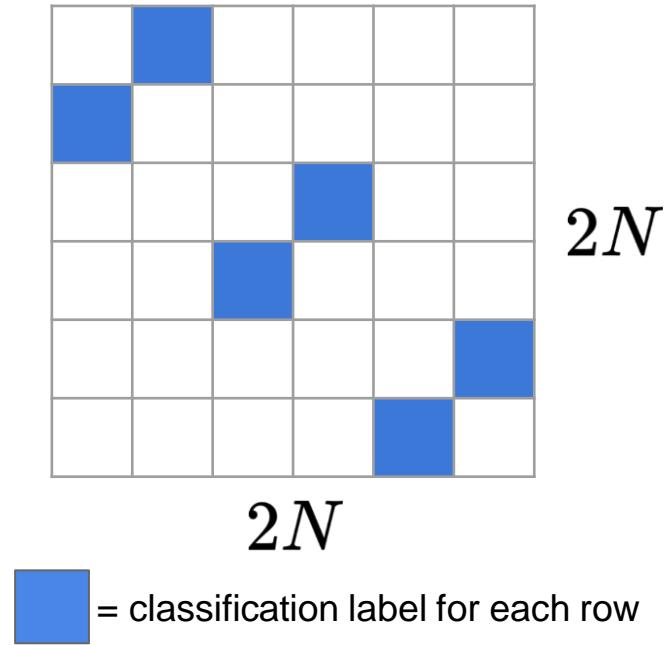
# SimCLR: mini-batch training

$$s_{i,j} = \frac{z_i^T z_j}{\|z_i\| \|z_j\|}$$

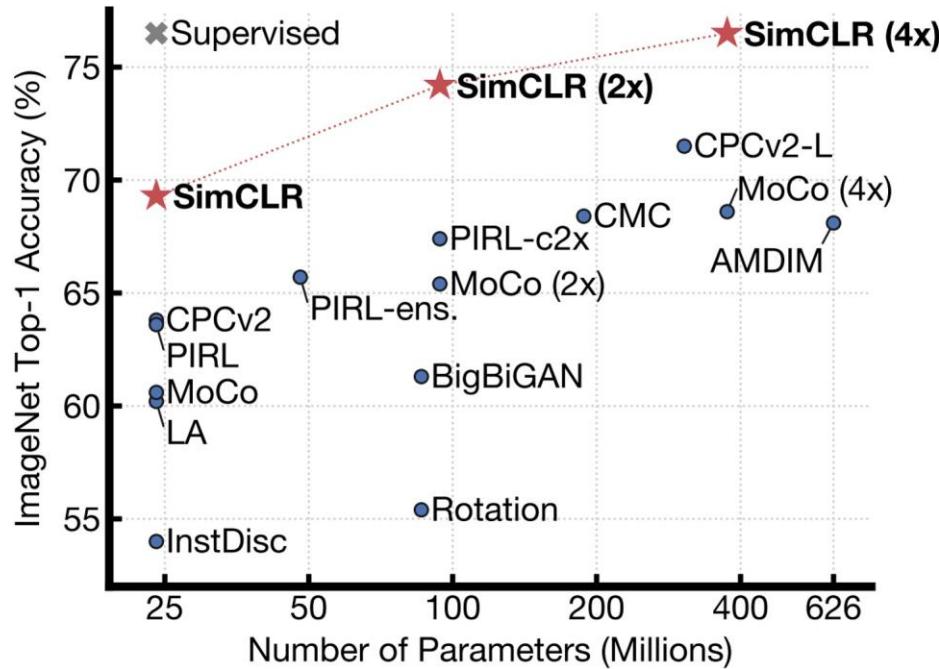
“Affinity matrix”



\*We use a slightly different formulation in the assignment.  
You should follow the assignment instructions.



# Training linear classifier on SimCLR features



Train feature encoder on **ImageNet** (entire training set) using SimCLR.

Freeze feature encoder, train a linear classifier on top with labeled data.

Source: [Chen et al., 2020](#)

# Semi-supervised learning on SimCLR features

Method	Architecture	Label fraction		
		1%	10%	Top 5
Supervised baseline	ResNet-50	48.4	80.4	
<i>Methods using other label-propagation:</i>				
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	
<i>Methods using representation learning only:</i>				
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×)	<b>85.8</b>	<b>92.6</b>	

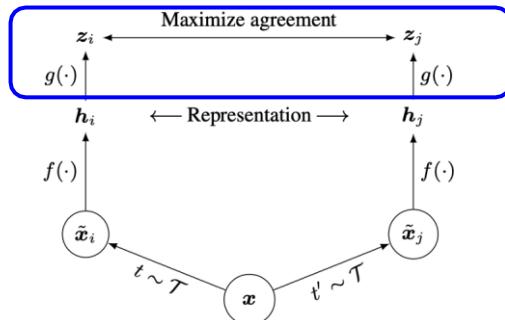
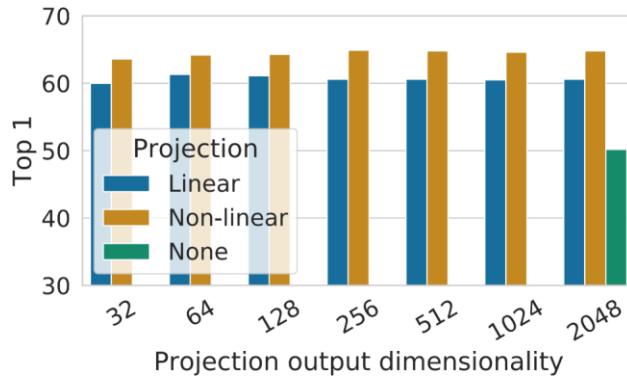
Table 7. ImageNet accuracy of models trained with few labels.

Train feature encoder on **ImageNet** (entire training set) using SimCLR.

**Finetune** the encoder with 1% / 10% of labeled data on ImageNet.

Source: [Chen et al., 2020](#)

# SimCLR design choices: projection head



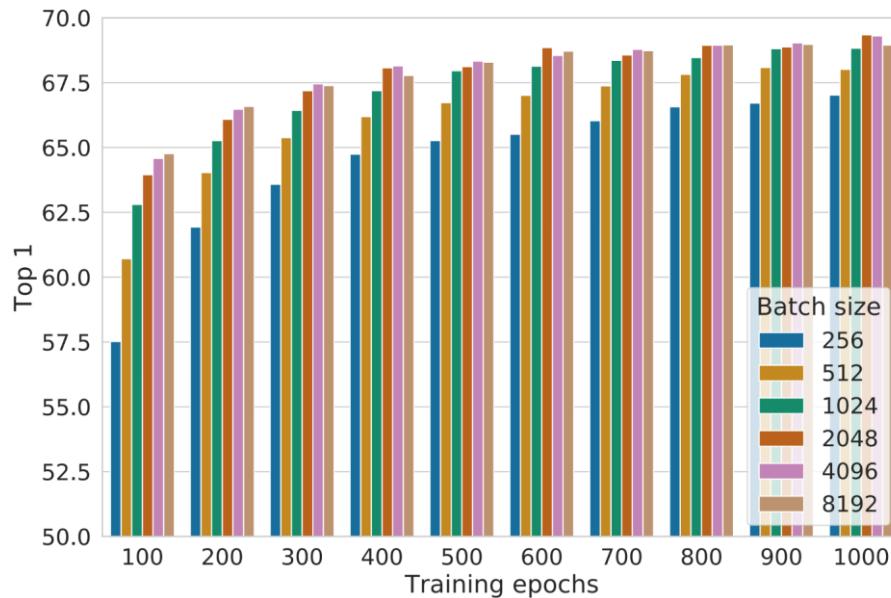
Linear / non-linear projection heads improve representation learning.

A possible explanation:

- contrastive learning objective may discard useful information for downstream tasks
- representation space  $\mathbf{z}$  is trained to be invariant to data transformation.
- by leveraging the projection head  $\mathbf{g}(\cdot)$ , more information can be preserved in the  $\mathbf{h}$  representation space

Source: [Chen et al., 2020](#)

# SimCLR design choices: large batch size



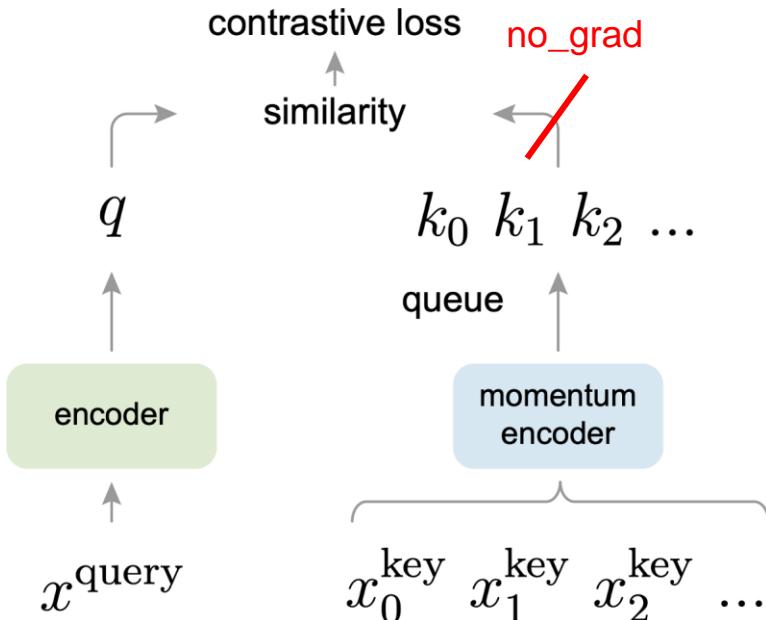
Large training batch size is crucial for SimCLR!

Large batch size causes large memory footprint during backpropagation:  
**requires distributed training on TPUs (ImageNet experiments)**

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

Source: [Chen et al., 2020](#)

# Momentum Contrastive Learning (MoCo)

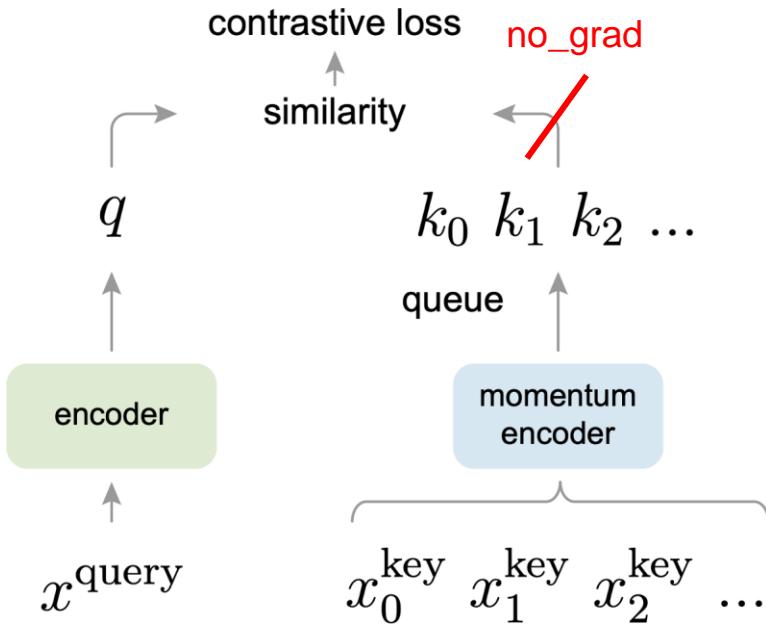


## Key differences to SimCLR:

- Keep a running **queue** of keys (negative samples).
- Compute gradients and update the encoder **only through the queries**.
- Decouple min-batch size with the number of keys: can support **a large number of negative samples**.

Source: [He et al., 2020](#)

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

Source: [He et al., 2020](#)

# MoCo

Generate a positive pair  
by sampling data  
augmentation functions

No gradient through  
the key

Update the FIFO  
negative sample queue

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

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

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
    x_q = aug(x) # a randomly augmented version
    x_k = aug(x) # another randomly augmented version

    q = f_q.forward(x_q) # queries: NxC
    k = f_k.forward(x_k) # keys: NxC
    k = k.detach() # no gradient to keys

    # positive logits: Nx1
    l_pos = bmm(q.view(N, 1, C), k.view(N, C, 1))

    # negative logits: NxK
    l_neg = mm(q.view(N, C), queue.view(C, K))

    # logits: Nx(1+K)
    logits = cat([l_pos, l_neg], dim=1)

    # contrastive loss, Eqn.(1)
    labels = zeros(N) # positives are the 0-th
    loss = CrossEntropyLoss(logits/t, labels)

    # SGD update: query network
    loss.backward()
    update(f_q.params)

    # momentum update: key network
    f_k.params = m*f_k.params+(1-m)*f_q.params

    # update dictionary
    enqueue(queue, k) # enqueue the current minibatch
    dequeue(queue) # dequeue the earliest minibatch
```

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

Use the running  
queue of keys as the  
negative samples

InfoNCE loss

Update  $f_k$  through  
momentum

Source: [He et al., 2020](#)

# “MoCo V2”

## Improved Baselines with Momentum Contrastive Learning

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

A hybrid 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 (no TPU required!).

Source: [Chen et al., 2020](#)

# MoCo vs. SimCLR vs. MoCo V2

case	unsup. pre-train				ImageNet acc.	VOC detection		
	MLP	aug+	cos	epochs		AP <sub>50</sub>	AP	AP <sub>75</sub>
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	<b>82.5</b>	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	<b>800</b>	<b>71.1</b>	<b>82.5</b>	<b>57.4</b>	<b>64.0</b>

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.

## Key takeaways:

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

Source: [Chen et al., 2020](#)

# MoCo vs. SimCLR vs. MoCo V2

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

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).

## Key takeaways:

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.
- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).

Source: [Chen et al., 2020](#)

# MoCo vs. SimCLR vs. MoCo V2

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

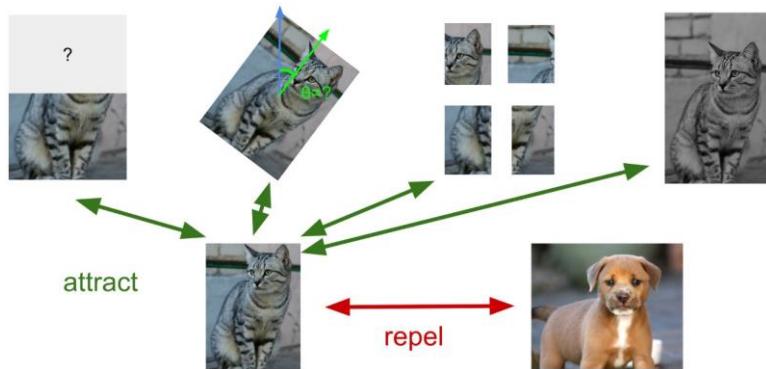
Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. <sup>†</sup>: based on our estimation.

## Key takeaways:

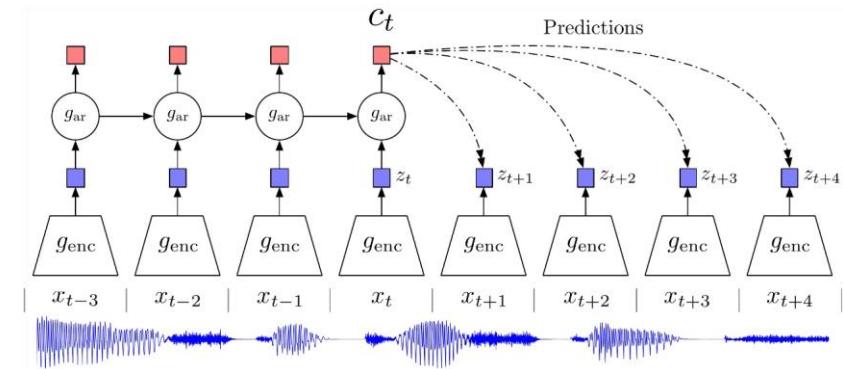
- Non-linear projection head and strong data augmentation are crucial for contrastive learning.
- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).
- ... all with much smaller memory footprint! (“end-to-end” means SimCLR here)

Source: [Chen et al., 2020](#)

# Instance vs. Sequence Contrastive Learning



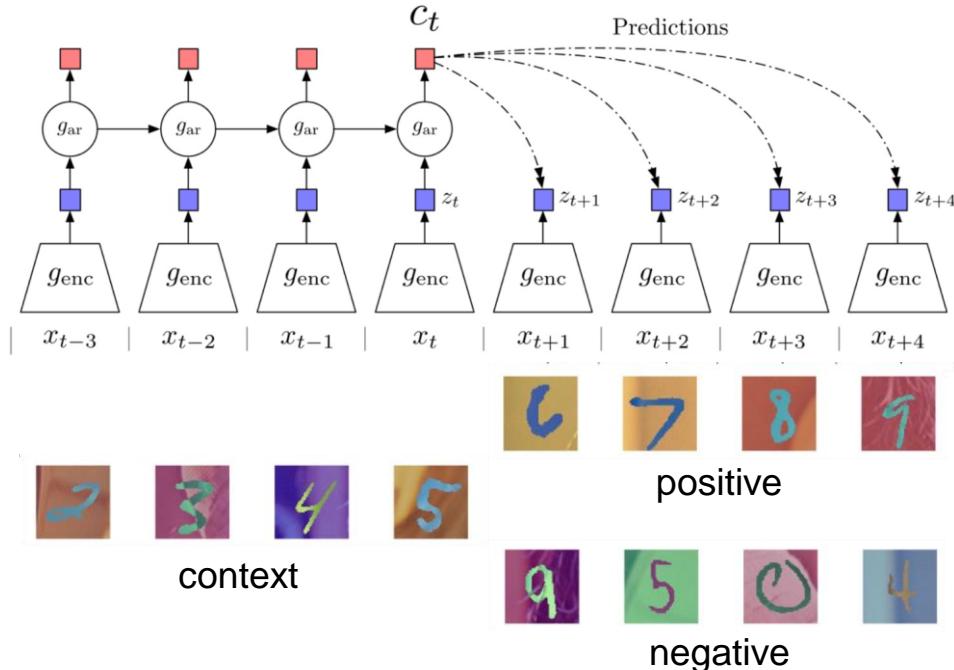
**Instance-level contrastive learning:**  
contrastive learning based on  
**positive & negative instances.**  
Examples: SimCLR, MoCo



Source: [van den Oord et al., 2018](#)

**Sequence-level contrastive learning:**  
contrastive learning based on  
**sequential / temporal orders.**  
Example: **Contrastive Predictive Coding (CPC)**

# Contrastive Predictive Coding (CPC)



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

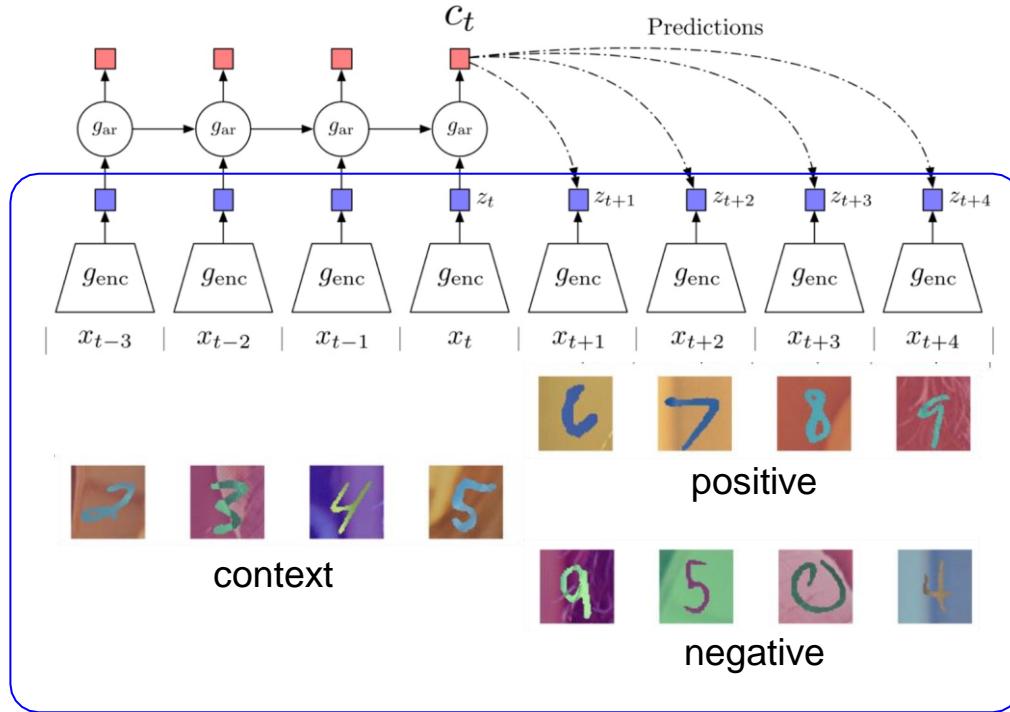
**Predictive:** the model has to predict future patterns given the current context.

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

Figure [source](#)

Source: [van den Oord et al., 2018](#),

# Contrastive Predictive Coding (CPC)

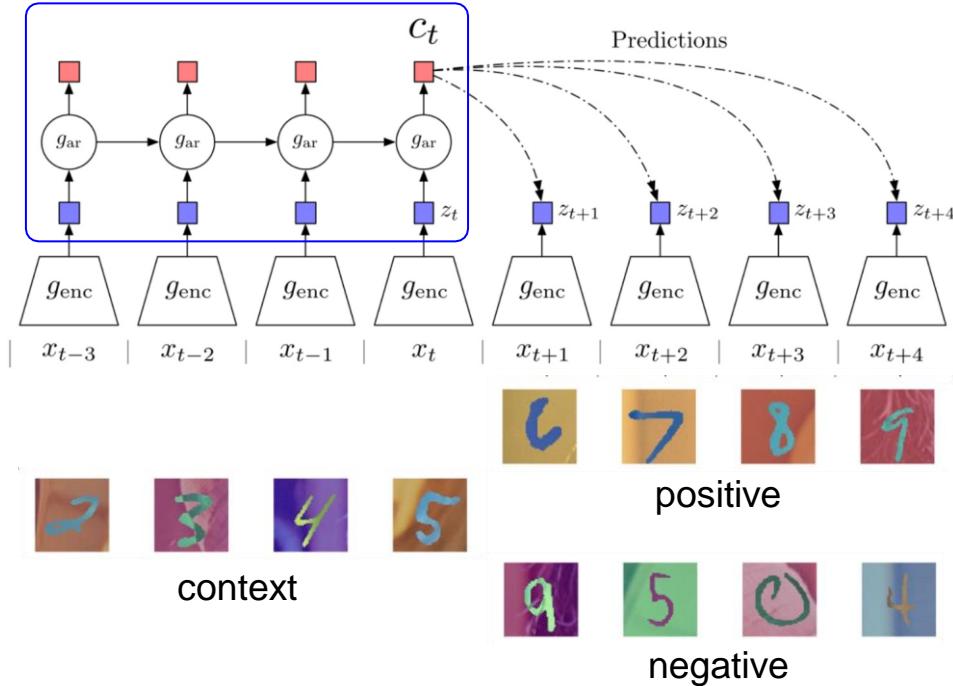


1. Encode all samples in a sequence into vectors  $\mathbf{z}_t = \mathbf{g}_{enc}(\mathbf{x}_t)$

Figure [source](#)

Source: [van den Oord et al., 2018](#),

# Contrastive Predictive Coding (CPC)

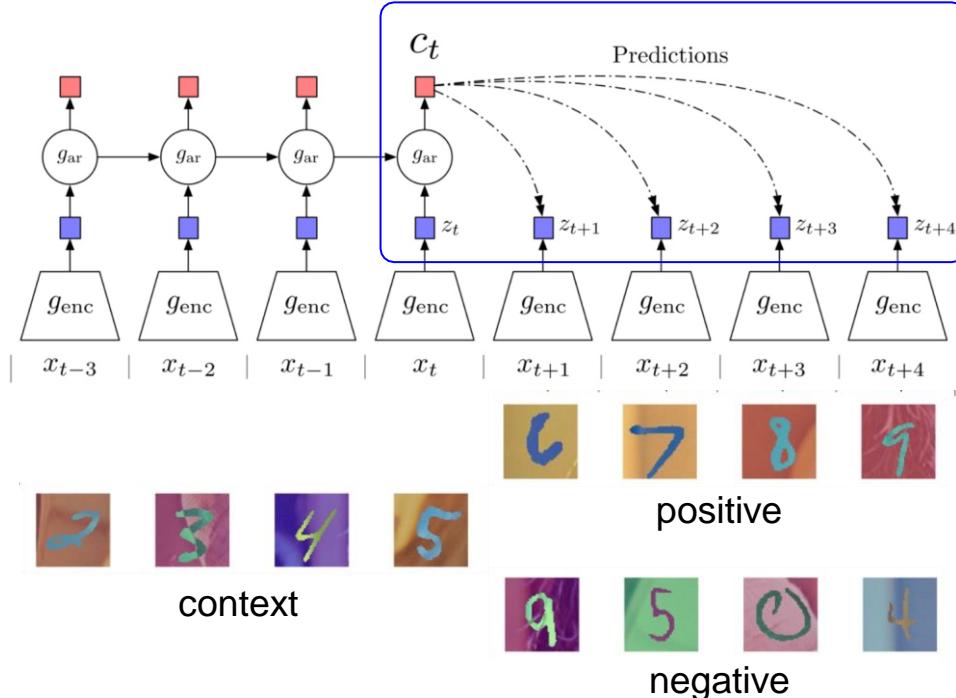


1. Encode all samples in a sequence into vectors  $\mathbf{z}_t = \mathbf{g}_{enc}(\mathbf{x}_t)$
2. Summarize context (e.g., half of a sequence) into a context code  $\mathbf{c}_t$  using an auto-regressive model ( $\mathbf{g}_{ar}$ ). The original paper uses GRU-RNN here.

Figure [source](#)

Source: [van den Oord et al., 2018](#),

# Contrastive Predictive Coding (CPC)



1. Encode all samples in a sequence into vectors  $\mathbf{z}_t = \mathbf{g}_{enc}(\mathbf{x}_t)$
2. Summarize context (e.g., half of a sequence) into a context code  $\mathbf{c}_t$  using an auto-regressive model ( $\mathbf{g}_{ar}$ )
3. Compute InfoNCE loss between the context  $\mathbf{c}_t$  and future code  $\mathbf{z}_{t+k}$  using the following **time-dependent score function**:

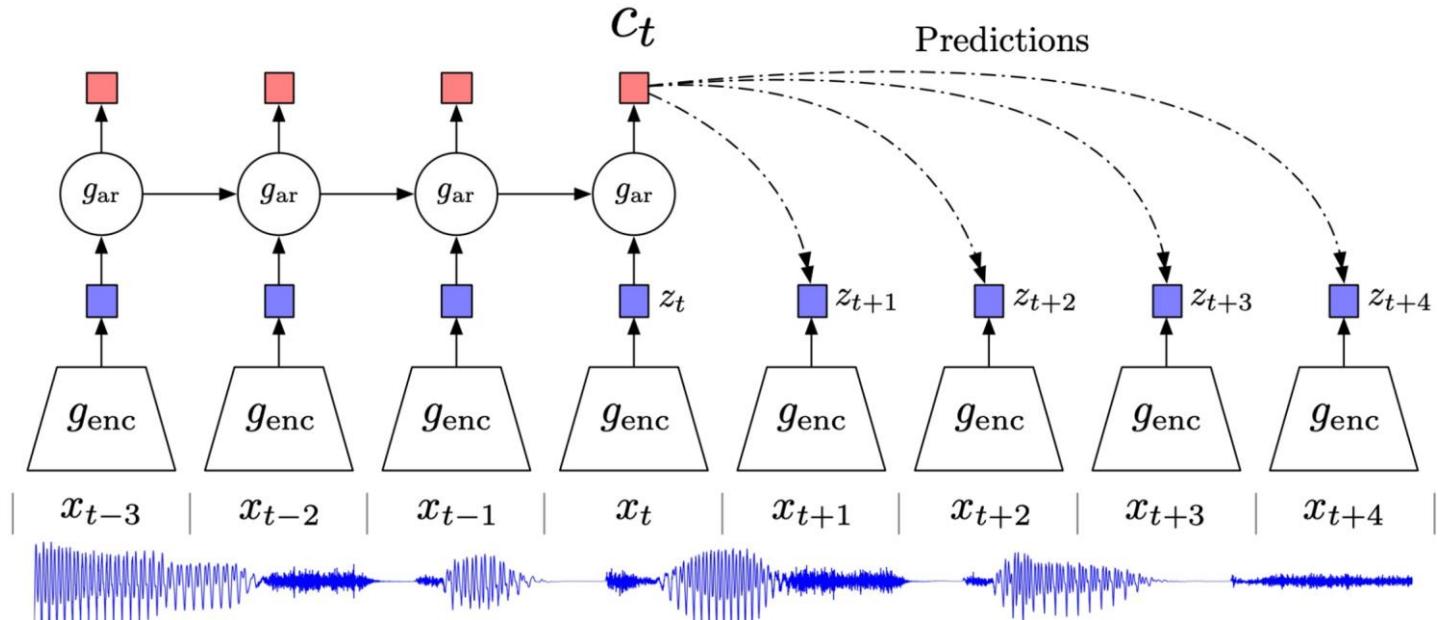
$$s_k(z_{t+k}, c_t) = z_{t+k}^T W_k c_t$$

, where  $W_k$  is a trainable matrix.

Figure [source](#)

Source: [van den Oord et al., 2018](#),

# CPC example: modeling audio sequences



Source: [van den Oord et al., 2018](#),

# CPC example: modeling audio sequences

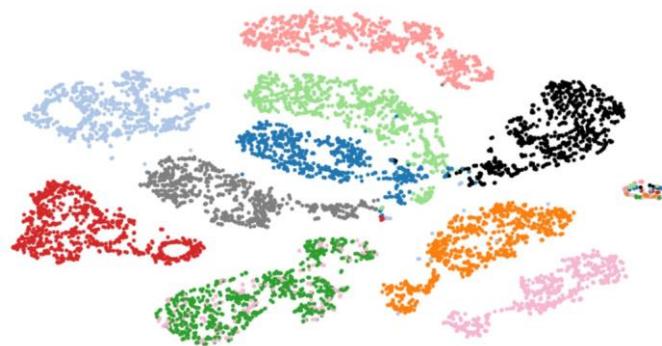


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

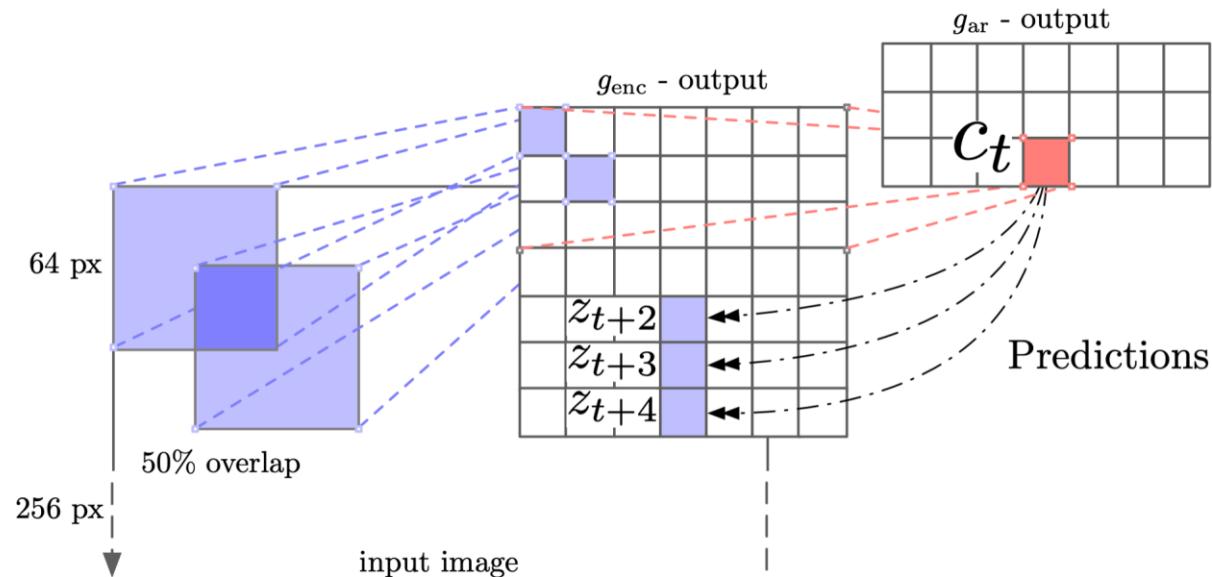
Method	ACC
<b>Phone classification</b>	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
<b>Speaker classification</b>	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

Linear classification on trained representations (LibriSpeech dataset)

Source: [van den Oord et al., 2018](#),

# CPC example: modeling visual context

**Idea:** split image into patches, model rows of patches from top to bottom as a sequence. I.e., use top rows as context to predict bottom rows.



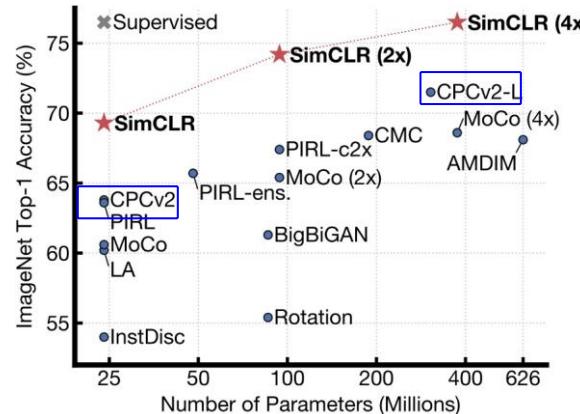
Source: [van den Oord et al., 2018](#),

# CPC example: modeling visual context

Method	Top-1 ACC
<b>Using AlexNet conv5</b>	
Video [28]	29.8
Relative Position [11]	30.4
BiGan [35]	34.8
Colorization [10]	35.2
Jigsaw [29] *	38.1
<b>Using ResNet-V2</b>	
Motion Segmentation [36]	27.6
Exemplar [36]	31.5
Relative Position [36]	36.2
Colorization [36]	39.6
<b>CPC</b>	<b>48.7</b>

Table 3: ImageNet top-1 unsupervised classification results. \*Jigsaw is not directly comparable to the other AlexNet results because of architectural differences.

- Compares favorably with other pretext task-based self-supervised learning method.
- Doesn't do as well compared to newer instance-based contrastive learning methods on image feature learning.



Source: [van den Oord et al., 2018](#),

# Summary: Contrastive Representation Learning

A general formulation for contrastive learning:

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

InfoNCE loss: N-way classification among positive and 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](#))

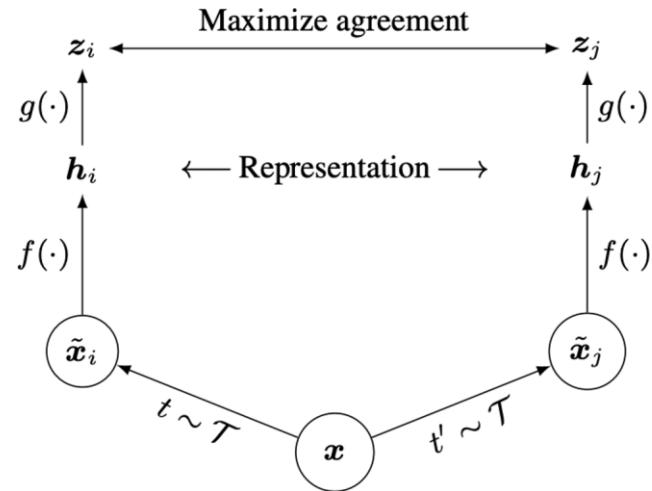
A *lower bound* on the mutual information between  $f(x)$  and  $f(x^+)$

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

# Summary: Contrastive Representation Learning

**SimCLR**: a simple framework for contrastive representation learning

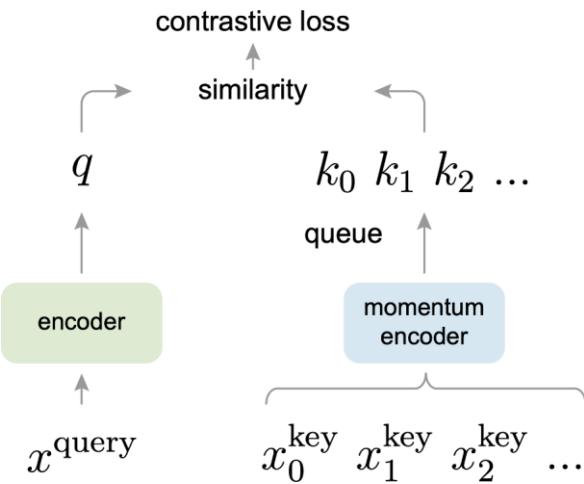
- **Key ideas**: non-linear projection head to allow flexible representation learning
- Simple to implement, effective in learning visual representation
- Requires large training batch size to be effective; large memory footprint



# Summary: Contrastive Representation Learning

**MoCo** (v1, v2): contrastive learning using momentum sample encoder

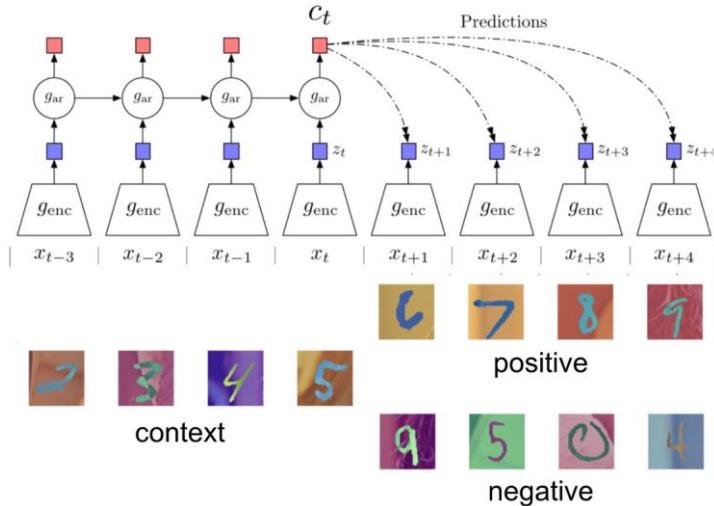
- Decouples negative sample size from minibatch size; allows large batch training without TPU
- MoCo-v2 combines the key ideas from SimCLR, i.e., nonlinear projection head, strong data augmentation, with momentum contrastive learning



# Summary: Contrastive Representation Learning

**CPC:** sequence-level contrastive learning

- Contrast “right” sequence with “wrong” sequence.
- InfoNCE loss with a time-dependent score function.
- Can be applied to a variety of learning problems, but not as effective in learning image representations compared to instance-level methods.



# Other examples: MoCo v3

## An Empirical Study of Training Self-Supervised Vision Transformers

“This paper does not describe a novel method.”

Xinlei Chen\* Saining Xie\* Kaiming He  
Facebook AI Research (FAIR)

Code: <https://github.com/facebookresearch/moco-v3>

### Abstract

*This paper does not describe a novel method. Instead, it studies a straightforward, incremental, yet must-know baseline given the recent progress in computer vision: self-supervised learning for Vision Transformers (ViT). While the training recipes for standard convolutional networks have been highly mature and robust, the recipes for ViT are yet to be built, especially in the self-supervised scenarios where training becomes more challenging. In this work, we go back to basics and investigate the effects of several fundamental components for training self-supervised ViT. We observe that instability is a major issue that degrades accuracy, and it can be hidden by apparently good results. We reveal that these results are indeed partial failure, and they can be improved when training is made more stable. We benchmark ViT results in MoCo v3 and several other self-supervised frameworks, with ablations in various aspects. We discuss the currently positive evidence as well as challenges and open questions. We hope that this work will provide useful data points and experience for future research.*

framework	model	params	acc. (%)
<i>linear probing:</i>			
iGPT [9]	iGPT-L	1362M	69.0
iGPT [9]	iGPT-XL	6801M	72.0
MoCo v3	ViT-B	86M	76.7
MoCo v3	ViT-L	304M	77.6
MoCo v3	ViT-H	632M	78.1
MoCo v3	ViT-BN-H	632M	79.1
MoCo v3	ViT-BN-L/7	304M	<b>81.0</b>
<i>end-to-end fine-tuning:</i>			
masked patch pred. [16]	ViT-B	86M	79.9 <sup>†</sup>
MoCo v3	ViT-B	86M	83.2
MoCo v3	ViT-L	304M	<b>84.1</b>

Table 1. **State-of-the-art Self-supervised Transformers** in ImageNet classification, evaluated by linear probing (top panel) or end-to-end fine-tuning (bottom panel). Both iGPT [9] and masked patch prediction [16] belong to the masked auto-encoding paradigm. MoCo v3 is a contrastive learning method that compares two ( $224 \times 224$ ) crops. ViT-B, -L, -H are the Vision Transformers proposed in [16]. ViT-BN is modified with BatchNorm, and “77” denotes a patch size of  $7 \times 7$ . <sup>†</sup>: pre-trained in JFT-300M.

# Other examples: Masked Autoencoder

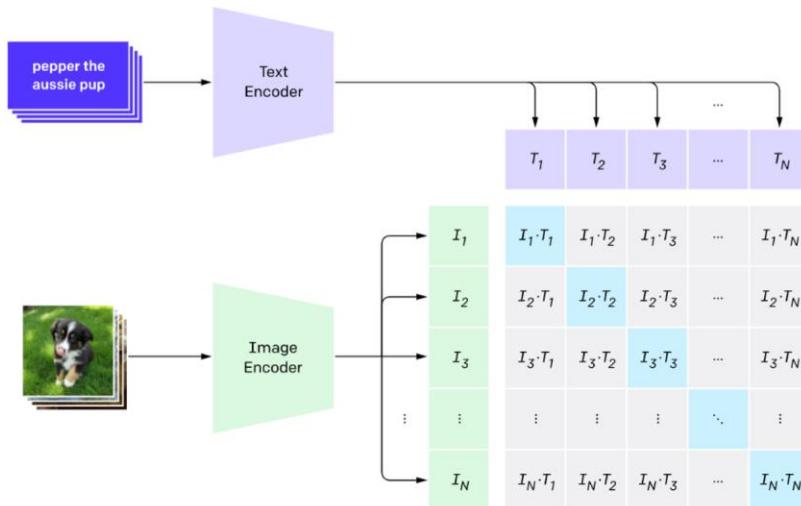


method	pre-train data	ViT-B	ViT-L	ViT-H	ViT-H <sub>448</sub>
scratch, our impl.	-	82.3	82.6	83.1	-
DINO [5]	IN1K	82.8	-	-	-
MoCo v3 [9]	IN1K	83.2	84.1	-	-
BEiT [2]	IN1K+DALLE	83.2	85.2	-	-
MAE	IN1K	<u>83.6</u>	<u>85.9</u>	<u>86.9</u>	<b>87.8</b>

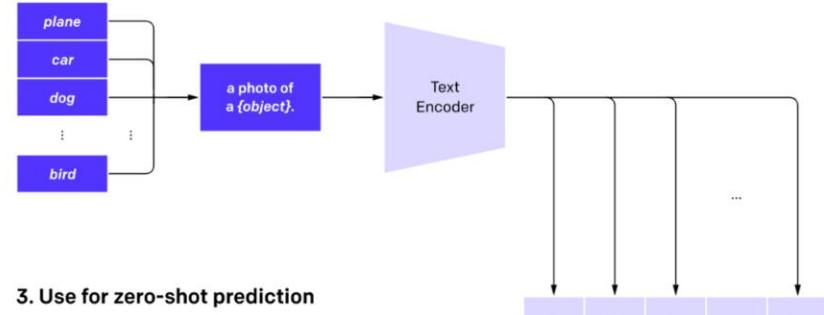
# Other examples: CLIP

Contrastive learning between image and natural language sentences

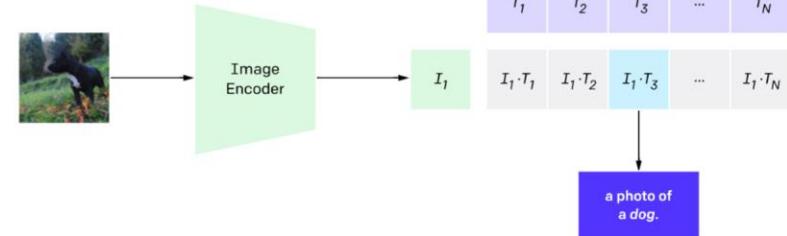
## 1. Contrastive pre-training



## 2. Create dataset classifier from label text



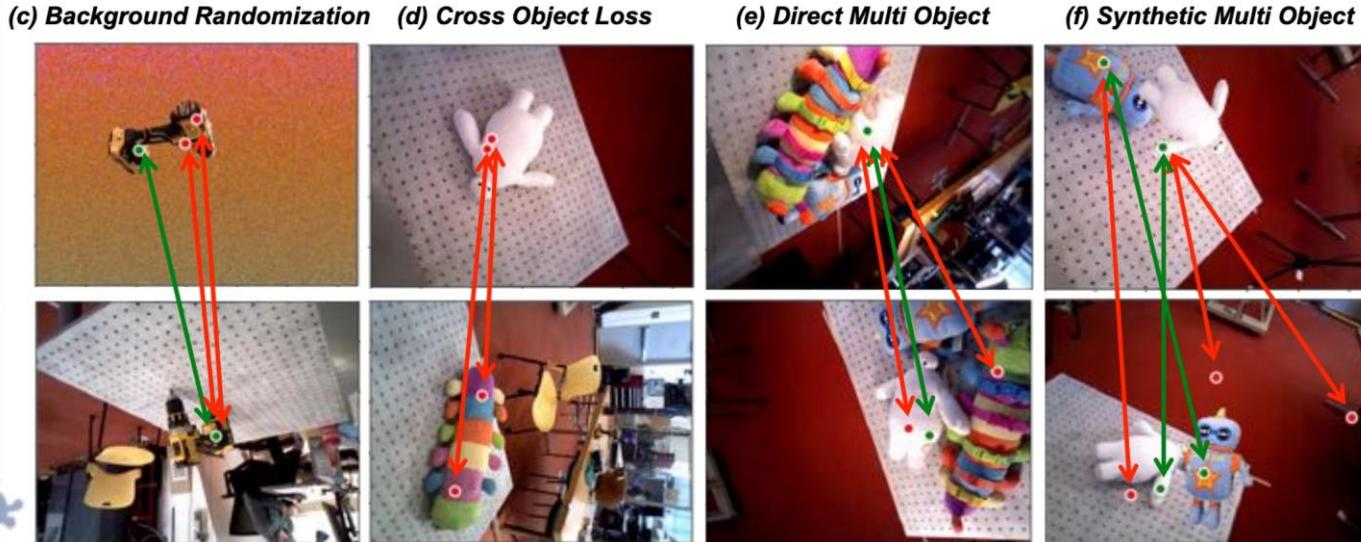
## 3. Use for zero-shot prediction



CLIP (*Contrastive Language–Image Pre-training*) Radford *et al.*, 2021

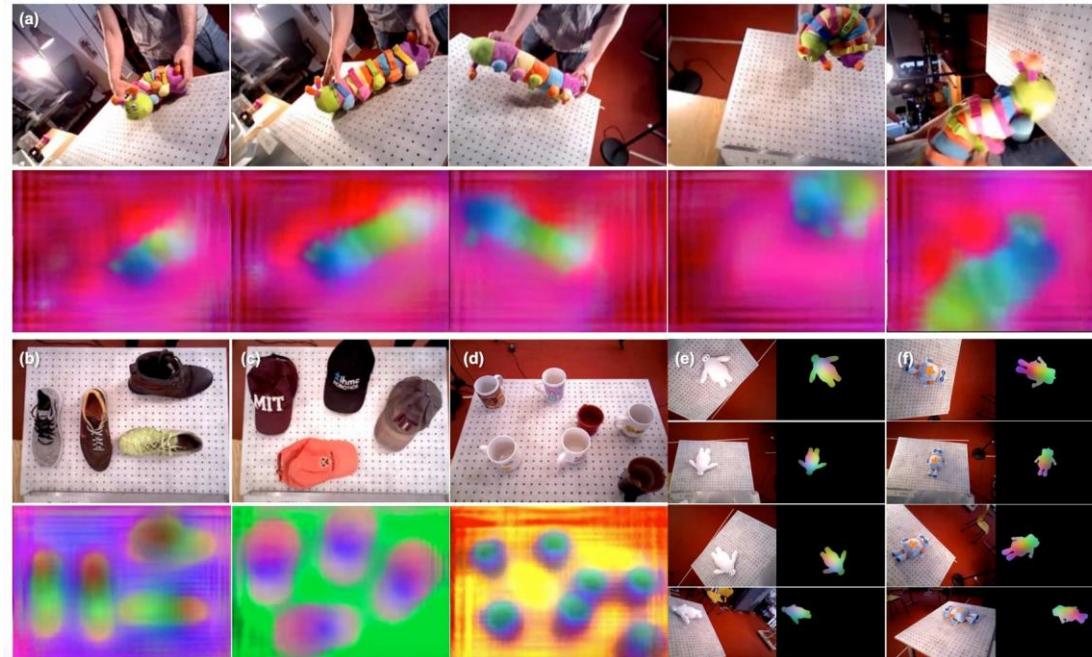
# Other examples: Dense Object Net

Contrastive learning on pixel-wise feature descriptors



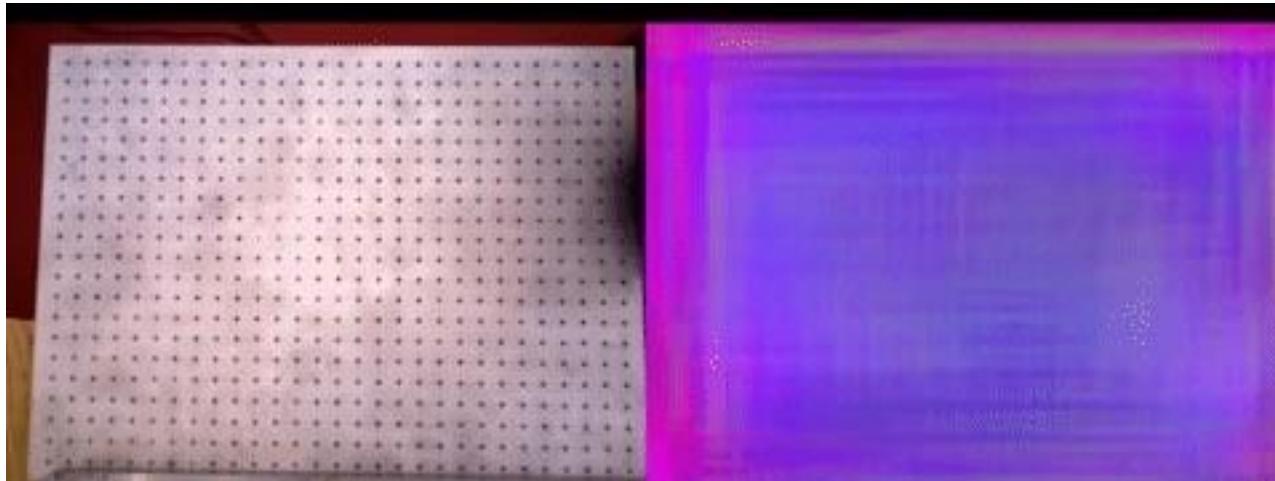
Dense Object Net, Florence et al., 2018

# Other examples: Dense Object Net



Dense Object Net, Florence et al., 2018

# Other examples: Dense Object Net



Dense Object Net, Florence et al., 2018

# Other examples: DINO

## Emerging Properties in Self-Supervised Vision Transformers

Mathilde Caron<sup>1,2</sup>

Hugo Touvron<sup>1,3</sup>

Ishan Misra<sup>1</sup>

Hervé Jegou<sup>1</sup>

Julien Mairal<sup>2</sup>

Piotr Bojanowski<sup>1</sup>

Armand Joulin<sup>1</sup>

<sup>1</sup> Facebook AI Research

<sup>2</sup> Inria\*

<sup>3</sup> 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.

# Other examples: DINO v2

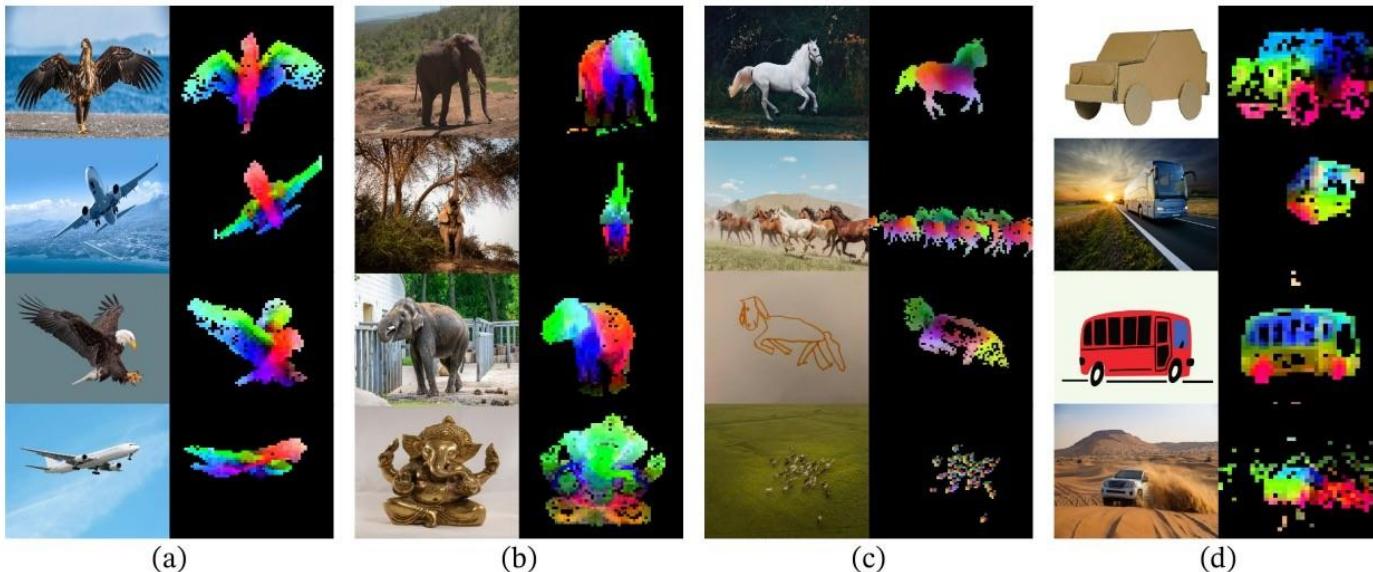
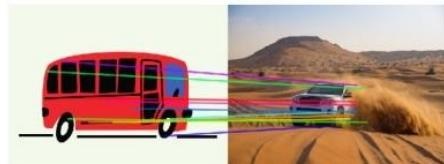
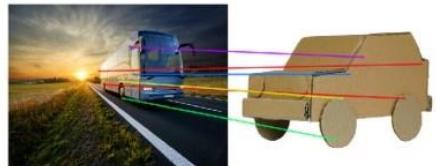
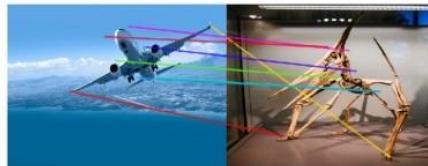


Figure 1: **Visualization of the first PCA components.** We compute a PCA between the patches of the images from the same column (a, b, c and d) and show their first 3 components. Each component is matched to a different color channel. Same parts are matched between related images despite changes of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

# Other examples: DINO v2



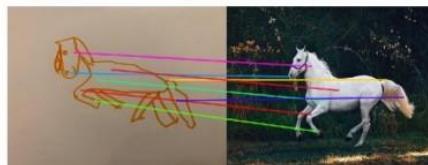
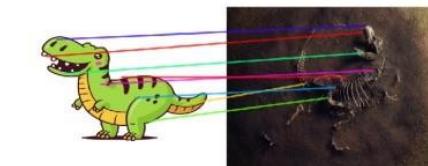
(Vehicles)



(Birds / Airplanes)



(Elephants)



(Drawings / Animals)

# Other examples: DINO v2

