Lecture 12: Self-Supervised Learning

Administrative

- Midterm next Tuesday (5/12 during lecture time) – Don’t be late; Look for Ed posts for logistic details.
- Midterm review section tomorrow, Friday, 5/9 at 12:30 pm.
- Project proposal feedback is available on Gradescope.

Administrative

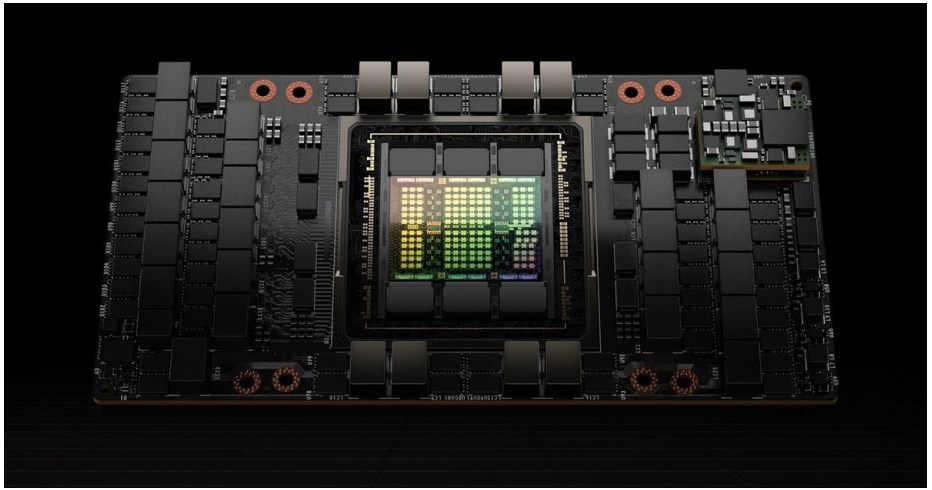
- Cloud Credits for Projects - **$350**
  - $200 Modal
  - $100 AWS
  - $50 GCP

Student Spotlights

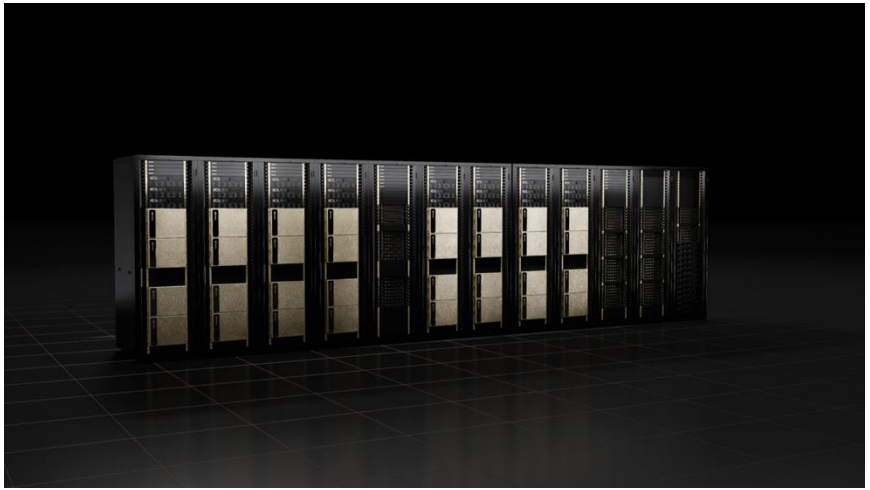
- Thank you for your engagement and improving the course experience!
  - Eyas Taifour (Ed, Ed2)
  - Sky Wang (Ed, Ed2)
  - Cheng-Suen Stephen Chan (Ed)
  - Anonymous (Ed)
  - Jeremy Hsieh (Ed)
  - Jan Kopanski (Ed)
  - Benedito Faustinoni Neto (Ed)
  - Andrei Simion (Ed)
  - Warren Chan
  - Yuliia Murakami
  - Adem Rimpapa
  - Arunabh Sharma
  - Timothy Yu

Previous Lecture: GPUs and How to Train On Them

A bit about GPU hardware



How to train on lots of GPUs



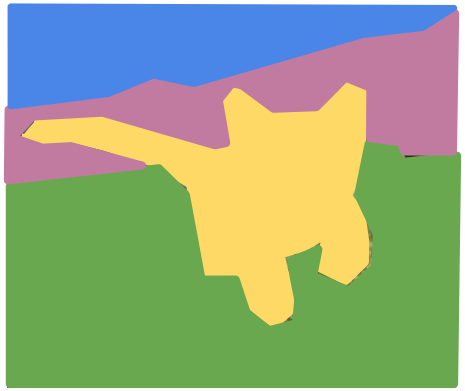
Lots of Computer Vision Tasks

Classification

CAT

No spatial extent

Semantic Segmentation



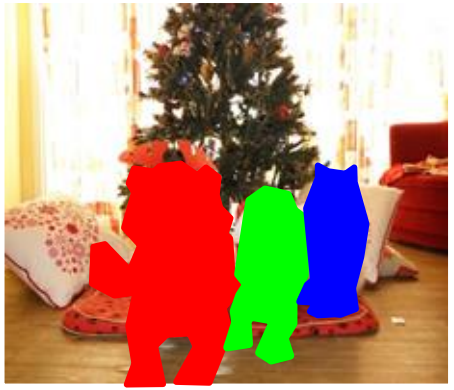
GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

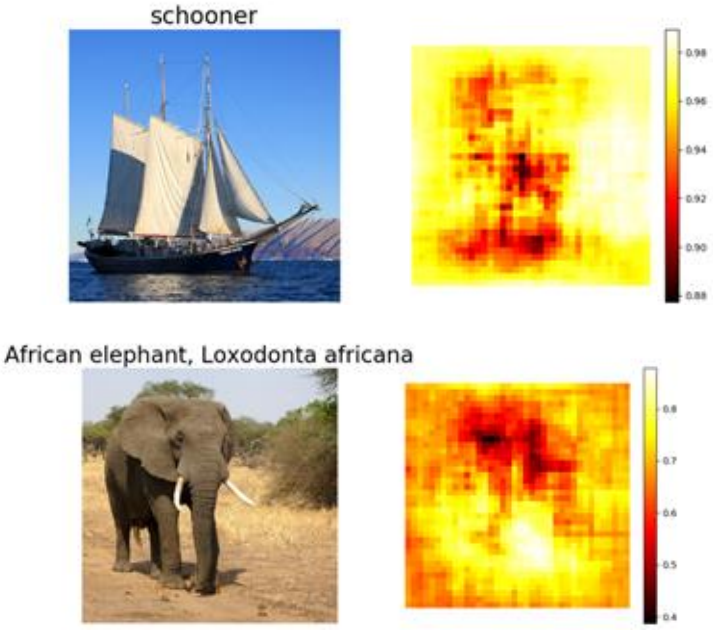
Instance Segmentation

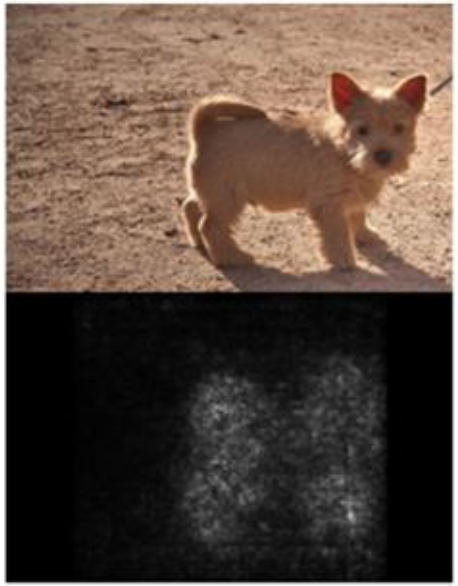


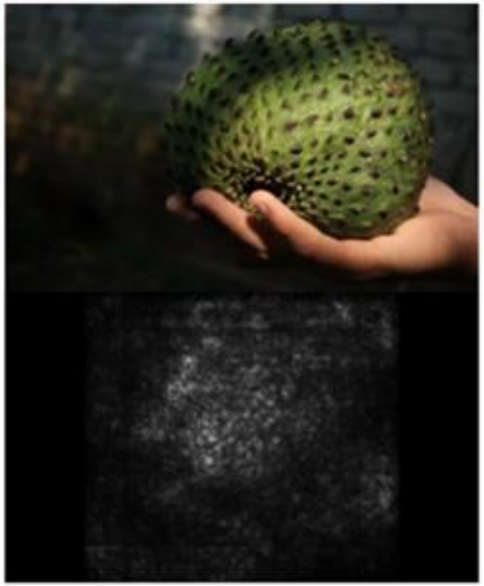
DOG, DOG, CAT

Multiple Object

Last Week: Visualizing and Understanding







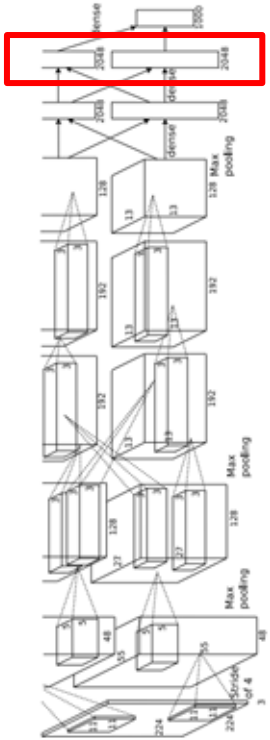
Last Week: Visualizing and Understanding

Recall: Nearest neighbors in pixel space

Test image

L2 Nearest neighbors in feature space

4096-dim vector



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

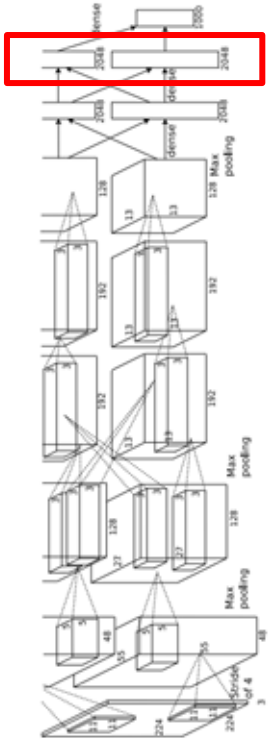
Learned Representations

Recall: Nearest neighbors in pixel space

Test image

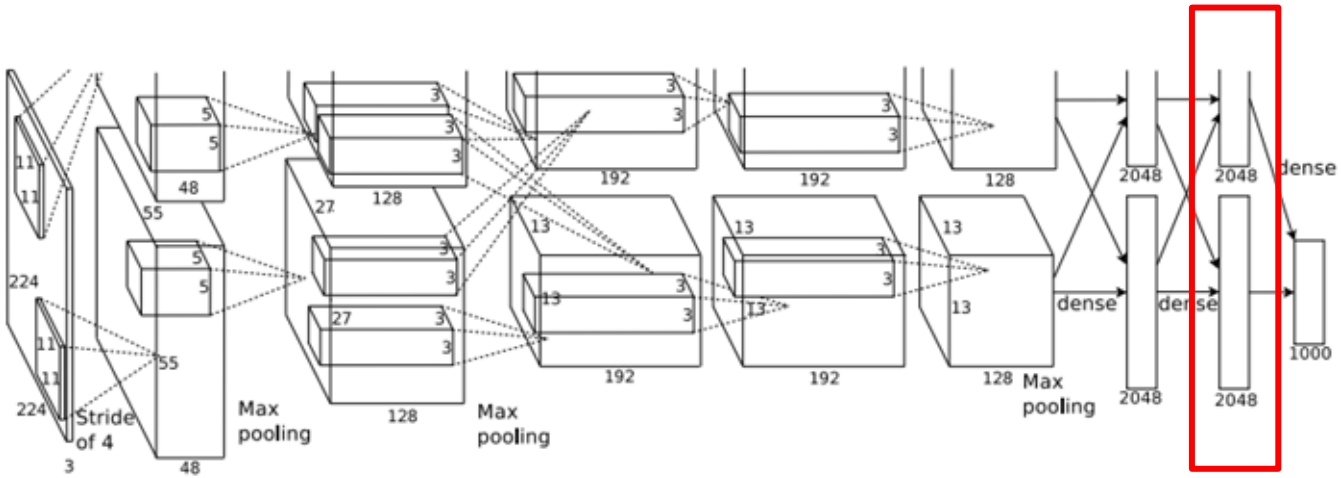
L2 Nearest neighbors in feature space

4096-dim vector



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

Learned Representations



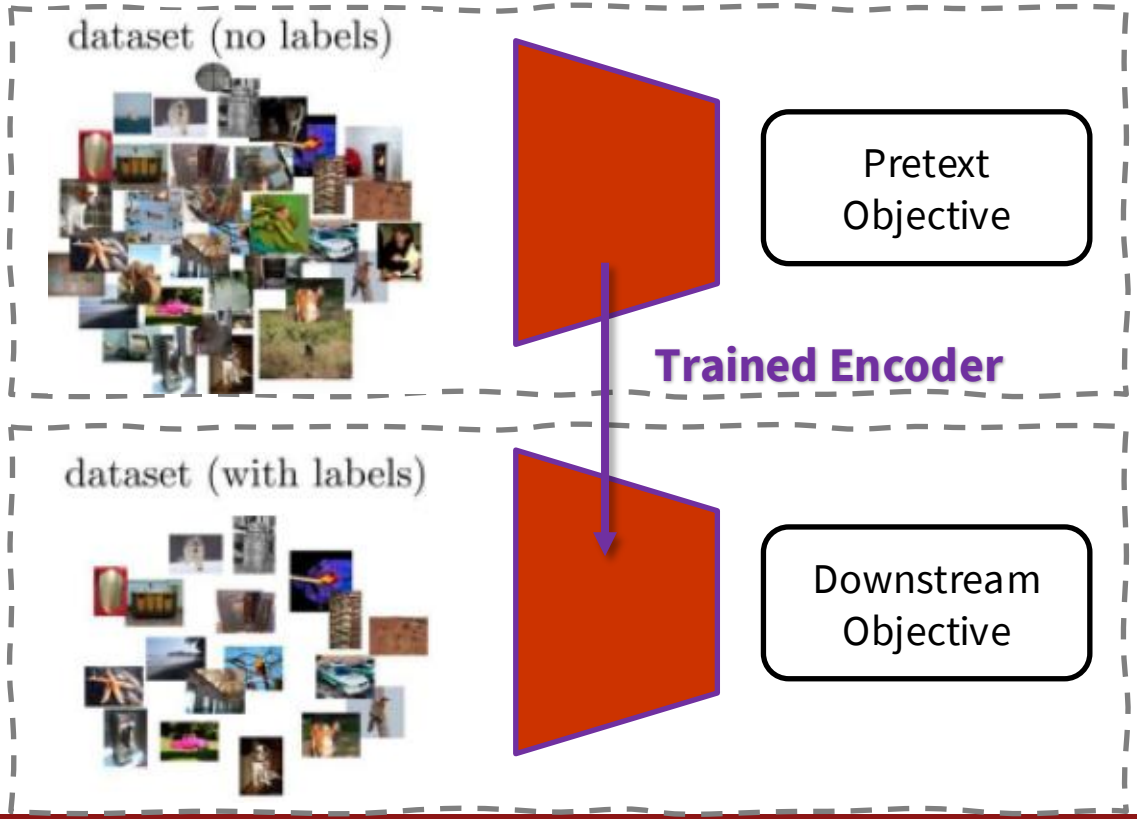
What is the problem with large-scale training?

- **We need a lot of labeled data**

Is there a way we can train neural networks without the need for huge manually labeled datasets?

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

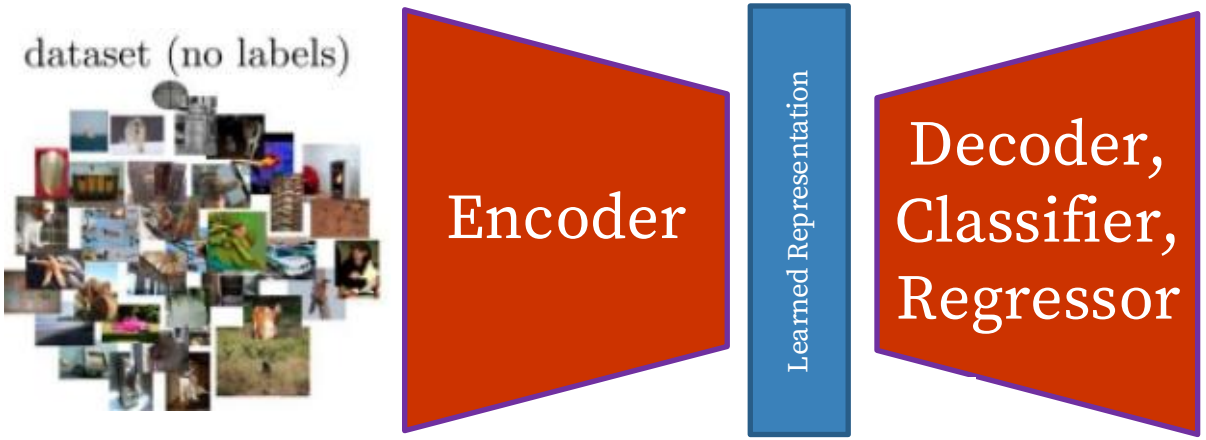
Self-Supervised Learning



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

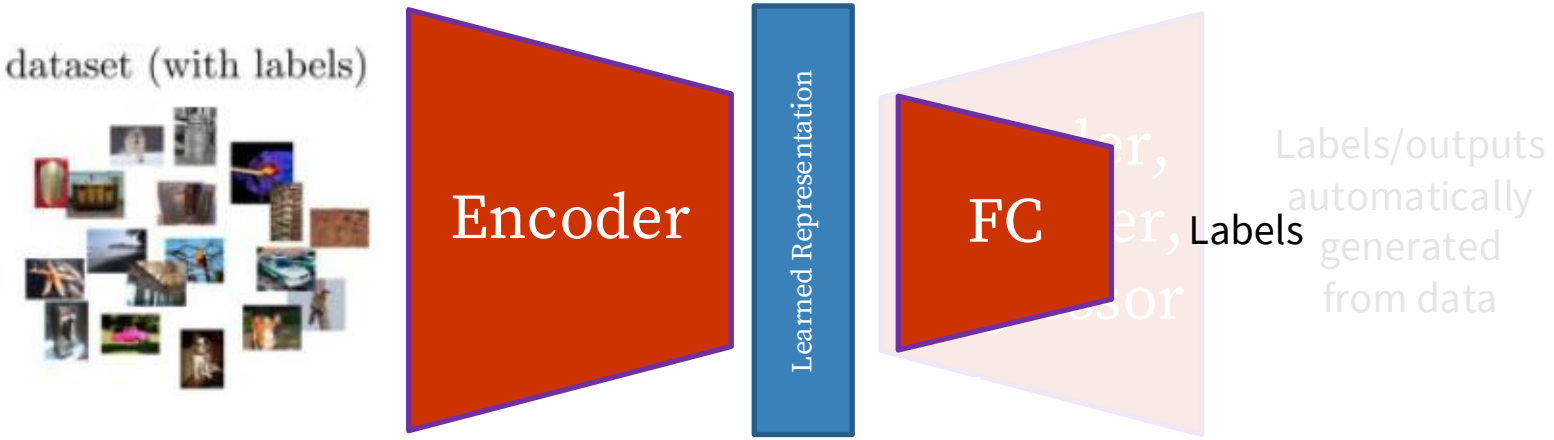
Downstream Task
- The application you care about
- You do not have large datasets
- The dataset is labeled

Self-Supervised Learning – Pretext Task



Labels/outputs automatically generated from data

Self-Supervised Learning – Downstream Task (supervised)



Semi-supervised

Self-supervised pretext tasks

Example: learn to predict image transformations / complete corrupted images

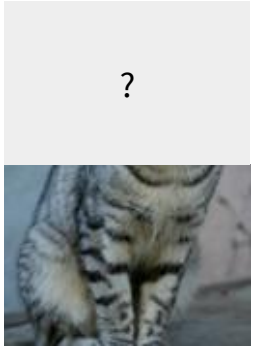
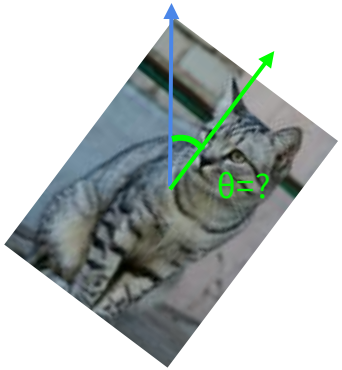


image completion



rotation prediction

“jigsaw puzzle”

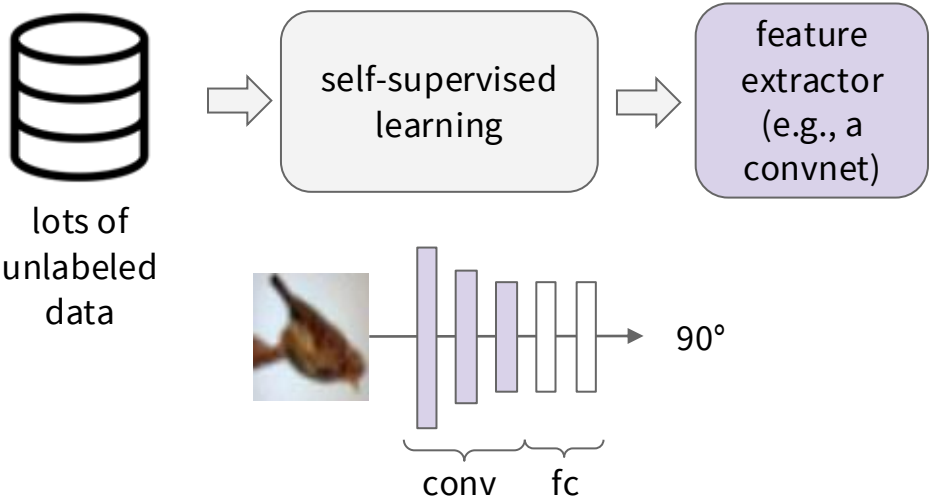
colorization

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

How to evaluate a self-supervised learning method?

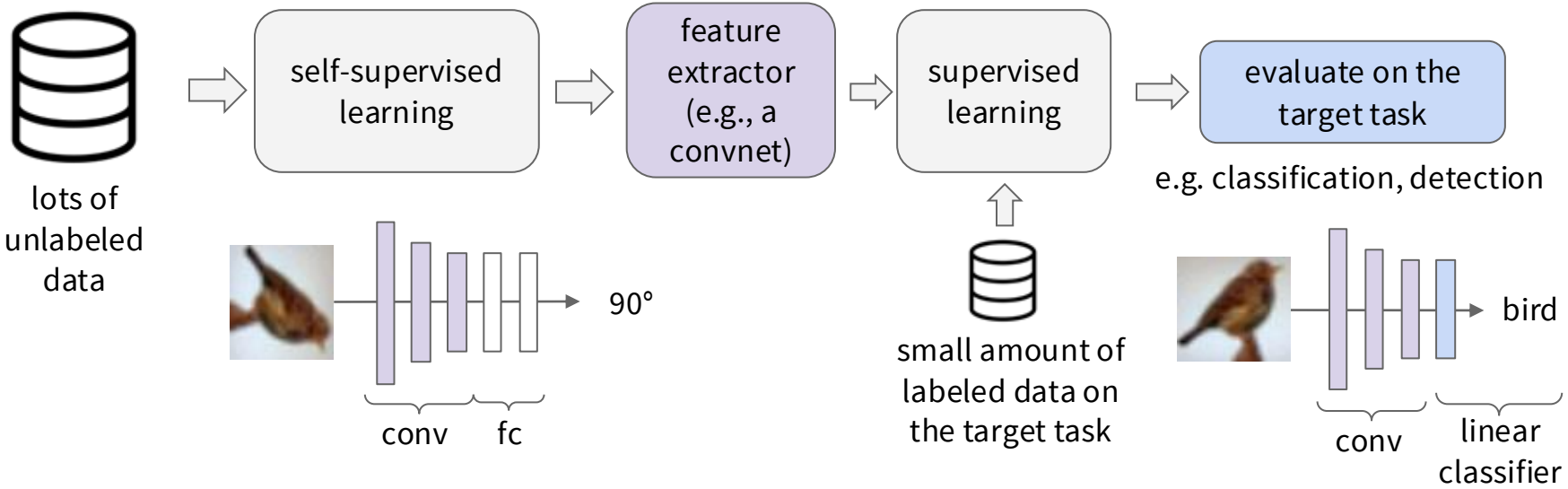
- **Pretext Task Performance**
  - Measure how well the model performs on the task it was trained on without labels.
- **Representation Quality**
  - Evaluate the quality of the learned representations
    - *Linear Evaluation Protocol:* Train a linear classifier on the leaerned representations;
   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

How to evaluate 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?



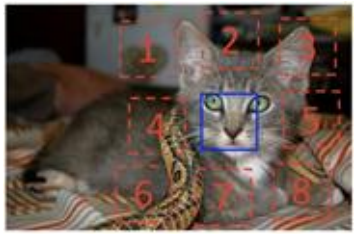
1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

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

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)

robot / reinforcement learning

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

• • •

Today's Agenda

Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring
- Reconstruction-based learning (MAE)

Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC
- Self-Distillation Without Labels, DINO

Today's Agenda

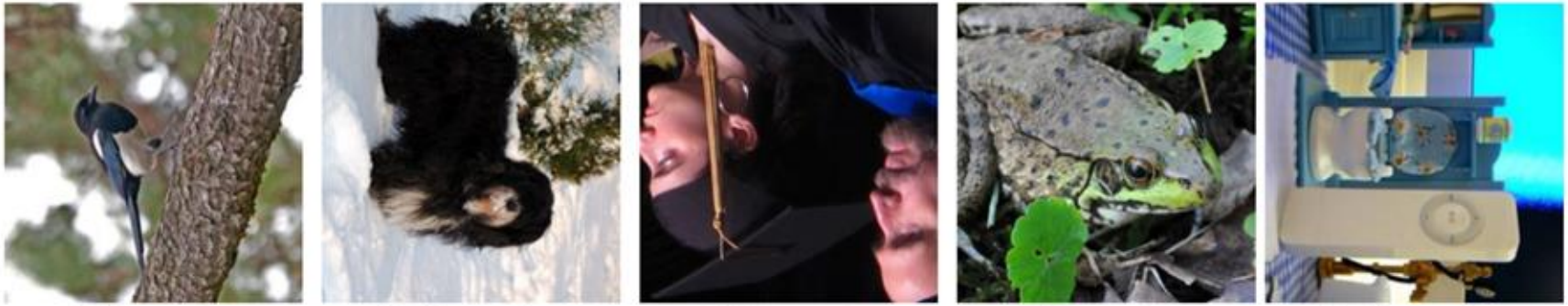
Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring
- Reconstruction-based learning (MAE)

Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC
- Self-Distillation Without Labels, DINO

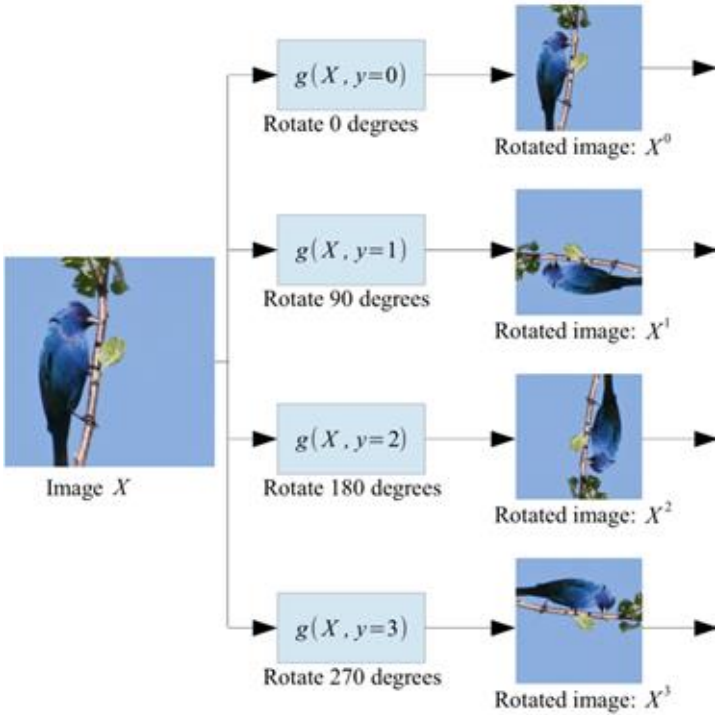
Pretext task: predict rotations



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

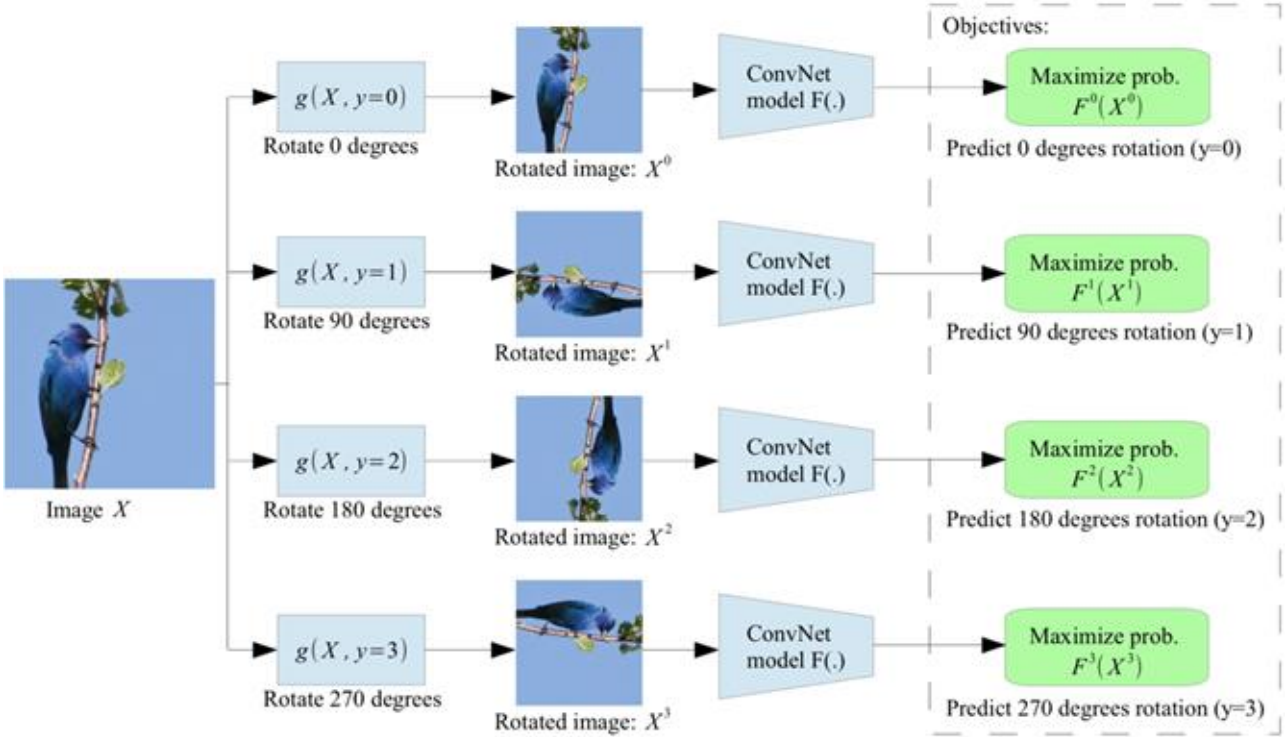


Self-supervised learning by rotating the entire input images.

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

(Image source: Gidaris et al. 2018)

Pretext task: predict rotations

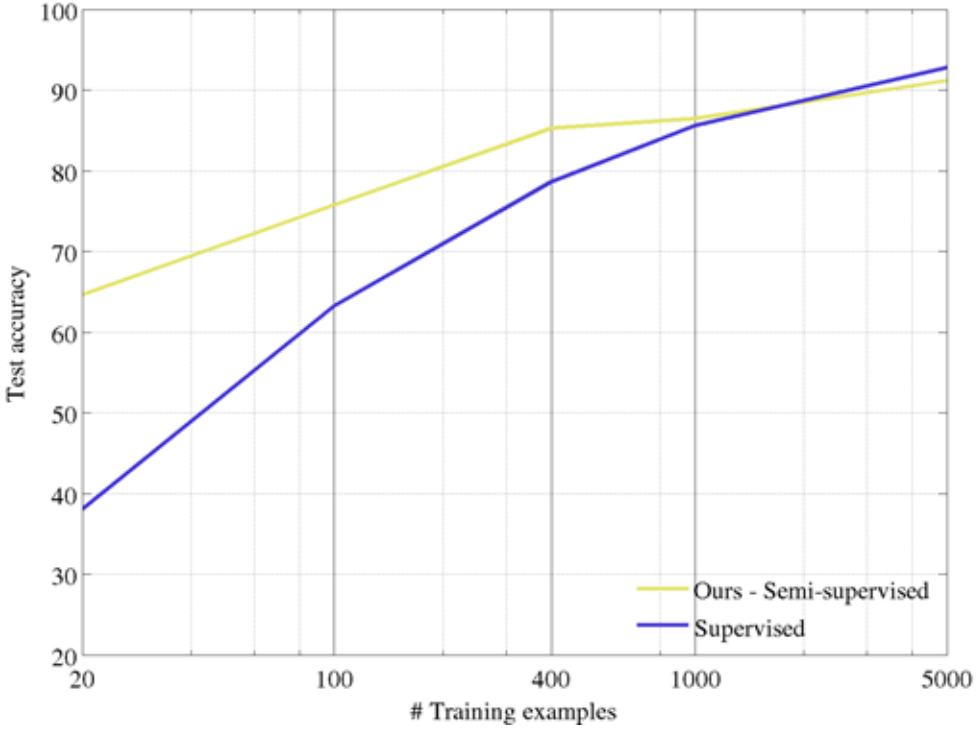


Self-supervised learning by rotating the entire input images.

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

Pretrained with full ImageNet supervision

No pretraining

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

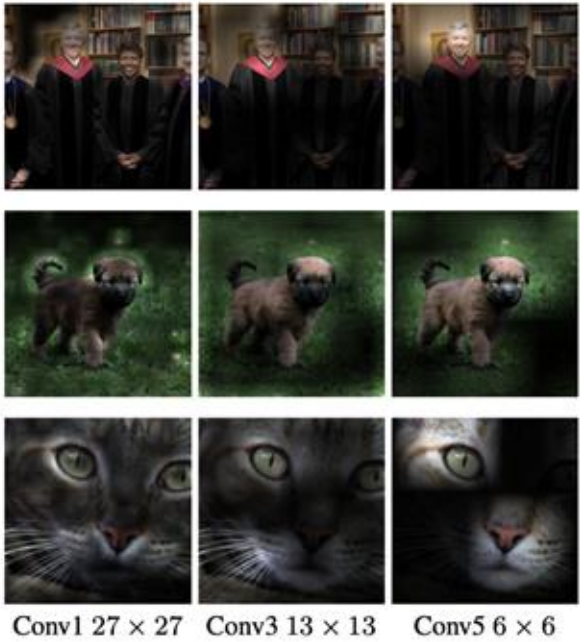
Finetune on labeled data from Pascal VOC 2007.

Self-supervised learning with rotation prediction

source: Gidaris et al. 2018

Visualize learned visual attentions

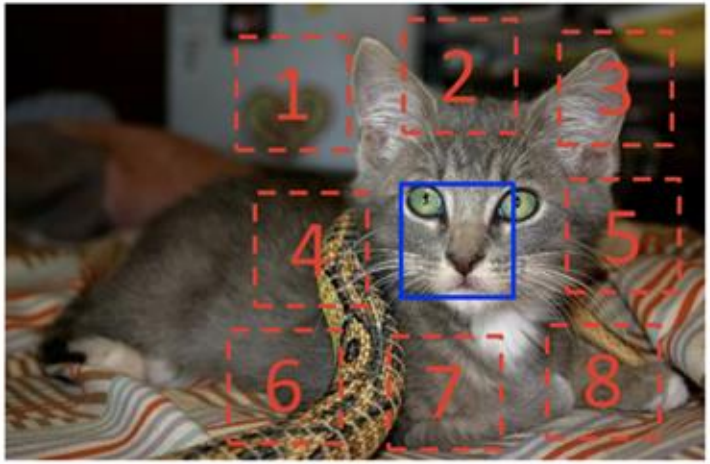
(a) **Attention maps of supervised model**



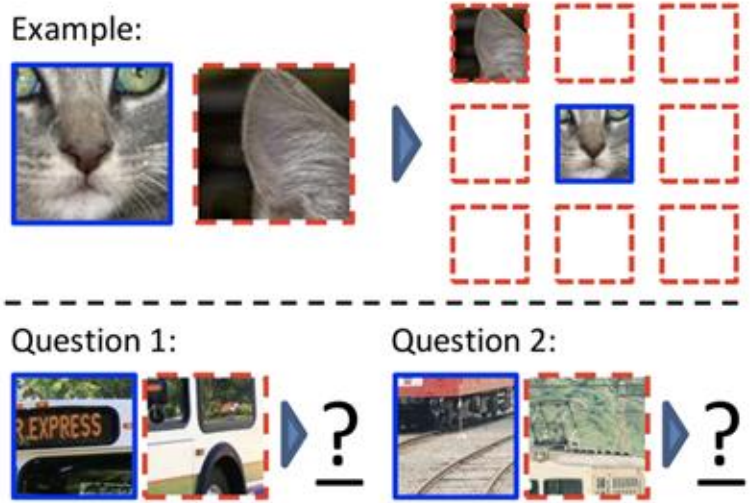
(b) **Attention maps of our self-supervised model**

(Image source: Gidaris et al. 2018)

Pretext task: predict relative patch locations

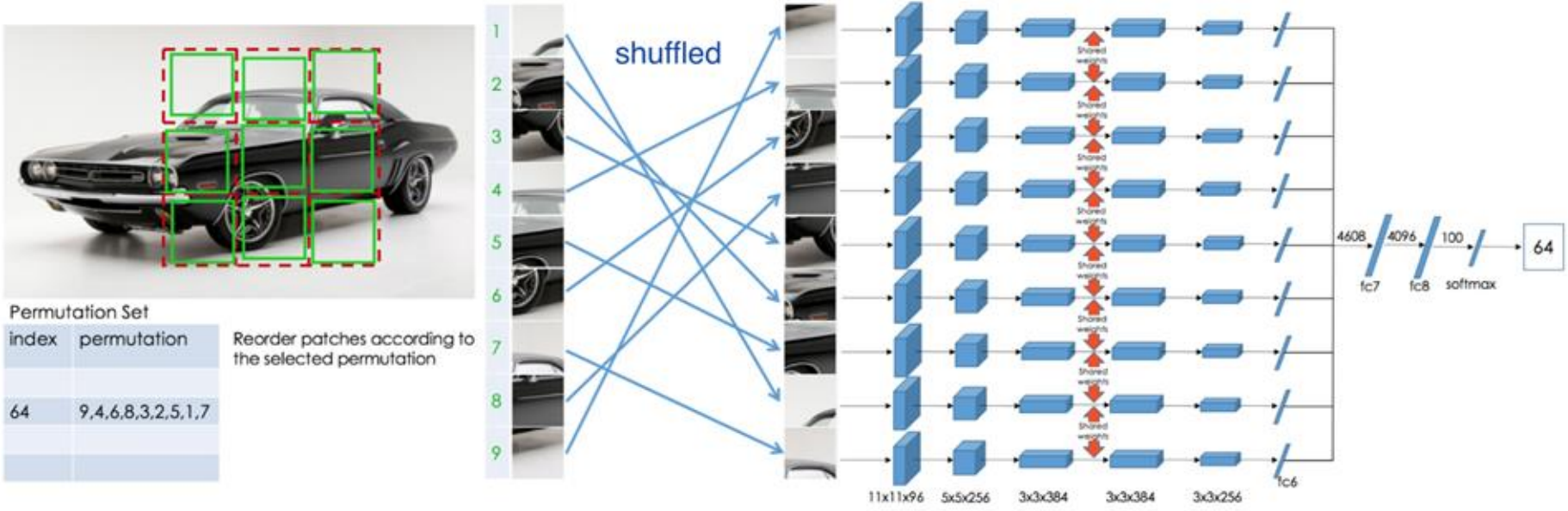


X = (,); Y = 3



(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*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%**

"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)

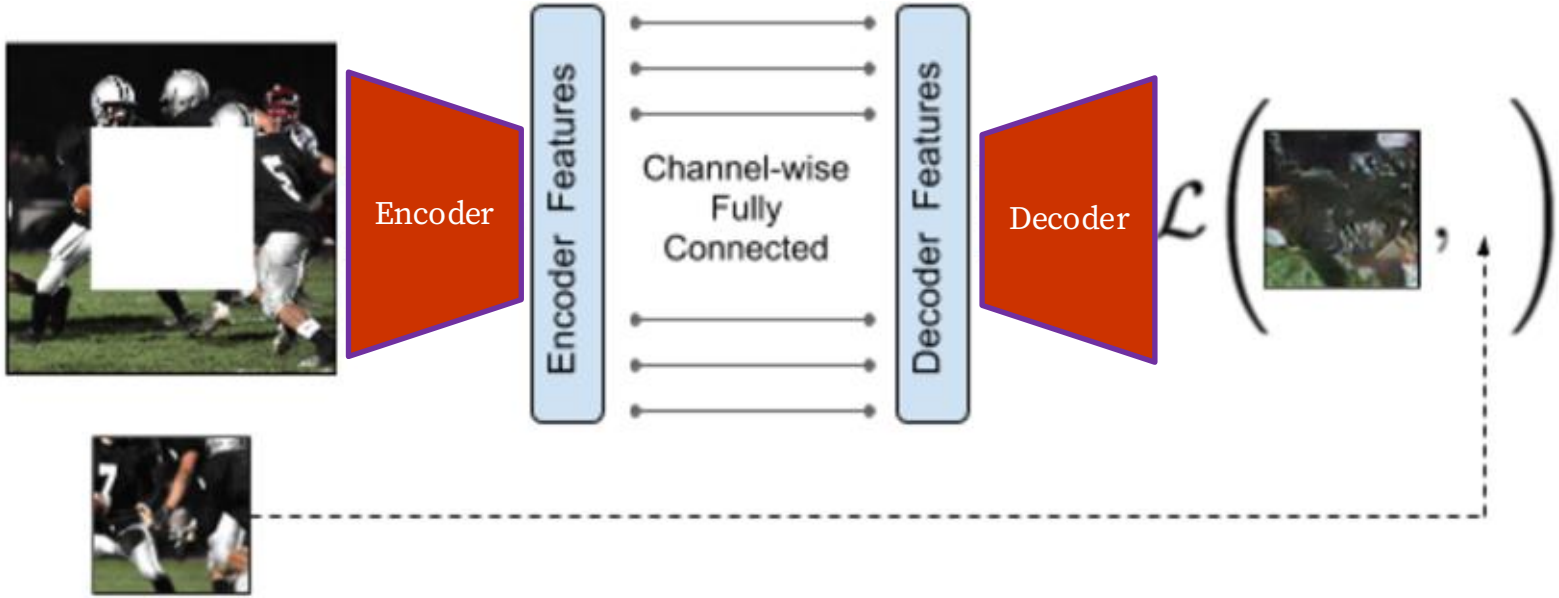
Pretext task: predict missing pixels (inpainting)

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

Source: Pathak et al., 2016

Learning to inpaint by reconstruction

Auto Encoder



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

(We will talk about adversarial learning in the next lecture)

Loss = reconstruction + adversarial learning

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

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

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

Element wise multiplication

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

Encoder

Adversarial loss between “real” images and inpainted images

Source: Pathak et al., 2016

Inpainting evaluation

Input (context) reconstruction adversarial recon + adv

Source: Pathak et al., 2016

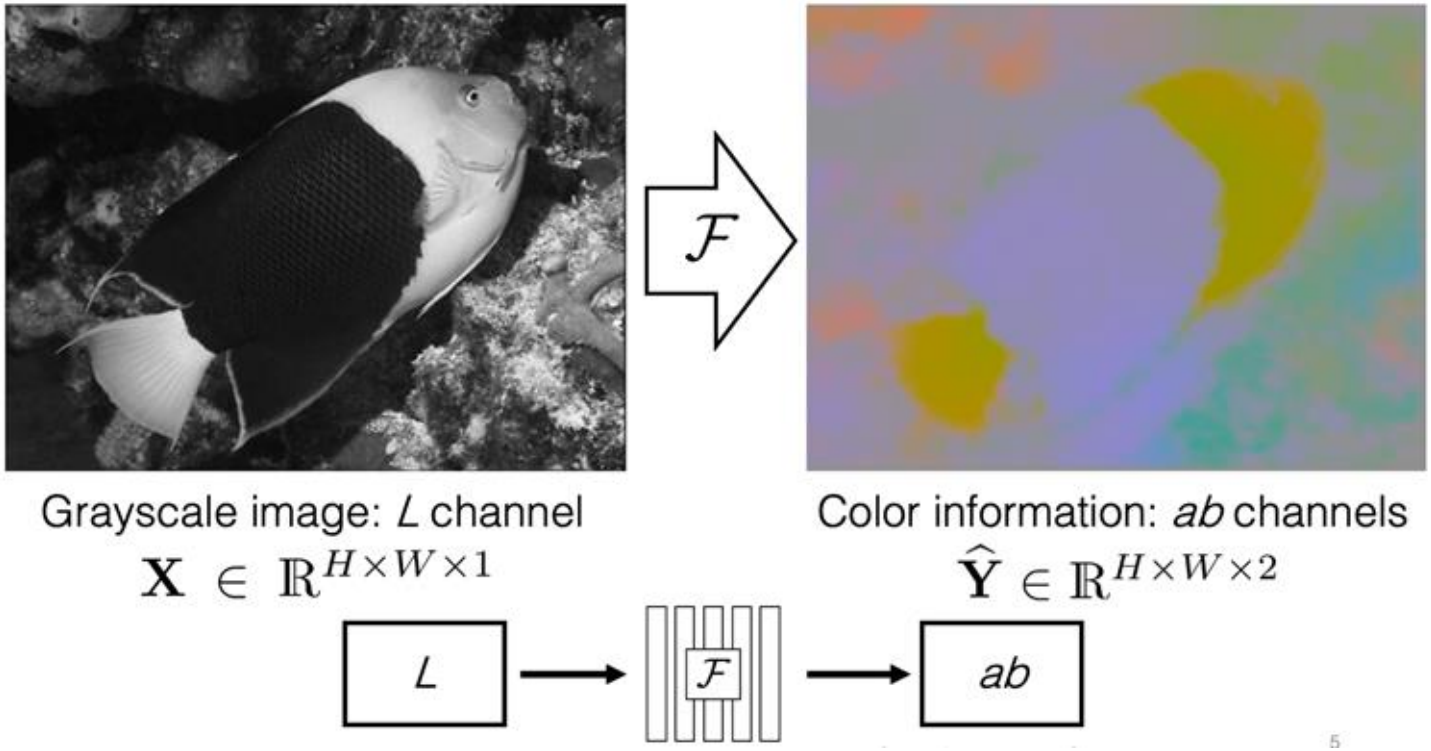
Transfer learned features to supervised learning

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

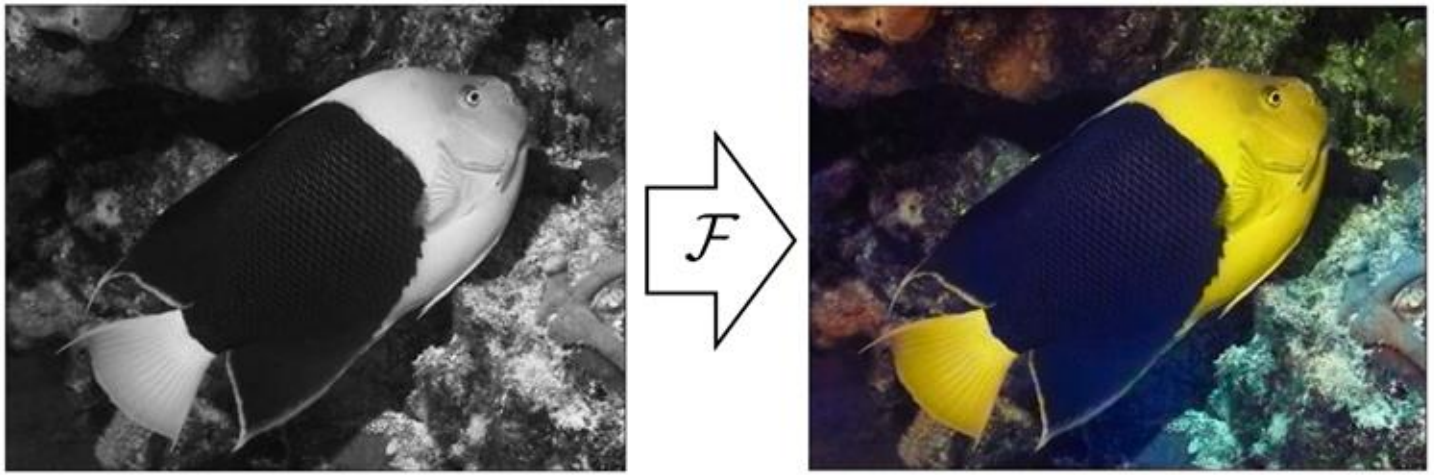
Source: Pathak et al., 2016

Pretext task: image coloring



Source: Richard Zhang / Phillip Isola

Pretext task: image coloring



Grayscale image: L channel

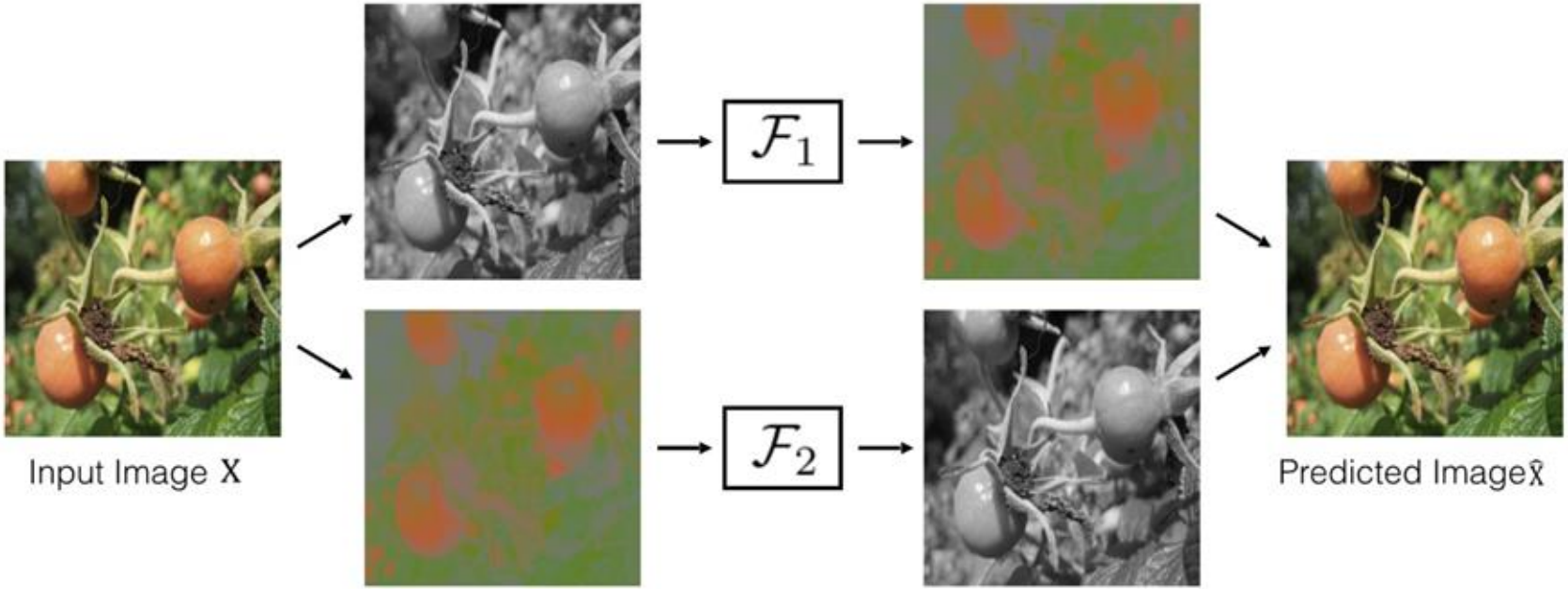
$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$

Concatenate (L,ab) channels

$(\mathbf{X}, \hat{\mathbf{Y}})$

Source: Richard Zhang / Phillip Isola

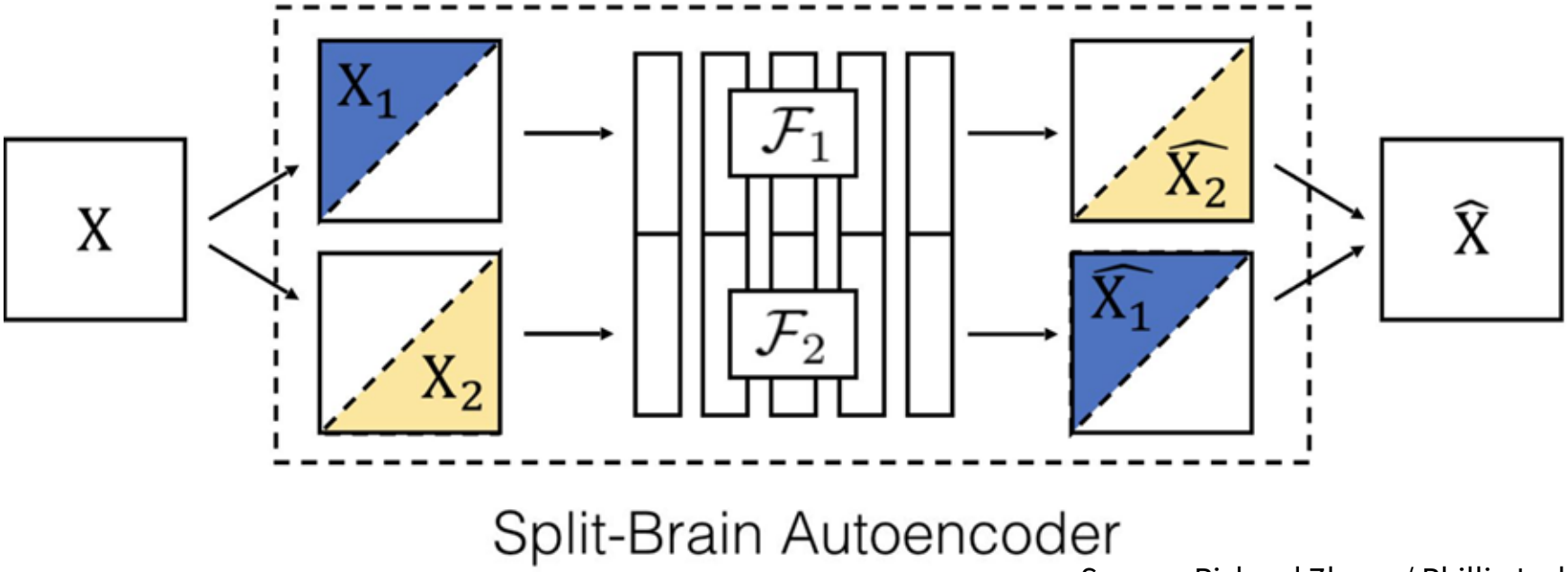
Learning features from colorization: Split-brain Autoencoder



Source: Richard Zhang / Phillip Isola

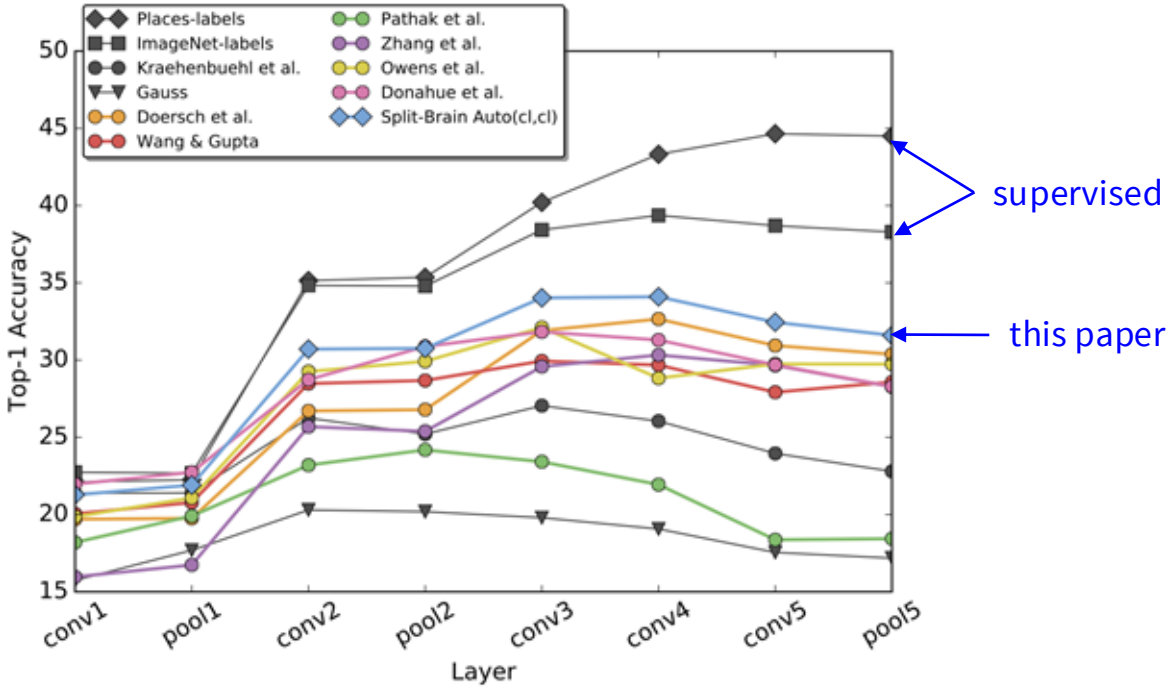
Learning features from colorization: Split-brain Autoencoder

Idea: cross-channel predictions



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

how should I color these frames?

t = 0

t = 1

t = 2

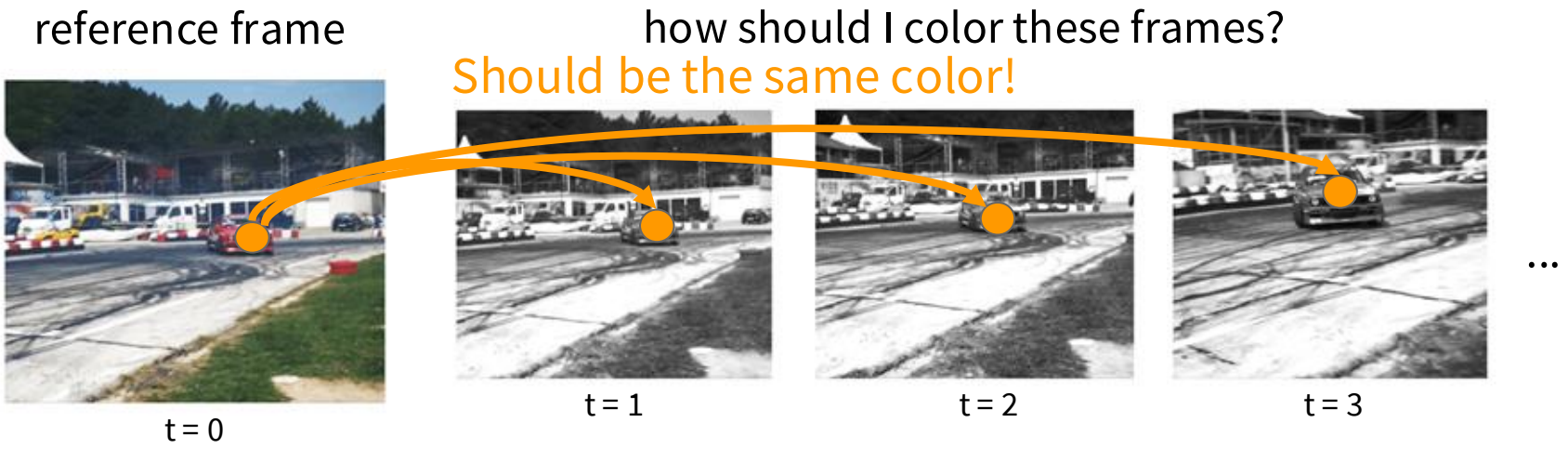
t = 3

...

Source: Vondrick et al., 2018

Pretext task: video coloring

Idea: model the temporal coherence of colors in videos



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

Source: Vondrick et al., 2018

Colorizing videos (qualitative)

reference frame

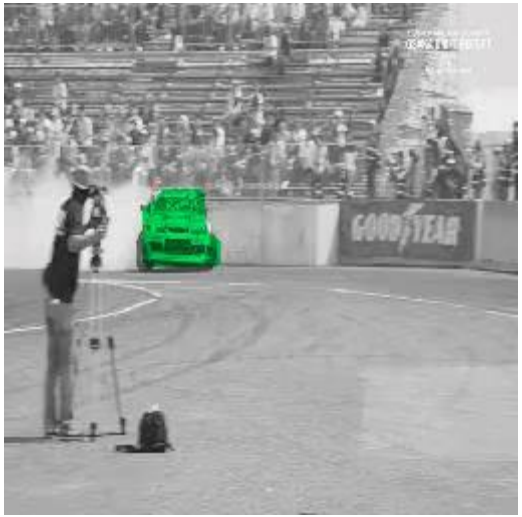
target frames (gray)

predicted color

Source: Google AI blog post

Tracking emerges from colorization

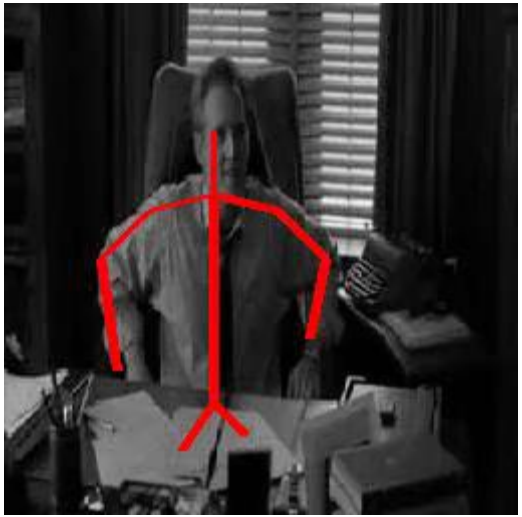
Propagate segmentation masks using learned attention



Source: Google AI blog post

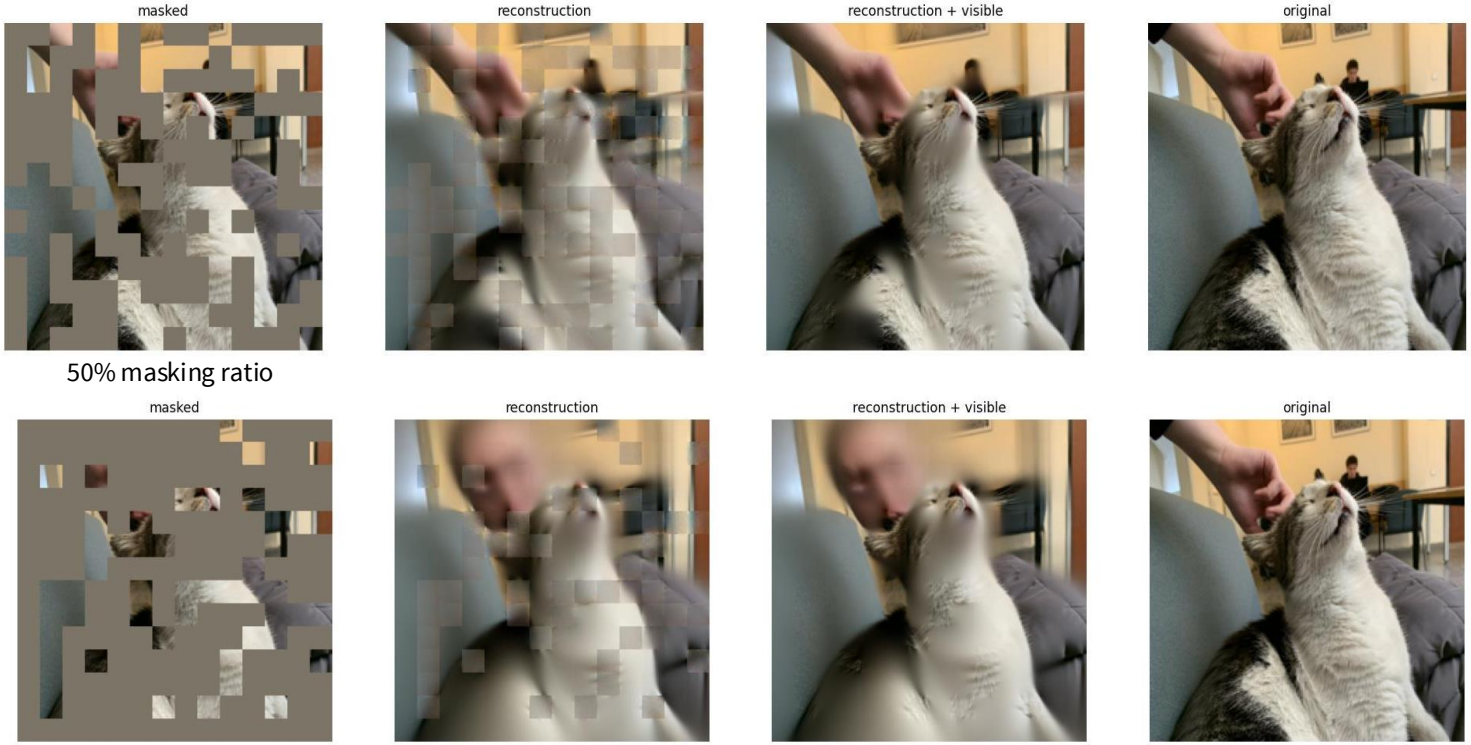
Tracking emerges from colorization

Propagate pose keypoints using learned attention



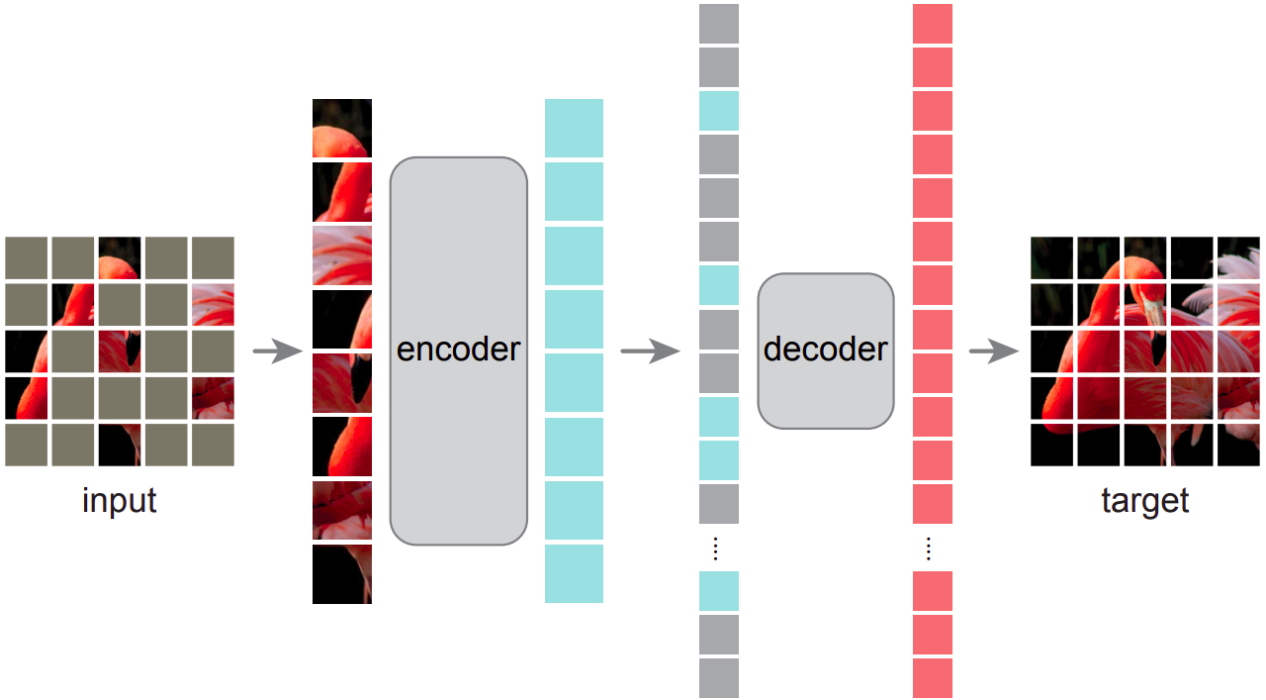
Source: Google AI blog post

Masked Auto Encoders (MAE)
Reconstruction with a larger masked portion



He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

Masked Auto Encoders (MAE)



He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

Masking Methods

- Similar to the original ViT,
  divide the input into non-overlapping patches.
- Uniformly sample a very large proportion (75%) of these patches and mask them
- Masking a high ratio makes predicting the task challenging and meaningful.
- Also, not using mask tokens and picking a high sampling ratio (masking most of the image, e.g., 75%, and sampling a small visible portion, e.g., 25%, to feed into the encoder) enables the encoder to be very large.

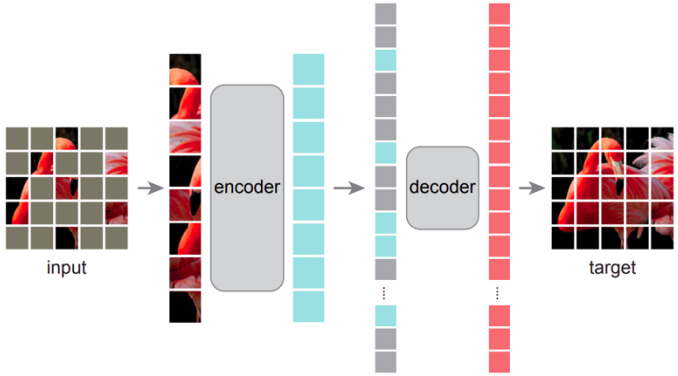


Figure 6. MAE architecture from the paper

He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

MAE Encoder

- The encoder only operates on unmasked patches (25%)
- Embeds the patches by linear projection and add positional embeddings
- Uses transformer blocks
- Since the input patches are a small part of the input, the encoder is chosen to be very large. (encoder has over 9 times computations per token vs decoder)

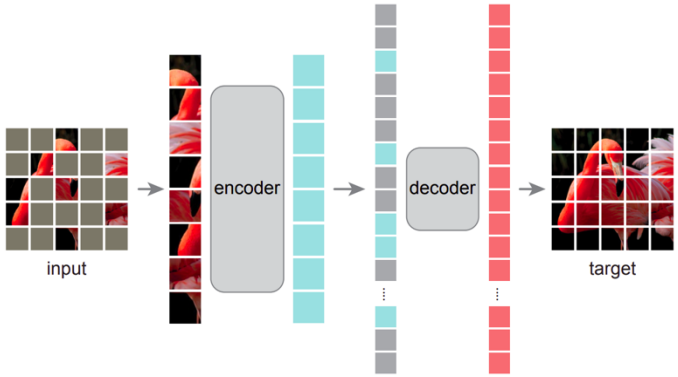


Figure 6. MAE architecture from the paper

He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

MAE Decoder

- Merges the encoder outputs with the shared mask tokens in previously masked places, adding positional encodings to them.
- Uses transformer blocks, followed by a linear projection for finalizing pixel reconstruction.
- Is solely responsible for reconstruction, meaning it is not used post-training. Hence, it is independent of the encoder design, making it flexible (unlike traditional AEs or UNet). This is an asymmetrical autoencoder design.

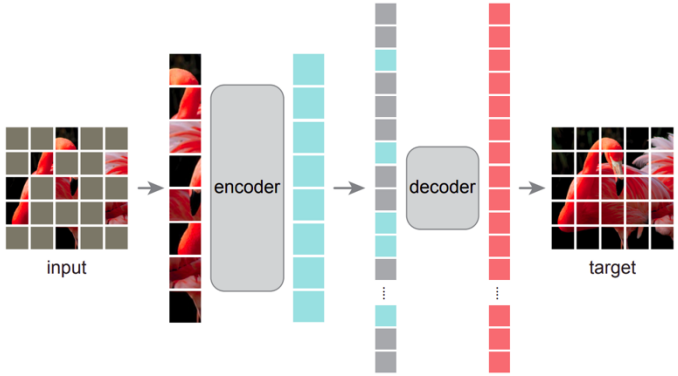


Figure 6. MAE architecture from the paper

He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

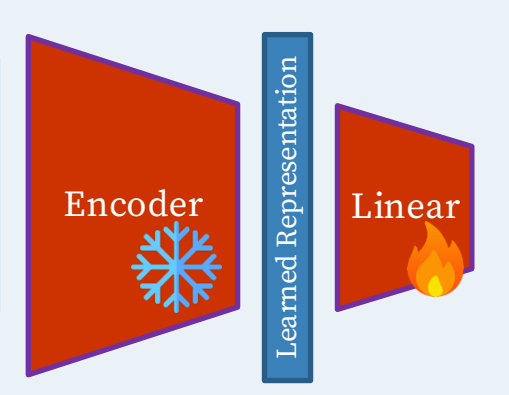
Reconstruction

- The MSE (mean squared error loss) in the pixel space between the input image and the reconstructed image is adopted.
- Loss is only computed for masked patches

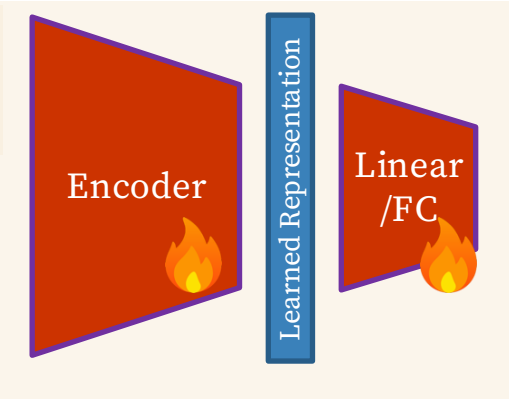
He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

Linear Probing vs Full Fine-tuning

- In linear probing, the pre-trained model is fixed, and only one linear layer is added at the end, to predict the labels (or produce the output). This method is used to assess the quality of representations from a pre-trained feature extraction model.



- In fine-tuning, pre-trained model is further trained (not fixed), and one or more layers, possibly with non-linearities are added.



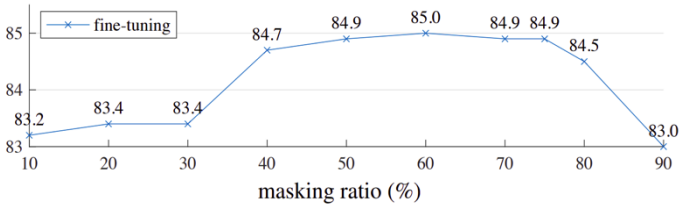
- linear probing: provides a measure of representation quality of a pre-training in restricted conditions
- fine-tuning: exploits models near-true potential to adopt for new tasks

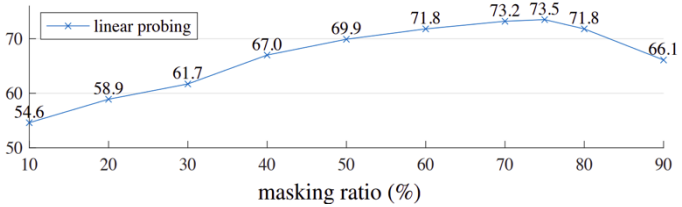
He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

Ablation Studies

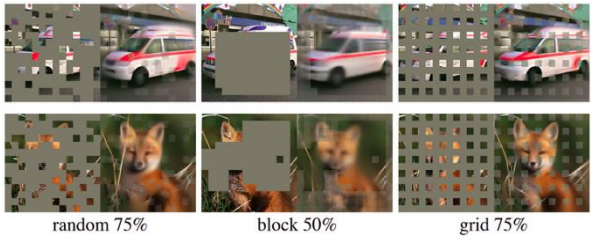
So many modeling/hyperparameter choices:

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



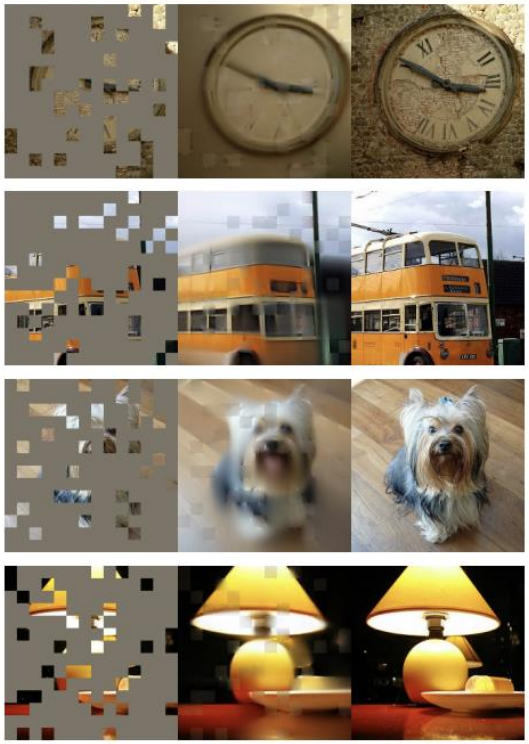


case	ratio	ft	lin
random	75	**84.9**	**73.5**
block	50	83.9	72.3
block	75	82.8	63.9
grid	75	84.0	66.0



He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

Masked Autoencoder – Comparisons



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**

He et al., 2021 Masked Autoencoders Are Scalable Vision Learners

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 often do not 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 often do not 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

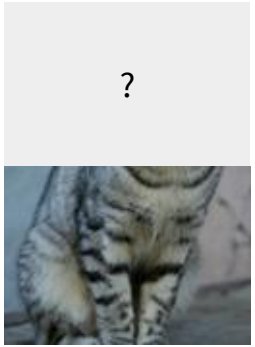
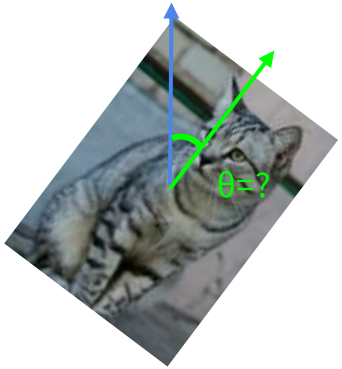


image completion



rotation prediction

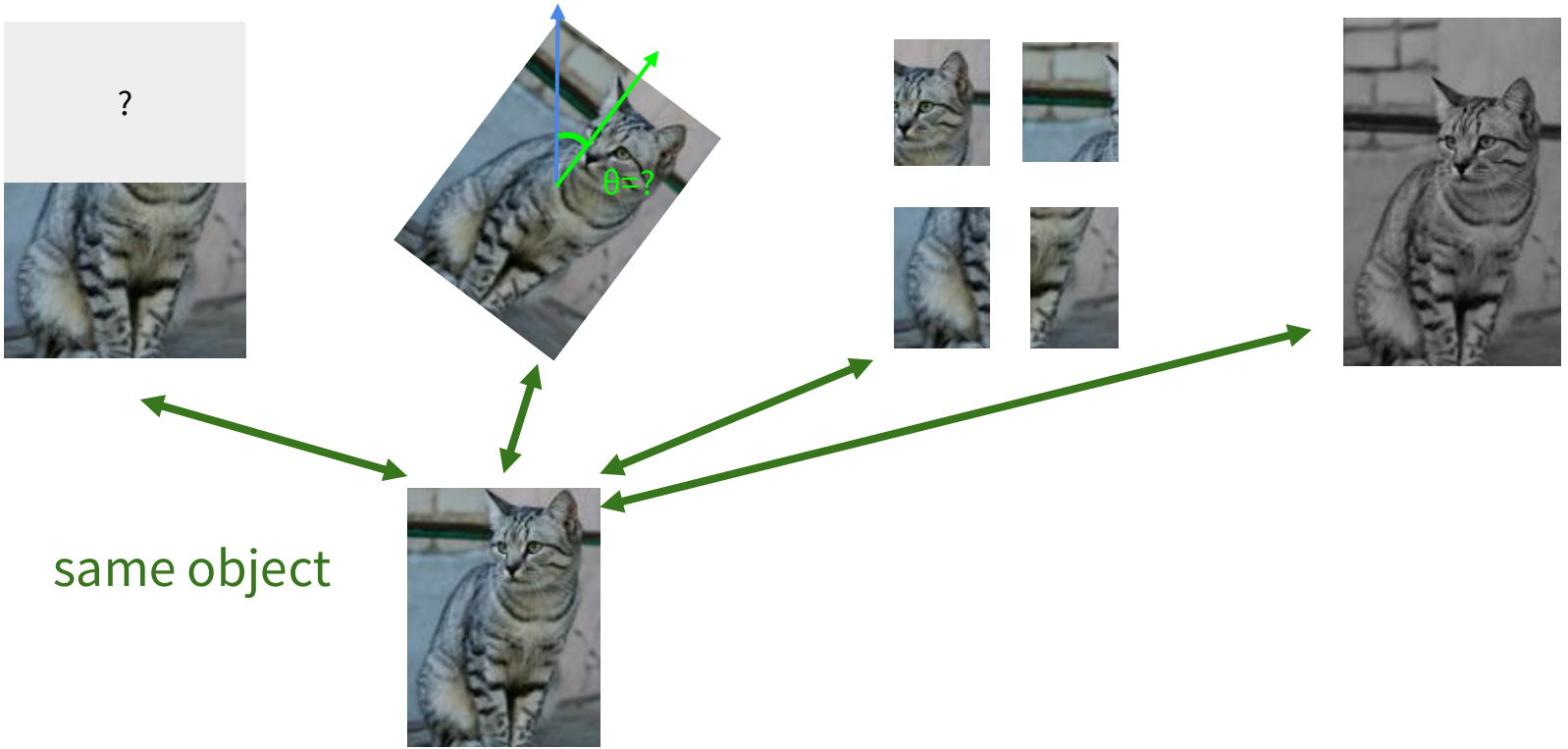
“jigsaw puzzle”

colorization

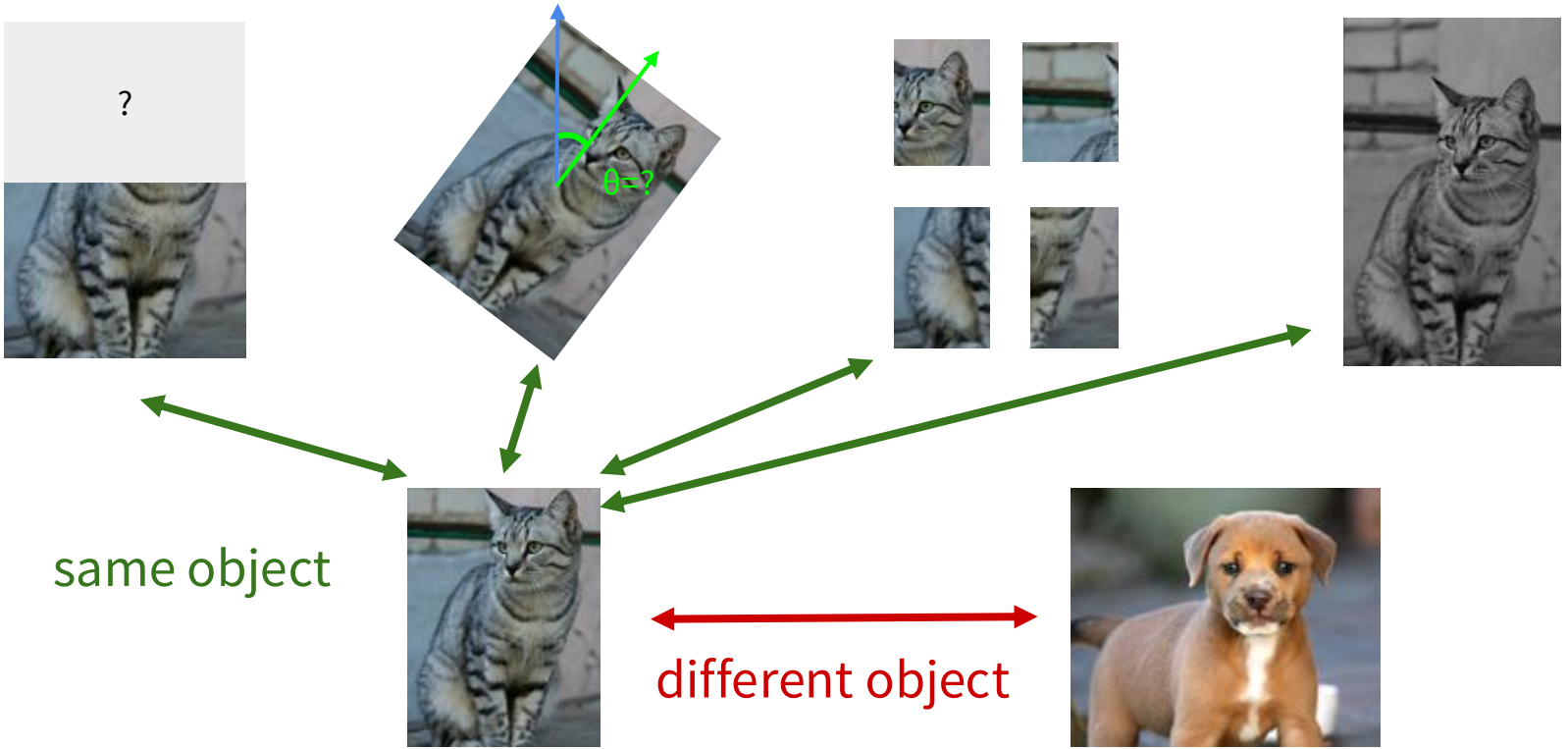
Learned representations may be tied to a specific pretext task!

Can we come up with a more general pretext task?

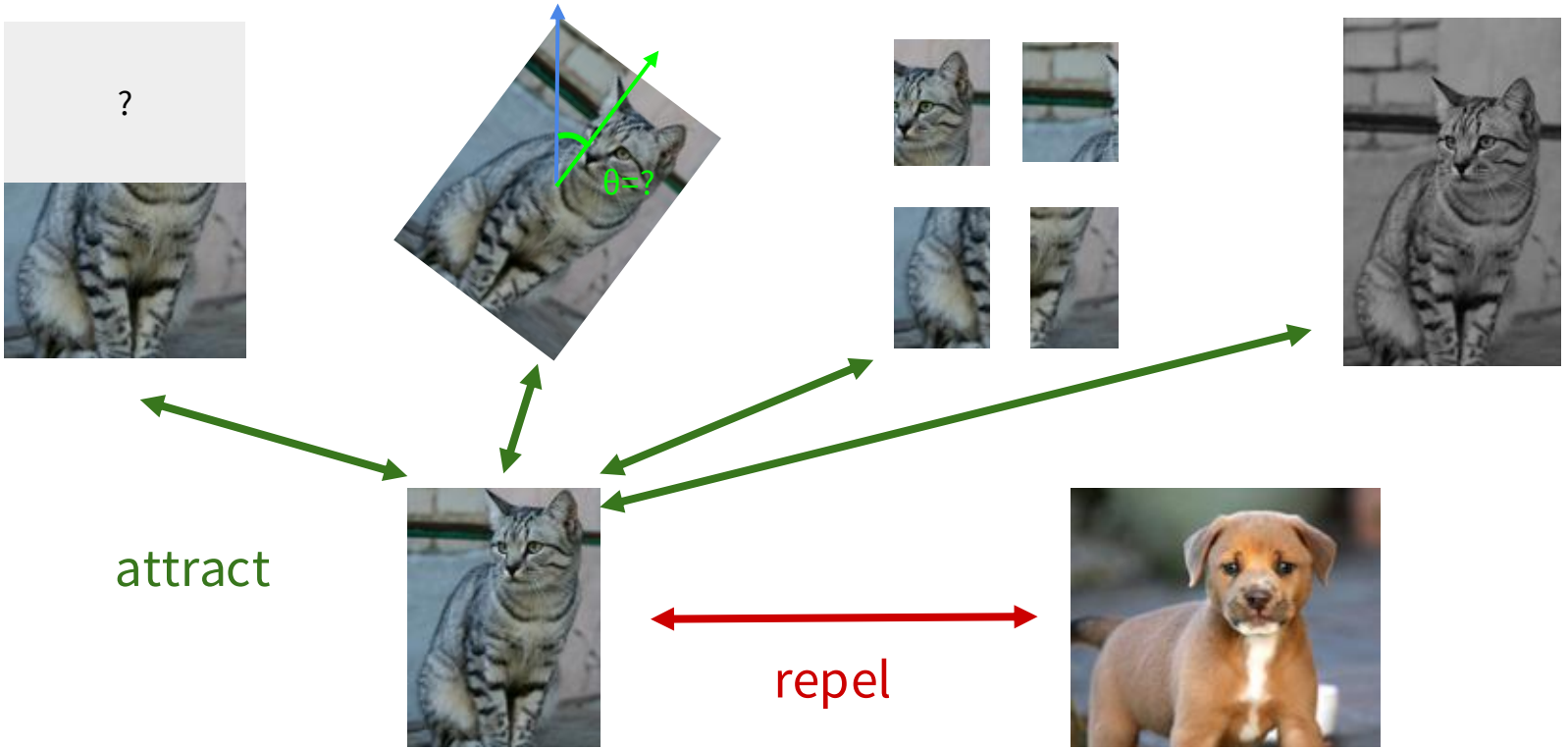
A more general pretext task?



A more general pretext task?



Contrastive Representation Learning



Today's Agenda

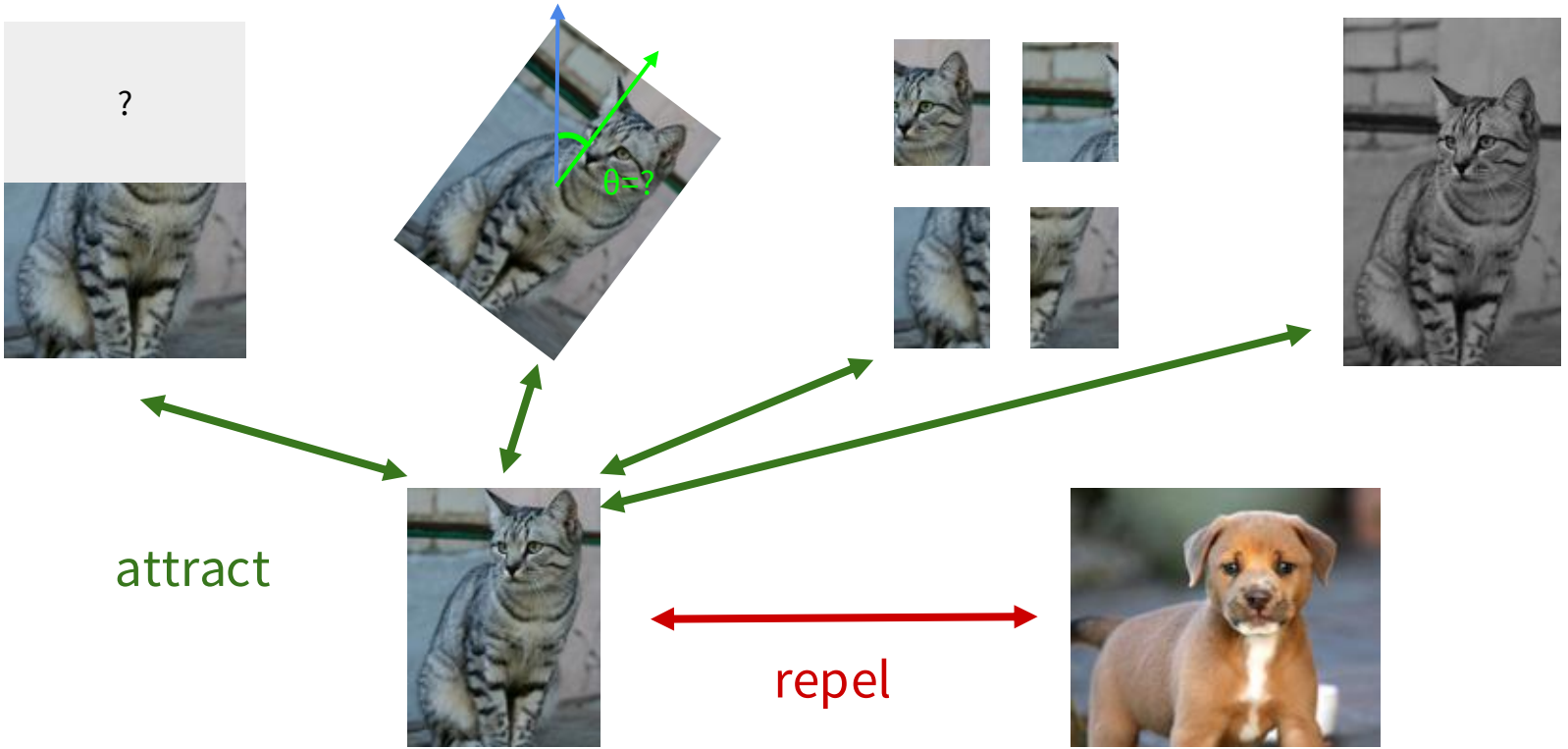
Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring
- Reconstruction-based learning (MAE)

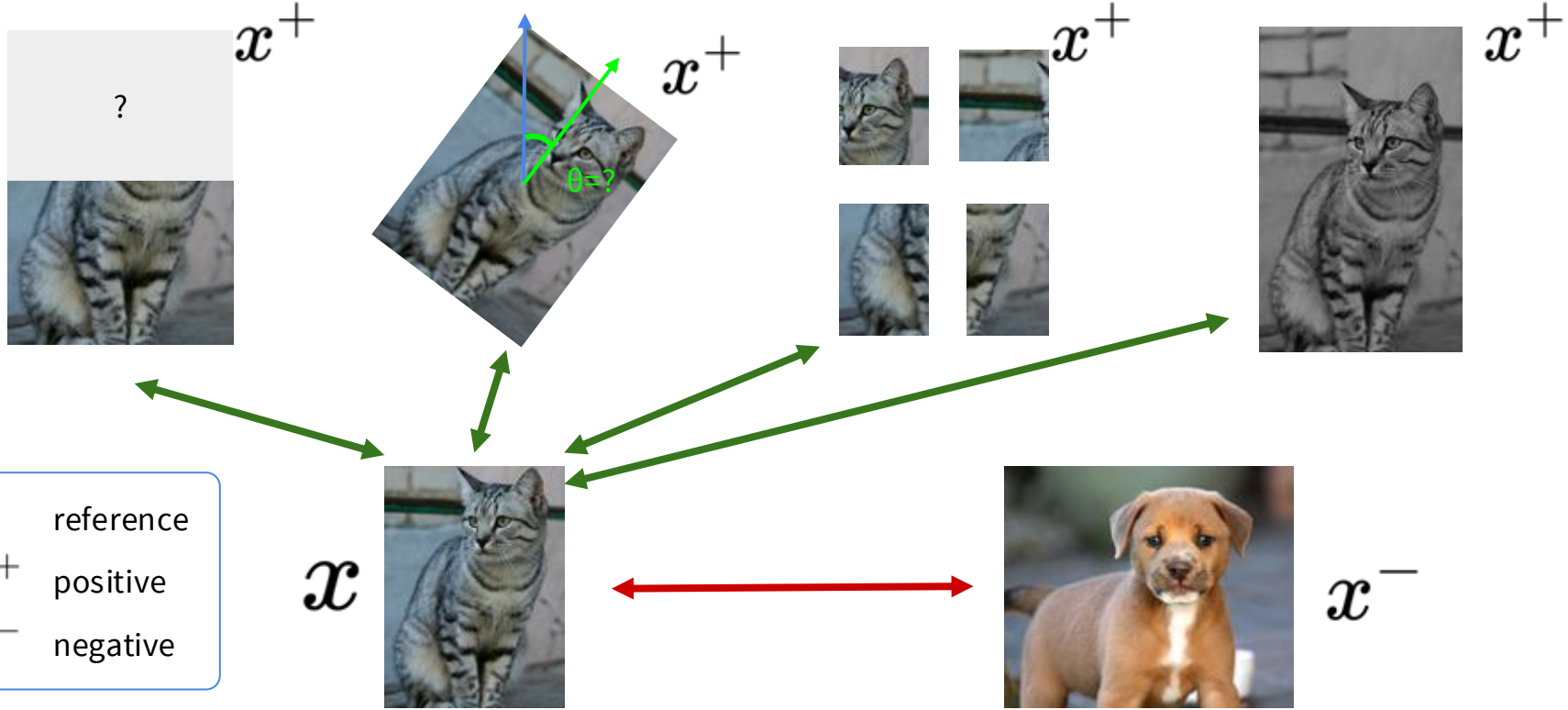
Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC
- Self-Distillation Without Labels, DINO

Contrastive Representation Learning



Contrastive Representation Learning



A formulation of contrastive learning

What we want:

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

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:

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

x

x^+

x

x_1^-

x_2^-

x_3^-

...

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

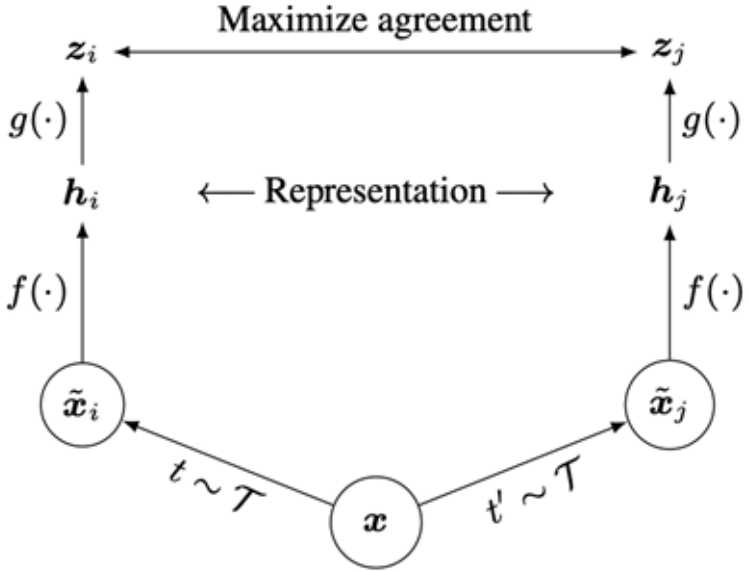
Cosine similarity as the score function:

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

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

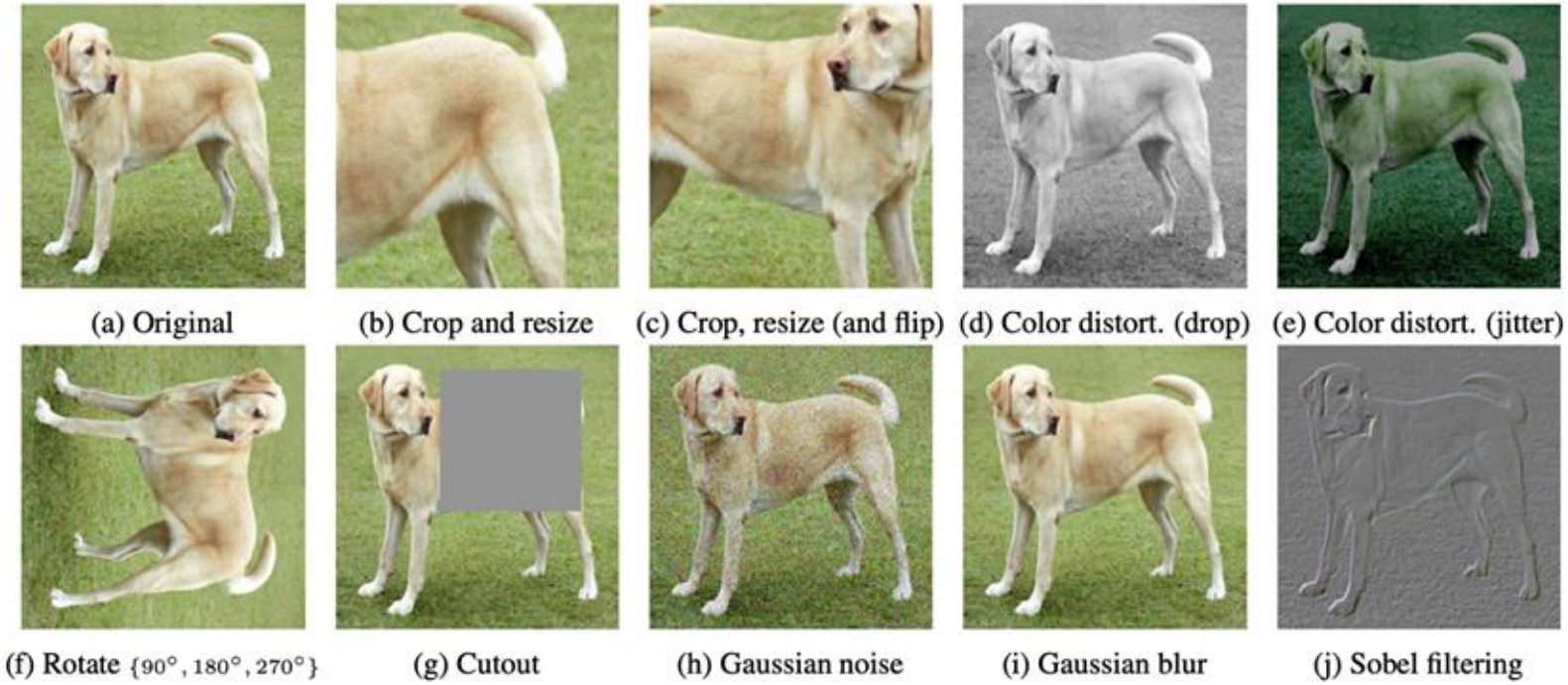
Generate positive samples through data augmentation:

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



Source: Chen et al., 2020

SimCLR: generating positive samples from data augmentation



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 τ, 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 $g(\cdot)$

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

Source: Chen et al., 2020

SimCLR

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, 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 $g(\cdot)$

Generate a positive pair by sampling data augmentation functions

*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

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, 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 $g(\cdot)$

Generate a positive pair by sampling data augmentation functions

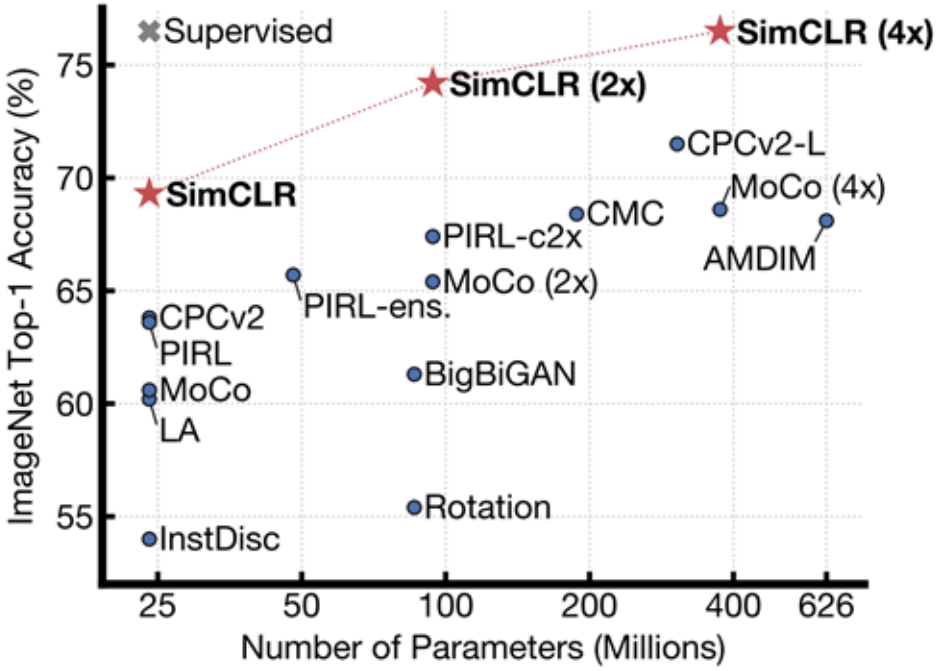
Iterate through and use each of the 2N sample as reference, compute average loss

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

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

Source: Chen et al., 2020

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 (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**

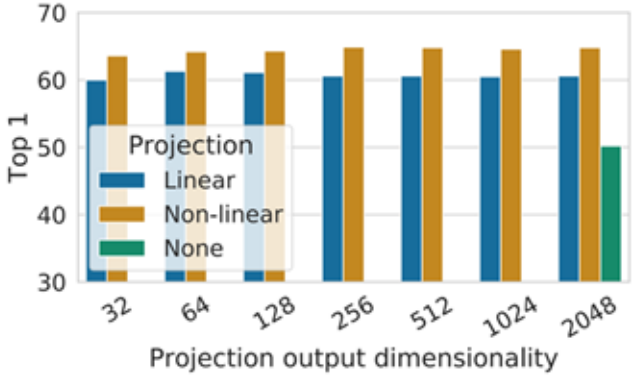
Table 7. ImageNet accuracy of models trained with few labels.

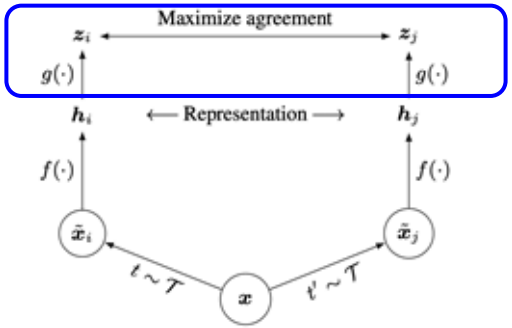
Train feature encoder on ImageNet (entire training set) using SimCLR.

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 z is trained to be invariant to data transformation.
- by leveraging the projection head g(·), more information can be preserved in the h representation space

Source: Chen et al., 2020

SimCLR design choices: large batch size

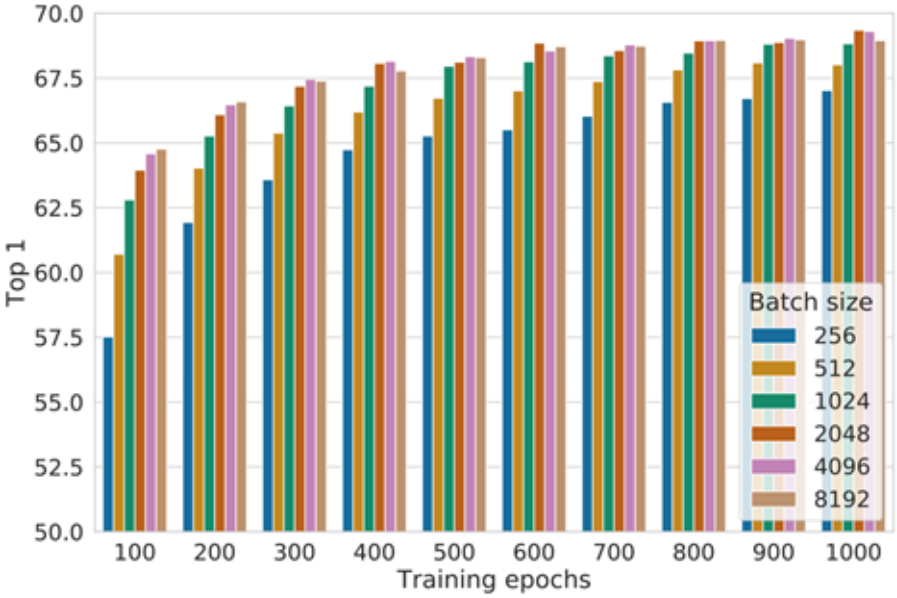


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

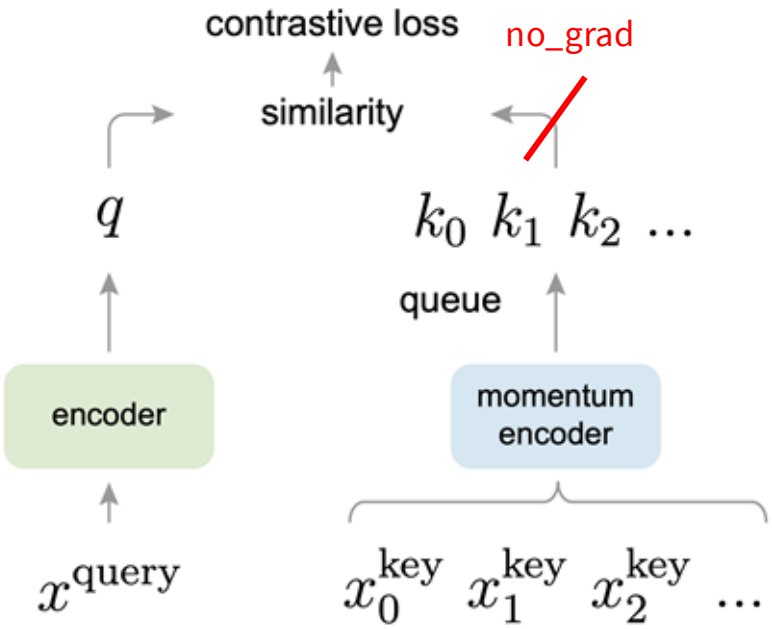
Large training batch size is crucial for SimCLR!

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

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

Source: Chen et al., 2020

Momentum Contrastive Learning (MoCo)

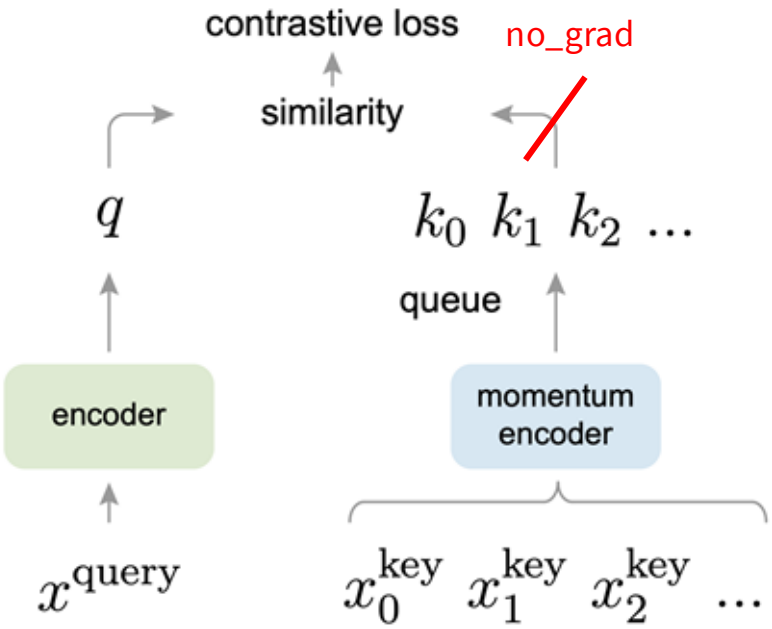


Key differences to SimCLR:

- 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:

$$\theta_{\text{k}} \leftarrow m\theta_{\text{k}} + (1-m)\theta_{\text{q}}$$

Source: He et al., 2020

MoCo

Generate a positive pair by sampling data augmentation functions

No gradient through the key

Use the running queue of keys as the negative samples

InfoNCE loss

Update f_k through momentum

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.

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 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.

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	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).

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	**5.0G**	**53 hrs**
end-to-end	256	7.4G	65 hrs
end-to-end	4096	93.0G†	n/a

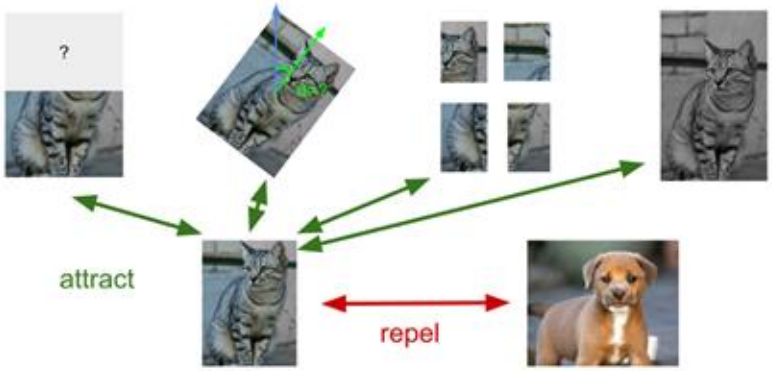
Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

Key takeaways:

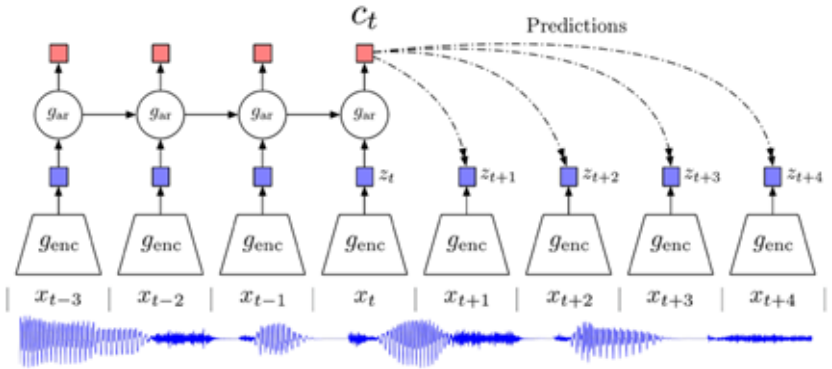
- Non-linear projection head and strong data augmentation are crucial for contrastive learning.
- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).
- … all with much smaller memory footprint! ("end-to-end" means SimCLR here)

Source: Chen et al., 2020

Instance vs. Sequence Contrastive Learning



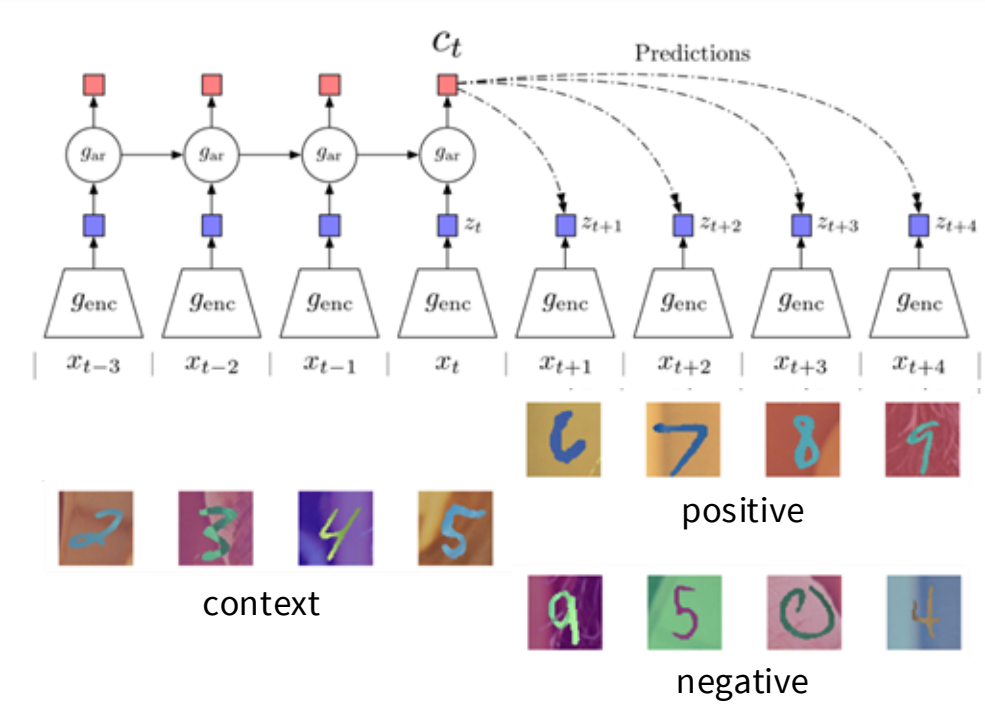
Instance-level contrastive learning:
contrastive learning based on
positive & negative instances.
Examples: SimCLR, MoCo



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.

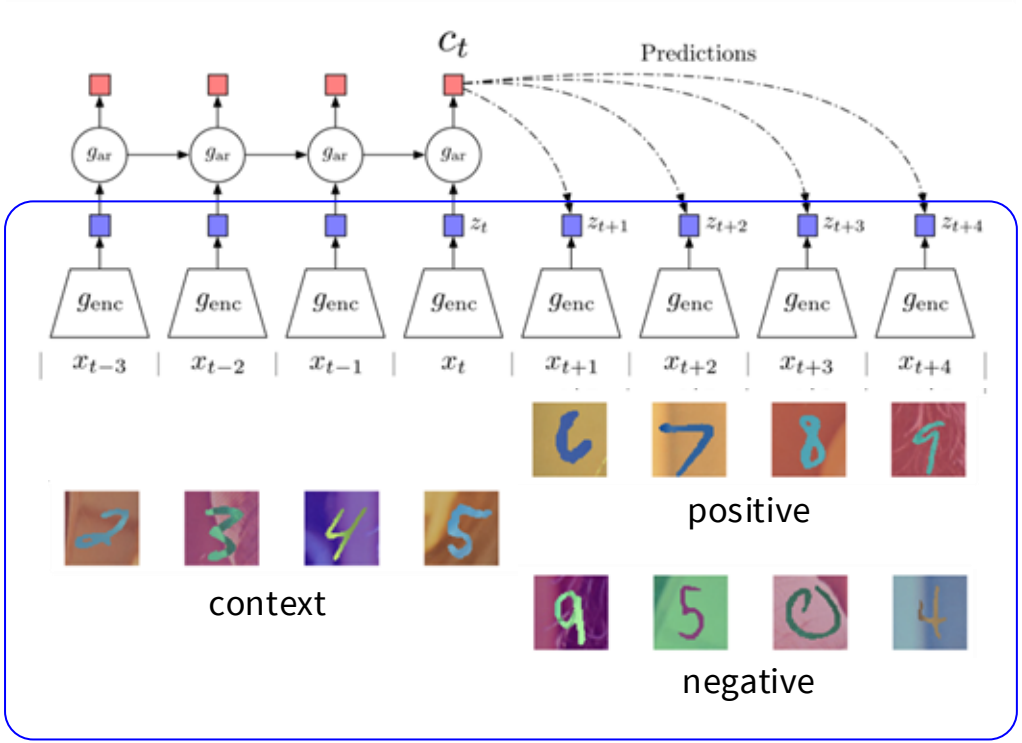
Predictive: the model has to predict future patterns given the current context.

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)

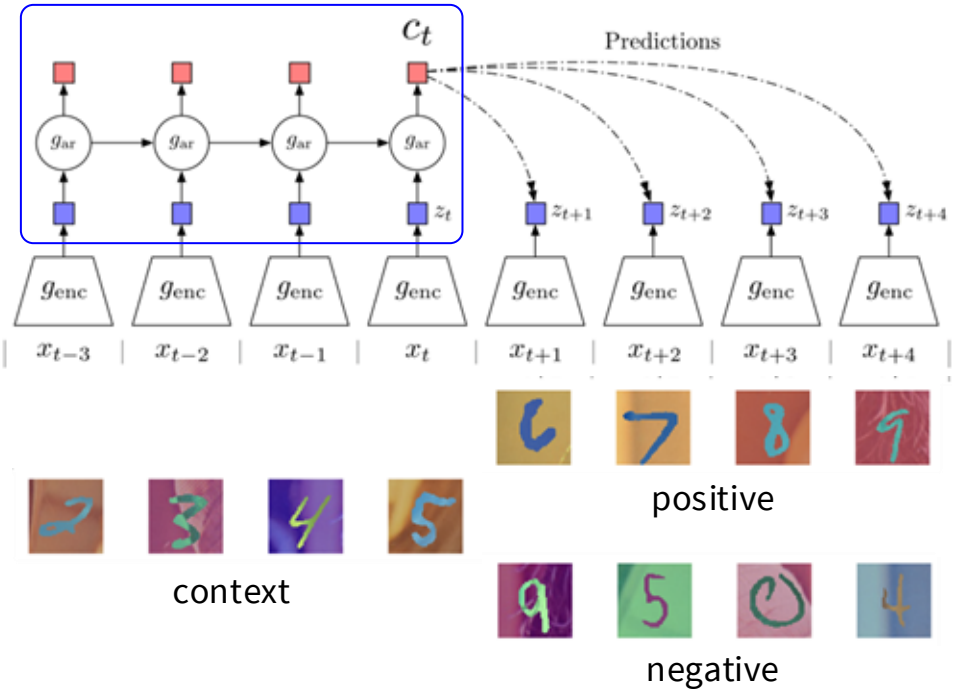


1. Encode all samples in a sequence into vectors $z_t = g_{enc}(x_t)$

Figure source

Source: van den Oord et al., 2018,

Contrastive Predictive Coding (CPC)



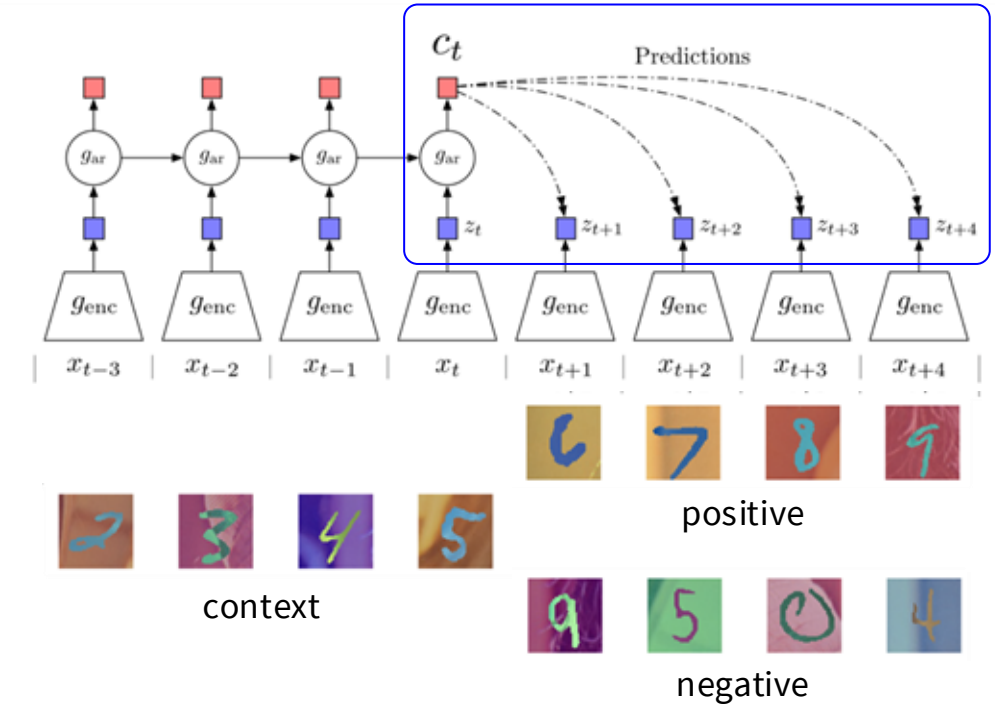
1. Encode all samples in a sequence into vectors $z_t = g_{enc}(x_t)$

2. Summarize context (e.g., half of a sequence) into a context code c_t using an auto-regressive model (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 $z_t = g_{enc}(x_t)$

2. Summarize context (e.g., half of a sequence) into a context code c_t using an auto-regressive model (g_{ar})

3. Compute InfoNCE loss between the context c_t and future code z_{t+k} using the following time-dependent score function:

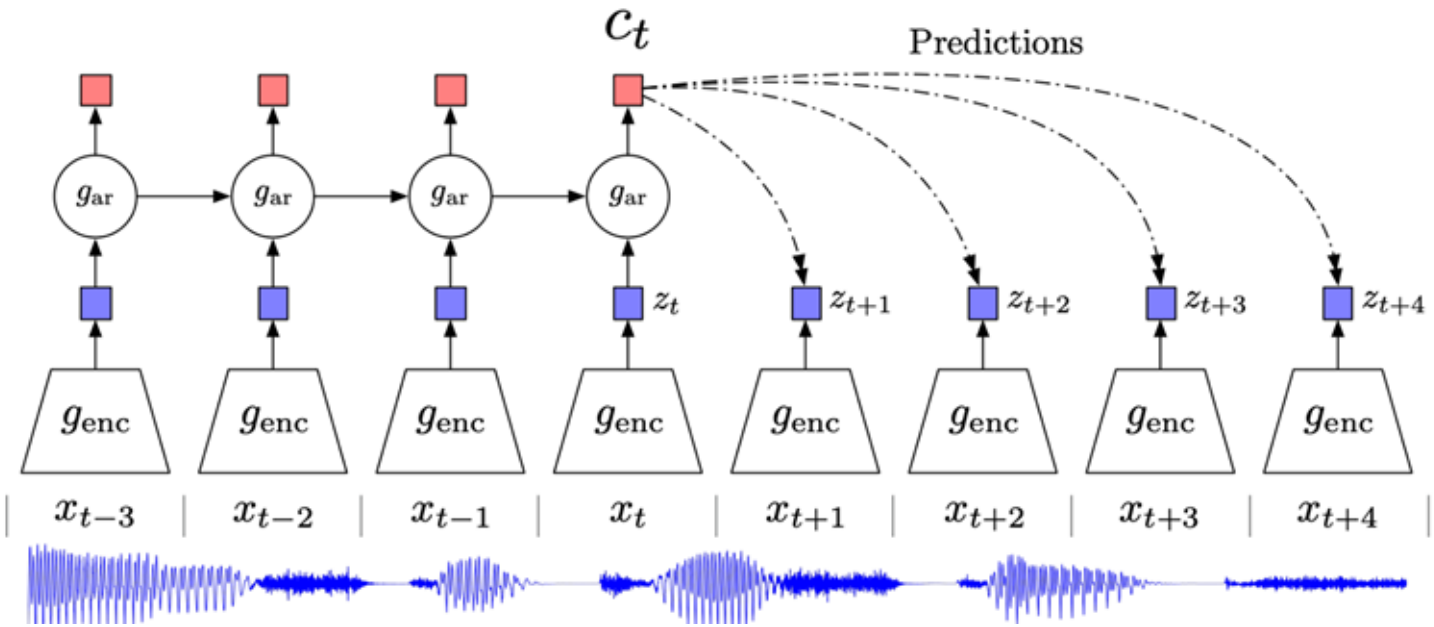
$$s_k(z_{t+k}, c_t) = z_{t+k}^T W_k c_t$$

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

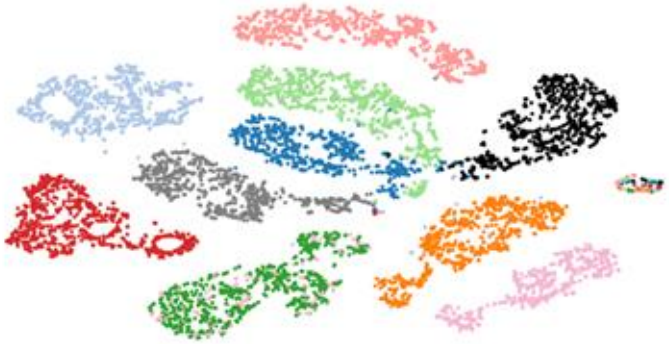


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

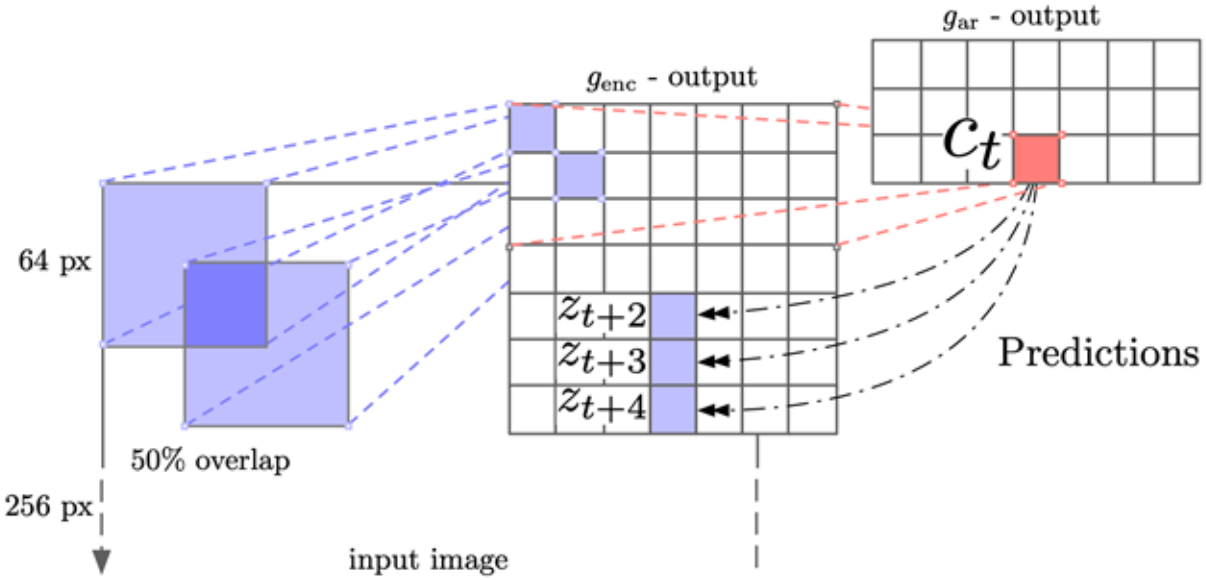
Method	ACC
Phone classification	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
Speaker classification	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

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,

DINO: Self-Distillation with No Labels

Emerging Properties in Self-Supervised Vision Transformers

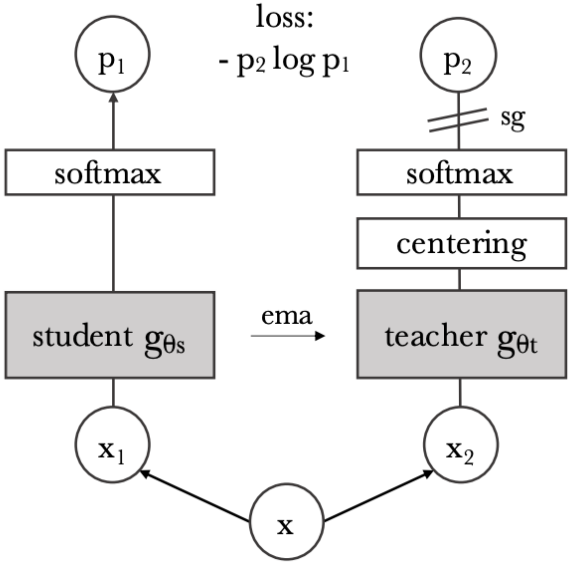
Mathilde Caron[1,2] Hugo Touvron[1,3] Ishan Misra[1] Hervé Jegou[1]
Julien Mairal[2] Piotr Bojanowski[1] Armand Joulin[1]

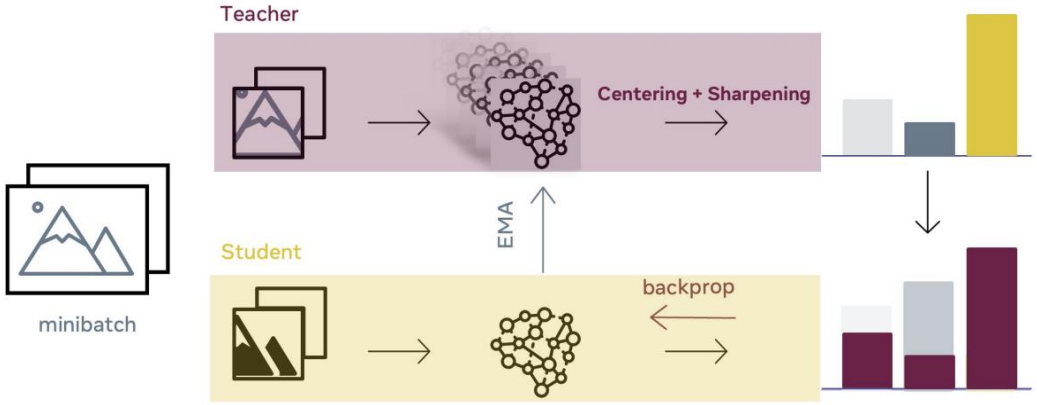
[1] Facebook AI Research [2] Inria* [3] Sorbonne University

Figure 1: **Self-attention from a Vision Transformer with 8×8 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.

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

DINO





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

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

DINO

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()
```

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

DINO v2

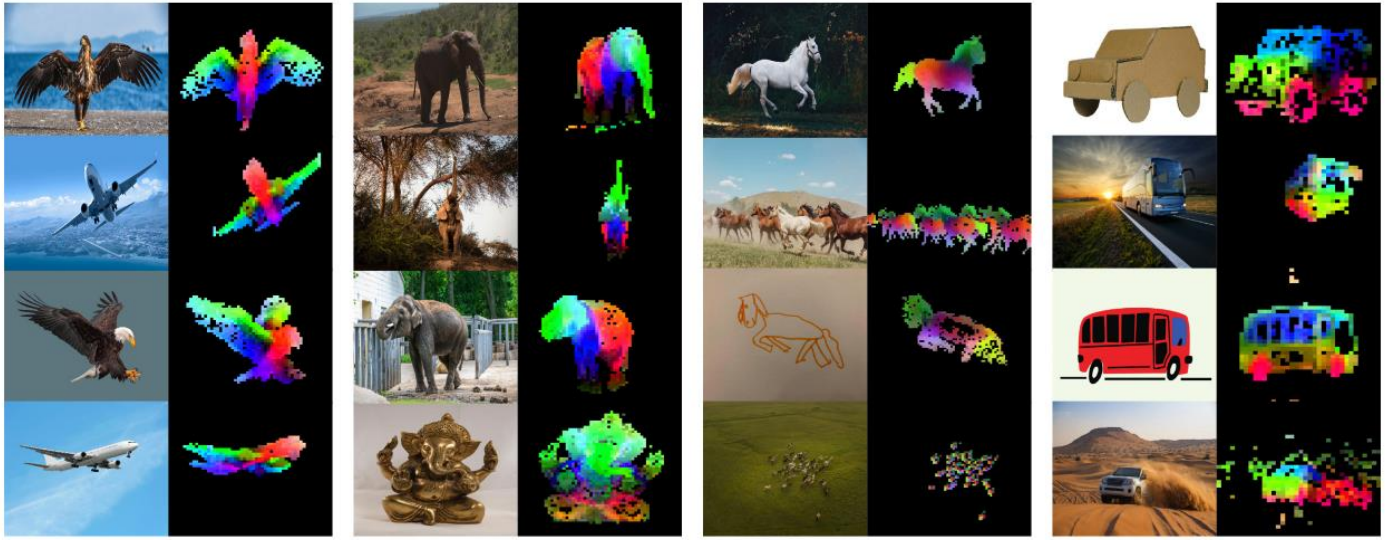


Figure 1: **Visualization of the first PCA components.** We compute a PCA between the patches of the images from the same column (a, b, c and d) and show their first 3 components. Each component is matched to a different color channel. Same parts are matched between related images despite changes of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

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

Summary: Contrastive Representation Learning

A general formulation for contrastive learning:

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

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)

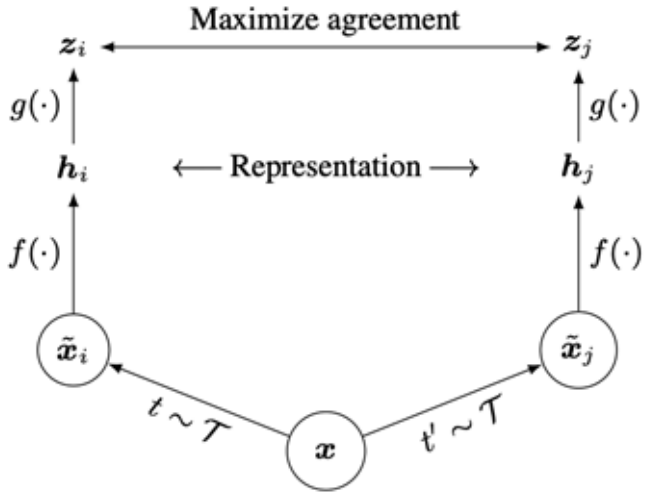
A lower bound on the mutual information between f(x) and $f(x^+)$

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

Summary: Contrastive Representation Learning

SimCLR: a simple framework for contrastive representation learning

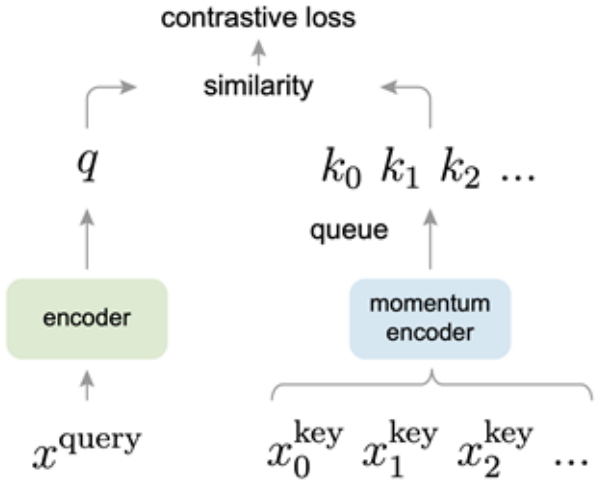
- Key ideas: non-linear projection head to allow flexible representation learning
- Simple to implement, effective in learning visual representation
- Requires large training batch size to be effective; large memory footprint



Summary: Contrastive Representation Learning

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

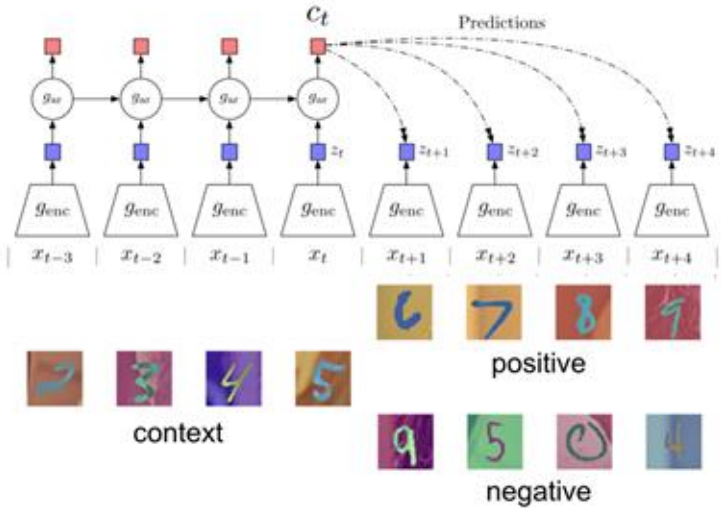
- Decouples negative sample size from minibatch size; allows large batch training without TPU
- MoCo-v2 combines the key ideas from SimCLR, i.e., nonlinear projection head, strong data augmentation, with momentum contrastive learning



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.



Next time: Generative Models