Lecture 9:
Self-Supervised Learning

Recall: Supervised

Data: (x, y)
x is the input data, y is the output label.

Goal: Learn a function f: x -> y

Example: in image classification, x is the image and y is the object category

Problem: Supervised Learning is Expensive!

Assume that we want to label re-label ImageNet's 1.4 Million images.

How much will it cost?

Problem: Supervised Learning is Expensive!

Assume that we want to label re-label ImageNet's 1.4 Million images.

How much will it cost?

(1,400,000 images)	(Small to medium sized dataset)
$\times$ (10 seconds/image)	(Fast annotation)
$\times$ (1/3600 hours/second)	
$\times$ ($15 / hour)	(Low wage paid to annotator)

Problem: Supervised Learning is Expensive!

Assume that we want to label re-label ImageNet's 1.4 Million images.

How much will it cost?

(1,400,000 images)	(Small to medium sized dataset)
× (10 seconds/image)	(Fast annotation)
× (1/3600 hours/second)	
× ($15 / hour)	(Low wage paid to annotator)
= $58,333	

Assumptions:

- one annotator per image,
- no benefits / payroll tax / crowdsourcing fee for annotators;
- not accounting for front end developer time to set up tasks for annotators.
- Real costs could easily be 3x this or more: **>$175,000**

Problem: Supervised Learning is Expensive!

Assume that we want to label CLIP's 1B images. (GPT also needs billions of documents)

How much will it cost?

(1,000,000,000 images)	(Small to medium sized dataset)
× (10 seconds/image)	(Fast annotation)
× (1/3600 hours/second)	
× ($15 / hour)	(Low wage paid to annotator)
= $41,666,667	

41 Million dollars (again, not including all other costs)

Supervised Learning is Not How We Learn

Babies don't get supervision for everything they see!

Baby image is CC0 public domain

Solution: self-supervised learning

Lets build methods that learn from ”raw” data – no annotations required

Unsupervised Learning: Model isn’t told what to predict. Older terminology, not used as much today.

Self-Supervised Learning: Model is trained to predict some naturally occurring signal in the raw data rather than human annotations.

Solution: self-supervised learning

Lets build methods that learn from ”raw” data – no annotations required

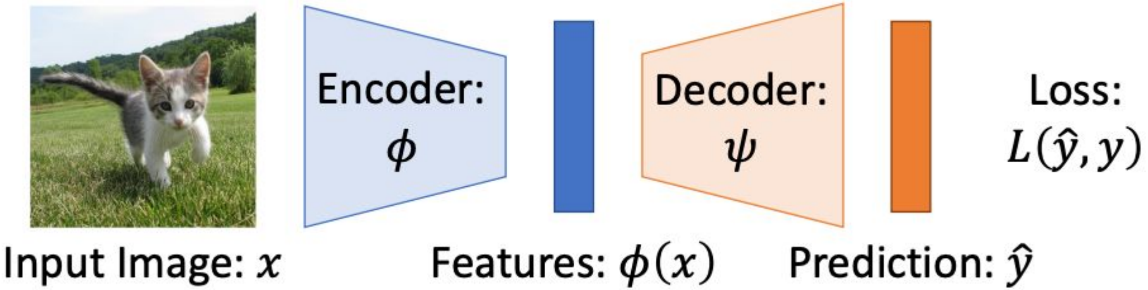
Unsupervised Learning: Model isn’t told what to predict. Older terminology, not used as much today.

Self-Supervised Learning: Model is trained to predict some naturally occurring signal in the raw data rather than human annotations.

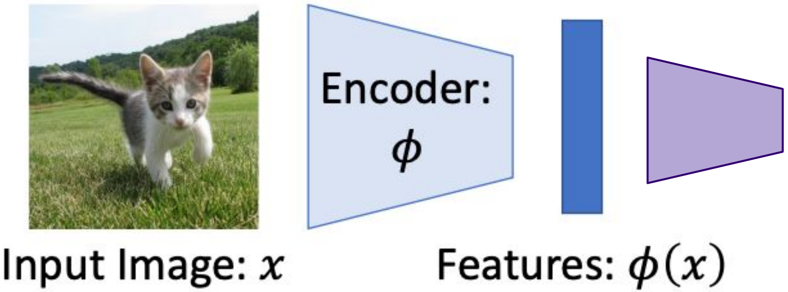
Semi-Supervised Learning: Train jointly with some labeled data and (a lot) of unlabeled data.

Self-Supervised Learning: Pretext then Transfer

Step 1: <u>Pretrain</u> a network on a <u>pretext task</u> that doesn't require supervision



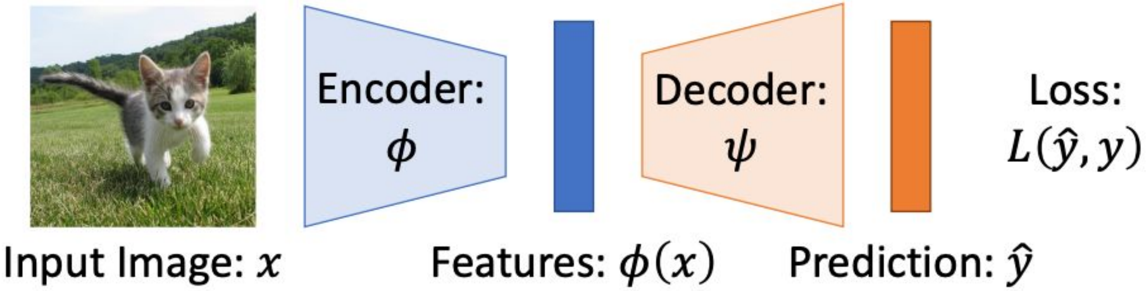
Step 2: Transfer encoder to <u>downstream tasks</u> via linear classifiers, KNN, finetuning



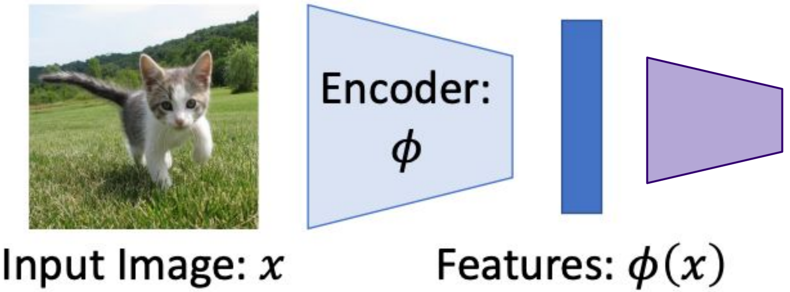
Downstream tasks:
Image classification, object detection, semantic segmentation

Goal of Self-Supervised Learning: Define pre-text tasks that do better than supervised learning

Step 1: Pretrain a network on a pretext task that doesn't require supervision



Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



Downstream tasks:
Image classification, object detection, semantic segmentation

Self-Supervised Learning: Pretext Tasks

Generative: Predict part of the input signal

- Autoencoders (sparse, denoising, masked)
- Autoregressive
- GANs
- Colorization
- Inpainting

Discriminative: Predict something about the input signal

- Context prediction
- Rotation • Clustering
- Contrastive

Multimodal: Use some additional signal in addition to RGB images

- Video
- 3D
- Sound
- Language

Self-supervised pretext tasks

Example: learn to predict image transformations / complete corrupted images

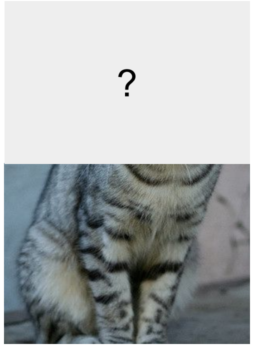
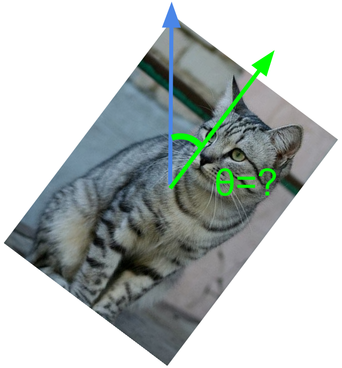


image completion



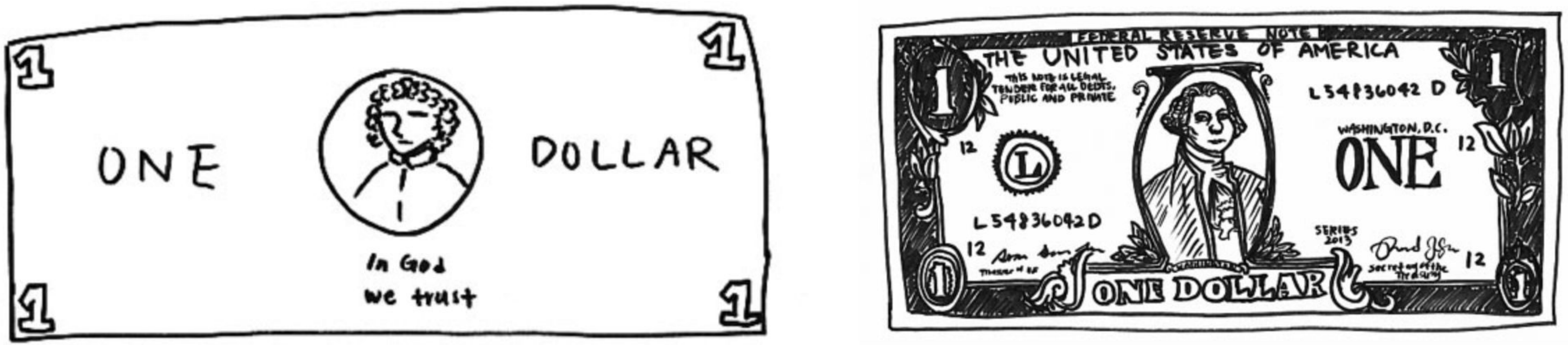
rotation prediction

“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 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

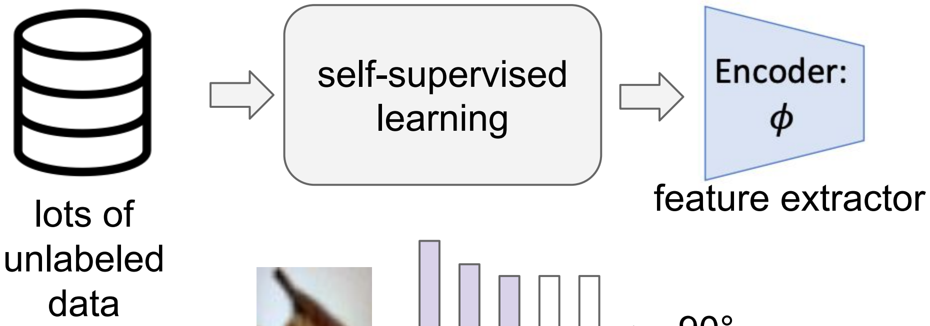
Source: [Anand, 2020]

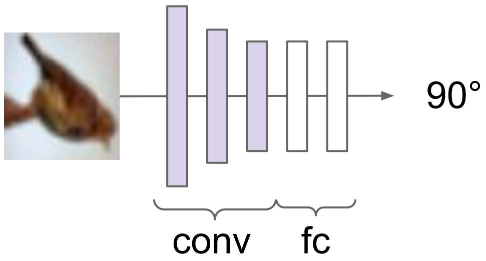
How to evaluate a self-supervised learning method?

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*

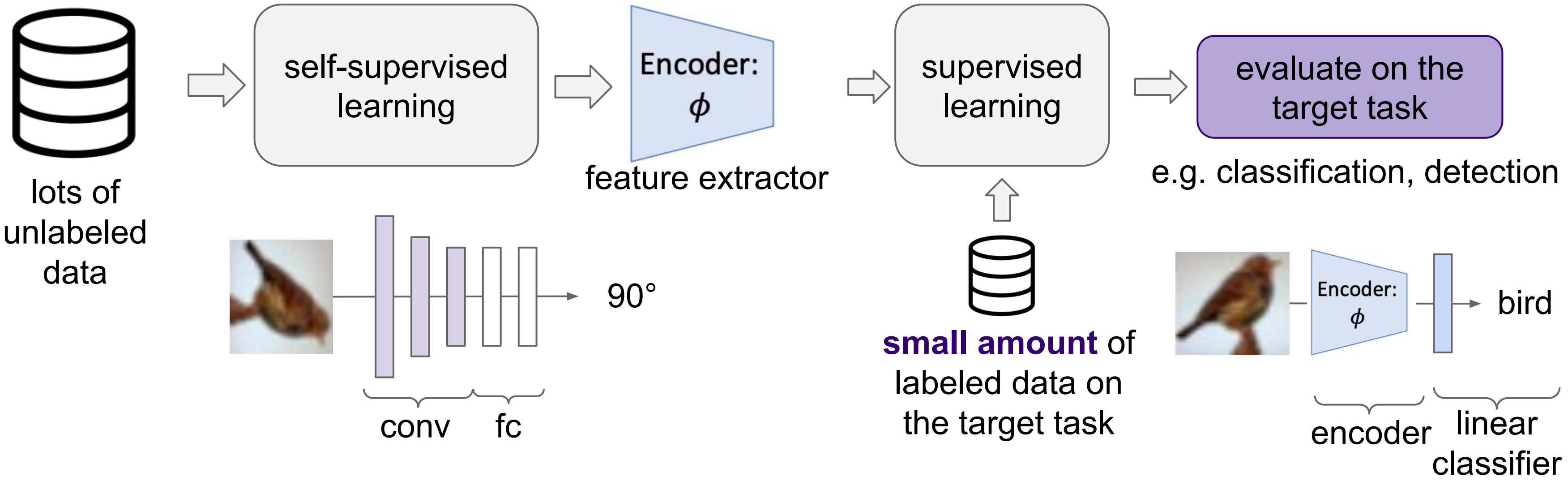
How to evaluate a self-supervised learning method?





Step 1: <u>Pretrain</u> a network on a <u>pretext task</u> that doesn't require supervision

How to evaluate a self-supervised learning method?

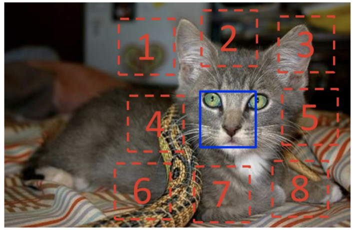


Step 1: Pretrain a network on a pretext task that doesn't require supervision

Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

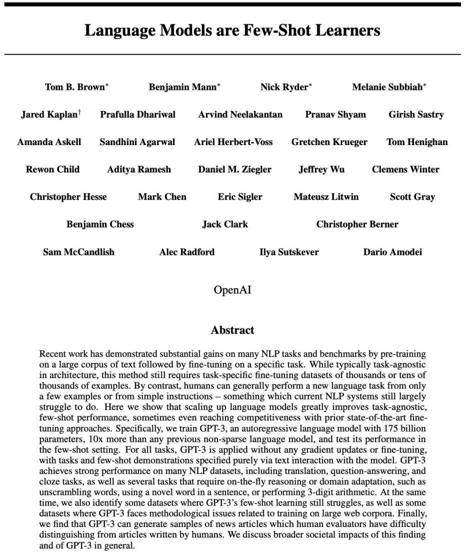
Broader picture

computer vision



Doersch et al., 2015

language modeling



Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

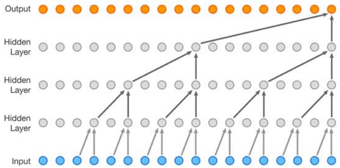
OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

GPT3 (Brown, Mann, Ryder, Subbiah et al., 2020)

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

Contrastive representation learning

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

Today’s Agenda

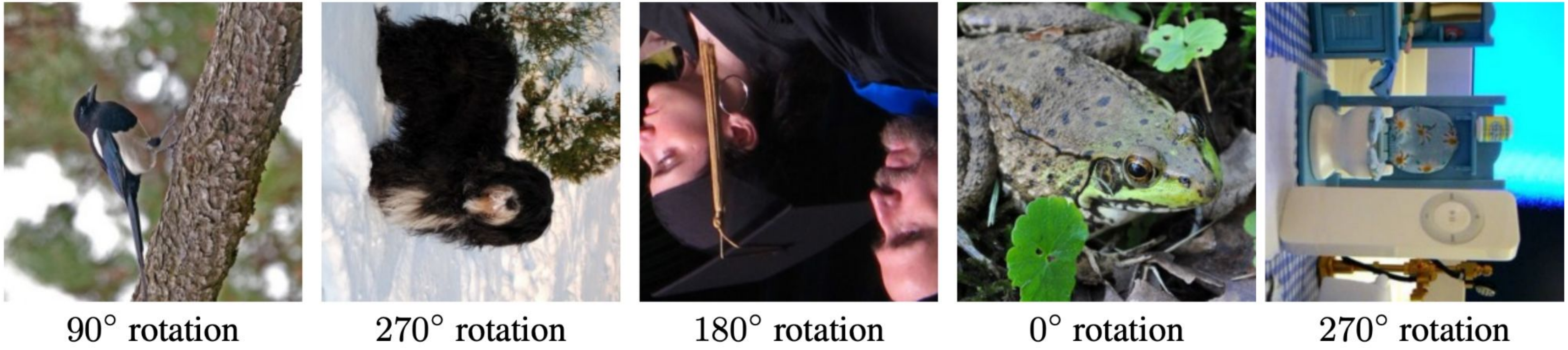
Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning

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

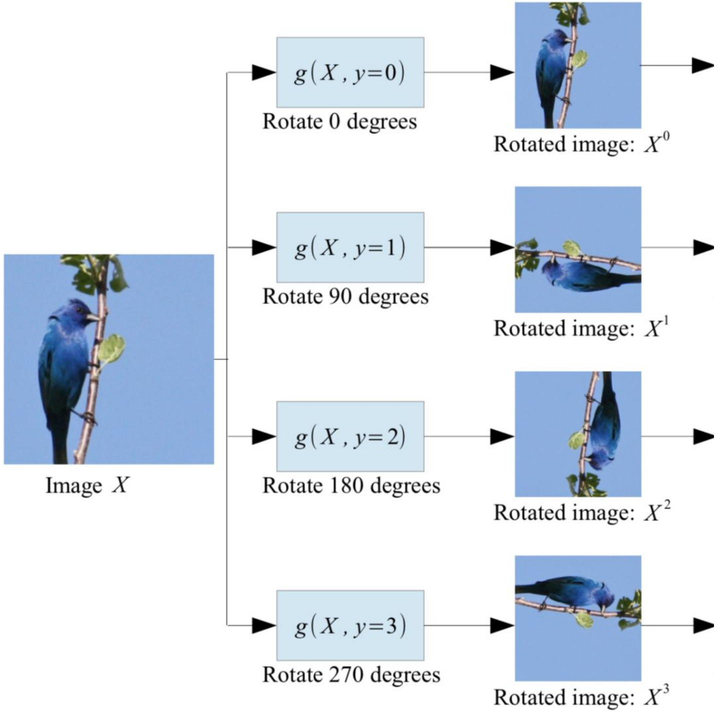
Pretext task: predict rotations



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

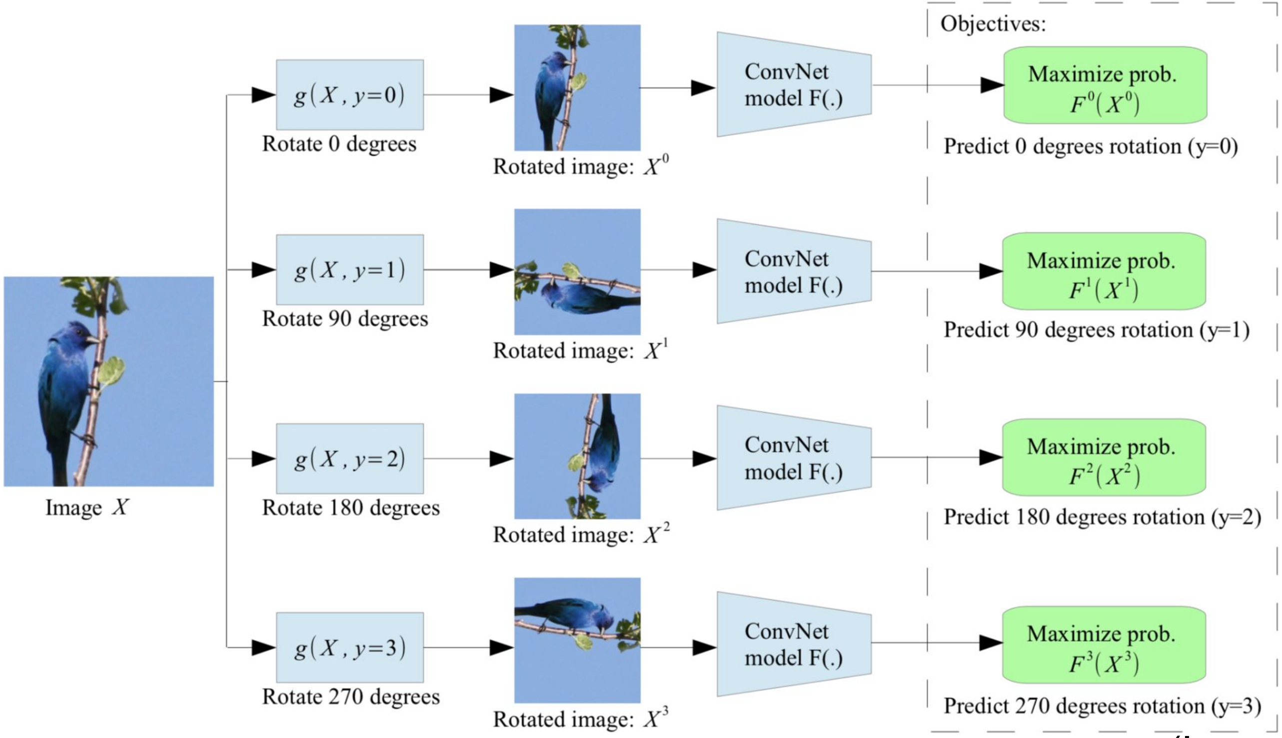


Self-supervised learning by rotating the entire input images.

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

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

Pretext task: predict rotations

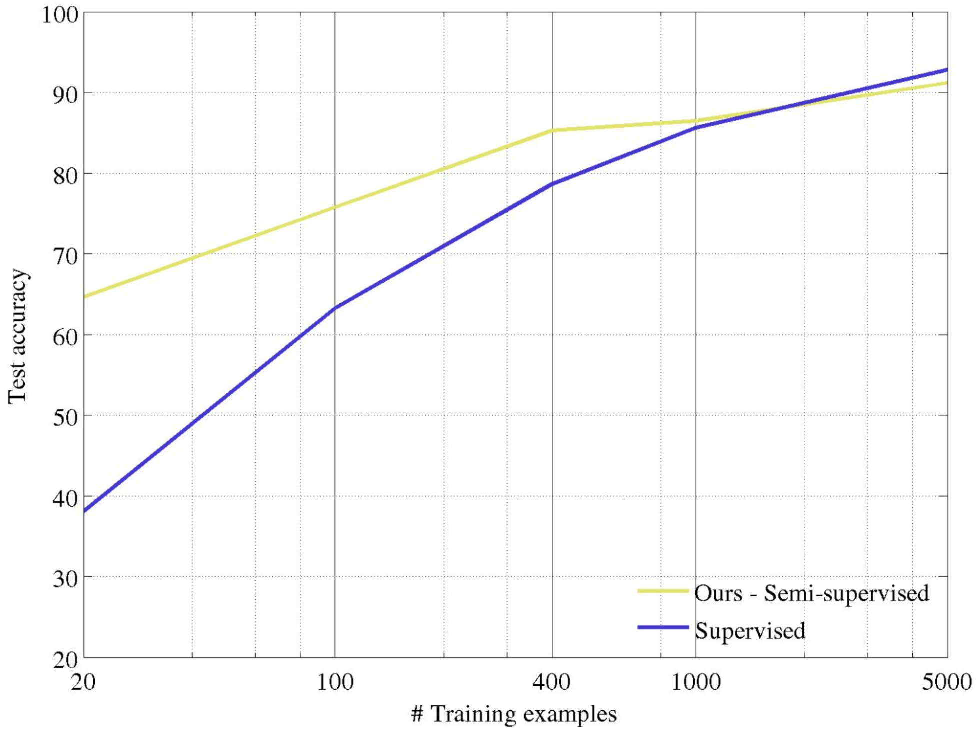


Self-supervised learning by rotating the entire input images.

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

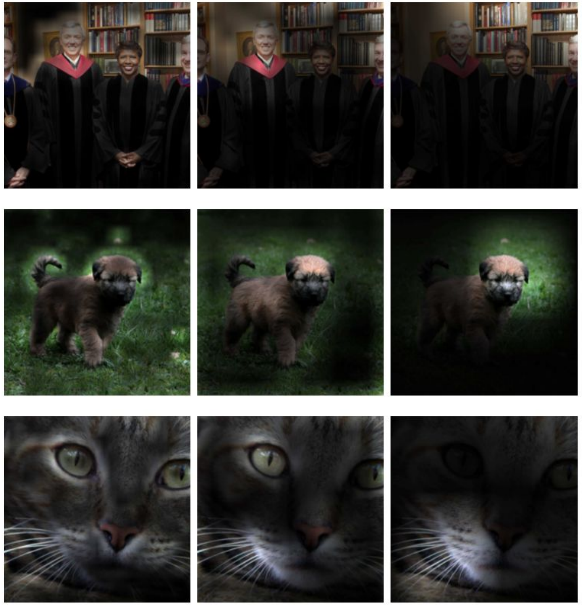
Self-supervised learning with rotation prediction

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

Finetune on labeled data from **Pascal VOC 2007**.

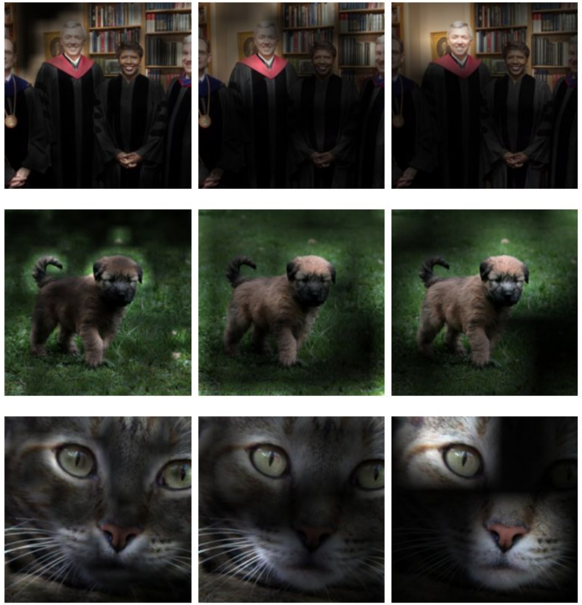
source: [Gidaris et al. 2018]

Visualize learned visual attentions



Conv1 27 × 27 Conv3 13 × 13 Conv5 6 × 6

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

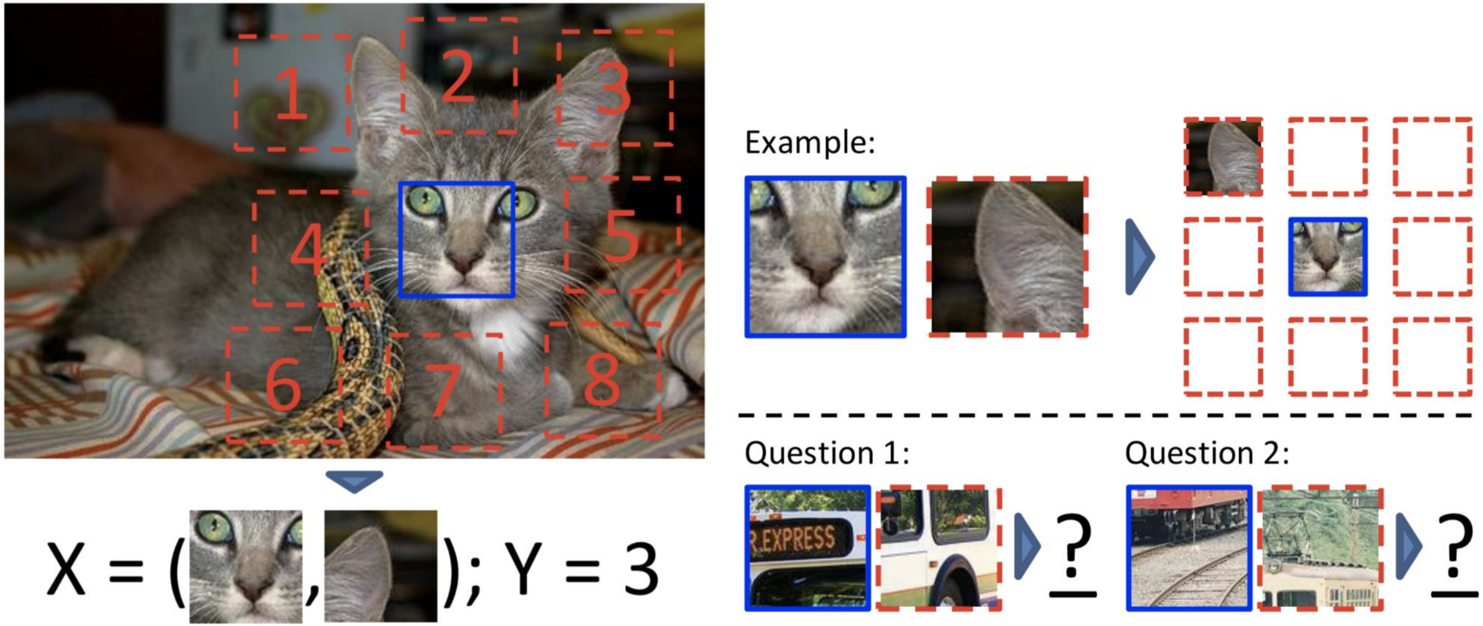


Conv1 27 × 27 Conv3 13 × 13 Conv5 6 × 6

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

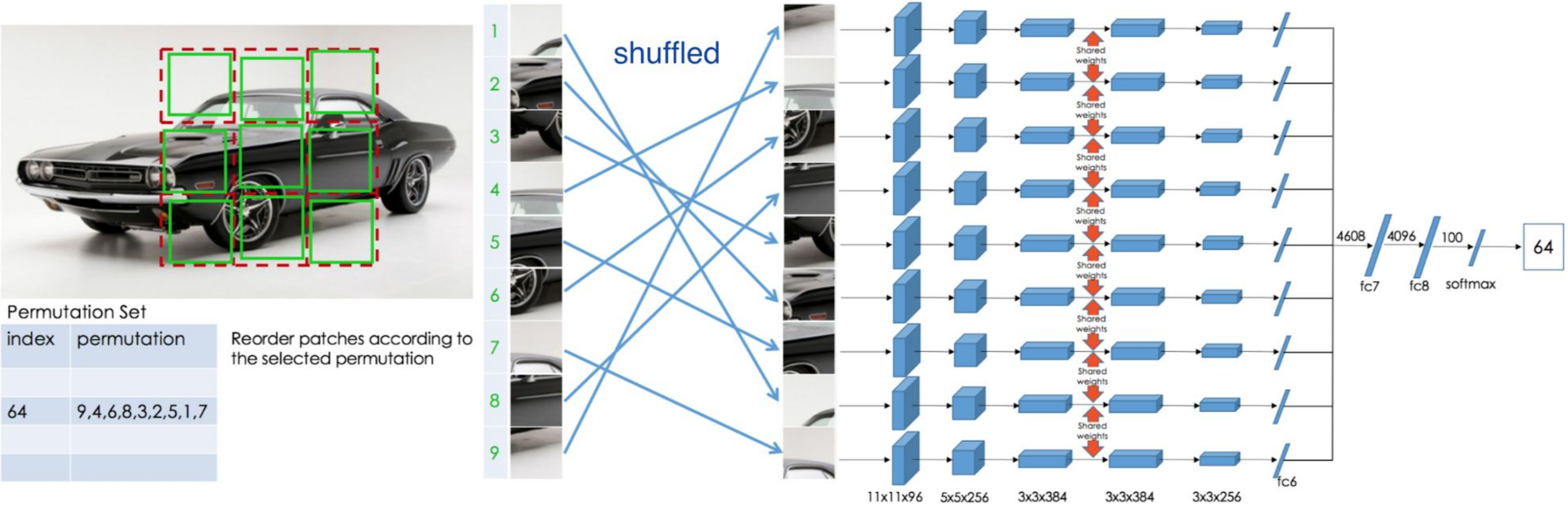
(Image source: Gidaris et al. 2018)

Pretext task: predict relative patch locations



(Image source: Doersch et al., 2015)

Pretext task: solving "jigsaw puzzles"



(Image source: Noroozi & Favaro, 2016)

Transfer learned features to supervised learning

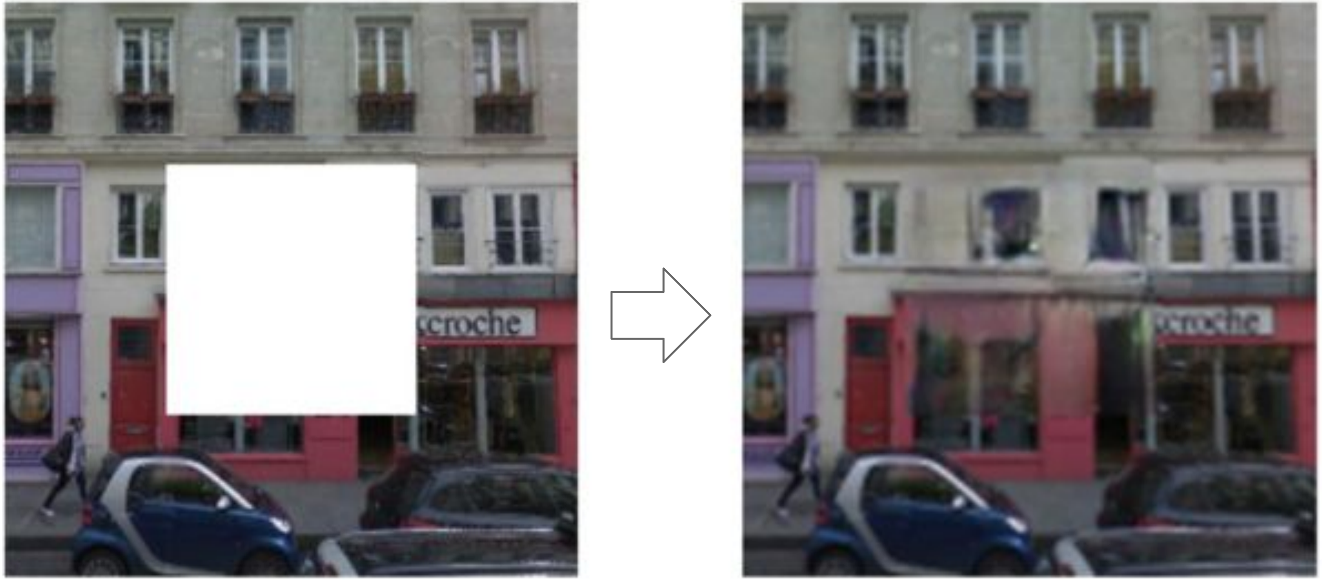
Table 1: Results on PASCAL VOC 2007 Detection and Classification. The results of the other methods are taken from Pathak *et al.* [30].

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](#))

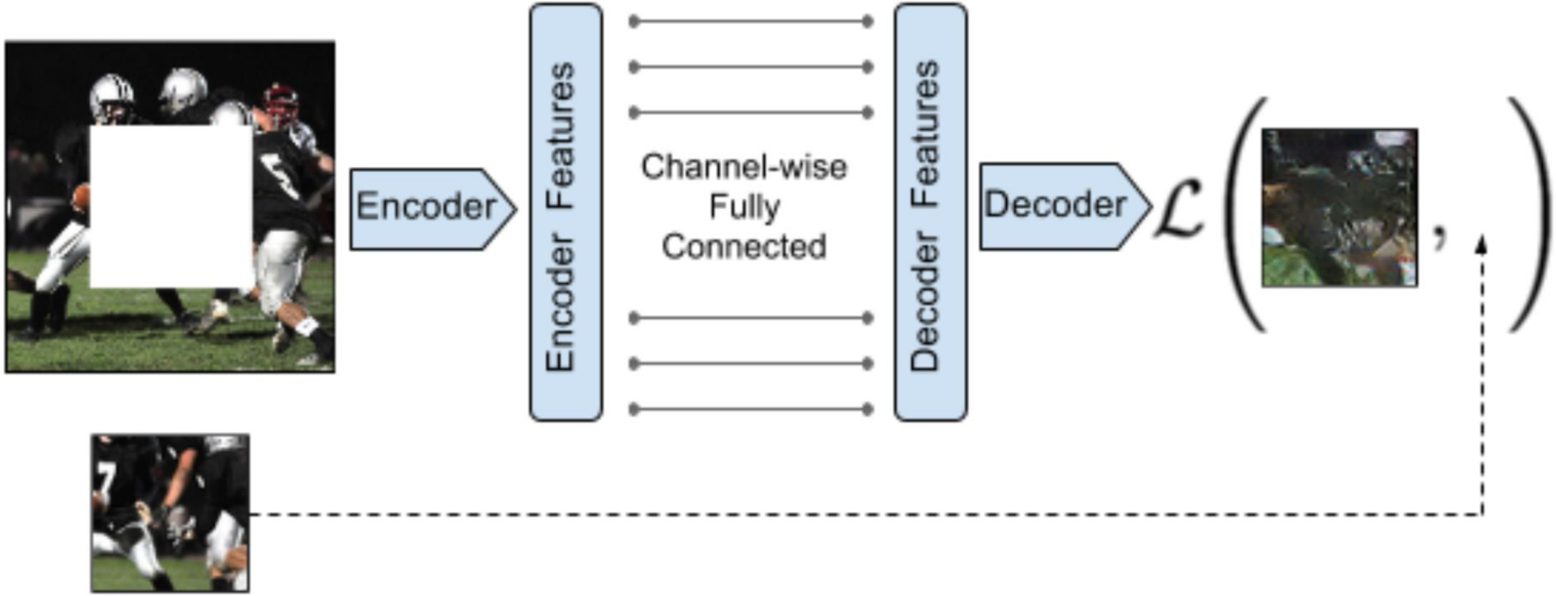
Pretext task: predict missing pixels (inpainting)



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

Source: [Pathak et al., 2016]

Learning to inpaint by reconstruction



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*

Source: Pathak et al., 2016

Inpainting evaluation

Input (context) reconstruction adversarial recon + adv

Source: [Pathak et al., 2016]

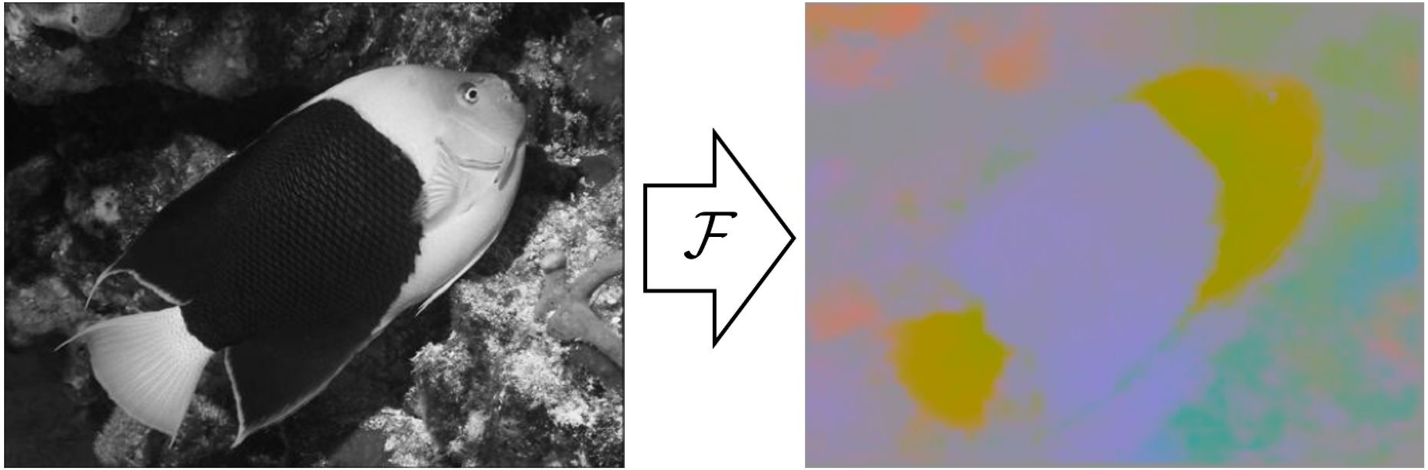
Transfer learned features to supervised learning

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

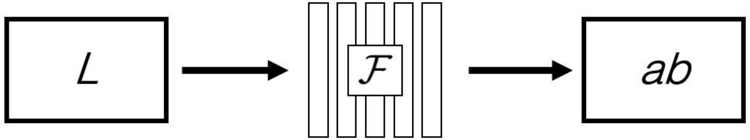


Grayscale image: *L* channel

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

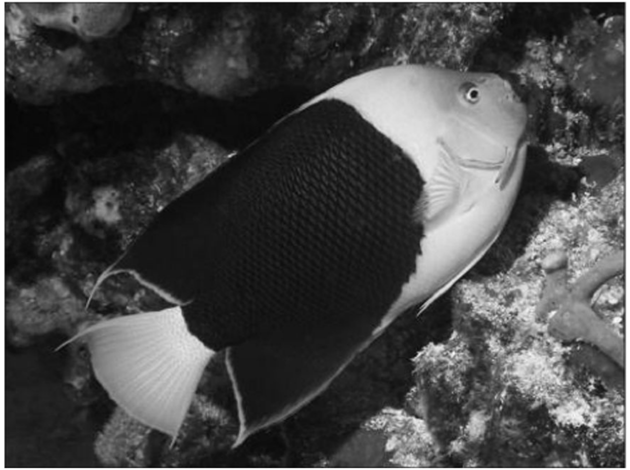
Color information: *ab* channels

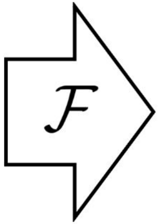
$$\widehat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 2}$$



Source: Richard Zhang / Phillip Isola

Pretext task: image coloring



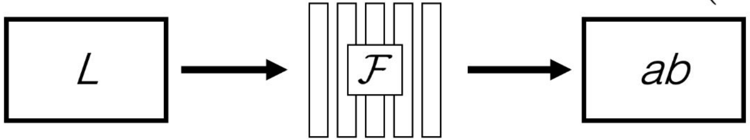


Grayscale image: *L* channel

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

Concatenate (*L*,*ab*) channels

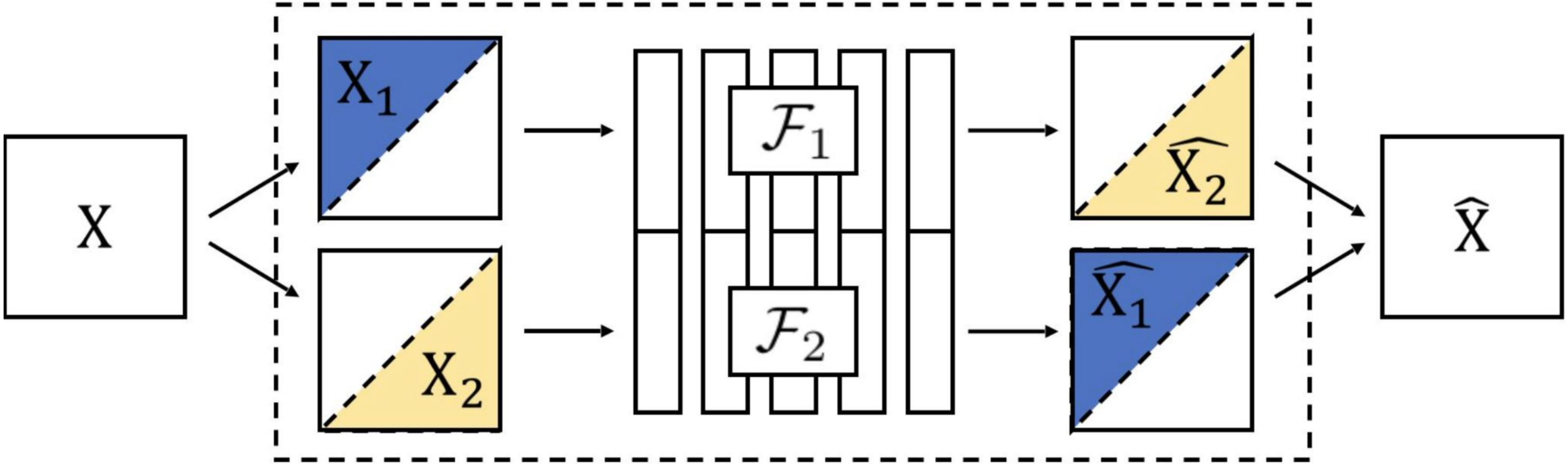
$$(\mathbf{X}, \widehat{\mathbf{Y}})$$



Source: Richard Zhang / Phillip Isola

Learning features from colorization: Split-brain Autoencoder

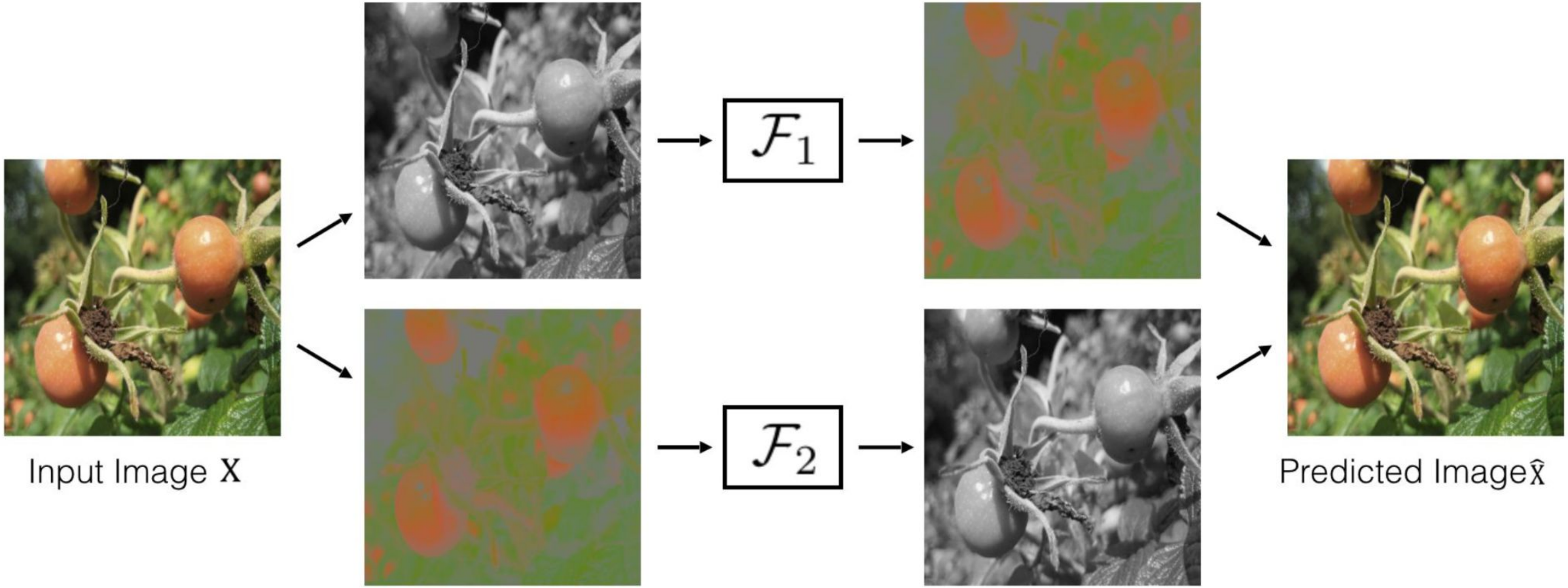
Idea: cross-channel predictions



Split-Brain Autoencoder

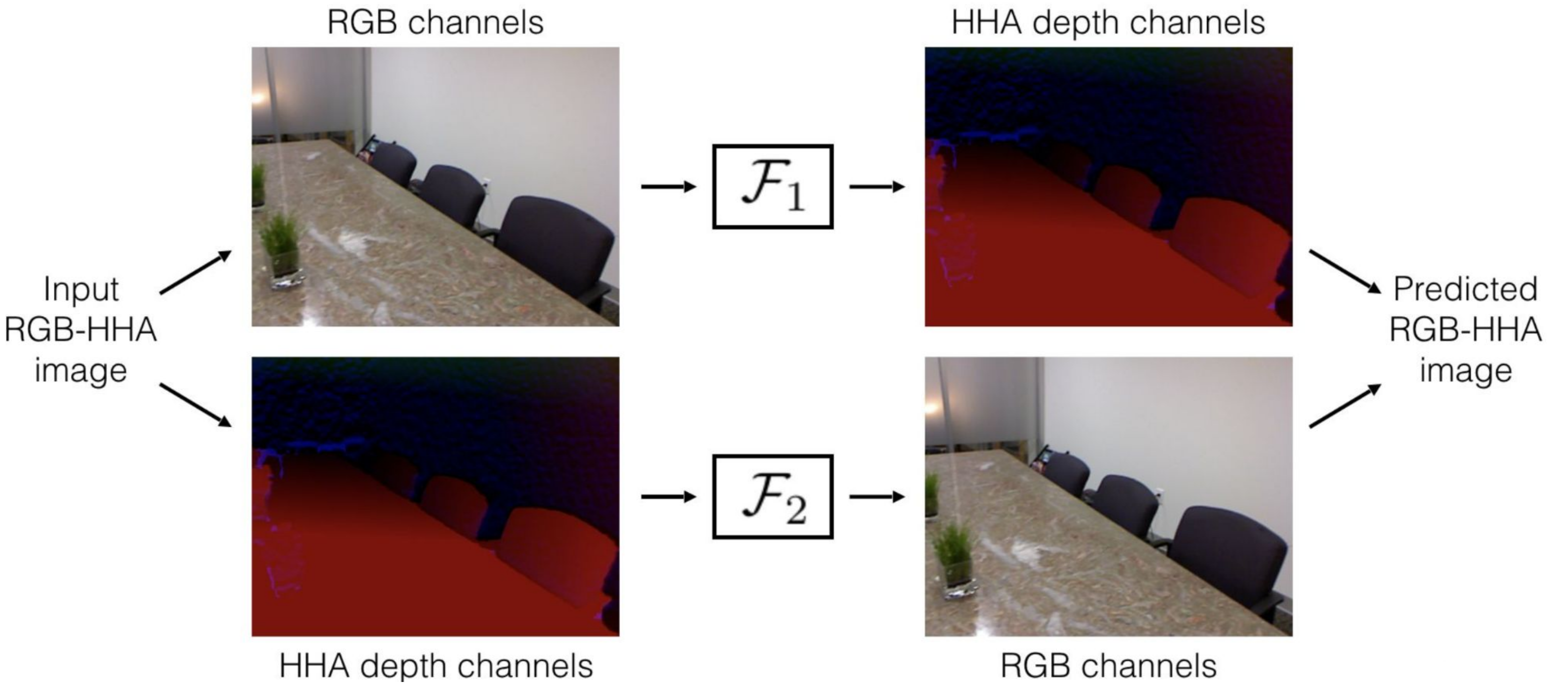
Source: Richard Zhang / Phillip Isola

Learning features from colorization: Split-brain Autoencoder



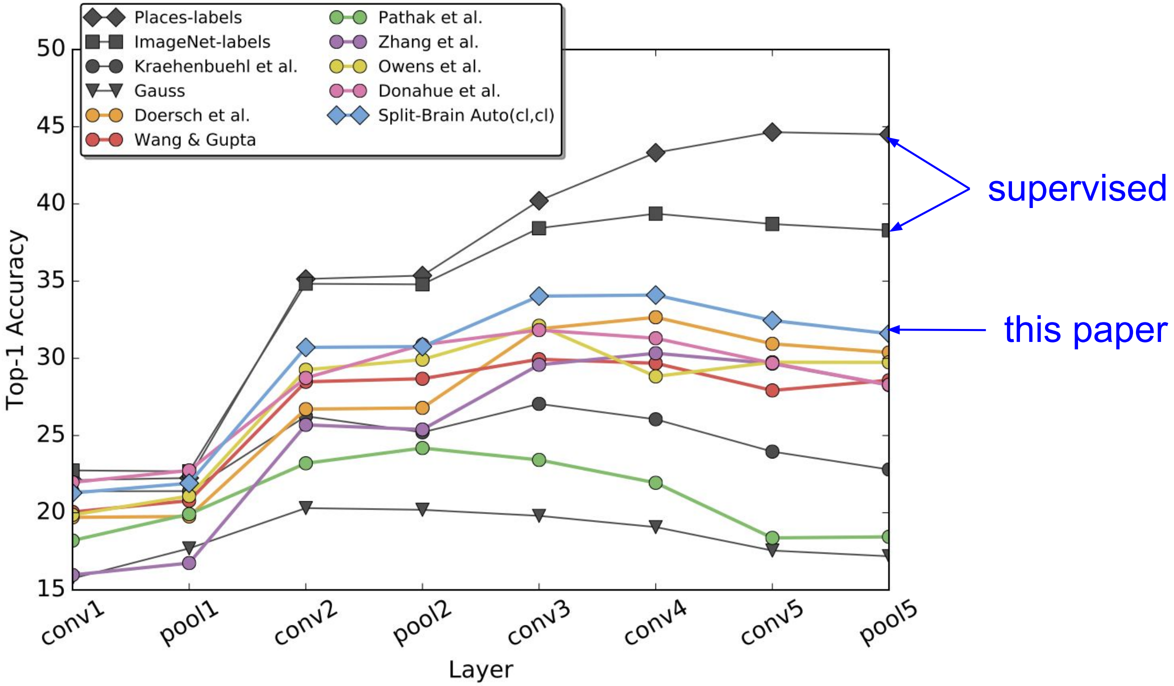
Source: Richard Zhang / Phillip Isola

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]

Real world application: 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

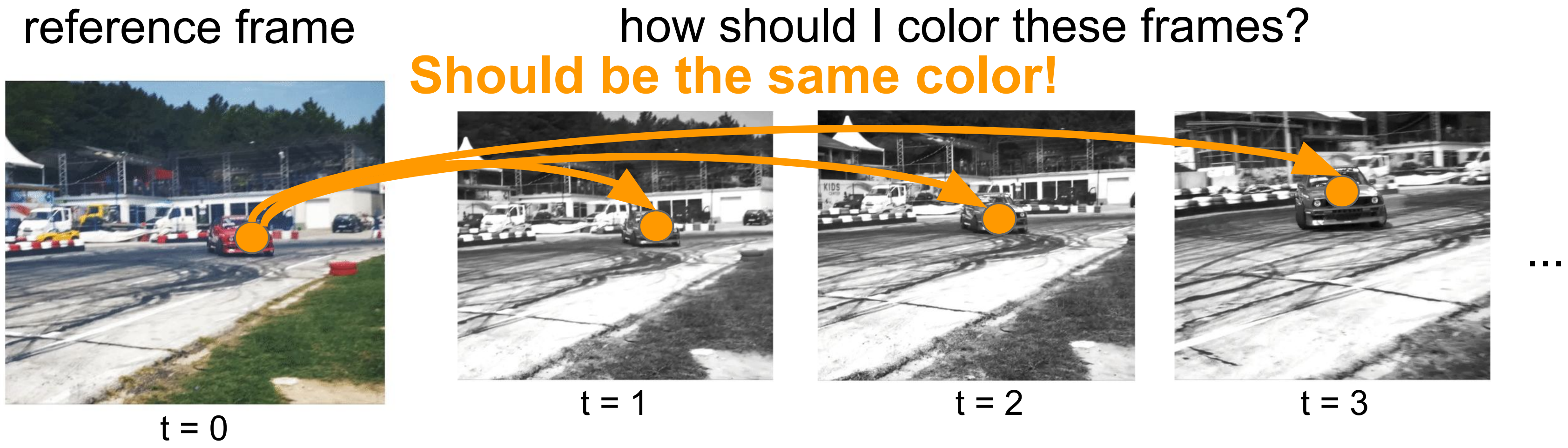
t = 3

...

Source: Vondrick et al., 2018

Pretext task: video coloring

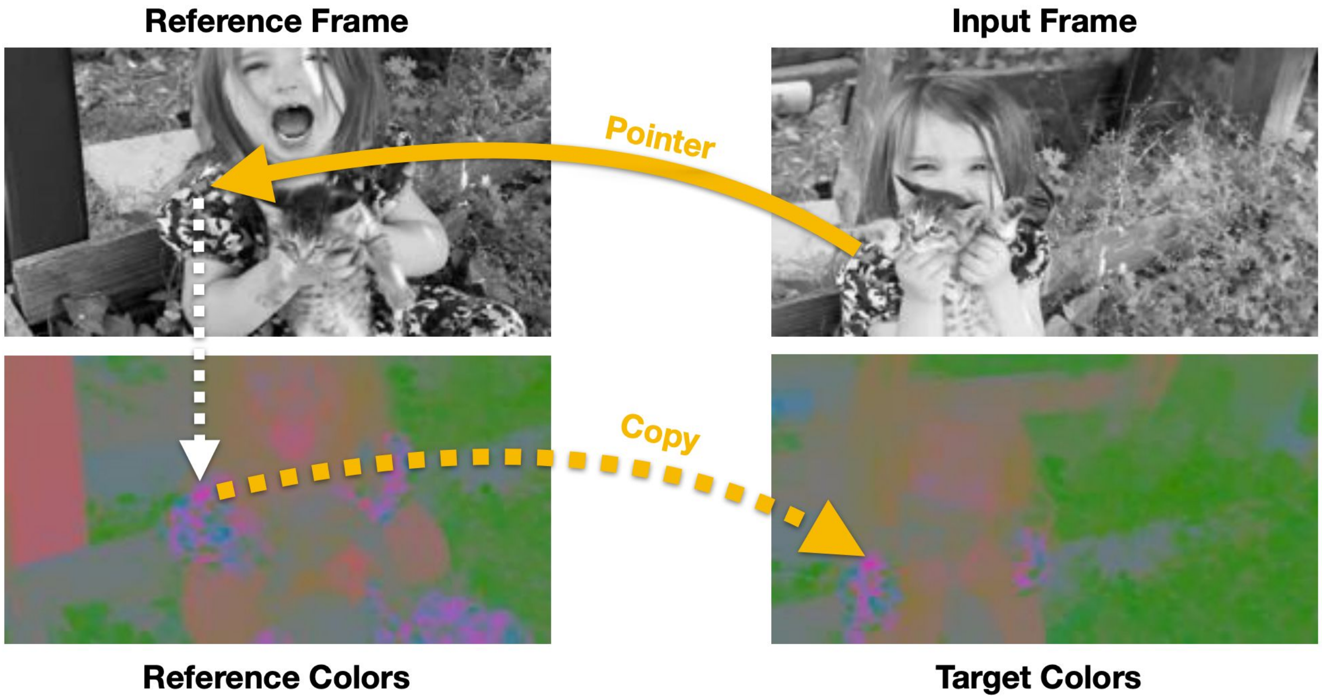
Idea: model the *temporal coherence* of colors in videos



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

Source: [Vondrick et al., 2018]

Learning to color videos



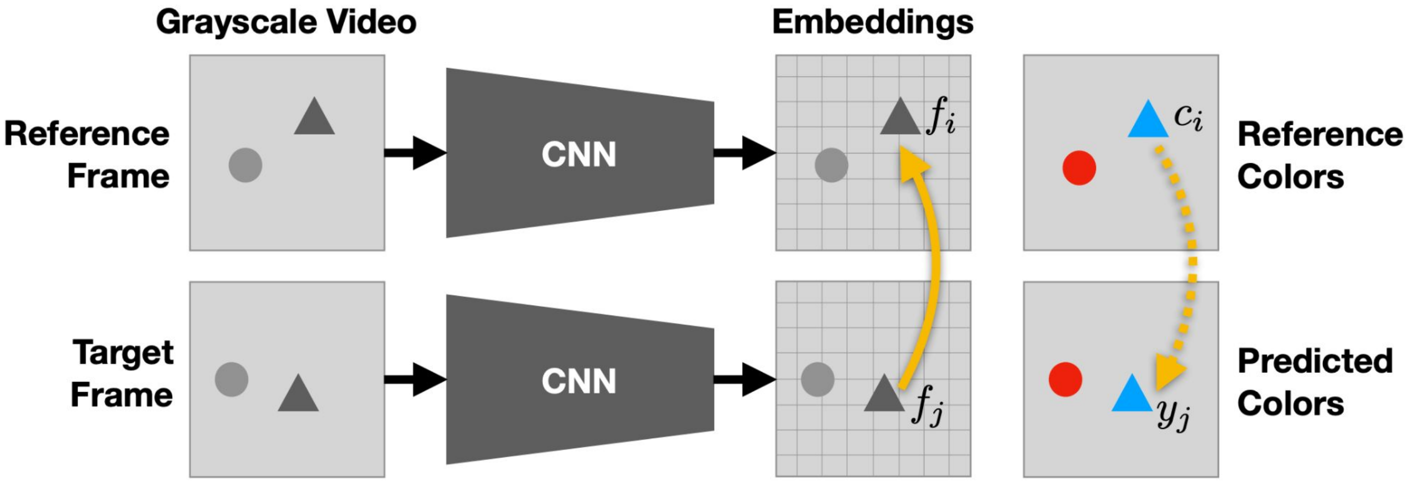
Learning objective:

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

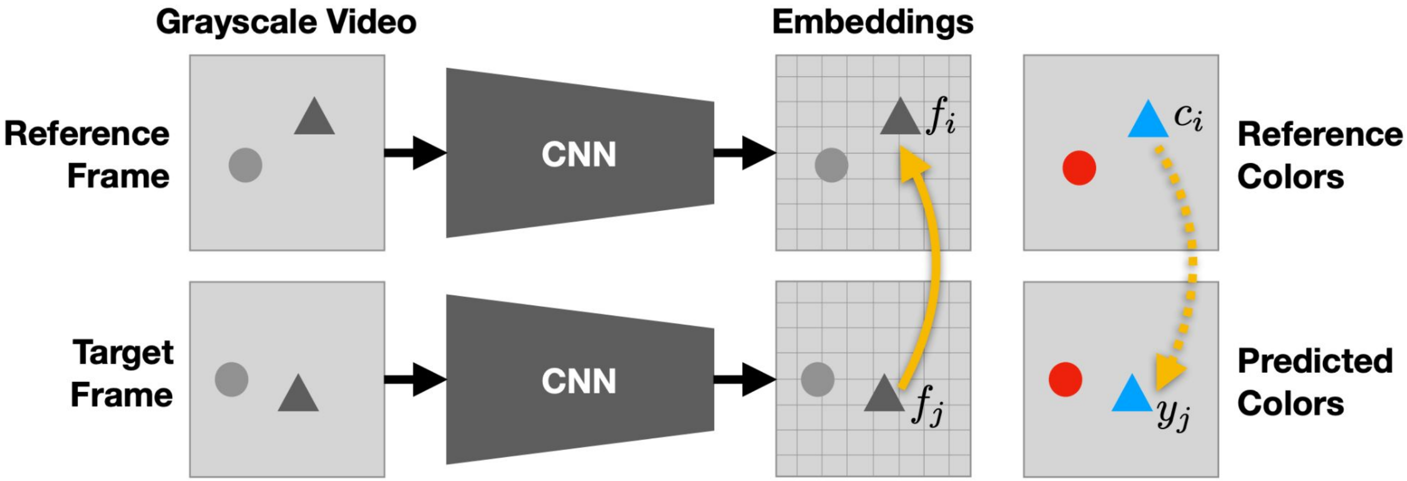


attention map on the reference frame

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

Source: [Vondrick et al., 2018]

Learning to color videos



attention map on the reference frame

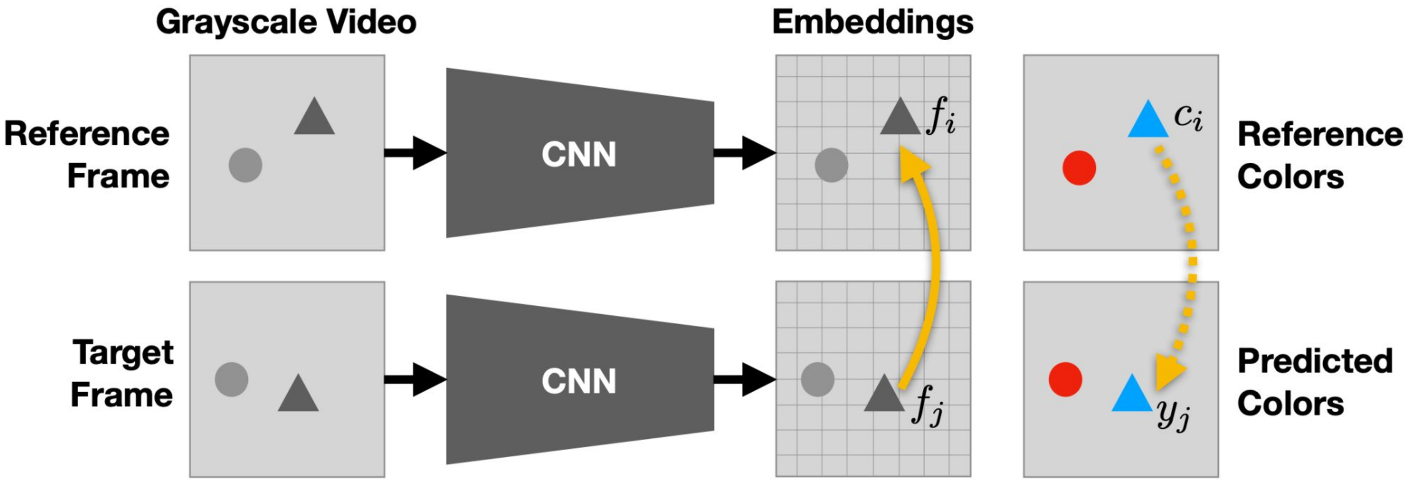
$$A_{ij} = \frac{\exp\left(f_i^T f_j\right)}{\sum_k \exp\left(f_k^T f_j\right)}$$

predicted color = weighted sum of the reference color

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

Source: [Vondrick et al., 2018]

Learning to color videos



attention map on the reference frame

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

predicted color = weighted sum of the reference color

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

loss between predicted color and ground truth color

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

Source: [Vondrick et al., 2018]

Colorizing videos (qualitative)

reference frame

target frames (gray)

predicted color

Source: [Google AI blog post]

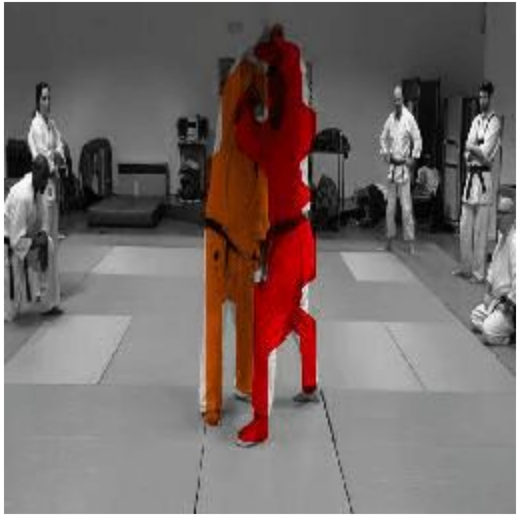
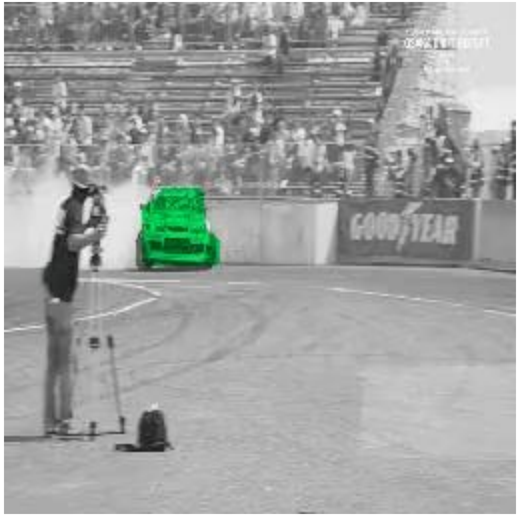
Colorizing videos (qualitative)

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 generally useful for all downstream tasks.

Pretext tasks from image transformations

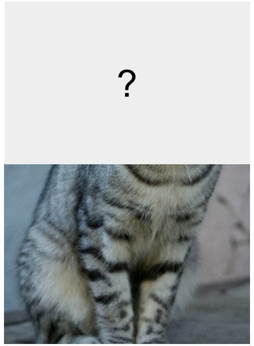
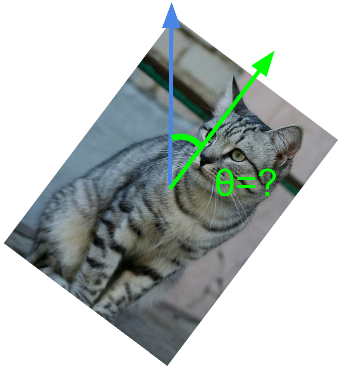


image completion



rotation prediction

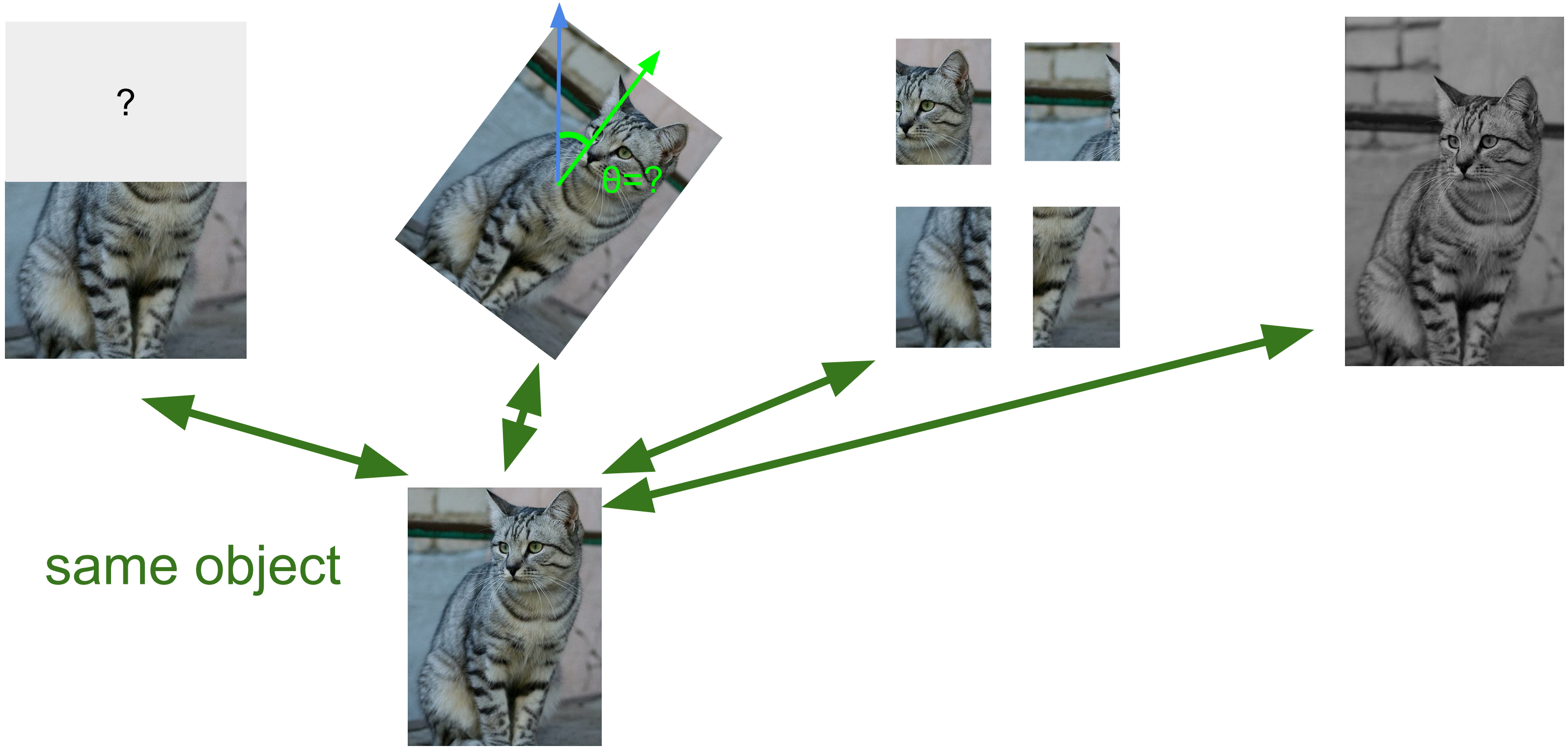
“jigsaw puzzle”

colorization

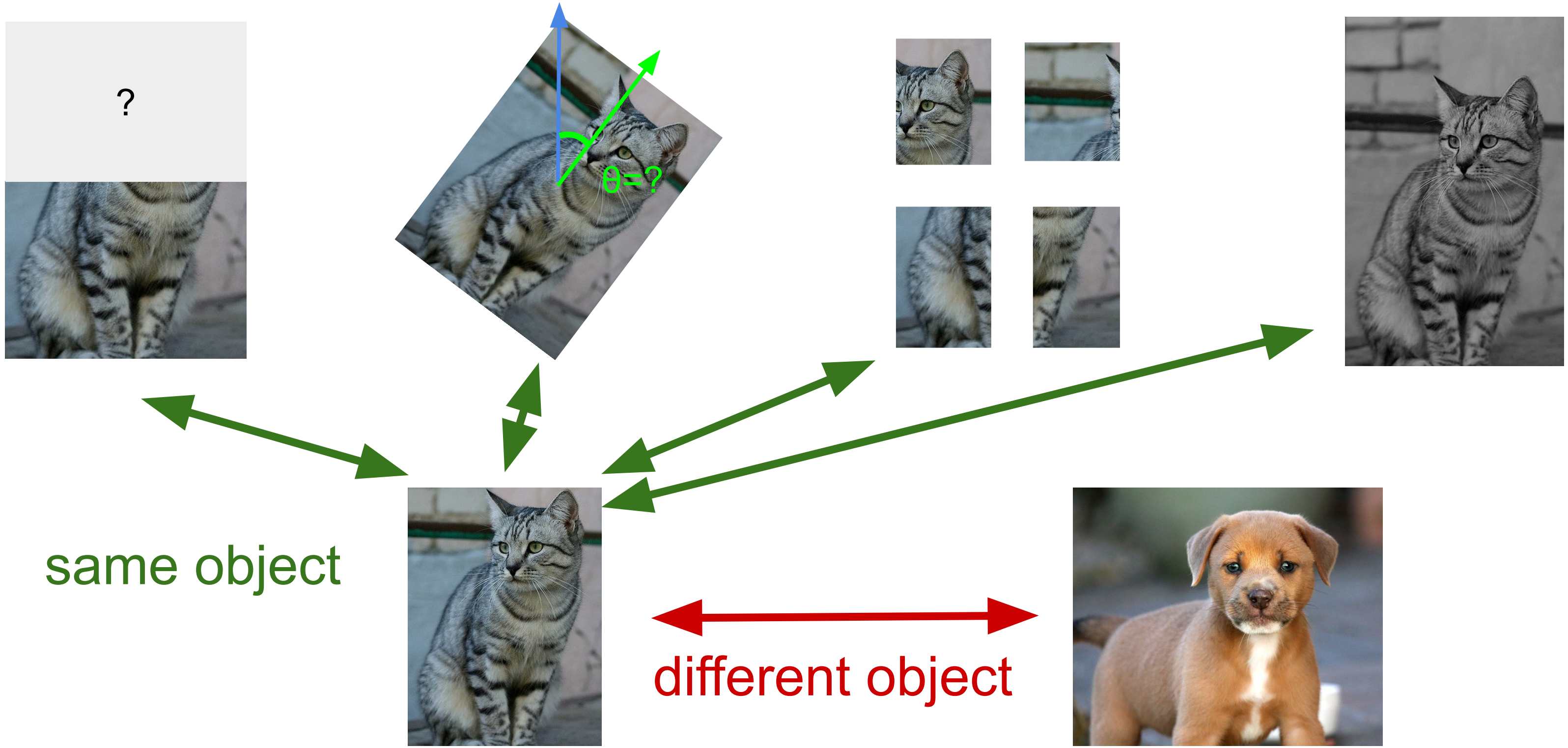
Learned representations may be tied to a specific pretext task!

Can we come up with a more general pretext task?

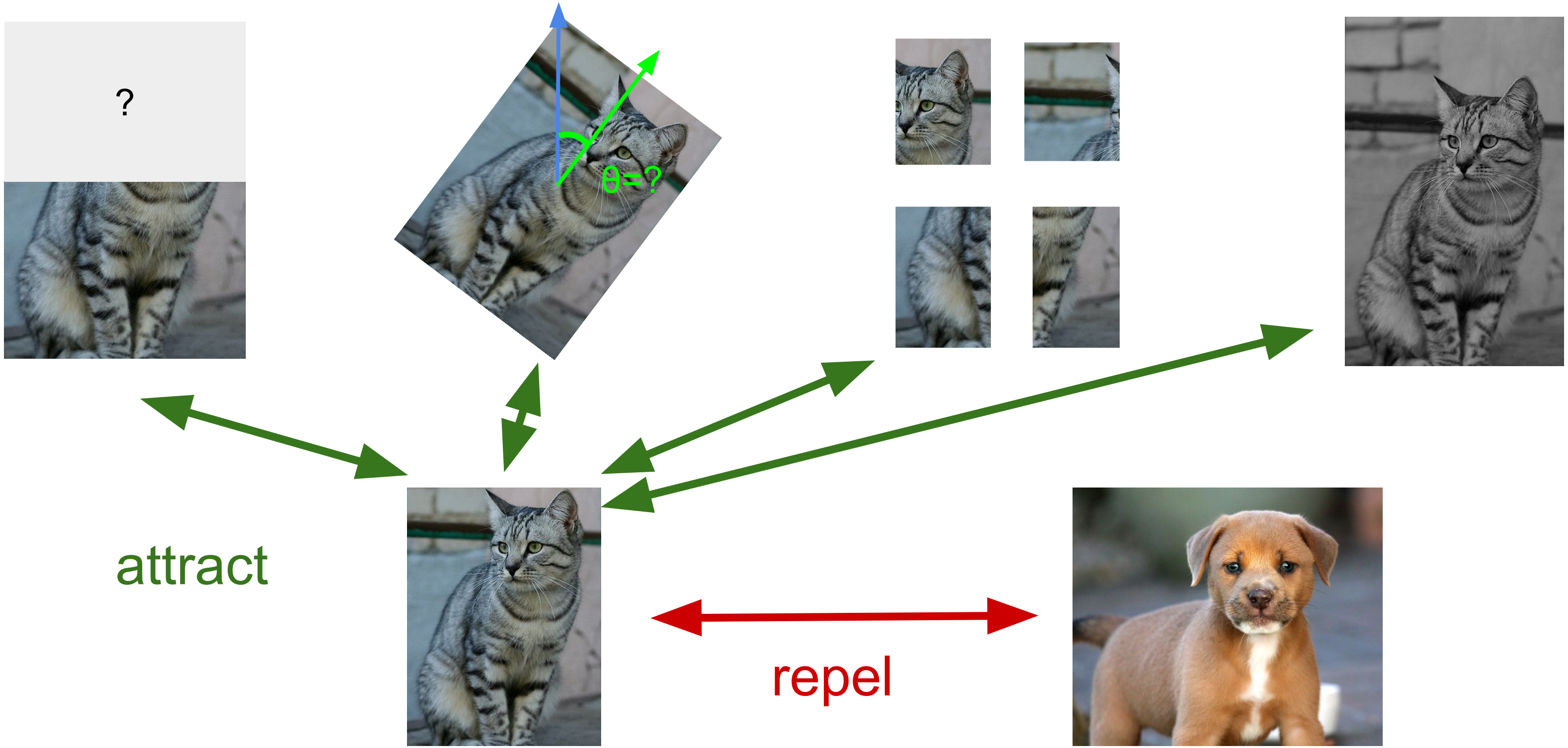
A more general pretext task?



A more general pretext task?



Contrastive Representation Learning



Today's Agenda

Pretext tasks from image transformations

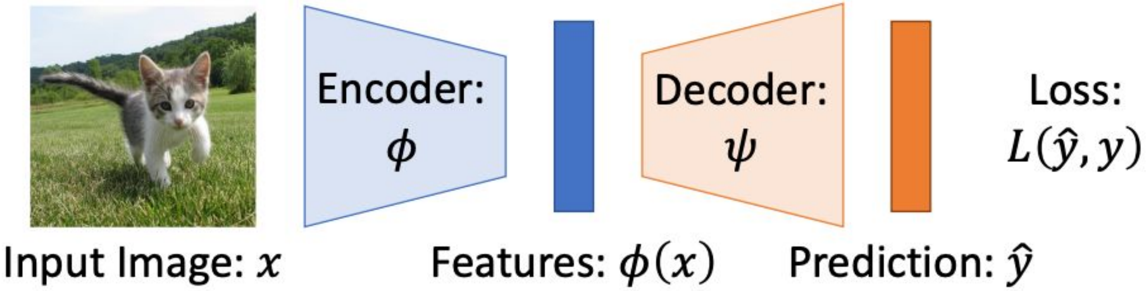
- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning

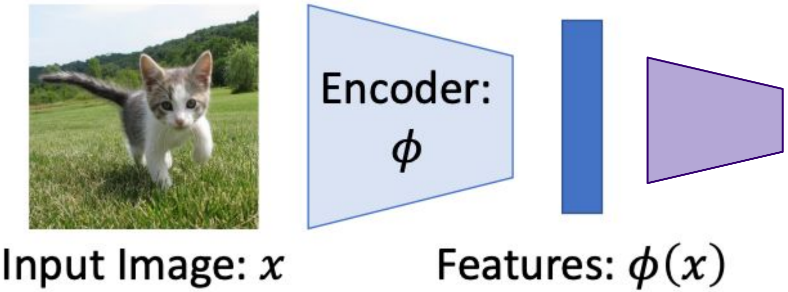
- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

Self-Supervised Learning: Pretext then Transfer

Step 1: Pretrain a network on a pretext task that doesn't require supervision



Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

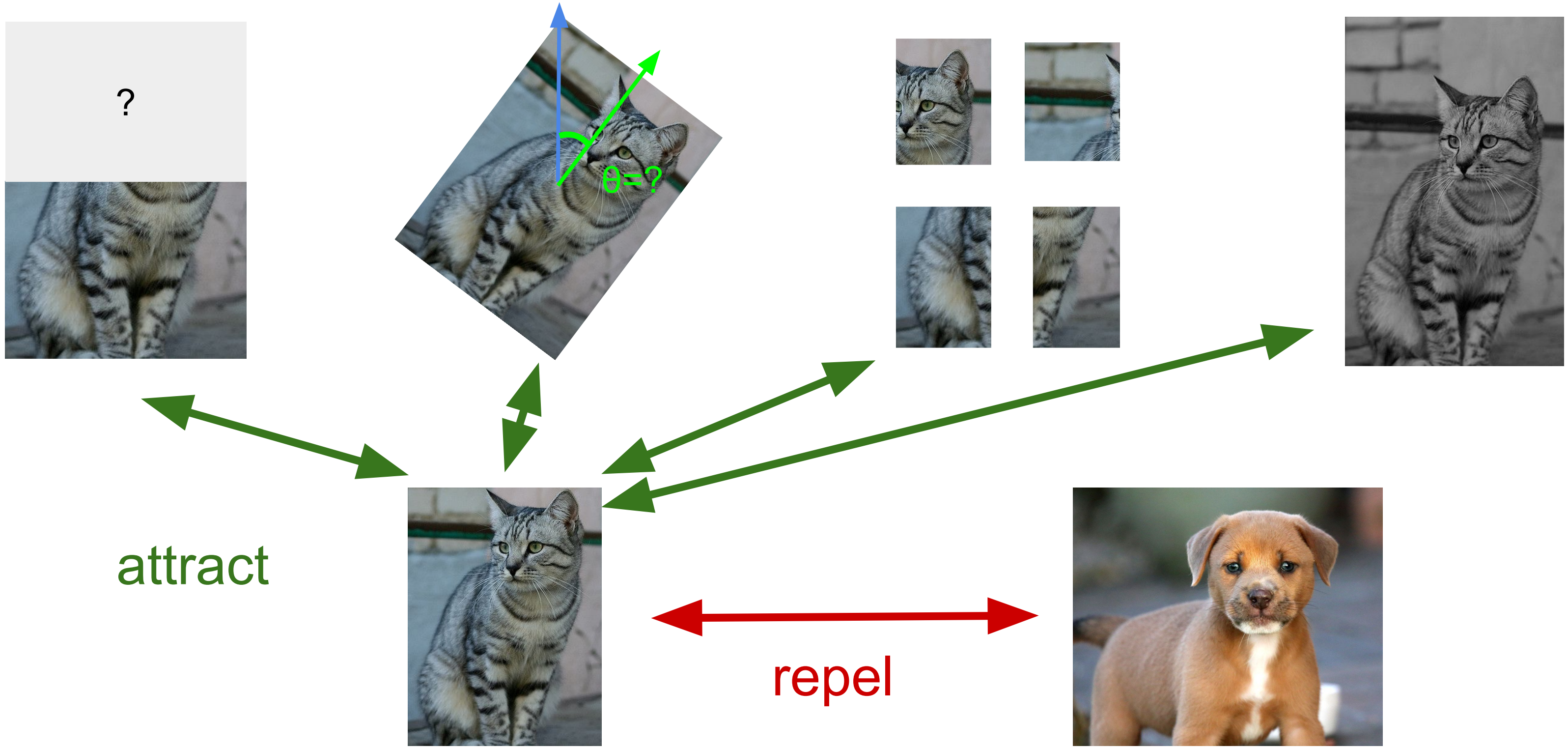


Downstream tasks:
Image classification, object detection, semantic segmentation

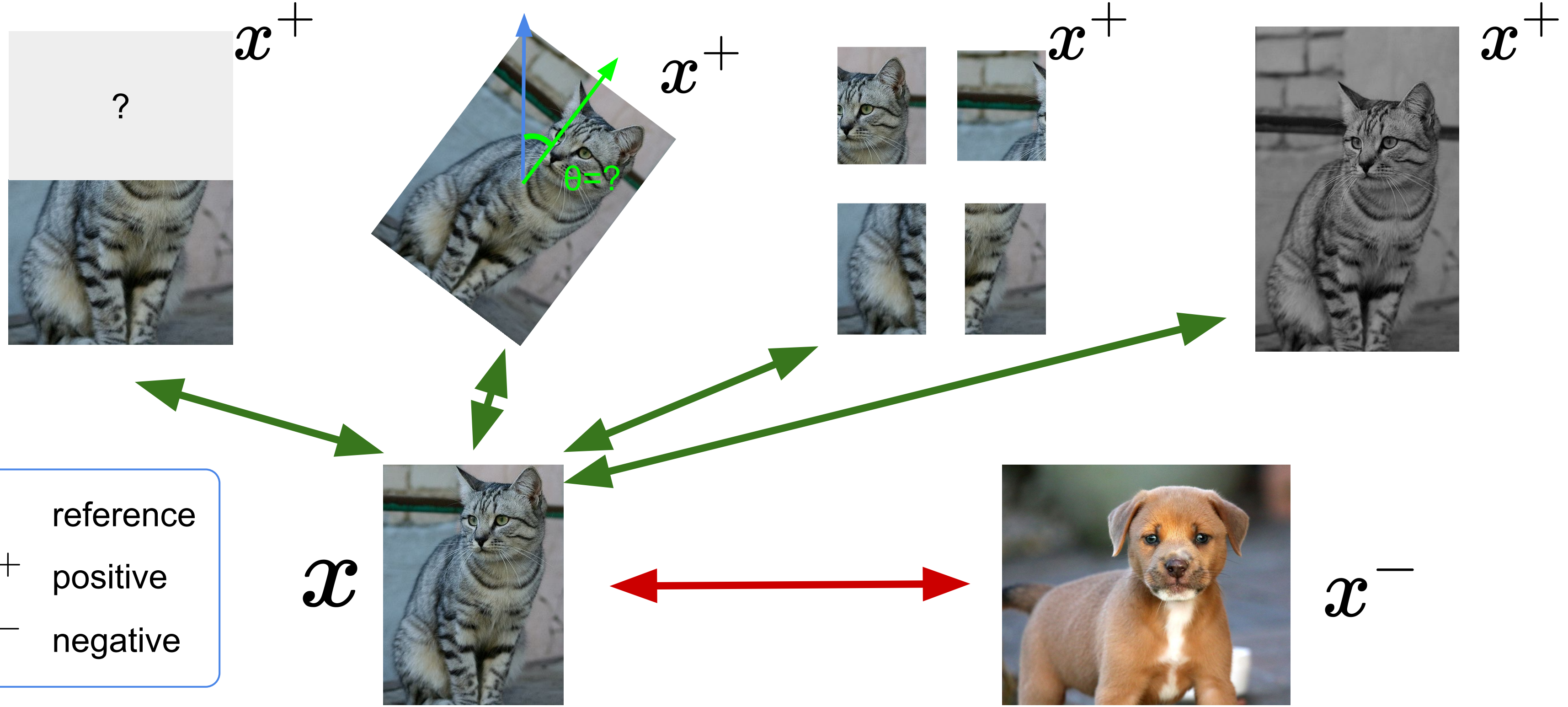
Summary: pretext tasks from image transformations

- 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 generally useful for all downstream tasks.

Contrastive Representation Learning



Contrastive Representation Learning



x	reference
x^+	positive
x^-	negative

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:

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

Q. What does this loss function remind you of?

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:

$$L = -\mathbb{E}_X \left[\log \frac{\overline{\exp(s(f(x), f(x^+))}}{\underbrace{\exp(s(f(x), f(x^+))}_{\text{score for the positive pair}} + \underbrace{\sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-))}_{\text{score for the N-1 negative pairs}}} \right]$$

This seems familiar …

A formulation of contrastive learning

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

$$L = -\mathbb{E}_X \left[\log \frac{\overline{\exp(s(f(x), f(x^+))}}{\underbrace{\exp(s(f(x), f(x^+))}_{\text{score for the positive pair}} + \underbrace{\sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-))}_{\text{score for the N-1 negative pairs}}} \right]$$

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:

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

Very similar to the softmax classifier we talked about a few lectures ago.

- We want to compare the reference image against all other positive and negative images.
- We can exponentiate and normalize these scores like we did with the softmax classifier.
- And we get the above similar equation.

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

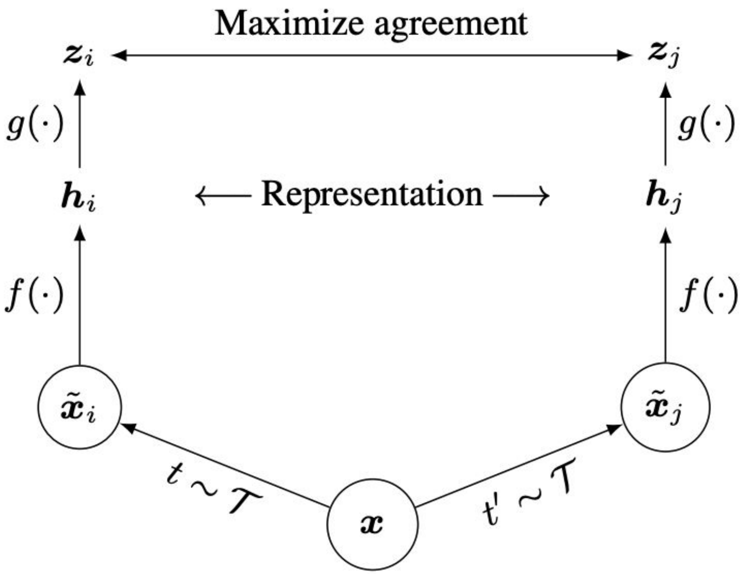
Cosine similarity as the score function:

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

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

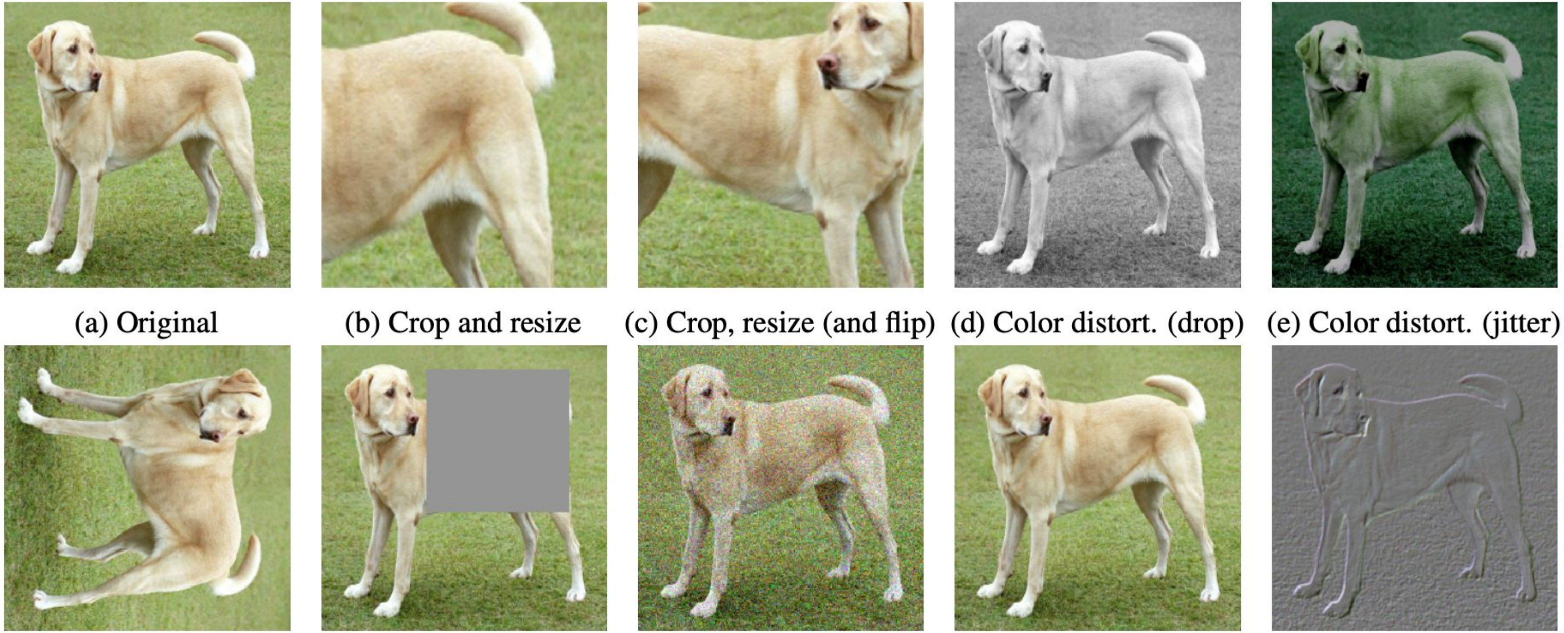
Generate positive samples through data augmentation:

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

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.

Source: Chen et al., 2020

SimCLR

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

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**
    $\boxed{\text{draw two augmentation functions } t \sim \mathcal{T}, t' \sim \mathcal{T}}$
    # the first augmentation
    $\boxed{\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)}$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\boxed{\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)}$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # 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** $\boxed{\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  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

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

Source: Chen et al., 2020

SimCLR

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

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 $k \in \{1, \dots, N\}$ **do**
    $\boxed{\text{draw two augmentation functions } t \sim \mathcal{T}, t' \sim \mathcal{T}}$
    # the first augmentation
    $\boxed{\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)}$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\boxed{\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)}$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \dots, 2N\}$ and $j \in \{1, \dots, 2N\}$ **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** $\boxed{\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  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

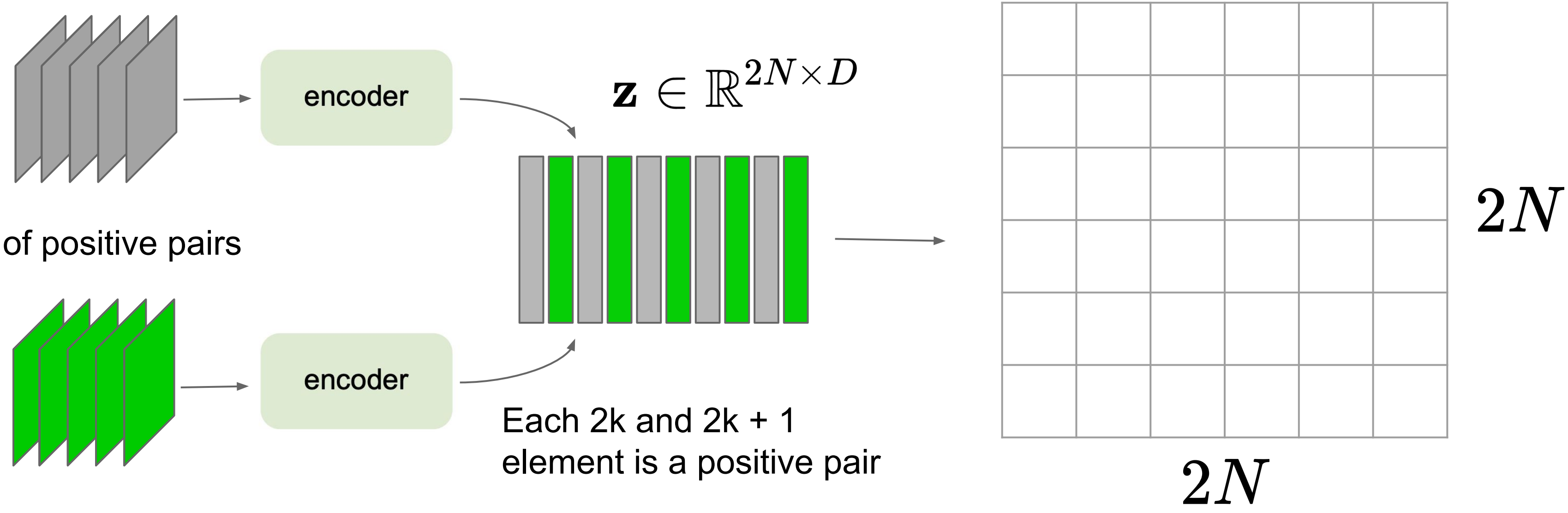
InfoNCE loss:
Use all non-positive samples in the batch as x^-

Source: [Chen et al., 2020]

SimCLR: mini-batch training

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

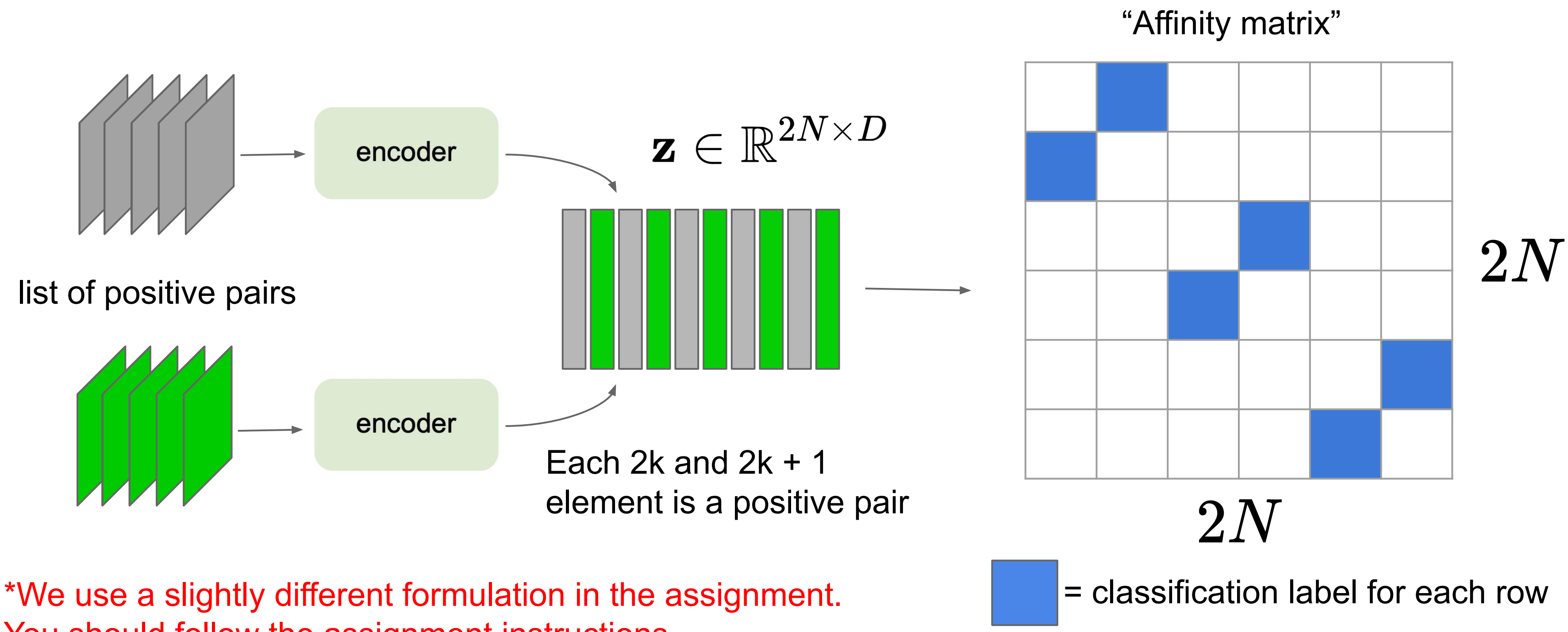
"Affinity matrix"



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

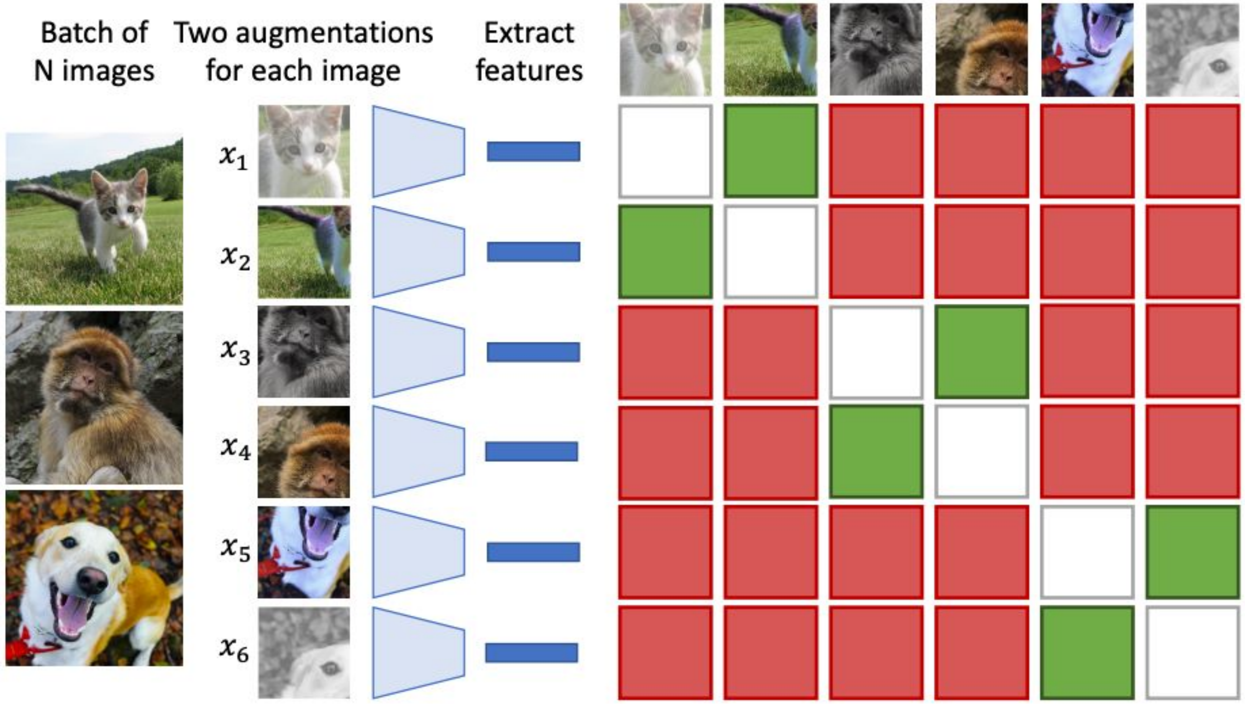
SimCLR: mini-batch training

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

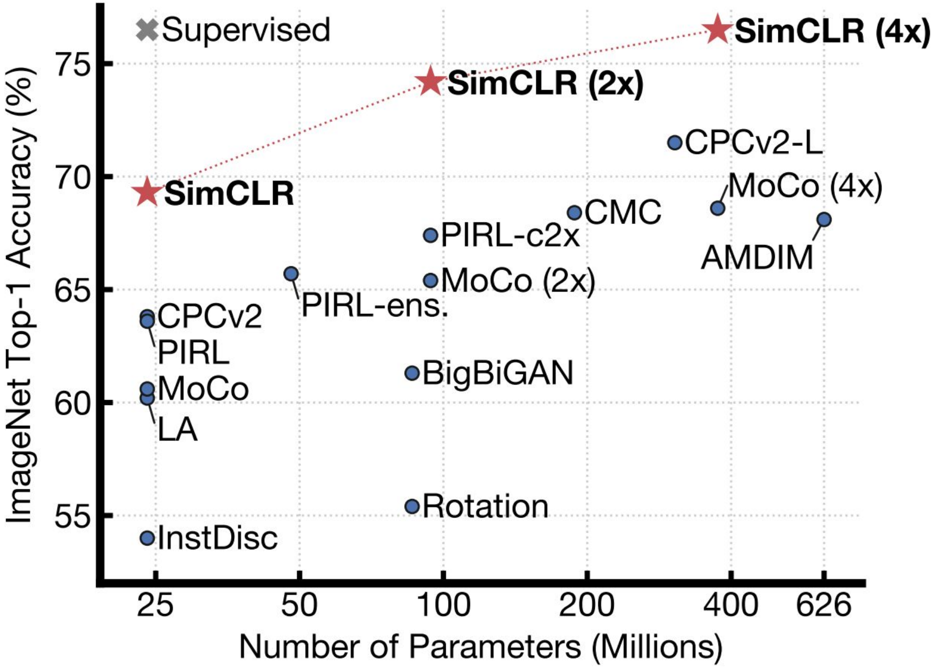


*We use a slightly different formulation in the assignment.
You should follow the assignment instructions.

SimCLR: what a batch looks like



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%	Label fraction 10%
		Top 5	
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**

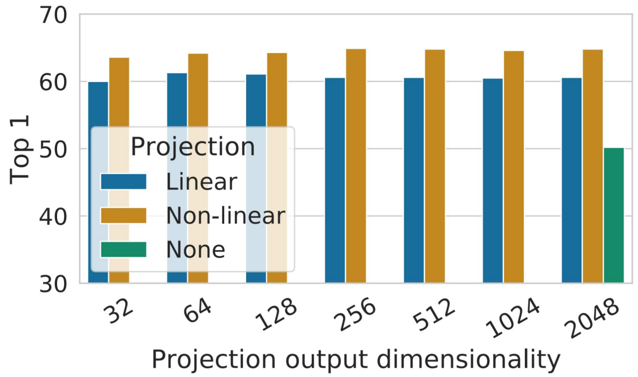
Table 7. ImageNet accuracy of models trained with few labels.

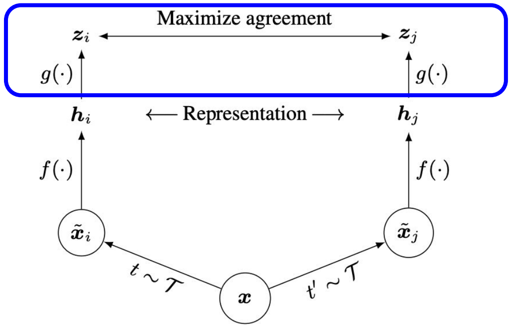
Train feature encoder on **ImageNet** (entire training set) using SimCLR.

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

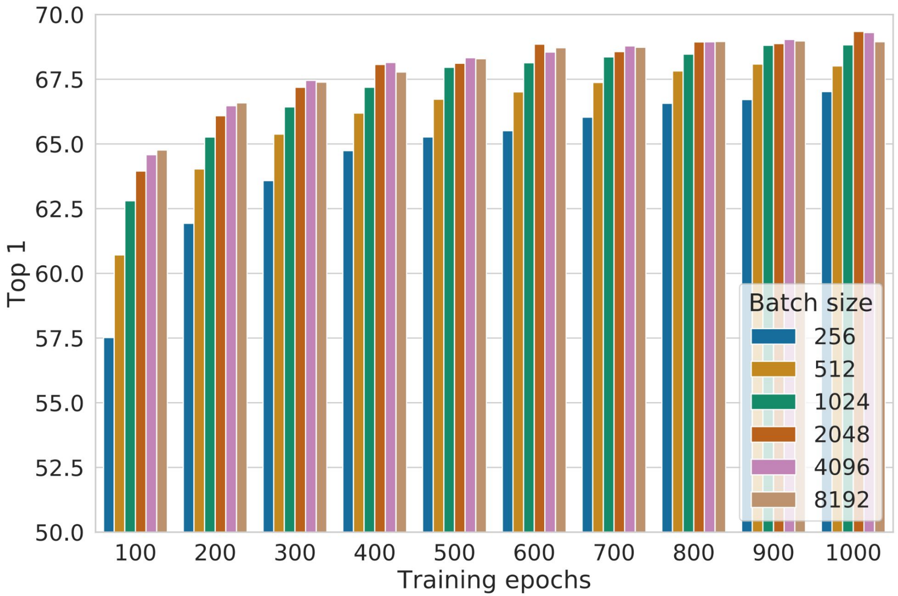


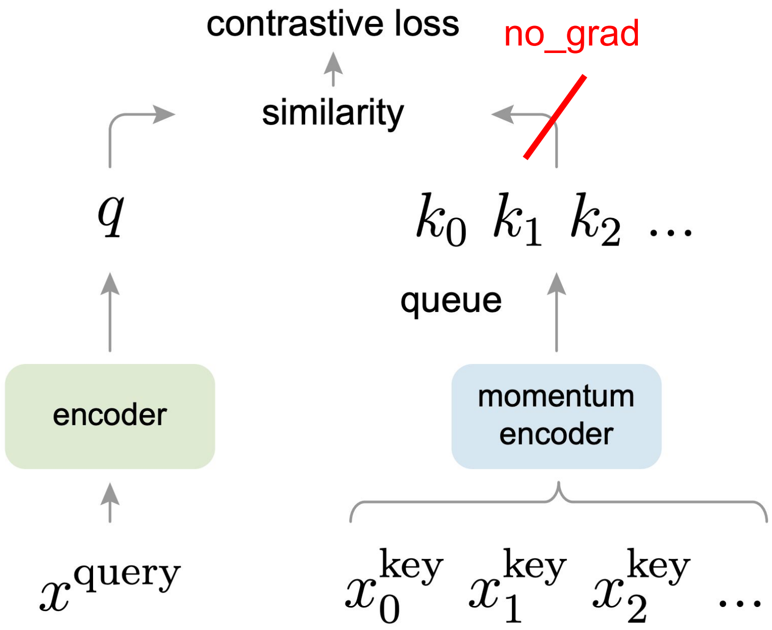
Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

Large training batch size is crucial for SimCLR!

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

Source: [Chen et al., 2020]

Momentum Contrastive Learning (MoCo)

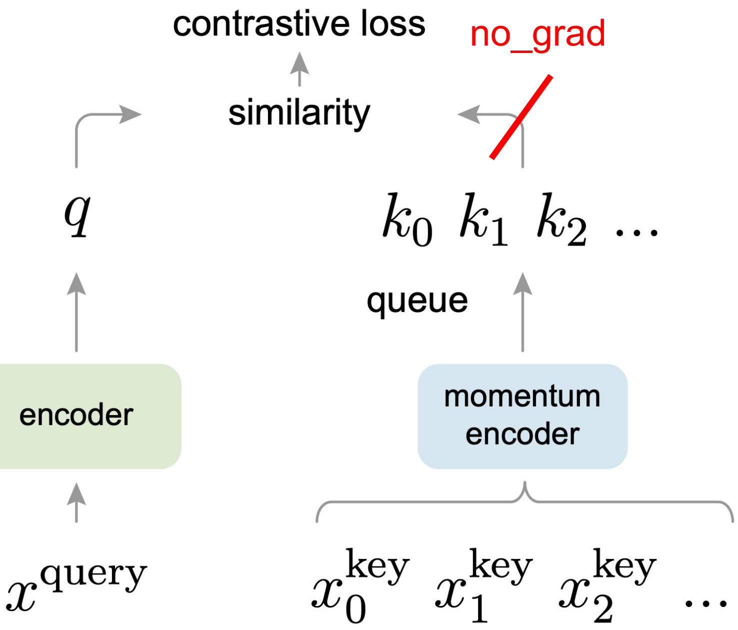


Key differences to SimCLR:

- 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:

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

Source: He et al., 2020

MoCo

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.

Generate a positive pair by sampling data augmentation functions

No gradient through the negative samples

Use the running queue of keys as the negative samples

InfoNCE loss

Update f_k through momentum

Update the FIFO negative sample queue

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

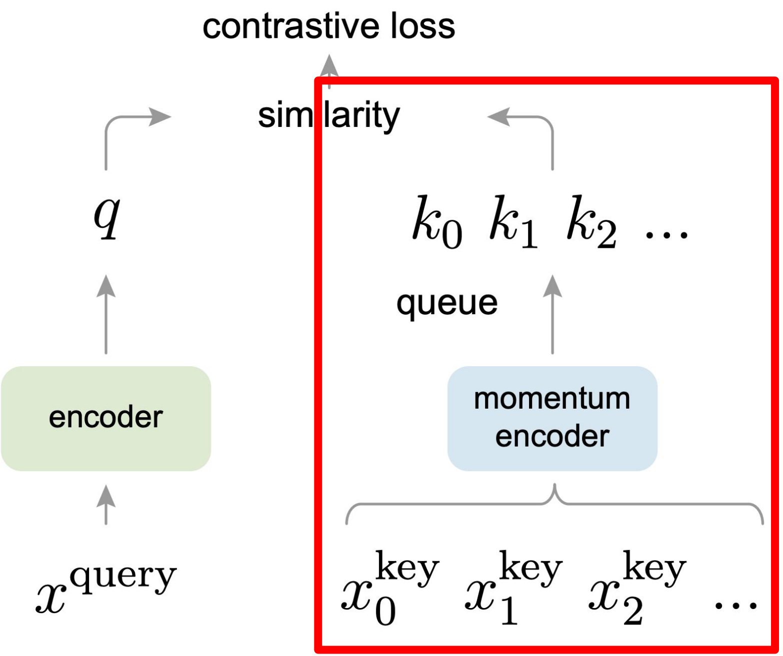
Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

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)

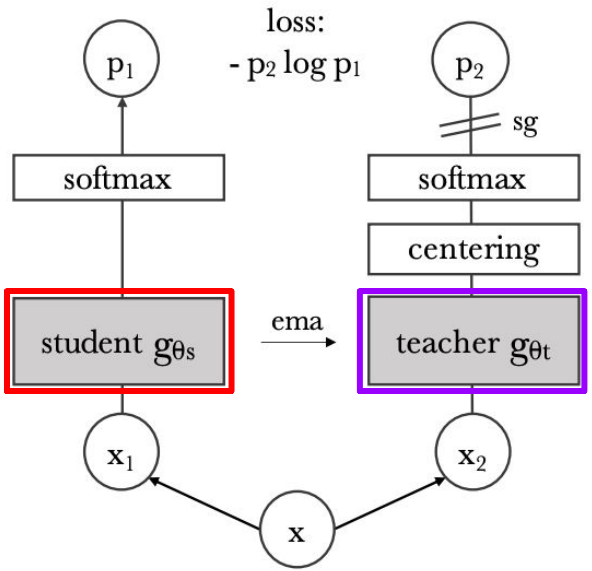
Source: Chen et al., 2020

Problem with MoCoV2: Need to keep around a set of negatives



Do we need these negatives?

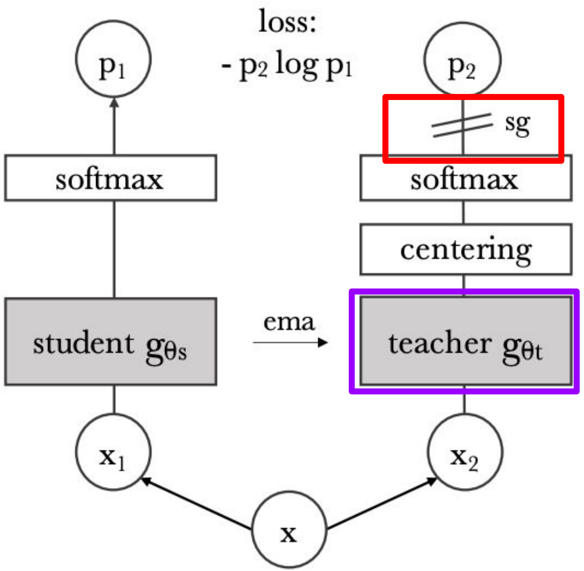
Solution: DINO: self-distillation with no labels



- Similar to SimCLR and MOCO but with one big difference: no negatives
- Reformulates contrastive learning as knowledge distillation between a student and a teacher model.

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

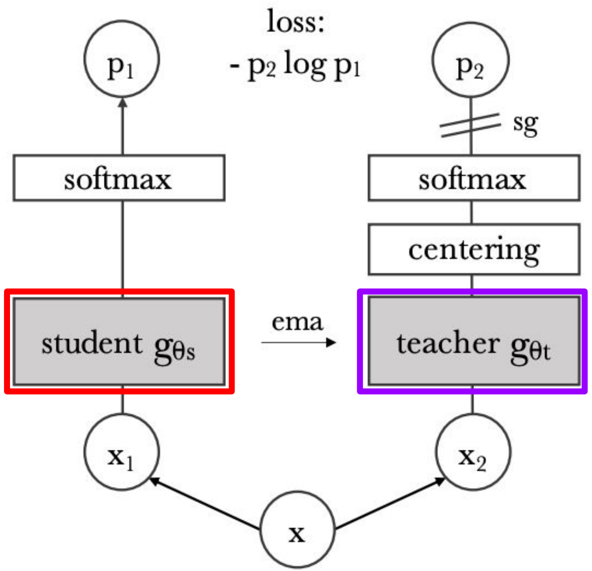
Solution: DINO: self-distillation with no labels



- The teacher model is not trained: sg stands for stop-gradient: meaning that gradients are prevented from flowing back.

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

Problem: But how do we choose the teacher model?



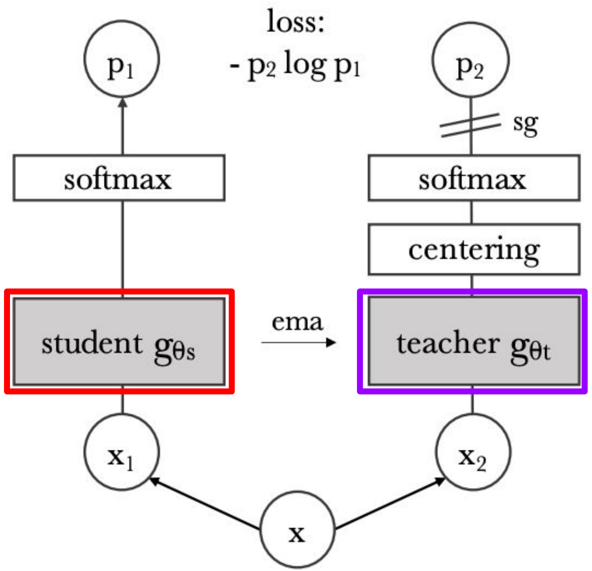
- The teacher model is like the momentum encoder. It is a running average of the student model

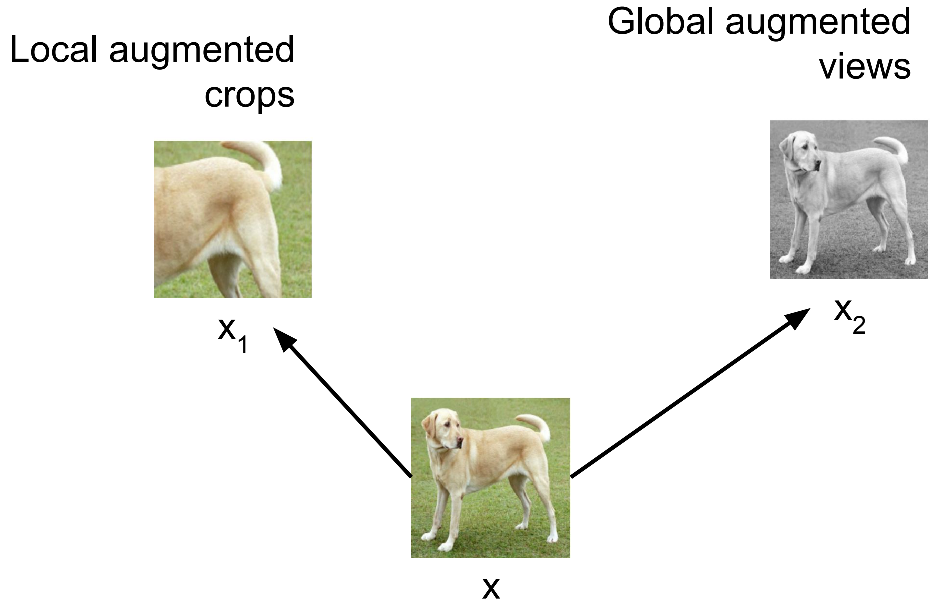
$$\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s$$

- The teacher sees a global view augmentation of the image
- Student only sees augmented local crops of the image

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

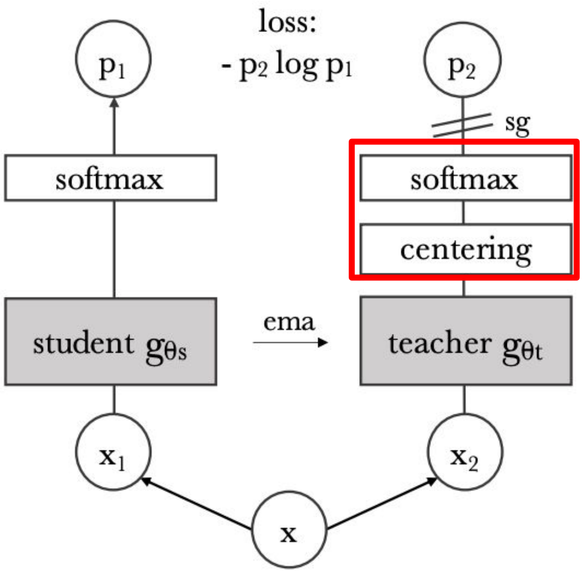
Problem: But how do we choose the teacher model?





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

Problem: But how do we choose the teacher model?



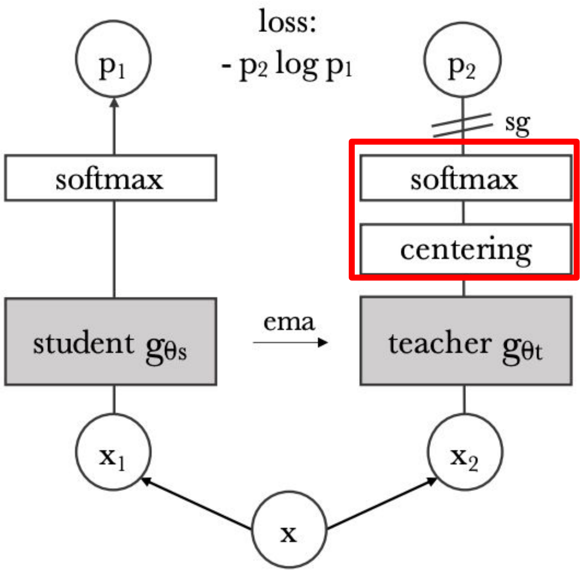
Training tricks:

- **Centering**: prevents one dimension from dominating.
  - A constant value c is added to all dimensions of the teacher's output.
  - c is a running average of outputs

$$g_t(x) \leftarrow g_t(x) + c, \; c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i)$$

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

Problem: But how do we choose the teacher model?



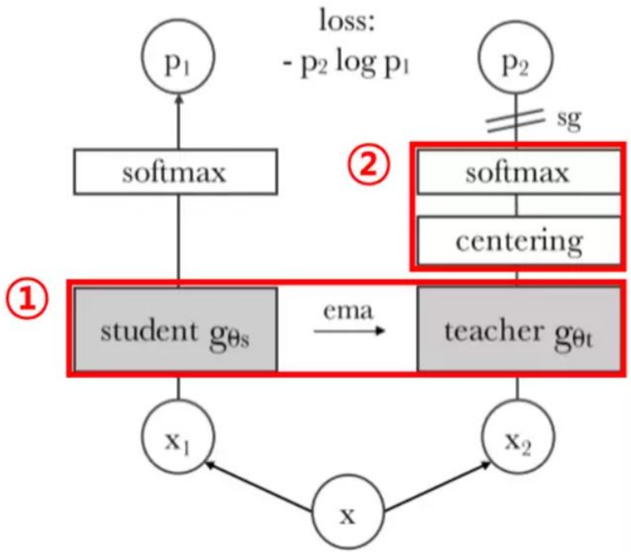
Training tricks:

- **Sharpening**: Opposite of centering.
  - A temperature (Tau) hyperparameter is used to sharpen the distributions towards one dimension.

$$\frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K}\exp(g_{\theta_s}(x)^{(k)}/\tau_s)}$$

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

DINO code



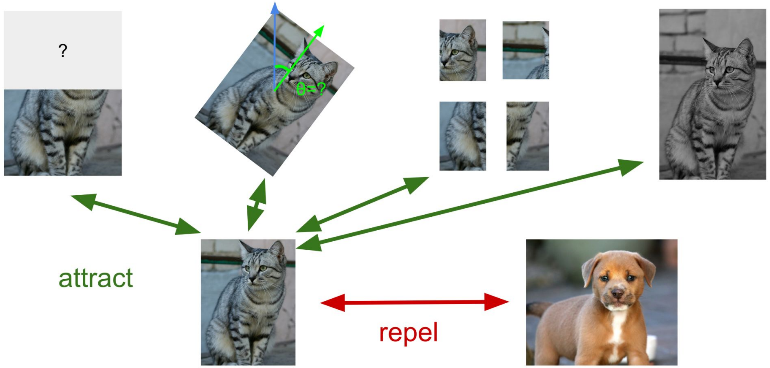
Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

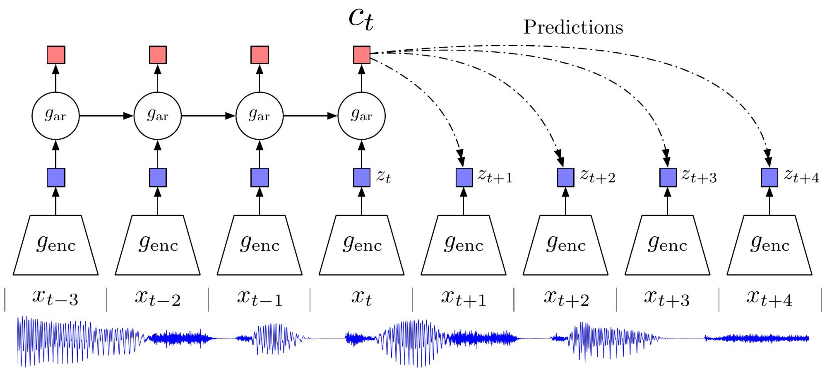
```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    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()
```

Results: DINO. **Also DINO V2 just released last week**

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [12]	RN50	23	1237	69.1	60.7
MoCov2 [15]	RN50	23	1237	71.1	61.9
InfoMin [67]	RN50	23	1237	73.0	65.3
BarlowT [81]	RN50	23	1237	73.2	66.0
OBoW [27]	RN50	23	1237	73.8	61.9
BYOL [30]	RN50	23	1237	74.4	64.8
DCv2 [10]	RN50	23	1237	75.2	67.1
SwAV [10]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**

Instance vs. Sequence Contrastive Learning



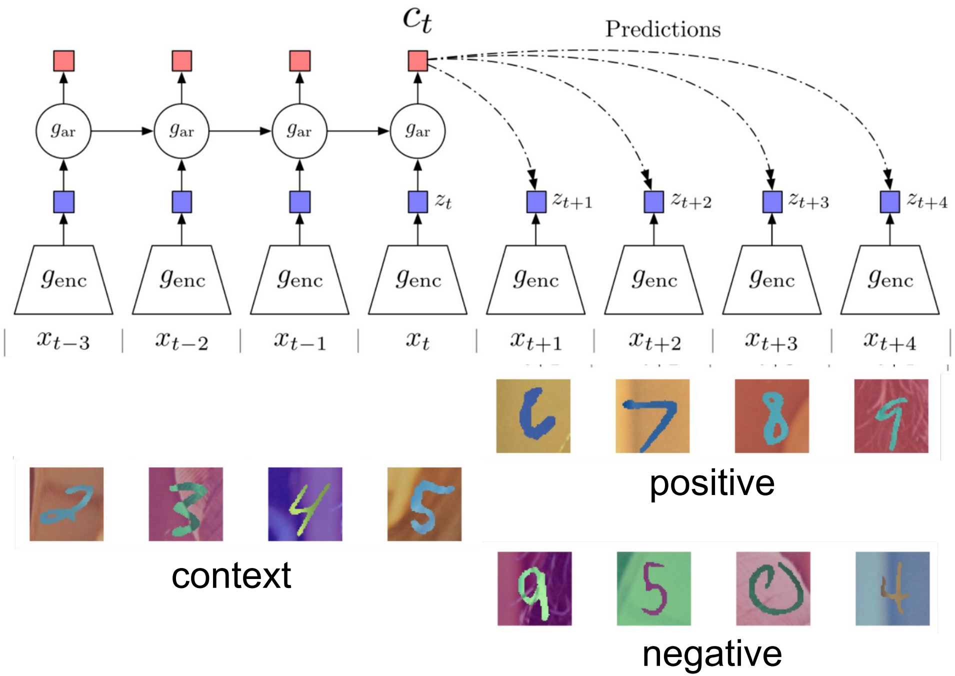


Source: [van den Oord et al., 2018]

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

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)

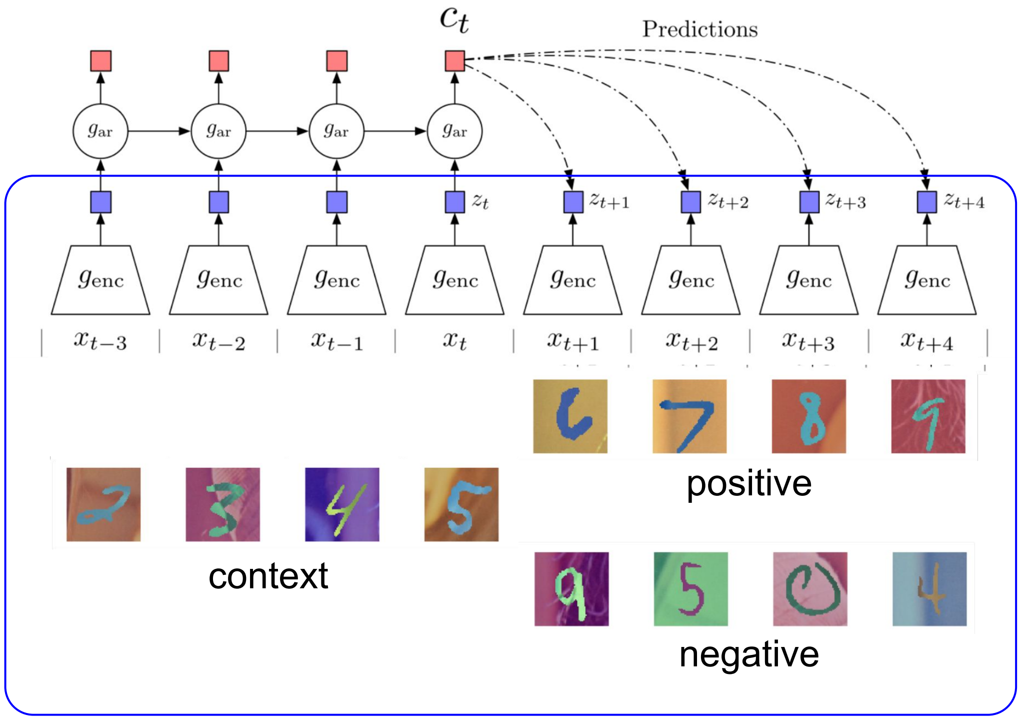
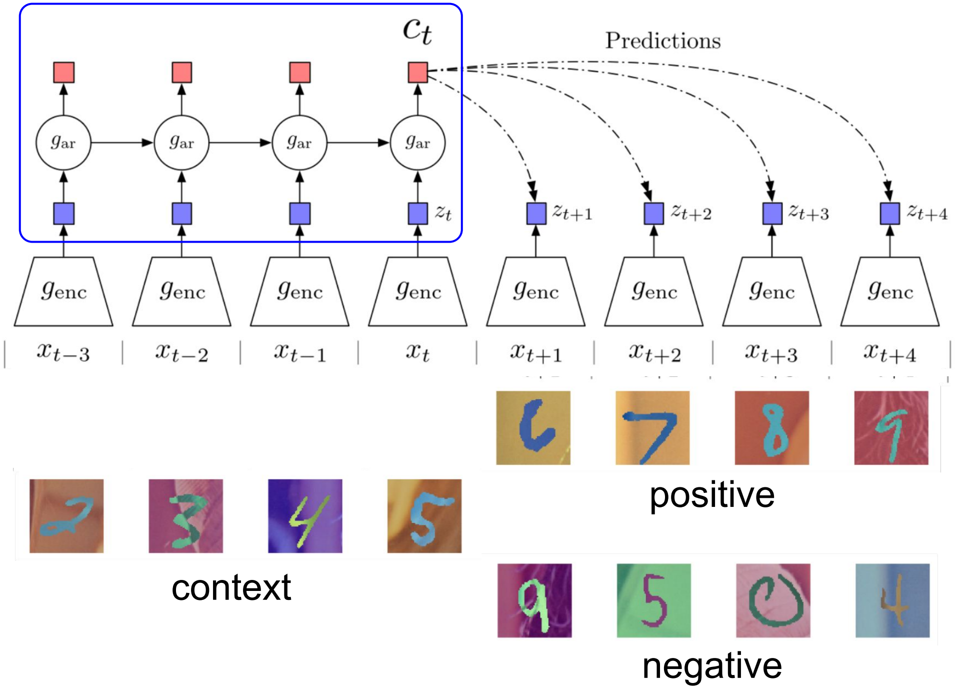


Figure source

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

Source: van den Oord et al., 2018,

Contrastive Predictive Coding (CPC)



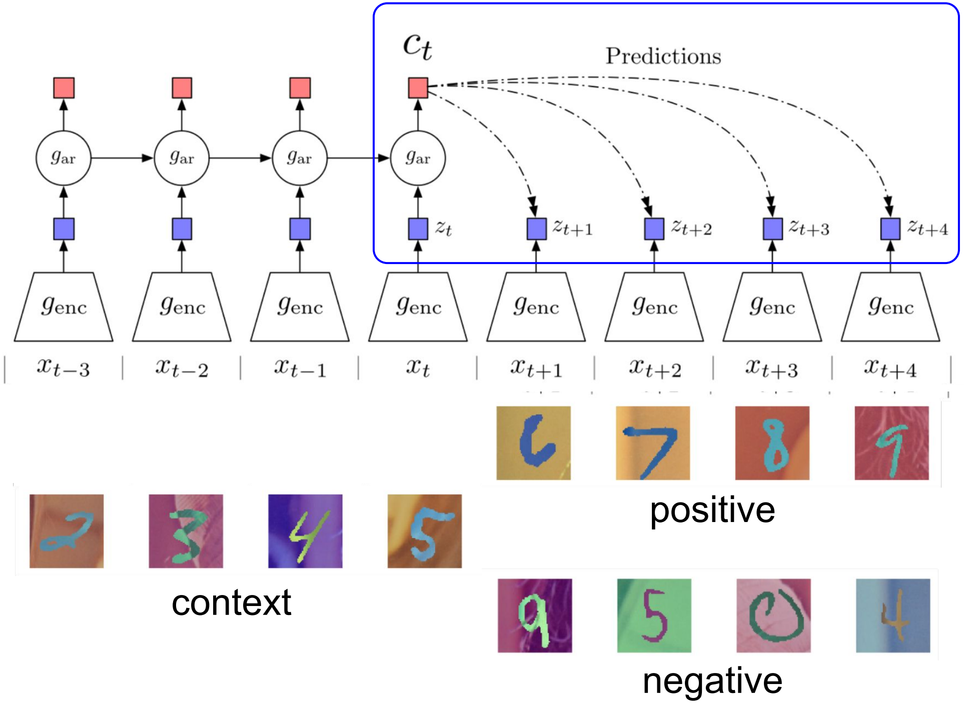
1. Encode all samples in a sequence into vectors $\boldsymbol{z_t = g_{enc}(x_t)}$

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

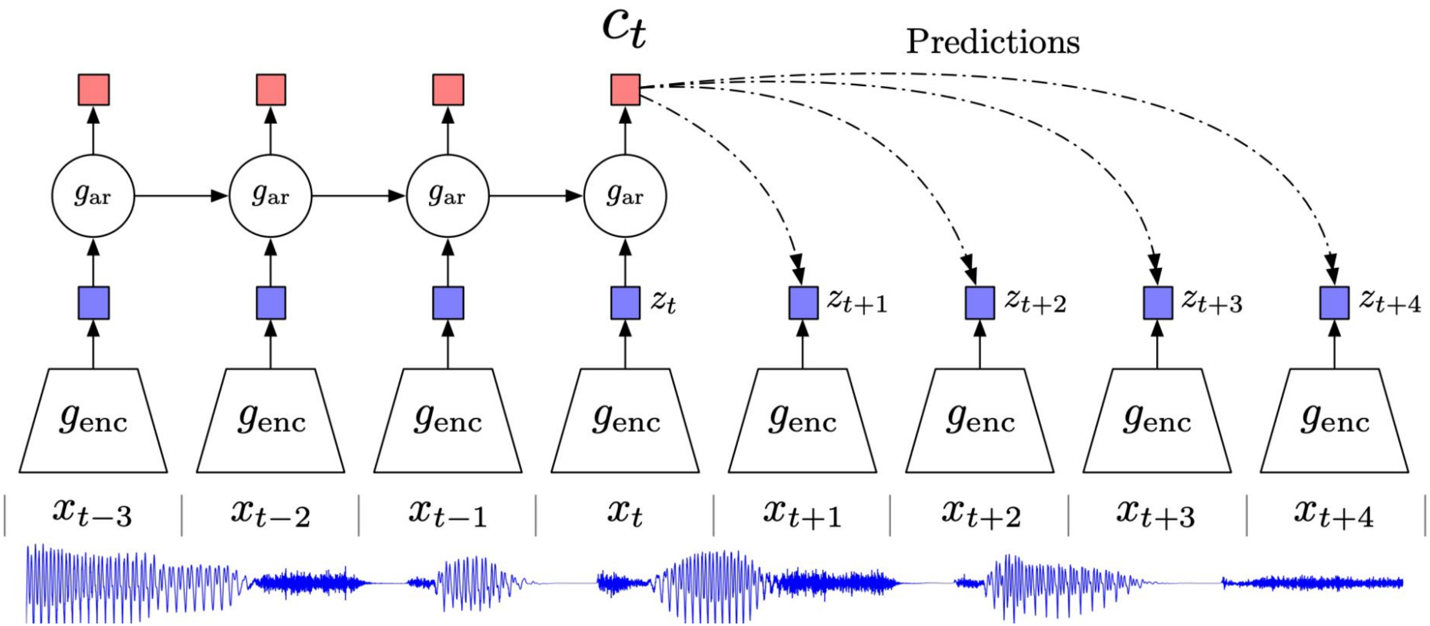
$$s_k(z_{t+k}, c_t) = z_{t+k}^T W_k c_t$$

, 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

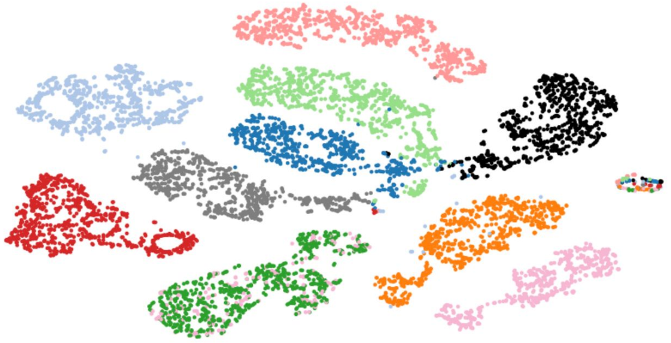


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

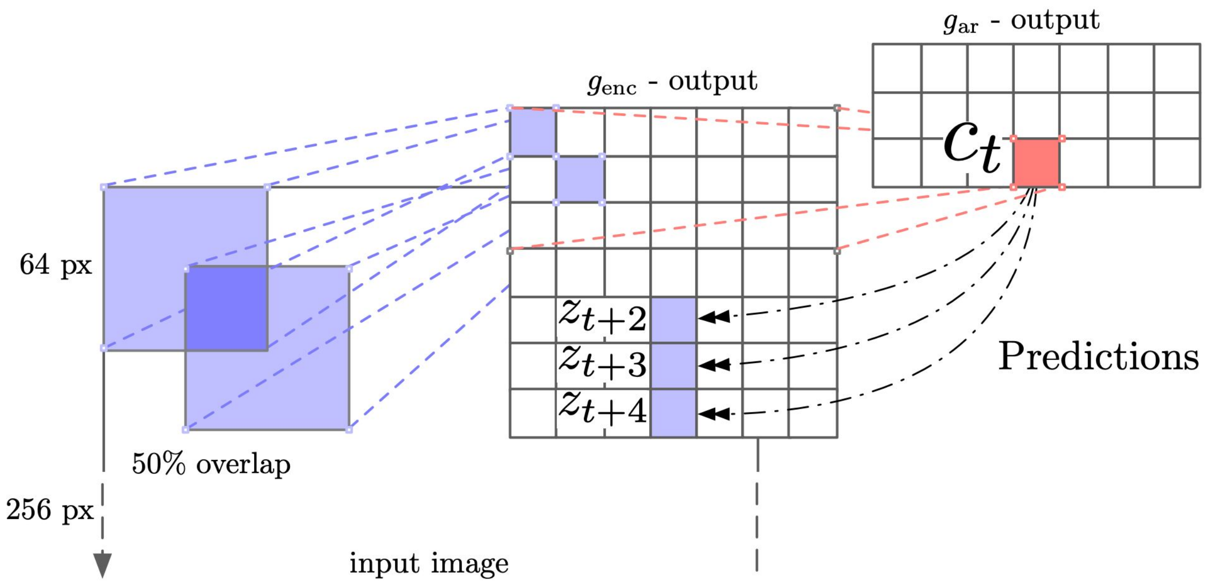
Method	ACC
Phone classification	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
Speaker classification	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

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.



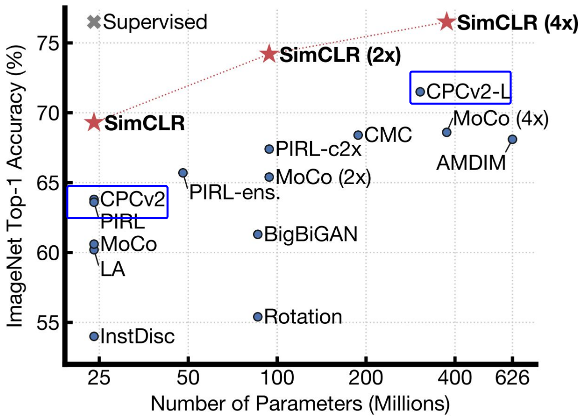
Source: van den Oord et al., 2018,

CPC example: modeling visual context

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

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:

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

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

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

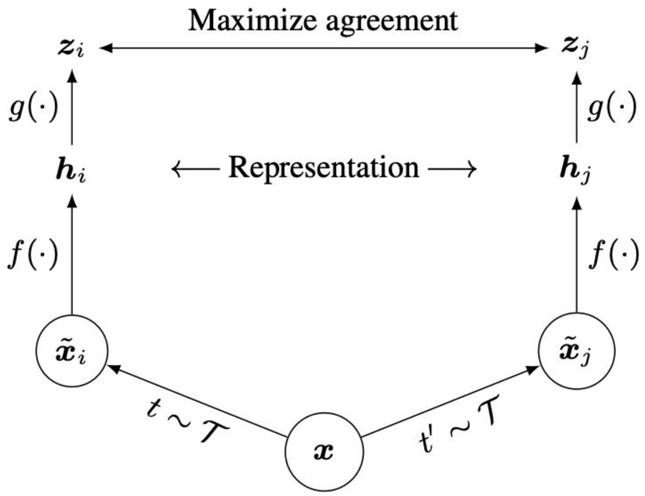
Commonly known as the InfoNCE loss (van den Oord et al., 2018)
A *lower bound* on the mutual information between *f(x)* and *f(x⁺)*

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

Summary: Contrastive Representation Learning

SimCLR: a simple framework for contrastive representation learning

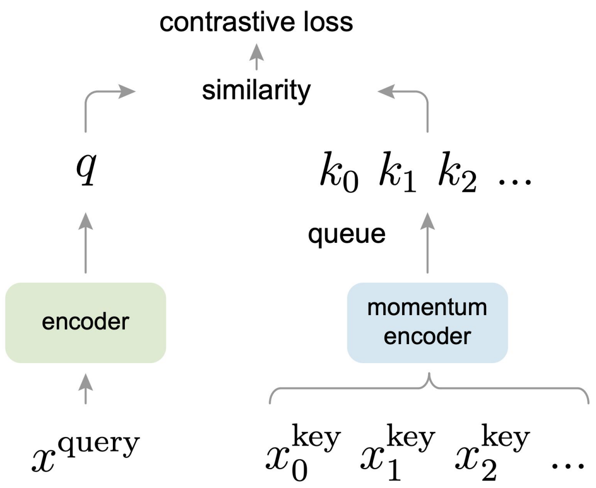
- **Key ideas**: non-linear projection head to allow flexible representation learning
- Simple to implement, effective in learning visual representation
- Requires large training batch size to be effective; large memory footprint



Summary: Contrastive Representation Learning

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

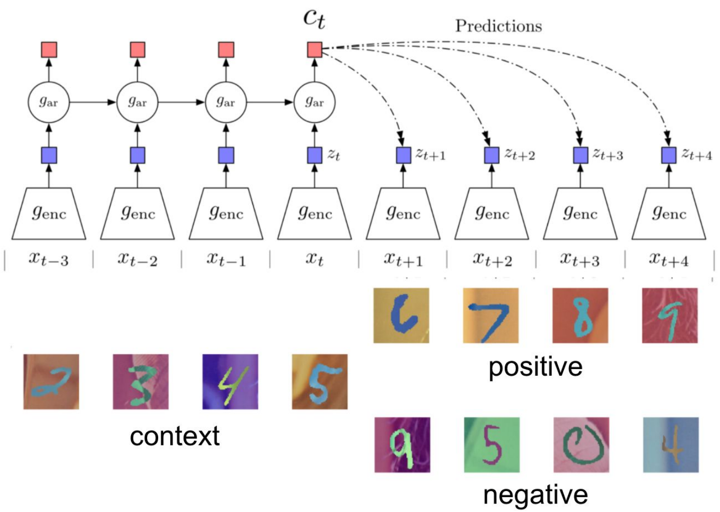
- Decouples negative sample size from minibatch size; allows large batch training without TPU
- MoCo-v2 combines the key ideas from SimCLR, i.e., nonlinear projection head, strong data augmentation, with momentum contrastive learning



Summary: Contrastive Representation Learning

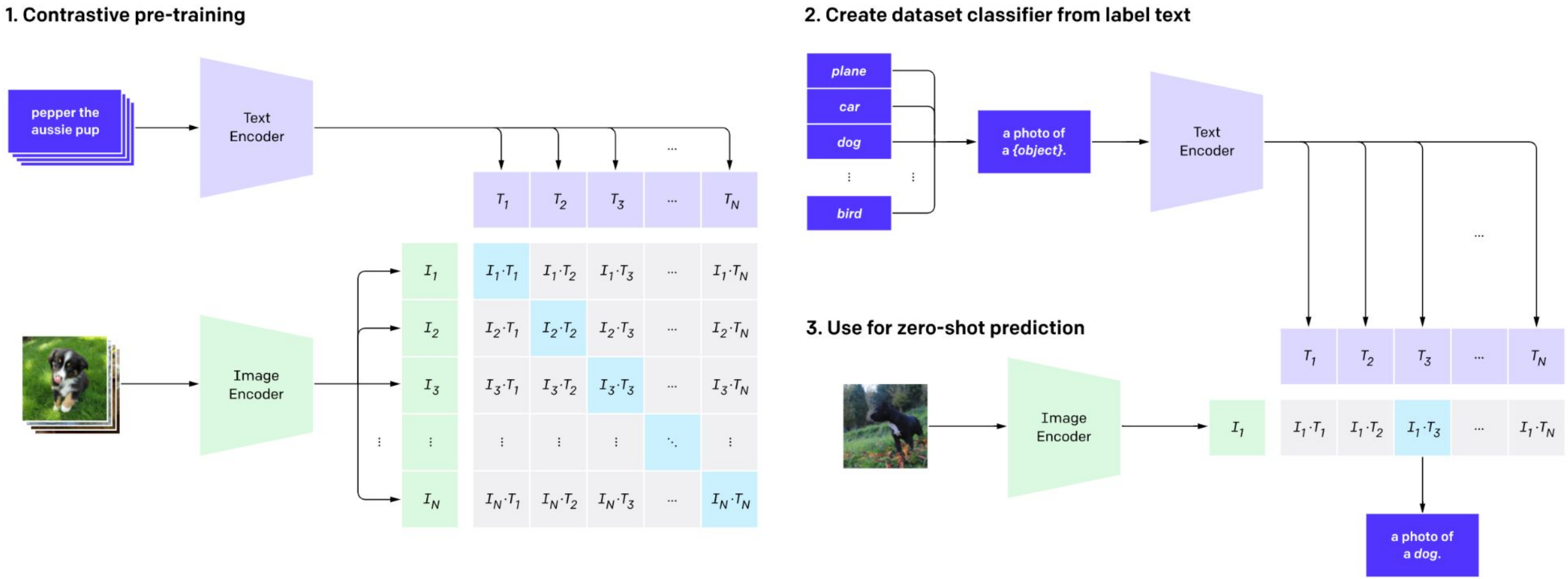
CPC: sequence-level contrastive learning

- 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

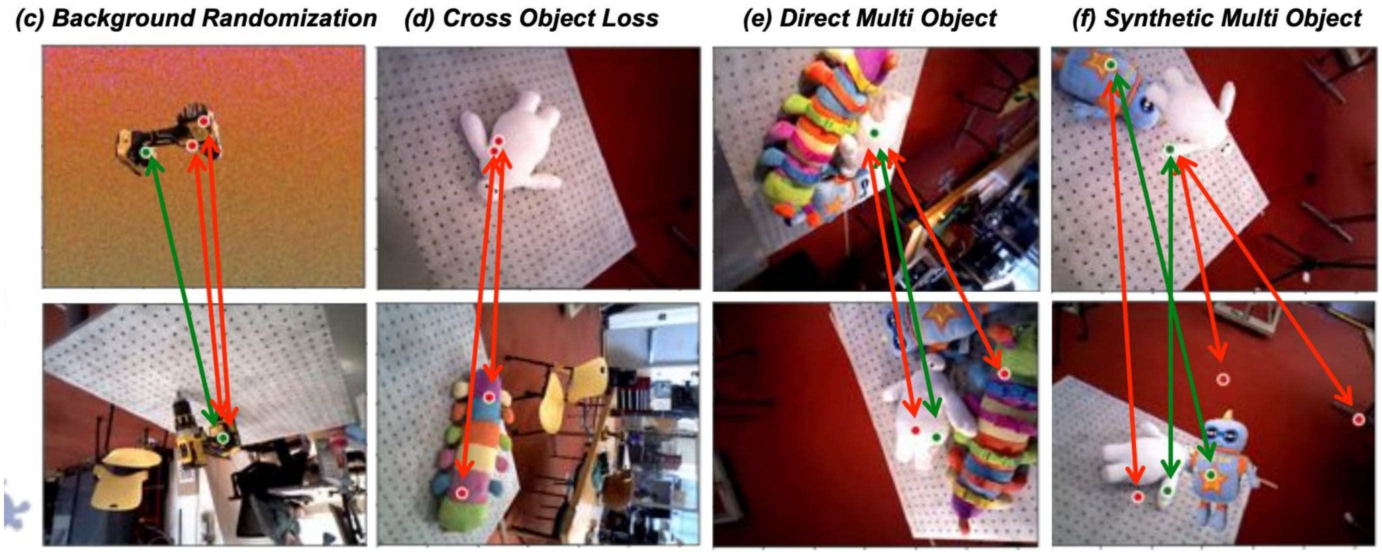
Contrastive learning between image and natural language sentences



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

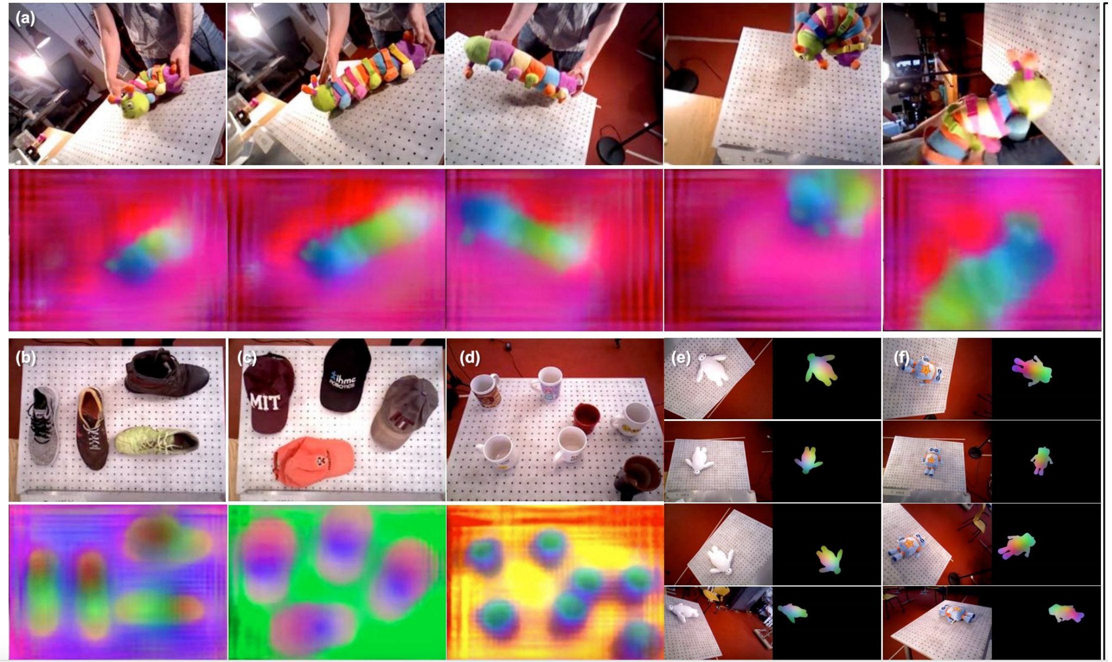
Other examples

Contrastive learning on pixel-wise feature descriptors



Dense Object Net, Florence et al., 2018

Other examples



Dense Object Net, Florence et al., 2018

Other examples

Next time: **Vision and Language + RNNs**

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

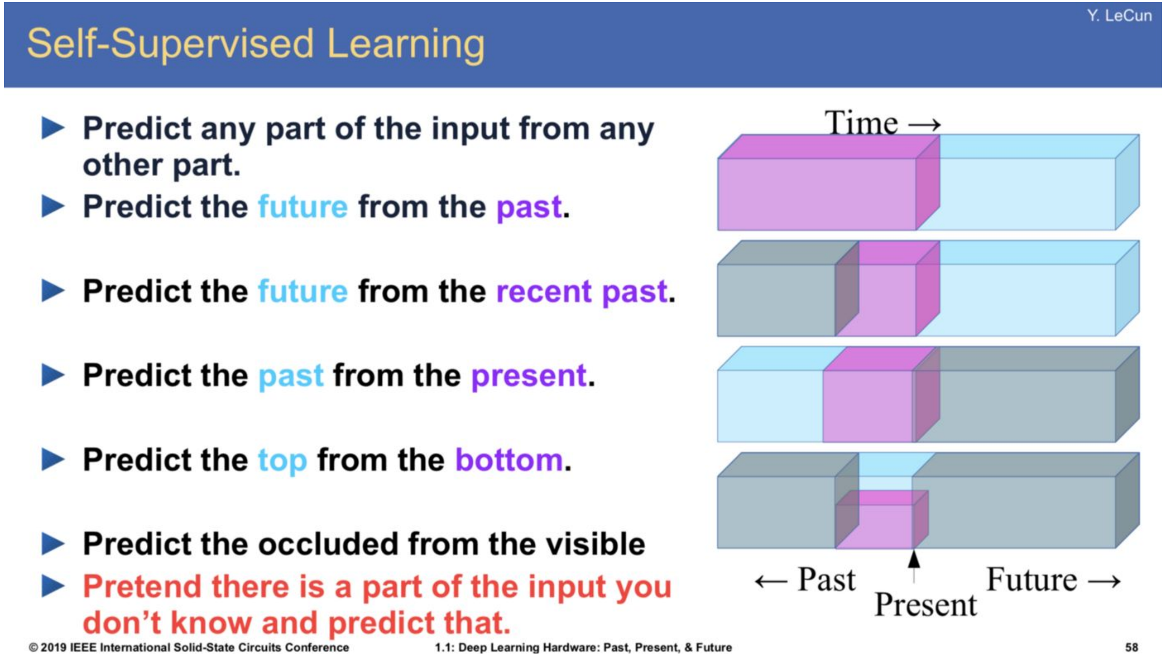
Frontier:

- Contrastive Language Image Pre-training (CLIP)

Frontier: Contrastive Language–Image Pre-training (CLIP)

Self-Supervised Learning

General idea: pretend there is a part of the data you don't know and train the neural network to predict that.



Source: Lecun 2019 Keynote at ISSCC

“The Cake of Learning”

downstream tasks

feature extractor

Learn good features through self-supervision

Y. LeCun

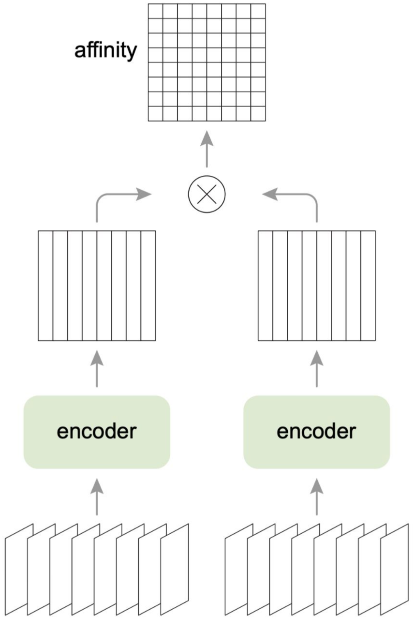
How Much Information is the Machine Given during Learning?

- **“Pure” Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

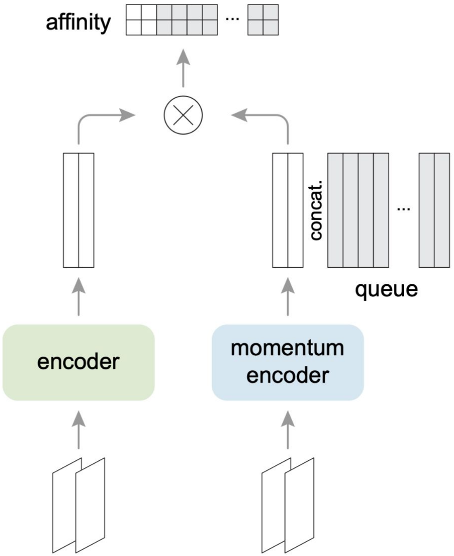
© 2019 IEEE International Solid-State Circuits Conference 1.1: Deep Learning Hardware: Past, Present, & Future 59

Source: Lecun 2019 Keynote at ISSCC

Can we do better?



SimCLR



Momentum Contrast (MoCo)

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