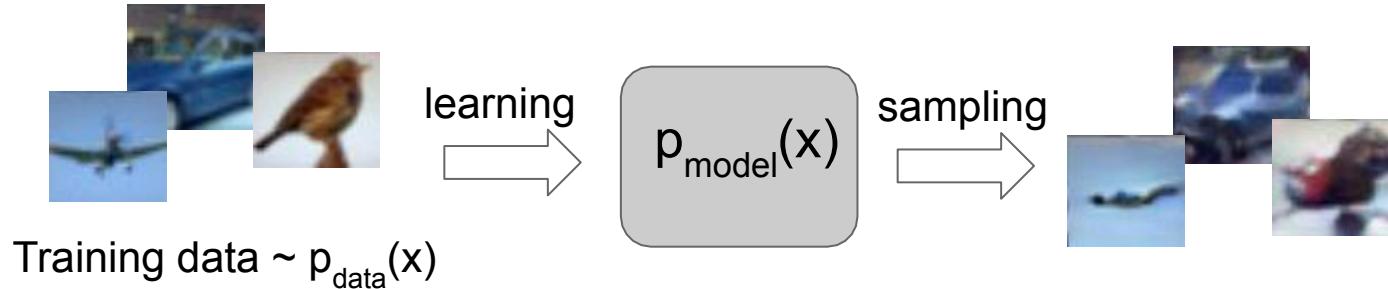

Self-Supervised Learning

Sanket H Kalwar

Generative Modeling

Given training data, generate new samples from same distribution



Objectives:

1. Learn $p_{\text{model}}(x)$ that approximates $p_{\text{data}}(x)$
2. Sampling new x from $p_{\text{model}}(x)$

Generative vs. Self-supervised Learning

- Both aim to learn from data without manual label annotation.
- Generative learning aims to model **data distribution** $p_{data}(x)$, e.g., generating realistic images.
- Self-supervised learning methods solve “pretext” tasks that produce **good features** for downstream tasks.
 - Learn with supervised learning objectives, e.g., classification, regression.
 - Labels of these pretext tasks are generated *automatically*

Self-supervised pretext

tasks
Example: learn to predict image transformations / complete corrupted images

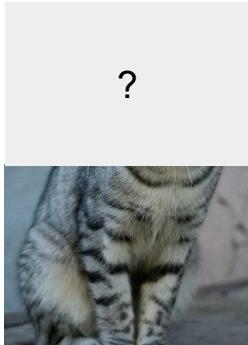
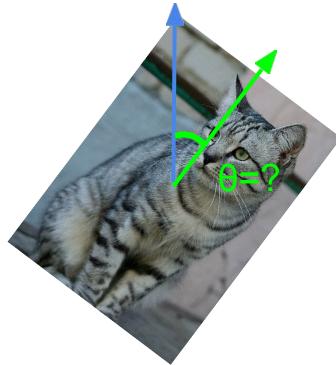


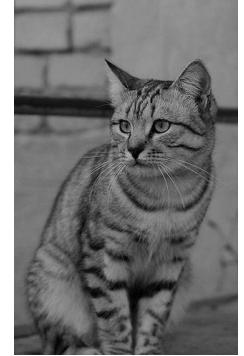
image completion



rotation prediction



“jigsaw puzzle”



colorization

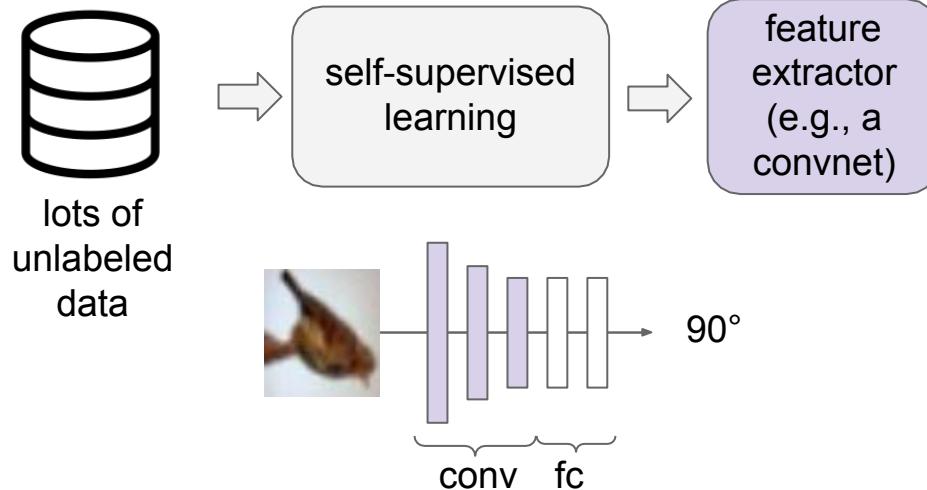
1. Solving the pretext tasks allow the model to learn good features.
2. We can automatically generate labels for the pretext tasks.

How to evaluate a self-supervised learning method?

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

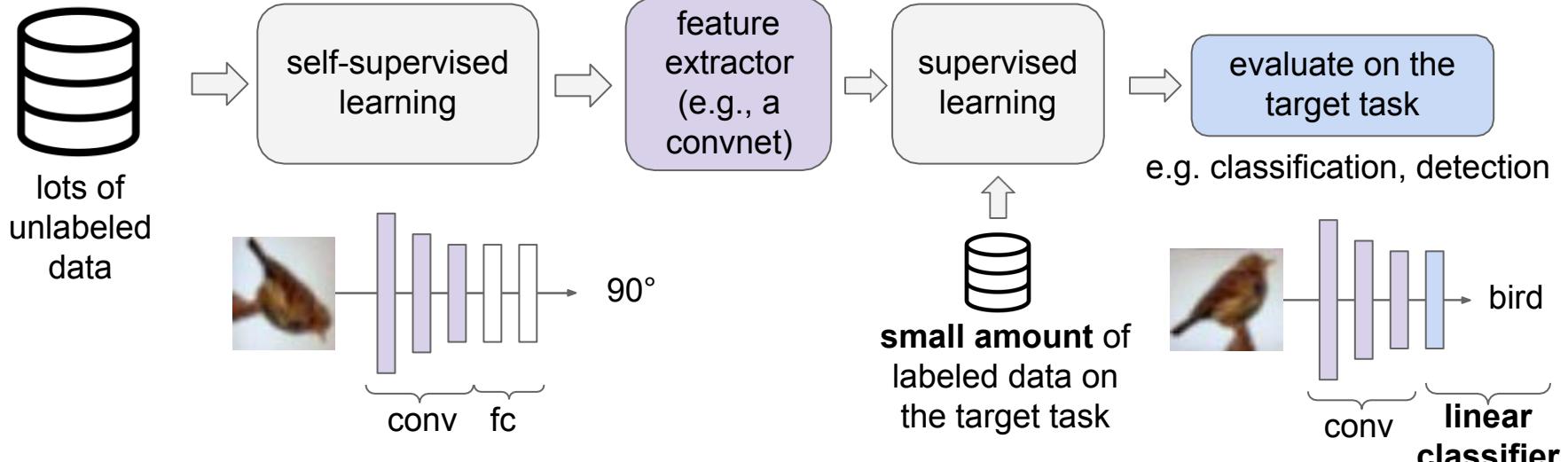
Evaluate the learned feature encoders on downstream *target tasks*

How to evaluate a self-supervised learning method?



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

How to evaluate a self-supervised learning method?



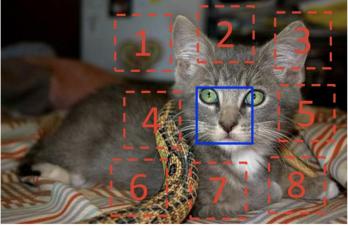
1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

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

Broader picture

Today's lecture

computer vision



Doersch et al., 2015

robot / reinforcement learning



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

language modeling

Language Models are Few-Shot Learners

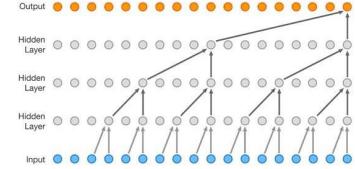
Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse 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. In this paper, we show that scaling up a sequence model with cross-entropy loss to a single example per class 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 regime. Finally, GPT-3 is applied without task-specific pre-training 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, it exhibits some limitations when used as a few-shot learner, such as in well-known 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)

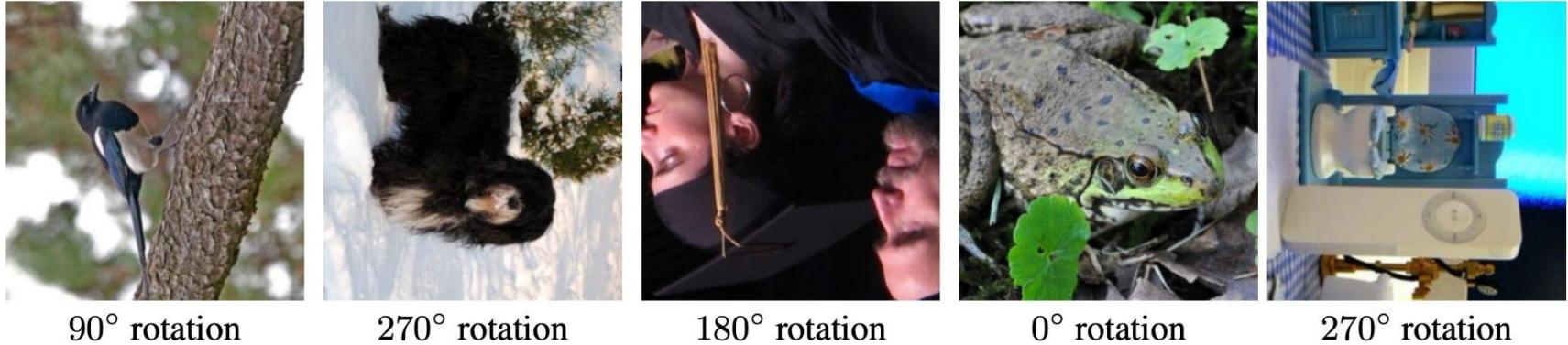
speech synthesis



Wavenet (van den Oord et al., 2016)



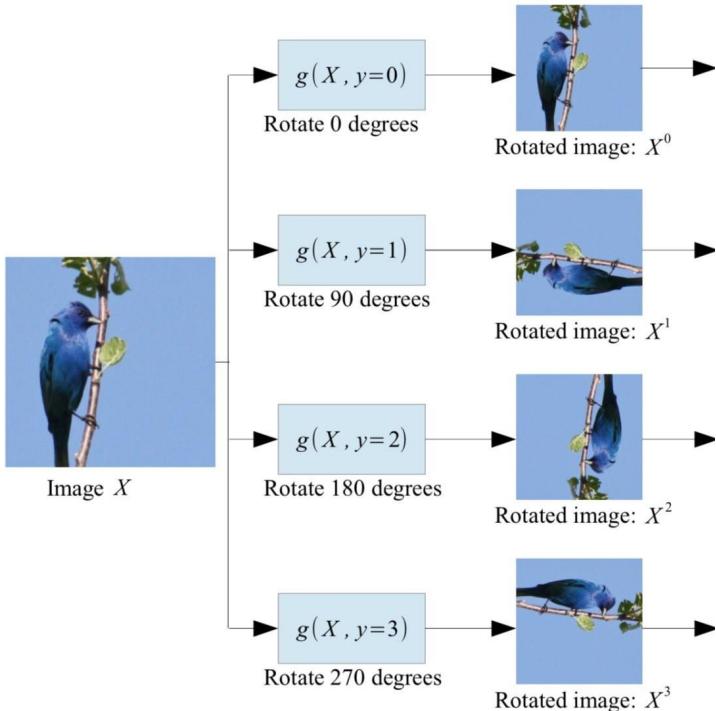
Pretext task: predict rotations



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

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

Pretext task: predict rotations

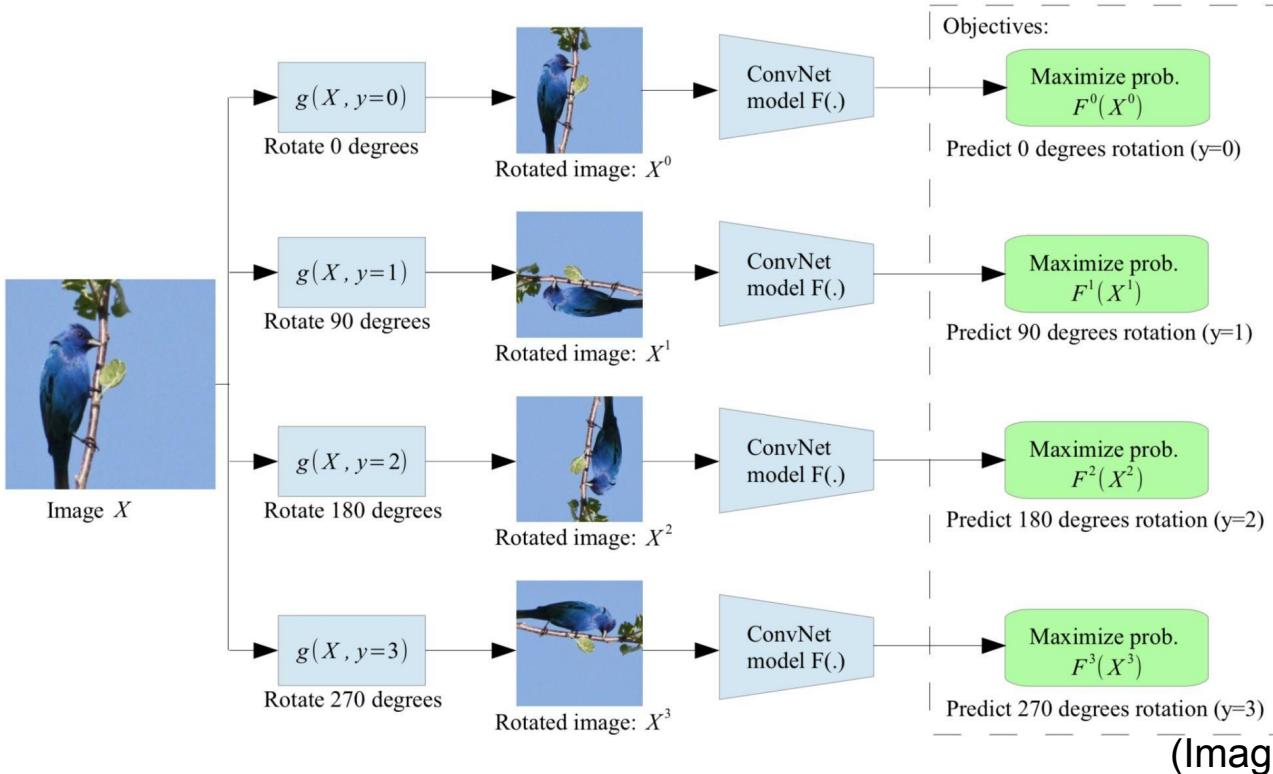


Self-supervised learning by rotating the entire input images.

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

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

Pretext task: predict rotations

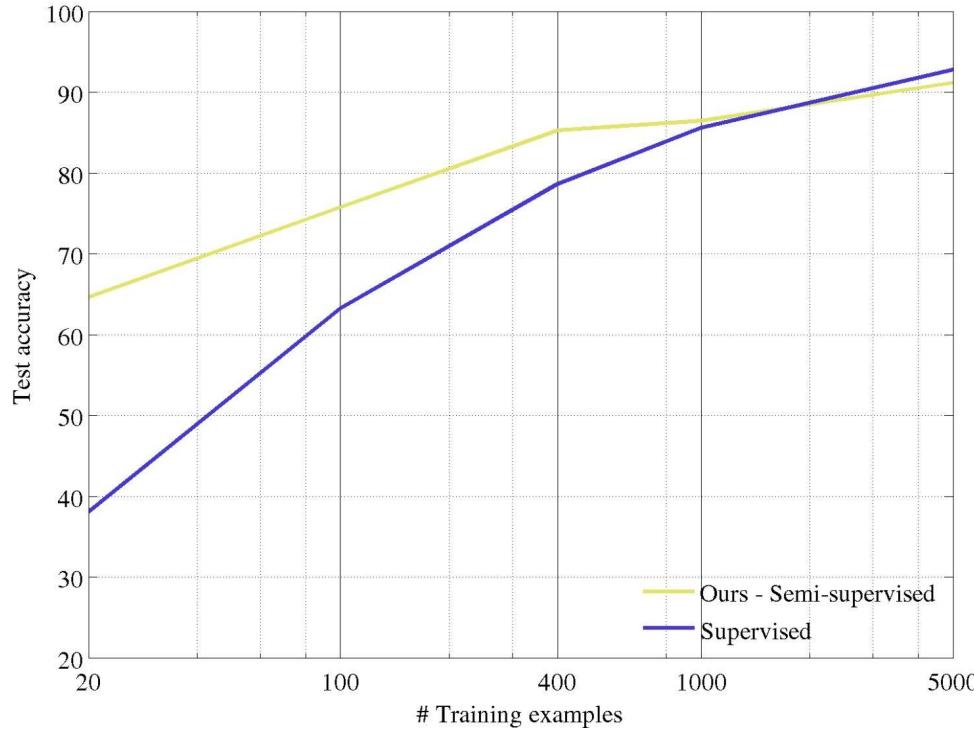


Self-supervised learning by rotating the entire input images.

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

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

Evaluation on semi-supervised learning



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

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

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

Transfer learned features to supervised learning

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

Self-supervised learning with rotation prediction

Pretrained with full
ImageNet supervision

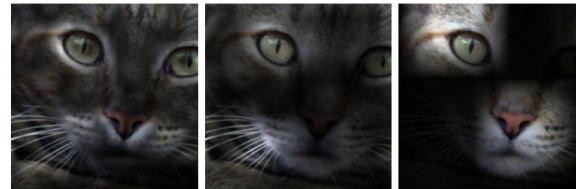
No pretraining

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

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

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

Visualize learned visual attentions



Conv1 27×27 Conv3 13×13 Conv5 6×6

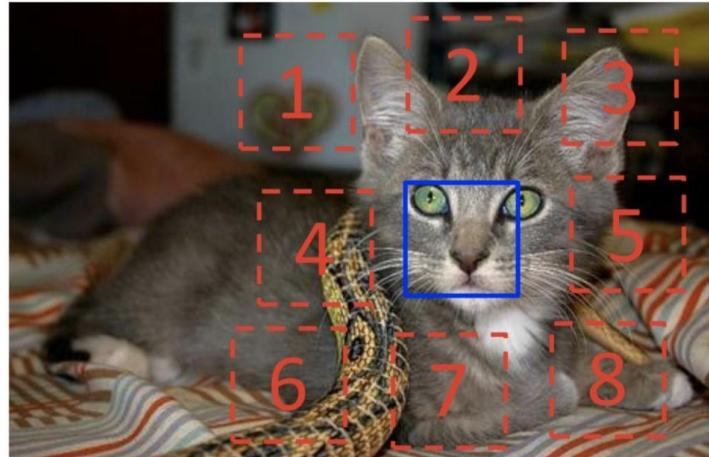
Conv1 27×27 Conv3 13×13 Conv5 6×6

(a) Attention maps of supervised model

(b) Attention maps of our self-supervised model

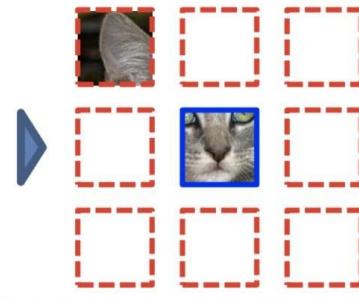
(Image source: [Gidaris et al. 2018](#))

Pretext task: predict relative patch locations



$$X = (\underset{\downarrow}{\text{[cat eye, snake tail]}}, \text{?}); Y = 3$$

Example:



Question 1:

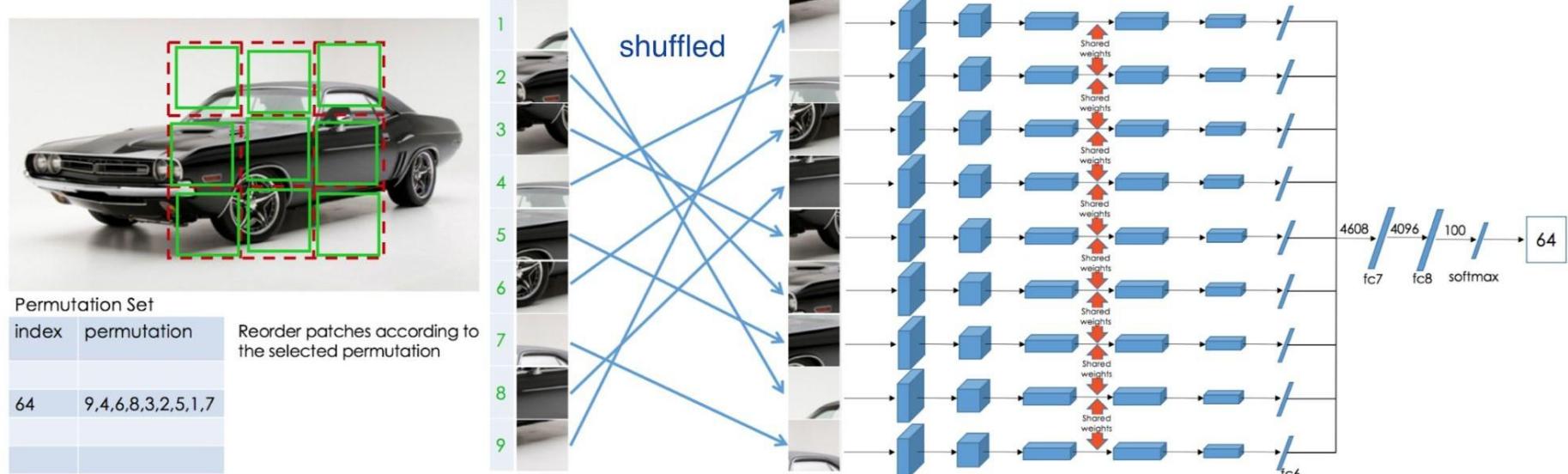


Question 2:



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

Pretext task: solving “jigsaw puzzles”



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

Transfer learned features to supervised learning

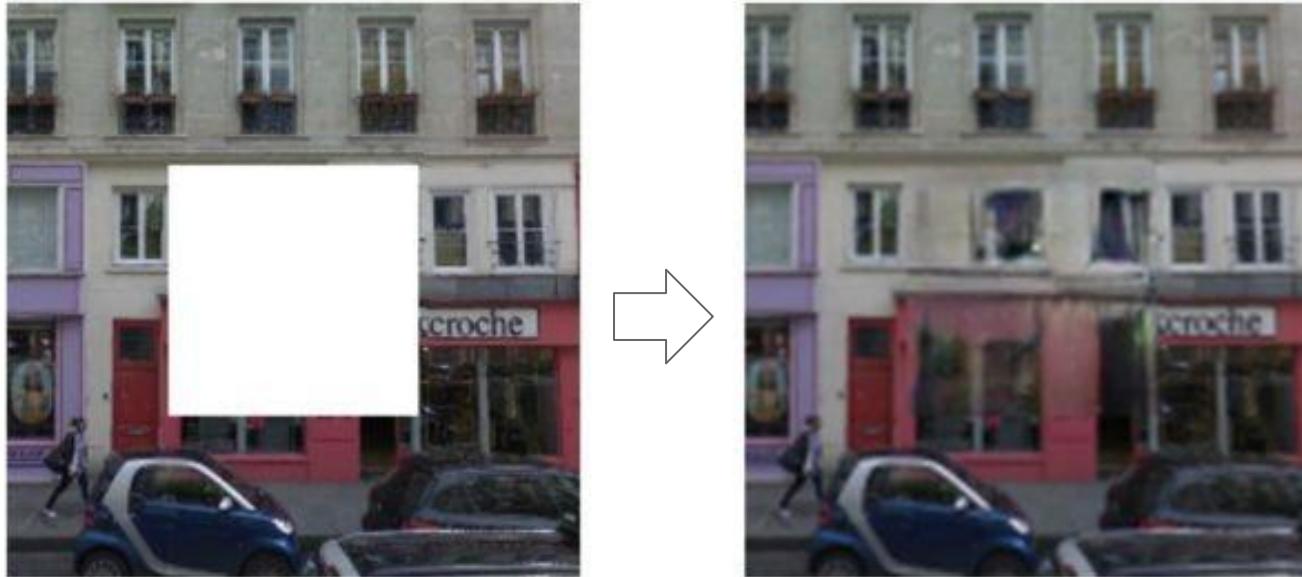
Table 1: Results on PASCAL VOC 2007 Detection and Classification. The results of the other methods are taken from Pathak *et al.* [30].

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

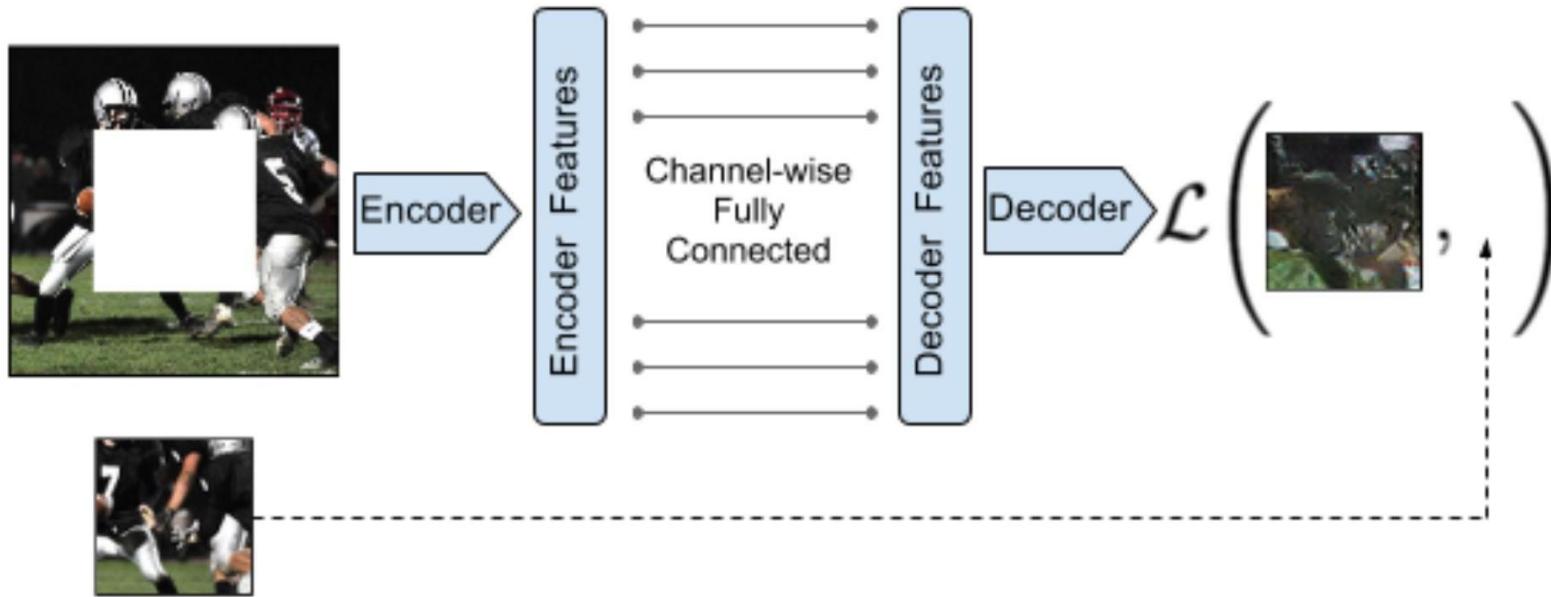
Pretext task: predict missing pixels (inpainting)



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

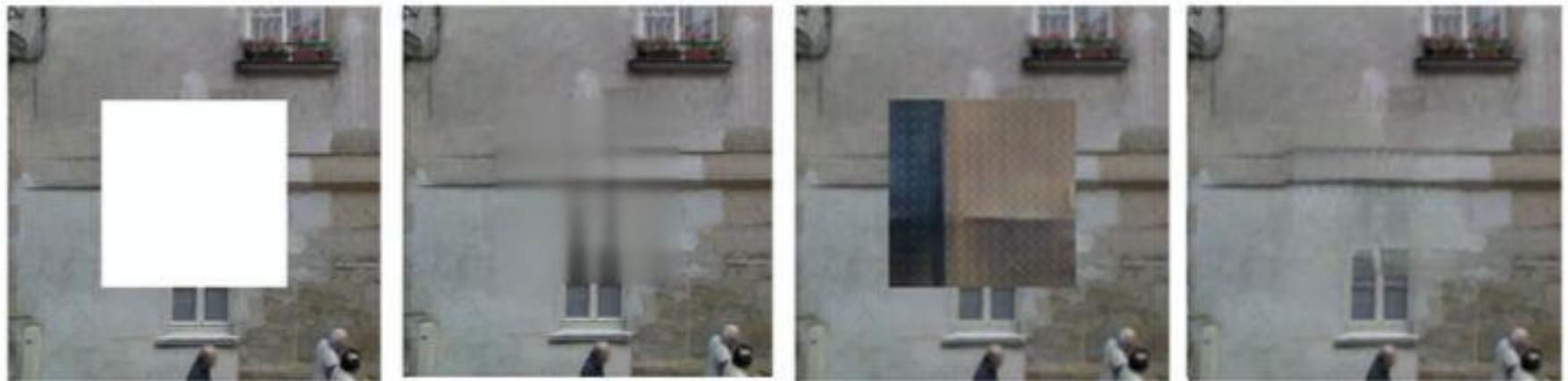
$$L(x) = L_{recon}(x) + L_{adv}(x)$$

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

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

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

Inpainting evaluation



Input (context)

reconstruction

adversarial

recon + adv

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

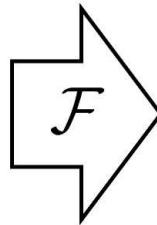
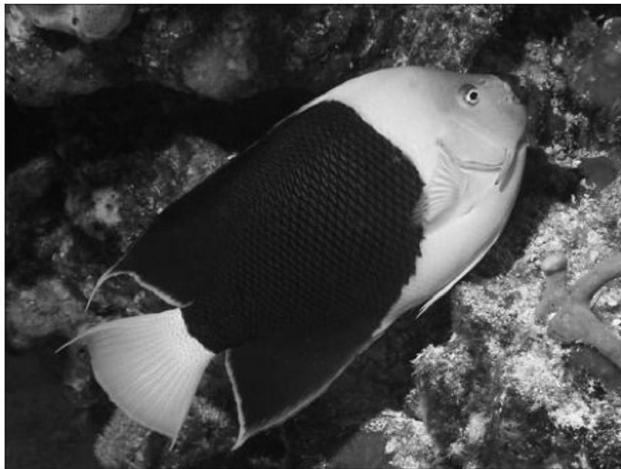
Transfer learned features to supervised learning

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

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

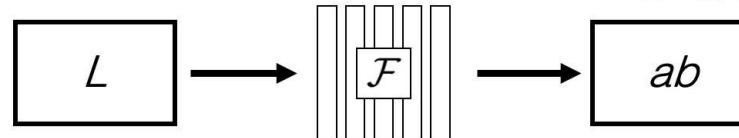
Source: [Pathak et al., 2016](#)

Pretext task: image coloring



Grayscale image: L channel

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

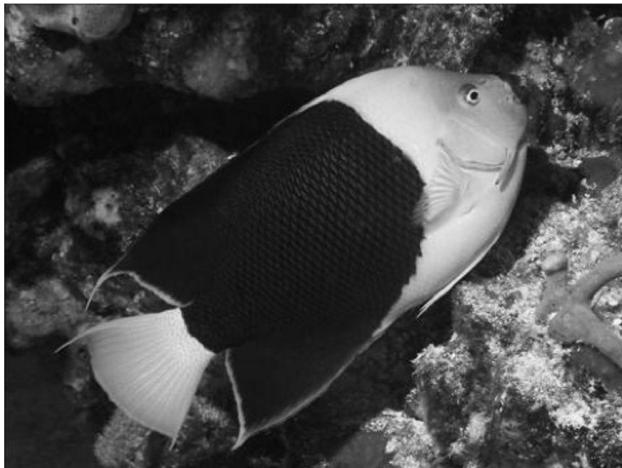


Color information: ab channels

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

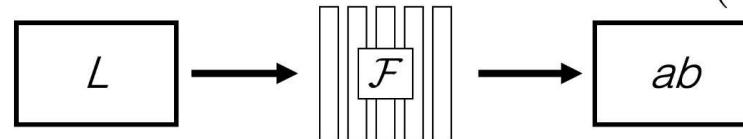
Source: Richard Zhang / Phillip Isola ⁵

Pretext task: image coloring



Grayscale image: L channel

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

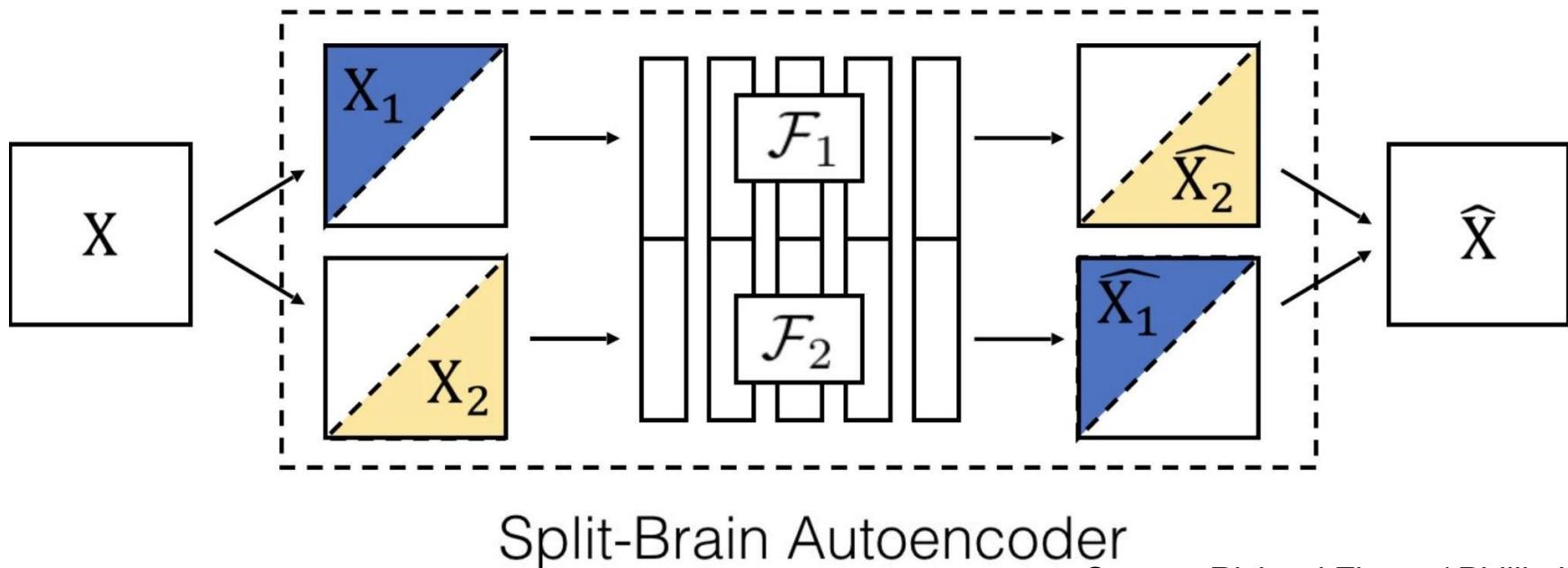


Concatenate (L, ab) channels
 $(\mathbf{X}, \hat{\mathbf{Y}})$

Source: Richard Zhang / Phillip Isola

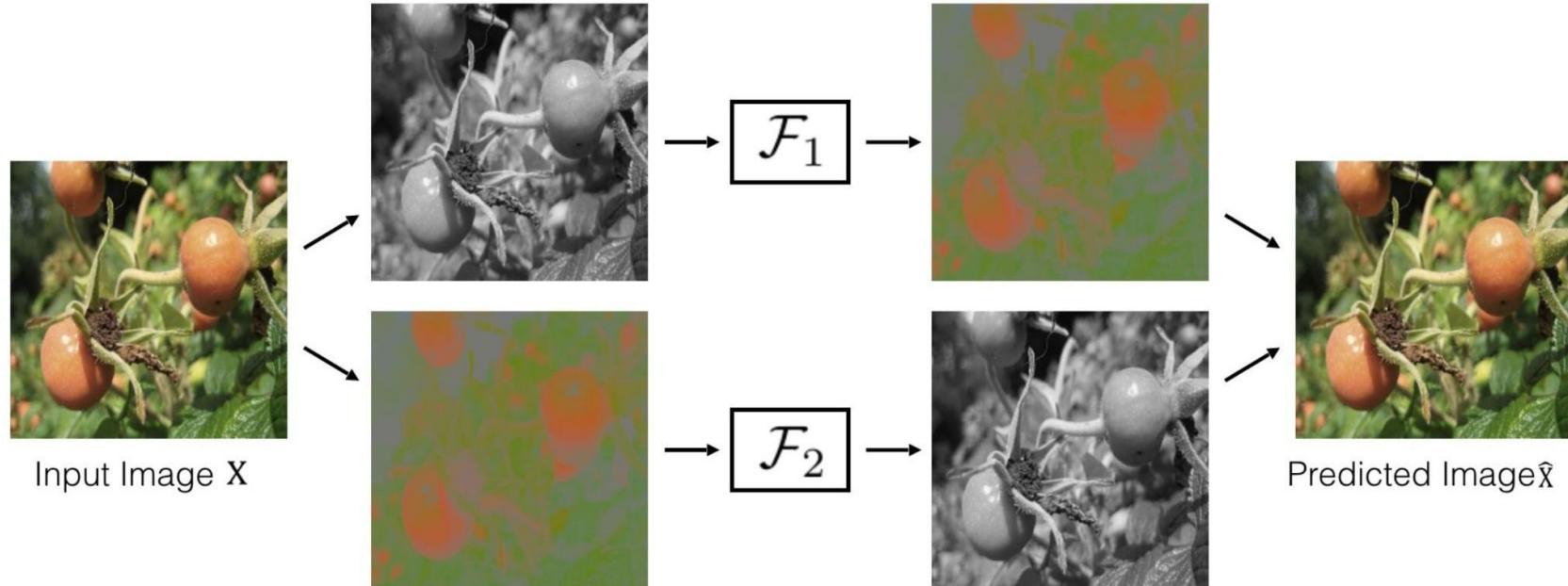
Learning features from colorization: Split-brain Autoencoder

Idea: cross-channel predictions



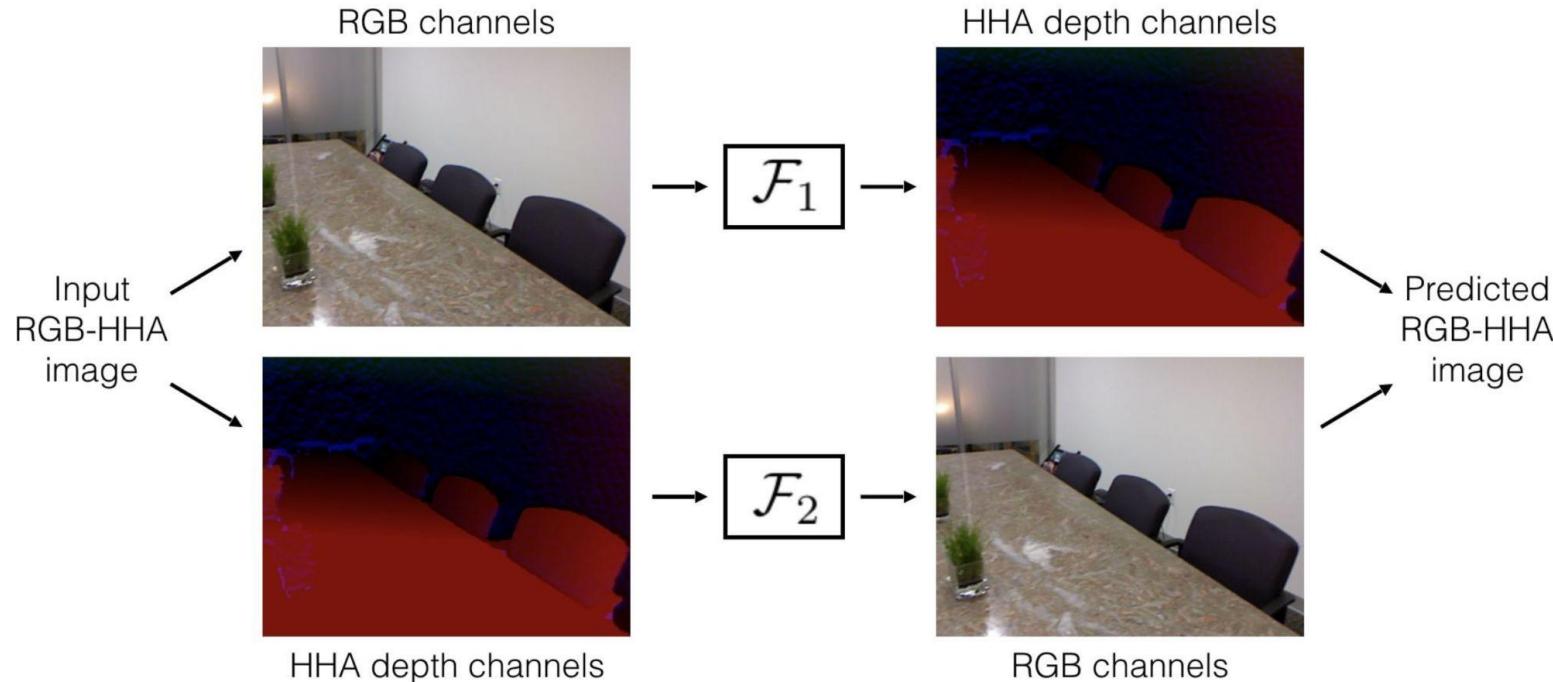
Source: Richard Zhang / Phillip Isola

Learning features from colorization: Split-brain Autoencoder



Source: Richard Zhang / Phillip Isola

Learning features from colorization: Split-brain Autoencoder



Source: Richard Zhang / Phillip Isola

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

reference frame



$t = 0$

how should I color these frames?

Should be the same color!

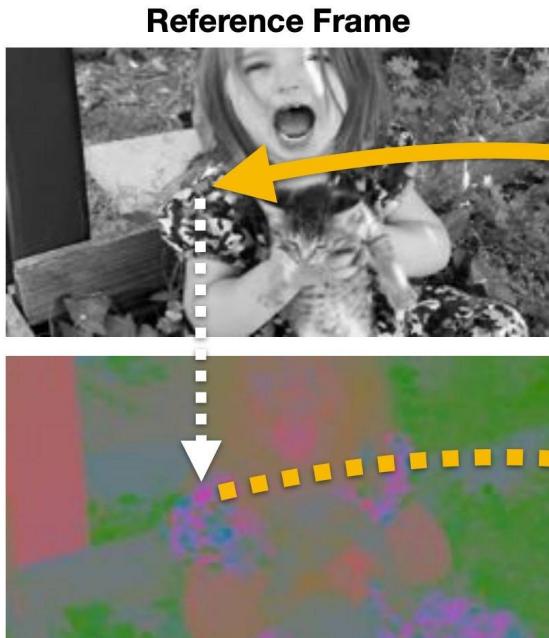


$t = 1 \ t = 2 \ t = 3$

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

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

Learning to color videos



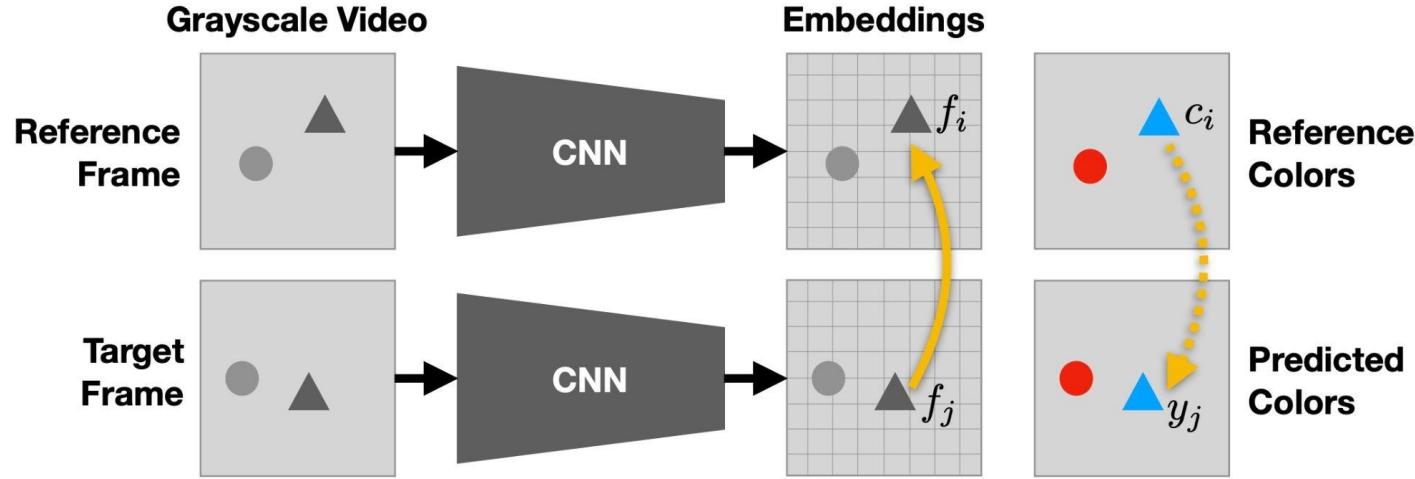
Learning objective:

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

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

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

Learning to color videos

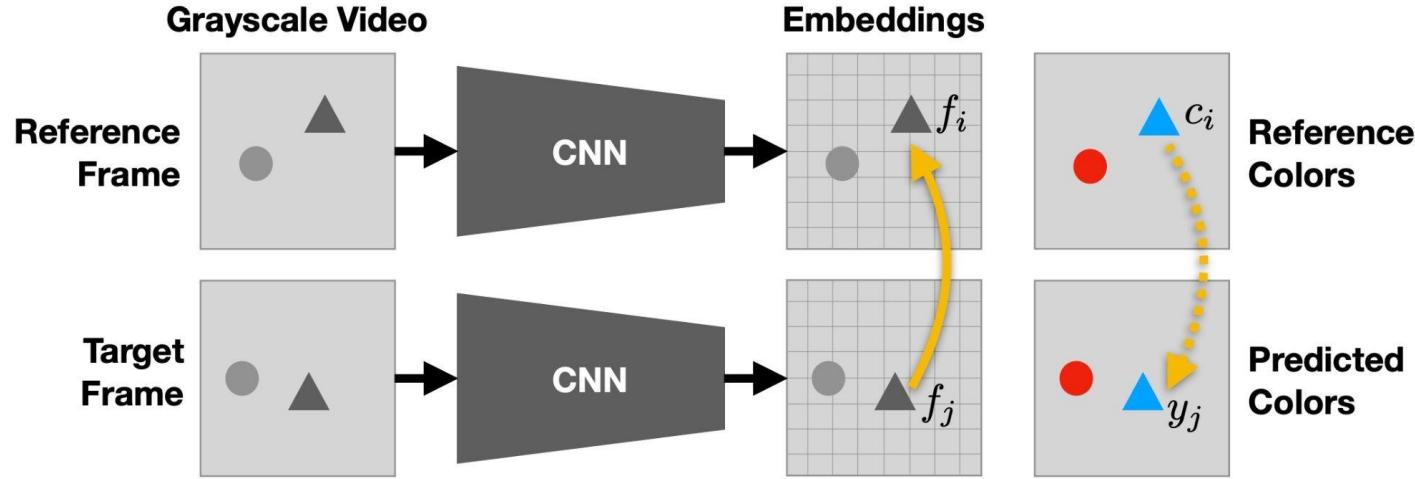


attention map on the
reference frame

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

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

Learning to color videos



attention map on the
reference frame

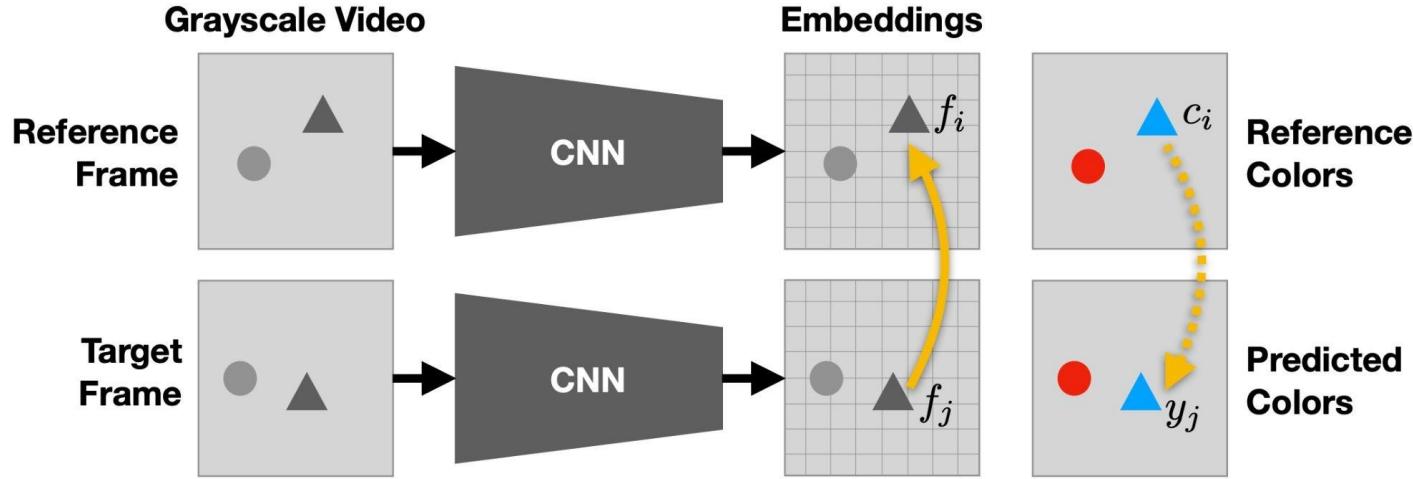
predicted color = weighted
sum of the reference color

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

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

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

Learning to color videos



attention map on the
reference frame

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

predicted color = weighted
sum of the reference color

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

loss between predicted color
and ground truth color

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

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

Colorizing videos (qualitative)

reference frame target frames (gray)



predicted color



Source: [Google AI blog post](#)

Colorizing videos (qualitative)

reference frame target frames (gray)



predicted color



Source: [Google AI blog post](#)

Tracking emerges from colorization

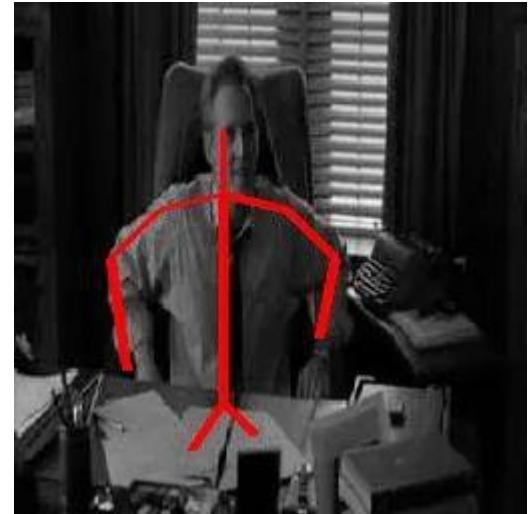
Propagate segmentation masks using learned attention



Source: [Google AI blog post](#)

Tracking emerges from colorization

Propagate pose keypoints using learned attention



Source: [Google AI blog post](#)

Summary: pretext tasks from image transformations

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

Summary: pretext tasks from image transformations

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

Pretext tasks from image transformations

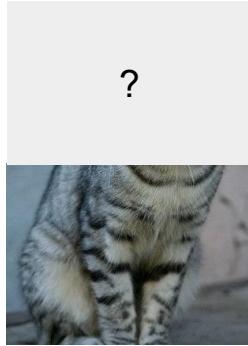
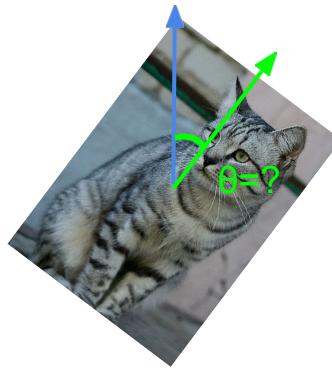


image completion



rotation prediction



“jigsaw puzzle”

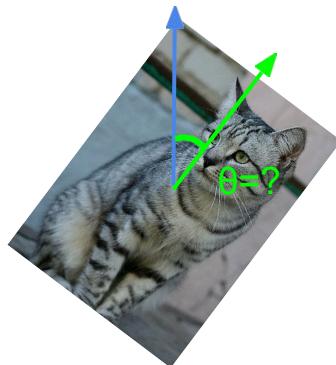


colorization

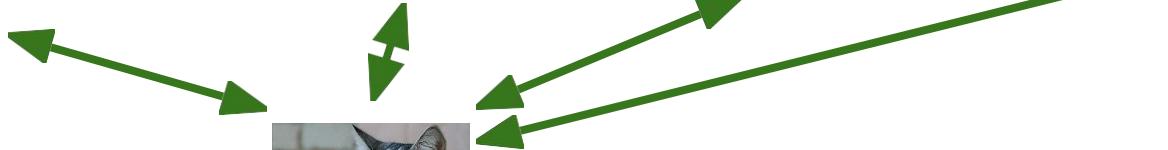
Learned representations may be tied to a specific pretext task!

Can we come up with a more general pretext task?

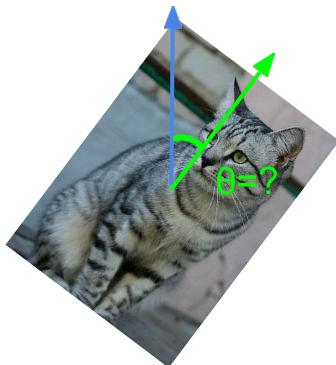
A more general pretext task?



same object



A more general pretext task?



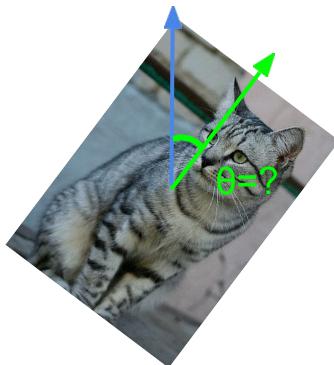
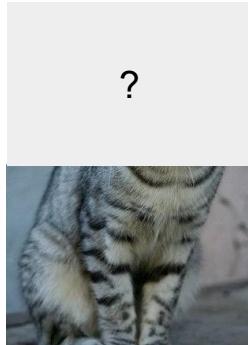
same object



different object



Contrastive Representation Learning



attract

repel



Today's Agenda

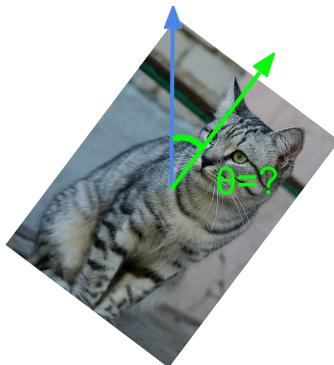
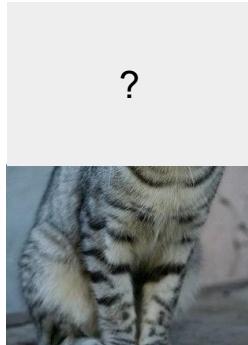
Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning

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

Contrastive Representation Learning

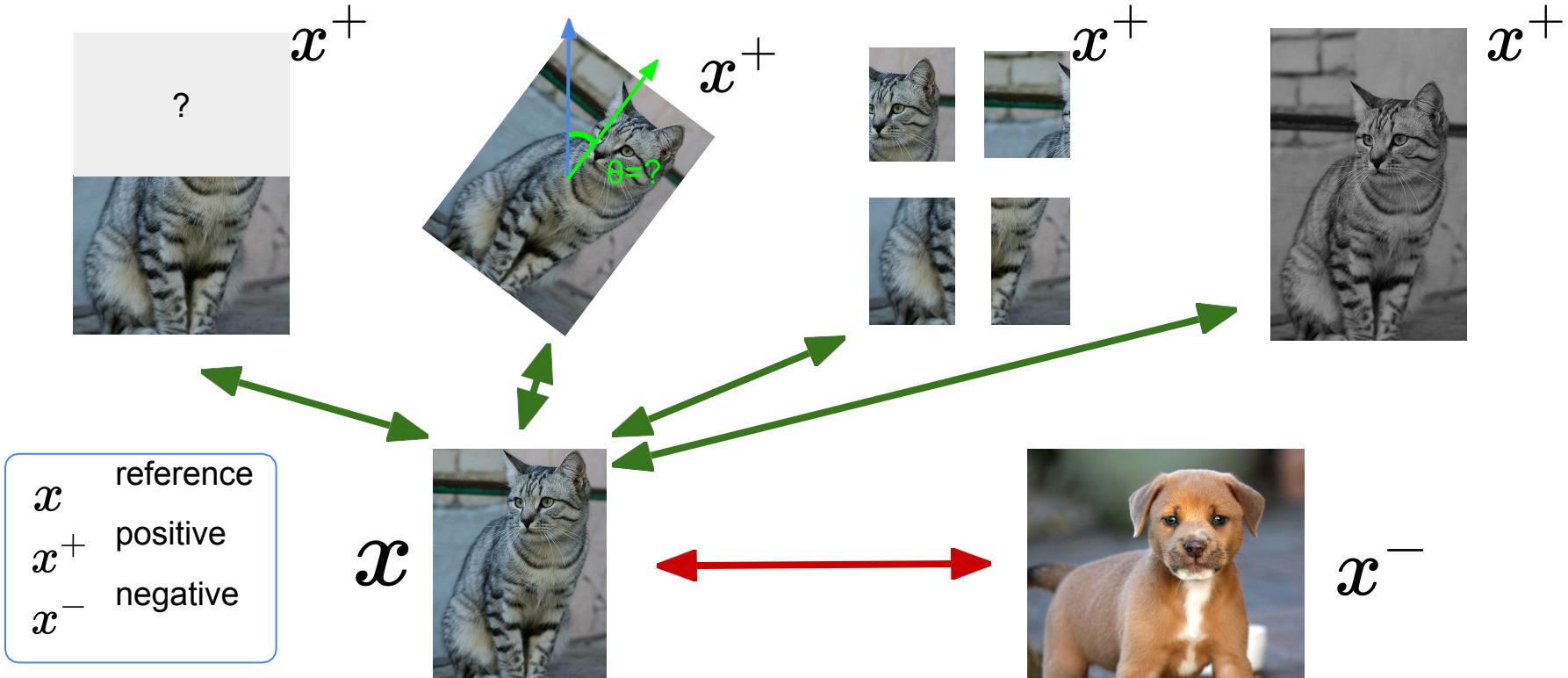


attract

repel



Contrastive Representation Learning



A formulation of contrastive learning

What we want:

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

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

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

A formulation of contrastive learning

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

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

A formulation of contrastive learning

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

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

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

A formulation of contrastive learning

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

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

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:

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

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

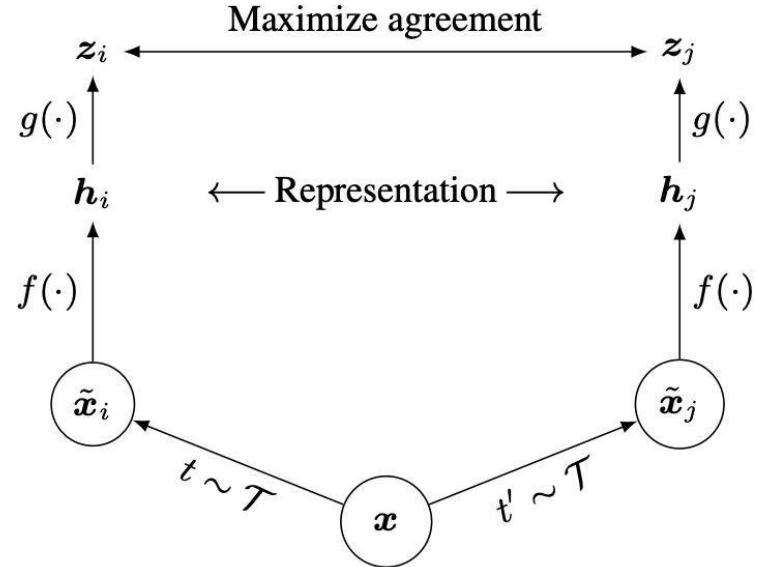
Cosine similarity as the score function:

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

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

Generate positive samples through data augmentation:

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



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

SimCLR: generating positive samples from data augmentation



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



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



(g) Cutout



(h) Gaussian noise



(i) Gaussian blur



(j) Sobel filtering

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

SimCLR

Generate a positive pair
by sampling data
augmentation functions

Algorithm 1 SimCLR's main learning algorithm.

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do
    for all  $k \in \{1, \dots, N\}$  do
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$ 
        # the first augmentation
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection
        # the second augmentation
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection
    end for
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity
    end for
    define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$ 
     $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$ 
    update networks  $f$  and  $g$  to minimize  $\mathcal{L}$ 
end for
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 
```

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

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

SimCLR

Generate a positive pair
by sampling data
augmentation functions

Algorithm 1 SimCLR's main learning algorithm.

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .  
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do  
    for all  $k \in \{1, \dots, N\}$  do  
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$   
        # the first augmentation  
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$   
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation  
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection  
        # the second augmentation  
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$   
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation  
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection  
    end for  
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do  
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity  
    end for  
    define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$   
     $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$   
    update networks  $f$  and  $g$  to minimize  $\mathcal{L}$   
end for  
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 
```

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

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

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

SimCLR

Generate a positive pair
by sampling data
augmentation functions

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

Algorithm 1 SimCLR's main learning algorithm.

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .  
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do  
    for all  $k \in \{1, \dots, N\}$  do  
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$   
        # the first augmentation  
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$   
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation  
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection  
        # the second augmentation  
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$   
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation  
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection  
    end for  
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do  
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity  
    end for  
    define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$   
    ▶  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$   
    update networks  $f$  and  $g$  to minimize  $\mathcal{L}$   
    end for  
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 
```

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

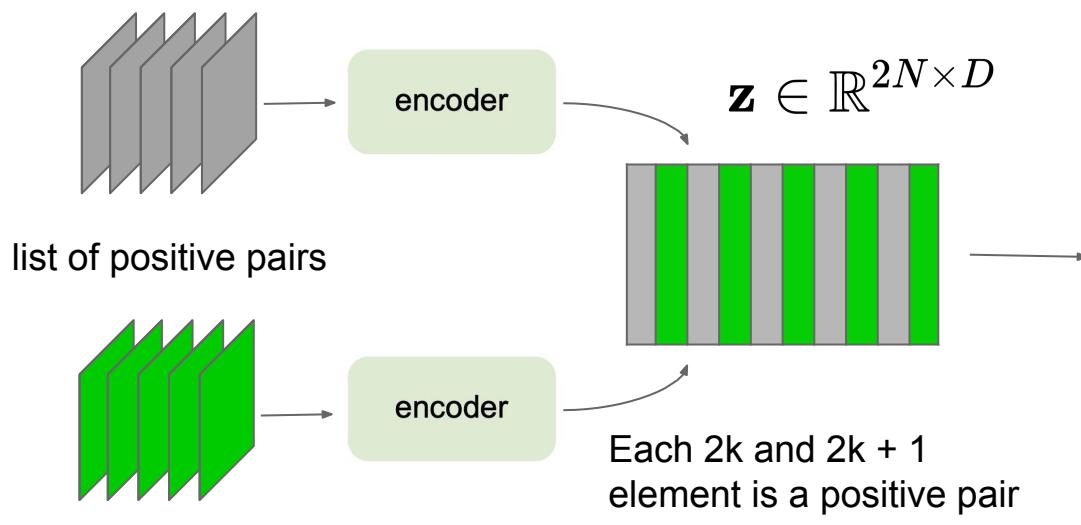
InfoNCE loss:
Use all non-positive samples in the batch as x^-

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

SimCLR: mini-batch training

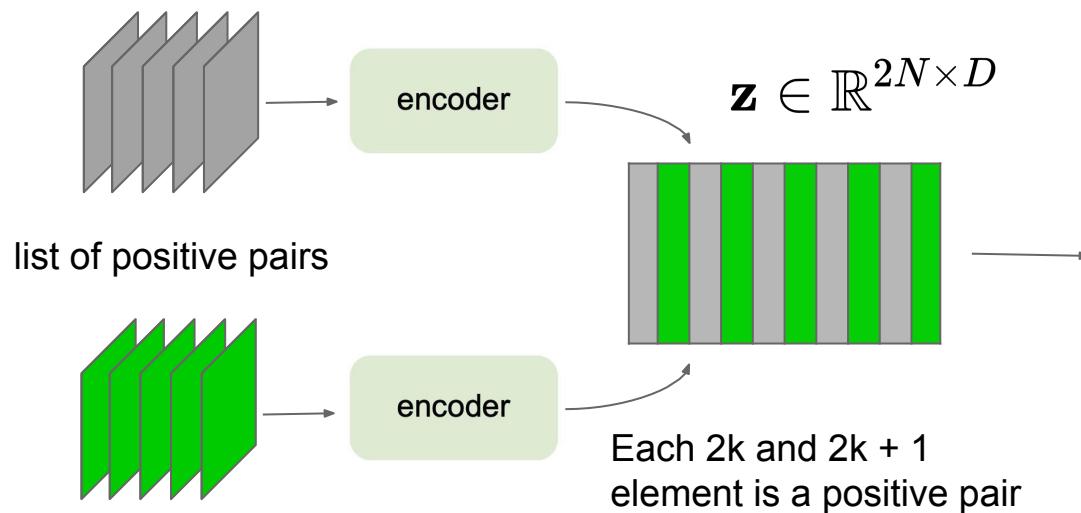
$$s_{i,j} = \frac{z_i^T z_j}{\|z_i\| \|z_j\|}$$

“Affinity matrix”



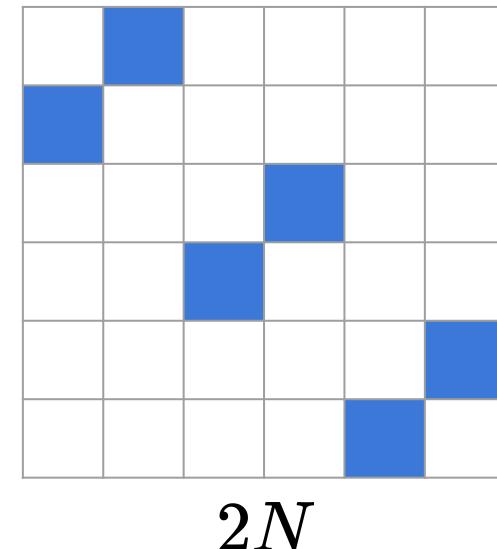
*We use a slightly different formulation in the assignment.
You should follow the assignment instructions.

SimCLR: mini-batch training



$$s_{i,j} = \frac{\mathbf{z}_i^T \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|}$$

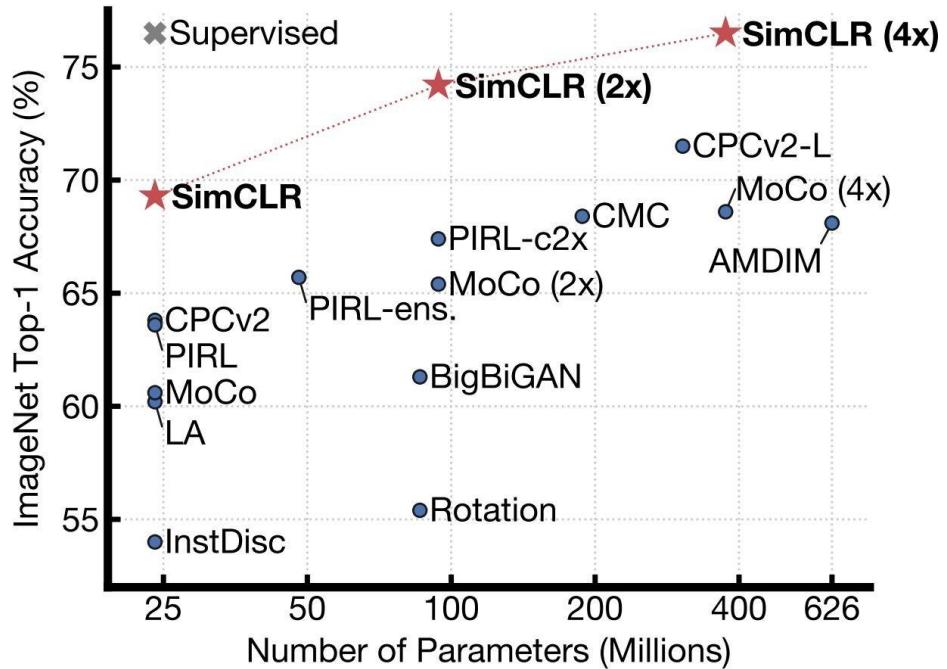
“Affinity matrix”



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

= classification label for each row

Training linear classifier on SimCLR features



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

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

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

Semi-supervised learning on SimCLR features

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

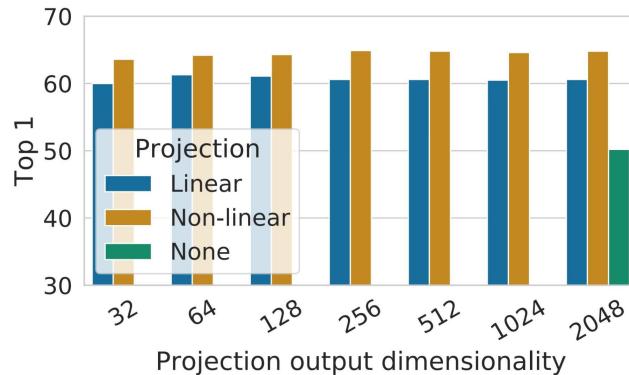
Table 7. ImageNet accuracy of models trained with few labels.

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](#)

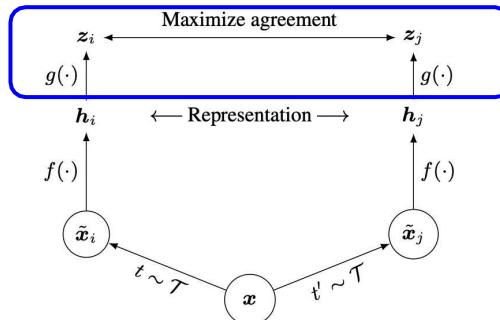
SimCLR design choices: projection head



Linear / non-linear projection heads improve representation learning.

A possible explanation:

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



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

SimCLR design choices: large batch size

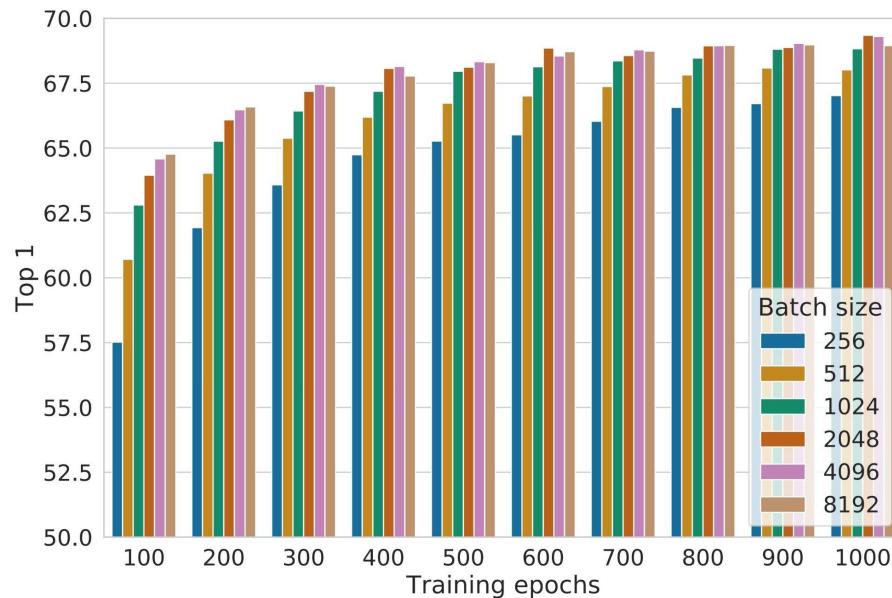


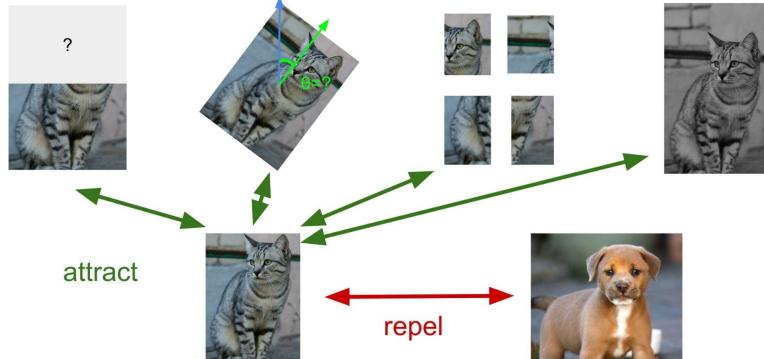
Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.¹⁰

Large training batch size is crucial for SimCLR!

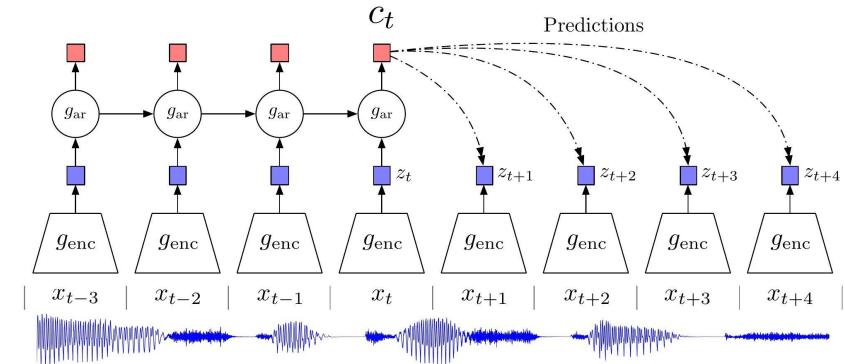
Large batch size causes large memory footprint during backpropagation:
requires distributed training on TPUs
(ImageNet experiments)

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

Instance vs. Sequence Contrastive Learning



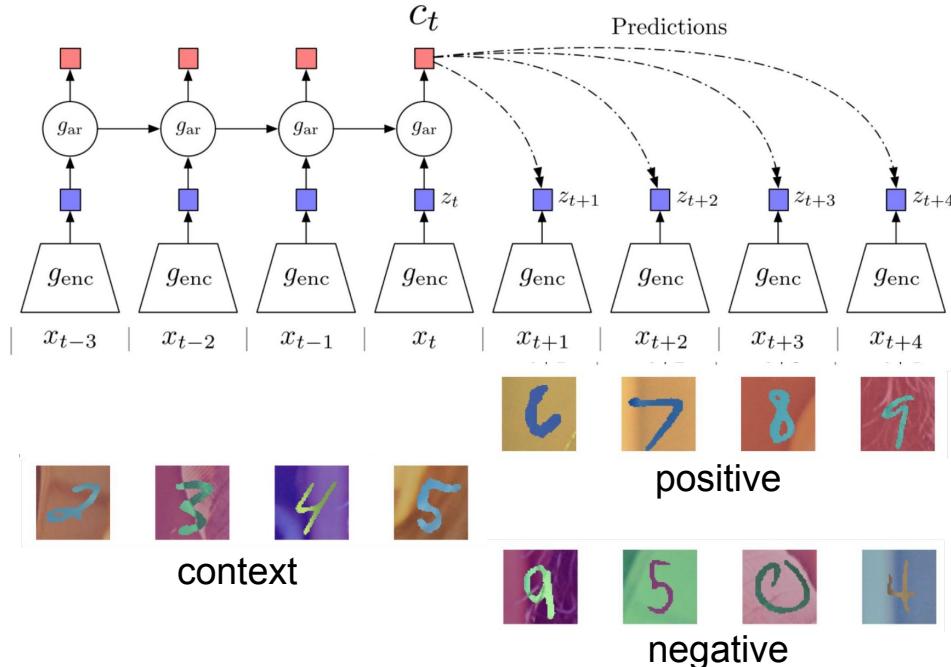
Instance-level contrastive learning:
contrastive learning based on
positive & negative instances.
Examples: SimCLR



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

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

Contrastive Predictive Coding (CPC)



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

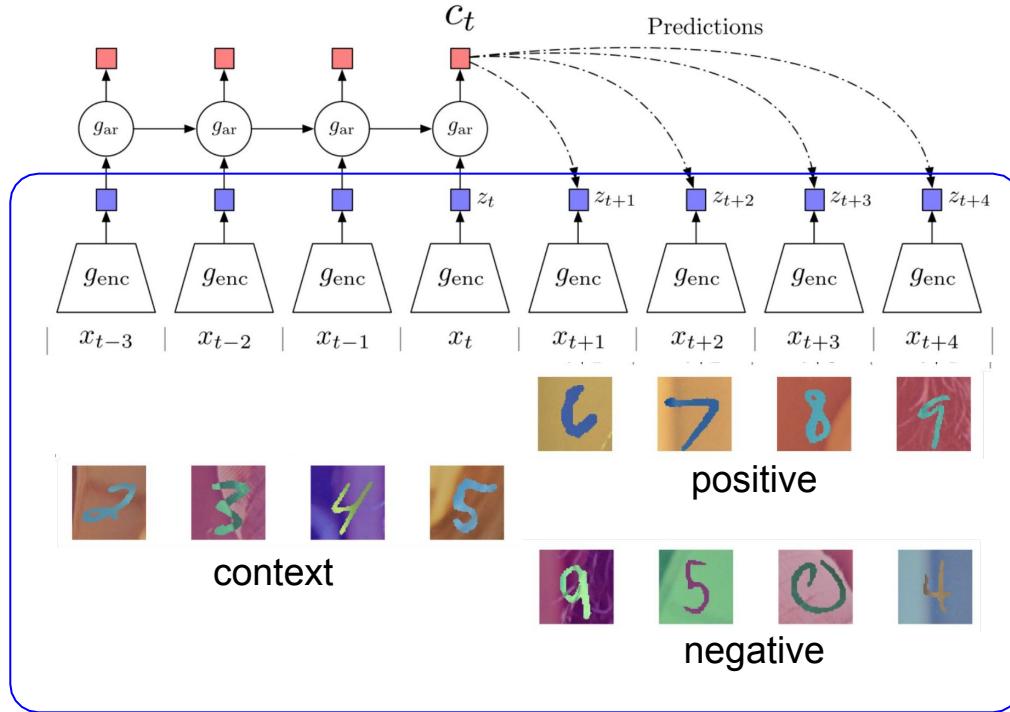
Predictive: the model has to predict future patterns given the current context.

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

Figure [source](#)

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

Contrastive Predictive Coding (CPC)

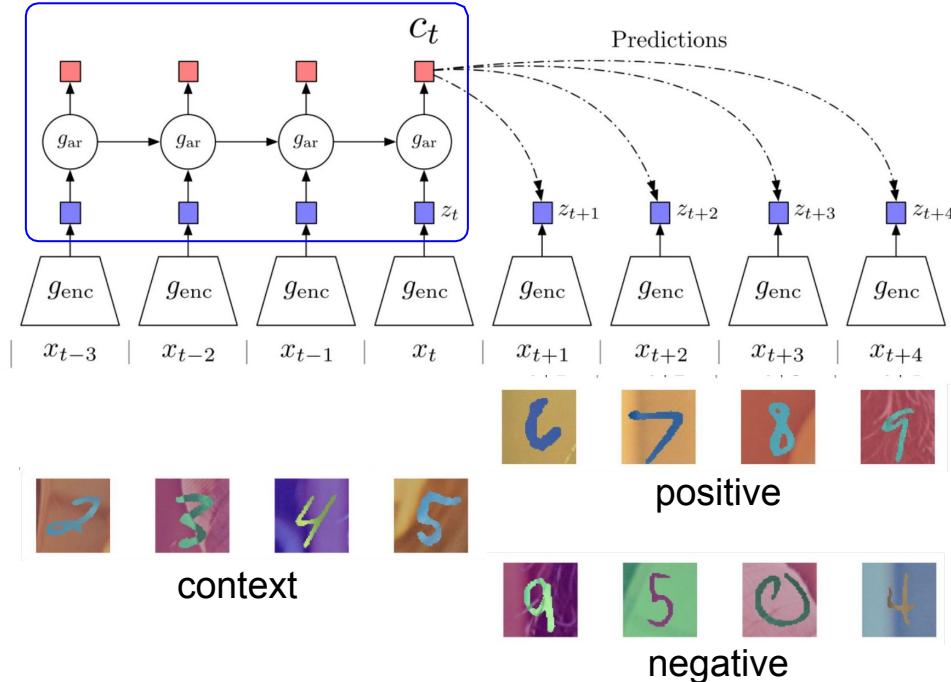


1. Encode all samples in a sequence into vectors $\mathbf{z}_t = g_{enc}(x_t)$

Figure [source](#)

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

Contrastive Predictive Coding (CPC)

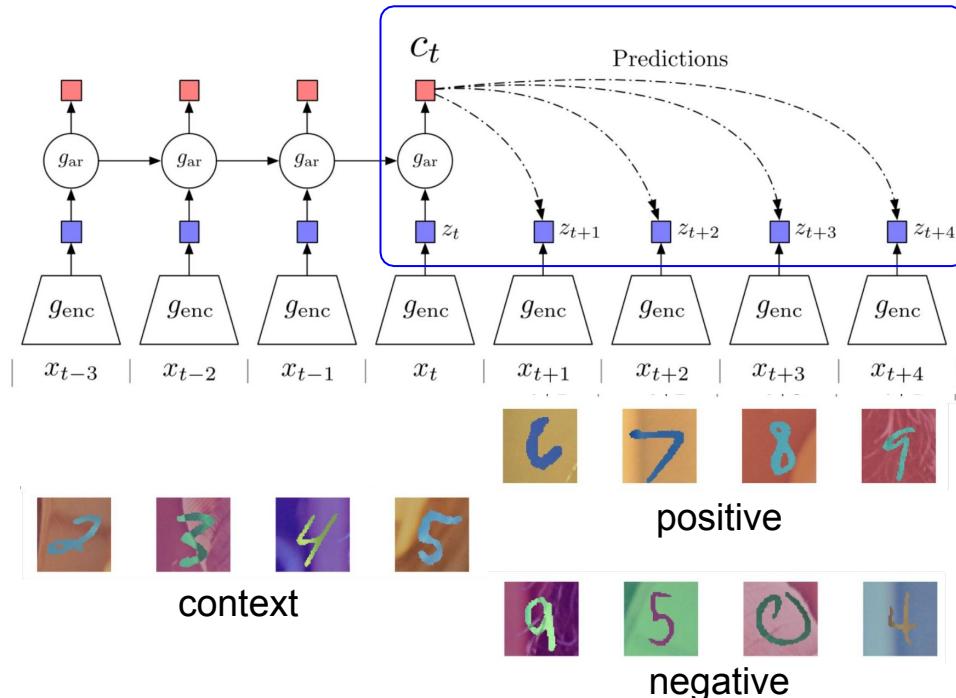


1. Encode all samples in a sequence into vectors $\mathbf{z}_t = \mathbf{g}_{\text{enc}}(\mathbf{x}_t)$
2. Summarize context (e.g., half of a sequence) into a context code \mathbf{c}_t using an auto-regressive model (\mathbf{g}_{ar}). The original paper uses GRU-RNN here.

Figure [source](#)

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

Contrastive Predictive Coding (CPC)



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

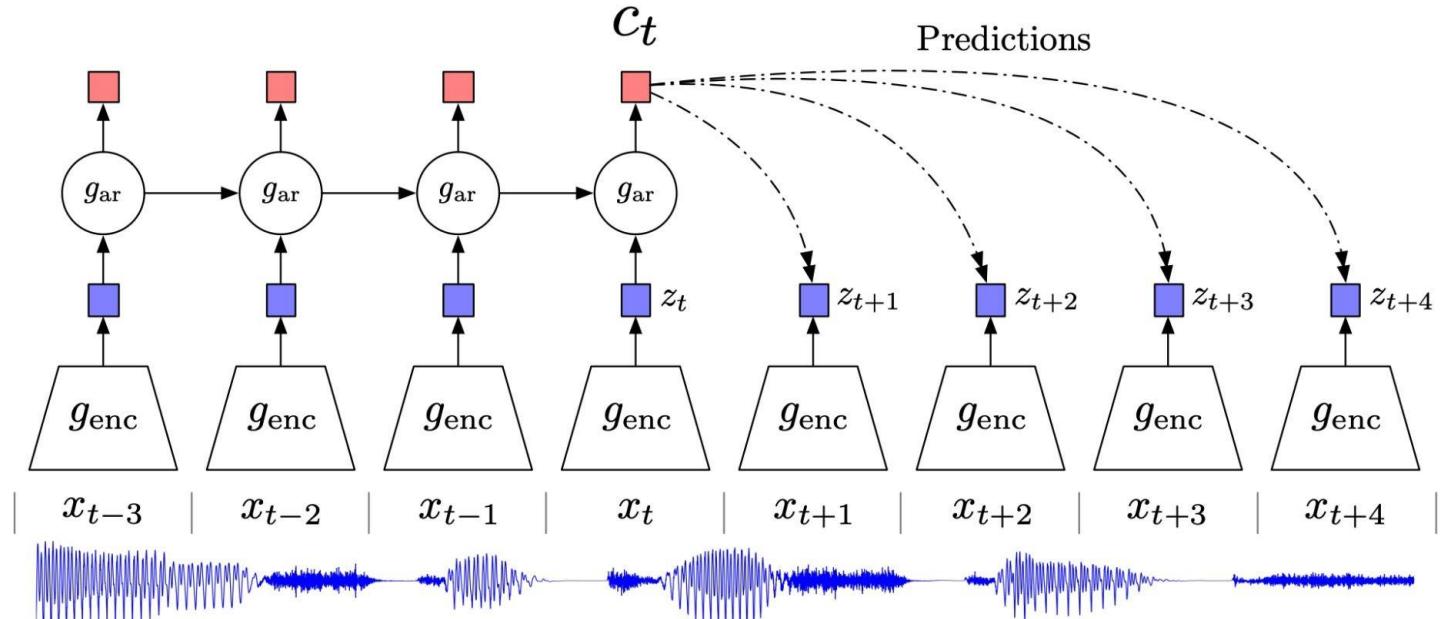
$$s_k(z_{t+k}, c_t) = z_{t+k}^T W_k c_t$$

, where W_k is a trainable matrix.

Figure [source](#)

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

CPC example: modeling audio sequences



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

CPC example: modeling audio sequences

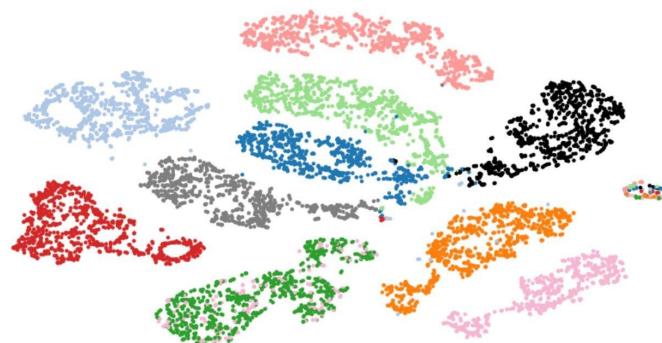


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

Method	ACC
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)

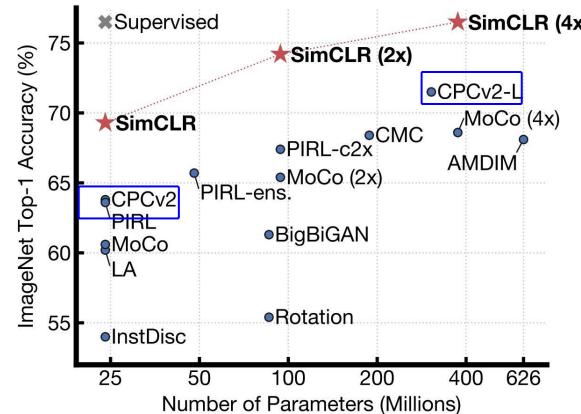
Source: [van den Oord et al., 2018](#),

CPC example: modeling visual context

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:

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

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

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

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

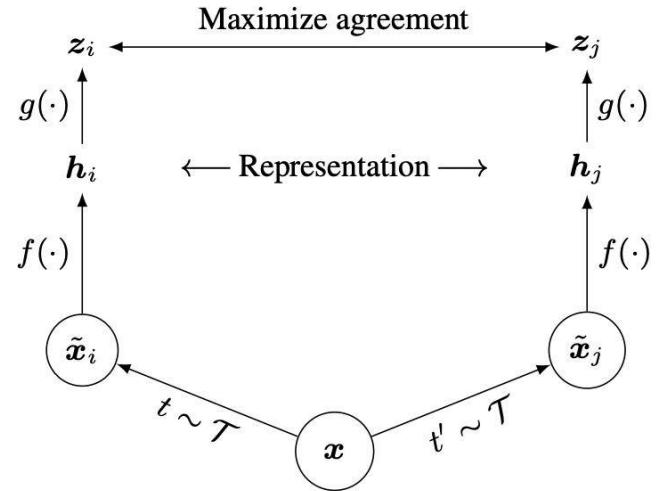
A *lower bound* on the mutual information between $f(x)$ and $f(x^+)$

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

Summary: Contrastive Representation Learning

SimCLR: a simple framework for contrastive representation learning

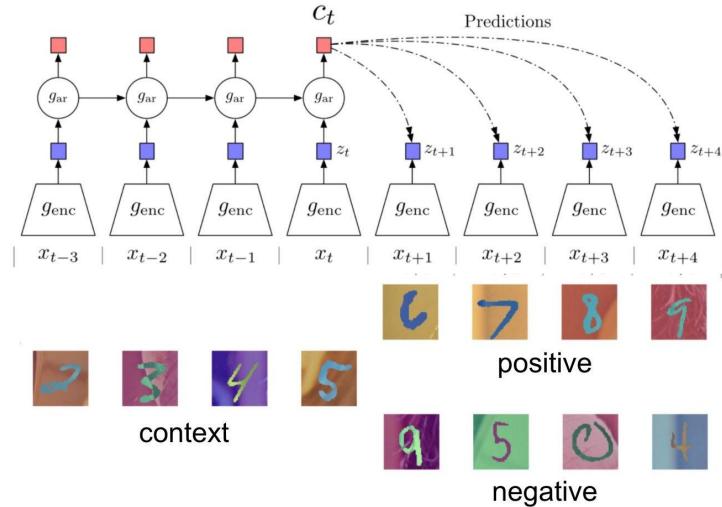
- **Key ideas**: non-linear projection head to allow flexible representation learning
- Simple to implement, effective in learning visual representation
- Requires large training batch size to be effective; large memory footprint



Summary: Contrastive Representation Learning

CPC: sequence-level contrastive learning

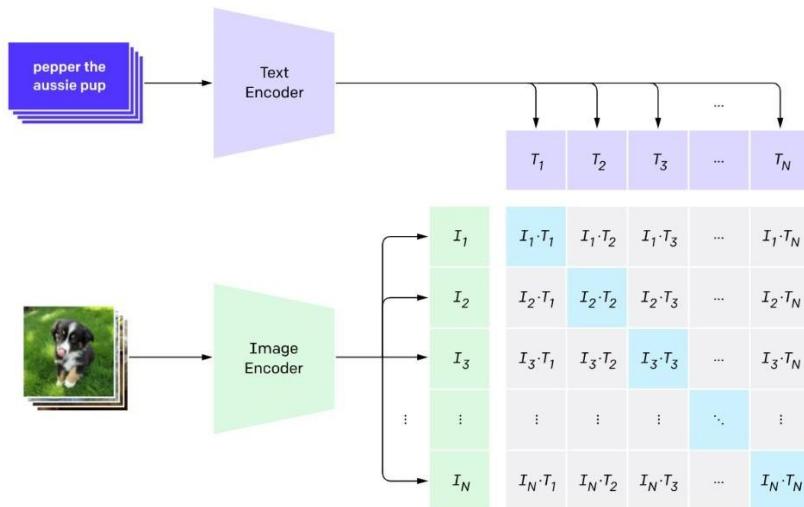
- Contrast “right” sequence with “wrong” sequence.
- InfoNCE loss with a time-dependent score function.
- Can be applied to a variety of learning problems, but not as effective in learning image representations compared to instance-level methods.



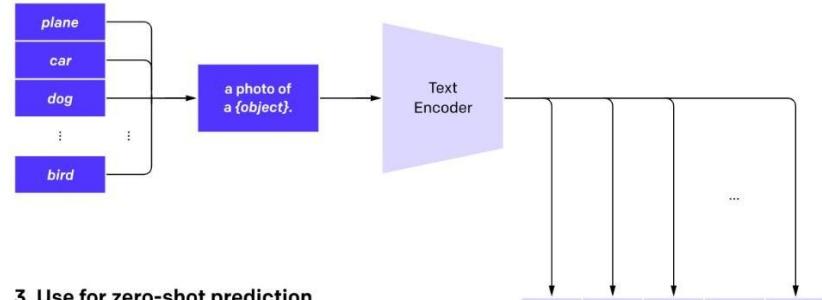
Other examples

Contrastive learning between image and natural language sentences

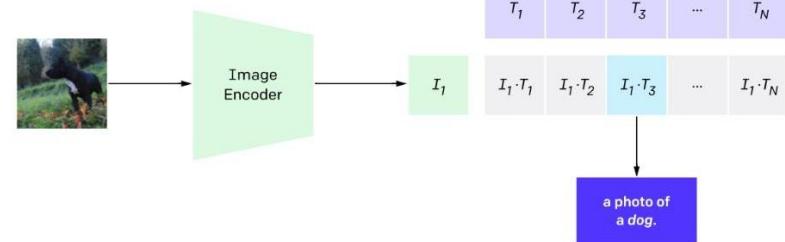
1. Contrastive pre-training



2. Create dataset classifier from label text



3. Use for zero-shot prediction



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

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.

Y. LeCun

Self-Supervised Learning

- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.
- ▶ Predict the **occluded** from the **visible**
- ▶ **Pretend there is a part of the input you don't know and predict that.**

Time →

← Past Present Future →

“The Cake of Learning”

Y. LeCun

How Much Information is the Machine Given during Learning?

downstream
tasks

feature
extractor

Learn good
features through
self-supervision

- ▶ “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**



Thank-You