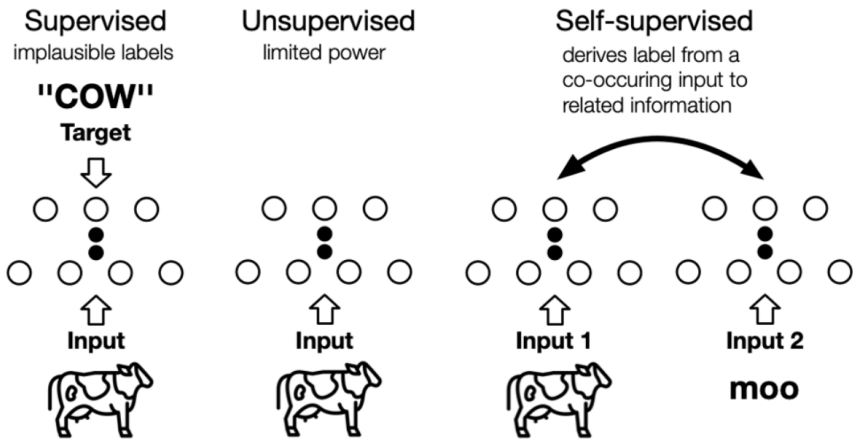
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Self-supervised Learning

(2026)

Lecturer: Xue Yang

yangxue.site





上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Contents

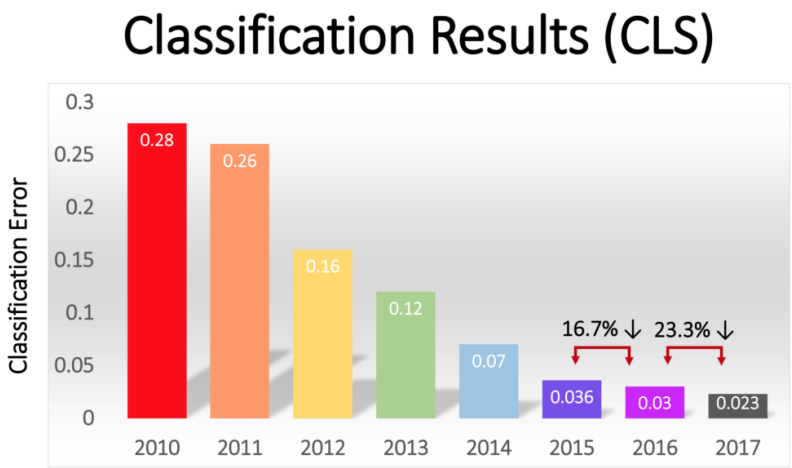
- Introduction
- Traditional SSL
- Modern SSL



ImageNet Story

- ImageNet is the large-scale benchmark for image classification
  - ✓ training: 1k categories, 1k images per category
  - ✓ testing: 100k images





ILSVRC

Lots of Computer Vision Tasks

Classification



CAT

Semantic Segmentation



GRASS, CAT, TREE, SKY

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

To some extent, any visual task can now be solved

- Collect a large-scale dataset labelled for the task of interest
- Specify a training objective and neural network architecture
- Train the model and deploy
- Good and sufficient data is all you need

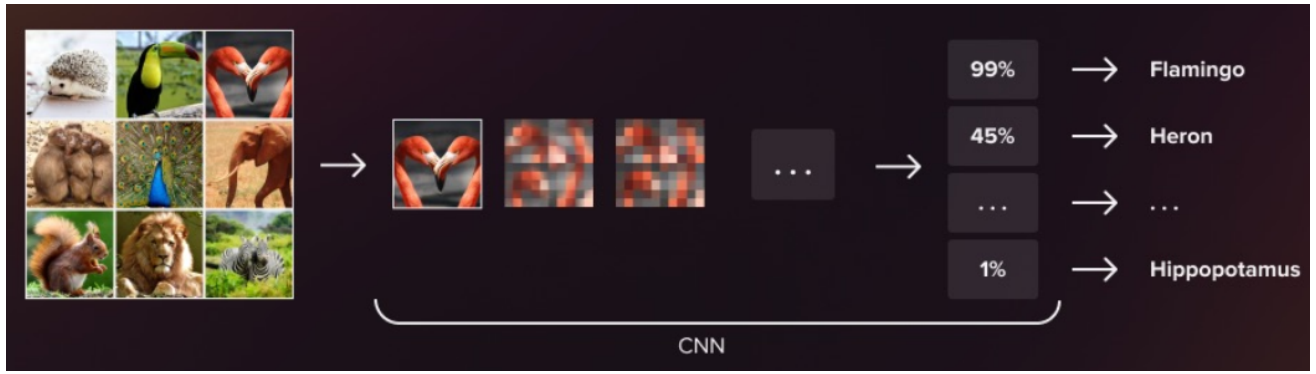
The key question is: how can we train a high-performance model with limited data?

Transfer Learning

- **Train from Scratch**

Given a dataset with large number of manual annotations:

$$D_{src} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$
$$\hat{y}_i = \Phi_{\text{cls}}(\cdot) \circ \Phi_{\text{feat}}(x_i; \Theta)$$



- **Transfer Learning**

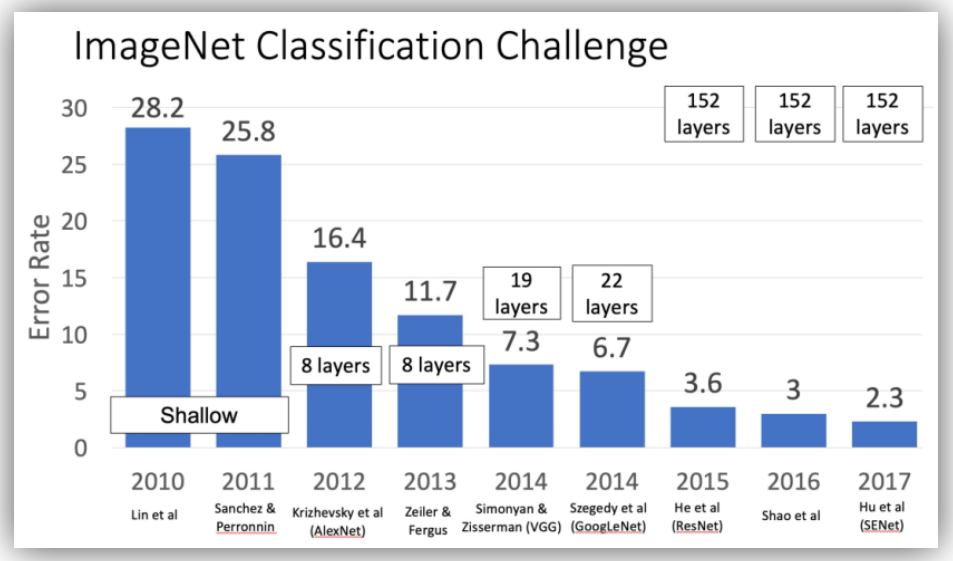
Given a dataset with small number of manual annotations (k << n):

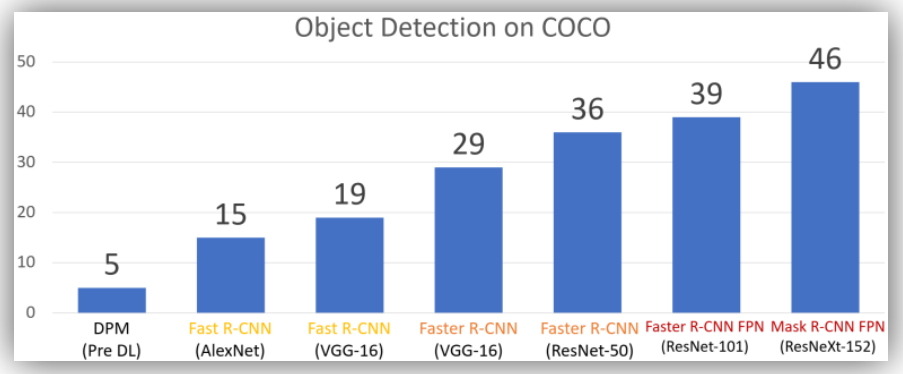
$$D_{tgt} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$
$$\hat{y}_i = \Psi_{\text{cls}}(\cdot) \circ \Phi_{\text{feat}}(x_i; \Theta)$$



Transfer Learning

- Improvements in CNN architectures lead to improvements in many downstream tasks thanks to transfer learning!

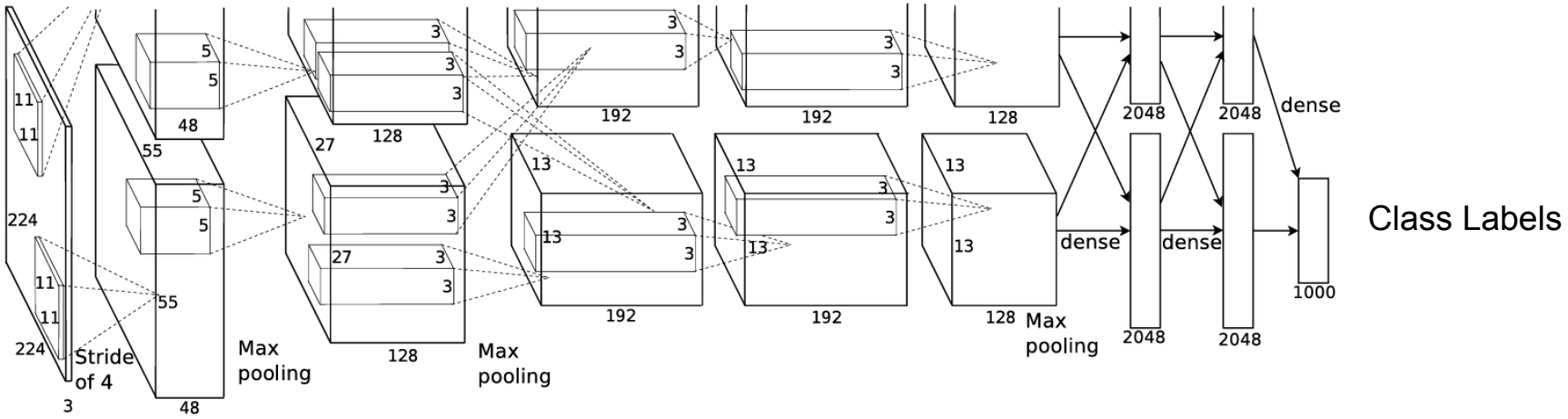




Learned Representations

- What is the problem with large-scale training?
  - ✓ We need a lot of labeled data
- Is there a way we can train neural networks without the need for huge manually labeled datasets?





Introduction

Here, we find it

I always knew unsupervised learning was the right thing to do.

---Geoffrey E. Hinton

Basically it's the idea of learning to represent the world before learning a task, and this is what babies do.

--- Yann LeCun

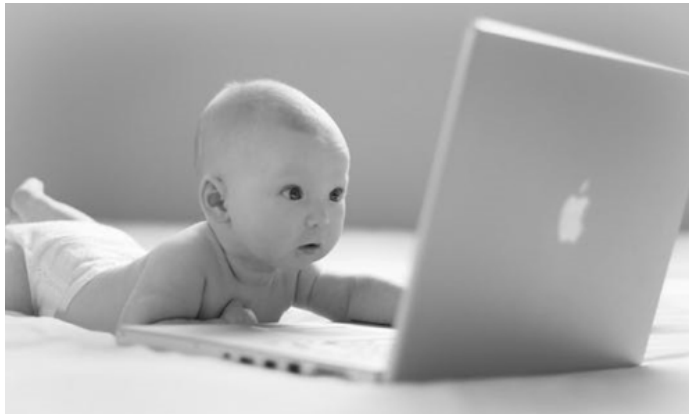
And so if we can build models of the world where we have the right abstractions, where we can pin down those changes to just one or a few variables, then we will be able to adapt to those changes because we don't need as much data, as much observation in order to figure out what has changed.

--- Yoshua Bengio



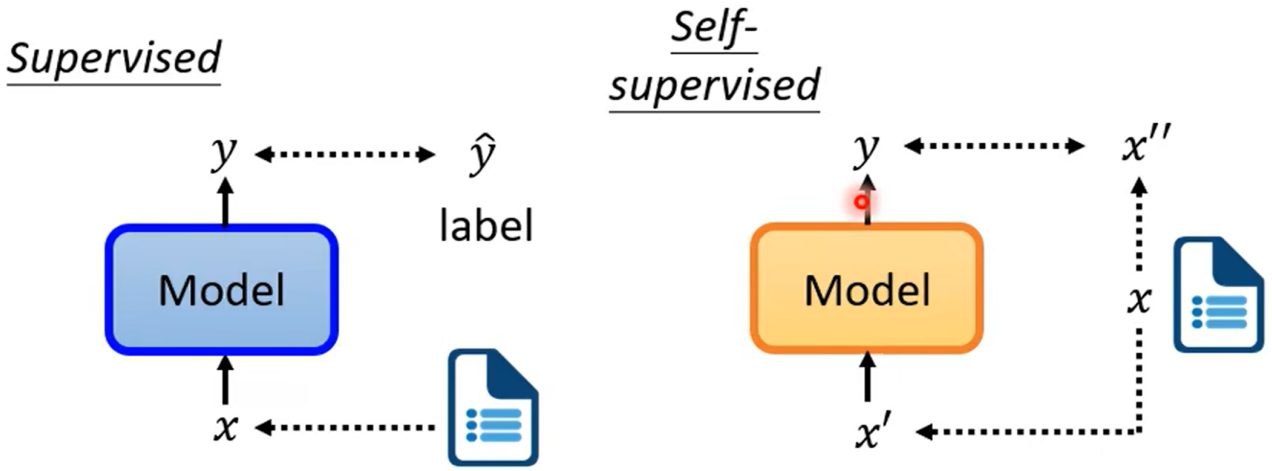
Why self-supervised learning (SSL)

- Human infants adopt very different ways for learning:
  - ✓ mainly from observation and manipulation
  - ✓ only require few explicit supervision
  - ✓ learn from multimodal data, e.g. vision, sound, touch, etc.
  - ✓ learn progressively, e.g. curriculum learning



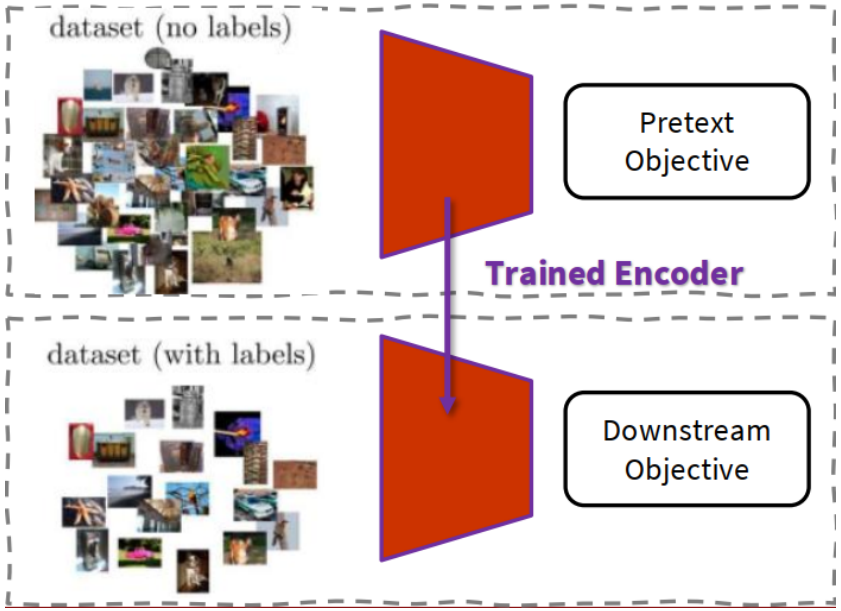
Why self-supervised learning (SSL)

- Self-supervised learning is an alternative for learning representation:
  - ✓ save cost for manual annotations
  - ✓ more scalable training, e.g. billions of images are generated everyday
  - ✓ more robust visual representation learning



Why self-supervised learning (SSL)

- A form of unsupervised learning where the data itself provides the supervision.
- With hold some part of the data, and task the network with predicting it, termed as pretext task.
- While solving the pretext task, the network is forced to learn what we really care about, e.g. a semantic representation.



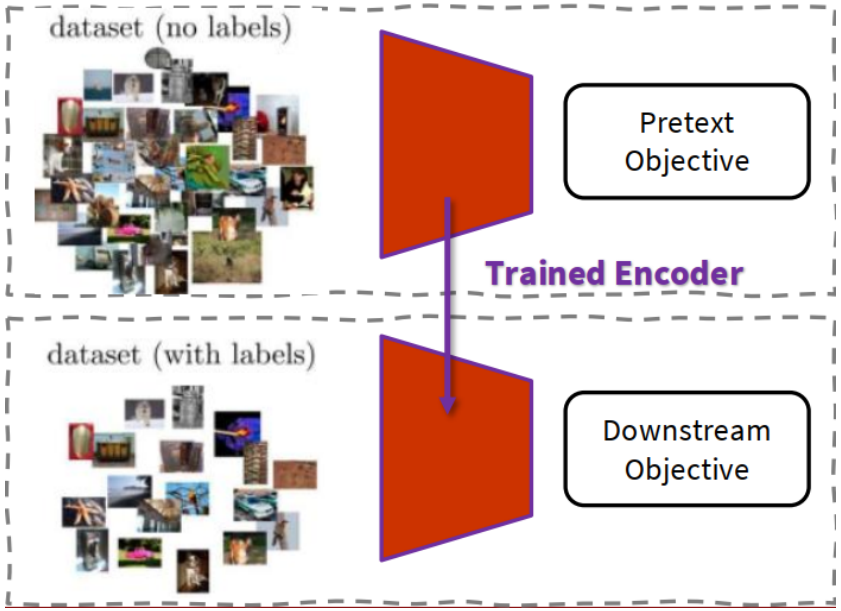
What is self-supervised learning (SSL)

Pretext Task

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

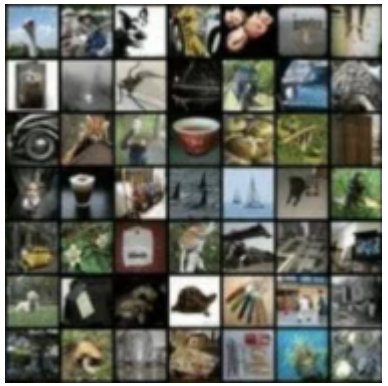
Downstream Task

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



Pretext Task

Dataset (no labels)



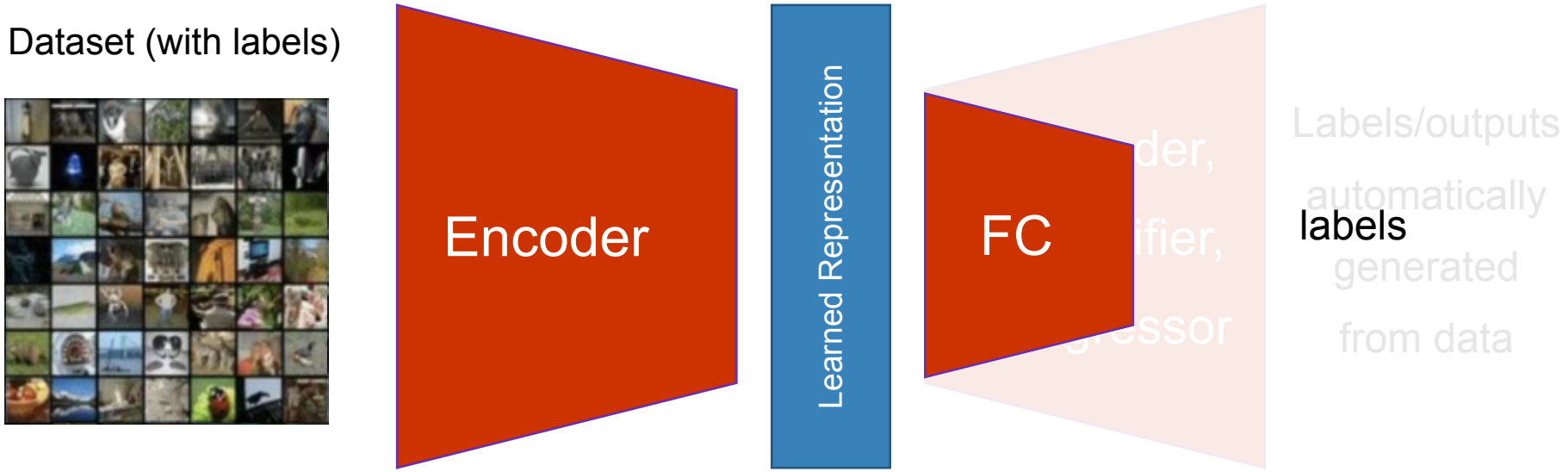
Encoder

Learned Representation

Decoder, Classifier, Regressor

Labels/outputs automatically generated from data

Downstream Task



Pretext Tasks

- Example: learn to predict image transformations / complete corrupted images
  - ✓ Solving the pretext tasks allow the model to learn good features.
  - ✓ We can automatically generate labels for the pretext tasks.

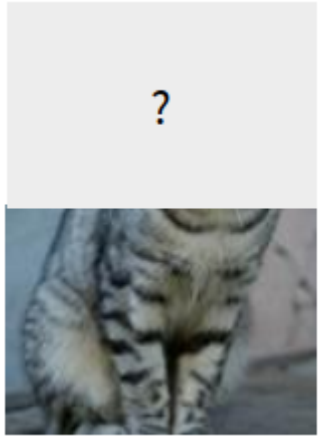


image completion



rotation prediction



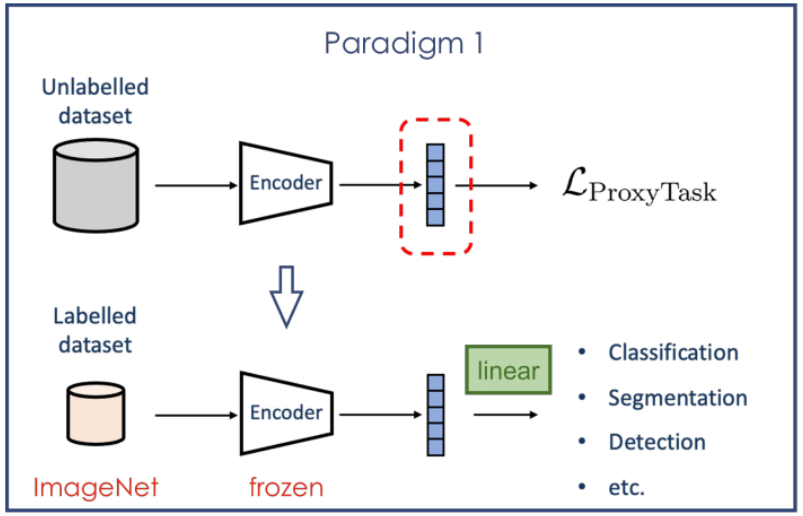
“jigsaw puzzle”

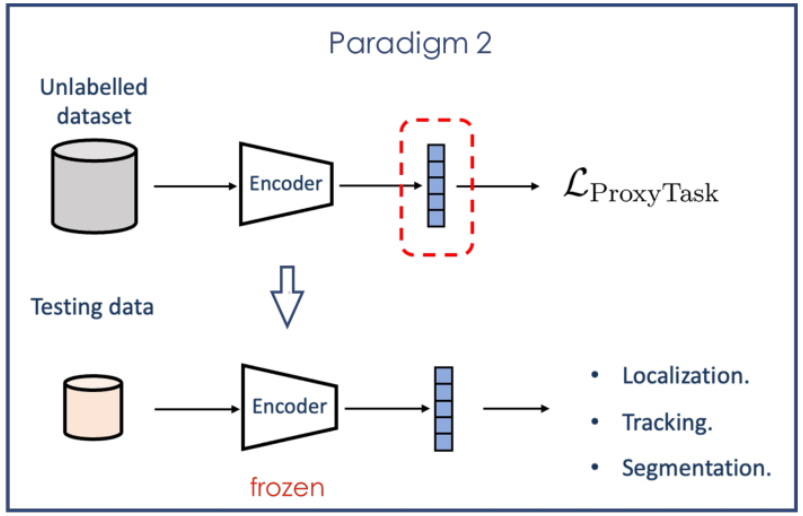


colorization

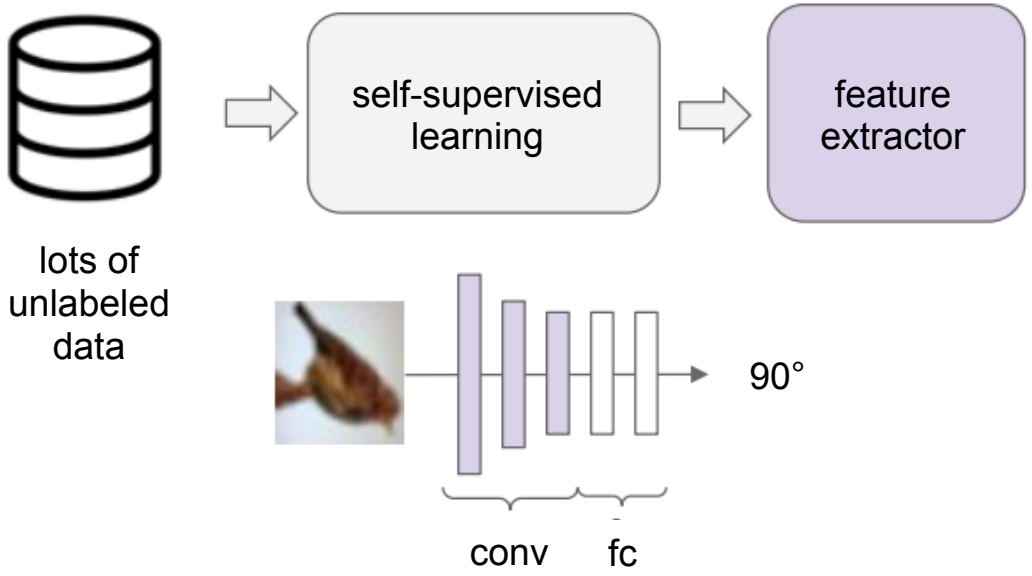
How to evaluate a self-supervised learning method?

- There are generally two paradigms for evaluating the representation:
  - ✓ Evaluate on ImageNet with linear probing, i.e. freeze the feature encoder, and only train the classifier with labels
  - ✓ Evaluate on downstream tasks without fine-tuning, e.g. zero-shot transfer



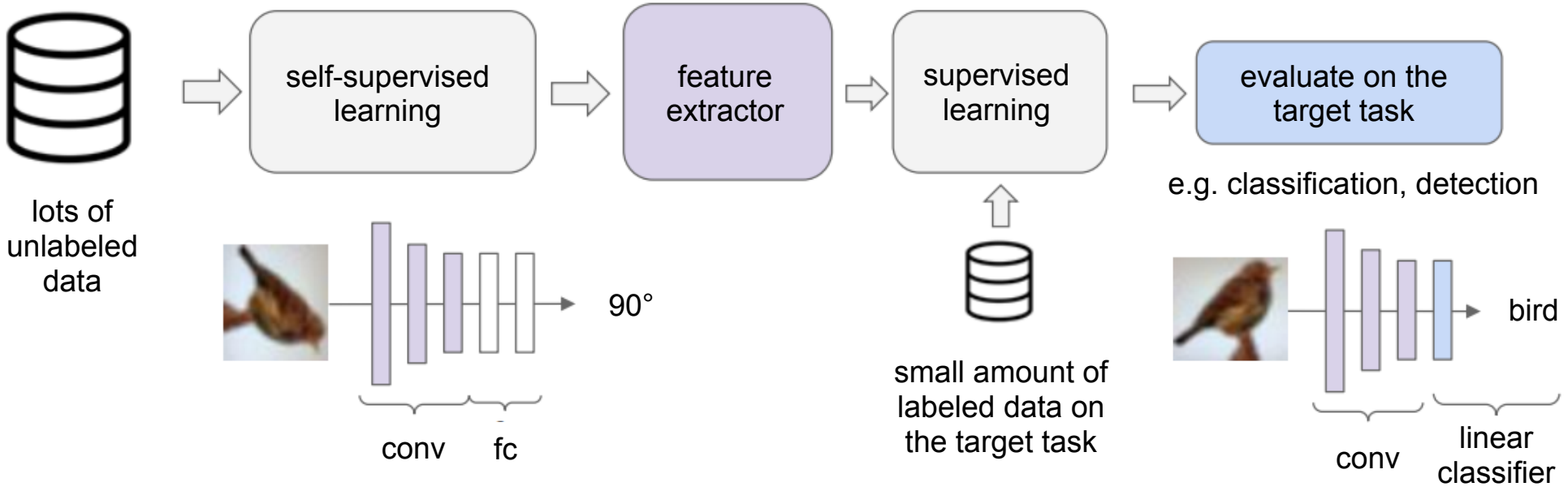


How to evaluate a self-supervised learning method?



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

Introduction

Broader picture

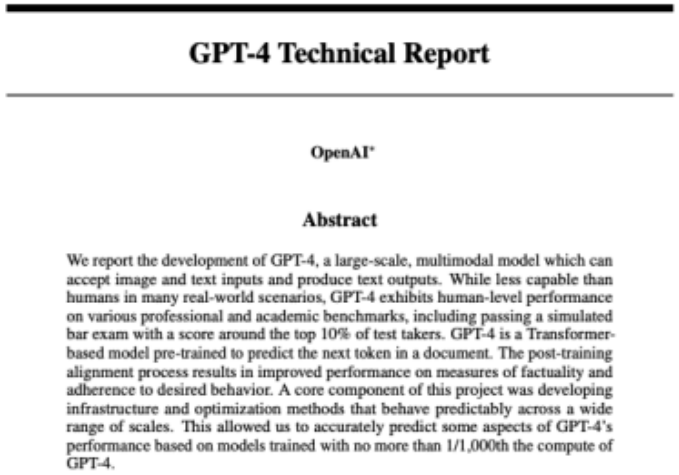
Today's lecture

computer vision



Doersch et al., 2015

language modeling



GPT-4 Technical Report

OpenAI*

Abstract

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

GPT-4 (OpenAI 2023)

speech synthesis



Wavenet (van den Oord et al.,
2016)

robot / reinforcement learning



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

• • •

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

Contents

- Introduction
- **Traditional SSL**
  - **Rotation, Inpainting, Rearrangement, Coloring**
  - **Reconstruction-based learning: MAE**
- Modern SSL

Pretext task: predict relative patch locations

Unsupervised Visual Representation Learning by Context Prediction

Carl Doersch[1,2] Abhinav Gupta[1] Alexei A. Efros[2]

[1] School of Computer Science
Carnegie Mellon University

[2] Dept. of Electrical Engineering and Computer Science
University of California, Berkeley

Abstract

This work explores the use of spatial context as a source of free and plentiful supervisory signal for training a rich visual representation. Given only a large, unlabeled image collection, we extract random pairs of patches from each image and train a convolutional neural net to predict the position of the second patch relative to the first. We argue that doing well on this task requires the model to learn to recognize objects and their parts. We demonstrate that the feature representation learned using this within-image context indeed captures visual similarity across images. For example, this representation allows us to perform unsupervised visual discovery of objects like cats, people, and even birds from the Pascal VOC 2011 detection dataset. Furthermore, we show that the learned ConvNet can be used in the R-

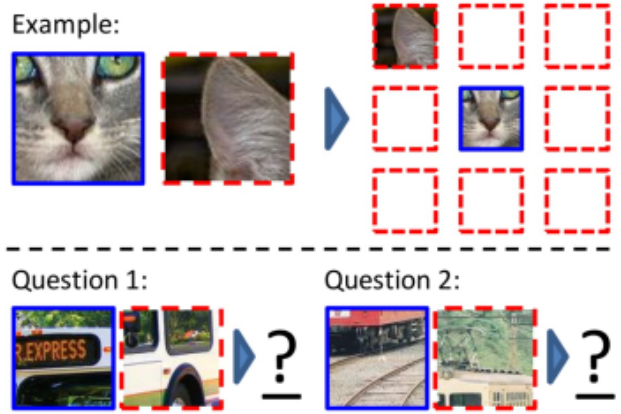
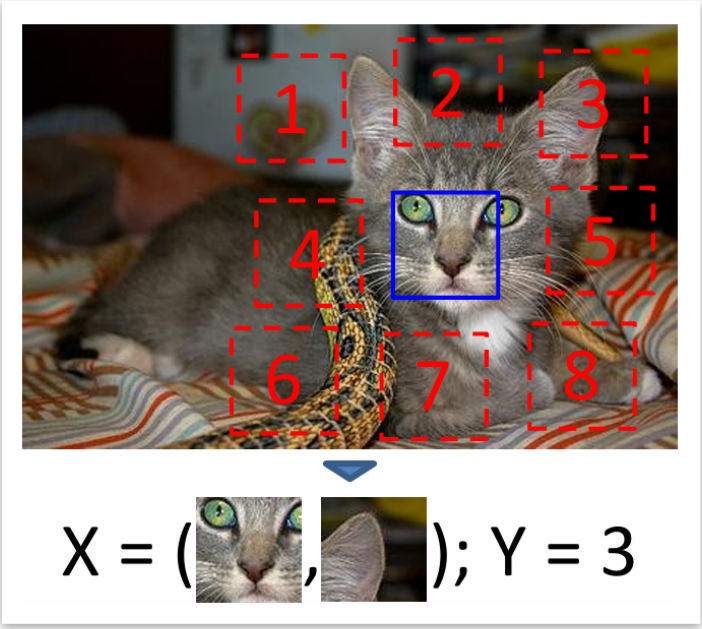
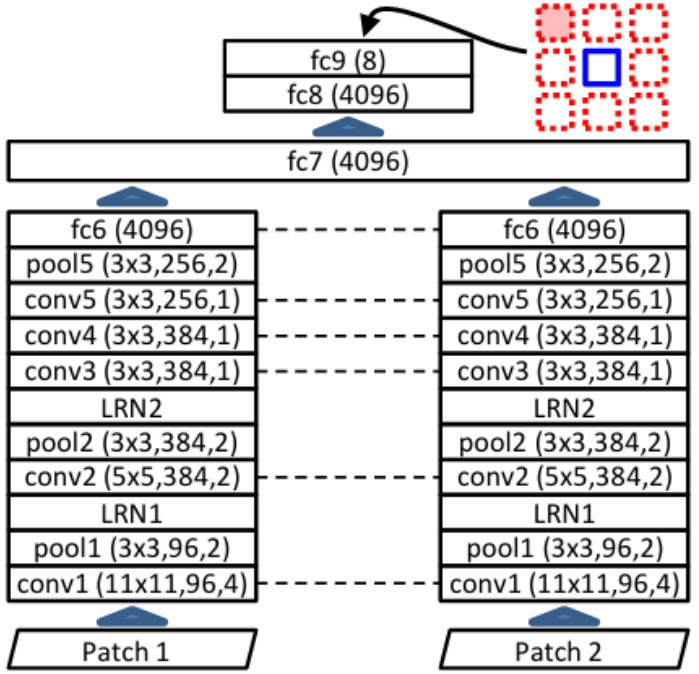


Figure 1. Our task for learning patch representations involves randomly sampling a patch (blue) and then one of eight possible neighbors (red). Can you guess the spatial configuration for the two pairs of patches? Note that the task is much easier once you

Pretext task: predict relative patch locations

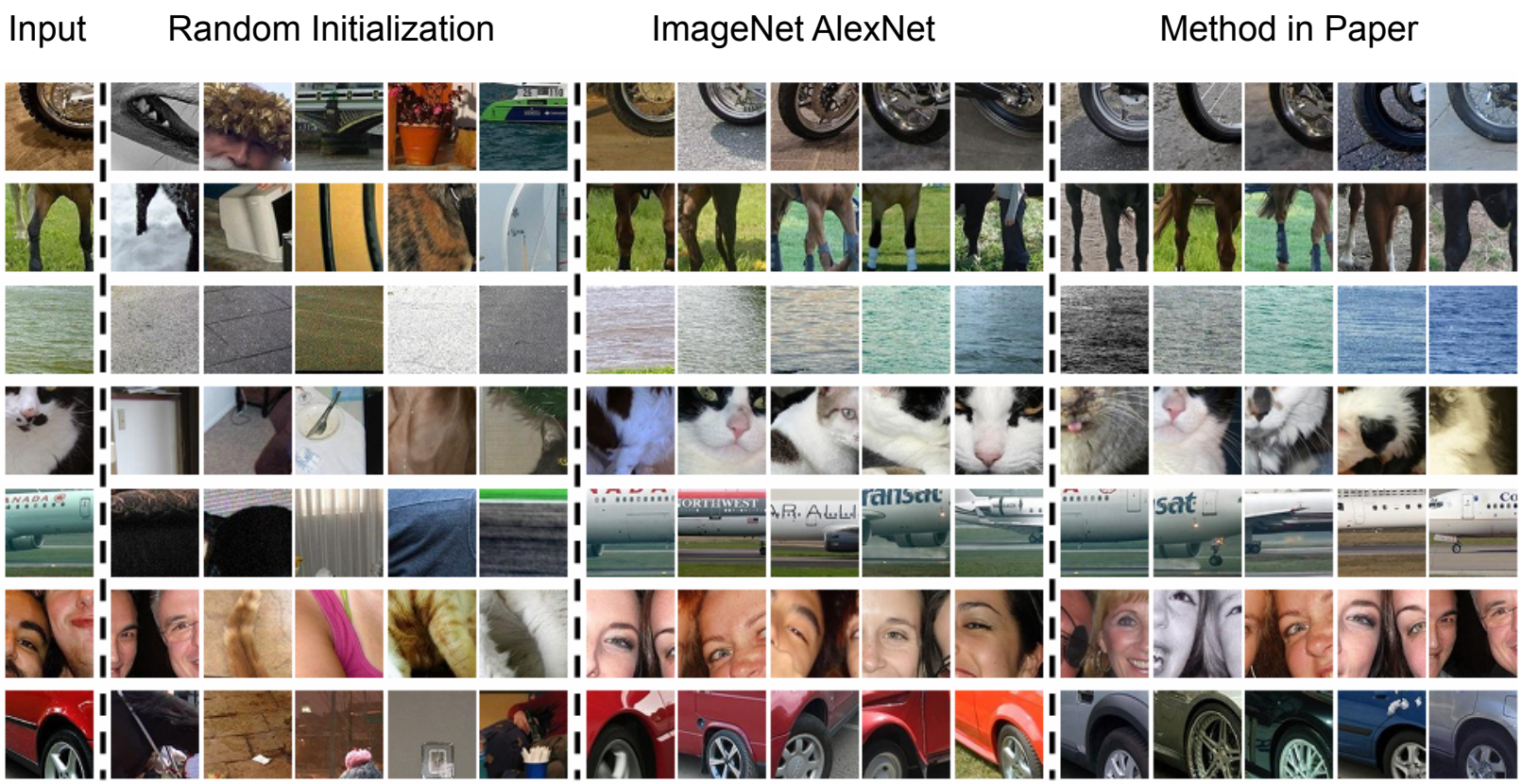
- Randomly sample patches, and task the network to predict their relative position.
- If the model solves this task, it may have learnt useful representation on object structure.





Traditional Self-supervised Learning

What does pretext task learn? ---Nearest Neighbors



What does pretext task learn? ---Object Detection

VOC-2007 Test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM-v5[17]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
[8] w/o context	52.6	52.6	19.2	25.4	18.7	47.3	56.9	42.1	16.6	41.4	41.9	27.7	47.9	51.5	29.9	20.0	41.1	36.4	48.6	53.2	38.5
Regionlets[58]	54.2	52.0	20.3	24.0	20.1	55.5	68.7	42.6	19.2	44.2	49.1	26.6	57.0	54.5	43.4	16.4	36.6	37.7	59.4	52.3	41.7
Scratch-R-CNN[2]	49.9	60.6	24.7	23.7	20.3	52.5	64.8	32.9	20.4	43.5	34.2	29.9	49.0	60.4	47.5	28.0	42.3	28.6	51.2	50.0	40.7
Scratch-Ours	52.6	60.5	23.8	24.3	18.1	50.6	65.9	29.2	19.5	43.5	35.2	27.6	46.5	59.4	46.5	25.6	42.4	23.5	50.0	50.6	39.8
Ours-projection	58.4	62.8	33.5	27.7	24.4	58.5	68.5	41.2	26.3	49.5	42.6	37.3	55.7	62.5	49.4	29.0	47.5	28.4	54.7	56.8	45.7
Ours-color-dropping	60.5	66.5	29.6	28.5	26.3	56.1	70.4	44.8	24.6	45.5	45.4	35.1	52.2	60.2	50.0	28.1	46.7	42.6	54.8	58.6	46.3
Ours-Yahoo100m	56.2	63.9	29.8	27.8	23.9	57.4	69.8	35.6	23.7	47.4	43.0	29.5	52.9	62.0	48.7	28.4	45.1	33.6	49.0	55.5	44.2
ImageNet-R-CNN[21]	64.2	69.7	50	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
K-means-rescale [31]	55.7	60.9	27.9	30.9	12.0	59.1	63.7	47.0	21.4	45.2	55.8	40.3	67.5	61.2	48.3	21.9	32.8	46.9	61.6	51.7	45.6
Ours-rescale [31]	61.9	63.3	35.8	32.6	17.2	68.0	67.9	54.8	29.6	52.4	62.9	51.3	67.1	64.3	50.5	24.4	43.7	54.9	67.1	52.7	51.1
ImageNet-rescale [31]	64.0	69.6	53.2	44.4	24.9	65.7	69.6	69.2	28.9	63.6	62.8	63.9	73.3	64.6	55.8	25.7	50.5	55.4	69.3	56.4	56.5
VGG-K-means-rescale	56.1	58.6	23.3	25.7	12.8	57.8	61.2	45.2	21.4	47.1	39.5	35.6	60.1	61.4	44.9	17.3	37.7	33.2	57.9	51.2	42.4
VGG-Ours-rescale	71.1	72.4	54.1	48.2	29.9	75.2	78.0	71.9	38.3	60.5	62.3	68.1	74.3	74.2	64.8	32.6	56.5	66.4	74.0	60.3	61.7
VGG-ImageNet-rescale	76.6	79.6	68.5	57.4	40.8	79.9	78.4	85.4	41.7	77.0	69.3	80.1	78.6	74.6	70.1	37.5	66.0	67.5	77.4	64.9	68.6

Table 1. Mean Average Precision on VOC-2007.

Pretext task: solving “jigsaw puzzles”

Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

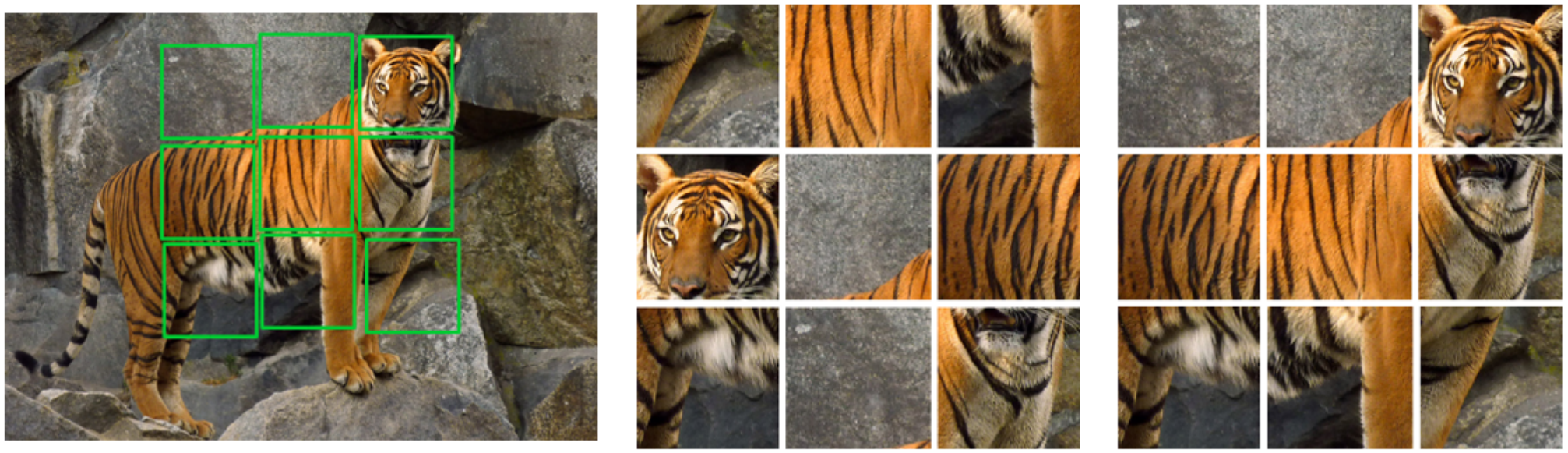
Mehdi Noroozi and Paolo Favaro

Institute for Informatiks
University of Bern
{noroozi,paolo.favaro}@inf.unibe.ch

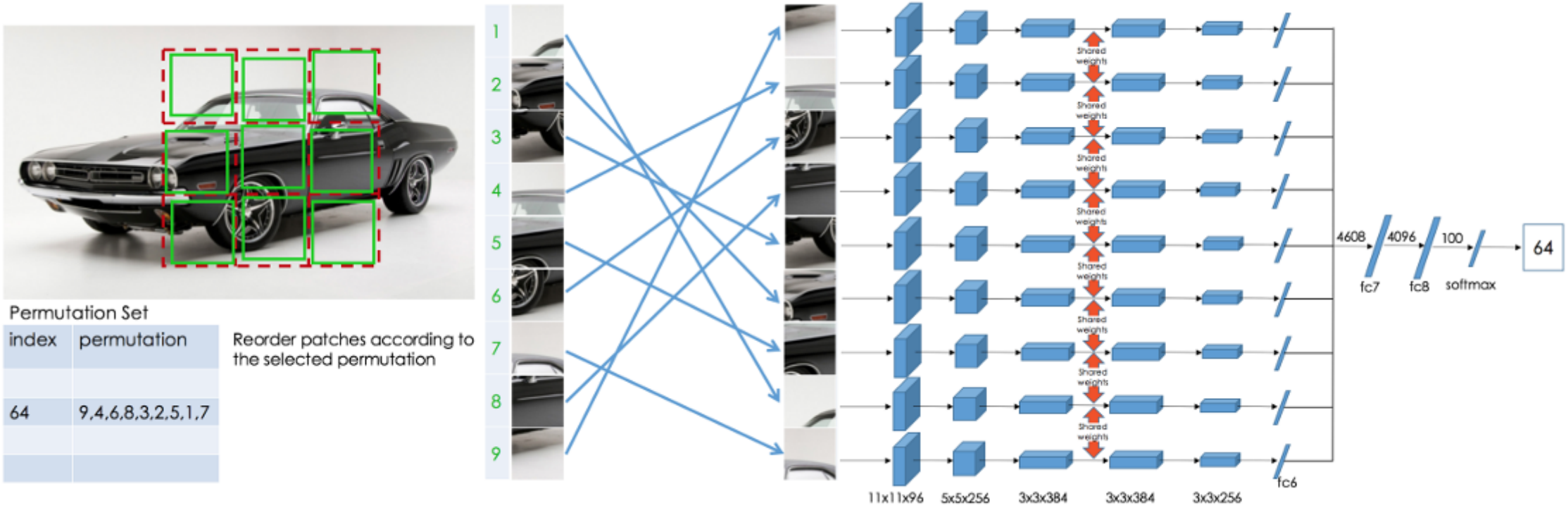
Abstract. In this paper we study the problem of image representation learning without human annotation. By following the principles of self-supervision, we build a convolutional neural network (CNN) that can be trained to solve Jigsaw puzzles as a *pretext* task, which requires no

Pretext task: solving “jigsaw puzzles”

- Randomly shuffle the image patches.
- Train a network to recover the image, e.g. their relative positions.
- Scale the pairwise relation to multi-patch permutations.



Pretext task: solving “jigsaw puzzles”



Pretext task: solving "jigsaw puzzles"

- "Ours" is feature learned from solving image Jigsaw
- Doersch et al. is the method with relative patch location

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%**

Pretext task: predict rotations

UNSUPERVISED REPRESENTATION LEARNING BY PREDICTING IMAGE ROTATIONS

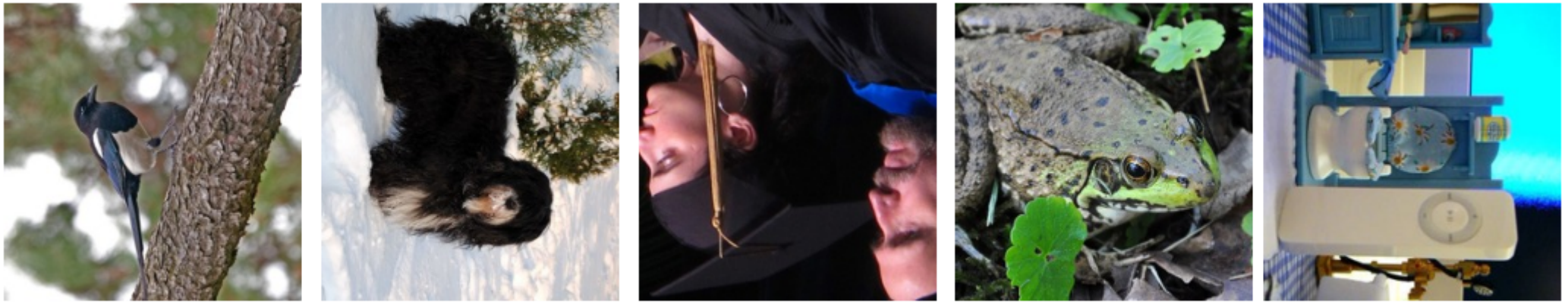
Spyros Gidaris, Praveer Singh, Nikos Komodakis
University Paris-Est, LIGM
Ecole des Ponts ParisTech
{spyros.gidaris,praveer.singh,nikos.komodakis}@enpc.fr

ABSTRACT

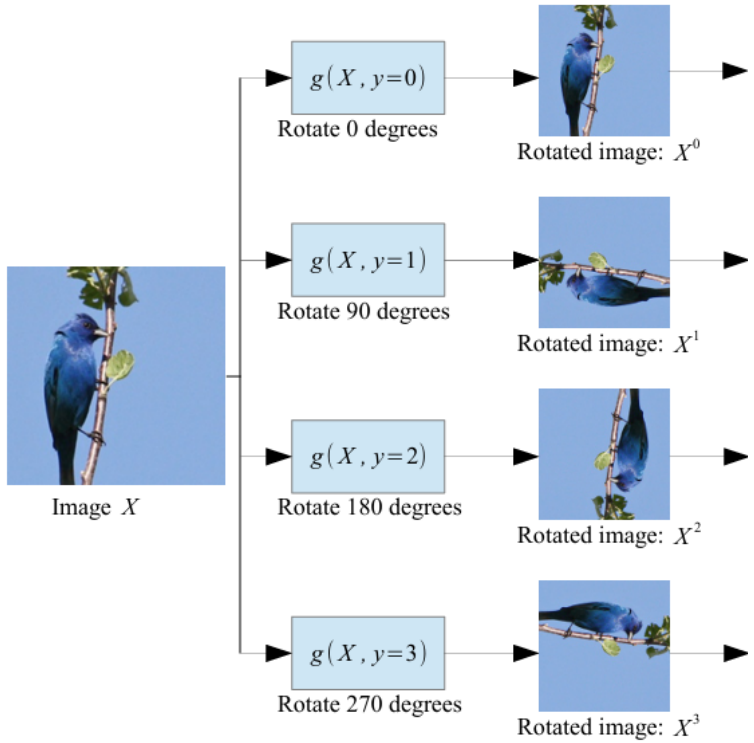
Over the last years, deep convolutional neural networks (ConvNets) have transformed the field of computer vision thanks to their unparalleled capacity to learn high level semantic image features. However, in order to successfully learn those features, they usually require massive amounts of manually labeled data, which is both expensive and impractical to scale. Therefore, unsupervised semantic feature learning, i.e., learning without requiring manual annotation effort, is of crucial importance in order to successfully harvest the vast amount of visual data that are

Pretext task: predict rotations

- **Hypothesis (rotation invariance)**: 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.



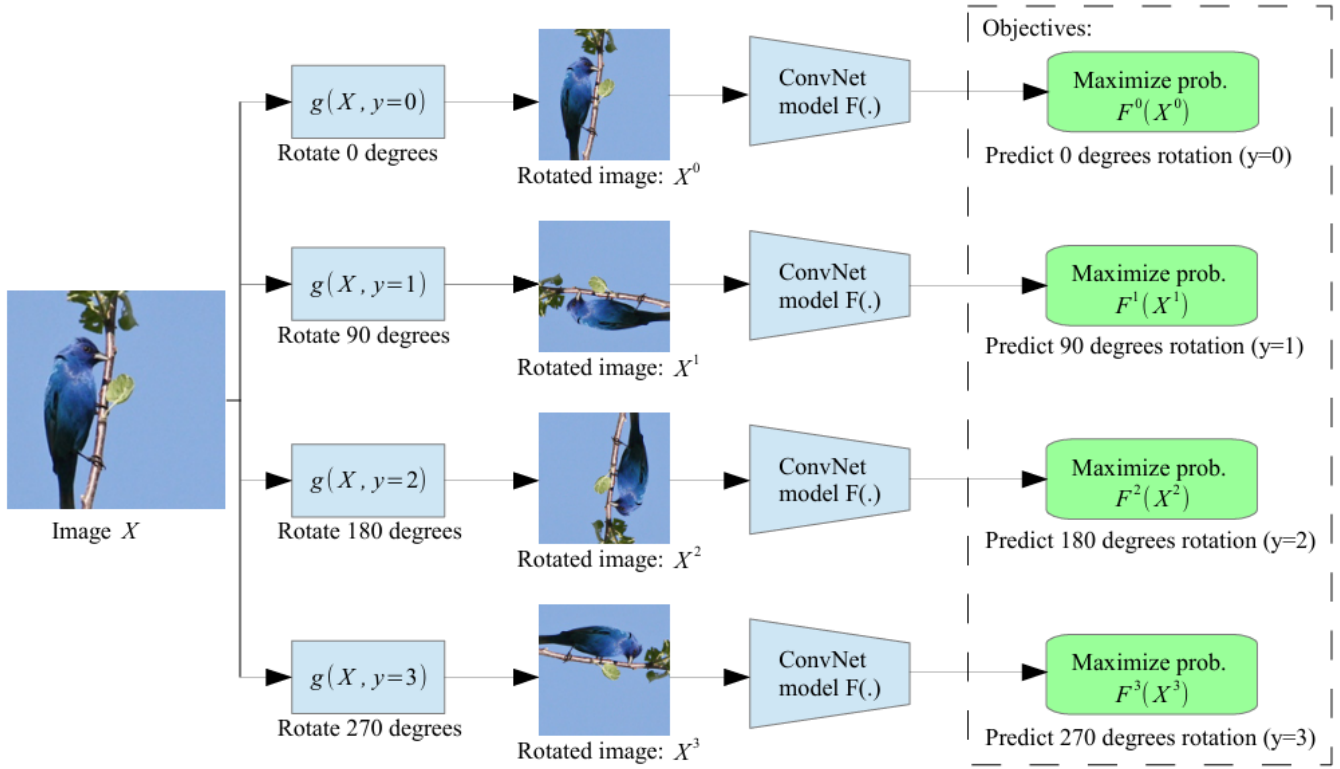
Pretext task: predict rotations



Self-supervised learning by rotating the entire input images.

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

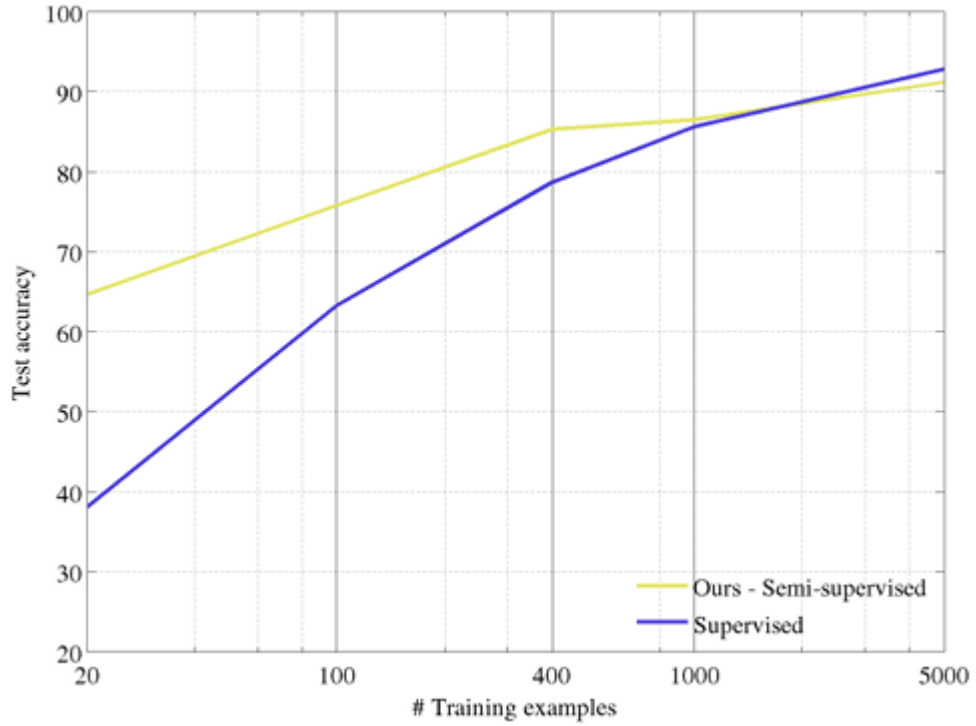
Pretext task: predict rotations



Self-supervised learning by rotating the entire input images.

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

Pretext task: predict rotations



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

Freeze conv1 + conv2

Learn conv3 + linear layers with subset of labeled CIFAR10 data (classification).

Pretext task: predict rotations

	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)

Finetune on labeled data from Pascal VOC 2007.

Pretext task: image coloring

Colorful Image Colorization

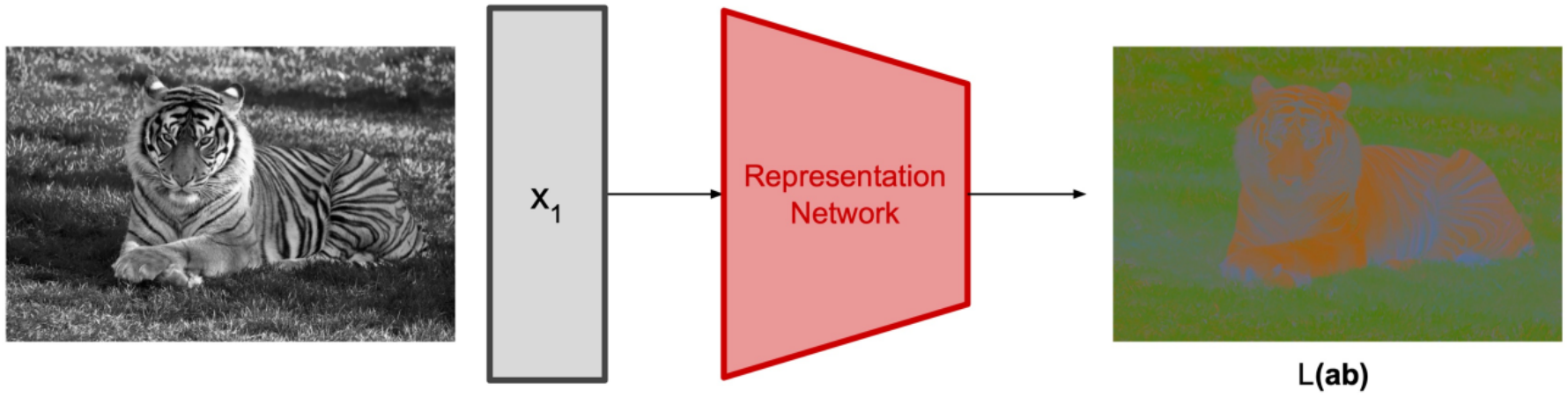
Richard Zhang, Phillip Isola, Alexei A. Efros
{rich.zhang,isola,efros}@eecs.berkeley.edu

University of California, Berkeley

Abstract. Given a grayscale photograph as input, this paper attacks the problem of hallucinating a *plausible* color version of the photograph. This problem is clearly underconstrained, so previous approaches have either relied on significant user interaction or resulted in desaturated colorizations. We propose a fully automatic approach that produces vibrant and realistic colorizations. We embrace the underlying uncertainty of the

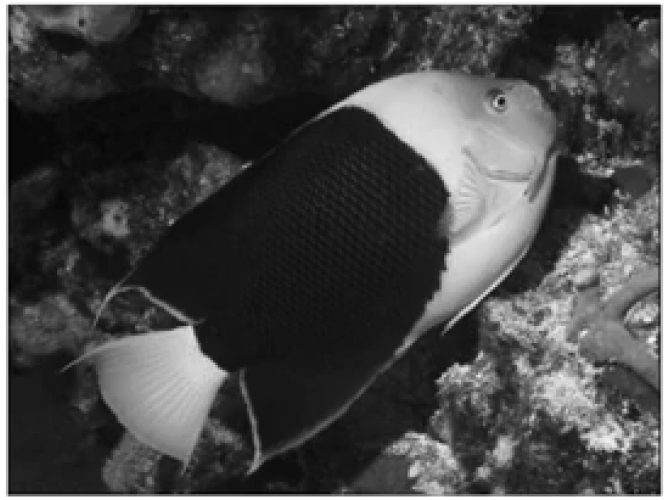
Pretext task: image coloring

- Feed grayscale images as input, and task the model to colorize it.
- Intuitively, if the network can colorize the image, it has to understand what the objects look like, e.g. the object patterns.



Pretext task: image coloring

- Lab Color Space:



Grayscale image: *L* channel

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

L



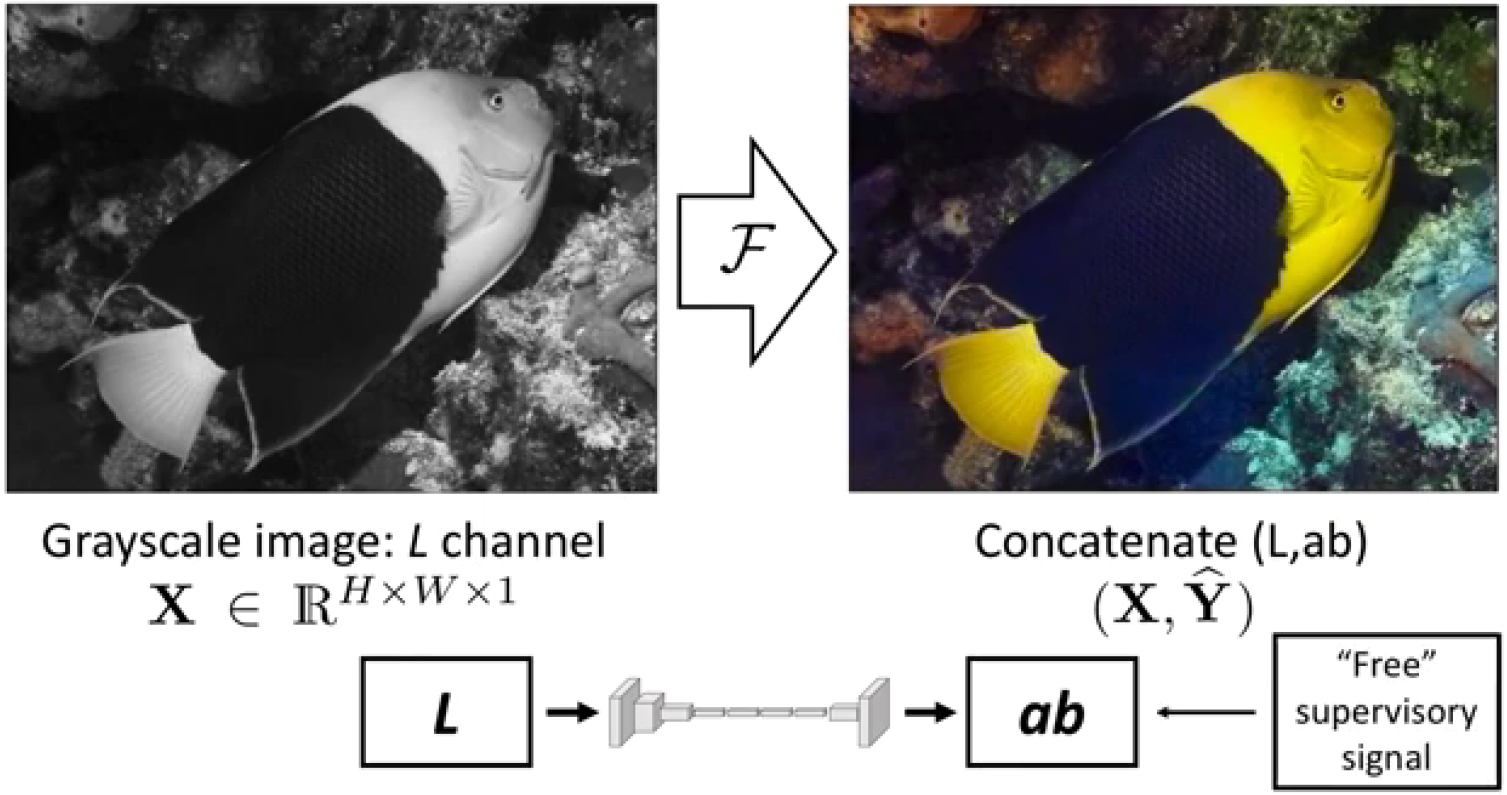
Color information: *ab* channels

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

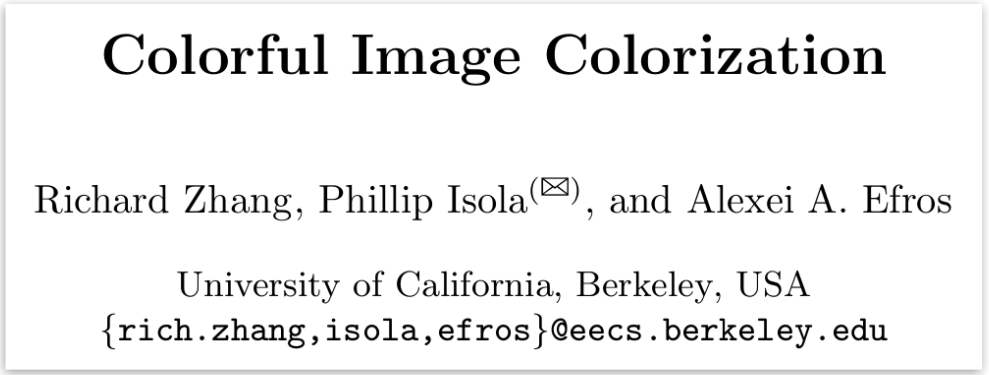
ab

Pretext task: image coloring

- Lab Color Space:



Pretext task: image coloring

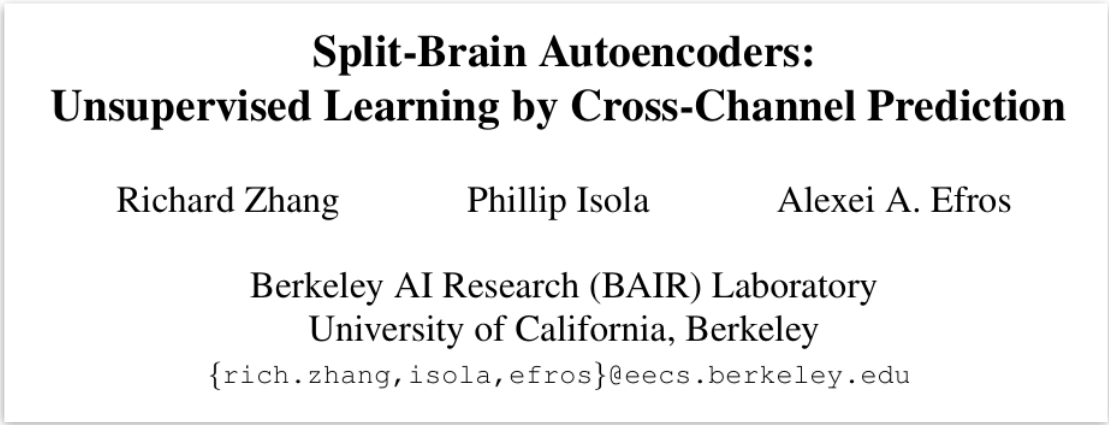


Colorful Image Colorization

Richard Zhang, Phillip Isola(✉), and Alexei A. Efros

University of California, Berkeley, USA
{rich.zhang,isola,efros}@eecs.berkeley.edu

ECCV 2016



Split-Brain Autoencoders:
Unsupervised Learning by Cross-Channel Prediction

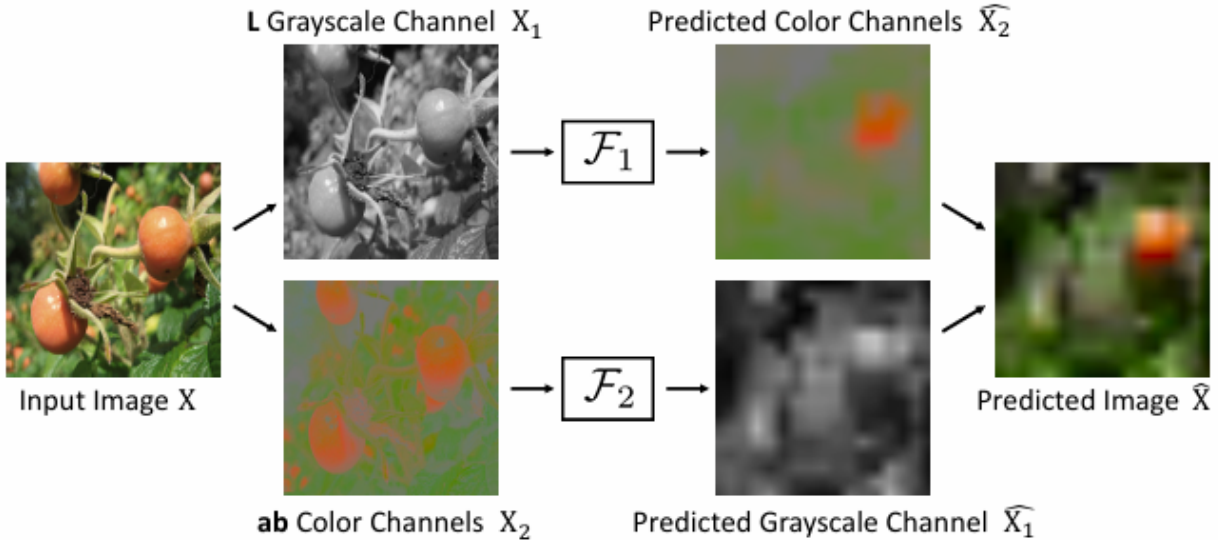
Richard Zhang Phillip Isola Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory
University of California, Berkeley
{rich.zhang,isola,efros}@eecs.berkeley.edu

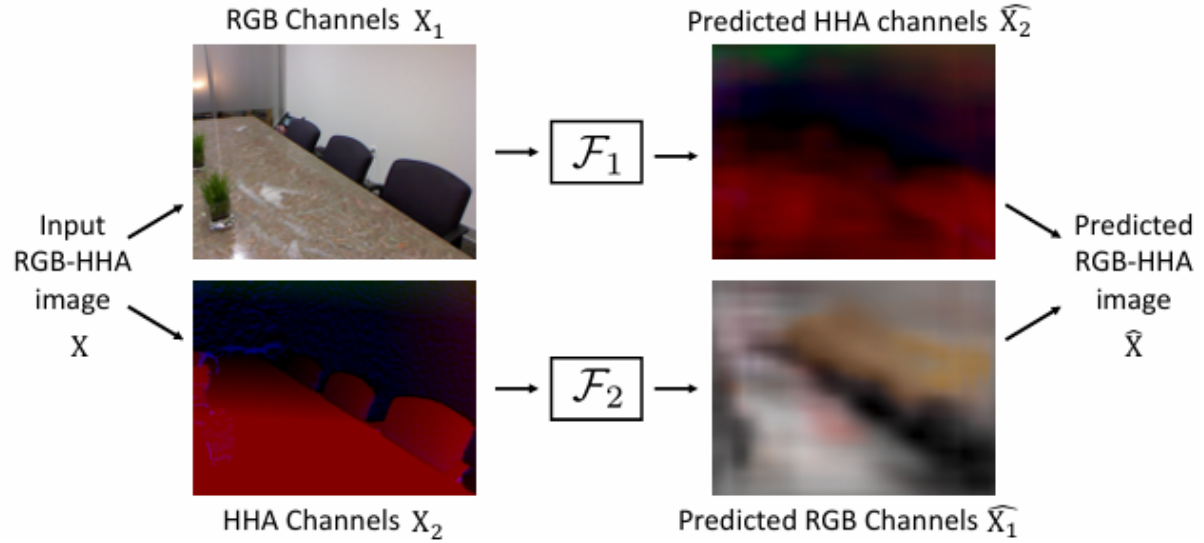
CVPR 2017

Pretext task: image coloring

- Learning features from colorization: Split-brain Autoencoder



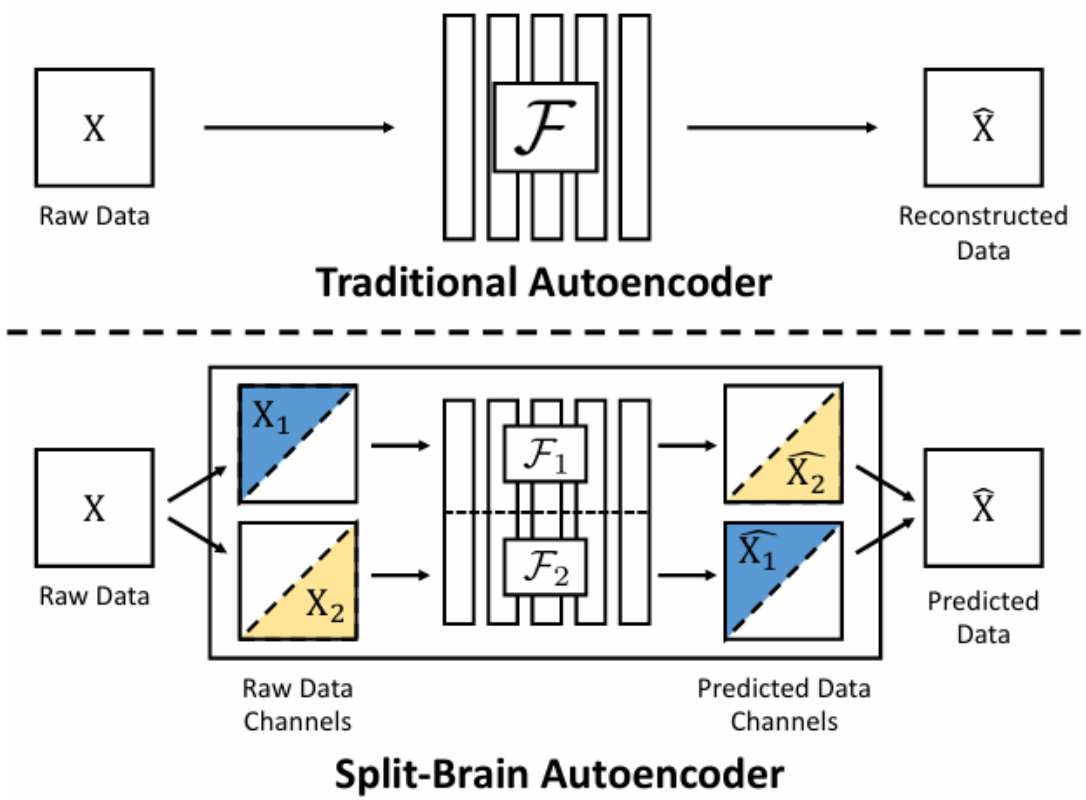
(a) Lab Images



(b) RGB-D Images

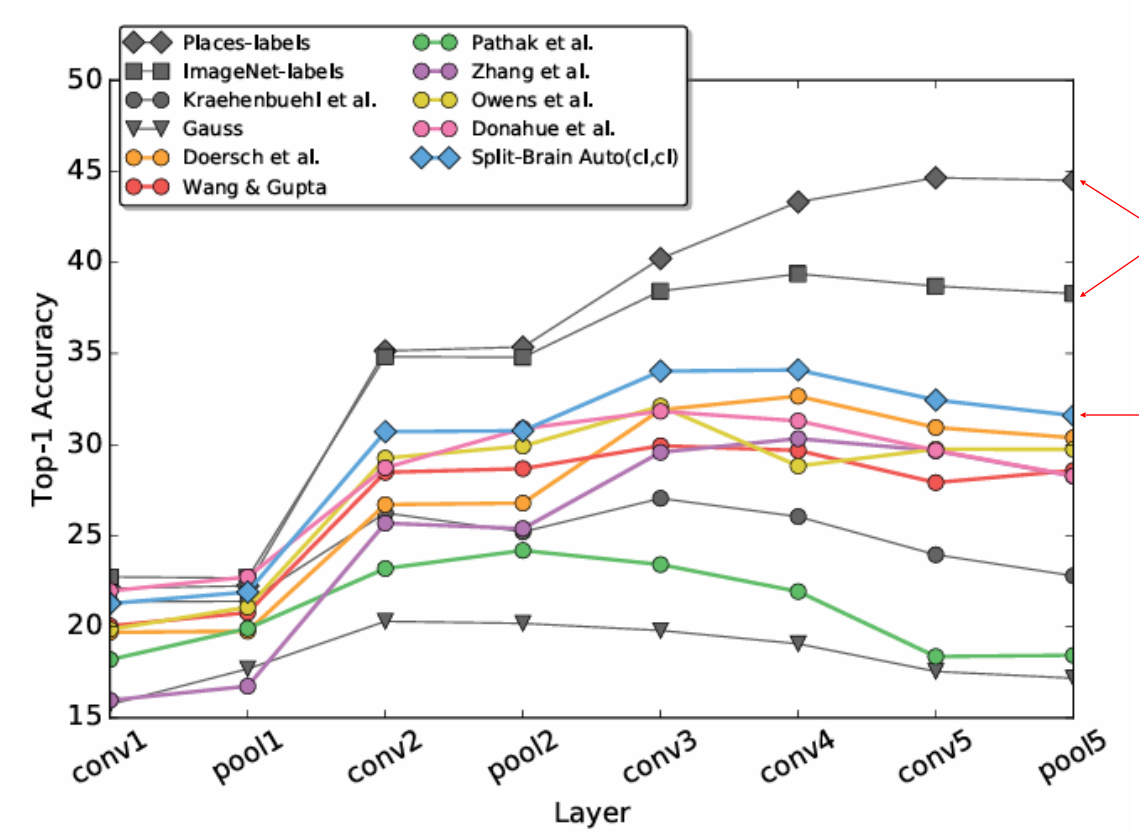
Pretext task: image coloring

- Split-brain Autoencoder Idea: cross-channel predictions



Pretext task: image coloring

- Transfer learned features to supervised learning



Supervised

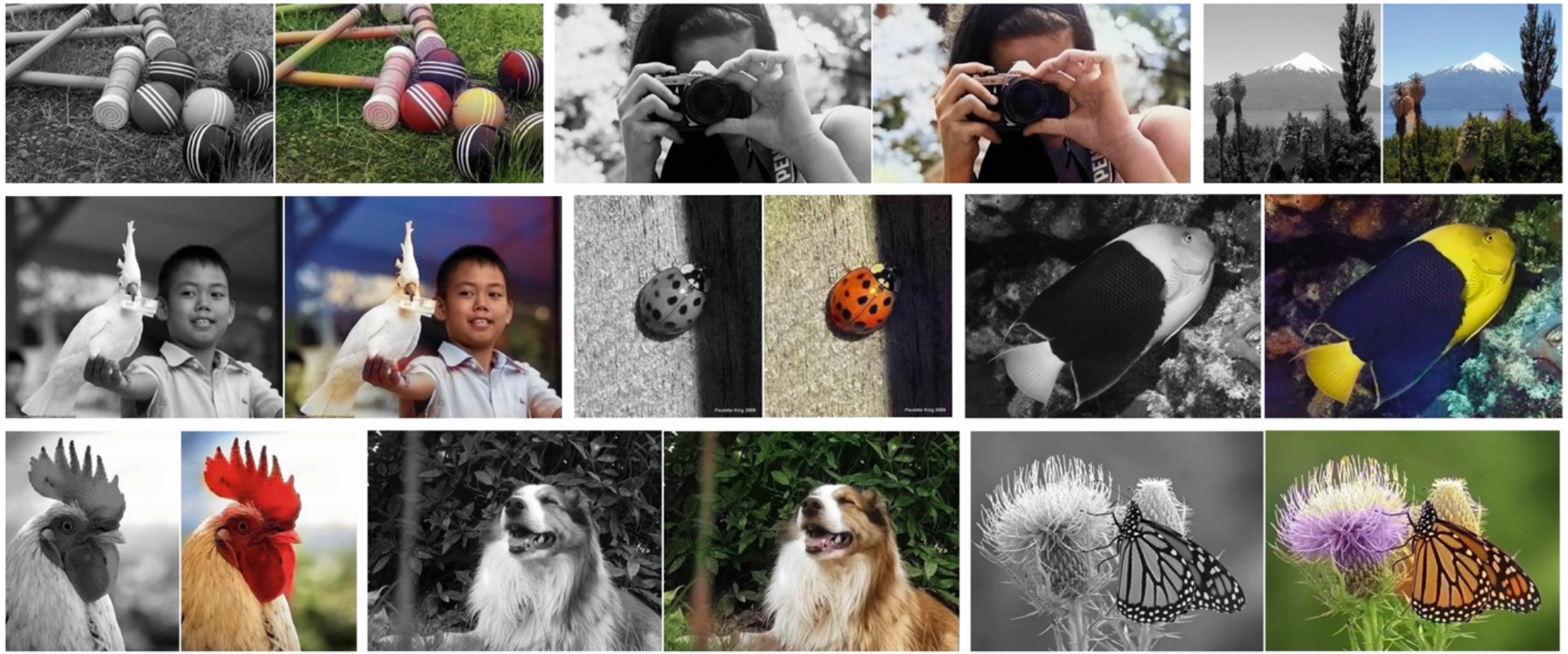
Split-brain Autoencoder

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

Use concatenated features from F1 and F2

Labeled data is from the Places (Zhou 2016).

Pretext task: image coloring



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

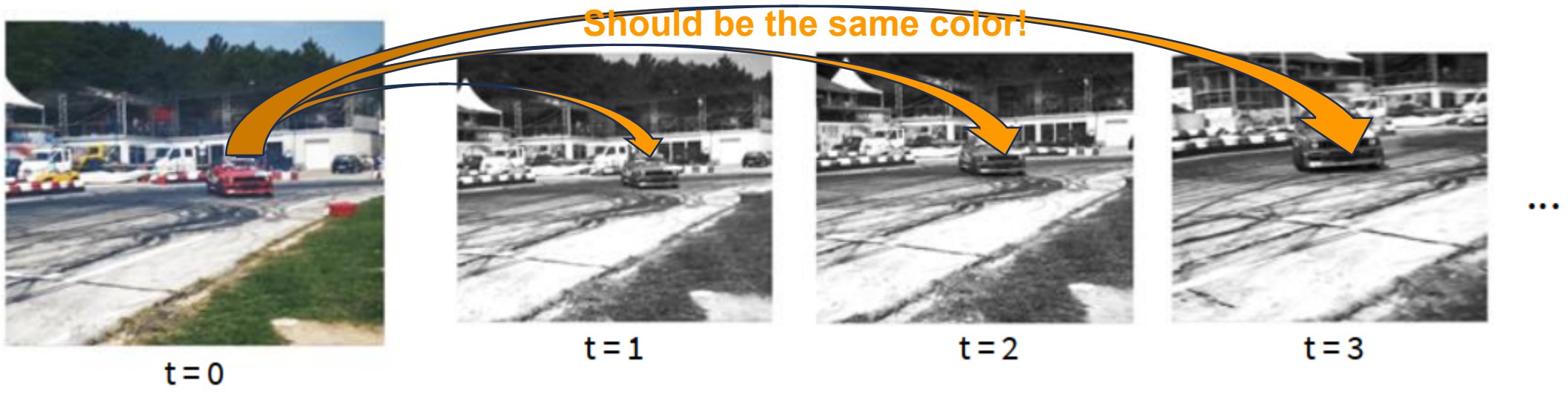
...

Pretext task: video coloring

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

reference frame

how should I color these frames?

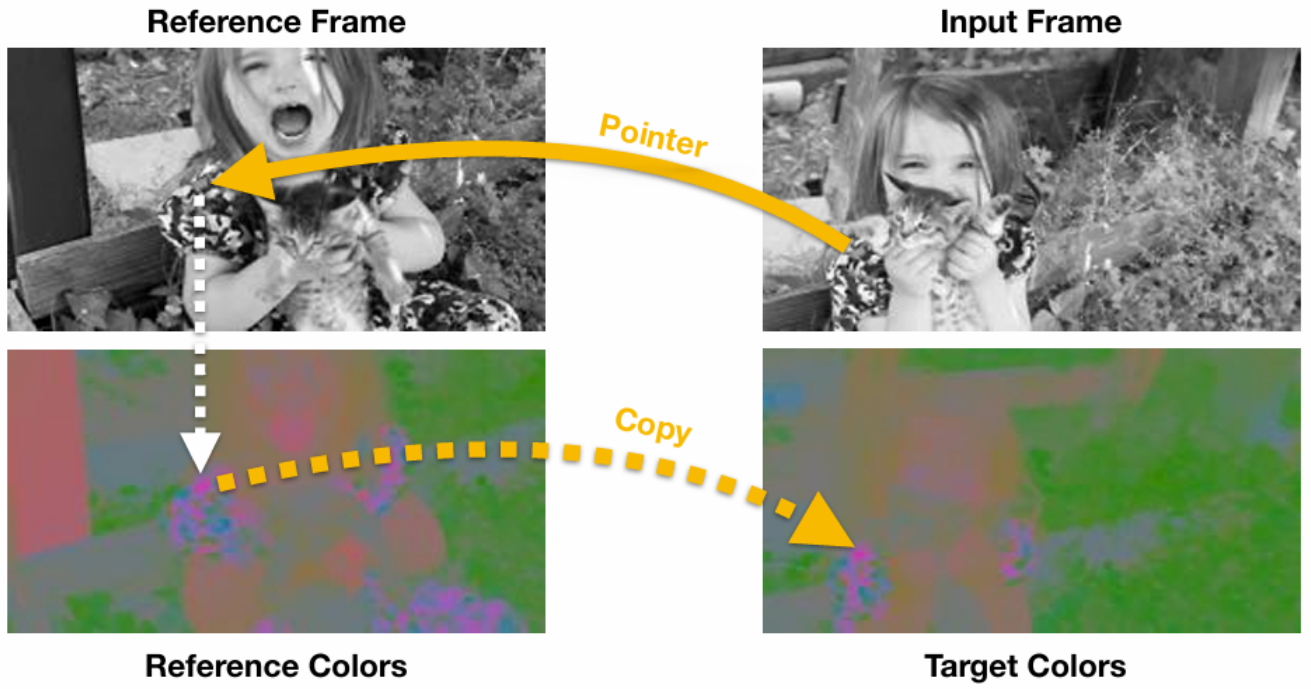


Pretext task: video coloring

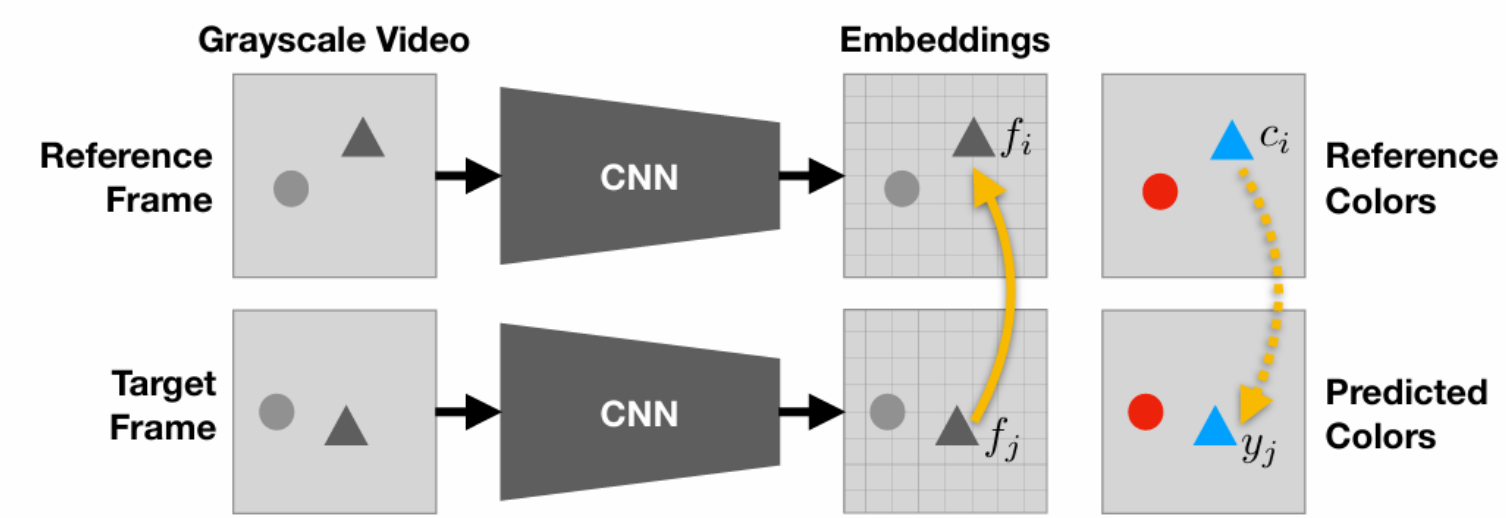
- Learning objective:

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

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



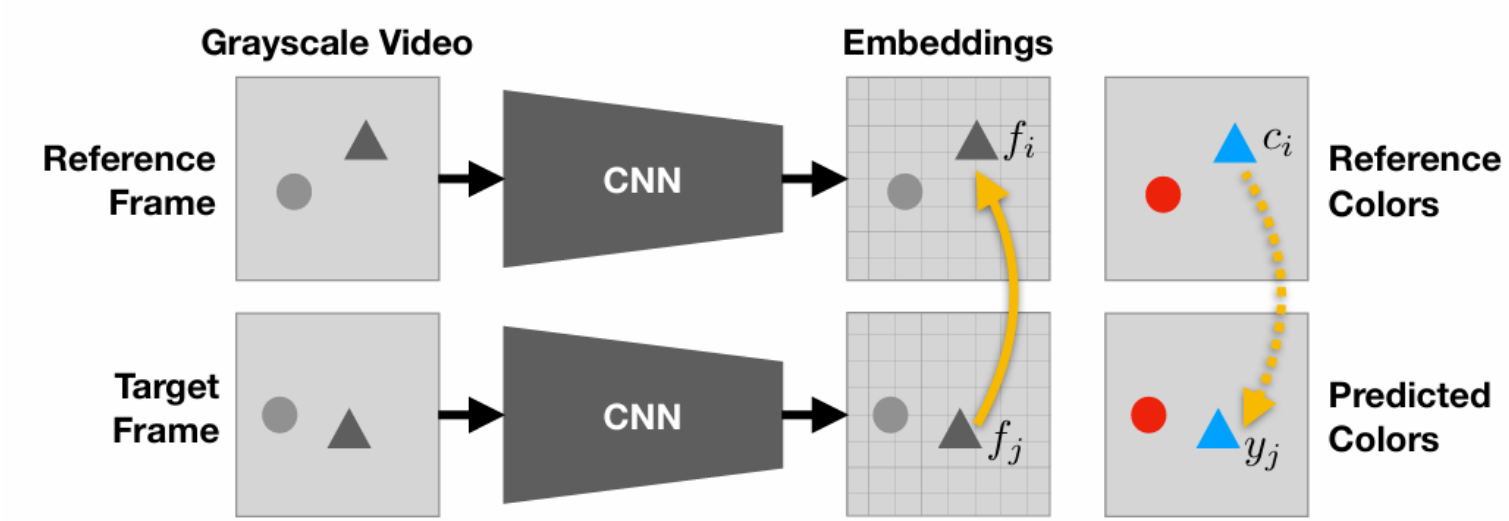
Pretext task: video coloring



attention map on the reference frame

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

Pretext task: video coloring

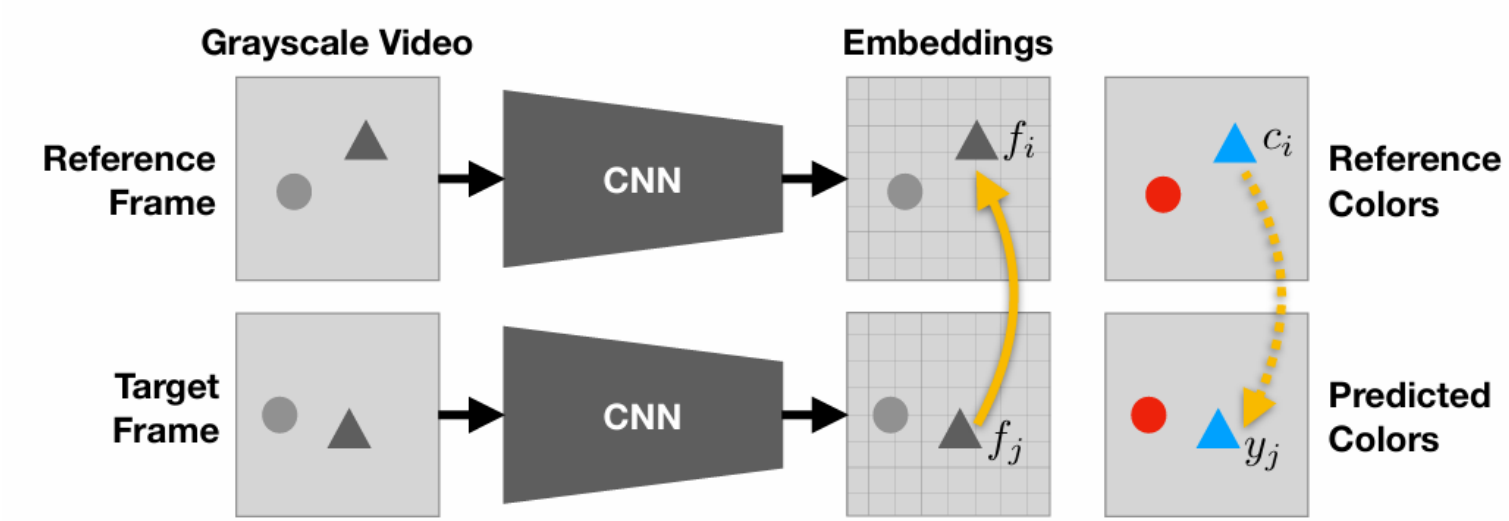


attention map on the reference frame

predicted color = weighted sum of the reference color

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

Pretext task: video coloring



attention map on the reference frame

predicted color = weighted sum of the reference color

loss between predicted color and ground truth color

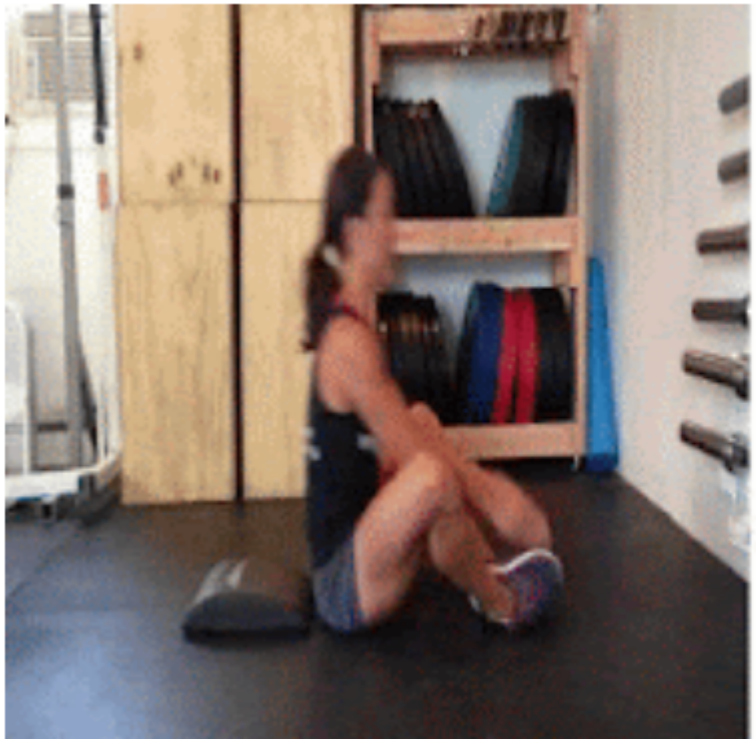
$$A_{ij} = \frac{\exp\left(f_i^T f_j\right)}{\sum_k \exp\left(f_k^T f_j\right)} \qquad y_j = \sum_i A_{ij} c_i \qquad \min_\theta \sum_j \mathcal{L}\left(y_j, c_j\right)$$

Pretext task: video coloring

reference frame

target frames (gray)

predicted color







Pretext task: video coloring

reference frame



target frames (gray)



predicted color



Pretext task: video coloring

- Tracking emerges from colorization
- Propagate segmentation masks using learned attention







Pretext task: video coloring

- Tracking emerges from colorization
- Propagate pose keypoints using learned attention







Pretext task: predict missing pixels (inpainting)

Context Encoders: Feature Learning by Inpainting

Deepak Pathak Philipp Krähenbühl Jeff Donahue Trevor Darrell Alexei A. Efros
University of California, Berkeley
{pathak,philkr,jdonahue,trevor,efros}@cs.berkeley.edu

Abstract

We present an unsupervised visual feature learning algorithm driven by context-based pixel prediction. By analogy with auto-encoders, we propose Context Encoders – a convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. In order to succeed at this task, context encoders need to both understand the content of the entire image, as well as produce a plausible hypothesis for the missing part(s). When training context encoders, we have experimented with both a standard pixel-wise reconstruction loss, as well as a reconstruction plus an adversarial loss. The latter produces much sharper results because it can better handle multiple modes in the output. We found that a context encoder learns a representation that captures not just appearance but also the semantics of visual structures. We quantitatively demonstrate the effectiveness of our learned features for CNN pre-training on classification, detection, and segmentation tasks. Furthermore, context encoders can be used for semantic inpainting tasks, either stand-alone or as initialization for non-parametric methods.

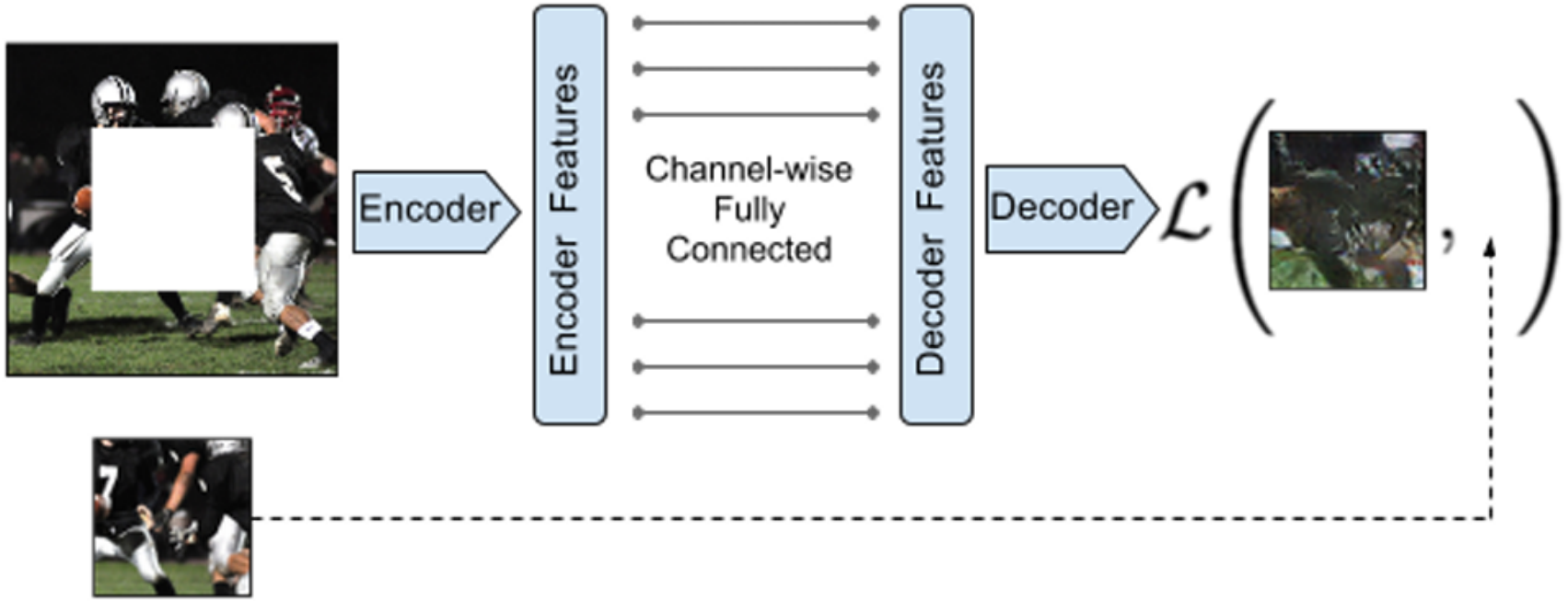


(a) Input context (b) Human artist

(c) Context Encoder ($L2$ loss) (d) Context Encoder ($L2$ + Adversarial loss)

Pretext task: predict missing pixels (inpainting)

- Randomly mask some regions from the image.
- Train a network to recover the missing regions.



Pretext task: predict missing pixels (inpainting)

- Loss = reconstruction + adversarial learning
- Adversarial loss between “real” images and inpainted images

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

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

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

Element wise multiplication

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

Encoder

Pretext task: predict missing pixels (inpainting)



Input (context)



reconstruction



adversarial



recon + adv

Pretext task: predict missing pixels (inpainting)

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

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%

MAE: Masked Autoencoders Are Scalable Vision Learners

Masked Autoencoders Are Scalable Vision Learners

Kaiming He*,† Xinlei Chen* Saining Xie Yanghao Li Piotr Dollár Ross Girshick

*equal technical contribution †project lead

Facebook AI Research (FAIR)

Abstract

This paper shows that masked autoencoders (MAE) are scalable self-supervised learners for computer vision. Our MAE approach is simple: we mask random patches of the input image and reconstruct the missing pixels. It is based on two core designs. First, we develop an asymmetric *encoder-decoder architecture, with an encoder that operates only on the visible subset of patches (without mask tokens), along with a lightweight decoder that reconstructs the original image from the latent representation and mask tokens. Second, we find that masking a high proportion of the input image,* e.g., *75%, yields a nontrivial and meaningful self-supervisory task. Coupling these two designs enables us to train large models efficiently and effectively: we accelerate training (by 3× or more) and improve accuracy. Our scalable approach allows for learning high-capacity models that generalize well:* e.g., *a vanilla ViT-Huge model achieves the best accuracy (87.8%) among*

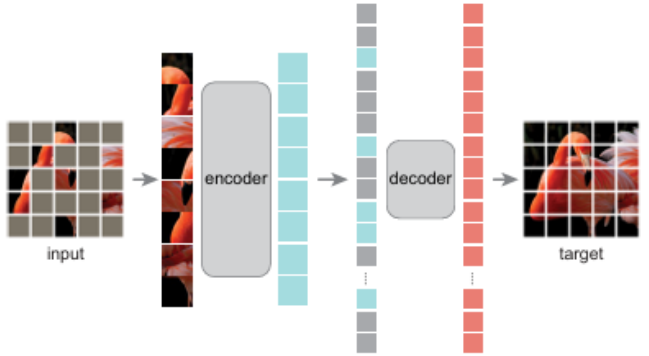
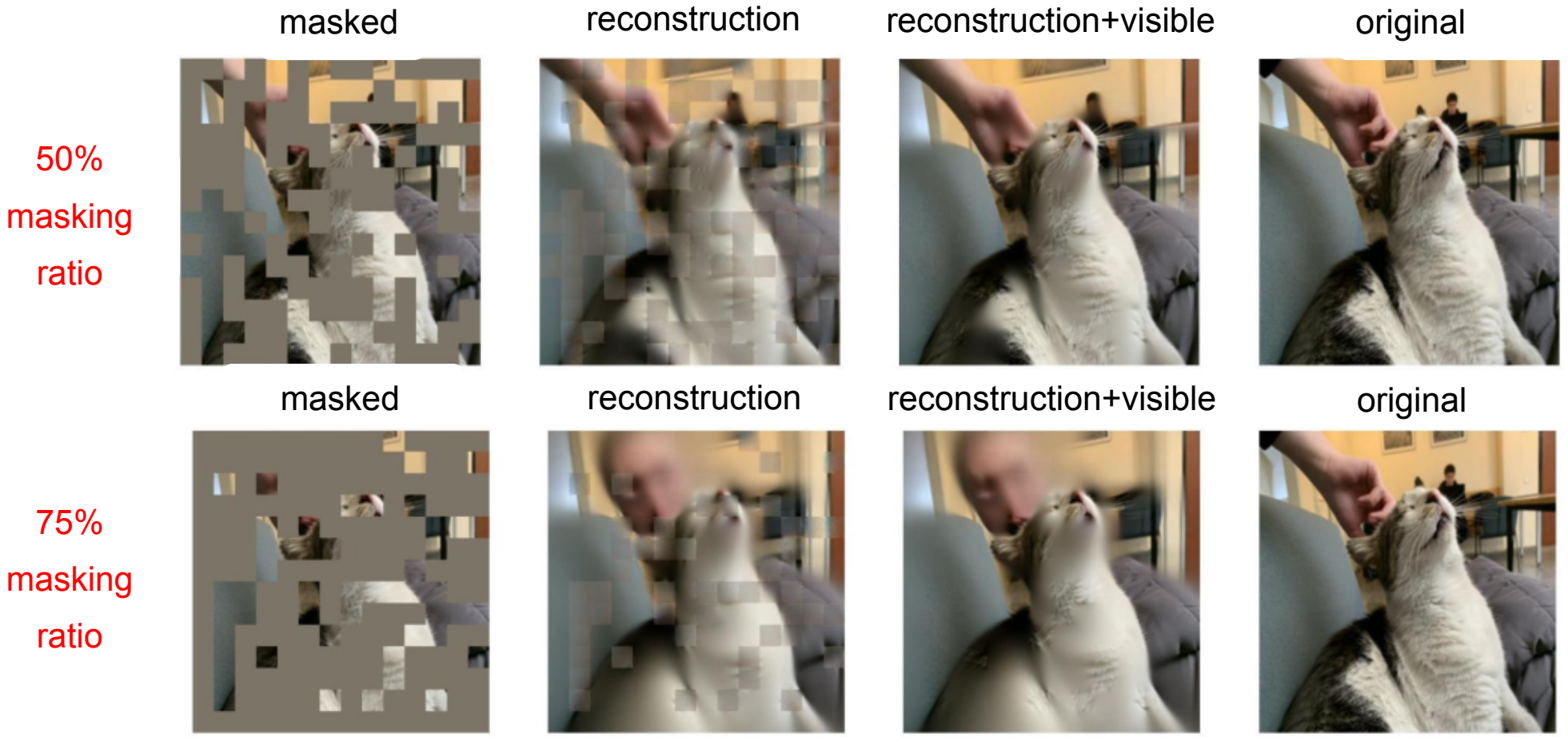


Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

Reconstruction with a larger masked portion



How does MAE work?

- **Masking Methods:**

✓ Similar to the original ViT, divide the input into non-overlapping patches.

✓ Uniformly sample a very large proportion (75%) of these patches and mask them.

✓ Masking a high ratio makes predicting the task challenging and meaningful.



input

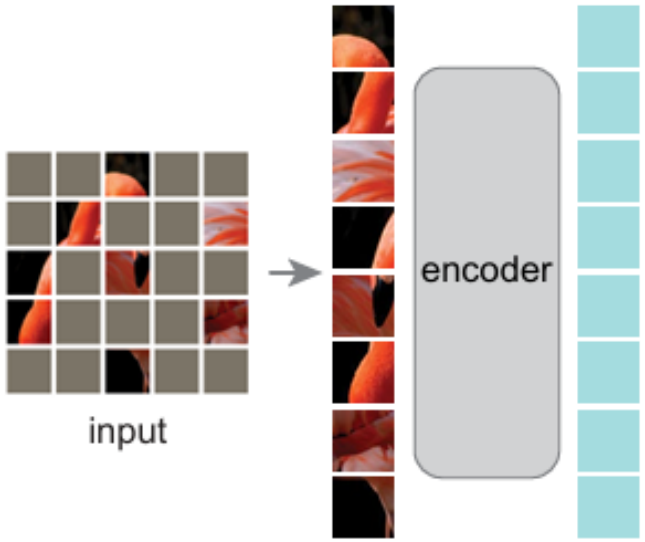
How does MAE work?

- **MAE Encoder**

✓ The encoder only operates on unmasked patches (25%)

✓ Uses transformer blocks, Embeds the patches by linear projection and add positional embeddings

✓ Since the input patches are a small part of the input, the encoder is chosen to be very large. (encoder has over 9 times computations per token vs decoder)



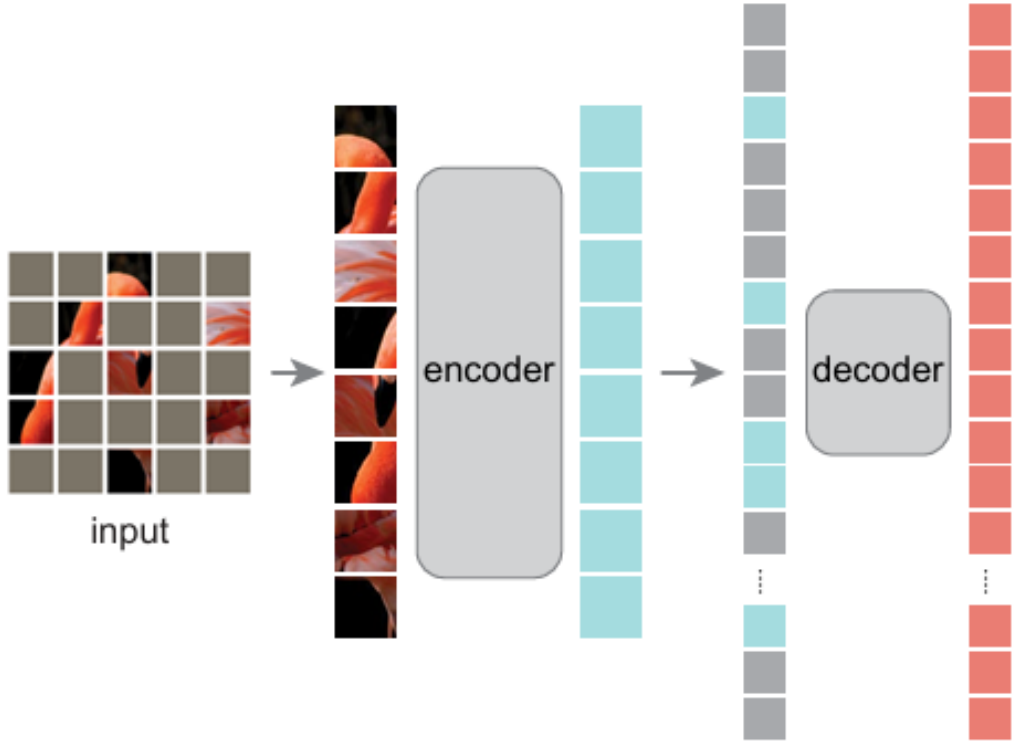
How does MAE work?

- **MAE Decoder**

✓ Merges the encoder outputs with the shared mask tokens in previously masked places, adding positional encodings.

✓ Uses transformer blocks, followed by a linear projection for finalizing pixel reconstruction.

✓ Is solely responsible for reconstruction, meaning it is not used posttraining. Hence, it is independent of the encoder design, making it flexible. This is an asymmetrical autoencoder design.

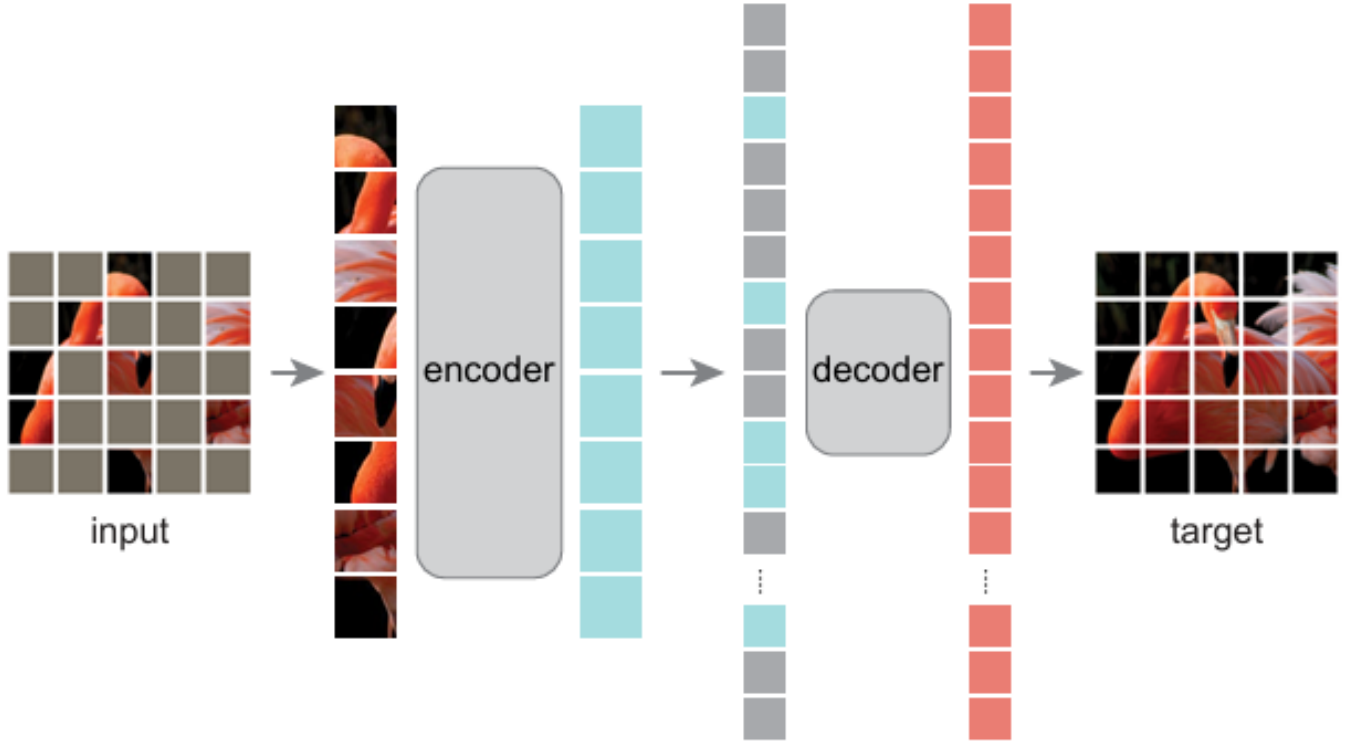


How does MAE work?

- **Reconstruction**

✓ The MSE (mean squared error loss) in the pixel space between the input image and the reconstructed image is adopted.

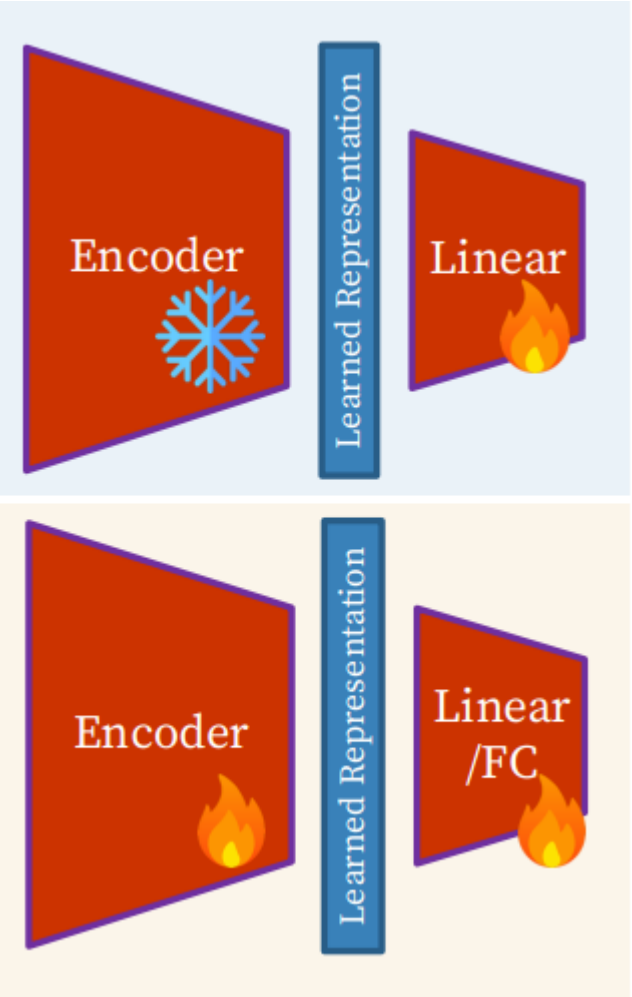
✓ Loss is only computed for masked patches



Linear Probing vs Full Fine-tuning

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

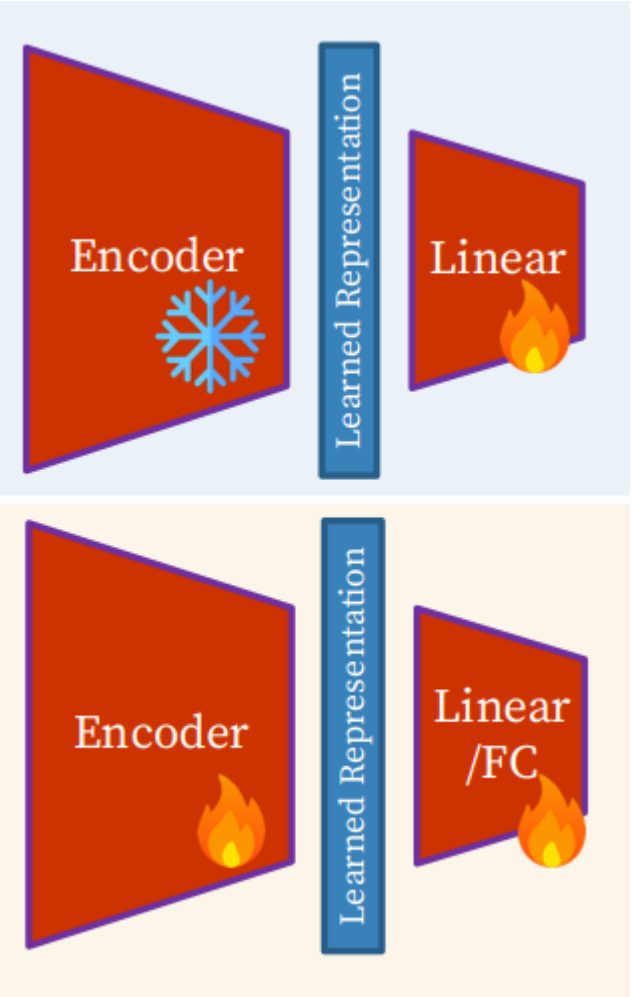
- In fine-tuning, pre-trained model is further trained (not fixed), and one or more layers, possibly with non-linearities are added.



Linear Probing vs Full Fine-tuning

- **Linear probing**: provides a measure of representation quality of a pre-training in restricted conditions

- **Fine-tuning**: exploits models near-true potential to adopt for new tasks



Ablation Studies

- So many modeling/hyperparameter choices:

✓ Masking ratio

✓ Decoder depth

✓ Decoder width

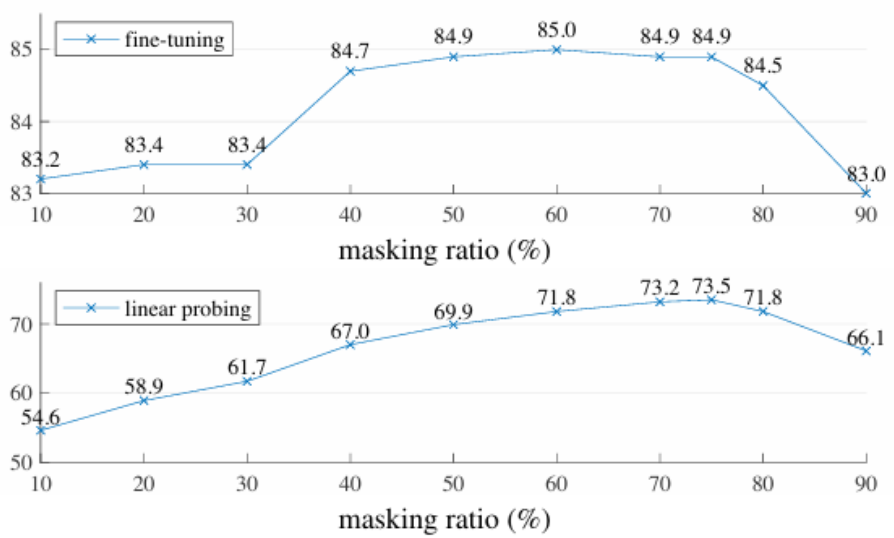
✓ Mask token (used or not in encoder)

✓ Reconstruction target

✓ Data augmentation

✓ Mask sampling method

✓ Training schedule



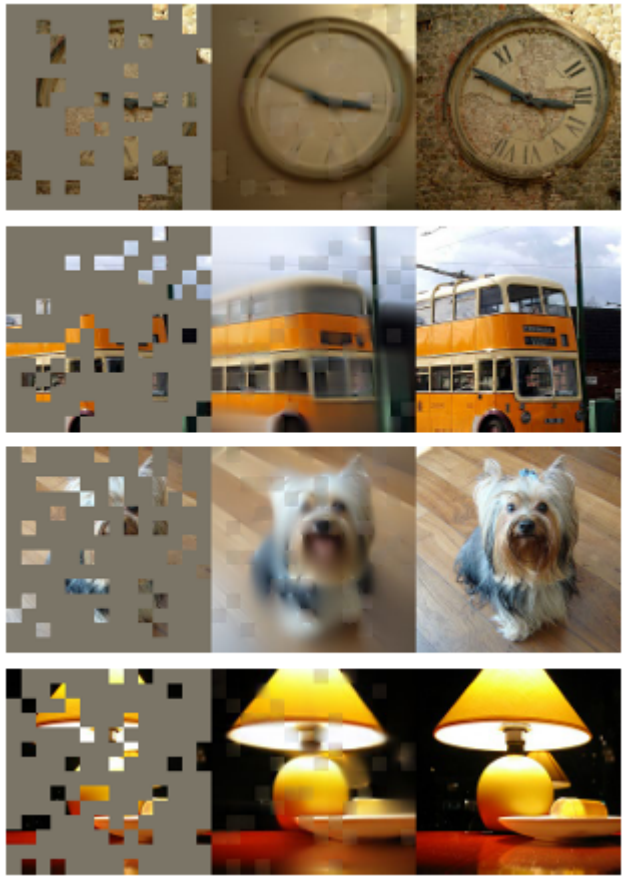
case	ratio	ft	lin
random	75	**84.9**	**73.5**
block	50	83.9	72.3
block	75	82.8	63.9
grid	75	84.0	66.0



mask sampling strategies

Traditional Self-supervised Learning

Comparisons



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

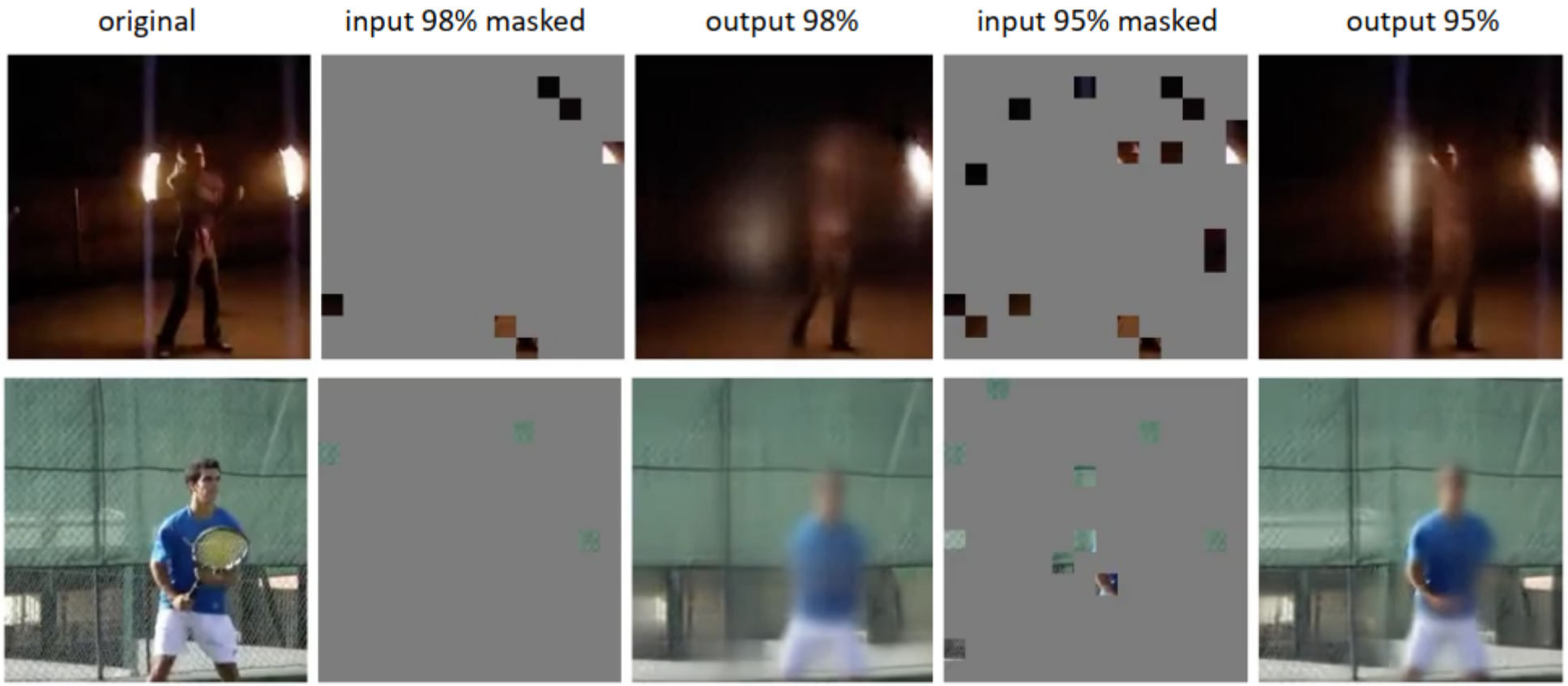
Decoder depth

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

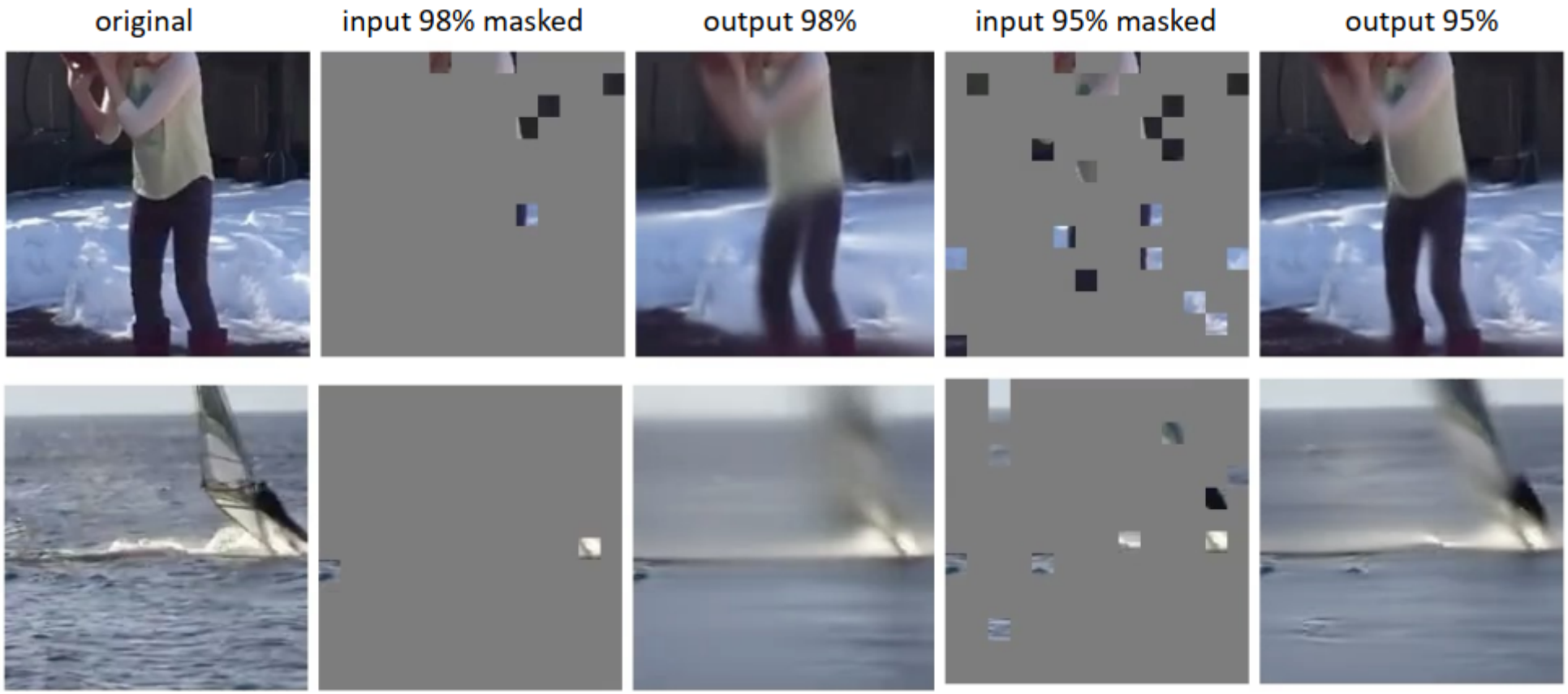
Decoder width

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

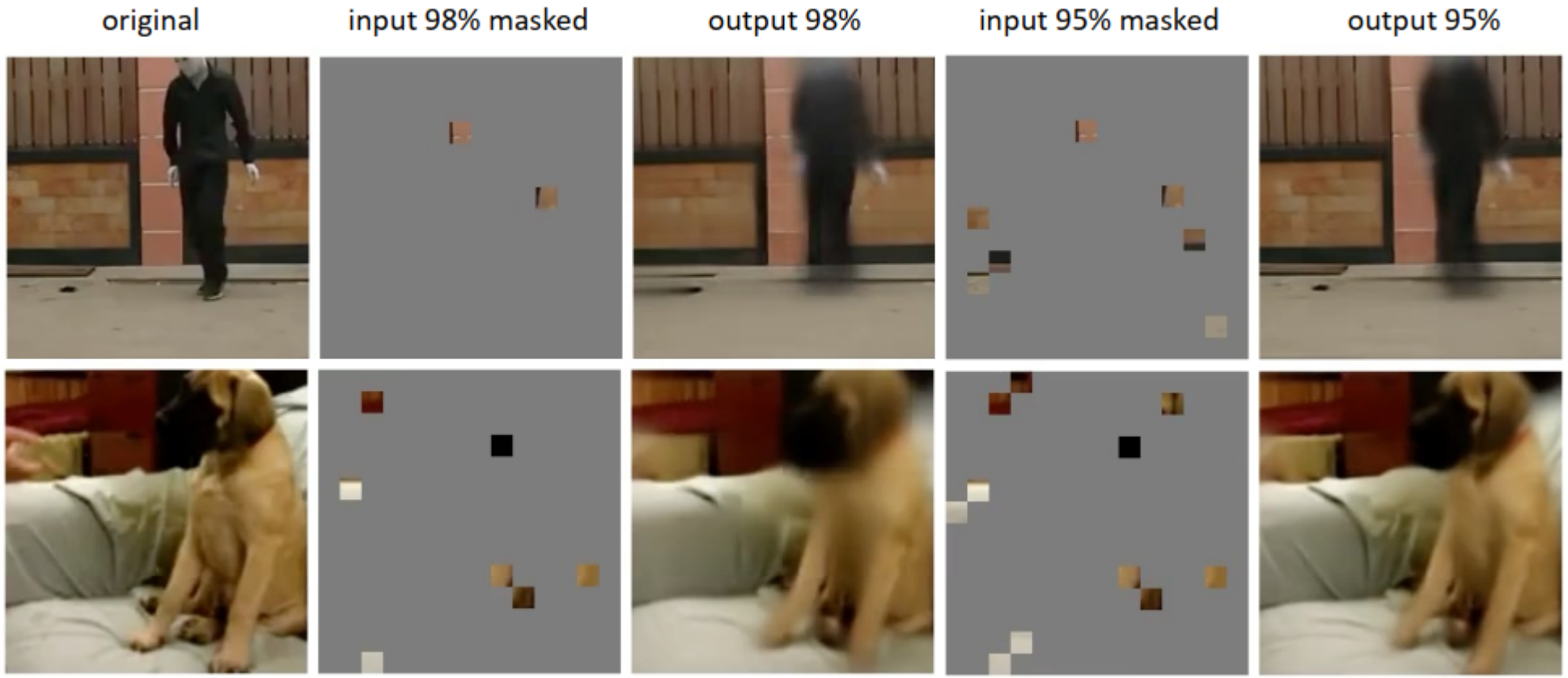
MAE visualizations



MAE visualizations



MAE visualizations



Summary

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

Traditional Self-supervised Learning

Summary

- Problems:
  - ✓ Coming up with individual pretext tasks is tedious.
  - ✓ The learned representations may not be general.

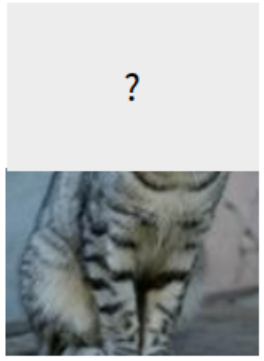
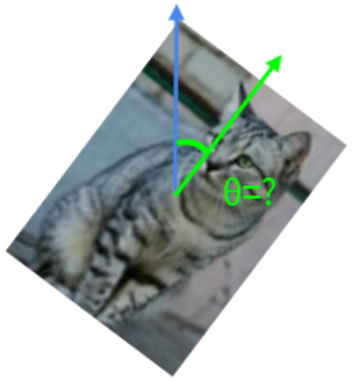


image completion



rotation prediction



“jigsaw puzzle”



colorization

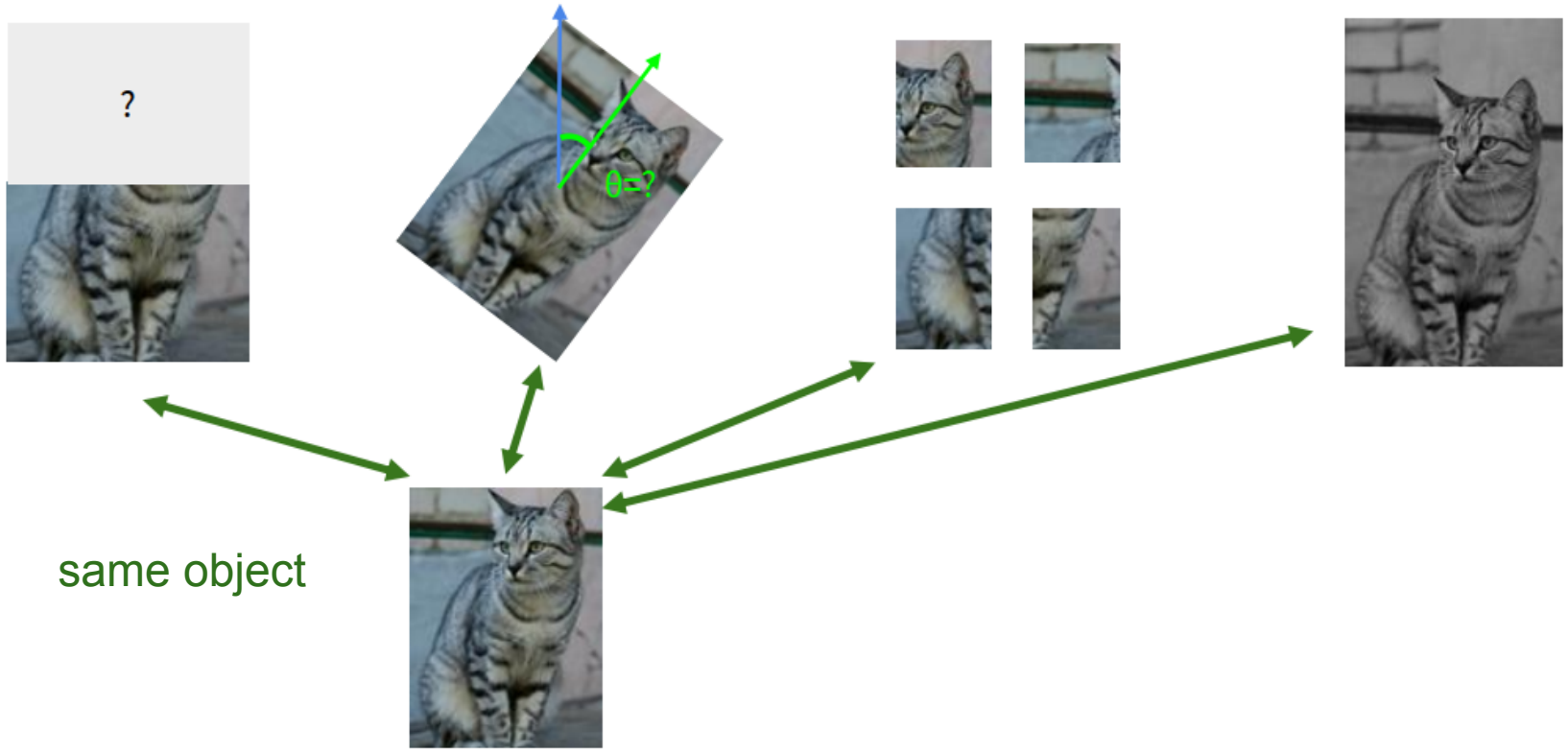
上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

Contents

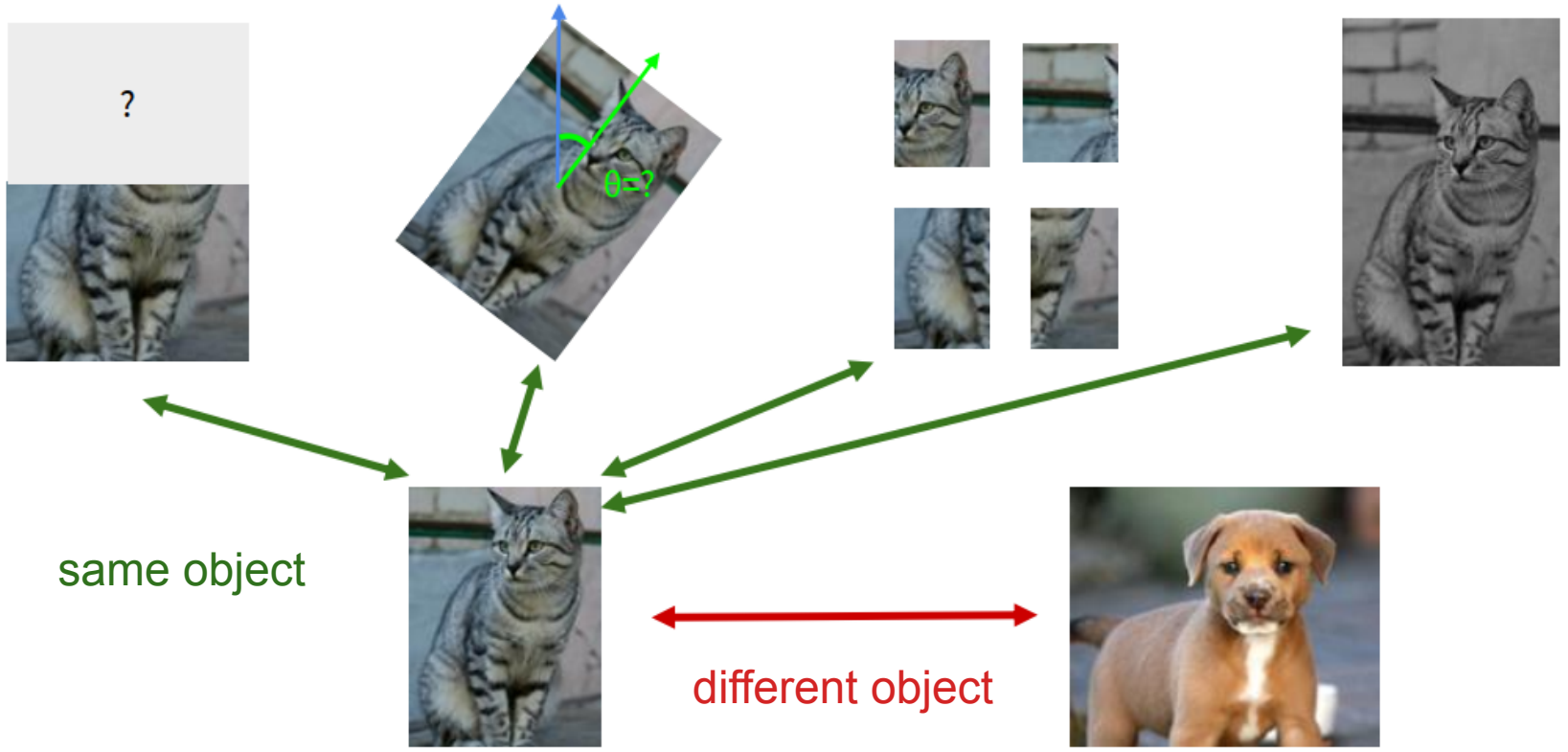
- Introduction
- Traditional SSL
- **Modern SSL**
  - **Contrastive learning: SimCLR, MOCO**
  - **Self-Distillation without labels: DINO**



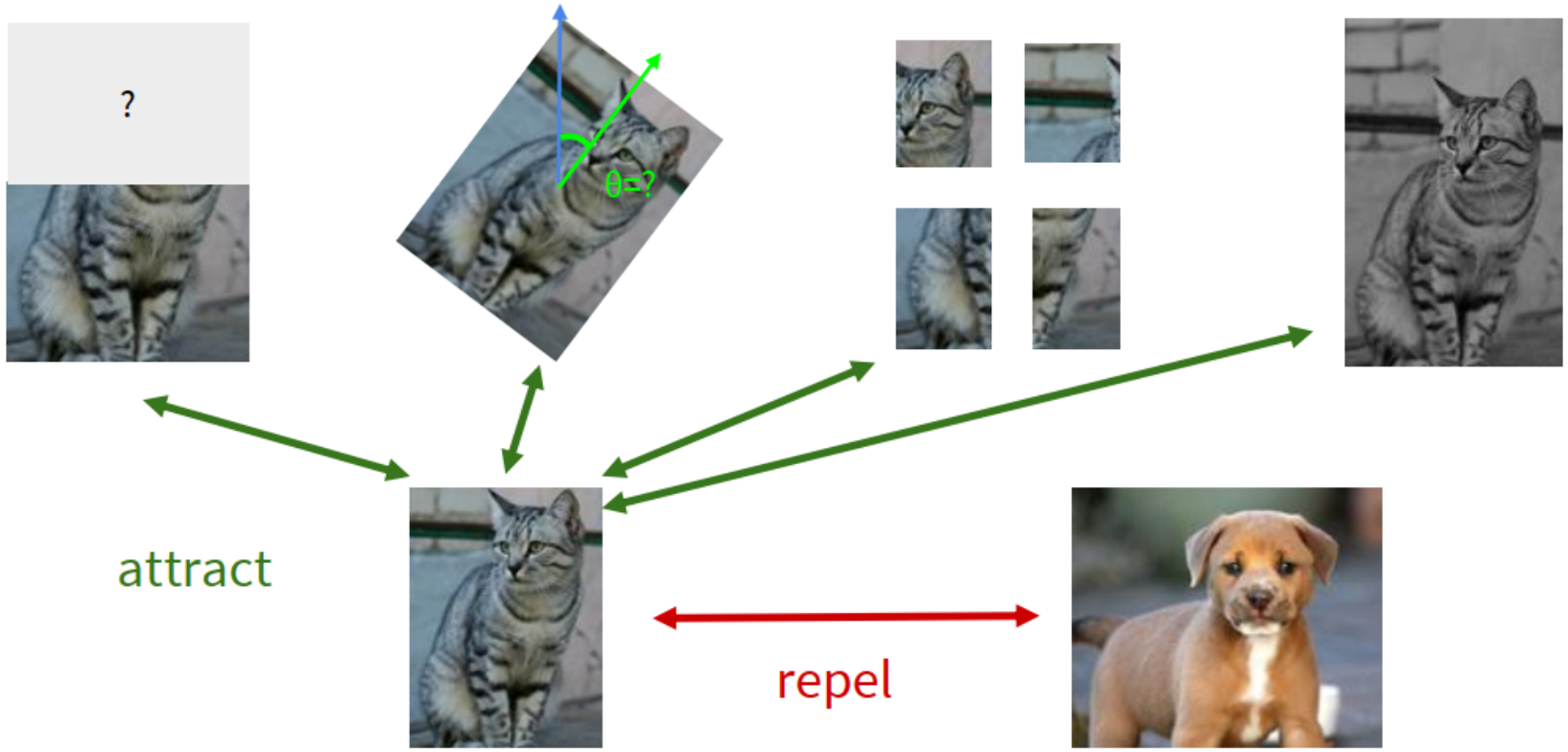
A more general pretext task?



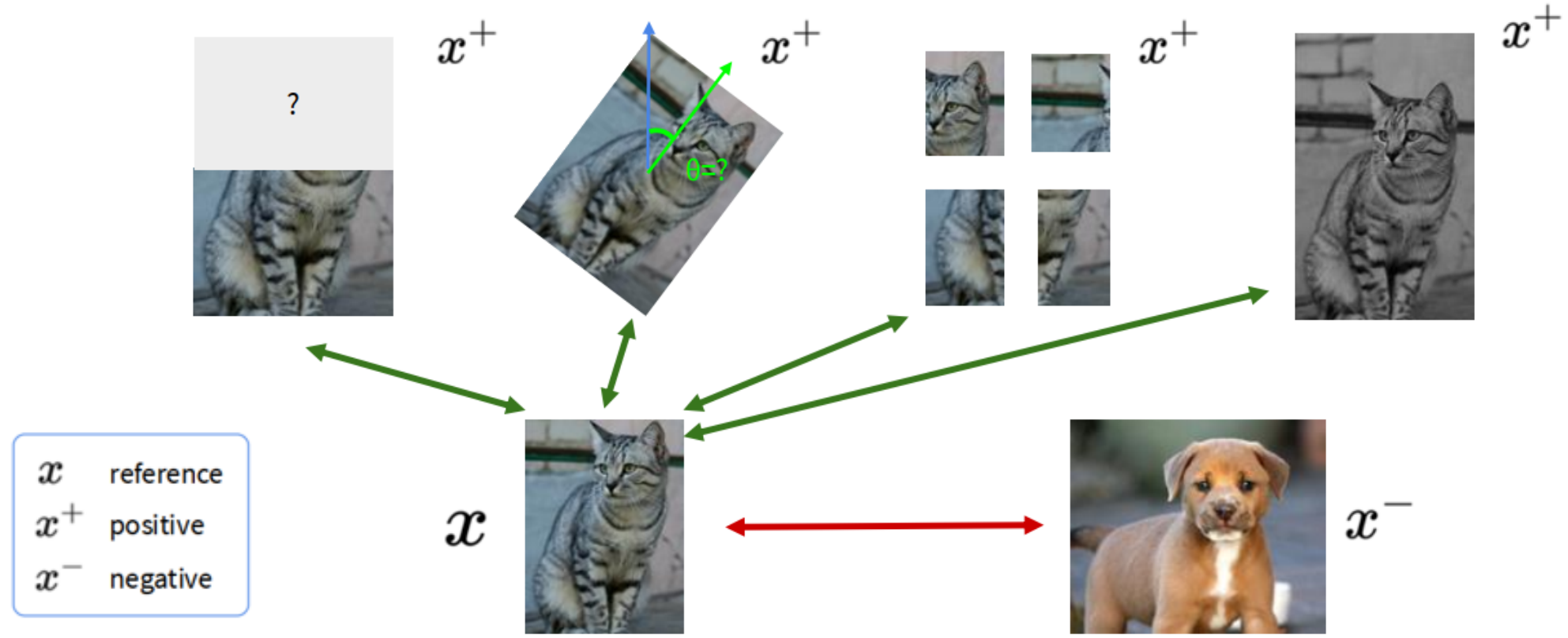
A more general pretext task?



Contrastive Representation Learning



Contrastive Representation Learning



A formulation of contrastive learning

- What we want:

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

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

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

A formulation of contrastive learning

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

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



x



x^+



x



x_1^-



x_2^-



x_3^-

...

A formulation of contrastive learning

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

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

score for the positive pair

score for the N-1 negative pairs

- This seems familiar …

A formulation of contrastive learning

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

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

score for the positive pair

score for the N-1 negative pairs

- This seems familiar …
- Cross entropy loss for a N-way softmax classifier!
- I.e., learn to find the positive sample from the N samples

A formulation of contrastive learning

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

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

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

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

- The larger the negative sample size (N), the tighter the bound

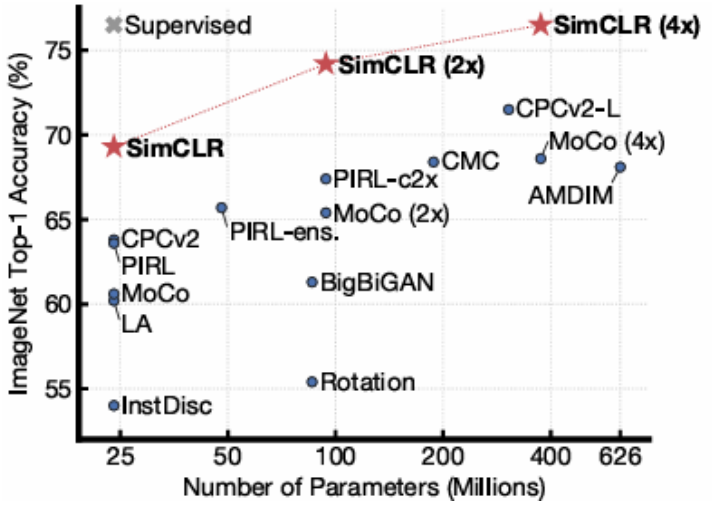
SimCLR: A Simple Framework for Contrastive Learning

A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen [1] Simon Kornblith [1] Mohammad Norouzi [1] Geoffrey Hinton [1]

Abstract

This paper presents *SimCLR*: a simple framework for contrastive learning of visual representations. We simplify recently proposed contrastive self-supervised learning algorithms without requiring specialized architectures or a memory bank. In order to understand what enables the contrastive prediction tasks to learn useful representations, we systematically study the major components of our framework. We show that (1) composition of data augmentations plays a critical role in defining effective predictive tasks, (2) introducing a learnable nonlinear transformation between the repre-



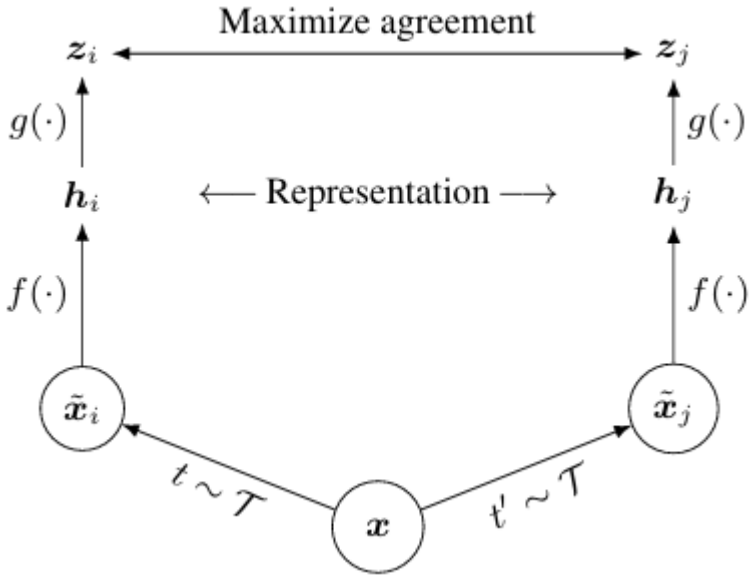
SimCLR: A Simple Framework for Contrastive Learning

- Cosine similarity as the score function:

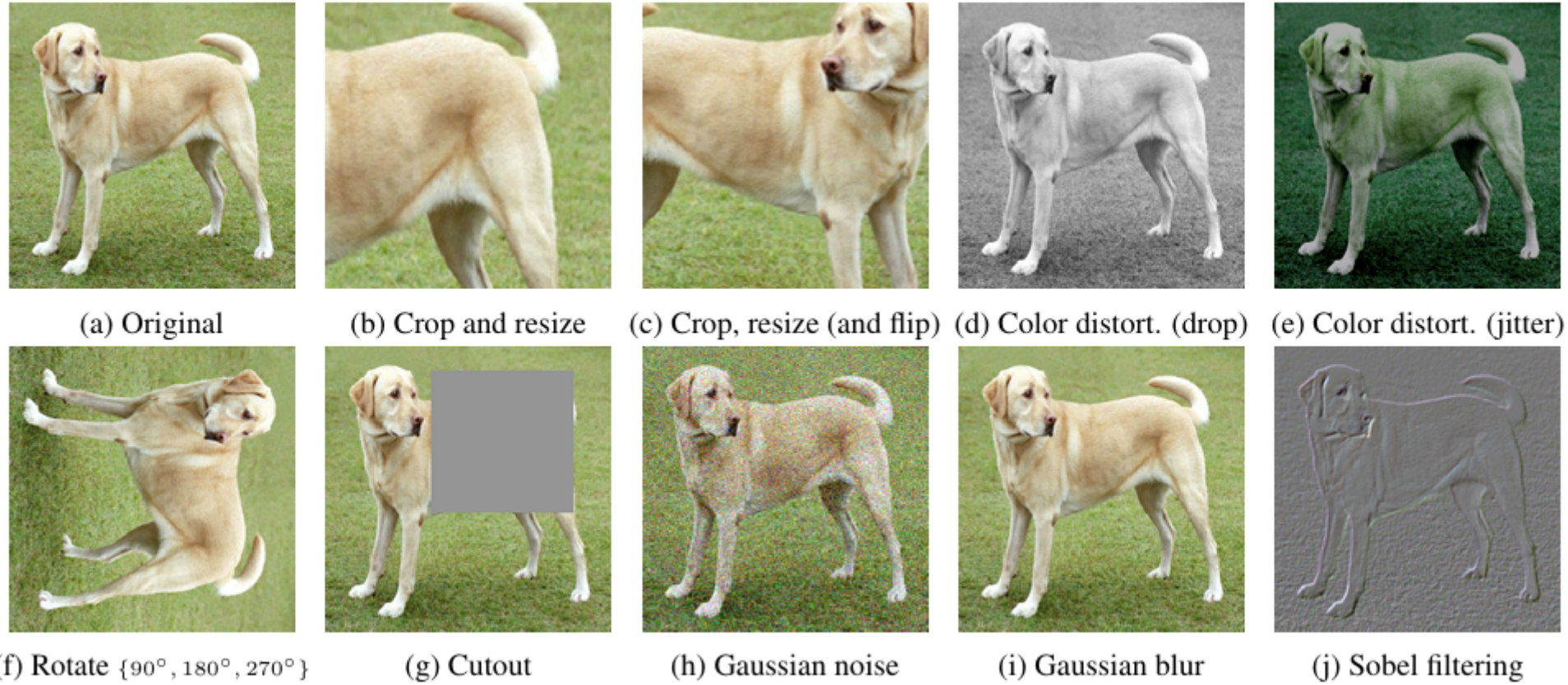
$$s(u, v) = \frac{u^T v}{||u||||v||}$$

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

- Generate positive samples through data augmentation: random cropping, random color distortion, and random blur.



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

SimCLR

Generate a positive pair by sampling data augmentation functions

Algorithm 1 SimCLR's main learning algorithm.

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

SimCLR

Generate a positive pair by sampling data augmentation functions

Algorithm 1 SimCLR's main learning algorithm.

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

InfoNCE loss:

Use all non-positive samples in the batch as x-

SimCLR

Generate a positive pair by sampling data augmentation functions

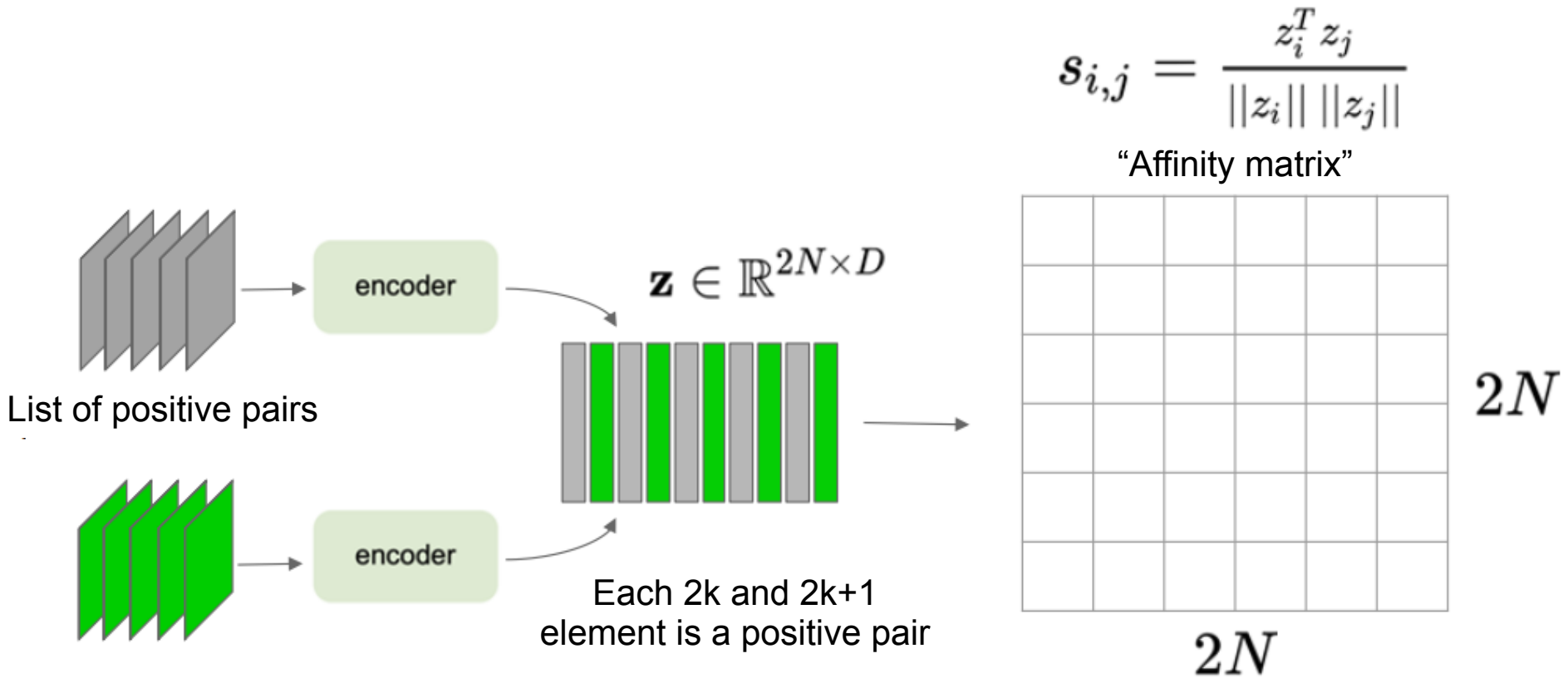
Algorithm 1 SimCLR's main learning algorithm.

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

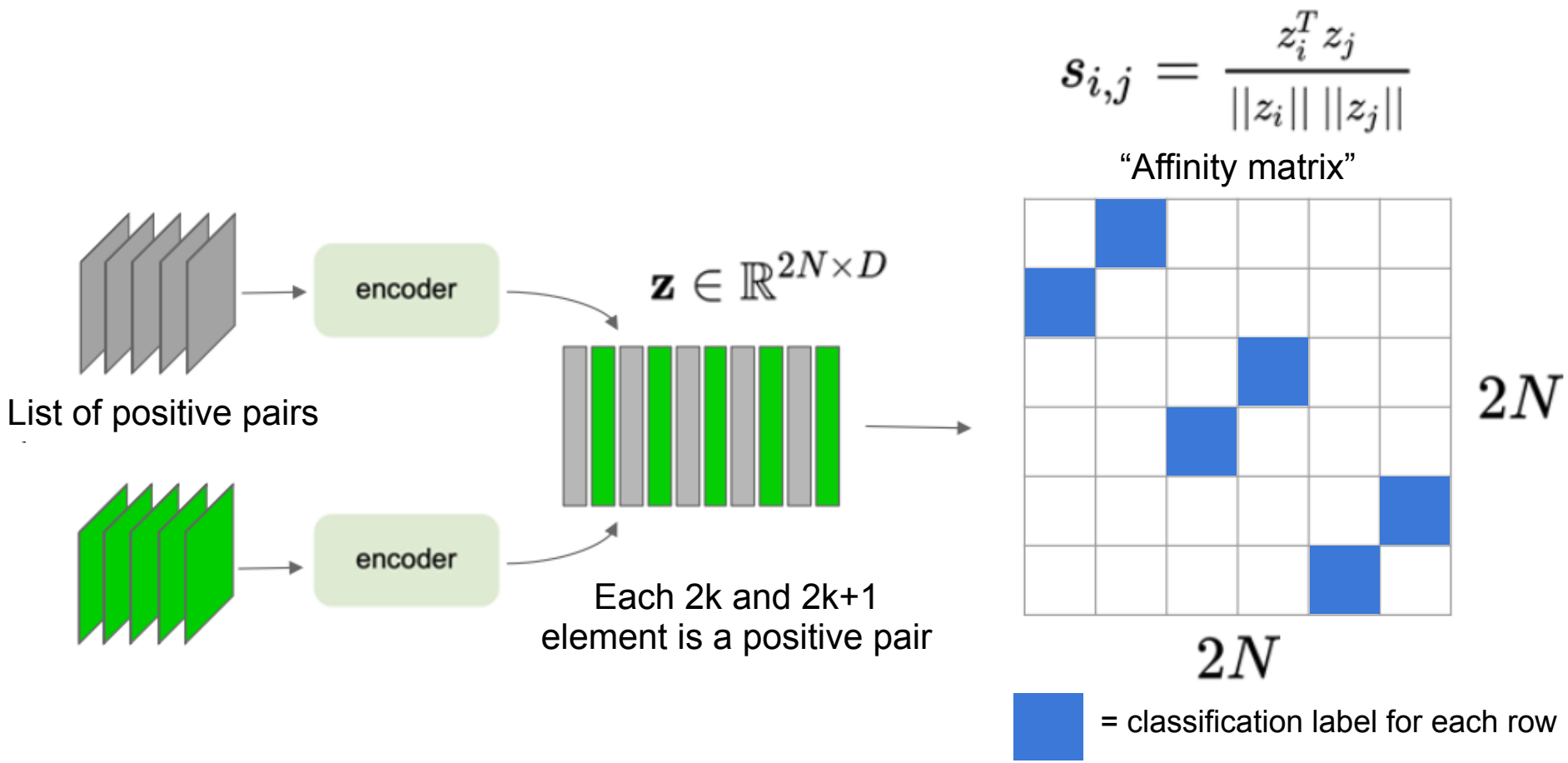
Iterate through and use each of the 2N sample as reference, compute average loss

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

SimCLR: mini-batch training

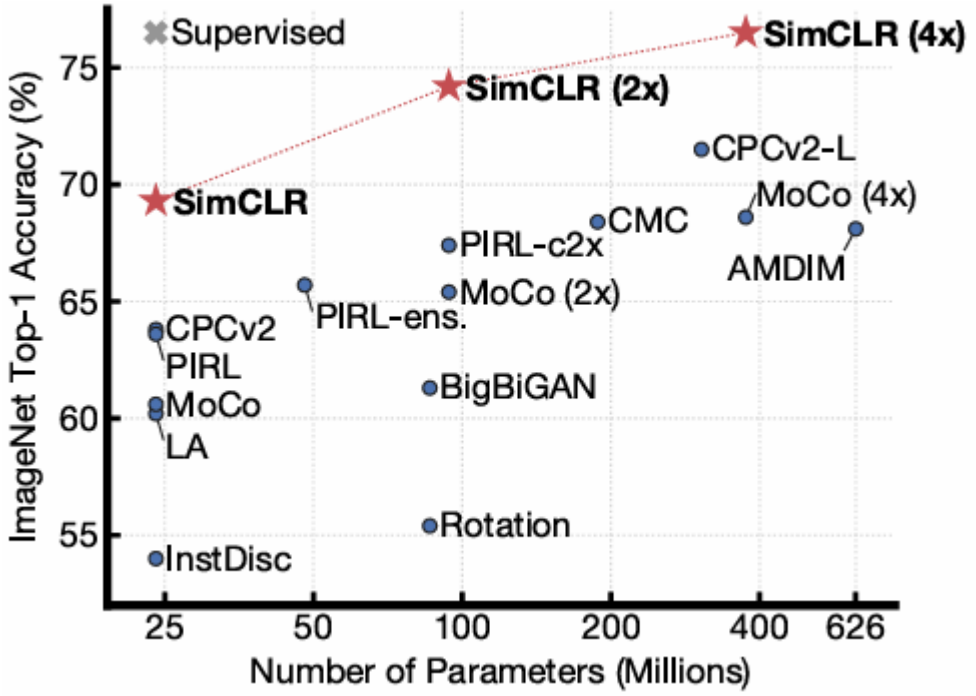


SimCLR: mini-batch training



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.



Modern Self-supervised Learning

Semi-supervised learning on SimCLR features

- Train feature encoder on ImageNet (entire training set) using SimCLR.
- Finetune the encoder with 1% / 10% of labeled data on ImageNet.

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

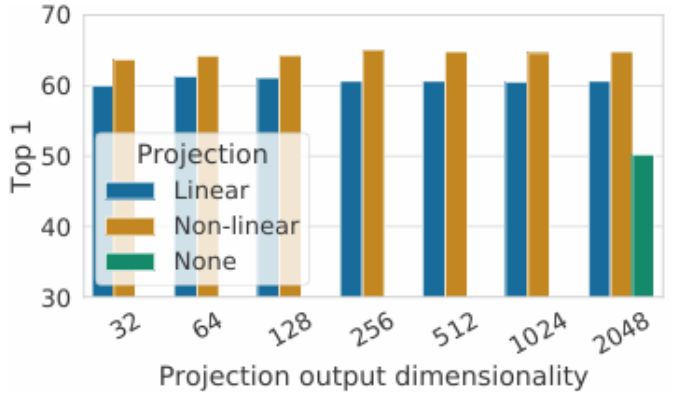
SimCLR design choices: projection head

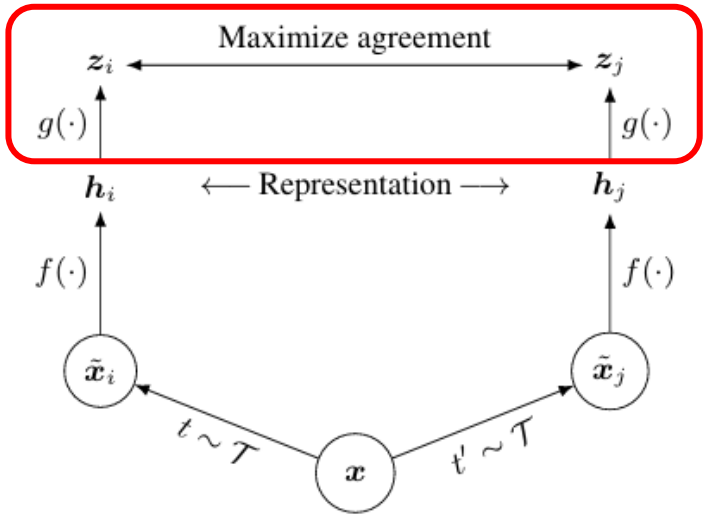
- Linear / non-linear projection heads improve representation learning.
- A possible explanation:

✓ Contrastive learning objective may discard useful information for downstream tasks.

✓ Representation space z is trained to be invariant to data transformation.

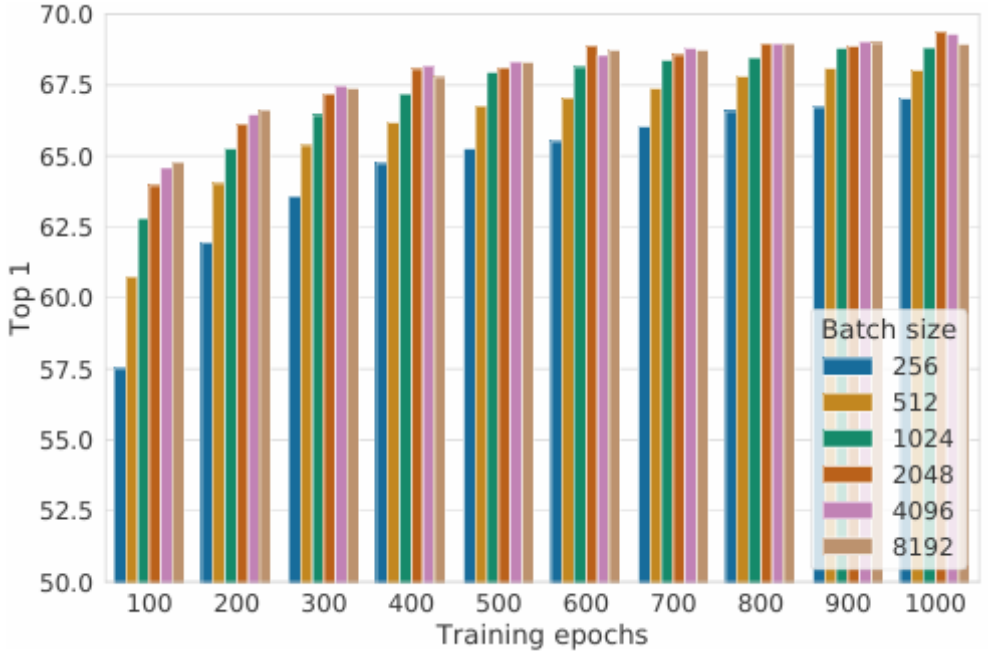
✓ By leveraging the projection head g(·), more information can be preserved in the h representation space.





SimCLR design choices: large batch size

- Large training batch size is crucial for SimCLR!
- Large batch size causes large memory footprint during backpropagation: requires distributed training on TPUs



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

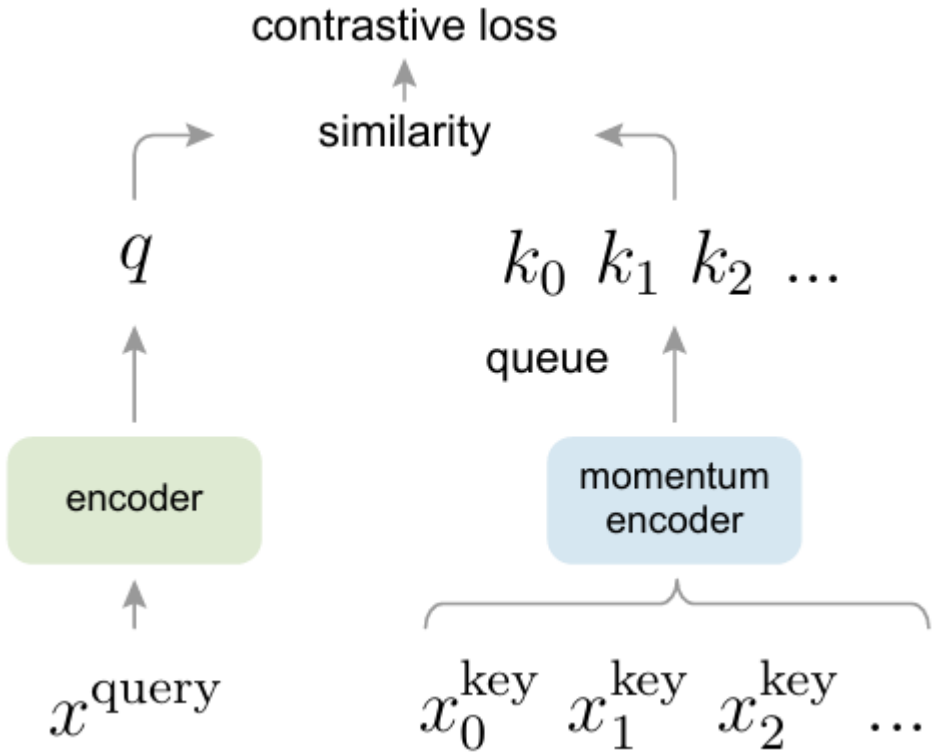
Momentum Contrastive Learning (MoCo)

- 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.



Momentum Contrastive Learning (MoCo)

- Key differences to SimCLR:

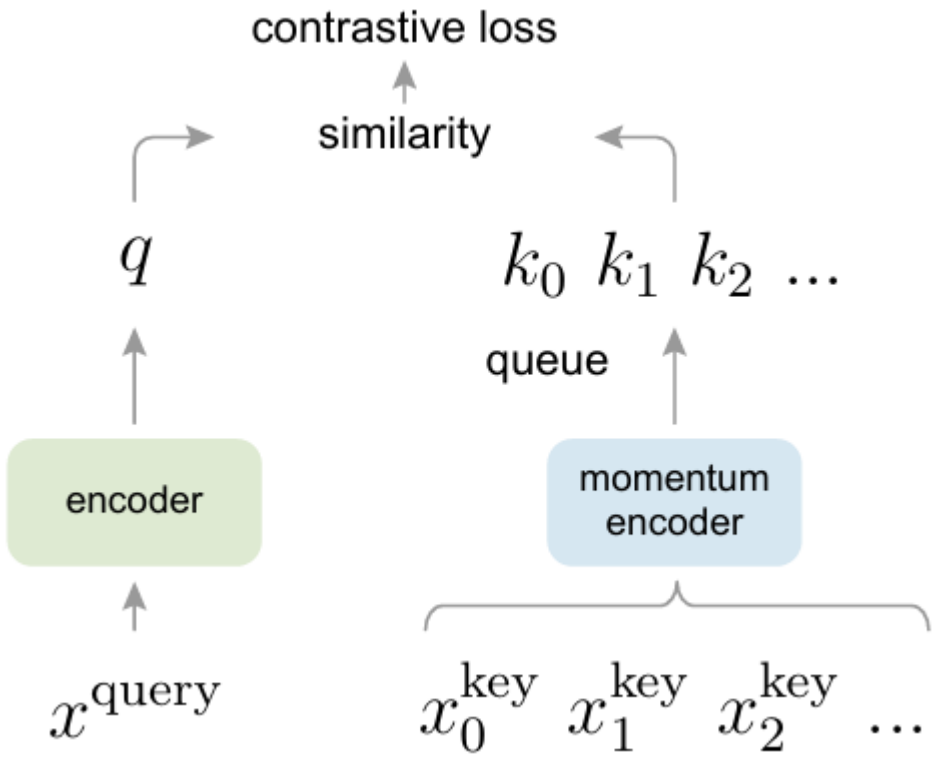
✓ Keep a running queue of keys (negative samples).

✓ Compute gradients and update the encoder only through the queries.

✓ Decouple min-batch size with the number of keys: can support a large number of negative samples.

✓ The key encoder is slowly progressing through the momentum update rules:

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



MoCo

Generate a positive pair by sampling data augmentation functions

No gradient through the key

Update the FIFO negative sample queue

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

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

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

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

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

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

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

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

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

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

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

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

Use the running queue of keys as the negative samples

InfoNCE loss

Update f_k through momentum

MoCo V2

- 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!).

Improved Baselines with Momentum Contrastive Learning

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

MoCo vs. SimCLR vs. MoCo V2

- Key takeaways:
  - ✓ Non-linear projection head and strong data augmentation are crucial for contrastive learning.

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.

MoCo vs. SimCLR vs. MoCo V2

- Key takeaways:
  - ✓ Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).

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).

MoCo vs. SimCLR vs. MoCo V2

- 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)

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.

DINO: Self-Distillation with No Labels

Emerging Properties in Self-Supervised Vision Transformers

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

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

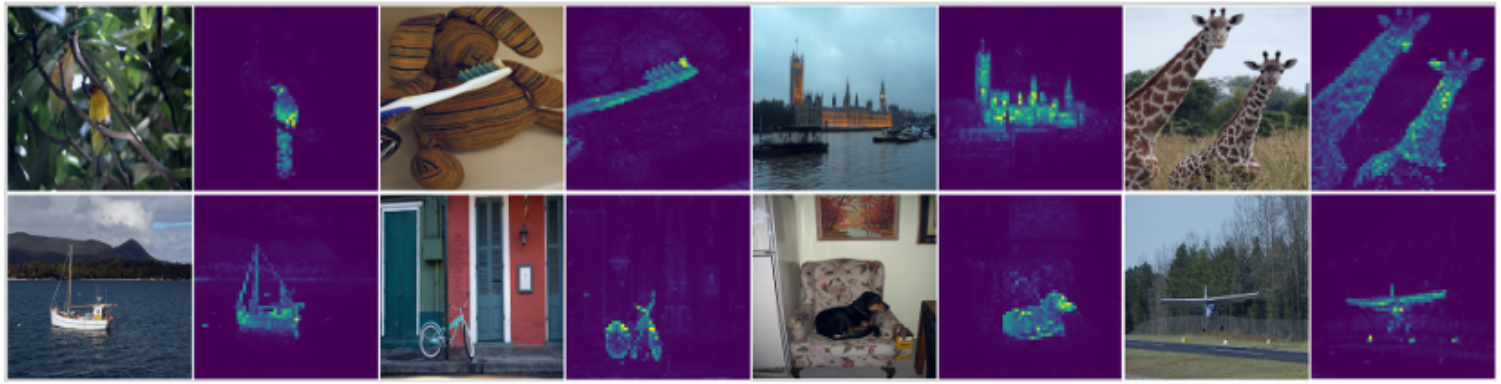


Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific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

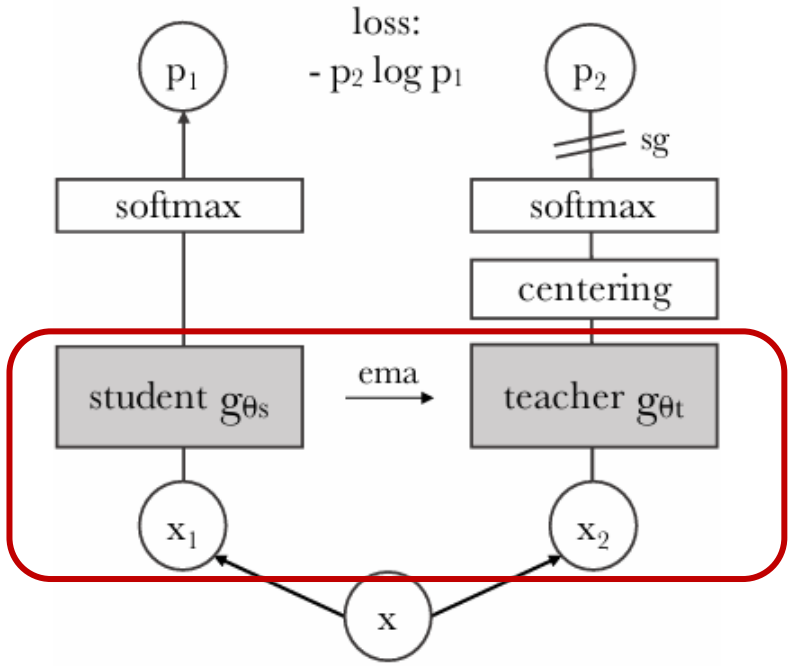
DINO

- **Input processing:**

✓ Two random augmentations are performed on the same image to generate global views (large crops) and local views (local crops).

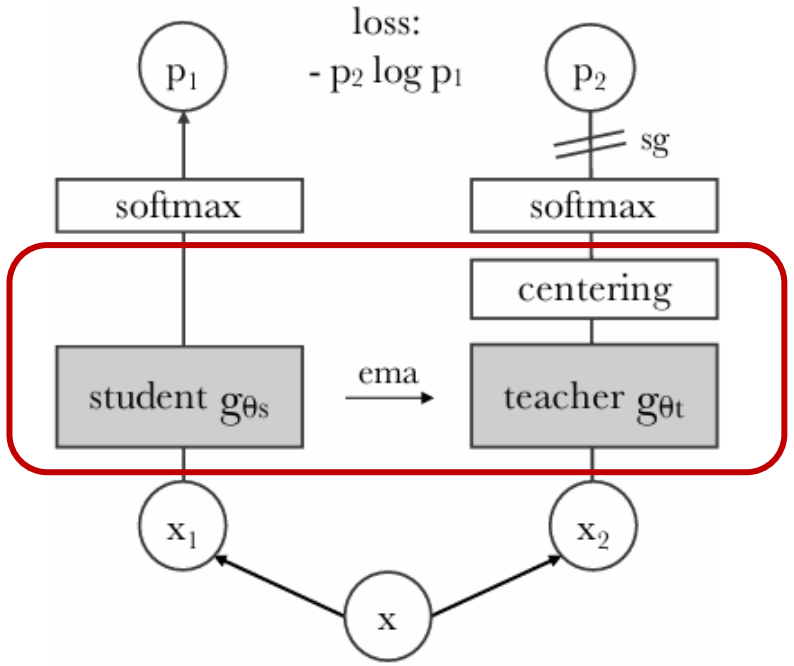
✓ **Global views:** Large patches (typically occupying 50%-100% of the original image).

✓ **Local views:** Small patches (typically occupying 20%-50% of the original image), multiple views are generated (e.g., 8).



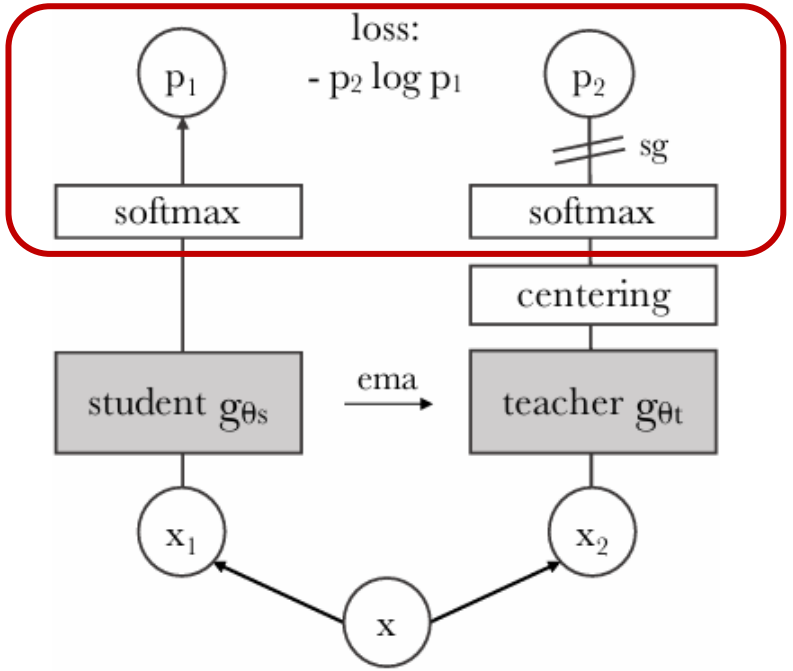
DINO

- **Teacher Model**: Input only global view (large tile), output probability distribution.
  - ✓ Teacher output undergoes Centering + Sharpening processing.
  - ✓ Gradients are not calculated to avoid mode collapse.
- **Student Model**: Input global view + local views (1 large image + 8 small images).
  - ✓ Predicted outputs are calculated for all views.



DINO

- Cross-entropy loss is used to measure the consistency between student and teacher predictions:
  - ✓ For each view, the student output matches the teacher's output for the global view.
  - ✓ The total loss is a weighted sum of the losses for all views (with a higher weight for the global view).



DINO

- **Centering**:

✓ Maintaining a moving average (mean) of teacher outputs.

✓ Subtracting this mean from each batch prevents any one dimension from dominating.

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()
```

DINO

- **Sharpening**:

✓ Using a low temperature parameter $\tau_t < 1$ makes the teacher output distribution sharper.

✓ This forces the model to make more deterministic predictions, enhancing feature discrimination.

✓ Students use a higher temperature $\tau_s > \tau_t$.:

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

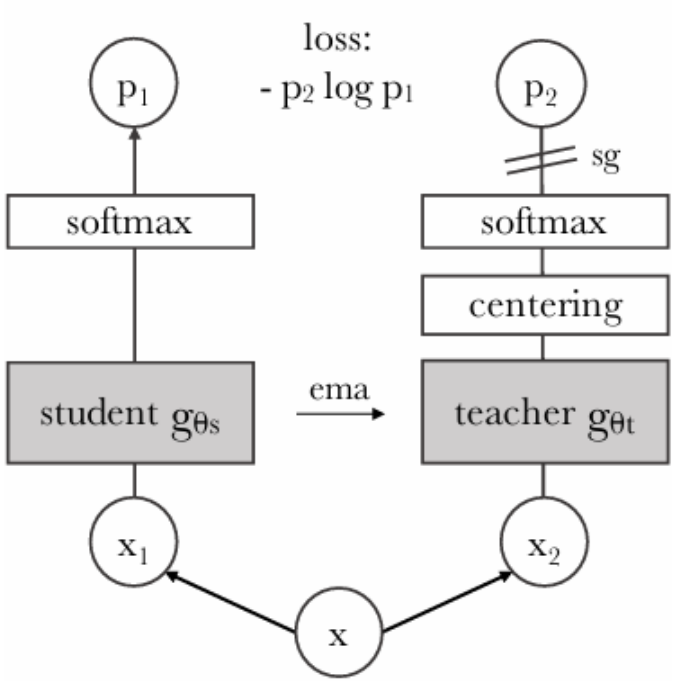
    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

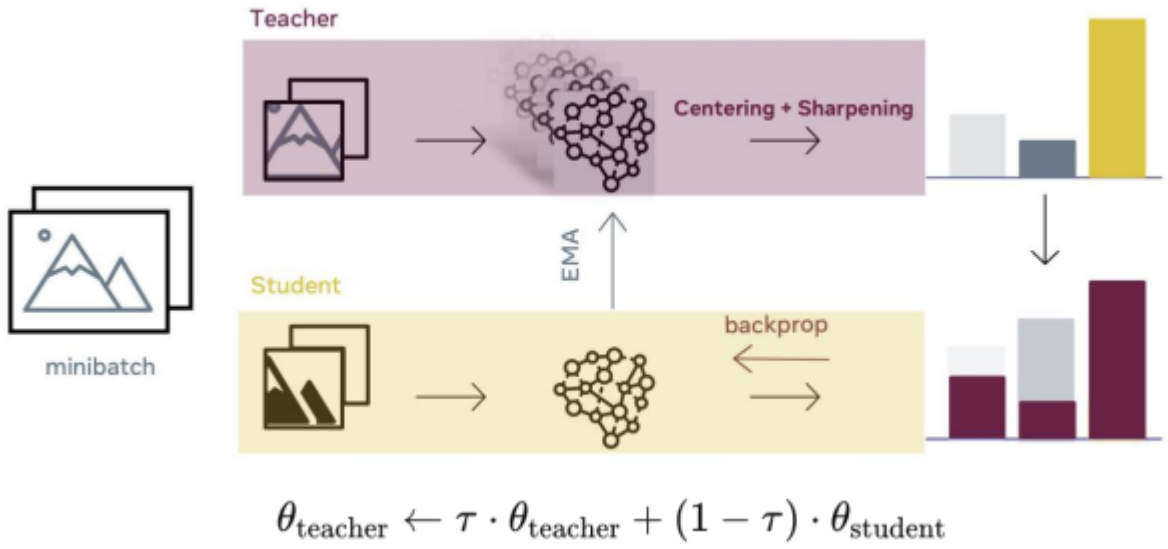
    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

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()
```

DINO

- The combination of momentum encoder and centering effectively prevents pattern collapse.
- No contrastive learning or negative samples are required.

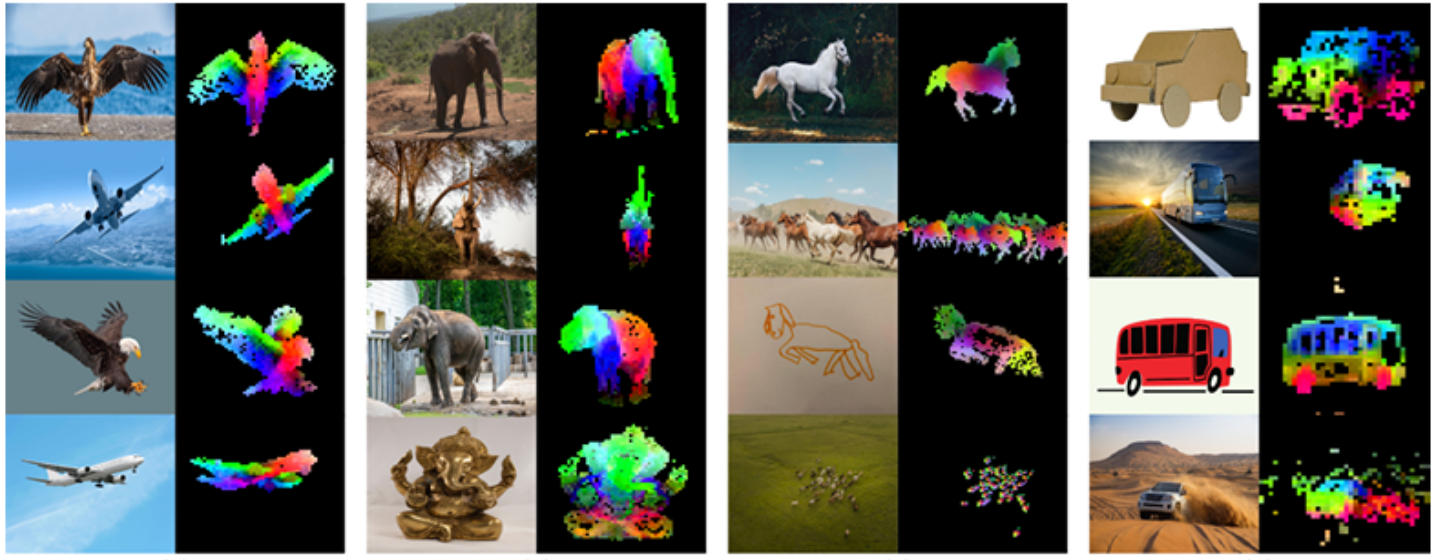




$$\theta_{\text{teacher}} \leftarrow \tau \cdot \theta_{\text{teacher}} + (1 - \tau) \cdot \theta_{\text{student}}$$

DINO v2

- **Data**: 10x larger + carefully selected deduplication
- **Model**: ViT-Giant + hierarchical decoder
- **Training**: Distillation target upgrade + longer LR schedule
- **Regulation**: KoLeo + MSE patch, locking in high-quality geometry



Thanks

Welcome to join VisionXLab

yangxue.site

