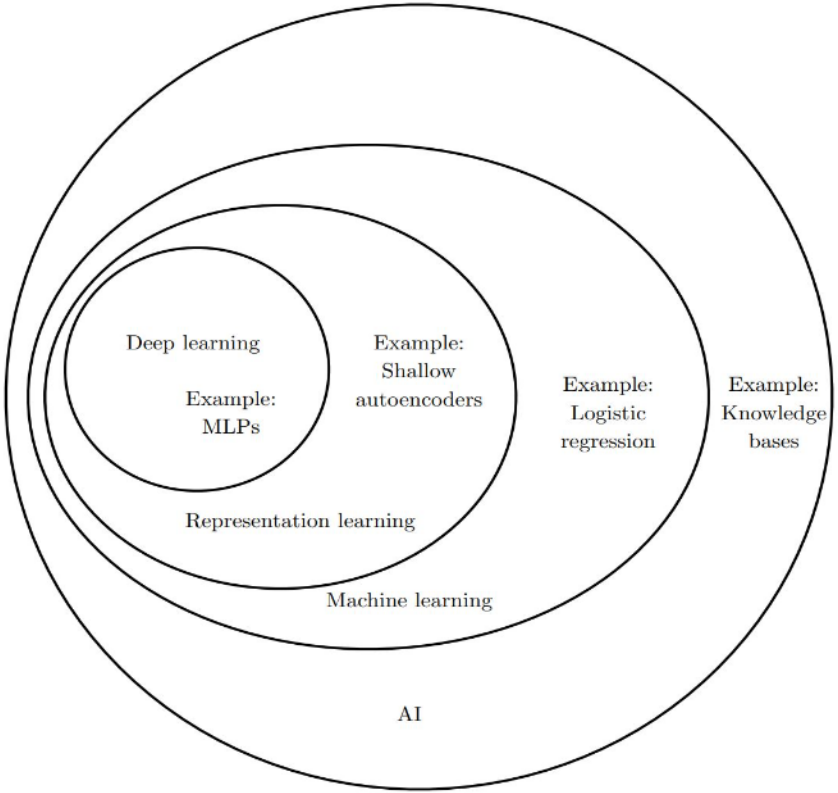
Deep Learning is one way to learn features



Deep Unsupervised Learning:

1. Learn representations without labels
2. Subset of Deep Learning, which is a subset of Representation Learning, which is a subset of Machine Learning

Self-Supervised Learning

1. Often used interchangeably with unsupervised learning
2. Self-Supervised: Create your own supervision through pretext tasks

Why Self-Supervised Learning?

- Expense of producing a new dataset for each task
  - Prepare labeling manuals, categories, hiring humans, creating GUIs, storage pipelines, etc.
- Good supervision may not be cheap (ex: medicine, legal)
- Take advantage of vast amount of unlabeled data on the internet (images, videos, language).
- Cognitive motivation: How animals / babies learn

What is Self-Supervised Learning?

- A version of unsupervised learning where data provides the supervision.
- In general, withhold some part of the data and the task a neural network to predict it from the remaining parts.
- Details decide what proxy loss or pretext task the network tries to solve, and depending on the quality of the task, good semantic features can be obtained without actual labels.

unsupervised learning aims to discover patterns or relationships in data without explicit labels, while self-supervised learning leverages surrogate tasks to create artificial labels and learn meaningful representations from the data. Self-supervised learning can be seen as a specific technique within the broader domain of unsupervised learning.

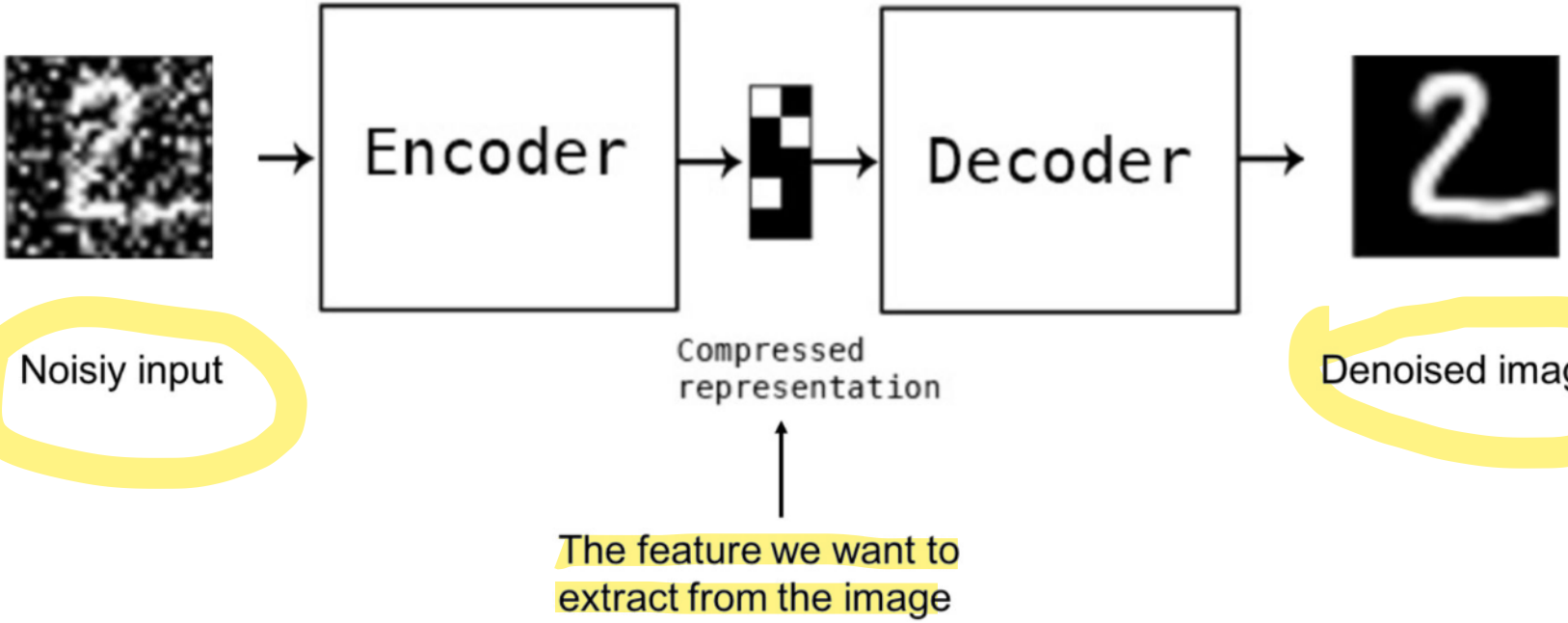
Motivation

- Supervised learning success story is heavily because of the utility of pre-trained classifier features for commercially useful downstream tasks like segmentation, detection, etc.
- Recipe is clear: Collect a large labeled dataset, train a model, deploy. Good data and sufficient data are what you need.
- Goal of self-supervised learning:
  - Learn equally good (if not better) features without supervision
  - Be able to deploy similar quality systems without relying on too many labels for the downstream tasks
  - Generalize better potentially because you learn more about the world

Cognitive Principles

- Reconstruct from a corrupted (or partial) version
  - Denoising Autoencoder
  - In-painting
  - Colorization, Split-Brain Autoencoder
- Visual common sense tasks
  - Relative patch prediction
  - Jigsaw puzzles
  - Rotation
- Contrastive Learning
  - word2vec
  - Contrastive Predictive Coding (CPC)
  - Instance Discrimination
  - Recent State-of-the-art progress

Denoising Autoencoder



Denoising Autoencoder

Masking noise refers to a type of noise where a portion of the input data is randomly masked or hidden. This is typically achieved by setting a certain fraction of values in the data to zero or some other predefined value. The masked values are then considered as missing or unknown during the training process.For example, if we have an input sequence [1, 2, 3, 4, 5], applying masking noise might result in [1, 0, 3, 4, 0]. Here, a fraction of the values (e.g., 40%) has been masked with zeros.

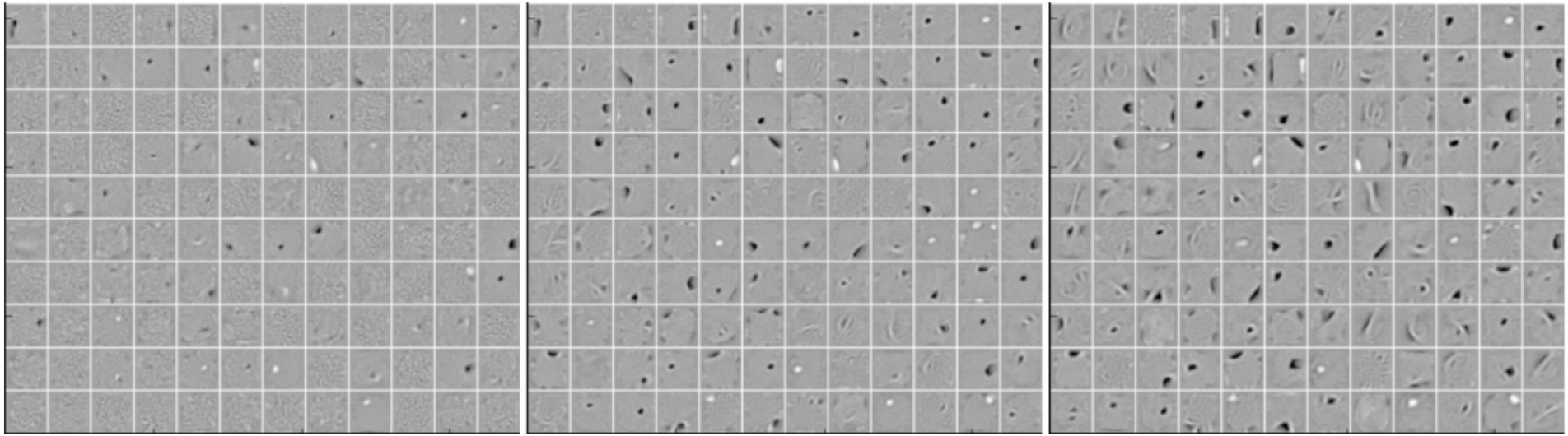
- Additive isotropic *Gaussian noise* (GS): $\tilde{\mathbf{x}}|\mathbf{x} \sim \mathcal{N}(\mathbf{x}, \sigma^2 I)$;
- *Masking noise* (MN): a fraction ν of the elements of $\mathbf{x}$ (chosen at random for each example) is forced to 0;
- *Salt-and-pepper noise* (SP): a fraction ν of the elements of $\mathbf{x}$ (chosen at random for each example) is set to their minimum or maximum possible value (typically 0 or 1) according to a fair coin flip.

Salt and pepper noise, also known as impulse noise, is a type of noise where random pixels or elements in the input data are replaced with extreme values. Typically, these extreme values are the maximum and minimum possible values within the data range.
For example, in an image, salt and pepper noise may cause some pixels to become completely white (maximum intensity) or completely black (minimum intensity), resembling grains of salt and pepper scattered randomly.

Vincent et al 2010

Masking noise focuses on handling missing or unknown data, while salt and pepper noise tests the model's robustness against outliers or extreme values.

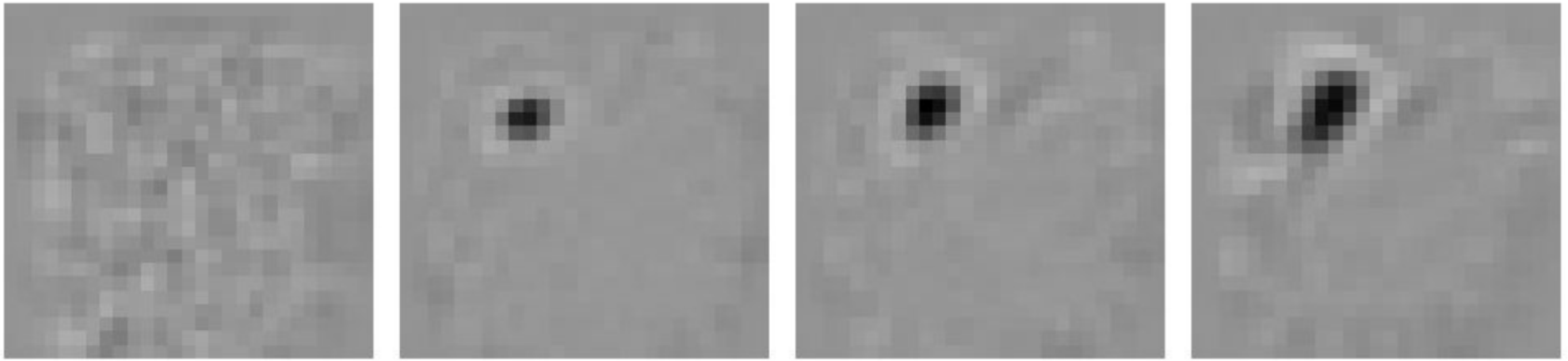
Denoising Autoencoder



(a) No destroyed inputs (b) 25% destruction (c) 50% destruction

Vincent et al 2010

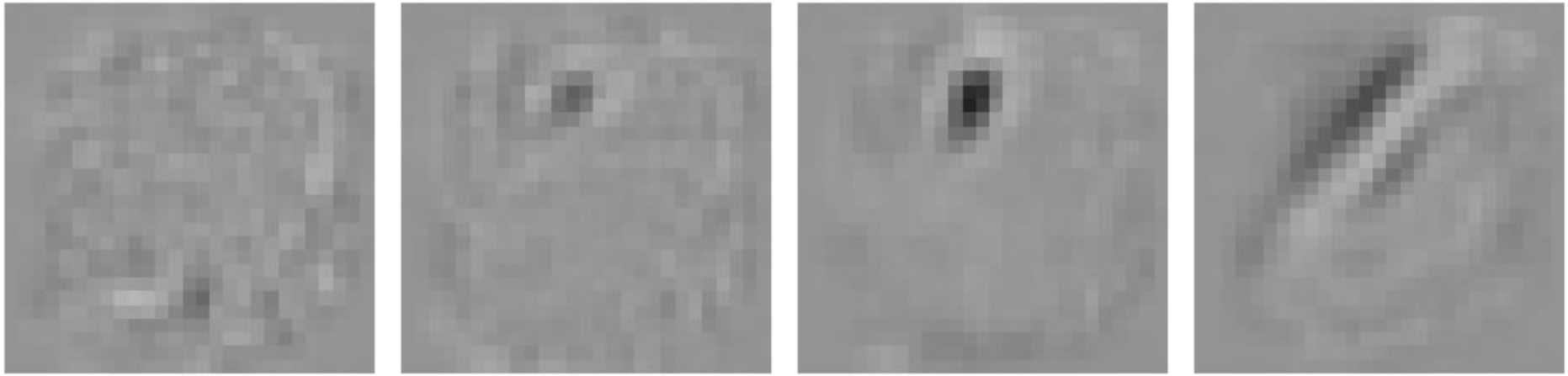
Denoising Autoencoder



(d) Neuron A (0%, 10%, 20%, 50% destruction)

Vincent et al 2010

Denoising Autoencoder



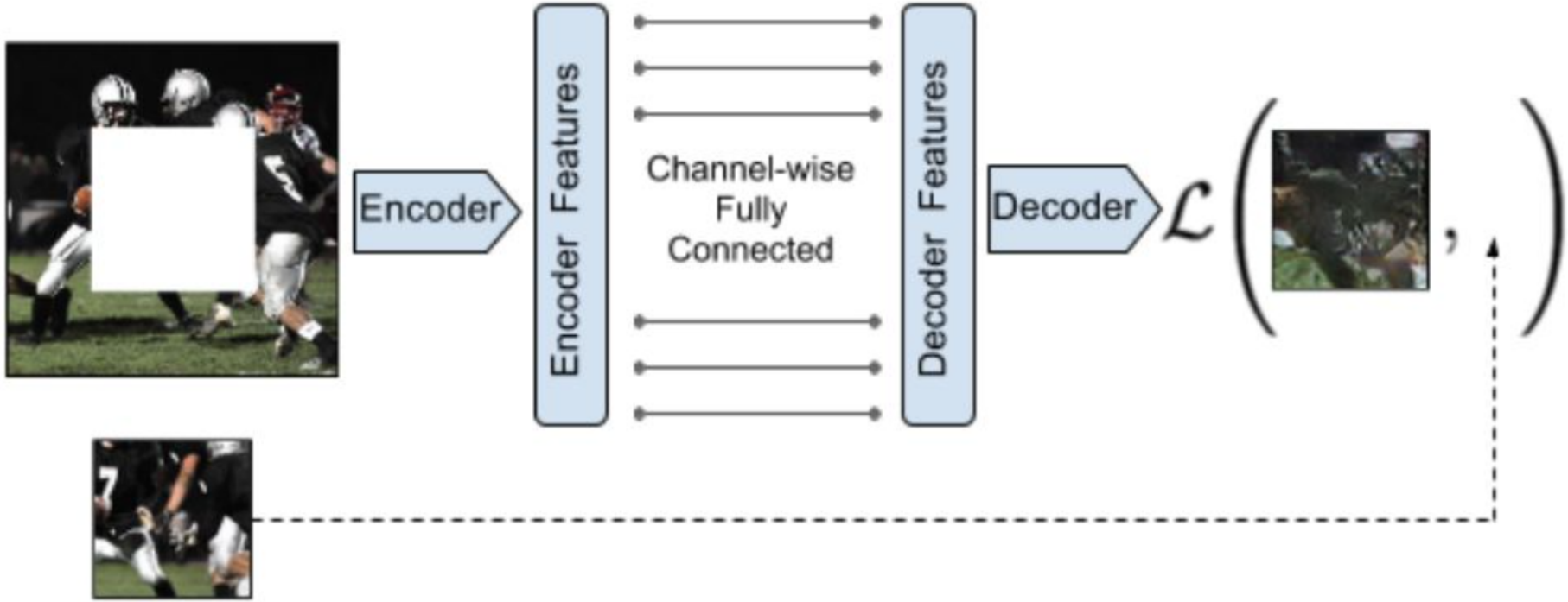
(e) Neuron B (0%, 10%, 20%, 50% destruction)

Vincent et al 2010

Predict missing pieces

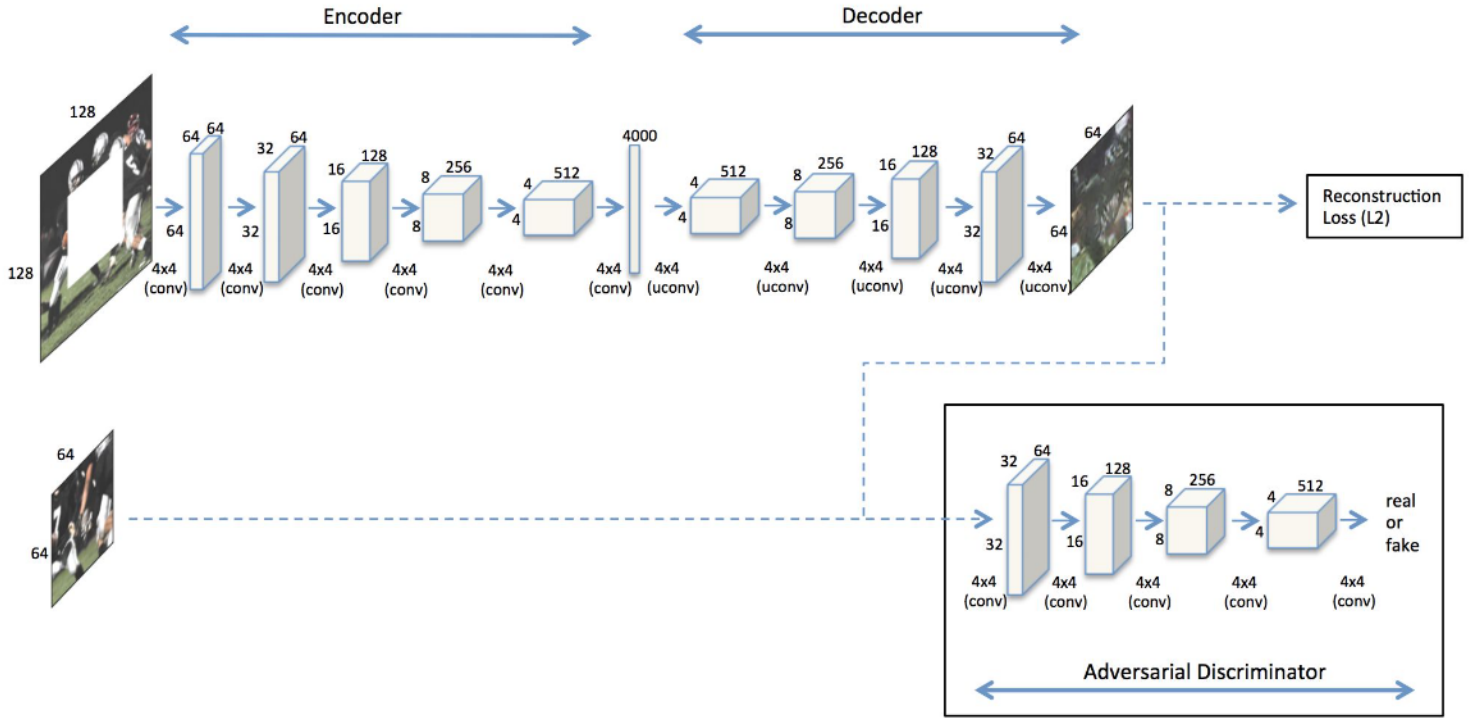
Pathak et al 2016

Context Encoders



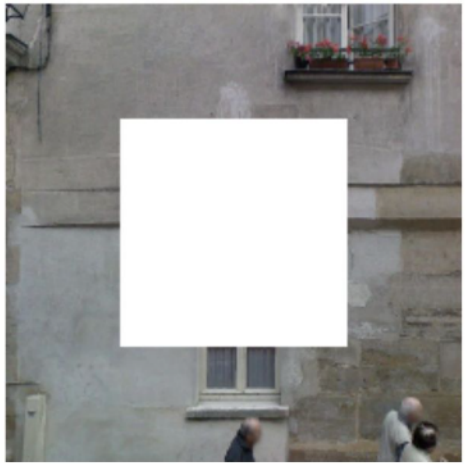
Pathak et al 2016

Context Encoders



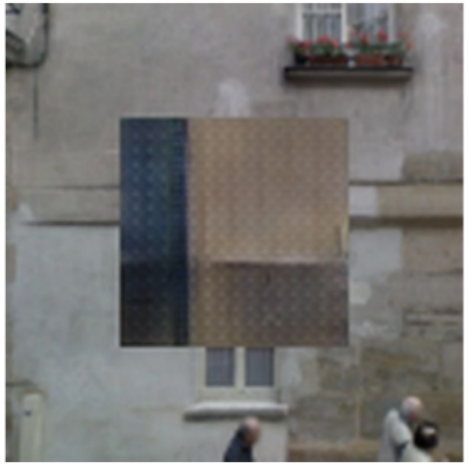
Pathak et al 2016

Context Encoders



Input Image

L2 Loss



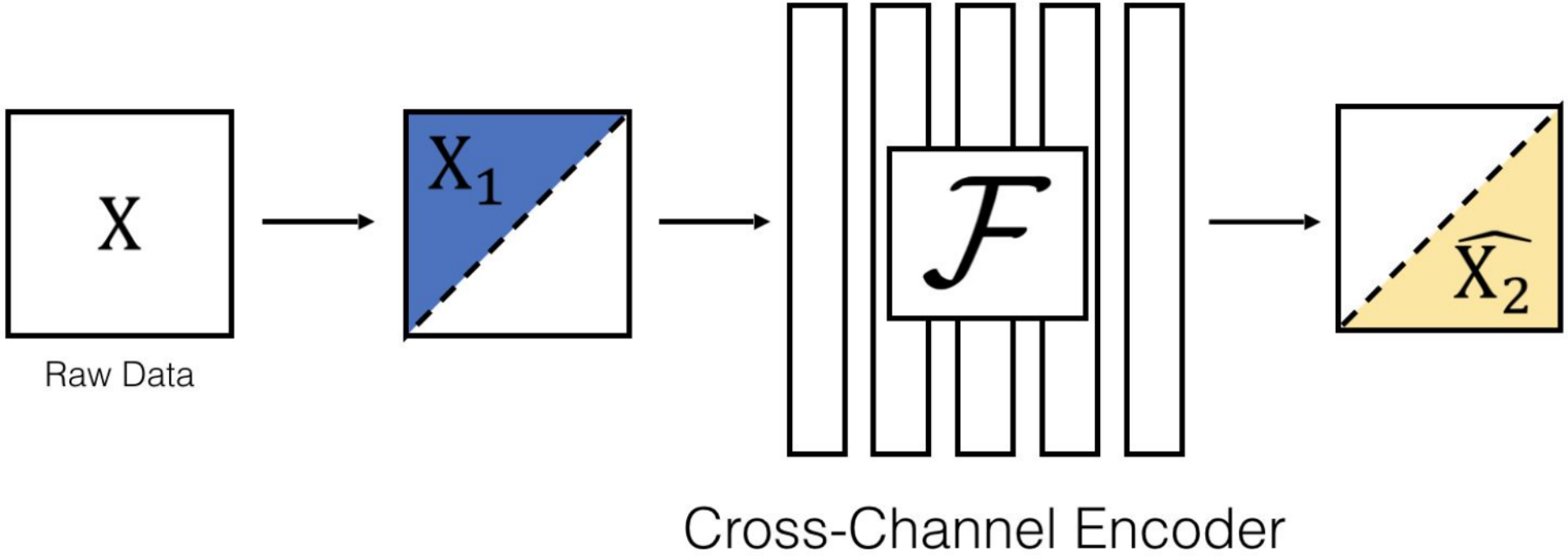
Adversarial Loss

Joint Loss

The joint loss combines both the reconstruction loss and the adversarial loss, typically by summing or weighting them together

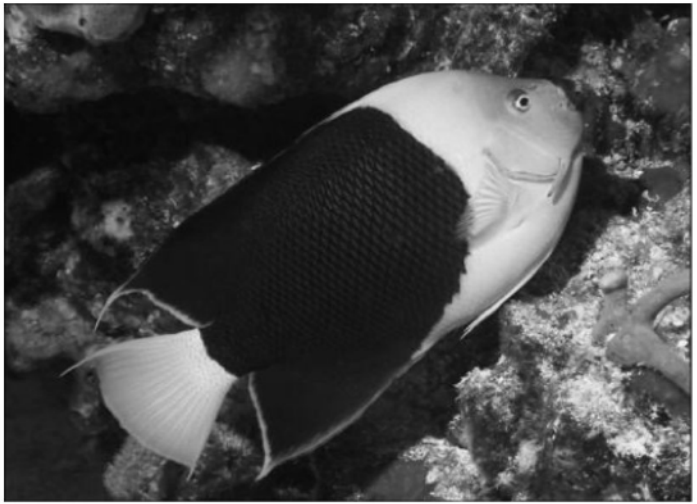
Pathak et al 2016

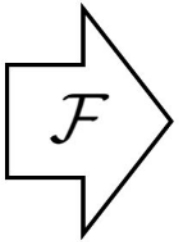
Predicting one view from another



Slide: Richard Zhang

Predicting one view from another





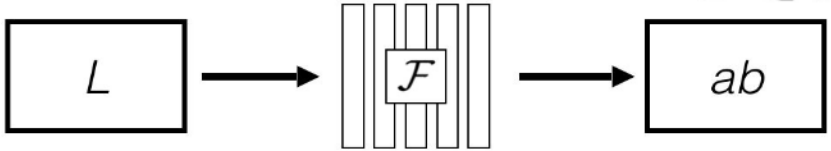
Grayscale image: *L* channel

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

Color information: *ab* channels

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

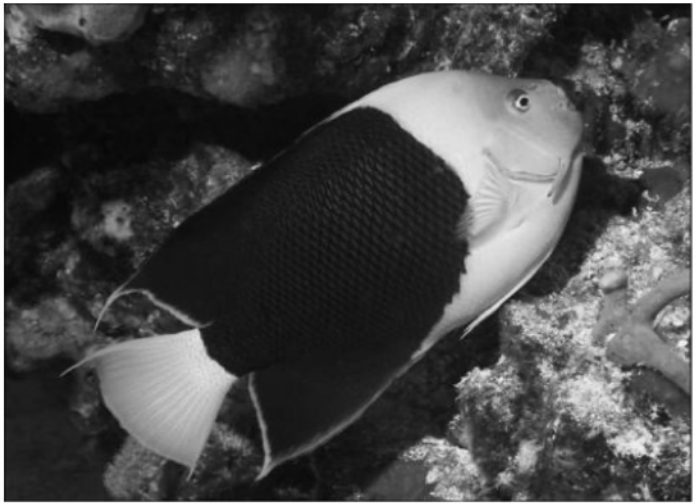
The goal of the cross-channel encoder is to create a shared latent space where the common features between the views are represented.



This encoder should be capable of capturing the shared information between the views while discarding view-specific details.

Slide: Richard Zhang

Predicting one view from another



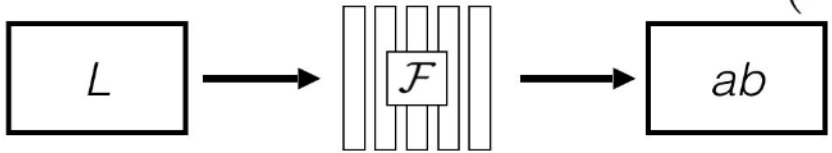
$\mathcal{F}$

Grayscale image: *L* channel

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

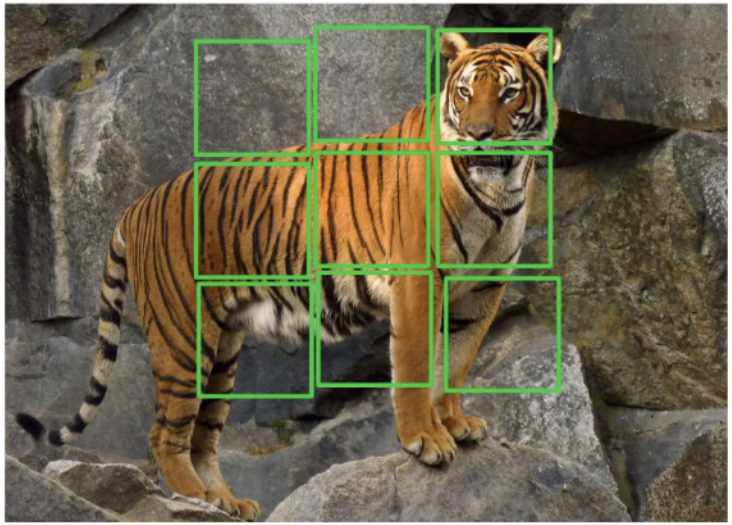
Concatenate (*L*,*ab*) channels

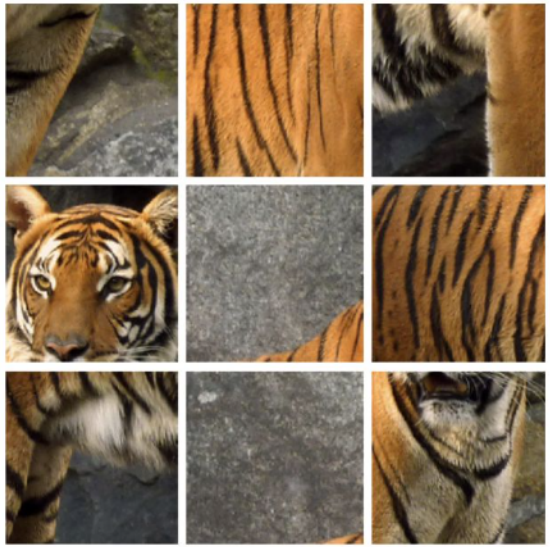
$(\mathbf{X}, \widehat{\mathbf{Y}})$



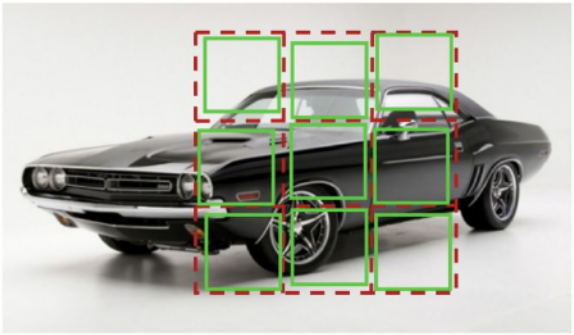
Slide: Richard Zhang

Solving Jigsaw Puzzles





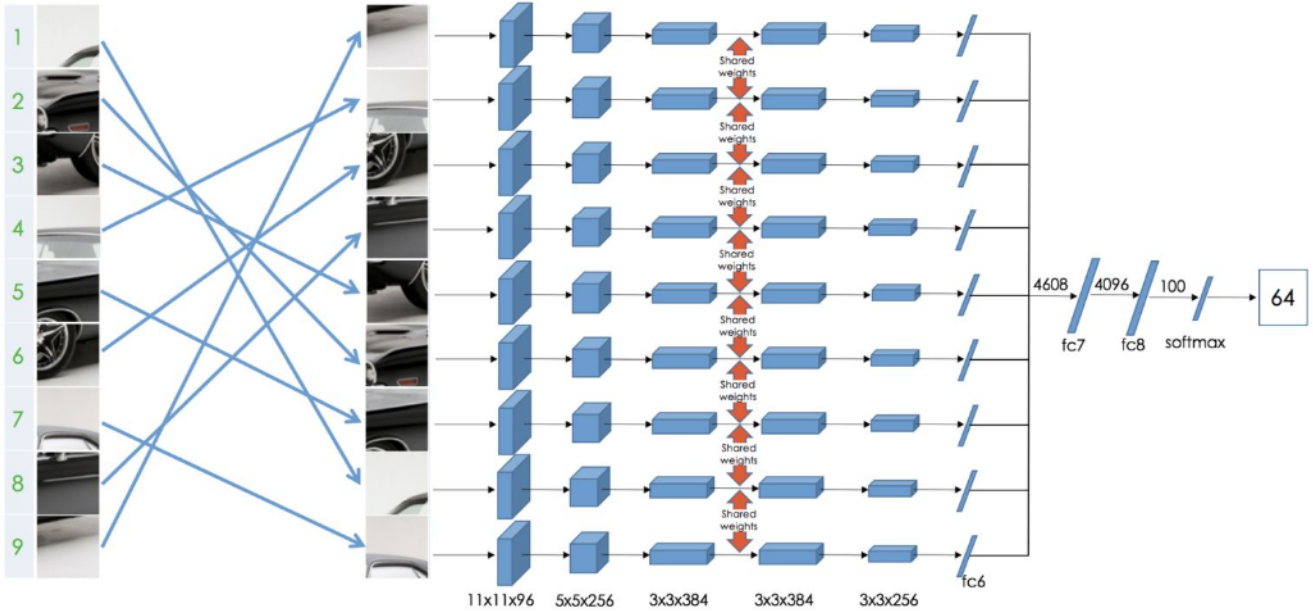
Solving Jigsaw Puzzles



Permutation Set

index	permutation
64	9,4,6,8,3,2,5,1,7

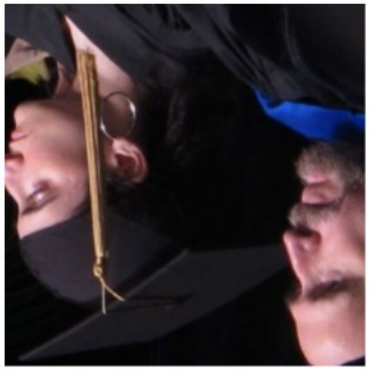
Reorder patches according to the selected permutation



Rotation

90° rotation

270° rotation



180° rotation

0° rotation

270° rotation

Rotation

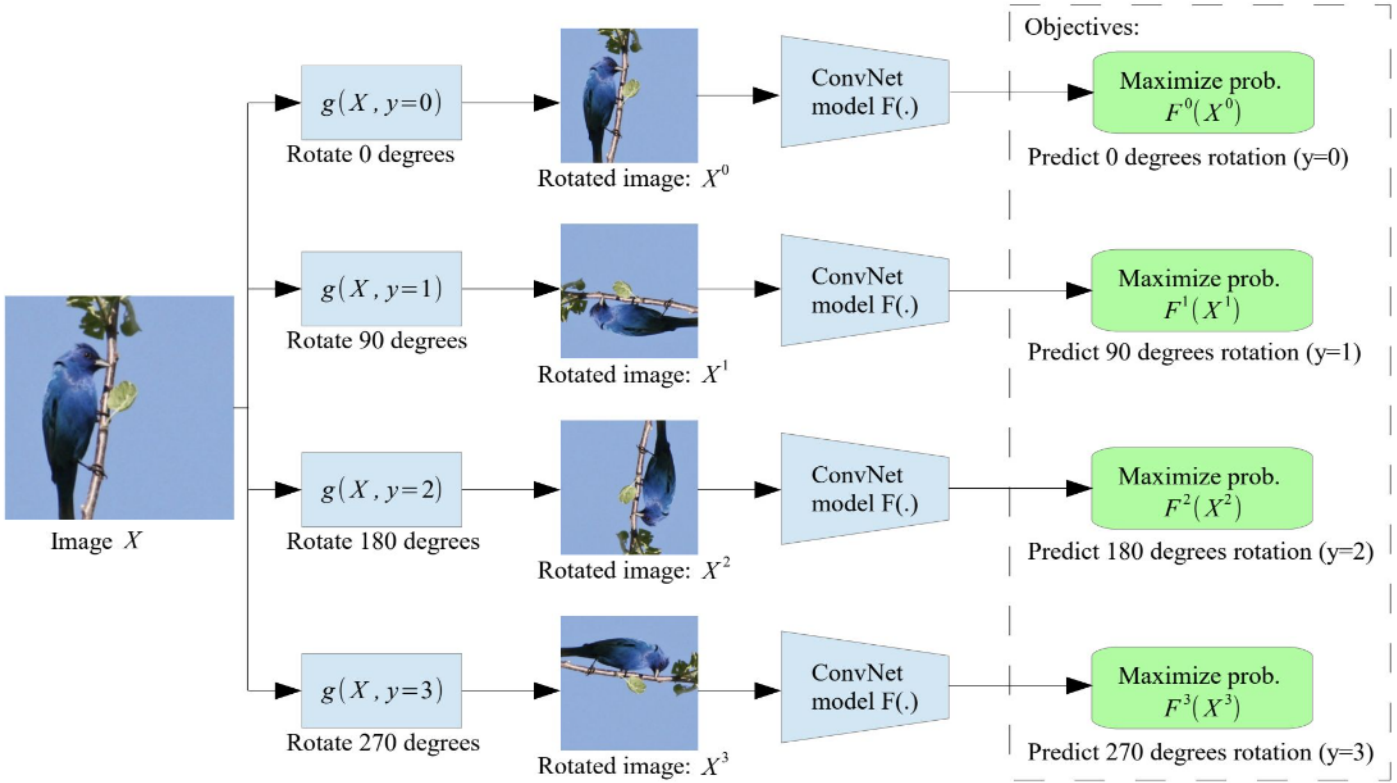


Image X
$g(X, y=0)$
Rotate 0 degrees
Rotated image: X^0
$g(X, y=1)$
Rotate 90 degrees
Rotated image: X^1
$g(X, y=2)$
Rotate 180 degrees
Rotated image: X^2
$g(X, y=3)$
Rotate 270 degrees
Rotated image: X^3
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
Objectives:
Maximize prob. $F^0(X^0)$
Predict 0 degrees rotation (y=0)
Maximize prob. $F^1(X^1)$
Predict 90 degrees rotation (y=1)
Maximize prob. $F^2(X^2)$
Predict 180 degrees rotation (y=2)
Maximize prob. $F^3(X^3)$
Predict 270 degrees rotation (y=3)

Rotation

# Rotations	Rotations	CIFAR-10 Classification Accuracy
4	0°, 90°, 180°, 270°	**89.06**
8	0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°	88.51
2	0°, 180°	87.46
2	90°, 270°	85.52

Rotation

Method	Conv4	Conv5
ImageNet labels from (Bojanowski & Joulin, 2017)	59.7	59.7
Random from (Noroozi & Favaro, 2016)	27.1	12.0
Tracking Wang & Gupta (2015)	38.8	29.8
Context (Doersch et al., 2015)	45.6	30.4
Colorization (Zhang et al., 2016a)	40.7	35.2
Jigsaw Puzzles (Noroozi & Favaro, 2016)	45.3	34.6
BIGAN (Donahue et al., 2016)	41.9	32.2
NAT (Bojanowski & Joulin, 2017)	-	36.0
(Ours) RotNet	50.0	43.8

Temporal coherence of color

Task: given a color video …
Colorize all frames of a gray scale version using a reference frame

Reference Frame

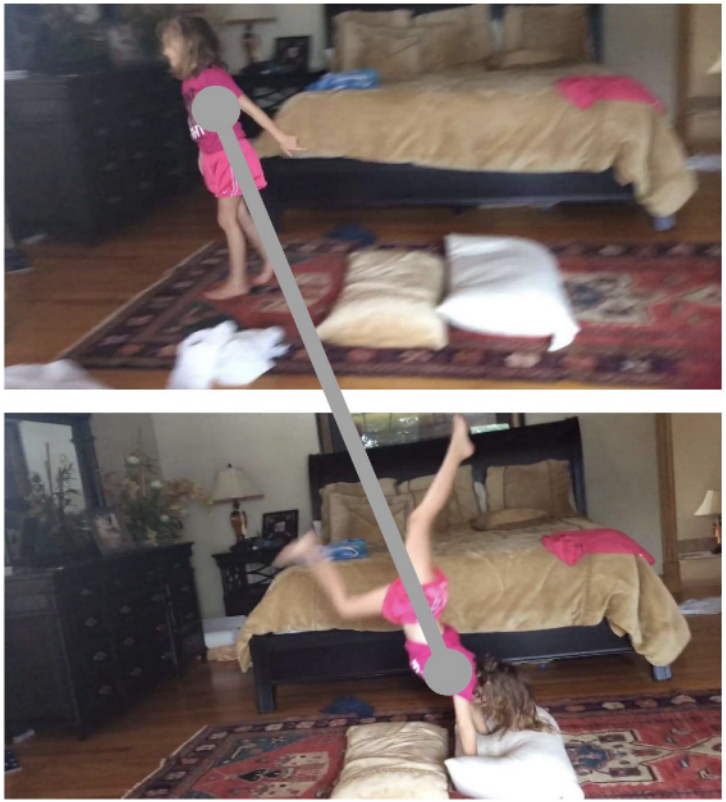
Gray-scale Video

Slide: Zisserman

Temporal coherence of color

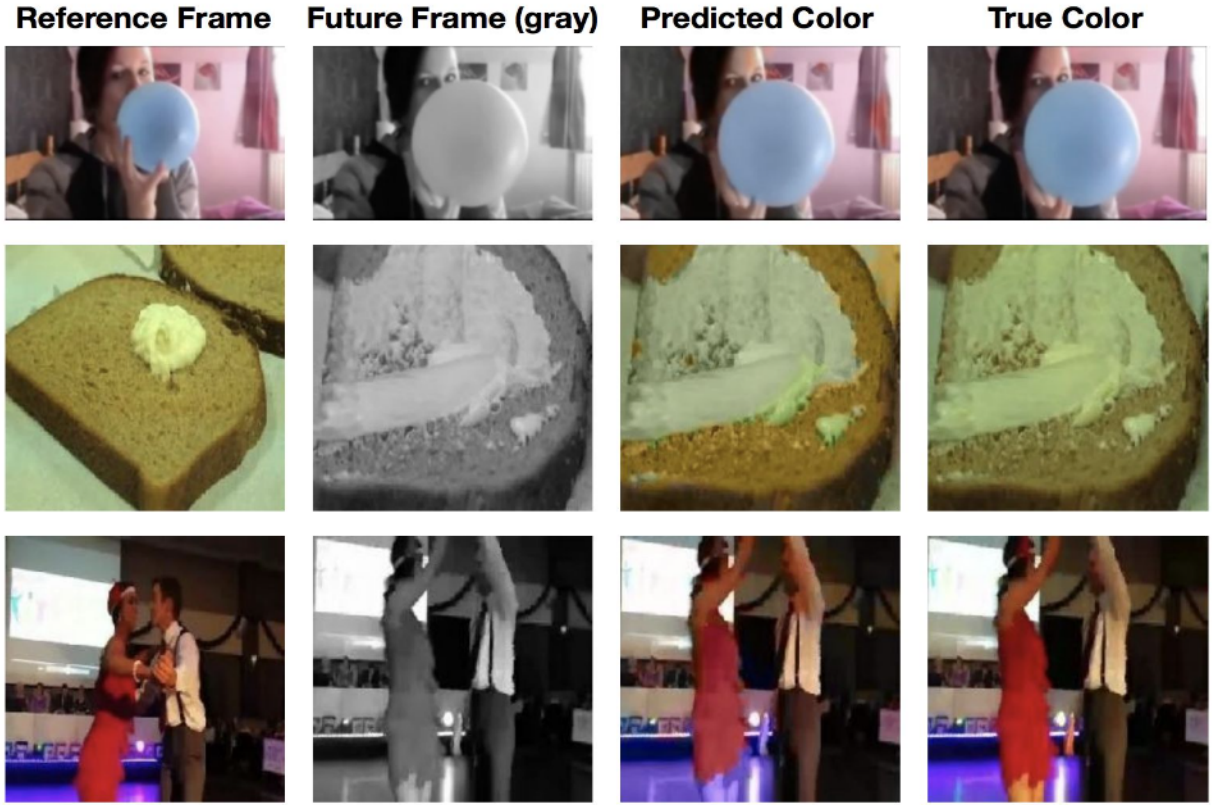
Slide: Zisserman

Temporal coherence of color

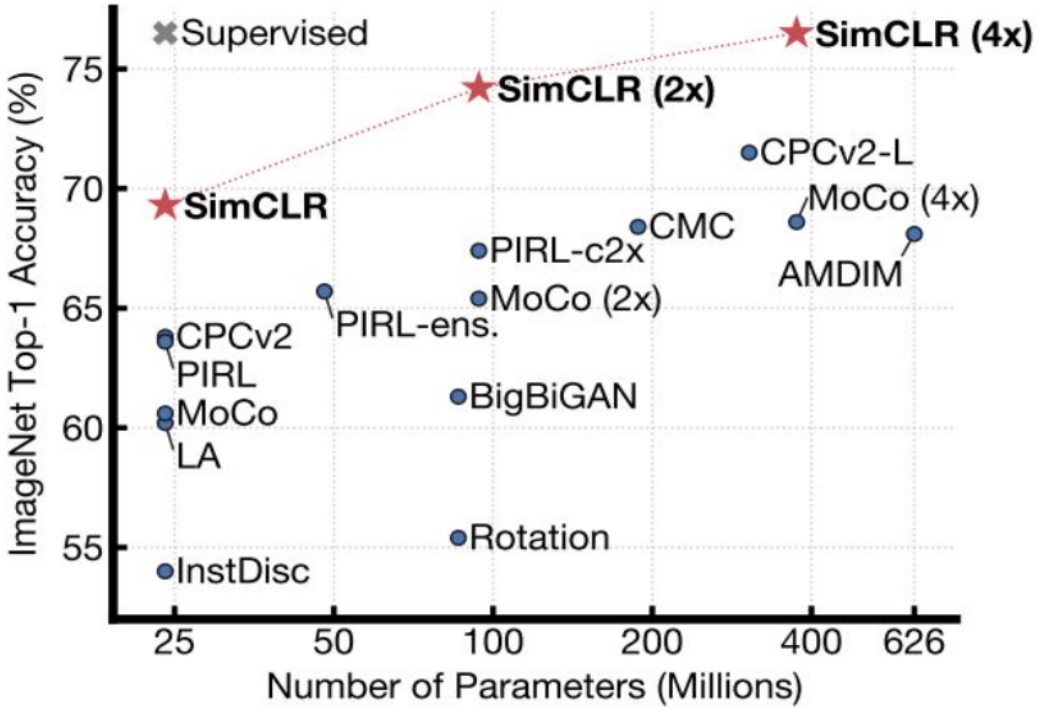


Slide: Zisserman

Tracking emerges from colorization



SimCLR


Supervised
SimCLR (4x)
SimCLR (2x)
CPCv2-L
MoCo (4x)
SimCLR
CMC
PIRL-c2x
AMDIM
MoCo (2x)
PIRL-ens.
CPCv2
PIRL
BigBiGAN
MoCo
LA
Rotation
InstDisc
ImageNet Top-1 Accuracy (%)
75
70
65
60
55
25
50
100
200
400
626
Number of Parameters (Millions)

Instance Discrimination

attract

repel

1. MoCo
2. SimCLR