Self Supervised Learning

R. Venkatesh Babu

Organization

- How to learn rich and useful features from unlabeled data
- Proxy tasks for representation learning
- Improving data/performance efficiency of downstream 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.

Goal of self-supervised learning:

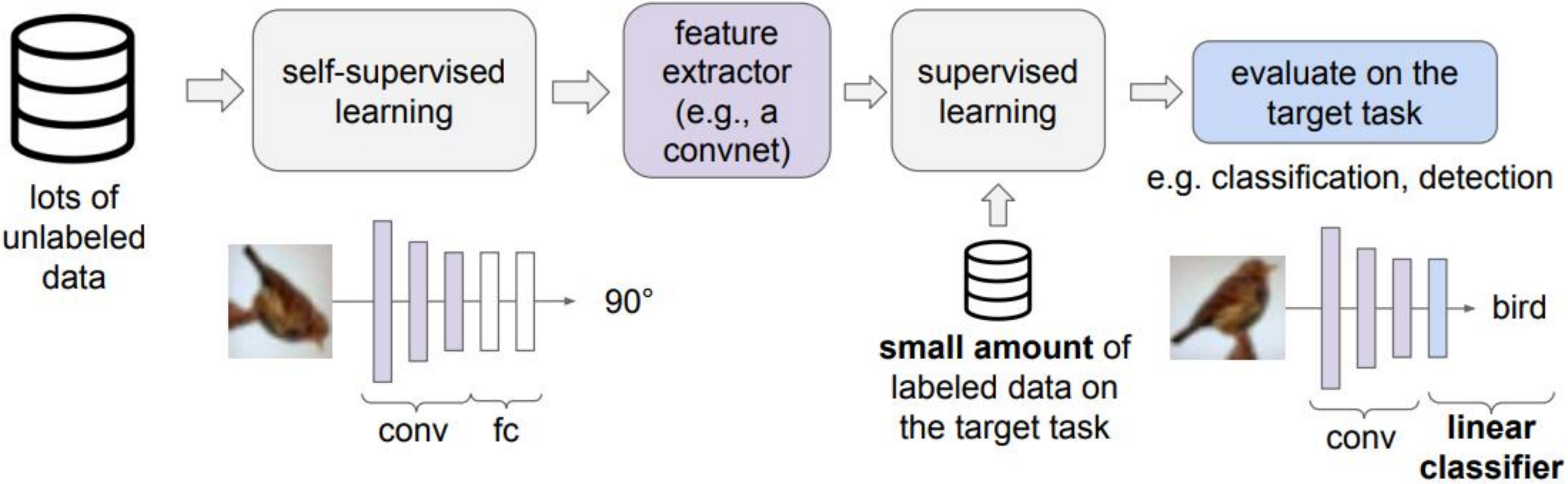
- Learn equally good (if not better) features without supervision
- Be able to deploy similar quality systems without relying on too many labels for the downstream tasks
- Generalize better potentially because you learn more about the world

How to evaluate SSL methods?

- 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 SSL methods?



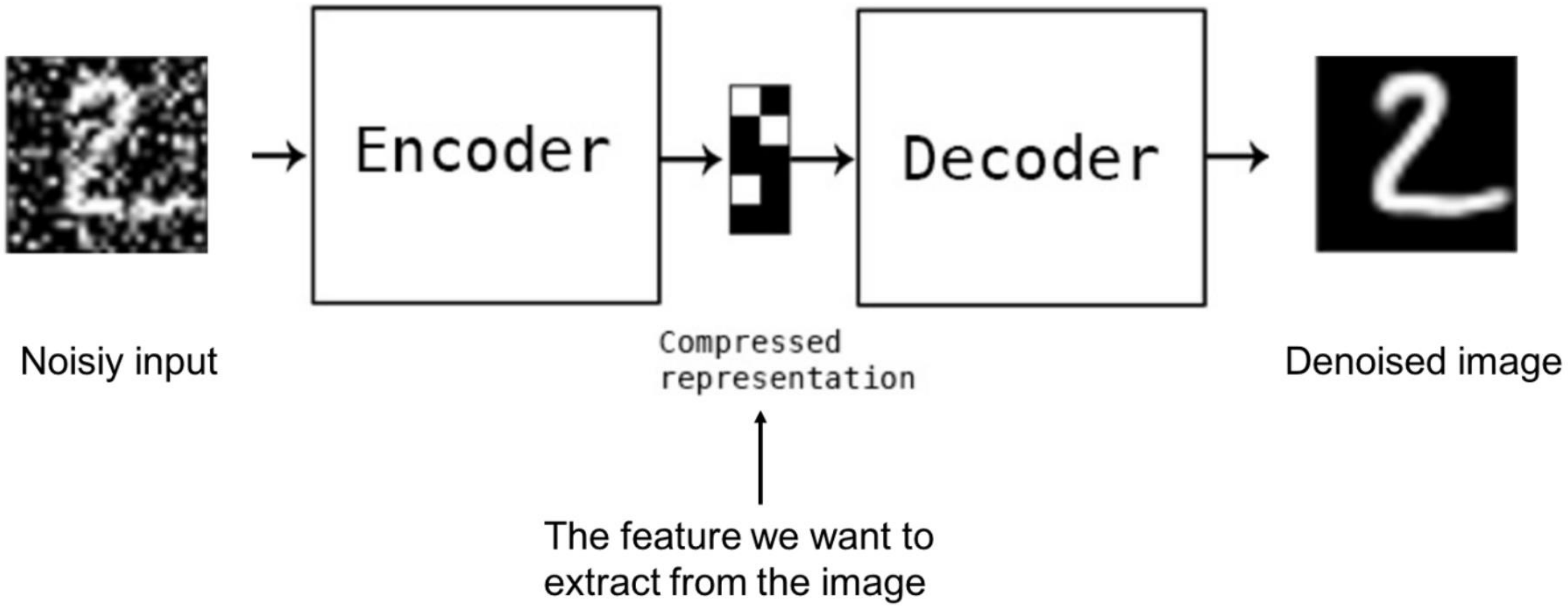
1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

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

Cognitive Principle

- 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

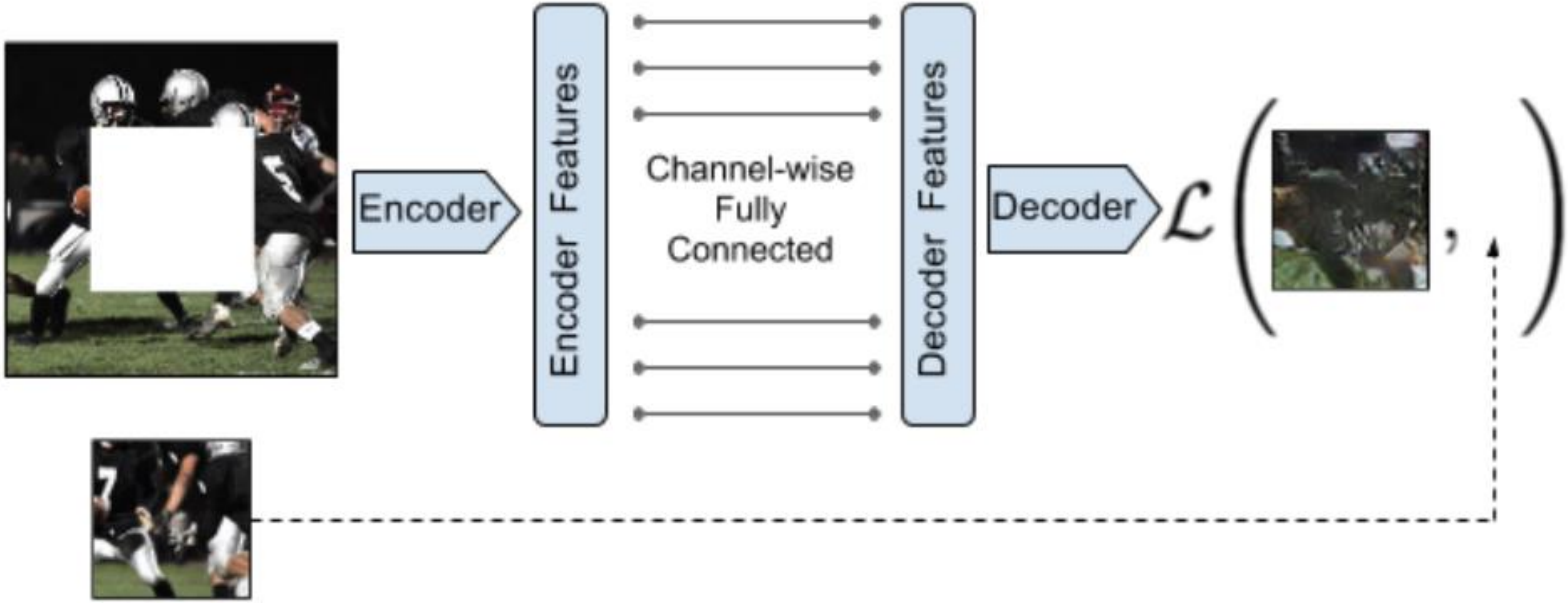
Denoising Auto-encoder



Predict missing pieces

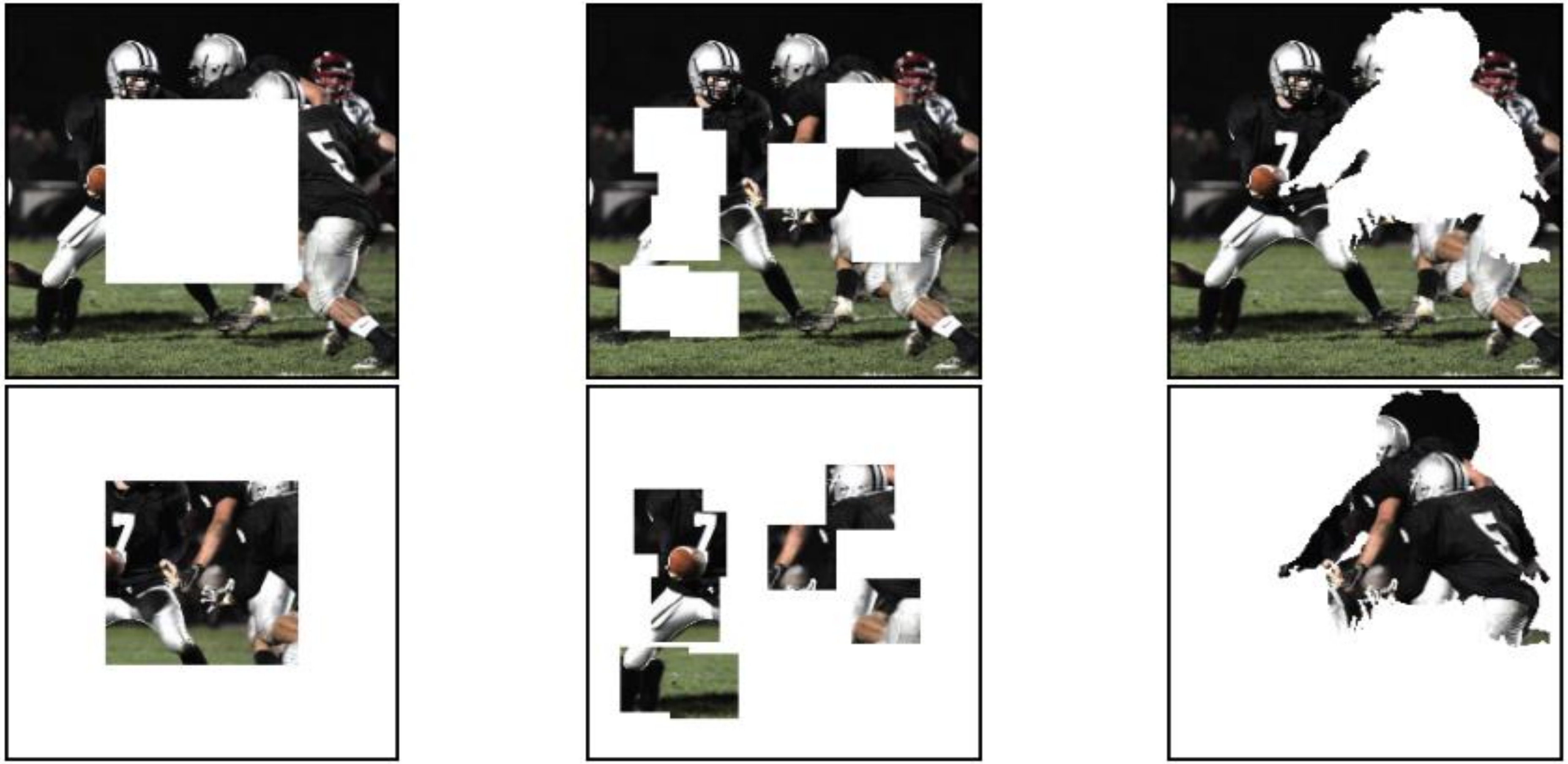
Pathak et al 2016

Context Encoders



Pathak et al 2016

Context Encoders



(a) Central region (b) Random block (c) Random region

Pathak et al 2016

Context Encoders

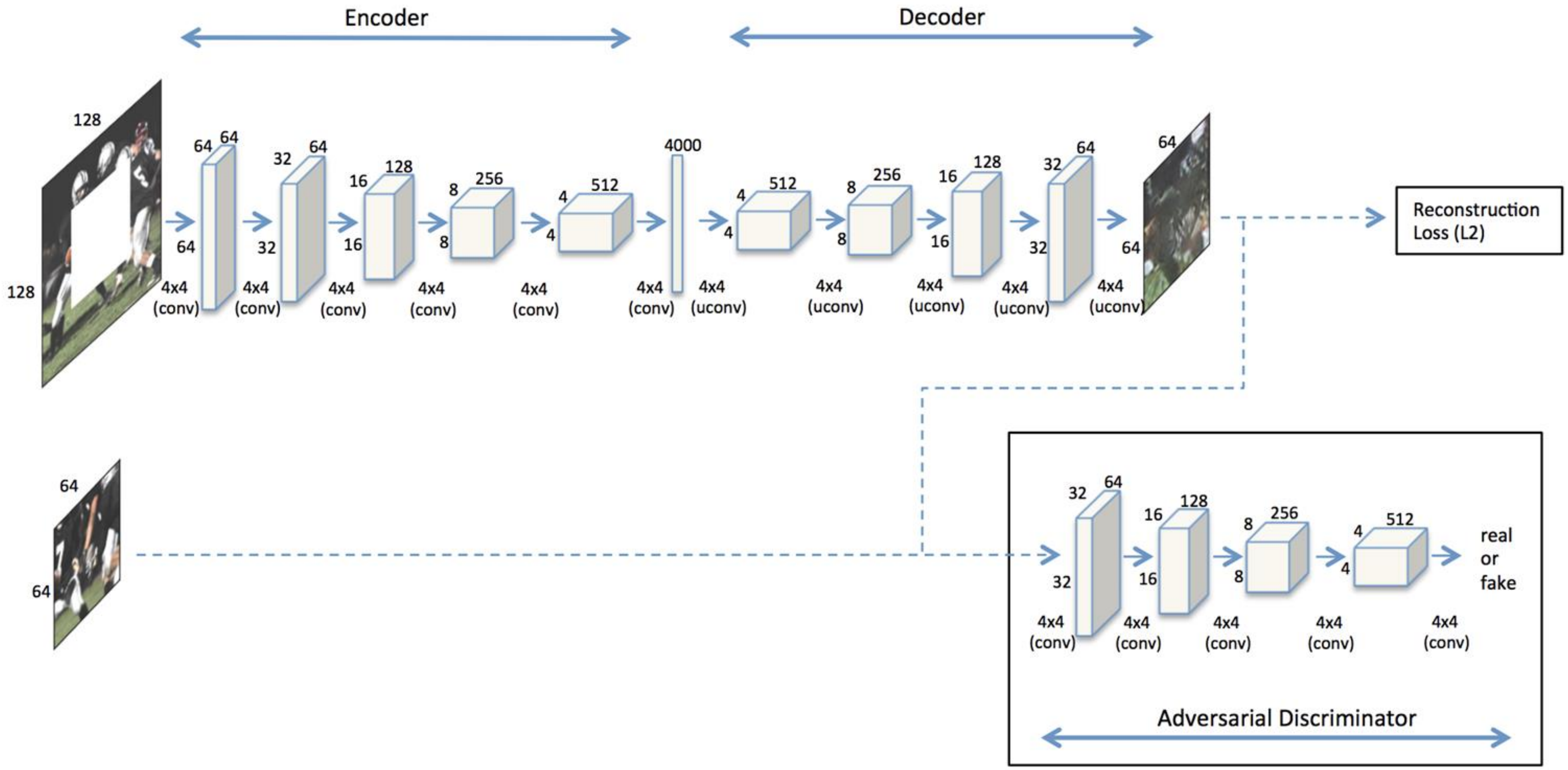
$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2$$

$$\begin{aligned}\mathcal{L}_{adv} = \max_{D} \quad & \mathbb{E}_{x \in \mathcal{X}}[\log(D(x)) \\ & + \log(1 - D(F((1 - \hat{M}) \odot x)))]\end{aligned}$$

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}$$

Pathak et al 2016

Context Encoders



Pathak et al 2016

Context Encoders

Input Image L2 Loss Adversarial Loss Joint Loss

Pathak et al 2016

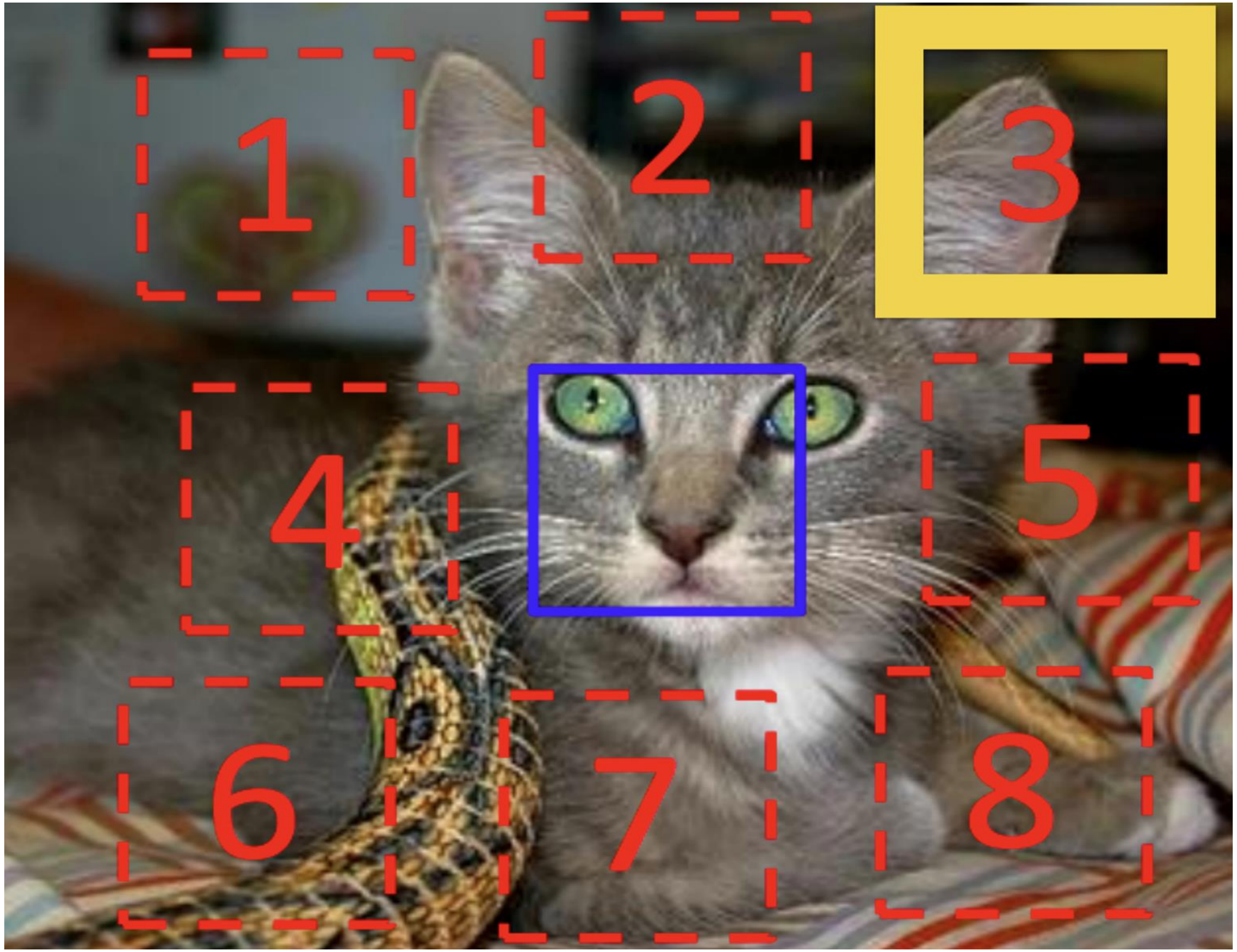
Context Encoders

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%	-
Doersch *et al.* [7]	context	4 weeks	55.3%	**46.6%**	-
Wang *et al.* [39]	motion	1 week	**58.4%**	44.0%	-
Ours	context	14 hours	56.5%	44.5%	**29.7%**

Table 2: Quantitative comparison for classification, detection and semantic segmentation. Classification and Fast-RCNN Detection results are on the PASCAL VOC 2007 test set. Semantic segmentation results are on the PASCAL VOC 2012 validation set from the FCN evaluation described in Section 5.2.3, using the additional training data from [18], and removing overlapping images from the validation set [28].

Pathak et al 2016

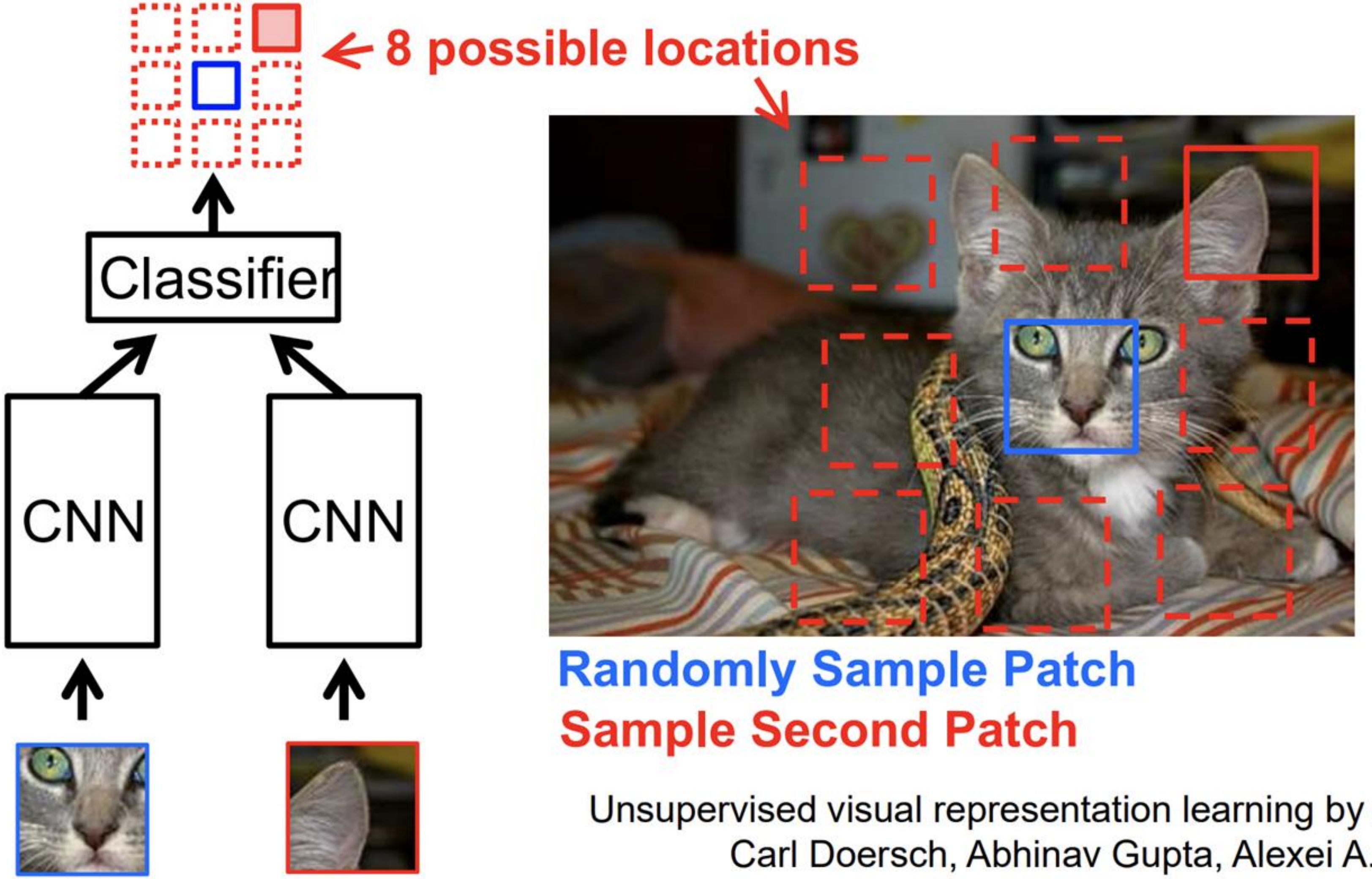
Relative Position of Image Patches



Doersch, Gupta, Efros

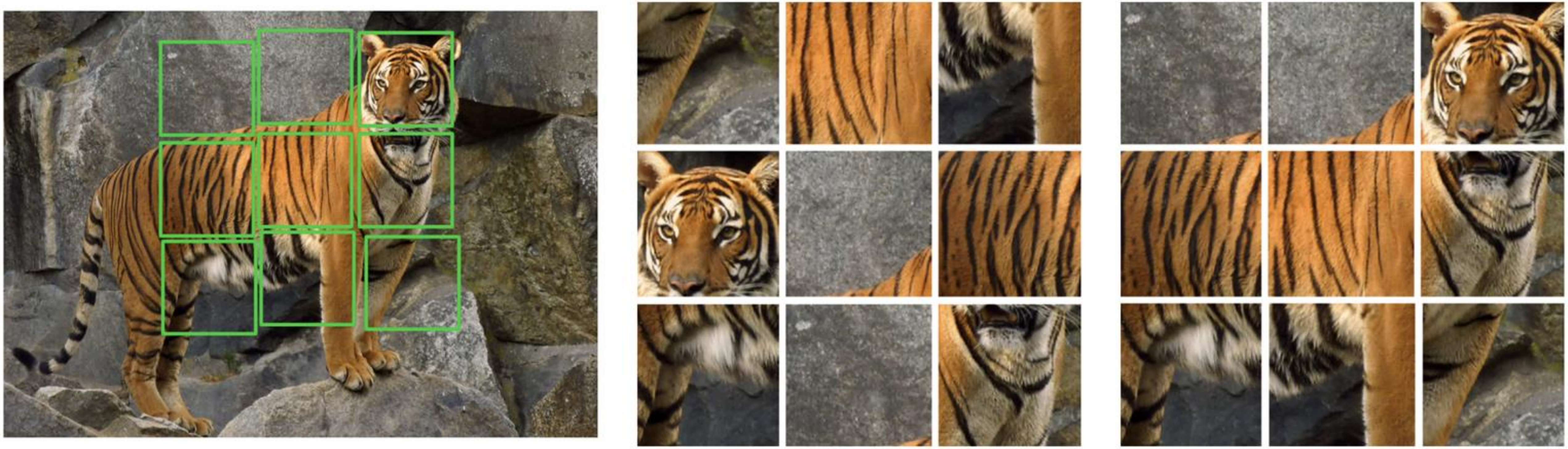
Slide: Zisserman

Relative Position of Image Patches

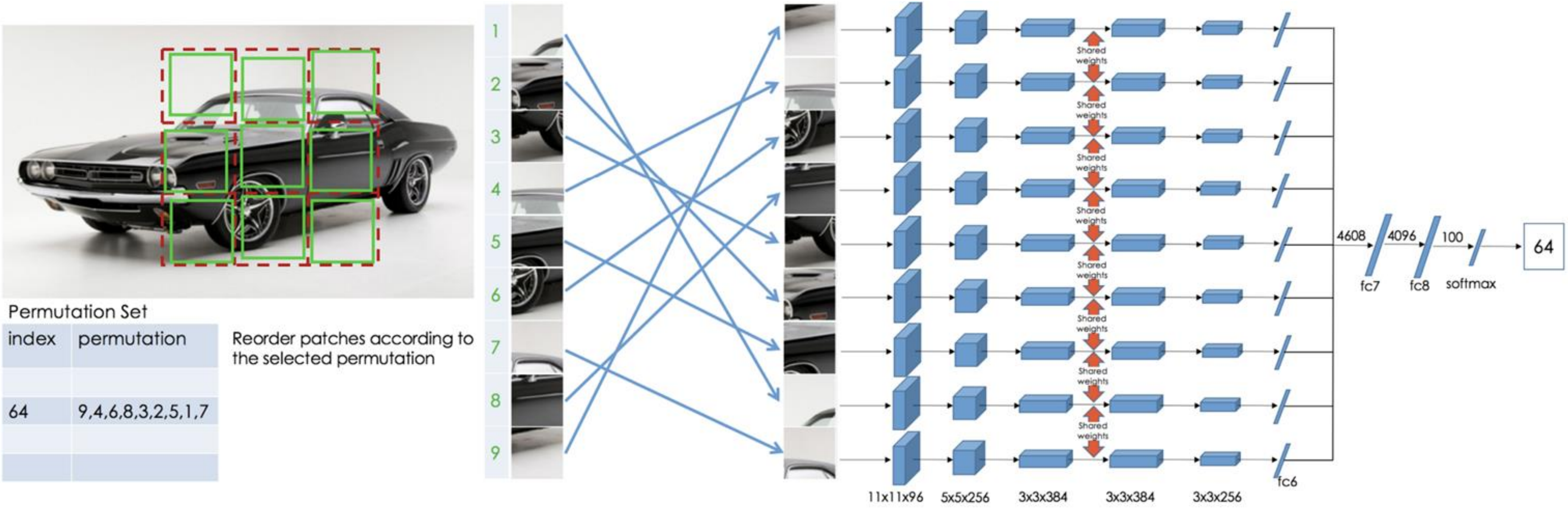


Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

Solving Jigsaw Puzzles



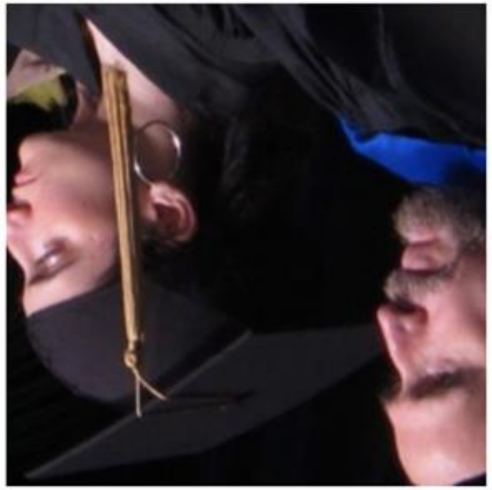
Solving Jigsaw Puzzles



Rotation

90° rotation

270° rotation



180° rotation

0° rotation

270° rotation

Rotation

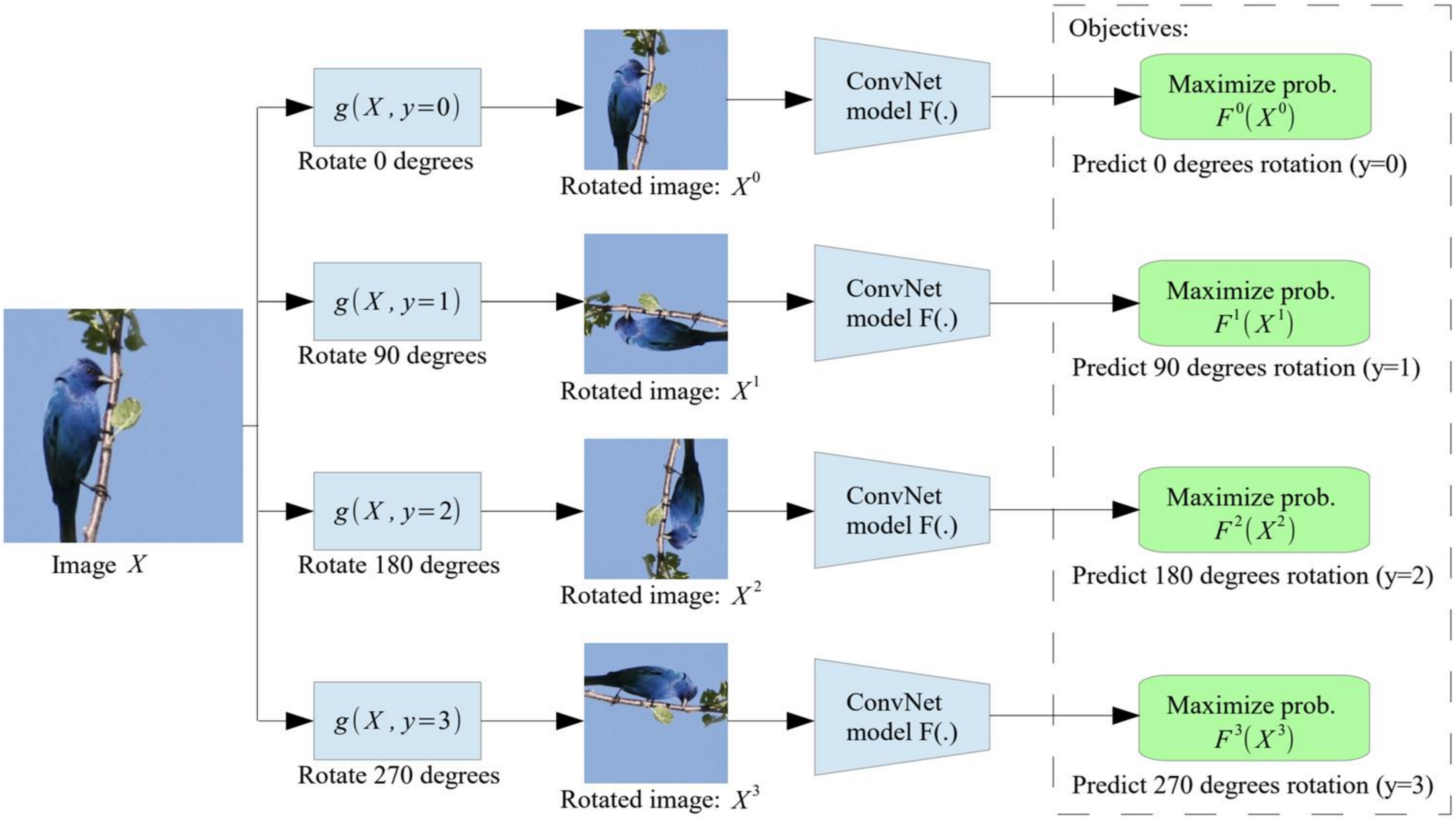


Image X
$g(X, y=0)$
Rotate 0 degrees
$g(X, y=1)$
Rotate 90 degrees
$g(X, y=2)$
Rotate 180 degrees
$g(X, y=3)$
Rotate 270 degrees
Rotated image: X^0
Rotated image: X^1
Rotated image: X^2
Rotated image: X^3
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
Objectives:
Maximize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

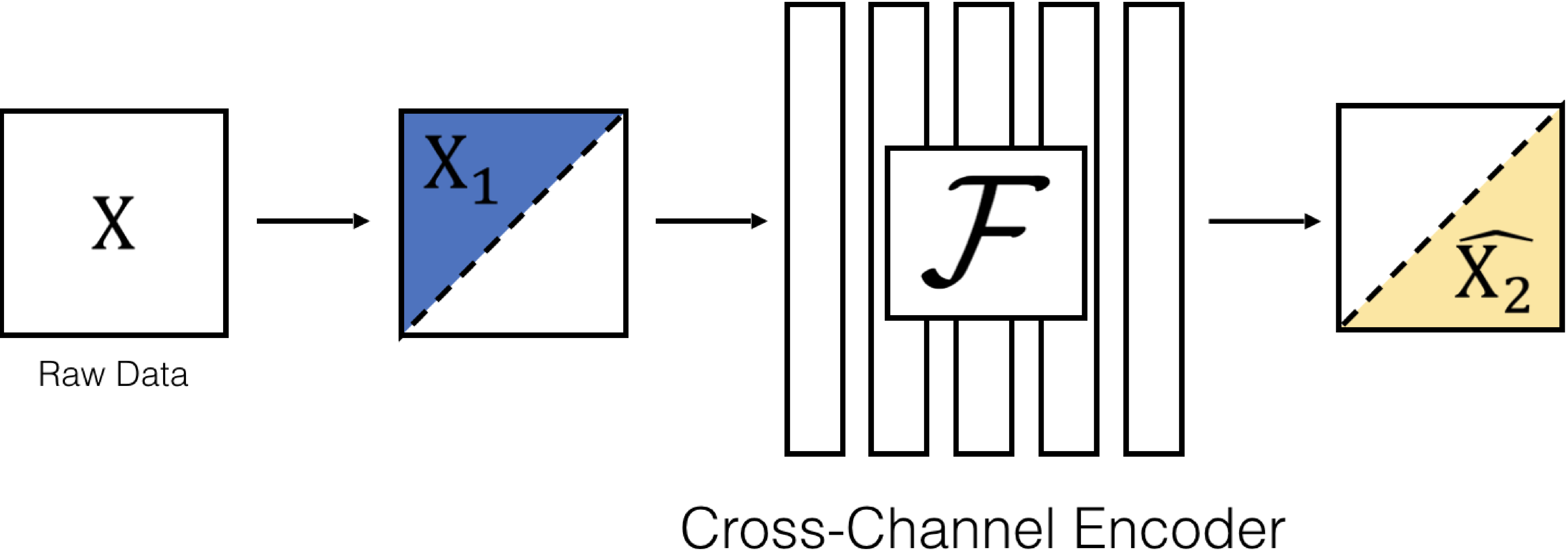
Rotation

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

Rotation

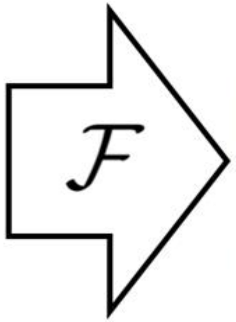
Method	Conv1	Conv2	Conv3	Conv4	Conv5
ImageNet labels	19.3	36.3	44.2	48.3	50.5
Random	11.6	17.1	16.9	16.3	14.1
Random rescaled Krähenbühl et al. (2015)	17.5	23.0	24.5	23.2	20.6
Context (Doersch et al., 2015)	16.2	23.3	30.2	31.7	29.6
Context Encoders (Pathak et al., 2016b)	14.1	20.7	21.0	19.8	15.5
Colorization (Zhang et al., 2016a)	12.5	24.5	30.4	31.5	30.3
Jigsaw Puzzles (Noroozi & Favaro, 2016)	18.2	28.8	34.0	33.9	27.1
BIGAN (Donahue et al., 2016)	17.7	24.5	31.0	29.9	28.0
Split-Brain (Zhang et al., 2016b)	17.7	29.3	35.4	35.2	32.8
Counting (Noroozi et al., 2017)	18.0	30.6	34.3	32.5	25.7
(Ours) RotNet	**18.8**	**31.7**	**38.7**	**38.2**	**36.5**

Predicting one view from another



Slide: Richard Zhang

Predicting one view from another



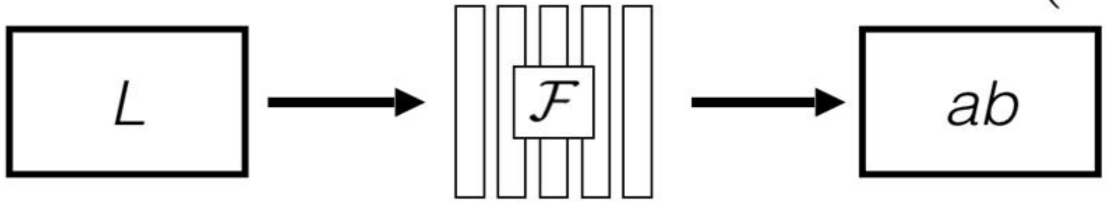
L channel
$\mathbb{R}^{H\times W\times 1}$

Concatenate (*L*,*ab*) channels

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

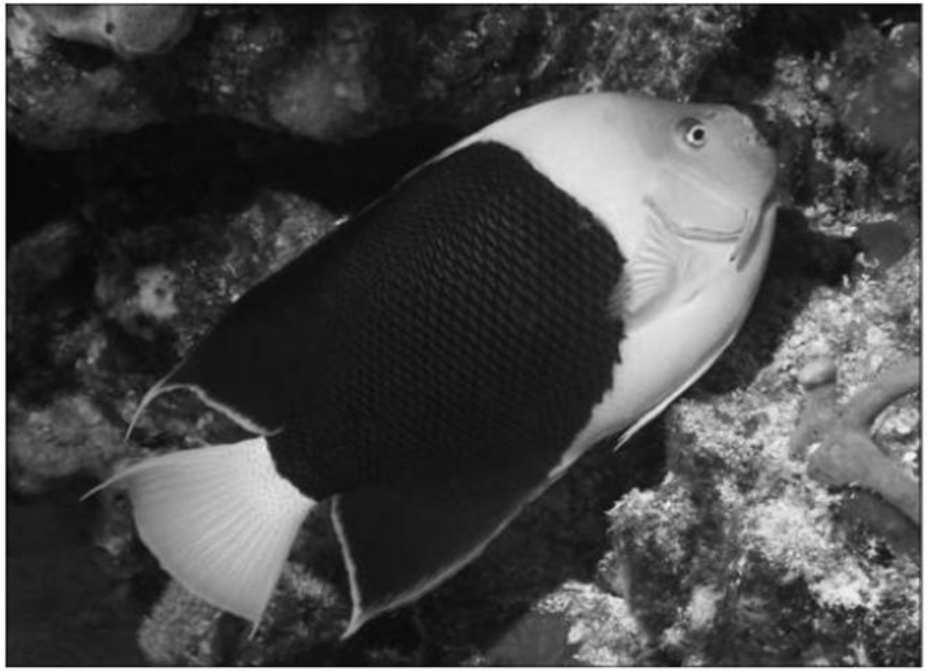
Color information: *ab* channels

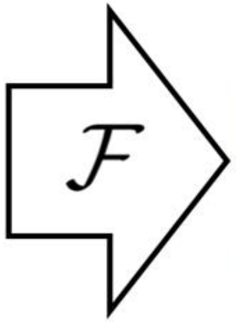
$$\widehat{\mathbf{Y}} \in \mathbb{R}^{H\times W\times 2}$$



Slide: Richard Zhang

Predicting one view from another





Concatenate (L,ab) channels

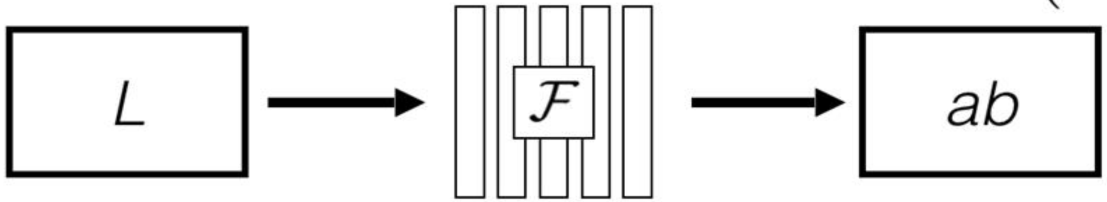
$$(\mathbf{X}, \widehat{\mathbf{Y}})$$

Grayscale image: L channel

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

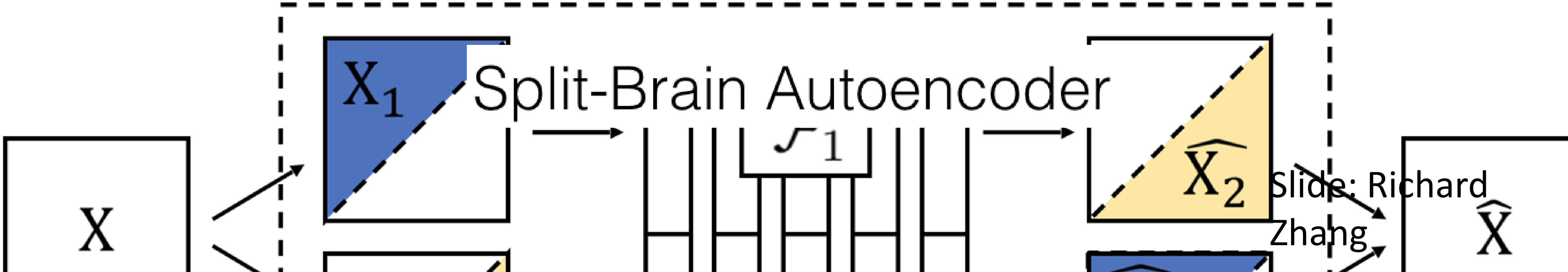
Concatenate (L,ab) channels

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



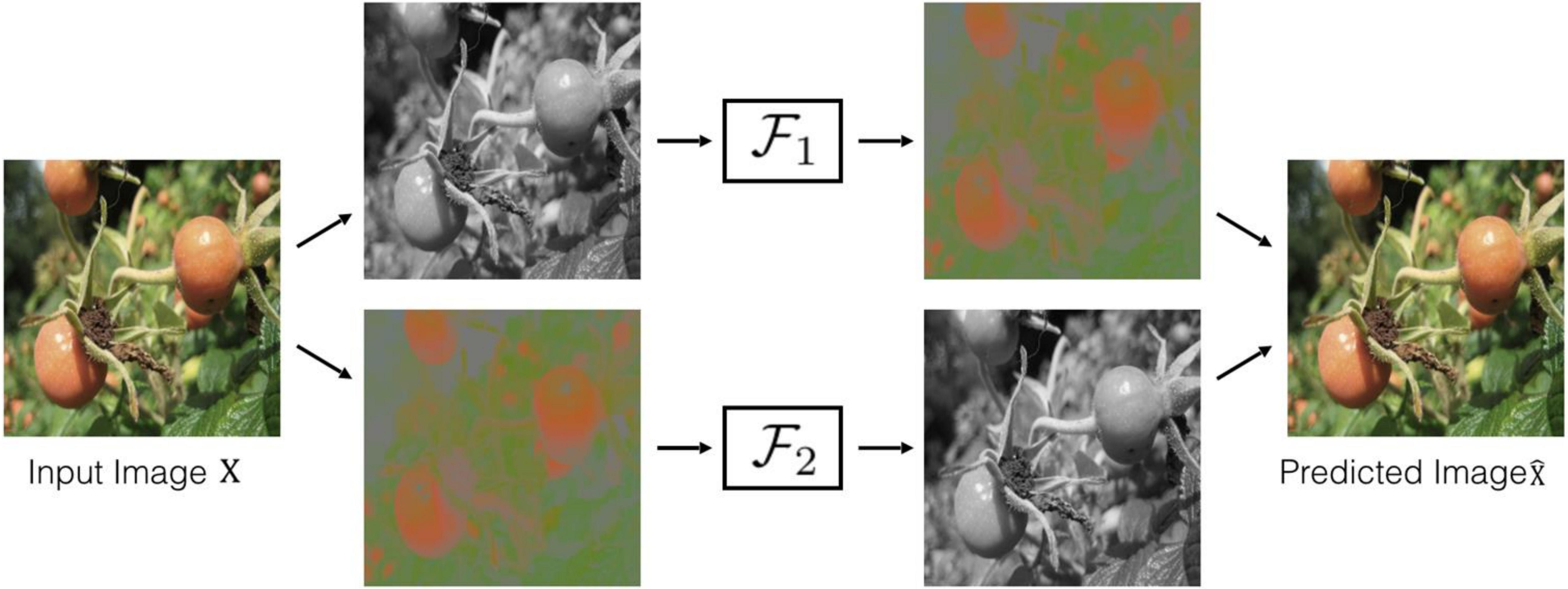
Slide: Richard Zhang

Predicting one view from another



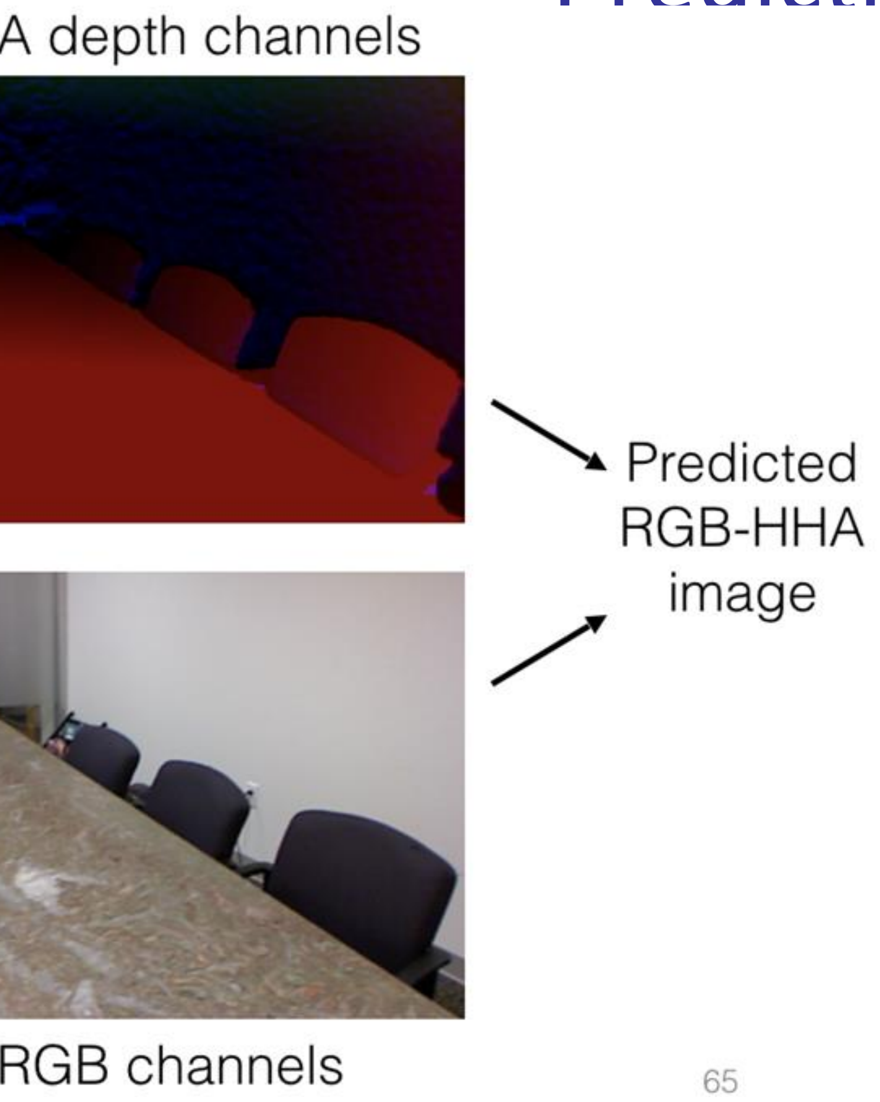
Slide: Richard Zhang

Predicting one view from another



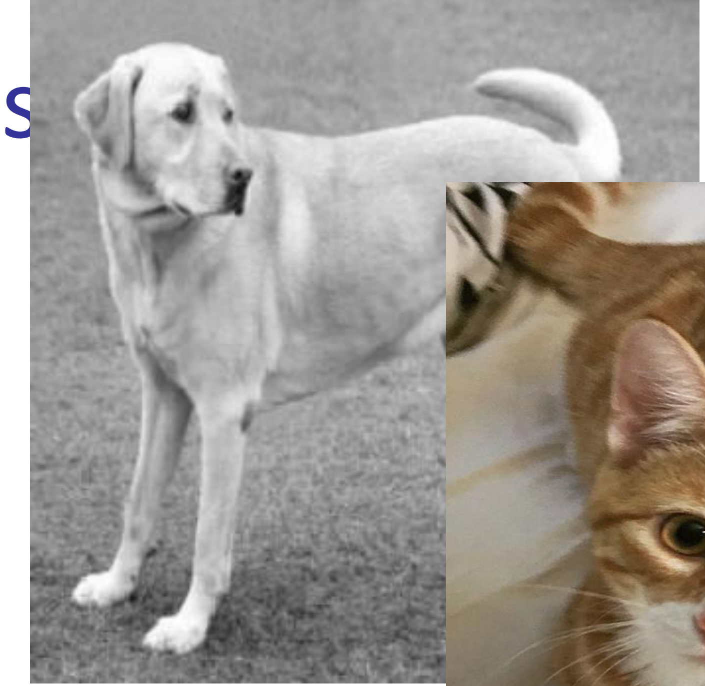
Slide: Richard Zhang

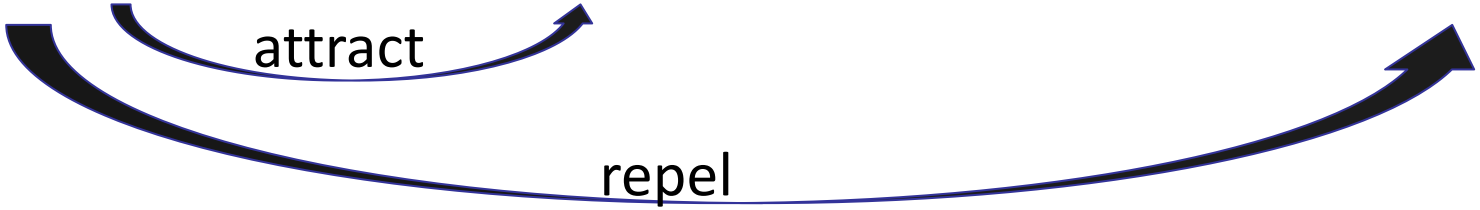
Predicting one view from another



Slide: Richard Zhang

Instance Dis





Instance Discrimination

attract

repel

1. MoCo
2. SimCLR

Formulation of Contrastive Learning

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

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

- Cross entropy loss for a N-way softmax classifier!
- i.e., learn to find the positive sample from the N samples

SimCLR: A Simple Framework for Contrastive Learning

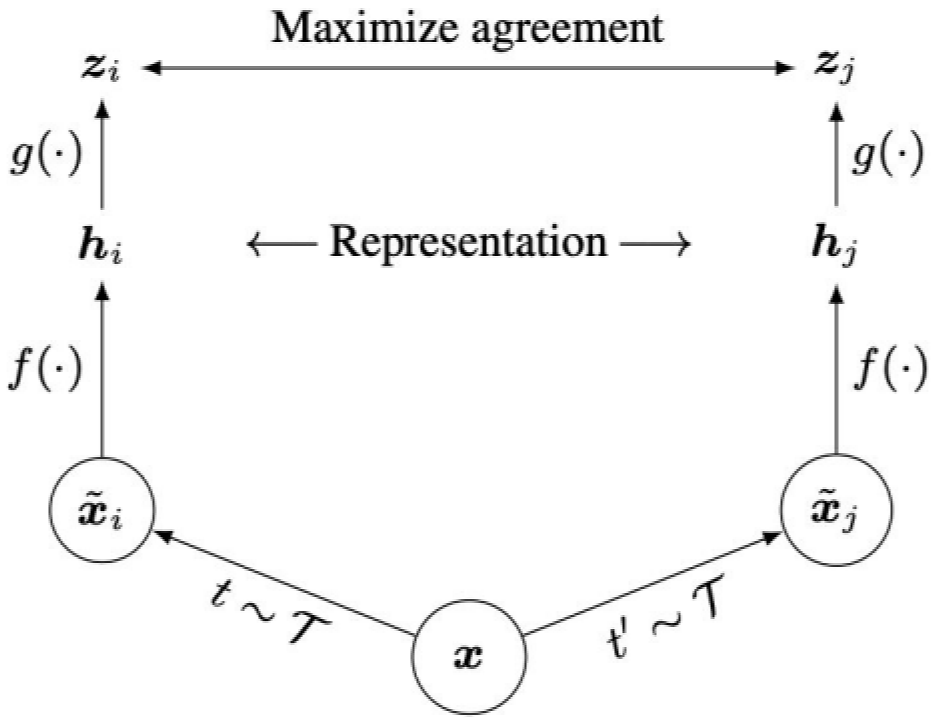
Cosine similarity as the score function:

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

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

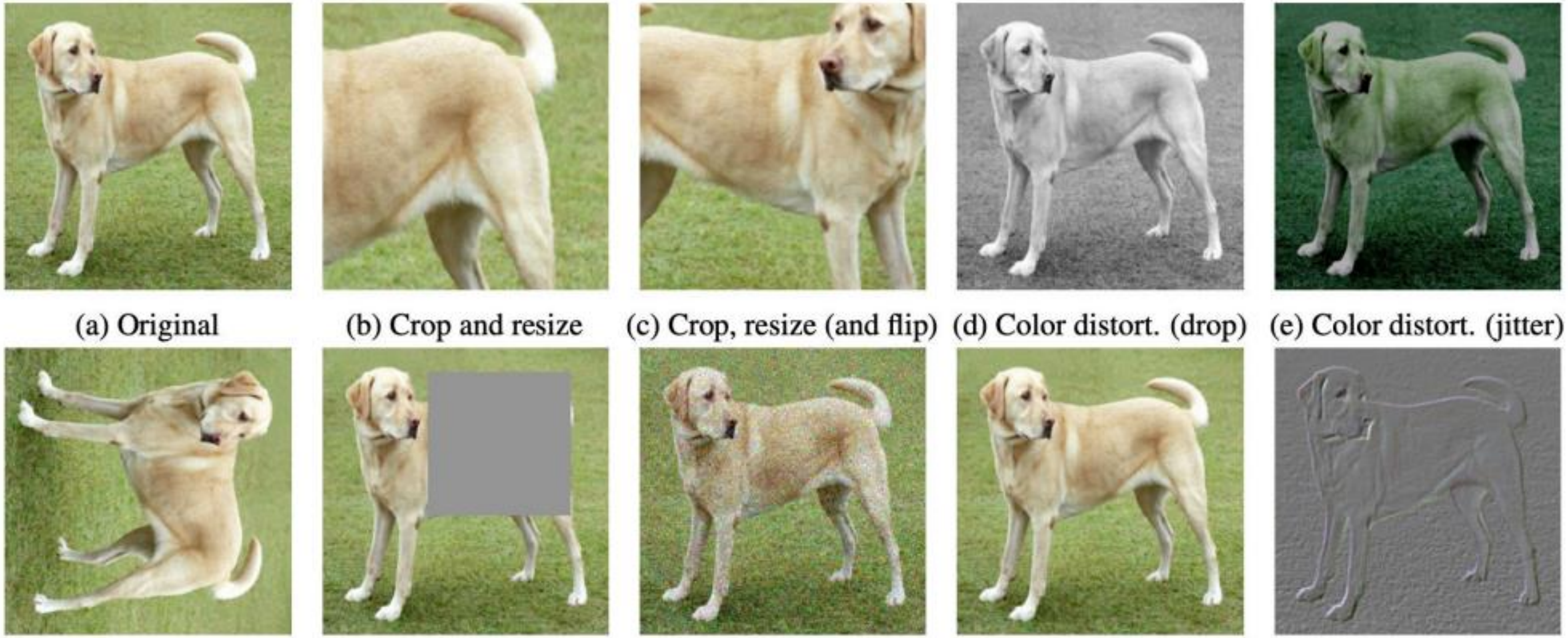
Generate positive samples through data augmentation:

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



"A Simple Framework for Contrastive Learning of Visual Representations", Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton, ICML'20

SimCLR: generating positive samples from data augmentation



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

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

SimCLR

Algorithm 1 SimCLR's main learning algorithm.

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

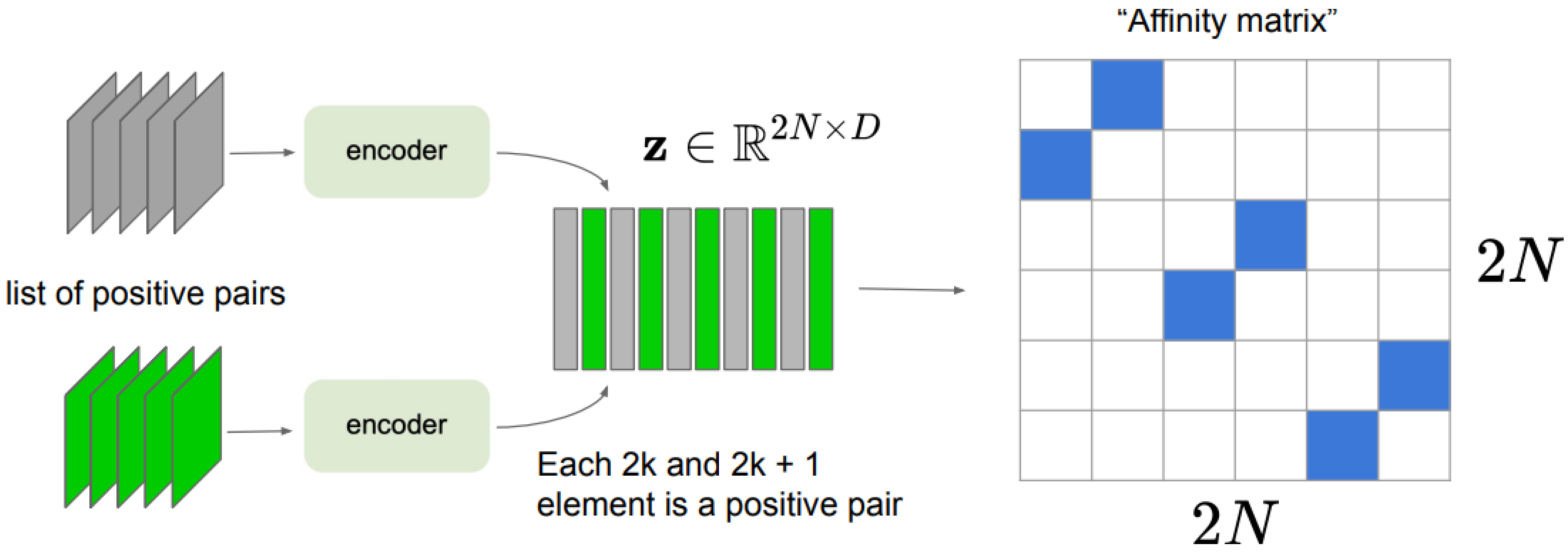
Generate a positive pair by sampling data augmentation functions

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

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

SimCLR: mini-batch training

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



Effect of Batch Size

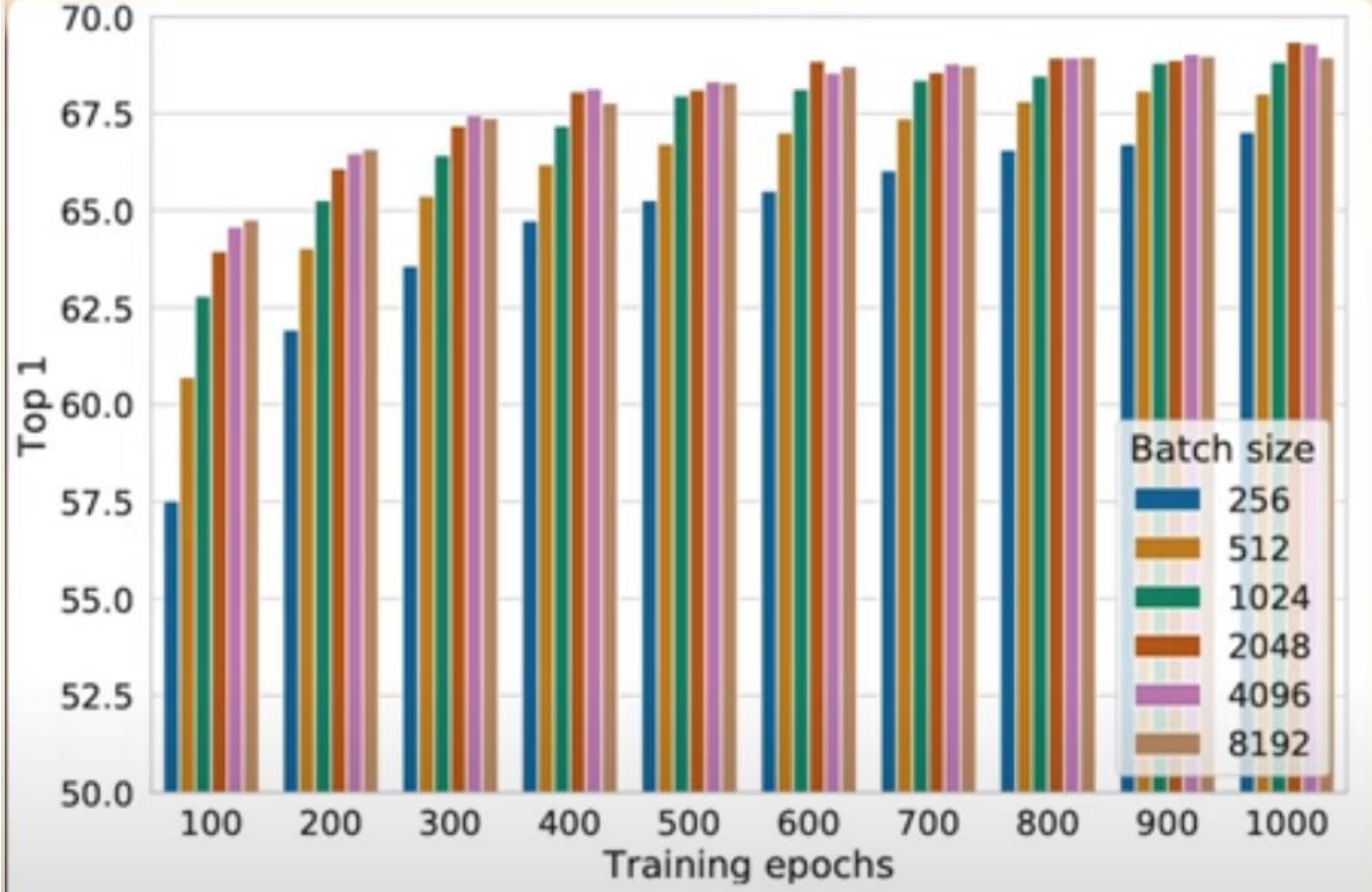


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

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.