# CS294-158 Deep Unsupervised Learning

## Lecture 7 Self-Supervised Learning

Pieter Abbeel, Xi (Peter) Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

UC Berkeley

# Course so far...

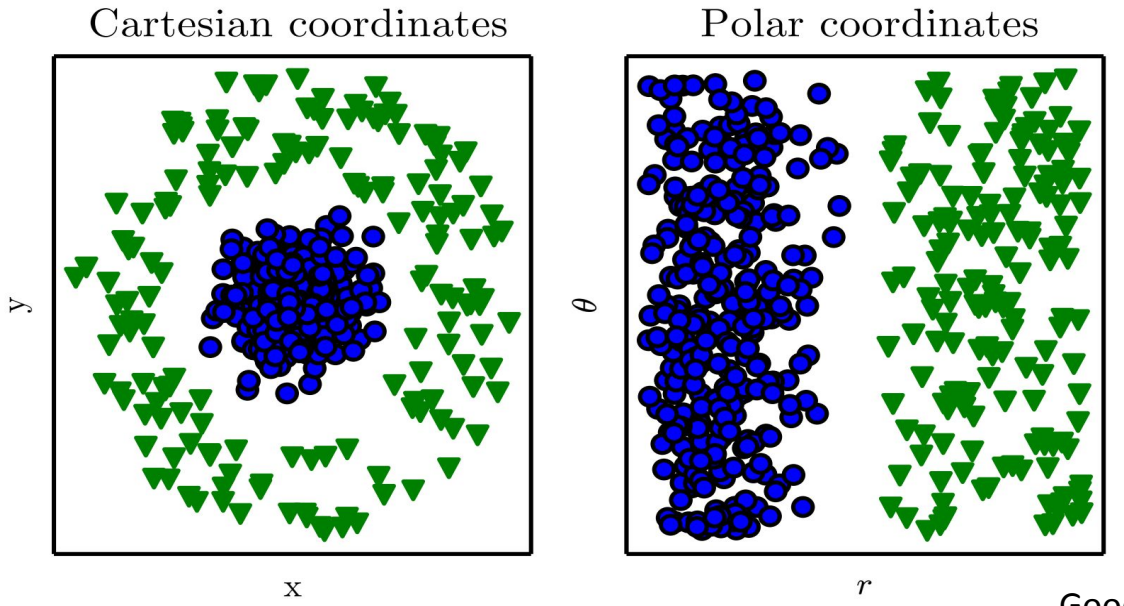
- Density modeling
  - Autoregressive, Normalizing Flows, Variational Inference
- Implicit models
  - Generative Adversarial Networks, Energy based models
- Applications of generative modeling

# Today…

- How do learn rich and useful features from raw unlabeled data that can be useful for several downstream tasks?
- What are the various pretext (proxy) tasks that can be used to learn representations from unlabeled data?
- How can we improve data-efficiency and performance of downstream tasks with a good pre-trained network?

# Representation Learning

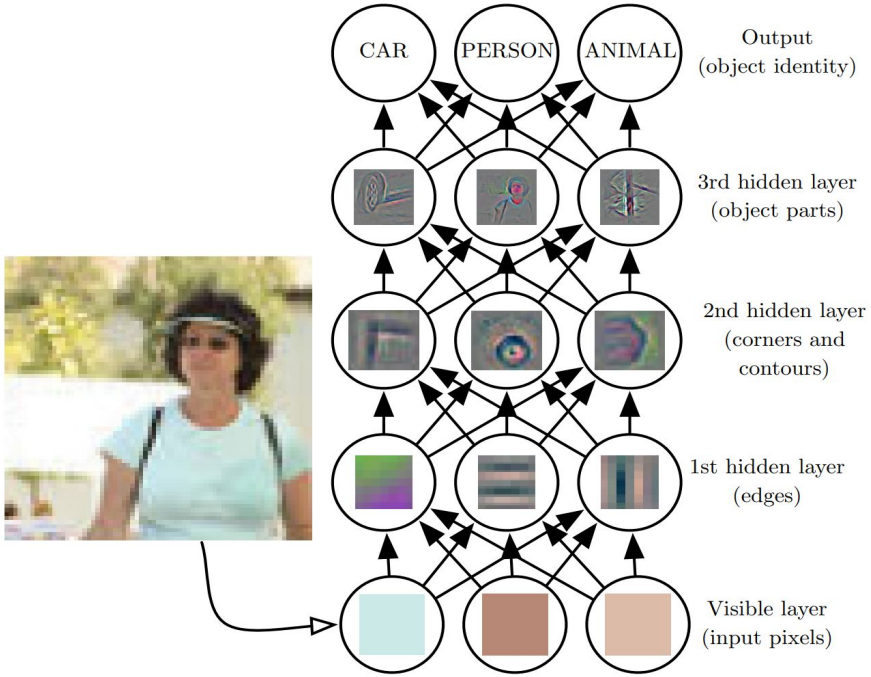
## Representations Matter




Goodfellow

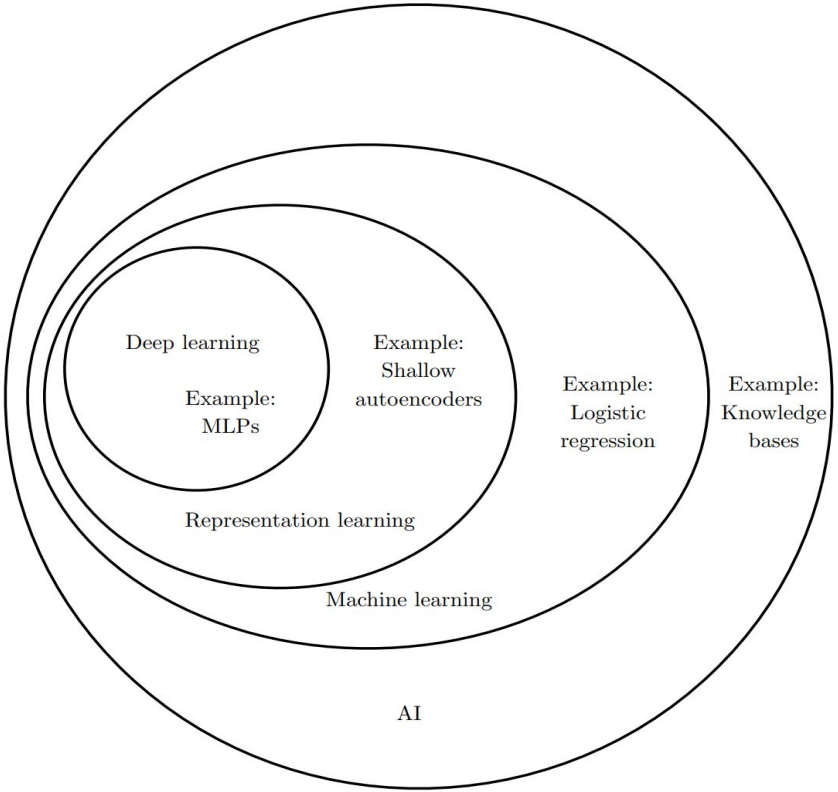
# Depth refines representations

## Depth: Repeated Composition



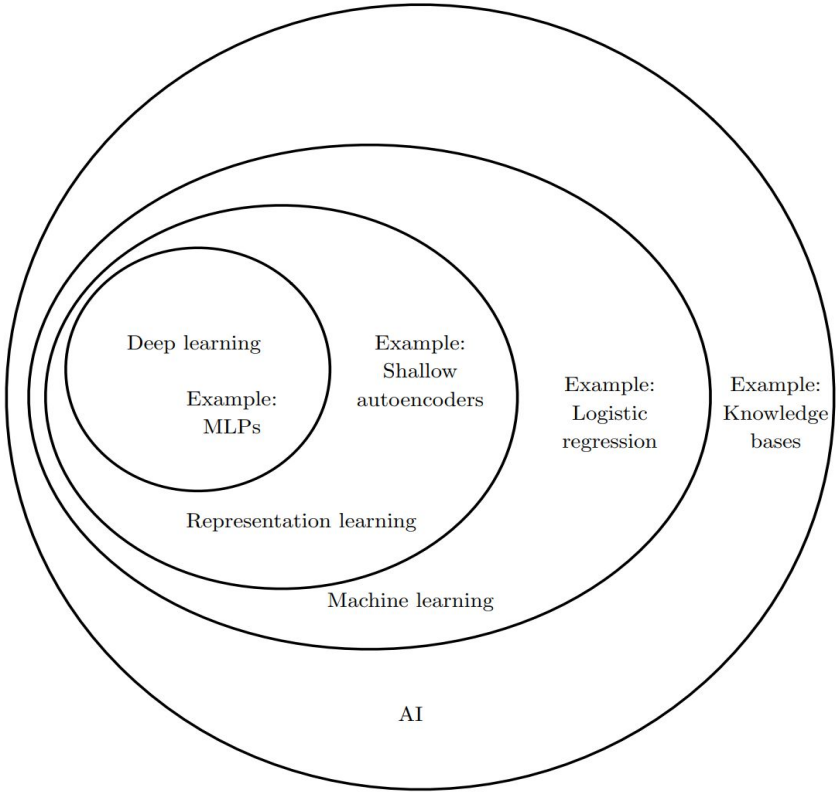

Goodfellow

# Deep Learning is one way to learn features




Goodfellow

# Deep Learning is one way to learn features

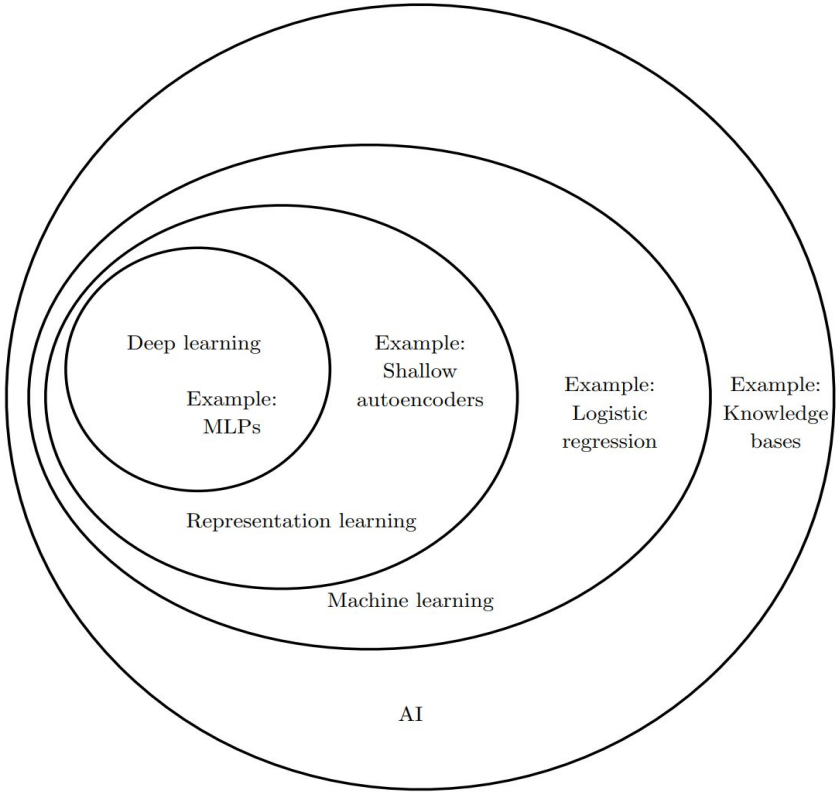



Deep Unsupervised Learning:

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

Goodfellow

# 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

# Why Self-Supervised Learning?

Give a robot a label and you feed it for a second, teach a robot to label and you feed it for a lifetime - Pierre Sermanet

# 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.

# 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

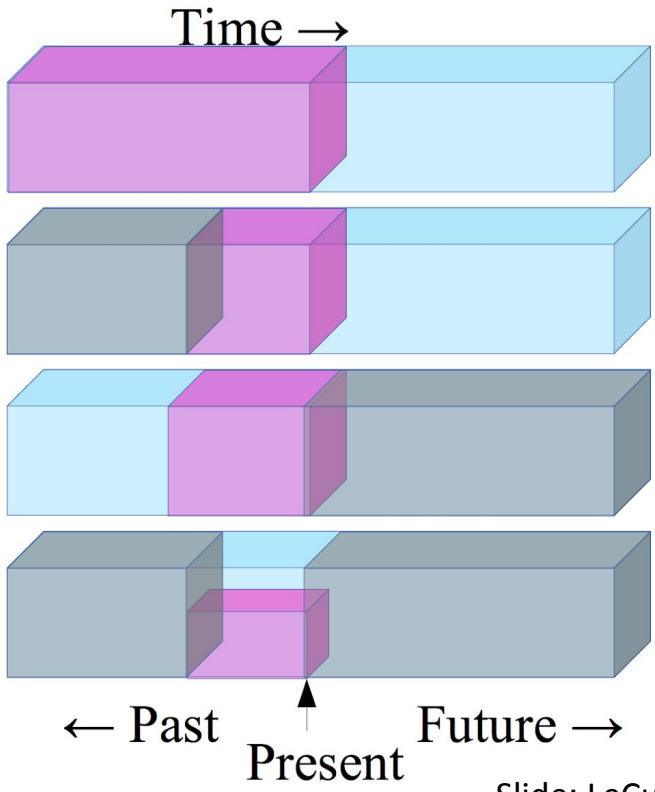
# Motivation

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

Yann LeCun's cake

# Motivation

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

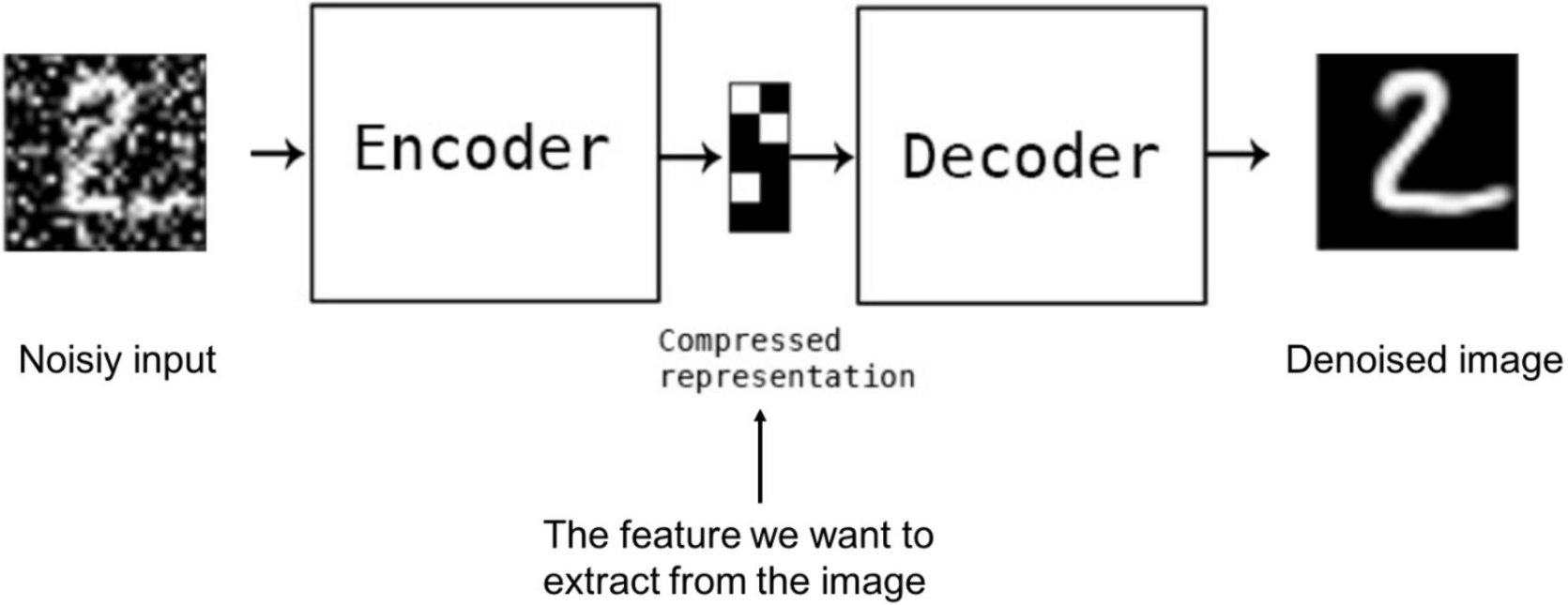



Slide: LeCun

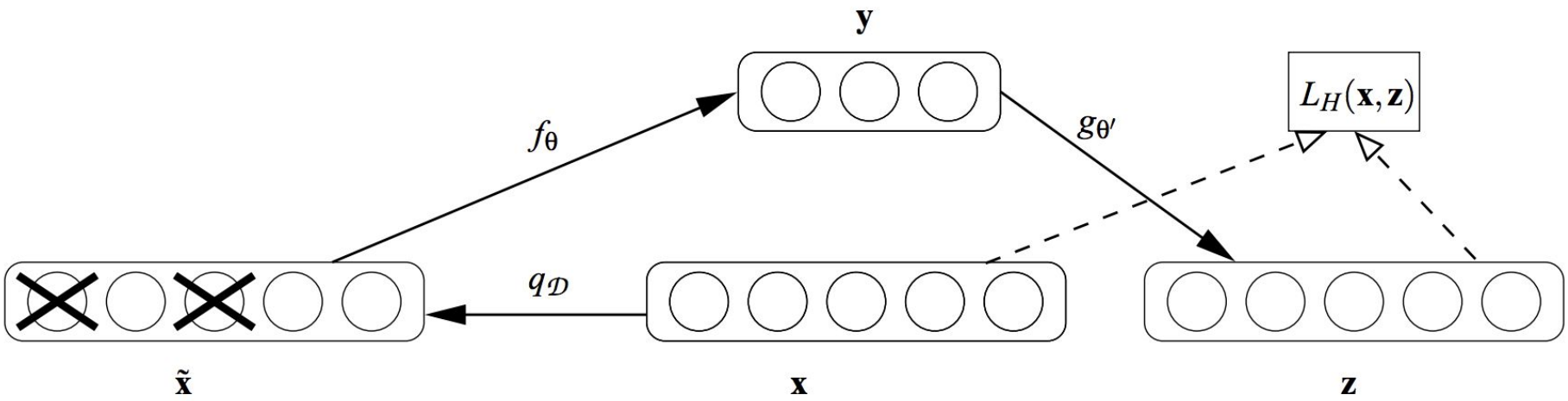
# Cognitive Principles

- 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



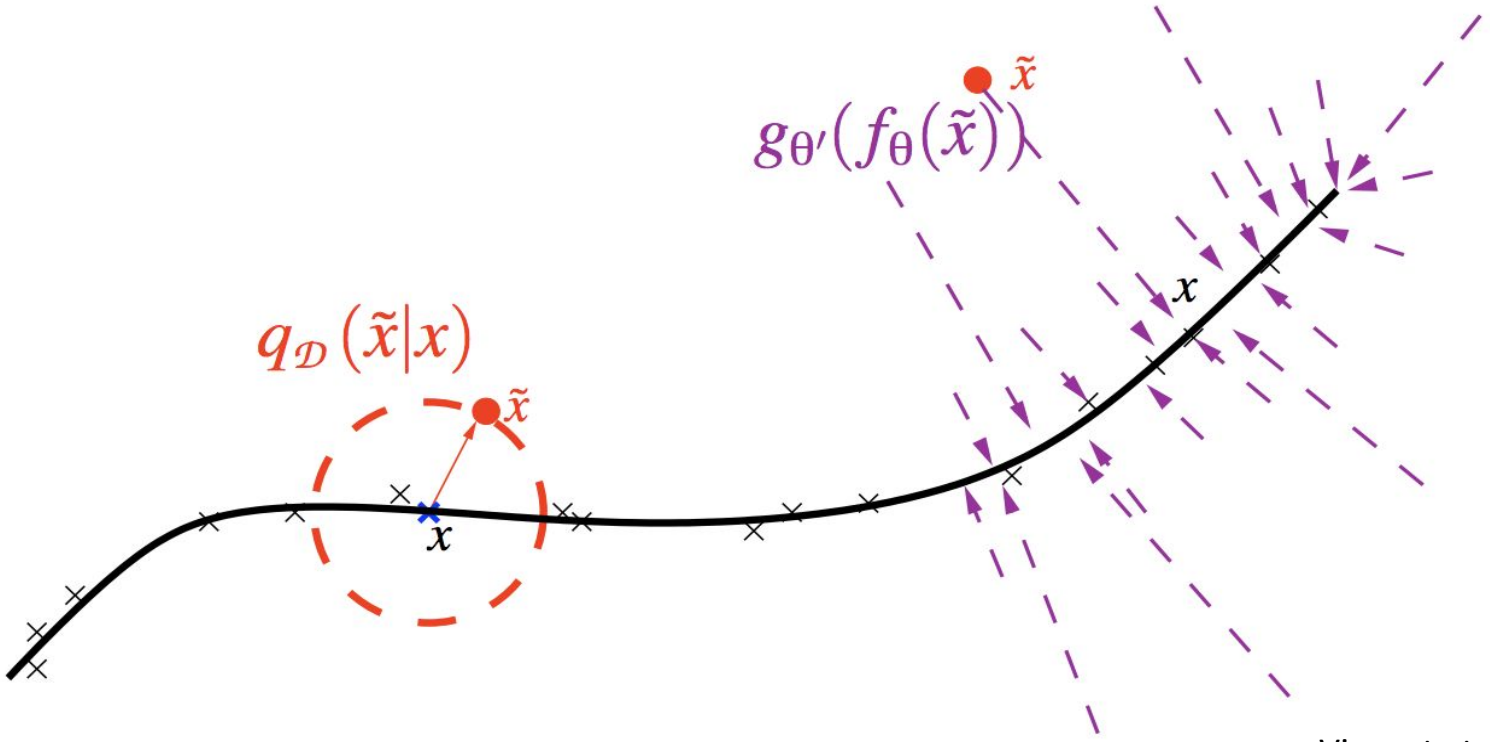

Vincent et al 2010

# Denoising Autoencoder

- Additive isotropic *Gaussian noise* (GS): $\tilde{\mathbf{x}}|\mathbf{x} \sim \mathcal{N}(\mathbf{x}, \sigma^2 I)$;
- *Masking noise* (MN): a fraction $\nu$ of the elements of $\mathbf{x}$ (chosen at random for each example) is forced to 0;
- *Salt-and-pepper noise* (SP): a fraction $\nu$ 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.

Vincent et al 2010

# Denoising Autoencoder




Vincent et al 2010

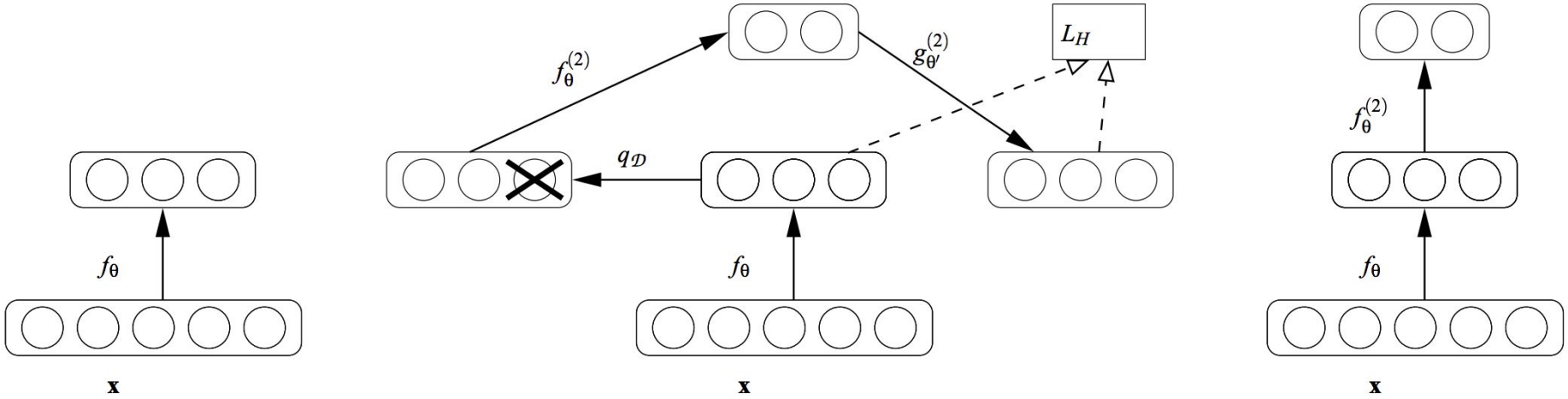
# Emphasizing corrupted dimensions

$$L_{2,\alpha}(\mathbf{x},\mathbf{z}) = \alpha\left(\sum_{j\in\mathcal{J}(\tilde{\mathbf{x}})}(\mathbf{x}_j-\mathbf{z}_j)^2\right)+\beta\left(\sum_{j\notin\mathcal{J}(\tilde{\mathbf{x}})}(\mathbf{x}_j-\mathbf{z}_j)^2\right)$$

$$\begin{aligned} L_{\mathrm{IH},\alpha}(\mathbf{x},\mathbf{z}) \quad = \quad & \alpha\left(-\sum_{j\in\mathcal{J}(\tilde{\mathbf{x}})}[\mathbf{x}_j\log\mathbf{z}_j+(1-\mathbf{x}_j)\log(1-\mathbf{z}_j)]\right) \\ & +\beta\left(-\sum_{j\notin\mathcal{J}(\tilde{\mathbf{x}})}[\mathbf{x}_j\log\mathbf{z}_j+(1-\mathbf{x}_j)\log(1-\mathbf{z}_j)]\right) \end{aligned}$$

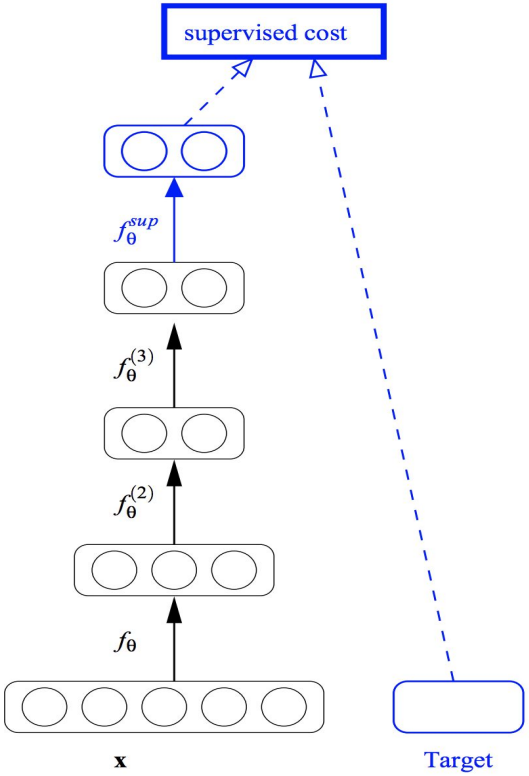
Vincent et al 2010

# Stacked Denoising Autoencoder



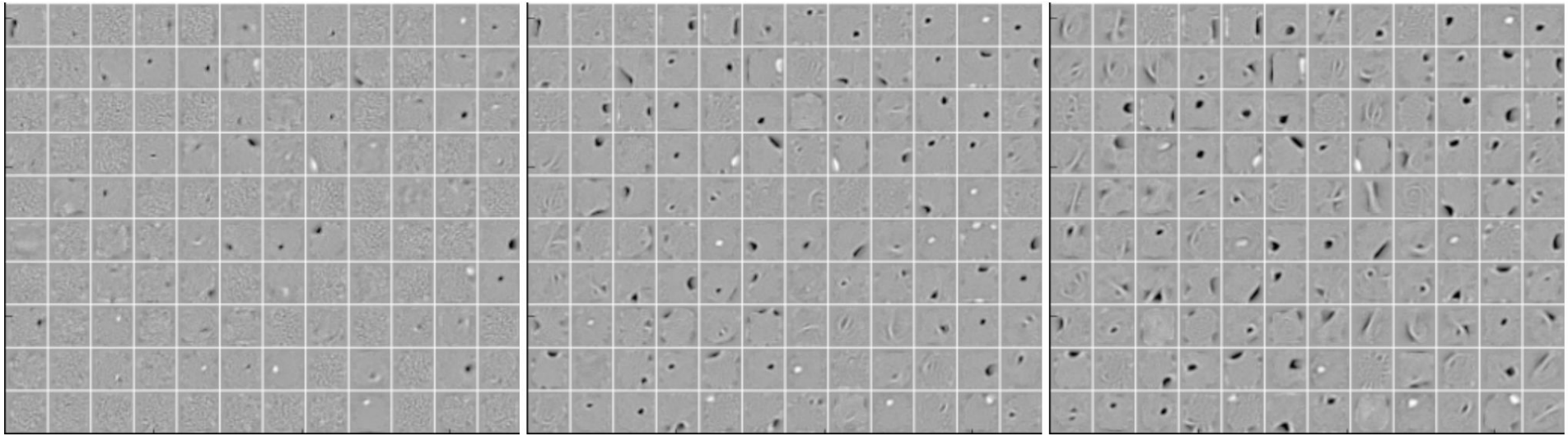

Vincent et al 2010

# Denoising Autoencoder




Vincent et al 2010

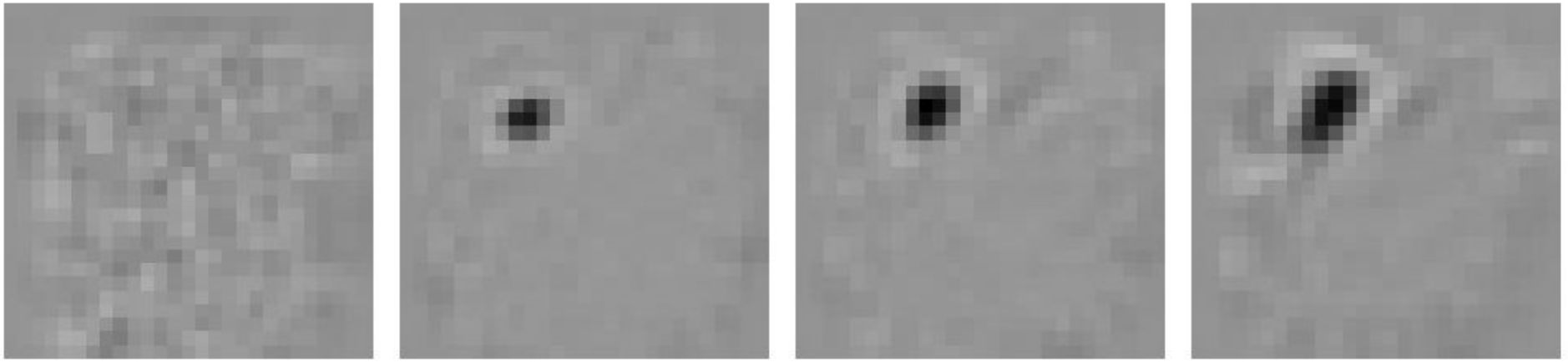
# Denoising Autoencoder



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

Vincent et al 2010

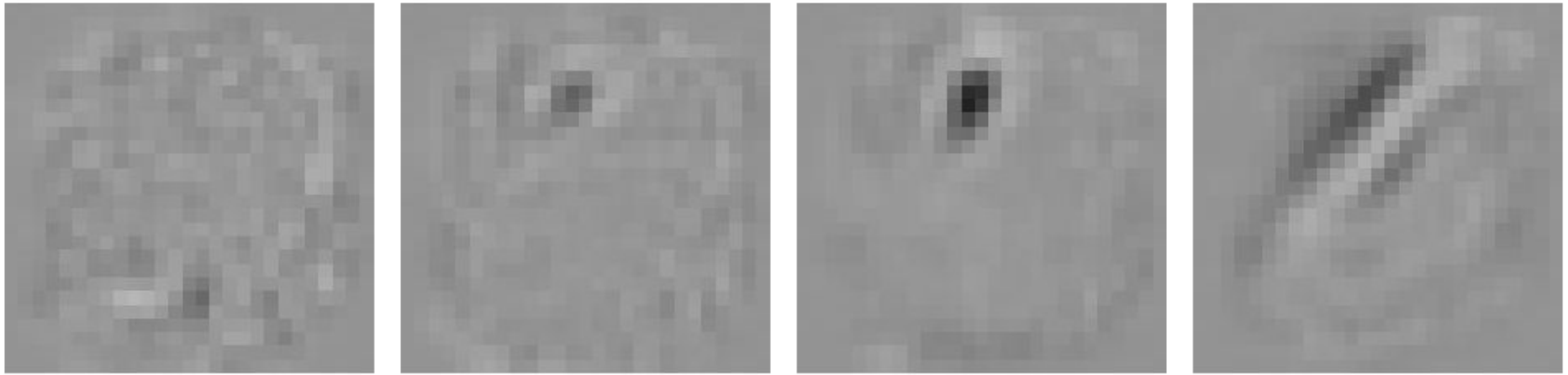
# Denoising Autoencoder



(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

# Denoising Autoencoder

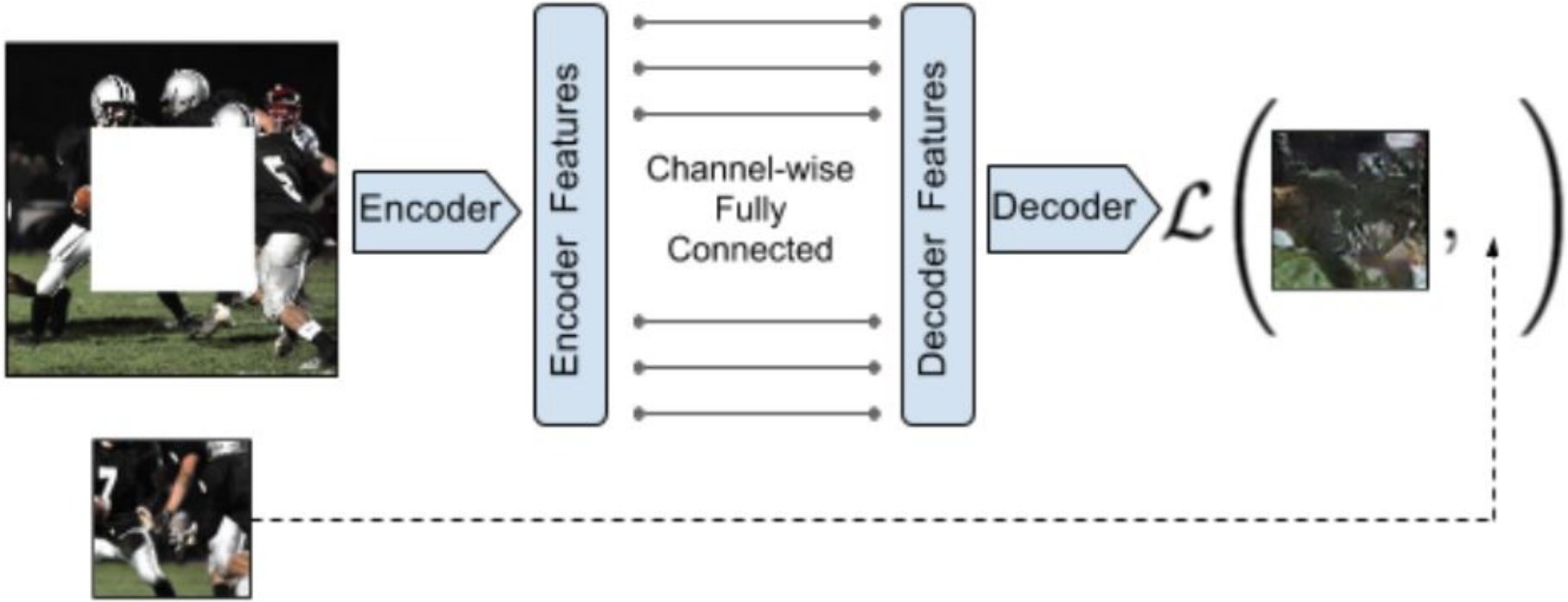
| Dataset | $\text{SVM}_{rbf}$ | $\text{SVM}_{poly}$ | DBN-1 | SAA-3 | DBN-3 | SdA-3 ($\nu$) |
|---|---|---|---|---|---|---|
| *basic* | **3.03±0.15** | 3.69±0.17 | 3.94±0.17 | 3.46±0.16 | 3.11±0.15 | **2.80±0.14** (10%) |
| *rot* | 11.11±0.28 | 15.42±0.32 | 14.69±0.31 | **10.30±0.27** | **10.30±0.27** | **10.29±0.27** (10%) |
| *bg-rand* | 14.58±0.31 | 16.62±0.33 | 9.80±0.26 | 11.28±0.28 | **6.73±0.22** | 10.38±0.27 (40%) |
| *bg-img* | 22.61±0.37 | 24.01±0.37 | **16.15±0.32** | 23.00±0.37 | **16.31±0.32** | **16.68±0.33** (25%) |
| *rot-bg-img* | 55.18±0.44 | 56.41±0.43 | 52.21±0.44 | 51.93±0.44 | 47.39±0.44 | **44.49±0.44** (25%) |
| *rect* | **2.15±0.13** | **2.15±0.13** | 4.71±0.19 | 2.41±0.13 | 2.60±0.14 | **1.99±0.12** (10%) |
| *rect-img* | 24.04±0.37 | 24.05±0.37 | 23.69±0.37 | 24.05±0.37 | 22.50±0.37 | **21.59±0.36** (25%) |
| *convex* | 19.13±0.34 | 19.82±0.35 | 19.92±0.35 | **18.41±0.34** | **18.63±0.34** | **19.06±0.34** (10%) |

Vincent et al 2010

# Predict missing pieces

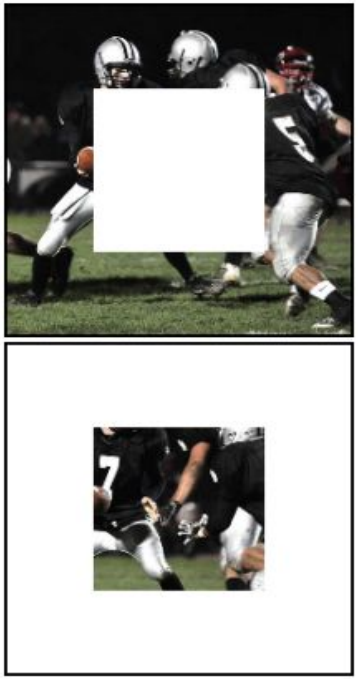
Pathak et al 2016

# Context Encoders

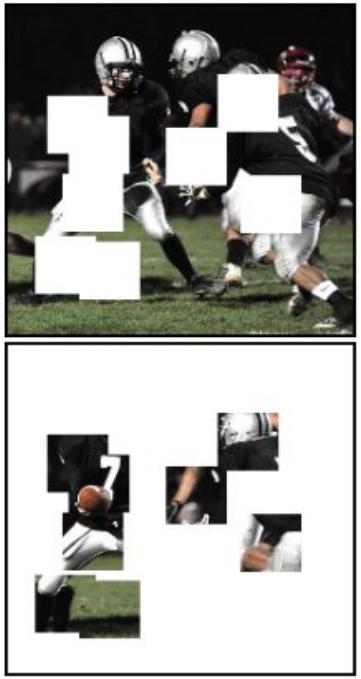



Pathak et al 2016

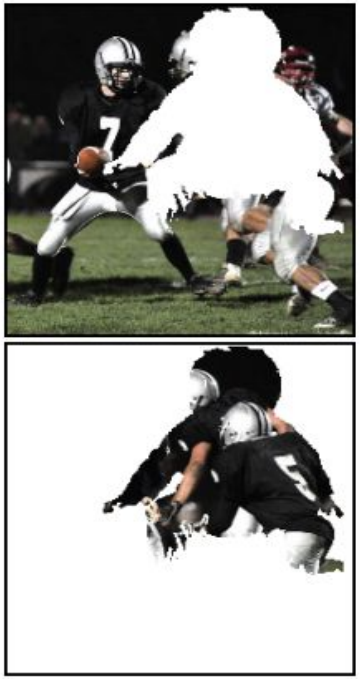
# Context Encoders



(a) Central region



(b) Random block



(c) Random region

Pathak et al 2016

# Context Encoders

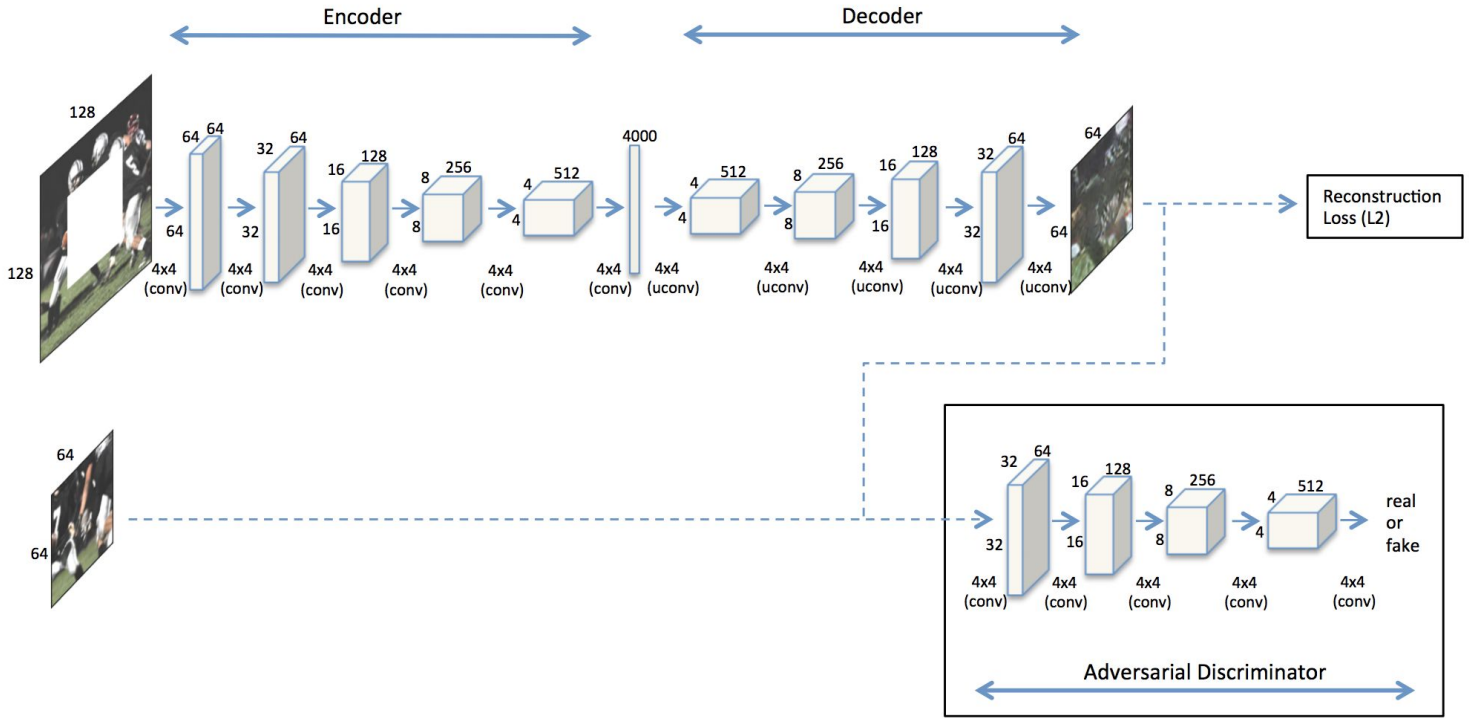
$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2$$

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

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

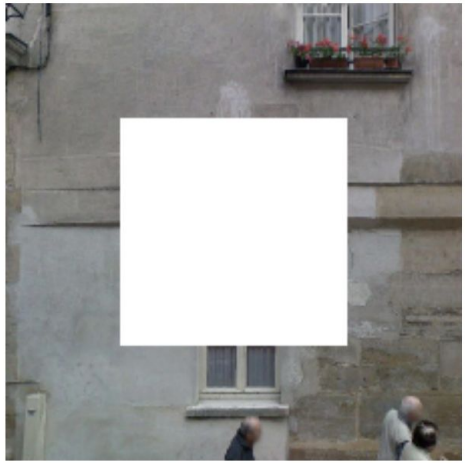
Pathak et al 2016

# Context Encoders



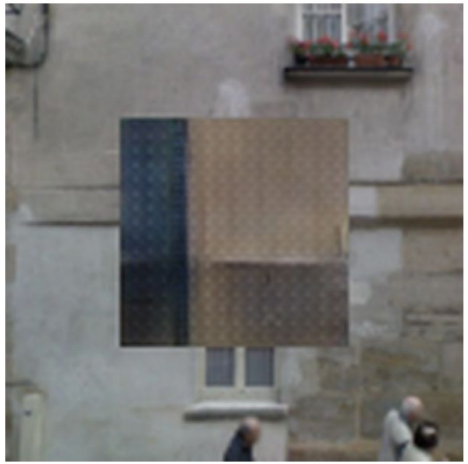

Pathak et al 2016

# Context Encoders

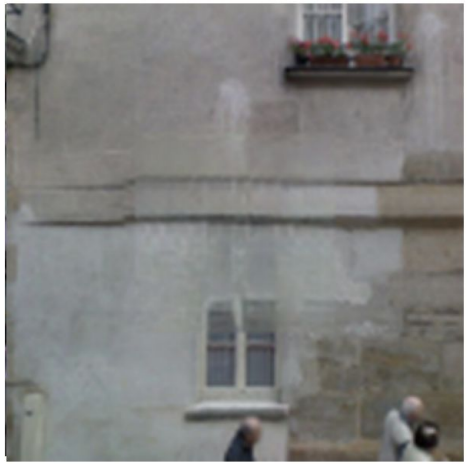


Input Image

L2 Loss



Adversarial Loss



Joint Loss

Pathak et al 2016

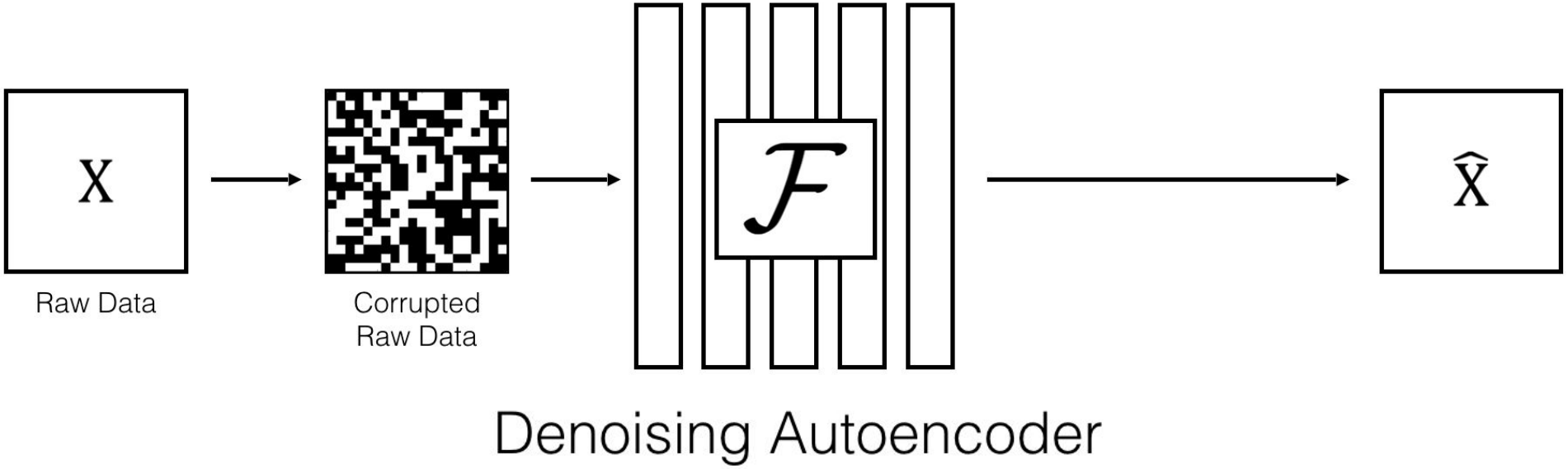
# Context Encoders

| 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].

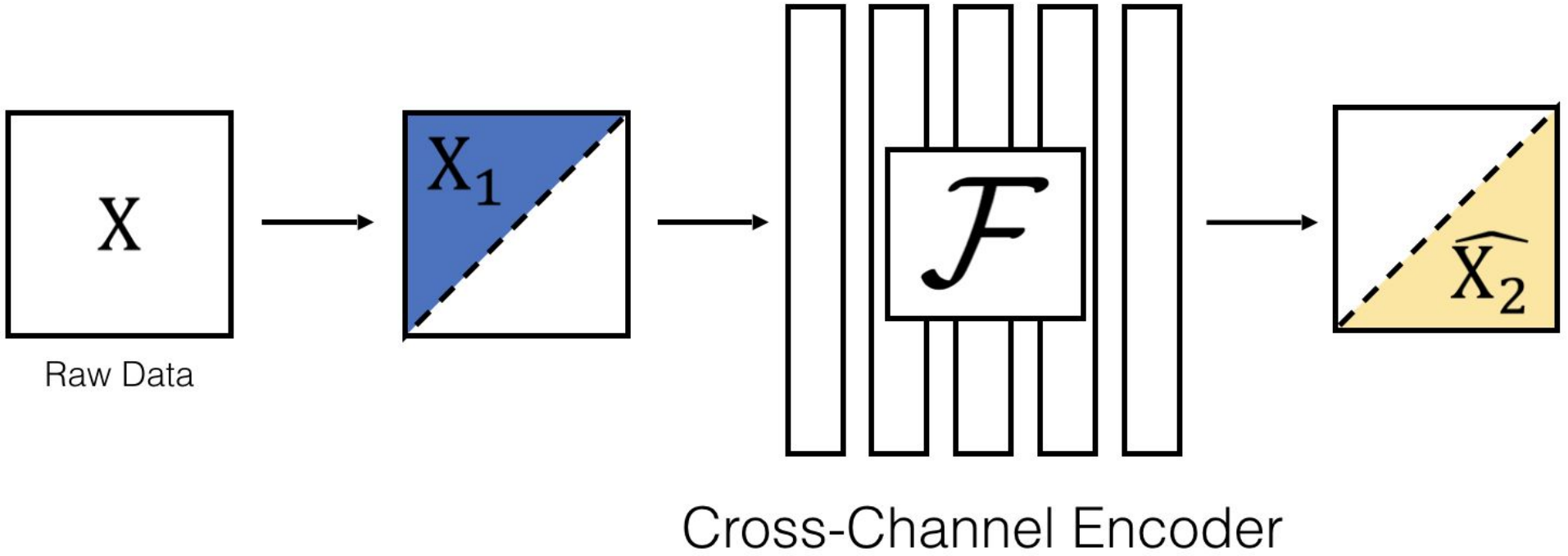
Pathak et al 2016

# Predicting one view from another



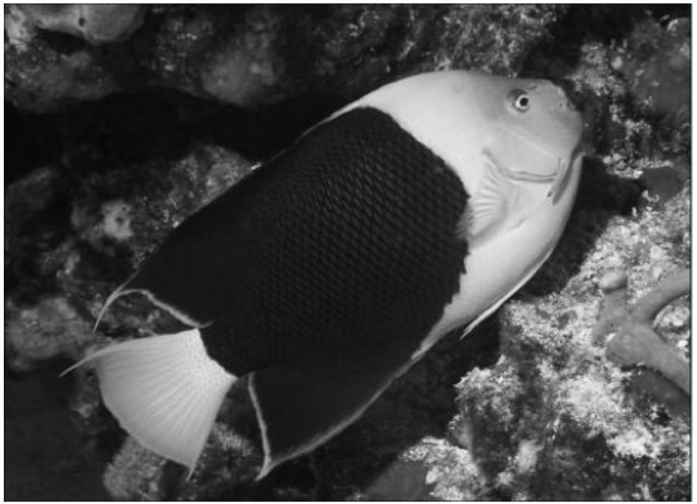

Slide: Richard Zhang

# Predicting one view from another




Slide: Richard Zhang

# Predicting one view from another



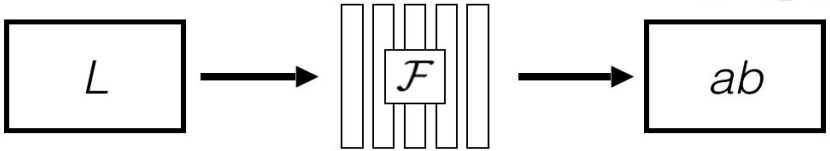
$\mathcal{F}$

Grayscale image: *L* channel

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

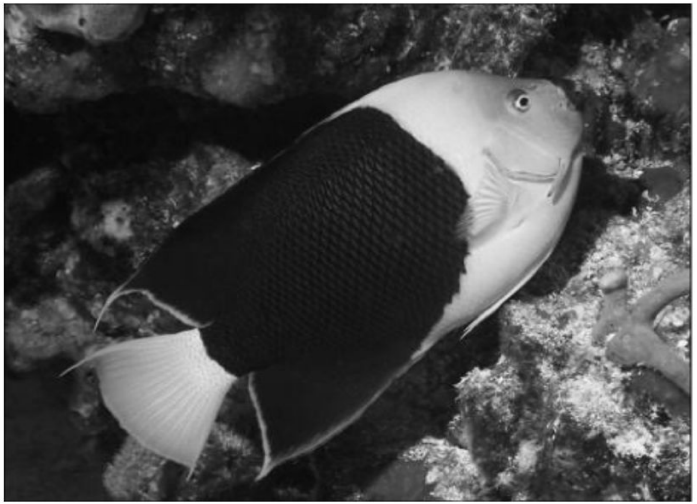
Color information: *ab* channels

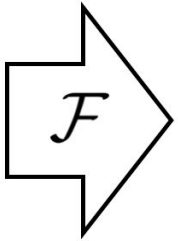
$\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 2}$




Slide: Richard Zhang

# Predicting one view from another



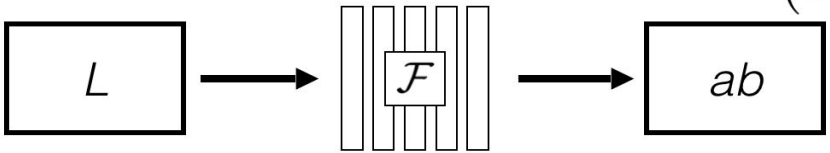



Grayscale image: *L* channel

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

Concatenate (*L*,*ab*) channels

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




Slide: Richard Zhang

# Predicting one view from another

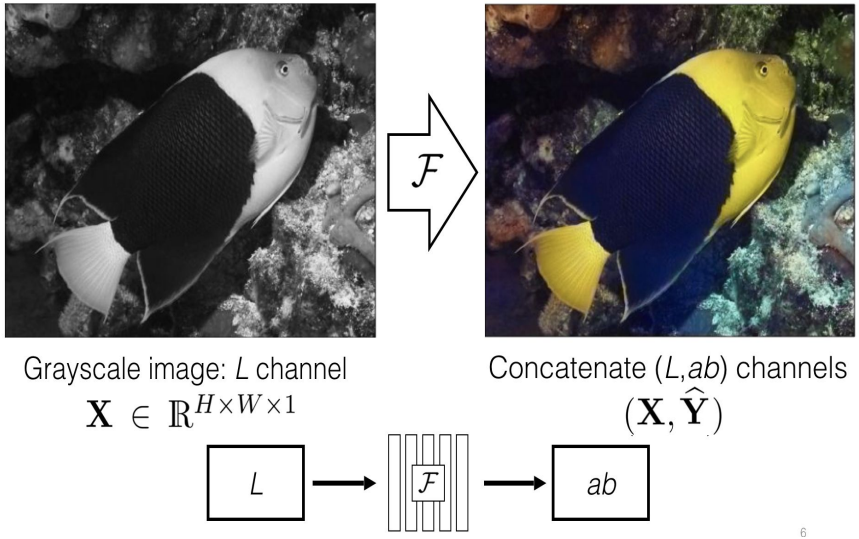
Ground Truth

L2 regression

Pixelwise classification

Slide: Richard Zhang

# Predicting one view from another

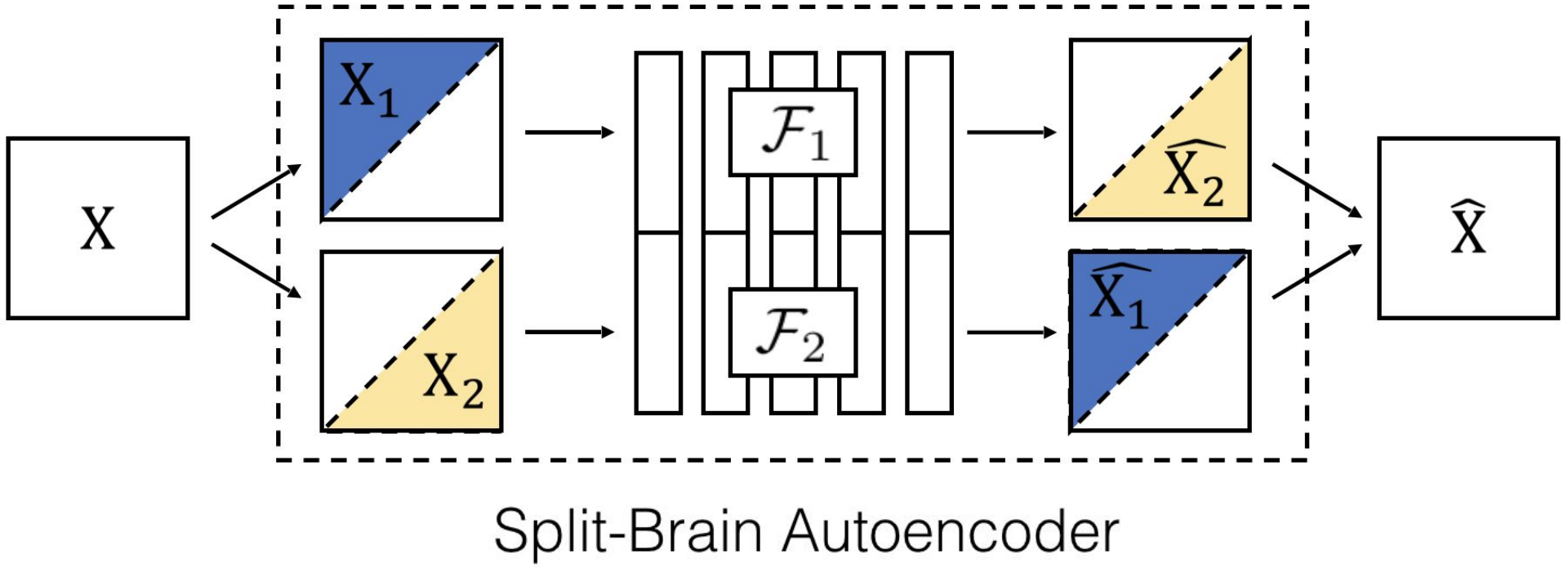



$$\mathrm{L}_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \|\mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}\|_2^2$$

$$\mathrm{L}(\widehat{\mathbf{Z}}, \mathbf{Z}) = -\frac{1}{HW} \sum_{h,w} \sum_{q} \mathbf{Z}_{h,w,q} \log(\widehat{\mathbf{Z}}_{h,w,q})$$

Slide: Richard Zhang

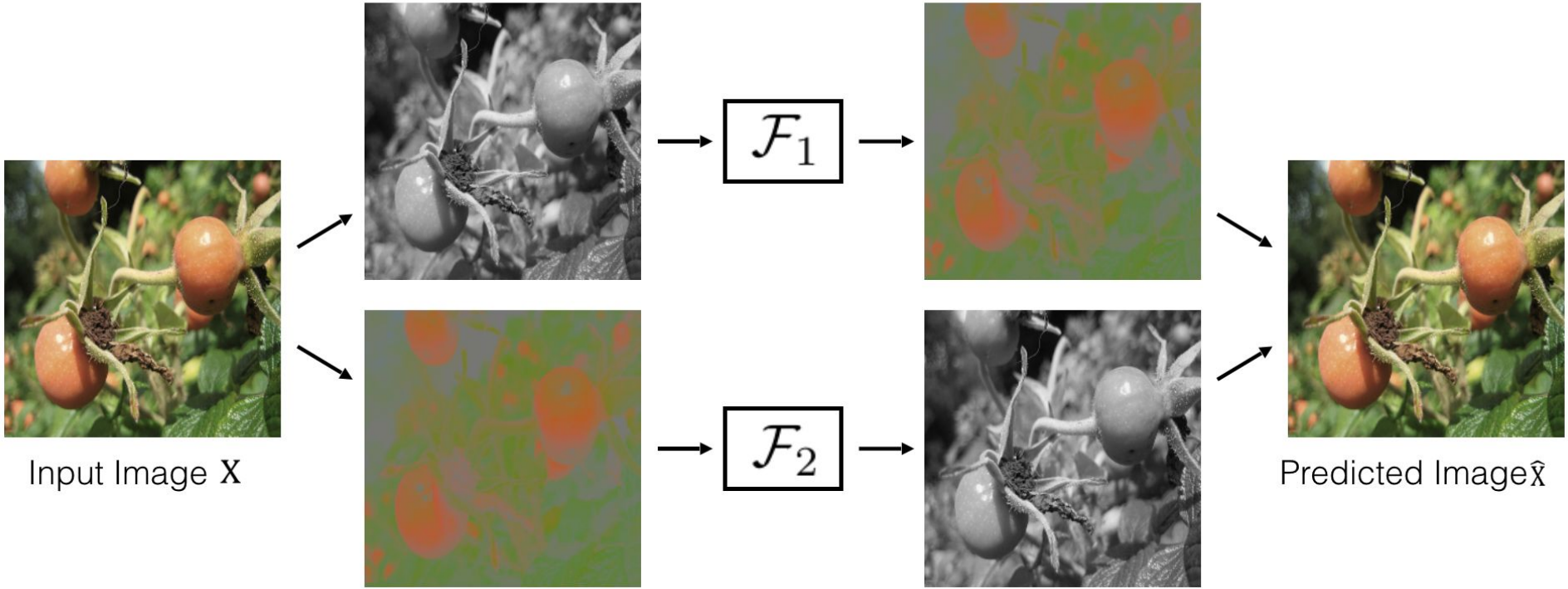
# Predicting one view from another




Split-Brain Autoencoder

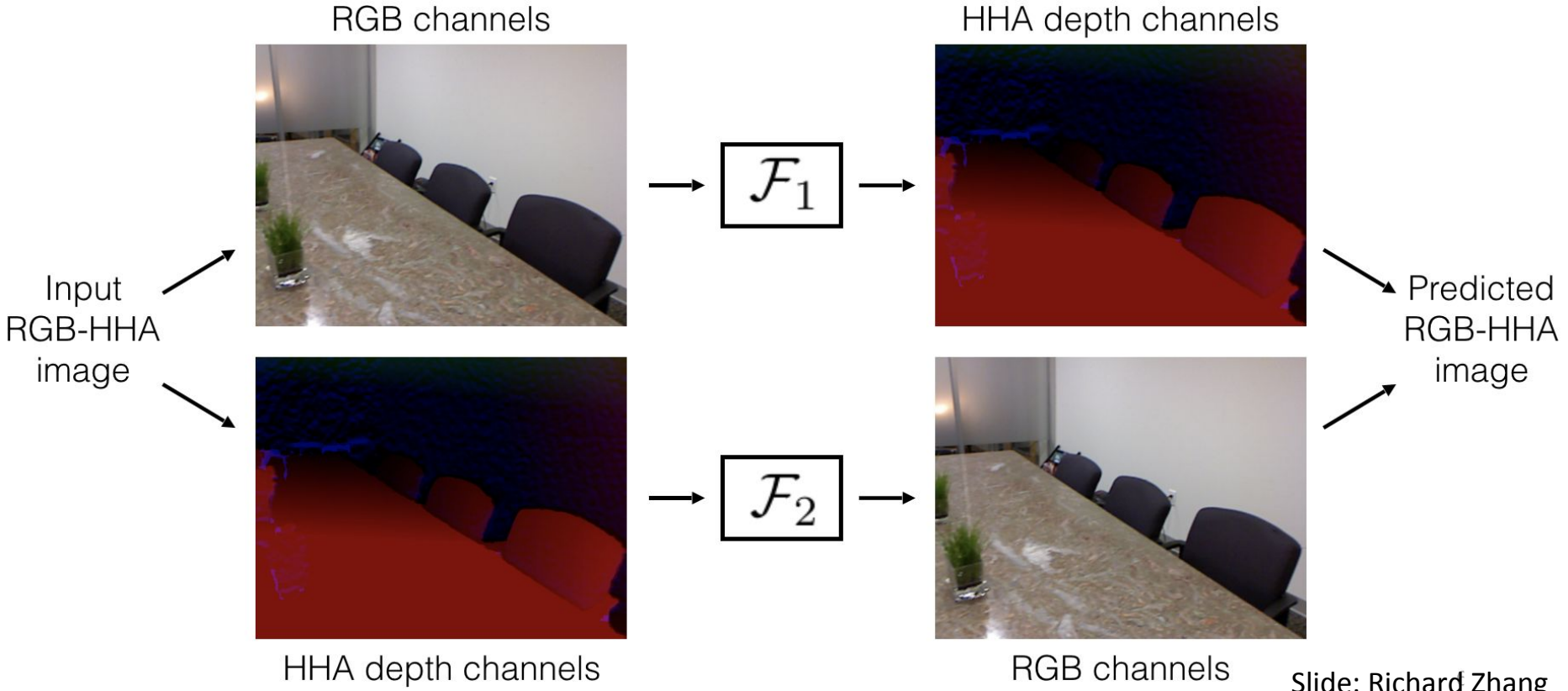
Slide: Richard Zhang

# Predicting one view from another




Slide: Richard Zhang

# Predicting one view from another




Slide: Richard Zhang

# Relative Position of Image Patches

**Unsupervised Visual Representation Learning by Context Prediction**

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

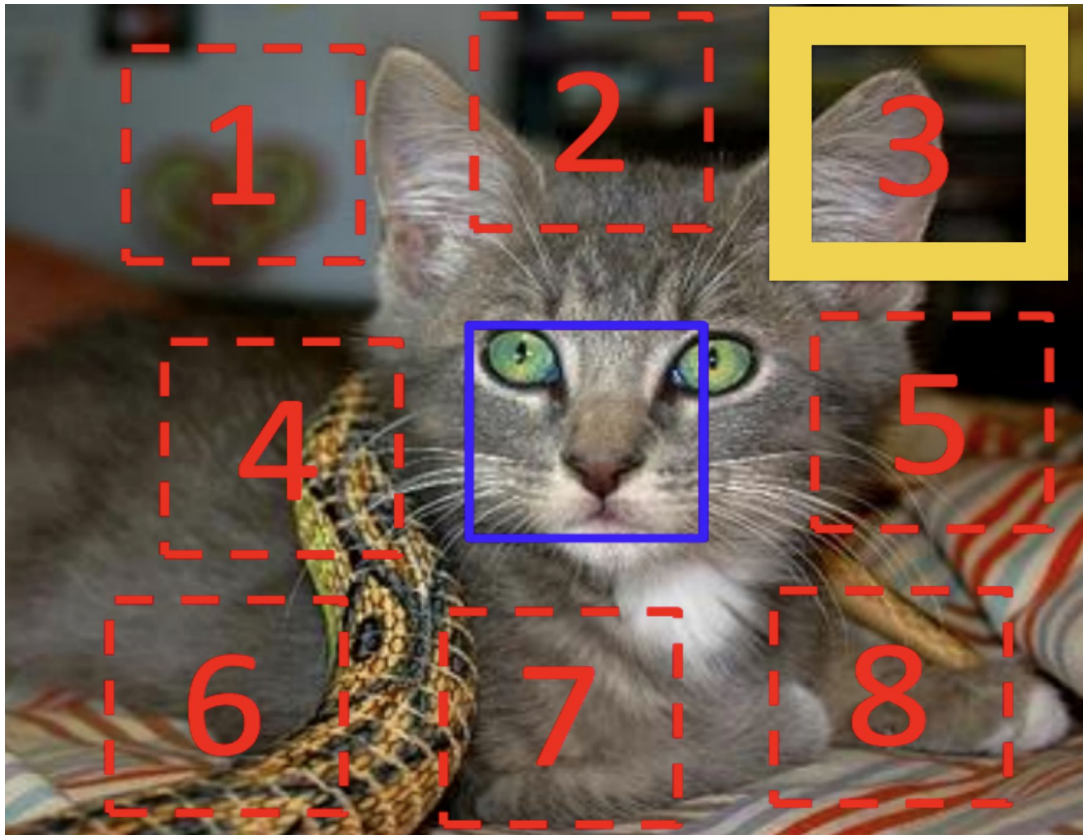
[1] School of Computer Science
Carnegie Mellon University

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

Task: Predict the relative position of the second patch with respect to the first

Slide: Zisserman

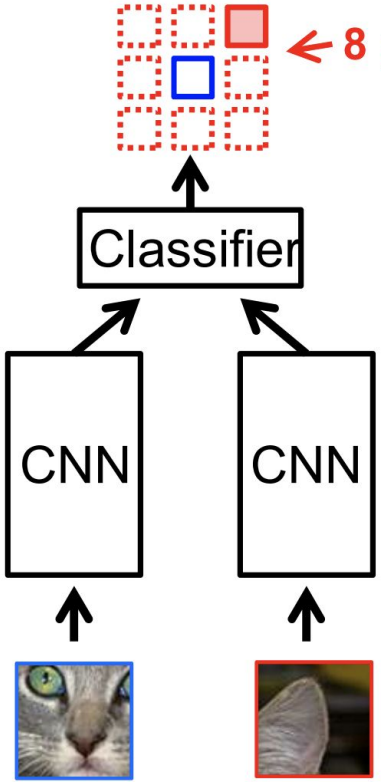
# Relative Position of Image Patches

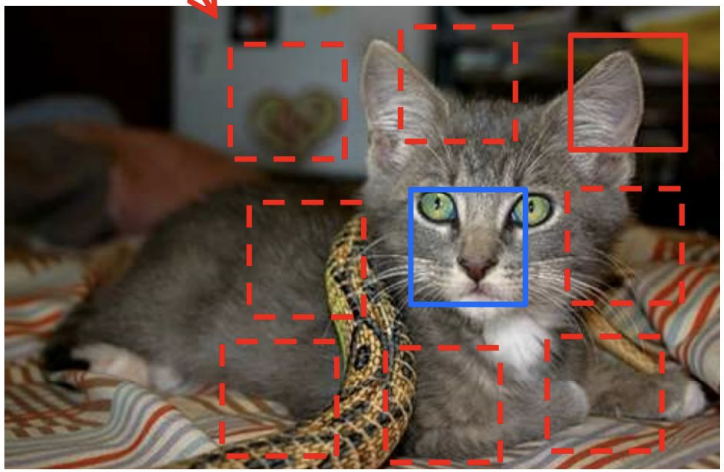



Doersch, Gupta, Efros

Slide: Zisserman

# Relative Position of Image Patches



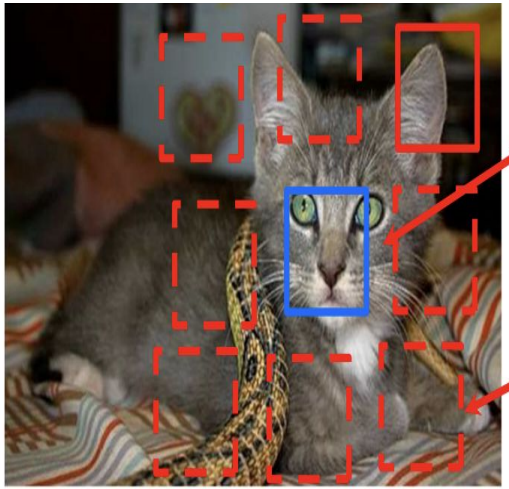



**Randomly Sample Patch**

**Sample Second Patch**

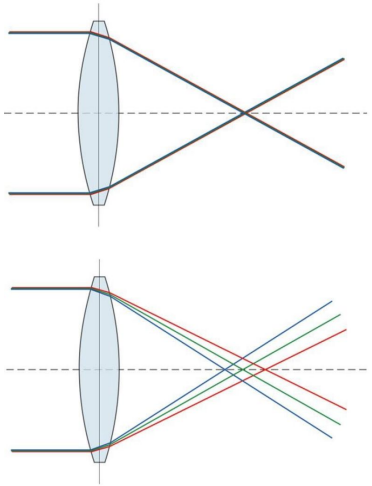
Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

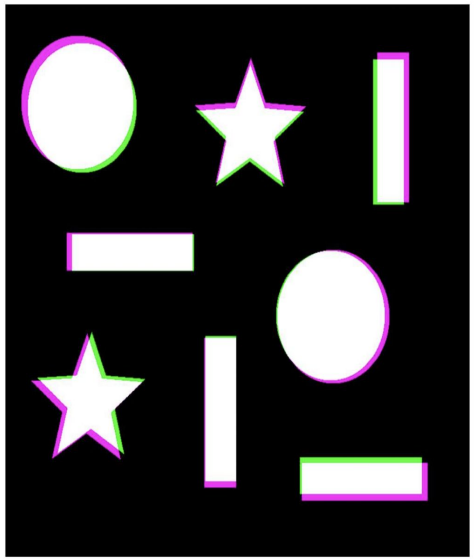
# Relative Position of Image Patches



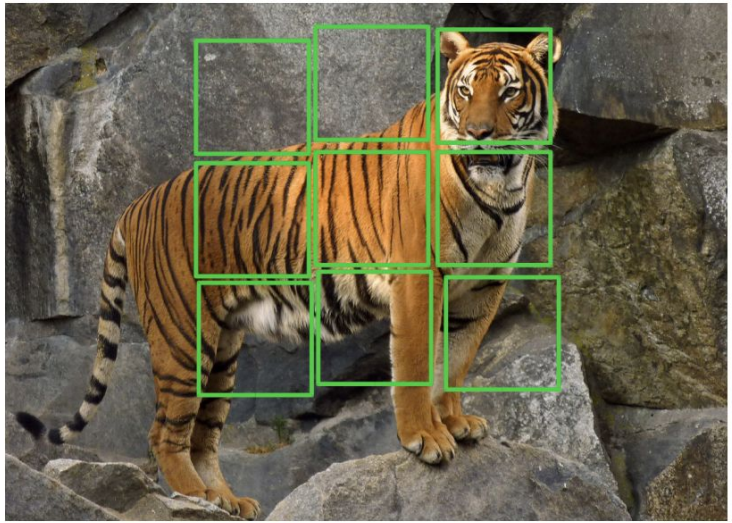
Include a gap

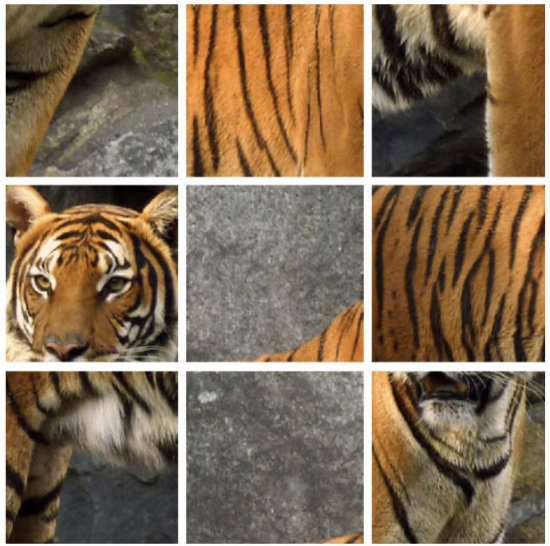
Jitter the patch locations

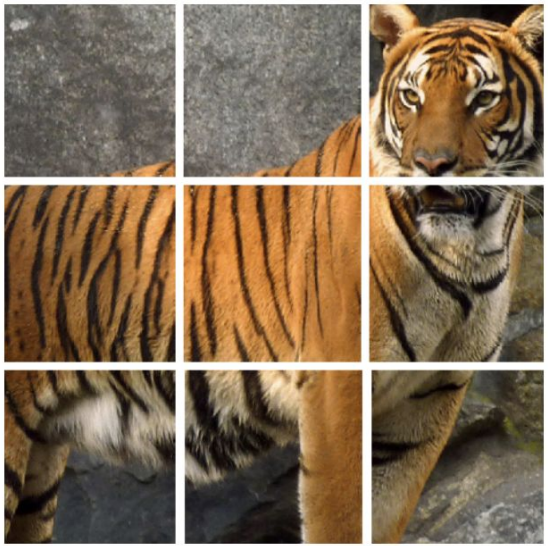




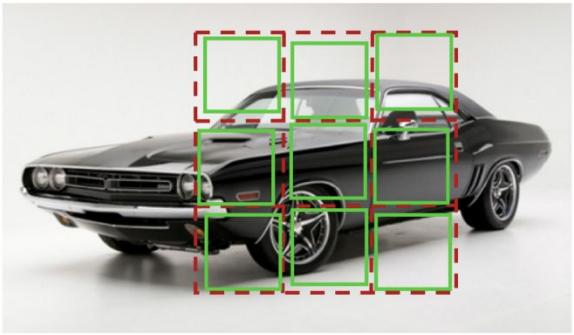
# Solving Jigsaw Puzzles







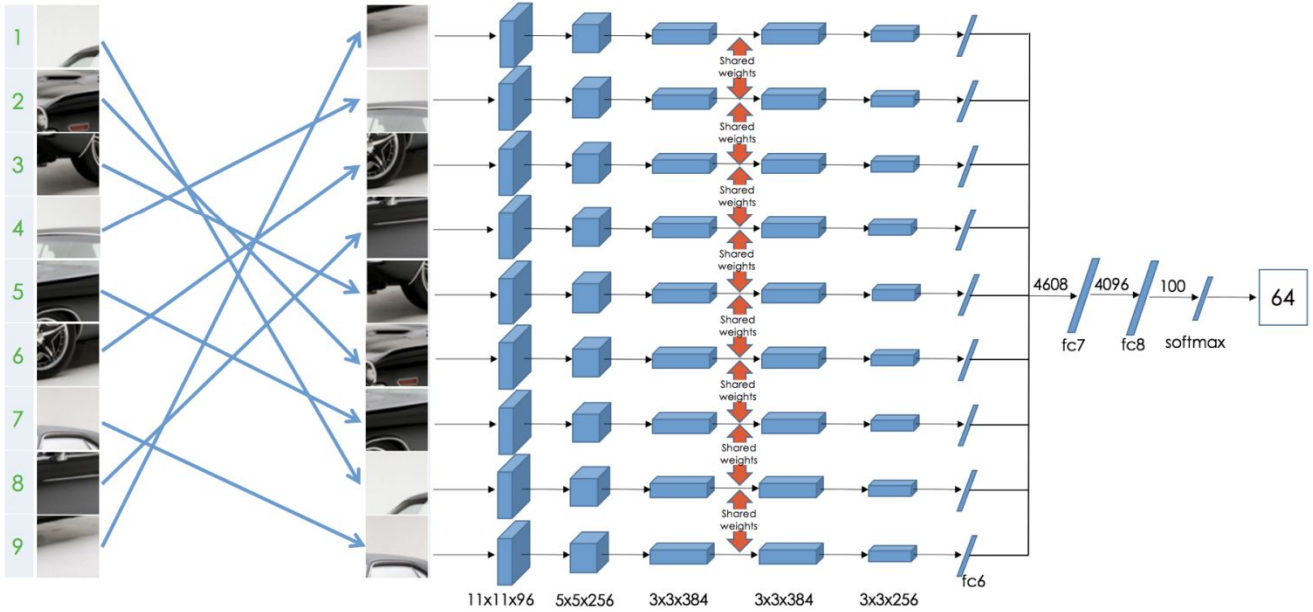
# Solving Jigsaw Puzzles



Permutation Set

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

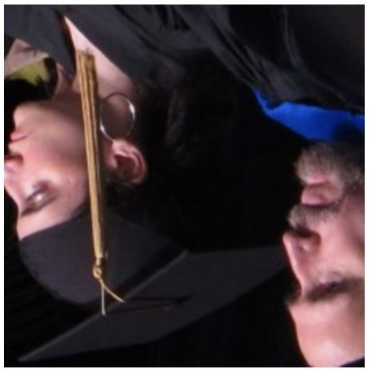
Reorder patches according to the selected permutation



# Rotation

90° rotation

270° rotation



180° rotation

0° rotation

270° rotation

# Rotation

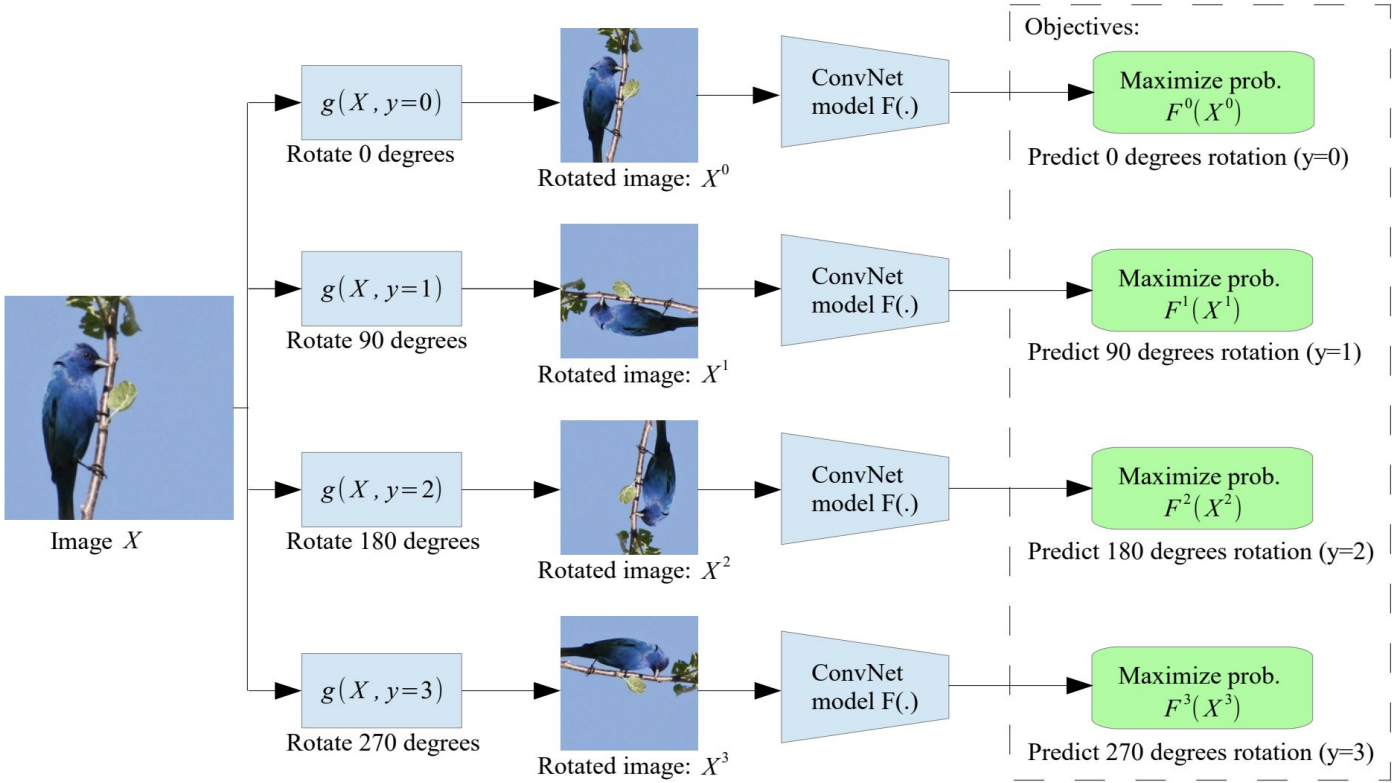


Image $X$
$g(X, y=0)$
Rotate 0 degrees
Rotated image: $X^0$
$g(X, y=1)$
Rotate 90 degrees
Rotated image: $X^1$
$g(X, y=2)$
Rotate 180 degrees
Rotated image: $X^2$
$g(X, y=3)$
Rotate 270 degrees
Rotated image: $X^3$
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
Objectives: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

# 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** |

# Rotation

| | 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** |

# Temporal coherence of color

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

Reference Frame

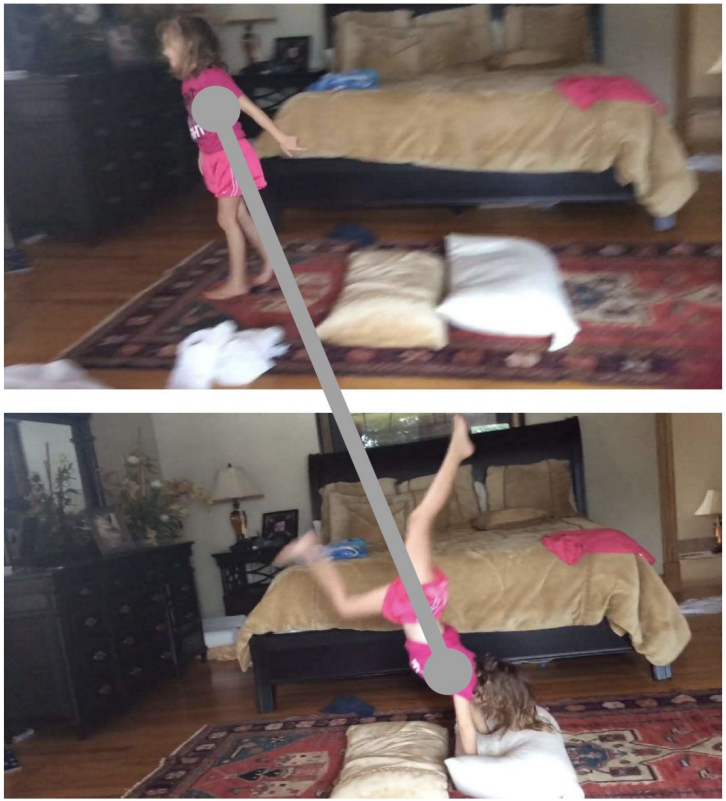
Gray-scale Video

Slide: Zisserman

# Temporal coherence of color

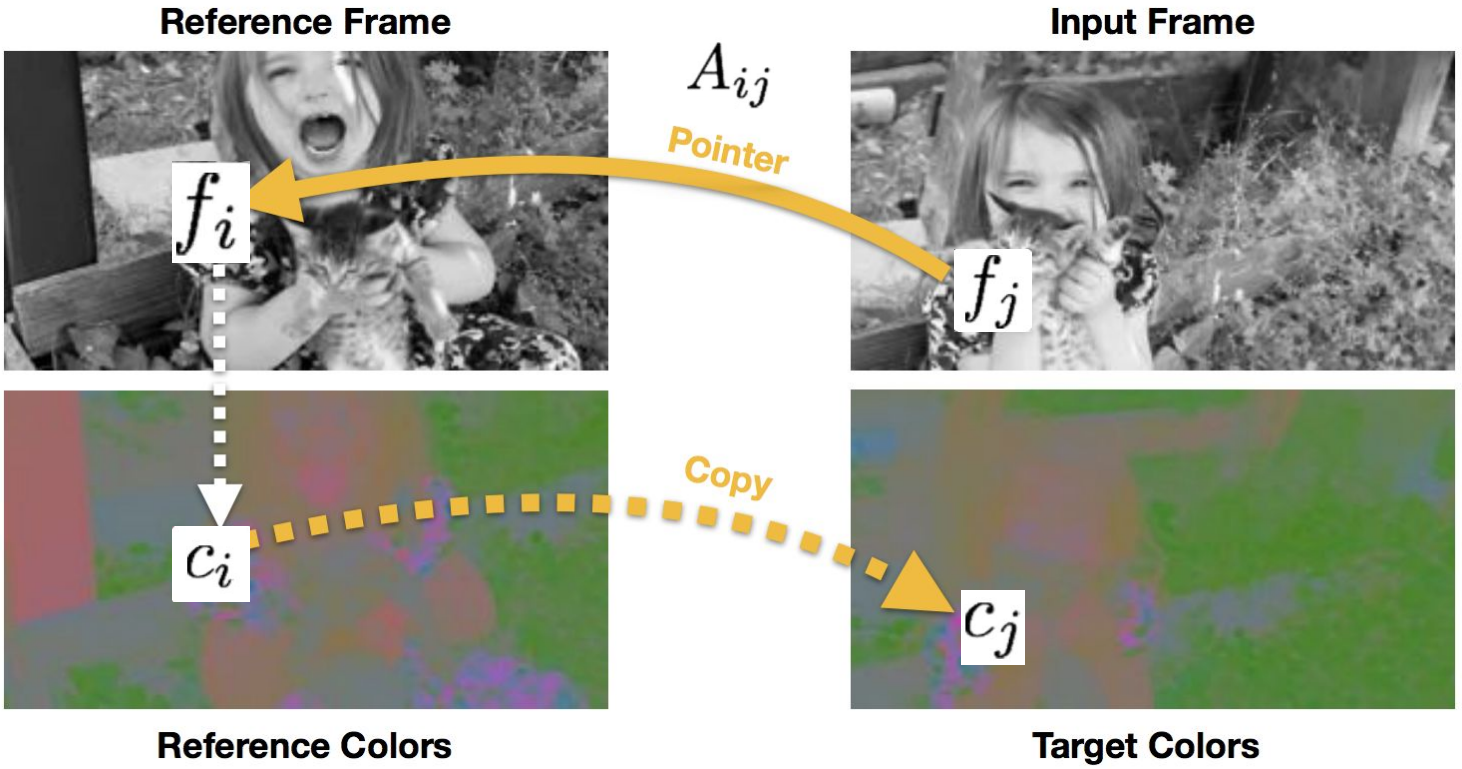
Slide: Zisserman

# Temporal coherence of color

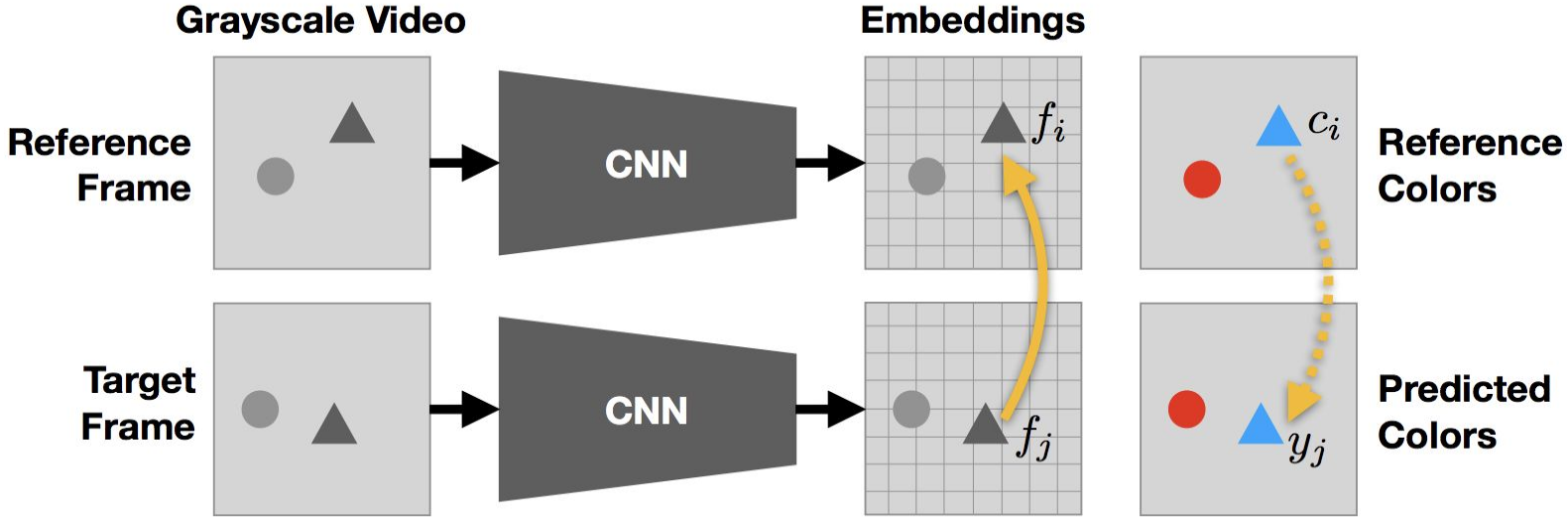


Slide: Zisserman

# Tracking emerges from colorization

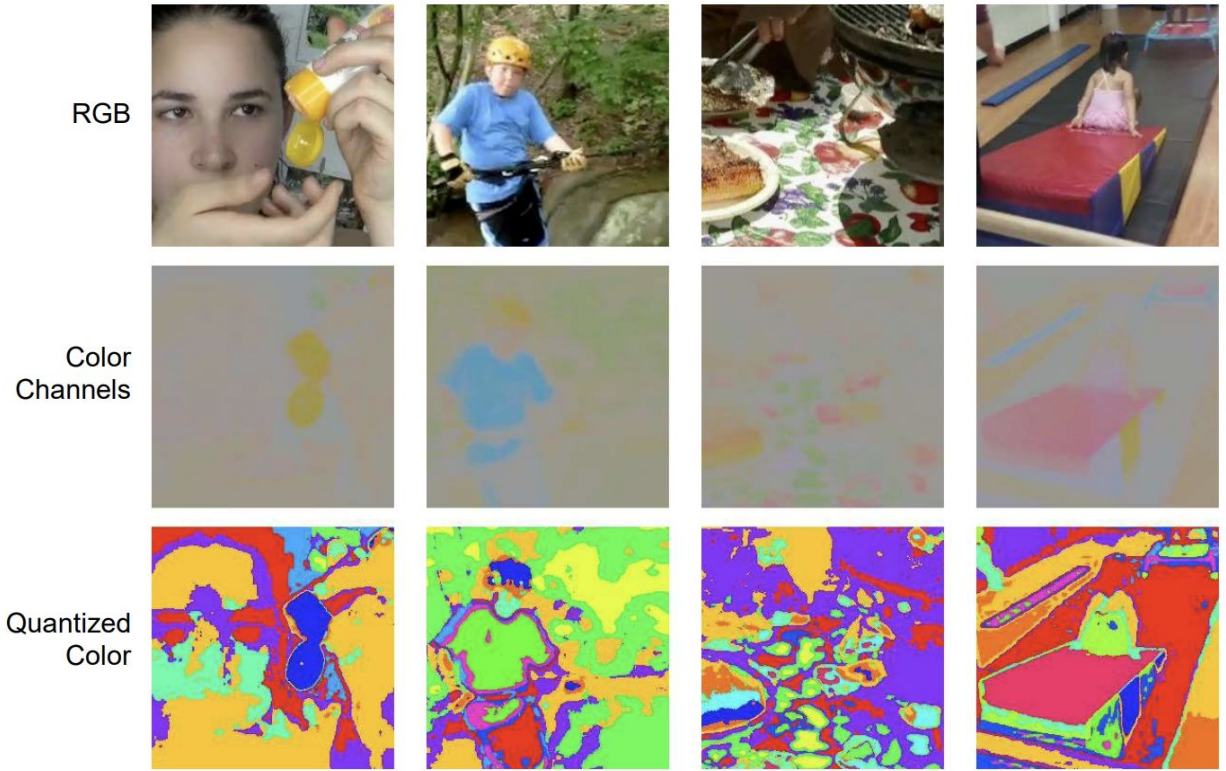


# Tracking emerges from colorization



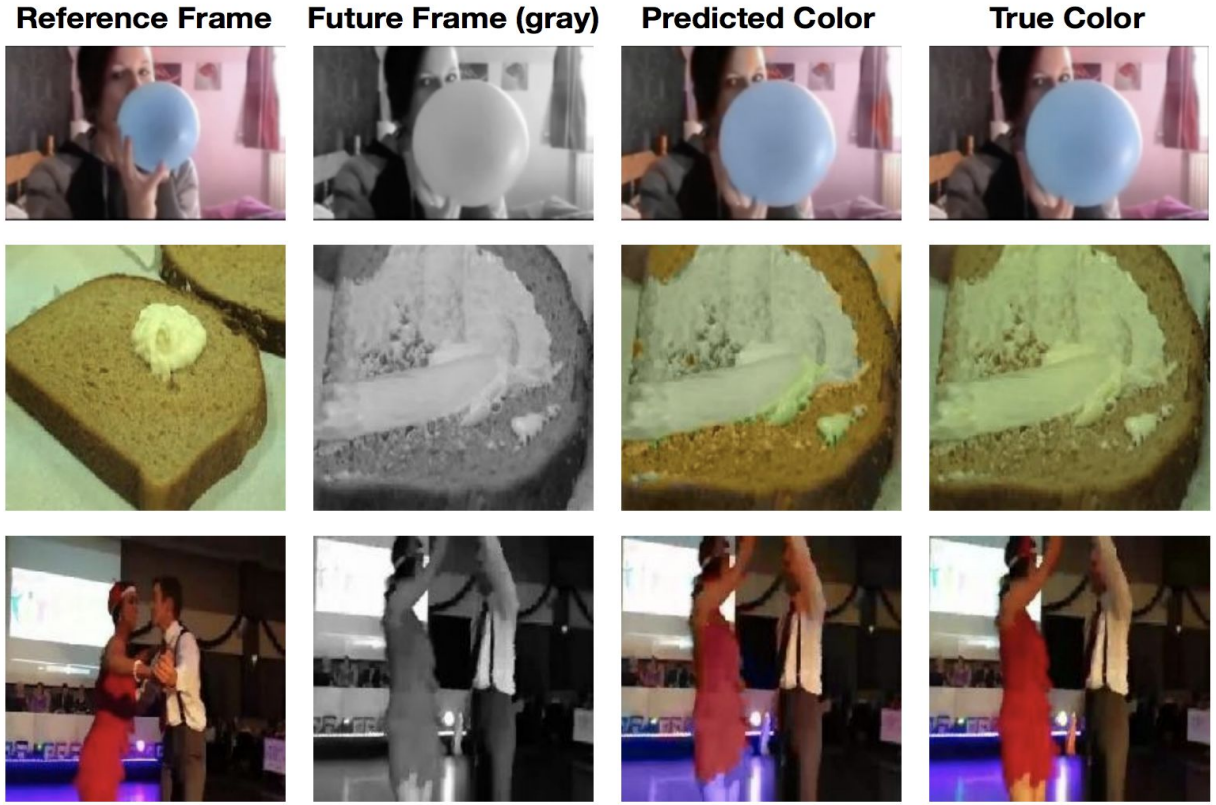

$$A_{ij} = \frac{\exp\left(f_i^T f_j\right)}{\sum_k \exp\left(f_k^T f_j\right)} \qquad \hat{c}_j = \sum_i A_{ij} c_i \qquad \min_f \mathcal{L}\left(c_j, \sum_i A_{ij} c_i\right)$$

# Tracking emerges from colorization

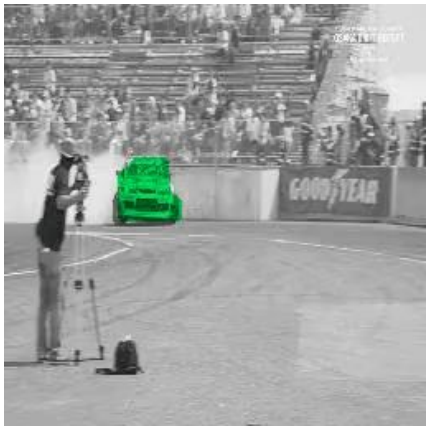



Slide: Zisserman

# Tracking emerges from colorization



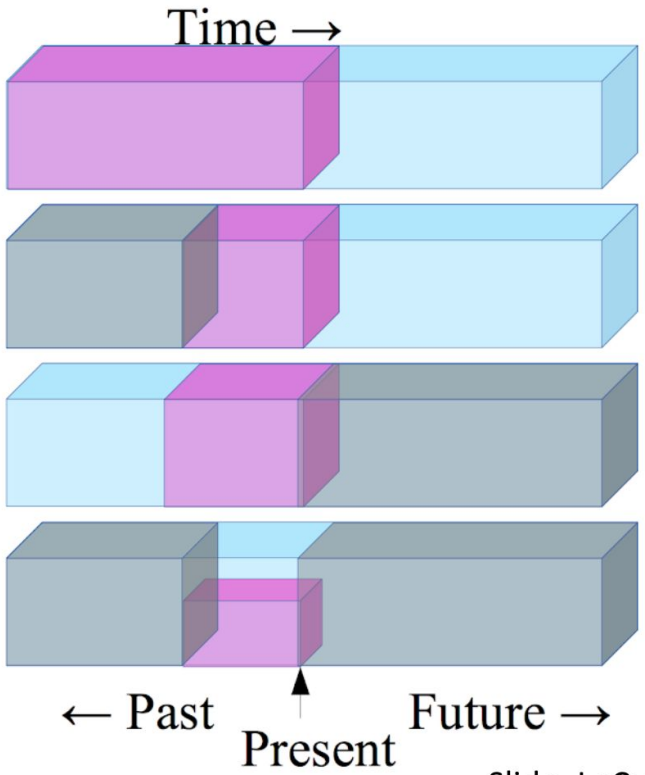
# Tracking emerges from colorization




GIFs from Google AI Blog post

# Predicting neighbouring context

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




Slide: LeCun

# Word Embeddings

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

(From 224n Stanford)

# Word Embeddings

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

The resulting counts matrix will then be:

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

(From 224n Stanford)

# Word Embeddings

**Applying SVD to $X$:**

$$\underset{|V|}{\underset{|V|}{X}} = \underset{|V|}{\underset{|V|}{\begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_{|V|} \\ | & | & & | \end{bmatrix}}} \underset{|V|}{\underset{|V|}{\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{|V|} \end{bmatrix}}} \underset{|V|}{\underset{|V|}{\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \\ - & v_{|V|} & - \end{bmatrix}}}$$

(From 224n Stanford)

# Word Embeddings

SVD approach suffers from:

- Sparsity
- SVD computation costs
- Infrequent words
- Noise from frequent words
- There are hacks to fix some of these (ex TF-IDF) but still not very reliable

(From 224n Stanford)

# n-gram Language Models

Unigram

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=1}^{n} P(w_i)$$

Bigram

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=2}^{n} P(w_i | w_{i-1})$$

(From 224n Stanford)

# word2vec

## Efficient Estimation of Word Representations in Vector Space

**Tomas Mikolov**
Google Inc., Mountain View, CA
tmikolov@google.com

**Kai Chen**
Google Inc., Mountain View, CA
kaichen@google.com

**Greg Corrado**
Google Inc., Mountain View, CA
gcorrado@google.com

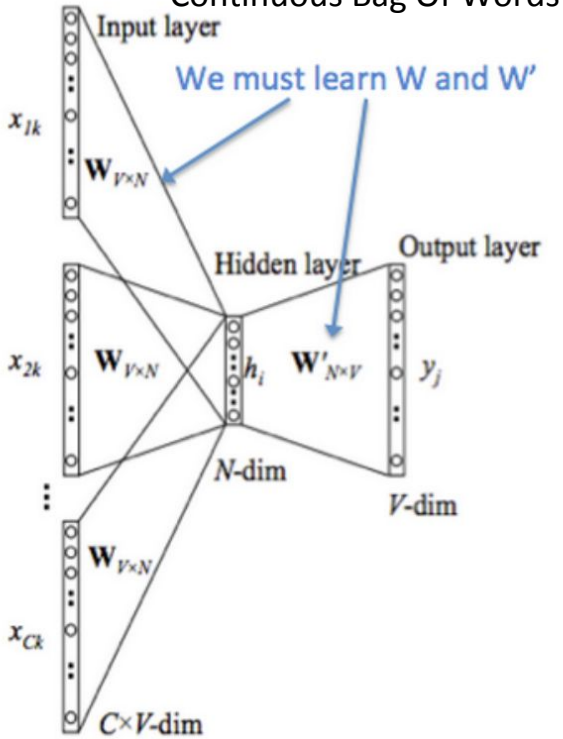
**Jeffrey Dean**
Google Inc., Mountain View, CA
jeff@google.com

### Abstract

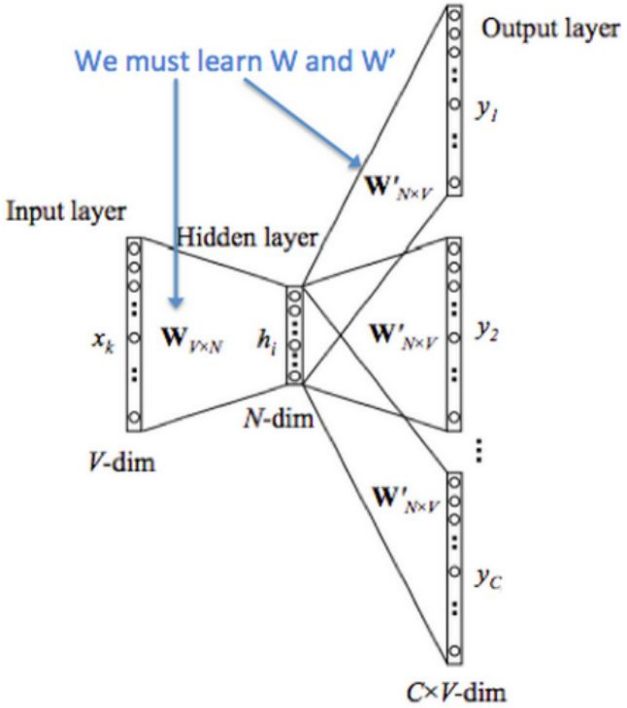
We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

# word2vec

Continuous Bag Of Words (CBOW)



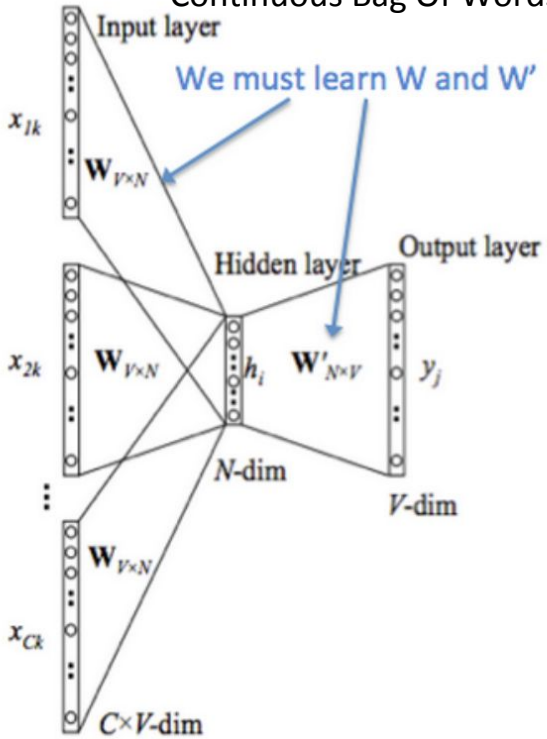

Skip Gram




(From 224n Stanford)

# word2vec - CBOW

Continuous Bag Of Words (CBOW)

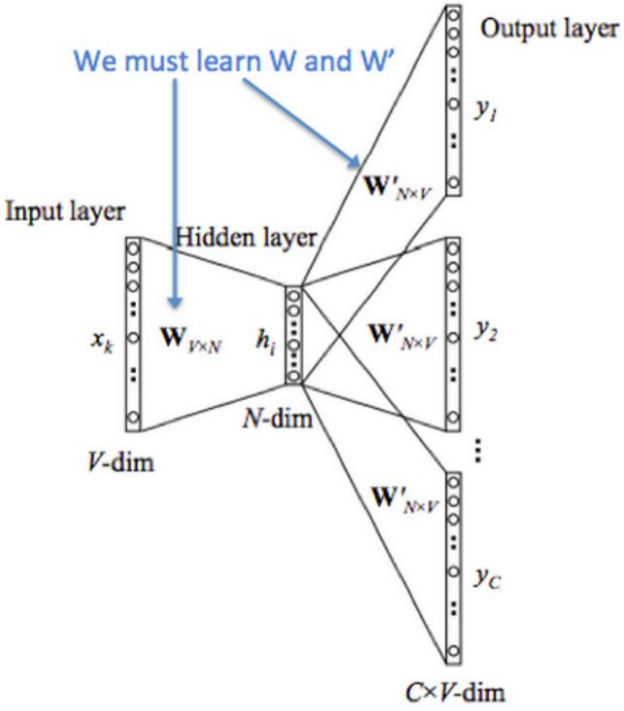



$$\begin{aligned}\text{minimize } J &= -\log P(w_c|w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}) \\ &= -\log P(u_c|\hat{v}) \\ &= -\log \frac{\exp(u_c^T\hat{v})}{\sum_{j=1}^{|V|}\exp(u_j^T\hat{v})} \\ &= -u_c^T\hat{v} + \log\sum_{j=1}^{|V|}\exp(u_j^T\hat{v})\end{aligned}$$

(From 224n Stanford)

# word2vec - Skip Gram

Skip Gram




$$\begin{aligned}
\text{minimize } J &= -\log P(w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}|w_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} P(w_{c-m+j}|w_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} P(u_{c-m+j}|v_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} \frac{\exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|}\exp(u_k^T v_c)} \\
&= -\sum_{j=0,j\neq m}^{2m} u_{c-m+j}^T v_c + 2m\log\sum_{k=1}^{|V|}\exp(u_k^T v_c)
\end{aligned}$$

$$\begin{aligned}
J &= -\sum_{j=0,j\neq m}^{2m} \log P(u_{c-m+j}|v_c) \\
&= \sum_{j=0,j\neq m}^{2m} H(\hat{y}, y_{c-m+j})
\end{aligned}$$

(From 224n Stanford)

# word2vec - Skip Gram

Skip-gram model

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^{\top}v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_{w}}^{\top}v_{w_I}\right)}$$

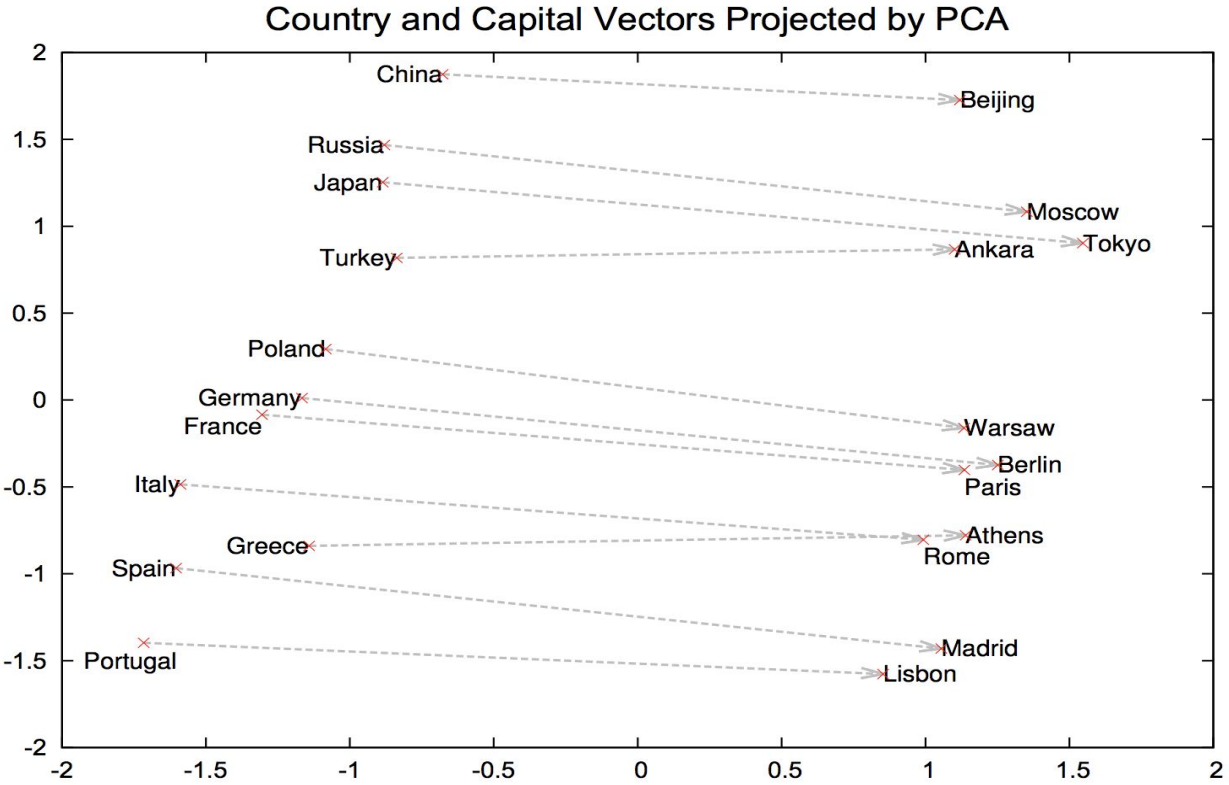
Don't have to have the denominator over all words in the vocabulary

- Can use negative sampling

$$\log\sigma({v'_{w_O}}^{\top}v_{w_I})+\sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\left[\log\sigma(-{v'_{w_i}}^{\top}v_{w_I})\right]$$

$$P(w_i)=1-\sqrt{\frac{t}{f(w_i)}}$$

# word2vec


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Warsaw
Germany
France
Berlin
Paris
Italy
Athens
Greece
Rome
Spain
Madrid
Portugal
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

# word2vec

| Newspapers | | | |
|---|---|---|---|
| New York | New York Times | Baltimore | Baltimore Sun |
| San Jose | San Jose Mercury News | Cincinnati | Cincinnati Enquirer |
| NHL Teams | | | |
| Boston | Boston Bruins | Montreal | Montreal Canadiens |
| Phoenix | Phoenix Coyotes | Nashville | Nashville Predators |
| NBA Teams | | | |
| Detroit | Detroit Pistons | Toronto | Toronto Raptors |
| Oakland | Golden State Warriors | Memphis | Memphis Grizzlies |
| Airlines | | | |
| Austria | Austrian Airlines | Spain | Spainair |
| Belgium | Brussels Airlines | Greece | Aegean Airlines |
| Company executives | | | |
| Steve Ballmer | Microsoft | Larry Page | Google |
| Samuel J. Palmisano | IBM | Werner Vogels | Amazon |

# word2vec

| | NEG-15 with $10^{-5}$ subsampling | HS with $10^{-5}$ subsampling |
|---|---|---|
| Vasco de Gama | Lingsugur | Italian explorer |
| Lake Baikal | Great Rift Valley | Aral Sea |
| Alan Bean | Rebbeca Naomi | moonwalker |
| Ionian Sea | Ruegen | Ionian Islands |
| chess master | chess grandmaster | Garry Kasparov |

Table 4: Examples of the closest entities to the given short phrases, using two different models.

| Czech + currency | Vietnam + capital | German + airlines | Russian + river | French + actress |
|---|---|---|---|---|
| koruna | Hanoi | airline Lufthansa | Moscow | Juliette Binoche |
| Check crown | Ho Chi Minh City | carrier Lufthansa | Volga River | Vanessa Paradis |
| Polish zolty | Viet Nam | flag carrier Lufthansa | upriver | Charlotte Gainsbourg |
| CTK | Vietnamese | Lufthansa | Russia | Cecile De |

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

# word2vec

| Model (training time) | Redmond | Havel | ninjutsu | graffiti | capitulate |
|---|---|---|---|---|---|
| Collobert (50d) (2 months) | conyers<br>lubbock<br>keene | plauen<br>dzerzhinsky<br>osterreich | reiki<br>kohona<br>karate | cheesecake<br>gossip<br>dioramas | abdicate<br>accede<br>rearm |
| Turian (200d) (few weeks) | McCarthy<br>Alston<br>Cousins | Jewell<br>Arzu<br>Ovitz | -<br>-<br>- | gunfire<br>emotion<br>impunity | -<br>-<br>- |
| Mnih (100d) (7 days) | Podhurst<br>Harlang<br>Agarwal | Pontiff<br>Pinochet<br>Rodionov | -<br>-<br>- | anaesthetics<br>monkeys<br>Jews | Mavericks<br>planning<br>hesitated |
| Skip-Phrase (1000d, 1 day) | Redmond Wash.<br>Redmond Washington<br>Microsoft | Vaclav Havel<br>president Vaclav Havel<br>Velvet Revolution | ninja<br>martial arts<br>swordsmanship | spray paint<br>grafitti<br>taggers | capitulation<br>capitulated<br>capitulating |

Table 6: Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.

# Contrastive Predictive Coding

## Representation Learning with Contrastive Predictive Coding

**Aaron van den Oord**
DeepMind
avdnoord@google.com

**Yazhe Li**
DeepMind
yazhe@google.com

**Oriol Vinyals**
DeepMind
vinyals@google.com

**Abstract**

While supervised learning has enabled great progress in many applications, unsupervised learning has not seen such widespread adoption, and remains an important and challenging endeavor for artificial intelligence. In this work, we propose a universal unsupervised learning approach to extract useful representations from high-dimensional data, which we call Contrastive Predictive Coding. The key insight of our model is to learn such representations by predicting the future in *latent* space by using powerful autoregressive models. We use a probabilistic contrastive loss which induces the latent space to capture information that is maximally useful to predict future samples. It also makes the model tractable by using negative sampling. While most prior work has focused on evaluating representations for a particular modality, we demonstrate that our approach is able to learn useful representations achieving strong performance on four distinct domains: speech, images, text and reinforcement learning in 3D environments.

# Contrastive Predictive Coding

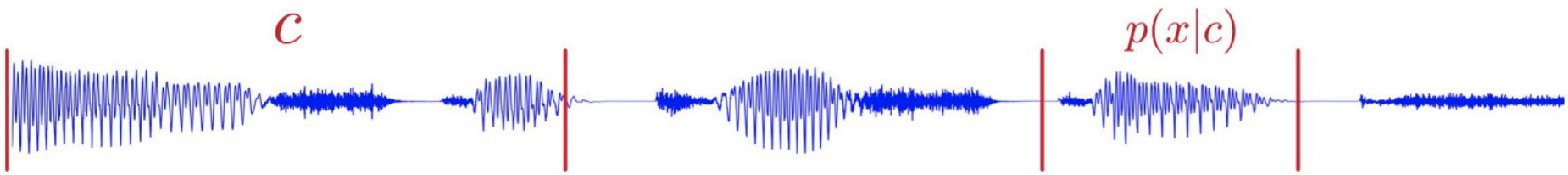



Figure from Alex Graves

# Contrastive Predictive Coding


Figure from Alex Graves

# Contrastive Predictive Coding

$$\max_{\theta} MI(c; z)$$

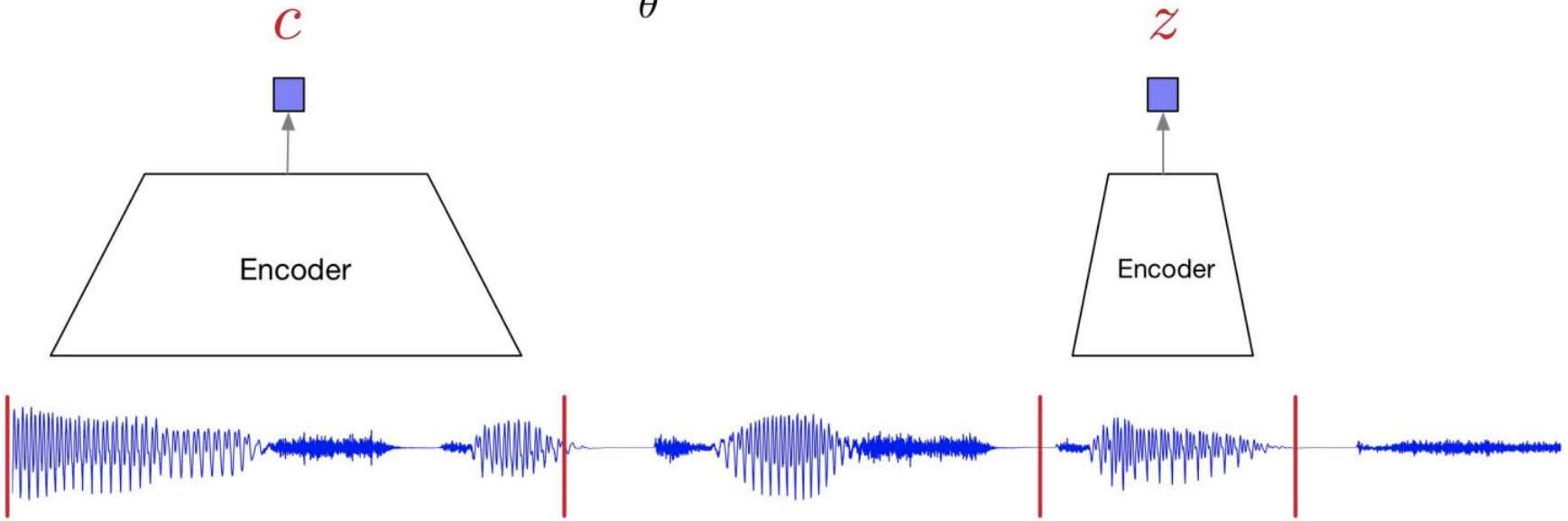



Figure from Alex Graves

# Contrastive Predictive Coding

$$\frac{\exp f(c, z_i)}{\sum_j \exp f(c, z_j)}$$

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

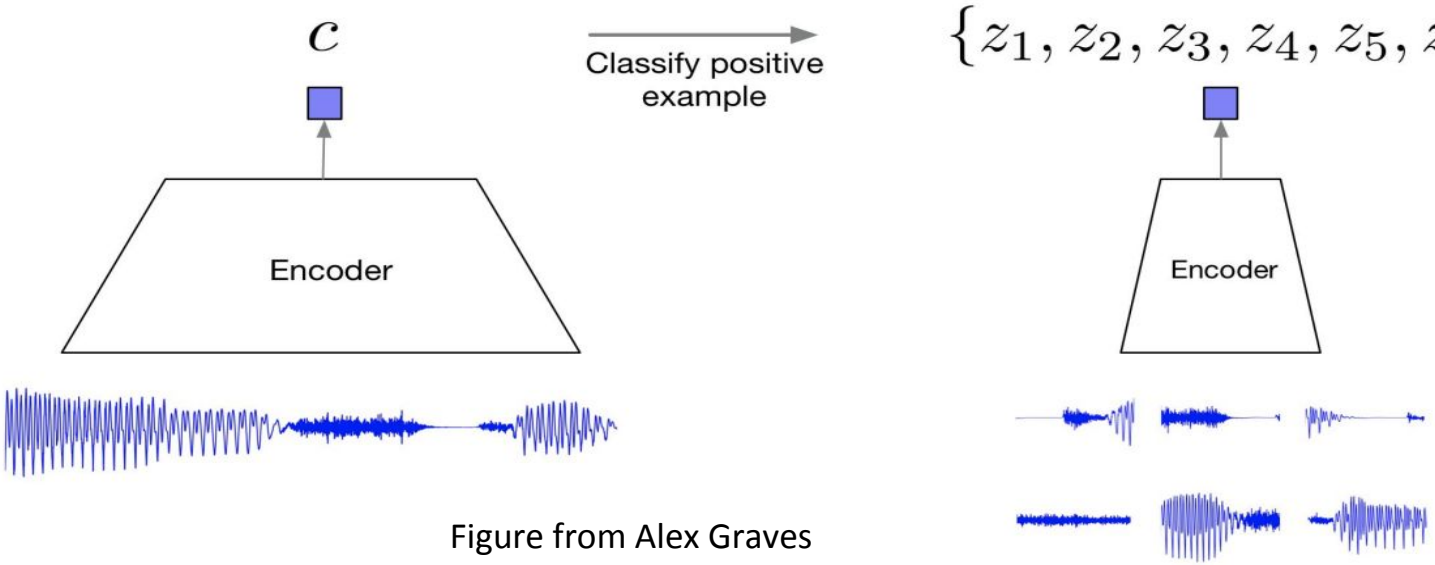



Figure from Alex Graves

# Contrastive Predictive Coding

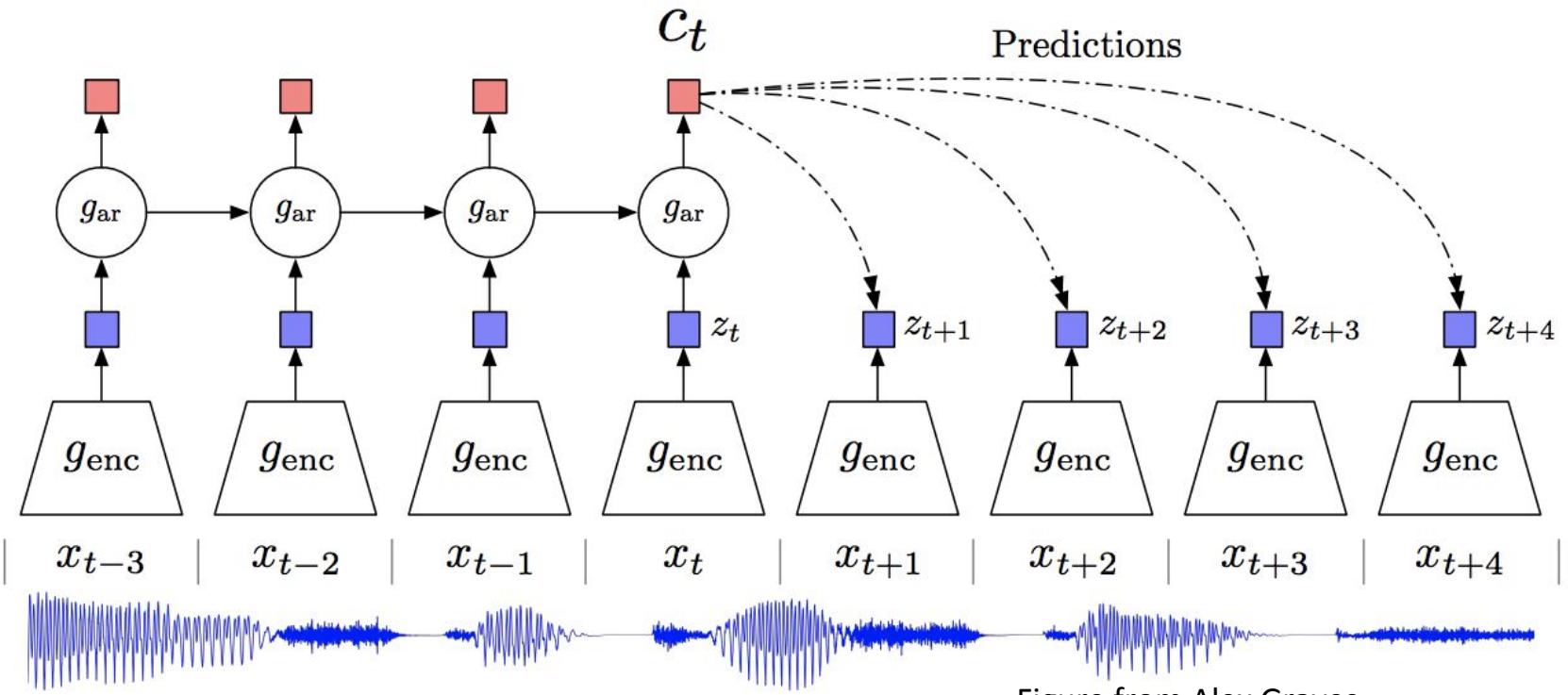
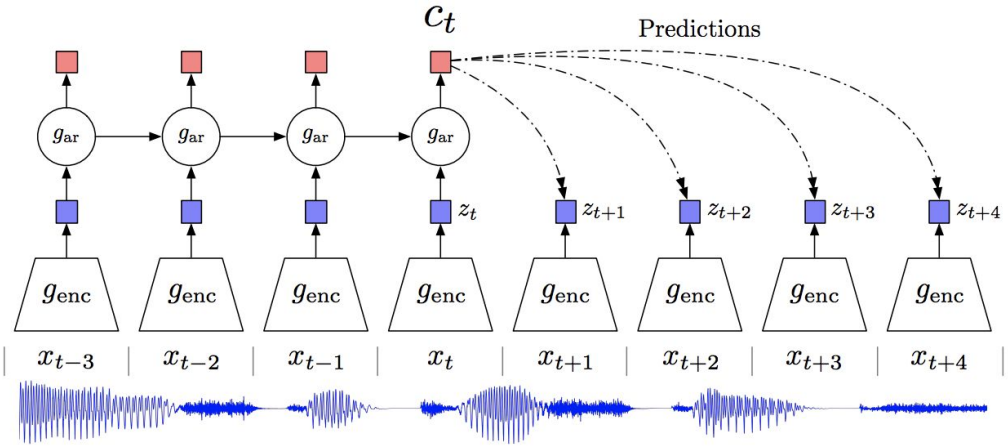



Figure from Alex Graves

# Contrastive Predictive Coding

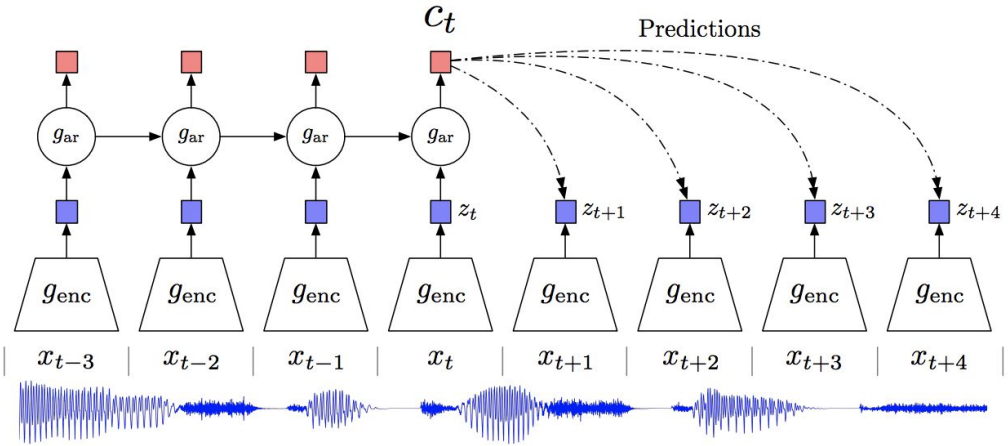



$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_X\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

Figure from Alex Graves

# Contrastive Predictive Coding




$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_X\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

Figure from Alex Graves

# Contrastive Predictive Coding

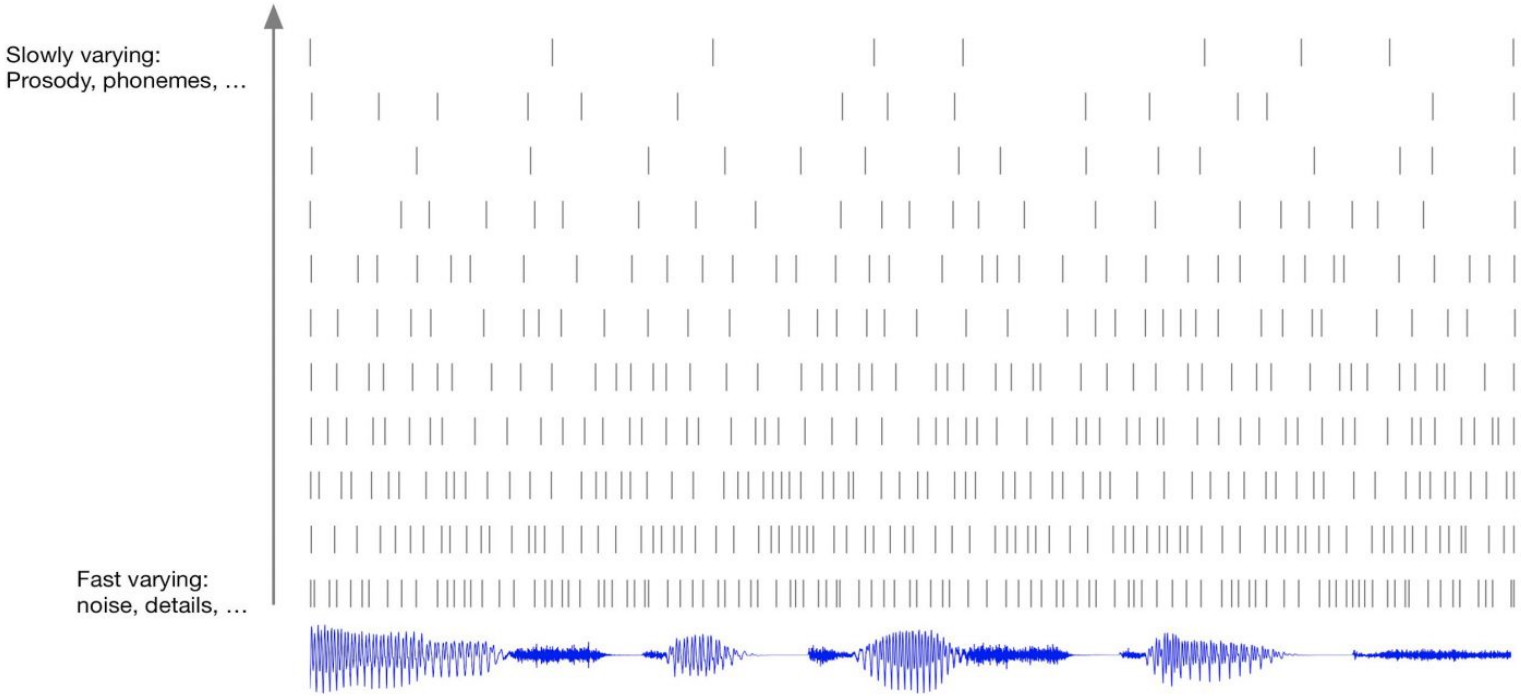



Figure from Alex Graves

# Contrastive Predictive Coding

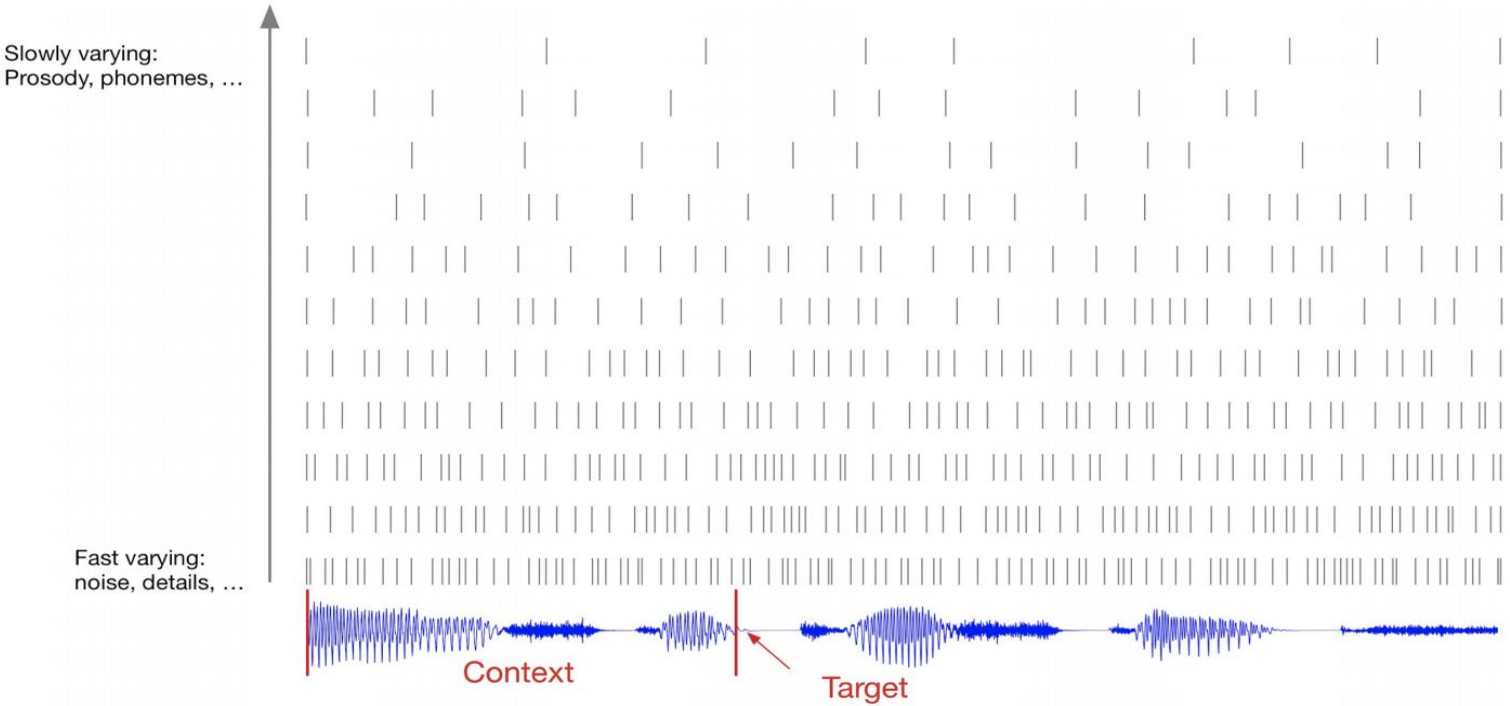



Figure from Alex Graves

# Contrastive Predictive Coding

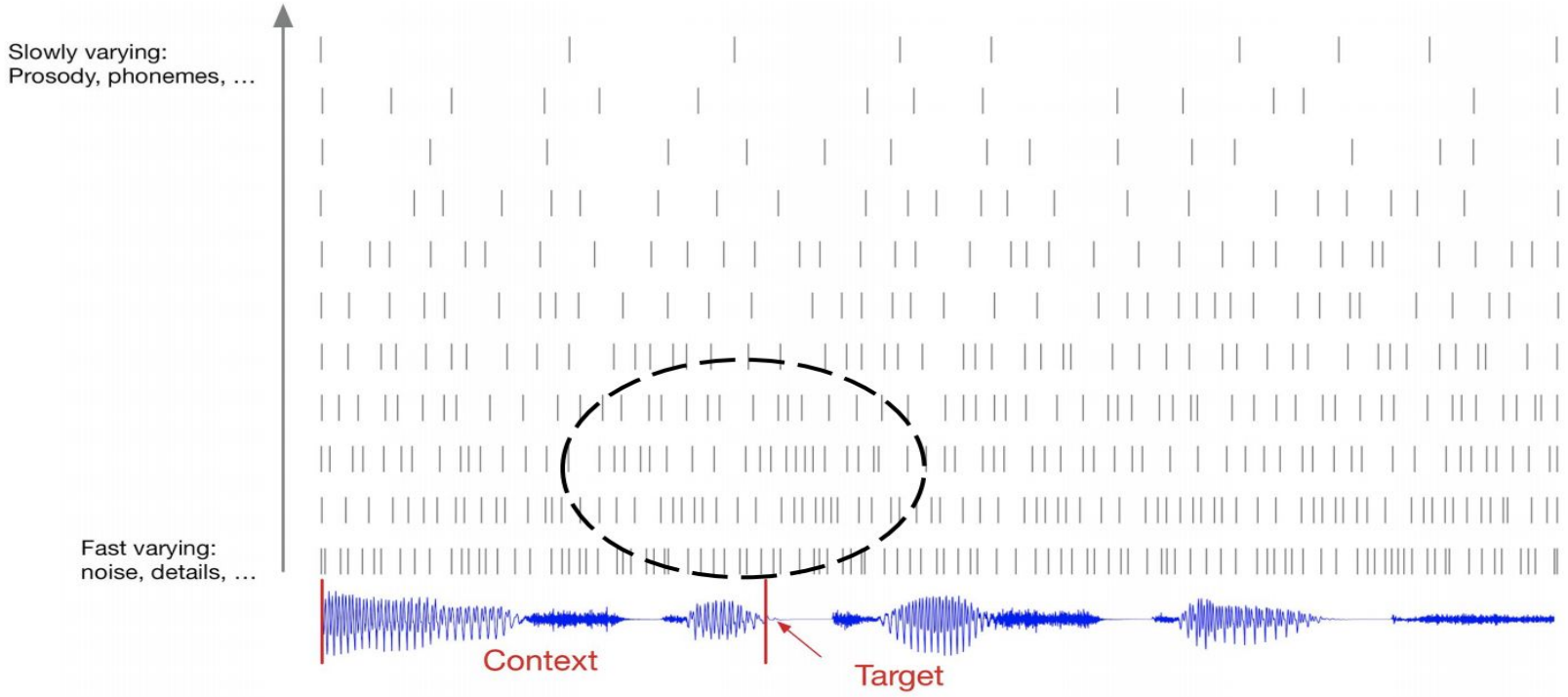



Figure from Alex Graves

# Contrastive Predictive Coding

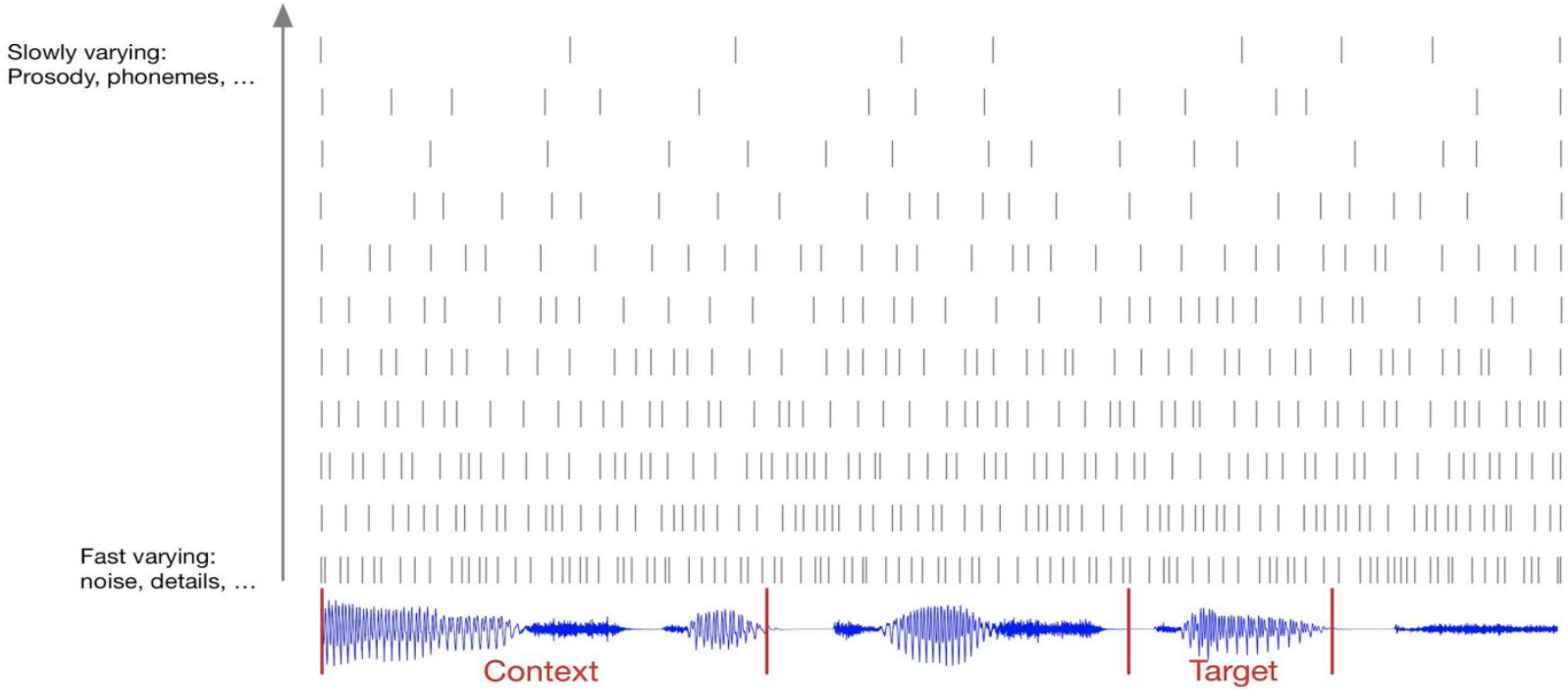



Figure from Alex Graves

# Contrastive Predictive Coding

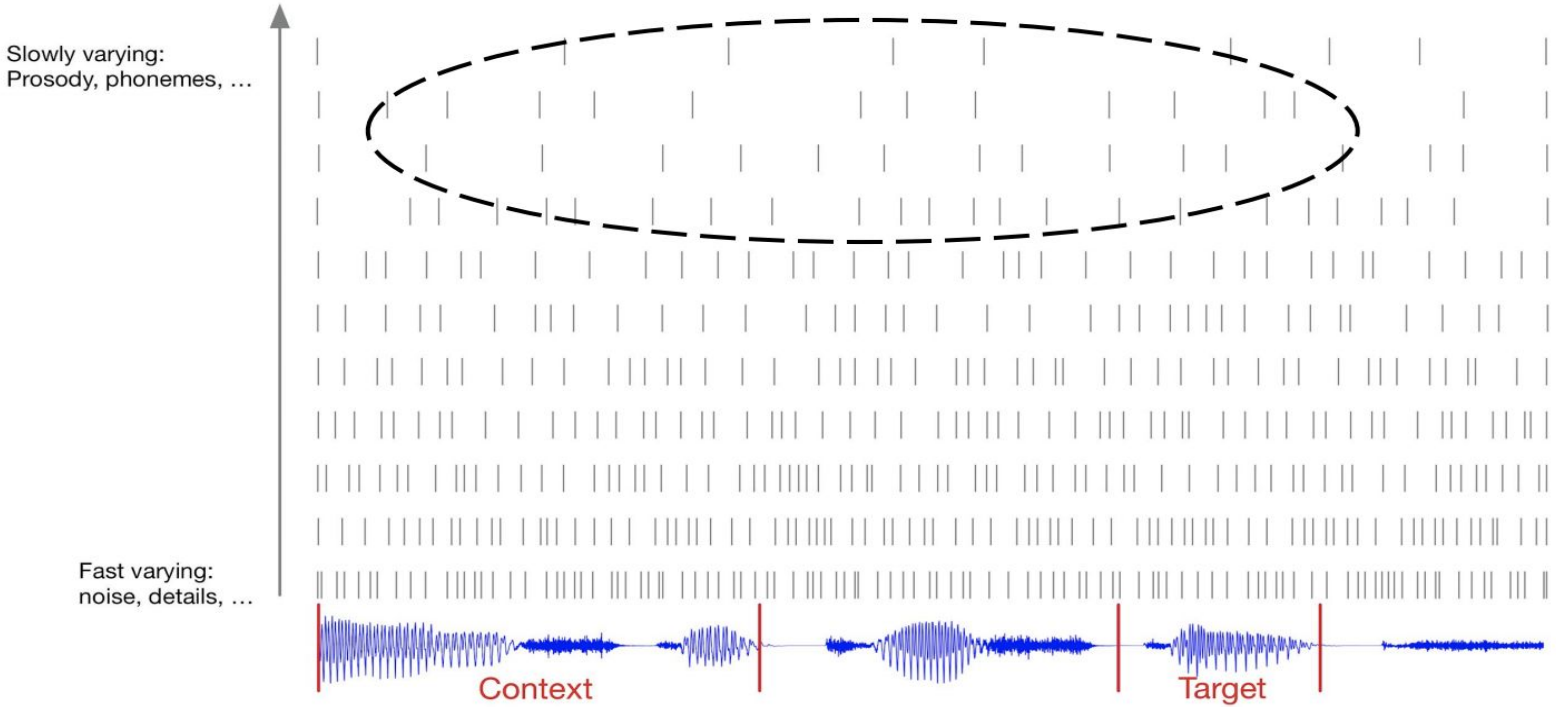



Figure from Alex Graves

# CPC - Speech

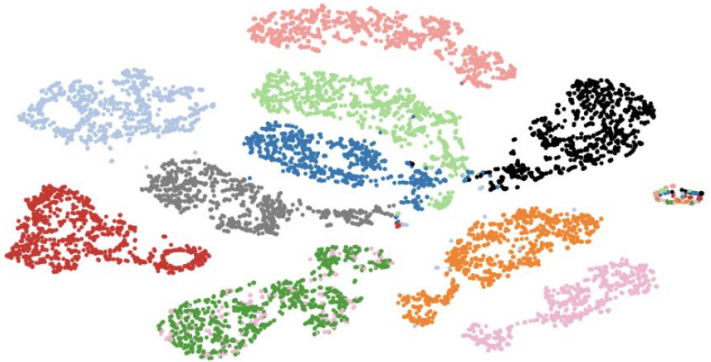


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

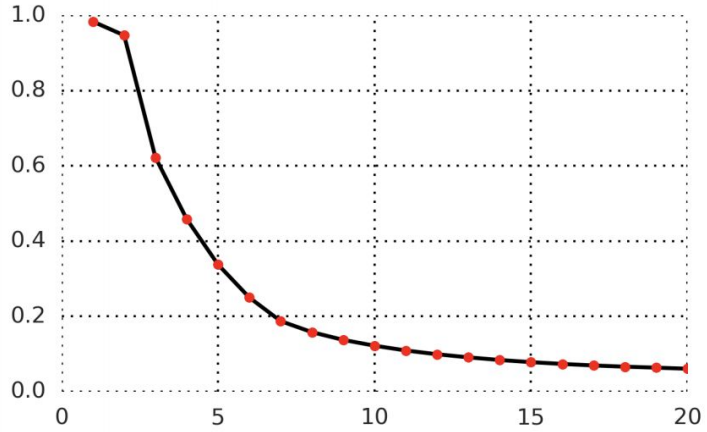



Figure 3: Average accuracy of predicting the positive sample in the contrastive loss for 1 to 20 latent steps in the future of a speech waveform. The model predicts up to 200ms in the future as every step consists of 10ms of audio.

# CPC - Speech

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

Table 1: LibriSpeech phone and speaker classification results. For phone classification there are 41 possible classes and for speaker classification 251. All models used the same architecture and the same audio input sizes.

| Method | ACC |
|---|---|
| **#steps predicted** | |
| 2 steps | 28.5 |
| 4 steps | 57.6 |
| 8 steps | 63.6 |
| 12 steps | 64.6 |
| 16 steps | 63.8 |
| **Negative samples from** | |
| Mixed speaker | 64.6 |
| Same speaker | 65.5 |
| Mixed speaker (excl.) | 57.3 |
| Same speaker (excl.) | 64.6 |
| Current sequence only | 65.2 |

Table 2: LibriSpeech phone classification ablation experiments. More details can be found in Section 3.1.

# CPC - ImageNet

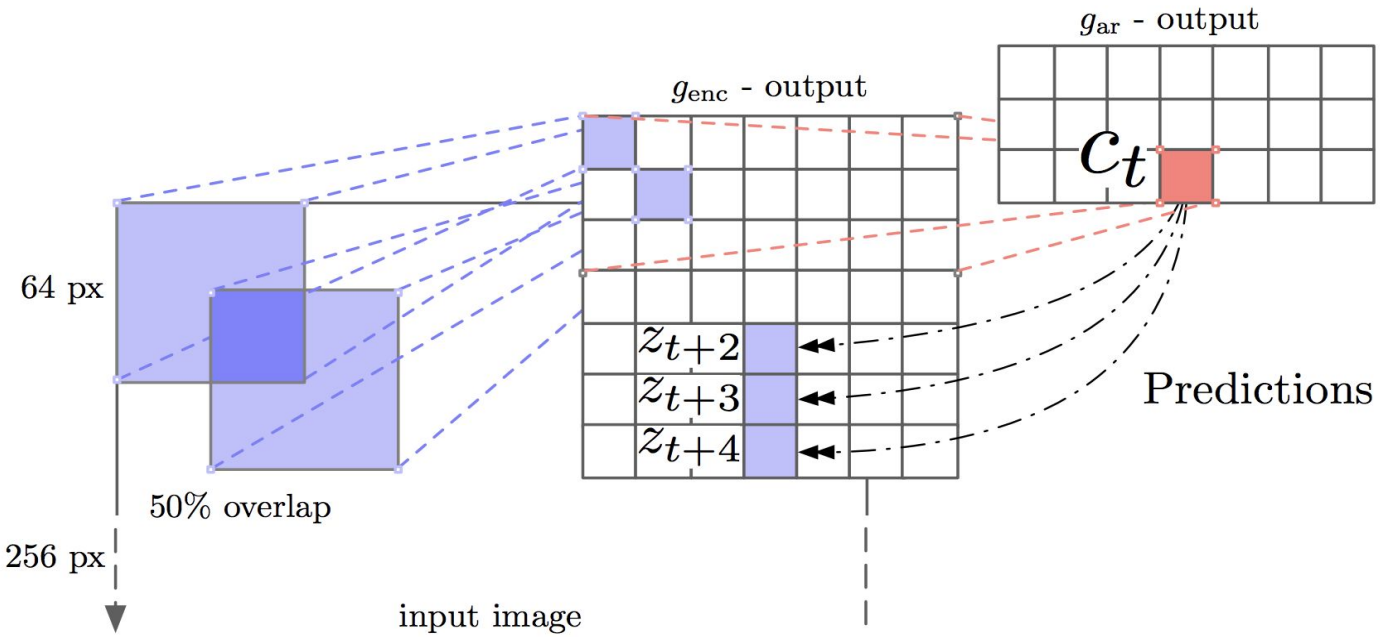



Figure 4: Visualization of Contrastive Predictive Coding for images (2D adaptation of Figure 1).

# CPC - ImageNet

# CPC - ImageNet

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

Table 3: ImageNet top-1 unsupervised classification results. *Jigsaw is not directly comparable to the other AlexNet results because of architectural differences.

| Method | Top-5 ACC |
|---|---|
| Motion Segmentation (MS) | 48.3 |
| Exemplar (Ex) | 53.1 |
| Relative Position (RP) | 59.2 |
| Colorization (Col) | 62.5 |
| Combination of MS + Ex + RP + Col | 69.3 |
| **CPC** | **73.6** |

Table 4: ImageNet top-5 unsupervised classification results. Previous results with MS, Ex, RP and Col were taken from [36] and are the best reported results on this task.

# CPC - ImageNet

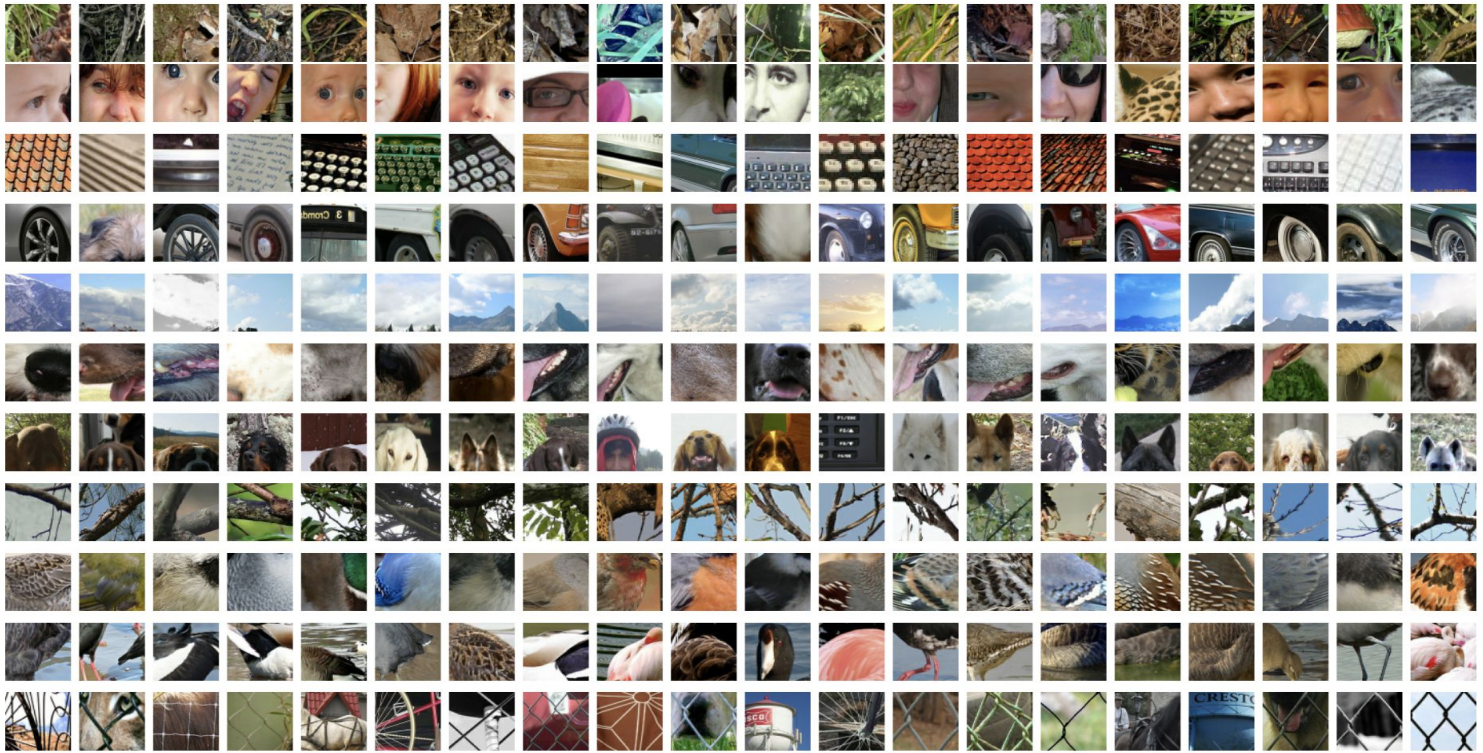


Figure 5: Every row shows image patches that activate a certain neuron in the CPC architecture.

# CPC - Natural Language Processing

| Method | MR | CR | Subj | MPQA | TREC |
|---|---|---|---|---|---|
| Paragraph-vector [40] | 74.8 | 78.1 | 90.5 | 74.2 | 91.8 |
| Skip-thought vector [26] | 75.5 | 79.3 | 92.1 | 86.9 | 91.4 |
| Skip-thought + LN [41] | 79.5 | 82.6 | 93.4 | 89.0 | - |
| CPC | 76.9 | 80.1 | 91.2 | 87.7 | 96.8 |

Table 5: Classification accuracy on five common NLP benchmarks. We follow the same transfer learning setup from Skip-thought vectors [26] and use the BookCorpus dataset as source. [40] is an unsupervised approach to learning sentence-level representations. [26] is an alternative unsupervised learning approach. [41] is the same skip-thought model with layer normalization trained for 1M iterations.

Oord, Li, Vinyals 2018

# CPC - Reinforcement Learning

Auxiliary loss is on policy
Predict 30 steps in the future

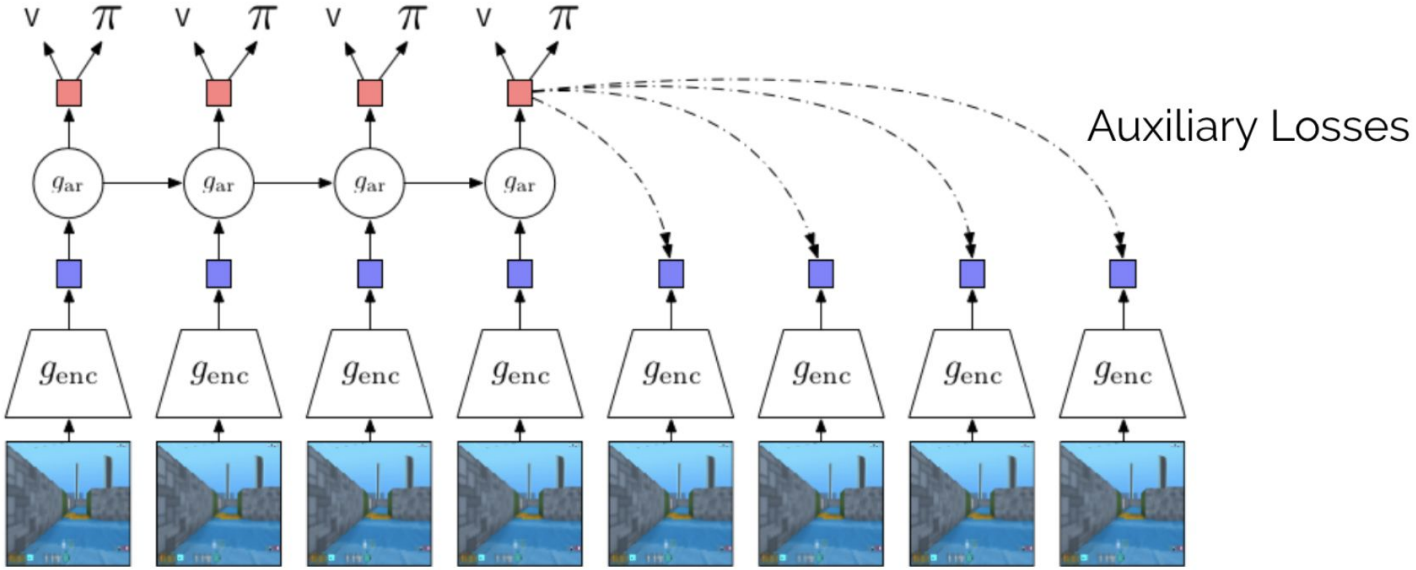



Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet

## Data-Efficient Image Recognition with Contrastive Predictive Coding

**Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, Aaron van den Oord**
DeepMind
London, UK

### Abstract

Human observers can learn to recognize new categories of images from a handful of examples, yet doing so with machine perception remains an open challenge. We hypothesize that data-efficient recognition is enabled by representations which make the variability in natural signals more predictable. We therefore revisit and improve Contrastive Predictive Coding, an unsupervised objective for learning such representations. This new implementation produces features which support state-of-the-art linear classification accuracy on the ImageNet dataset. When used as input for non-linear classification with deep neural networks, this representation allows us to use 2–5× less labels than classifiers trained directly on image pixels. Finally, this unsupervised representation substantially improves transfer learning to object detection on PASCAL VOC-2007, surpassing fully supervised pre-trained ImageNet classifiers.

# CPCv2 - Large Scale CPC on ImageNet

Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet

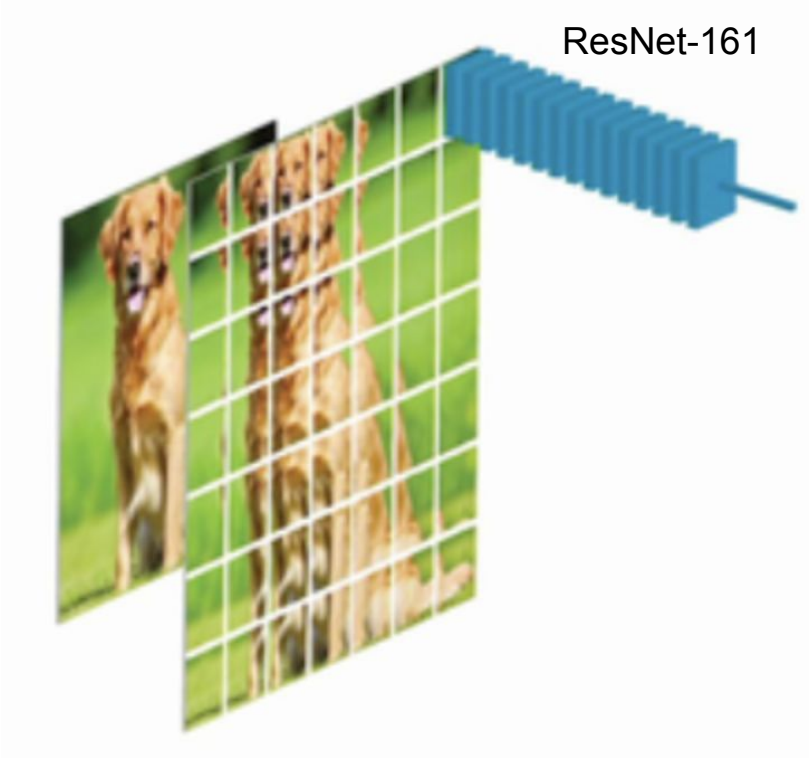



Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet

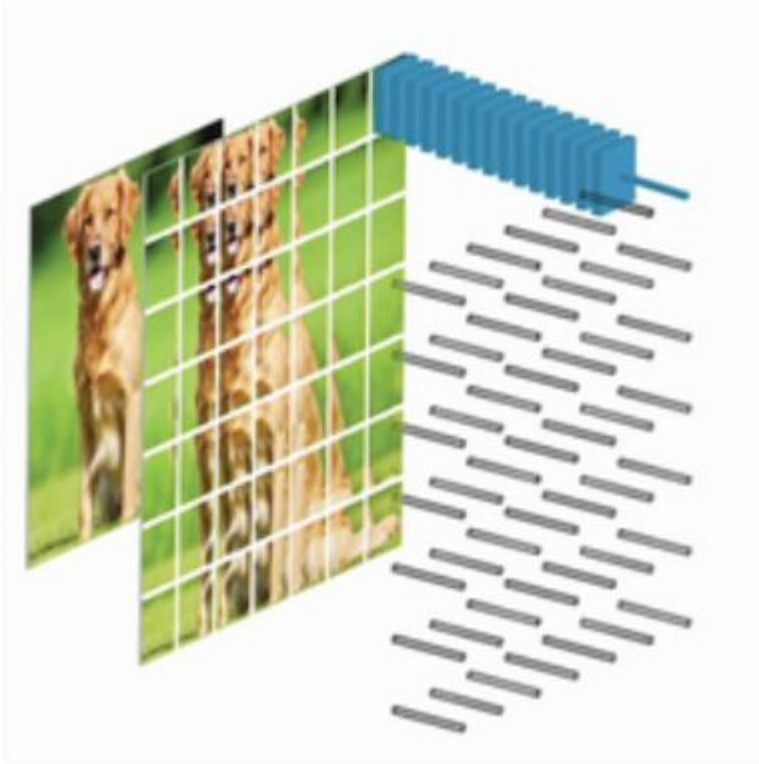


Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet

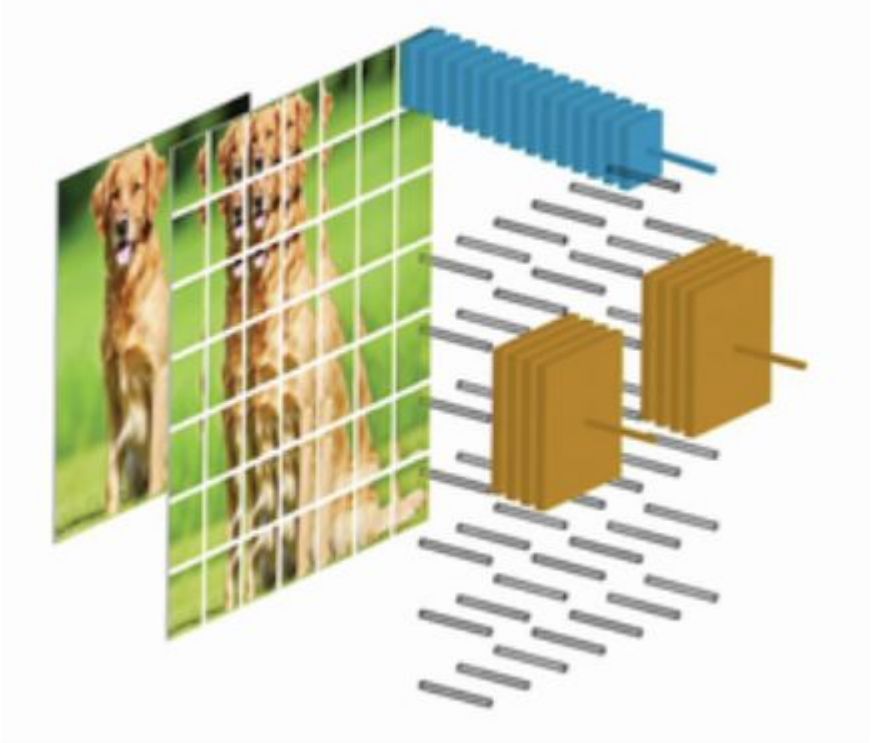


Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet

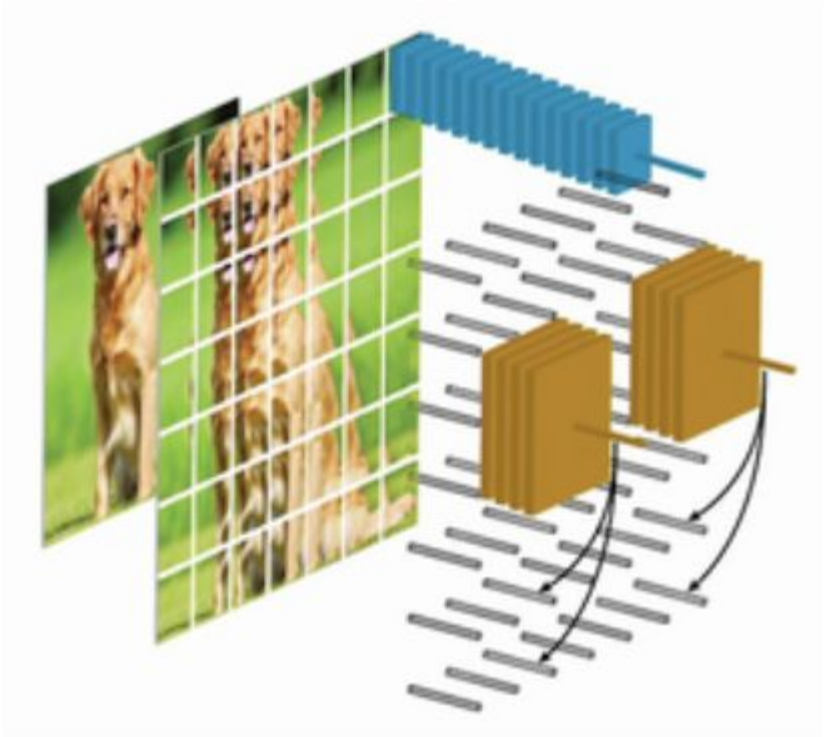


Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet

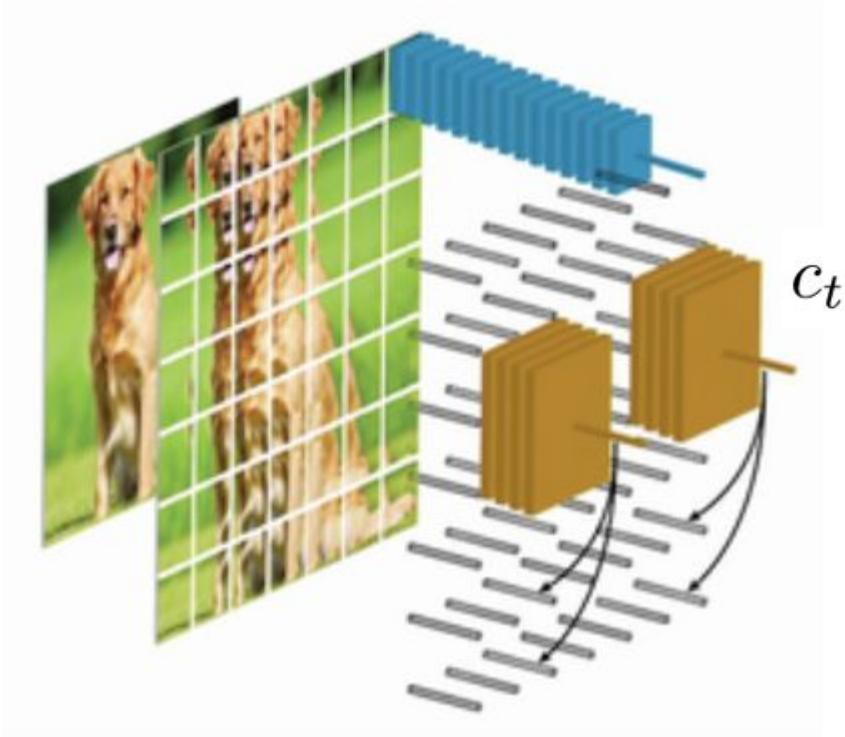



Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet

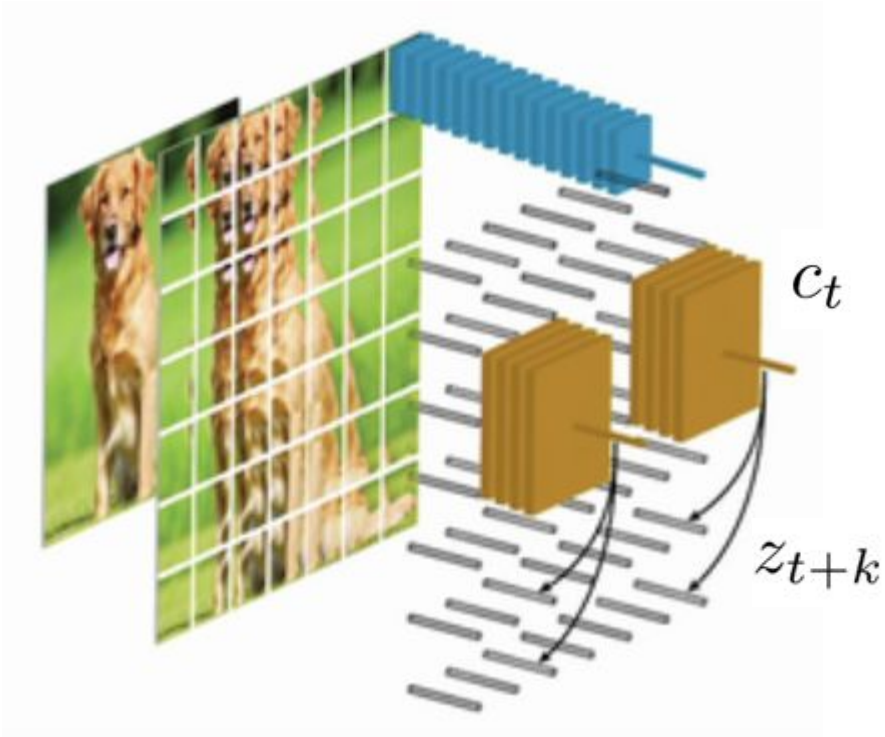
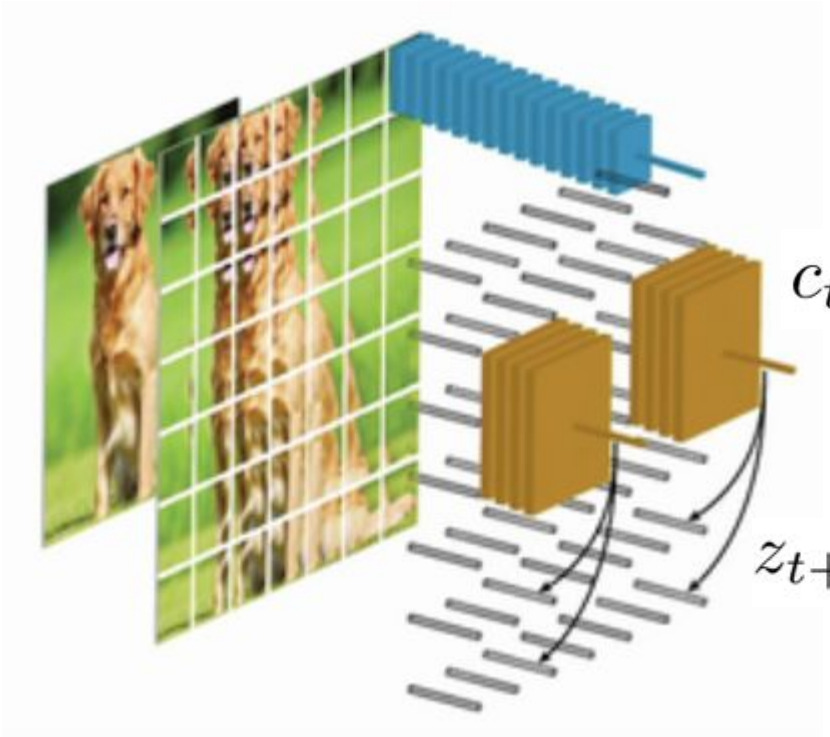



Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet




$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

Figure from Aaron Van den Oord

# CPCv2 - Large Scale CPC on ImageNet

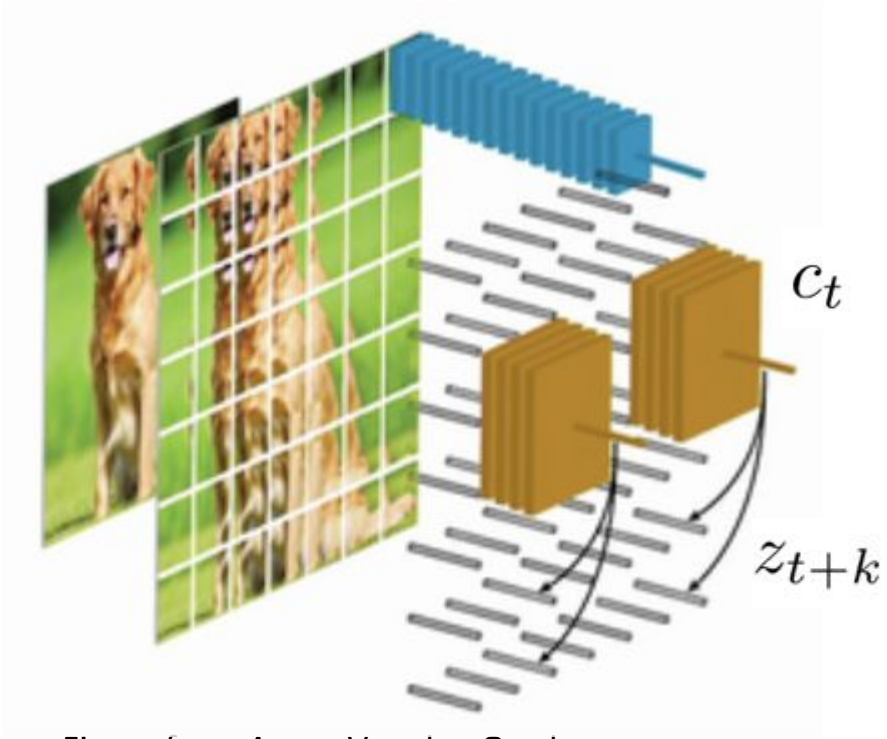



Figure from Aaron Van den Oord

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_{\mathrm{N}} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

# CPCv2 - Large Scale CPC on ImageNet

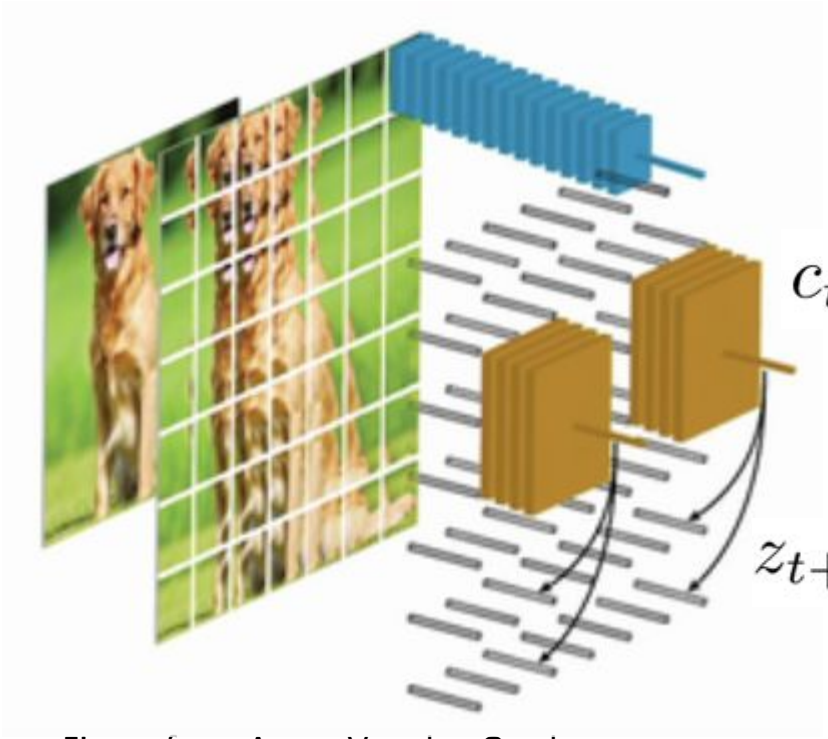



Figure from Aaron Van den Oord

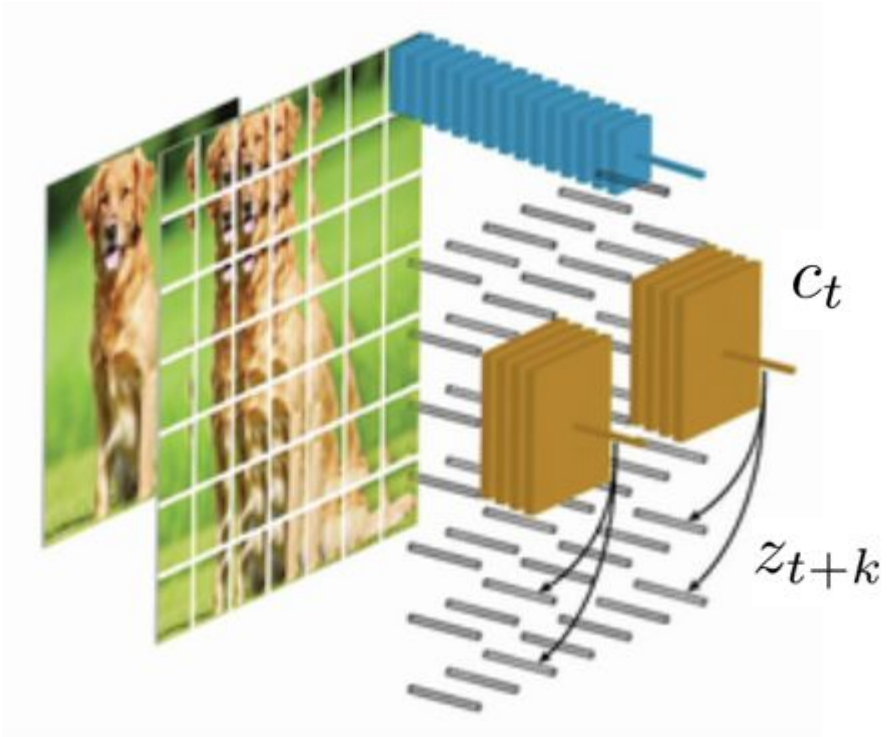
$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_X\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

Negatives

1. Other patches within image
2. Patches from other images

# CPCv2 - Large Scale CPC on ImageNet




**InfoNCE Loss**

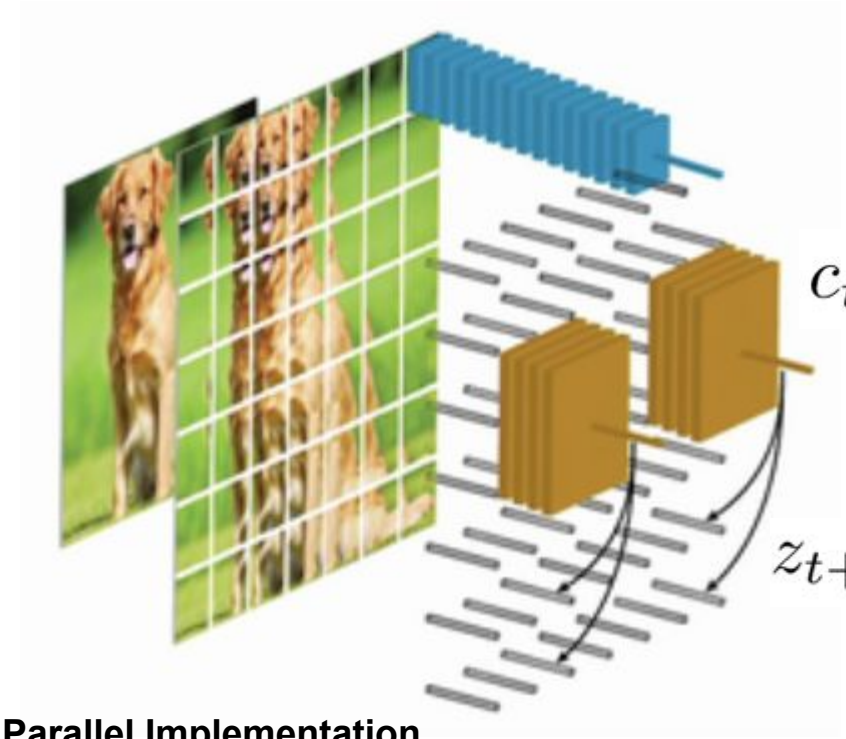
$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_X\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

Negatives

1. Other patches within image
2. Patches from other images

# CPCv2 - Large Scale CPC on ImageNet




**Parallel Implementation with PixelCNN (masked conv) and 1x1 conv**

**InfoNCE Loss**

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_X\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

Negatives

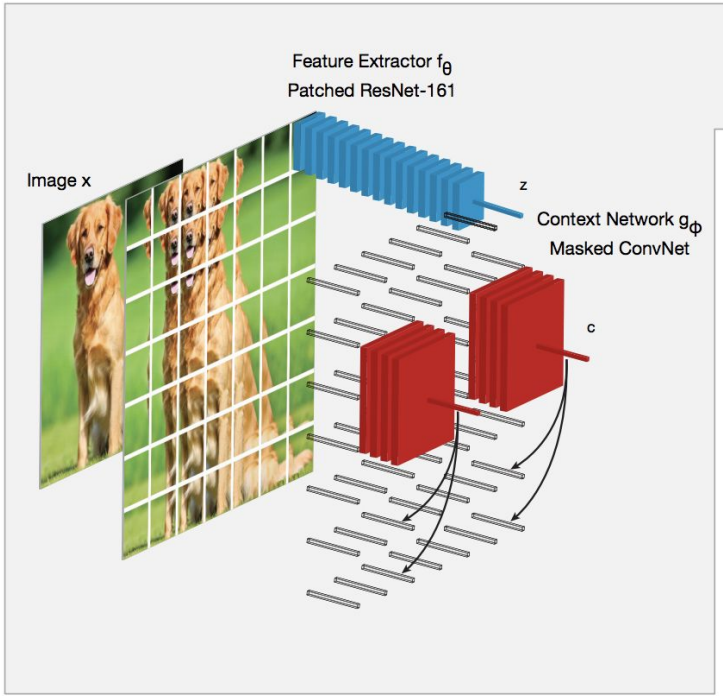
1. Other patches within image
2. Patches from other images

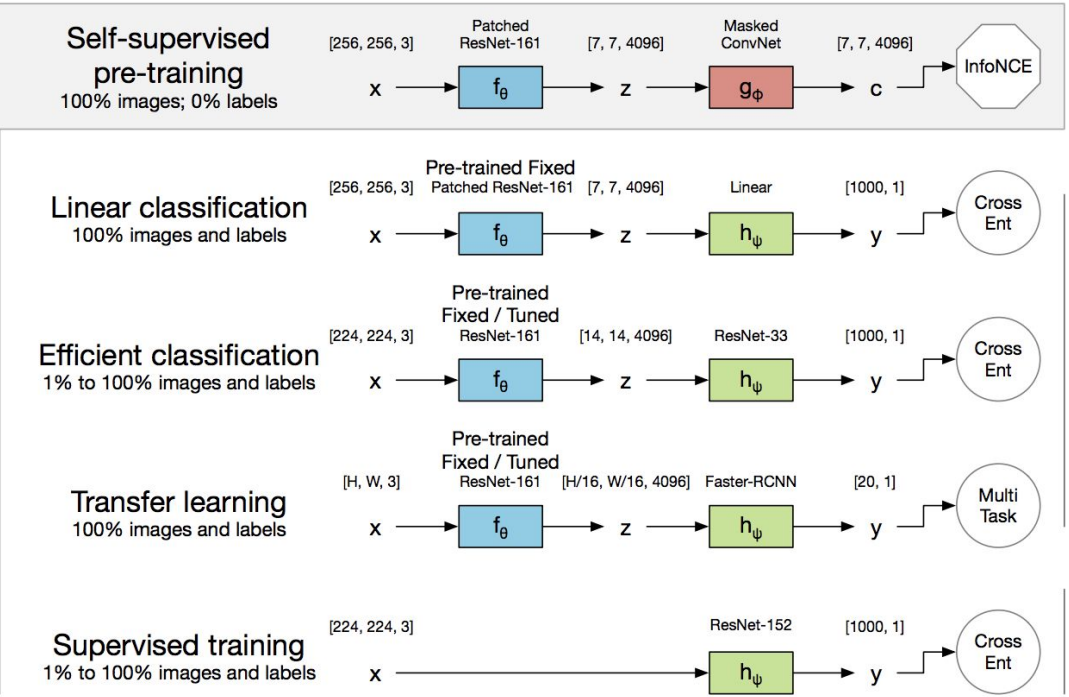
# CPCv2 - Large Scale CPC on ImageNet

## Contrastive Predictive Coding 2.0 (CPCv2)

- Train CPC on unlabeled ImageNet
- Train as long as possible (500 epochs) - 1 week
- Augment every patch with a lot of spatial and color augmentation [**extremely crucial**]
- Effective number of negatives = number of instances * number of patches per instance = 16 * 36 = 576

# CPCv2 - Large Scale CPC on ImageNet

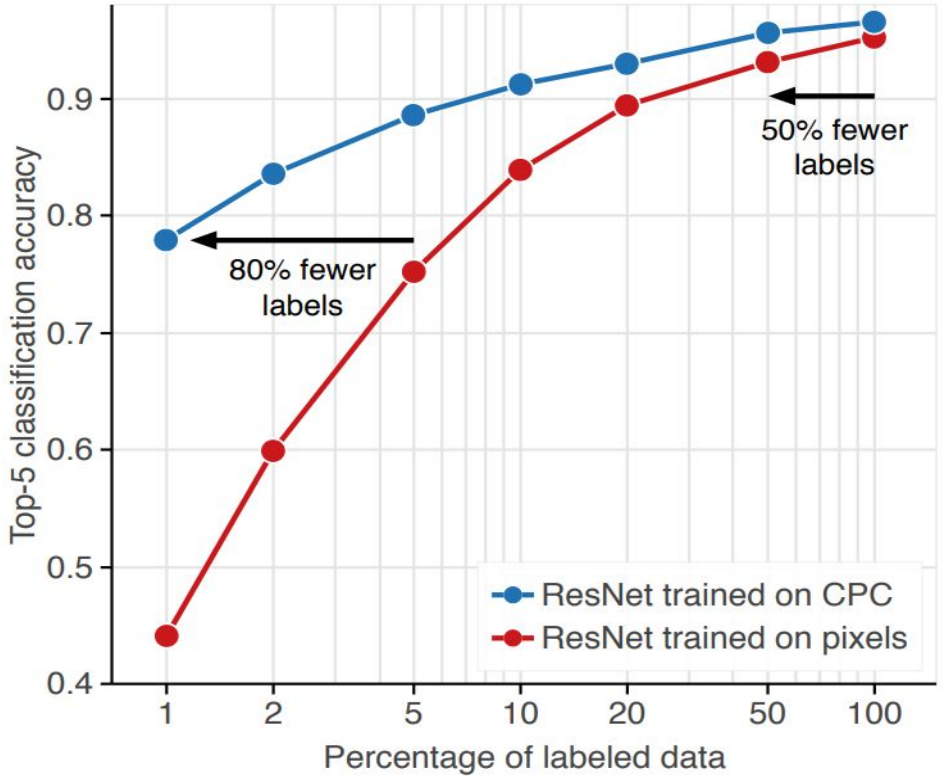


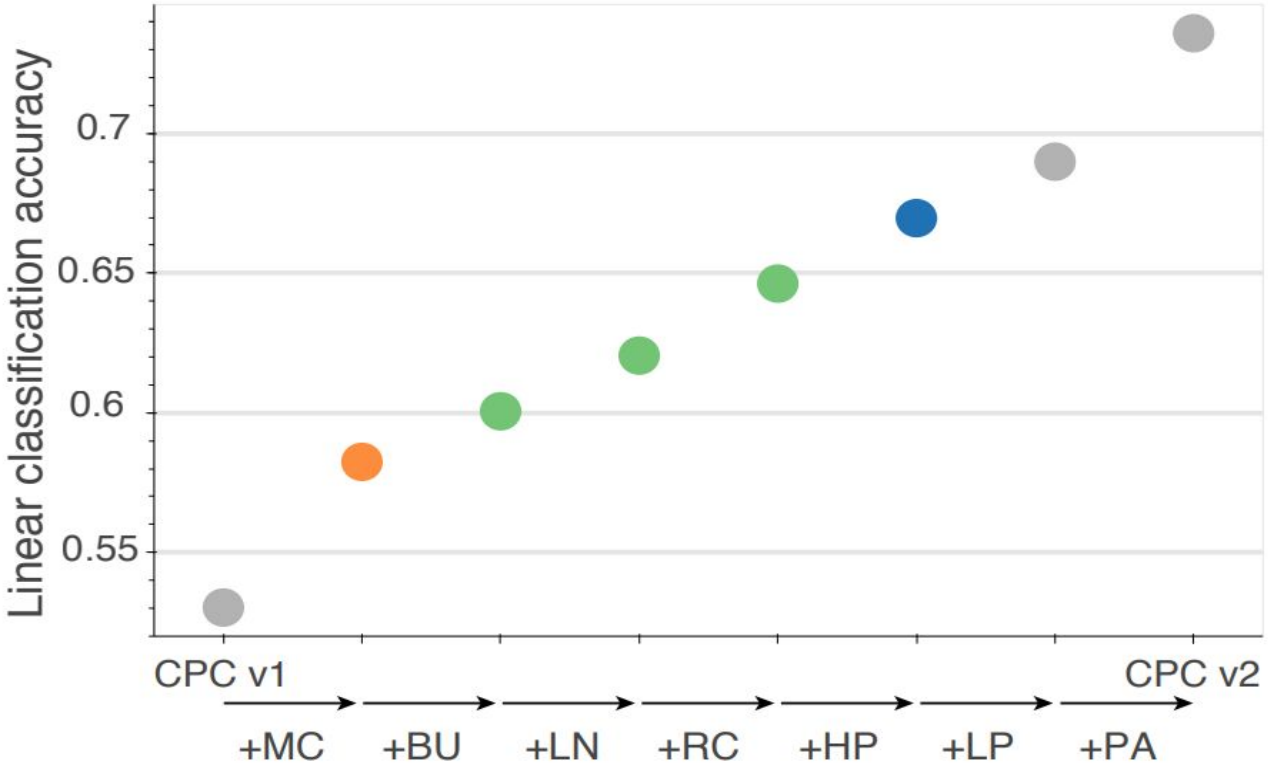
# CPCv2 - Linear Classification

| Method | Architecture | Params. (M) | Top-1 Acc. |
|---|---|---|---|
| Motion Segmentation | ResNet-101 | 28 | 27.6 |
| Exemplar | ResNet-101 | 28 | 31.5 |
| Relative Position | ResNet-101 | 28 | 36.2 |
| Colorization | ResNet-101 | 28 | 39.6 |
| CPC v1 | ResNet-101 | 28 | 48.7 |
| Rotation | RevNet-50 ×4 | 86 | 55.4 |
| BigBiGAN | RevNet-50 ×4 | 86 | 61.3 |
| AMDIM | Custom-103 | 626 | 68.1 |
| CMC | ResNet-50 ×2 | 188 | 68.4 |
| Momentum Contrast | ResNet-50 ×4 | 375 | 68.6 |
| **CPC v2** | ResNet-161 | 305 | **71.5** |
| Local Aggregation | ResNet-50 | 24 | 60.2 |
| Momentum Contrast | ResNet-50 | 24 | 60.6 |
| **CPC v2** | ResNet-50 | 24 | **63.8** |

# CPCv2 - Data-Efficient Image Recognition



# CPCv1 ---> CPCv2


Linear classification accuracy
0.7
0.65
0.6
0.55
CPC v1
CPC v2
+MC
+BU
+LN
+RC
+HP
+LP
+PA

# CPCv2 - Data-Efficient Supervised Learning

| Method | Architecture | Top-5 accuracy | | | | |
|---|---|---|---|---|---|---|
| Labeled data | | 1% | 5% | 10% | 50% | 100% |
| †Supervised baseline | ResNet-200 | 44.1 | 75.2⋆ | 83.9 | 93.1 | 95.2# |
| ***Methods using label-propagation:*** | | | | | | |
| Pseudolabeling [63] | ResNet-50 | 51.6 | - | 82.4 | - | - |
| VAT + Entropy Minimization [63] | ResNet-50 | 47.0 | - | 83.4 | - | - |
| Unsup. Data Augmentation [61] | ResNet-50 | - | - | 88.5 | - | - |
| Rotation + VAT + Ent. Min. [63] | ResNet-50 ×4 | - | - | **91.2** | - | 95.0 |
| ***Methods using representation learning only:*** | | | | | | |
| Instance Discrimination [60] | ResNet-50 | 39.2 | - | 77.4 | - | - |
| Rotation [63] | ResNet-152 ×2 | 57.5 | - | 86.4 | - | - |
| ResNet on BigBiGAN (fixed) | RevNet-50 ×4 | 55.2 | 73.7 | 78.8 | 85.5 | 87.0 |
| ResNet on AMDIM (fixed) | Custom-103 | 67.4 | 81.8 | 85.8 | 91.0 | 92.2 |
| ResNet on CPC v2 (fixed) | ResNet-161 | 77.1 | 87.5 | 90.5 | 95.0 | 96.2 |
| ResNet on CPC v2 (fine-tuned) | ResNet-161 | **77.9**⋆ | **88.6** | **91.2** | **95.6**# | **96.5** |

# CPCv2 - PASCAL VOC-07 Detection

| Method | Architecture | mAP |
|---|---|---|
| ***Transfer from labeled data:*** | | |
| Supervised baseline | ResNet-152 | 74.7 |
| ***Transfer from unlabeled data:*** | | |
| Exemplar [17] by [13] | ResNet-101 | 60.9 |
| Motion Segmentation [47] by [13] | ResNet-101 | 61.1 |
| Colorization [64] by [13] | ResNet-101 | 65.5 |
| Relative Position [14] by [13] | ResNet-101 | 66.8 |
| Multi-task [13] | ResNet-101 | 70.5 |
| Instance Discrimination [60] | ResNet-50 | 65.4 |
| Deep Cluster [7] | VGG-16 | 65.9 |
| Deeper Cluster [8] | VGG-16 | 67.8 |
| Local Aggregation [66] | ResNet-50 | 69.1 |
| Momentum Contrast [25] | ResNet-50 | 74.9 |
| Faster-RCNN trained on CPC v2 | ResNet-161 | **76.6** |

# Instance Discrimination

attract

repel

# Instance Discrimination

attract

repel

1. MoCo
2. SimCLR

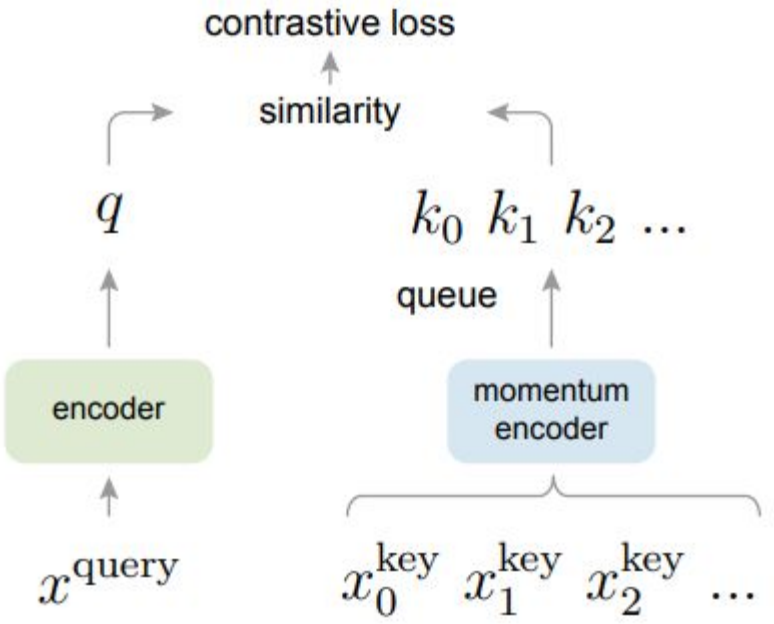
# Momentum Contrast (MoCo)

**Momentum Contrast for Unsupervised Visual Representation Learning**

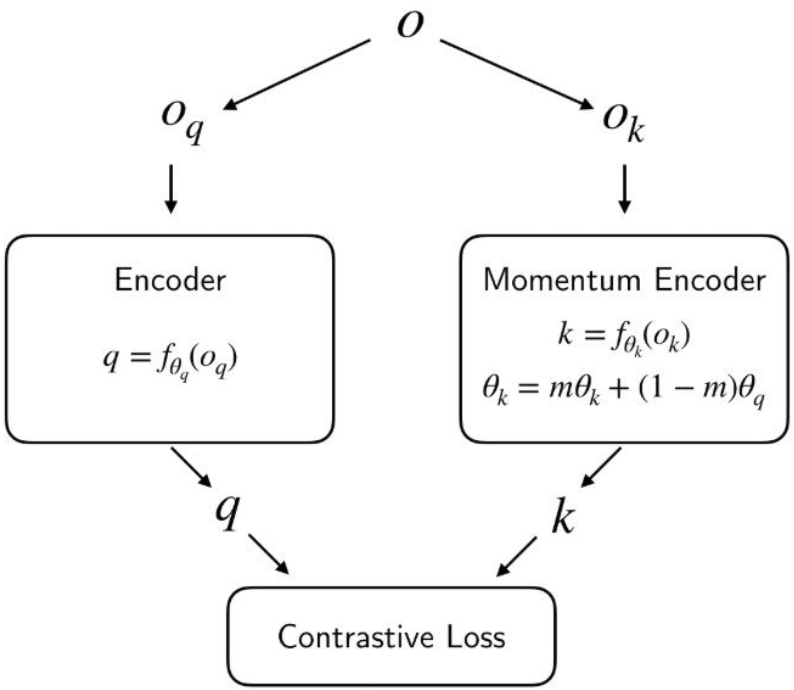
Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick

Facebook AI Research (FAIR)

# Momentum Contrast (MoCo)



# Momentum Contrast (MoCo)




$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$

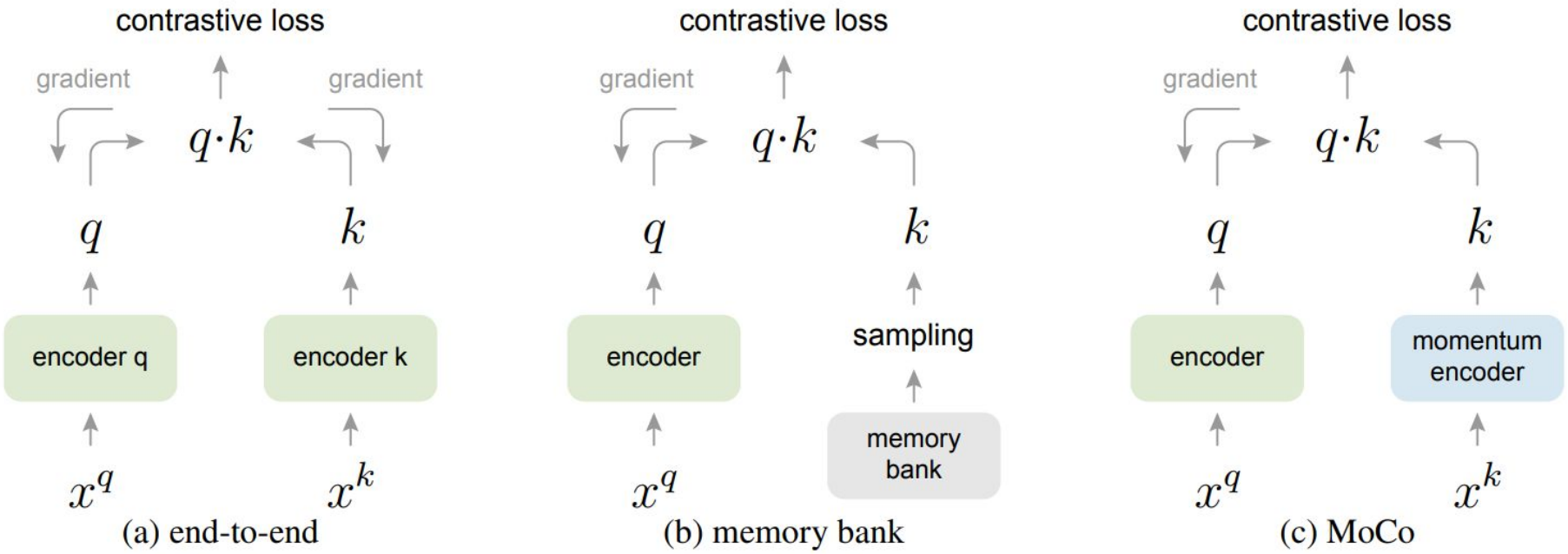
# Momentum Contrast (MoCo)

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

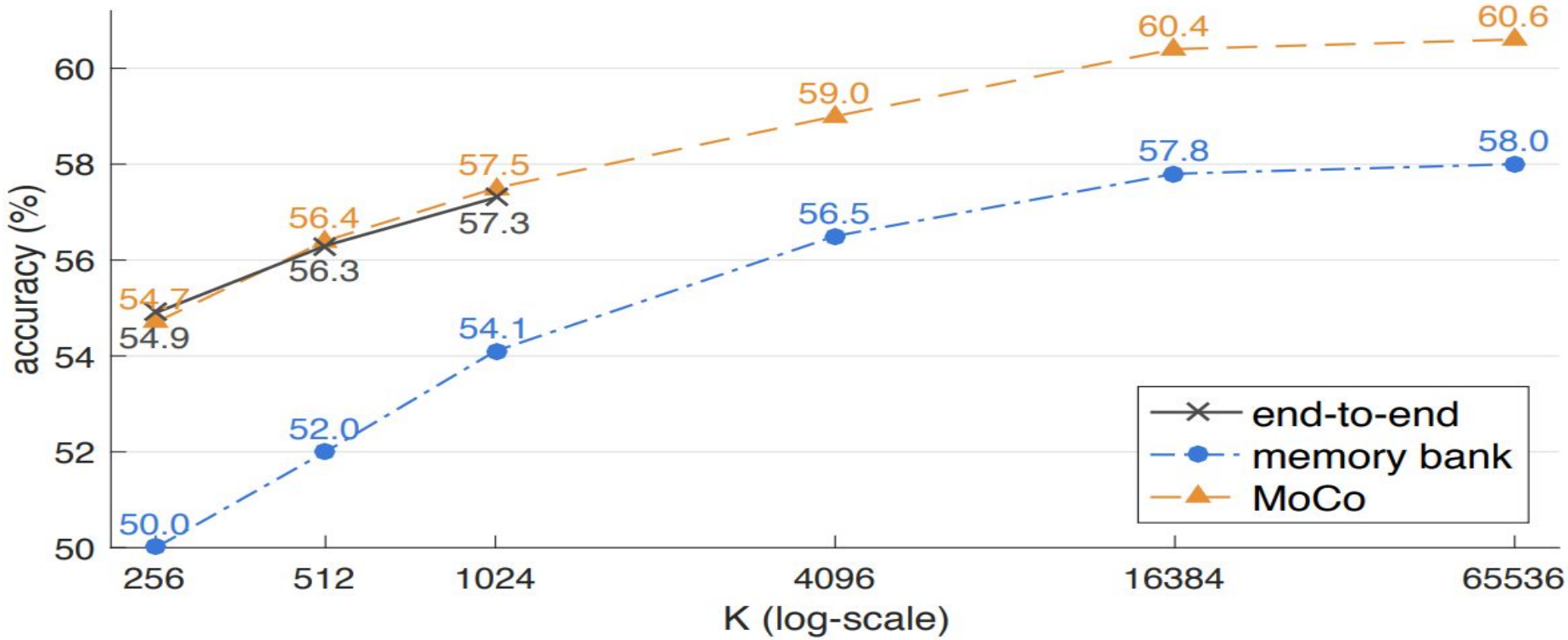
```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

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

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

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

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

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

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

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

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

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

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

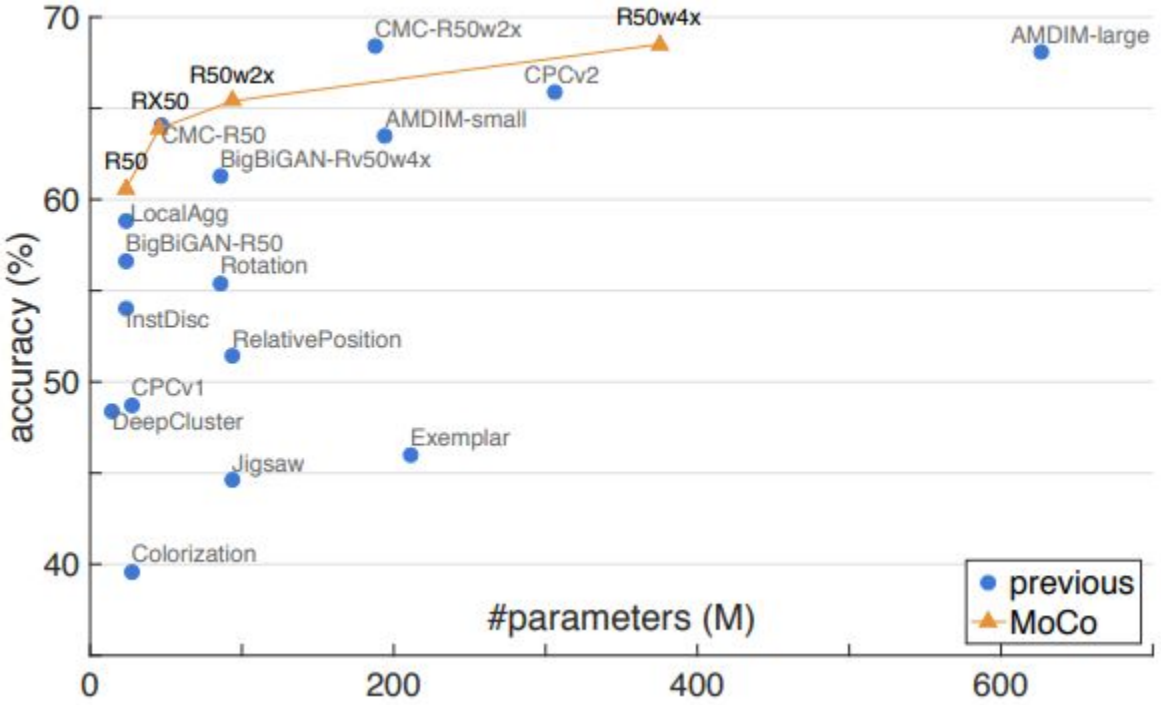
# Momentum Contrast (MoCo)



# Momentum Contrast (MoCo)



# Momentum Contrast (MoCo)

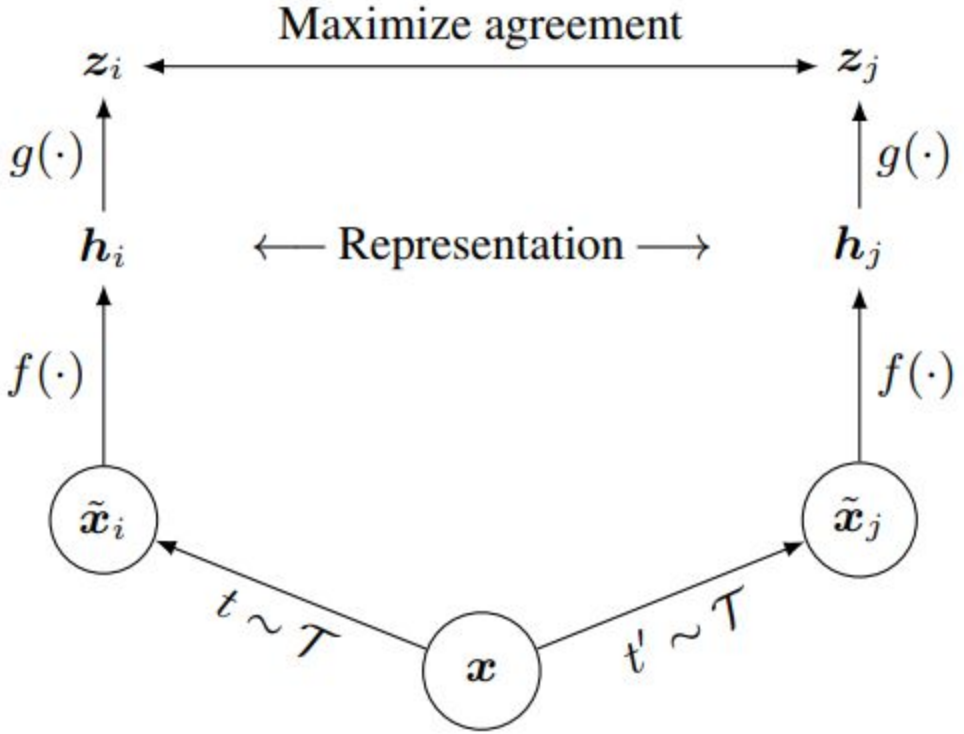


# SimCLR

## A Simple Framework for Contrastive Learning of Visual Representations

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

# SimCLR

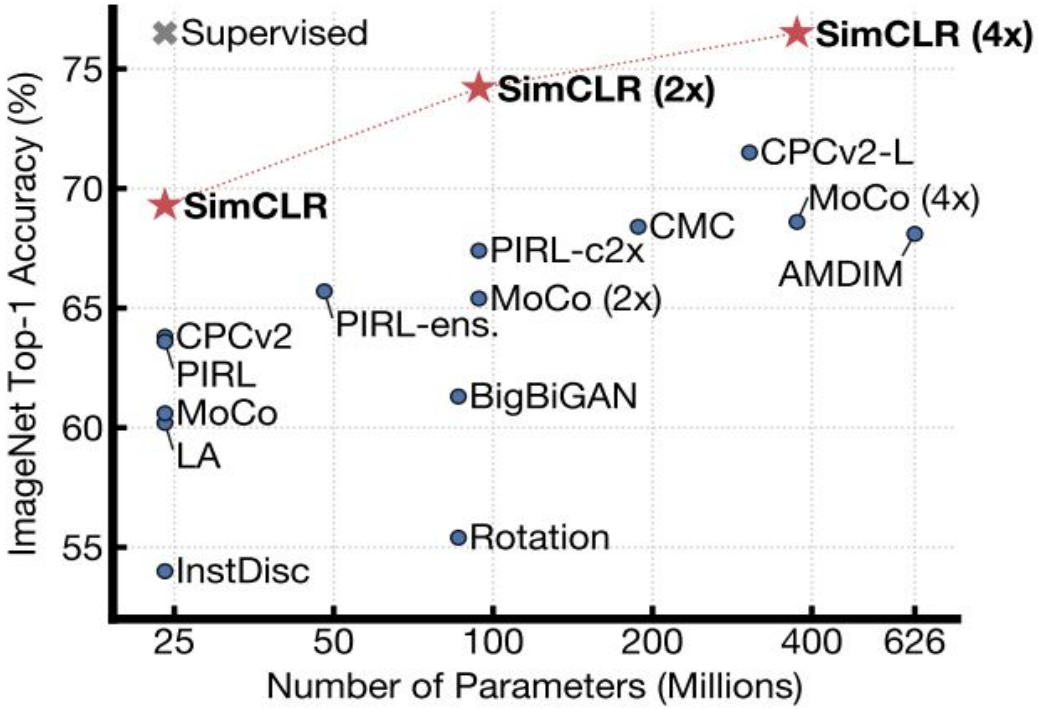

Maximize agreement
$z_i$
$z_j$
$g(\cdot)$
$g(\cdot)$
$h_i$
Representation
$h_j$
$f(\cdot)$
$f(\cdot)$
$\tilde{x}_i$
$\tilde{x}_j$
$t \sim \mathcal{T}$
$t' \sim \mathcal{T}$
$x$

# SimCLR

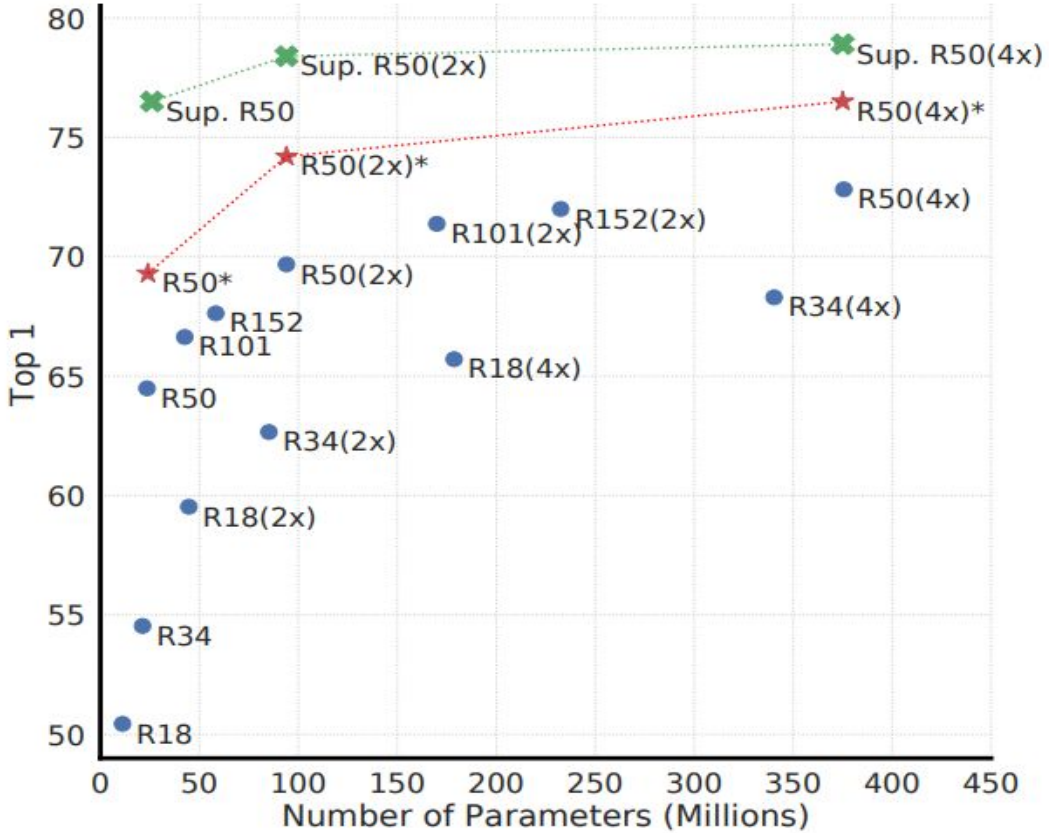
**Algorithm 1** SimCLR's main learning algorithm.

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

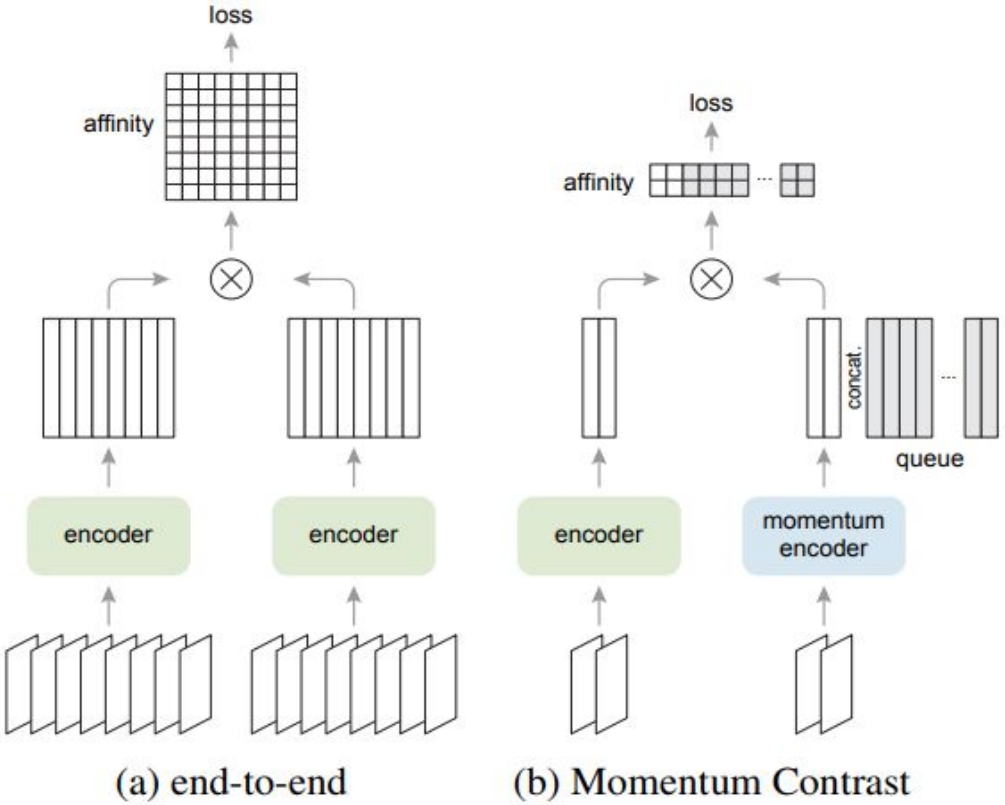
# SimCLR


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)

# SimCLR


Top 1
80
75
70
65
60
55
50
0
50
100
150
200
250
300
350
400
450
Number of Parameters (Millions)
Sup. R50(4x)
Sup. R50(2x)
Sup. R50
R50(4x)*
R50(2x)*
R50(4x)
R152(2x)
R101(2x)
R50*
R50(2x)
R34(4x)
R152
R101
R18(4x)
R50
R34(2x)
R18(2x)
R34
R18

# MoCov2 vs SimCLR



# MoCov2 vs SimCLR

| 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** |

# MoCov2 vs SimCLR

| 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** |