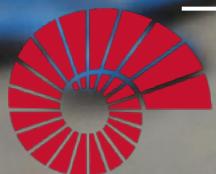


COMP547

DEEP UNSUPERVISED LEARNING

Lecture #12 – Self-Supervised Learning



KOÇ
UNIVERSITY

Aykut Erdem // Koç University // Spring 2021

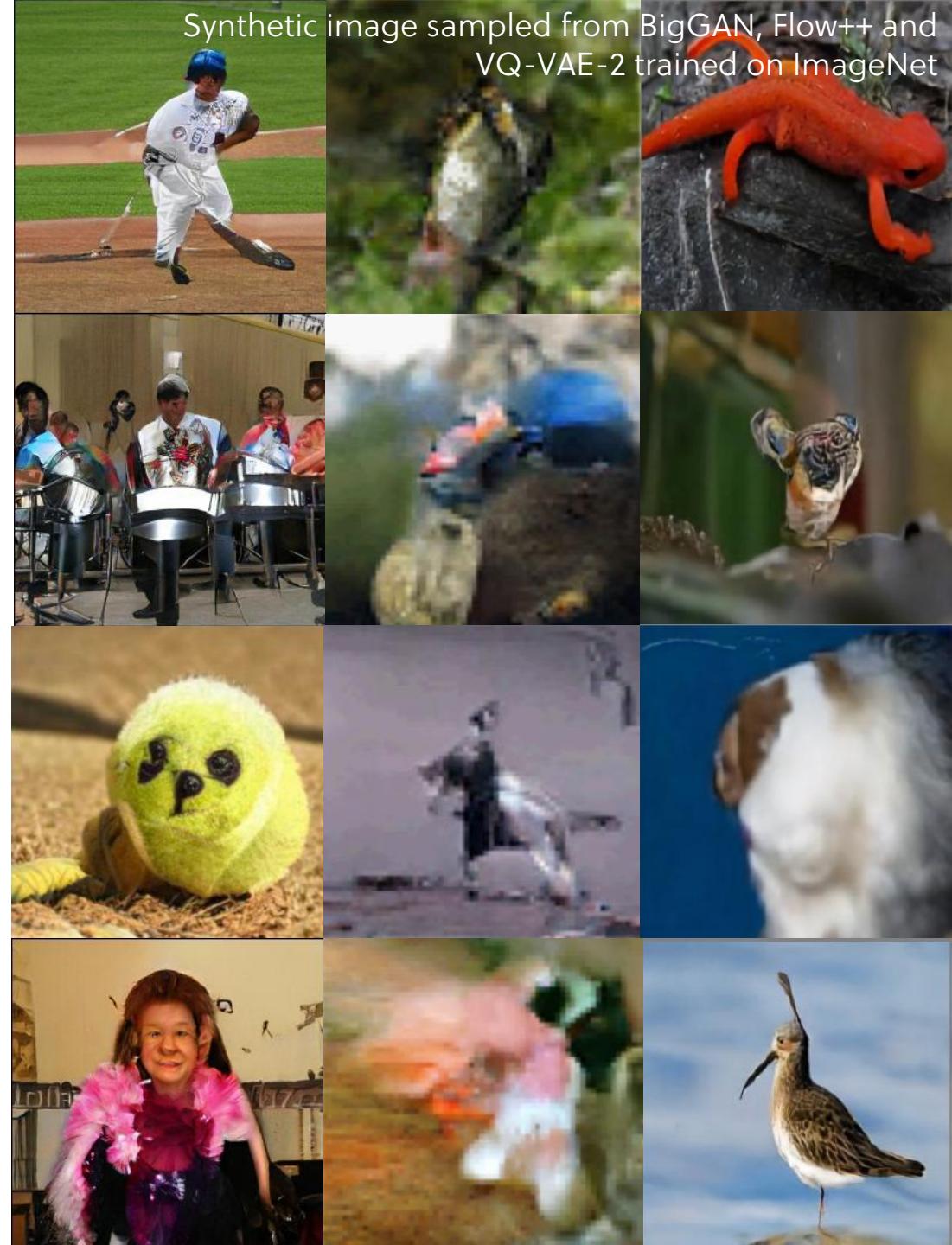
Good news, everyone!

- Don't forget to open a private channel on COMP547 workspace for your course projects!



Previously on COMP547

- Autoregressive models
- Flow models
- Latent Variable models
- Implicit models



Lecture overview

- Motivation
- Reconstruct from a corrupted (or partial) version
- Visual common-sense tasks
- Contrastive Learning

Disclaimer: Much of the material and slides for this lecture were borrowed from
—Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas' Berkeley CS294-158 class
—Aaron Courville's Université de Montréal IFT6268 class

Lecture overview

- Motivation
- Reconstruct from a corrupted (or partial) version
- Visual common-sense tasks
- Contrastive Learning

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 we 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?

Learning “really useful” representations

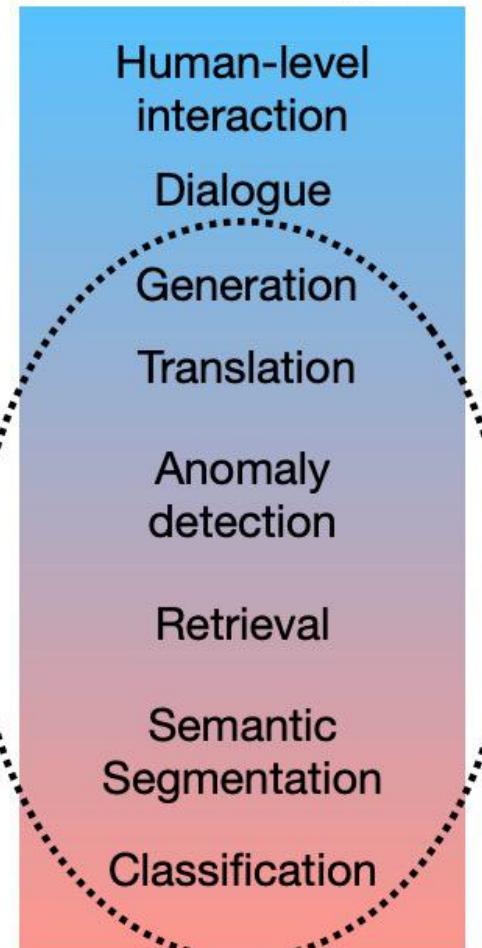
- Longstanding dream of the Deep Learning community:
 - Use unsupervised learning to learn some feature representation that can be used to support effective supervised learning (like classification)

Input



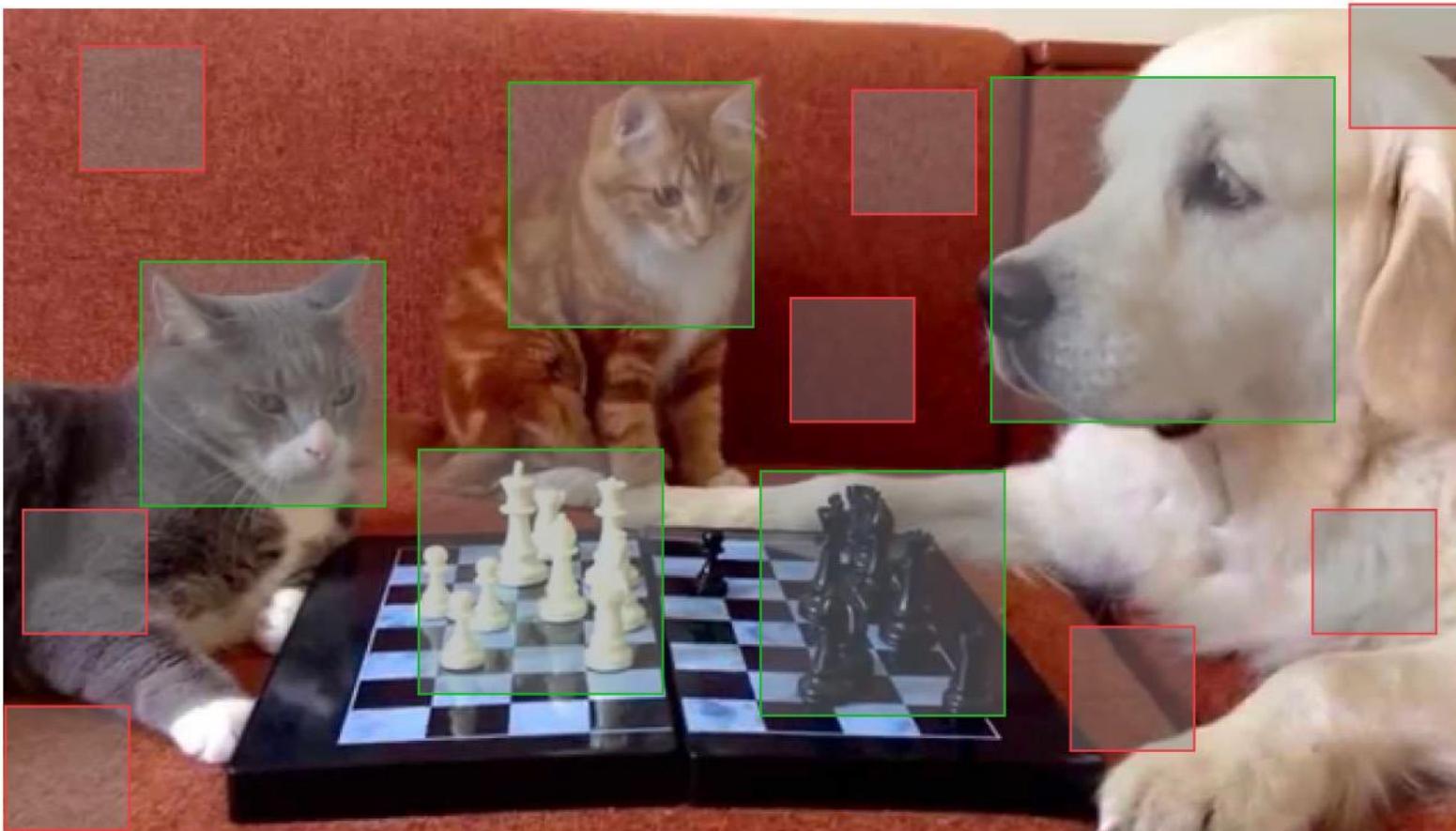
Extract
Information

Tasks



More utility = better
(Task) Generalization \Leftrightarrow Understanding

We don't need generation/reconstruction



- Interesting thing
- Not interesting thing

- Generative models (in principle) care about all the pixels

Representation Learning for Supervised Learning

- Using generative models (AEs, VAEs, etc) have largely been ineffective with two caveats:
 1. Natural Language Modelling (all SOTA models are build on BERT-like representations)
 2. In the very small dataset regime, unsupervised learning can actually help.
- Gradient-based supervised training with the right model (eg. CNNs for vision problems) has been very difficult to beat with unsupervised methods.

It's worth asking ... Why?

A speculative answer:

- Most (essentially all) existing unsupervised methods learn features that are overwhelmingly low-level (nonsemantic).
 - The features describe superficial aspects of the data and preserve few of the invariances that one would want from a representation learning scheme.
- Modern supervised learning methods (i.e. with NN) learn layers of representations that learn the relevant axes of variance in the data.
 - Eg. Higher level features of a CNN trained to recognize car makes and models should be relatively invariant to color but very sensitive to subtle differences in shape.

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

How Much Information Does the Machine Need to Predict?

Y LeCun

■ "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**

■ Unsupervised/Predictive Learning (cake)

- ▶ The machine predicts any part of its input for any observed part.
- ▶ Predicts future frames in videos
- ▶ **Millions of bits per sample**

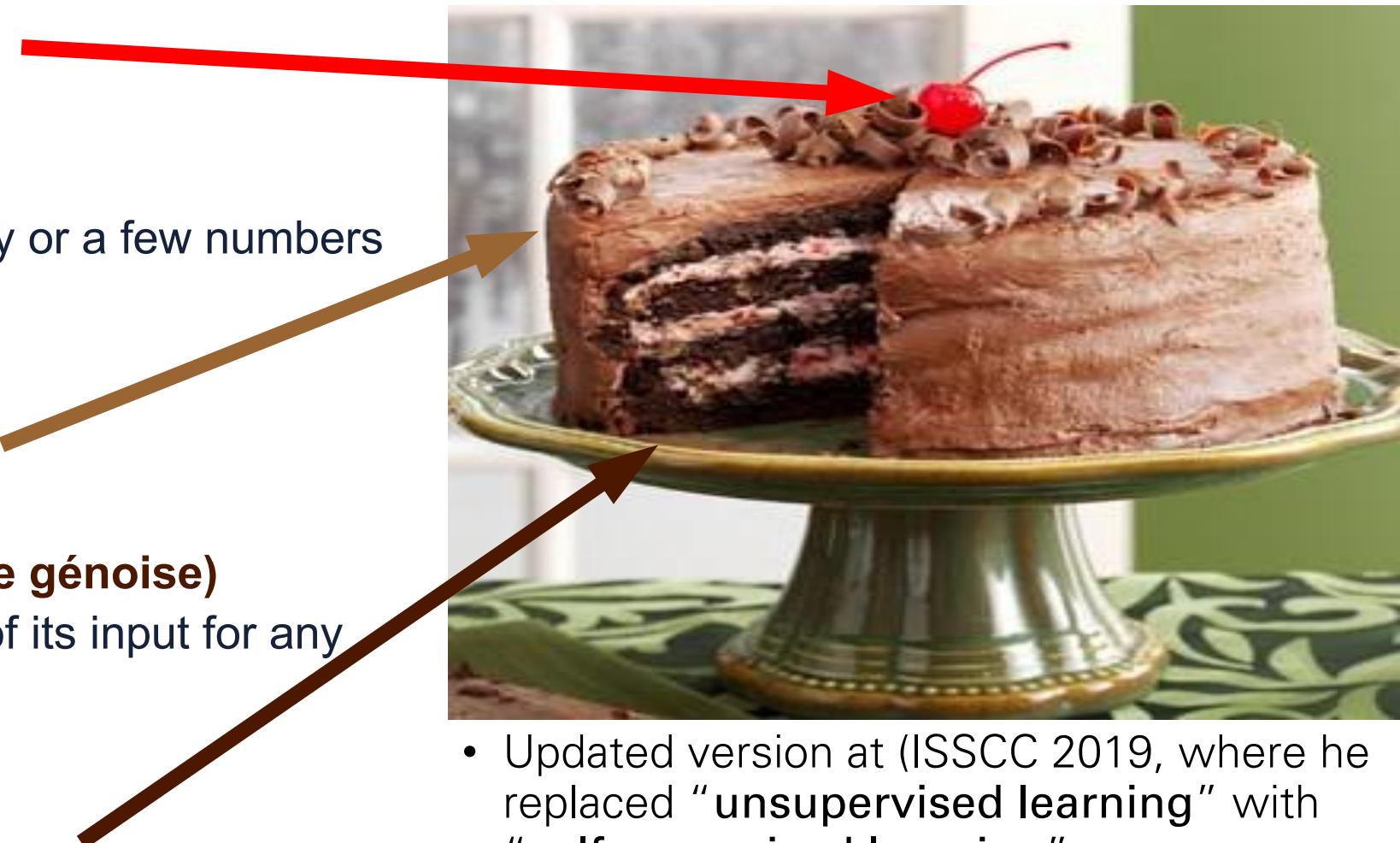


■ (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

- LeCun's original cake analogy slide, presented at his keynote speech in NIPS 2016.

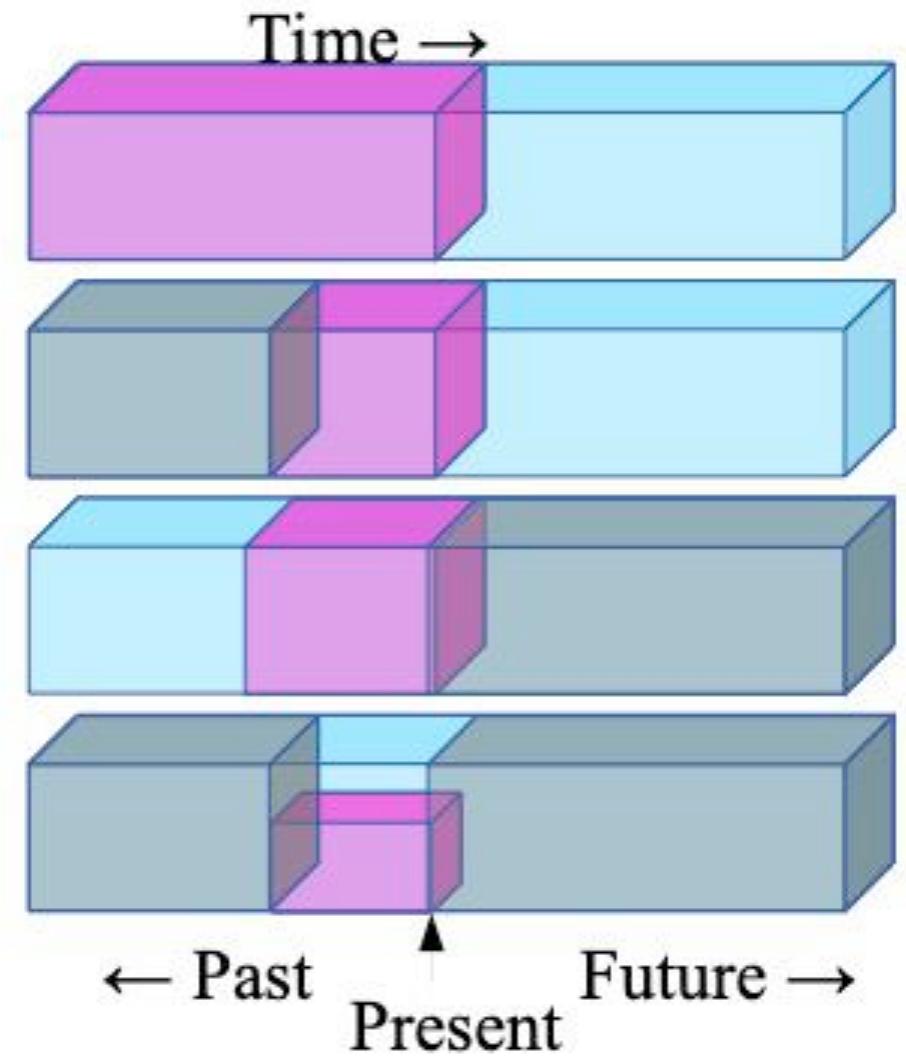
How Much Information is the Machine Given during Learning?

- ▶ “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**



Self-Supervised/Predictive Learning

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



What/Why Self-Supervision?

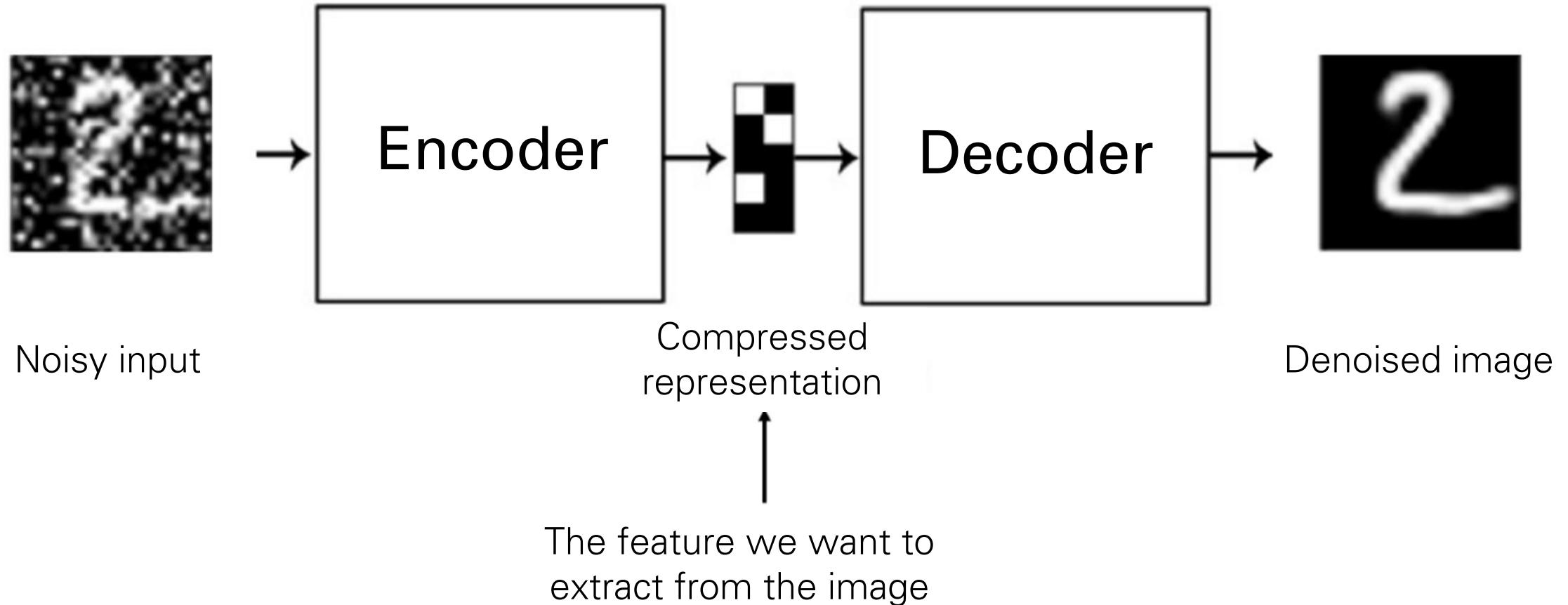
Self-supervision: Recover useful/semantic representations by training models to answer specific questions about the data.

- Good:
 - Can procedurally generate potentially infinite amounts of annotation.
 - We can borrow tricks from supervised learning without labels.
 - Focus on only the information that you need (e.g., not pixels).
 - Answering these questions requires more fundamental understanding of data.
- Not so good: designing good questions also requires some fundamental understanding of the data (e.g., structure).

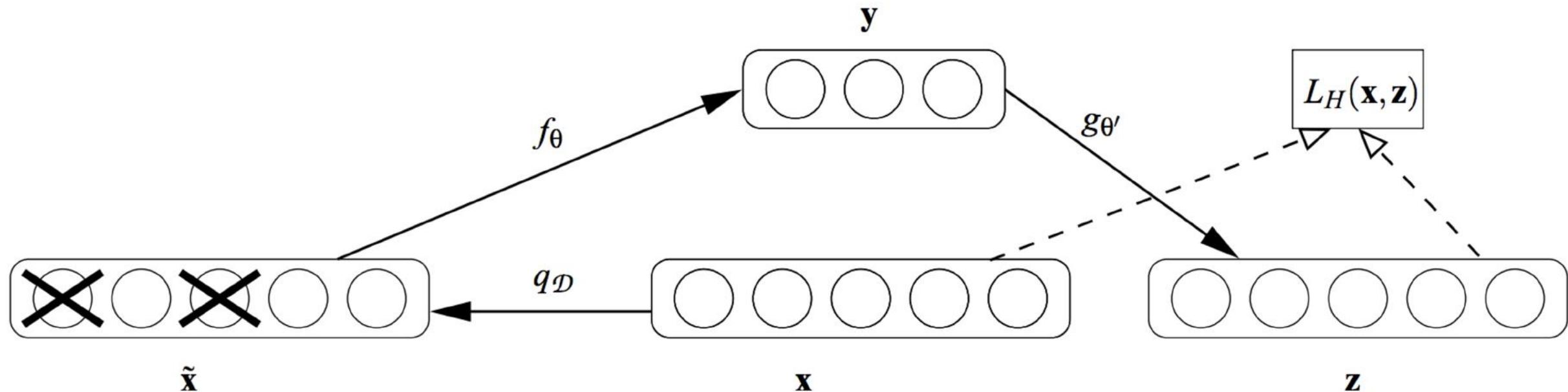
Lecture overview

- Motivation
- Reconstruct from a corrupted (or partial) version
 - Denoising Autoencoder
 - In-painting
 - Colorization, Split-Brain Autoencoder
- Visual common-sense tasks
- Contrastive Learning

Denoising Autoencoder



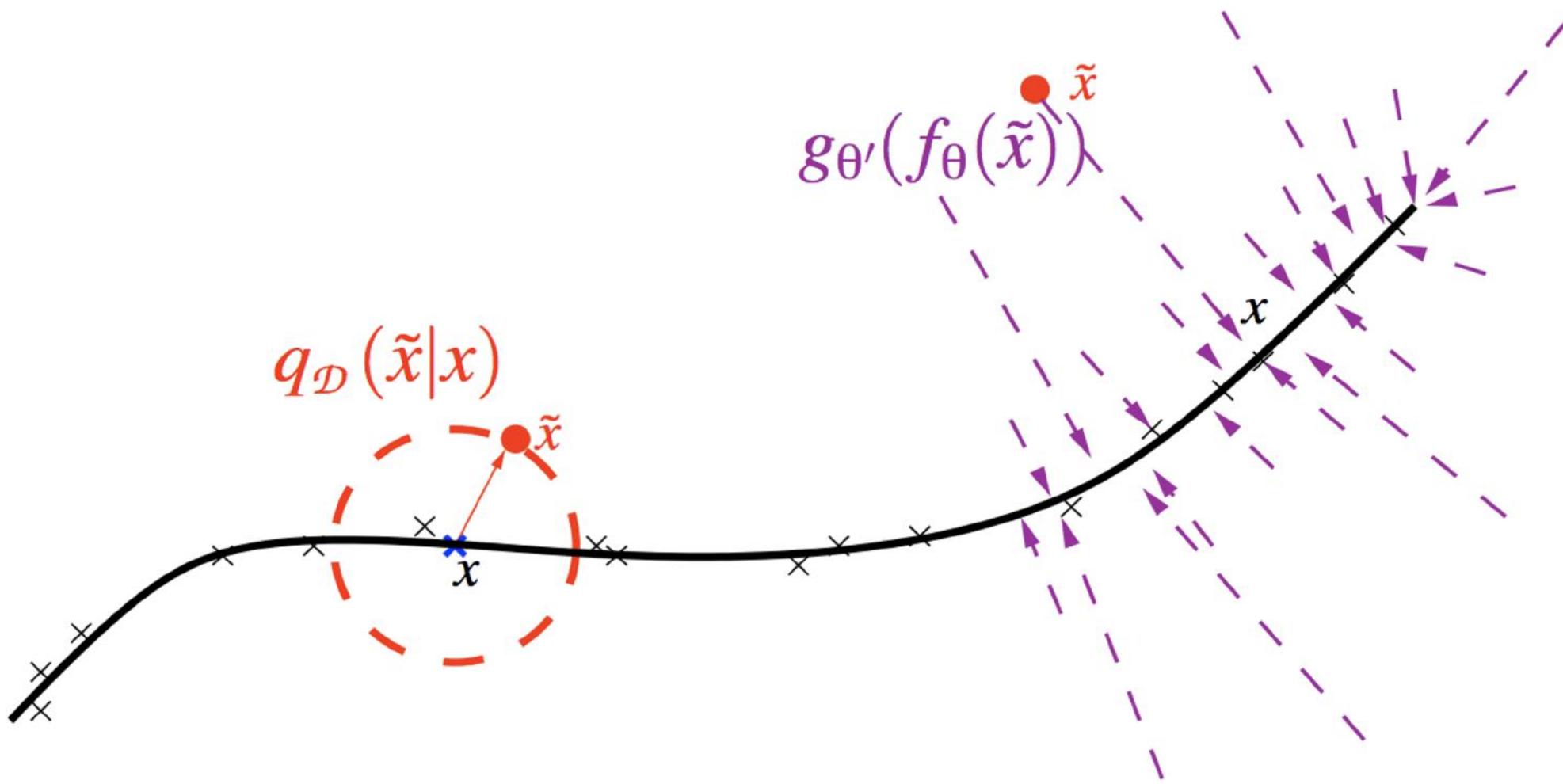
Denoising Autoencoder



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 v of the elements of \mathbf{x} (chosen at random for each example) is forced to 0;
 - *Salt-and-pepper noise* (SP): a fraction v 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.

Denoising Autoencoder

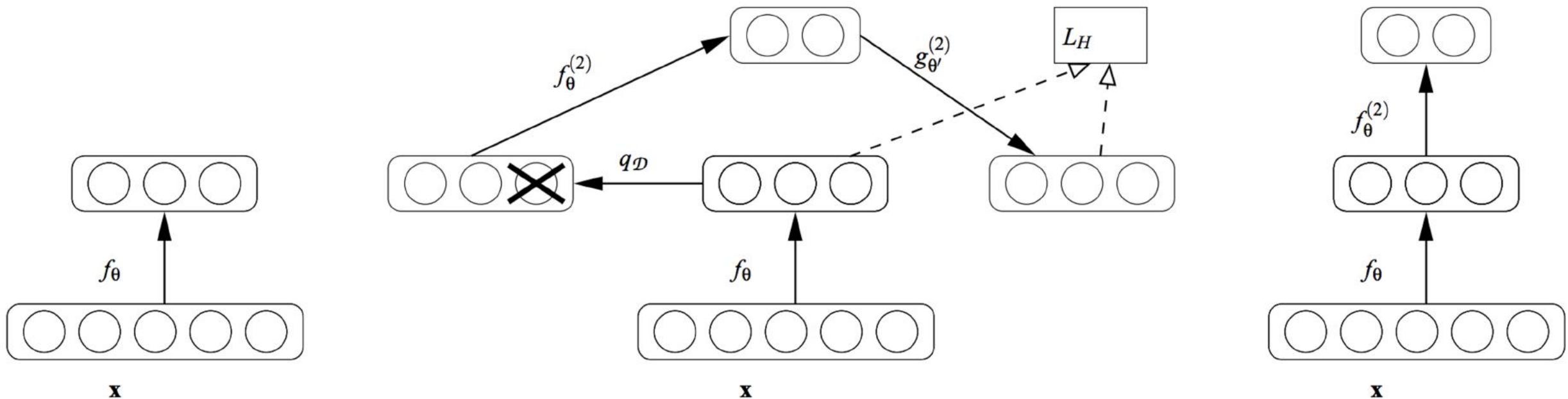


Emphasizing corrupted dimensions

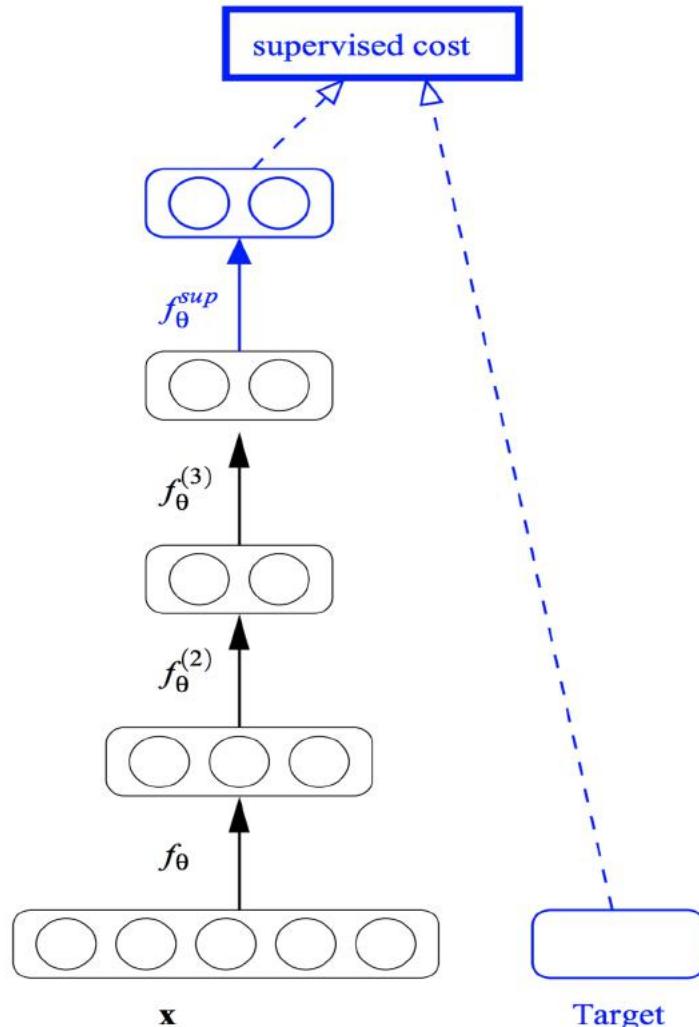
$$L_{2,\alpha}(\mathbf{x}, \mathbf{z}) = \alpha \left(\sum_{j \in \mathcal{J}(\tilde{\mathbf{x}})} (\mathbf{x}_j - \mathbf{z}_j)^2 \right) + \beta \left(\sum_{j \notin \mathcal{J}(\tilde{\mathbf{x}})} (\mathbf{x}_j - \mathbf{z}_j)^2 \right)$$

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

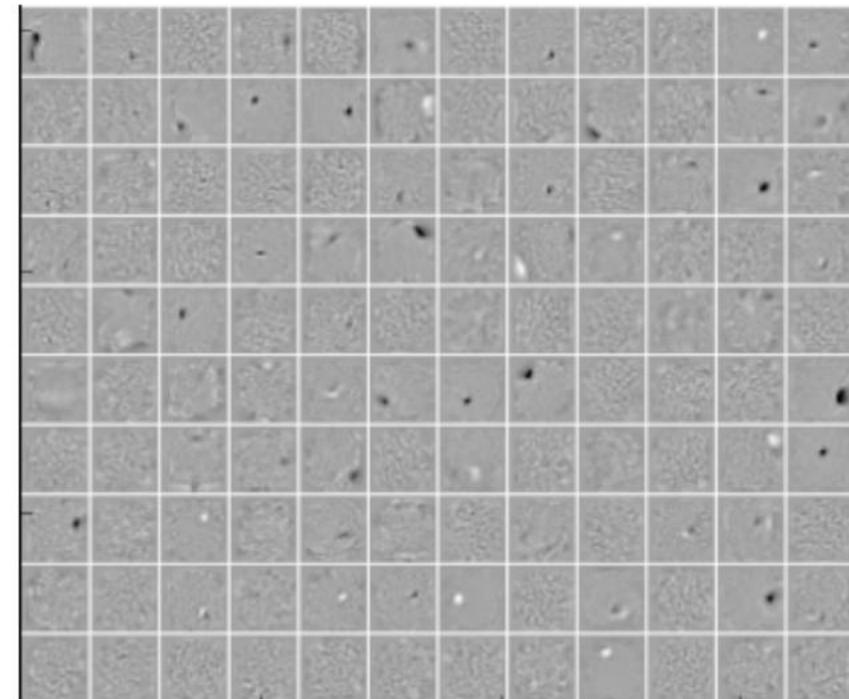
Stacked Denoising Autoencoder



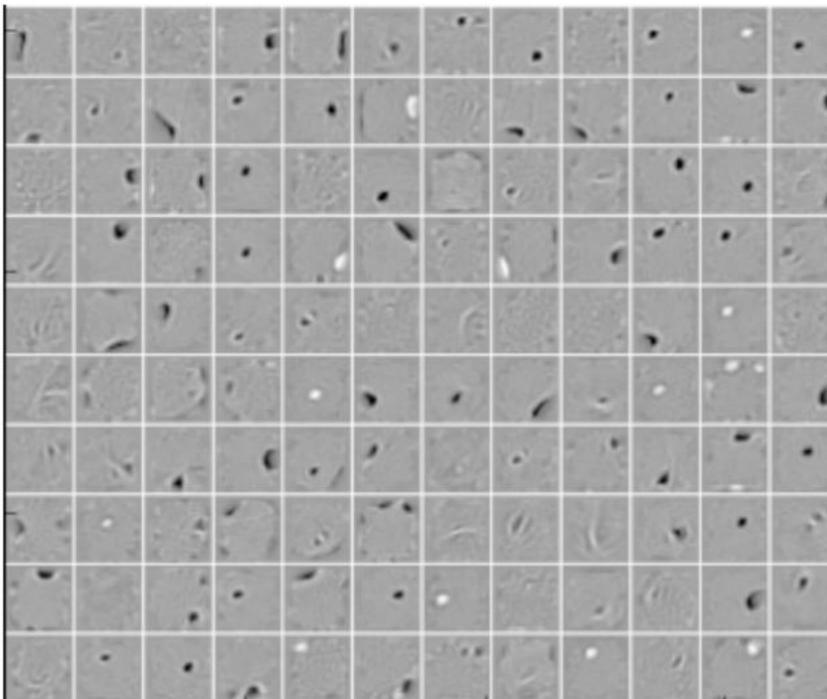
Denoising Autoencoder



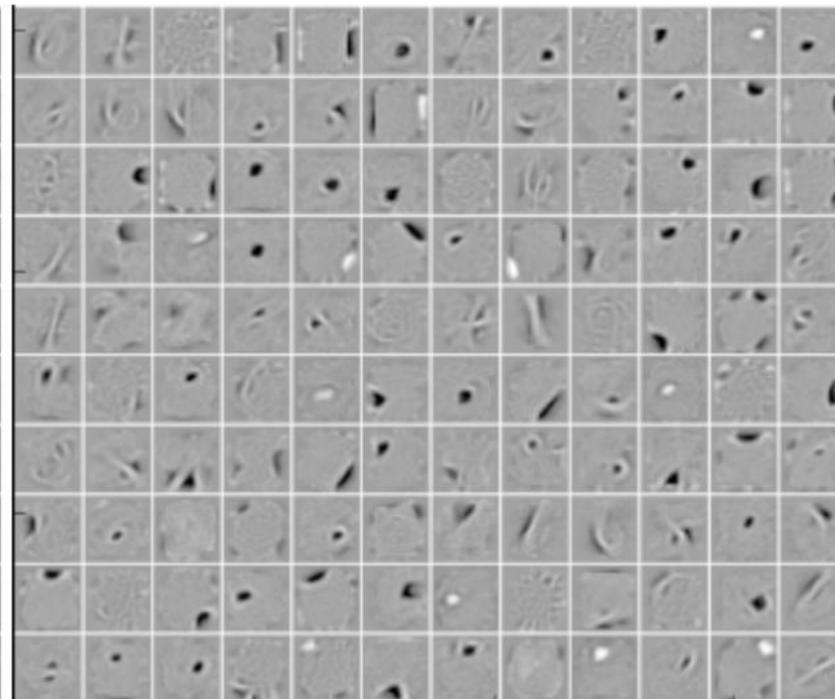
Denoising Autoencoder



(a) No destroyed inputs

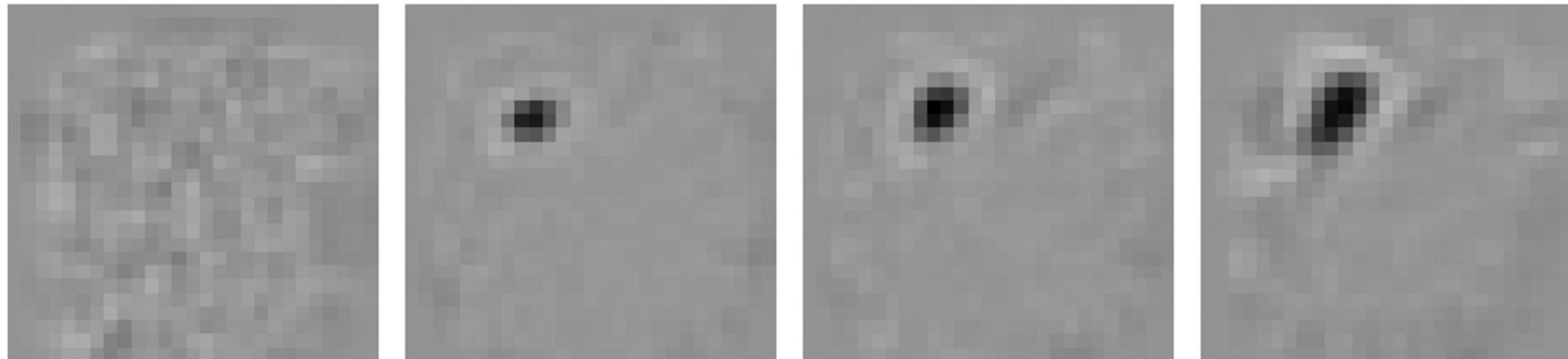


(b) 25% destruction



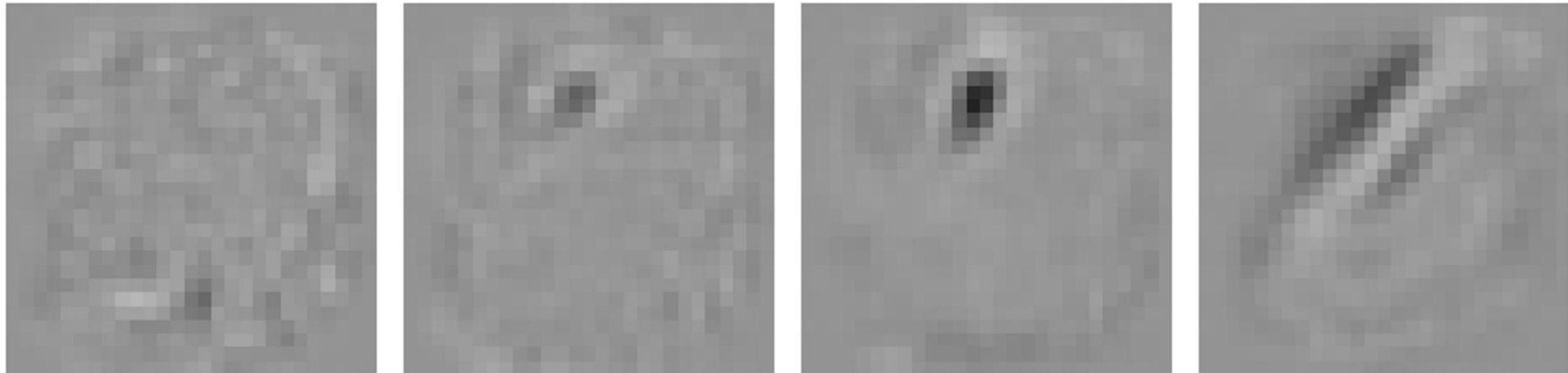
(c) 50% destruction

Denoising Autoencoder



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

Denoising Autoencoder



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

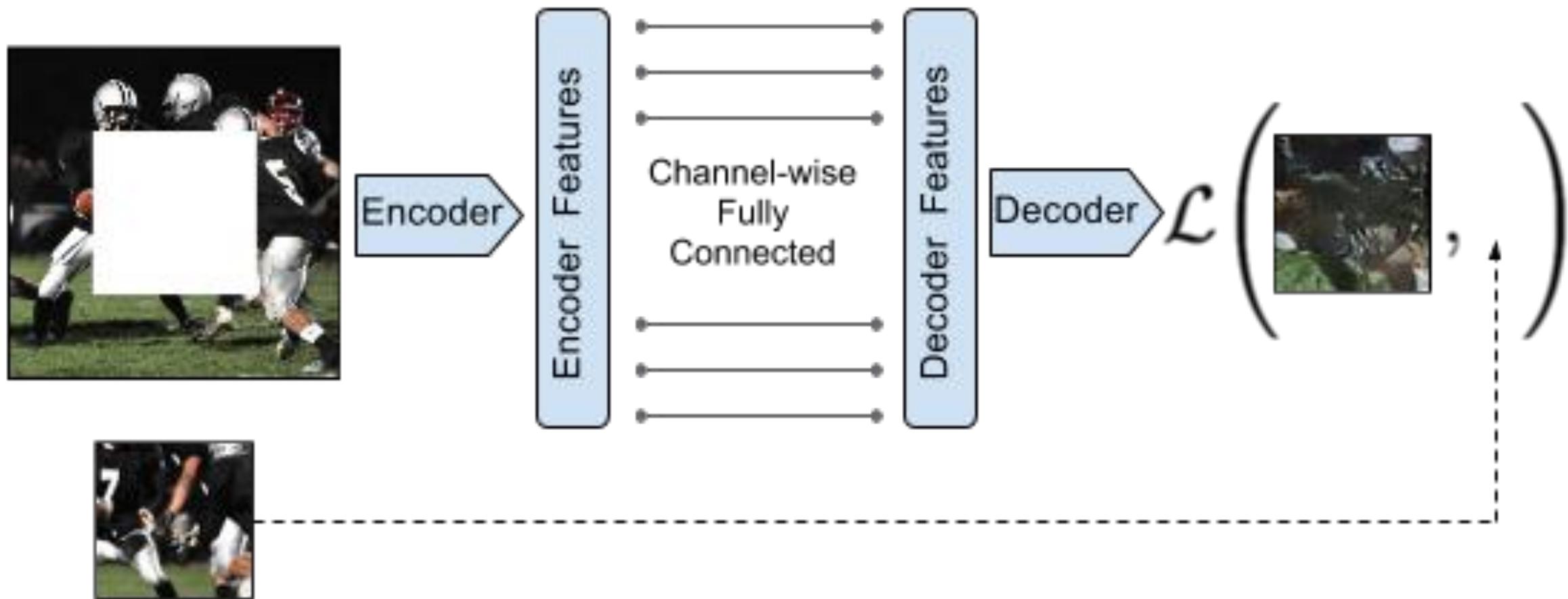
Denoising Autoencoder

Dataset	SVM _{rbf}	SVM _{poly}	DBN-1	SAA-3	DBN-3	SdA-3 (ν)
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%)

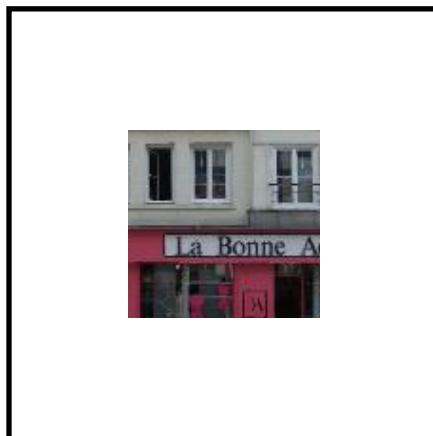
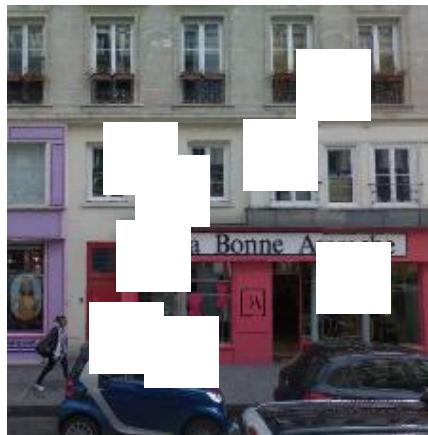
Predict missing pieces



Context Encoders



Context Encoders



(a) Center Region

(b) Random Blocks

(c) Random Shapes

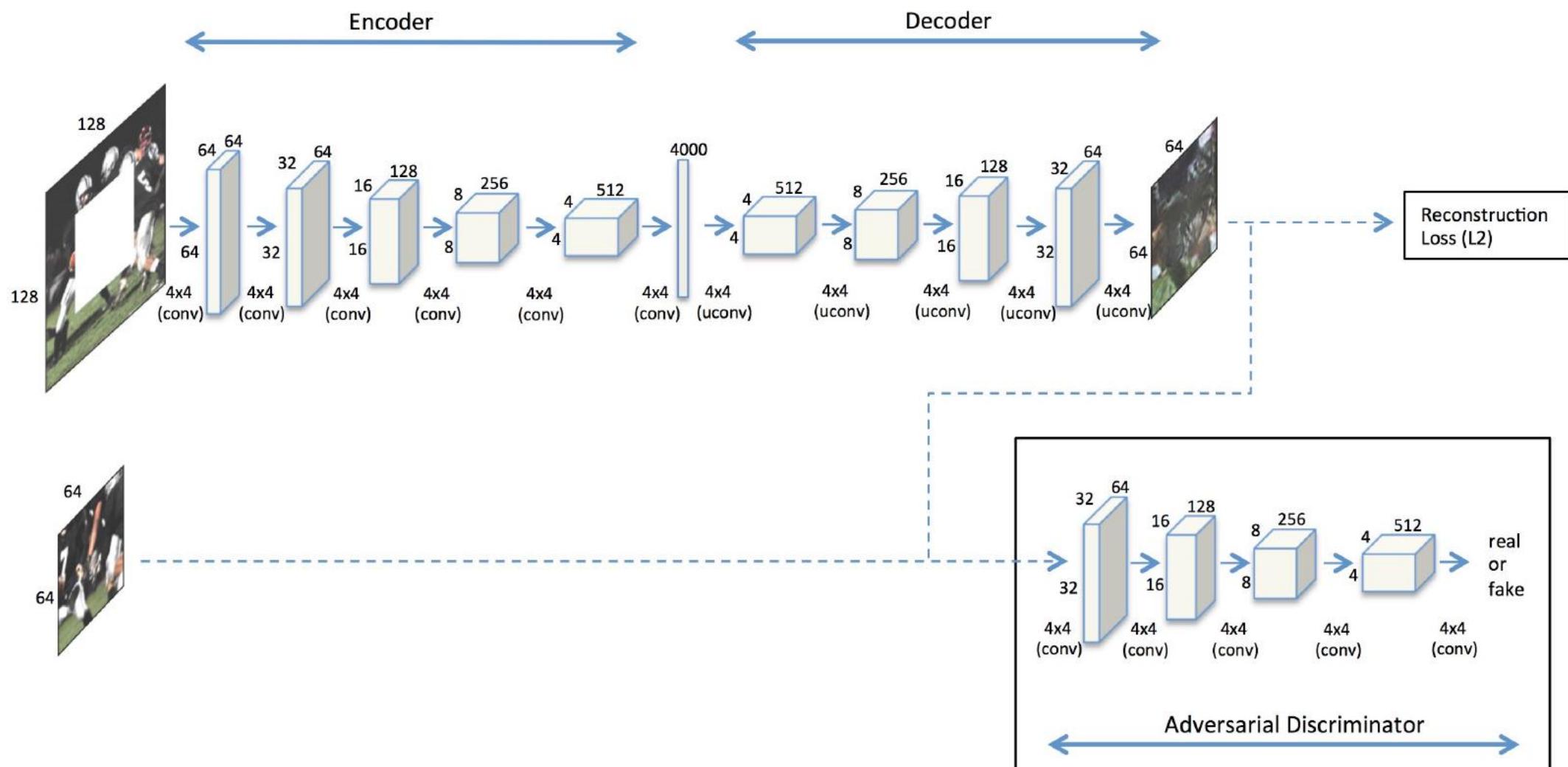
Context Encoders

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

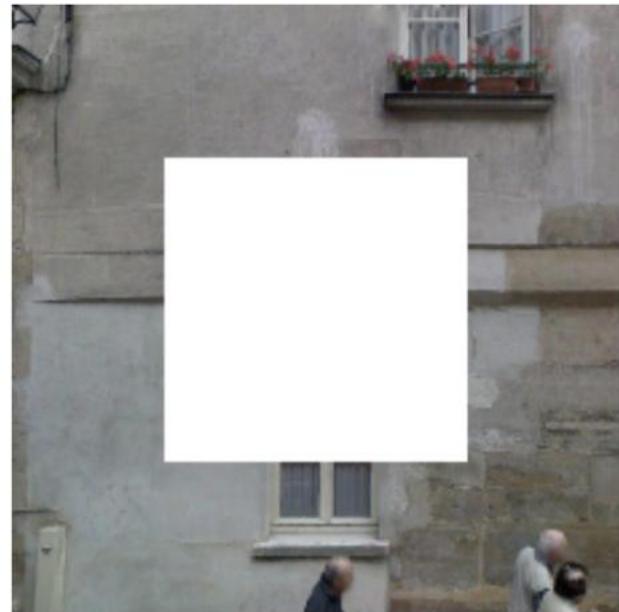
$$\begin{aligned}\mathcal{L}_{adv} = \max_D \mathbb{E}_{x \in \mathcal{X}} & [\log(D(x)) \\ & + \log(1 - D(F((1 - \hat{M}) \odot x)))]\end{aligned}$$

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

Context Encoders



Context Encoders



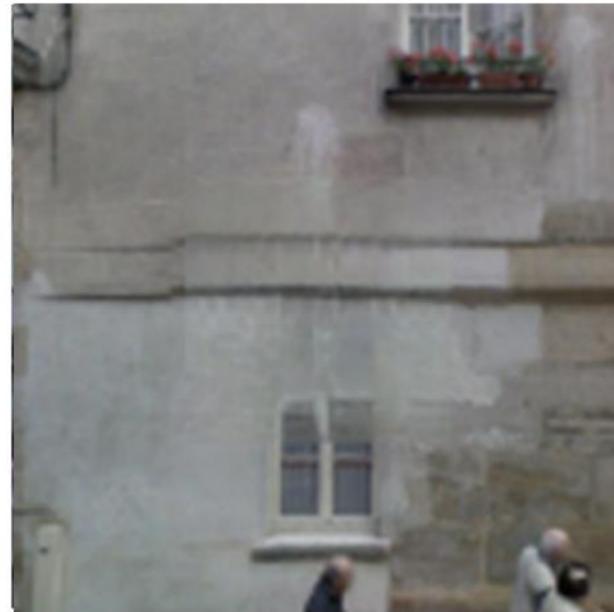
Input Image



L2 Loss



Adversarial Loss



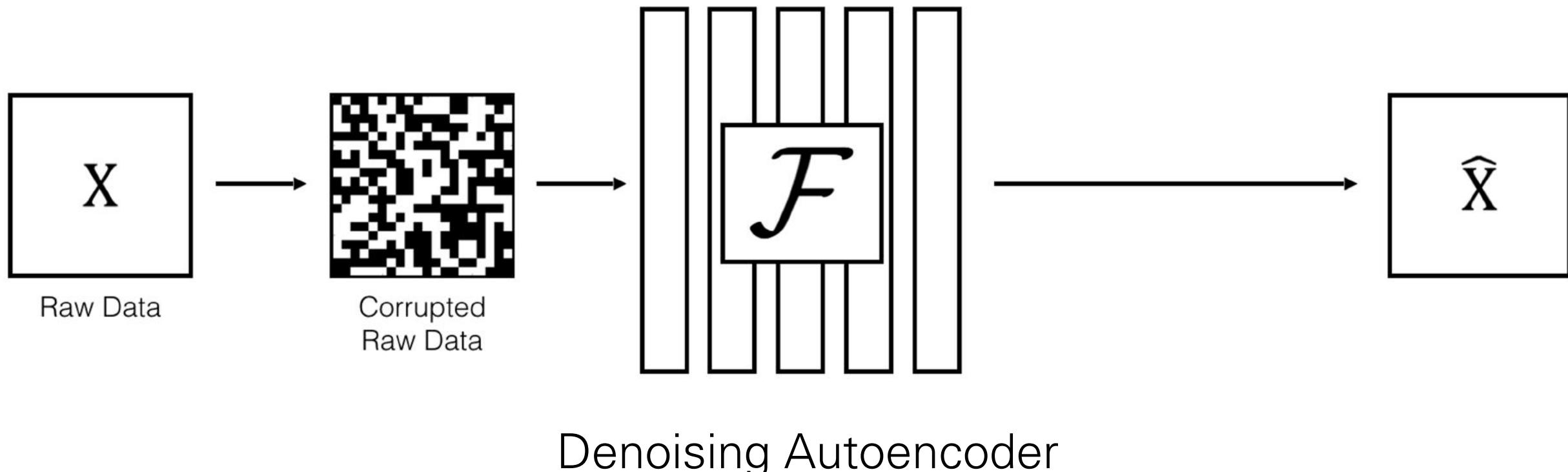
Joint Loss

Context Encoders

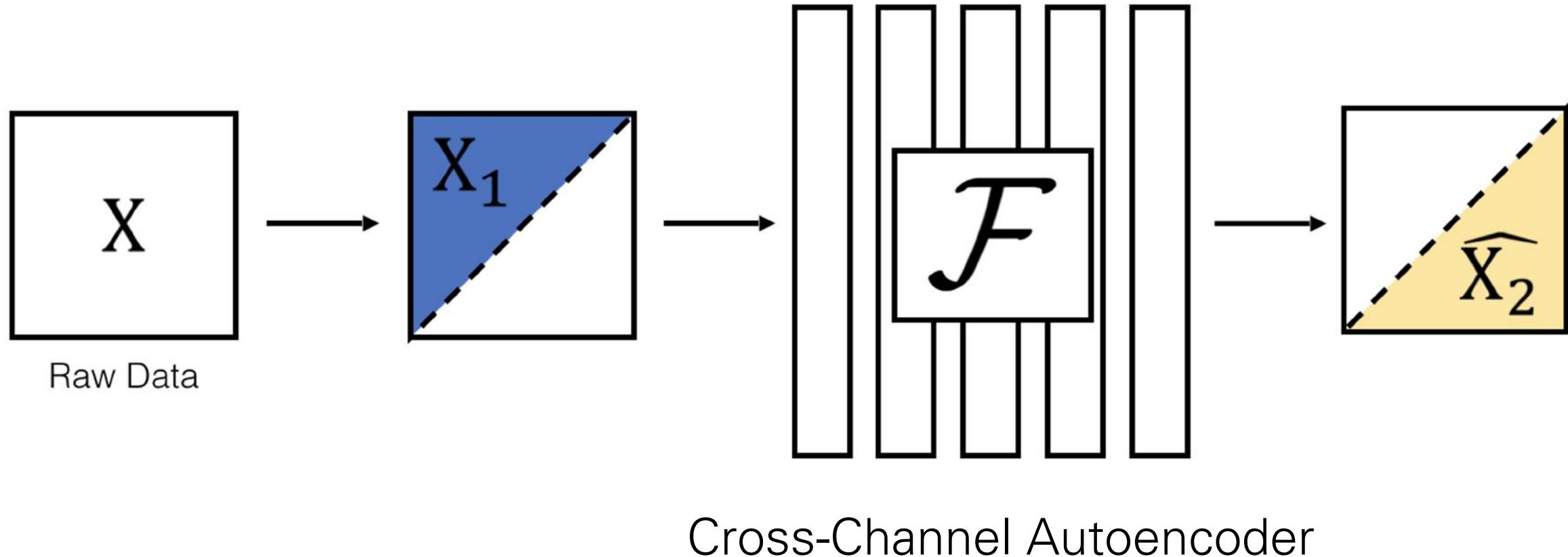
Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	78.2%	56.8%	48.0%
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal <i>et al.</i> [1]	egomotion	10 hours	52.9%	41.8%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

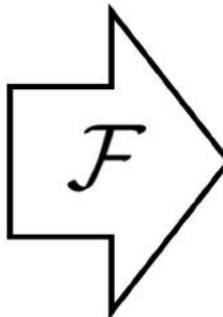
Predicting one view from another



Predicting one view from another

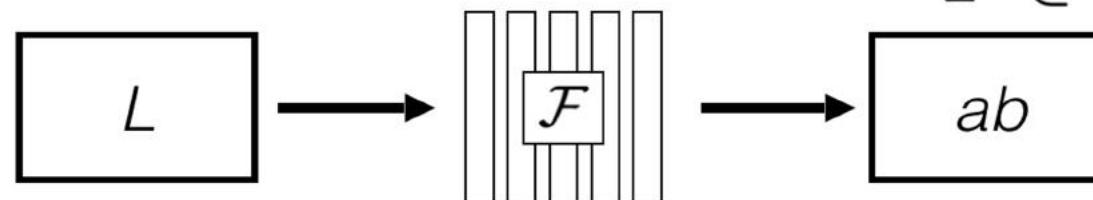


Predicting one view from another



Grayscale image: L channel

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



Color information: ab channels

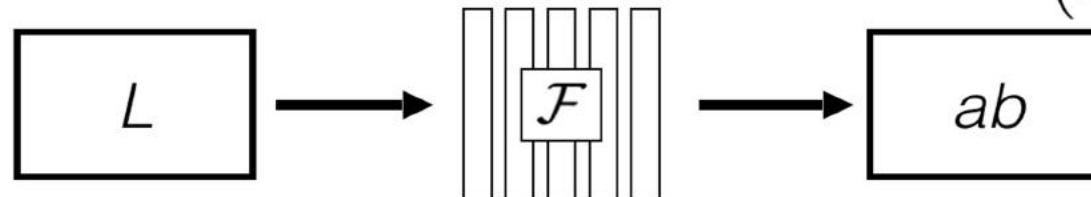
$$\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 2}$$

Predicting one view from another



Grayscale image: L channel

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



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

Predicting one view from another



Ground Truth

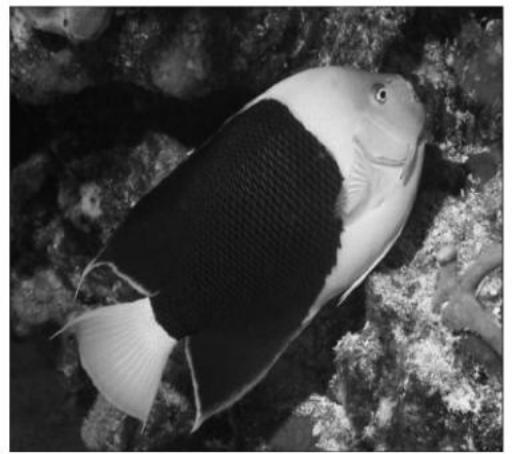


L2 regression



Pixelwise
classification

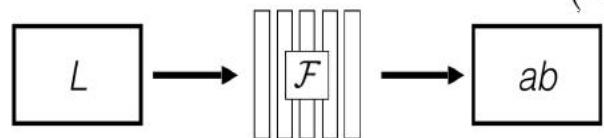
Predicting one view from another



Grayscale image: L channel
 $\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$



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

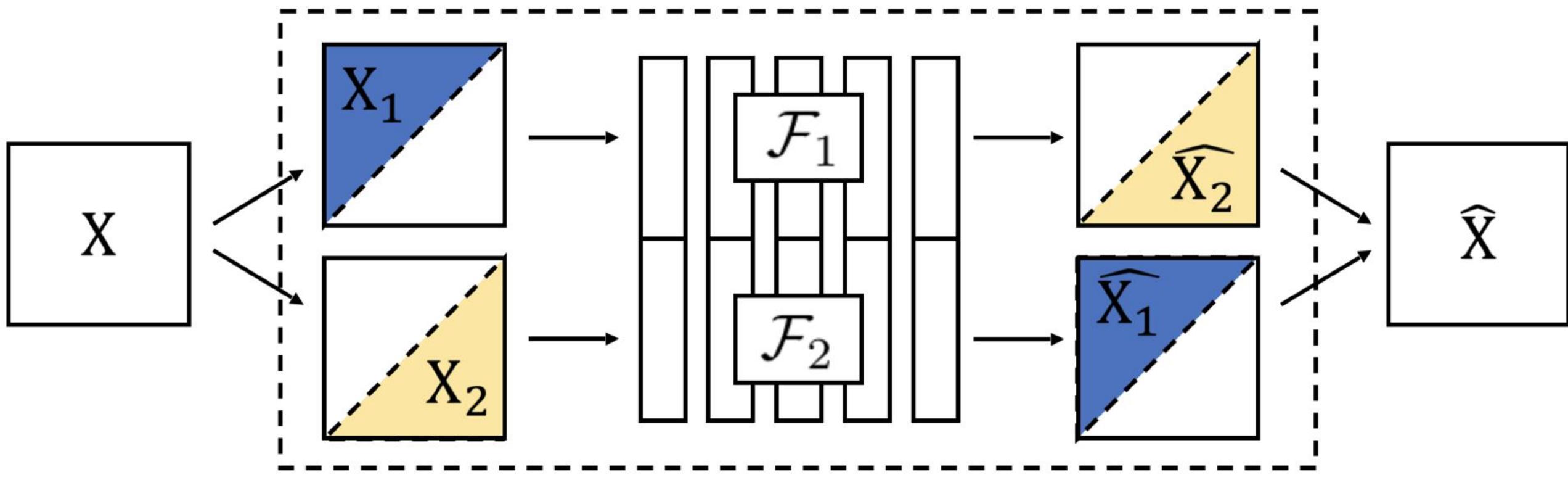


$$L_2(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \|\mathbf{Y}_{h,w} - \hat{\mathbf{Y}}_{h,w}\|_2^2$$

6

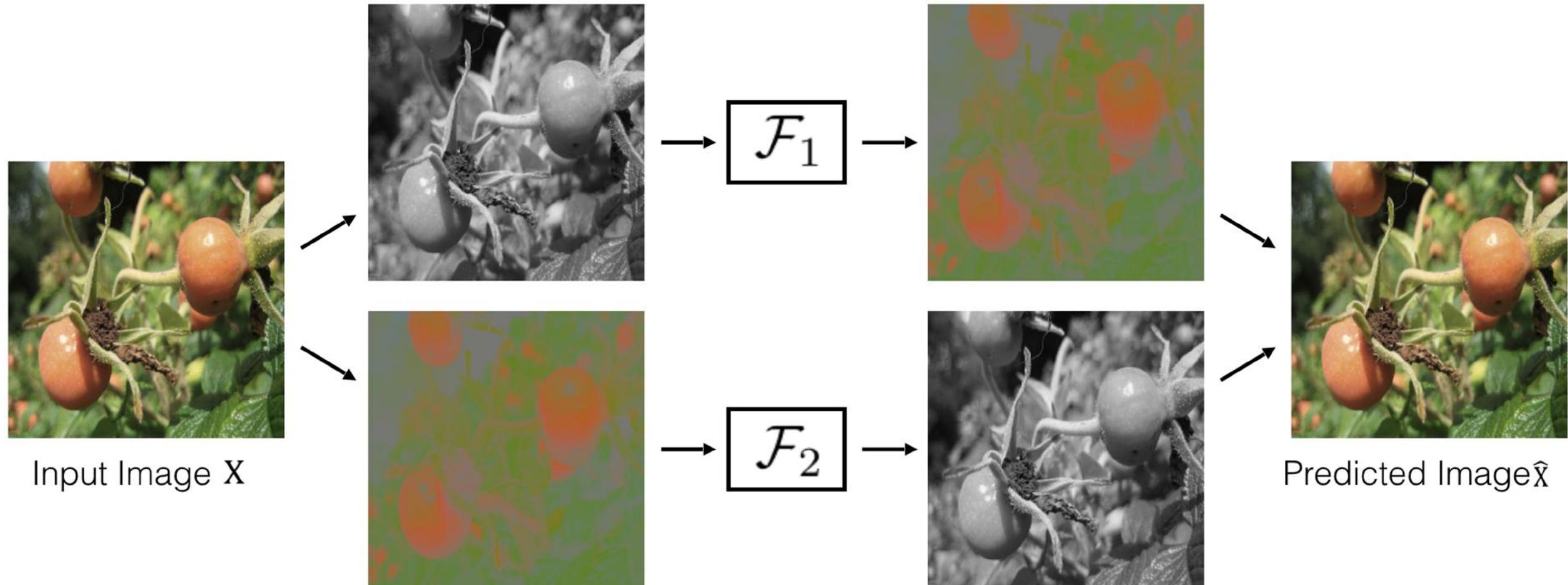
$$L(\hat{\mathbf{Z}}, \mathbf{Z}) = -\frac{1}{HW} \sum_{h,w} \sum_q \mathbf{z}_{h,w,q} \log(\hat{\mathbf{z}}_{h,w,q})$$

Predicting one view from another

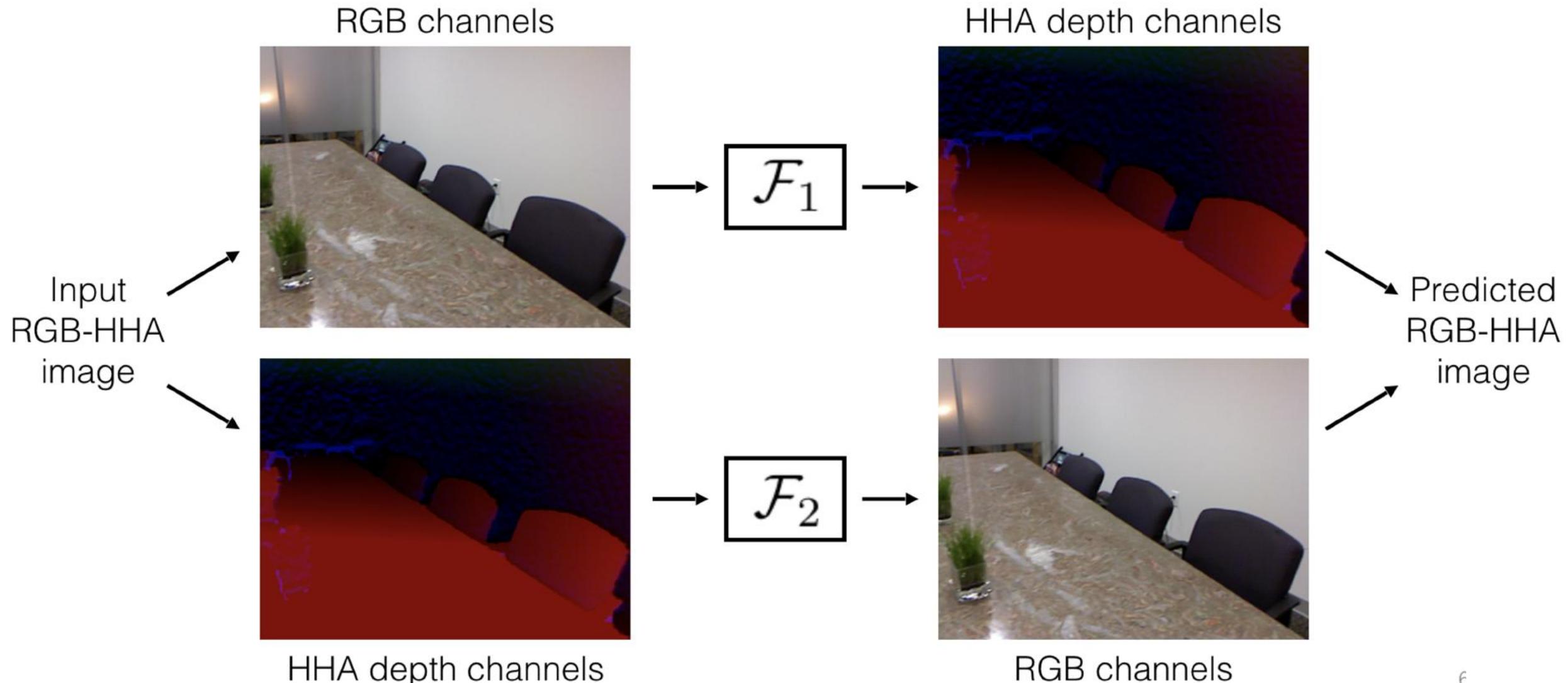


Split-Brain Autoencoder

Predicting one view from another



Predicting one view from another



€

Lecture overview

- Motivation
- Reconstruct from a corrupted (or partial) version
- Visual common-sense tasks
 - Relative patch prediction
 - Jigsaw puzzles
 - Rotation
- Contrastive Learning

Relative Position of Image Patches

Unsupervised Visual Representation Learning by Context Prediction

Carl Doersch^{1,2} Abhinav Gupta¹ Alexei A. Efros²

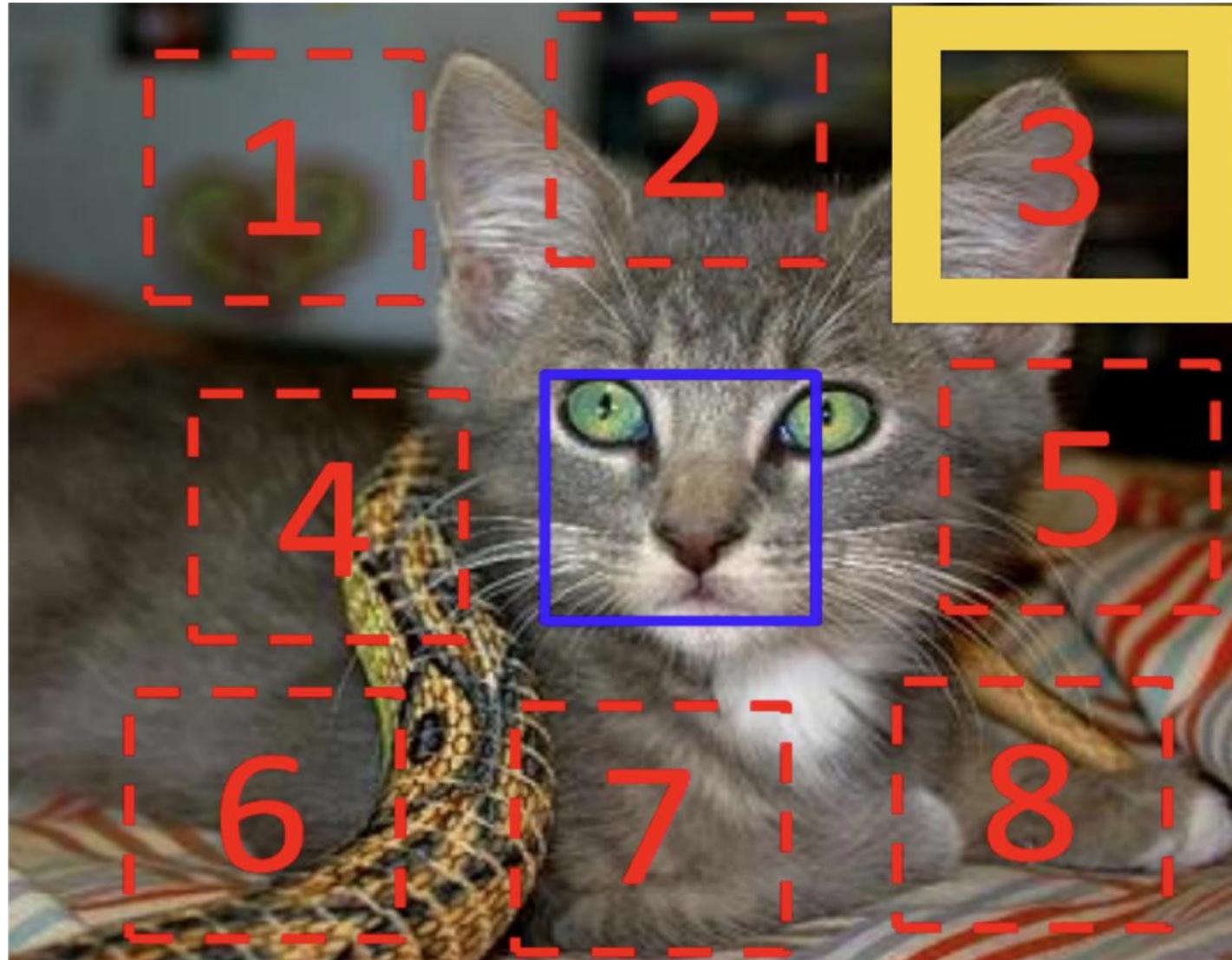
¹ School of Computer Science
Carnegie Mellon University

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



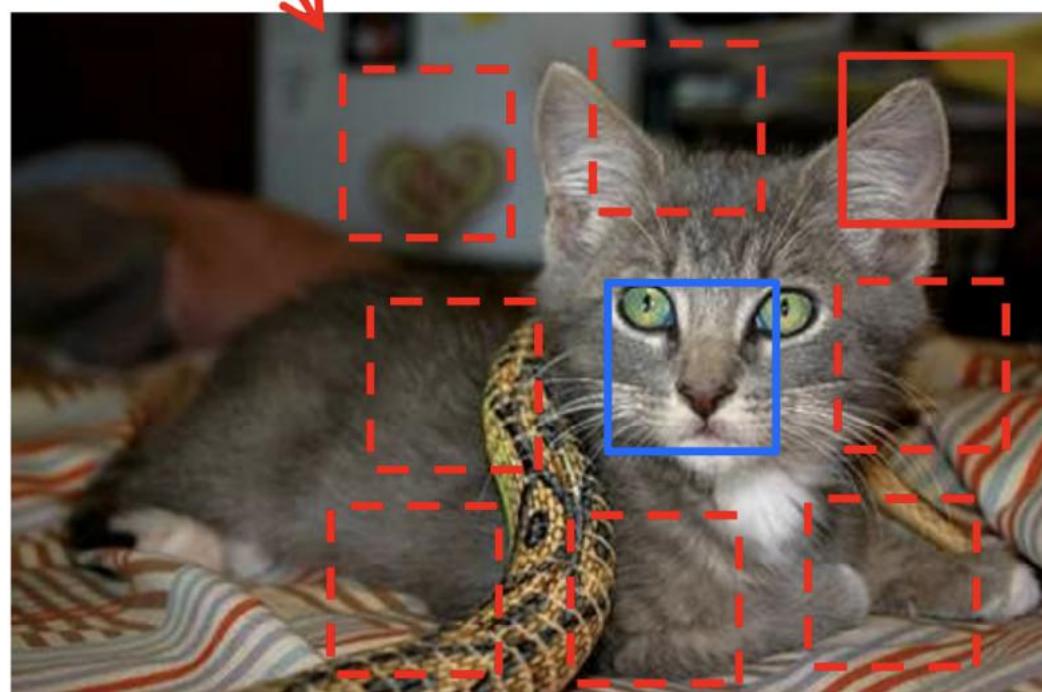
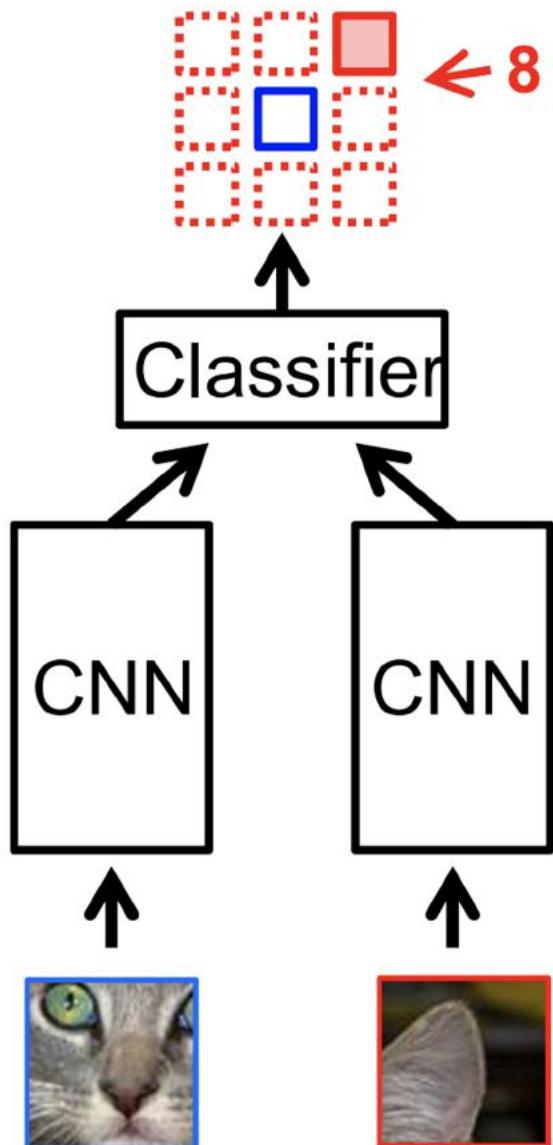
Task: Predict the relative position of the second patch with respect to the first

Relative Position of Image Patches



Doersch,
Gupta, Efros

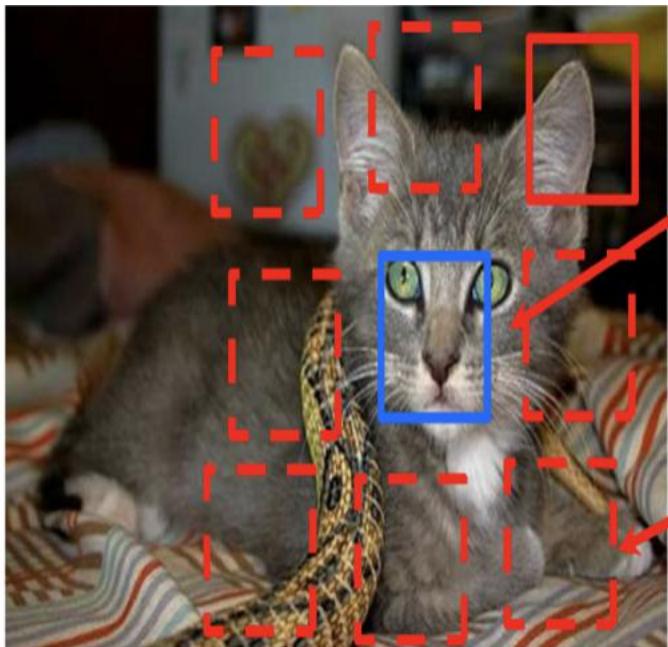
Relative Position of Image Patches



**Randomly Sample Patch
Sample Second Patch**

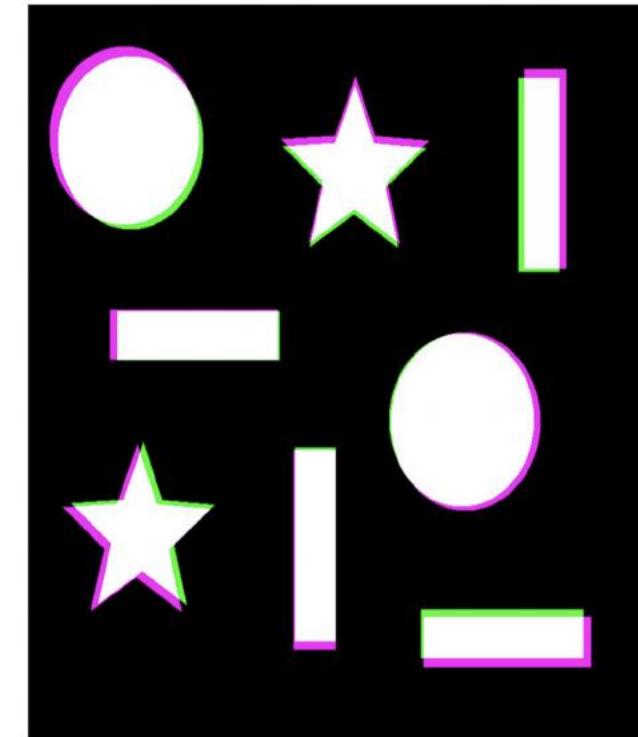
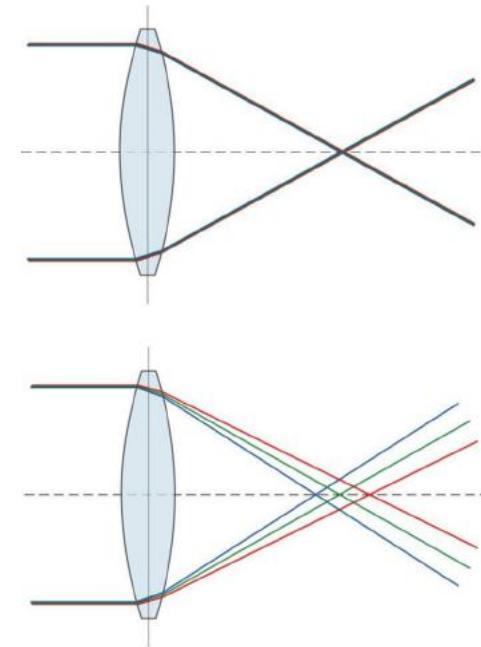
Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

Relative Position of Image Patches

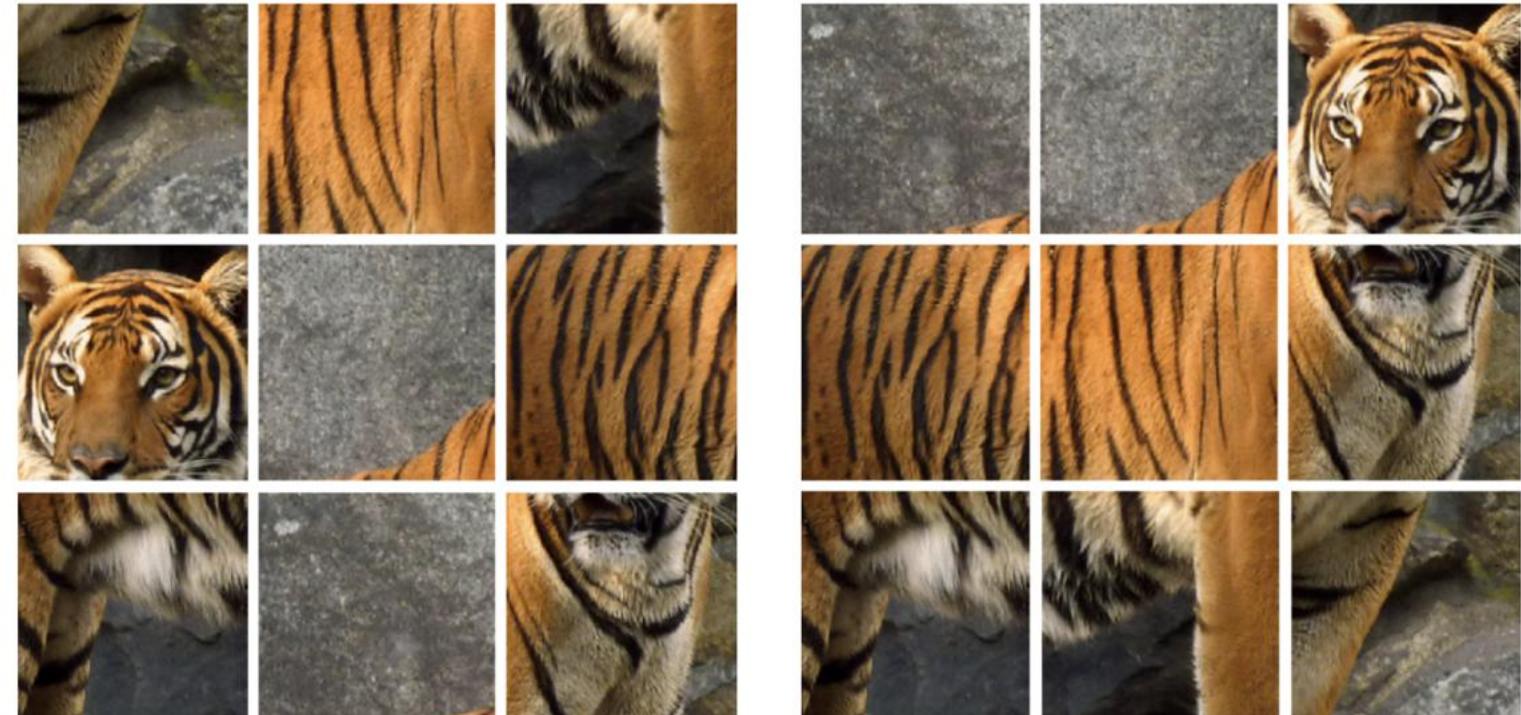
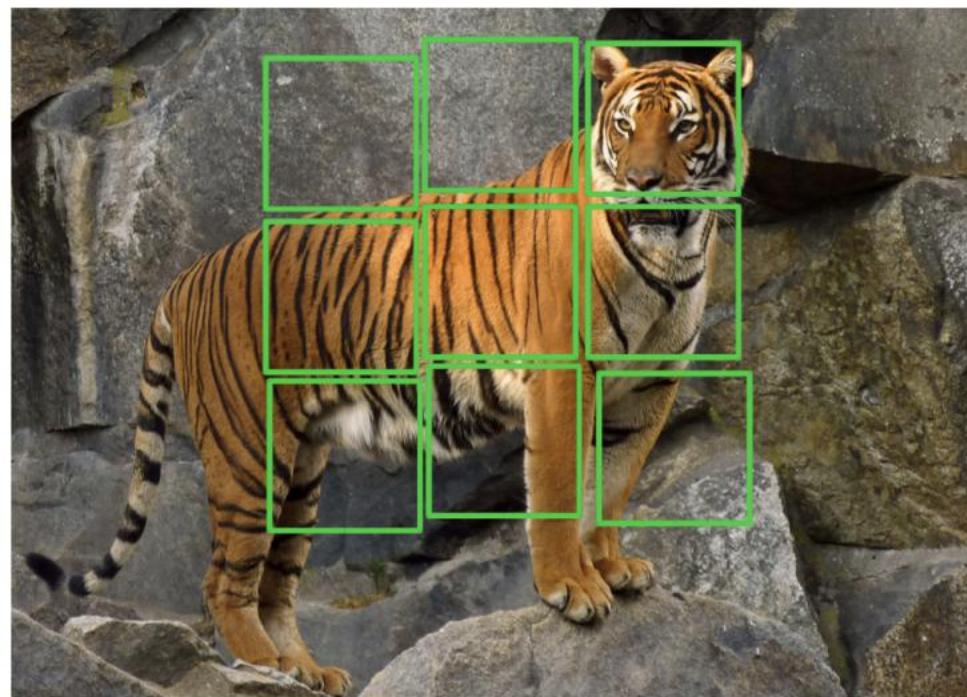


Include a gap

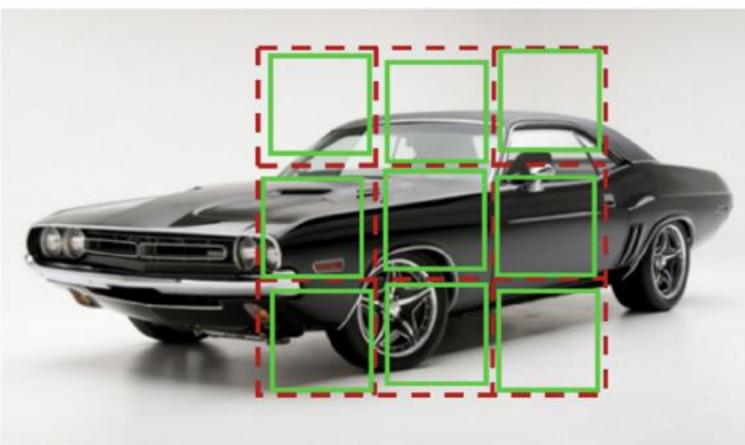
Jitter the patch locations



Solving Jigsaw Puzzles



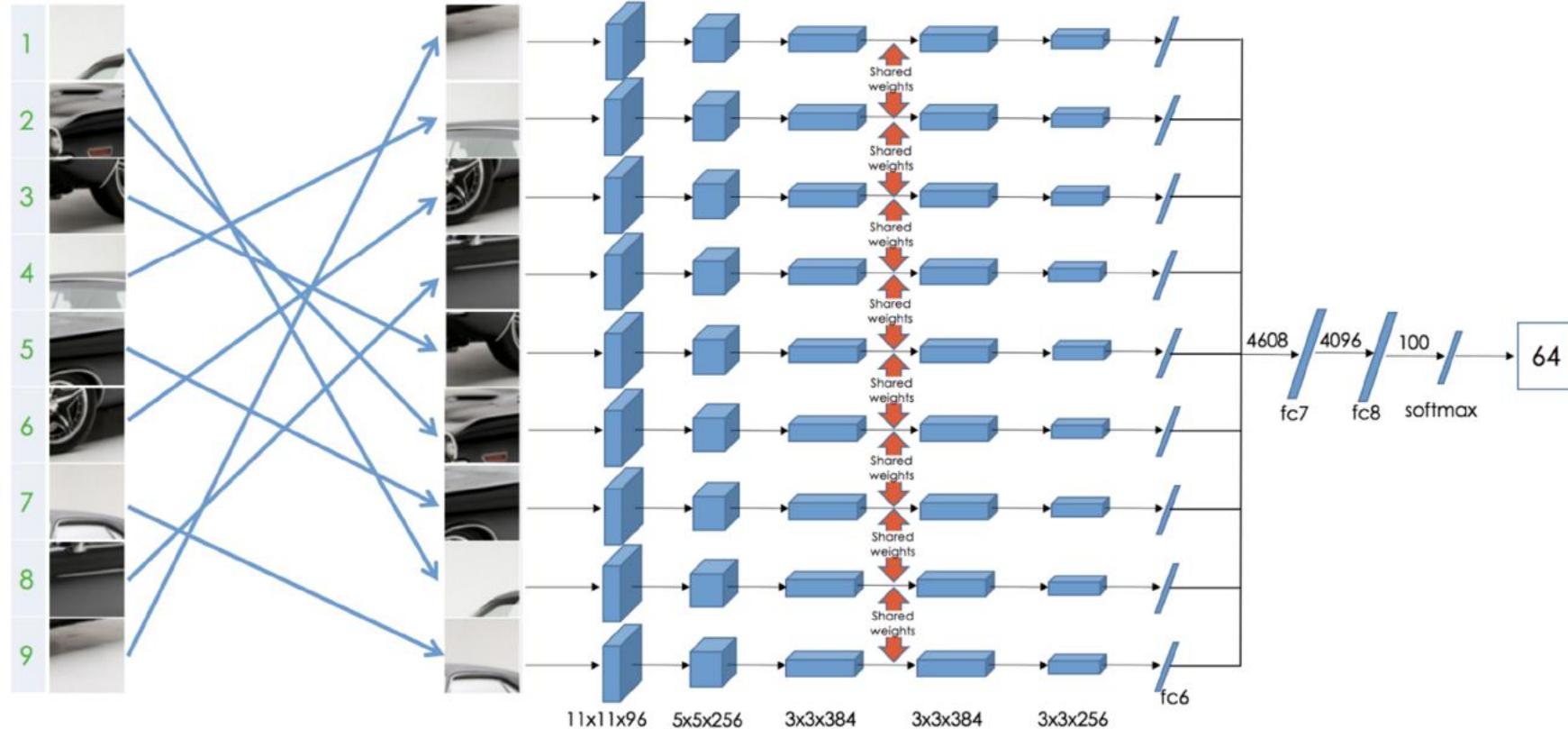
Solving Jigsaw Puzzles



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

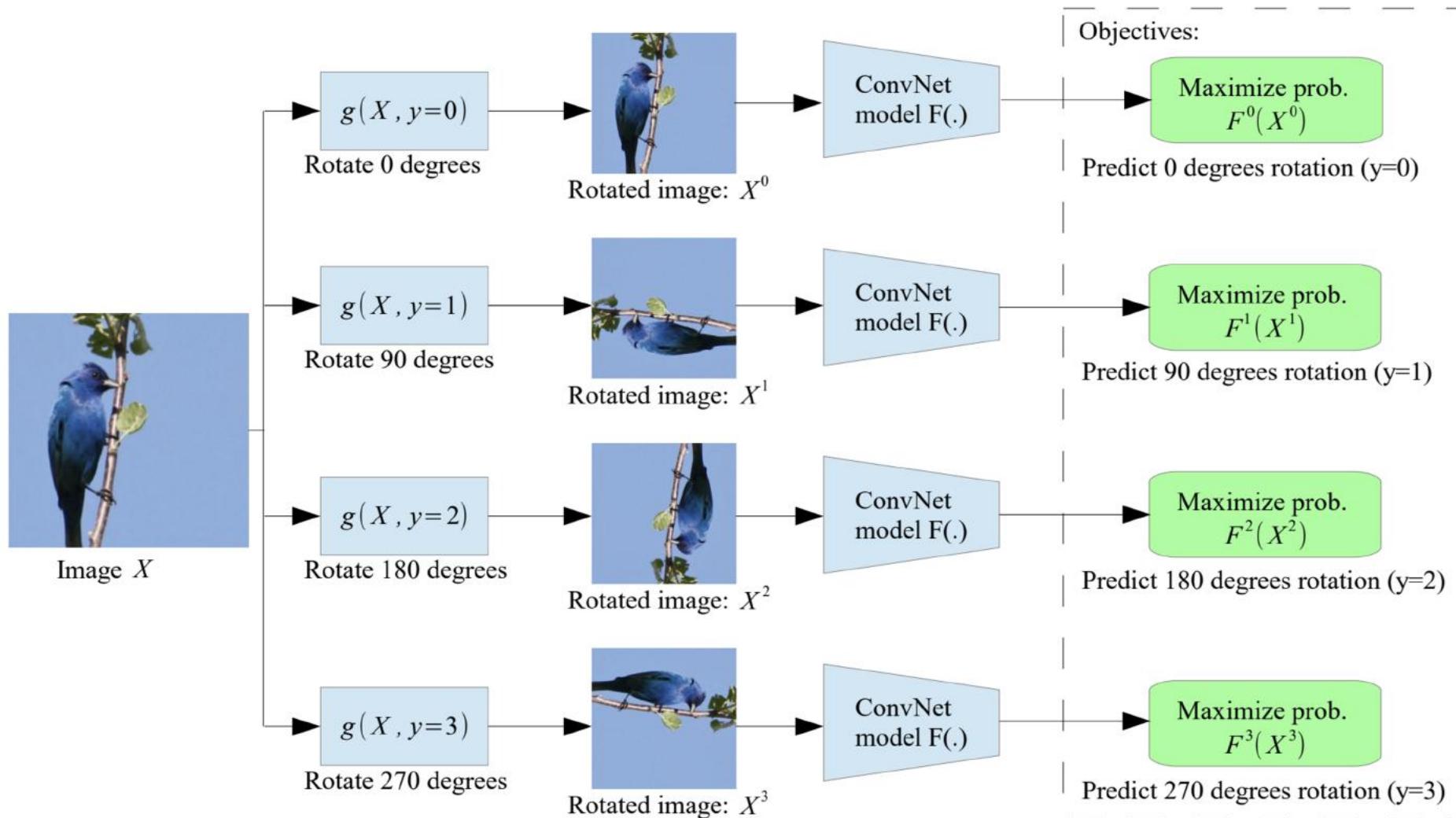


0° rotation



270° rotation

Rotation



Rotation

# Rotations	Rotations	CIFAR-10 Classification Accuracy
4	$0^\circ, 90^\circ, 180^\circ, 270^\circ$	89.06
8	$0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, 225^\circ, 270^\circ, 315^\circ$	88.51
2	$0^\circ, 180^\circ$	87.46
2	$90^\circ, 270^\circ$	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
ImageNet labels	78.9	79.9	56.8
Random		53.3	43.4
Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6
Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9
Context Encoders (Pathak et al., 2016b)	34.6	56.5	44.5
Tracking (Wang & Gupta, 2015)	55.6	63.1	47.4
Context (Doersch et al., 2015)	55.1	65.3	51.1
Colorization (Zhang et al., 2016a)	61.5	65.6	46.9
BIGAN (Donahue et al., 2016)	52.3	60.1	46.9
Jigsaw Puzzles (Noroozi & Favaro, 2016)	-	67.6	53.2
NAT (Bojanowski & Joulin, 2017)	56.7	65.3	49.4
Split-Brain (Zhang et al., 2016b)	63.0	67.1	46.7
ColorProxy (Larsson et al., 2017)		65.9	
Counting (Noroozi et al., 2017)	-	67.7	51.4
(Ours) RotNet	70.87	72.97	54.4
			39.1

Temporal coherence of color

Task: given a color video ...

Colorize all frames of a gray scale version using a reference frame



Reference Frame

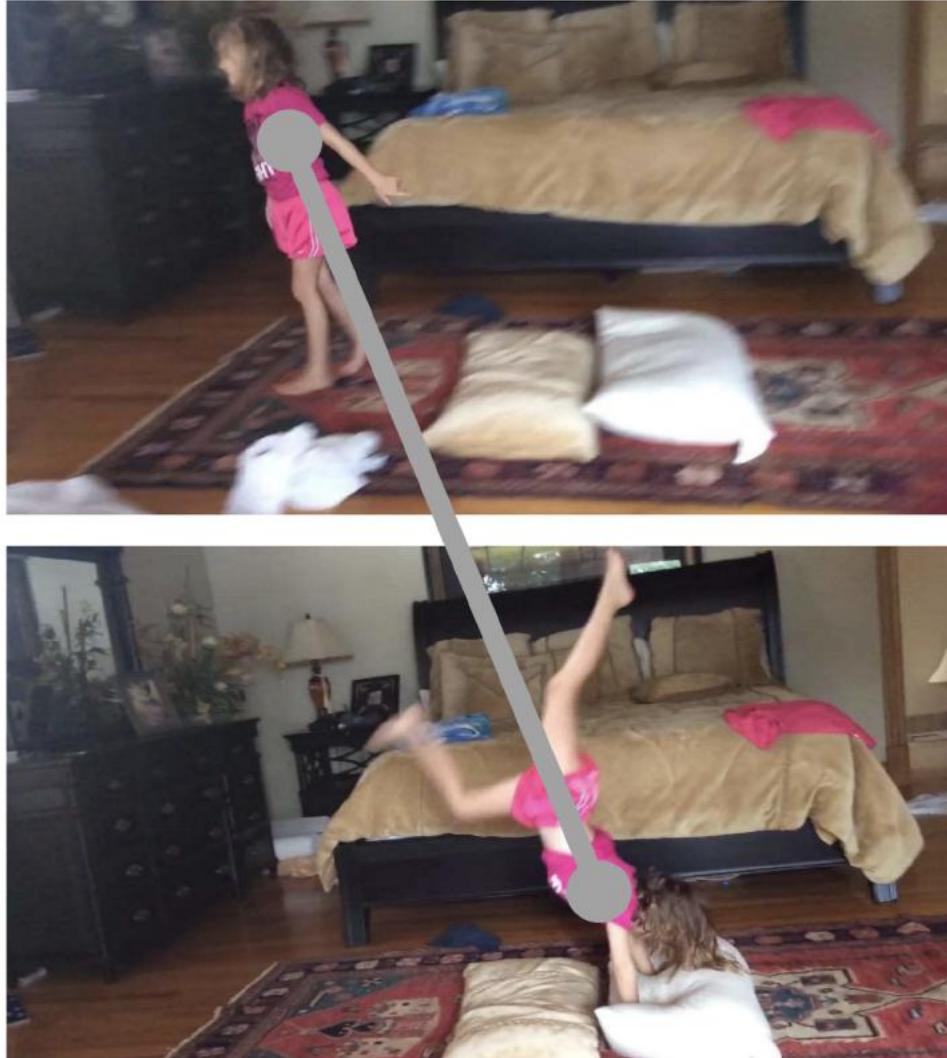


Gray-scale Video

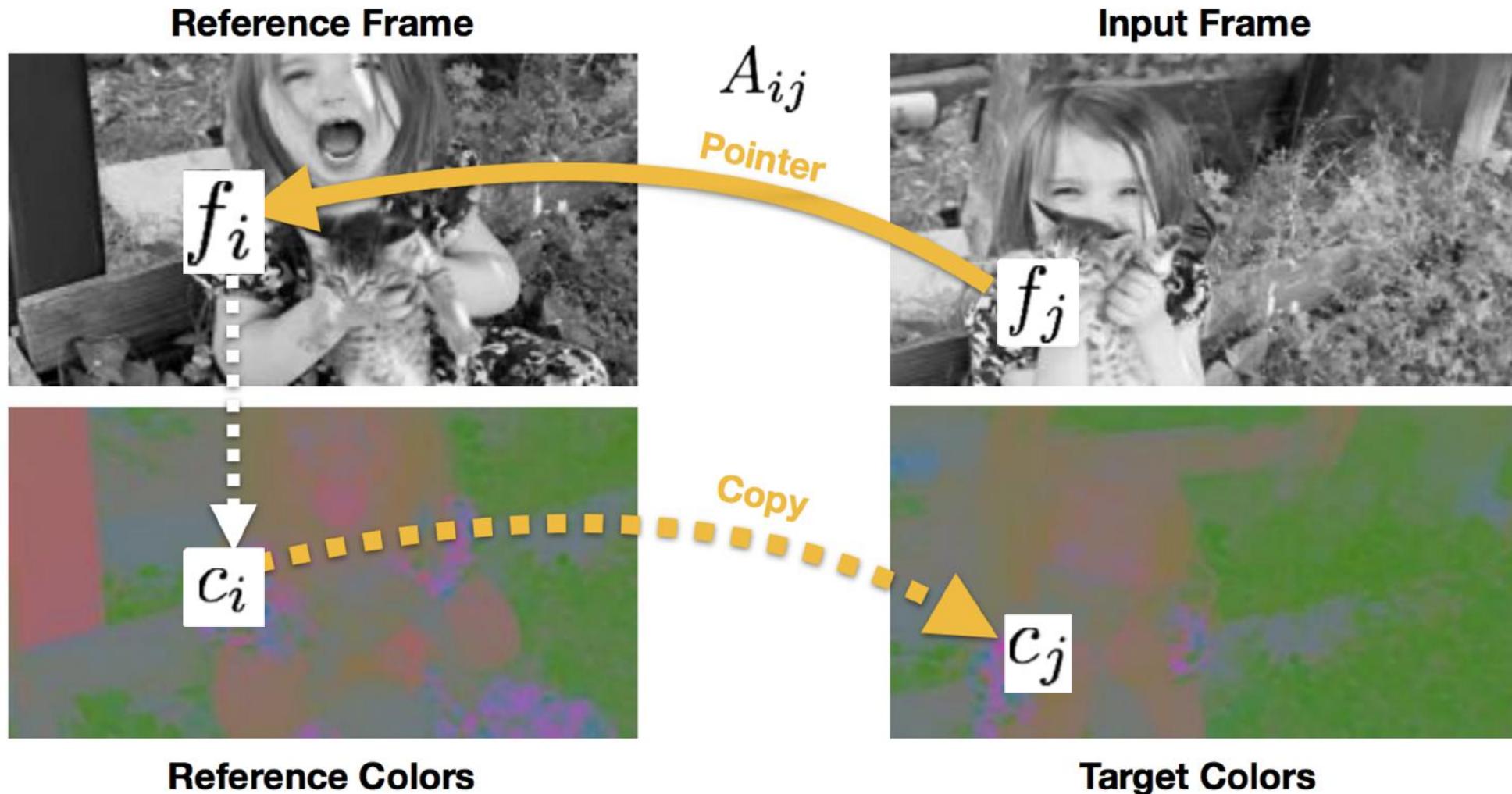
Temporal coherence of color



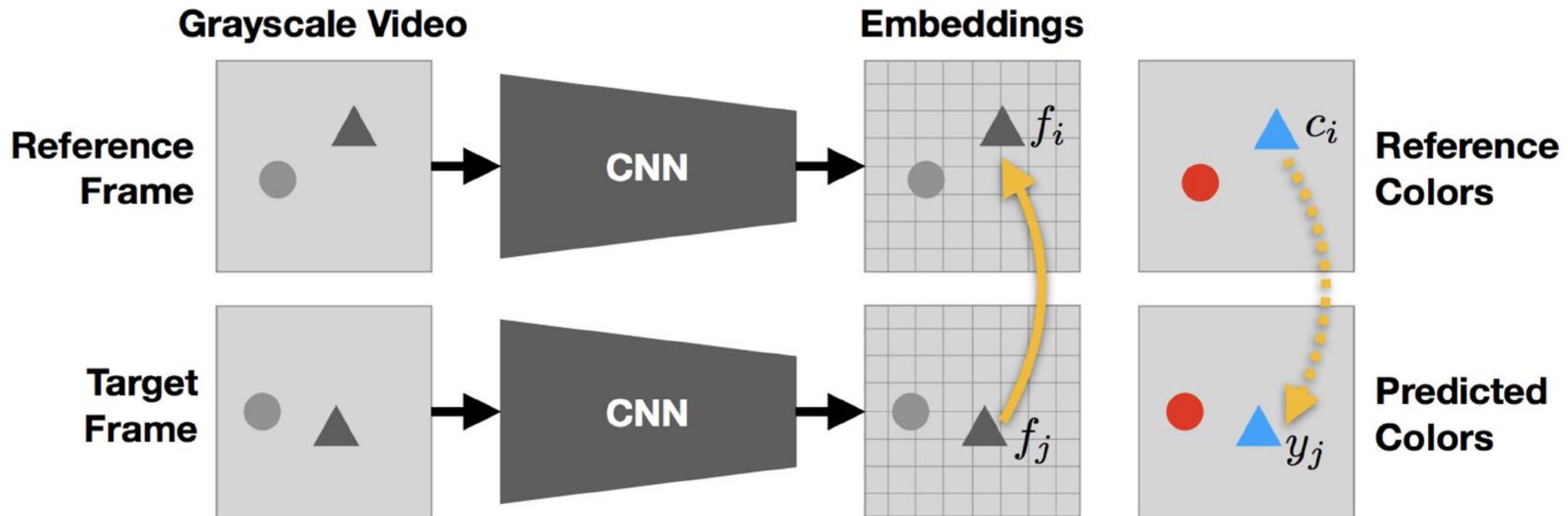
Temporal coherence of color



Tracking emerges from colorization



Tracking emerges from colorization

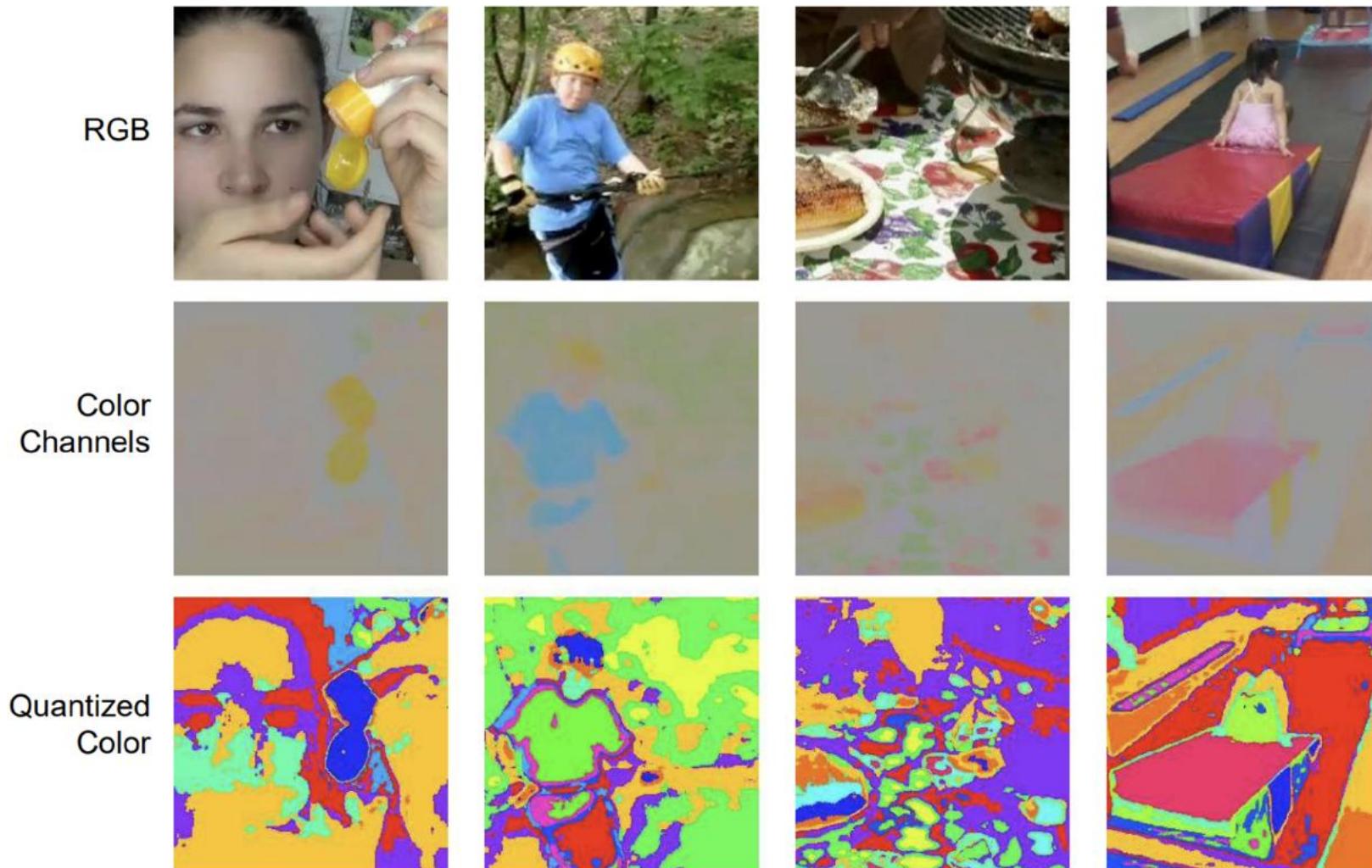


$$A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)}$$

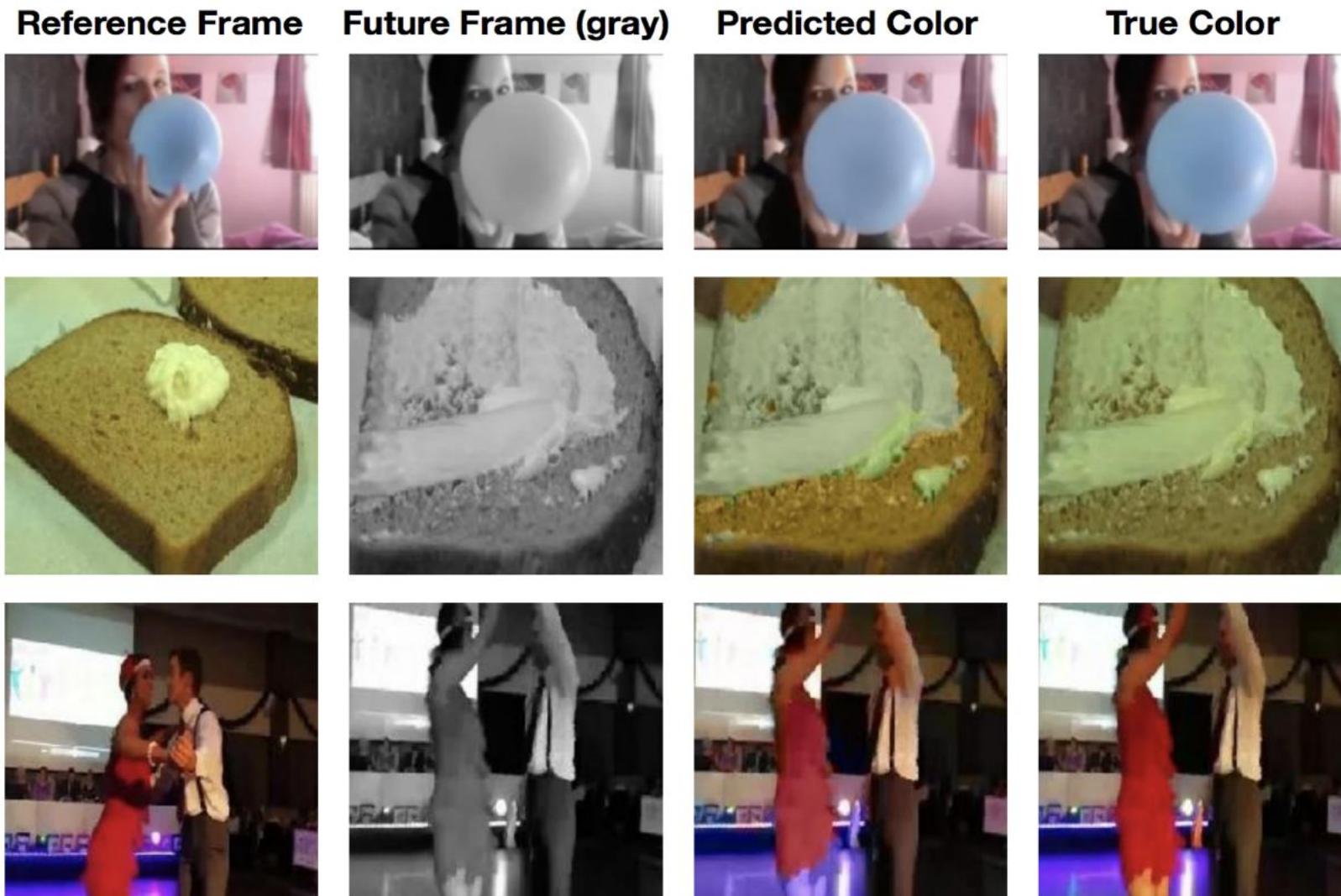
$$\hat{c}_j = \sum_i A_{ij} c_i$$

$$\min_f \mathcal{L} \left(c_j, \sum_i A_{ij} c_i \right)$$

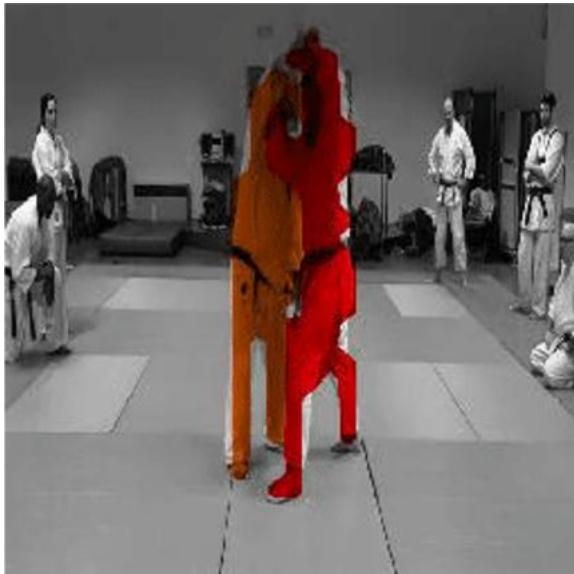
Tracking emerges from colorization



Tracking emerges from colorization



Tracking emerges from colorization

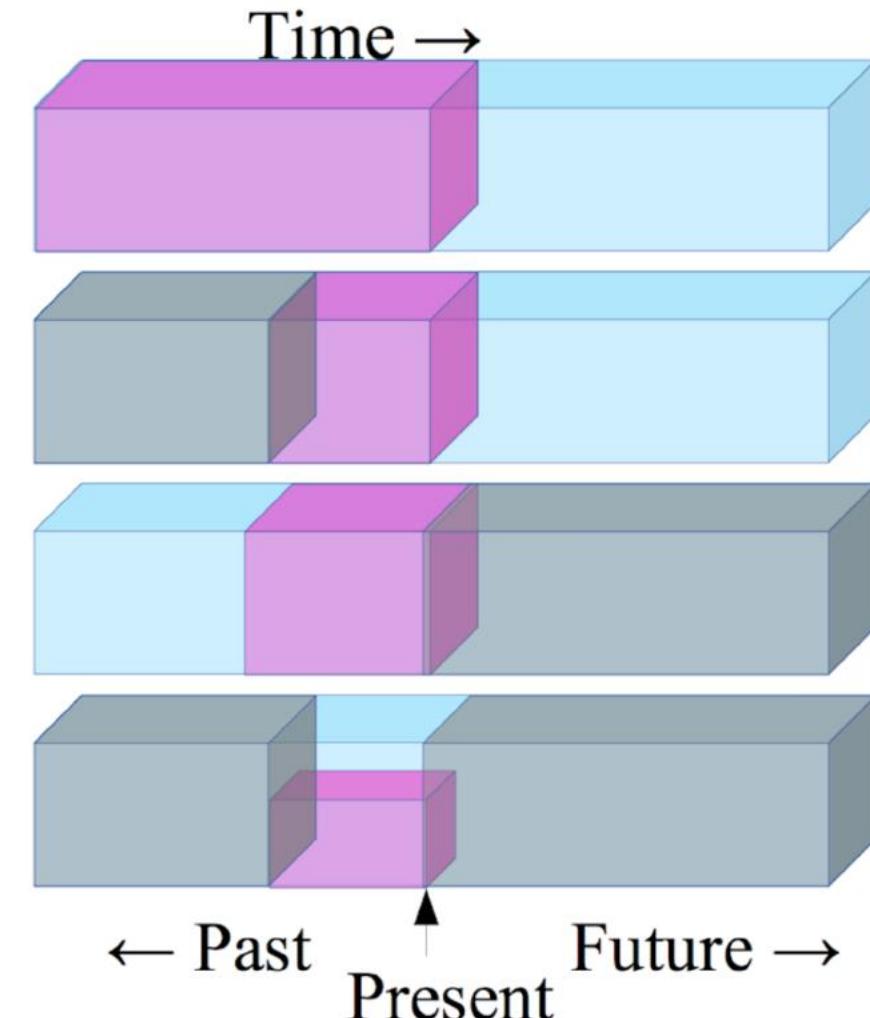


Lecture overview

- Motivation
- Reconstruct from a corrupted (or partial) version
- Visual common-sense tasks
- Contrastive Learning
 - Word2vec
 - Contrastive Predictive Coding (CPC)
 - Instance Discrimination
 - Recent State-of-the-art progress

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.**

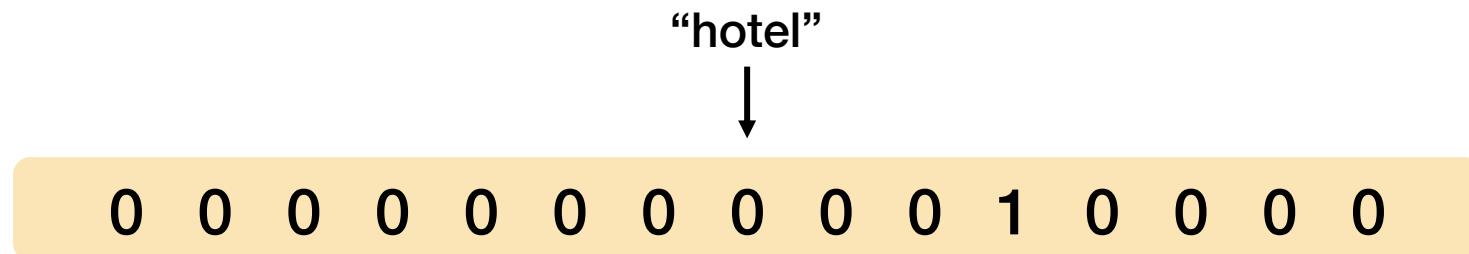


Word Embeddings

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

Word embeddings

- The vast majority of rule-based or statistical NLP and IR work regarded words as atomic symbols: **hotel, conference, walk**
- In vector space terms, this is a vector with one 1 and a lot of zeroes



- We now call this a **one-hot** representation.

Word embeddings

- The size of word vectors are equal to the number of words in the dictionary
 - Vector size is proportional to the size of the dictionary
20K (speech) – 50K (Penn Treebank) – 500K (A large dictionary) – 13M (Google 1T)
- One-hot vectors are **orthogonal**
- There is no natural notion of similarity in a set of one-hot vectors

$$\begin{matrix} \text{"motel"} & \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}^T \\ \text{"hotel"} & \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{matrix} = 0 \end{matrix}$$

Word embeddings

- You can get a lot of value by representing a word by means of its neighbors
- “You shall know a word by the company it keeps”
(J. R. Firth 1957:11)



- One of the most successful ideas of modern NLP

government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge



These words will represent
“banking”



Distributional hypothesis

- The meaning of a word is (can be approximated by, derived from) the set of contexts in which it occurs in texts

He filled the **wampimuk**, passed it around and we all drunk some

We found a little, hairy **wampimuk** sleeping behind the tree

Distributional semantics

he curtains open and the moon shining in on the barely
ars and the cold , close moon " . And neither of the w
rough the night with the moon shining so brightly , it
made in the light of the moon . It all boils down , wr
surely under a crescent moon , thrilled by ice-white
sun , the seasons of the moon ? Home , alone , Jay pla
m is dazzling snow , the moon has risen full and cold
un and the temple of the moon , driving out of the hug
in the dark and now the moon rises , full and amber a
bird on the shape of the moon over the trees in front
But I could n't see the moon or the stars , only the
rning , with a sliver of moon hanging among the stars
they love the sun , the moon and the stars . None of
the light of an enormous moon . The plash of flowing w
man 's first step on the moon ; various exhibits , aer
the inevitable piece of moon rock . Housing The Airsh
oud obscured part of the moon . The Allied guns behind

A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge [Landauer and Dumais'97]
From frequency to meaning: Vector space models of semantics [Turney ve Pantel'10]

Word Embeddings

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

The resulting counts matrix will then be:

$$X = \begin{matrix} & \begin{matrix} I & like & enjoy & deep & learning & NLP & flying & . \end{matrix} \\ \begin{matrix} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{matrix} & \left[\begin{matrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{matrix} \right] \end{matrix}$$

Word Embeddings

Applying SVD to X :

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

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

n-gram Language Models

Unigram

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

Bigram

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

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

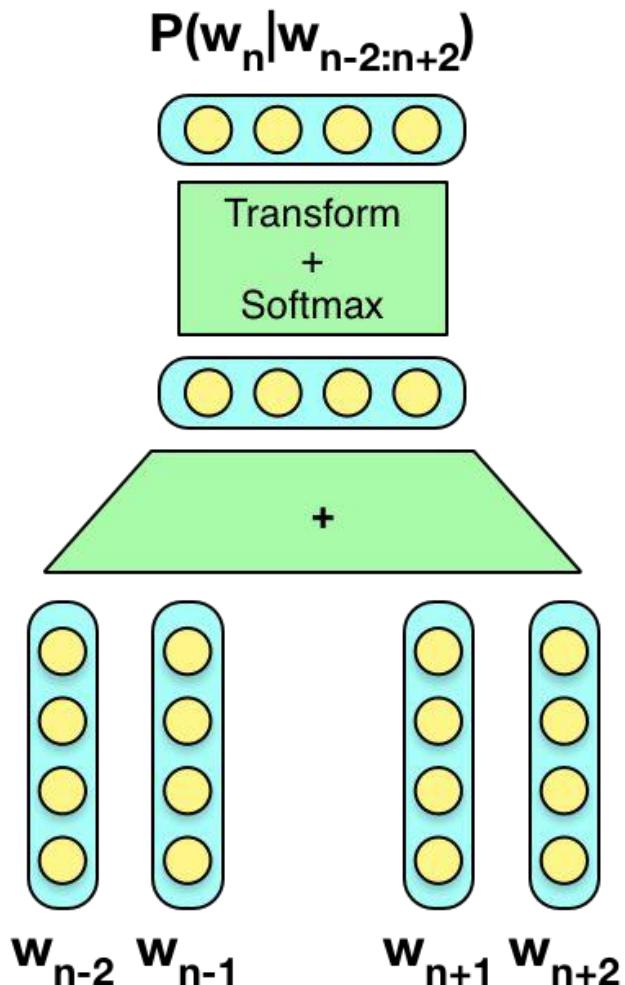
Google Inc., Mountain View, CA

jeff@google.com

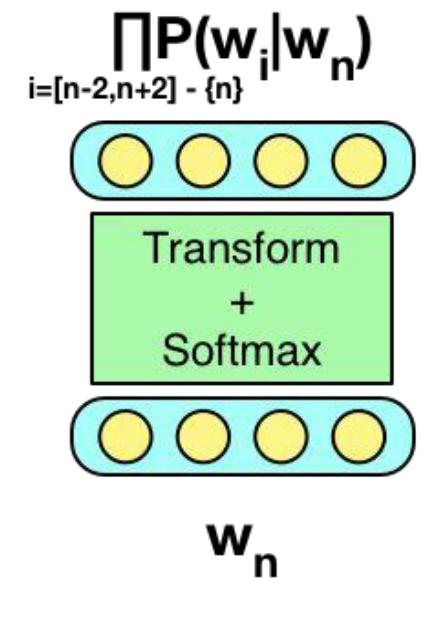
Abstract

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



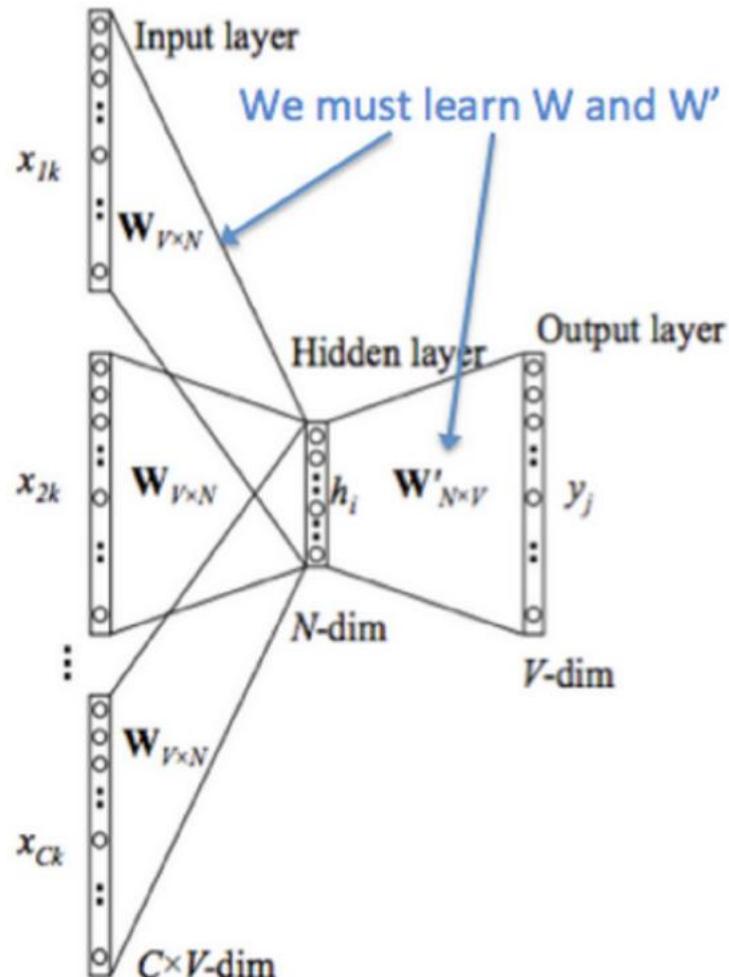
CBoW model



Skip-gram model

word2vec - CBOW

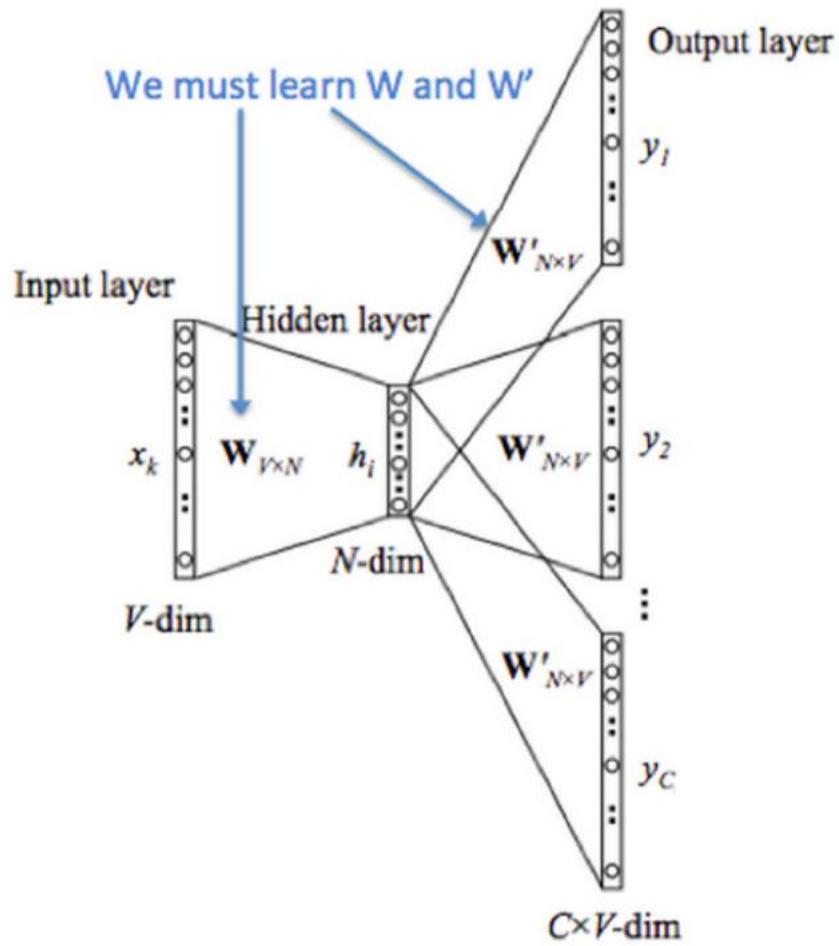
Continuous Bag Of Words (CBOW)



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

word2vec - Skip Gram

Skip Gram



$$\text{minimize } J = -\log P(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m} | w_c)$$

$$= -\log \prod_{j=0, j \neq m}^{2m} P(w_{c-m+j} | w_c)$$

$$= -\log \prod_{j=0, j \neq m}^{2m} P(u_{c-m+j} | v_c)$$

$$= -\log \prod_{j=0, j \neq m}^{2m} \frac{\exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|} \exp(u_k^T v_c)}$$

$$= - \sum_{j=0, j \neq m}^{2m} u_{c-m+j}^T v_c + 2m \log \sum_{k=1}^{|V|} \exp(u_k^T v_c)$$

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

word2vec - Skip Gram

Skip-gram
model

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

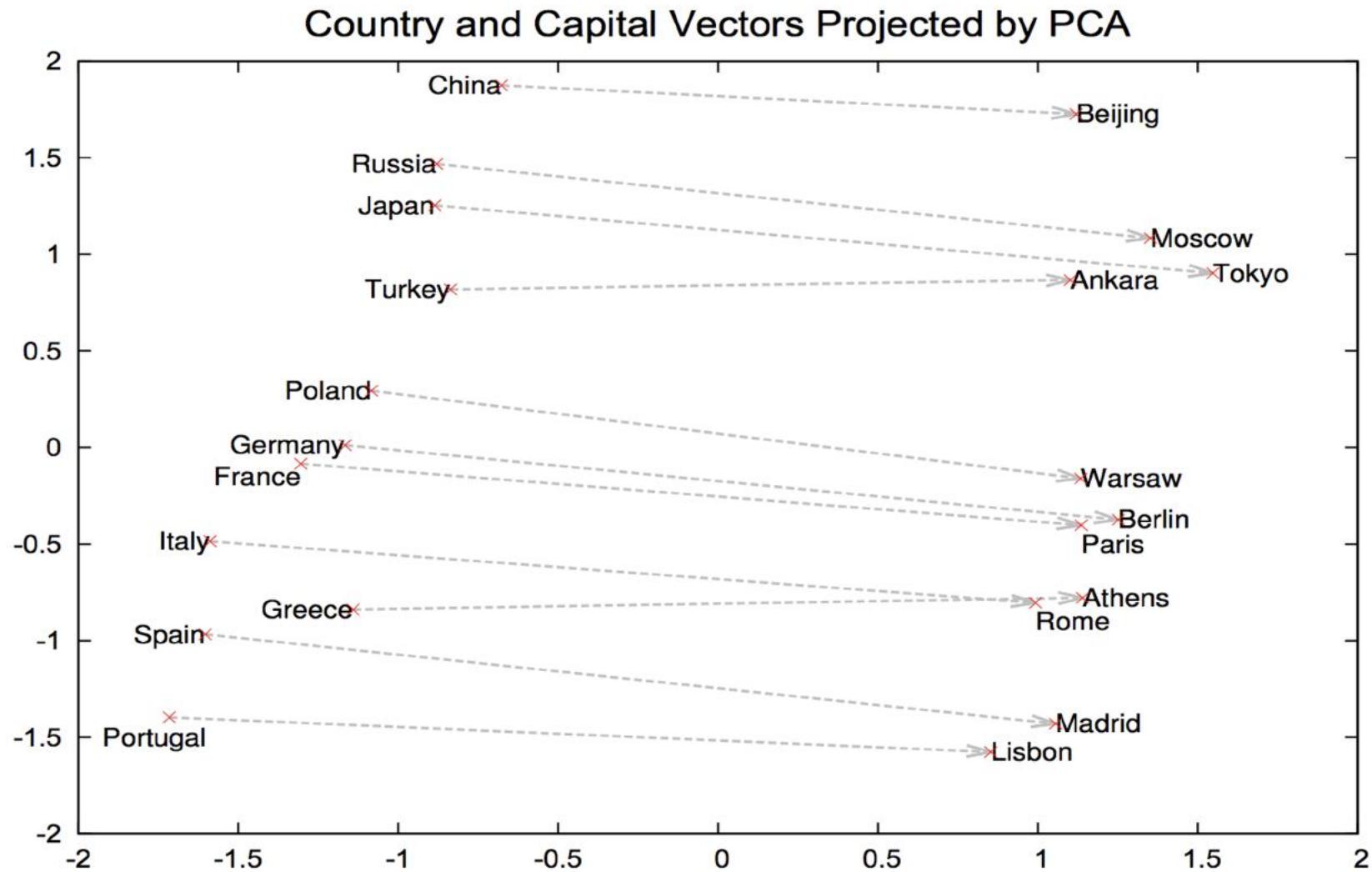
$$p(w_O | w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^W \exp\left({v'_{w}}^\top v_{w_I}\right)}$$

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

- Can use negative sampling

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right] \quad P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

word2vec



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	Rebecca 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 lubbock keene	plauen dzerzhinsky osterreich	reiki kohana karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated 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

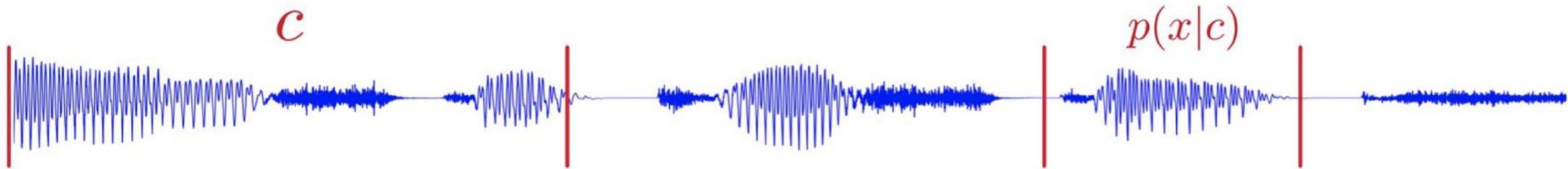
DeepMind

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.

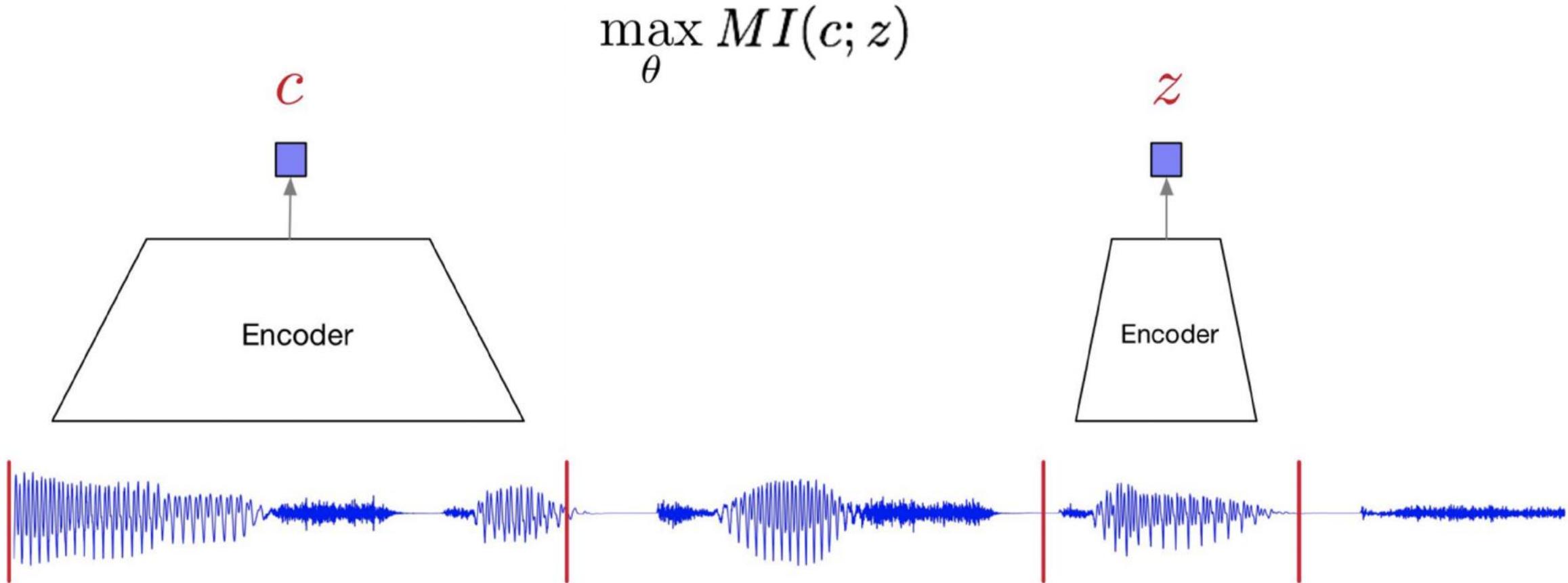
Contrastive Predictive Coding



Contrastive Predictive Coding



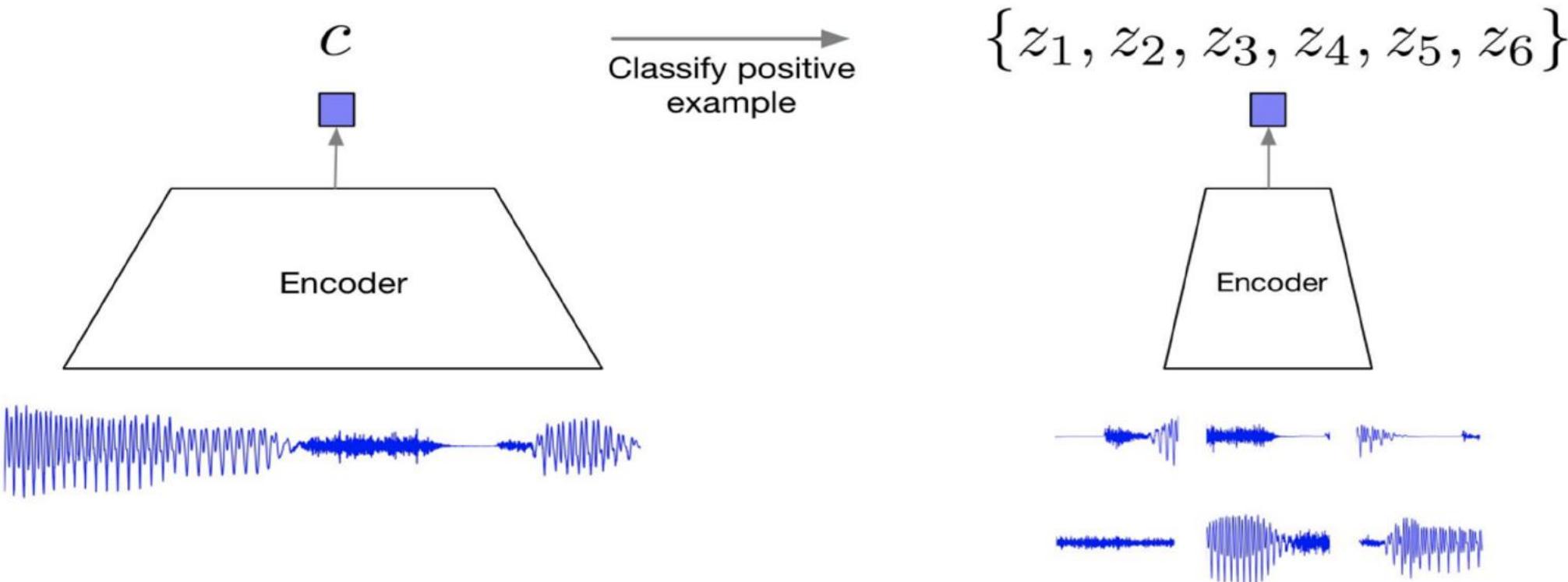
Contrastive Predictive Coding



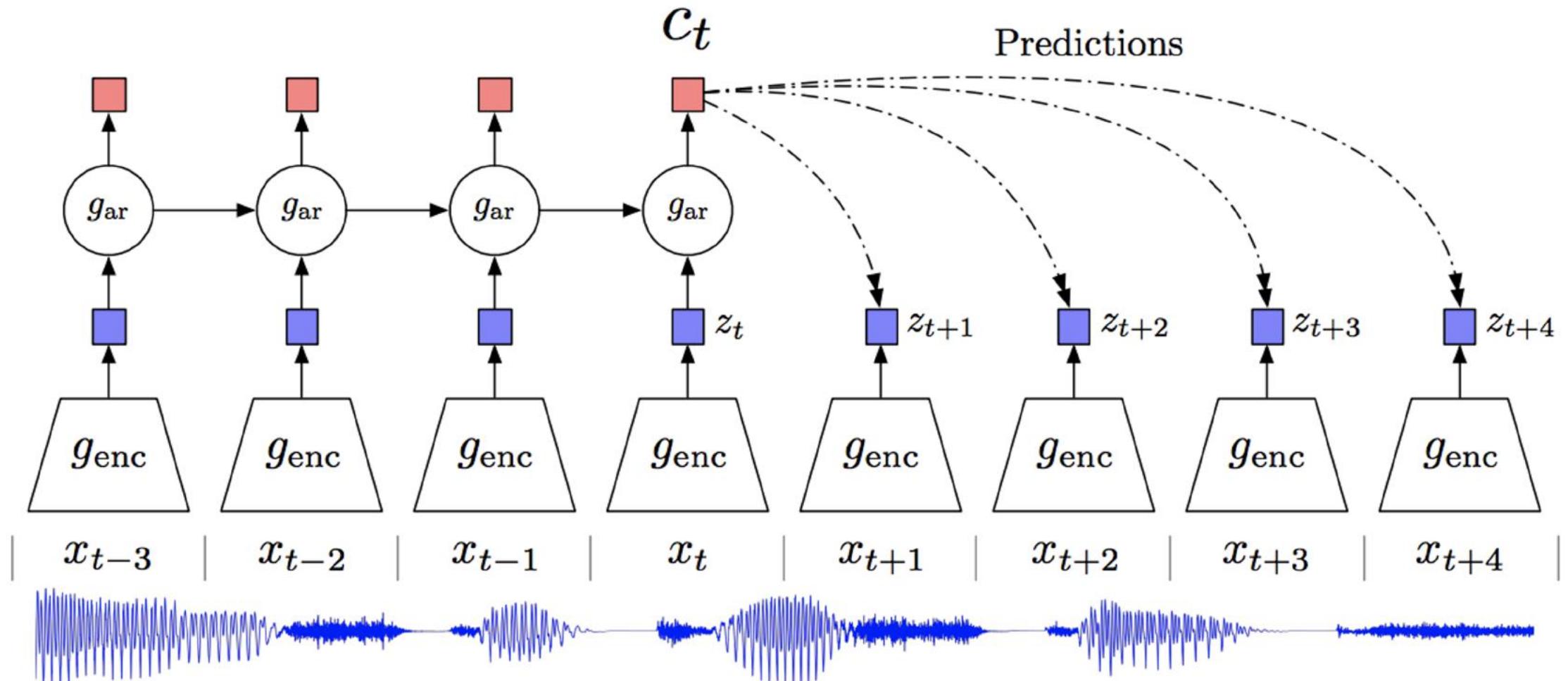
Contrastive Predictive Coding

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

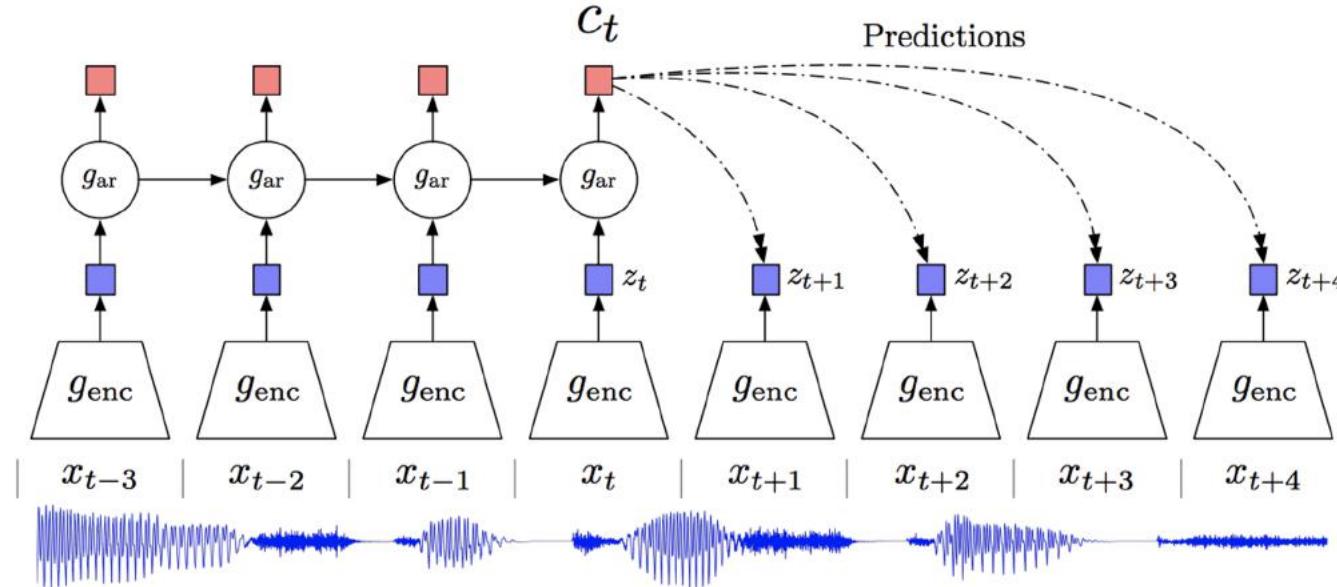
$$f_k(x_{t+k}, c_t) = \exp(z_{t+k}^T W_k c_t)$$



Contrastive Predictive Coding



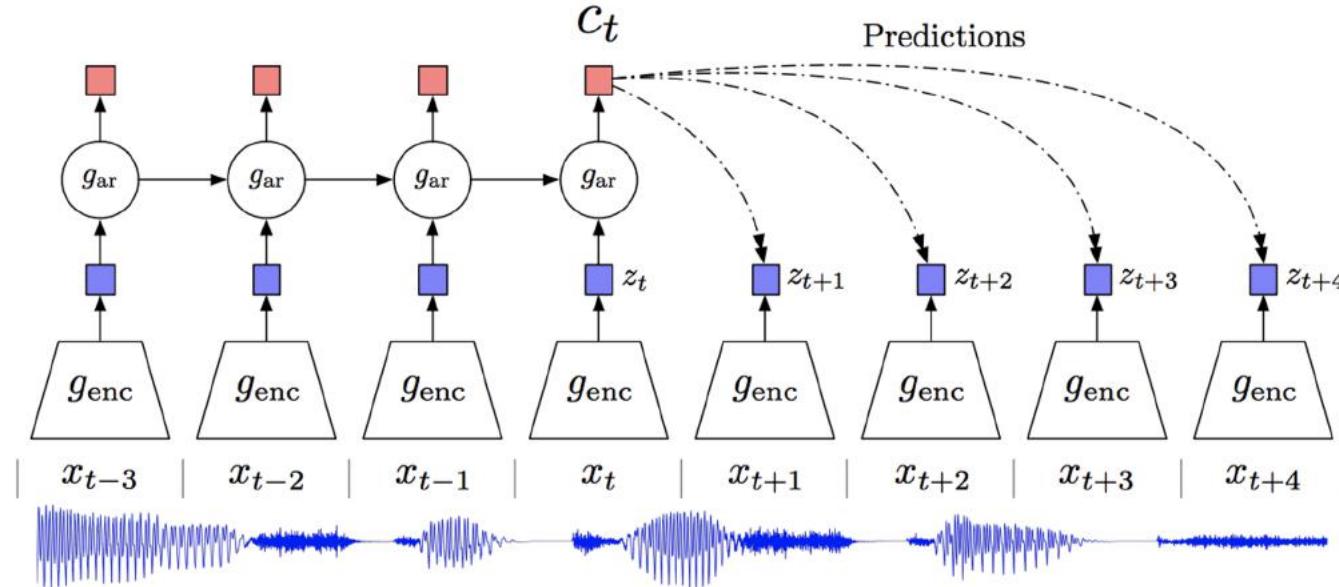
Contrastive Predictive Coding



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

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

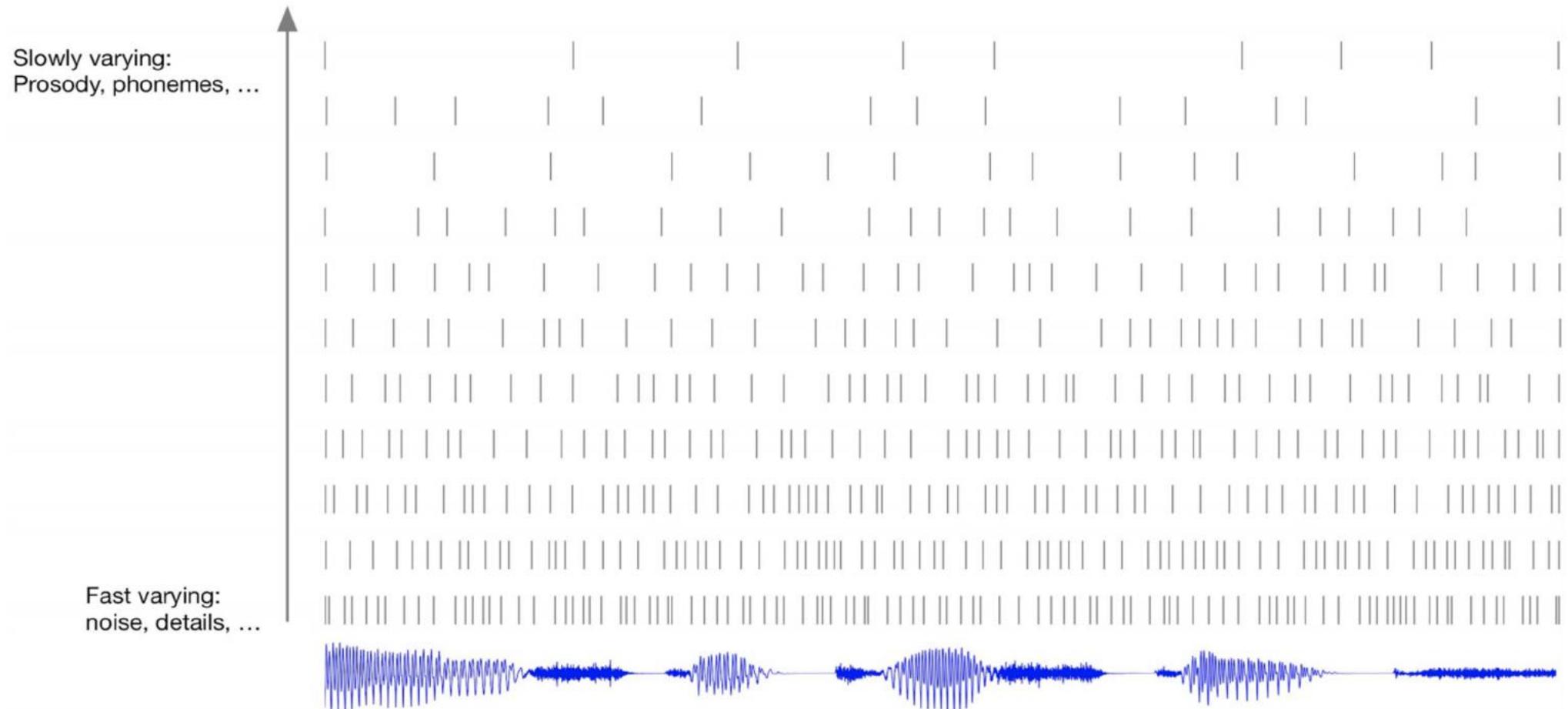
Contrastive Predictive Coding



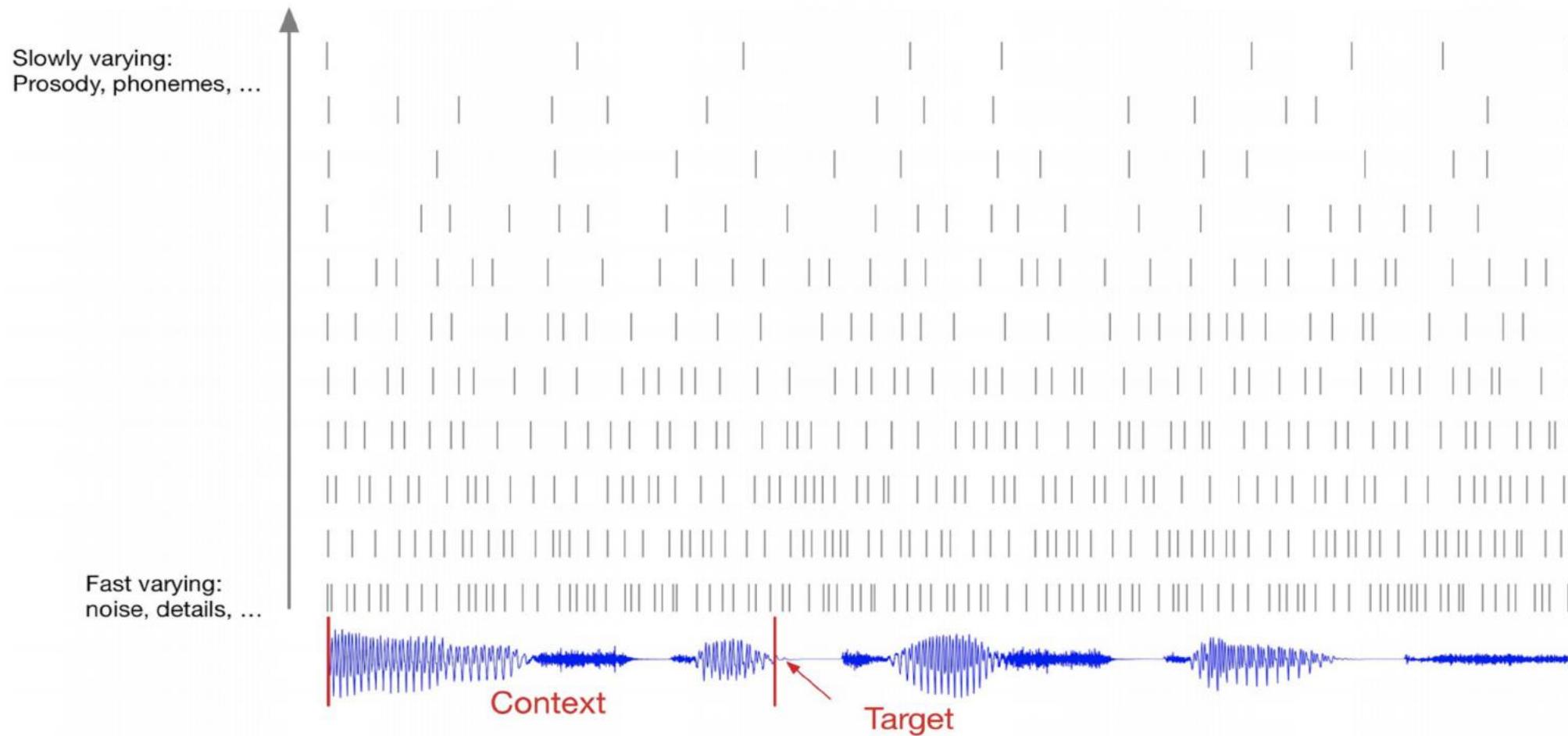
$$f_k(x_{t+k}, c_t) = \exp \left(z_{t+k}^T W_k c_t \right)$$

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

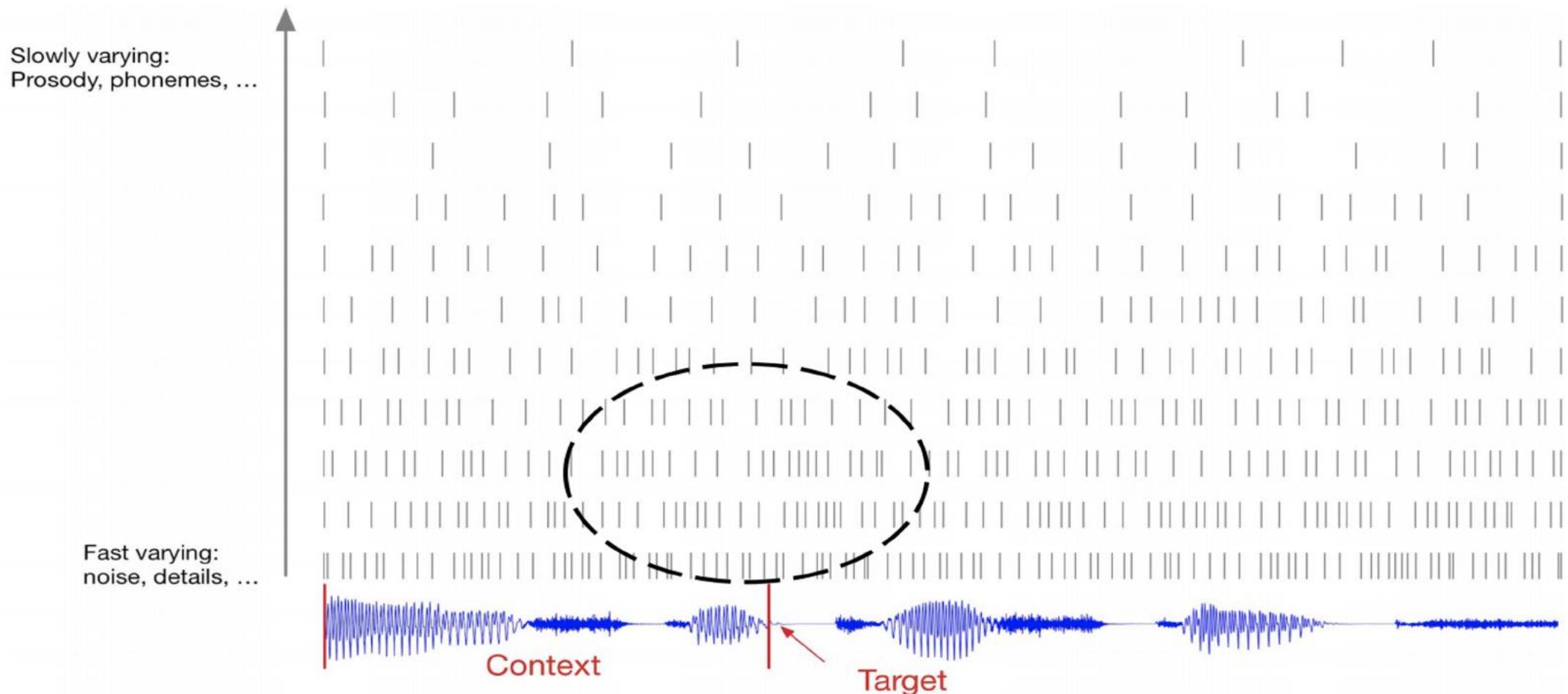
Contrastive Predictive Coding



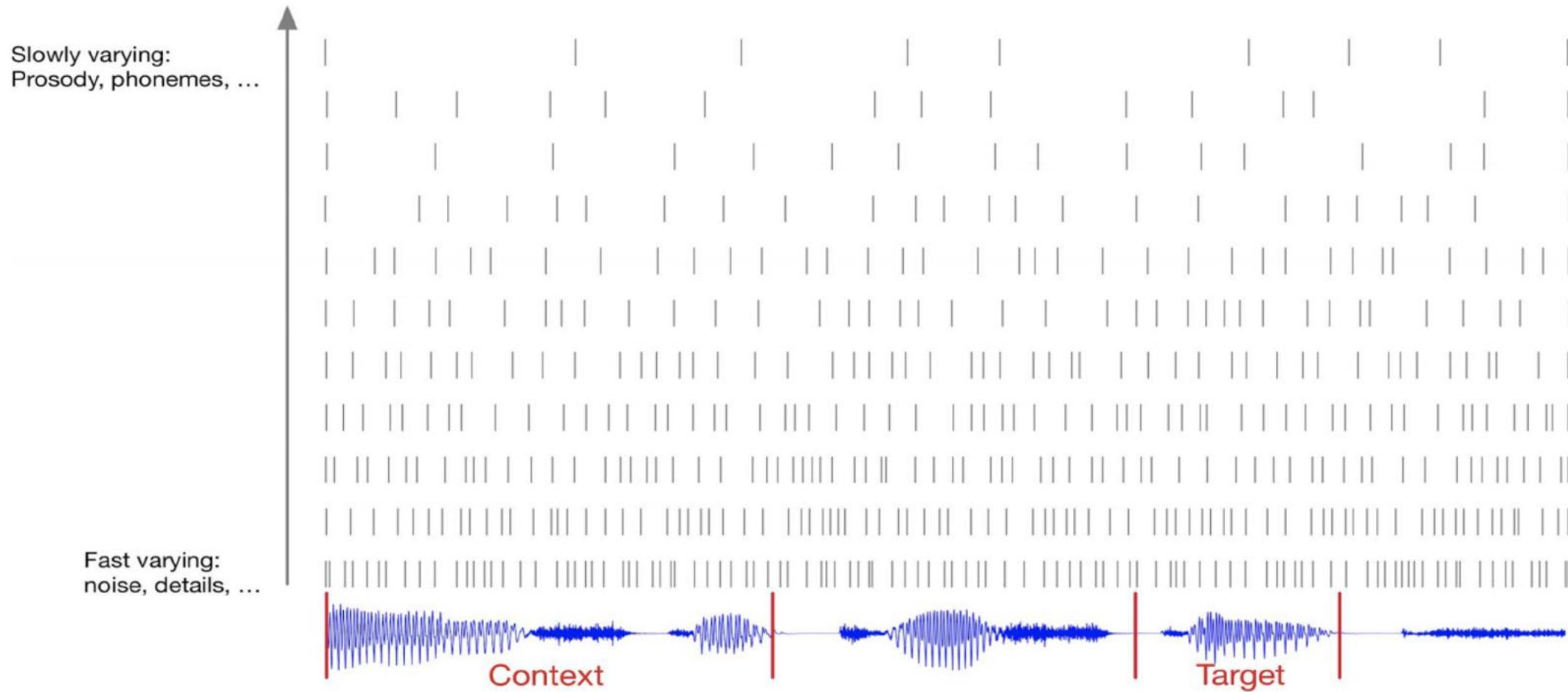
Contrastive Predictive Coding



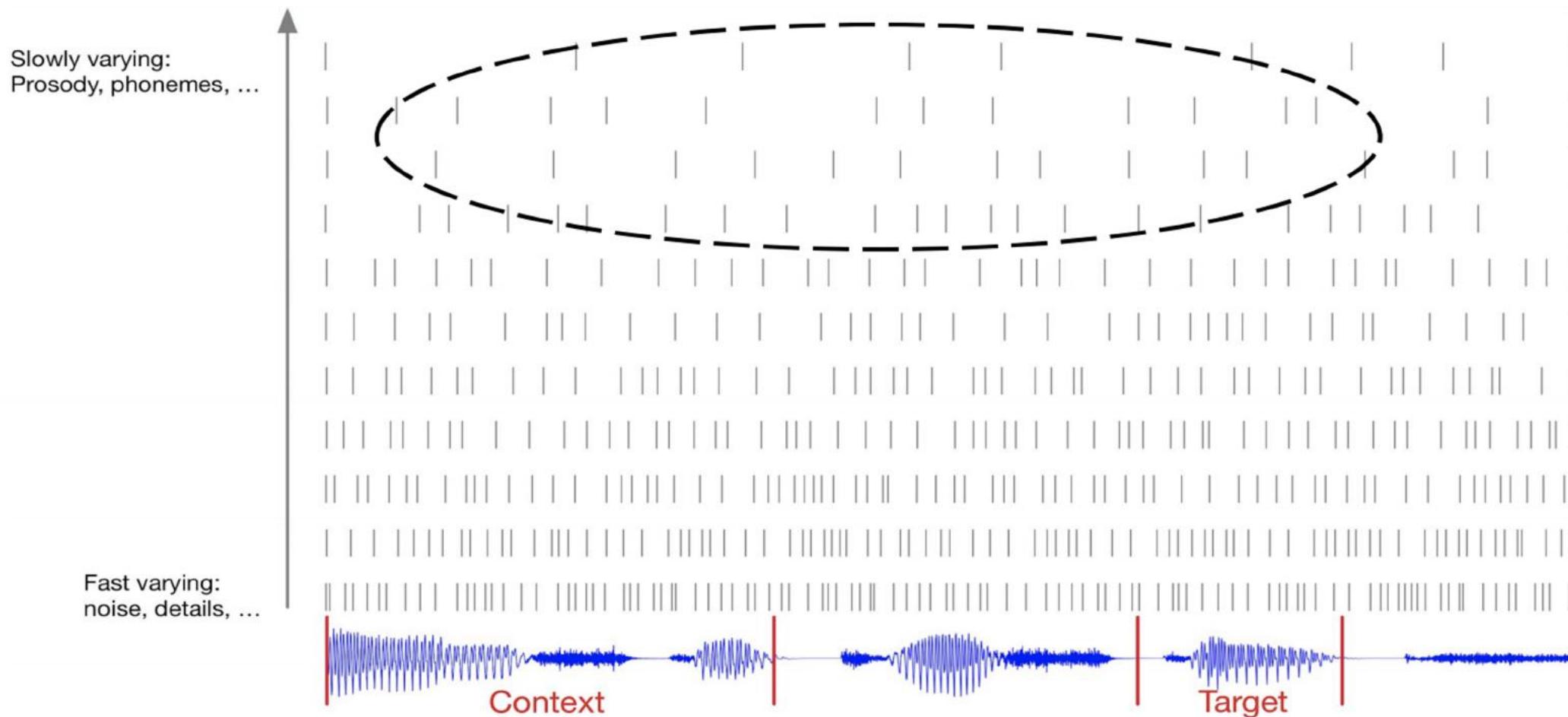
Contrastive Predictive Coding



Contrastive Predictive Coding



Contrastive Predictive Coding



CPC - Speech

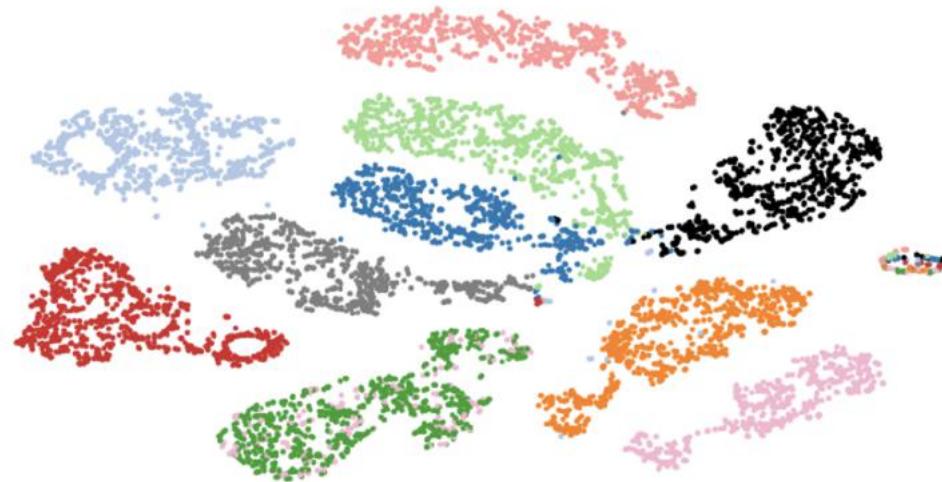


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

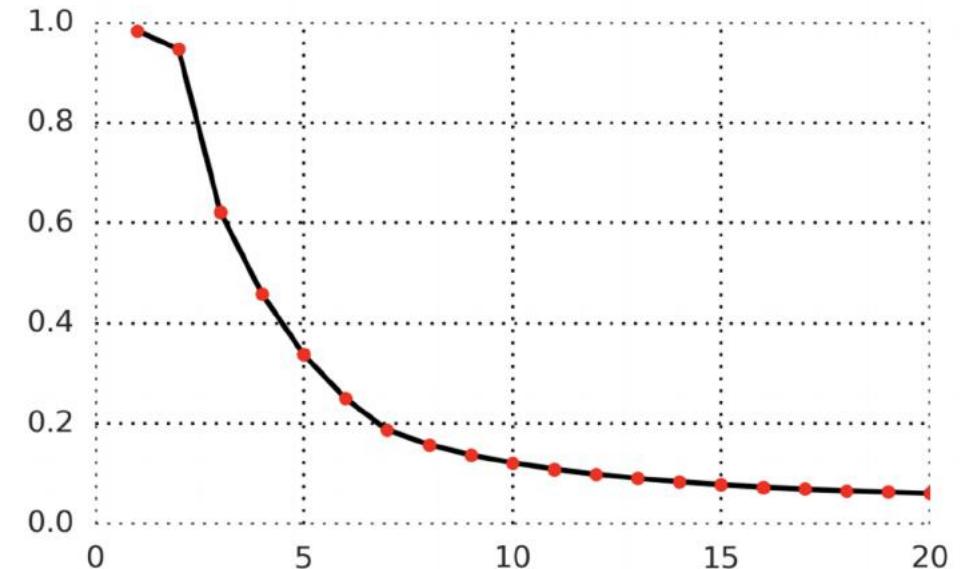


Figure 3: Average accuracy of predicting the positive sample in the contrastive loss for 1 to 20 latent steps in the future of a speech waveform. The model predicts up to 200ms in the future as every step consists of 10ms of audio.

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

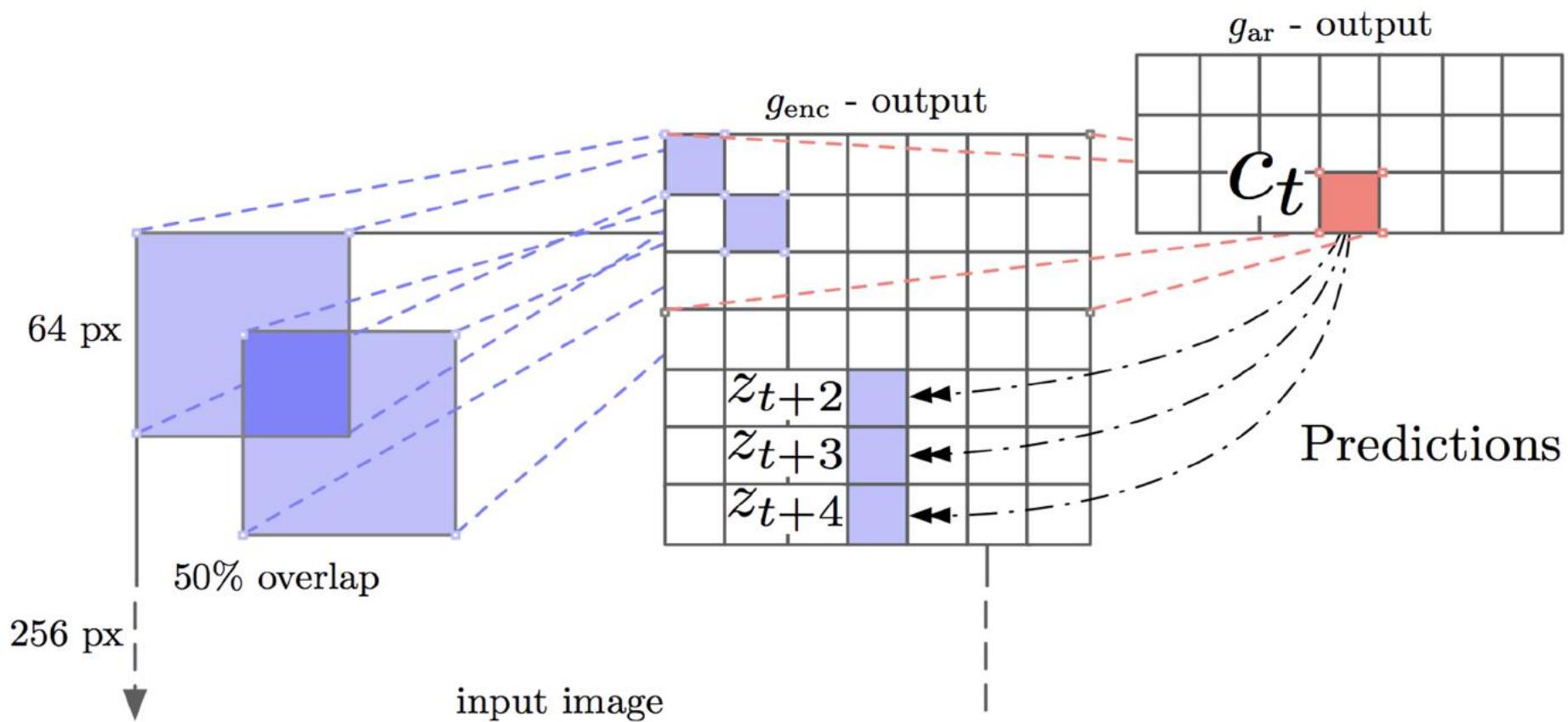
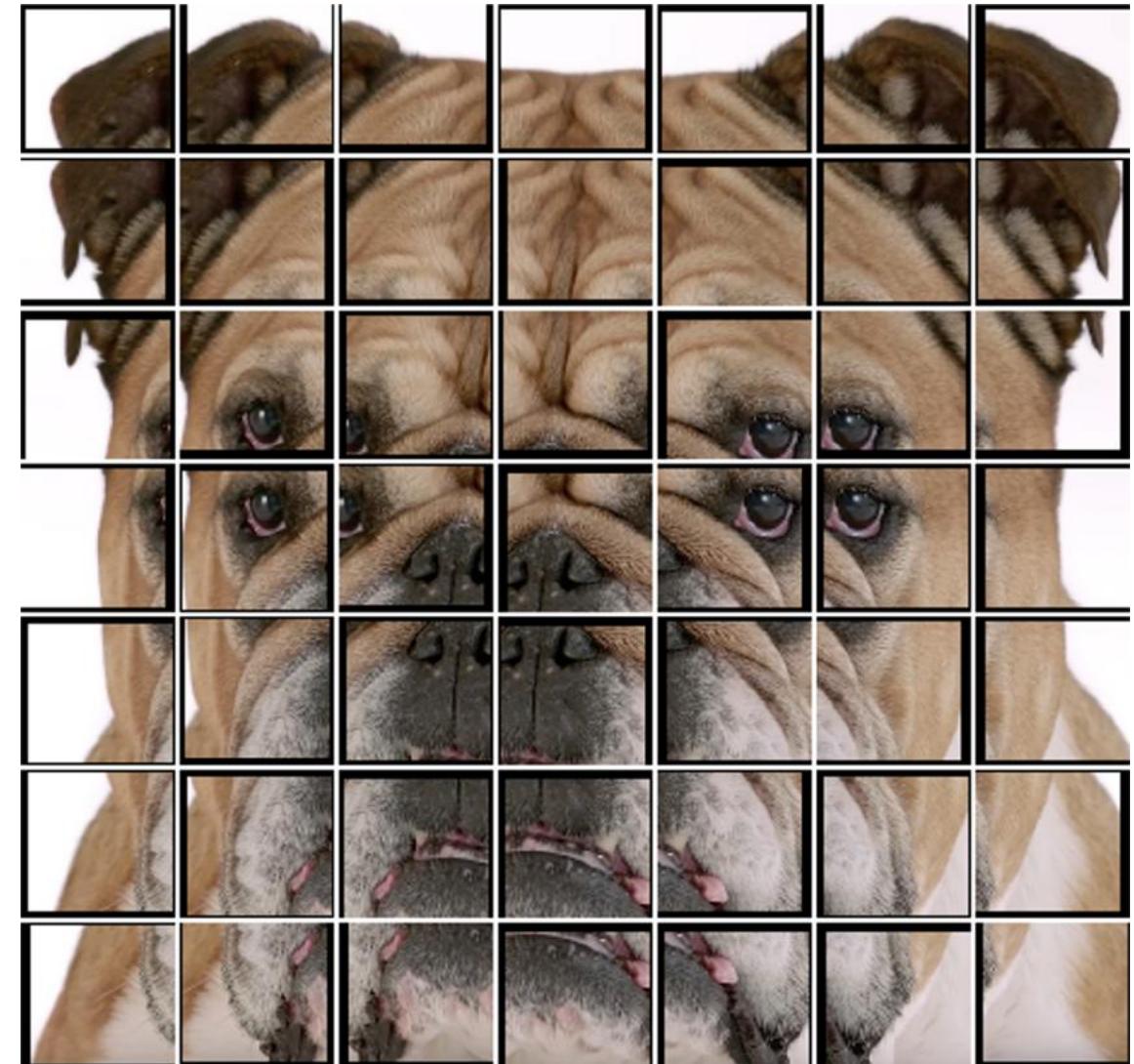


Figure 4: Visualization of Contrastive Predictive Coding for images (2D adaptation of Figure 1).

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

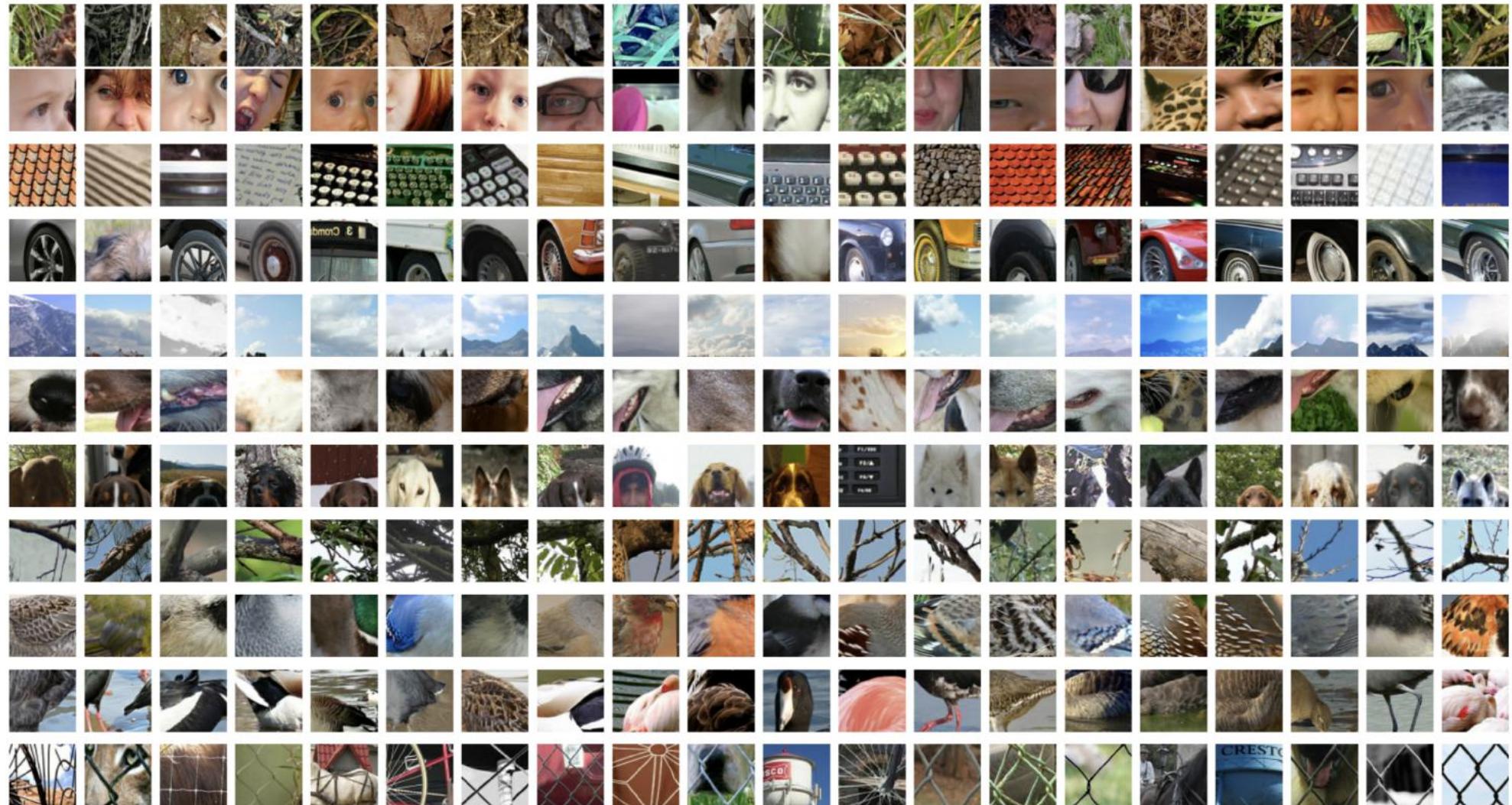


Figure 5: Every row shows image patches that activate a certain neuron in the CPC architecture.

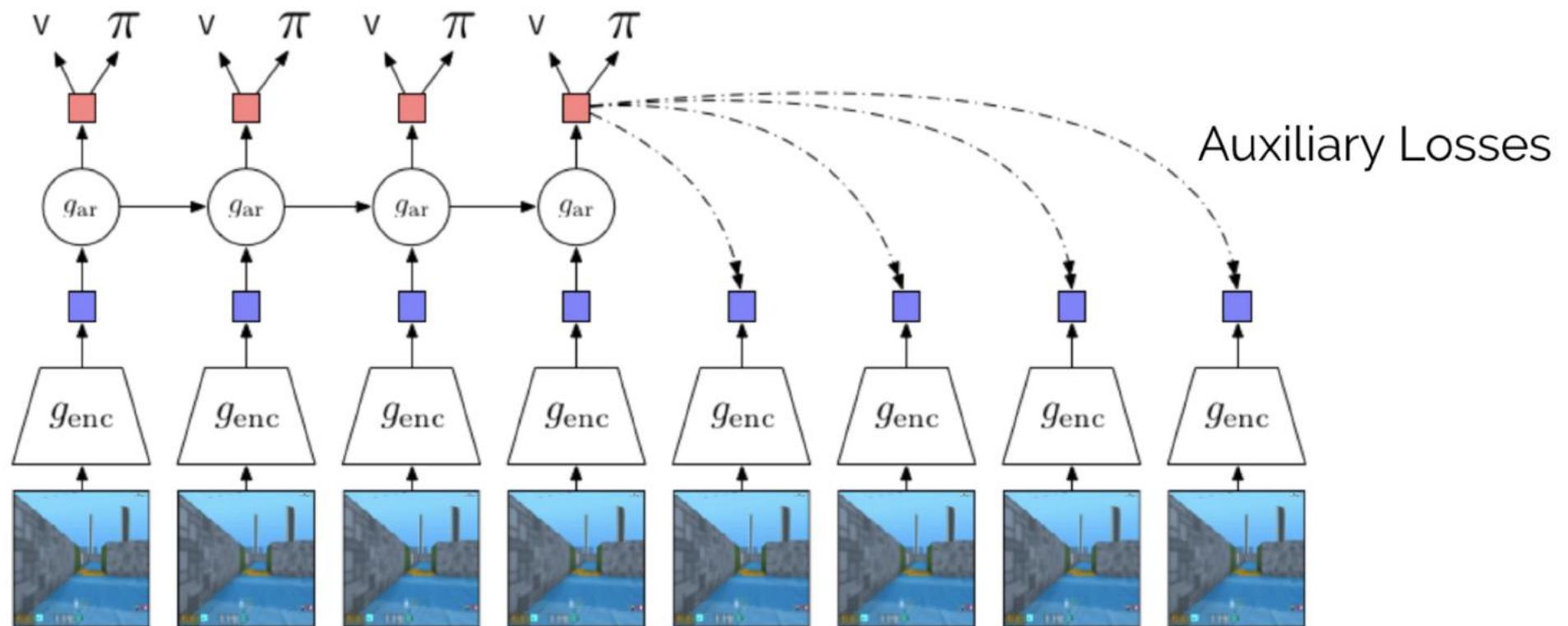
CPC - Natural Language Processing

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.

CPC - Reinforcement Learning

Auxiliary loss is on policy
Predict 30 steps in the future



CPCv2 - Large Scale CPC on ImageNet

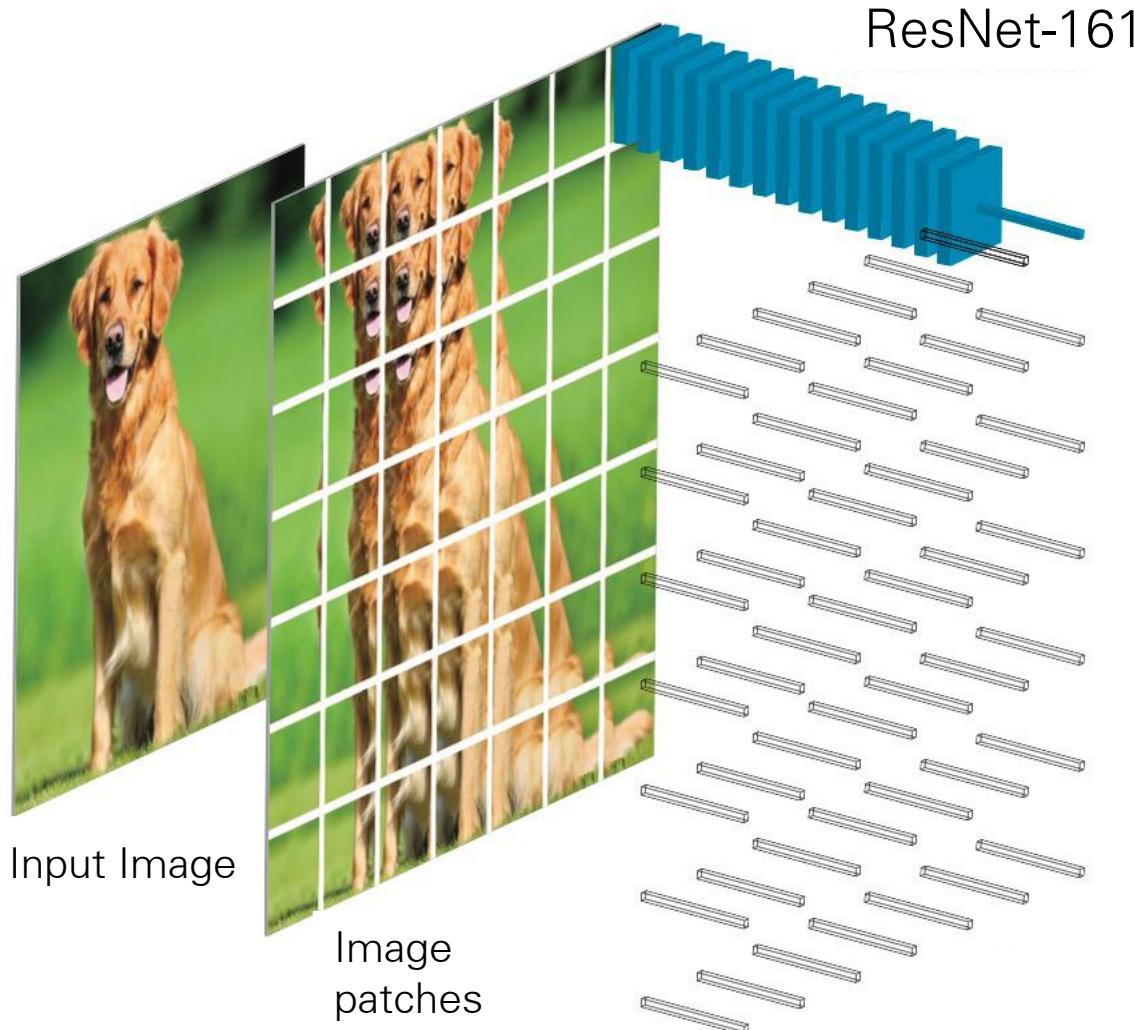
DATA-EFFICIENT IMAGE RECOGNITION WITH CONTRASTIVE PREDICTIVE CODING

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

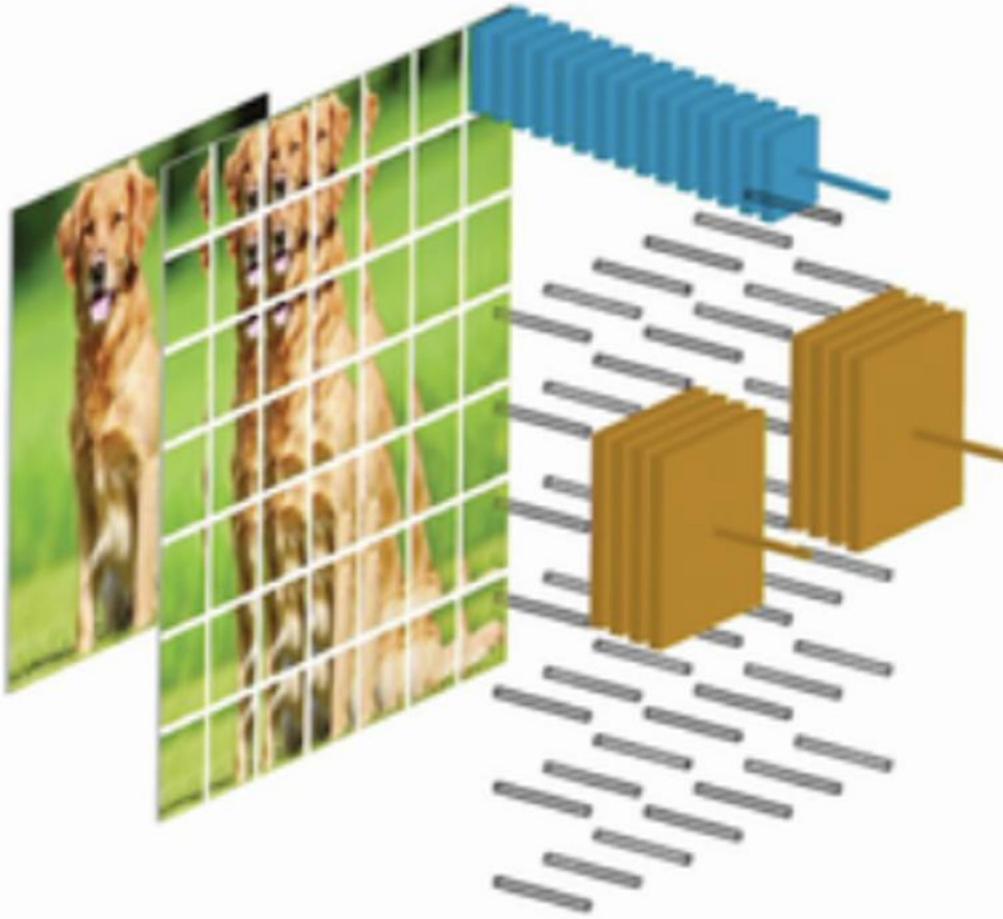
ABSTRACT

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 $2\text{--}5\times$ 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.

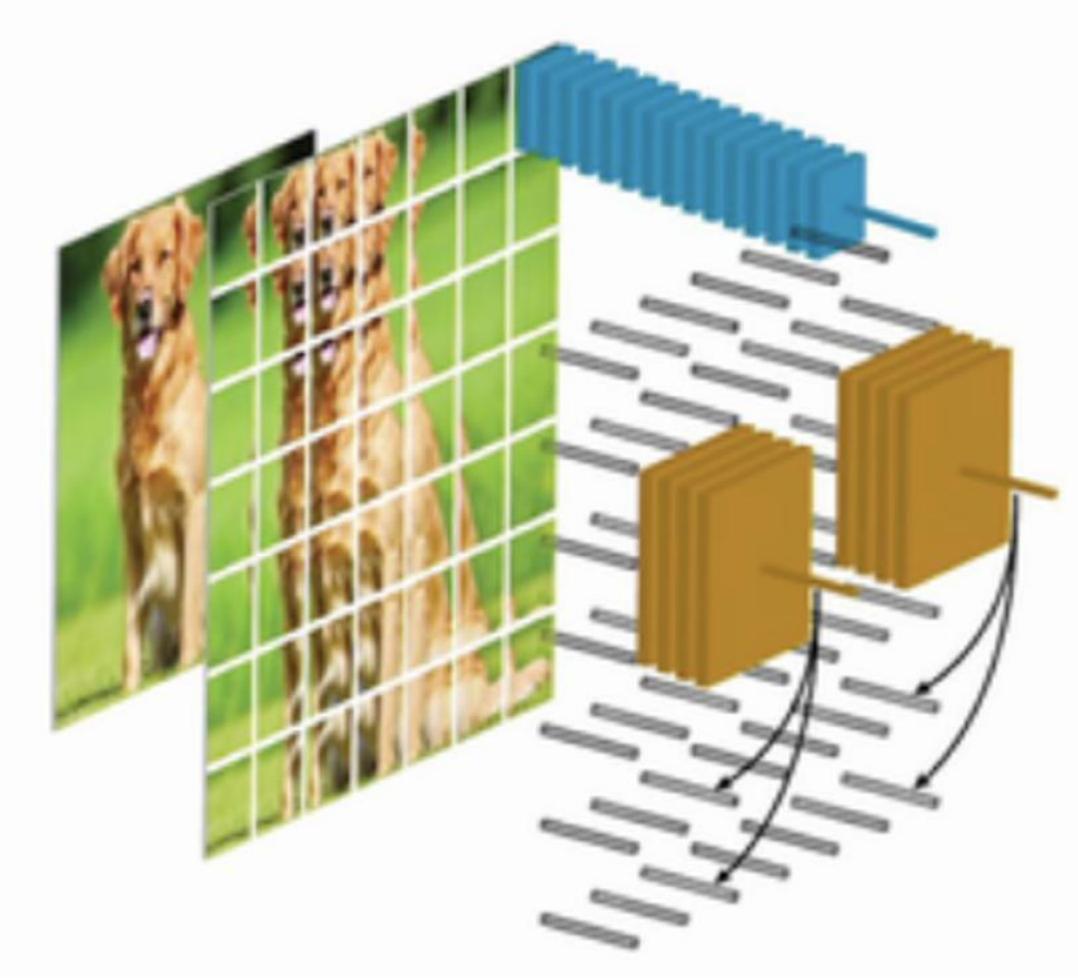
CPCv2 - Large Scale CPC on ImageNet



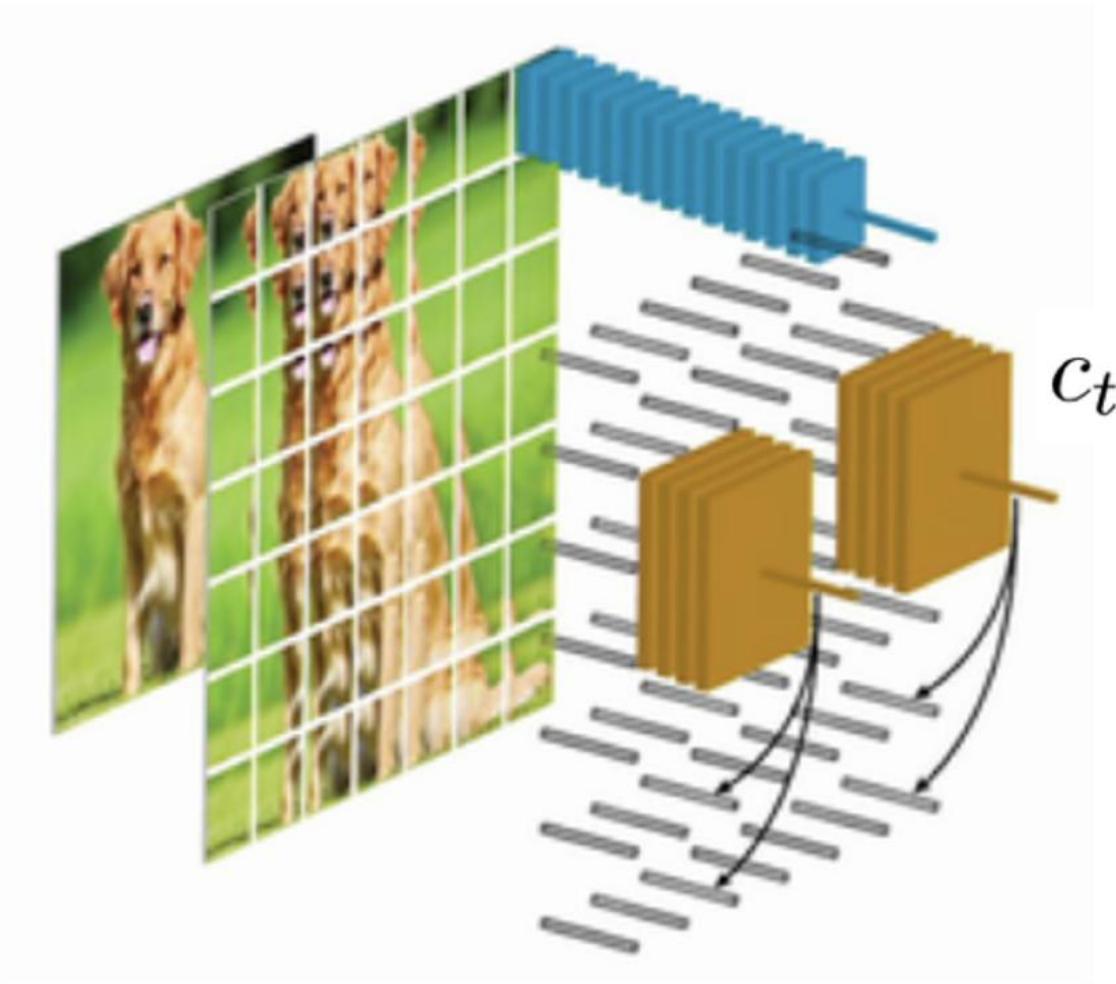
CPCv2 - Large Scale CPC on ImageNet



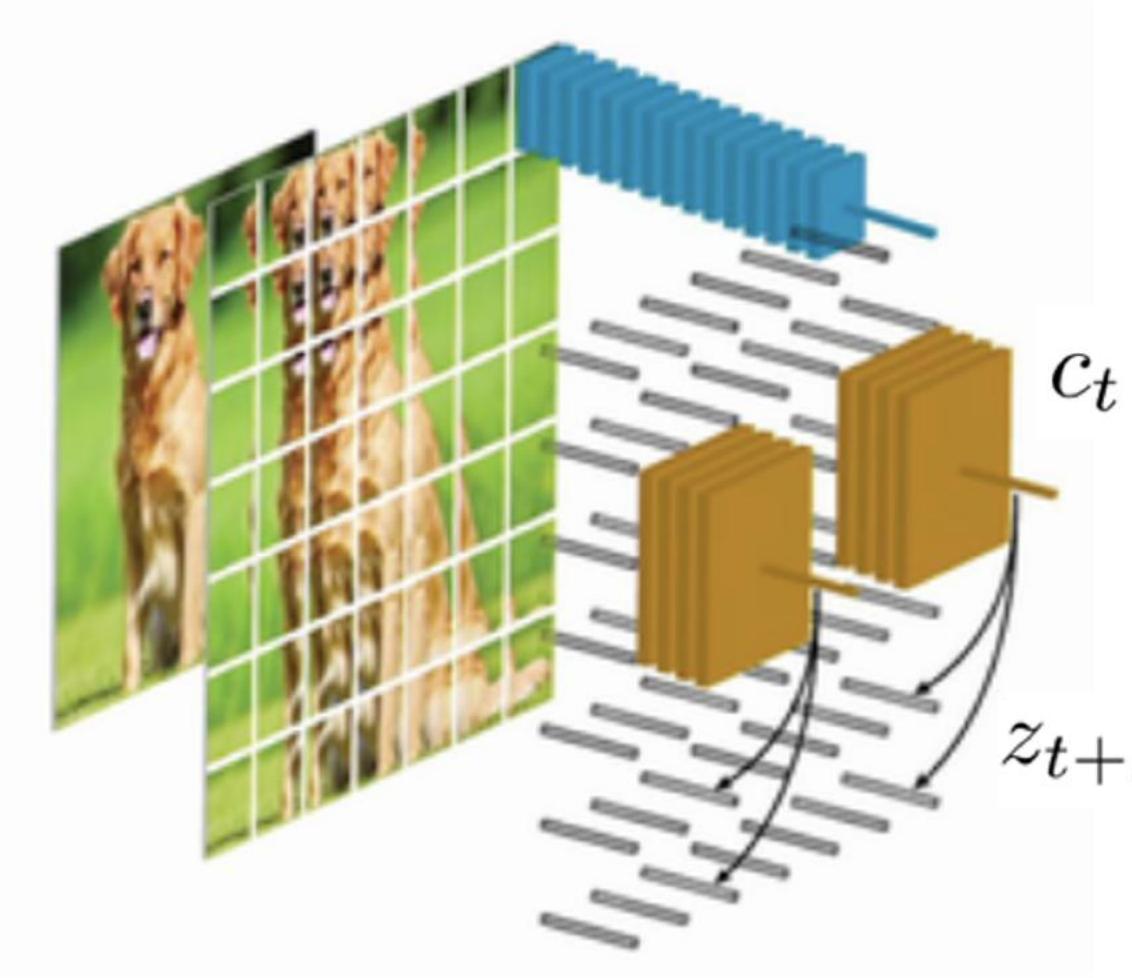
CPCv2 - Large Scale CPC on ImageNet



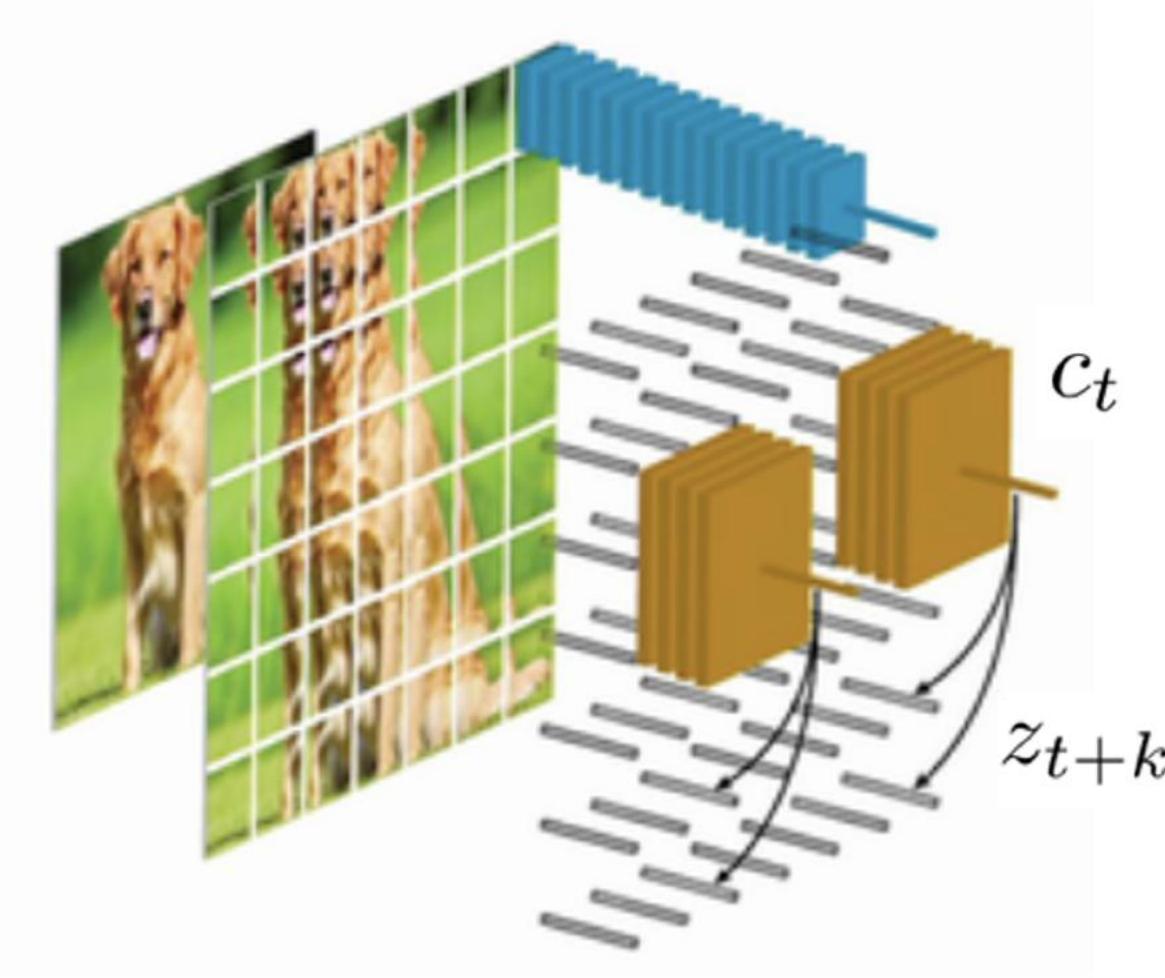
CPCv2 - Large Scale CPC on ImageNet



CPCv2 - Large Scale CPC on ImageNet

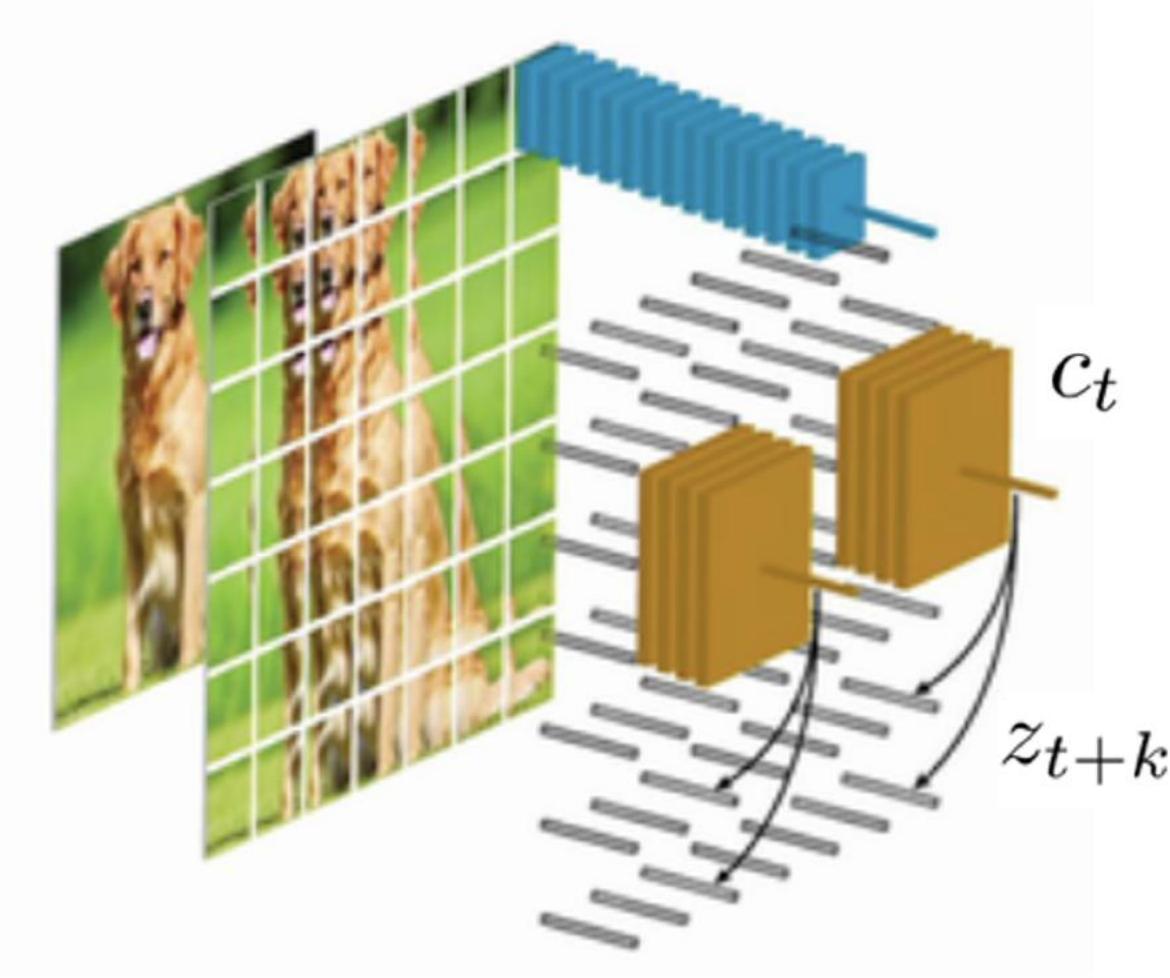


CPCv2 - Large Scale CPC on ImageNet



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

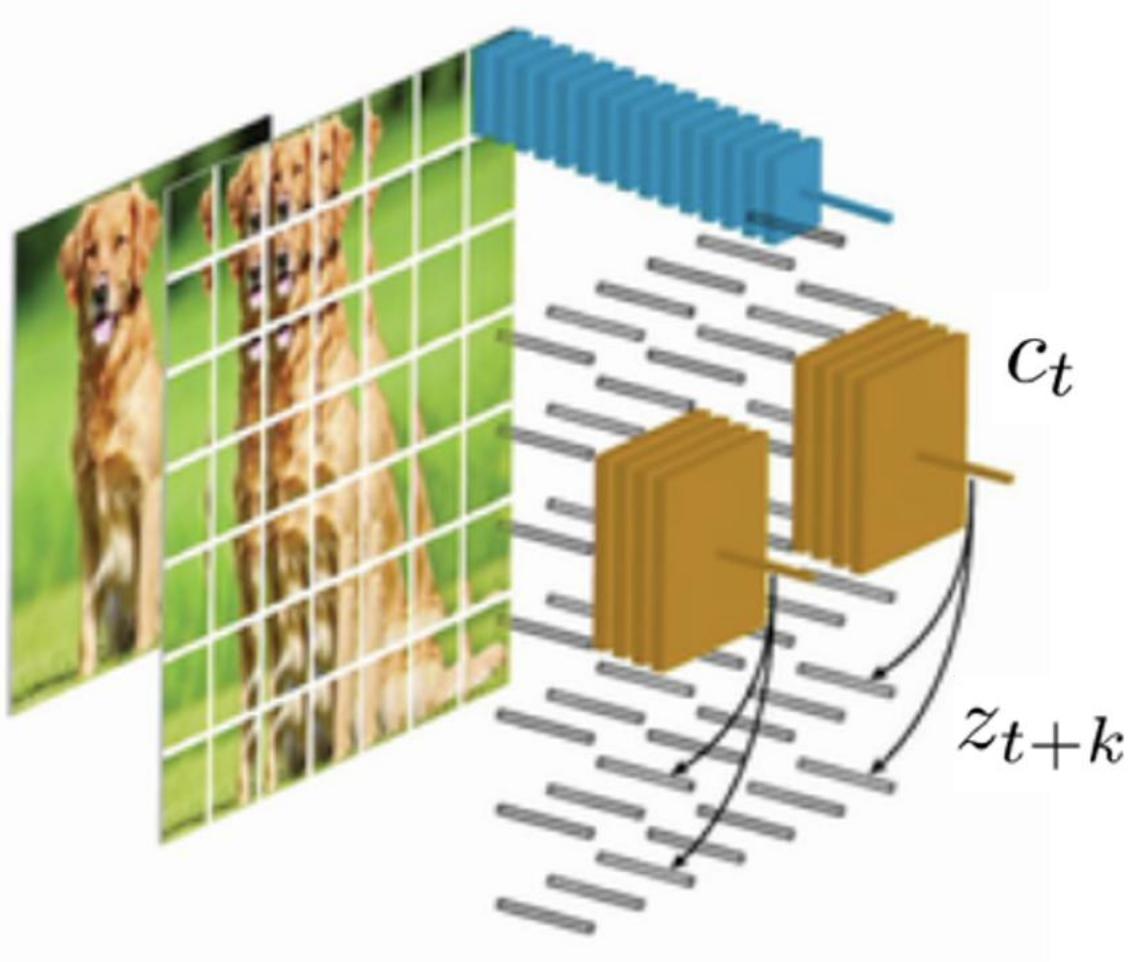
CPCv2 - Large Scale CPC on ImageNet



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

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

CPCv2 - Large Scale CPC on ImageNet



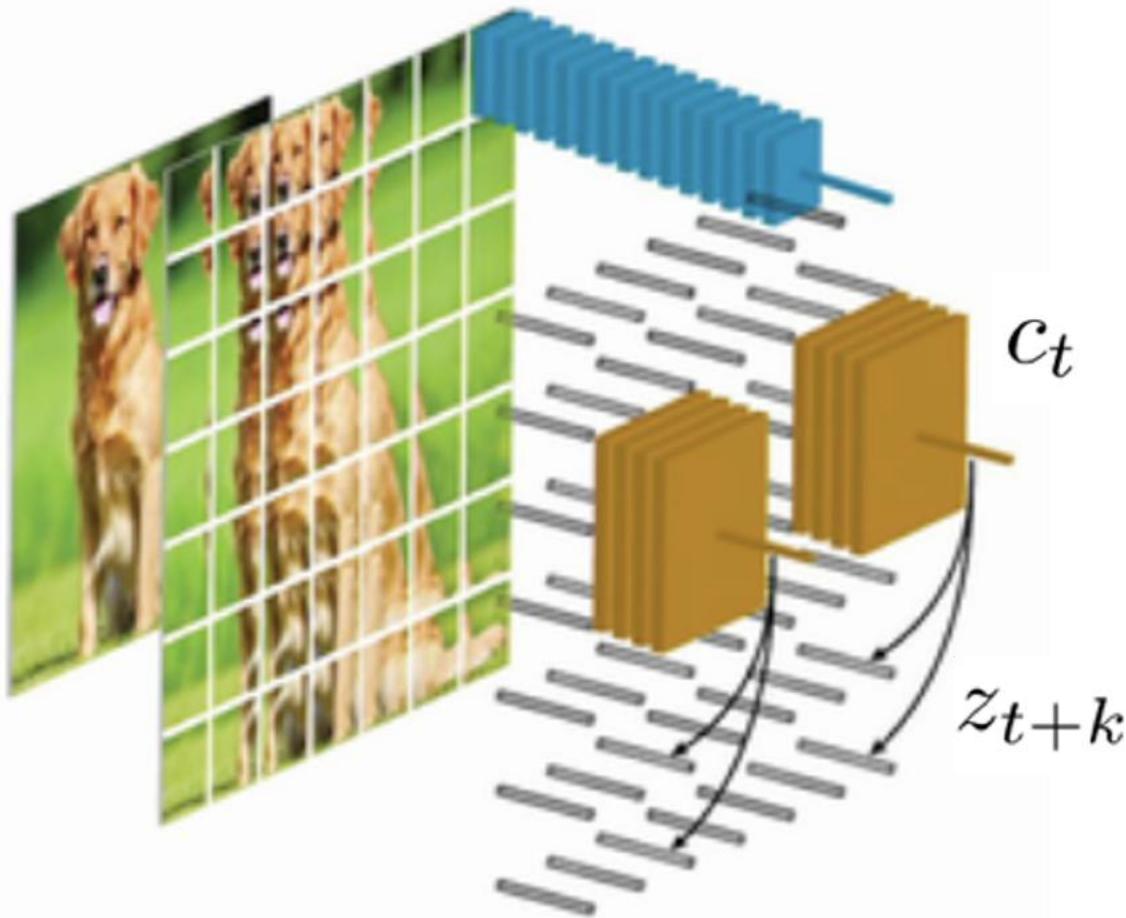
$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

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

↓ Negatives

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

CPCv2 - Large Scale CPC on ImageNet



InfoNCE Loss

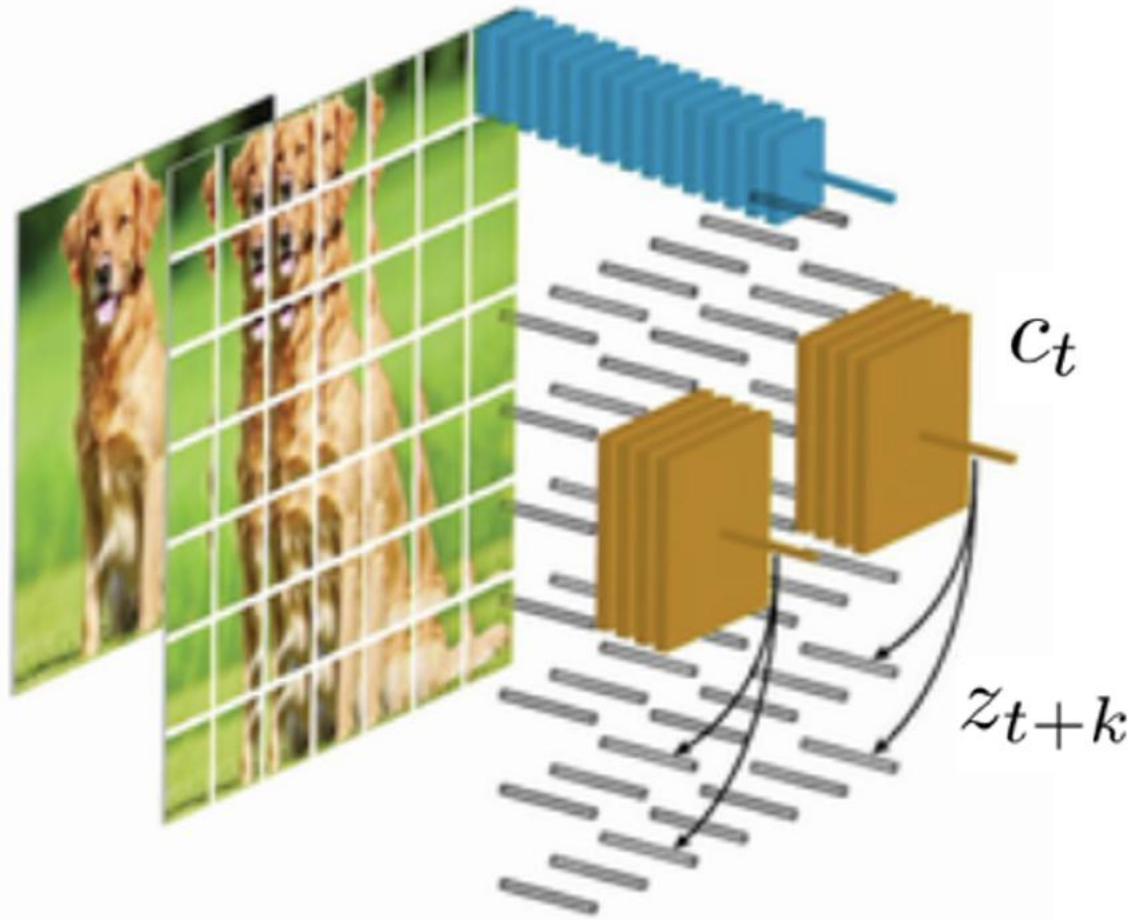
$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

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

↓
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

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

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

↓
Negatives

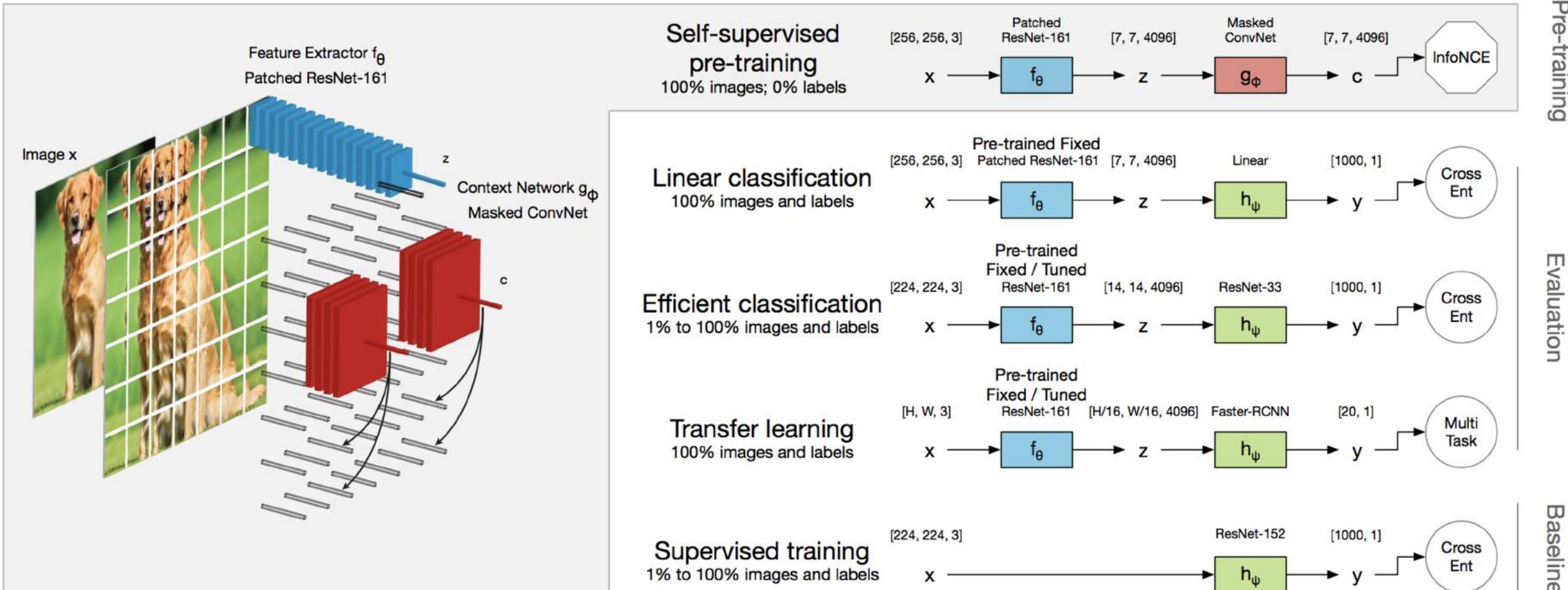
1. Other patches within image
2. Patches from other images

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 = $16 * 36 = 576$

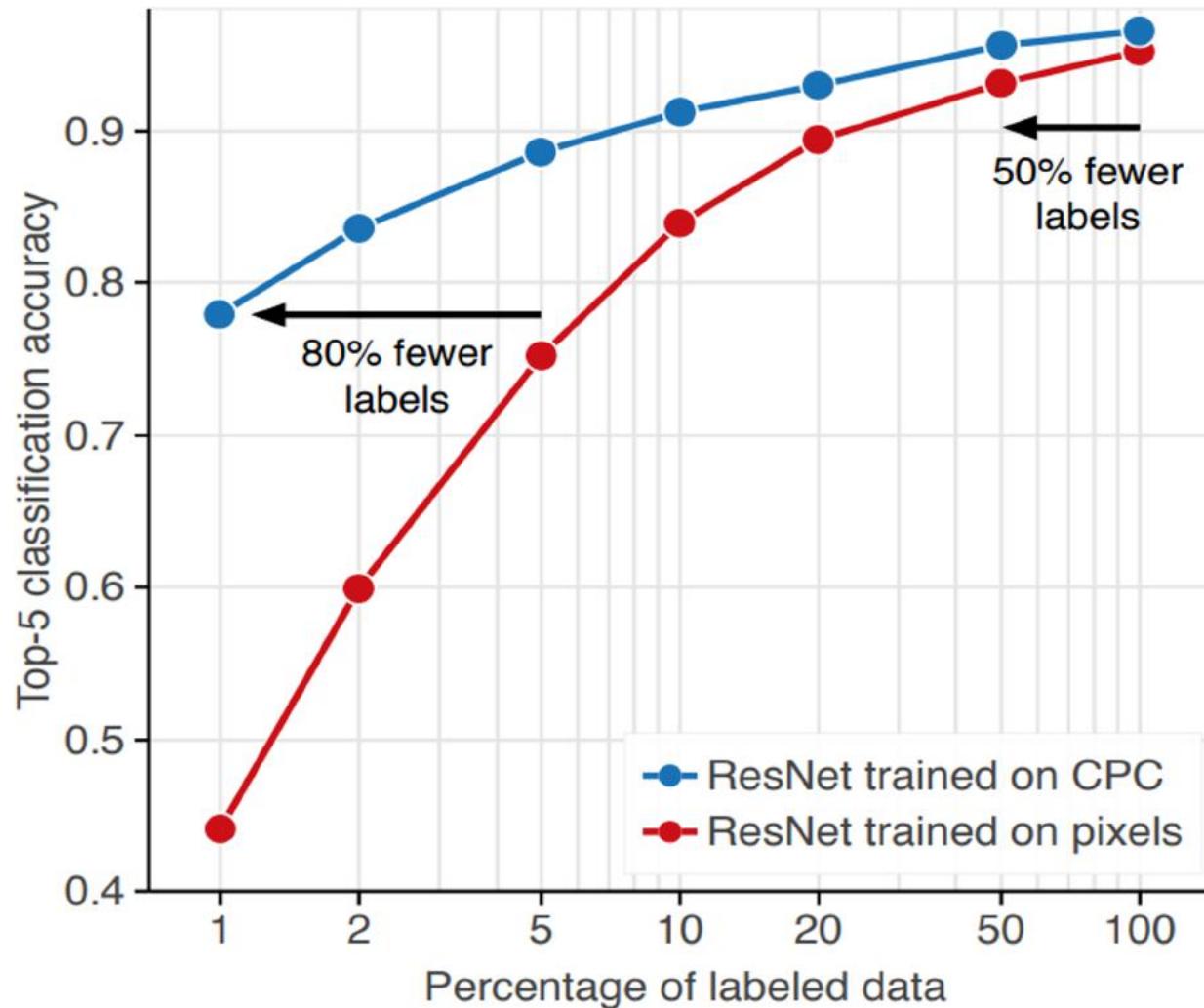
CPCv2 - Large Scale CPC on ImageNet



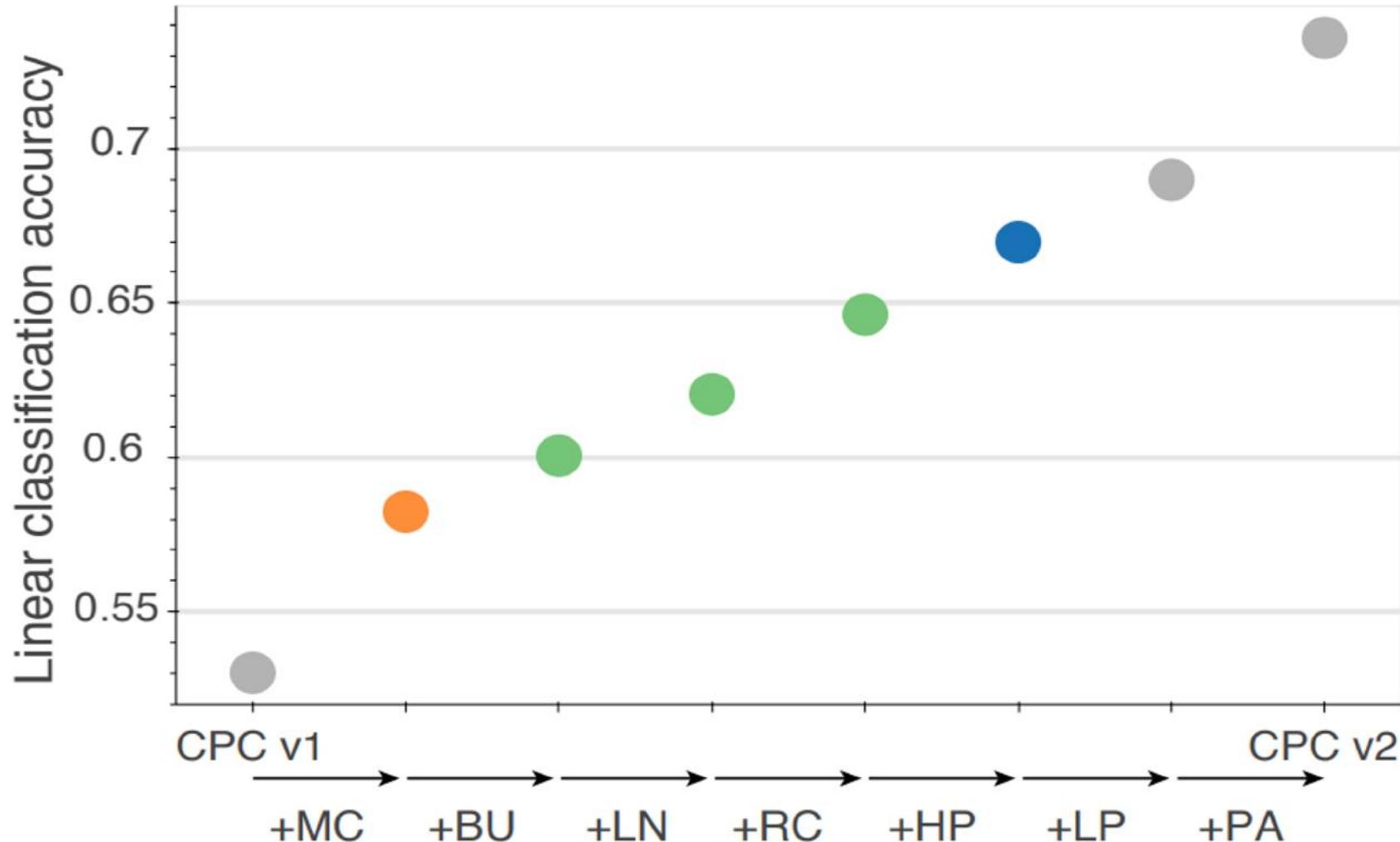
CPCv2 - Linear Classification

METHOD	PARAMS (M)	TOP-1	TOP-5
<i>Methods using ResNet-50:</i>			
INSTANCE DISCR. [1]	24	54.0	-
LOCAL AGGR. [2]	24	58.8	-
MoCo [3]	24	60.6	-
PIRL [4]	24	63.6	-
CPC v2 - RESNET-50	24	63.8	85.3
<i>Methods using different architectures:</i>			
MULTI-TASK [5]	28	-	69.3
ROTATION [6]	86	55.4	-
CPC v1 [7]	28	48.7	73.6
BIGBiGAN [8]	86	61.3	81.9
AMDIM [9]	626	68.1	-
CMC [10]	188	68.4	88.2
MoCo [2]	375	68.6	-
CPC v2 - RESNET-161	305	71.5	90.1

CPCv2 - Data-Efficient Image Recognition



CPCv1 → CPCv2



CPCv2 - Data-Efficient Supervised Learning

Method	Architecture	Top-5 accuracy				
		1%	5%	10%	50%	100%
Labeled data						
† 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 $\times 4$	-	-	91.2	-	95.0
Methods using representation learning only:						
Instance Discrimination [60]	ResNet-50	39.2	-	77.4	-	-
Rotation [63]	ResNet-152 $\times 2$	57.5	-	86.4	-	-
ResNet on BigBiGAN (fixed)	RevNet-50 $\times 4$	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
<i>Transfer from labeled data:</i>		
Supervised baseline	ResNet-152	74.7
<i>Transfer from unlabeled data:</i>		
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



Instance Discrimination



1. MoCo
2. SimCLR

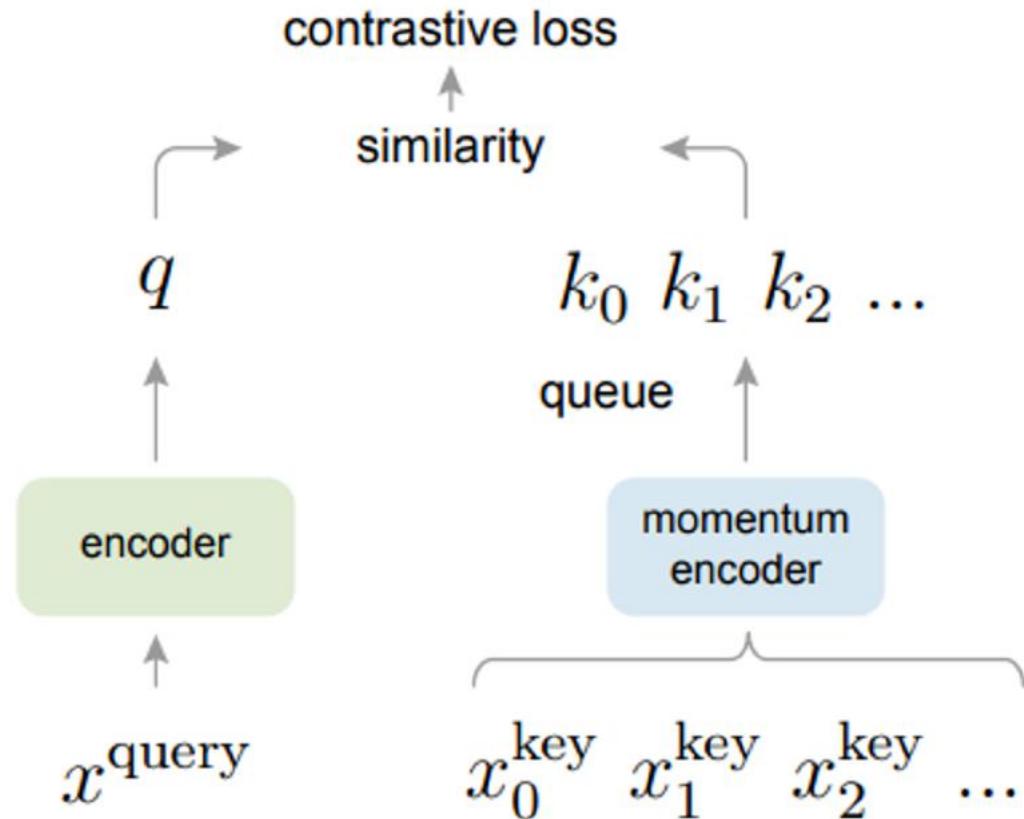
Momentum Contrast (MoCo)

Momentum Contrast for Unsupervised Visual Representation Learning

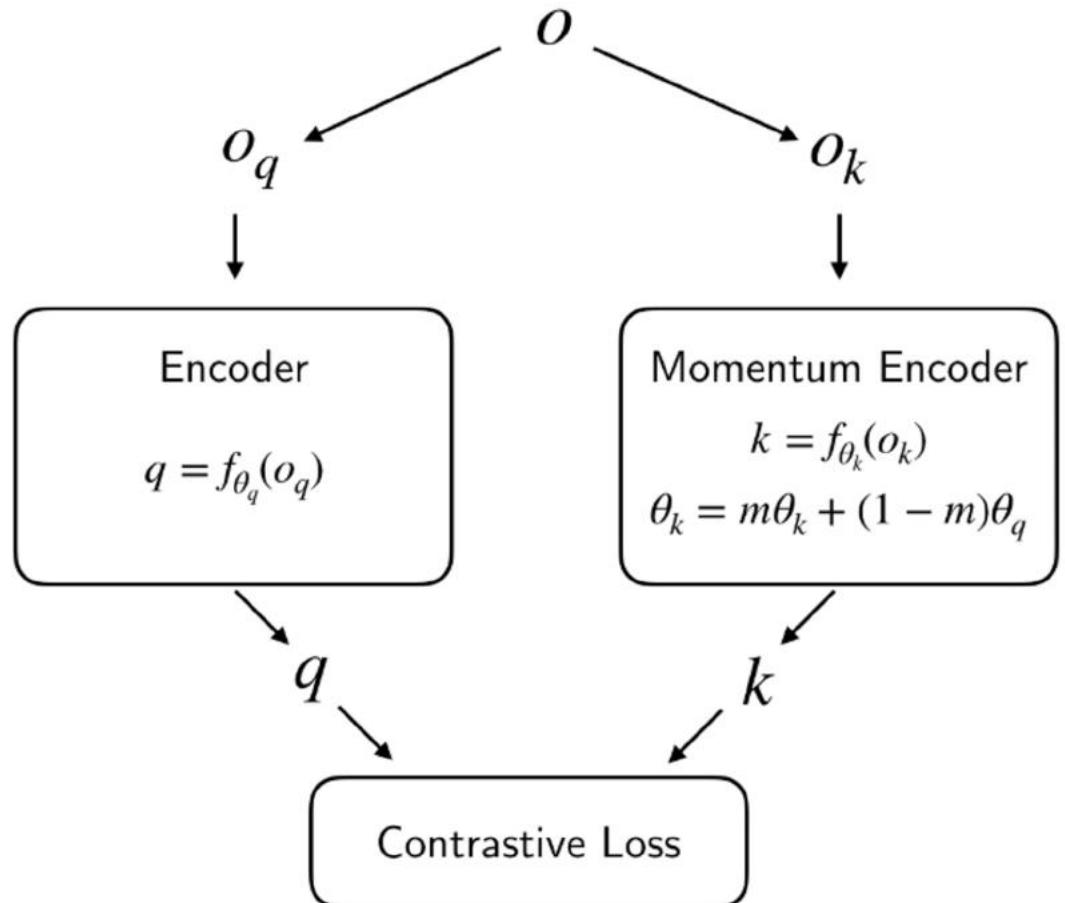
Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick

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)

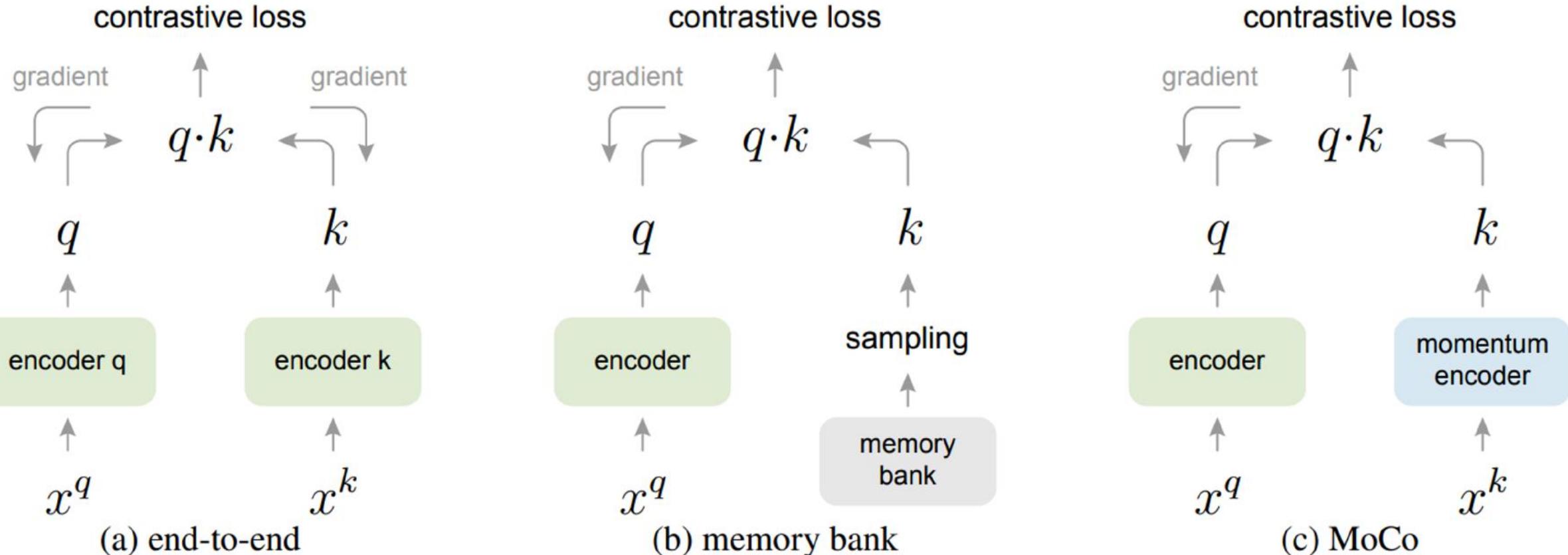
    # SGD update: query network
    loss.backward()
    update(f_q.params)

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

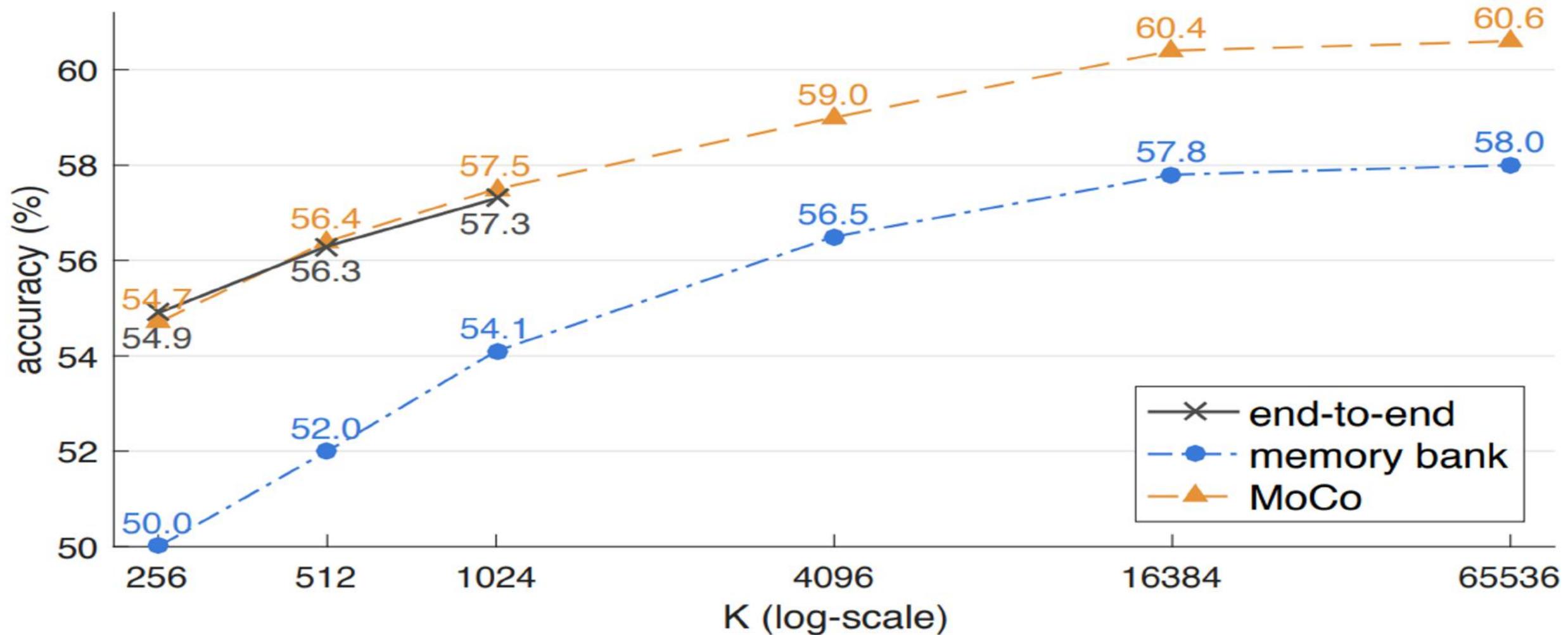
    # update dictionary
    enqueue(queue, k) # enqueue the current minibatch
    dequeue(queue) # dequeue the earliest minibatch
```

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

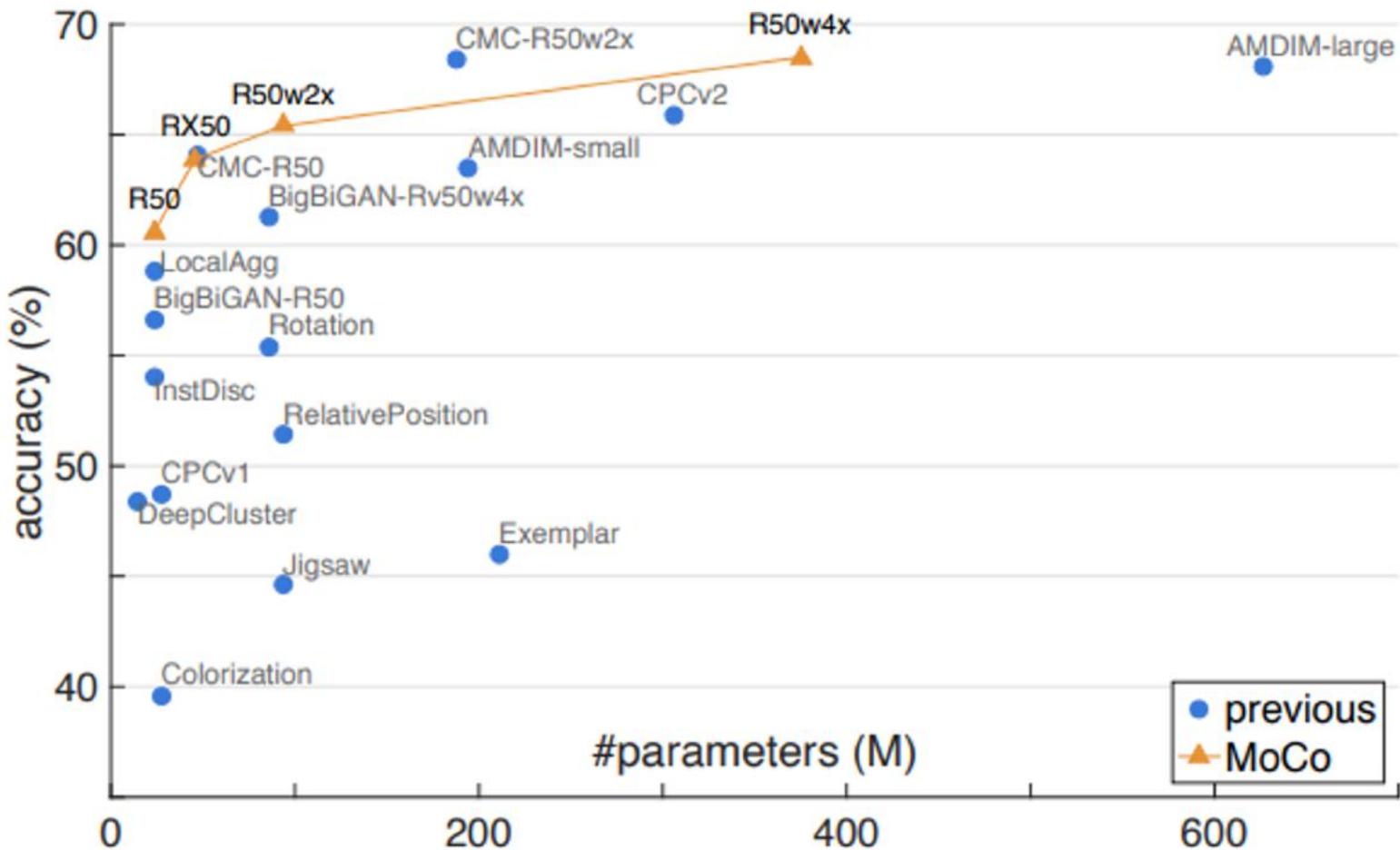
Momentum Contrast (MoCo)



Momentum Contrast (MoCo)



Momentum Contrast (MoCo)

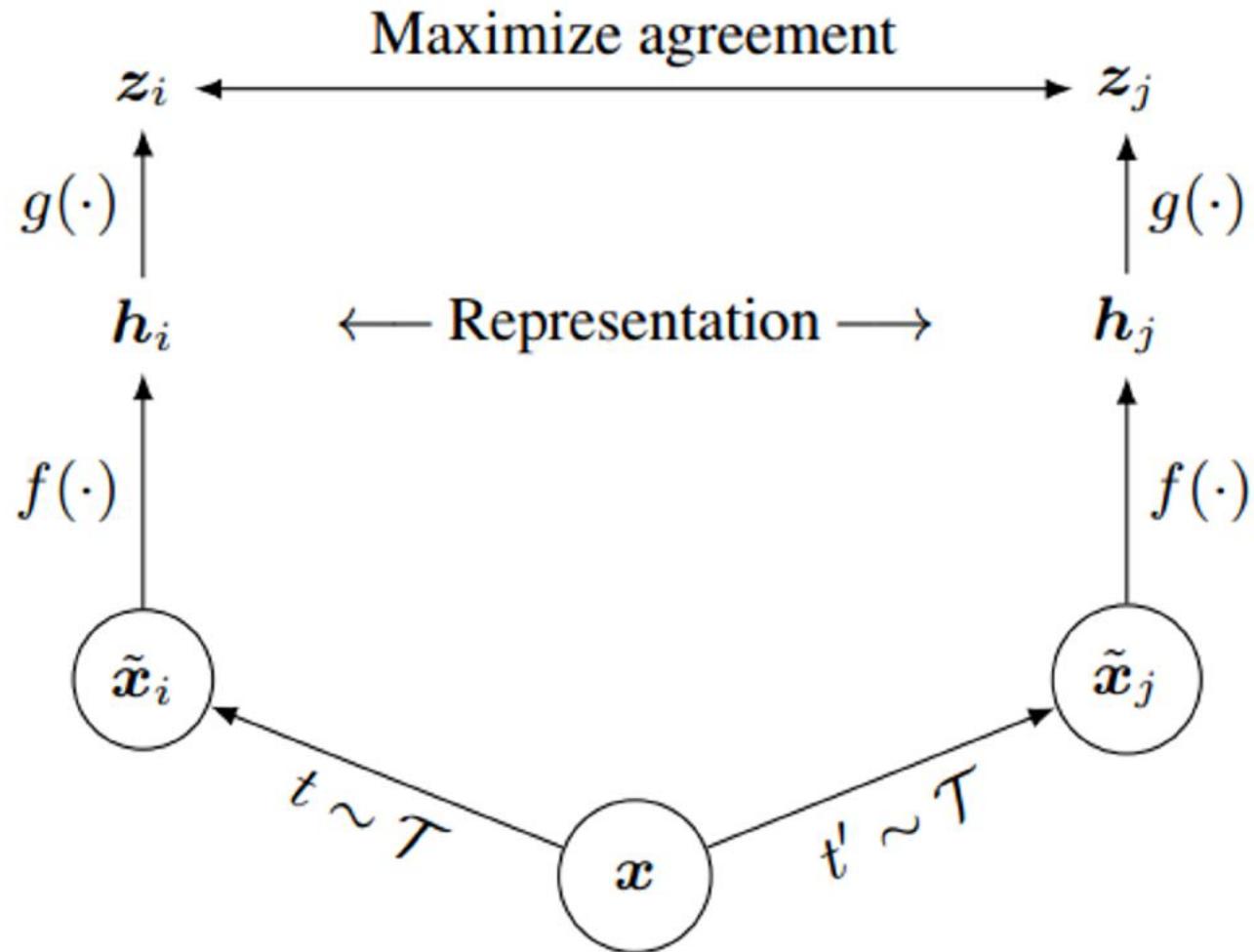


SimCLR

A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen¹ Simon Kornblith¹ Mohammad Norouzi¹ Geoffrey Hinton¹

SimCLR

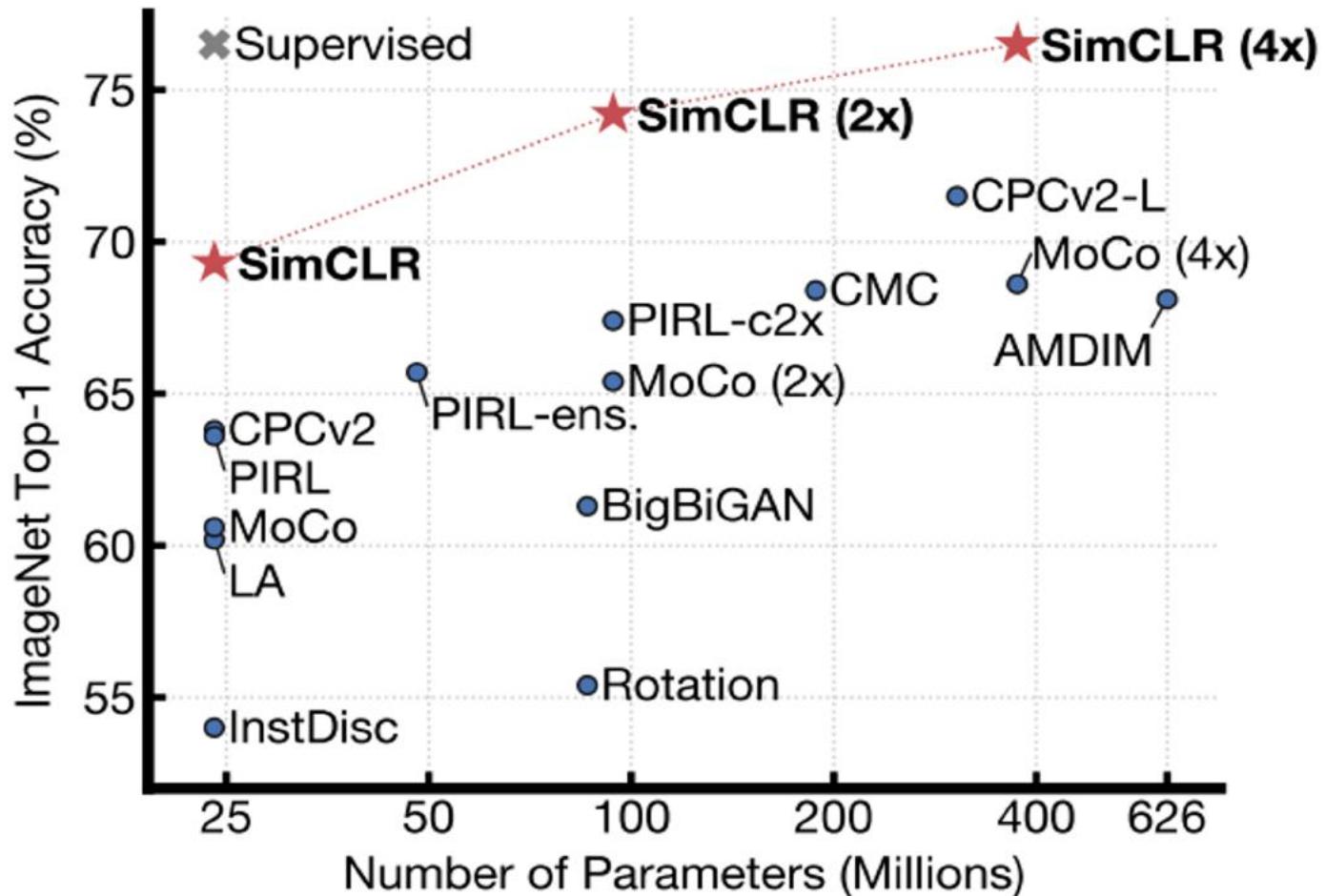


SimCLR

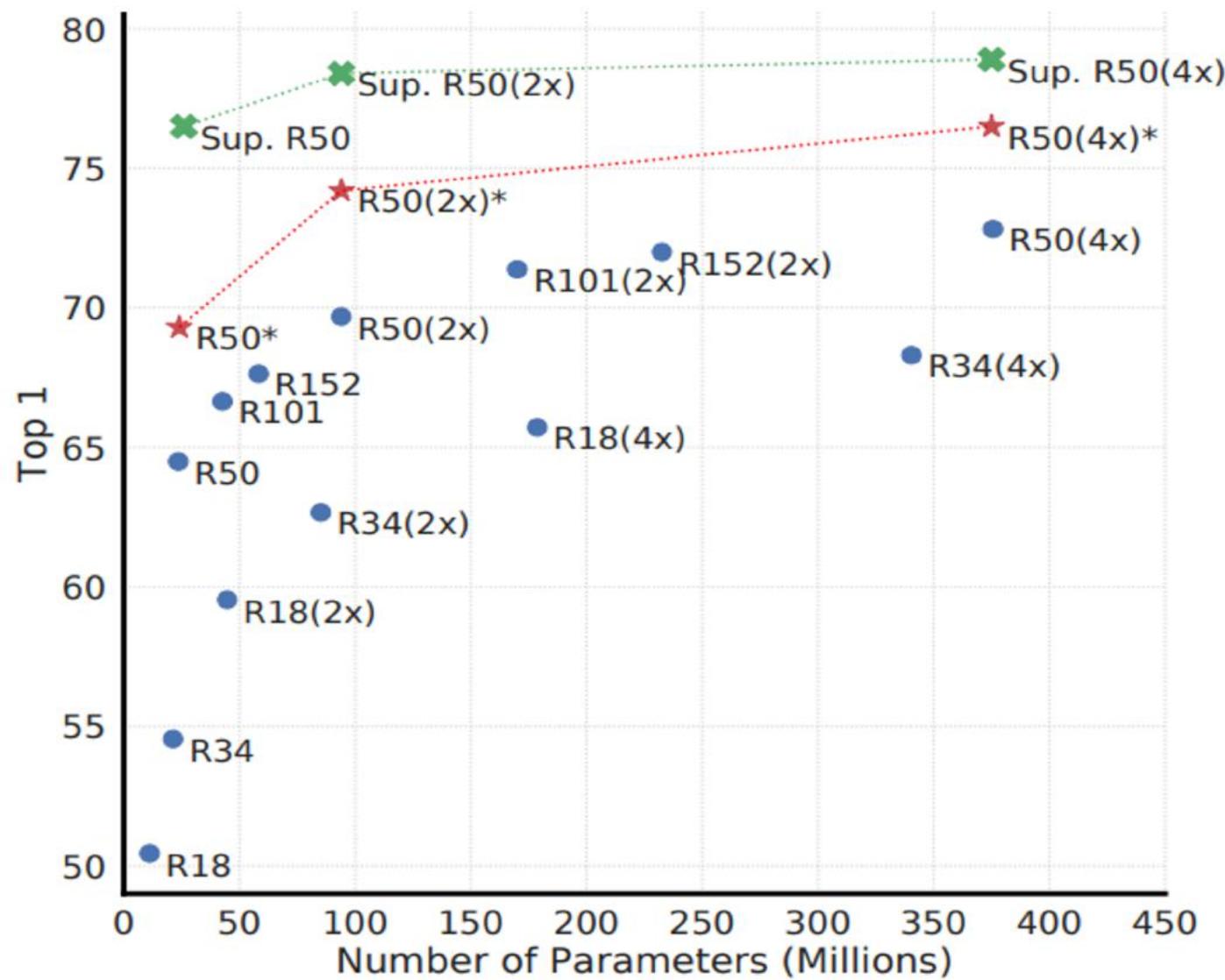
Algorithm 1 SimCLR's main learning algorithm.

input: batch size N , temperature τ , structure of f, g, \mathcal{T} .
for sampled minibatch $\{\mathbf{x}_k\}_{k=1}^N$ **do**
 for all $k \in \{1, \dots, N\}$ **do**
 draw two augmentation functions $t \sim \mathcal{T}, t' \sim \mathcal{T}$
 # the first augmentation
 $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$
 $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$ # representation
 $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$ # projection
 # the second augmentation
 $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$
 $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$ # representation
 $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$ # projection
 end for
 for all $i \in \{1, \dots, 2N\}$ and $j \in \{1, \dots, 2N\}$ **do**
 $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\tau \|\mathbf{z}_i\| \|\mathbf{z}_j\|)$ # 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

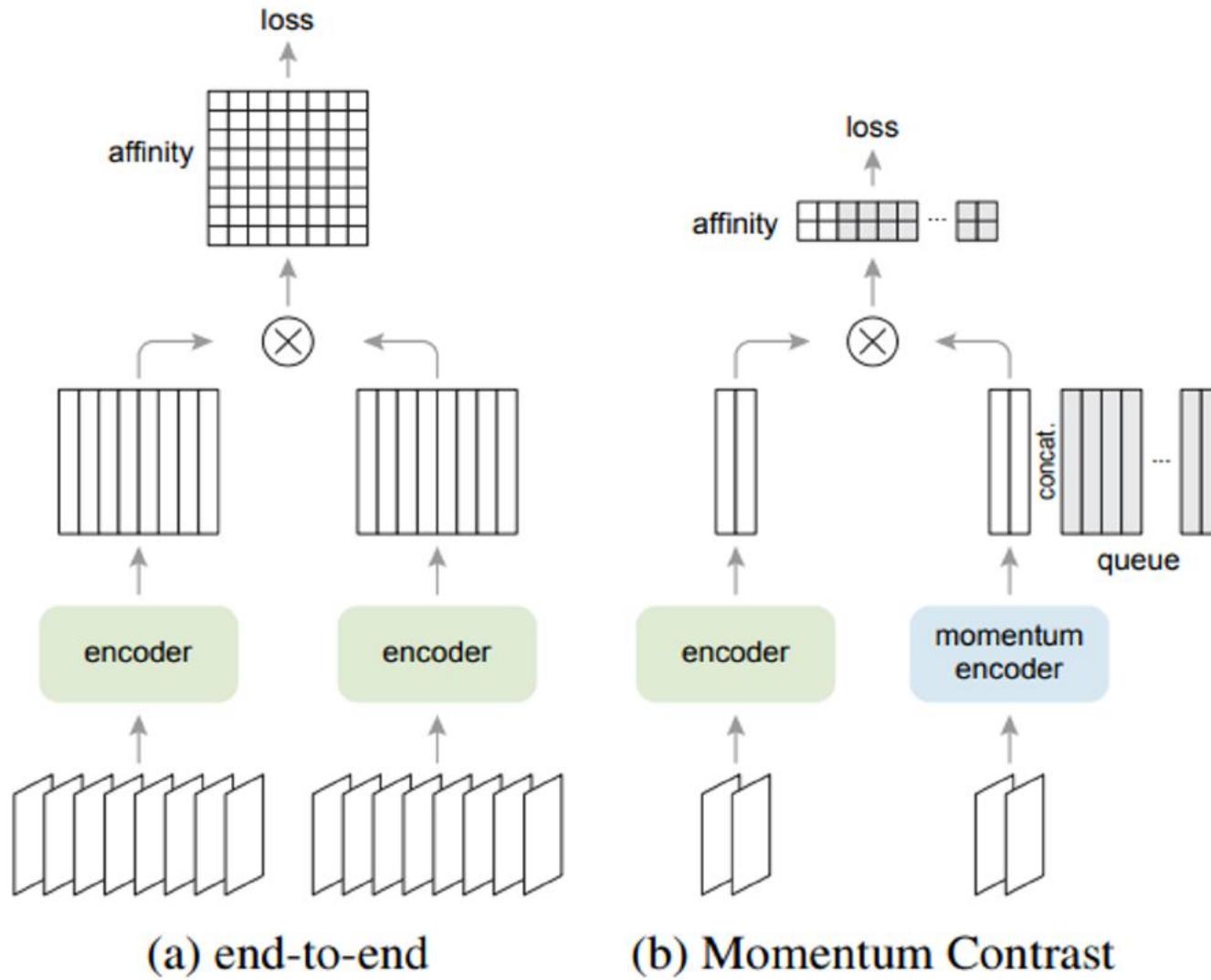
SimCLR



SimCLR



MoCov2 vs SimCLR



MoCov2 vs SimCLR

case	unsup. pre-train				ImageNet acc.	VOC detection		
	MLP	aug+	cos	epochs		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	MLP	aug+	unsup. pre-train			ImageNet acc.
			cos	epochs	batch	
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
<i>results of longer unsupervised training follow:</i>						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	71.1

Bootstrap Your Own Latent A New Approach to Self-Supervised Learning

Jean-Bastien Grill^{*1} Florian Strub^{*1} Florent Altché^{*1} Corentin Tallec^{*1} Pierre H. Richemond^{*1,2}
Elena Buchatskaya¹ Carl Doersch¹ Bernardo Avila Pires¹ Zhaohan Daniel Guo¹
Mohammad Gheshlaghi Azar¹ Bilal Piot¹ Koray Kavukcuoglu¹ Rémi Munos¹ Michal Valko¹

¹DeepMind ²Imperial College

[jbgrill,fstrub,altche,corentint,richemond]@google.com

Abstract

We introduce **Bootstrap Your Own Latent** (BYOL), a new approach to self-supervised image representation learning. BYOL relies on two neural networks, referred to as *online* and *target* networks, that interact and learn from each other. From an augmented view of an image, we train the online network to predict the target network representation of the same image under a different augmented view. At the same time, we update the target network with a slow-moving average of the online network. While state-of-the art methods intrinsically rely on negative pairs, BYOL achieves a new state of the art *without them*. BYOL reaches 74.3% top-1 classification accuracy on ImageNet using the standard linear evaluation protocol with a ResNet-50 architecture and 79.6% with a larger ResNet. We show that BYOL performs on par or better than the current state of the art on both transfer and semi-supervised benchmarks.

BYOL

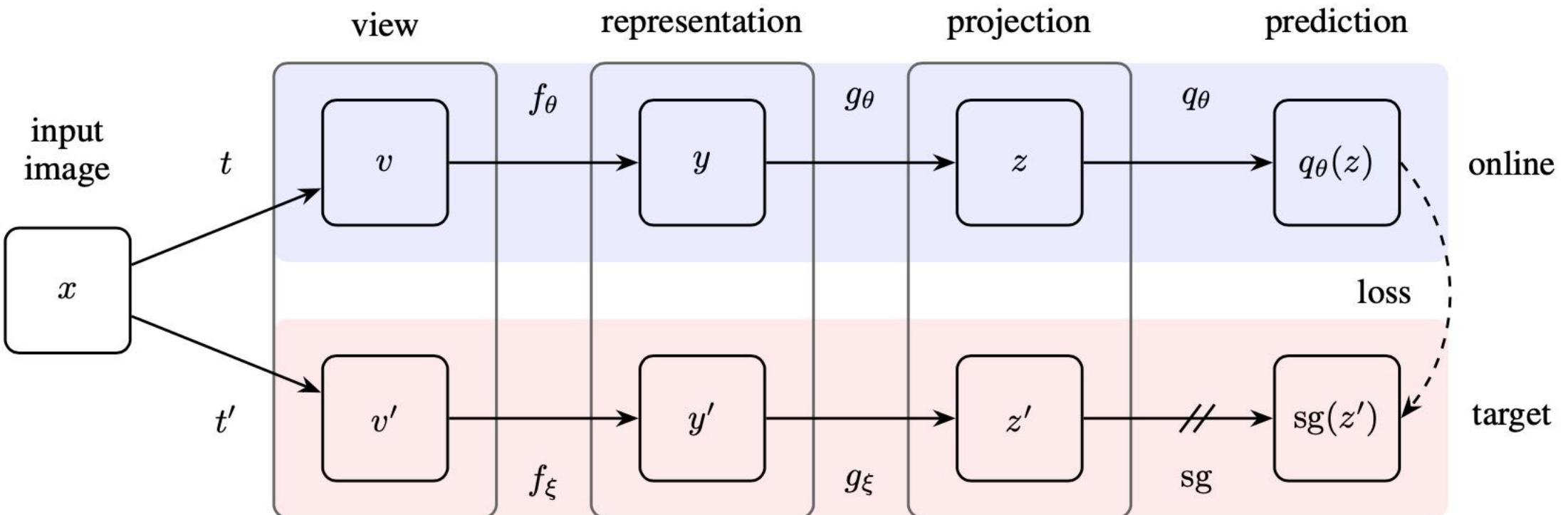
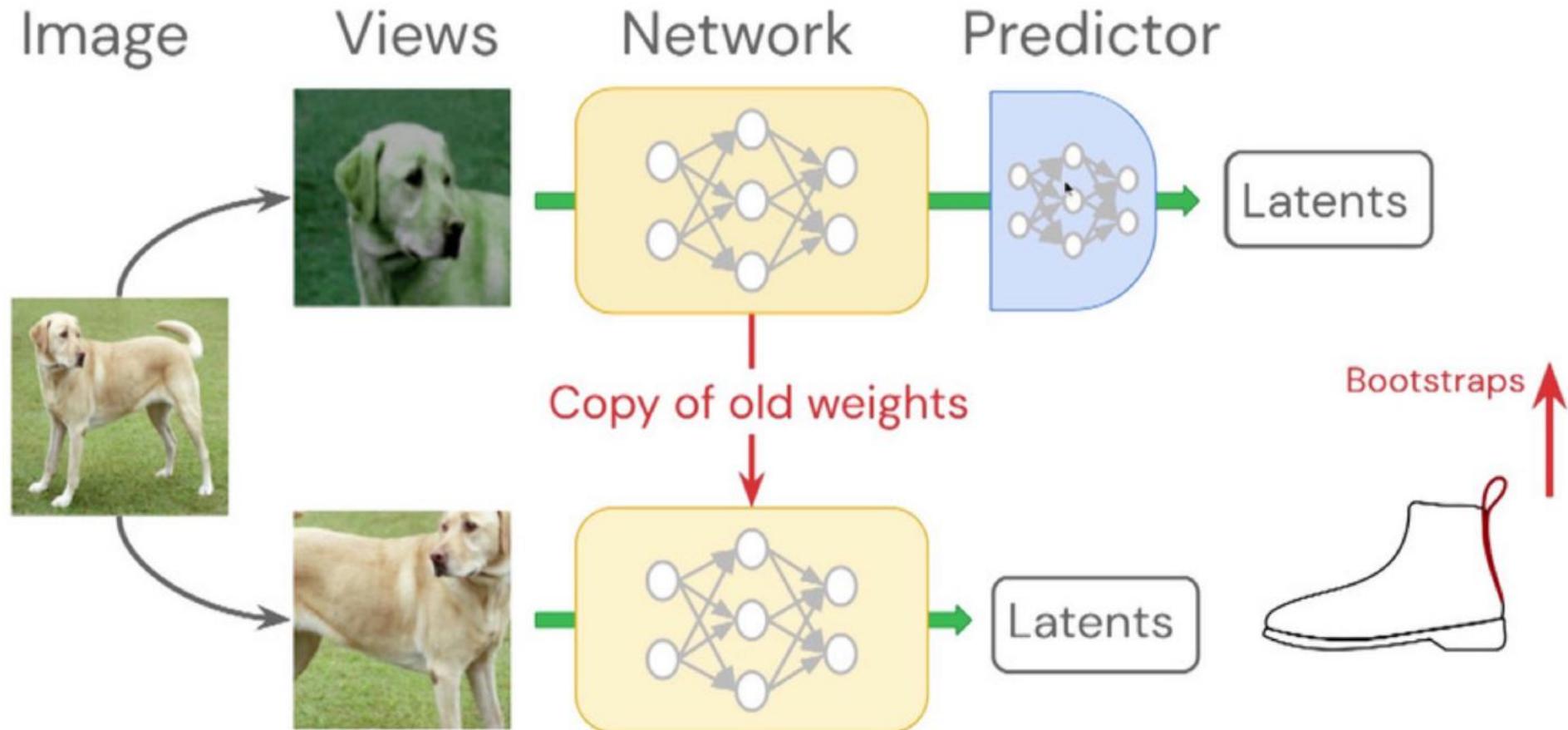


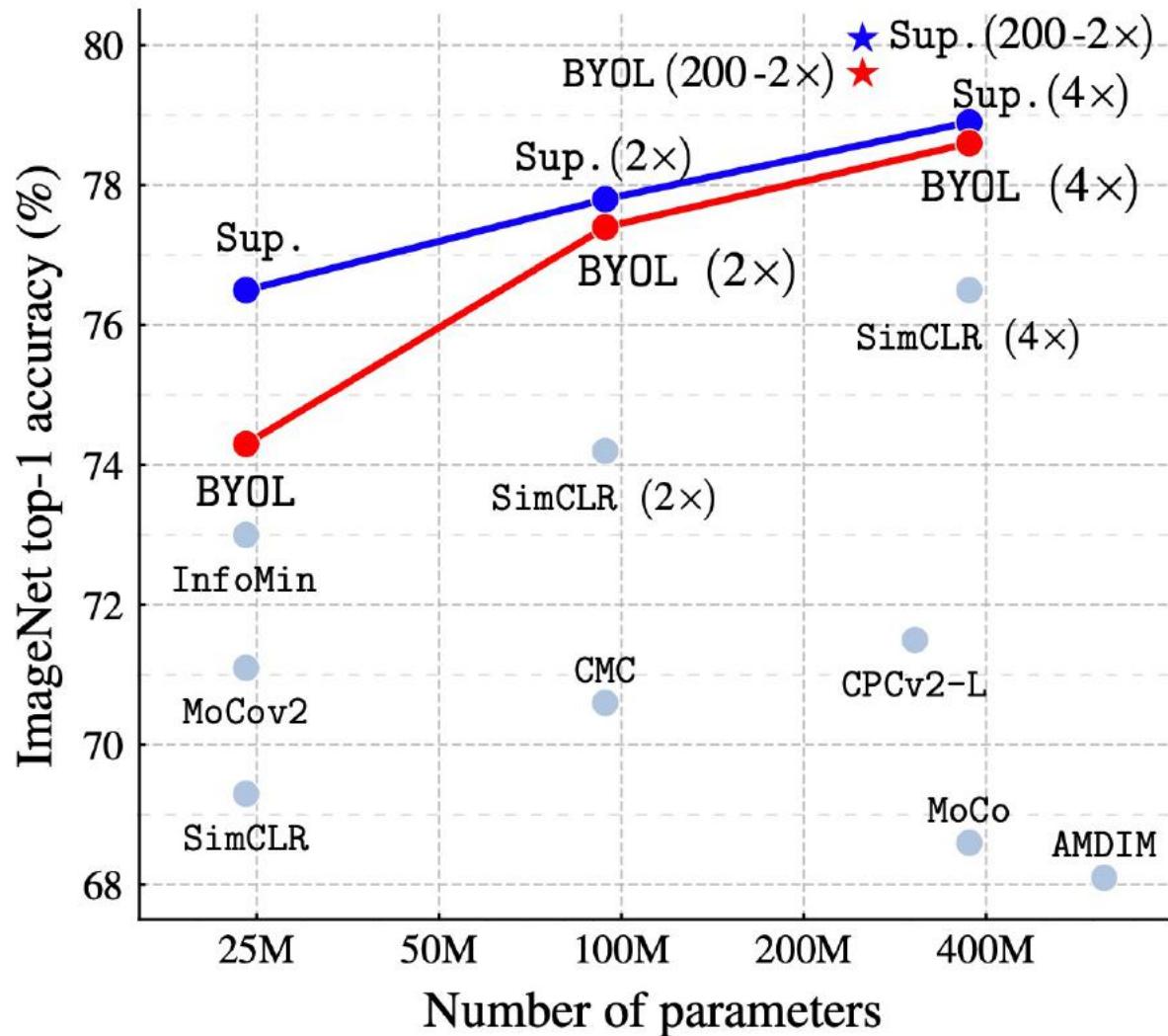
Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z)$ and $sg(z')$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded and y is used as the image representation.

BYOL



- Does not use negative examples!

BYOL



BYOL

Method	Top-1	Top-5
Local Agg.	60.2	-
PIRL [35]	63.6	-
CPC v2 [32]	63.8	85.3
CMC [11]	66.2	87.0
SimCLR [8]	69.3	89.0
MoCo v2 [37]	71.1	-
InfoMin Aug. [12]	73.0	91.1
BYOL (ours)	74.3	91.6

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1	Top-5
SimCLR [8]	ResNet-50 (2×)	94M	74.2	92.0
CMC [11]	ResNet-50 (2×)	94M	70.6	89.7
BYOL (ours)	ResNet-50 (2×)	94M	77.4	93.6
CPC v2 [32]	ResNet-161	305M	71.5	90.1
MoCo [9]	ResNet-50 (4×)	375M	68.6	-
SimCLR [8]	ResNet-50 (4×)	375M	76.5	93.2
BYOL (ours)	ResNet-50 (4×)	375M	78.6	94.2
BYOL (ours)	ResNet-200 (2×)	250M	79.6	94.8

(b) Other ResNet encoder architectures.

Table 1: Top-1 and top-5 accuracies (in %) under linear evaluation on ImageNet.

Next lecture:
Pretraining Language Models