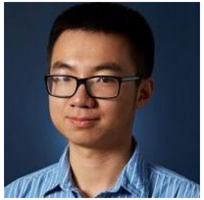
CS294-158 Deep Unsupervised Learning

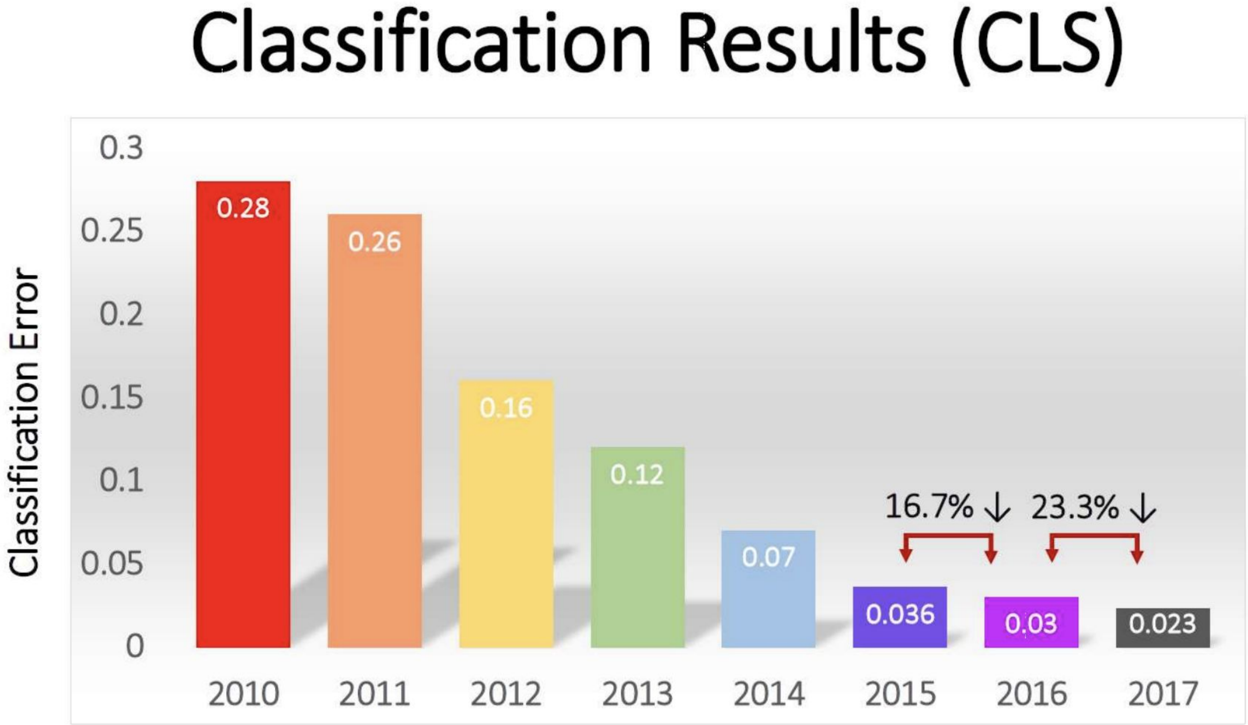
Lecture 6b+7: Non-Generative Representation Learning

Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas

UC Berkeley

Motivation



Slide: Andrew Zisserman

Motivation

- Features from pre-trained Imagenet can be used for other visual tasks with smaller datasets such as detection, segmentation, action recognition, fine-grained classification
- If you can construct a large dataset with good quality labels and train a large deep neural network, success is more or less guaranteed - Ilya Sutskever, NeurIPS 2014
- Goal of non-generative representation learning is to come up with unsupervised / self-supervised objectives that can be optimized on a lot of unlabeled data to produce similar quality features

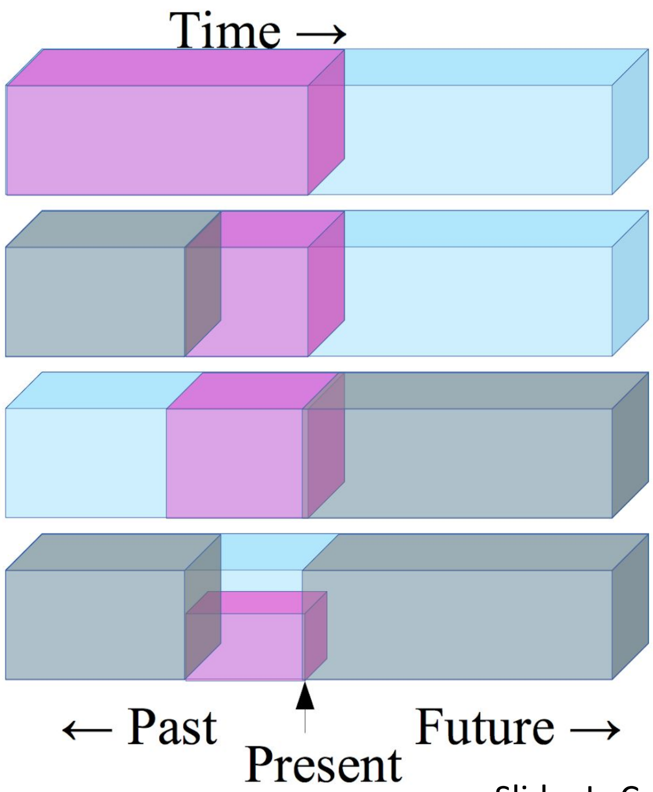
Motivation

- **"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.



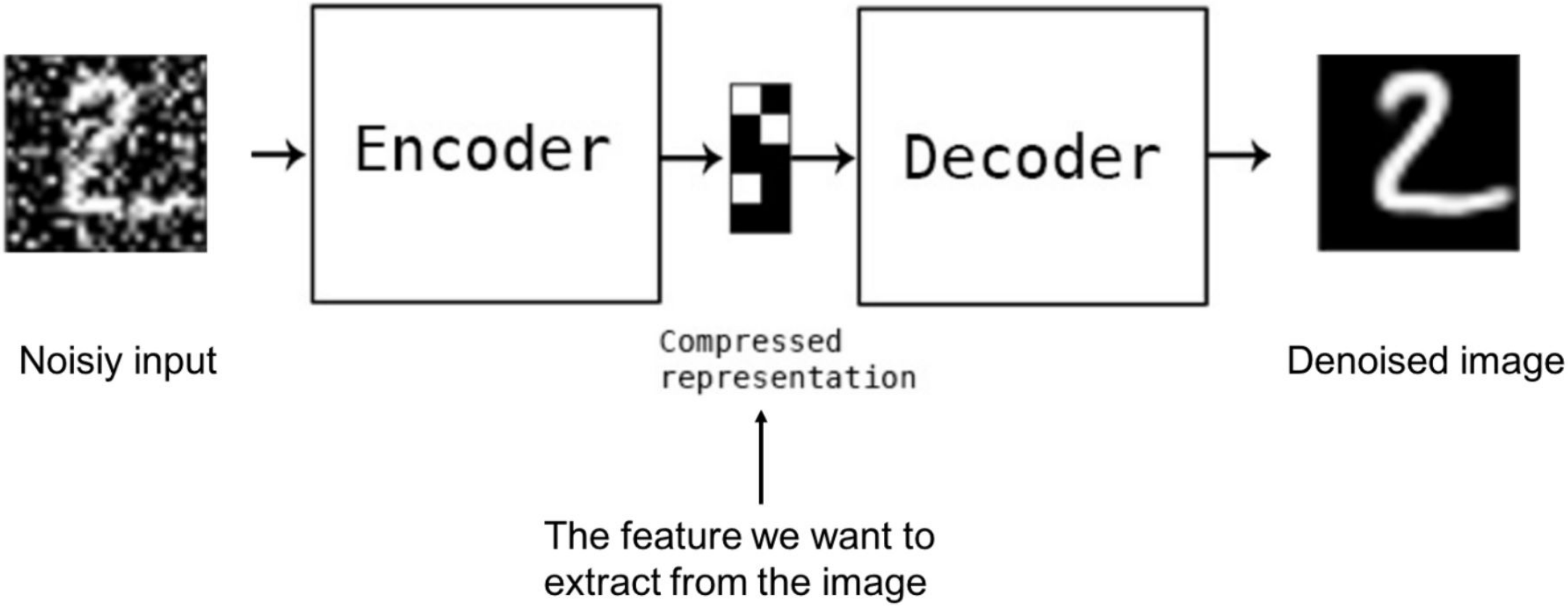
Slide: LeCun

Cognitive Principles

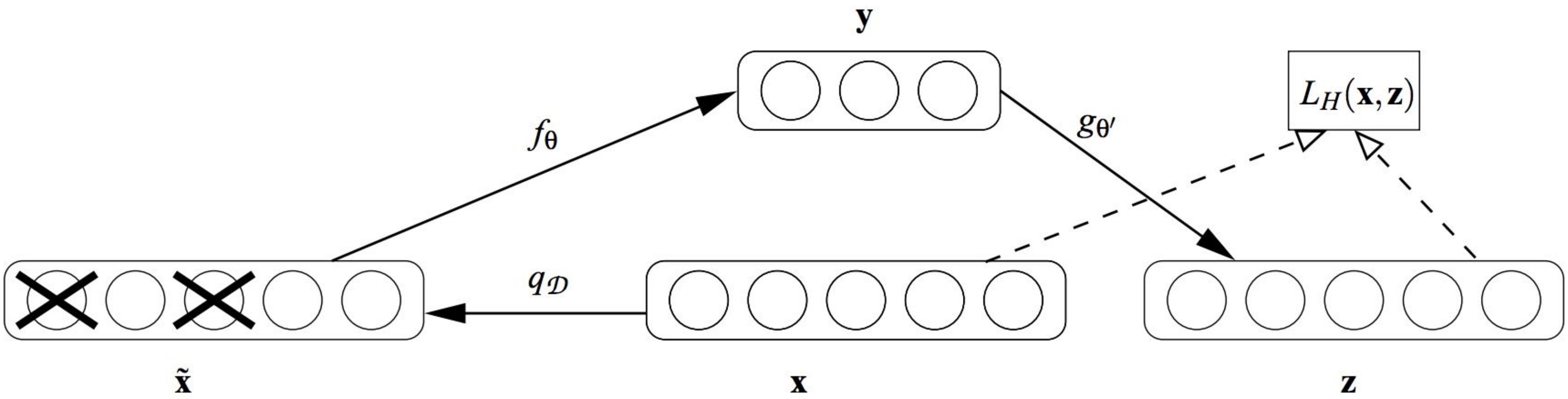
- Reconstruct from a corrupted version
  - Denoising Autoencoder
- Reconstruct missing parts
  - Image in-painting
- Predict one view of the data from another
  - Colorization, Split-Brain Autoencoders
- Visual common sense tasks
  - Relative patch prediction, Jigsaw puzzles, Rotation, Tracking from colorization
- Predict related / neighboring elements
  - word2vec, contrastive predictive coding, BERT

Next week

Recover from corrupted version



Denoising Autoencoder



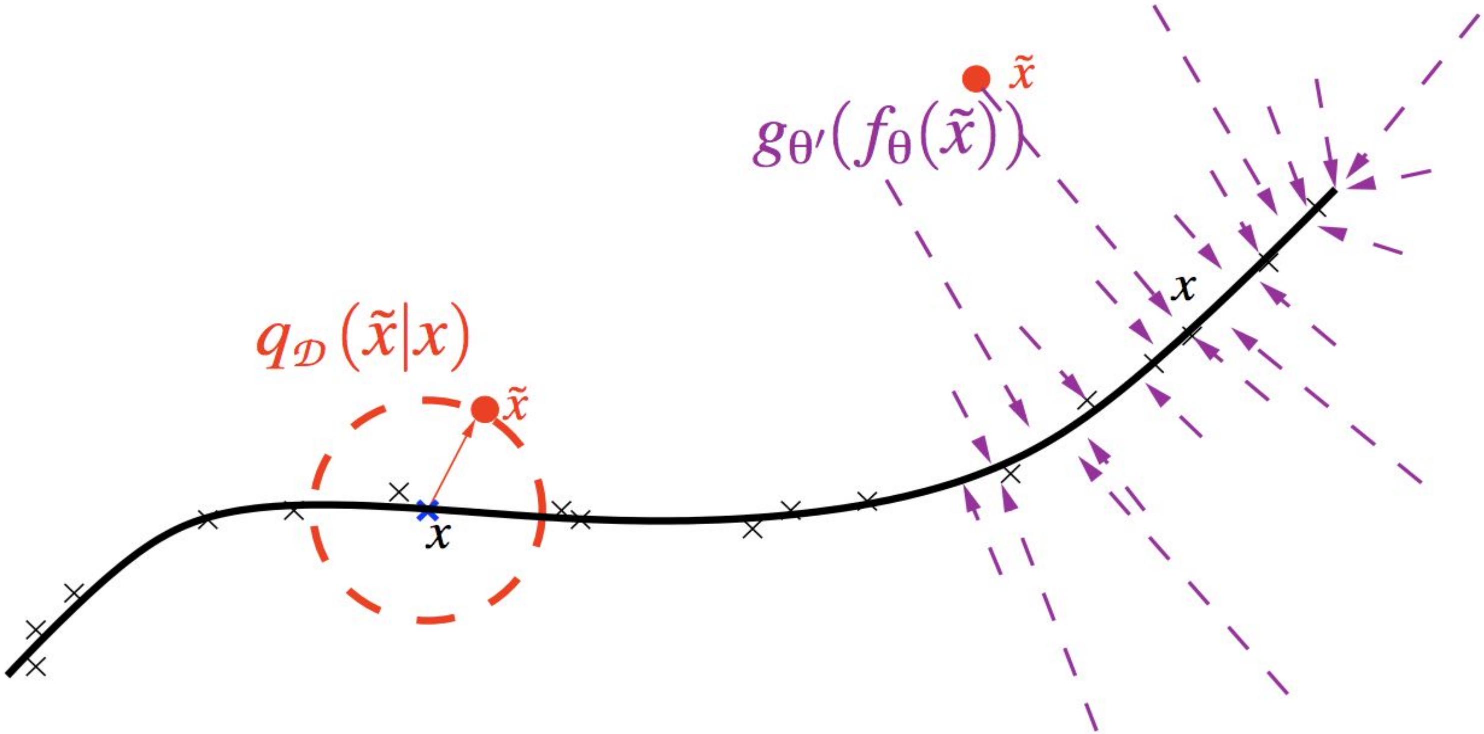
Vincent et al 2010

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 ν of the elements of $\mathbf{x}$ (chosen at random for each example) is forced to 0;
- *Salt-and-pepper noise* (SP): a fraction ν of the elements of $\mathbf{x}$ (chosen at random for each example) is set to their minimum or maximum possible value (typically 0 or 1) according to a fair coin flip.

Vincent et al 2010

Denoising Autoencoder



Vincent et al 2010

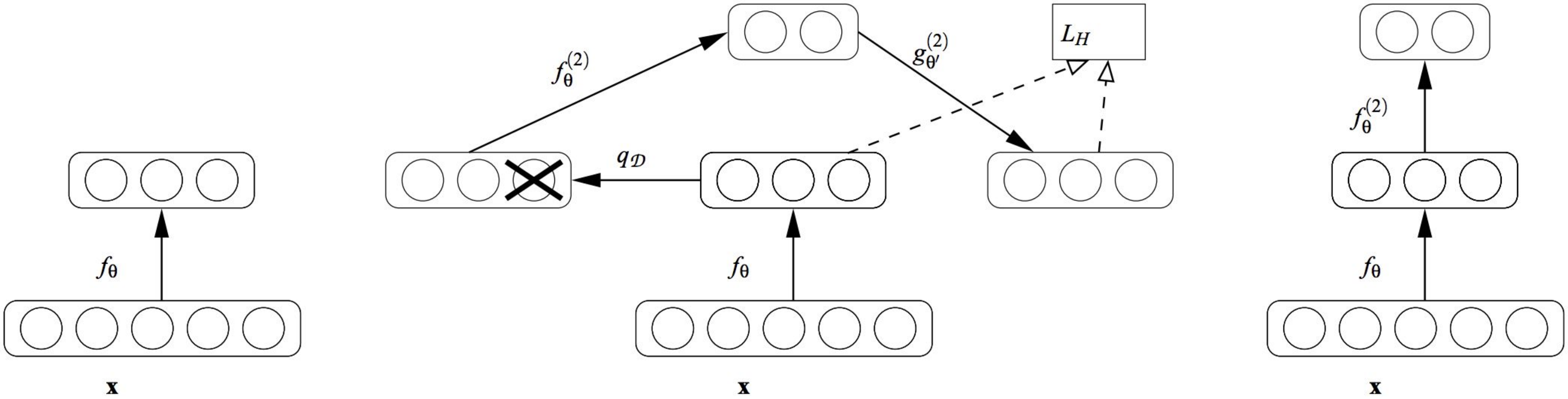
Emphasizing corrupted dimensions

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

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

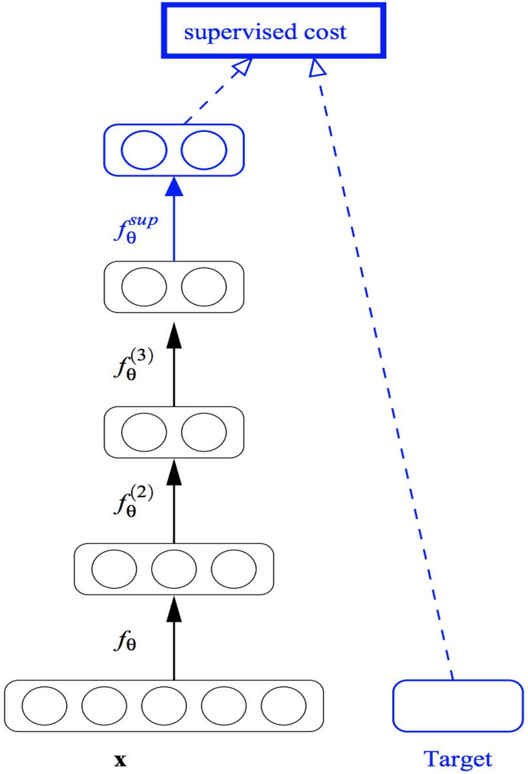
Vincent et al 2010

Stacked Denoising Autoencoder



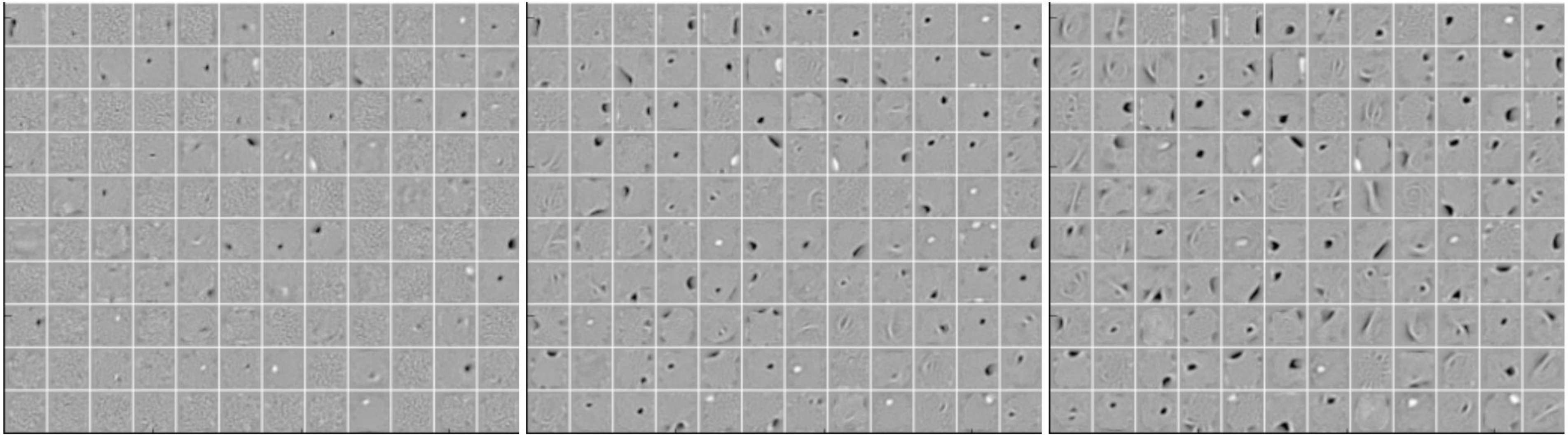
Vincent et al 2010

Denoising Autoencoder



Vincent et al 2010

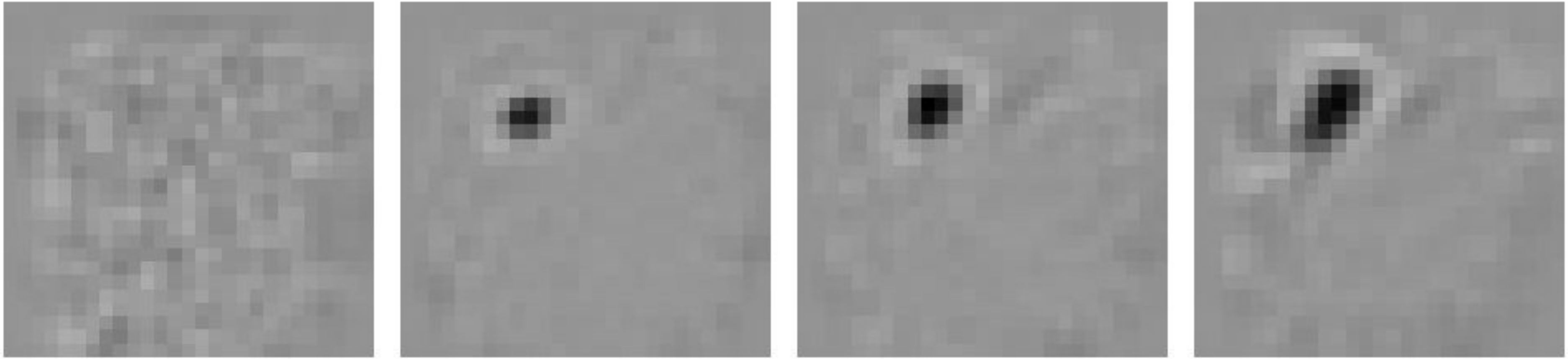
Denoising Autoencoder



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

Vincent et al 2010

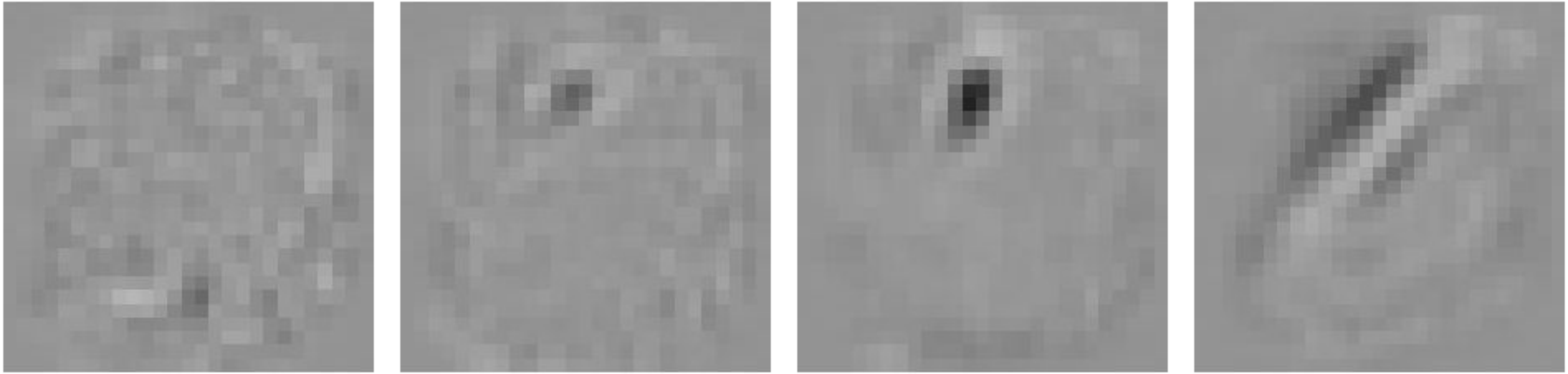
Denoising Autoencoder



(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

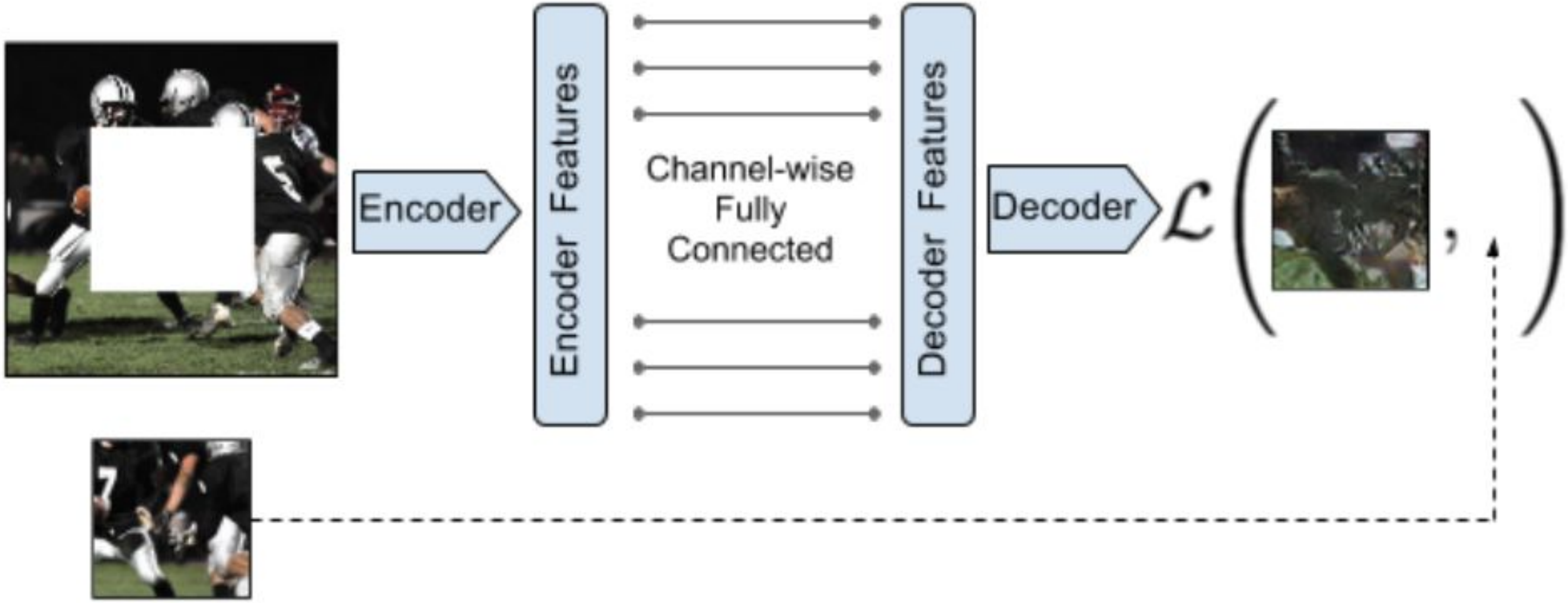
Dataset	$\mathbf{SVM}_{rbf}$	$\mathbf{SVM}_{poly}$	**DBN-1**	**SAA-3**	**DBN-3**	**SdA-3** (ν)
basic	**3.03±0.15**	3.69±0.17	3.94±0.17	3.46±0.16	3.11±0.15	**2.80±0.14** (10%)
rot	11.11±0.28	15.42±0.32	14.69±0.31	**10.30±0.27**	**10.30±0.27**	**10.29±0.27** (10%)
bg-rand	14.58±0.31	16.62±0.33	9.80±0.26	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%)

Vincent et al 2010

Predict missing pieces

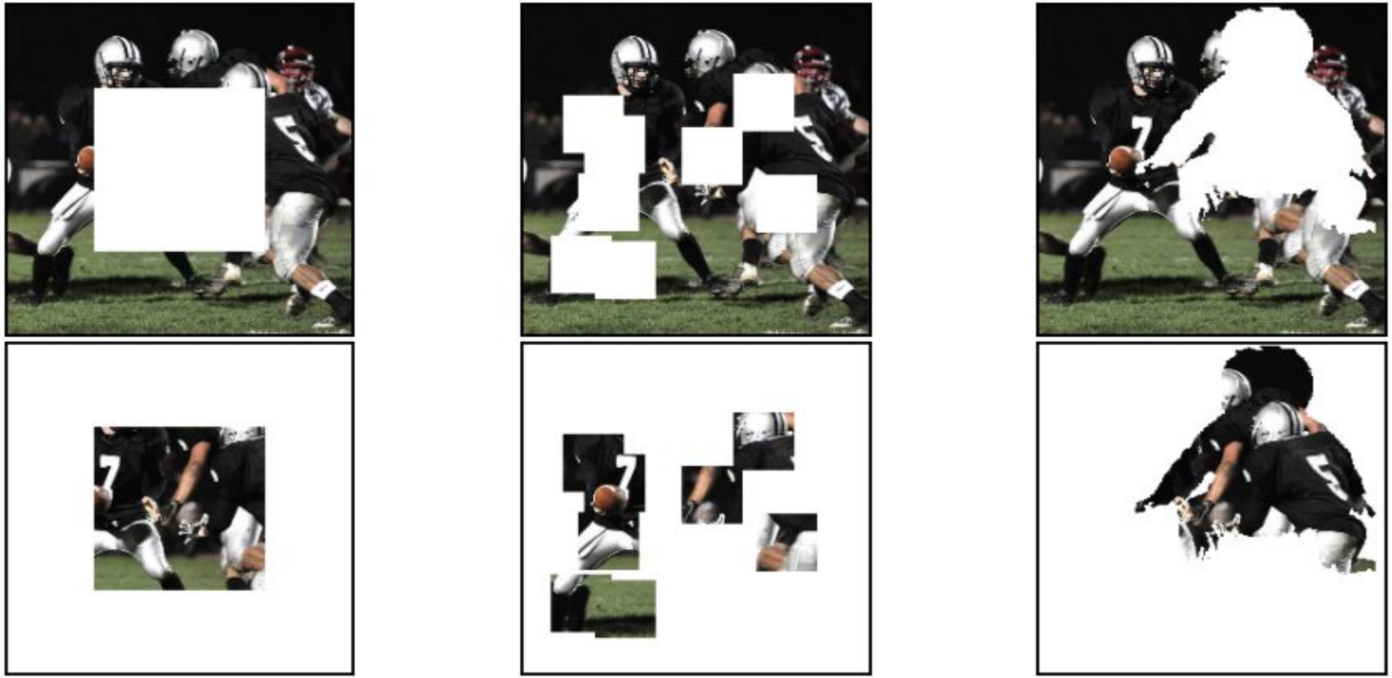
Pathak et al 2016

Context Encoders



Pathak et al 2016

Context Encoders



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

Pathak et al 2016

Context Encoders

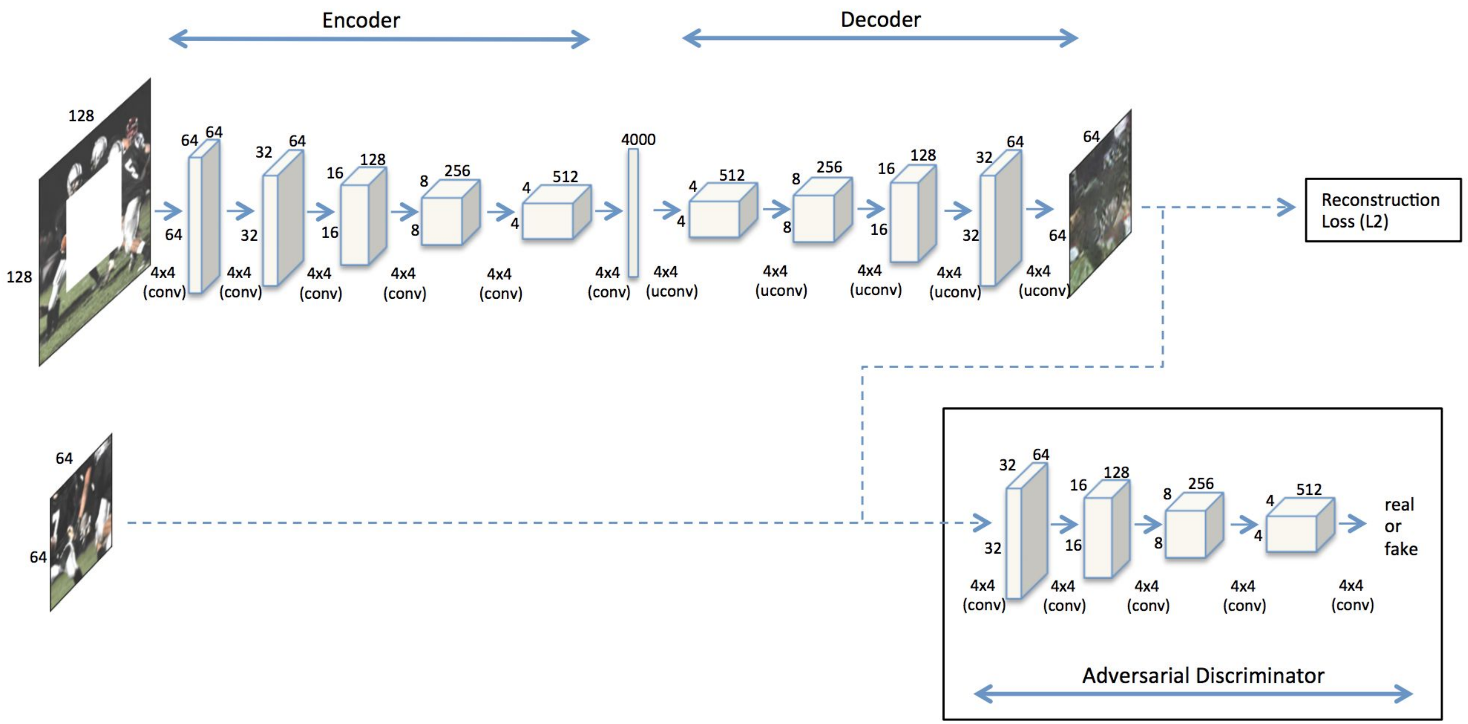
$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2$$

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

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

Pathak et al 2016

Context Encoders



Pathak et al 2016

Context Encoders

Input Image | L2 Loss | Adversarial Loss | Joint Loss

Pathak et al 2016

Context Encoders

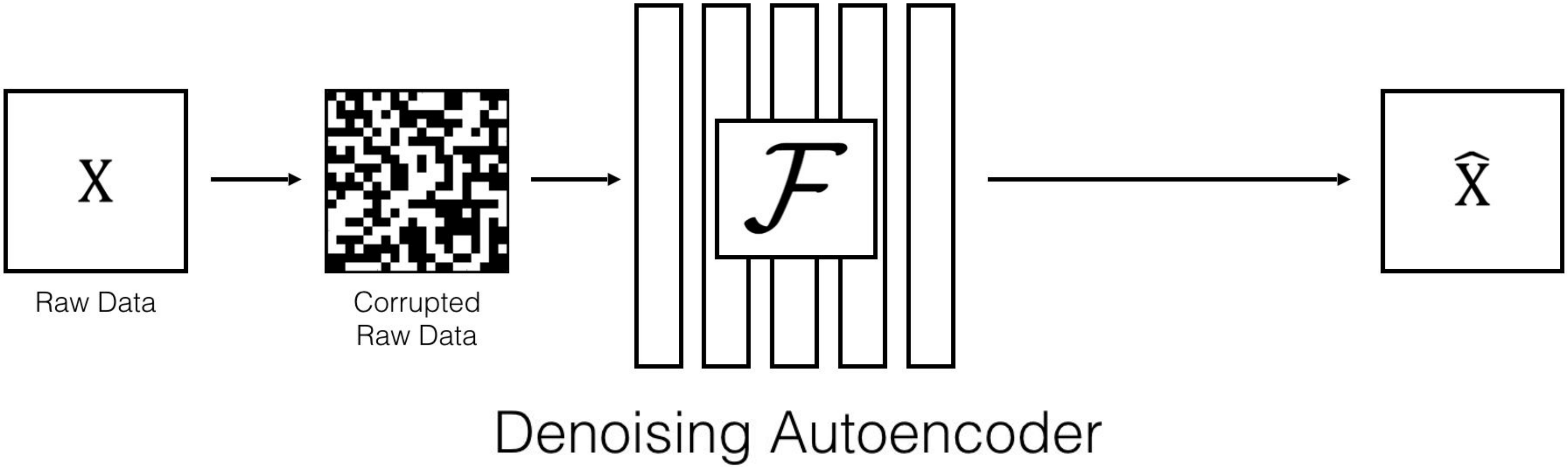
Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	**78.2%**	**56.8%**	**48.0%**
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal *et al.* [1]	egomotion	10 hours	52.9%	41.8%	-
Doersch *et al.* [7]	context	4 weeks	55.3%	**46.6%**	-
Wang *et al.* [39]	motion	1 week	**58.4%**	44.0%	-
Ours	context	14 hours	56.5%	44.5%	**29.7%**

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

Pathak et al 2016

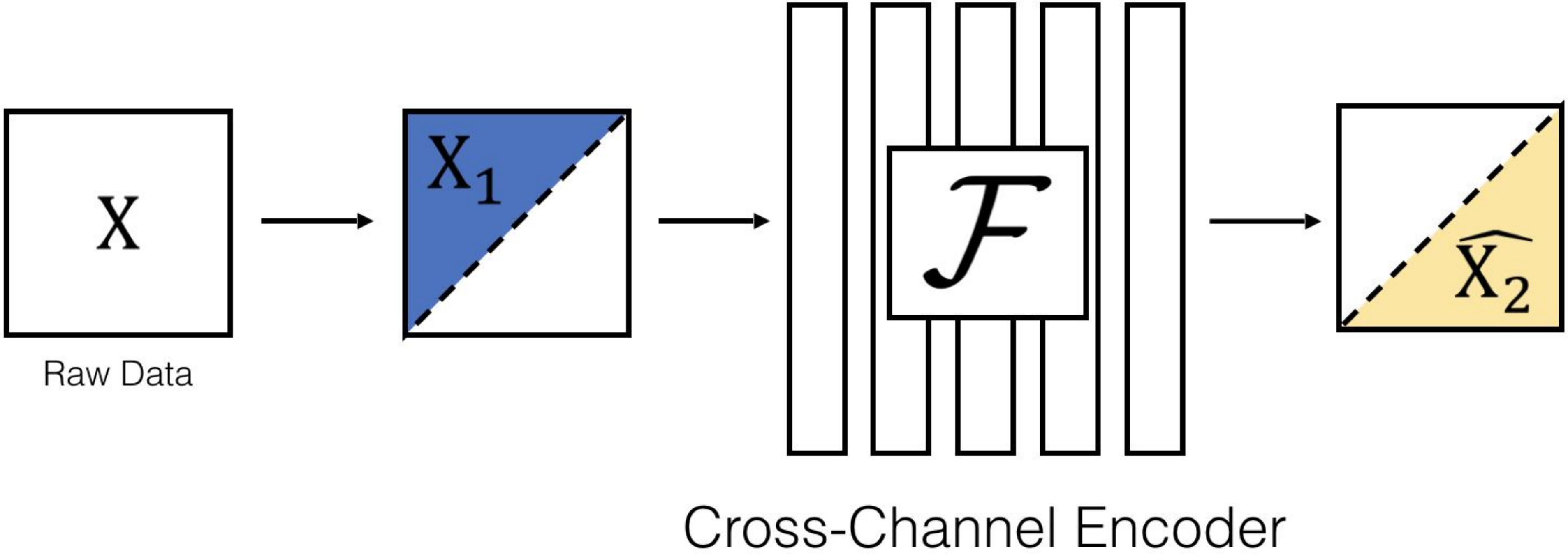
Important to note that these are based on fine-tuning and not freeze + linear features.

Predicting one view from another



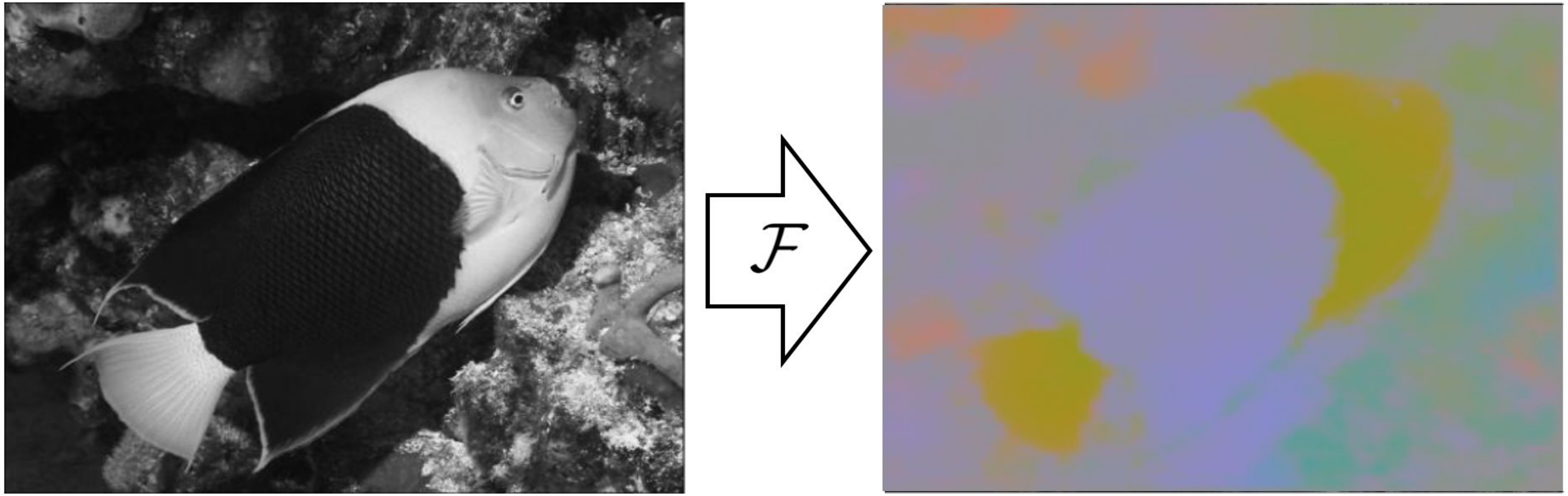
Slide: Richard Zhang

Predicting one view from another



Slide: Richard Zhang

Predicting one view from another

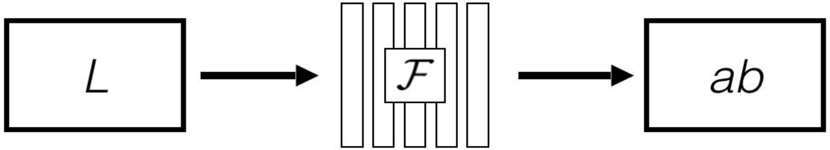


Grayscale image: *L* channel

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

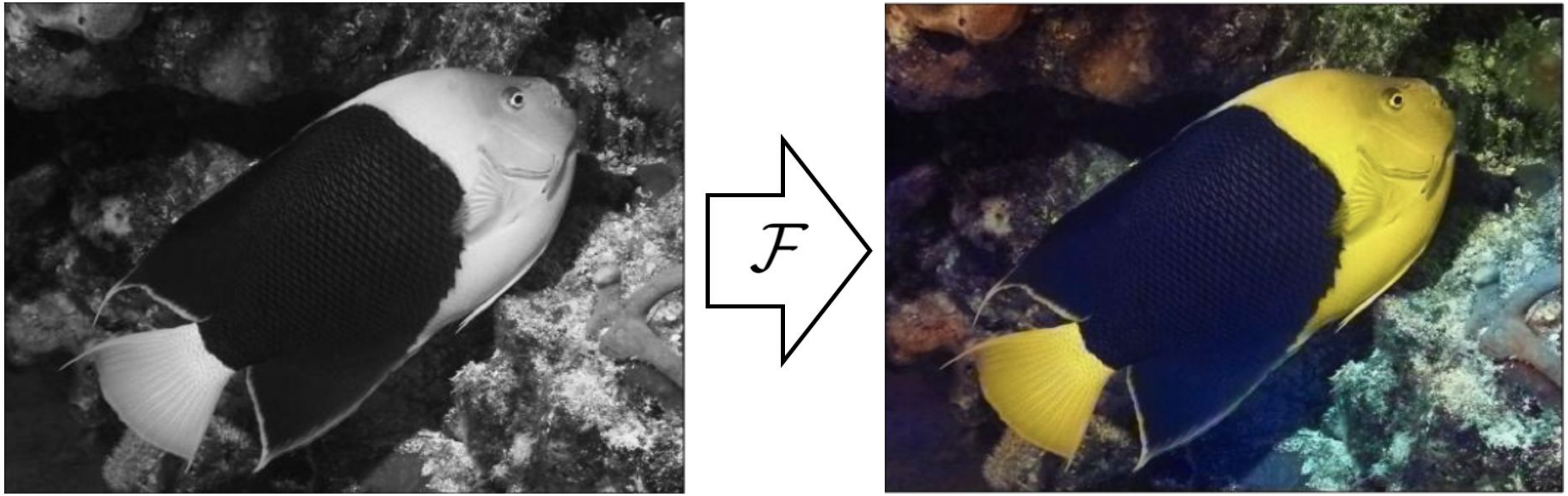
Color information: *ab* channels

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



Slide: Richard Zhang

Predicting one view from another

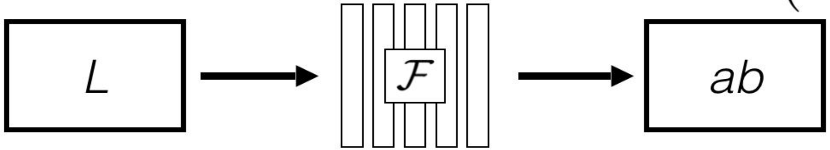


Grayscale image: *L* channel

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

Concatenate (*L*,*ab*) channels

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



Slide: Richard Zhang

Predicting one view from another

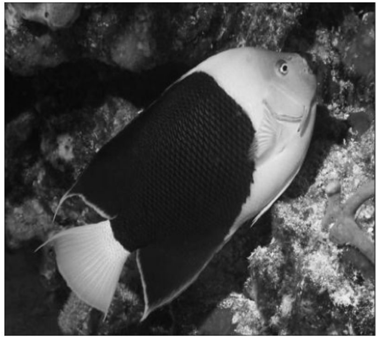
Ground Truth

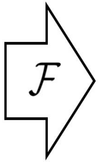
L2 regression

Pixelwise classification

Slide: Richard Zhang

Predicting one view from another



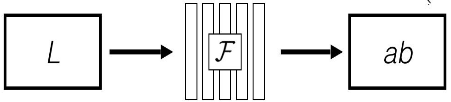


Grayscale image: *L* channel

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

Concatenate (*L*,*ab*) channels

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



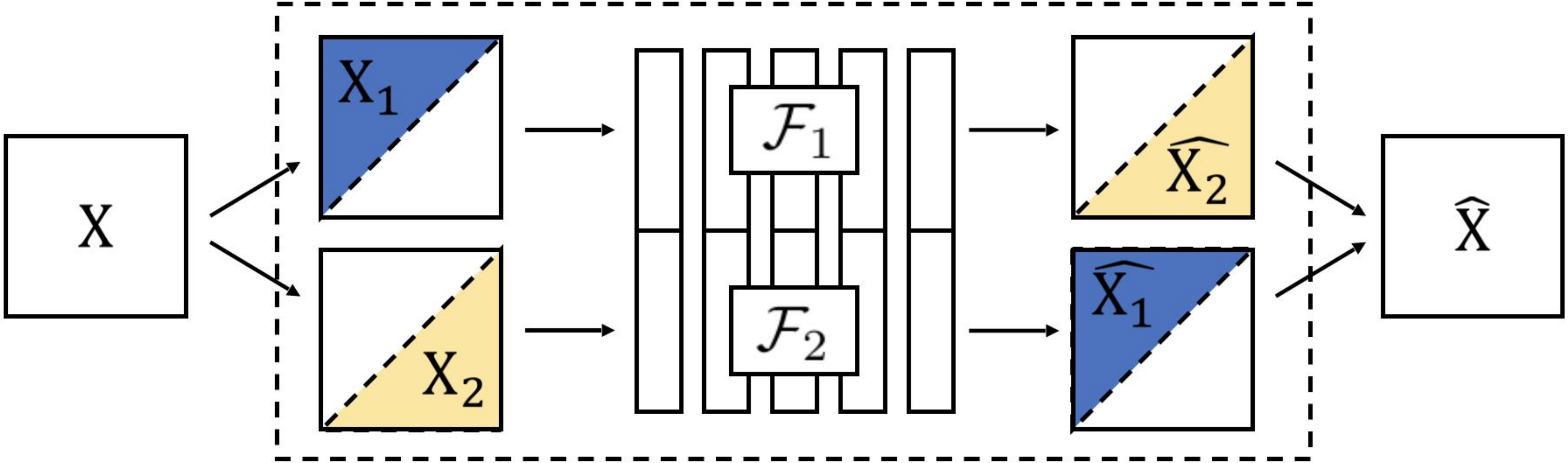
6

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

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

Slide: Richard Zhang

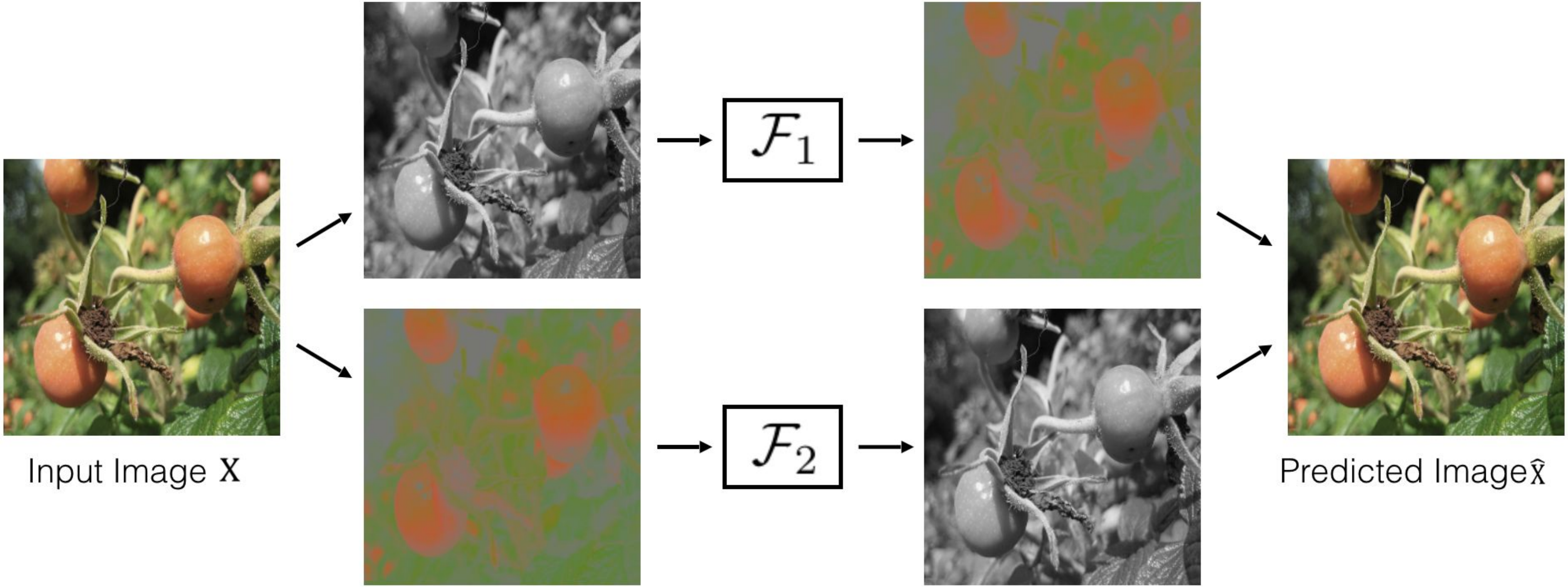
Predicting one view from another



Split-Brain Autoencoder

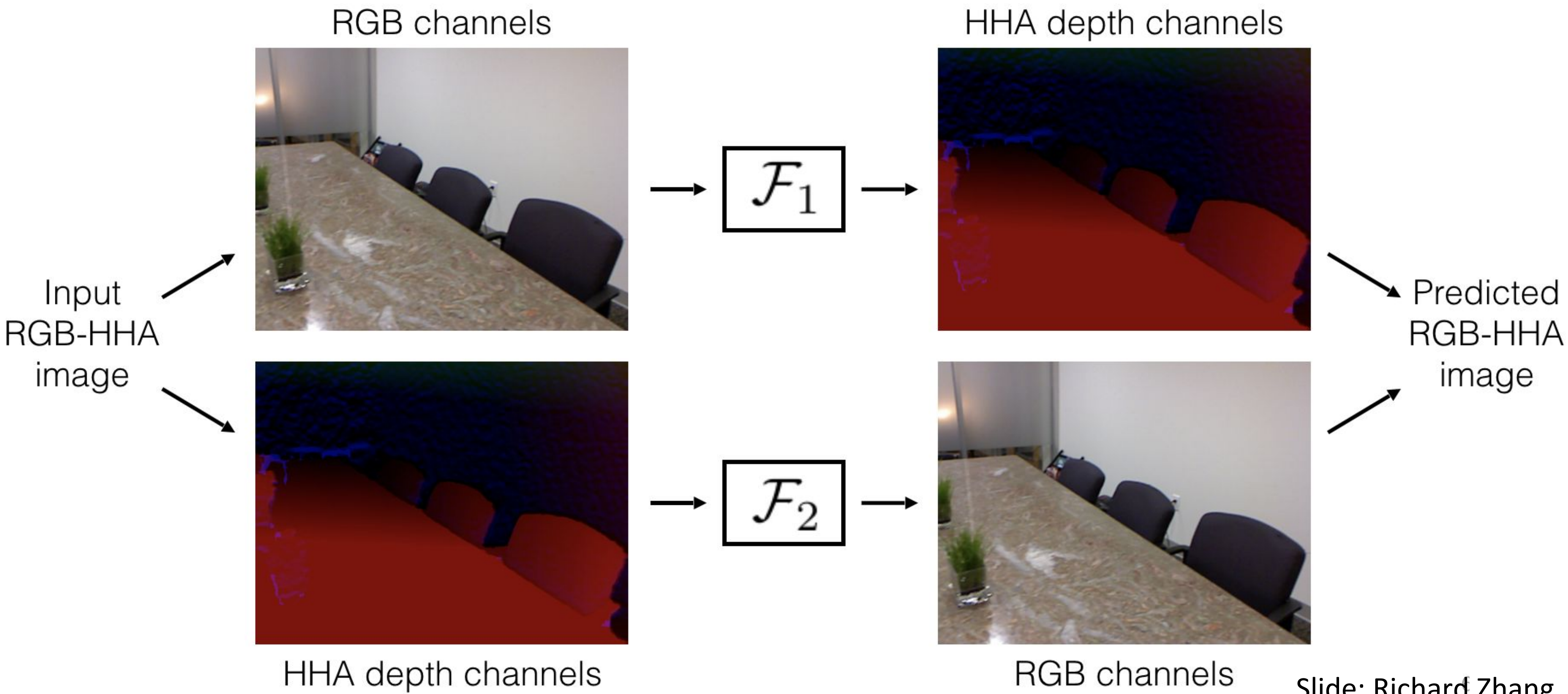
Slide: Richard Zhang

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]

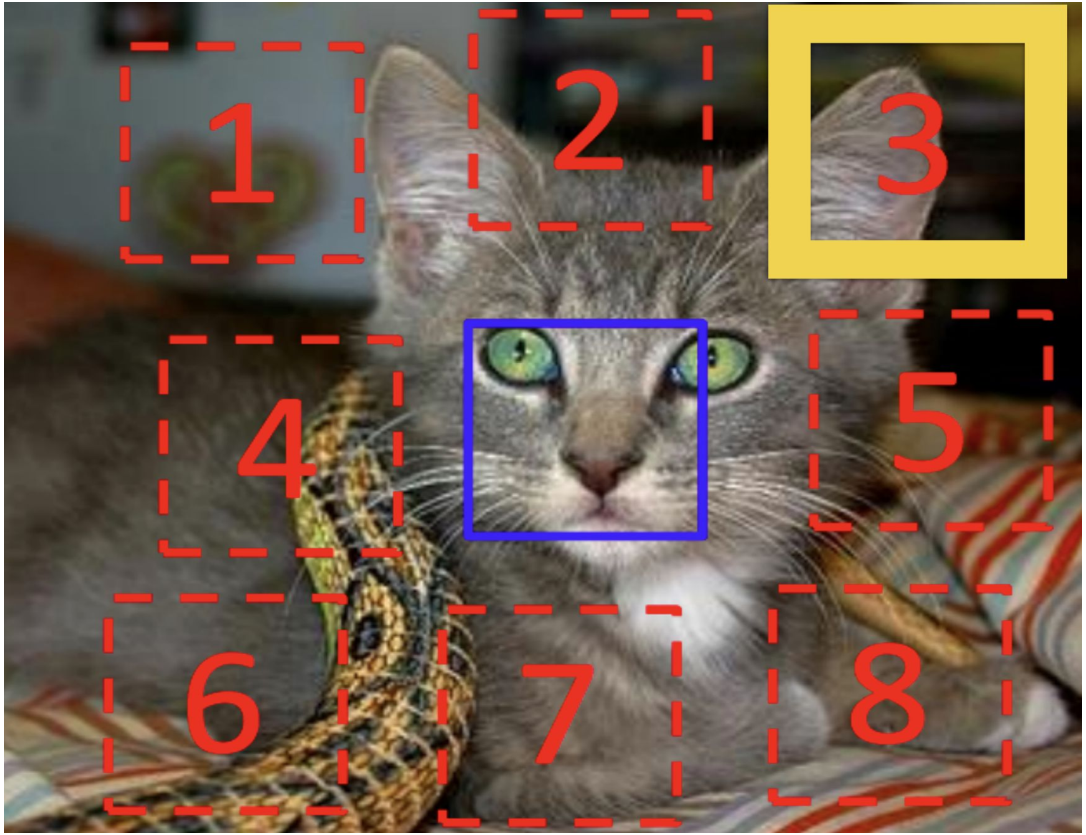
[1] School of Computer Science
Carnegie Mellon University

[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

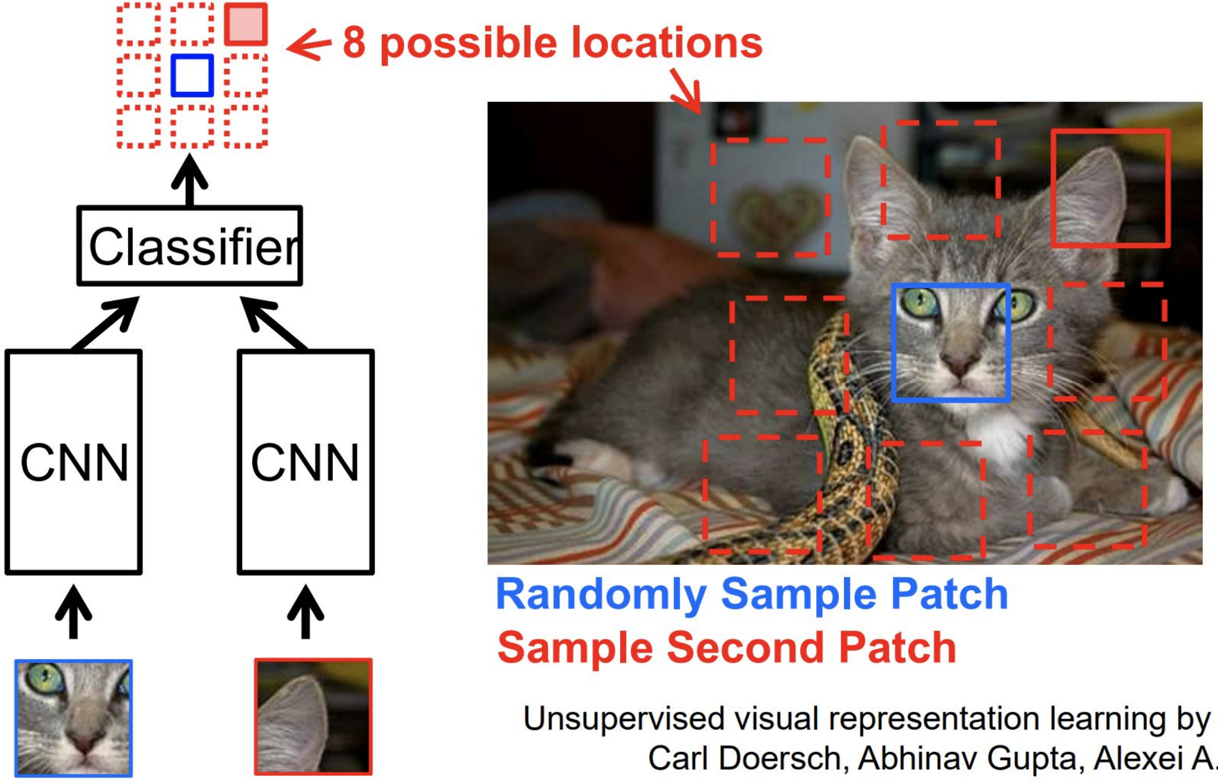
Relative Position of Image Patches



Doersch, Gupta, Efros

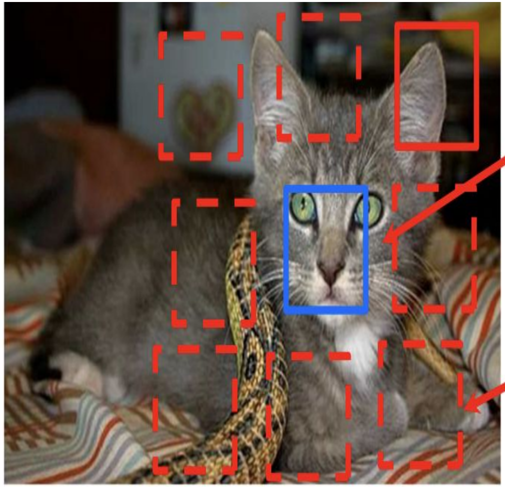
Slide: Zisserman

Relative Position of Image Patches



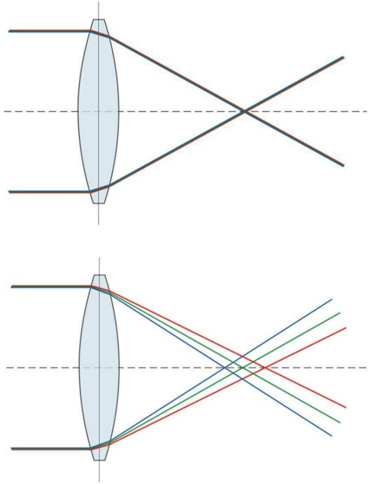
Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

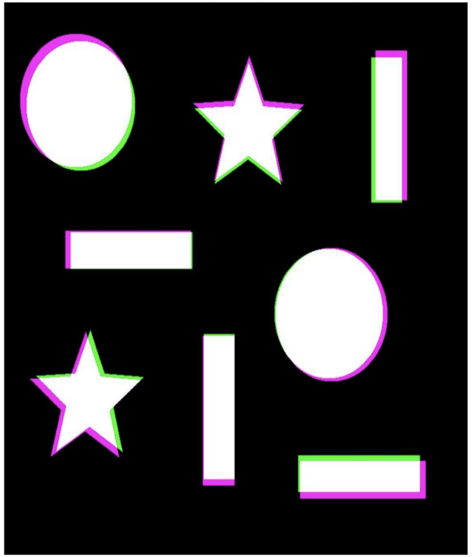
Relative Position of Image Patches



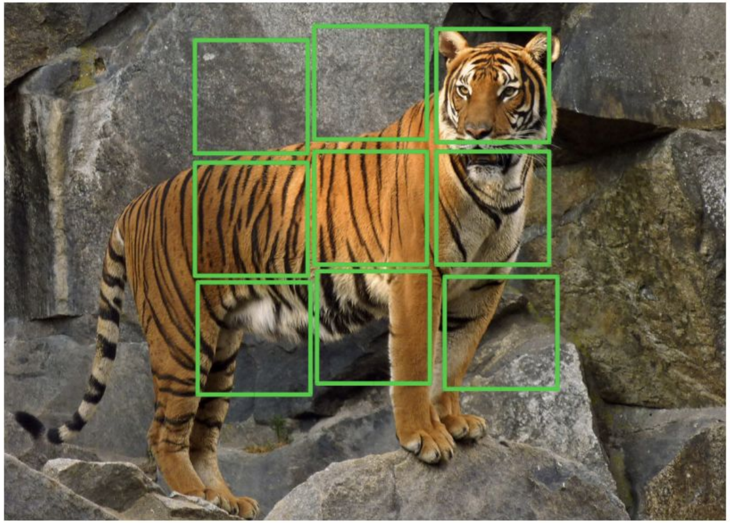
Include a gap

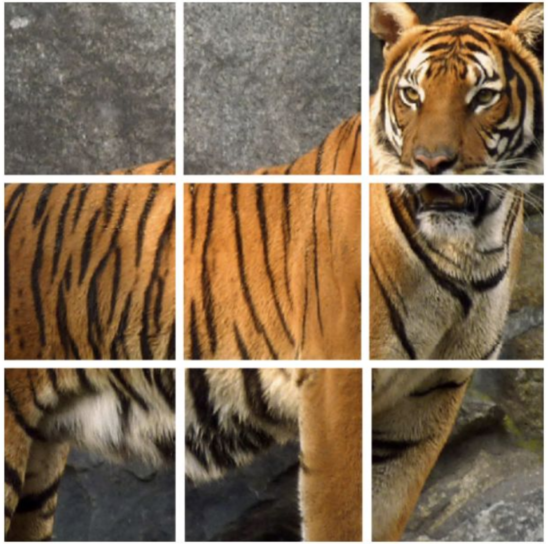
Jitter the patch locations



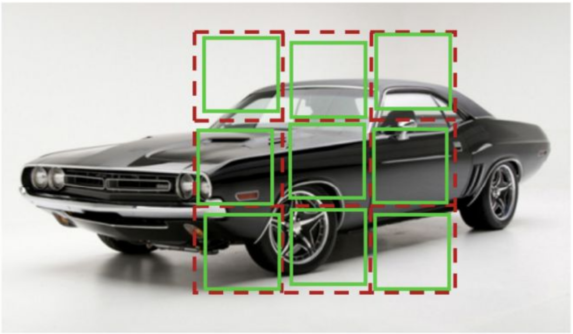


Solving Jigsaw Puzzles





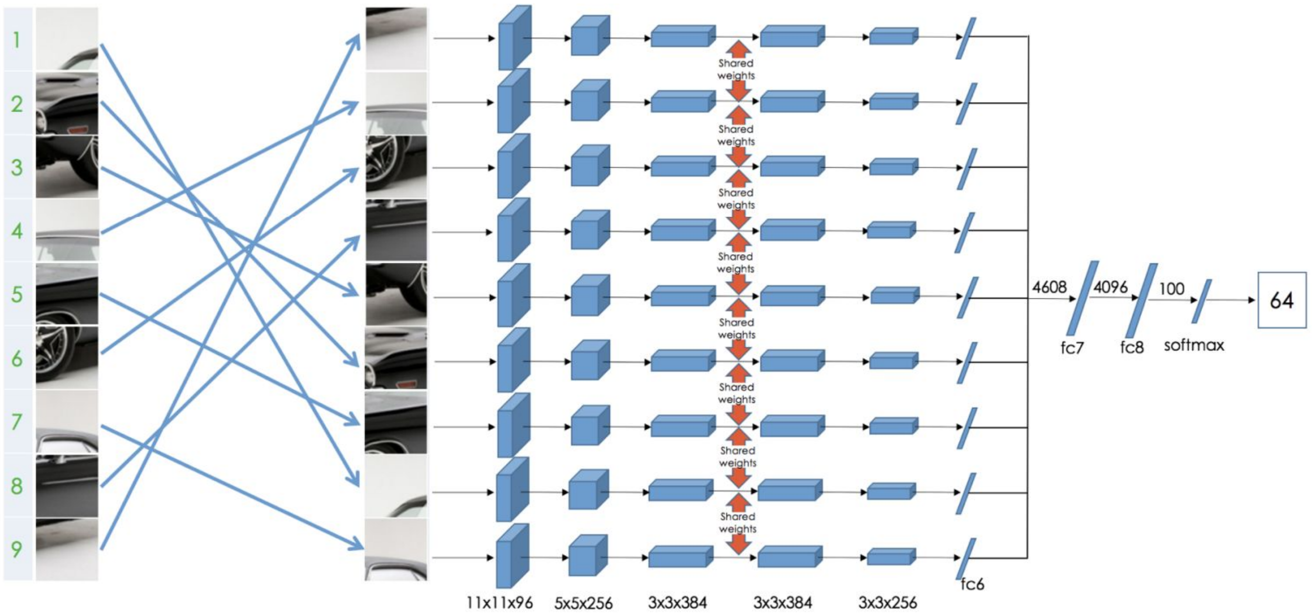
Solving Jigsaw Puzzles



Permutation Set

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

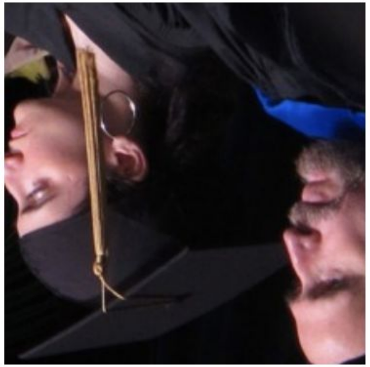
Reorder patches according to the selected permutation



Rotation

90° rotation

270° rotation

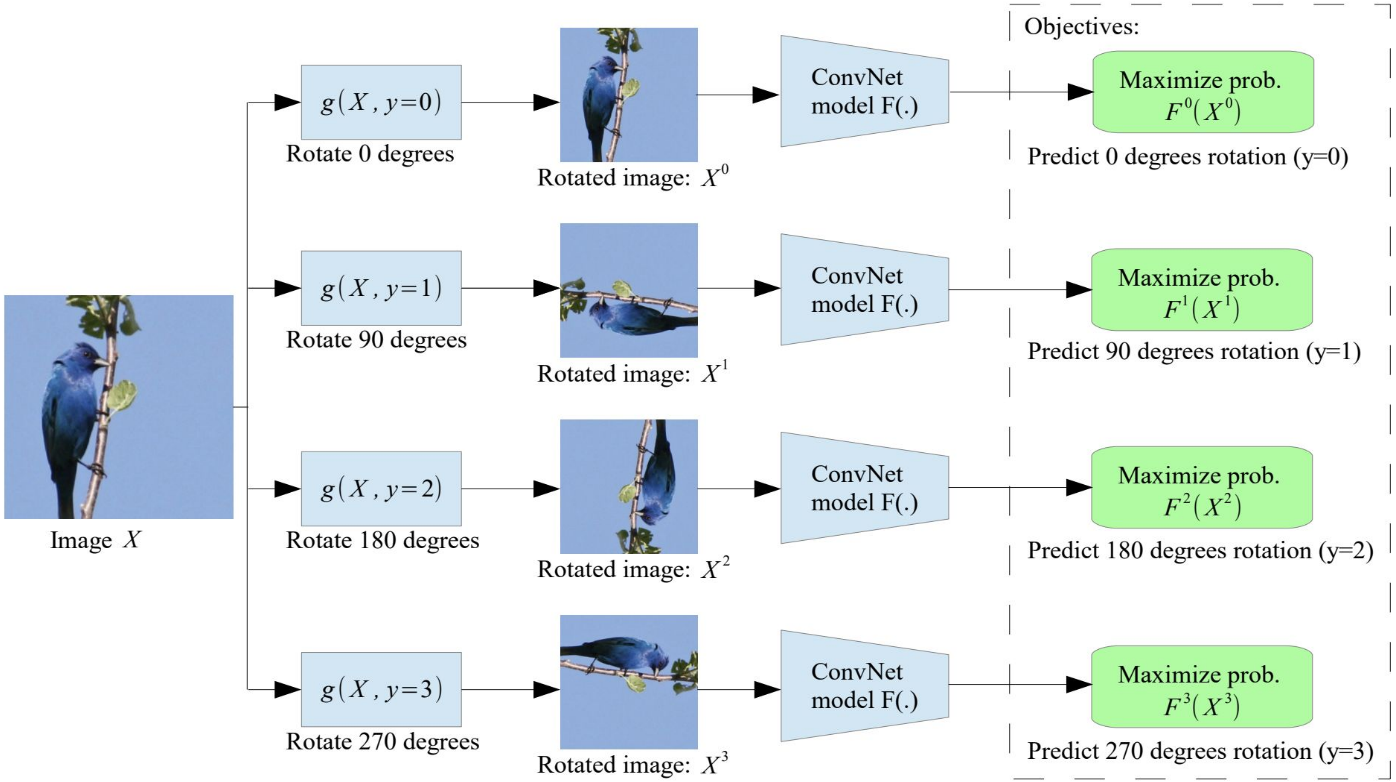


180° rotation

0° rotation

270° rotation

Rotation


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

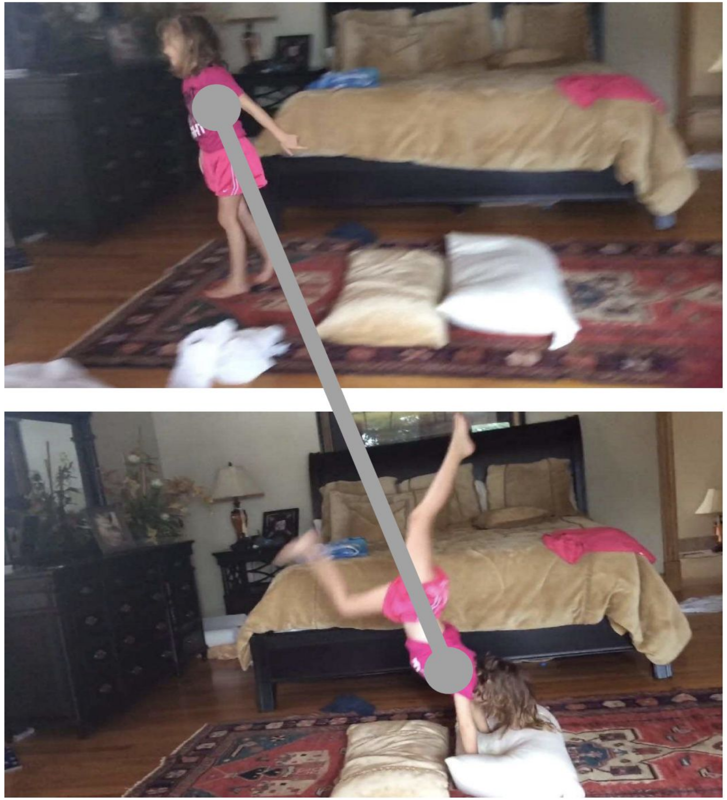
Gray-scale Video

Slide: Zisserman

Temporal coherence of color

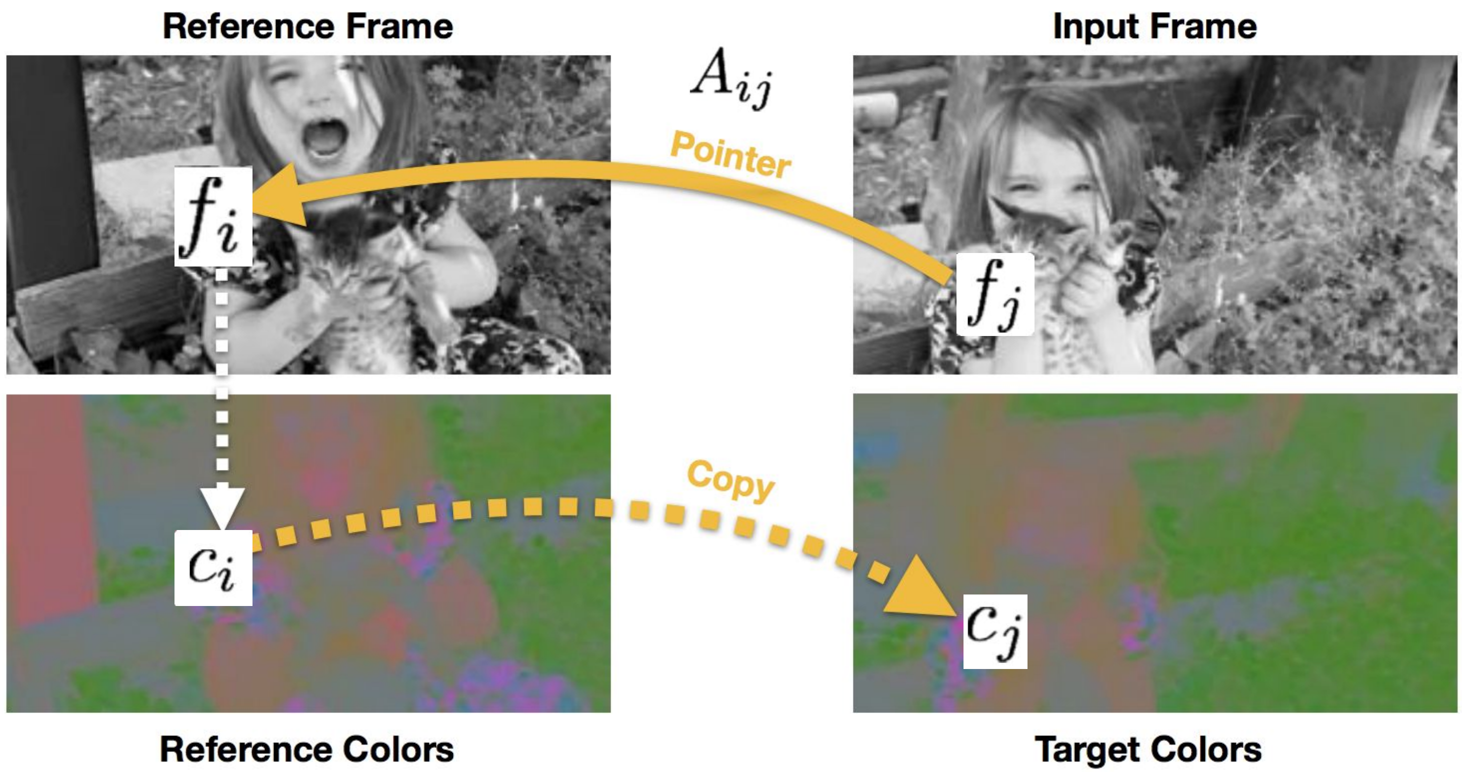
Slide: Zisserman

Temporal coherence of color

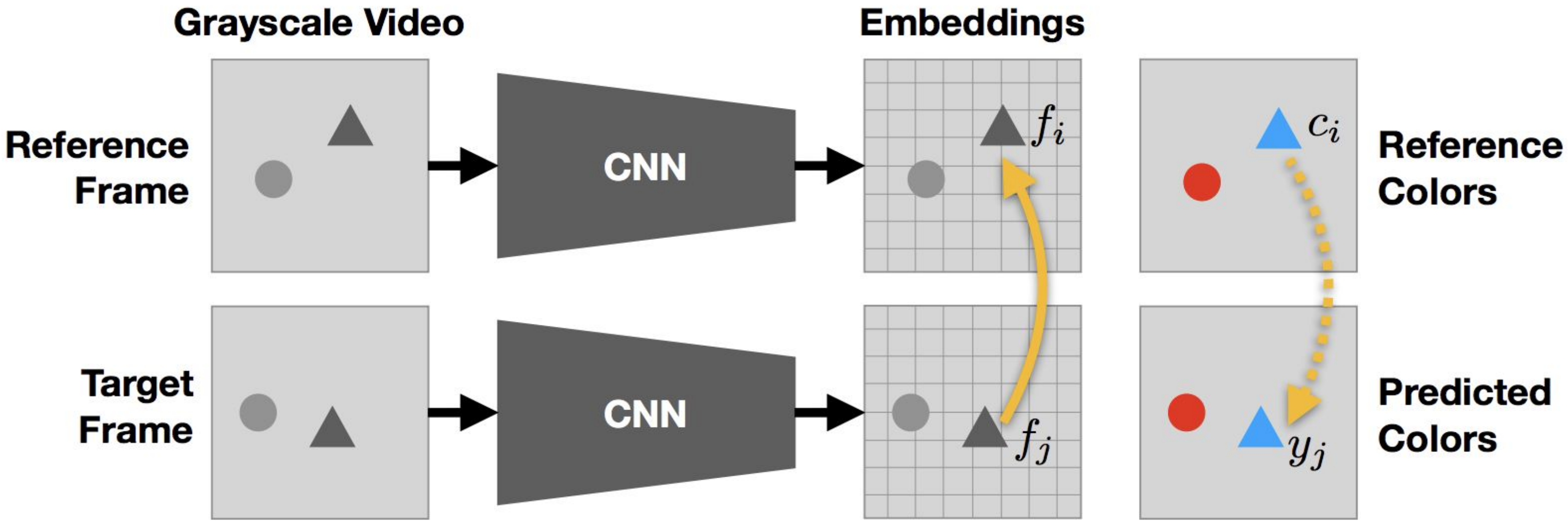


Slide: Zisserman

Tracking emerges from colorization

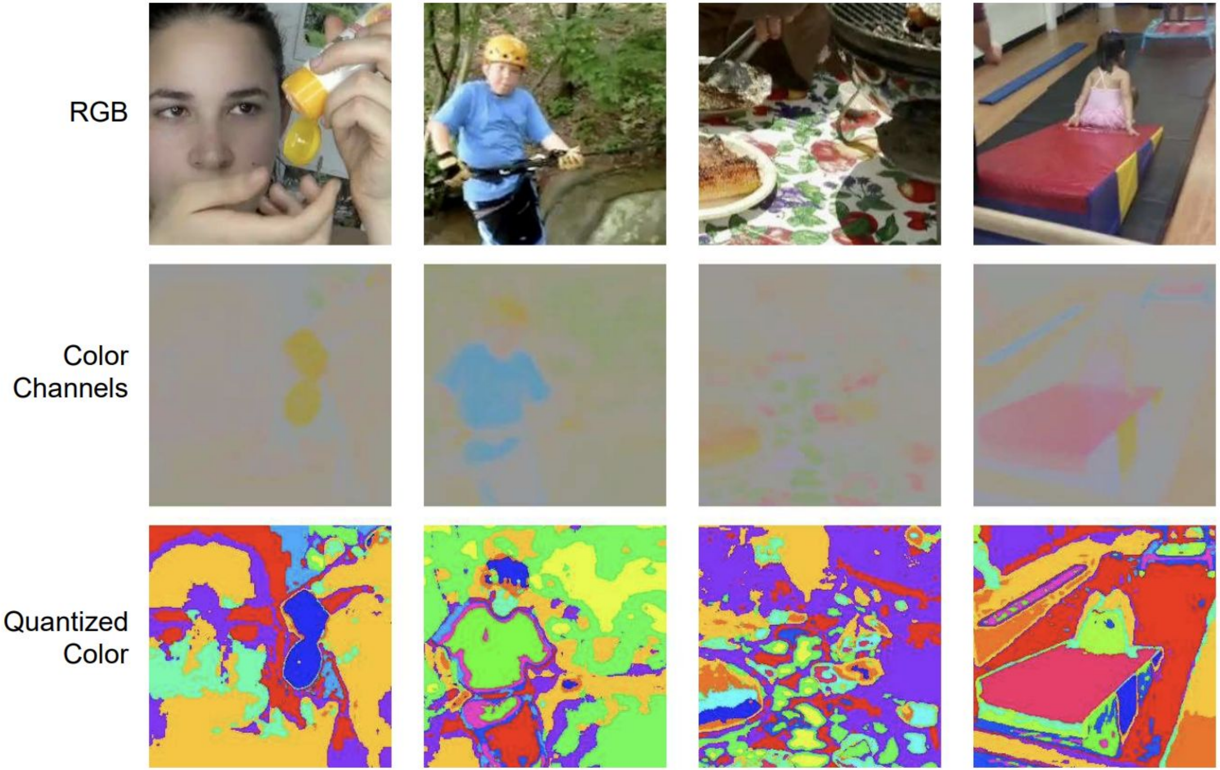


Tracking emerges from colorization



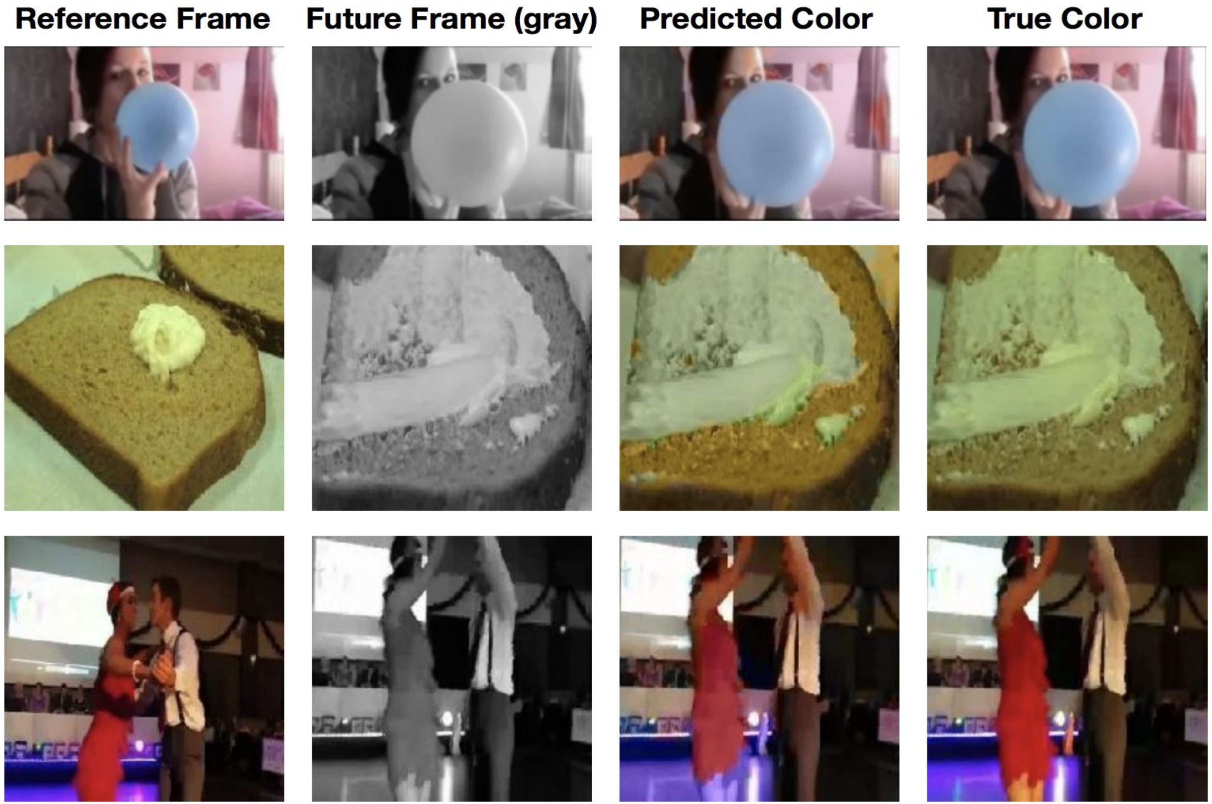
$$A_{ij} = \frac{\exp\left(f_i^T f_j\right)}{\sum_k \exp\left(f_k^T f_j\right)} \qquad \hat{c}_j = \sum_i A_{ij} c_i \qquad \min_f \mathcal{L}\left(c_j, \sum_i A_{ij} c_i\right)$$

Tracking emerges from colorization

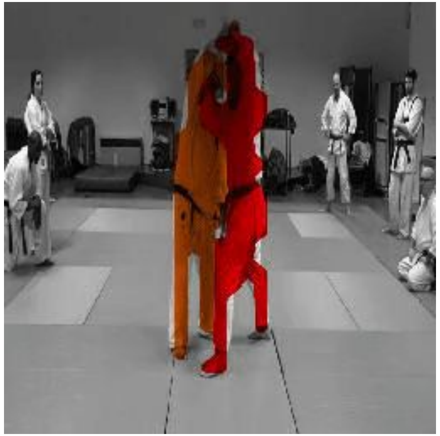


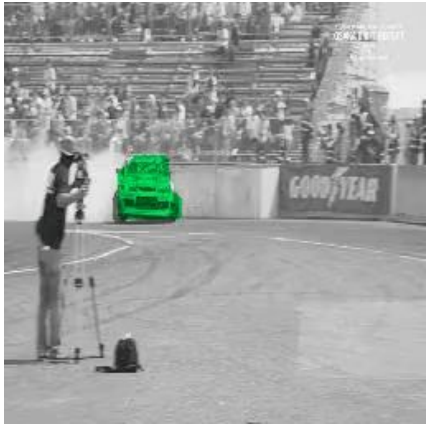
Slide: Zisserman

Tracking emerges from colorization



Tracking emerges from colorization

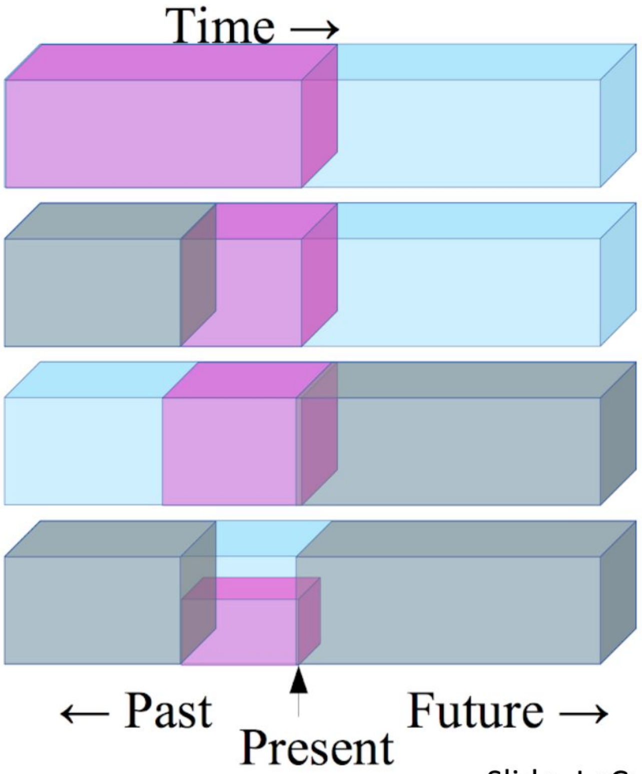




GIFs from Google AI Blog post

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:

$$
\begin{array}{cc} & |V| \\ |V| & \left[\begin{array}{ccc} & & \\ & X & \\ & & \end{array} \right] \end{array}
=
\begin{array}{cc} & |V| \\ |V| & \left[\begin{array}{ccc} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{array} \right] \end{array}
\begin{array}{cc} & |V| \\ |V| & \left[\begin{array}{ccc} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{array} \right] \end{array}
\begin{array}{cc} & |V| \\ |V| & \left[\begin{array}{ccc} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{array} \right] \end{array}
$$

(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

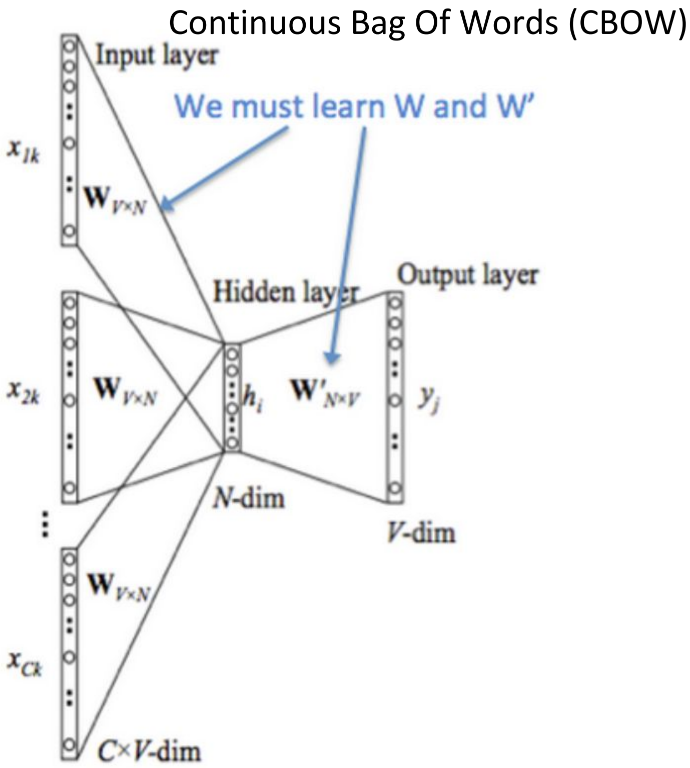
Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

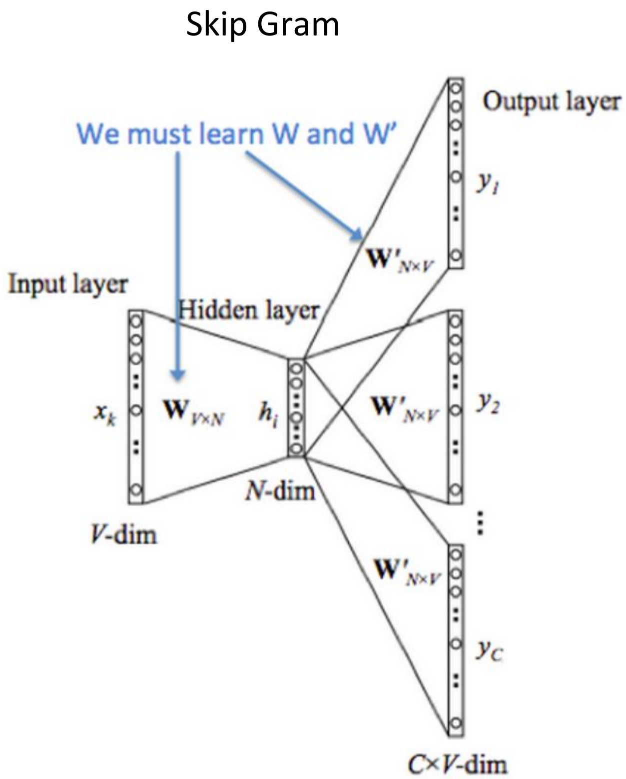
Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

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

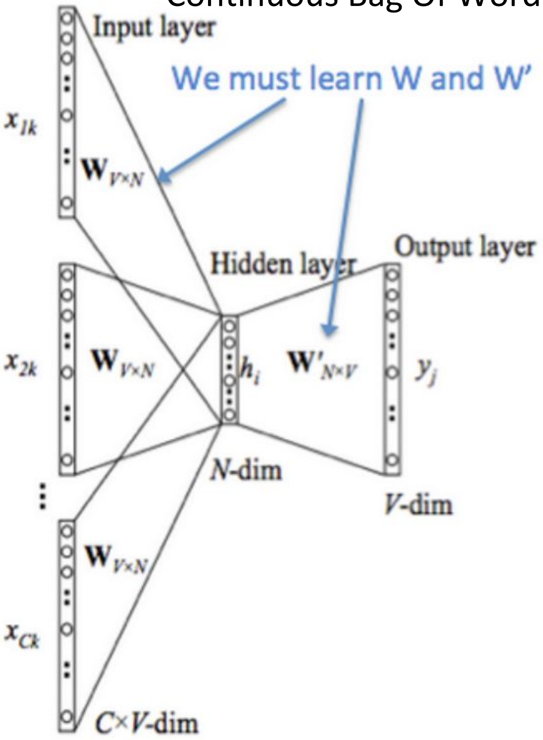




(From 224n Stanford)

word2vec - CBOW

Continuous Bag Of Words (CBOW)

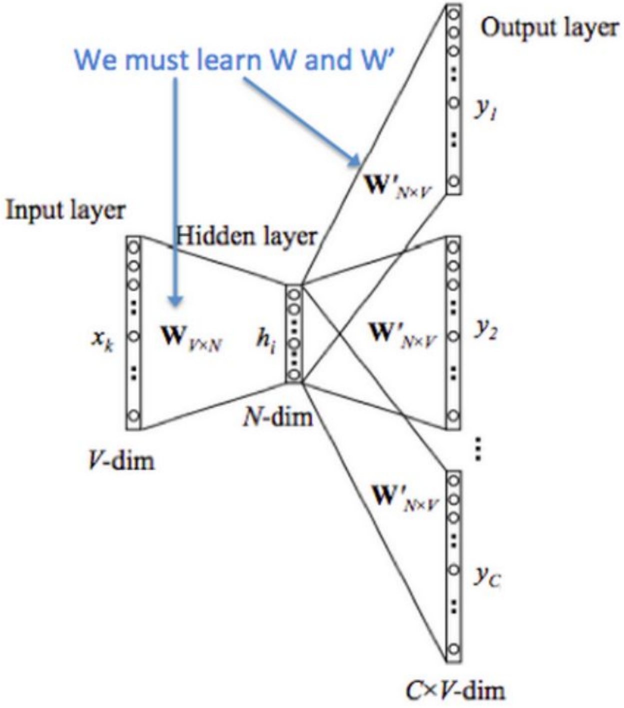


$$\begin{aligned}
\text{minimize } J &= -\log P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}) \\
&= -\log P(u_c | \hat{v}) \\
&= -\log \frac{\exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\
&= -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})
\end{aligned}$$

(From 224n Stanford)

word2vec - Skip Gram

Skip Gram



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

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

(From 224n Stanford)

word2vec - Skip Gram

Skip-gram model

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

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

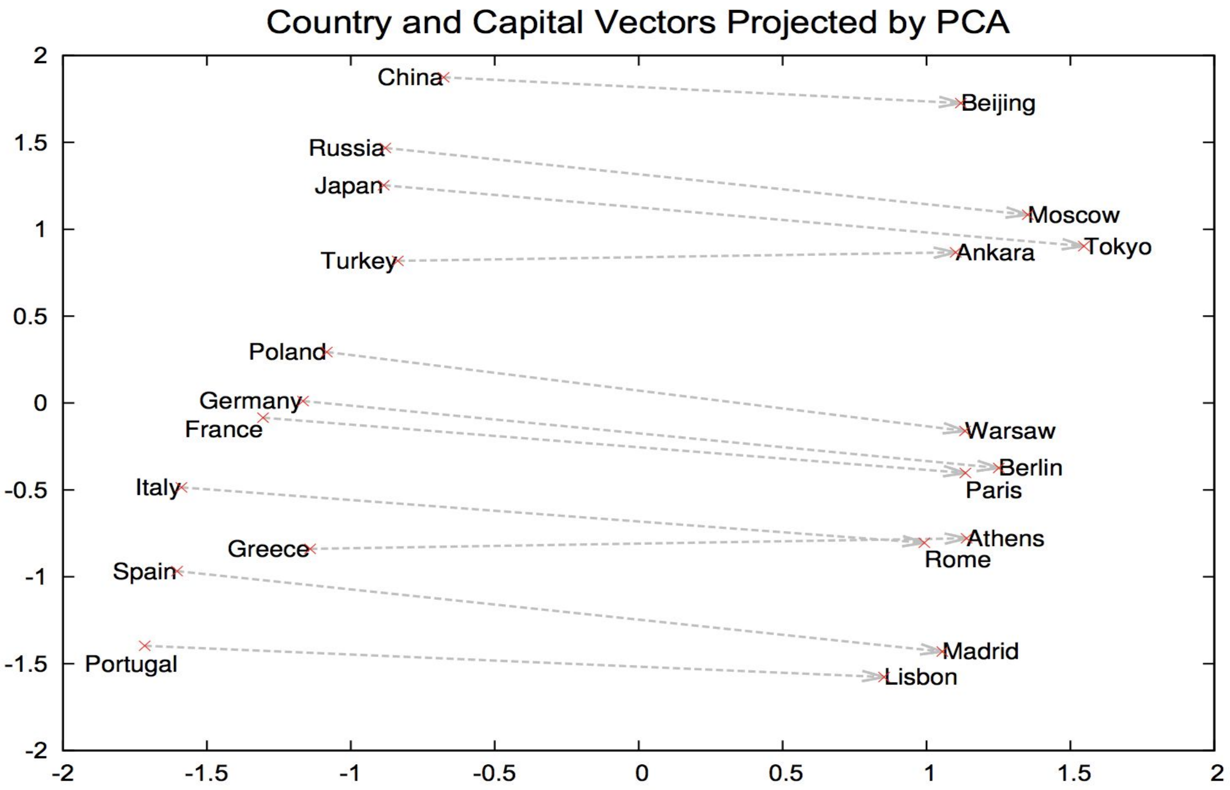
Don’t have to have the denominator over all words in the vocabulary

- 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
Germany
France
Warsaw
Berlin
Paris
Italy
Greece
Spain
Athens
Rome
Madrid
Lisbon
Portugal
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

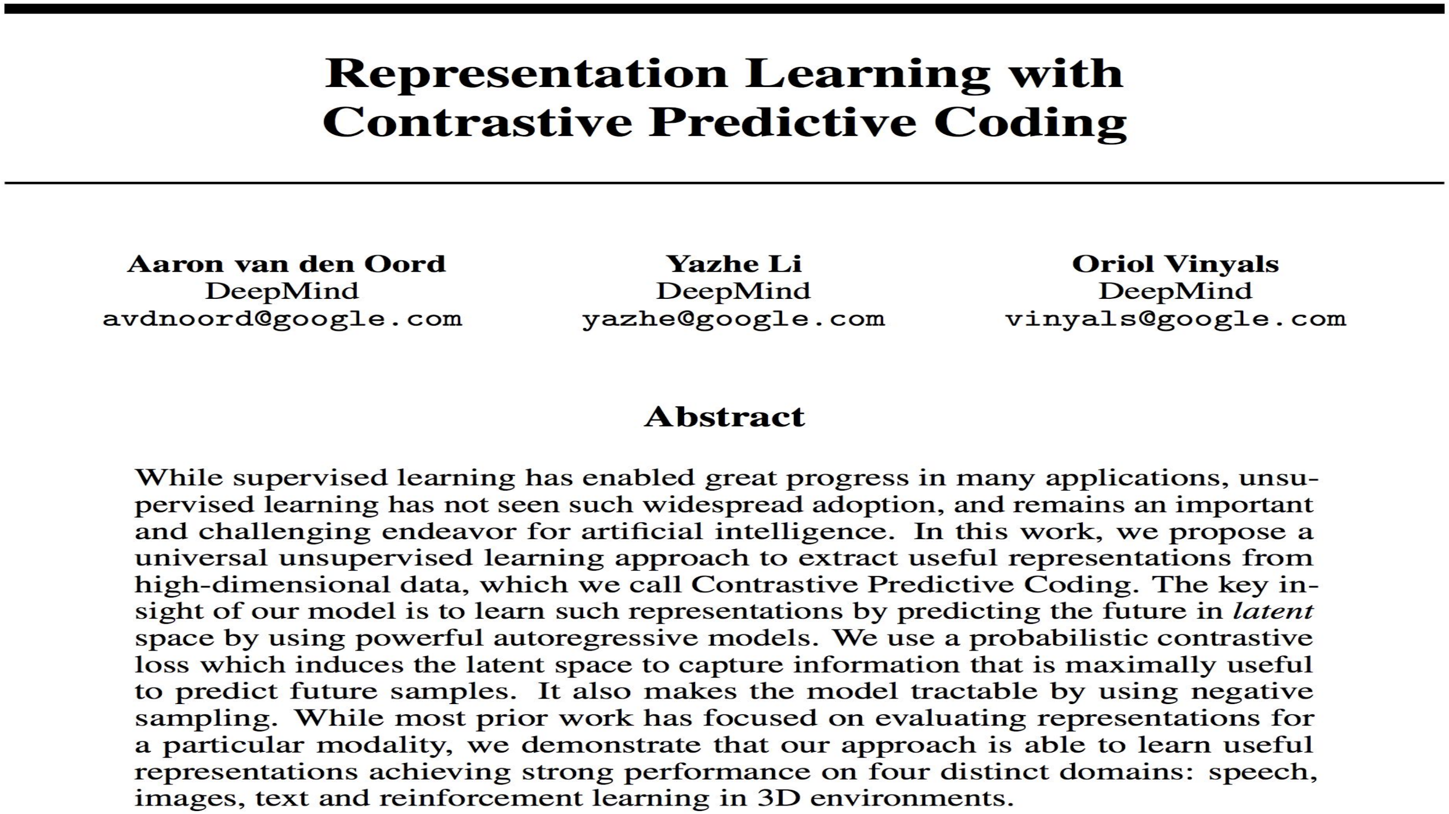
Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

word2vec

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona 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

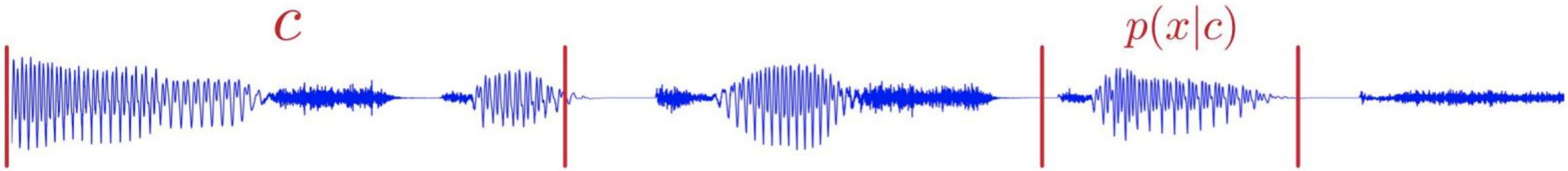


Figure from Alex Graves

Contrastive Predictive Coding

Figure from Alex Graves

Contrastive Predictive Coding

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

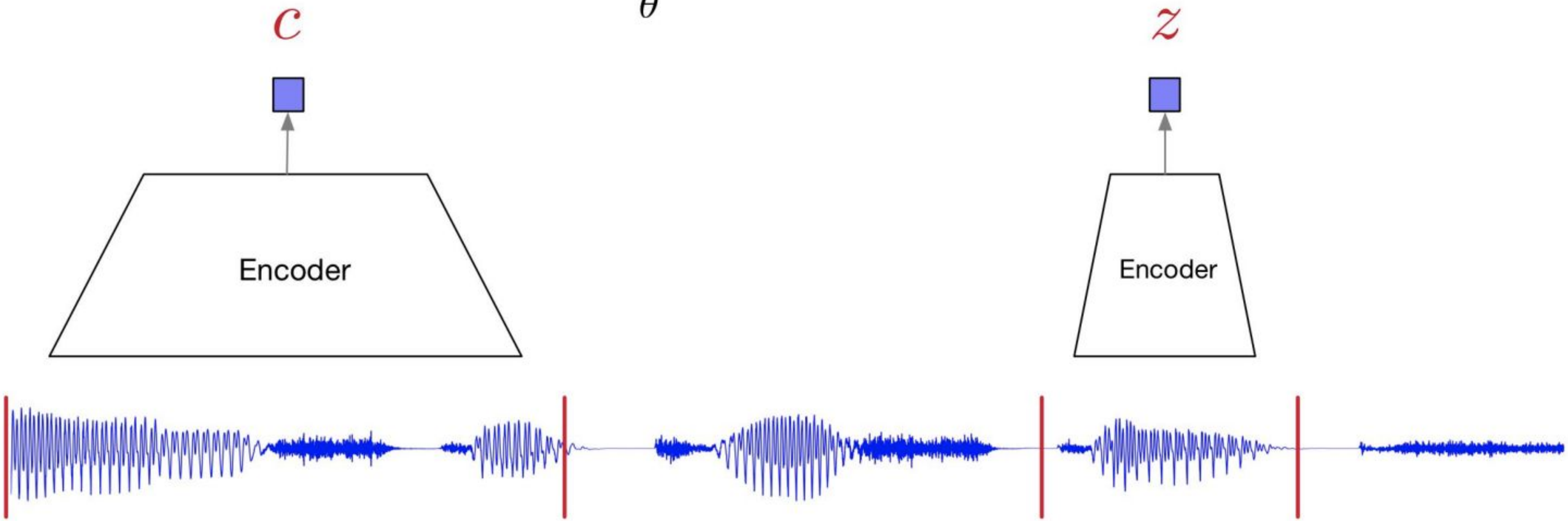


Figure from Alex Graves

Contrastive Predictive Coding

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

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

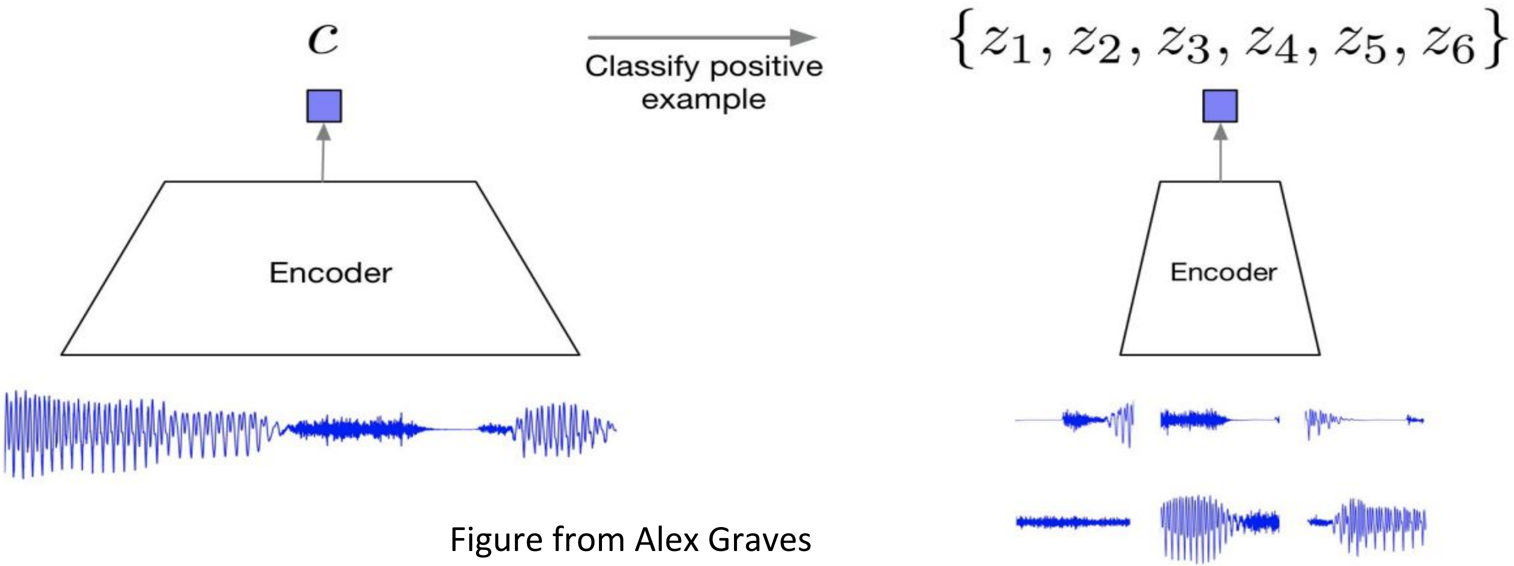


Figure from Alex Graves

Contrastive Predictive Coding

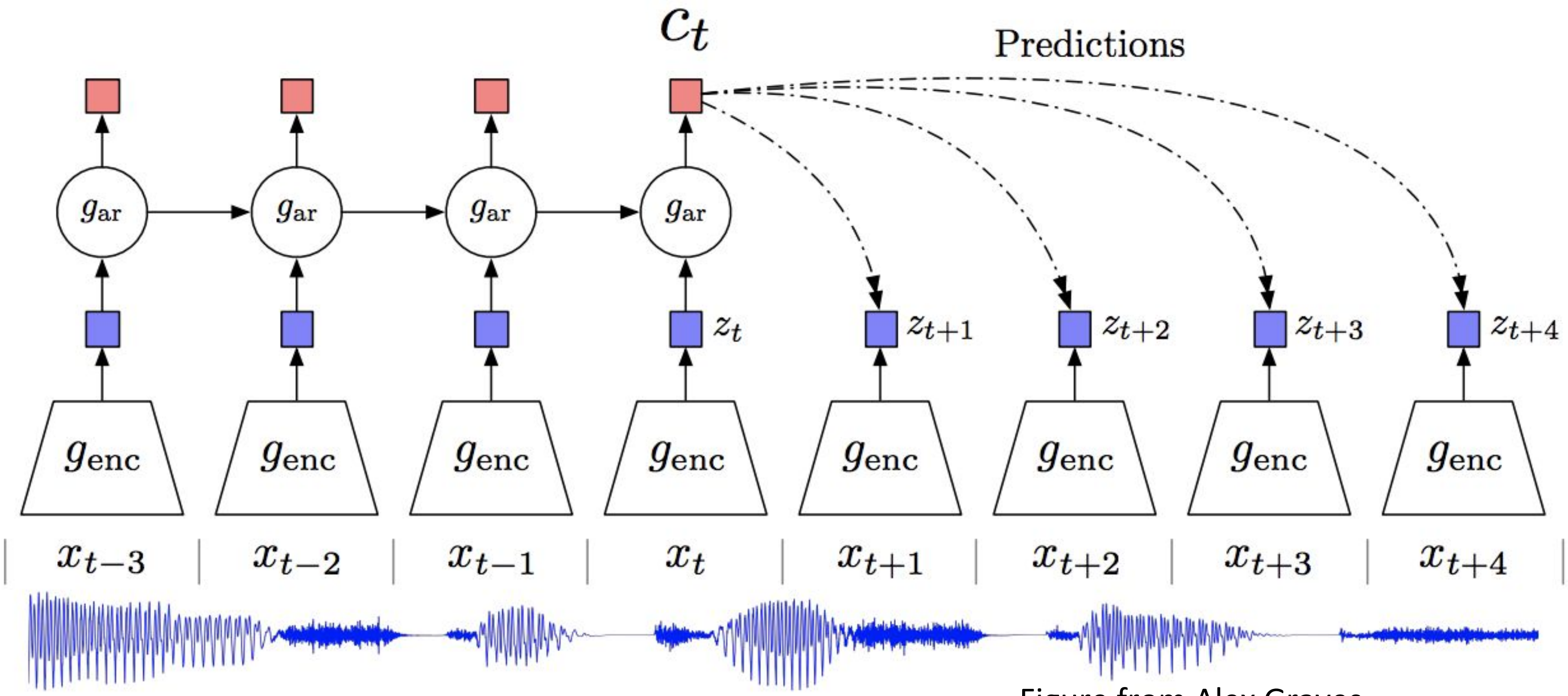
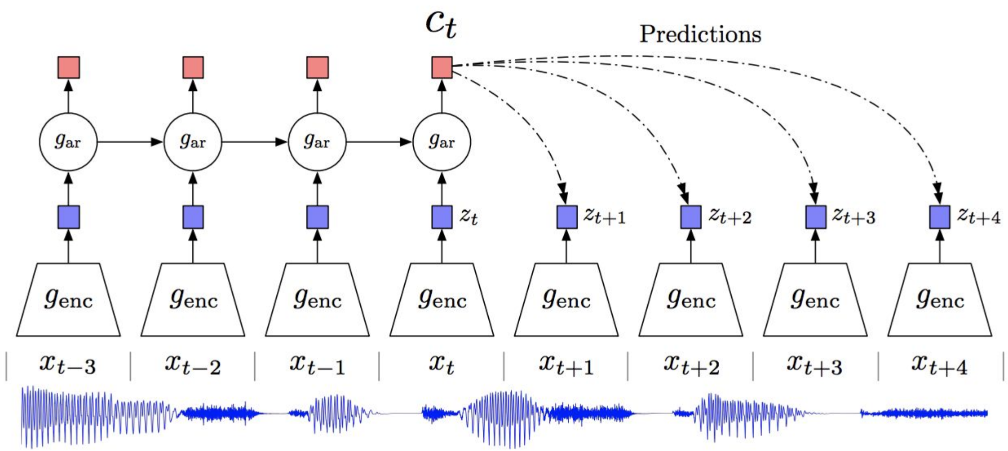


Figure from Alex Graves

Contrastive Predictive Coding

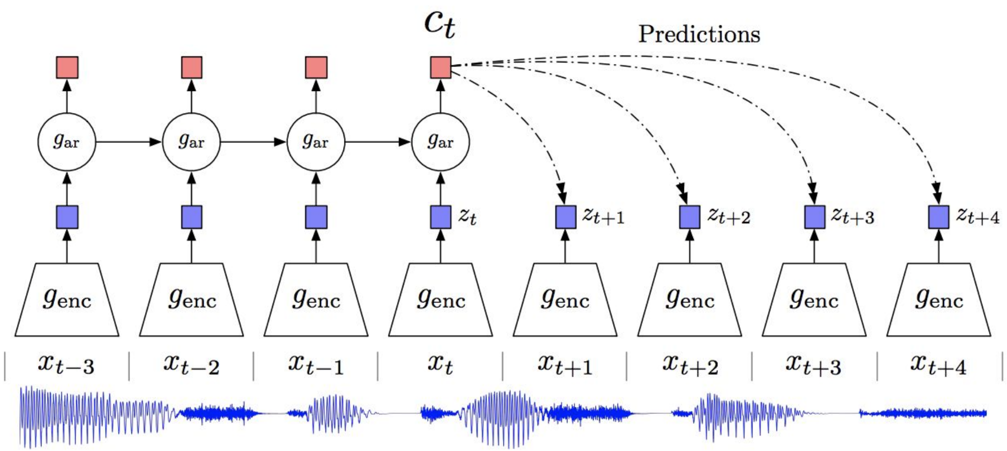


$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

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

Figure from Alex Graves

Contrastive Predictive Coding



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

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

Figure from Alex Graves

Contrastive Predictive Coding

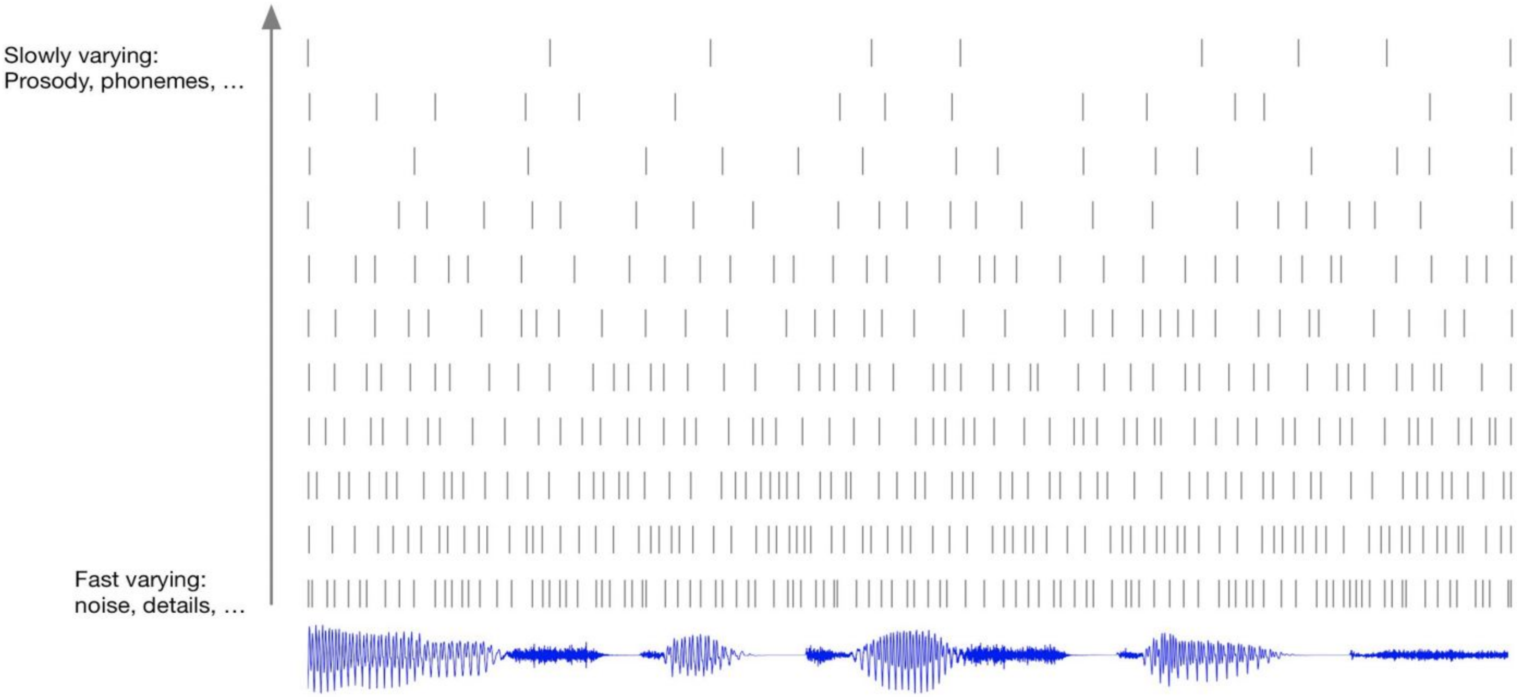


Figure from Alex Graves

Contrastive Predictive Coding

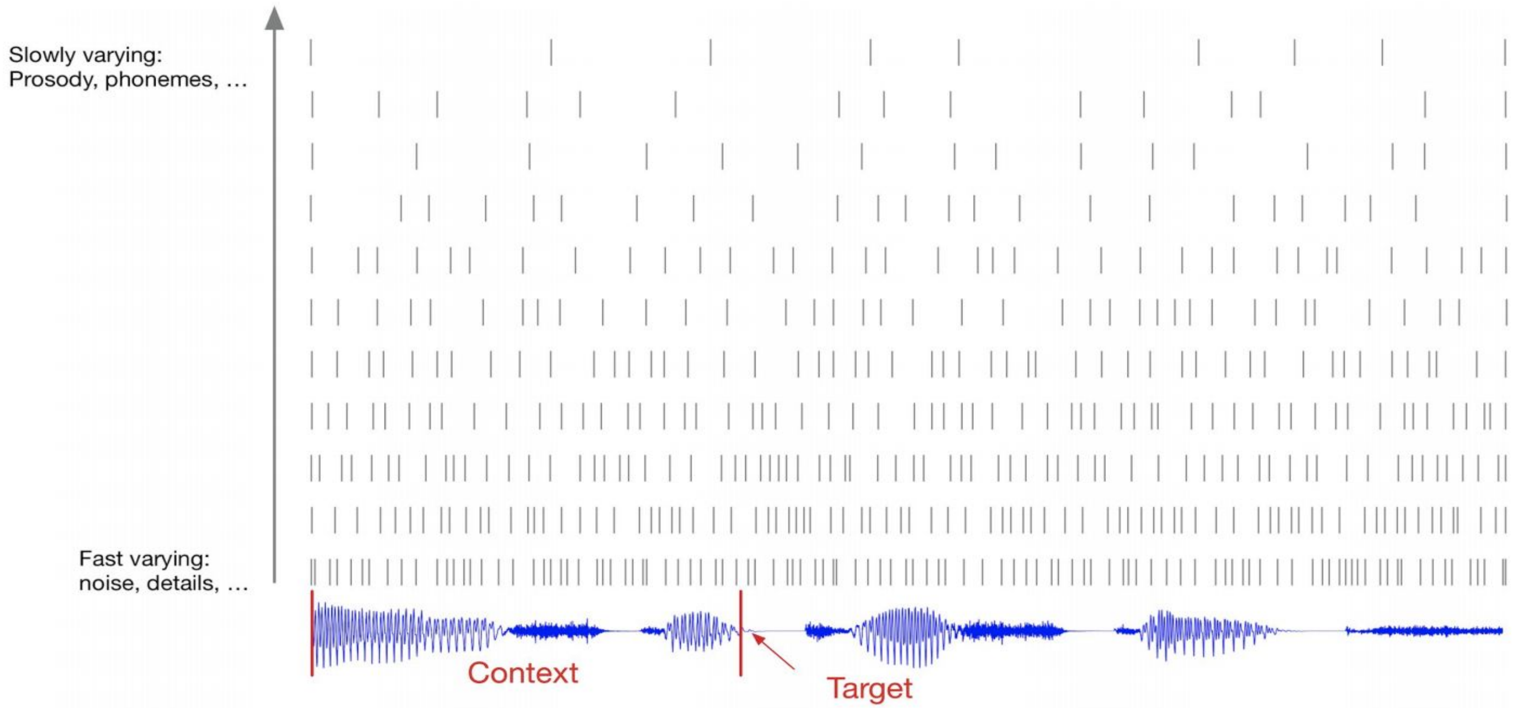


Figure from Alex Graves

Contrastive Predictive Coding

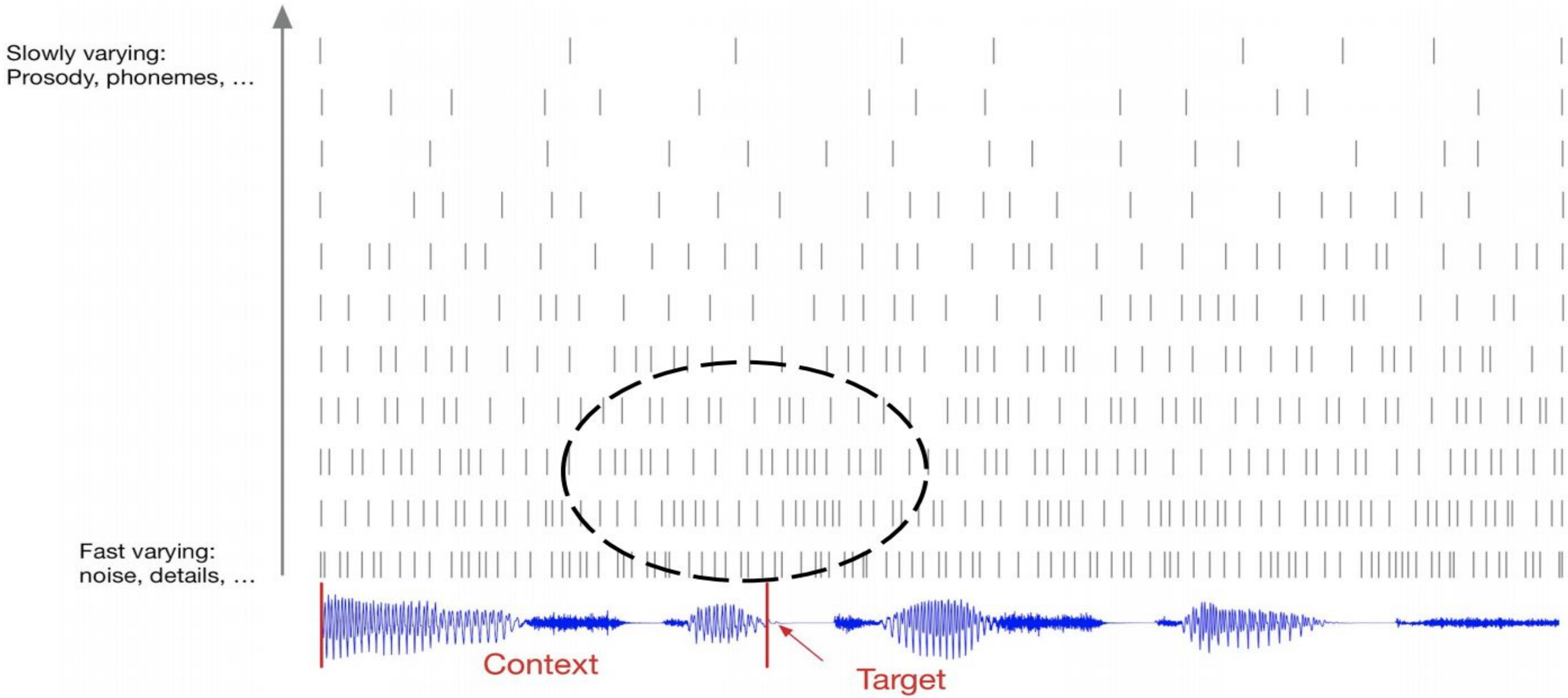


Figure from Alex Graves

Contrastive Predictive Coding

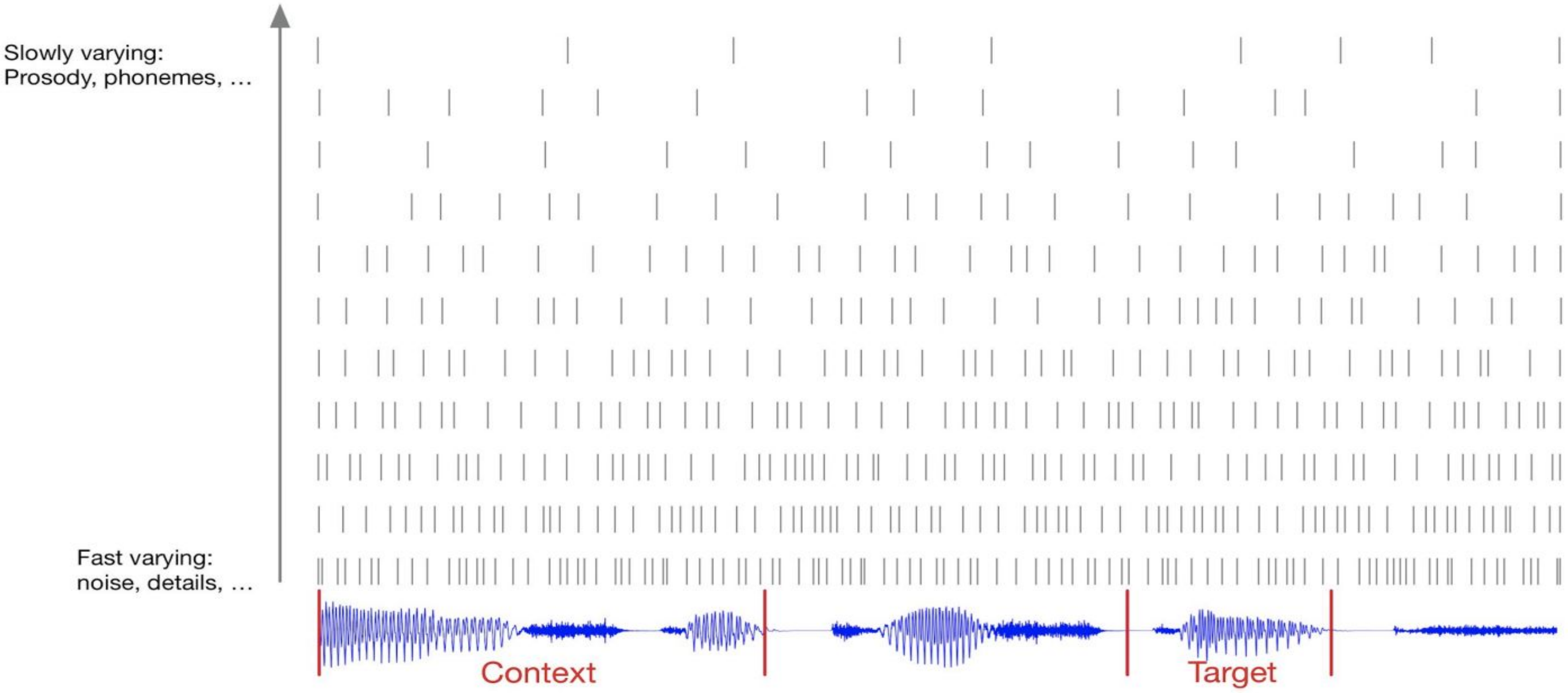


Figure from Alex Graves

Contrastive Predictive Coding

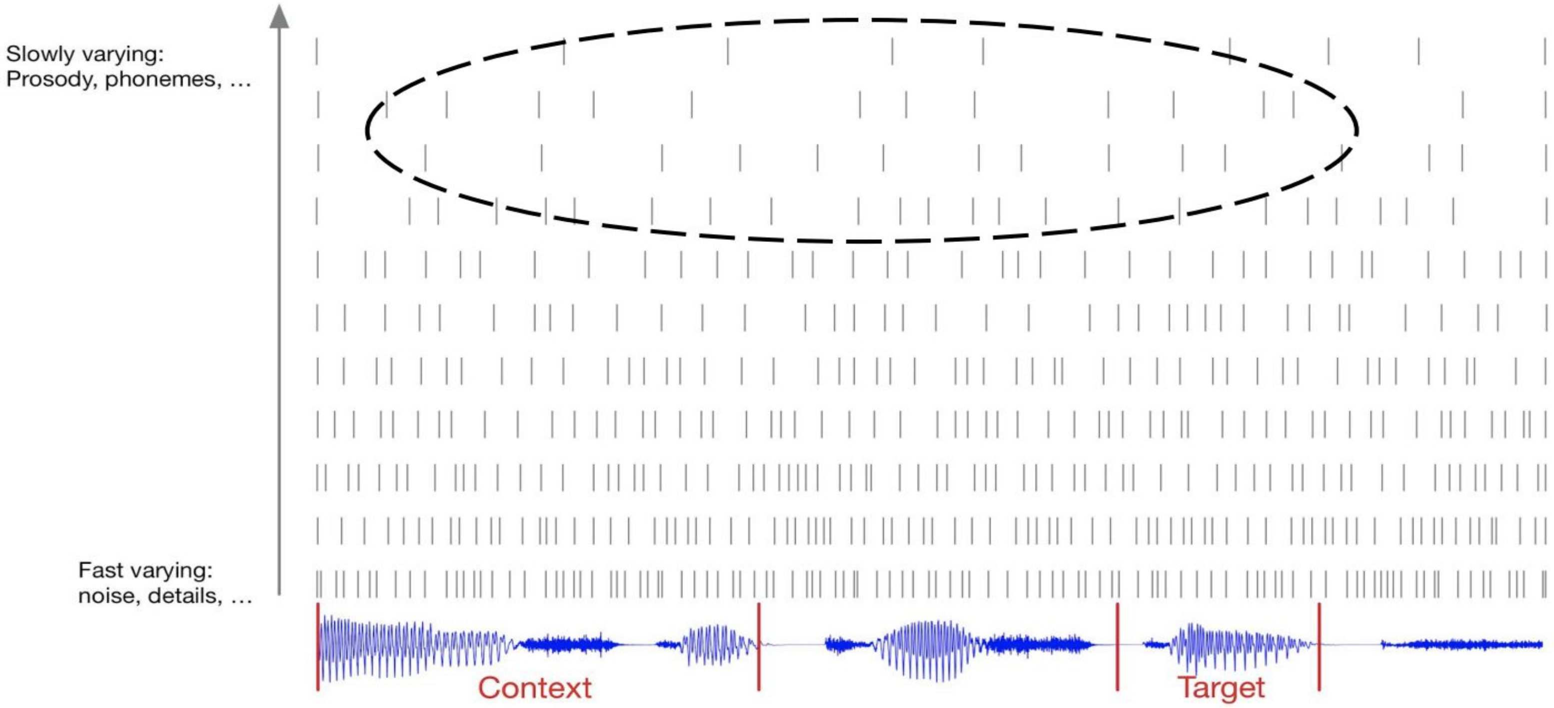


Figure from Alex Graves

CPC - Speech

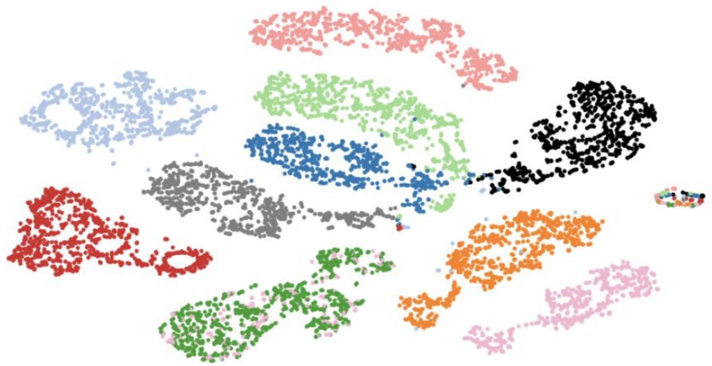


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

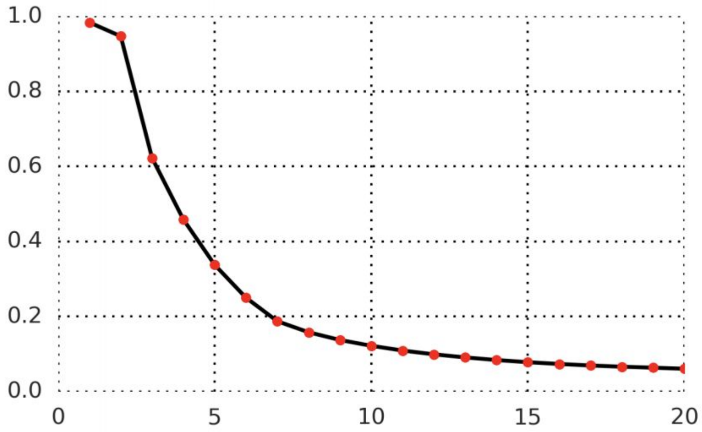


Figure 3: Average accuracy of predicting the positive sample in the contrastive loss for 1 to 20 latent steps in the future of a speech waveform. The model predicts up to 200ms in the future as every step consists of 10ms of audio.

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

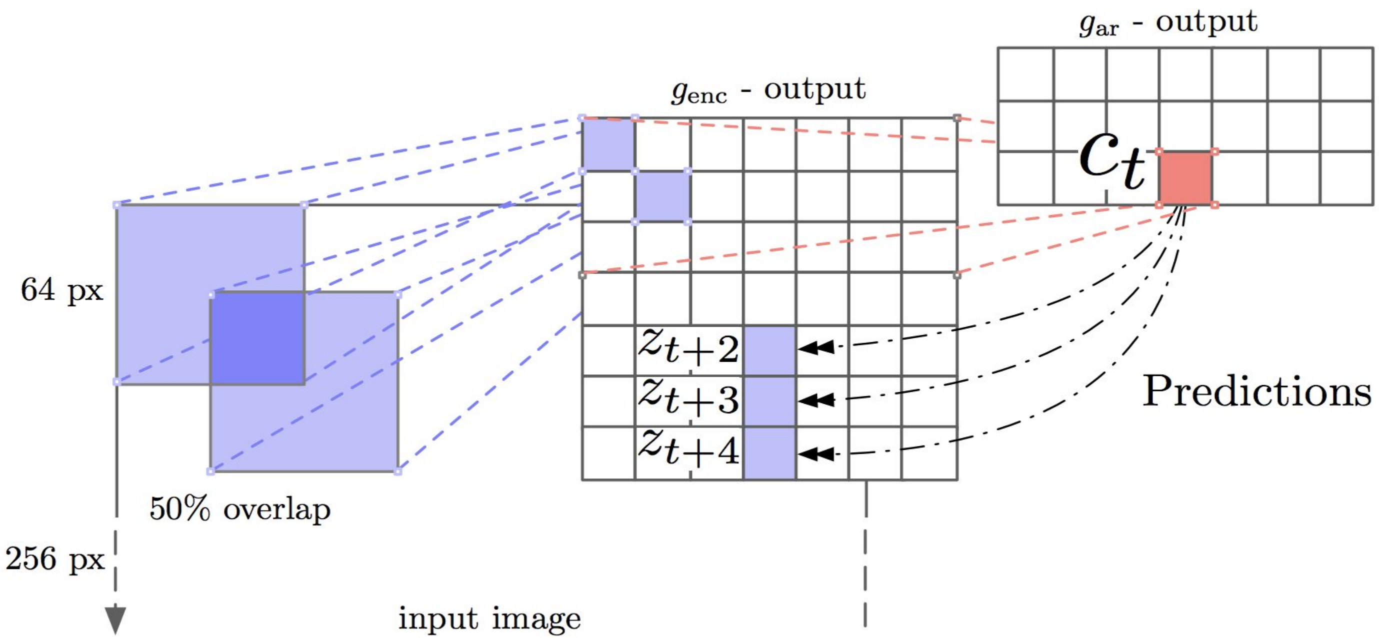
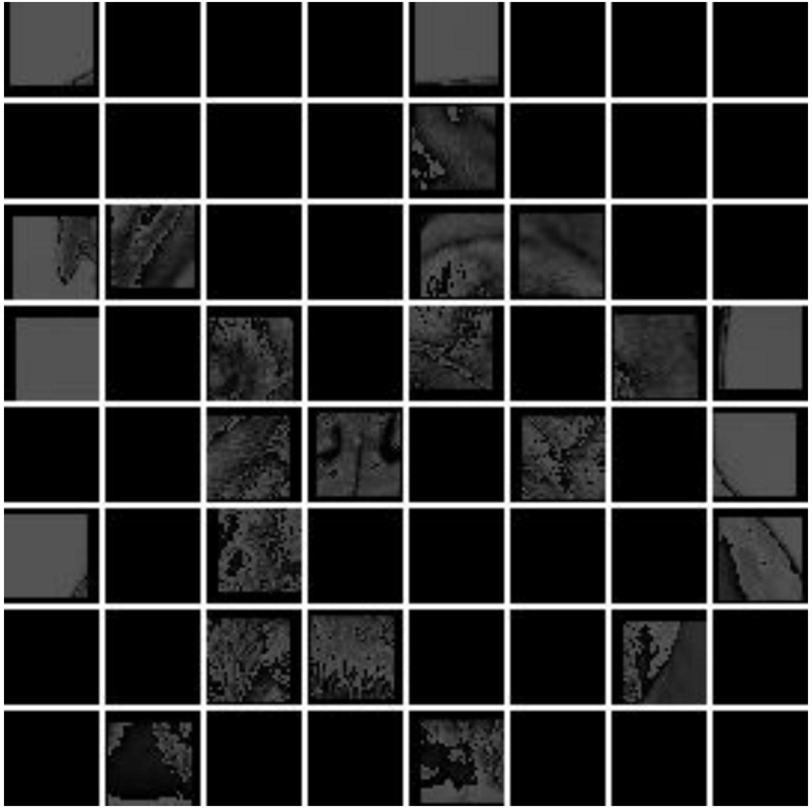


Figure 4: Visualization of Contrastive Predictive Coding for images (2D adaptation of Figure 1).

CPC - Imagenet

CPC - Imagenet



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

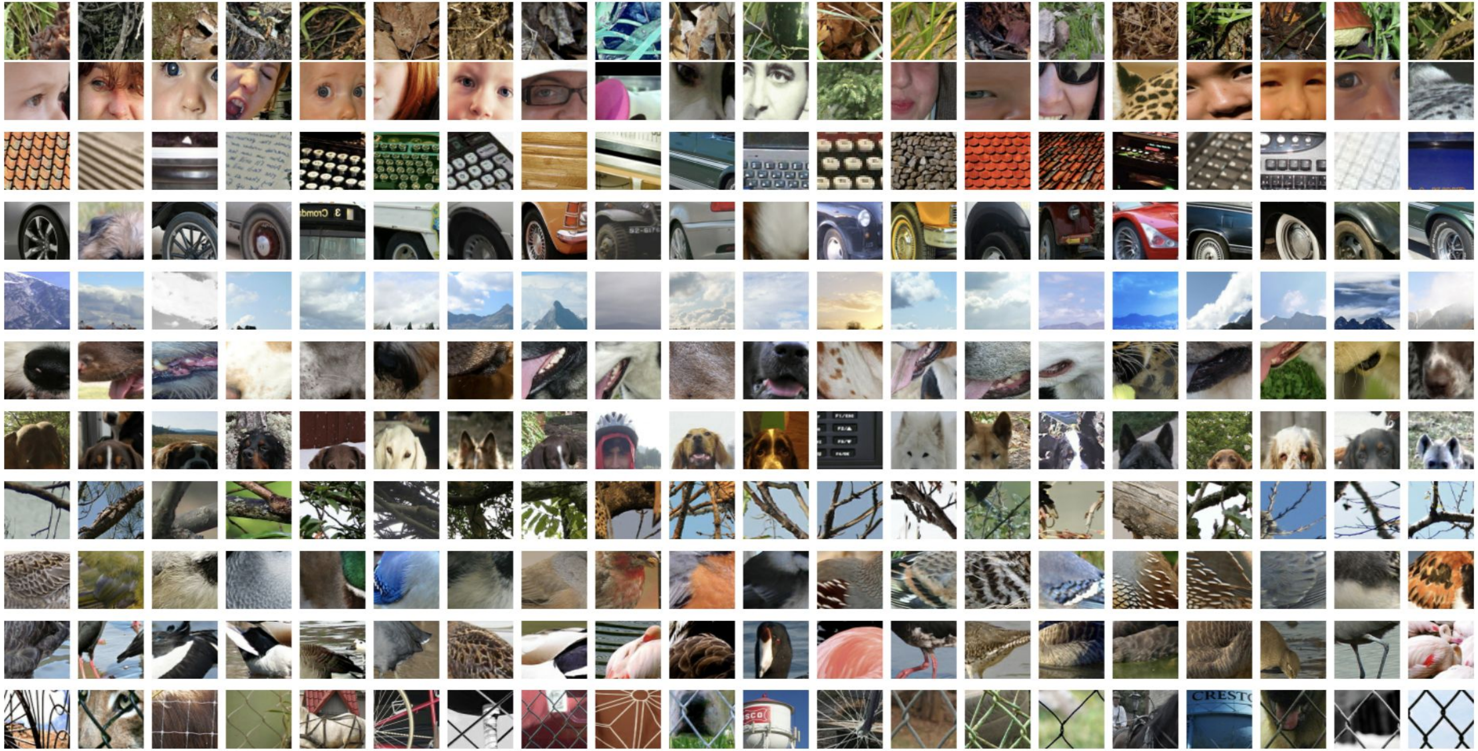


Figure 5: Every row shows image patches that activate a certain neuron in the CPC architecture.

CPC - NLP

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

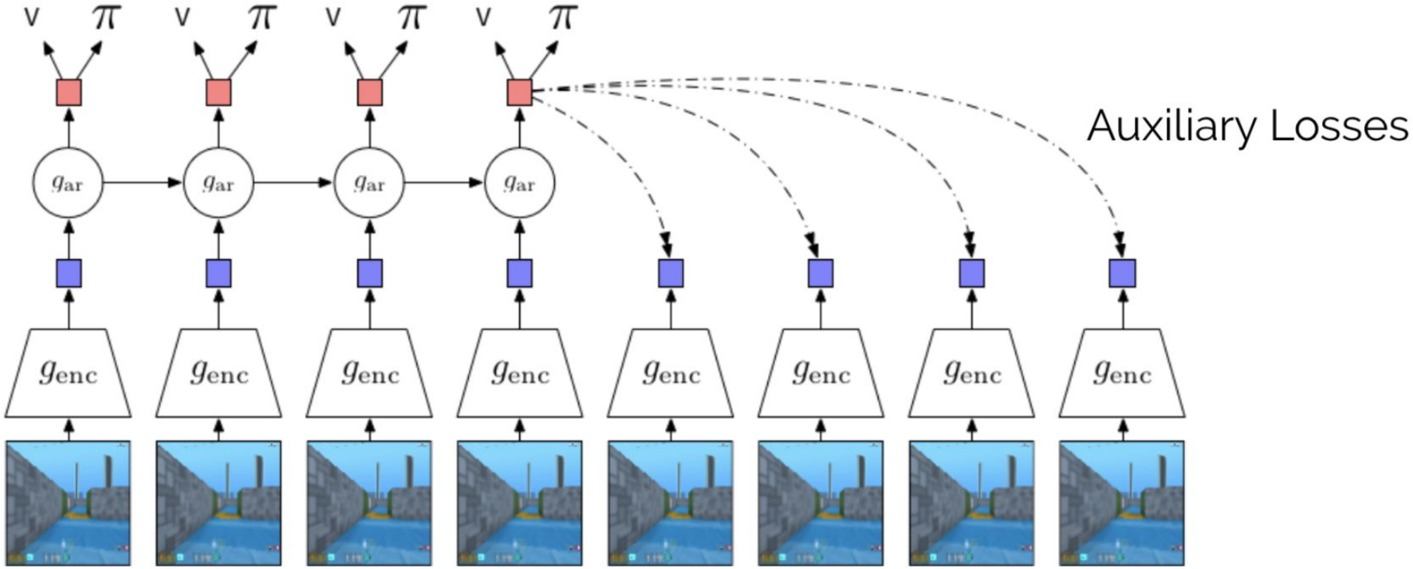


Figure from Aaron Van den Oord

BERT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Figures in subsequent slides taken from
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

BERT

king → [-0.5, -0.9, 1.4, …]

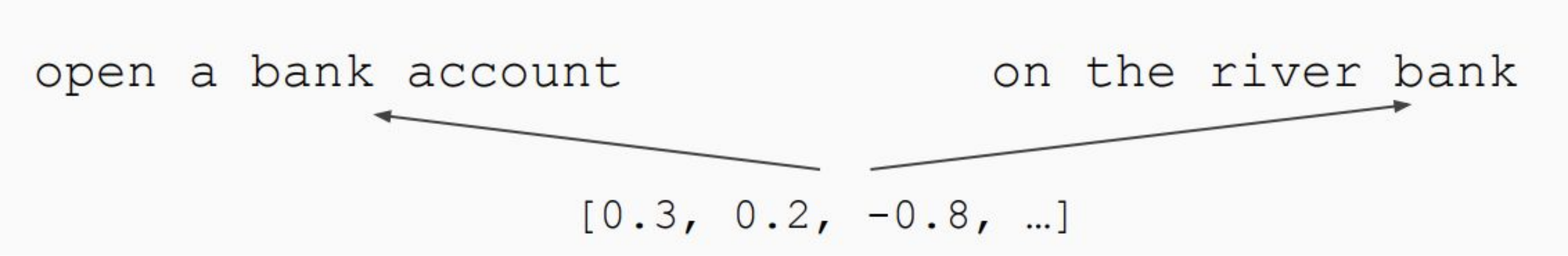
queen → [-0.6, -0.8, -0.2, …]

BERT

Typical

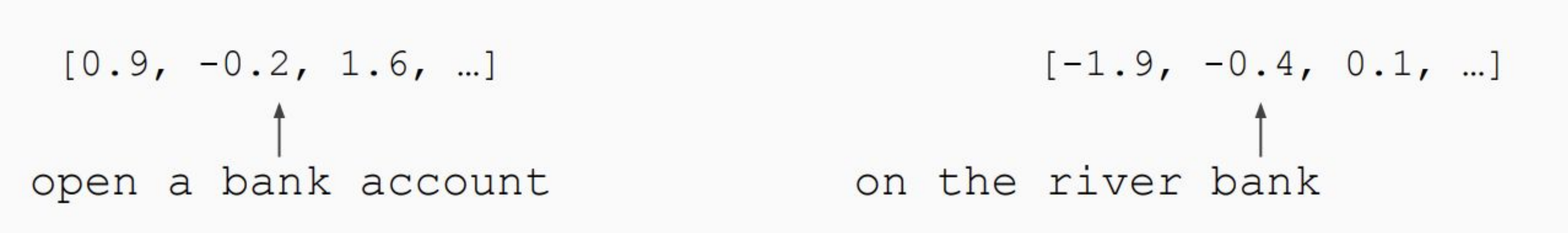
BERT

Problem: No context

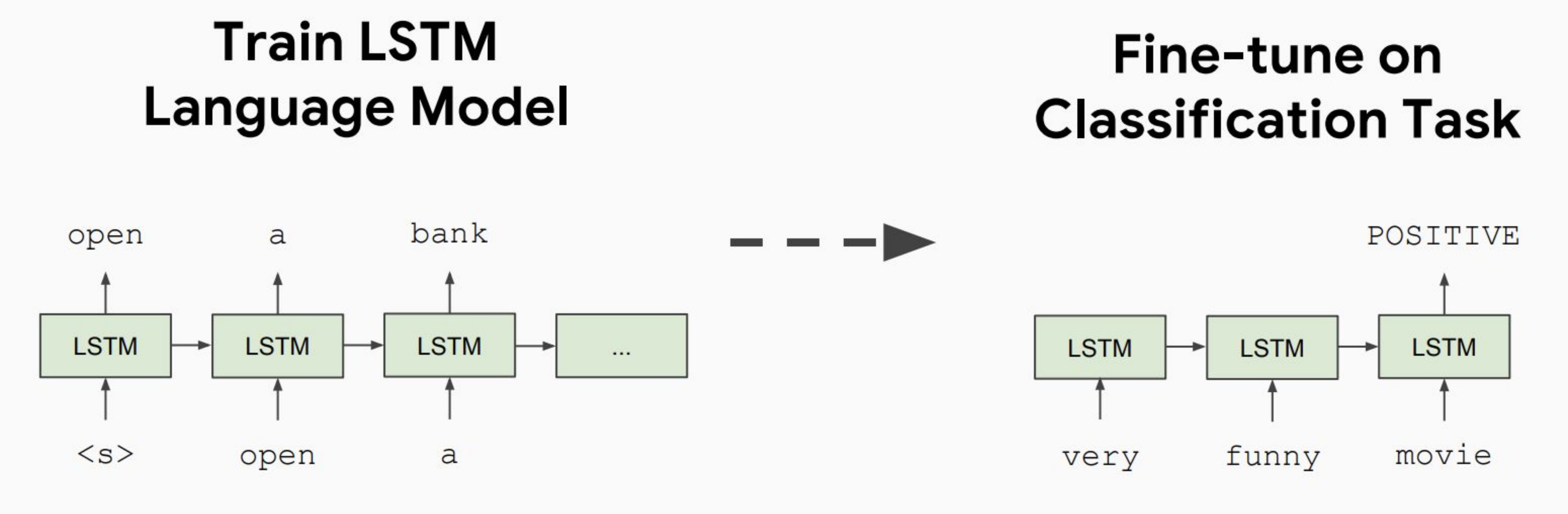


BERT

Solution: Encode with context

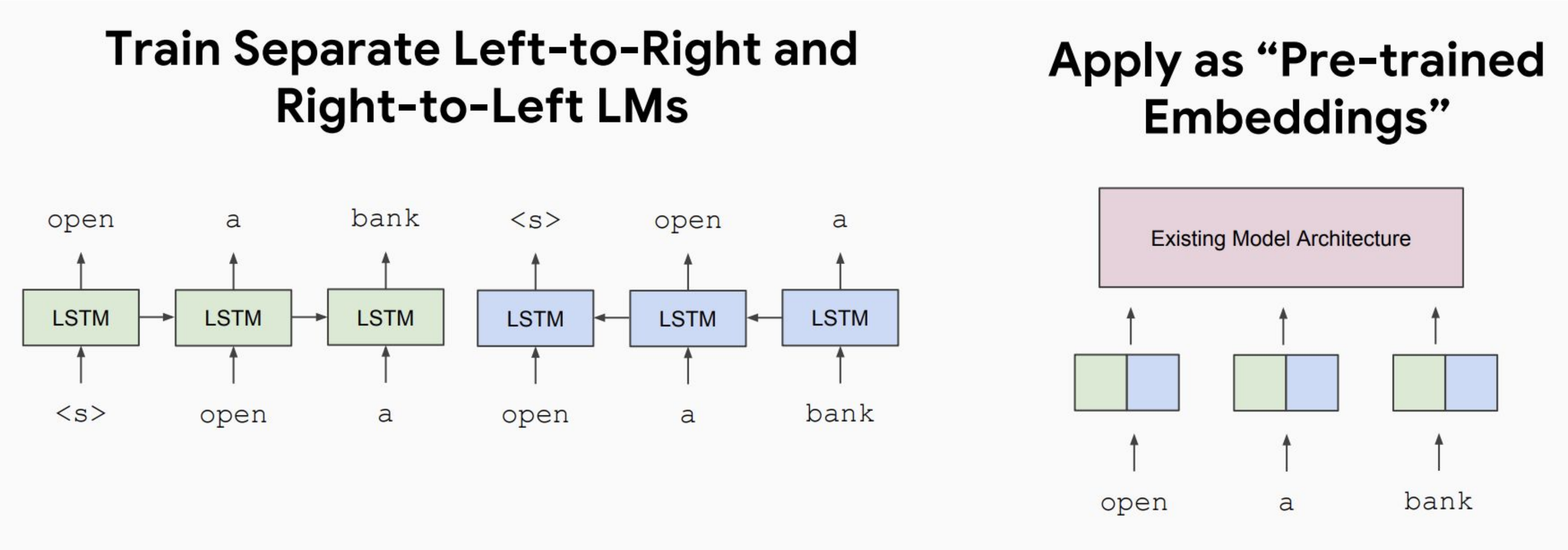


BERT



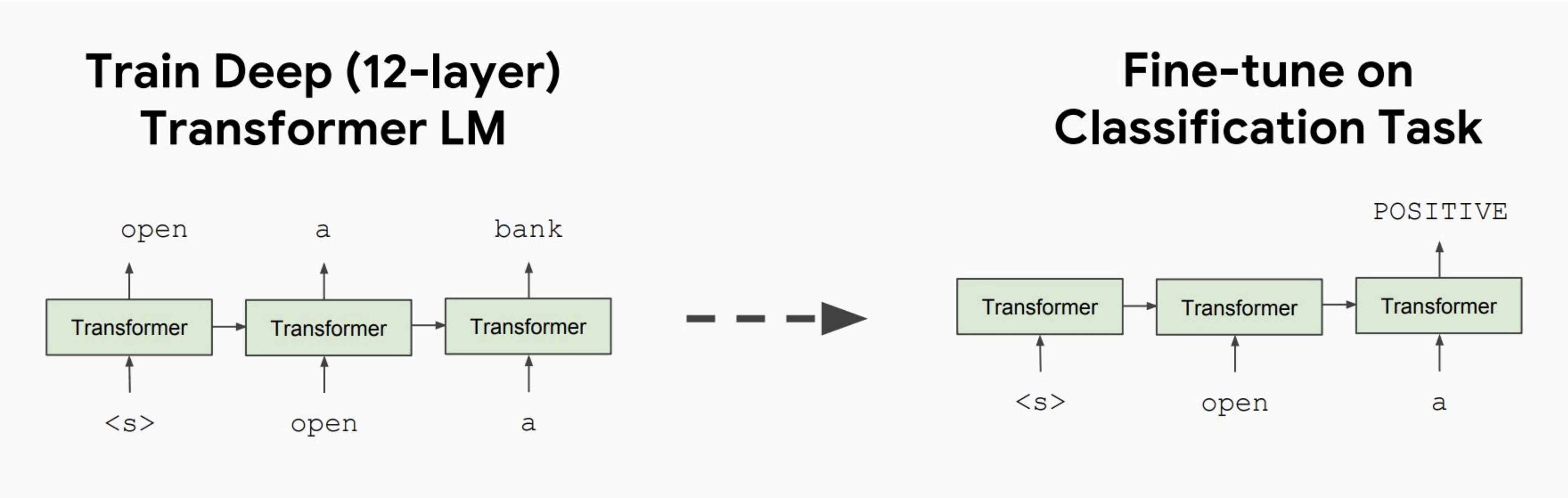
Seq2seq + fine-tune (SSL) - Google 2015

BERT



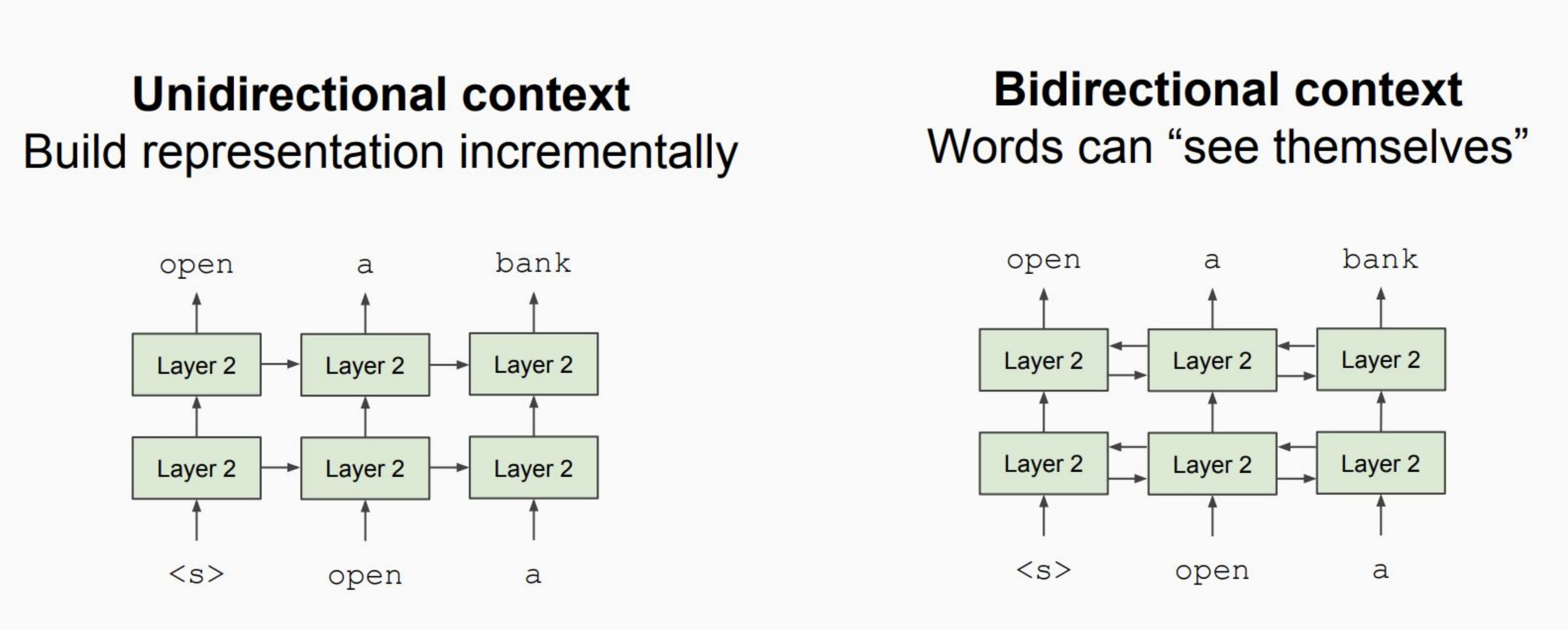
ELMO: University of Washington

BERT



OpenAI GPT-1: Alec Radford

BERT


Unidirectional context
Build representation incrementally
open
a
bank
Layer 2
Layer 2
Layer 2
Layer 2
Layer 2
Layer 2
<s>
open
a
Bidirectional context
Words can "see themselves"
open
a
bank
Layer 2
Layer 2
Layer 2
Layer 2
Layer 2
Layer 2
<s>
open
a

BERT

store gallon

the man went to the [MASK] to buy a [MASK] of milk

BERT - Masking

80%

went to the store → went to the [MASK]

10%

went to the store → went to the running

10%

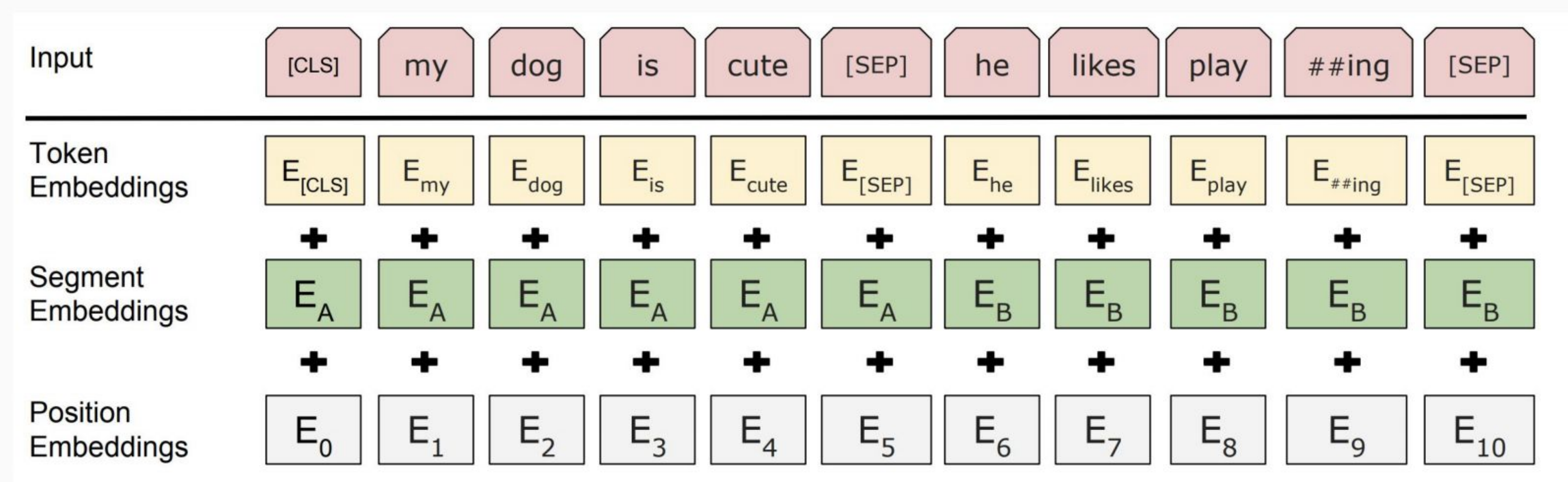
went to the store → went to the store

BERT

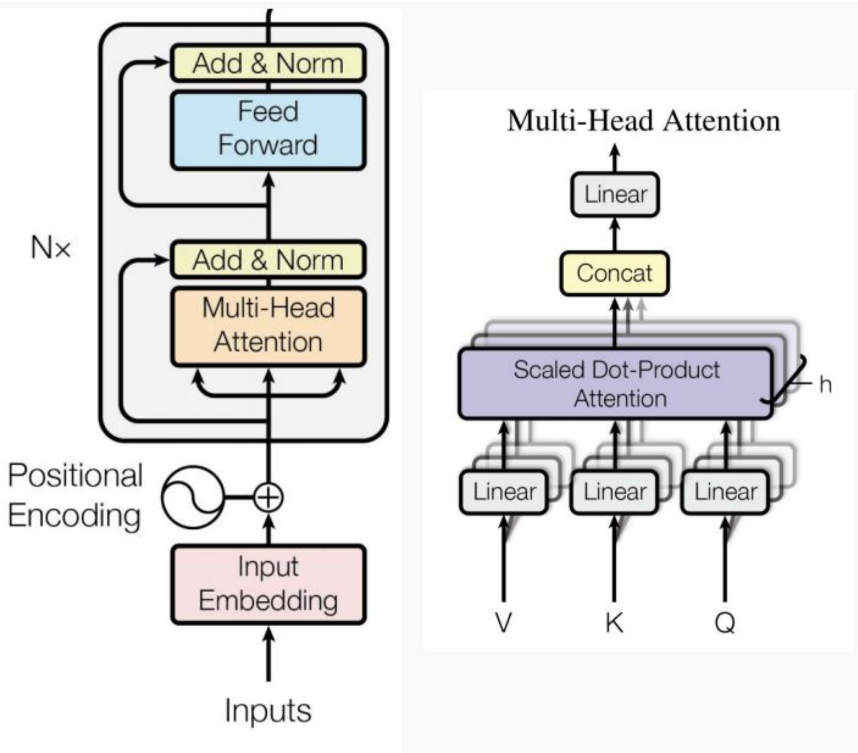
Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

BERT


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
E[CLS]
Emy
Edog
Eis
Ecute
E[SEP]
Ehe
Elikes
Eplay
E##ing
E[SEP]
Segment Embeddings
EA
EA
EA
EA
EA
EA
EB
EB
EB
EB
EB
Position Embeddings
E0
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10

BERT



Data: Wikipedia (2.5B words) + BookCorpus (800M words)

Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)

Training Time: 1M steps (~40 epochs)

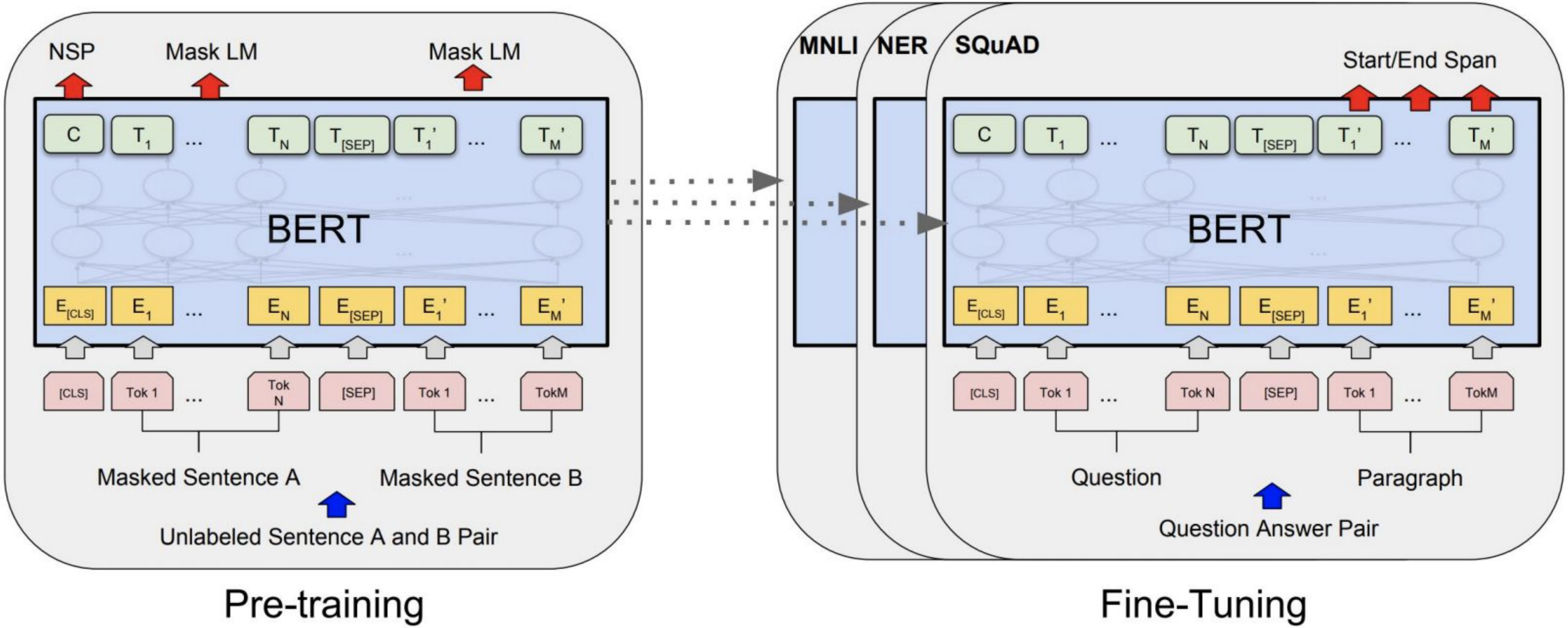
Optimizer: AdamW, 1e-4 learning rate, linear decay

`BERT-Base`: 12-layer, 768-hidden, 12-head

`BERT-Large`: 24-layer, 1024-hidden, 16-head

Trained on 4x4 or 8x8 TPU slice for 4 days

BERT


NSP
Mask LM
Mask LM
C
T_1
...
T_N
$T_{[SEP]}$
T_1'
...
T_M'
BERT
$E_{[CLS]}$
E_1
...
E_N
$E_{[SEP]}$
E_1'
...
E_M'
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
TokM
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair
Pre-training
MNLI
NER
SQuAD
Start/End Span
C
T_1
...
T_N
$T_{[SEP]}$
T_1'
...
T_M'
BERT
$E_{[CLS]}$
E_1
...
E_N
$E_{[SEP]}$
E_1'
...
E_M'
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
TokM
Question
Paragraph
Question Answer Pair
Fine-Tuning

BERT

GLUE Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

MultiNLI

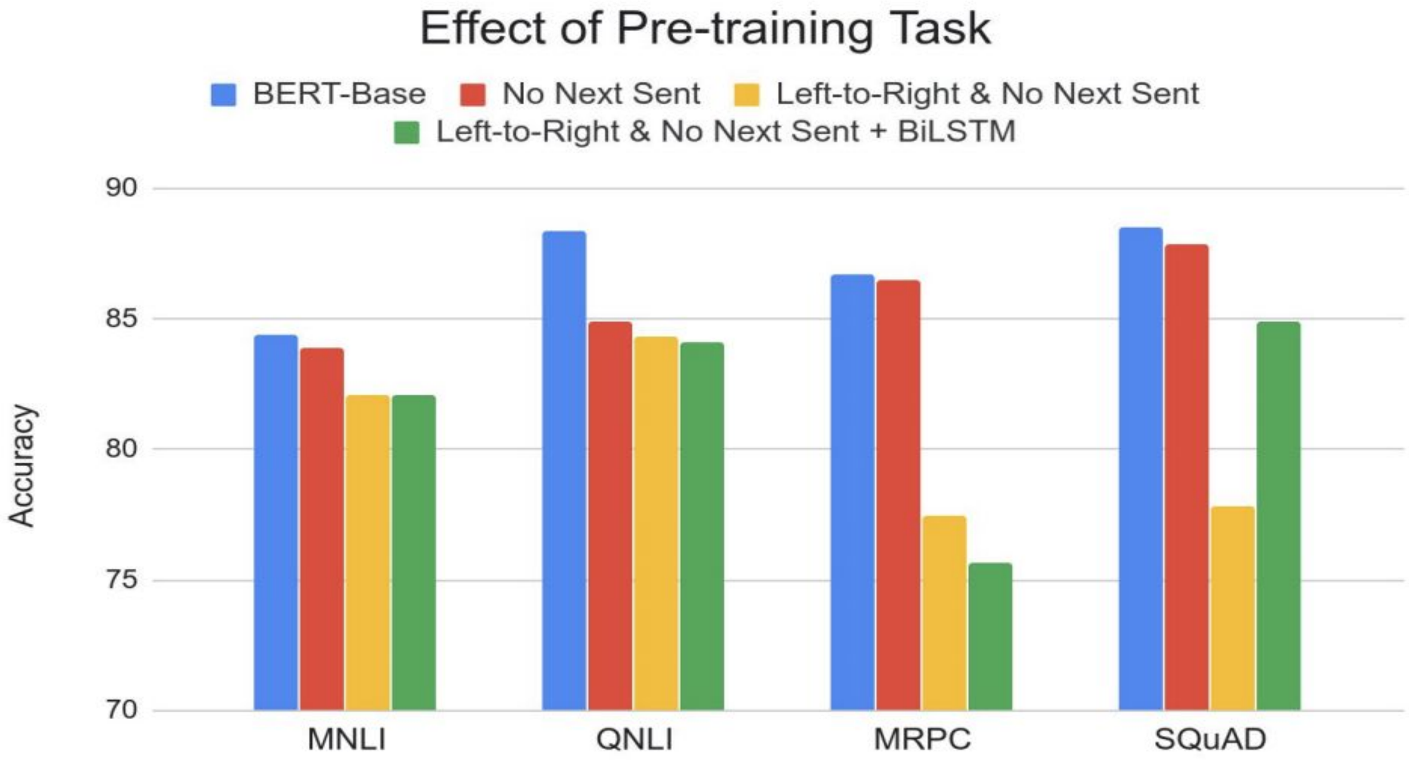
Premise: Hills and mountains are especially sanctified in Jainism.
Hypothesis: Jainism hates nature.
Label: Contradiction

CoLa

Sentence: The wagon rumbled down the road.
Label: Acceptable

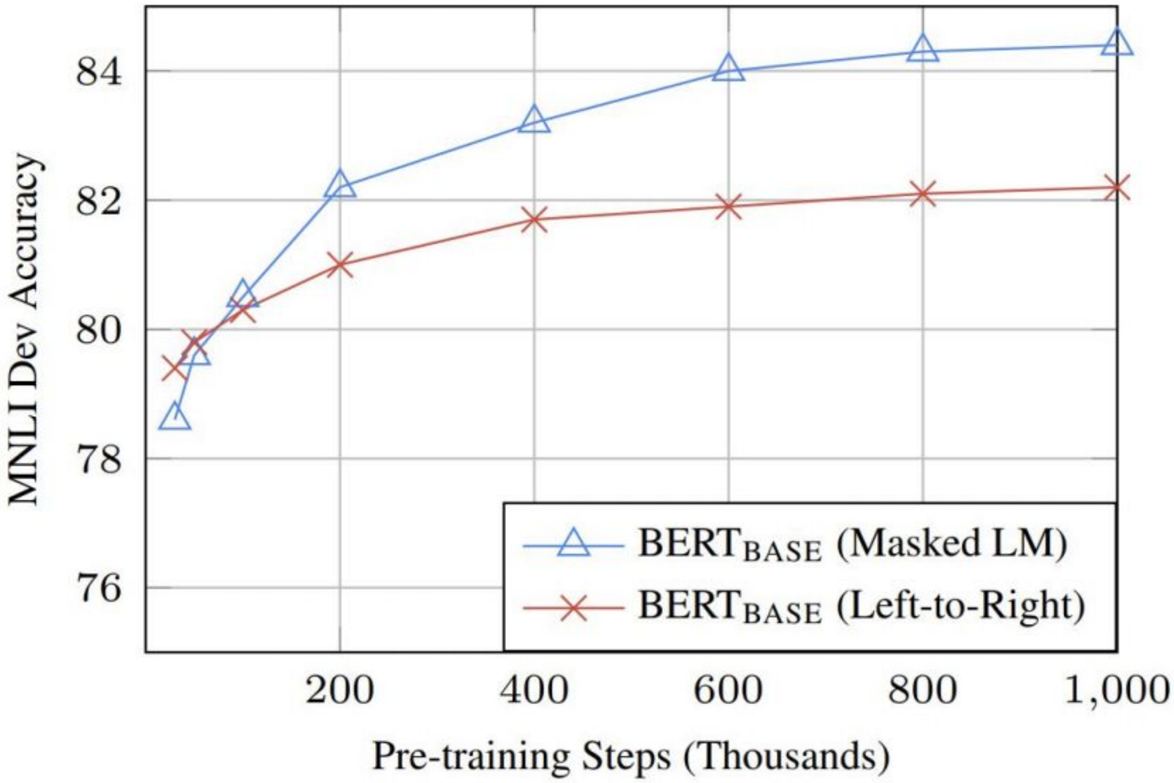
Sentence: The car honked down the road.
Label: Unacceptable

BERT



Effect of Pre-training Task
BERT-Base
No Next Sent
Left-to-Right & No Next Sent
Left-to-Right & No Next Sent + BiLSTM
Accuracy
90
85
80
75
70
MNLI
QNLI
MRPC
SQuAD

BERT



MNLI Dev Accuracy
84
82
80
78
76
200
400
600
800
1,000
Pre-training Steps (Thousands)
BERT$_{BASE}$ (Masked LM)
BERT$_{BASE}$ (Left-to-Right)

BERT

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6