CS 1678: Intro to Deep Learning
Advanced Topics

Prof. Adriana Kovashka
University of Pittsburgh
April 1, 2021

Plan for this lecture

- Alternative representations
  - I. Graph networks (pp 3-29)
- Alternative learning mechanisms
  - II. Self supervision (pp 30-69)
  - III. Reinforcement learning (pp 70-111)
- Alternative tasks
  - IV. Generation (pp 112-198)
- V. Bias and ethics (pp 199-257)

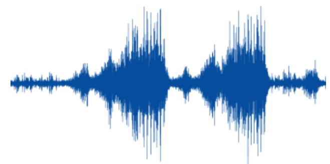
Part I: Graph Networks

- Types of graph networks
  - Graph convolutional networks
  - Graph attention networks
- Applications
  - Semi-supervised learning
  - Visual question answering

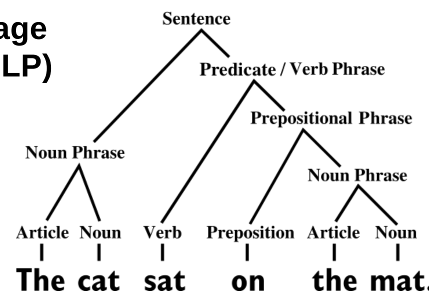
Types of data typically handled with Deep Learning

IMAGENET

Speech data

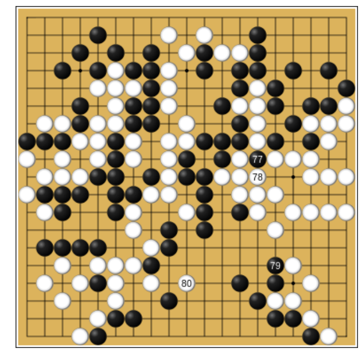


Natural language processing (NLP)



...

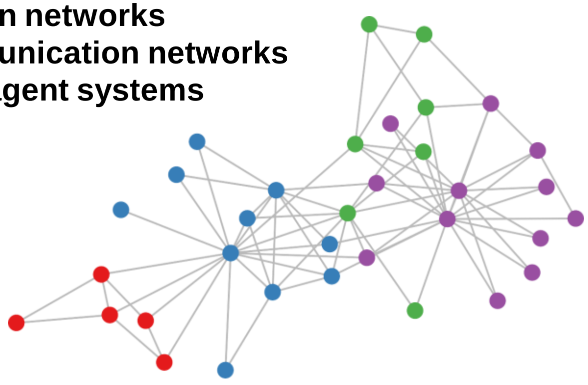
Grid games

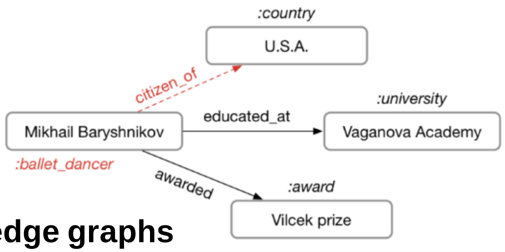


Graph-structured data

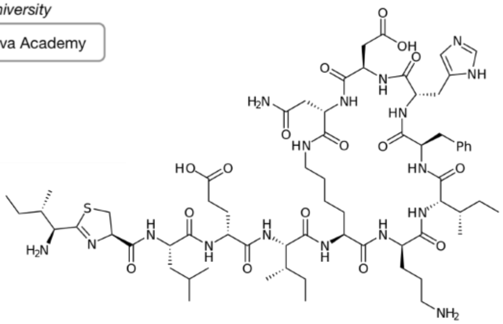
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

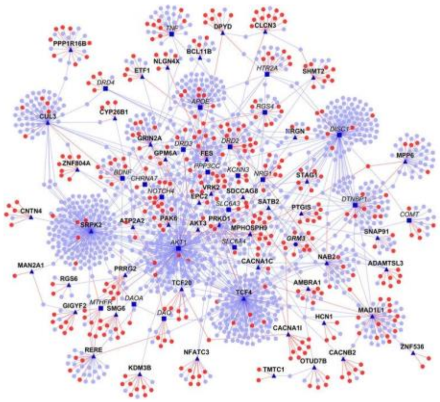




Knowledge graphs



Molecules



Protein interaction networks

Standard deep learning architectures like CNNs and RNNs don’t work here!

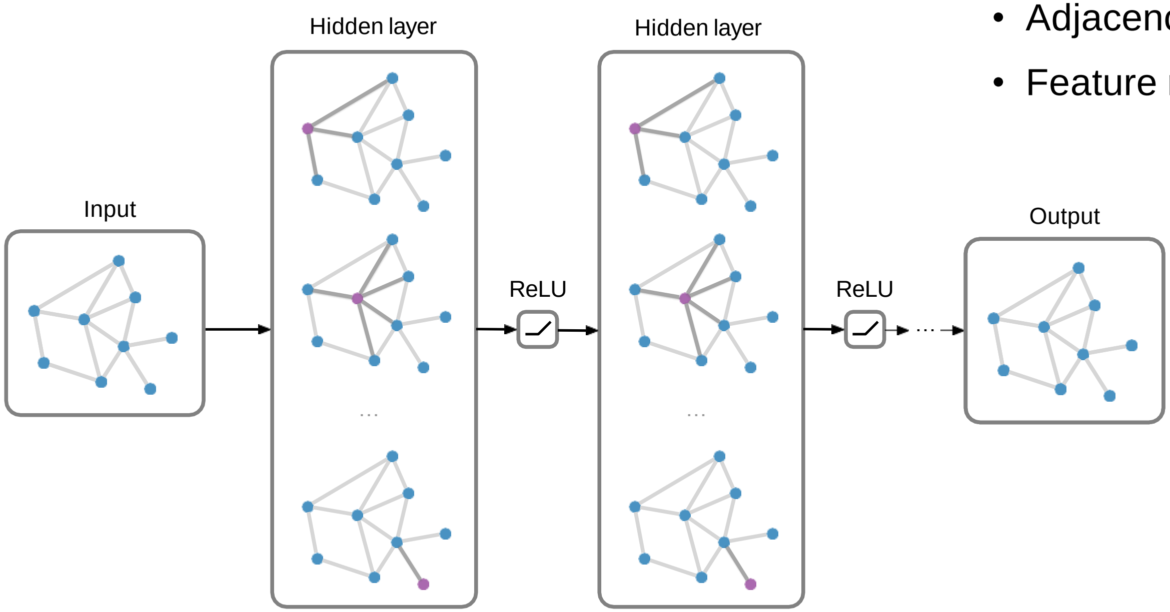
Road maps

Graph Neural Networks (GNNs)

The bigger picture:

Notation: $G = (\mathbf{A}, \mathbf{X})$

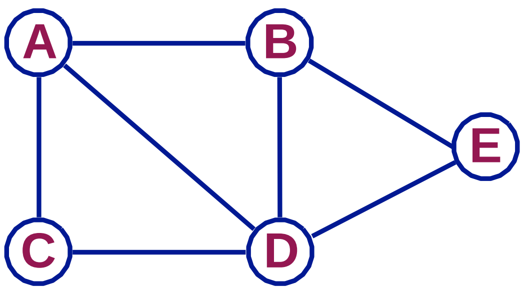
- Adjacency matrix $\mathbf{A} \quad \mathrm{R}^{N \times N}$
- Feature matrix $\mathbf{X} \quad \mathrm{R}^{N \times F}$



Main idea: Pass messages between pairs of nodes & agglomerate

Graph convolutional networks

Graph: $G = (\mathcal{V}, \mathcal{E})$



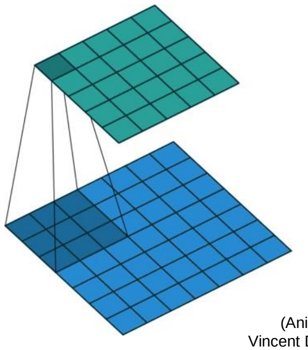
Adjacency matrix: A

$$
\begin{array}{c|c}
 & \begin{array}{ccccc} A & B & C & D & E \end{array} \\
\begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \left(\begin{array}{ccccc} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{array}\right)
\end{array}
$$

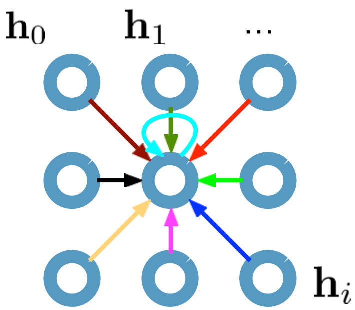
Kipf and Welling, “Semi-supervised learning with deep generative models”, ICLR 2017 (slides by Thomas Kipf)

Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

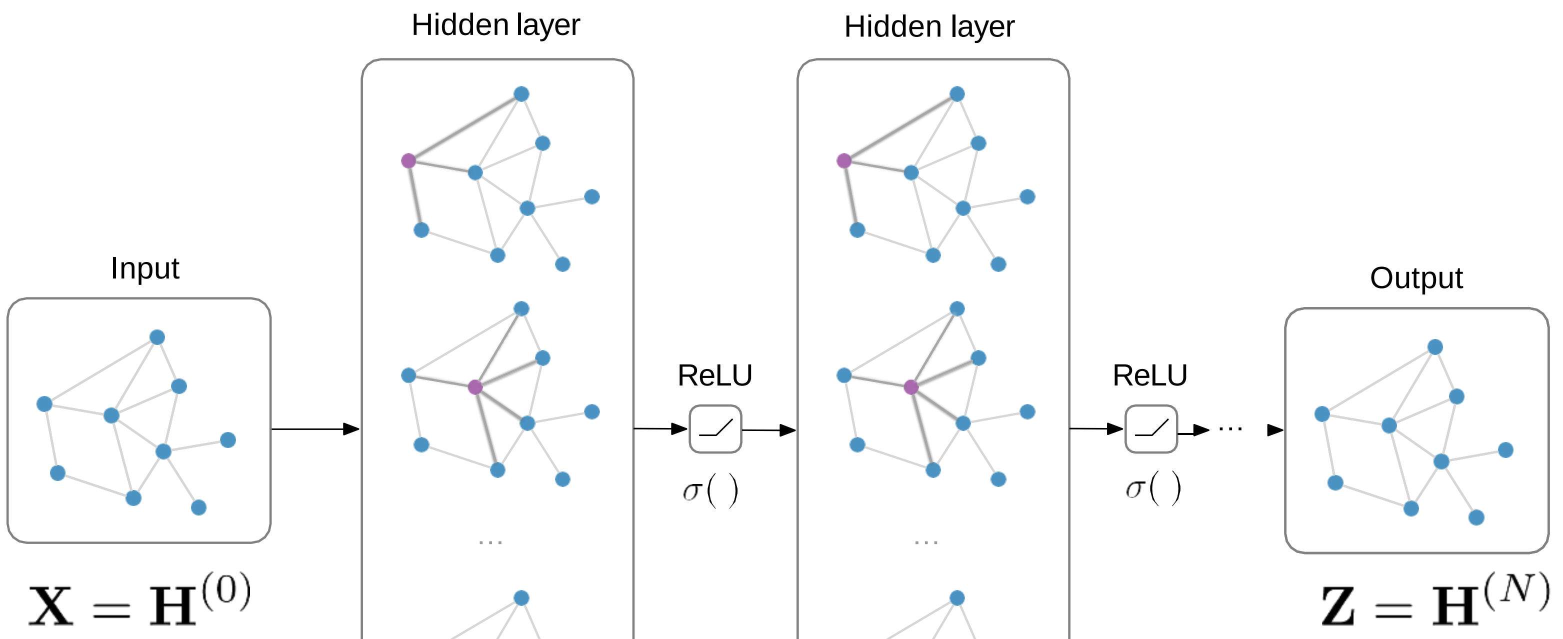
$\mathbf{h}_i$ *in* R^F are (hidden layer) activations of a pixel/node

Full update:

$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

Graph convolutional networks

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



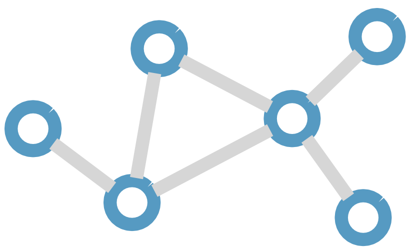
$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Kipf and Welling, "Semi-supervised learning with deep generative models", ICLR 2017 (slides by Thomas Kipf)

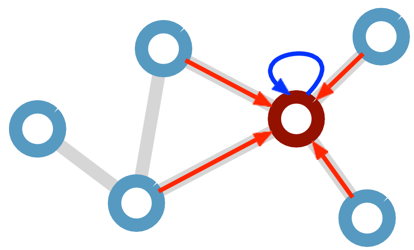
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

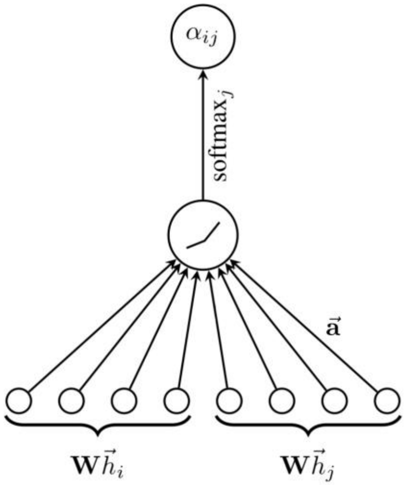
Scalability: subsample messages [Hamilton et al., NIPS 2017]

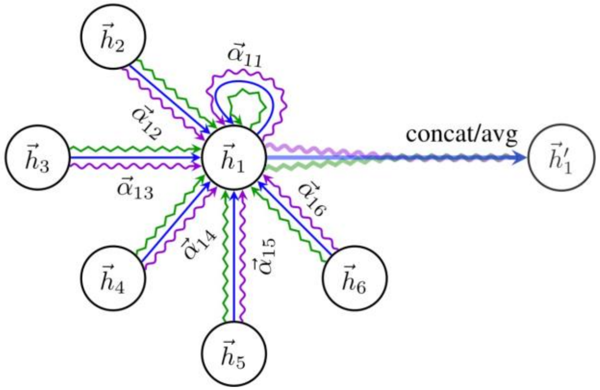
$\mathcal{N}_i$: neighbor indices

c_{ij} : norm. constant (fixed/trainable)

Graph neural networks with attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)

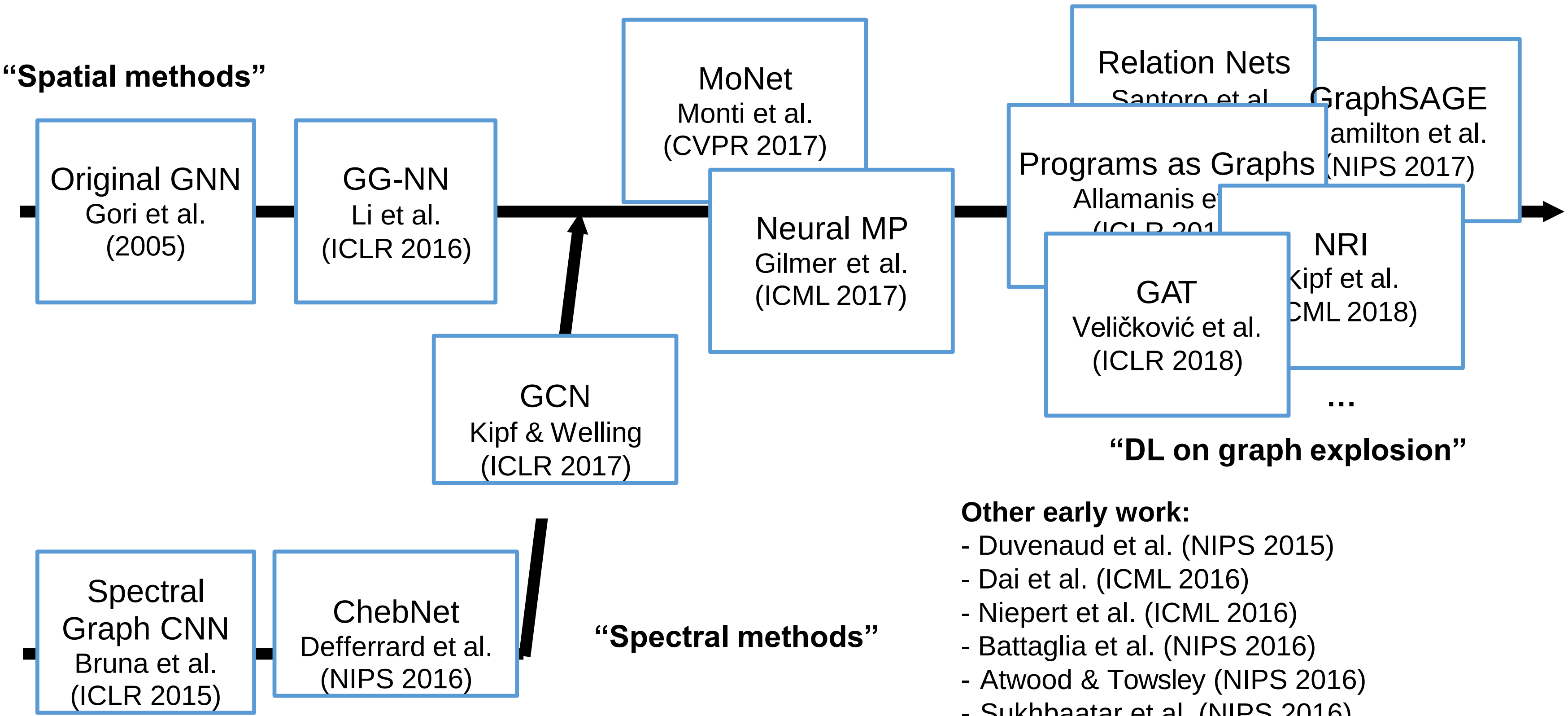




[Figure from Veličković et al. (ICLR 2018)]

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right) \qquad \alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_k]\right)\right)}$$

A brief history of graph neural nets



Other early work:

- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

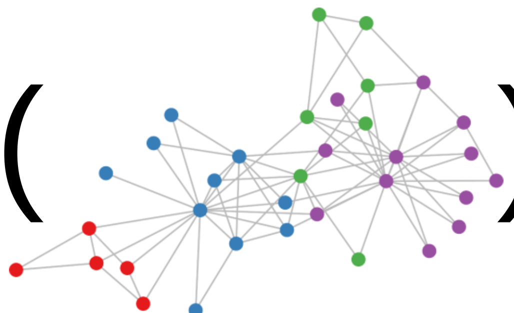
(slide inspired by Alexander Gaunt's talk on GNNs)

What do learned representations look like?

Forward pass through **untrained** 3-layer GCN model

Parameters initialized randomly

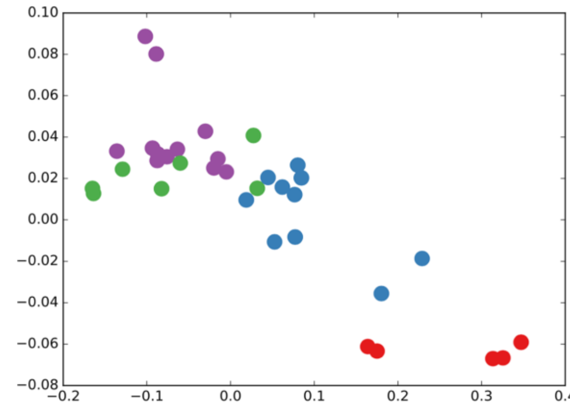
f(



) =

[Zachary's Karate Club]

2-dim output per node



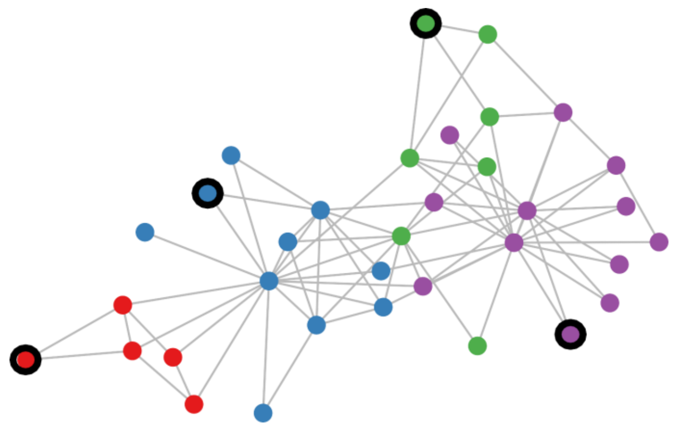
What else are graph representations good for?

Semi-supervised classification on graphs

Setting:

Some nodes are labeled (black circle)
All other nodes are unlabeled

Task:
Predict node label of unlabeled nodes



Evaluate loss on labeled nodes only:

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

$\mathcal{Y}_L$ set of labeled node

$\mathbf{Y}$ indices label matrix

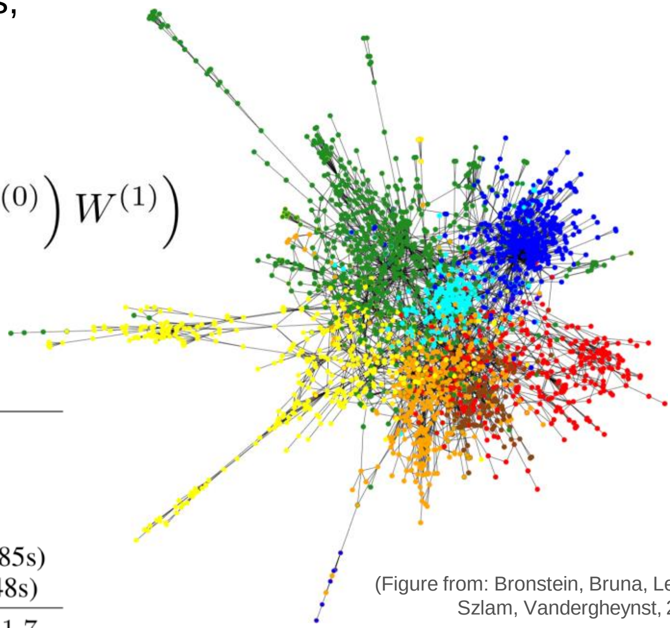
$\mathbf{Z}$ GCN output (after softmax)

Application: Classification on citation networks

Input: Citation networks (nodes are papers, edges are citation links, optionally bag-of-words features on nodes)

Target: Paper category (e.g. stat.ML, cs.LG, …)

Model: 2-layer GCN $Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\, W^{(1)}\Big)$



(Figure from: Bronstein, Bruna, LeCun, Szlam, Vandergheynst, 2016)

Classification results (accuracy)

no input features

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [24]	59.6	59.0	71.1	26.7
LP [27]	45.3	68.0	63.0	26.5
DeepWalk [18]	43.2	67.2	65.3	58.1
Planetoid* [25]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

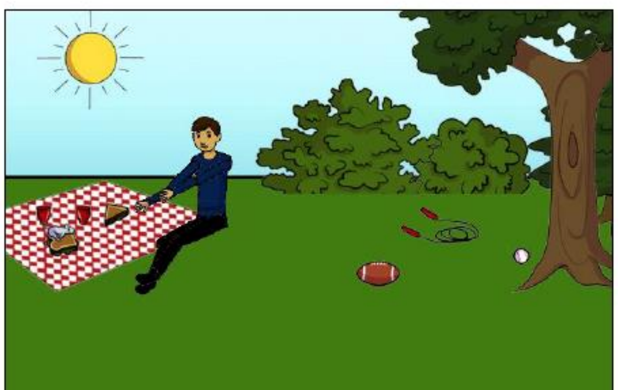
Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Visual Question Answering (VQA)

Task: Given an image and a natural language open-ended question, generate a natural language answer.

What color are her eyes?
What is the mustache made of?

How many slices of pizza are there?
Is this a vegetarian pizza?

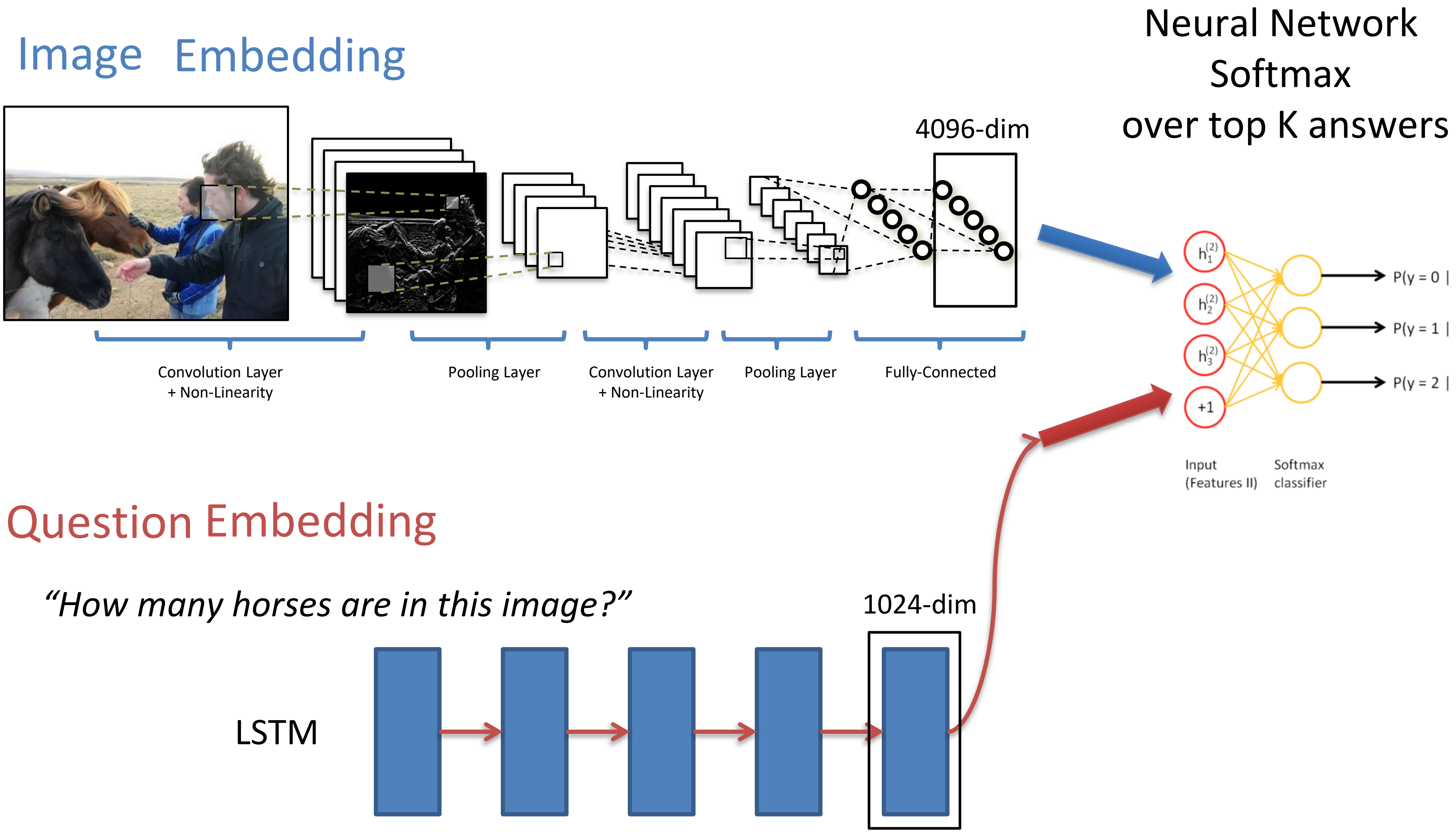


Is this person expecting company?
What is just under the tree?

Does it appear to be rainy?
Does this person have 20/20 vision?

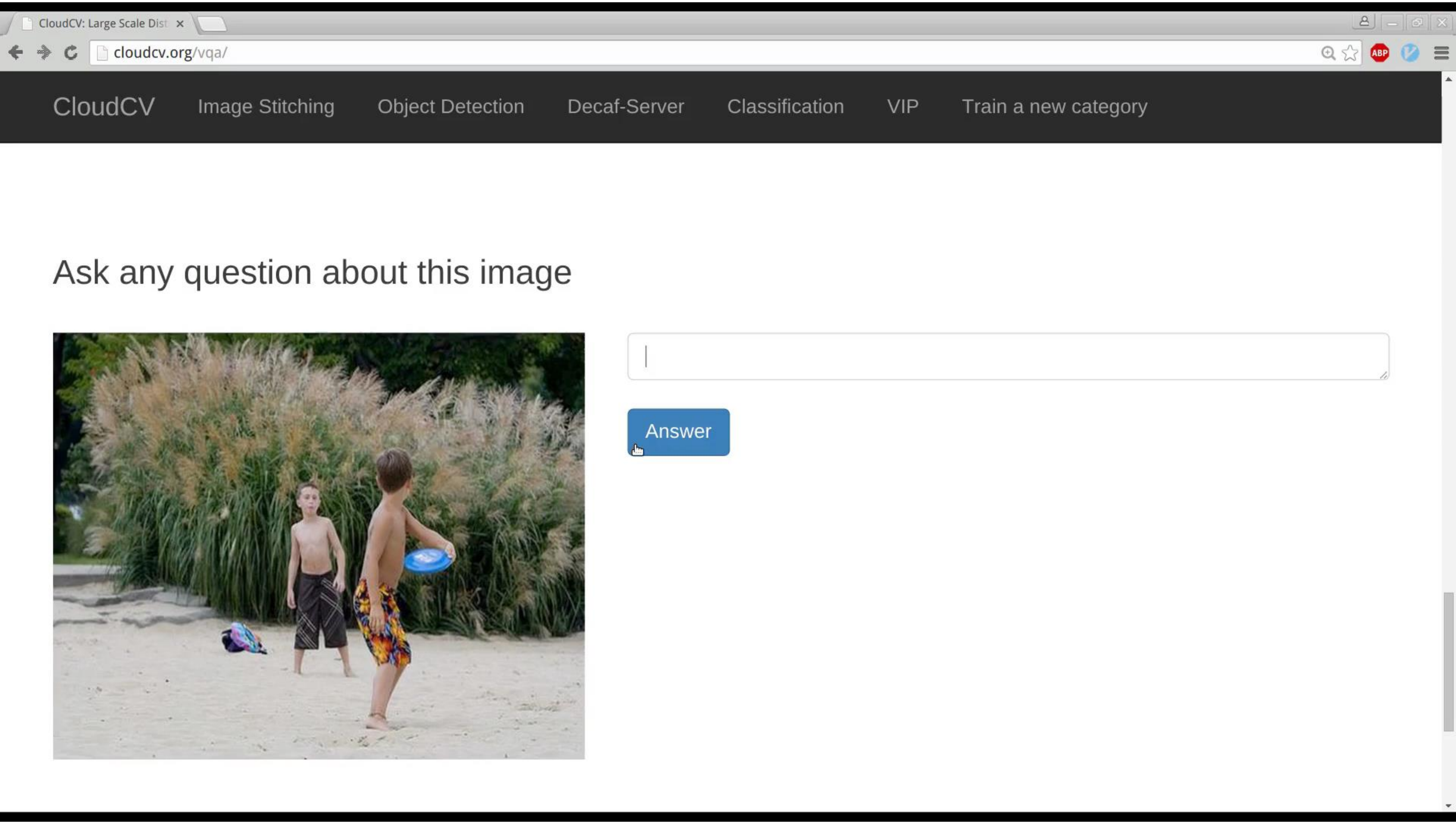
Agrawal et al., "[VQA: Visual Question Answering]", ICCV 2015

Visual Question Answering (VQA)



Agrawal et al., "[VQA: Visual Question Answering](#)", ICCV 2015

Visual Question Answering (VQA)



Agrawal et al., “VQA: Visual Question Answering”, ICCV 2015

Visual Question Answering (VQA)

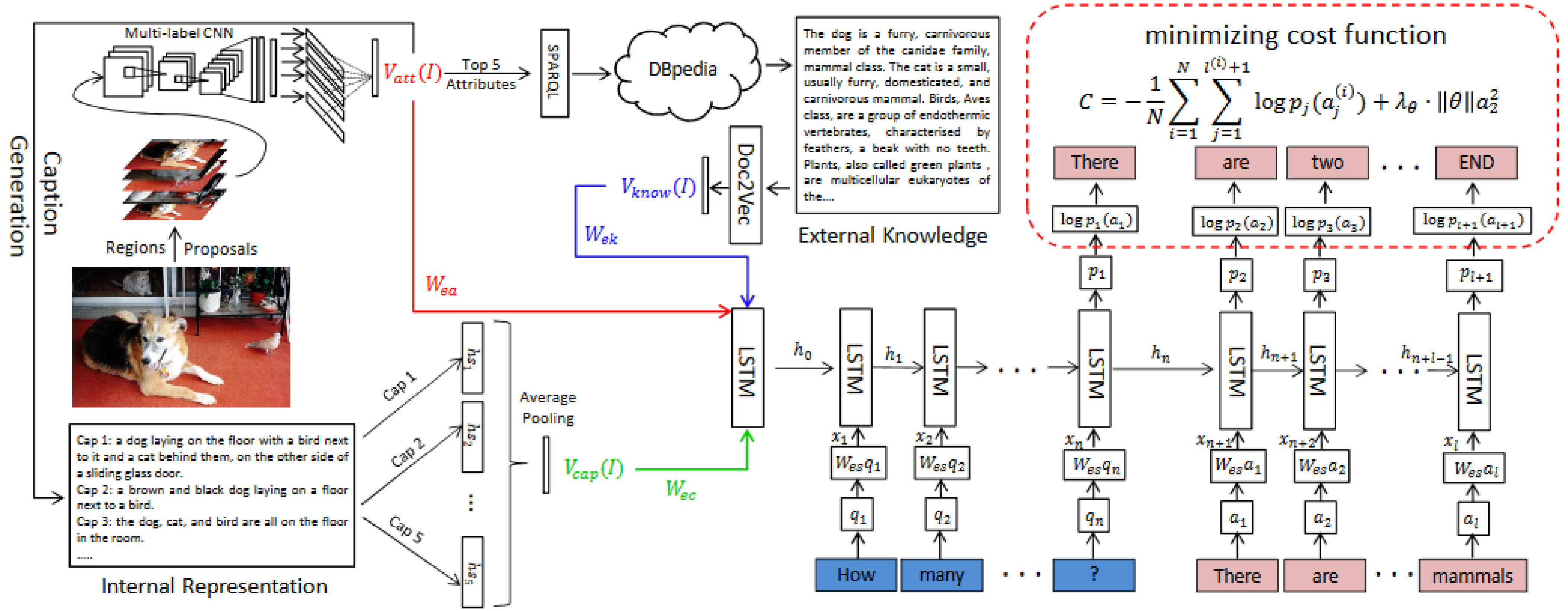


Figure 2. Our proposed framework: given an image, a CNN is first applied to produce the attribute-based representation $V_{att}(I)$. The internal textual representation is made up of image captions generated based on the image-attributes. The hidden state of the caption-LSTM after it has generated the last word in each caption is used as its vector representation. These vectors are then aggregated as $V_{cap}(I)$ with average-pooling. The external knowledge is mined from the KB (in this case DBpedia) and the responses encoded by Doc2Vec, which produces a vector $V_{know}(I)$. The 3 vectors V are combined into a single representation of scene content, which is input to the VQA LSTM model which interprets the question and generates an answer.

Wu et al., “Ask Me Anything: Free-Form Visual Question Answering Based on Knowledge From External Sources”, CVPR 2016

Reasoning for VQA

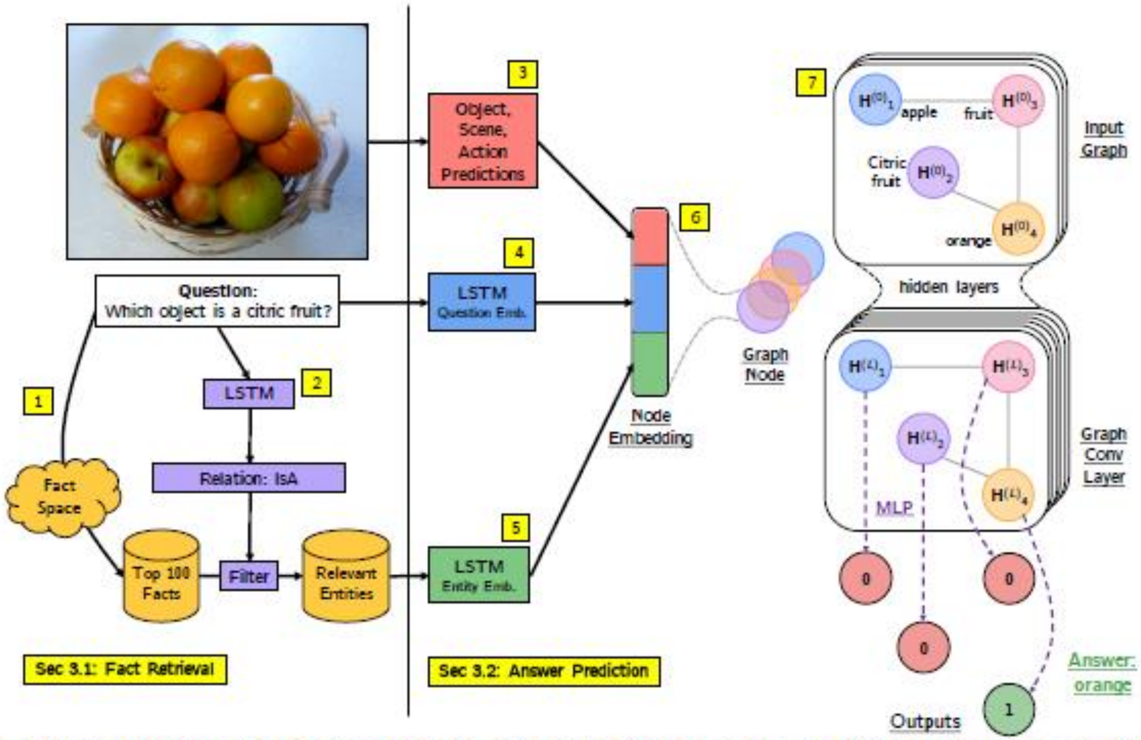


Figure 2: Outline of the proposed approach: Given an image and a question, we use a similarity scoring technique (1) to obtain relevant facts from the fact space. An LSTM (2) predicts the relation from the question to further reduce the set of relevant facts and its entities. An entity embedding is obtained by concatenating the visual concepts embedding of the image (3), the LSTM embedding of the question (4), and the LSTM embedding of the entity (5). Each entity forms a single node in the graph and the relations constitute the edges (6). A GCN followed by an MLP performs joint assessment (7) to predict the answer. Our approach is trained end-to-end.

Narasimhan and Schwing, “Out of the Box: Reasoning with Graph Convolution Nets for Factual Visual Question Answering”, NeurIPS 2018

Graphs for advertisements

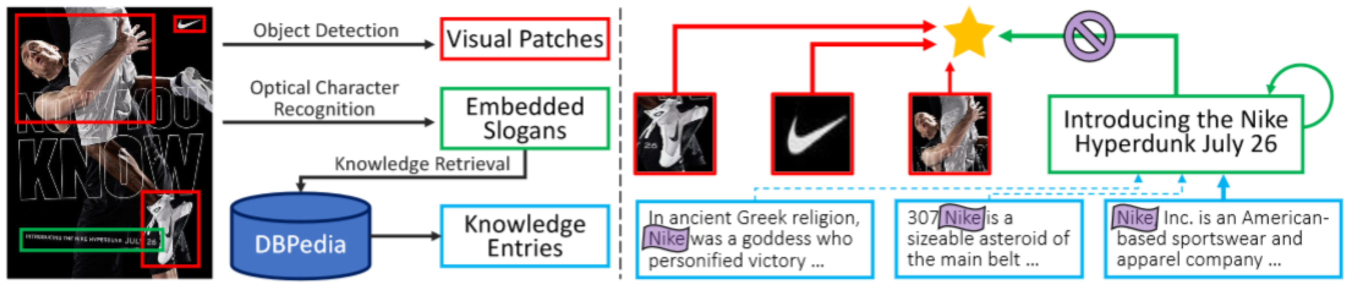
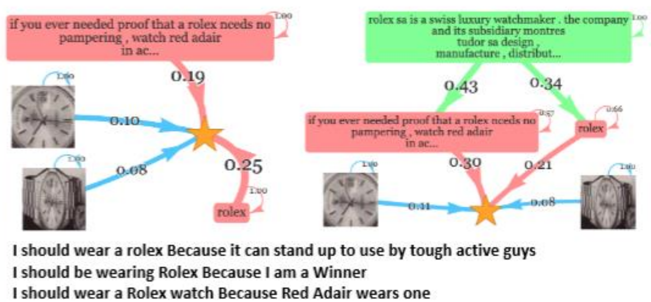


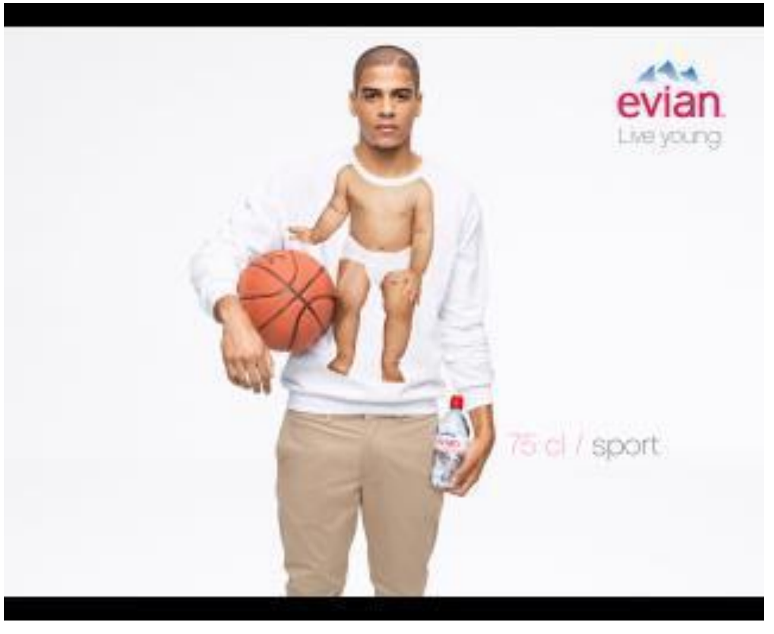
Figure 2: **Overview of the proposed model.** Given a single image ad, we first expand the representation using object detection and OCR, and also retrieve relevant knowledge based on slogan snippets (left). We build a graph-based model to infer the overall message using all available information (right). For more effective training, we mask query keywords and randomly drop certain knowledge pieces (shown in purple). More details are in Sec. 3.



Ye, Zhang and Kovashka

Decoding image advertisements

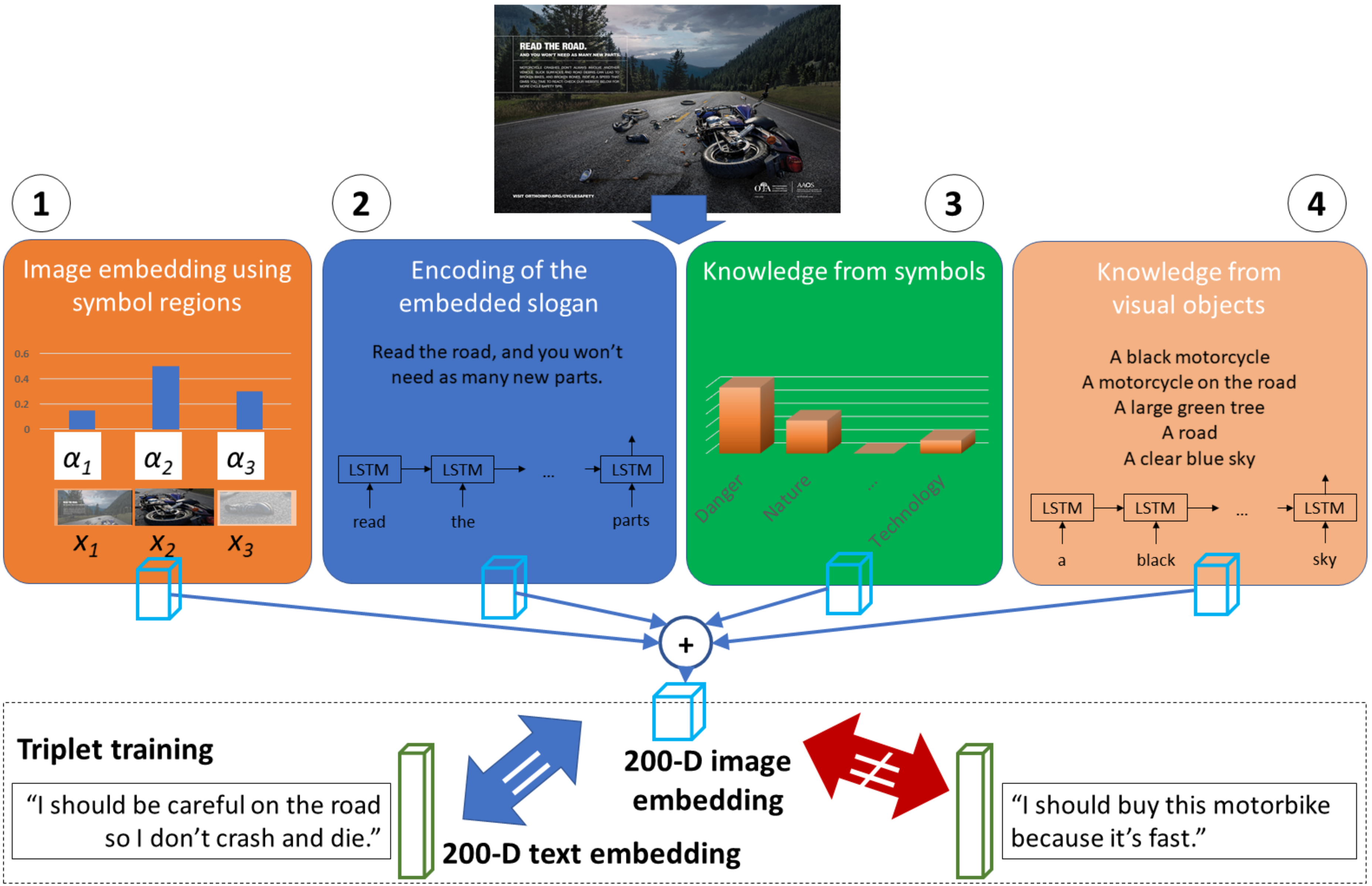
- What message does the ad convey (*action*), and what arguments does it provide for taking the suggested action (*reason*)?
- Multiple-choice task: Given *k* options for action-reason statements, pick one that matches the image



- I should drink evian because it helps you recover
- I should drink Evian because it will keep me like a baby
- I should buy Evian because it keeps us young

Hussain, Zhang, Zhang, Ye, Thomas, Agha, Ong and Kovashka, CVPR 2017

Retrieve the best action-reason statement



Ye et al., TPAMI 2019

Experimental results (image features only)

- We outperform prior art by a large margin, for both statement ranking and classification

	Rank (Lower ↓ is better)		Recall@3 (Higher ↑ is better)	
Method	PSA	Product	PSA	Product
2-WAY NETS	4.836 (± 0.090)	4.170 (± 0.023)	0.923 (± 0.016)	1.212 (± 0.004)
VSE	4.155 (± 0.091)	3.202 (± 0.019)	1.146 (± 0.017)	1.447 (± 0.004)
VSE++	4.139 (± 0.094)	3.110 (± 0.019)	1.197 (± 0.017)	1.510 (± 0.004)
HUSSAIN-RANKING	*3.854* (± 0.088)	*3.093* (± 0.019)	*1.258* (± 0.017)	*1.515* (± 0.004)
ADVISE (ours)	**3.013** (± 0.075)	**2.469** (± 0.015)	**1.509** (± 0.017)	**1.725** (± 0.004)

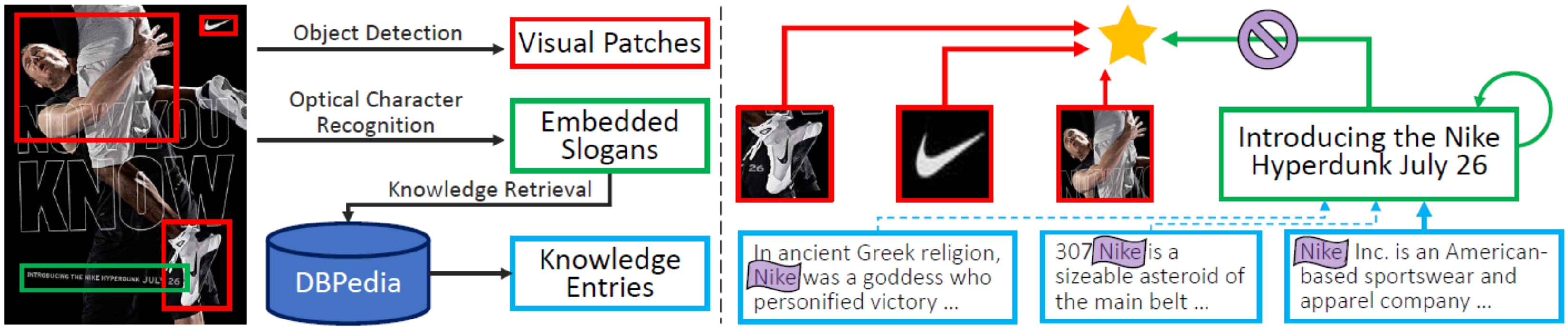
- Our methods accurately capture the rhetoric, even in deliberately confusing ads

VSE++ on Ads: I should wear Revlon makeup because it will make me more attractive"

ADVISE (ours): "I should stop smoking because it doesn't make me pretty"

Ye and Kovashka, ECCV 2018

Incorporating external knowledge



- Expand image representation using external knowledge (from DBPedia); represent regions, slogans, KB nuggets in a graph
- To prevent overfitting and break non-generalizable shortcuts, we randomly mask parts of training samples (e.g. slogan, words in KB nugget)

Ye, Zhang and Kovashka

Incorporating external knowledge

- Training via metric learning: match image to human-annotated action-reason statements
- Image representation is a graph
- Slogan node updates:

$$\mathbf{t}_i^{(1)} = \underbrace{\alpha_{i,0}\mathbf{t}_i^{(0)}}_{\text{original meaning}} + \underbrace{\sum_{j=1}^{|\phi(t_i)|} \alpha_{i,j}\mathbf{k}_{i,j}}_{\text{descriptions from extra knowledge}}$$

- Global node update:

$$\mathbf{h} = \underbrace{\sum_{i=1}^{|V|} \beta_i \mathbf{v}_i}_{\text{messages from proposals}} + \underbrace{\sum_{i=|V|+1}^{|V|+|T|} \beta_i \mathbf{t}_i^{(1)}}_{\text{messages from slogans}}$$

- Edge weights α, β allow model to choose what knowledge to use

Ye, Zhang and Kovashka

Incorporating external knowledge

- We stochastically mask aspects of training data, to prevent model from relying too much on word-matching or object-matching
- Three strategies; can also learn how to mask:
  - M_t randomly drops a detected textual (T) slogan, with a probability of 0.5
  - M_s randomly sets the KB query words (e.g. "WWF" or "Nike") in the human-annotated statements (S) to the out-of-vocabulary token, with probability 0.5
  - M_k replaces the DBpedia queries in the retrieved knowledge contents with the out-of-vocabulary token

Ye, Zhang and Kovashka

Incorporating external knowledge

- Outperform prior state of the art

Methods	Accuracy (%)
VSE [31]	62.0
ADNET [6]	65.0
ADVISE [31]	69.0
CYBERAGENT [18]	82.0
RHETORIC [32]	83.3
OURS	**87.3**

- Using external knowledge helps when data masked

Method	P@1	P@3	P@5	P@10	R@1	R@3	R@5	R@10	Min Rank	Avg Rank	Med Rank
Results on the Challenge-15 task											
V,T	**87.3**	76.6	55.1	30.6	**28.4**	74.2	87.9	97.5	1.26	3.02	2.77
V,T+K	**87.3**	76.6	55.1	30.6	**28.4**	74.3	87.9	97.6	1.25	3.02	2.77
V,T+K(M_t,M_s,M_k)	**87.3**	**77.5**	**55.9**	**30.8**	**28.4**	**75.2**	**89.2**	**98.2**	**1.23**	**2.91**	**2.69**
Results on the Sampled-100 task											
V,T	79.8	66.5	46.9	26.2	26.0	64.4	74.9	83.5	2.38	7.52	5.86
V,T+K	80.0	67.0	47.0	26.1	26.0	64.9	75.1	83.4	2.29	7.49	5.81
V,T+K(M_t,M_s,M_k)	**80.2**	**67.9**	**47.9**	**26.8**	**26.1**	**65.8**	**76.6**	**85.4**	**2.14**	**6.56**	**5.19**
Results on the Sampled-500 task											
V,T	**65.5**	52.3	37.8	21.7	**21.3**	50.5	60.4	69.0	8.18	30.1	21.6
V,T+K	65.4	52.3	38.0	21.9	**21.3**	50.6	60.7	69.6	7.60	30.0	21.4
V,T+K(M_t,M_s,M_k)	64.8	**52.4**	**38.3**	**22.1**	21.1	**50.7**	**61.1**	**70.6**	**6.89**	**25.1**	**18.2**

Ye, Zhang and Kovashka

Incorporating external knowledge

Quantitatively: Without masking we retrieve relevant info with accuracy 25%, vs 54% with masking.

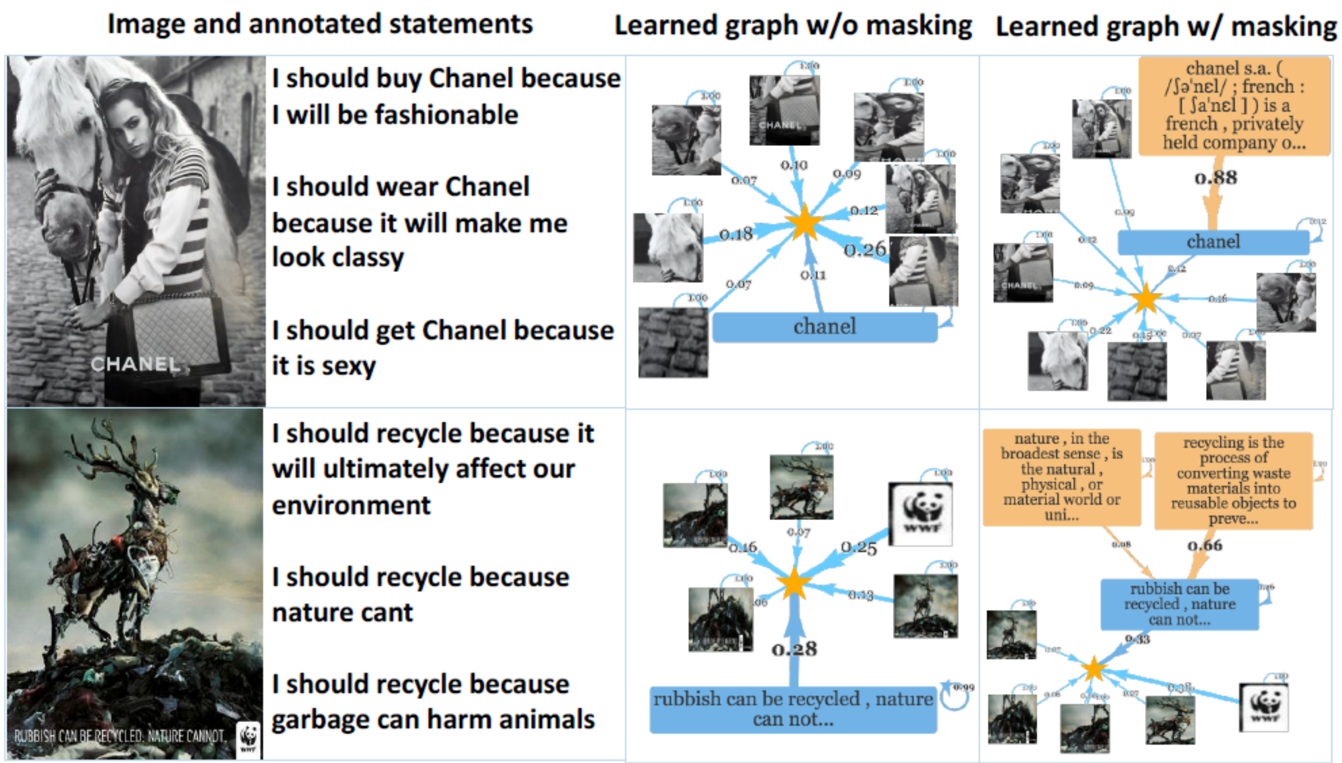


Fig. 4: **Examples of the learned graphs (best with zoom).** We show the ad image and annotated action-reason statements on the left, the graph learned without masking in the middle, and that learned with masking (our approach) on the right. We show slogans in blue, DBpedia comments in orange, and the global node as a star. **Arrow thickness is correlated with learned weights α, β.** For visualization we removed all edges with small weights (threshold=0.05). We see our method more effectively leverages external information.

Ye, Zhang and Kovashka

Part II: Self-Supervised Learning

- Learn representations from context in raw data
- Language – predict nearby words [*already covered]*
  - Word2Vec
  - Transformers, BERT
- Vision – predict pixels from other pixels
  - Predict nearby patches in an image
  - Predict order of frames in a video
  - Predict what you will see as you move
  - Predict physics

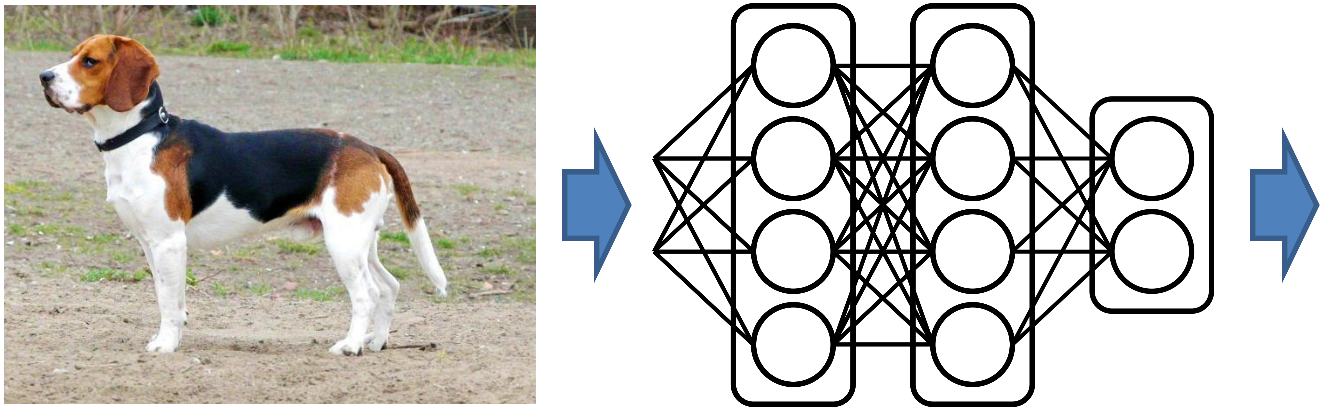
Jitendra Malik: "**Supervision** is the opium of the AI researcher"
Alyosha Efros: "The AI revolution will not be **supervised**"
Yann LeCun: “**Self-supervised** learning is the cake, **supervised** learning is the icing on the cake, **reinforcement learning** is the cherry on the cake"

Unsupervised Visual Representation Learning by Context Prediction

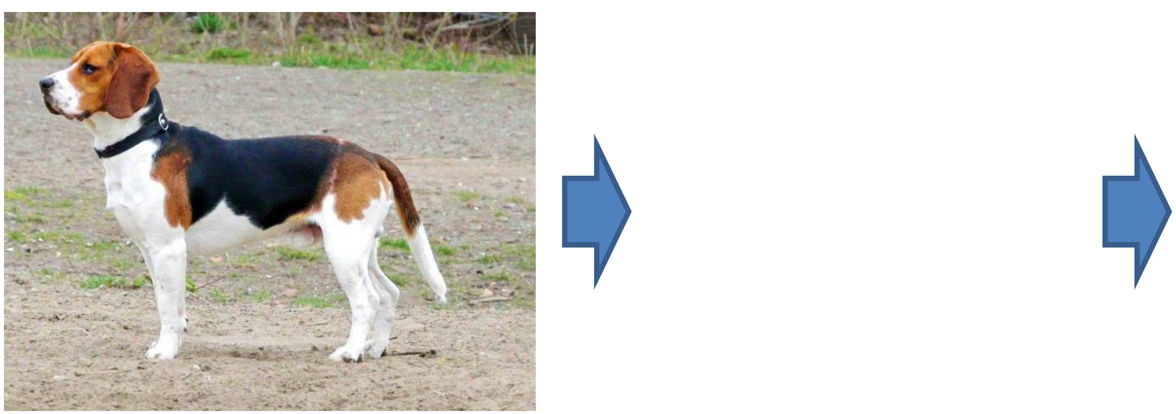
Carl Doersch, Alexei Efros and Abhinav Gupta

ICCV 2015

ImageNet + Deep Learning



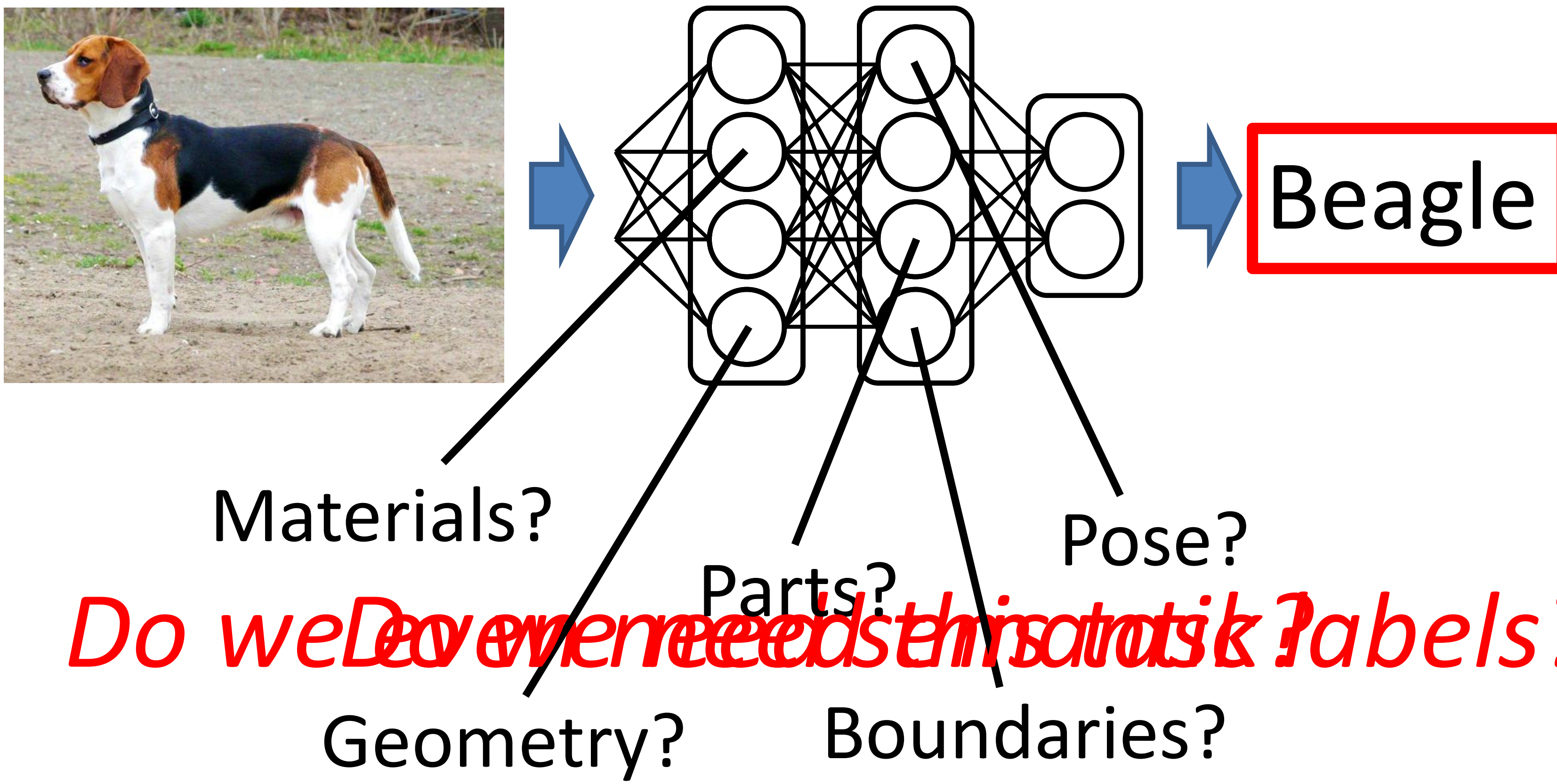
Beagle



- Image Retrieval
- Detection (RCNN)
- Segmentation (FCN)
- Depth Estimation
- ...

Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

ImageNet + Deep Learning



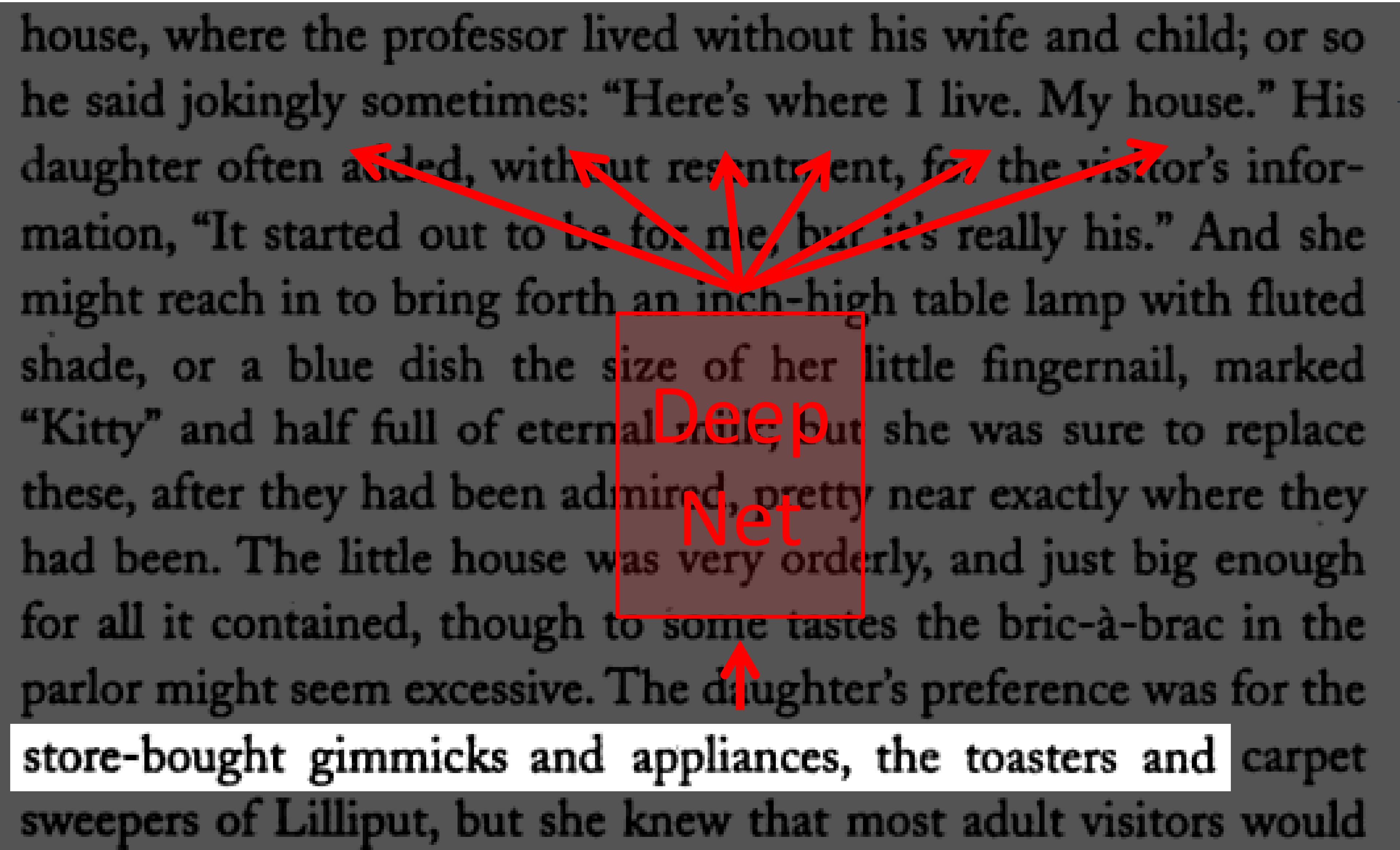
Do we need semantic labels?

Do we even need this task?

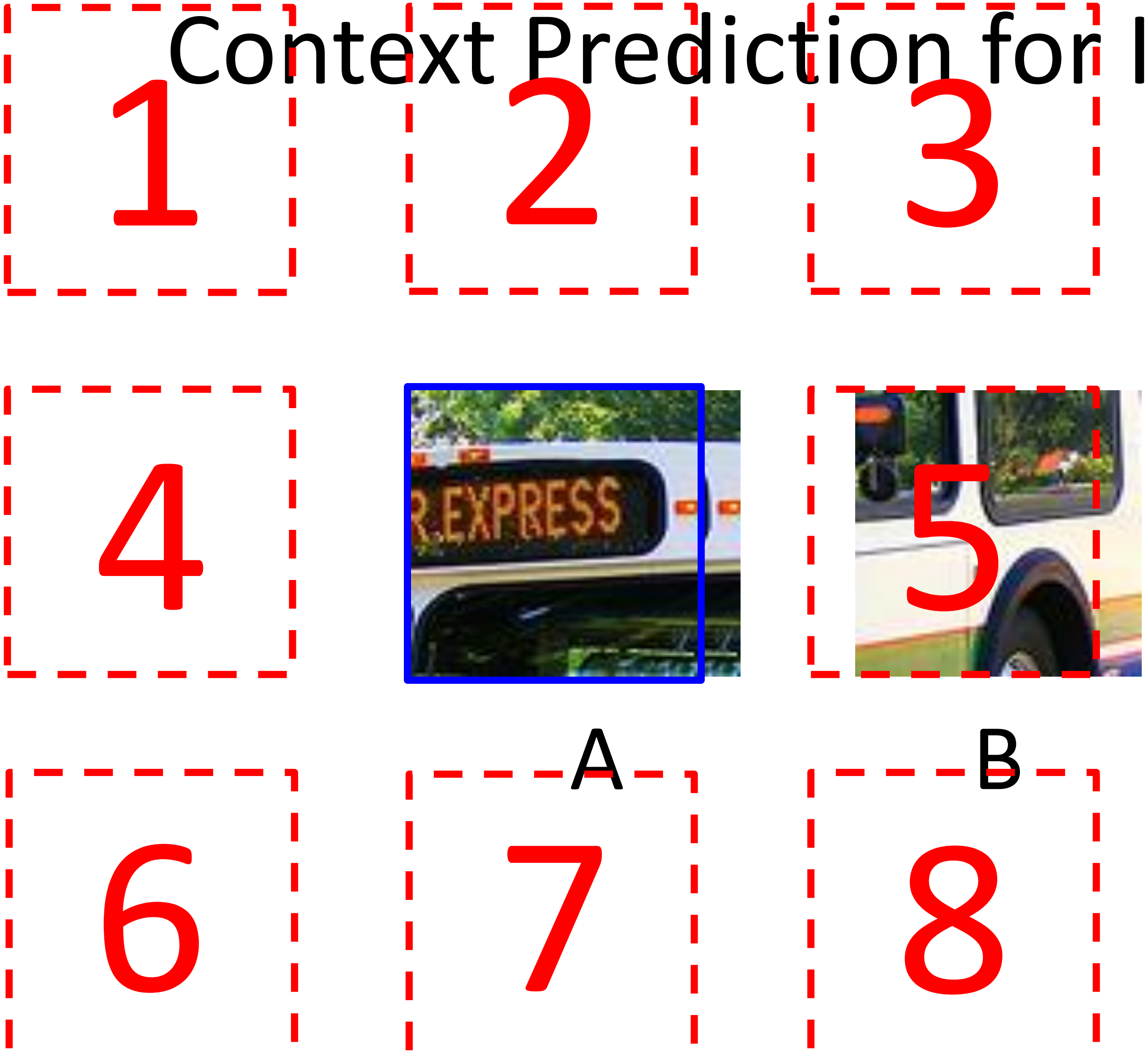
Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Context as Supervision

[Collobert & Weston 2008; Mikolov et al. 2013]

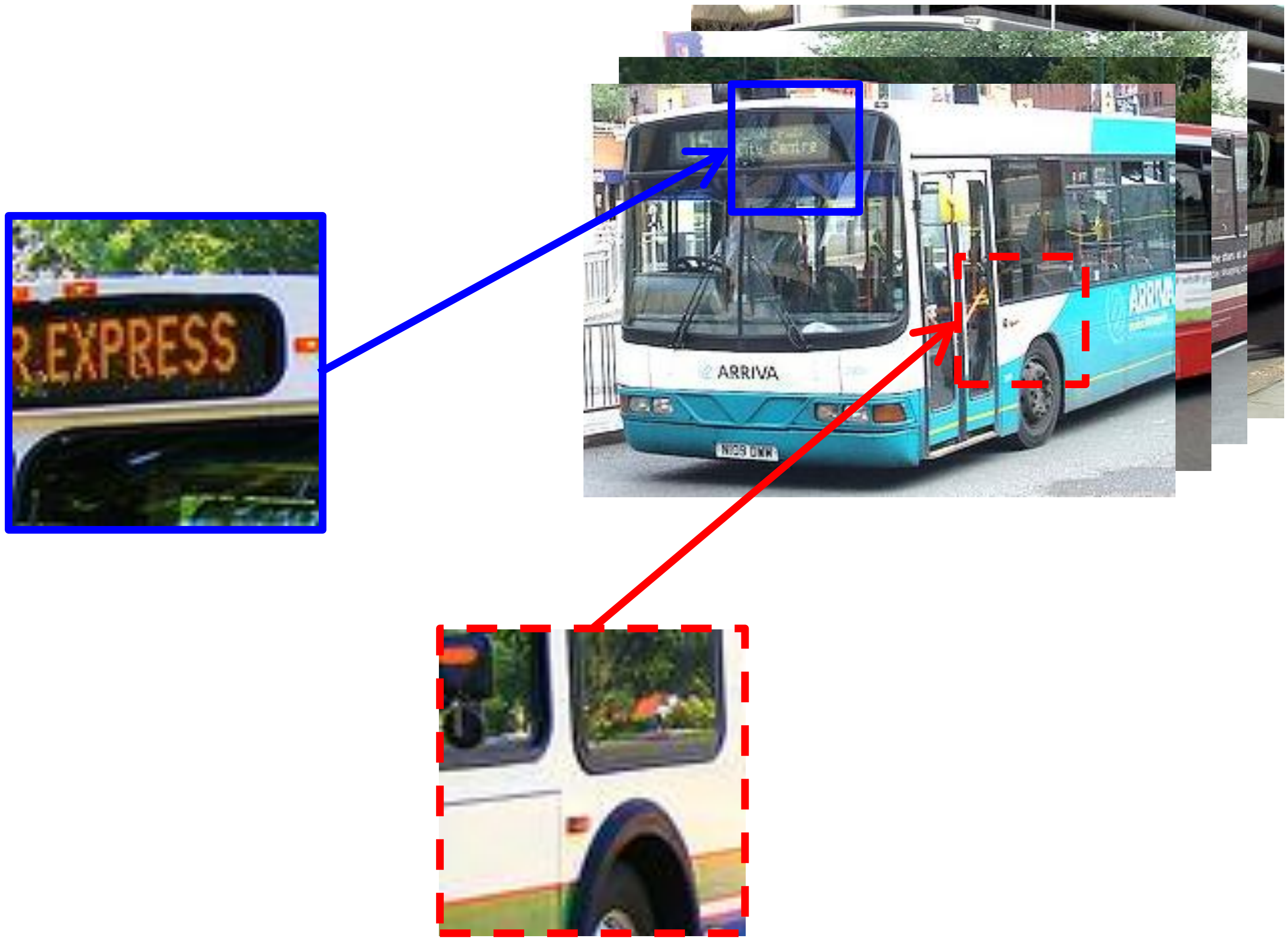


Context Prediction for Images



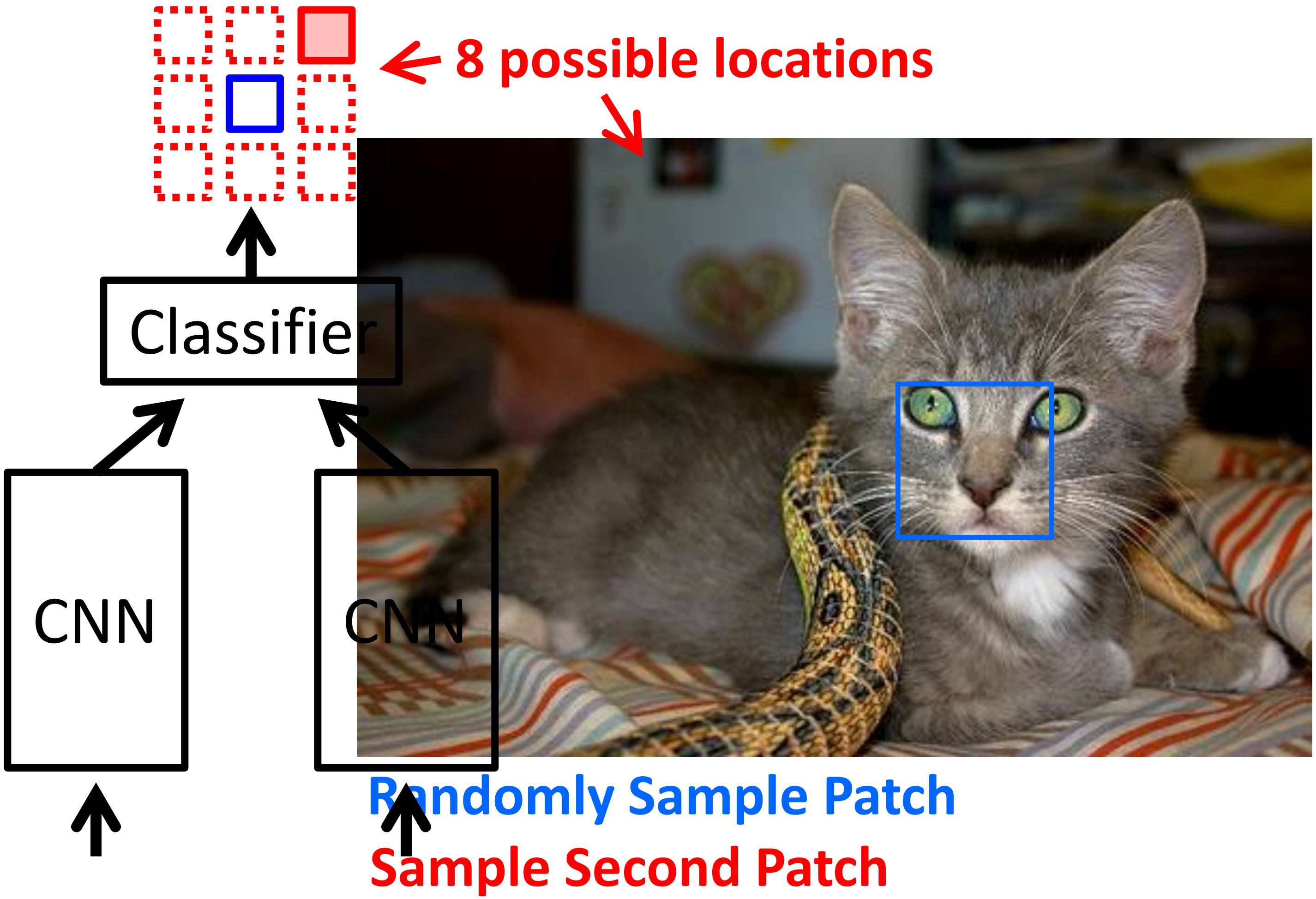
Doersch et al., "Unsupervised Visual Representation Learning by Context Prediction", ICCV 2015

Semantics from a non-semantic task

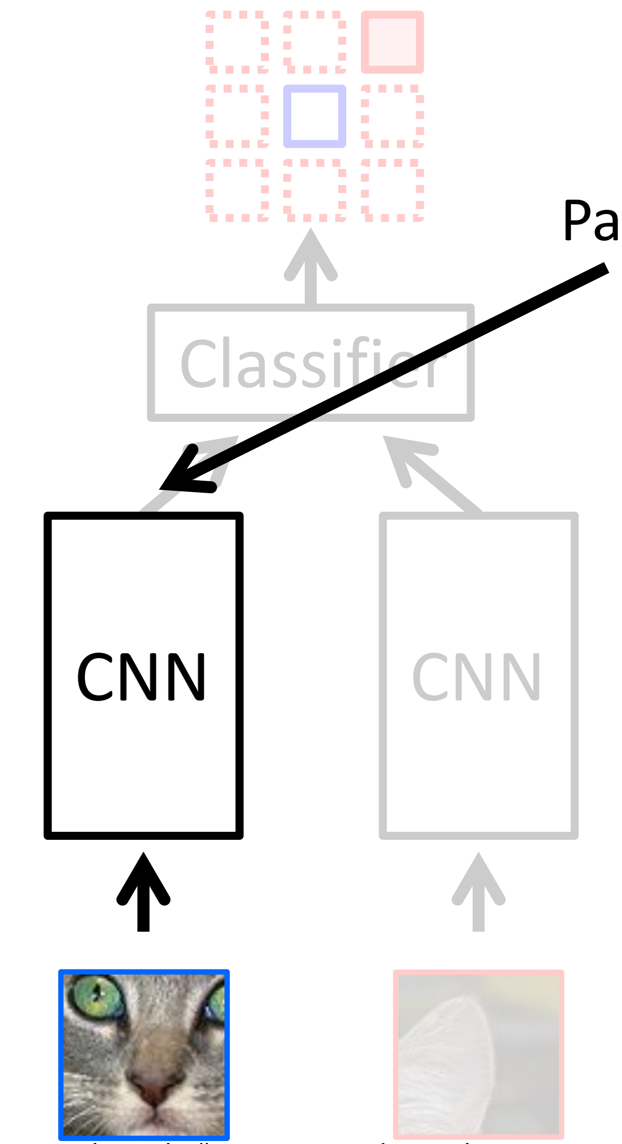


Doersch et al., "Unsupervised Visual Representation Learning by Context Prediction", ICCV 2015

Relative Position Task



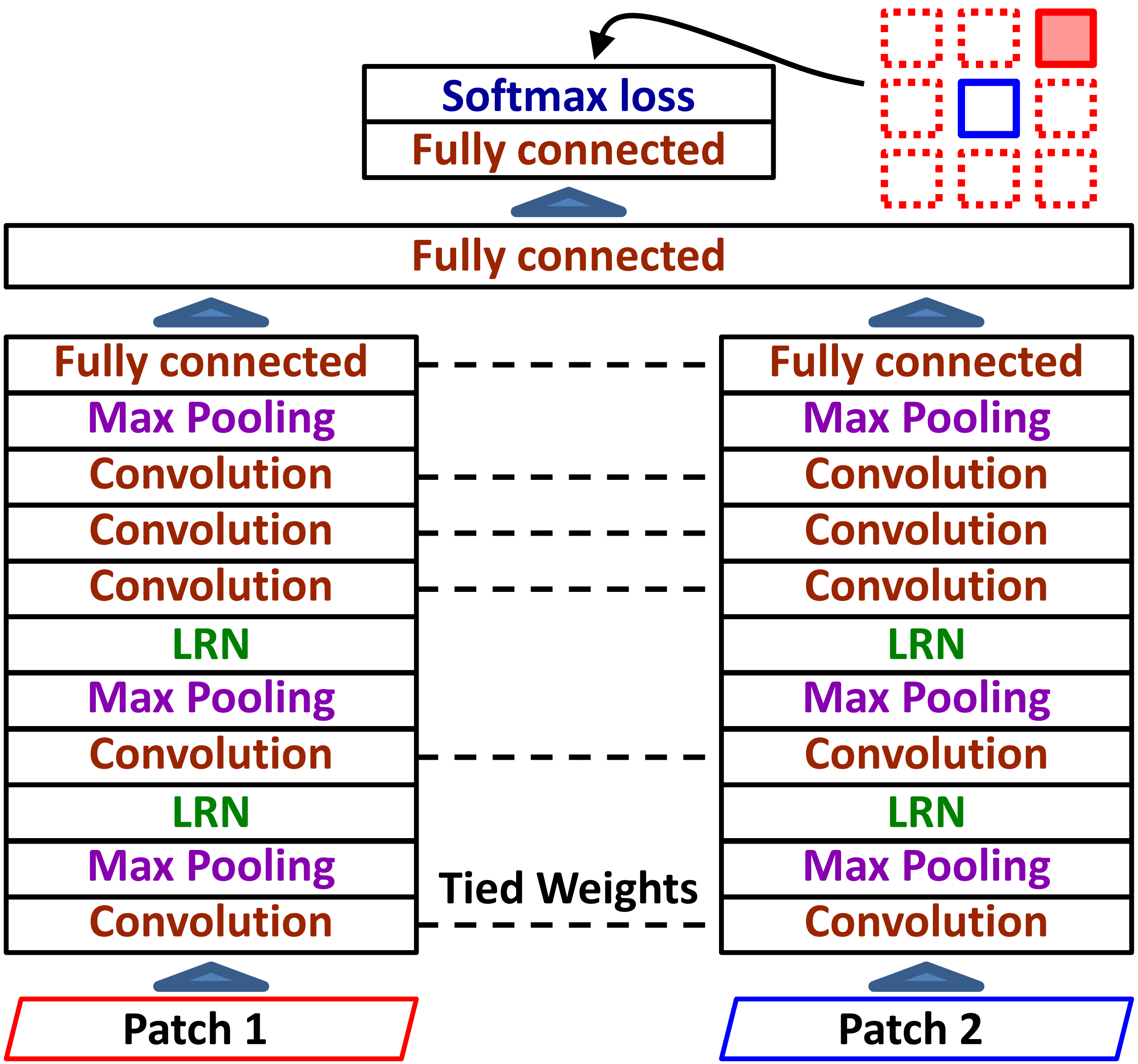
Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015



Note: connects ***across*** instances!

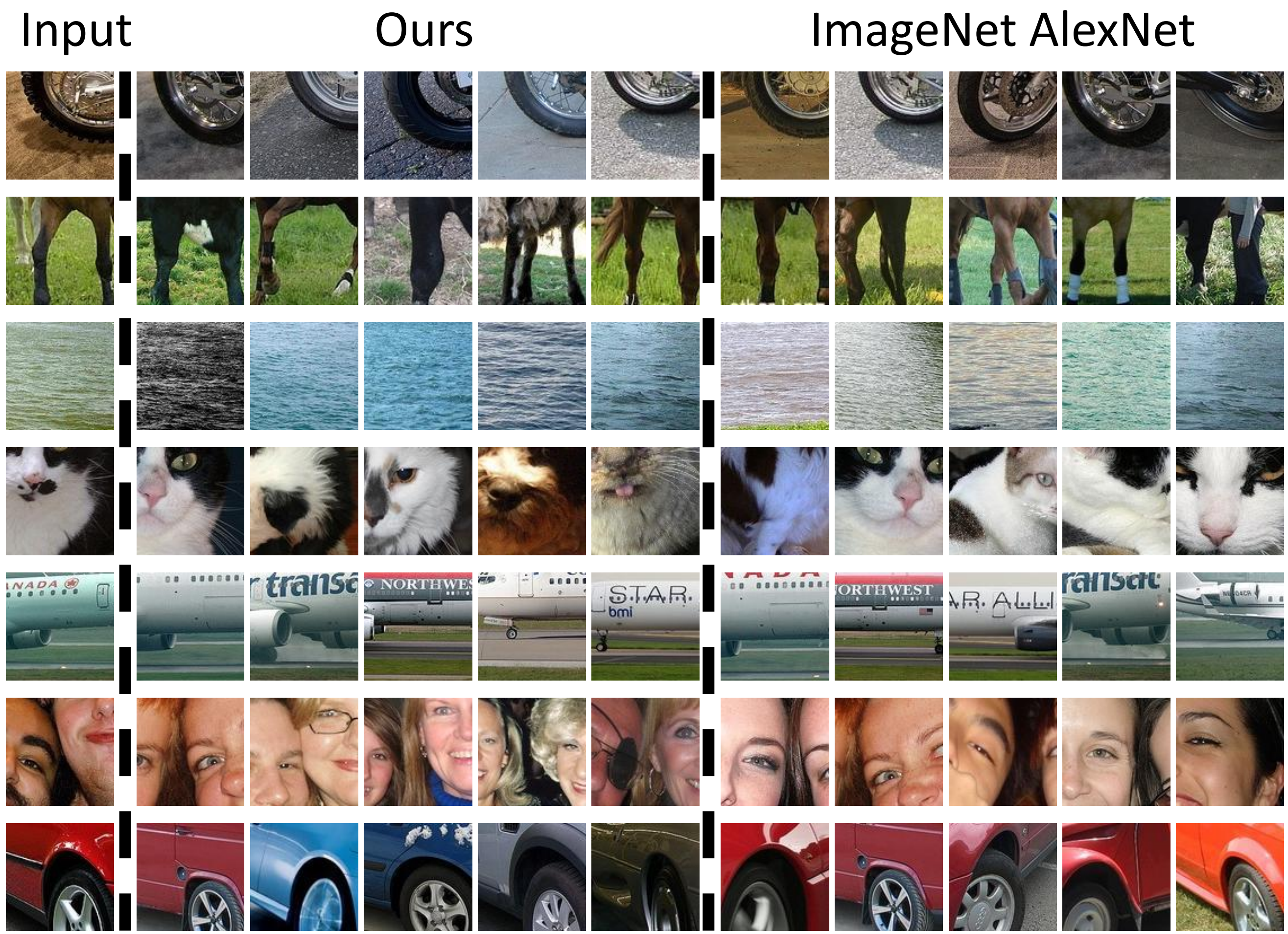
Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Architecture



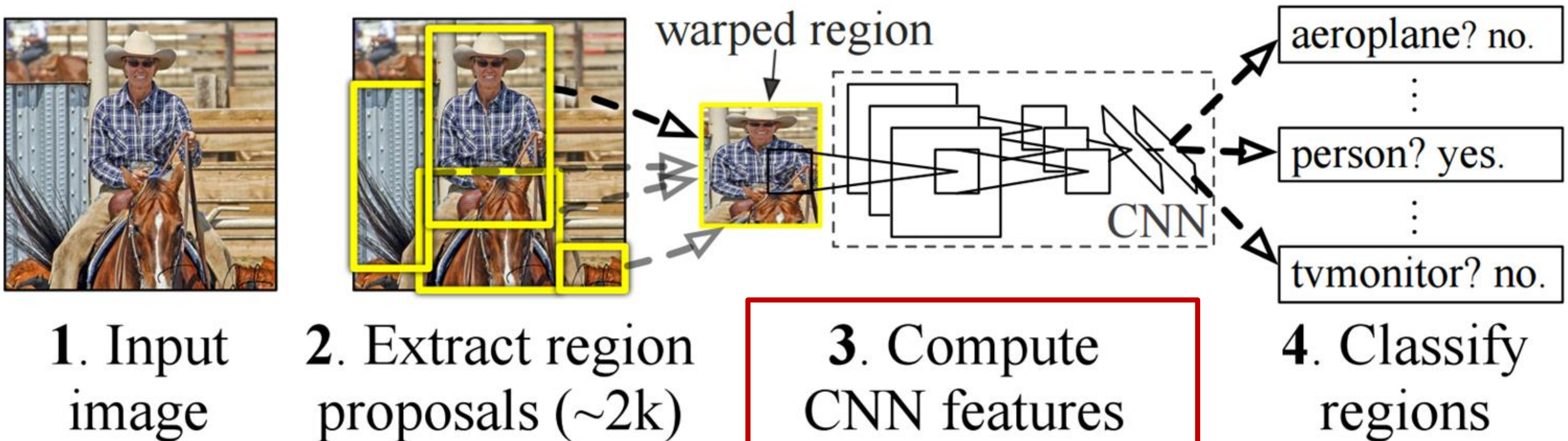
Doersch et al., "Unsupervised Visual Representation Learning by Context Prediction", ICCV 2015

What is learned?



Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Pre-Training for R-CNN



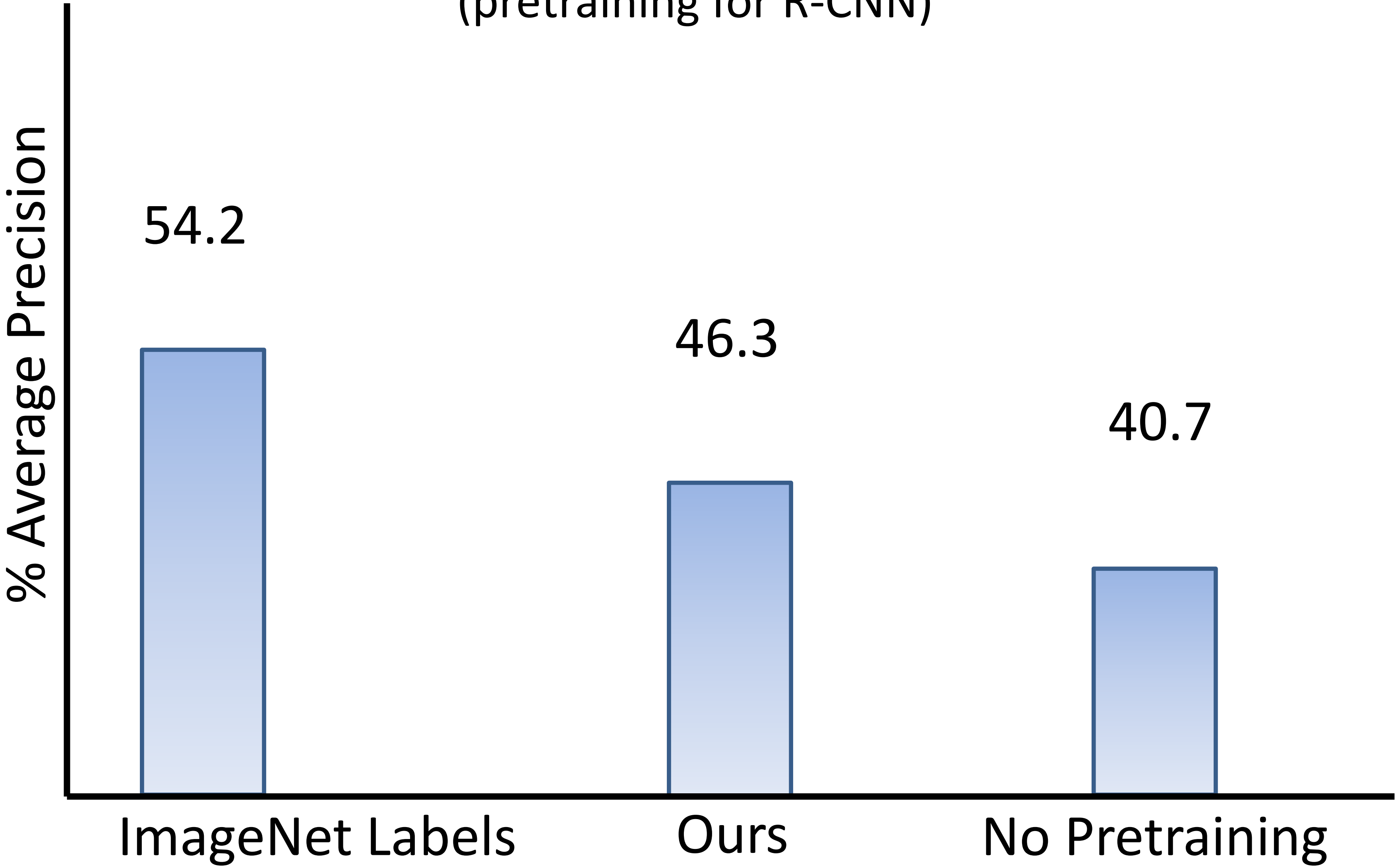
Pre-train on relative-position task, w/o labels

Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

[Girshick et al. 2014]

VOC 2007 Performance

(pretraining for R-CNN)



Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Which will be better?

- Option 1: pretrain (unsup) on dataset B
- Option 2: pretrain (sup) on dataset A
- Test on dataset B

Shuffle and Learn: Unsupervised Learning using Temporal Order Verification

Ishan Misra, C. Lawrence Zitnick, and Martial Hebert

ECCV 2016

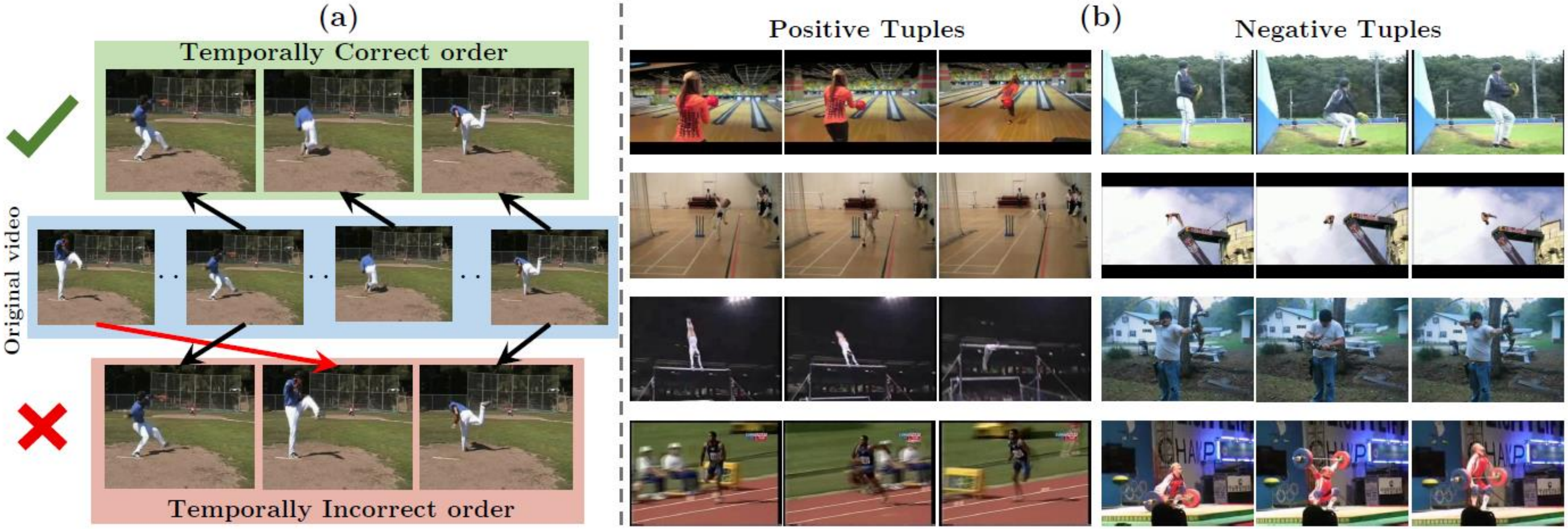


Fig. 1: **(a)** A video imposes a natural temporal structure for visual data. In many cases, one can easily verify whether frames are in the correct temporal order (shuffled or not). Such a simple sequential verification task captures important spatiotemporal signals in videos. We use this task for unsupervised pre-training of a Convolutional Neural Network (CNN). **(b)** Some examples of the automatically extracted positive and negative tuples used to formulate a classification task for a CNN.

Misra et al., "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification", ECCV 2016

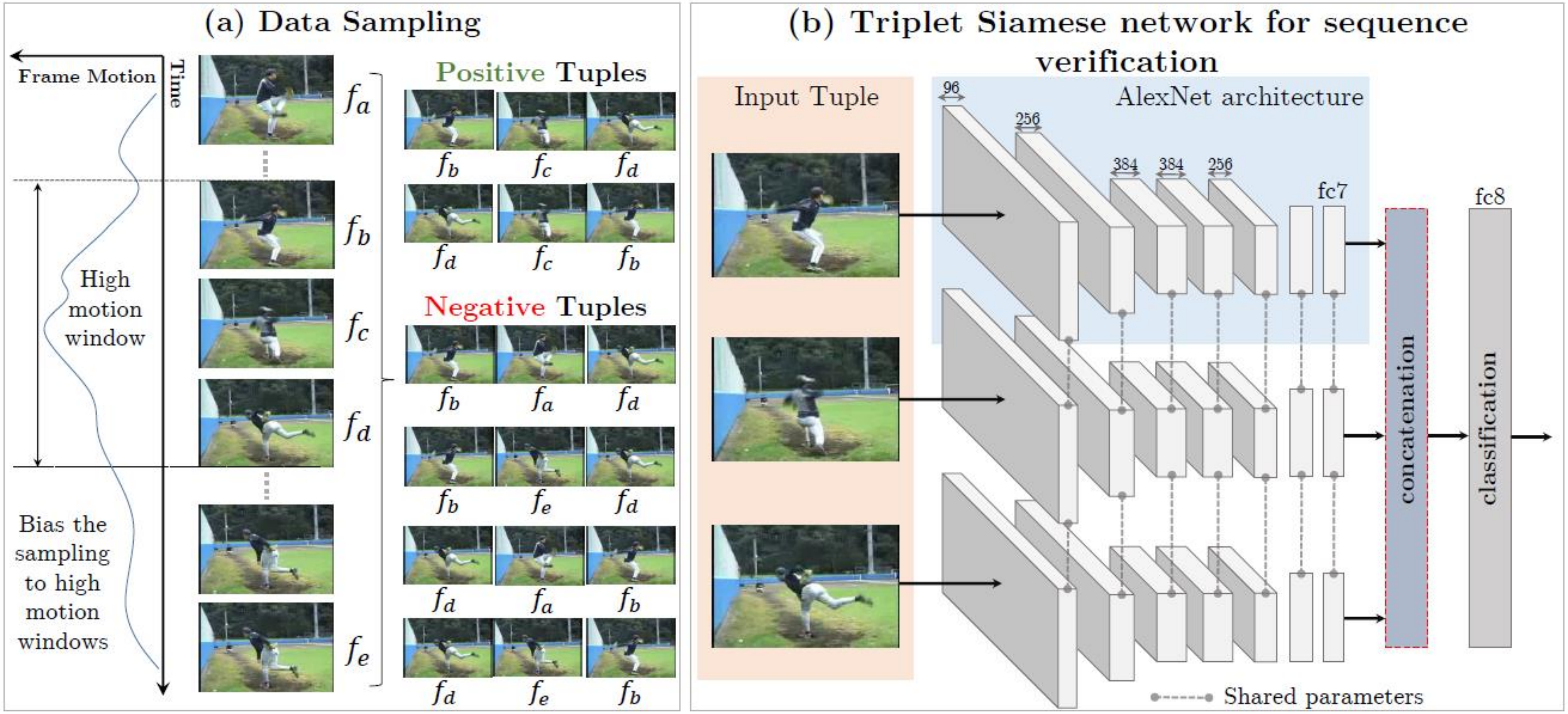


Fig. 2: **(a)** We sample tuples of frames from high motion windows in a video. We form positive and negative tuples based on whether the three input frames are in the correct temporal order. **(b)** Our triplet Siamese network architecture has three parallel network stacks with shared weights upto the `fc7` layer. Each stack takes a frame as input, and produces a representation at the `fc7` layer. The concatenated `fc7` representations are used to predict whether the input tuple is in the correct temporal order.

Misra et al., "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification", ECCV 2016

Table 2: Mean classification accuracies over the 3 splits of UCF101 and HMDB51 datasets. We compare different initializations and finetune them for action recognition.

Dataset	Initialization	Mean Accuracy
UCF101	Random	38.6
	(Ours) Tuple verification	**50.2**
HMDB51	Random	13.3
	UCF Supervised	15.2
	(Ours) Tuple verification	**18.1**

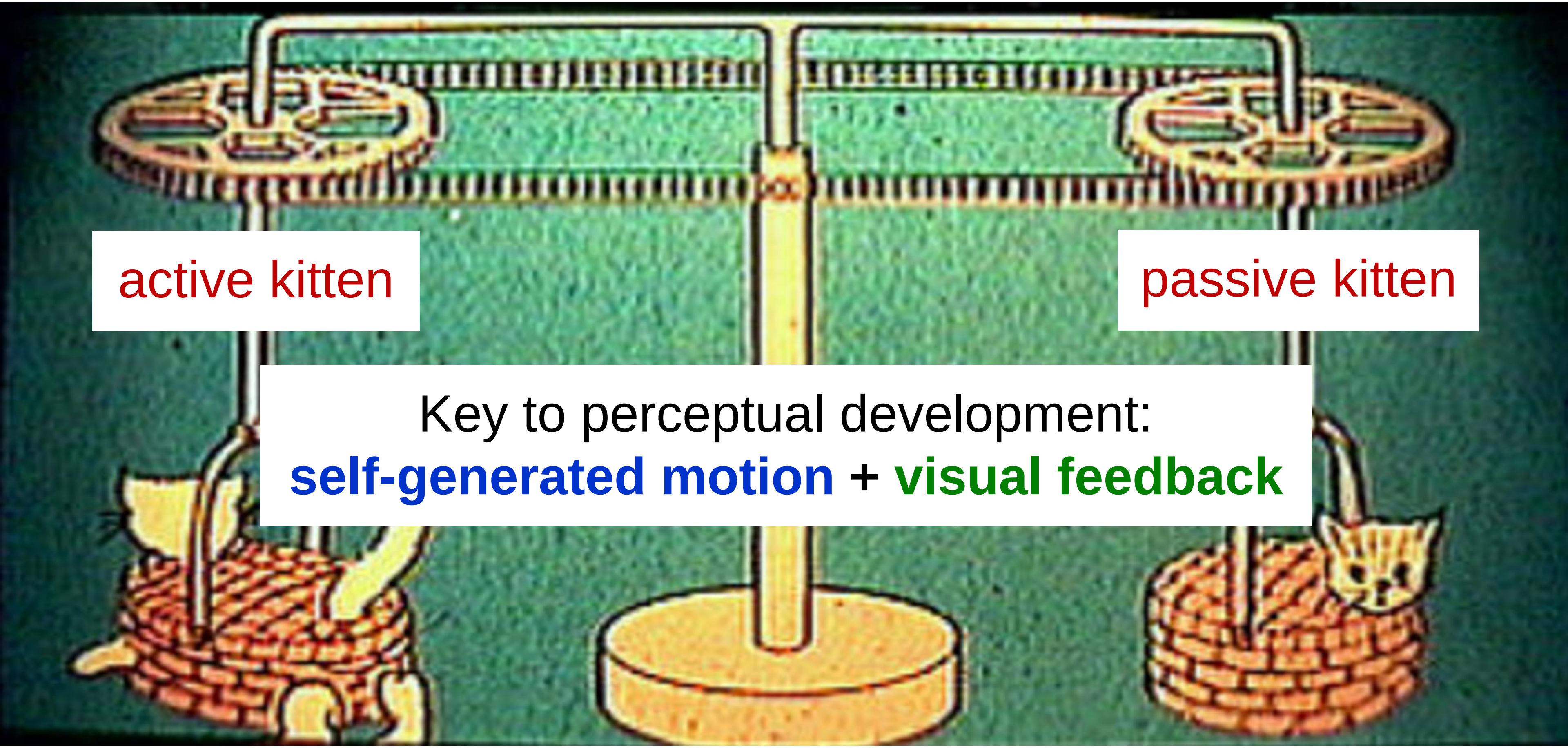
Misra et al., "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification", ECCV 2016

Learning image representations tied to ego-motion

Dinesh Jayaraman and Kristen Grauman

ICCV 2015

The kitten carousel experiment
[Held & Hein, 1963]



Key to perceptual development:
self-generated motion + visual feedback

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Problem with today's visual learning

Status quo: Learn from "disembodied" bag of labeled snapshots.

Our goal: Learn in the context of acting and moving in the world.

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Our idea: Ego-motion ↔ vision

Goal: Teach computer vision system the connection:
"how I move" ↔ "how my visual surroundings change"

Ego-motion motor signals

+

Unlabeled video

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Ego-motion ↔ vision: view prediction

After moving:

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Ego-motion ↔ vision for recognition

Learning this connection requires:

- Depth, 3D geometry
- Semantics
- Context

Also key to recognition!

Can be learned without manual labels!

Our approach: unsupervised feature learning using egocentric video + motor signals

Approach idea: Ego-motion equivariance

Invariant features: unresponsive to some classes of transformations

$$\mathbf{z}(g\mathbf{x}) \approx \mathbf{z}(\mathbf{x})$$

Equivariant features : *predictably* responsive to some classes of transformations, through simple mappings (e.g., linear)

"equivariance map"

$$\mathbf{z}(g\mathbf{x}) \approx M_g \mathbf{z}(\mathbf{x})$$

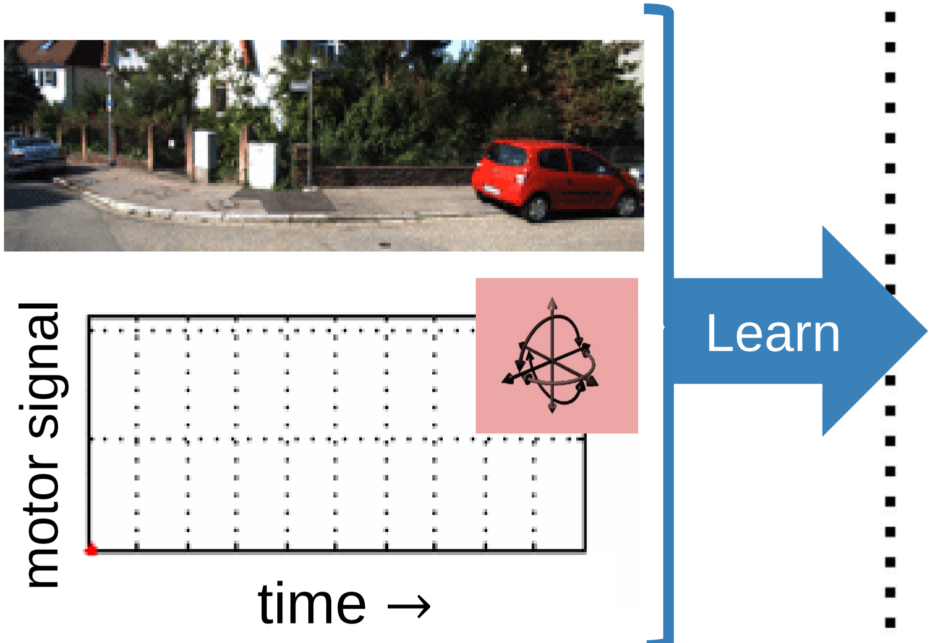
Invariance *discards* information;
equivariance *organizes* it.

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Approach idea: Ego-motion equivariance

Training data

Unlabeled video +
motor signals



Equivariant embedding

organized by ego-motions

Pairs of frames related by
similar ego-motion should
be related by same
feature transformation

Jayaraman and Grauman, “Learning image representations tied to ego-motion”, ICCV 2015

Approach overview

Our approach: unsupervised feature learning using egocentric video + motor signals

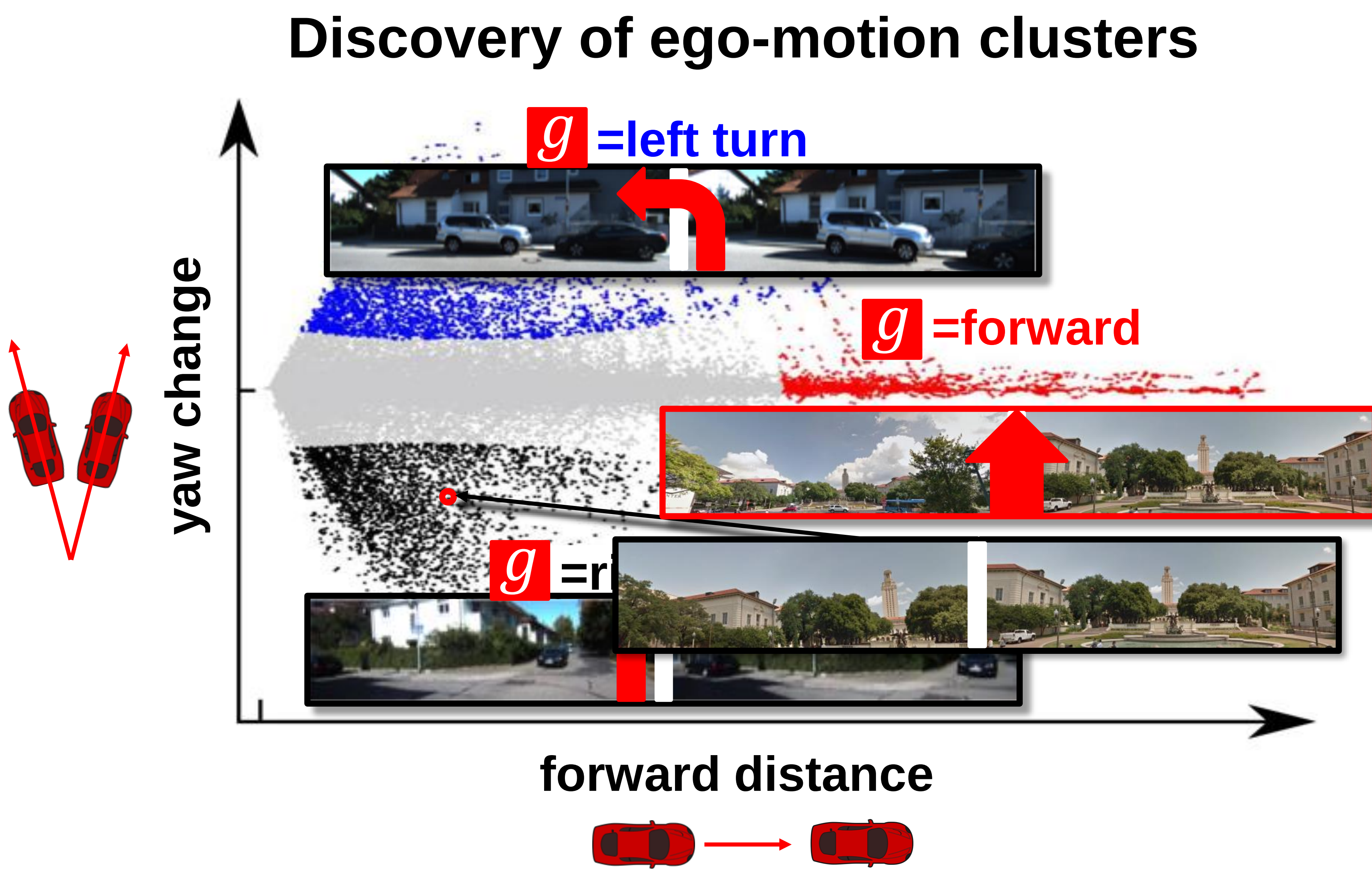
1. Extract training frame pairs from video

2. Learn ego-motion-equivariant image features

3. Train on target recognition task in parallel

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

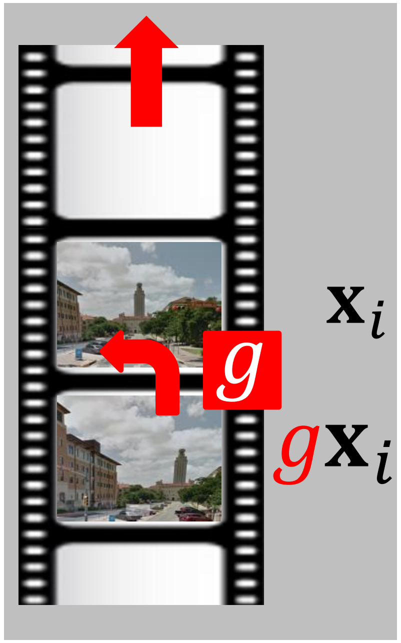
Training frame pair mining



Jayaraman and Grauman, “Learning image representations tied to ego-motion”, ICCV 2015

Ego-motion equivariant feature learning

Given:

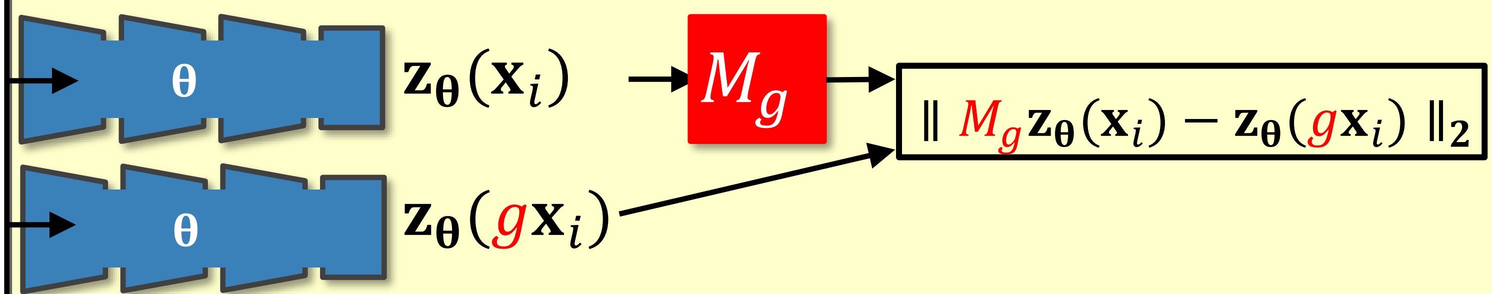


class y_k

Desired: for all motions g and all images $\mathbf{x}$,

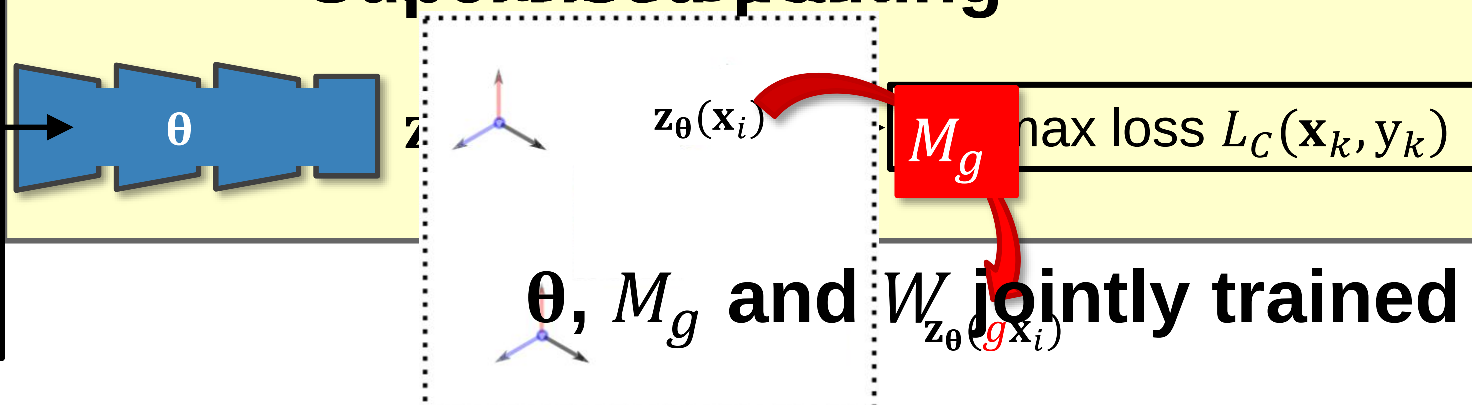
$$\mathbf{z}_\theta(g\mathbf{x}) \approx M_g \mathbf{z}_\theta(\mathbf{x})$$

Unsupervised training



Supervised training

Feature space



θ, M_g and W jointly trained

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Results: Recognition

Learn from **unlabeled car video** (KITTI)

Geiger et al, IJRR '13

Exploit features for **static scene classification** (SUN, 397 classes)

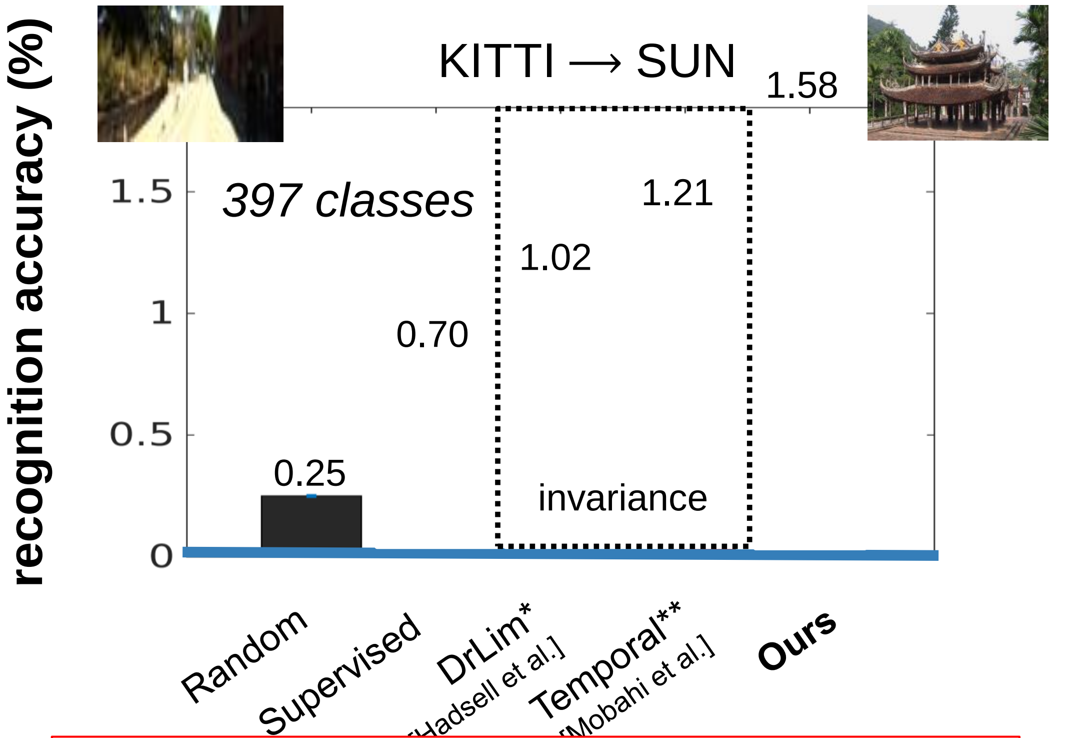
Apse
Window seat
Art school
Library
Auditorium
Bus interior
Cathedral
Freeway
Guardhouse

Xiao et al, CVPR '10

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Results: Recognition

Do ego-motion equivariant features improve recognition?



6 labeled training examples per class

Up to 30% accuracy increase over state of the art!

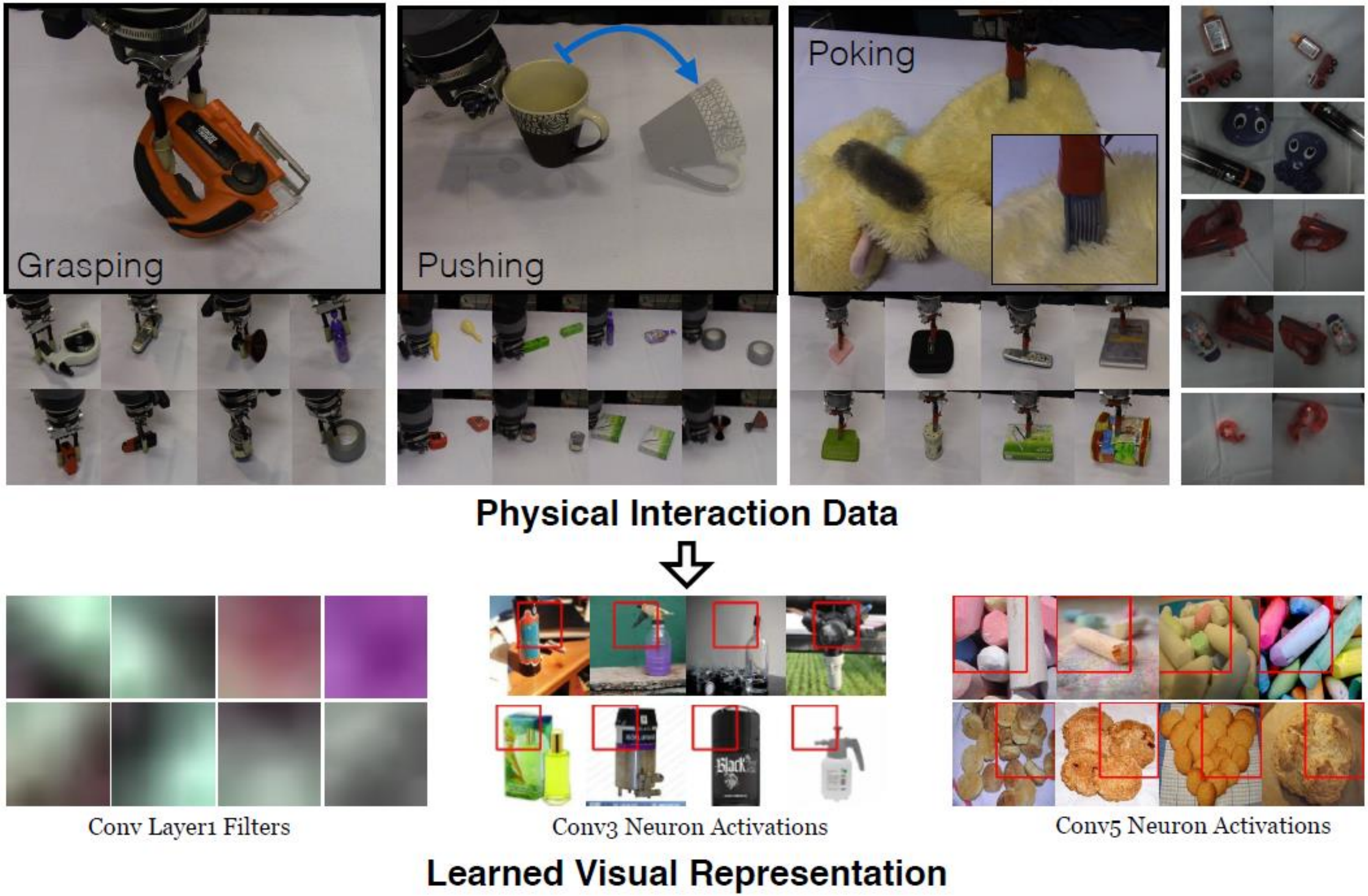
Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

The Curious Robot: Learning Visual Representations via Physical Interactions

Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han,
Yong-Lae Park, and Abhinav Gupta

ECCV 2016

Embodied representations



Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Grasping

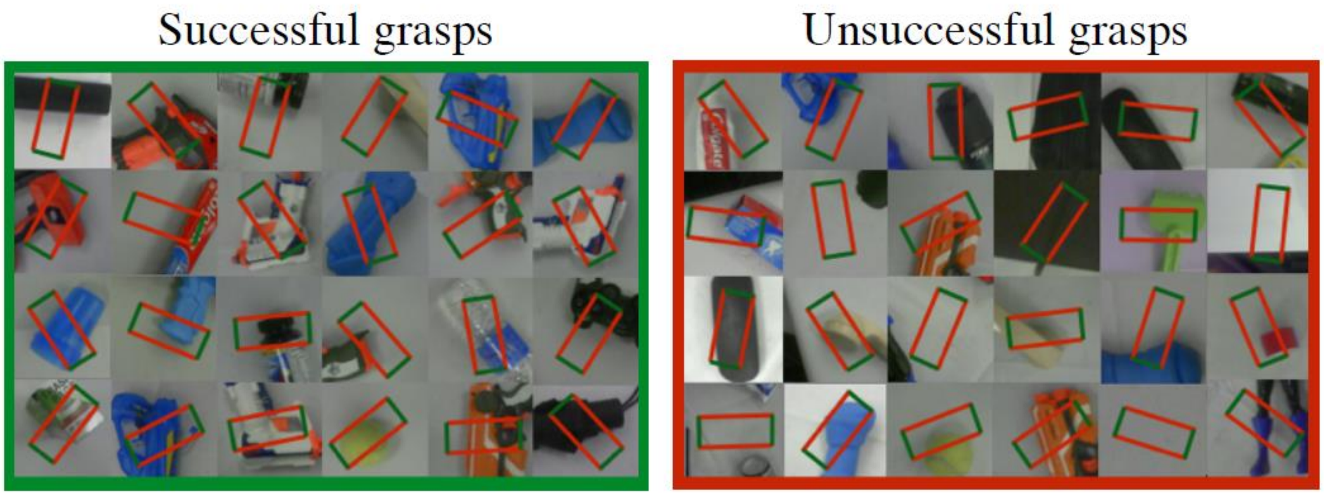


Fig. 2. Examples of successful (left) and unsuccesful grasps (right). We use a patch based representation: given an input patch we predict 18-dim vector which represents whether the center location of the patch is graspable at 0°, 10°, $\ldots 170^\circ$.

Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Pushing

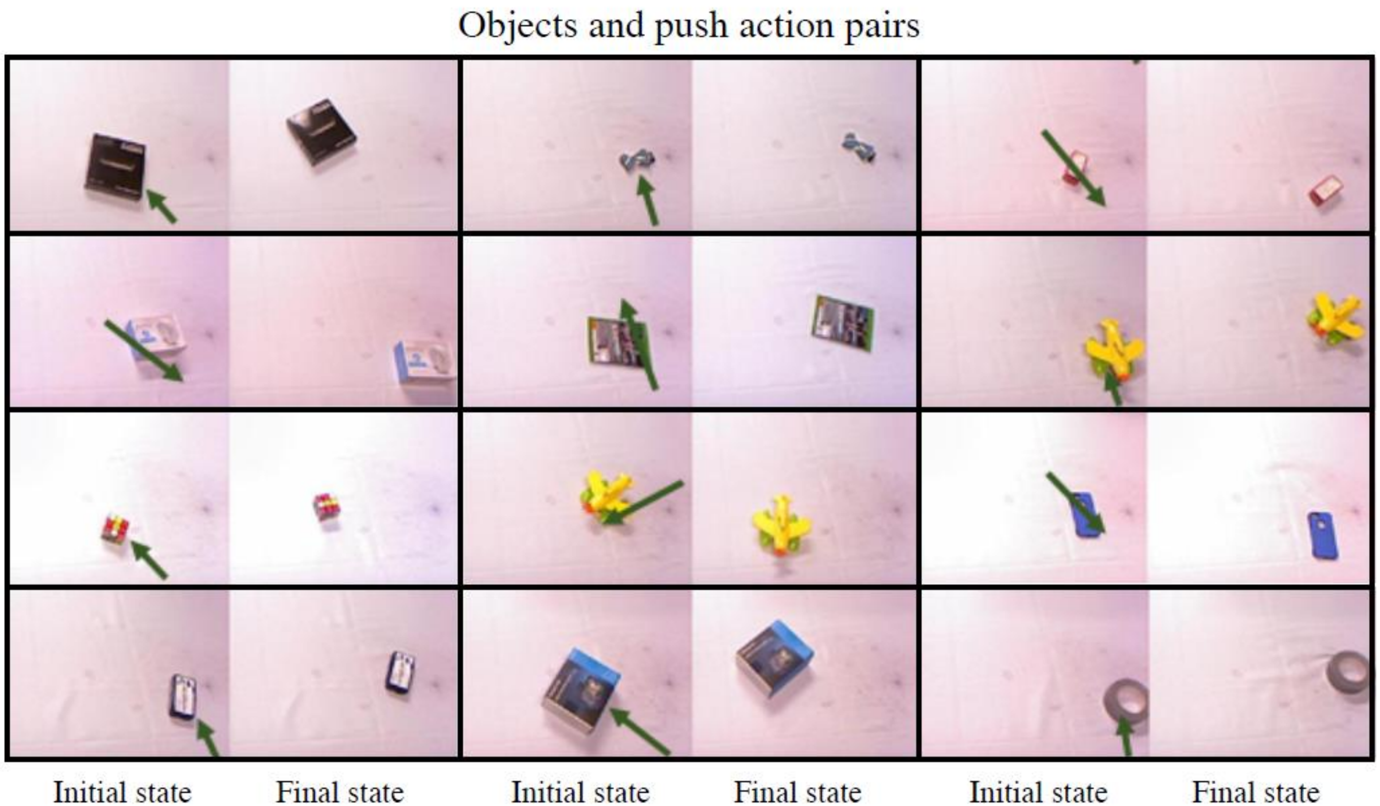


Fig. 4. Examples of initial state and final state images taken for the push action. The arrows demonstrate the direction and magnitude of the push action.

Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Poking

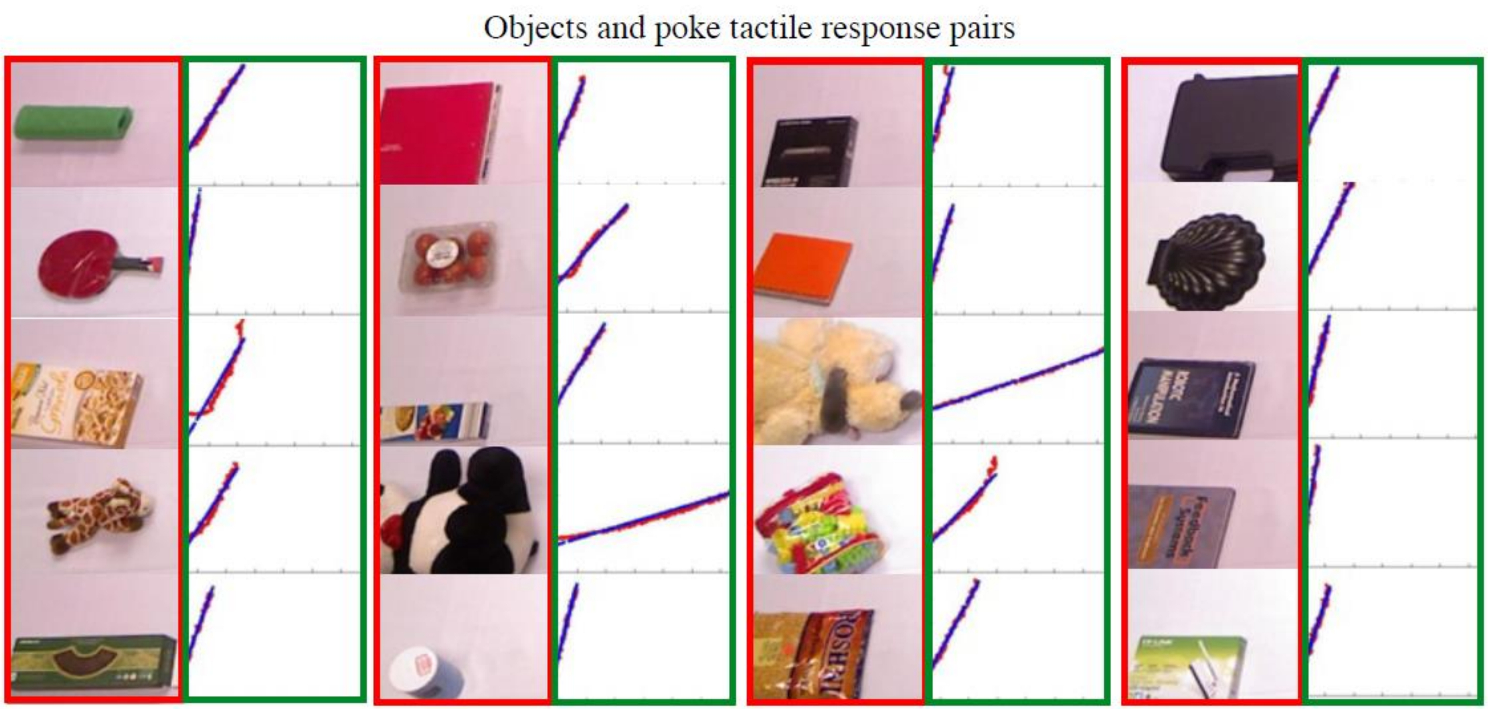


Fig. 6. Examples of the data collected by the poking action. On the left we show the object poked, and on the right we show force profiles as observed by the tactile sensor.

Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Pose/viewpoint invariance

Fig. 7. Examples of objects in different poses provided to the embedding network.

Pinto et al., “The Curious Robot: Learning Visual Representations via Physical Interactions”, ECCV 2016

Representations from interactions

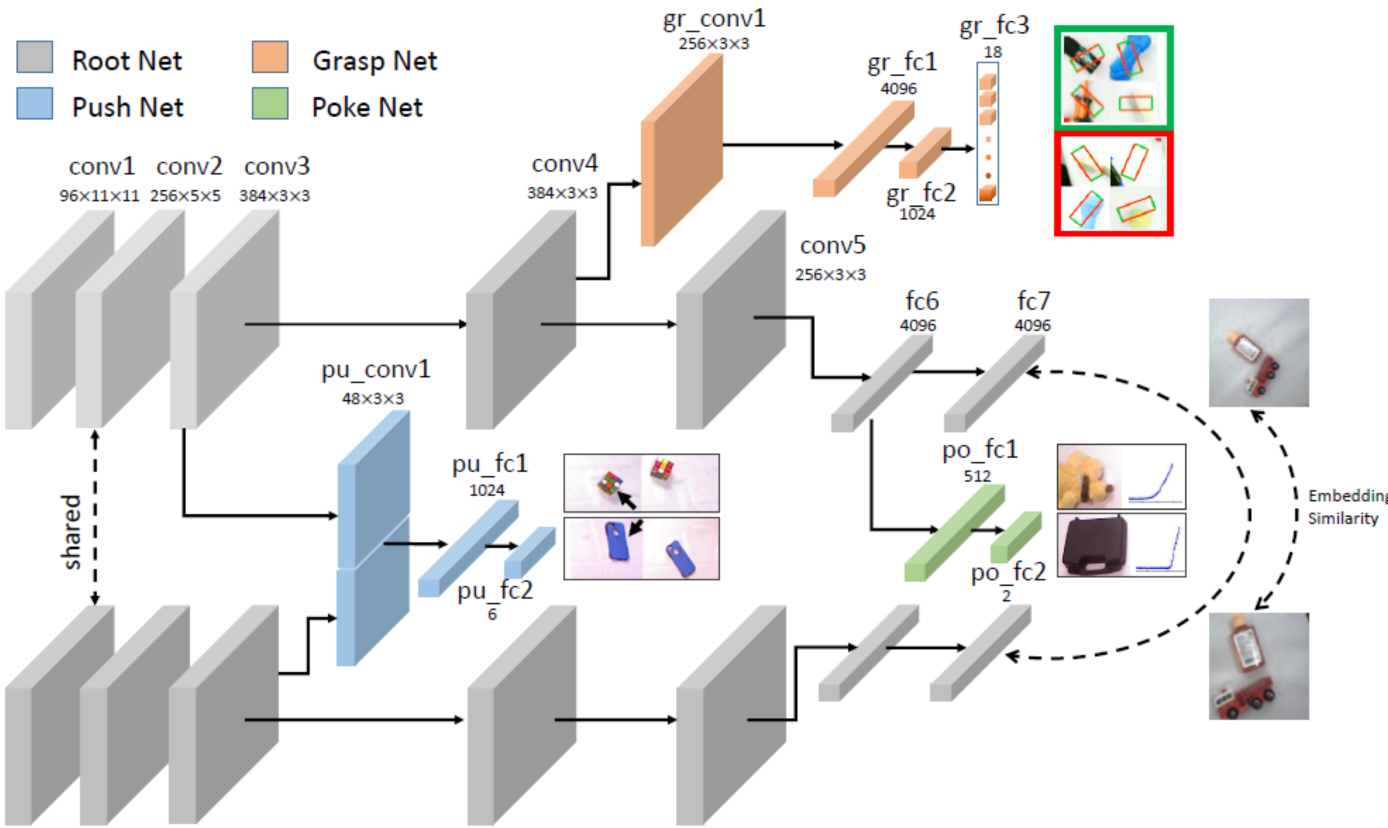


Fig. 8. Our shared convolutional architecture for four different tasks.

Pinto et al., “The Curious Robot: Learning Visual Representations via Physical Interactions”, ECCV 2016

Classification/retrieval performance

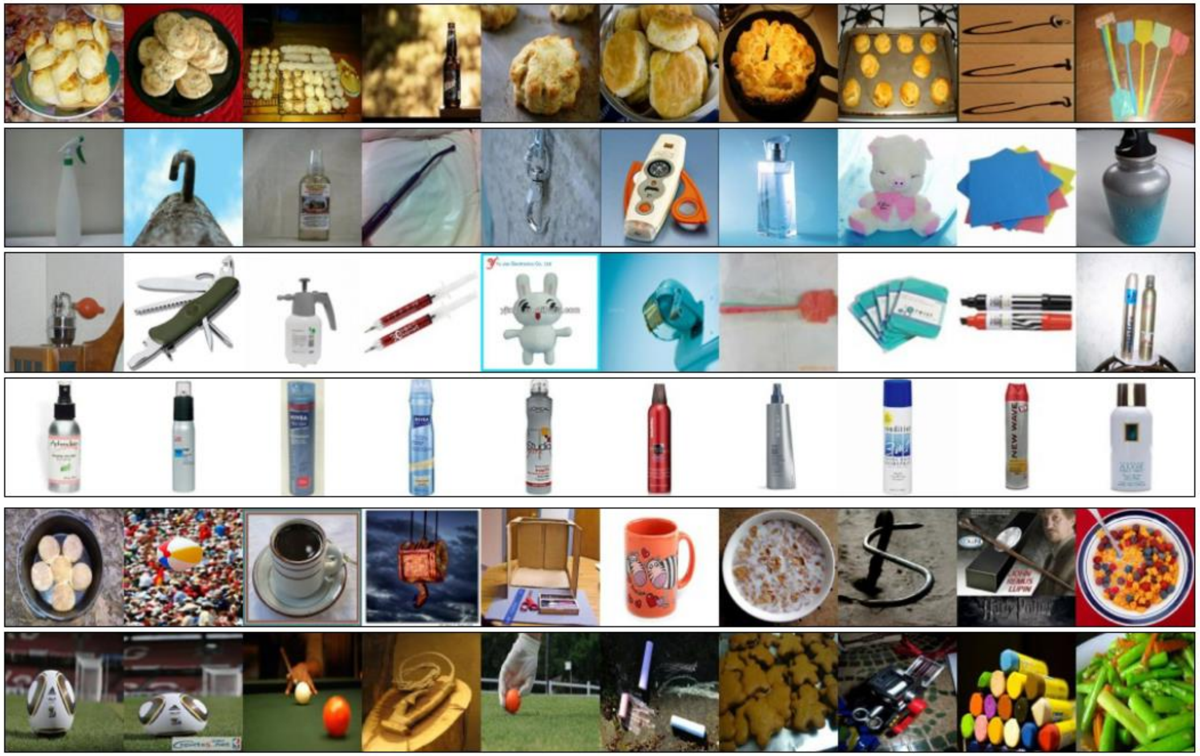


Fig. 10. The first column corresponds to query image and rest show the retrieval. Note how the network learns that cups and bowls are similar (row 5).

Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Classification/retrieval performance

Table 1. Classification accuracy on ImageNet Household, UW RGBD and Caltech-256

	Household	UW RGBD	Caltech-256
Root network with random init.	0.250	0.468	0.242
Root network trained on robot tasks (**ours**)	0.354	0.693	0.317
AlexNet trained on ImageNet	0.625	0.820	0.656

Table 2. Image Retrieval with Recall@k metric

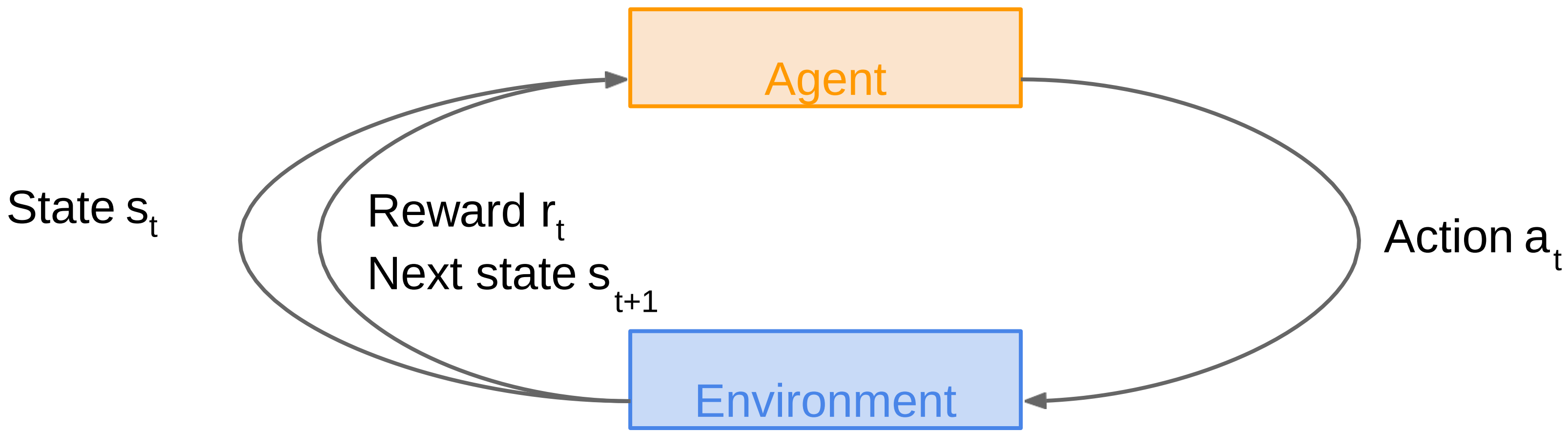
	Instance level				Category level			
	k=1	k=5	k=10	k=20	k=1	k=5	k=10	k=20
Random Network	0.062	0.219	0.331	0.475	0.150	0.466	0.652	0.800
Our Network	0.720	0.831	0.875	0.909	0.833	0.918	0.946	0.966
AlexNet	0.686	0.857	0.903	0.941	0.854	0.953	0.969	0.982

Pinto et al., “The Curious Robot: Learning Visual Representations via Physical Interactions”, ECCV 2016

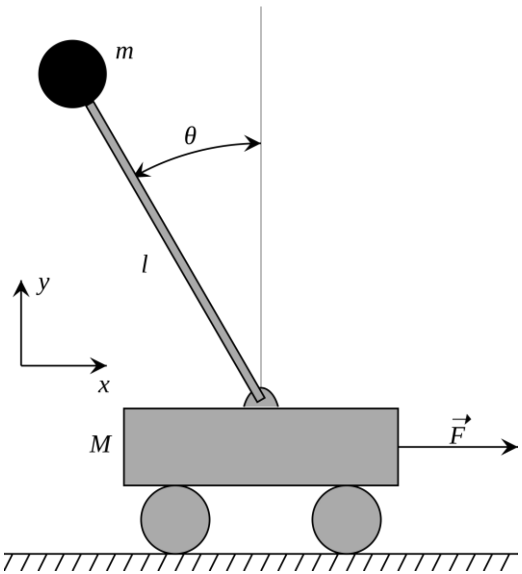
Part III: Reinforcement Learning

- Basics: actions, states, rewards, MDP
- Different techniques (Q learning, policy gradients, actor-critic, etc.)
- Example applications

Reinforcement Learning



Cart-Pole Problem



Objective: Balance a pole on top of a movable cart

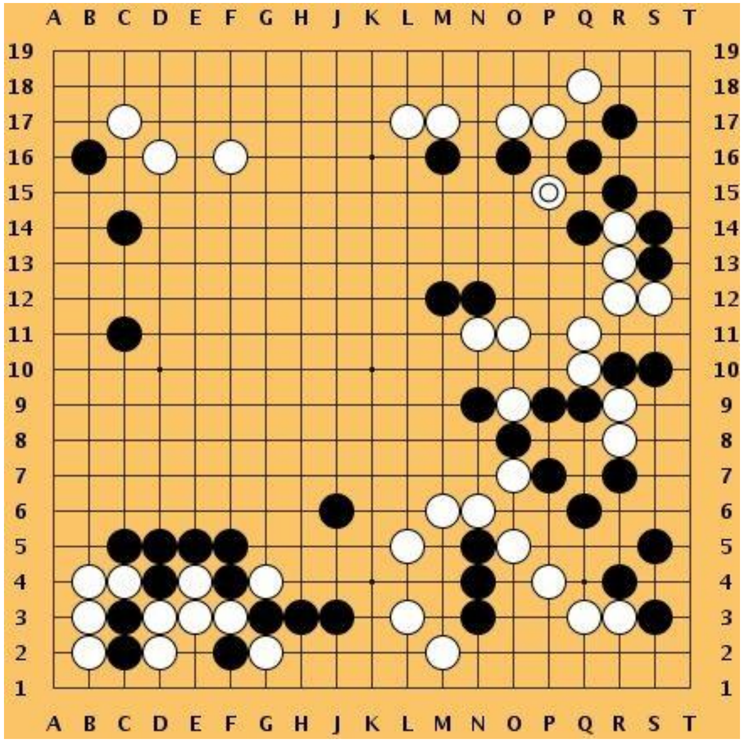
State: angle, angular speed, position, horizontal velocity
Action: horizontal force applied on the cart
Reward: 1 at each time step if the pole is upright

Atari Games

Objective: Complete the game with the highest score

State: Raw pixel inputs of the game state
Action: Game controls e.g. Left, Right, Up, Down
Reward: Score increase/decrease at each time step

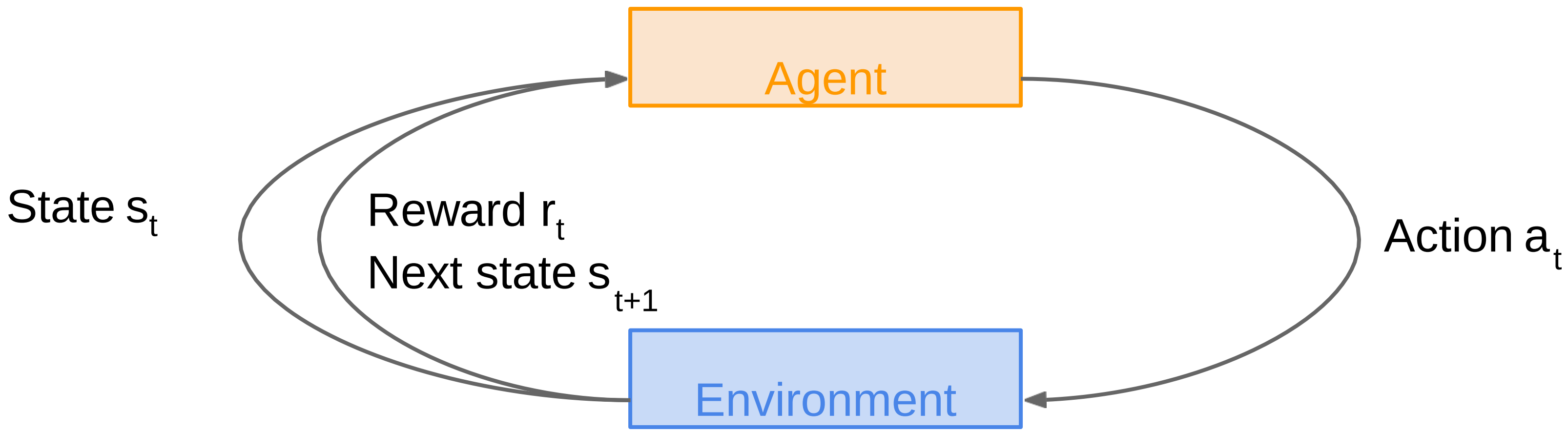
Go



Objective: Win the game!

State: Position of all pieces
Action: Where to put the next piece down
Reward: 1 if win at the end of the game, 0 otherwise

How can we mathematically formalize the RL problem?



Markov Decision Process

- Mathematical formulation of the RL problem
- **Markov property**: Current state completely characterises the state of the world

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

Markov Decision Process

- At time step t=0, environment samples initial state $s_0 \sim p(s_0)$
- Then, for t=0 until done:
  - Agent selects action a_t
  - Environment samples reward $r_t \sim R(. | s_t, a_t)$
  - Environment samples next state $s_{t+1} \sim P(. | s_t, a_t)$
  - Agent receives reward r_t and next state s_{t+1}

- A policy u is a function from S to A that specifies what action to take in each state
- **Objective**: find policy u* that maximizes cumulative discounted reward: $\sum_{t \geq 0} \gamma^t r_t$

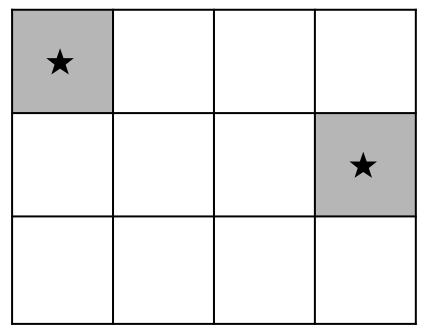
A simple MDP: Grid World

actions = {

1. right
2. left
3. up
4. down

}

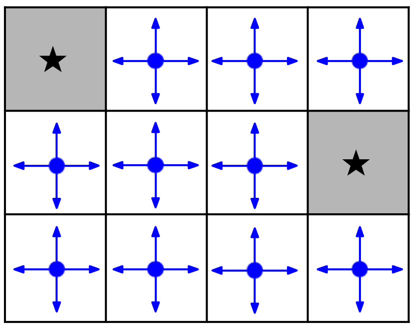
states



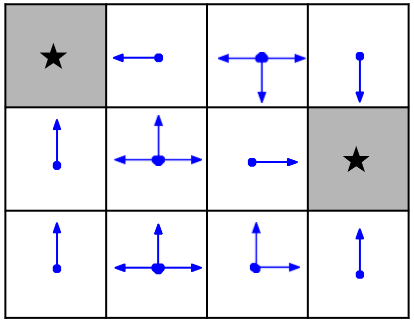
Set a negative "reward" for each transition (e.g. $r = -1$)

Objective: reach one of terminal states (greyed out) in least number of actions

A simple MDP: Grid World



Random Policy



Optimal Policy

The optimal policy u*

We want to find optimal policy u* that maximizes the sum of rewards.

How do we handle the randomness (initial state, transition probability…)?

The optimal policy u*

We want to find optimal policy u* that maximizes the sum of rewards.

How do we handle the randomness (initial state, transition probability…)?
Maximize the **expected sum of rewards!**

Formally: $\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right]$ with $s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$

Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) s_0, a_0, r_0, s_1, a_1, r_1, …

How good is a state?
The **value function** at state s, is the expected cumulative reward from following the policy from state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

How good is a state-action pair?
The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Bellman equation

The optimal Q-value function Q*
is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Q* satisfies the following **Bellman equation**:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s, a\right]$$

Intuition: if the optimal state-action values for the next time-step Q*(s',a') are known, then the optimal strategy is to take the action that maximizes the expected value of
$r + \gamma Q^*(s',a')$

The optimal policy u* corresponds to taking the best action in any state as specified by Q*

Solving for the optimal policy: Q-learning

Q-learning: Use a function approximator to estimate the action-value function

$$Q(s, a; \theta) \approx Q^*(s, a)$$

function parameters (weights)

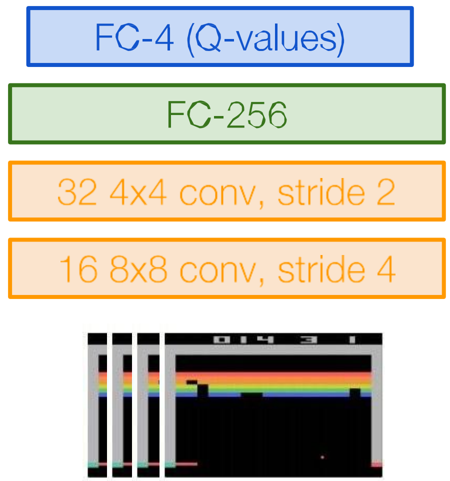
If the function approximator is a deep neural network => **deep q-learning**!

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Q-network Architecture

$Q(s, a; \theta)$:
neural network
with weights θ

A single feedforward pass to compute Q-values for all actions from the current state => efficient!



Last FC layer has 4-d output (if 4 actions), corresponding to $Q(s_t, a_1)$, $Q(s_t, a_2)$, $Q(s_t, a_3)$, $Q(s_t, a_4)$

Number of actions between 4-18 depending on Atari game

Current state s_t: 84x84x4 stack of last 4 frames
(after RGB->grayscale conversion, downsampling, and cropping)

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N

Initialize action-value function Q with random weights

for episode $= 1, M$ **do**

  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$

  for $t = 1, T$ **do**

    With probability ϵ select a random action a_t

    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$

    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3

  end for

end for

Initialize replay memory, Q-network

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do** ← Play M episodes (full games)
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Initialize state (starting game screen pixels) at the beginning of each episode

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

For each timestep t of the game

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

With small probability, select a random action (explore), otherwise select greedy action from current policy

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

Take the action (a_t), and observe the reward r_t and next state s_{t+1}

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Store transition in replay memory

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Experience Replay:
Sample a random minibatch of transitions from replay memory and perform a gradient descent step

https://arxiv.org/pdf/1312.5602.pdf

Policy Gradients

What is a problem with Q-learning?
The Q-function can be very complicated!

Example: a robot grasping an object has a very high-dimensional state => hard to learn exact value of every (state, action) pair

But the policy can be much simpler: just close your hand
Can we learn a policy directly, e.g. finding the best policy from a collection of policies?

Policy Gradients

Formally, let's define a class of parameterized policies: $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Gradient ascent on policy parameters!

REINFORCE Algorithm (Williams 1992)

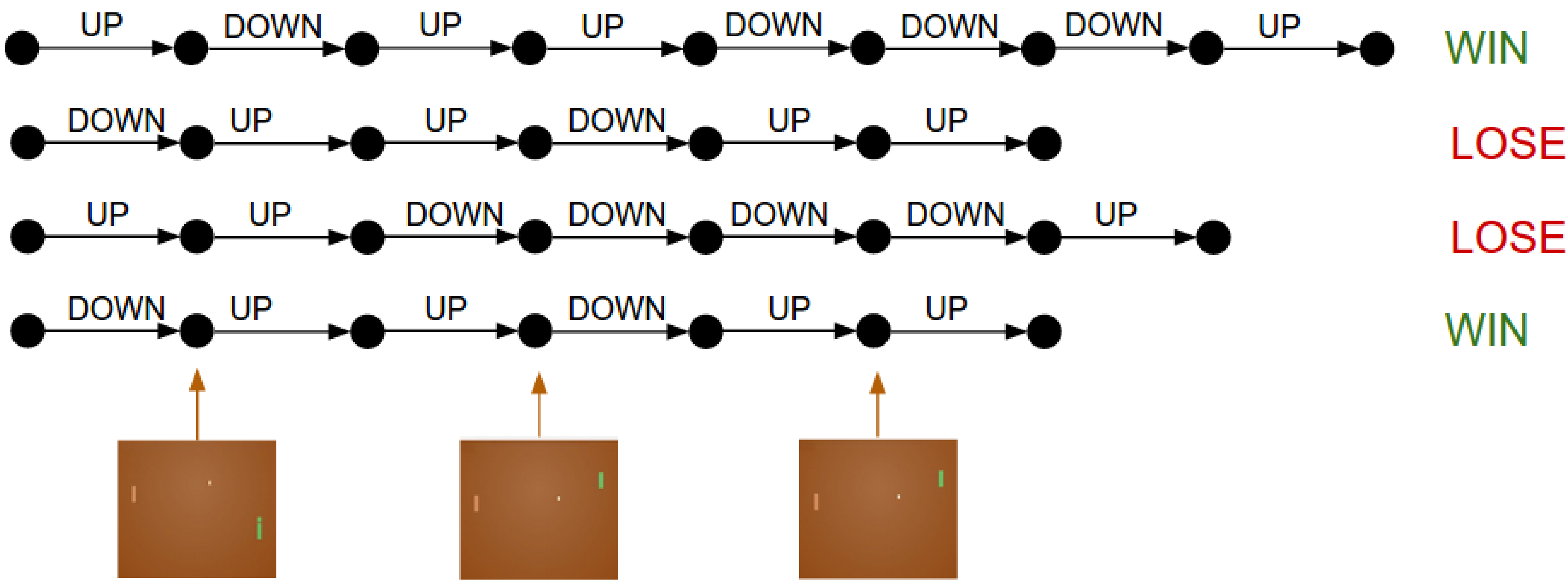
Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Interpretation:

- If *r(τ)* is high, push up the probabilities of the actions seen
- If *r(τ)* is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. But in expectation, it averages out!

Policy Gradients



REINFORCE Algorithm (Williams 1992)

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Interpretation:

- If *r(τ)* is high, push up the probabilities of the actions seen
- If *r(τ)* is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. But in expectation, it averages out!

However, this also suffers from high variance because credit assignment is really hard. Can we help the estimator?

Variance Reduction

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} \left(\sum_{t'\geq t} r_{t'}\right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Second idea: Use discount factor γ to ignore delayed effects

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} \left(\sum_{t'\geq t} \gamma^{t'-t} r_{t'}\right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Variance Reduction: Baseline

Problem: The raw value of a trajectory isn't necessarily meaningful. For example, if rewards are all positive, you keep pushing up probabilities of actions.

What is important then? Whether a reward is better or worse than what you expect to get

Idea: Introduce a baseline function dependent on the state.
Concretely, estimator is now:

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} \left(\sum_{t'\geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

How to choose the baseline?

Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Intuitively, we are happy with an action a_t in a state s_t if $Q^\pi(s_t, a_t) - V^\pi(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

Using this, we get the estimator:

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

Policy Gradients tl;dr

- Objective: $\sum_i A_i \log p(y_i|x_i)$
- x_i = state
- y_i = sampled action
- A_i = "advantage" e.g. +1/-1 for win/lose in simplest version, or discounted, or improvement over "baseline"

Adapted from Andrej Karpathy

Policy Gradients vs Q-Learning

- Policy gradients suffers from high variance and instability; might want to make gradients smaller (e.g. relative to a baseline)
- Policy gradients can handle continuous action spaces (Gaussian policy)
- Estimating exact value of state-action pairs vs choosing what actions to take (value not important)
- Step-by-step (did I correctly estimate the reward at this time) vs delayed feedback (run policy and wait until game terminates)

Actor-Critic Algorithm

We can combine Policy Gradients and Q-learning by training both an **actor** (the policy) and a **critic** (the Q-function).

- The actor decides which action to take, and the critic tells the actor how good its action was and how it should adjust
- Also alleviates the task of the critic as it only has to learn the values of (state, action) pairs generated by the policy
- Can also incorporate Q-learning tricks e.g. experience replay
- Define by the advantage function how much an action was better than expected

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

RL for navigation

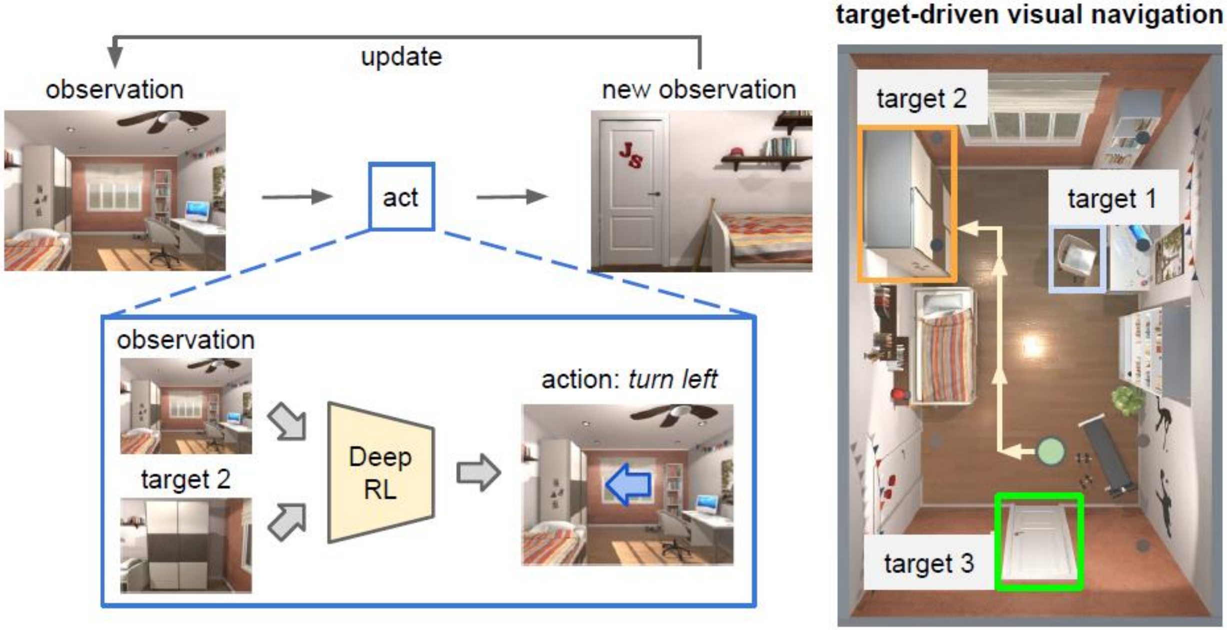


Fig. 1. The goal of our deep reinforcement learning model is to navigate towards a visual target with a minimum number of steps. Our model takes the current observation and the image of the target as input and generates an action in the 3D environment as the output. Our model learns to navigate to different targets in a scene without re-training.

Zhu et al., "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning", ICRA 2017

RL for navigation

(a) (b)

Figure 1: Our goal is to use scene priors to improve navigation in unseen scenes and towards novel objects. (a) There is no mug in the field of view of the agent, but the likely location for finding a mug is the cabinet near the coffee machine. (b) The agent has not seen a mango before, but it infers that the most likely location for finding a mango is the fridge since similar objects such as apple appear there as well. The most likely locations are shown with the orange box.

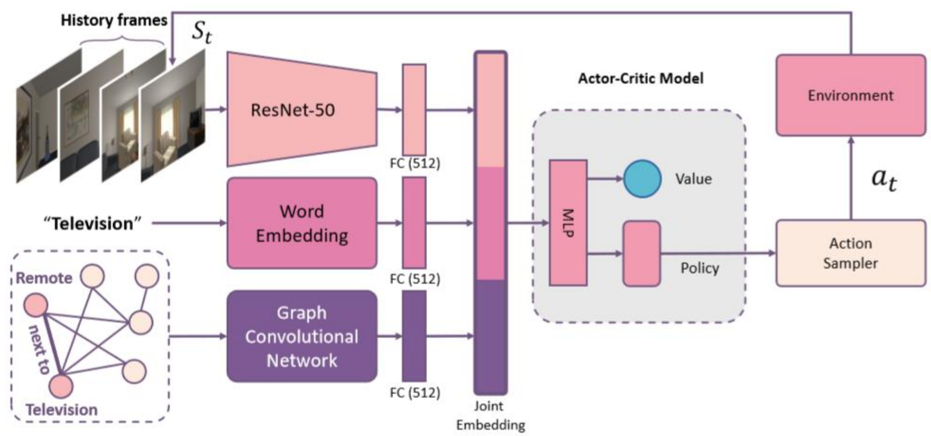


Figure 2: **Overview of the architecture.** Our model to incorporate semantic knowledge into semantic navigation. Specifically, we learn a policy network that decides an action based on the visual features of the current state, the semantic target category feature and the features extracted from the knowledge graph. We extract features from the parts of the knowledge graph that are activated.

Yang et al., "Visual Semantic Navigation using Scene Priors", ICLR 2019

RL for question-answering

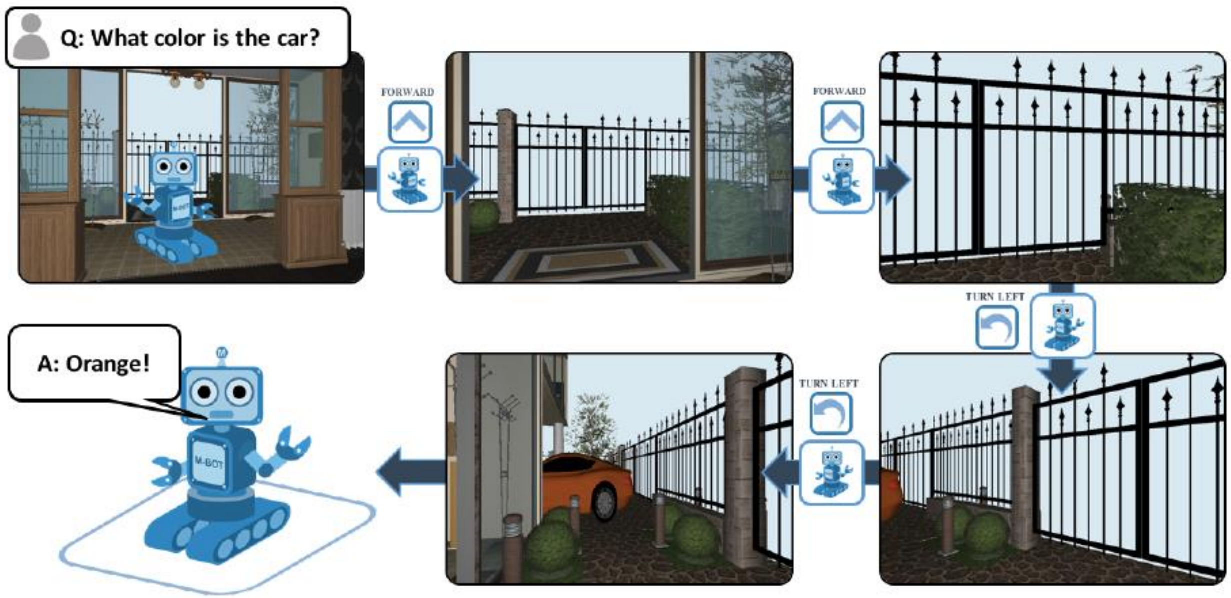


Figure 1: Embodied Question Answering – EmbodiedQA– tasks agents with navigating rich 3D environments in order to answer questions. These agents must jointly learn language understanding, visual reasoning, and goal-driven navigation to succeed.

Das et al., “Embodied Question Answering”, CVPR 2018

RL for question-answering

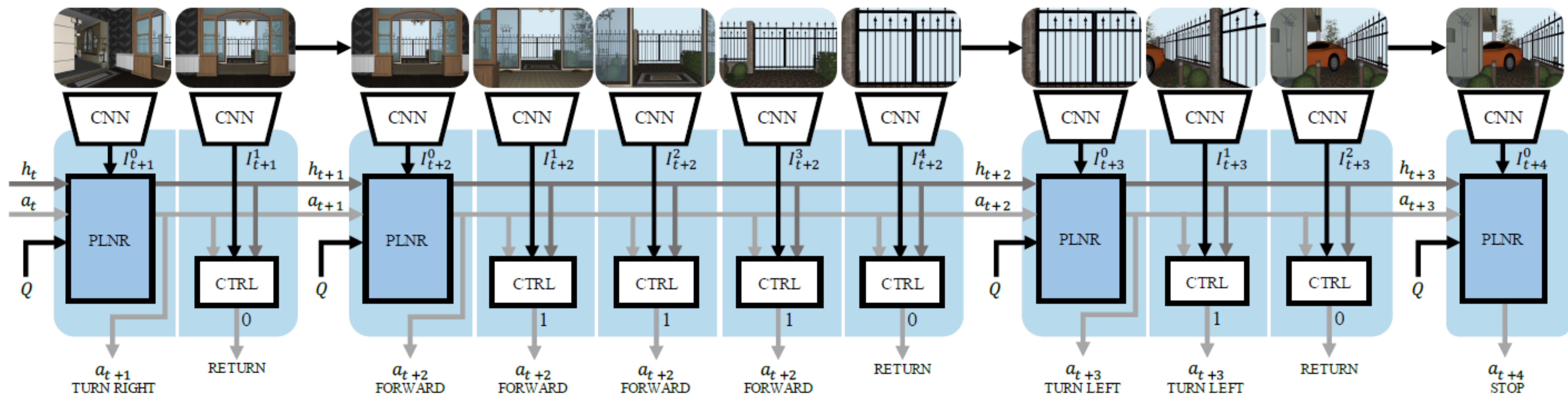


Figure 4: Our PACMAN navigator decomposes navigation into a planner and a controller. The planner selects actions and the controller executes these actions a variable number of times. This enables the planner to operate on shorter timescales, strengthening gradient flows.

Das et al., "Embodied Question Answering", CVPR 2018

RL for object detection

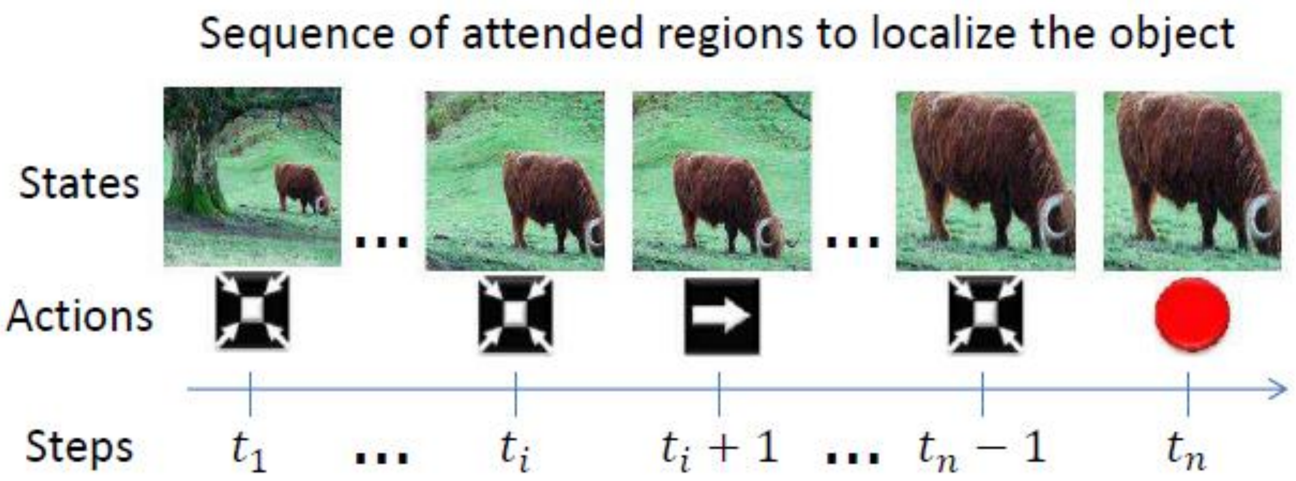


Figure 1. A sequence of actions taken by the proposed algorithm to localize a cow. The algorithm attends regions and decides how to transform the bounding box to progressively localize the object.

Caicedo and Lazebnik, “Active Object Localization with Deep Reinforcement Learning”, ICCV 2015

RL for object detection

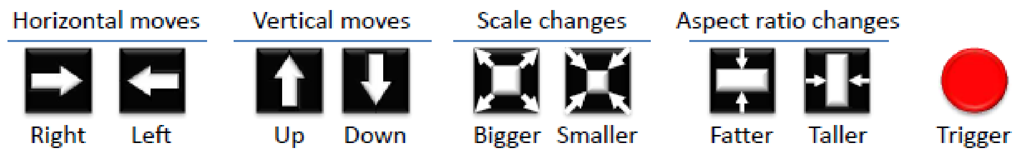
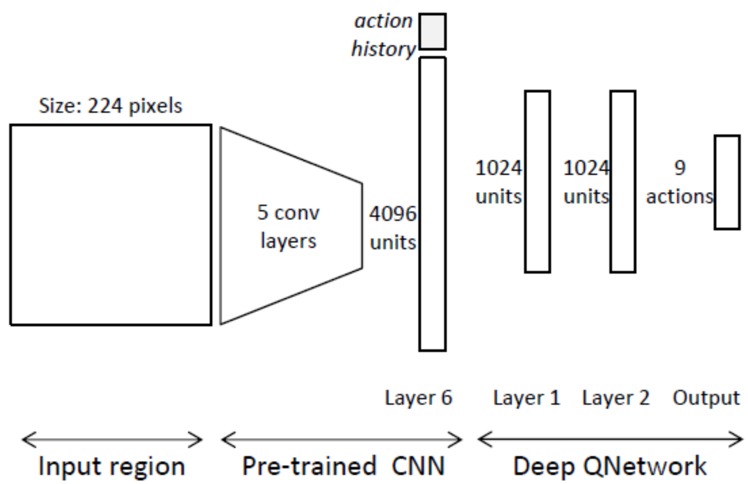


Figure 2. Illustration of the actions in the proposed MDP, giving 4 degrees of freedom to the agent for transforming boxes.

$$R_a(s, s') = sign\left(IoU(b', g) - IoU(b, g)\right) \qquad R_\omega(s, s') = \begin{cases} +\eta & \text{if } IoU(b, g) \geq \tau \\ -\eta & \text{otherwise} \end{cases}$$

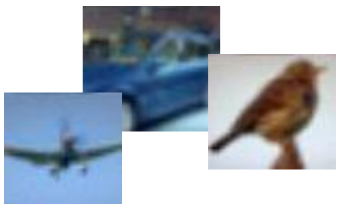


Caicedo and Lazebnik, “Active Object Localization with Deep Reinforcement Learning”, ICCV 2015

Part IV: Generation

- Motivation and taxonomy of methods
- Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)
- Applications and variants of GANs
- Dealing with sparse data, progressive training

Generative Models



Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

Serena Young

Why Generative Models?

- Realistic samples for artwork, super-resolution, colorization, etc.



- Generative models can be used to enhance training datasets with diverse synthetic data
- Generative models of time-series data can be used for simulation

Adapted from Serena Young

Taxonomy of Generative Models

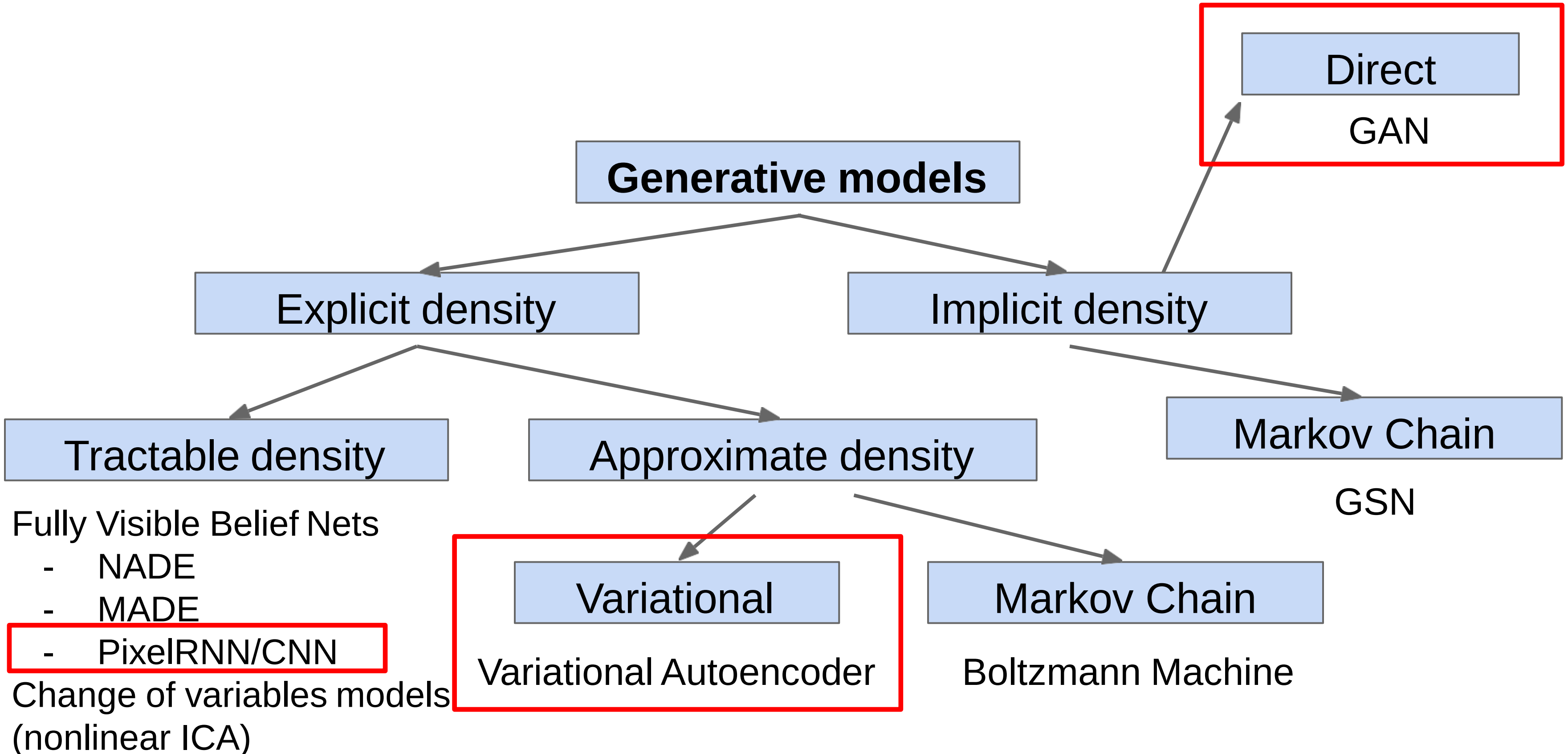


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

PixelRNN and PixelCNN

Fully visible belief network

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Will need to define ordering of "previous pixels"

Then maximize likelihood of training data

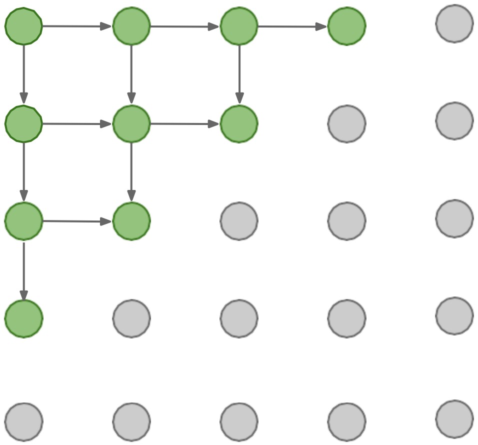
Complex distribution over pixel values => Express using a neural network!

PixelRNN

[van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)



Drawback: sequential generation is slow!

PixelCNN

[van der Oord et al. 2016]

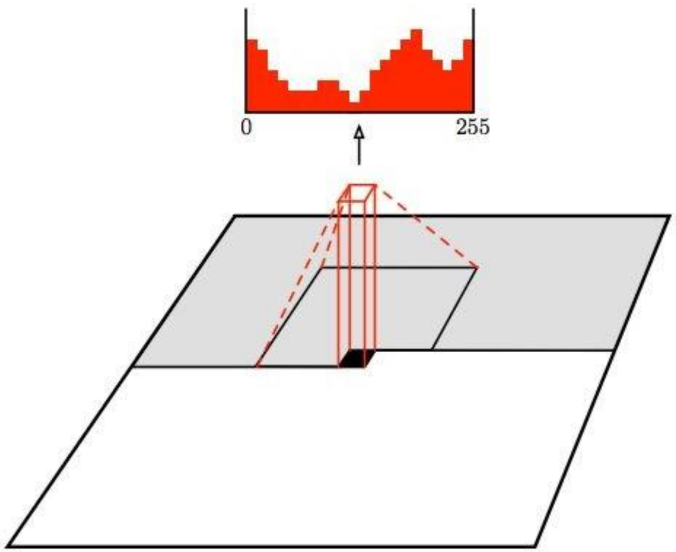
Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training: maximize likelihood of training images

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Softmax loss at each pixel



Training is faster than PixelRNN (can parallelize convolutions since context region values known from training images)

Generation must still proceed sequentially => still slow

Variational Autoencoders (VAEs)

Some background: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

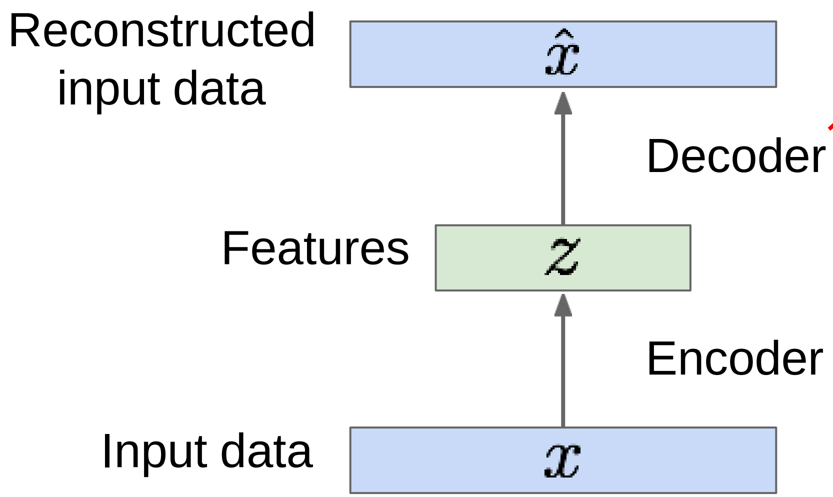
Features z

Encoder

Input data x

Some background: Autoencoders

How to learn this feature representation?
Train such that features can be used to reconstruct original data
"Autoencoding" - encoding itself



Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN (upconv)

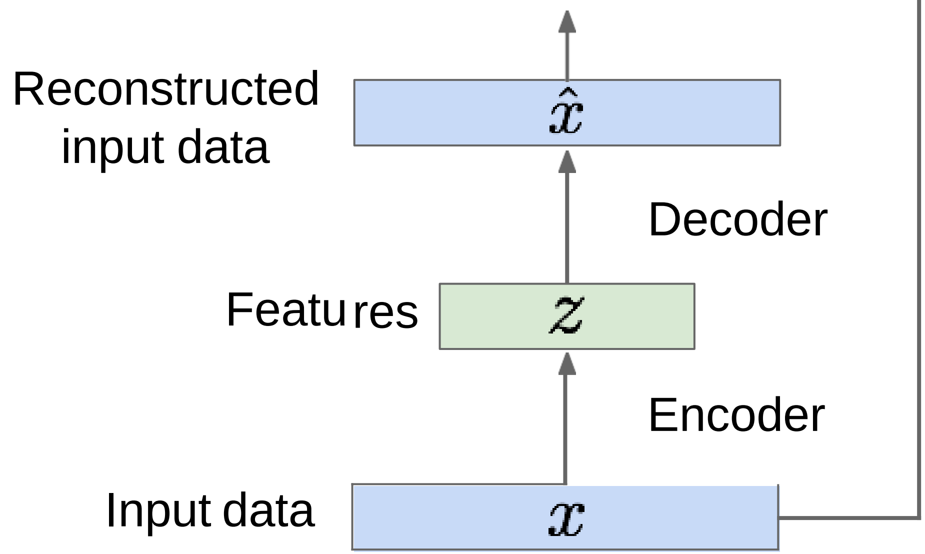
Some background: Autoencoders

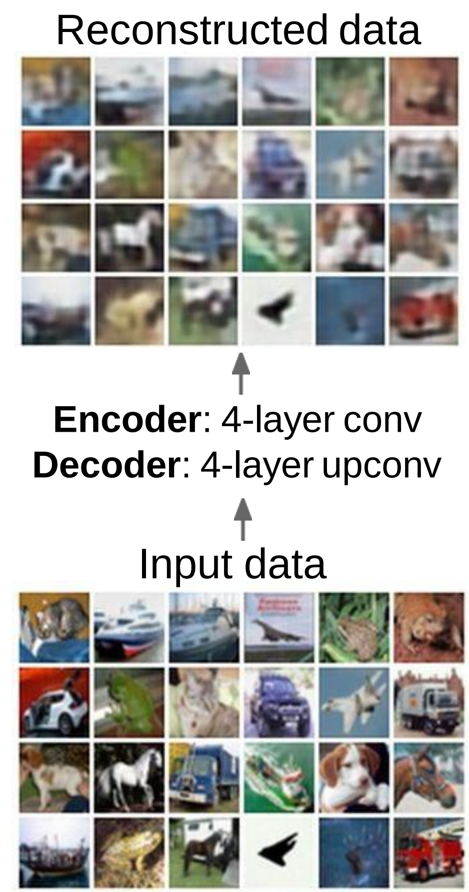
Train such that features can be used to reconstruct original data

Doesn't use labels!

L2 Loss function:

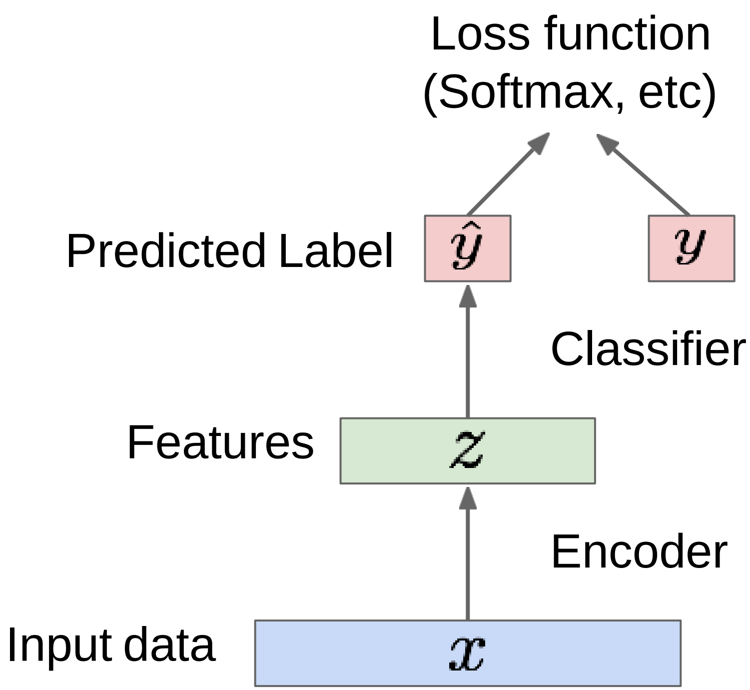
$$\|x - \hat{x}\|^2$$





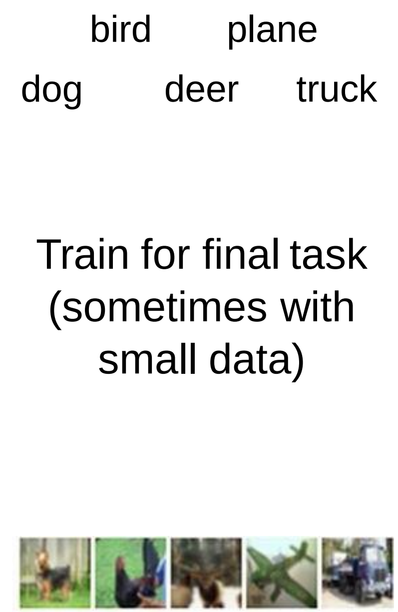
Some background: Autoencoders

Encoder can be used to initialize a **supervised** model

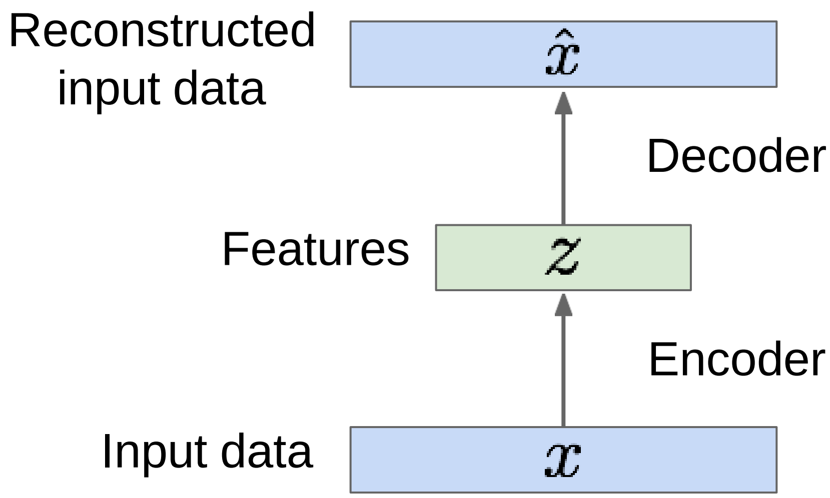


Fine-tune encoder jointly with classifier

Train for final task (sometimes with small data)



Some background: Autoencoders



Features capture factors of variation in training data. Can we *generate* new images from an autoencoder?

Adapted from Serena Young

Variational Autoencoders

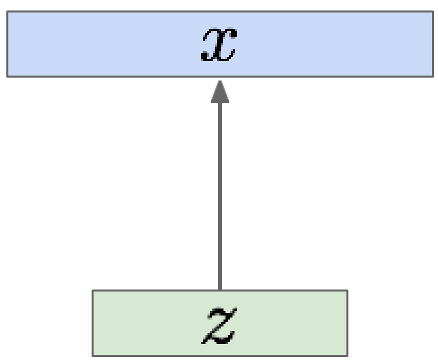
Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation **z**

Intuition (remember from autoencoders!): **x** is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

Sample from conditional

$p_{\theta^*}(x \mid z^{(i)})$



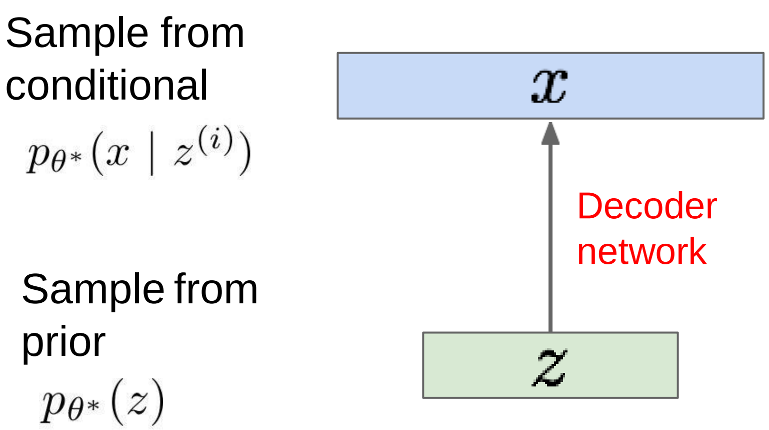
Sample from prior

$p_{\theta^*}(z)$

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Adapted from Serena Young

Variational Autoencoders



We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian.

Conditional p(x|z) is complex (generates image) => represent with neural network

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Adapted from Serena Young

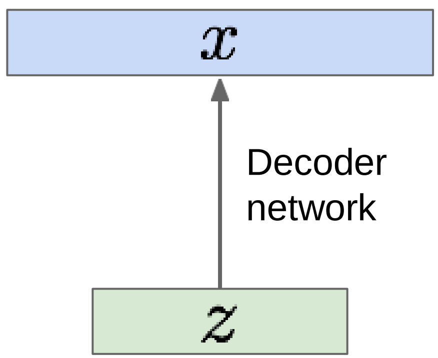
Variational Autoencoders

Sample from conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from prior

$p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Learn model parameters to maximize likelihood of training data

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Now with latent z

Q: What is the problem with this?

Intractable!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Adapted from Serena Young

Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$

Simple Gaussian prior

Intractable to compute
p(x|z) for every z!

Decoder neural network

Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$

Intractable data likelihood

- Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that *approximates* $p_\theta(z|x)$
- This allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize – overviewed briefly on next few slides (feel free to skip when reviewing)

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Adapted from Serena Young

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

We want to maximize the data likelihood

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Serena Young

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

We want to maximize the data likelihood

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \boxed{\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\boxed{\geq 0}}$$

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

Reconstruct the input data

Make approximate posterior distribution close to prior

$$\boxed{\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)}$$

Variational lower bound ("ELBO")

$$\boxed{\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)}$$

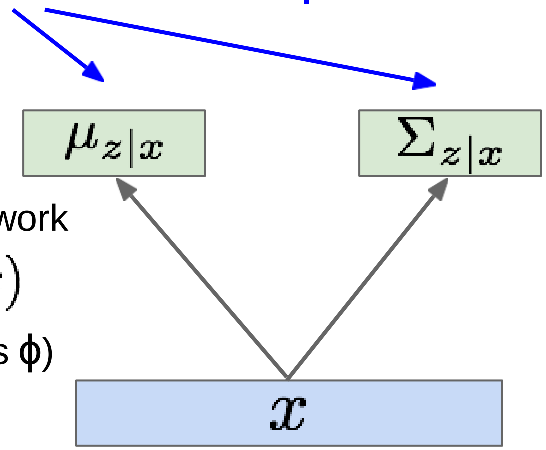
Training: Maximize lower bound

Serena Young

Variational Autoencoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

Mean and covariance of **z | x**

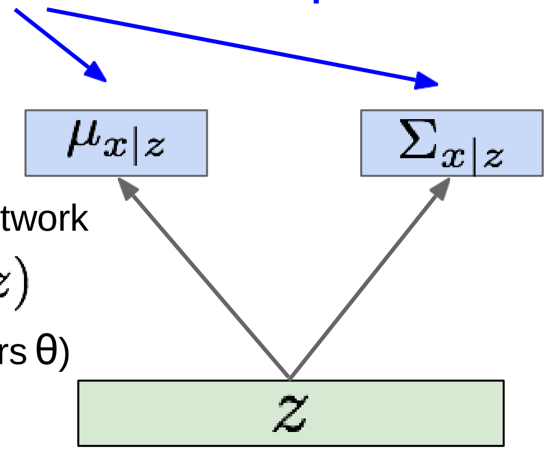


Encoder network

$q_\phi(z|x)$

(parameters ϕ)

Mean and covariance of **x | z**



Decoder network

$p_\theta(x|z)$

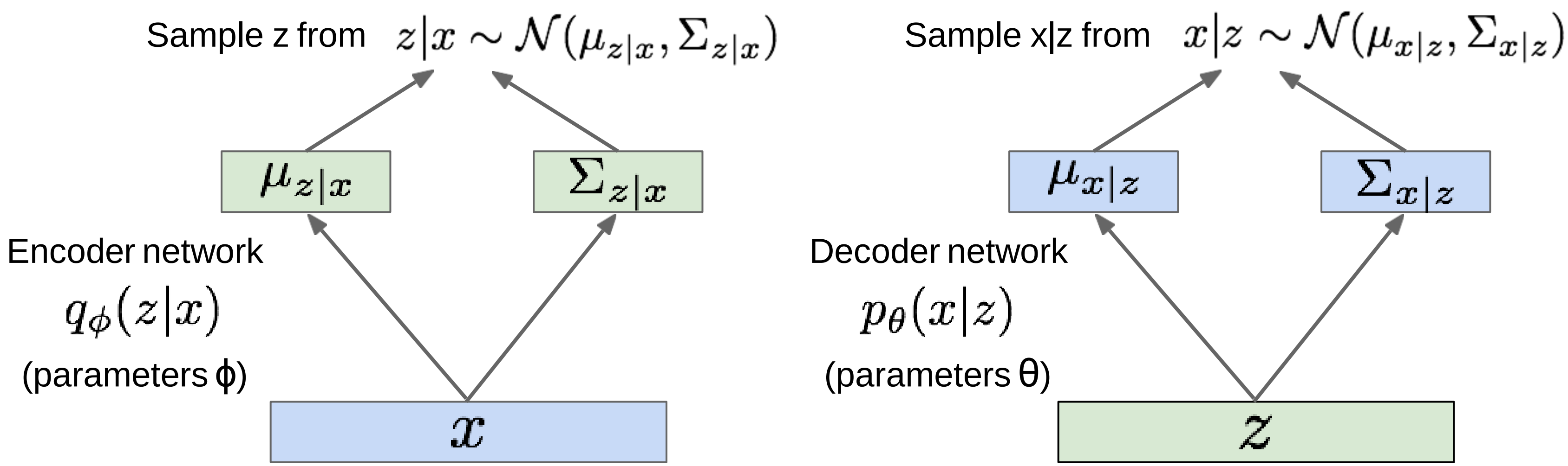
(parameters θ)

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Adapted from Serena Young

Variational Autoencoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



Encoder and decoder networks also called "recognition"/"inference" and "generation" networks

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

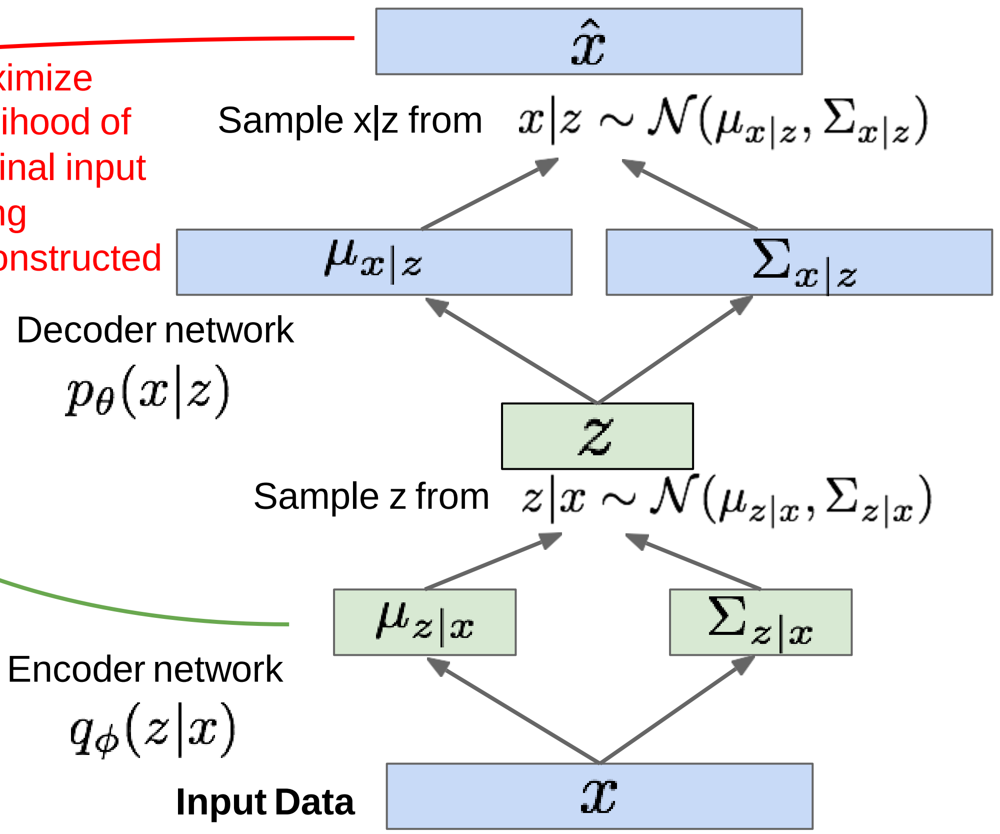
Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior

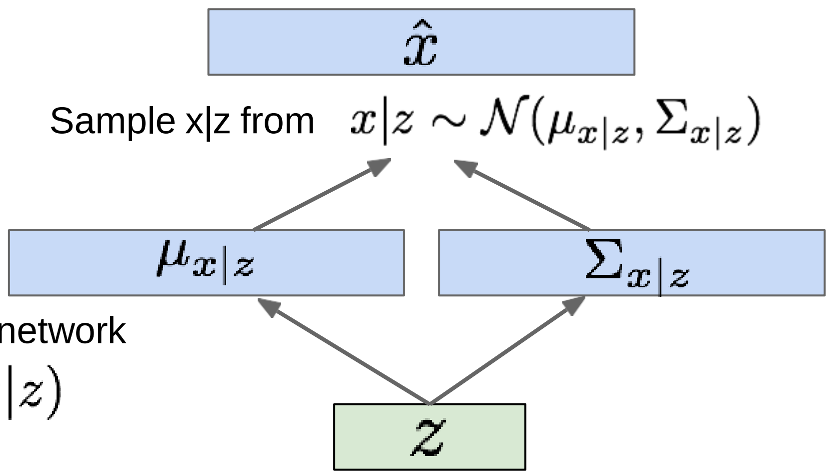
For every minibatch of input data: compute this forward pass, and then backprop!



Serena Young

VAEs: Generating Data

Sample z from prior
Use decoder network



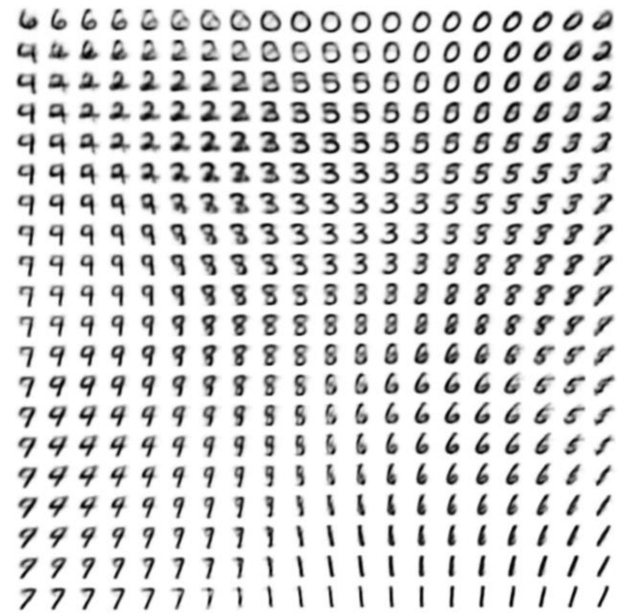
Decoder network
$p_\theta(x|z)$

Sample z from $z \sim \mathcal{N}(0, I)$

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Data manifold for 2-d **z**

Vary $\mathbf{z_1}$



Vary $\mathbf{z_2}$

Adapted from Serena Young

VAEs: Generating Data

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

Also good feature representation that can be computed using $q_\phi(z|x)$!

Degree of smile

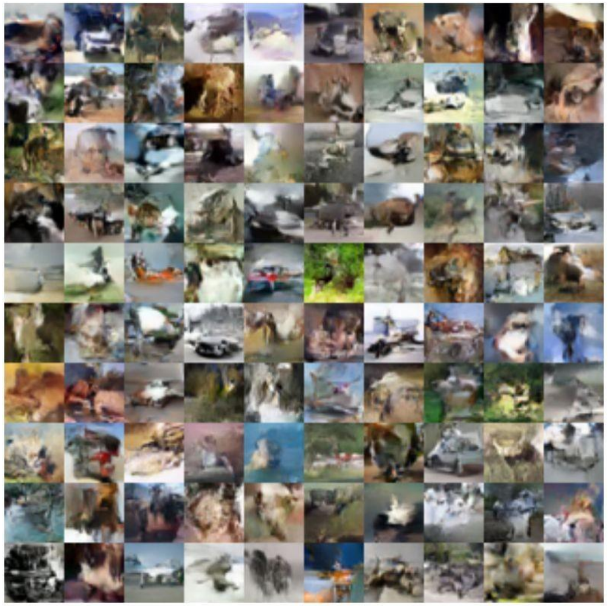
Vary $\mathbf{z}_1$

Vary $\mathbf{z}_2$

Head pose

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VAEs: Generating Data



32x32 CIFAR-10

Labeled Faces in the Wild

Generating with little data for ads

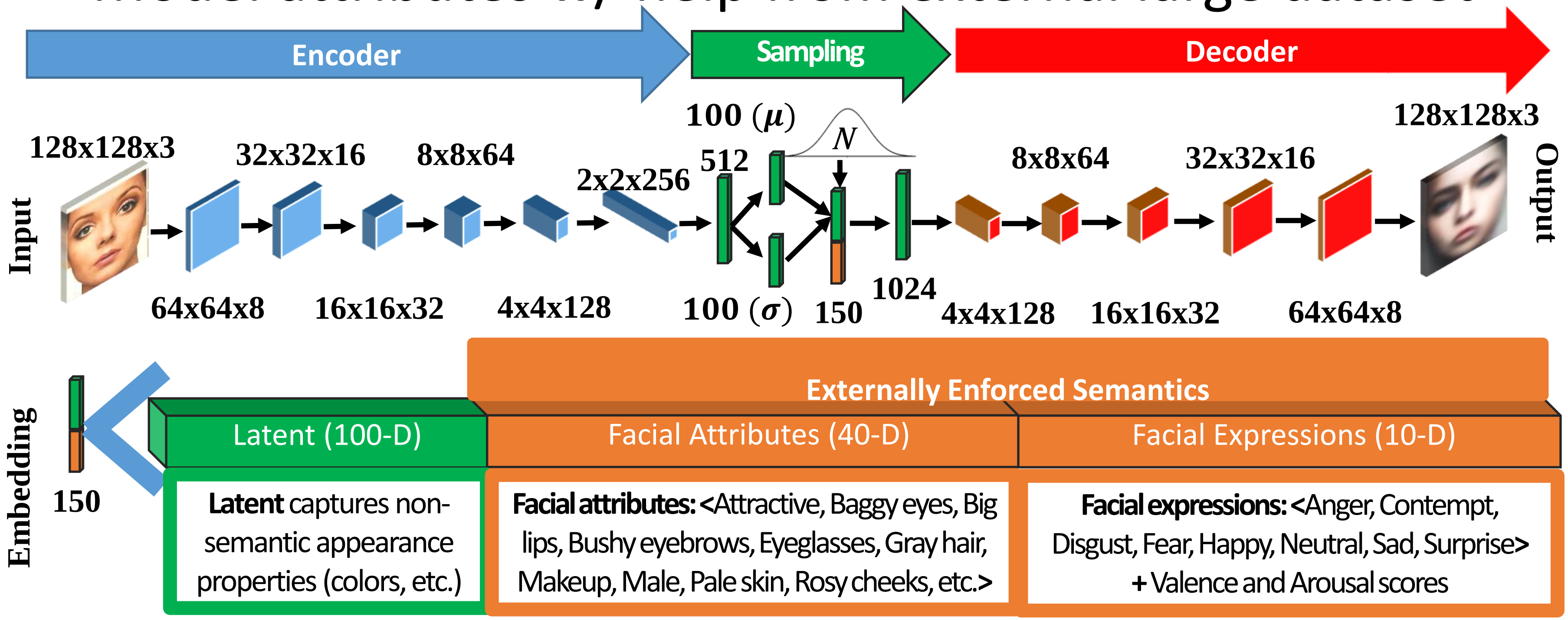
- Faces are persuasive and carry meaning/sentiment

- We learn to generate faces appropriate for each ad category
- Because our data is so diverse yet limited in count, standard approaches that directly model pixel distributions don't work well

Thomas and Kovashka, BMVC 2018

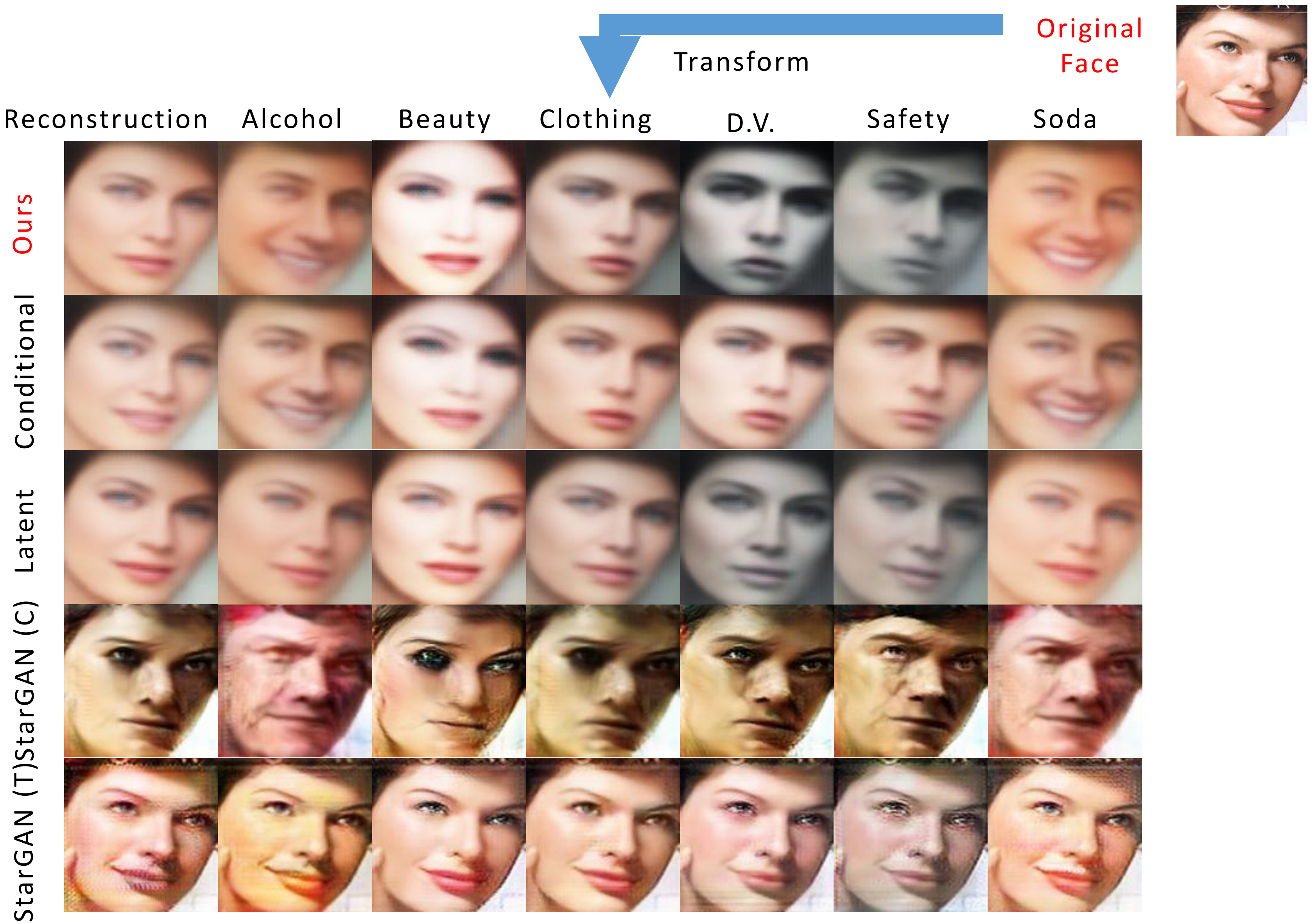
Generating with little data for ads

- Instead we model the distribution over *attributes* for each category (e.g. domestic violence ads contain “black eye”, beauty contains “red lips”)
- Generate an image with the attributes of an ad class
- Model attributes w/ help from external large dataset



Thomas and Kovashka, BMVC 2018

Generating with little data for ads



Thomas and Kovashka, BMVC 2018

Variational Autoencoders

Probabilistic spin to traditional autoencoders => allows generating data
Defines an intractable density => derive and optimize a lower bound

Pros:
- Principled approach to generative models
- Allows inference of q(z|x), can be useful feature representation for other tasks

Cons:
- Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don't work with any explicit density function!
Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game

Generative Adversarial Networks

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

Generative Adversarial Networks

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

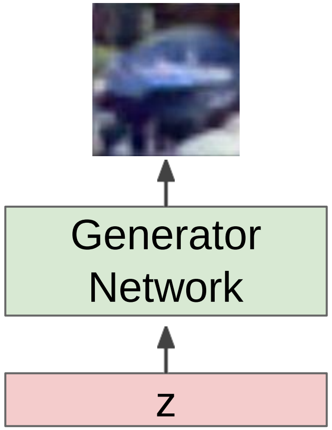
Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

A: A neural network!

Output: Sample from training distribution



Input: Random noise

Training GANs: Two-player game

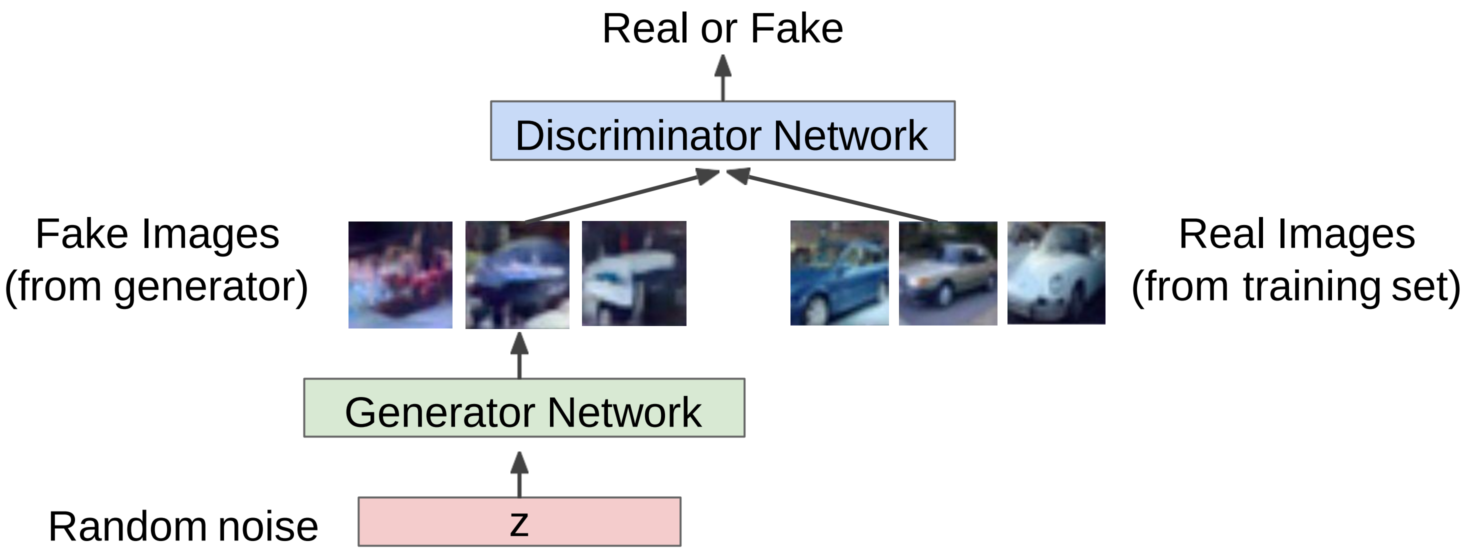
Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Training GANs: Two-player game

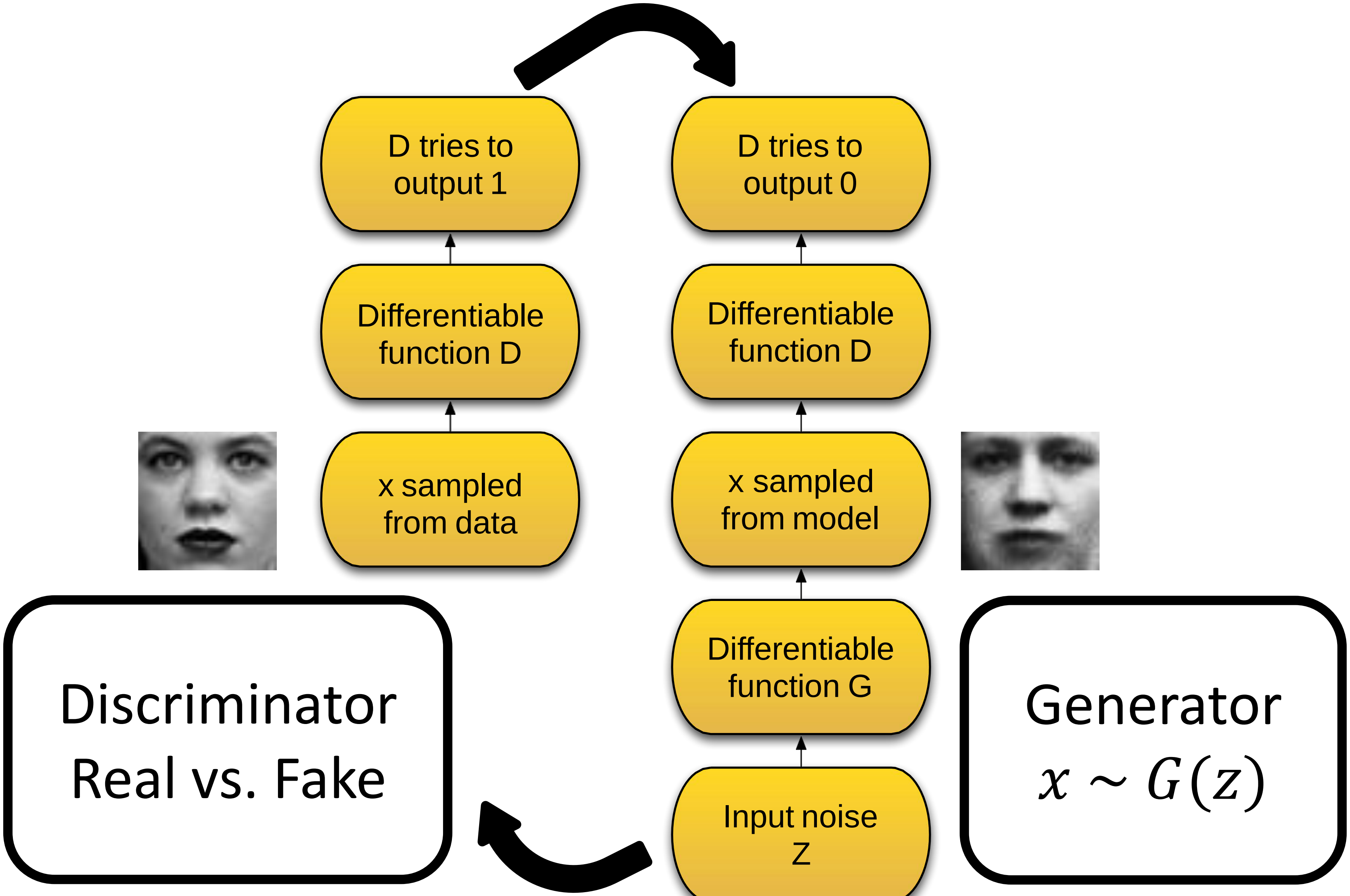
Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



Serena Young

Adversarial Networks Framework



Ian Goodfellow

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\boxed{\min_{\theta_g}}\boxed{\max_{\theta_d}}\left[\mathbb{E}_{x\sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z\sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}})\right]$$

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

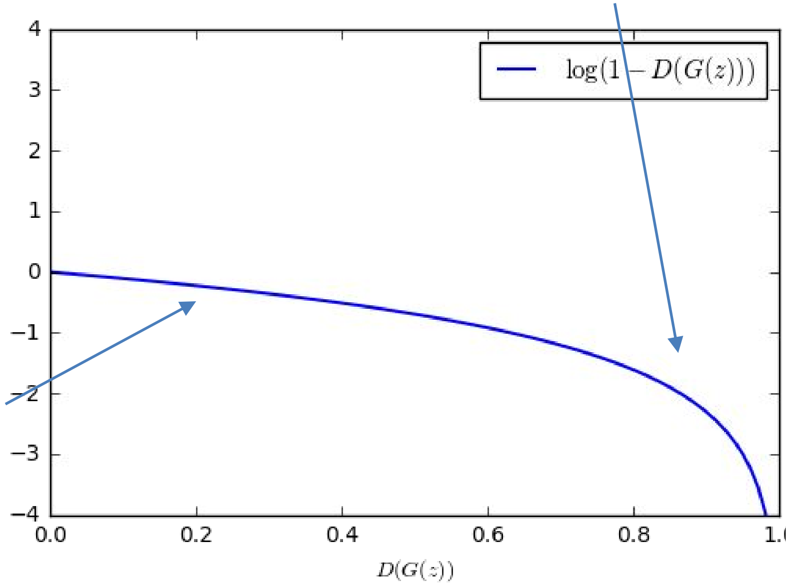
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



Adapted from Serena Young

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

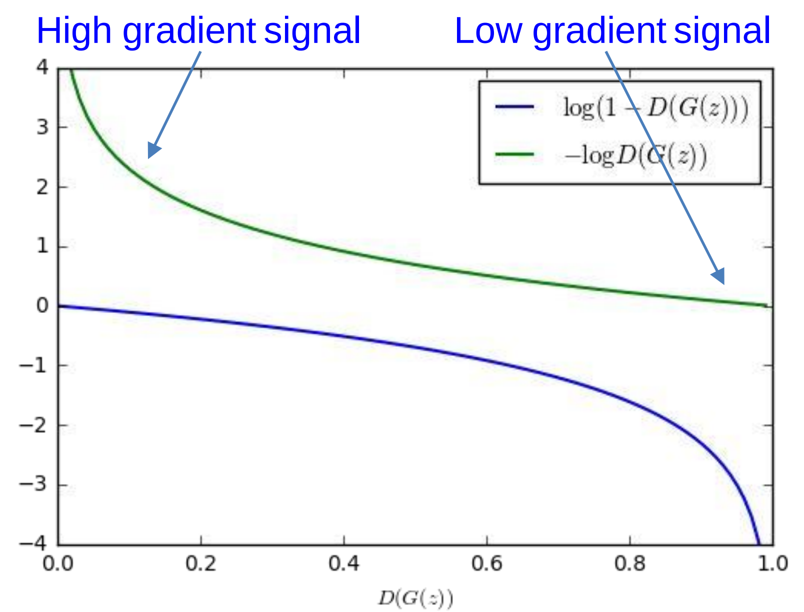
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



Adapted from Serena Young

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Putting it together: GAN training algorithm

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

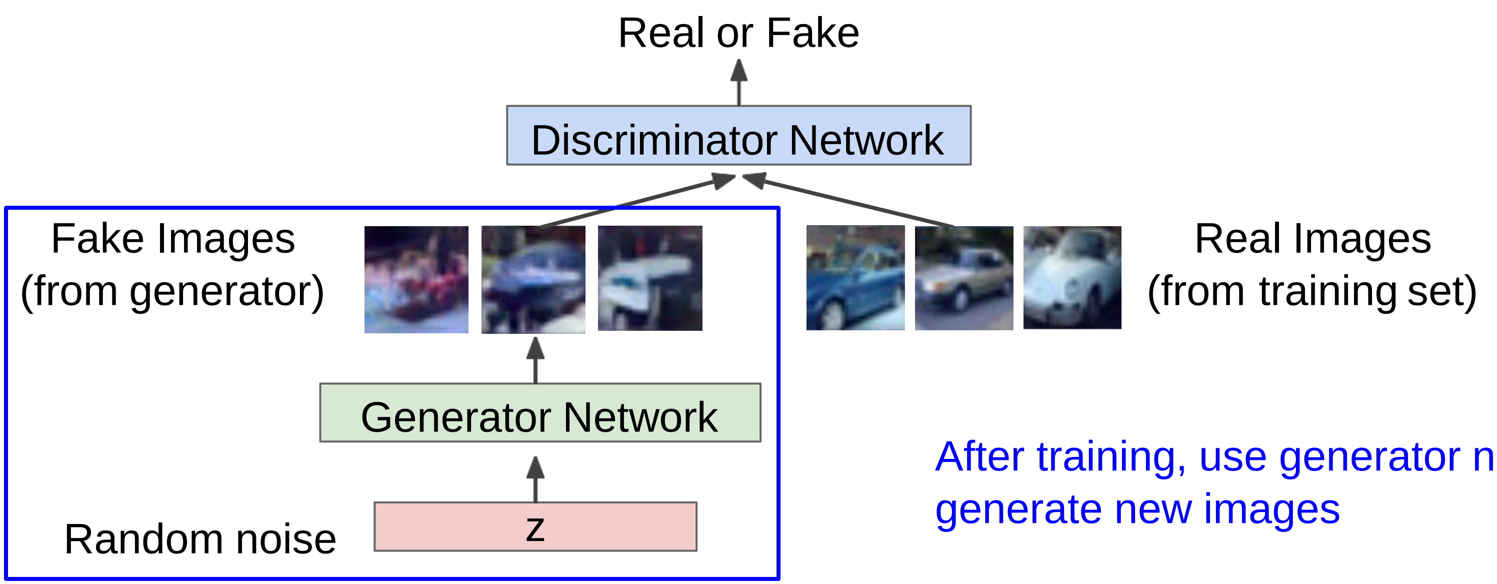
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



After training, use generator network to generate new images

Alternative loss functions

Name	Paper Link	Value Function
GAN	Arxiv	$L_D^{GAN} = E[\log(D(x))] + E[\log(1 - D(G(z)))]$ $L_G^{GAN} = E[\log(D(G(z)))]$
LSGAN	Arxiv	$L_D^{LSGAN} = E[(D(x) - 1)^2] + E[D(G(z))^2]$ $L_G^{LSGAN} = E[(D(G(z)) - 1)^2]$
WGAN	Arxiv	$L_D^{WGAN} = E[D(x)] - E[D(G(z))]$ $L_G^{WGAN} = E[D(G(z))]$ $W_D \leftarrow clip\_by\_value(W_D, -0.01, 0.01)$
WGAN_GP	Arxiv	$L_D^{WGAN\_GP} = L_D^{WGAN} + \lambda E[(\lvert\nabla D(\alpha x - (1 - \alpha G(z)))\rvert - 1)^2]$ $L_G^{WGAN\_GP} = L_G^{WGAN}$
DRAGAN	Arxiv	$L_D^{DRAGAN} = L_D^{GAN} + \lambda E[(\lvert\nabla D(\alpha x - (1 - \alpha x_p))\rvert - 1)^2]$ $L_G^{DRAGAN} = L_G^{GAN}$
CGAN	Arxiv	$L_D^{CGAN} = E[\log(D(x, c))] + E[\log(1 - D(G(z), c))]$ $L_G^{CGAN} = E[\log(D(G(z), c))]$
infoGAN	Arxiv	$L_{D,Q}^{infoGAN} = L_D^{GAN} - \lambda L_I(c, c')$ $L_G^{infoGAN} = L_G^{GAN} - \lambda L_I(c, c')$
ACGAN	Arxiv	$L_{D,Q}^{ACGAN} = L_D^{GAN} + E[P(class = c \mid x)] + E[P(class = c \mid G(z))]$ $L_G^{ACGAN} = L_G^{GAN} + E[P(class = c \mid G(z))]$
EBGAN	Arxiv	$L_D^{EBGAN} = D_{AE}(x) + \max(0, m - D_{AE}(G(z)))$ $L_G^{EBGAN} = D_{AE}(G(z)) + \lambda \cdot PT$
BEGAN	Arxiv	$L_D^{BEGAN} = D_{AE}(x) - k_t D_{AE}(G(z))$ $L_G^{BEGAN} = D_{AE}(G(z))$ $k_{t+1} = k_t + \lambda(\gamma D_{AE}(x) - D_{AE}(G(z)))$

https://github.com/hwalsuklee/tensorflow-generative-model-collections
https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

GAN training is challenging

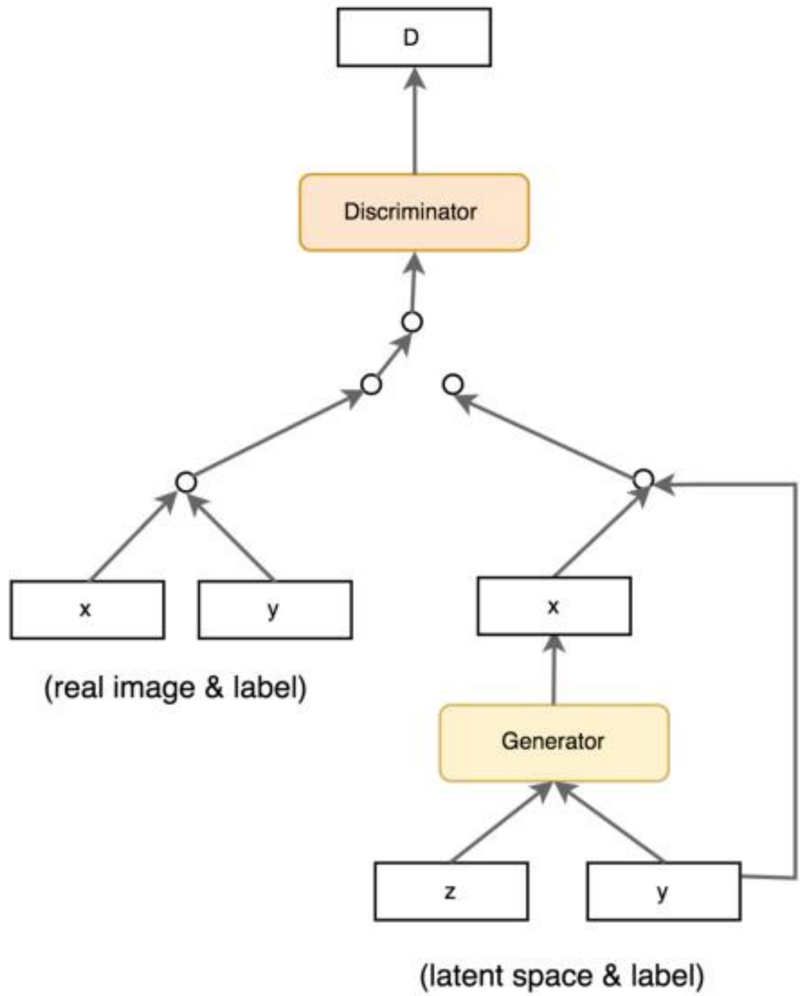
- Vanishing gradient – when discriminator is very good
- Mode collapse – too little diversity in the samples generated
- Lack of convergence because hard to reach Nash equilibrium
- Loss metric doesn't always correspond to image quality; Frechet Inception Distance (FID) is a decent choice

Tips and tricks

- Use batchnorm, ReLU
- Regularize norm of gradients
- Use one of the new loss functions
- Add noise to inputs or labels
- Append image similarity to avoid mode collapse
- Use labels, extra info when available (CGAN)
- …

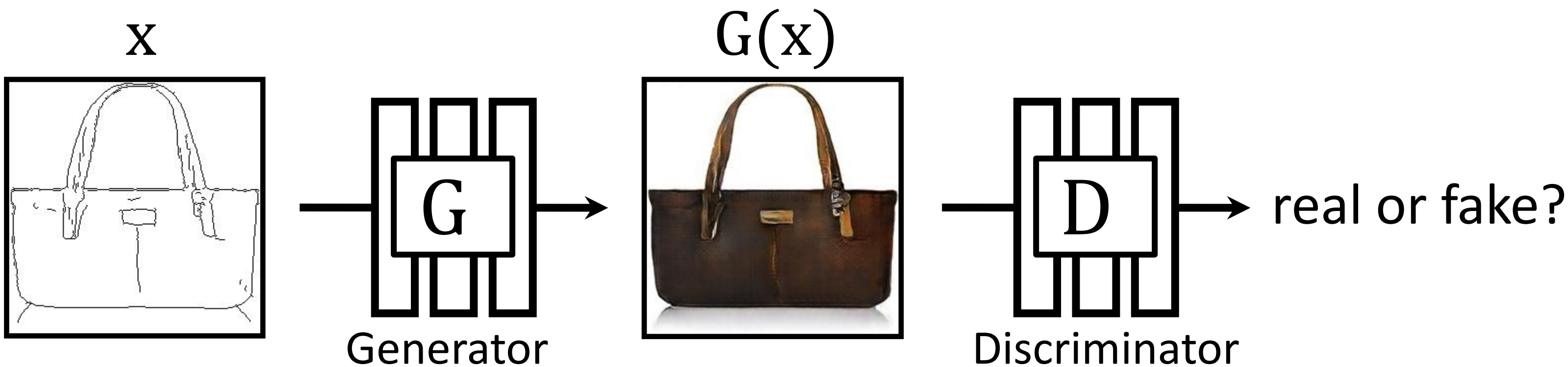
https://github.com/soumith/talks/blob/master/2017-ICCV_Venice/How_To_Train_a_GAN.pdf
https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b

Conditional GANs



https://medium.com/@jonathan_hui/gan-cgan-infogan-using-labels-to-improve-gan-8ba4de5f9c3d

GANs

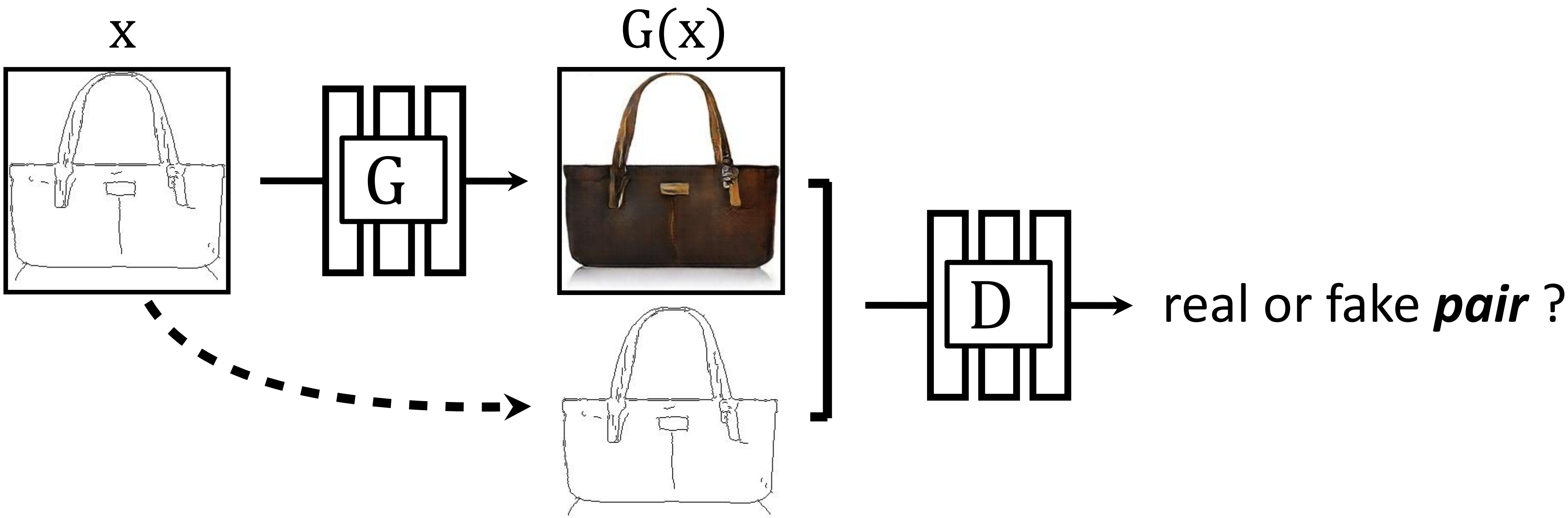


G: generate fake samples that can fool D

D: classify fake samples vs. real images

[Goodfellow et al. 2014]

Conditional GANs



Adapted from Jun-Yan Zhu

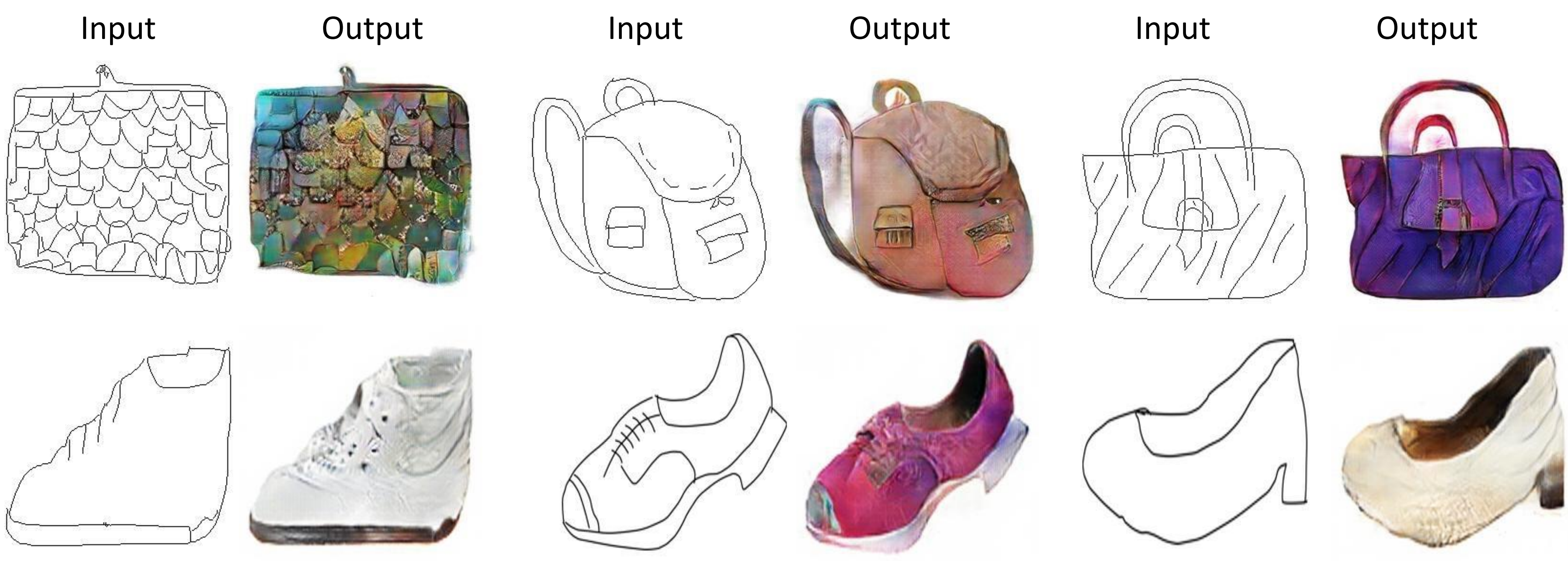
Edges → Images

Input Output Input Output Input Output

Edges from [Xie & Tu, 2015]

Pix2pix / CycleGAN

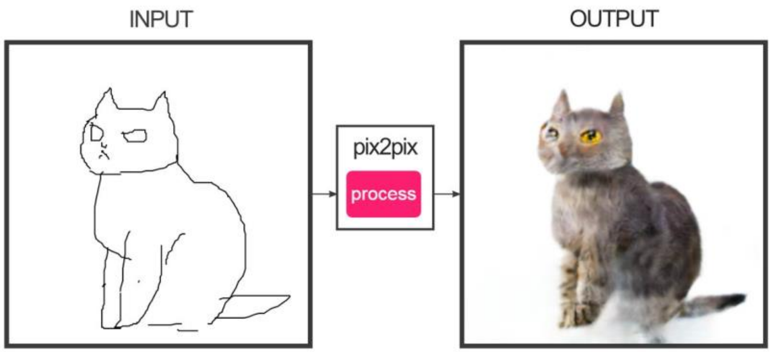
Sketches → Images



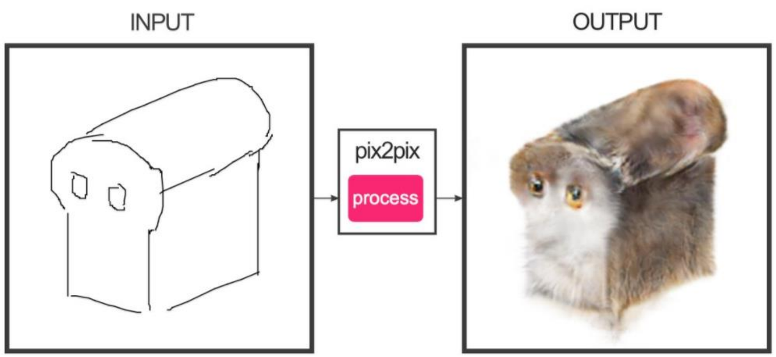
Trained on Edges → Images

Data from [Eitz, Hays, Alexa, 2012]

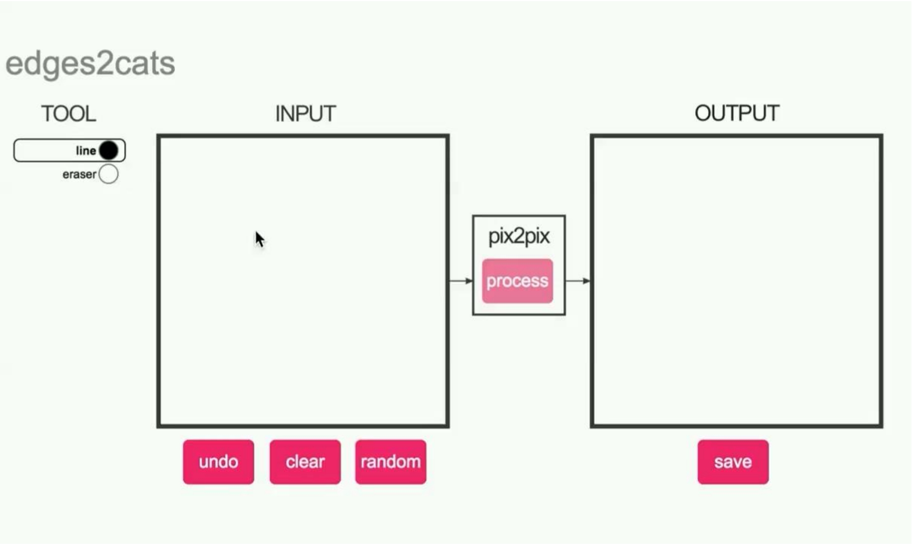
#edges2cats [Christopher Hesse]



@gods_tail



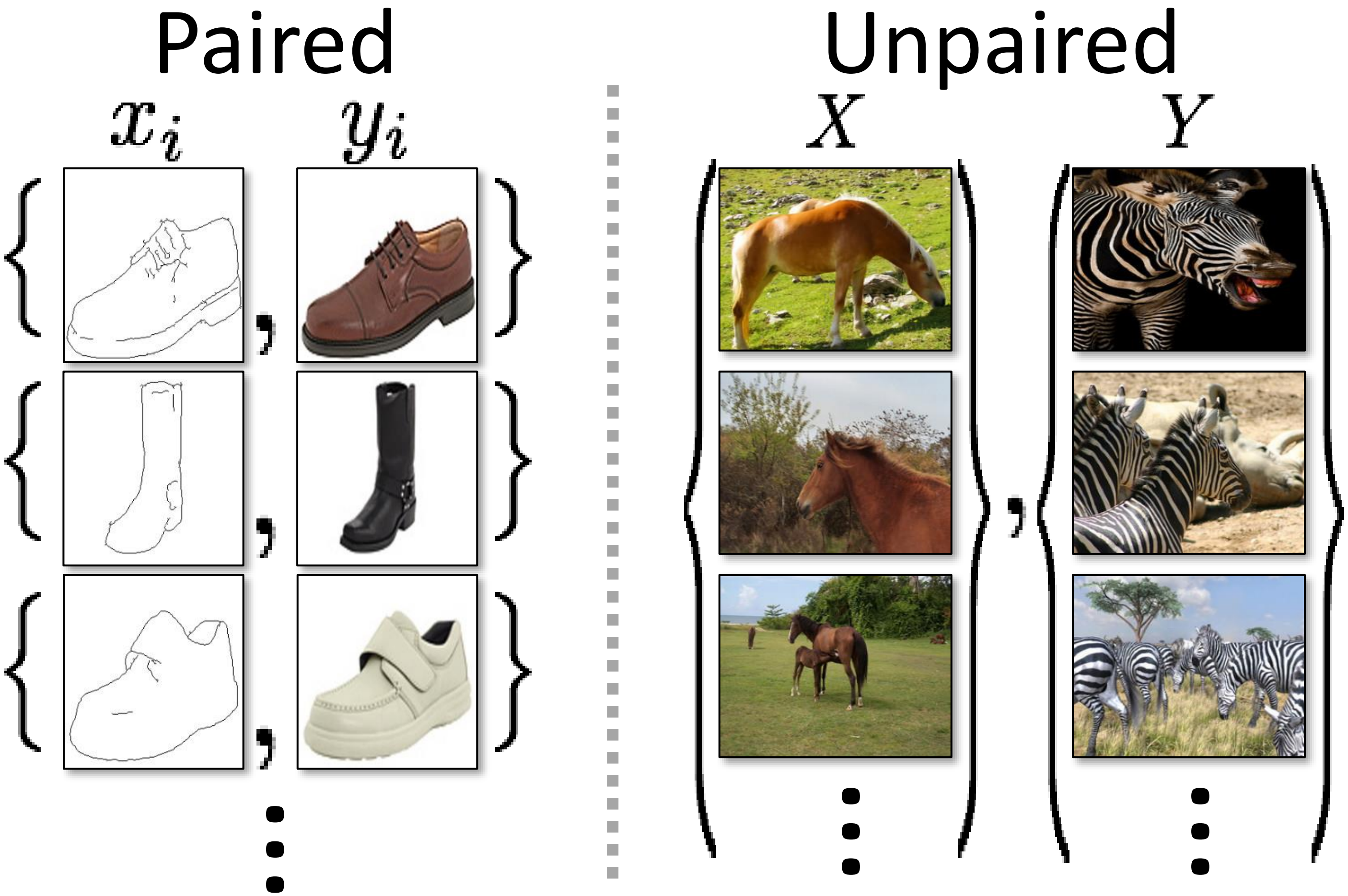
Ivy Tasi @ivymyt



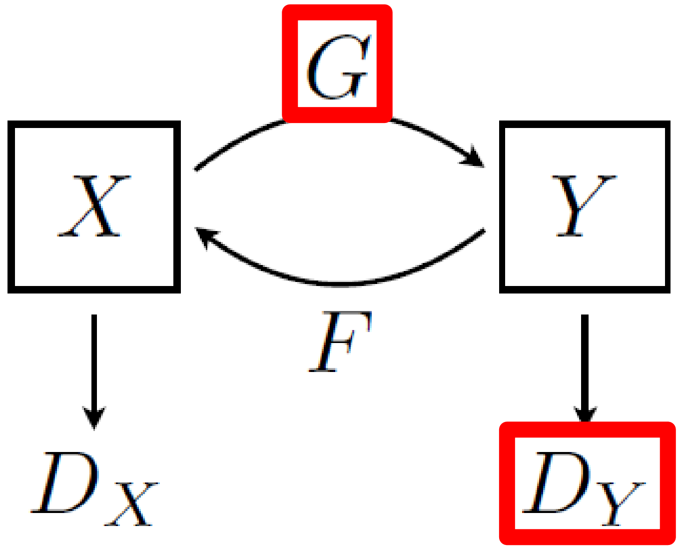
@matthematician

Vitaly Vidmirov @vvid

https://affinelayer.com/pixsrv/


Paired
x_i
y_i
Unpaired
X
Y

Cycle Consistency

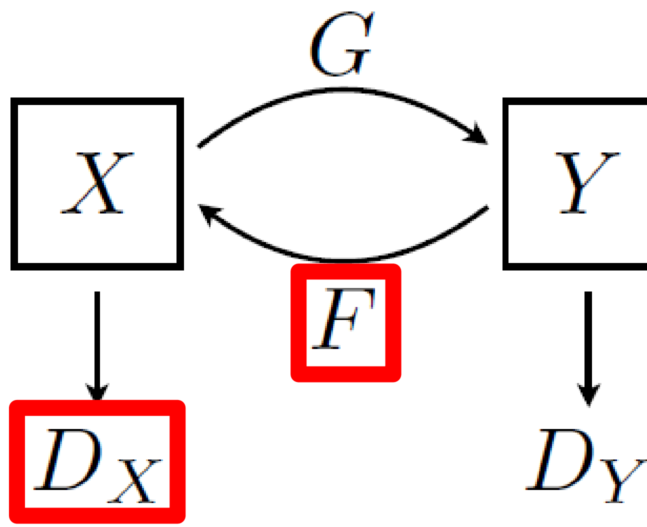


Discriminator $\mathrm{D_Y}$: $L_{GAN}(G(x), y)$

Real zebras vs. generated zebras

Zhu et al., "[Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks]", ICCV 2017

Cycle Consistency

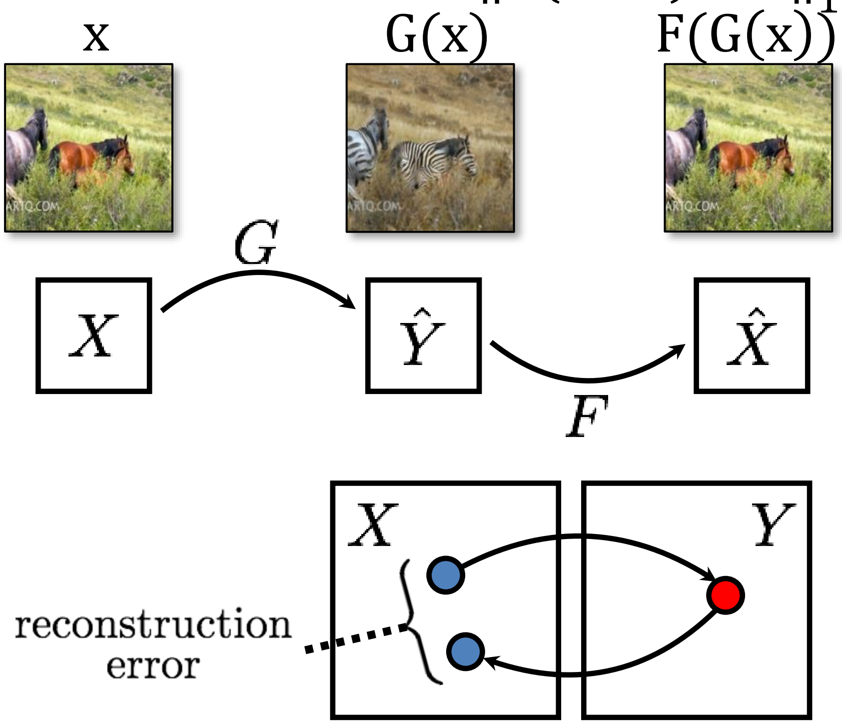


Discriminator D_Y: $L_{GAN}(G(x), y)$
Real zebras vs. generated zebras
Discriminator D_X: $L_{GAN}(F(y), x)$
Real horses vs. generated horses

Zhu et al., "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks", ICCV 2017

Cycle Consistency

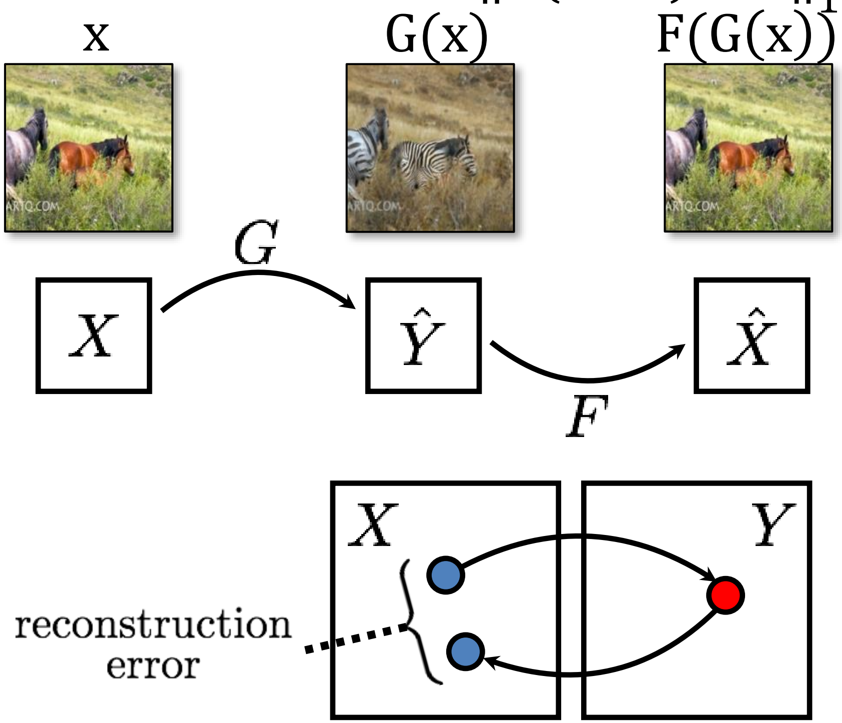
Forward cycle loss: $\left\| F(G(x)) - x \right\|_1$



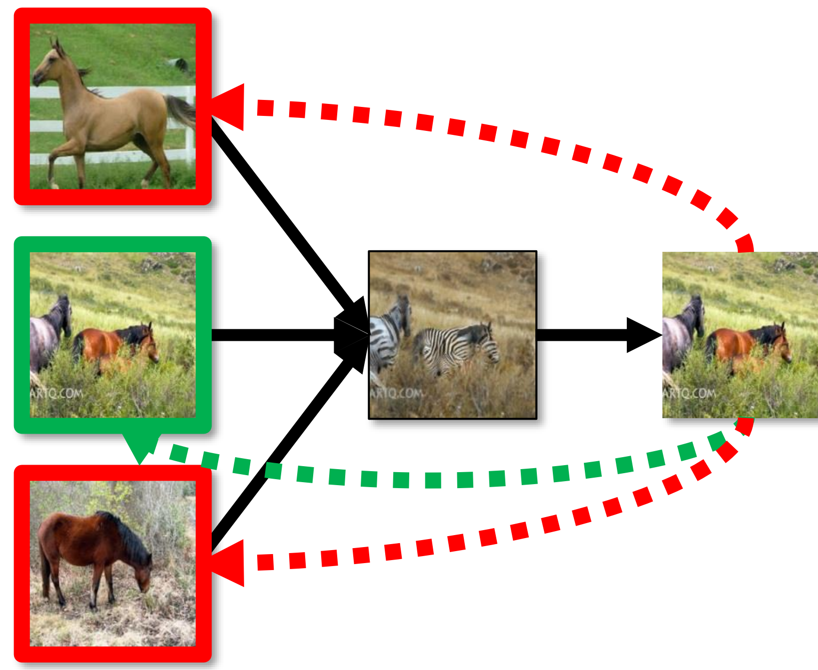
Zhu et al., "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks", ICCV 2017

Cycle Consistency

Forward cycle loss: $\left\|F(G(x)) - x\right\|_1$



Small Large cycle loss



Helps cope with mode collapse

Zhu et al., "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks", ICCV 2017

Training Details: Objective

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

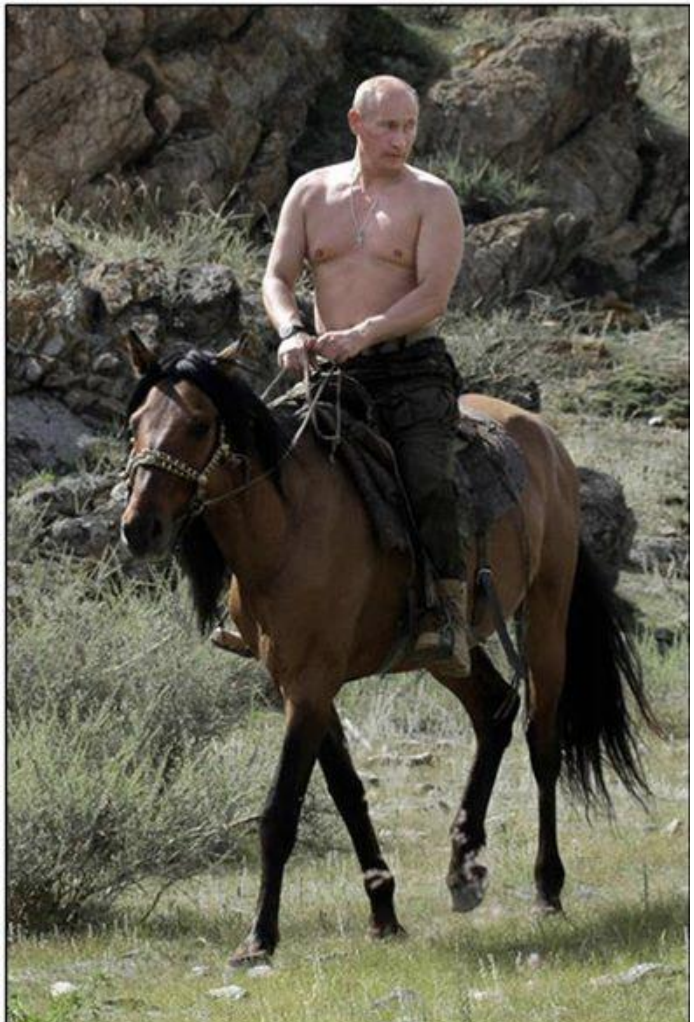
Zhu et al., "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks", ICCV 2017

Pix2pix / CycleGAN

Pix2pix / CycleGAN

Pix2pix / CycleGAN

Pix2pix / CycleGAN



Pix2pix / CycleGAN

Pix2pix / CycleGAN

Celebrities Who Never Existed

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

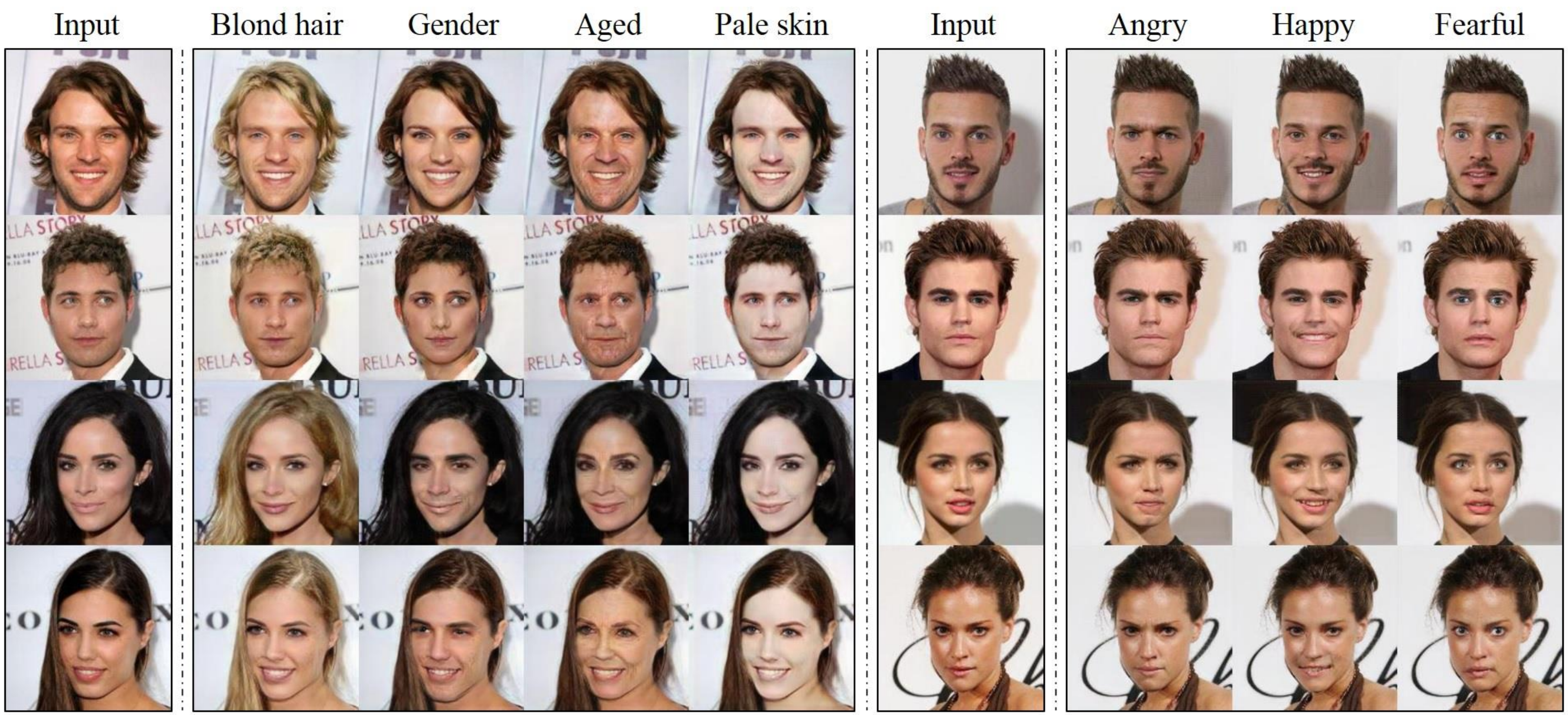
Creative Adversarial Networks

CAN: Top ranked by human subjects

(Elgammal et al., 2017)

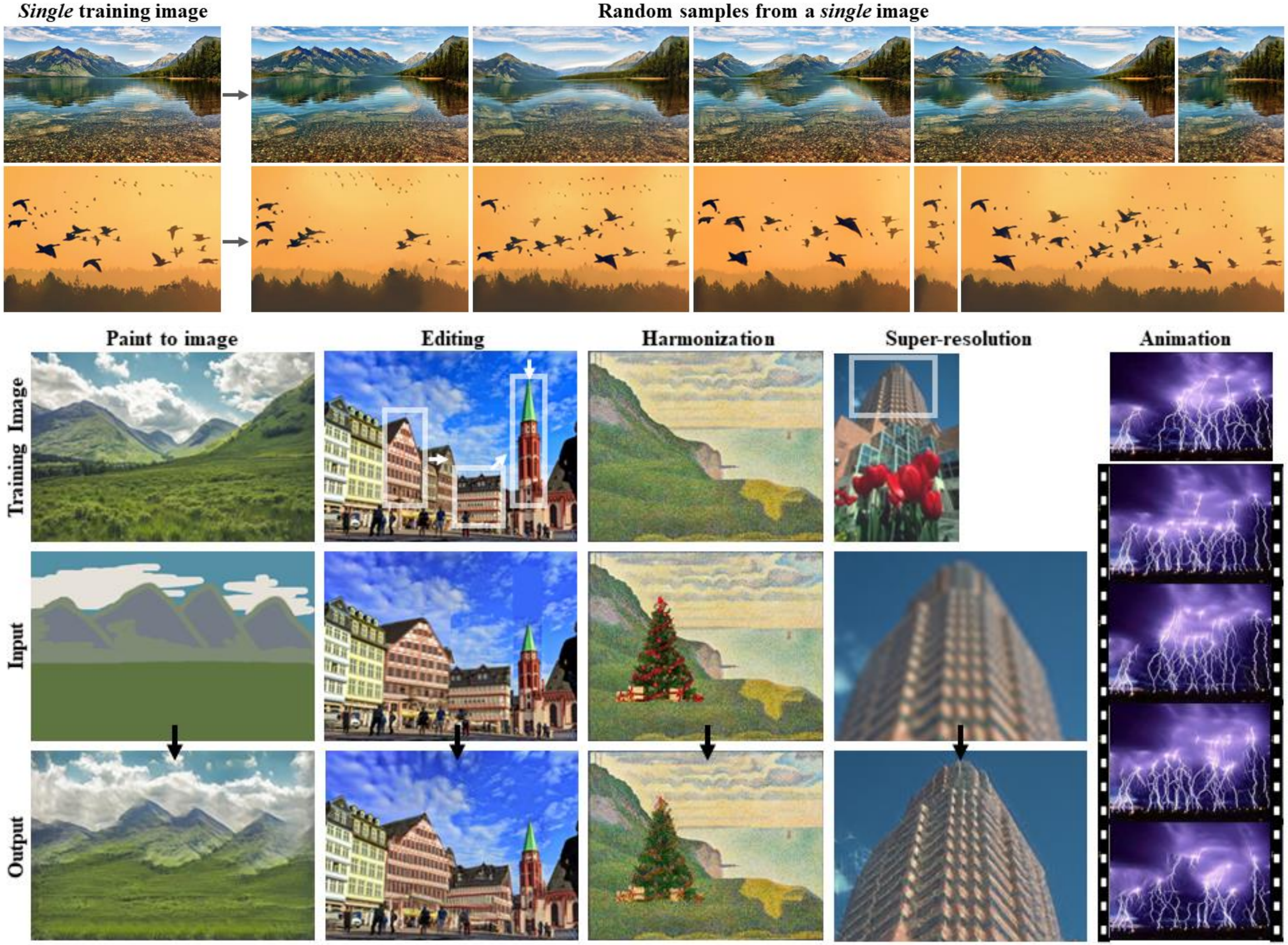
Ian Goodfellow

StarGAN



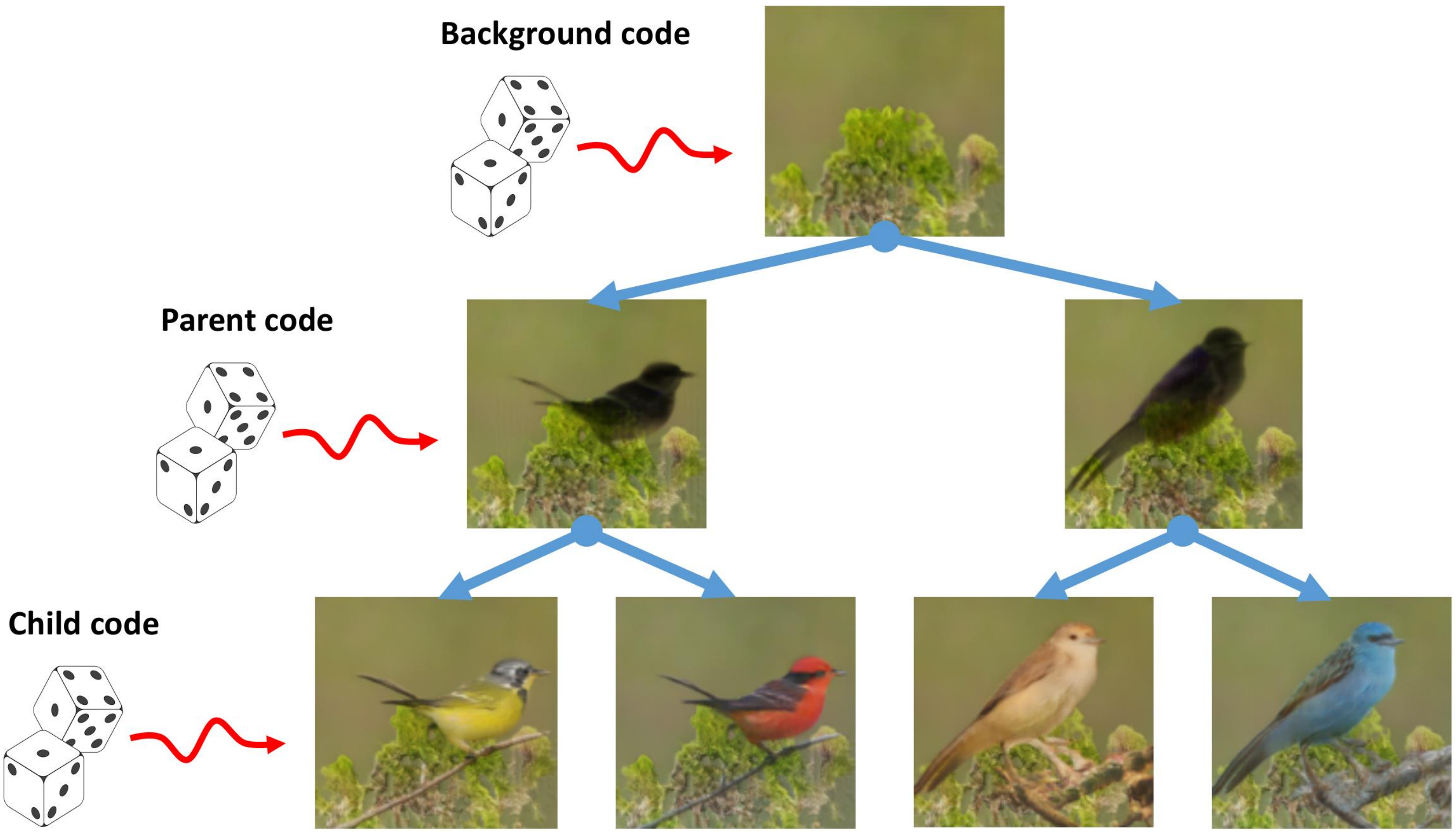
Choi et al., “StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation”, CVPR 2018

SinGAN



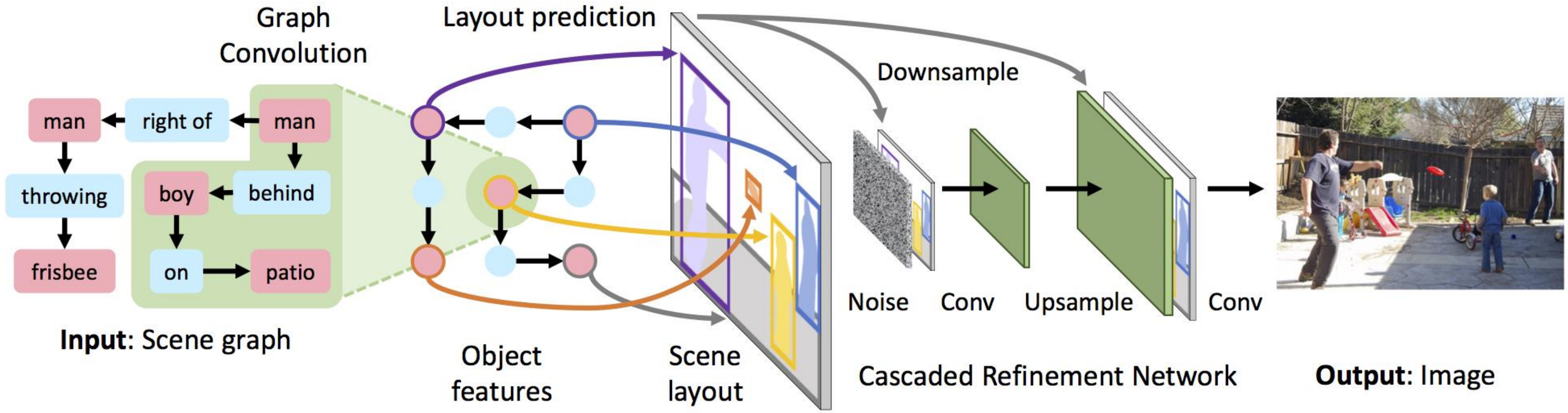
Shaham et al., "SinGAN: Learning a Generative Model from a Single Natural Image", ICCV 2019

Stagewise generation



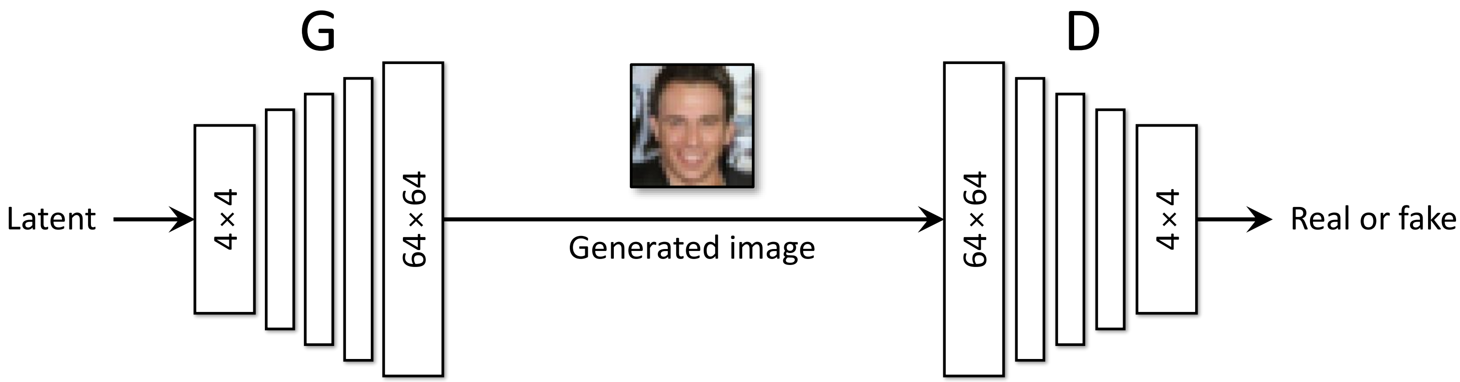
Singh et al., "[FineGAN: Unsupervised Hierarchical Disentanglement for Fine-Grained Object Generation and Discovery]()", CVPR 2019

Stagewise generation



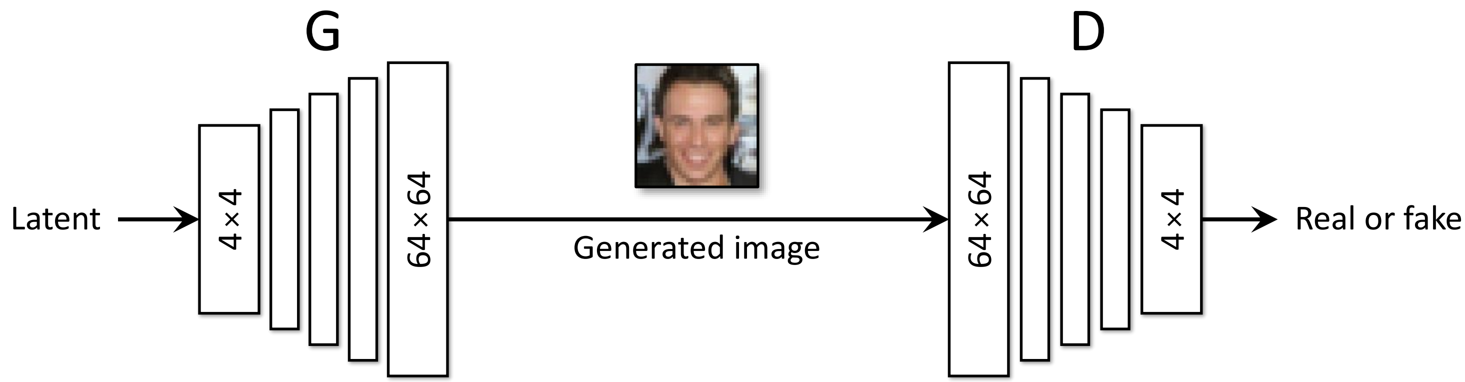
Johnson et al., "Image Generation from Scene Graphs", CVPR 2018

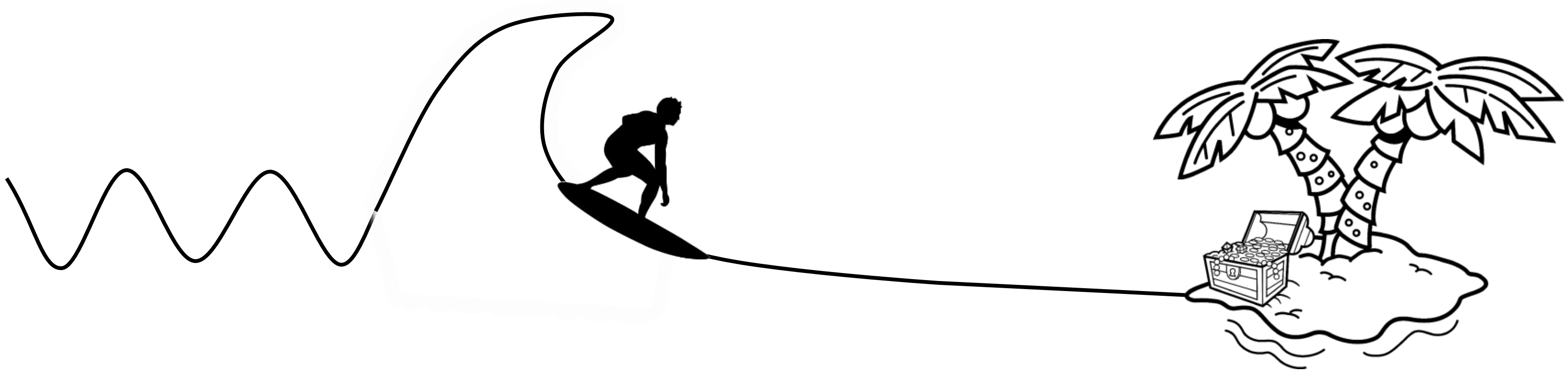
Progressive generation



Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

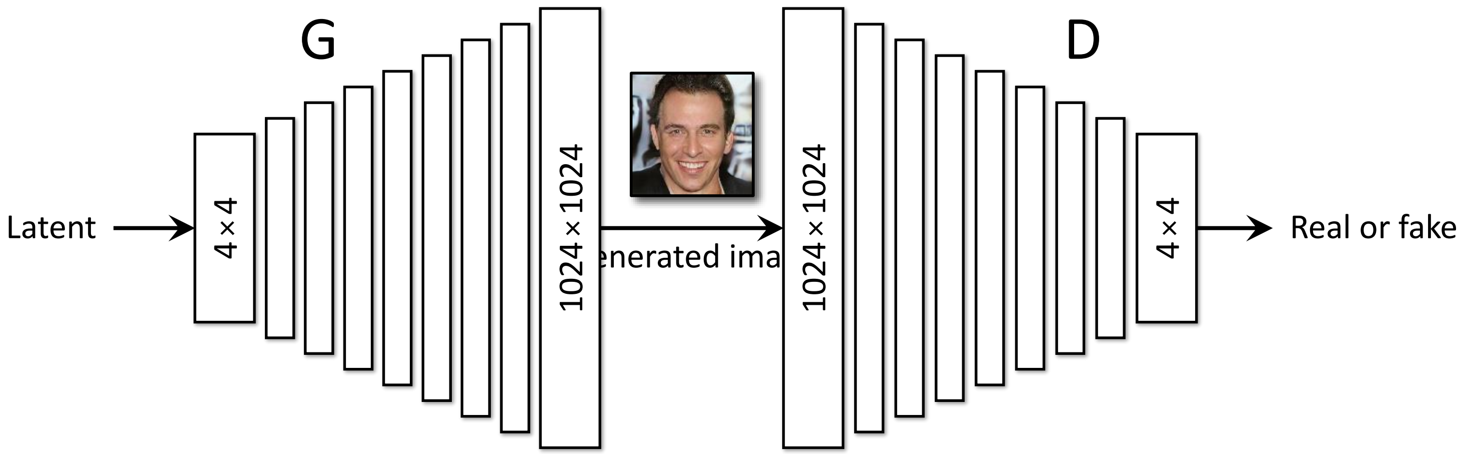
Progressive generation

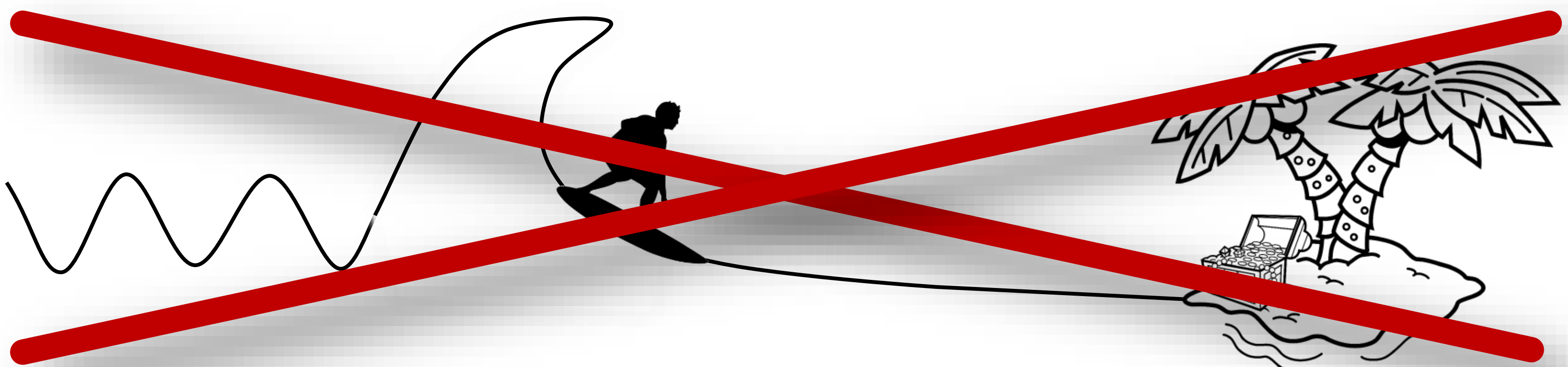




Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

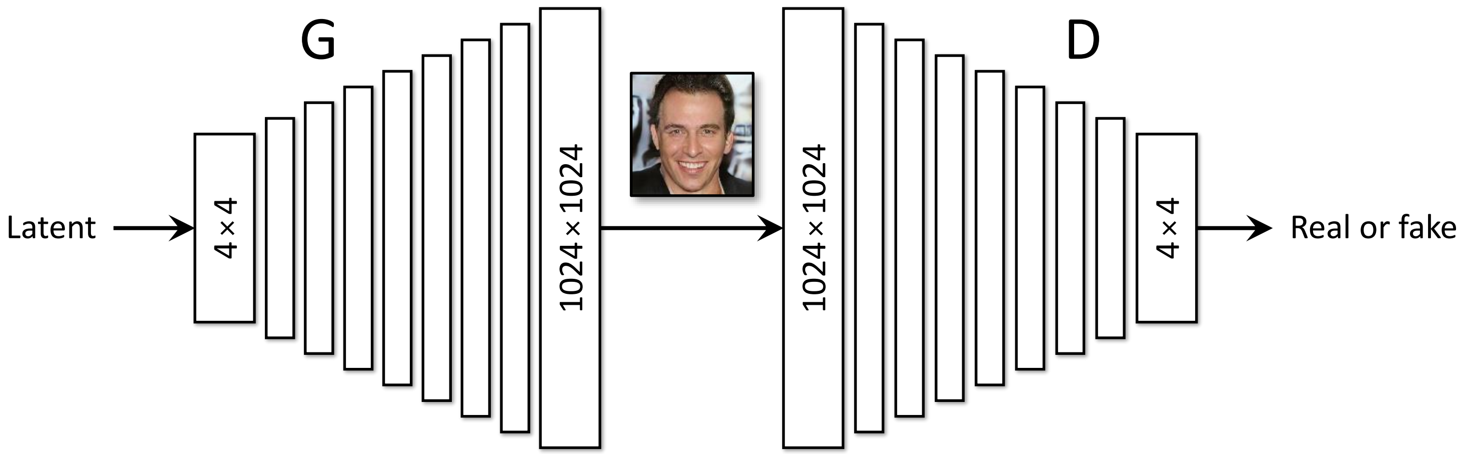
Progressive generation

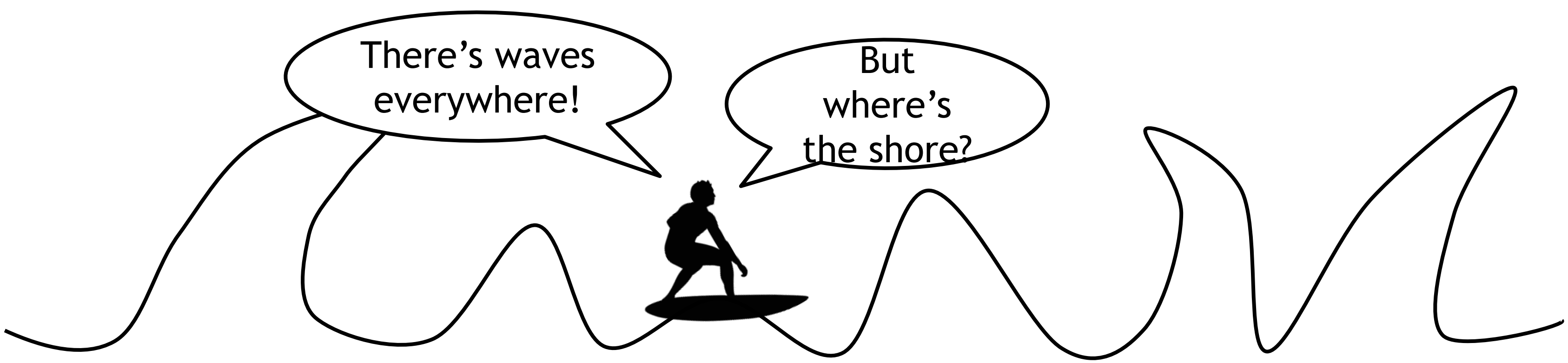




Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR 2018

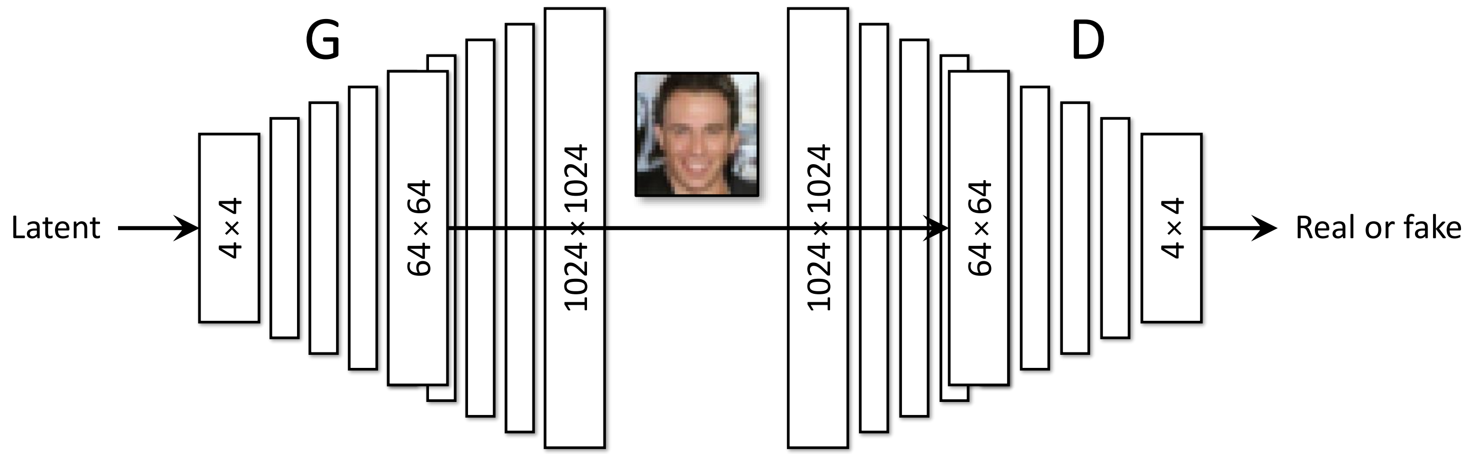
Progressive generation

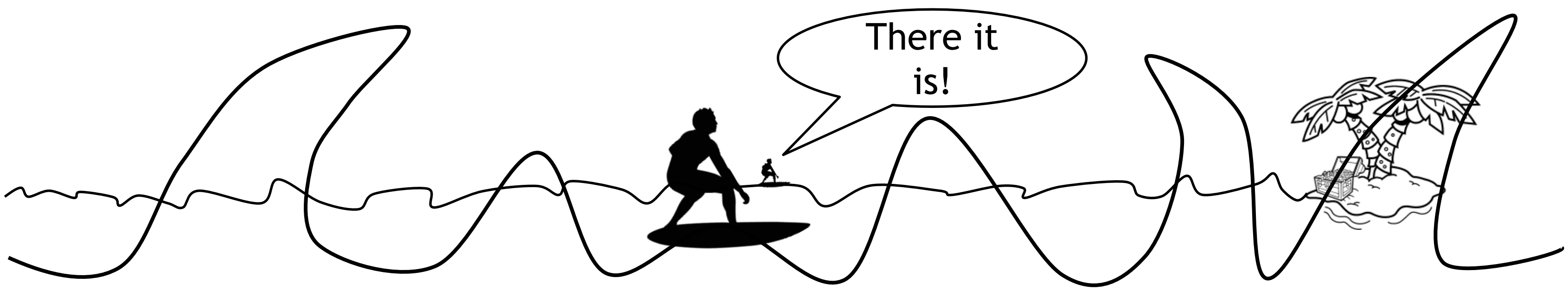




Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR 2018

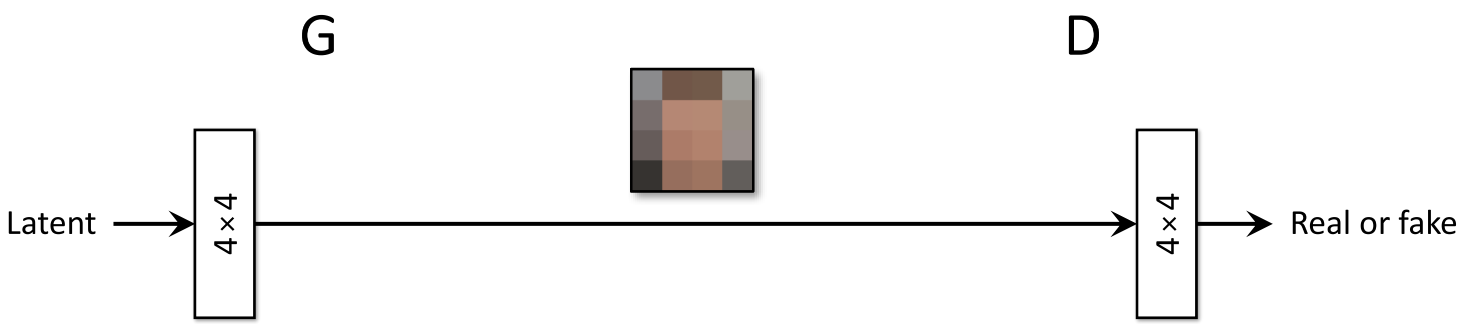
Progressive generation

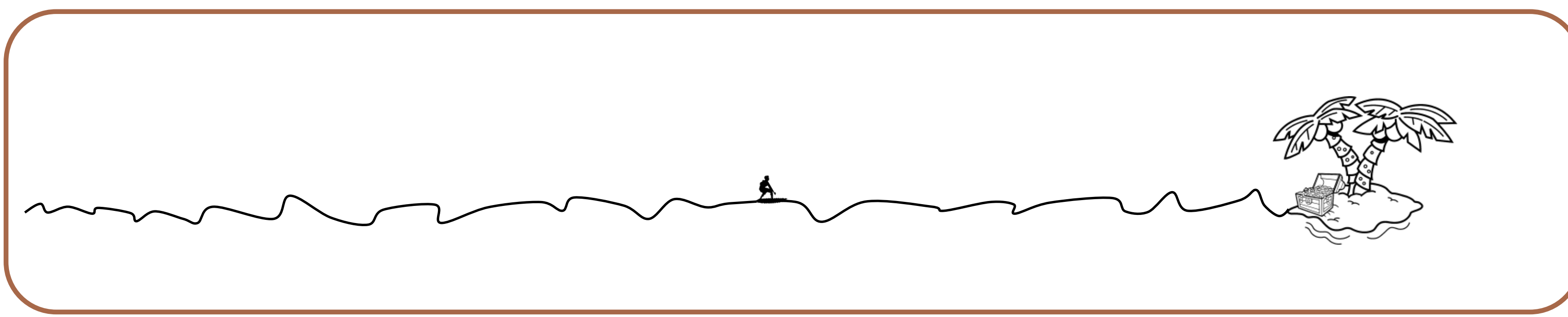




Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR 2018

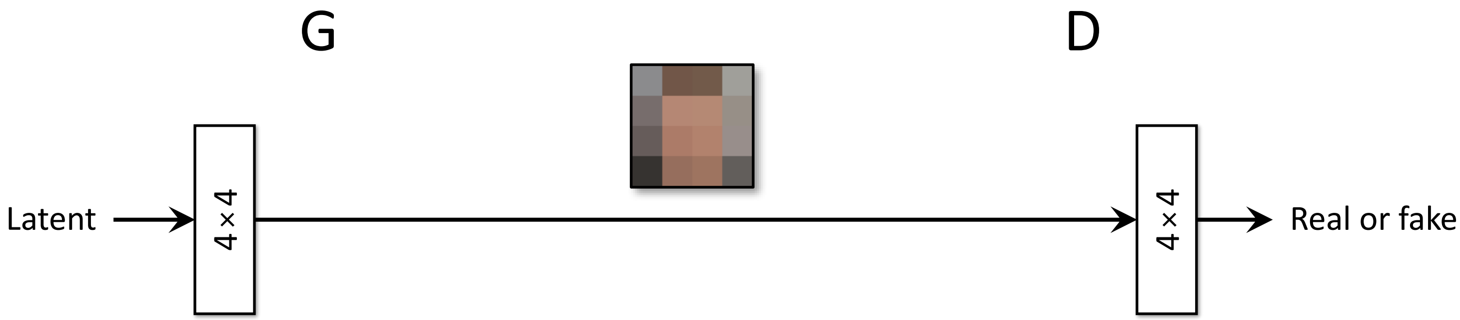
Progressive generation

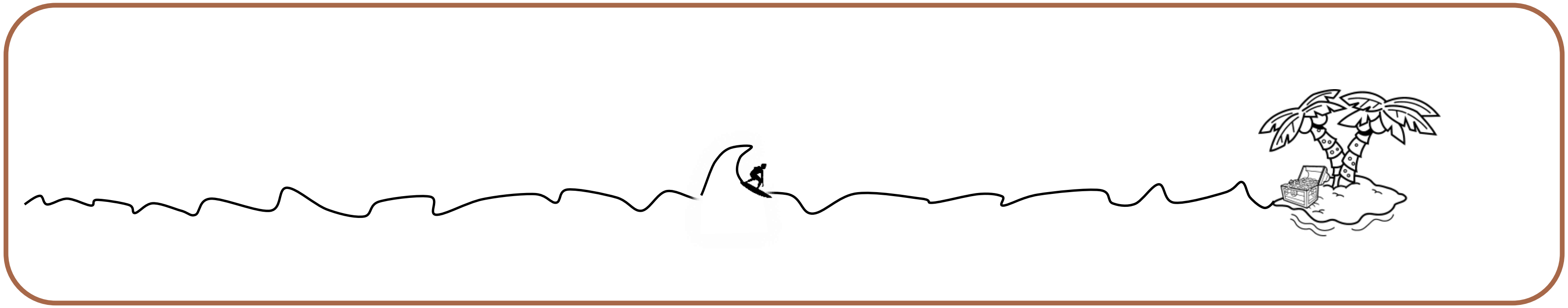




Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

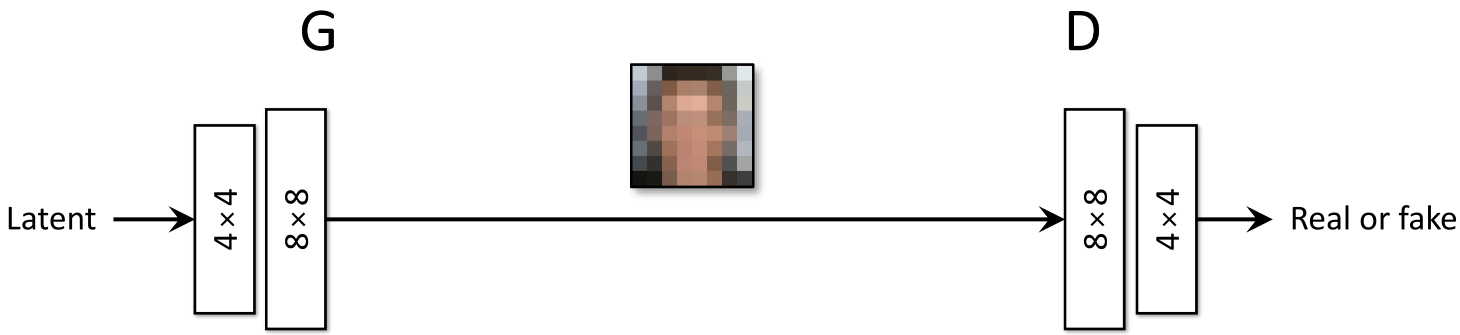
Progressive generation

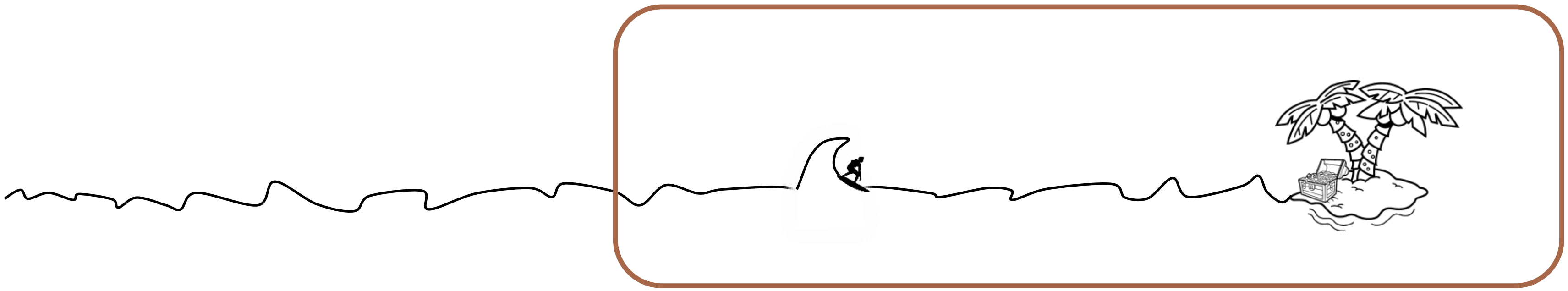




Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR 2018

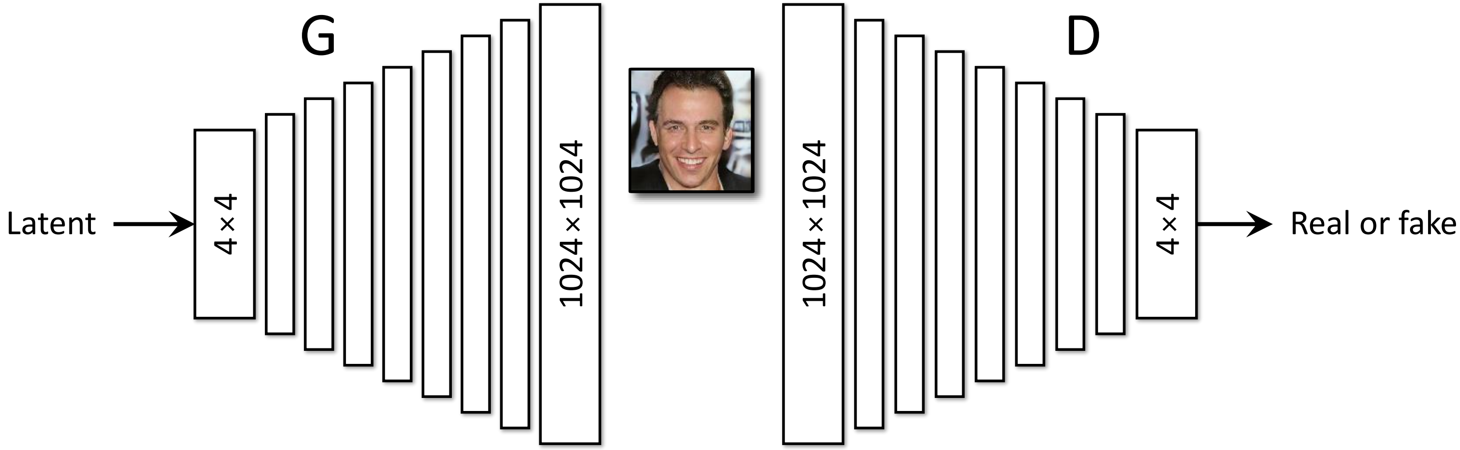
Progressive generation

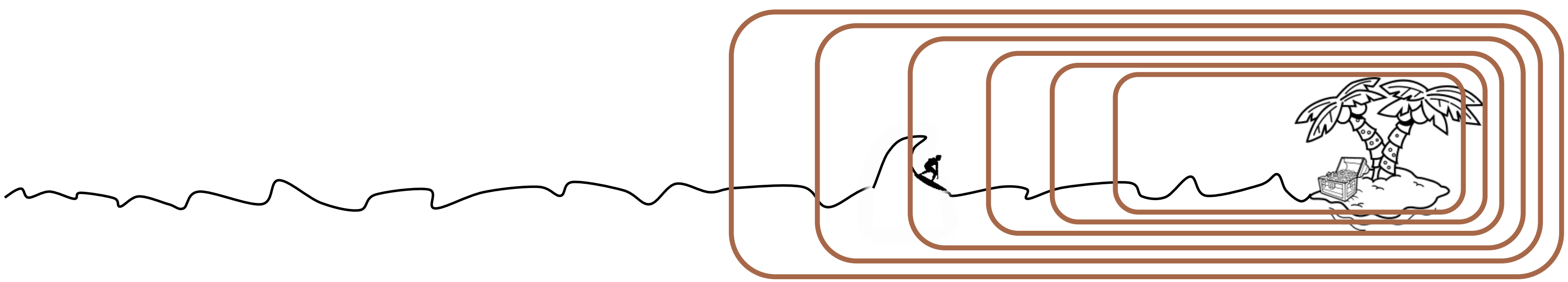




Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

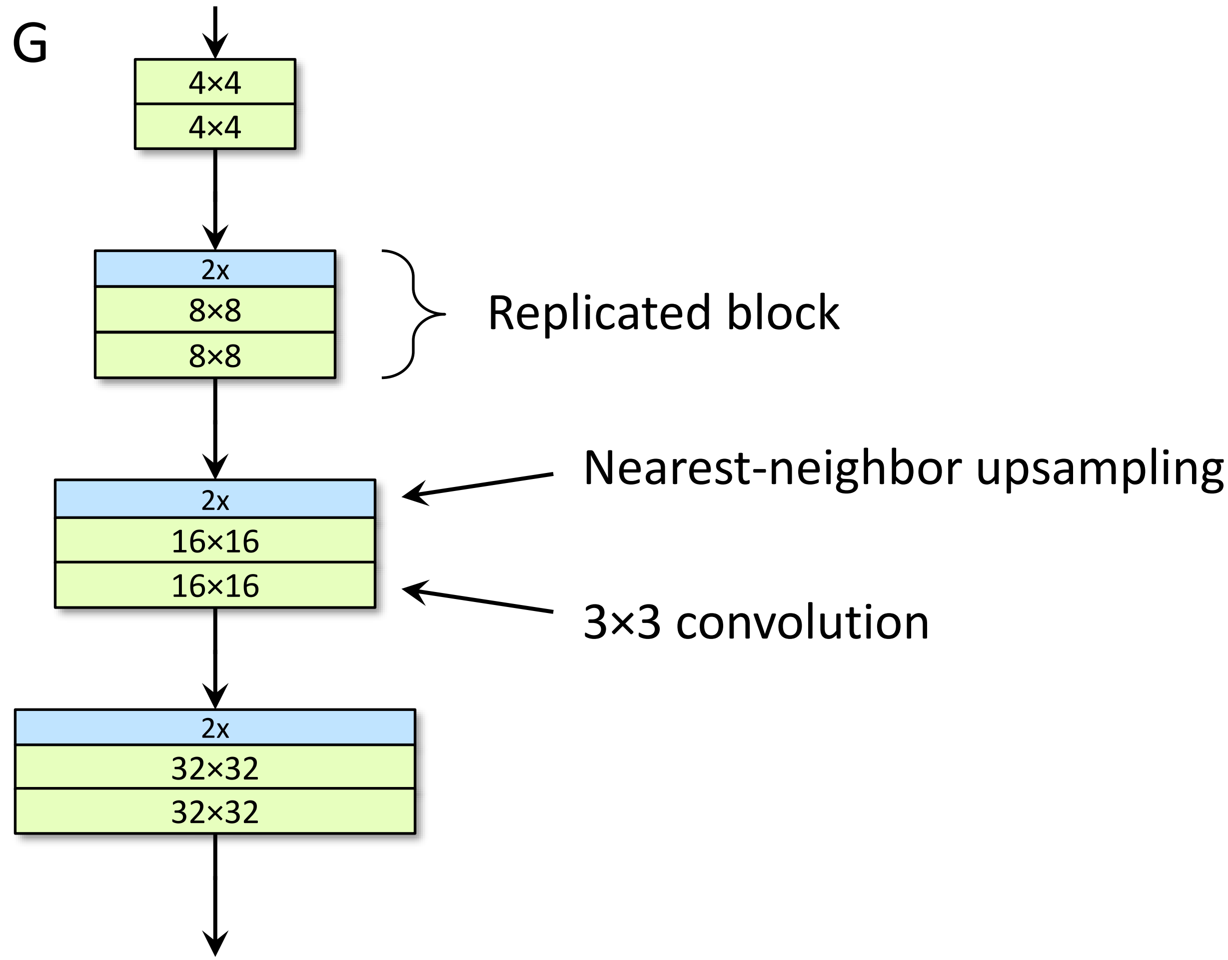
Progressive generation





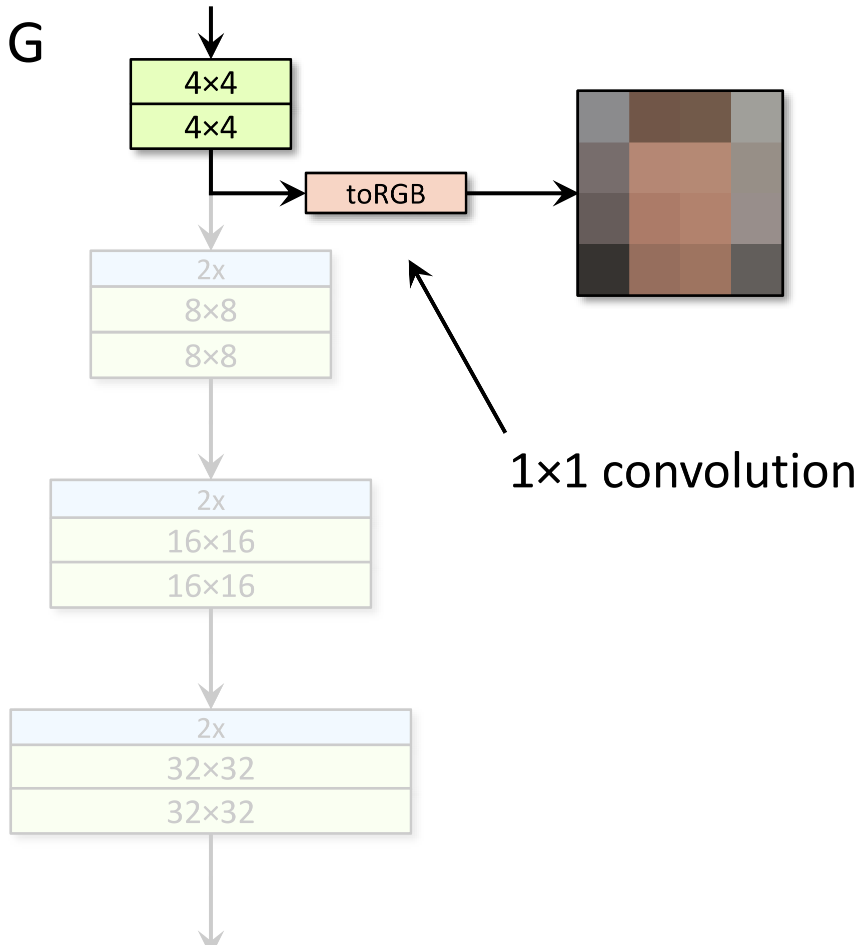
Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

Progressive generation



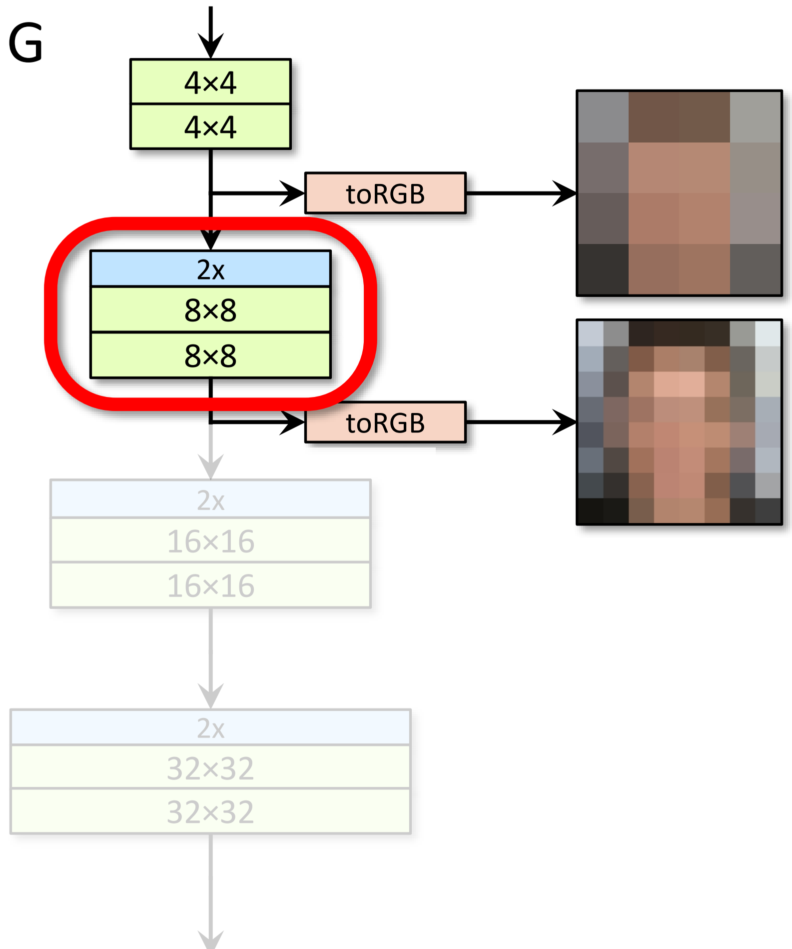
Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

Progressive generation



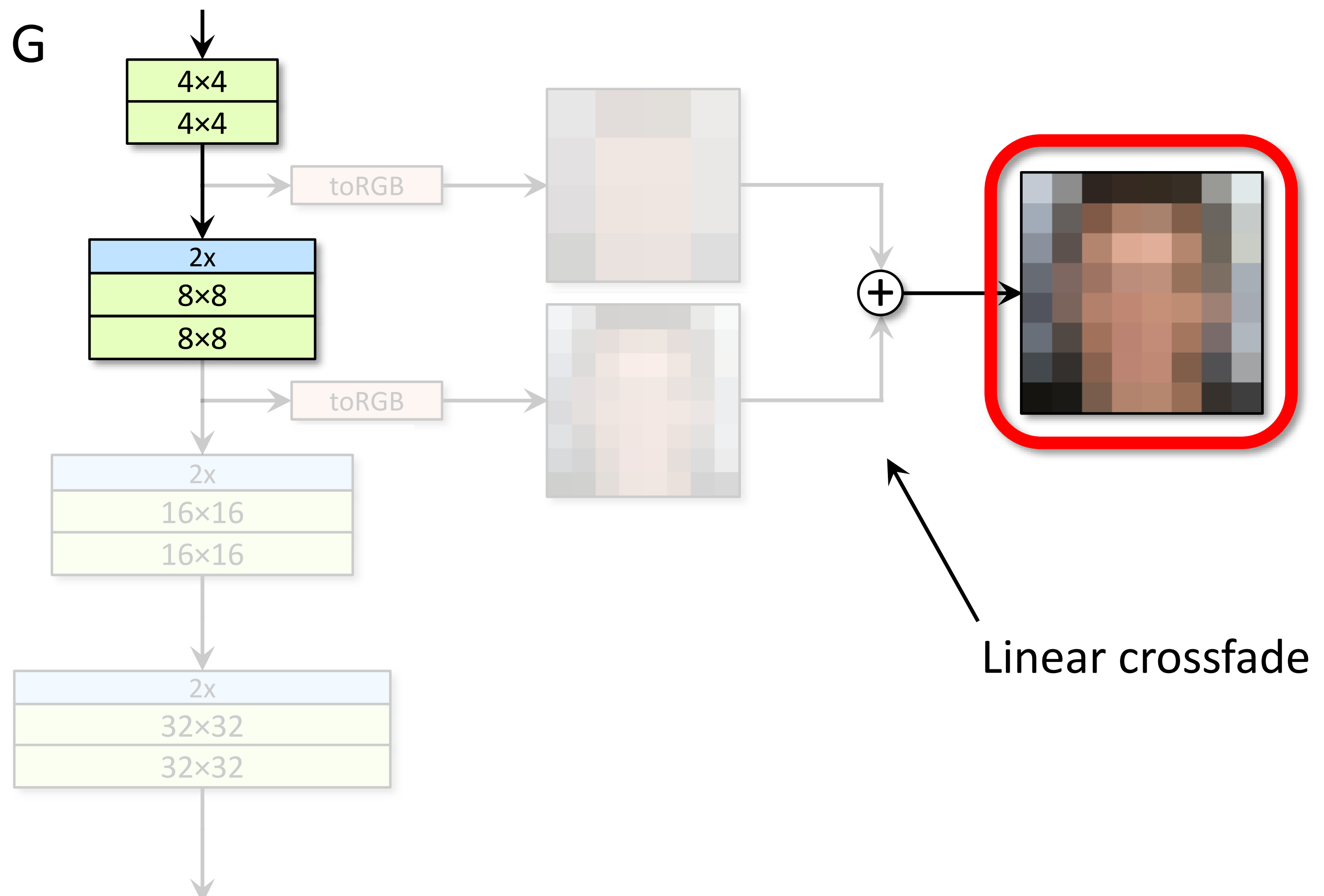
Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR 2018

Progressive generation



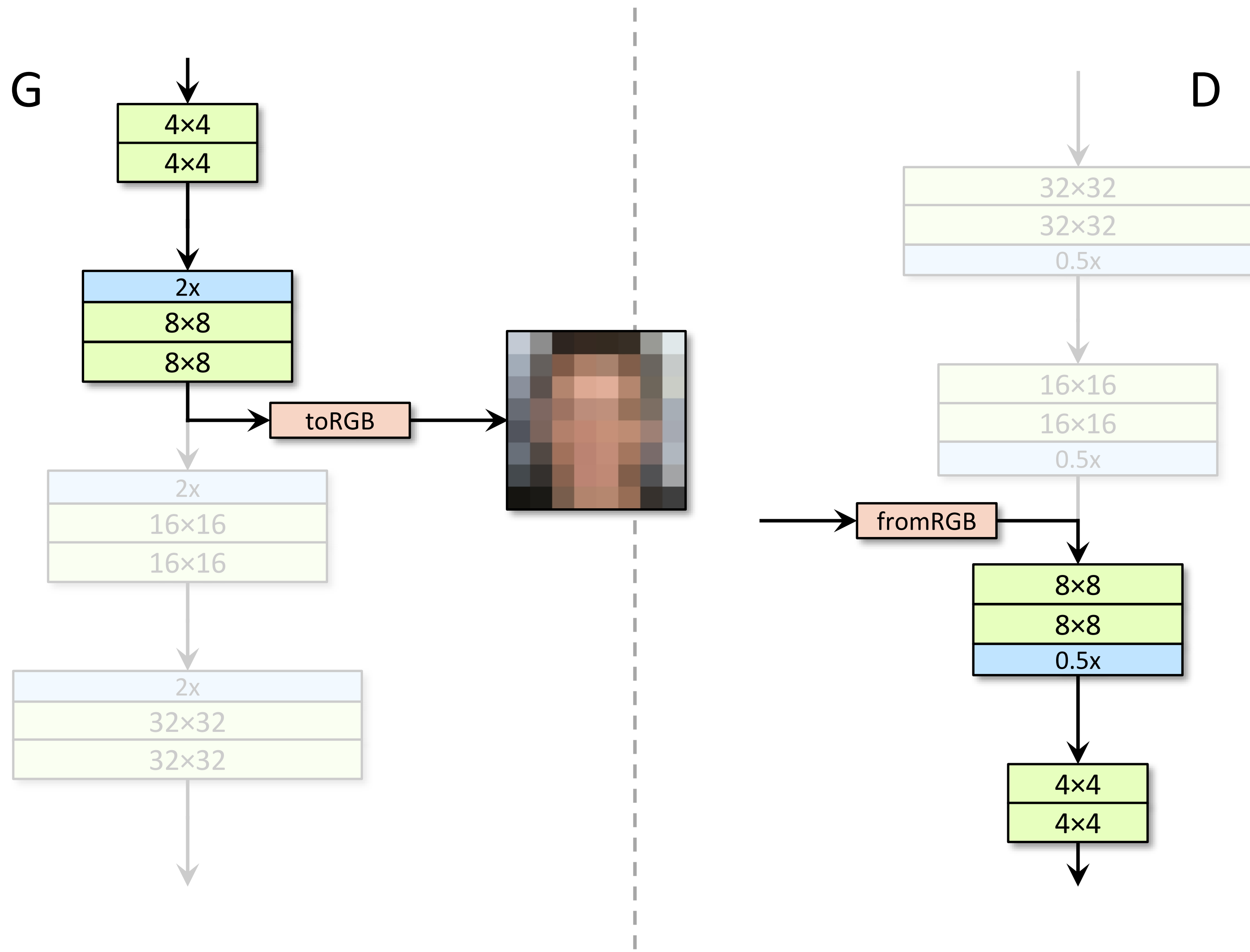
Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

Progressive generation



Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR 2018

Progressive generation



Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

Part V: Ethics (Politics, Privacy, Bias)

- Deep fakes
- Privacy
- Security and adversarial perturbations
- Bias
- AI for the people

“Deepfakes”

https://www.technologyreview.com/s/611726/the-defense-department-has-produced-the-first-tools-for-catching-deepfakes/
https://www.niemanlab.org/2018/11/how-the-wall-street-journal-is-preparing-its-journalists-to-detect-deepfakes/

You can be anyone you want…

Karras et al., “A Style-Based Generator Architecture for Generative Adversarial Networks”, CVPR 2019, https://arxiv.org/pdf/1812.04948.pdf

Detection methods

FaceForensics++: Learning to Detect Manipulated Facial Images

Andreas Rössler[1] Davide Cozzolino[2] Luisa Verdoliva[2] Christian Riess[3]
Justus Thies[1] Matthias Nießner[1]
[1]Technical University of Munich [2]University Federico II of Naples [3]University of Erlangen-Nuremberg

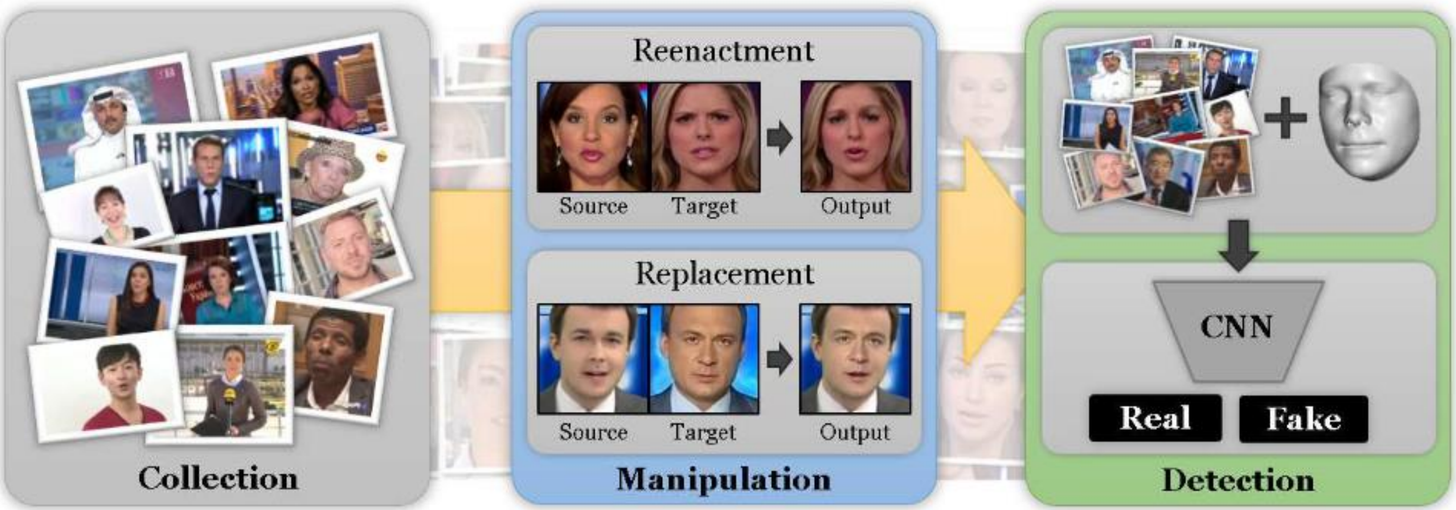


Figure 1: *FaceForensics++* is a dataset of facial forgeries that enables researchers to train deep-learning-based approaches in a supervised fashion. The dataset contains manipulations created with four state-of-the-art methods, namely, *Face2Face*, *FaceSwap*, *DeepFakes*, and *NeuralTextures*.

Rossler et al.. "FaceForensics++: Learning to Detect Manipulated Facial Images", ICCV 2019

Detection methods

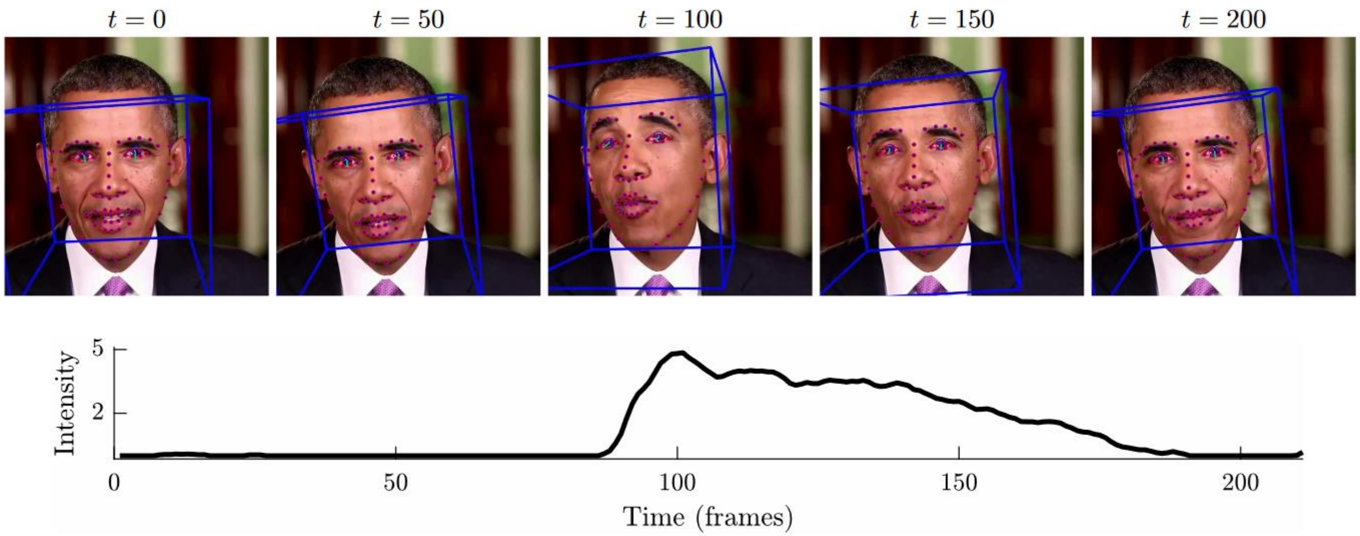
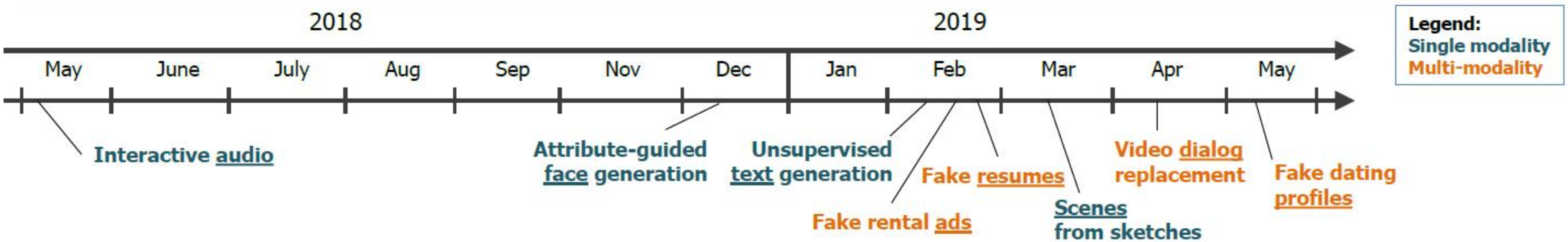


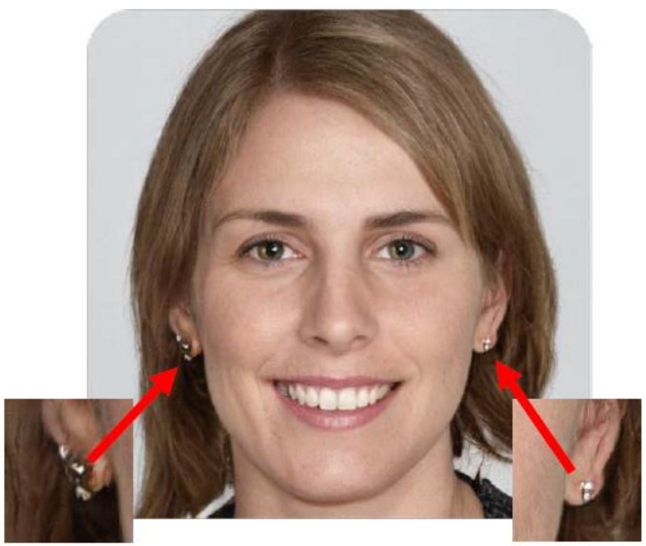
Figure 1. Shown above are five equally spaced frames from a 250-frame clip annotated with the results of OpenFace tracking. Shown below is the intensity of one action unit AU01 (eye brow lift) measured over this video clip.

“We describe a forensic technique that models facial expressions and movements that typify an individual’s speaking pattern. Although not visually apparent, these correlations are often violated by the nature of how deep-fake videos are created and can, therefore, be used for authentication.

Agarwal et al., “Protecting World Leaders Against Deep Fakes”, CVPR Workshops, 2019

Incredible Pace of Synthetic Media Generation





ENTIRE GUEST SUITE

Luxury Condo 3 Bed + 3 Bath
Port Melbourne

- 8 guests
- 3 bedrooms
- 4beds
- 2 baths

Bathroom (with seating for 2 more people), basin and eclectic French garden and kitchen. 24/7 carpeted charc. Laundrymemberly : More balcony – Garden – Metro, Liverpool Street (15 min walk) Walking distance to Wyckofferdon

State of the Art Detection is Statistically Based, Narrow, or Both

Audio: ASVspoof

Hand-crafted Features

Neural Networks

Temporal Neural Networks

Fusion

(Lavrentyeva et al. 2017)

Text: GLTR

Input text

Word Prediction Probability

NY Times: Here's a pop quiz for you

AI: I've been a gamer for over ten years.

High — Low

Word Predictability

AI methods choose more predictable next-words than humans, statistically

(MIT-IBM Watson AI lab, HarvardNLP 2019)

Image/Video: DARPA MediFor

Noise Fingerprint Network

Manipulation detection heatmap

(MediFor: USC/ISI, Univ. Naples 2019)

Matt Turek

Expected Threats

Targeted Personal Attacks

Peele 2017

AI Multimedia Algorithms

Highly realistic video

Generated Events at Scale

AI Multimedia Algorithms

On a rainy spring day, a vast, violent group gathered in front of the US Capitol to protest recent cuts in Social Security.

Text

Video & Audio

Image

1000s +

Believable fake events

Ransomfake concept: Identity Attacks as a service (IAaaS)

Bricman 2019

AI Multimedia Algorithms

Forged Evidence

Identity Attacks

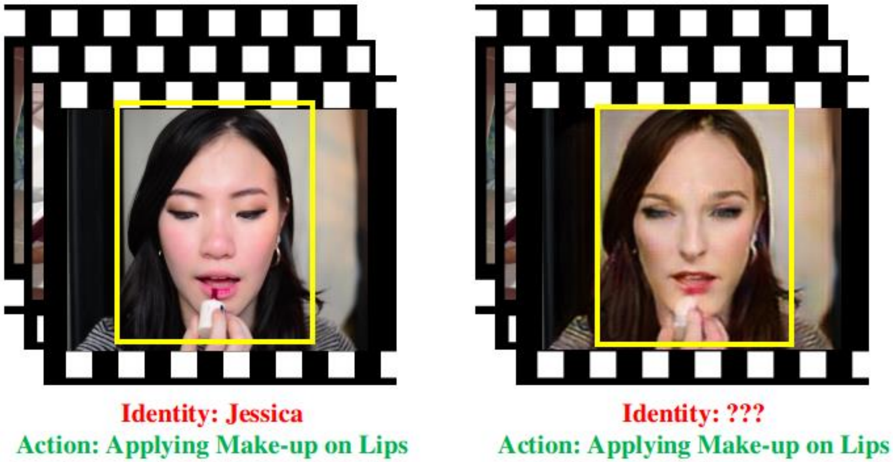
Examples of possible fakes:

- Substance abuse
- Foreign contacts
- Compromising events
- Social media postings
- Financial inconsistencies
- Forging identity

Undermines key individuals and organizations

Matt Turek

GANs for Privacy (Action Detection)

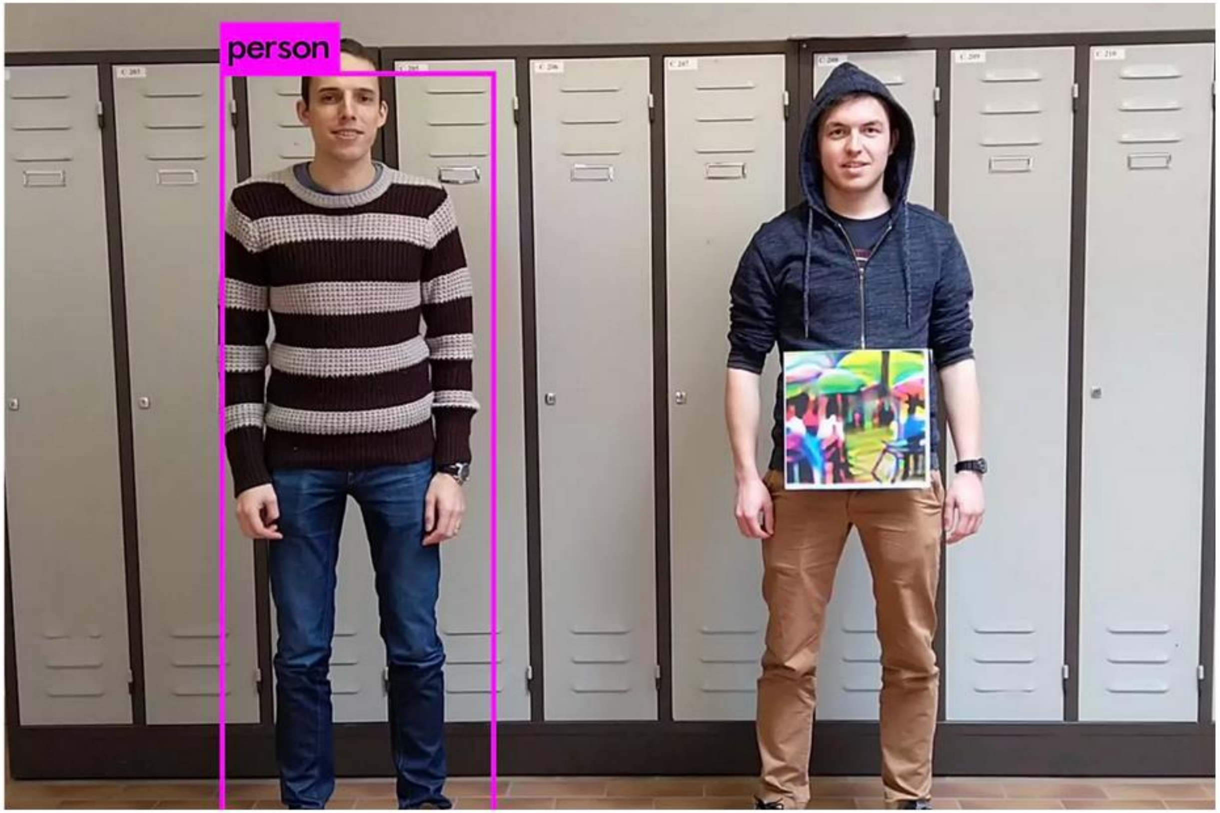


Ren et al., “Learning to Anonymize Faces for Privacy Preserving Action Detection”, ECCV 2018

Adversarial Attacks

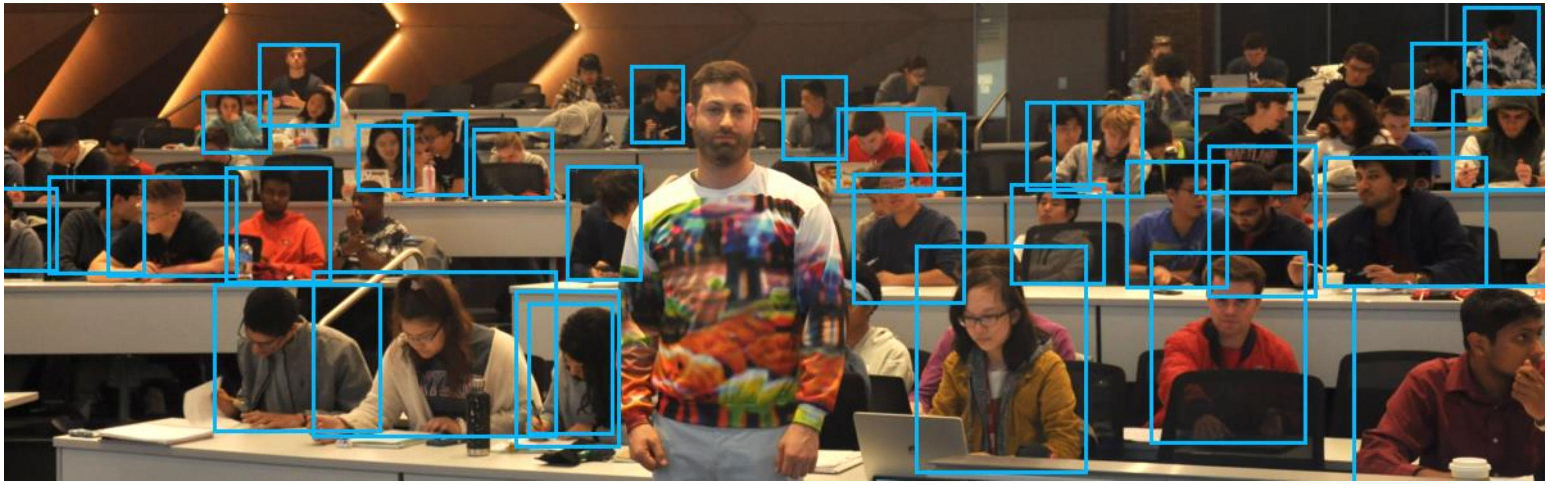
https://bair.berkeley.edu/blog/2017/12/30/yolo-attack/

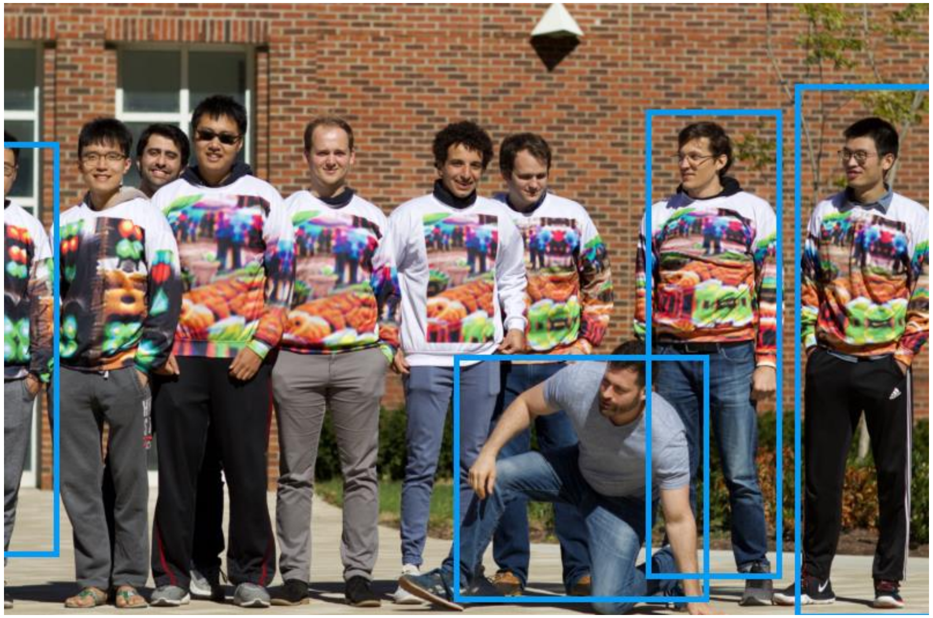
Adversarial Attacks



https://www.theverge.com/2019/4/23/18512472/fool-ai-surveillance-adversarial-example-yolov2-person-detection

Adversarial Attacks





Tom Goldstein https://www.cs.umd.edu/~tomg/projects/invisible/

Adversarial Attacks

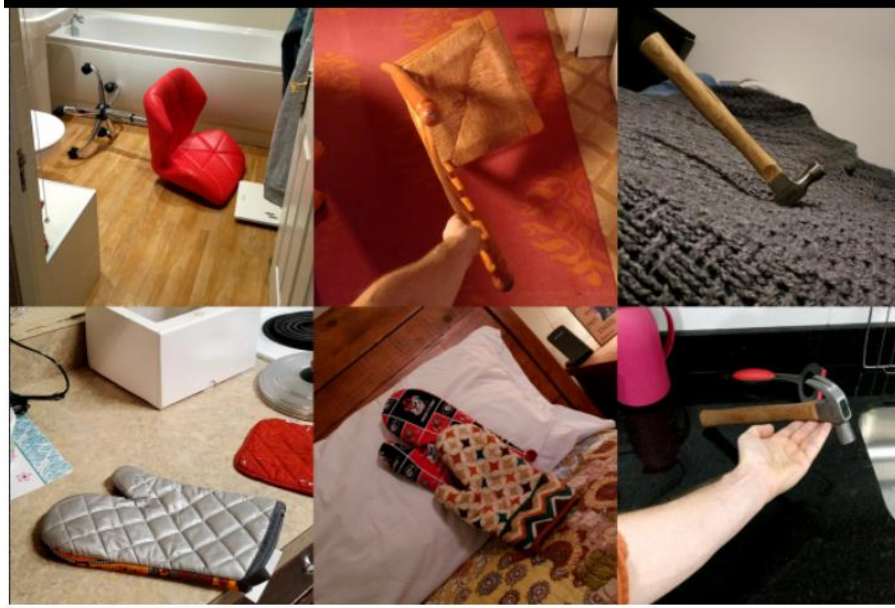
Massachusetts Institute of Technology

NEWS | VIDEO | SOCIAL | FOLLOW MIT

MIT News

ON CAMPUS AND AROUND THE WORLD

Browse or Search



ObjectNet, a dataset of photos created by MIT and IBM researchers, shows objects from odd angles, in multiple orientations, and against varied backgrounds to better represent the complexity of 3D objects. The researchers hope the dataset will lead to new computer vision techniques that perform better in real life.

Photo collage courtesy of the researchers.

This object-recognition dataset stumped the world's best computer vision models

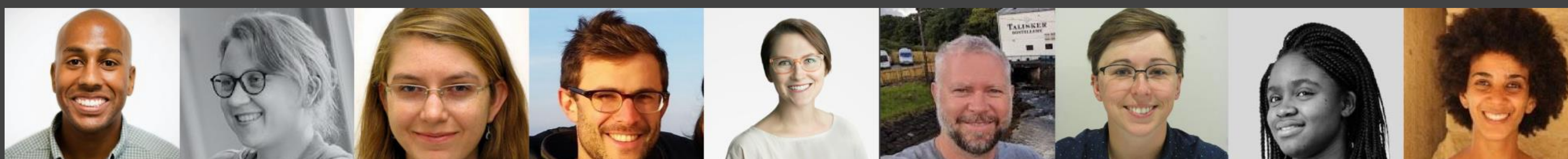
Objects are posed in varied positions and shot at odd angles to spur new AI techniques.

http://news.mit.edu/2019/object-recognition-dataset-stumped-worlds-best-computer-vision-models-1210

Conversation AI

Bias in the Vision and Language of Artificial Intelligence

Margaret Mitchell
Senior Research Scientist
Google AI



Andrew Zaldivar | Me | Simone Wu | Parker Barnes | Lucy Vasserman | Ben Hutchinson | Elena Spitzer | Deb Raji | Timnit Gebru

Adrian Benton | Brian Zhang | Dirk Hovy | Josh Lovejoy | Alex Beutel | Blake Lemoine | Hee Jung Ryu | Hartwig Adam | Blaise Agüera y Arcas

What do you see?

- Bananas
- Stickers
- Dole Bananas
- Bananas at a store
- Bananas on shelves
- Bunches of bananas
- Bananas with stickers on them
- Bunches of bananas with stickers on them on shelves in a store

...We don't tend to say

Yellow Bananas

What do you see?

Green Bananas

Unripe Bananas

What do you see?

Ripe Bananas

Bananas with spots

Bananas good for banana bread

What do you see?

Yellow Bananas?

Yellow **is prototypical for bananas**

Prototype Theory

One purpose of categorization is to **reduce the infinite differences** among stimuli **to** behaviourally and **cognitively usable proportions**

There may be some central, prototypical notions of items that arise from stored typical properties for an object category (Rosch, 1975)

May also store exemplars (Wu & Barsalou, 2009)

Fruit

Bananas
"Basic Level"

Unripe Bananas,
Cavendish Bananas

A man and his son are in a terrible accident and are rushed to the hospital in critical care.

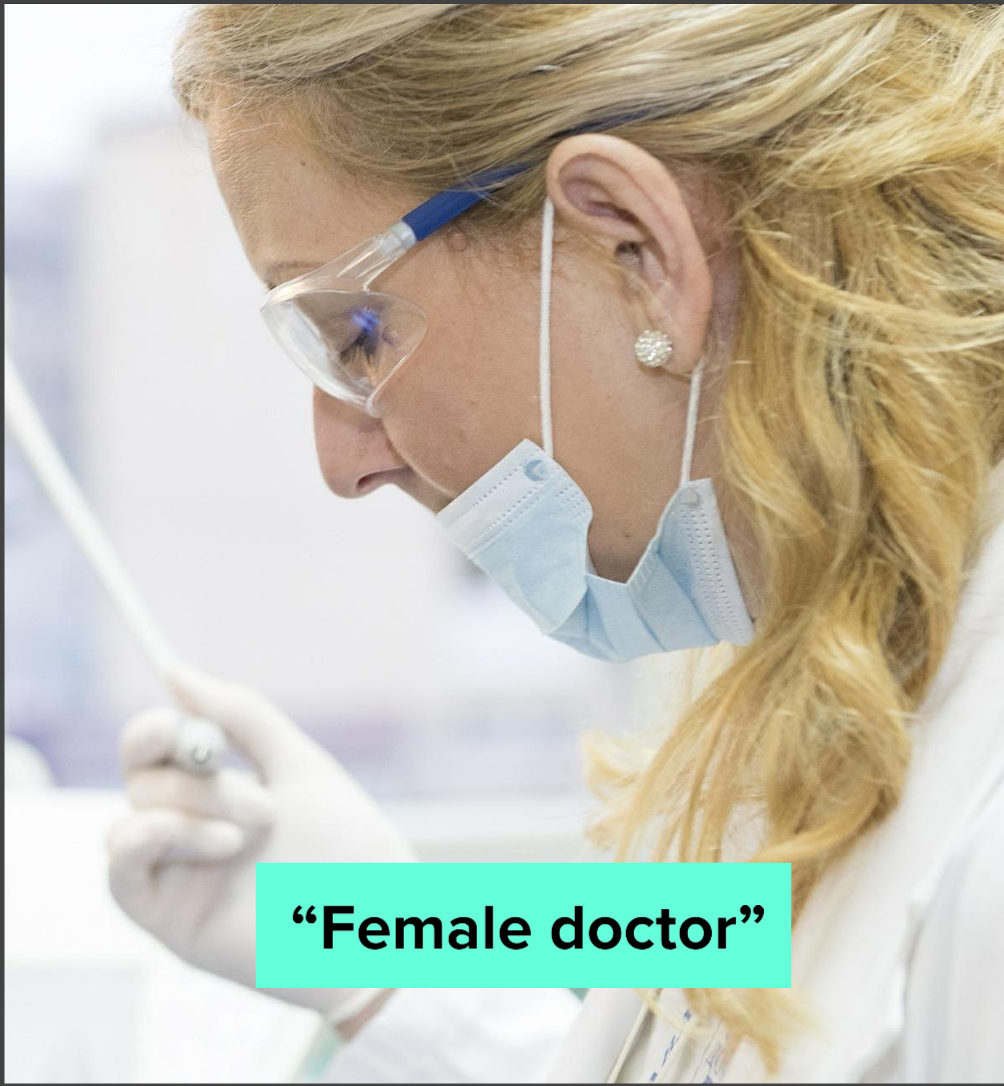
The doctor looks at the boy and exclaims "I can't operate on this boy, he's my son!"

How could this be?

A man and his son are in a terrible accident and are rushed to the hospital in critical care.

The doctor looks at the boy and exclaims "I can't operate on this boy, he's my son!"

How could this be?




“Doctor”


“Female doctor”

The majority of test subjects overlooked the possibility that the doctor is a she - including men, women, and self-described feminists.

Wapman & Belle, Boston University

Human Reporting Bias

The **frequency** with which **people write** about actions, outcomes, or properties is **not a reflection of real-world frequencies** or the degree to which a property is characteristic of a class of individuals

Bias in Language

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

Figure 1: The most extreme occupations as projected on to the *she−he* gender direction on g2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded.

Gender stereotype *she-he* analogies.

sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

Gender appropriate *she-he* analogies.

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Figure 2: **Analogy examples**. Examples of automatically generated analogies for the pair *she-he* using the procedure described in text. For example, the first analogy is interpreted as *she*:*sewing* :: *he*:*carpentry* in the original w2vNEWS embedding. Each automatically generated analogy is evaluated by 10 crowd-workers are to whether or not it reflects gender stereotype. Top: illustrative gender stereotypic analogies automatically generated from w2vNEWS, as rated by at least 5 of the 10 crowd-workers. Bottom: illustrative generated gender-appropriate analogies.

Bolukbasi et al., “Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings”, NIPS 2016

Bias in Language

he (158)

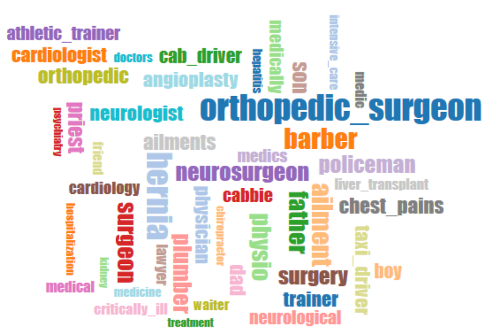
she (42)

Adjectives

Or type your own words...

doctor

he (47)



she (153)

http://wordbias.umiacs.umd.edu/

Bias in Vision

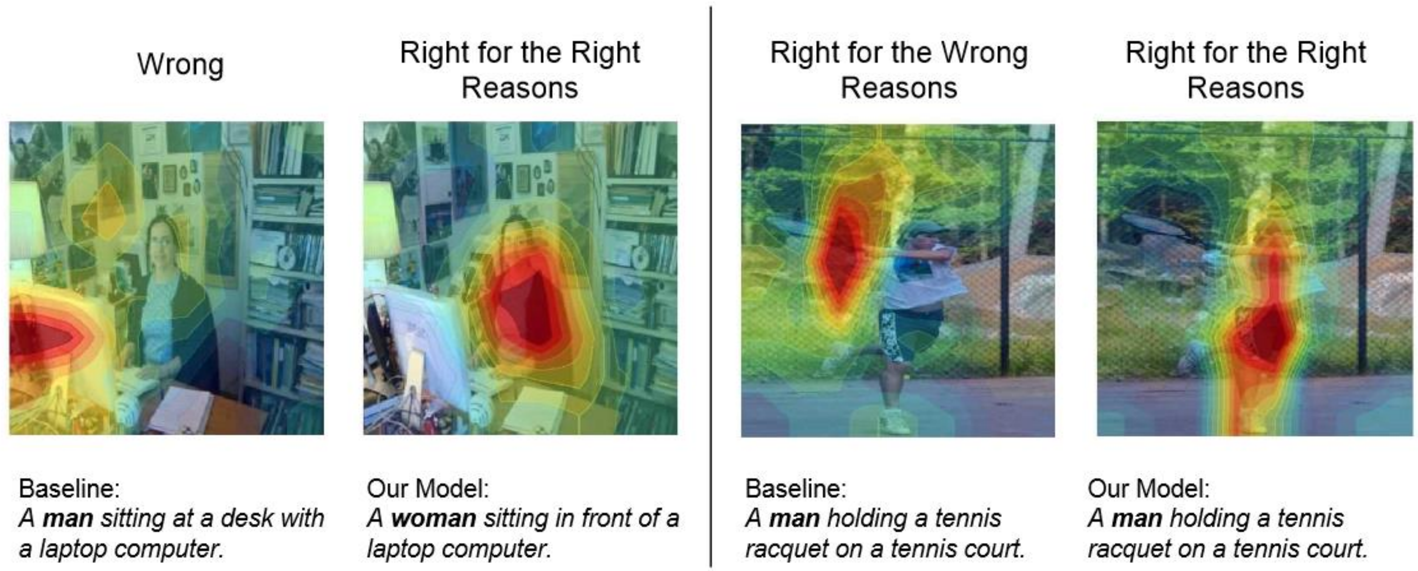


Fig. 1: Examples where our proposed model (Equalizer) corrects bias in image captions. The overlaid heatmap indicates which image regions are most important for predicting the gender word. On the left, the baseline predicts gender incorrectly, presumably because it looks at the laptop (not the person). On the right, the baseline predicts the gender correctly but it does not look at the person when predicting gender and is thus not acceptable. In contrast, our model predicts the correct gender word and correctly considers the person when predicting gender.

Burns et al., "Women also Snowboard: Overcoming Bias in Captioning Models", ECCV 2018

Bias in Vision

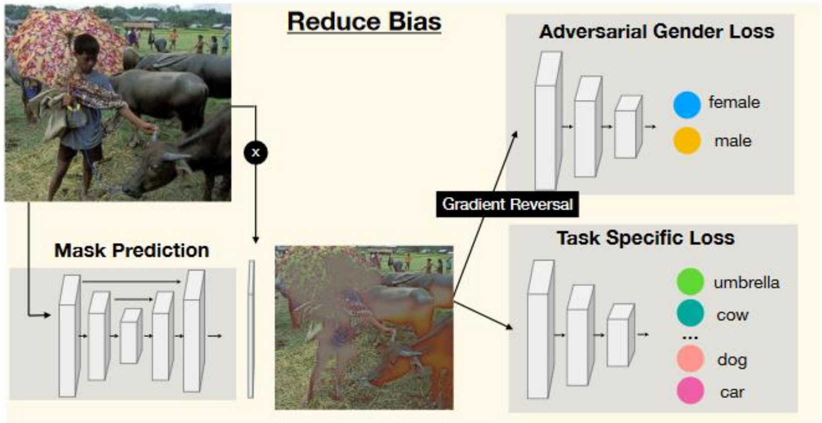


Figure 2. In our bias mitigation approach, we learn a task-specific model with an adversarial loss that removes features corresponding to a protected variable from an intermediate representation in the model – here we illustrate our pipeline to visualize the removal of features in image space through an auto-encoder network.

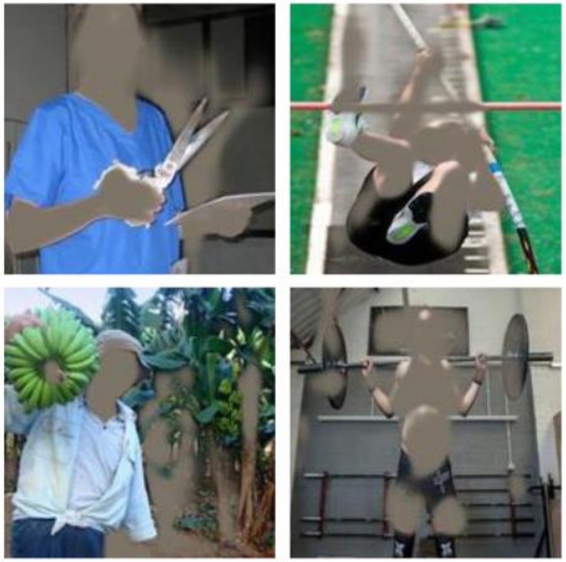
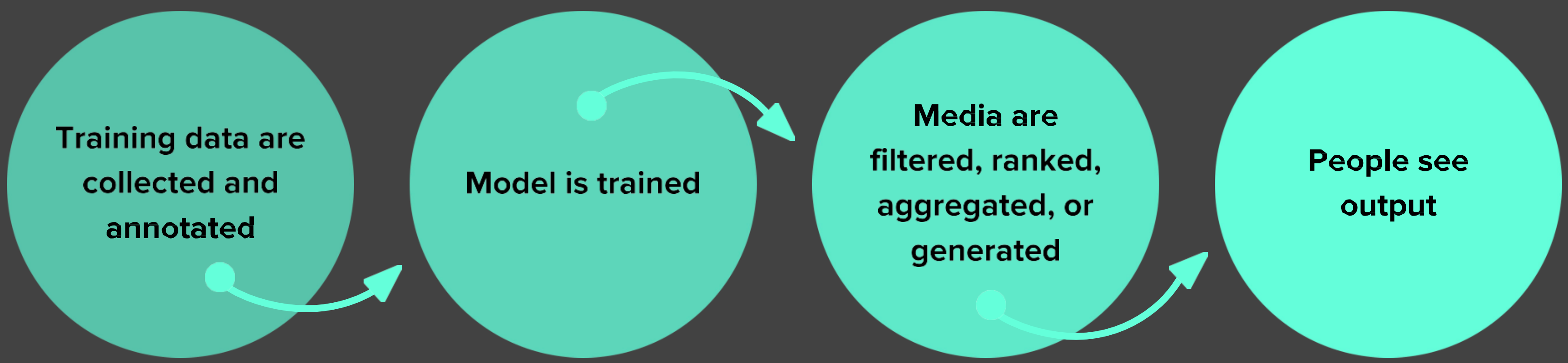


Figure 3. Images after adversarial removal of gender when applied to the image space. The objective was to preserve information about objects and verbs, e.g. scissors, banana (COCO) or vaulting, lifting (imSitu) while removing gender correlated features.

Wang et al., "Balanced Datasets Are Not Enough: Estimating and Mitigating Gender Bias in Deep Image Representations", ICCV 2019


Training data are collected and annotated
Model is trained
Media are filtered, ranked, aggregated, or generated
People see output

Biases in Data

Margaret Mitchell

Biases in Data

Selection Bias: Selection does not reflect a random sample

CREDIT

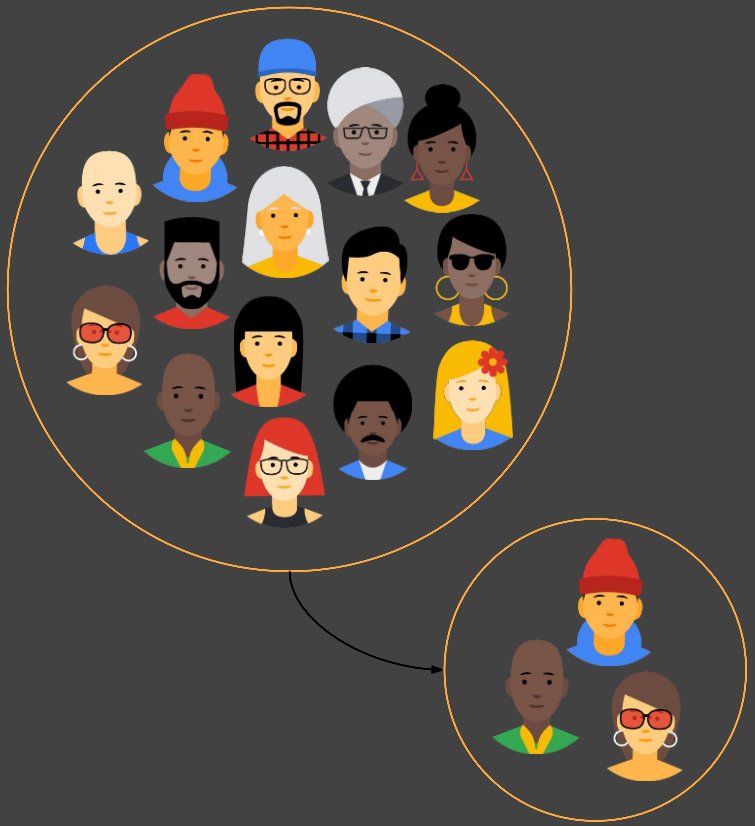
Biases in Data

Out-group homogeneity bias: Tendency to see outgroup members as more alike than ingroup members

Margaret Mitchell

Biases in Data →
Biased Data Representation

It's possible that you have an appropriate amount of data for every group you can think of but that some groups are represented less positively than others.



Biases in Data → Biased Labels

Annotations in your dataset will reflect the worldviews of your annotators.

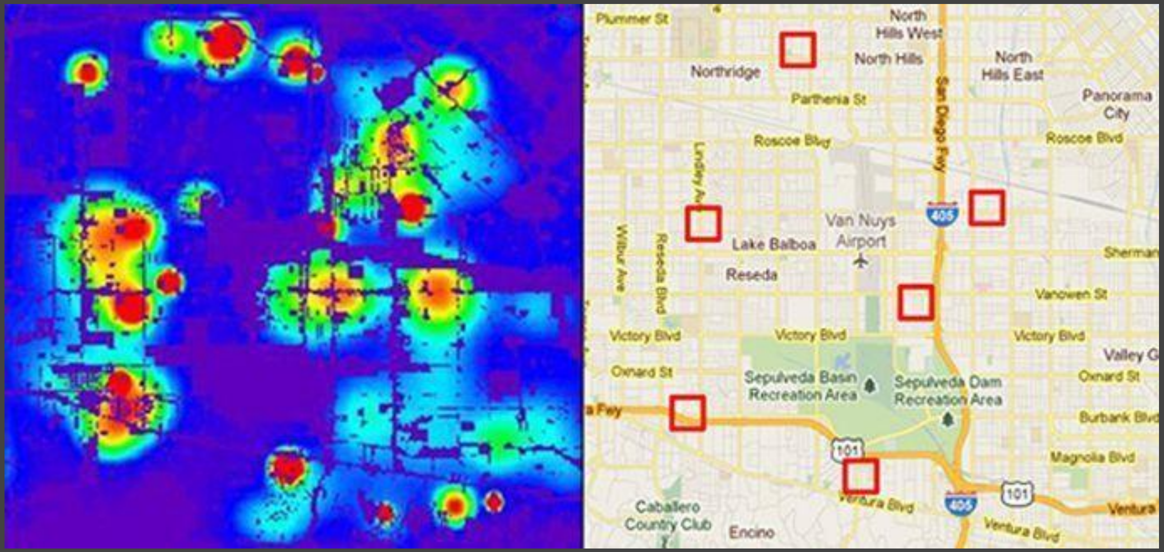
https://ai.googleblog.com/2018/09/introducing-inclusive-images-competition.html

Predicting Future Criminal Behavior

Margaret Mitchell

Predicting Policing

- Algorithms identify potential crime hot-spots
- Based on where crime is previously reported, not where it is known to have occurred
- Predicts future events from past



CREDIT

[Smithsonian. *Artificial Intelligence Is Now Used to Predict Crime. But Is It Biased?* 2018]

Predicting Sentencing

- Prater (who is white) rated **low risk** after shoplifting, despite two armed robberies; one attempted armed robbery.
- Borden (who is black) rated **high risk** after she and a friend took (but returned before police arrived) a bike and scooter sitting outside.
- Two years later, Borden has not been charged with any new crimes. Prater serving 8-year prison term for grand theft.

CREDIT

ProPublica. Northpointe: Risk in Criminal Sentencing. 2016.

Predicting Criminality

Israeli startup, Faception

> *"Faception is first-to-technology and first-to-market with proprietary computer vision and machine learning technology for profiling people and* ***revealing their personality based only on their facial image****."*

Offering specialized engines for recognizing "High IQ", "White-Collar Offender", "Pedophile", and "Terrorist" from a face image.

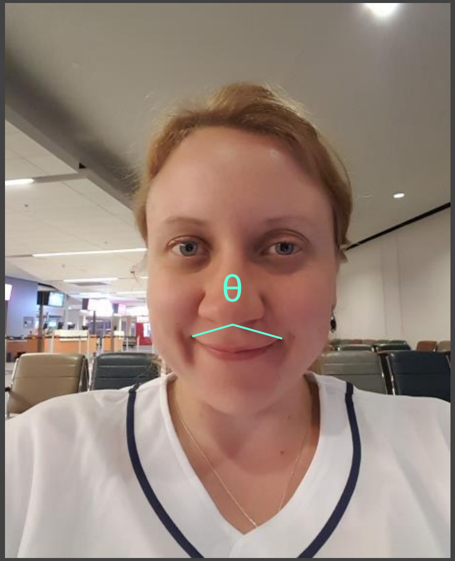
Main clients are in homeland security and public safety.

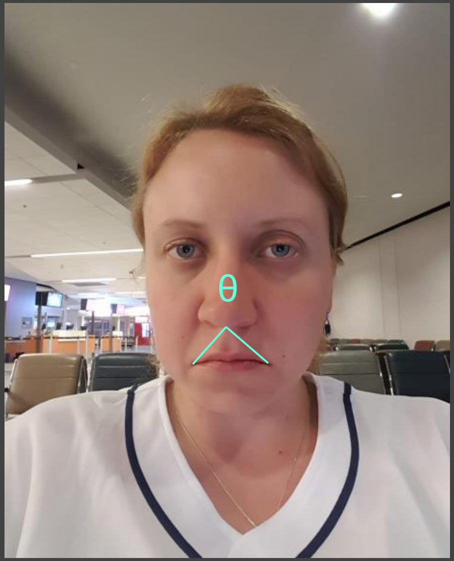
Predicting Criminality

“Automated Inference on Criminality using Face Images” Wu and Zhang, 2016. arXiv

1,856 closely cropped images of faces; Includes “wanted suspect” ID pictures from specific regions.

“[…] angle θ from nose tip to two mouth corners is on average 19.6% smaller for criminals than for non-criminals …”

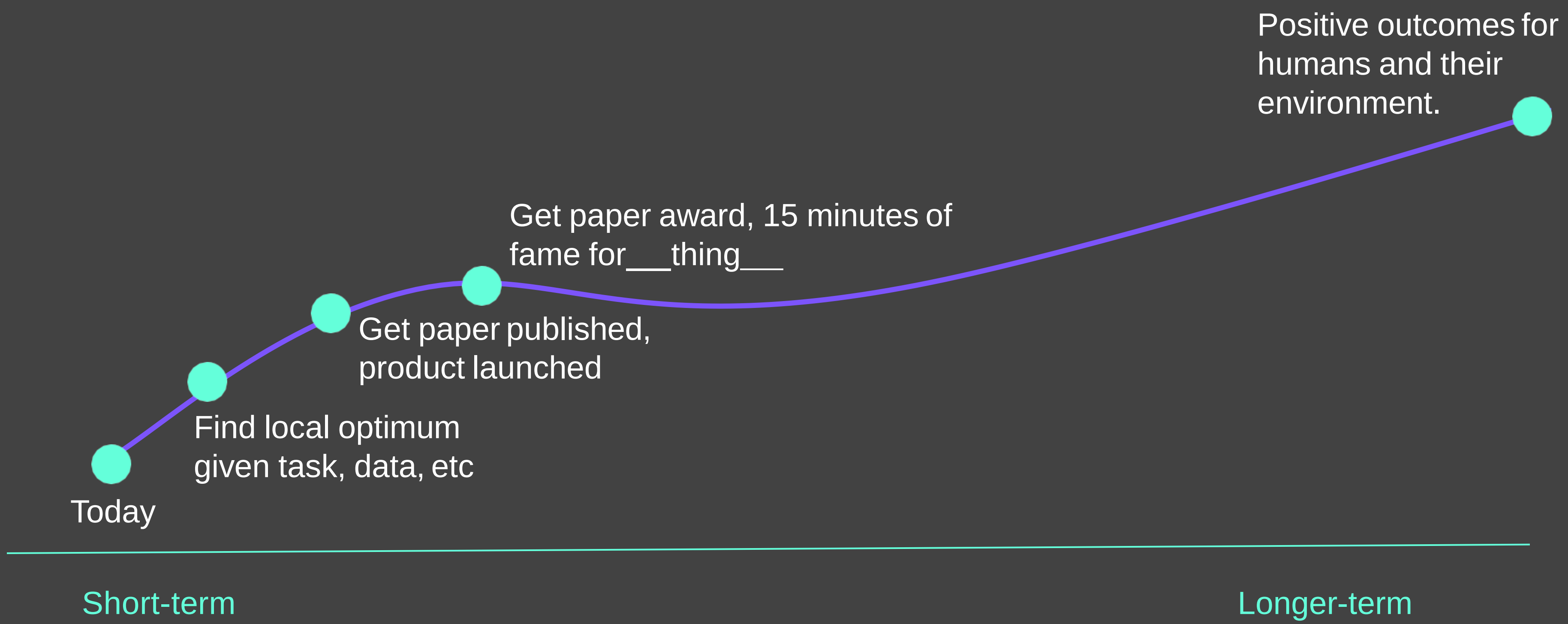




See our longer piece on Medium, “Physiognomy’s New Clothes”

It’s up to **us** to influence how AI evolves.

Margaret Mitchell


Positive outcomes for humans and their environment.
Get paper award, 15 minutes of fame for___thing___
Get paper published, product launched
Find local optimum given task, data, etc
Today
Short-term
Longer-term

H

The development of AI should be guided by a concern for its impact on **human** society.

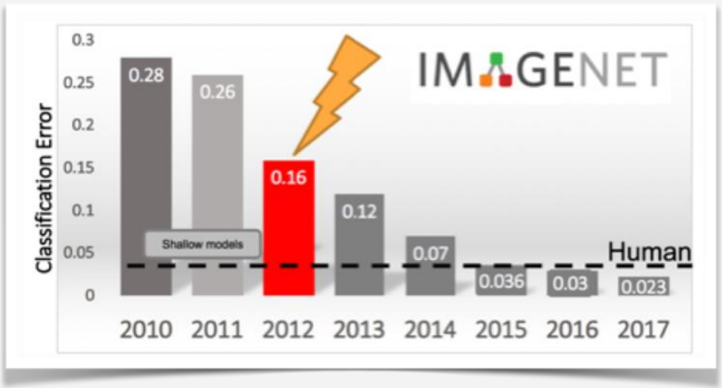
A

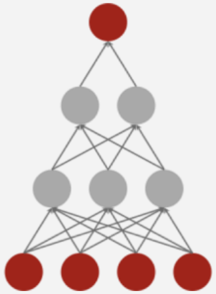
AI should **augment** human skills, not replace them.

I

AI must incorporate more of the versatility, nuance, and depth of the human **intellect.**

From academic backwater to center of attention in 5 years





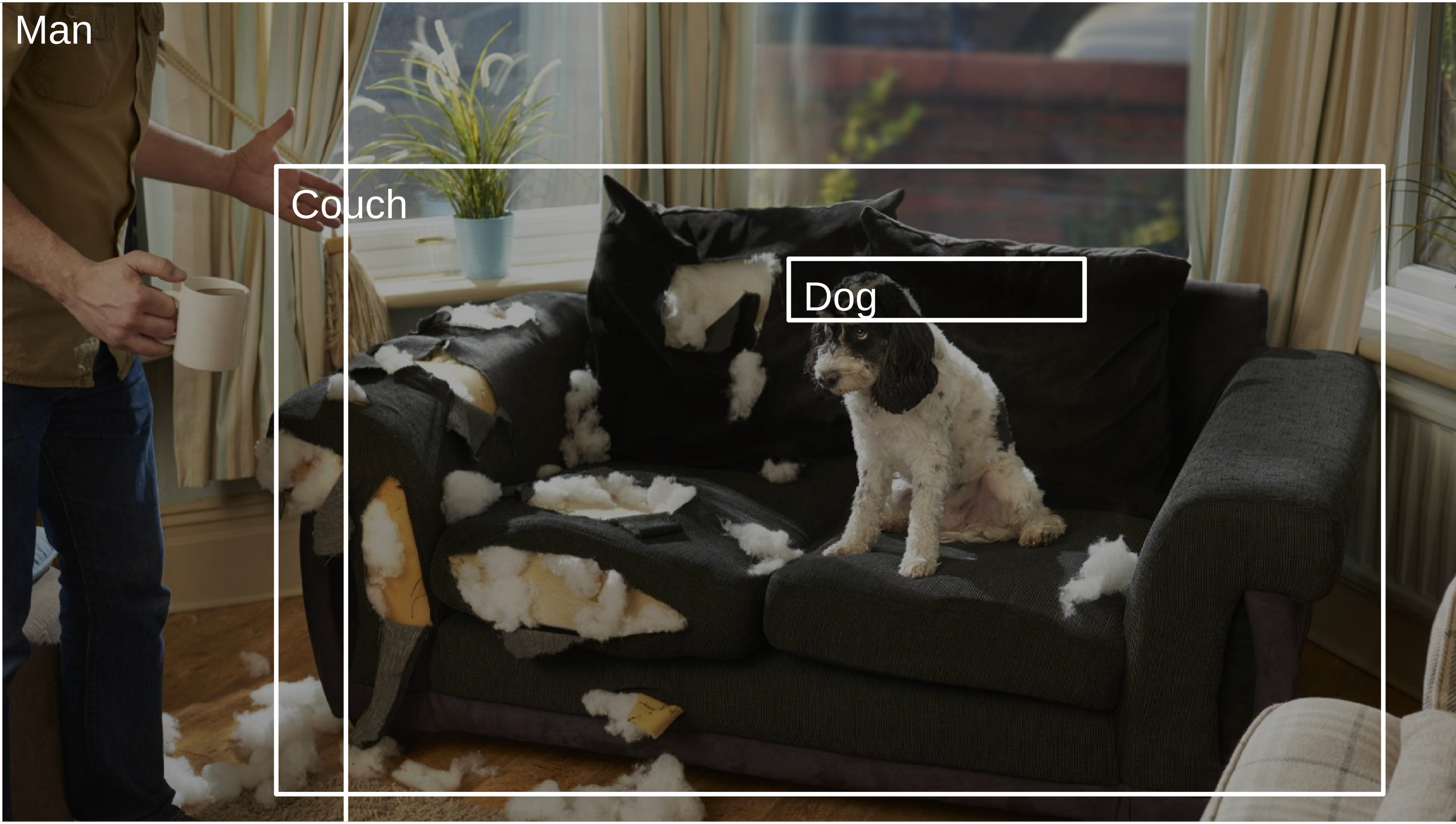
The Deep Learning Revolution

What happened?

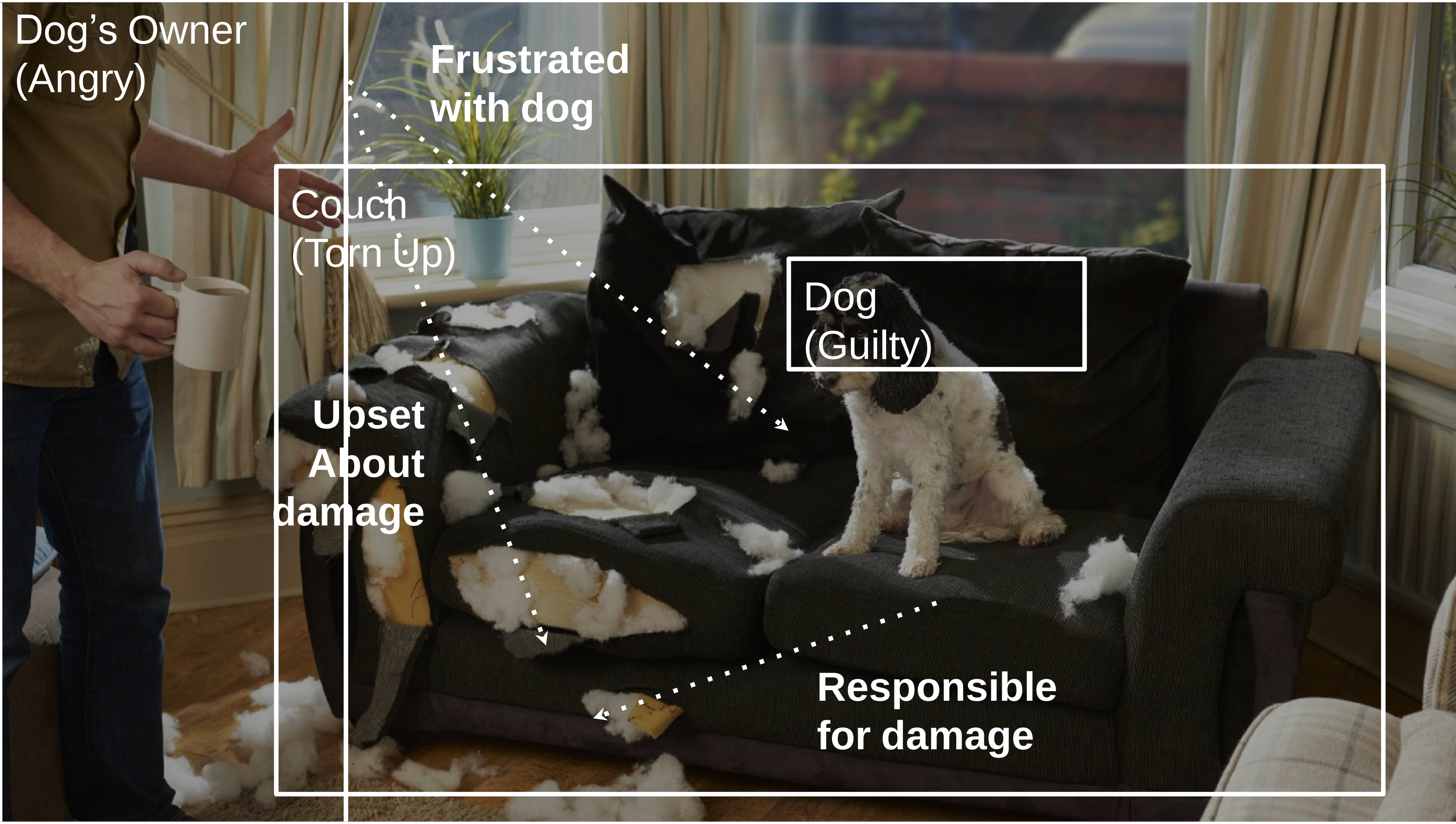
I am hurt

Hello, hurt! 😊

The limits of chatbot conversation



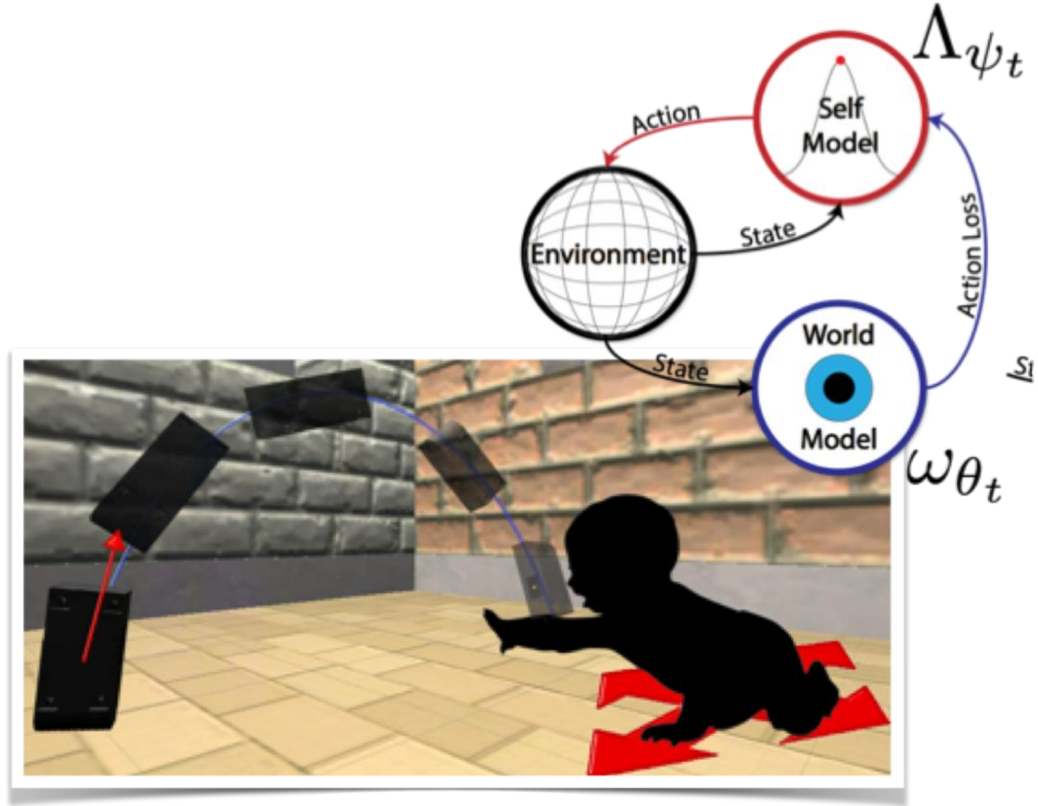
Fei-Fei Li


Dog's Owner
(Angry)
Frustrated
with dog
Couch
(Torn Up)
Dog
(Guilty)
Upset
About
damage
Responsible
for damage


Dog’s Owner
(Angry)
Frustrated
with dog
Couch
(Torn Up)
Context
Situational
Awareness
Prior
Knowledge
Responsible
for damage

Curiosity-based Learning

- A baby's learning is exploratory, curiosity-driven, multi-modal, active and social.
- Can we model this process and apply it in machines?



Mrowca, Haber, Fei-Fei & Yamins, *CogSci,* 2018

“Thinking slow”
Commonsense knowledge and reasoning

- Reasoning requires combining previously acquired knowledge to address new tasks
- Can a neural network reason more like a human?

He served chicken.

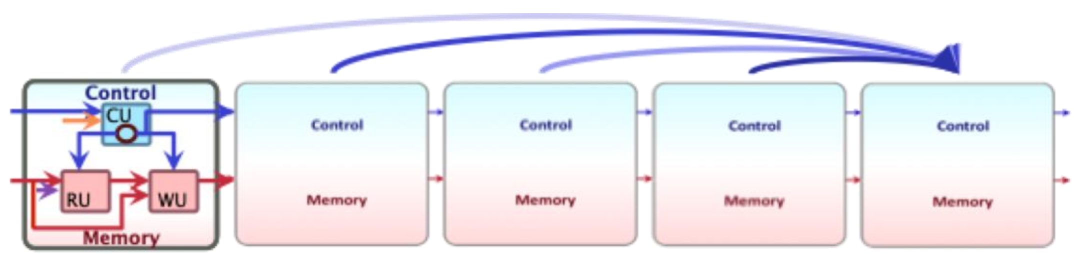
Not like serving a tennis ball

He probably cooked it first!

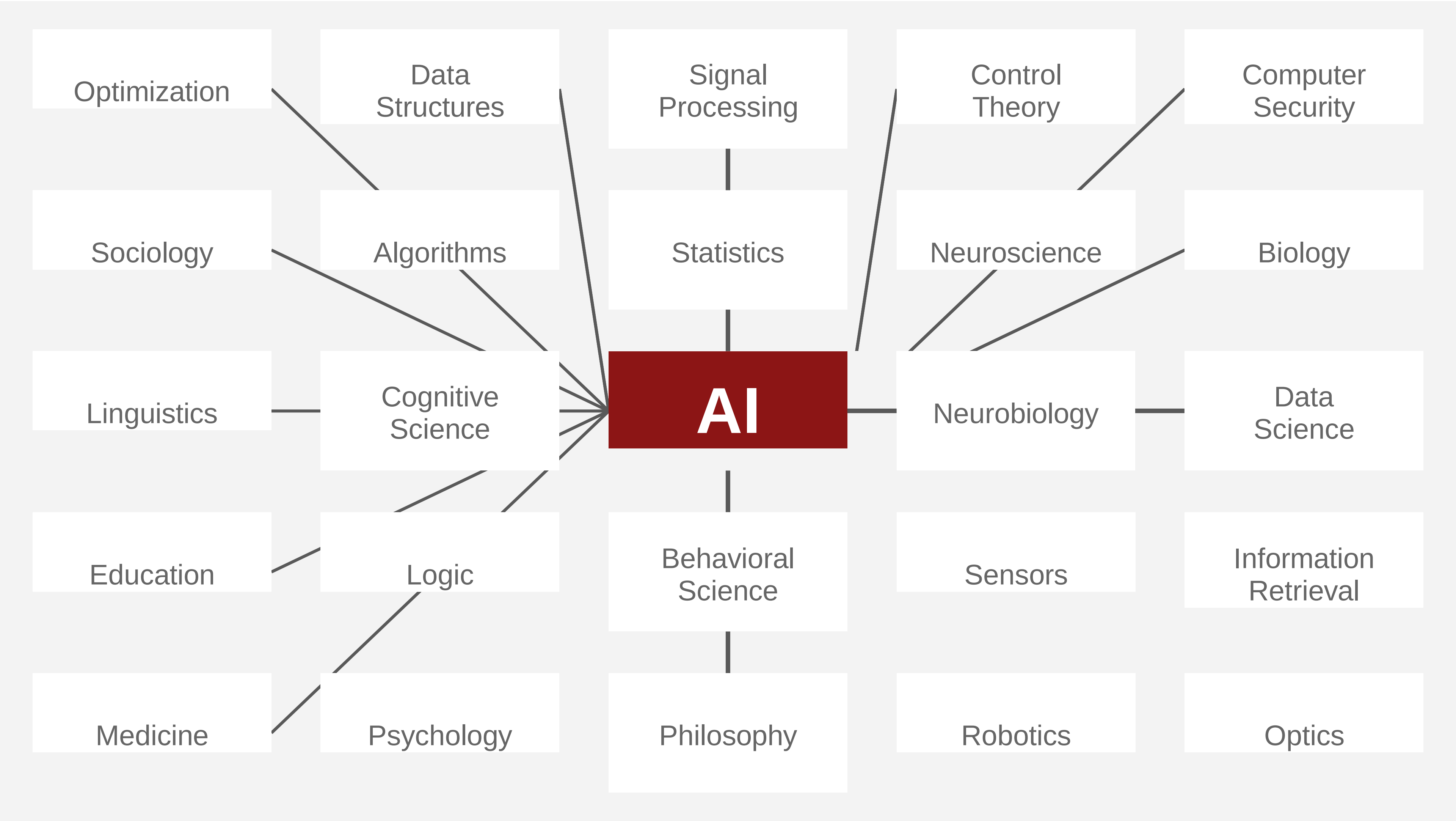
The trophy wouldn’t fit in the suitcase because it was too big.

The trophy?

The suitcase?



Hudson and Manning, 2018


Optimization
Data Structures
Signal Processing
Control Theory
Computer Security
Sociology
Algorithms
Statistics
Neuroscience
Biology
Linguistics
Cognitive Science
AI
Neurobiology
Data Science
Education
Logic
Behavioral Science
Sensors
Information Retrieval
Medicine
Psychology
Philosophy
Robotics
Optics

Fei-Fei Li

~50%

current work activities can be theoretically automated now

100%

current work activities can be potentially **enhanced** by intelligent technology

Enhancing human care with intelligent systems

Hospital-Acquired Infections

99,000 Deaths

Annually

Unmonitored Elderly Fall Injuries

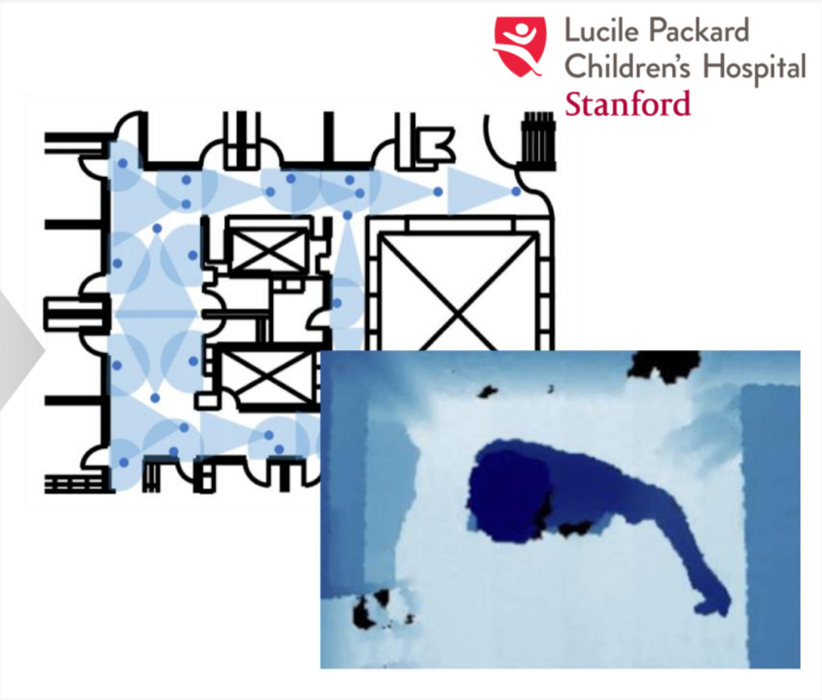
$36.4 Billion

Annually

A. Houser, W. Fox-Grage & K. Ujvari, *AARP Public Policy Institute*, 2012)

Airtek Indoor Air Solutions. 2014. Calfee.

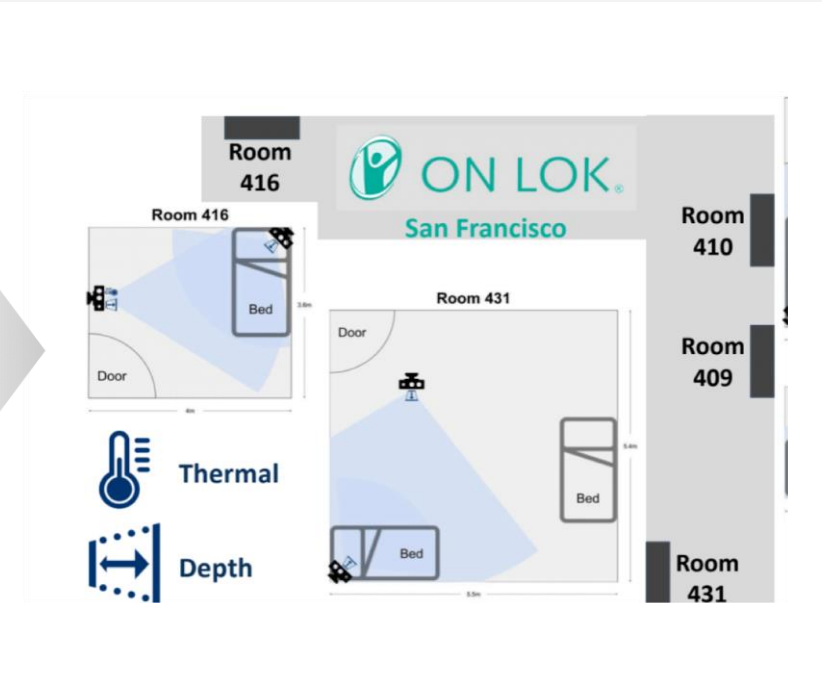
Annual Review of Medicine 2012



From: Inconsistent hand hygiene

To: Intelligent monitors placed throughout hospitals

A. Haque, A. Singh, A. Alahi, S. Yeung, M. Guo, A. Luo, J. Jopling, L. Downing, W. Beninati, T, Platchek, A. Milstein & L. Fei-Fei, *Under review*
A. Haque, E. Peng, A. Luo, A. Alahi,S. Yeung & L. Fei-Fei, *ECCV*, 2016



From: Ineffective wearables, lack of human caretakers

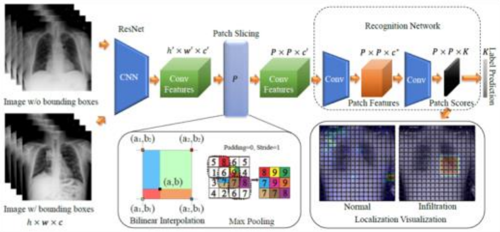
To: Intelligent monitors placed throughout senior living homes

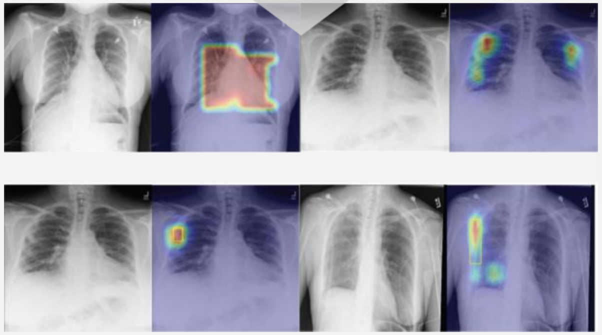
A. Luo, T. Hsieh, R. Rege, A. Mehra,G. Pusiol, L. Downing, A. Milstein & L. Fei-Fei. *In preparation.*

Giving human specialists more time


Lowers costs
Improves safety and outcomes
Reduces burden on human caregivers

An algorithm for automating simple radiology analysis





More time for human specialists to do what they do best

Z. Li, C. Wang, M. Han, Y. Xue, W. Wei, Li-J. Li, L. Fei-Fei, *CVPR*, *2018*