

Object Detection using Deep Learning



DOG, DOG, CAT



[Image credit](#)

Roadmap

Classification



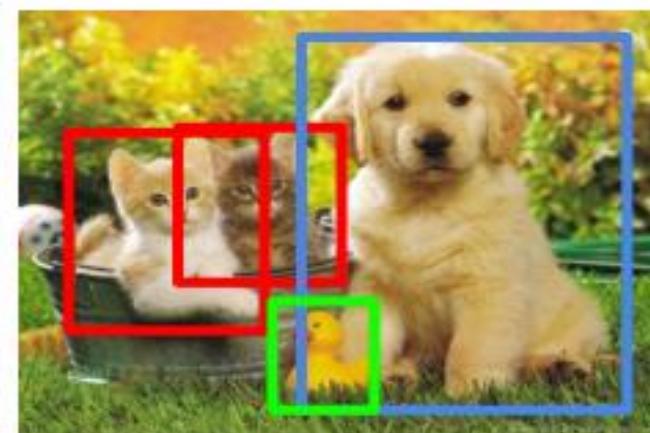
CAT

Classification + Localization



CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation



CAT, DOG, DUCK

Single object

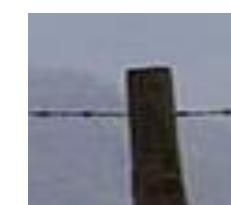
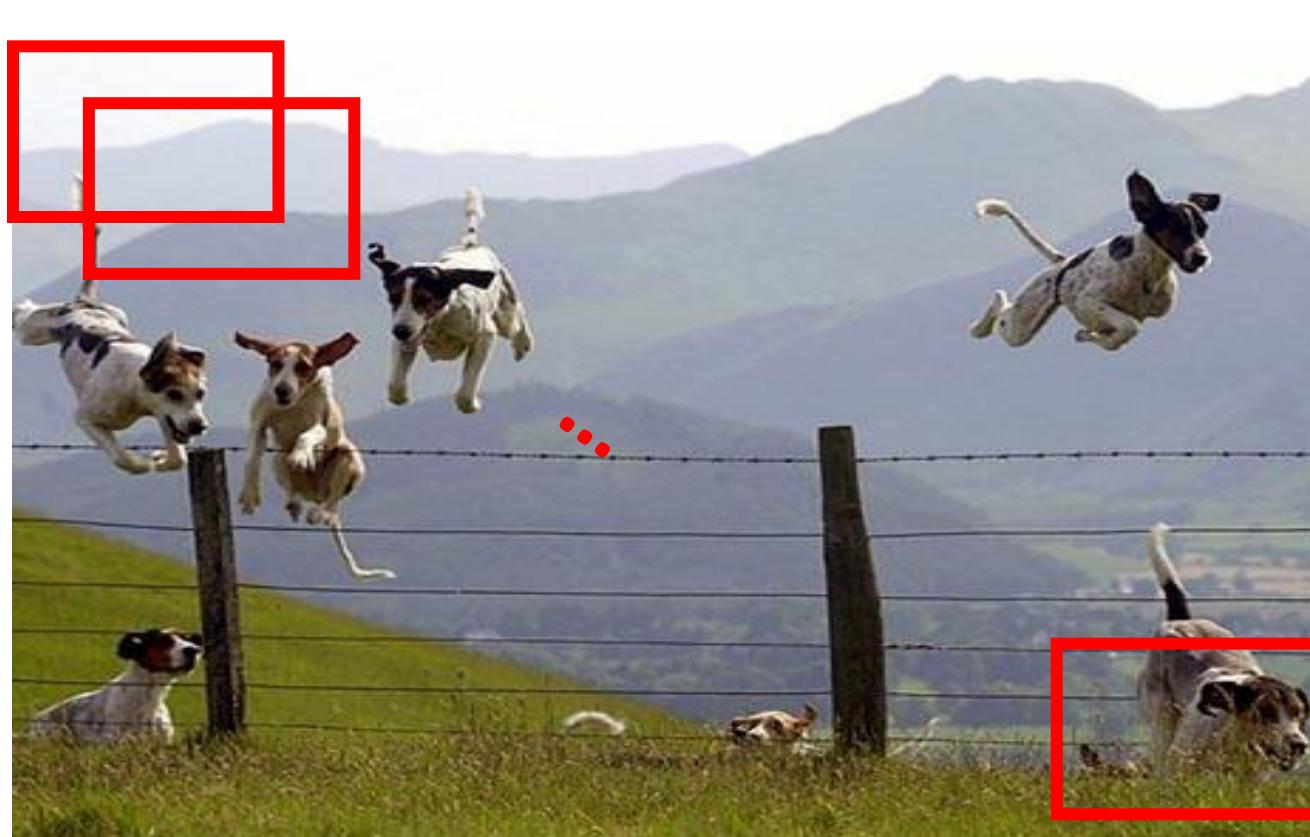
Multiple objects

Today's class

- Object category detection
- Deep learning methods
 - Two-stage models: R-CNN
 - One-stage models: YOLO, SSD, Retina Net

Object Category Detection

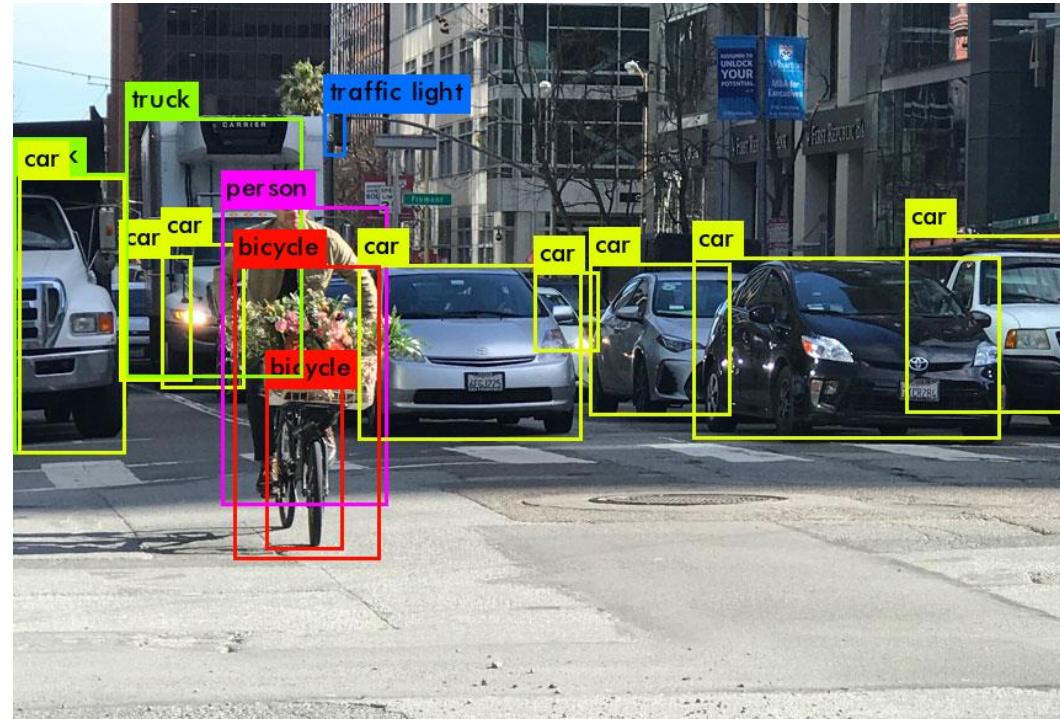
- Focus on object search: “Where is it?”
- Build templates that quickly differentiate object patch from background patch



**Object or
Non-Object?**

What are the challenges of object detection?

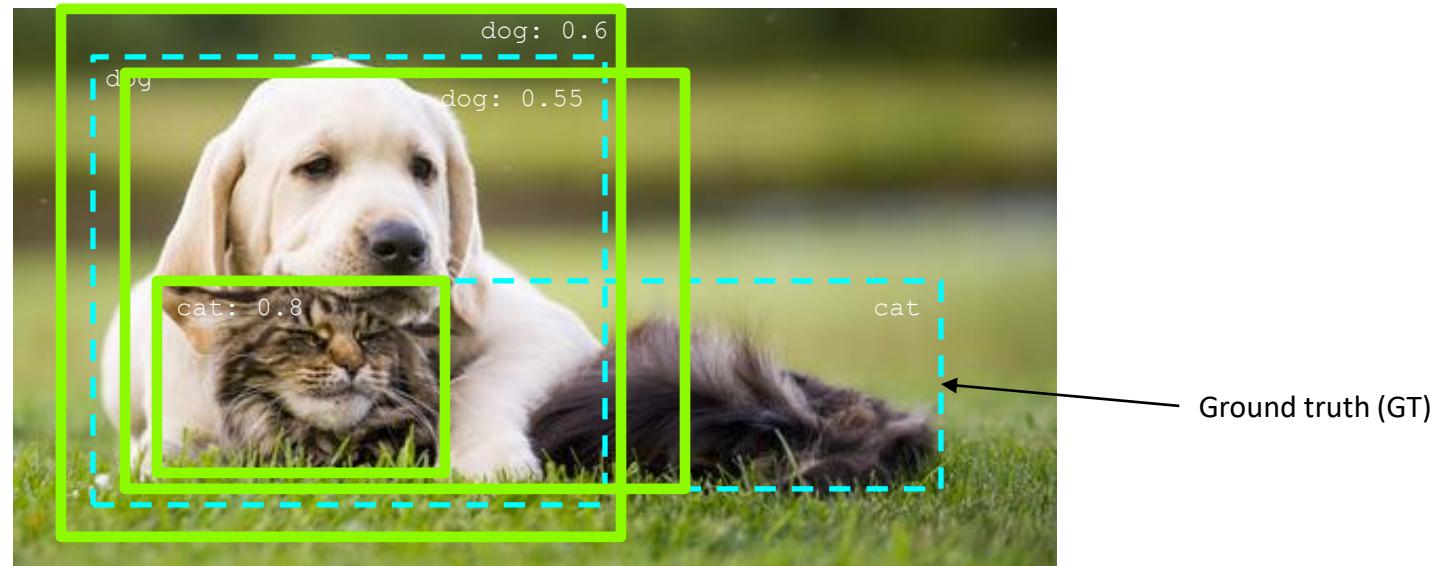
- Images may contain more than one class, multiple instances from the same class
- Bounding box localization
- Evaluation



[Image source](#)

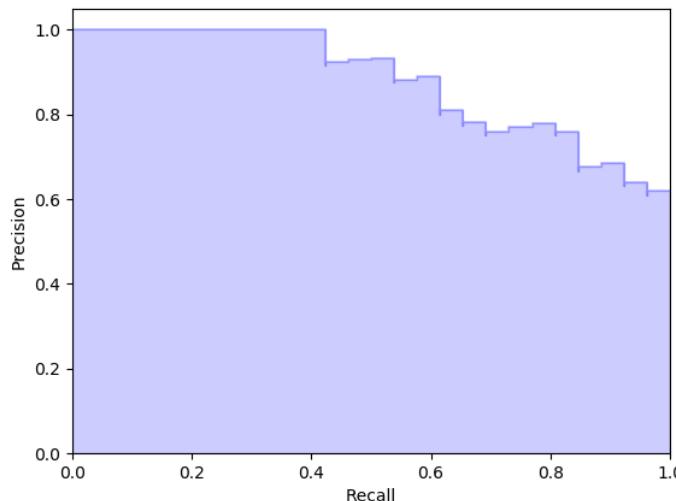
Object detection evaluation

- At test time, predict bounding boxes, class labels, and confidence scores
- For each detection, determine whether it is a true or false positive
 - PASCAL criterion: $\text{Area}(\text{GT} \cap \text{Det}) / \text{Area}(\text{GT} \cup \text{Det}) > 0.5$
 - For multiple detections of the same ground truth box, only one is considered a true positive



Object detection evaluation

- At test time, predict bounding boxes, class labels, and confidence scores
- For each detection, determine whether it is a true or false positive
- For each class, sort detections from highest to lowest confidence, plot Recall-Precision curve and compute Average Precision (area under the curve)
- Take mean of AP over classes to get mAP



Precision: true positive detections / total detections

Recall: true positive detections / total positive test instances

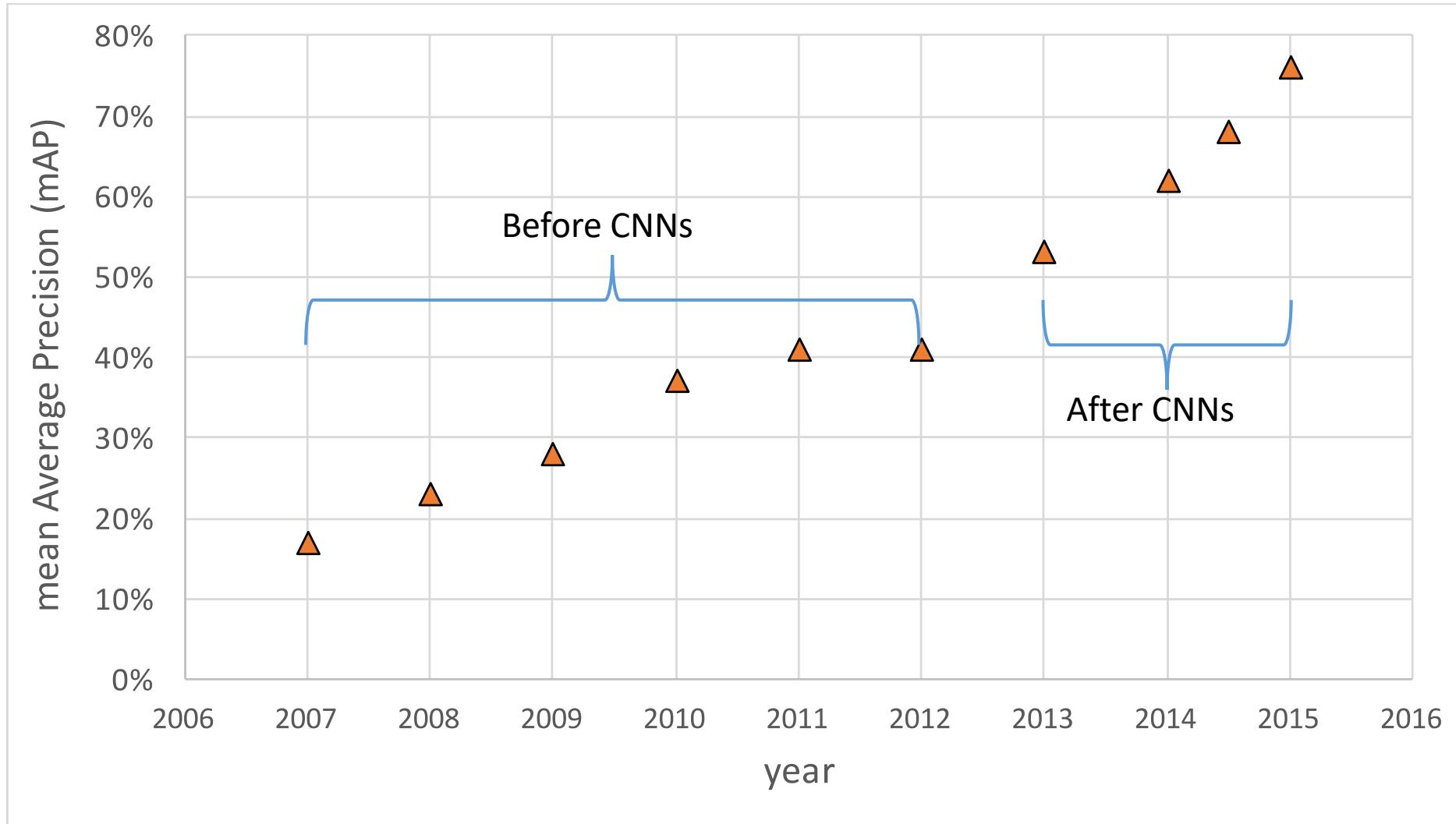
PASCAL VOC Challenge (2005-2012)



- 20 challenge classes:
 - *Person*
 - *Animals*: bird, cat, cow, dog, horse, sheep
 - *Vehicles*: airplane, bicycle, boat, bus, car, motorbike, train
 - *Indoor*: bottle, chair, dining table, potted plant, sofa, tv/monitor
- Dataset size (by 2012): 11.5K training/validation images, 27K bounding boxes, 7K segmentations

Progress on PASCAL detection

PASCAL VOC



Current benchmark: COCO

What is COCO?



COCO is a large-scale object detection, segmentation, and captioning dataset.
COCO has several features:

- ✓ Object segmentation
- ✓ Recognition in context
- ✓ Superpixel stuff segmentation
- ✓ 330K images (>200K labeled)
- ✓ 1.5 million object instances
- ✓ 80 object categories
- ✓ 91 stuff categories
- ✓ 5 captions per image
- ✓ 250,000 people with keypoints



<http://cocodataset.org/#home>

COCO dataset: Tasks

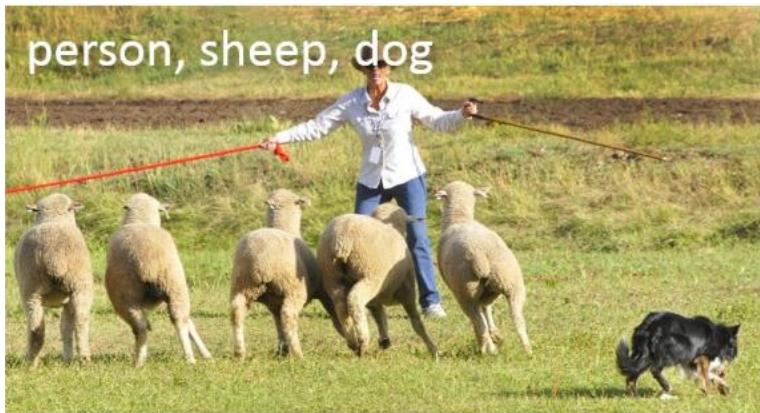
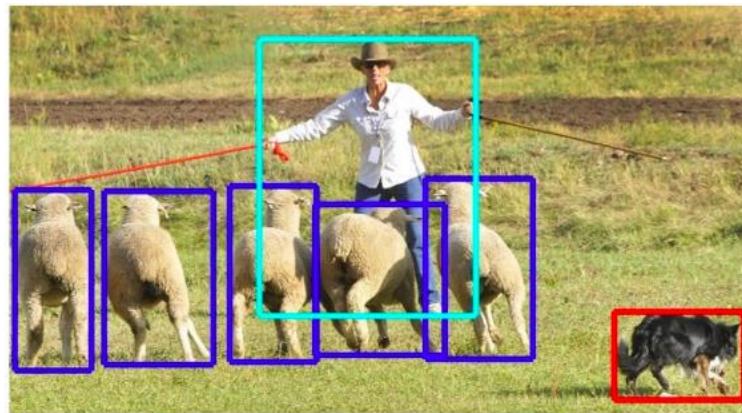
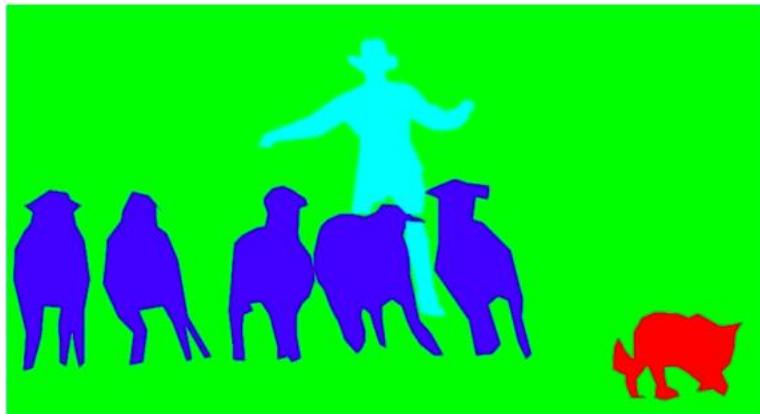


image classification



object detection



semantic segmentation

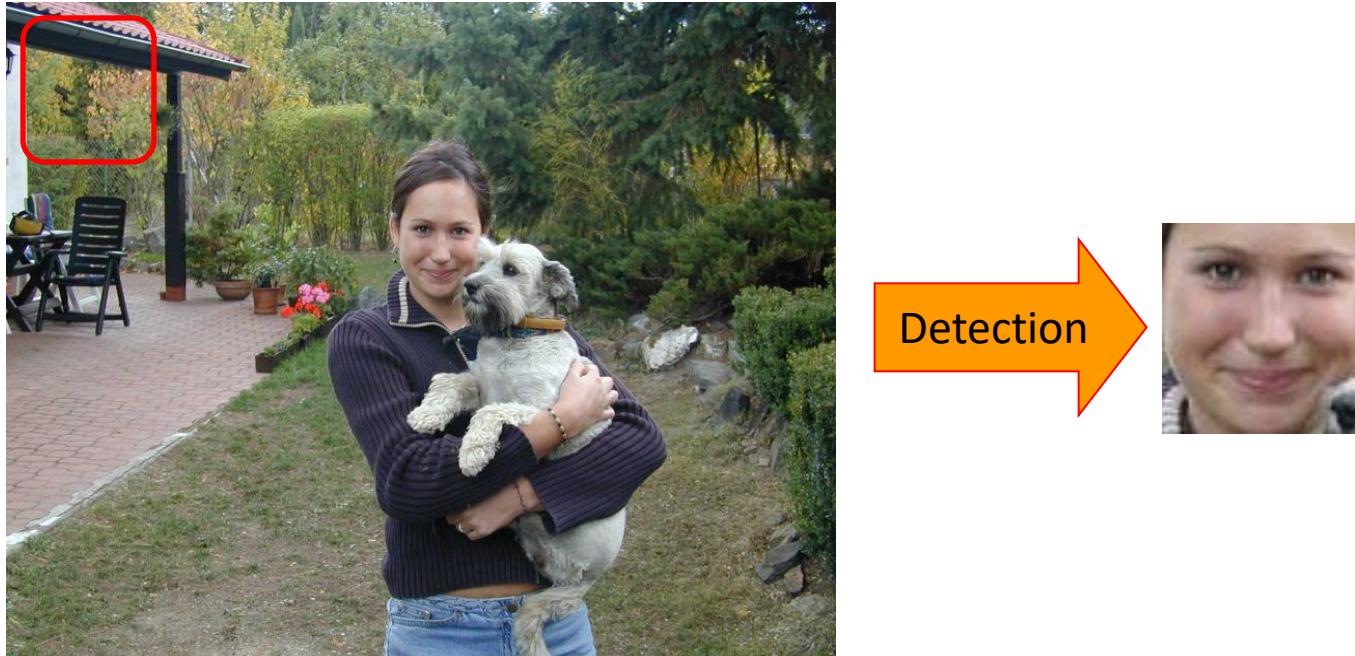


instance segmentation

- Also:
 - keypoint prediction,
 - captioning,
 - question answering
 - ...

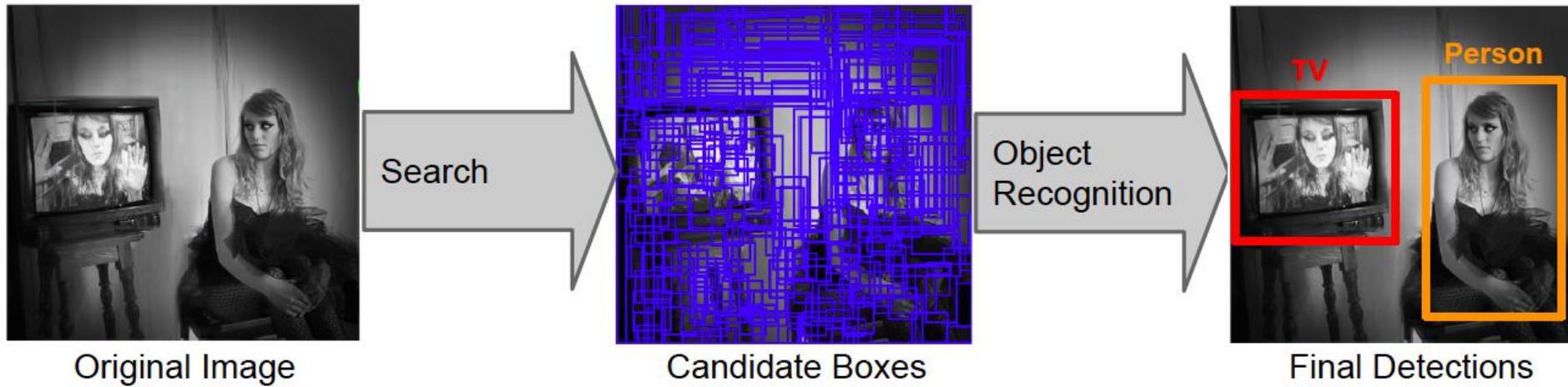
- Leaderboard: <http://cocodataset.org/#detection-leaderboard>
- Official COCO challenges no longer include detection
 - Emphasis has shifted to instance segmentation and dense semantic segmentation

Approaches to detection: Sliding windows



- Slide a window across the image and evaluate a detection model at each location
 - Thousands of windows to evaluate: efficiency and low false positive rates are essential
 - Difficult to extend to a large range of scales, aspect ratios

Approaches to detection: Object proposals



- Generate and evaluate a few hundred region proposals
 - Proposal mechanism can take advantage of low-level perceptual organization cues
 - Proposal mechanism can be category-specific or category-independent, hand-crafted or trained
 - Classifier can be slower but more powerful

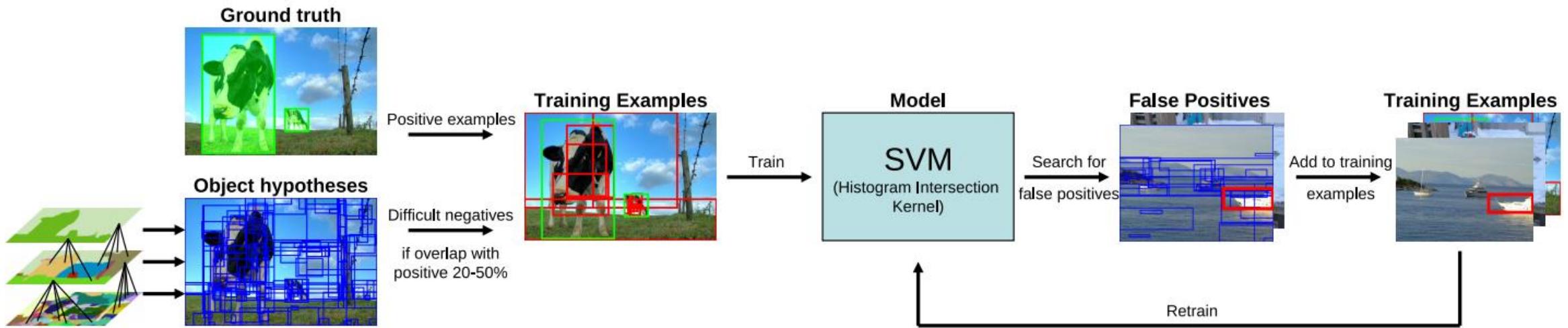
Selective search for detection

- Use hierarchical segmentation: start with small *superpixels* and merge based on diverse cues



J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, [Selective Search for Object Recognition](#), IJCV 2013

Selective search for detection

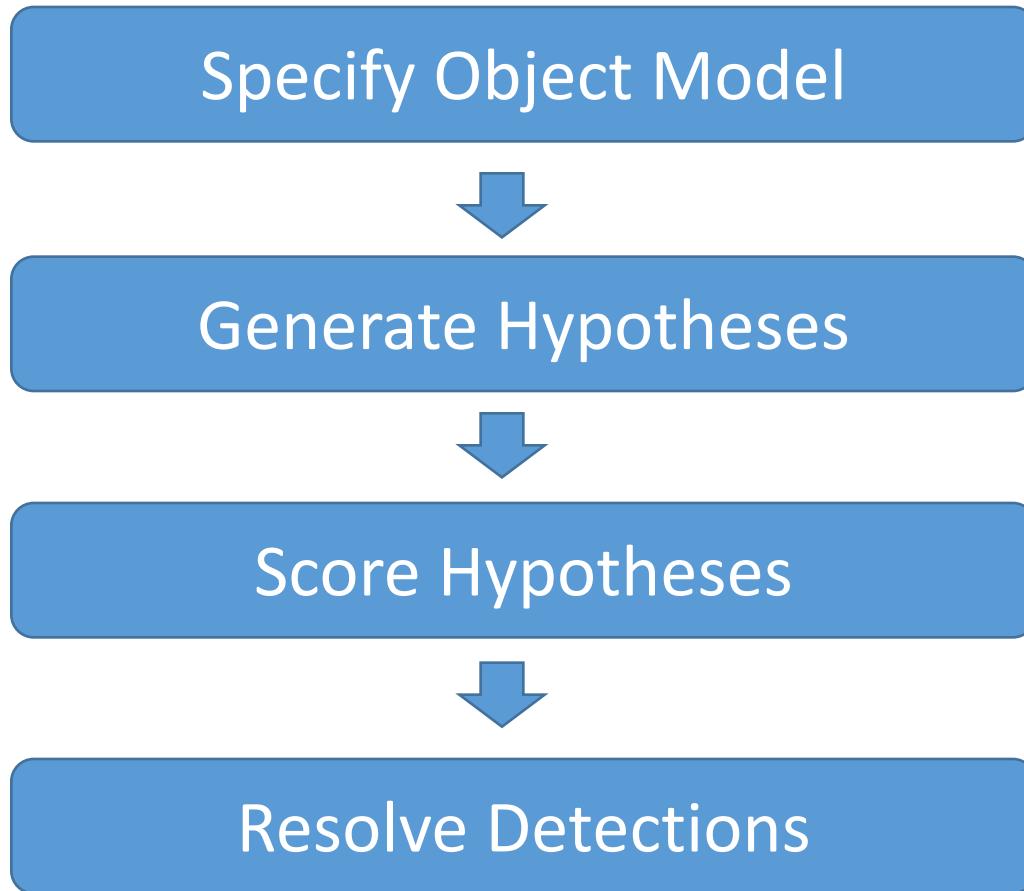


- Feature extraction: color SIFT, codebook of size 4K, spatial pyramid with four levels = 360K dimensions

Approaches to detection

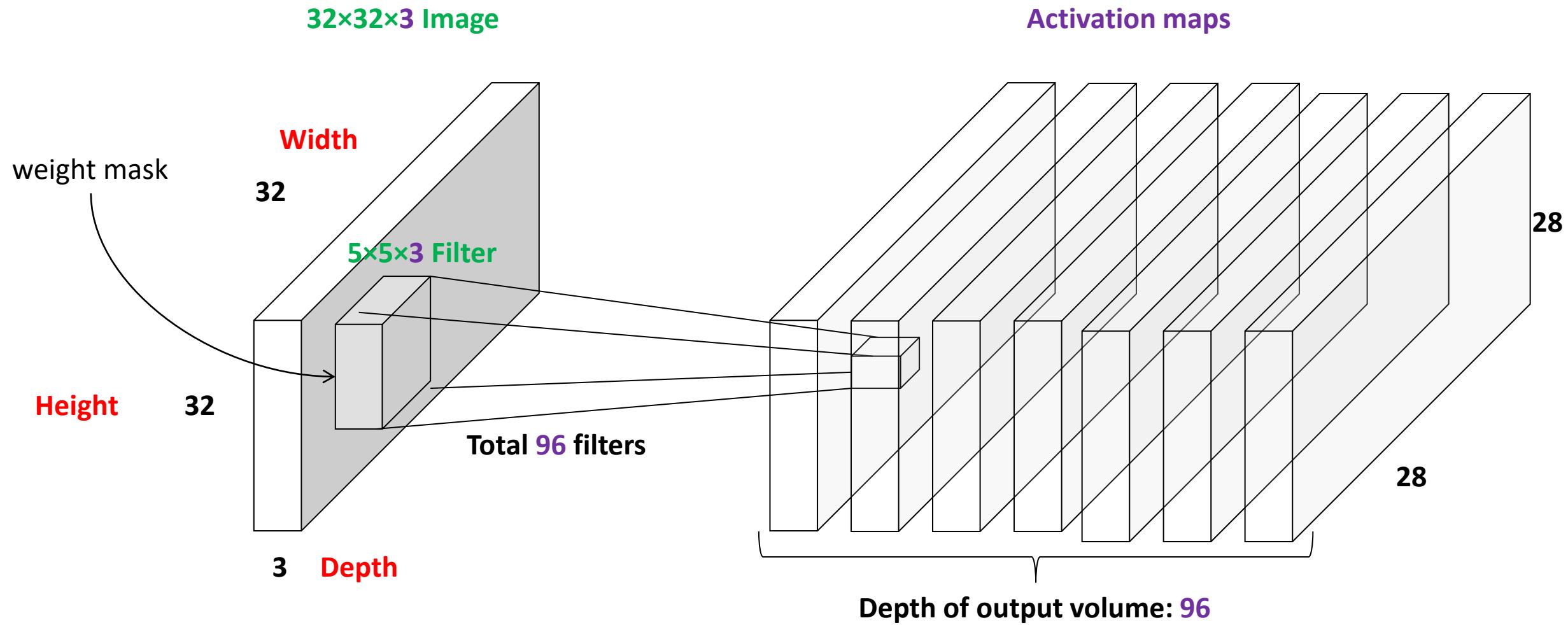
- Before ~2010, dominated by sliding windows
- 2010-2013: proposal-driven
- Deep learning approaches started as proposal-driven, but have evolved back toward sliding windows
- Most recently, “global” methods are becoming more common

General Process of Object Recognition

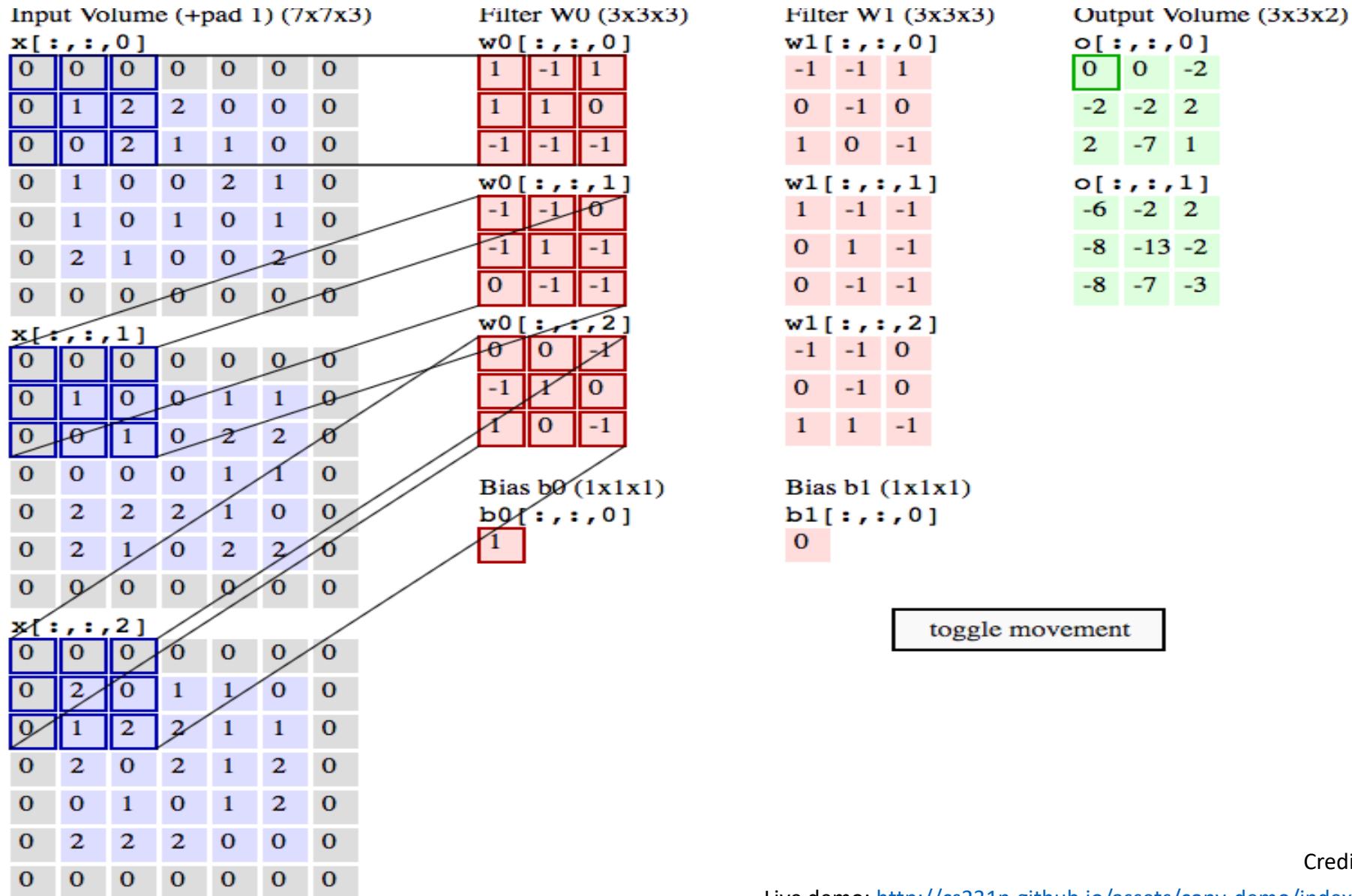


Convolutional Neural Network (Quick Recap)

Recap – Convolutional layer



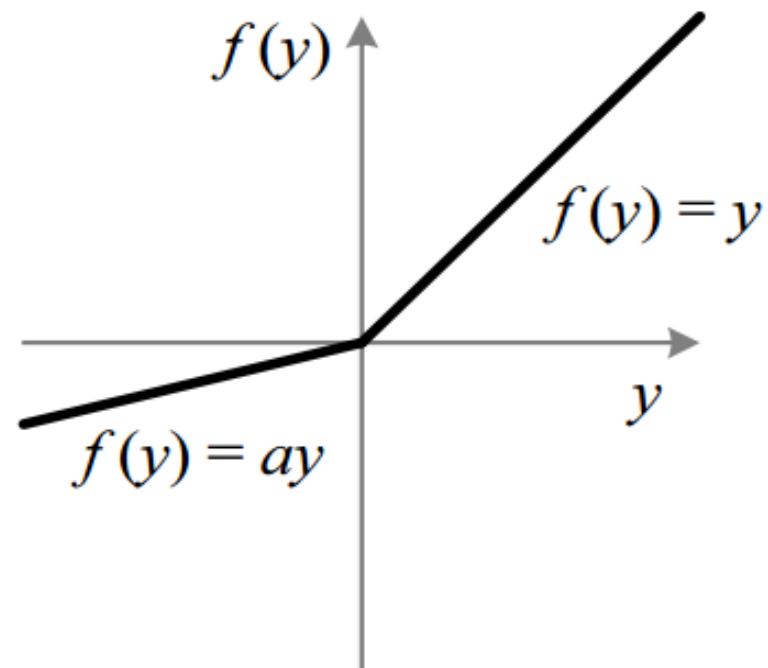
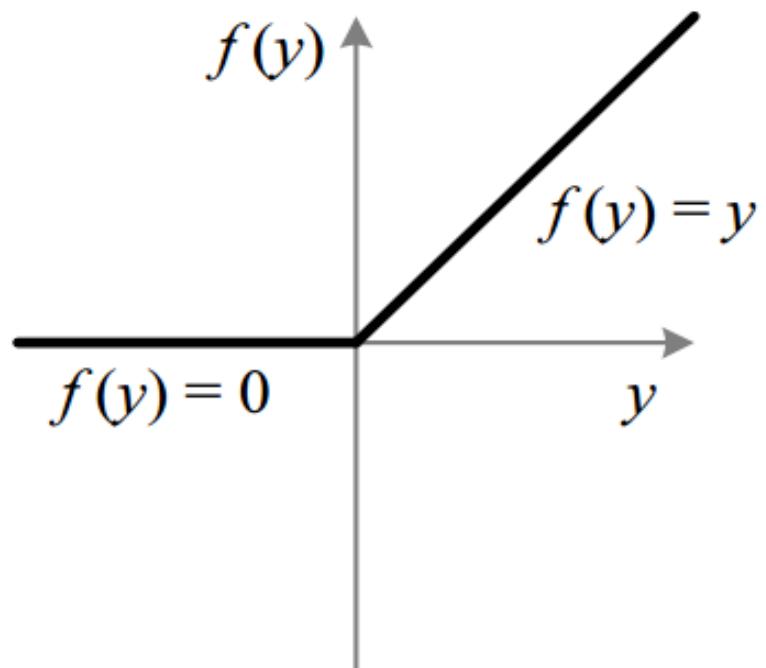
How it works?



Credit: Andrej Karpathy

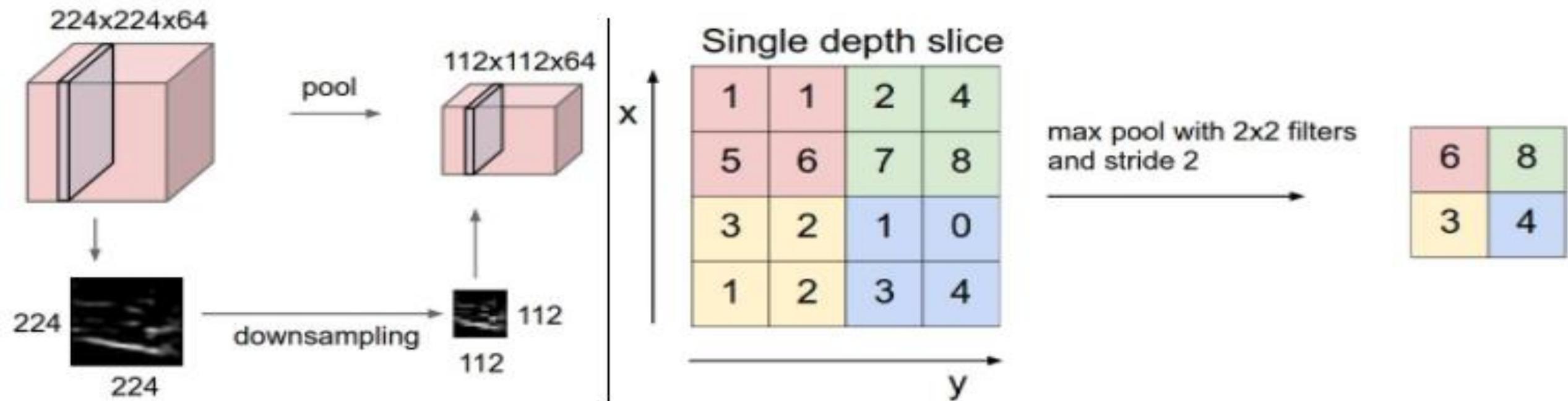
Live demo: <http://cs231n.github.io/assets/conv-demo/index.html>

Recap – Activation



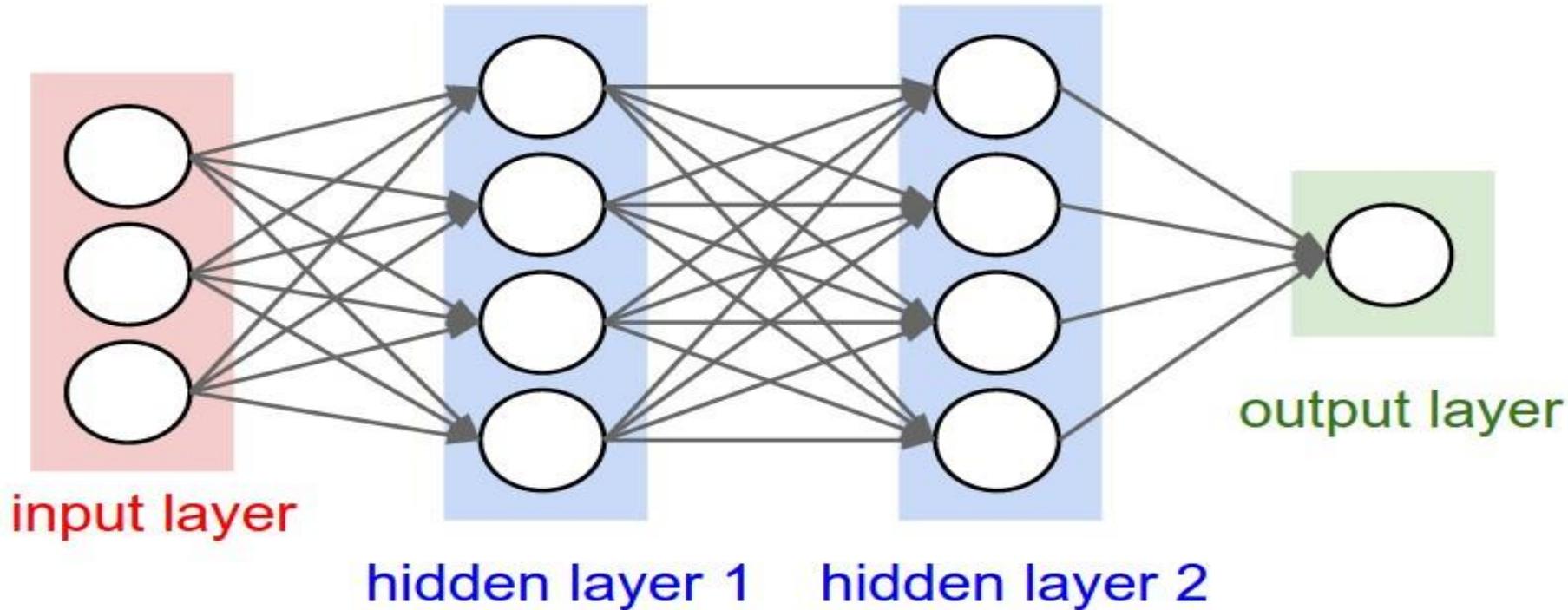
- Introduce the non-linearity

Recap – Pooling layer



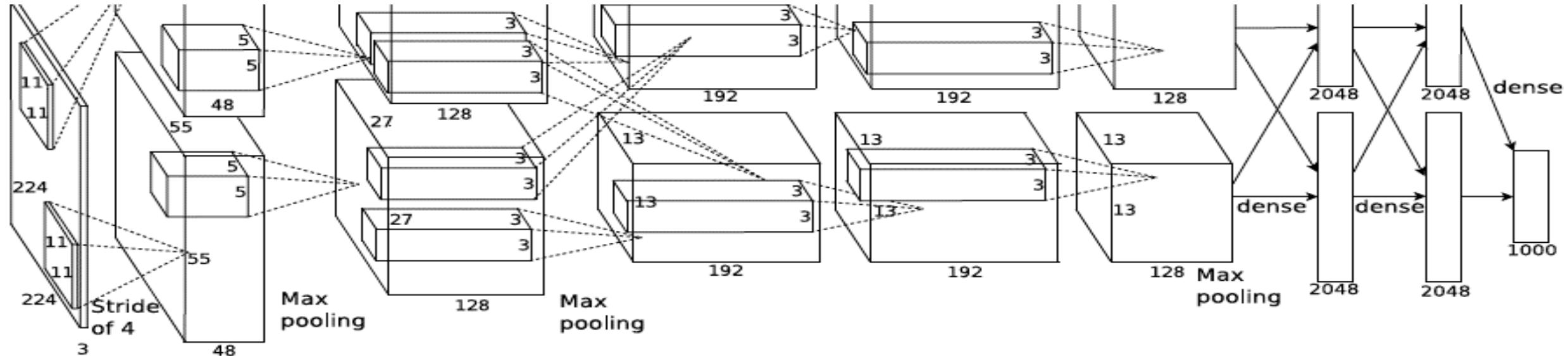
- Reduce the feature size
- Introduce a bit invariance (translation, rotation)

Recap – Fully-connected layer



- Each output node is connected to all the input nodes
- Fixed number of input nodes
- Fixed number of output nodes

Put them all together



- Train the deep convolutional neural net with simple chain-rule (a.k.a back propagation)

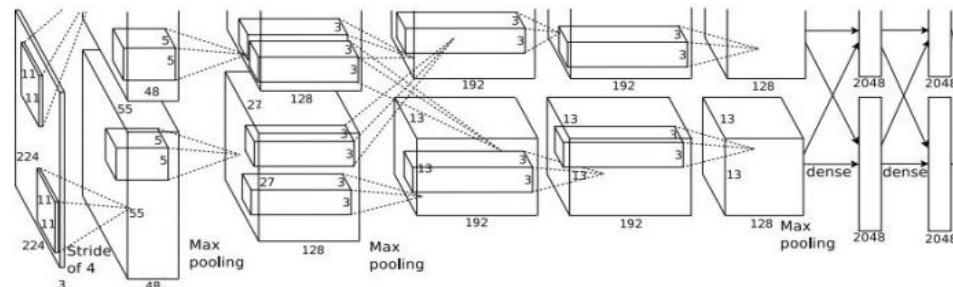
CNN methods for object detection

CNN as feature extractor



CNN as feature extractor

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

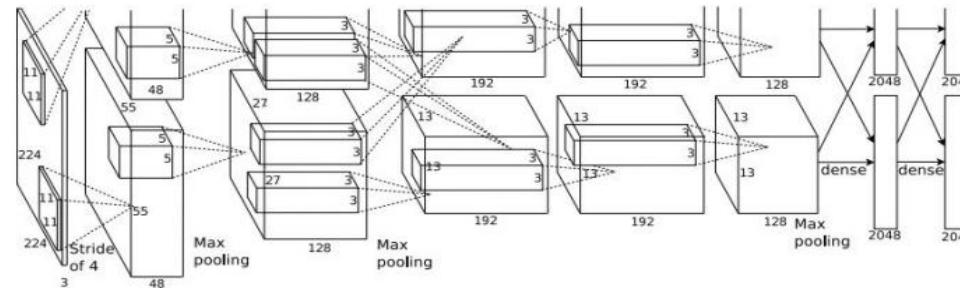


Dog? NO
Cat? NO
Background? YES

CNN as feature extractor



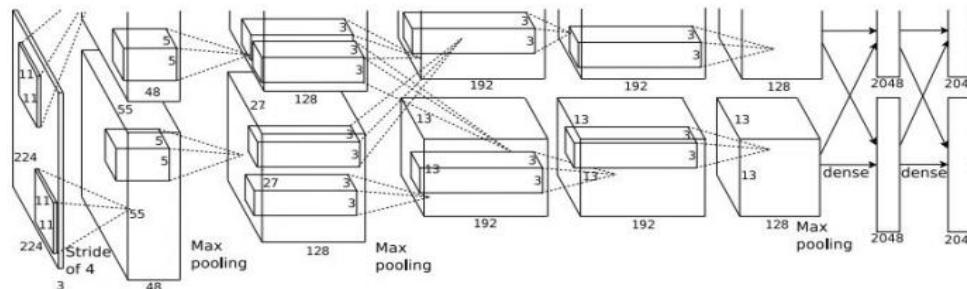
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

CNN as feature extractor

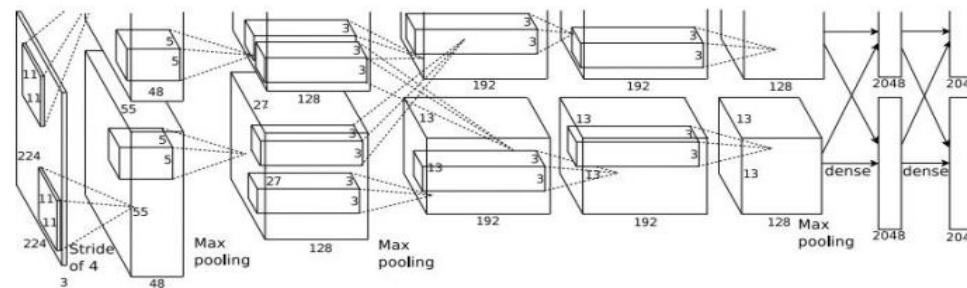
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

CNN as feature extractor

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

CNN as feature extractor

- What could be the problems?

CNN as feature extractor

- What could be the problems?
 - Suppose we have a 600×600 image, if sliding window size is 20×20 , then have $(600-20+1) \times (600-20+1) = \sim 330,000$ windows

CNN as feature extractor

- What could be the problems?

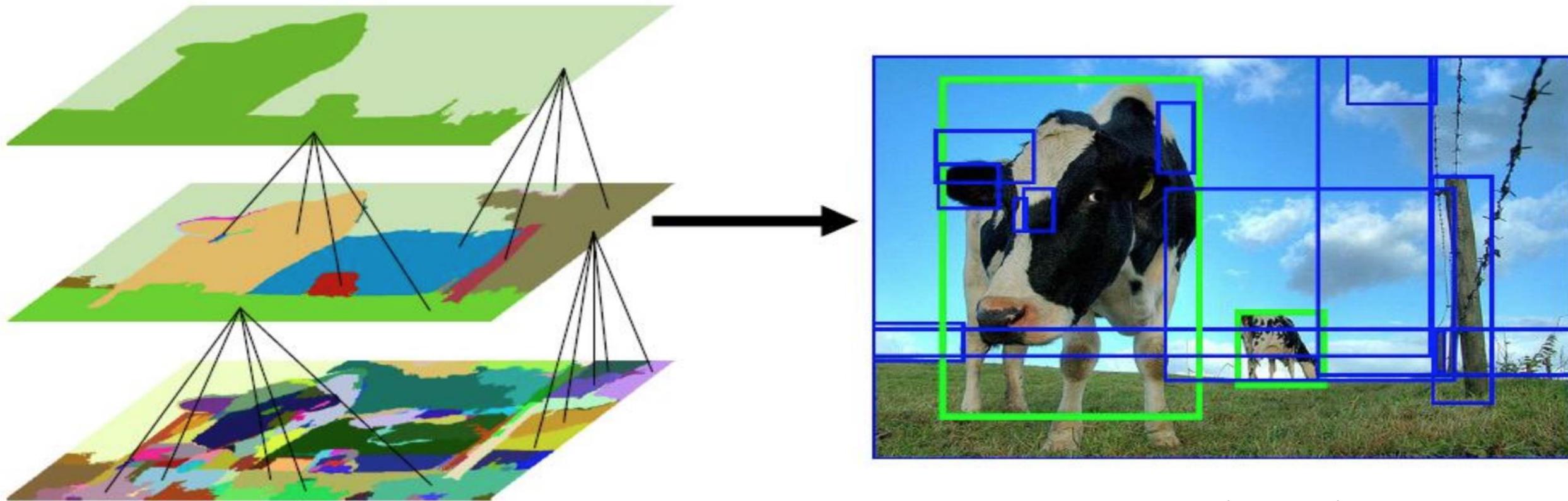
- Suppose we have a 600×600 image, if sliding window size is 20×20 , then have $(600-20+1) \times (600-20+1) = \sim 330,000$ windows
- Sometimes we want to have more accurate results -> multi-scale detection
 - Resize image
 - Multi-scale sliding window

CNN as feature extractor

- What could be the problems?
 - Suppose we have a 600×600 image, if sliding window size is 20×20 , then have $(600-20+1) \times (600-20+1) = \sim 330,000$ windows
 - Sometimes we want to have more accurate results -> multi-scale detection
 - Resize image
 - Multi-scale sliding window
 - For each image, we need to do the forward pass in the CNN for $\sim 330,000$ times. -> Slow!!!

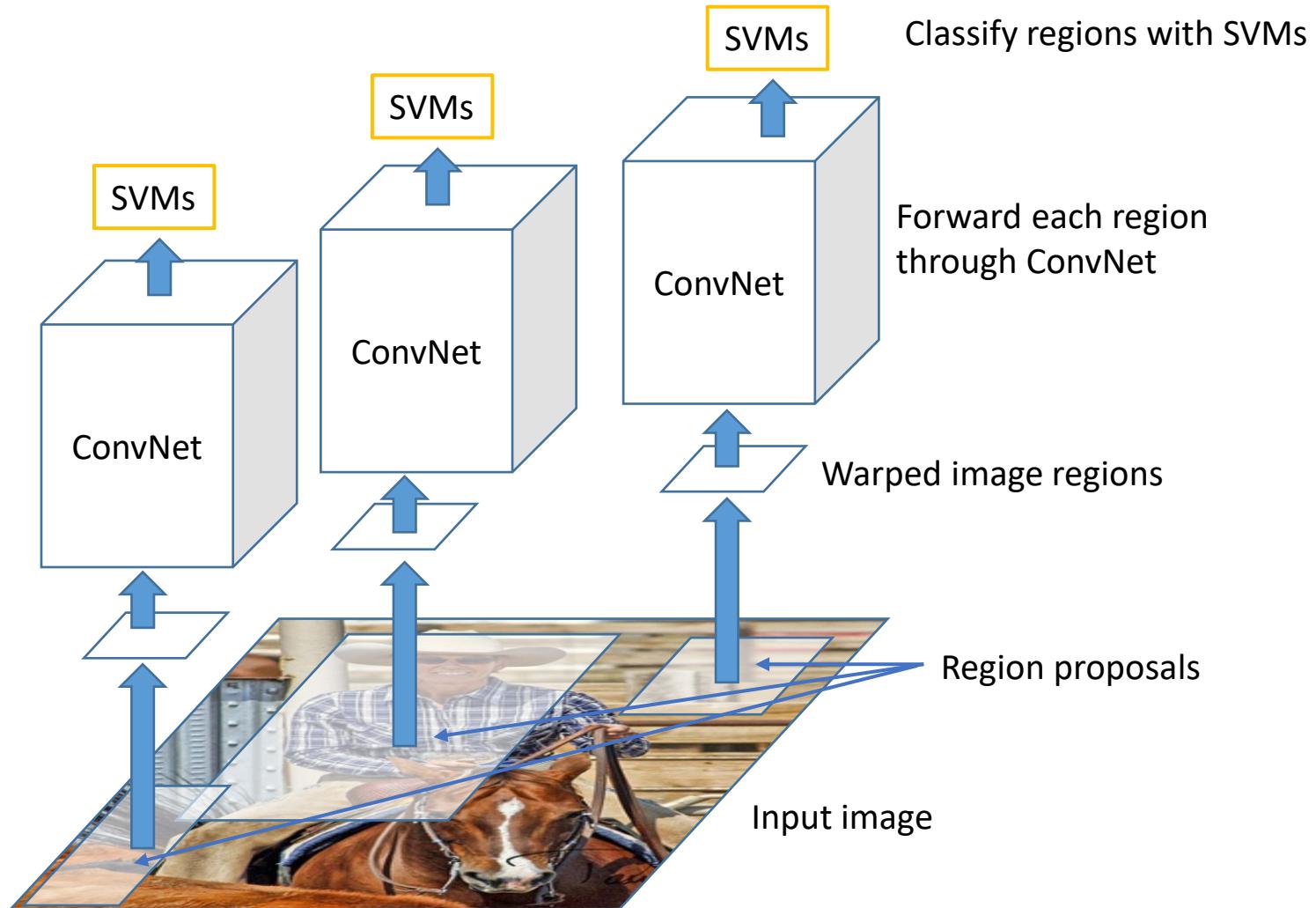
Region Proposal

- Solution
 - Use some fast algorithms to filter out some regions first, only feed the potential region (region proposals) into CNN
 - E.g. selective search



R-CNN: Region proposals + CNN features

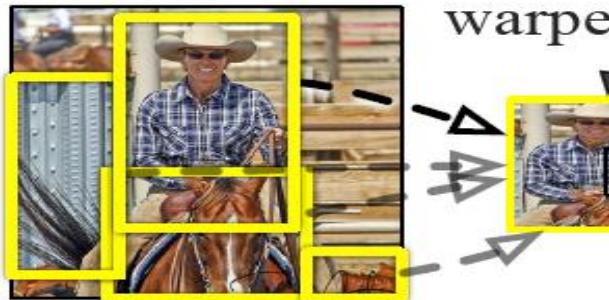
Source: R. Girshick



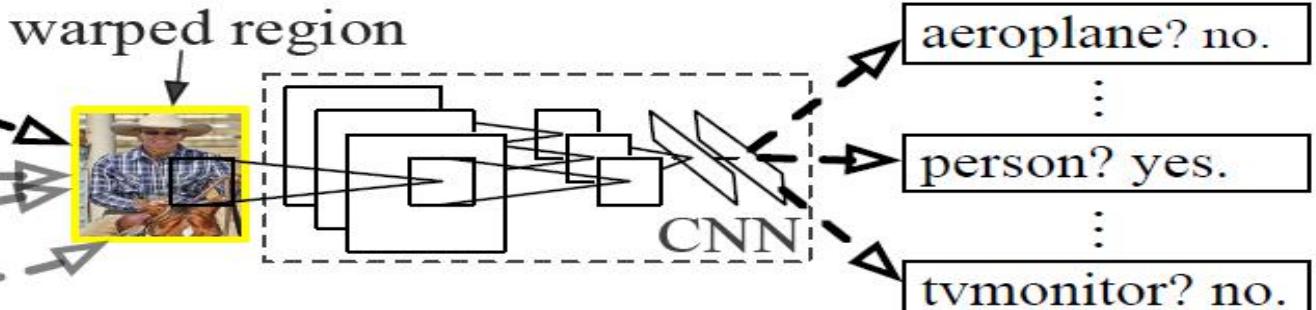
R-CNN (Girshick et al. CVPR 2014)



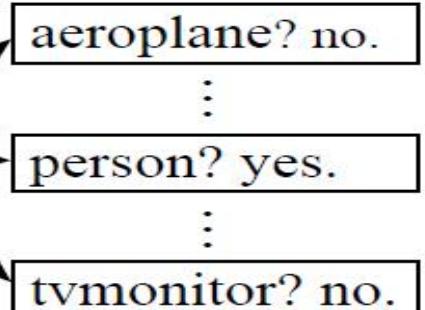
1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features



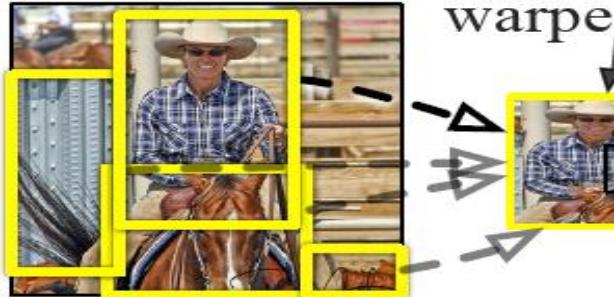
4. Classify regions

- Replace sliding windows with “selective search” region proposals (Uijlings et al. IJCV 2013)
- Extract rectangles around regions and resize to 227x227
- Extract features with fine-tuned CNN (that was initialized with network trained on ImageNet before training)
- Classify last layer of network features with SVM, refine bounding box localization (bbox regression) simultaneously

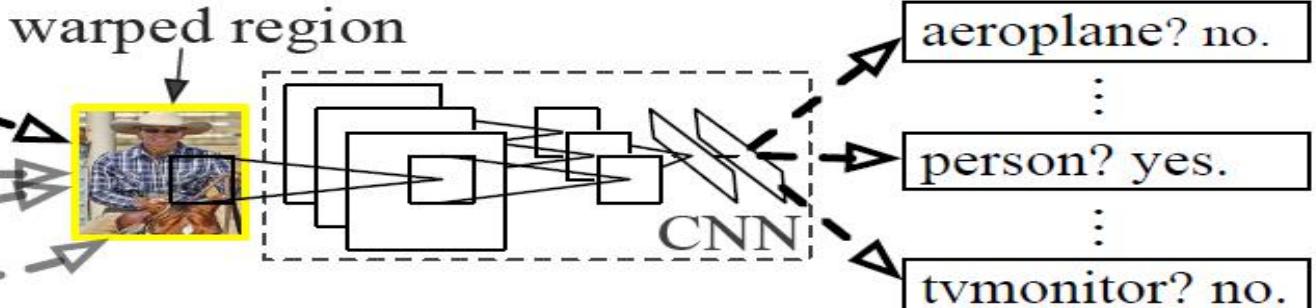
R-CNN (Girshick et al. CVPR 2014)



1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features

4. Classify regions

- **Regions:** ~2000 Selective Search proposals
- **Network:** AlexNet *pre-trained* on ImageNet (1000 classes), *fine-tuned* on PASCAL (21 classes)
- **Final detector:** warp proposal regions, extract fc7 network activations (4096 dimensions), classify with linear SVM
- **Bounding box regression** to refine box locations
- **Performance:** mAP of 53.7% on PASCAL 2010
(vs. 35.1% for Selective Search and 33.4% for Deformable Part Models)

R-CNN pros and cons

- **Pros**
 - Much more accurate than previous approaches!
 - Any deep architecture can immediately be “plugged in”
- **Cons**
 - Not a single end-to-end system
 - Fine-tune network with softmax classifier (log loss)
 - Train post-hoc linear SVMs (hinge loss)
 - Train post-hoc bounding-box regressions (least squares)
 - Training was slow (84h), took up a lot of storage
 - 2000 CNN passes per image
 - Inference (detection) was slow (47s / image with VGG16)

Bounding Box Regression

- Intuition

- If you observe part of the object, according to the seen examples, you should be able to refine the localization
- E.g. given the red box below, since you've seen many airplanes, you know this is not a good localization, you will adjust it to the green one



Bounding Box Regression

- Intuition

- If you observe part of the object, according to the seen examples, you should be able to refine the localization
- E.g. given the red box below, since you've seen many airplanes, you know this is not a good localization, you will adjust it to the green one



R-CNN (Girshick et al. CVPR 2014)

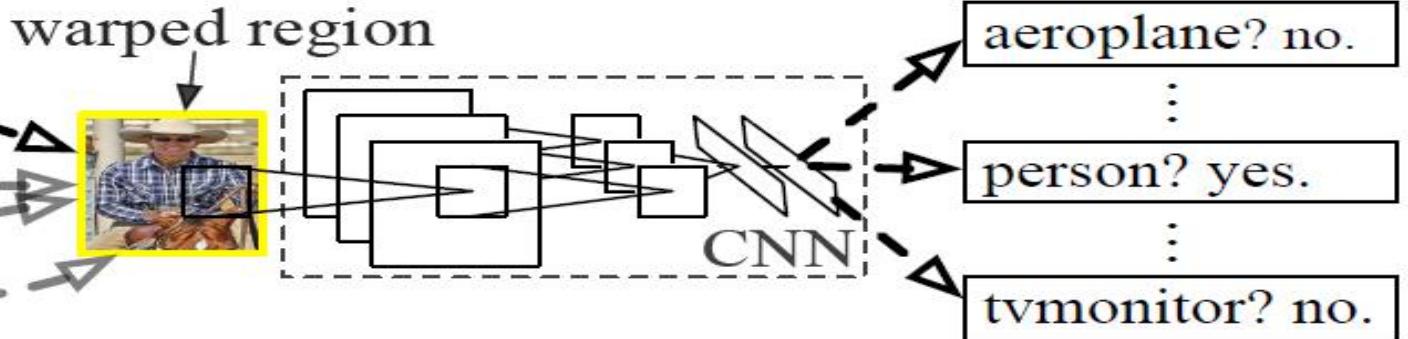
- What could be the problems?



1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features

4. Classify regions

R-CNN (Girshick et al. CVPR 2014)

- What could be the problems?

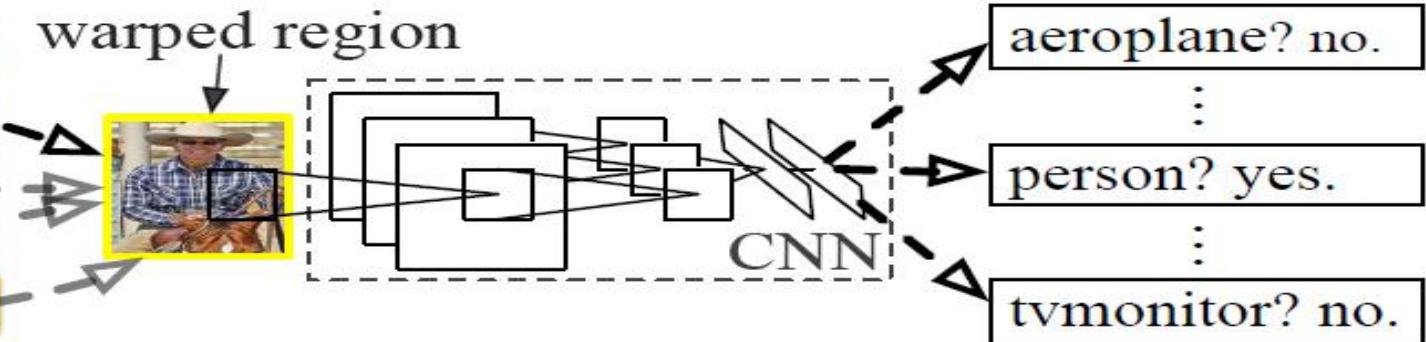
- Repetitive computation! For overlapping regions, we feed it multiple times into CNN



1. Input image



2. Extract region proposals (~2k)

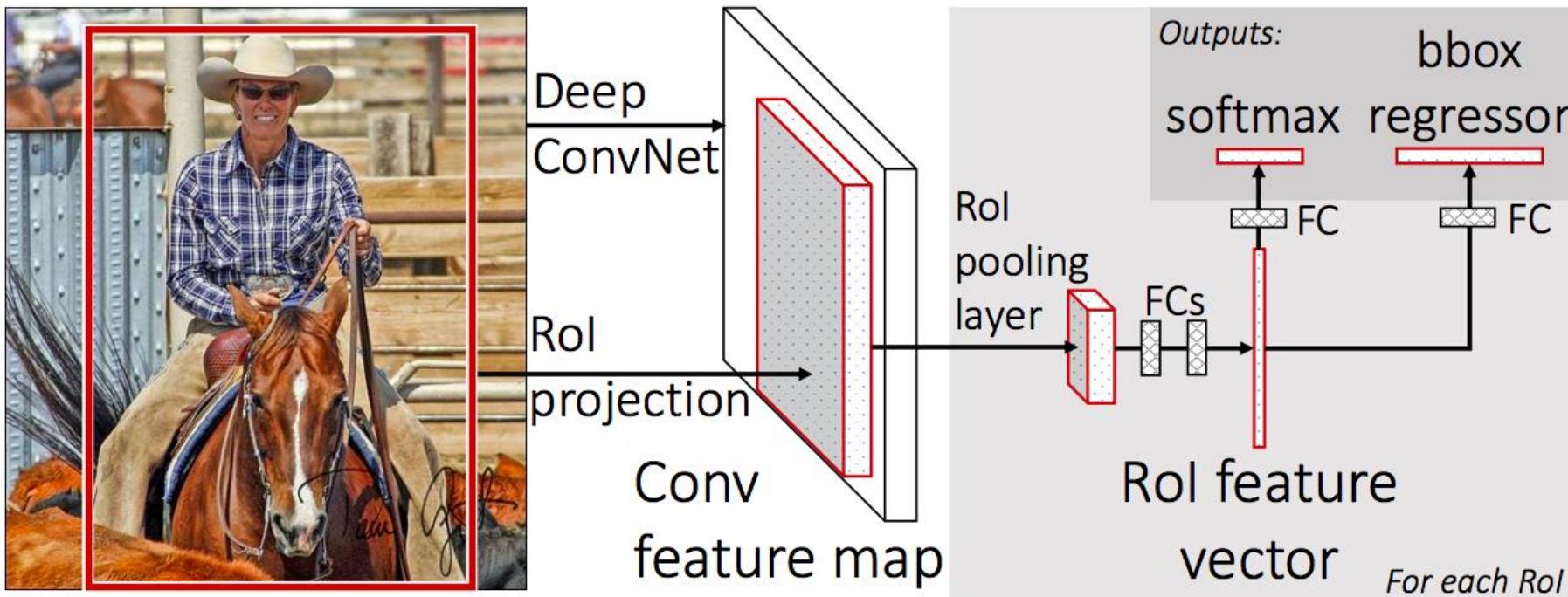


3. Compute CNN features

4. Classify regions

Fast R-CNN (Girshick ICCV 2015)

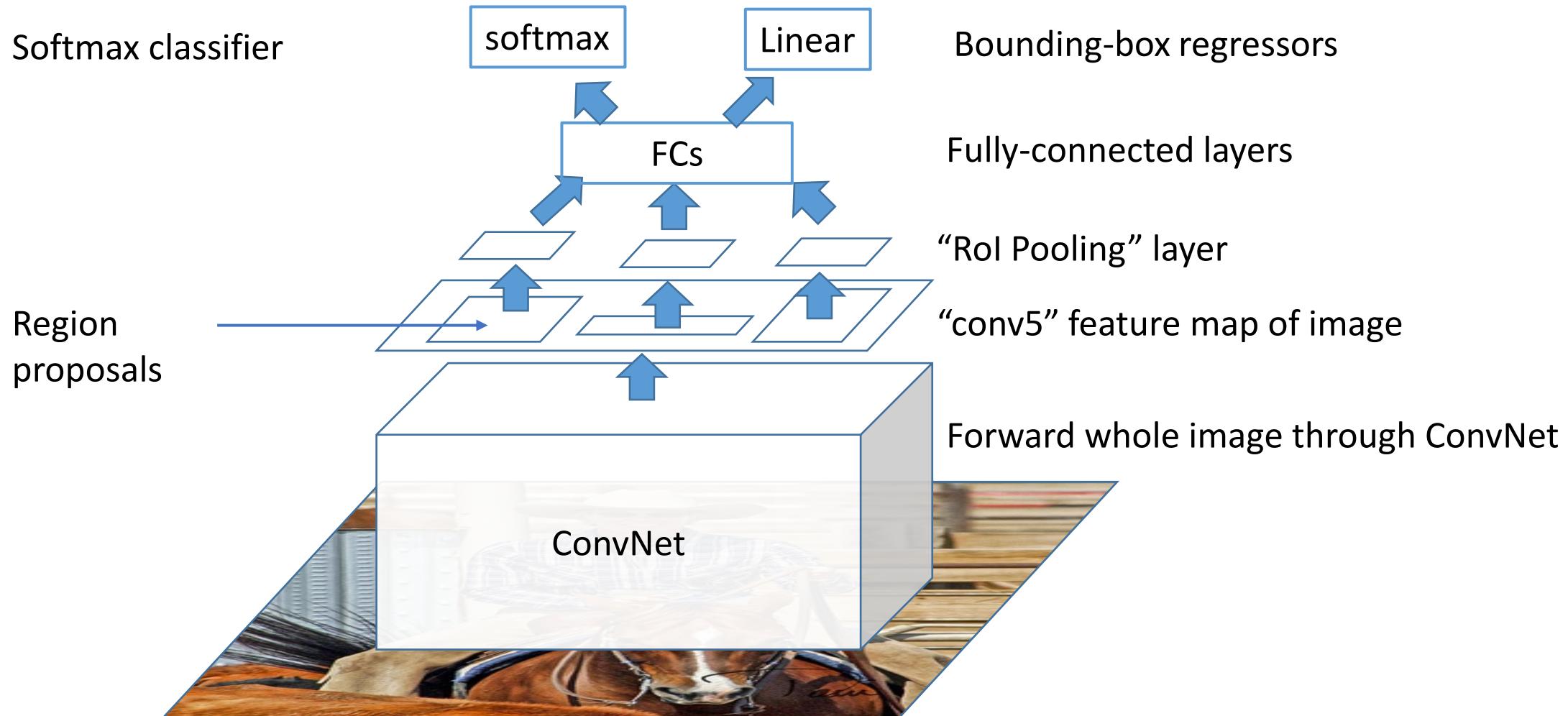
- Solution
 - Why not feed the whole image into CNN only once! Then crop features instead of image itself



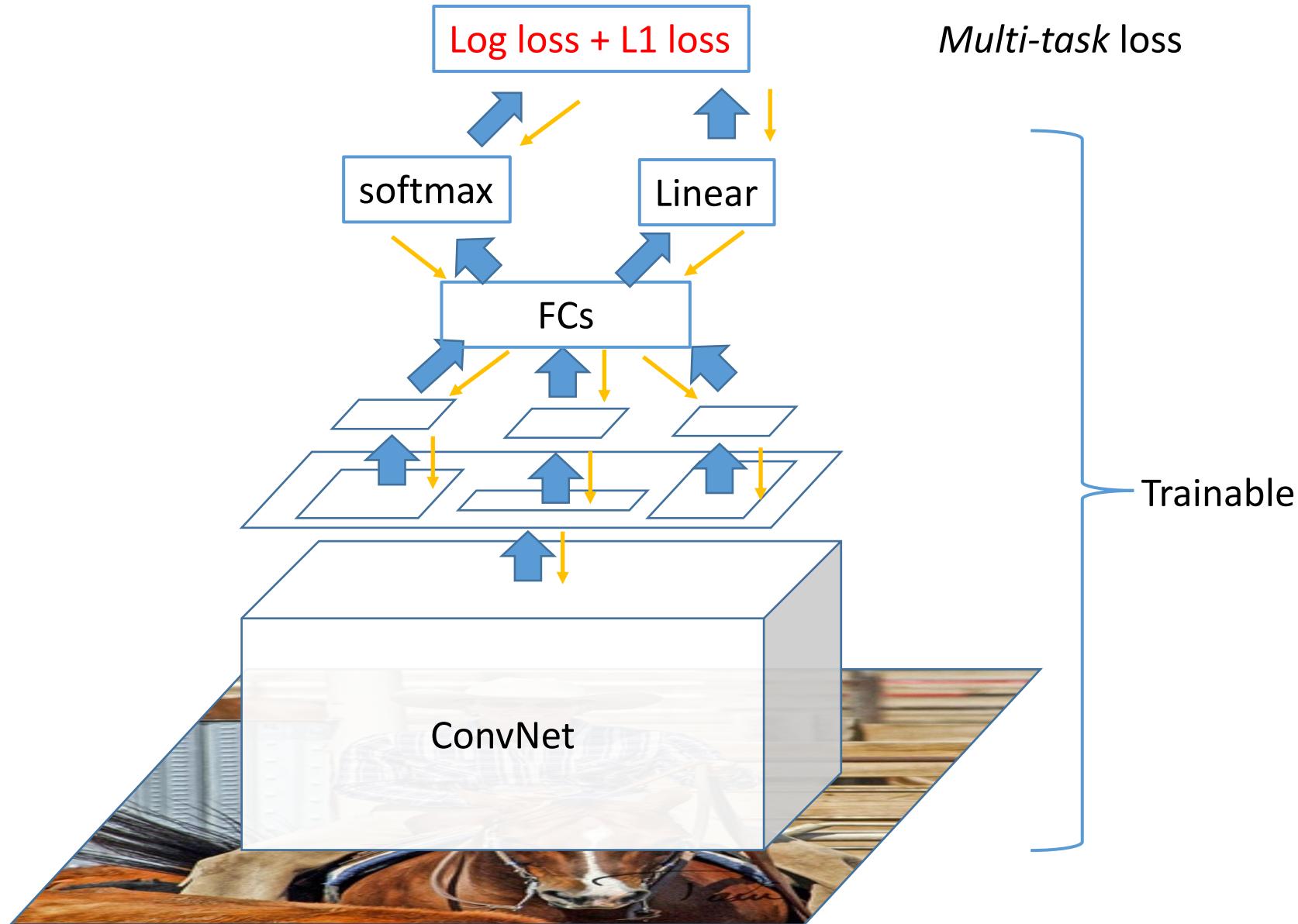
- For each ROI, network predicts probabilities for $C + 1$ classes (class 0 is background) and four bounding box offsets for C classes

Fast R-CNN (Girshick ICCV 2015)

Rather than using post-hoc bounding-box regressors, bounding-box regression is implemented as an additional linear layer in the network



Fast R-CNN (Girshick ICCV 2015)



<https://arxiv.org/pdf/1504.08083.pdf>

Fast R-CNN results

	Fast R-CNN	R-CNN
Train time (h)	9.5	84
- Speedup	8.8x	
Test time / image	0.32s	47.0s
- Test speedup	146x	
mAP	66.9%	66.0% (vs. 53.7% for AlexNet)

Timings exclude object proposal time, which is equal for all methods.

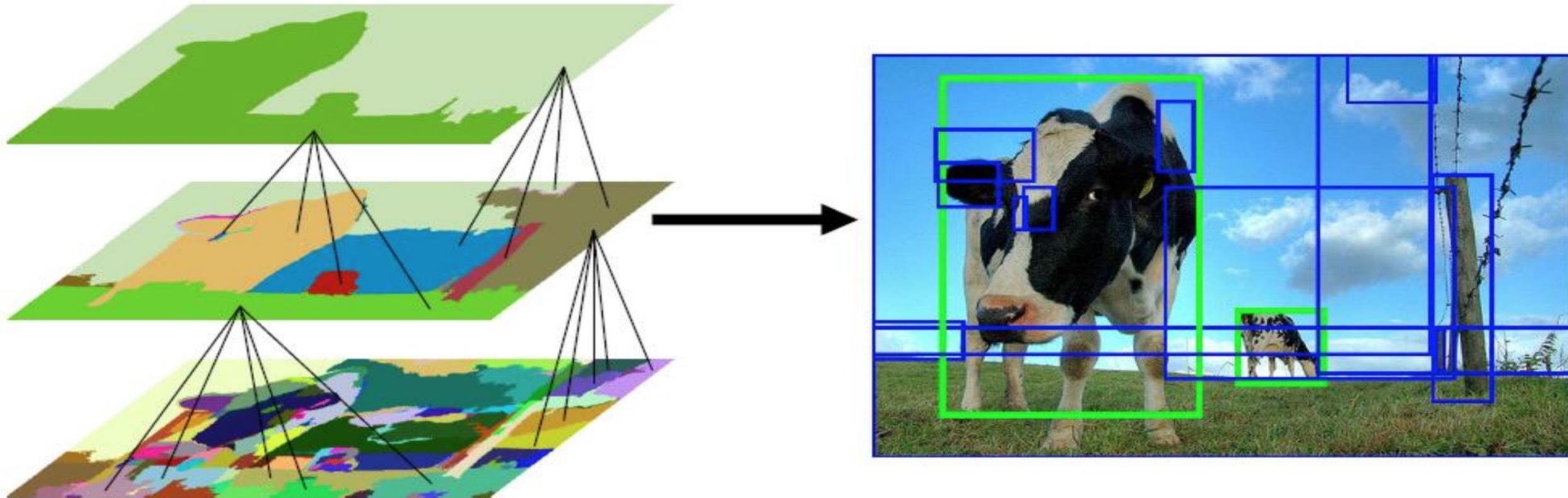
All methods use VGG16.

Fast R-CNN (Girshick ICCV 2015)

- What could be the problems?

Fast R-CNN (Girshick ICCV 2015)

- What could be the problems?
 - Why we need the region proposal pre-processing step? That's not “deep learning” at all. Not cool!



Faster R-CNN (Ren et al. NIPS 2015)

- Solution

- Why not generate region proposals using CNN??!

-> RPN

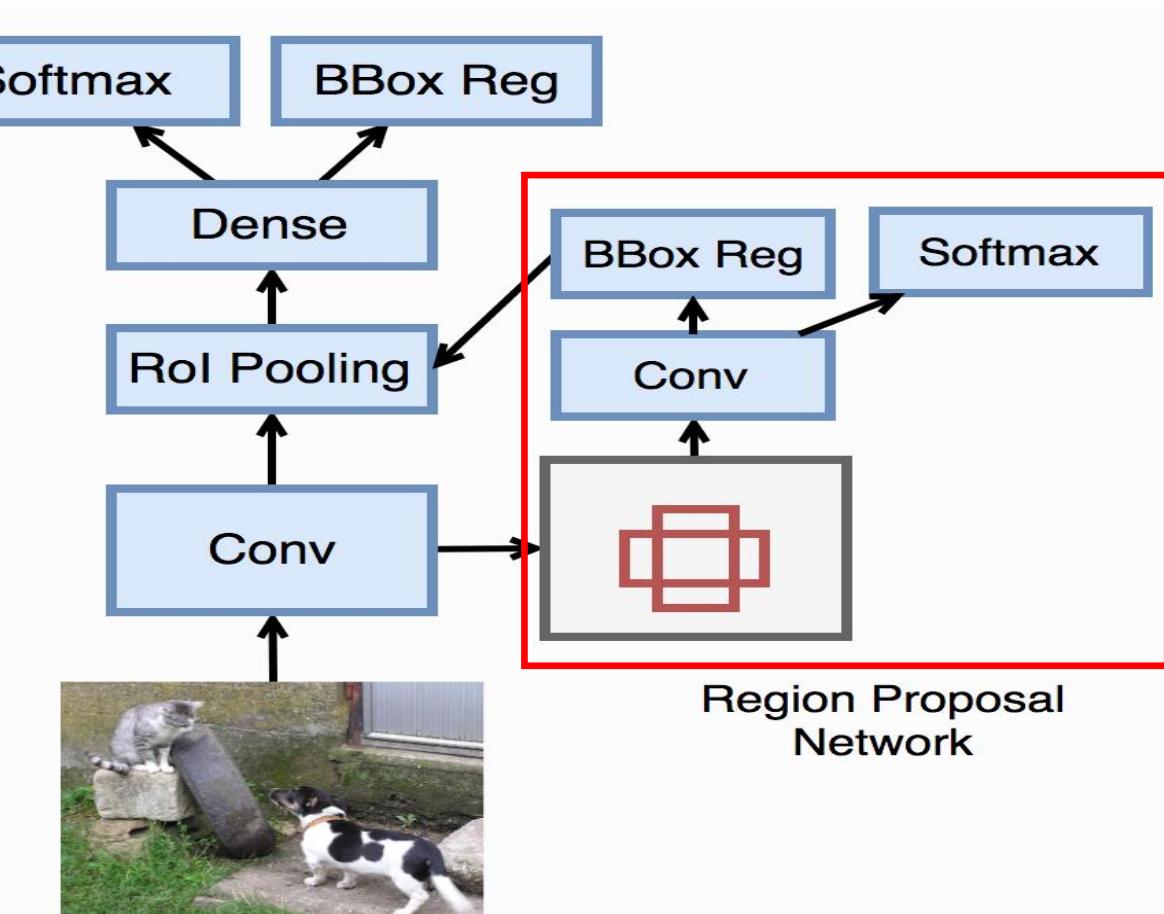
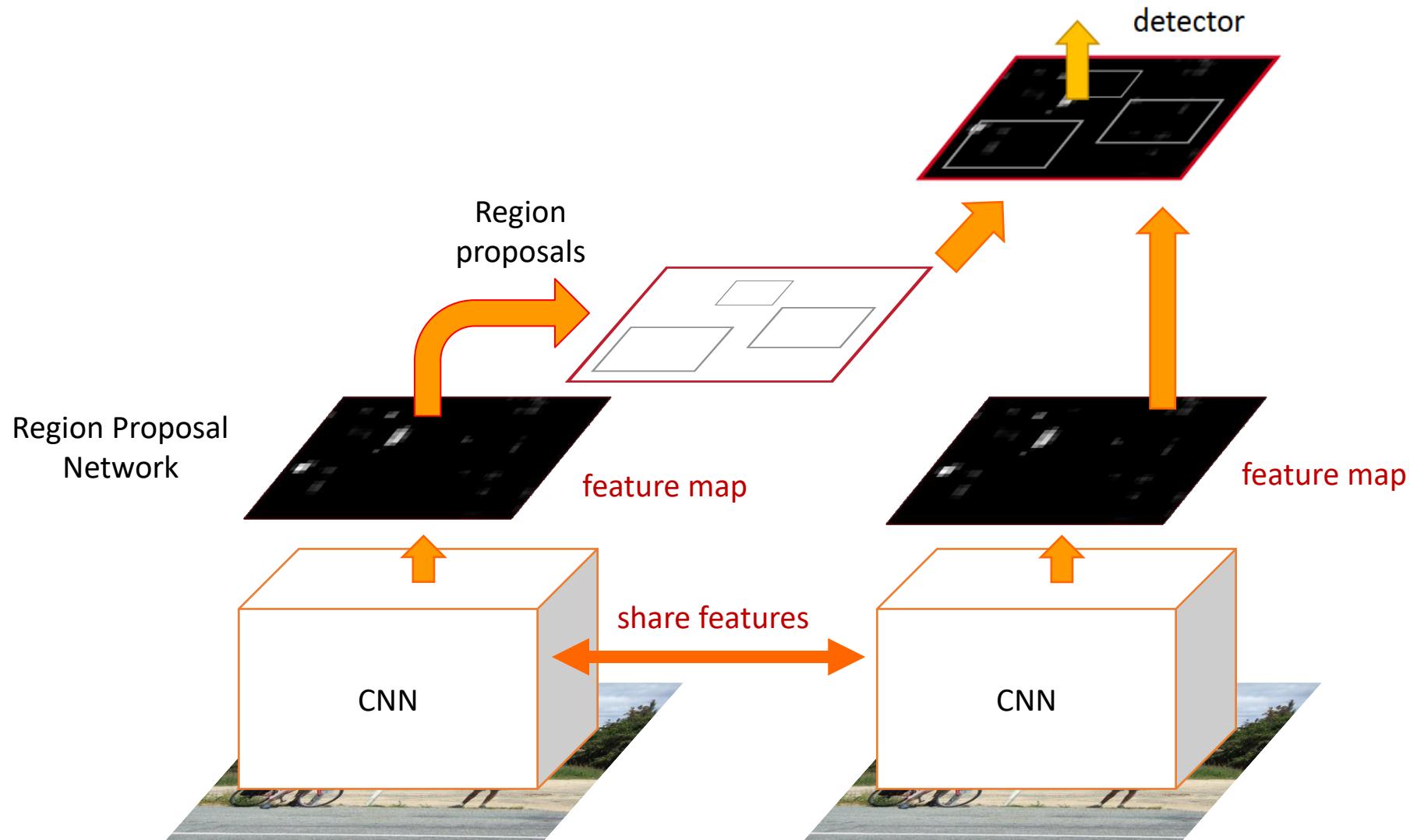


Image credit:

http://zh.gluon.ai/chapter_computer-vision/object-detection.html

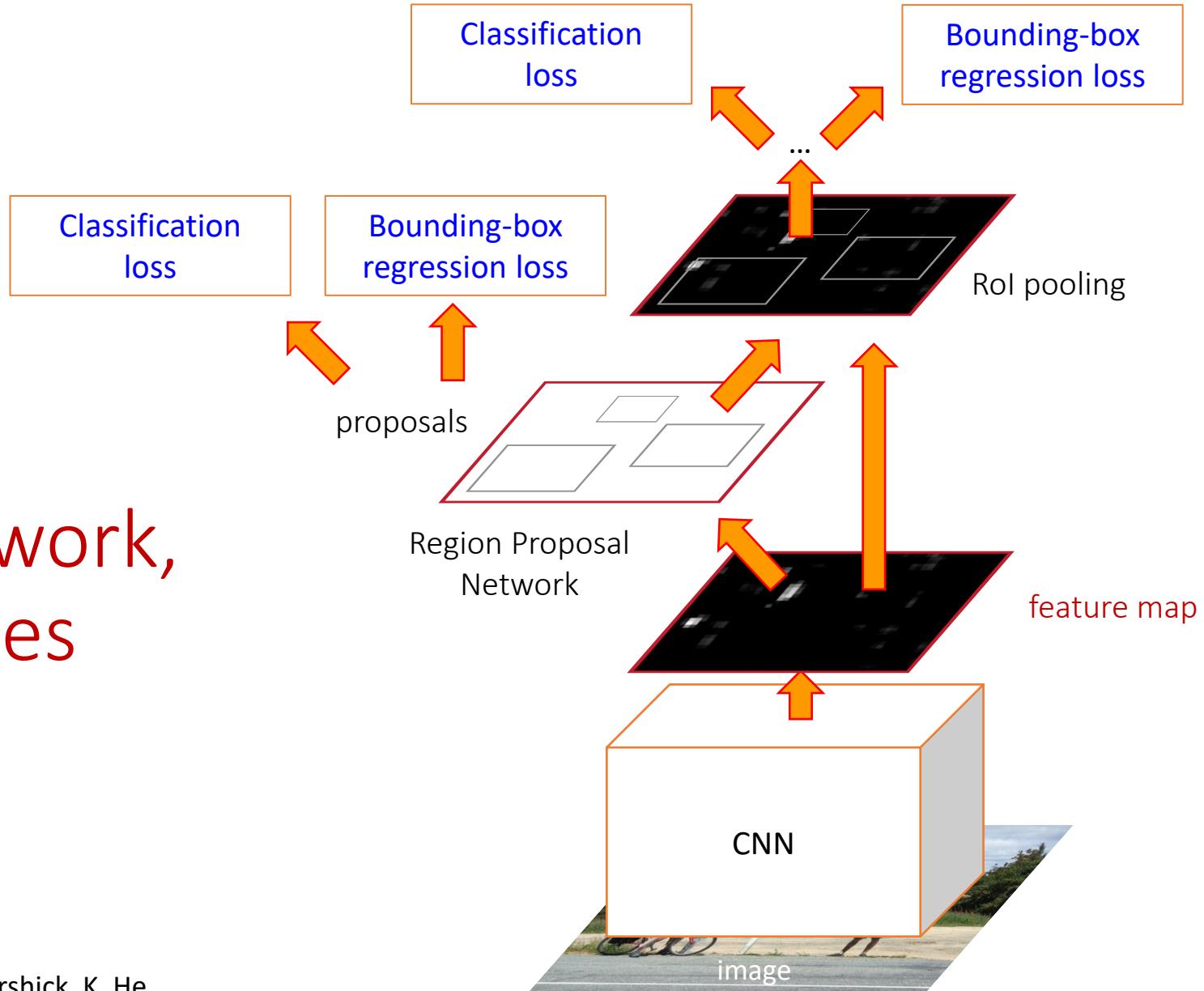
Faster R-CNN (Ren et al. NIPS 2015)



S. Ren, K. He, R. Girshick, and J. Sun, [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks](#), NIPS 2015

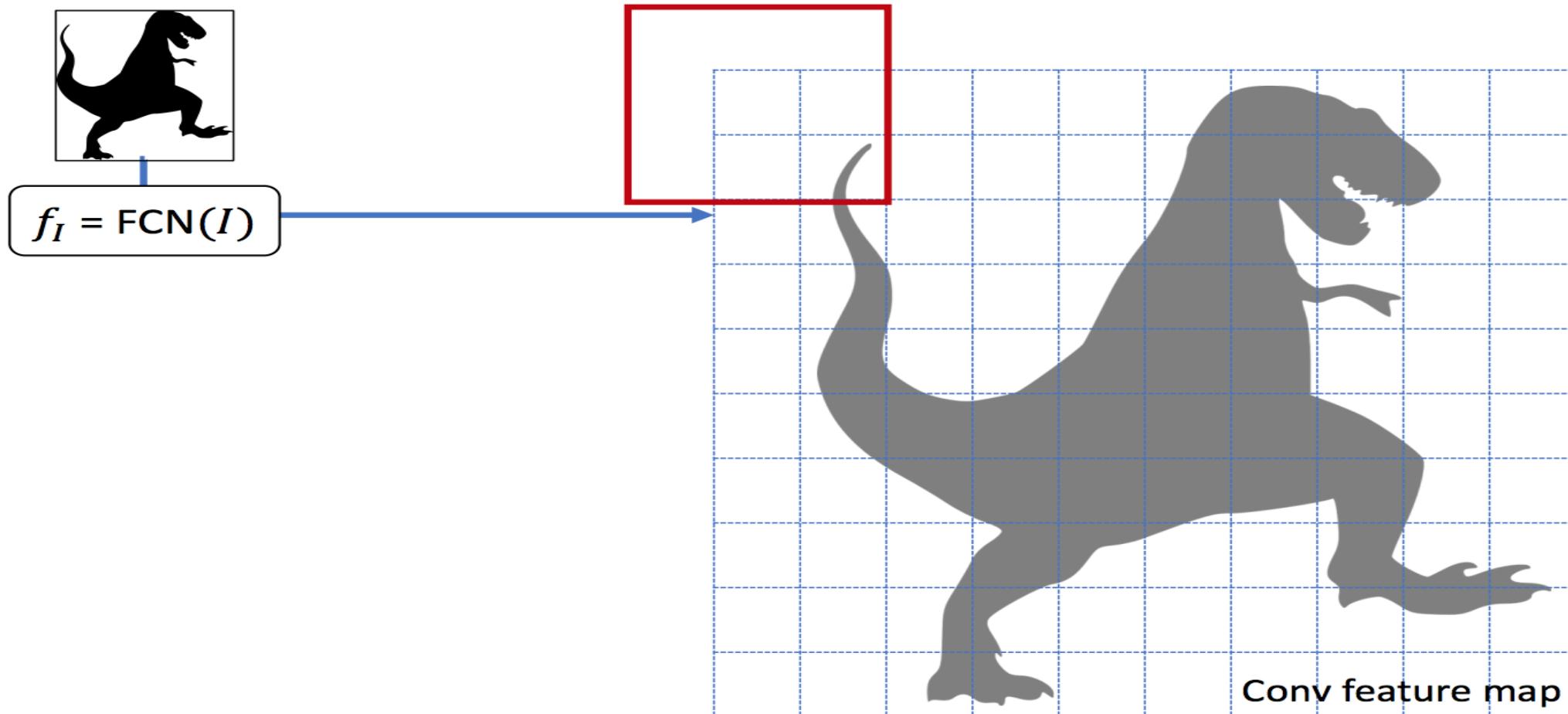
Faster R-CNN (Ren et al. NIPS 2015)

One network,
four losses



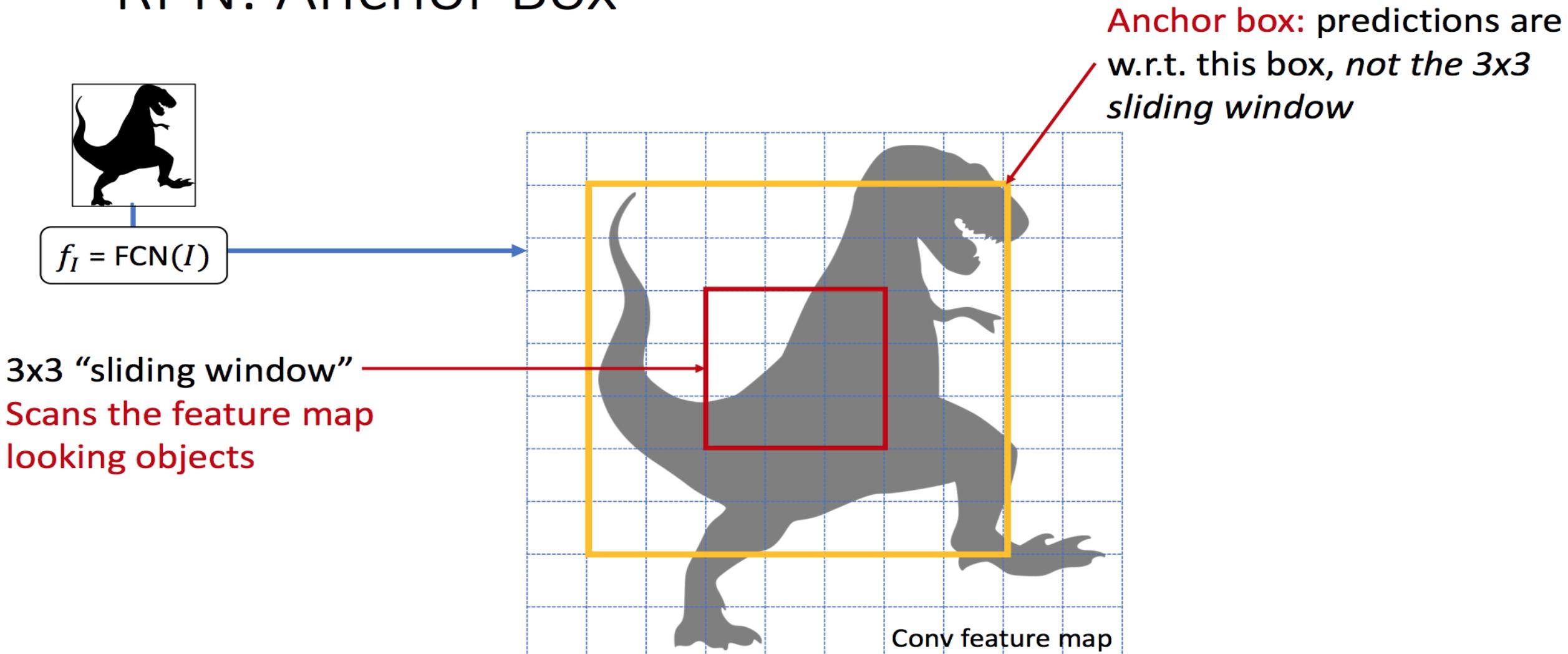
Faster R-CNN (Ren et al. NIPS 2015)

RPN: Region Proposal Network



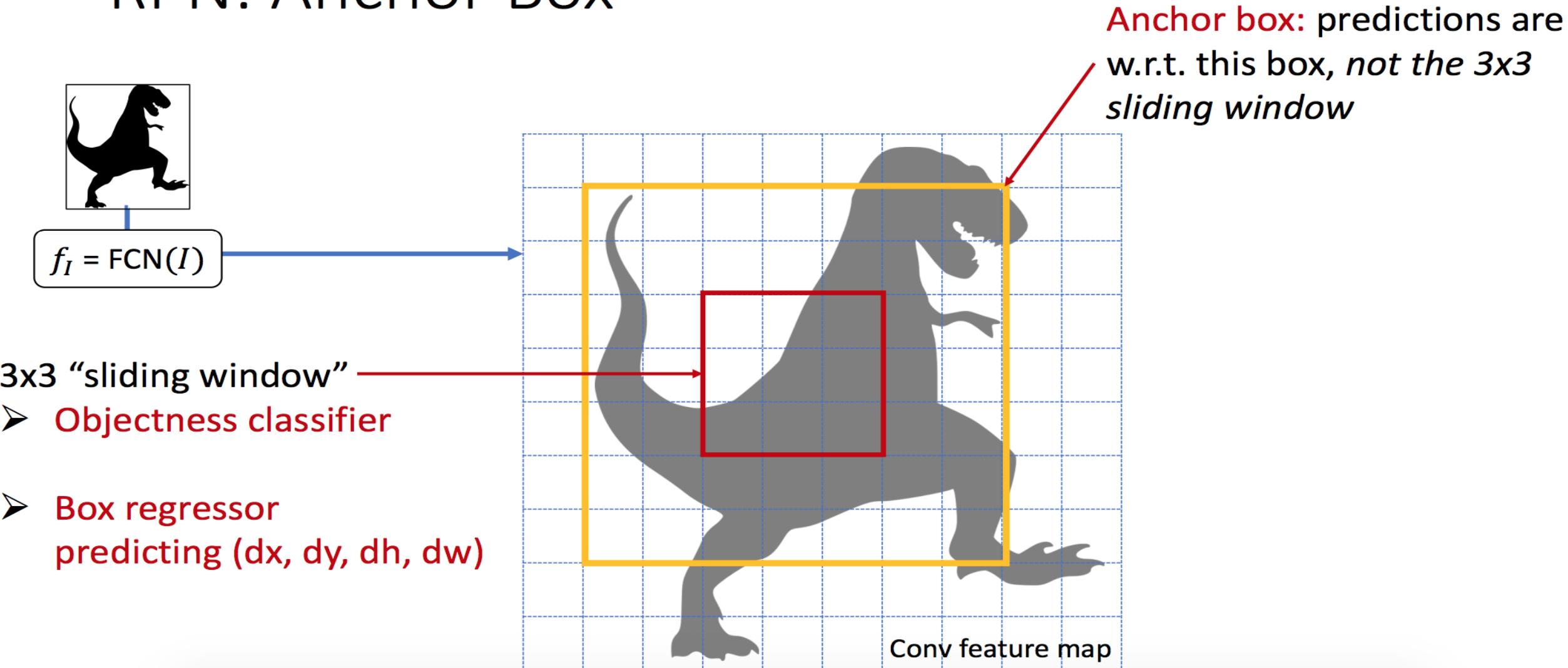
Faster R-CNN (Ren et al. NIPS 2015)

RPN: Anchor Box



Faster R-CNN (Ren et al. NIPS 2015)

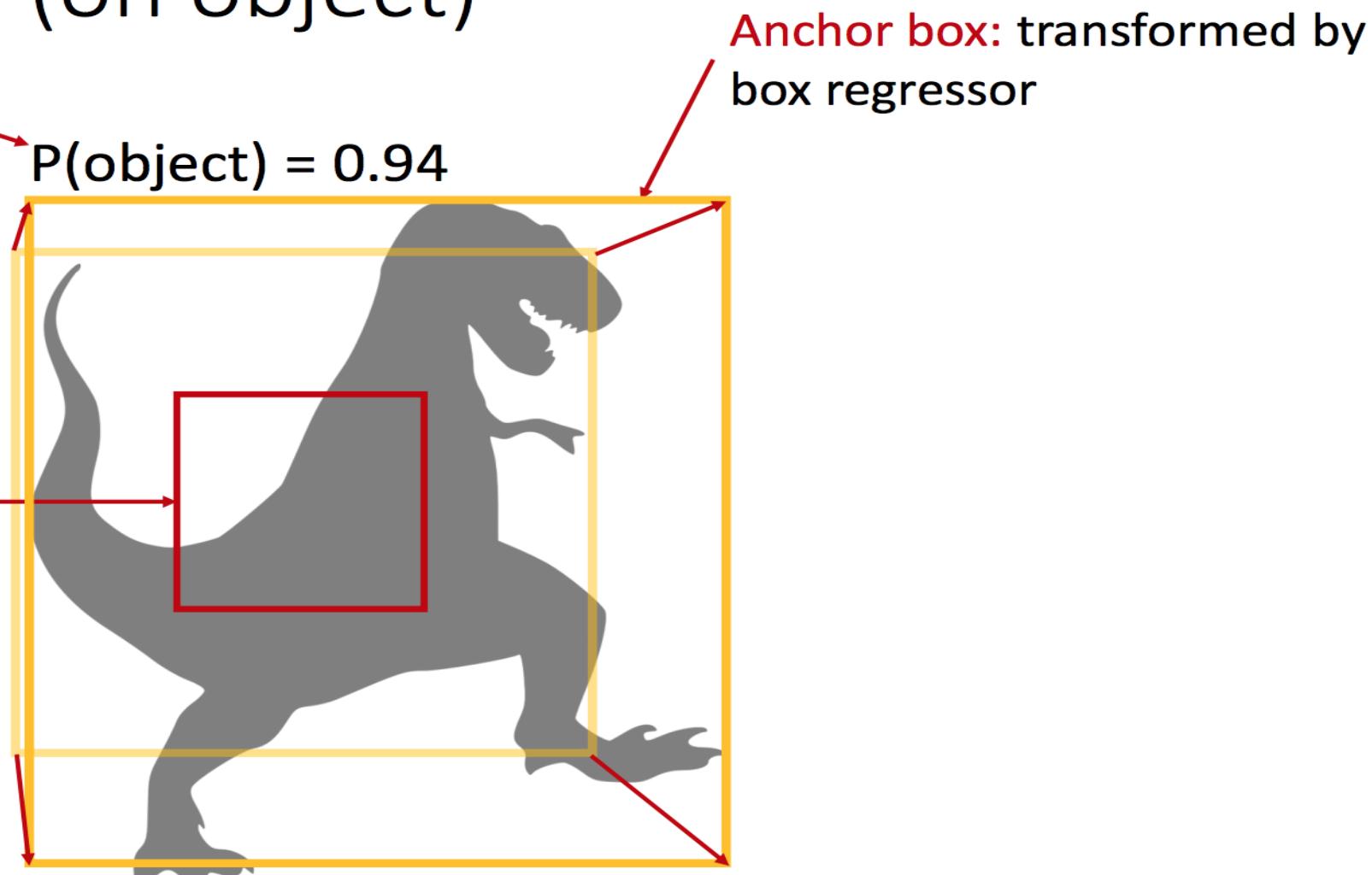
RPN: Anchor Box



Faster R-CNN (Ren et al. NIPS 2015)

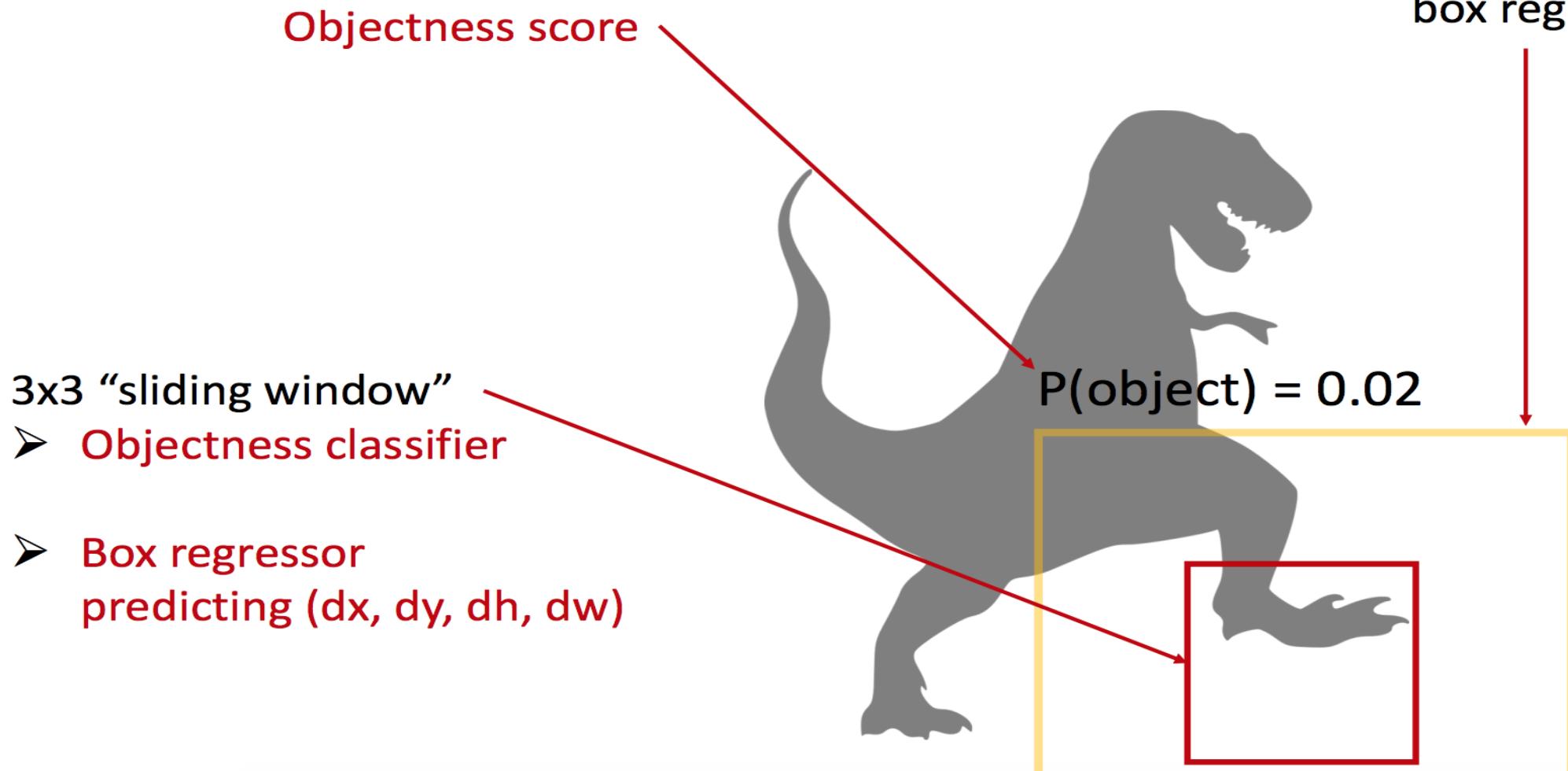
RPN: Prediction (on object)

- Objectness score
- 3x3 “sliding window”
 - Objectness classifier
 - Box regressor predicting (dx, dy, dh, dw)



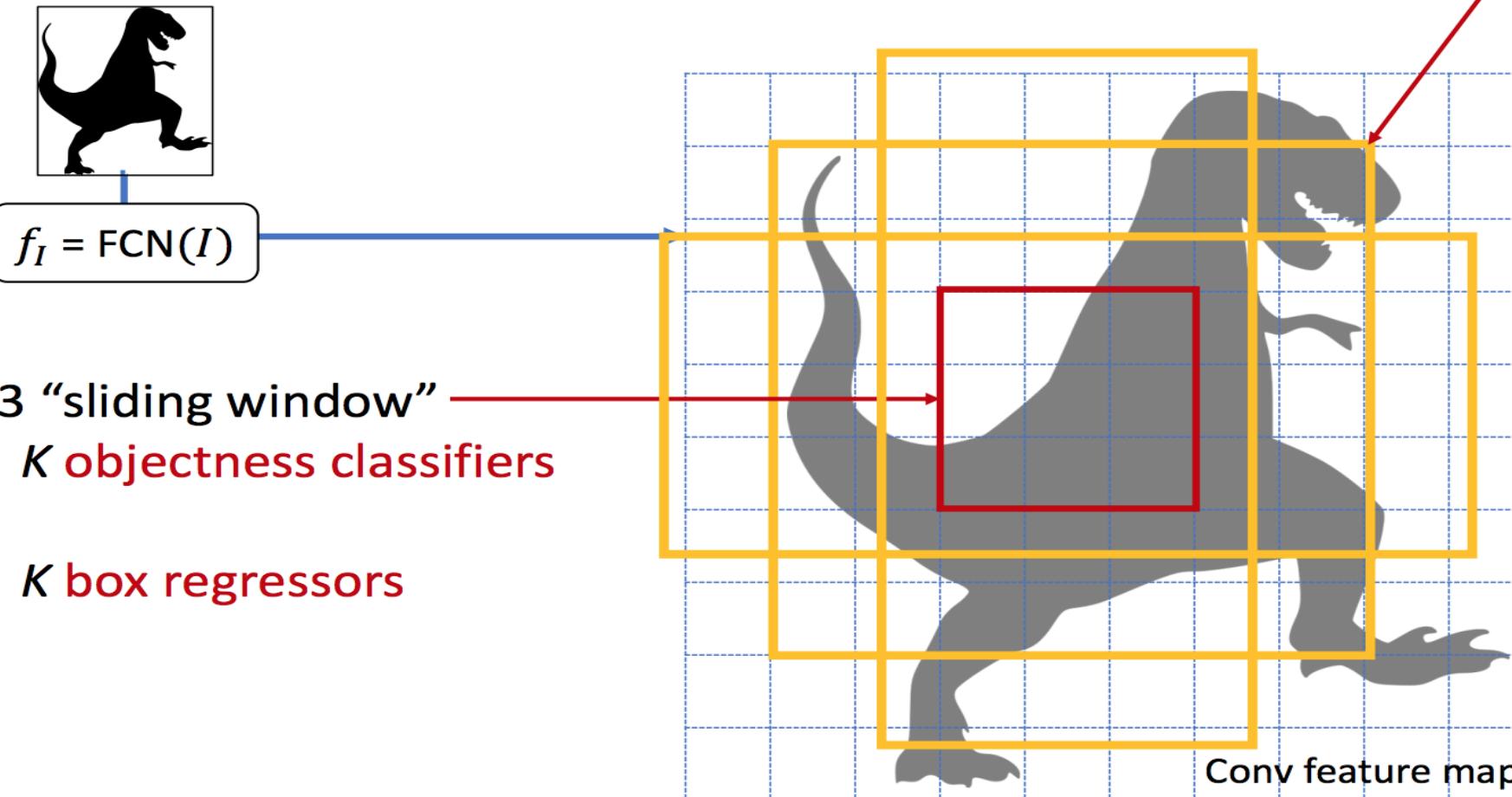
Faster R-CNN (Ren et al. NIPS 2015)

RPN: Prediction (off object)



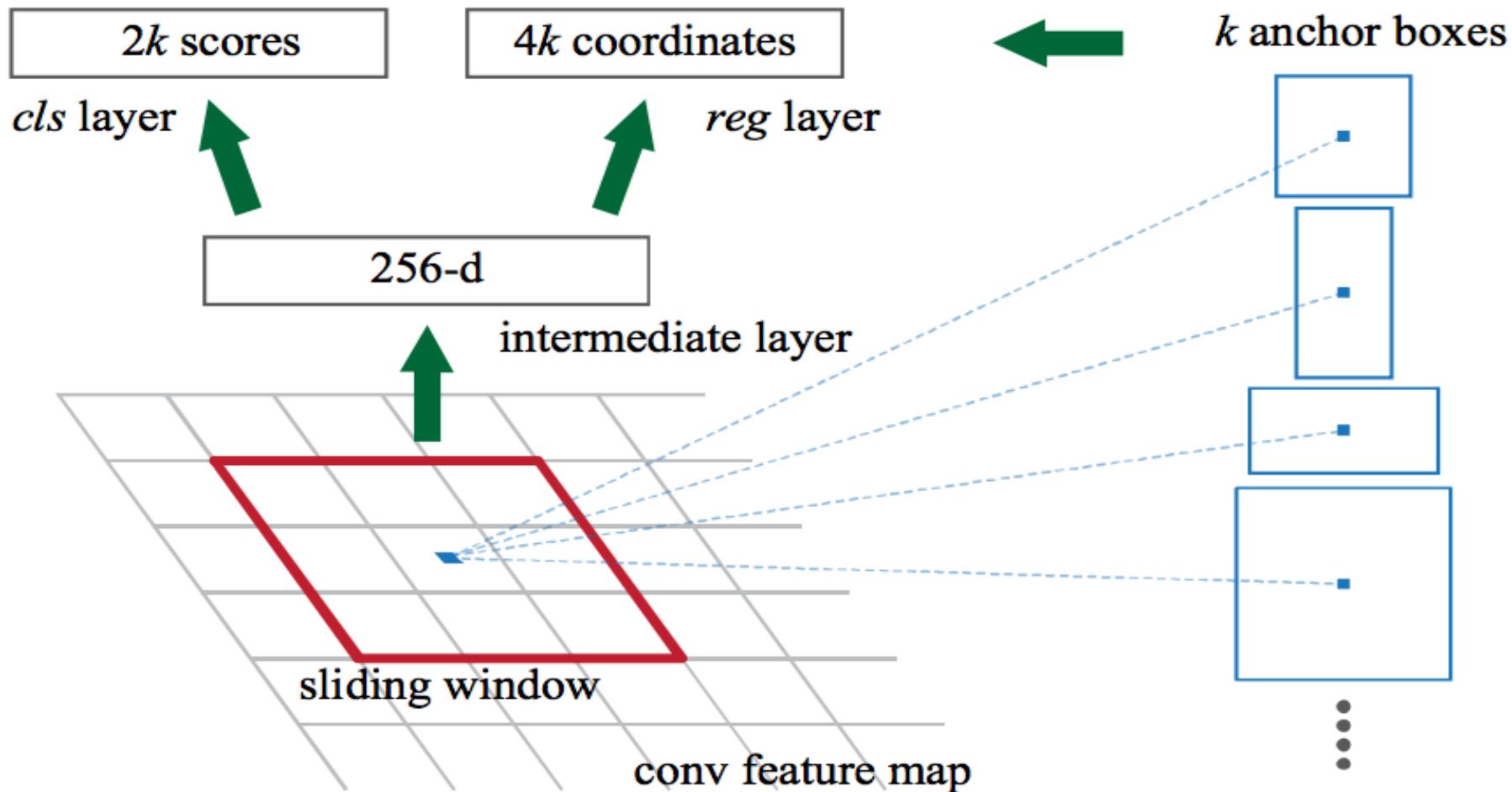
Faster R-CNN (Ren et al. NIPS 2015)

RPN: Multiple Anchors



Faster R-CNN (Ren et al. NIPS 2015)

Region proposal network



Faster R-CNN (Ren et al. NIPS 2015)

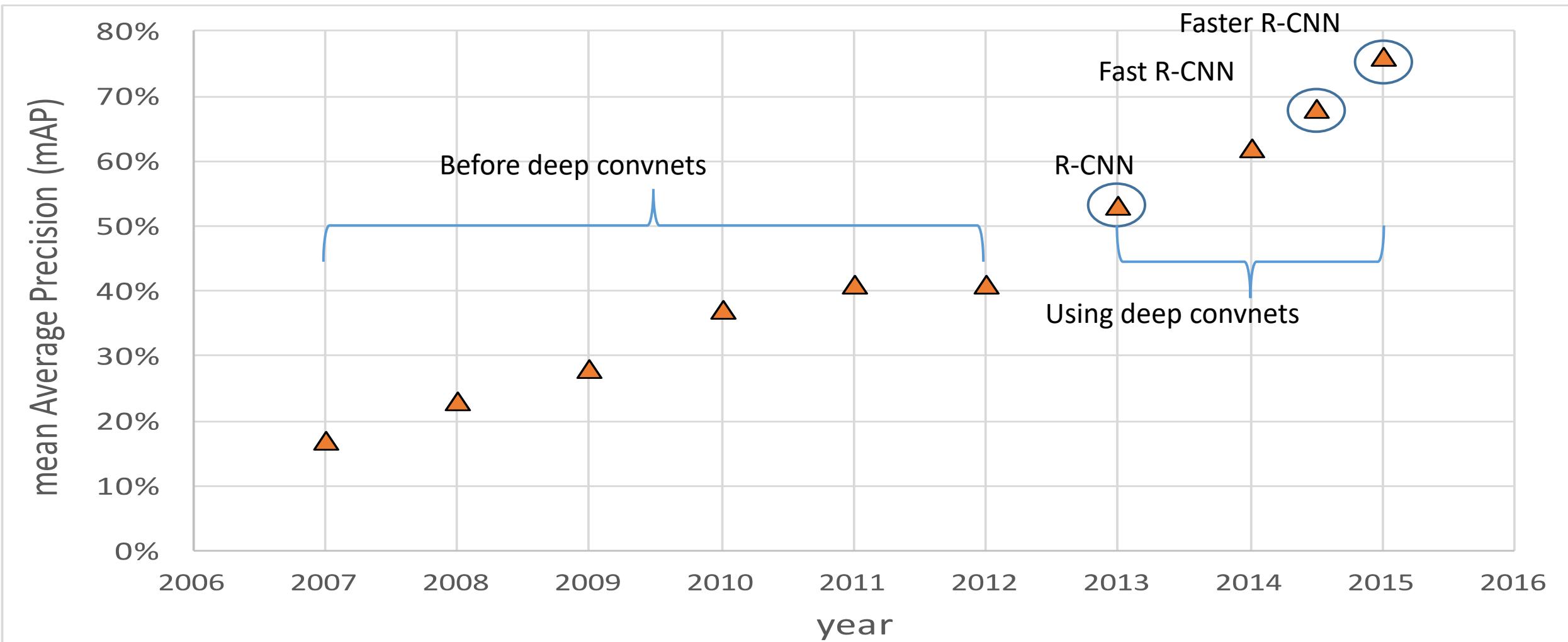
Faster R-CNN results

system	time	07 data	07+12 data
R-CNN	~50s	66.0	-
Fast R-CNN	~2s	66.9	70.0
Faster R-CNN	198ms	69.9	73.2

detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet

Faster R-CNN (Ren et al. NIPS 2015)

Progress on PASCAL VOC database



Faster R-CNN (Ren et al. NIPS 2015)

- What could be the problems

Faster R-CNN (Ren et al. NIPS 2015)

- What could be the problems
 - Two-stage detection pipeline is still too slow to apply on real-time images and videos

One-stage detection

- Solution
 - Don't generate object proposals!
 - Consider a tiny subset of the output space by design; directly classify this small set of boxes

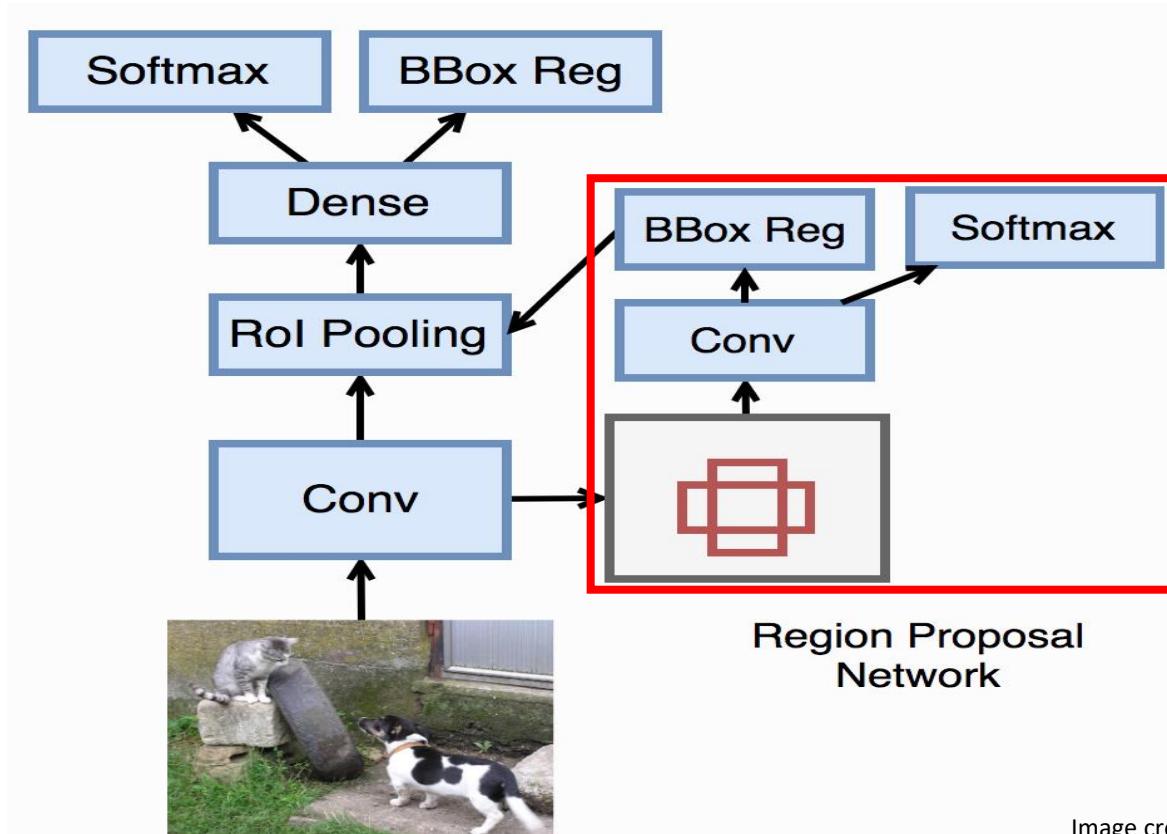
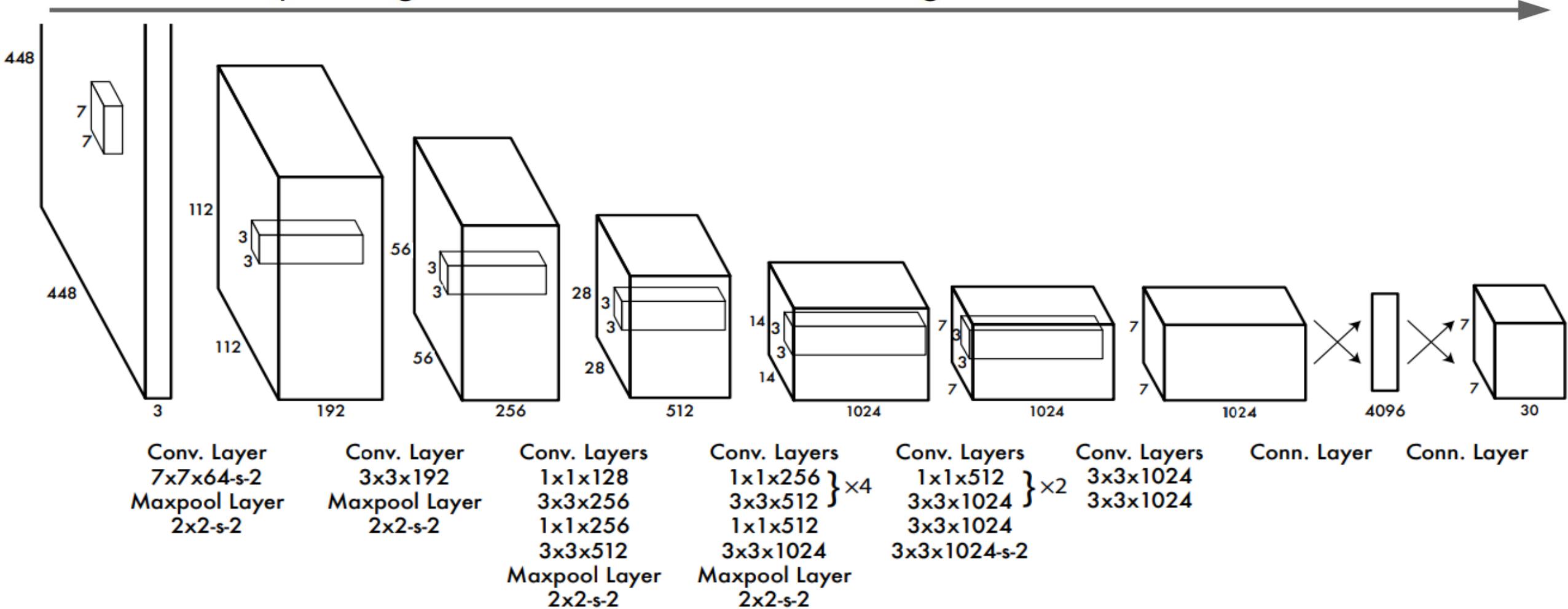


Image credit:
http://zh.gluon.ai/chapter_computer-vision/object-detection.html

You Only Look Once (YOLO)

Go from input image to tensor of scores with one big convolutional network!

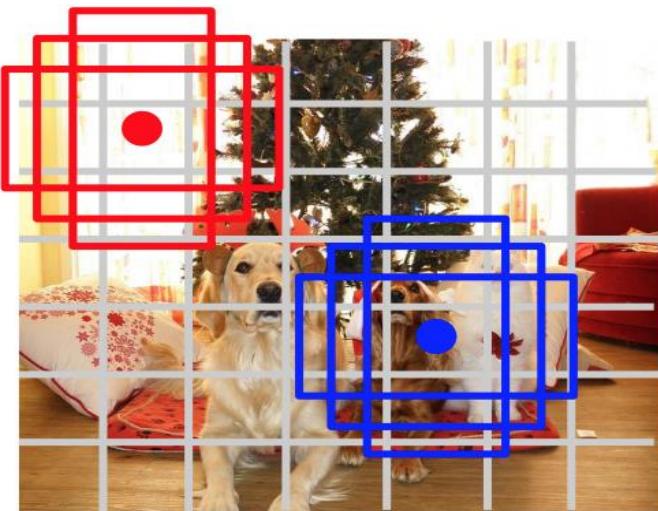


You Only Look Once (YOLO)

Go from input image to tensor of scores with one big convolutional network!



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

Image a set of **base boxes**
centered at each grid cell

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
(dx , dy , dh , dw , confidence)
- Predict scores for each of C classes (including background as a class)

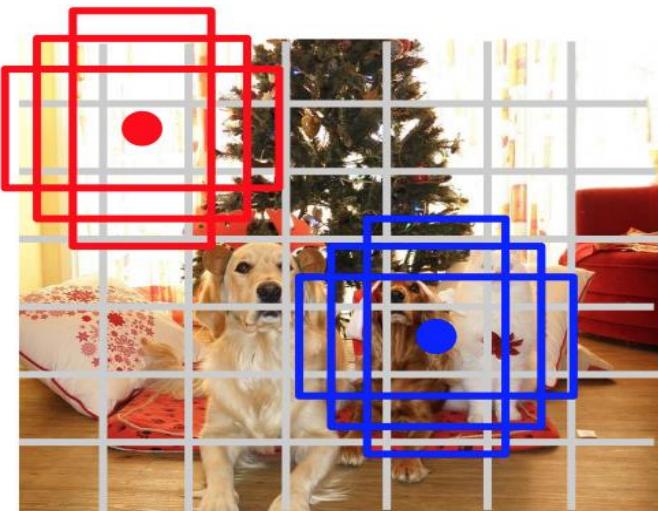
Output:
 $7 \times 7 \times (5 * B + C)$

You Only Look Once (YOLO)

Go from input image to tensor of scores with one big convolutional network!



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

Image a set of **base boxes**
centered at each grid cell

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
(dx , dy , dh , dw , confidence)
- Predict scores for each of C classes (including background as a class)

Output:
 $7 \times 7 \times (5 * B + C)$

$B = 2$ in experiments

$C = 20$ in PASCAL VOC

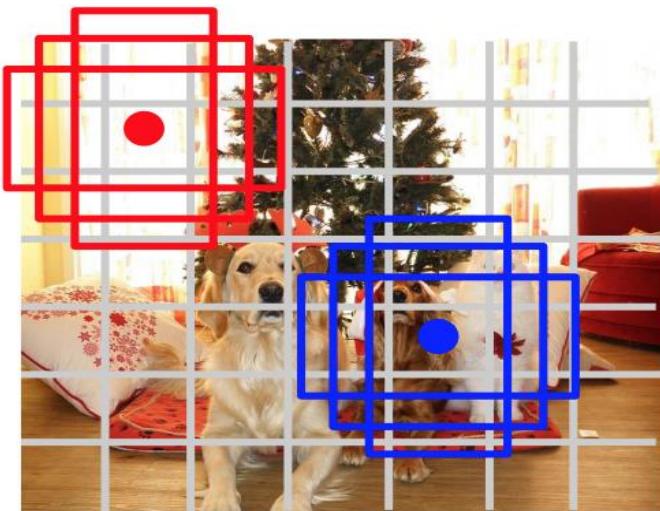
Final prediction = $7 \times 7 \times 30$ tensor.

You Only Look Once (YOLO)

Go from input image to tensor of scores with one big convolutional network!



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

Image a set of **base boxes**
centered at each grid cell

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
(dx , dy , dh , dw , confidence)
- Predict scores for each of C classes (including background as a class)

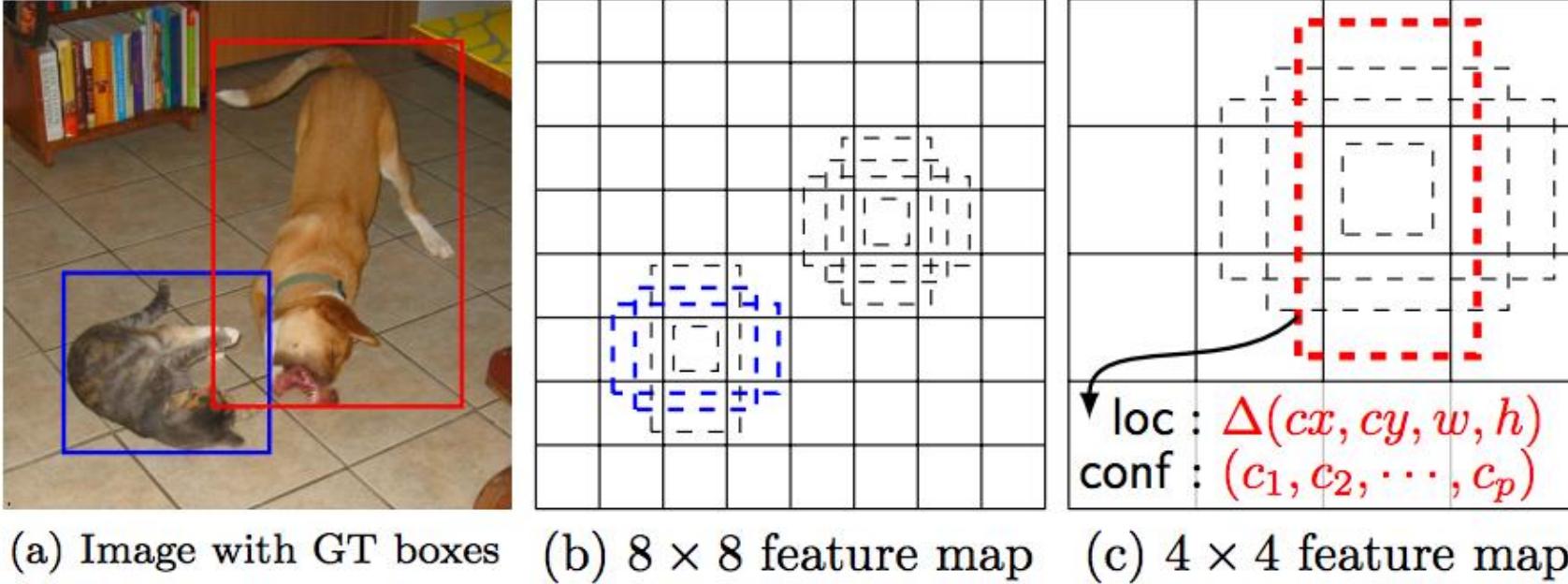
Output:
 $7 \times 7 \times (5 * B + C)$

•Very efficient but lower accuracy

You Only Look Once (YOLO)

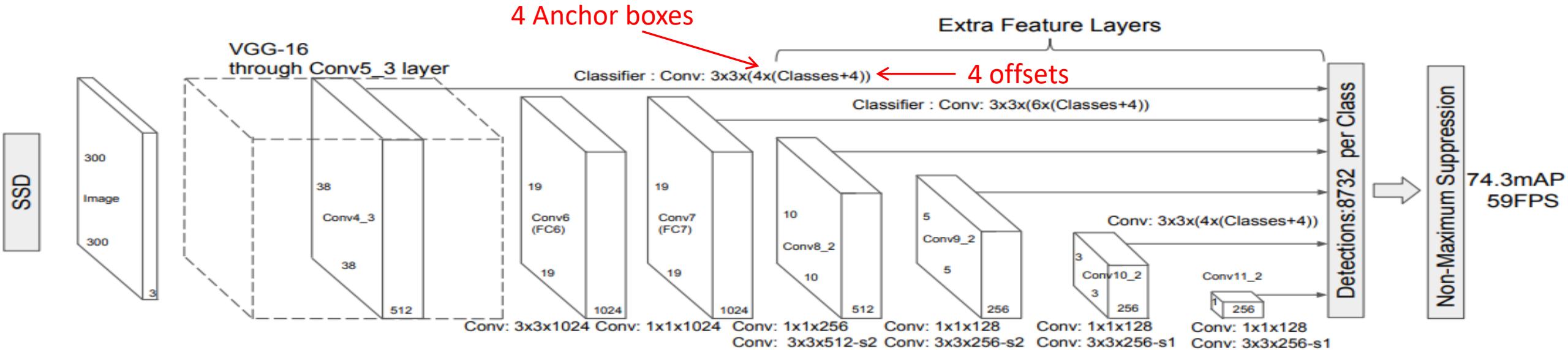
- Each grid cell predicts only two boxes and can only have one class – this limits the number of nearby objects that can be predicted
- Localization accuracy suffers compared to Fast(er) R-CNN due to coarser features, errors on small boxes
- 7x speedup over Faster R-CNN (45-155 FPS vs. 7-18 FPS)

SSD: Single Shot MultiBox Detector



- Similarly to YOLO, predict bounding boxes directly from conv maps
- Unlike YOLO, do not use FC layers and predict different size boxes from conv maps at different resolutions
- Similarly to RPN, use anchors

SSD: Single Shot MultiBox Detector



- Run a small 3×3 sized convolutional kernel to predict the bounding boxes and classification probability.
- SSD also uses anchor boxes at various aspect ratio similar to Faster-RCNN and learns the off-set rather than learning the box.
- In order to handle the scale, SSD predicts bounding boxes after multiple convolutional layers.

SSD: Single Shot MultiBox Detector



One-stage detection

- What could be the problems?

One-stage detection

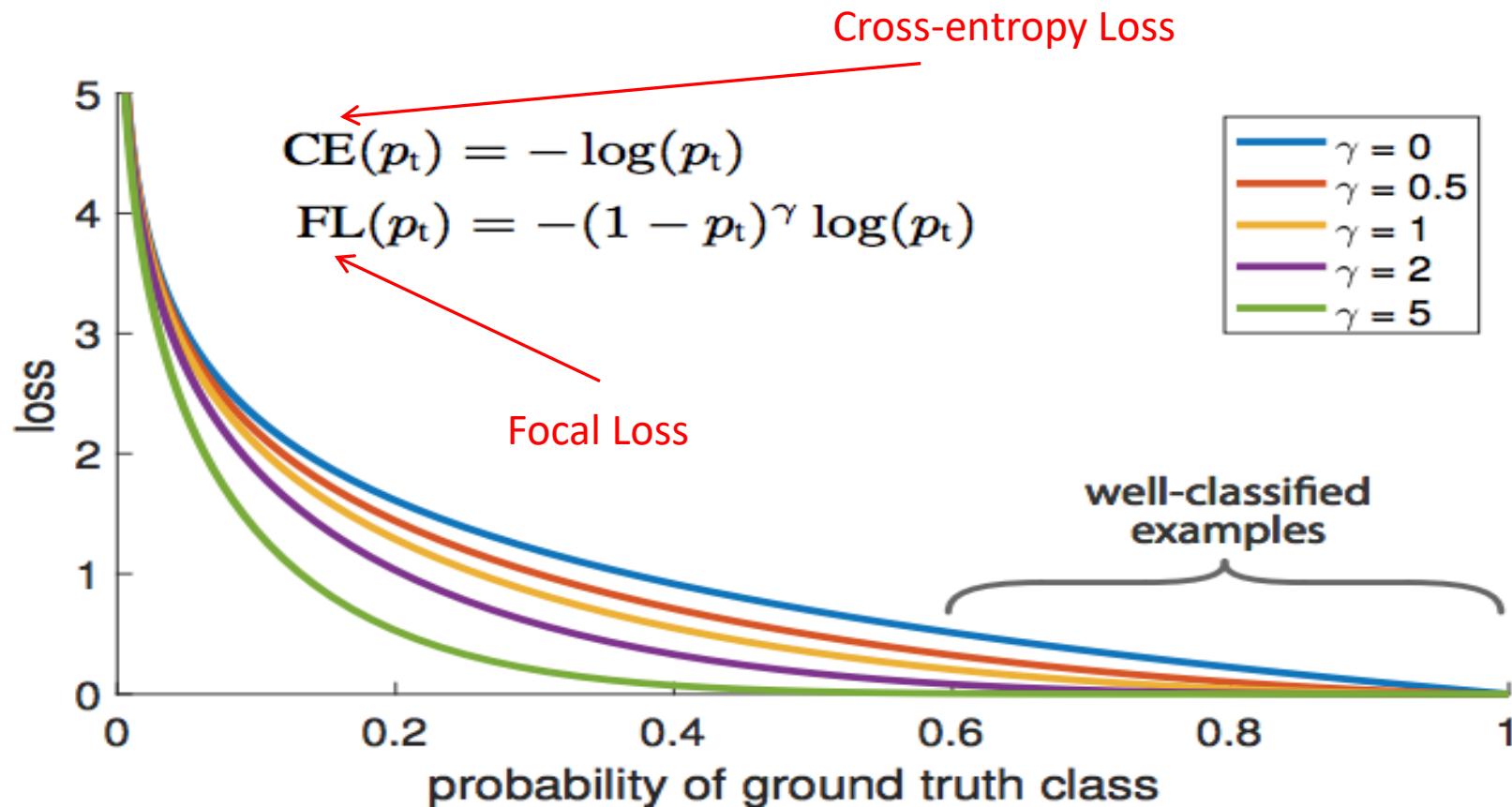
- What could be the problems?
 - The extreme foreground-background class imbalance
-> we have a lot more negative examples.

One-stage detection

- What could be the problems?
 - The extreme foreground-background class imbalance
-> we have a lot more negative examples.
 - The vast number of easy negatives overwhelms the detector during training.

RetinaNet (Lin et al. ICCV 2017)

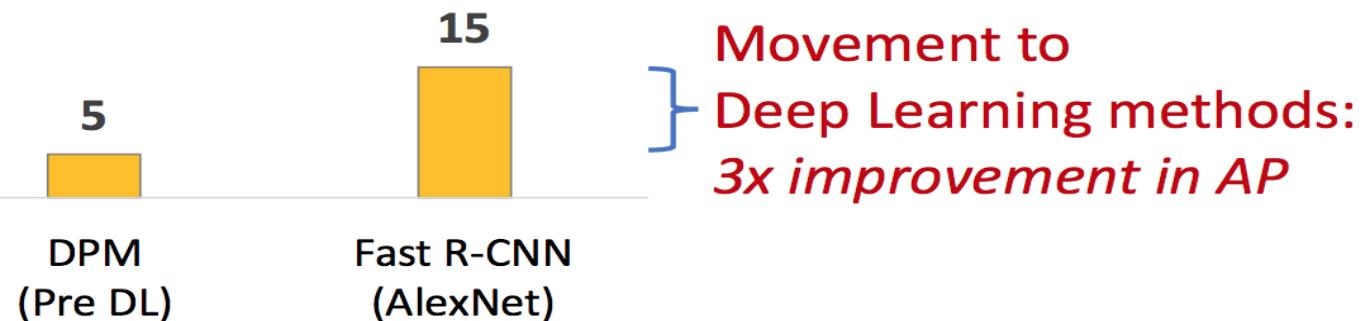
- Solution
 - For easy negative examples, down-weight the loss, so that the gradients from these example have smaller impact to the model



COCO Object Detection Average Precision (%)

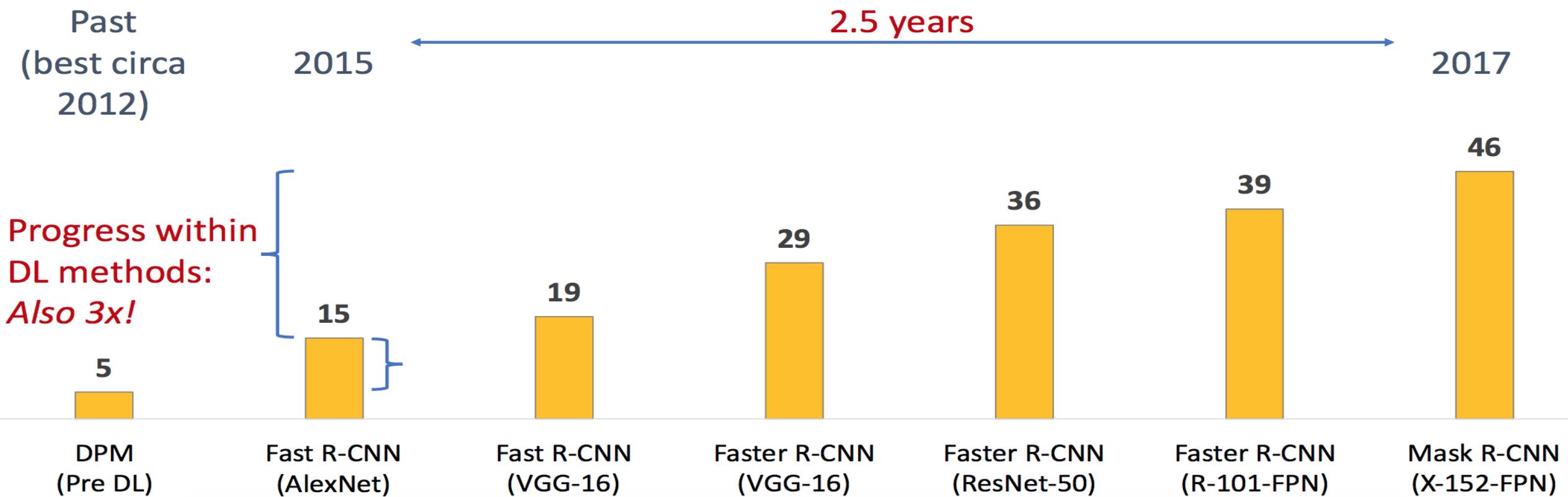
Past
(best circa
2012)

2015



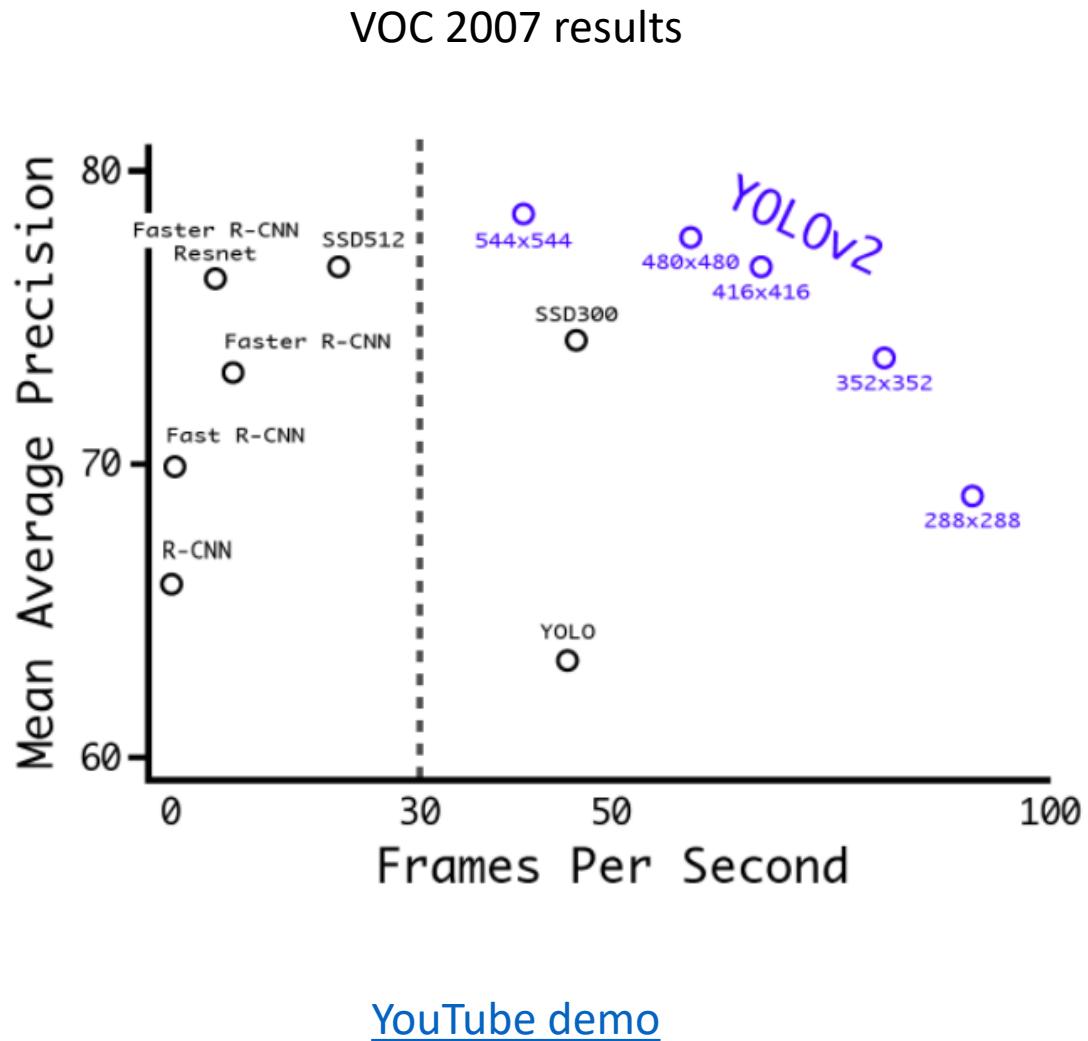
Movement to
Deep Learning methods:
3x improvement in AP

COCO Object Detection Average Precision (%)



YOLO v2

- Remove FC layer, do convolutional prediction with anchor boxes instead
- Increase resolution of input images and conv feature maps
- Improve accuracy using batch normalization and other tricks



YOLO v3

YOLOv3: An Incremental Improvement

Joseph Redmon, Ali Farhadi

University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320×320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP₅₀ in 51 ms on a Titan X, compared to 57.5 AP₅₀ in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at <https://pjreddie.com/yolo/>.

1. Introduction

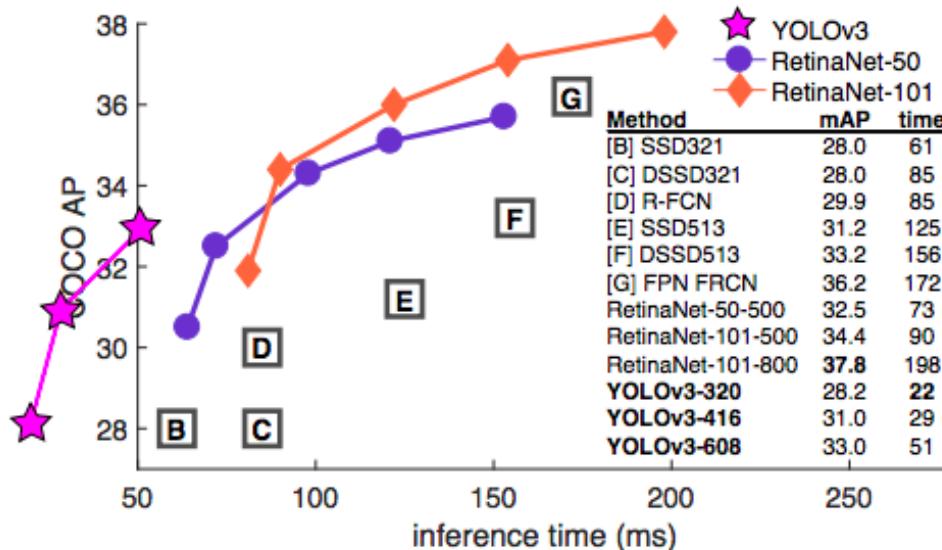
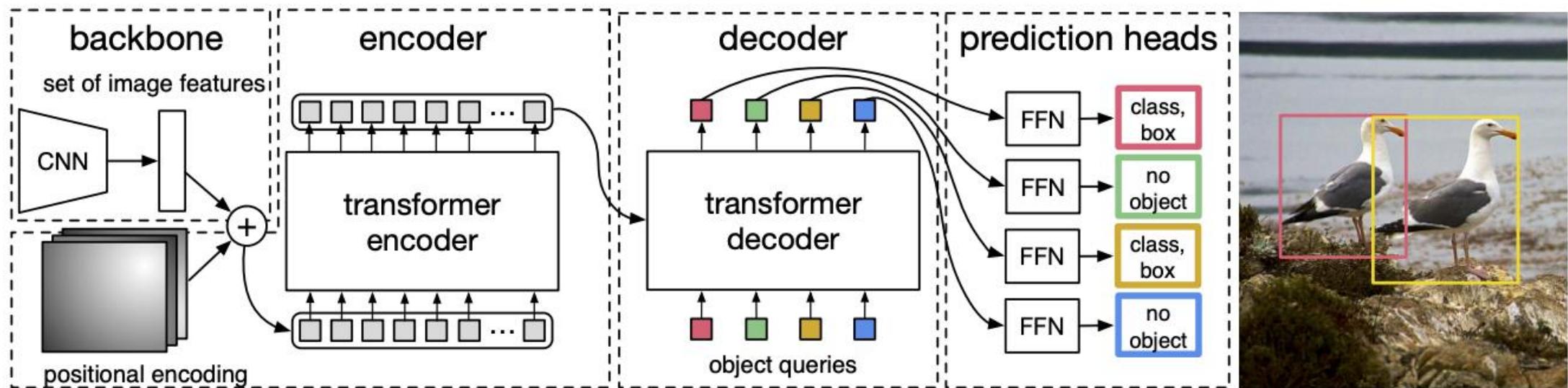
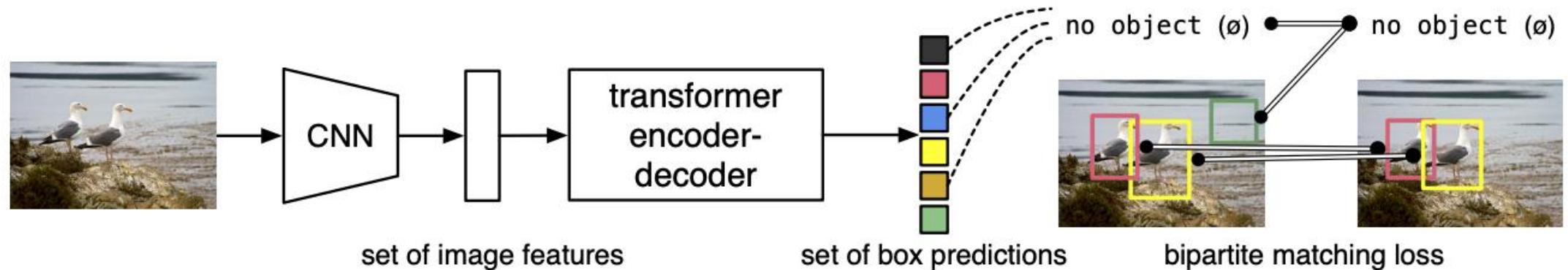


Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

Summary so far

- R-CNN: region proposals + CNN on cropped, resampled regions
- Fast R-CNN: region proposals + RoI pooling on top of a conv feature map
- Faster R-CNN: RPN + RoI pooling
- Next generation of detectors: YOLO, SSD, RetinaNet
 - Direct prediction of BB offsets, class scores on top of conv feature maps
 - Get better context by combining feature maps at multiple resolutions
- Most recent developments: architectures borrowed from dense prediction, transformers

Detection Transformer (DETR)



Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Next class

- Image Segmentation

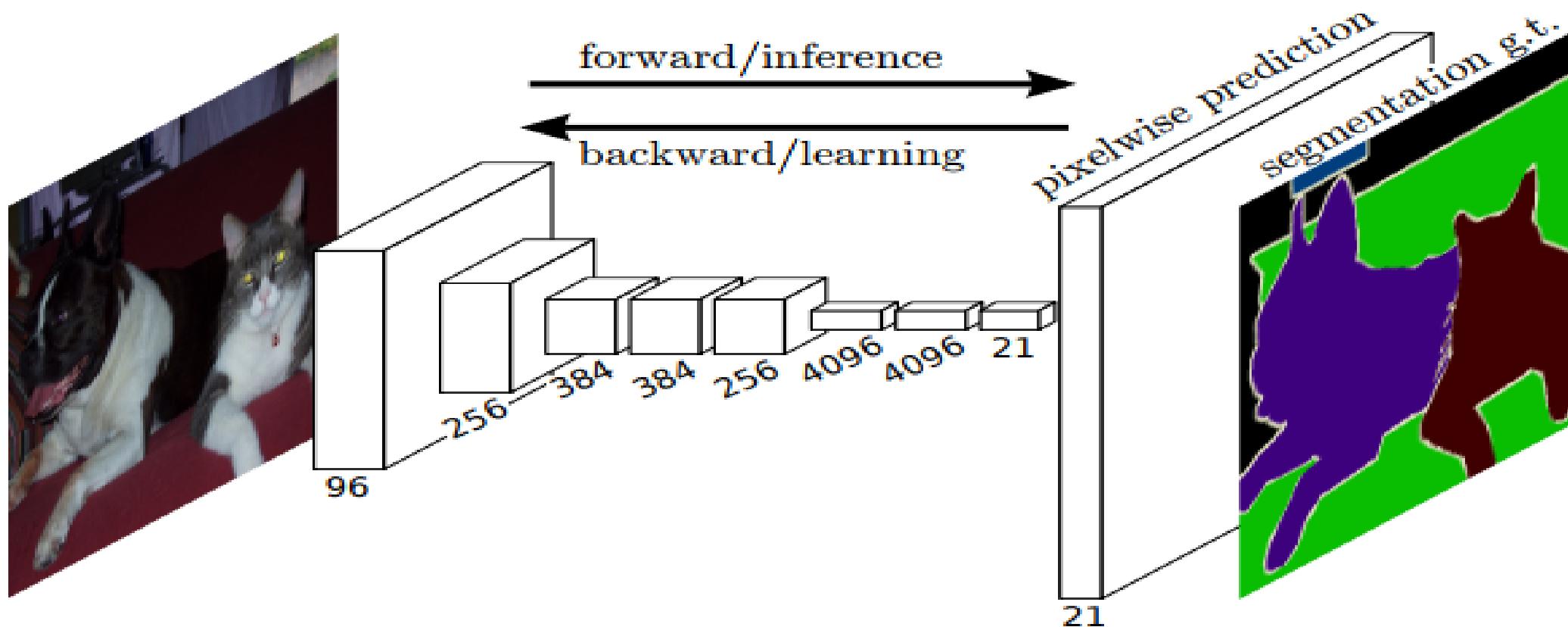
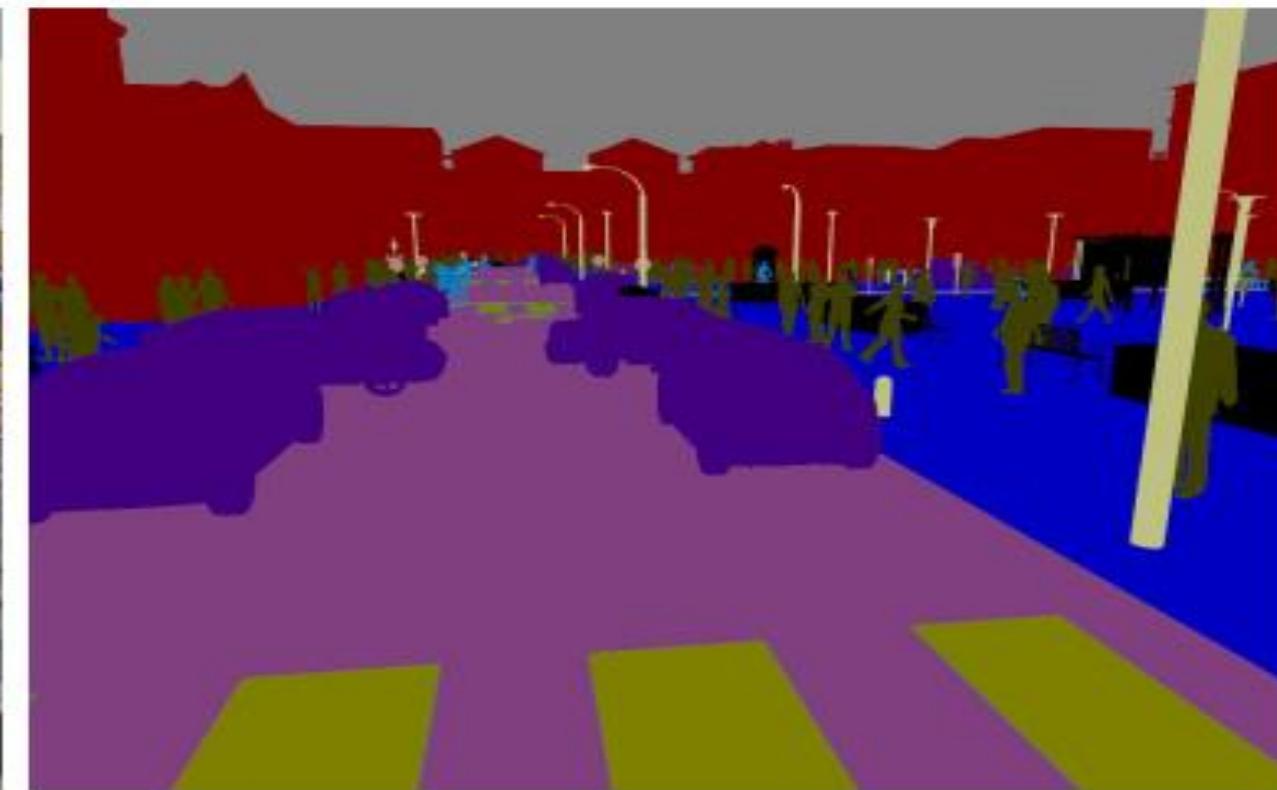


Image Segmentation using Deep Learning



Previous Class: Object bounding box detections

Deep learning based detectors

Two-stage detectors

R-CNN

Fast R-CNN

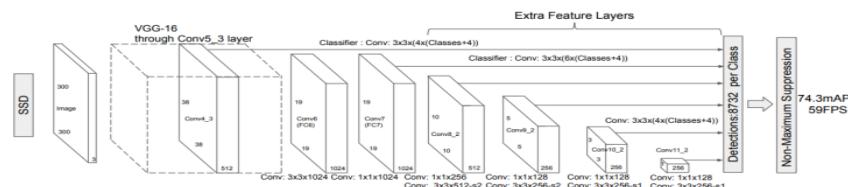
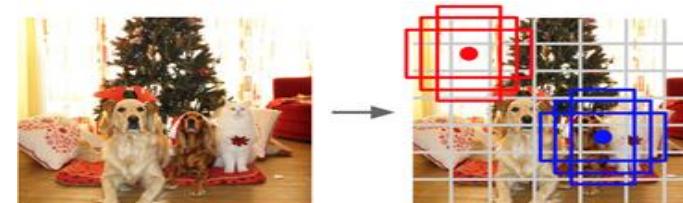
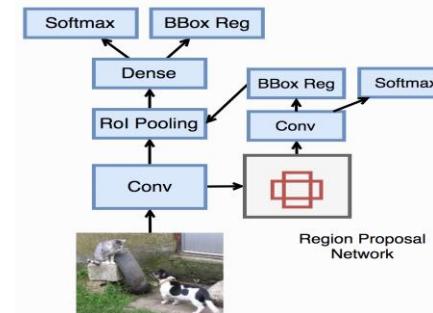
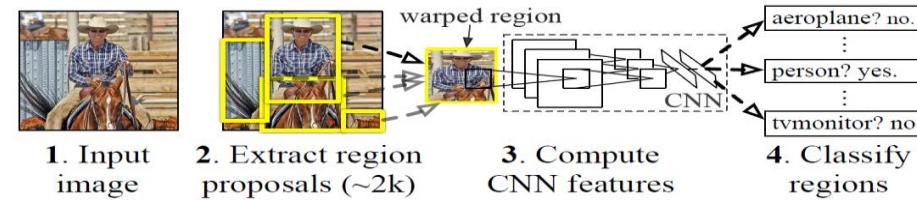
Faster R-CNN

One-stage detectors

YOLO

SSD

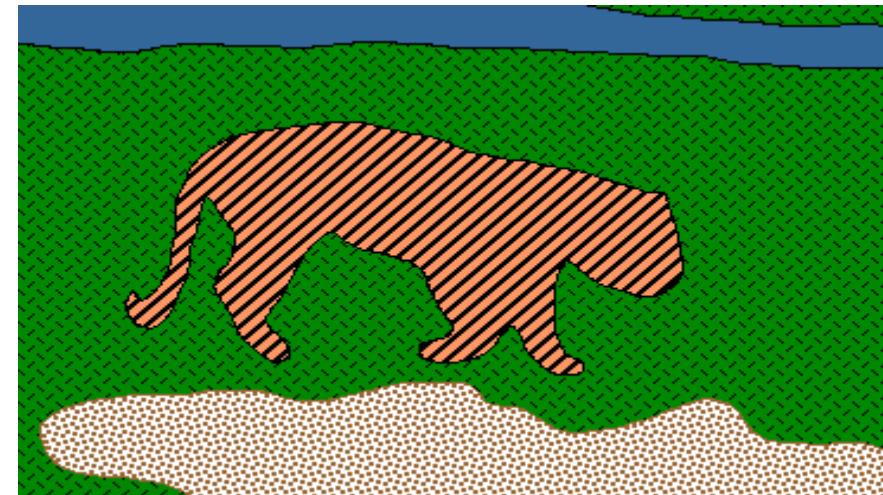
RetinaNet



Today's class

Image Segmentation

Group pixels into meaningful or perceptually similar regions



Outline

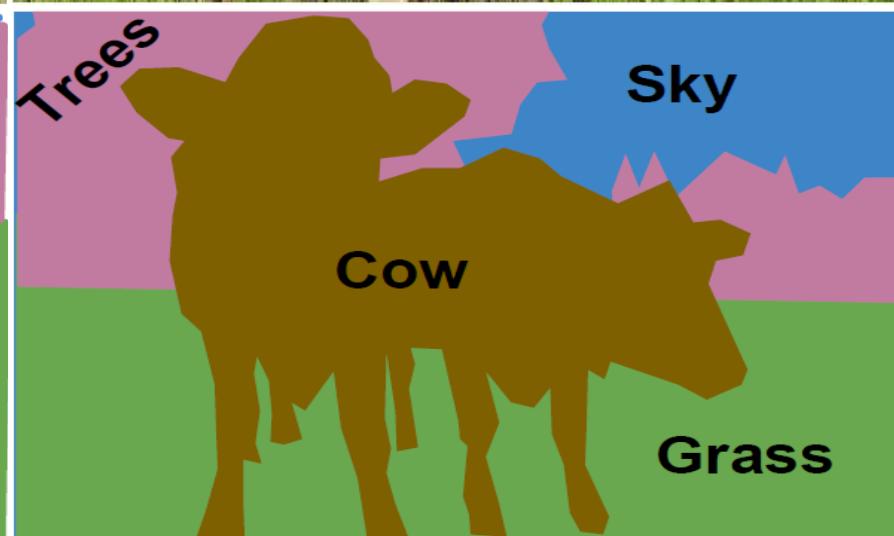
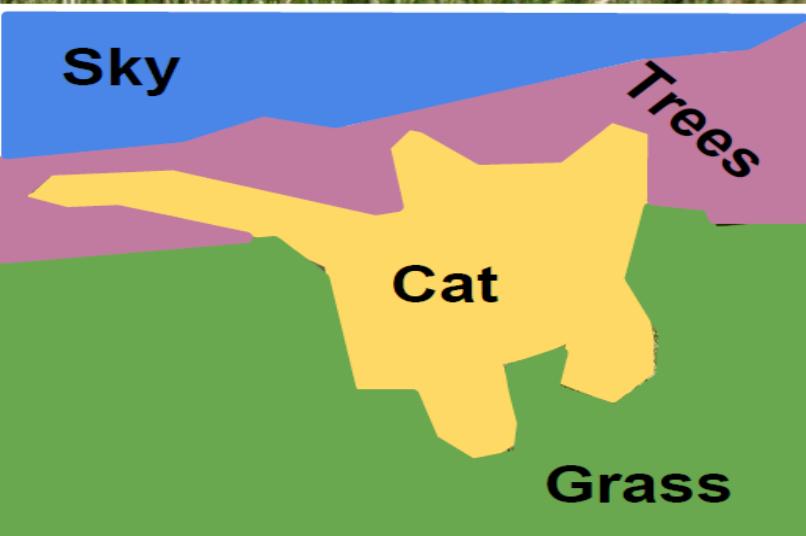
- Early “hacks”
 - Sliding window
 - Fully convolutional networks
- Deep network operations for dense prediction
 - Transposed convolutions
 - Unpooling
 - Dilated convolutions
- Instance segmentation
 - Mask R-CNN
- Other dense prediction problems

Semantic Segmentation

Label each pixel in the image with a category label

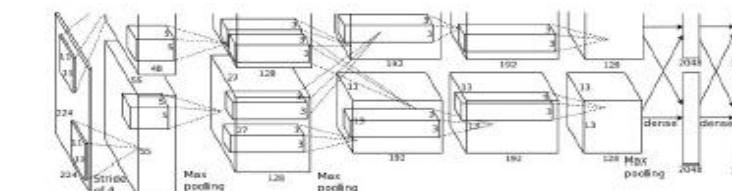
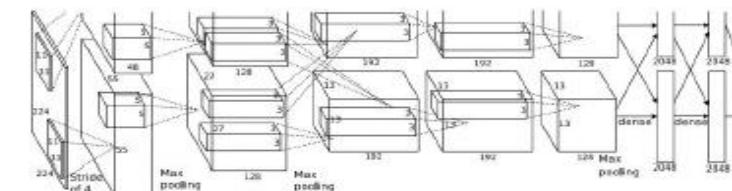
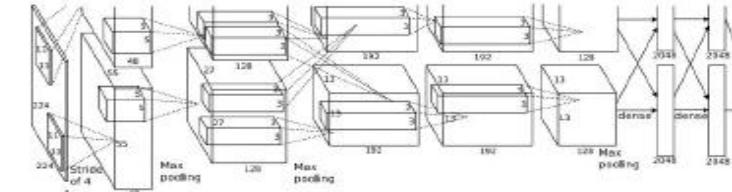
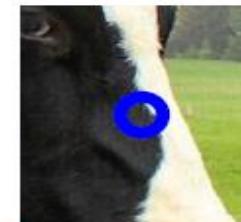
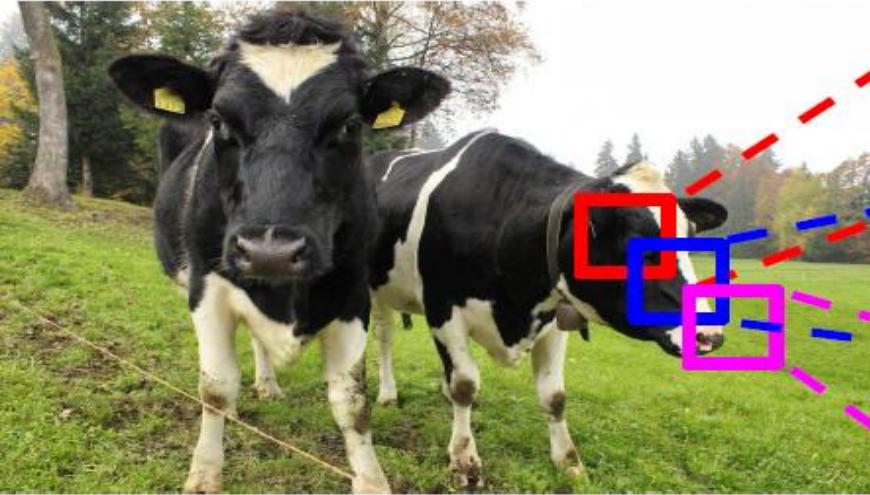
Don't differentiate instances, only care about pixels

These images are CC0 public domain [source1](#) [source2](#)



Semantic Segmentation: Sliding Window

Full image

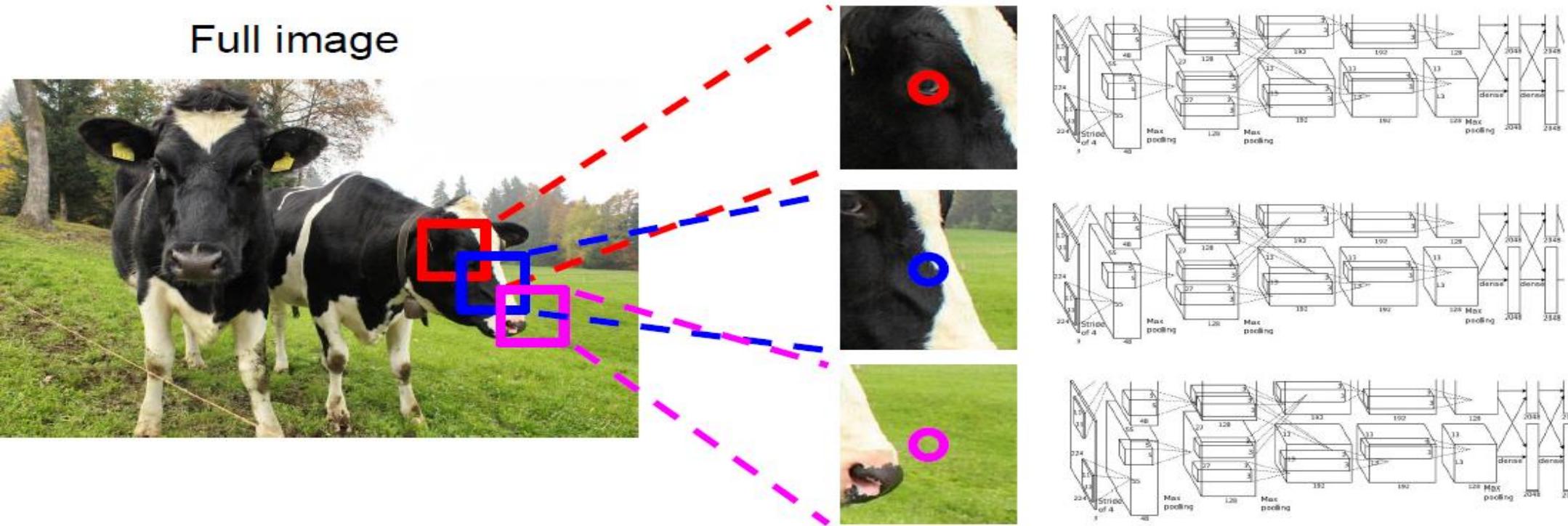


Cow

Cow

Grass

Semantic Segmentation: Sliding Window



Problem:

Very inefficient!

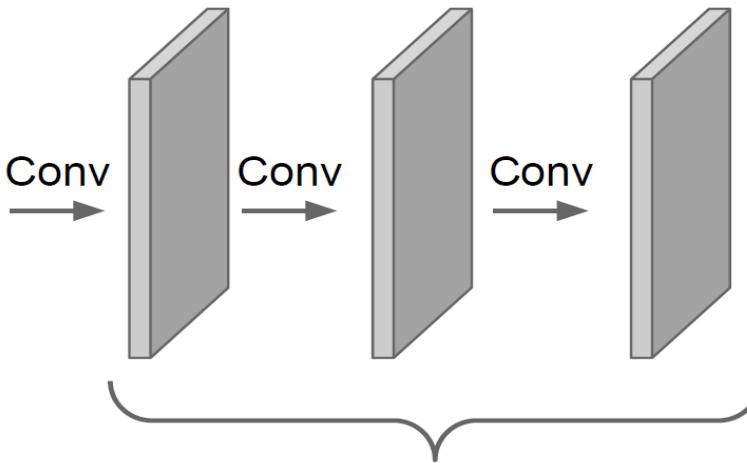
Not reusing shared features between overlapping patches

Semantic Segmentation: Fully Convolutional

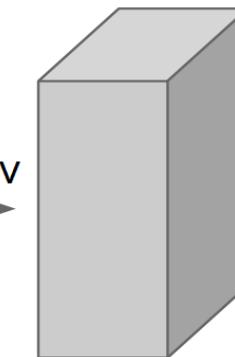
Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



Input:
 $3 \times H \times W$

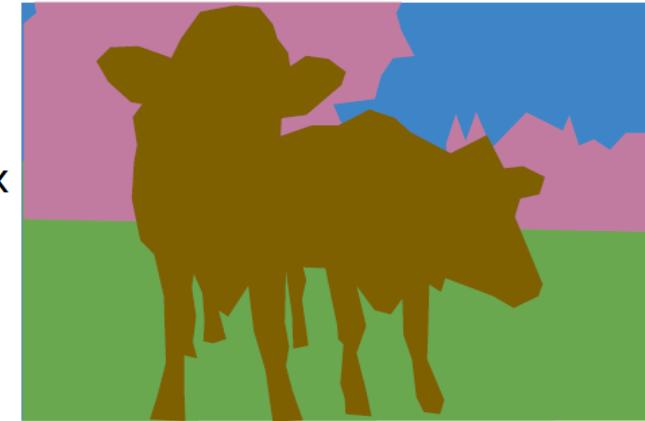


Convolutions:
 $D \times H \times W$



Scores:
 $C \times H \times W$

argmax



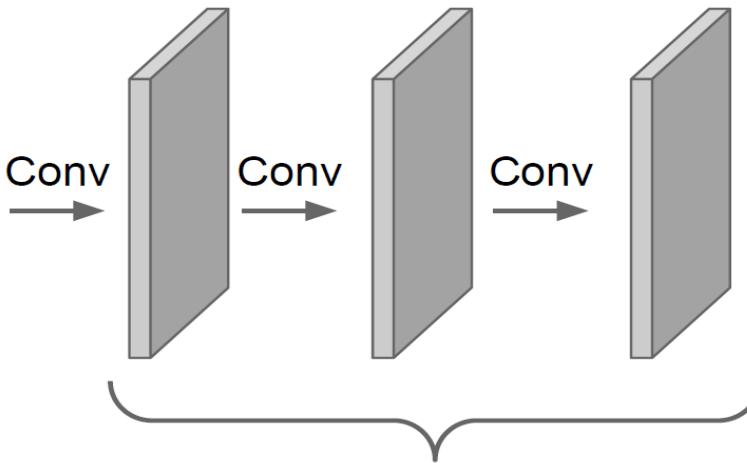
Predictions:
 $H \times W$

Semantic Segmentation: Fully Convolutional

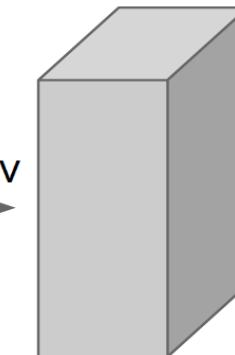
Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



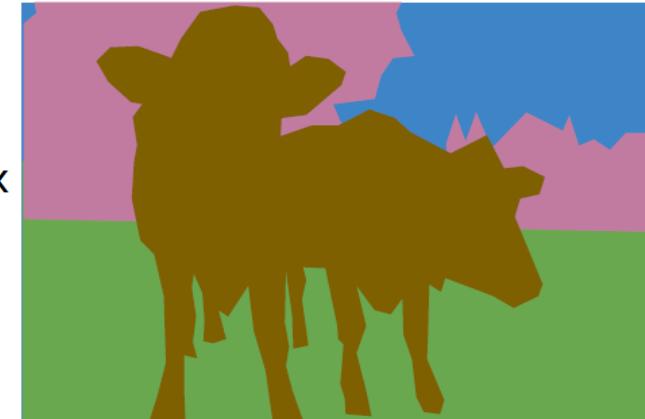
Input:
 $3 \times H \times W$



Convolutions:
 $D \times H \times W$



argmax



Predictions:
 $H \times W$

Problem:

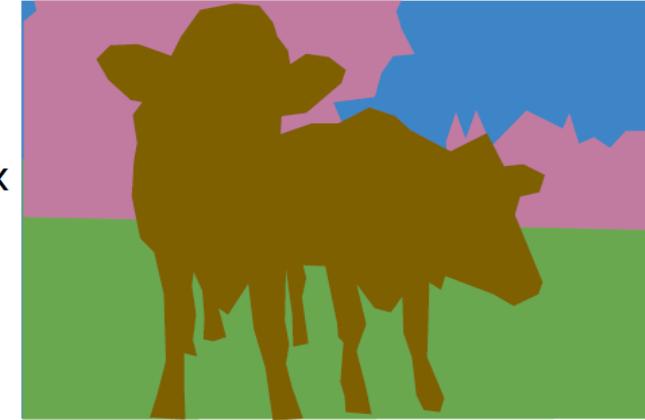
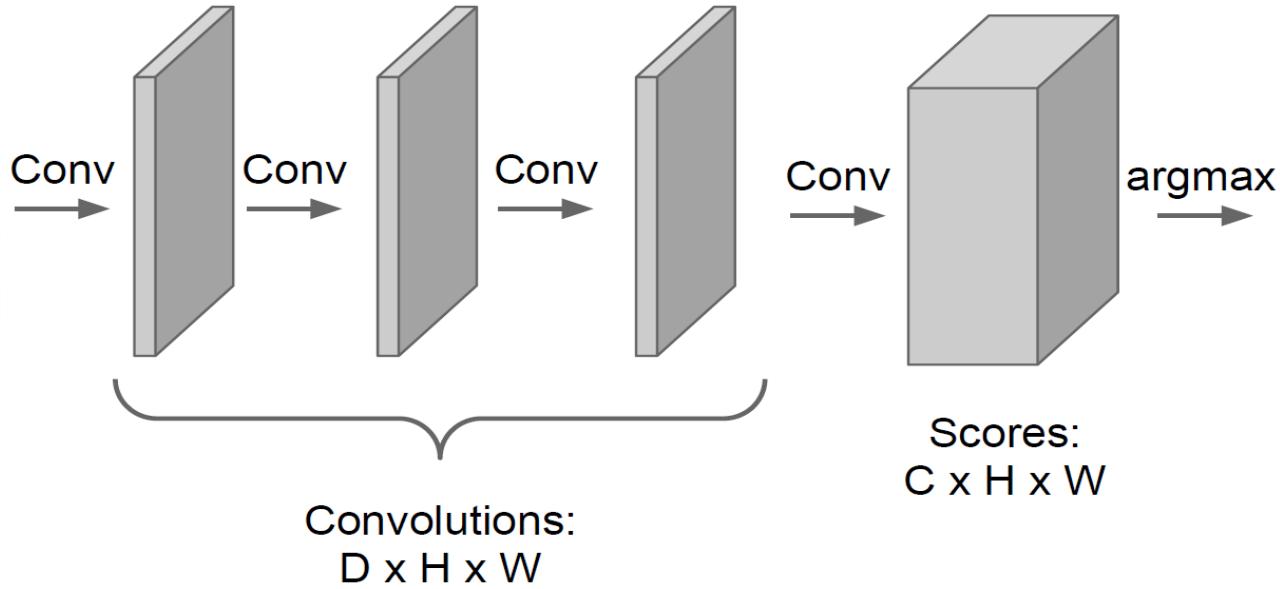
Convolutions at original image resolution will be very expensive ...

Semantic Segmentation: Fully Convolutional

Design network as a bunch of convolutional layers, with
downsampling and **upsampling** inside the network!



Input:
 $3 \times H \times W$



Predictions:
 $H \times W$

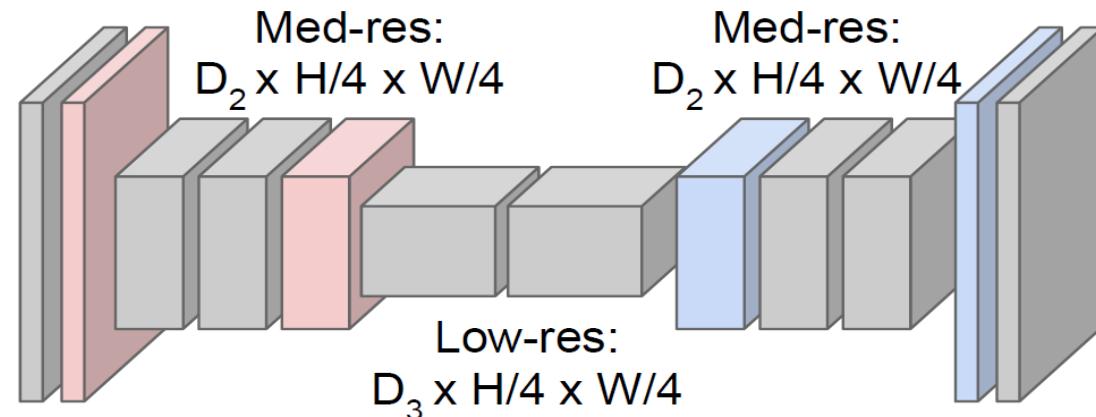
Semantic Segmentation: Fully Convolutional

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with
downsampling and **upsampling** inside the network!



Upsampling:
??



Predictions:
 $H \times W$

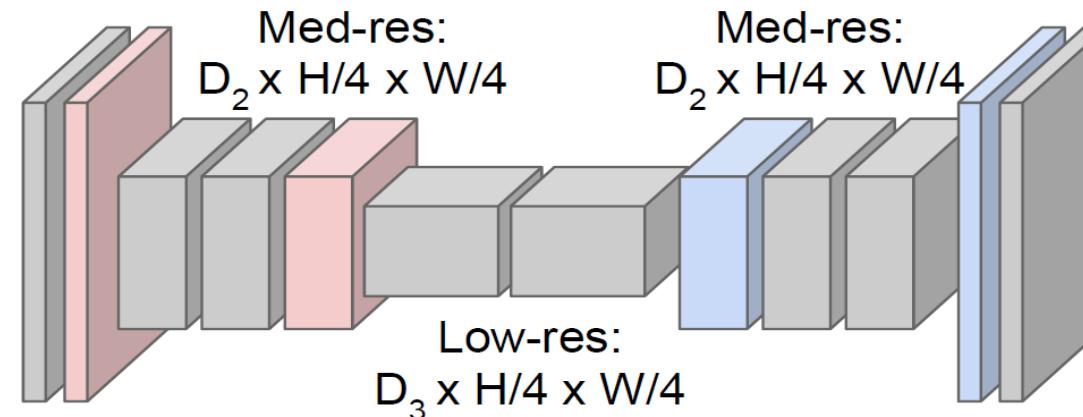
Semantic Segmentation: Fully Convolutional

Downsampling:
Pooling, strided
convolution

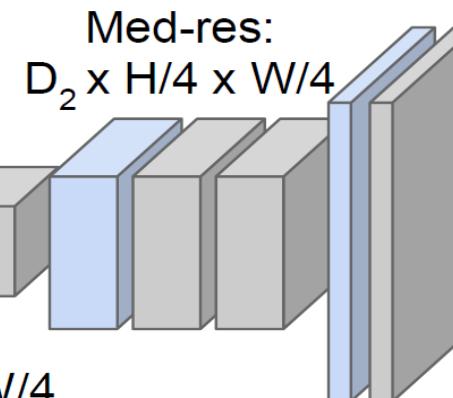


Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with
downsampling and **upsampling** inside the network!



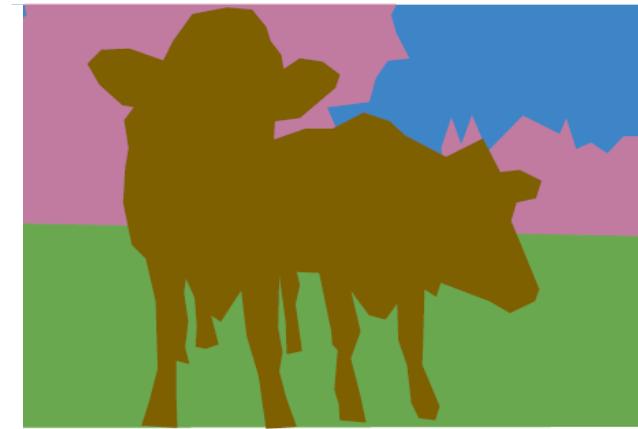
High-res:
 $D_1 \times H/2 \times W/2$



Low-res:
 $D_3 \times H/4 \times W/4$

High-res:
 $D_1 \times H/2 \times W/2$

Upsampling:
Unpooling or strided
transpose convolution



Predictions:
 $H \times W$

Upsampling: Unpooling

Nearest Neighbor

1	2
3	4



1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Input: 2×2

Output: 4×4

Upsampling: Unpooling

Nearest Neighbor

1	2
3	4



1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

Input: 2 x 2

“Bed of Nails”

1	2
3	4



1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

Input: 2 x 2

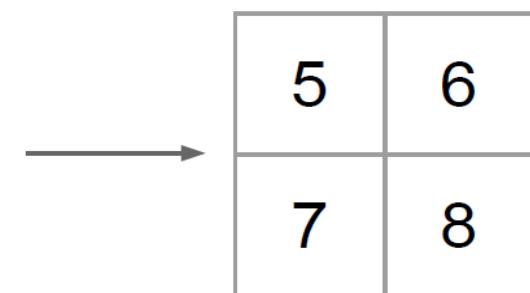
Upsampling: Max Unpooling

Max Pooling

Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4×4



Output: 2×2

Upsampling: Max Unpooling

Max Pooling

Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4×4

5	6
7	8

Output: 2×2

Max Unpooling

Use positions from pooling layer

1	2
3	4

Input: 2×2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4×4

Upsampling: Max Unpooling

Max Pooling

Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4×4

Corresponding pairs of
downsampling and
upsampling layers

5	6
7	8

Output: 2×2

Max Unpooling

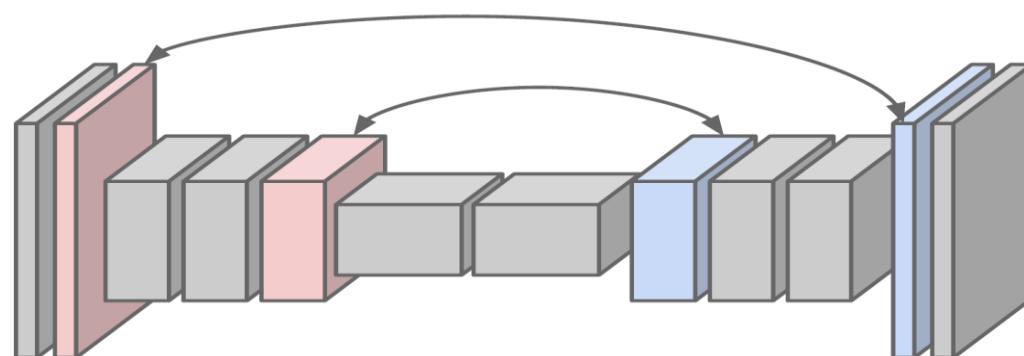
Use positions from pooling layer

1	2
3	4

Input: 2×2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4×4



Upsampling: Transpose Convolution

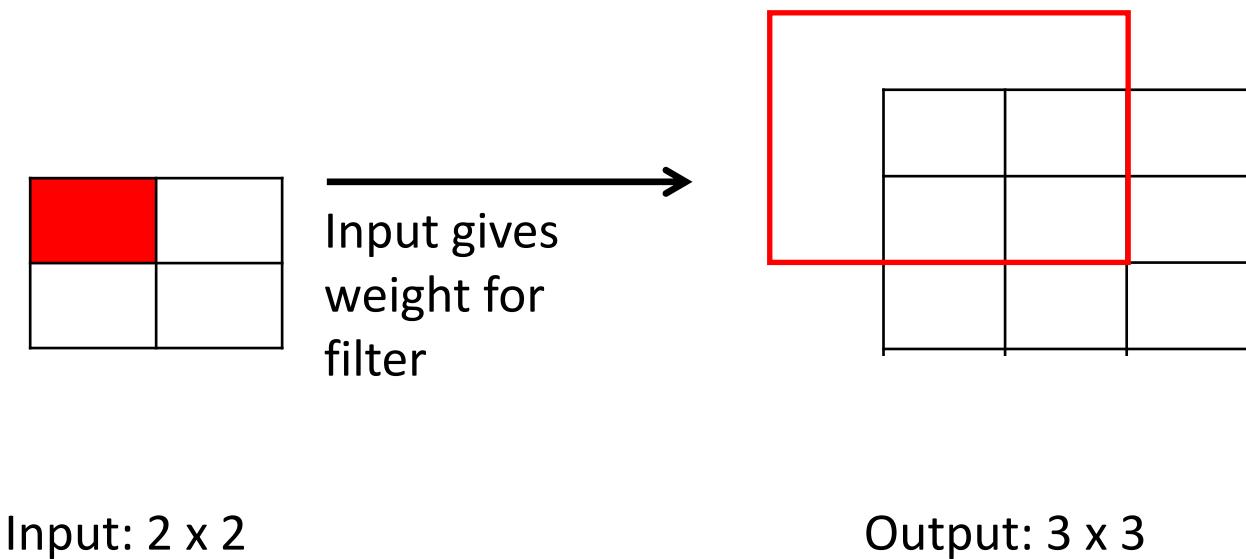
3 x 3 **transpose** convolution, stride 2 pad 1

Input: 2 x 2

Output: 3 x 3

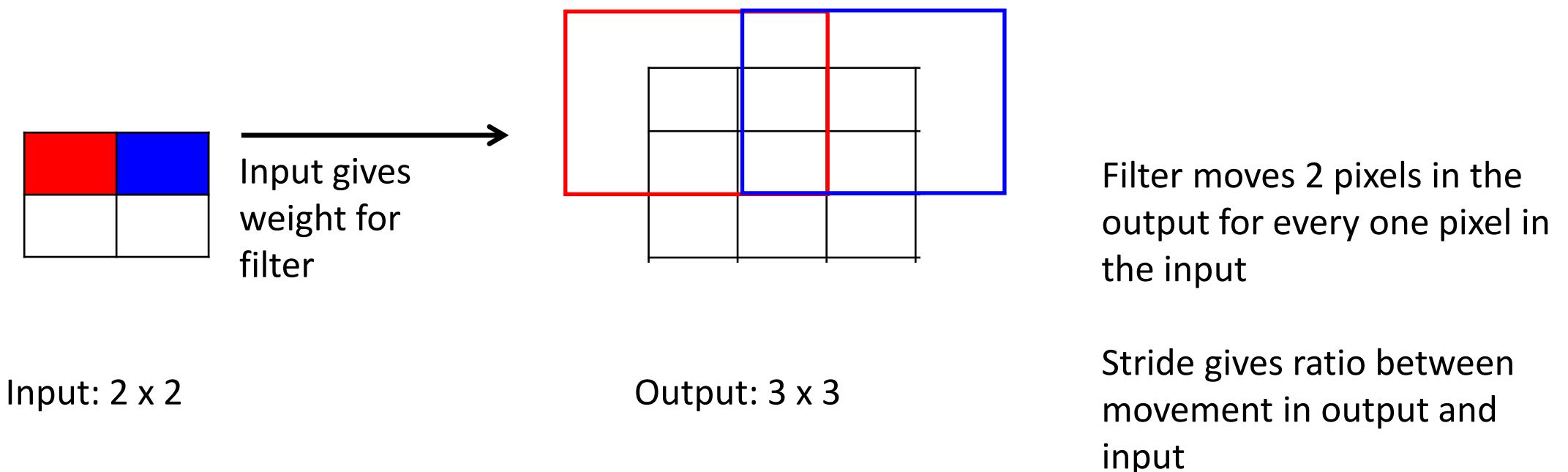
Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1



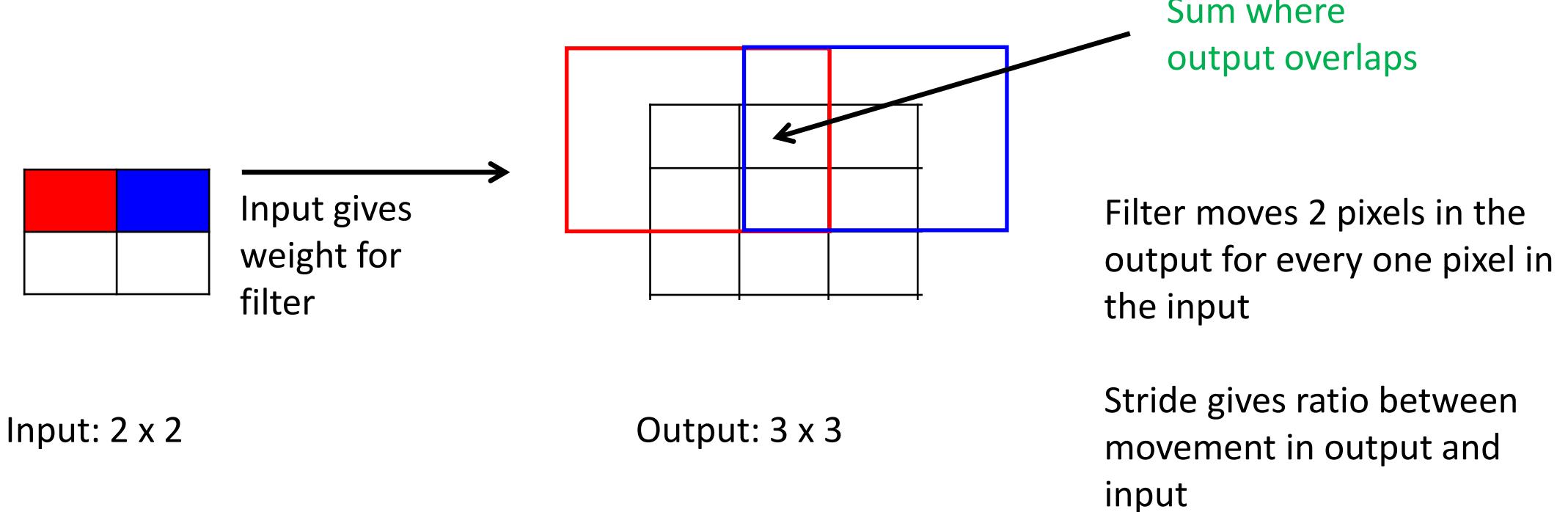
Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1



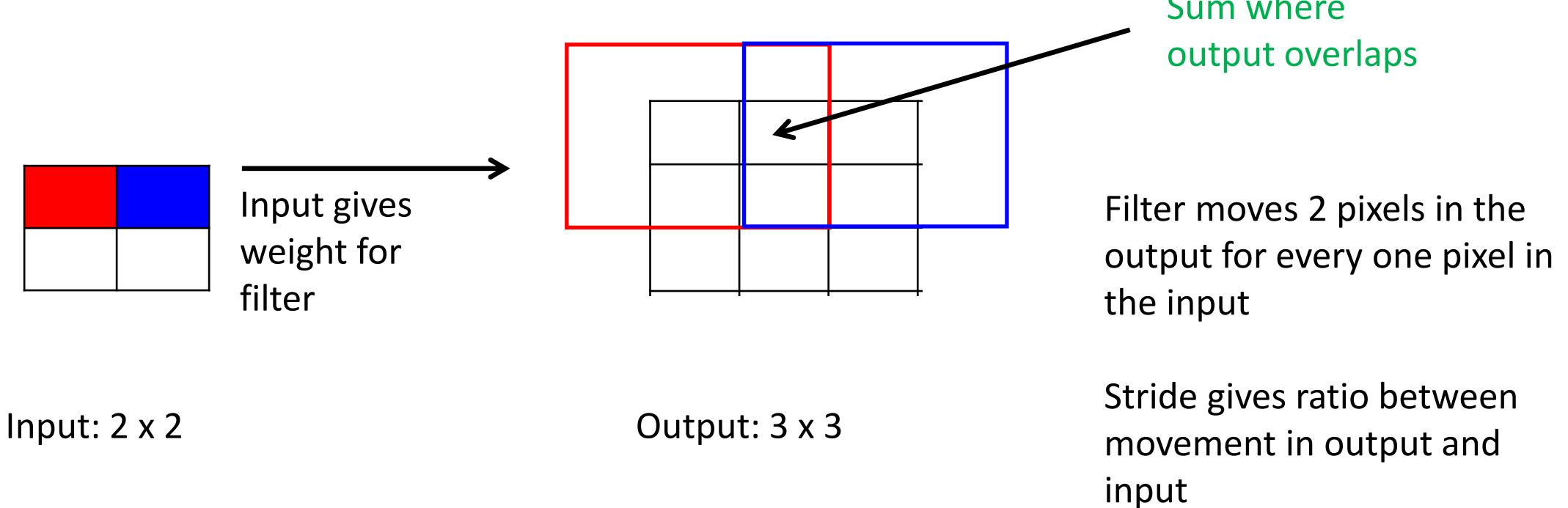
Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1



Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

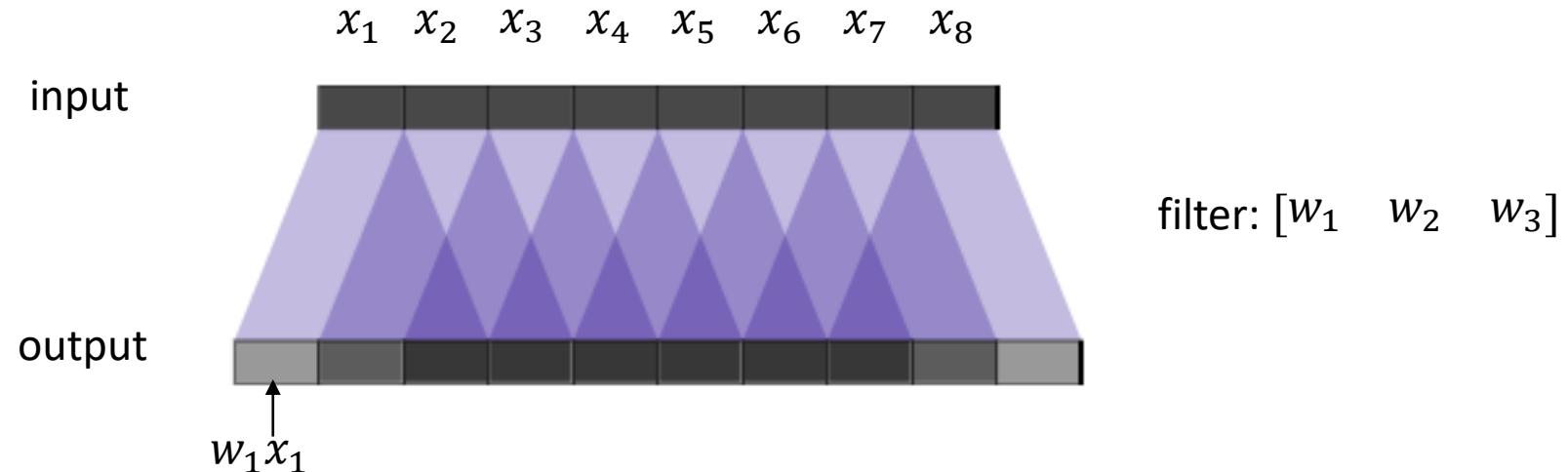


Actual output: 5*5

Need to crop 3*3 centrally

Upsampling in a deep network

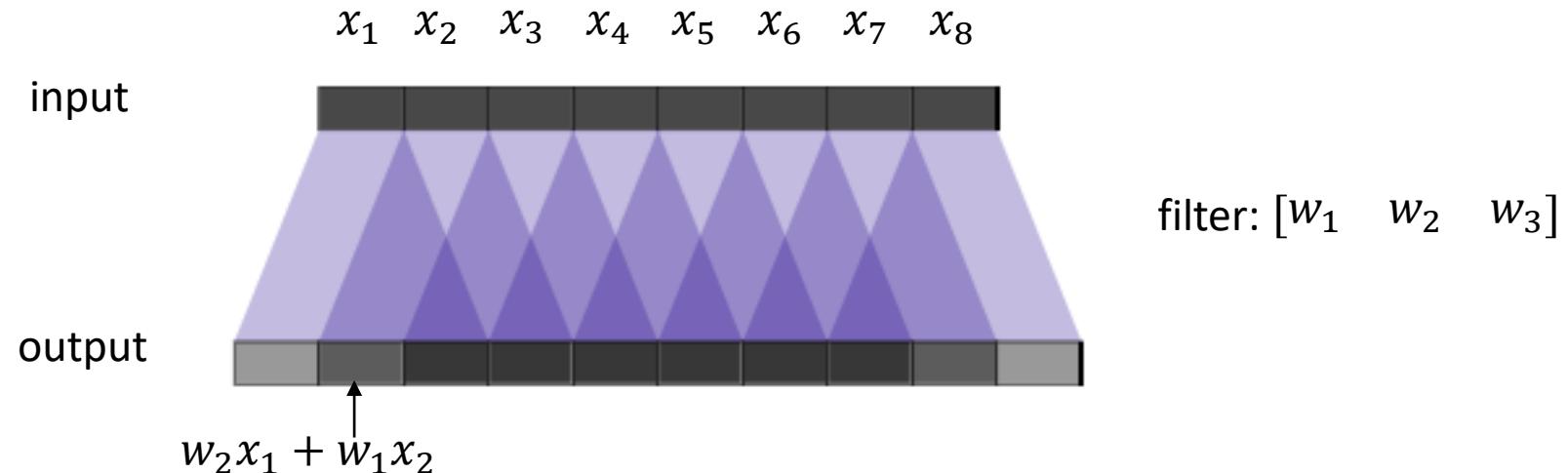
- 1D example



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

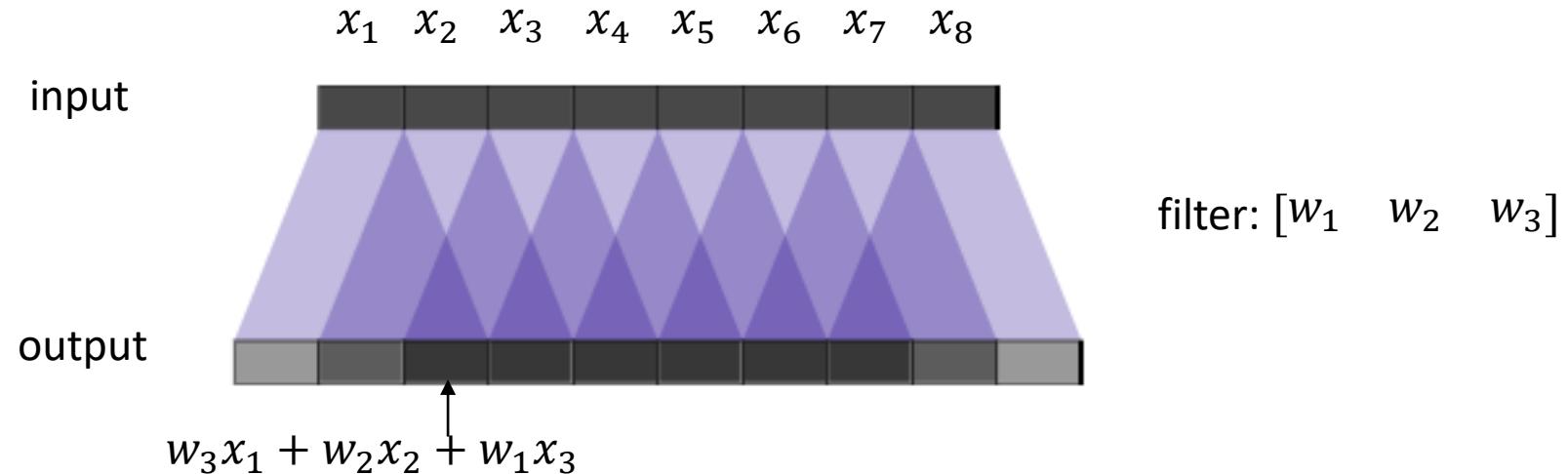
- 1D example



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

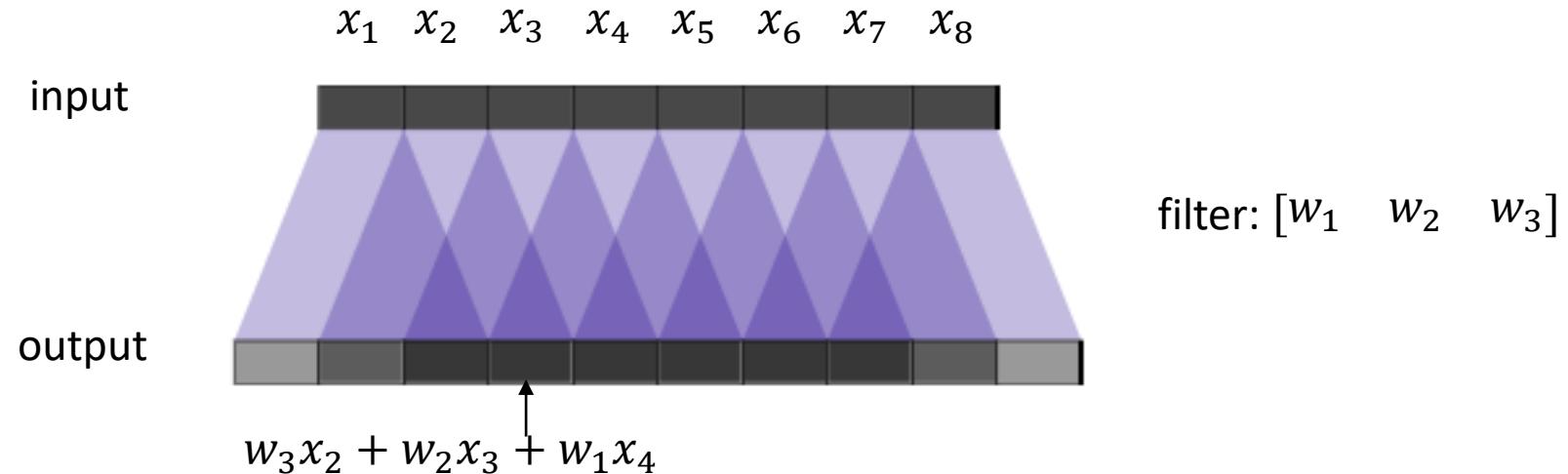
- 1D example



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

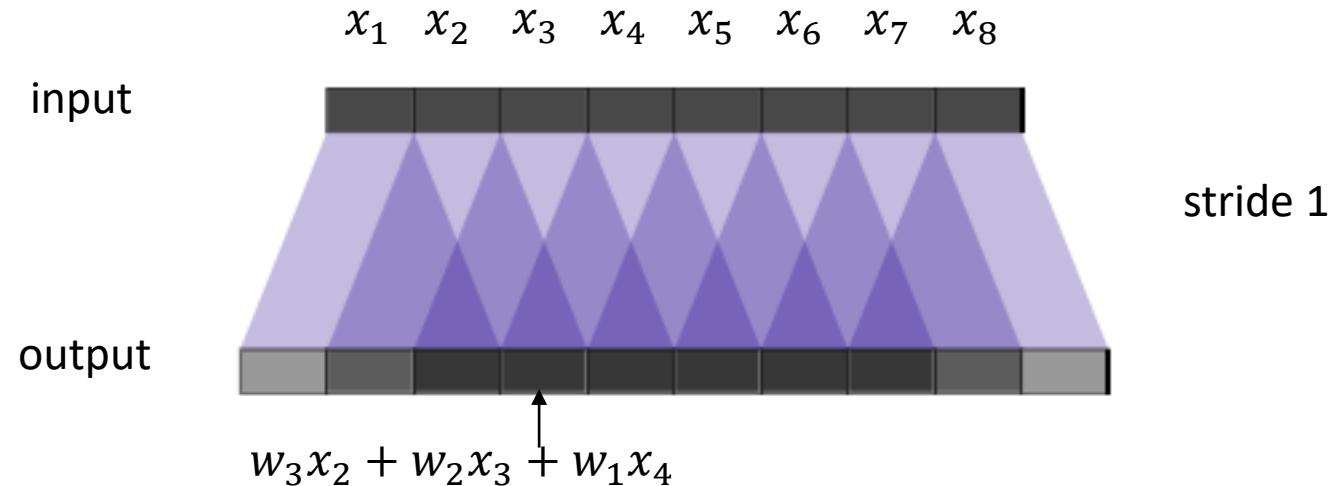
- 1D example



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

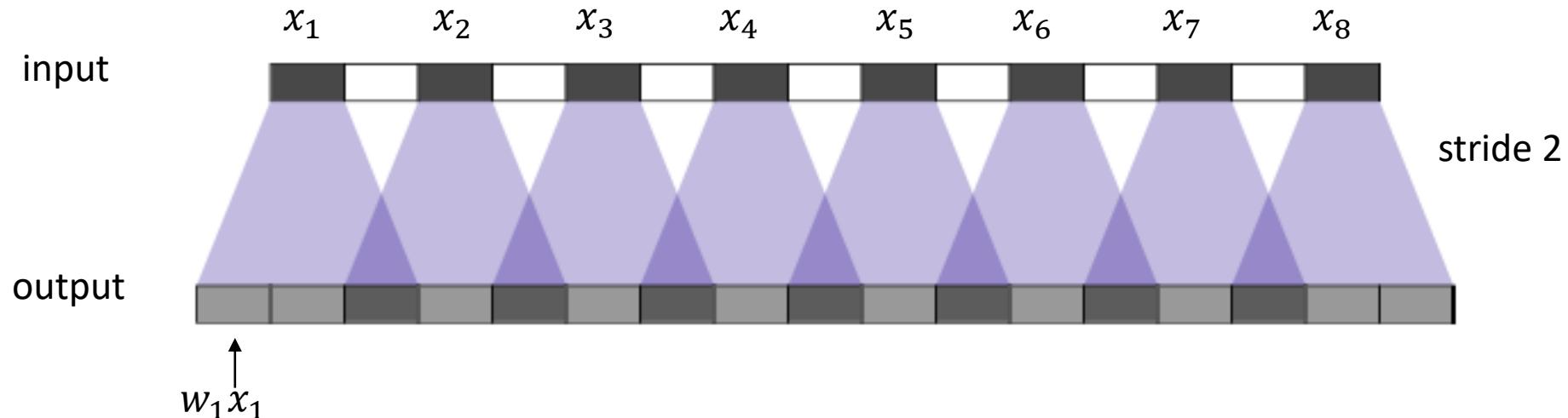
- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

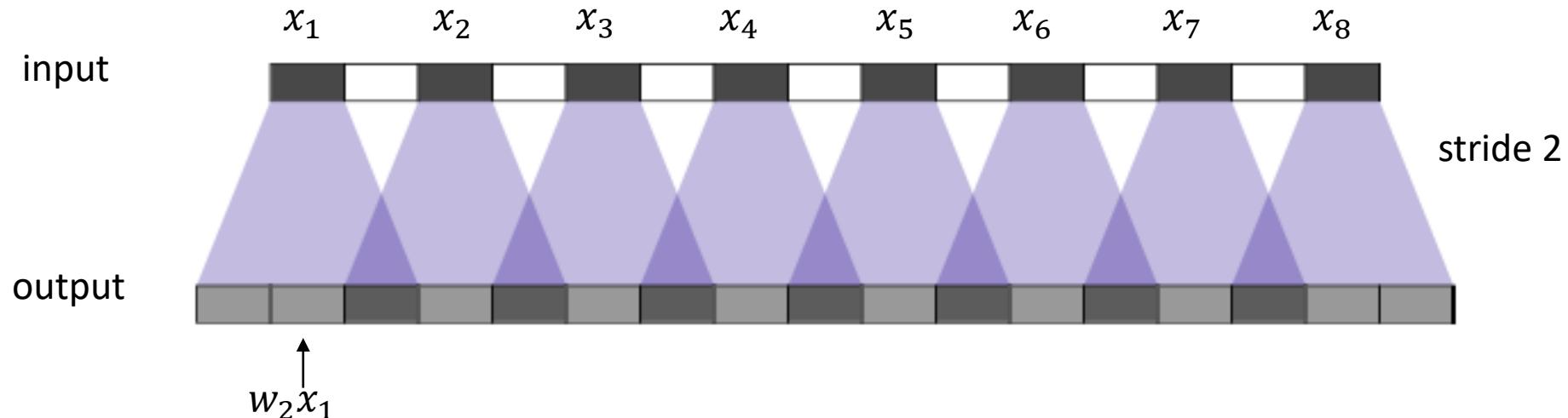
- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

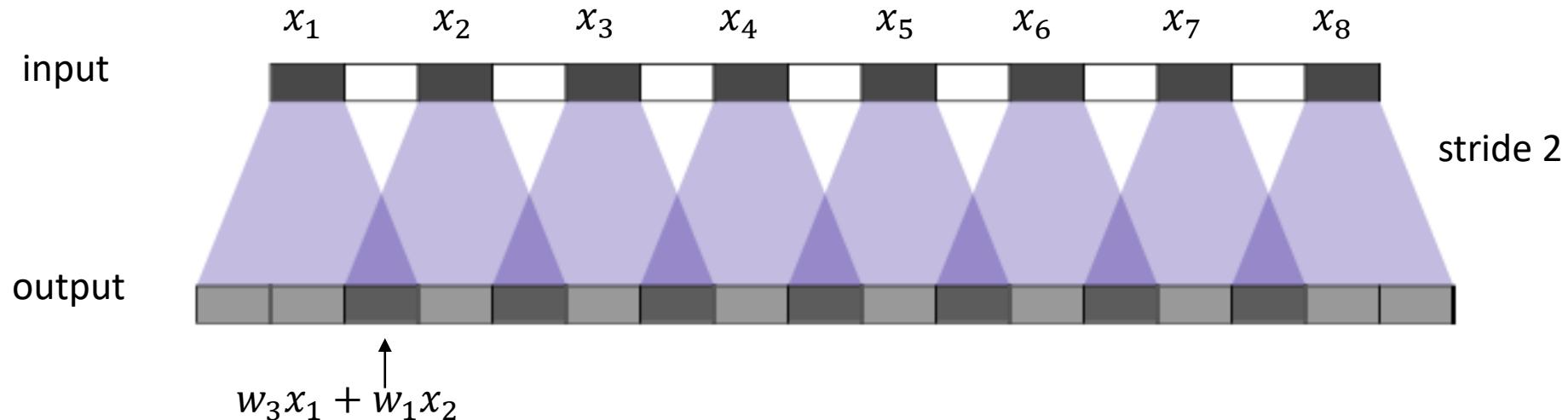
- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

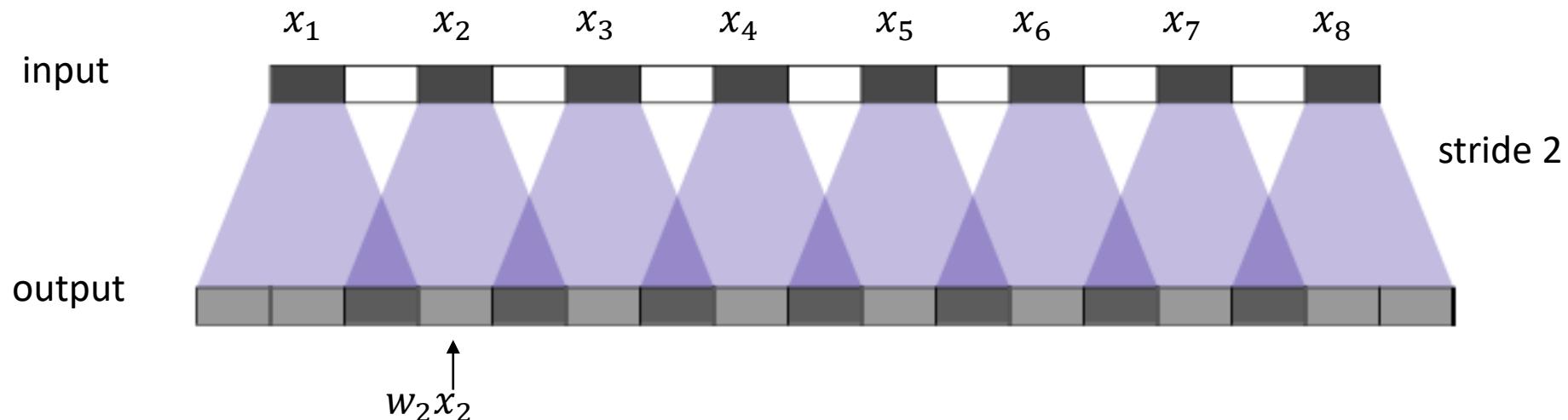
- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

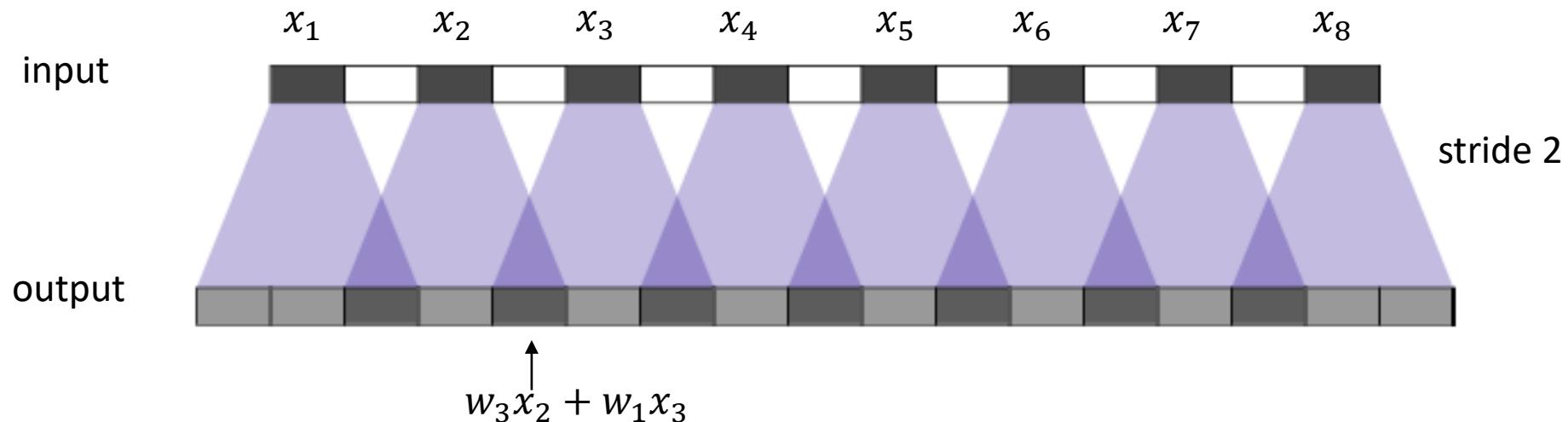
- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

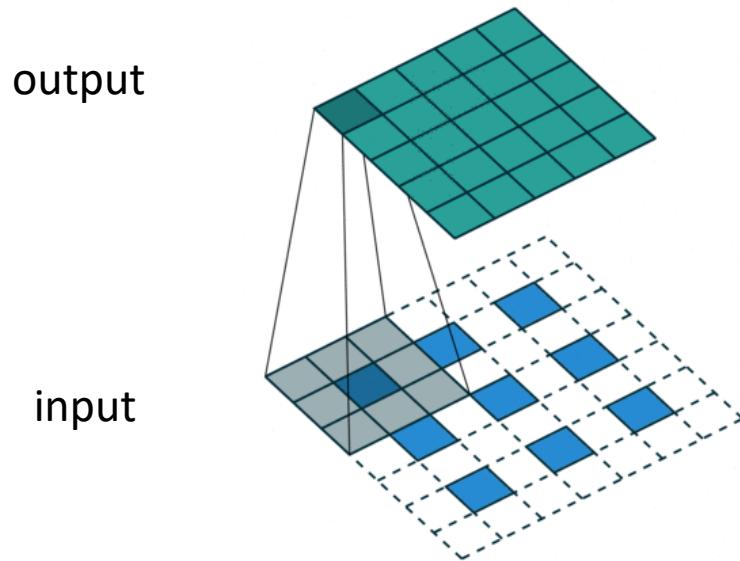
- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1



Animation: <https://distill.pub/2016/deconv-checkerboard/>

Upsampling in a deep network

- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1
 - For stride 2, dilate the input by inserting rows and columns of zeros between adjacent entries
 - Sometimes called convolution with *fractional input stride* 1/2

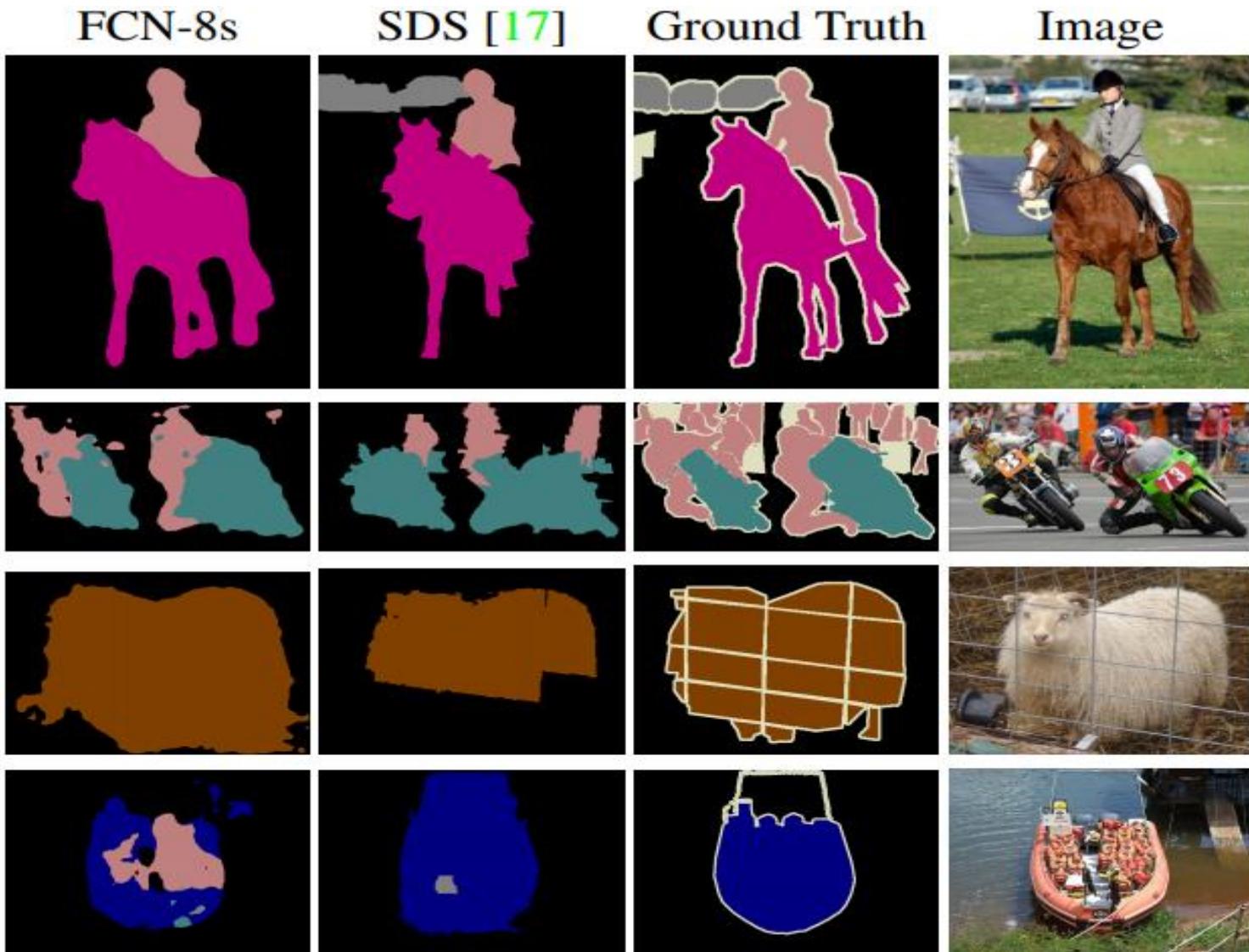


Q: What 3x3 filter would correspond to bilinear upsampling?

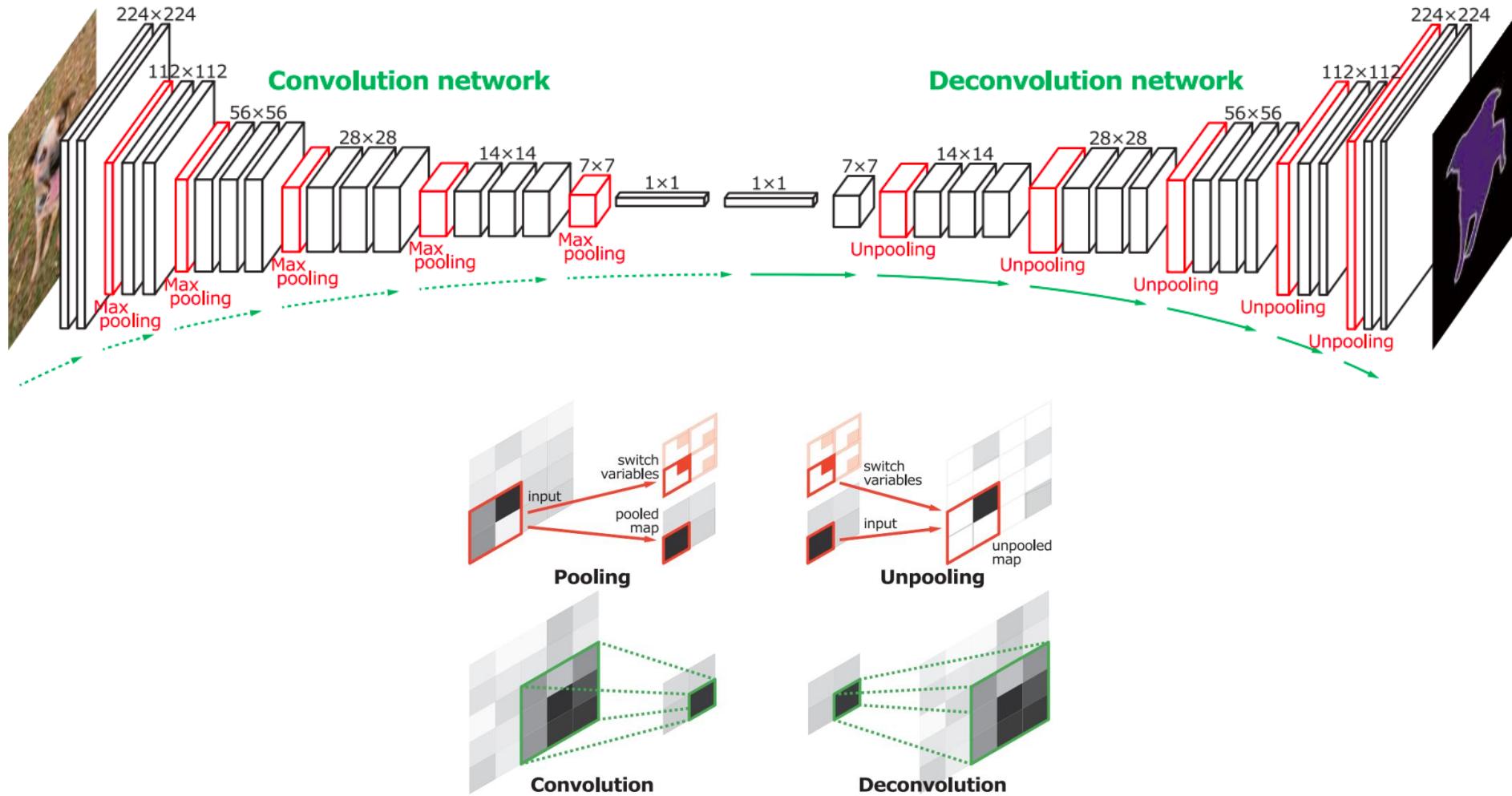
$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$
$\frac{1}{2}$	1	$\frac{1}{2}$
$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$

“Fully convolutional (FCN)” results

- Takes advantage of pre-training from classification
- Applied to objects and scenes (NYUD v2)
- But feature pooling reduces spatial sensitivity and resolution



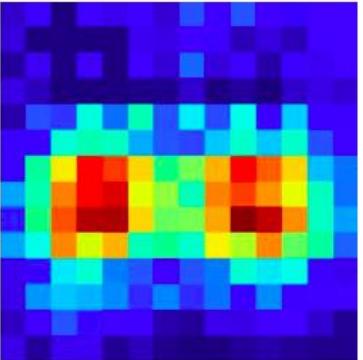
DeconvNet



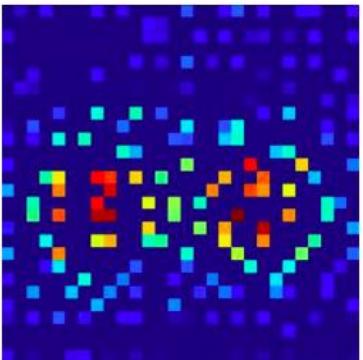
DeconvNet



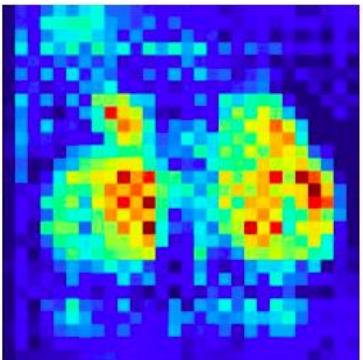
Original image



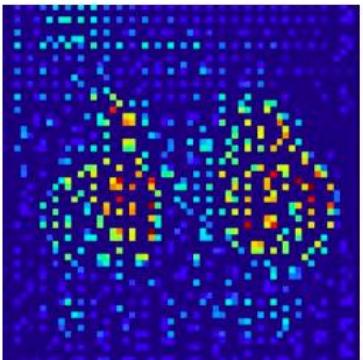
14x14 deconv



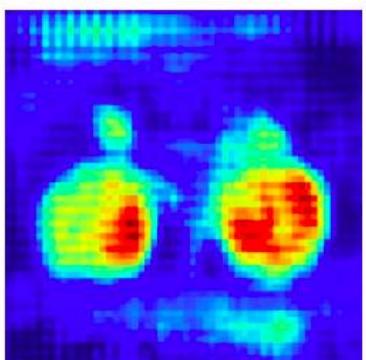
28x28 unpooling



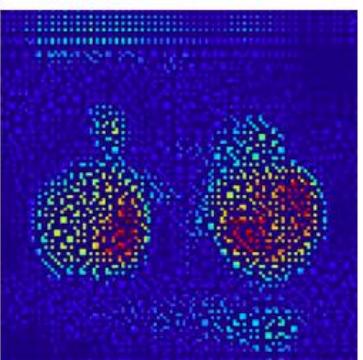
28x28 deconv



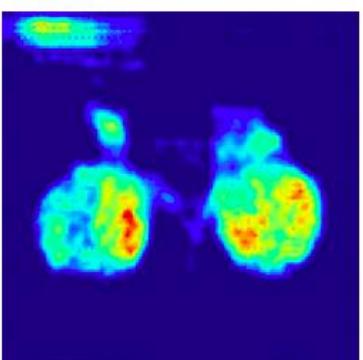
54x54 unpooling



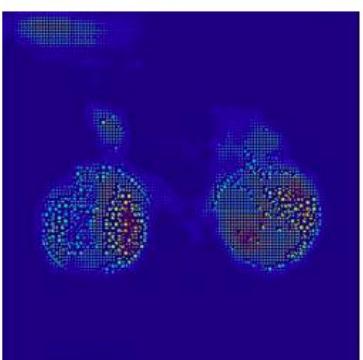
54x54 deconv



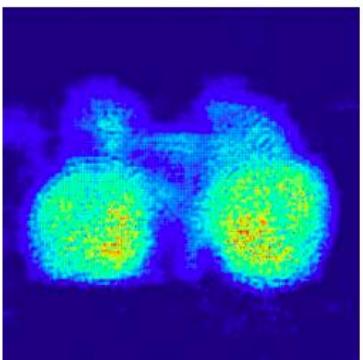
112x112 unpooling



112x112 deconv



224x224 unpooling

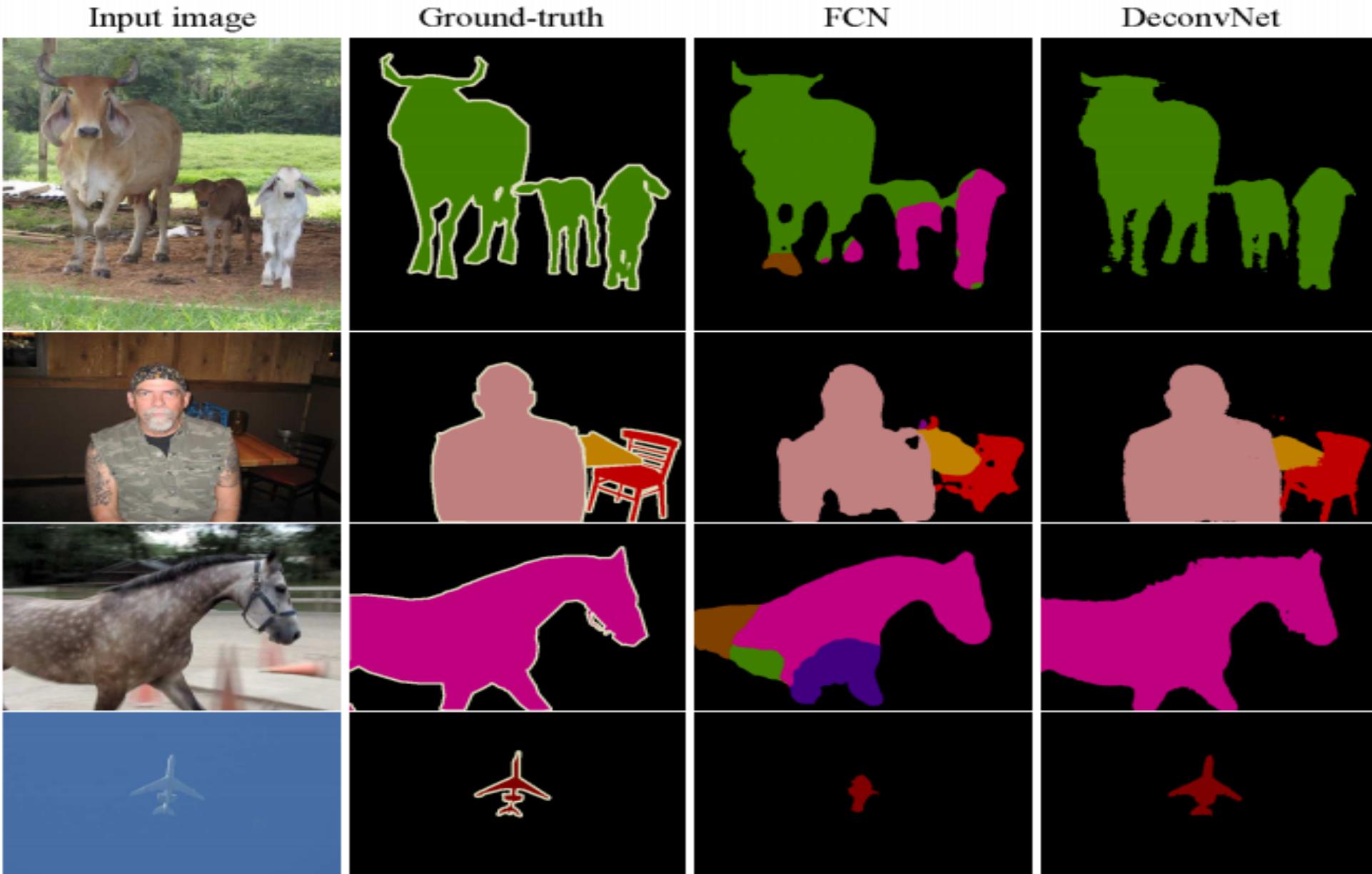


224x224 deconv

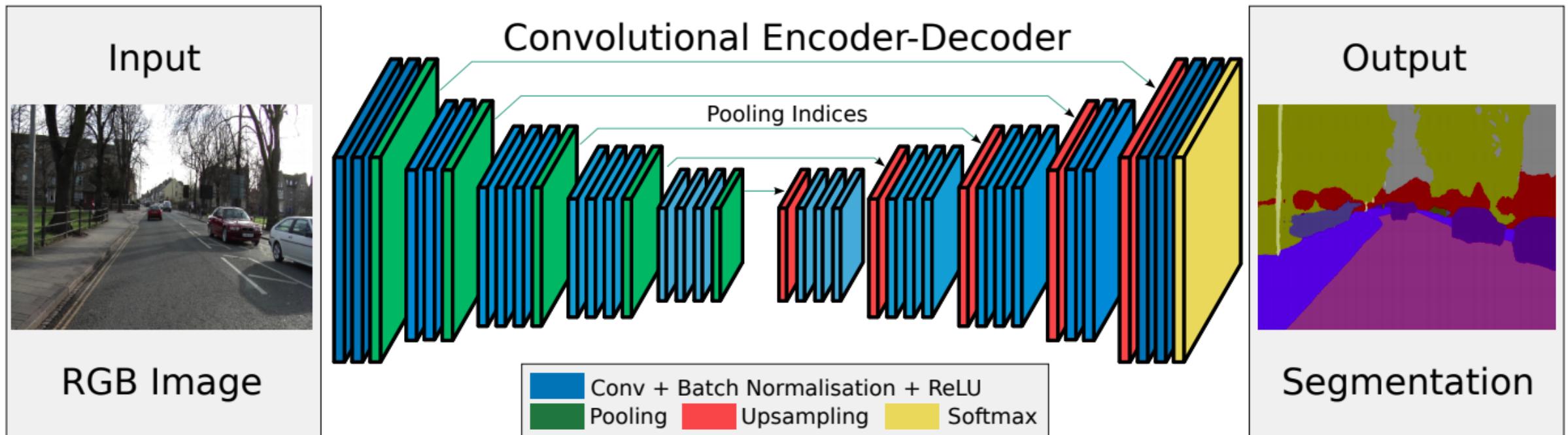
DeconvNet results

PASCAL VOC 2012	mIoU
FCN-8	62.2
DeconvNet	69.6
Ensemble of DeconvNet and FCN	71.7

“DeconNet” results



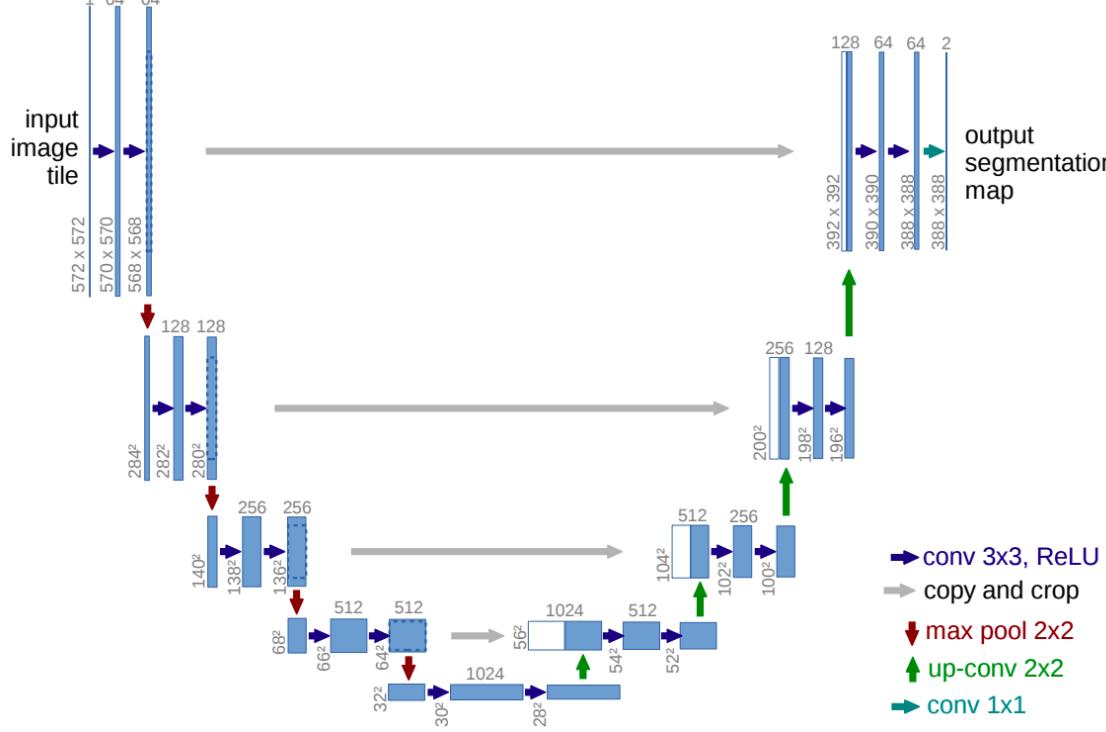
Similar architecture: SegNet



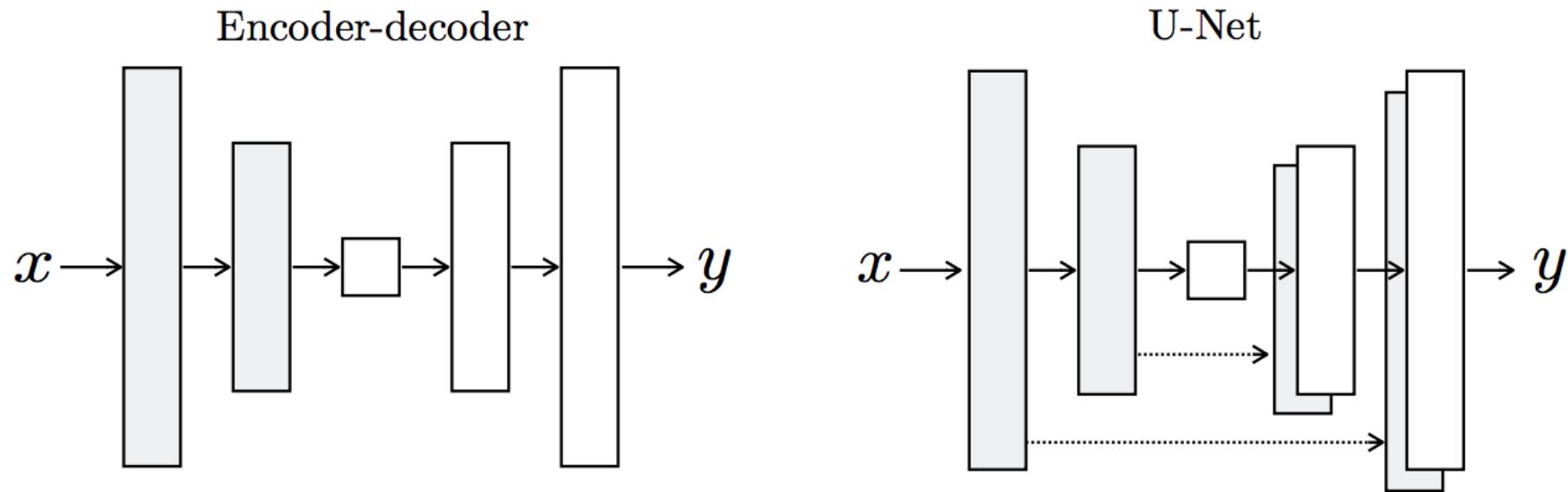
Drop the FC layers,
get better results

U-Net

- Like FCN, fuse upsampled higher-level feature maps with higher-resolution, lower-level feature maps
- Unlike FCN, fuse by concatenation, predict at the end



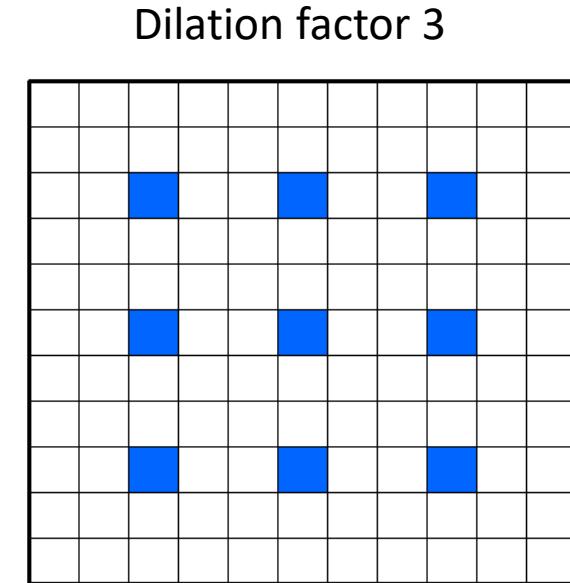
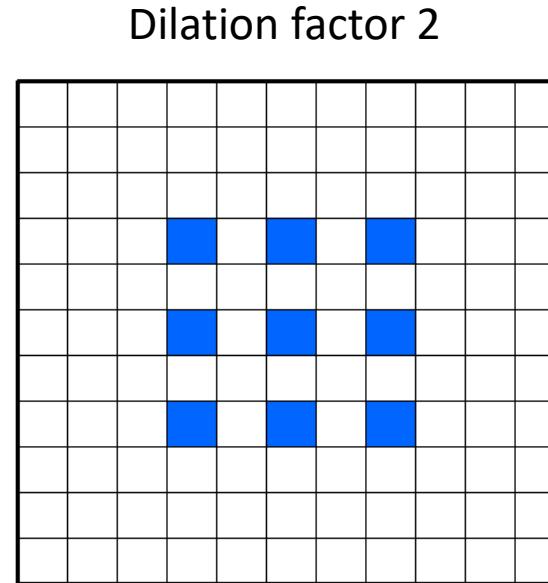
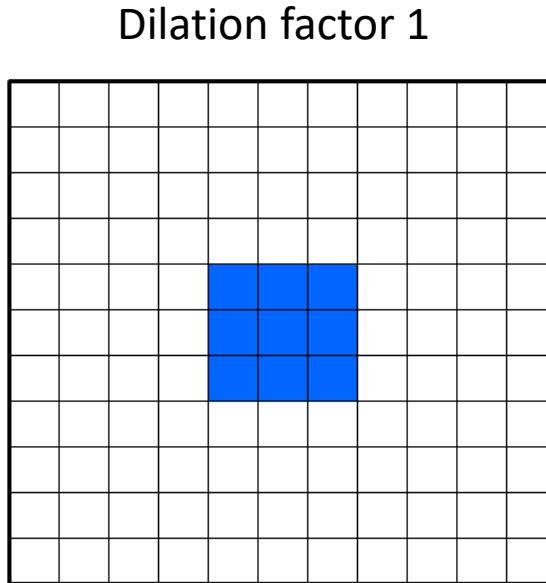
Summary of upsampling architectures



[Figure source](#)

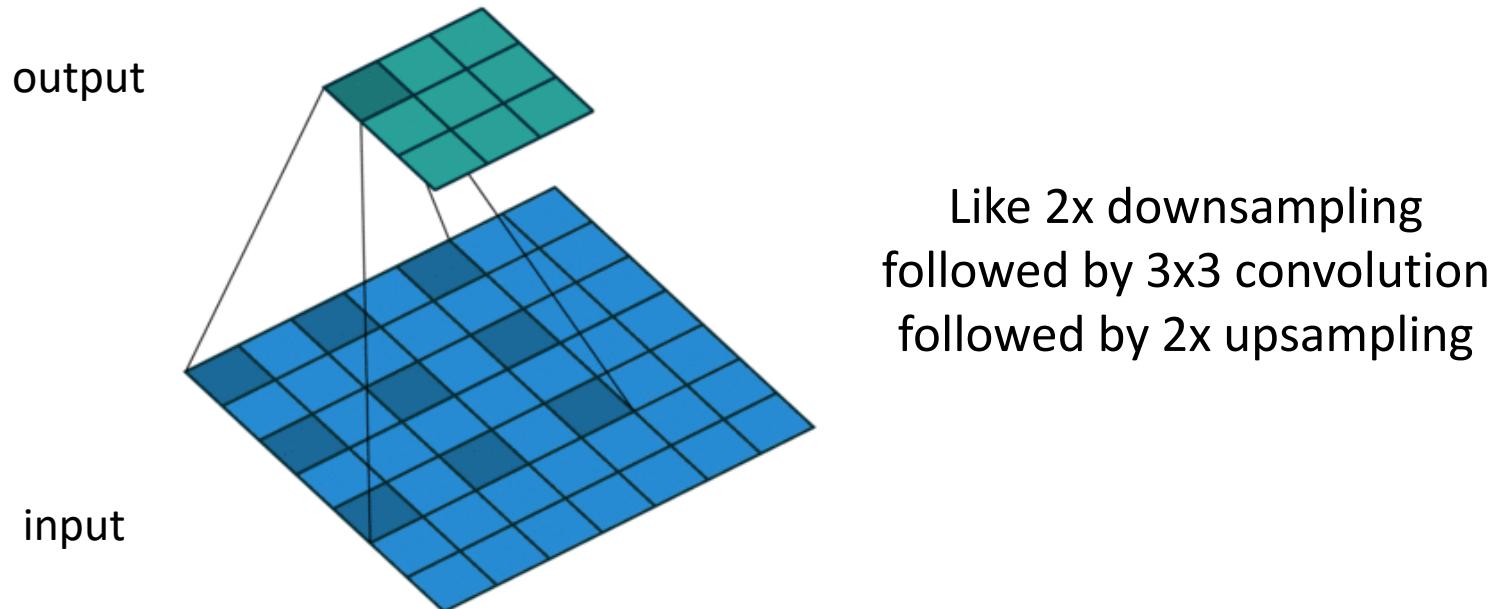
Dilated convolutions

- Idea: instead of reducing spatial resolution of feature maps, use a large sparse filter
 - Also known as *à trous* convolution

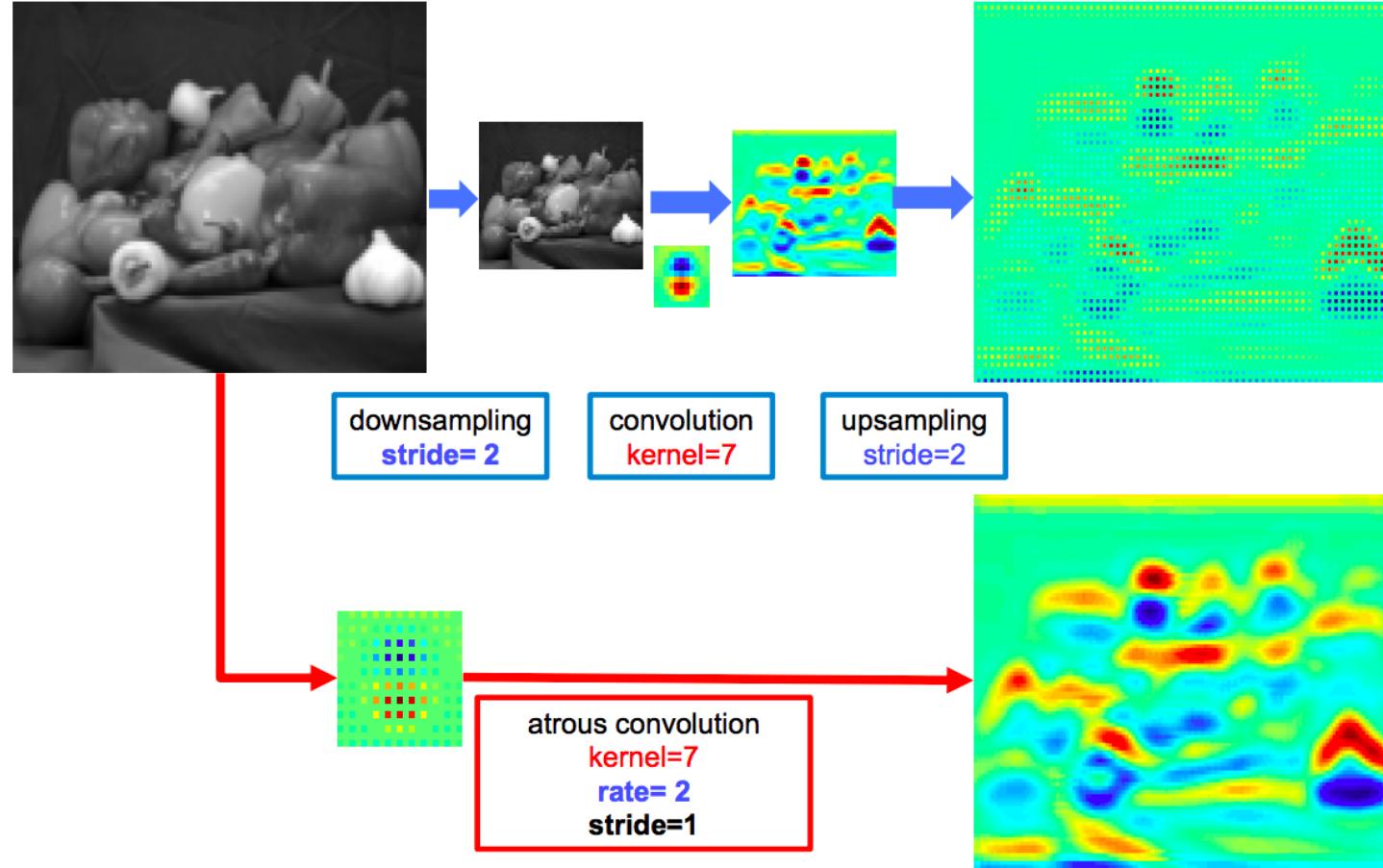


Dilated convolutions

- Idea: instead of reducing spatial resolution of feature maps, use a large sparse filter



Dilated convolutions



L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. Yuille, [DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs](#), PAMI 2017

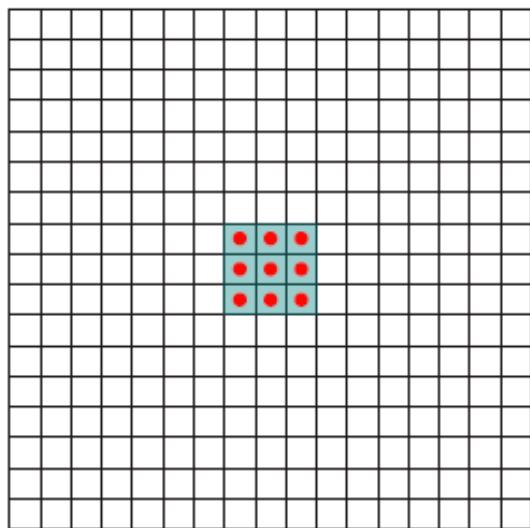
Dilated convolutions

- Can be used in FCN to remove downsampling:
change stride of max pooling layer from 2 to 1,
dilate subsequent convolutions by factor of 2.

Dilated convolutions

- Can increase receptive field size exponentially with a linear growth in the number of parameters

Feature map 1 (F_1)
produced from F_0 by 1-
dilated convolution



Receptive field: 3x3

Receptive field: 7x7

Receptive field: 15x15

Dilated convolutions

- Context module with dilation
 - Returns same number of feature maps at the same resolution as the input, so can be plugged in to replace components of existing dense prediction architectures
 - Requires identity initialization

Layer	1	2	3	4	5	6	7	8
Convolution	3×3	3×3	3×3	3×3	3×3	3×3	3×3	1×1
Dilation	1	1	2	4	8	16	1	1
Truncation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Receptive field	3×3	5×5	9×9	17×17	33×33	65×65	67×67	67×67
Output channels								
Basic	C	C	C	C	C	C	C	C
Large	$2C$	$2C$	$4C$	$8C$	$16C$	$32C$	$32C$	C

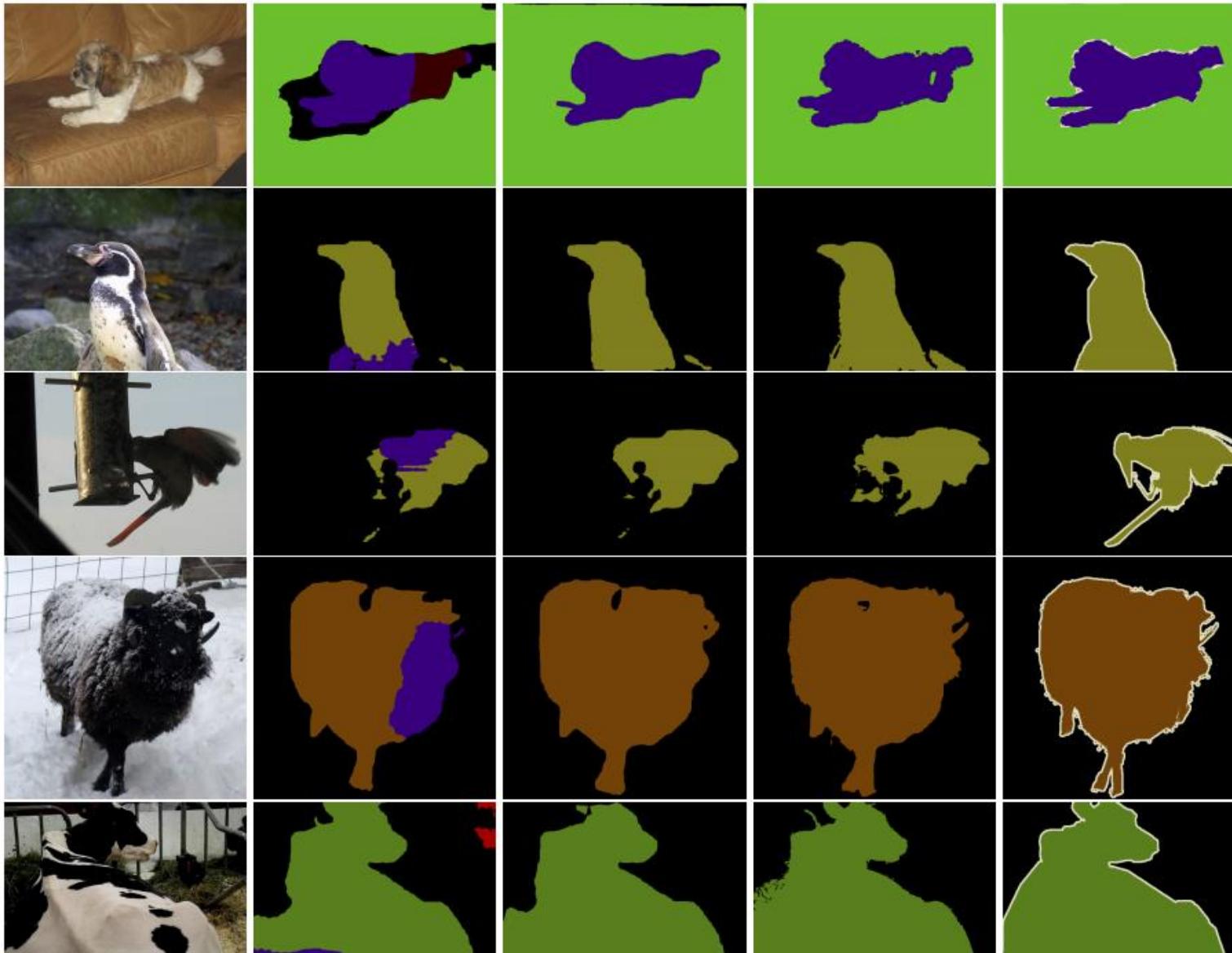
Dilated convolutions: Evaluation

Results on VOC 2012

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean IoU
Front end	86.3	38.2	76.8	66.8	63.2	87.3	78.7	82	33.7	76.7	53.5	73.7	76	76.6	83	51.9	77.8	44	79.9	66.3	69.8
Front + Basic	86.4	37.6	78.5	66.3	64.1	89.9	79.9	84.9	36.1	79.4	55.8	77.6	81.6	79	83.1	51.2	81.3	43.7	82.3	65.7	71.3
Front + Large	87.3	39.2	80.3	65.6	66.4	90.2	82.6	85.8	34.8	81.9	51.7	79	84.1	80.9	83.2	51.2	83.2	44.7	83.4	65.6	72.1

*Front end: re-implementation of FCN-8 with last two pooling layers dropped (5% better than original FCN-8)

Dilated convolutions: Evaluation



(a) Image

(b) Front end

(c) + Context

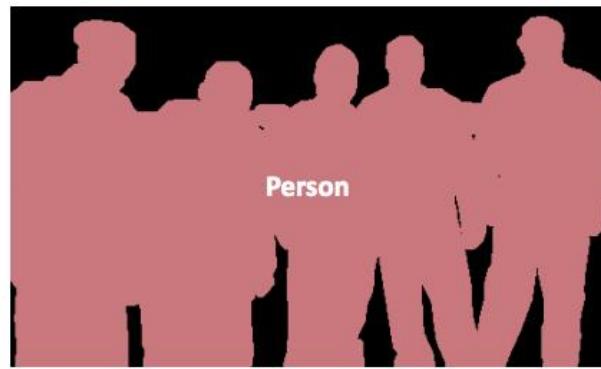
(d) + CRF-RNN

(e) Ground truth

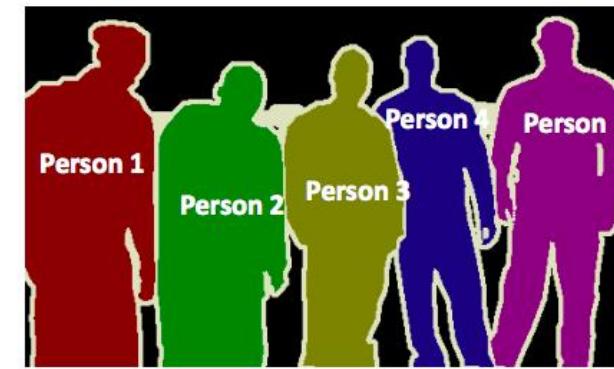
Instance segmentation



Object Detection



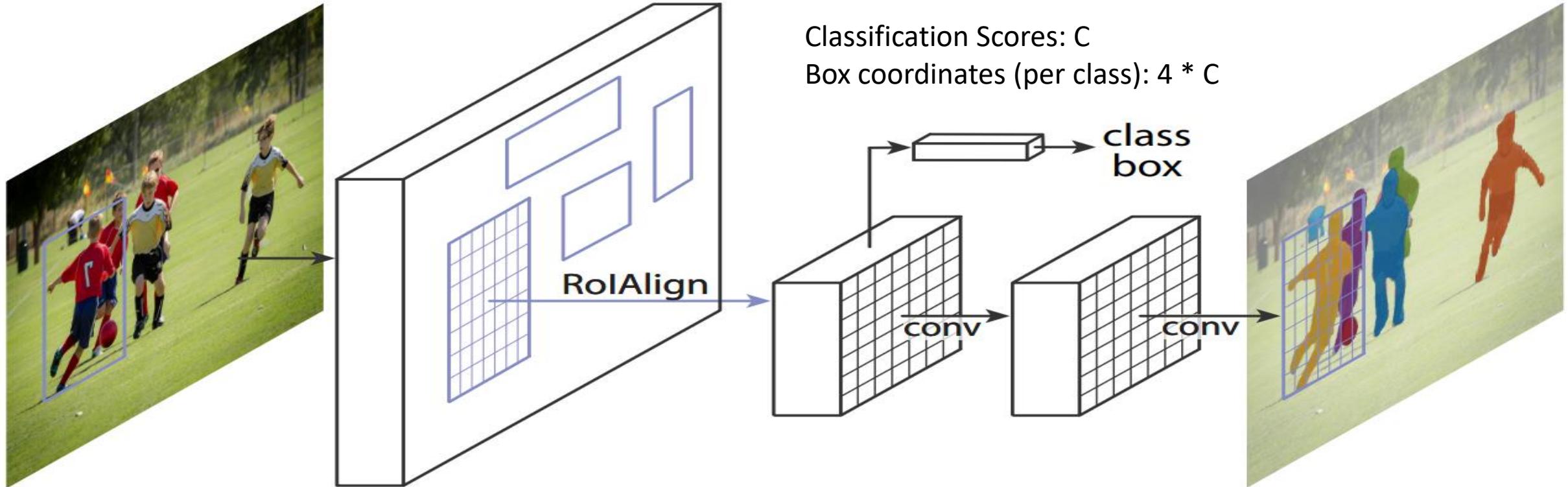
Semantic Segmentation



Instance Segmentation



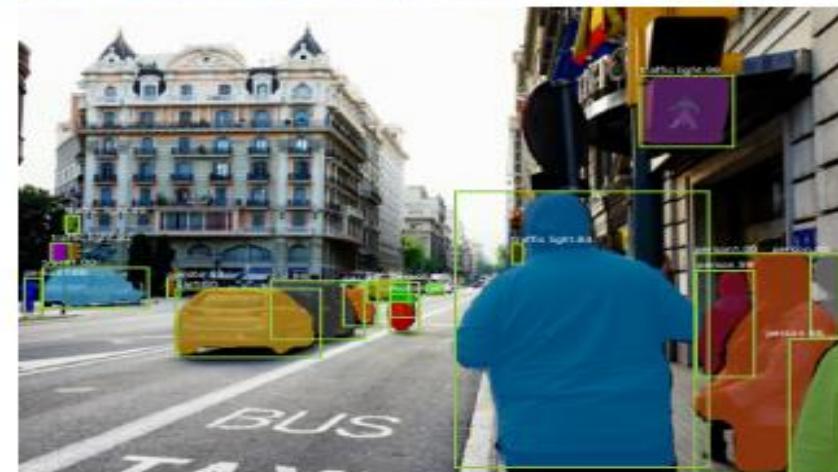
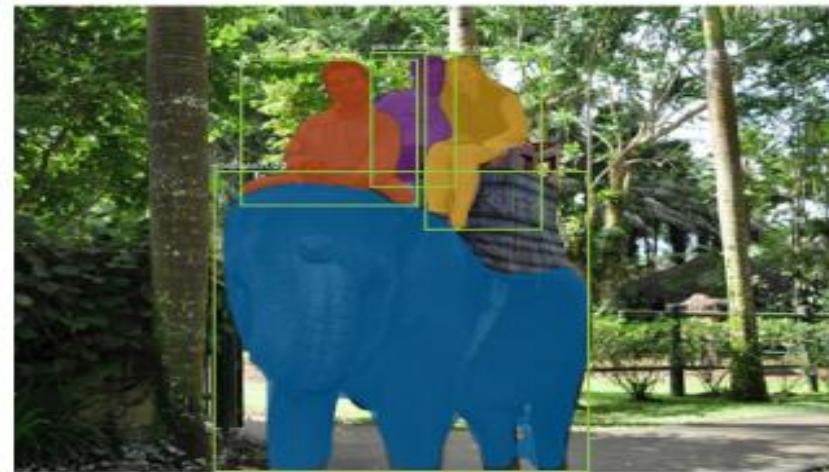
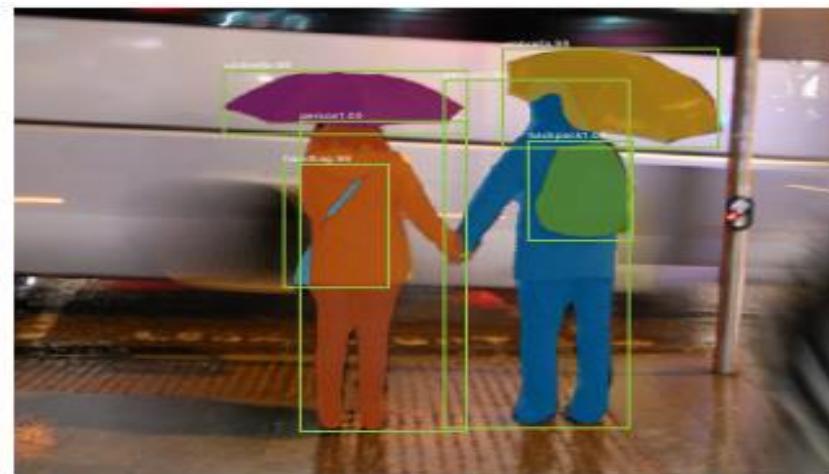
Instance Segmentation: Mask R-CNN (He et al. ICCV 2017)



Mask R-CNN - extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.

Instance Segmentation: Mask R-CNN (He et al. ICCV 2017)



Mask R-CNN results on the COCO test set. These results are based on ResNet-101, achieving a mask AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

Instance segmentation results on COCO

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	37.1	60.0	39.4	16.9	39.9	53.5

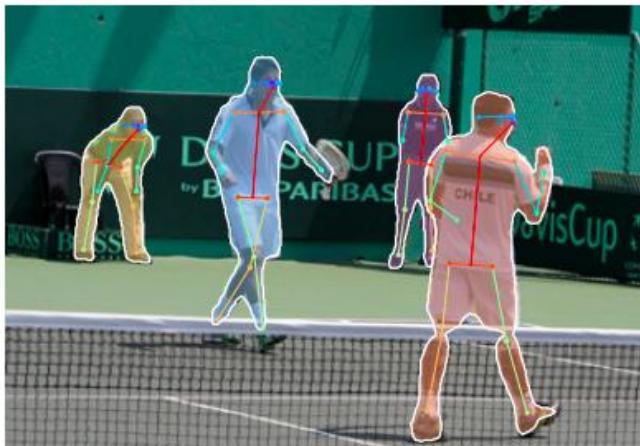
AP at different IoU
thresholds

AP for different size
instances

Other dense prediction problems

Keypoint prediction

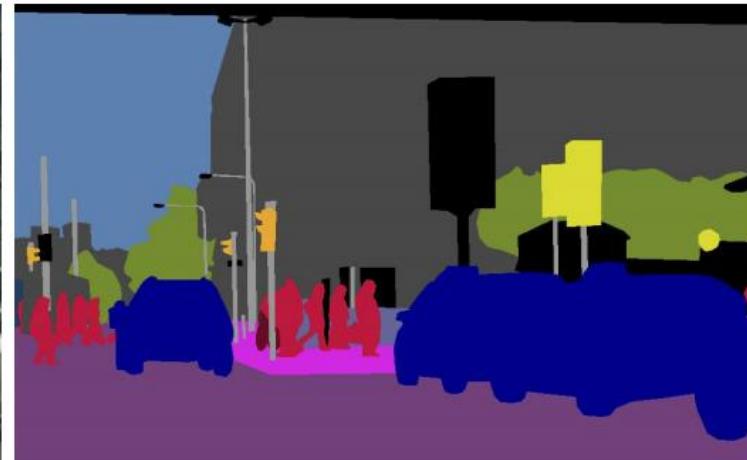
- Given K keypoints, train model to predict $K m \times m$ one-hot maps with cross-entropy losses over m^2 outputs



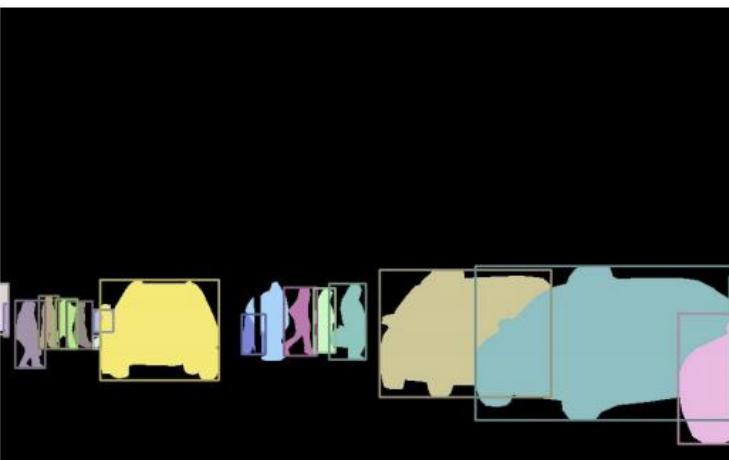
Panoptic Segmentation



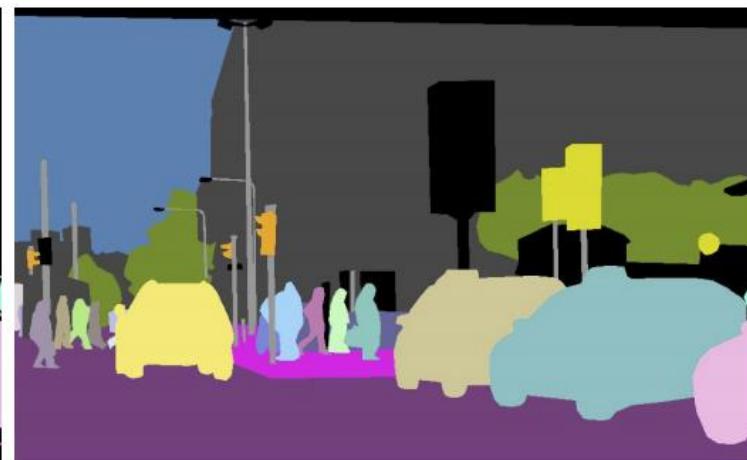
(a) image



(b) semantic segmentation



(c) instance segmentation



(d) panoptic segmentation

Panoptic feature pyramid networks

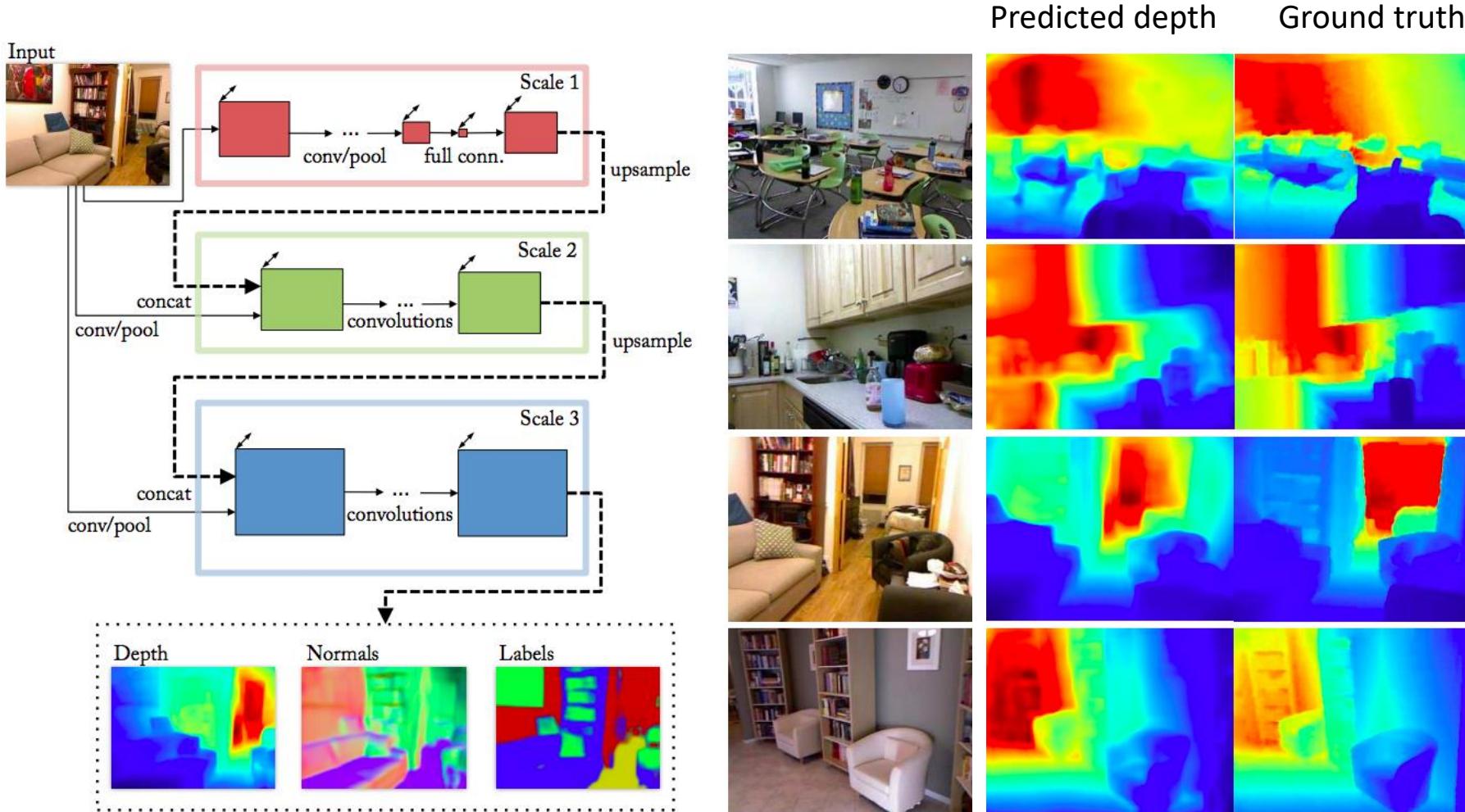


Figure 2: Panoptic FPN results on COCO (top) and Cityscapes (bottom) using a single ResNet-101-FPN network.

Even more dense prediction problems

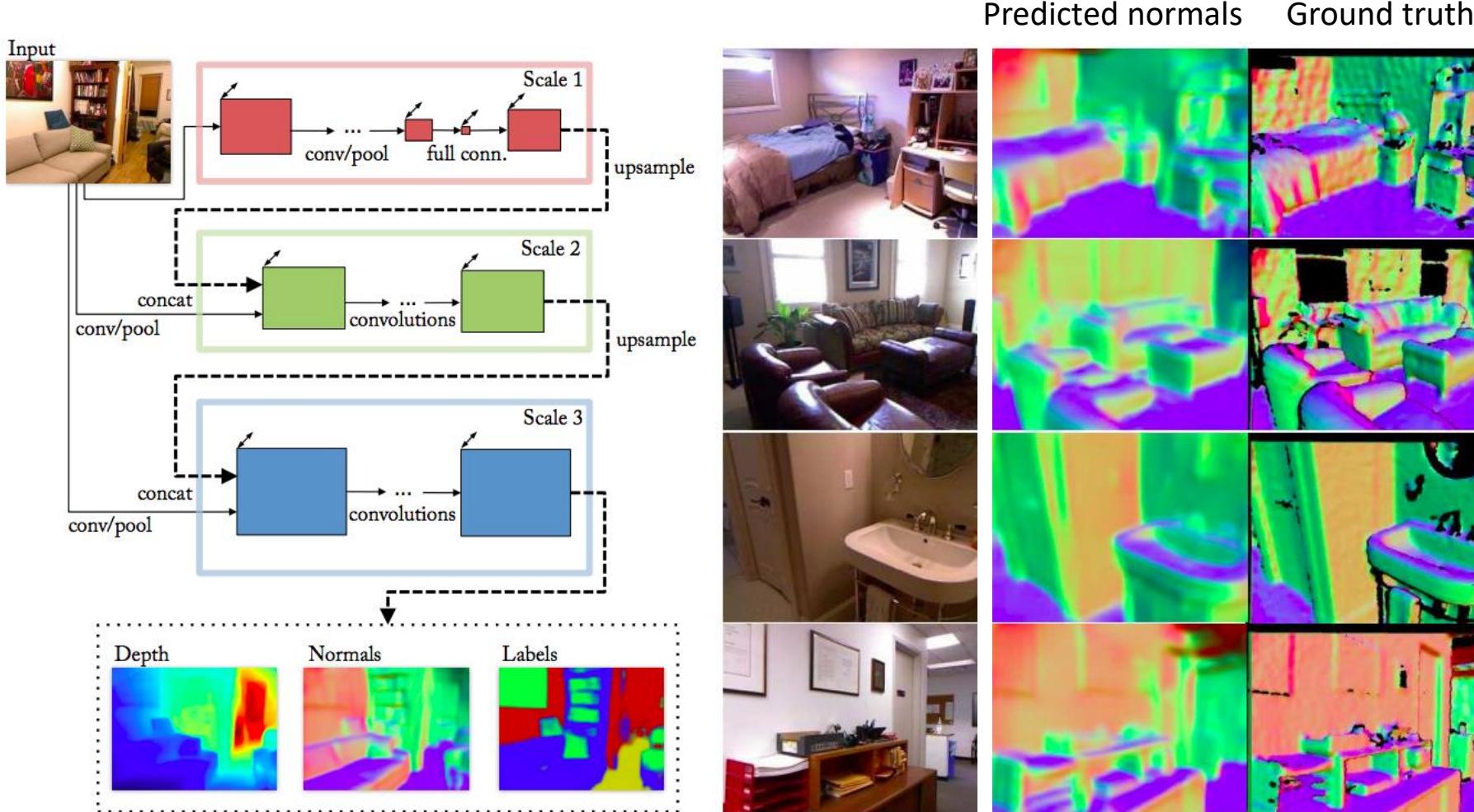
- Depth estimation
- Surface normal estimation
- Colorization
-

Depth and normal estimation



D. Eigen and R. Fergus, [Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture](#), ICCV 2015

Depth and normal estimation



D. Eigen and R. Fergus, [Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture](#), ICCV 2015

Estimation of everything at the same time

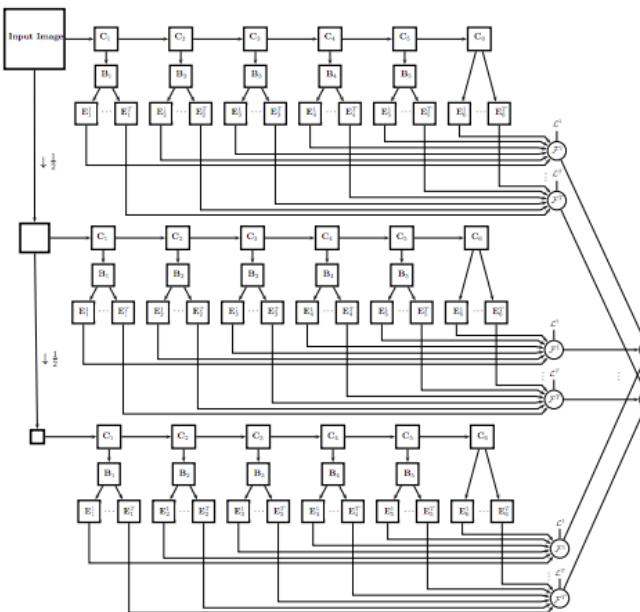
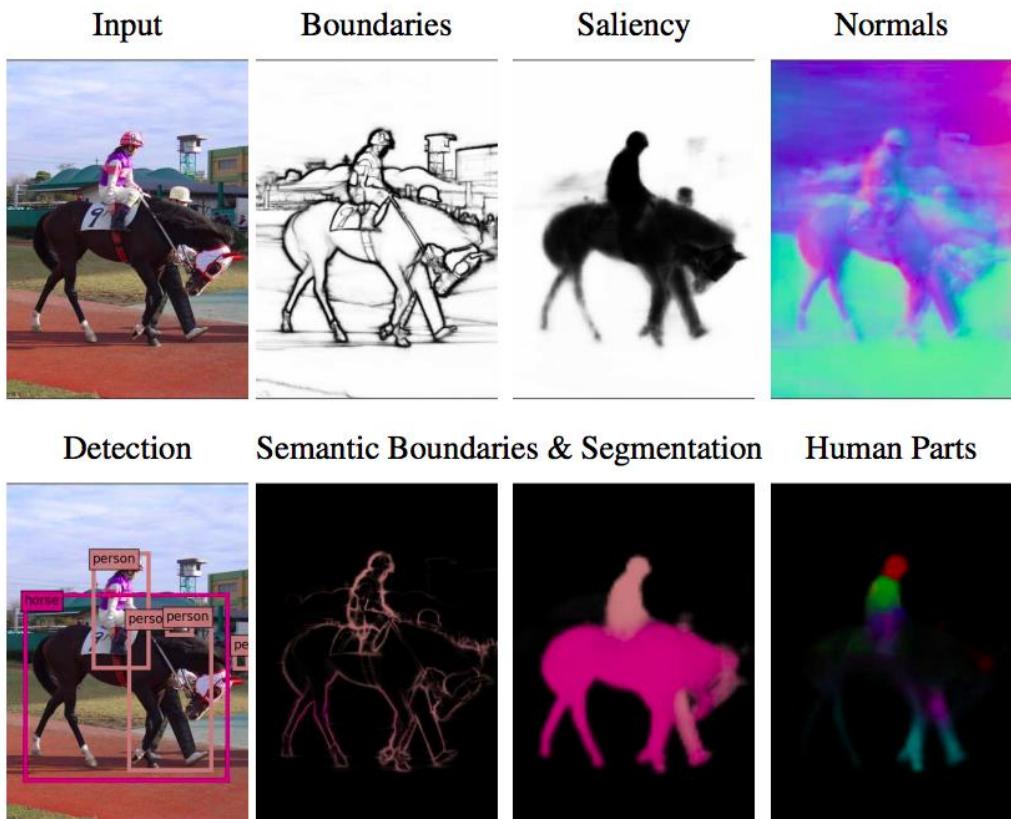
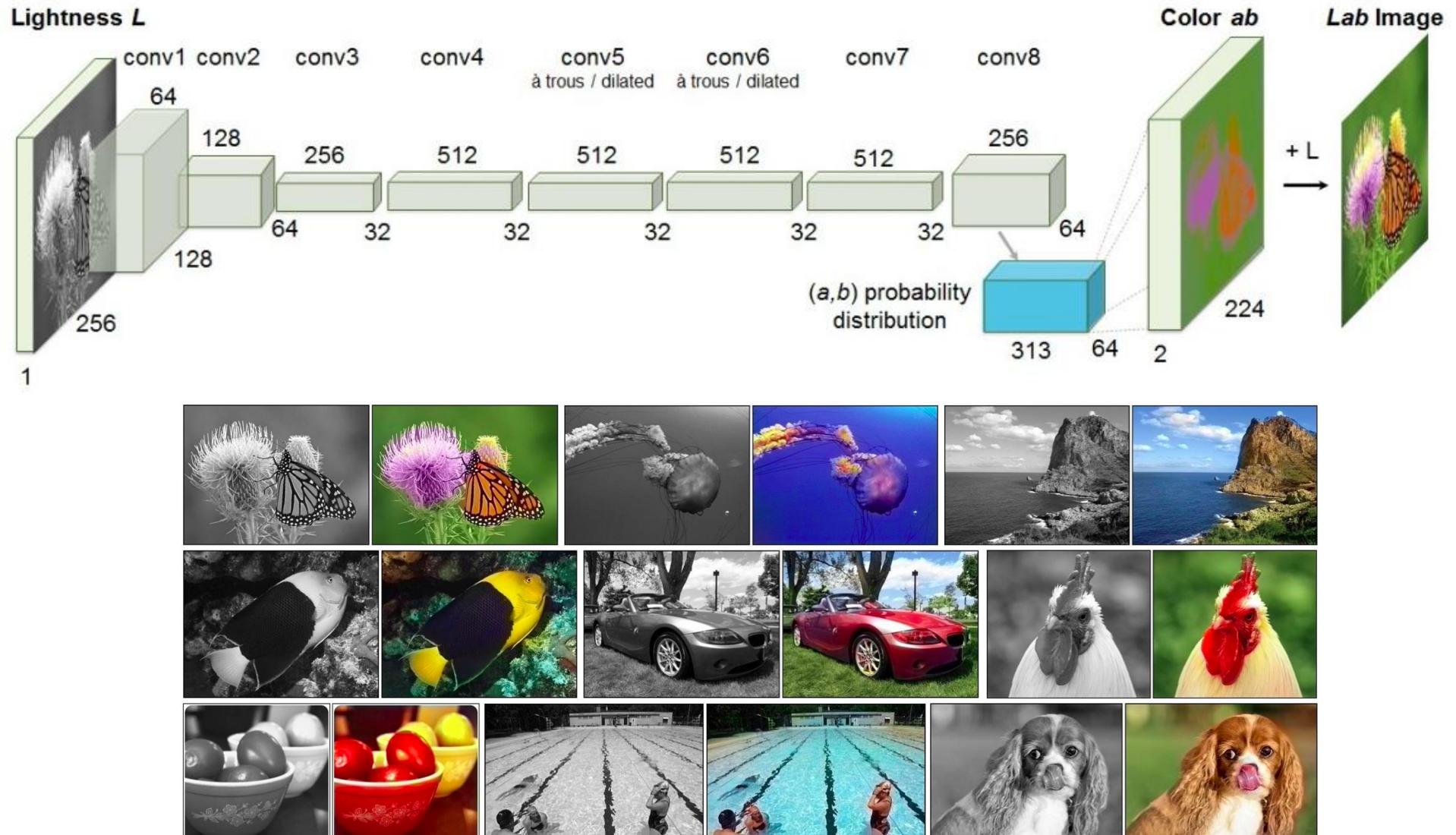


Figure 2: UberNet architecture: an image pyramid is formed by successive down-sampling operations, and each image is processed by a CNN with tied weights; the responses of the network at consecutive layers (C_i) are processed with Batch Normalization (B_i) and then fed to task-specific skip layers (E_i^t); these are combined across network layers (\mathcal{F}^t) and resolutions (\mathcal{S}^t) and trained using task-specific loss functions (\mathcal{L}^t), while the whole architecture is jointly trained end-to-end. For simplicity we omit the interpolation and detection layers mentioned in the text.

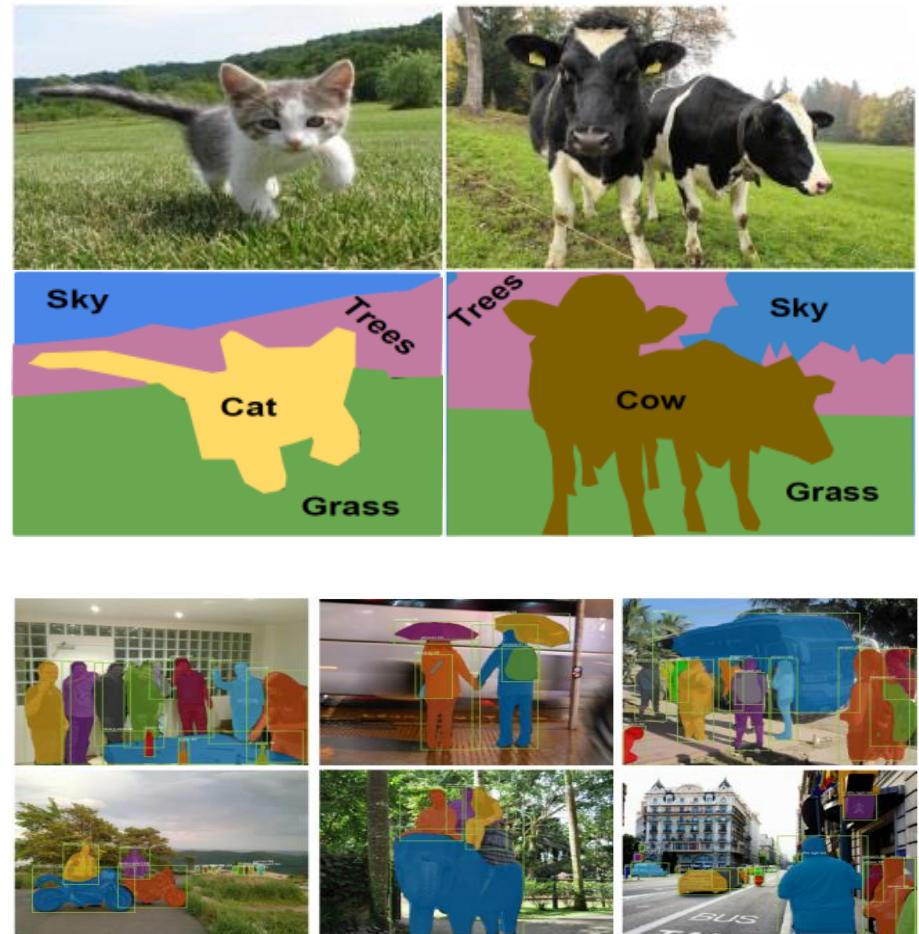
I. Kokkinos, [UberNet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision using Diverse Datasets and Limited Memory](#),

Colorization



Things to remember

- Semantic Segmentation
 - Sliding window inefficient
 - Fully Convolutional: Convolutions at original image resolution is very expensive
 - Fully Convolutional: Down sampling and Up sampling
 - Unsampling usning Unpooling and Transpose Convolution
 - DeconvNet: deeper network
- Instance Segmentation
 - Mak-RCNN: Extension of faster RCNN

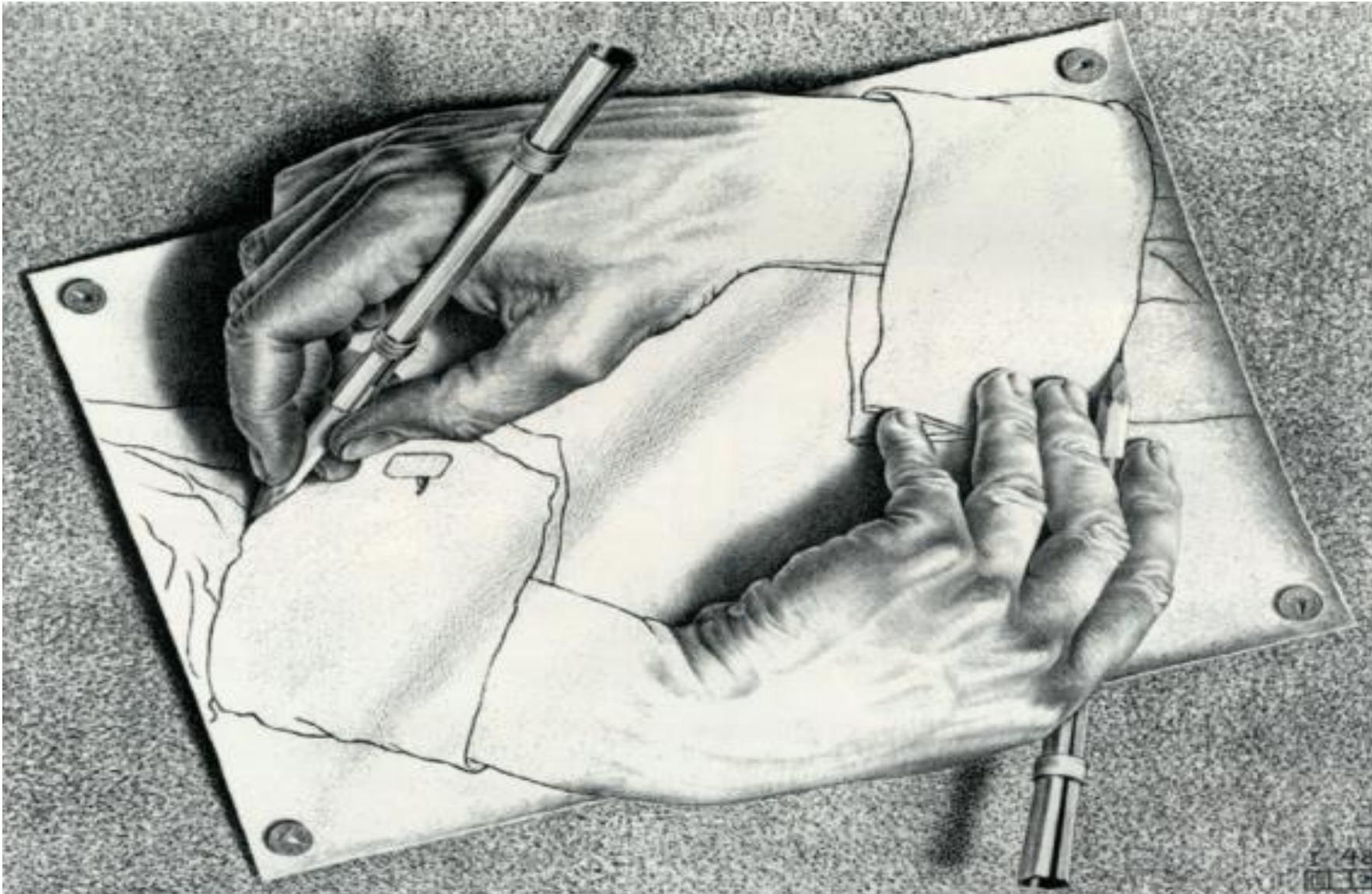


Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

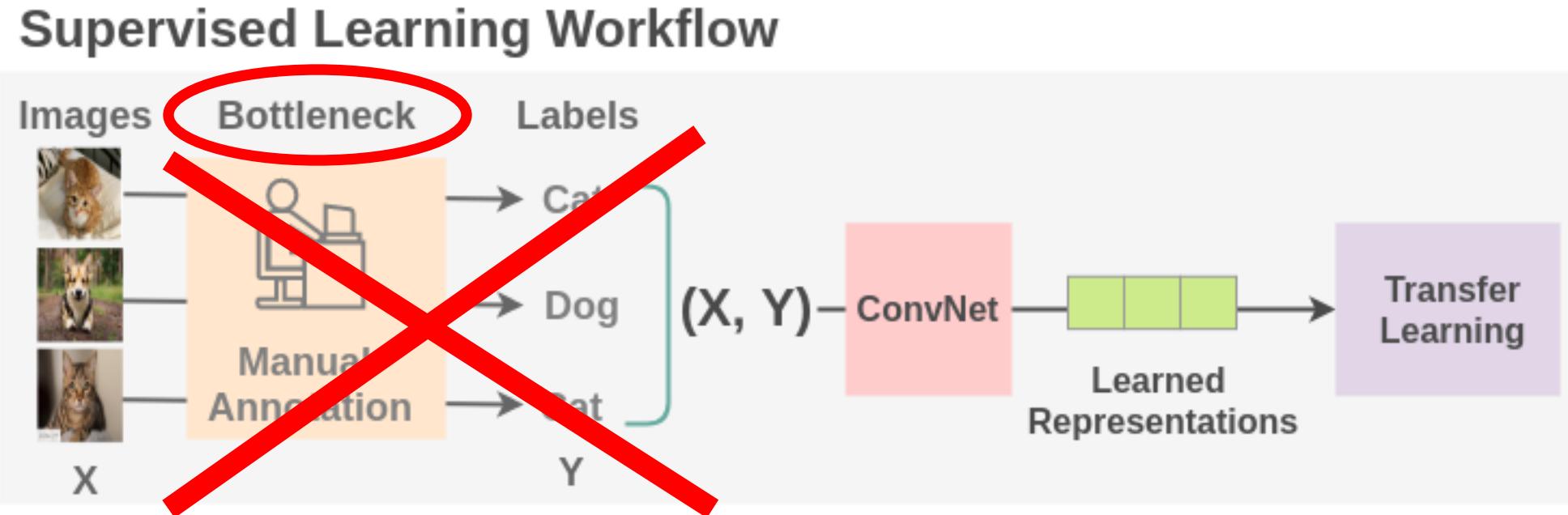
Self-supervised learning



M.C. Escher, *Drawing Hands* (1948) – via A. Efros

Motivation

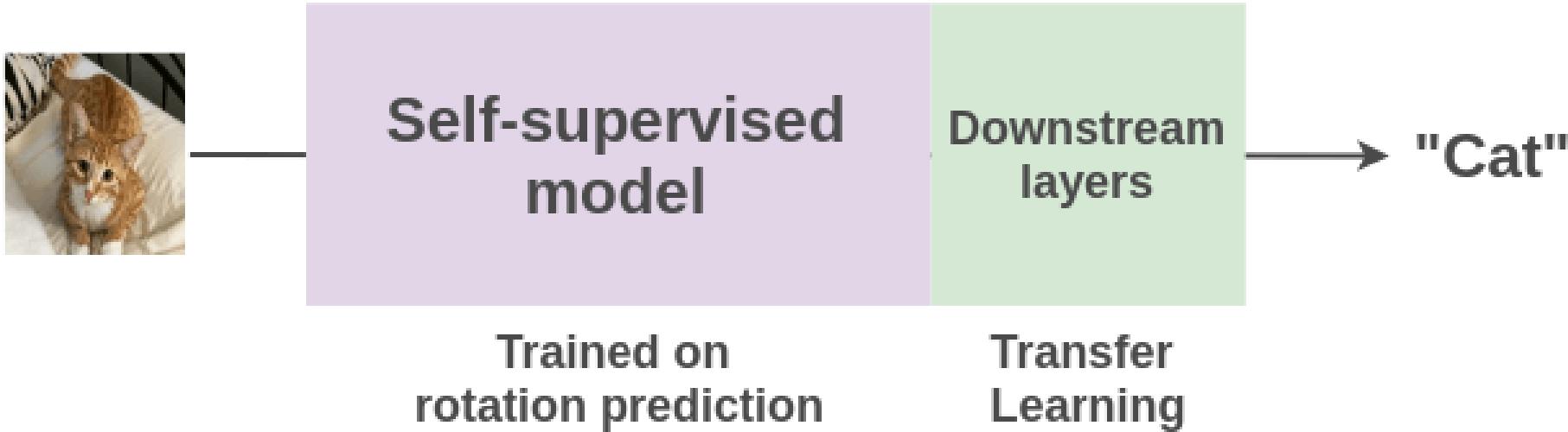
- Overcoming reliance on *supervised pre-training*



Can we design the task in such a way that we can generate virtually unlimited labels from our existing images and use that to learn the representations?

[Figure source](#)

Motivation



Once we learn representations from these millions of images, we can use transfer learning to fine-tune it on some supervised task like image classification of cats vs dogs with very few examples.

Self-supervised vs. unsupervised learning

- The terms are sometimes used interchangeably in the literature, but self-supervised learning is a particular kind of unsupervised learning
- **Self-supervised learning:** the learner “makes up” labels from the data and then solves a supervised task
- **Unsupervised learning:** any kind of learning without labels
 - Clustering and quantization
 - Dimensionality reduction, manifold learning
 - Density estimation
 - Learning to sample

Types of self-supervised learning

Data prediction



Transformation prediction



Contrastive learning



Self-supervised learning: Outline

- Data prediction
 - Colorization, Superresolution, Inpainting, Cross channel encoding
- Transformation prediction
 - Context prediction, jigsaw puzzle solving, rotation prediction
- Automatic Label Generation
 - Image clustering, Synthetic imagery
- Contrastive learning
 - PIRL, MoCo, SimCLR, SWaV
- Self-supervision beyond still images
 - Audio, video, language

Self-supervised learning: Outline

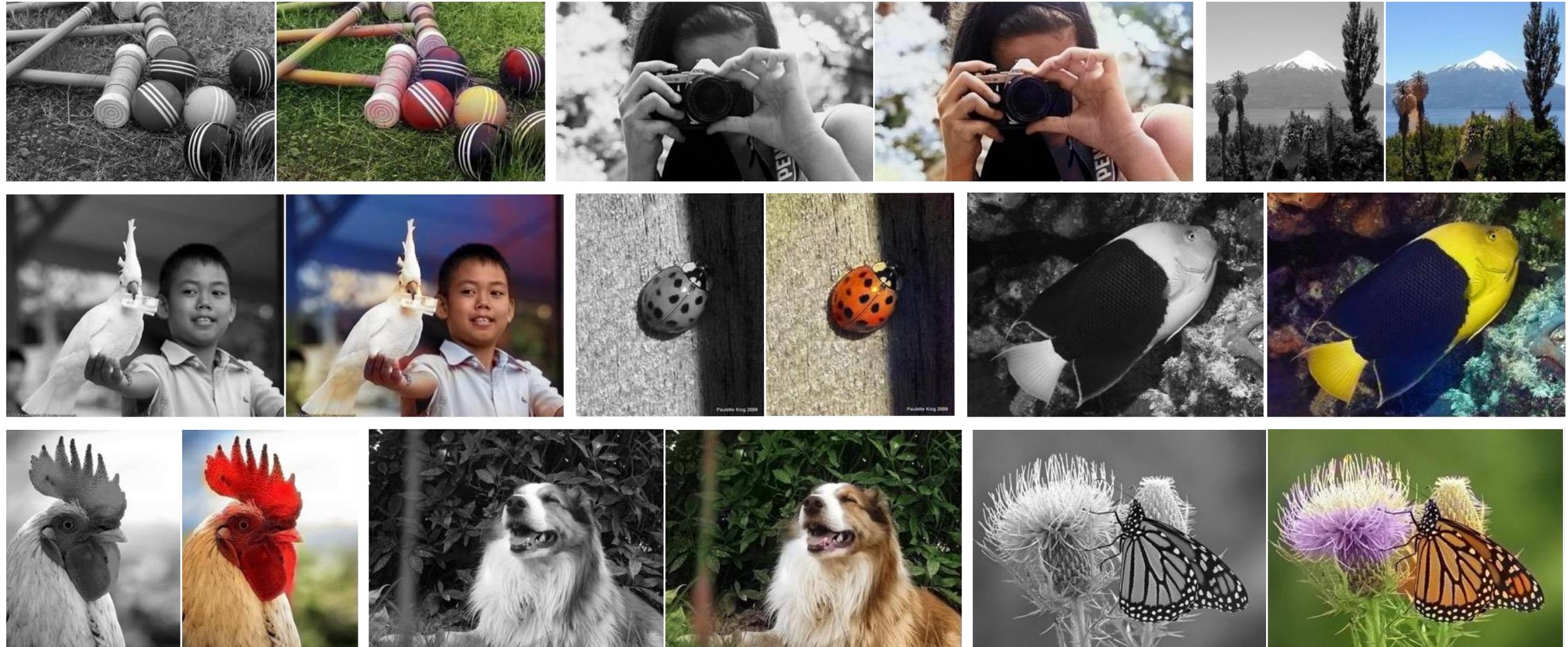
- Data prediction
 - Colorization, Superresolution, Inpainting, Cross channel encoding

Self-Supervision as data prediction

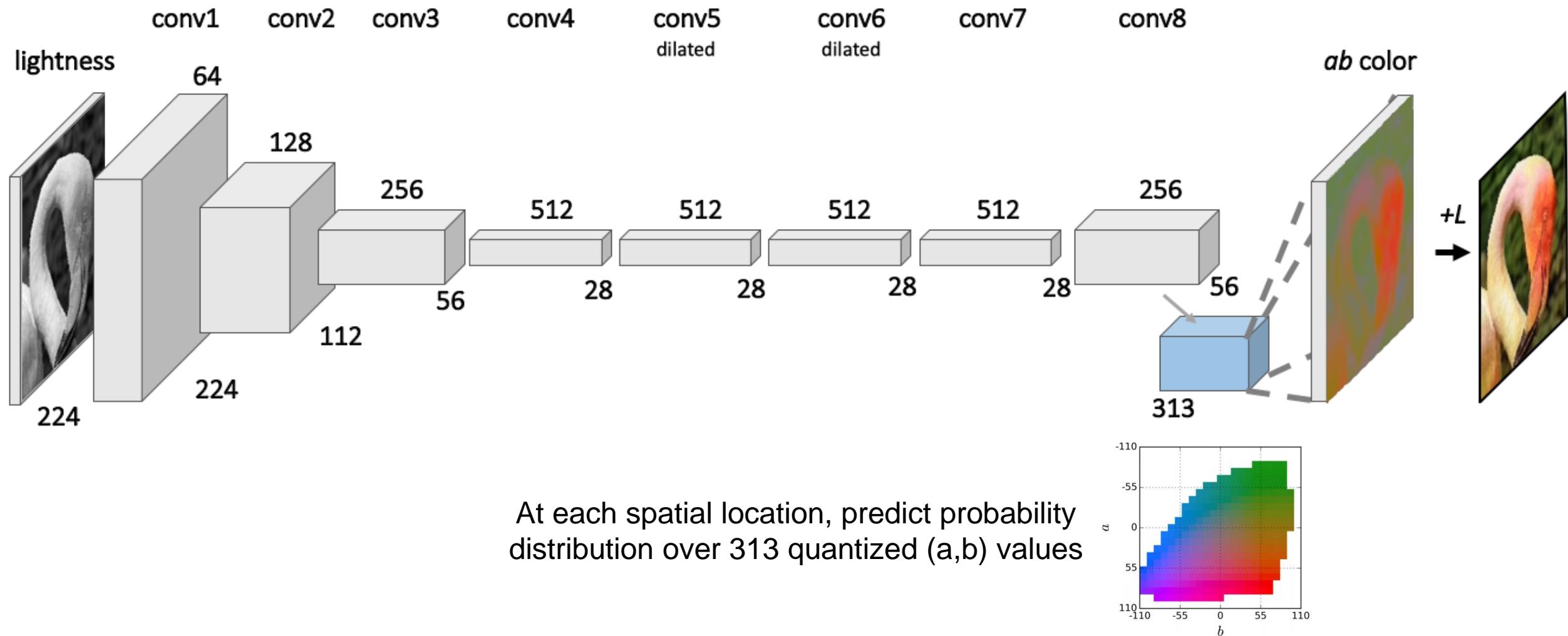


- Colorization
- Superresolution
- Inpainting
- Cross-channel encoding

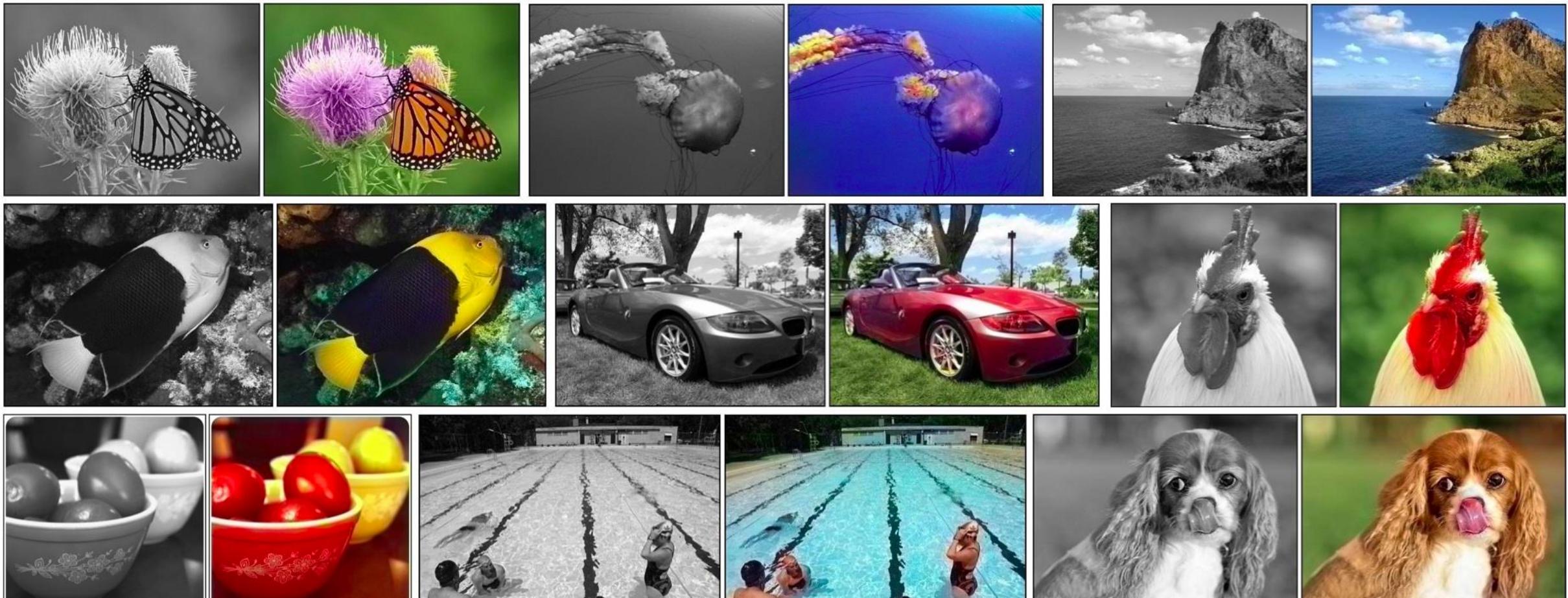
Colorization



Colorization: Architecture



Colorization: Results



Failure Cases



© 2002 Rainmaker

Inherent Ambiguity



Grayscale

Inherent Ambiguity



Prediction



Ground Truth

Self-Supervision as data prediction



- Colorization
- Superresolution
- Inpainting
- Cross-channel encoding

Image Superresolution

What if we prepared training pairs of (small, upscaled) images by downsampling millions of images we have freely available?

Training Data Generation for Superresolution



...



Make 2x smaller



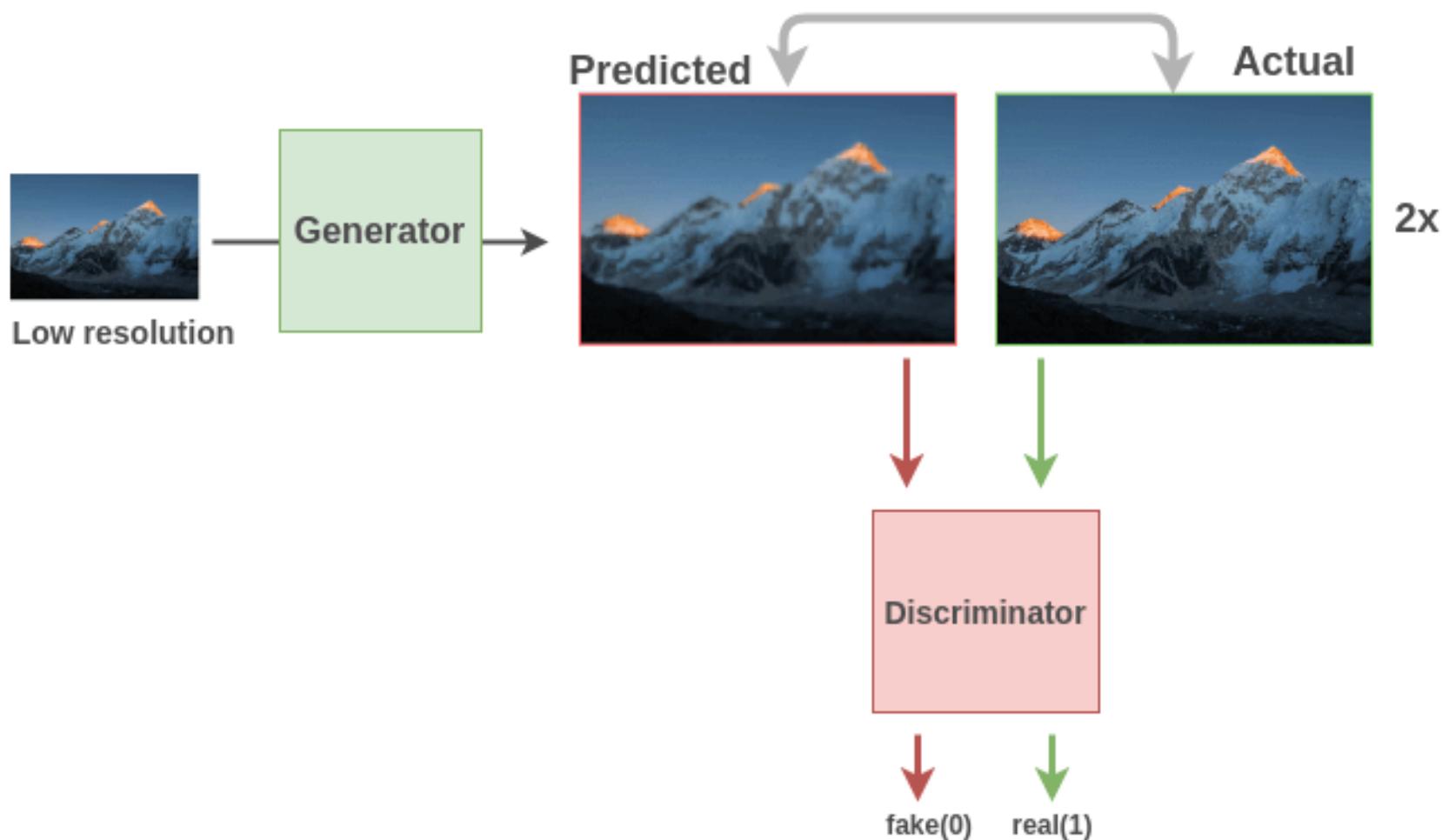
...



[Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network](#)

Image Superresolution

SRGAN



[Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network](#)

Self-Supervision as data prediction



- Colorization
- Superresolution
- Inpainting
- Cross-channel encoding

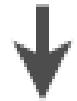
Image Inpainting

What if we prepared training pairs of (corrupted, fixed) images by randomly removing part of images?

Image Inpainting Data Generation



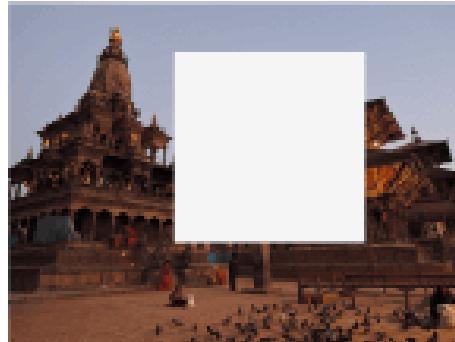
...



random missing region



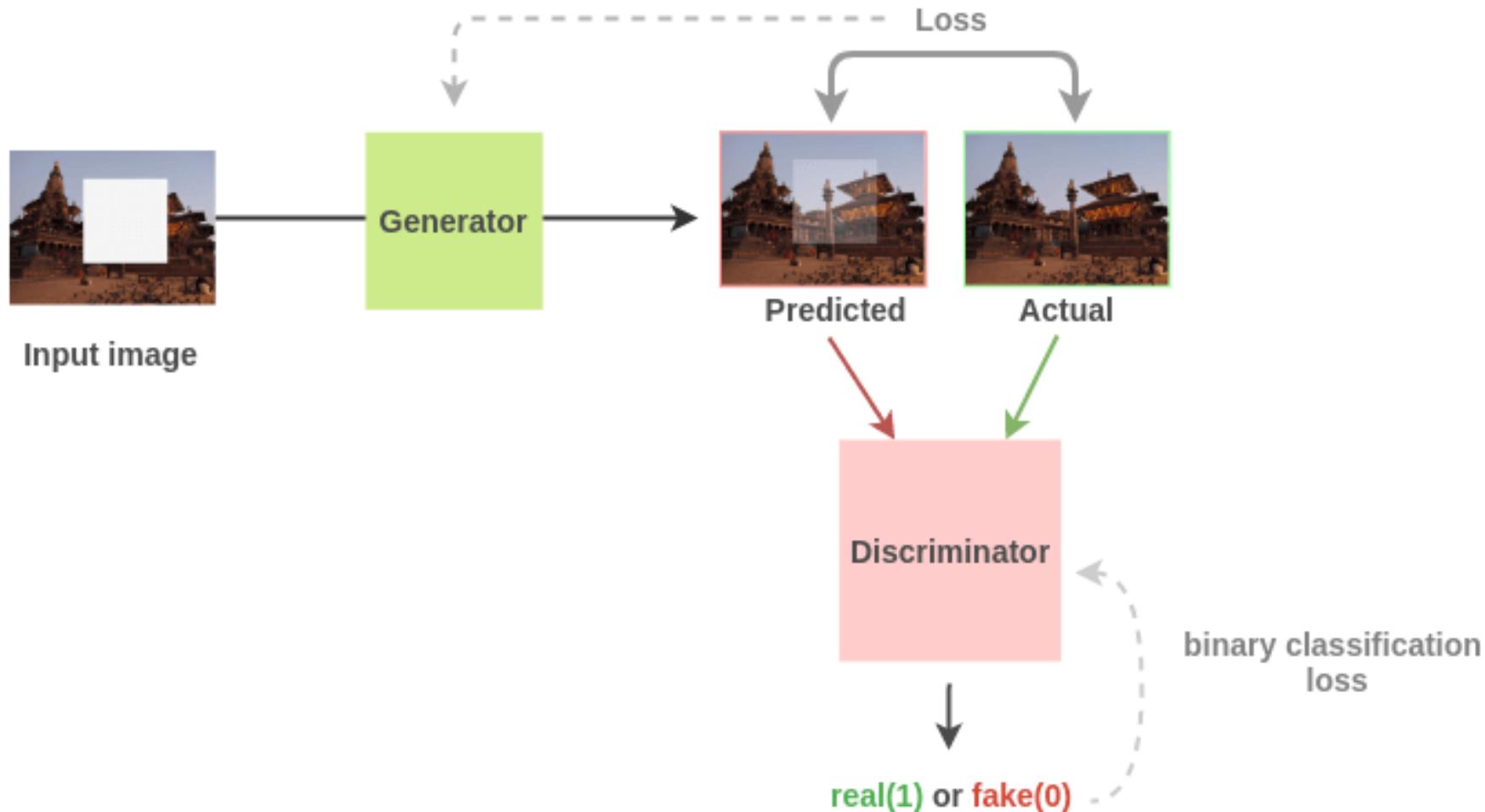
...



[Context encoders: Feature learning by inpainting](#)

Image Inpainting

Image Inpainting



[Context encoders: Feature learning by inpainting](#)

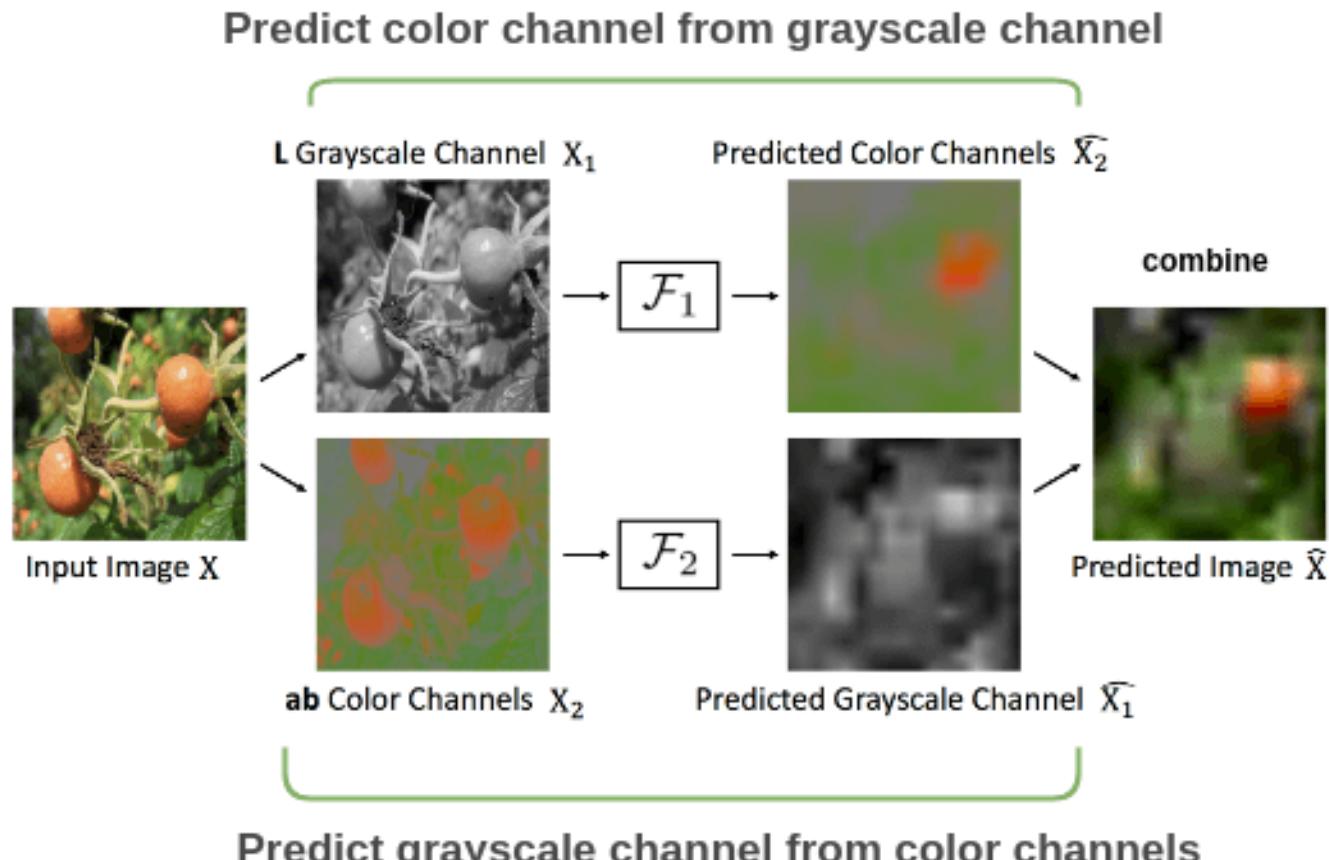
Self-Supervision as data prediction



- Colorization
- Superresolution
- Inpainting
- Cross-channel encoding

Cross-channel encoding

What if we predict one channel of the image from the other channel and combine them to reconstruct the original image?

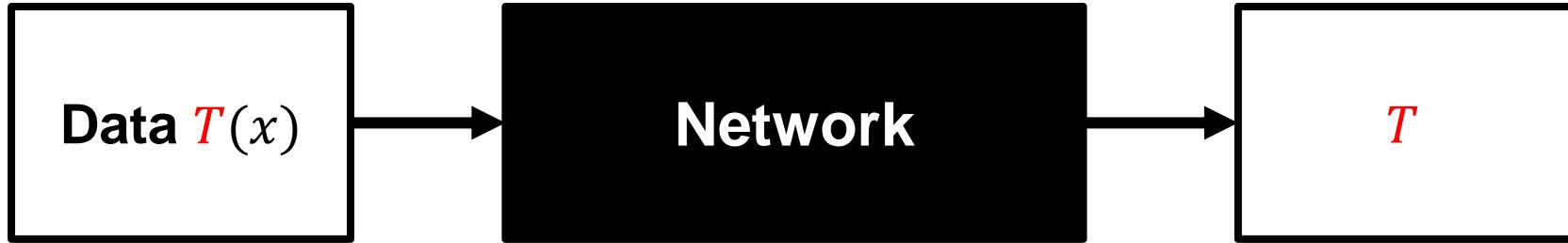


Example adapted from “Split-Brain Autoencoder”

Self-supervised learning: Outline

- Data prediction
 - Colorization, Superresolution, Inpainting, Cross channel encoding
- Transformation prediction
 - Context prediction, jigsaw puzzle solving, rotation prediction

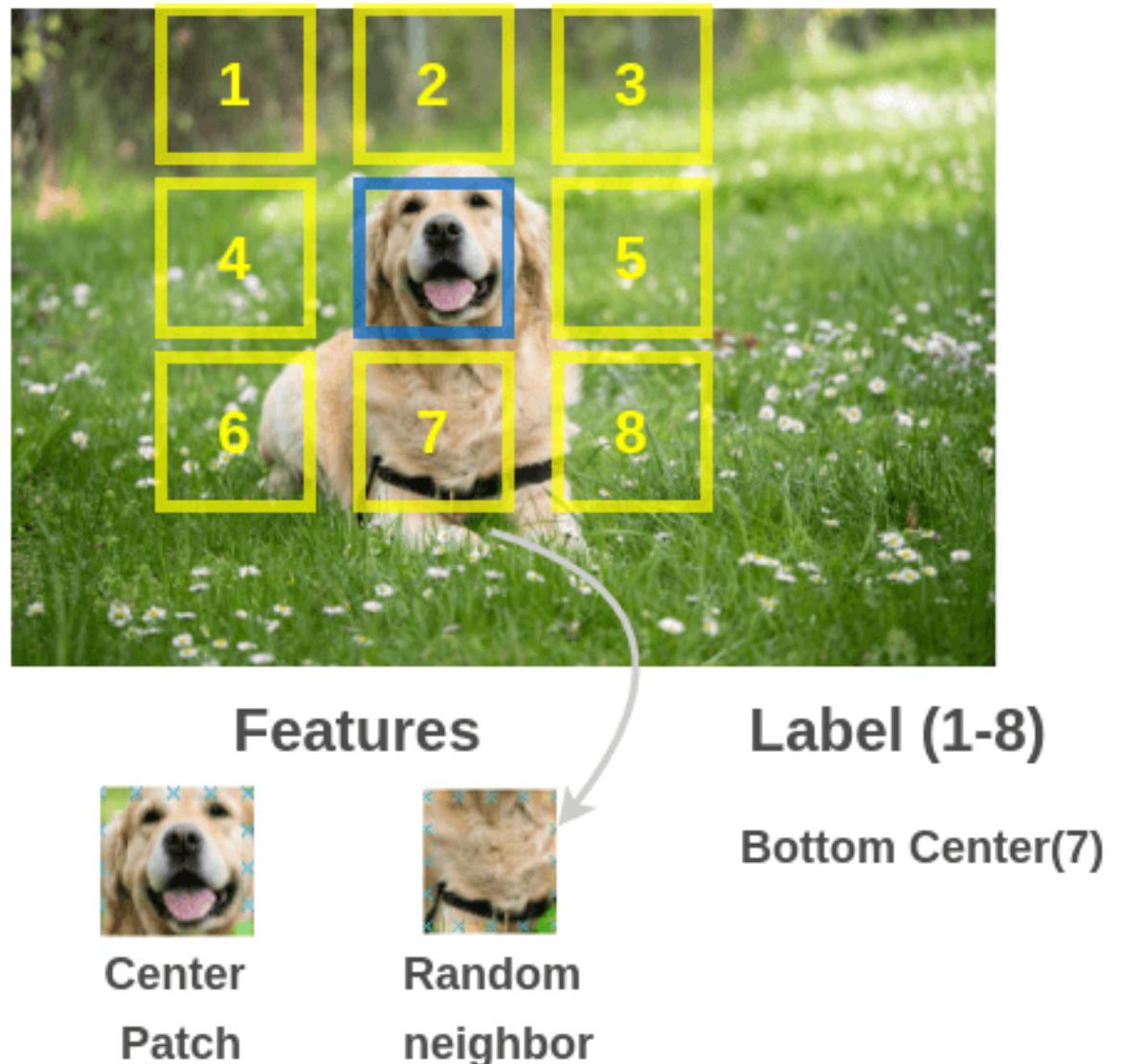
Self-supervision by transformation prediction



- Context prediction
- Jigsaw puzzle solving
- Rotation prediction

Context prediction

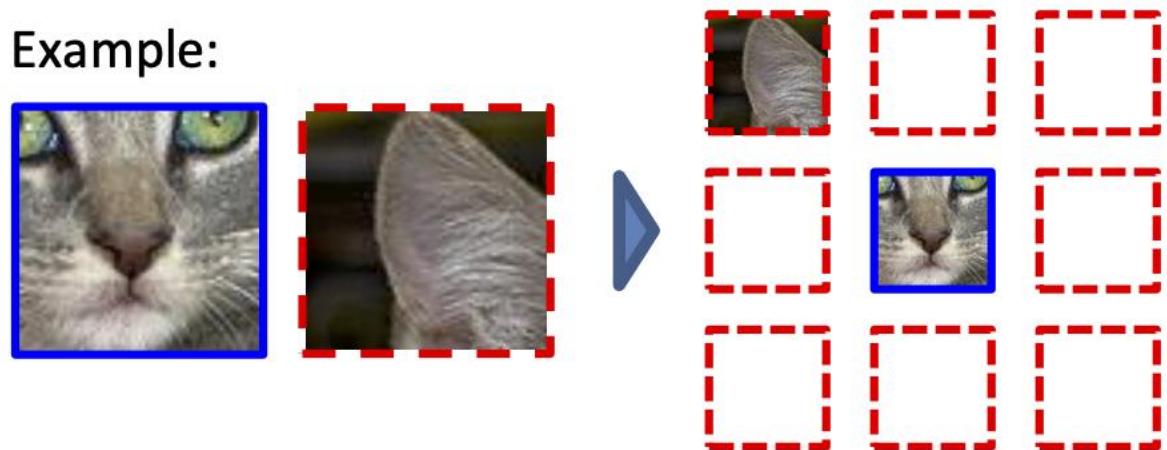
What if we prepared training pairs of (image-patch, neighbor) by randomly taking an image patch and one of its neighbors around it from large, unlabeled image collection?



Context prediction

- *Pretext task:* randomly sample a patch and one of 8 neighbors
- Guess the spatial relationship between the patches

Example:

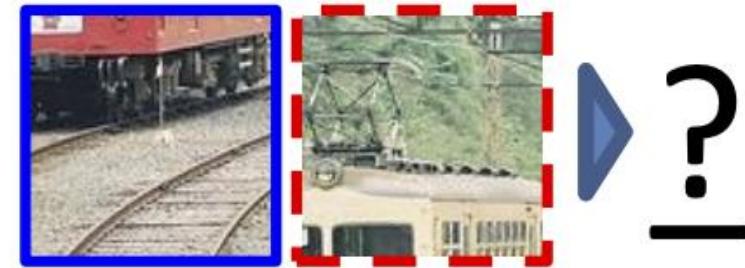


Question 1:



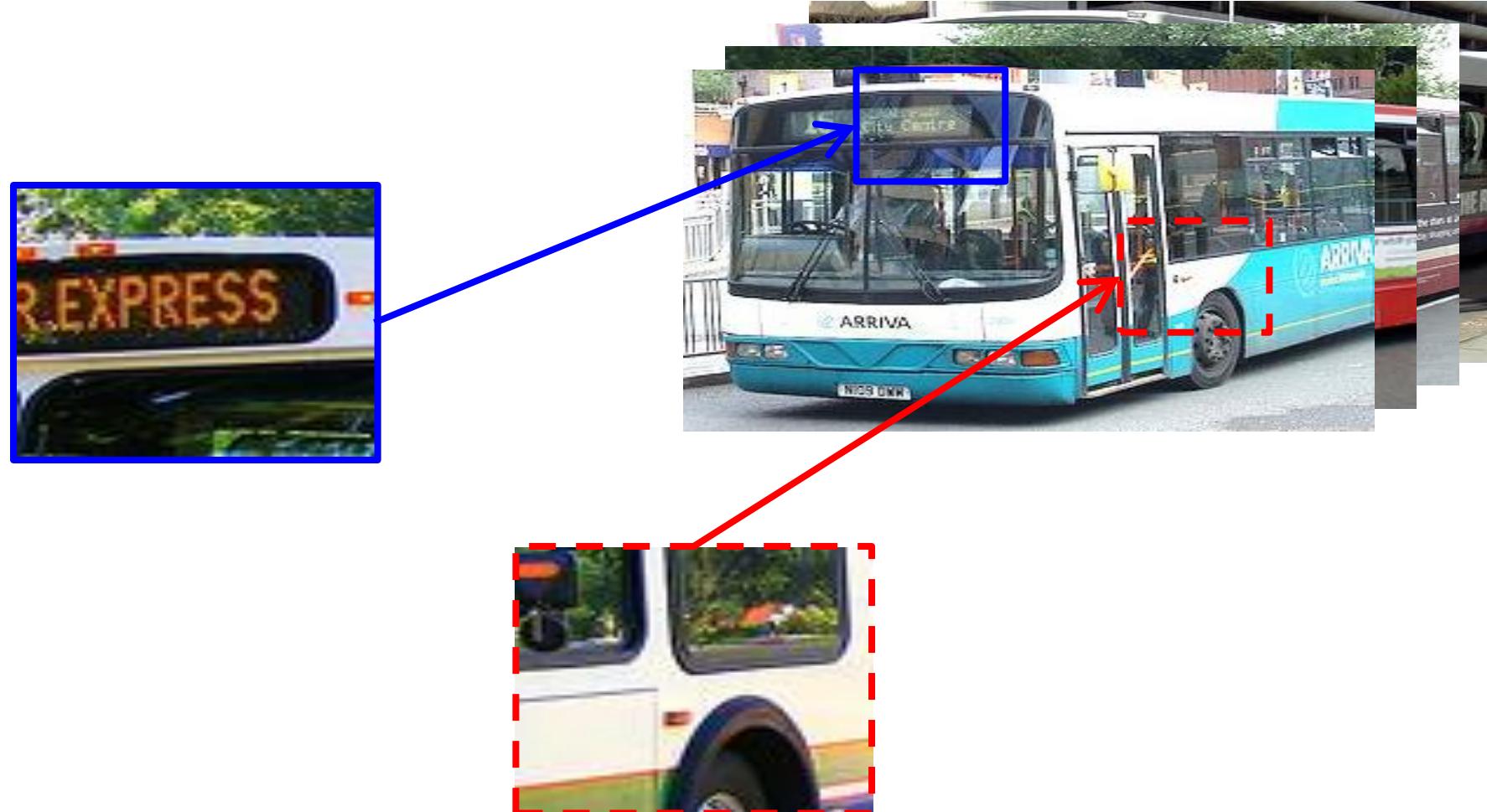
A: Bottom right

Question 2:



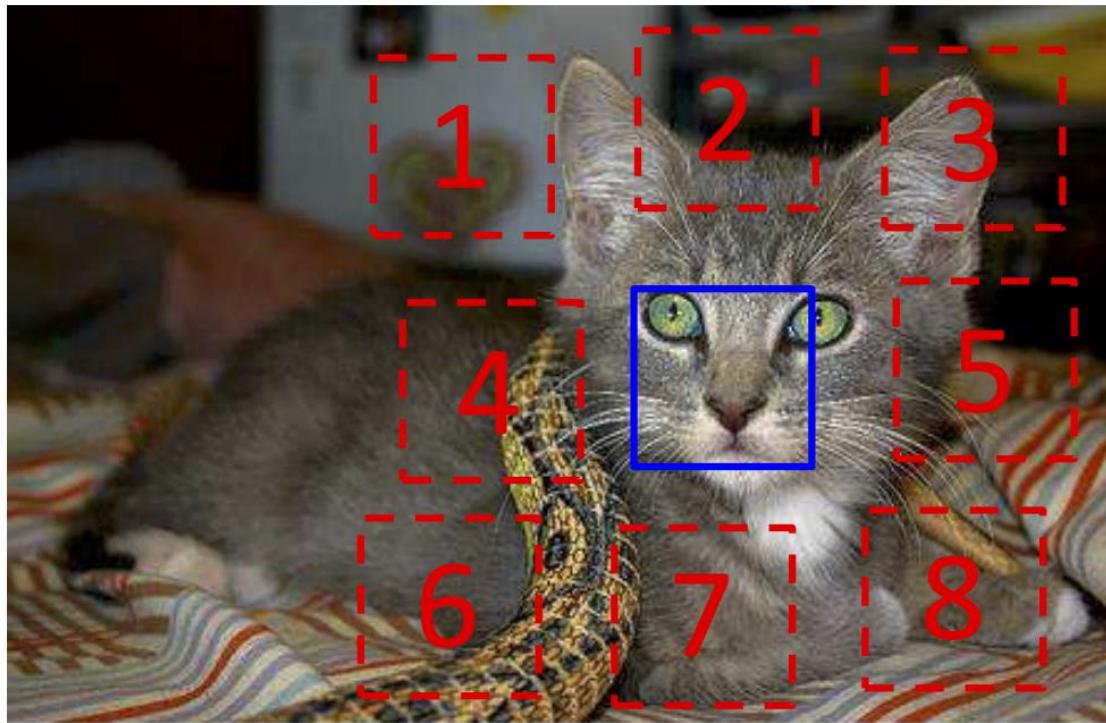
A: Top center

Context prediction: Semantics from a non-semantic task



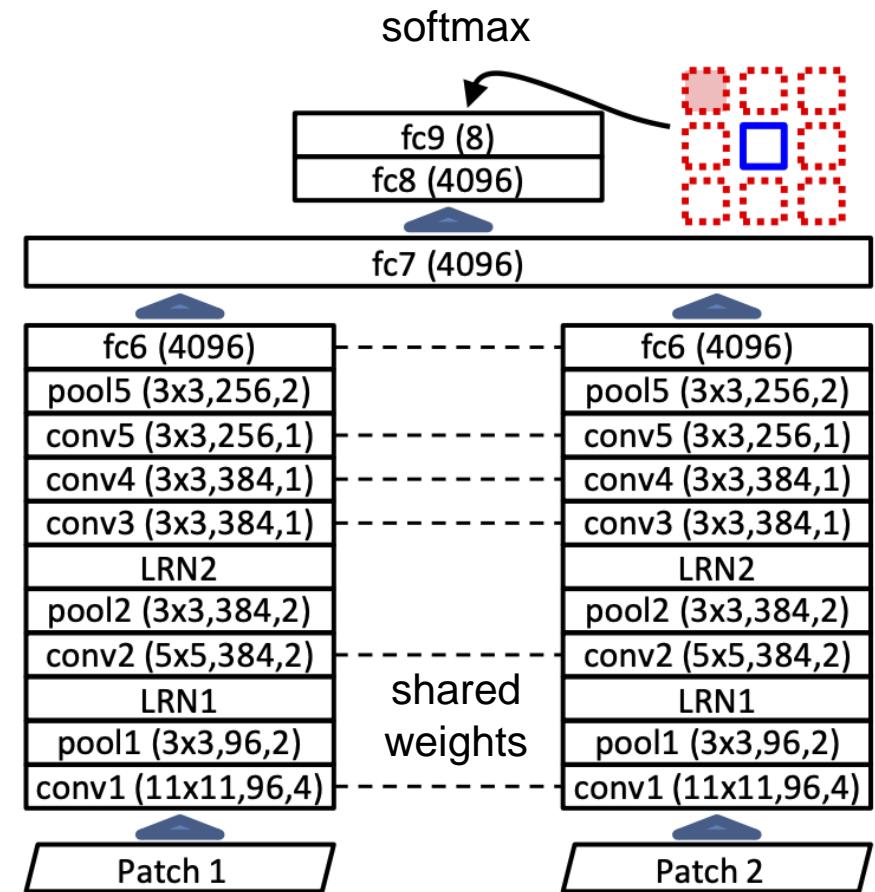
Source: A. Efros

Context prediction: Details



Prevent “cheating”: sample patches with gaps, pre-process to overcome chromatic aberration

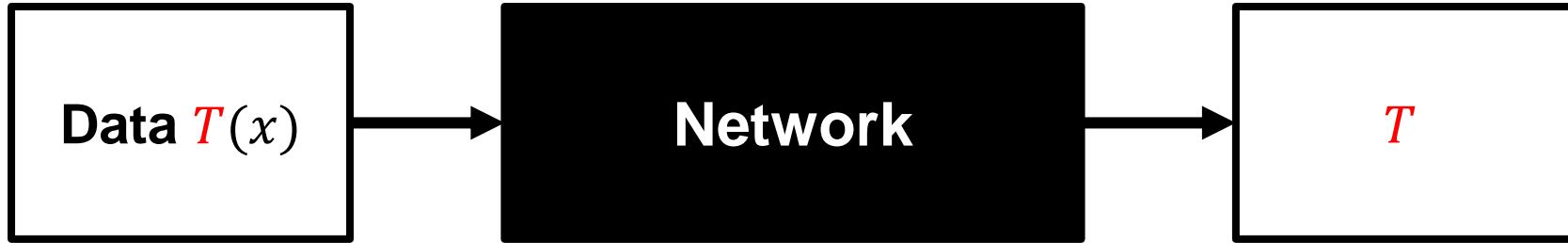
AlexNet-like architecture



Context prediction: Results

- Use learned weights in R-CNN model to perform detection on PASCAL VOC 2007
- Unsupervised pre-training is 5% mAP better than training from scratch, but still 8% below pre-training with ImageNet label supervision

Self-supervision by transformation prediction

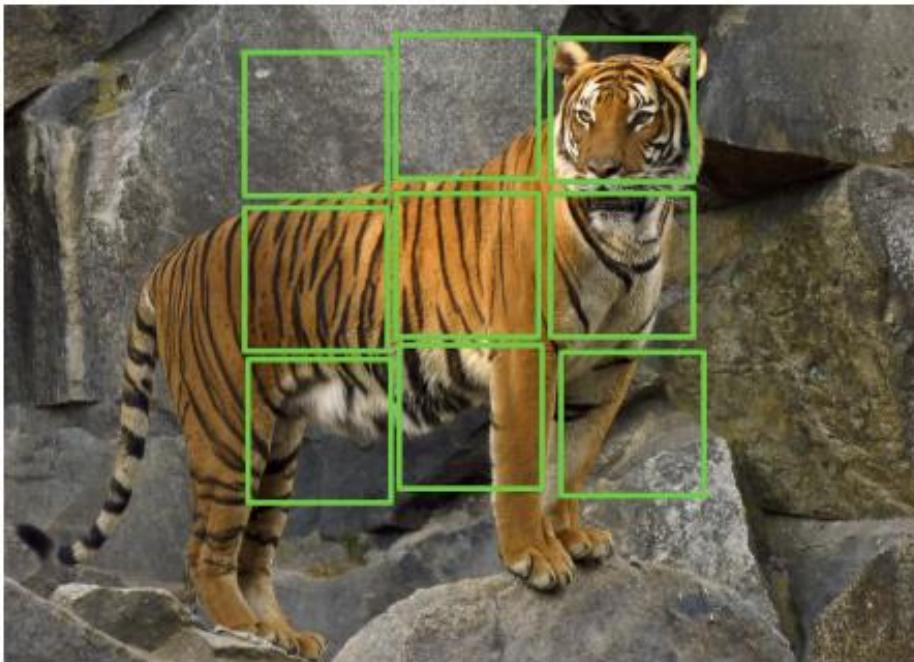


- Context prediction
- Jigsaw puzzle solving
- Rotation prediction

Jigsaw puzzle solving

What if we prepared training pairs of (shuffled, ordered) puzzles by randomly shuffling patches of images?

Crop out tiles



Shuffle



Pretext task: reassemble



Claim: jigsaw solving is easier than context prediction, trains faster, transfers better

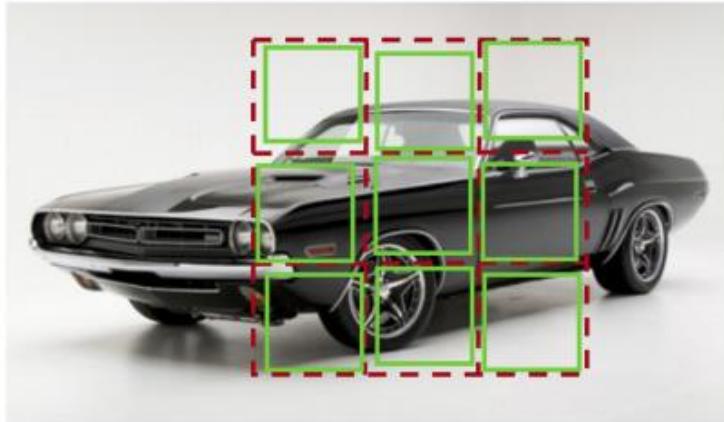
Jigsaw puzzle solving: Details

Possible Shuffles = 362880



```
● ● ●  
from itertools import permutations  
>> x = list(range(9))  
>> len(list(permutations(x, 9)))  
362880
```

Jigsaw puzzle solving: Details

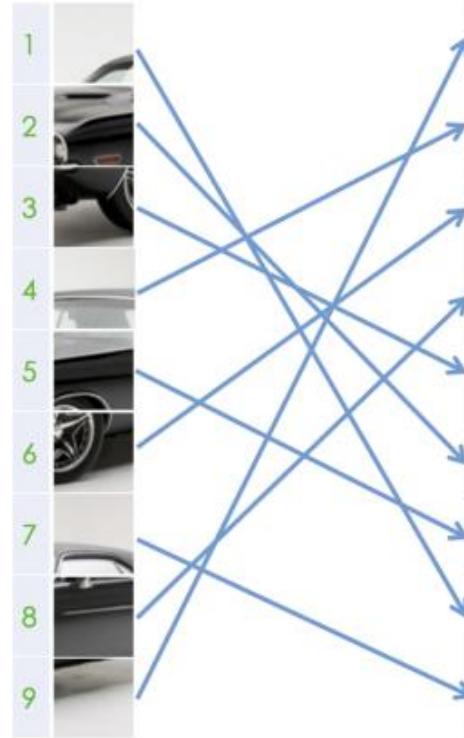


Permutation Set

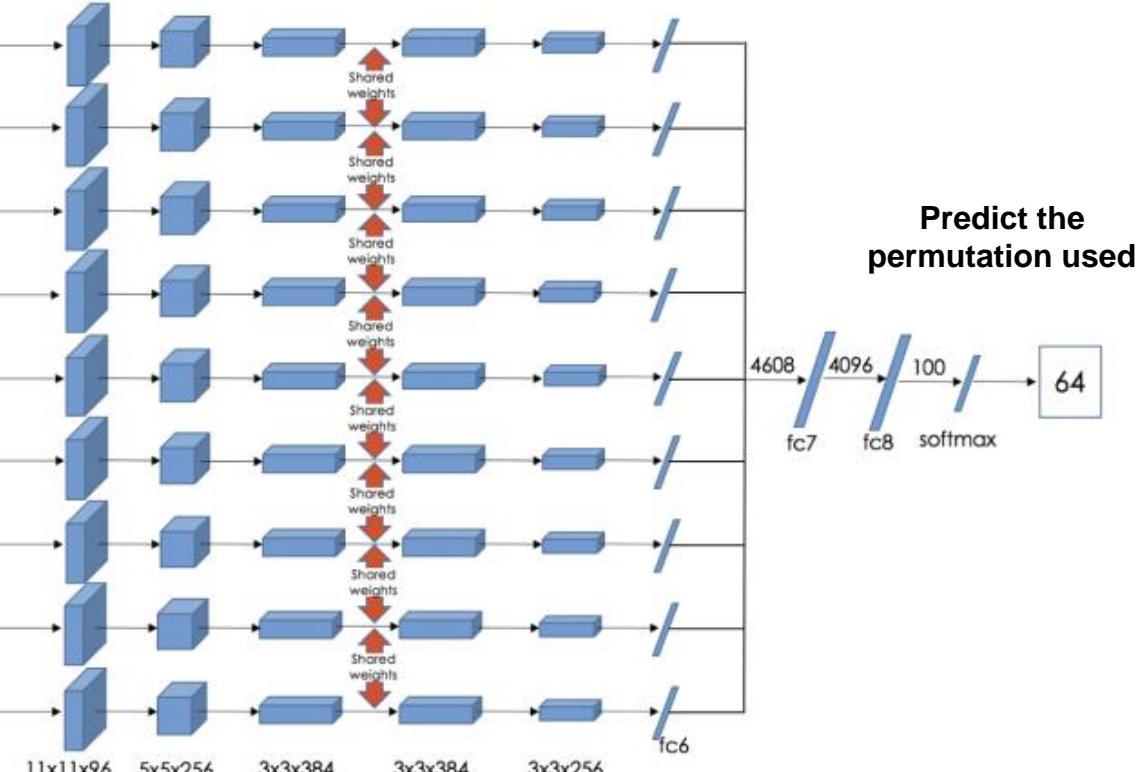
index	permutation
64	9,4,6,8,3,2,5,1,7

Reorder patches according to the selected permutation

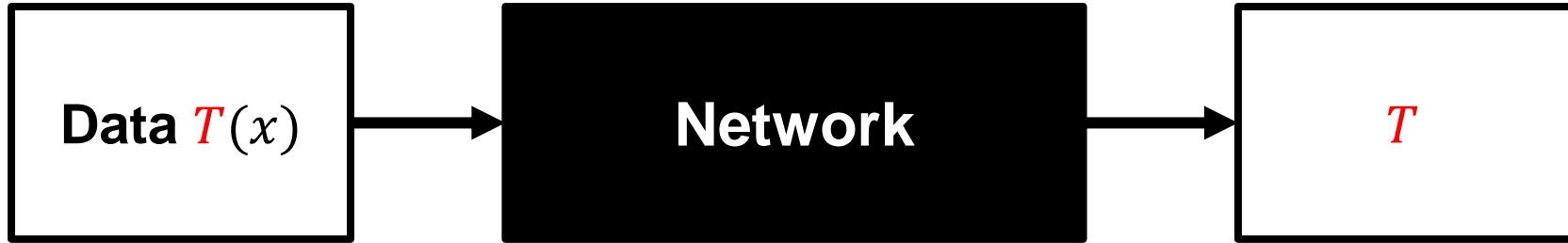
Predetermined set of 100 permutations (out of 362,880 possible)



Context free network (CFN)



Self-supervision by transformation prediction



- Context prediction
- Jigsaw puzzle solving
- Rotation prediction

Rotation prediction

What if we prepared training pairs of (rotated-image, rotation-angle) by randomly rotating images by (0, 90, 180, 270) from large, unlabeled image collection?

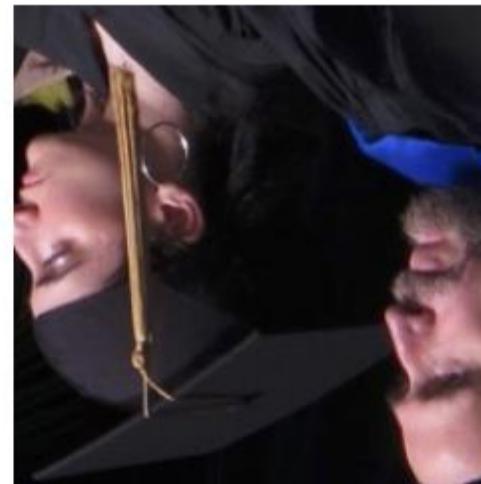
- Pretext task: recognize image rotation (0, 90, 180, 270 degrees)



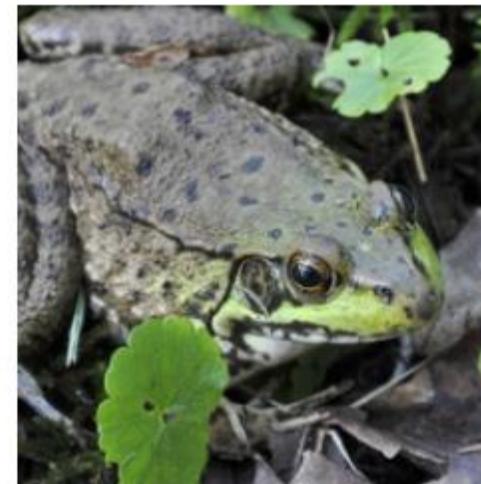
90° rotation



270° rotation



180° rotation

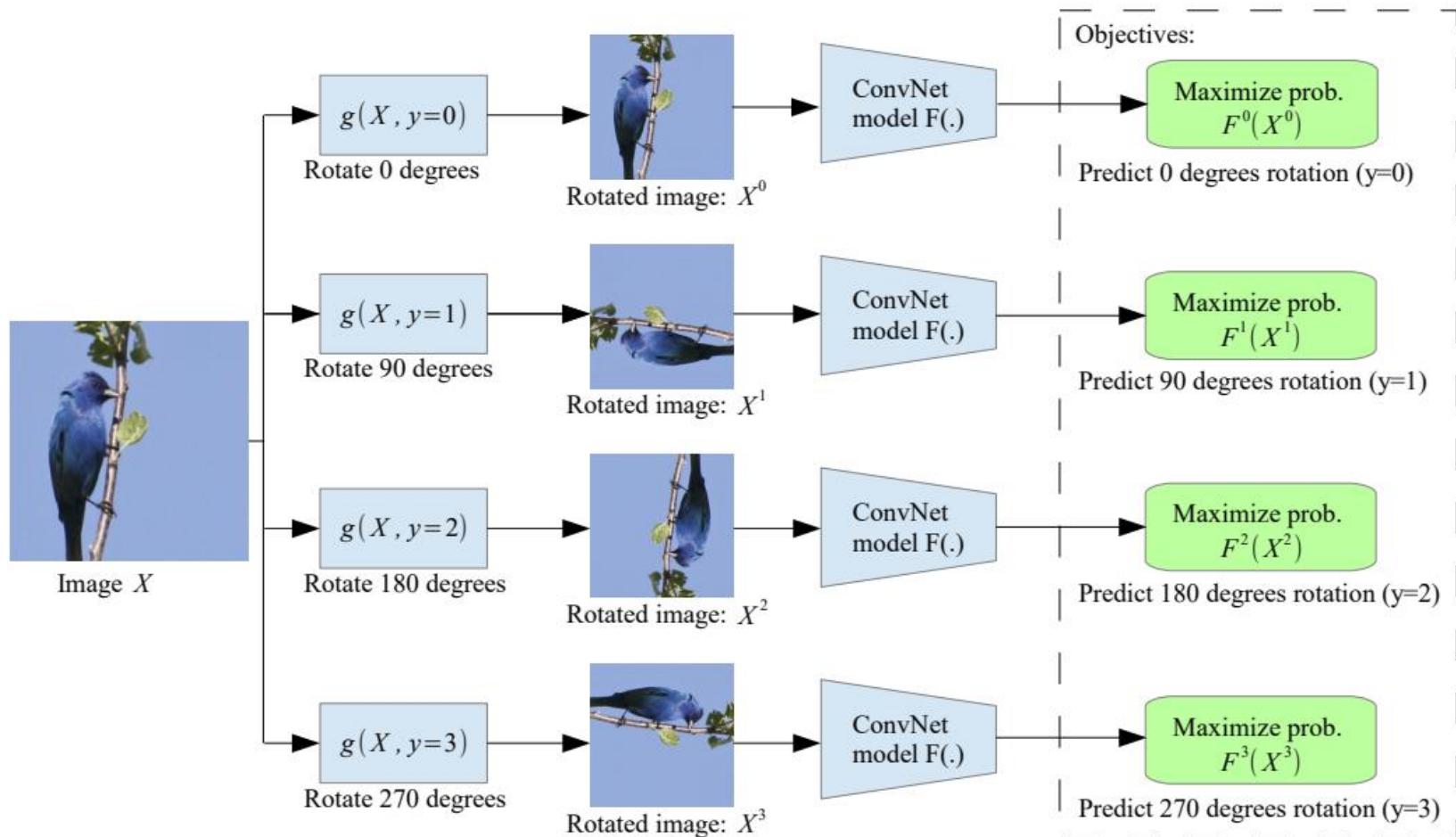


0° rotation



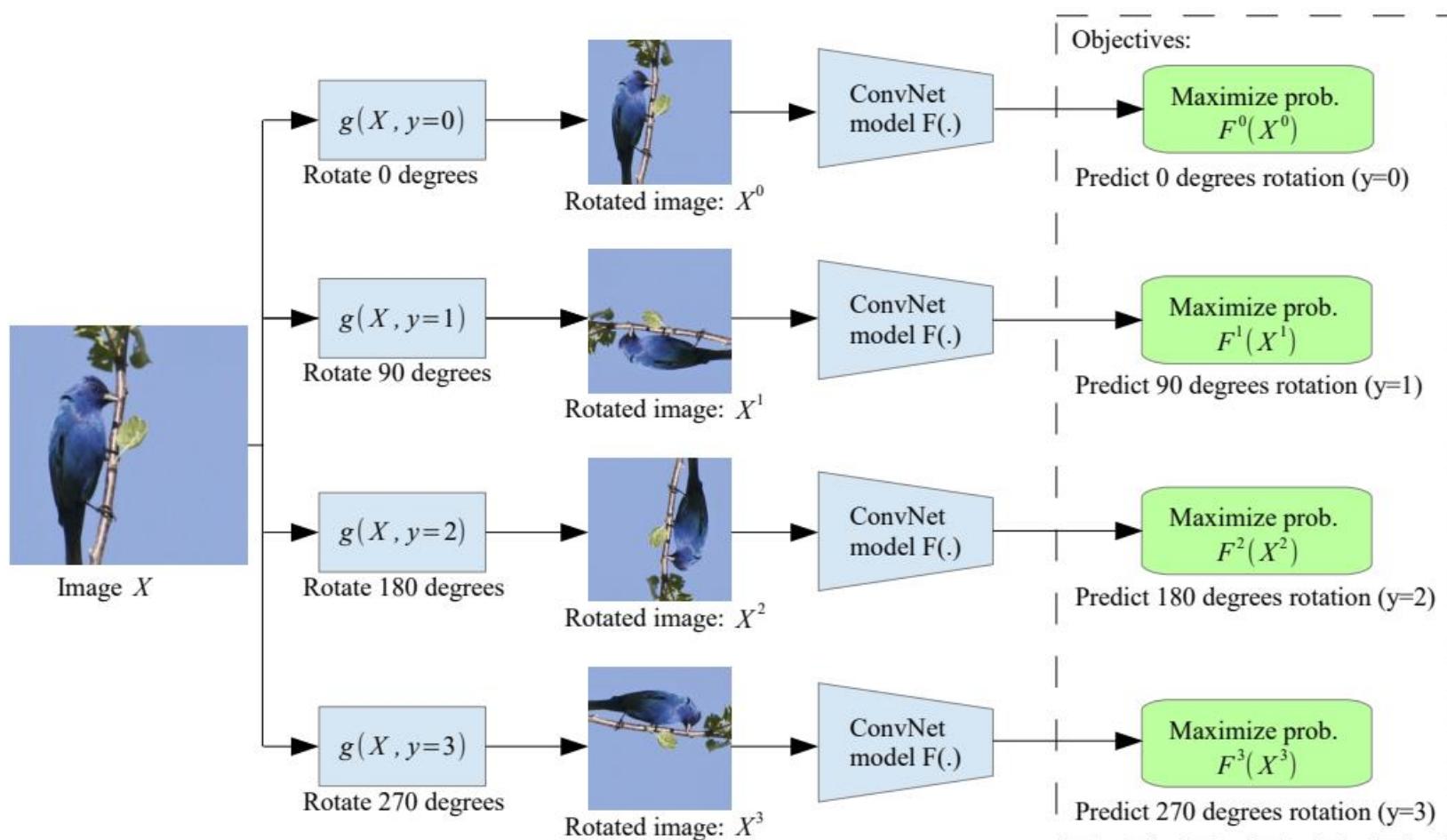
270° rotation

Rotation prediction



During training, feed in all four rotated versions of an image in the same mini-batch

Rotation prediction



Though a very simple idea, the model has to understand location, types and pose of objects in an image to solve this task and as such, the representations learned are useful for downstream tasks.

During training, feed in all four rotated versions of an image in the same mini-batch

Rotation prediction: PASCAL VOC Transfer results

Method	Classification	Detection (mAP)	Segmentation (mIoU)
Supervised (ImageNet)	79.9	56.8	48.0
Colorization	65.6	46.9	35.6
Context	65.3	51.1	
Jigsaw	67.6	53.2	37.6
Rotation	73.0	54.4	39.1

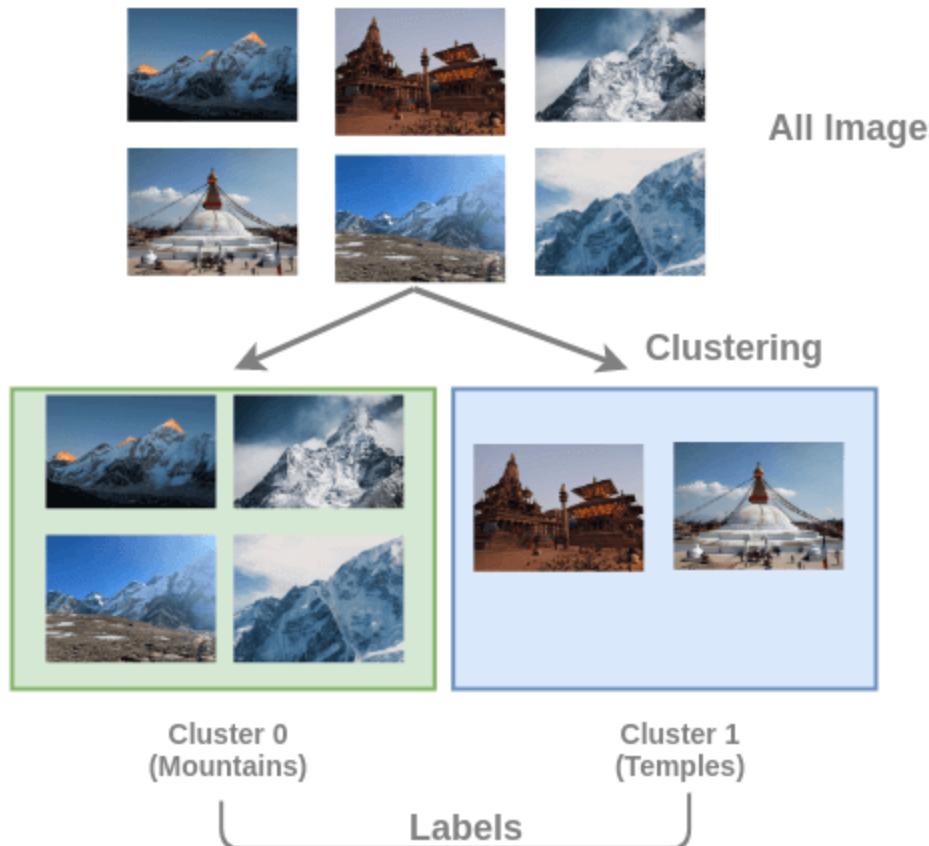
Self-supervised learning: Outline

- Data prediction
 - Colorization, Superresolution, Inpainting, Cross channel encoding
- Transformation prediction
 - Context prediction, jigsaw puzzle solving, rotation prediction
- **Automatic Label Generation**
 - Image clustering, Synthetic imagery

Image Clustering

What if we prepared training pairs of (image, cluster-number) by performing clustering on large, unlabeled image collection?

Label Generation by Clustering

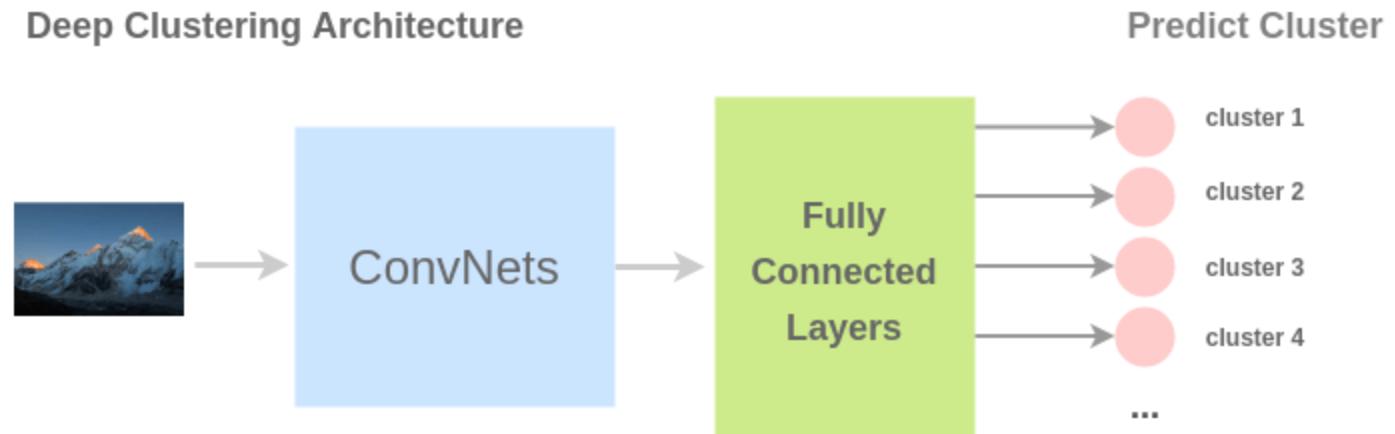
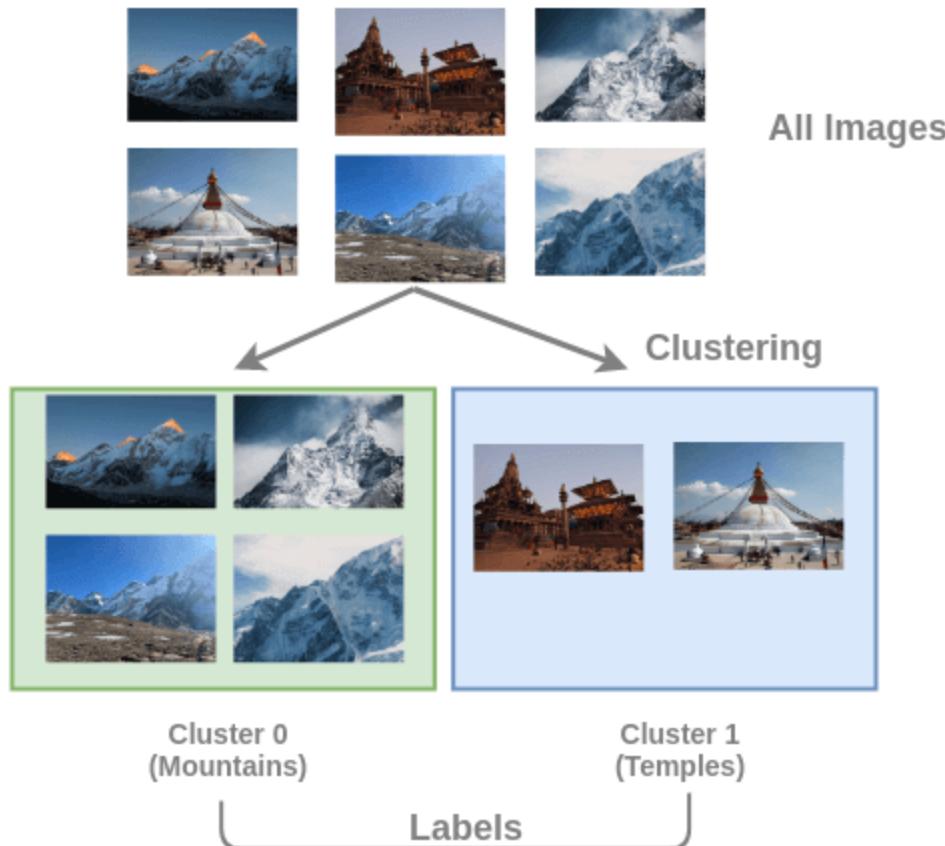


[Deep clustering for unsupervised learning of visual features](#)
[Self-labelling via simultaneous clustering and representation learning](#)
[CliqueCNN: Deep Unsupervised Exemplar Learning](#)

Image Clustering

What if we prepared training pairs of (image, cluster-number) by performing clustering on large, unlabeled image collection?

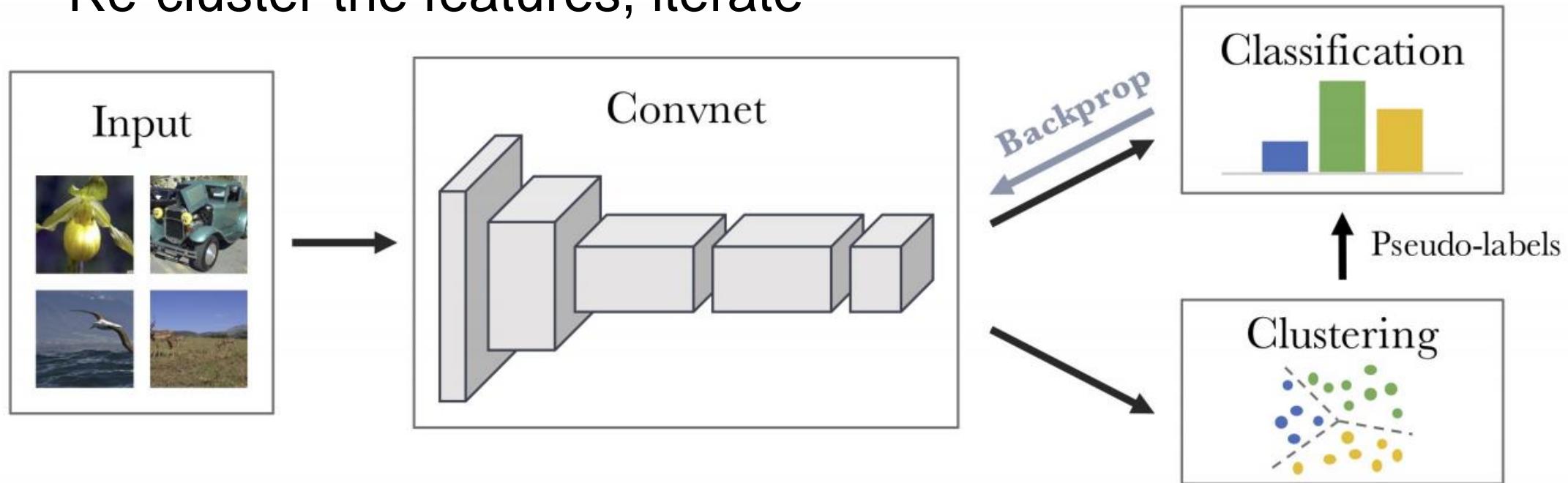
Label Generation by Clustering



[Deep clustering for unsupervised learning of visual features](#)
[Self-labelling via simultaneous clustering and representation learning](#)
[CliqueCNN: Deep Unsupervised Exemplar Learning](#)

Deep Clustering

- Cluster the features to obtain pseudo-labels
- Use pseudo-label prediction as pretext task to train the network
- Re-cluster the features, iterate

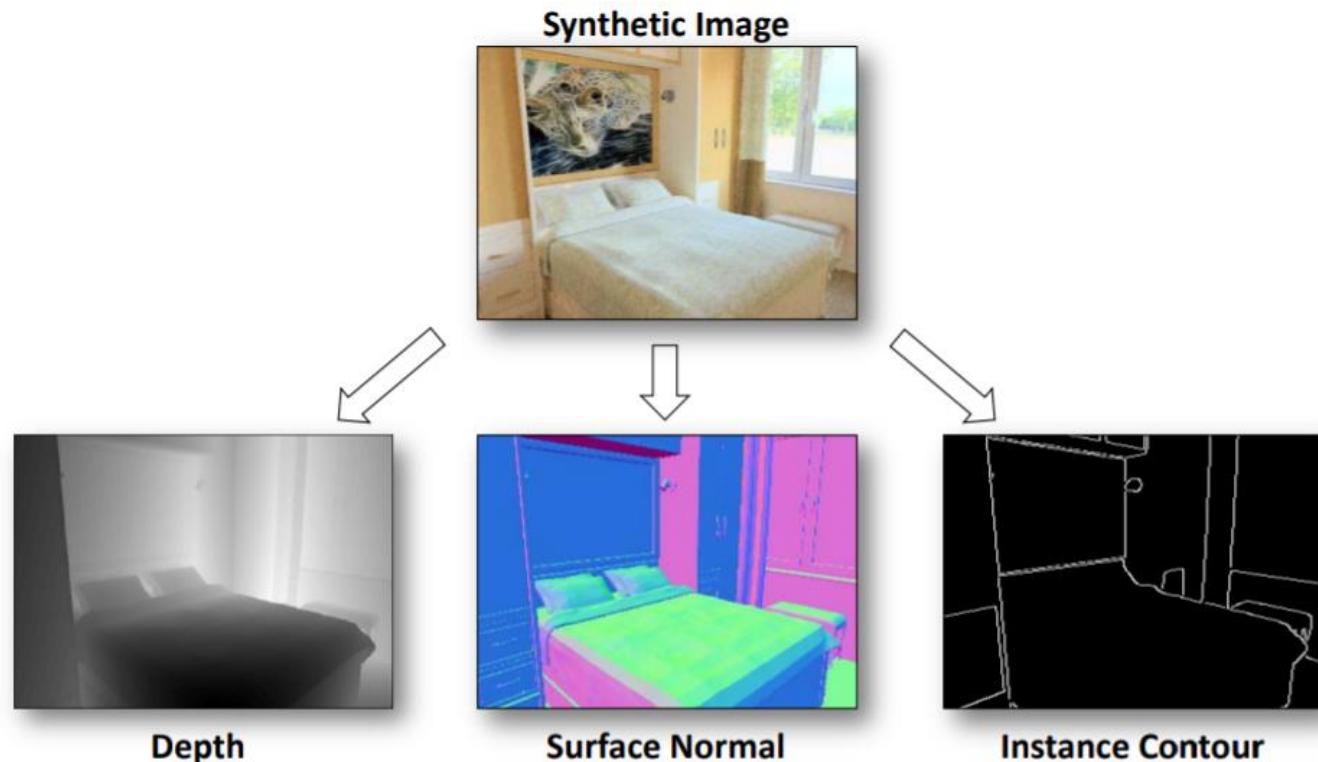


Deep Clustering: PASCAL VOC Transfer results

Method	Classification	Detection (mAP)	Segmentation (mIoU)
Supervised (ImageNet)	79.9	56.8	48.0
Colorization	65.6	46.9	35.6
Context	65.3	51.1	
Jigsaw	67.6	53.2	37.6
Rotation	73.0	54.4	39.1
DeepCluster	73.7	55.4	45.1

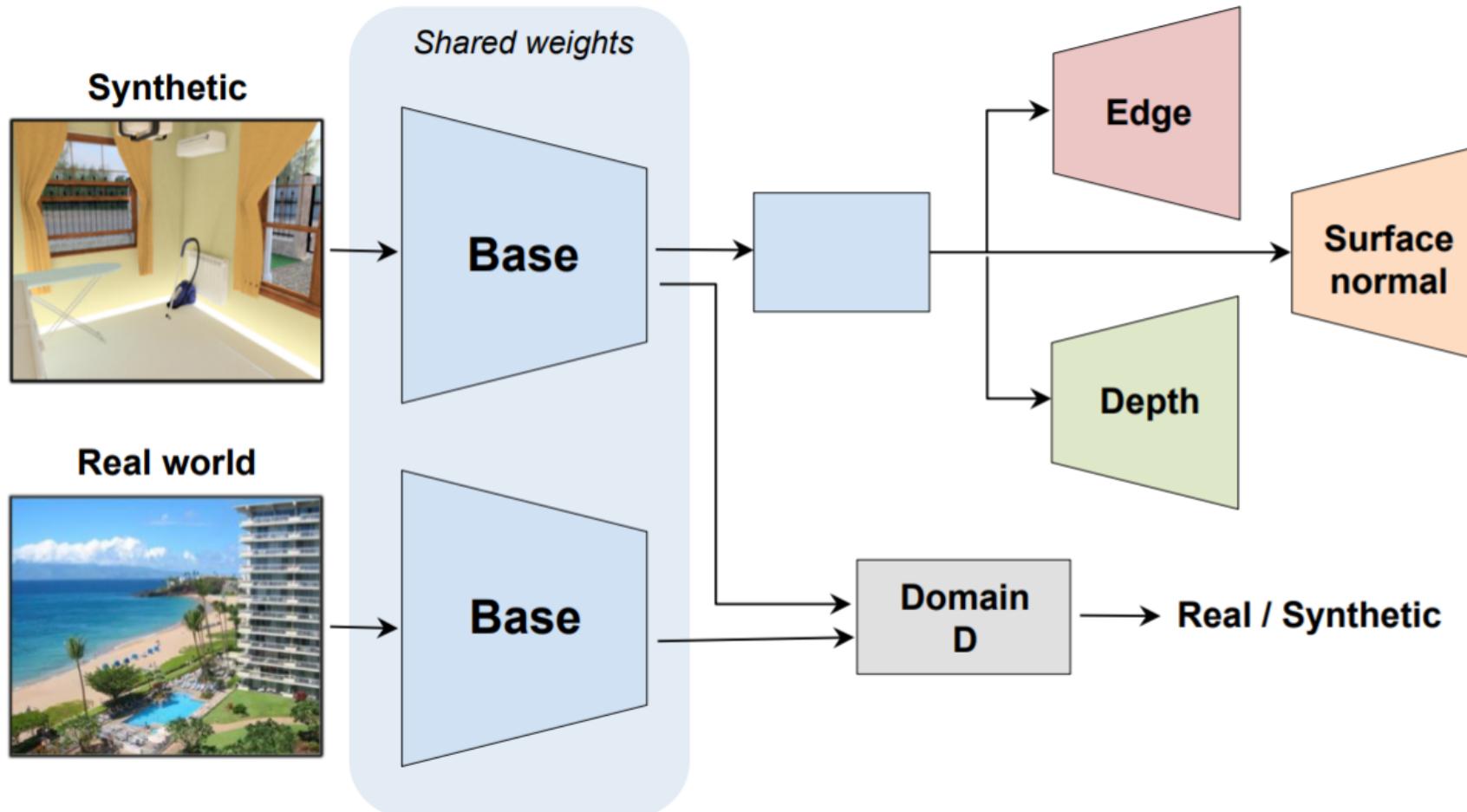
Synthetic Imagery

What if we prepared training pairs of (image, properties) by generating synthetic images using graphics engines and adapting it to real images?



Synthetic Imagery

The upper net takes a synthetic image and predicts its depth, surface normal, and instance contour (edge) map. The bottom net extracts features from a real-world image. The domain discriminator D tries to differentiate real and synthetic features. The learned blue modules are used for transfer learning on real-world tasks.



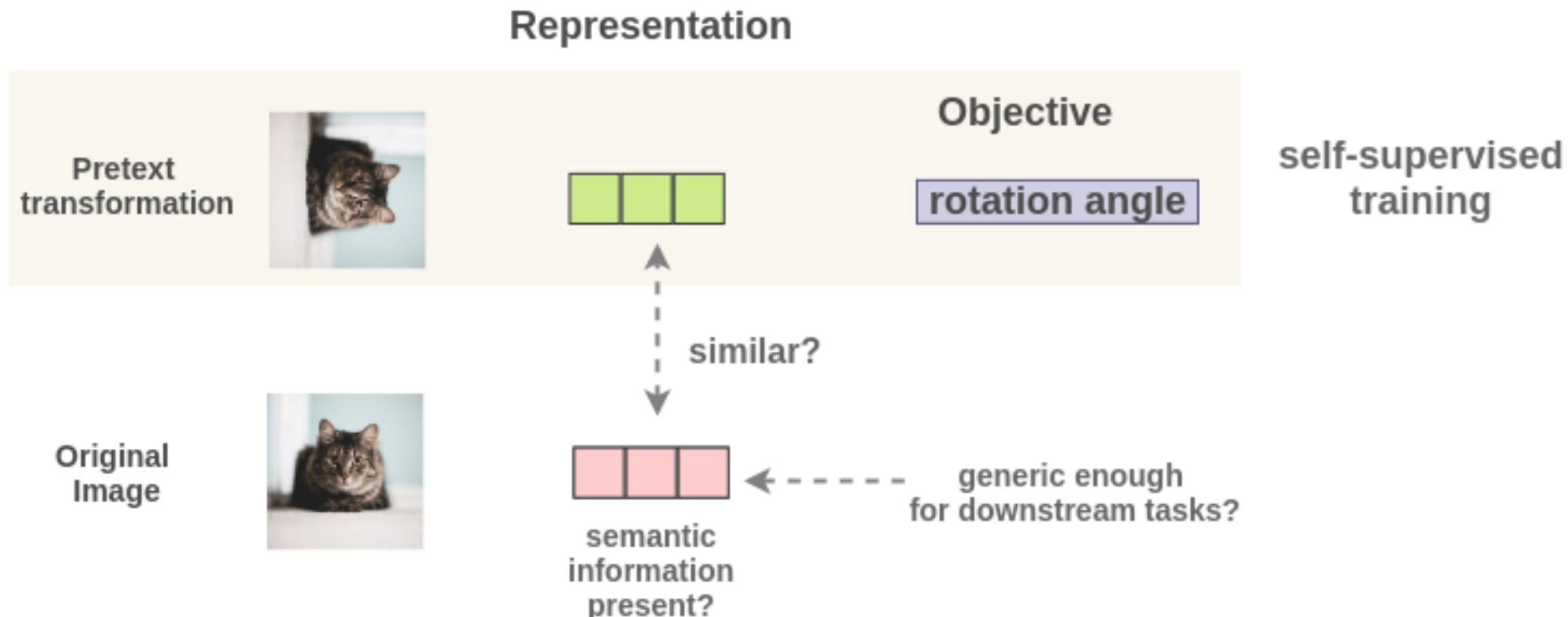
Cross-Domain Self-supervised Multi-task Feature Learning using Synthetic Imagery

Self-supervised learning: Outline

- Data prediction
 - Colorization, Superresolution, Inpainting, Cross channel encoding
- Transformation prediction
 - Context prediction, jigsaw puzzle solving, rotation prediction
- Automatic Label Generation
 - Image clustering, Synthetic imagery
- Contrastive learning
 - PIRL, MoCo, SimCLR, SWaV

Problems with earlier approaches

- As such, the image representations learned can overfit to the transformation and not generalize well on downstream tasks.
- The representations will be **covariant** with the transformation.
- It will only encode essential information to predict the transformation and could discard useful semantic information.



[Figure source](#)

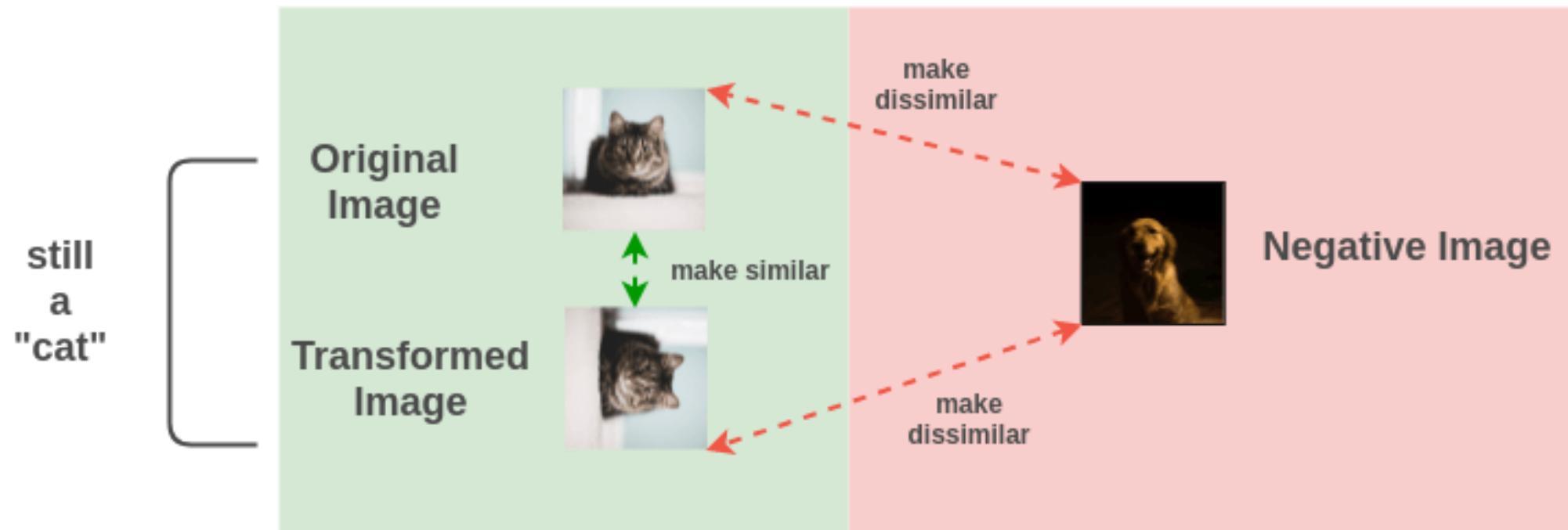
Contrastive methods

- Encourage representations of transformed versions of the same image to be the same and different images to be different



Contrastive methods

- Encourage representations of transformed versions of the same image to be the same and different images to be different



[Figure source](#)

Contrastive loss formulation

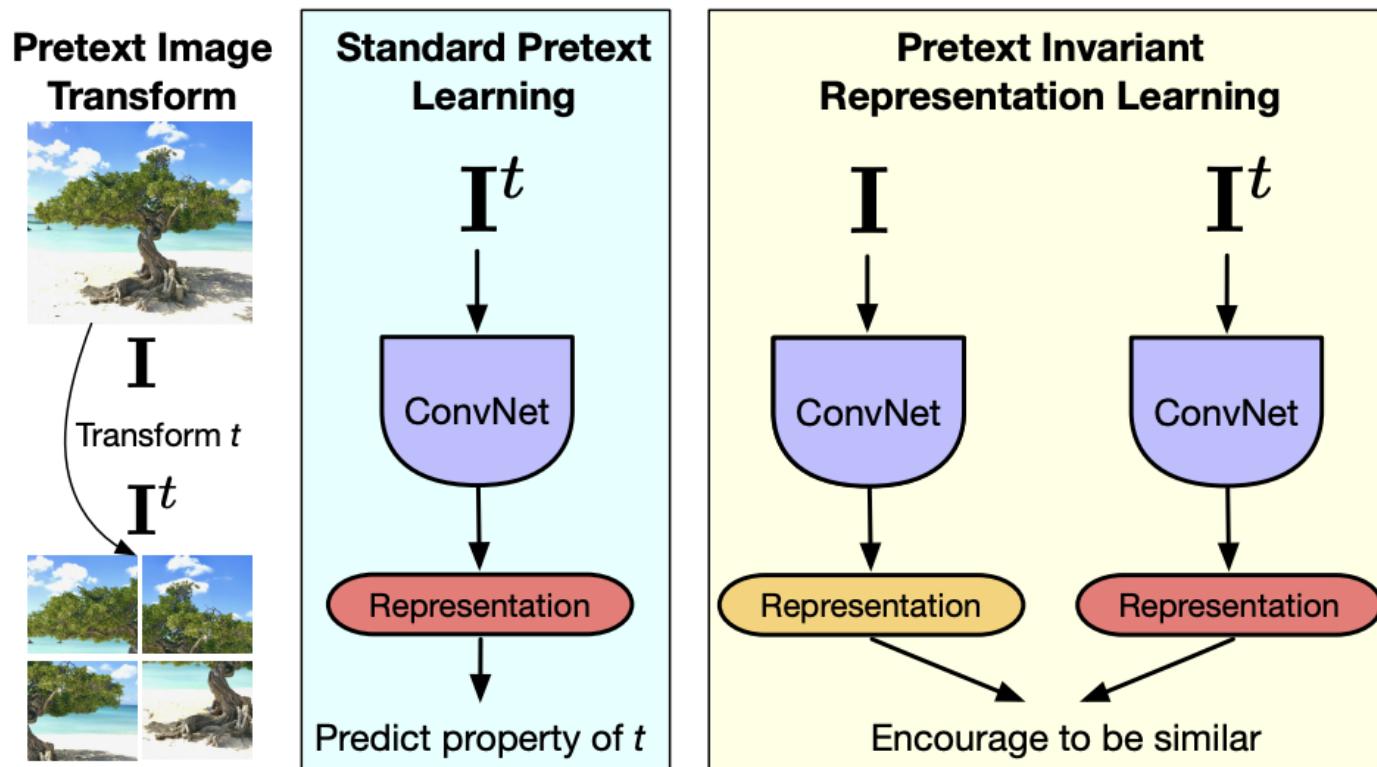
- Given: query point x , positive samples x^+ , negative samples x^-
 - Positives are typically transformed versions of x , negatives are random examples from the same mini-batch or *memory bank*
- Key idea: learn representation to make x similar to x^+ , dissimilar from x^- (similarity is measured by dot product of normalized features OR cosine similarity)
- Intuitively, contrastive loss for x, x^+ is the loss of a softmax classifier that tries to classify x as x^+ :

$$l(x, x^+) = -\log \frac{\exp(f(x)^T f(x^+)/\tau)}{\exp(f(x)^T f(x^+)/\tau) + \sum_{j=1}^N \exp(f(x)^T f(x_j^-)/\tau)}$$

- τ is the *temperature* hyperparameter (determines how concentrated the softmax is)

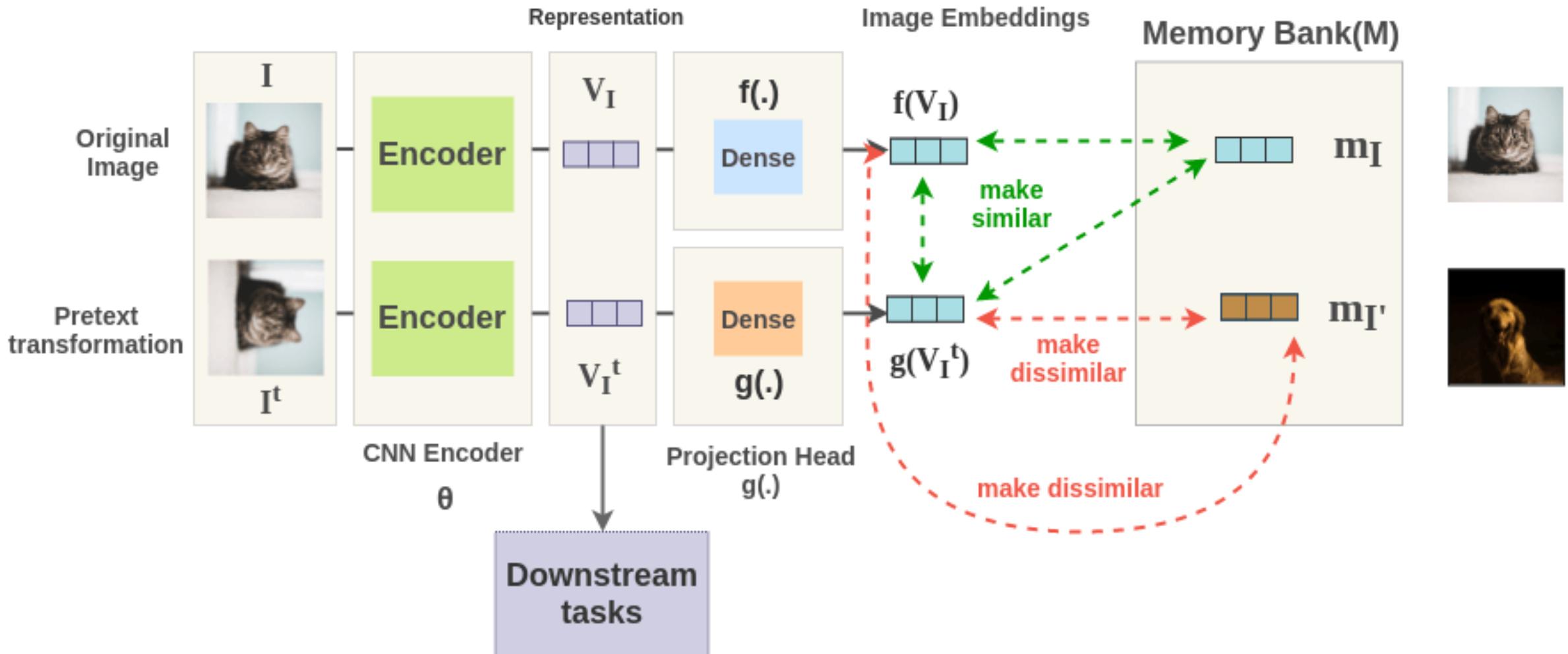
Pretext-invariant representation learning (PIRL)

- Key idea: instead of predicting the transformation of the input, learn a representation *invariant* to the transformation



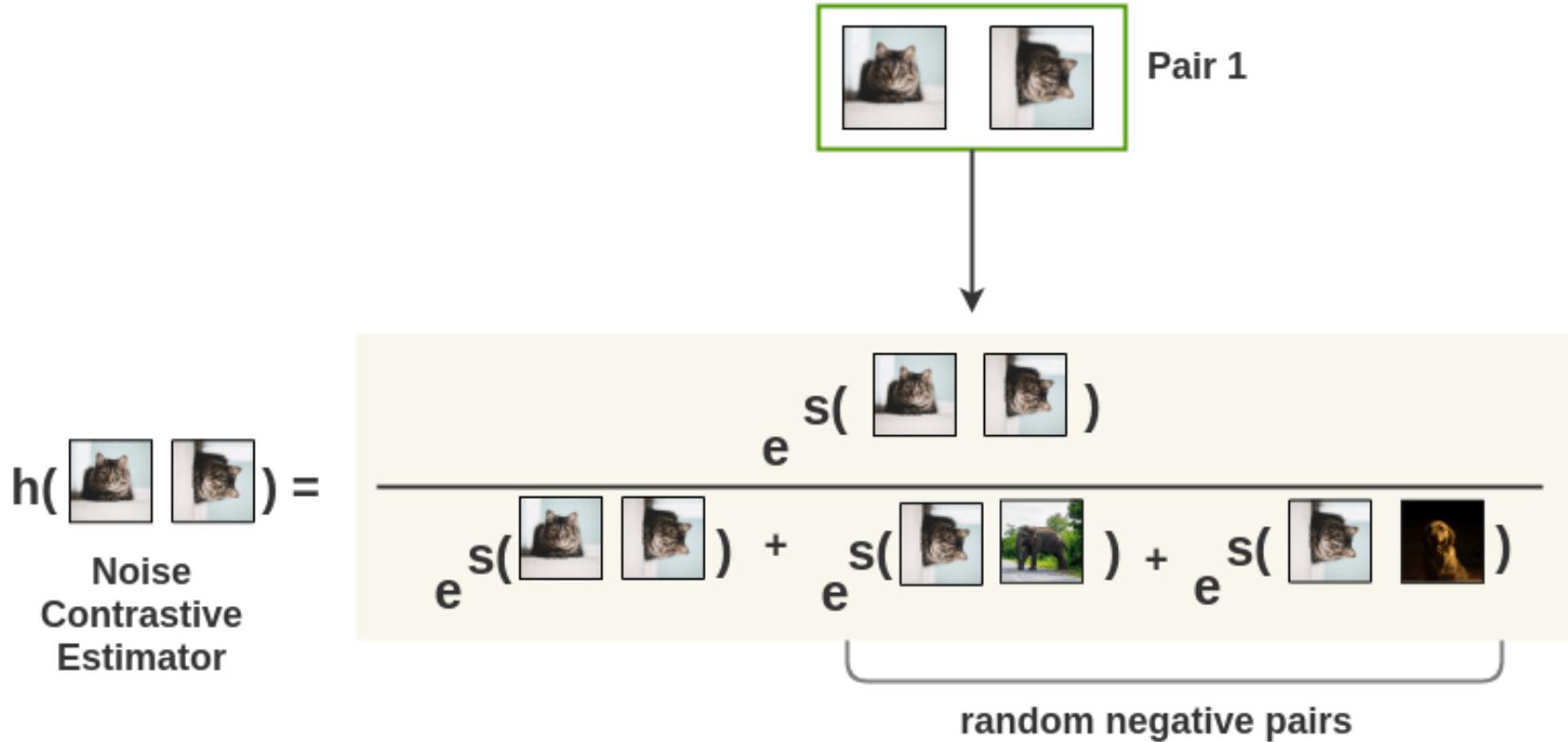
Pretext-invariant representation learning (PIRL)

PIRL Generic Framework



Pretext-invariant representation learning (PIRL)

PIRL uses Noise Contrastive Estimator (NCE)



$$L_{NCE}(I, I^t) = -\log[h(f(V_I), g(V_{I^t}))] - \sum_{I' \in D_N} \log[1 - h(g(V_{I^t}), f(V_{I'}))]$$

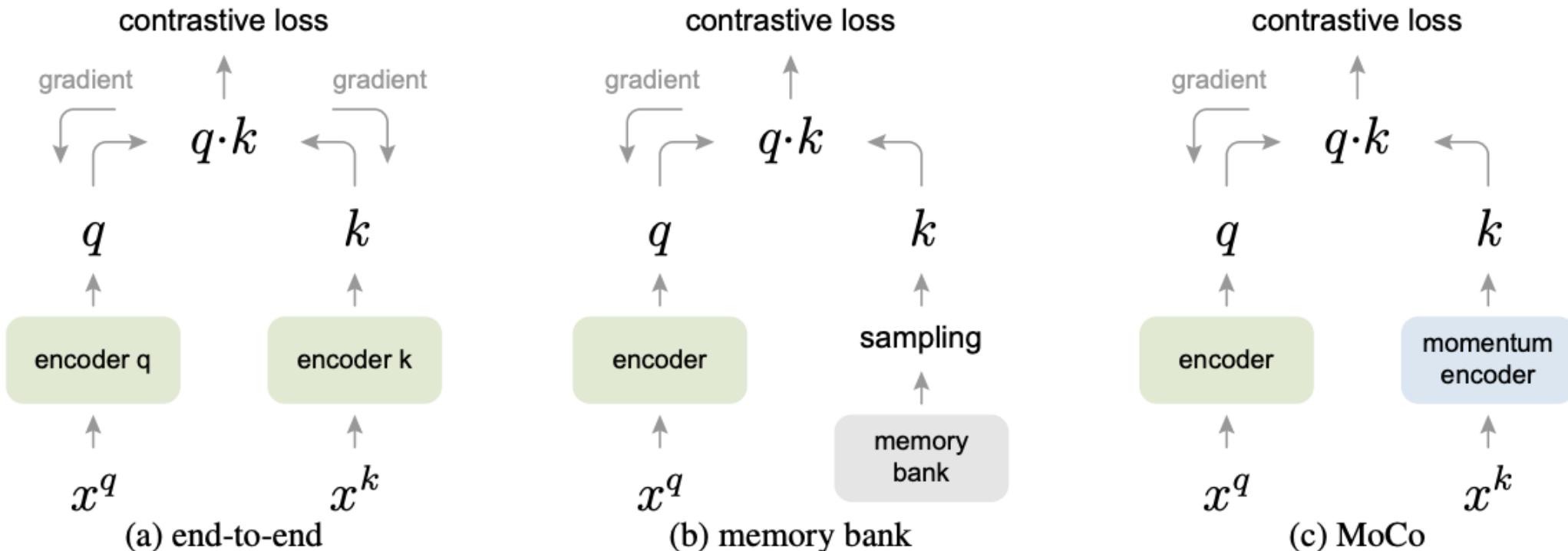
PIRL: Results

Method	Network	AP ^{all}	AP ⁵⁰	AP ⁷⁵	ΔAP ⁷⁵
Supervised	R-50	52.6	81.1	57.4	=0.0
Jigsaw [19]	R-50	48.9	75.1	52.9	-4.5
Rotation [19]	R-50	46.3	72.5	49.3	-8.1
NPID++ [72]	R-50	52.3	79.1	56.9	-0.5
PIRL (ours)	R-50	54.0	<u>80.7</u>	59.7	+2.3
CPC-Big [26]	R-101	–	70.6*	–	
CPC-Huge [26]	R-170	–	72.1*	–	
MoCo [24]	R-50	55.2*†	81.4*†	61.2*†	

Table 1: Object detection on VOC07+12 using Faster R-CNN. Detection AP on the VOC07 test set after finetuning Faster R-CNN models (keeping BatchNorm fixed) with a ResNet-50 backbone pre-trained using self-supervised learning on ImageNet. Results for supervised ImageNet pre-training are presented for reference. Numbers with * are adopted from the corresponding papers. Method with † finetunes BatchNorm. PIRL significantly outperforms supervised pre-training without extra pre-training data or changes in the network architecture. Additional results in Table 6.

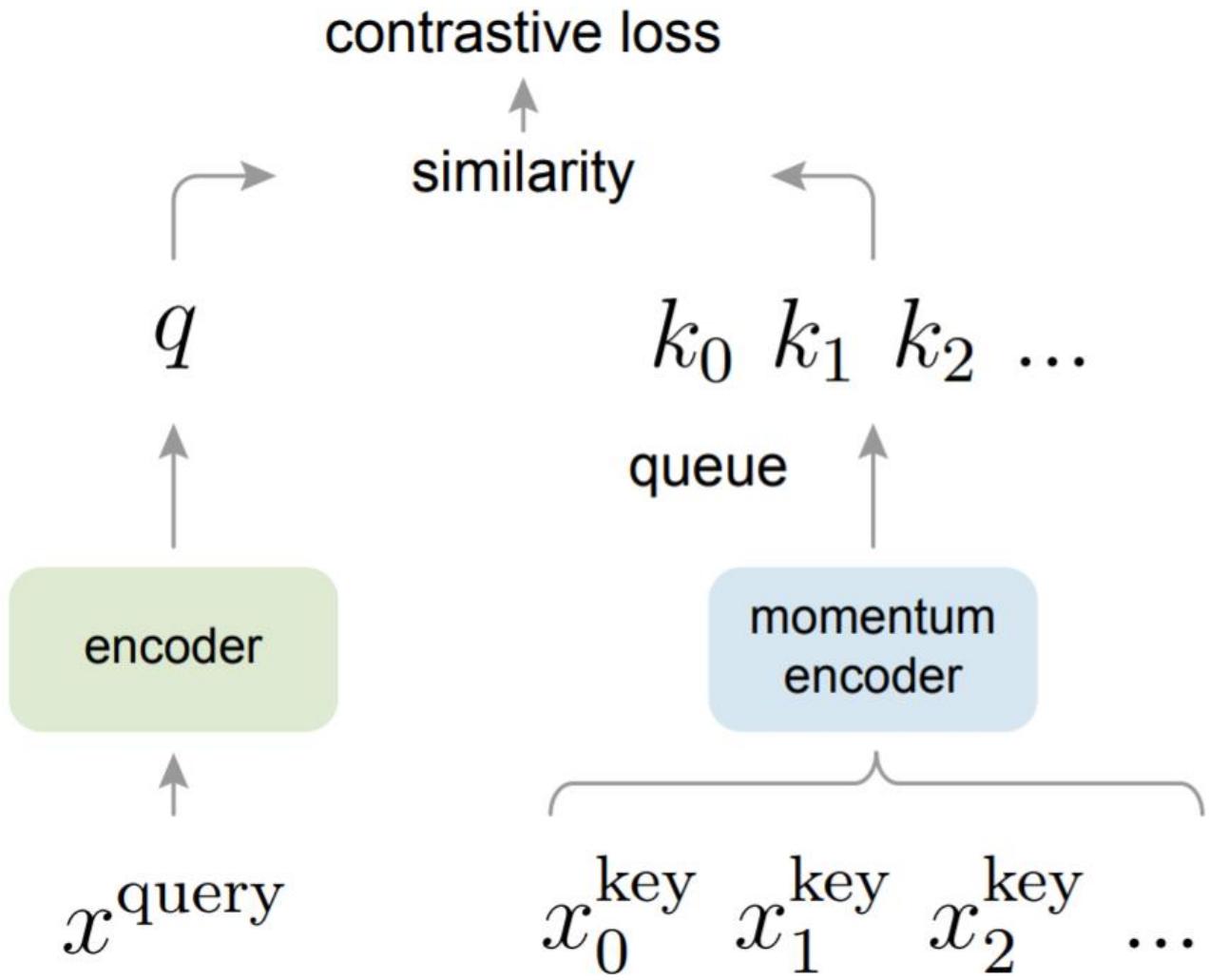
Momentum contrast

- Use instance discrimination as pretext task, transform query and key by random augmentations, use queue encoded by a momentum encoder instead of memory bank



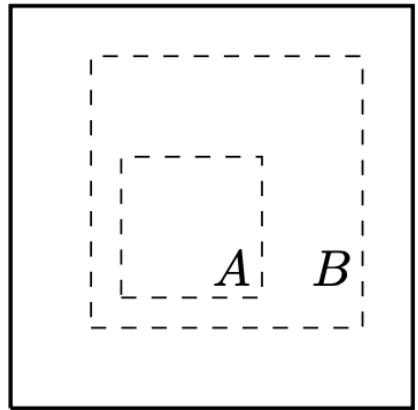
Momentum contrast

- Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss.
- The dictionary keys $\{k_0, k_1, k_2, \dots\}$ are defined on-the-fly by a set of data samples.
- The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size.
- The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder.
- This method enables a large and consistent dictionary for learning visual representations.

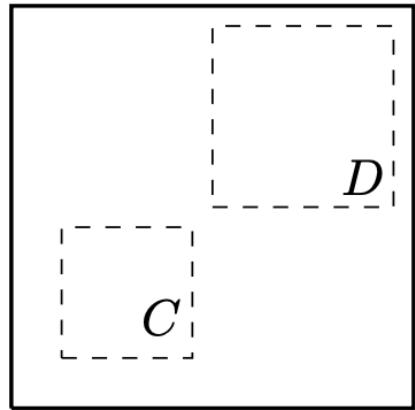


SimCLR

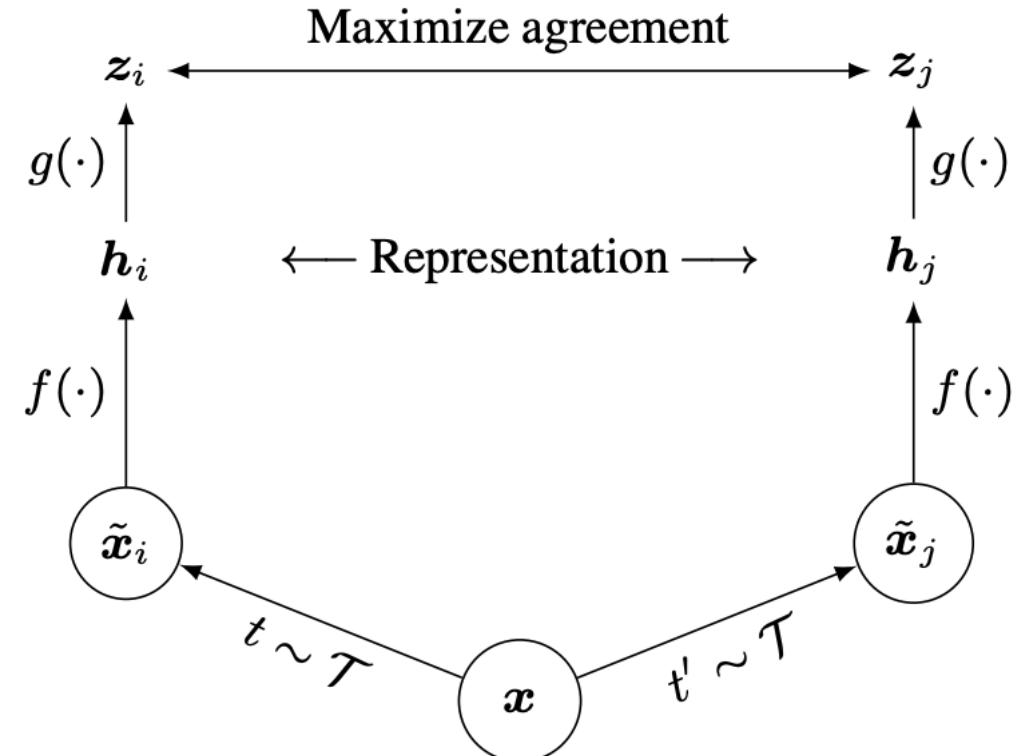
- Form two views of the input by composing data augmentations
 - Cropping and resizing, color distortion, blur



(a) Global and local views.

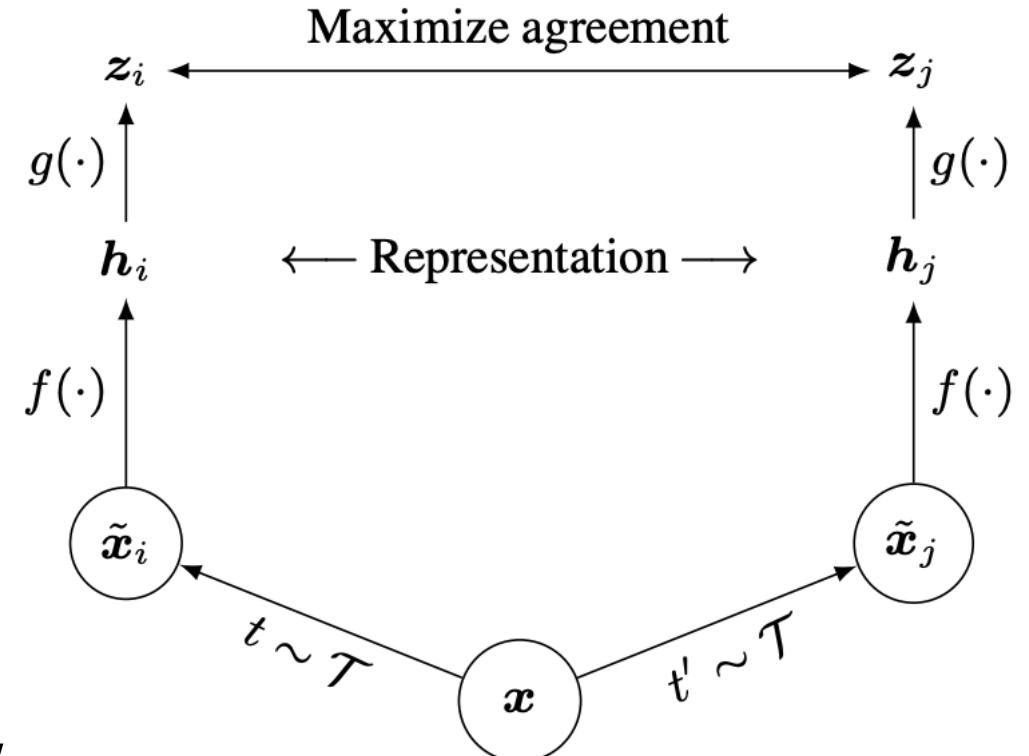


(b) Adjacent views.



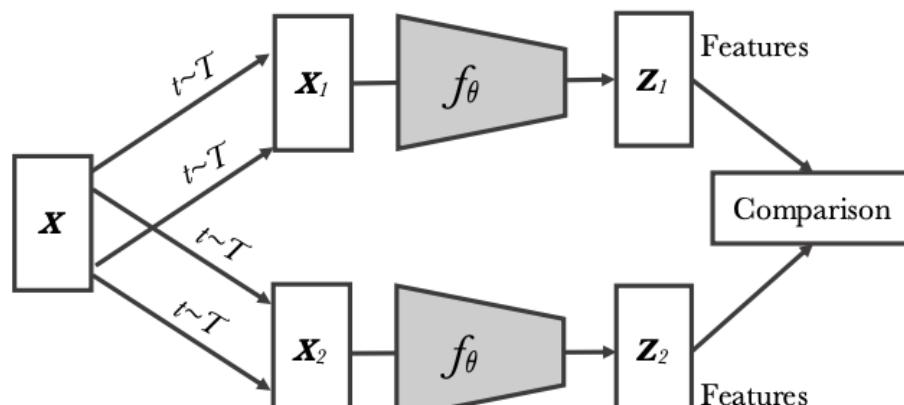
SimCLR

- Form two views of the input by composing data augmentations
 - Cropping and resizing, color distortion, blur
- No memory bank, large mini-batch size (on cloud TPU)
- Introduce nonlinear transformation between representation and contrastive loss (or, use representation a few layers below the contrastive loss)

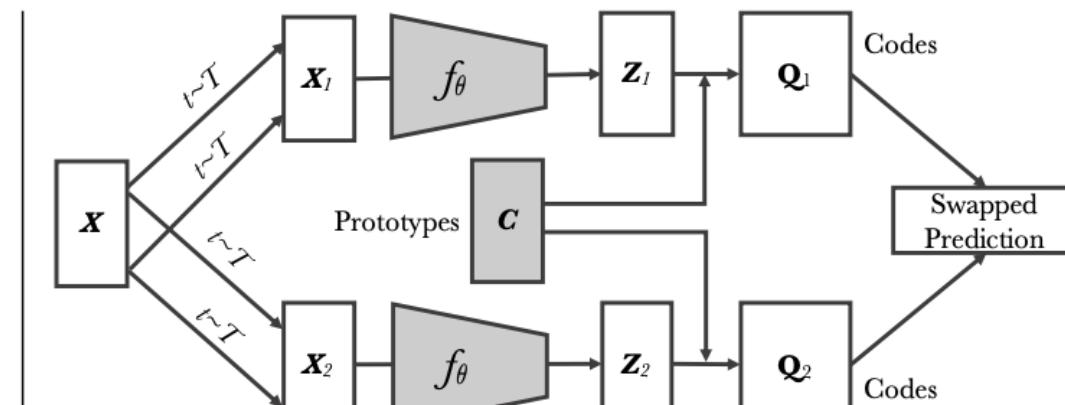


Swapping Assignments Between Views (SWaV)

- Predict cluster assignment of one “view” (transformed version of input image) from representation of another “view”
 - Prototypes or cluster centers are learned online within mini-batch
- Simply put, it uses a swapped prediction mechanism where it predicts the code of a view from the representation of another view.
- Once again, data augmentation strategy matters

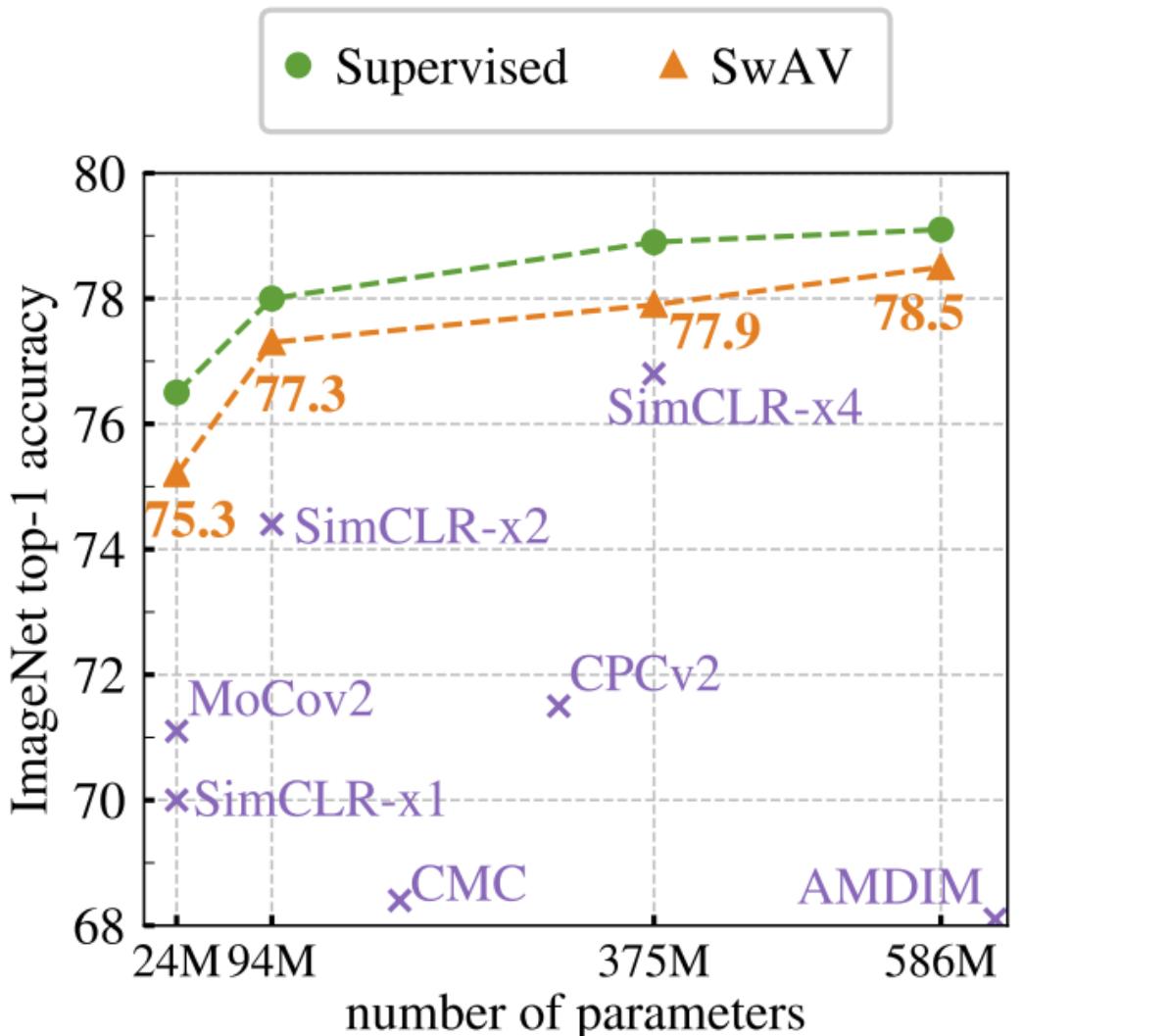


Contrastive instance learning



Swapping Assignments between Views (Ours)

SWaV: Results



Supervised
SWaV

Object Detection	
VOC07+12 (Faster R-CNN)	COCO (DETR)
81.3	40.8
82.6	42.1

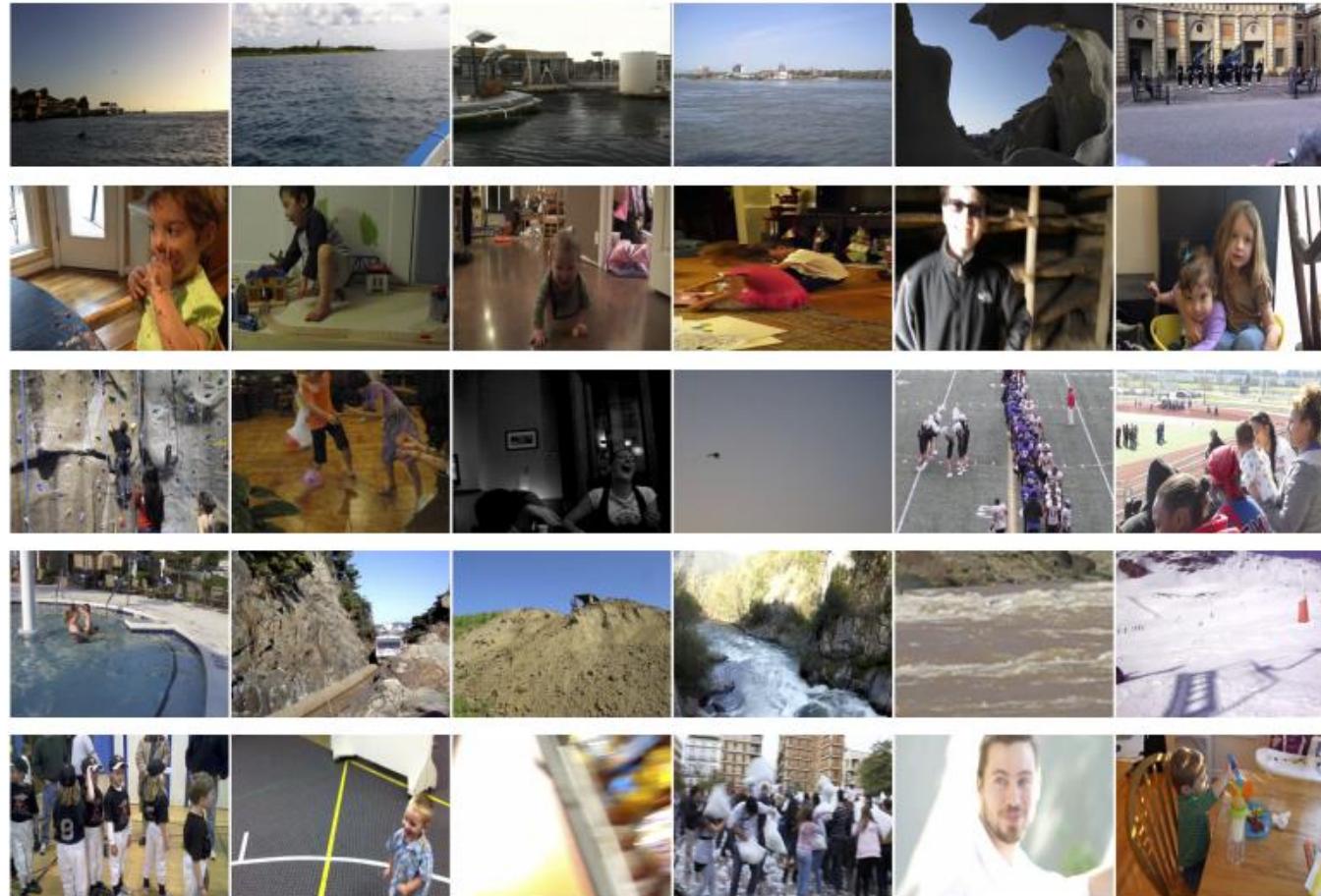
Why do contrastive methods work?

- L2 normalization of features (before computing dot product to estimate similarity) is important ([Wang and Isola, 2020](#))
- The essential property of the loss is enforcing closeness of positive features while maximizing uniformity of the distribution of features over the hypersphere ([Wang and Isola, 2020](#))
- The choice of data augmentation operations or transformations between two positive “views” is also important and needs further study ([Tian et al., 2020](#))

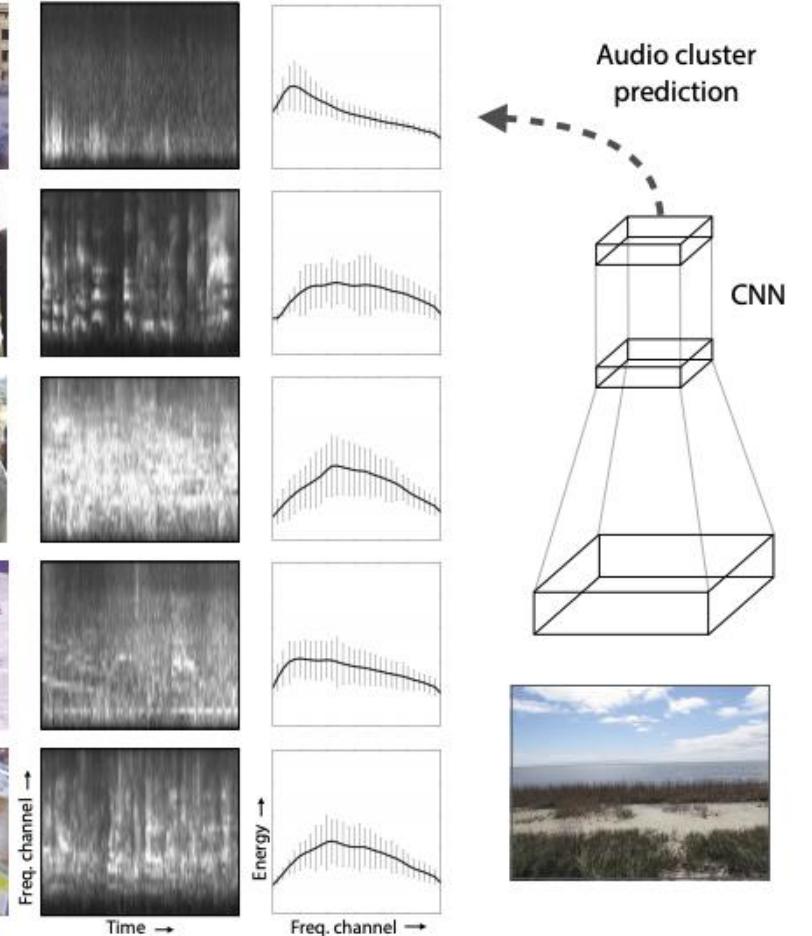
Self-supervised learning: Outline

- Data prediction
 - Colorization, Superresolution, Inpainting, Cross channel encoding
- Transformation prediction
 - Context prediction, jigsaw puzzle solving, rotation prediction
- Automatic Label Generation
 - Image clustering, Synthetic imagery
- Contrastive learning
 - PIRL, MoCo, SimCLR, SWaV
- **Self-supervision beyond still images**
 - Audio, video, language

Learning from audio



(a) Images grouped by audio cluster



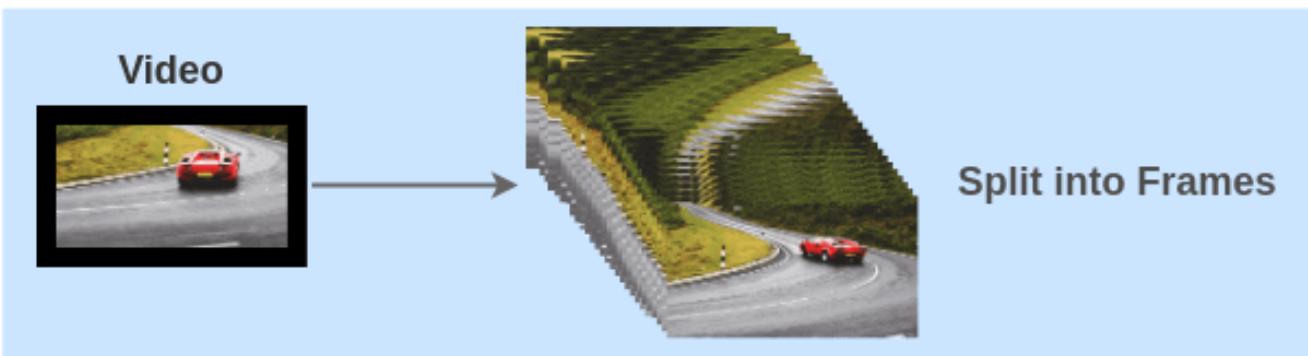
(b) Clustered audio stats. (c) CNN model

Self-Supervised Learning From Video

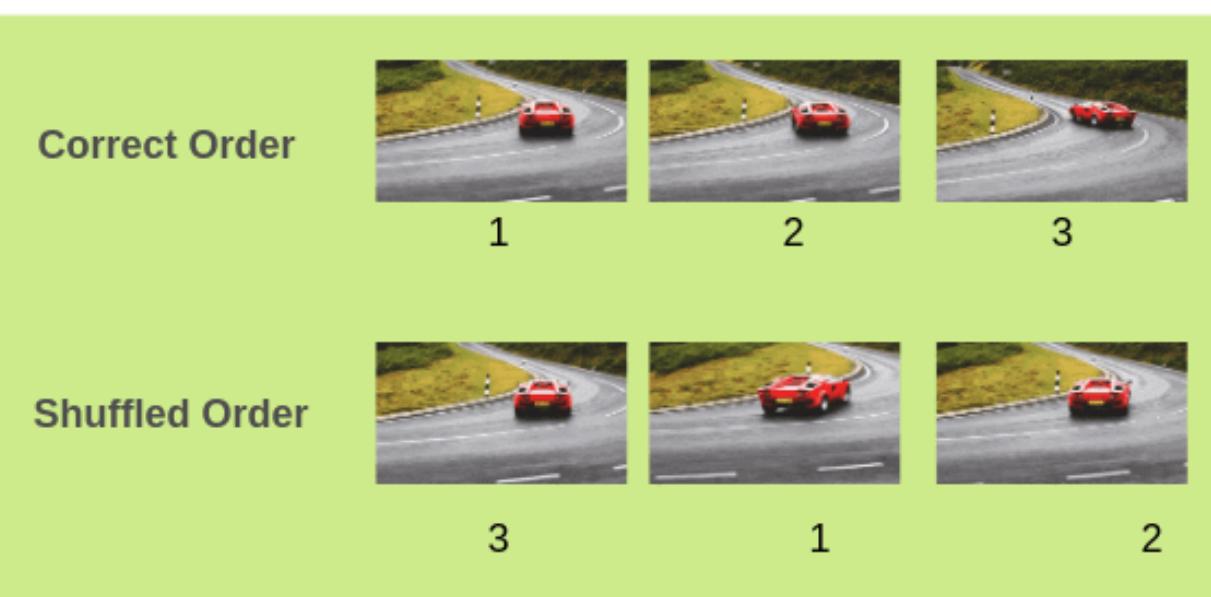
Frame Order Verification

What if we prepared training pairs of (video frames, correct/ incorrect order) by shuffling frames from videos of objects in motion?

Frame Order Training Data Generation



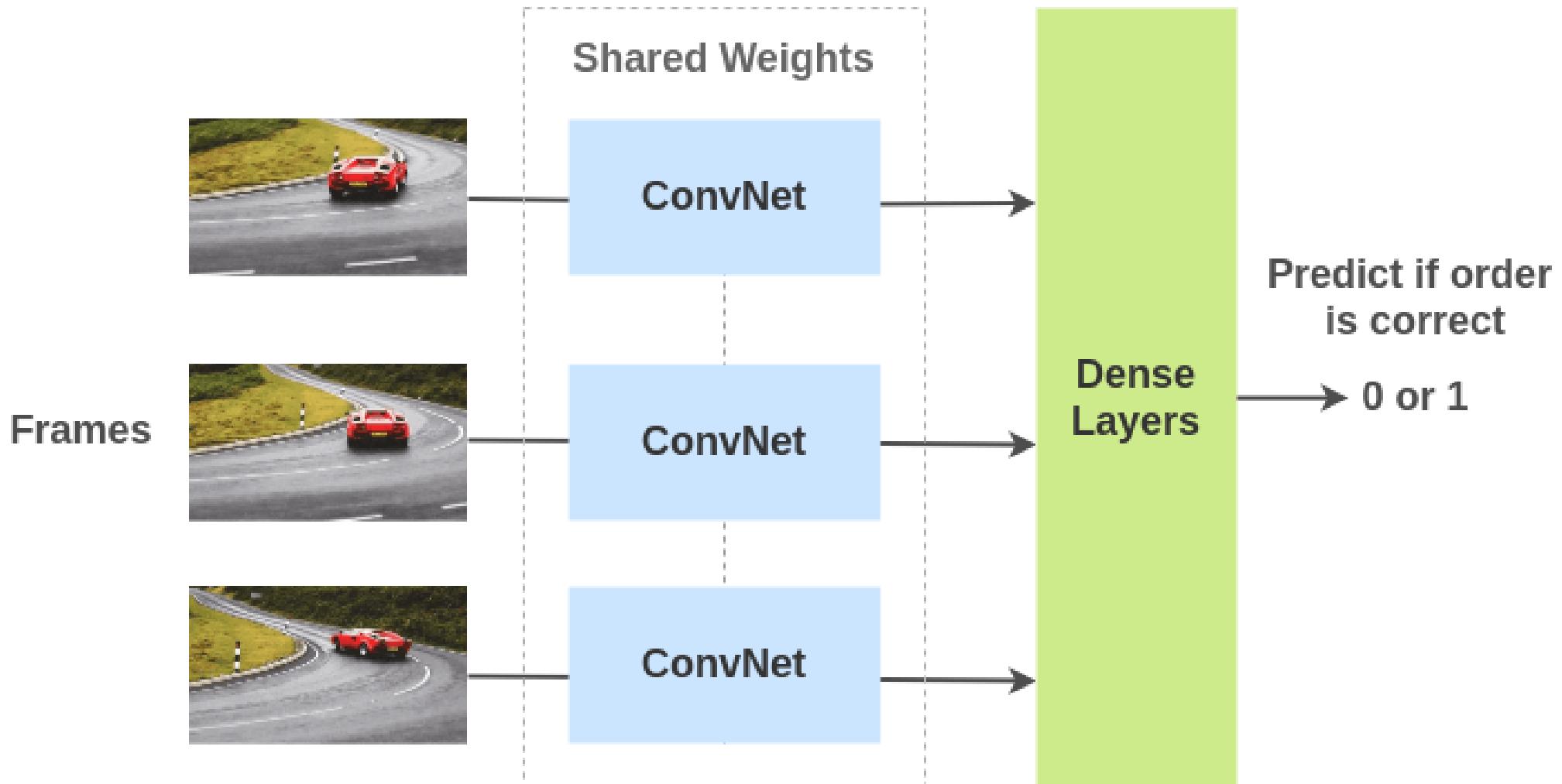
Prepare Pairs



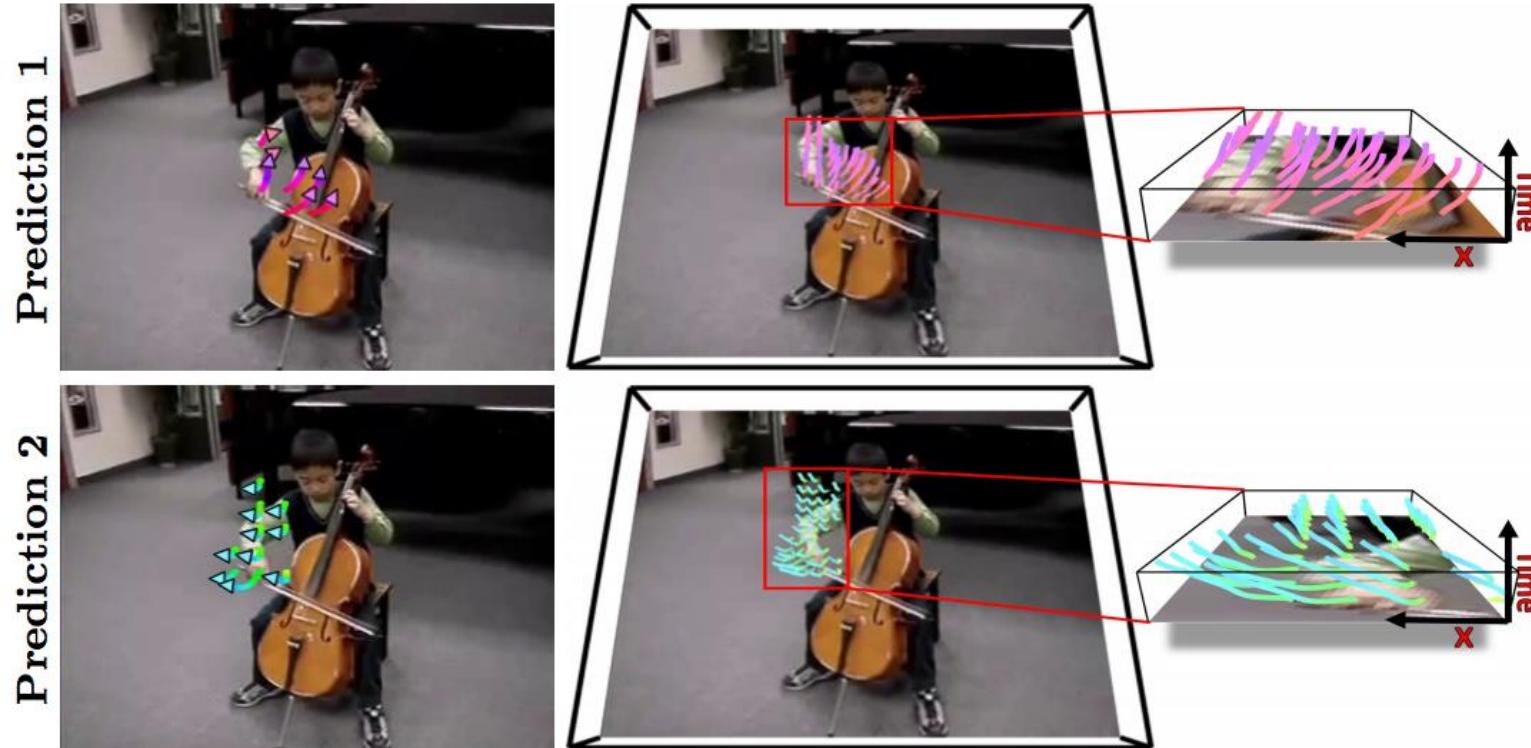
[Figure source](#)

Self-Supervised Learning From Video

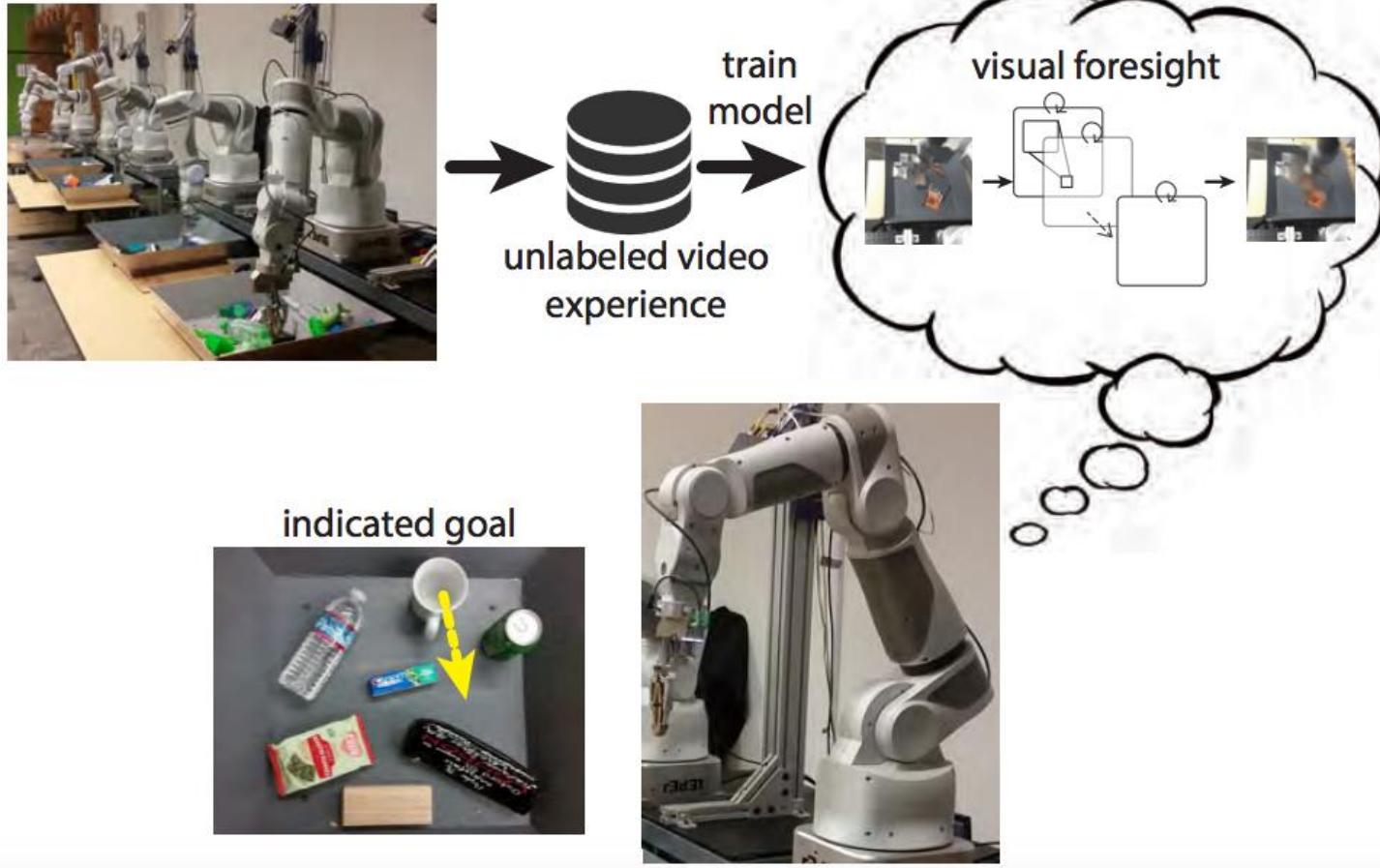
Shuffle and Learn Architecture



Future prediction

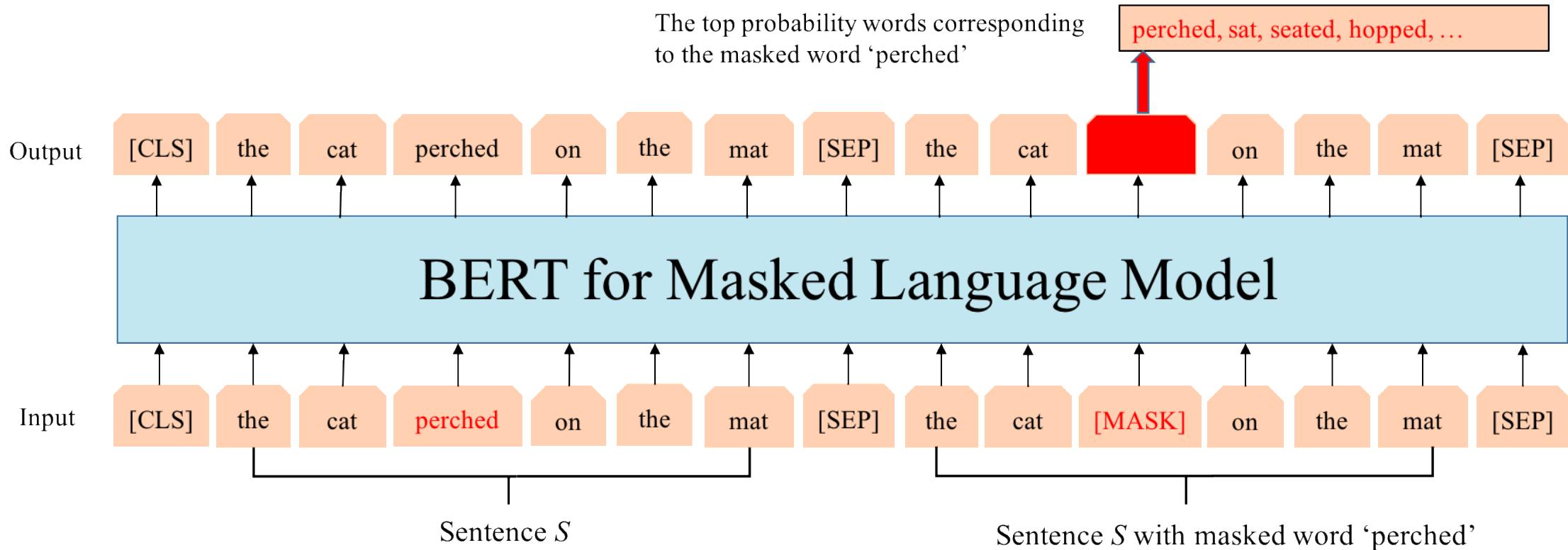


Future prediction



Self-supervised learning in NLP (coming up)

- word2vec, GloVe, BERT, ELMO, GPT, ...



For further reading

<https://github.com/jason718/awesome-self-supervised-learning>

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Visualizing and explaining neural networks



<https://deeplearning.net/deepdreamgenerator.com/>

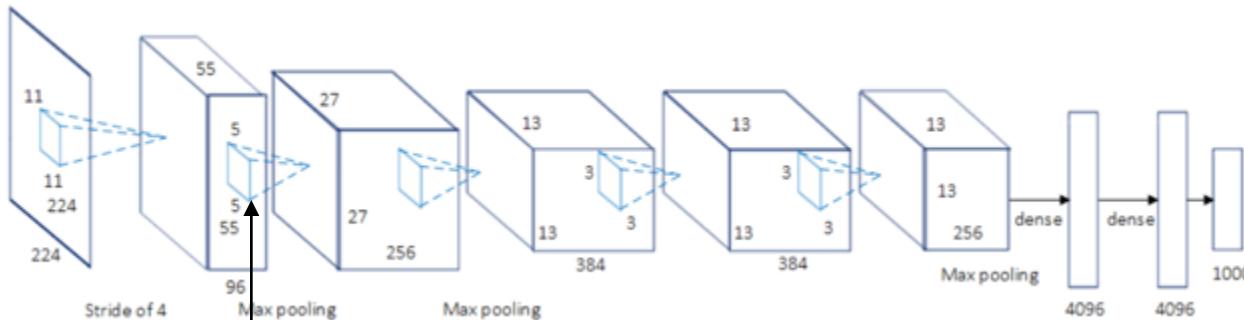
Outline

- Basic visualization techniques
- Mapping activations back to the image
- Synthesizing images to maximize activation
- Saliency maps
- Quantifying interpretability of units

Overview and basic visualization techniques



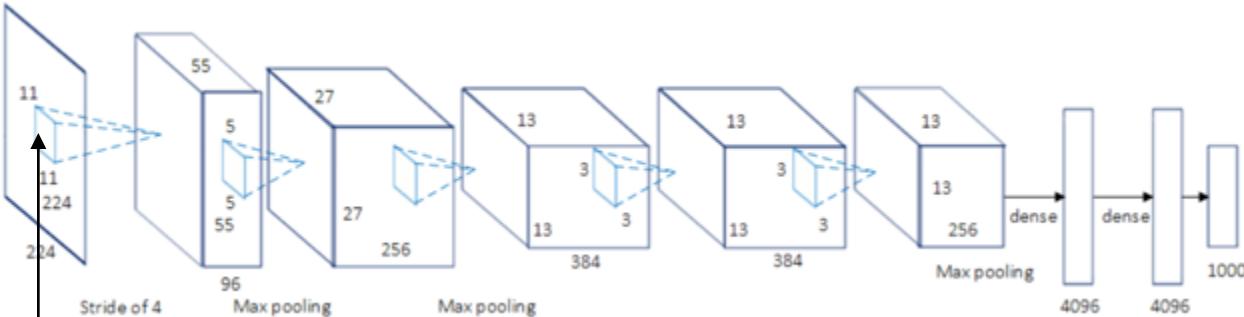
Overview and basic visualization techniques



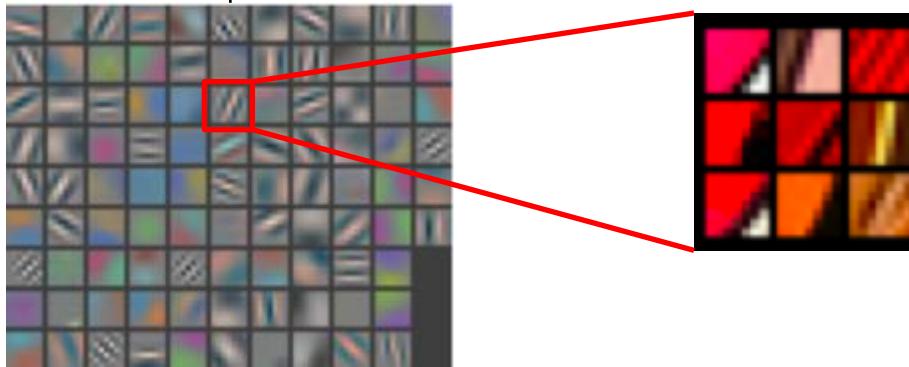
Not too helpful for subsequent layers

Features from a CIFAR10 network, via [Stanford CS231n](#)

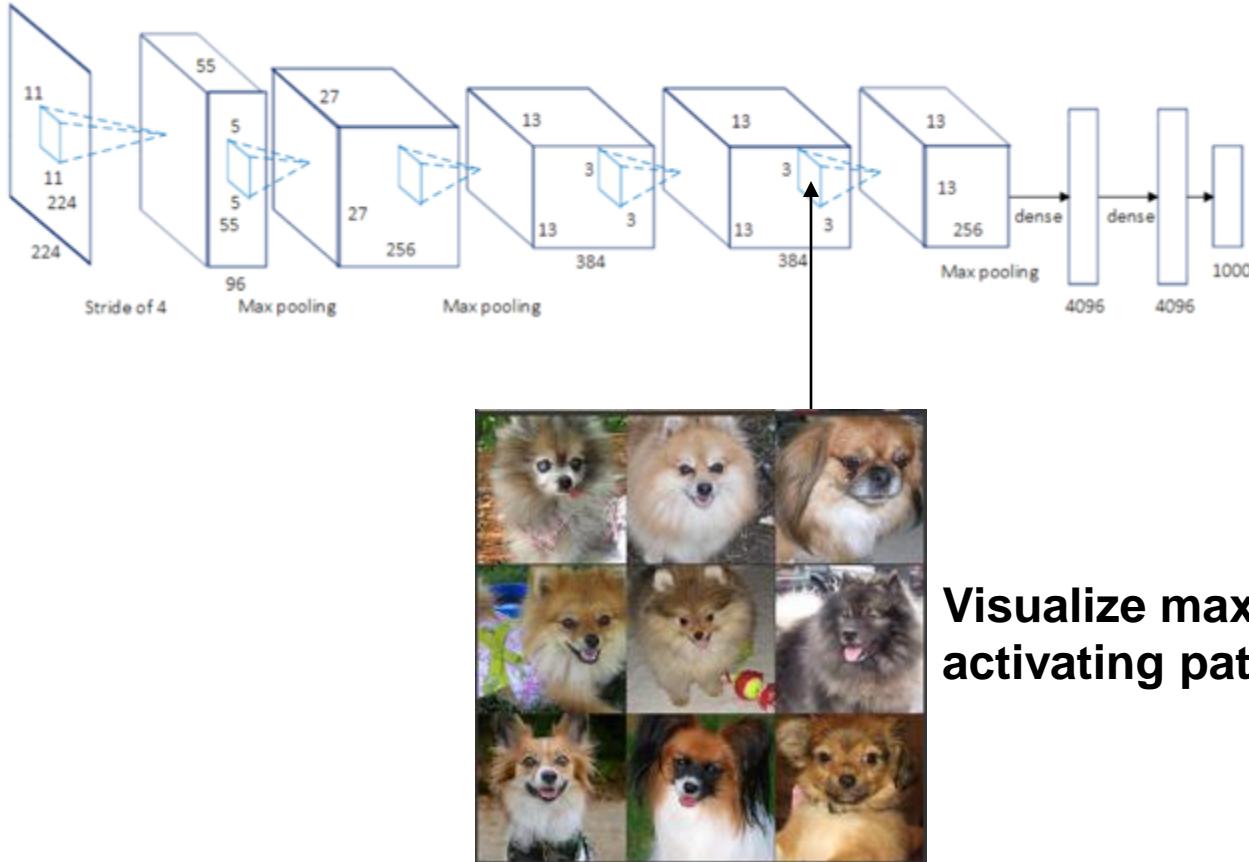
Overview and basic visualization techniques



Visualize maximally activating patches:
pick a unit; run many images through the network; visualize patches that produce the highest output values

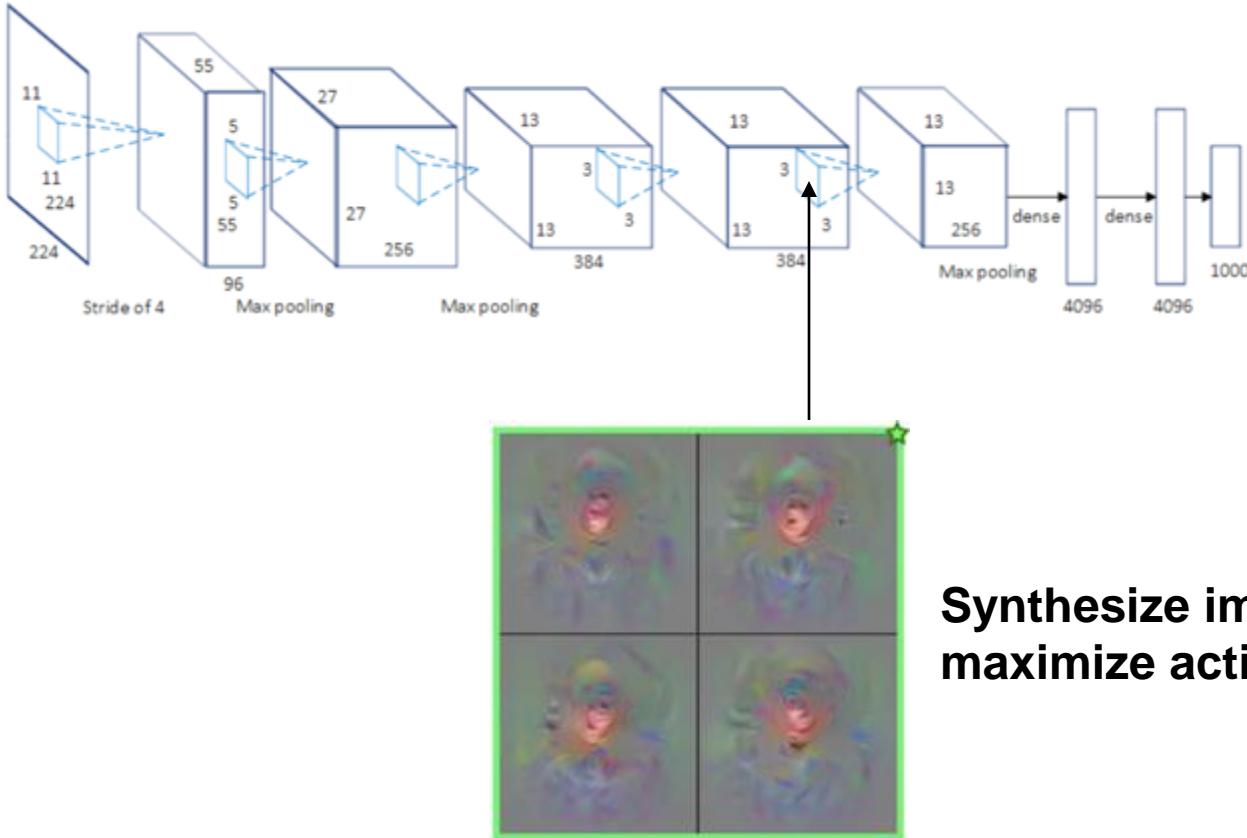


Overview and basic visualization techniques

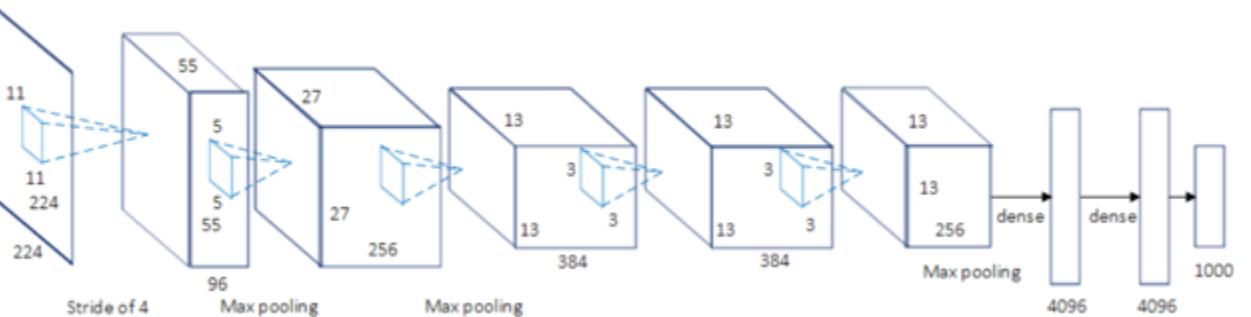


Visualize maximally activating patches

Overview and basic visualization techniques



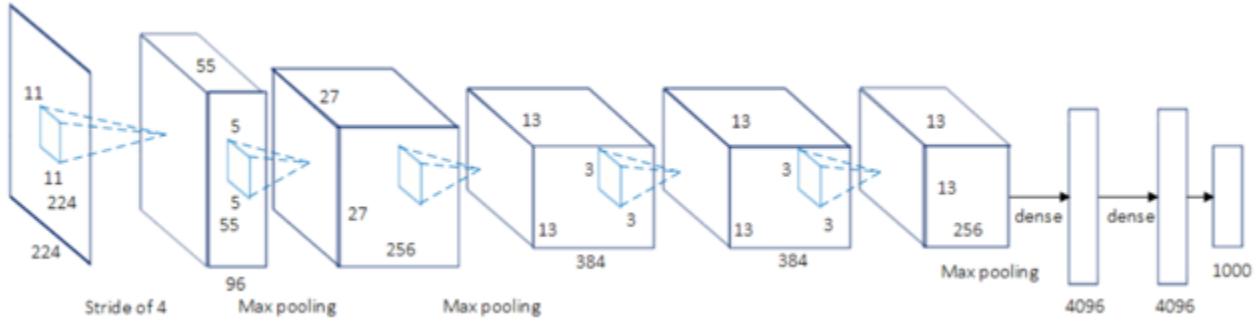
Overview and basic visualization techniques



What about FC layers?

Visualize nearest neighbor images according to activation vectors

Overview and basic visualization techniques



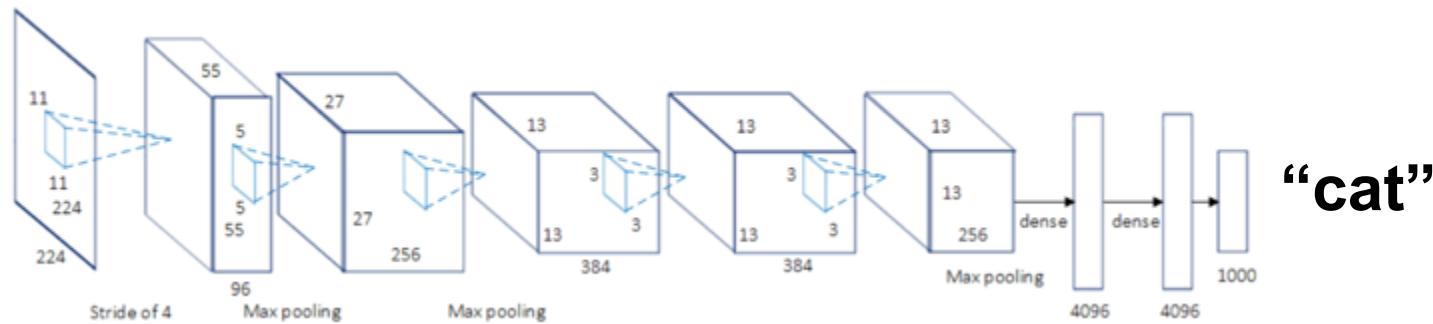
What about FC layers?

Fancy dimensionality reduction, e.g., [t-SNE](#)

Source: [Andrej Karpathy](#)

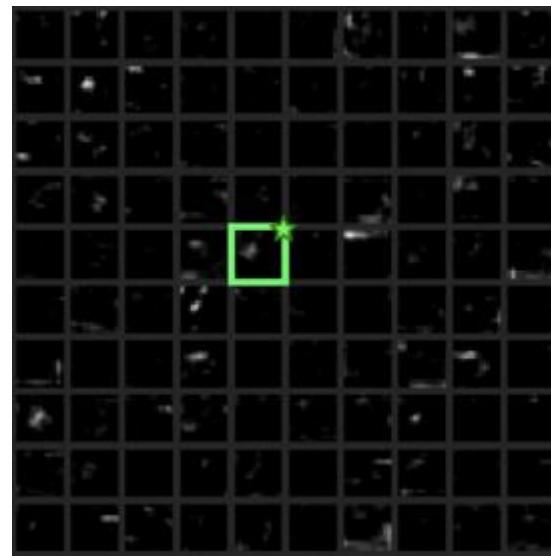
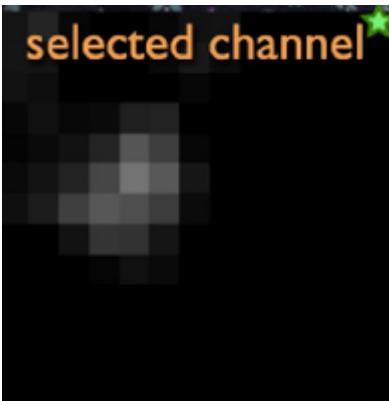
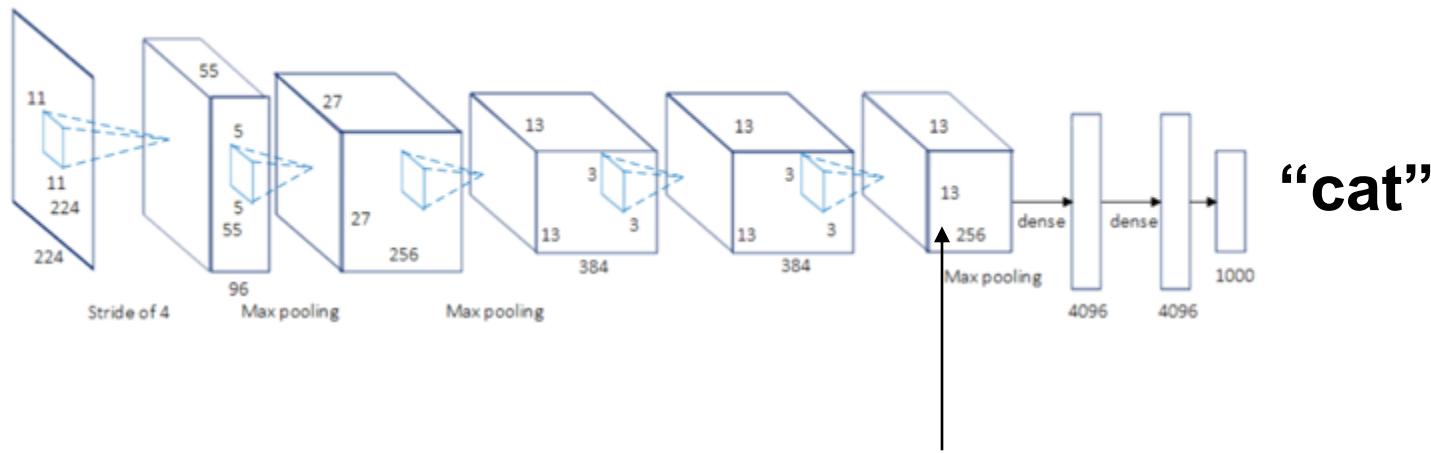
Overview and basic visualization techniques

Given: a particular input image



Overview and basic visualization techniques

Given: a particular input image

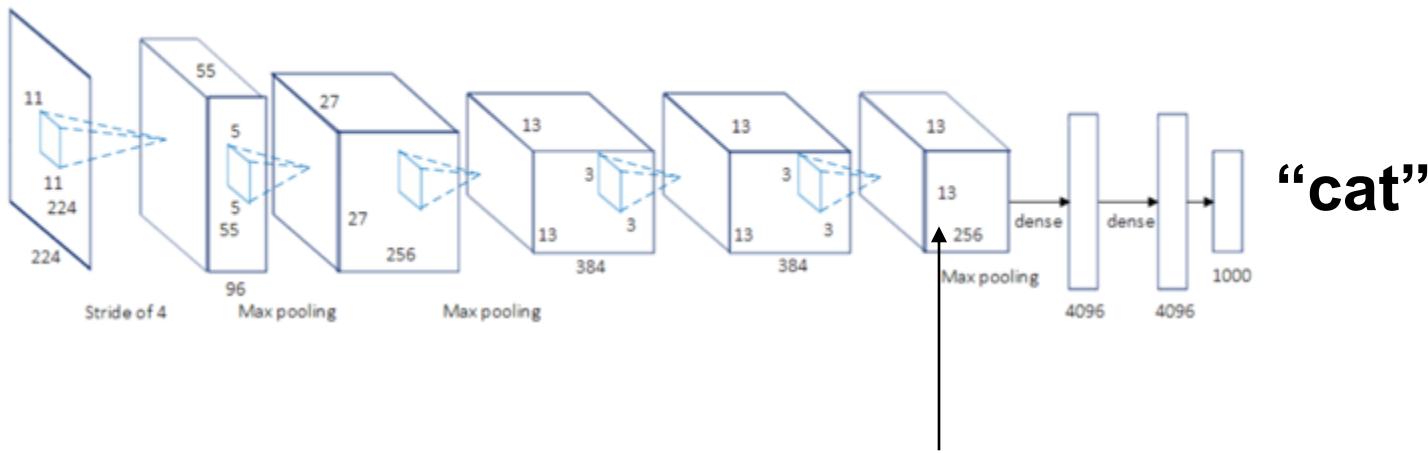


[Source](#)

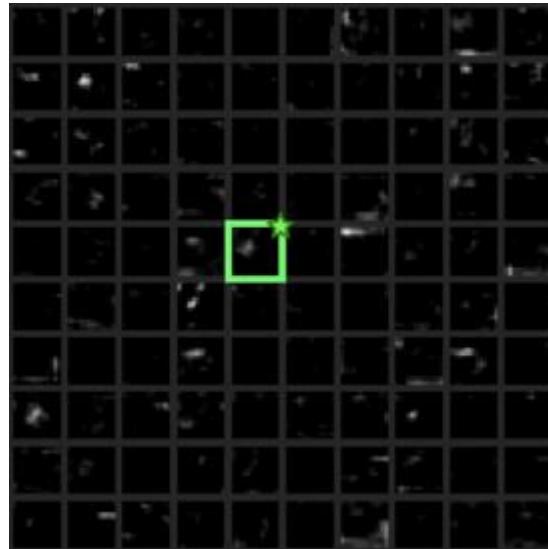
Visualize activations
for this image

Overview and basic visualization techniques

Given: a particular input image



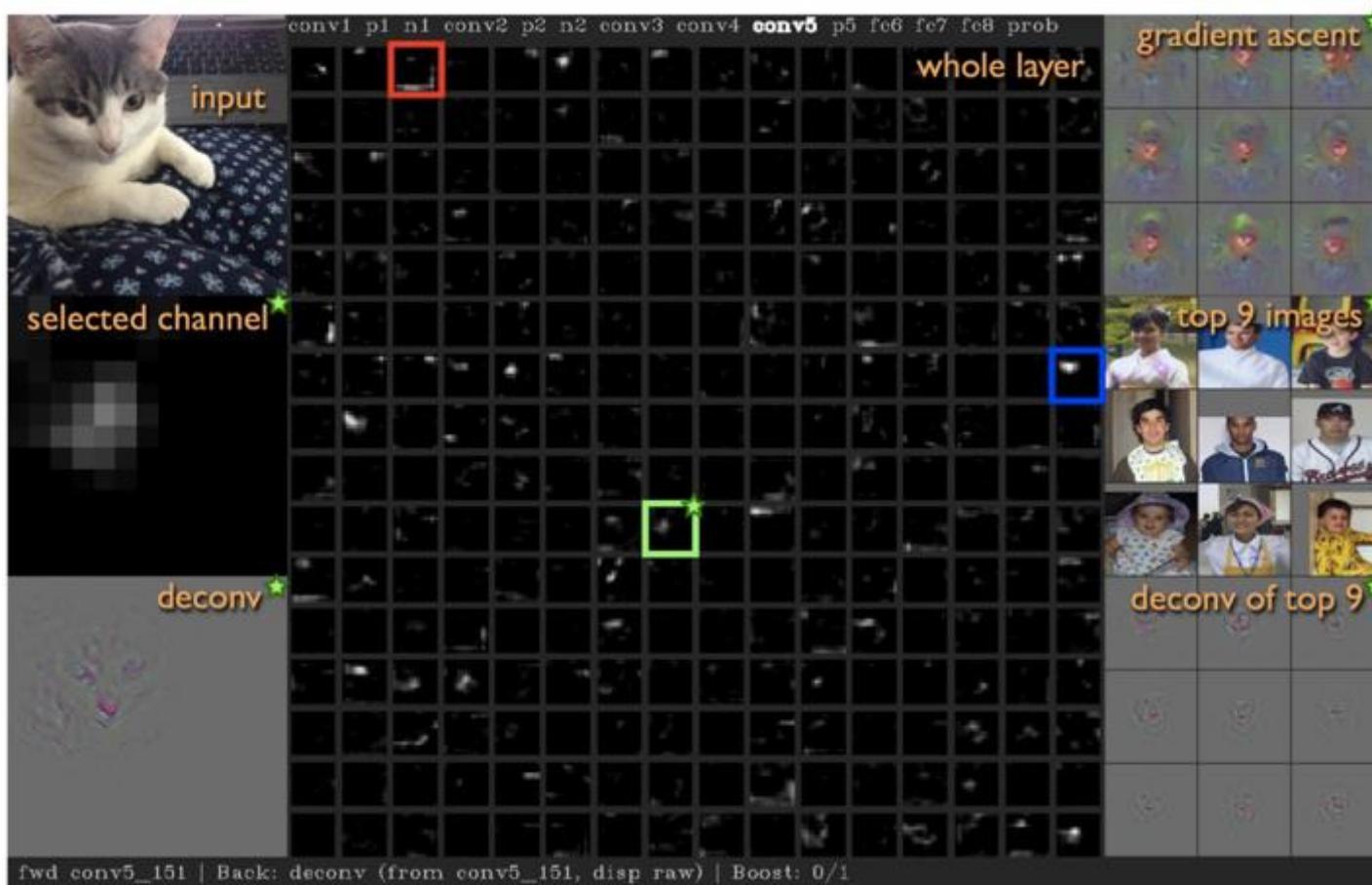
Visualize pixel values responsible for the activation



[Source](#)

Visualize activations for this image

Deep visualization toolbox



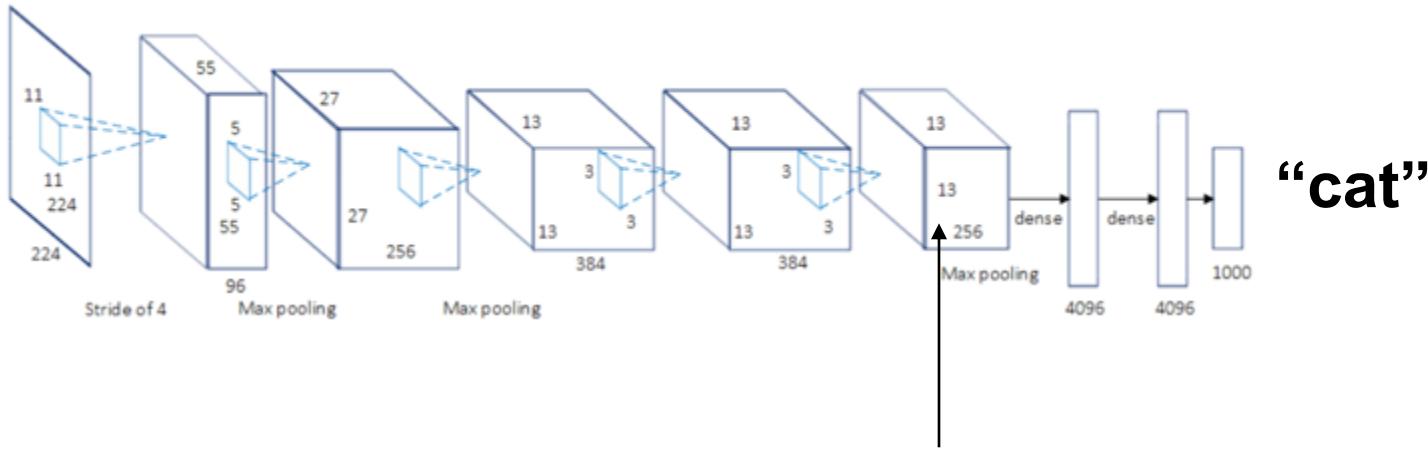
[YouTube video](#)

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, [Understanding neural networks through deep visualization](#), ICML DL workshop, 2015

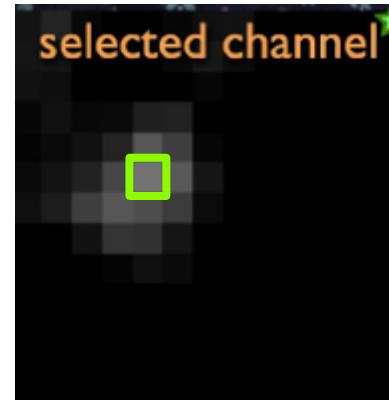
Outline

- Basic visualization techniques
- Mapping activations back to the image
- Synthesizing images to maximize activation
- Saliency maps
- Quantifying interpretability of units

Mapping activations back to pixels

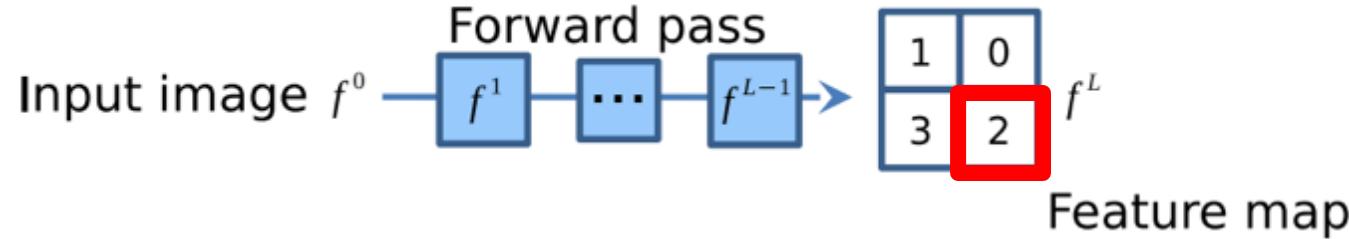


- Let's take a single value in an intermediate feature map and propagate its gradient back to the original image pixels
- What does this tell us?



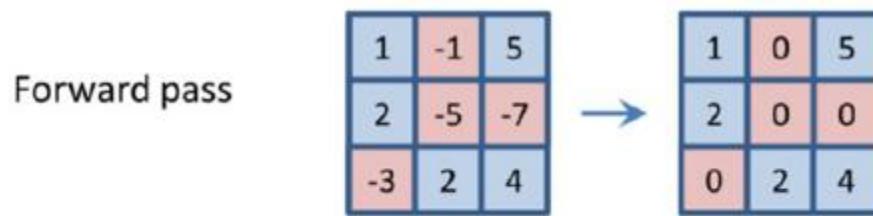
Mapping activations back to pixels

1. Forward an image through the network
2. Choose a feature map and an activation
3. Zero out all values except for the one of interest
4. Propagate that value back to the image



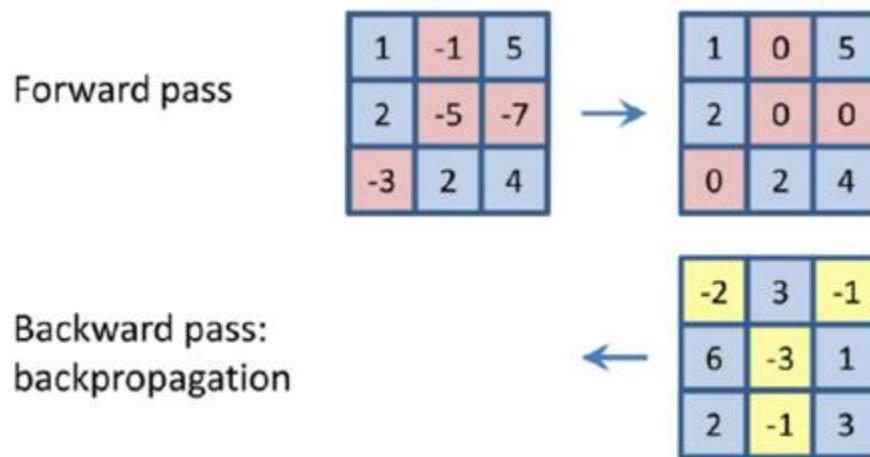
Mapping activations back to pixels

- Commonly used methods differ in how they treat the ReLU



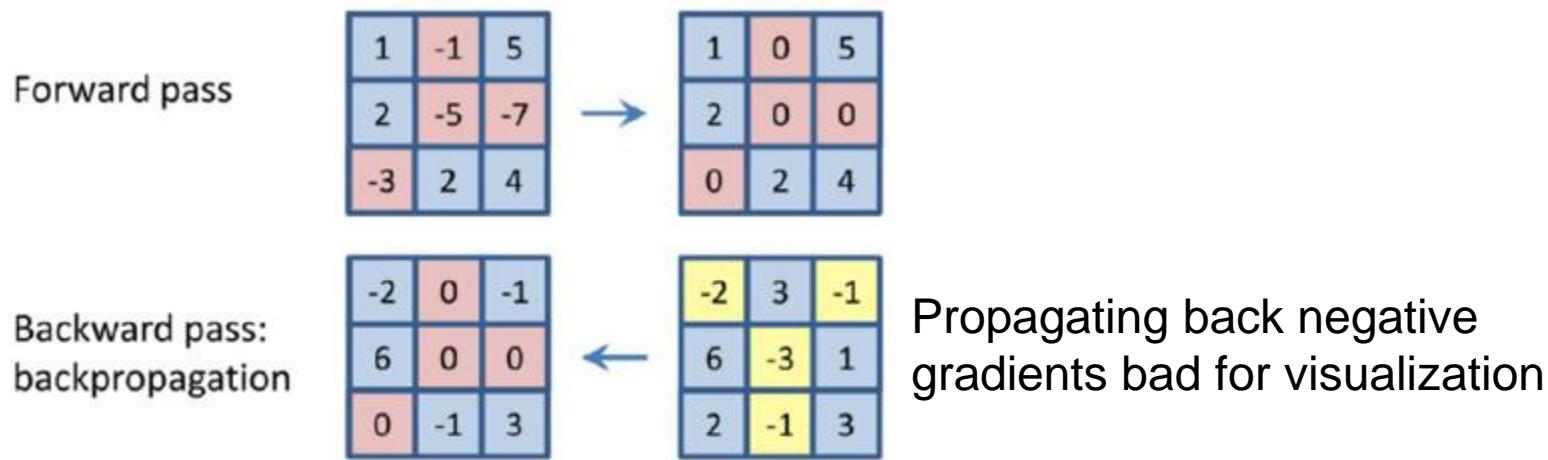
Mapping activations back to pixels

- Commonly used methods differ in how they treat the ReLU



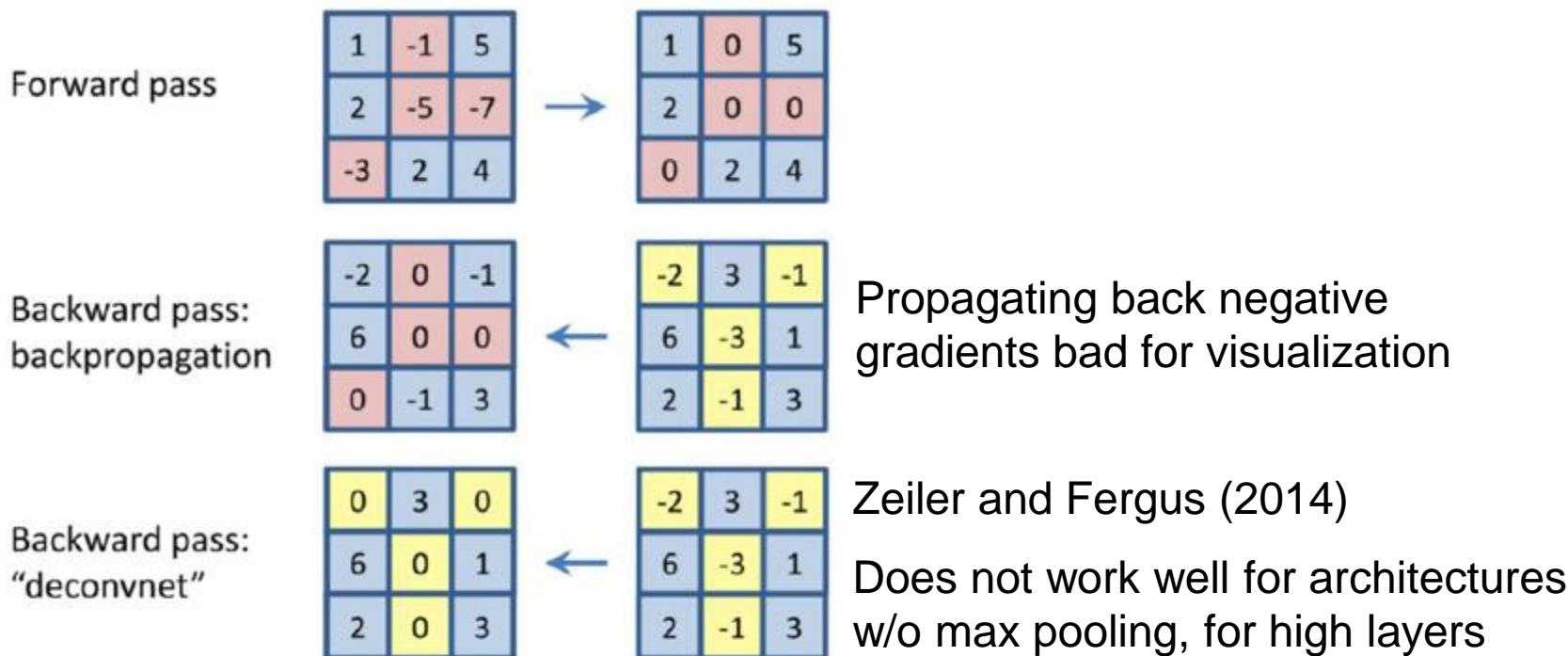
Mapping activations back to pixels

- Commonly used methods differ in how they treat the ReLU



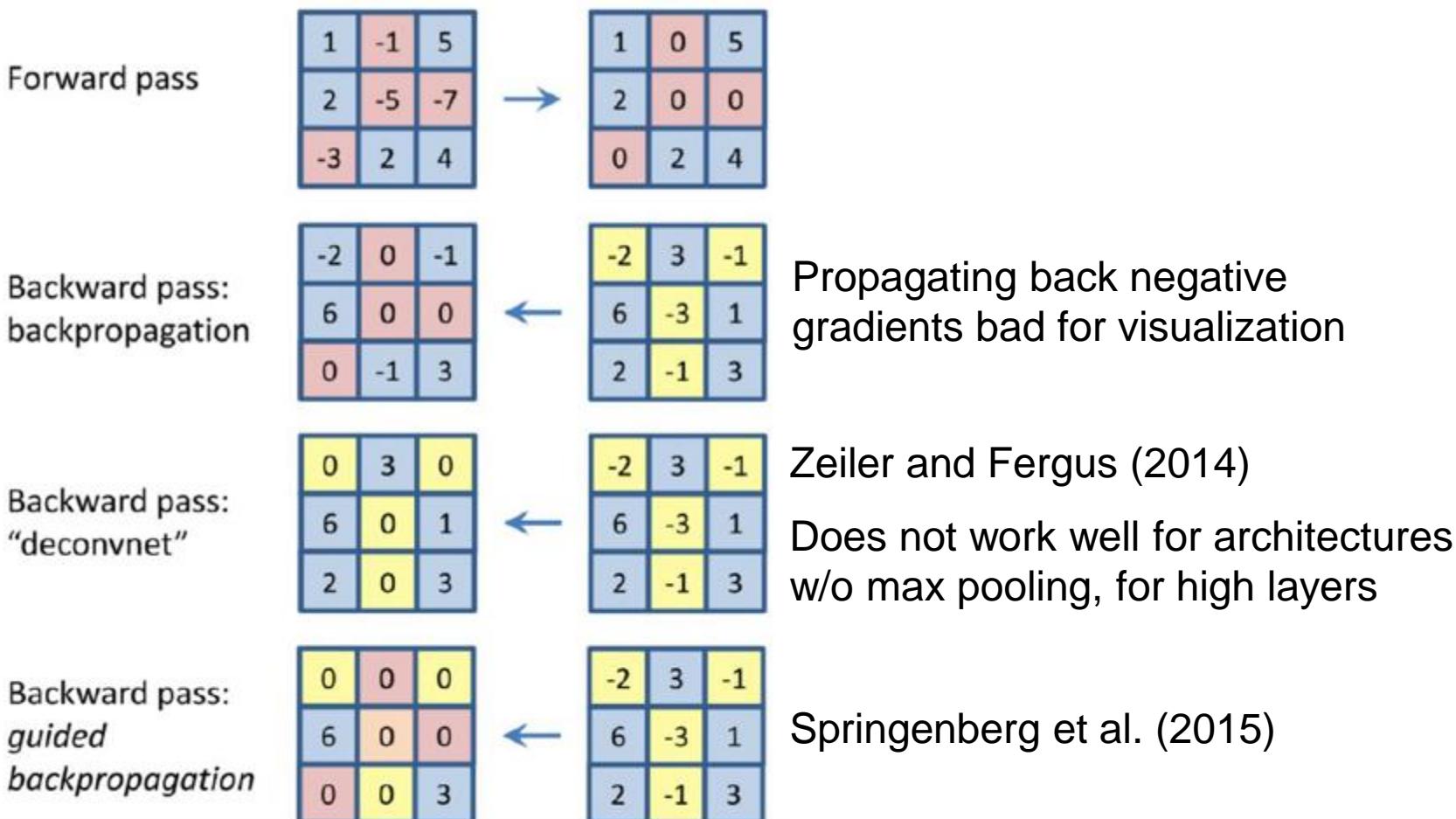
Mapping activations back to pixels

- Commonly used methods differ in how they treat the ReLU



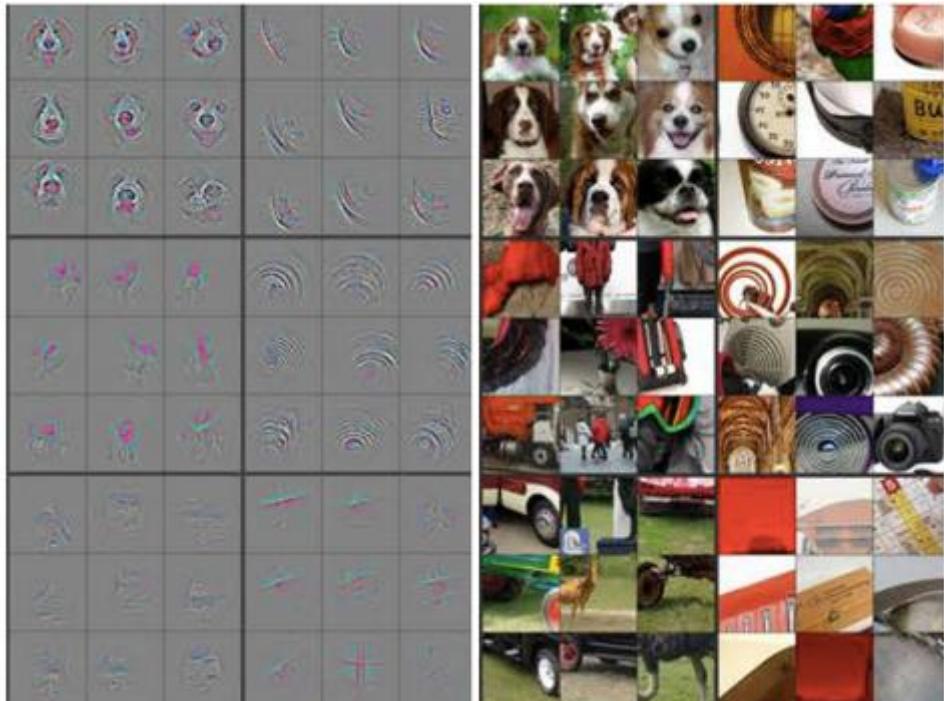
Mapping activations back to pixels

- Commonly used methods differ in how they treat the ReLU

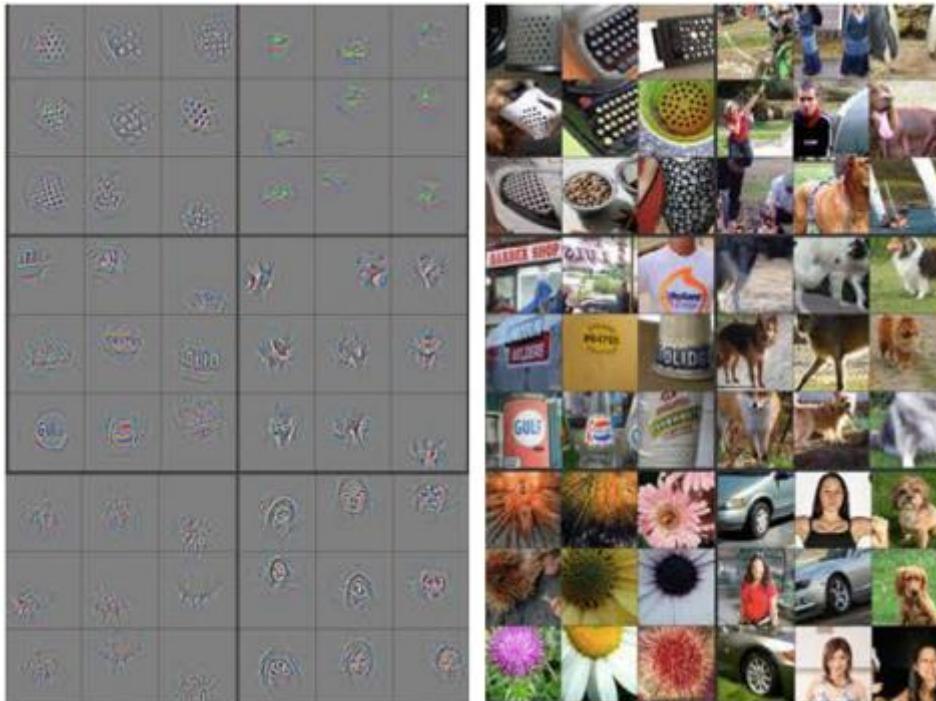


Deconvnet visualization

AlexNet Layer 4



AlexNet Layer 5



Guided backpropagation visualization

guided backpropagation



corresponding image crops



guided backpropagation



corresponding image crops



Outline

- Basic visualization techniques
- Mapping activations back to the image
- Synthesizing images to maximize activation
- Saliency maps
- Quantifying interpretability of units

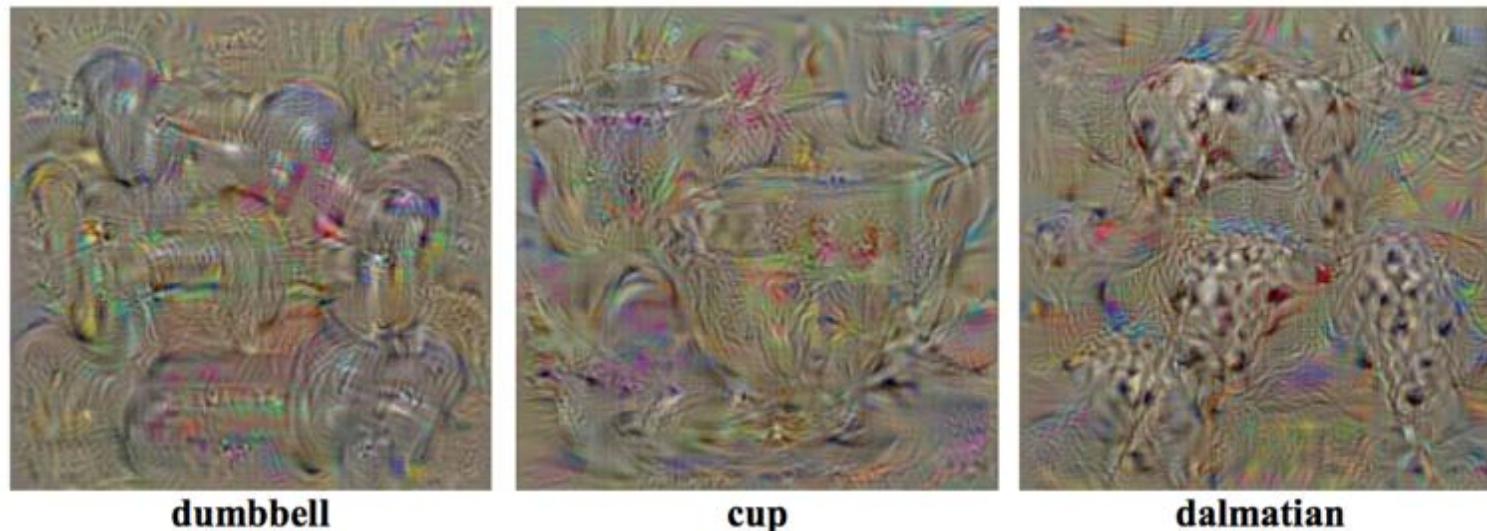
Visualization by optimization (model inversion)

- How can we synthesize images that maximize activation of a given neuron?
- Basic approach: find image x maximizing target activation $f(x)$ subject to *natural image regularization penalty* $R(x)$:

$$x^* = \arg \max_x f(x) - \lambda R(x)$$

Visualization by optimization (model inversion)

- Maximize $f(x) - \lambda R(x)$
 - $f(x)$ is score for a category *before softmax*
 - $R(x)$ is L2 regularization
 - Perform *gradient ascent* starting with zero image, add dataset mean to result



K. Simonyan, A. Vedaldi, and A. Zisserman, [Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps](#), ICLR 2014

Visualization by optimization (model inversion)

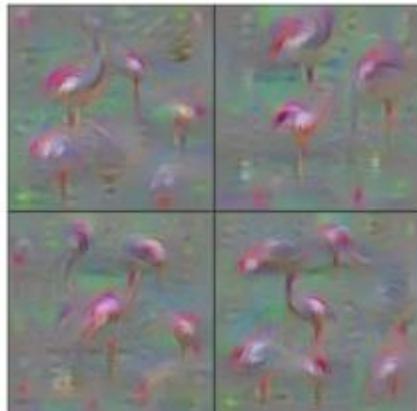
- Alternative approach to regularization:
at each step of gradient ascent, apply operator r that regularizes the image:

$$x \leftarrow r \left(x + \eta \frac{\partial f}{\partial x} \right)$$

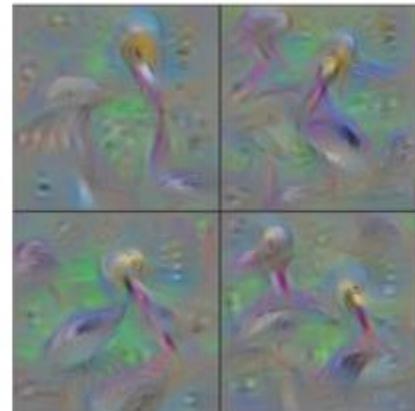
- Combination that gives good-looking results:
 - L2 decay
 - Gaussian blur (every few iterations)
 - Clip pixel values with small magnitude
 - Clip pixel values with small contribution to the activation (estimated by product of pixel value and gradient)

Visualization by optimization (model inversion)

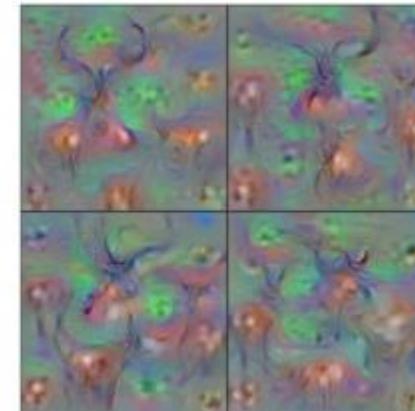
- Example visualizations:



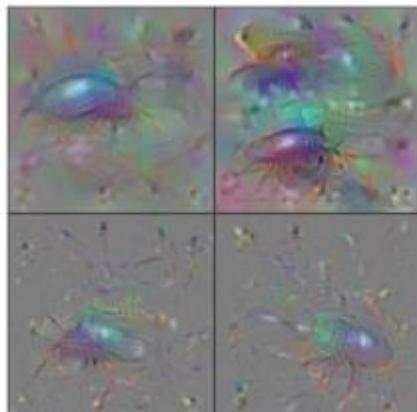
Flamingo



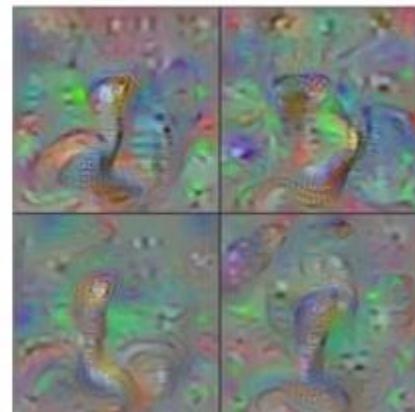
Pelican



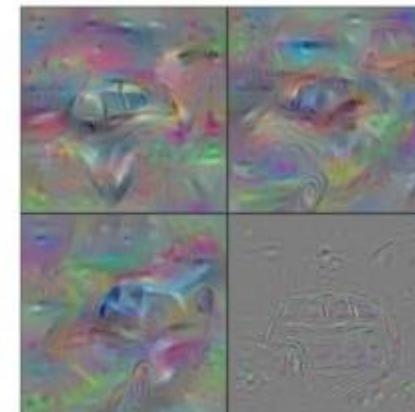
Hartebeest



Ground Beetle



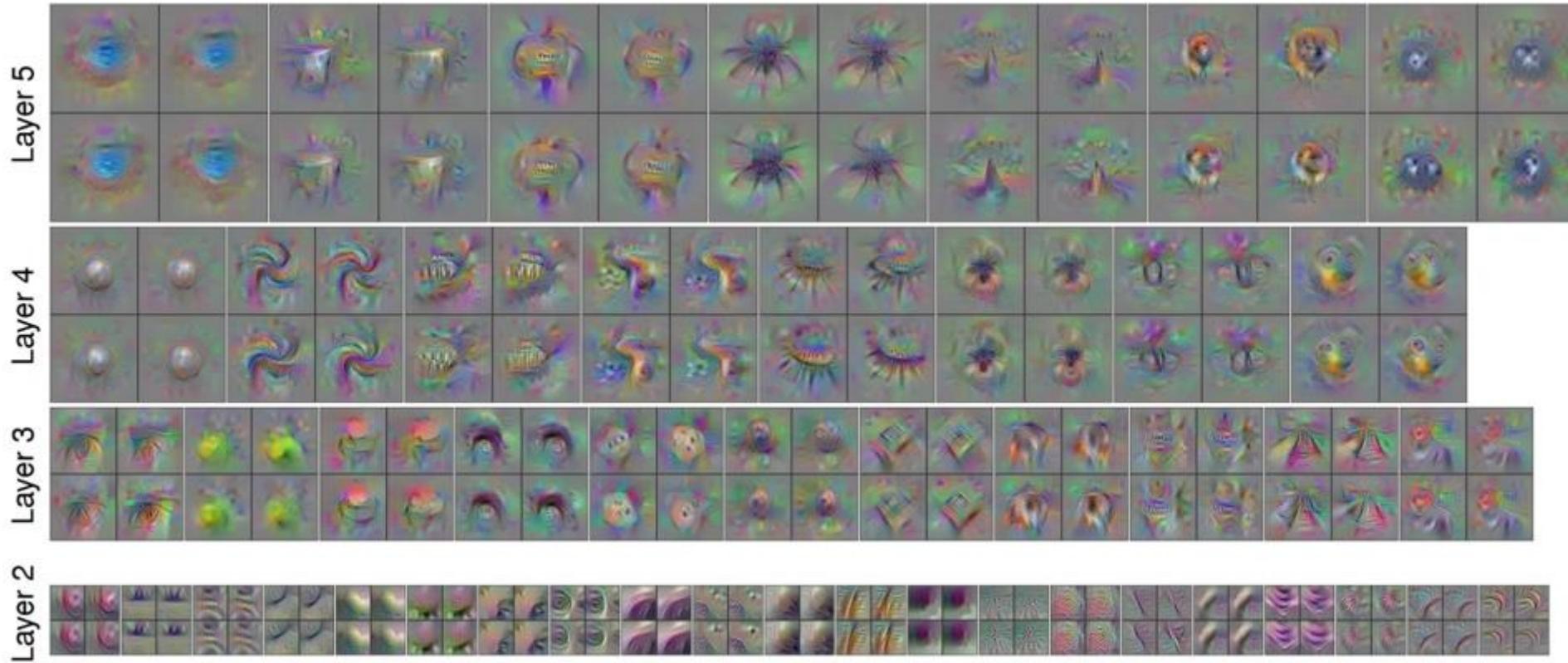
Indian Cobra



Station Wagon

Visualization by optimization (model inversion)

- Example visualizations of intermediate features:



J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, [Understanding neural networks through deep visualization](#), ICML DL workshop, 2015

Multifaceted feature visualization

- Key idea: most neurons in high layers respond to a mix of different patterns or “facets”
- For coherent visualizations, zero in on individual facets



A. Nguyen, J. Yosinski, J. Clune, [Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks](#), ICML workshop, 2016

Multifaceted feature visualization

- Key idea: most neurons in high layers respond to a mix of different patterns or “facets”
- For coherent visualizations, zero in on individual facets
- Algorithm:
 - Cluster FC activations of training images to identify facets
 - For each facet, initialize optimization with mean image of that facet
 - To attempt to produce image of a single object, use *center-biased regularization* (start with blurry image, gradually increase resolution and update center pixels more than edge pixels)

Multifaceted feature visualization



A. Nguyen, J. Yosinski, J. Clune, [Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks](#), ICML workshop, 2016

Google DeepDream

Choose an image and a layer in a CNN; repeat:

1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
 - Equivalent to maximizing $\sum_i f_i^2(x)$
3. Backward: Compute gradient w.r.t. image
4. Update image (with some tricks)

Source: [Stanford CS231n](#)

<https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

<https://deeplearning.net/deepdream/>

Outline

- Basic visualization techniques
- Mapping activations back to the image
- Synthesizing images to maximize activation
- Saliency maps
- Quantifying interpretability of units

Saliency maps

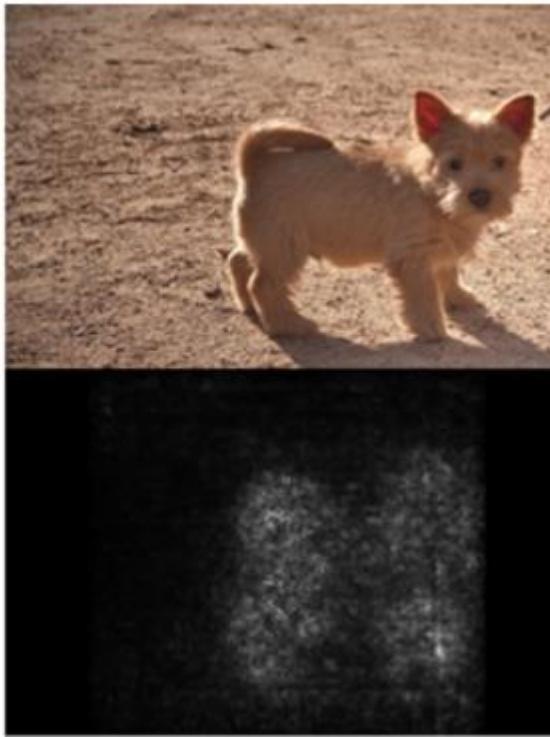
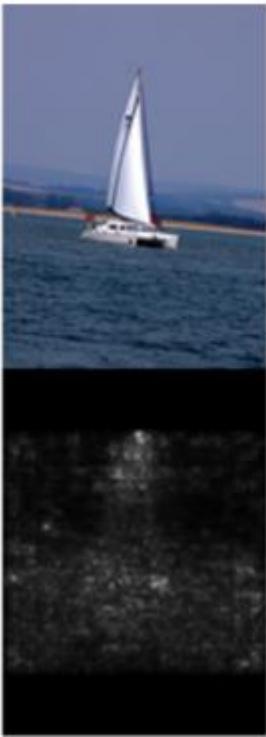
- Which parts of the image played the most important role in the network's decision?

Prediction: “car” 64%



“White box” saliency via gradients

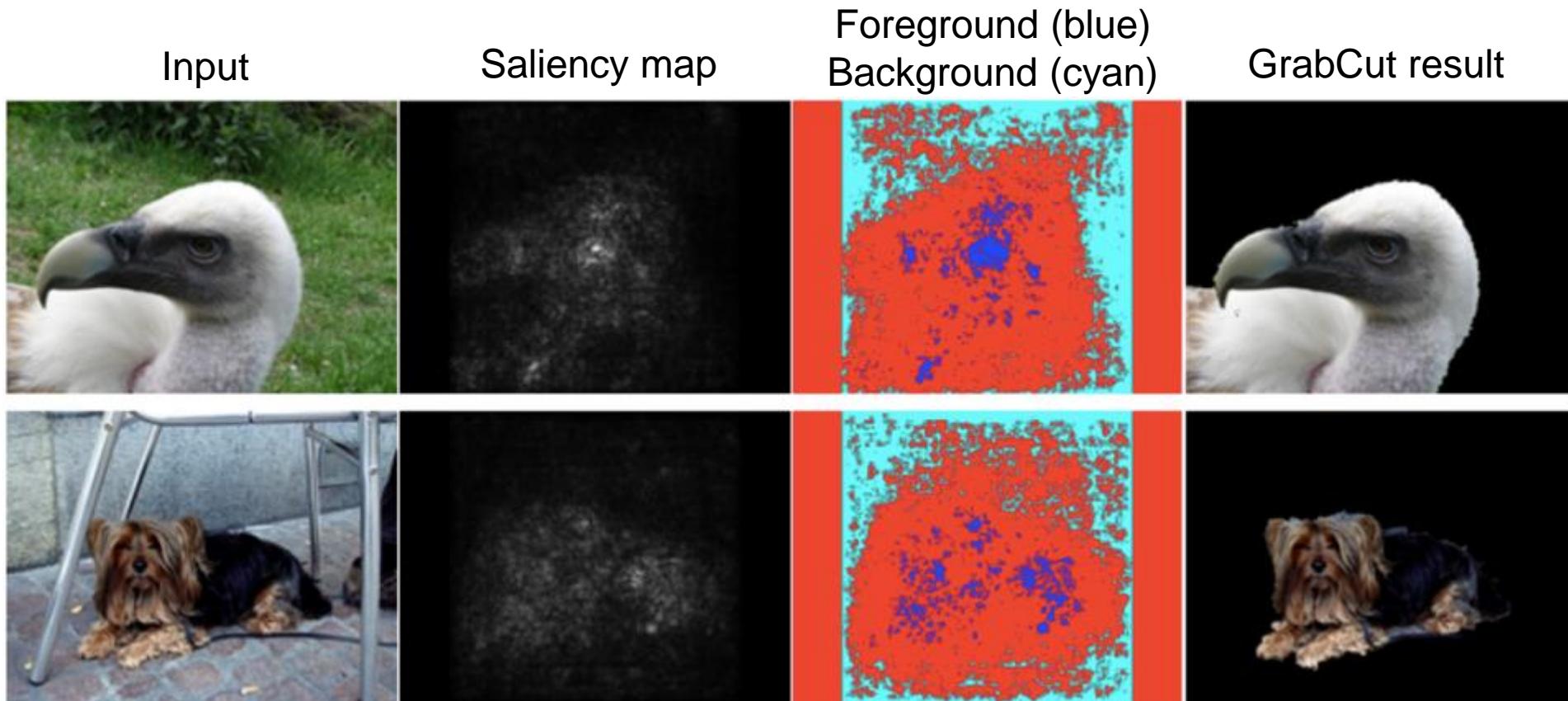
- Backpropagate gradient of class score (before softmax) to the image, display max of absolute values across color channels



K. Simonyan, A. Vedaldi, and A. Zisserman, [Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps](#), ICLR 2014

“White box” saliency via gradients

- Can be used for *weakly supervised segmentation*:



K. Simonyan, A. Vedaldi, and A. Zisserman, [Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps](#), ICLR 2014

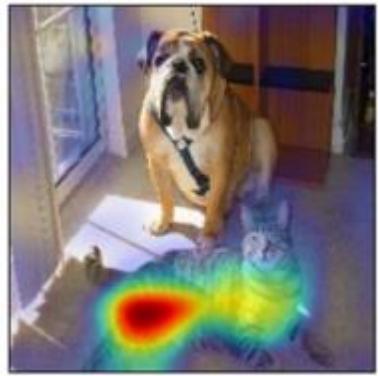
Gradient-weighted class activation mapping (Grad-CAM)



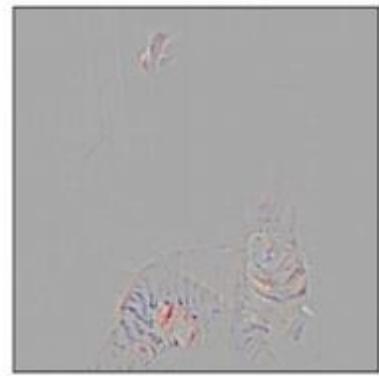
(a) Original Image



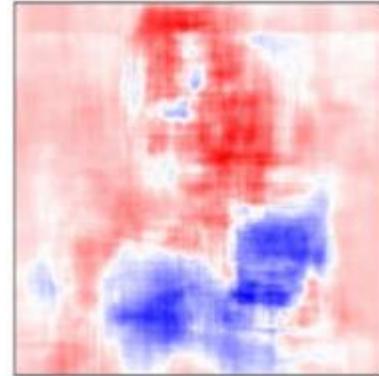
(b) Guided Backprop 'Cat'



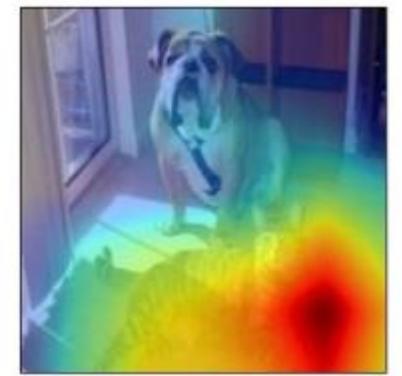
(c) Grad-CAM 'Cat'



(d) Guided Grad-CAM 'Cat'



(e) Occlusion map for 'Cat'



(f) ResNet Grad-CAM 'Cat'



(g) Original Image



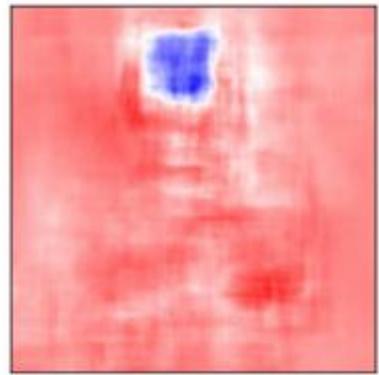
(h) Guided Backprop 'Dog'



(i) Grad-CAM 'Dog'



(j) Guided Grad-CAM 'Dog'



(k) Occlusion map for 'Dog'



(l) ResNet Grad-CAM 'Dog'

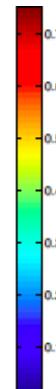
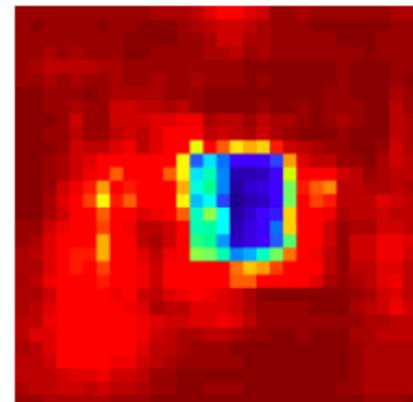
“Black box” saliency via masking

- Slide square occluder across image, see how class score changes

Input image

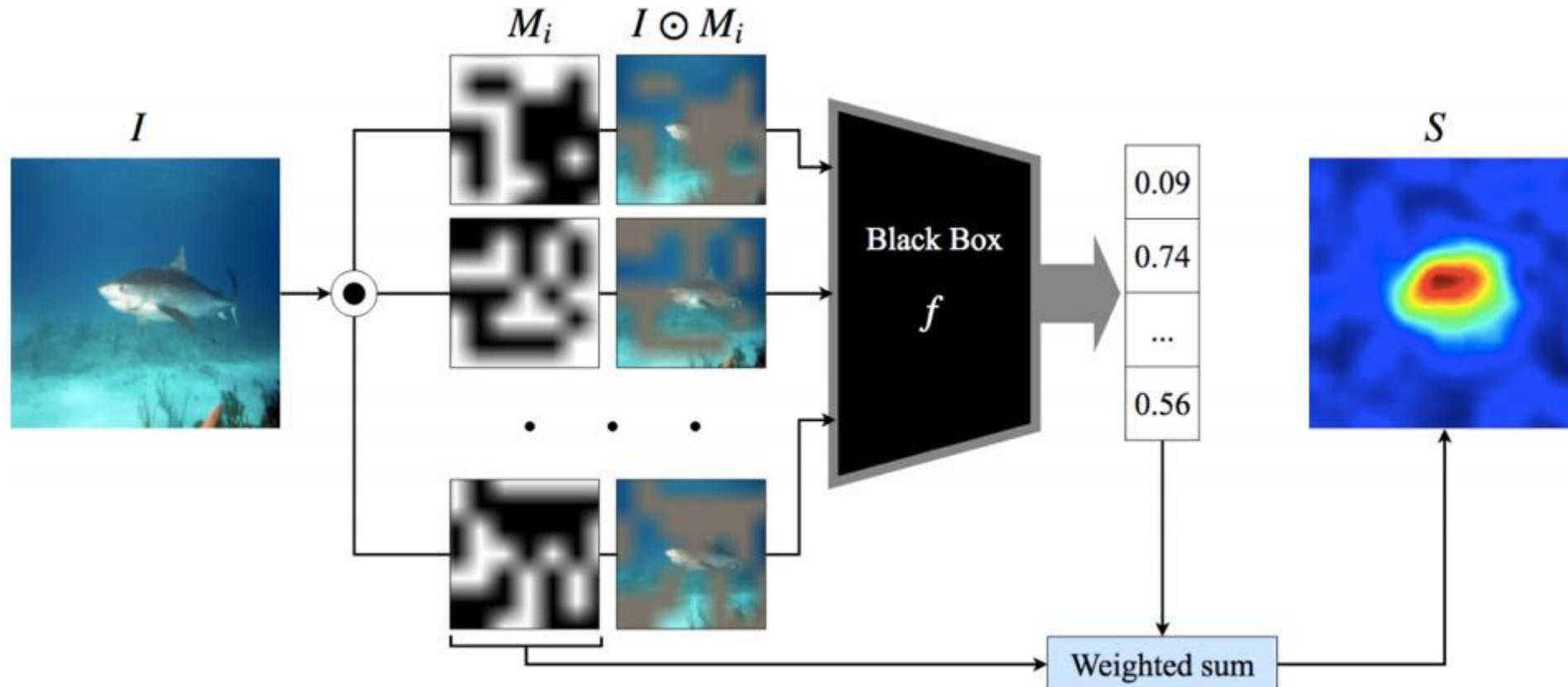


Correct class probability as function of occluder position



“Black box” saliency via masking

- Saliency of a class at a pixel is expected score for that class over all masks where the pixel is visible

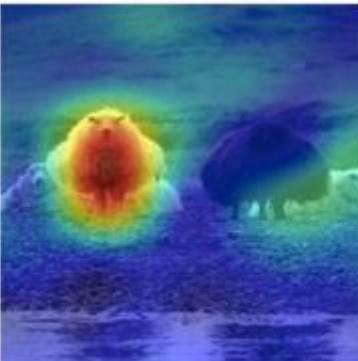


“Black box” saliency via masking

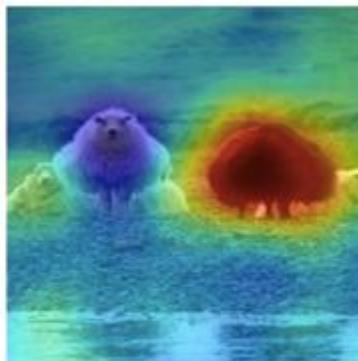
- Saliency of a class at a pixel is expected score for that class over all masks where the pixel is visible



(a) Sheep - 26%, Cow - 17%



(b) Importance map of ‘sheep’



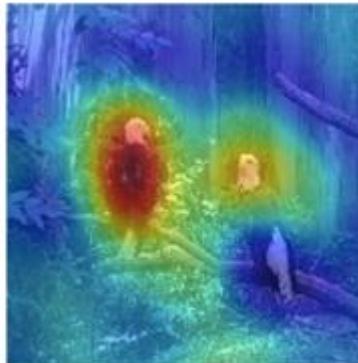
(c) Importance map of ‘cow’



(d) Bird - 100%, Person - 39%



(e) Importance map of ‘bird’



(f) Importance map of ‘person’

“Black box” saliency via masking

- Application: detecting model/dataset bias

Prediction: “cow” 76%



“Black box” saliency via masking

- Application: detecting model/dataset bias



Baseline: A **man** sitting at a desk
with a laptop computer.



Improved model: A **woman** sitting
in front of a laptop computer.

Outline

- Basic visualization techniques
- Mapping activations back to the image
- Synthesizing images to maximize activation
- Saliency maps
- Quantifying interpretability of units

Quantifying interpretability of units

- From the beginning, people have observed that many units in higher layers seem to fire on meaningful concepts
- But how can we quantify this?

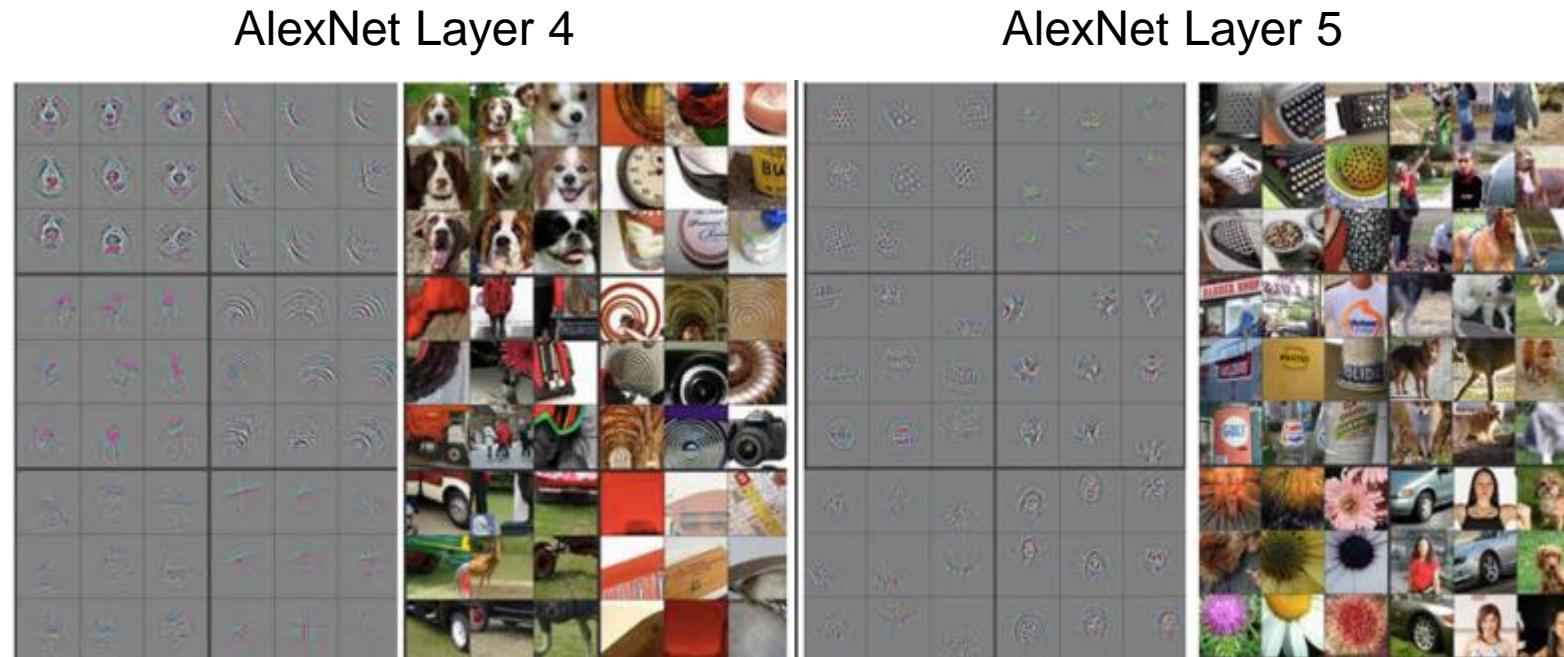
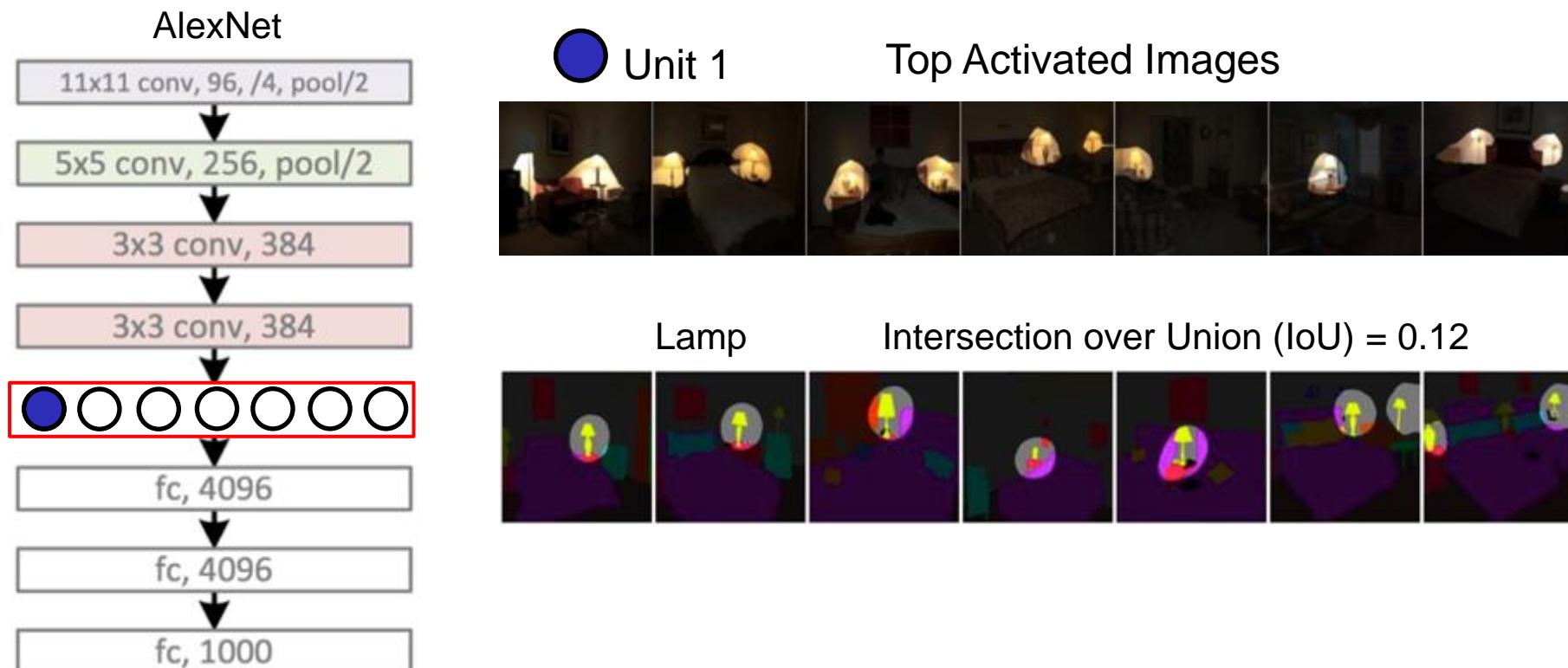


Figure: Zeiler & Fergus

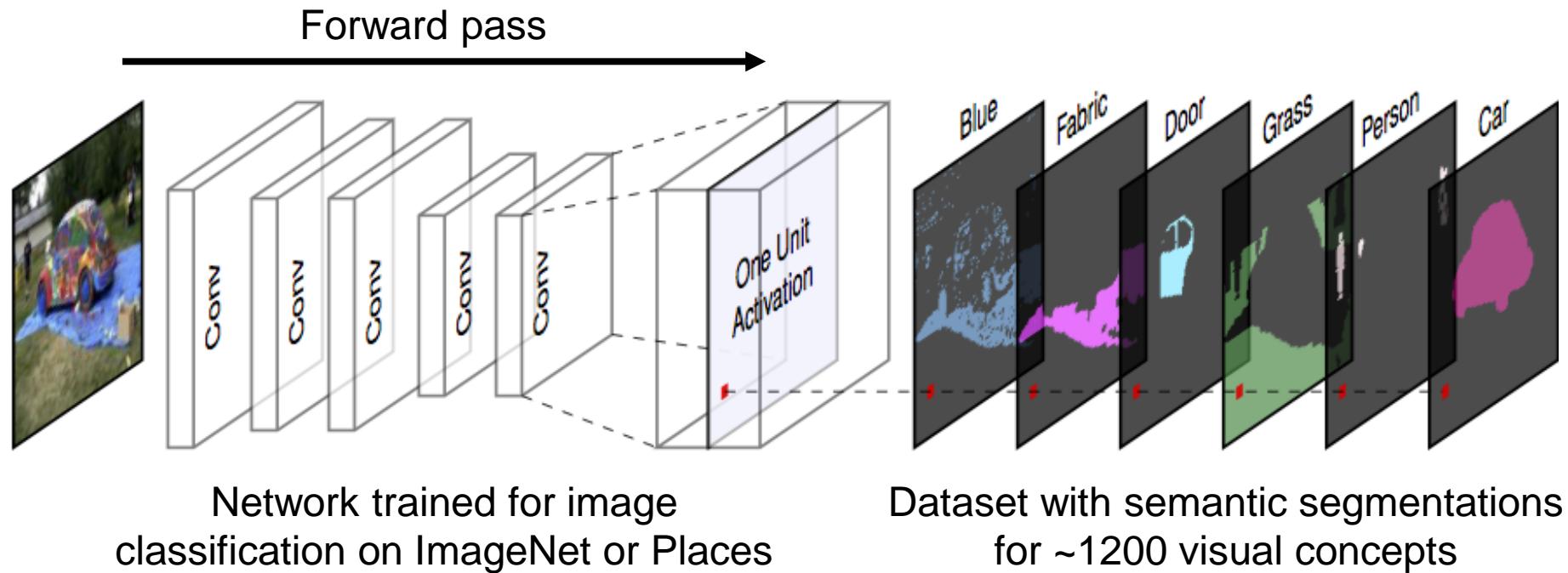
Quantifying interpretability of units

- For a given unit, measure the overlap between areas of high activation and semantic segmentations for a large set of visual concepts



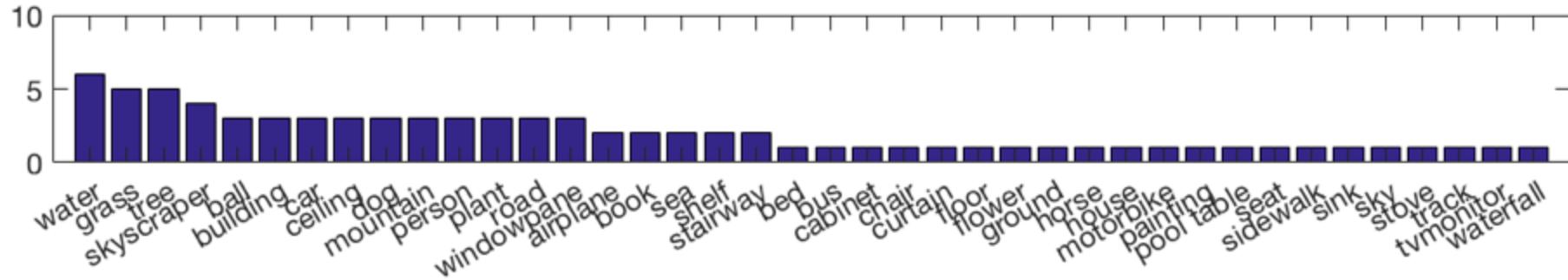
Quantifying interpretability of units

- For a given unit, measure the overlap between areas of high activation and semantic segmentations for a large set of visual concepts



Quantifying interpretability of units

Histogram of object detectors for Places AlexNet conv5 units
81/256 units with IoU > 0.04



conv5 unit 79

car (object)

IoU=0.13



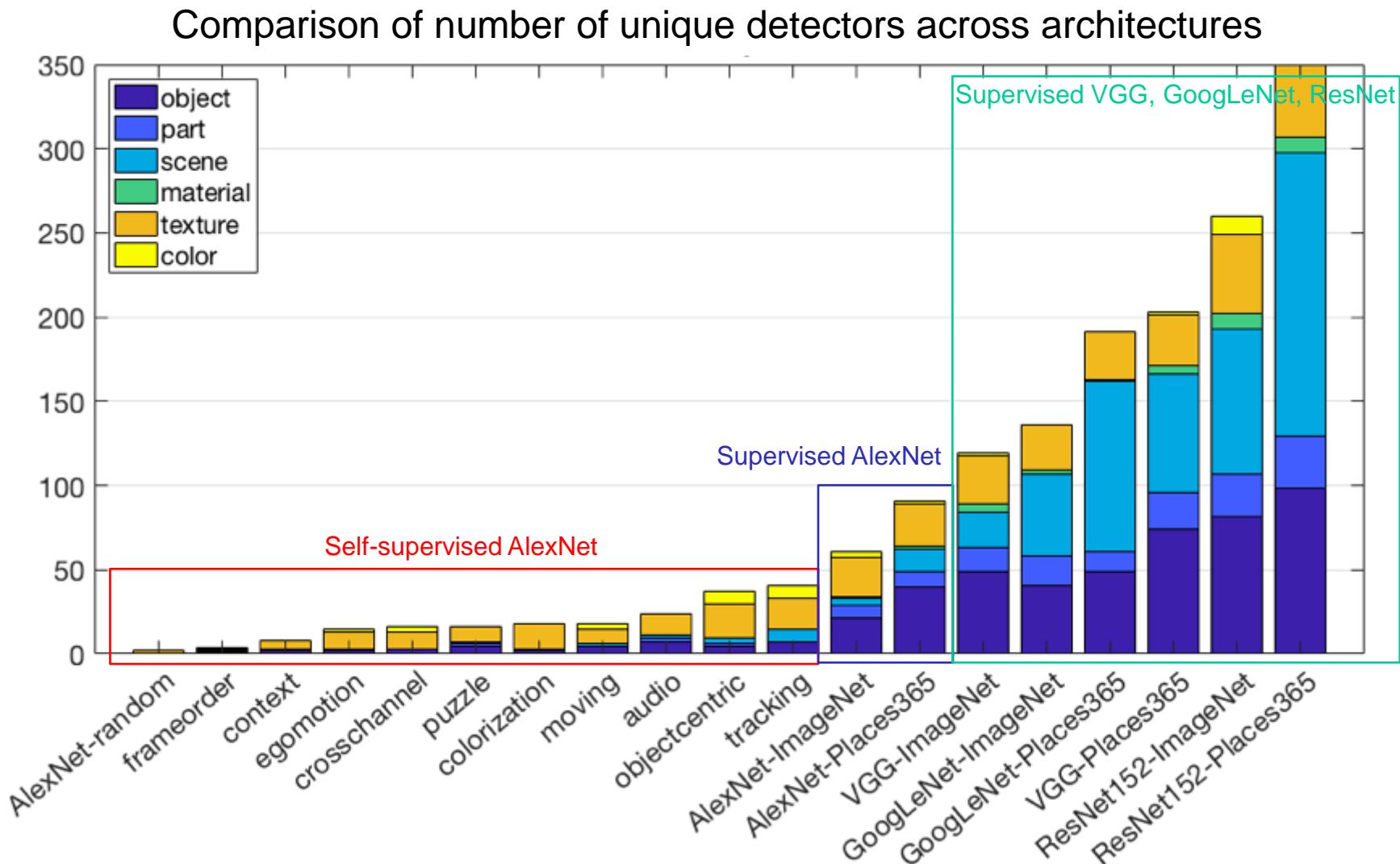
conv5 unit 107

road (object)

IoU=0.15



Quantifying interpretability of units



Summary

- Basic visualization techniques
 - Showing weights, top activated patches, nearest neighbors
- Mapping activations back to the image
 - Deconvolution
 - Guided back-propagation
- Synthesizing images to maximize activation
 - Gradient ascent with natural image regularization
- Saliency maps
 - “White box” vs. “black box”
- Explainability/interpretability
 - Explaining network decisions, detecting bias
 - Quantifying interpretability of intermediate units

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Fooling neural networks



[Image source](#)

Generating preferred inputs

- Recall: we can use gradient ascent to generate weird-looking images to maximize activation of a given unit



dumbbell



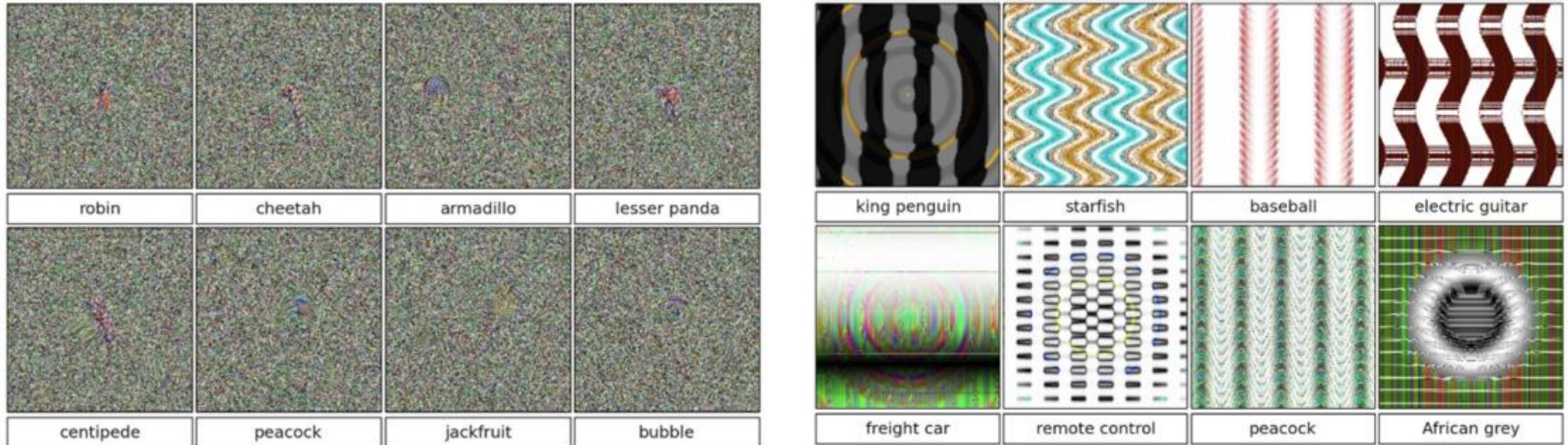
cup



dalmatian

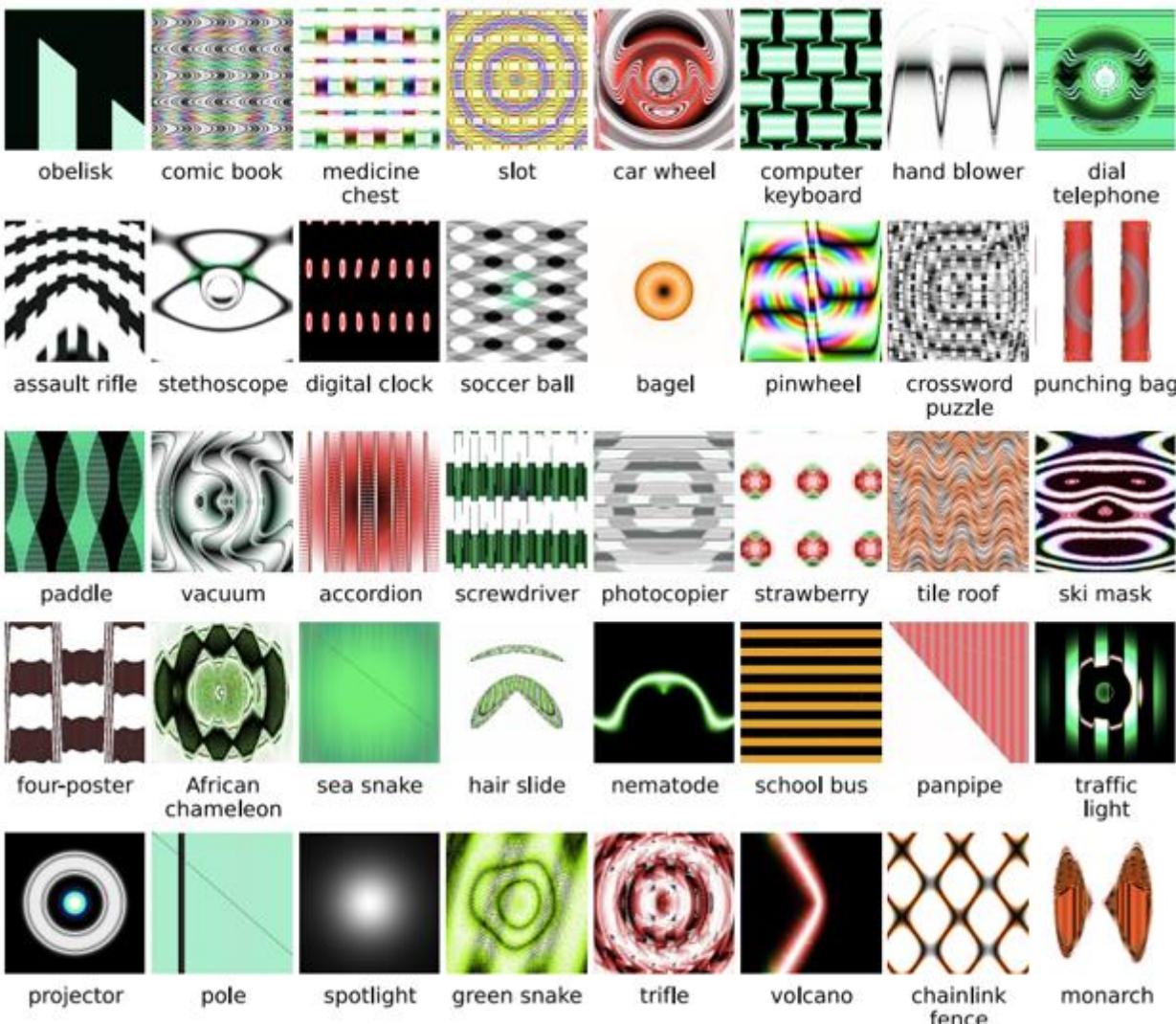
Generating preferred inputs

- Related finding: it is easy to generate meaningless images that will be classified as any given class with high confidence



A. Nguyen, J. Yosinski, J. Clune, [Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images](#), CVPR 2015

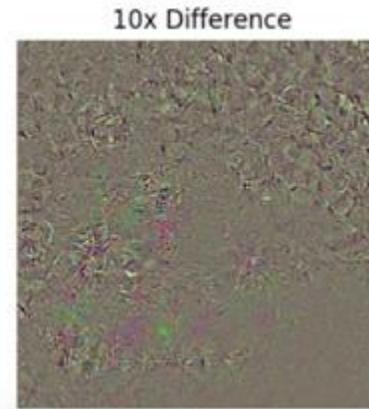
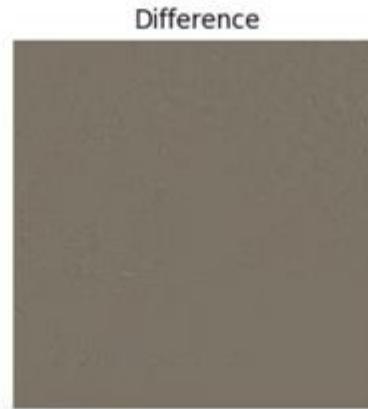
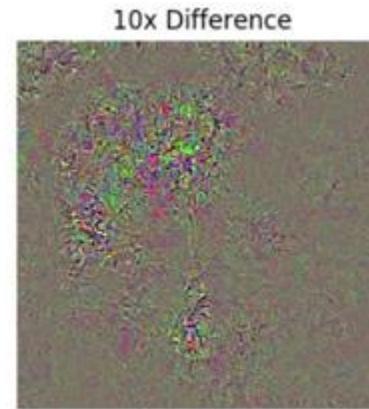
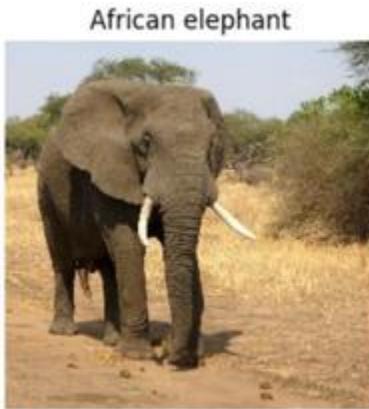
Generating preferred inputs



A. Nguyen, J. Yosinski, J. Clune, [Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images](#), CVPR 2015

Adversarial examples

- We can “fool” a neural network by imperceptibly perturbing an input image so it is misclassified



Adversarial examples: Outline

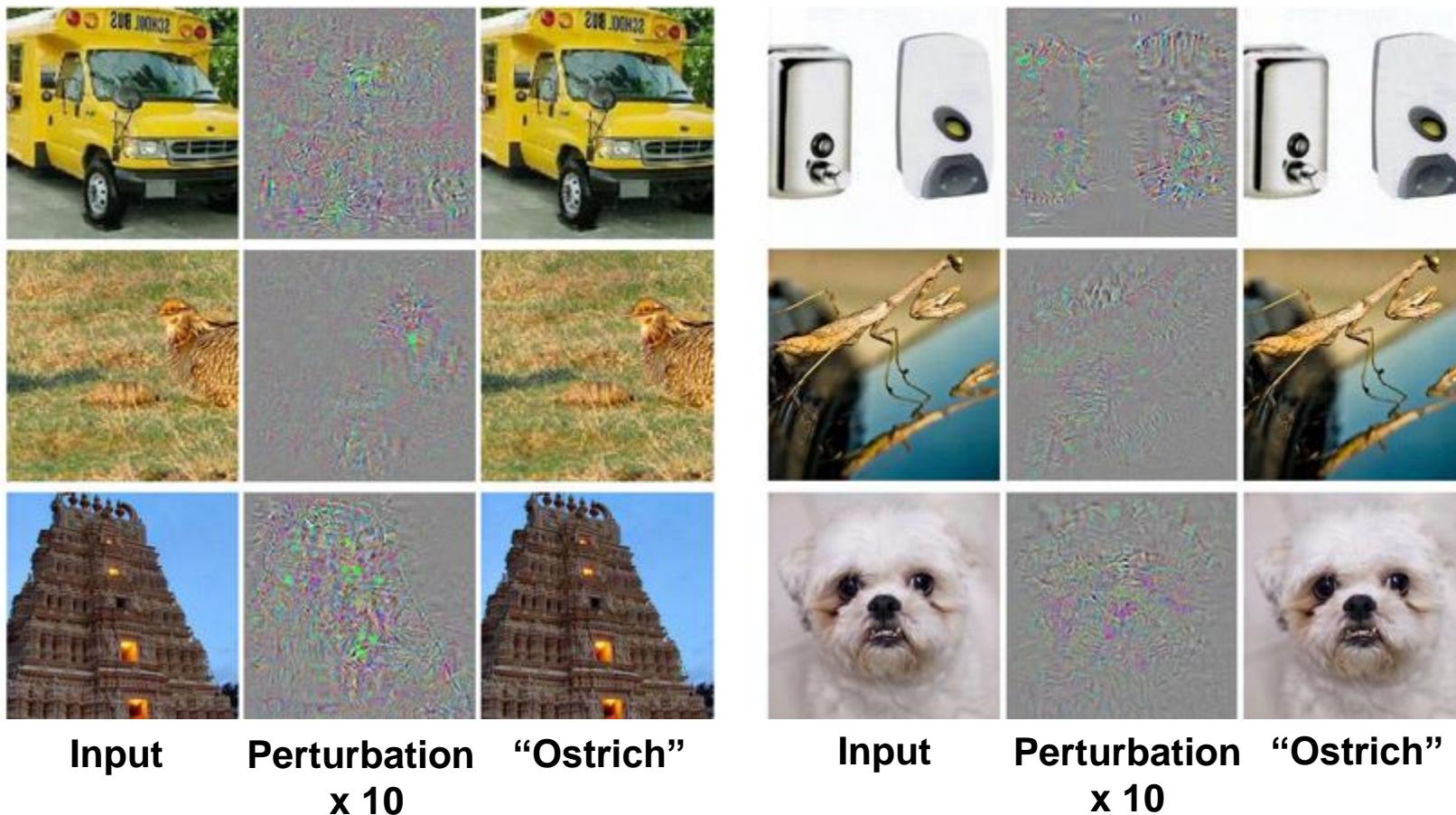
- Generating adversarial examples
 - Finding smallest “fooling” transformation
 - Gradient ascent
 - Fast gradient sign, iterative variants
 - Universal adversarial perturbations
- Why are neural networks easy to fool?
- Defending against adversarial examples
 - Adversarial training
 - Learning to reject adversarial examples
 - Robust architectures
 - Image pre-processing
- “Open” topics
 - Broadening the scope of adversarial examples
 - Adversarial examples and human perception

Finding the smallest adversarial perturbation

- Start with correctly classified image \mathbf{x}
- Find perturbation \mathbf{r} minimizing $\|\mathbf{r}\|_2$ such that
 - $\mathbf{x} + \mathbf{r}$ is misclassified (or classified as specific target class)
 - All values of $\mathbf{x} + \mathbf{r}$ are in the valid range
- This is constrained non-convex optimization, which the authors solve with L-BFGS
 - **Limited-memory BFGS** (L-BFGS or LM-BFGS) is an optimization algorithm in the family of quasi-Newton methods

Finding the smallest adversarial perturbation

- Sample results:



Gradient ascent

- Rather than searching for the smallest possible perturbation, it is easier to take small gradient steps in desired direction
- Decrease score (increase loss) of *correct* class y^* :

$$x \leftarrow x - \eta \frac{\partial f(x, y^*)}{\partial x} \quad \text{or} \quad x \leftarrow x + \eta \frac{\partial L(x, y^*)}{\partial x}$$

- Increase score (decrease loss) of *incorrect* target class \hat{y} :

$$x \leftarrow x + \eta \frac{\partial f(x, \hat{y})}{\partial x} \quad \text{or} \quad x \leftarrow x - \eta \frac{\partial L(x, \hat{y})}{\partial x}$$

Fooling a linear classifier

- Increase score of target class \hat{y} :

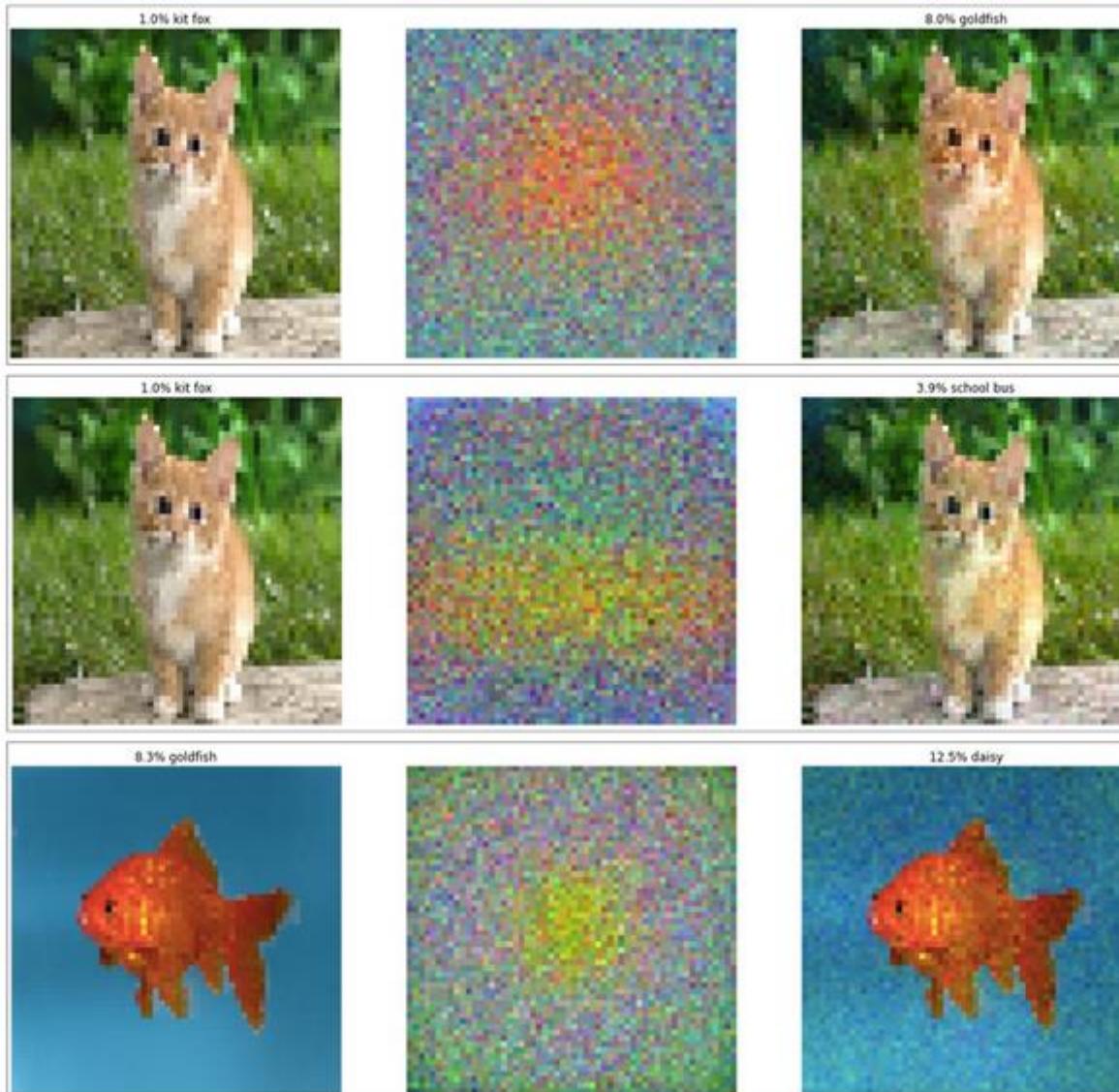
$$x \leftarrow x + \eta \frac{\partial f(x, \hat{y})}{\partial x}$$

- For a linear classifier with $f(x, \hat{y}) = w^T x$:

$$x \leftarrow x + \eta w$$

- To fool a linear classifier, add a small multiple of the target class weights to the test example

Fooling a linear classifier



Analysis of the linear case

- Response of classifier with weights w to adversarial example $x + r$:

$$w^T(x + r) = w^Tx + w^Tr$$

- Suppose the pixel values have precision *Epsilon* (ϵ), i.e., the classifier is normally expected to predict the same class for x and $x + r$ as long as $\|r\|_\infty \leq \epsilon$
- How to choose r to maximize the increase in activation w^Tr subject to $\|r\|_\infty \leq \epsilon$?

$$r = \epsilon \operatorname{sgn}(w)$$

Analysis of the linear case

- Response of classifier with weights w to adversarial example $x + r$, $r = \epsilon \operatorname{sgn}(w)$:

$$w^T(x + r) = w^T x + \epsilon w^T \operatorname{sgn}(w)$$

- If w has dimensionality d and average element magnitude m , how much will the activation increase?
 - By ϵdm , i.e., linearly as a function of d
 - The higher the dimensionality, the easier it is to make many small changes to the input that cause a large change in the output

Toy example

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1

$$w^T x = -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3$$

$$\sigma(w^T x) = \frac{1}{1 + e^{-(-3)}} = 0.047$$

Toy example

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
$x + r$	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

$$w^T x = -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3$$

$$\sigma(w^T x) = \frac{1}{1 + e^{-(-3)}} = 0.047$$

$$w^T(x + r) = -3 + 10 * 0.5 = 2$$

$$\sigma(w^T(x + r)) = \frac{1}{1 + e^{-2}} = 0.88$$

Generating adversarial examples

- **Fast gradient sign method:** Find the gradient of the loss w.r.t. correct class y^* , take element-wise sign, update in resulting direction:

$$x \leftarrow x + \epsilon \operatorname{sgn} \left(\frac{\partial L(x, y^*)}{\partial x} \right)$$



x
“panda”
57.7% confidence

+ .007 ×



$\operatorname{sign}(\nabla_x J(\theta, x, y))$
“nematode”
8.2% confidence

=



$x + \epsilon \operatorname{sign}(\nabla_x J(\theta, x, y))$
“gibbon”
99.3 % confidence

Generating adversarial examples

- **Fast gradient sign method:**

$$x \leftarrow x + \epsilon \operatorname{sgn} \left(\frac{\partial L(x, y^*)}{\partial x} \right)$$

- **Iterative gradient sign method:** take multiple smaller steps until misclassified, each time clip result to be within ϵ -neighborhood of original image
- **Least likely class method:** try to misclassify image as class \hat{y} with *smallest* initial score:

$$x \leftarrow x - \epsilon \operatorname{sgn} \left(\frac{\partial L(x, \hat{y})}{\partial x} \right)$$

Generating adversarial examples

Comparison of
methods for $\epsilon = 32$



Generating adversarial examples

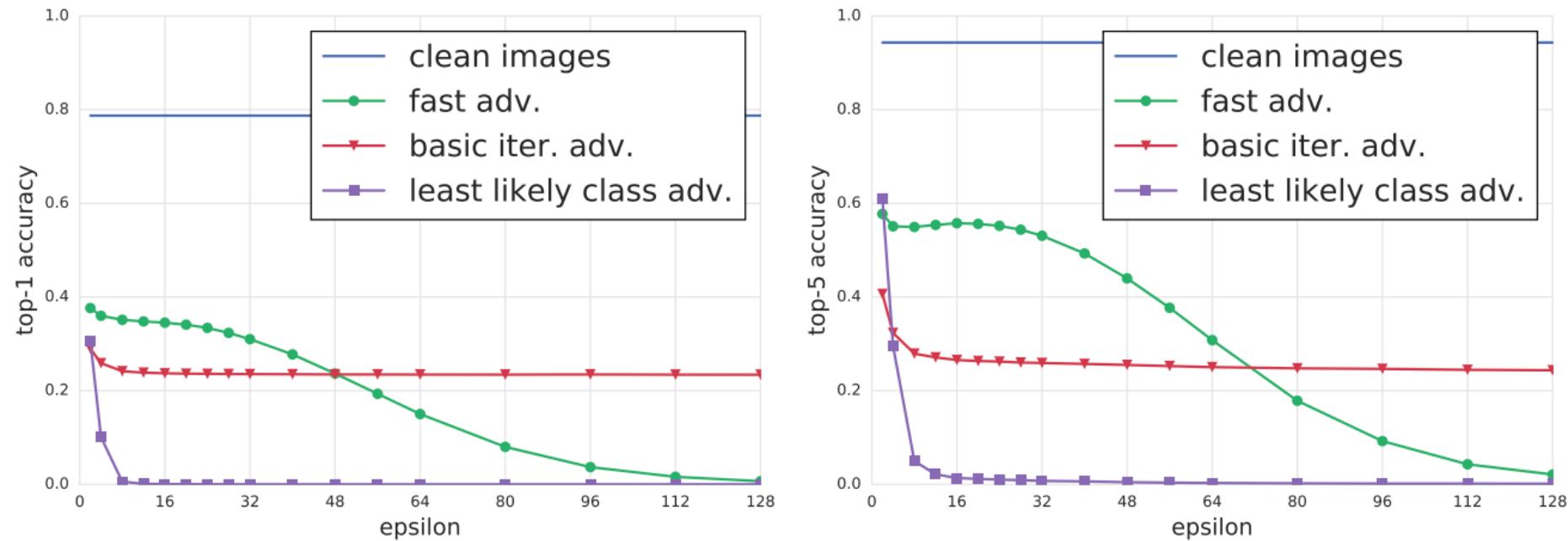


Figure 2: Top-1 and top-5 accuracy of Inception v3 under attack by different adversarial methods and different ϵ compared to “clean images” — unmodified images from the dataset. The accuracy was computed on all 50,000 validation images from the ImageNet dataset. In these experiments ϵ varies from 2 to 128.

Printed adversarial examples

- “Black box” attack on a cell phone app: take a clean image, add perturbation, print out, classify with TensorFlow Camera Demo app



(a) Image from dataset

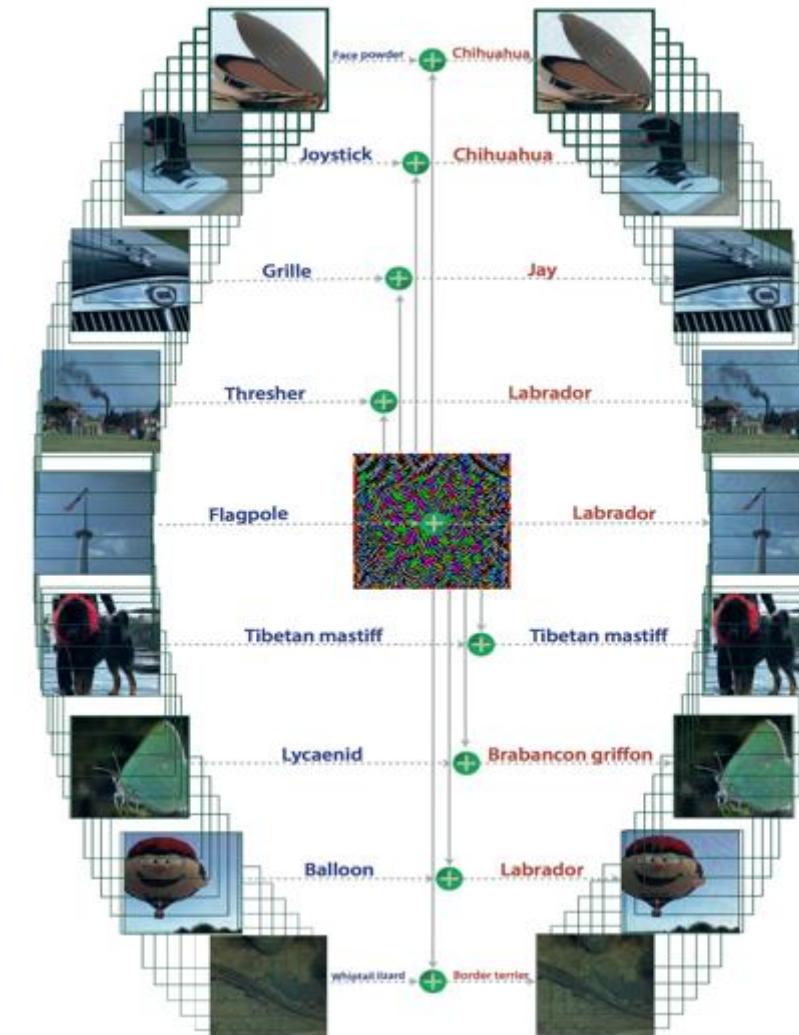
(b) Clean image

(c) Adv. image, $\epsilon = 4$

(d) Adv. image, $\epsilon = 8$

Universal adversarial perturbations

- Goal: for a given network, find an *image-independent* perturbation vector that causes *all images* to be misclassified with high probability



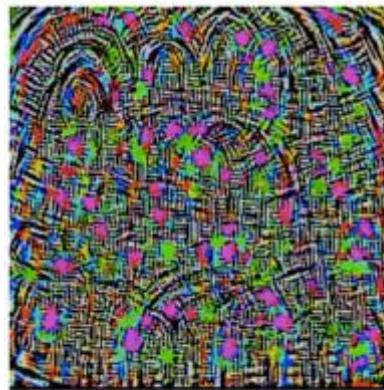
Universal adversarial perturbations

Approach:

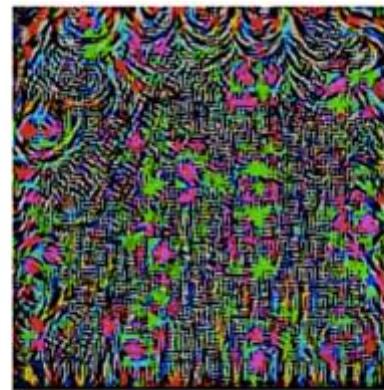
- Start with $r = 0$
- Cycle through training examples x_i (in multiple passes)
 - If $x_i + r$ is misclassified, skip to x_{i+1}
 - Find minimum perturbation Δr that takes $x_i + r + \Delta r$ to another class
 - Update $r \leftarrow r + \Delta r$, enforce $\|r\| \leq \epsilon$
- Terminate when fooling rate on training examples reaches target value

Universal adversarial perturbations

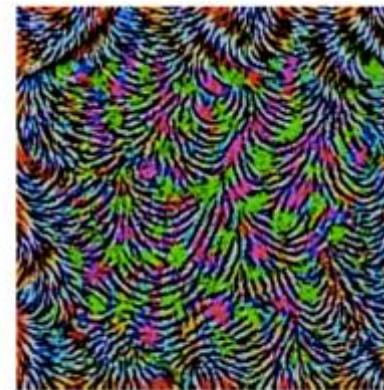
- Perturbation vectors computed from different architectures:



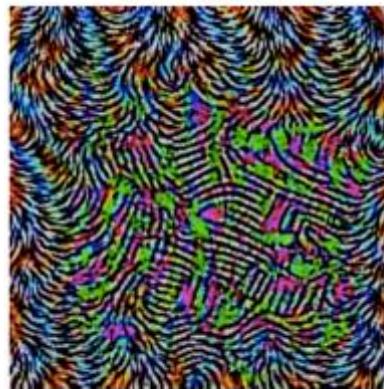
(a) CaffeNet



(b) VGG-F



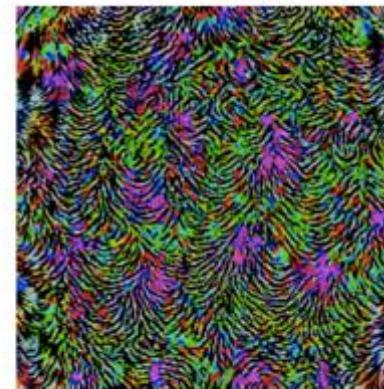
(c) VGG-16



(d) VGG-19

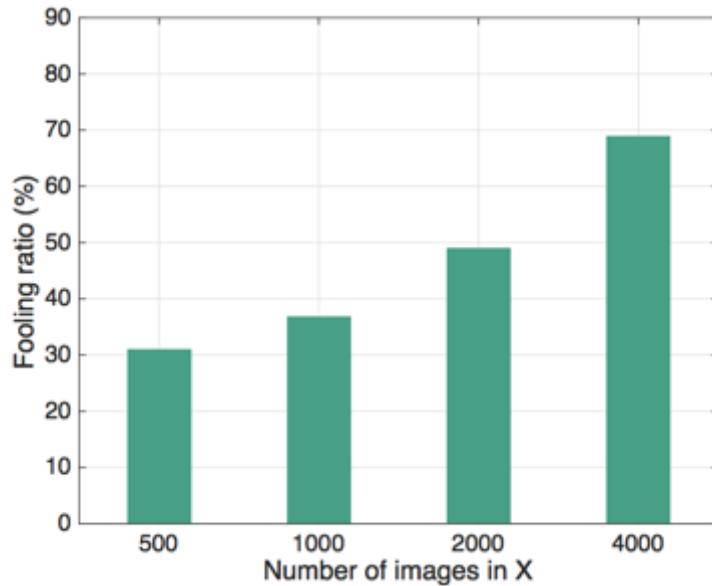


(e) GoogLeNet



(f) ResNet-152

Universal adversarial perturbations



Fooling ratio on validation set vs.
training set (X) size for GoogLeNet

Fooling rates on different models after training on 10,000 images

		CaffeNet [8]	VGG-F [2]	VGG-16 [17]	VGG-19 [17]	GoogLeNet [18]	ResNet-152 [6]
ℓ_2	X	85.4%	85.9%	90.7%	86.9%	82.9%	89.7%
	Val.	85.6	87.0%	90.3%	84.5%	82.0%	88.5%
ℓ_∞	X	93.1%	93.8%	78.5%	77.8%	80.8%	85.4%
	Val.	93.3%	93.7%	78.3%	77.8%	78.9%	84.0%

Universal adversarial perturbations

- Universal perturbations turn out to generalize well across models!

Fooling rate when computing a perturbation for one model (rows) and testing it on others (columns)

	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	93.7%	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	93.3%	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	78.9%	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	78.3%	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	77.8%	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	84.0%

Properties of adversarial examples

- For any input image, it is usually easy to generate a very similar image that gets misclassified by the same network
- To obtain an adversarial example, one does not need to do precise gradient ascent
- Adversarial images can (somewhat) survive transformations like being printed and photographed
- It is possible to attack many images with the same perturbation
- Adversarial examples that can fool one network have a high chance of fooling a network with different parameters and even architecture

Why are deep networks easy to fool?

- Networks are “too linear”: it is easy to manipulate output in a predictable way given the input
- The input dimensionality is high, so one can get a large change in the output by changing individual inputs by small amounts
- Neural networks can fit anything, but nothing prevents them from behaving erratically between training samples
 - Counter-intuitively, a network can both generalize well on natural images and be susceptible to adversarial examples
 - Adversarial examples generalize well because different models learn similar functions when trained to perform the same task (or because adversarial examples are a function of the data rather than of the network)?

Adversarial examples: Outline

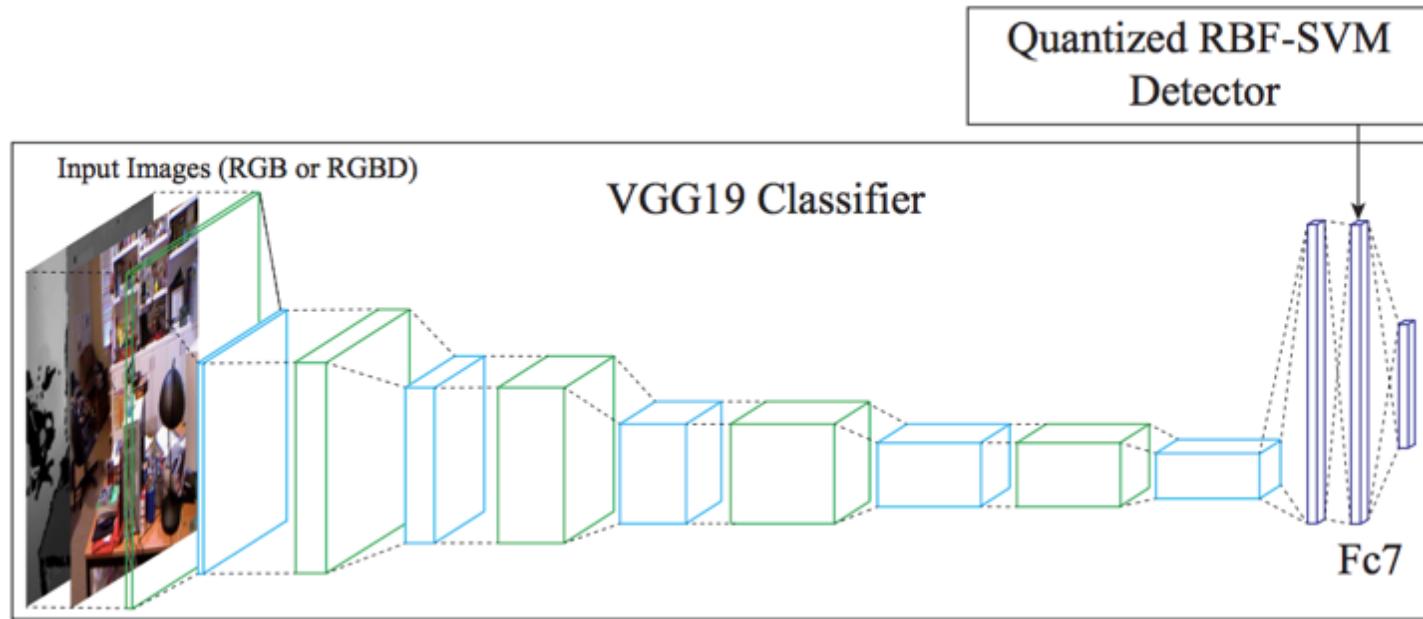
- Generating adversarial examples
 - Finding smallest “fooling” transformation
 - Gradient ascent
 - Fast gradient sign, iterative variants
 - Universal adversarial perturbations
- Why are neural networks easy to fool?
- Defending against adversarial examples
 - Adversarial training
 - Learning to reject adversarial examples
 - Robust architectures
 - Image pre-processing

Defending against adversarial examples

- Adversarial training: networks can be made somewhat resistant by augmenting or regularizing training with adversarial examples ([Goodfellow et al.](#) 2015, [Tramer et al.](#) 2018)

Defending against adversarial examples

- Train a separate model to reject adversarial examples:
SafetyNet
 - uses **radial basis function kernel (RBF) - SVM**



Defending against adversarial examples

- Robust architectures

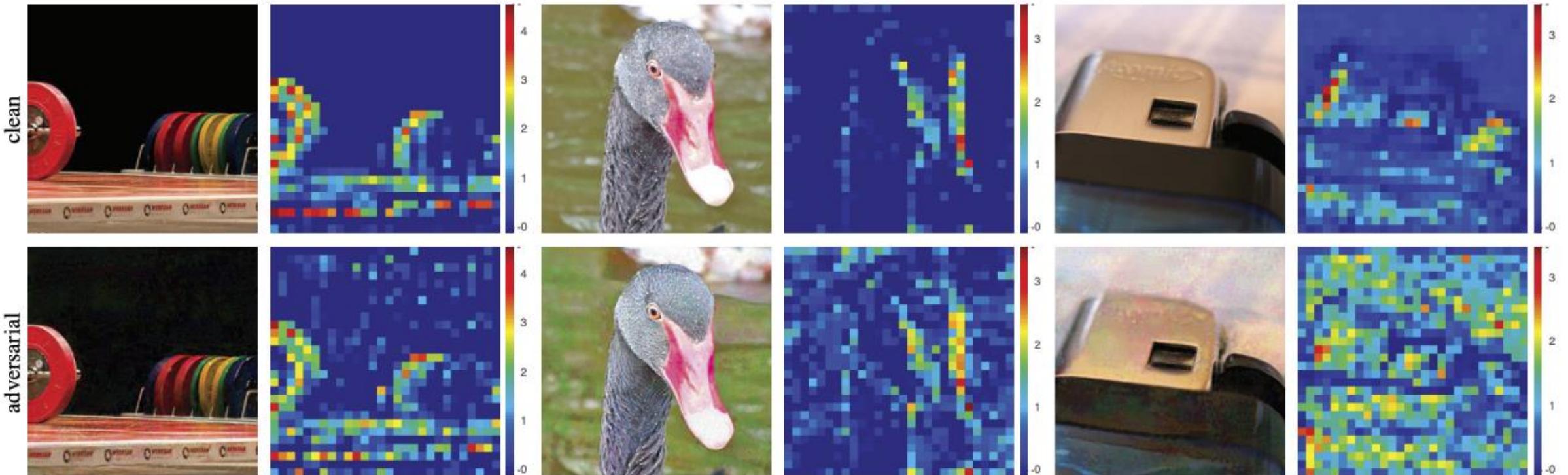
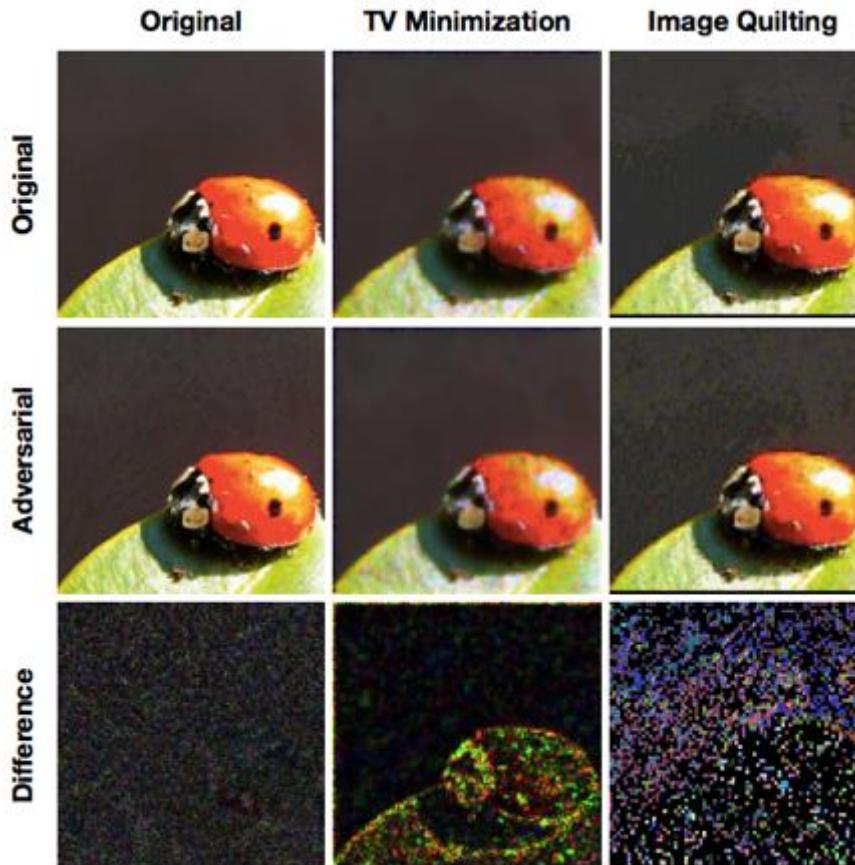


Figure 2. More examples similar to Figure 1. We show feature maps corresponding to clean images (top) and to their adversarial perturbed versions (bottom). The feature maps for each pair of examples are from the same channel of a res_3 block in the same ResNet-50 trained on clean images. The attacker has a maximum perturbation $\epsilon = 16$ in the pixel domain.

Defending against adversarial examples

- Pre-process input images to disrupt adversarial perturbations



Adversarial examples: Outline

- Generating adversarial examples
 - Finding smallest “fooling” transformation
 - Gradient ascent
 - Fast gradient sign, iterative variants
 - Universal adversarial perturbations
- Why are neural networks easy to fool?
- Defending against adversarial examples
 - Adversarial training
 - Learning to reject adversarial examples
 - Robust architectures
 - Image pre-processing
- “Open” topics
 - Broadening the scope of adversarial examples
 - Adversarial examples and human perception

Adversarial examples for detection

- It is much harder to fool a detector like Faster R-CNN or YOLO than a classifier; larger perturbations are required



Adversarial examples for detection

- It is much harder to fool a detector like Faster R-CNN or YOLO than a classifier; larger perturbations are required
- It is even harder to fool a detector with physical objects



"All three patterns reliably fool detectors when mapped into videos. However, physical instances of these patterns are not equally successful. The first two stop signs, as physical objects, only occasionally fool Faster RCNN; the third one, which has a much more extreme pattern, is more effective."

Robust adversarial examples

Color Channel Perturbation



1st Row: Clean Image, 2nd Row: CCP-Fixed Attack, 3rd Row: CCP-Variable Attack

K. Jayendra, S.R. Dubey, S. Chakraborty, [Color Channel Perturbation Attacks for Fooling Convolutional Neural Networks and A Defense Against Such Attacks](#), IEEE TAI 2020

Robust adversarial examples

3D printed adversarial object ([YouTube video](#))



classified as turtle



classified as rifle



classified as other

<https://blog.openai.com/robust-adversarial-inputs/>

Adversarial examples and humans

- Adversarial examples that are designed to transfer across multiple architectures can also be shown to confuse the human visual system in rapid presentation settings

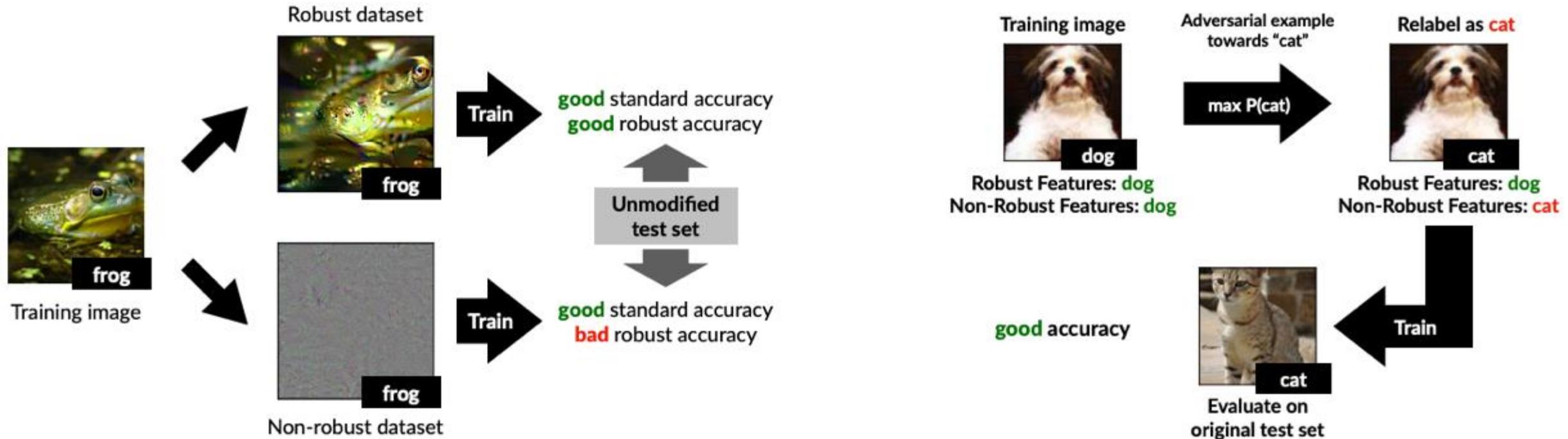
original



adv



Adversarial examples are not bugs, they are features



Disentangle features into robust and non-robust

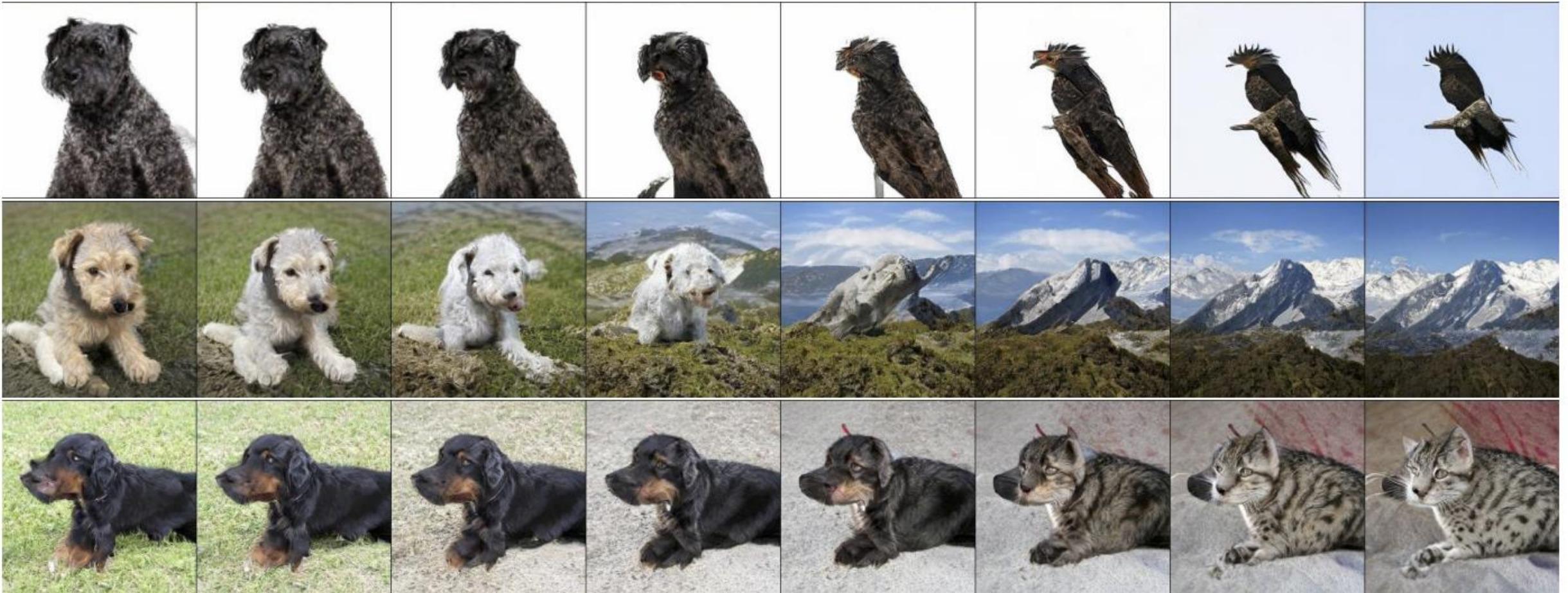
Construct a dataset which appears mislabeled to humans (via adversarial examples) but results in good accuracy on the original test set

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Generative adversarial networks



[BigGAN](#) (2018)

Outline

- Unsupervised Learning
- Generative tasks
- Original GAN formulations
 - NSGAN
 - DCGAN
- Other popular formulations
 - WGAN, WGAN-GP
 - LSGAN
- State-of-the-art architectures
 - Progressive GAN, StyleGAN
- Evaluating GANs
- Visualizing and controlling GANs

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map $x \rightarrow y$

Examples: Classification,
regression, object detection,
semantic segmentation,
image captioning, etc.

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map $x \rightarrow y$

Examples: Classification,
regression, object detection,
semantic segmentation,
image captioning, etc.



→ **Cat**

Classification

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map $x \rightarrow y$

Examples: Classification,
regression, object detection,
semantic segmentation,
image captioning, etc.



DOG, DOG, CAT

Object Detection

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map $x \rightarrow y$

Examples: Classification,
regression, object detection,
semantic segmentation,
image captioning, etc.



GRASS, CAT,
TREE, SKY

Semantic Segmentation

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map $x \rightarrow y$

Examples: Classification,
regression, object detection,
semantic segmentation,
image captioning, etc.



A cat sitting on a suitcase on the floor

Image Captioning

Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn some underlying
hidden structure of the data

Examples: Clustering,
dimensionality reduction, feature
learning, density estimation, etc.

Supervised vs Unsupervised Learning

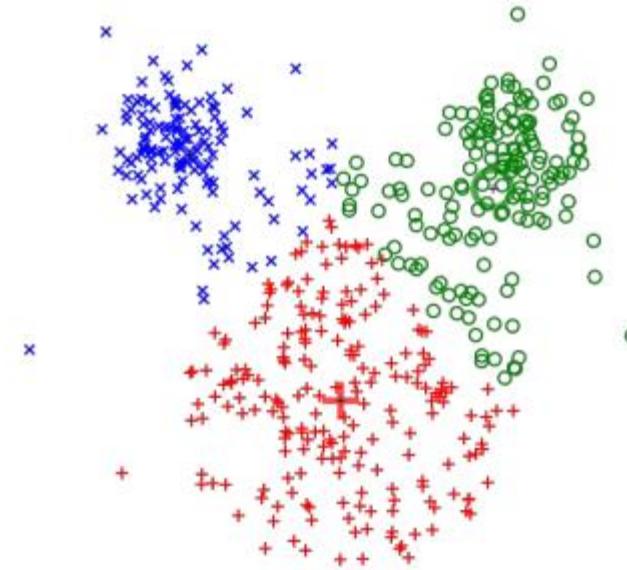
Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



K-Means Clustering

Supervised vs Unsupervised Learning

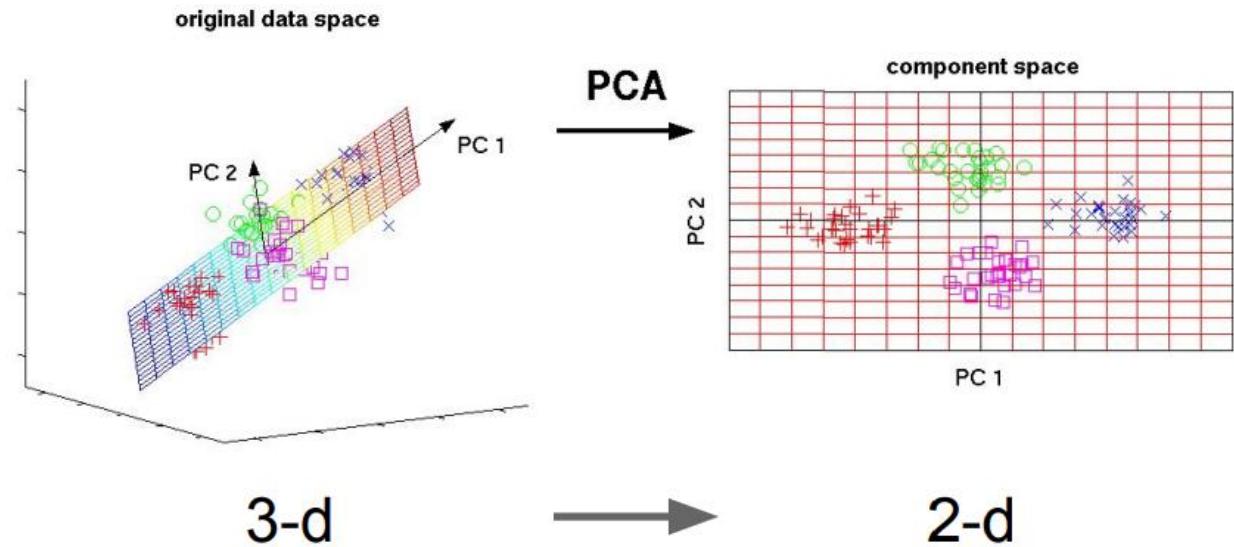
Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



(Principal Component Analysis)
Dimensionality Reduction

Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x

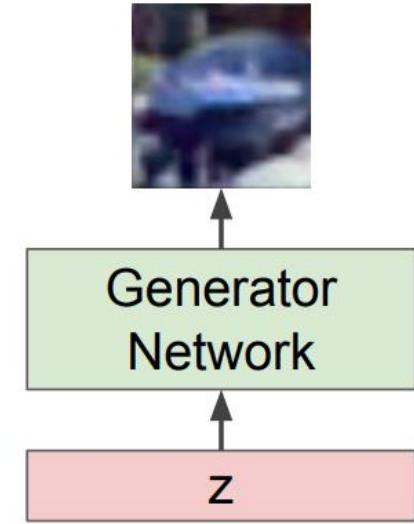
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Output: Sample from training distribution

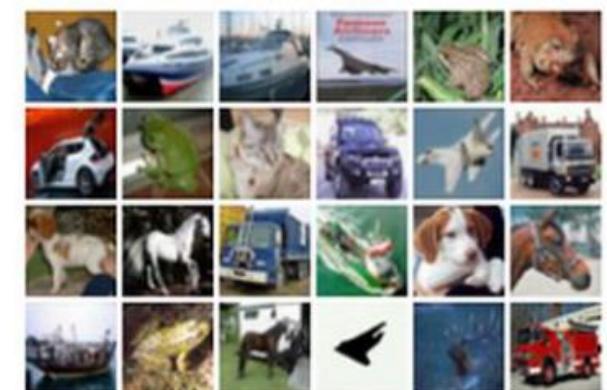
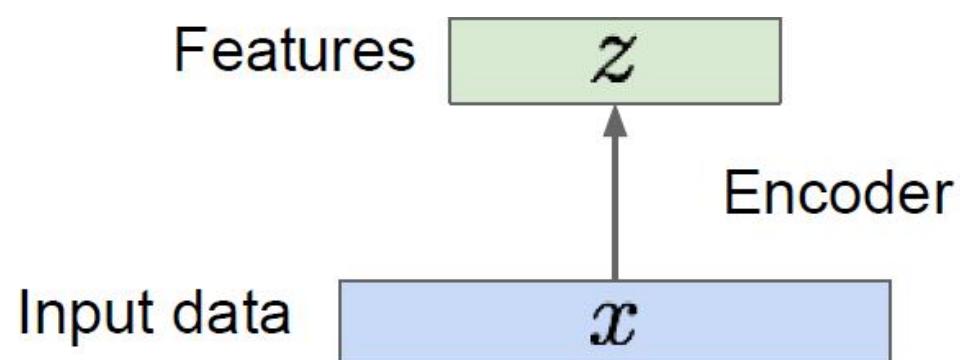
Input: Random noise



**Generative Adversarial Networks
(Distribution learning)**

Autoencoders

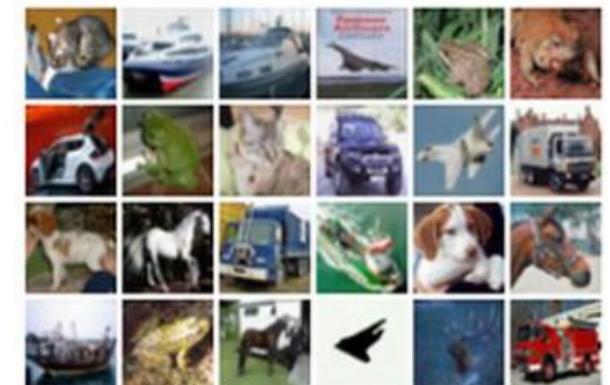
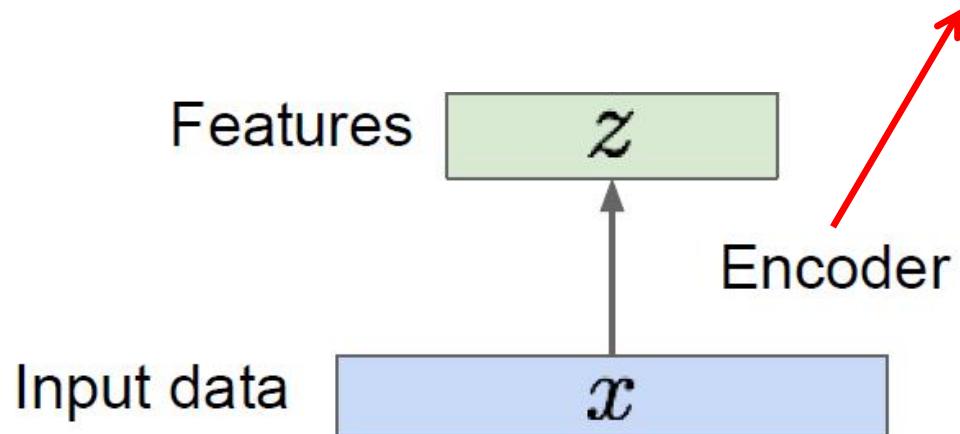
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data



Autoencoders

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

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

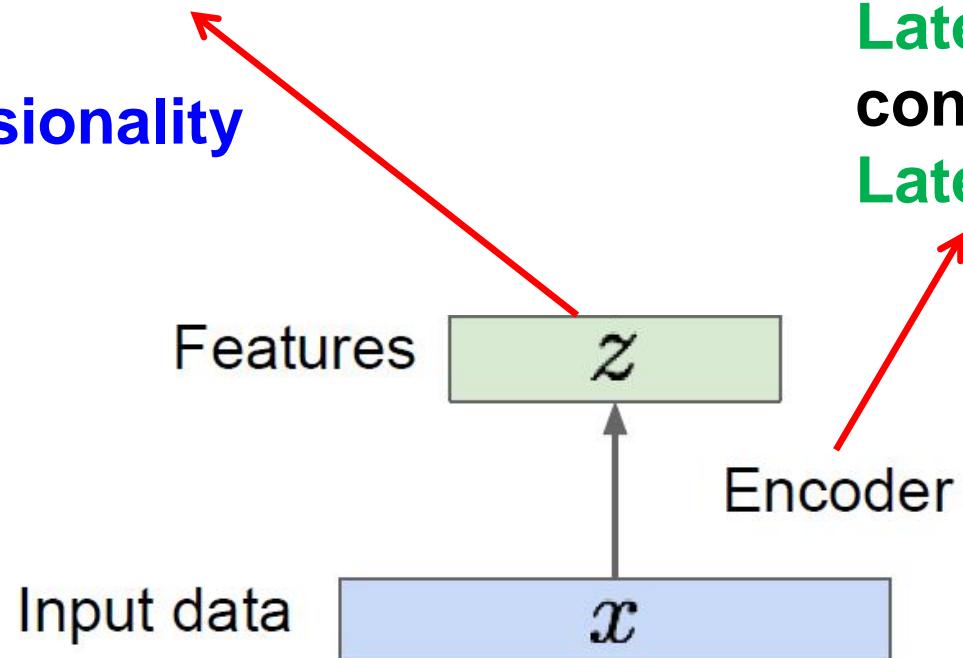


Autoencoders

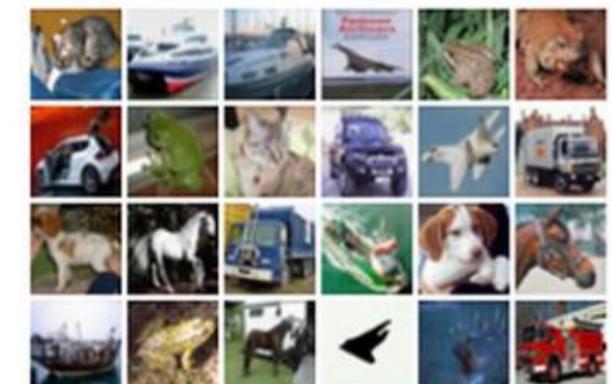
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

**Z usually smaller than X
(Dimensionality Reduction)**

Q: Why dimensionality reduction?



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



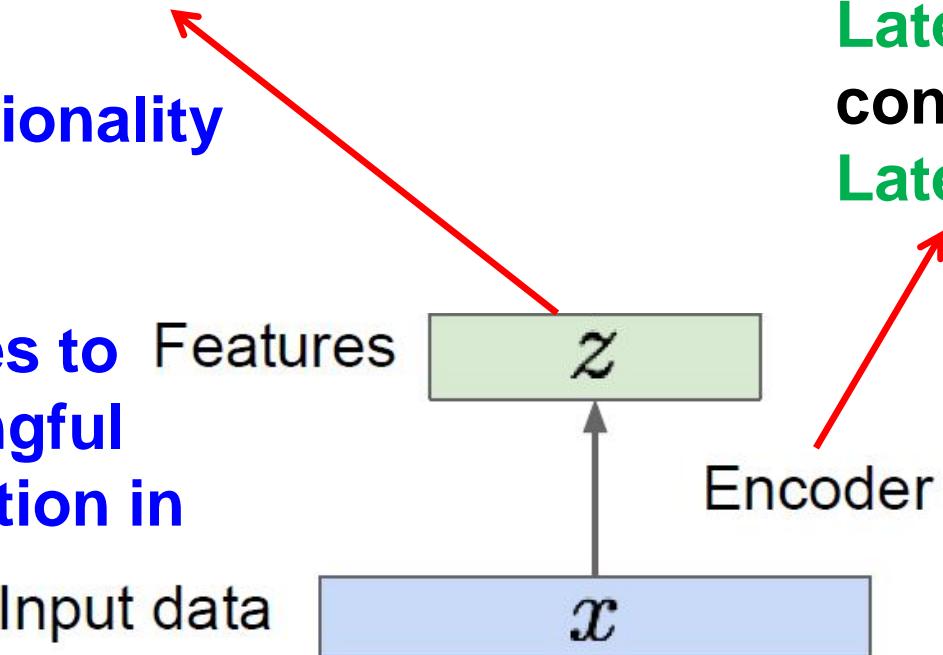
Autoencoders

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

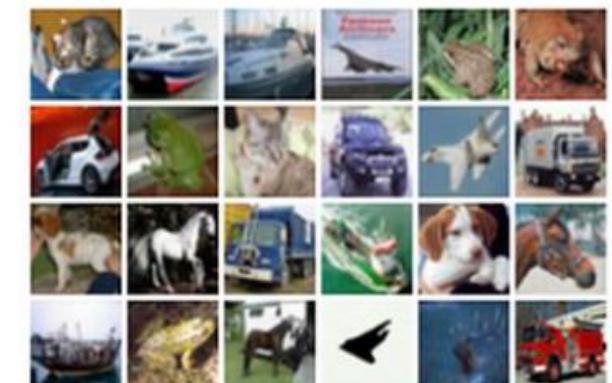
**Z usually smaller than X
(Dimensionality Reduction)**

Q: Why dimensionality reduction?

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

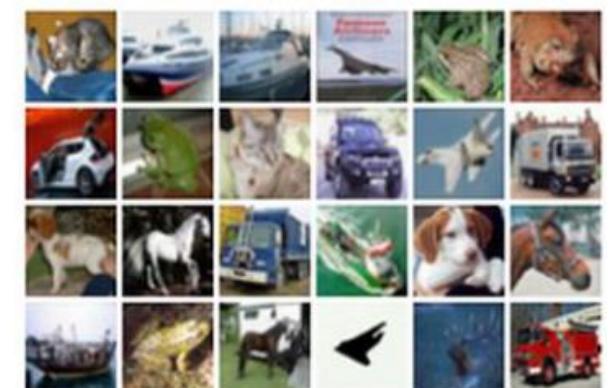
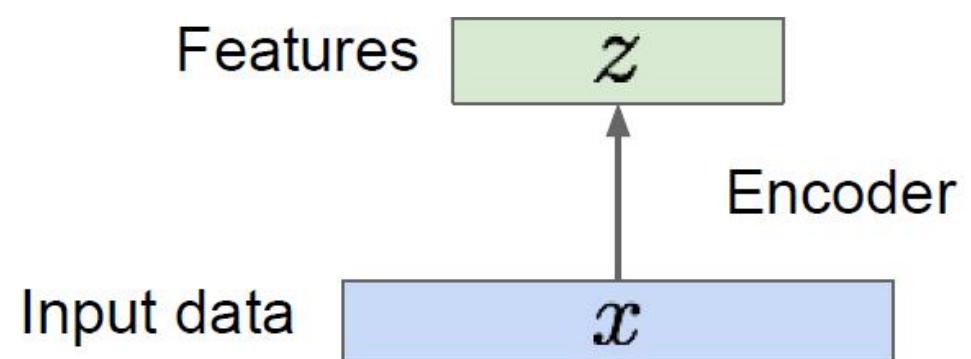


Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



Autoencoders

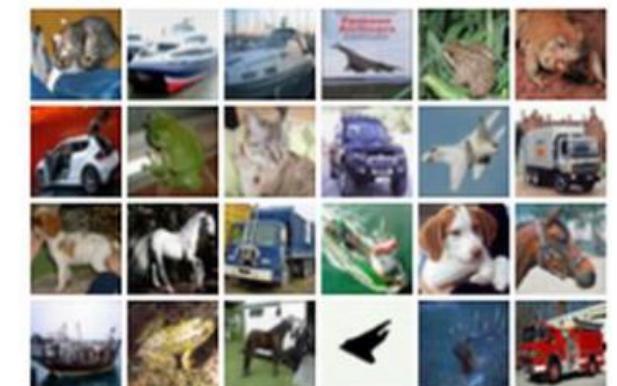
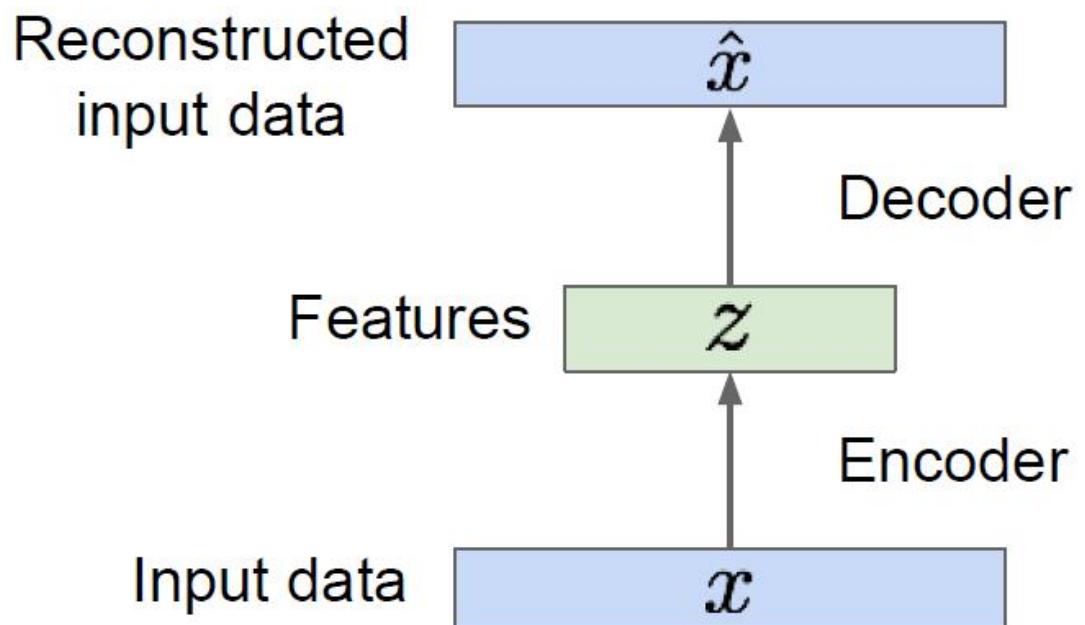
How to learn this feature representation?



Autoencoders

How to learn this feature representation?

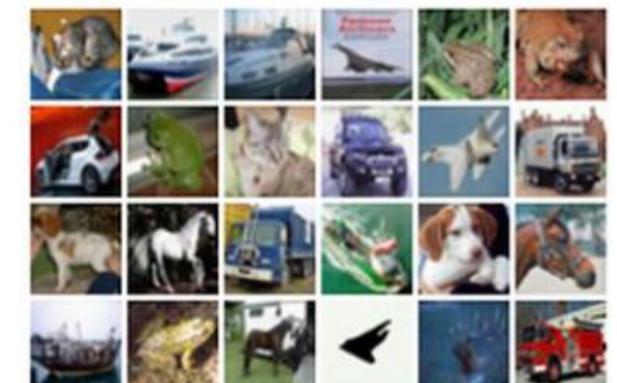
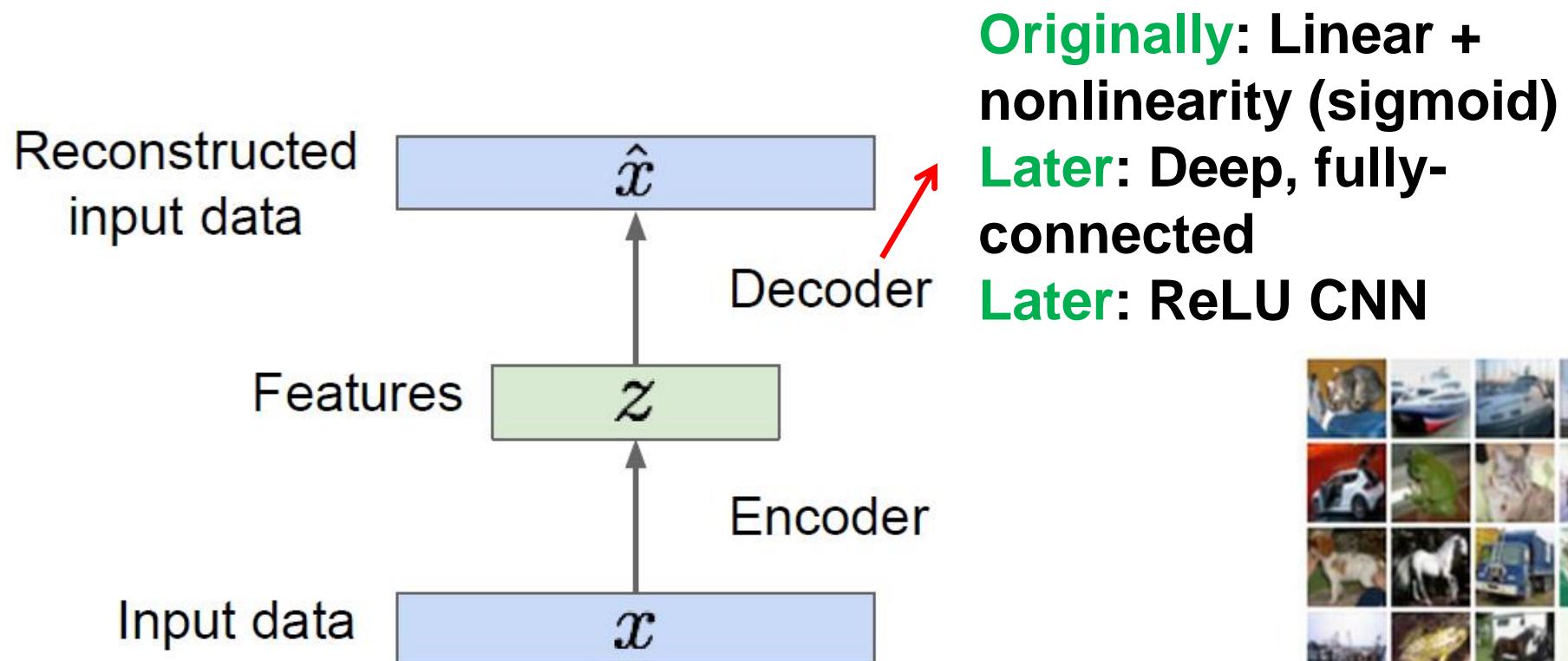
Train such that features can be used to reconstruct original data “Autoencoding” - encoding itself



Autoencoders

How to learn this feature representation?

Train such that features can be used to reconstruct original data “Autoencoding” - encoding itself



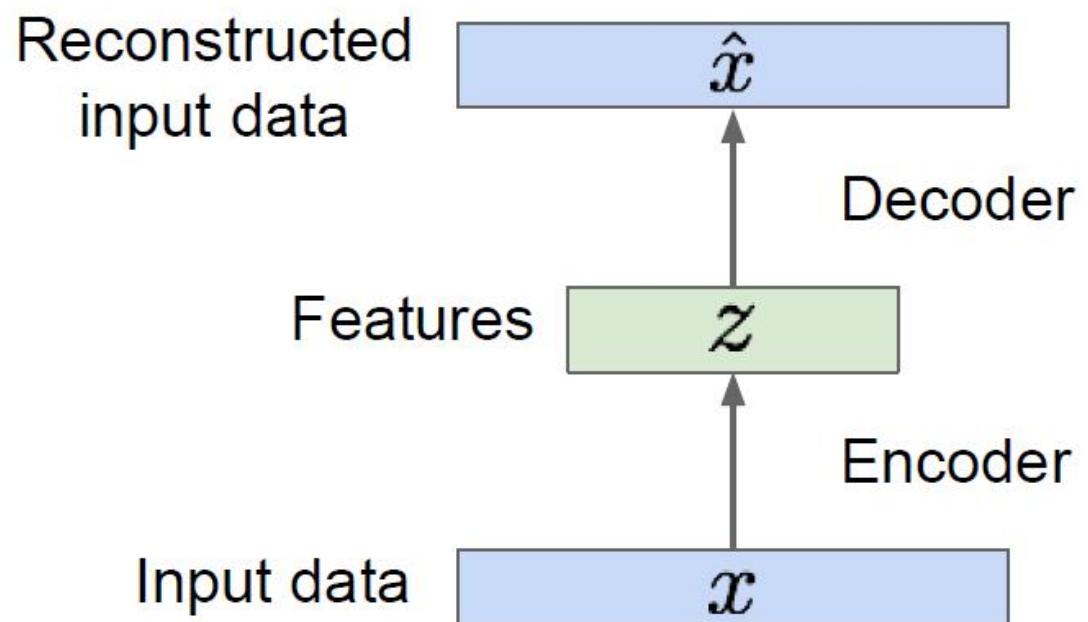
Autoencoders

Reconstructed Data

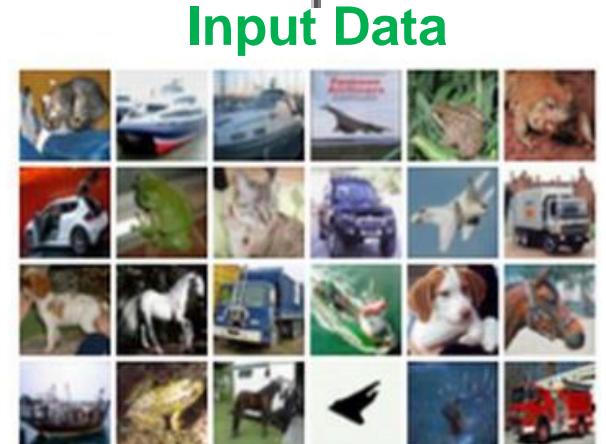


How to learn this feature representation?

Train such that features can be used to reconstruct original data “Autoencoding” - encoding itself



Decoder: 4-layer upconv
Encoder: 4-layer conv

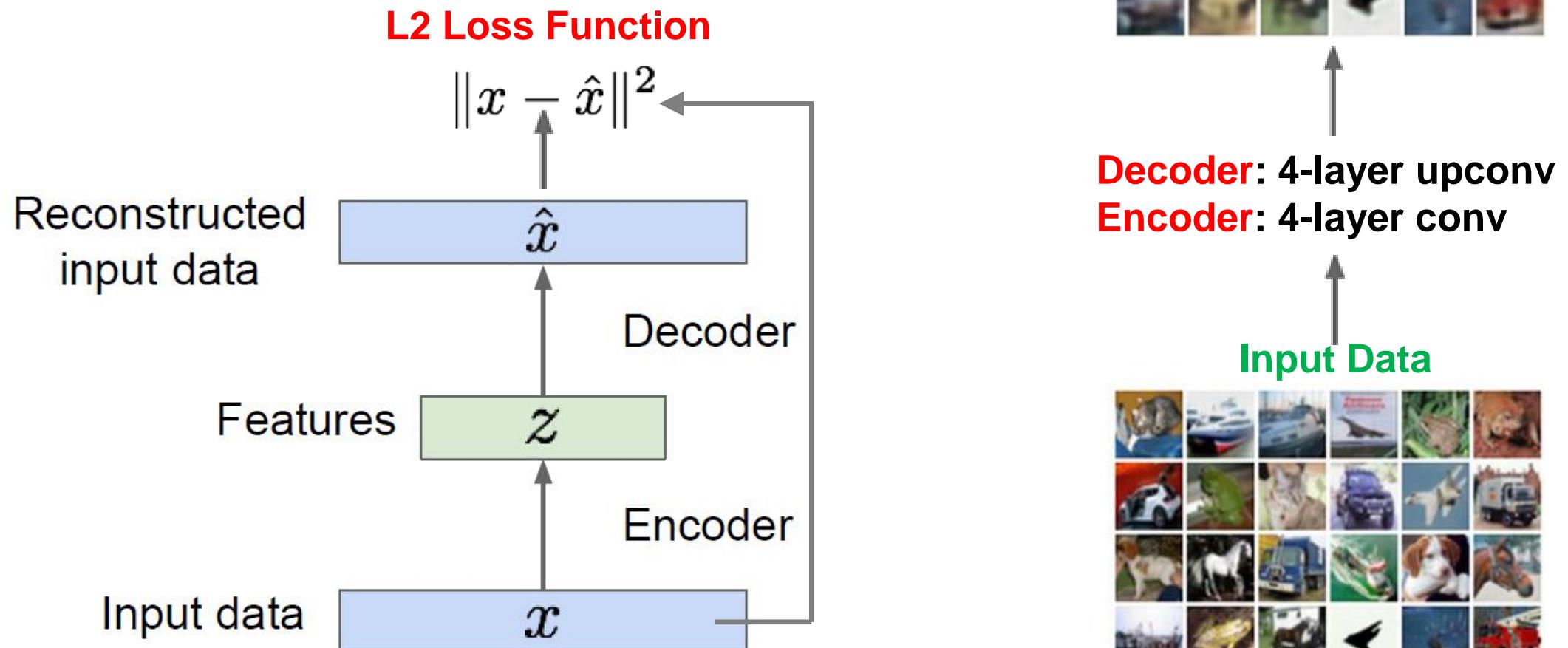


Autoencoders

Reconstructed Data



Train such that features can be used to reconstruct original data



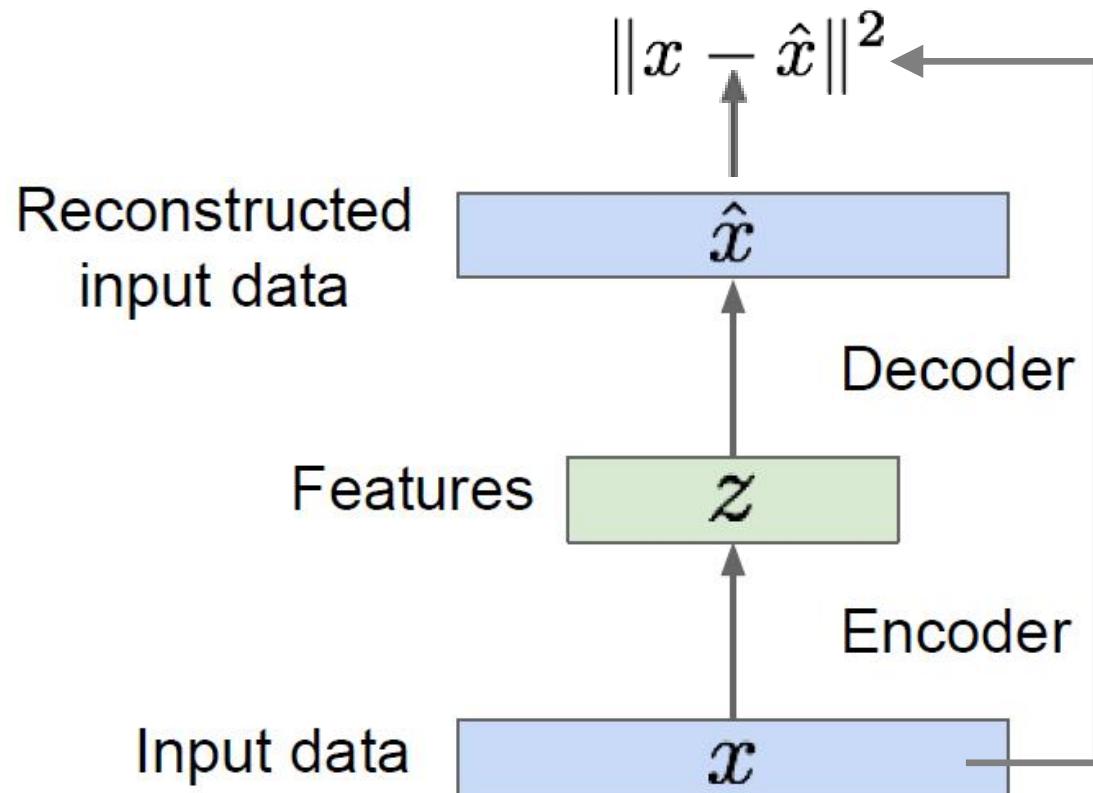
Autoencoders

Reconstructed Data



Doesn't use labels!

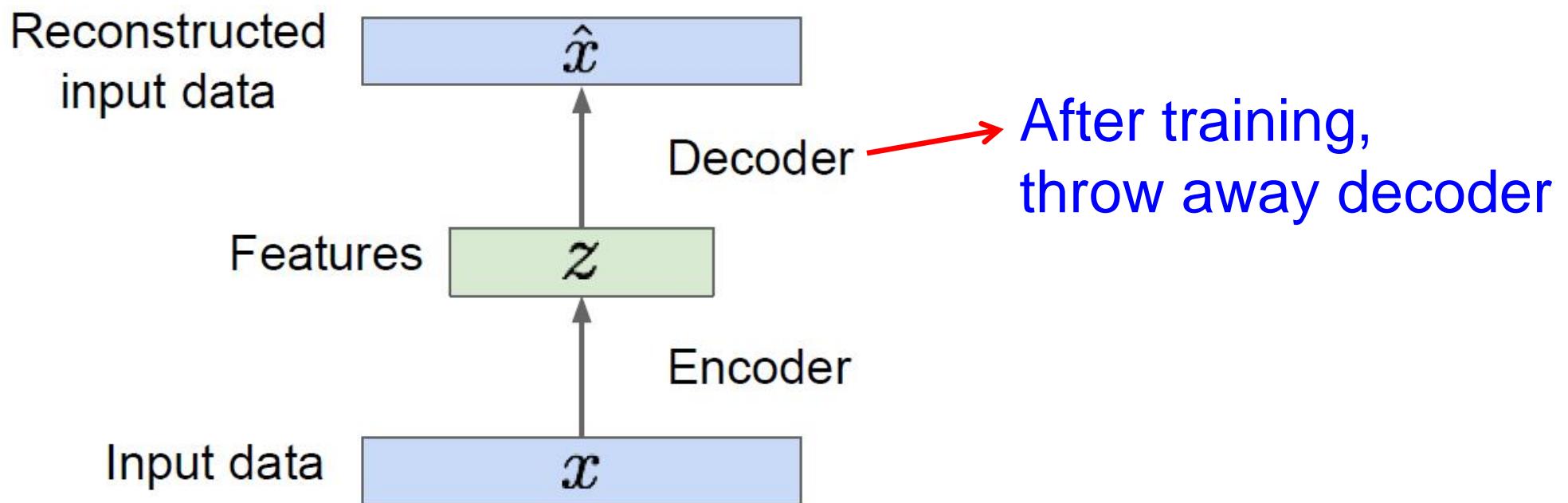
L2 Loss Function



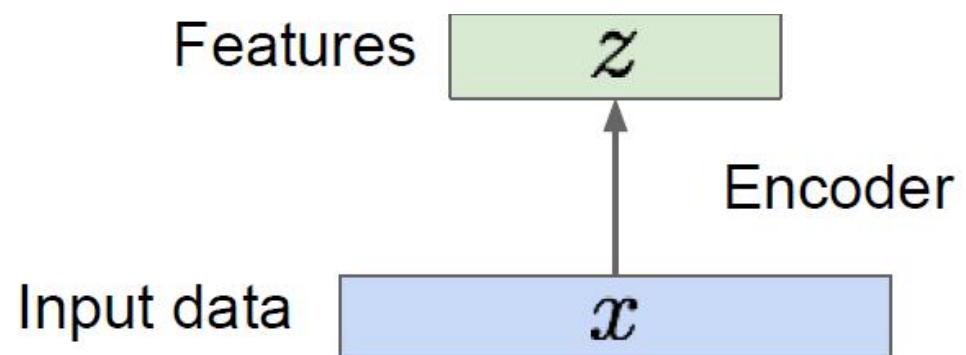
Decoder: 4-layer upconv
Encoder: 4-layer conv



Autoencoders

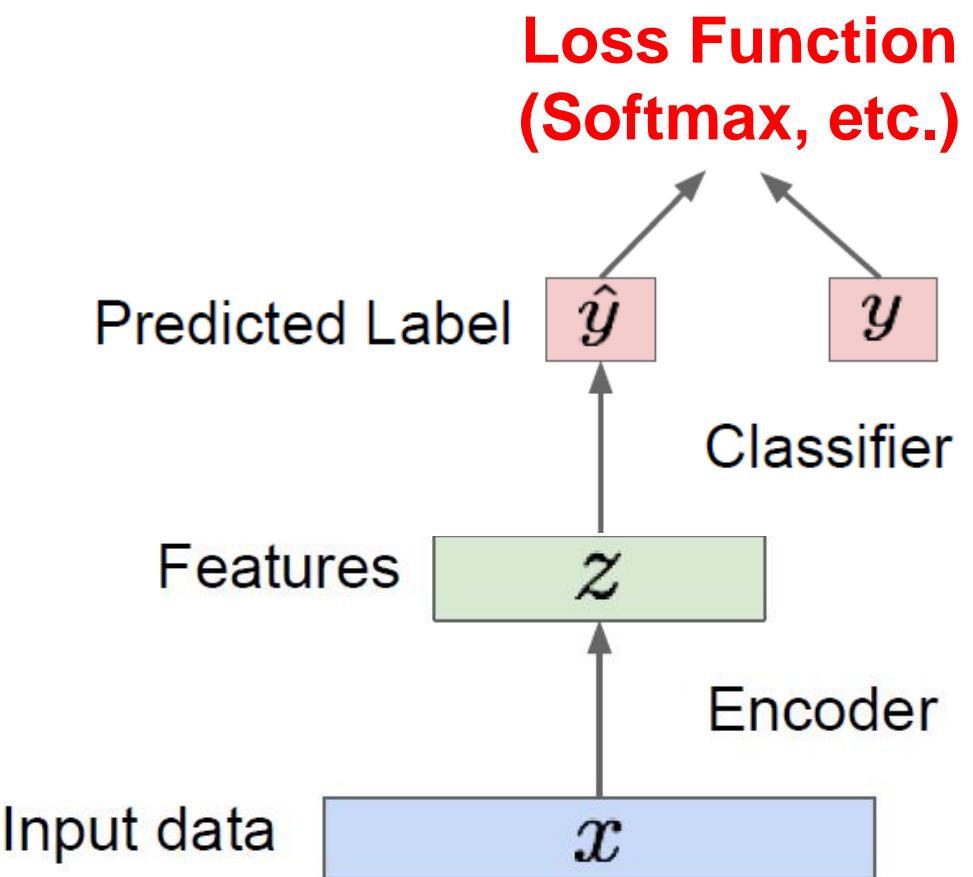


Autoencoders



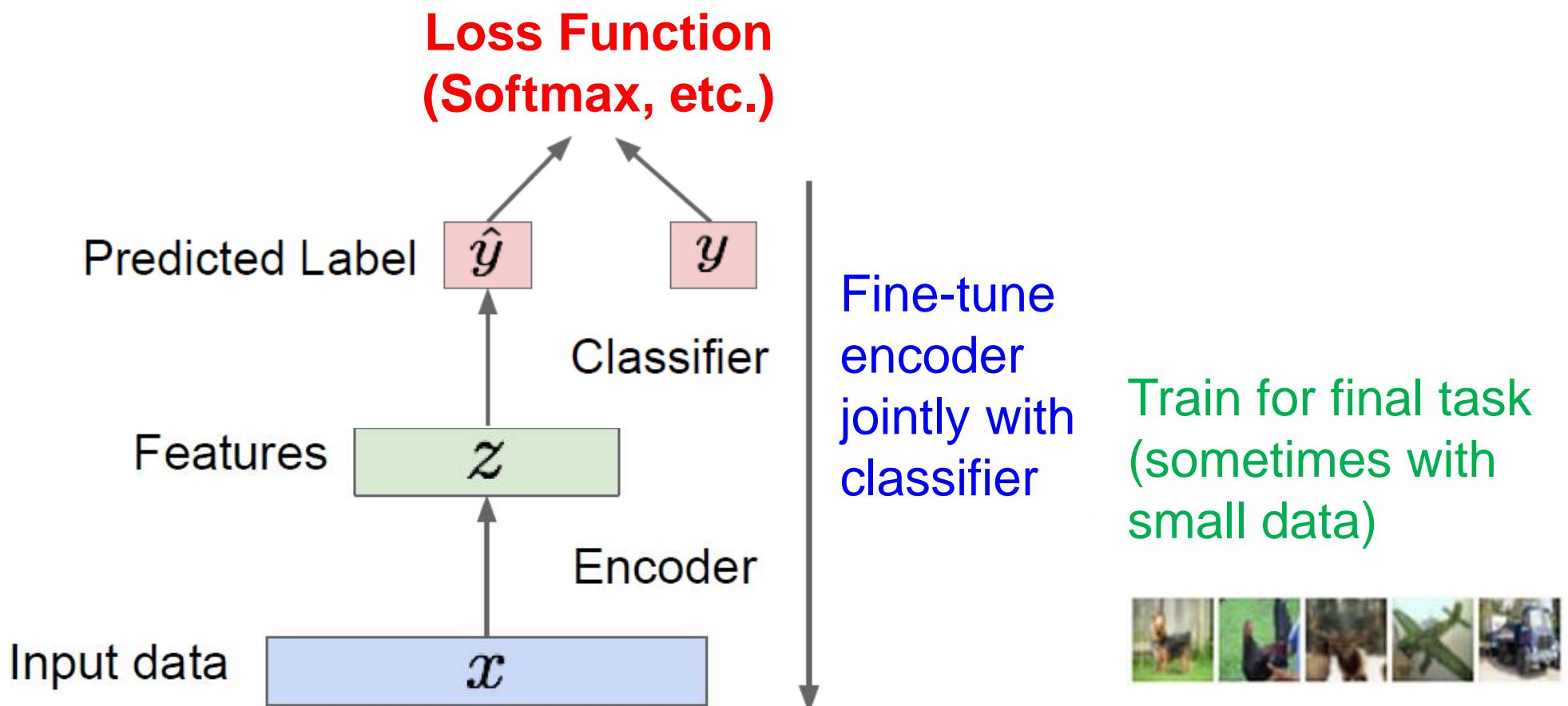
Autoencoders

Encoder can be used to initialize a supervised model



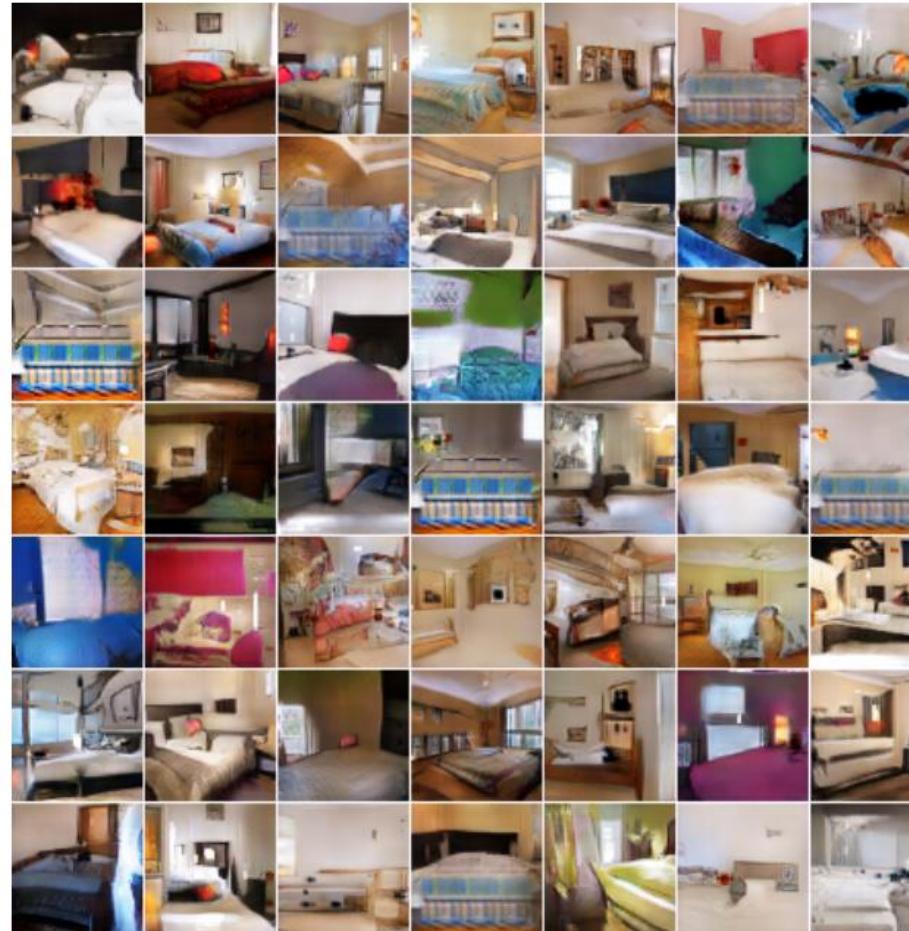
Autoencoders

Encoder can be used to initialize a supervised model



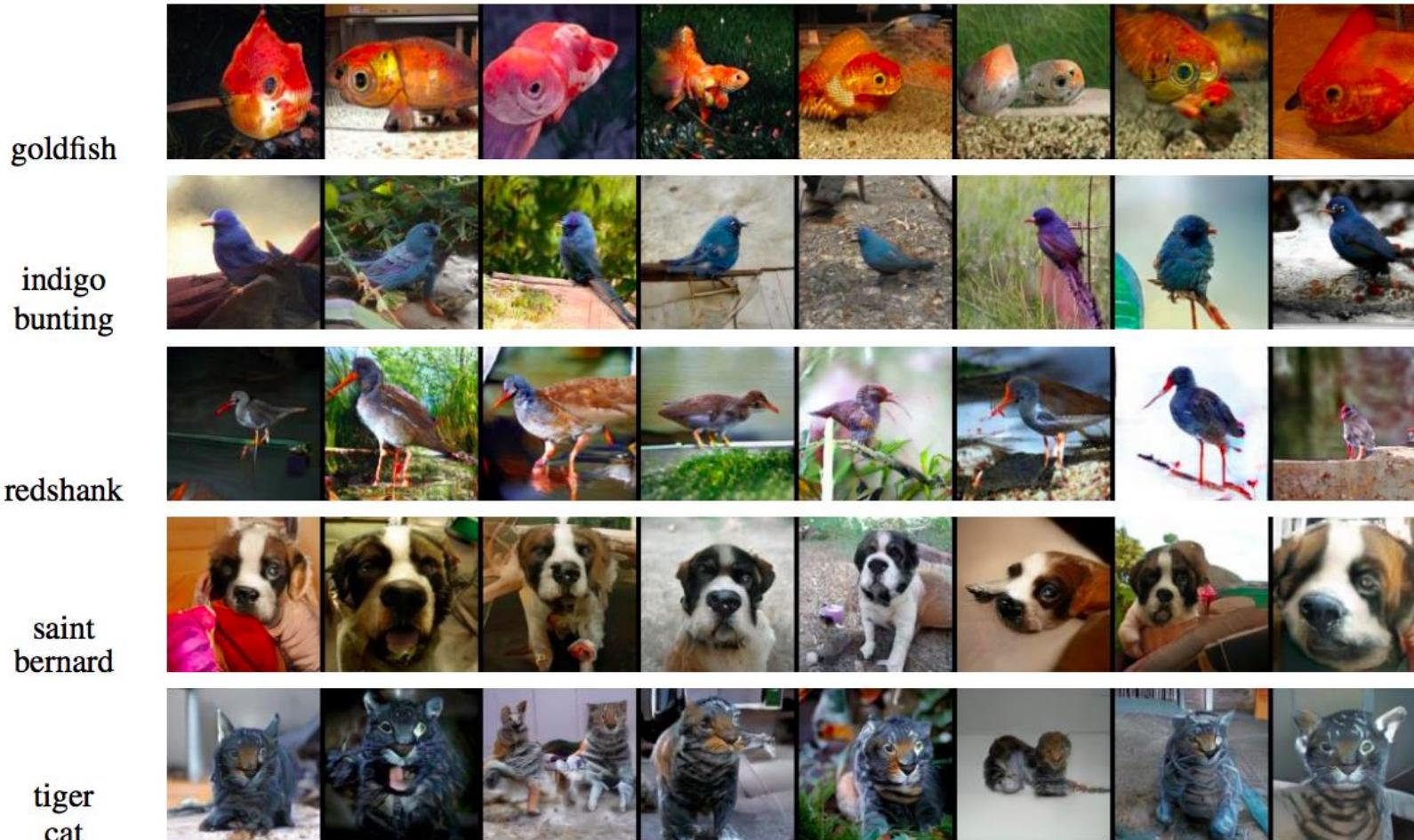
Generative tasks

- Generation (from scratch): learn to sample from the distribution represented by the training set
 - *Unsupervised learning* task



Generative tasks

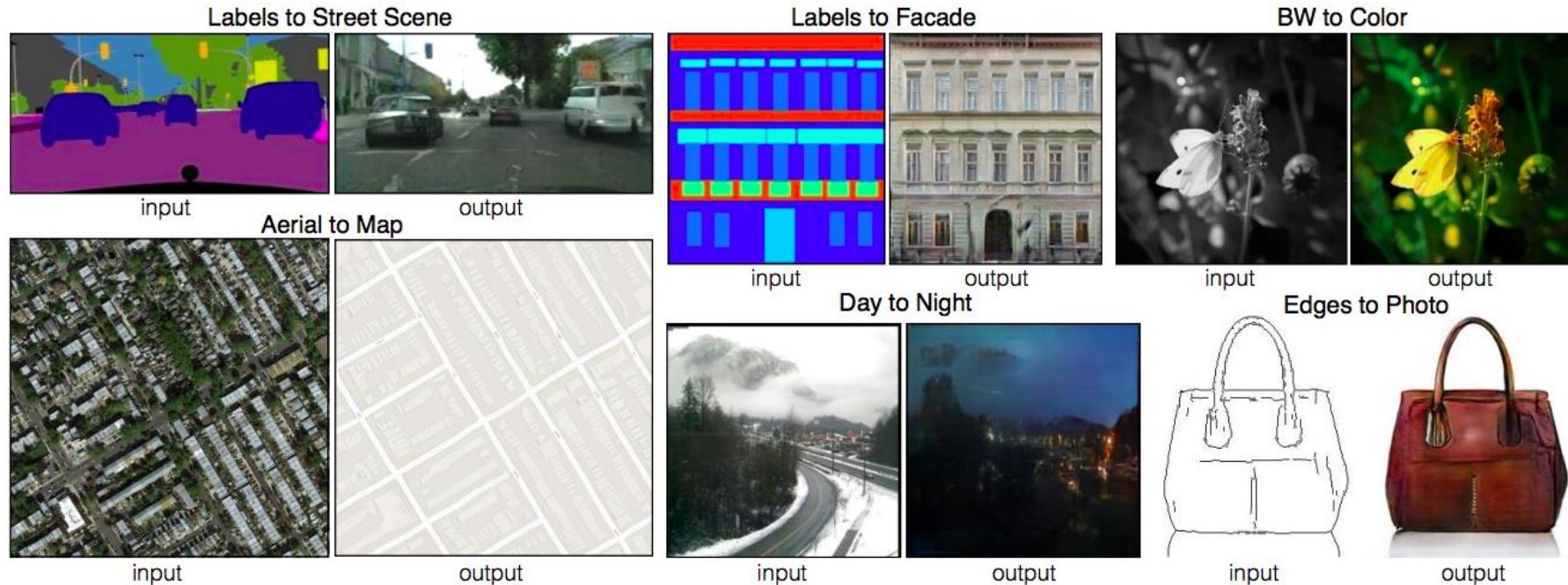
- Generation conditioned on class label



[Figure source](#)

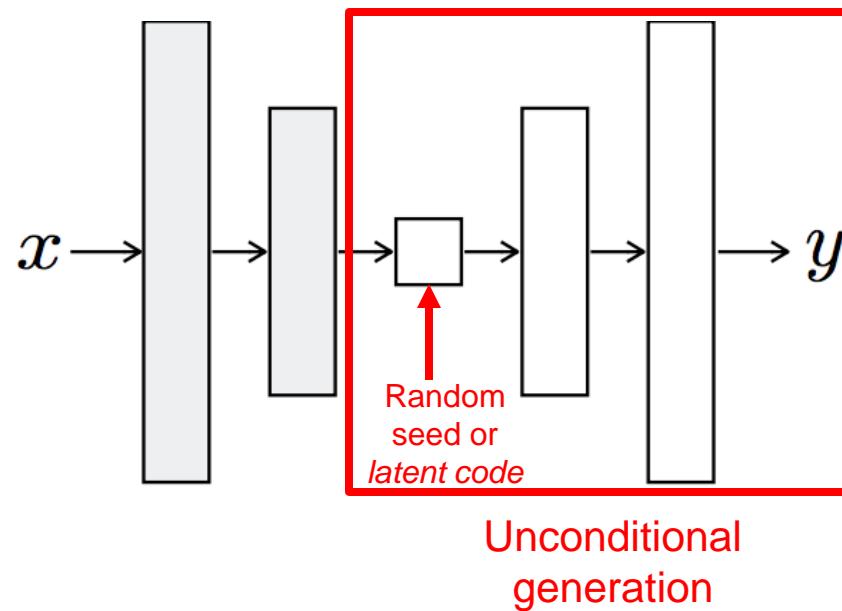
Generative tasks

- Generation conditioned on image (image-to-image translation)



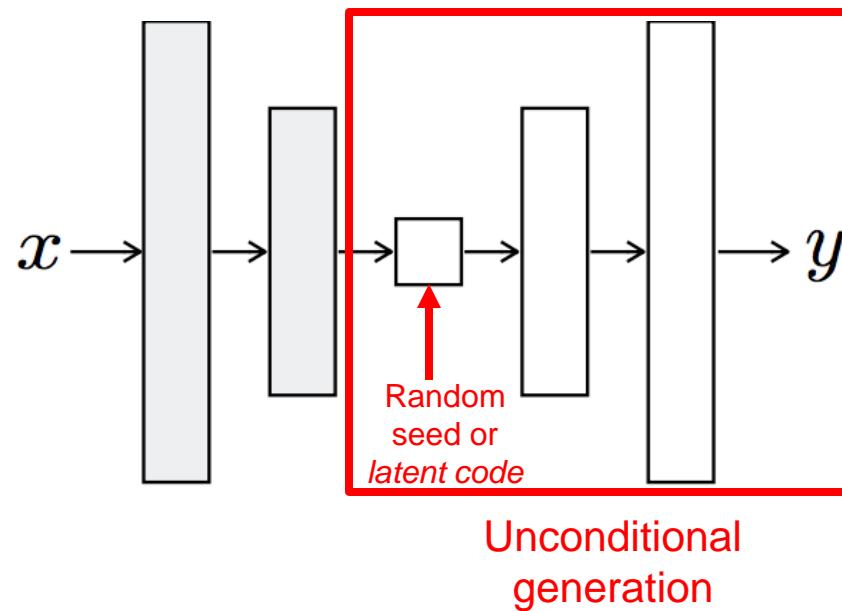
Designing a network for generative tasks

1. We need an architecture that can generate an image
 - Recall upsampling architectures for dense prediction



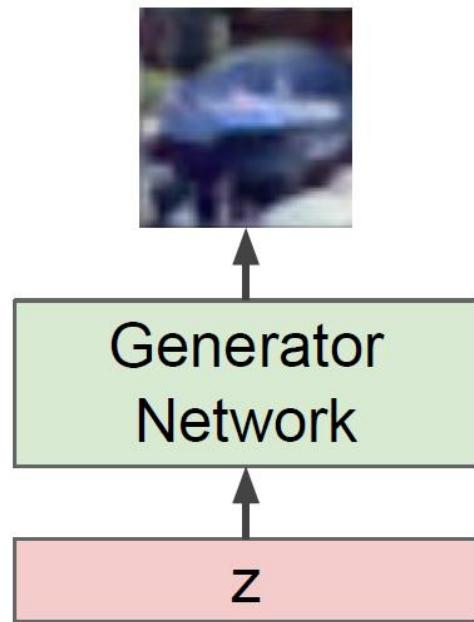
Designing a network for generative tasks

1. We need an architecture that can generate an image
 - Recall upsampling architectures for dense prediction
 - Sample from a simple distribution, e.g. random noise.
 - Learn transformation to training distribution.



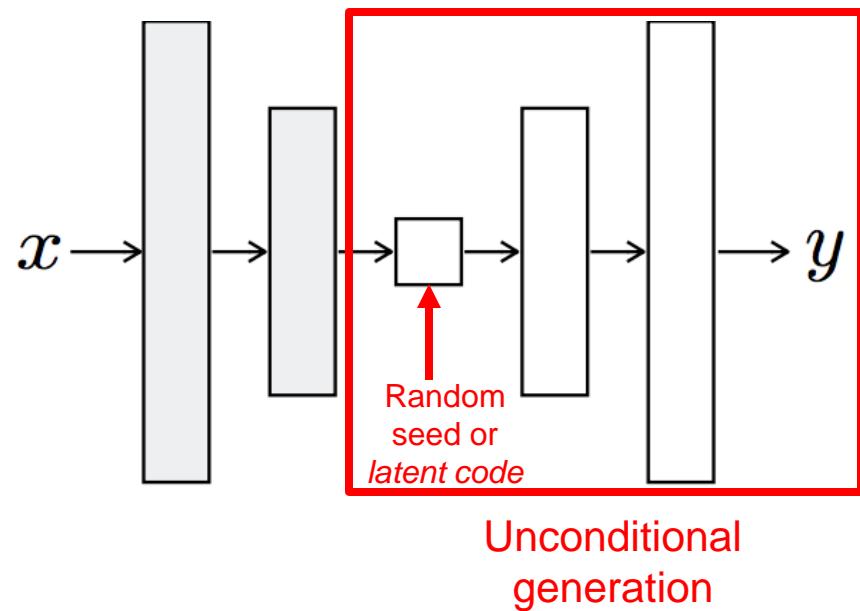
Output: Sample from
training distribution

Input: Random noise



Designing a network for generative tasks

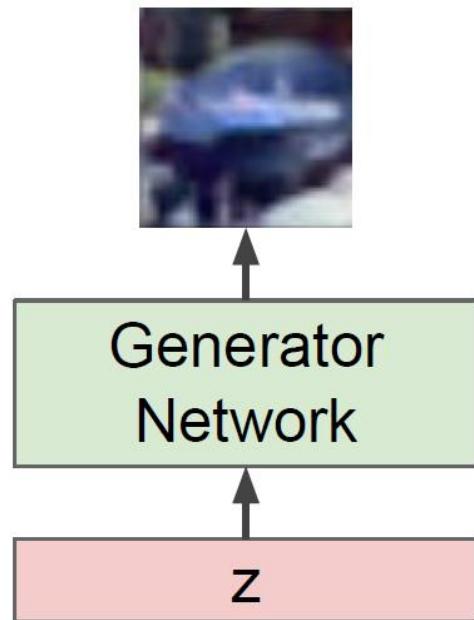
1. We need an architecture that can generate an image
 - Recall upsampling architectures for dense prediction
 - Sample from a simple distribution, e.g. random noise.
 - Learn transformation to training distribution.



Output: Sample from
training distribution

**A neural network can be
used to represent
this complex
transformation?**

Input: Random noise



Designing a network for generative tasks

1. We need an architecture that can generate an image
 - Recall upsampling architectures for dense prediction
 - Sample from a simple distribution, e.g. random noise.
 - Learn transformation to training distribution.

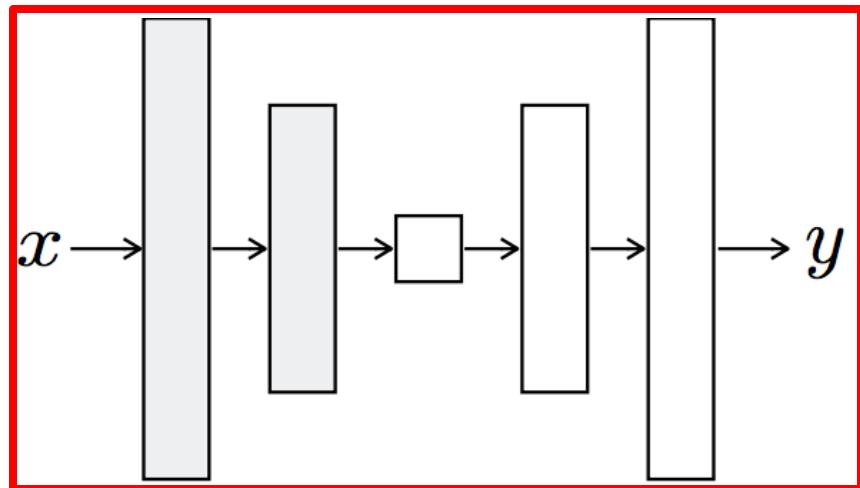
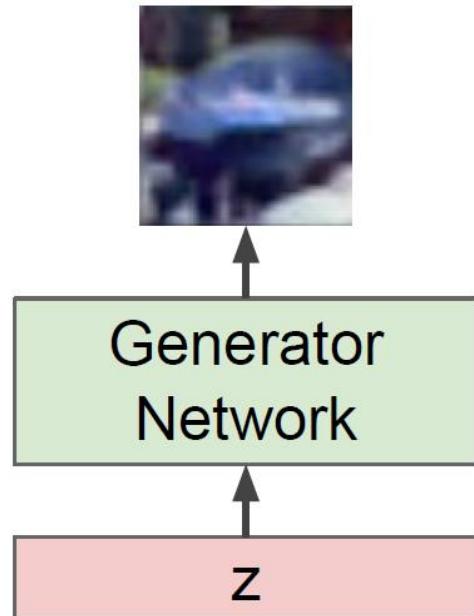


Image-to-image translation

Output: Sample from
training distribution

**A neural network can be
used to represent
this complex
transformation?**

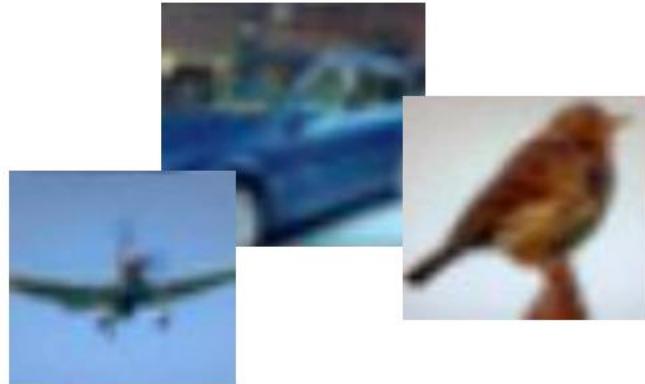
Input: Random noise



Designing a network for generative tasks

1. We need an architecture that can generate an image
 - Recall upsampling architectures for dense prediction
2. We need to design the right loss function

Learning to sample



Training data $x \sim p_{\text{data}}$



Generated samples $x \sim p_{\text{model}}$

We want to learn p_{model} that matches p_{data}

Generative adversarial networks

- Train two networks with opposing objectives:
 - **Generator:** learns to generate samples
 - **Discriminator:** learns to distinguish between generated and real samples

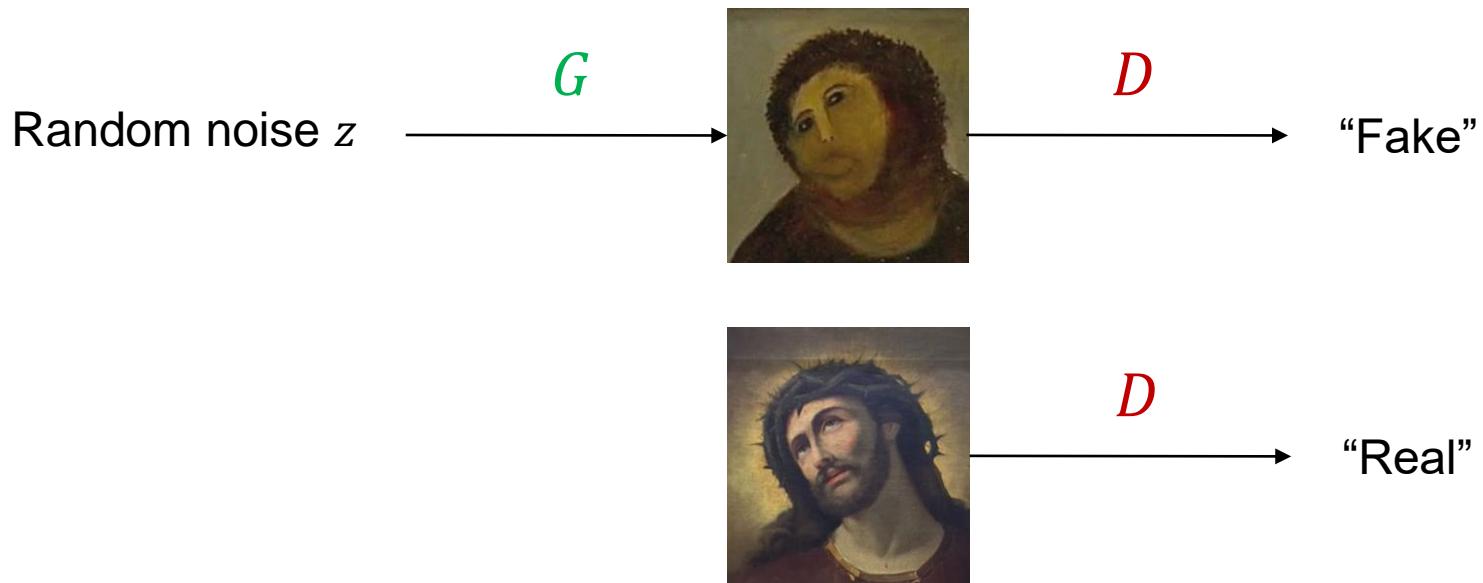
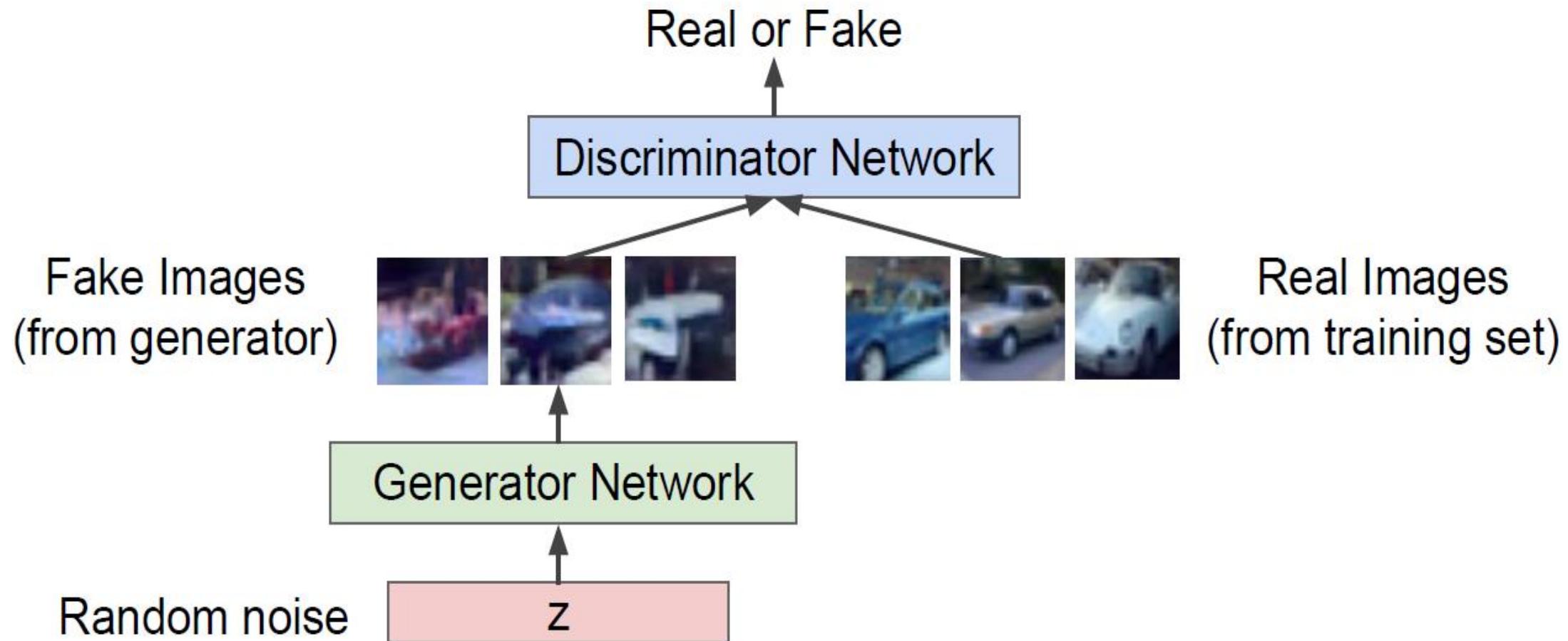


Figure adapted
from [F. Fleuret](#)

Generative adversarial networks

Generator network: try to fool the discriminator by generating real-looking images

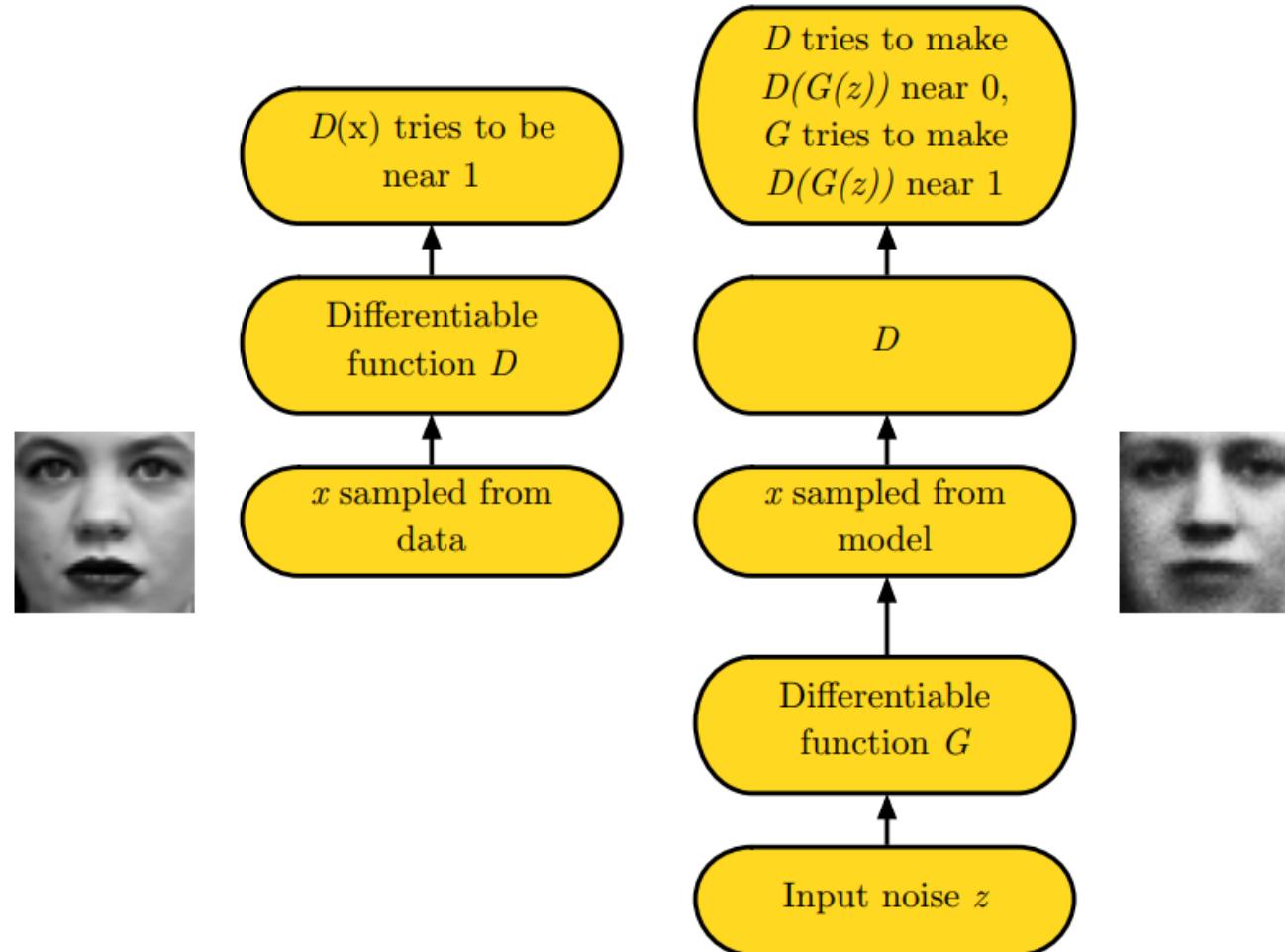
Discriminator network: try to distinguish between real and fake images



Generative adversarial networks

Generator network: try to fool the discriminator by generating real-looking images

Discriminator network: try to distinguish between real and fake images



GAN objective

- The discriminator $D(x)$ should output the probability that the sample x is real
 - That is, we want $D(x)$ to be close to 1 for real data and close to 0 for fake
- Expected conditional log likelihood for real and generated data:

$$\begin{aligned} & \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x \sim p_{\text{gen}}} \log(1 - D(x)) \\ &= \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \end{aligned}$$

We seed the generator with noise z
drawn from a simple distribution p
(Gaussian or uniform)

GAN objective

$$V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$$

- The discriminator wants to correctly distinguish real and fake samples:

$$D^* = \arg \max_D V(G, D)$$

- The generator wants to fool the discriminator:

$$G^* = \arg \min_G V(G, D)$$

- Train the generator and discriminator jointly in a *minimax game*

GAN objective: Theoretical properties

$$V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$$

- Assuming unlimited capacity for generator and discriminator and unlimited training data:
 - The objective $\min_G \max_D V(G, D)$ is equivalent to *Jensen-Shannon divergence* between p_{data} and p_{gen} and global optimum (*Nash equilibrium*) is given by $p_{\text{data}} = p_{\text{gen}}$

GAN objective: Theoretical properties

$$V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$$

- Assuming unlimited capacity for generator and discriminator and unlimited training data:
 - The objective $\min_G \max_D V(G, D)$ is equivalent to *Jensen-Shannon divergence* between p_{data} and p_{gen} and global optimum (*Nash equilibrium*) is given by $p_{\text{data}} = p_{\text{gen}}$
 - If at each step, D is allowed to reach its optimum given G , and G is updated to decrease $V(G, D)$, then p_{gen} will eventually converge to p_{data}

GAN training

$$V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$$

- Alternate between

- *Gradient ascent* on discriminator:

$$D^* = \arg \max_D V(G, D)$$

- *Gradient descent* on generator (minimize log-probability of discriminator being right):

$$\begin{aligned} G^* &= \arg \min_G V(G, D) \\ &= \arg \min_G \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \end{aligned}$$

- In practice, do *gradient ascent* on generator (maximize log-probability of discriminator being wrong):

$$G^* = \arg \max_G \mathbb{E}_{z \sim p} \log(D(G(z)))$$

Non-saturating GAN loss (NSGAN)

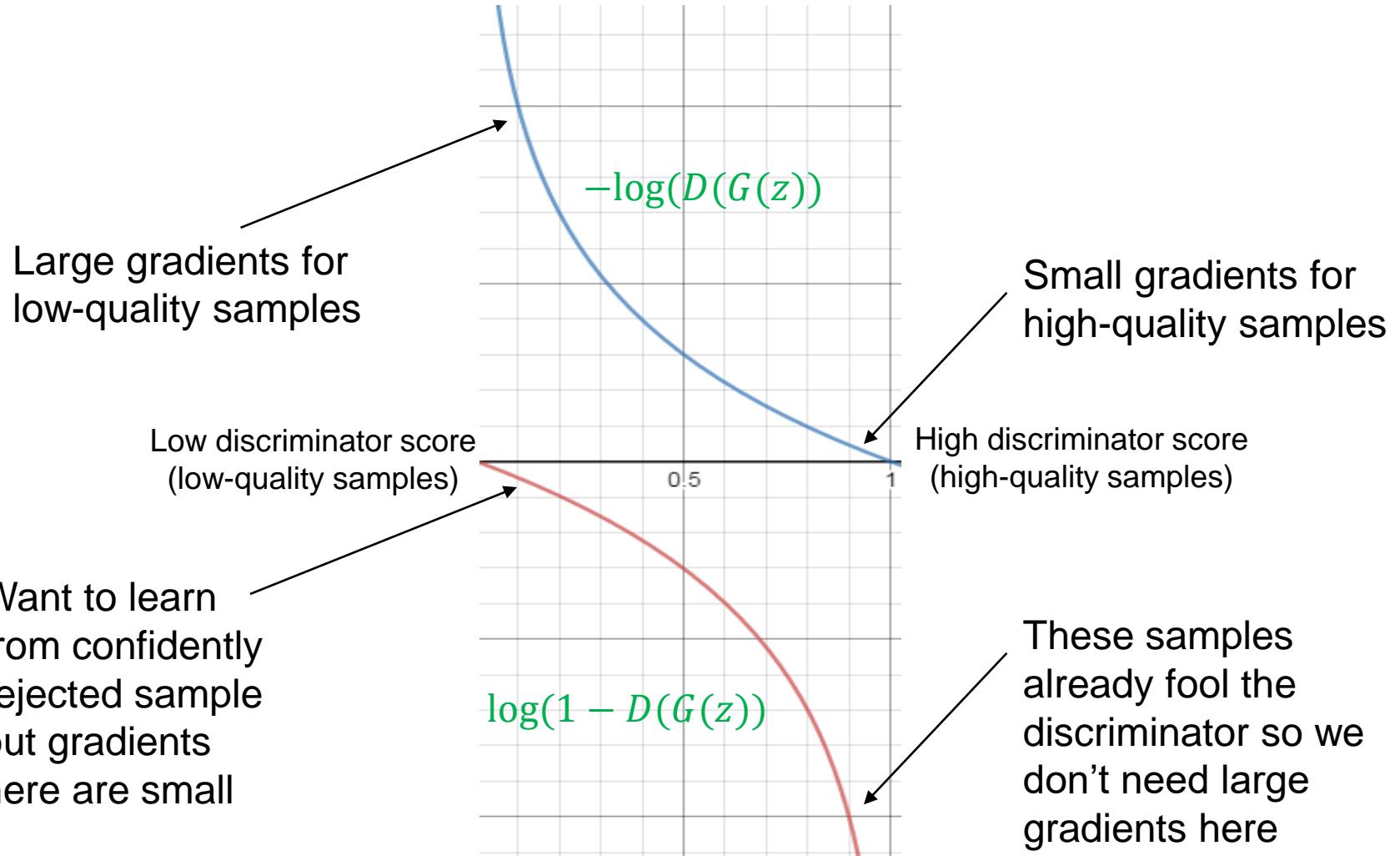
$$\min_{w_G} \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \quad \text{vs.} \quad \max_{w_G} \mathbb{E}_{z \sim p} \log(D(G(z)))$$

Minimize log-probability of
discriminator being right

Maximize log-probability of
discriminator being wrong

Non-saturating GAN loss (NSGAN)

$$\min_{w_G} \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \quad \text{vs.} \quad \max_{w_G} \mathbb{E}_{z \sim p} \log(D(G(z)))$$



NSGAN training algorithm

- Update discriminator:
 - Repeat for k steps:
 - Sample mini-batch of noise samples z_1, \dots, z_m and mini-batch of real samples x_1, \dots, x_m
 - Update parameters of D by stochastic gradient ascent on
$$\frac{1}{m} \sum_m [\log D(x_m) + \log(1 - D(G(z_m)))]$$
 - Update generator:
 - Sample mini-batch of noise samples z_1, \dots, z_m
 - Update parameters of G by stochastic gradient ascent on
$$\frac{1}{m} \sum_m \log D(G(z_m))$$
- Repeat until happy with results

NSGAN training algorithm

- Update discriminator:
 - Repeat for k steps:
 - Sample mini-batch of noise samples z_1, \dots, z_m and mini-batch of real samples x_1, \dots, x_m
 - Update parameters of D by stochastic gradient ascent on
$$\frac{1}{m} \sum_m [\log D(x_m) + \log(1 - D(G(z_m)))]$$

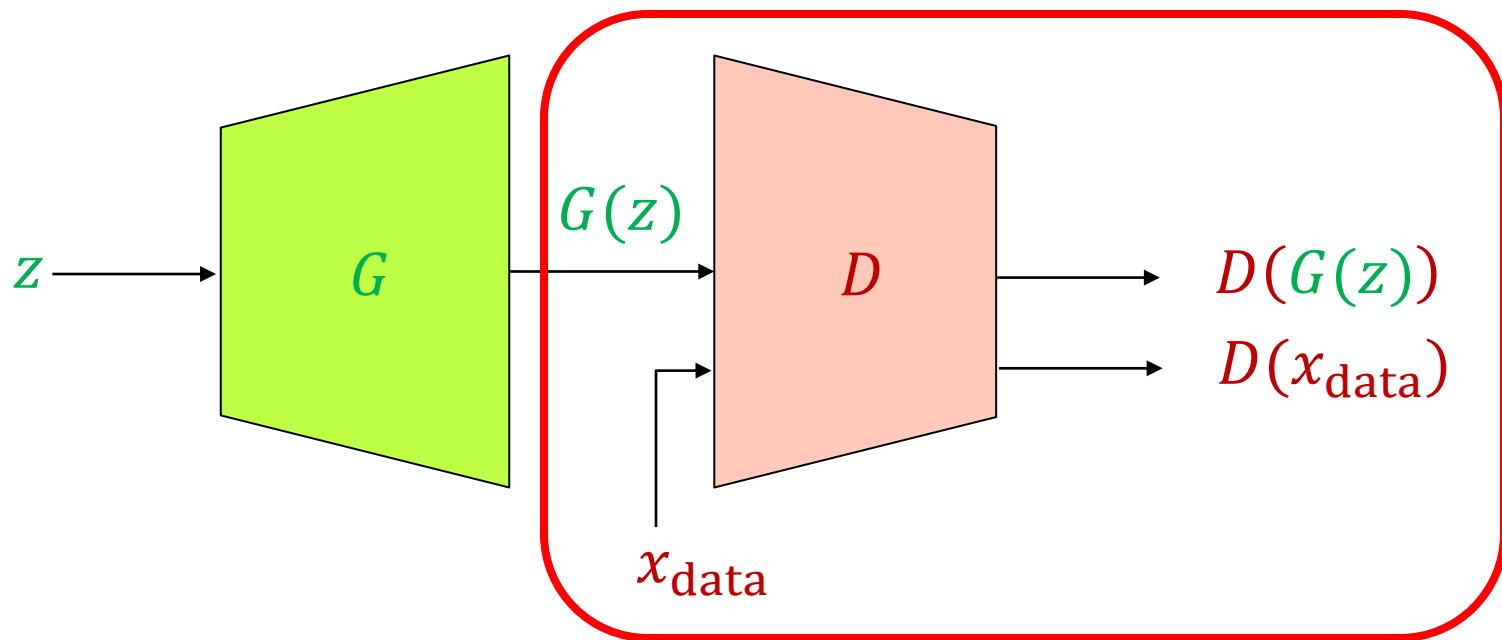
Some find $k=1$ more stable, others use $k > 1$, no best rule.

Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!

- Update generator:
 - Sample mini-batch of noise samples z_1, \dots, z_m
 - Update parameters of G by stochastic gradient ascent on
$$\frac{1}{m} \sum_m \log D(G(z_m))$$
- Repeat until happy with results

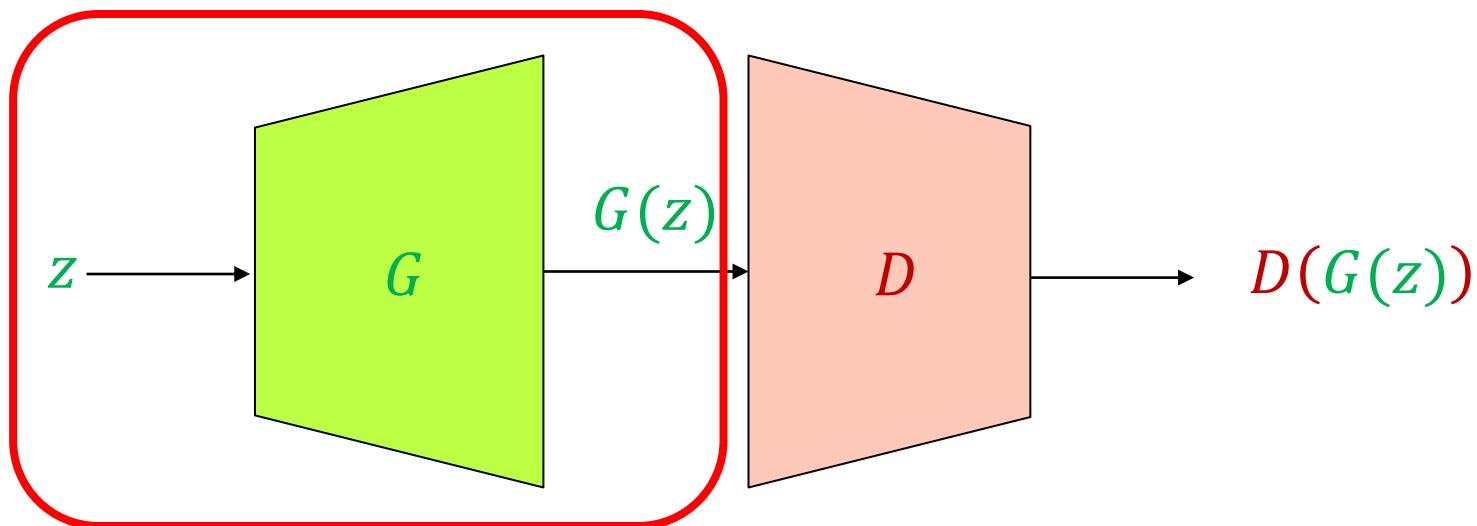
GAN: Conceptual picture

- Update discriminator: push $D(x_{\text{data}})$ close to 1 and $D(G(z))$ close to 0
 - The generator is a “black box” to the discriminator



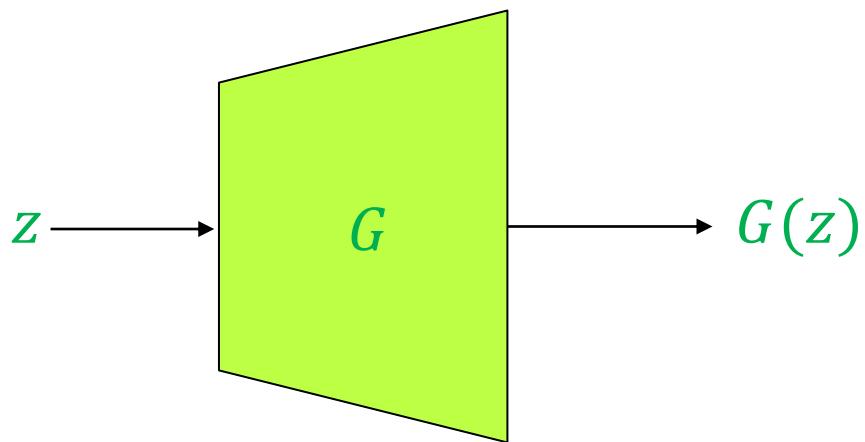
GAN: Conceptual picture

- Update generator: increase $D(G(z))$
 - Requires back-propagating through the composed generator-discriminator network (i.e., the discriminator cannot be a black box)
 - The generator is exposed to real data only via the output of the discriminator (and its gradients)



GAN: Conceptual picture

- Test time – the discriminator is discarded



Original GAN results

MNIST digits



Toronto Face Dataset



Nearest real image for
sample to the left

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair,
A. Courville, Y. Bengio, [Generative adversarial nets](#), NIPS 2014

Original GAN results

CIFAR-10 (FC networks)



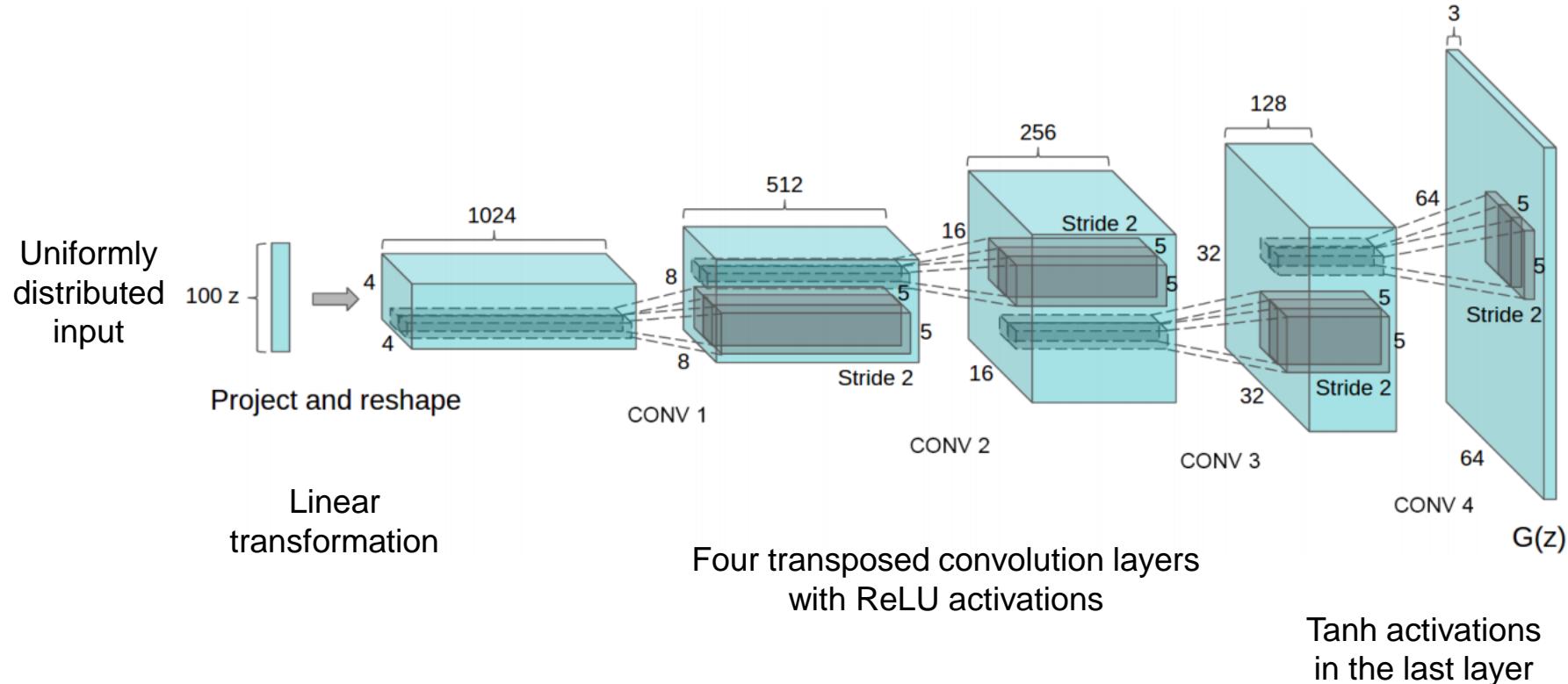
CIFAR-10 (conv networks)



I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair,
A. Courville, Y. Bengio, [Generative adversarial nets](#), NIPS 2014

DCGAN

- Early, influential convolutional architecture for generator



DCGAN

- Early, influential convolutional architecture for generator
- Discriminator architecture:
 - Don't use pooling, only strided convolutions
 - Use Leaky ReLU activations (sparse gradients cause problems for training)
 - Use only one FC layer before the softmax output
 - Use batch normalization after most layers (in the generator also)

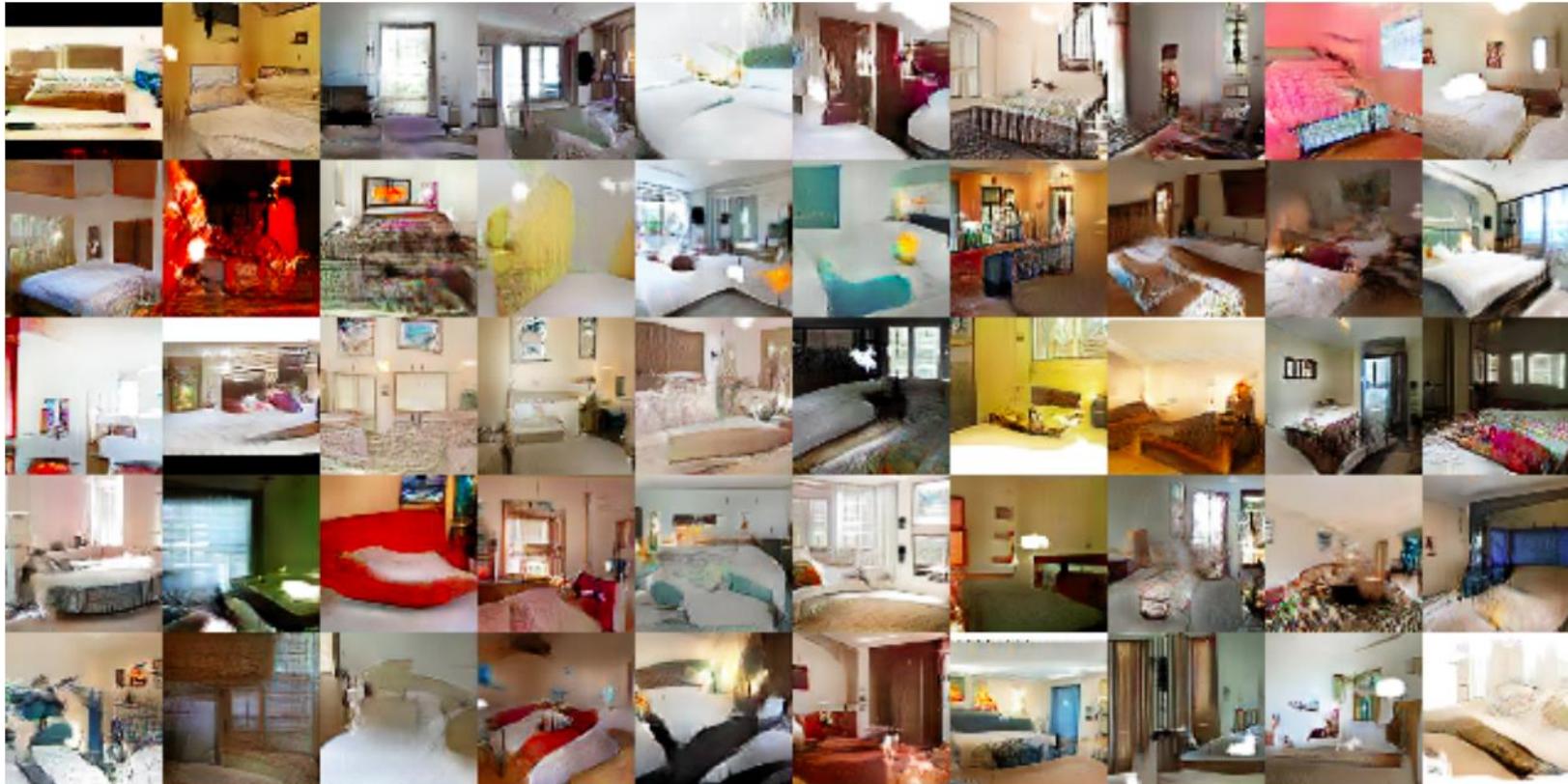
DCGAN results

Generated bedrooms after one epoch



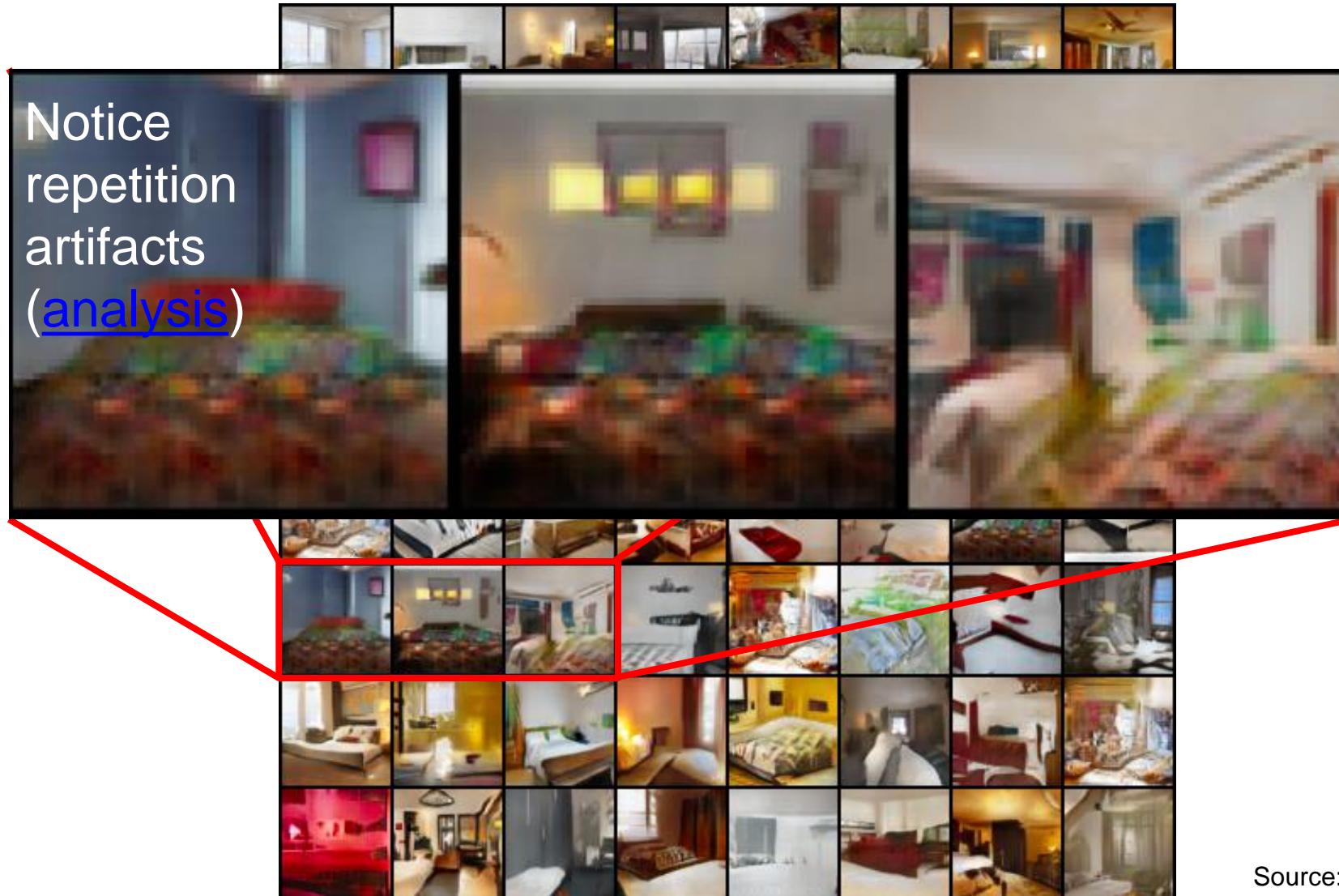
DCGAN results

Generated bedrooms after five epochs



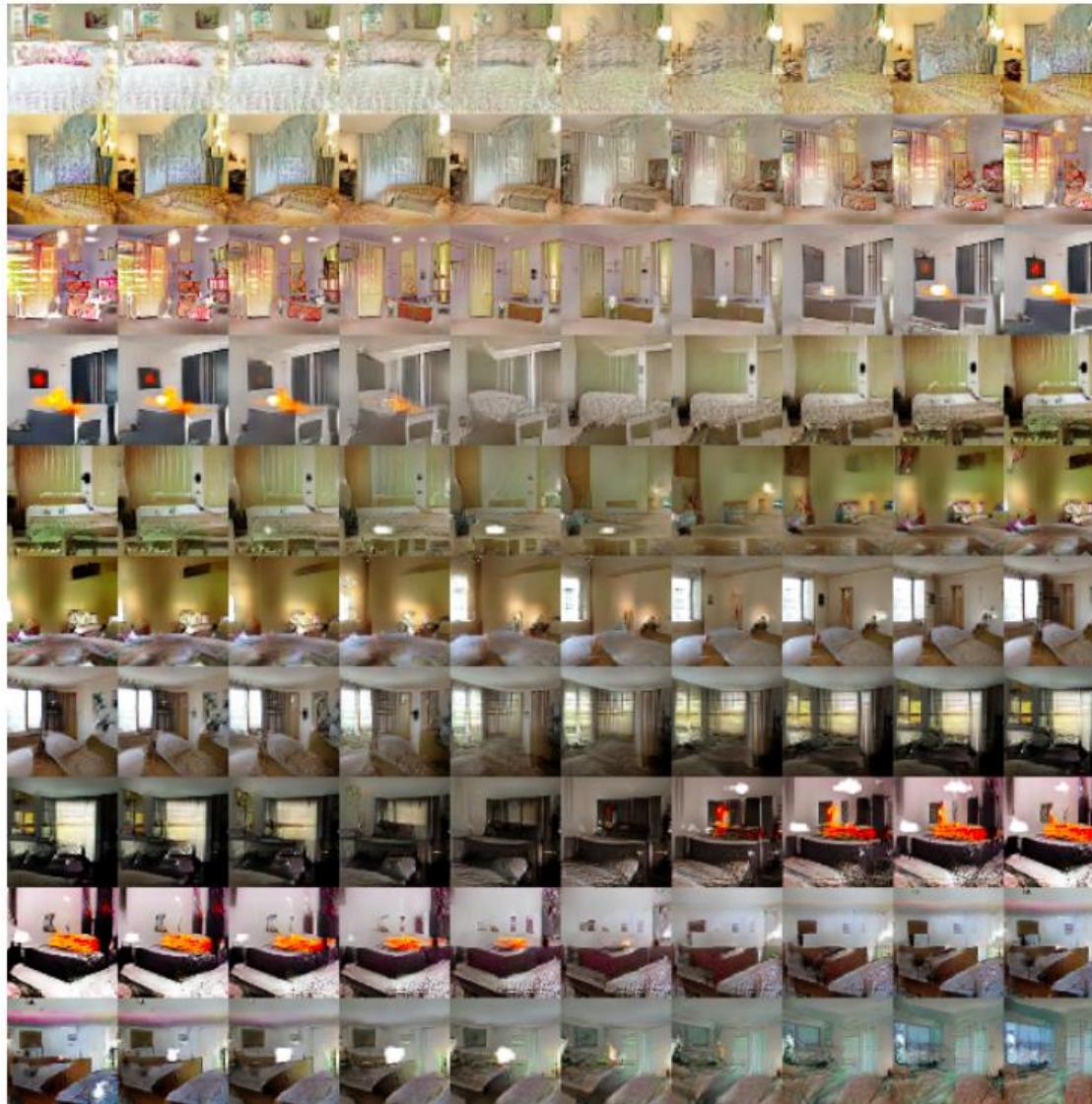
DCGAN results

More bedrooms



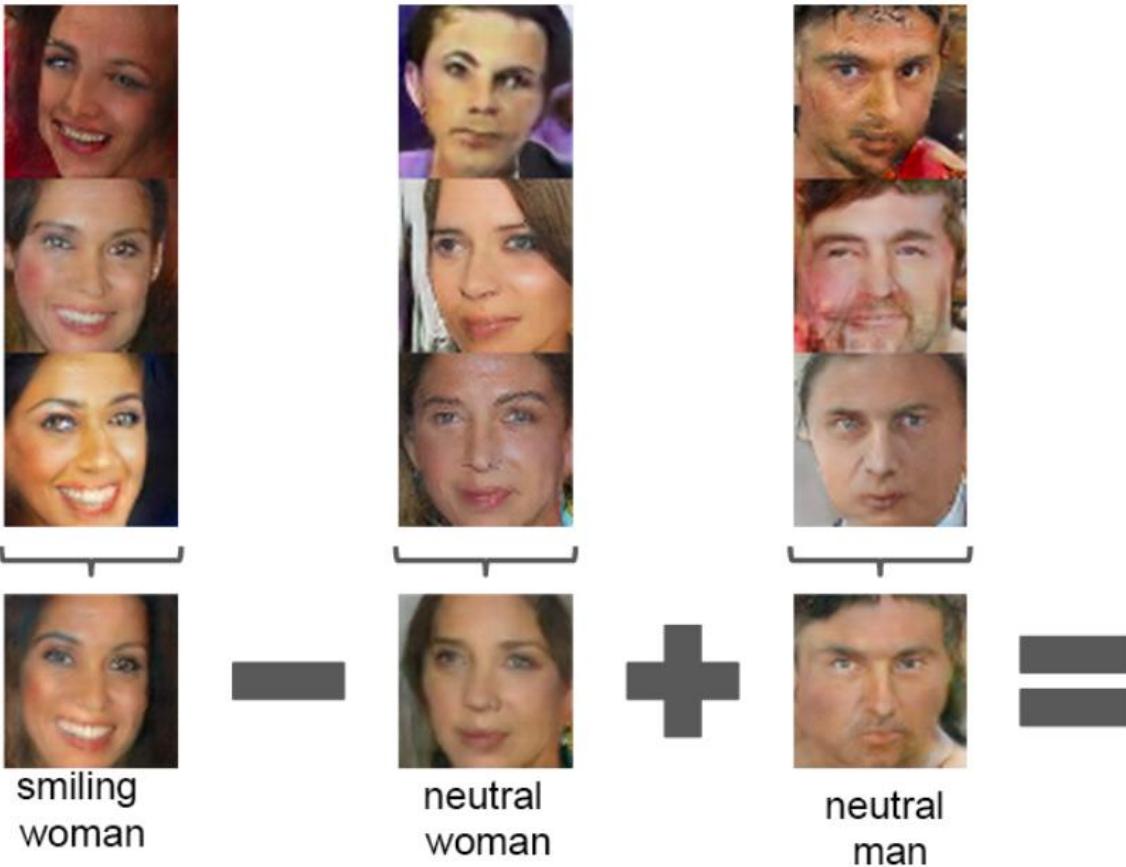
DCGAN results

Interpolation between different points in the z space



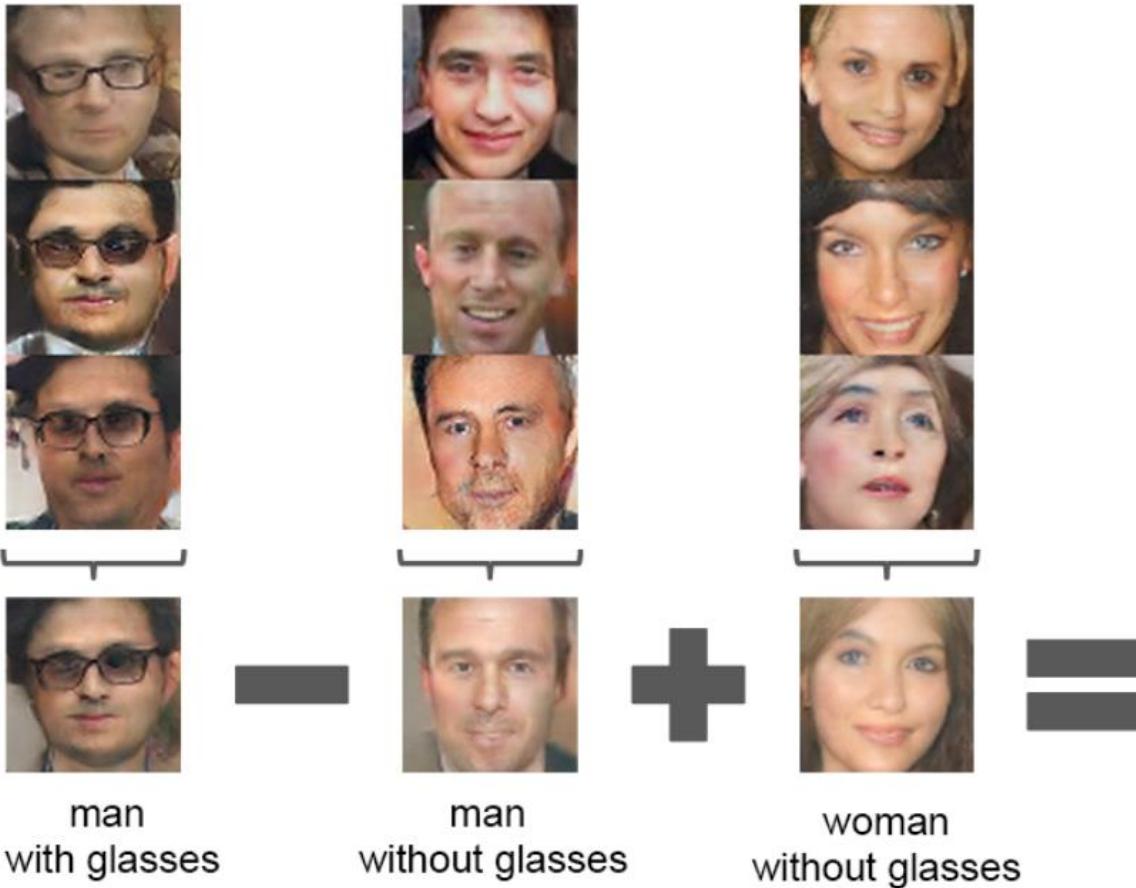
DCGAN results

- Vector arithmetic in the z space



DCGAN results

- Vector arithmetic in the z space



DCGAN results

- Pose transformation by adding a “turn” vector



Outline

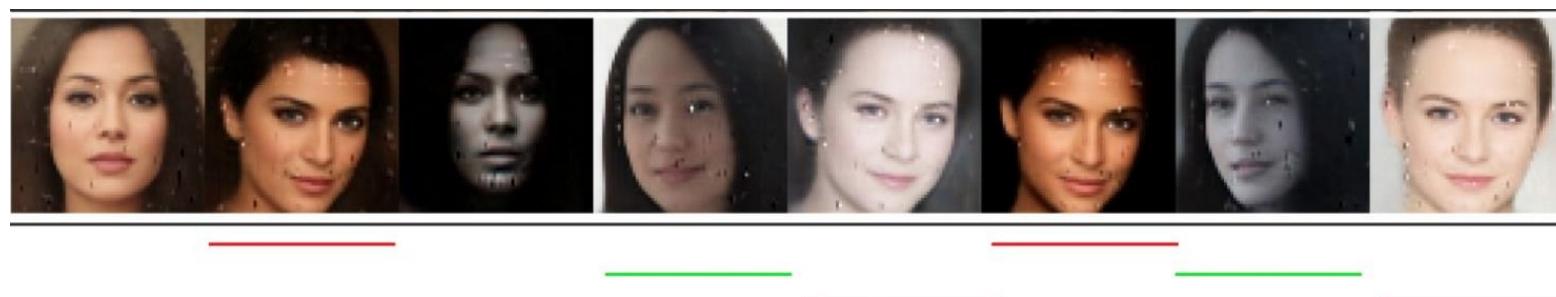
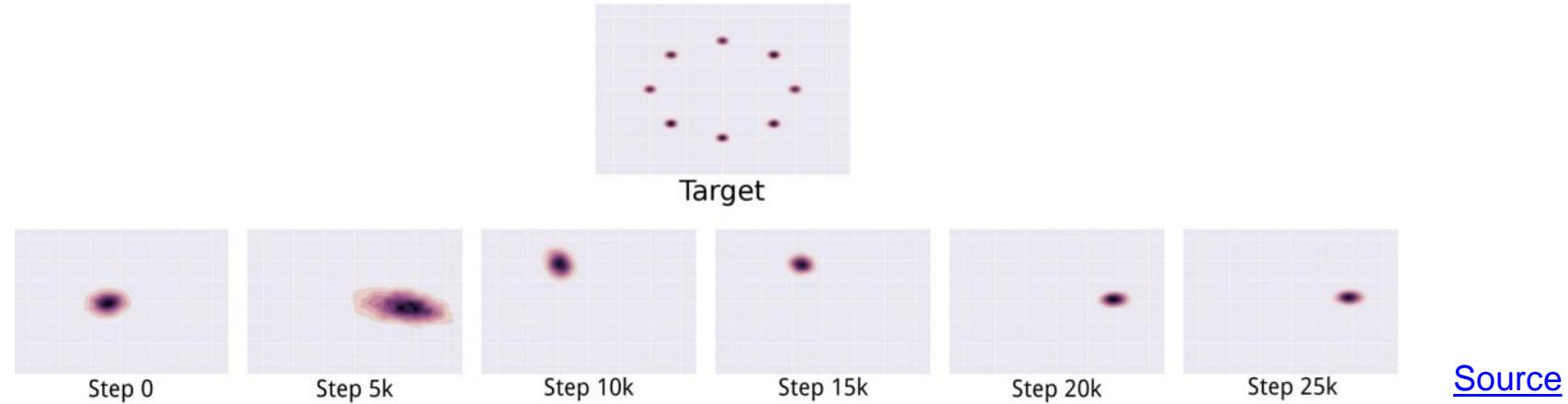
- Generative tasks
- Original GAN formulations
 - NSGAN
 - DCGAN
- Other popular formulations
 - WGAN, WGAN-GP
 - LSGAN

Problems with GAN training

- Stability
 - Parameters can oscillate or diverge, generator loss does not correlate with sample quality
 - Behavior very sensitive to hyperparameter selection

Problems with GAN training

- Mode collapse
 - Generator ends up modeling only a small subset of the training data



Some popular GAN flavors

- WGAN and improved WGAN (WGAN-GP)
- LSGAN

Wasserstein GAN (WGAN)

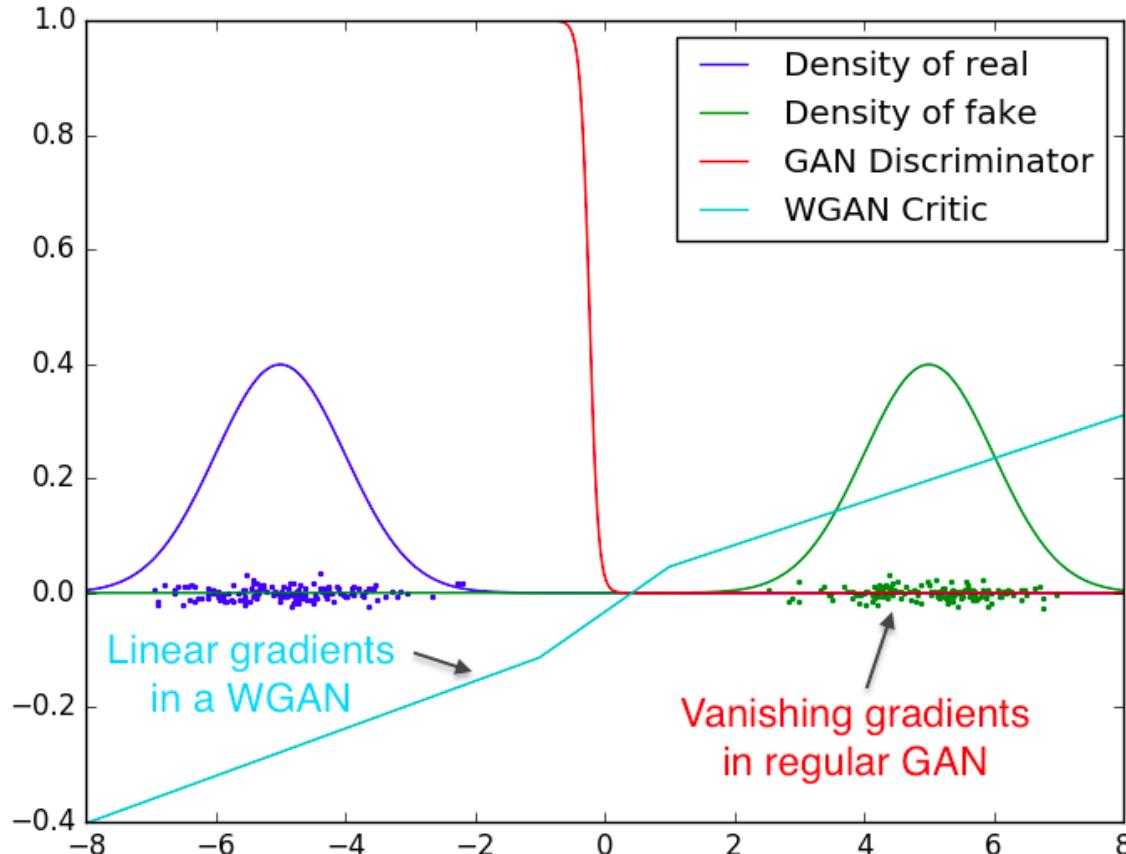
- Motivated by *Wasserstein or Earth mover's distance*, which is an alternative to JS divergence for comparing distributions
 - In practice, use linear activation instead of sigmoid in the discriminator and drop the logs from the objective:

$$\min_G \max_D [\mathbb{E}_{x \sim p_{\text{data}}} D(x) - \mathbb{E}_{z \sim p} D(G(z))]$$

- Due to theoretical considerations, important to ensure smoothness of discriminator
- This paper's suggested method is clipping weights to fixed range $[-c, c]$

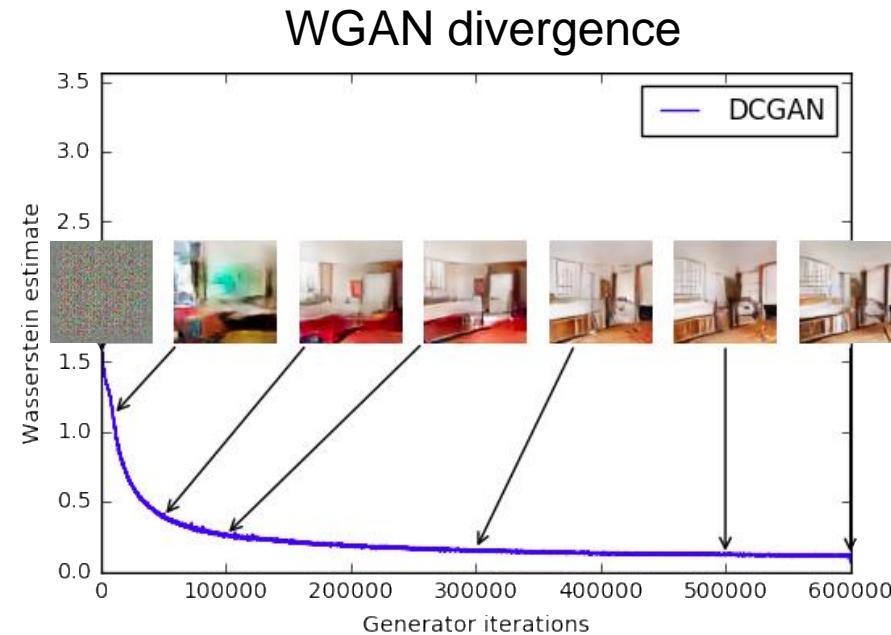
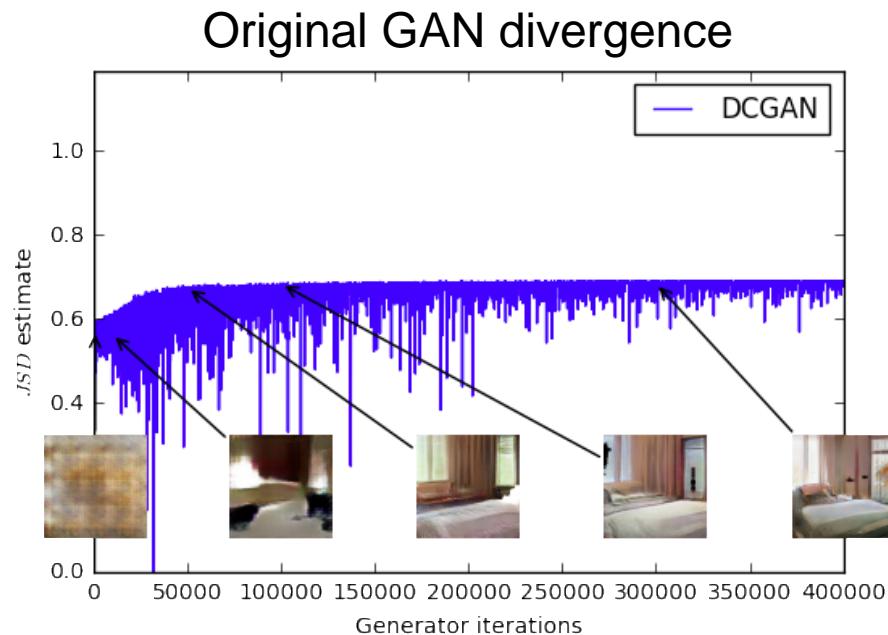
Wasserstein GAN (WGAN)

- Benefits (claimed)
 - Better gradients, more stable training



Wasserstein GAN (WGAN)

- Benefits (claimed)
 - Better gradients, more stable training
 - Objective function value is more meaningfully related to quality of generator output



Improved Wasserstein GAN (WGAN-GP)

- Weight clipping leads to problems with discriminator training
- Improved Wasserstein discriminator loss:

$$\mathbb{E}_{\tilde{x} \sim p_{\text{gen}}} D(\tilde{x}) - \mathbb{E}_{x \sim p_{\text{real}}} D(x)$$

$$+ \lambda \mathbb{E}_{\hat{x} \sim p_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

Unit norm gradient penalty on
points \hat{x} obtained by interpolating
real and generated samples

Improved Wasserstein GAN: Results

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G : DCGAN, D : DCGAN)			
G : No BN and a constant number of filters, D : DCGAN			
G : 4-layer 512-dim ReLU MLP, D : DCGAN			
No normalization in either G or D			
Gated multiplicative nonlinearities everywhere in G and D			
tanh nonlinearities everywhere in G and D			
101-layer ResNet G and D			

Least Squares GAN (LSGAN)

- Use least squares cost for generator and discriminator
 - Equivalent to minimizing Pearson χ^2 divergence

$$D^* = \arg \min_D [\mathbb{E}_{x \sim p_{\text{data}}} (D(x) - 1)^2 + \mathbb{E}_{z \sim p} (D(G(z)))^2]$$

Push discrim.
response on real
data close to 1 Push response on
generated data close to 0

$$G^* = \arg \min_G \mathbb{E}_{z \sim p} (D(G(z)) - 1)^2$$

Push response on
generated data close to 1

Least Squares GAN (LSGAN)

- Benefits (claimed)
 - Higher-quality images



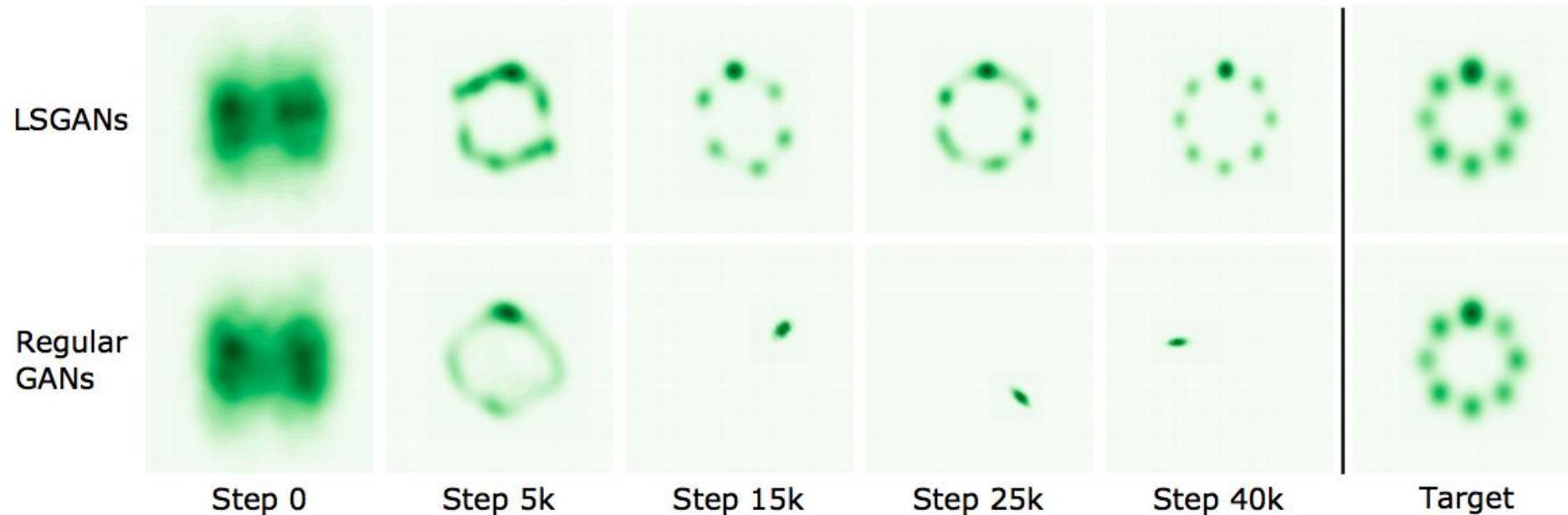
(a) Generated images (112×112) by LSGANs.



(b) Generated images (112×112) by DCGANs.

Least Squares GAN (LSGAN)

- Benefits (claimed)
 - Higher-quality images
 - More stable and resistant to mode collapse



Are GANs created equal?

- From the abstract:

“We find that most models can reach similar scores with enough hyperparameter optimization and random restarts. This suggests that improvements can arise from a higher computational budget and tuning more than fundamental algorithmic changes ... We did not find evidence that any of the tested algorithms consistently outperforms the non-saturating GAN introduced in Goodfellow et al. (2014)”

Outline

- Generative tasks
- Original GAN formulations
 - NSGAN
 - DCGAN
- Other popular formulations
 - WGAN, WGAN-GP
 - LSGAN
- State-of-the-art architectures
 - Progressive GAN, StyleGAN

Recent progress in GANs



Ian Goodfellow
@goodfellow_ian



4.5 years of GAN progress on face generation.

arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434

arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196

arxiv.org/abs/1812.04948



6:40 PM · Jan 14, 2019



Recent progress in GANs

EBGAN (2017)



ENERGY-
BASED
GAN

BigGAN (2018)



Progressive GANs

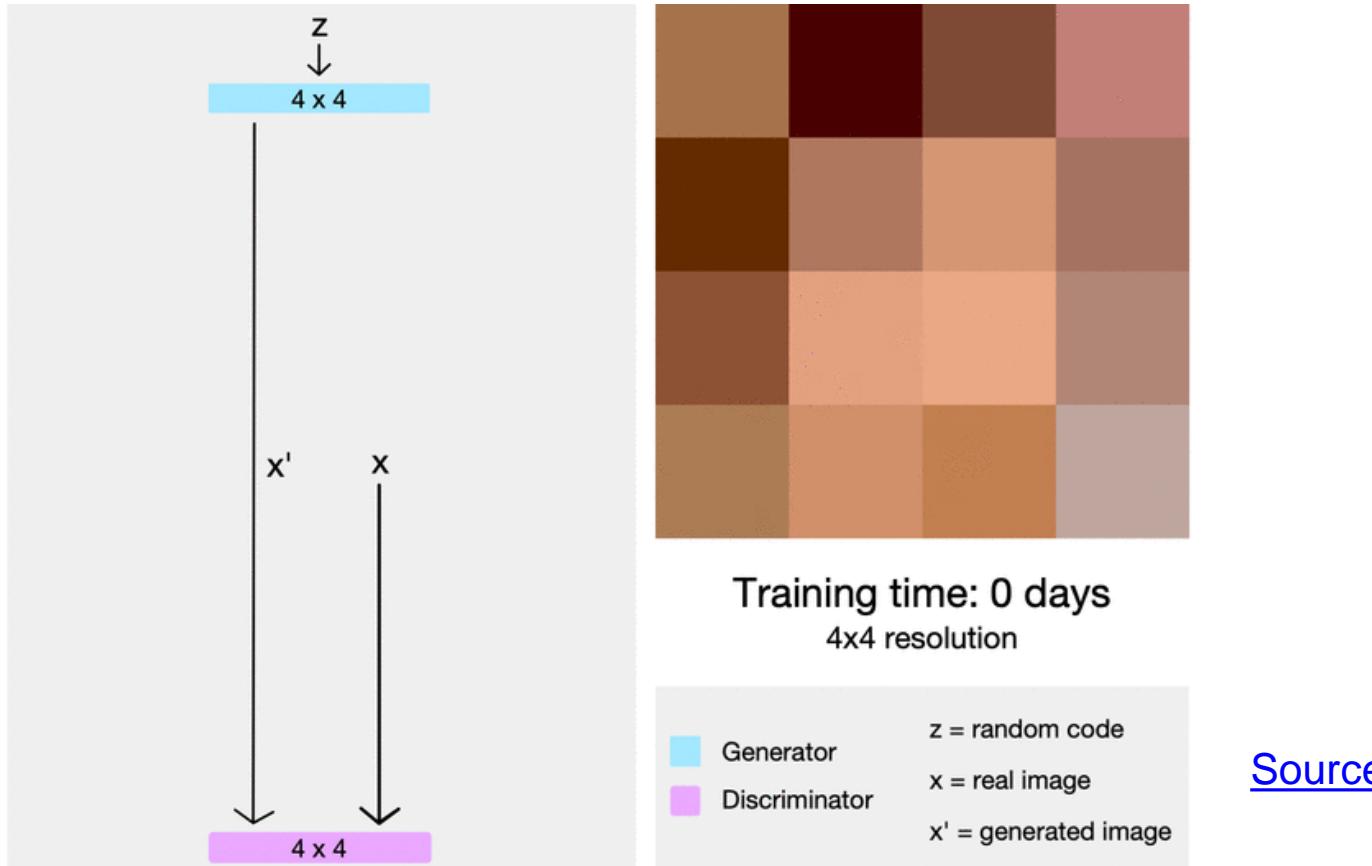
Realistic face images up to 1024 x 1024 resolution



T. Karras, T. Aila, S. Laine, J. Lehtinen. [Progressive Growing of GANs for Improved Quality, Stability, and Variation](#). ICLR 2018

Progressive GANs

- Key idea: train lower-resolution models, gradually add layers corresponding to higher-resolution outputs

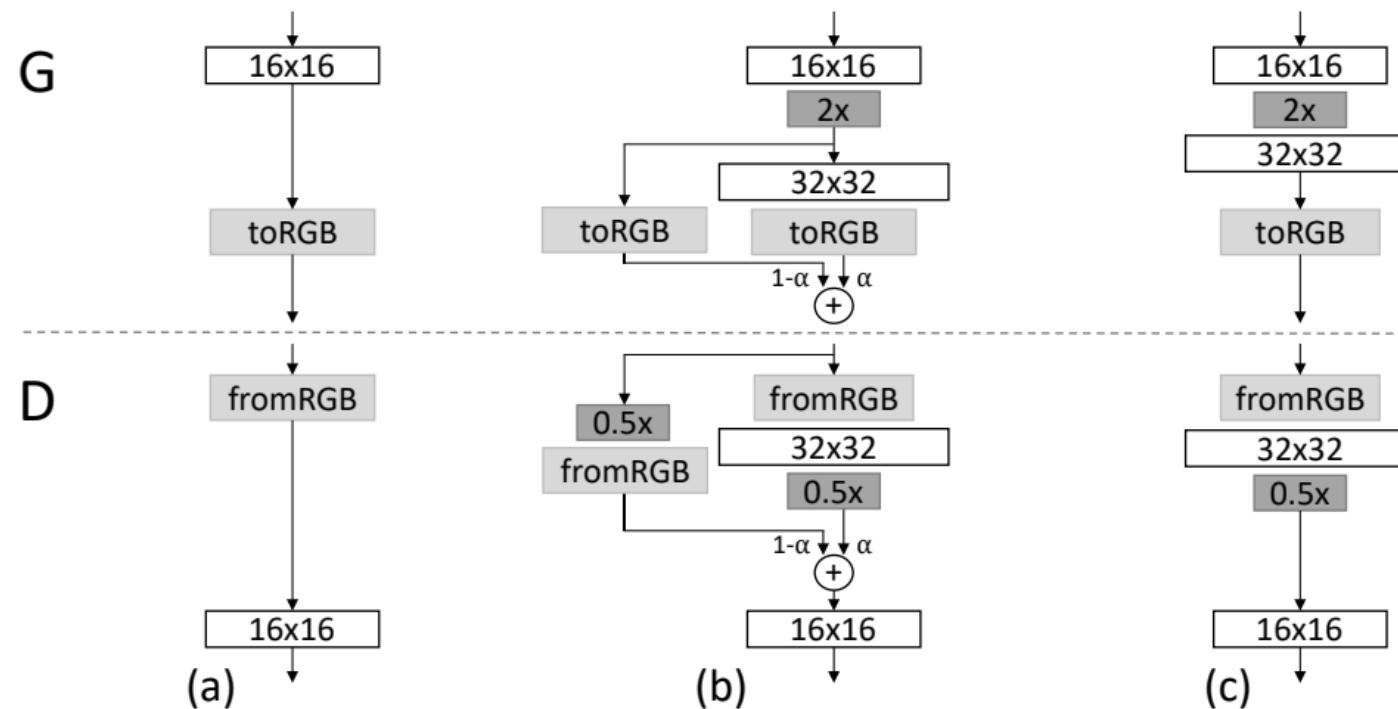


T. Karras, T. Aila, S. Laine, J. Lehtinen. [Progressive Growing of GANs for Improved Quality, Stability, and Variation](#). ICLR 2018

Progressive GANs

- Key idea: train lower-resolution models, gradually add layers corresponding to higher-resolution outputs

Transition from 16x16 to 32x32 images



Progressive GANs: Implementation details

- Loss: WGAN-GP loss (preferred) or LSGAN
- Architectures:
 - Nearest neighbor upsampling (2×2 replication) followed by regular convolutions instead of transposed conv layers
 - Average pooling instead of striding for downsampling in discriminator
 - Leaky ReLUs used in discriminator and generator
 - Per-pixel response normalization in generator: rescale feature vector in each pixel to unit length after each conv layer
- Use of minibatch standard deviation in discriminator (append to feature map)
- Exponential moving average of generator weights for display

Progressive GANs: Results

256 x 256 results for LSUN categories



POTTEDPLANT

HORSE

SOFA

BUS

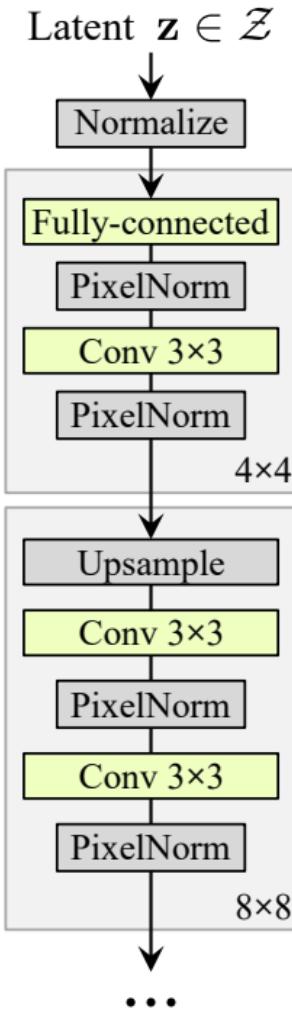
CHURCHOUTDOOR

BICYCLE

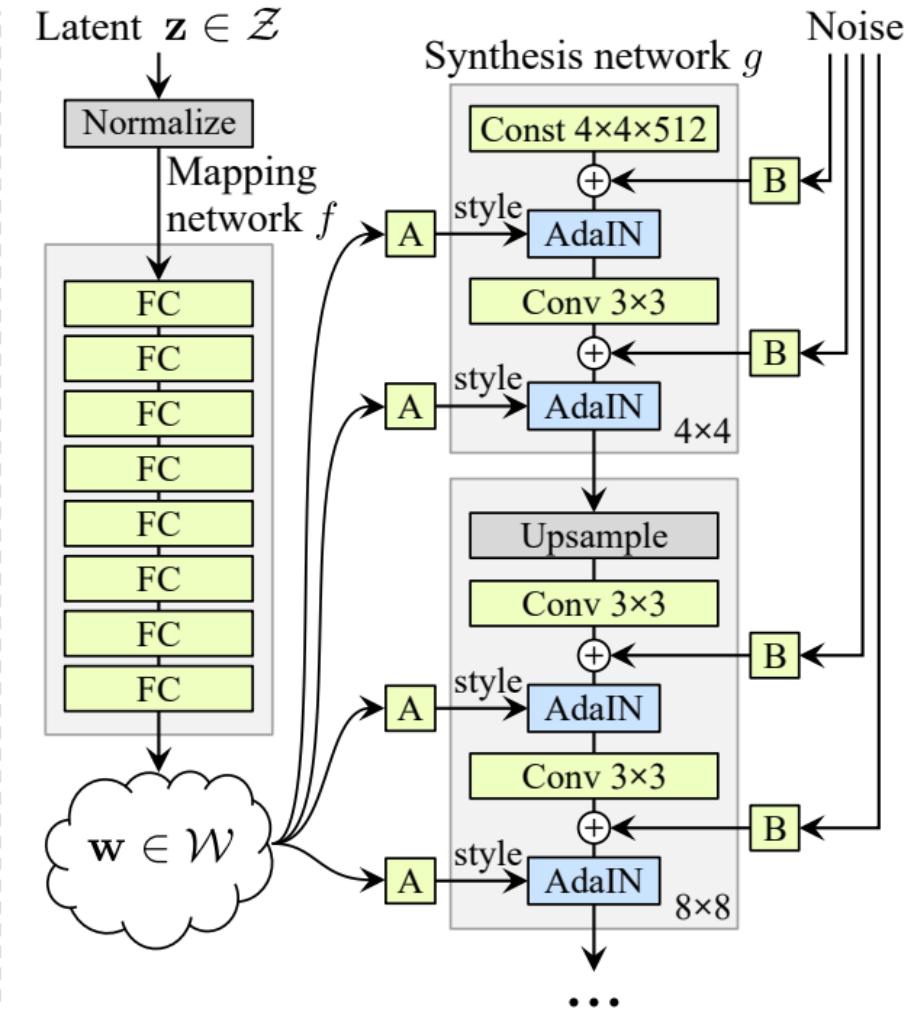
TVMONITOR

StyleGAN

- Built on top of Progressive GAN
- Start with learned constant (instead of noise vector)
- Use a mapping network to produce a *style code* w using learned affine transformations A
- Use *adaptive instance normalization* (AdaIN): scale and bias each feature map using learned style values
- Add noise after each convolution and before nonlinearity (enables stochastic detail)



(a) Traditional

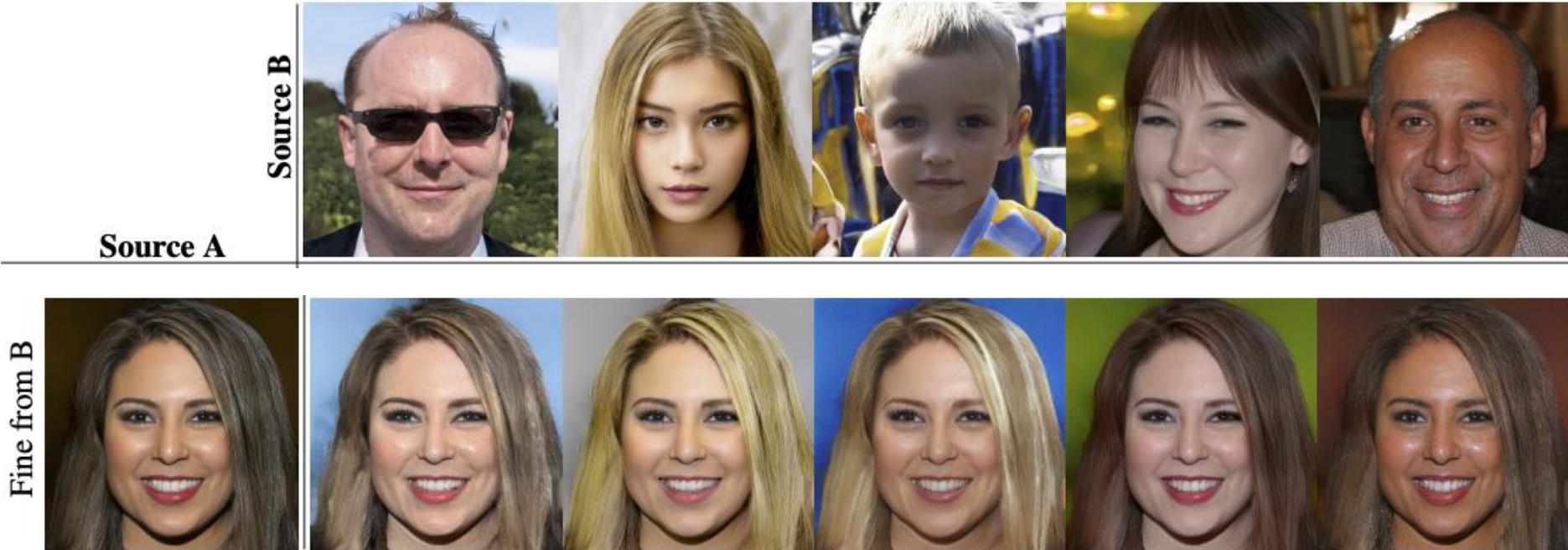


(b) Style-based generator

StyleGAN: Results



Mixing styles



“Two sets of images were generated from their respective latent codes (sources A and B); the rest of the images were generated by copying a specified subset of styles from source B and taking the rest from source A.”

Mixing styles

Middle styles from source B



Source A

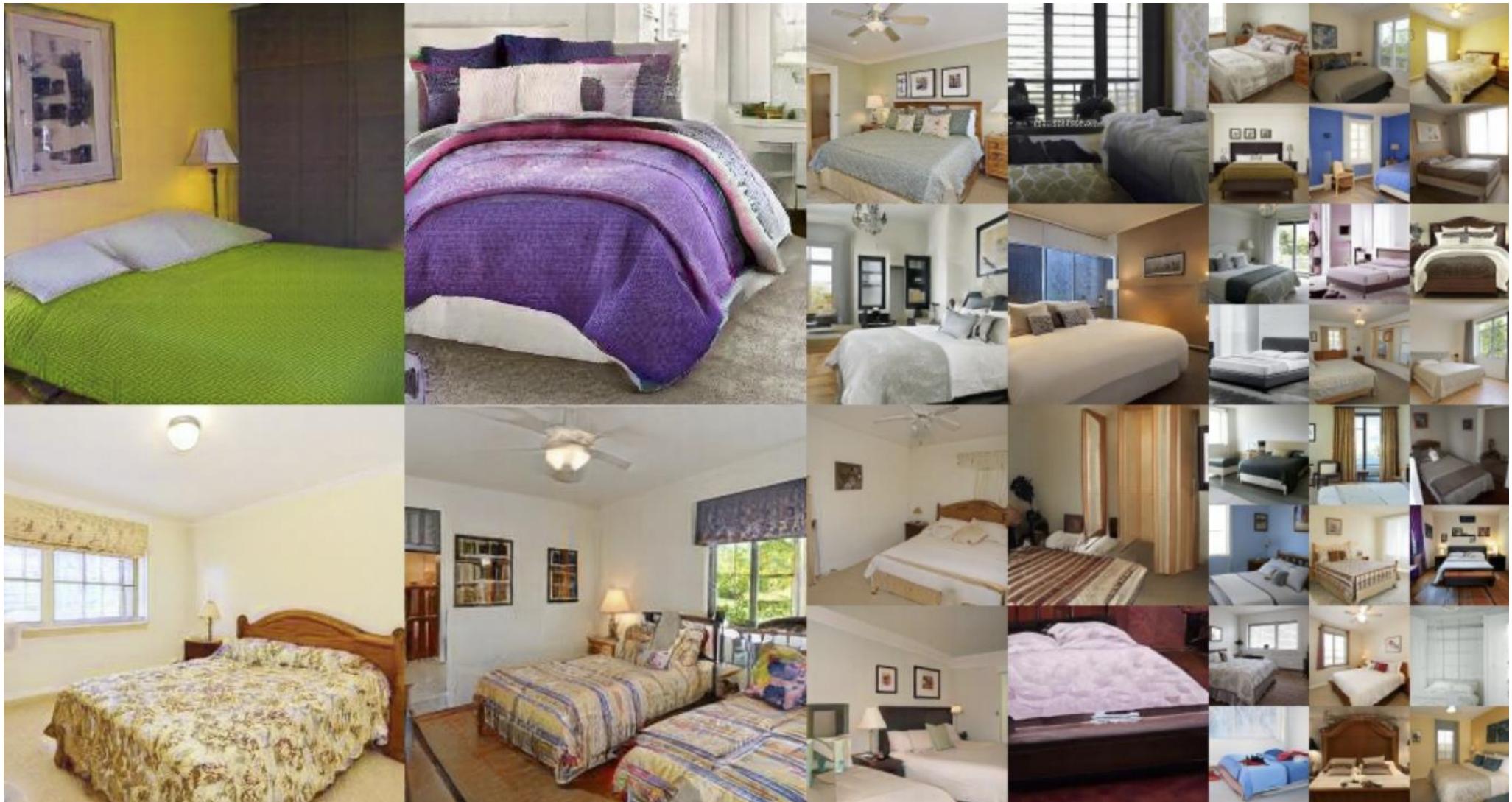
Source B

Mixing styles

Coarse styles from source B



StyleGAN: Bedrooms



StyleGAN: Cars



StyleGAN2

- Change normalization, remove progressive growing to address StyleGAN artifacts



Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.



Figure 6. Progressive growing leads to “phase” artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

Outline

- Generative tasks
- Original GAN formulations
 - NSGAN
 - DCGAN
- Other popular formulations
 - WGAN, WGAN-GP
 - LSGAN
- State-of-the-art architectures
 - Progressive GAN, StyleGAN
- Evaluating GANs

How to evaluate GANs?

- Showing pictures of samples is not enough, especially for simpler datasets like MNIST, CIFAR, faces, bedrooms, etc.
- We cannot directly compute the likelihoods of high-dimensional samples (real or generated), or compare their distributions
- Many GAN approaches claim mainly to improve stability, which is hard to evaluate

GAN evaluation: Human studies

- Example: Turing test

Instructions

**Examples of real images**

**Examples of images generated by a computer**

We present you pictures that are either computer generated or are real photographs. Your task is to choose which one are which.

Images contain pictures of airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. If you cannot clearly recognize what's the class of the object, then it's likely to be a generated image.

SET CHECKBOX ON IMAGES THAT LOOK LIKE GENERATED BY A COMPUTER.

<input type="checkbox"/> 	<input type="checkbox"/> 	<input type="checkbox"/> 
<input type="checkbox"/> 	<input type="checkbox"/> 	<input type="checkbox"/> 
<input type="checkbox"/> 	<input type="checkbox"/> 	<input type="checkbox"/> 

Submit

GAN evaluation: Inception score (IS)

- Key idea: generators should produce images with a variety of recognizable object classes
- Defined as $IS(G) = \exp[\mathbb{E}_{x \sim G} KL(P(y|x) \parallel P(y))]$ where $P(y|x)$ is the posterior label distribution returned by an image classifier (e.g., InceptionNet) for sample x
 - If x contains a recognizable object, entropy of $P(y|x)$ should be low
 - If generator generates images of diverse objects, the marginal distribution $P(y)$ should have high entropy

GAN evaluation: Inception score (IS)

- Disadvantages
 - A GAN that simply memorizes the training data (overfitting) or outputs a single image per class (mode dropping) could still score well
 - Is sensitive to network weights, not necessarily valid for generative models not trained on ImageNet, can be gamed ([Barratt & Sharma 2018](#))



Figure 1. Sample of generated images achieving an Inception Score of 900.15. The maximum achievable Inception Score is 1000, and the highest achieved in the literature is on the order of 10.

GAN evaluation: Fréchet Inception Distance (FID)

- Key idea: fit simple distributions (Gaussians) to statistics of feature activations for real and generated data; estimate divergence parametrically
 - Pass generated samples through a network (InceptionNet), compute activations for a chosen layer
 - Estimate multivariate mean and covariance of activations, compute Fréchet distance to those of real data

GAN evaluation: Fréchet Inception Distance (FID)

- Key idea: fit simple distributions (Gaussians) to statistics of feature activations for real and generated data; estimate divergence parametrically
 - Pass generated samples through a network (InceptionNet), compute activations for a chosen layer
 - Estimate multivariate mean and covariance of activations, compute Fréchet distance to those of real data

In mathematics, the **Fréchet distance** is a [measure of similarity](#) between [curves](#) that takes into account the location and ordering of the points along the curves. It is named after [Maurice Fréchet](#).

----- Source: en.wikipedia.org

GAN evaluation: Fréchet Inception Distance (FID)

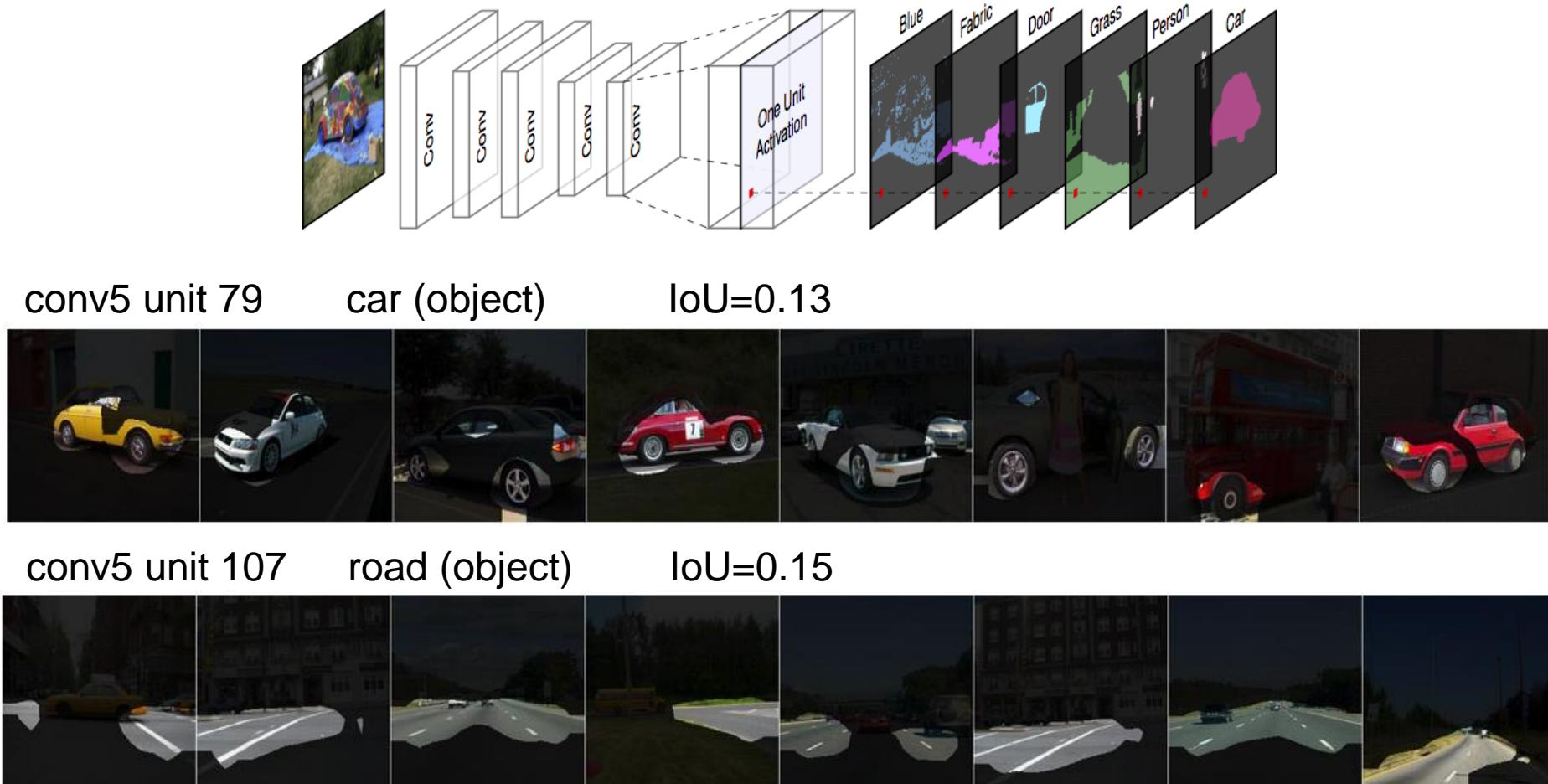
- Key idea: fit simple distributions (Gaussians) to statistics of feature activations for real and generated data; estimate divergence parametrically
 - Pass generated samples through a network (InceptionNet), compute activations for a chosen layer
 - Estimate multivariate mean and covariance of activations, compute Fréchet distance to those of real data
- Advantages: correlated with visual quality of samples and human judgment, can detect mode dropping (unlike IS)
- Disadvantage: cannot detect overfitting (like IS)

Outline

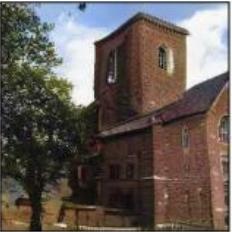
- Generative tasks
- Original GAN formulations
 - NSGAN
 - DCGAN
- Other popular formulations
 - WGAN, WGAN-GP
 - LSGAN
- State-of-the-art architectures
 - Progressive GAN, StyleGAN
- Evaluating GANs
- Visualizing and controlling GANs

GAN Dissection

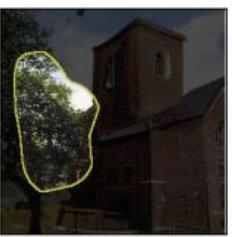
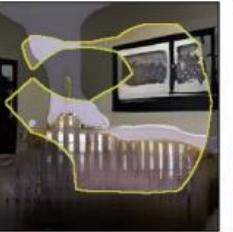
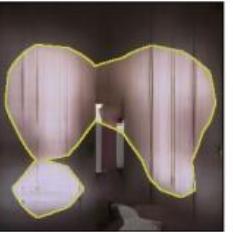
- Recall: network dissection



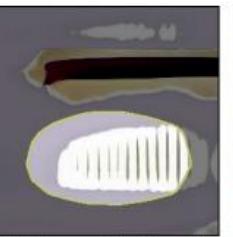
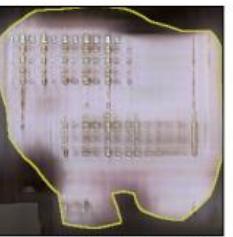
GAN Dissection



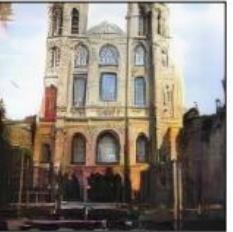
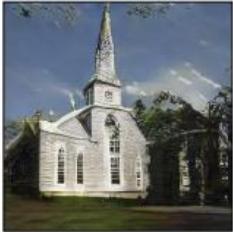
(a) Generate images of churches



(b) Identify GAN units that match trees



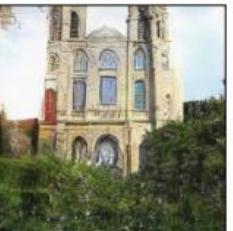
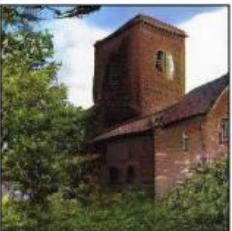
(e) Identify GAN units that cause artifacts



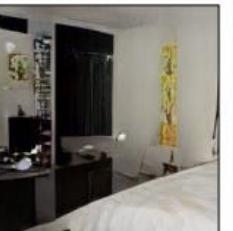
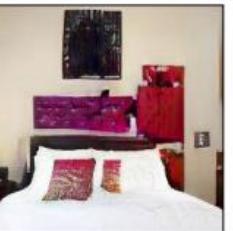
(c) Ablating units removes trees



(f) Bedroom images with artifacts



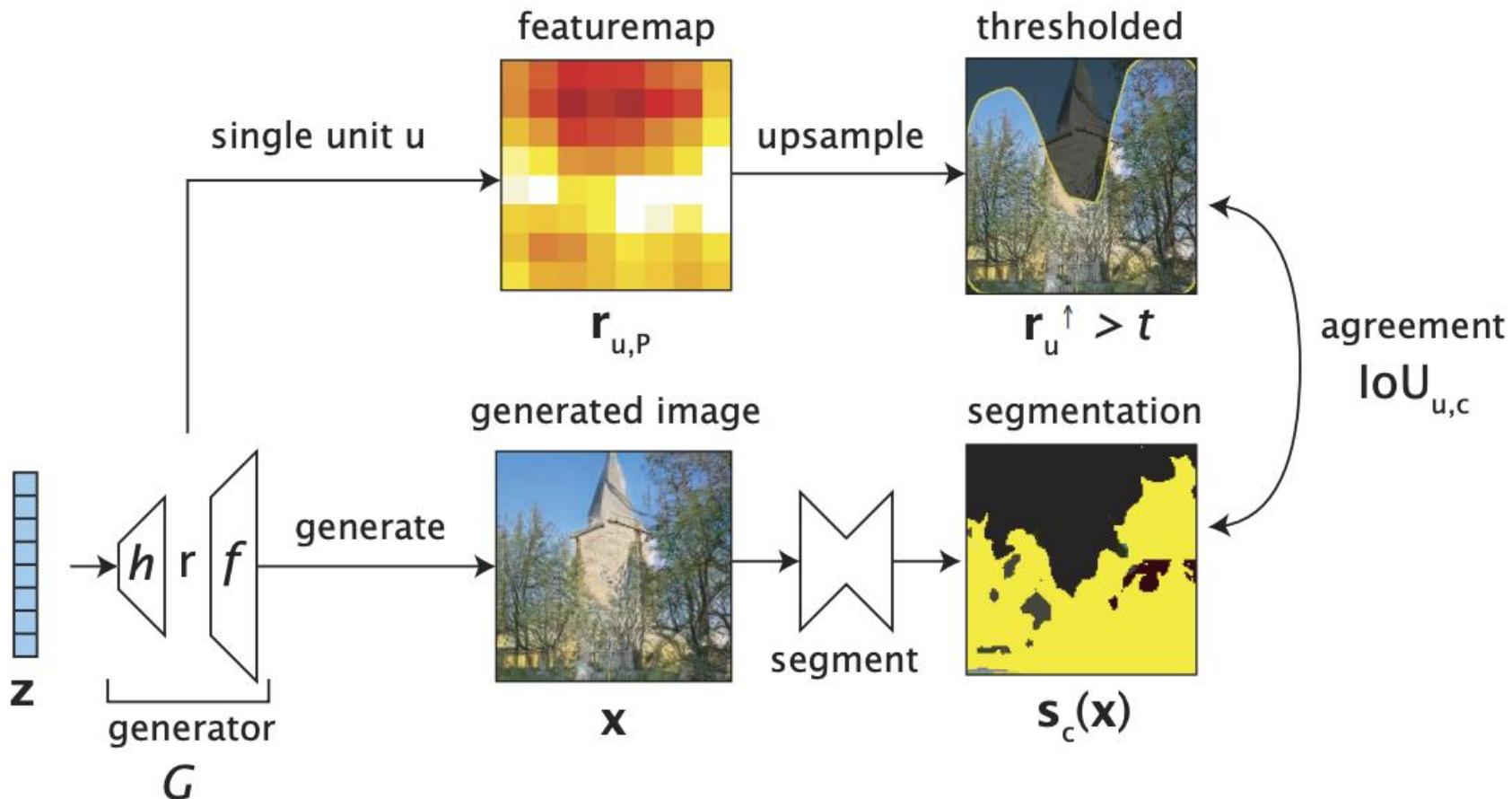
(d) Activating units adds trees



(g) Ablating “artifact” units improves results

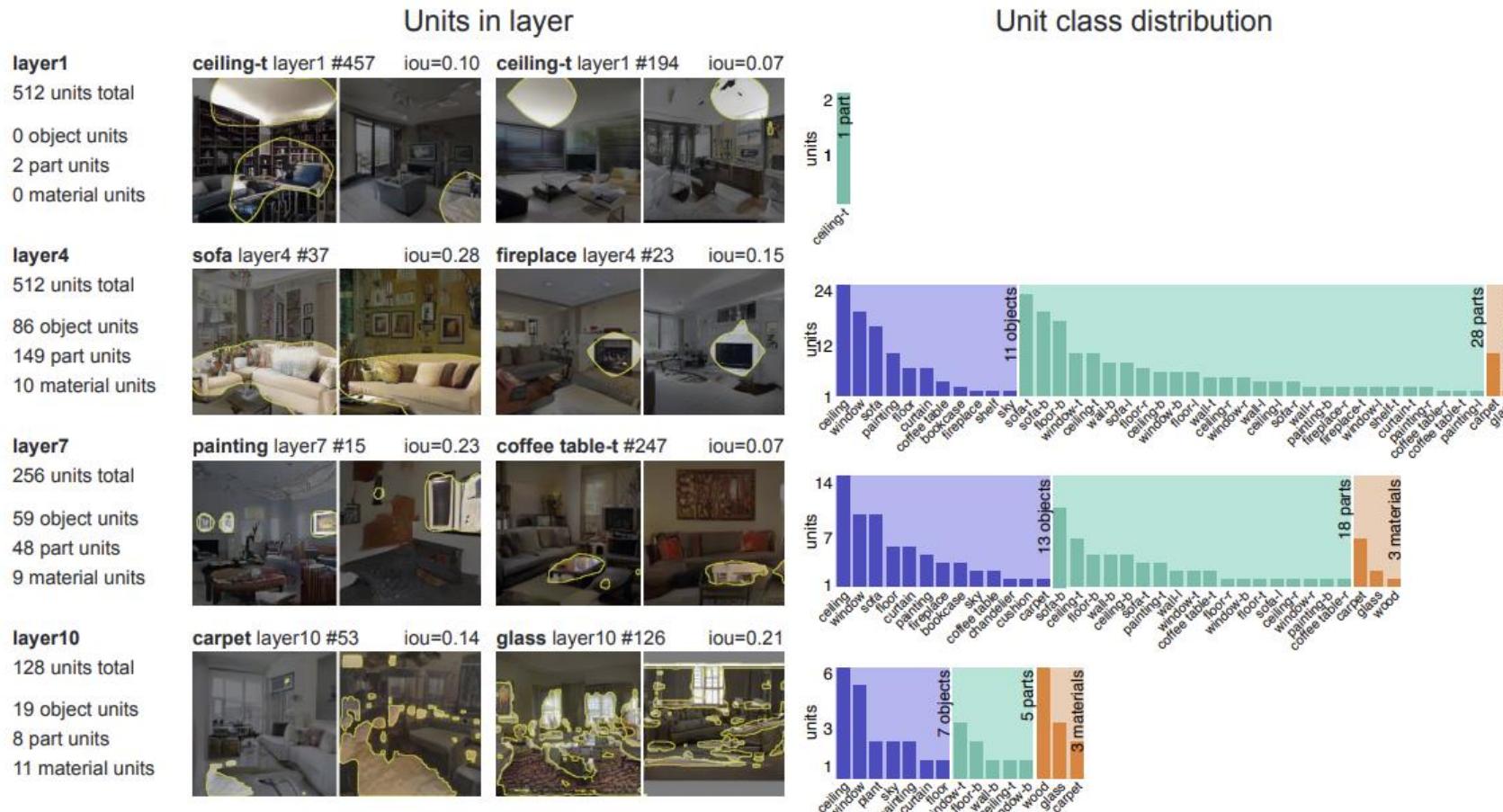
GAN Dissection

- Dissection:



GAN Dissection

- Interpreting units at different levels of a progressive GAN (trained on “bedroom”):



GANPaint demo



Input photo

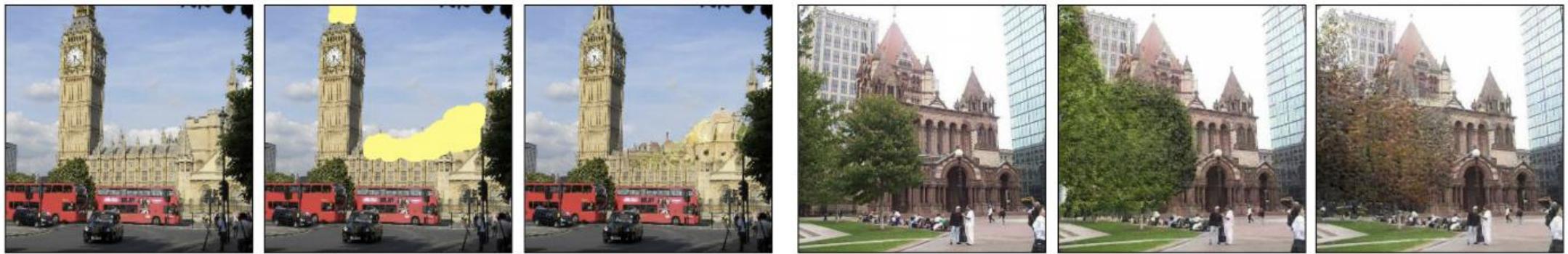
Remove chairs

Output result

Input photo

Add windows

Output result



Input photo

Change rooftops

Output result

Input photo

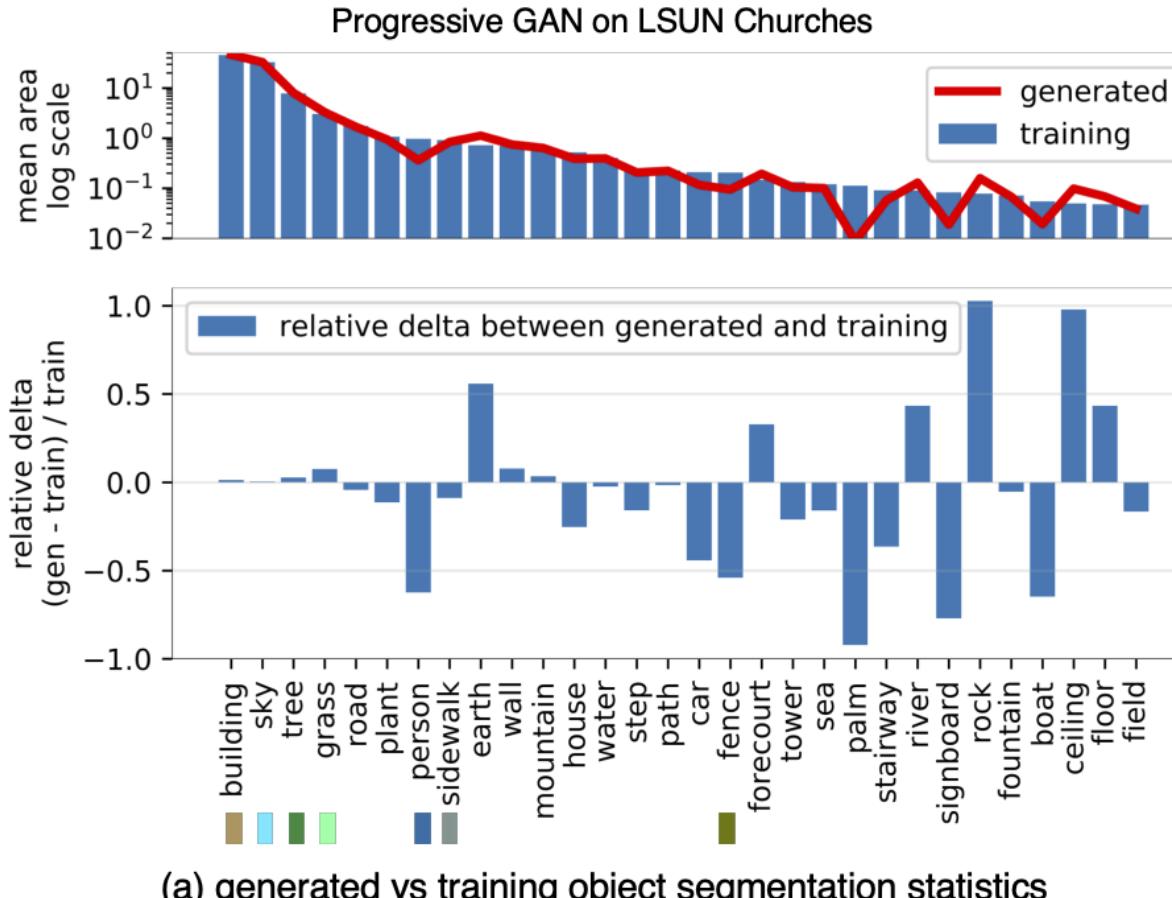
Restyle trees for spring

Restyle trees for autumn

<https://ganpaint.io/demo/?project=church>

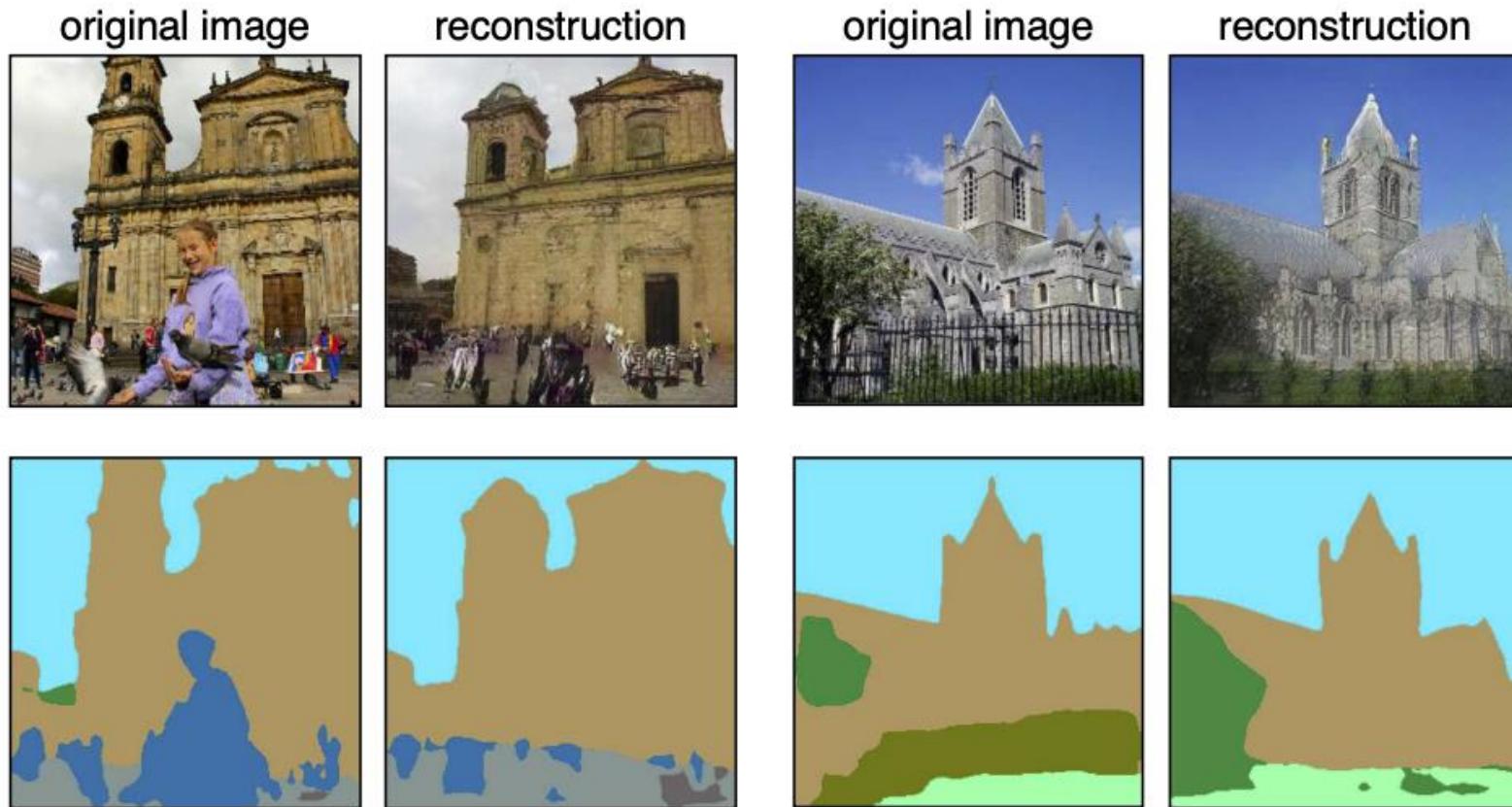
Seeing what a GAN cannot generate

- Compare semantic segmentations of real and generated images, see which classes get dropped by the GAN



Seeing what a GAN cannot generate

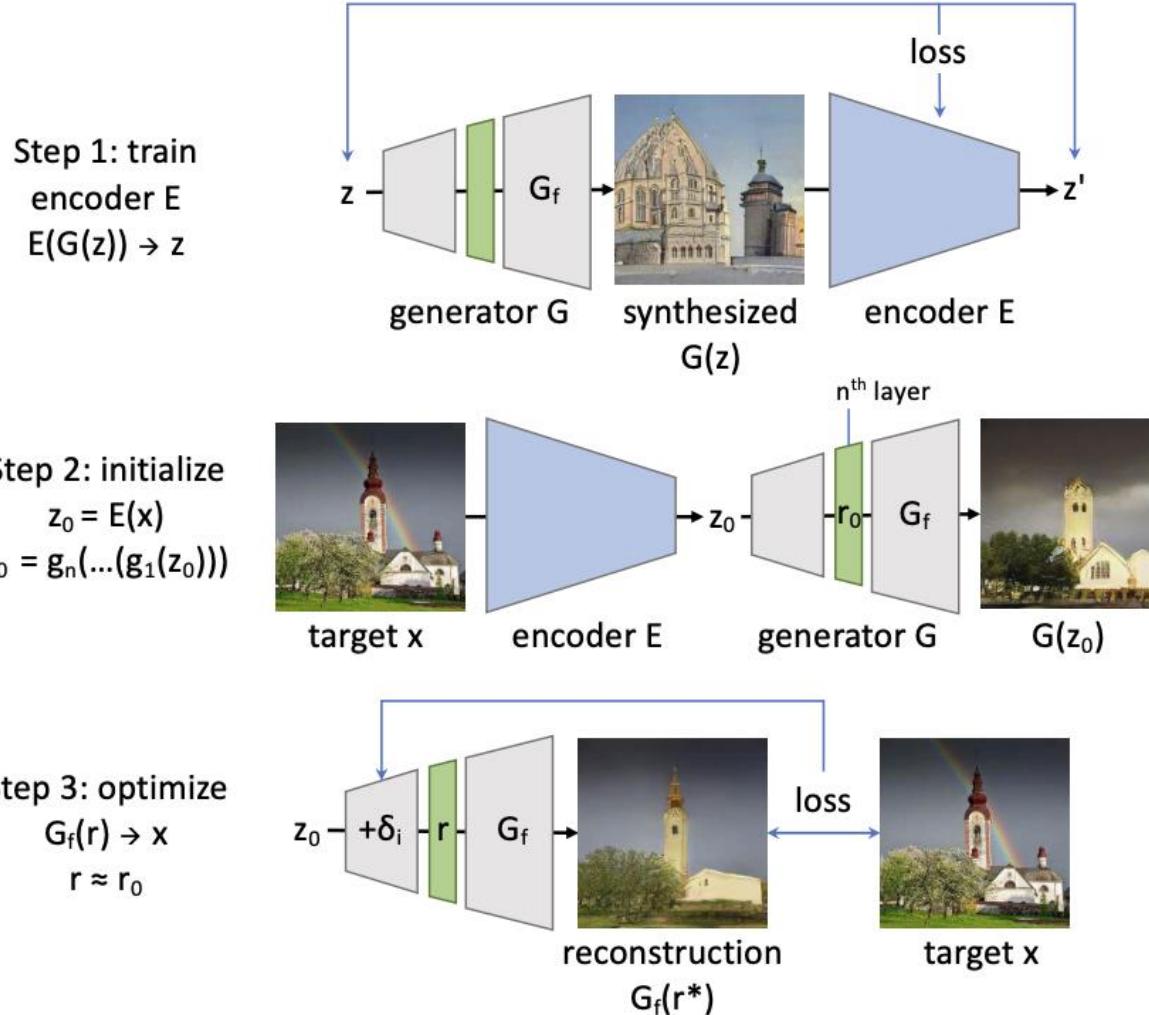
- Given real images, try to find “closest” generated images and see what gets omitted



(b) real images vs. reconstructions

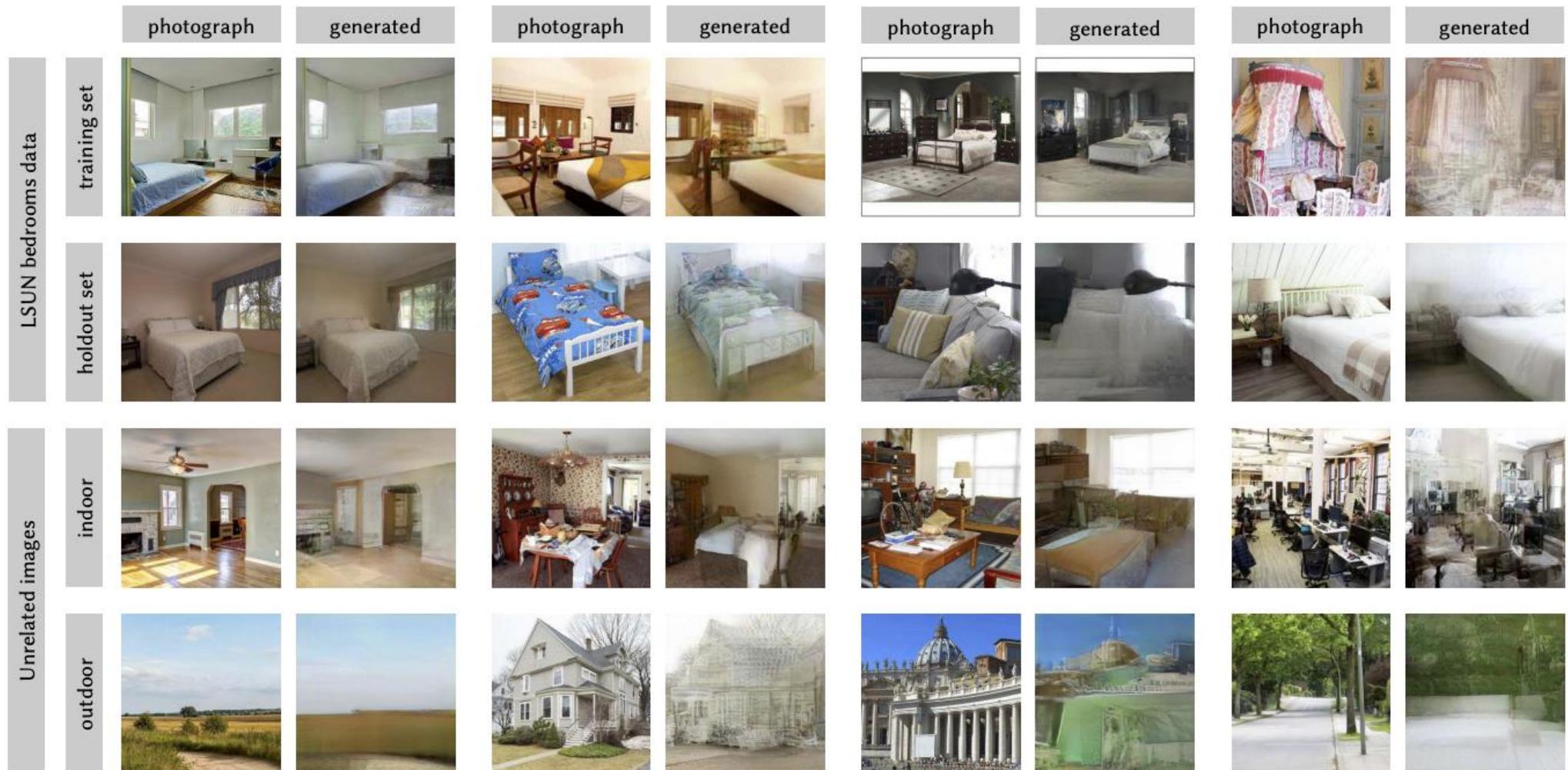
Seeing what a GAN cannot generate

- Layer inversion method:



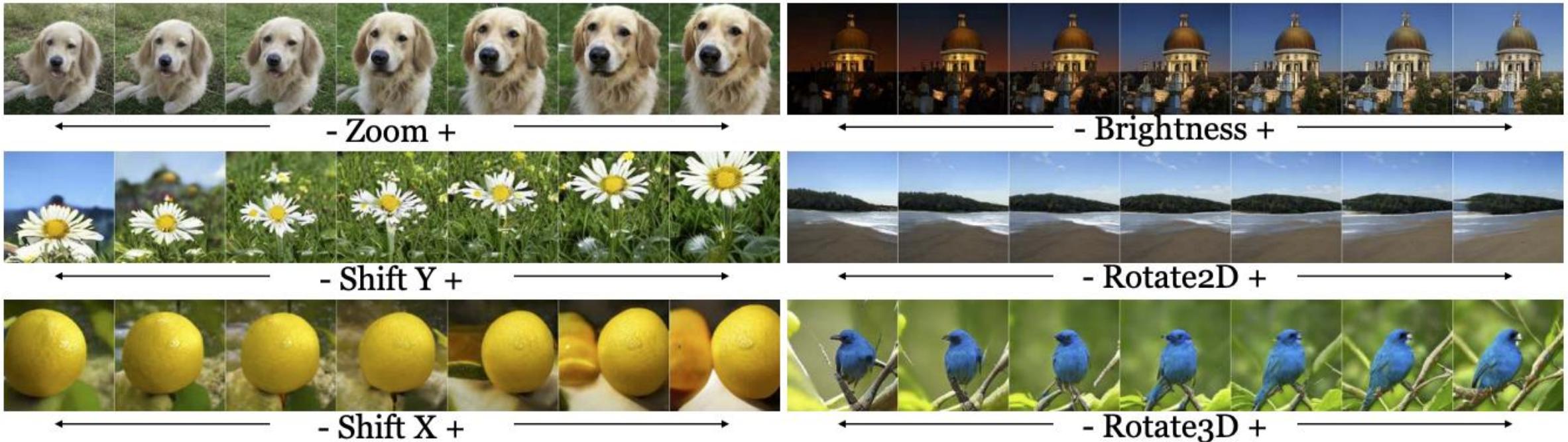
Seeing what a GAN cannot generate

- Reconstruction results for “bedroom” GAN



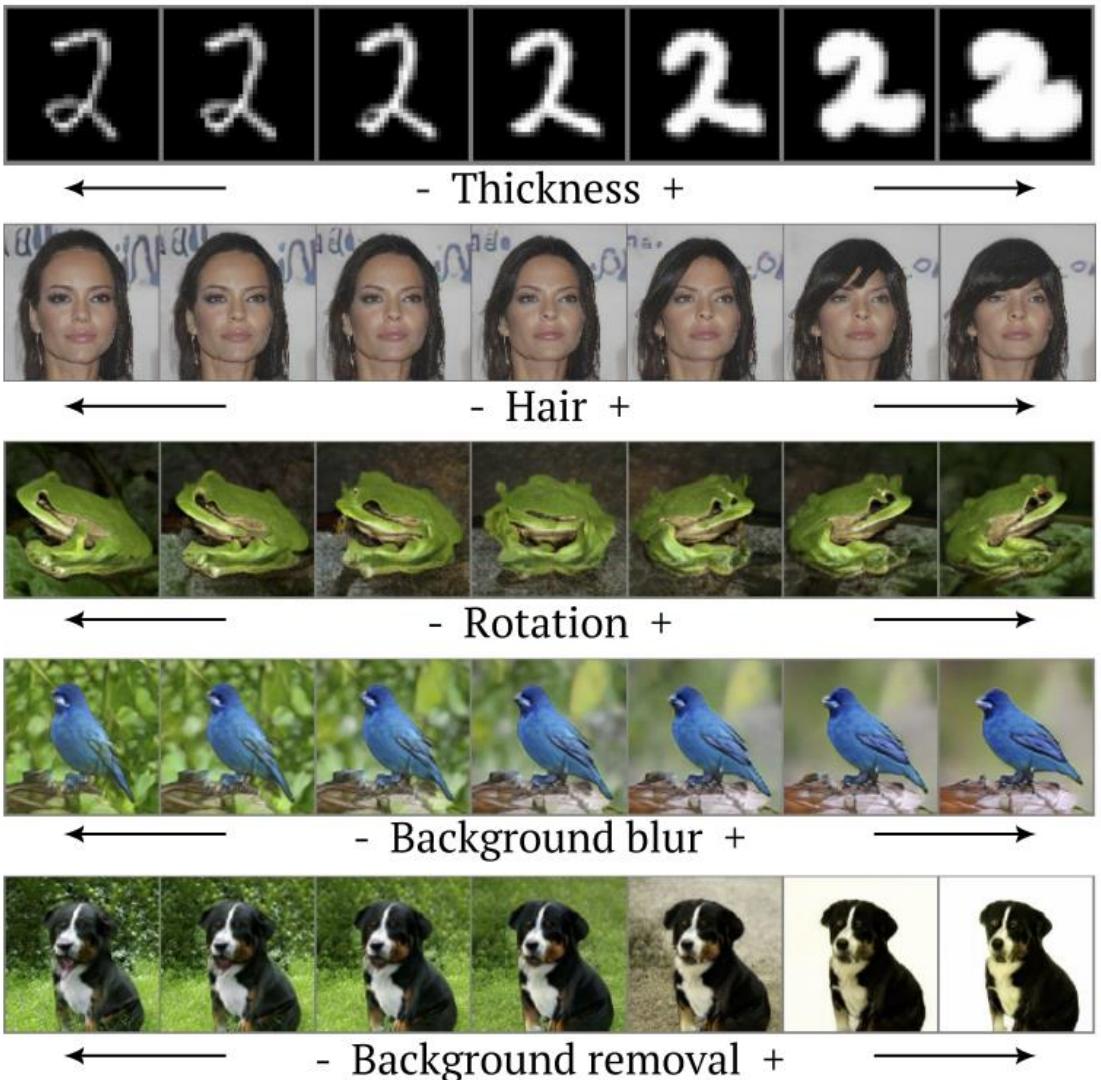
Traversing the GAN latent space

- Try to find simple “walks” in the latent space of GANs to achieve various meaningful transformations to explore structure of that space and test GANs’ ability to interpolate between training samples



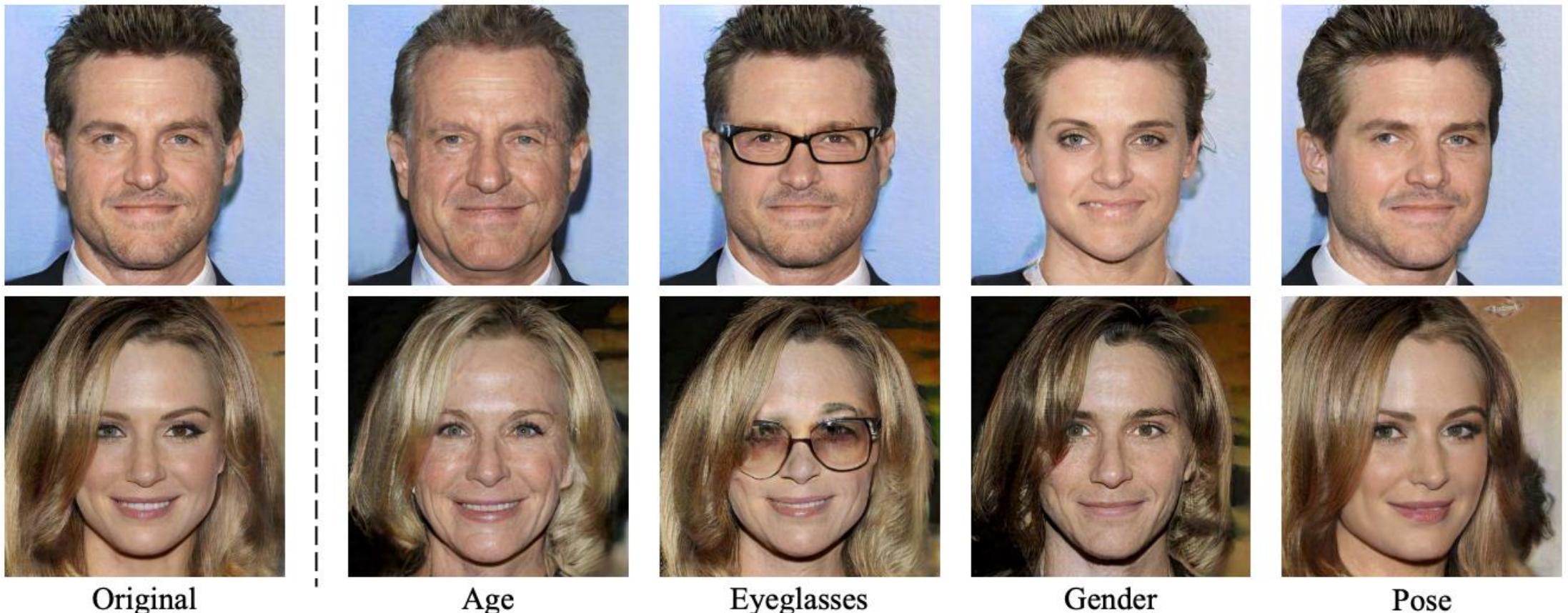
Traversing the GAN latent space

- Goal: learn a set of directions inducing “disentangled” image transformations that are easy to distinguish from each other



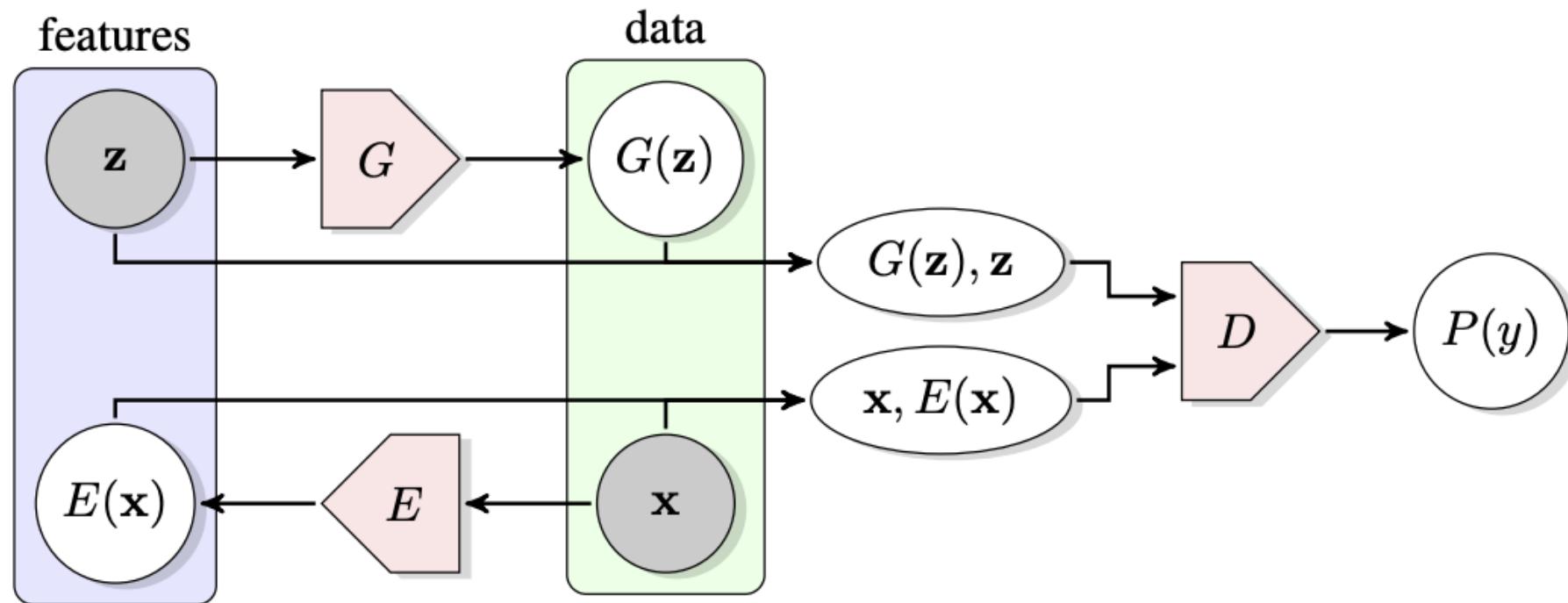
GAN editing

- Supervised training to find latent space directions corresponding to pose, smile, age, gender, eyeglasses



GANs for representation learning

- Bidirectional GAN (BiGAN): simultaneously train generator and encoder (mapping from images to z vectors or approximate inverse of the generator), show that the encoder creates a latent representation useful for other tasks



GANs for representation learning: BigBiGAN

- Train BiGAN with BigGAN architecture on ImageNet
- Show that bidirectional framework improves image generation
- Encoder representation gives results comparable to state-of-the-art self-supervised models on ImageNet classification

GANs for representation learning: BigBiGAN

Real images x



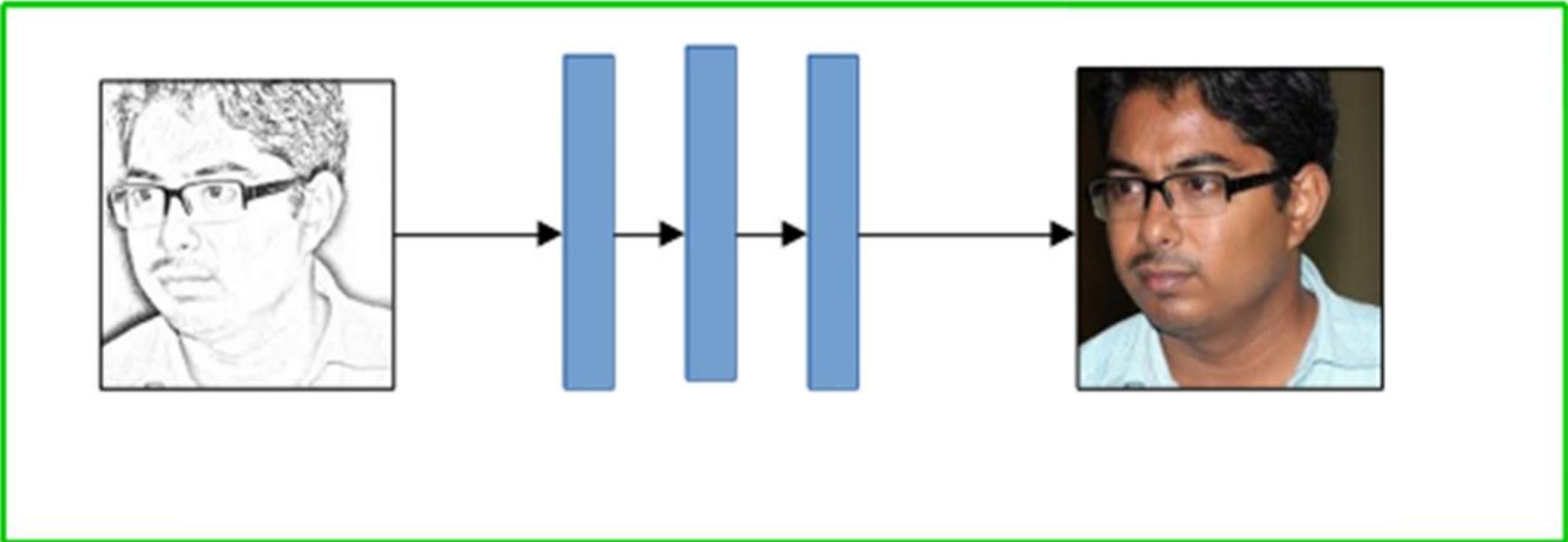
Reconstructions $G(E(x))$

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Conditional GANs

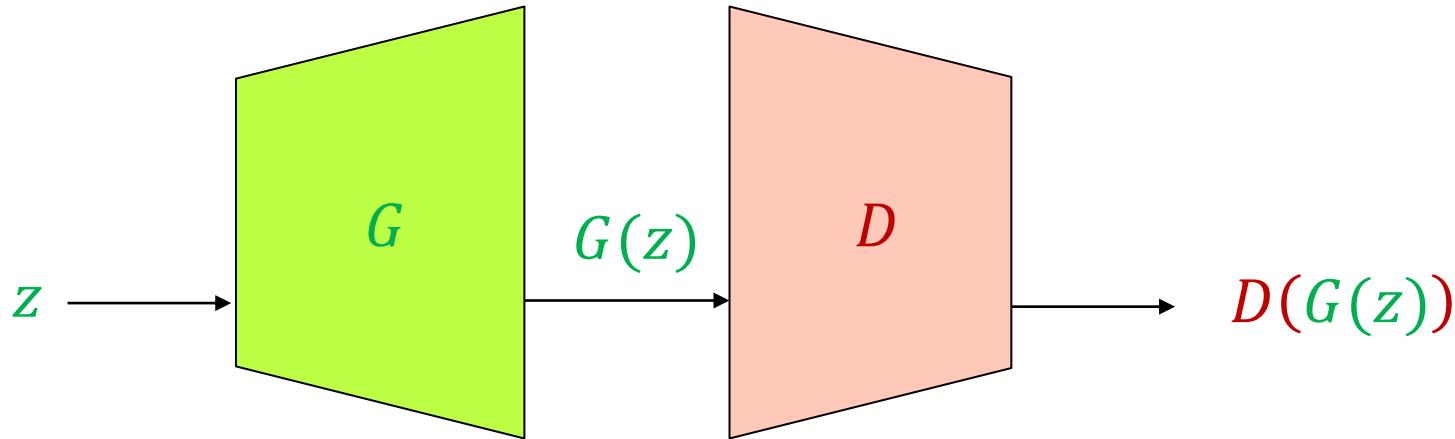


Outline

- Introduction
- Generation conditioned on class
 - Self-attention GAN
 - BigGAN
- Generation conditioned on image
 - Paired image-to-image translation: pix2pix
 - Unpaired image-to-image translation: CycleGAN
- Recent trends

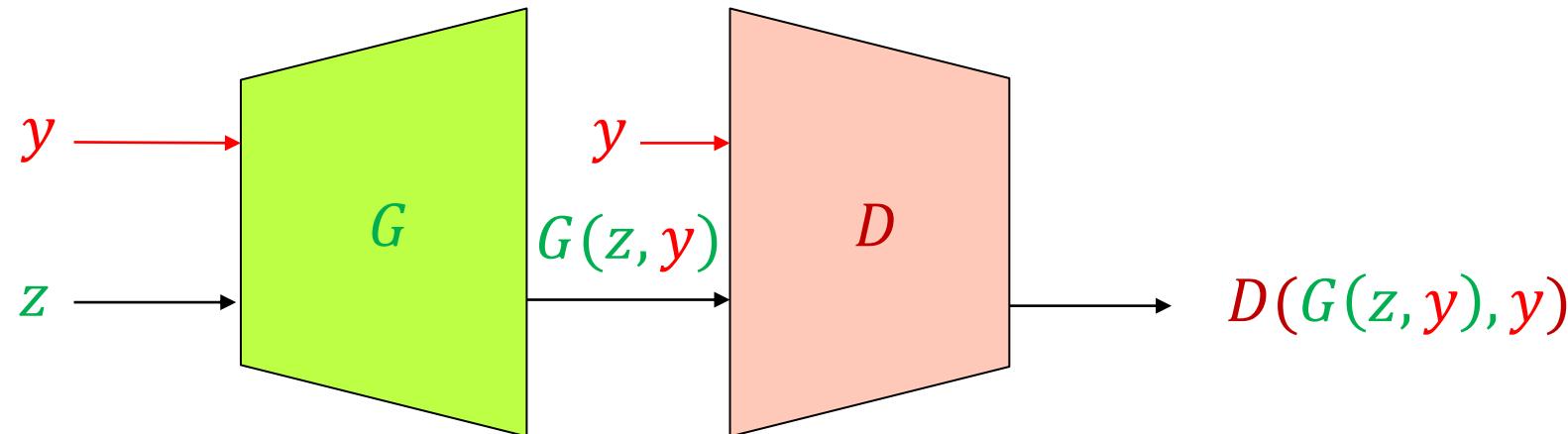
Conditional generation

- Suppose we want to condition the generation of samples on discrete side information (label) y
 - How do we add y to the basic GAN framework?



Conditional generation

- Suppose we want to condition the generation of samples on discrete side information (label) y
 - How do we add y to the basic GAN framework?



Conditional generation

- Example: simple network for generating 28 x 28 MNIST digits

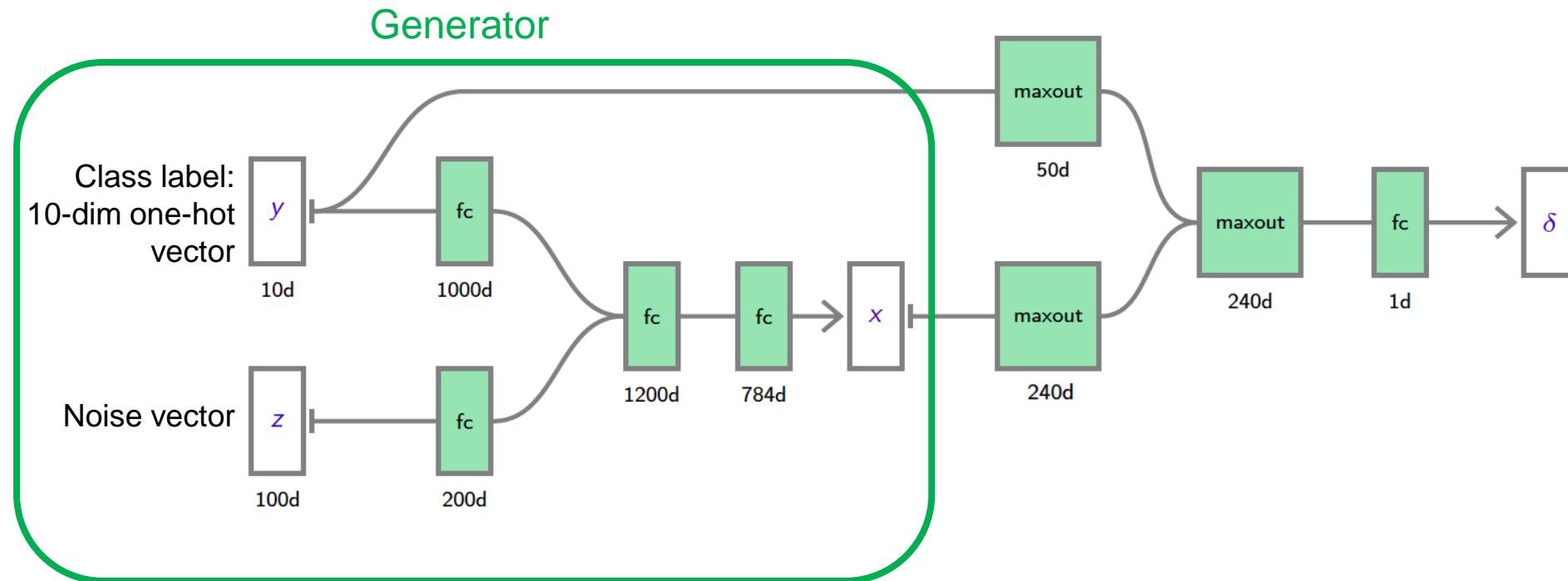


Figure source:
[F. Fleuret](#)

Conditional generation

- Example: simple network for generating 28 x 28 MNIST digits

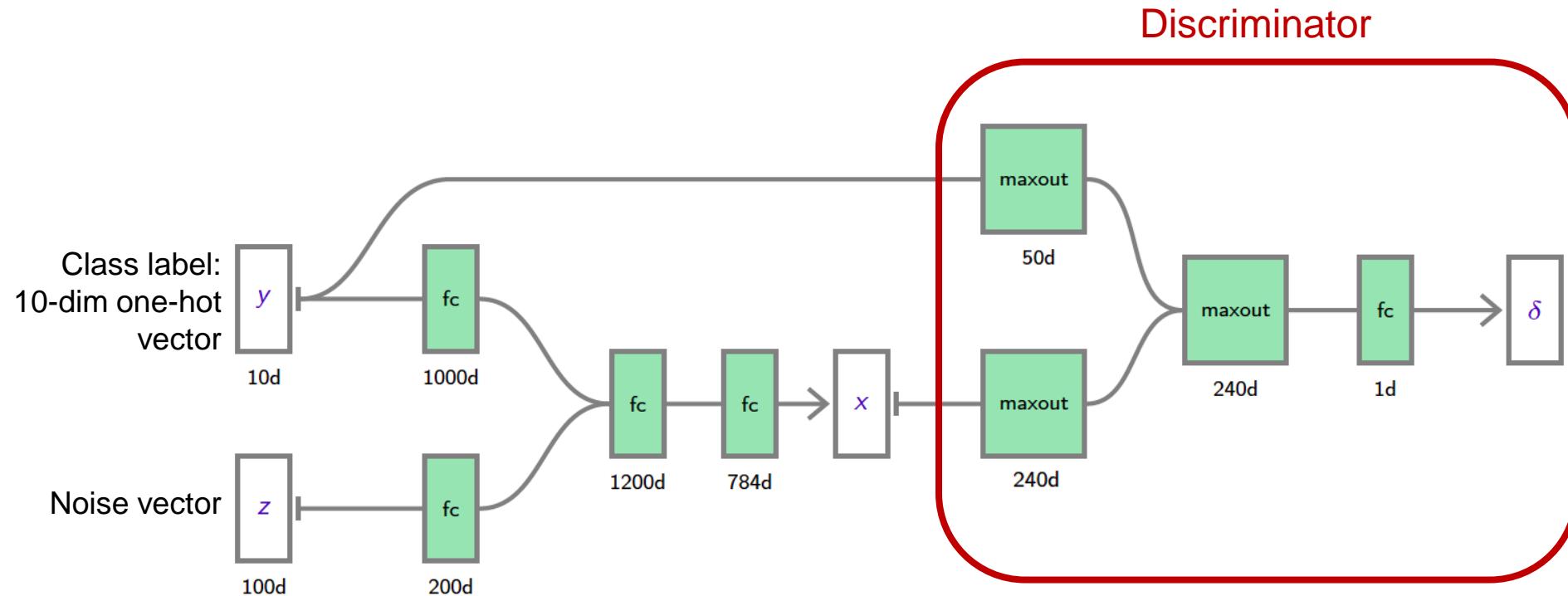
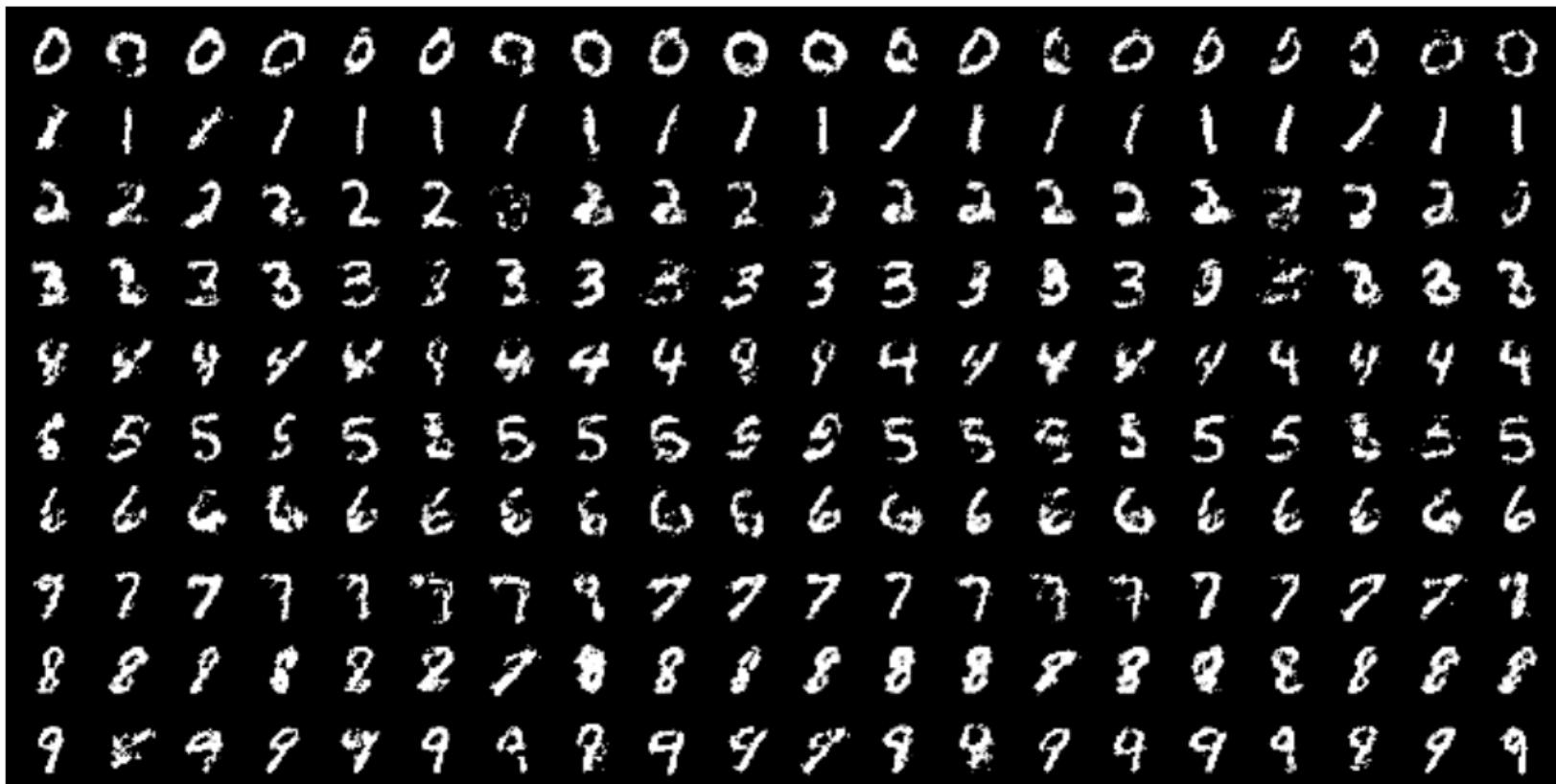


Figure source:
[F. Fleuret](#)

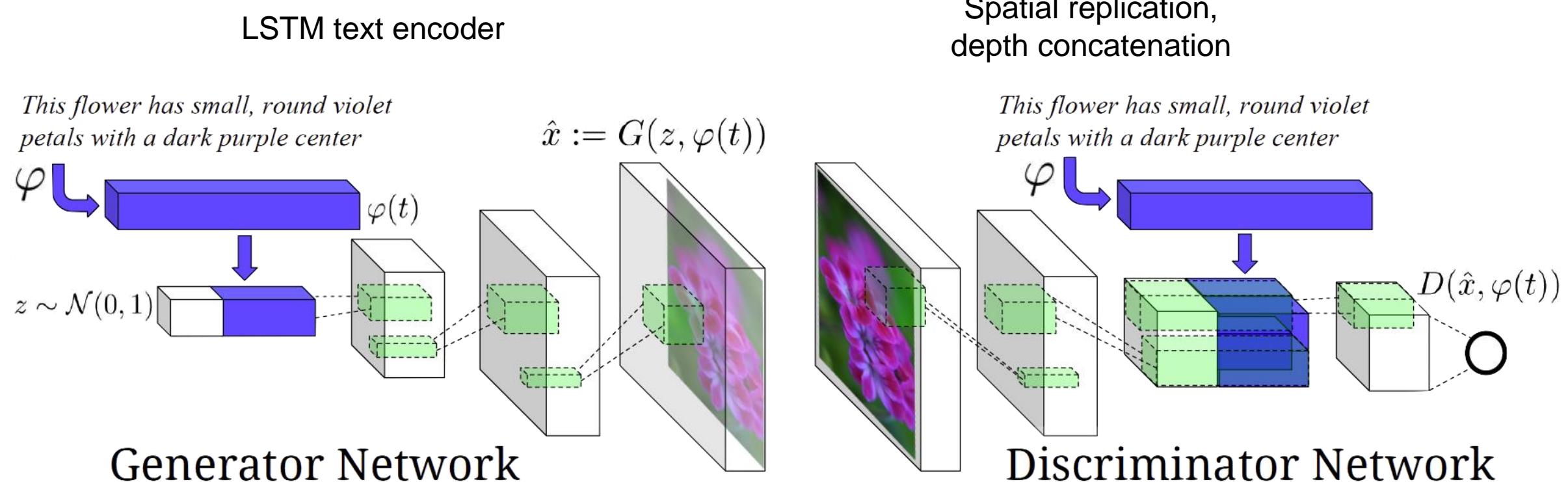
Conditional generation

- Example: simple network for generating 28 x 28 MNIST digits



Conditional generation

- Another example: text-to-image synthesis



Conditional generation

- Another example: text-to-image synthesis

Previously unseen
captions (zero-shot
setting)

this small bird has a pink
breast and crown, and black
primaries and secondaries.



the flower has petals that
are bright pinkish purple
with white stigma



this magnificent fellow is
almost all black with a red
crest, and white cheek patch.



this white and yellow flower
have thin white petals and a
round yellow stamen



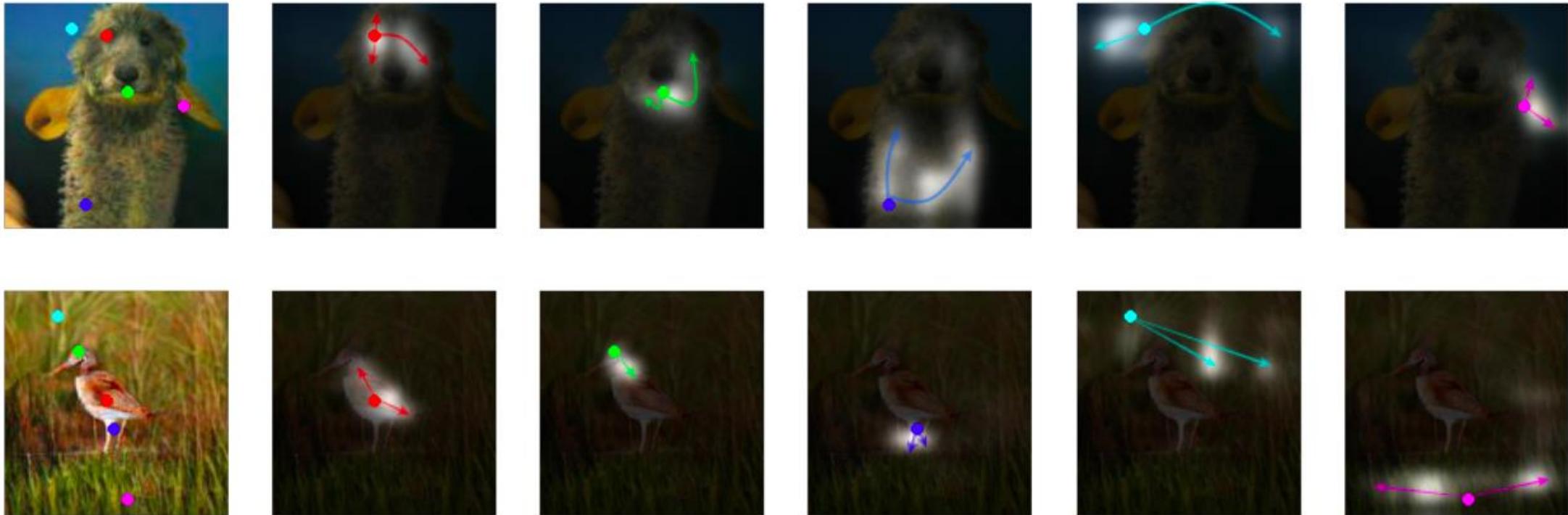
Captions seen in
the training set

Outline

- Introduction
- Generation conditioned on class
 - Self-attention GAN
 - BigGAN

Self-attention GAN (SAGAN)

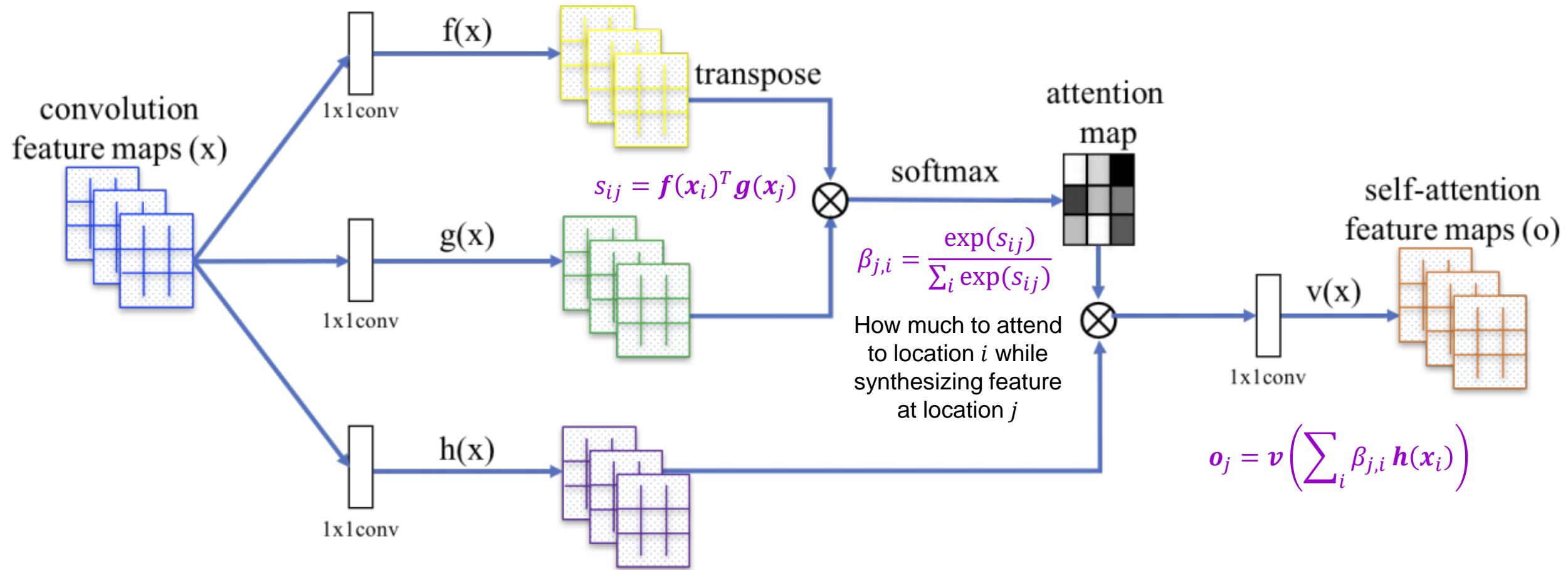
- Adaptive receptive fields to capture non-local structure



SAGAN generates images by leveraging complementary features in distant portions of the image rather than local regions of fixed shape to generate consistent objects/scenarios. In each row, the first image shows five representative query locations with color coded dots. The other five images are attention maps for those query locations, with corresponding color coded arrows summarizing the most-attended regions.

Self-attention GAN

- Adaptive receptive fields to capture non-local structure
(based on [Wang et al.](#), 2018)



Self-attention GAN: Implementation details

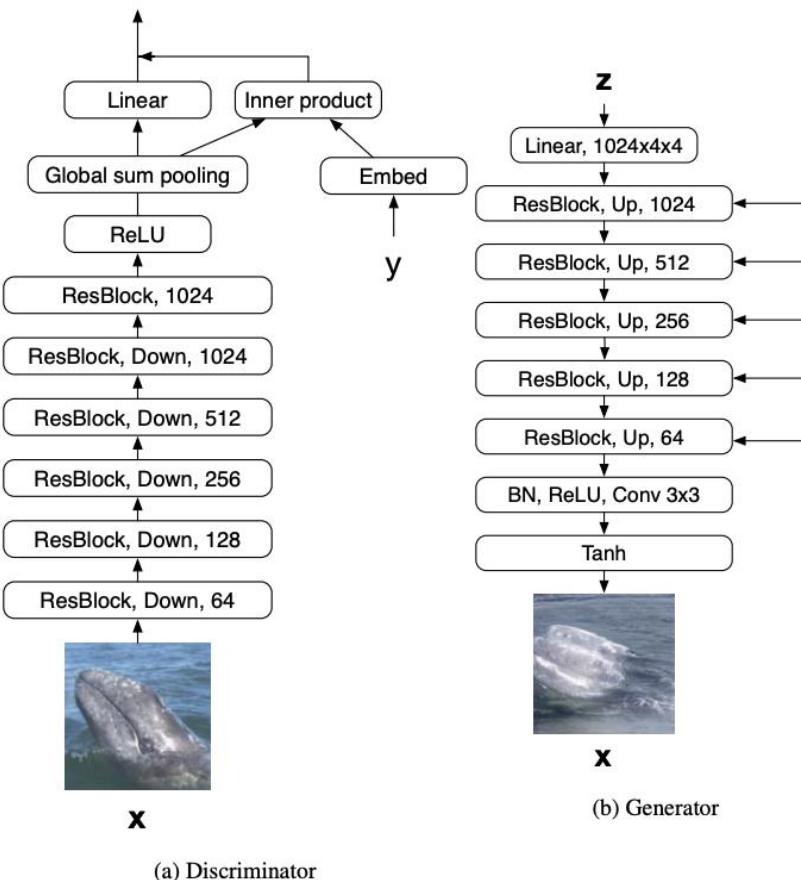
- Hinge loss formulation:

$$L_D = -\mathbb{E}_{(x,y) \sim p_{\text{data}}} [\min(0, D(x, y) - 1)] \\ -\mathbb{E}_{z \sim p_z, y \sim p_{\text{data}}} [\min(0, -D(G(z, y), y) - 1)]$$

$$L_G = -\mathbb{E}_{z \sim p_z, y \sim p_{\text{data}}} D(G(z, y), y)$$

Self-attention GAN: Implementation details

- Hinge loss formulation
- Conditioning the discriminator: *projection* ([Miyato & Koyama, 2018](#))
- Conditioning the generator: *conditional batch norm*



Self-attention GAN: Implementation details

- Hinge loss formulation
- Conditioning the discriminator: *projection* ([Miyato & Koyama, 2018](#))
- Conditioning the generator: *conditional batch norm*
- *Spectral normalization* for generator and discriminator ([Miyato et al., 2018](#)) – divide weight matrices by largest singular value (estimated)
- Different learning rates for generator and discriminator (TTUR – [Heusel et al., 2017](#))

Self-attention GAN: Results

- 128 x 128 ImageNet

goldfish



indigo
bunting



redshank

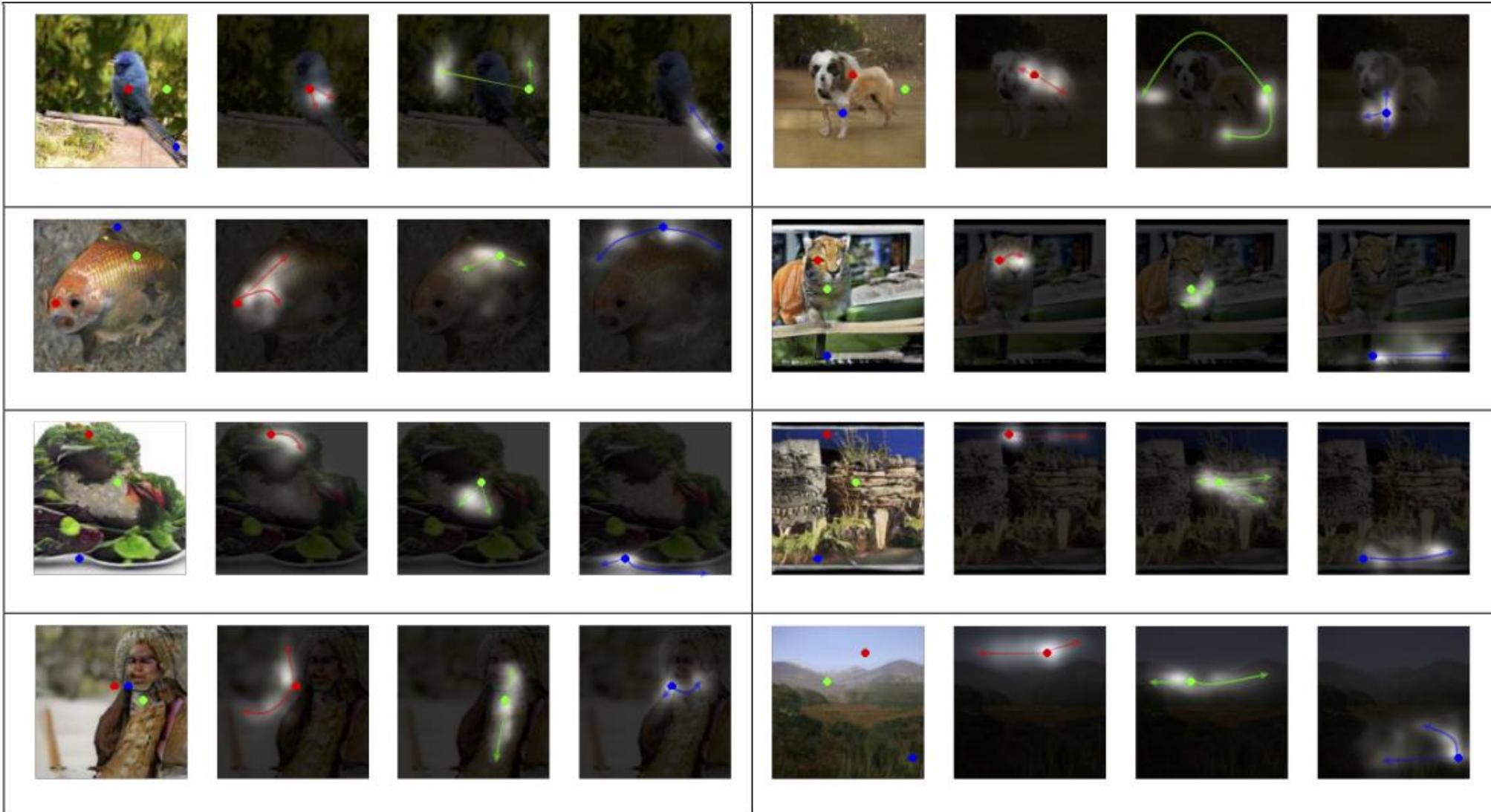


Saint
Bernard



Self-attention GAN: Results

- Attention map visualization



BigGAN

- Scale up SA-GAN to generate ImageNet images up to 512 x 512 resolution



BigGAN: Implementation details

- 8x larger batch size, 50% more channels (2x more parameters) than baseline SA-GAN
- Hierarchical latent space: feed (transformations of) z vector into multiple layers of the generator

BigGAN: Implementation details

- 8x larger batch size, 50% more channels (2x more parameters) than baseline SA-GAN
- Hierarchical latent space: feed (transformations of) z vector into multiple layers of the generator
- Truncation trick: at test time, resample the values of the z vector with magnitude above a chosen threshold
 - Trade off diversity for image quality



“The effects of increasing truncation. From left to right, the threshold is set to 2, 1, 0.5, 0.04.”

BigGAN: Implementation details

- 8x larger batch size, 50% more channels (2x more parameters) than baseline SA-GAN
- Hierarchical latent space: feed (transformations of) z vector into multiple layers of the generator
- Truncation trick: at test time, resample the values of the z vector with magnitude above a chosen threshold
- Lots of other tricks (initialization, training, etc.)
- Training observed to be unstable, but good results are achieved “just before collapse”
- Evidence that discriminator memorizes the training data, but the generator doesn’t

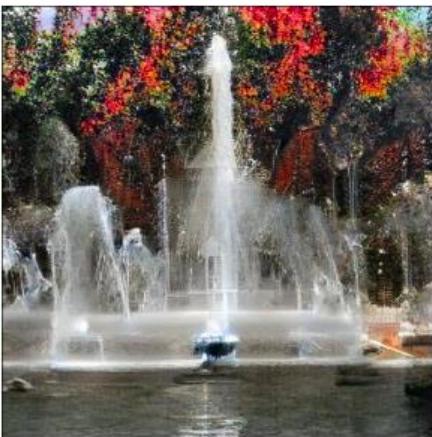
BigGAN: Results

- Samples at 256 x 256 resolution:



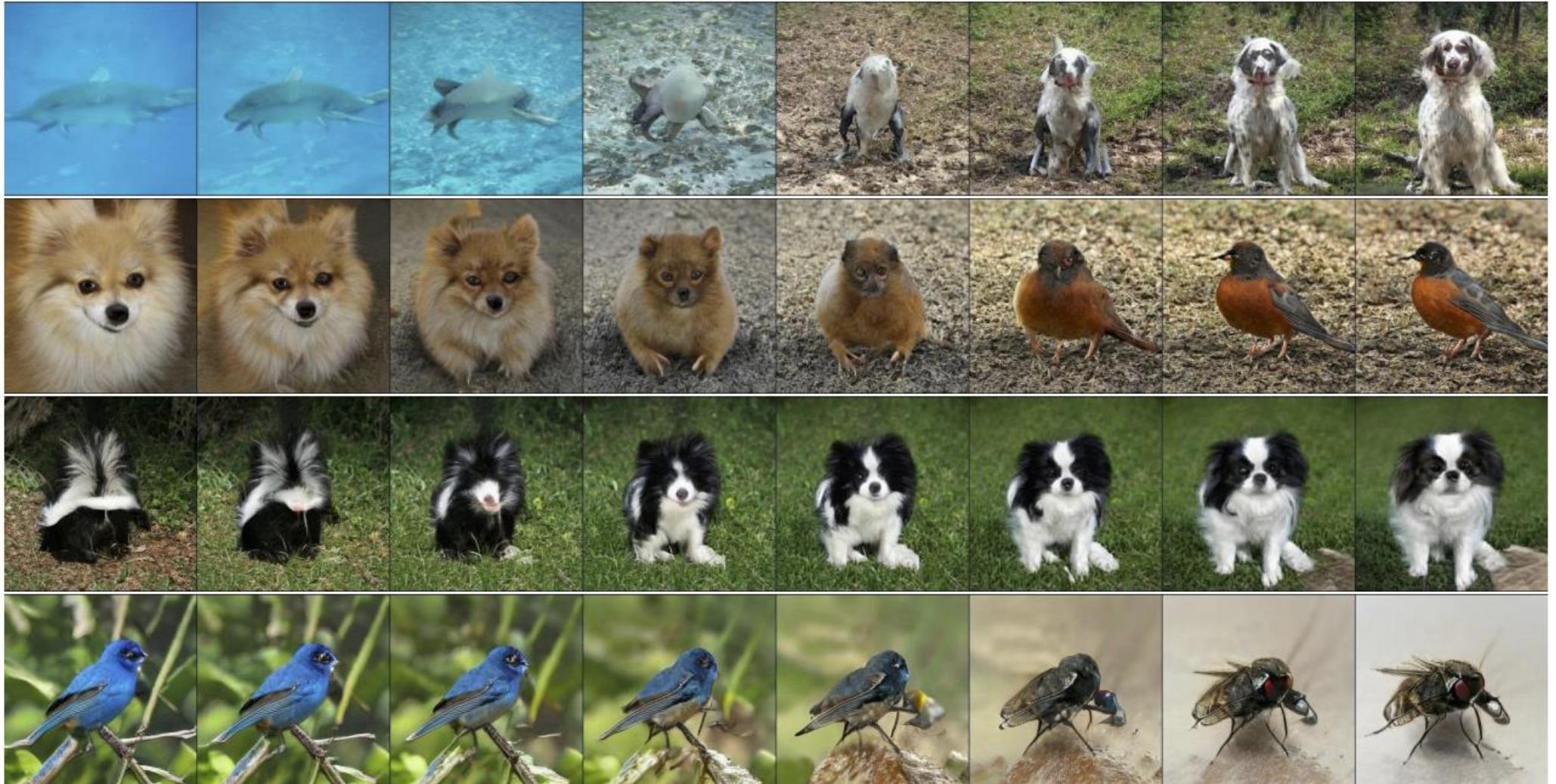
BigGAN: Results

- Samples at 512 x 512 resolution:



BigGAN: Results

- Interpolation between c (class embeddings), z (noise vector)



BigGAN: Results

- Difficult classes:



Conditional GANs: Outline

- Introduction
- Generation conditioned on class
 - Self-attention GAN
 - BigGAN
- **Generation conditioned on image**
 - Paired image-to-image translation: pix2pix
 - Unpaired image-to-image translation: CycleGAN

Image-to-image translation

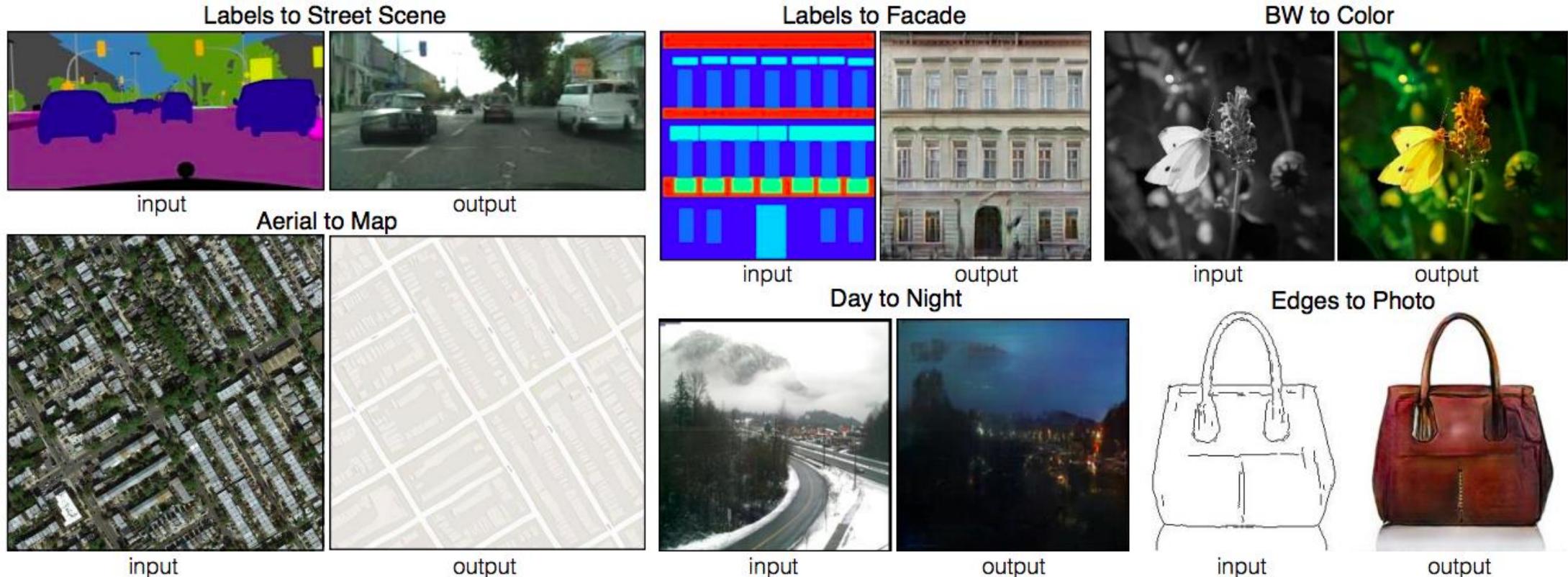


Image-to-image translation

- Produce modified image y conditioned on input image x
Generator receives x as input
 - Discriminator receives an x, y pair and has to decide whether it is real or fake

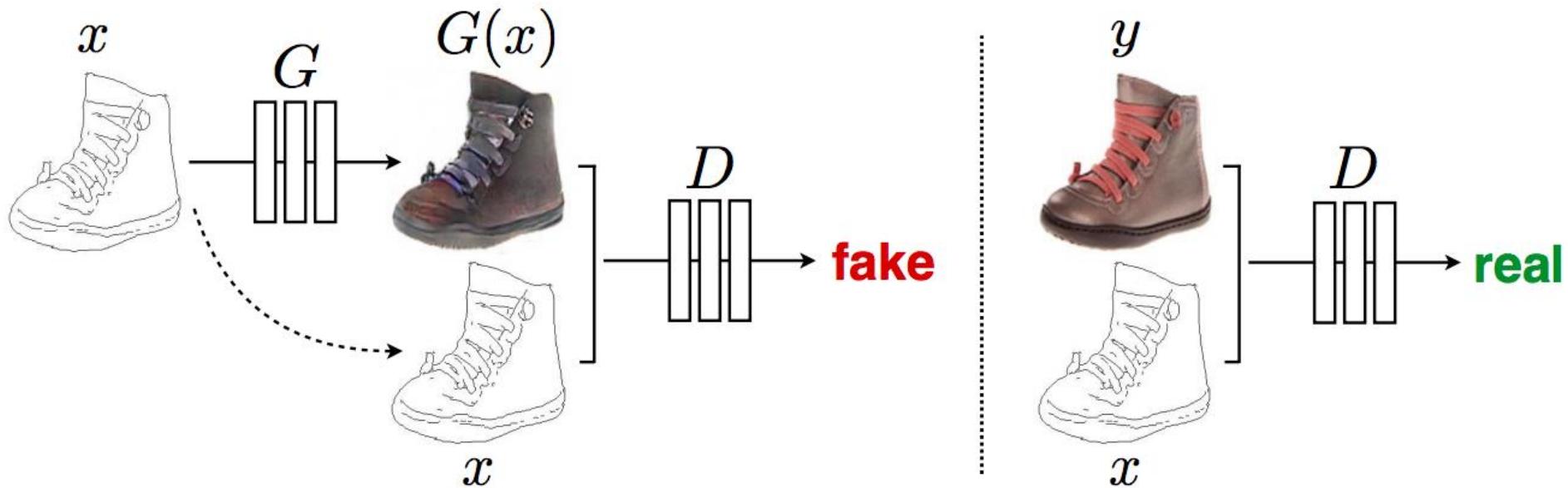
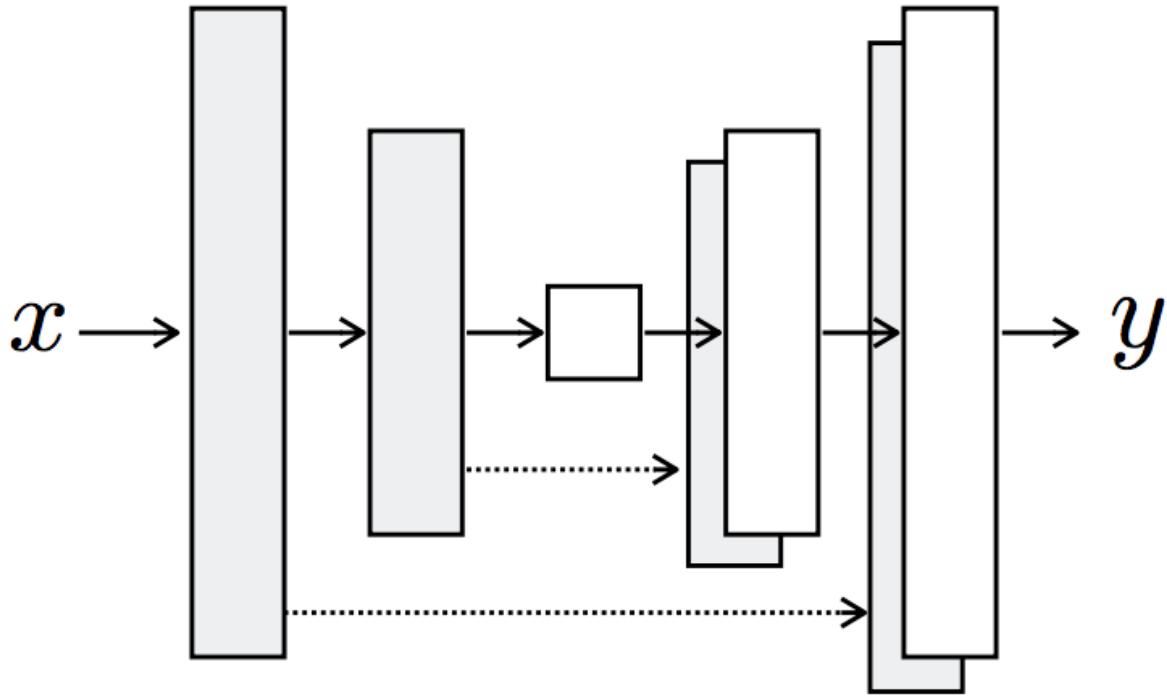


Image-to-image translation

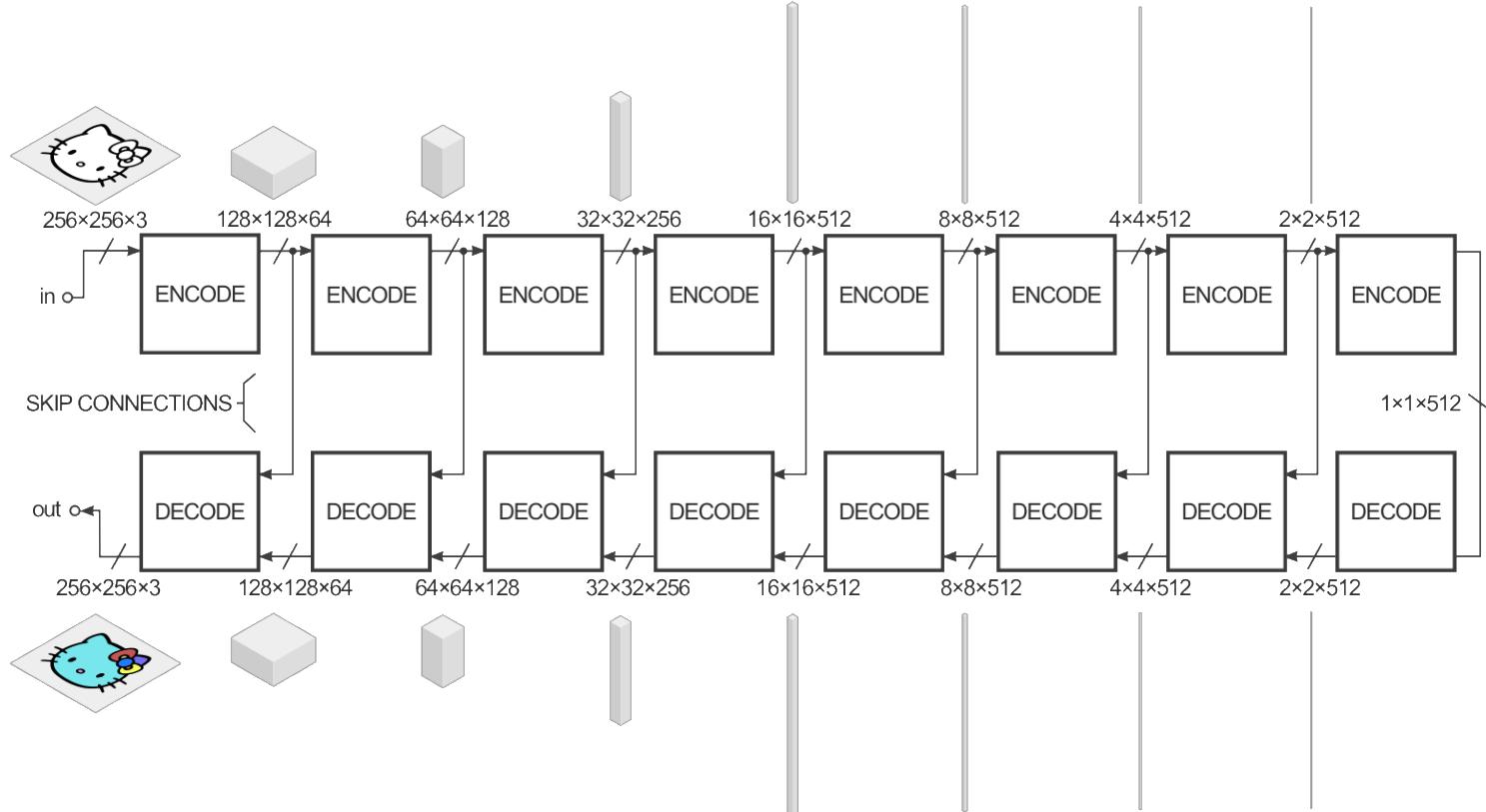
- Generator architecture: U-Net



- Note: z is not used as input, transformation is basically deterministic

Image-to-image translation

- Generator architecture: U-Net



Encode: convolution → BatchNorm → ReLU

Decode: transposed convolution → BatchNorm → ReLU

[Figure source](#)

Image-to-image translation

- Generator architecture: U-Net

Effect of adding skip connections to the generator



Image-to-image translation

- Generator loss: GAN loss plus L1 reconstruction penalty

$$G^* = \arg \min_G \max_D \mathcal{L}_{GAN}(G, D) + \lambda \sum_i \|y_i - G(x_i)\|_1$$

Generated output
 $G(x_i)$ should be close to
ground truth target y_i

Image-to-image translation

- Generator loss: GAN loss plus L1 reconstruction penalty

$$G^* = \arg \min_G \max_D \mathcal{L}_{GAN}(G, D) + \lambda \sum_i \|y_i - G(x_i)\|_1$$



Image-to-image translation

- Discriminator: PatchGAN
 - Given input image x and second image y , decide whether y is a ground truth target or produced by the generator

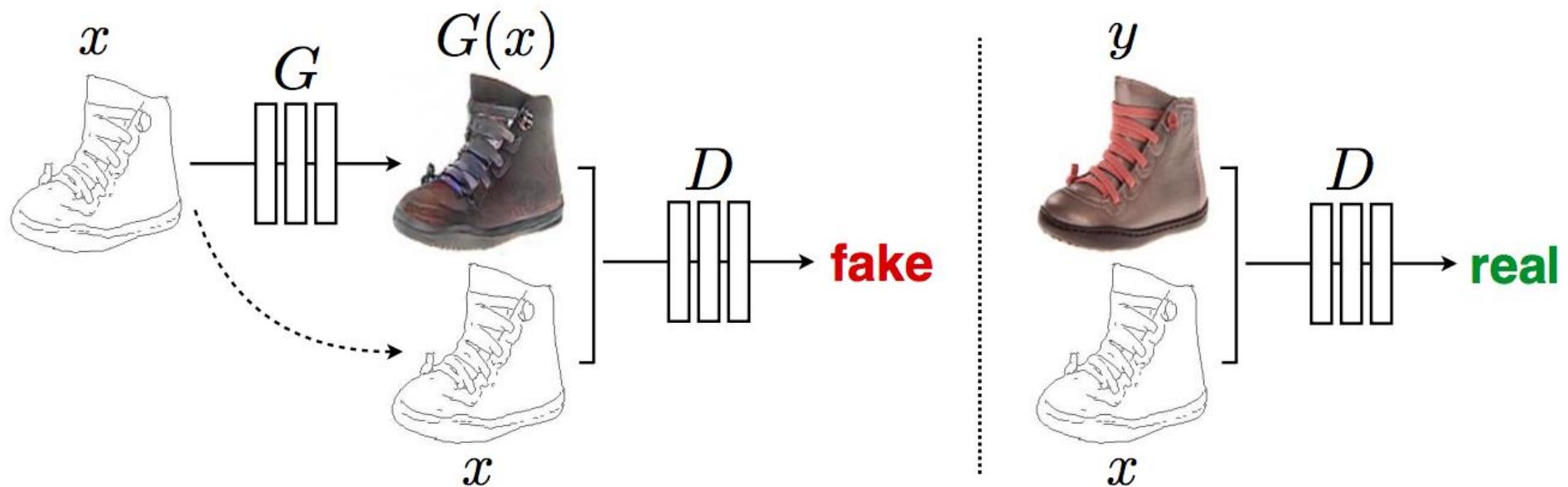
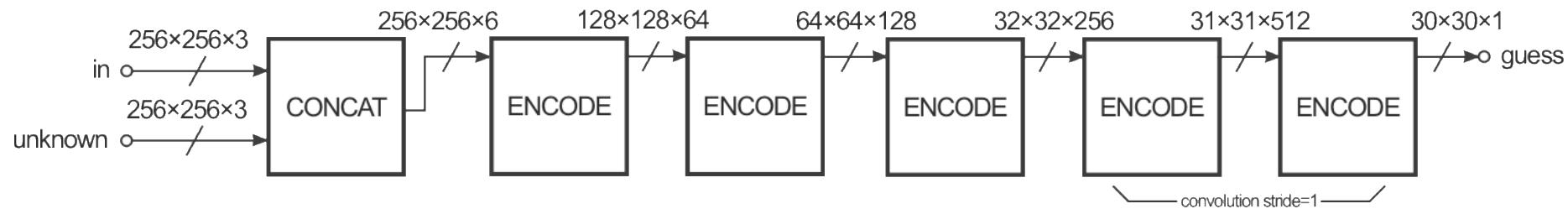


Image-to-image translation

- Discriminator: PatchGAN
 - Given input image x and second image y , decide whether y is a ground truth target or produced by the generator
 - Output is a 30×30 map where each value (0 to 1) represents the quality of the corresponding section of the output image, these values are averaged to obtain final discriminator loss
 - Fully convolutional network, effective patch size can be increased by increasing the depth



[Figure source](#)

Image-to-image translation

- Discriminator: PatchGAN
 - Given input image x and second image y , decide whether y is a ground truth target or produced by the generator
 - Output is a 30×30 map where each value (0 to 1) represents the quality of the corresponding section of the output image, these values are averaged to obtain final discriminator loss
 - Fully convolutional network, effective patch size can be increased by increasing the depth

Effect of discriminator patch size on generator output

L1

1×1

16×16

70×70

286×286



Image-to-image translation: Results

- Translating between maps and aerial photos



Image-to-image translation: Results

- Translating between maps and aerial photos
- Human study:

Loss	Photo → Map		Map → Photo
	% Turkers labeled <i>real</i>	% Turkers labeled <i>real</i>	% Turkers labeled <i>real</i>
L1	2.8% ± 1.0%		0.8% ± 0.3%
L1+cGAN	6.1% ± 1.3%		18.9% ± 2.5%

Image-to-image translation: Results

- Semantic labels to scenes

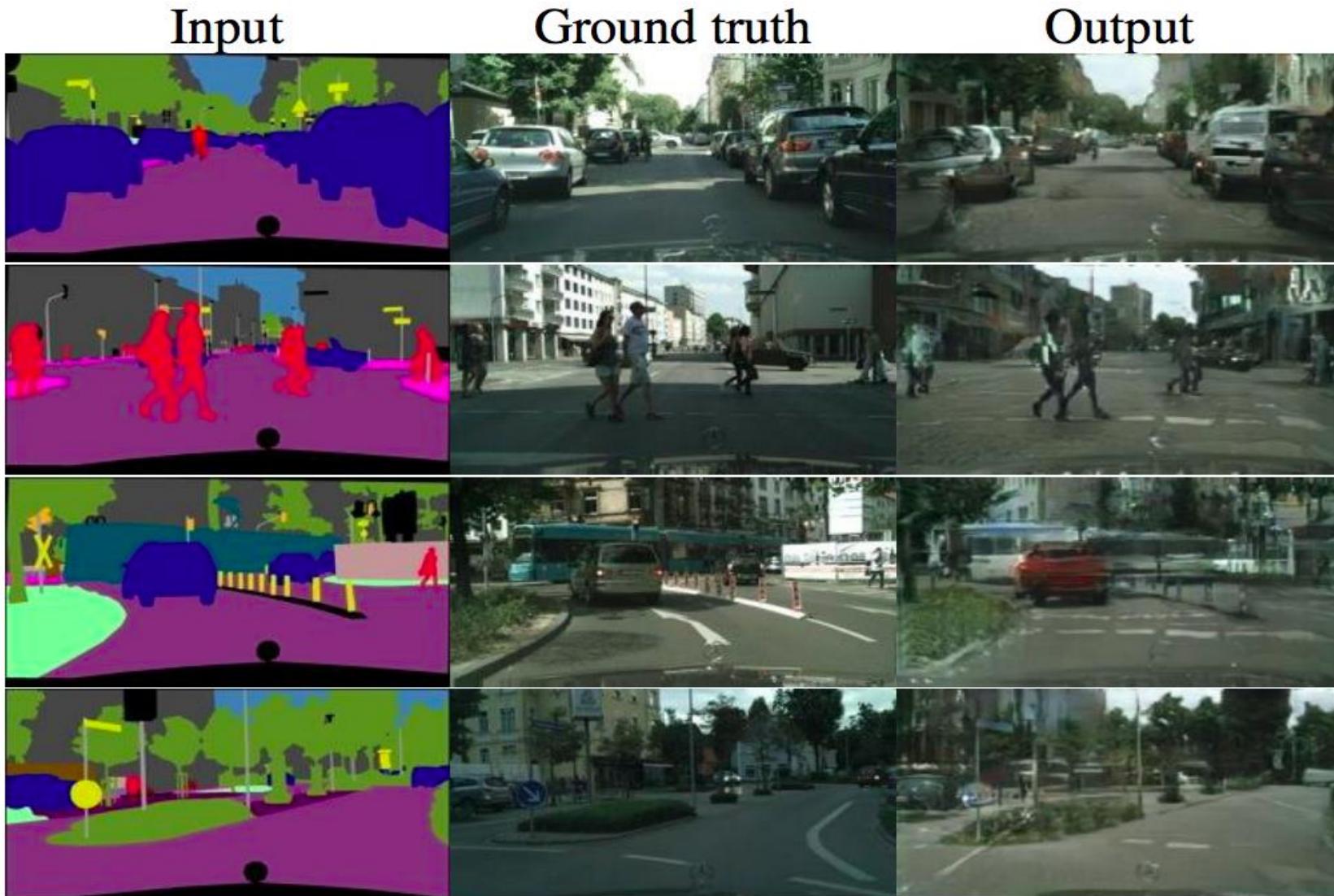


Image-to-image translation: Results

- Semantic labels to scenes
- Evaluation: FCN score
 - The higher the quality of the output, the better the FCN should do at recovering the original semantic labels

FCN — Fully Convolutional Network

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.42	0.15	0.11
GAN	0.22	0.05	0.01
cGAN	0.57	0.22	0.16
L1+GAN	0.64	0.20	0.15
L1+cGAN	0.66	0.23	0.17
Ground truth	0.80	0.26	0.21

Image-to-image translation: Results

- Scenes to semantic labels

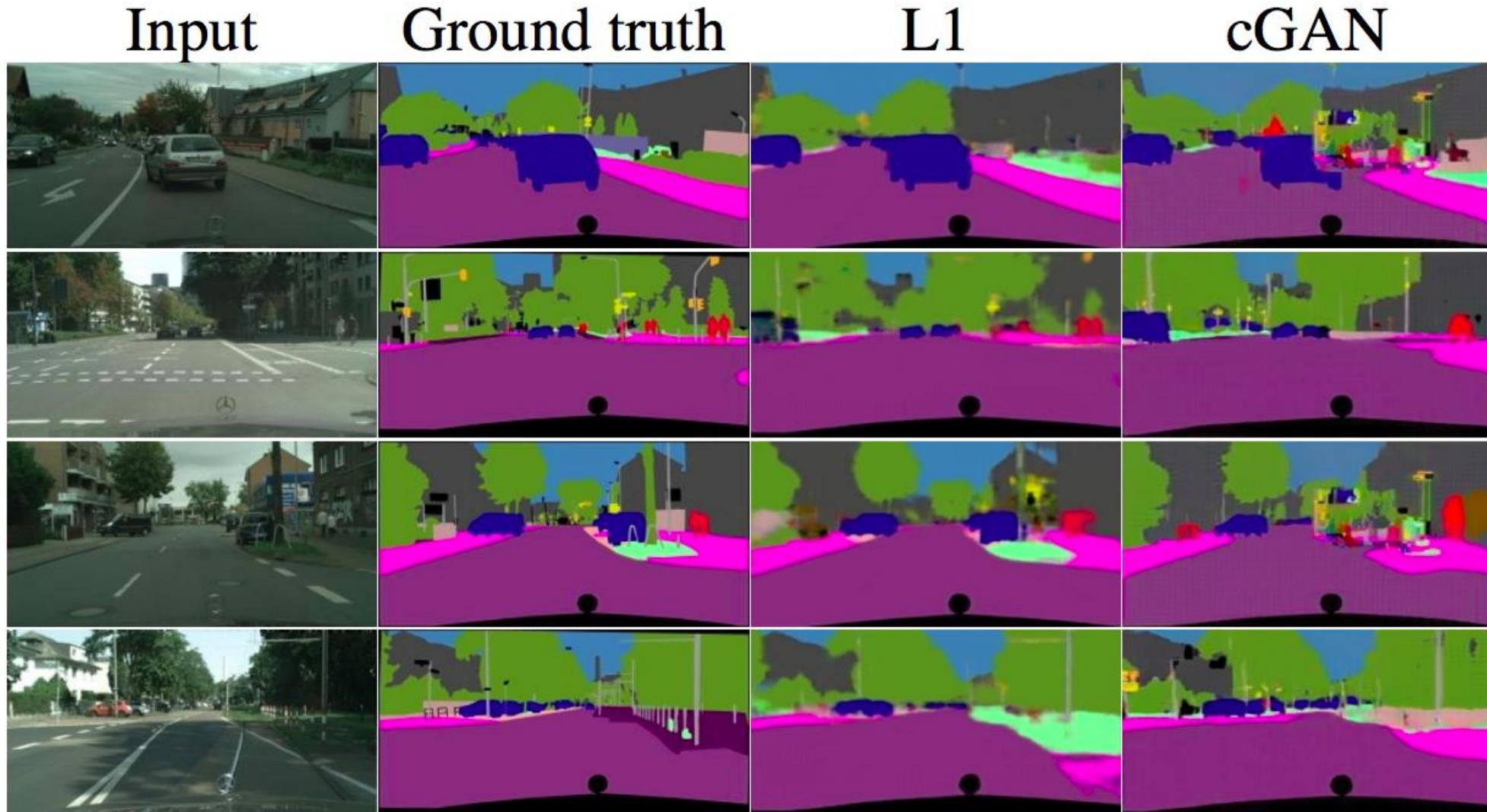


Image-to-image translation: Results

- Scenes to semantic labels
- Accuracy is worse than that of regular FCNs or generator with L1 loss

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.86	0.42	0.35
cGAN	0.74	0.28	0.22
L1+cGAN	0.83	0.36	0.29

Image-to-image translation: Results

- Semantic labels to facades

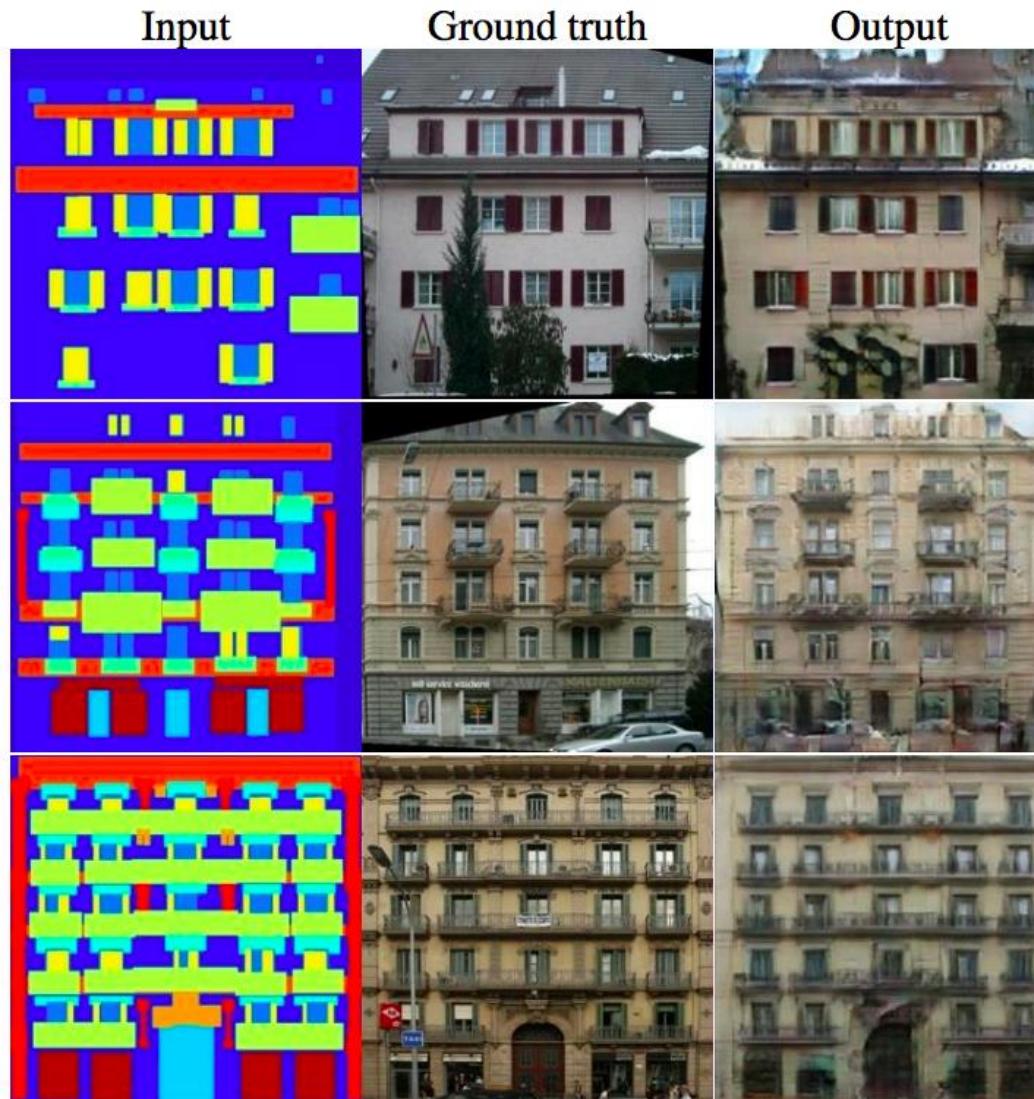


Image-to-image translation: Results

- Day to night

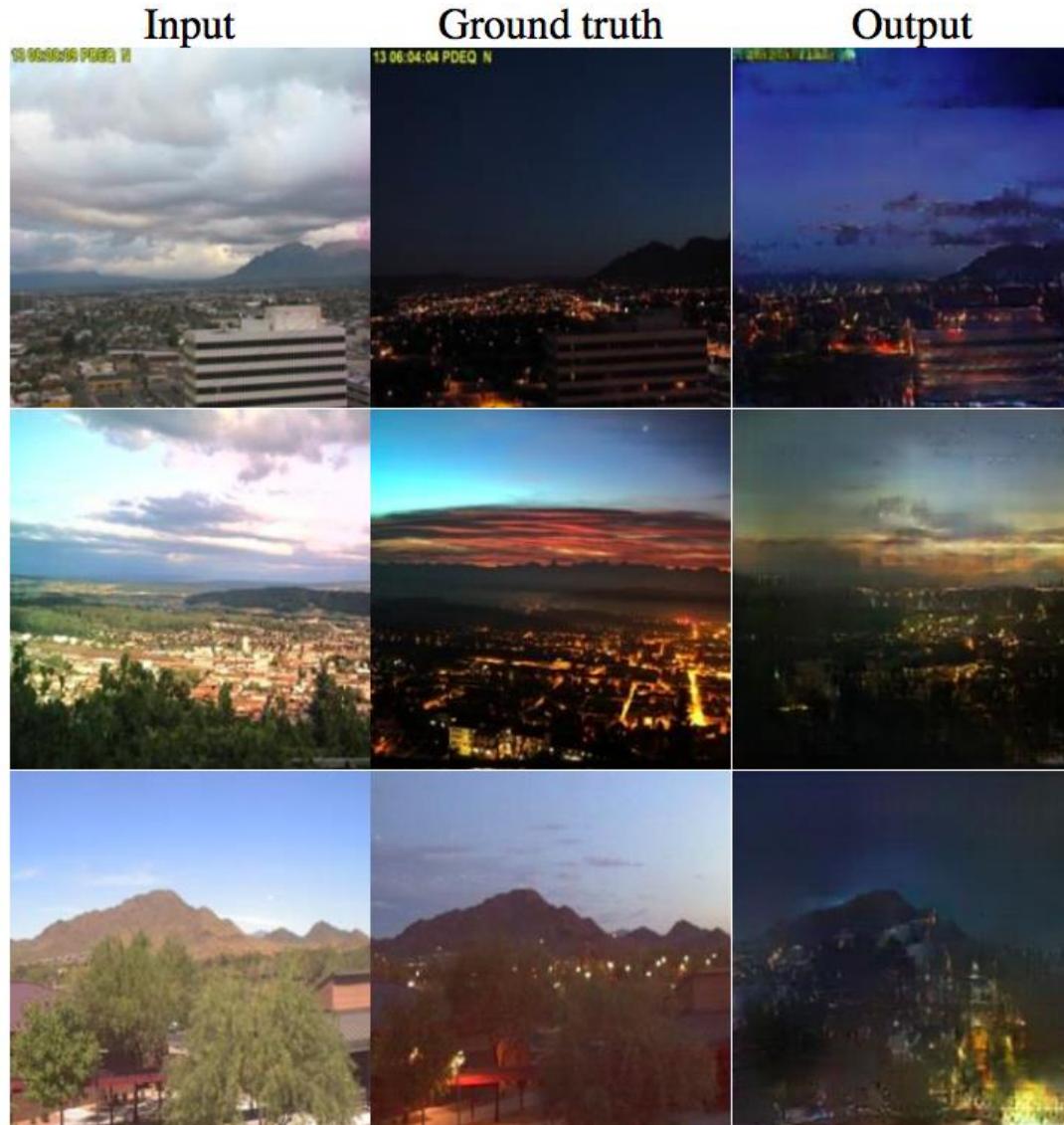


Image-to-image translation: Results

- Edges to photos



Image-to-image translation: Results

- [pix2pix demo](#)

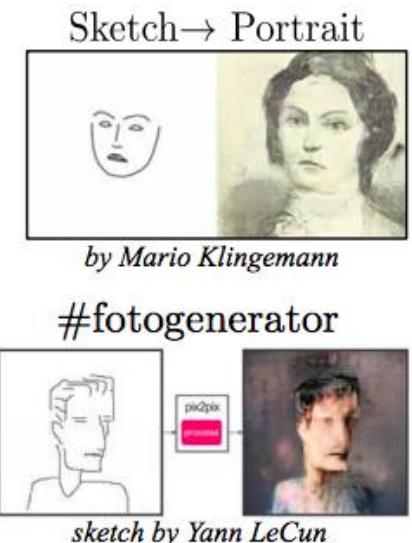
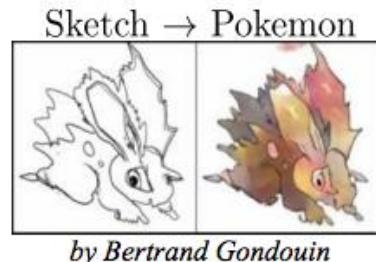
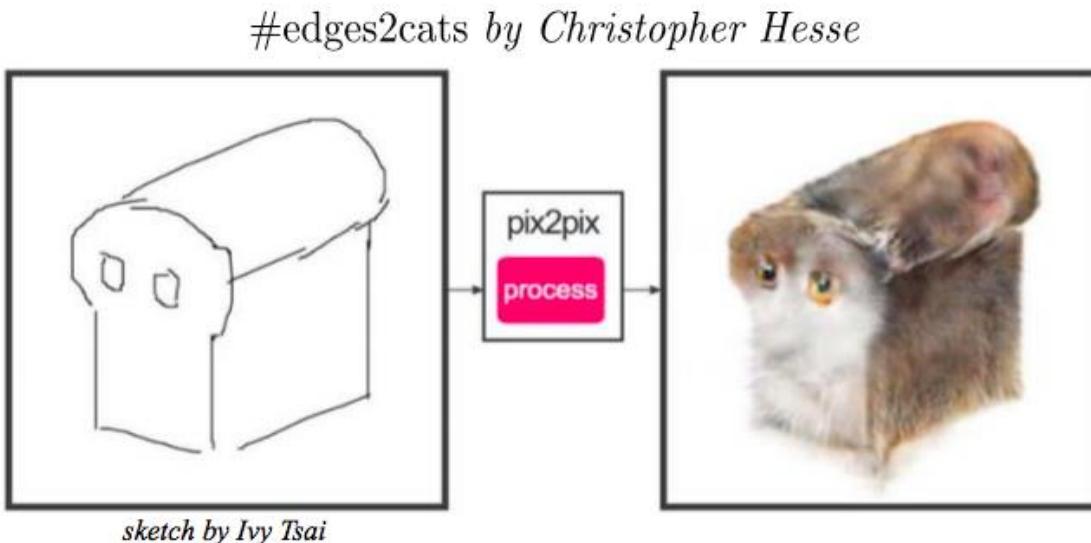
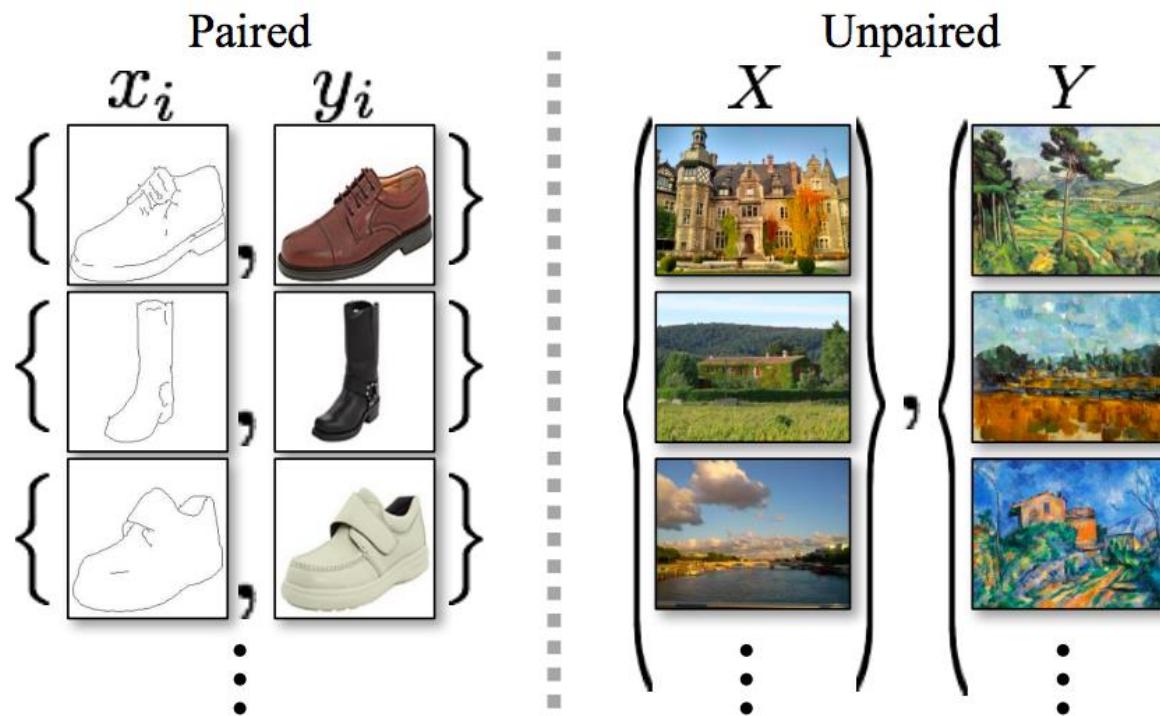


Image-to-image translation: Limitations

- Visual quality could be improved
- Requires x, y pairs for training
- Does not model conditional distribution $P(y|x)$, returns a single mode instead

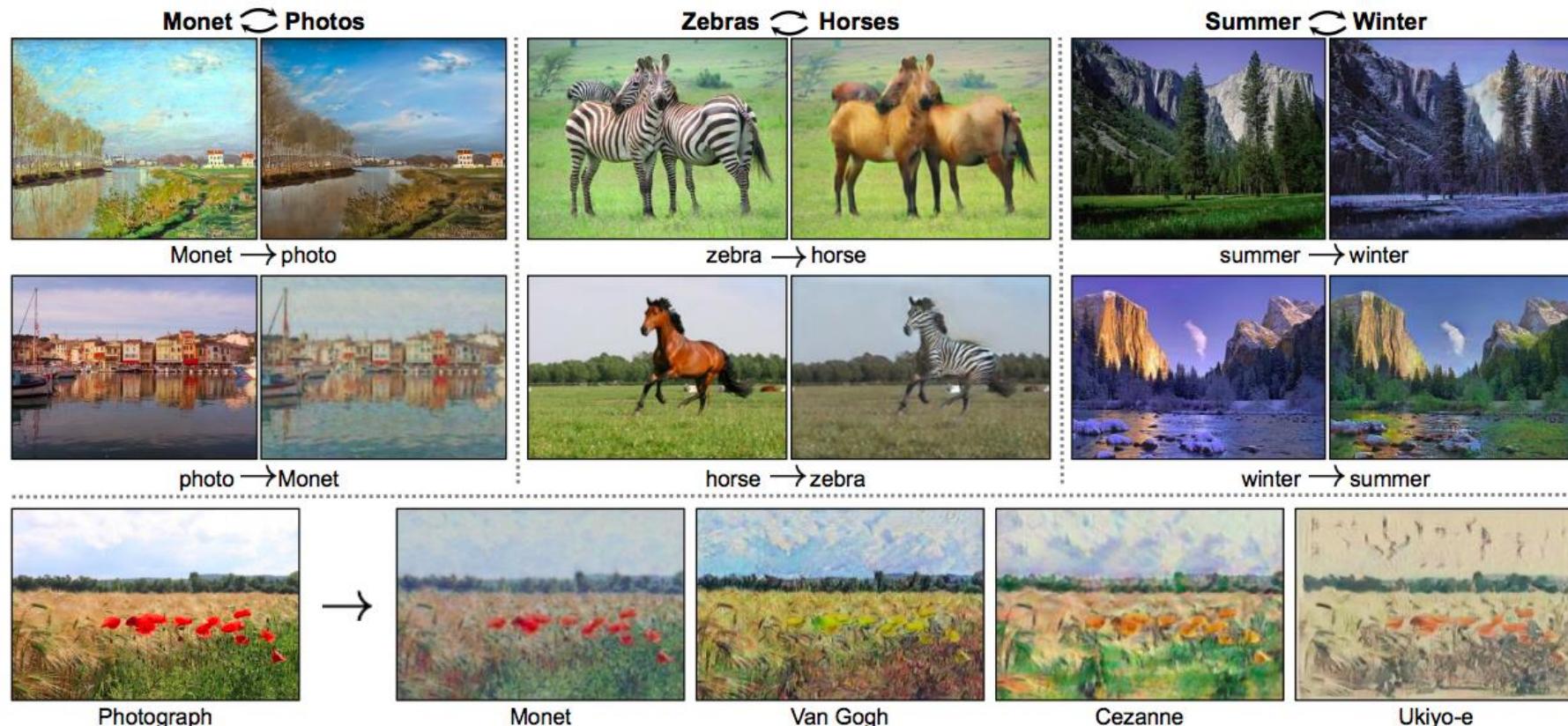
Unpaired image-to-image translation

- Given two unordered image collections X and Y , learn to “translate” an image from one into the other and vice versa



Unpaired image-to-image translation

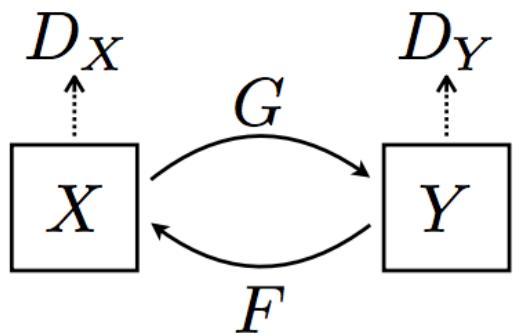
- Given two unordered image collections X and Y , learn to “translate” an image from one into the other and vice versa



J.-Y. Zhu, T. Park, P. Isola, A. Efros, [Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks](#), ICCV 2017

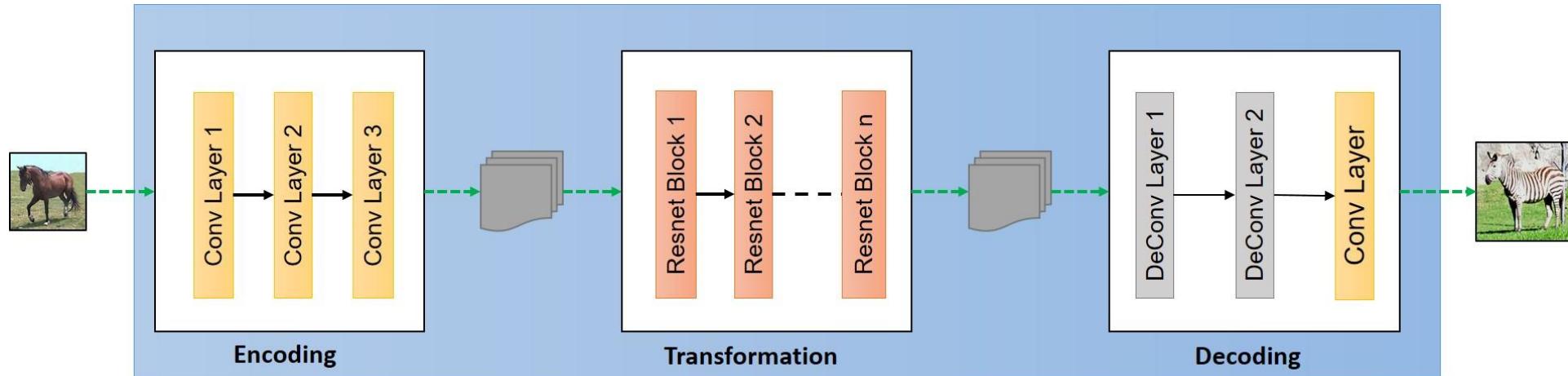
CycleGAN

- Given: domains X and Y
- Train two generators G and F and two discriminators D_X and D_Y
 - G translates from X to Y , F translates from Y to X
 - D_X recognizes images from X , D_Y from Y
 - *Cycle consistency*: we want $F(G(x)) \approx x$ and $G(F(y)) \approx y$



CycleGAN: Architecture

- Generators (based on [Johnson et al., 2016](#)):



[Figure source](#)

- Discriminators: PatchGAN on 70 x 70 patches

CycleGAN: Loss

- Requirements:
 - G translates from X to Y , F translates from Y to X
 - D_X recognizes images from X , D_Y from Y
 - We want $F(G(x)) \approx x$ and $G(F(y)) \approx y$
- CycleGAN discriminator loss: LSGAN

$$\mathcal{L}_{\text{GAN}}(D_Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[(D_Y(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D_Y(G(x))^2]$$

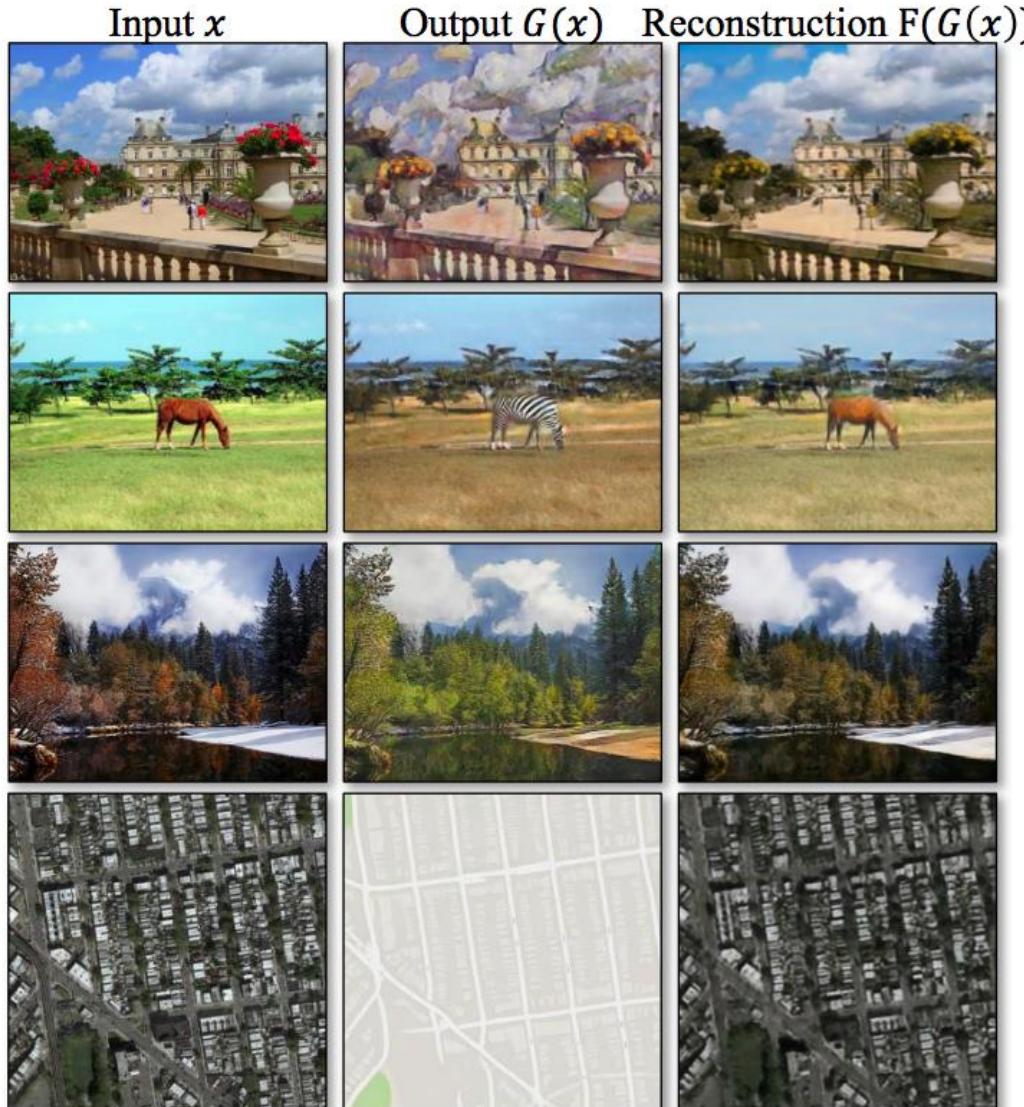
$$\mathcal{L}_{\text{GAN}}(D_X) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[(D_X(x) - 1)^2] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[D_X(F(y))^2]$$

- CycleGAN generator loss:

$$\begin{aligned} \mathcal{L}_{\text{cyc}}(G, F) = & \mathbb{E}_{x \sim p_{\text{data}}(x)}[D_Y(G(x) - 1)^2] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[D_X(F(y) - 1)^2] \\ & + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1] \end{aligned}$$

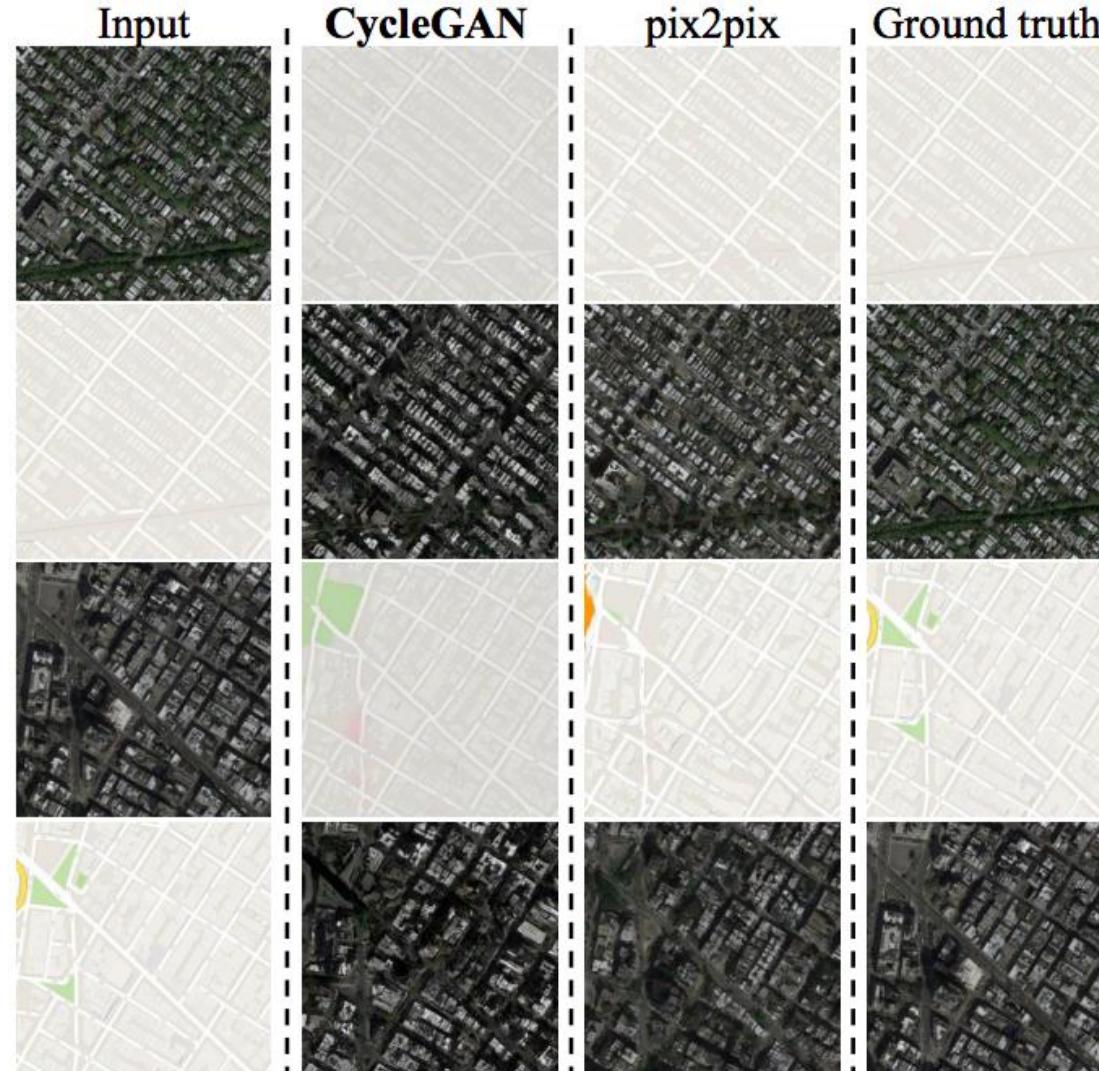
CycleGAN

- Illustration of cycle consistency:



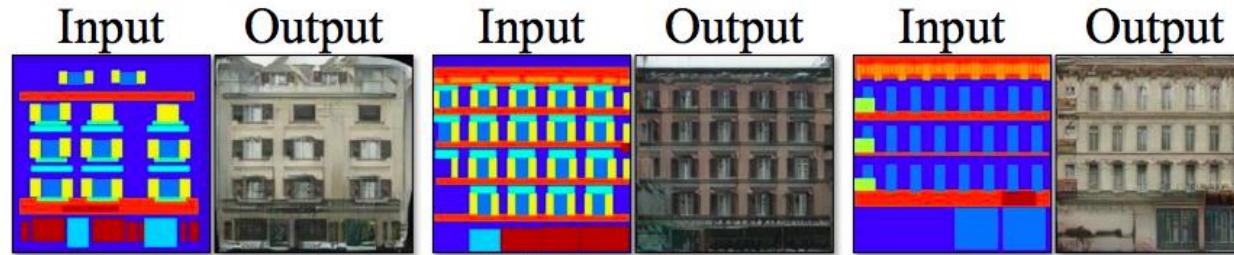
CycleGAN: Results

- Translation between maps and aerial photos

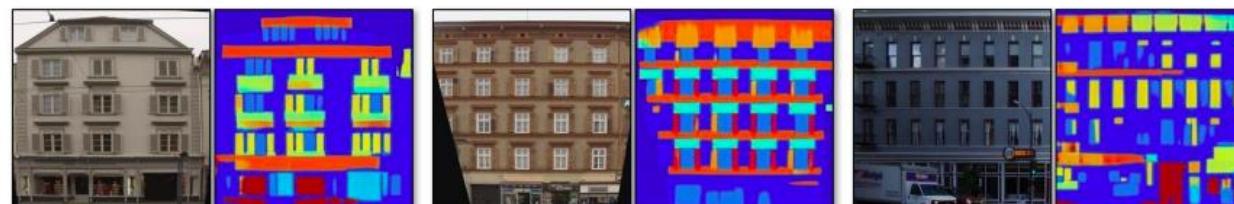


CycleGAN: Results

- Other pix2pix tasks



label → facade



facade → label



edges → shoes



shoes → edges

CycleGAN: Results

- Scene to labels and labels to scene
 - Worse performance than pix2pix due to lack of paired training data

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [32]	0.40	0.10	0.06
BiGAN/ALI [9, 7]	0.19	0.06	0.02
SimGAN [46]	0.20	0.10	0.04
Feature loss + GAN	0.06	0.04	0.01
CycleGAN (ours)	0.52	0.17	0.11
pix2pix [22]	0.71	0.25	0.18

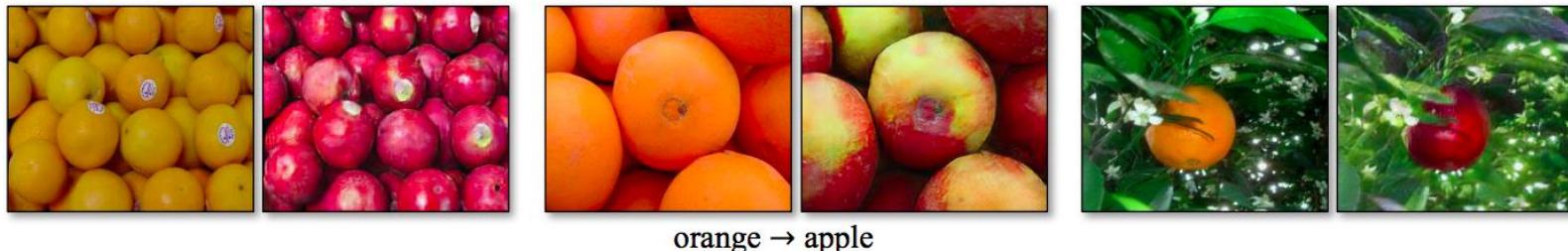
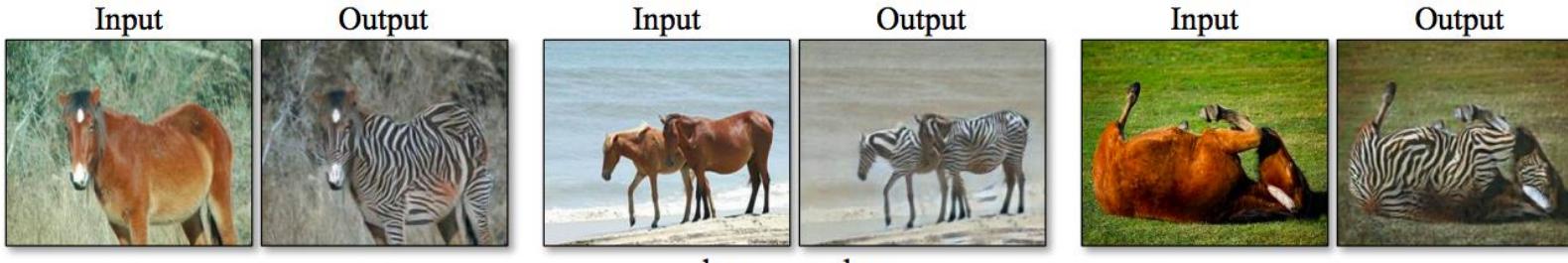
Table 2: FCN-scores for different methods, evaluated on Cityscapes labels→photo.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [32]	0.45	0.11	0.08
BiGAN/ALI [9, 7]	0.41	0.13	0.07
SimGAN [46]	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	0.58	0.22	0.16
pix2pix [22]	0.85	0.40	0.32

Table 3: Classification performance of photo→labels for different methods on cityscapes.

CycleGAN: Results

- Tasks for which paired data is unavailable

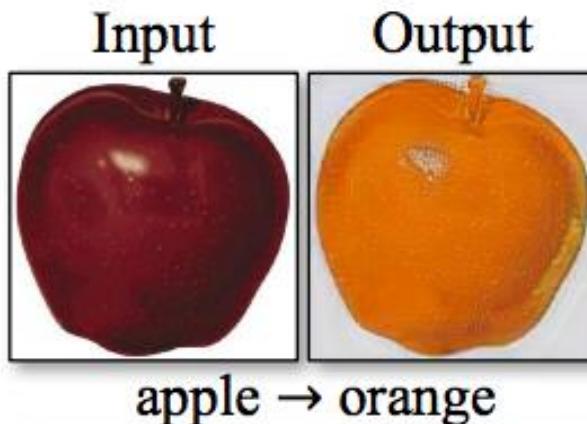


CycleGAN: Results

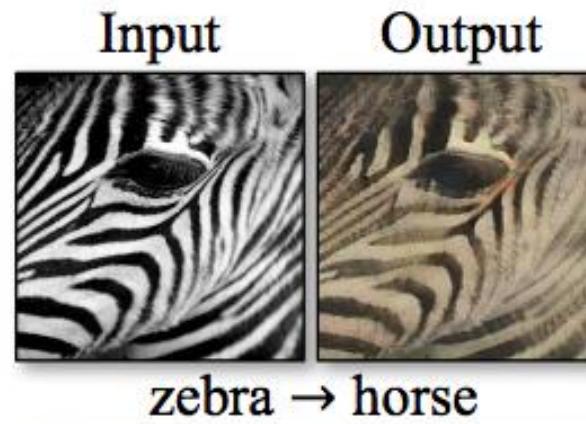
- Style transfer



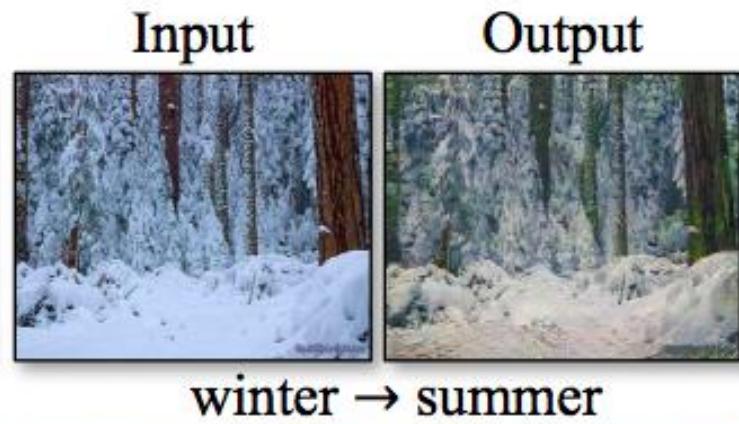
CycleGAN: Failure cases



apple → orange



zebra → horse



winter → summer



dog → cat



cat → dog



Monet → photo



photo → Ukiyo-e

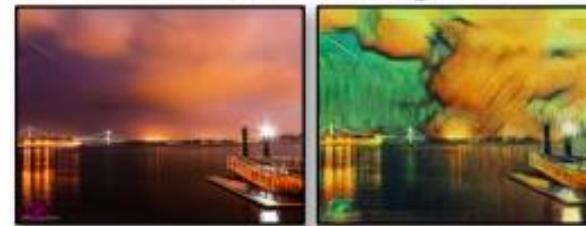


photo → Van Gogh



iPhone photo → DSLR photo

CycleGAN: Failure cases

Input



Output



horse → zebra

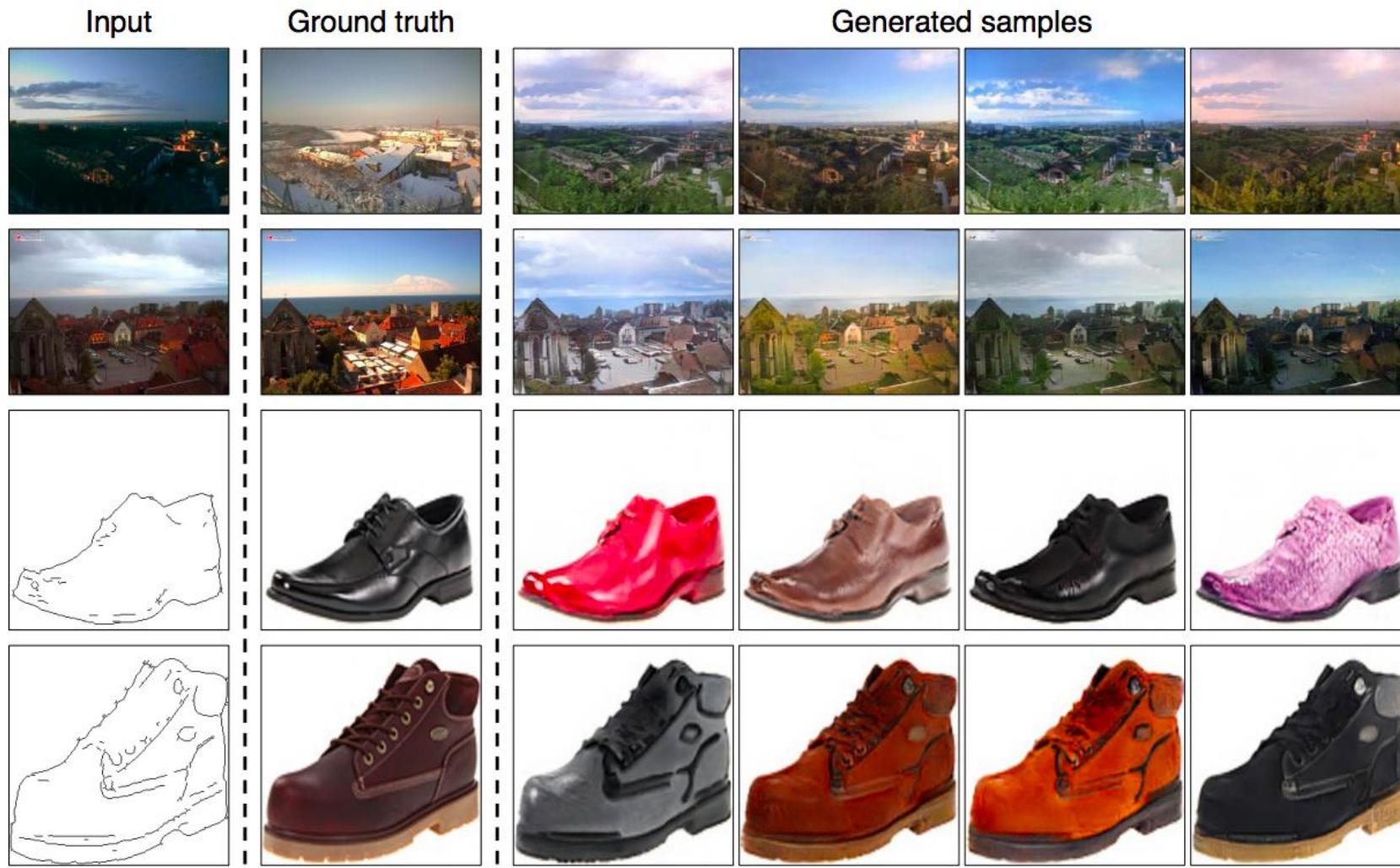
CycleGAN: Limitations

- Cannot handle shape changes (e.g., dog to cat)
- Can get confused on images outside of the training domains (e.g., horse with rider)
- Cannot close the gap with paired translation methods
- Does not account for the fact that one transformation direction may be more challenging than the other

Outline

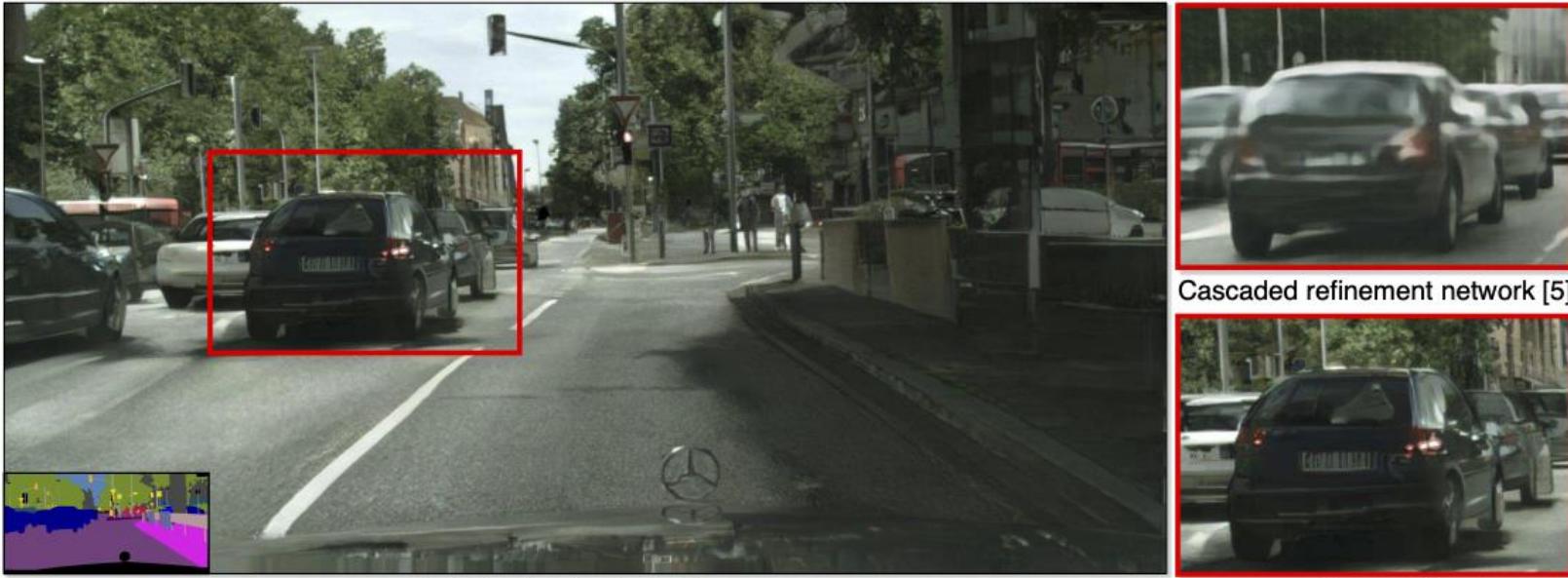
- Introduction
- Generation conditioned on class
 - Self-attention GAN
 - BigGAN
- Generation conditioned on image
 - Paired image-to-image translation: pix2pix
 - Unpaired image-to-image translation: CycleGAN
- Some recent highlights

Multimodal image-to-image translation (CycleGAN)



J.Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, E. Shechtman,
[Toward Multimodal Image-to-Image Translation](#), NIPS 2017

High-resolution, high-quality pix2pix



(a) Synthesized result

Our result



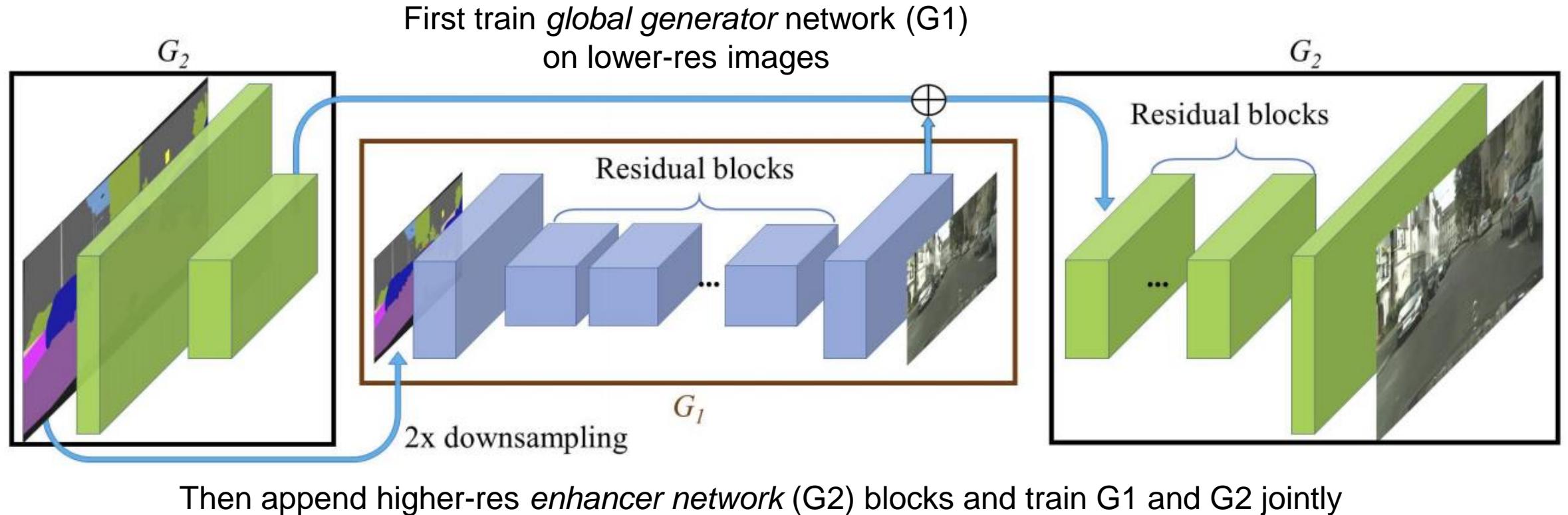
(b) Application: Change label types



(c) Application: Edit object appearance

High-resolution, high-quality pix2pix

- Two-scale generator architecture (up to 2048 x 1024 resolution)

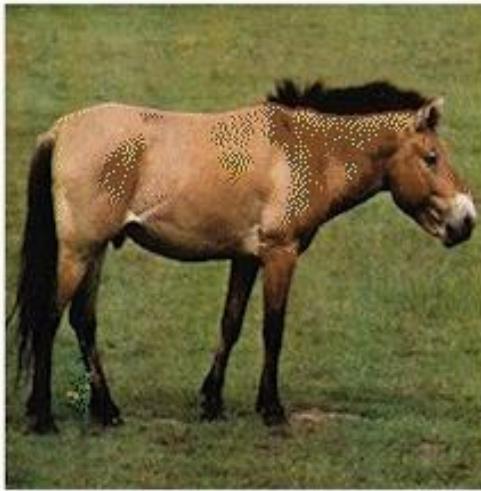


High-resolution, high-quality pix2pix

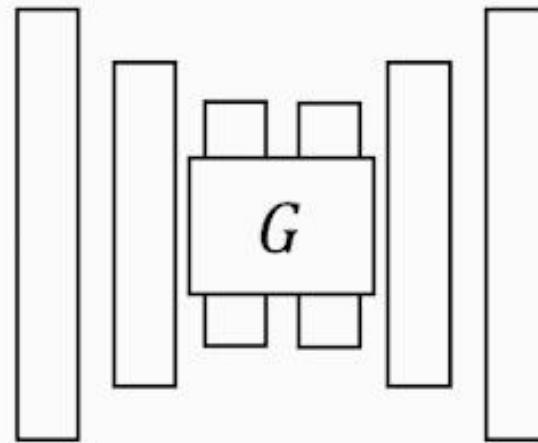
- Two-scale generator architecture (up to 2048 x 1024 resolution)
- Three-scale discriminator architecture (full res, 2x and 4x downsampled)
- Incorporate feature matching loss into discriminator

Contrastive Learning for Unpaired Translation (CUT)

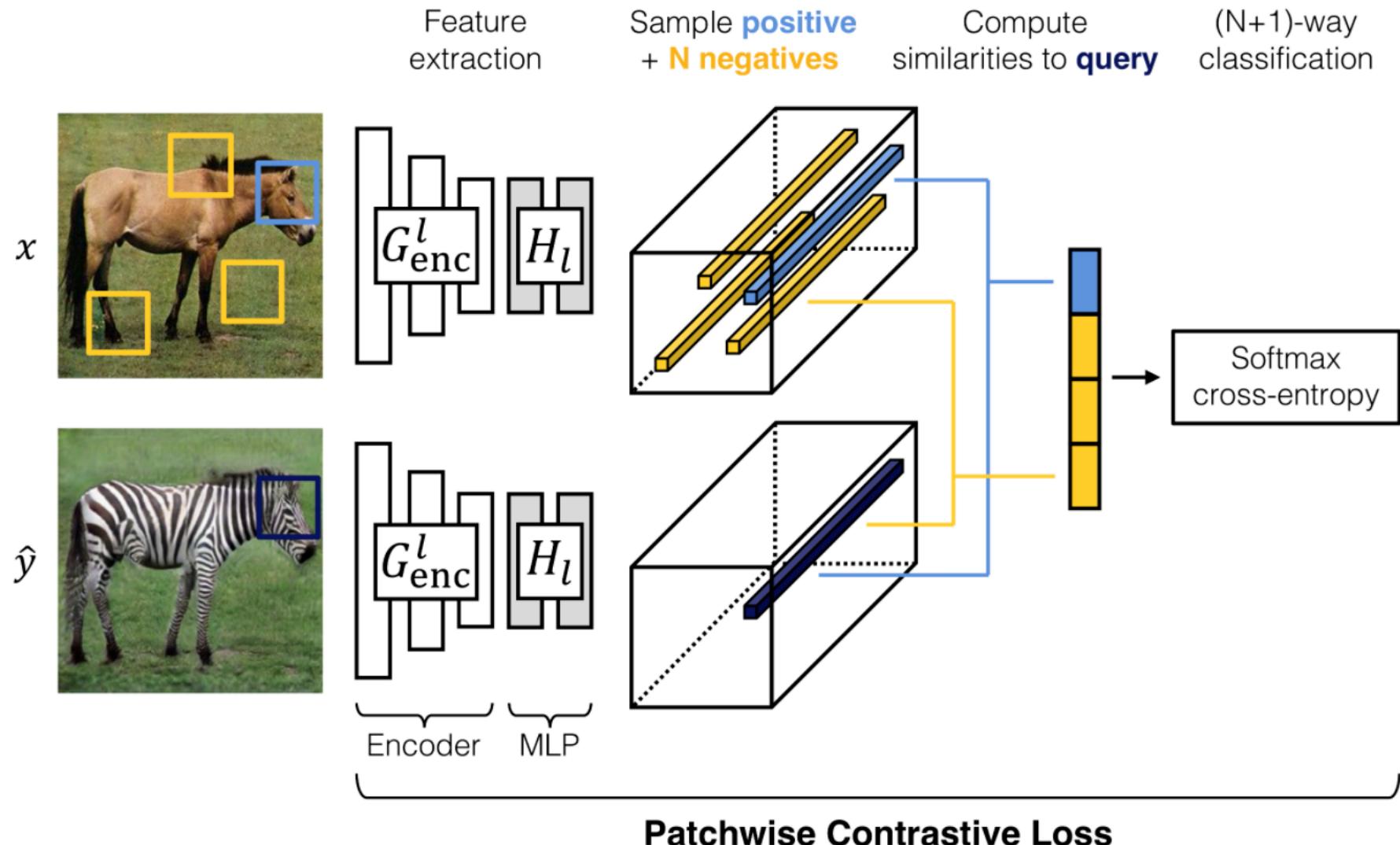
Input (horse)



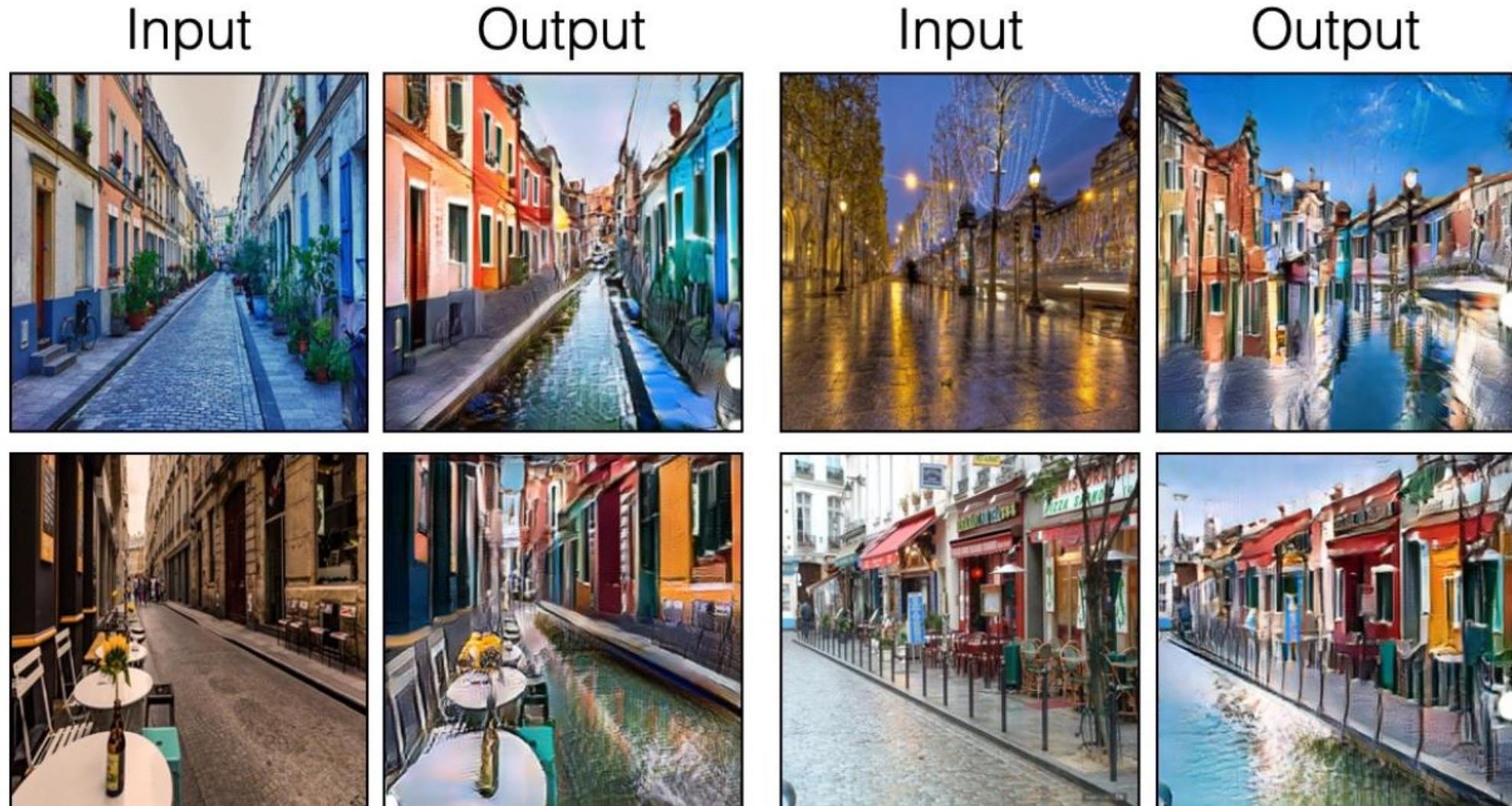
Output (zebra)



Contrastive Learning for Unpaired Translation (CUT)



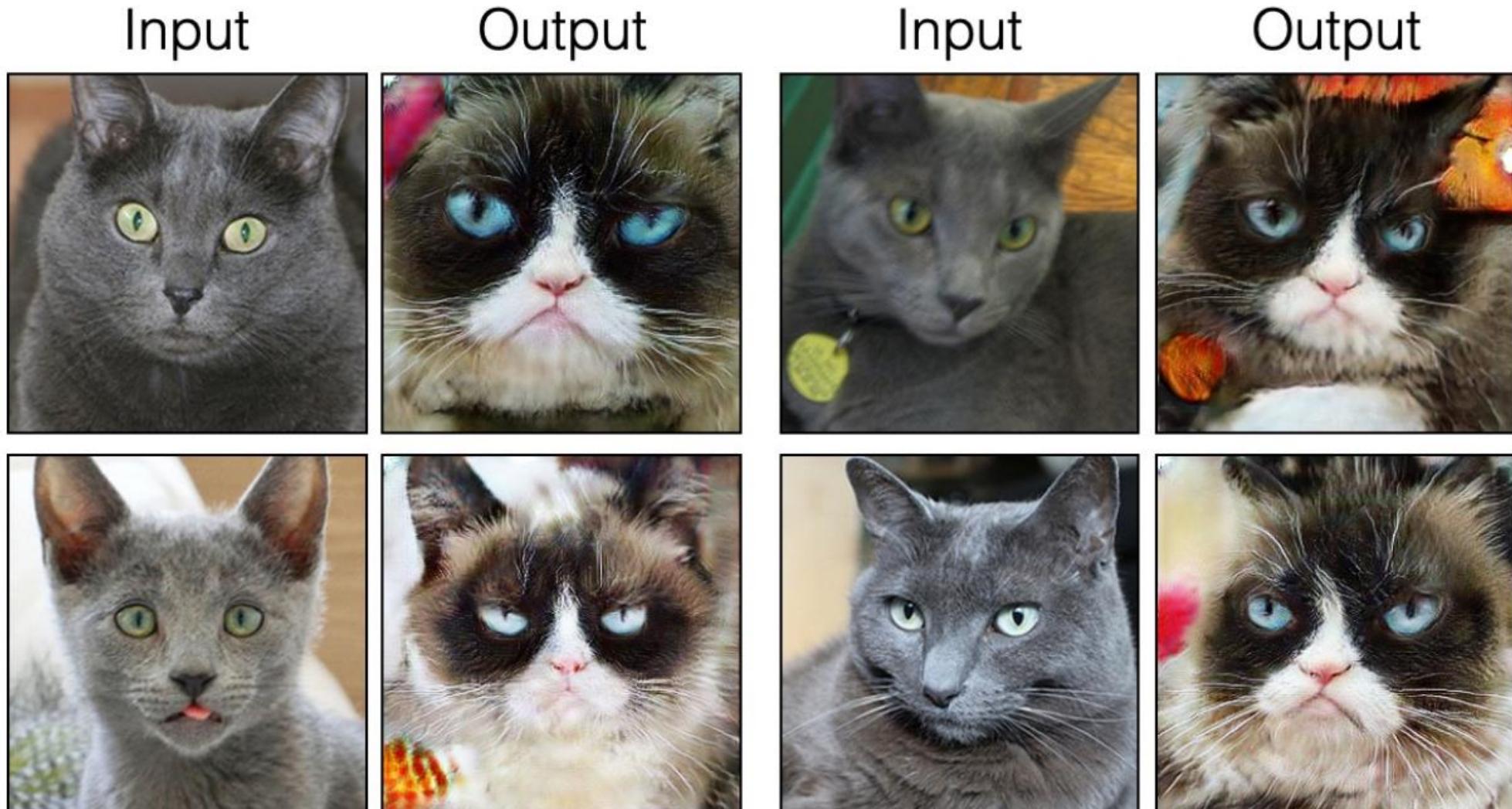
Contrastive Learning for Unpaired Translation (CUT)



Paris to Burano Streets

Contrastive Learning for Unpaired Image-to-Image Translation. ECCV 2020

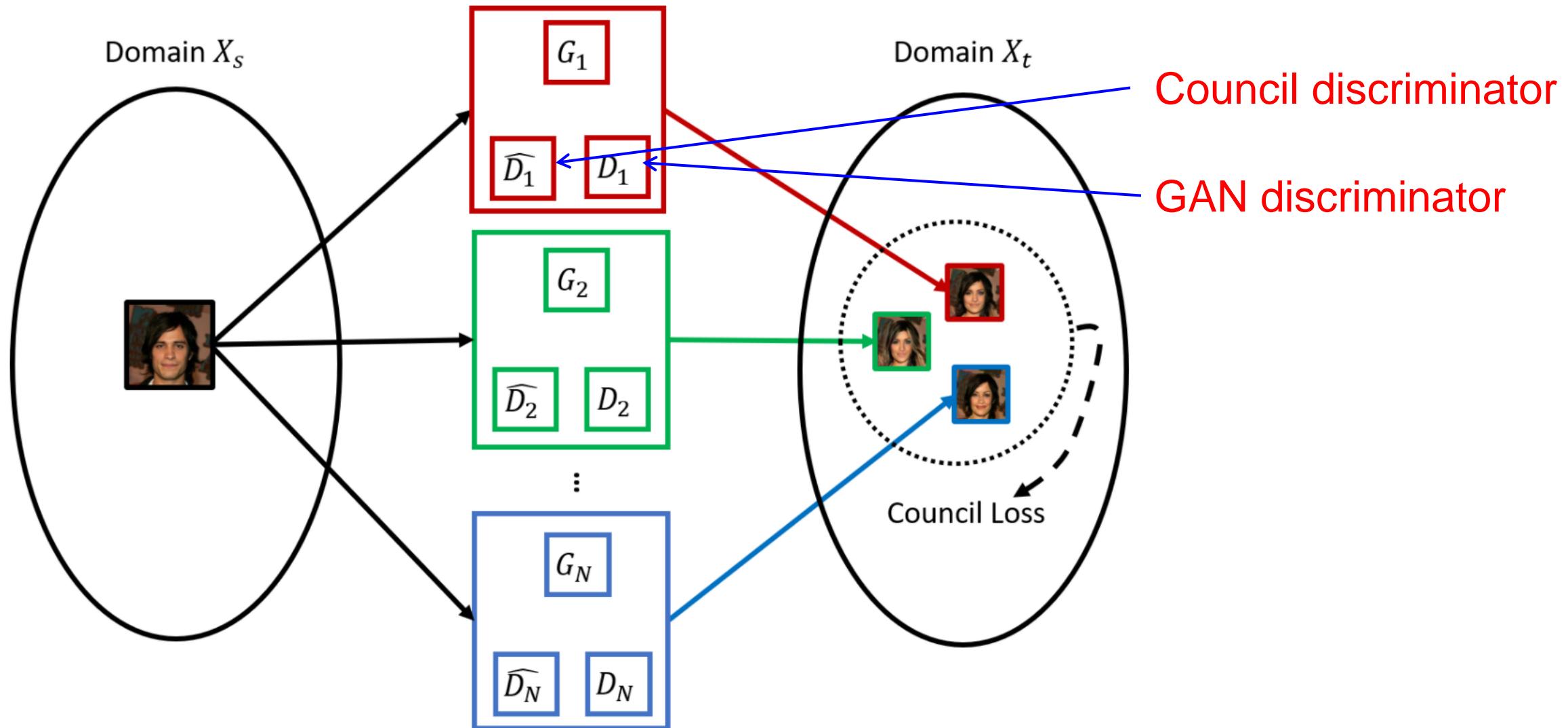
Contrastive Learning for Unpaired Translation (CUT)



Russian Blue -> Grumpy Cats

Contrastive Learning for Unpaired Image-to-Image Translation. ECCV 2020

Council-GAN



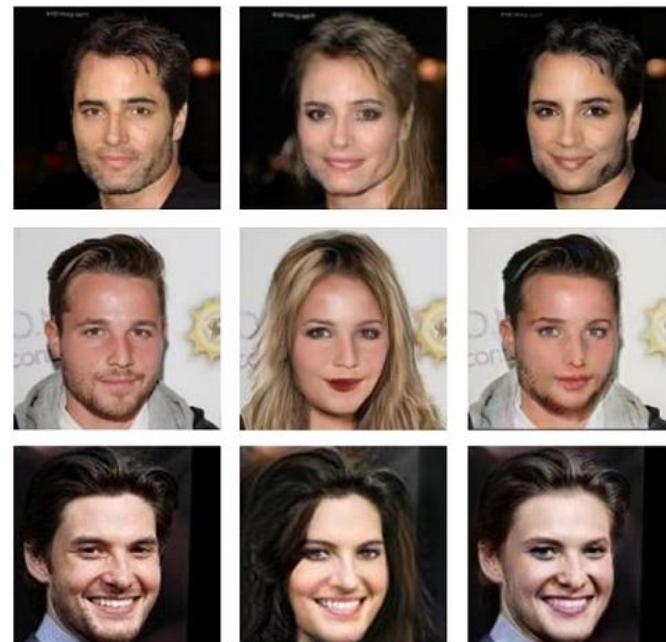
Council-GAN

Unpaired Image to Image Translation

Selfie-to-anime



Male-to-female



Glasses removal



input

Ours-1

Ours-2

U-GAT-IT

input

Ours

Star-GAN

input

Ours

Fixed-Point

“The GAN Zoo”

- 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling ([github](#))
- 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction ([github](#))
- 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning ([github](#))
- ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks ([github](#))
- ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- ACTual - ACTual: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- AdvGAN - Generating adversarial examples with adversarial networks
- AE-GAN - AE-GAN: adversarial eliminating with GAN
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference ([github](#))
- AlignGAN - AlignGAN: Learning to Align Cross-Domain Images with Conditional Generative Adversarial Networks
- AM-GAN - Activation Maximization Generative Adversarial Nets
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- APE-GAN - APE-GAN: Adversarial Perturbation Elimination with GAN
- ARAE - Adversarially Regularized Autoencoders for Generating Discrete Structures ([github](#))
- ARDA - Adversarial Representation Learning for Domain Adaptation
- ARIGAN - ARIGAN: Synthetic Arabidopsis Plants using Generative Adversarial Network
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- AttGAN - Arbitrary Facial Attribute Editing: Only Change What You Want
- AttnGAN - AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks
- b-GAN - Generative Adversarial Nets from a Density Ratio Estimation Perspective
- Bayesian GAN - Deep and Hierarchical Implicit Models
- Bayesian GAN - Bayesian GAN ([github](#))
- BCGAN - Bayesian Conditional Generative Adversarial Networks
- BCGAN - Bidirectional Conditional Generative Adversarial networks
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BGAN - Binary Generative Adversarial Networks for Image Retrieval ([github](#))
- BicycleGAN - Toward Multimodal Image-to-Image Translation ([github](#))
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- C-GAN - Face Aging with Contextual Generative Adversarial Nets
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training ([github](#))
- CA-GAN - Composition-aided Sketch-realistic Portrait Generation
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks ([github](#))
- CAN - CAN: Creative Adversarial Networks, Generating Art by Learning About Styles and Deviating from Style Norms
- CapsuleGAN - CapsuleGAN: Generative Adversarial Capsule Network
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CatGAN - CatGAN: Coupled Adversarial Transfer for Domain Generation
- CausalGAN - CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training
- CC-GAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks ([github](#))
- CDCGAN - Simultaneously Color-Depth Super-Resolution with Conditional Generative Adversarial Network
- CFG-GAN - Composite Functional Gradient Learning of Generative Adversarial Models
- CGAN - Conditional Generative Adversarial Nets
- CGAN - Controllable Generative Adversarial Network
- Chekhov GAN - An Online Learning Approach to Generative Adversarial Networks
- CipherGAN - Unsupervised Cipher Cracking Using Discrete GANs
- CM-GAN - CM-GANs: Cross-modal Generative Adversarial Networks for Common Representation Learning
- CoAtt-GAN - Are You Talking to Me? Reasoned Visual Dialog Generation through Adversarial Learning
- CoGAN - Coupled Generative Adversarial Networks
- ComboGAN - ComboGAN: Unrestrained Scalability for Image Domain Translation ([github](#))

“The GAN Zoo”

- ConceptGAN - [Learning Compositional Visual Concepts with Mutual Consistency](#)
- Conditional cycleGAN - [Conditional CycleGAN for Attribute Guided Face Image Generation](#)
- contrast-GAN - [Generative Semantic Manipulation with Contrasting GAN](#)
- Context-RNN-GAN - [Contextual RNN-GANs for Abstract Reasoning Diagram Generation](#)
- Coulomb GAN - [Coulomb GANs: Provably Optimal Nash Equilibria via Potential Fields](#)
- Cover-GAN - [Generative Steganography with Kerckhoffs' Principle based on Generative Adversarial Networks](#)
- Cramér GAN - [The Cramer Distance as a Solution to Biased Wasserstein Gradients](#)
- Cross-GAN - [Crossing Generative Adversarial Networks for Cross-View Person Re-identification](#)
- crVAE-GAN - [Channel-Recurrent Variational Autoencoders](#)
- CS-GAN - [Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets](#)
- CVAE-GAN - [CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training](#)
- CycleGAN - [Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks \(github\)](#)
- D-GAN - [Differential Generative Adversarial Networks: Synthesizing Non-linear Facial Variations with Limited Number of Training Data](#)
- D2GAN - [Dual Discriminator Generative Adversarial Nets](#)
- DA-GAN - [DA-GAN: Instance-level Image Translation by Deep Attention Generative Adversarial Networks \(with Supplementary Materials\)](#)
- DAGAN - [Data Augmentation Generative Adversarial Networks](#)
- DAN - [Distributional Adversarial Networks](#)
- DCGAN - [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks \(github\)](#)
- DeblurGAN - [DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks \(github\)](#)
- Defense-GAN - [Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models](#)
- DeliGAN - [DeLiGAN : Generative Adversarial Networks for Diverse and Limited Data \(github\)](#)
- DF-GAN - [Learning Disentangling and Fusing Networks for Face Completion Under Structured Occlusions](#)
- DiscoGAN - [Learning to Discover Cross-Domain Relations with Generative Adversarial Networks](#)
- DistanceGAN - [One-Sided Unsupervised Domain Mapping](#)
- DM-GAN - [Dual Motion GAN for Future-Flow Embedded Video Prediction](#)
- DNA-GAN - [DNA-GAN: Learning Disentangled Representations from Multi-Attribute Images](#)
- dp-GAN - [Differentially Private Releasing via Deep Generative Model](#)
- DP-GAN - [DP-GAN: Diversity-Promoting Generative Adversarial Network for Generating Informative and Diversified Text](#)
- DPGAN - [Differentially Private Generative Adversarial Network](#)
- DR-GAN - [Representation Learning by Rotating Your Faces](#)
- DRAGAN - [How to Train Your DRAGAN \(github\)](#)
- DRGAN - [Discriminative Region Proposal Adversarial Networks for High-Quality Image-to-Image Translation](#)
- DSP-GAN - [Depth Structure Preserving Scene Image Generation](#)
- DTN - [Unsupervised Cross-Domain Image Generation](#)
- DualGAN - [DualGAN: Unsupervised Dual Learning for Image-to-Image Translation](#)
- Dualing GAN - [Dualing GANs](#)
- Dynamics Transfer GAN - [Dynamics Transfer GAN: Generating Video by Transferring Arbitrary Temporal Dynamics from a Source Video to a Single Target Image](#)
- EBGAN - [Energy-based Generative Adversarial Network](#)
- ecGAN - [eCommerceGAN : A Generative Adversarial Network for E-commerce](#)
- ED//GAN - [Stabilizing Training of Generative Adversarial Networks through Regularization](#)
- EGAN - [Enhanced Experience Replay Generation for Efficient Reinforcement Learning](#)
- EnergyWGAN - [Energy-relaxed Wasserstein GANs \(EnergyWGAN\): Towards More Stable and High Resolution Image Generation](#)
- ExGAN - [Eye In-Painting with Exemplar Generative Adversarial Networks](#)
- ExposureGAN - [Exposure: A White-Box Photo Post-Processing Framework \(github\)](#)
- ExprGAN - [ExprGAN: Facial Expression Editing with Controllable Expression Intensity](#)
- f-CLSWGAN - [Feature Generating Networks for Zero-Shot Learning](#)
- f-GAN - [f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization](#)
- FF-GAN - [Towards Large-Pose Face Frontalization in the Wild](#)
- FIGAN - [Frame Interpolation with Multi-Scale Deep Loss Functions and Generative Adversarial Networks](#)
- Fila-GAN - [Synthesizing Filamentary Structured Images with GANs](#)
- First Order GAN - [First Order Generative Adversarial Networks](#)
- Fisher GAN - [Fisher GAN](#)
- Flow-GAN - [Flow-GAN: Bridging implicit and prescribed learning in generative models](#)
- FSEGAN - [Exploring Speech Enhancement with Generative Adversarial Networks for Robust Speech Recognition](#)
- FTGAN - [Hierarchical Video Generation from Orthogonal Information: Optical Flow and Texture](#)
- FusedGAN - [Semi-supervised FusedGAN for Conditional Image Generation](#)

“The GAN Zoo”

- FusionGAN - Learning to Fuse Music Genres with Generative Adversarial Dual Learning
- G2-GAN - Geometry Guided Adversarial Facial Expression Synthesis
- GAGAN - GAGAN: Geometry-Aware Generative Adversarial Networks
- GAMN - Generative Adversarial Mapping Networks
- GAN - Generative Adversarial Networks (github)
- GAN-ATV - A Novel Approach to Artistic Textual Visualization via GAN
- GAN-CLS - Generative Adversarial Text to Image Synthesis (github)
- GAN-RS - Towards Qualitative Advancement of Underwater Machine Vision with Generative Adversarial Networks
- GAN-sep - GANs for Biological Image Synthesis (github)
- GAN-VFS - Generative Adversarial Network-based Synthesis of Visible Faces from Polarimetric Thermal Faces
- GANCS - Deep Generative Adversarial Networks for Compressed Sensing Automates MRI
- GANDI - Guiding the search in continuous state-action spaces by learning an action sampling distribution from off-target samples
- GANG - GANGs: Generative Adversarial Network Games
- GANosaic - GANosaic: Mosaic Creation with Generative Texture Manifolds
- GAP - Context-Aware Generative Adversarial Privacy
- GAWWN - Learning What and Where to Draw (github)
- GC-GAN - Geometry-Contrastive Generative Adversarial Network for Facial Expression Synthesis
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data (github) 
- GeoGAN - Generating Instance Segmentation Annotation by Geometry-guided GAN
- Geometric GAN - Geometric GAN
- GLCA-GAN - Global and Local Consistent Age Generative Adversarial Networks
- GMAN - Generative Multi-Adversarial Networks
- GMM-GAN - Towards Understanding the Dynamics of Generative Adversarial Networks
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending (github)
- GP-GAN - GP-GAN: Gender Preserving GAN for Synthesizing Faces from Landmarks
- GPU - A generative adversarial framework for positive-unlabeled classification
- GRAN - Generating images with recurrent adversarial networks (github)

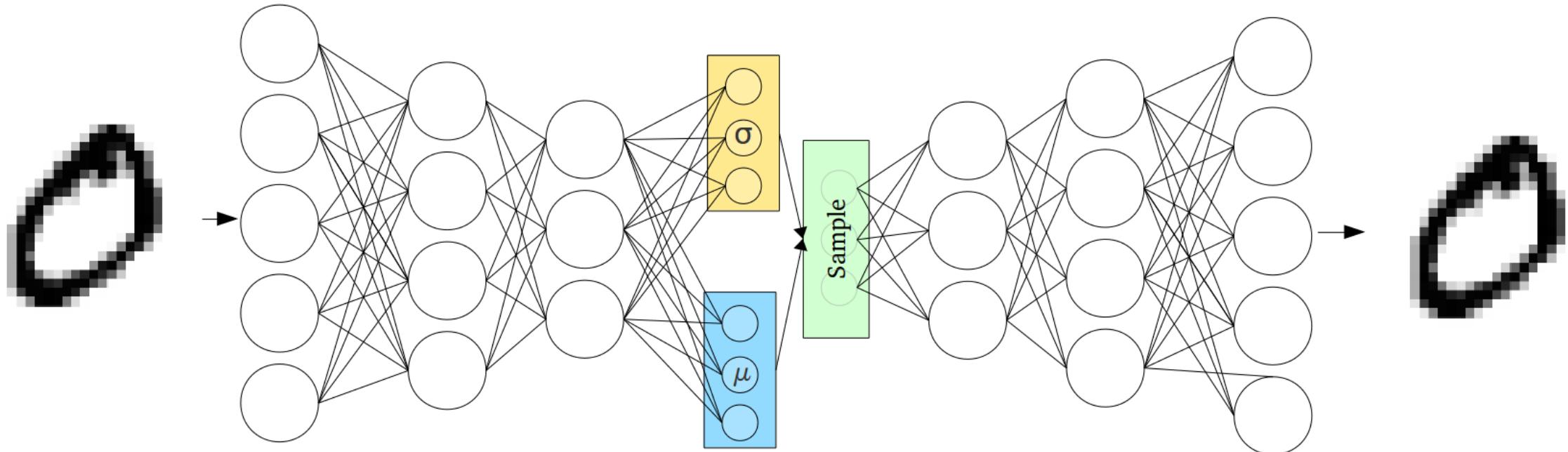
And Many More

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Variational Autoencoders (VAEs)

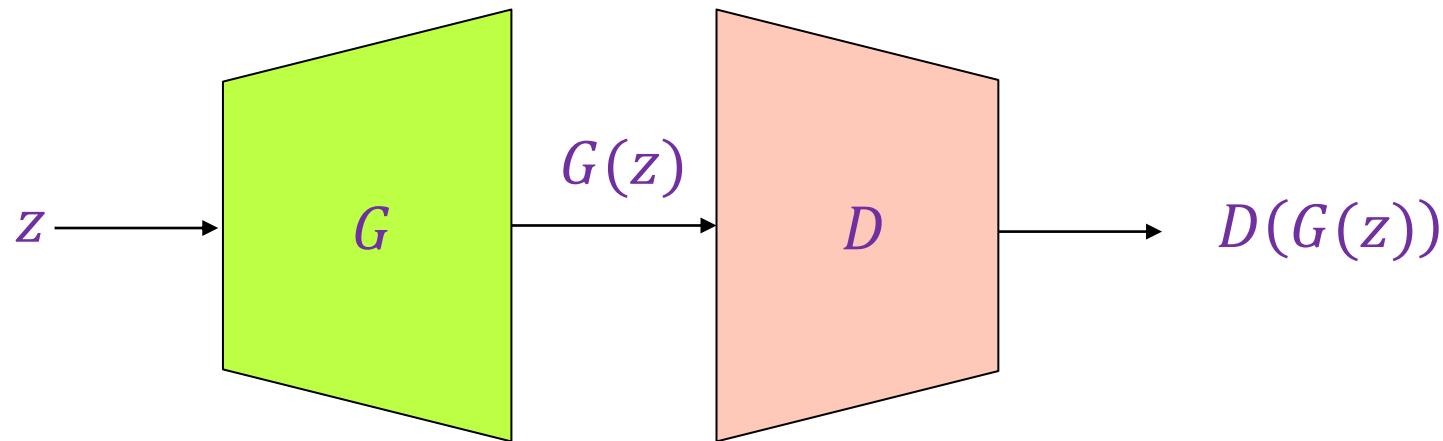


Outline

- Basic VAE formulation
- Highlights of recent work

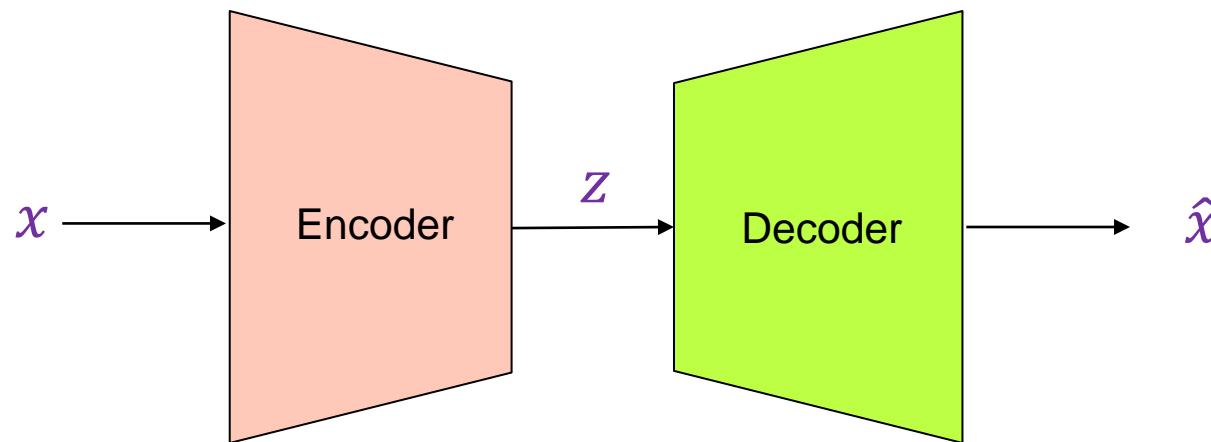
Recall: GANs

- Training:
 - Discriminator: low scores for fake data, high scores for real data
 - Generator: increase discriminator score on fake data
- Test time: discard discriminator and use generator to sample from learned distribution



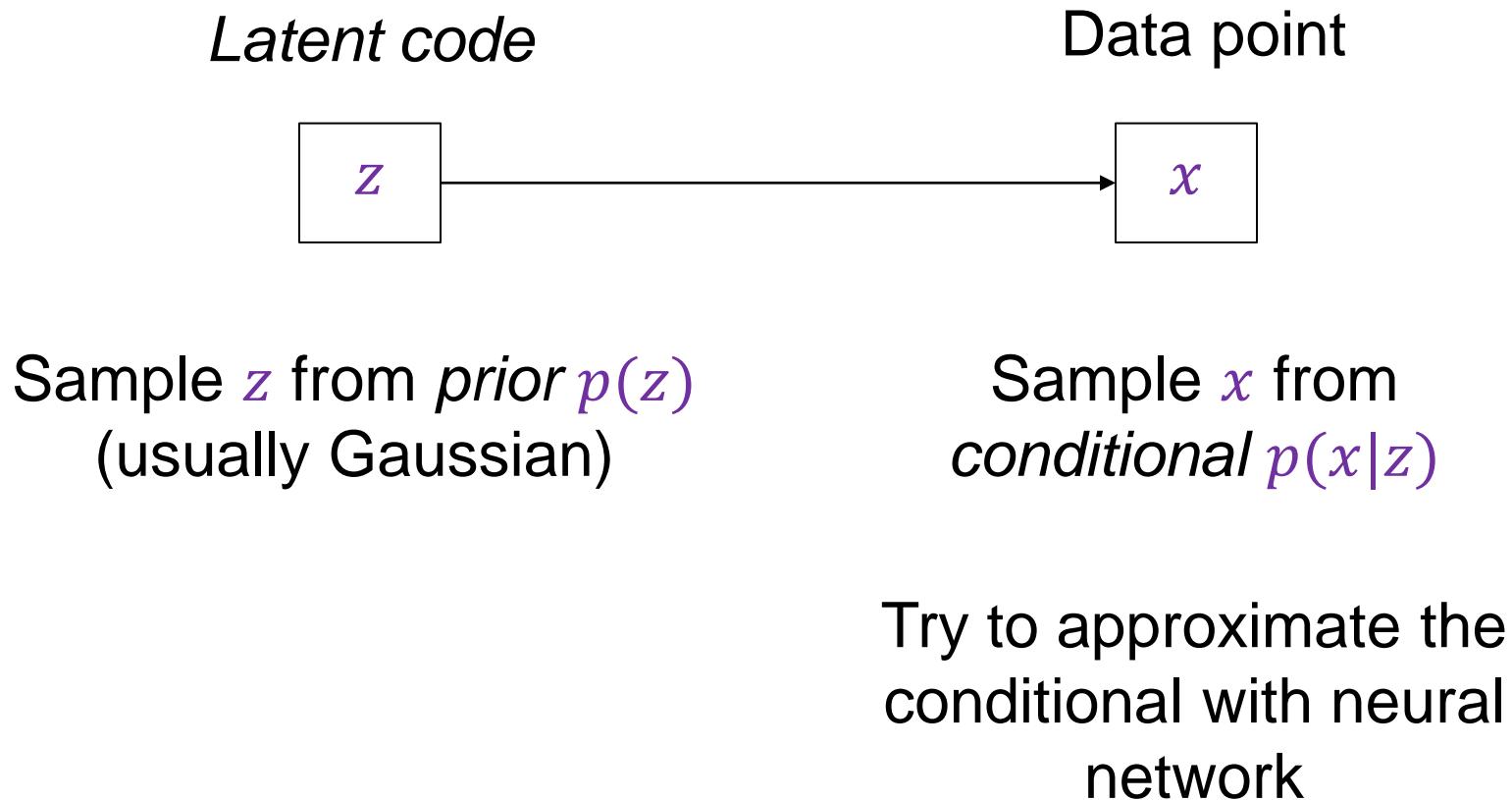
Variational autoencoders: Overview

- Probabilistic formulation based on *variational Bayes* framework
- At training time, jointly learn *encoder* and *decoder* by maximizing (a bound on) the data likelihood
- At test time, discard encoder and use decoder to sample from the learned distribution



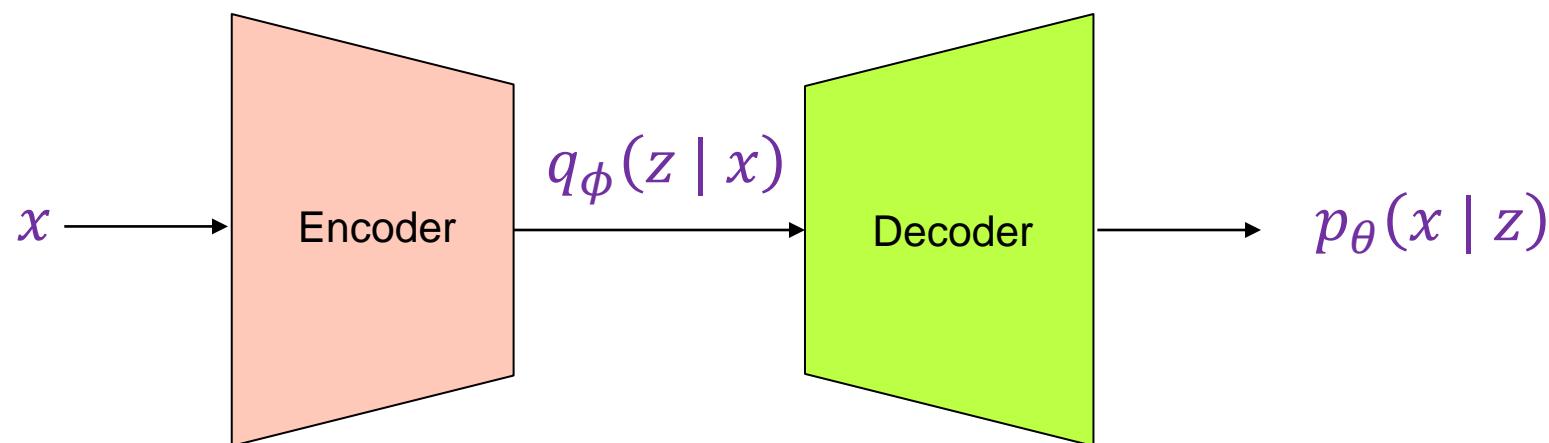
Variational autoencoders: Overview

- Probabilistic generative model of the data distribution:



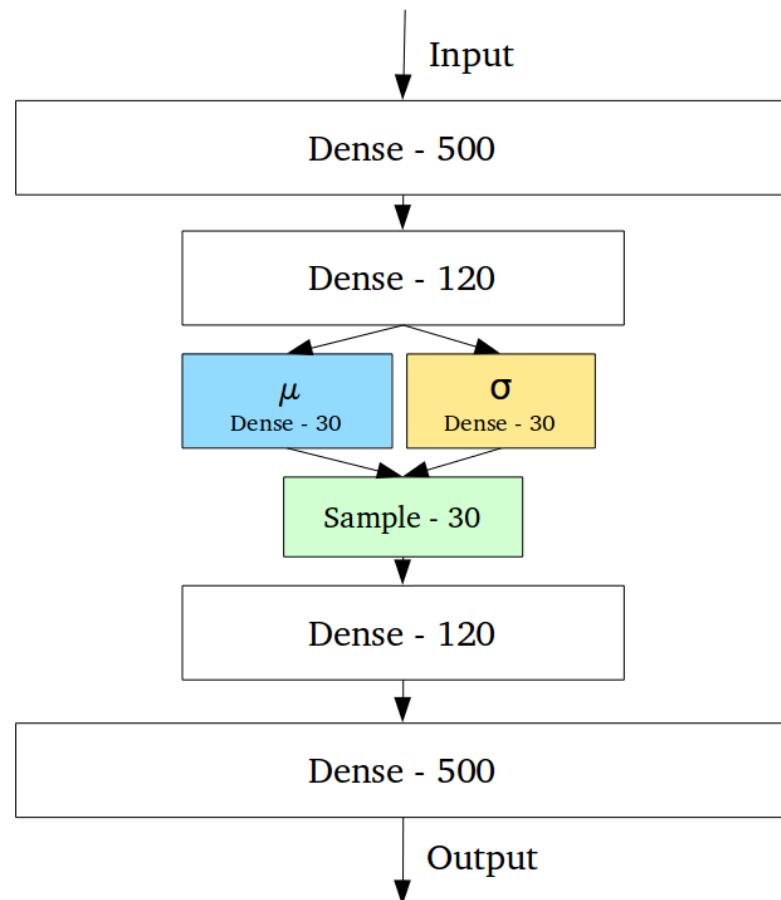
Variational autoencoders: Overview

- At training time, jointly learn *encoder* and *decoder*
- **Encoder:** given inputs x , output $q_\phi(z | x)$
 - Specifically, output mean and (diagonal) covariance, or $\mu_{z|x}$ and $\Sigma_{z|x}$, so that $q_\phi(z | x) = N(\mu_{z|x}, \Sigma_{z|x})$
- **Decoder:** given z , output $p_\theta(x | z)$
 - Specifically, output $\mu_{x|z}$ and $\Sigma_{x|z}$ so that $p_\theta(x | z) = N(\mu_{x|z}, \Sigma_{x|z})$
- **Training objective:** (a bound on) data likelihood under the model



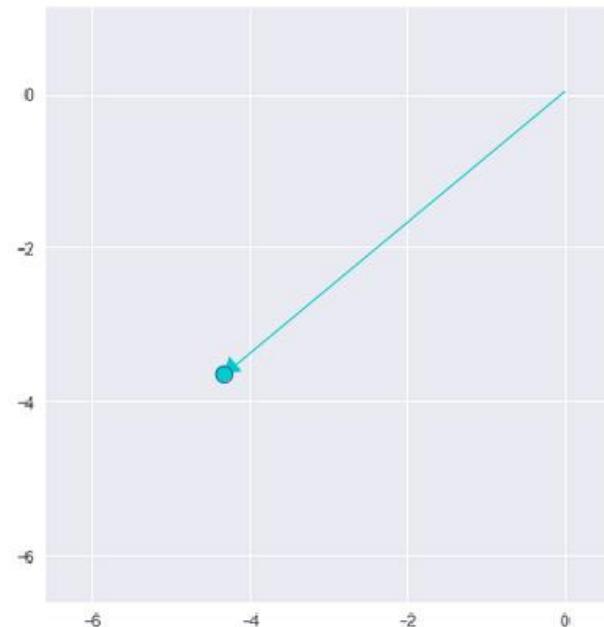
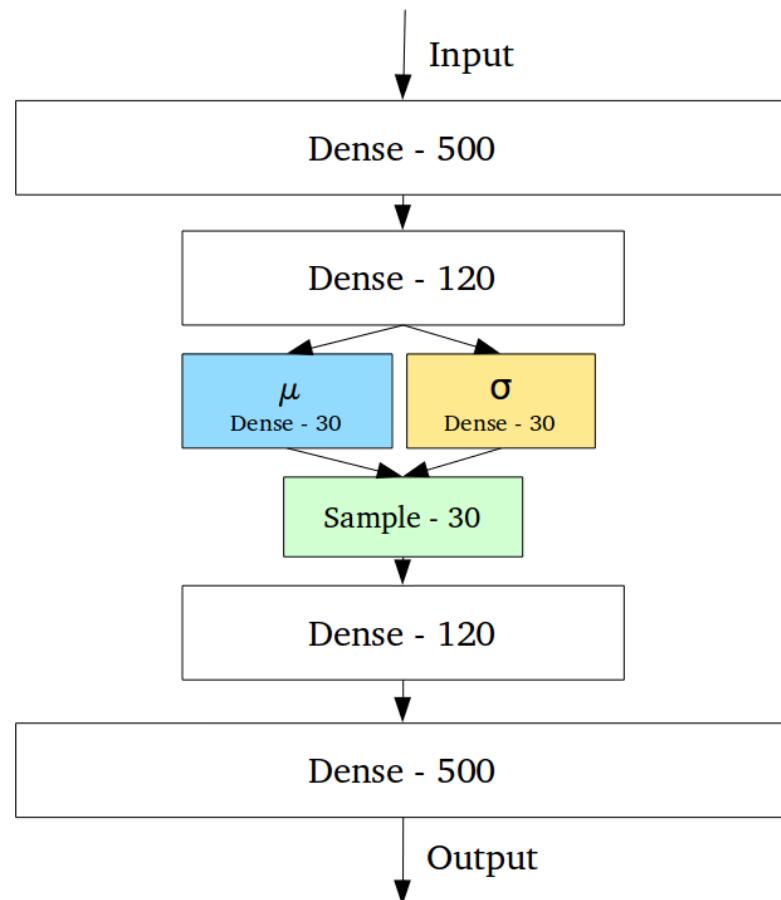
Variational autoencoders: Overview

- VAE makes its encoder not output an encoding vector of size n , rather, outputting two vectors of size n : a vector of means, μ , and another vector of standard deviations, σ .

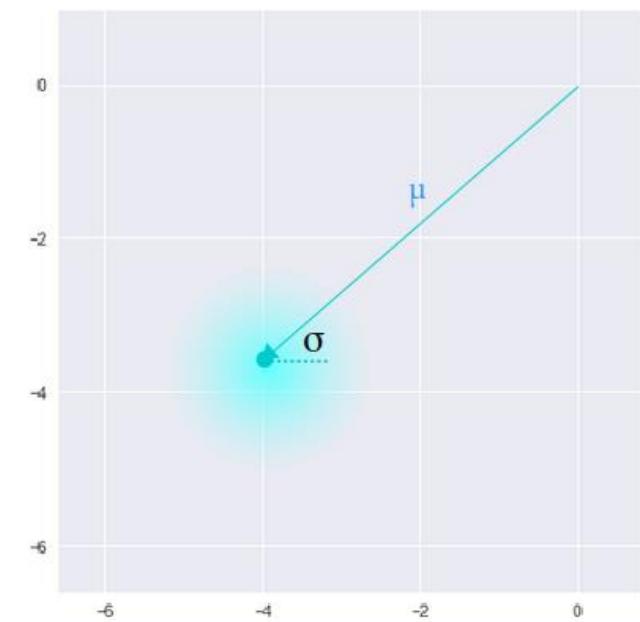


Variational autoencoders: Overview

- VAE makes its encoder not output an encoding vector of size n , rather, outputting two vectors of size n : a vector of means, μ , and another vector of standard deviations, σ .



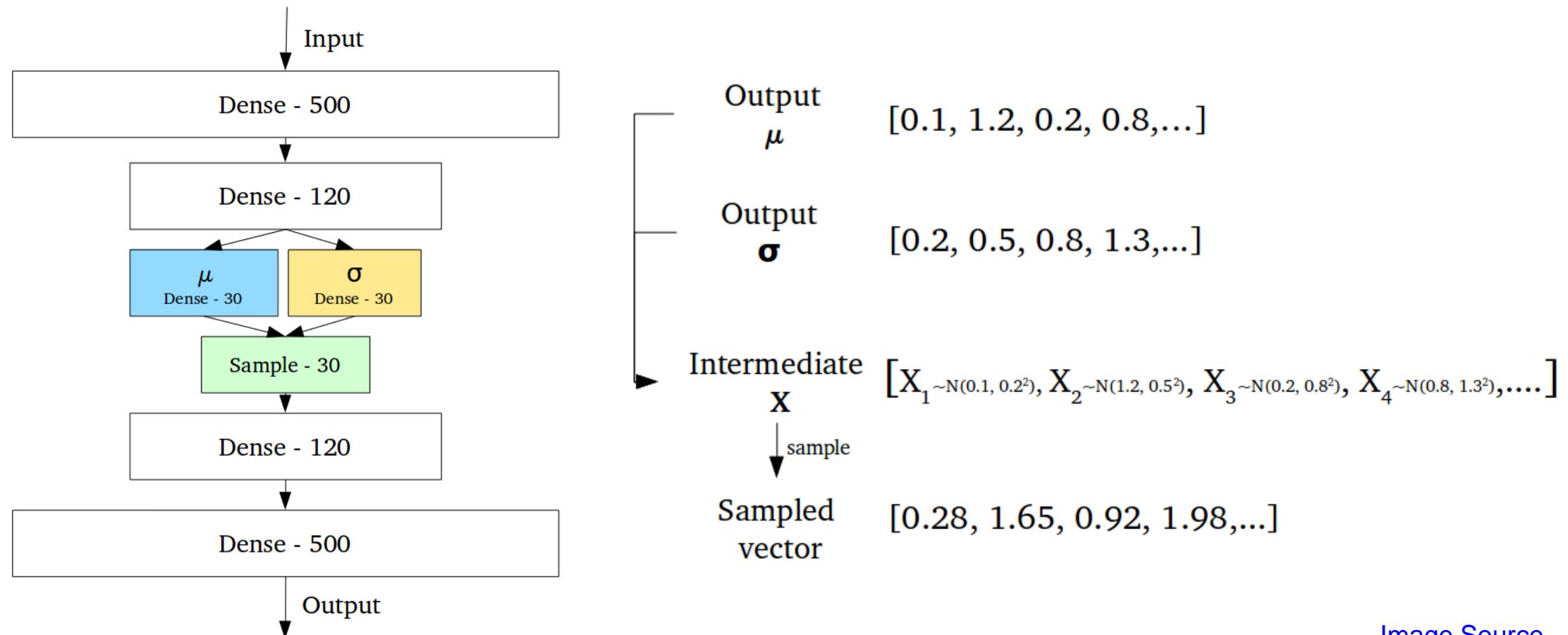
Standard Autoencoder
(direct encoding coordinates)



Variational Autoencoder
(μ and σ initialize a probability distribution)

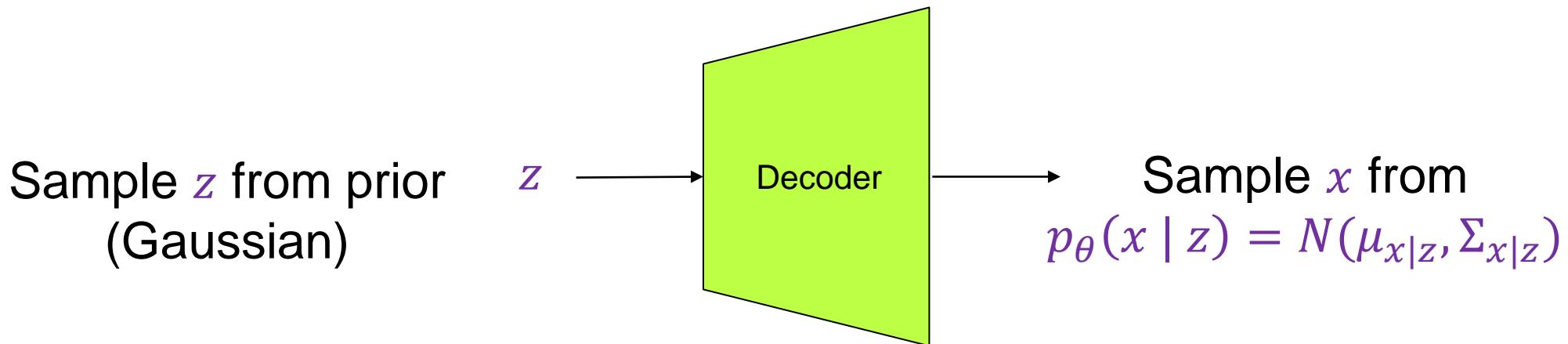
Variational autoencoders: Overview

- VAE makes its encoder not output an encoding vector of size n , rather, outputting two vectors of size n : a vector of means, μ , and another vector of standard deviations, σ .



Variational autoencoders: Overview

- At test time, discard encoder and use decoder to sample from
 $p_{\theta}(x | z) = N(\mu_{x|z}, \Sigma_{x|z})$



Variational autoencoders: Training

- Objective: maximize the *variational lower bound* on the data likelihood:

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x), p(z))$$

Variational autoencoders: Training

- Objective: maximize the *variational lower bound* on the data likelihood:

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) || p(z))$$

1. Run training point x through encoder to get a distribution over latent codes z

Variational autoencoders: Training

- Objective: maximize the *variational lower bound* on the data likelihood:

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x), p(z))$$

1. Run training point x through encoder to get a distribution over latent codes z
2. Encoder output should match the prior $p(z)$

Variational autoencoders: Training

- Objective: maximize the *variational lower bound* on the data likelihood:

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x), p(z))$$

1. Run training point x through encoder to get a distribution over latent codes z
2. Encoder output should match the prior $p(z)$
3. **Sample code z from encoder output**

Variational autoencoders: Training

- Objective: maximize the *variational lower bound* on the data likelihood:

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x), p(z))$$

1. Run training point x through encoder to get a distribution over latent codes z
2. Encoder output should match the prior $p(z)$
3. Sample code z from encoder output
4. Run sampled z through decoder to get a distribution over data samples

Variational autoencoders: Training

- Objective: maximize the *variational lower bound* on the data likelihood:

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x), p(z))$$

1. Run training point x through encoder to get a distribution over latent codes z
2. Encoder output should match the prior $p(z)$
3. Sample code z from encoder output
4. Run sampled z through decoder to get a distribution over data samples
5. Original input should be likely under the distribution output in (4)

Variational autoencoders: Training

- Objective: maximize the *variational lower bound* on the data likelihood:

$$\log p_\theta(x) \geq \underbrace{\mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)]}_{\text{Data likelihood}} - \underbrace{D_{KL}(q_\phi(z|x), p(z))}_{\text{Regularization}}$$

Variational autoencoders: Training

- Objective: maximize the *variational lower bound* on the data likelihood:

$$\log p_\theta(x) \geq \underbrace{\mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)]}_{\text{Data likelihood}} - \underbrace{D_{KL}(q_\phi(z|x), p(z))}_{\text{Regularization}}$$

- Objective for the entire dataset:

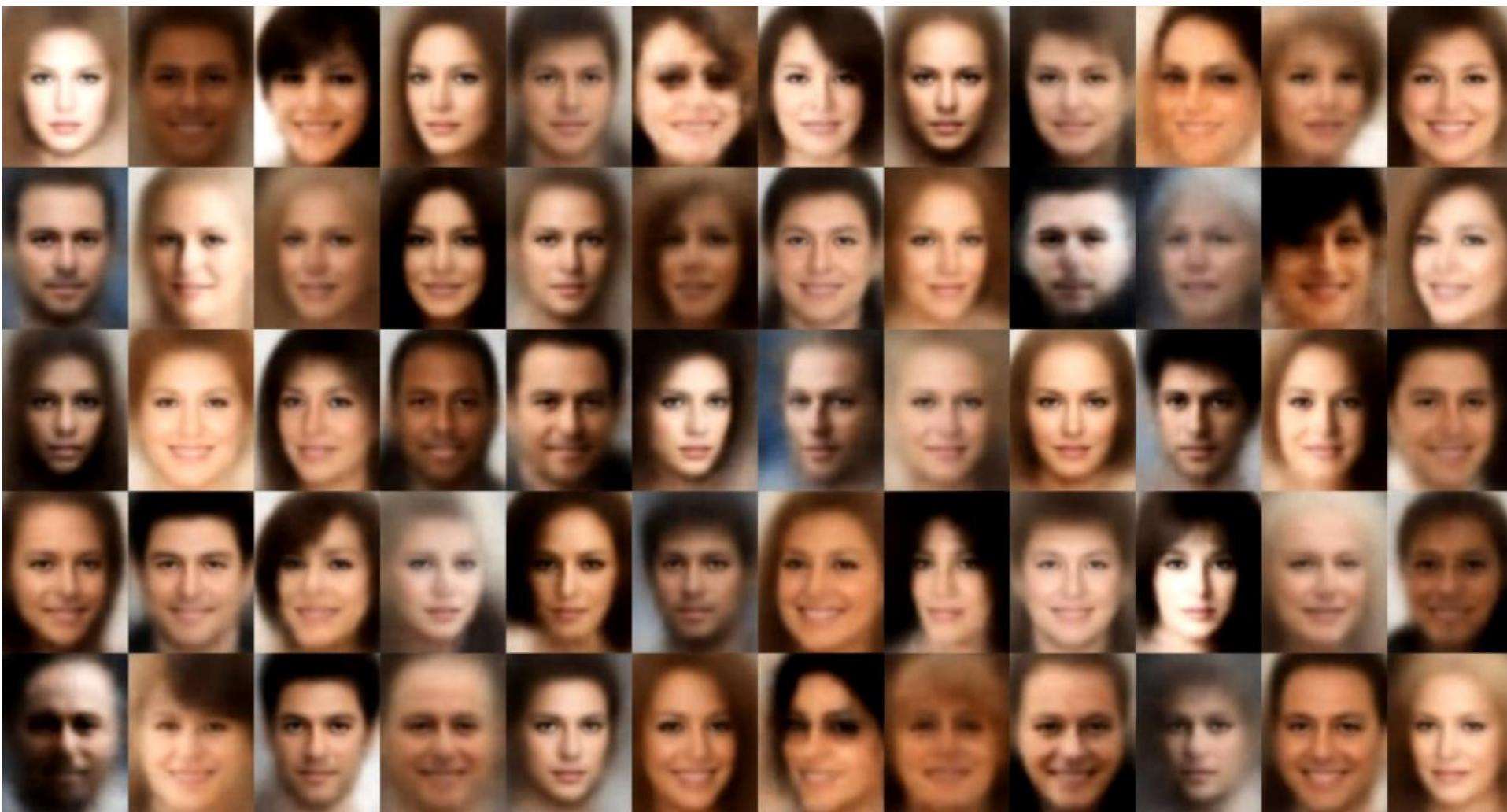
$$\mathbb{E}_{x \sim D} \left[\mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x), p(z)) \right]$$

Original results

- Learned 2D manifolds:



Variational autoencoders: Generating data

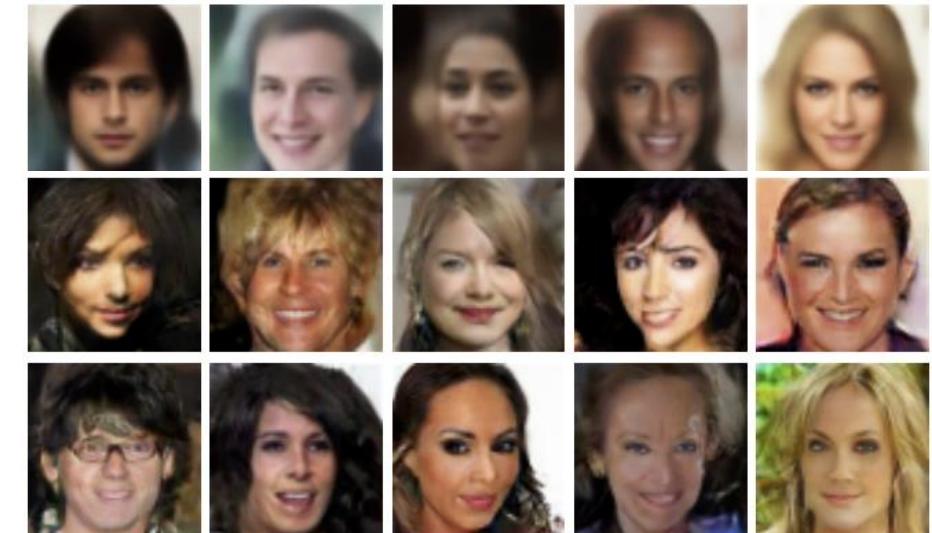
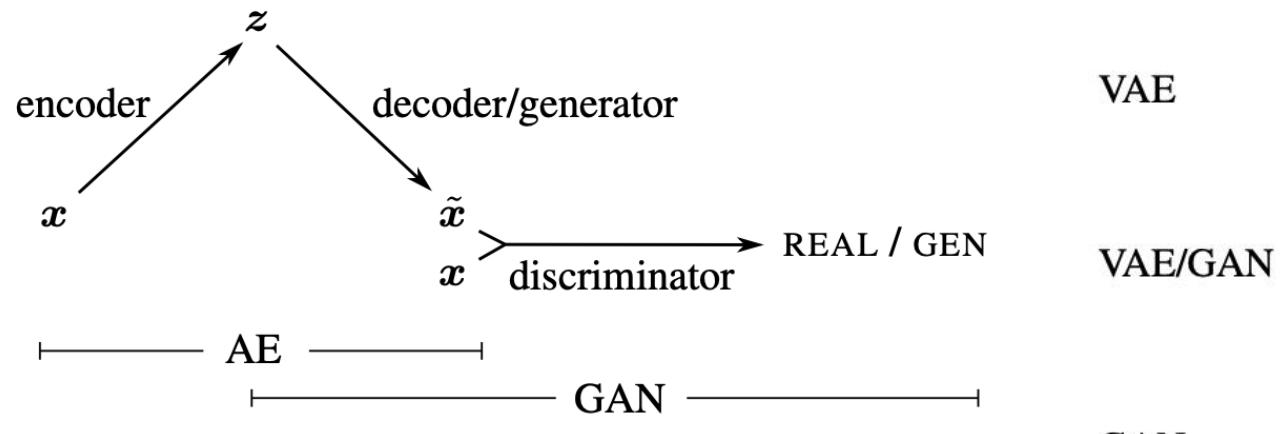


Basic VAE framework: Summary

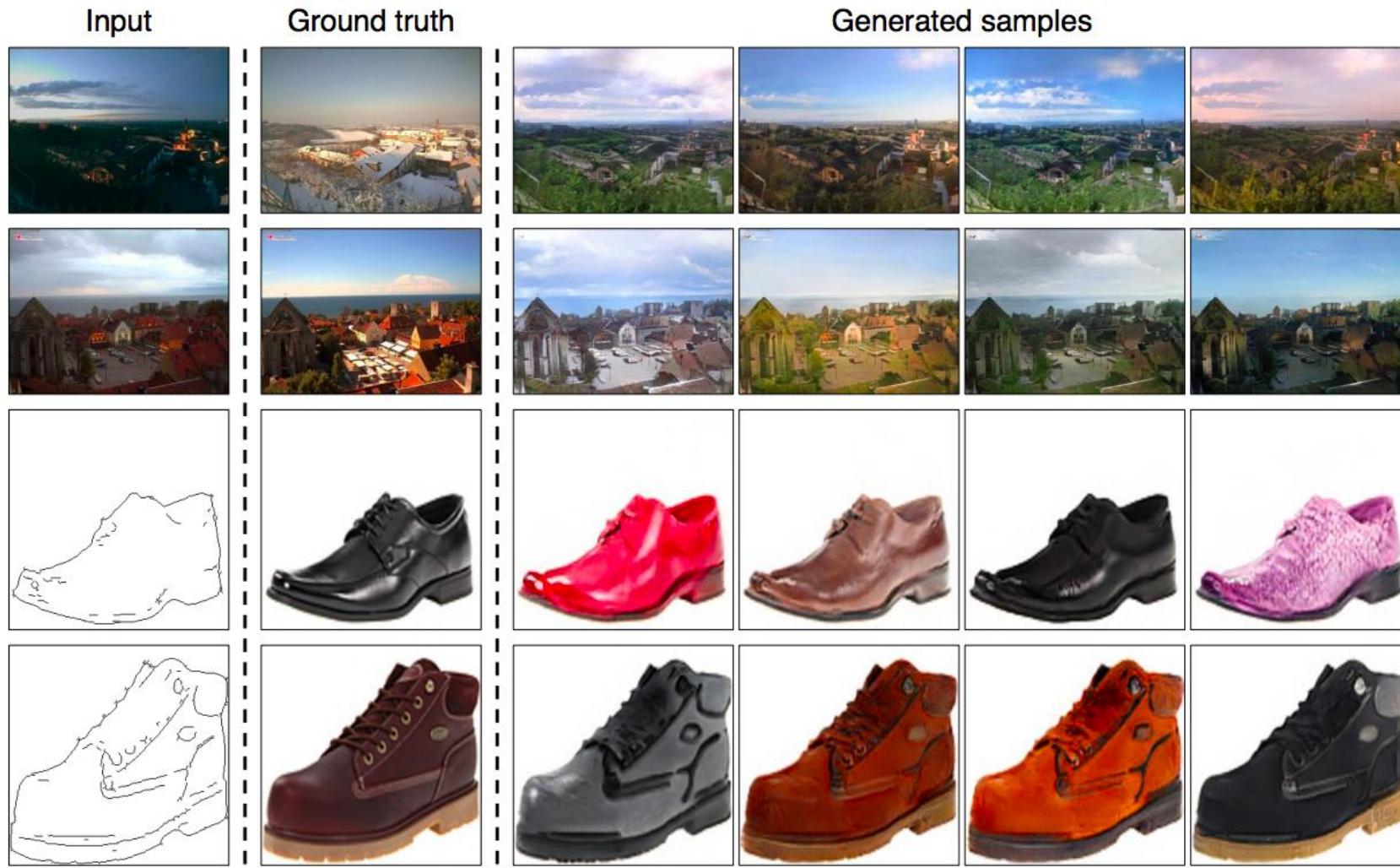
- Pros:
 - Principled mathematical formalism for generative models
 - Allows inference of code given image, can be useful for controlling the latent space
- Cons:
 - Samples blurrier and lower quality compared to GANs
- Active areas of research:
 - More powerful and flexible approximations for relevant probability distributions
 - Combining VAEs and GANs
 - Incorporating structure in latent variables, e.g., hierarchical or categorical distributions

Combining VAEs and GANs

- Define decoder probability model $p_{\theta}(x|z)$ not in terms of reconstruction errors in pixel space, but in terms of errors in discriminator feature space



Combining VAEs and GANs: BicycleGANs

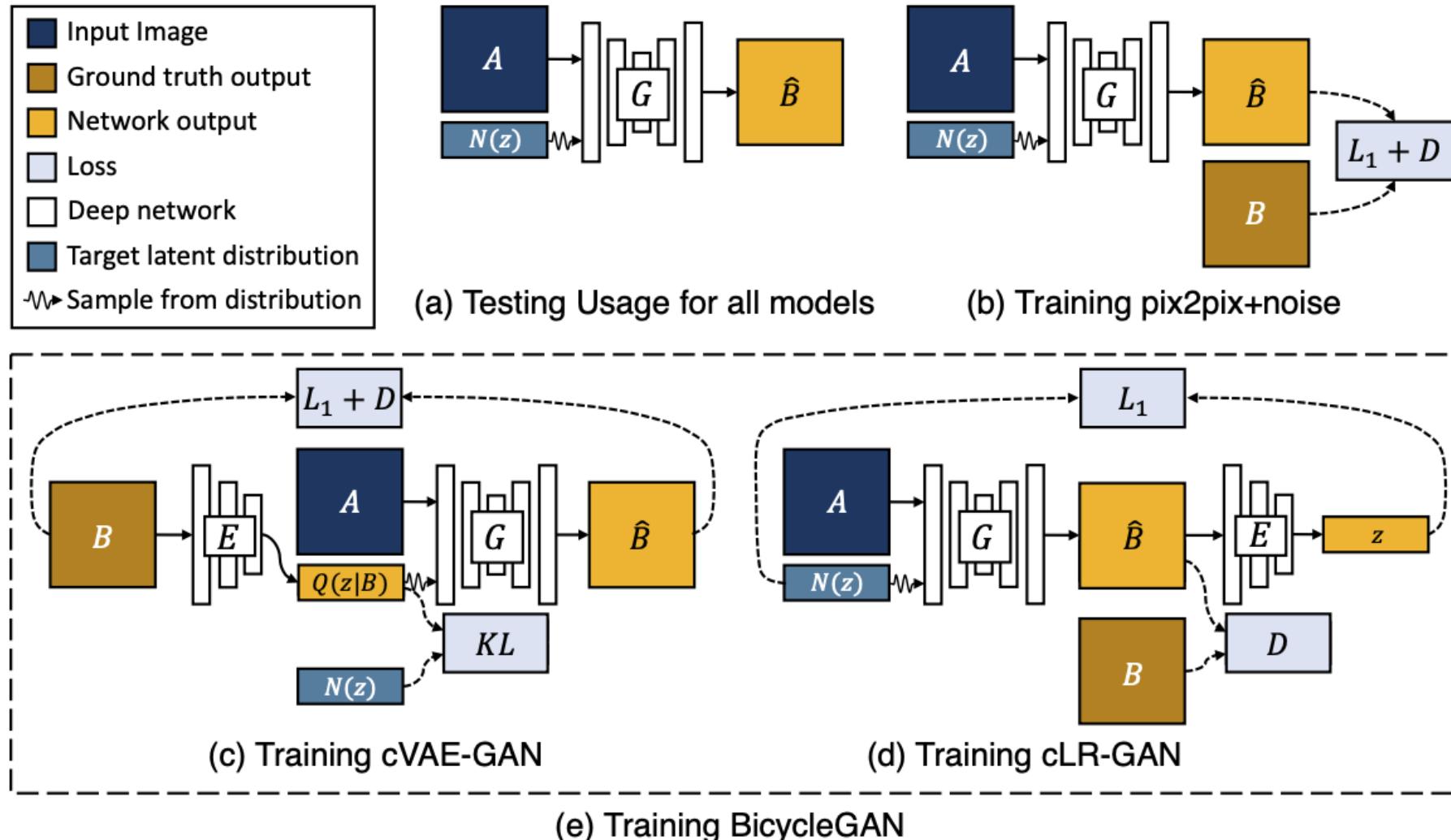


J.Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, E. Shechtman,
[Toward Multimodal Image-to-Image Translation](#), NIPS 2017

Combining VAEs and GANs: BicycleGANs

- Key ideas:
 - Image-to-image translation is a *one-to-many* problem. Need to model conditional distribution of output given input parametrized by z
 - To prevent mode collapse (or many-to-one mapping from z to output), need to encourage the mapping between output and latent code to be invertible
 - Propose BicycleGAN framework to simultaneously learn mappings in both directions

Combining VAEs and GANs: BicycleGANs

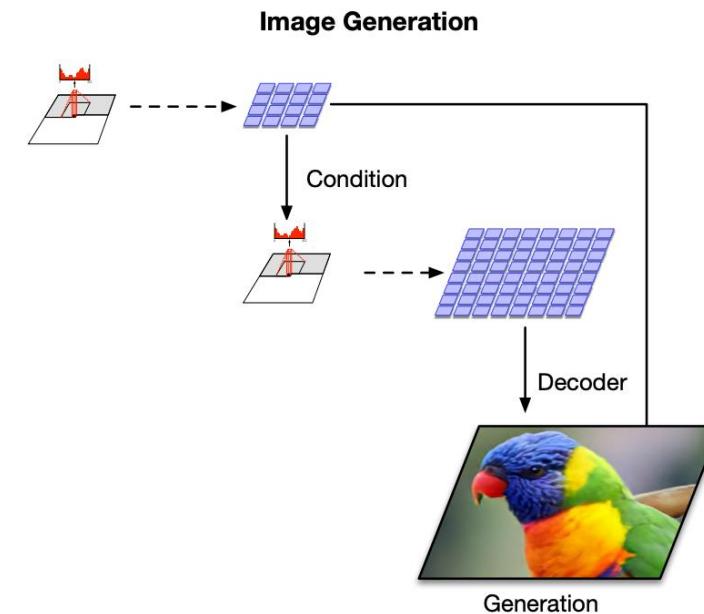
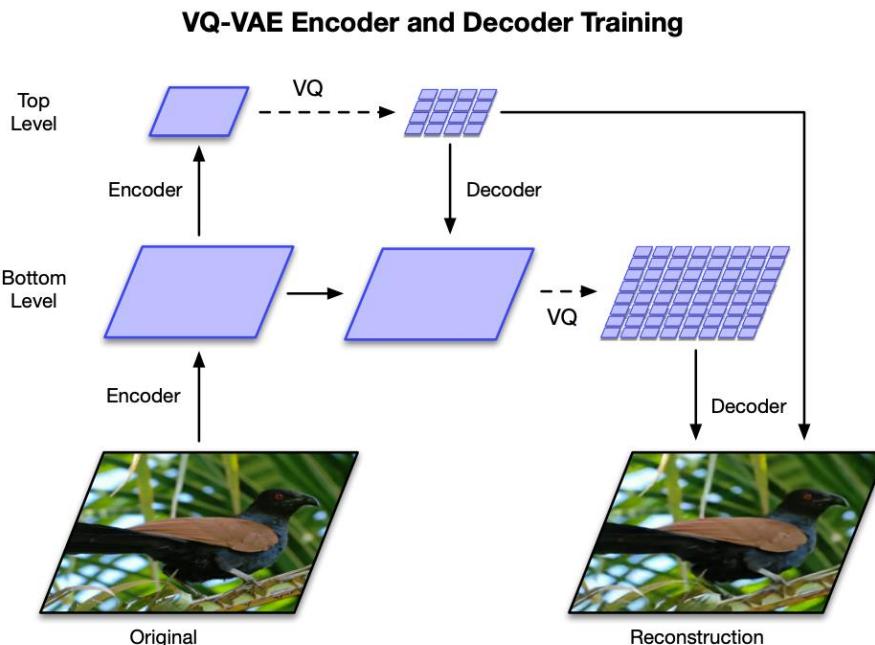


Generating better samples: VQ-VAE-2

- Combining VAE and autoregressive models:

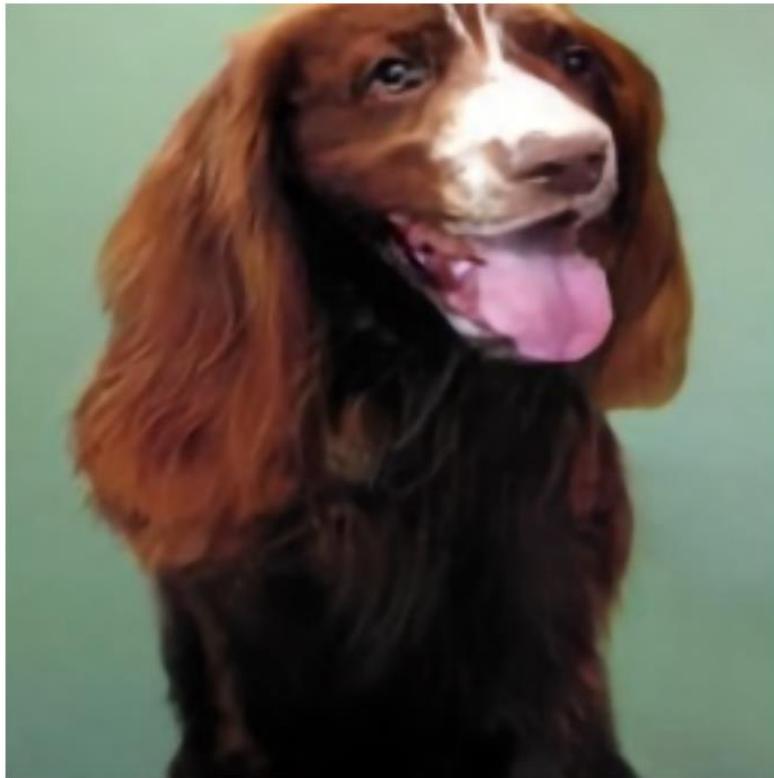
Train a VAE-like model to generate multiscale grids of latent codes

Use a multiscale autoregressive model (PixelCNN) to sample in latent code space



Generating better samples: VQ-VAE-2

- 256 x 256 class-conditional samples, trained on ImageNet:



Generating better samples: VQ-VAE-2

- 256 x 256 class-conditional samples, trained on ImageNet:

Redshank



Pekinese



Papillon



Drake

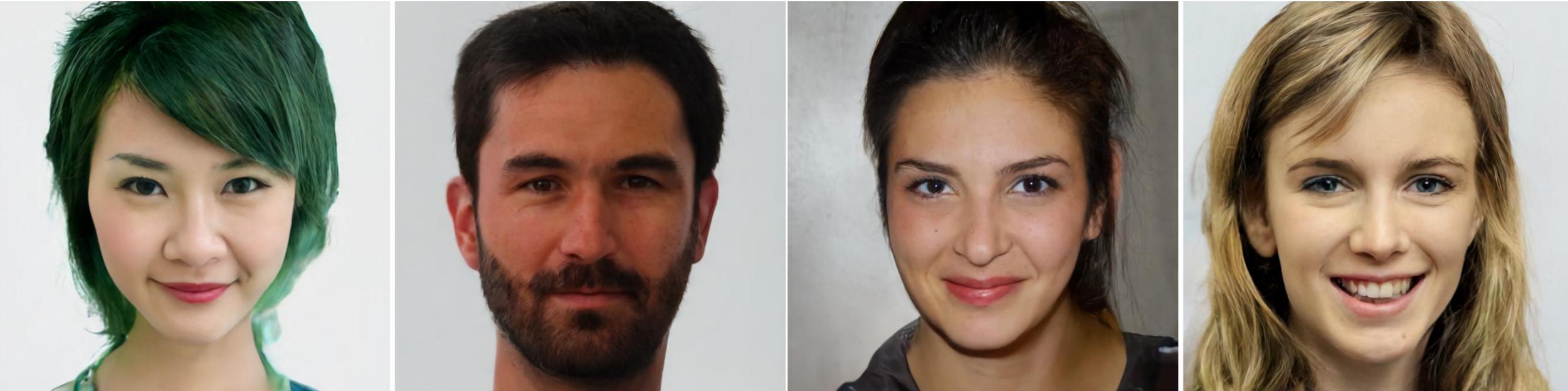


Spotted Salamander



Generating better samples: VQ-VAE-2

- 1024 x 1024 generated faces, trained on FFHQ:



Generating better samples: Hierarchical VAE

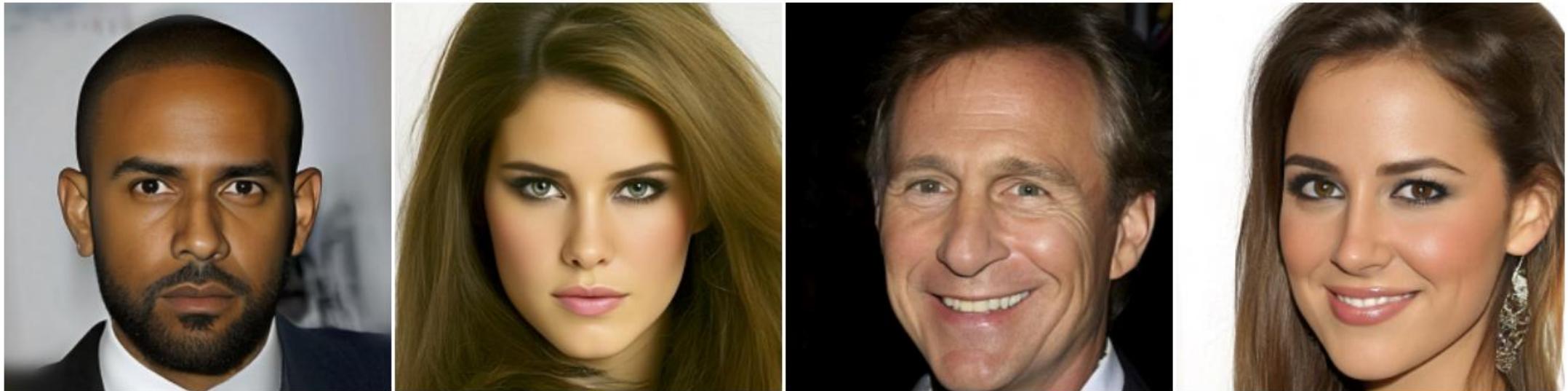


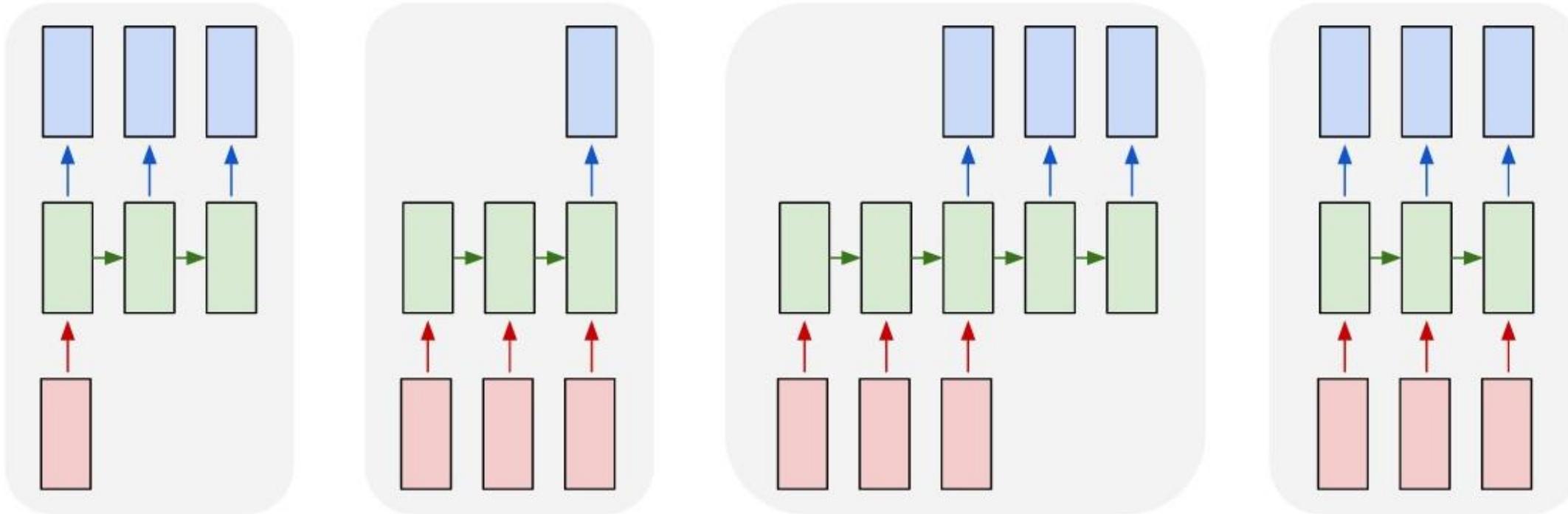
Figure 1: 256×256-pixel samples generated by NVAE, trained on CelebA HQ [28].

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Recurrent neural networks



Outline

- Examples of sequential prediction tasks
- Common recurrent units
 - Vanilla RNN unit (and how to train it)
 - Long Short-Term Memory (LSTM)
 - Gated Recurrent Unit (GRU)
- Recurrent network architectures
- Applications in (a bit) more detail
 - Sequence classification
 - Language modeling
 - Image captioning
 - Machine translation

Sequential prediction tasks

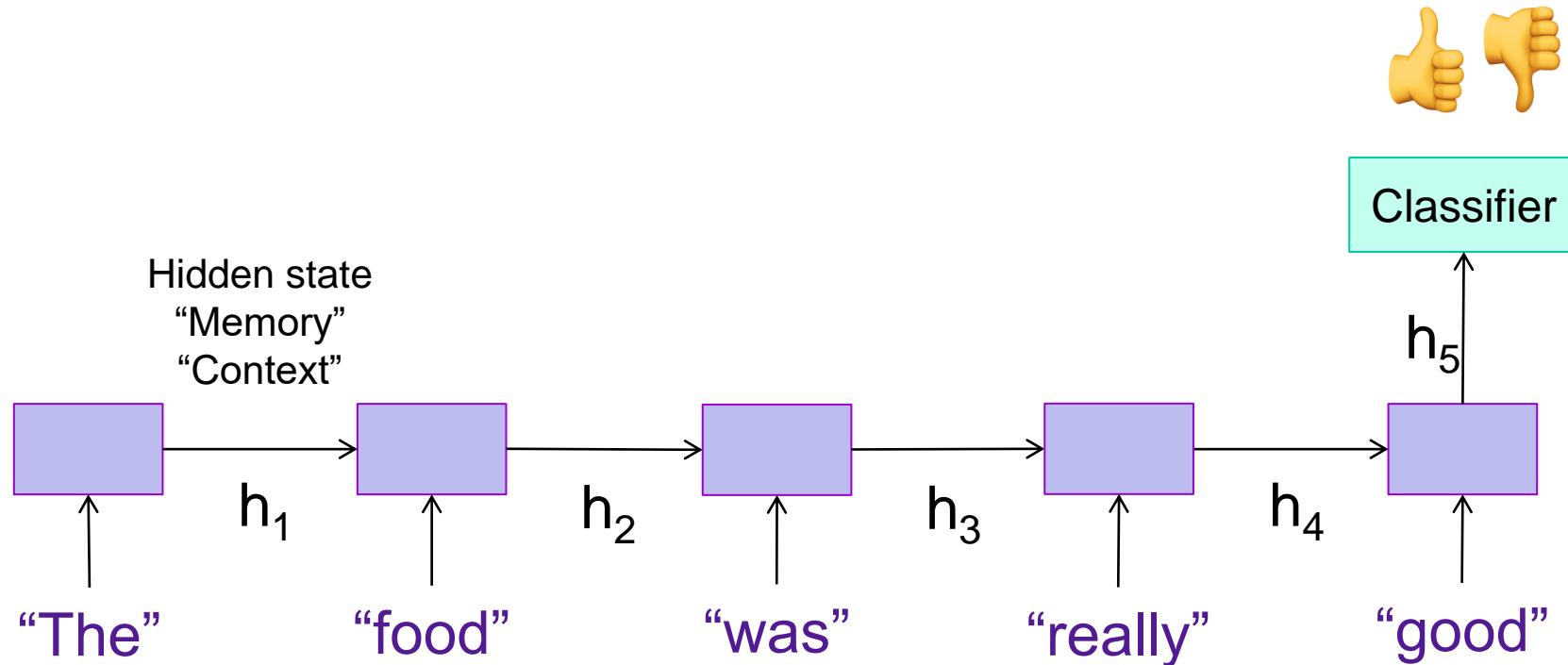
- So far, we focused mainly on prediction problems with fixed-size inputs and outputs
- But what if the input and/or output is a variable-length sequence?

Example 1: Sentiment classification

- Goal: classify a text sequence (e.g., restaurant, movie or product review, Tweet) as having positive or negative sentiment
 - “The food was really good”
 - “The vacuum cleaner broke within two weeks”
 - “The movie had slow parts, but overall was worth watching”
- What makes this problem challenging?
- What feature representation or predictor structure can we use for this problem?

Example 1: Sentiment classification

- Recurrent model:



Example 2: Text generation

- Sample from the distribution of a given text corpus
(also known as language modeling)

Follow

DeepDrumpf
@DeepDrumpf

I'm a Neural Network trained on Trump's transcripts. Priming text in []s. Donate (gofundme.com/deepdrumpf) to interact! Created by @hayesbh.

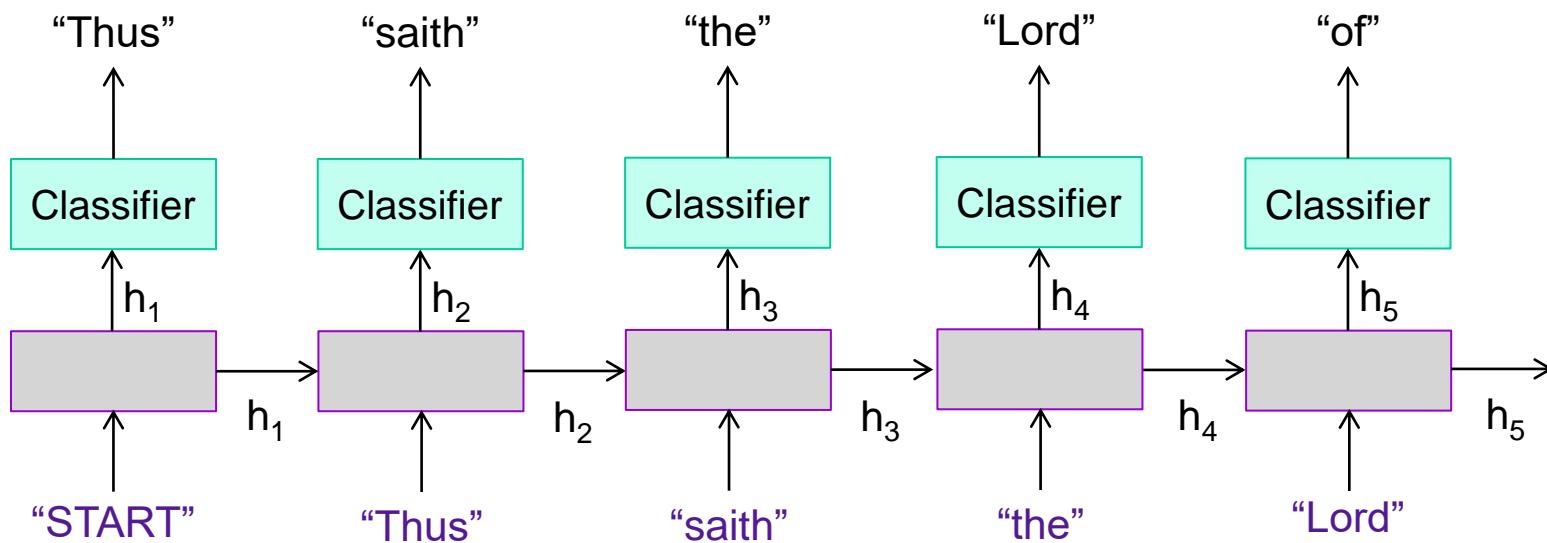
Joined March 2016

7 Following 24.6K Followers

Tweets	Tweets & replies	Media	Likes
 DeepDrumpf @DeepDrumpf · May 31, 2017 [Despite the negative press #covfefe] look at what's going on. They shoot media. Usually that's a bad sign of things to come. 6 38 124	 DeepDrumpf @DeepDrumpf · Apr 7, 2017 When I have to build a hotel, we're bombing the hell out of them. Lots of money. To those suffering, I say vote for Donald. #SyriaStrikes 1 71 173	 DeepDrumpf @DeepDrumpf · Mar 20, 2017 Replies to @Thomas1774Paine There will be no amnesty. It is going to pass because the people are going to be gone. I'm giving a mandate. #ComeyHearing @Thomas1774Paine	...

Example 2: Text generation

- Sample from the distribution of a given text corpus (also known as language modeling)
- Can be done one character or one word at a time:



Example 3: Image caption generation



A cat sitting on a suitcase on the floor



A cat is sitting on a tree branch



A dog is running in the grass with a frisbee



A white teddy bear sitting in the grass



Two people walking on the beach with surfboards



A tennis player in action on the court

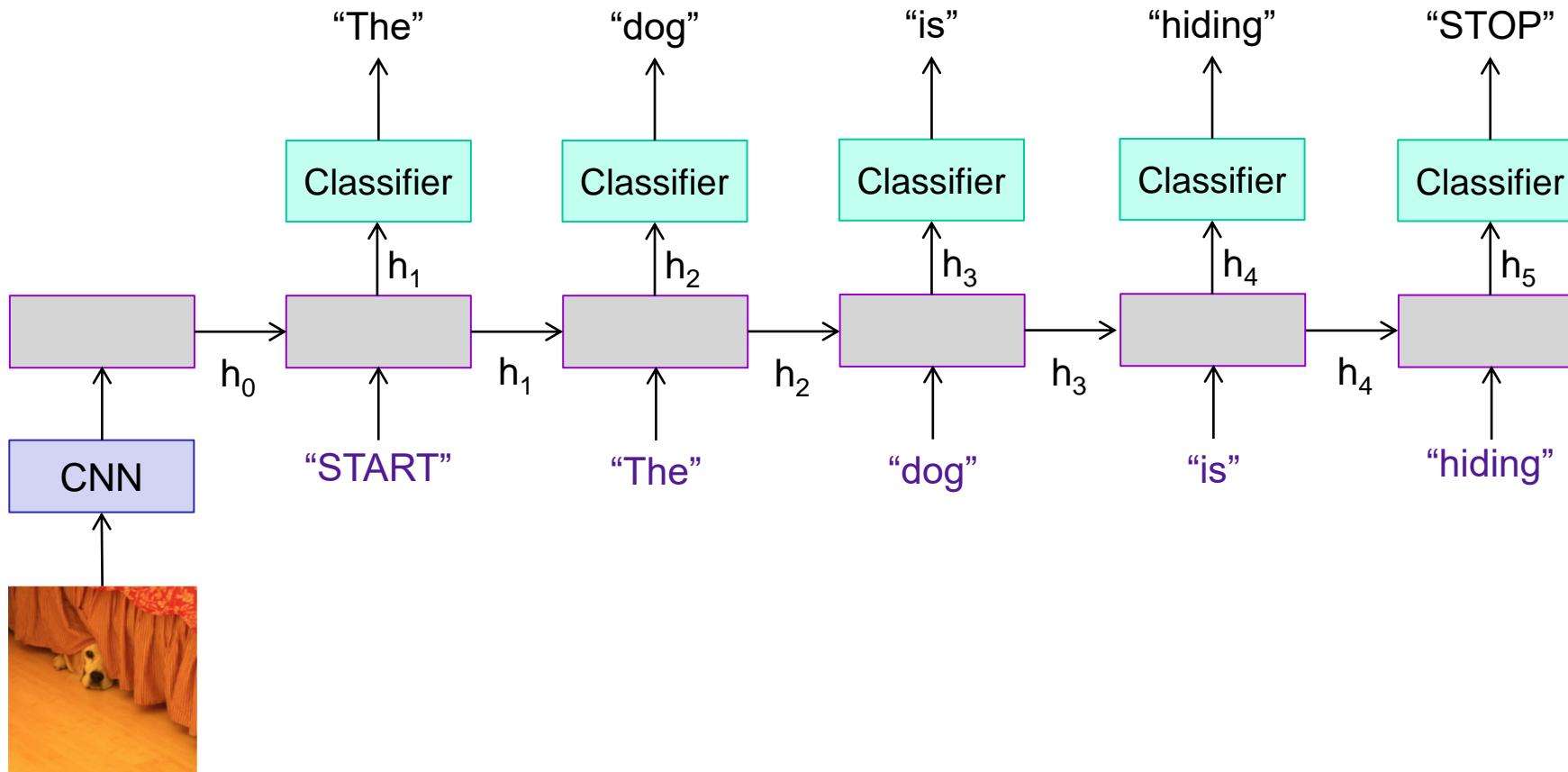


Two giraffes standing in a grassy field



A man riding a dirt bike on a dirt track

Example 3: Image caption generation



Example 4: Machine translation

The screenshot shows the Google Translate interface with two columns of text. The left column is in French and the right column is in English. Both columns contain a poem by Charles Baudelaire.

Left Column (French):

Correspondances
La Nature est un temple où de vivants piliers
Laissent parfois sortir de confuses paroles;
L'homme y passe à travers des forêts de symboles
Qui l'observent avec des regards familiers.
Comme de longs échos qui de loin se confondent
Dans une ténèbreuse et profonde unité,
Vaste comme la nuit et comme la clarté,
Les parfums, les couleurs et les sons se répondent.
Il est des parfums frais comme des chairs d'enfants,
Doux comme les hautbois, verts comme les prairies,
— Et d'autres, corrompus, riches et triomphants,
Ayant l'expansion des choses infinies,
Comme l'ambre, le musc, le benzoin et l'encens,
Qui chantent les transports de l'esprit et des sens.
— Charles Baudelaire

Right Column (English):

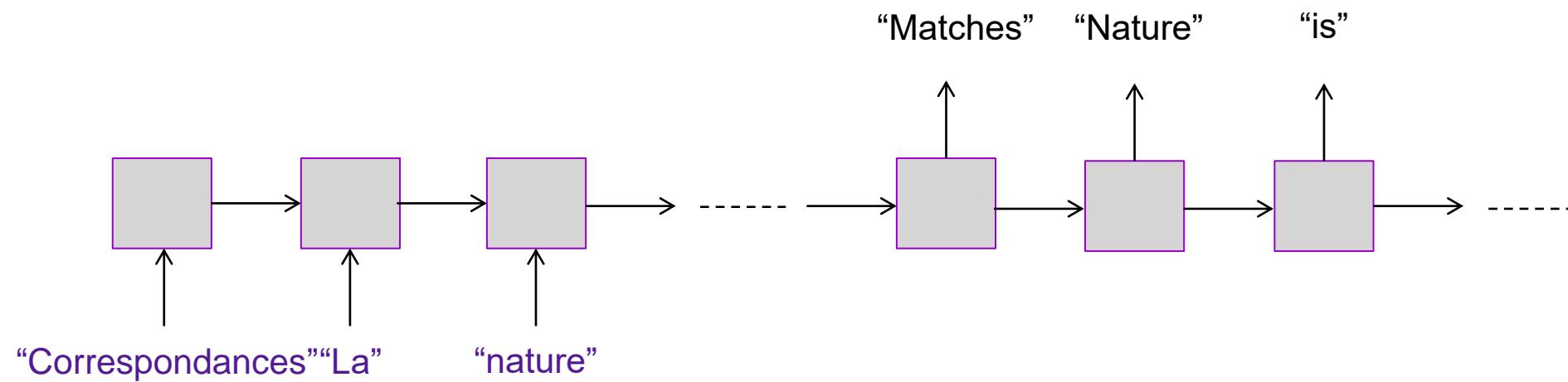
Matches
Nature is a temple where living pillars
Sometimes let out confused words;
Man goes through symbol forests
Which observe him with familiar eyes.
Like long echoes that by far merge
In a dark and deep unity,
As vast as the night and as clarity,
The perfumes, the colors and the sounds answer each other.
There are fresh perfumes like children's flesh,
Sweet like oboes, green like meadows,
- And others, corrupt, rich and triumphant,
Having the expansion of infinite things,
Like amber, musk, benzoin and incense,
Who sing the transports of the mind and the senses.
- Charles Baudelaire

At the bottom of the interface, there are icons for microphone, keyboard, and a progress bar indicating 693/5000 characters have been typed.

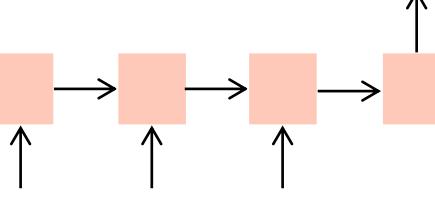
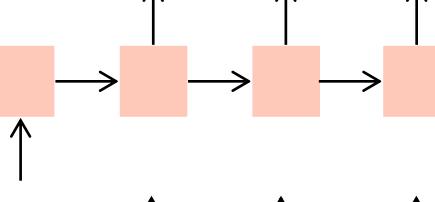
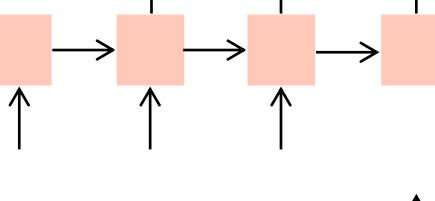
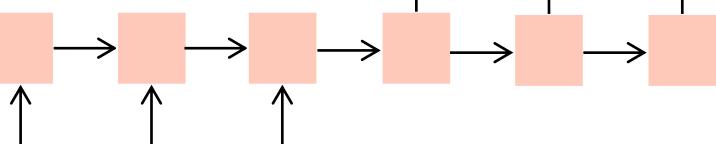
<https://translate.google.com/>

Example 4: Machine translation

- Multiple input – multiple output (or sequence to sequence) scenario:



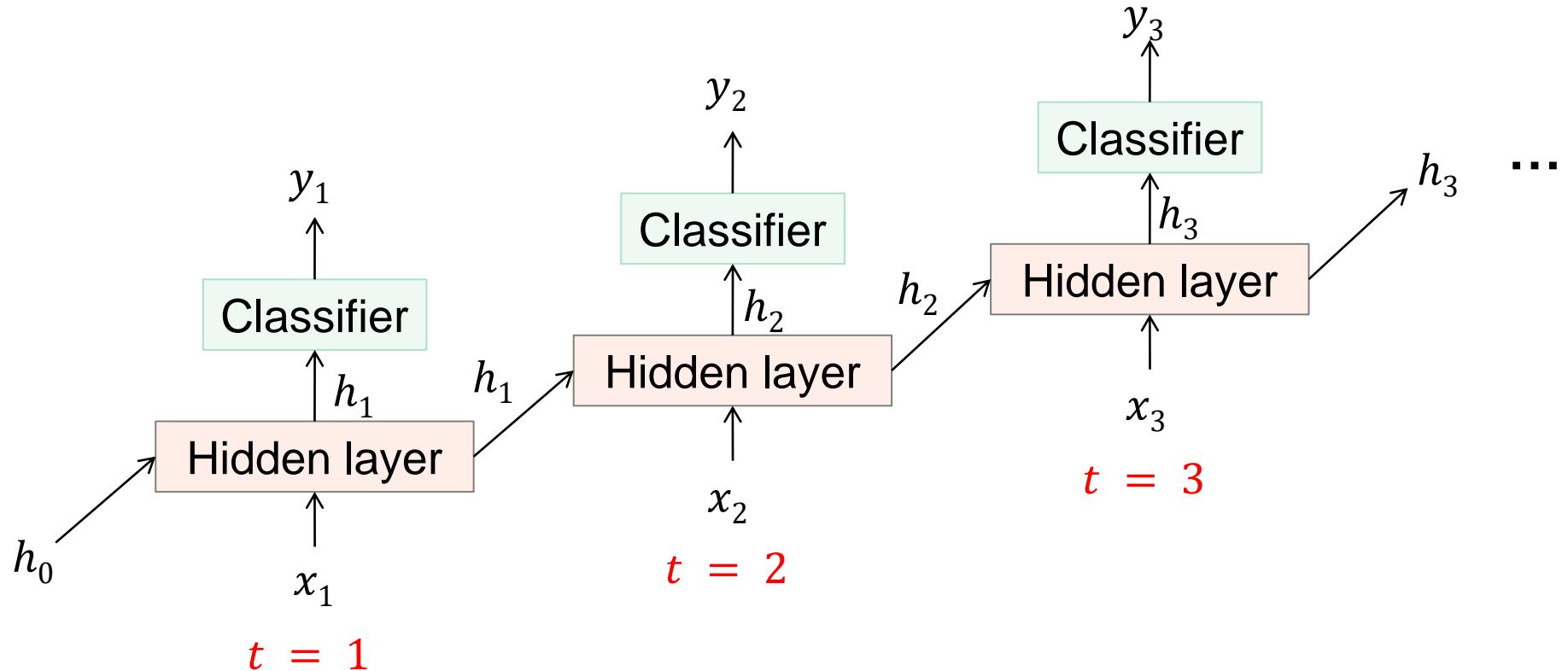
Summary: Input-output scenarios

Single - Single		Feed-forward Network
Multiple - Single		Sequence Classification
Single - Multiple		Sequence generation, captioning
Multiple - Multiple		Sequence generation, captioning
Multiple - Multiple		Translation

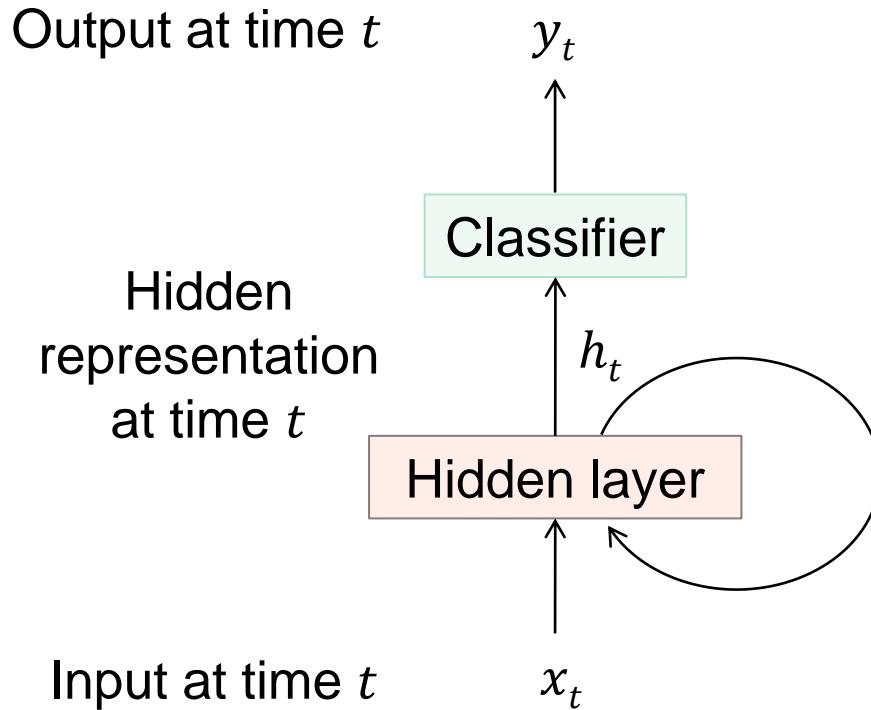
Outline

- Examples of sequential prediction tasks
- Common recurrent units
 - Vanilla RNN unit
 - Long Short-Term Memory (LSTM)
 - Gated Recurrent Unit (GRU)

Recurrent unit



Recurrent unit

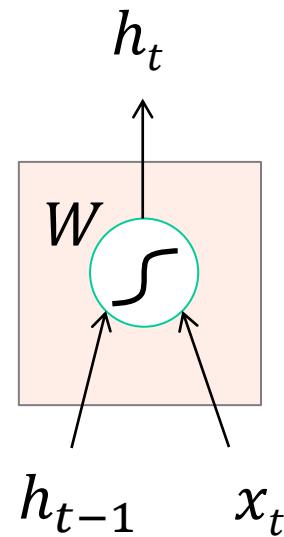


Recurrence:

$$h_t = f_W(x_t, h_{t-1})$$

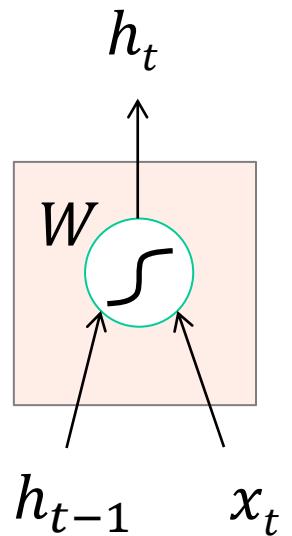
new state function of W input at time t old state

Vanilla RNN cell

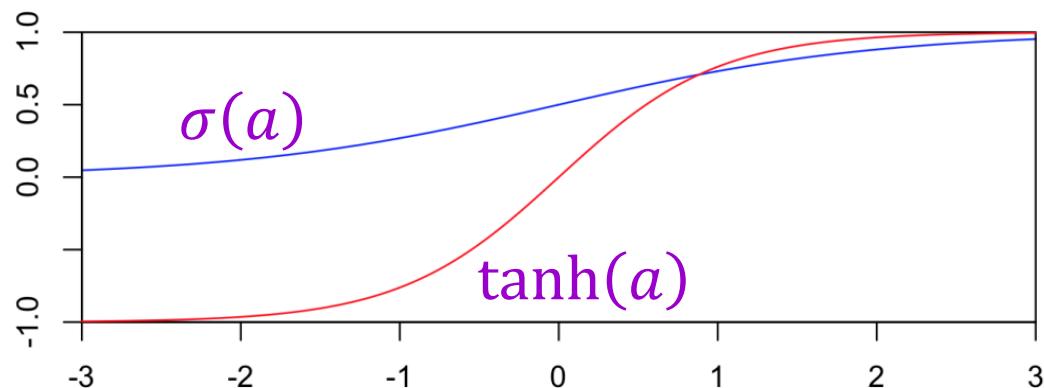


$$\begin{aligned} h_t &= f_W(x_t, h_{t-1}) \\ &= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \end{aligned}$$

Vanilla RNN cell

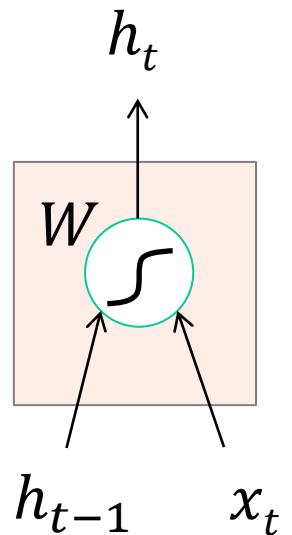


$$\begin{aligned} h_t &= f_W(x_t, h_{t-1}) \\ &= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \end{aligned}$$

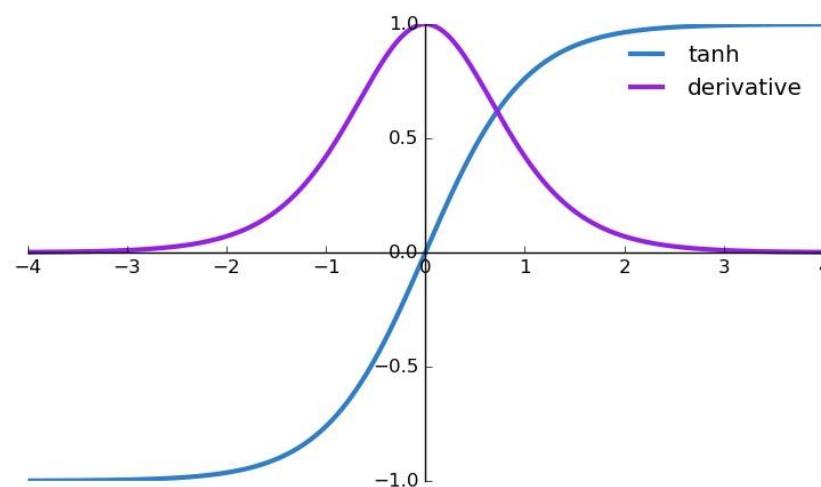


$$\begin{aligned} \tanh(a) &= \frac{e^a - e^{-a}}{e^a + e^{-a}} \\ &= 2\sigma(2a) - 1 \end{aligned}$$

Vanilla RNN cell

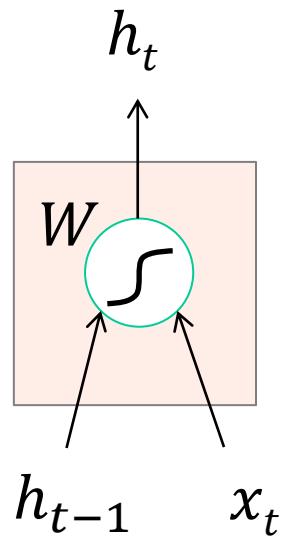


$$\begin{aligned} h_t &= f_W(x_t, h_{t-1}) \\ &= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \end{aligned}$$

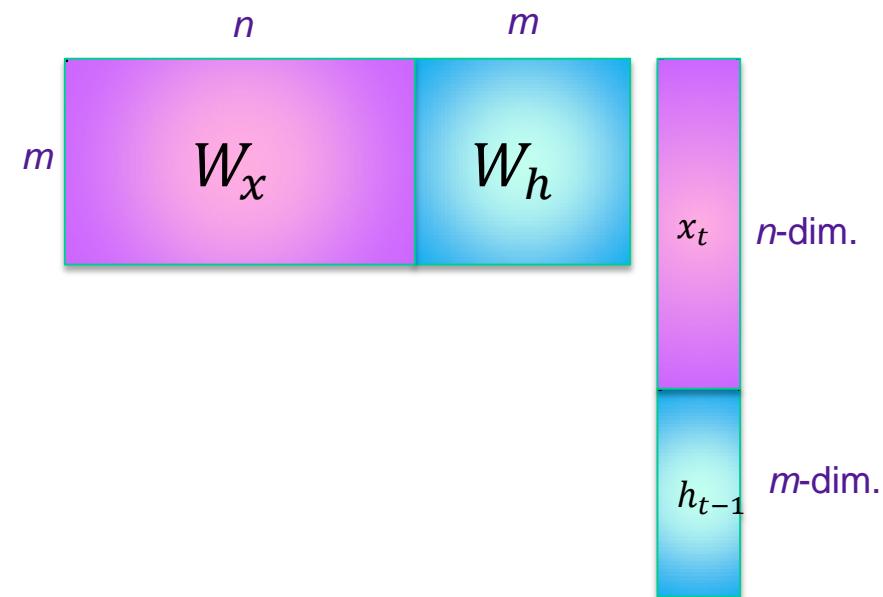


$$\frac{d}{da} \tanh(a) = 1 - \tanh^2(a)$$

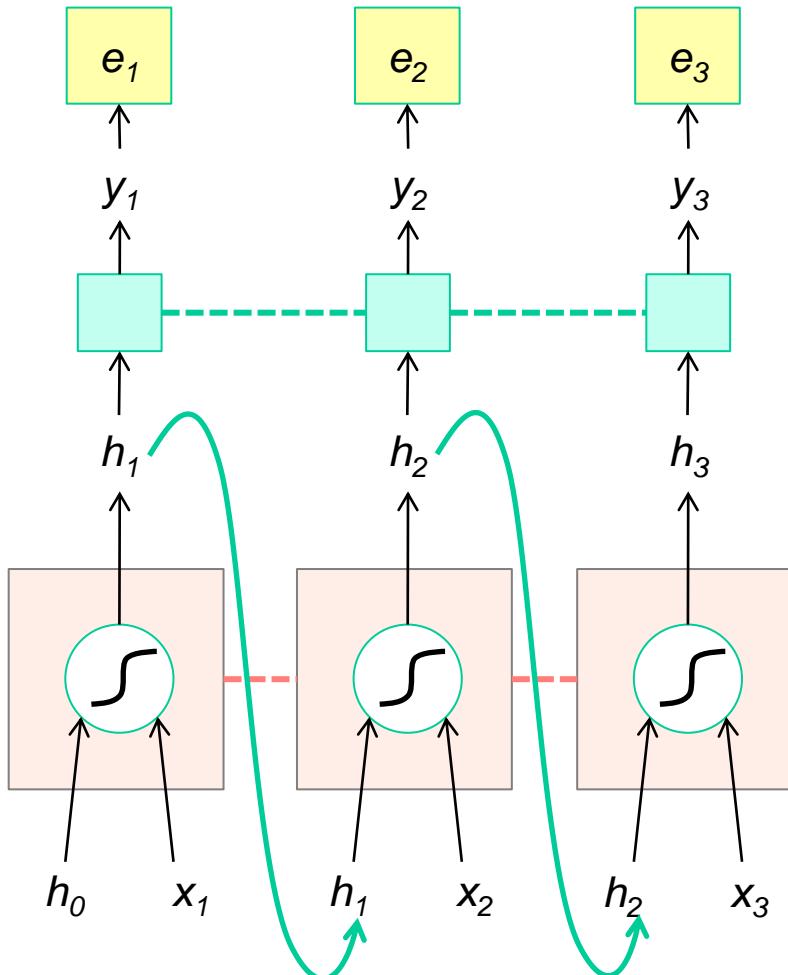
Vanilla RNN cell



$$\begin{aligned} h_t &= f_W(x_t, h_{t-1}) \\ &= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \\ &= \tanh(W_x x_t + W_h h_{t-1}) \end{aligned}$$



RNN forward pass



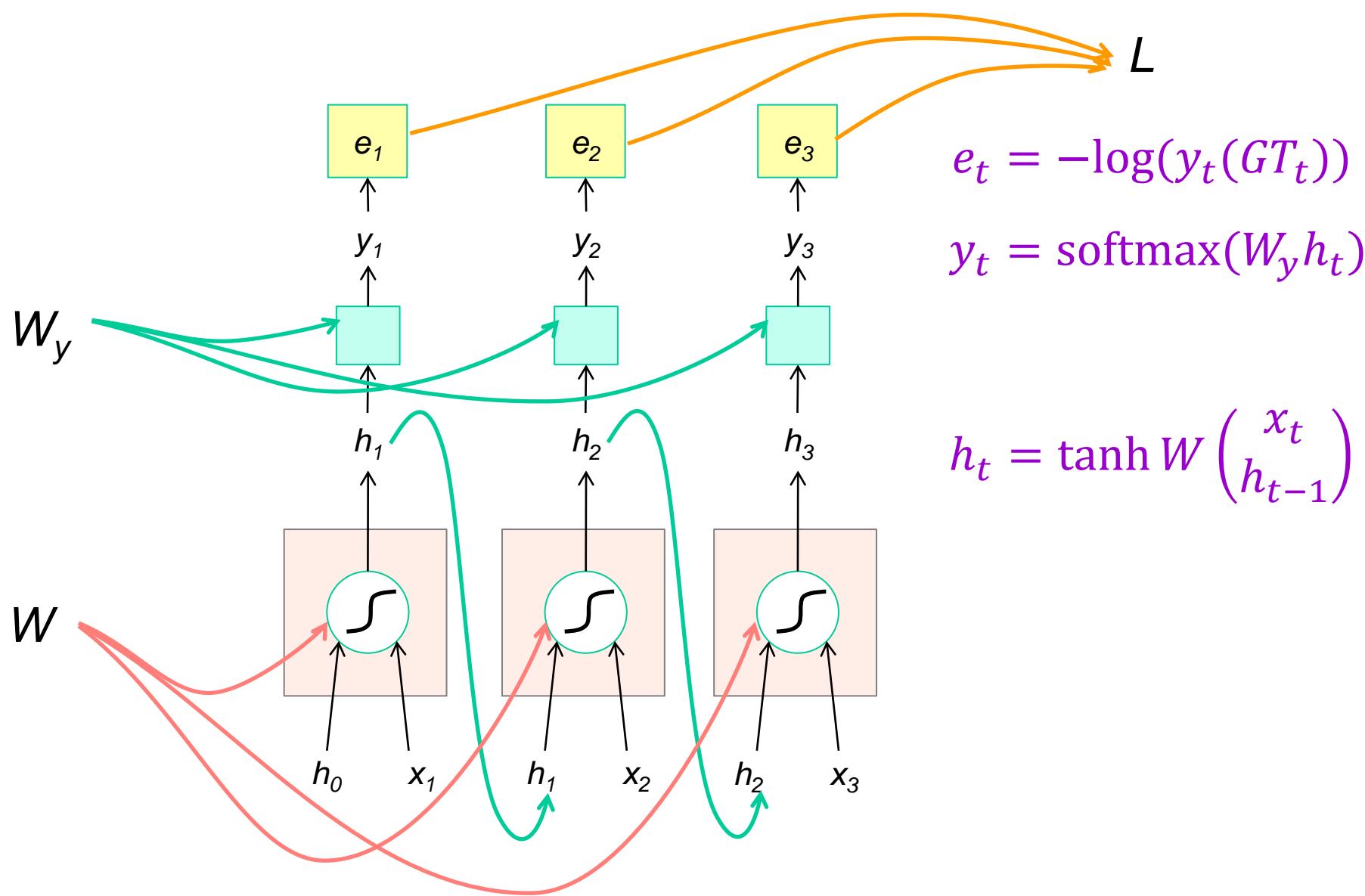
$$e_t = -\log(y_t(GT_t))$$

$$y_t = \text{softmax}(W_y h_t)$$

$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

----- shared weights

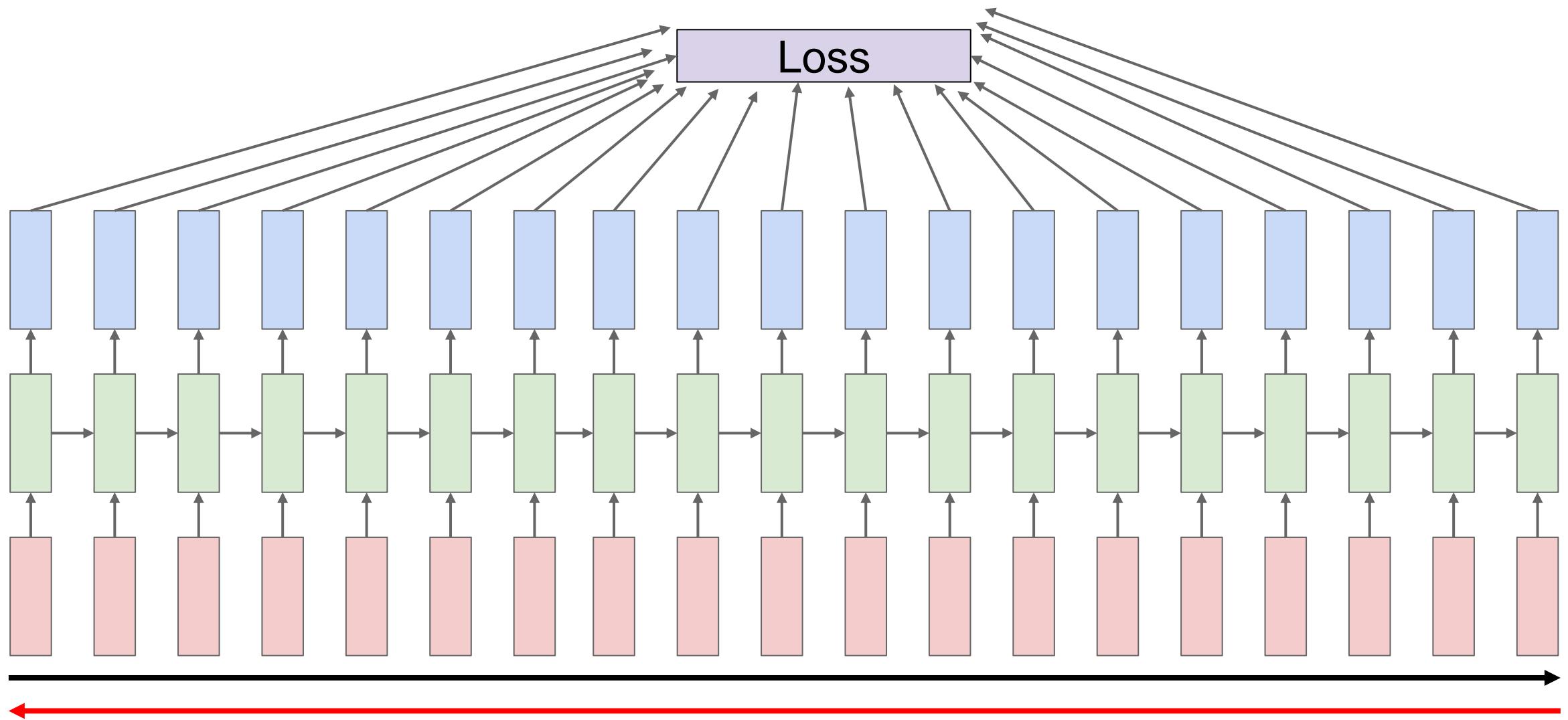
RNN forward pass: Computation graph



Training: Backpropagation through time (BPTT)

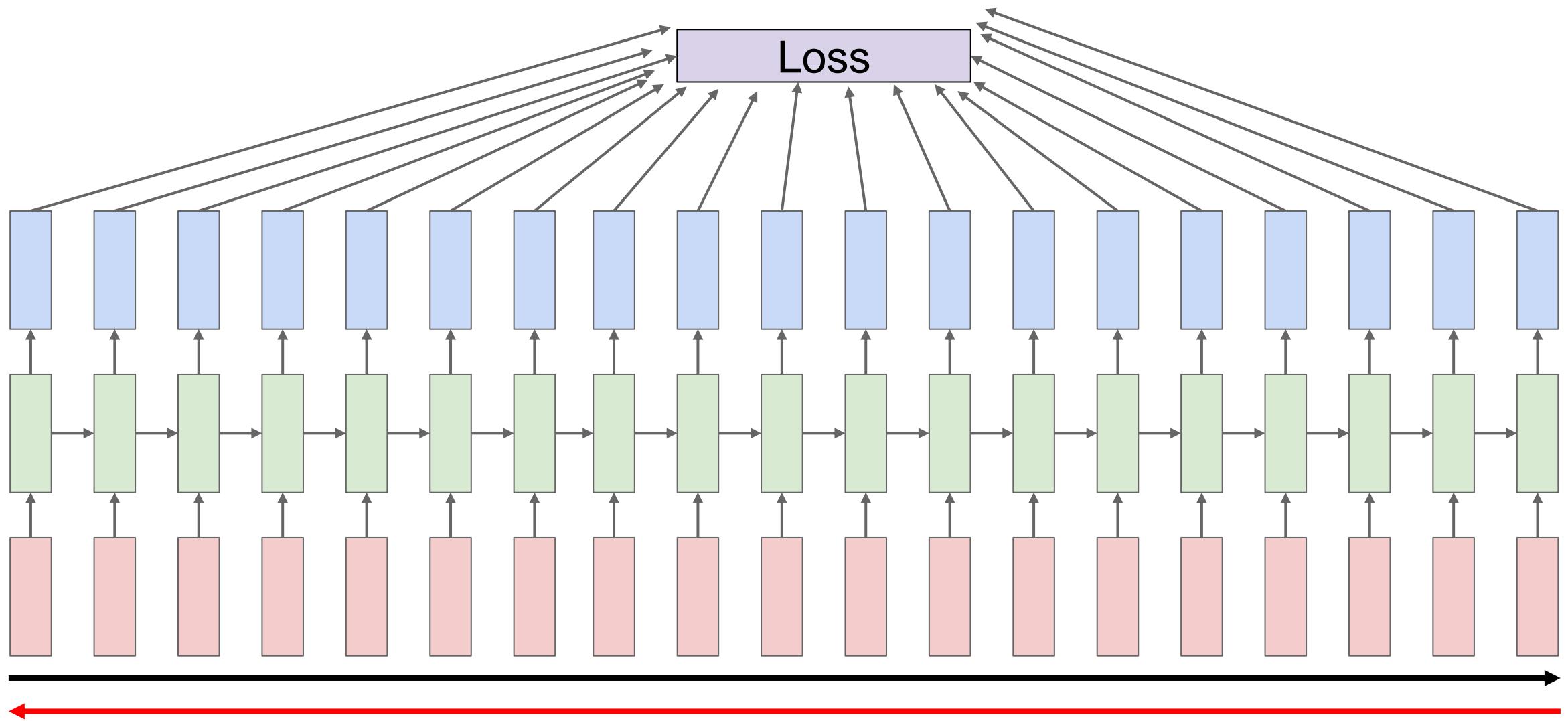
- The unfolded network (used during forward pass) is treated as one big feed-forward network that accepts the whole time series as input
- The weight updates are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights

Backpropagation through time



Forward through entire sequence to compute loss, then backward to compute gradient

Backpropagation through time



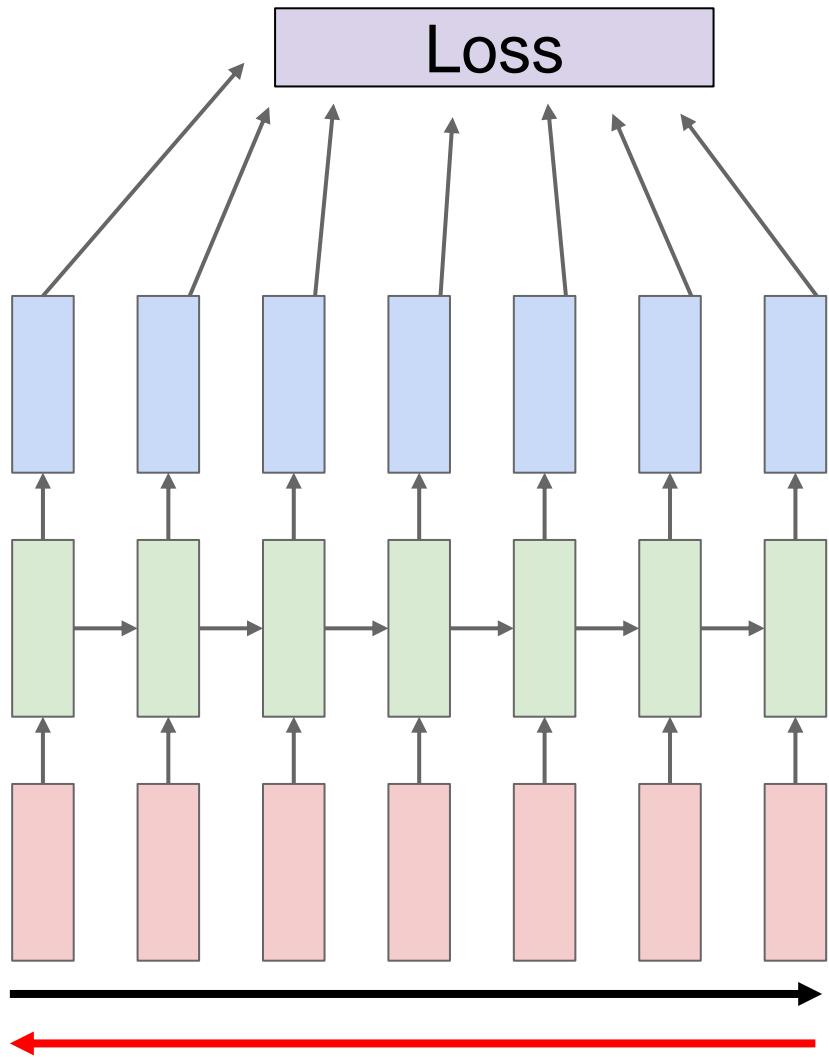
Problem: Takes a lot of memory for long sequences!

Training: Backpropagation through time (BPTT)

- The unfolded network (used during forward pass) is treated as one big feed-forward network that accepts the whole time series as input
- The weight updates are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights
- In practice, *truncated* BPTT is used: run the RNN forward k_1 time steps, propagate backward for k_2 time steps

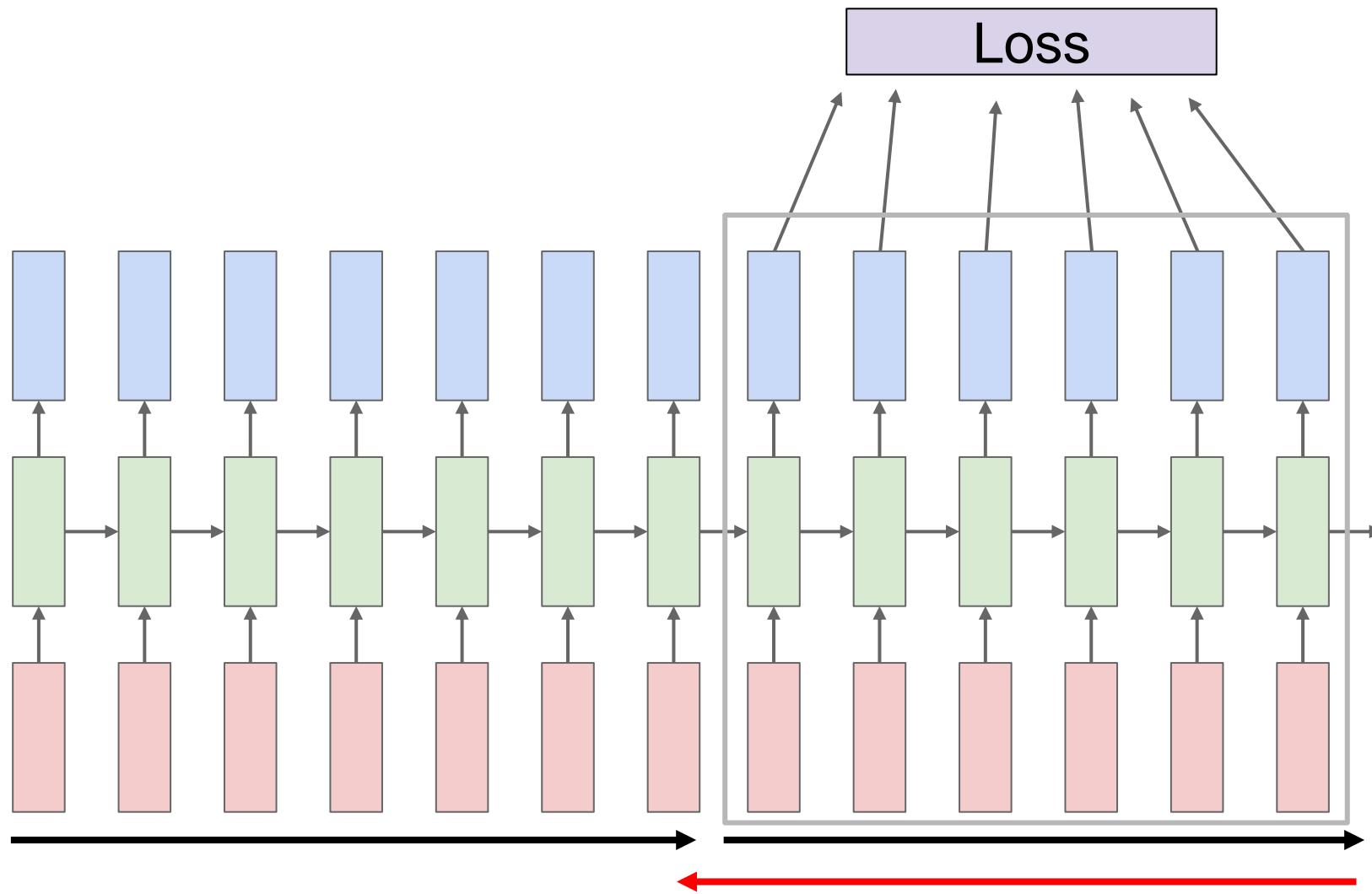
<https://machinelearningmastery.com/gentle-introduction-backpropagation-time/>
http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf

Truncated backpropagation through time



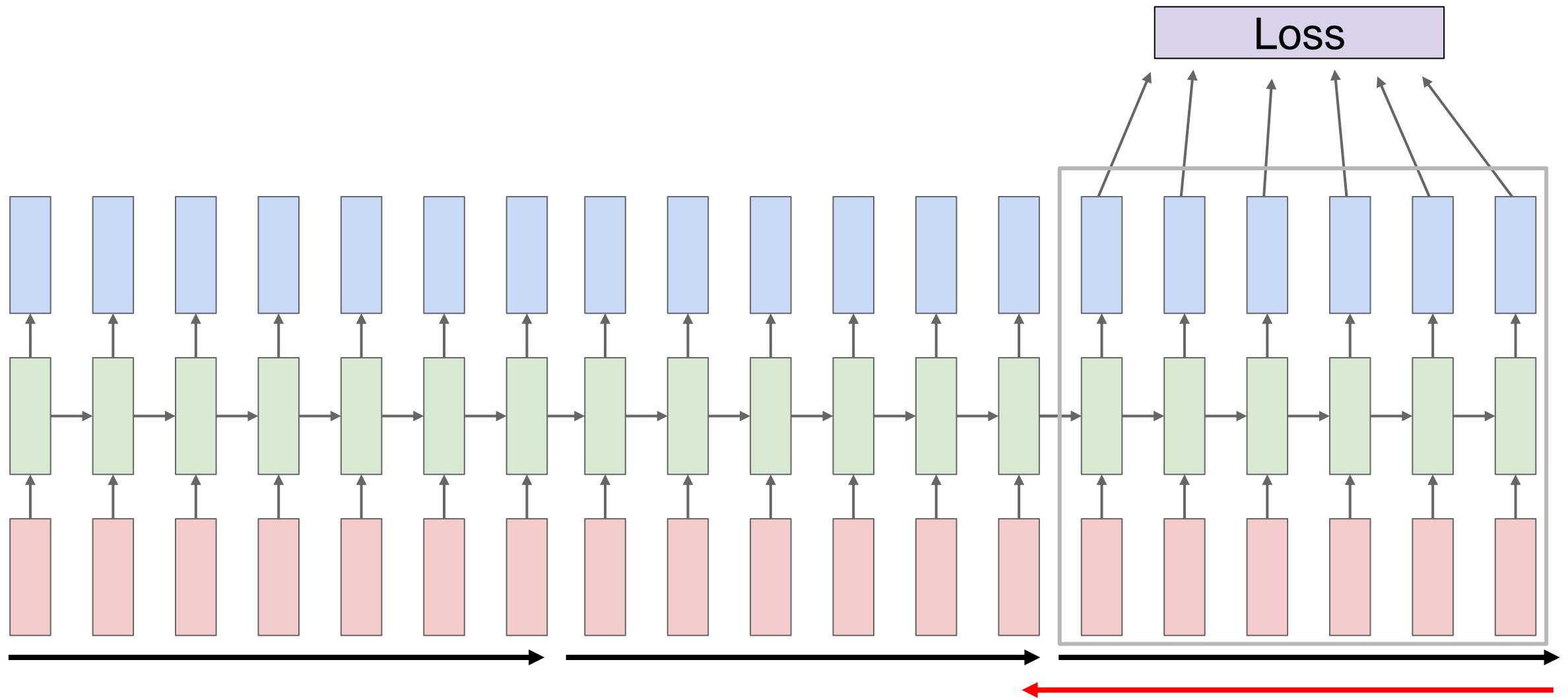
Run forward and backward
through chunks of the sequence
instead of whole sequence

Truncated backpropagation through time

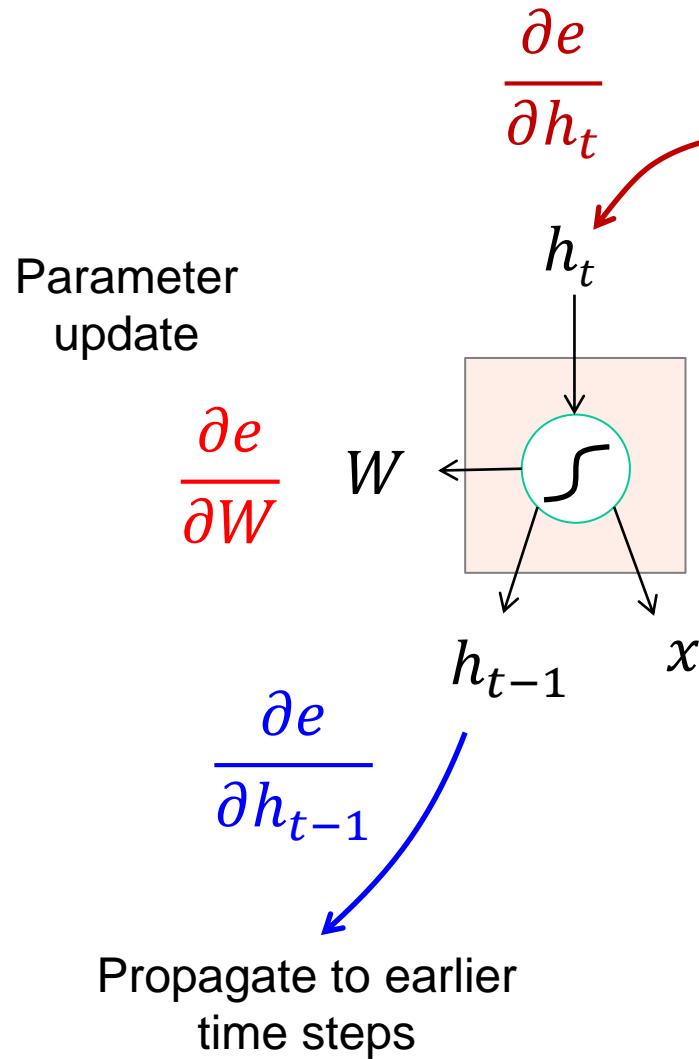


Carry hidden states forward in time further, but only backpropagate for some smaller number of steps

Truncated backpropagation through time



RNN backward pass



$\frac{\partial e}{\partial h_t}$

Error from
predictions at
future steps

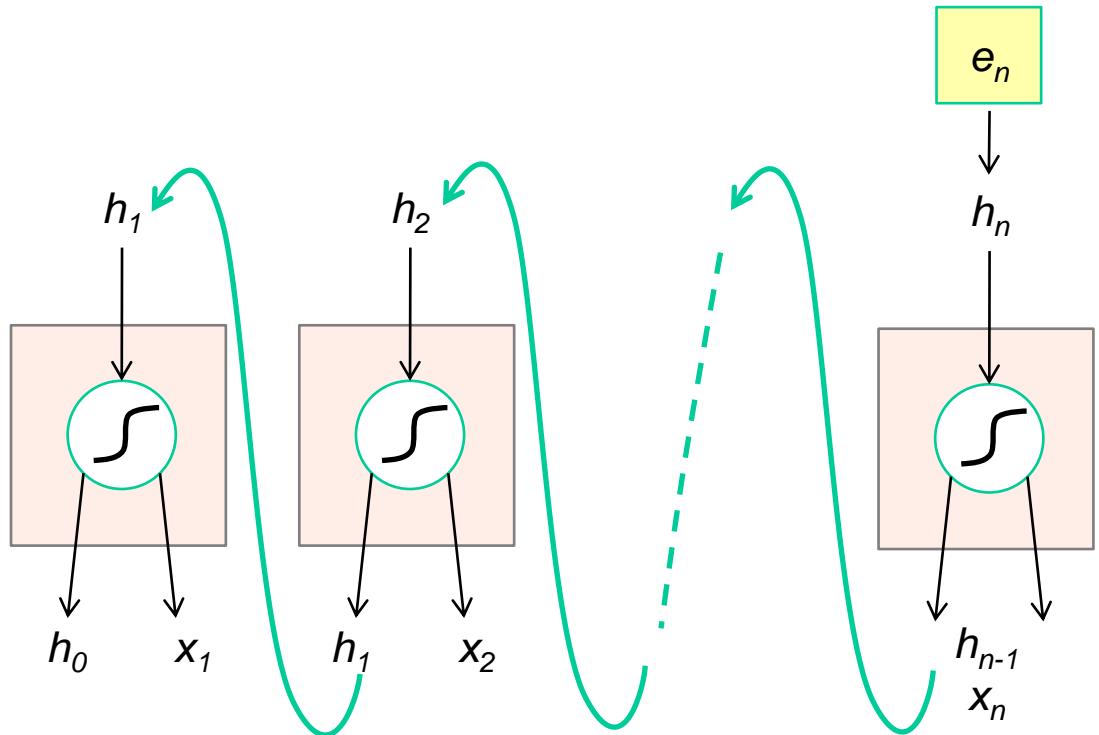
$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

$$\frac{\partial e}{\partial W_h} = \frac{\partial e}{\partial h_t} \odot (1 - \tanh^2(W_x x_t + W_h h_{t-1})) h_{t-1}^T$$

$$\frac{\partial e}{\partial W_x} = \frac{\partial e}{\partial h_t} \odot (1 - \tanh^2(W_x x_t + W_h h_{t-1})) x_t^T$$

$$\frac{\partial e}{\partial h_{t-1}} = W_h^T (1 - \tanh^2(W_x x_t + W_h h_{t-1})) \odot \frac{\partial e}{\partial h_t}$$

Vanishing and exploding gradients



$$\frac{\partial e}{\partial h_{t-1}} = W_h^T (1 - \tanh^2(W_x x_t + W_h h_{t-1})) \odot \frac{\partial e}{\partial h_t}$$

Computing gradient for h_0 involves many multiplications by W_h^T (and rescalings between 0 and 1)

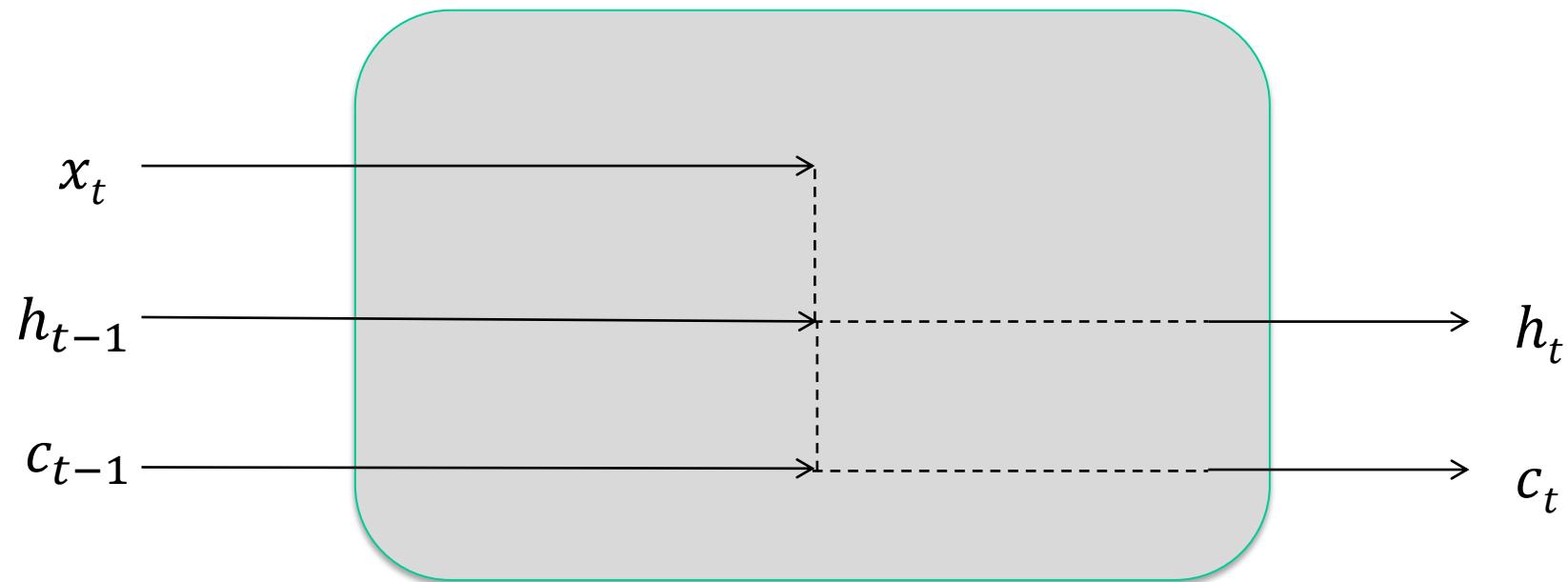
Gradients will *vanish* if largest singular value of W_h is less than 1 and *explode* if it's greater than 1

Outline

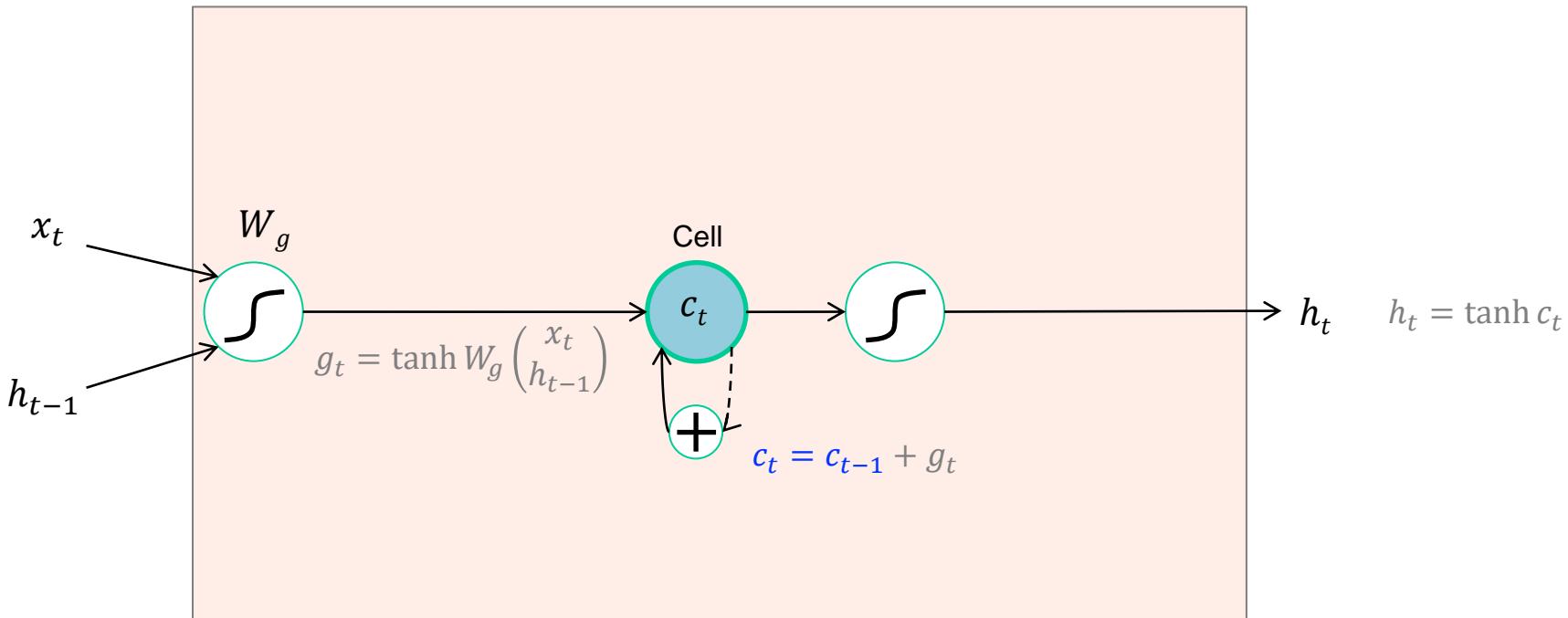
- Examples of sequential prediction tasks
- Common recurrent units
 - Vanilla RNN unit (and how to train it)
 - Long Short-Term Memory (LSTM)
 - Gated Recurrent Unit (GRU)

Long short-term memory (LSTM)

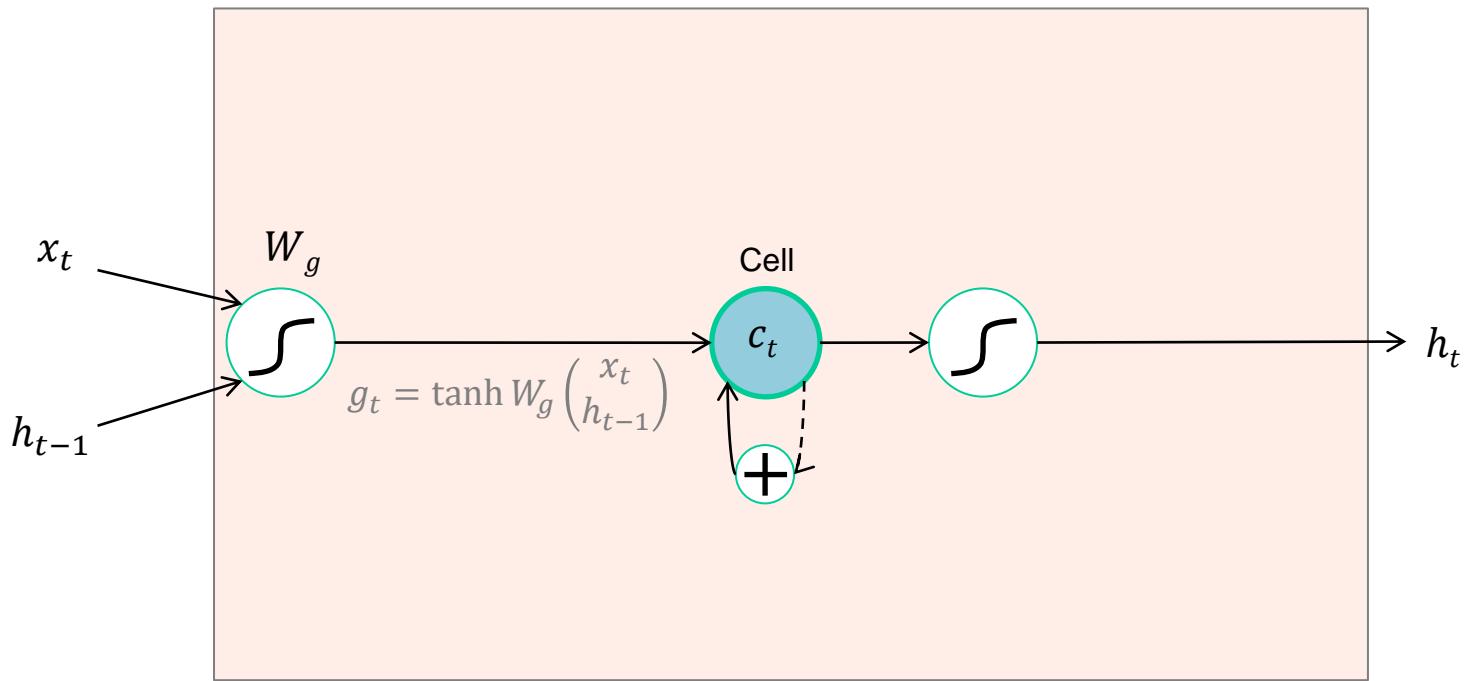
- Add a *memory cell* that is not subject to matrix multiplication or squashing, thereby avoiding gradient decay



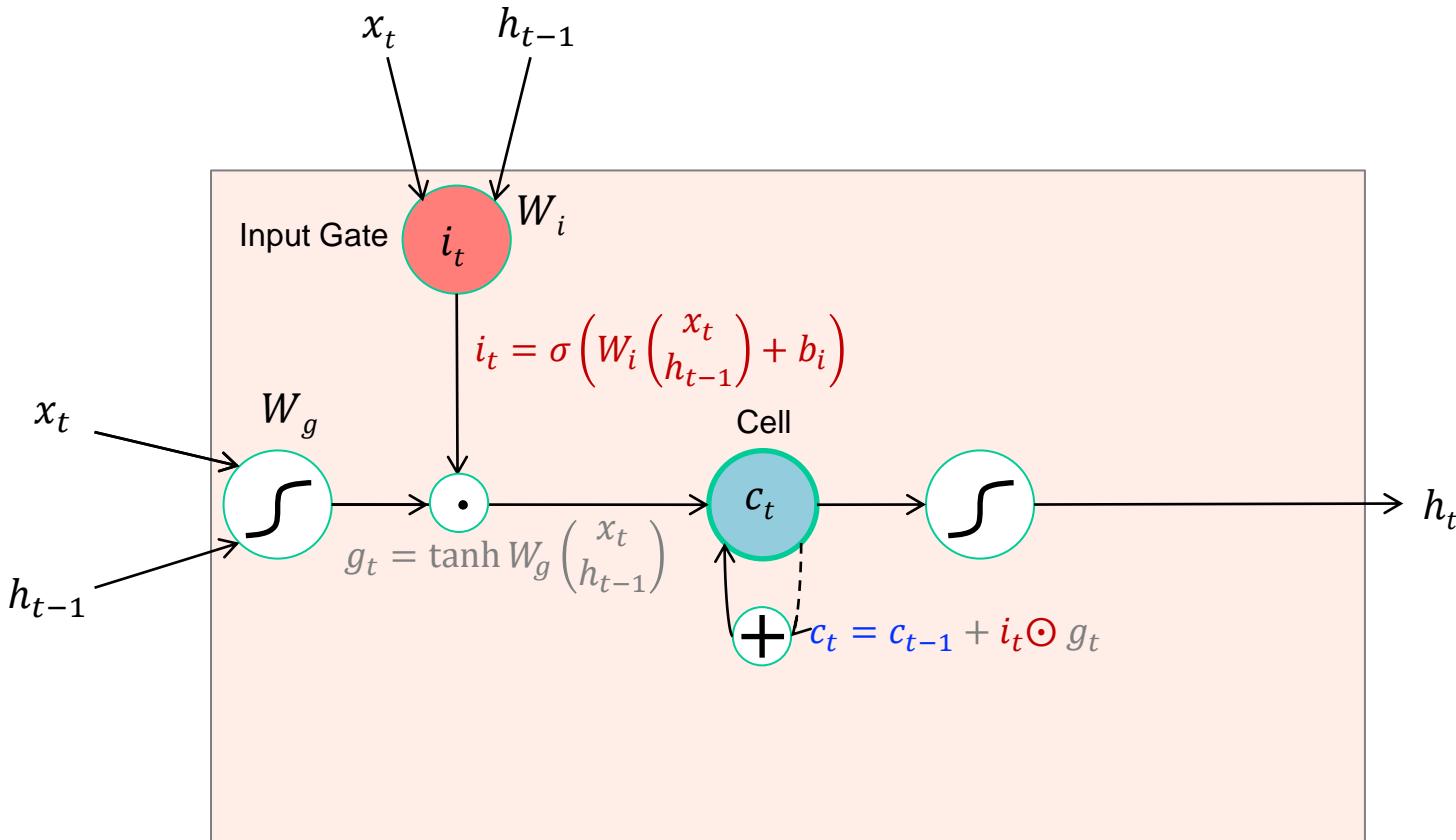
The LSTM cell



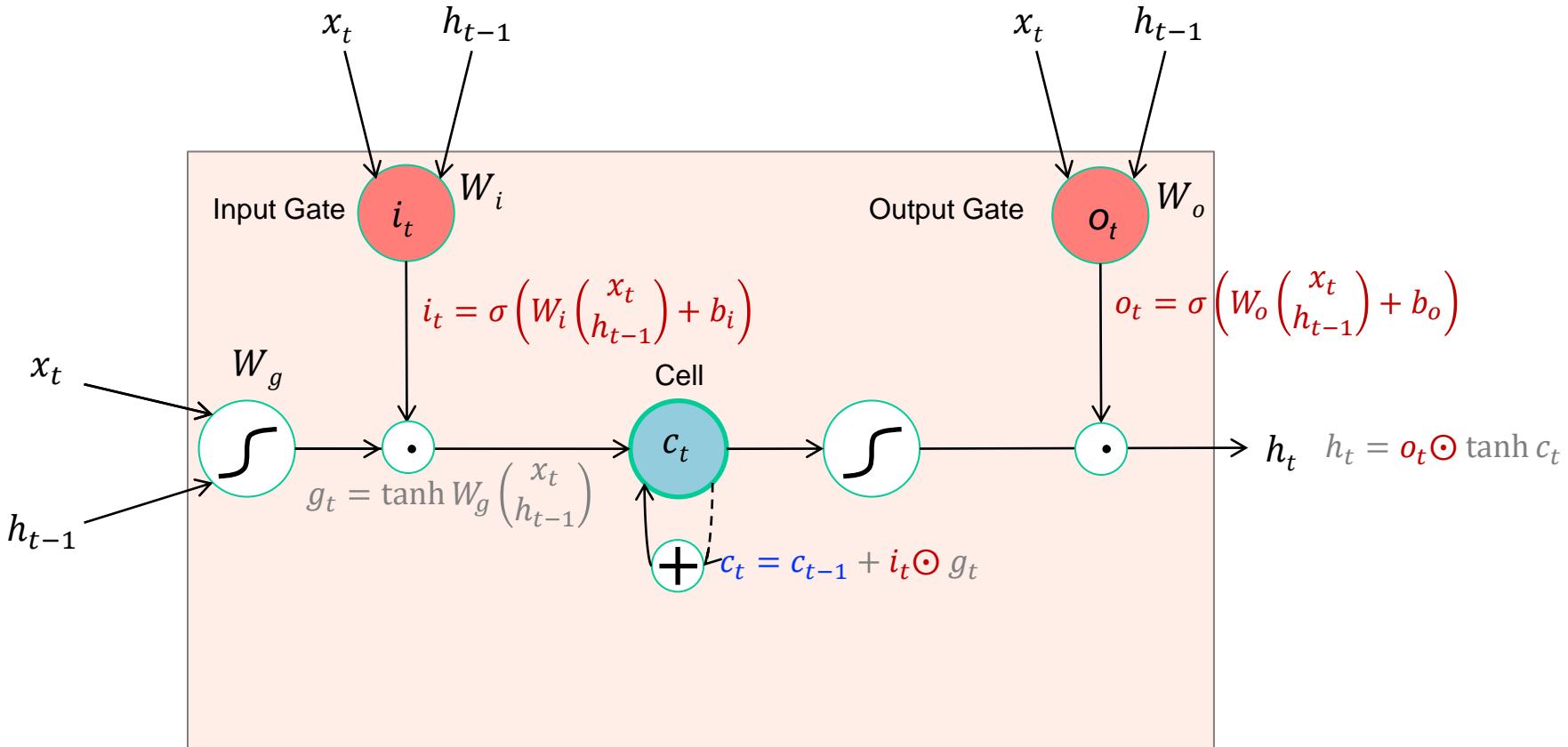
The LSTM cell



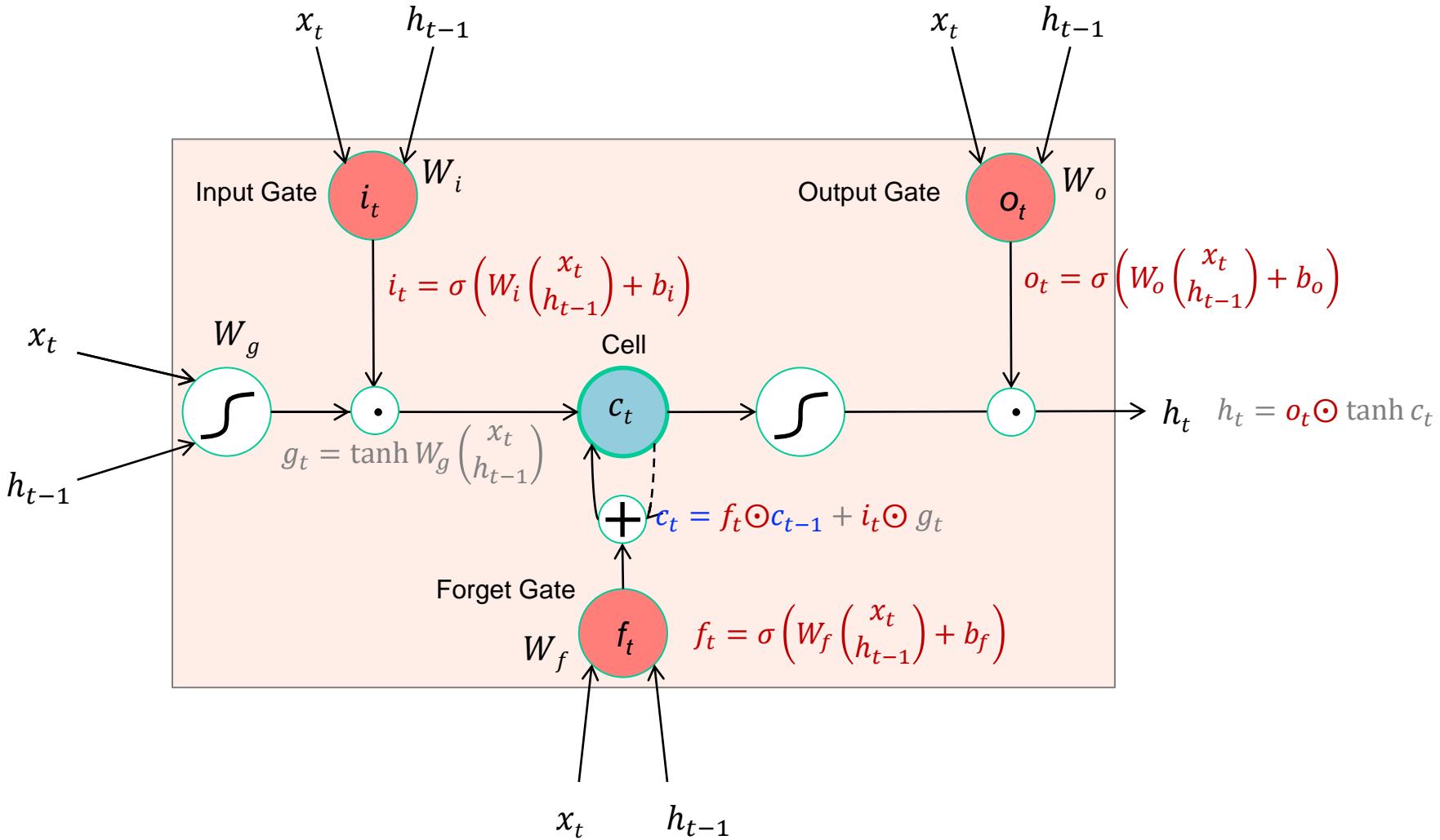
The LSTM cell



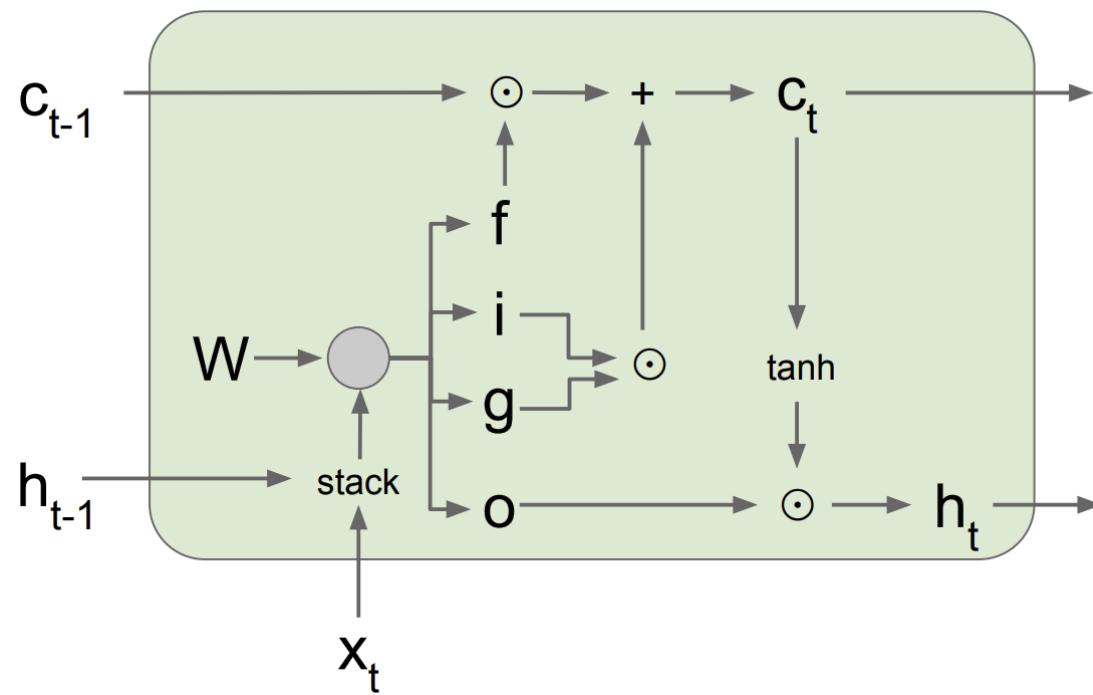
The LSTM cell



The LSTM cell



LSTM forward pass summary

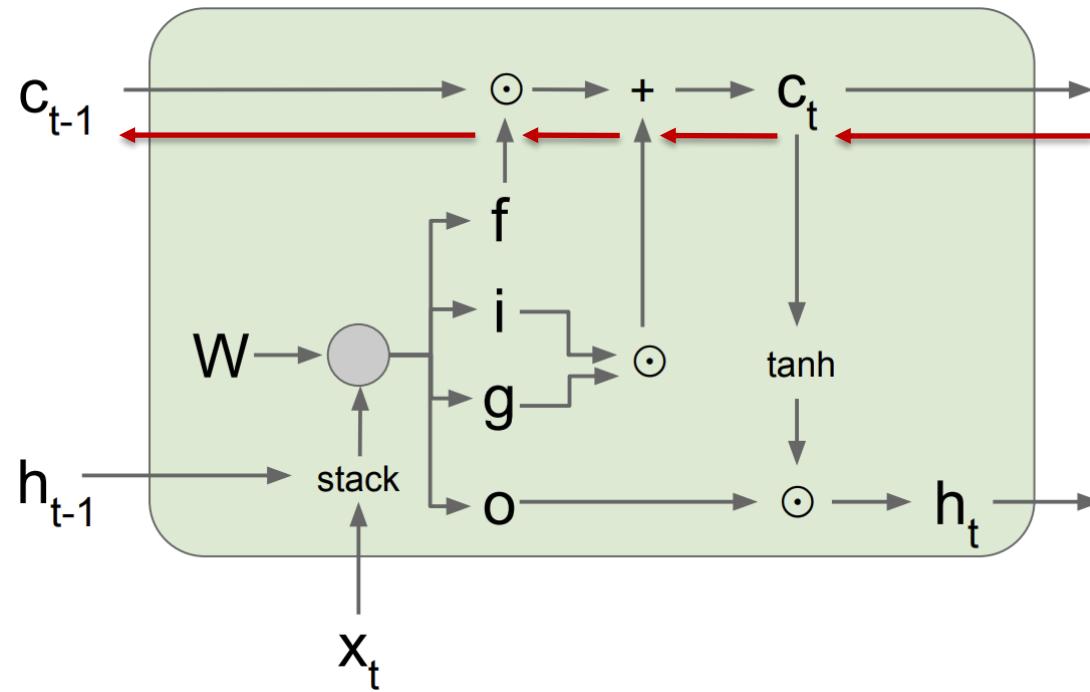


$$\begin{pmatrix} g_t \\ i_t \\ f_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} \begin{pmatrix} W_g \\ W_i \\ W_f \\ W_o \end{pmatrix} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh c_t$$

LSTM backward pass

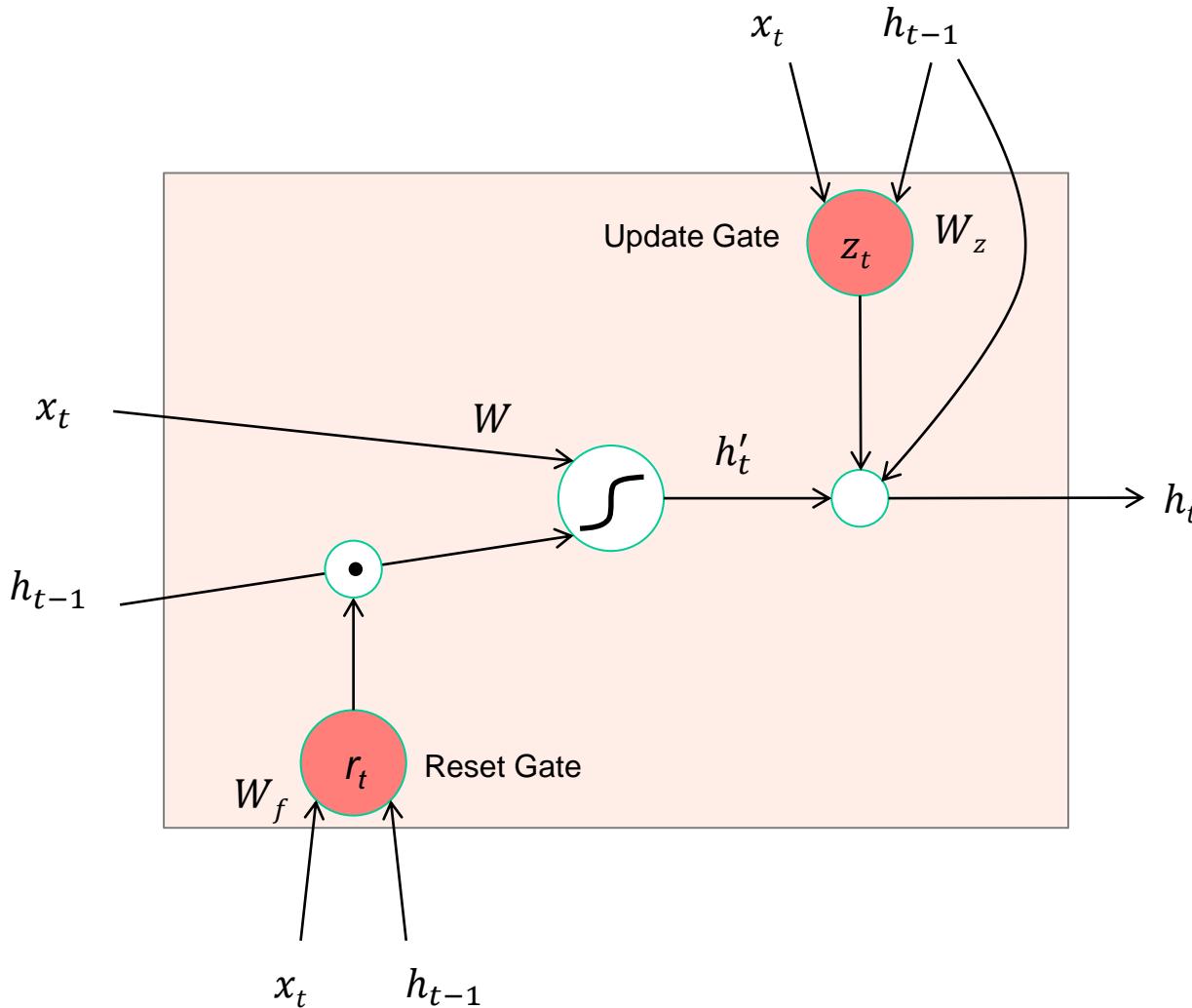


Gradient flow from c_t to c_{t-1} only involves back-propagating through addition and elementwise multiplication, not matrix multiplication or tanh

For complete details: [Illustrated LSTM Forward and Backward Pass](#)

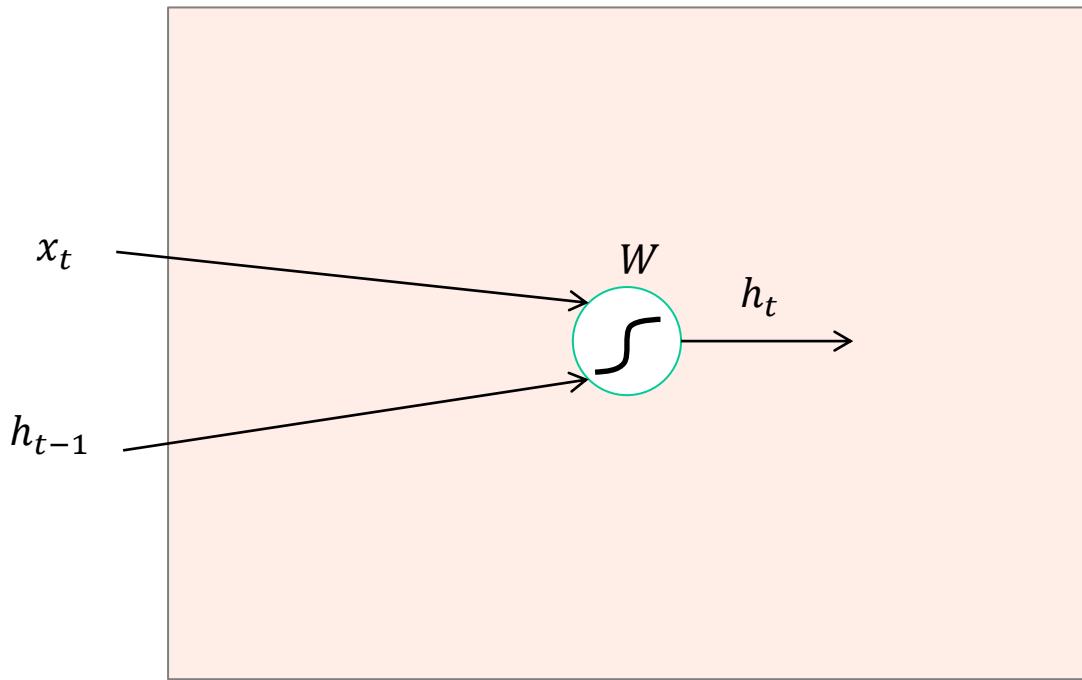
[Figure source](#)

Gated recurrent unit (GRU)



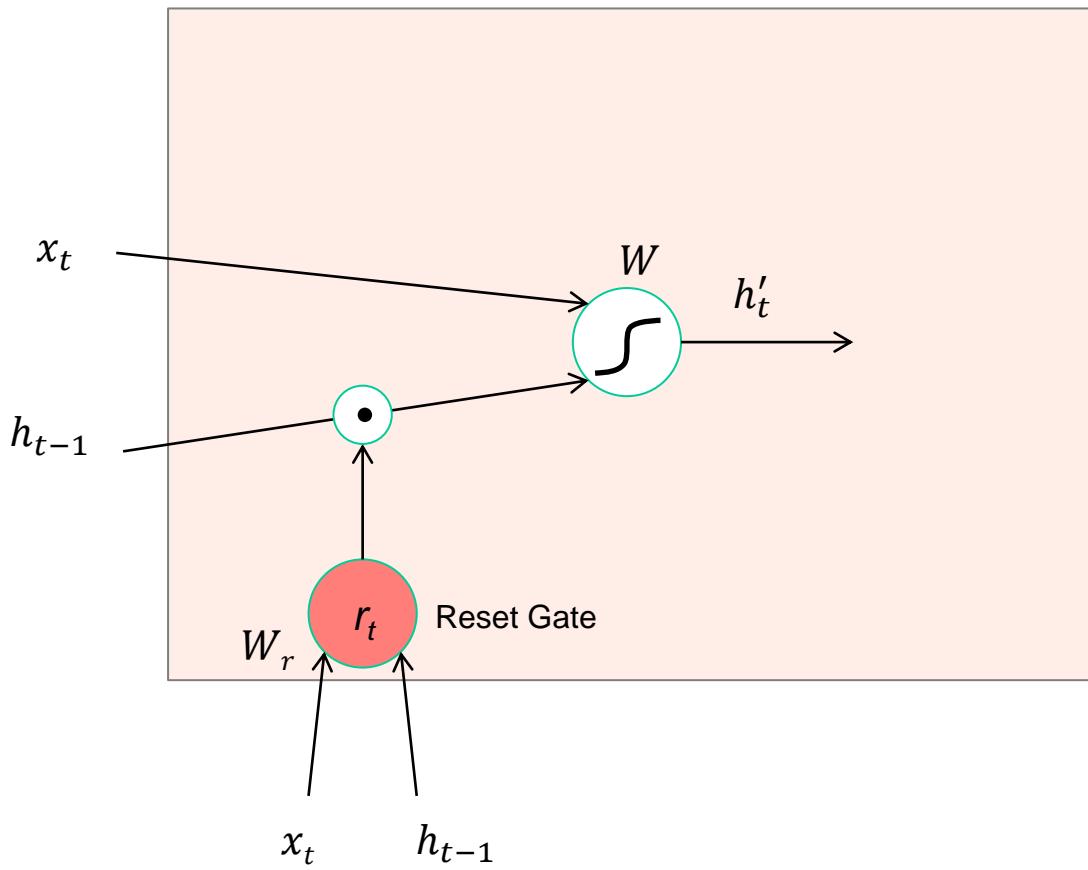
- Get rid of separate cell state
- Merge “forget” and “output” gates into “update” gate

Gated recurrent unit (GRU)



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

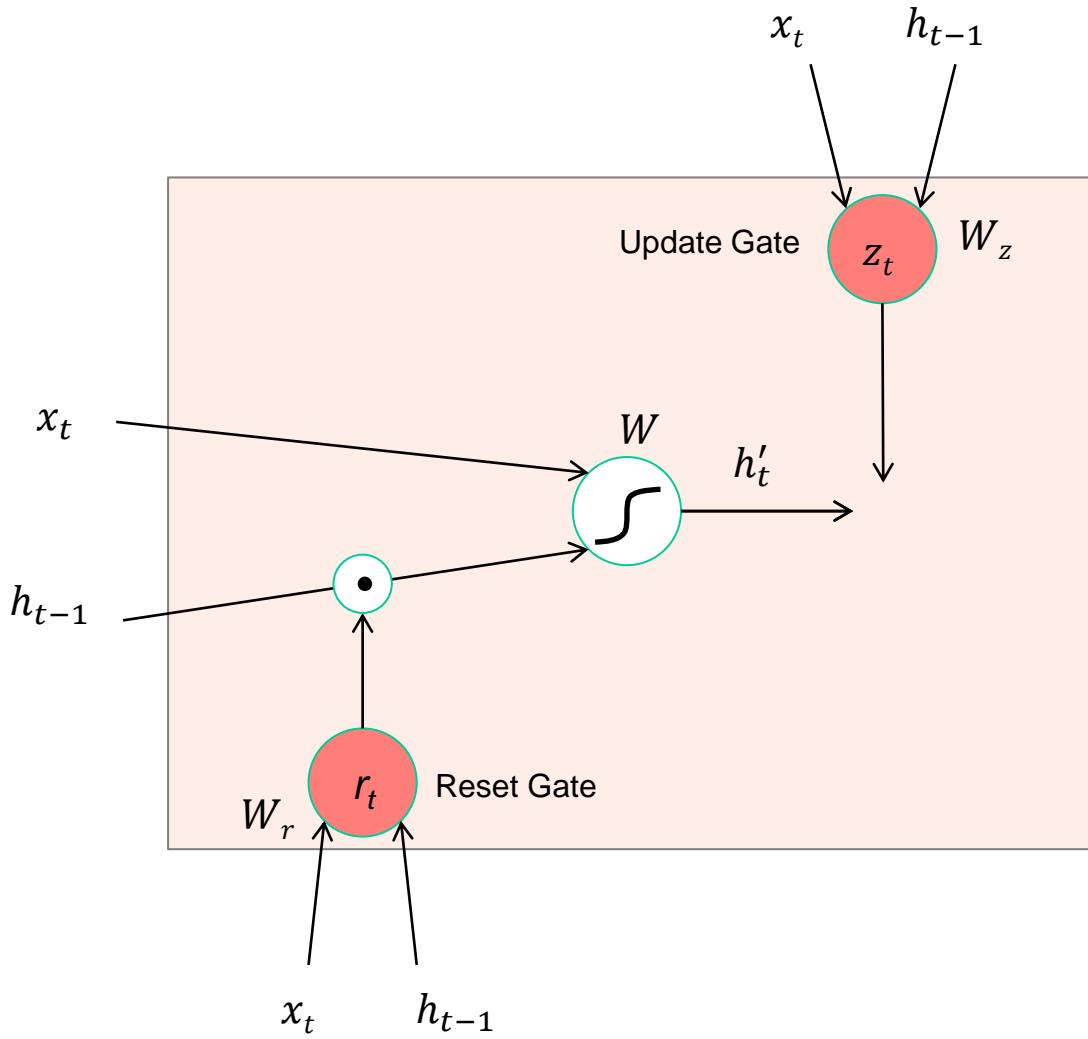
Gated recurrent unit (GRU)



$$r_t = \sigma \left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_t \right)$$

$$h'_t = \tanh W \left(r_t \odot \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \right)$$

Gated recurrent unit (GRU)

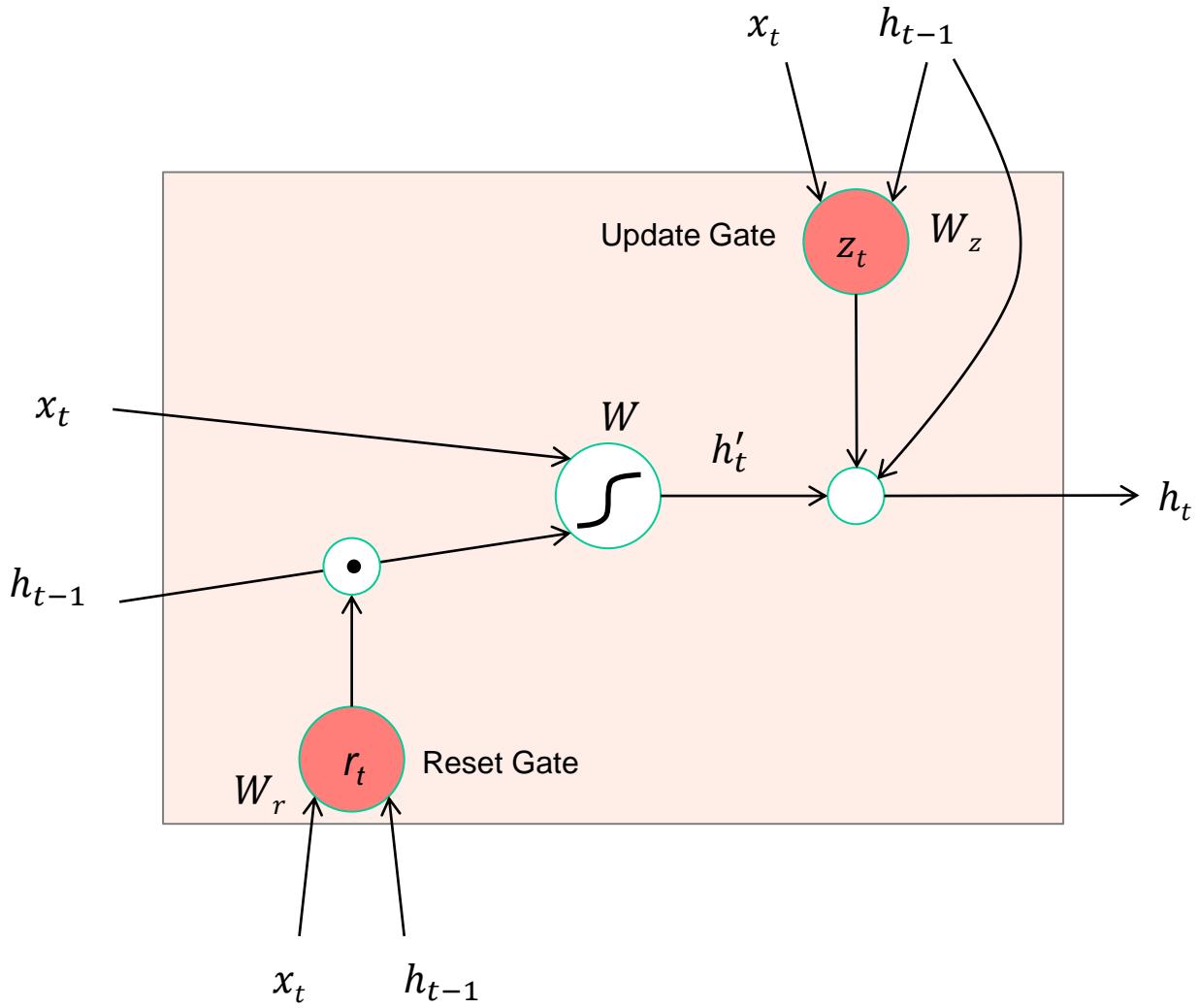


$$r_t = \sigma(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_r)$$

$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \odot h_{t-1} \end{pmatrix}$$

$$z_t = \sigma(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_z)$$

Gated recurrent unit (GRU)



$$r_t = \sigma(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_r)$$

$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \odot h_{t-1} \end{pmatrix}$$

$$z_t = \sigma(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_z)$$

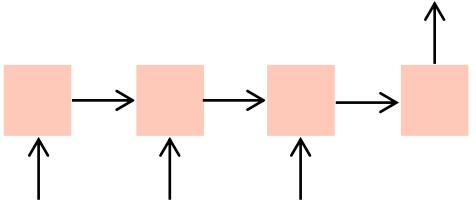
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t$$

Outline

- Examples of sequential prediction tasks
- Common recurrent units
 - Vanilla RNN unit (and how to train it)
 - Long Short-Term Memory (LSTM)
 - Gated Recurrent Unit (GRU)
- Recurrent network architectures

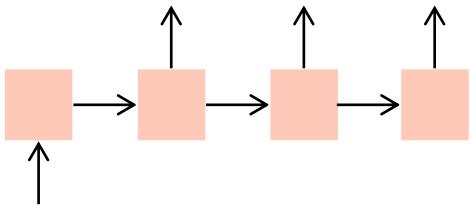
Summary: Input-output scenarios

Multiple -
Single



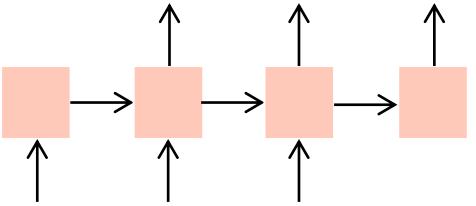
Sequence
Classification

Single -
Multiple



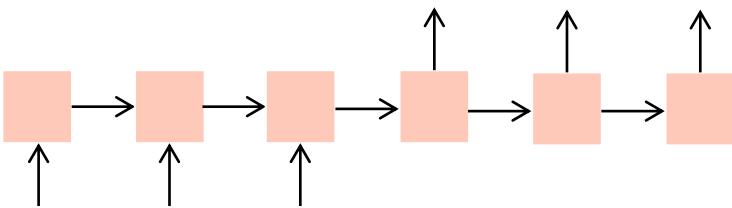
Sequence generation,
captioning

Multiple -
Multiple



Sequence generation,
captioning

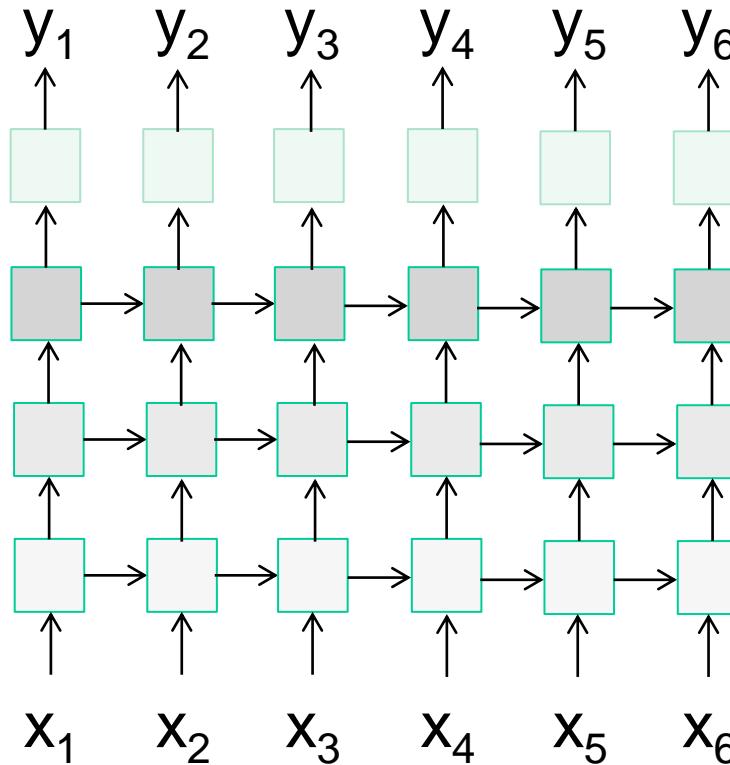
Multiple -
Multiple



Translation

Multi-layer RNNs

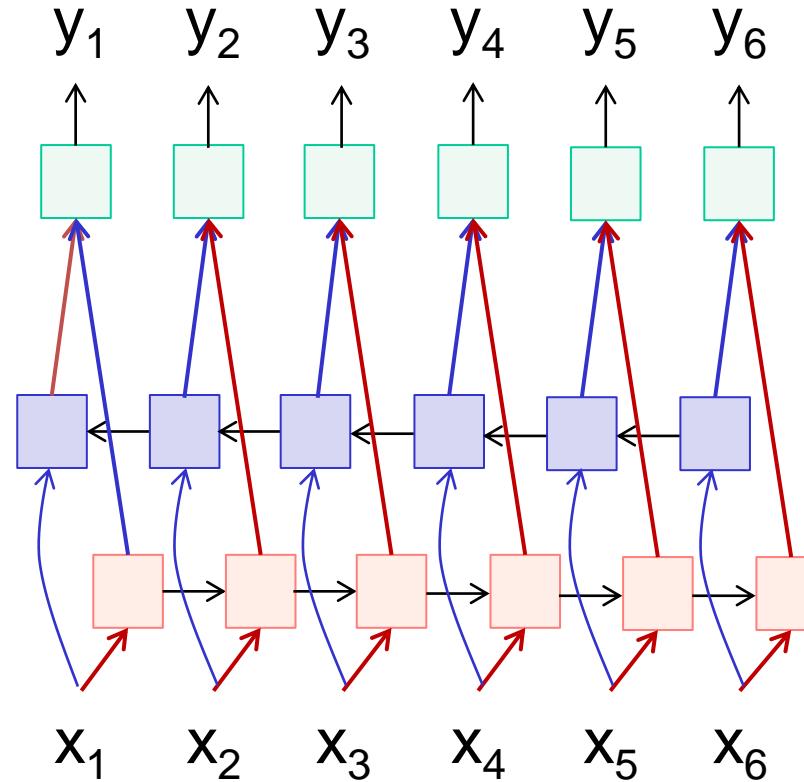
- We can of course design RNNs with multiple hidden layers



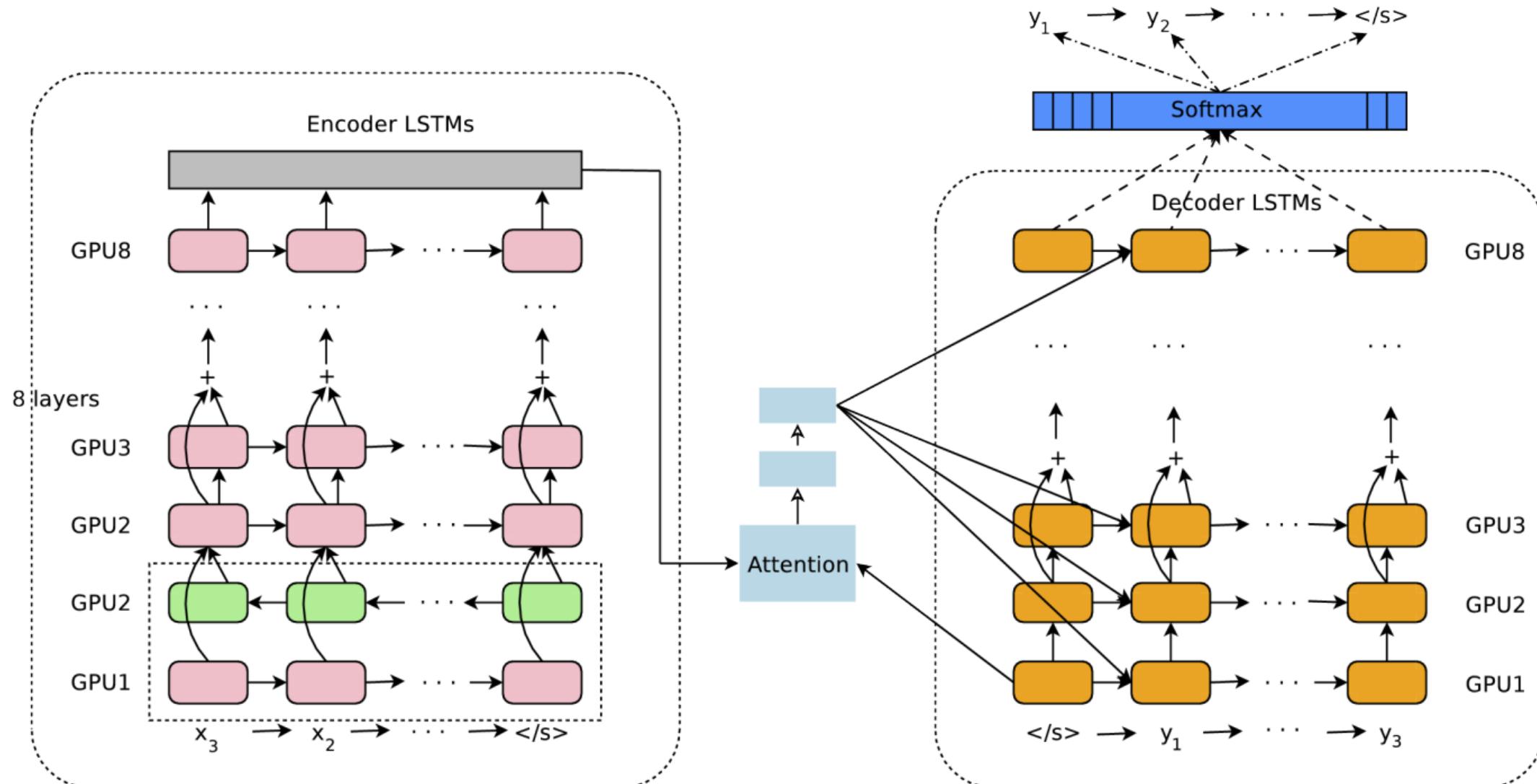
- Anything goes: skip connections across layers, across time, ...

Bi-directional RNNs

- RNNs can process the input sequence in forward and in the reverse direction (common in speech recognition)



Google Neural Machine Translation (GNMT)

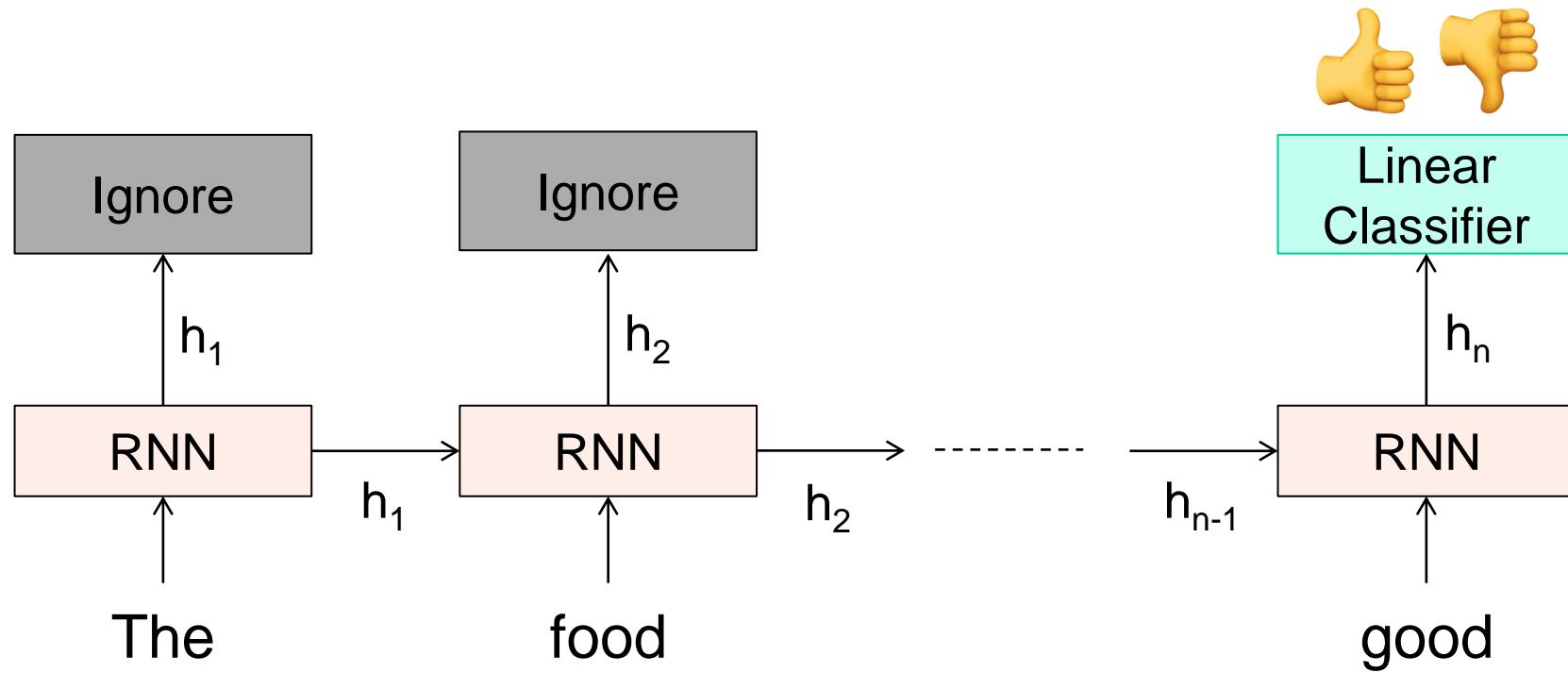


Y. Wu et al., [Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation](#), arXiv 2016

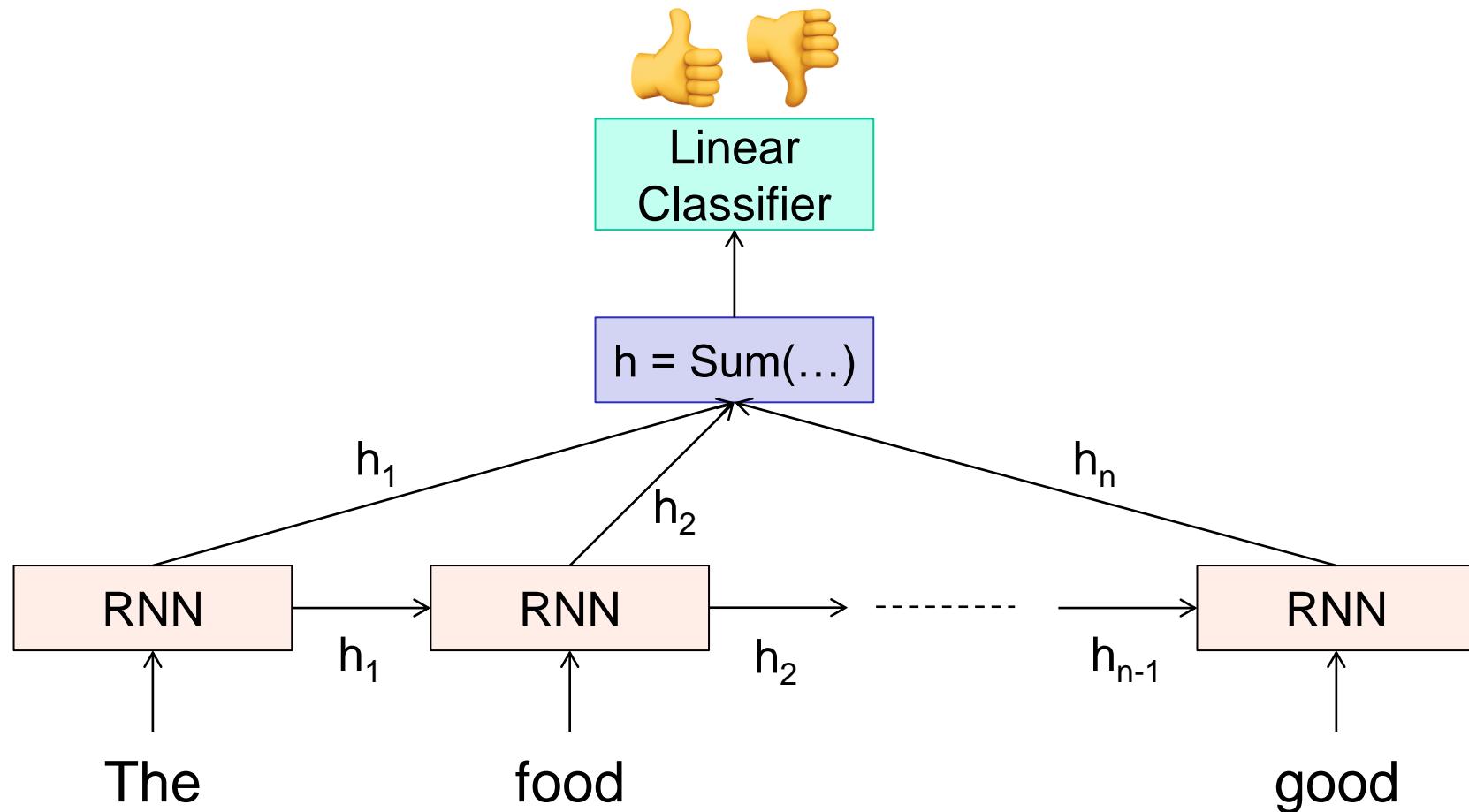
Outline

- Examples of sequential prediction tasks
- Common recurrent units
 - Vanilla RNN unit
 - Long Short-Term Memory (LSTM)
 - Gated Recurrent Unit (GRU)
- Recurrent network architectures
- Applications in (a bit) more detail
 - Sequence classification
 - Language modeling
 - Image captioning
 - Machine translation

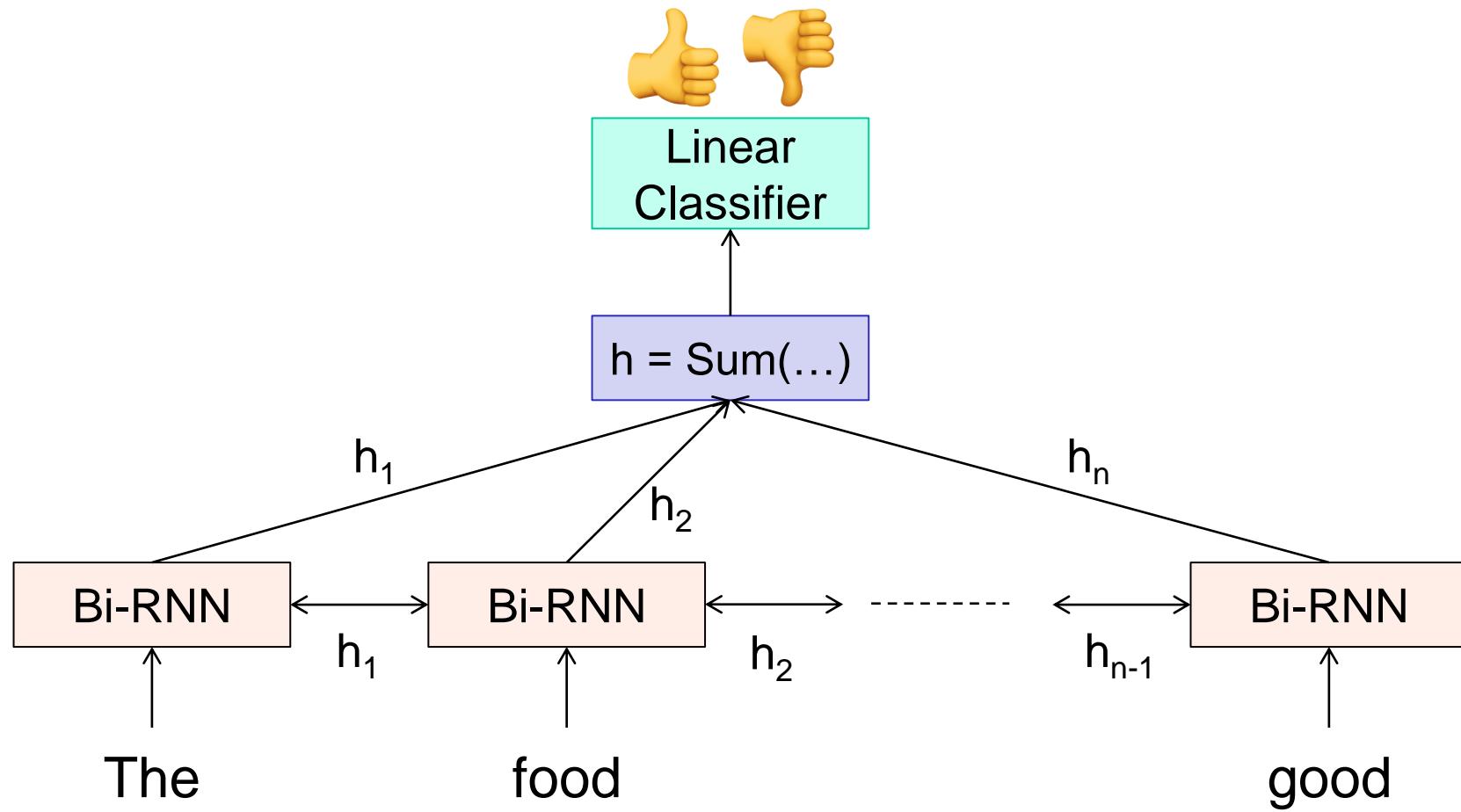
Sequence classification



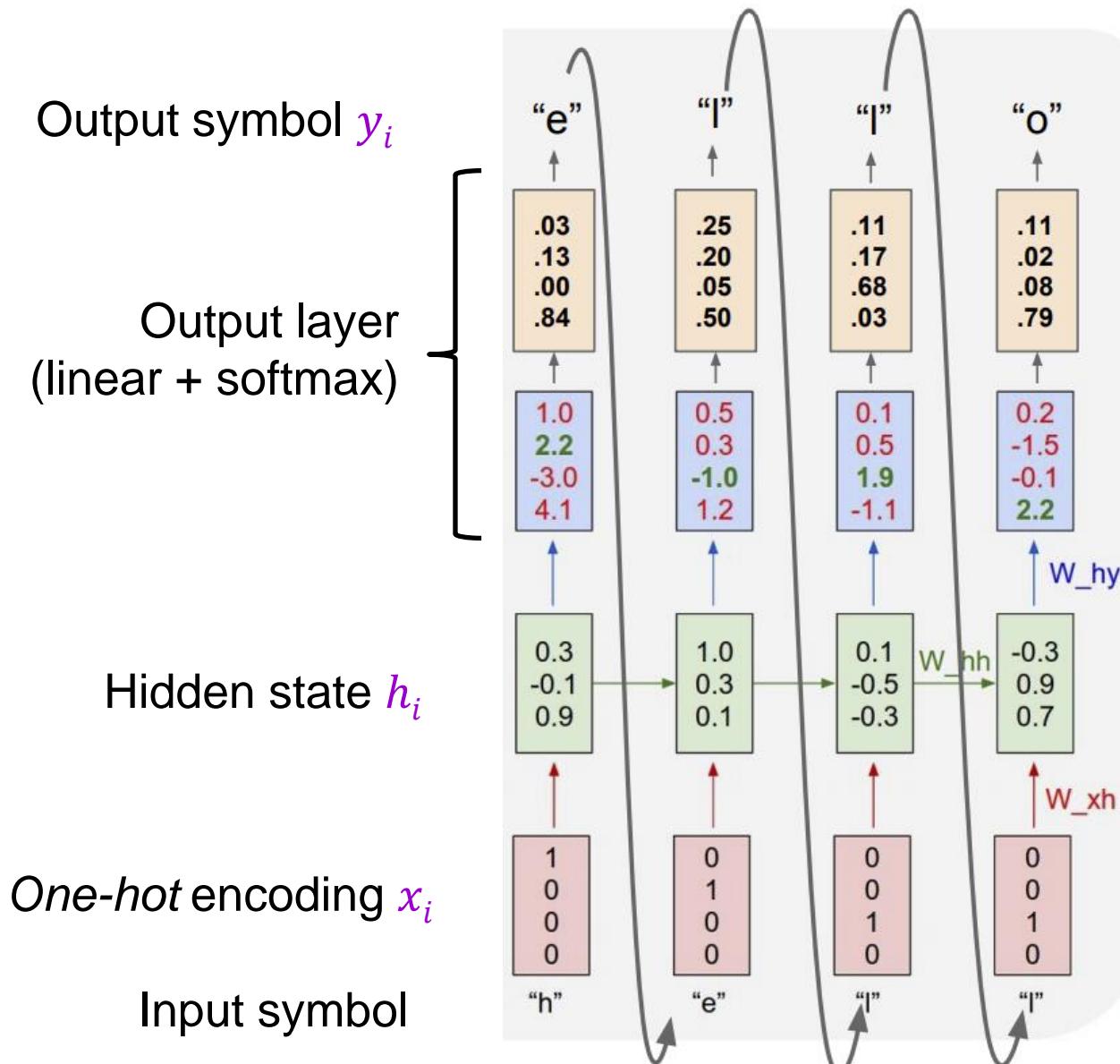
Sequence classification



Sequence classification



Language modeling: Character RNN



$$\begin{aligned} p(y_1, y_2, \dots, y_n) \\ = \prod_{i=1}^n p(y_i | y_1, \dots, y_{i-1}) \\ \approx \prod_{i=1}^n P_W(y_i | h_i) \end{aligned}$$

Language modeling: Character RNN

100th
iteration

tyntd-iafhatawiaoahrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ train more

300th
iteration

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwyl fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ train more

700th
iteration

Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.

↓ train more

2000th
iteration

"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.

Searching for interpretable hidden units

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

Searching for interpretable hidden units

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

line position tracking cell

Searching for interpretable hidden units

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!!(current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```

if statement cell

Searching for interpretable hidden units

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
    struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
        (void *)adf->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM \\'%s\\' is invalid\n",
            df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

quote/comment cell

Searching for interpretable hidden units

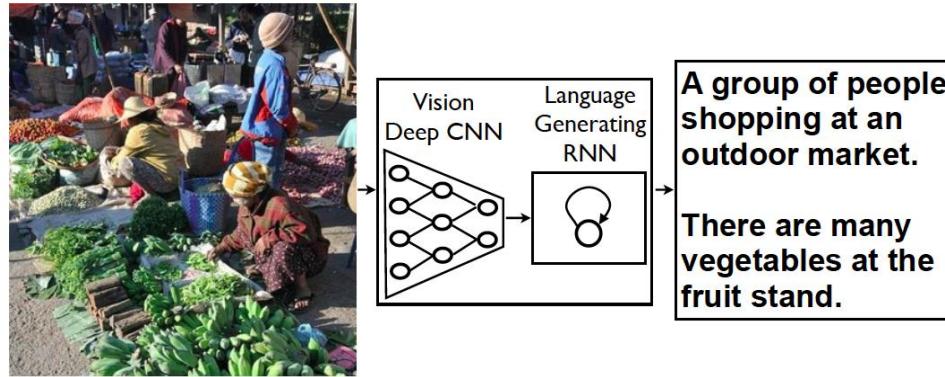
```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

code depth cell

RNNs: Outline

- Examples of sequential prediction tasks
- Common recurrent units
 - Vanilla RNN unit
 - Long Short-Term Memory (LSTM)
 - Gated Recurrent Unit (GRU)
- Recurrent network architectures
 - Multilayer, bidirectional, skip connections
- Applications in (a bit) more detail
 - Sequence classification
 - Language modeling
 - Image captioning
 - Machine translation

Image caption generation



Training time

- Maximize likelihood of reference captions

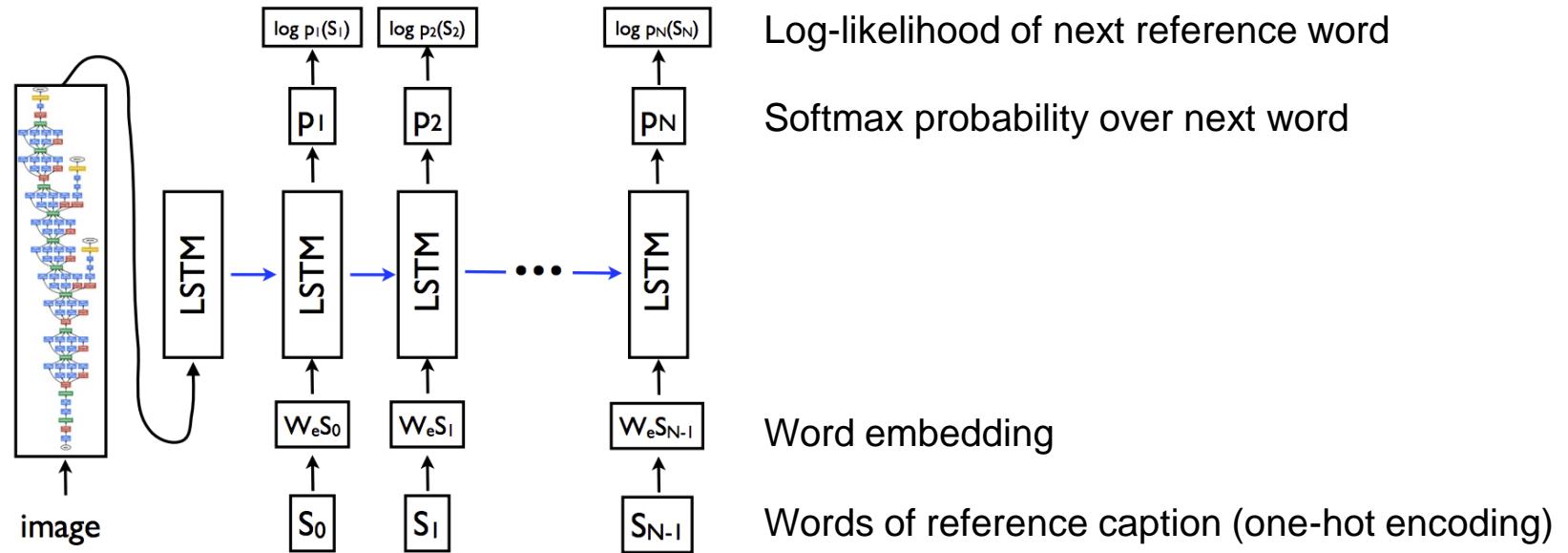


Image caption generation: Training time

- Minimize negative log-likelihood of the ground truth caption $Y^* = (Y_1^*, \dots, Y_N^*)$ given image I :

$$L(I, Y^*) = - \sum_{t=1}^N \log P_W(Y_t^* | Y_1^*, \dots, Y_{t-1}^*, I)$$

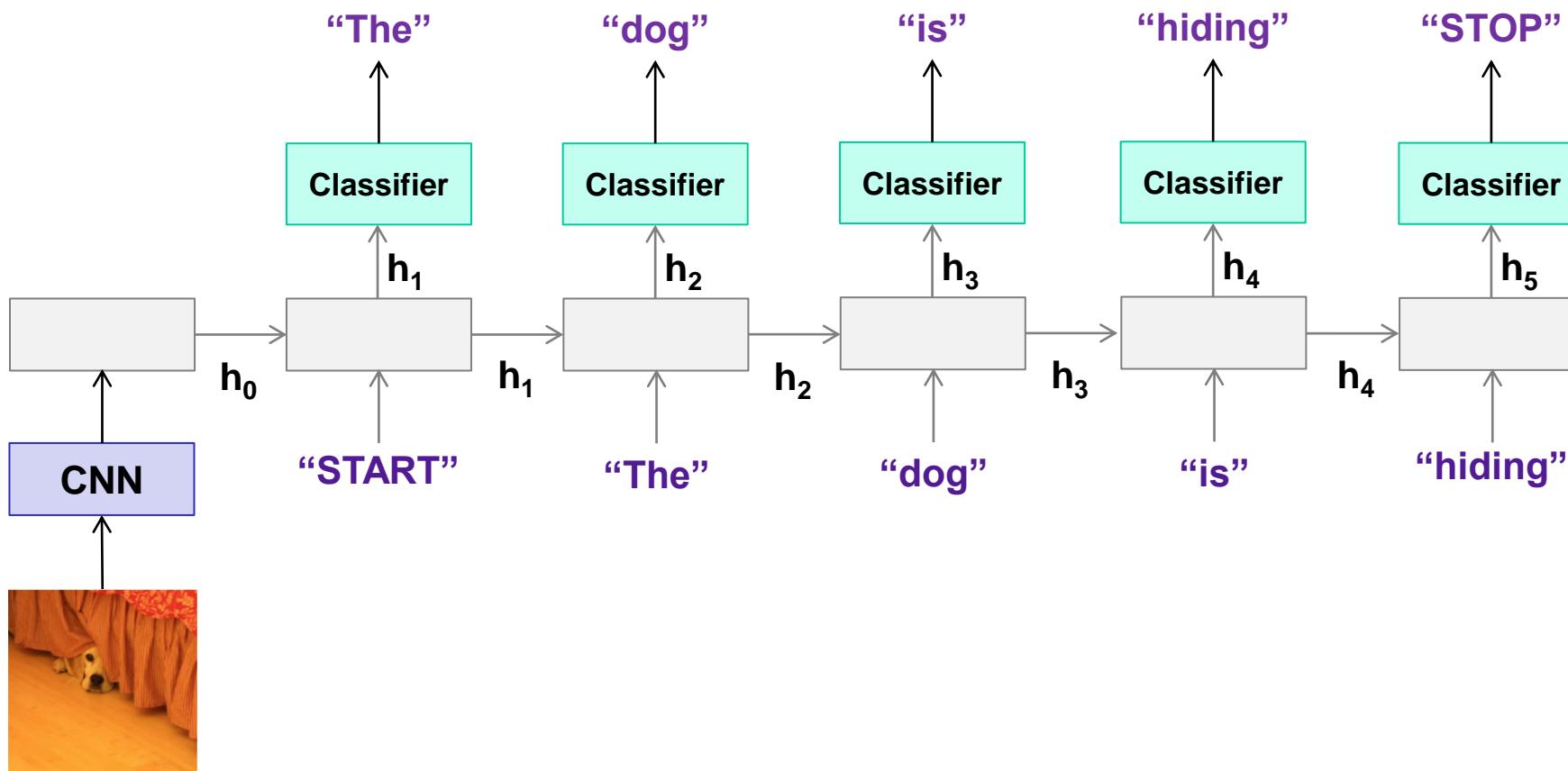


Image caption generation: Test time

- Sample next word according to posterior distribution of classifier
 - Sentences quickly become incoherent
- Always choose the highest-likelihood word
 - Does this necessarily maximize the likelihood of the overall sentence?

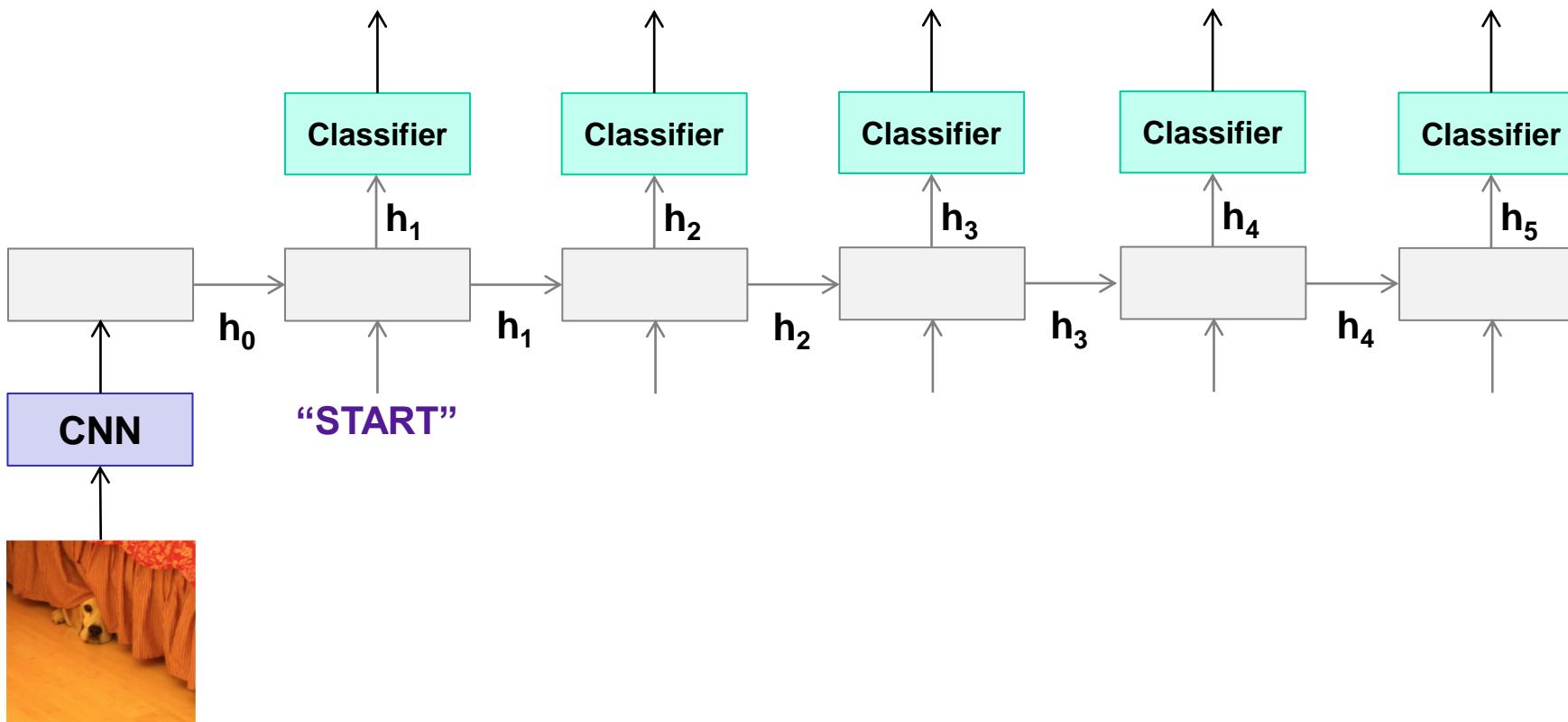


Image caption generation: Test time

- Beam search:
 - Maintain k (*beam width*) top-scoring candidate sentences according to sum of per-word log-likelihoods (or some other score)
 - At each step, generate all their successors and keep the best k

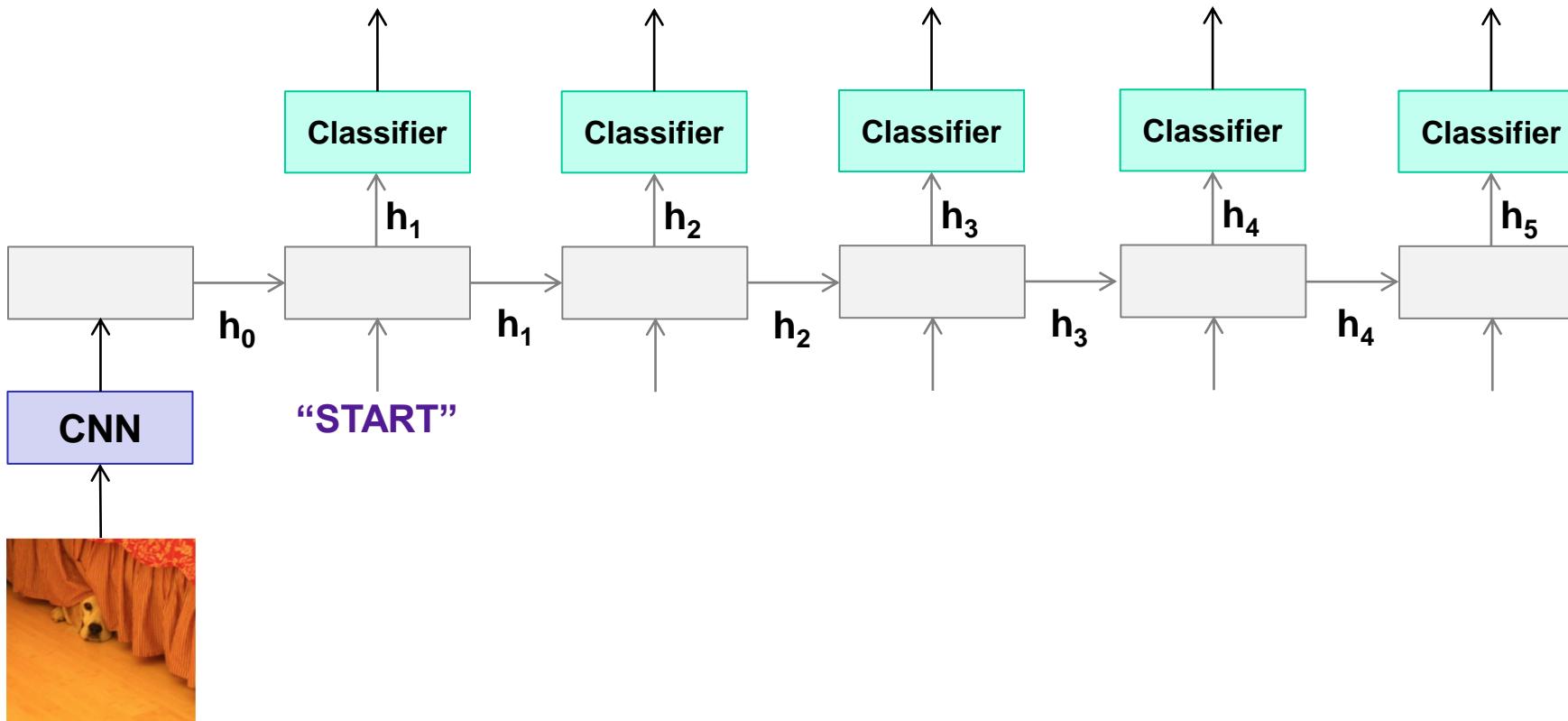


Image caption generation: Beam search

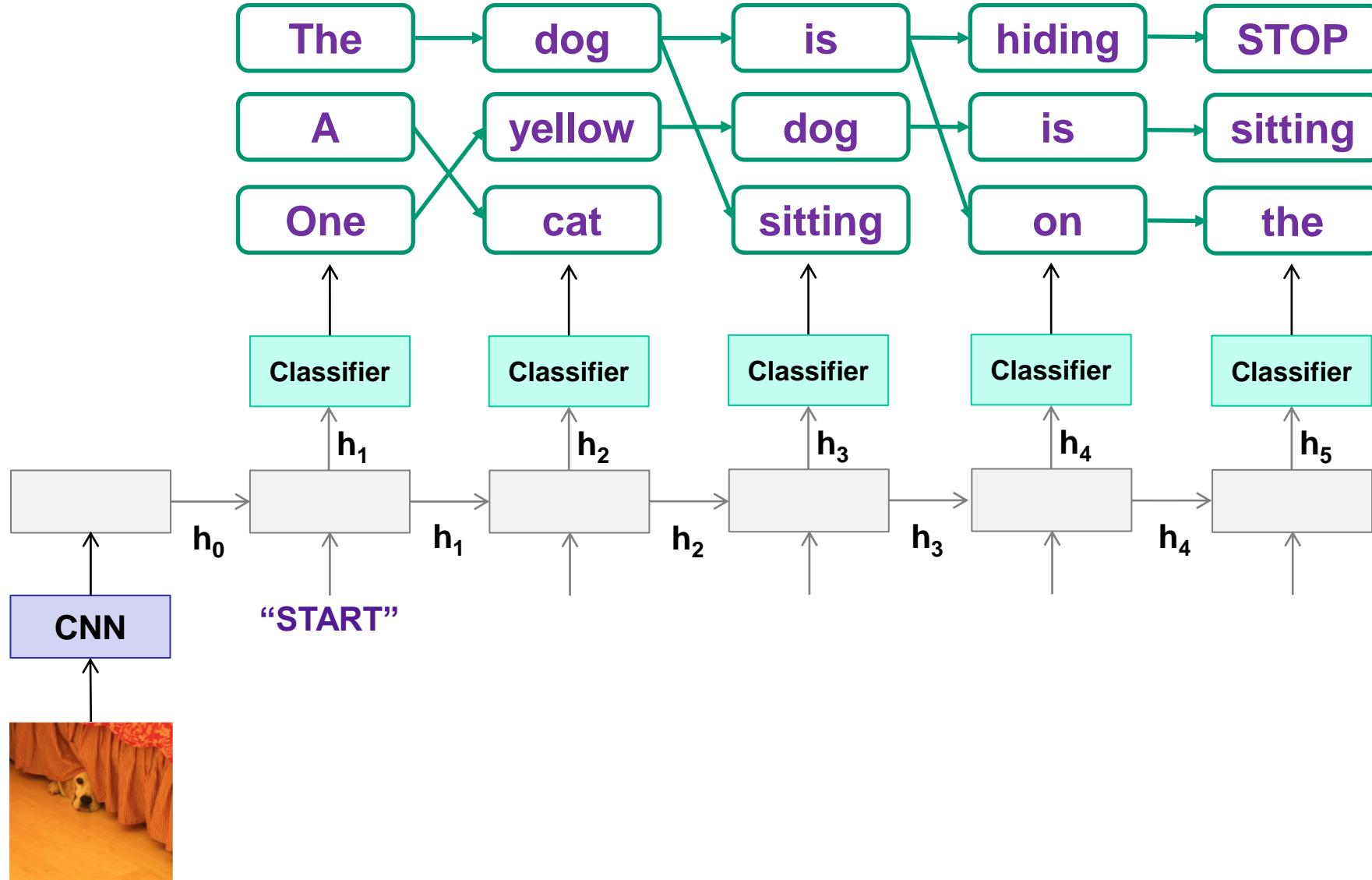


Image caption generation: Example outputs

A person riding a motorcycle on a dirt road.



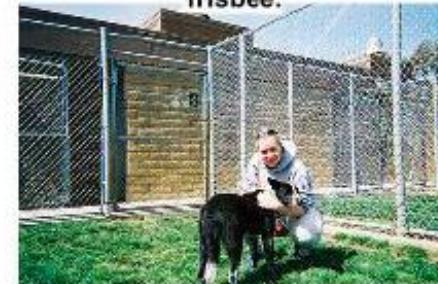
Two dogs play in the grass.



A skateboarder does a trick on a ramp.



A dog is jumping to catch a frisbee.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



A little girl in a pink hat is blowing bubbles.



A refrigerator filled with lots of food and drinks.



A herd of elephants walking across a dry grass field.



A close up of a cat laying on a couch.



A red motorcycle parked on the side of the road.



A yellow school bus parked in a parking lot.



Describes without errors

Describes with minor errors

Somewhat related to the image

Unrelated to the image

How to evaluate image captioning?

Reference sentences (written by human annotators):



- “A dog hides underneath a bed with its face peeking out of the bed skirt”
- “The small white dog is peeking out from under the bed”
- “A dog is peeking its head out from underneath a bed skirt”
- “A dog peeking out from under a bed”
- “A dog that is under a bed on the floor”

Generated sentence:

- “A dog is hiding”

BLEU: Bilingual Evaluation Understudy

- **N-gram precision:** count the number of n-gram matches between candidate and reference translation, divide by total number of n-grams in candidate translation
 - Clip counts by the maximum number of times an n-gram occurs in any reference translation
 - Multiply by *brevity penalty* to penalize short translations
- Most commonly used measure for image captioning and machine translation despite multiple shortcomings



Overview

Challenges

Download

Evaluate

Leaderboard

Table-C5

Table-C40

2015 Captioning Challenge

Last update: June 8, 2015. Visit [CodaLab](#) for the latest results.

	CIDEr-D	Meteor	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4
m-RNN (Baidu/ UCLA) ^[16]	0.886	0.238	0.524	0.72	0.553	0.41	0.302
m-RNN ^[15]	0.817	0.210	0.501	0.710	0.515	0.301	0.299
MSR Captions ^[1]	0.817	0.210	0.501	0.710	0.515	0.301	0.308
Google ^[4]	CIDEr-D	CIDEr: Consensus-based Image Description Evaluation					0.309
Berkeley LRCC ^[17]	METEOR	Meteor Universal: Language Specific Translation Evaluation for Any Target Language					0.277
Nearest Neighbors ^[18]	Rouge-L	ROUGE: A Package for Automatic Evaluation of Summaries					0.28
MSR ^[8]	BLEU	BLEU: a Method for Automatic Evaluation of Machine Translation					0.291
Montreal/Toronto ^[10]	0.85	0.243	0.513	0.689	0.515	0.372	0.268
PicSOM ^[13]	0.833	0.231	0.505	0.683	0.51	0.377	0.281
Tsinghua Bigeye ^[14]	0.673	0.207	0.49	0.671	0.494	0.35	0.241
MLBL ^[7]	0.74	0.219	0.499	0.666	0.498	0.362	0.26
Human ^[5]	0.854	0.252	0.484	0.663	0.469	0.321	0.217

Metrics

CIDEr-D

CIDEr: Consensus-based Image Description Evaluation

METEOR

Meteor Universal: Language Specific Translation Evaluation for Any Target Language

Rouge-L

ROUGE: A Package for Automatic Evaluation of Summaries

BLEU

BLEU: a Method for Automatic Evaluation of Machine Translation



Overview

Challenges

Download

Evaluate

Leaderboard

Table-C5

Table-C40

2015 Captioning Challenge

Last update: June 8, 2015. Visit [CodaLab](#) for the latest results.

	M1	M2	M3	M4	M5
Human ^[5]	0.638	0.675	4.836	3.428	0.352
Google ^[4]	0.272	0.217	4.107	0.740	0.222
MSR ^[8]	M1	Percentage of captions that are evaluated as better or equal to human caption.			
Montreal	M2	Percentage of captions that pass the Turing Test.			
MSR Ca	M3	Average correctness of the captions on a scale 1-5 (incorrect - correct).			
Berkeley	M4	Average amount of detail of the captions on a scale 1-5 (lack of details - very detailed).			
m-RNN ^[1]	M5	Percentage of captions that are similar to human description.			
Nearest Neighbor ^[11]	0.216	0.255	3.801	2.716	0.196
PicSOM ^[13]	0.202	0.250	3.965	2.552	0.182
Brno University ^[3]	0.194	0.213	3.079	3.482	0.154
m-RNN (Baidu/ UCLA) ^[16]	0.190	0.241	3.831	2.548	0.195
MIL ^[6]	0.168	0.197	3.349	2.915	0.159
MLBL ^[7]	0.167	0.196	3.659	2.420	0.156

Generative model for diverse captioning

- We would like to sample diverse captions given an image to accurately reflect intrinsic open-endedness of the task



LSTM + beam search output lacks diversity

a close up of a plate of food with a sandwich on a table
a close up of a sandwich on a plate
a close up of a plate of food on a table
a close up of a plate of food with a sandwich on it
a close up of a plate of food on a white plate

Generative model for diverse captioning

- We would like to sample diverse captions given an image to accurately reflect intrinsic open-endedness of the task



LSTM + beam search output lacks diversity

a close up of a plate of food with a sandwich on a table
a close up of a sandwich on a plate
a close up of a plate of food on a table
a close up of a plate of food with a sandwich on it
a close up of a plate of food on a white plate

Conditional variational auto-encoder with additive Gaussian space (AG-CVAE)

a close up of a plate of food on a table
a table with a plate of food on it
a plate of food with a sandwich on it
a white plate topped with a plate of food
a plate of food on a table next to a cup of coffee

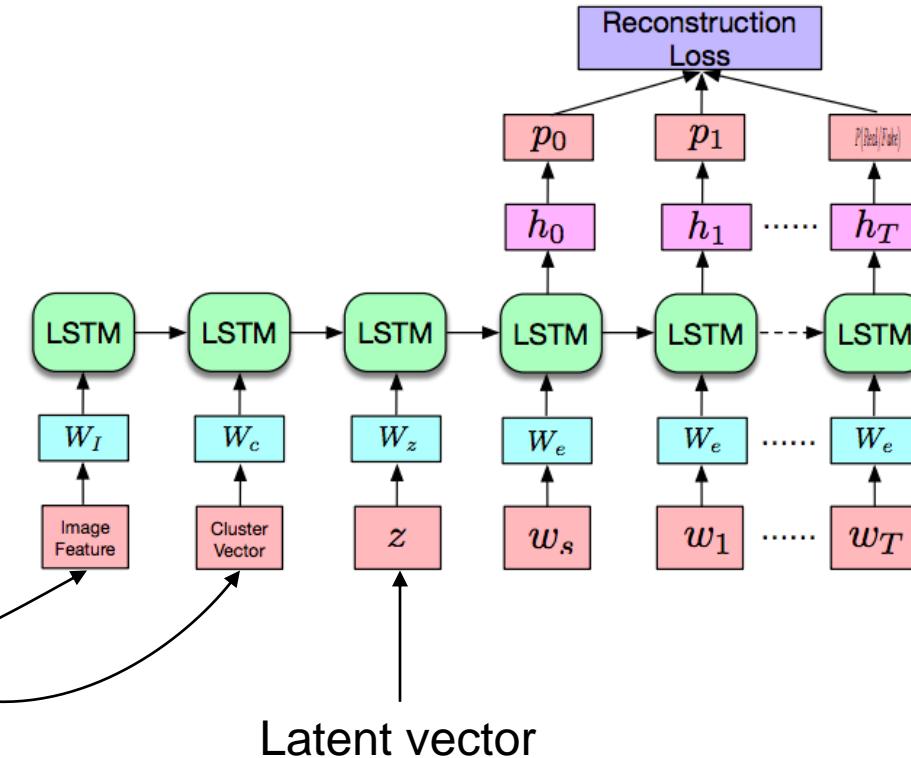
CVAE for captioning



Detector output

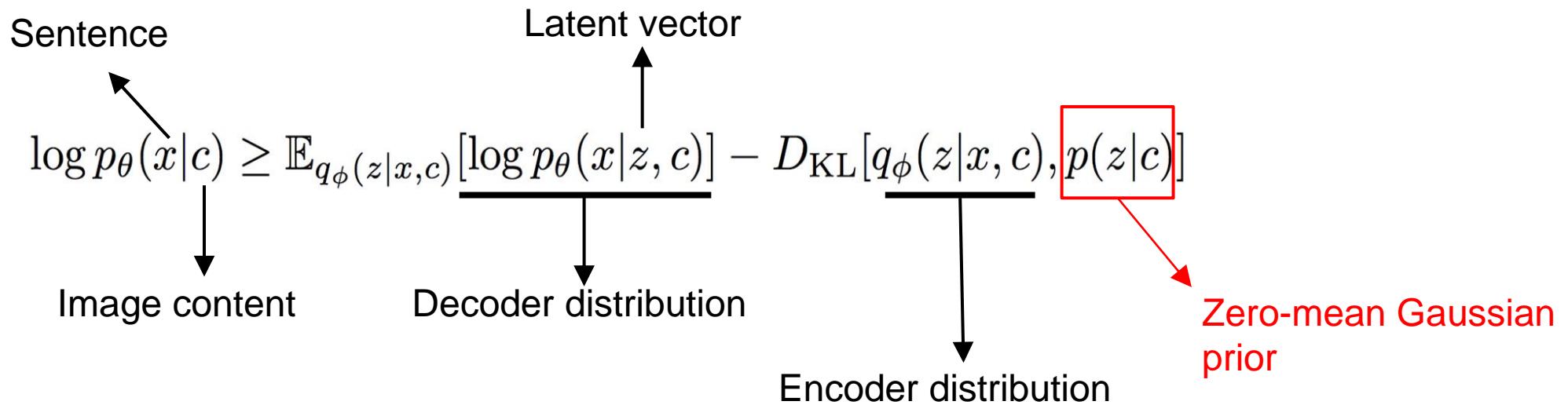
“teddy bear”
“dining table”
“cup”

Decoder



CVAE for captioning

Standard CVAE objective:

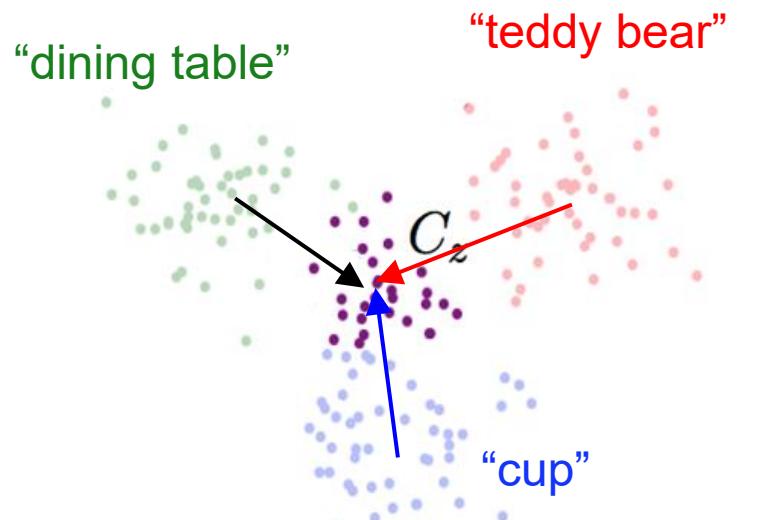


CVAE with additive Gaussian prior

Proposed objective: shift prior mean based on image content

$$\max_{\theta, \phi} \sum_{i=1}^N \log p_\theta(x^i | z^i, c^i) - D_{\text{KL}}[q_\phi(z|x, c), p(z|c)], \quad \text{s.t. } \forall i \ z^i \sim q_\phi(z|x, c).$$

$$p(z|c) = \mathcal{N}\left(z \left| \sum_{k=1}^K c_k \mu_k, \sigma^2 \mathbf{I} \right. \right)$$



Results

- More controllable captions: changing the conditioning vector of object labels changes the caption in a reasonable way



Predicted Object Labels:

'person' 'cup' 'donut' 'dining table'

AG-CVAE:

a woman sitting at a table with a cup of coffee
a person sitting at a table with a cup of coffee
a table with two plates of donuts and a cup of coffee
a woman sitting at a table with a plate of coffee
a man sitting at a table with a plate of food

LSTM Baseline:

a close up of a table with two plates of coffee
a close up of a table with a plate of food
a close up of a plate of food on a table
a close up of a table with two plates of food
a close up of a table with plates of food

Results

- More controllable captions: changing the conditioning vector of object labels changes the caption in a reasonable way



Object Labels: 'person'

AG-CVAE sentences:

a man and a woman standing in a room
a man and a woman are playing a game
a man standing next to a woman in a room
a man standing next to a woman in a field
a man standing next to a woman in a suit

Object Labels: 'person', 'remote'

AG-CVAE sentences:

a man and a woman playing a video game
a man and a woman are playing a video game
a man and woman are playing a video game
a man and a woman playing a game with a remote
a woman holding a nintendo wii game controller

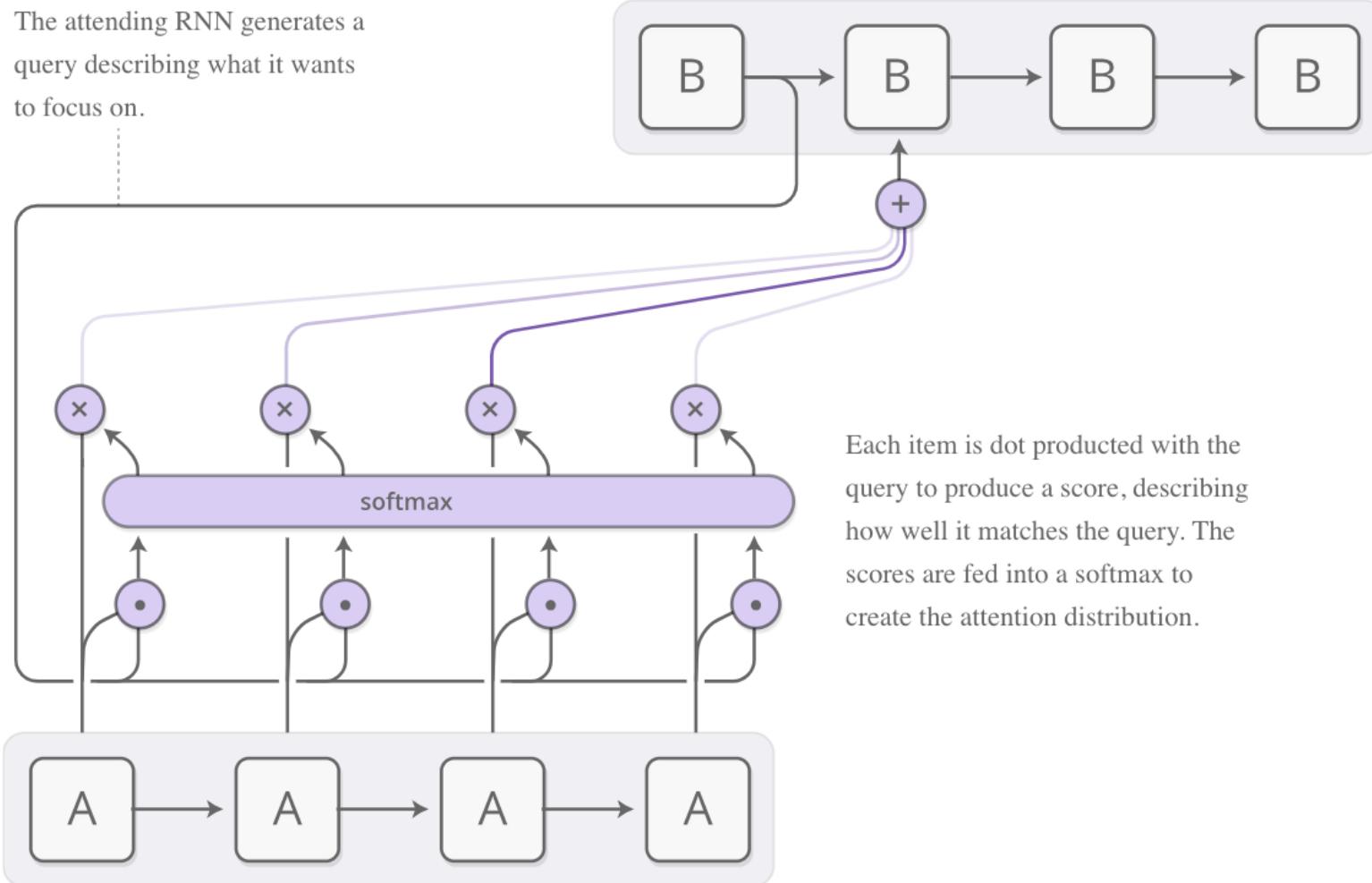
Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Sequence-to-sequence models with attention

The attending RNN generates a query describing what it wants to focus on.

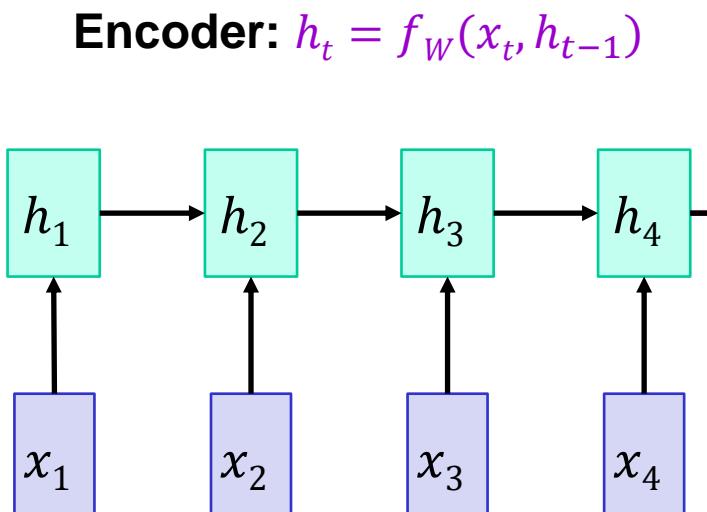


Outline

- Vanilla seq2seq with RNNs
- Seq2seq with RNNs and attention
- Image captioning with attention
- Convolutional seq2seq with attention

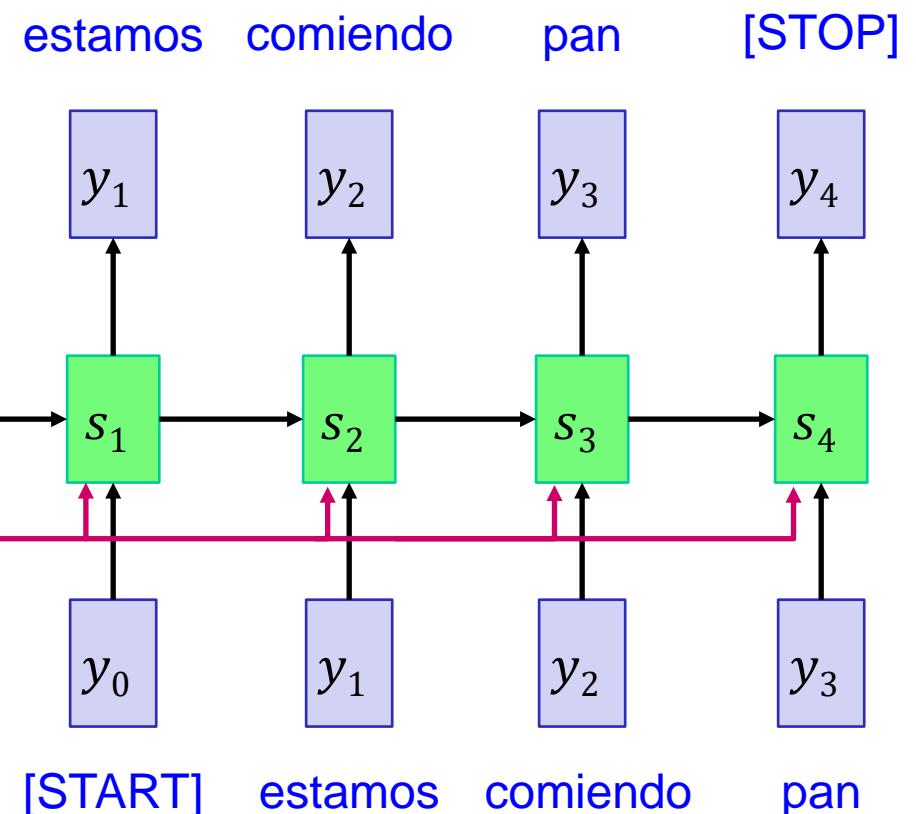
Sequence-to-sequence with RNNs

English to Spanish

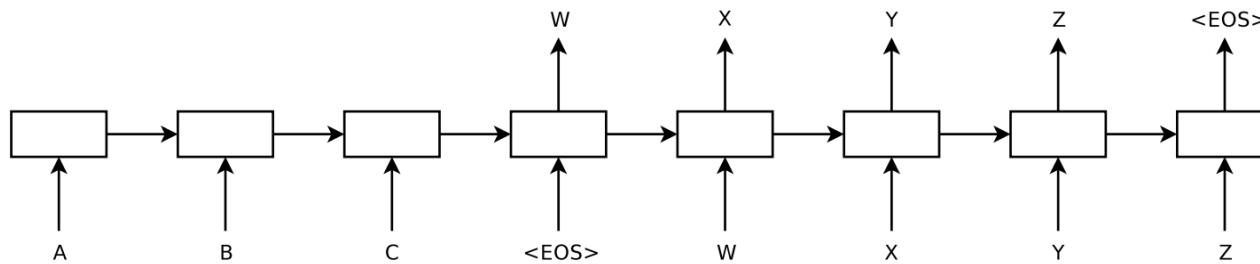


From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c = h_T$)

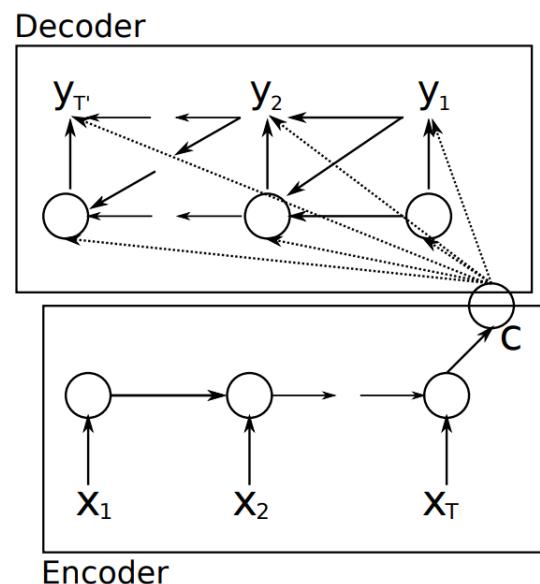
Decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c)$



Sequence-to-sequence with RNNs

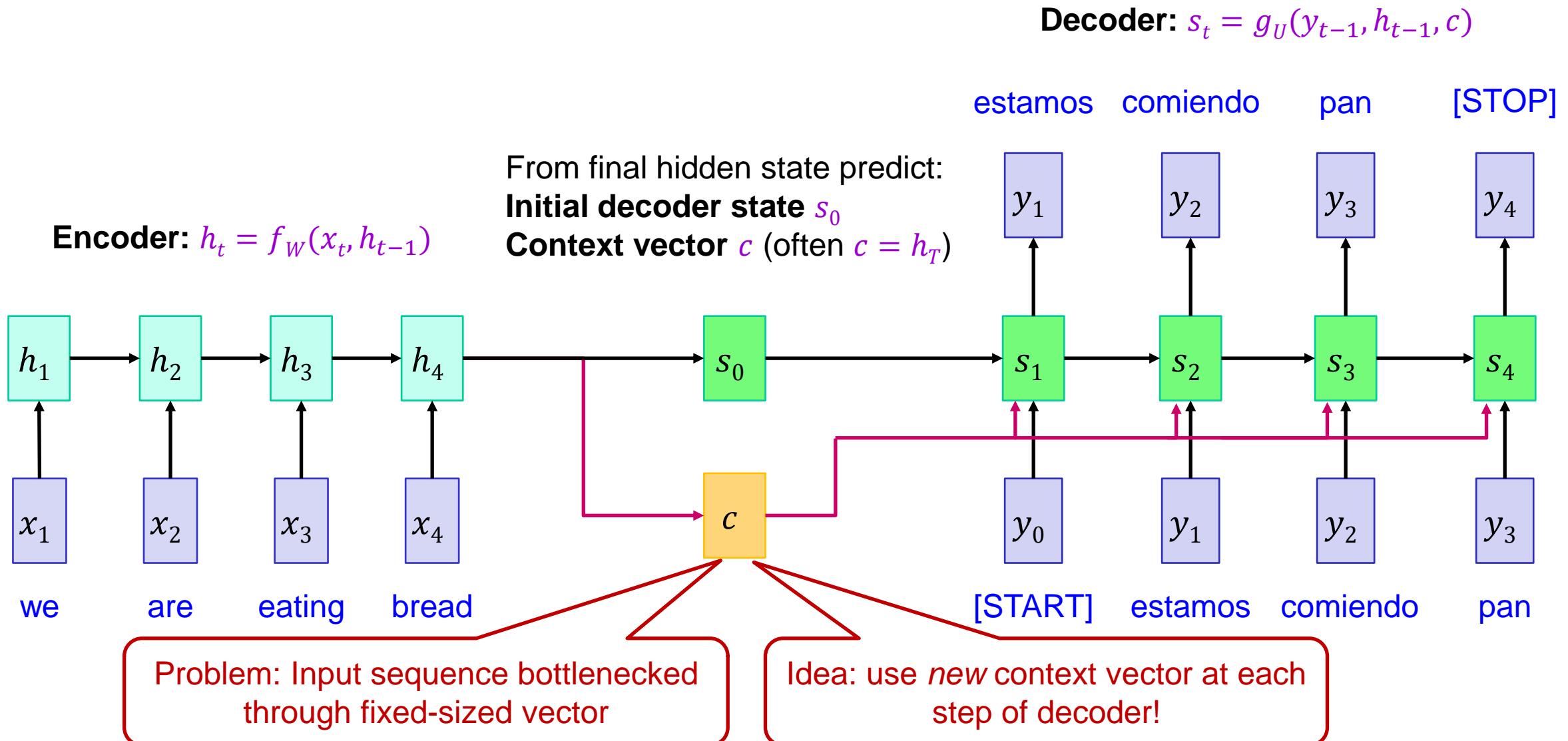


I. Sutskever, O. Vinyals, Q. Le, [Sequence to Sequence Learning with Neural Networks](#), NeurIPS 2014



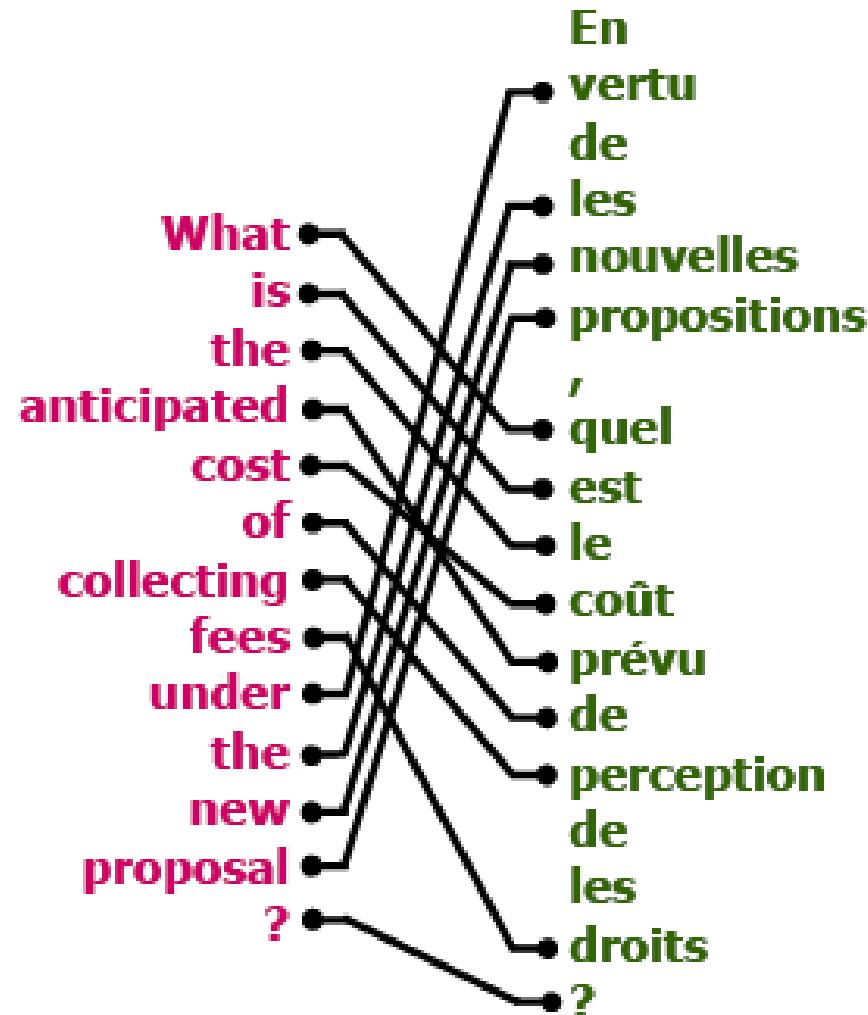
K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, [Learning phrase representations using RNN encoder-decoder for statistical machine translation](#), ACL 2014

Sequence-to-sequence with RNNs



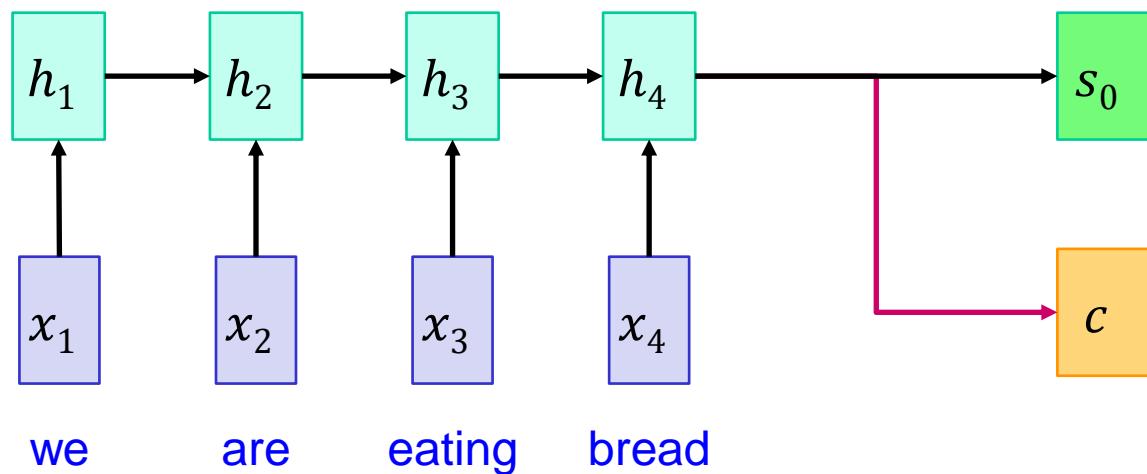
Sequence-to-sequence with RNNs and attention

- Intuition: translation requires *alignment*

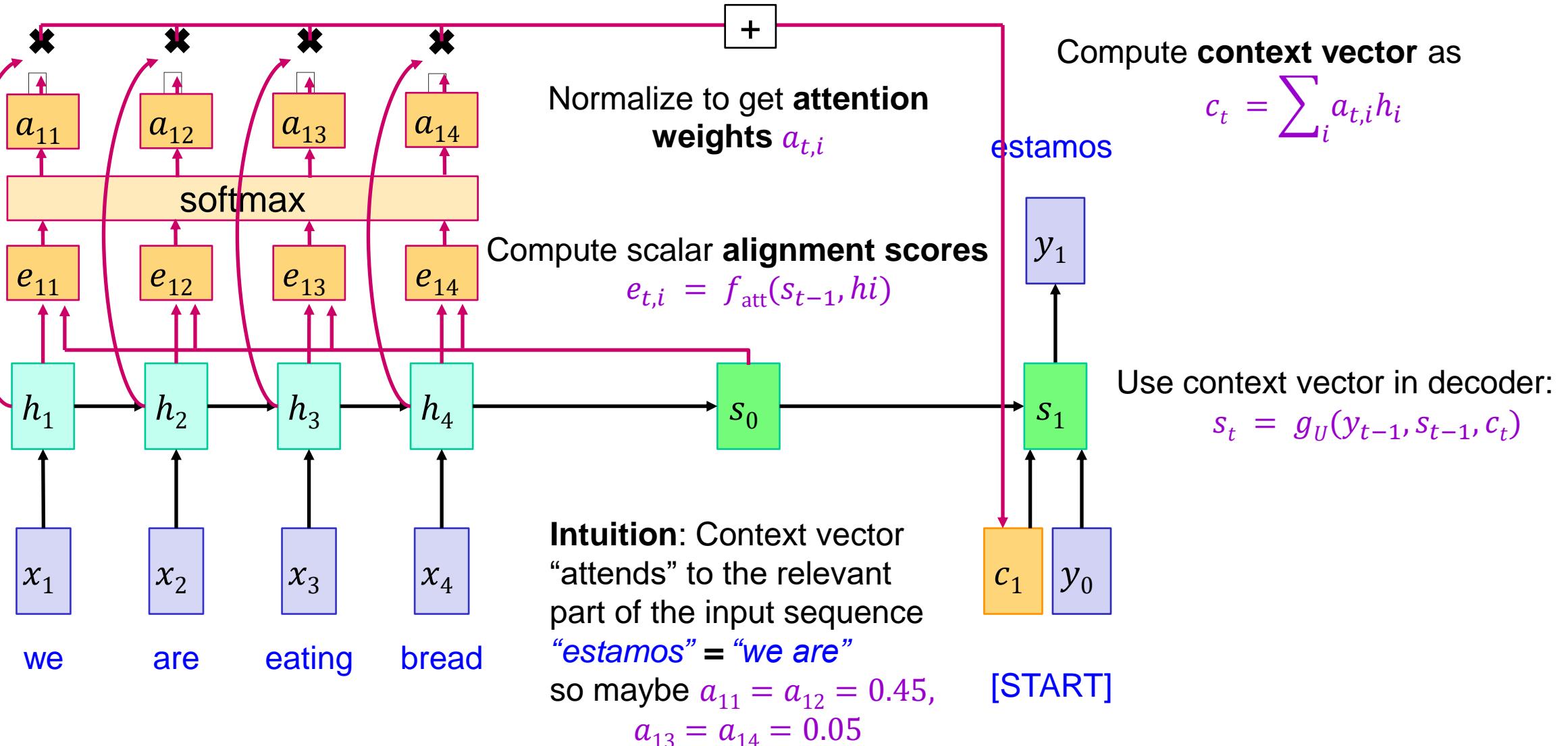


Sequence-to-sequence with RNNs and attention

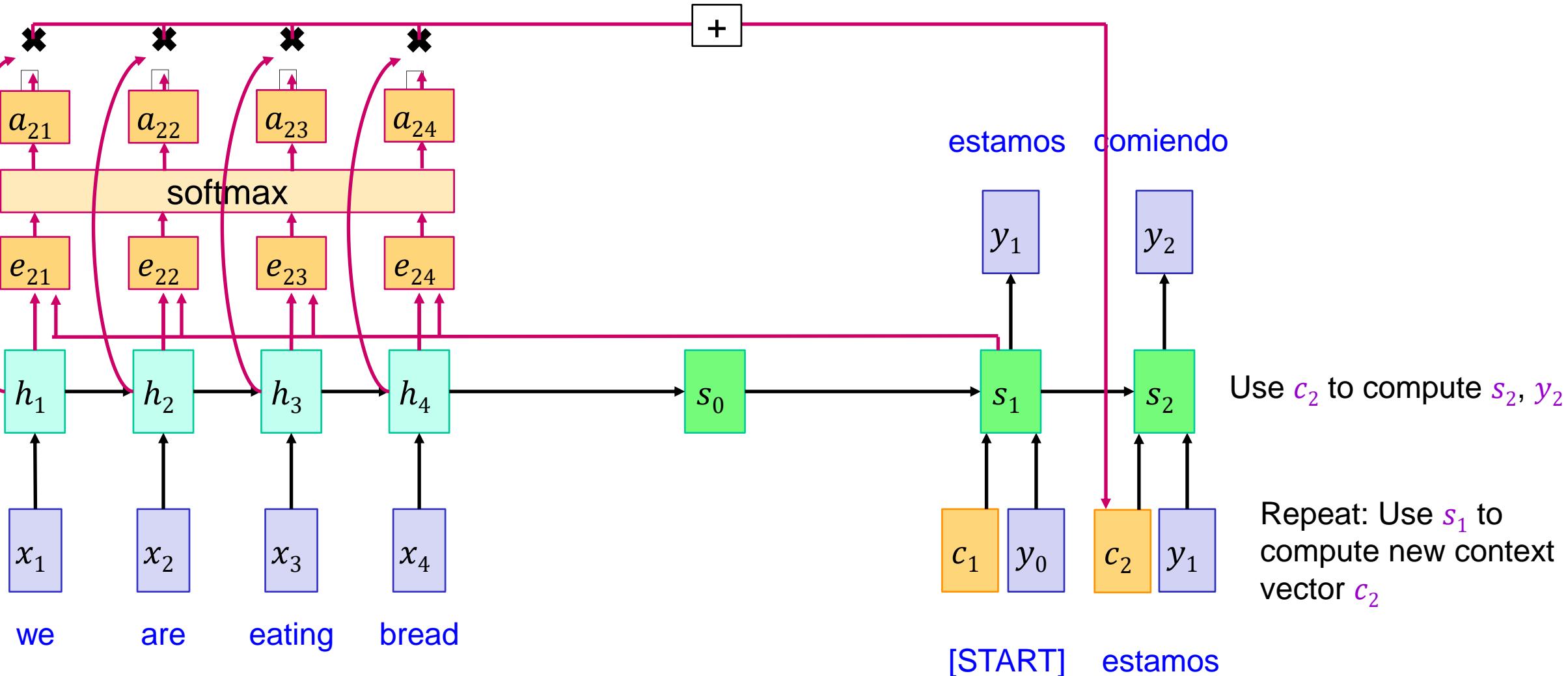
- At each timestep of decoder, context vector “looks at” different parts of the input sequence



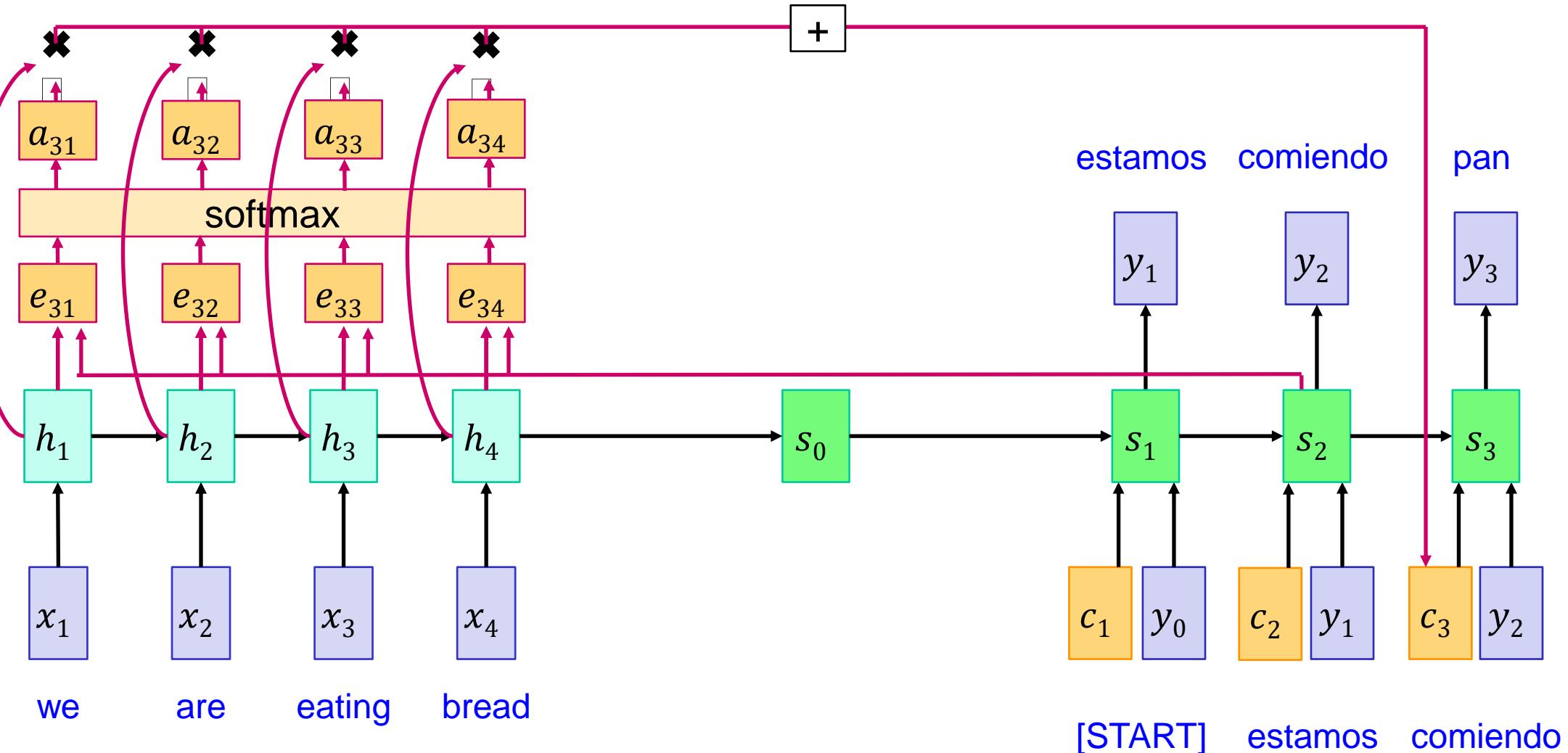
Sequence-to-sequence with RNNs and attention



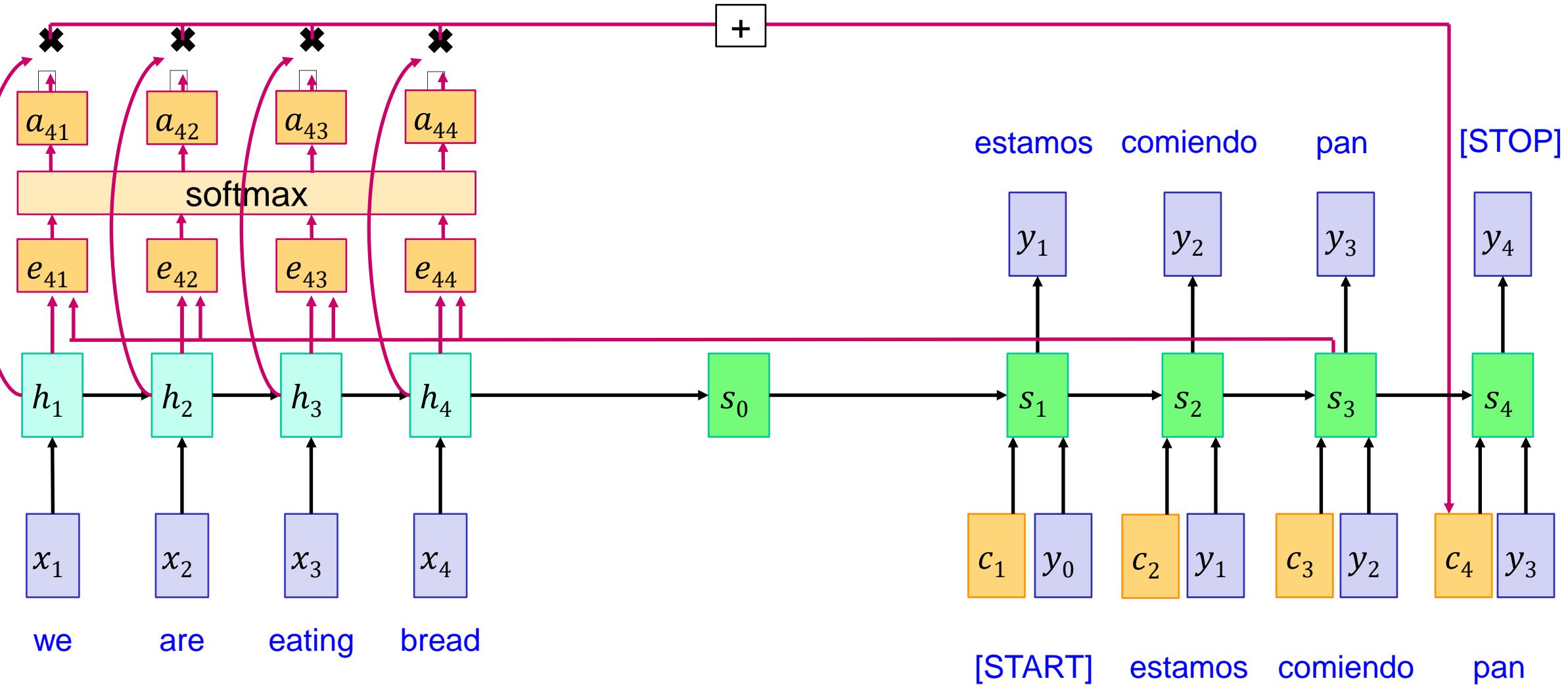
Sequence-to-sequence with RNNs and attention



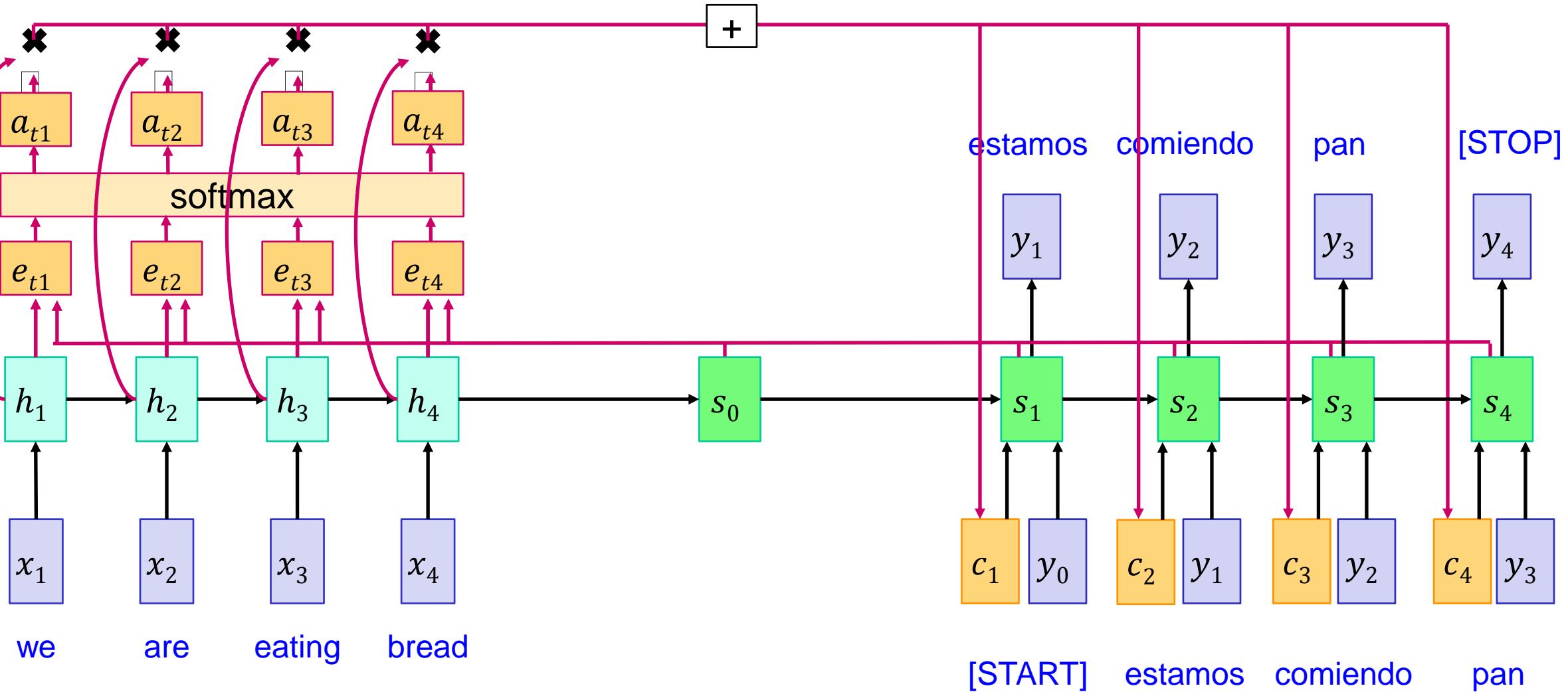
Sequence-to-sequence with RNNs and attention



Sequence-to-sequence with RNNs and attention

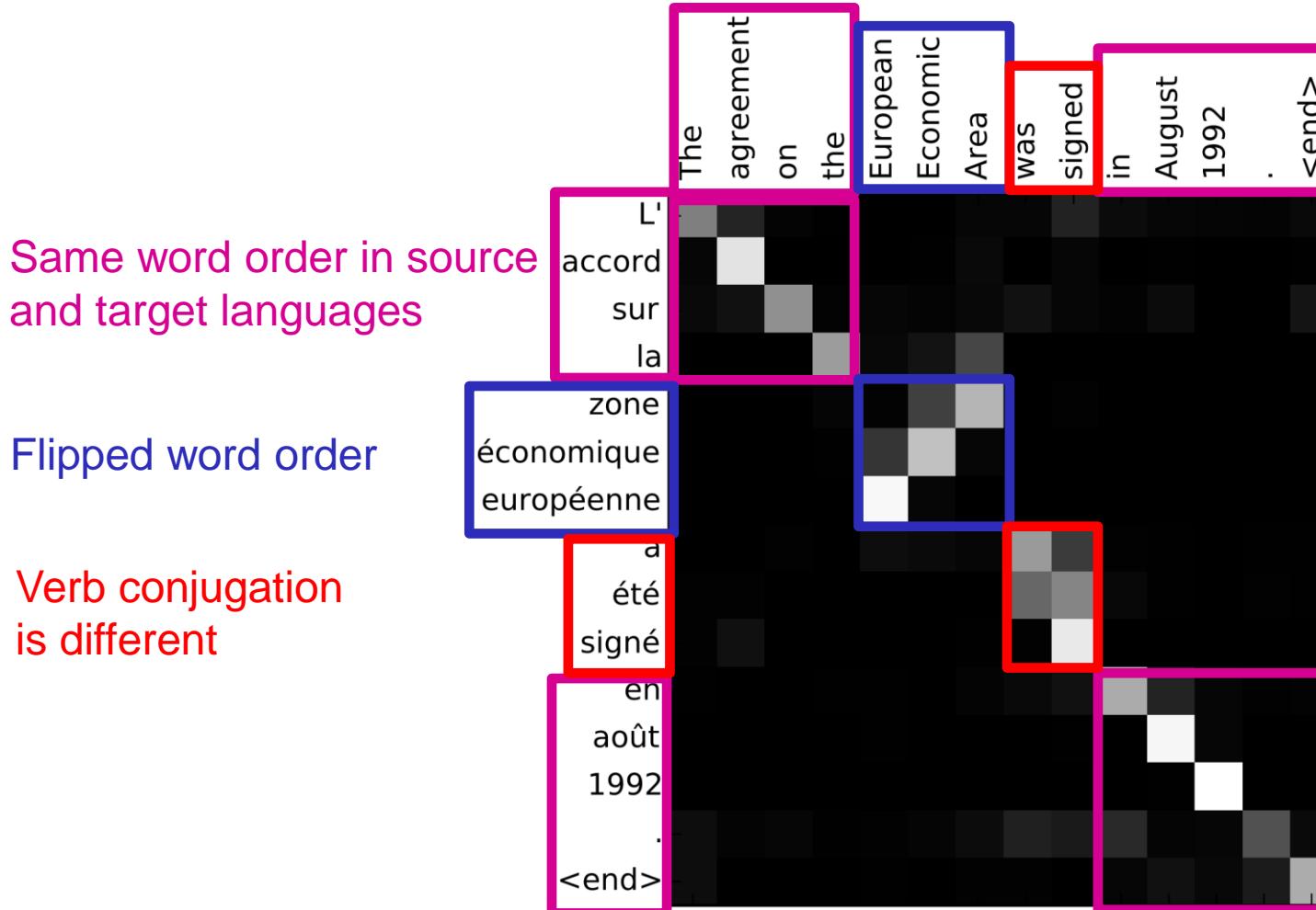


Sequence-to-sequence with RNNs and attention

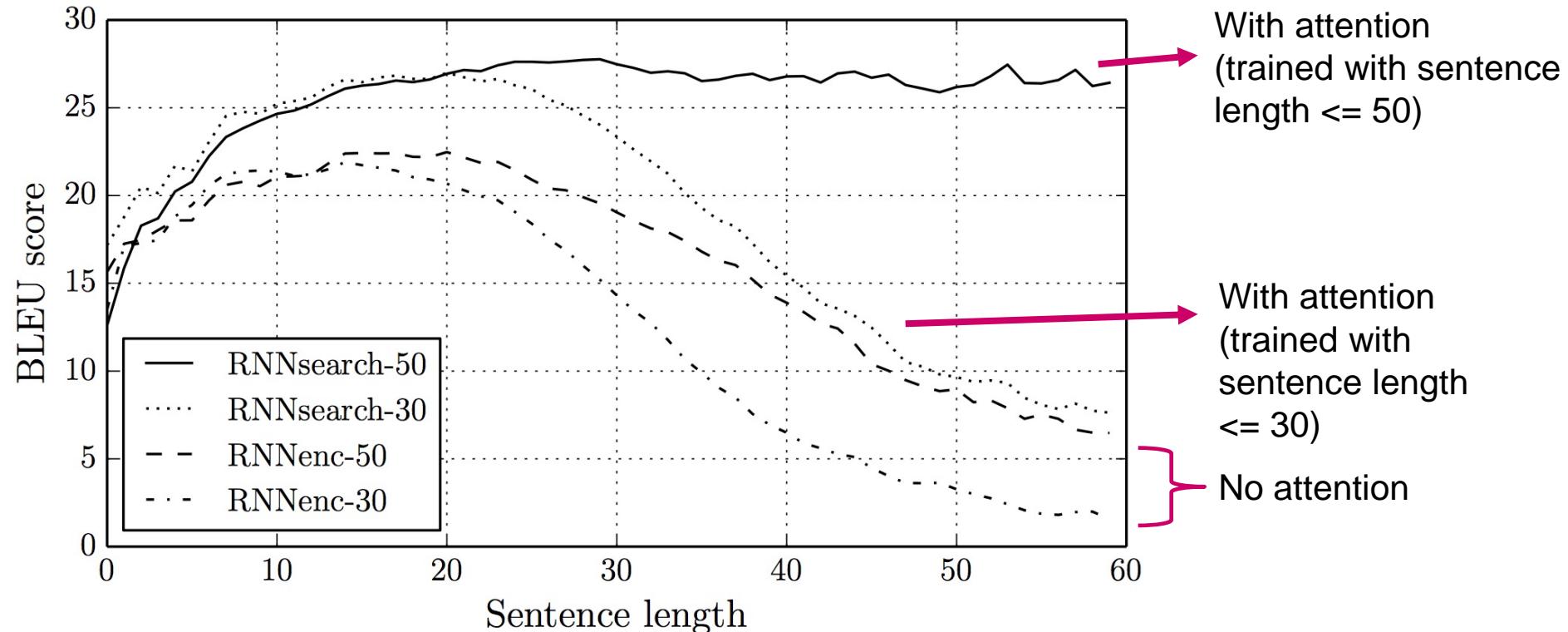


Sequence-to-sequence with RNNs and attention

- Visualizing attention weights:



Quantitative evaluation



Google Neural Machine Translation (GNMT)

Google's Neural Machine Translation System: Bridging the Gap
between Human and Machine Translation

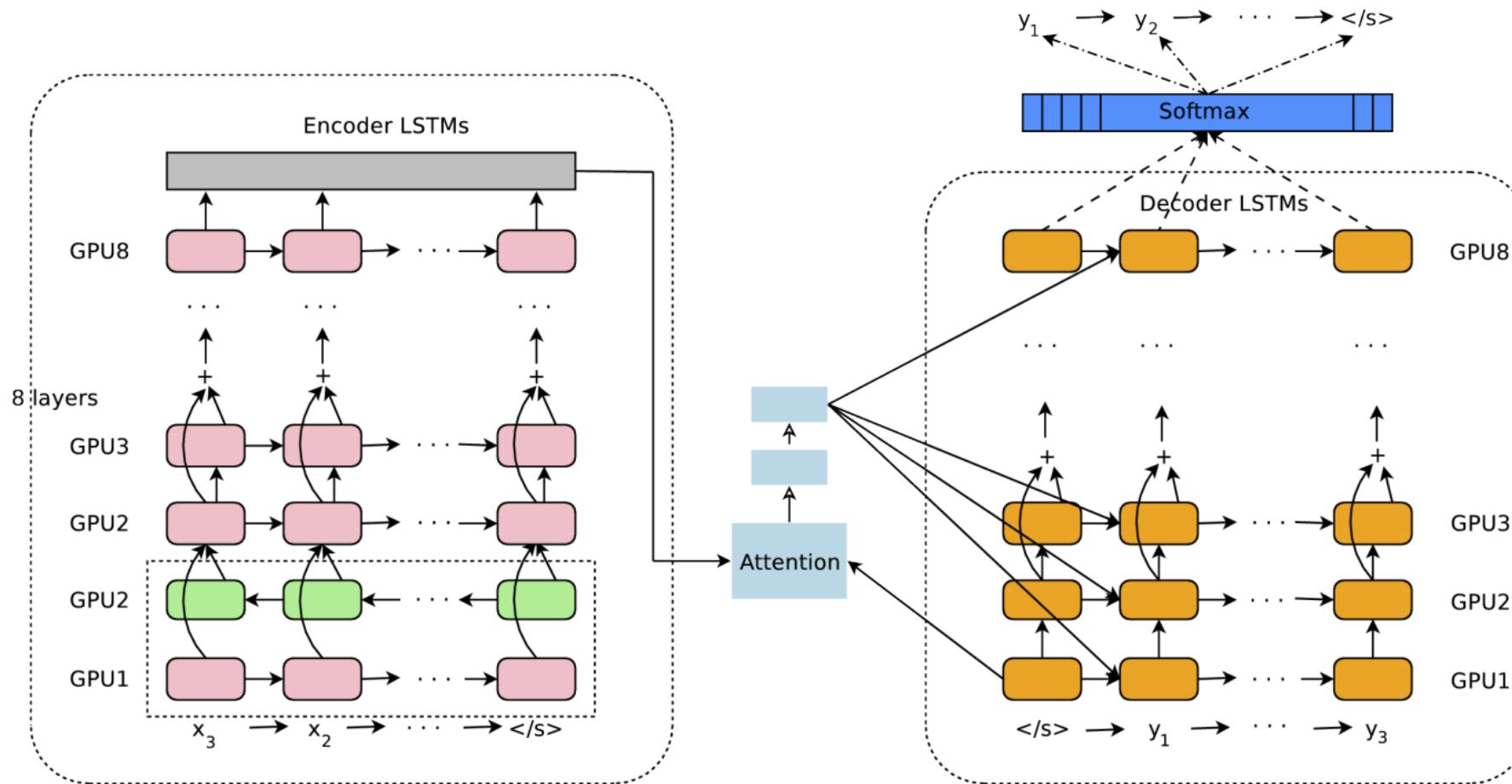
Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
`yonghui,schuster,zhifengc,qvl,mnorouzi@google.com`

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey,
Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser,
Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens,
George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa,
Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

Y. Wu et al., [Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation](#), arXiv 2016

<https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html>

Google Neural Machine Translation (GNMT)



Google Neural Machine Translation (GNMT)

- **Standard training objective:** maximize log-likelihood of ground truth output given input:

$$\sum_i \log P_W(Y_i^* | X_i)$$

- Not related to task-specific reward function (e.g., BLEU score)
- Does not encourage “better” incorrect sentences to get better likelihood
- **Refinement objective:** expectation of rewards over possible predicted sentences Y :

$$\sum_i \sum_Y P_W(Y | X_i) R(Y, Y_i^*)$$

- Use variant of BLEU score to compute reward
- Reward is not differentiable -- need *reinforcement learning* to train (initialize with ML-trained model)

Google Neural Machine Translation (GNMT)

- Human evaluation results on production data (500 randomly sampled sentences from Wikipedia and news websites)

Table 10: Mean of side-by-side scores on production data

	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.550	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

Side-by-side scores: range from 0 (“completely nonsense translation”) to 6 (“perfect translation”), produced by human raters fluent in both languages

PBMT: Translation by phrase-based statistical translation system used by Google

GNMT: Translation by GNMT system

Human: Translation by humans fluent in both languages

Outline

- Vanilla seq2seq with RNNs
- Seq2seq with RNNs and attention
- Image captioning with attention

Generalizing attention

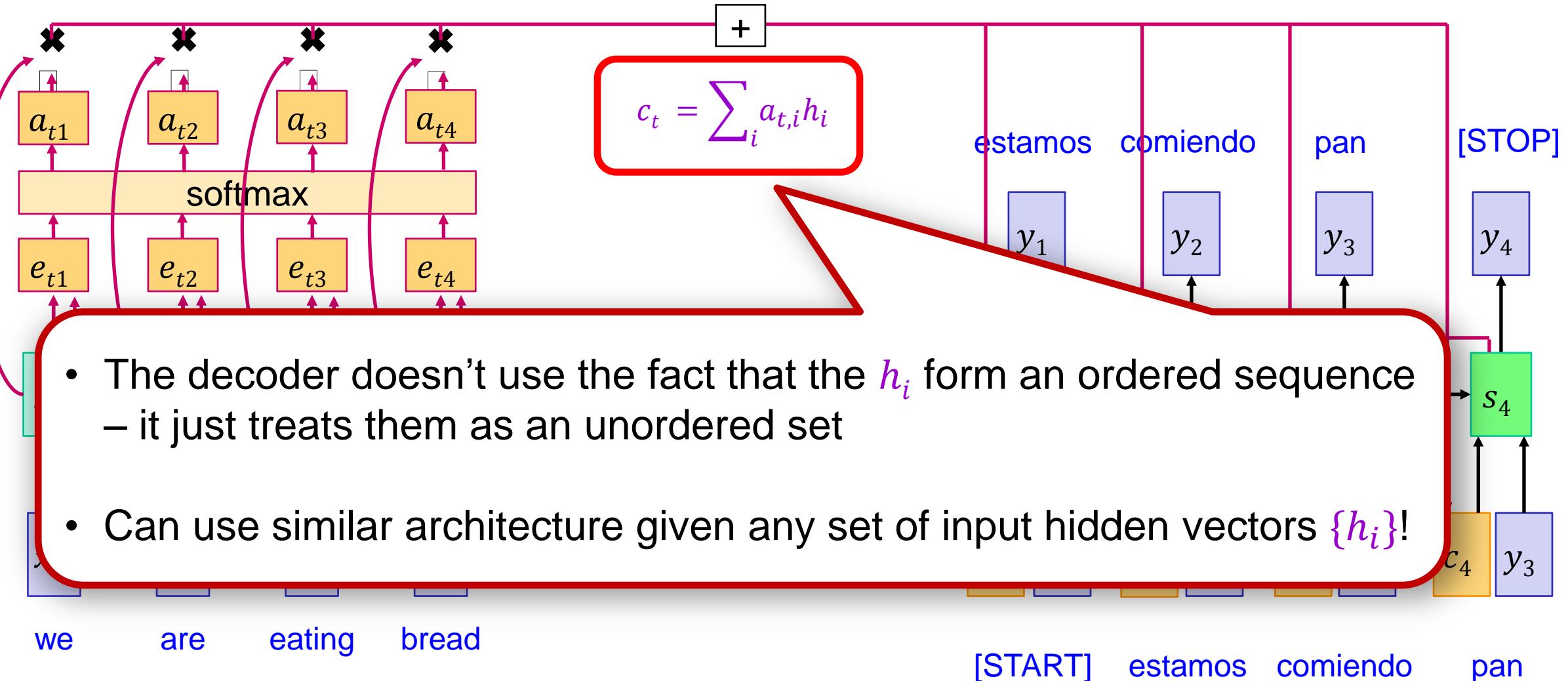


Image captioning with RNNs and attention

- Idea: pay attention to different parts of the image when generating different words
- Automatically learn this grounding of words to image regions without direct supervision

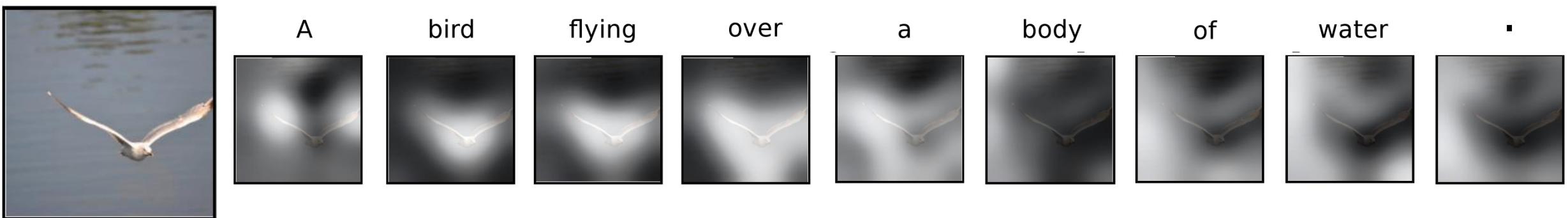
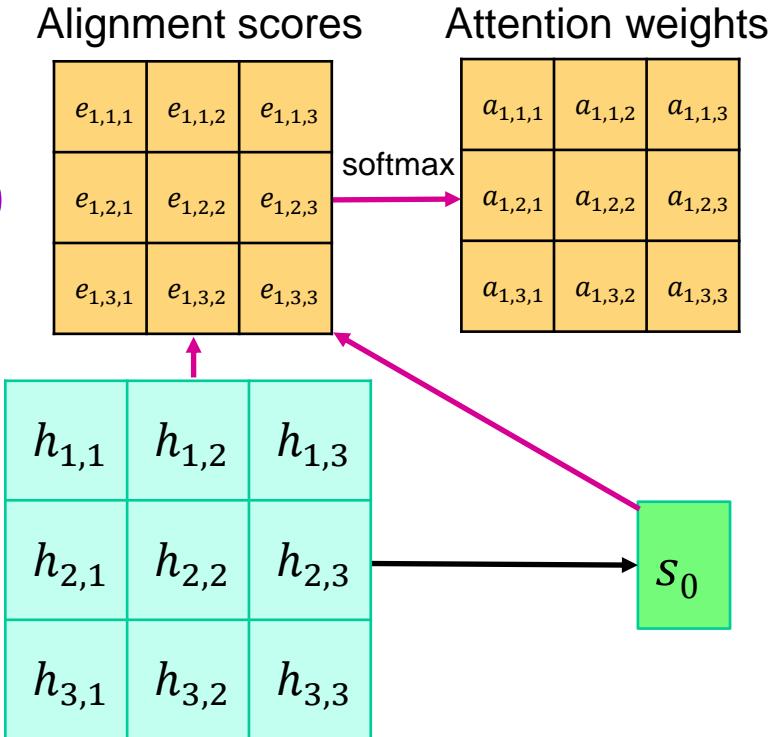


Image captioning with RNNs and attention

$$e_{t,i,j} = f_{\text{att}}(s_{t-1}, h_{i,j})$$



Use a CNN to compute a grid of features for an image

Image captioning with RNNs and attention

$$e_{t,i,j} = f_{\text{att}}(s_{t-1}, h_{i,j})$$



Use a CNN to compute a grid of features for an image

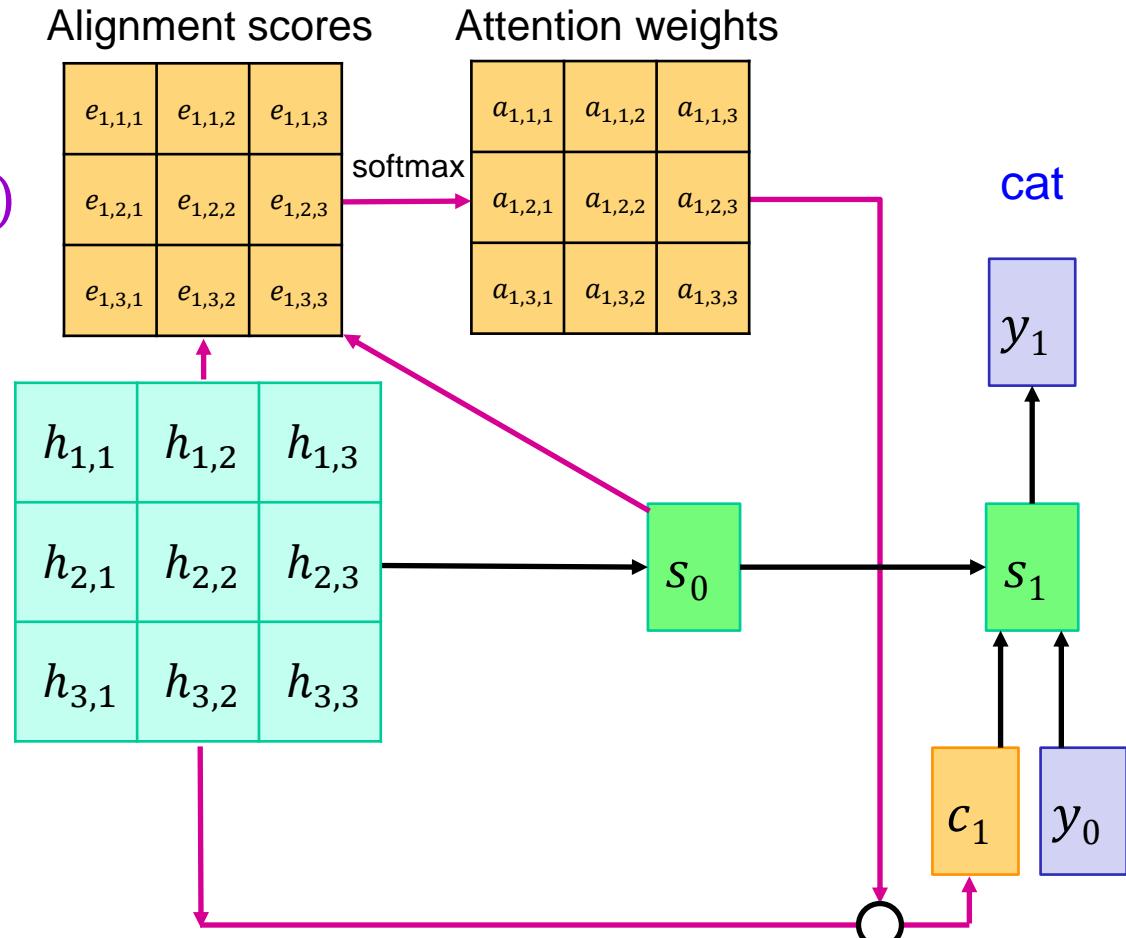
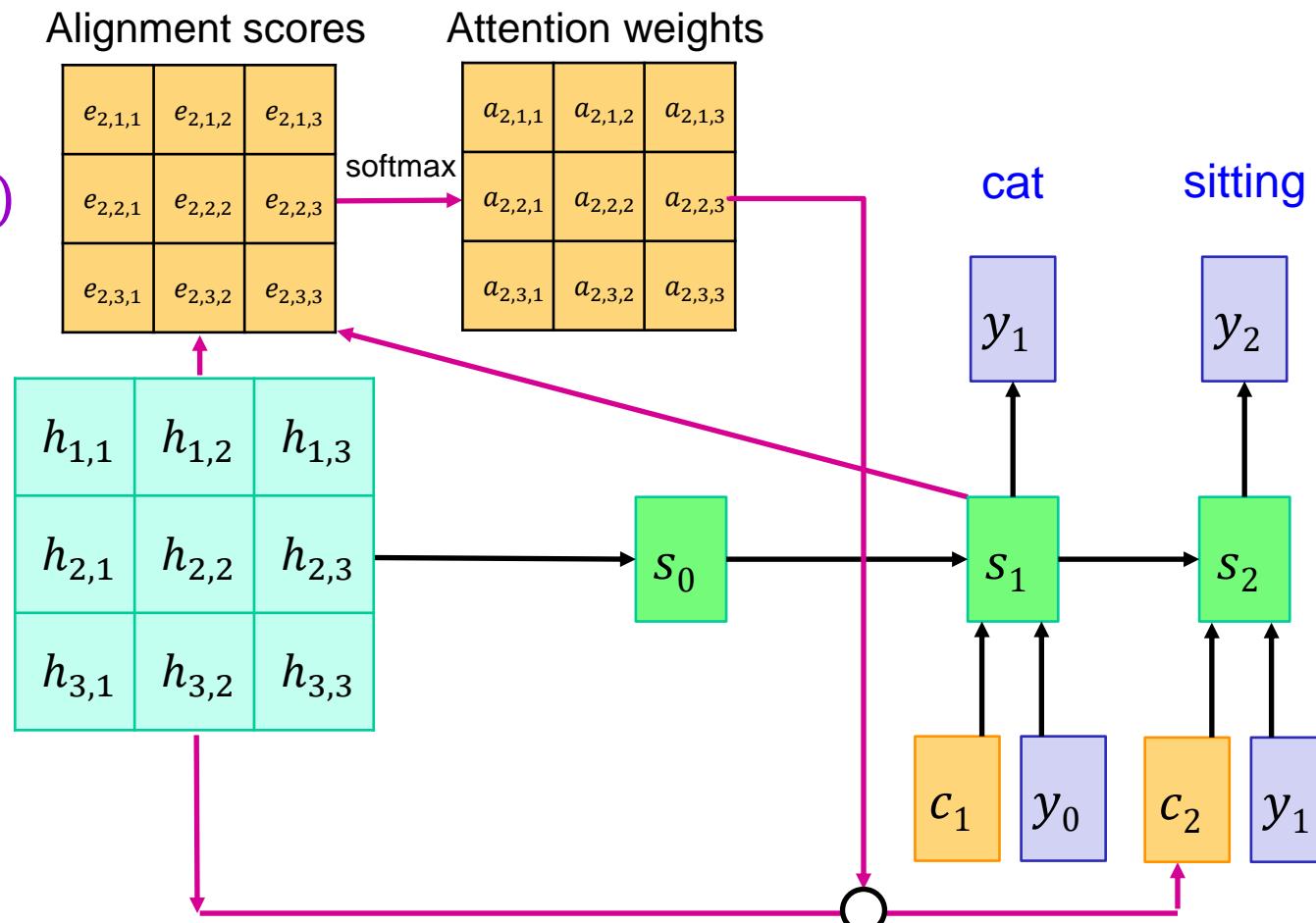


Image captioning with RNNs and attention

$$e_{t,i,j} = f_{\text{att}}(s_{t-1}, h_{i,j})$$



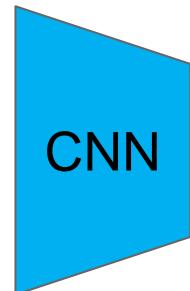
Use a CNN to compute a grid of features for an image



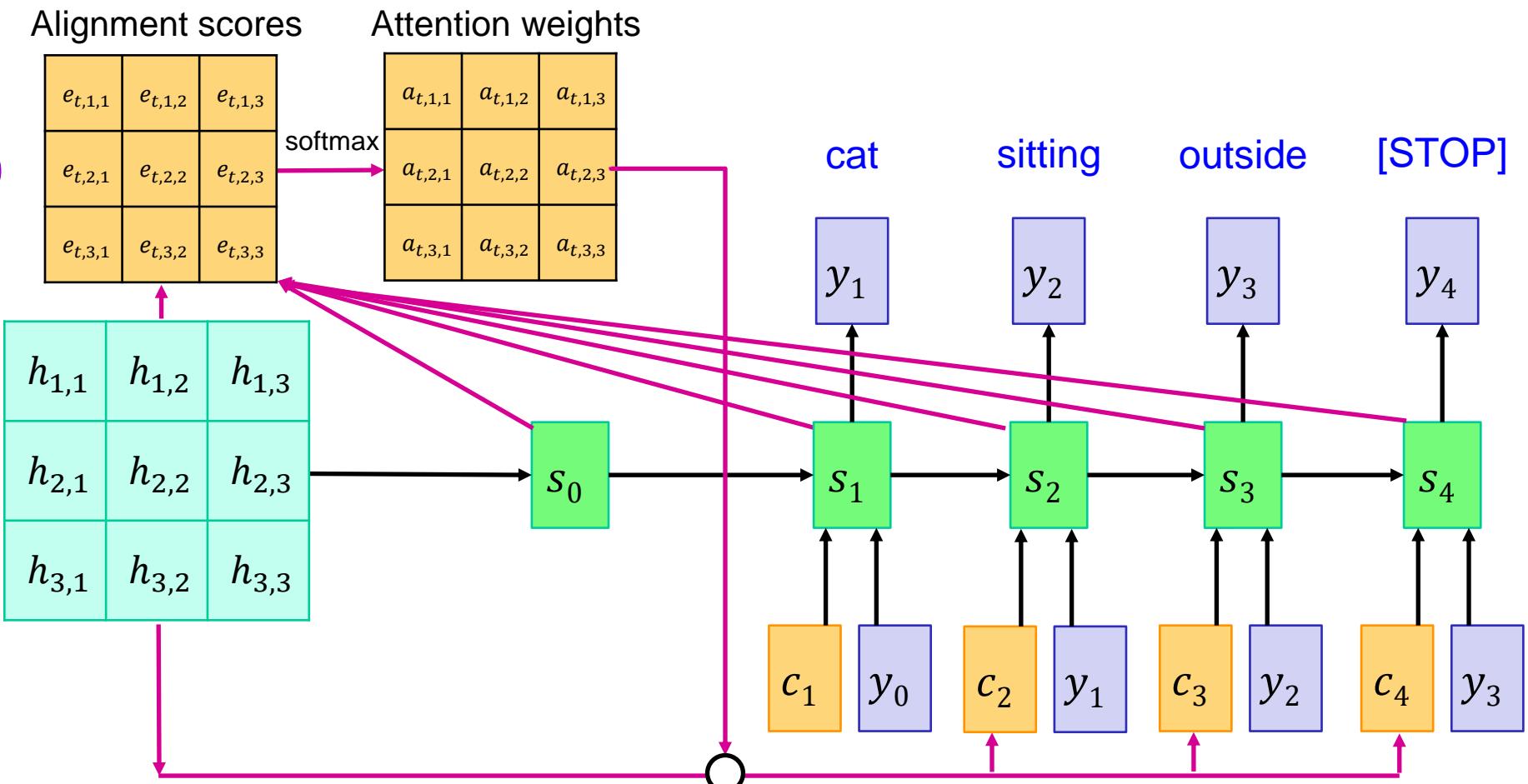
$$c_t = \sum_i a_{t,i,j} h_i$$

Image captioning with RNNs and attention

$$e_{t,i,j} = f_{\text{att}}(s_{t-1}, h_{i,j})$$



Use a CNN to compute a grid of features for an image



$$c_t = \sum_i a_{t,i,j} h_i$$

Each time step of decoder uses a different context vector that looks at different parts of the input image

Example results

- Good captions



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Example results

- Mistakes



A large white bird standing in a forest.



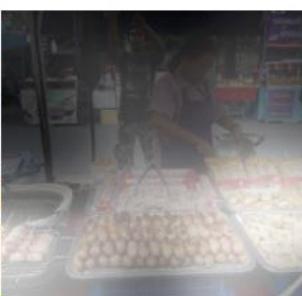
A woman holding a clock in her hand.



A man wearing a hat and
a hat on a skateboard.



A person is standing on a beach
with a surfboard.



A woman is sitting at a table
with a large pizza.



A man is talking on his cell phone
while another man watches.

Quantitative results

Dataset	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Flickr8k	Google NIC	63	41	27	-	-
	Soft-Attention	67	44.8	29.9	19.5	18.93
	Hard-Attention	67	45.7	31.4	21.3	20.30
Flickr30k	Google NIC	66.3	42.3	27.7	18.3	-
	Soft-Attention	66.7	43.4	28.8	19.1	18.49
	Hard-Attention	66.9	43.9	29.6	19.9	18.46
COCO	Google NIC	66.6	46.1	32.9	24.6	-
	Soft-Attention	70.7	49.2	34.4	24.3	23.90
	Hard-Attention	71.8	50.4	35.7	25.0	23.04

Soft attention is when we calculate the context vector as a weighted sum of the encoder hidden states.

Hard attention is when, instead of weighted average of all hidden states, we use **attention scores** to select a single hidden state.

X, Attend, and Y

“Show, attend, and tell” (*Xu et al, ICML 2015*)

Look at image, attend to image regions, produce question

“Ask, attend, and answer” (*Xu and Saenko, ECCV 2016*)

“Show, ask, attend, and answer” (*Kazemi and Elqursh, 2017*)

Read text of question, attend to image regions, produce answer

“Listen, attend, and spell” (*Chan et al, ICASSP 2016*)

Process raw audio, attend to audio regions while producing text

“Listen, attend, and walk” (*Mei et al, AAAI 2016*)

Process text, attend to text regions, output navigation commands

“Show, attend, and interact” (*Qureshi et al, ICRA 2017*)

Process image, attend to image regions, output robot control commands

“Show, attend, and read” (*Li et al, AAAI 2019*)

Process image, attend to image regions, output text

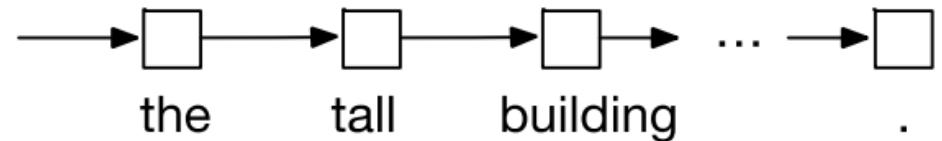
Outline

- Vanilla seq2seq with RNNs
- Seq2seq with RNNs and attention
- Image captioning with attention
- Convolutional seq2seq with attention

Recurrent vs. convolutional sequence models

- **Recurrent models:**

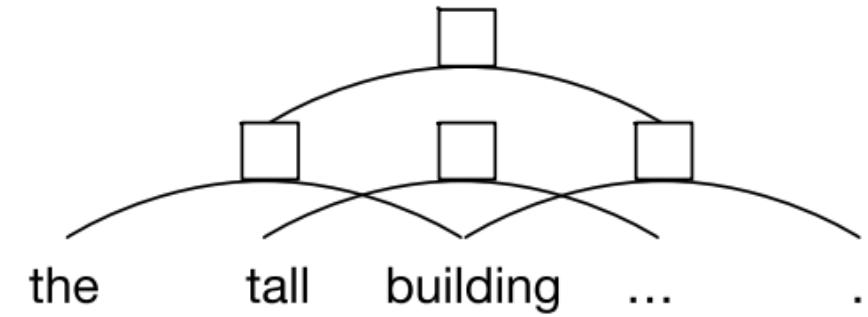
- Treat input as ordered sequence (inherently sequential processing)
- Build up context using the hidden vector



- **Convolutional models:**

- Treat input as a grid indexed by time and feature dimension
- Build up context using multiple layers of convolutions
- Processing can be parallel at training time, but convolutions must be *causal*

(A filter is called causal if the filter output does not depend on future inputs.)



[Image source](#) (Facebook AI Research)

WaveNet

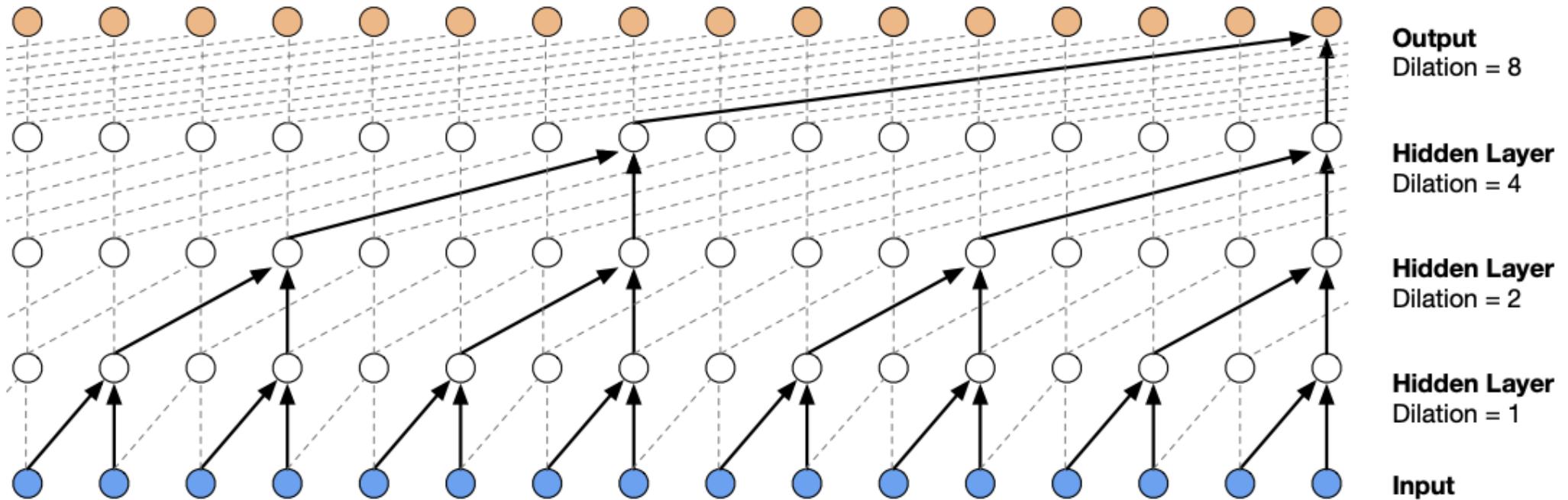
- Goal: generate raw audio
 - Represented as sequence of 16-bit integer values (can be quantized to 256 discrete levels), 16K samples per second
- Applications: text-to-speech, music generation
 - Also works for speech recognition



Figure 1: A second of generated speech.

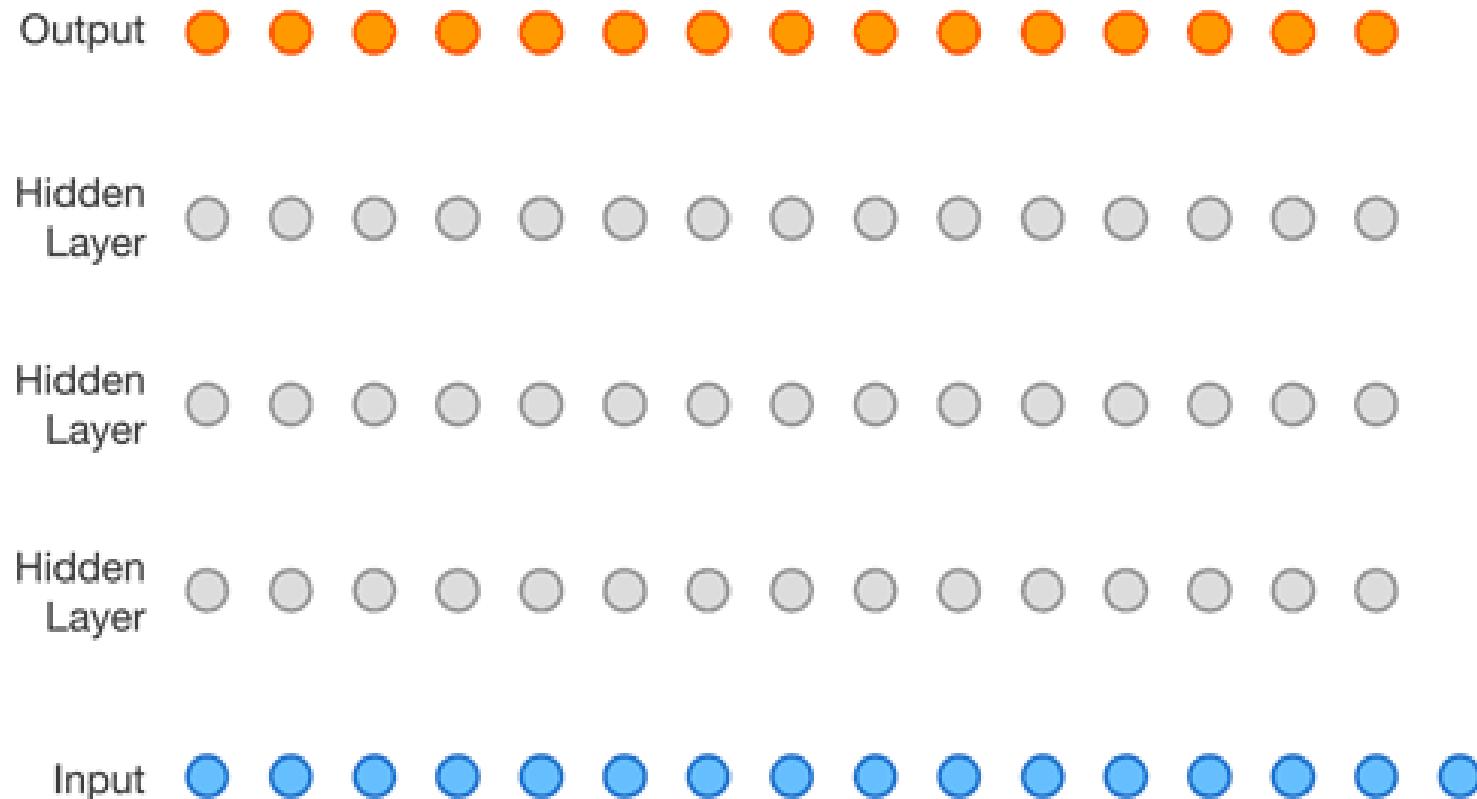
WaveNet

- Training time: compute predictions of all timesteps in parallel (conditioned on ground truth)



WaveNet

- Test time: feed each predicted sample back into the model to make prediction at next timestep



WaveNet: Results

- Text-to-speech with different speaker identities:



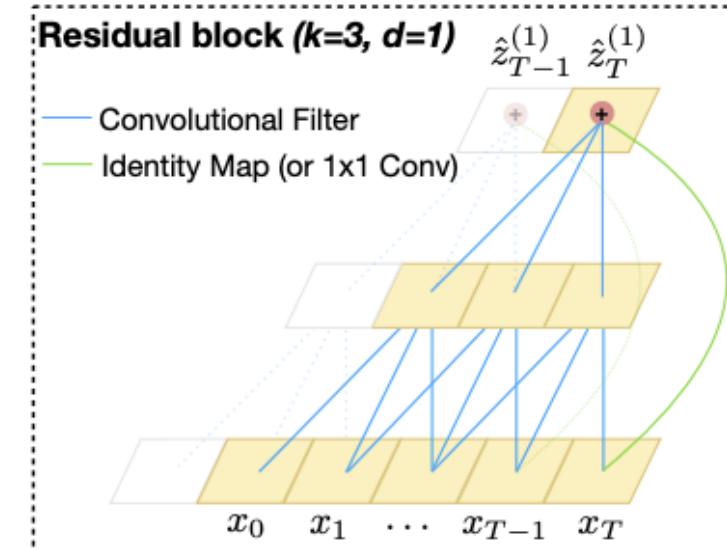
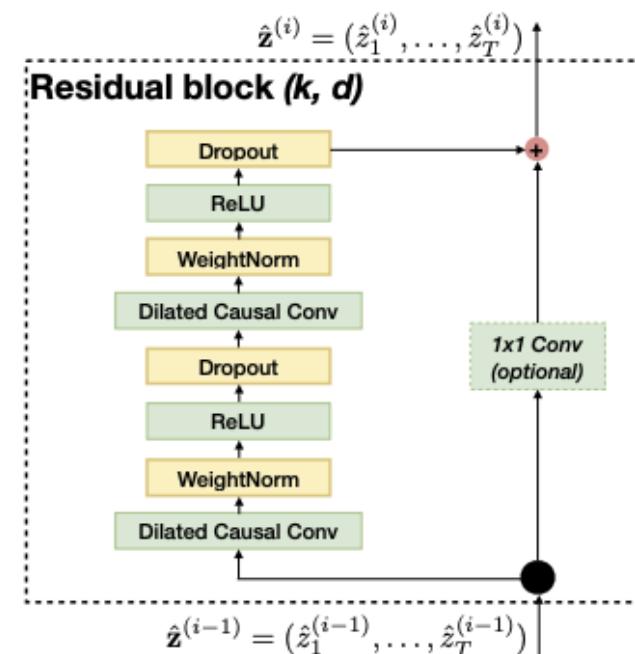
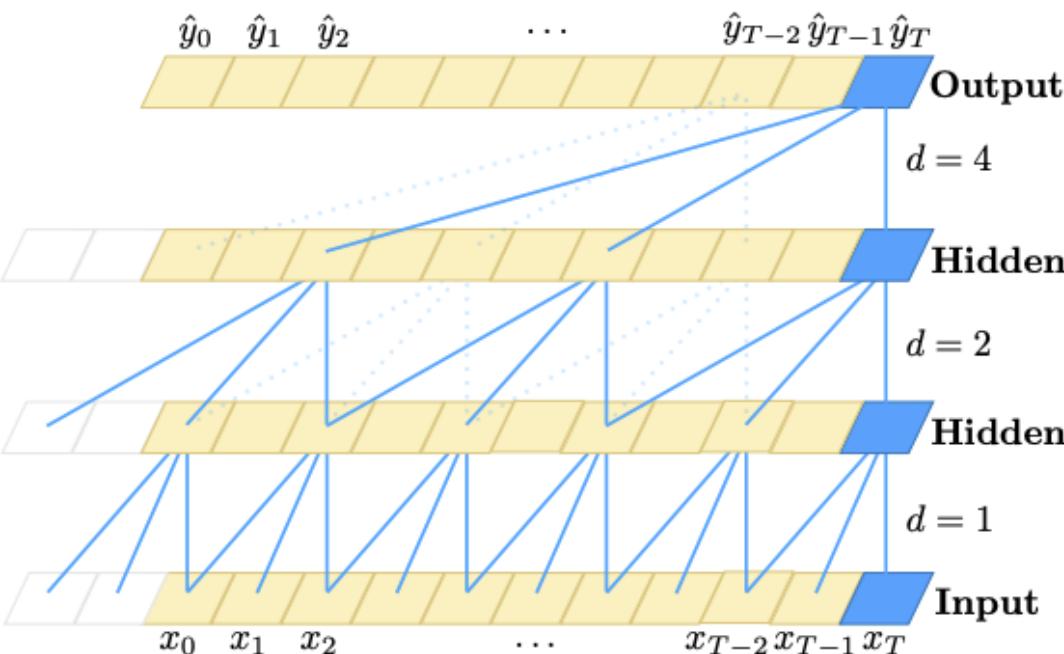
- Generated sample of classical piano music:



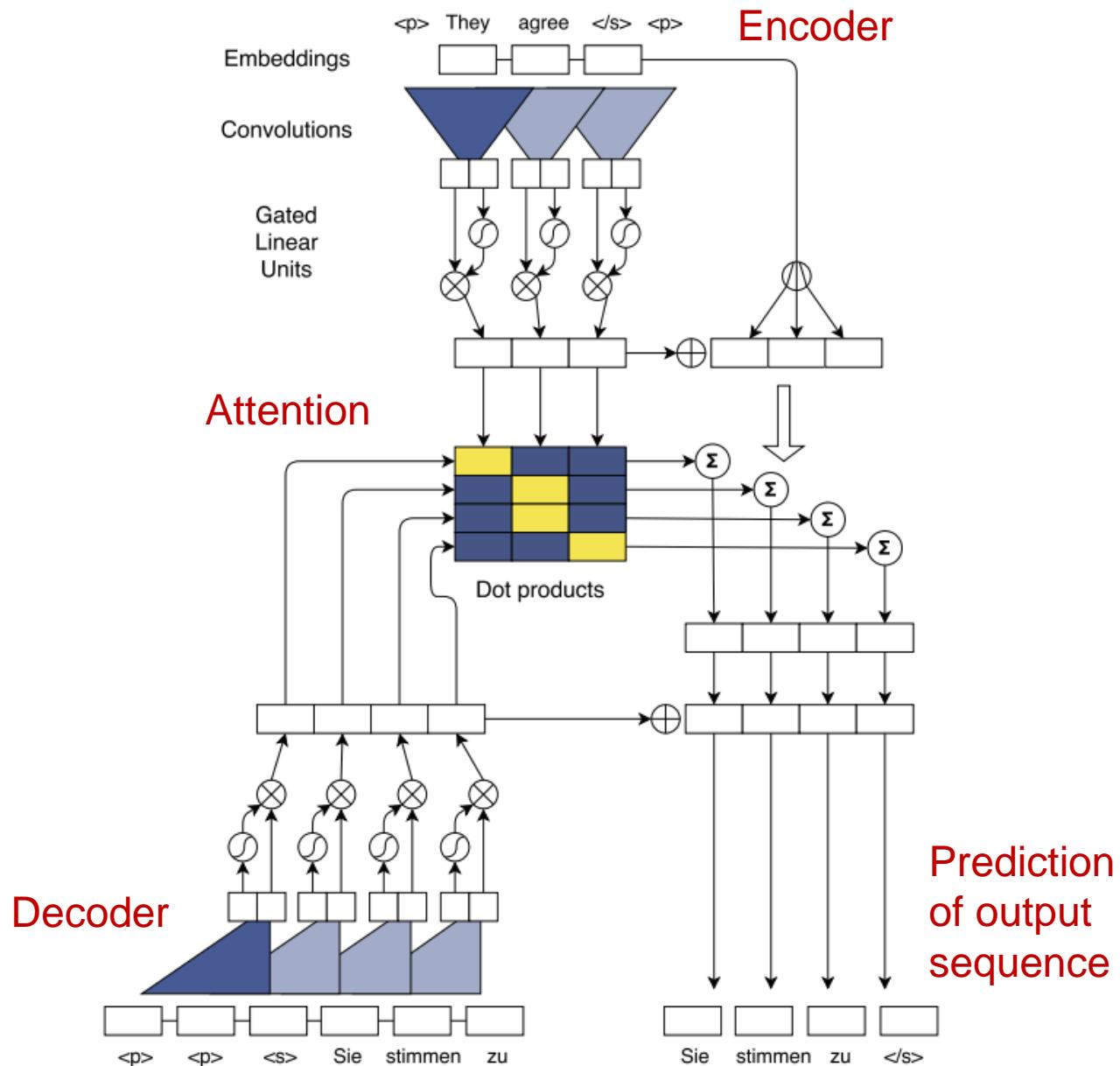
<https://deepmind.com/blog/article/wavenet-generative-model-raw-audio>

Temporal convolutional networks (TCNs)

- TCNs can be competitive with RNNs for a variety of sequence modeling tasks



Convolutional seq2seq with attention



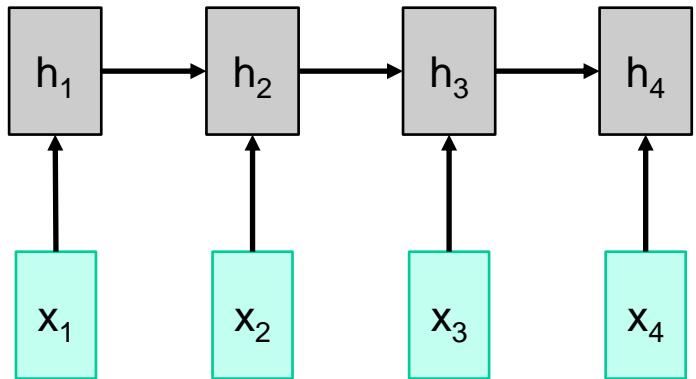
WMT'14 English-German	BLEU
Luong et al. (2015) LSTM (Word 50K)	20.9
Kalchbrenner et al. (2016) ByteNet (Char)	23.75
Wu et al. (2016) GNMT (Word 80K)	23.12
Wu et al. (2016) GNMT (Word pieces)	24.61
ConvS2S (BPE 40K)	25.16

WMT'14 English-French	BLEU
Wu et al. (2016) GNMT (Word 80K)	37.90
Wu et al. (2016) GNMT (Word pieces)	38.95
Wu et al. (2016) GNMT (Word pieces) + RL	39.92
ConvS2S (BPE 40K)	40.51

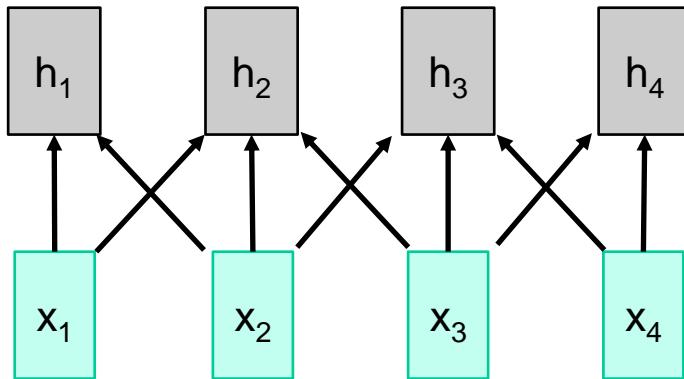
J. Gehring, M. Auli, D. Grangier, D. Yarats, Y. Dauphin, [Convolutional sequence to sequence learning](#), ICML 2017

Different ways of processing sequences

RNN



1D convolutional network



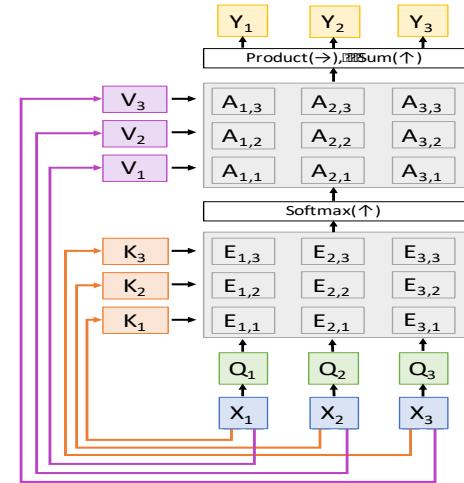
Works on **ordered sequences**

- Pros: Good at long sequences:
After one RNN layer, h_T "sees"
the whole sequence
- Cons: Not parallelizable: need
to compute hidden states
sequentially

Works on **multidimensional grids**

- Con: Bad at long sequences:
Need to stack many conv layers
for outputs to "see" the whole
sequence
- Pro: Highly parallel: Each output
can be computed in parallel

Self-attention and
Transformer



• Works on **sets of vectors**

- Pro: Good at long sequences:
after one self-attention layer,
each output "sees" all inputs!
- Pro: Highly parallel: Each
output can be computed in
parallel
- Con: Very memory-intensive

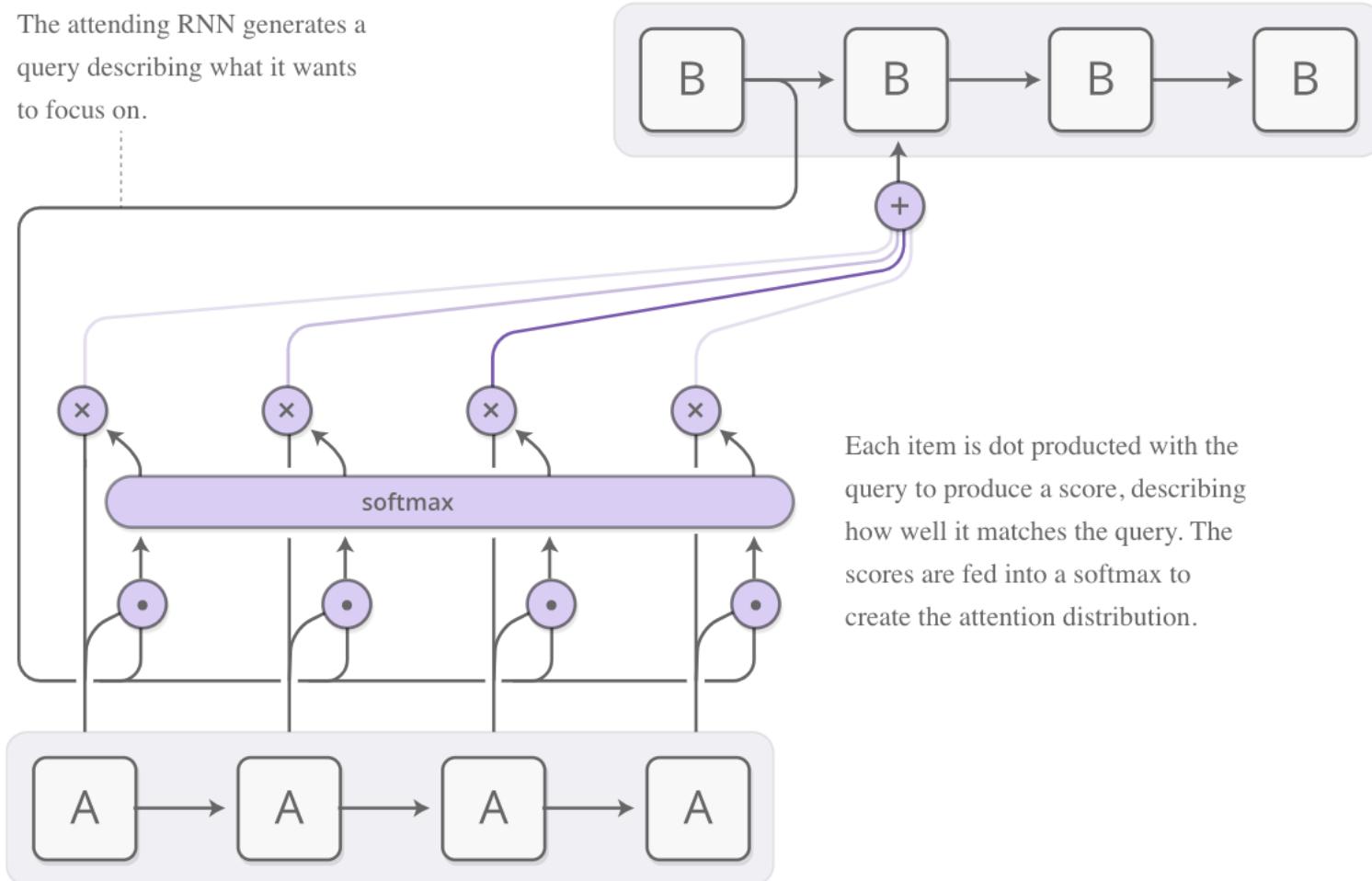
Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

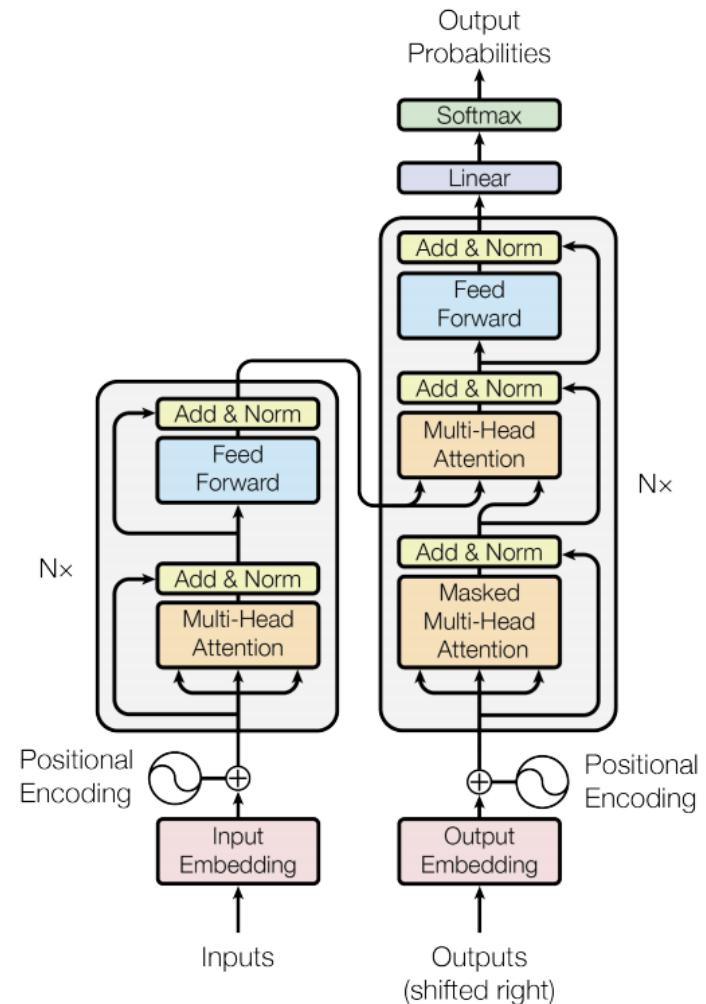
- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Self-Attention and Transformer

The attending RNN generates a query describing what it wants to focus on.

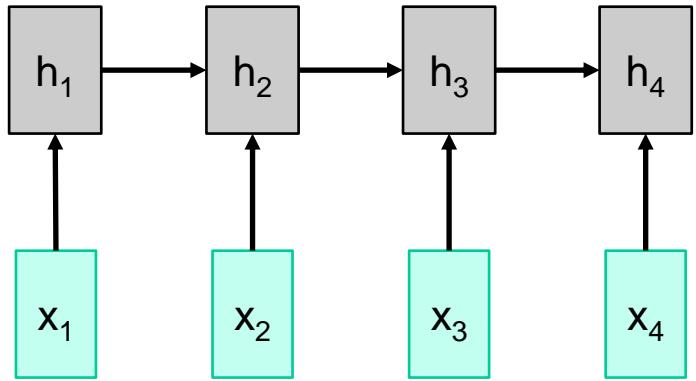


Each item is dot produced with the query to produce a score, describing how well it matches the query. The scores are fed into a softmax to create the attention distribution.

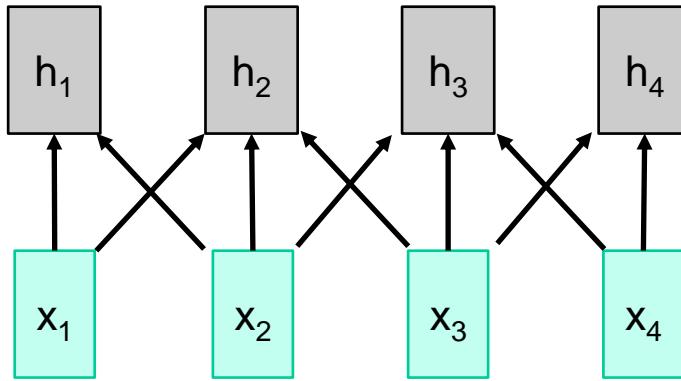


Different ways of processing sequences

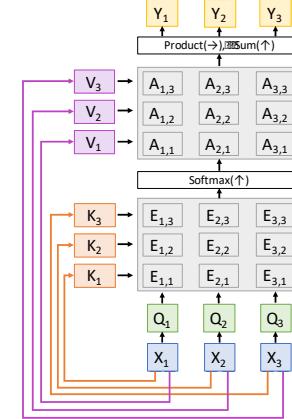
RNN



1D convolutional network



Self-Attention and Transformer



Works on **ordered sequences**

- Pros: Good at long sequences: the last hidden vector encapsulates the whole sequence
- Cons: Not parallelizable: need to compute hidden states sequentially

Works on **multidimensional grids**

- Con: Bad at long sequences: Need to stack many conv layers for outputs to “see” the whole sequence
- Pro: All outputs can be computed in parallel

- Works on **sets of vectors**

Outline

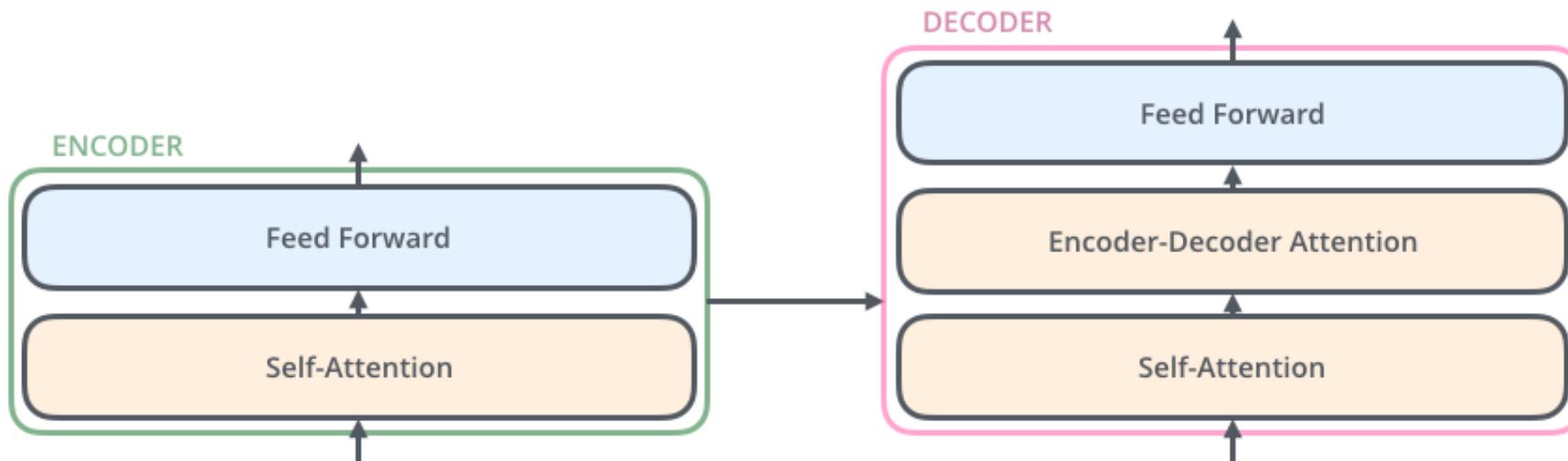
- Transformer architecture
 - Attention models
 - Implementation details
- Transformer-based language models
 - BERT
 - GPT and Other models
- Applications of transformers in vision

Attention is all you need

- Neural Machine Translation (**NMT**) architecture using only point-wise processing and attention (no recurrent units or convolutions)

Encoder: receives entire input sequence and outputs encoded sequence of the same length

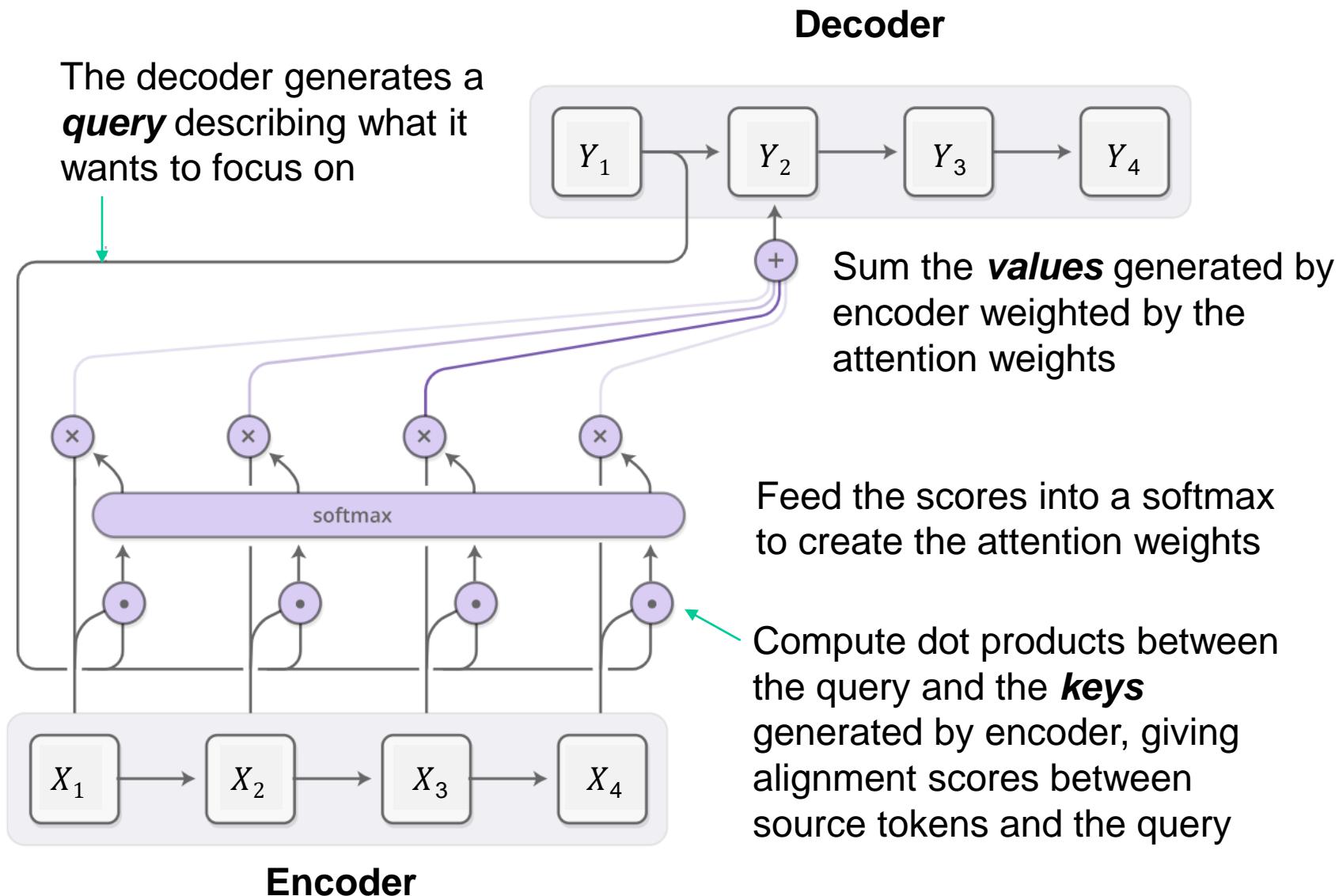
Decoder: predicts next token conditioned on encoder output and previously predicted tokens



Attention is all you need

- Neural Machine Translation (**NMT**) architecture using only point-wise processing and attention (no recurrent units or convolutions)
 - More efficient and parallelizable than recurrent or convolutional architectures, faster to train, better accuracy

Key-Value-Query attention model



Key-Value-Query attention model

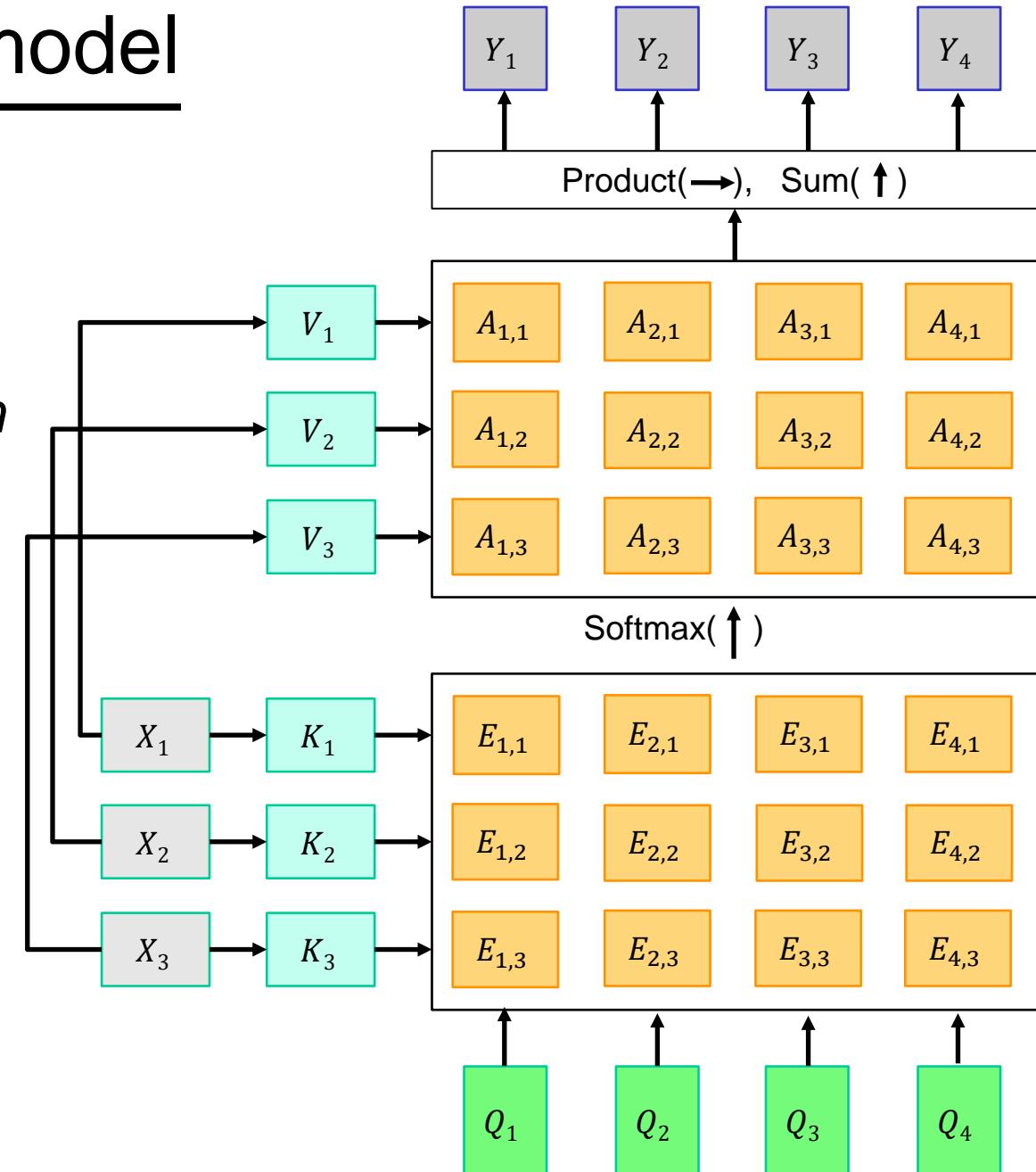
- Key vectors: $K = XW_K$
- Value Vectors: $V = XW_V$
- Query vectors
- Similarities: *scaled dot-product attention*

$$E_{i,j} = \frac{(Q_i \cdot K_j)}{\sqrt{D}}$$

(D is the dimensionality of the keys)

- Attn. weights: $A = \text{softmax}(E, \dim = 1)$
- Output vectors:

$$Y_i = \sum_j A_{i,j} V_j \quad \text{or} \quad Y = AV$$



Self-attention layer

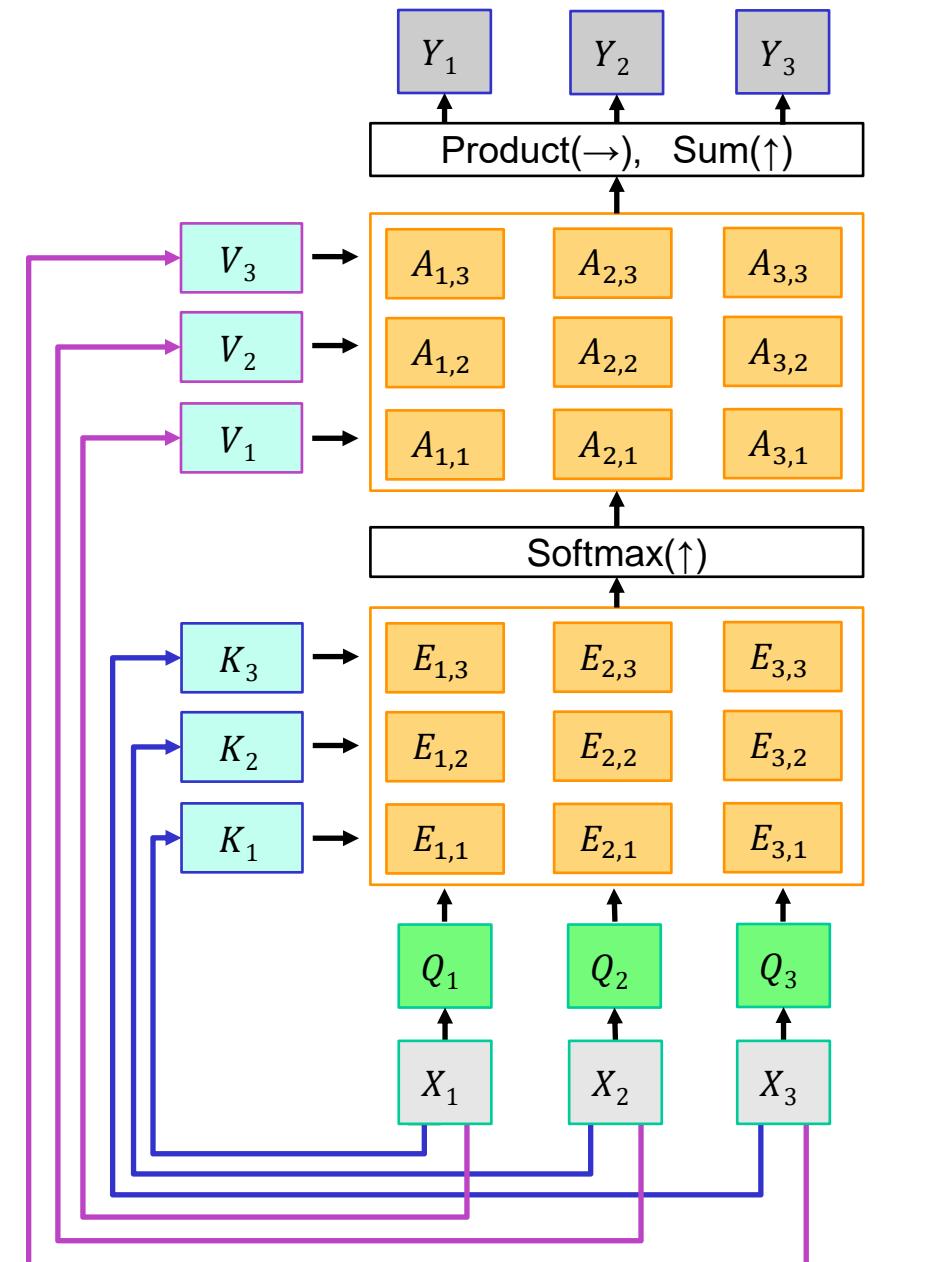
- Query vectors: $Q = XW_Q$
- Key vectors: $K = XW_K$
- Value vectors: $V = XW_V$
- Similarities: *scaled dot-product attention*

$$E_{i,j} = \frac{(Q_i \cdot K_j)}{\sqrt{D}}$$

(D is the dimensionality of the keys)

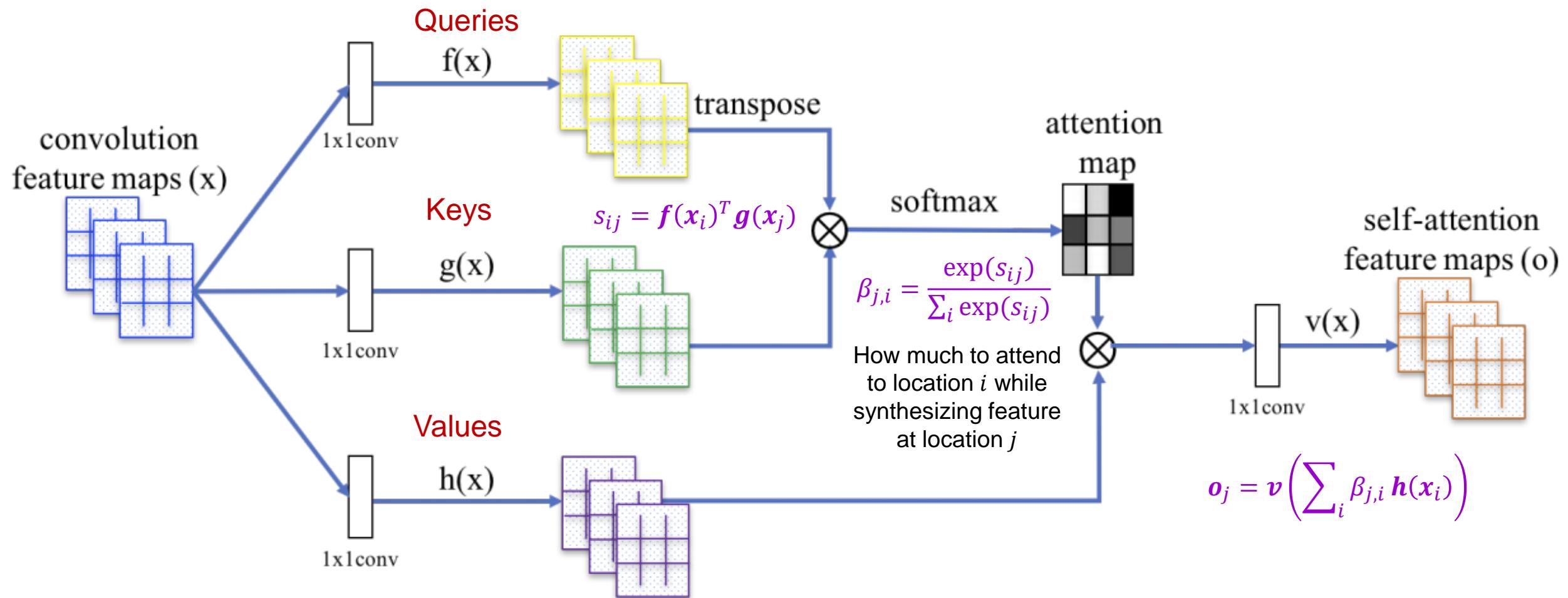
- Attn. weights: $A = \text{softmax}(E, \text{dim} = 1)$
- Output vectors:

$$Y_i = \sum_j A_{i,j} V_j \text{ or } Y = AV$$



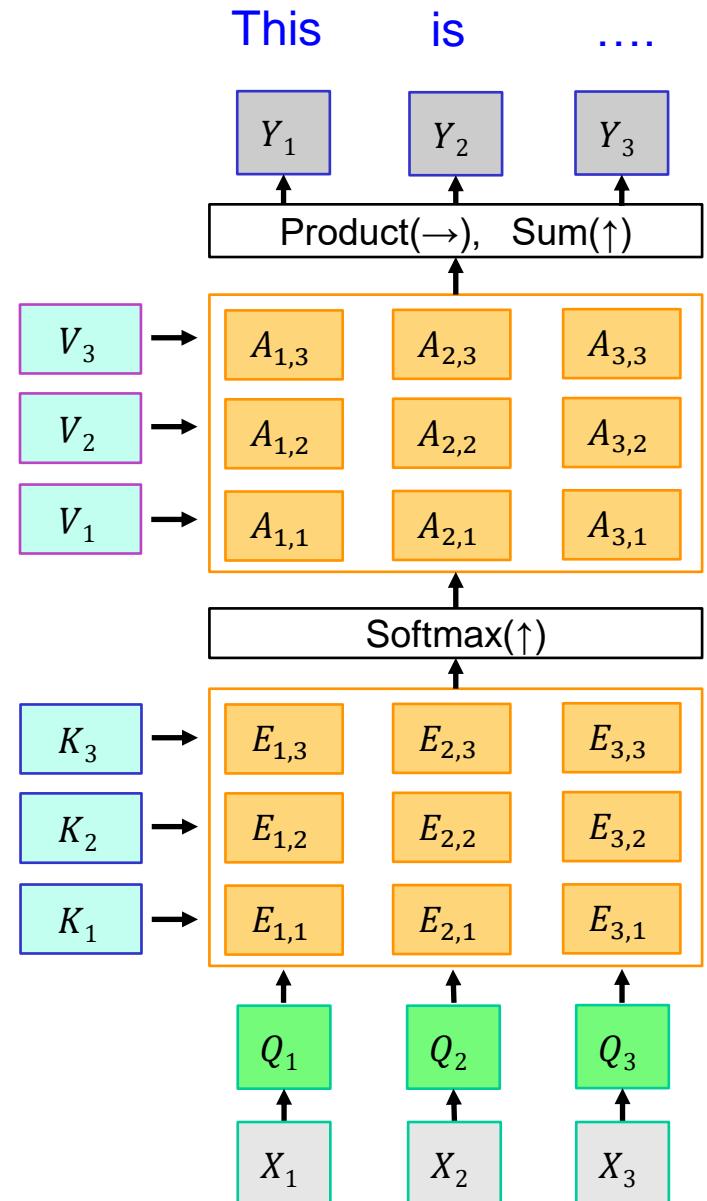
One query per input vector

Recall: Self-attention GAN



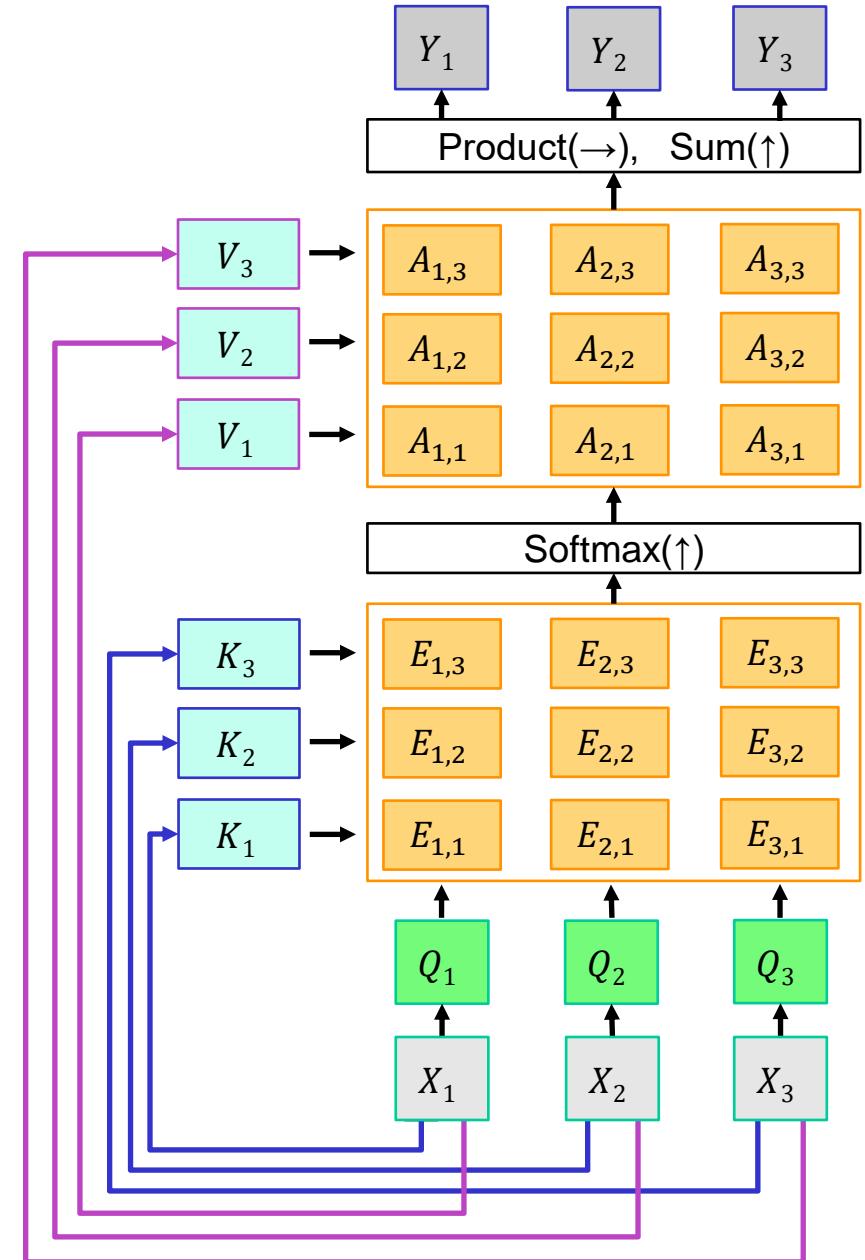
Masked self-attention layer

- The decoder should not “look ahead” in the output sequence



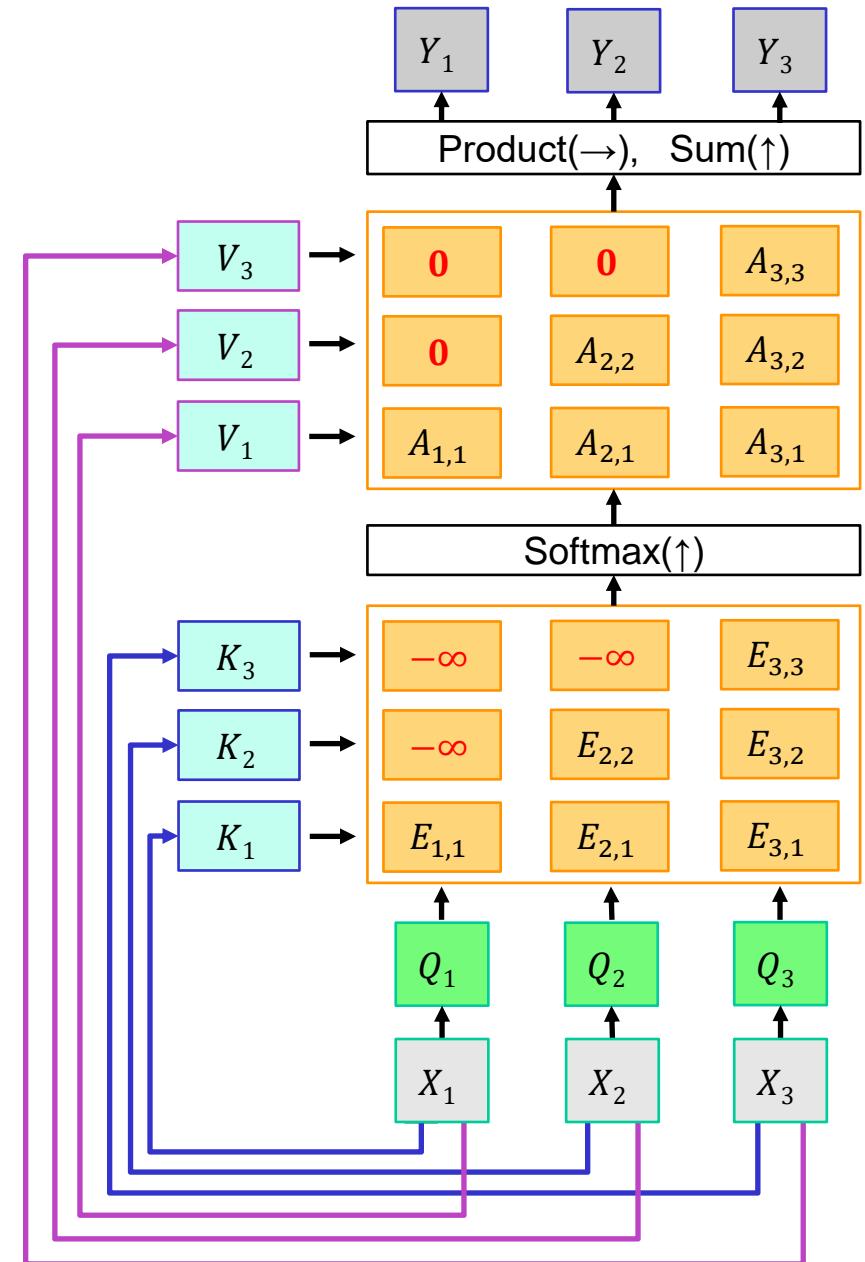
Masked self-attention layer

- The decoder should not “look ahead” in the output sequence

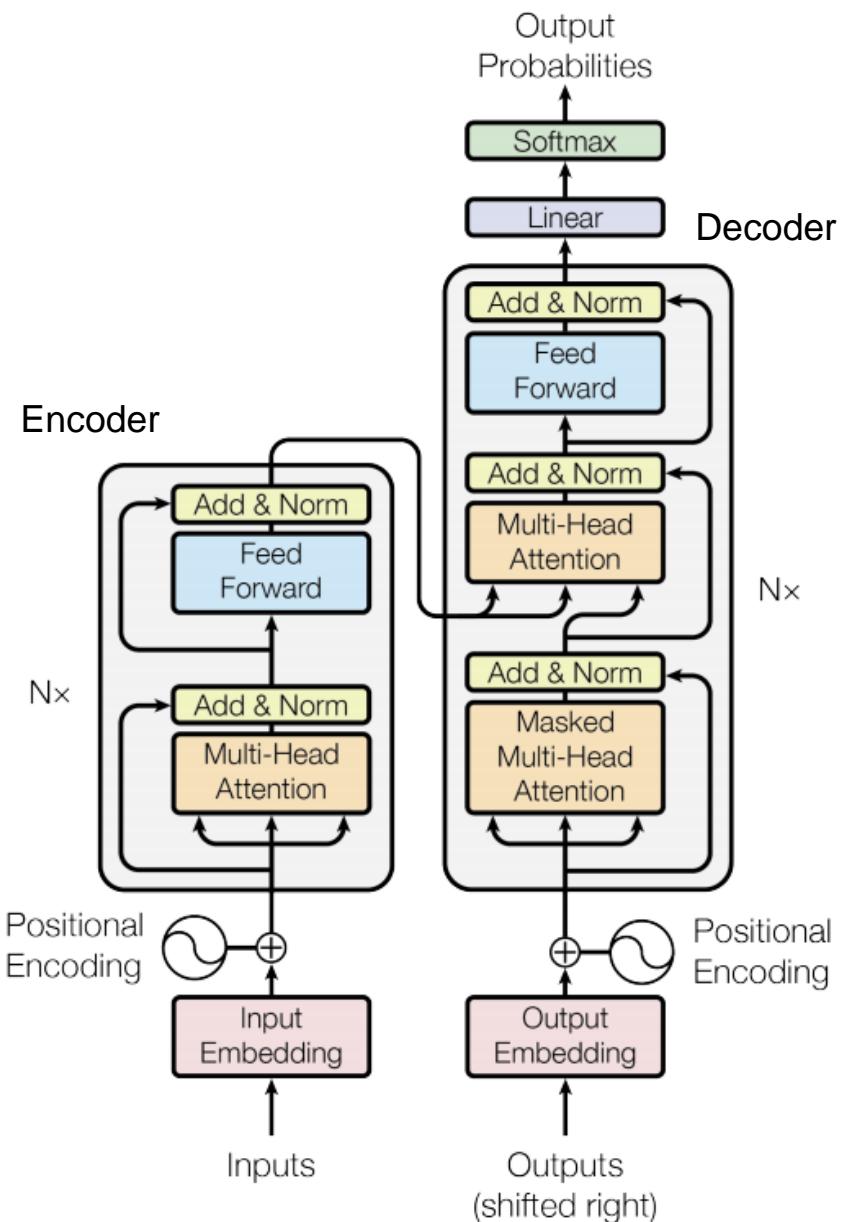


Masked self-attention layer

- The decoder should not “look ahead” in the output sequence

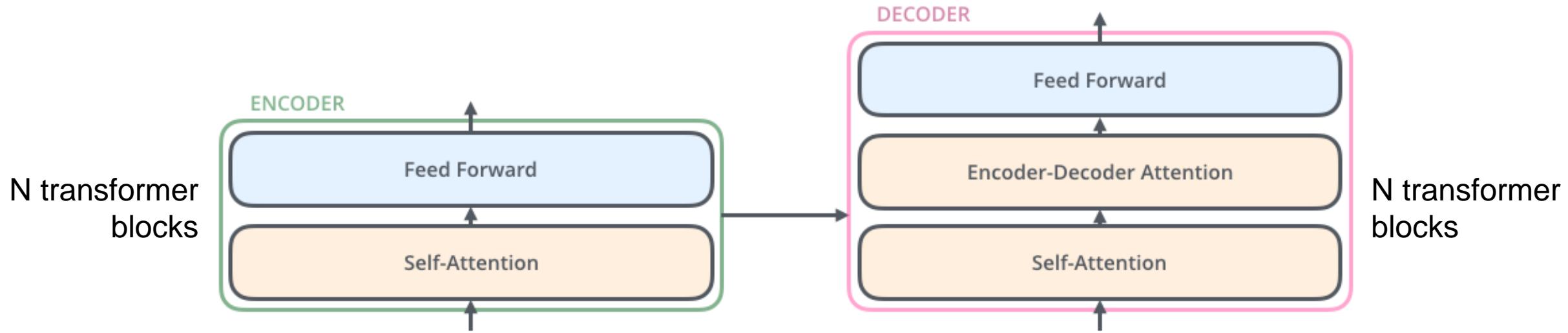


Transformer architecture: Details



A. Vaswani et al., [Attention is all you need](#), NeurIPS 2017

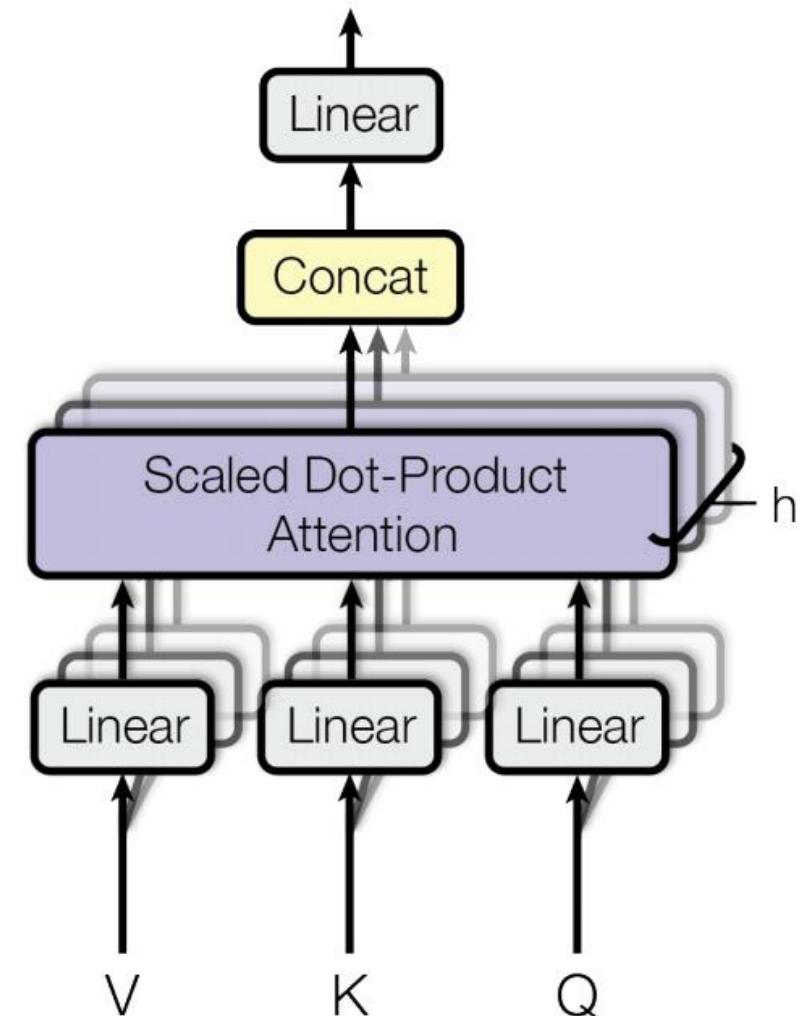
Attention mechanisms



- **Encoder self-attention:** queries, keys, and values come from previous layer of encoder
- **Decoder self-attention:** values corresponding to future decoder outputs are masked out
- **Encoder-decoder attention:** queries come from previous decoder layer, keys and values come from output of encoder

Multi-head attention

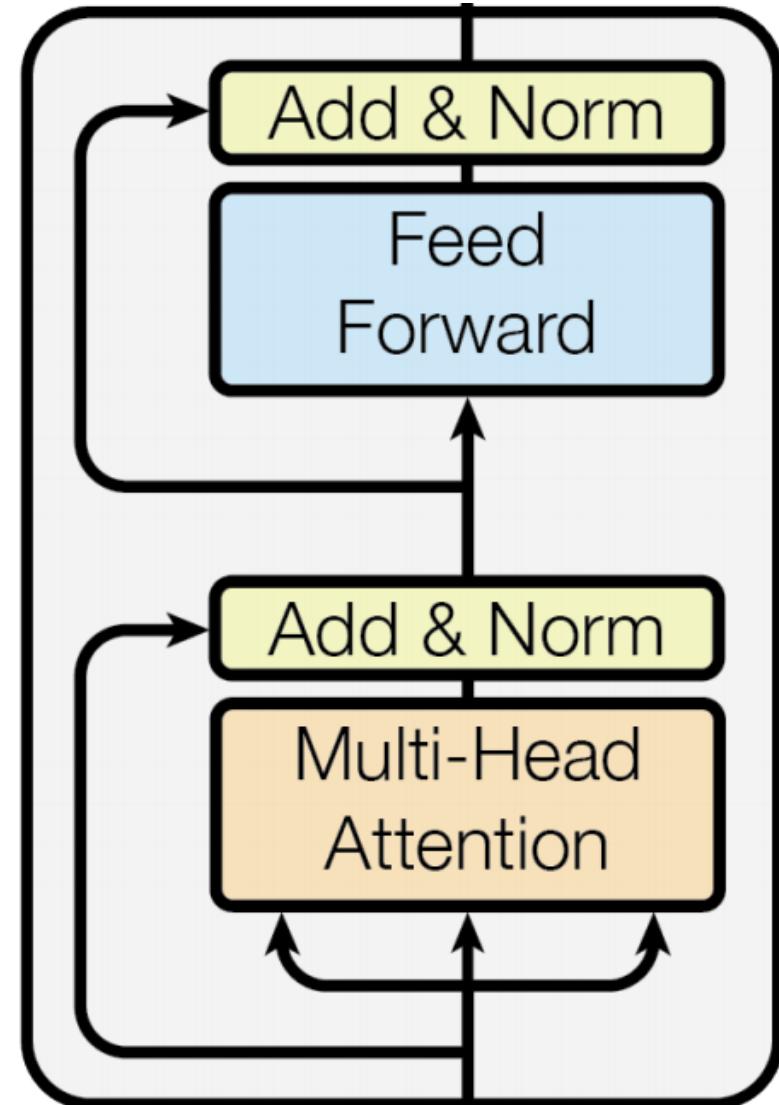
- Run h attention models in parallel on top of different linearly projected versions of Q, K, V ; concatenate and linearly project the results
- Intuition: enables model to attend to different kinds of information at different positions



Transformer blocks

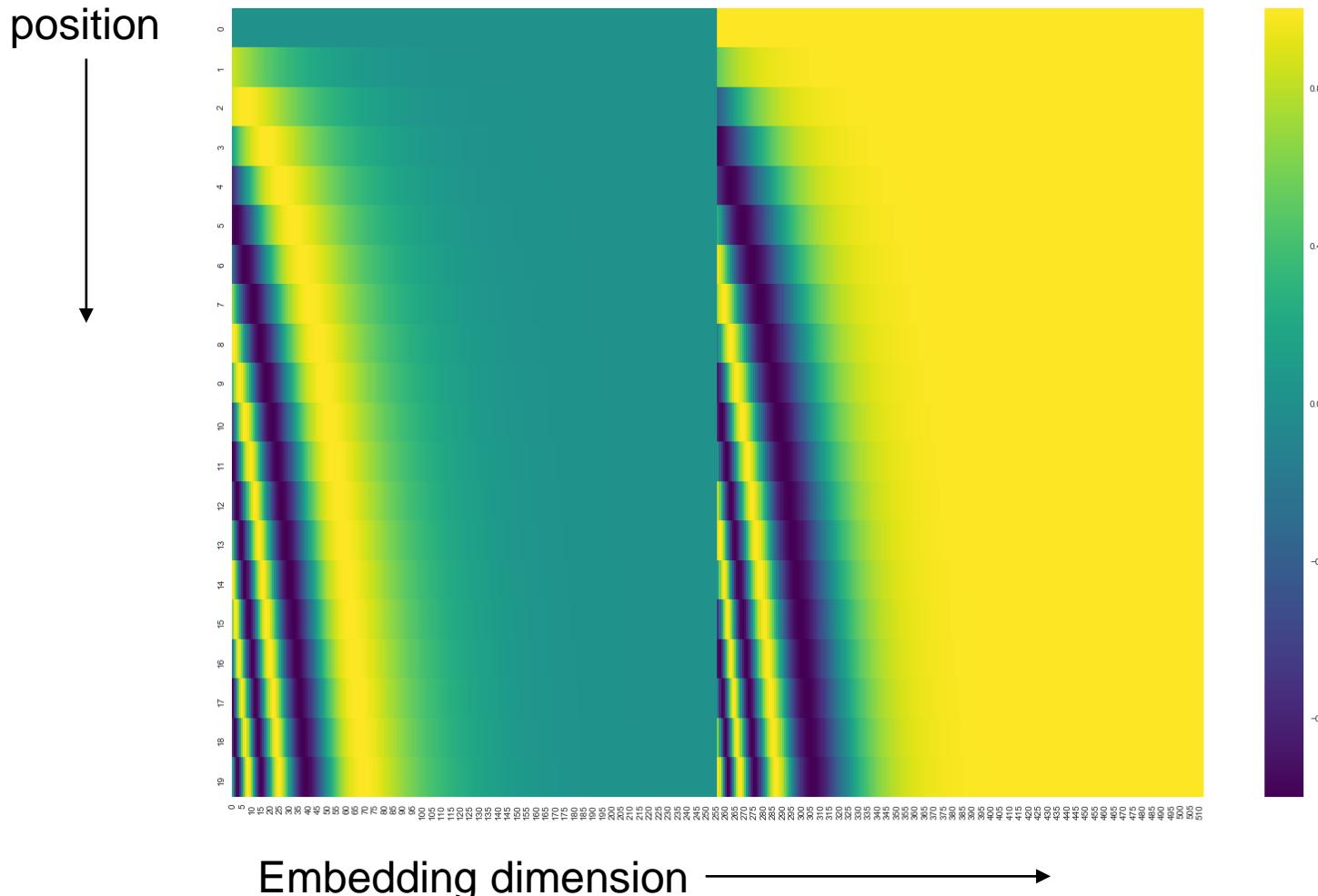
- A **Transformer** is a sequence of transformer blocks
 - Vaswani et al.: N=12 blocks, embedding dimension = 512, 6 attention heads
 - **Add & Norm:** residual connection followed by [layer normalization](#)
 - **Feedforward:** two linear layers with ReLUs in between, applied independently to each vector
 - Attention is the only interaction between inputs!

N×



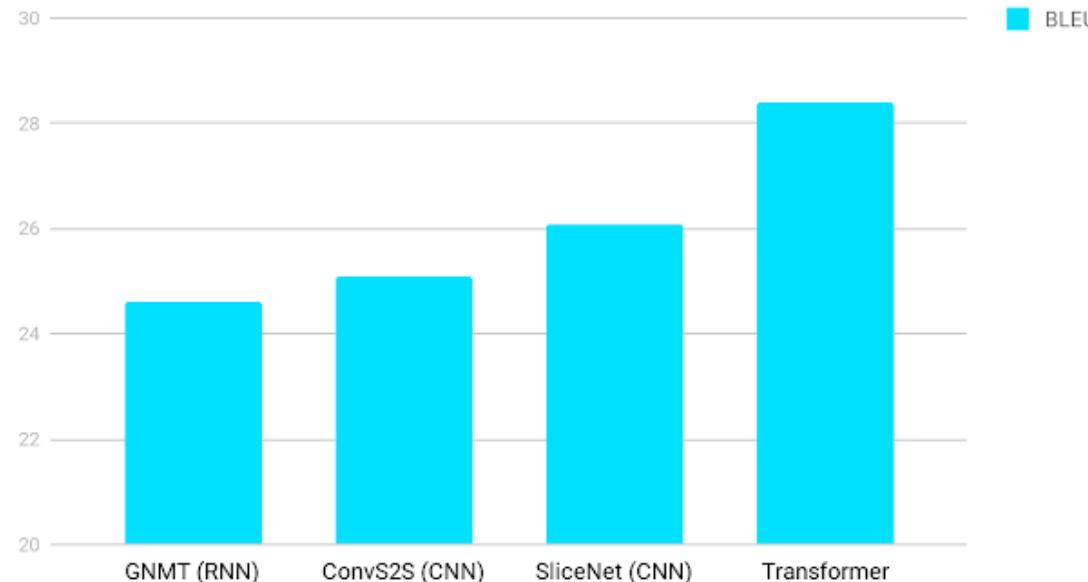
Positional encoding

- To give transformer information about ordering of tokens, add function of position (based on sines and cosines) to every input

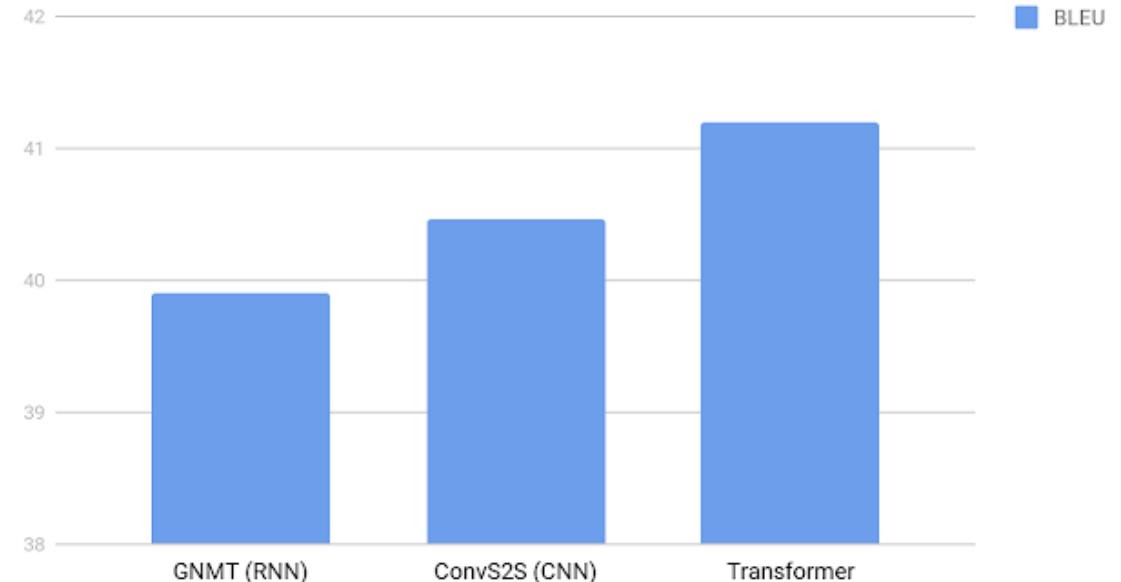


Results

English German Translation quality



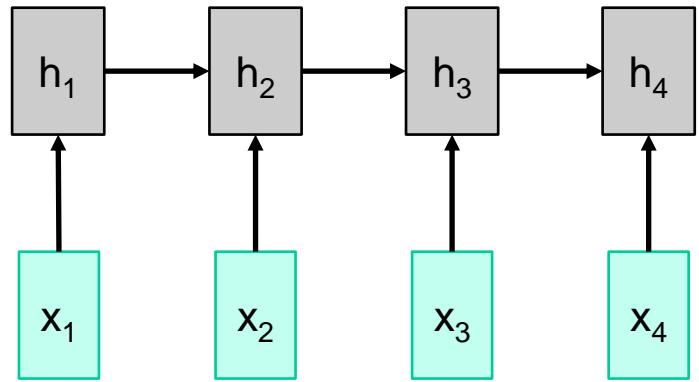
English French Translation Quality



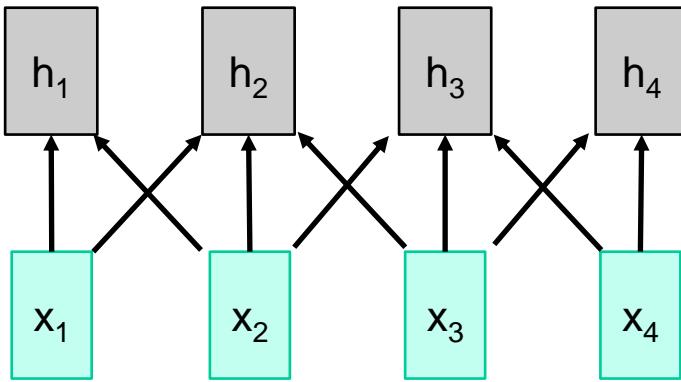
<https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>

Different ways of processing sequences

RNN



1D convolutional network



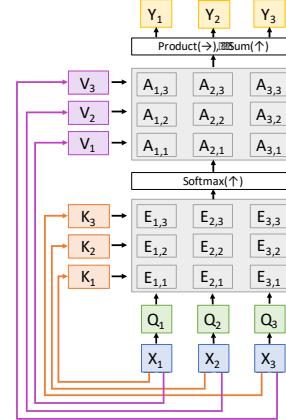
Works on **ordered sequences**

- Pros: Good at long sequences: the last hidden vector encapsulates the whole sequence
- Cons: Not parallelizable: need to compute hidden states sequentially

Works on **multidimensional grids**

- Con: Bad at long sequences: Need to stack many conv layers for outputs to “see” the whole sequence
- Pro: Highly parallel: Each output can be computed in parallel

Self-Attention and Transformer



- Works on **sets of vectors**
- Pro: Good at long sequences: after one self-attention layer, each output “sees” all inputs!
- Pro: Highly parallel: Each output can be computed in parallel
- Con: Very memory-intensive

Outline

- Transformer architecture
 - Attention models
 - Implementation details
- Transformer-based language models
 - BERT
 - GPT and Other models

Self-supervised language modeling with transformers

1. Download a lot of text from the internet
2. Train a transformer using a suitable pretext task
3. Fine-tune the transformer on desired NLP task

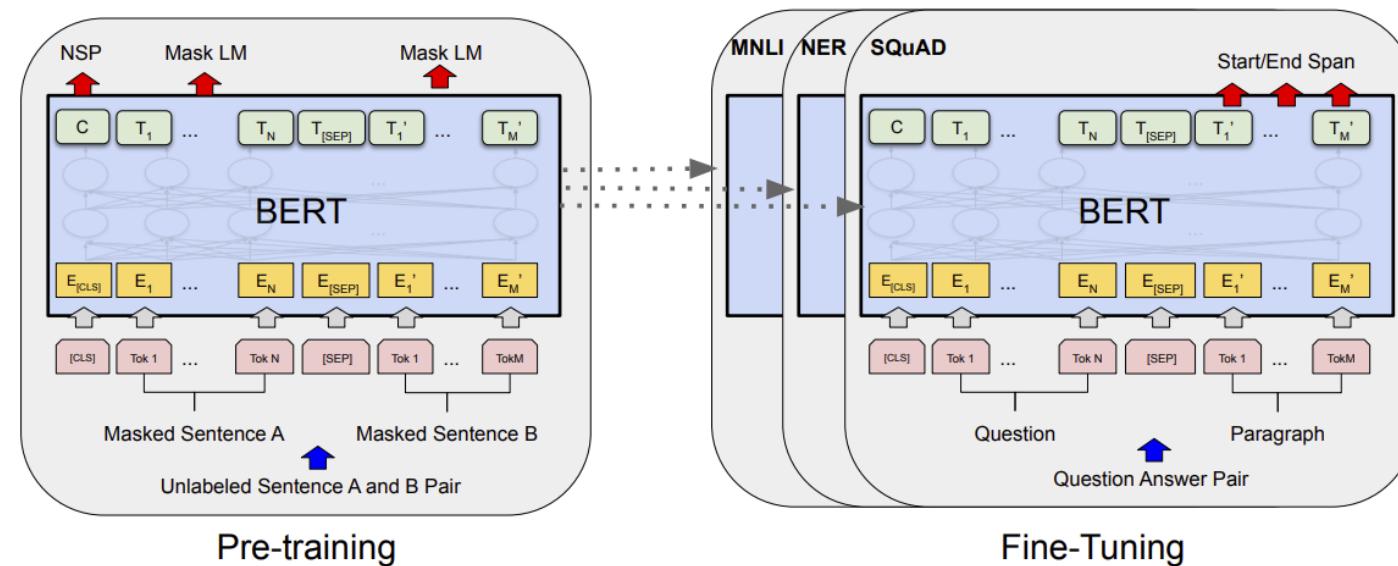
Model Alias	Org.	Article Reference
ULMfit	fast.ai	<i>Universal Language Model Fine-tuning for Text Classification</i> Howard and Ruder
 ELMo	AllenNLP	<i>Deep contextualized word representations</i> Peters et al.
OpenAI GPT	OpenAI	<i>Improving Language Understanding by Generative Pre-Training</i> Radford et al.
 BERT	Google	<i>BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding</i> Devlin et al.
XLM	Facebook	<i>Cross-lingual Language Model Pretraining</i> Lample and Conneau

[Image source](#)

Self-supervised language modeling with transformers

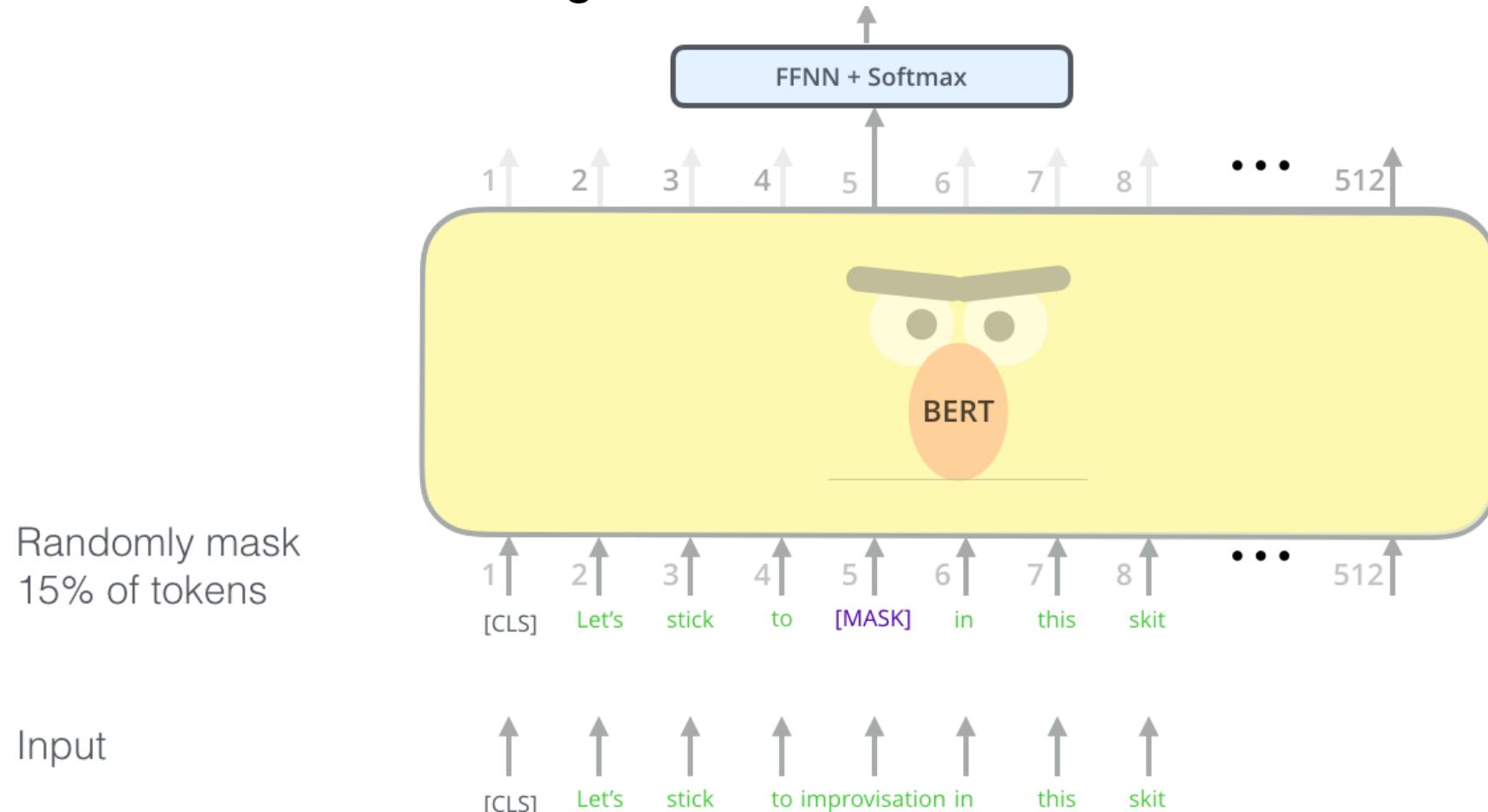
1. Download a lot of text from the internet
2. Train a transformer using a suitable pretext task
3. Fine-tune the transformer on desired NLP task

Bidirectional Encoder Representations from Transformers (BERT)



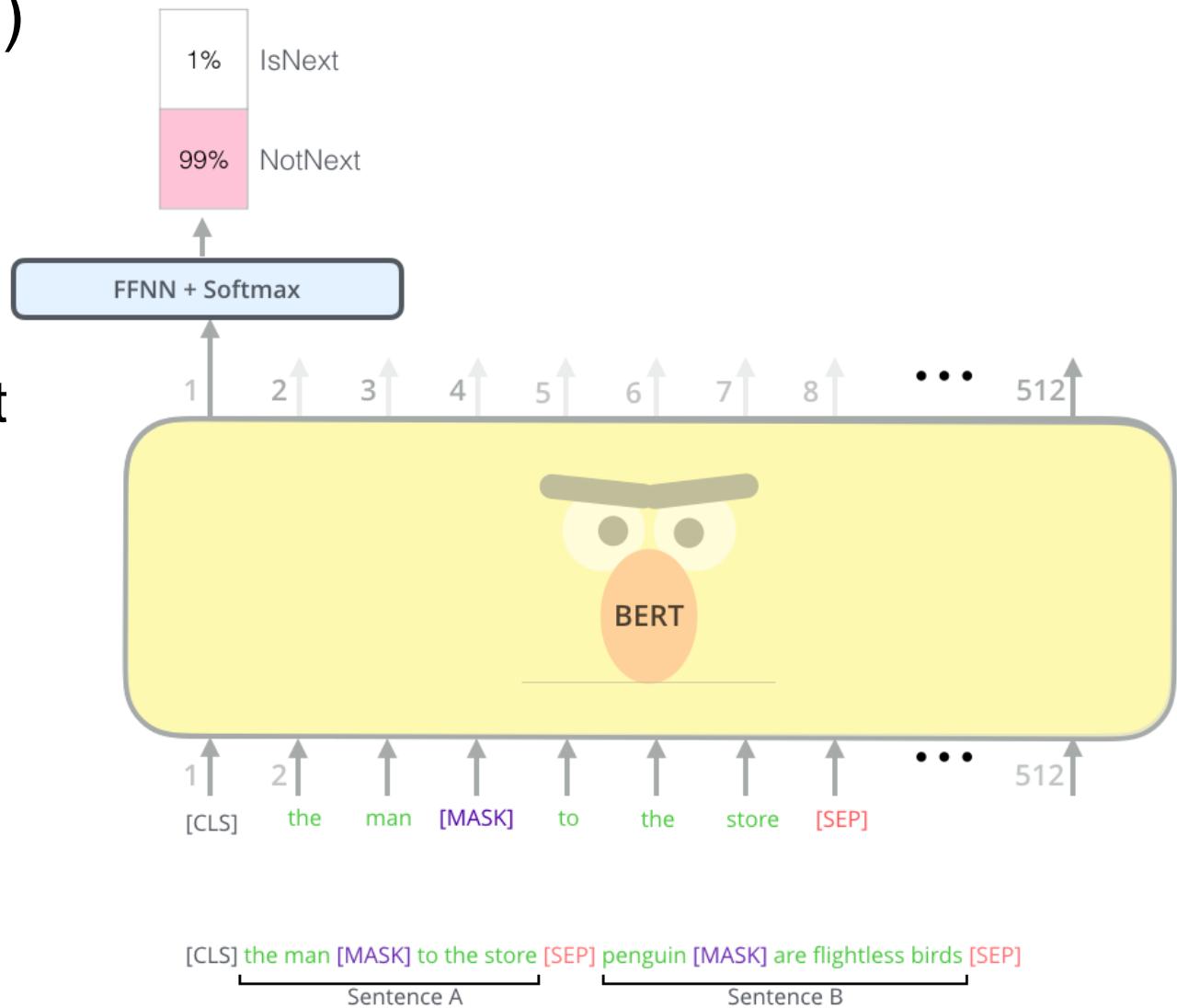
BERT: Pretext tasks

- Masked language model (MLM)
 - Randomly mask 15% of tokens in input sentences, goal is to reconstruct them using bidirectional context



BERT: Pretext tasks

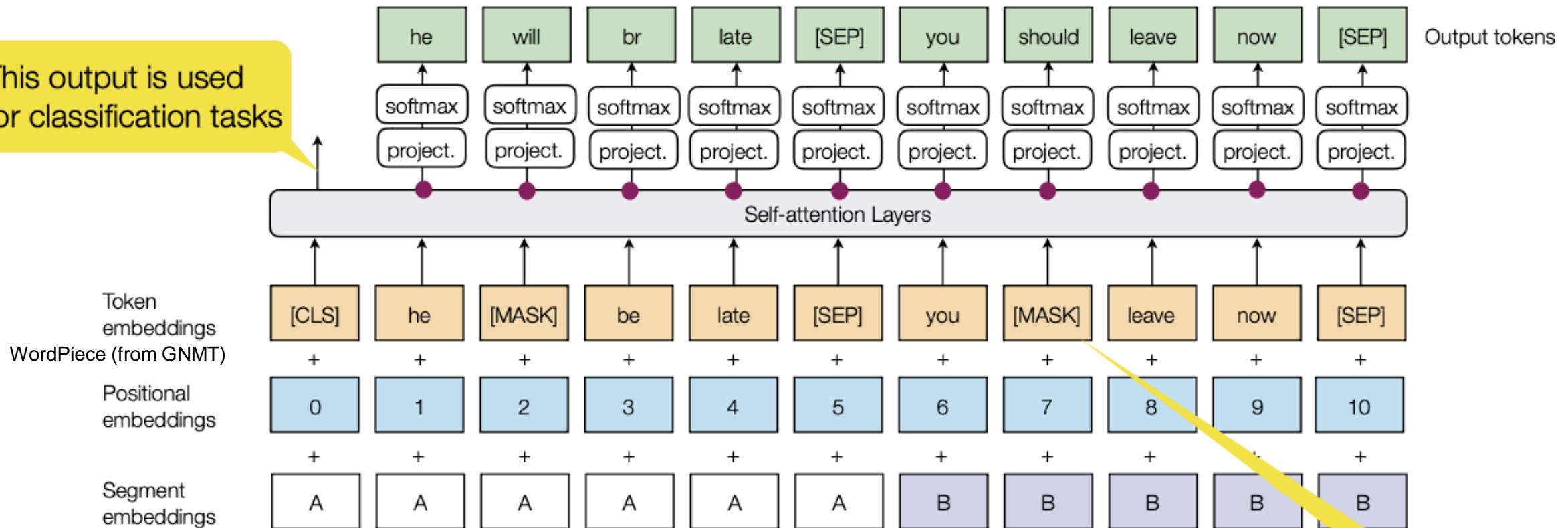
- Next sentence prediction (NSP)
 - Useful for Question Answering and Natural Language Inference tasks
 - In the training data, 50% of the time B is the actual sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence (labeled as NotNext).



[Image source](#)

BERT: More detailed view

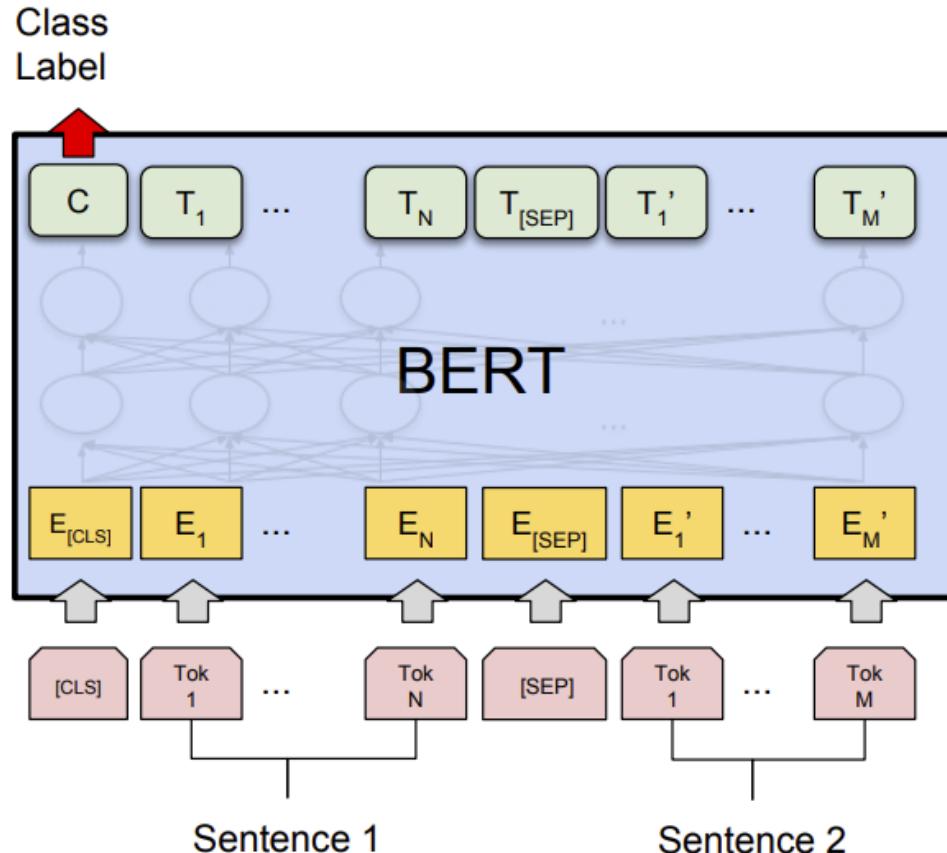
This output is used
for classification tasks



Trained on Wikipedia (2.5B words) + BookCorpus (800M words)

15% of tokens
get masked

BERT: Downstream tasks



(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

Textual entailment

Premises

A band performing on the corner of the street.

Musicians playing on the sidewalk.

Shoppers stop to listen to music.

Hypotheses

ENTAILS

The band is performing outdoors.

NEUTRAL

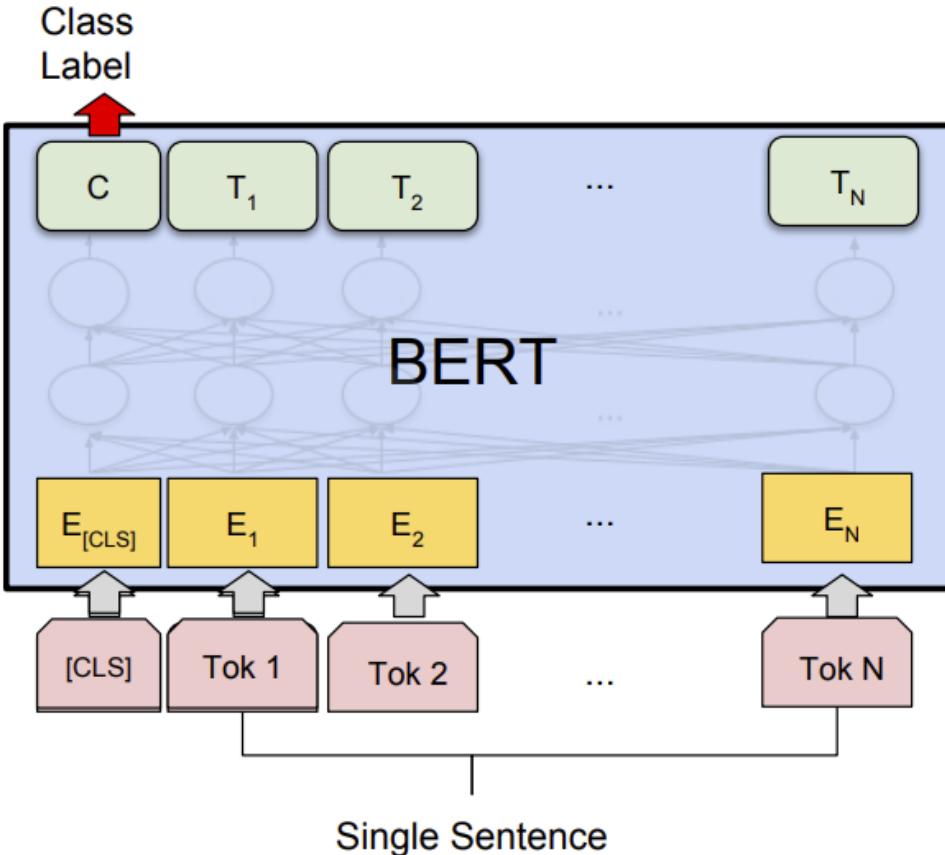
A bluegrass band plays for the passerby.

CONTRADICTS

The orchestra plays in the concert hall.

Source: J. Hockenmaier

BERT: Downstream tasks



(b) Single Sentence Classification Tasks:
SST-2, CoLA

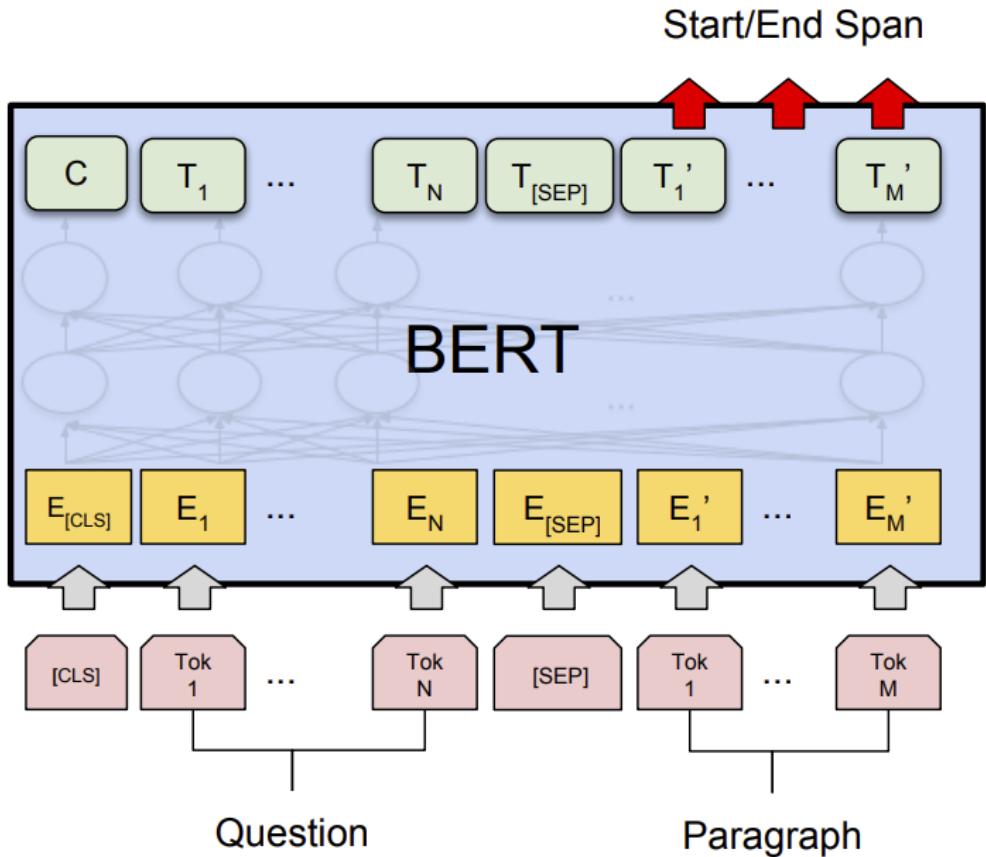
Sentiment classification, linguistic acceptability

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

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

[Image source](#)

BERT: Downstream tasks



(c) Question Answering Tasks:
SQuAD v1.1

Find span in paragraph that contains the answer

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called “showers”.

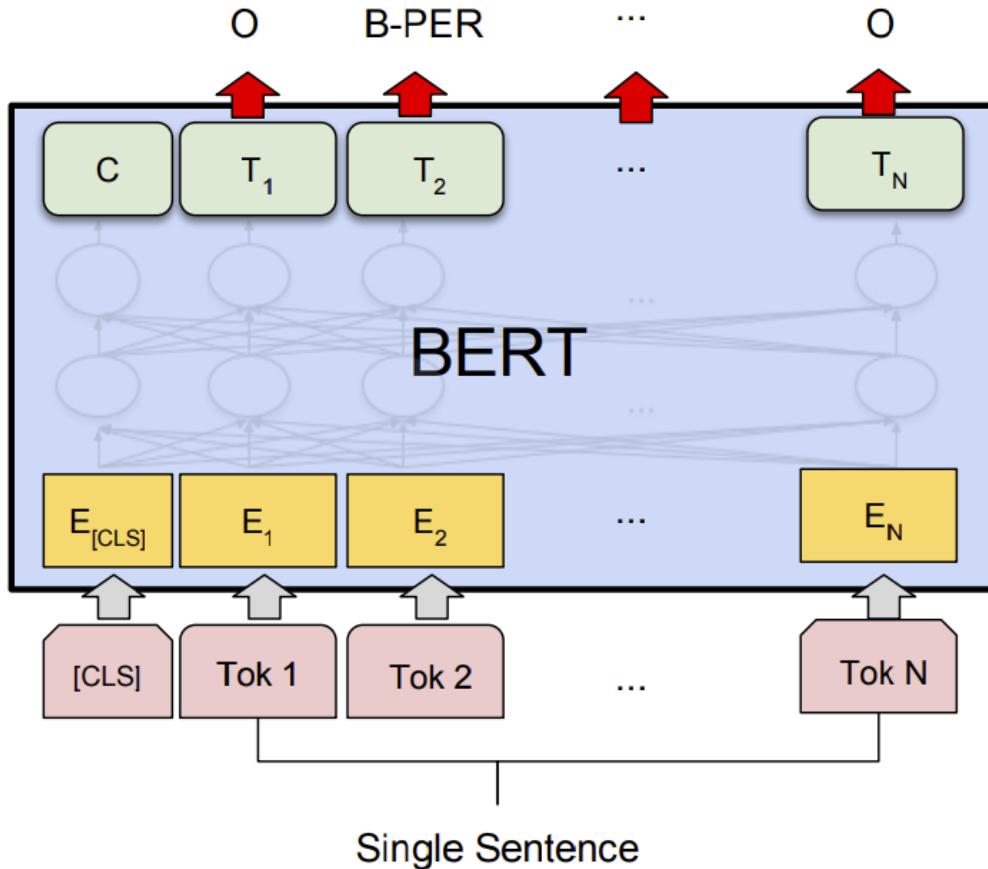
What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Source: [SQuAD v1.1 paper](#)

BERT: Downstream tasks



When Sebastian Thrun PERSON started at Google ORG in 2007 DATE, few people outside of the company took him seriously. "I can tell you very senior CEOs of major American NORP car companies would shake my hand and turn away because I wasn't worth talking to," said Thrun PERSON, now the co-founder and CEO of online higher education startup Udacity, in an interview with Recode ORG earlier this week DATE.

A little less than a decade later DATE, dozens of self-driving startups have cropped up while automakers around the world clamor, wallet in hand, to secure their place in the fast-moving world of fully automated transportation.

[Image source](#)

(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Named entity recognition

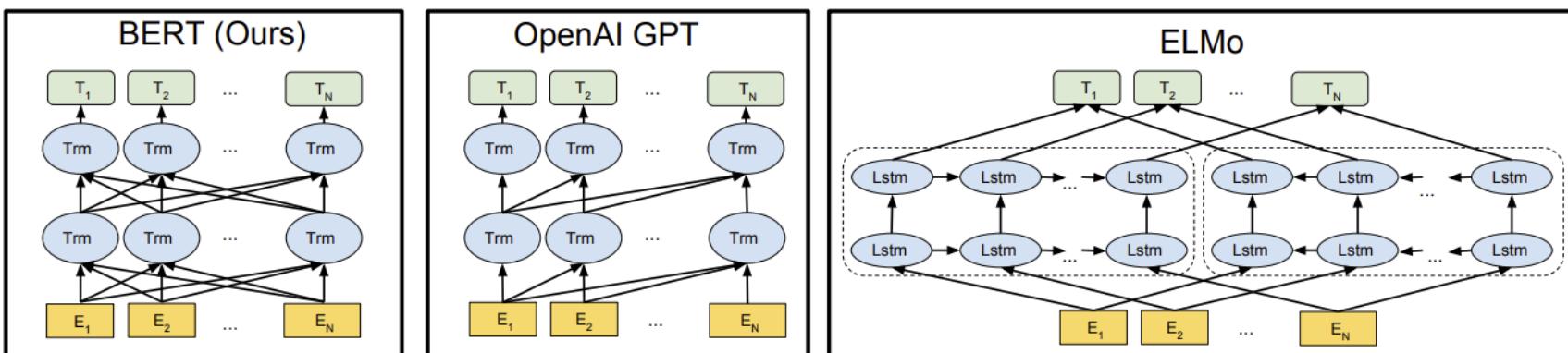
Outline

- Transformer architecture
 - Attention models
 - Implementation details
- Transformer-based language models
 - BERT
 - GPT and Other models

Other language models

Model Alias	Org.	Article Reference
ULMfit	fast.ai	<i>Universal Language Model Fine-tuning for Text Classification</i> Howard and Ruder
 ELMo	AllenNLP	<i>Deep contextualized word representations</i> Peters et al.
OpenAI GPT	OpenAI	<i>Improving Language Understanding by Generative Pre-Training</i> Radford et al.
 BERT	Google	<i>BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding</i> Devlin et al.
XLM	Facebook	<i>Cross-lingual Language Model Pretraining</i> Lample and Conneau

[Image source](#)



Scaling up transformers

Model	Layers	Hidden dim.	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)

Scaling up transformers

Model	Layers	Hidden dim.	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	4x TPU (4 days)
BERT-Large	24	1024	16	340M	13 GB	16x TPU (4 days)

Scaling up transformers

Model	Layers	Hidden dim.	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	4x TPU (4 days)
BERT-Large	24	1024	16	340M	13 GB	16x TPU (4 days)
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)

Yang et al., XLNet: Generalized Autoregressive Pretraining for Language Understanding, 2019
Liu et al., RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019

Scaling up transformers

Model	Layers	Hidden dim.	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	4x TPU (4 days)
BERT-Large	24	1024	16	340M	13 GB	16x TPU (4 days)
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40 GB	

Scaling up transformers

Model	Layers	Hidden dim.	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	4x TPU (4 days)
BERT-Large	24	1024	16	340M	13 GB	16x TPU (4 days)
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)

~\$430,000 on Amazon AWS!

Scaling up transformers

Model	Layers	Hidden dim.	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	4x TPU (4 days)
BERT-Large	24	1024	16	340M	13 GB	16x TPU (4 days)
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100 GPU

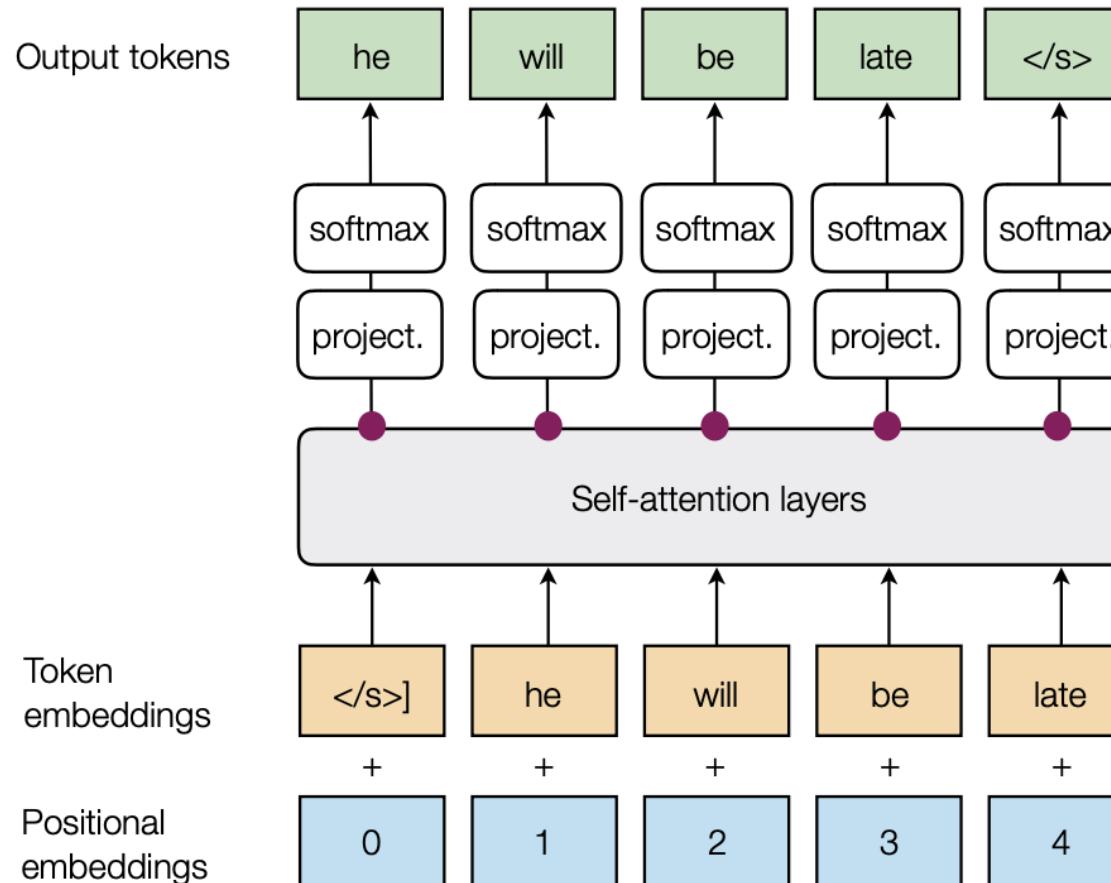
Scaling up transformers

Model	Layers	Hidden dim.	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	4x TPU (4 days)
BERT-Large	24	1024	16	340M	13 GB	16x TPU (4 days)
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100 GPU
GPT-3	96	12288	96	175B	694GB	?

~\$4.6M, 355 GPU-years
([source](#))

OpenAI GPT (Generative Pre-Training)

- Pre-training task: next token prediction



GPT-2 and GPT-3

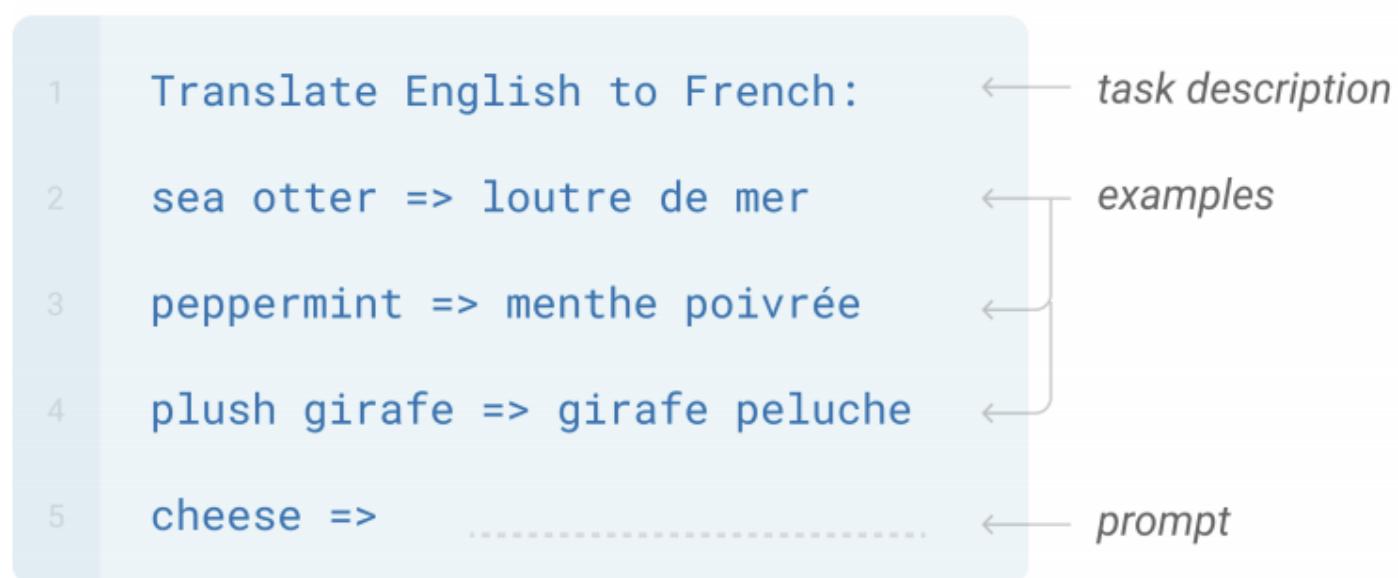
- Key idea: if the model and training datasets are big enough, model can adapt to new tasks without fine-tuning

GPT-2: A. Radford et al., [Language models are unsupervised multitask learners](#), 2019

GPT-3: T. Brown et al., [Language models are few-shot learners](#), arXiv 2020

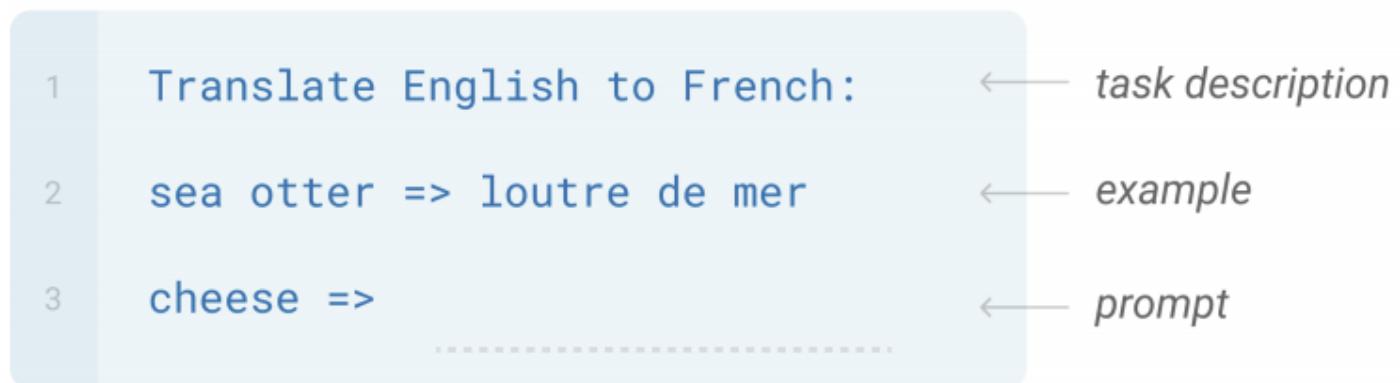
GPT-3

- Key idea: if the model and training datasets are big enough, model can adapt to new tasks *without fine-tuning*
- **Few-shot learning:** In addition to the task description, the model sees a few examples of the task



GPT-3

- Key idea: if the model and training datasets are big enough, model can adapt to new tasks *without fine-tuning*
- **One-shot learning:** In addition to the task description, the model sees a *single example* of the task



GPT-3

- Key idea: if the model and training datasets are big enough, model can adapt to new tasks *without fine-tuning*
- **Zero-shot learning:** The model sees the task description and *no* training examples



Task: Generate news article

Title: United Methodists Agree to Historic Split

Subtitle: Those who oppose gay marriage will form their own denomination

Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.

The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Gray: human prompts,
boldface: GPT-3
completions

(Three articles
provided as training
examples)

Figure 3.14: The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%).

Task: Use new word in sentence

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:

We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:

One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduckles.

A "yalubalu" is a type of vegetable that looks like a big pumpkin. An example of a sentence that uses the word yalubalu is:

I was on a trip to Africa and I tried this yalubalu vegetable that was grown in a garden there. It was delicious.

A "Burringo" is a car with very fast acceleration. An example of a sentence that uses the word Burringo is:

In our garage we have a Burringo that my father drives to work every day.

A "Gigamuru" is a type of Japanese musical instrument. An example of a sentence that uses the word Gigamuru is:

I have a Gigamuru that my uncle gave me as a gift. I love to play it at home.

To "screeg" something is to swing a sword at it. An example of a sentence that uses the word screeg is:

We screeghed at each other for several minutes and then we went outside and ate ice cream.

Gray: human prompts,
boldface: GPT-3
completions

Task: Correct grammar

Poor English input: I eated the purple berries.

Good English output: I ate the purple berries.

Poor English input: Thank you for picking me as your designer. I'd appreciate it.

Good English output: Thank you for choosing me as your designer. I appreciate it.

Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.

Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.

Poor English input: I'd be more than happy to work with you in another project.

Good English output: I'd be more than happy to work with you on another project.

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.

Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.

Poor English input: The patient was died.

Good English output: The patient died.

Poor English input: We think that Leslie likes ourselves.

Good English output: We think that Leslie likes us.

Poor English input: Janet broke Bill on the finger.

Good English output: Janet broke Bill's finger.

Poor English input: Mary arranged for, in St. Louis, John to rent a house cheap.

Good English output: Mary arranged for John to rent a house in St. Louis.

Gray: human prompts,
boldface: GPT-3
completions

Task: Generate poems

----- Generated Poem 1 -----

I must have shadows on the way
If I am to walk I must have
Each step taken slowly and alone
To have it ready made

And I must think in lines of grey
To have dim thoughts to be my guide
Must look on blue and green
And never let my eye forget
That color is my friend
And purple must surround me too

The yellow of the sun is no more
Intrusive than the bluish snow
That falls on all of us. I must have
Grey thoughts and blue thoughts walk with me
If I am to go away at all.

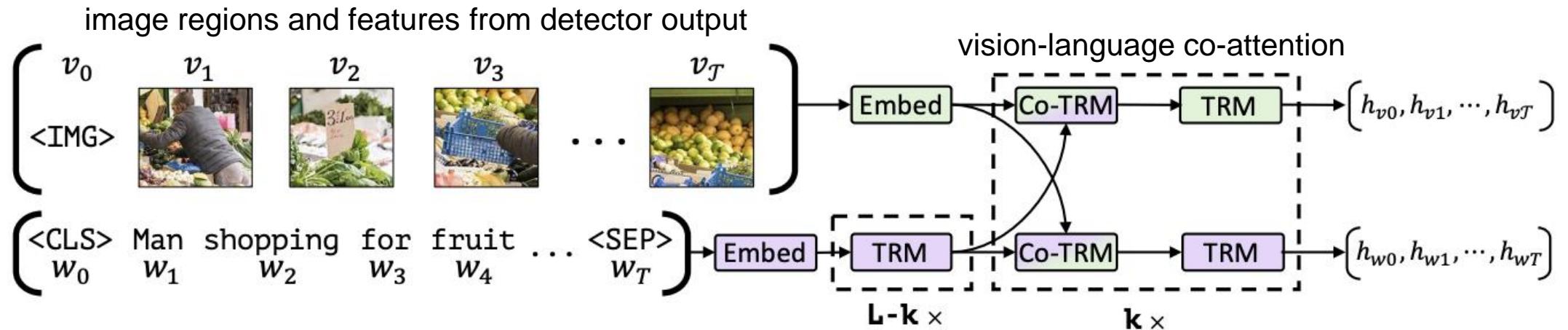
----- Generated Poem 4 -----

Nobody will come to this place. It is a road that leads nowhere.
The solitude is deep. The mountains are high.
But they are desolate, and they turn the traveler's face
Towards the North. All the sounds of the world are far away.
When the wind rises above the trees,
The boughs bow to the ground.
Even the birds that inhabit the tangle of weeds
That is the roadside cover, are silent. One listens,
But hears no roar of the forest. One is alone.
One will be taken.
One will be taken.
There is no utterance, there is no conversation,
But one is uneasy all the same....
There is a thin blue mist,
A darkness rising like smoke,
And within that darkness
A possession of the heart.
One will be taken.... It was here, and it will be here again-
Here, under this sky empty and full of light.

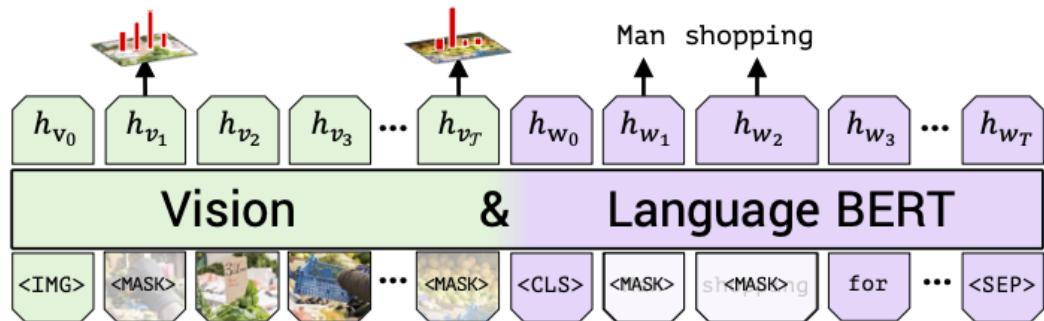
Transformers: Outline

- Transformer architecture
 - Attention models
 - Implementation details
- Transformer-based language models
 - BERT
 - GPT and Other models
- Applications of transformers in vision

Vision-and-language BERT

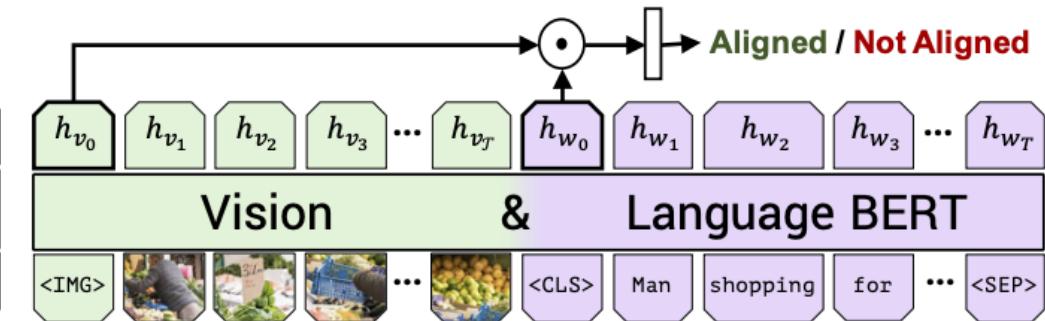


predict object class of masked out region



(a) Masked multi-modal learning

predict whether image and sentence go together



(b) Multi-modal alignment prediction

Detection Transformer (DETR)

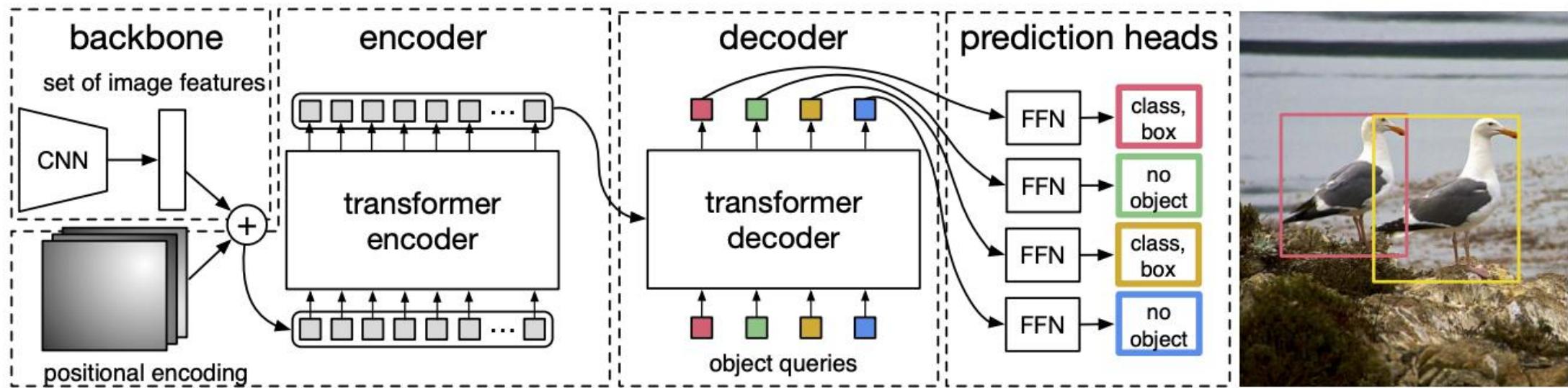
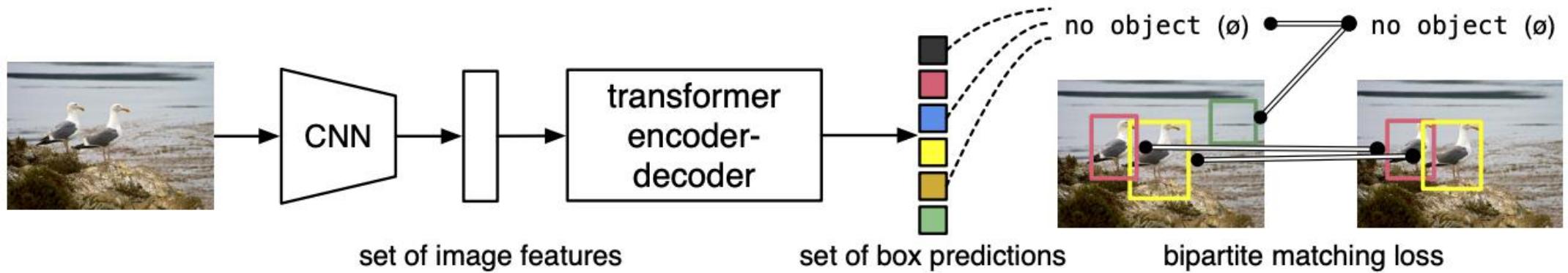


Image transformer

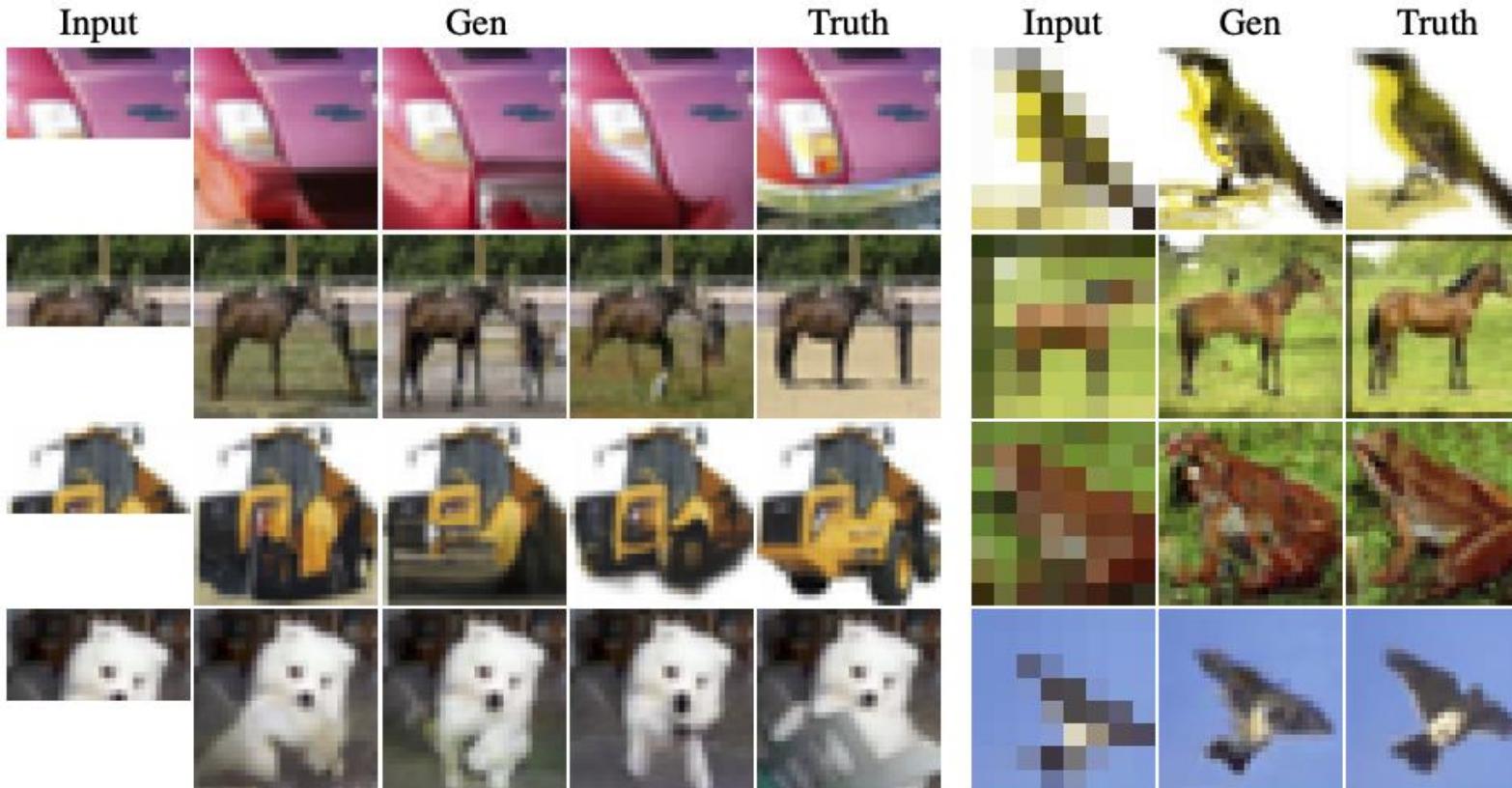
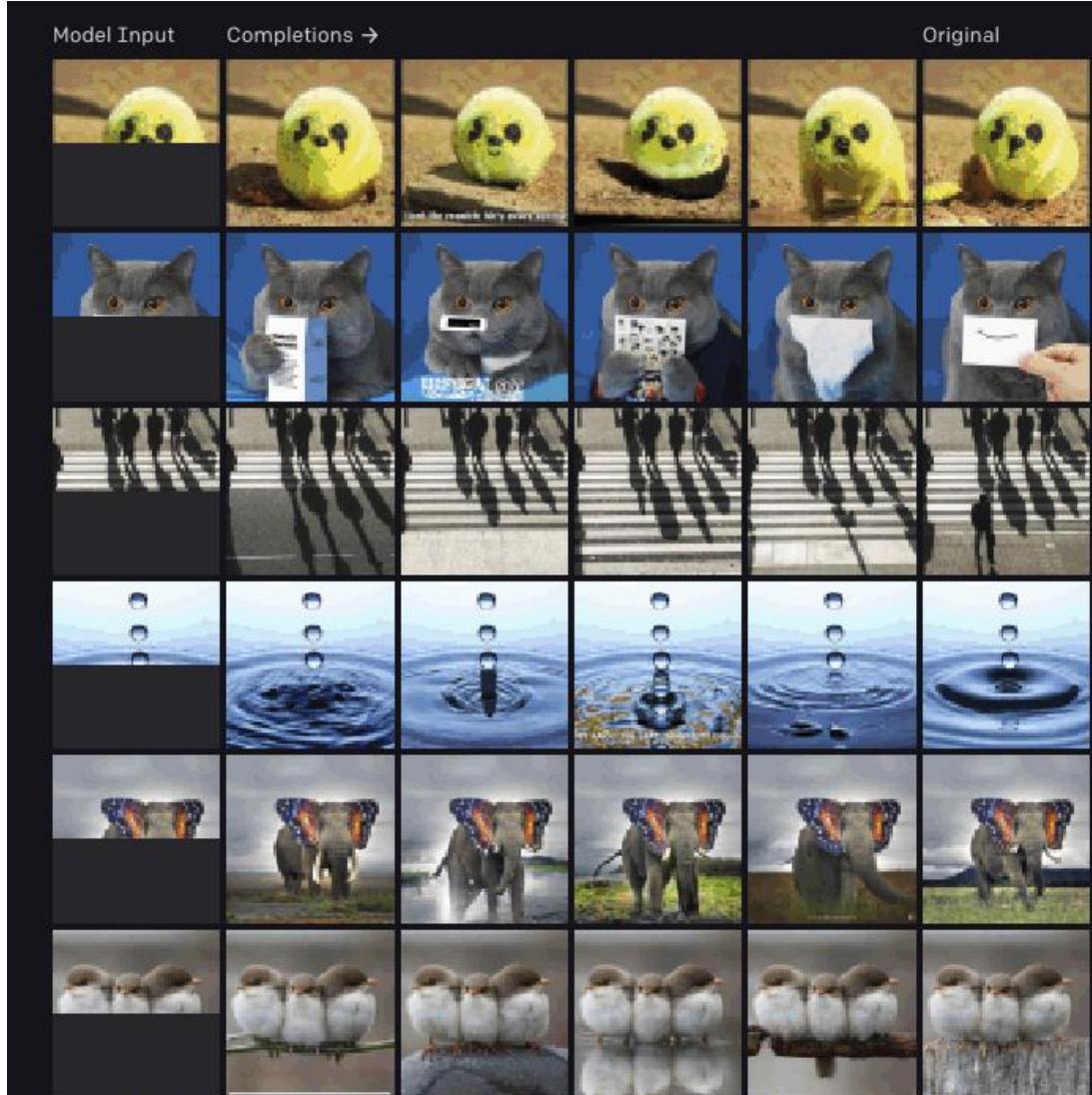
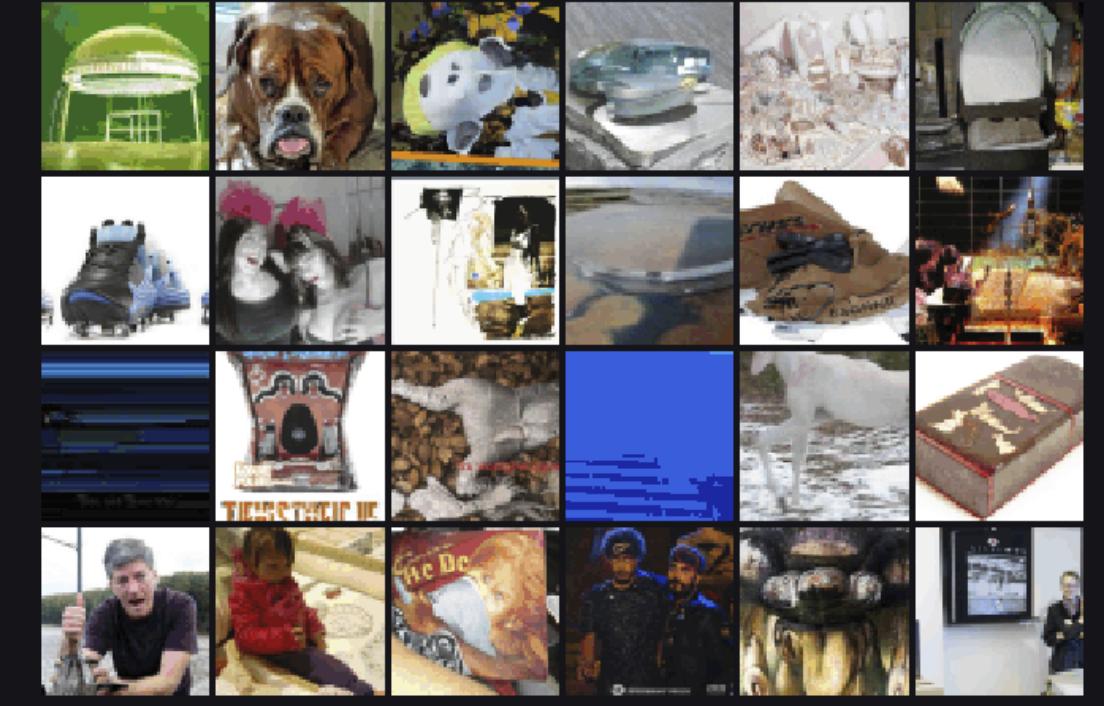


Table 2. On the left are image completions from our best conditional generation model, where we sample the second half. On the right are samples from our four-fold super-resolution model trained on CIFAR-10. Our images look realistic and plausible, show good diversity among the completion samples and observe the outputs carry surprising details for coarse inputs in super-resolution.

Image GPT



Samples



<https://openai.com/blog/image-gpt/>

Image GPT

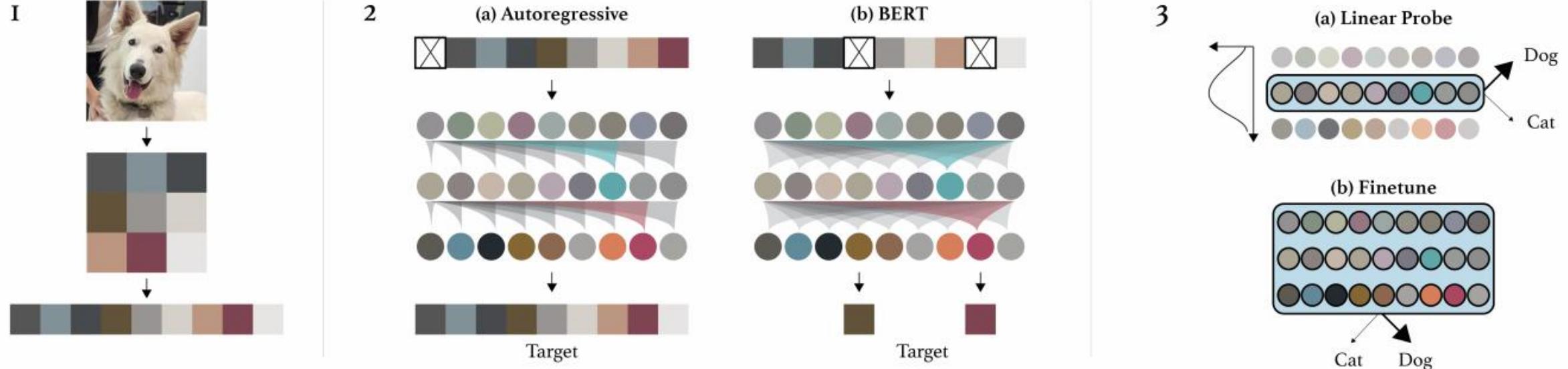


Figure 1. An overview of our approach. First, we pre-process raw images by resizing to a low resolution and reshaping into a 1D sequence. We then chose one of two pre-training objectives, auto-regressive next pixel prediction or masked pixel prediction. Finally, we evaluate the representations learned by these objectives with linear probes or fine-tuning.

Image GPT

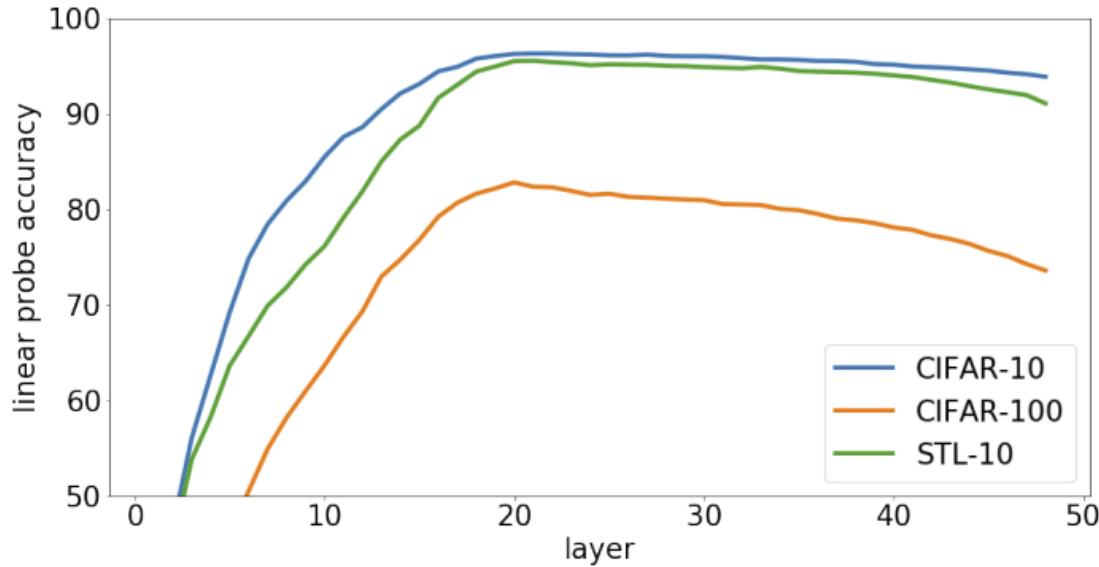


Figure 2. Representation quality depends on the layer from which we extract features. In contrast with supervised models, the best representations for these generative models lie in the middle of the network. We plot this unimodal dependence on depth by showing linear probes for iGPT-L on CIFAR-10, CIFAR-100, and STL-10.

Table 2. Comparing linear probe accuracies between our models and state-of-the-art self-supervised models. A blank input resolution (IR) corresponds to a model working at standard ImageNet resolution. We report the best performing configuration for each contrastive method, finding that our models achieve comparable performance.

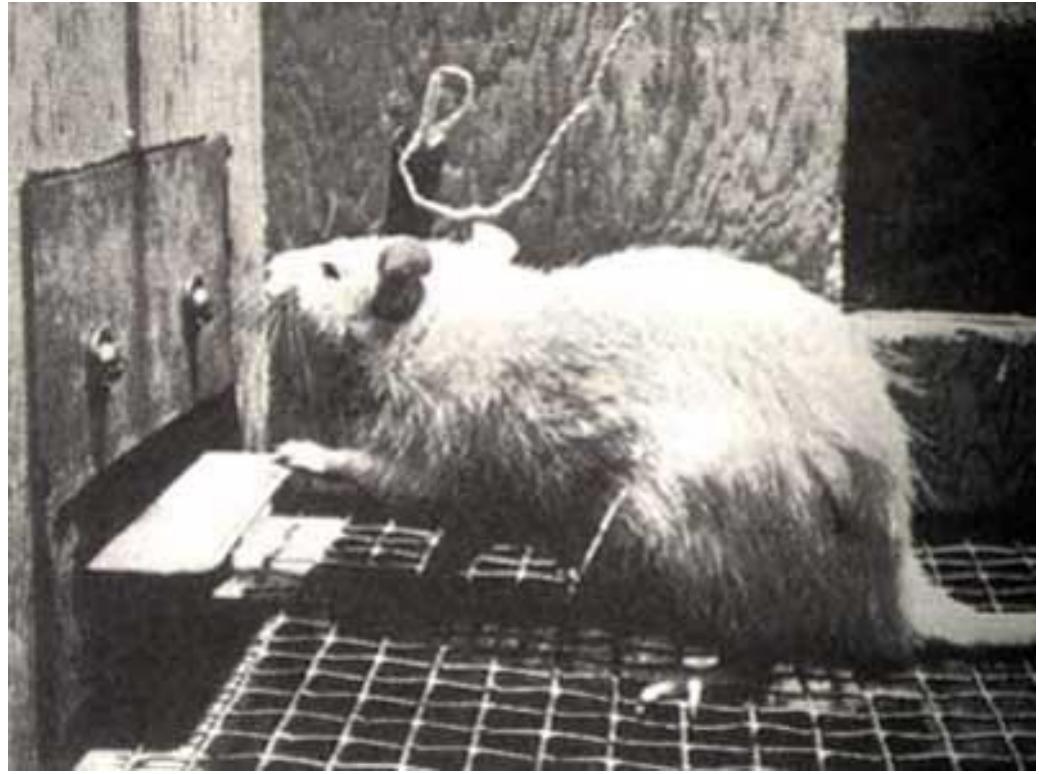
Method	IR	Params (M)	Features	Acc
Rotation	orig.	86	8192	55.4
iGPT-L	$32^2 \cdot 3$	1362	1536	60.3
BigBiGAN	orig.	86	8192	61.3
iGPT-L	$48^2 \cdot 3$	1362	1536	65.2
AMDIM	orig.	626	8192	68.1
MoCo	orig.	375	8192	68.6
iGPT-XL	$64^2 \cdot 3$	6801	3072	68.7
SimCLR	orig.	24	2048	69.3
CPC v2	orig.	303	8192	71.5
iGPT-XL	$64^2 \cdot 3$	6801	15360	72.0
SimCLR	orig.	375	8192	76.5

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More

Introduction to deep reinforcement learning

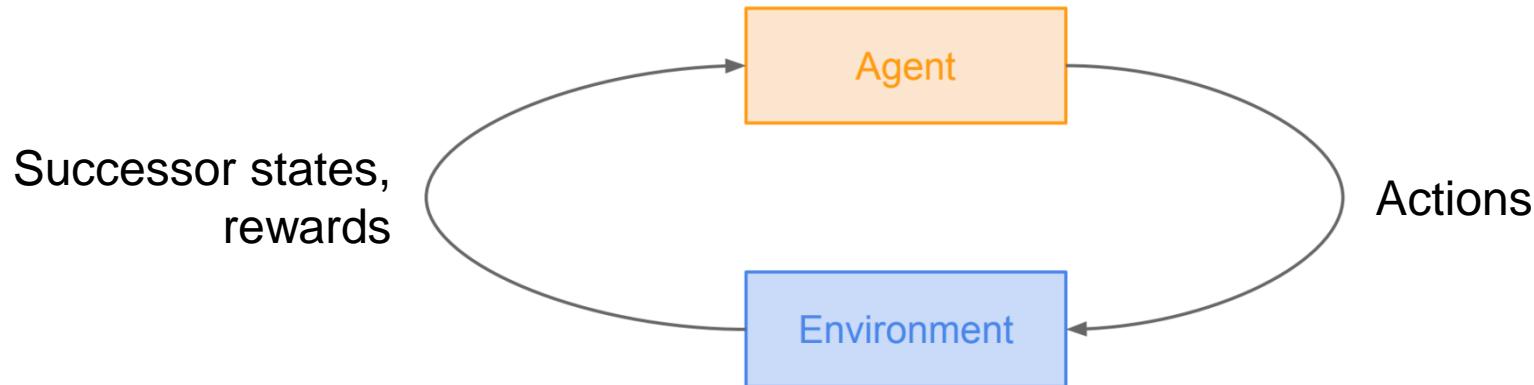


Outline

- Introduction to reinforcement learning
- Markov Decision Process (MDP) formalism
- The Bellman equation
- Q-learning
- Deep Q networks (DQN)
- Policy Gradient Methods

Reinforcement learning (RL)

- Setting: agent that can take *actions* affecting the *state* of the environment and observe occasional *rewards* that depend on the state
- Goal: learn a *policy* (mapping from states to actions) to maximize expected reward over time



RL vs. supervised learning

- **Reinforcement learning loop**
 - From state s , take action a determined by *policy* $\pi(s)$
 - Environment selects next state s' based on *transition model* $P(s'|s, a)$
 - Observe s' and reward $r(s')$, update policy
- **Supervised learning loop**
 - Get input x_i sampled from data distribution
 - Use model with parameters w to predict output y
 - Observe target output y_i and loss $l(w, x_i, y_i)$
 - Update w to reduce loss: $w \leftarrow w - \eta \nabla l(w, x_i, y_i)$

RL vs. supervised learning

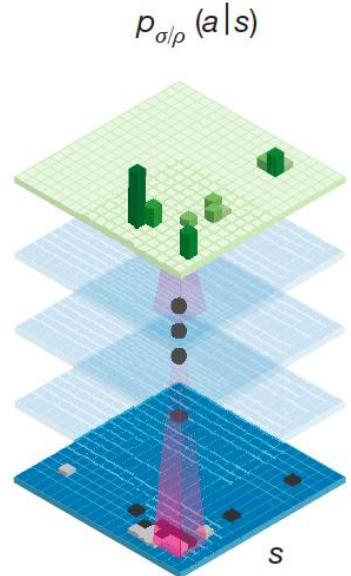
- **Reinforcement learning**
 - Agent's actions affect the environment and help to determine next observation
 - Rewards may be sparse
 - Rewards are not differentiable w.r.t. model parameters
- **Supervised learning**
 - Next input does not depend on previous inputs or agent's predictions
 - There is a supervision signal at every step
 - Loss is differentiable w.r.t. model parameters

Example applications of deep RL

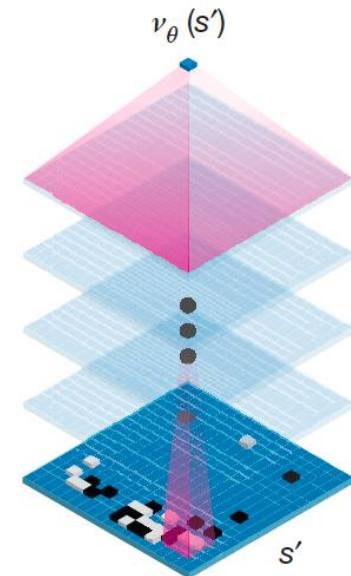
- AlphaGo and AlphaZero



Policy network



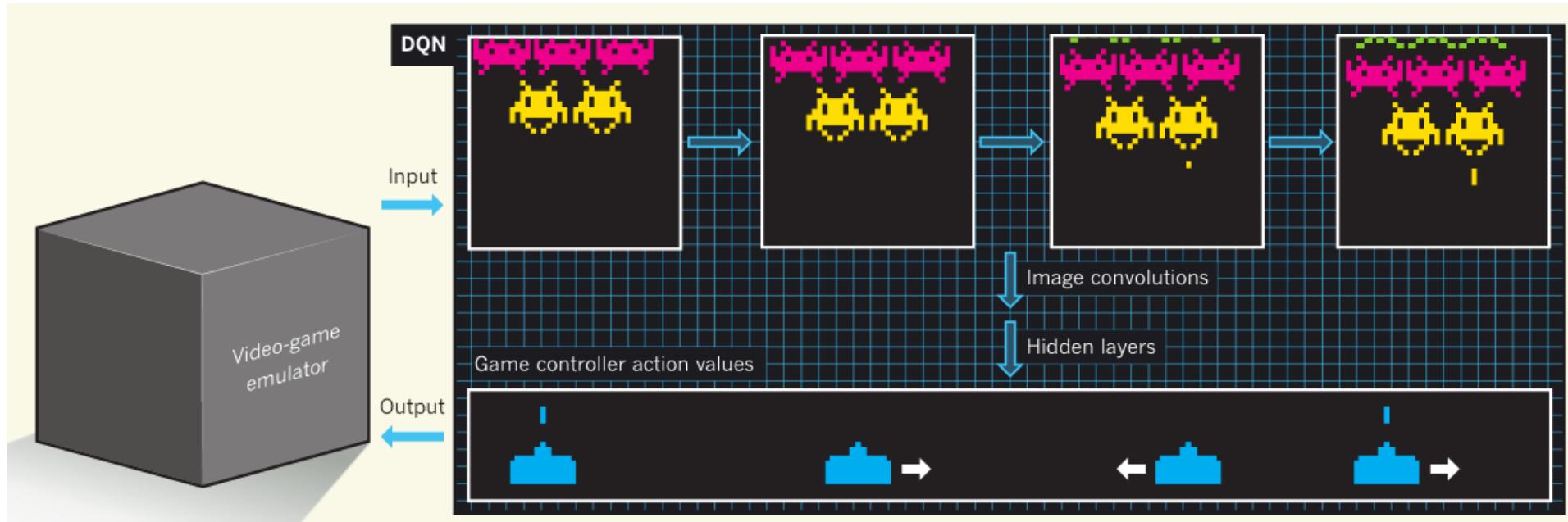
Value network



<https://deepmind.com/research/alphago/>

Example applications of deep RL

- Playing video games



[Video](#)

Example applications of deep RL

- Sensorimotor learning

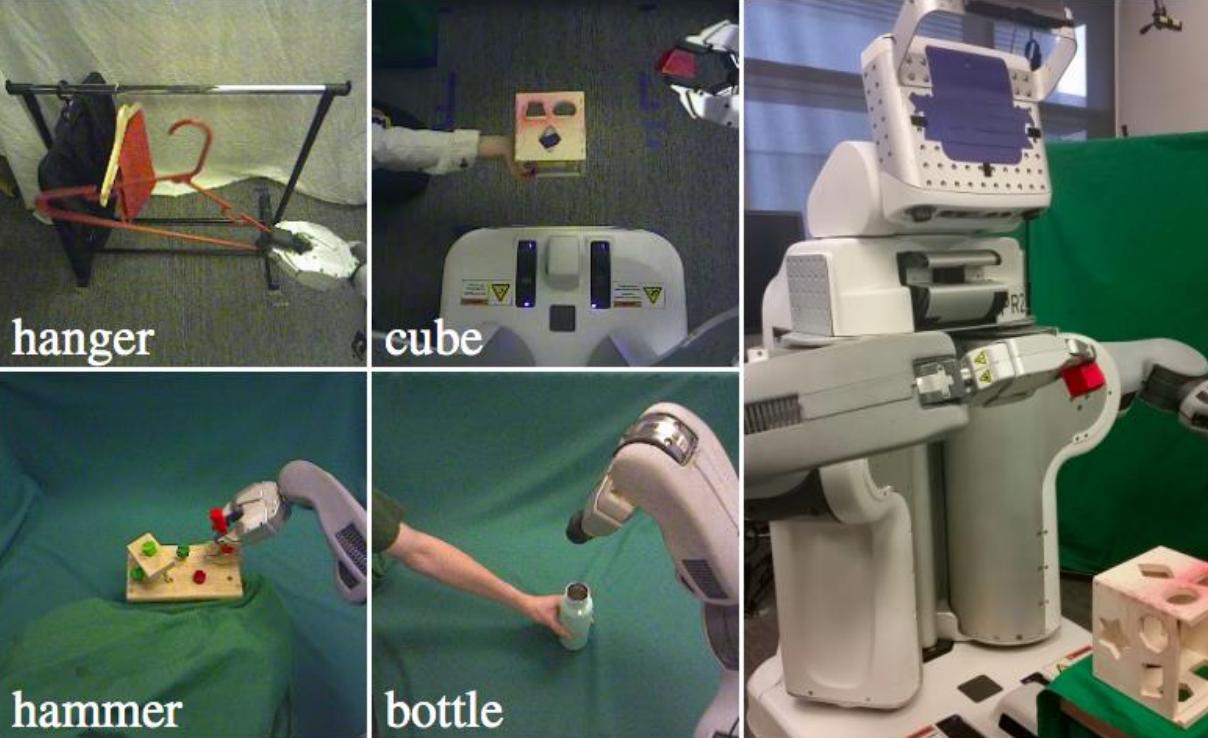
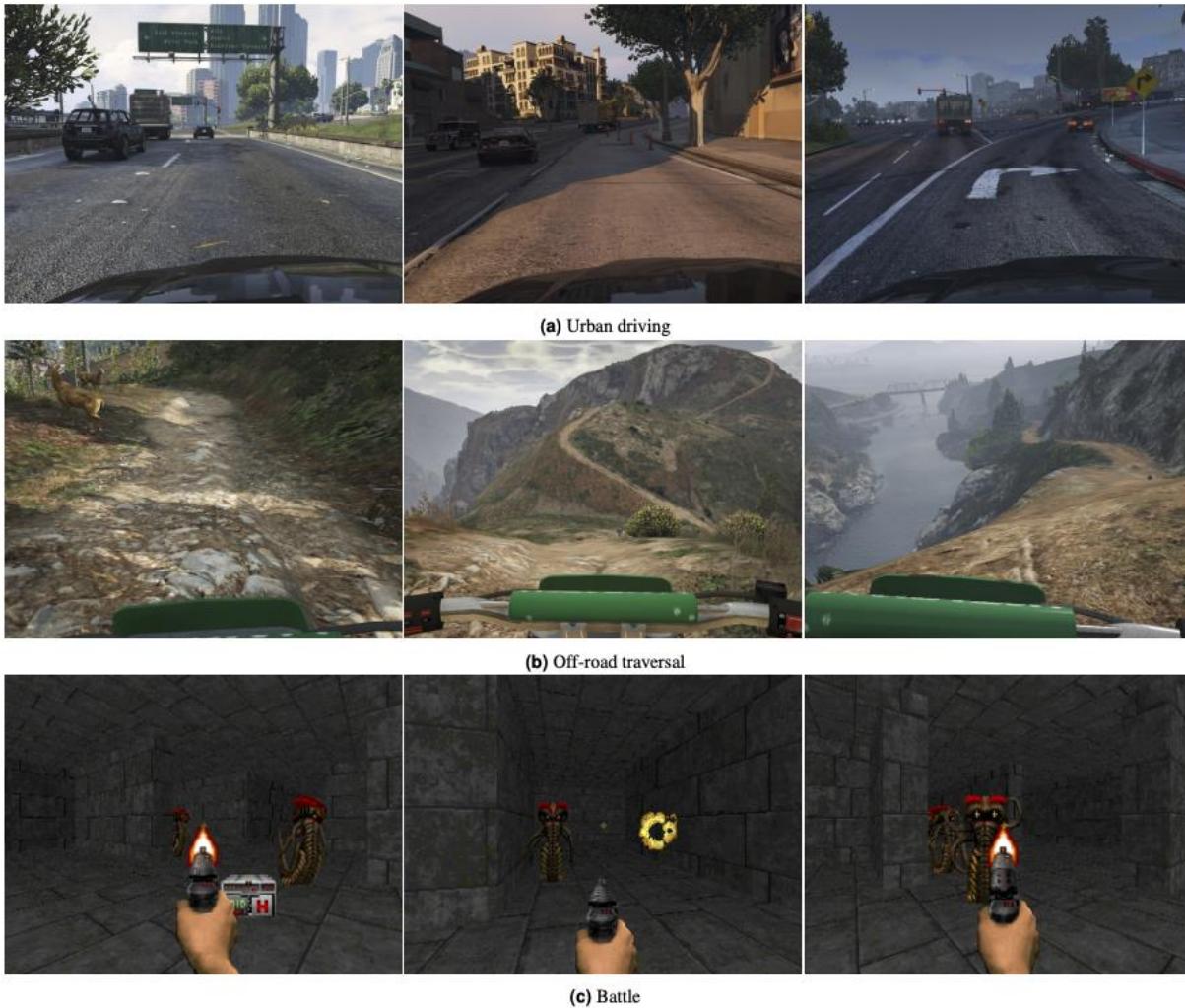


Fig. 1: Our method learns visuomotor policies that directly use camera image observations (left) to set motor torques on a PR2 robot (right).

[Video](#)

Example applications of deep RL

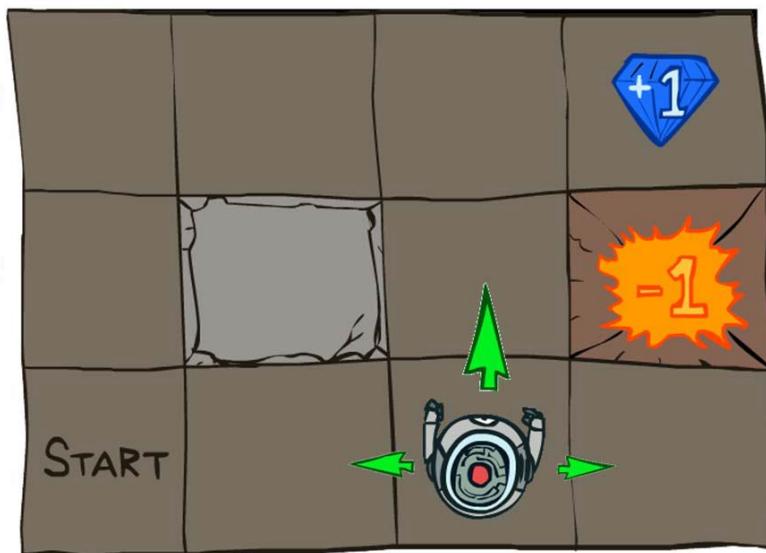
- Sensorimotor learning



Formalism: Markov Decision Processes

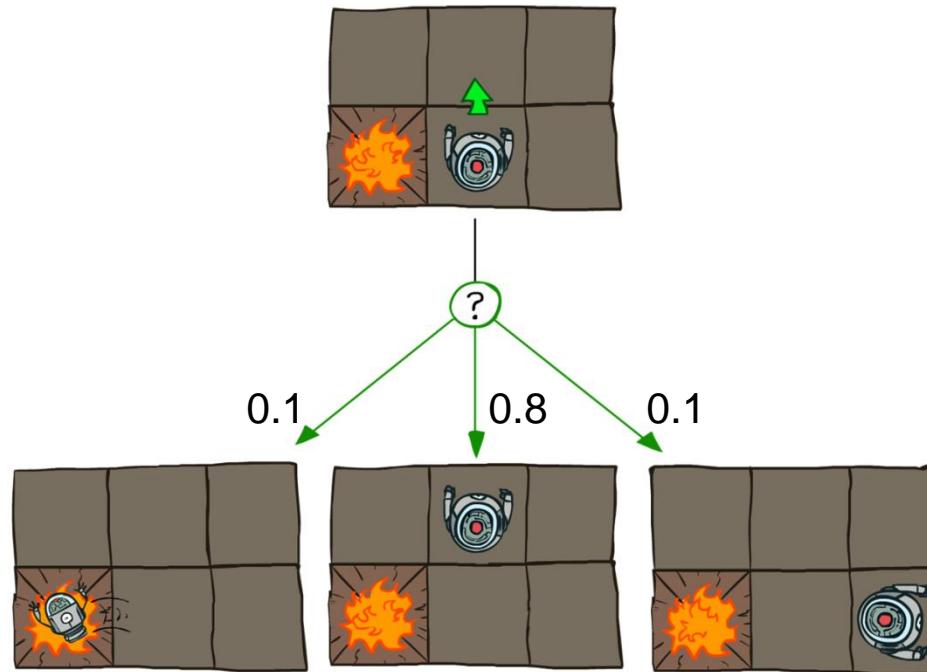
- Components:
 - **States** s , beginning with initial state s_0
 - **Actions** a
 - **Transition model** $P(s' | s, a)$
 - *Markov assumption:* the probability of going to s' from s depends only on s and a and not on any other past actions or states
 - **Reward function** $r(s)$
- **Policy** $\pi(s)$: the action that an agent takes in any given state
 - The “solution” to an MDP

Example MDP: Grid world



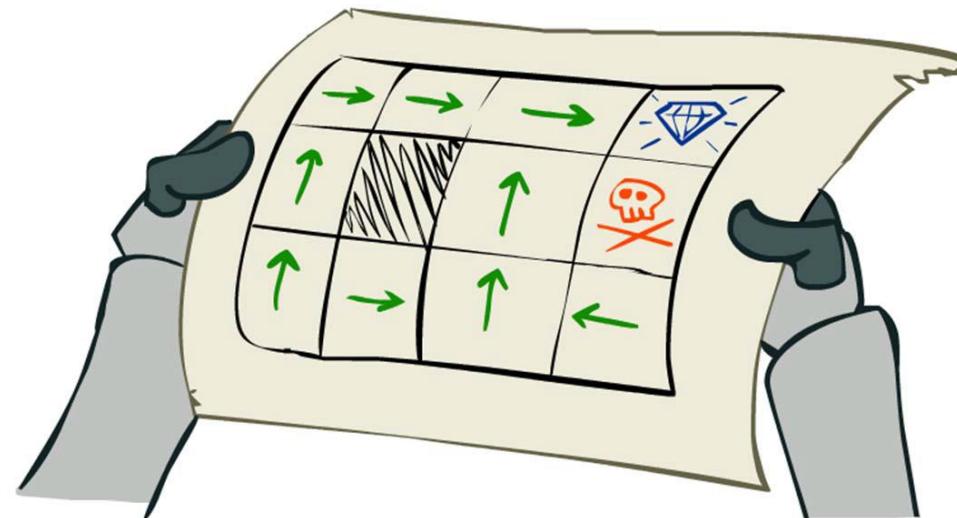
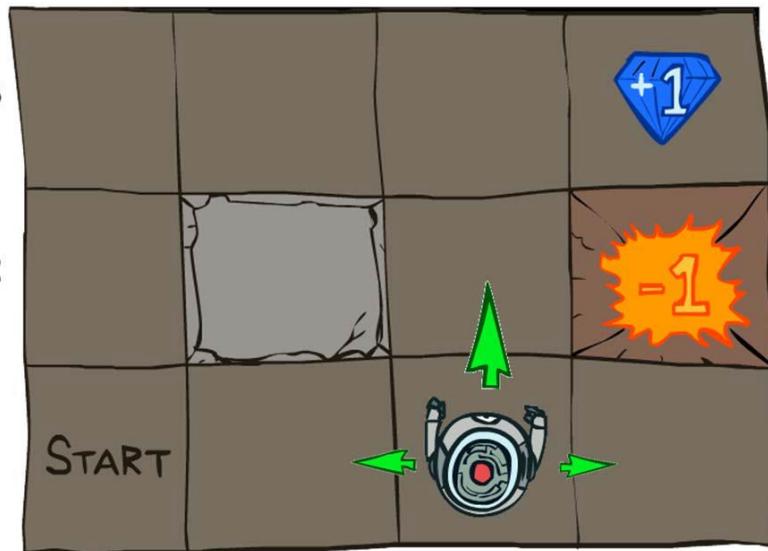
$r(s) = -0.04$ for
every non-terminal
state

Transition model:



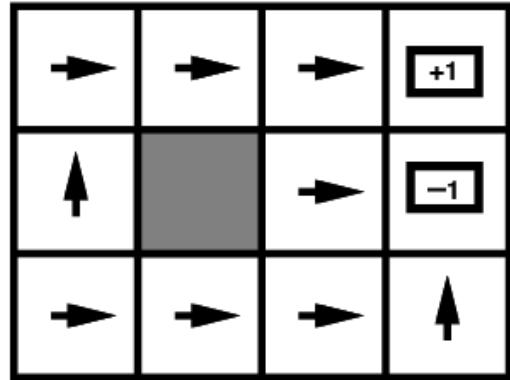
Example MDP: Grid world

- Goal: find the best policy

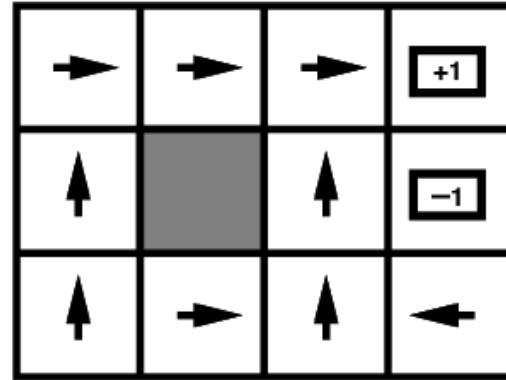


Example MDP: Grid world

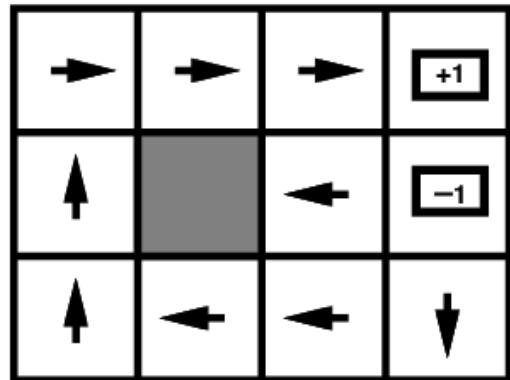
- Optimal policies for various values of $r(s)$:



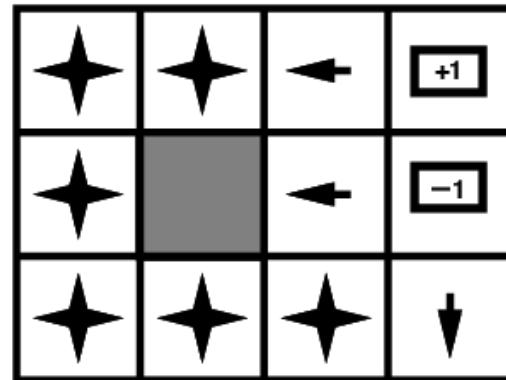
$$R(s) < -1.6284$$



$$-0.4278 < R(s) < -0.0850$$



$$-0.0221 < R(s) < 0$$



$$R(s) > 0$$

Rewards of state sequences

- Suppose that following policy π starting in state s_0 leads to a sequence s_0, s_1, s_2, \dots
- The *cumulative reward* of the sequence is $\sum_{t \geq 0} r(s_t)$
- **Problem:** state sequences can vary in length or even be infinite
- **Solution:** redefine cumulative reward as sum of rewards *discounted* by a factor γ



1

Worth Now



γ

Worth Next Step



γ^2

Worth In Two Steps

Discounting

- Discounted cumulative reward:

$$\begin{aligned} & r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \gamma^3 r(s_3) + \dots \\ &= \sum_{t \geq 0} \gamma^t r(s_t), \quad 0 < \gamma \leq 1 \end{aligned}$$

- Sum is bounded by $\frac{r_{\max}}{1-\gamma}$
- Helps algorithms converge

Value function

- The *value function* $V^\pi(s)$ of a state s w.r.t. policy π is the expected cumulative reward of following that policy starting in s :

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

with $a_t = \pi(s_t), s_{t+1} \sim P(\cdot | s_t, a_t)$

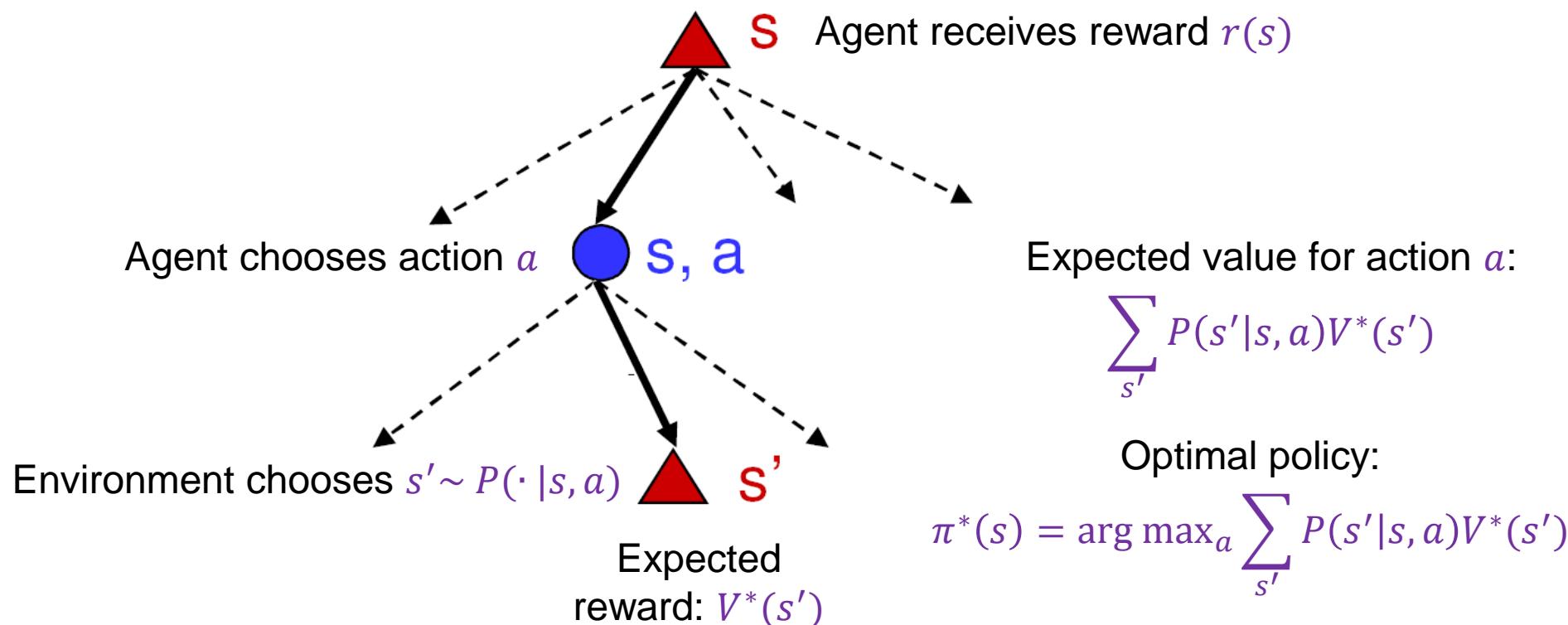
- The *optimal value* of a state is the value achievable by following the best possible policy:

$$V^*(s) = \max_{\pi} \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r(s_t) \mid s_0 = s, \pi \right]$$

The Bellman equation

- Recursive relationship between optimal values of successive states:

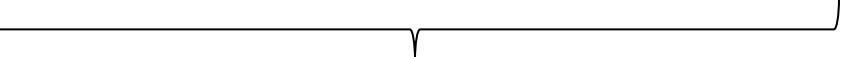
$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s, a)V^*(s')$$



The Bellman equation

- Recursive relationship between optimal values of successive states:

$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s, a)V^*(s')$$

Reward in current state 

Discounted expected future reward assuming agent follows the optimal policy

Outline

- Introduction to reinforcement learning
- Markov Decision Process (MDP) formalism
- The Bellman equation
- Q-learning

Q-learning

- Optimal policy in terms of the state value function:

$$\pi^*(s) = \arg \max_a \sum_{s'} P(s'|s, a) V^*(s')$$

- To use this in practice, we need to know the transition model
- It is more convenient to define the value of a *state-action pair*:

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

Q-value function

- The *optimal* Q-value:

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

- What is the relationship between $V^*(s)$ and $Q^*(s, a)$?

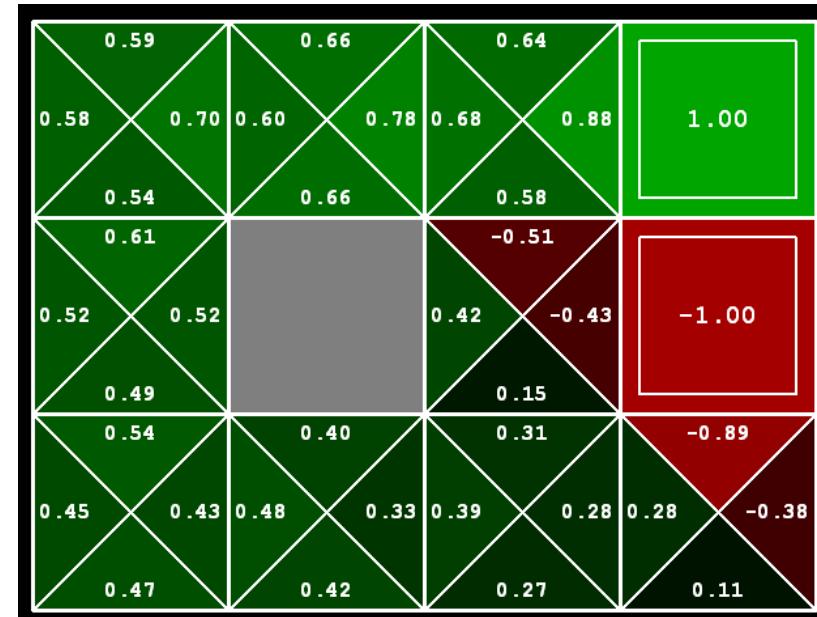
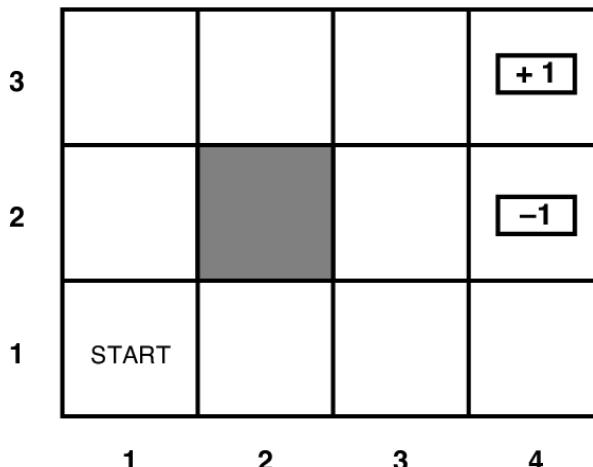
$$V^*(s) = \max_a Q^*(s, a)$$

- What is the optimal policy?

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

Q-value function

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



Bellman equation for Q-values

$$V^*(s) = \max_a Q^*(s, a)$$

- Regular Bellman equation:

$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s, a) V^*(s')$$

- Bellman equation for Q-values:

$$\begin{aligned} Q^*(s, a) &= r(s) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a') \\ &= \mathbb{E}_{s' \sim P(\cdot|s, a)} [r(s) + \gamma \max_{a'} Q^*(s', a') | s, a] \end{aligned}$$

Finding the optimal policy

- The Bellman equation is a constraint on Q-values of successive states:

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

- We could think of $Q^*(s, a)$ as a table indexed by states and actions, and try to solve the system of Bellman equations to fill in the unknown values of the table
- Problem:** state spaces for interesting problems are huge
- Solution:** approximate Q-values using a parametric function:

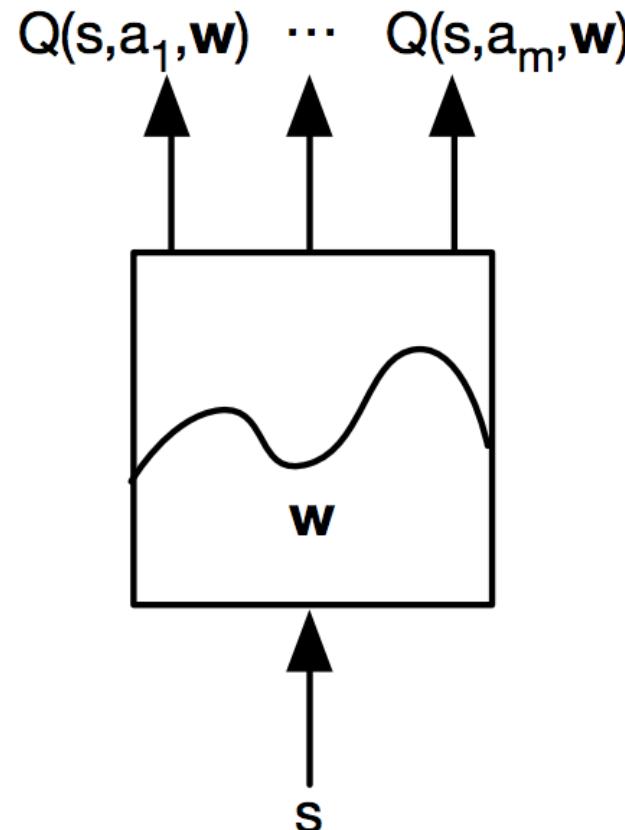
$$Q^*(s, a) \approx Q_w(s, a)$$

Outline

- Introduction to reinforcement learning
- Markov Decision Process (MDP) formalism and classical Bellman equation
- Q-learning
- Deep Q networks

Deep Q-learning

- Train a deep neural network to estimate Q-values:



Source: [D. Silver](#)

Deep Q-learning

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

- Idea: at each iteration i of training, update model parameters w_i to “nudge” the left-hand side toward the right-hand “target”:

$$y_i(s, a) = \mathbb{E}_{s' \sim P(\cdot | s, a)} [r(s) + \gamma \max_{a'} Q_{w_{i-1}}(s', a') | s, a]$$

- Loss function:

$$L_i(w_i) = \mathbb{E}_{s, a \sim \rho} [(y_i(s, a) - Q_{w_i}(s, a))^2]$$

where ρ is a *behavior distribution*

Deep Q-learning

- Target: $y_i(s, a) = \mathbb{E}_{s' \sim P(\cdot|s, a)} [r(s) + \gamma \max_{a'} Q_{w_{i-1}}(s', a')|s, a]$
- Loss: $L_i(w_i) = \mathbb{E}_{s, a \sim \rho} [(y_i(s, a) - Q_{w_i}(s, a))^2]$

- Gradient update:

$$\begin{aligned}\nabla_{w_i} L(w_i) &= \mathbb{E}_{s, a \sim \rho} [(y_i(s, a) - Q_{w_i}(s, a)) \nabla_{w_i} Q_{w_i}(s, a)] \\ &= \mathbb{E}_{s, a \sim \rho, s'} [(r(s) + \gamma \max_{a'} Q_{w_{i-1}}(s', a') - Q_{w_i}(s, a)) \nabla_{w_i} Q_{w_i}(s, a)]\end{aligned}$$

- SGD training: replace expectation by sampling *experiences* (s, a, s') using behavior distribution and transition model

Deep Q-learning in practice

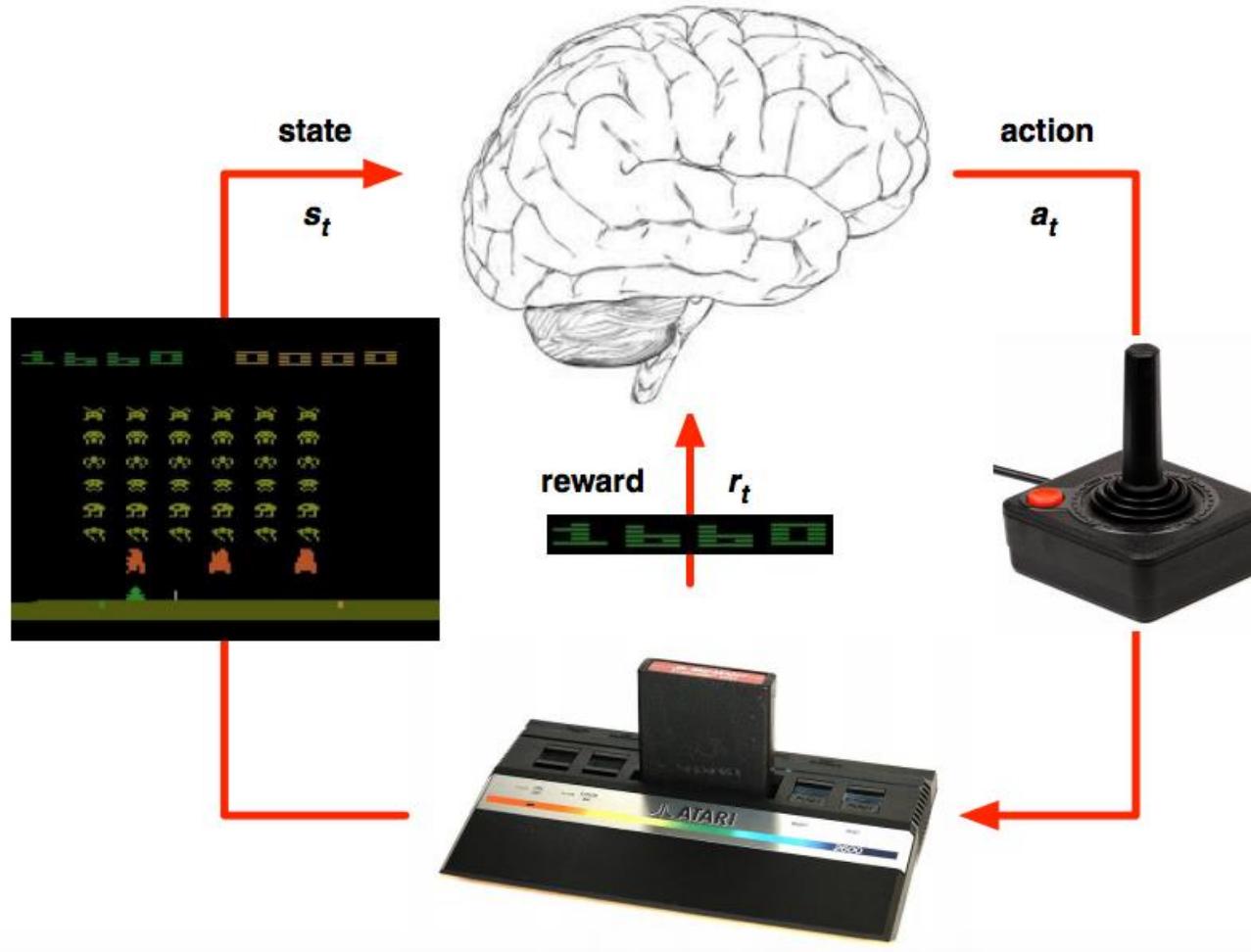
- Training is prone to instability
 - Unlike in supervised learning, the targets themselves are moving!
 - Successive experiences are correlated and dependent on the policy
 - Policy may change rapidly with slight changes to parameters, leading to drastic change in data distribution
- Solutions
 - Freeze target Q network
 - Use *experience replay*

Experience replay

- At each time step:
 - Take action a_t according to *epsilon-greedy policy*
 - Store experience $(s_t, a_t, r_{t+1}, s_{t+1})$ in *replay memory buffer*
 - Randomly sample *mini-batch* of experiences from the buffer

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
...
$s_t, a_t, r_{t+1}, s_{t+1}$

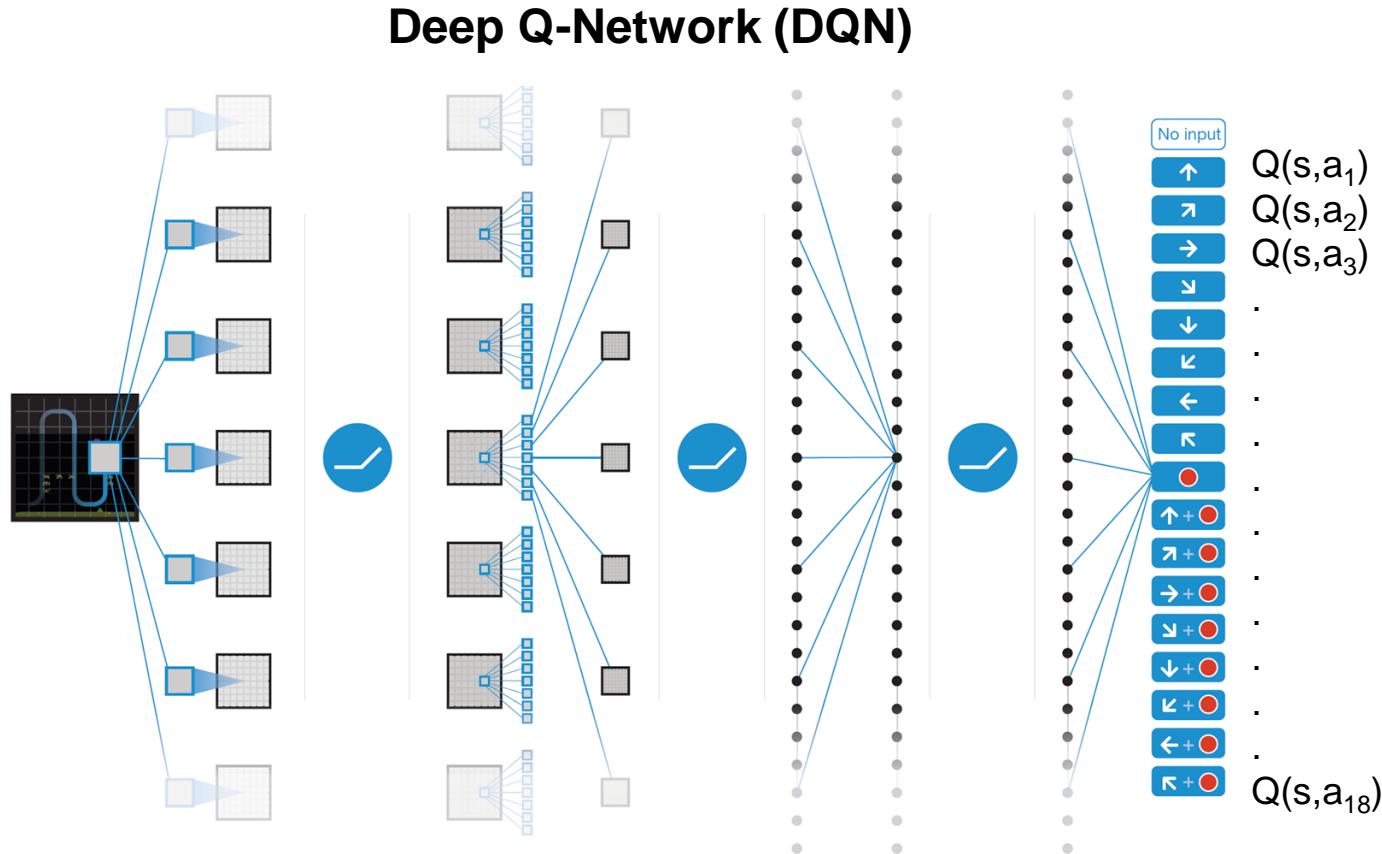
Deep Q-learning in Atari



V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller,
[Human-level control through deep reinforcement learning](#), *Nature* 2015

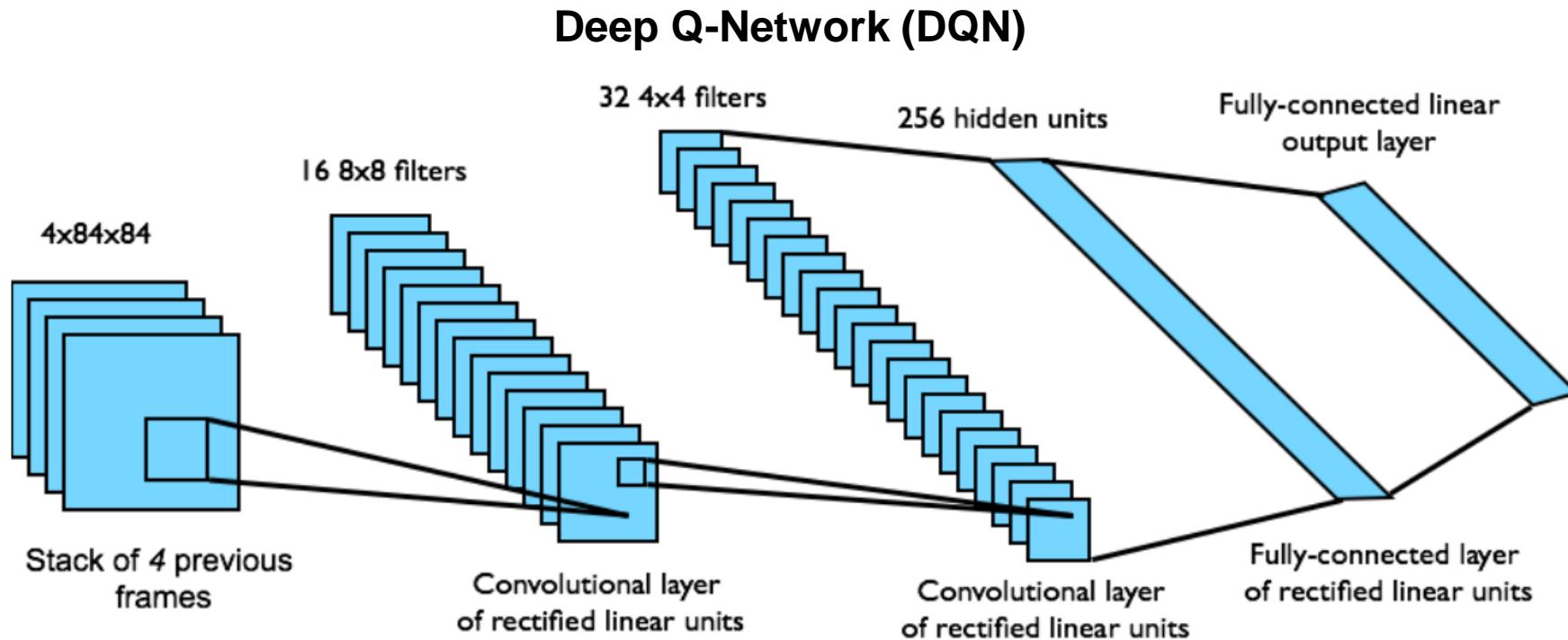
Deep Q-learning in Atari

- End-to-end learning of $Q(s, a)$ from pixels s
- Output is $Q(s, a)$ for 18 joystick/button configurations
- Reward is change in score for that step



Deep Q-learning in Atari

- Input state is stack of raw pixels (grayscale) from last 4 frames
- Network architecture and hyperparameters fixed for all games

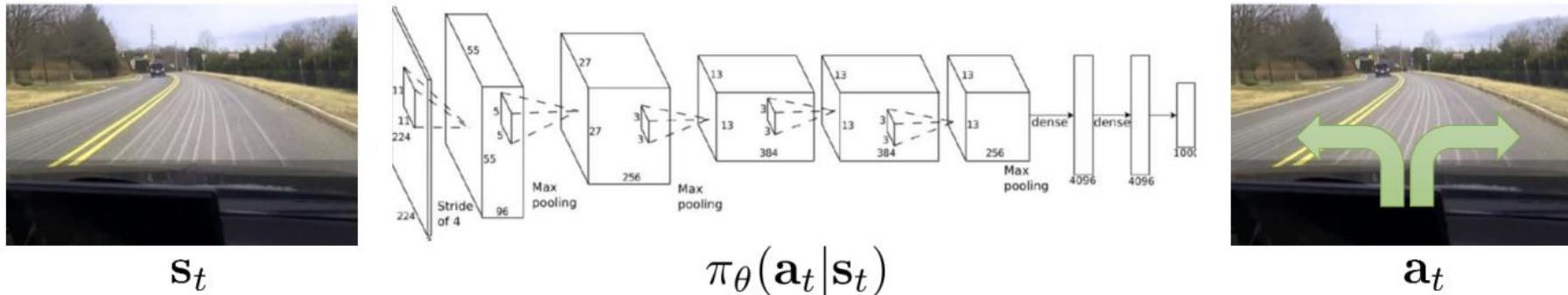


Breakout demo



<https://www.youtube.com/watch?v=TmPfTpjtdgg>

Policy Gradient Methods



Sources: [Stanford CS 231n](#), [Berkeley Deep RL course](#),
[David Silver's RL course](#)

Policy Gradient Methods

- Instead of indirectly representing the policy using Q-values, it can be more efficient to parameterize and learn it directly
 - Especially in large or continuous action spaces



Stochastic policy representation

- Learn a function giving the probability distribution over actions from the current state:

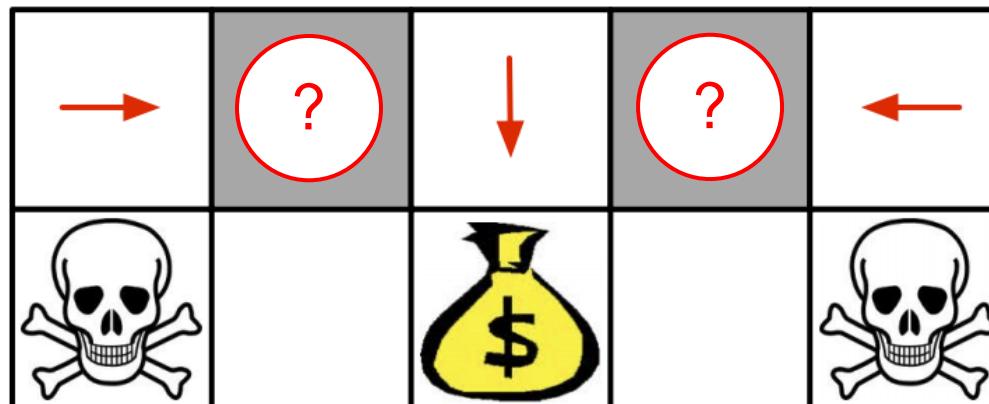
$$\pi_{\theta}(a|s) \approx P(a|s)$$

Stochastic policy representation

- Learn a function giving the probability distribution over actions from the current state:

$$\pi_{\theta}(a|s) \approx P(a|s)$$

- Why stochastic policies?
 - There are examples even of grid world scenarios where only a stochastic policy can reach optimality



Source:
[D. Silver](#)

The agent can't tell the difference between the gray cells

Stochastic policy representation

- Learn a function giving the probability distribution over actions from the current state:

$$\pi_\theta(a|s) \approx P(a|s)$$

- Why stochastic policies?
 - It's mathematically convenient!
 - Softmax policy:

$$\pi_\theta(a|s) = \frac{\exp(f_\theta(s, a))}{\sum_{a'} \exp(f_\theta(s, a'))}$$

- Gaussian policy (for continuous action spaces):

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a - f_\theta(s))^2}{2\sigma^2}\right)$$

Expected value of a policy

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

$$= \mathbb{E}_\tau[r(\tau)]$$

Expectation of return over *trajectories* $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$

$$= \int_{\tau} r(\tau) p(\tau; \theta) d\tau$$

Probability of trajectory τ
under policy with
parameters θ

Finding the policy gradient

$$J(\theta) = \int_{\tau} r(\tau)p(\tau; \theta)d\tau$$

$$\nabla_{\theta} J(\theta) = \int_{\tau} r(\tau)\nabla_{\theta} p(\tau; \theta)d\tau$$

$$= \int_{\tau} r(\tau)p(\tau; \theta) \frac{\nabla_{\theta} p(\tau; \theta)}{p(\tau; \theta)} d\tau$$

$$\begin{aligned} &= \int_{\tau} r(\tau)p(\tau; \theta)\nabla_{\theta} \log p(\tau; \theta) d\tau \\ &= \mathbb{E}_{\tau}[r(\tau)\nabla_{\theta} \log p(\tau; \theta)] \end{aligned}$$

Finding the policy gradient

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau} [r(\tau) \nabla_{\theta} \log p(\tau; \theta)]$$



Probability of trajectory
 $\tau = (s_0, a_0, s_1, a_1, \dots)$

$$p(\tau; \theta) = \prod_{t \geq 0} \pi_{\theta}(a_t | s_t) P(s_{t+1} | s_t, a_t)$$

$$\log p(\tau; \theta) = \sum_{t \geq 0} [\log \pi_{\theta}(a_t | s_t) + \log P(s_{t+1} | s_t, a_t)]$$

$$\nabla_{\theta} \log p(\tau; \theta) = \sum_{t \geq 0} \underbrace{\nabla_{\theta} \log \pi_{\theta}(a_t | s_t)}_{\text{The score function}}$$

The *score function*

Score function $\nabla_{\theta} \log \pi_{\theta}(a|s)$

- For softmax policy:

$$\pi_{\theta}(a|s) = \frac{\exp(f_{\theta}(s, a))}{\sum_{a'} \exp(f_{\theta}(s, a'))}$$

$$\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) = \nabla_{\theta} f_{\theta}(s, a) - \sum_{a'} \pi_{\theta}(a'|s) \nabla_{\theta} f_{\theta}(s, a')$$

- For Gaussian policy:

$$\pi_{\theta}(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a - f_{\theta}(s))^2}{2\sigma^2}\right)$$

$$\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) = \frac{(a - f_{\theta}(s))}{\sigma^2} \nabla_{\theta} f_{\theta}(s) - \text{const.}$$

Finding the policy gradient

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau} [r(\tau) \nabla_{\theta} \log p(\tau; \theta)]$$

$$\nabla_{\theta} \log p(\tau; \theta) = \sum_{t \geq 0} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau} \left[\underbrace{\left(\sum_{t \geq 0} \gamma^t r_t \right)}_{\text{Return of trajectory } \tau} \underbrace{\left(\sum_{t \geq 0} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right)}_{\text{Gradient of log-likelihood of actions under current policy}} \right]$$

- How do we estimate the gradient in practice?

Finding the policy gradient

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau} [r(\tau) \nabla_{\theta} \log p(\tau; \theta)]$$

$$\nabla_{\theta} \log p(\tau; \theta) = \sum_{t \geq 0} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

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

- Stochastic approximation: sample N trajectories τ_1, \dots, τ_N

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=0}^{T_i} \gamma^t r_{i,t} \right) \left(\sum_{t=0}^{T_i} \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \right)$$

REINFORCE algorithm

1. Sample N trajectories τ_i using current policy π_θ
2. Estimate the policy gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N r(\tau_i) \left(\sum_{t=0}^{T_i} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right)$$

3. Update parameters by gradient ascent:

$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$$

REINFORCE: Single-step version

1. In state s , sample action a using current policy π_θ , observe reward r

2. Estimate the policy gradient:

$$\nabla_\theta J(\theta) \approx r \nabla_\theta \log \pi_\theta(a|s)$$

3. Update parameters by gradient ascent:

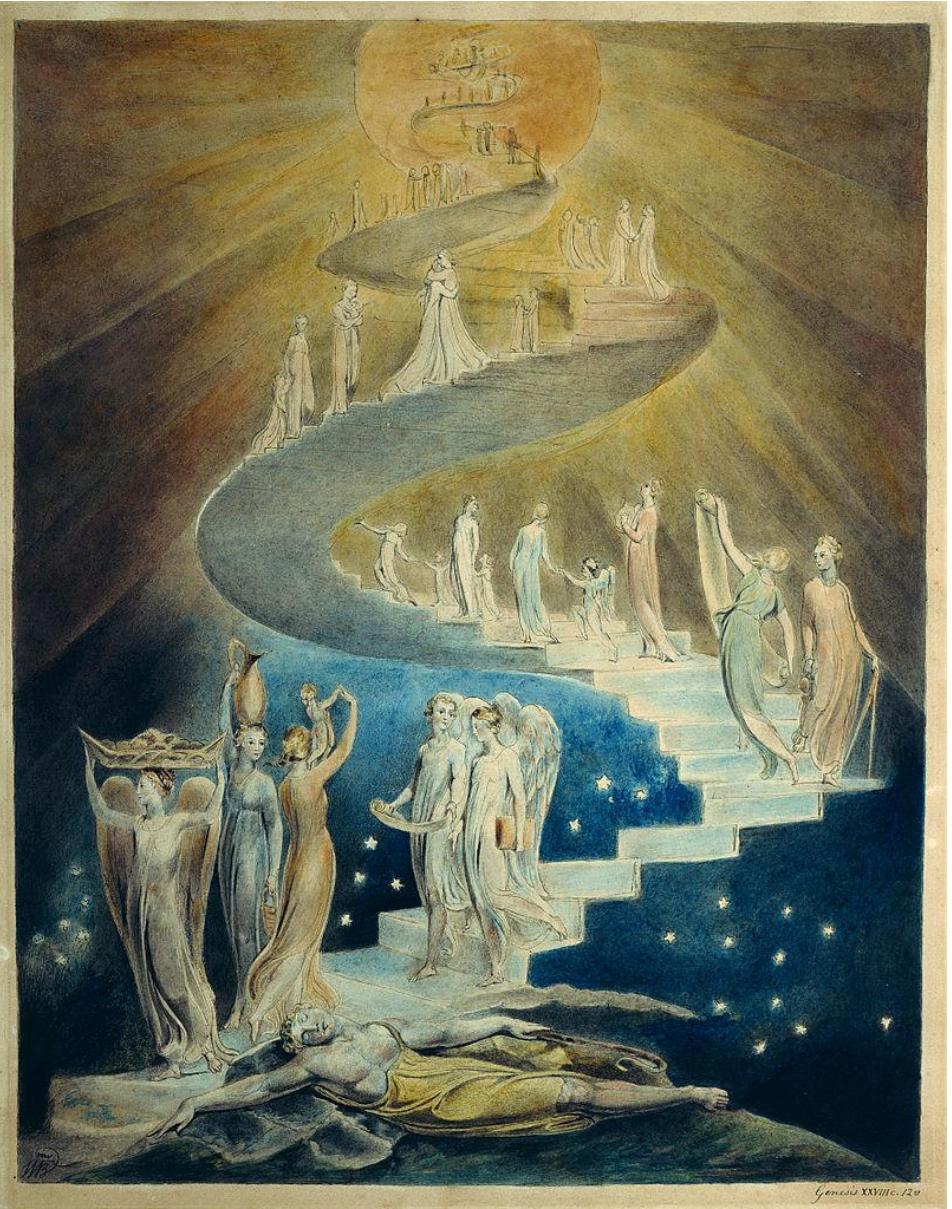
$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$$

- What effect does this update have?
 - Push up the probability of good actions, push down probability of bad actions

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More



William Blake, Jacob's Ladder (1805)

Deep learning trends

(a non-comprehensive, non-objective view)

Outline

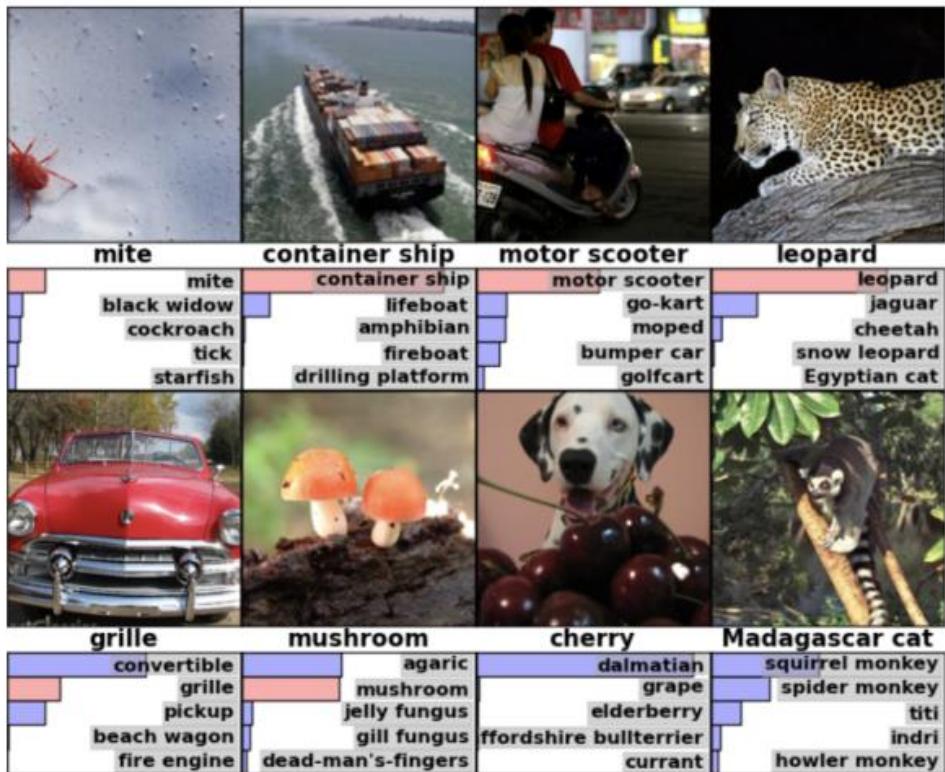
- Vision
- Embodied learning (RL, robotics)
- Language

Deep learning in vision: Where are we now?

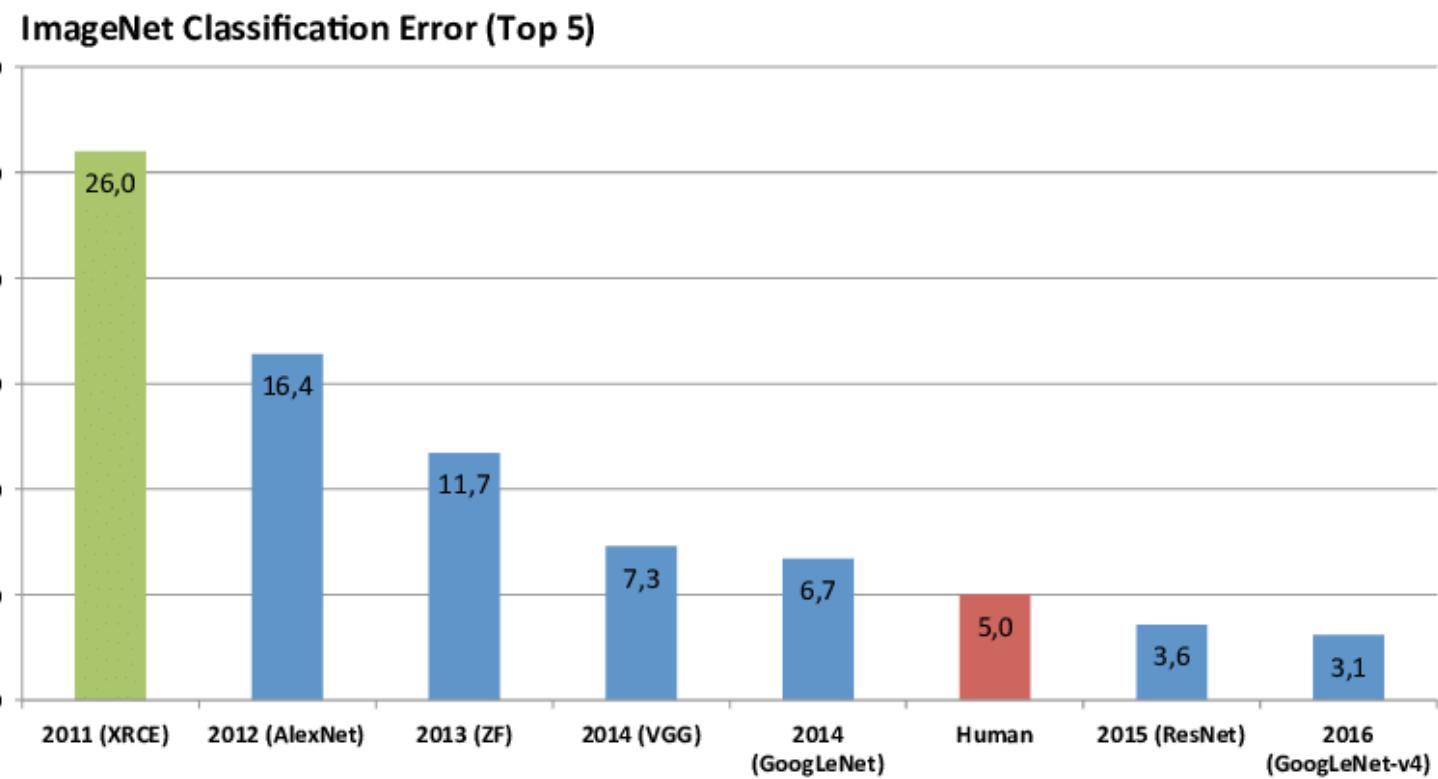
- Glass is half full?
 - DL methods *actually work* and are better than older methods in most ways
 - Good methodologies: standardized benchmarks and metrics for many problems, quantitative comparisons, ablation studies
 - Good infrastructure: deep learning packages, cloud services, etc.
 - Culture of code sharing and reproducibility
- Glass is half empty?
 - Too much focus on benchmarks and numbers
 - Number of papers is exploding but diversity of topics is not necessarily increasing
 - Cutting-edge research is becoming prohibitively resource-intensive
 - Core benchmarks (ImageNet, COCO) are likely saturating but bigger datasets (or larger amounts of computing power) are not yet generally accessible

ImageNet: Asset or liability?

- Performance on the basic classification task has saturated



[ILSVRC Challenge](#)



[Figure source](#)

ImageNet: Asset or liability?

- Performance on the basic classification task has saturated

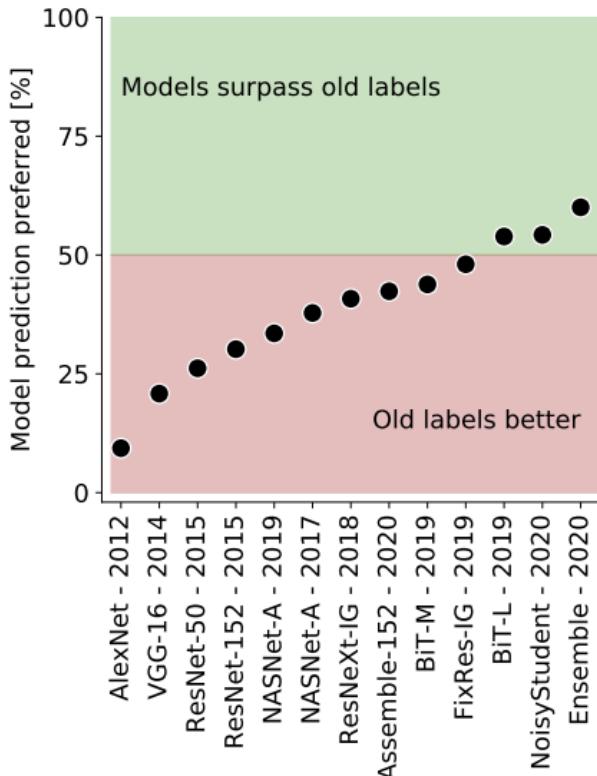


Figure 1: When presented with a model's prediction and the original ImageNet label, human annotators now prefer model predictions on average (Section 4). Nevertheless, there remains considerable progress to be made before fully capturing human preferences.

ImageNet: Asset or liability?

- Attaching labels to images is not very meaningful in the first place

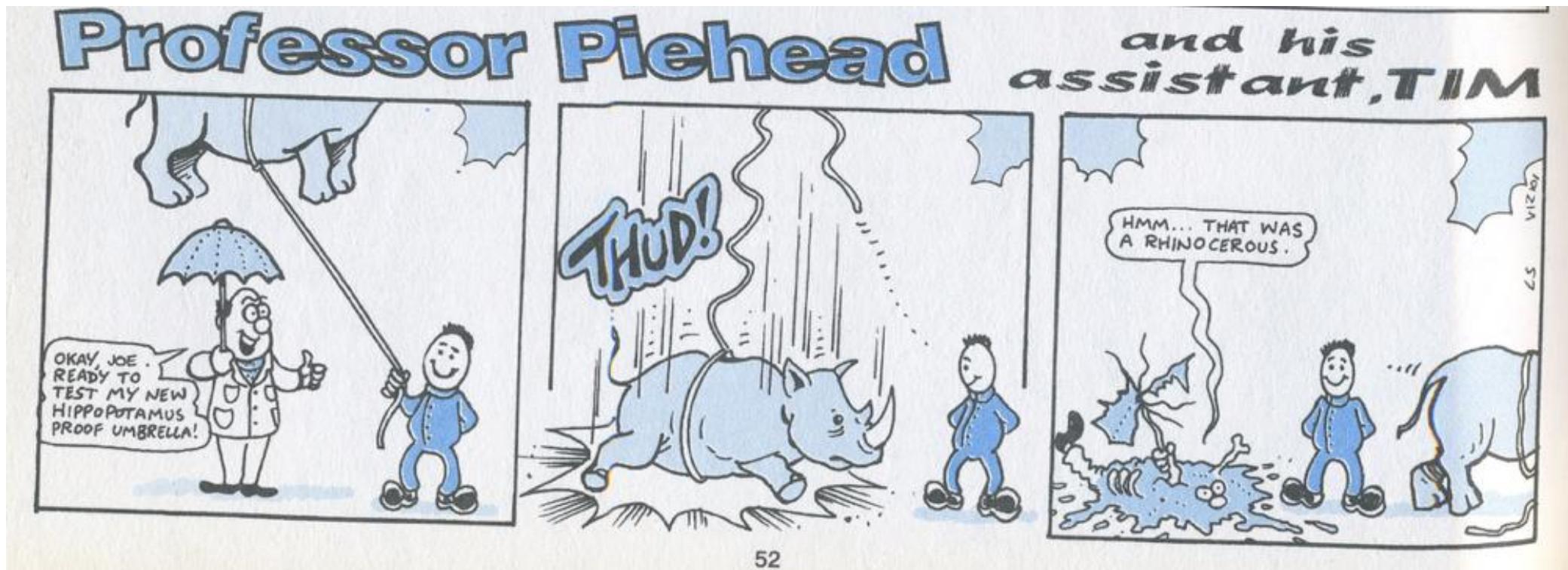


"Now! ... That should clear up
a few things around here!"

Favorite cartoon of J. Koenderink

ImageNet: Asset or liability?

- Attaching labels to images is not very meaningful in the first place

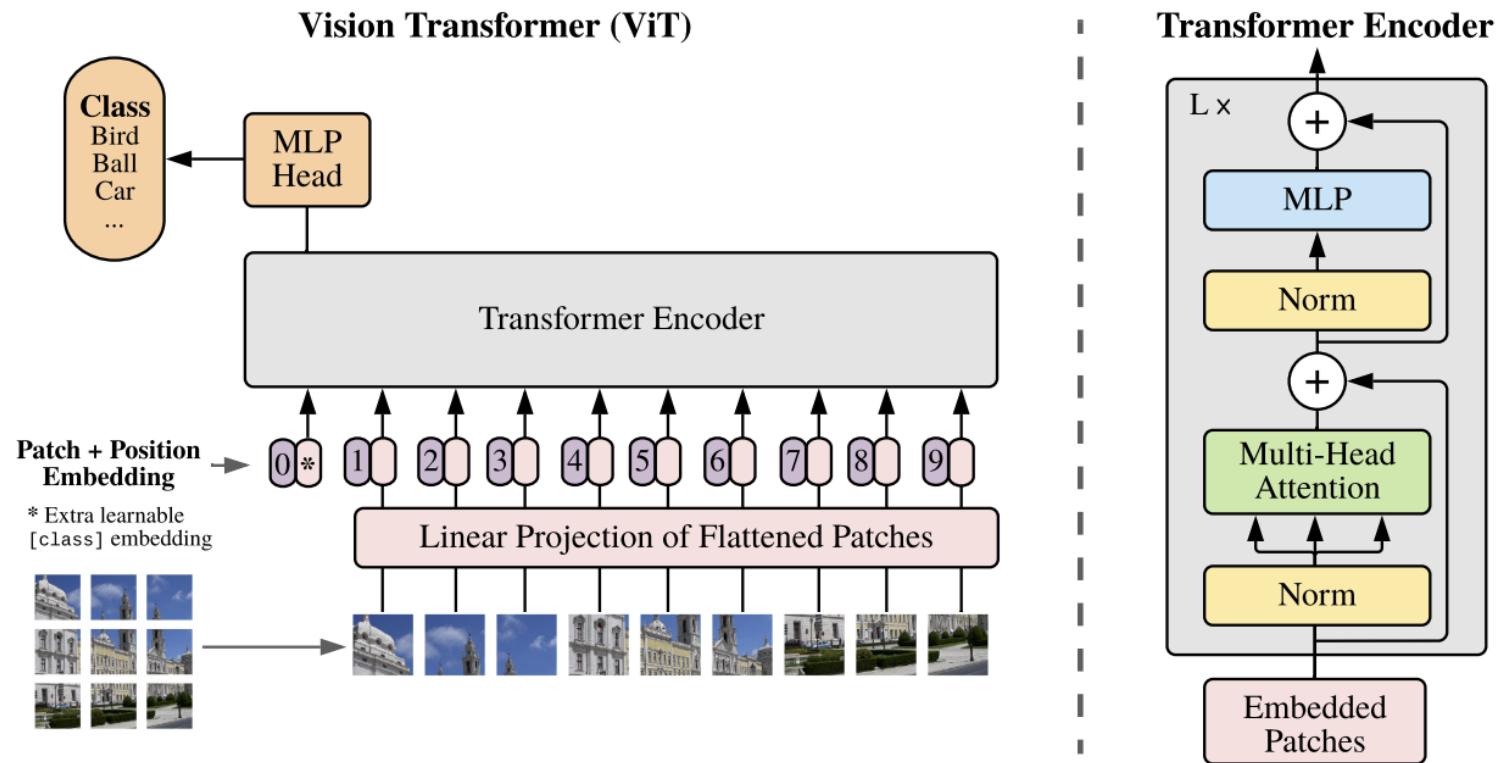


How can we move forward?

- Develop the next generation of architectures

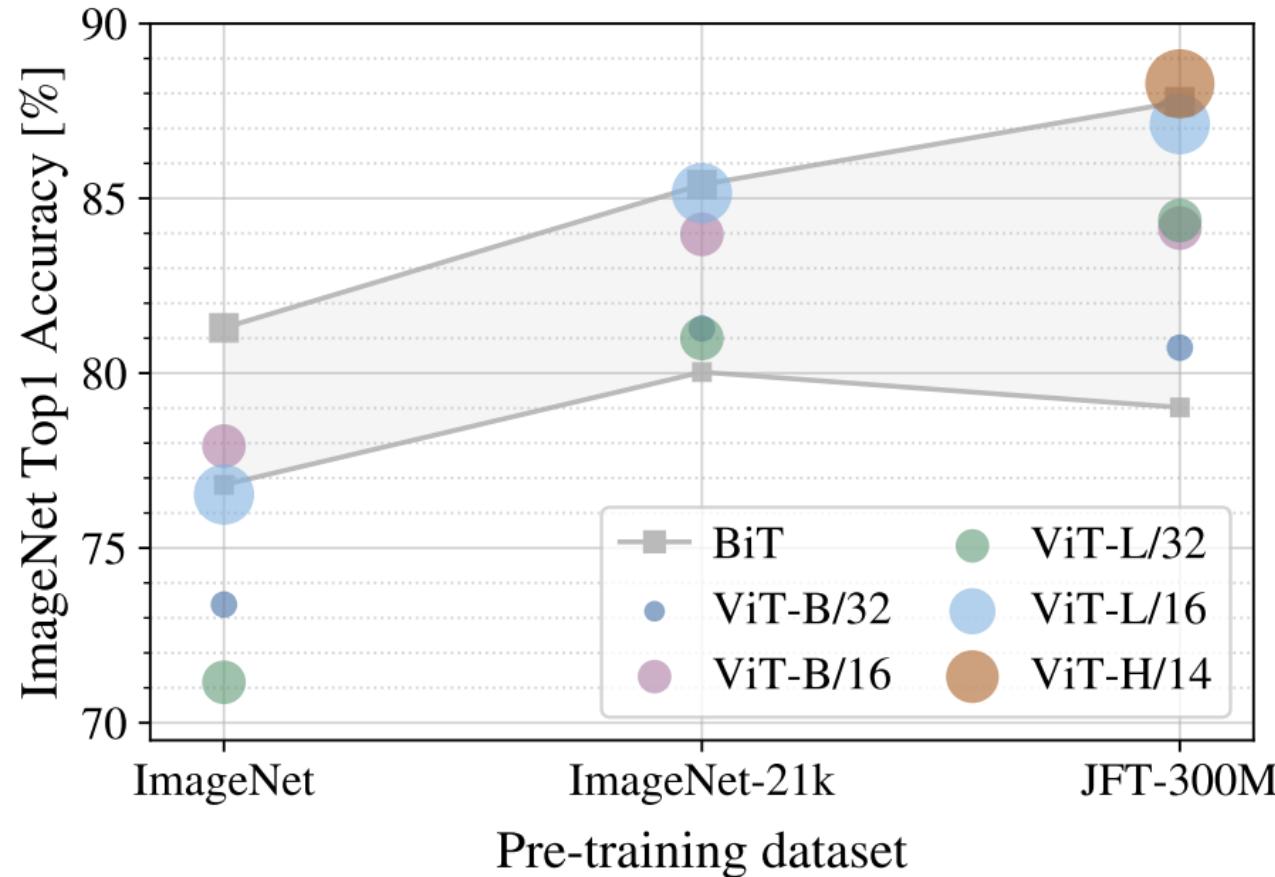
Beyond convolutional networks?

- Transformers for images



Beyond convolutional networks?

- Transformers for images



BiT: ResNet (Big Transfer)

ViT: Vision Transformer (Base/Large/Huge)

JFT-300M: [internal Google dataset](#) (not public)

How can we move forward?

- Develop the next generation of architectures
- Go beyond classification
 - “Rich” prediction tasks

“Rich” prediction tasks

Class label to image



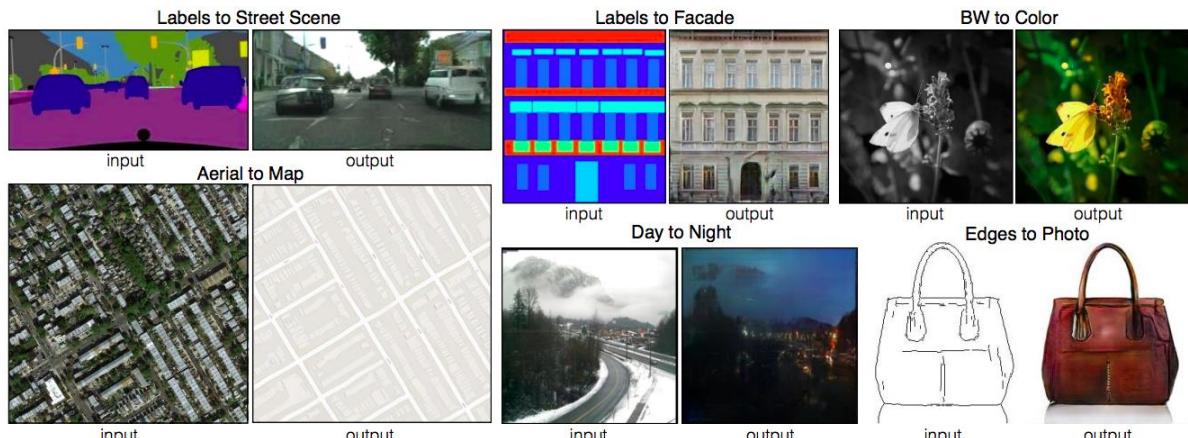
[Zhang et al. \(2019\)](#)

Text to image



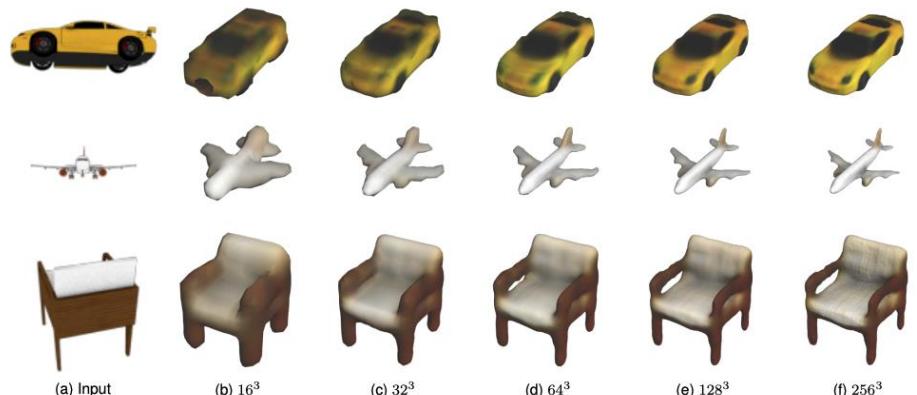
[Qiao et al. \(2019\)](#)

Image to image



[Isola et al. \(2017\)](#)

Image to 3D



[Hane et al. \(2019\)](#)

Learning 3D structure

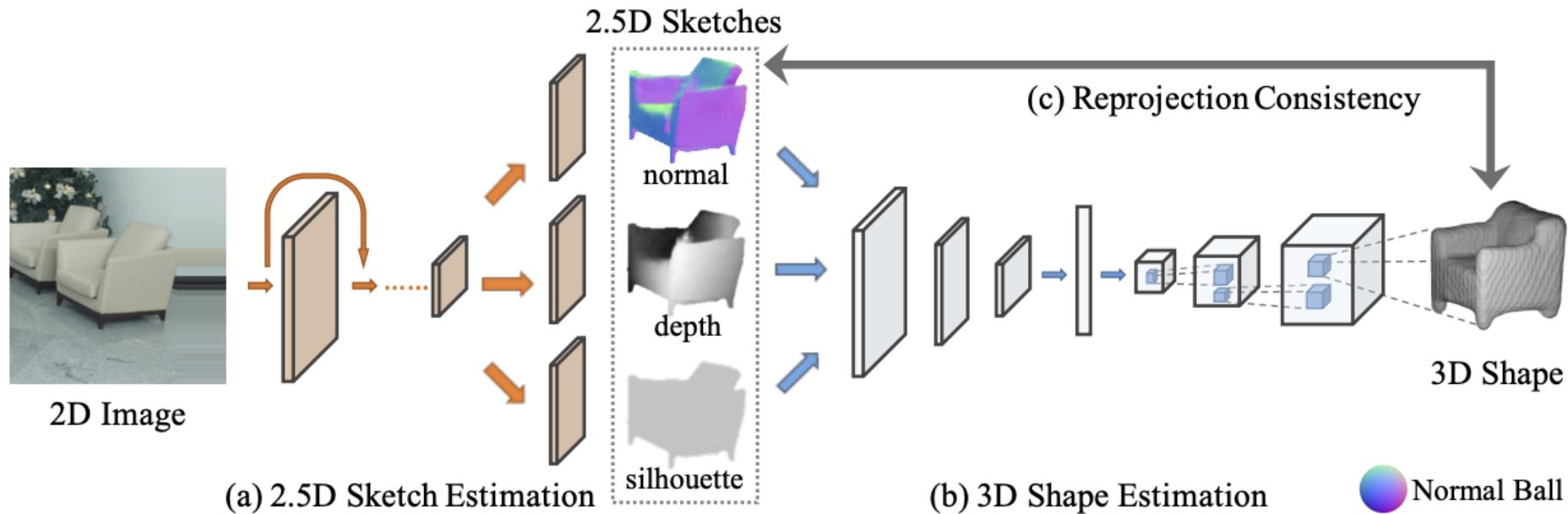
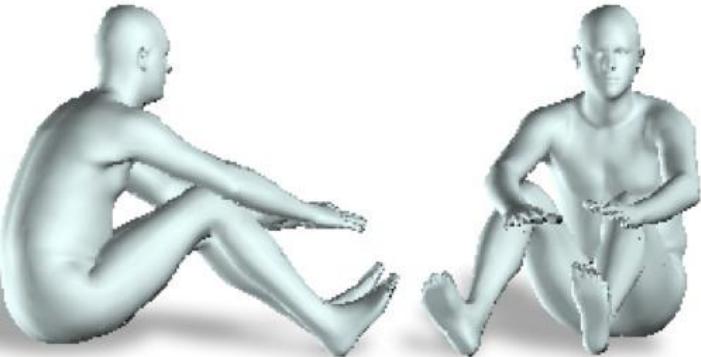
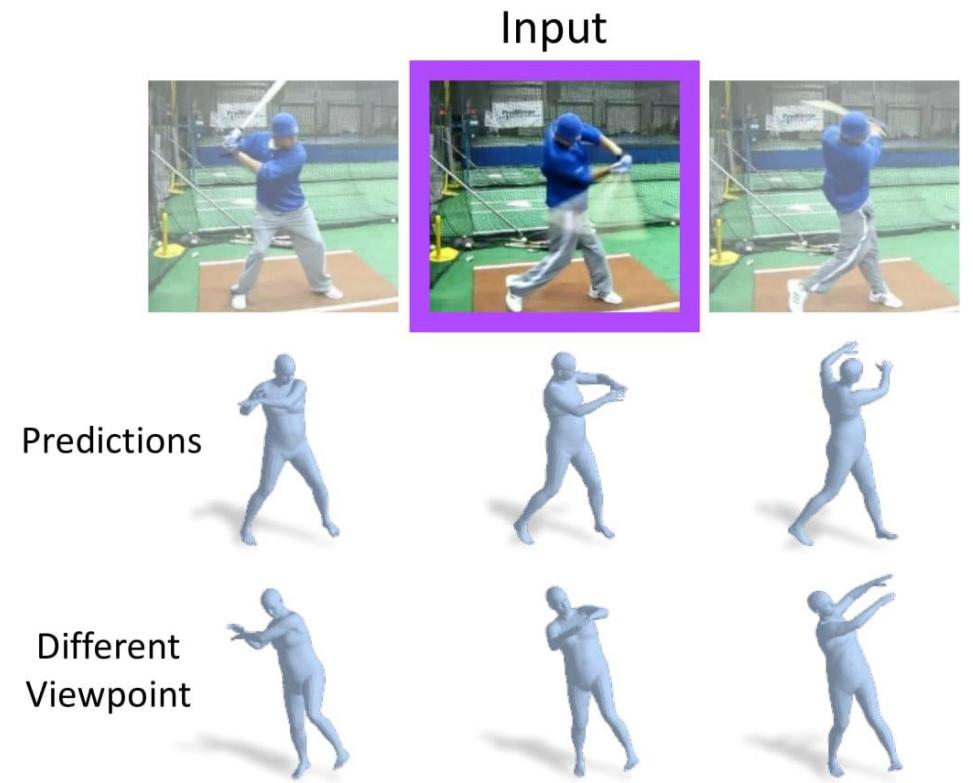


Figure 2: Our model (MarrNet) has three major components: (a) 2.5D sketch estimation, (b) 3D shape estimation, and (c) a loss function for reprojection consistency. MarrNet first recovers object normal, depth, and silhouette images from an RGB image. It then regresses the 3D shape from the 2.5D sketches. In both steps, it uses an encoding-decoding network. It finally employs a reprojection consistency loss to ensure the estimated 3D shape aligns with the 2.5D sketches. The entire framework can be trained end-to-end.

Learning 3D structure



F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero and M. Black,
[Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape
from a Single Image](#), ECCV 2016



A. Kanazawa, J. Zhang, P. Felsen, J. Malik,
[Learning 3D Human Dynamics from Video](#),
CVPR 2019

Learning skills from video



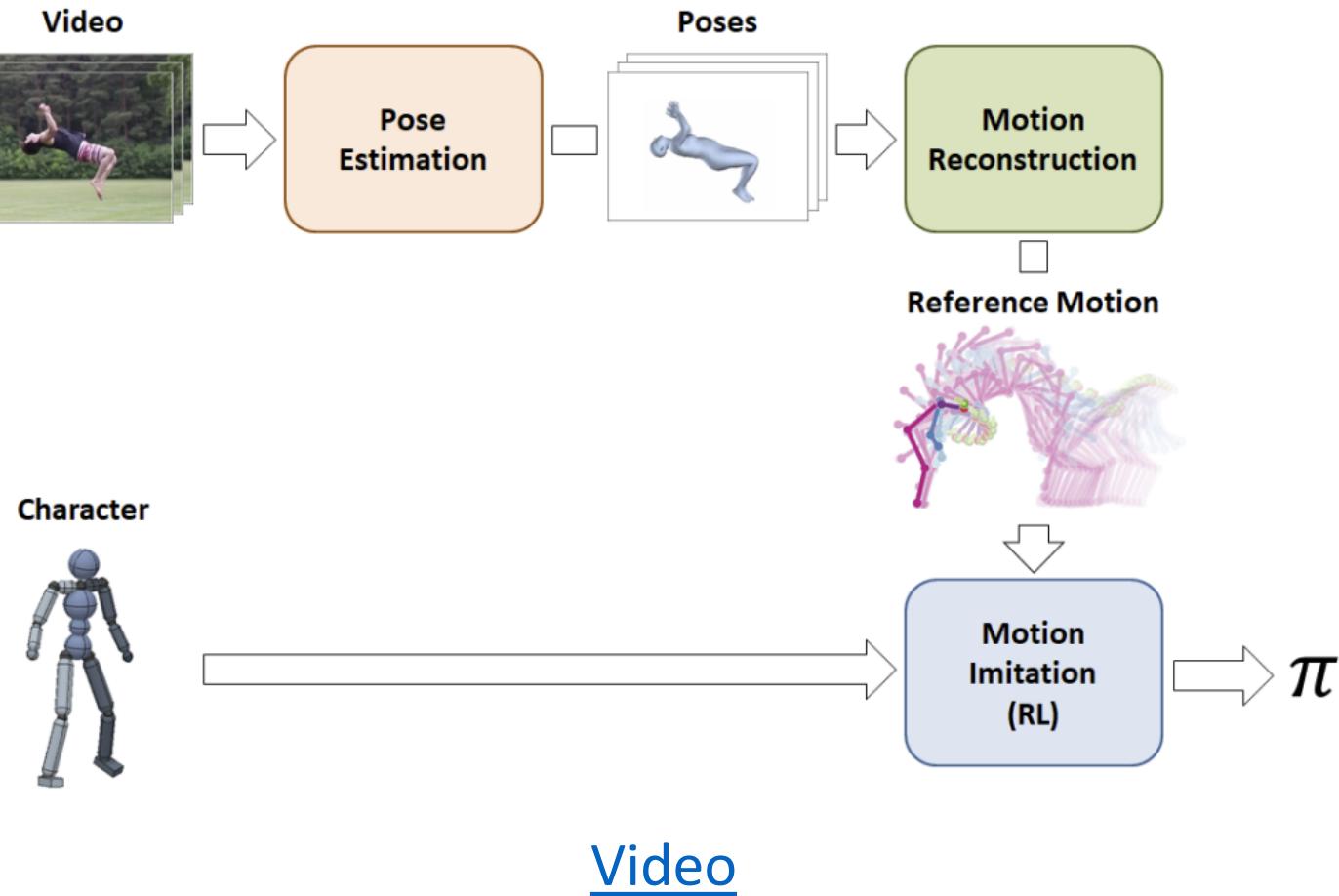
Fig. 1. Simulated characters performing highly dynamic skills learned by imitating video clips of human demonstrations. **Left:** Humanoid performing cartwheel B on irregular terrain. **Right:** Backflip A retargeted to a simulated Atlas robot.

[Video](#)

X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, S. Levine, [SFV: Reinforcement Learning of Physical Skills from Videos](#),

SIGGRAPH Asia 2018

Learning skills from video



X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, S. Levine, [SFV: Reinforcement Learning of Physical Skills from Videos](#), SIGGRAPH Asia 2018

How can we move forward?

- Develop the next generation of architectures
- Go beyond classification
 - “Rich” prediction tasks
 - Generation

Generation

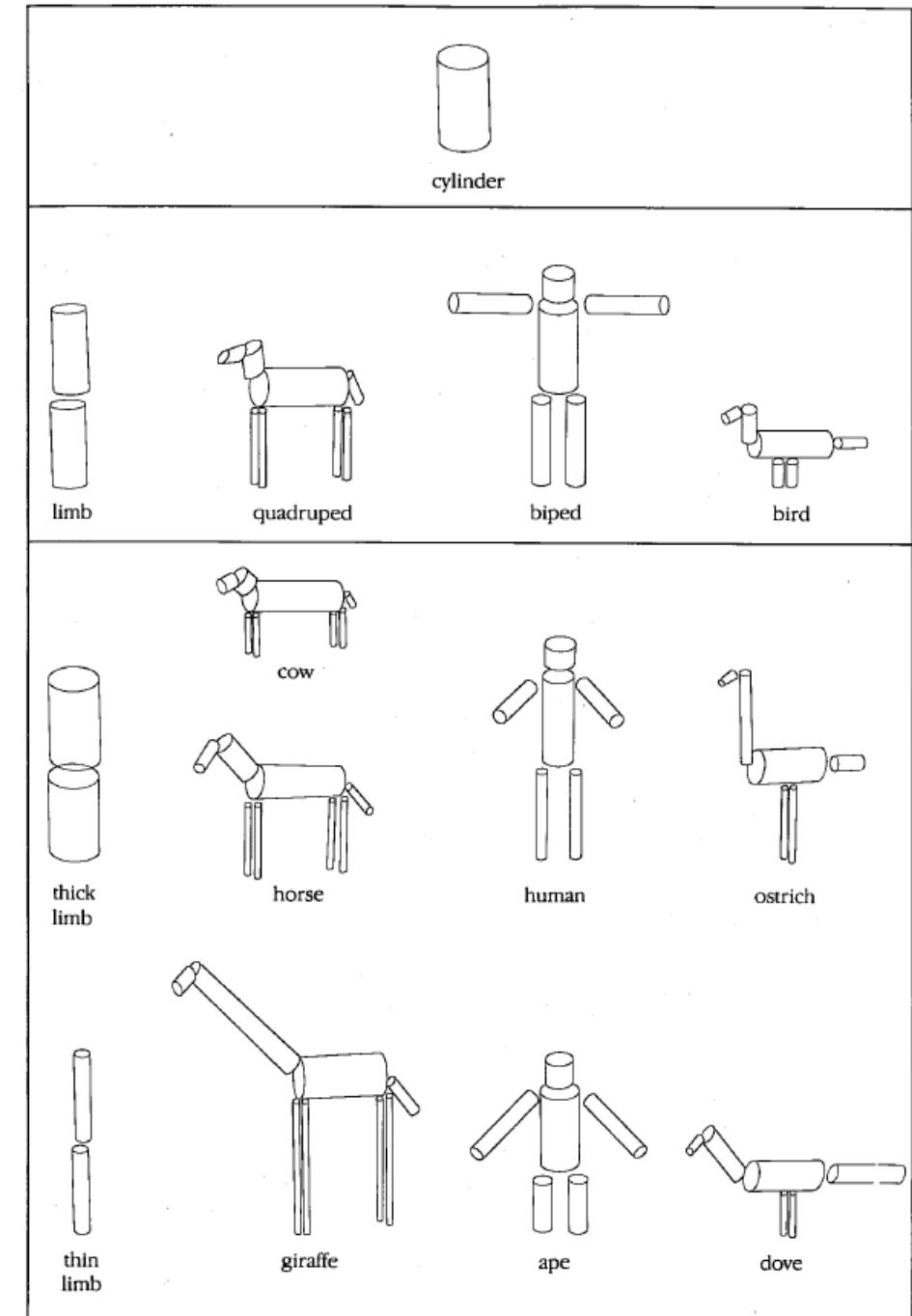
- State of the art: StyleGAN



T. Karras, S. Laine, T. Aila. [A Style-Based Generator Architecture for Generative Adversarial Networks](#). CVPR 2019

Generation

- May shed light on important open questions, such as:
 - Do we need explicitly compositional (part-based) object representations?



Source: D. Marr, Vision, 1982

Generation

- May shed light on important open questions, such as:
 - Do we need explicitly compositional (part-based) object representations?
- BigGAN proves that we don't?



A. Brock, J. Donahue, K. Simonyan, [Large scale GAN training for high fidelity natural image synthesis](#), ICLR 2019

Generation

- May shed light on important open questions, such as:
 - Do we need explicitly compositional (part-based) object representations?
- Or maybe we do?



A. Brock, J. Donahue, K. Simonyan, [Large scale GAN training for high fidelity natural image synthesis](#), ICLR 2019

Generation

- May shed light on important open questions, such as:
 - Do we need explicitly compositional (part-based) object representations?

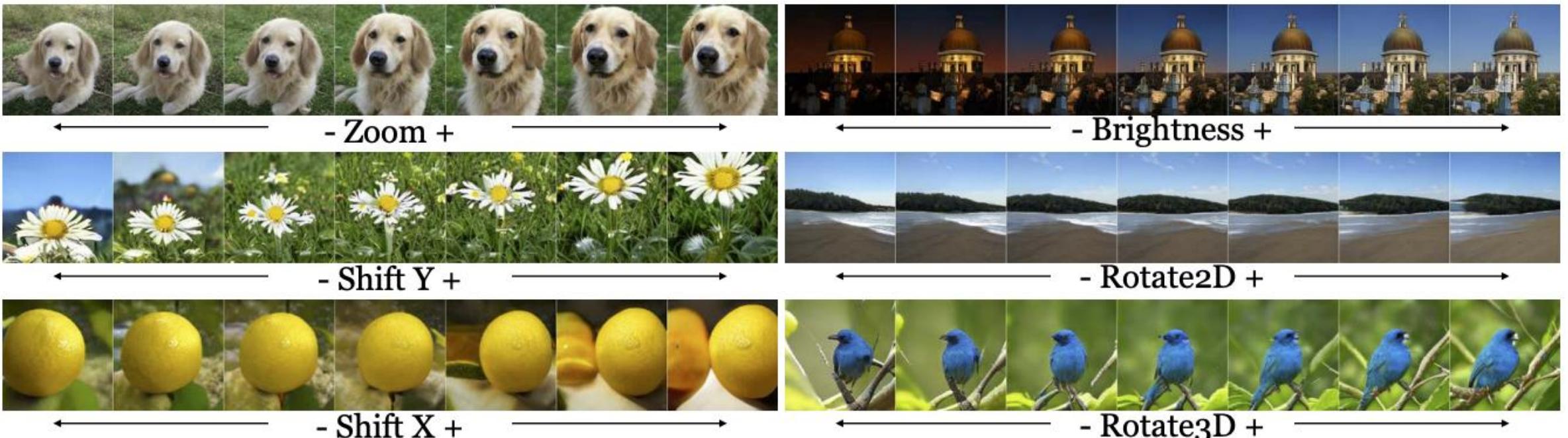


S. Azadi, D. Pathak, S. Ebrahimi, T. Darrell, [Compositional GAN: Learning Conditional Image Composition](#), arXiv 2019

Generation

- May shed light on important open questions, such as:
 - Do we need explicit 3D object representations?

Maybe we don't?



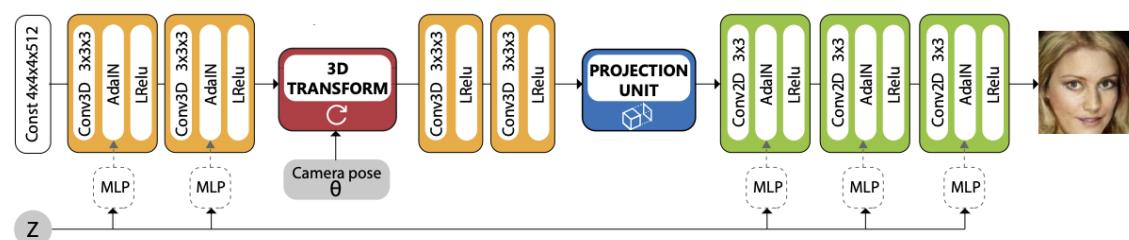
Generation

- May shed light on important open questions, such as:
 - Do we need explicit 3D object representations?

Or maybe we do?



Figure 1. HoloGAN learns to separate pose from identity (shape and appearance) only from unlabelled 2D images without sacrificing the visual fidelity of the generated images. All results shown here are sampled from HoloGAN for the same identities in each row but in different poses.

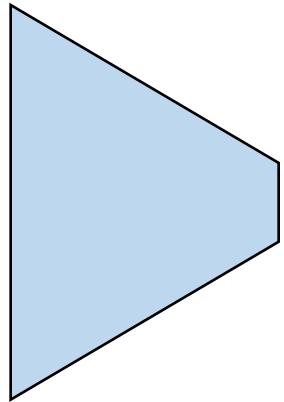


Possible ways forward

- Develop the next generation of architectures
- Go beyond classification
 - “Rich” prediction tasks
 - Generation
- Work on integrating discriminative and generative models

Integrating discriminative and generative models?

Discriminative model

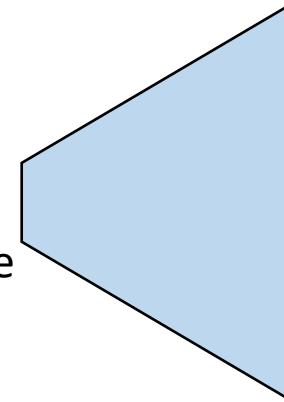


Label

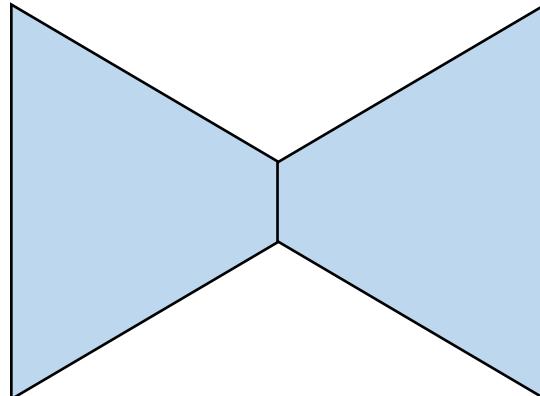
Generative model



Noise/
latent
variable

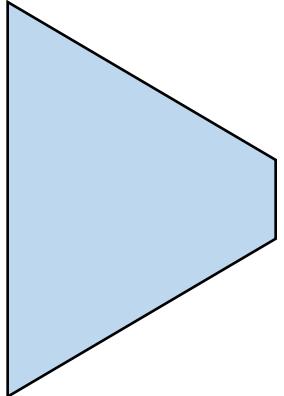


Encoder-decoder architecture



Integrating discriminative and generative models?

Discriminative model

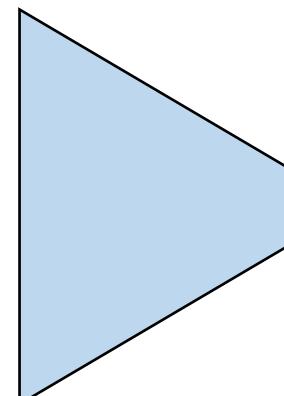
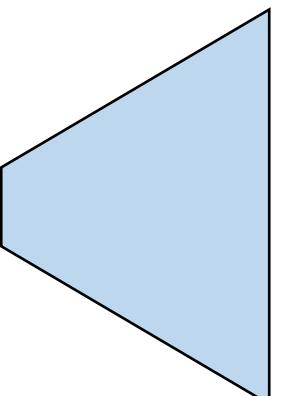


Generative model



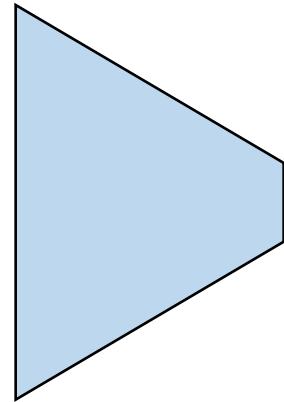
Noise/
latent
variable

GAN



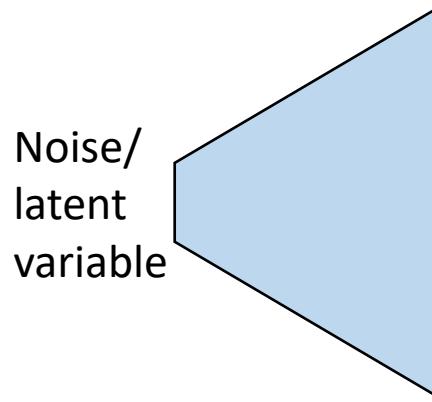
Integrating discriminative and generative models?

Discriminative model



Label

Generative model



Noise/
latent
variable



What else is there?

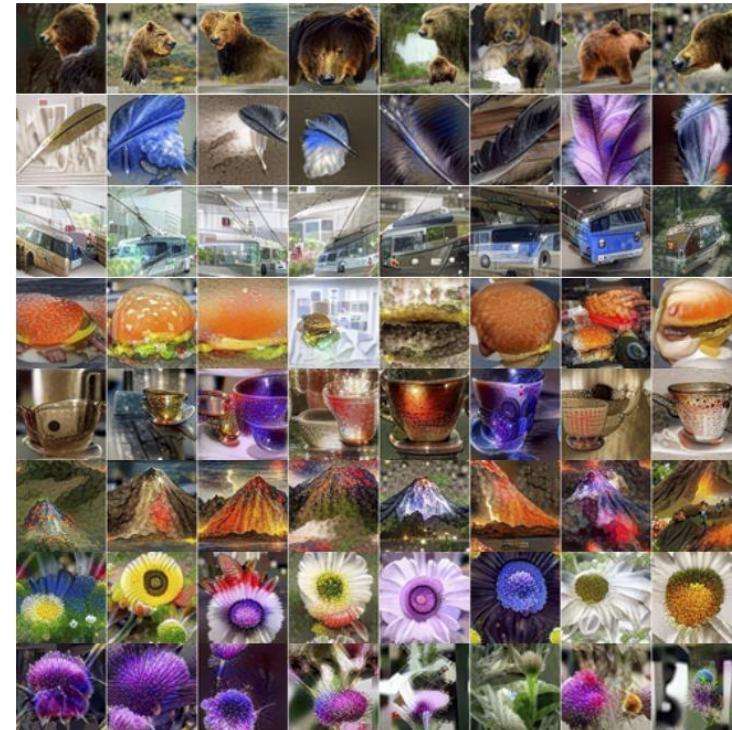
Integrating discriminative and generative models?

- Can “hallucination” help with recognition?
 - Low-shot learning, incremental learning, transfer learning



Figure 1. Given a single image of a novel visual concept, such as a blue heron, a person can visualize what the heron would look like in other poses and different surroundings. If computer recognition systems could do such hallucination, they might be able to learn novel visual concepts from less data.

[Wang et al. \(2018\)](#)



High-quality images synthesized by “inverting” an ImageNet-pretrained network

[Yin et al. \(2019\)](#)

Integrating discriminative and generative models?

- Can attribute-based manipulation of training examples help to train more accurate or less biased models?

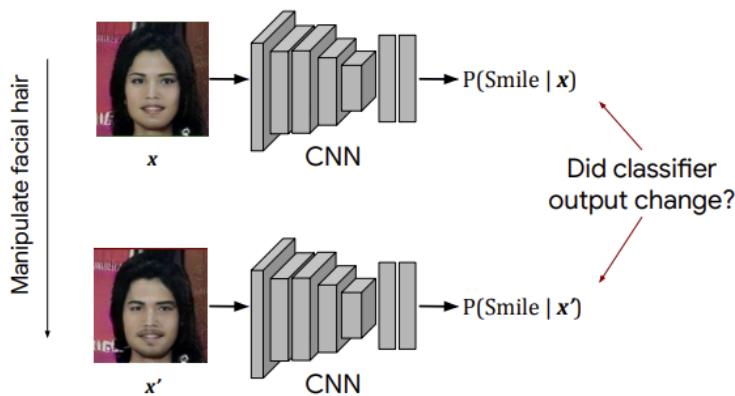


Figure 1: Counterfactual attribute sensitivity measures the effect of manipulating a specific property of an image on the output of a trained classifier. In this example, we consider the effect of adding facial hair on the output of a smiling classifier. If the classifier's output systematically changes as a result of adding or removing facial hair then a potentially undesirable bias has been detected since, all else being equal, facial hair should be irrelevant to the classification task.



Figure 1: Our approach learns to *actively create* training images for fine-grained attribute comparisons, focusing attention on informative pairs of images unavailable in the real image training pool. Here we show two example pairs it generated to improve learning of *open* (left) and *masculine* (right).

[Yu and Grauman \(2019\)](#)

Possible ways forward

- Develop the next generation of architectures
- Go beyond classification
 - “Rich” prediction tasks
 - Generation
- Work on integrating discriminative and generative models
- Move away from full supervision

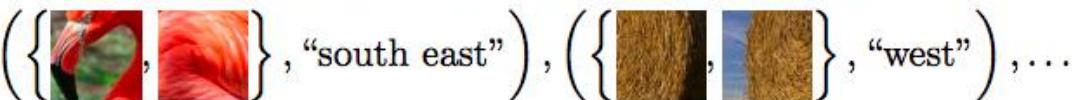
Self-supervised learning

- For still image recognition tasks, self-supervision is more cumbersome (so far) as getting enough supervised training data
- However, it is very attractive (and possibly unavoidable) for video, audio, and sensorimotor learning

Ex. 1: **Inpainting** (remove patch and then predict it)



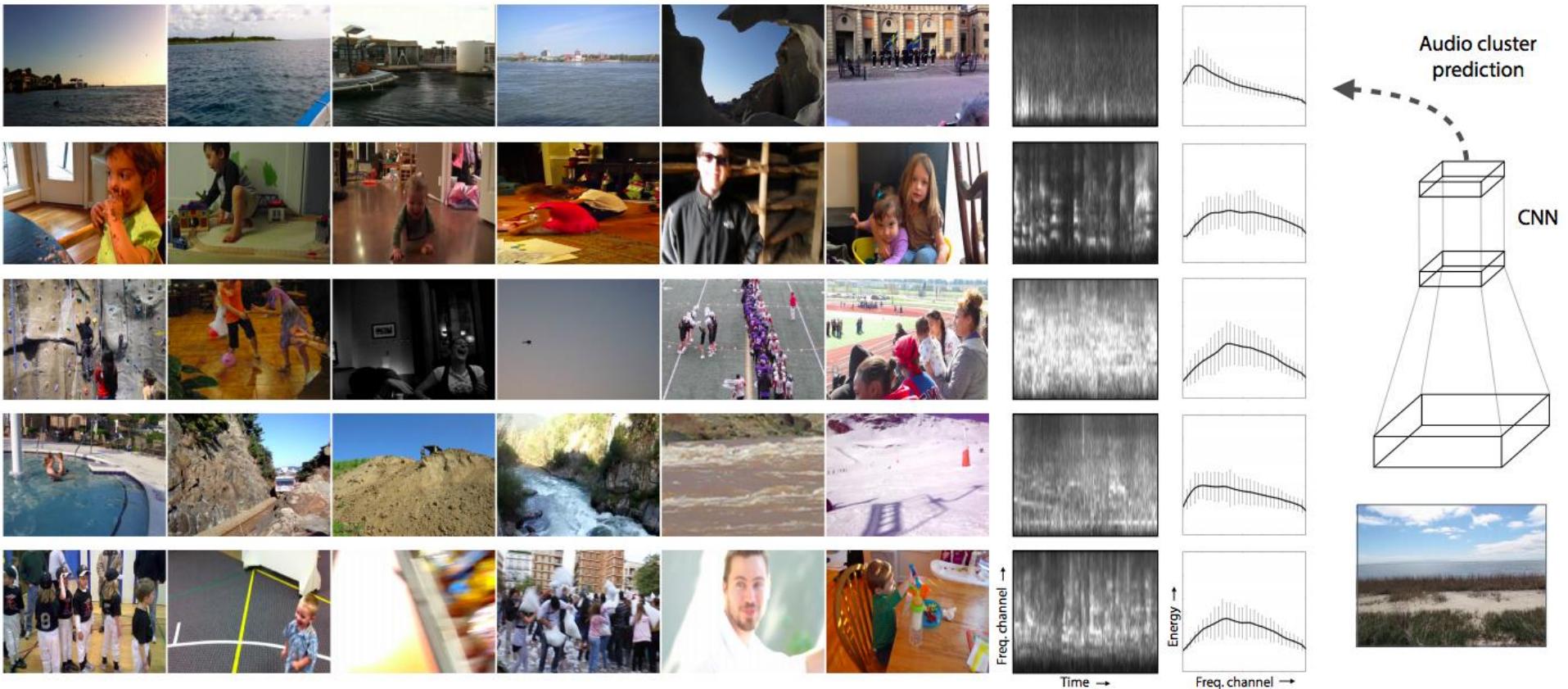
Ex. 2: **Context** (given two patches, predict their spatial relation)



Ex. 3: **Colorization** (predict color given intensity)



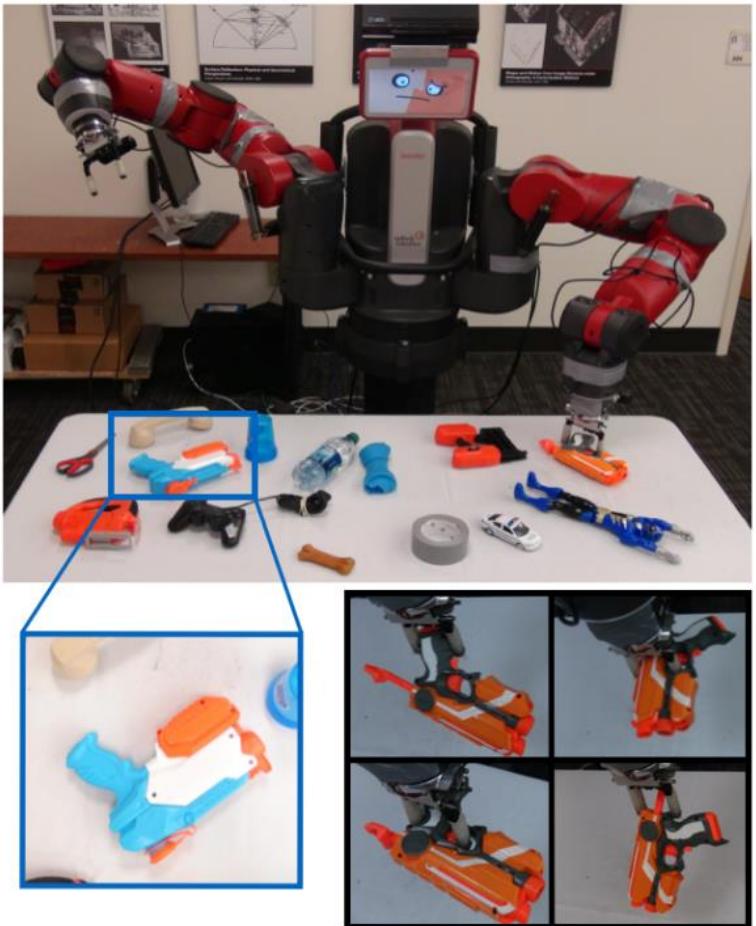
Cross-modal self-supervision



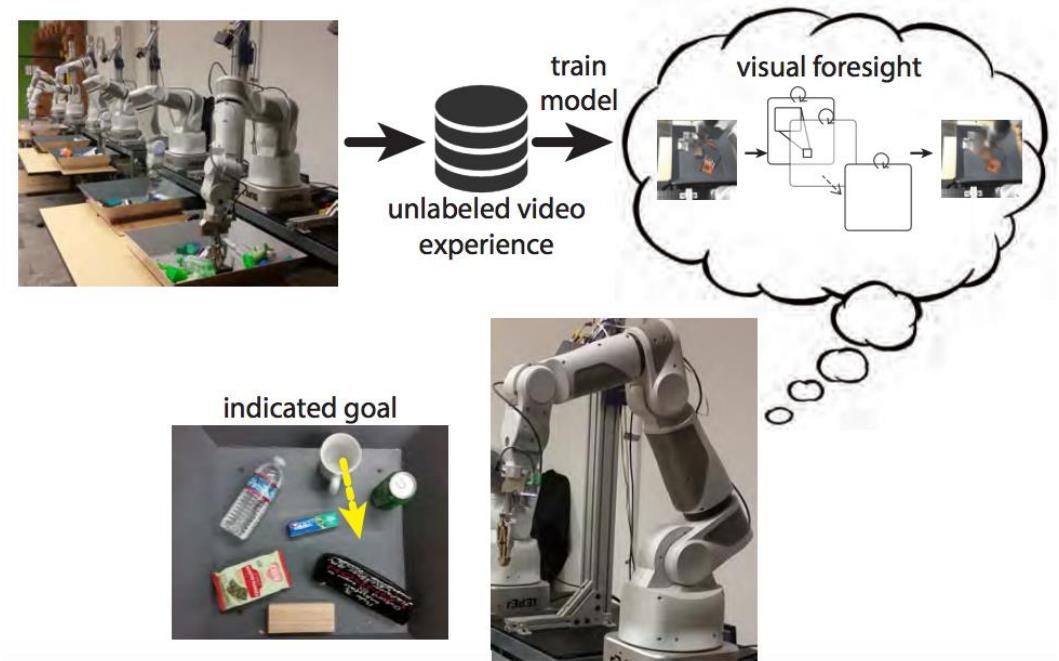
(a) Images grouped by audio cluster

(b) Clustered audio stats. (c) CNN model

Self-supervision from future prediction



L. Pinto and A. Gupta, [Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours](#), ICRA 2016



C. Finn and S. Levine. [Deep Visual Foresight for Planning Robot Motion](#). ICRA 2017

Self-supervision from future prediction

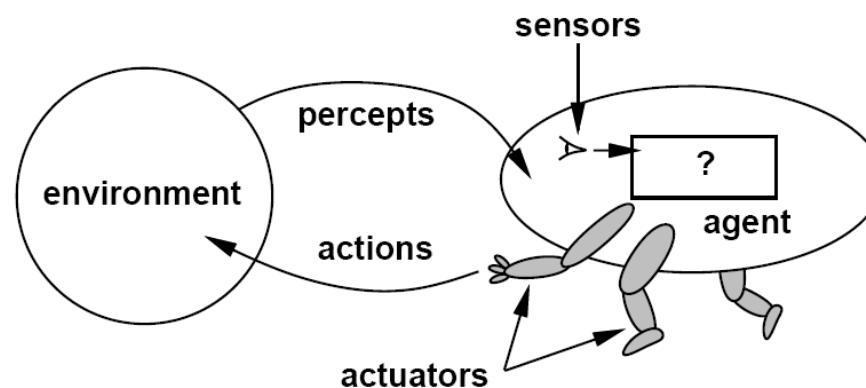
- Curiosity is also about predicting the effects of one's actions!



[Video](#)

Possible ways forward

- Develop the next generation of architectures
- Go beyond classification
 - “Rich” prediction tasks
 - Generation
- Work on integrating discriminative and generative models
- Move away from full supervision
- Focus on embodied vision, sensorimotor learning



Outline

- Vision
- Embodied learning (RL, robotics)

Embodied platforms

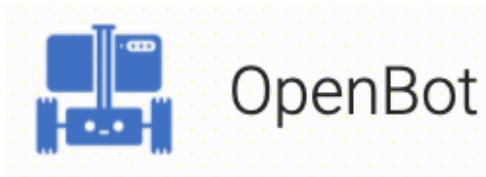
- Simulation: [AI2Thor](#), [Habitat](#)



- Real robots: [PyRobot](#)



- Robot on your smartphone: [OpenBot](#)



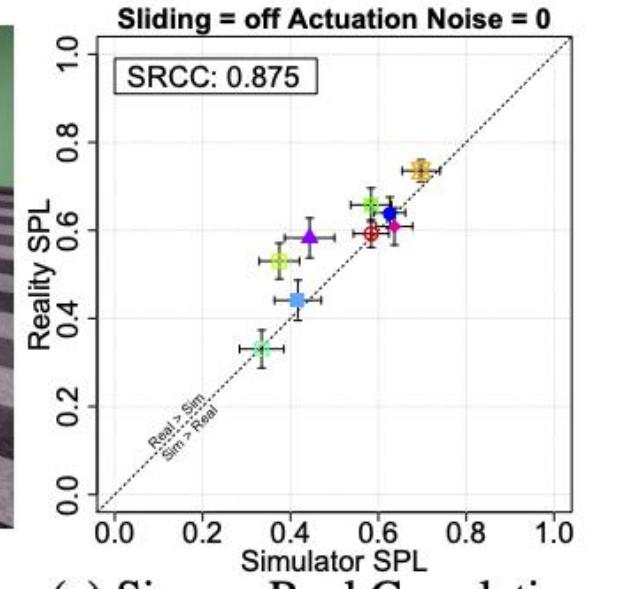
Can we trust simulators?



(a) Reality.



(b) Simulation.



(c) Sim-vs-Real Correlation.

A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, D. Batra, [Are We Making Real Progress in Simulated Environments? Measuring the Sim2Real Gap in Embodied Visual Navigation](#), arXiv 2019

Can we scale up robot learning?



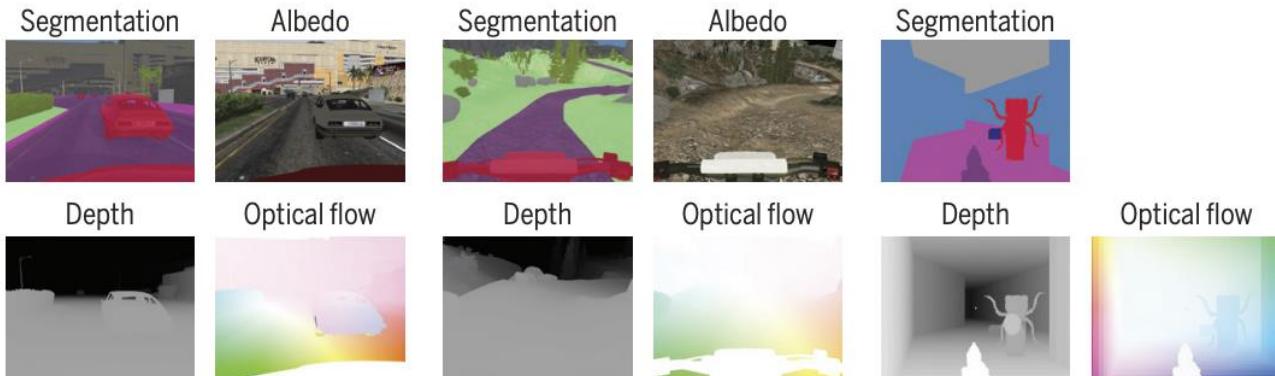
Figure 1. Our large-scale data collection setup, consisting of 14 robotic manipulators. We collected over 800,000 grasp attempts to train the CNN grasp prediction model.

What internal representations are needed for embodied tasks?

A

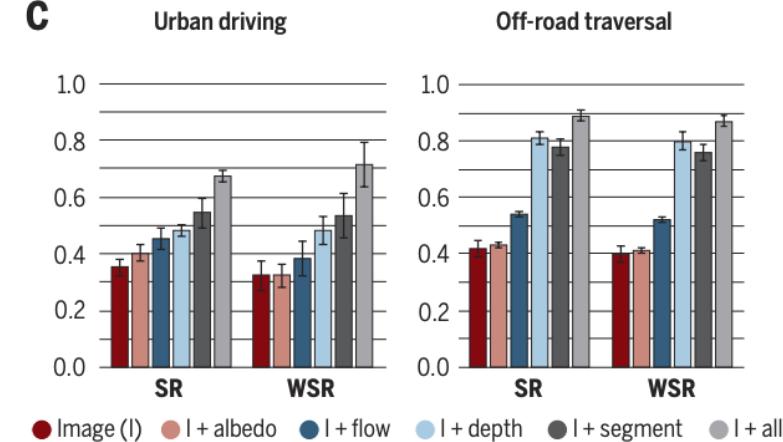


B

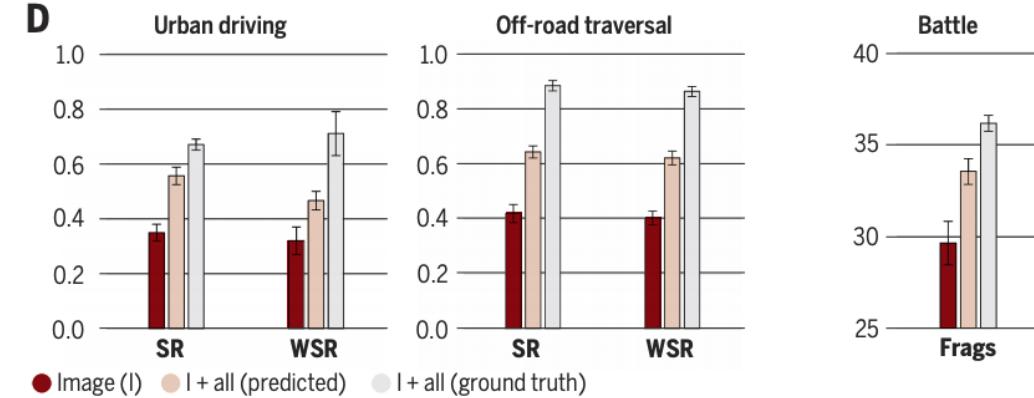


"Our main finding is that computer vision does matter. Models equipped with intermediate representations train faster, achieve higher task performance, and generalize better to previously unseen environments."

C



D

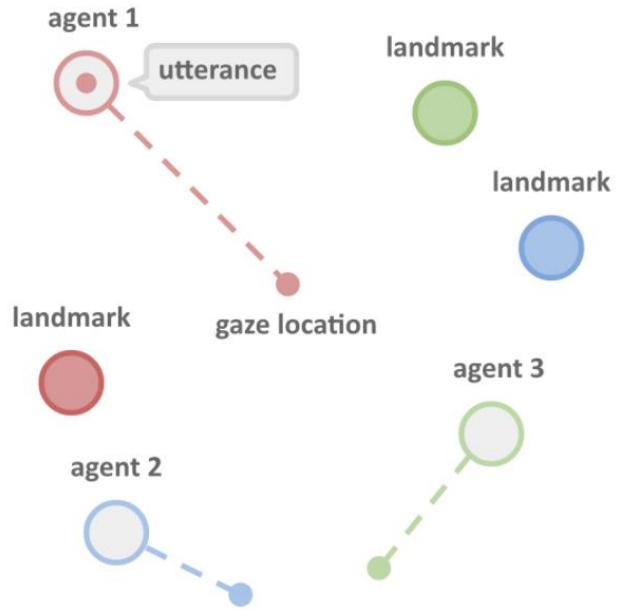


Multiple agents and communication

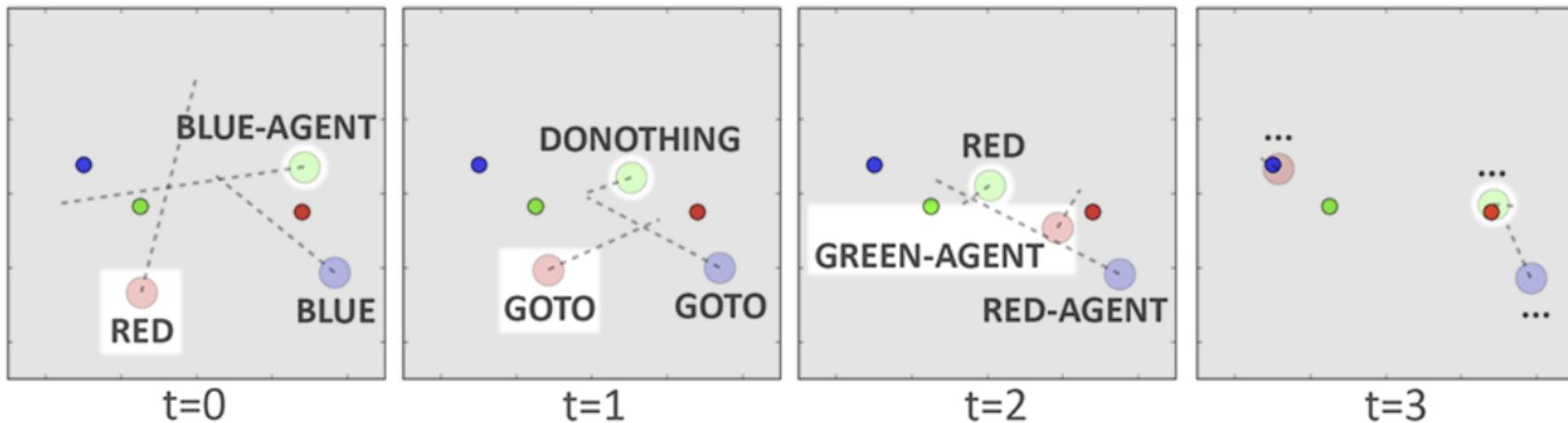


[Source](#)

Emergence of language



Our agents exist in a simple, 2D world, and are able to take actions such as moving to locations, looking at things, or saying things to communicate with other agents. In this picture, agent 1 is saying something while staring at a point in the center of the map.



In the above step-by-step run, at t=0 the red agent says a word corresponding to the red landmark (center right), then at t=1 says a word that is equivalent to 'Goto', then in t=2 says 'green-agent'. The green-agent hears its instructions and immediately moves to the red landmark.

<https://openai.com/blog/learning-to-communicate/>
[Video](#)

RL for negotiation

Divide these objects between you and another Turker. Try hard to get as many points as you can!

Send a message now, or enter the agreed deal!

Items	Value	Number You Get
	8	<input type="button" value="1"/>
	1	<input type="button" value="1"/>
	0	<input type="button" value="0"/>

Mark Deal Agreed



The screenshot shows a negotiation dialogue between two parties: 'Fellow Turker' and 'You'. The interface includes a list of items with their values and the number the user gets, a 'Mark Deal Agreed' button, and a message input field.

Fellow Turker: I'd like all the balls

You: Ok, if I get everything else

Fellow Turker: If I get the book then you have a deal

You: No way - you can have one hat and all the balls

Fellow Turker: Ok deal

Type Message Here:

Message

Figure 1: A dialogue in our Mechanical Turk interface, which we used to collect a negotiation dataset.

Outline

- Vision
- Embodied learning (RL, robotics)
- Language

Large-scale language models

Model Alias	Org.	Article Reference
ULMfit	fast.ai	<i>Universal Language Model Fine-tuning for Text Classification</i> Howard and Ruder
 ELMo	AllenNLP	<i>Deep contextualized word representations</i> Peters et al.
OpenAI GPT	OpenAI	<i>Improving Language Understanding by Generative Pre-Training</i> Radford et al.
 BERT	Google	<i>BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding</i> Devlin et al.
XLM	Facebook	<i>Cross-lingual Language Model Pretraining</i> Lample and Conneau

[Image source](#)

Should we be scared of GPT-3?

Opinion

The New York Times

How Do You Know a Human Wrote This?

Machines are gaining the ability to write, and they are getting terrifyingly good at it.



By **Farhad Manjoo**
Opinion Columnist

July 29, 2020

<https://www.nytimes.com/2020/07/29/opinion/gpt-3-ai-automation.html>

See also:

<https://www.nytimes.com/2020/11/24/science/artificial-intelligence-ai-gpt3.html>

MIT Technology Review

Opinion

GPT-3, Bloviator: OpenAI's language generator has no idea what it's talking about

Tests show that the popular AI still has a poor grasp of reality.

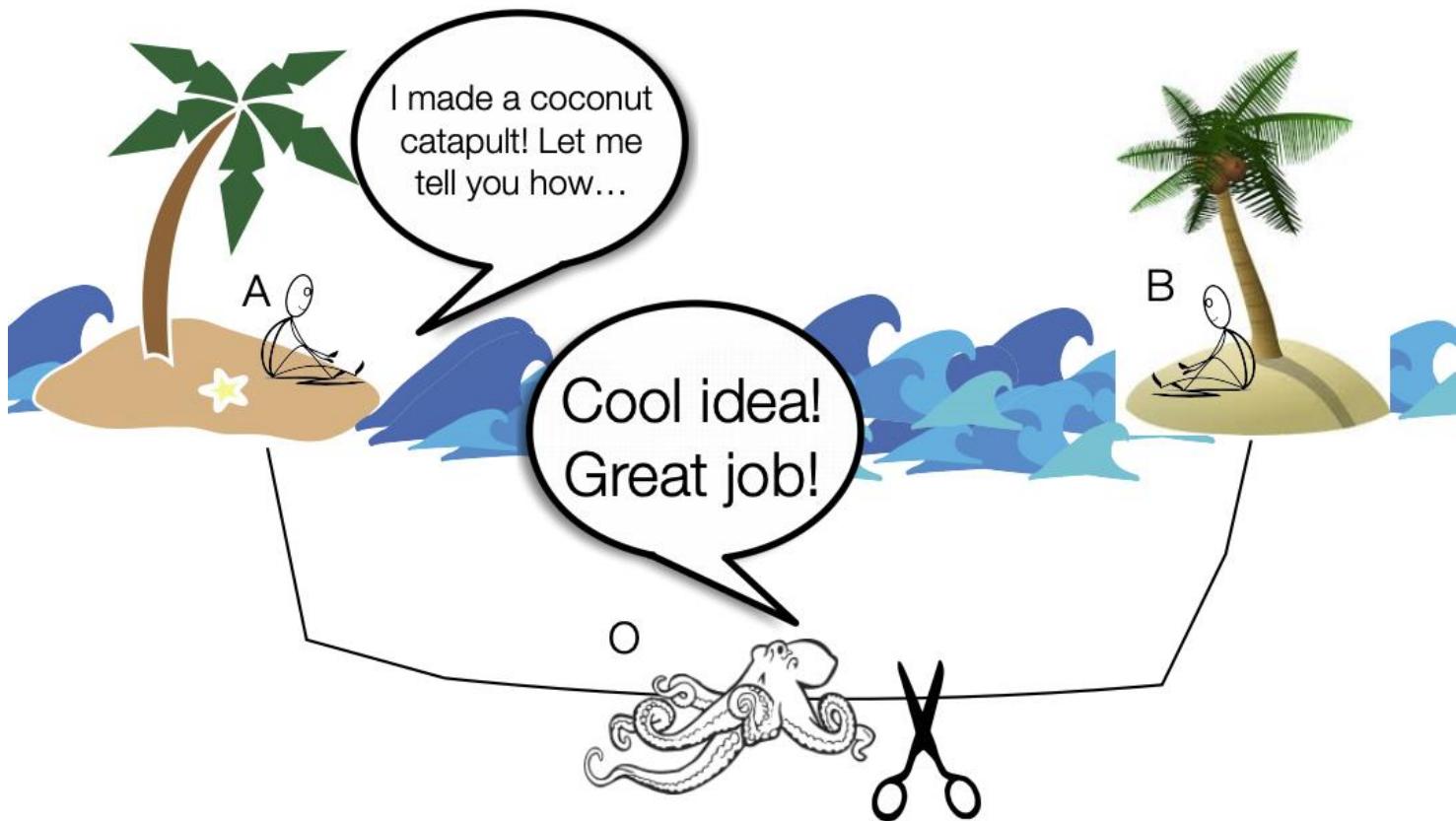
by **Gary Marcus** and **Ernest Davis**

August 22, 2020

<https://www.technologyreview.com/2020/08/22/1007539/gpt3-openai-language-generator-artificial-intelligence-ai-opinion/>

Language: The importance of grounding

- *Meaning* cannot be learned from *form* alone – it requires knowing about the relationship between language and the outside world



[Image source](#)

Further challenges

- Reasoning
- Memory
- Lifelong learning

Recent Trends

- Explainable Deep Learning
- Robust Deep Learning
- Light Weight Deep Learning (Network Pruning)
- Neural Architecture Search
- Few Shot, One Shot and Zero Shot Learning
- Incremental Learning
- Deep Learning in Multimedia
- Deep Learning in Other Domains
- And Many More

Parting thoughts

- The next breakthroughs are not likely to come cheaply
- Access to data, computation, and platforms will be key
- The next few years will make it clearer which problems have been truly solved and which ones have been underestimated
- The hard problems are getting into “AI-complete” territory

Acknowledgement

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More