CS 3710: Advanced Topics in AI

Introduction

Prof. Adriana Kovashka
University of Pittsburgh
August 28, 2023

Course Info

- **Course website:** http://people.cs.pitt.edu/~kovashka/cs3710_fa23
- **Instructor**: Adriana Kovashka (kovashka@cs.pitt.edu)
- **Office:** Sennott Square 5325
- **Class:** Mon/Wed, 9:30am-10:45am
- **Office hours**: by appointment
- **Zoom** (for some appointments)**:** https://pitt.zoom.us/s/4168010698

About the Instructor



Born 1985 in Sofia, Bulgaria



Got BA in 2008 at Pomona College, CA (Computer Science & Media Studies)



Got PhD in 2014 at University of Texas at Austin (Computer Vision)

Course Goals

- To learn about state-of-the-art approaches to computer vision tasks
- To think critically about vision approaches, see strengths, weaknesses, and connections between works
- To conduct research with contributions including novel methods, method comparison and method/data analysis

Note on Course Climate

- Some may be more familiar with the subject of the seminar, some less, and that’s ok!
- Please respect each other and listen to each other carefully
- You will need to collaborate on the course project–work hard, be fair and honest, and try to talk through problems
- Talk to instructor openly

Plan for First Three Classes

- Course logistics
- Students introduce themselves
- What is computer vision?
- Preview readings
- Paper presentation signups (by Sept. 1)
- Some basics
- Preliminary project pitches exercise

Policies and Schedule

https://people.cs.pitt.edu/~kovashka/cs3710_fa23/

Highlights:

- Paper presentations
- Project logistics
- Readings and schedule

Questions?

Blitz introductions (30 sec)

- What is your name?
- What one thing outside of school are you passionate about?
- What is your current or planned research about?
- What do you hope to get out of this class? (Optional)

- **When you speak, please remind me your name**

What is Computer Vision?

What is computer vision?

Done?

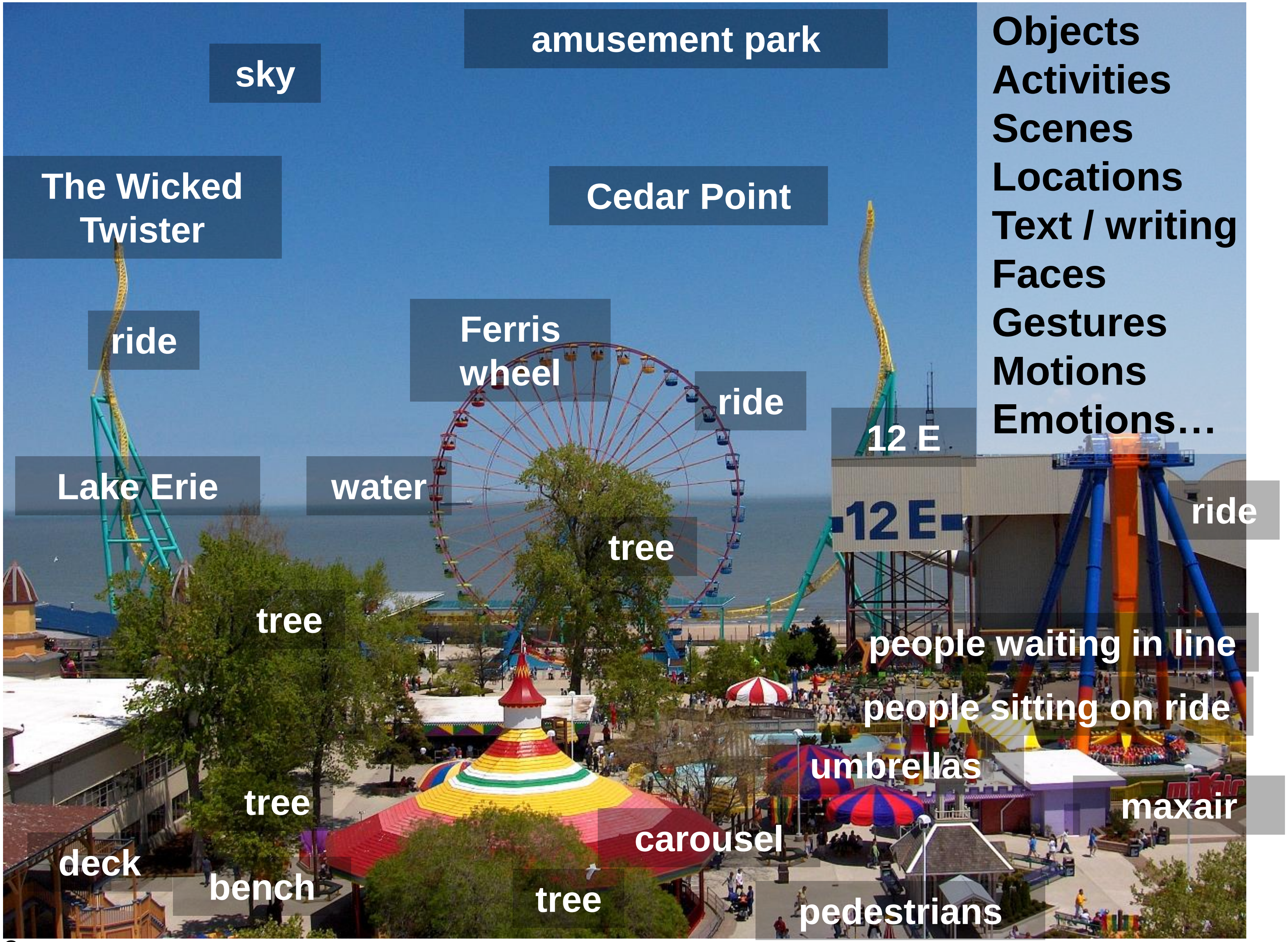
"We see with our brains, not with our eyes“ (Oliver Sacks and others)

What is computer vision?

- Automatic understanding of images and video
  - Algorithms and representations to allow a machine to recognize objects, people, scenes, and activities
  - Algorithms to mine, search, and interact with visual data
  - Computing properties and navigating within the 3D world using visual data
  - Generating realistic synthetic visual data

Adapted from Kristen Grauman

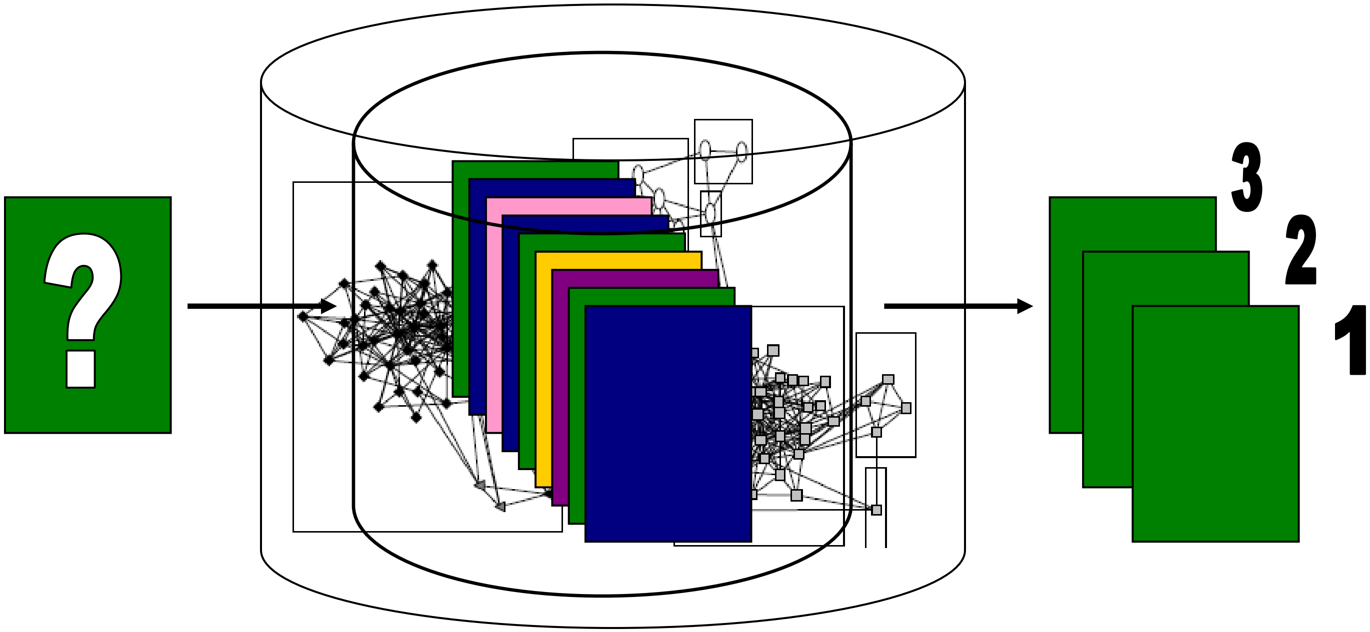
Perception and interpretation



- Objects
- Activities
- Scenes
- Locations
- Text / writing
- Faces
- Gestures
- Motions
- Emotions…

Kristen Grauman

Visual search, organization



Query **Image or video archives** **Relevant content**

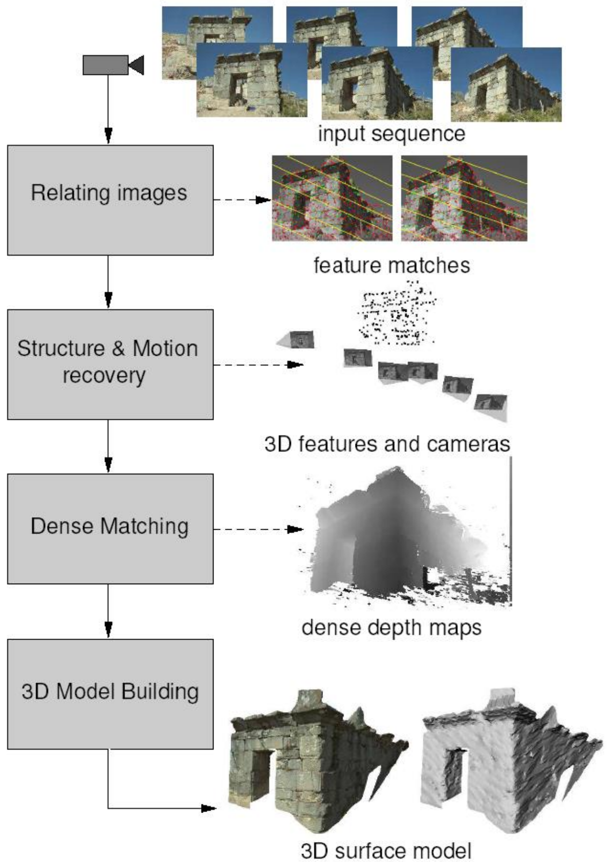
Kristen Grauman

Measurement

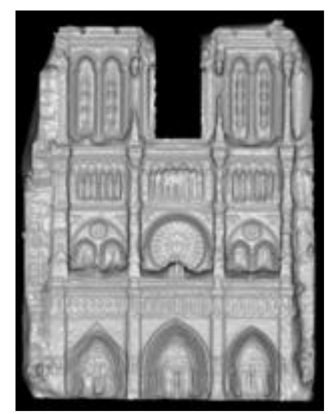
Real-time stereo

Pollefeys et al.

Structure from motion



Multi-view stereo for community photo collections



Goesele et al.

Slide credit: L. Lazebnik

Generation

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

Recognition in novel modalities

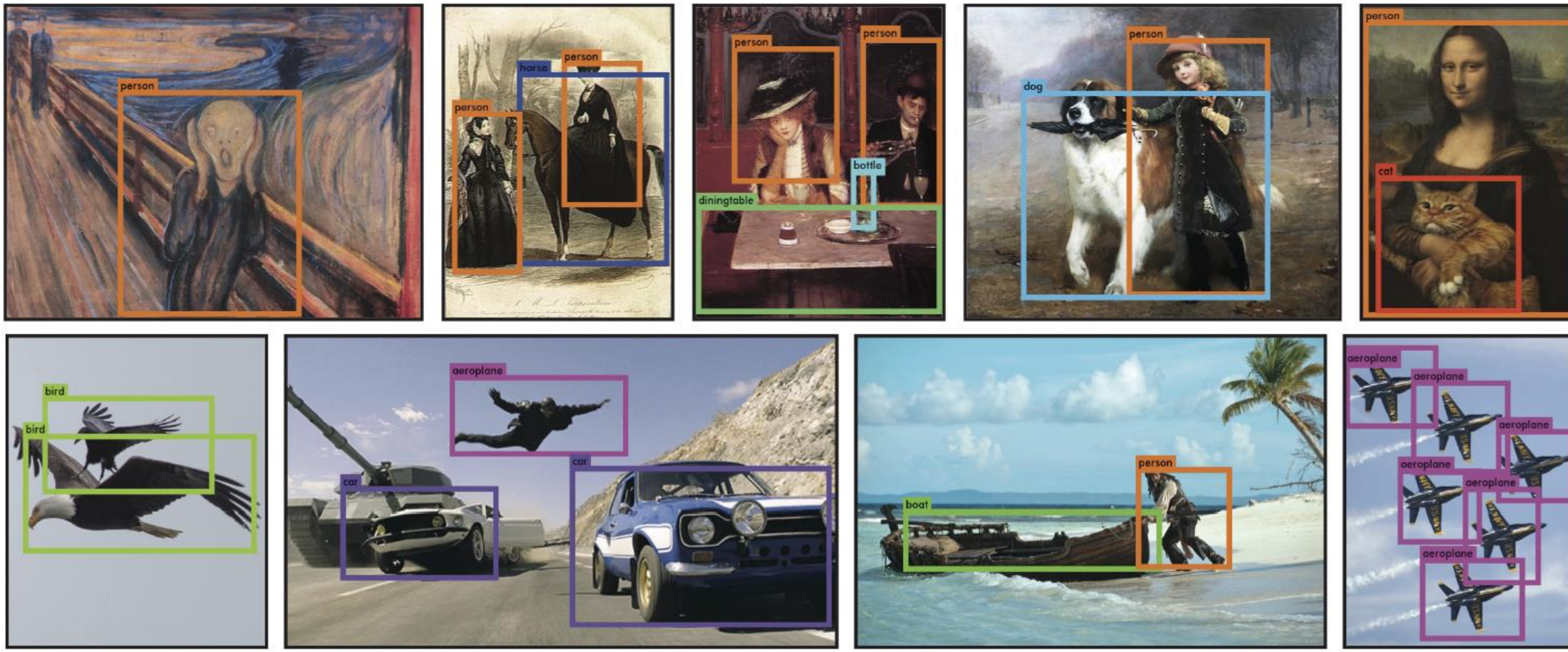
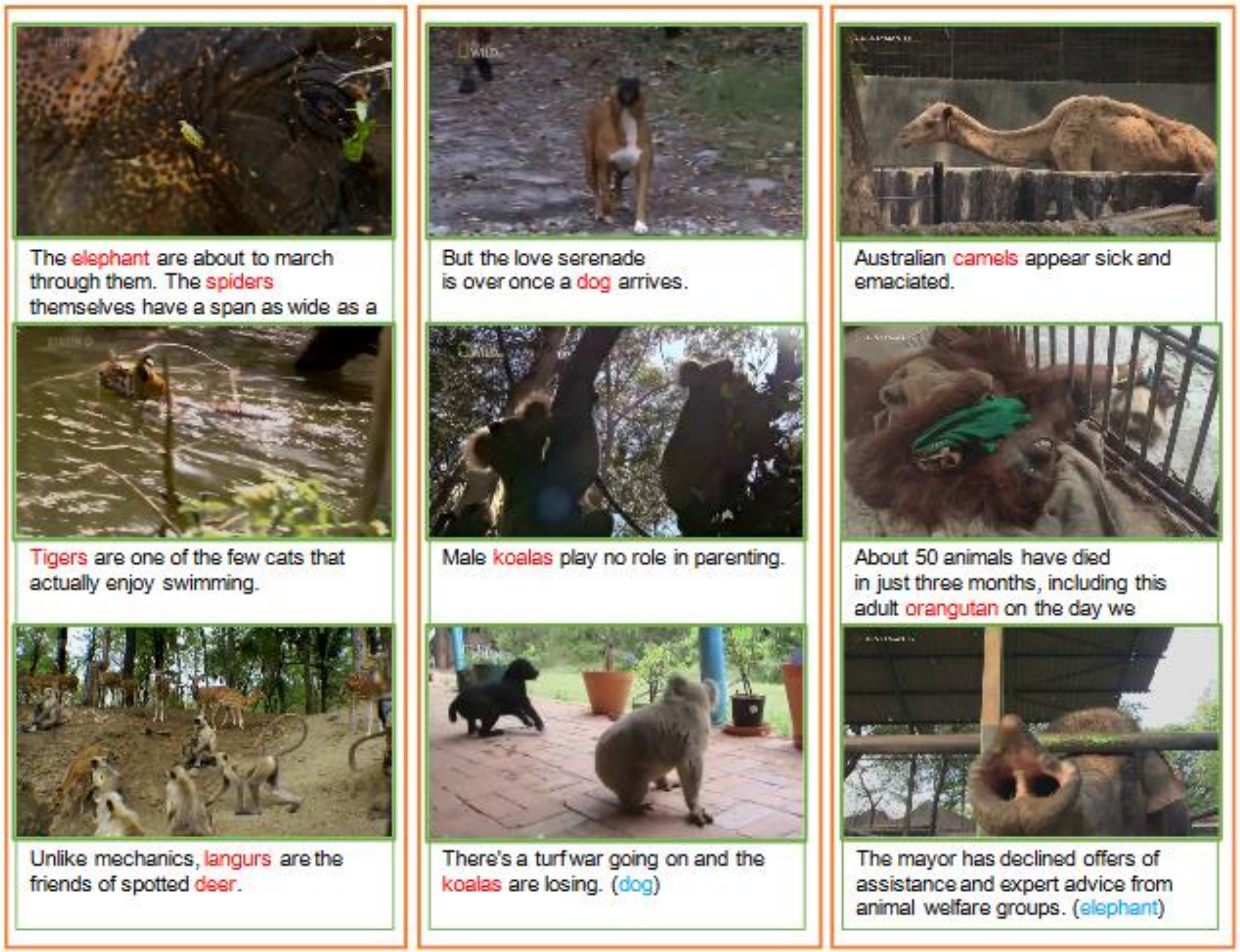


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

Redmon et al., CVPR 2016

Learning from weak supervision

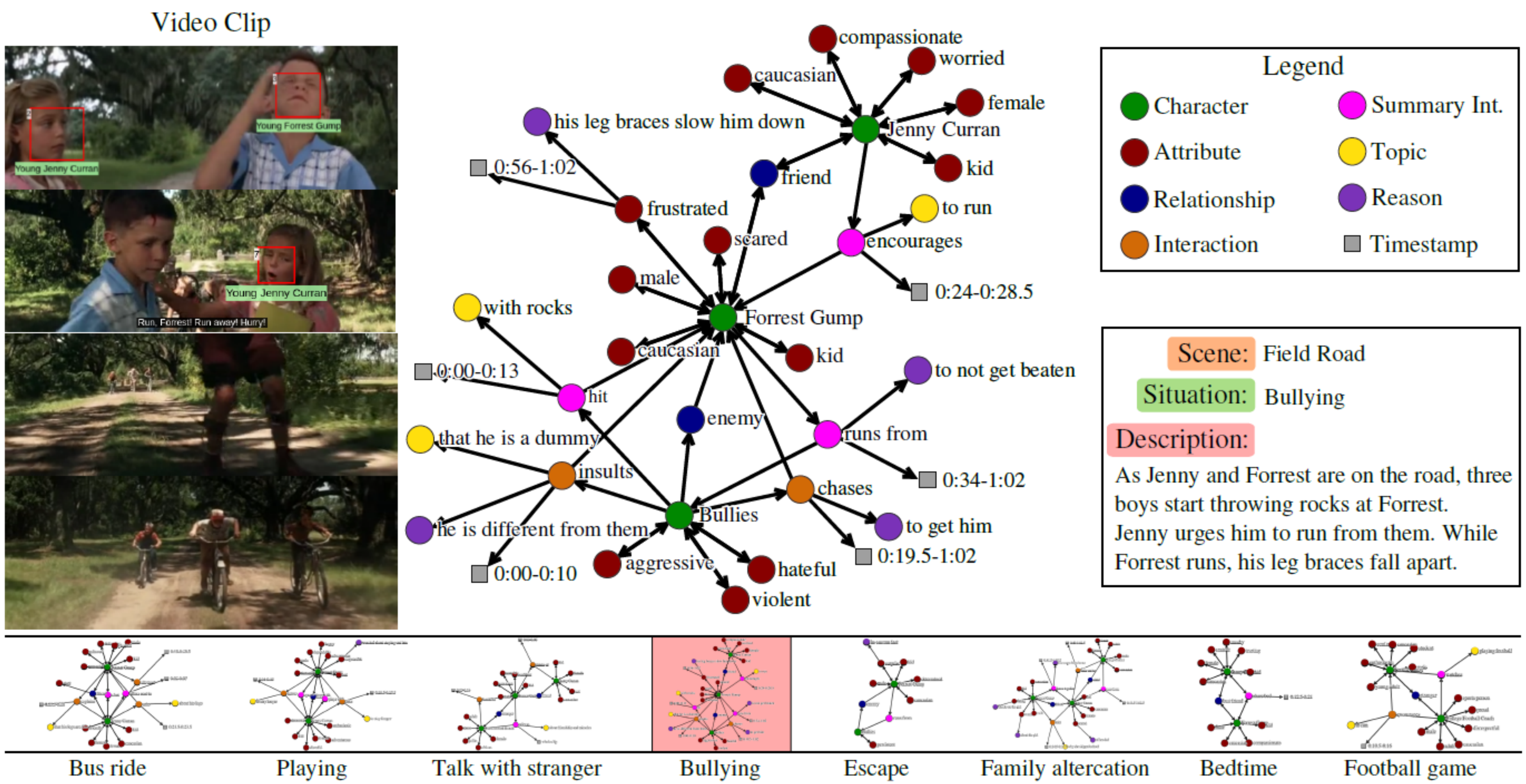


Chen et al., CVPR 2017

Understanding activities and intents

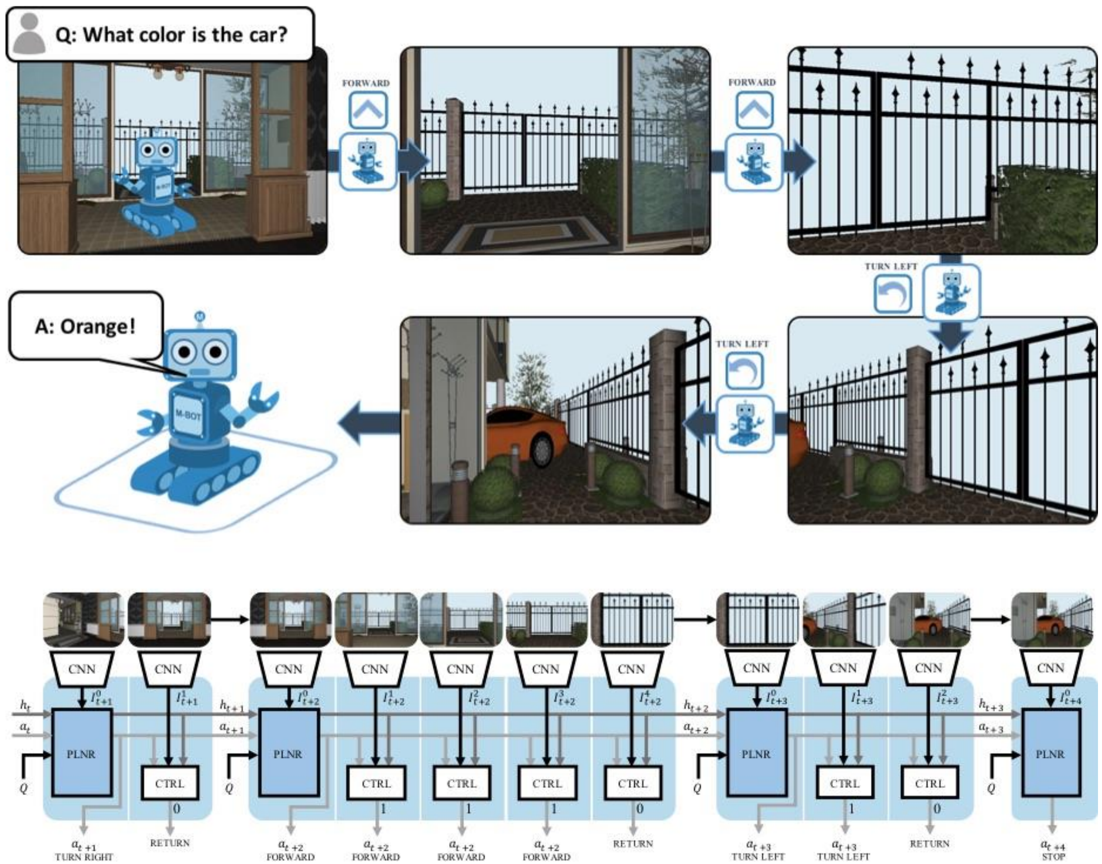
Bagautdinov et al., CVPR 2017

Understanding stories in film



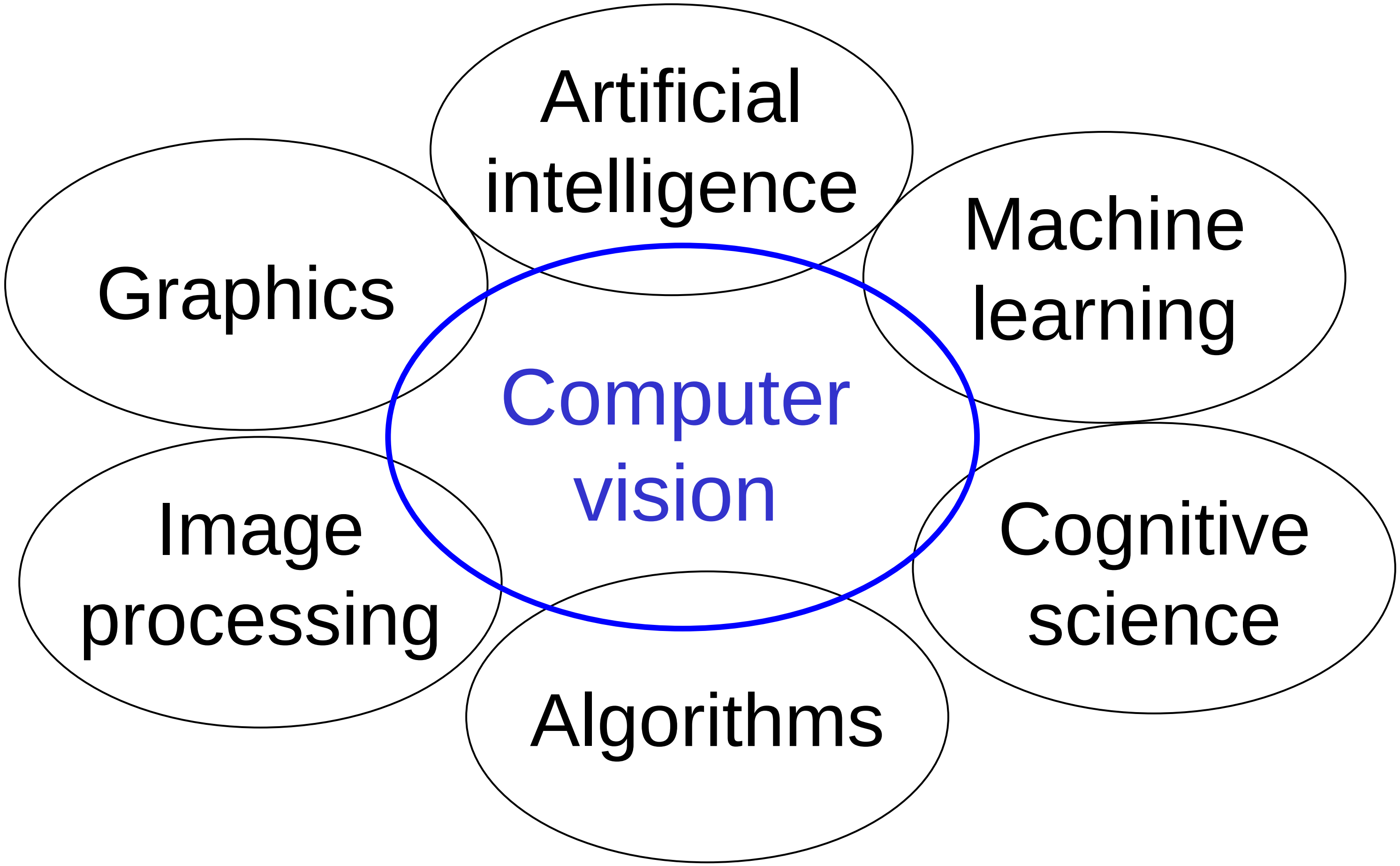
Vicol et al., CVPR 2018

Reasoning and acting: Embodied question answering



Das et al., CVPR 2018

Related disciplines



Kristen Grauman

Why vision?

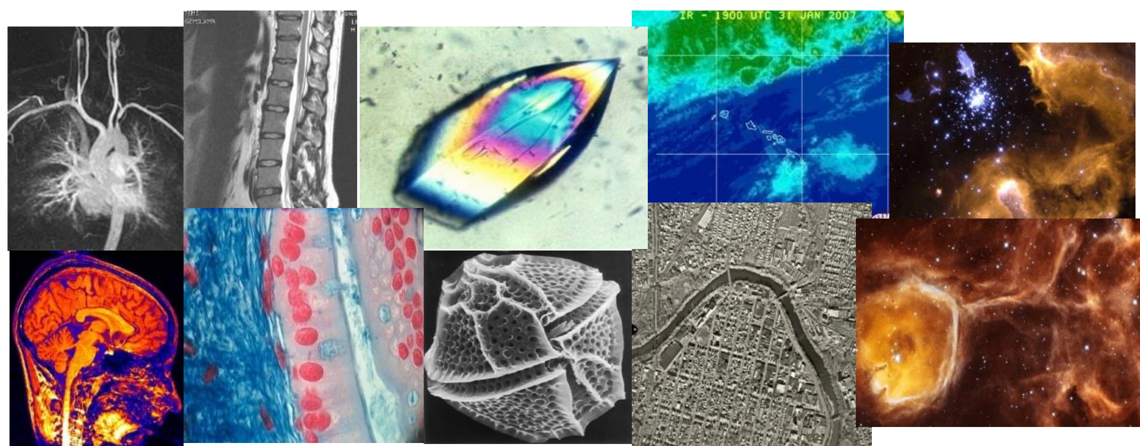
- Images and video are everywhere!

Personal photo albums

Movies, news, sports

Instagram flickr shutterstock YouTube gettyimages

Surveillance and security



Medical and scientific images

Adapted from Lana Lazebnik

Why vision?

- As image sources multiply, so do applications
  - Relieve humans of boring, easy tasks
  - Perception for robotics / autonomous agents
  - Organize and give access to visual content
  - Description of content for the visually impaired
  - Human-computer interaction
  - Fun applications (e.g. art styles to my photos)
  - …
  - What else?

Adapted from Kristen Grauman

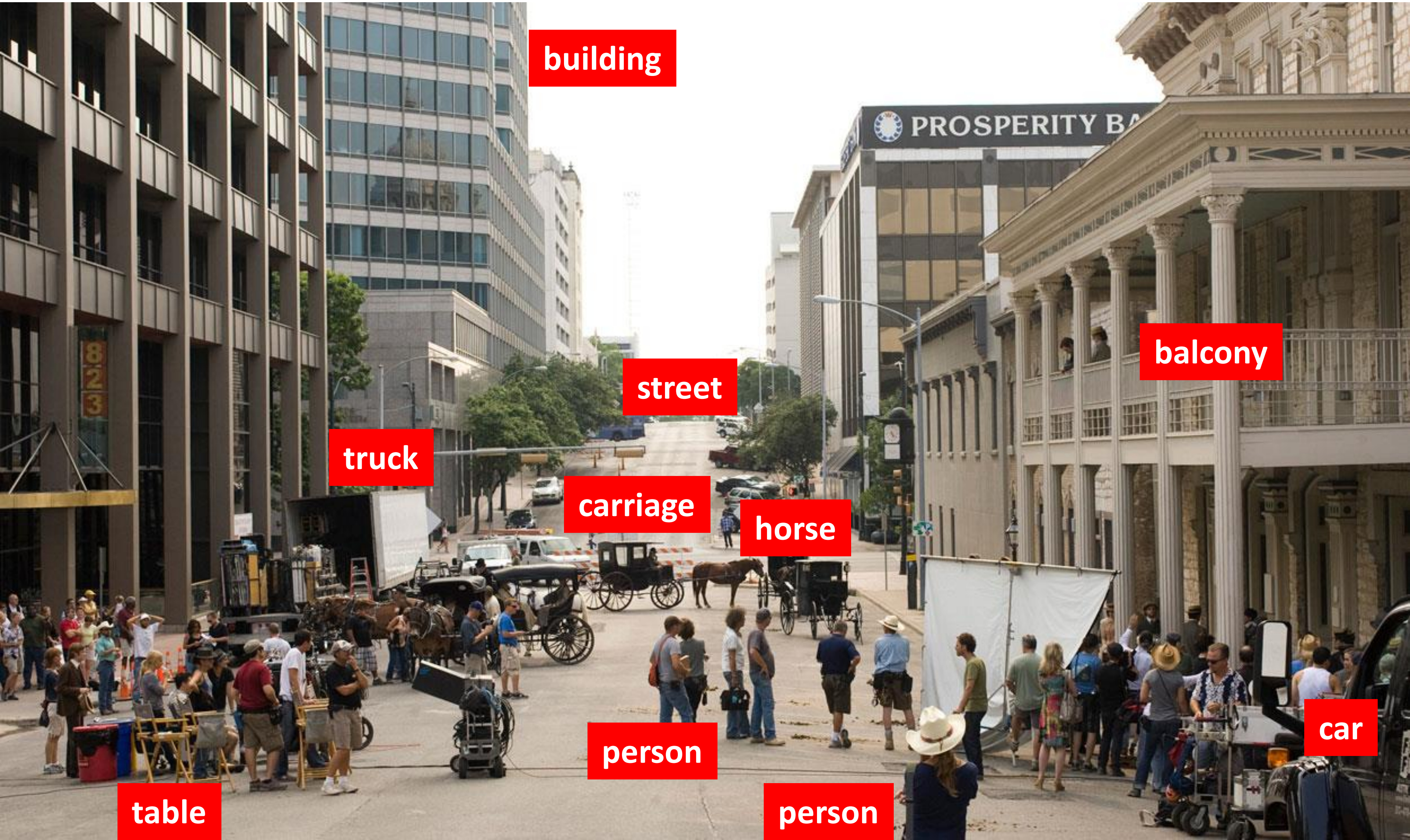
Seeing AI

YouTube link

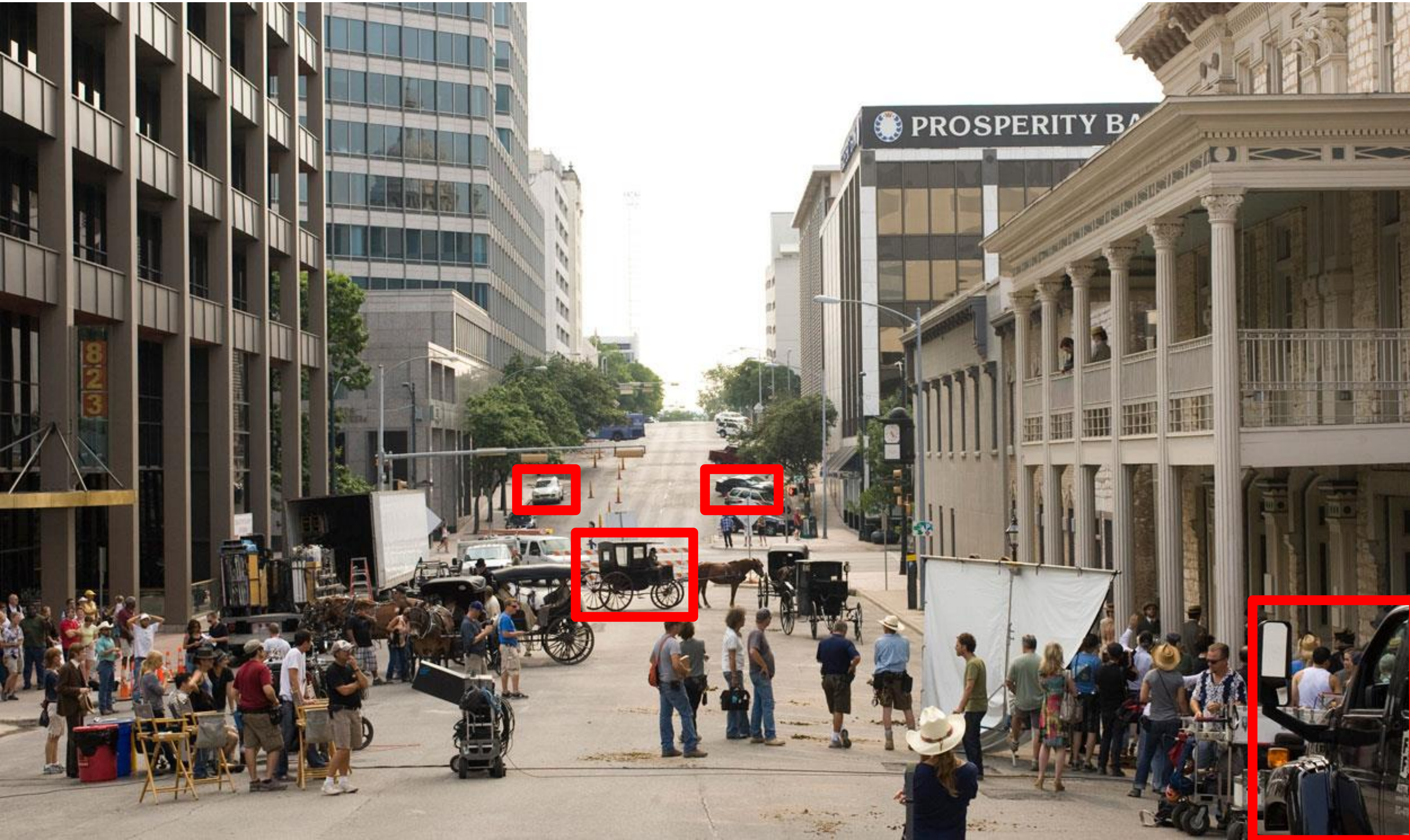
Microsoft Cognitive Services: Introducing the Seeing AI project

Why are these tasks challenging?

Recognition: What objects do you see?



Detection: Where are the cars?



Activity: What is this person doing?

Scene: Is this an indoor scene?

Instance: Which city? Which building?

Visual question answering:
Why is there a carriage in the street?

Why is vision difficult?

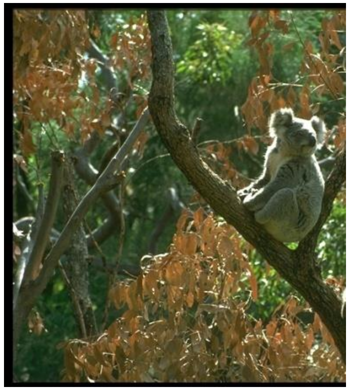
- Ill-posed problem: real world much more complex than what we can measure in images
  - 3D → 2D
  - Motion → static
- Impossible to literally “invert” image formation process with limited information
  - Need information outside of this particular image to generalize what image portrays (e.g. to resolve occlusion)

Adapted from Kristen Grauman

Challenges: many nuisance parameters

Illumination

Object pose



Clutter

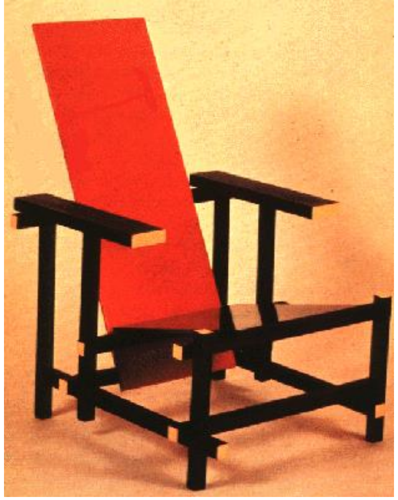
Occlusions

Intra-class appearance

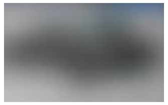
Viewpoint

Challenges: intra-class variation

CMOA Pittsburgh



Challenges: importance of context



slide credit: Fei-Fei, Fergus & Torralba

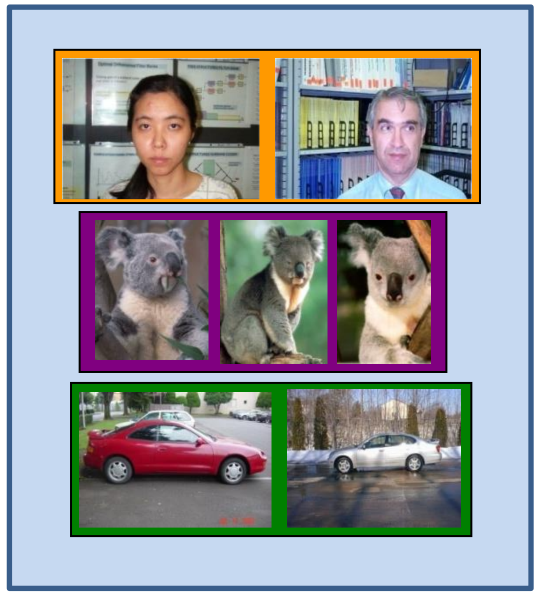
Challenges: Complexity

- Thousands to millions of pixels in an image
- 3,000-30,000 human recognizable object categories
- 30+ degrees of freedom in the pose of articulated objects (humans)
- Billions of images indexed by Google Image Search
- 1.424 billion smart camera phones sold in 2015
- About half of the cerebral cortex in primates is devoted to processing visual information [Felleman and van Essen 1991]

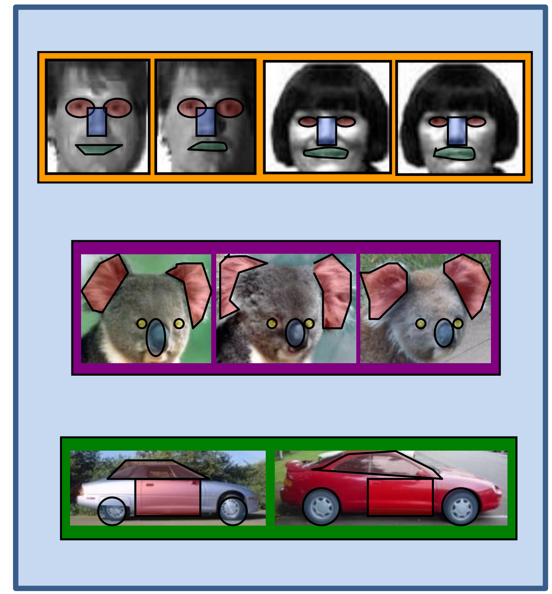
Challenges: Limited supervision

Less **More**

Unlabeled, multiple objects



Classes labeled, some clutter



Cropped to object, parts and classes labeled

Problem with categorization (Borges' Animals)

“These ambiguities, redundancies and deficiencies recall those that Dr. Franz Kuhn attributes to a certain Chinese dictionary entitled *The Celestial Emporium of Benevolent Knowledge*. In its remote pages it is written that animals can be divided into (a) those belonging to the Emperor, (b) those that are embalmed, (c) those that are tame, (d) pigs, (e) sirens, (f) imaginary animals, (g) wild dogs, (h) those included in this classification, (i) those that are crazy-acting, (j) those that are uncountable, (k) those painted with the finest brush made of camel hair, (l) miscellaneous, (m) those which have just broken a vase, and (n) those which, from a distance, look like flies.“

Jorge Luis Borges, **The Analytical Language of John Wilkins**, https://www.entish.org/essays/Wilkins.html

Preview of Readings

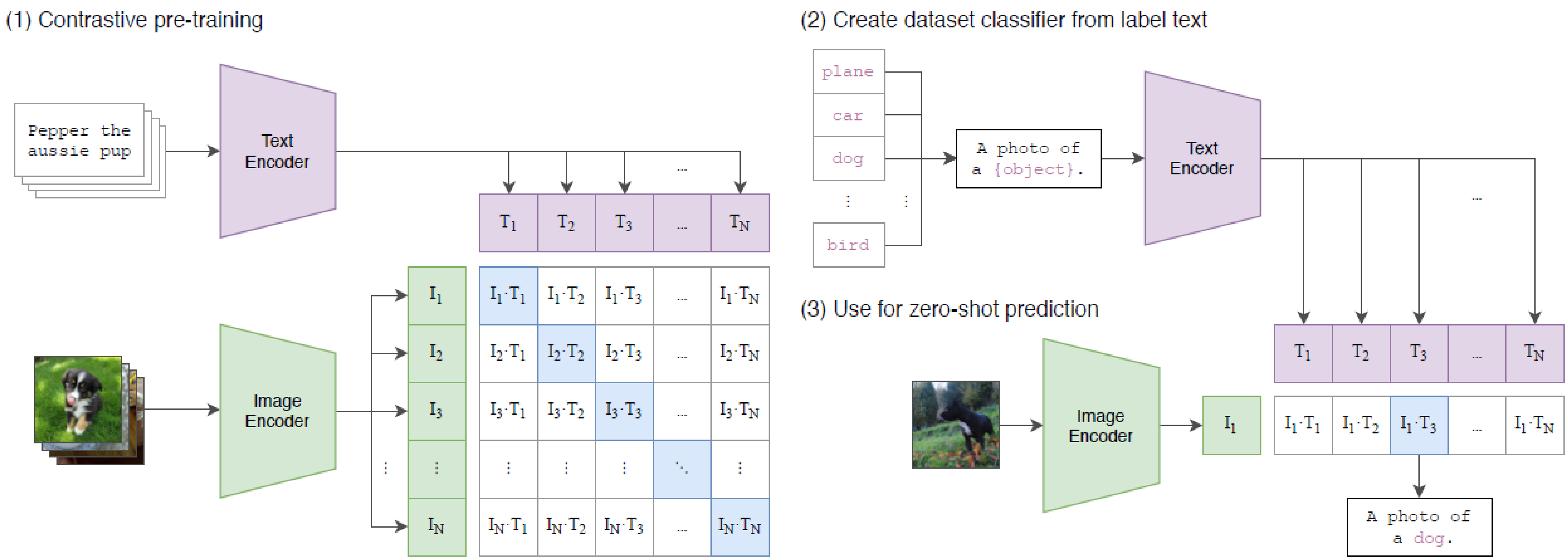


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Radford et al. "Learning Transferable Visual Models From Natural Language Supervision." ICML 2021.

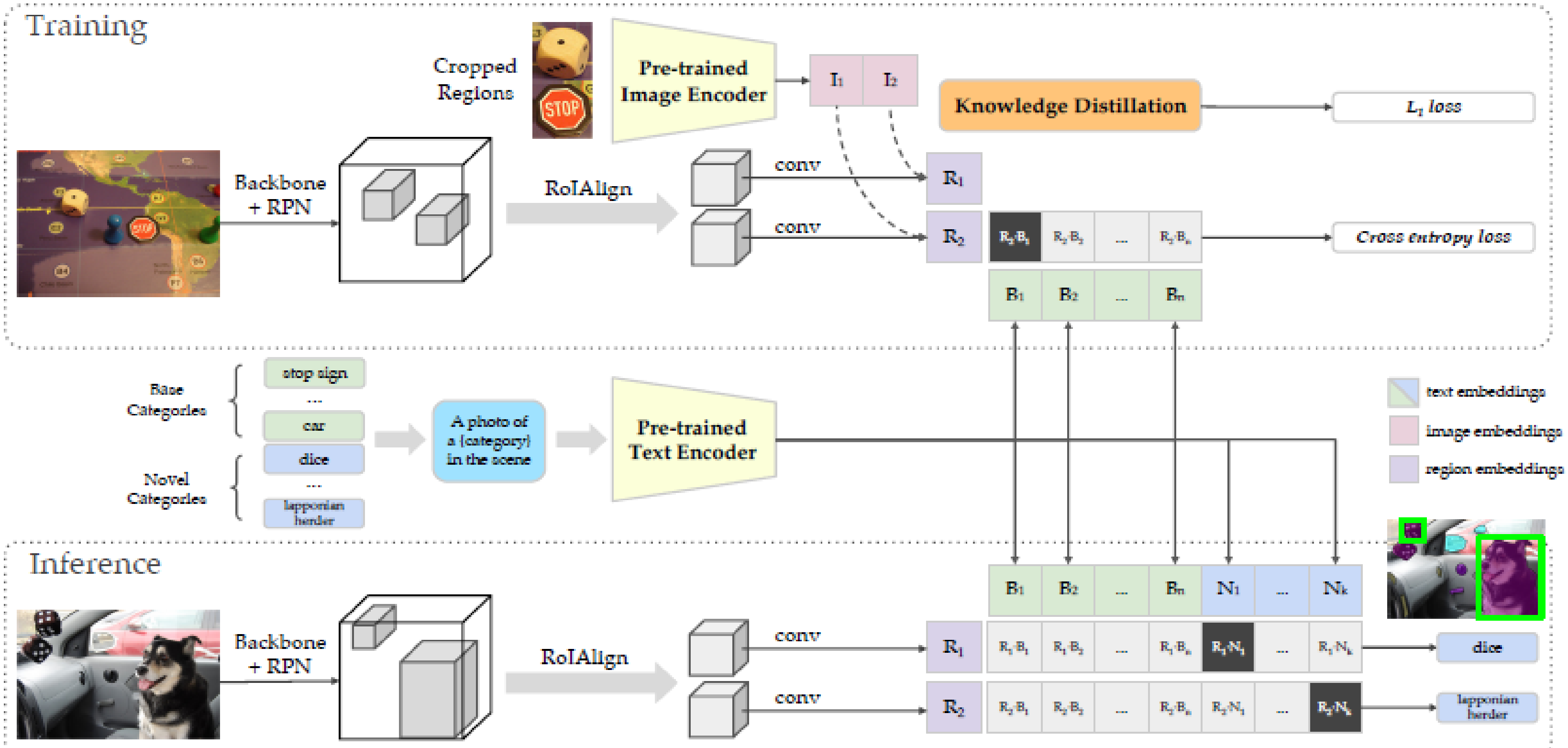


Figure 2: **An overview of using ViLD for open-vocabulary object detection.** ViLD distills the knowledge from a pretrained open-vocabulary image classification model. First, the category text embeddings and the image embeddings of cropped object proposals are computed, using the text and image encoders in the pretrained classification model. Then, ViLD employs the text embeddings as the region classifier (ViLD-text) and minimizes the distance between the region embedding and the image embedding for each proposal (ViLD-image). During inference, text embeddings of novel categories are used to enable open-vocabulary detection.

Gu et al. "Open-vocabulary Object Detection via Vision and Language Knowledge Distillation." ICLR 2021.

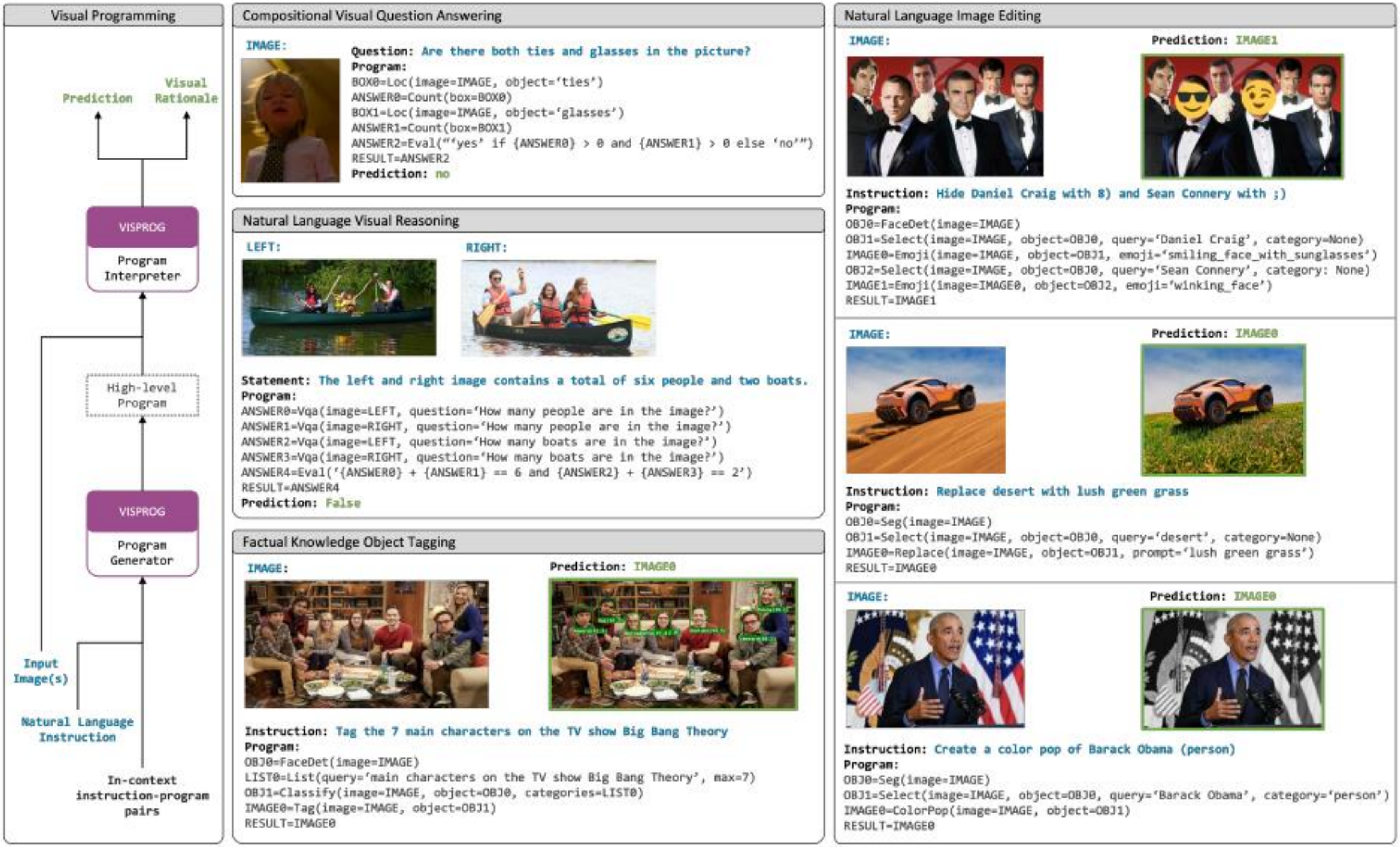


Figure 1. **VISPROG is a modular and interpretable neuro-symbolic system for compositional visual reasoning.** Given a few examples of natural language instructions and the desired high-level programs, VISPROG generates a program for any new instruction using *in-context learning* in GPT-3 and then executes the program on the input image(s) to obtain the prediction. VISPROG also summarizes the intermediate outputs into an interpretable *visual rationale* (Fig. 4). We demonstrate VISPROG on tasks that require composing a diverse set of modules for image understanding and manipulation, knowledge retrieval, and arithmetic and logical operations.

Gupta and Kembhavi. "Visual Programming: Compositional visual reasoning without training." CVPR 2023.

Caltech101

Prompt	Accuracy
a [CLASS].	82.68
a photo of [CLASS].	80.81
a photo of a [CLASS].	86.29
$[V]_1$ $[V]_2$ ... $[V]_M$ [CLASS].	**91.83**

(a)

Flowers102

Prompt	Accuracy
a photo of a [CLASS].	60.86
a flower photo of a [CLASS].	65.81
a photo of a [CLASS], a type of flower.	66.14
$[V]_1$ $[V]_2$ ... $[V]_M$ [CLASS].	**94.51**

(b)

Describable Textures (DTD)

Prompt	Accuracy
a photo of a [CLASS].	39.83
a photo of a [CLASS] texture.	40.25
[CLASS] texture.	42.32
$[V]_1$ $[V]_2$ ... $[V]_M$ [CLASS].	**63.58**

(c)

EuroSAT

Prompt	Accuracy
a photo of a [CLASS].	24.17
a satellite photo of [CLASS].	37.46
a centered satellite photo of [CLASS].	37.56
$[V]_1$ $[V]_2$ ... $[V]_M$ [CLASS].	**83.53**

(d)

Fig. 1 Prompt engineering vs Context Optimization (CoOp). The former needs to use a held-out validation set for words tuning, which is inefficient; the latter automates the process and requires only a few labeled images for learning.

Zhou et al. "Learning to Prompt for Vision-Language Models." IJCV 2022.

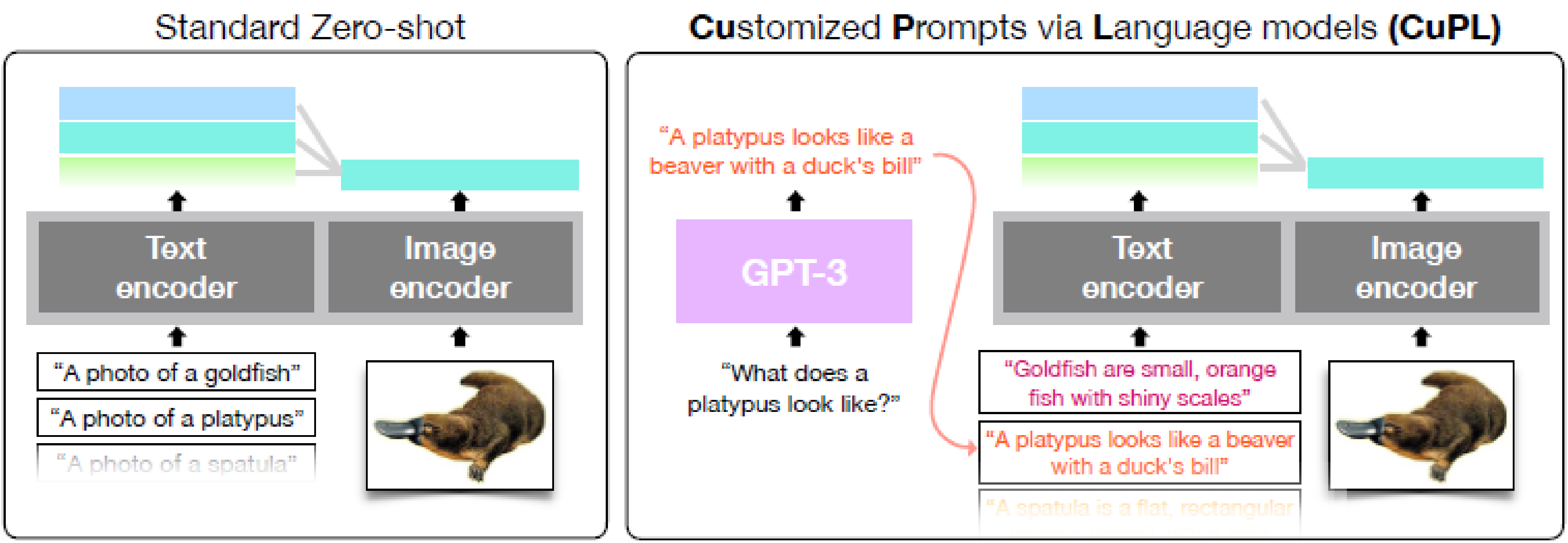


Figure 1: **Schematic of the method.** (Left) The standard method of a zero-shot open vocabulary image classification model (e.g., CLIP (Radford et al., 2021)). (Right) Our method of CuPL. First, an LLM generates descriptive captions for given class categories. Next, an open vocabulary model uses these captions as prompts for performing classification.

Pratt et al. "What does a platypus look like? Generating customized prompts for zero-shot image classification." ICCV 2023.

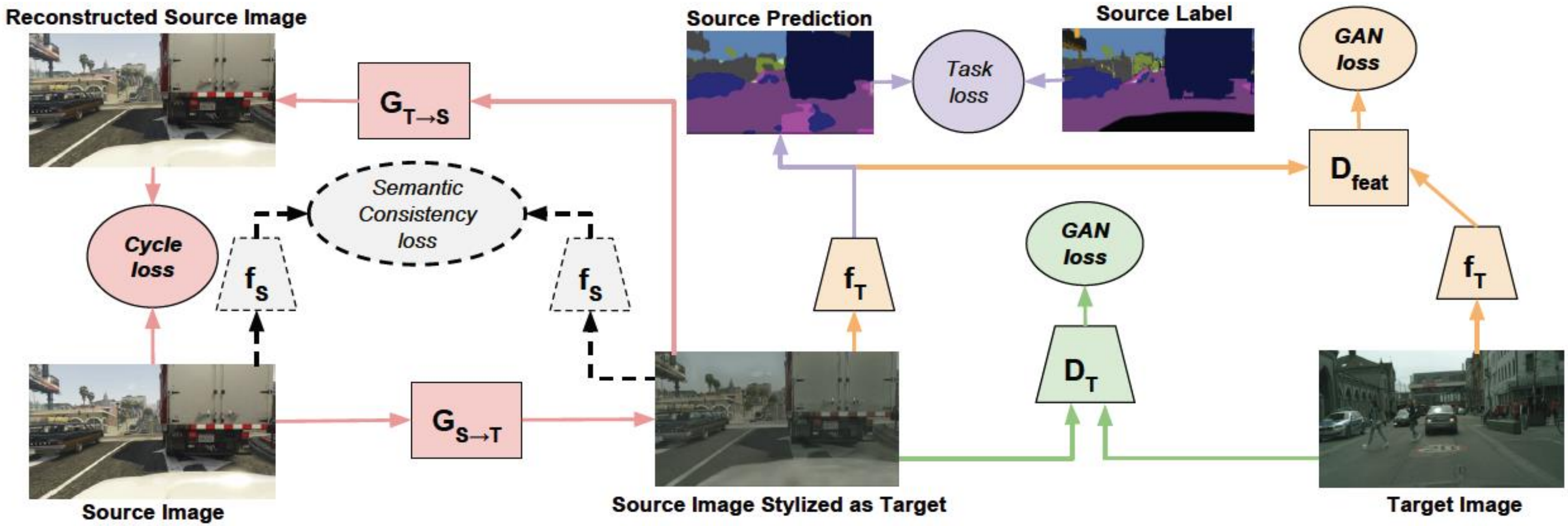
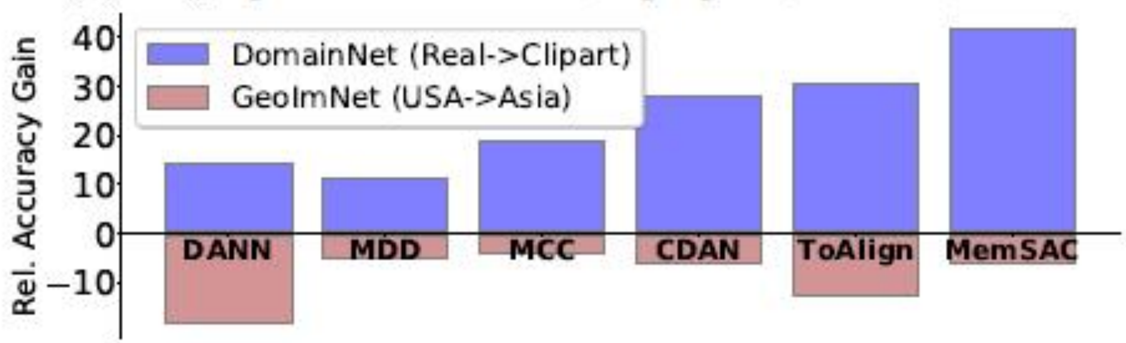


Figure 2: Cycle-consistent adversarial adaptation overview. By directly remapping source training data into the target domain, we remove the low-level differences between the domains, ensuring that our task model is well-conditioned on target data. We depict here the image-level adaptation as composed of the pixel GAN loss (green), the source cycle loss (red), and the source and target semantic consistency losses (black dashed) – used when needed to prevent label flipping. For clarity the target cycle is omitted. The feature-level adaptation is depicted as the feature GAN loss (orange) and the source task loss (purple).

Hoffman et al. "Cycada: Cycle-consistent adversarial domain adaptation." ICML 2018.

(a) Geographic bias manifested in proposed GeoNet dataset



(b) Unsupervised domain adaptation does not suffice on GeoNet

Kalluri et al. "GeoNet: Benchmarking Unsupervised Adaptation across Geographies." CVPR 2023.

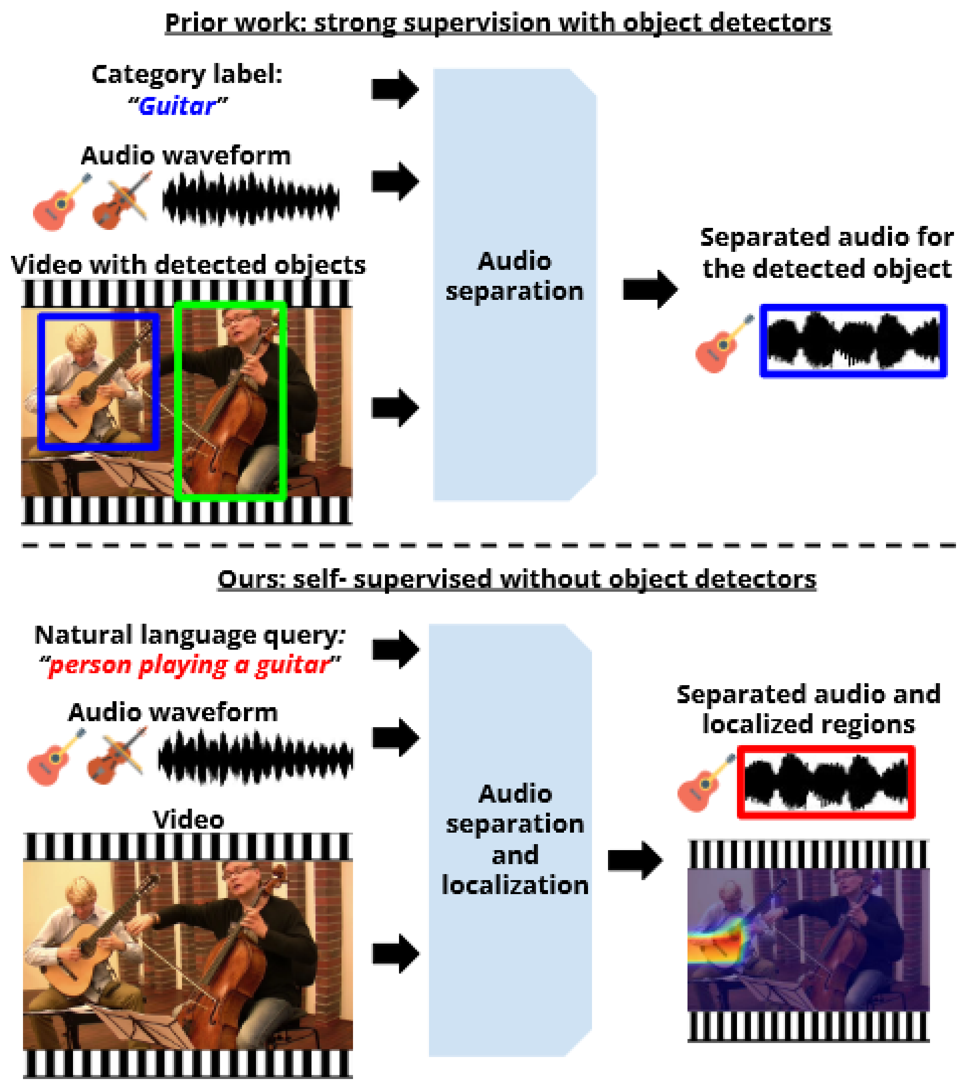


Figure 1. We propose to separate and localize audio sources based on a natural language query, by learning to align the modalities on completely unlabeled videos. In comparison, prior audio-visual sound separation approaches require object label supervision.

Tan et al. "Language-Guided Audio-Visual Source Separation via Trimodal Consistency." CVPR 2023.

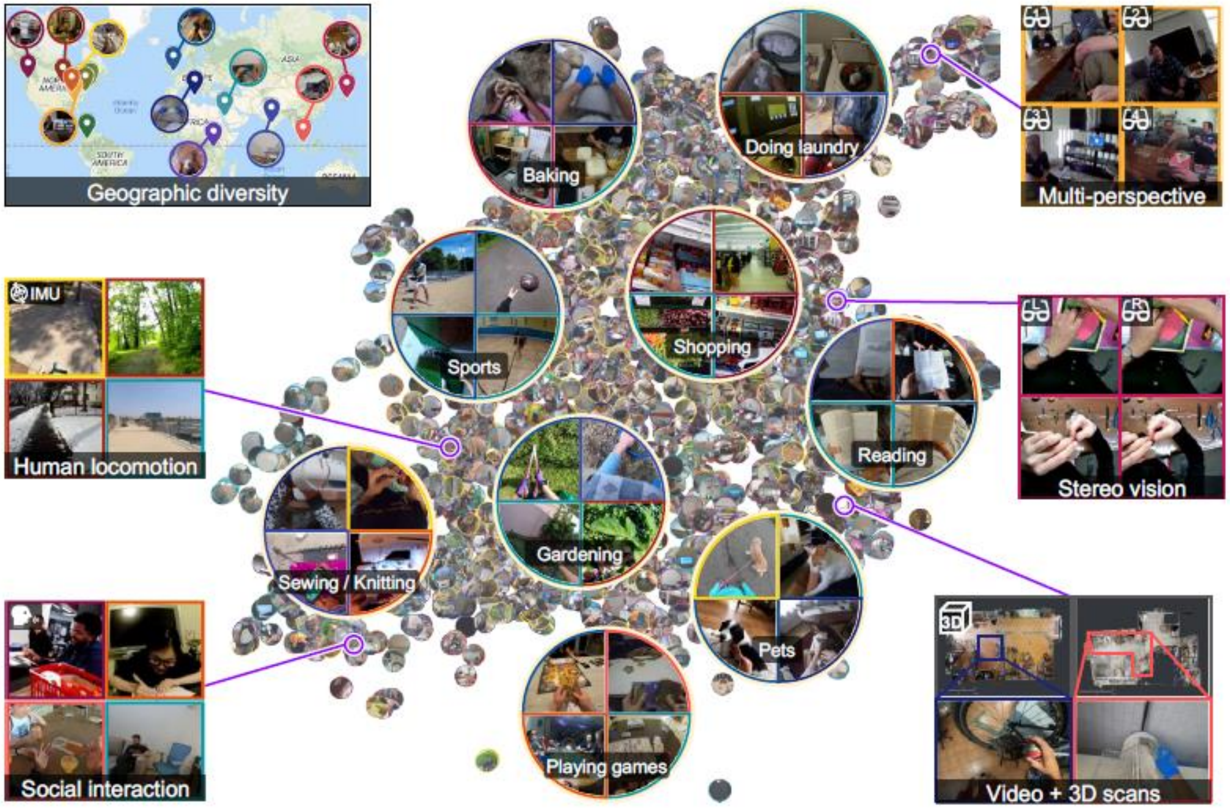


Figure 1. Ego4D is a massive-scale egocentric video dataset of daily life activity spanning 74 locations worldwide. Here we see a snapshot of the dataset (5% of the clips, randomly sampled) highlighting its diversity in geographic location, activities, and modalities. The data includes social videos where participants consented to remain unblurred. See https://ego4d-data.org/fig1.html for interactive figure.

Grauman et al. "Ego4D: Around the World in 3,000 Hours of Egocentric Video." CVPR 2022.

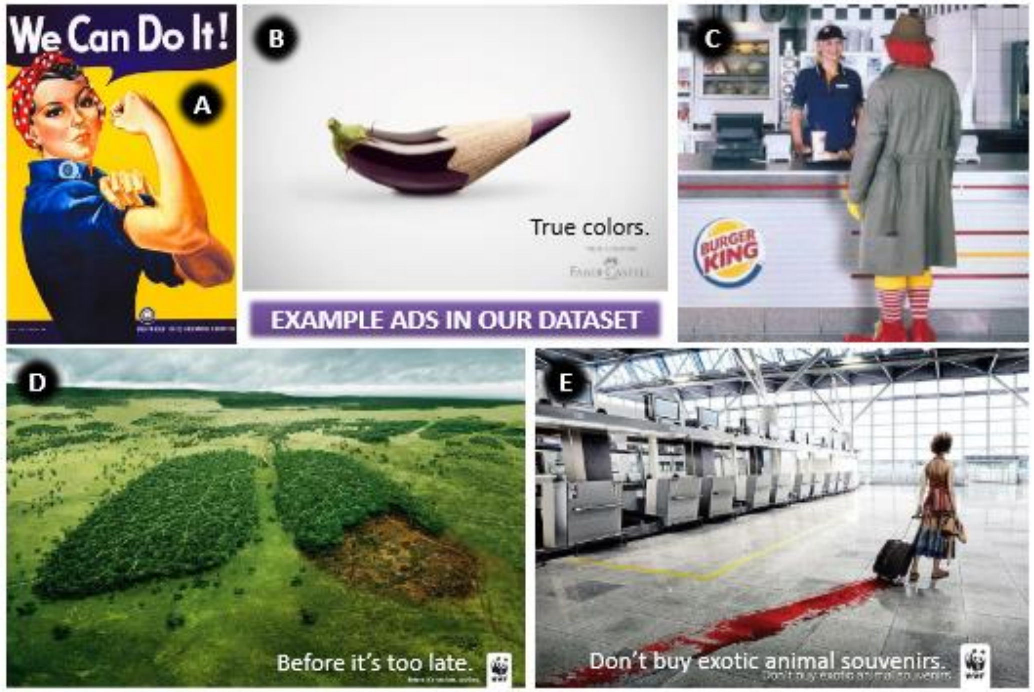


Fig. 1: Example advertisements from our dataset that require challenging visual recognition and reasoning. Despite the potential applications of understanding the messages of ads, this problem has not been tackled in computer vision.

Ye et al. "Interpreting the Rhetoric of Visual Advertisements." TPAMI 2019.

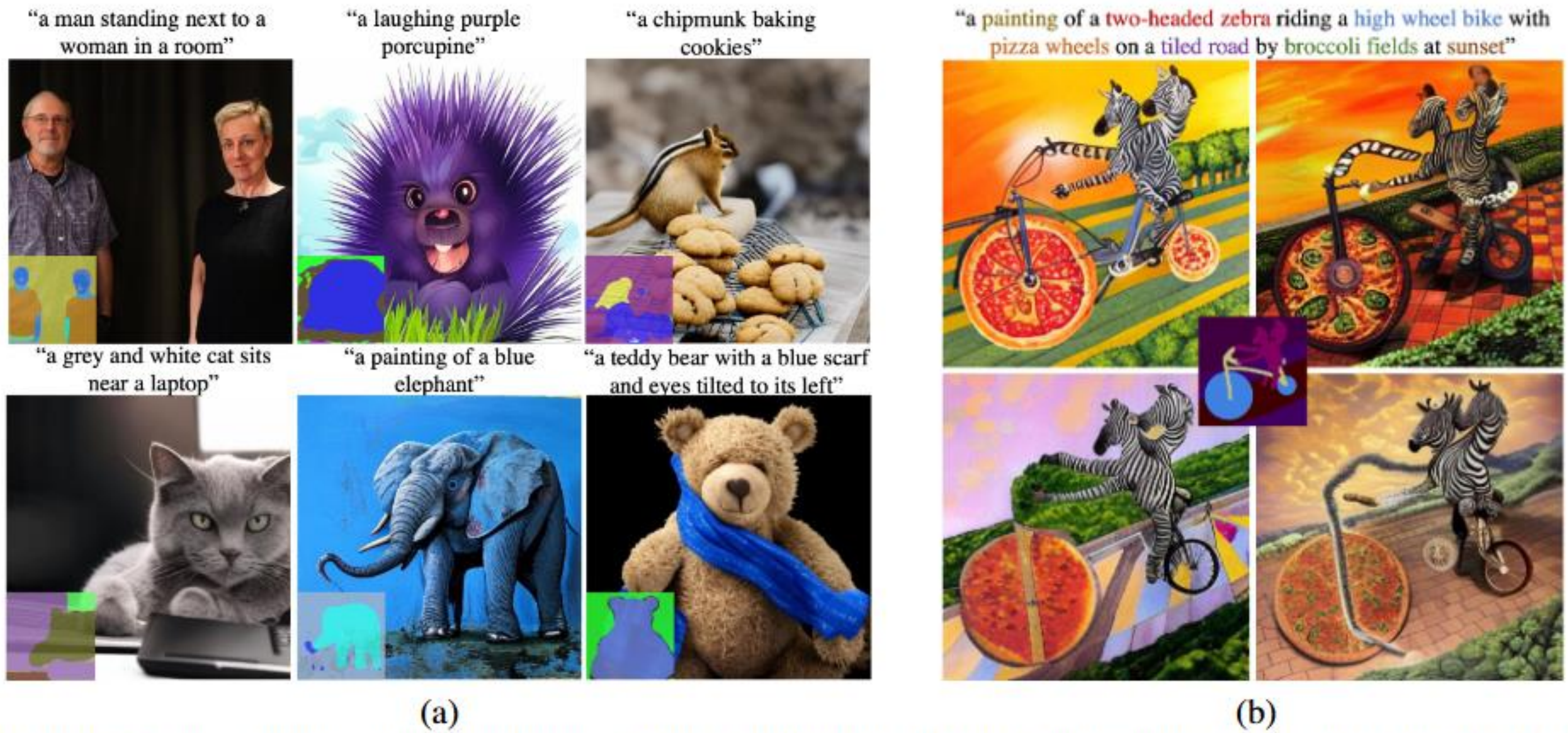


Figure 1. Make-A-Scene: Samples of generated images from text inputs (a), and a text and scene input (b). Our method is able to both generate the scene (a, bottom left) and image, or generate the image from text and a simple sketch input (b, center).

Gafni et al. "Make-A-Scene: Scene-Based Text-to-Image Generation with Human Priors." ECCV 2022.

Figure 1: LLMs have not interacted with their environment and observed the outcome of their responses, and thus are not grounded in the world. SayCan grounds LLMs via value functions of pretrained skills, allowing them to execute real-world, abstract, long-horizon commands on robots.

Ahn et al. "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances." CoRL 2023.

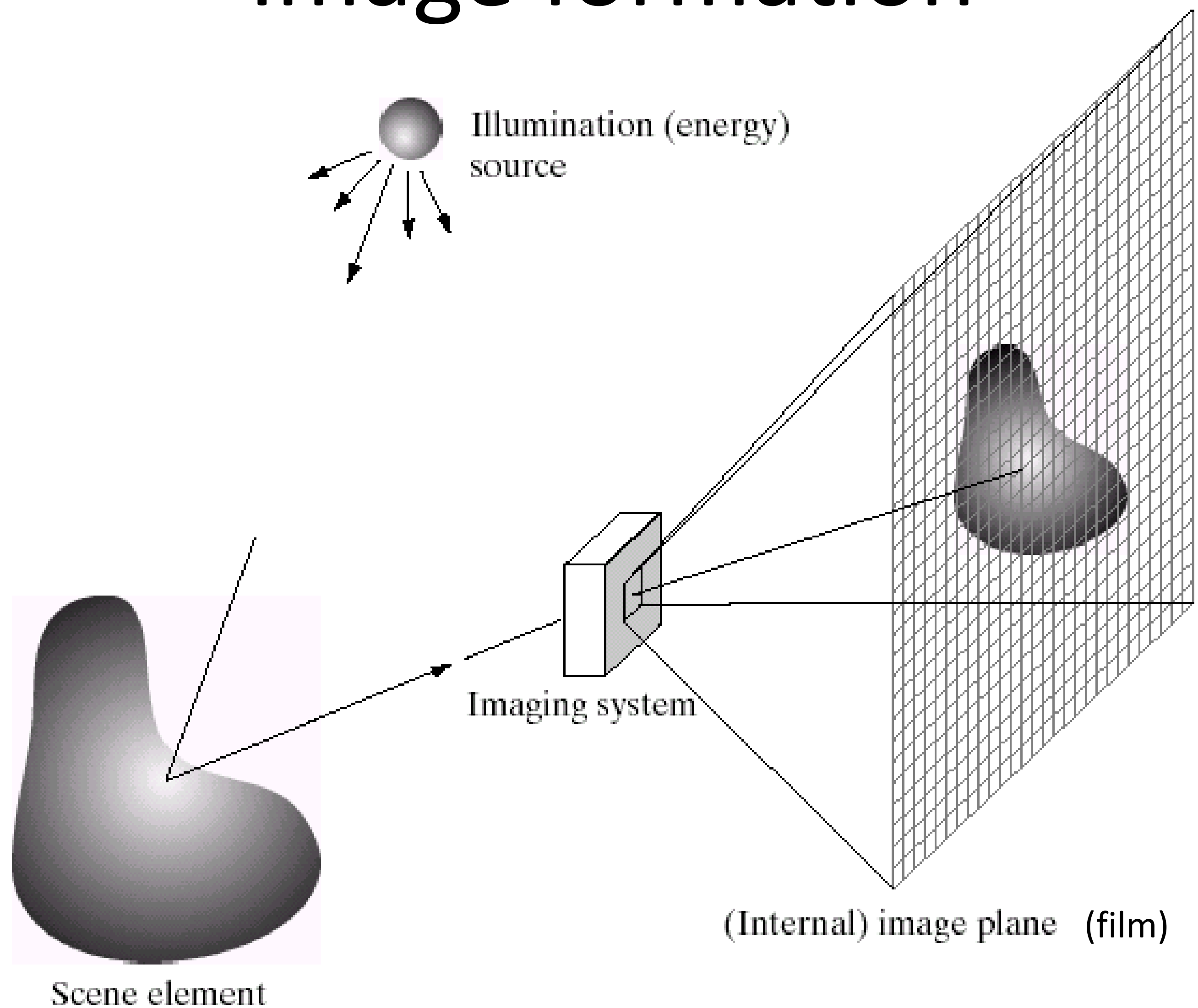
Paper Presentation Sign-ups

https://docs.google.com/spreadsheets/d/14g6_finym215LqDZ2qnAE57bU5j5M9uHz3Tmmgq484U/edit?usp=sharing

Some Basics
(breezing through to establish common ground…)

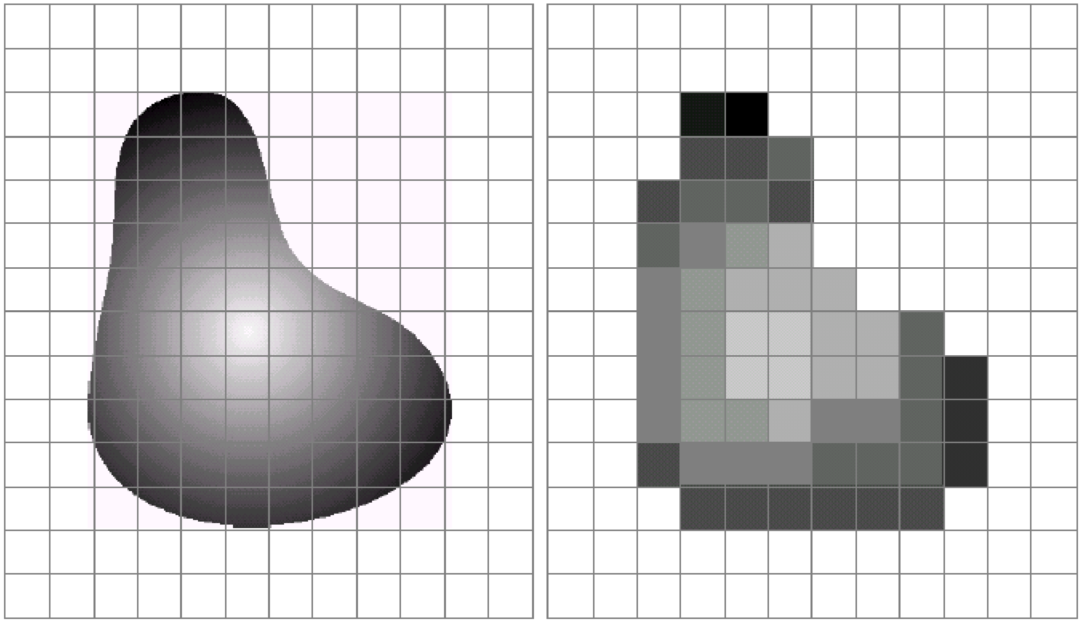
Convolutional networks
Recurrent networks
Transformers
Self-supervised learning

Image formation



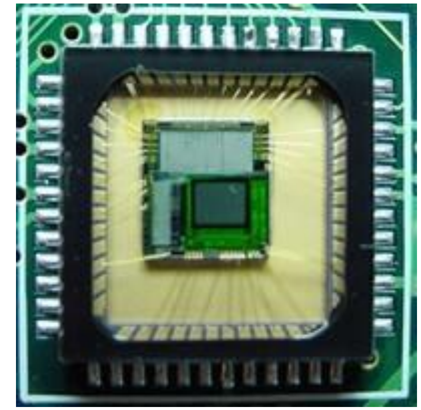
Slide credit: Derek Hoiem

Digital images



a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

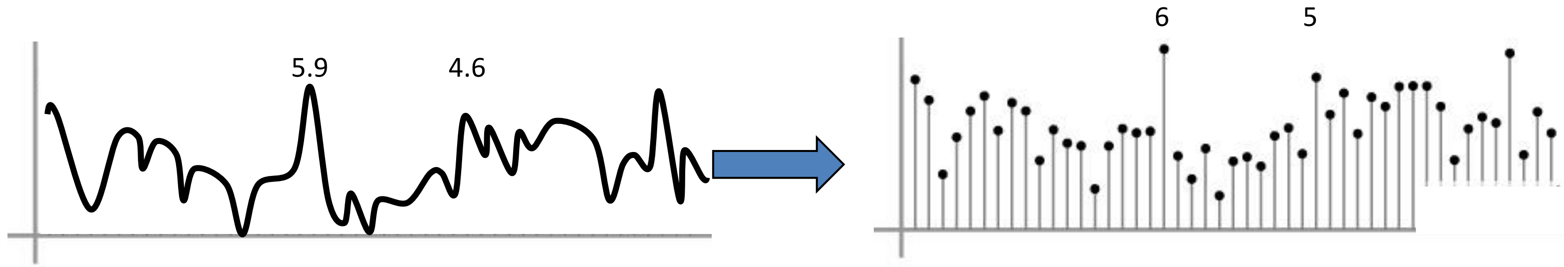


- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)

Digital images

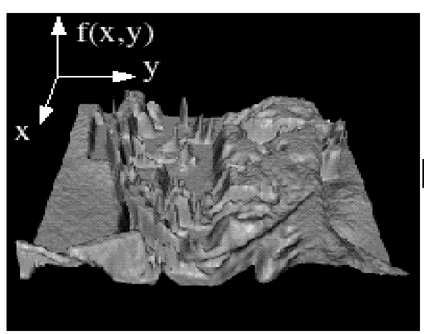
- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- What does quantizing signal look like?

1D



- Image thus represented as a matrix of integer values.

2D



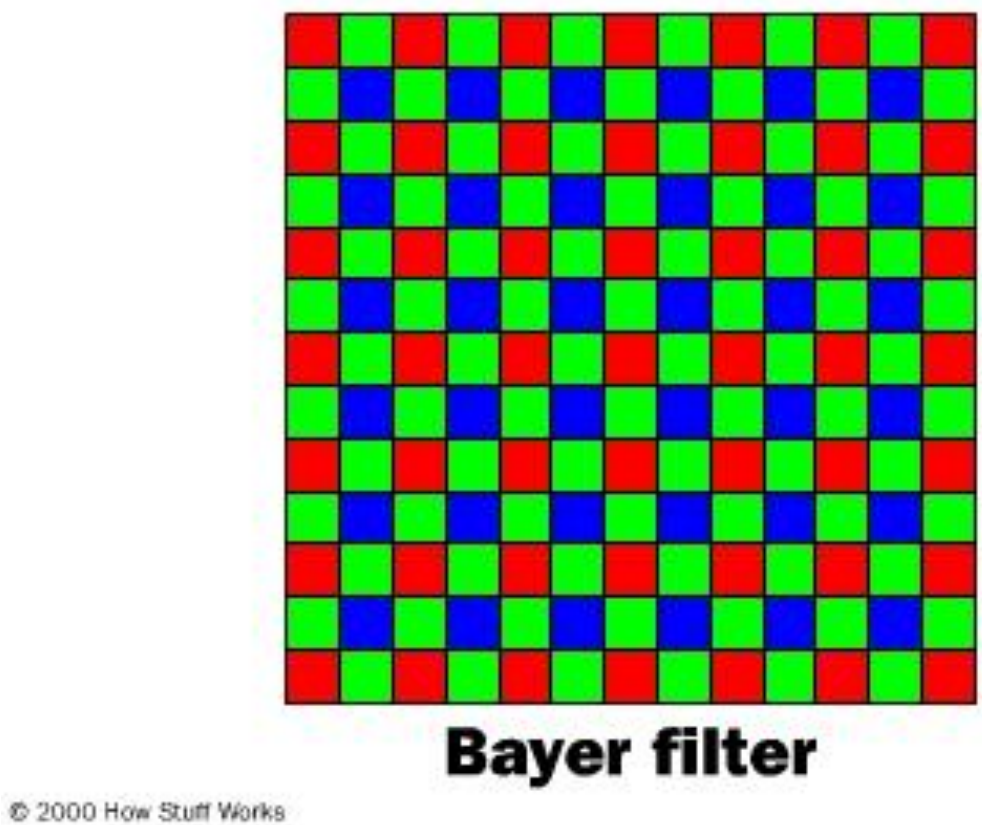
j

i

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Adapted from S. Seitz

Digital color images

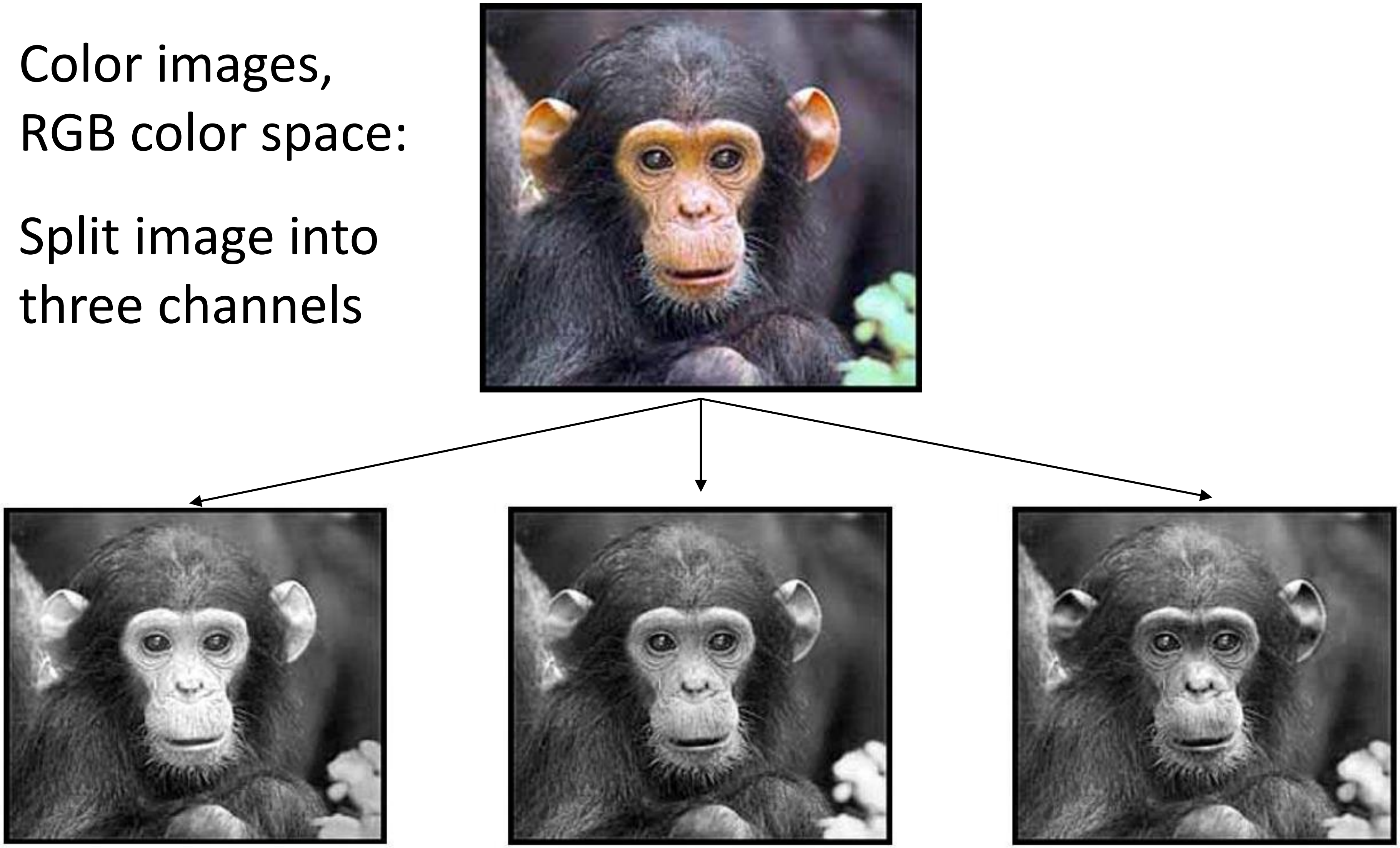


Slide credit: Kristen Grauman

Digital color images

Color images,
RGB color space:

Split image into
three channels



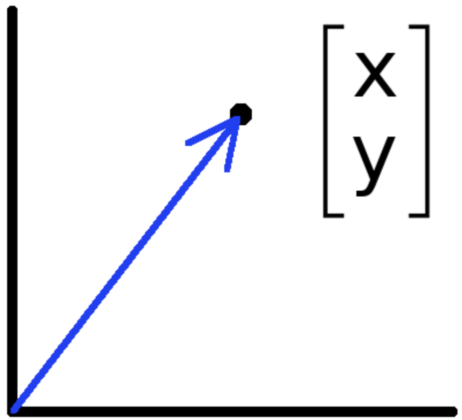
R G B

Adapted from Kristen Grauman

Images as Matrices

- Vectors and matrices are just collections of ordered numbers that represent something: movements in space, word counts, movie ratings, pixel brightnesses, etc.

Vectors have two main uses



- Vectors can represent an offset in 2D or 3D space
- Points are just vectors from the origin

- Data can also be treated as a vector
- Such vectors don’t have a geometric interpretation, but calculations like “distance” still have value

Vector

- A column vector $\mathbf{v} \in \mathbb{R}^{n\times 1}$ where

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

- A row vector $\mathbf{v}^T \in \mathbb{R}^{1\times n}$ where

$$\mathbf{v}^T = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$$

T denotes the transpose operation

Norms

- L1 norm

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$$

- L2 norm

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$$

- L^p norm (for real numbers $p \geq 1$)

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Distances

- L1 (Manhattan) distance

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

- L2 (Euclidean) distance

$$\boldsymbol{d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}}$$

$$d(p,q) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_i-q_i)^2 + \cdots + (p_n-q_n)^2}.$$

Example: Feature representation

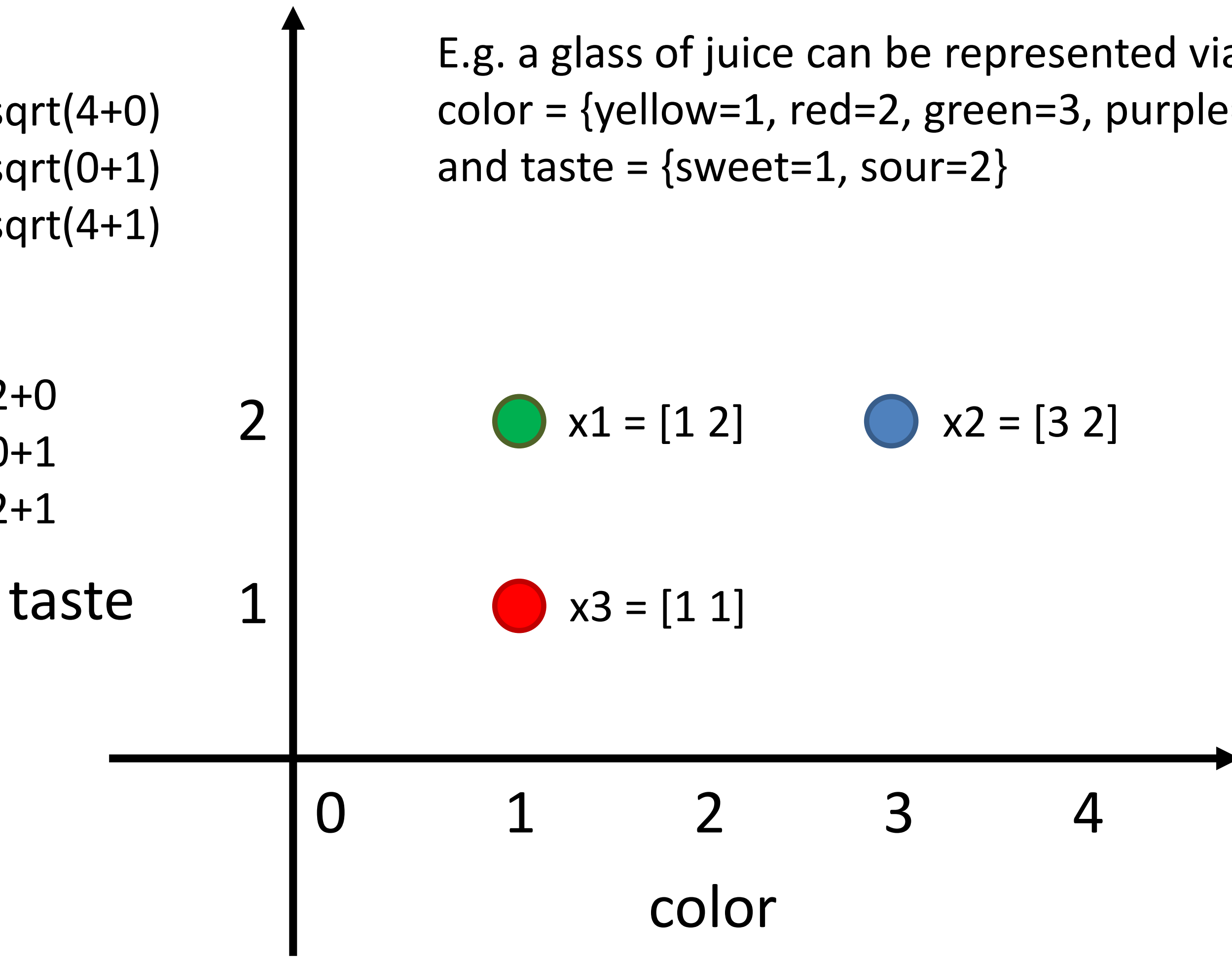
- A vector representing measurable characteristics of a data sample we have
- E.g. a glass of juice can be represented via its color = {yellow=1, red=2, green=3, purple=4} and taste = {sweet=1, sour=2}
- A given glass *i* can be represented as a vector: $\boldsymbol{x}_i$ = [3 2] represents *green, sour* juice
- For *D* features, this defines a *D*-dimensional space where we can measure similarity between samples

Example: Feature representation

L2 distance:
d(x1, x2) = sqrt(4+0)
d(x1, x3) = sqrt(0+1)
d(x2, x3) = sqrt(4+1)

L1 distance:
d(x1, x2) = 2+0
d(x1, x3) = 0+1
d(x2, x3) = 2+1

E.g. a glass of juice can be represented via its color = {yellow=1, red=2, green=3, purple=4} and taste = {sweet=1, sour=2}

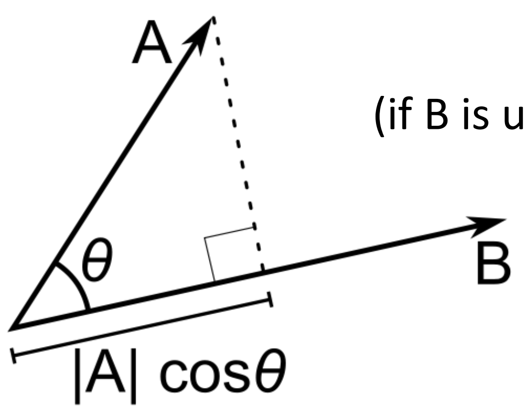


Inner (Dot) Product

- Multiply corresponding entries of two vectors and add up the result

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i \qquad \text{(scalar)}$$

- x·y is also |x||y|Cos(*angle between x and y*)
- If B is a unit vector, then A·B gives the length of A which lies in the direction of B (projection)



(if B is unit-length hence norm is 1)

Traditional Recognition Approach

Image/ Video Pixels → **Hand-designed feature extraction (SIFT+BOW, HOG)** → **Trainable classifier** → Object Class

- Features are key to recent progress in recognition, but research shows they're flawed… Where next?

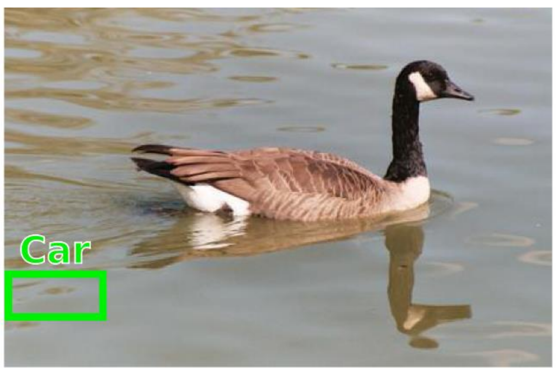


Fig. 1: An image from PASCAL and a high scoring car detection from DPM (Felzenszwalb et al, 2010b). Why did the detector fail?

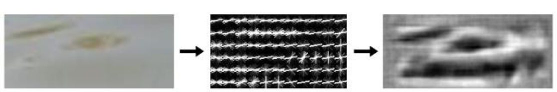
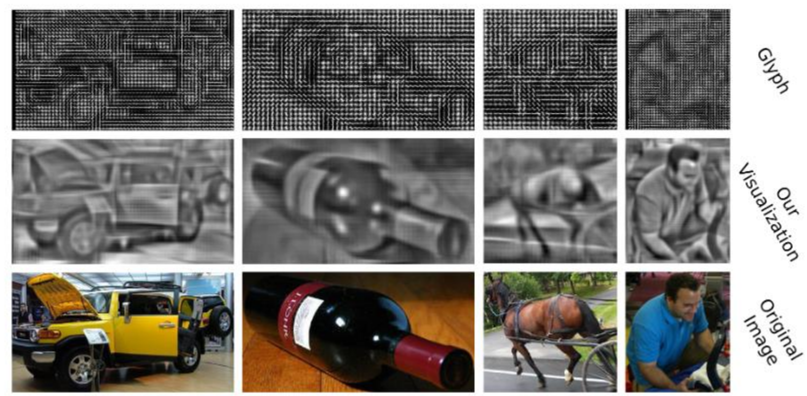
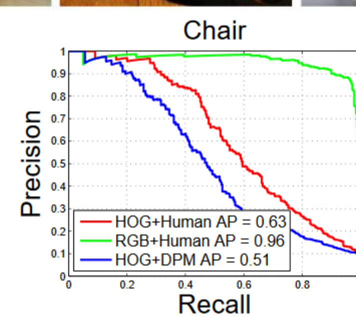


Fig. 2: We show the crop for the false car detection from Figure 1. On the right, we show our visualization of the HOG features for the same patch. Our visualization reveals that this false alarm actually looks like a car in HOG space.

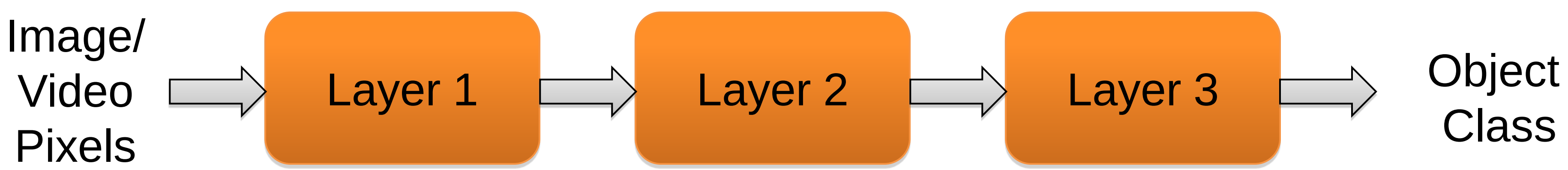




Adapted from Lana Lazebnik, figures from Vondrick: http://www.cs.columbia.edu/~vondrick/ihog/ijcv.pdf

What about learning the features?

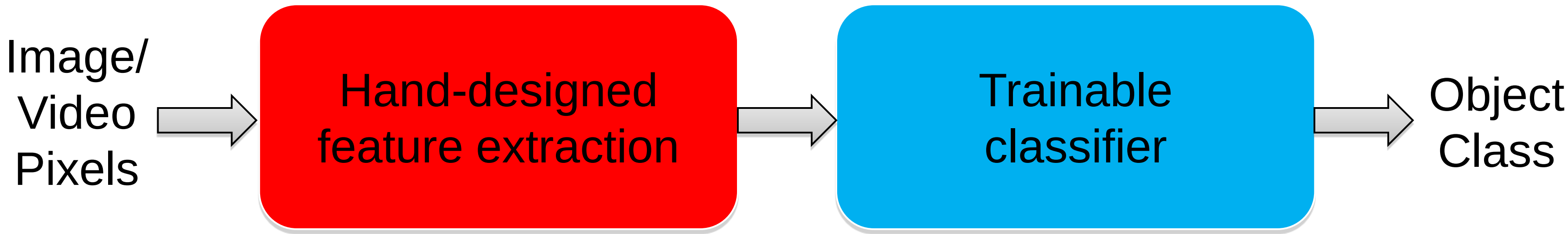
- Learn a *feature hierarchy* all the way from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly



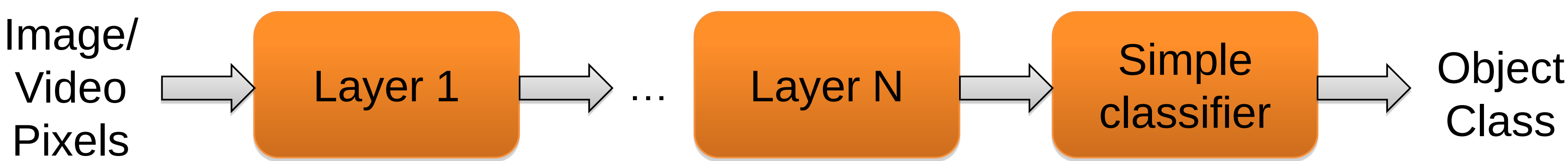
Lana Lazebnik

“Shallow” vs. “deep” architectures

Traditional recognition: “Shallow” architecture



Deep learning: “Deep” architecture



Neural network definition

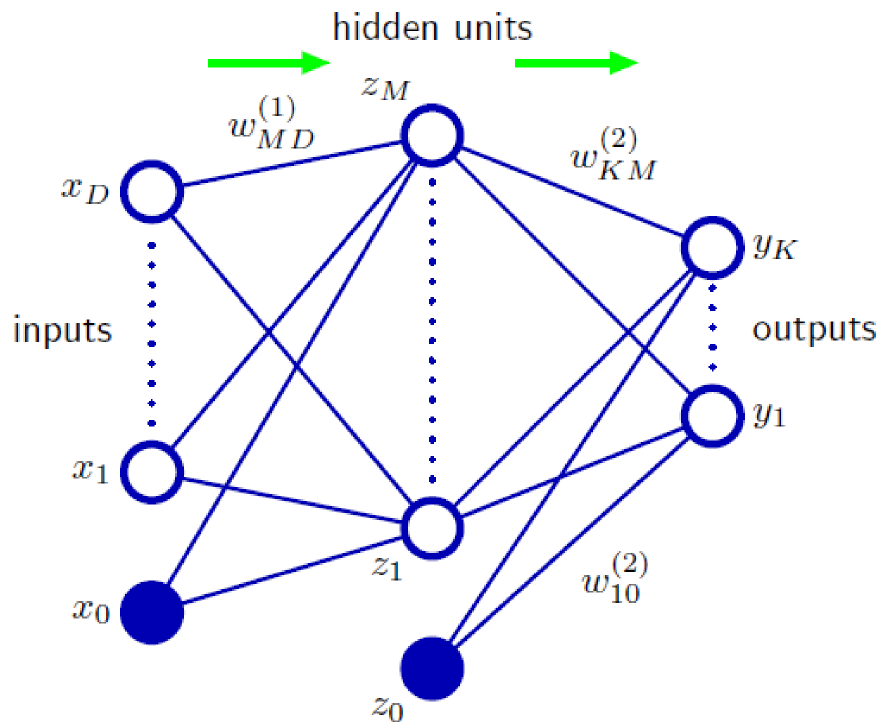


Figure 5.1 Network diagram for the two-layer neural network corresponding to (5.7). The input, hidden, and output variables are represented by nodes, and the weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links coming from additional input and hidden variables x_0 and z_0. Arrows denote the direction of information flow through the network during forward propagation.

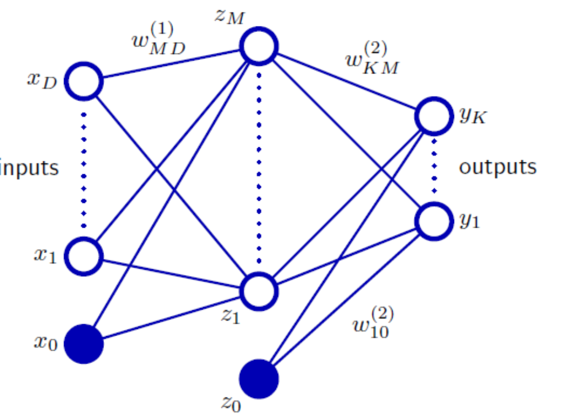
- Activations: $$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

Recall SVM: $w^T x + b$

- Nonlinear activation function *h* (e.g. sigmoid, RELU): $$z_j = h(a_j)$$

Figure from Christopher Bishop

Neural network definition



- Layer 2

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

- Layer 3 (final)

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

- Outputs (e.g. sigmoid/softmax)

(binary) $$y_k = \sigma(a_k) = \frac{1}{1 + \exp(-a_k)}$$

(multiclass) $$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

- Finally:

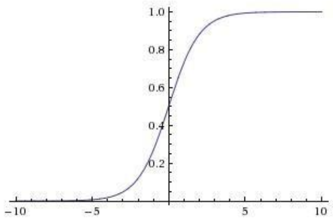
(binary)

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

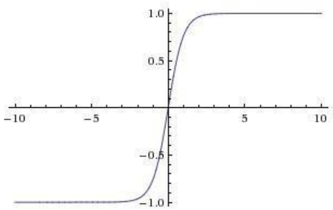
Activation functions

Sigmoid

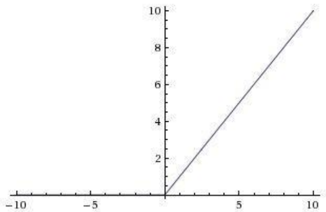
$$\sigma(x) = 1/(1 + e^{-x})$$



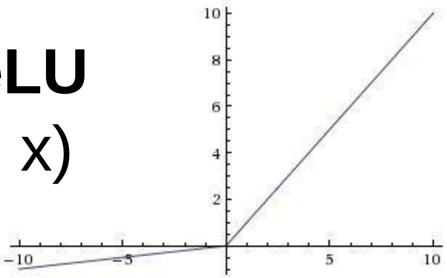
tanh tanh(x)



ReLU max(0,x)

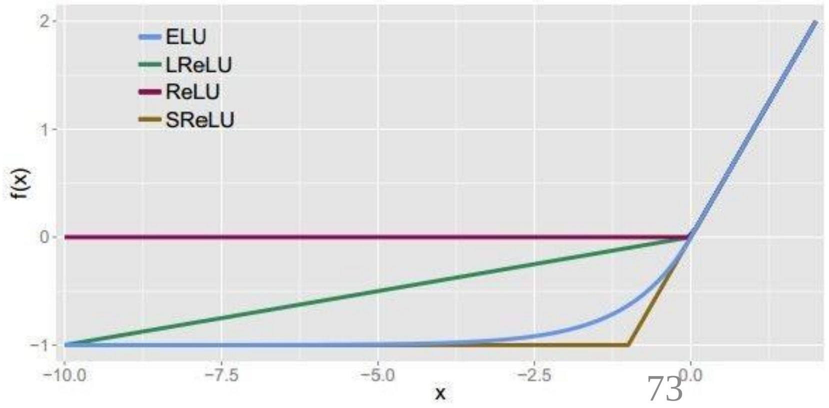


Leaky ReLU
max(0.1x, x)



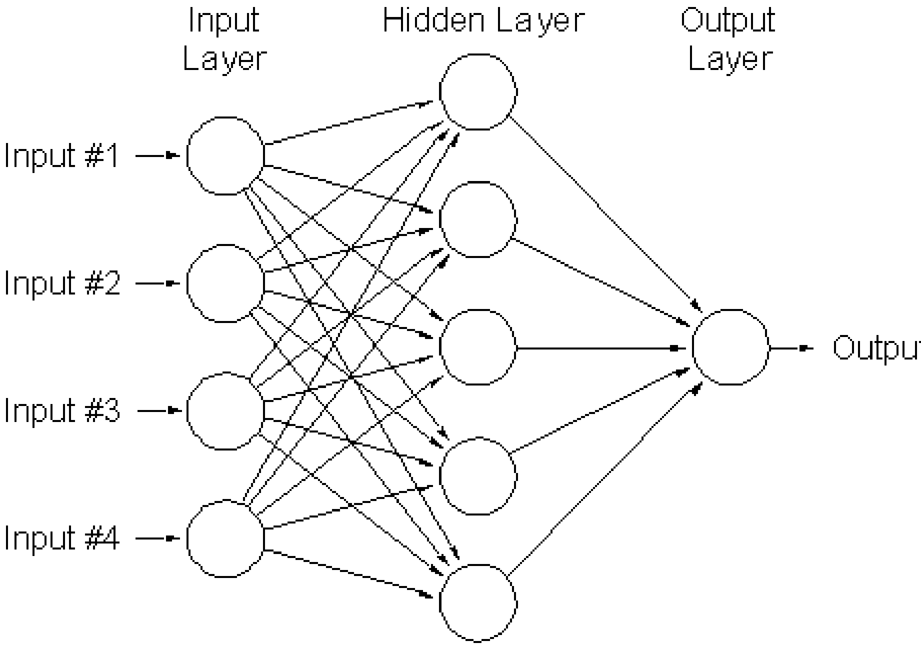
Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$



Andrej Karpathy

A multi-layer neural network

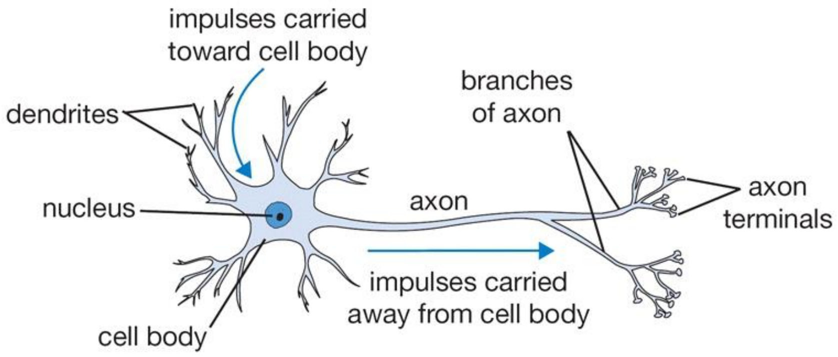


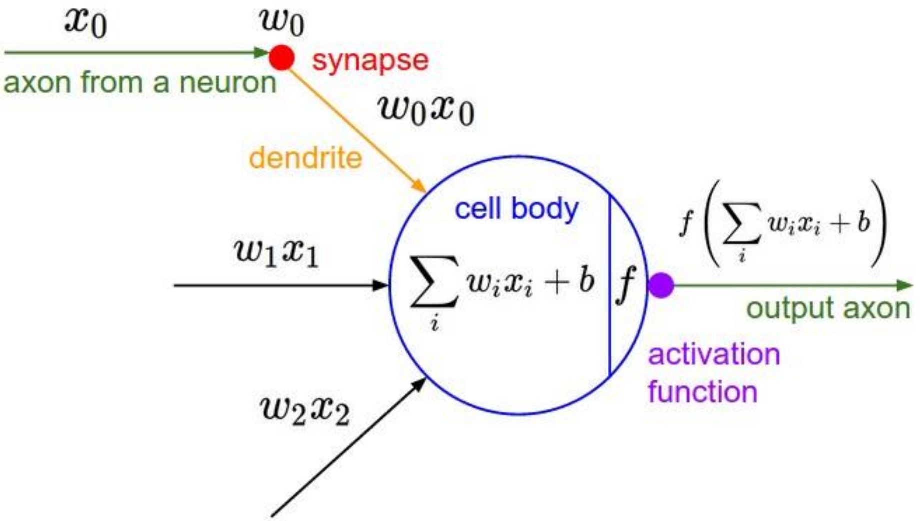
- *Nonlinear* classifier
- Can approximate any continuous function to arbitrary accuracy given sufficiently many hidden units

Lana Lazebnik

Inspiration: Neuron cells

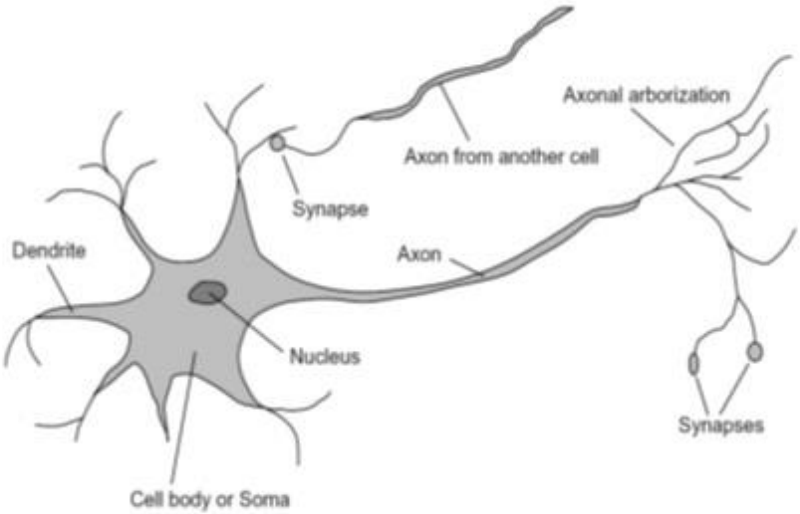
- Neurons
  - accept information from multiple inputs,
  - transmit information to other neurons.
- Multiply inputs by weights along edges
- Apply some function to the set of inputs at each node
- If output of function over threshold, neuron "fires"



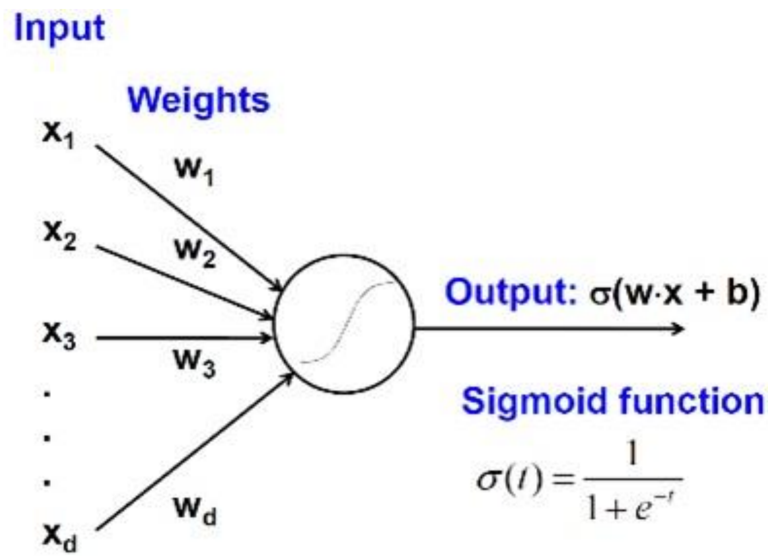


Text: HKUST, figures: Andrej Karpathy

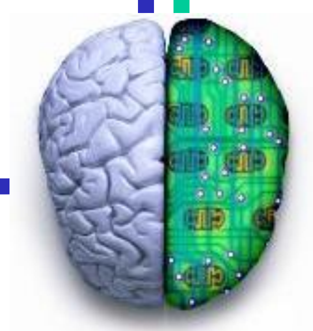
Biological analog



A biological neuron

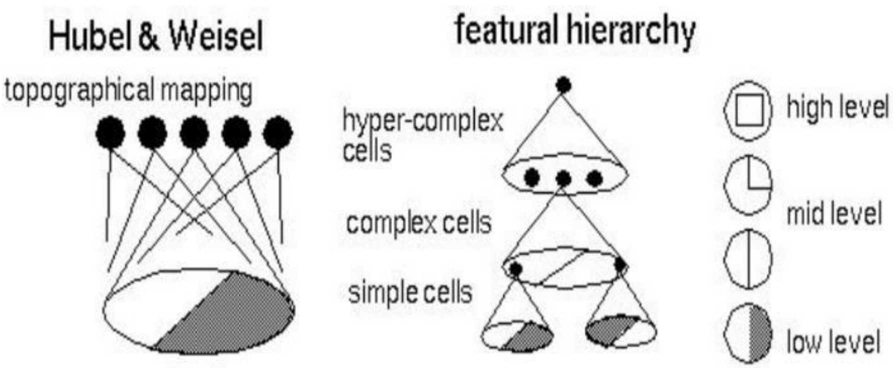


An artificial neuron

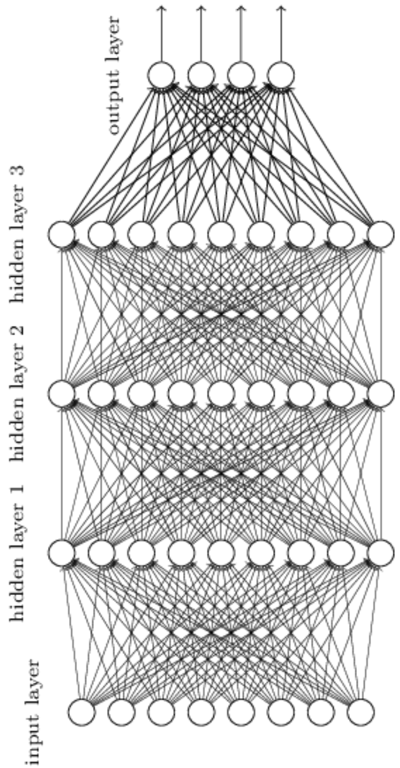


Jia-bin Huang

Biological analog



Hubel and Weisel's architecture



Multi-layer neural network

Adapted from Jia-bin Huang

Deep neural networks

- Lots of hidden layers
- Depth = power (usually)
- "With great power comes great responsibility"

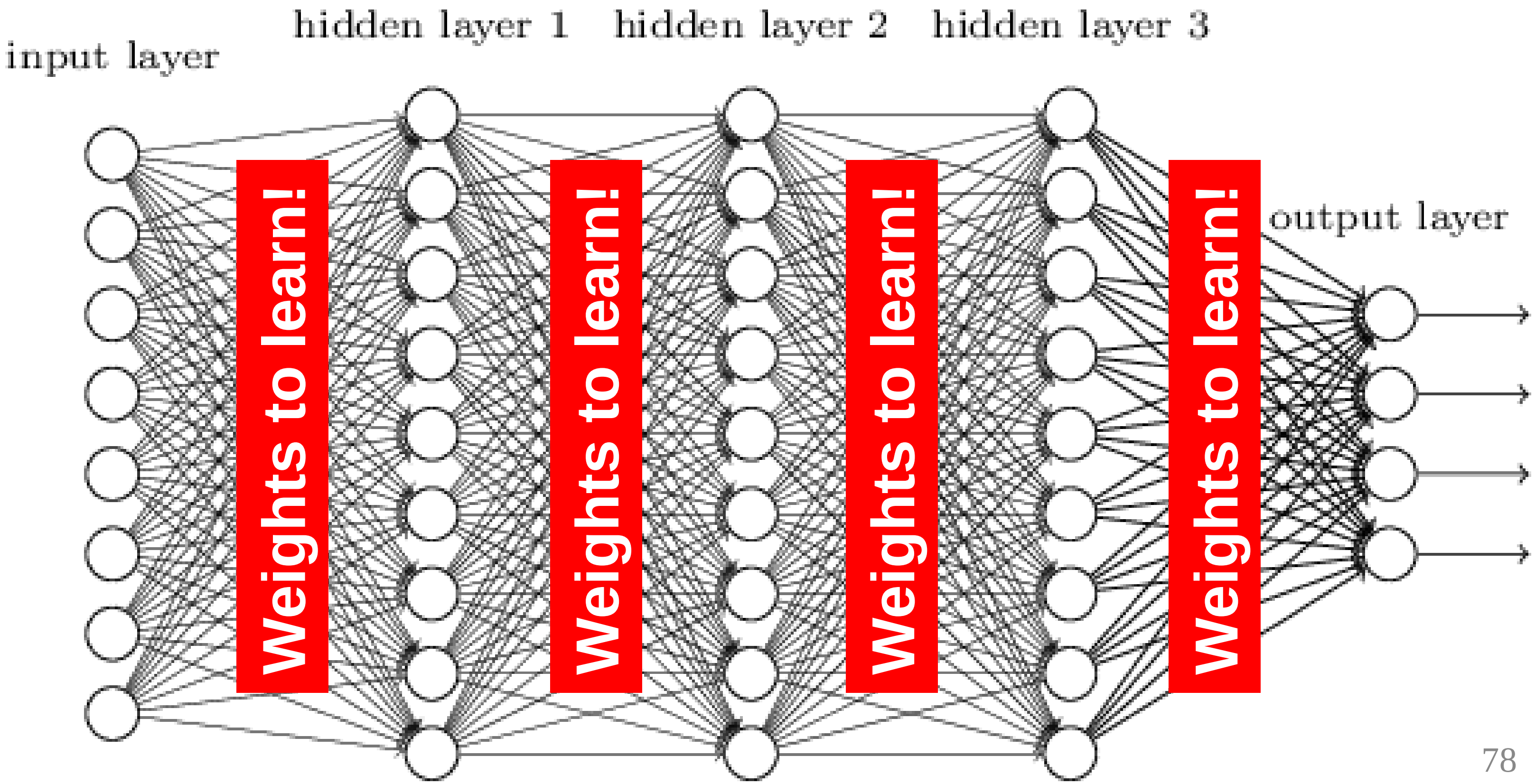
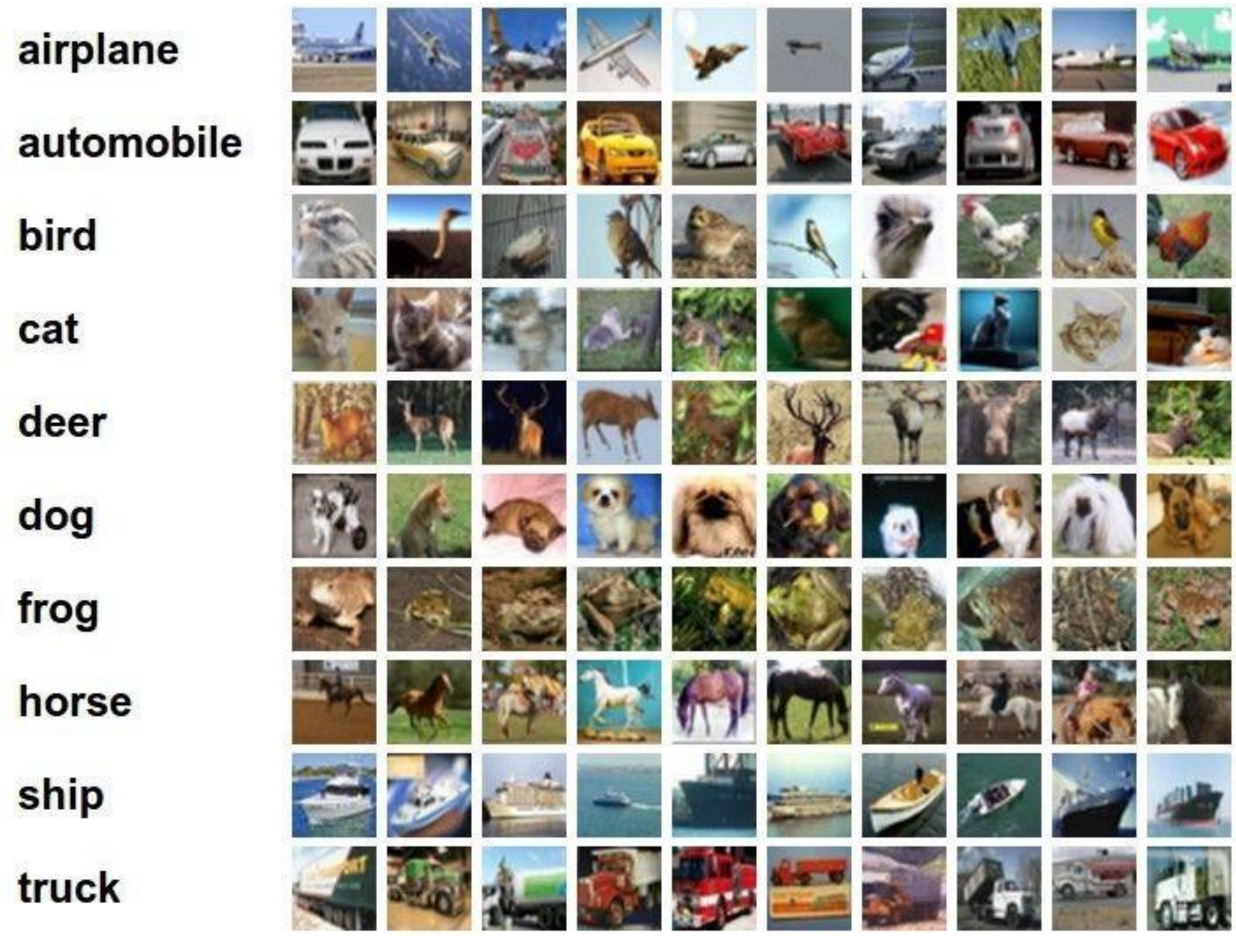


Figure from http://neuralnetworksanddeeplearning.com/chap5.html

How do we train deep neural networks?

- The goal is to find such a set of weights that allow the activations/outputs to match the desired output: $f(\mathbf{W}, \mathbf{x}_i) \sim y_i$
- Unfortunately, no closed-form solution for weights **W**, but we can express our objective
- We want to *minimize a* ***loss function*** (a function of the weights in the network), we'll do so iteratively
- For now let's simplify and assume there's a single layer of weights in the network

Classification goal



Example dataset: **CIFAR-10**
10 labels
50,000 training images
  each image is **32x32x3**
10,000 test images

Andrej Karpathy

Classification scores

$$f(x, W) = Wx$$

f(x,W)

10 numbers, indicating class scores

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Andrej Karpathy

Linear classifier

$$\boxed{f(x,W)} = \boxed{W}\boxed{x} \quad \boxed{(+b)}$$

3072x1

10x1

10x1

10x3072

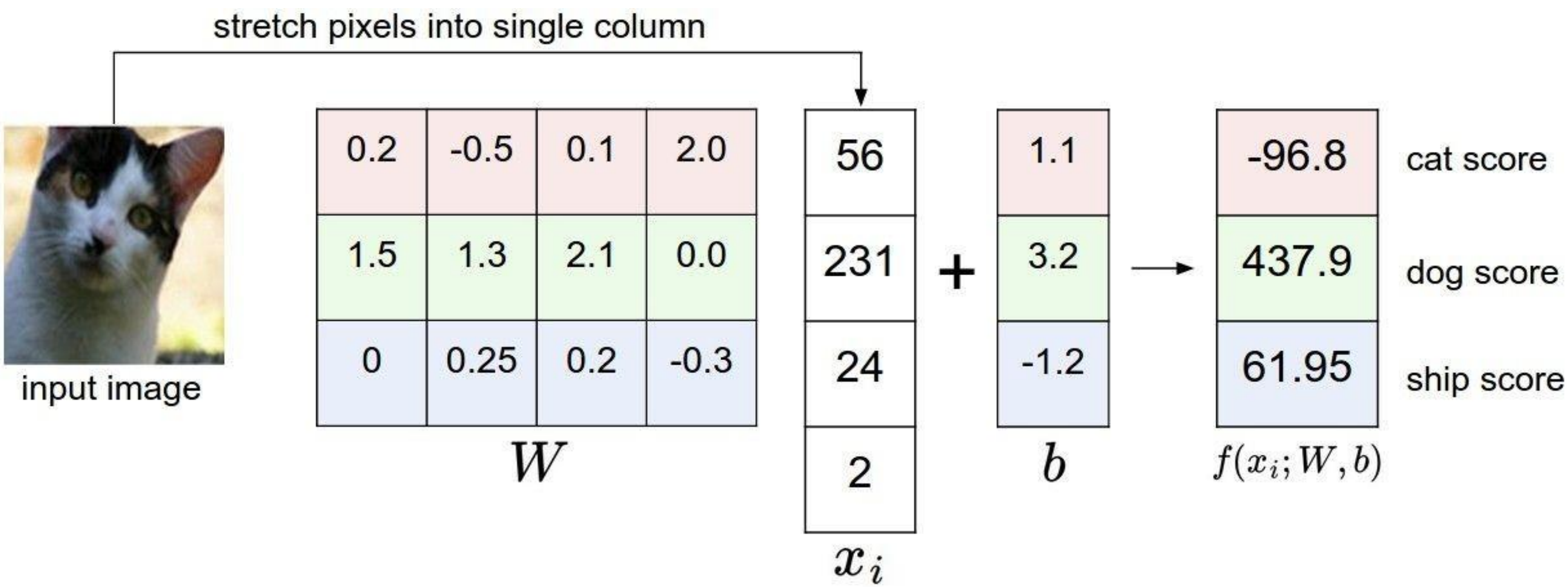
[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

parameters, or “weights”

Linear classifier

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Andrej Karpathy

Linear classifier

Going forward: Loss function/Optimization

			
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.

2. Come up with a way of efficiently finding the parameters that minimize the loss function **(optimization)**

Linear classifier

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

Want: $s_{y_i} >= s_j + 1$, for $j != y_i$
i.e. $s_j - s_{y_i} + 1 <= 0$

If true, loss is 0
If false, loss is magnitude of violation

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9		

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 5.3 + 1)
+ max(0, 5.6 + 1)
= 6.3 + 6.6
= 12.9

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

and the full training loss is the mean over all examples in the training data:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= 15.8 / 3 = **5.3**

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Weight Regularization

λ = regularization strength
(hyperparameter)

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

L2 regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

L1 regularization

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

Dropout (will see later)

Adapted from Andrej Karpathy

Another loss: Softmax (cross-entropy)

scores = unnormalized log probabilities of the classes

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

where $s = f(x_i; W)$

cat	**3.2**
car	5.1
frog	-1.7

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$\underbrace{L_i = -\log \underbrace{P(Y = y_i | X = x_i)}_{\text{maximize}}}_{\text{minimize}}$$

Adapted from Andrej Karpathy

Another loss: Softmax (cross-entropy)

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

	unnormalized log probabilities	exp →	unnormalized probabilities	normalize →	probabilities	
cat	**3.2**		**24.5**		**0.13**	→ L_i = -log(0.13) = **0.89**
car	5.1		164.0		0.87	
frog	-1.7		0.18		0.00	

unnormalized log probabilities

probabilities

Adapted from Andrej Karpathy

Other losses

- Triplet loss (Schroff, FaceNet)

$$\sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

a denotes anchor
p denotes positive
n denotes negative

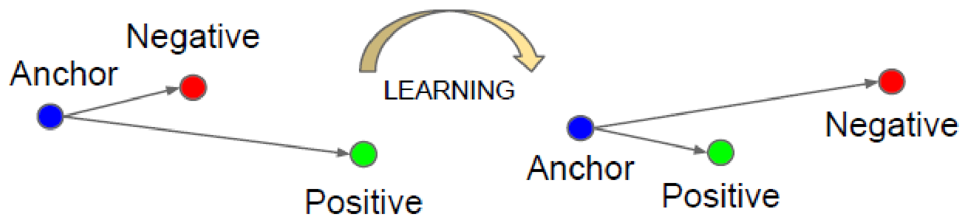


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

- Anything you want!

How to minimize the loss function?

Andrej Karpathy

How to minimize the loss function?

In 1-dimension, the derivative of a function is:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple dimensions, the **gradient** is the vector of (partial derivatives):

That is, for $f\colon \mathbf{R}^n \to \mathbf{R}$, its gradient $\nabla f\colon \mathbf{R}^n \to \mathbf{R}^n$ is defined at the point $\mathrm{p} = (x_1, \ldots, x_n)$ in *n*-dimensional space as the vector:[b]

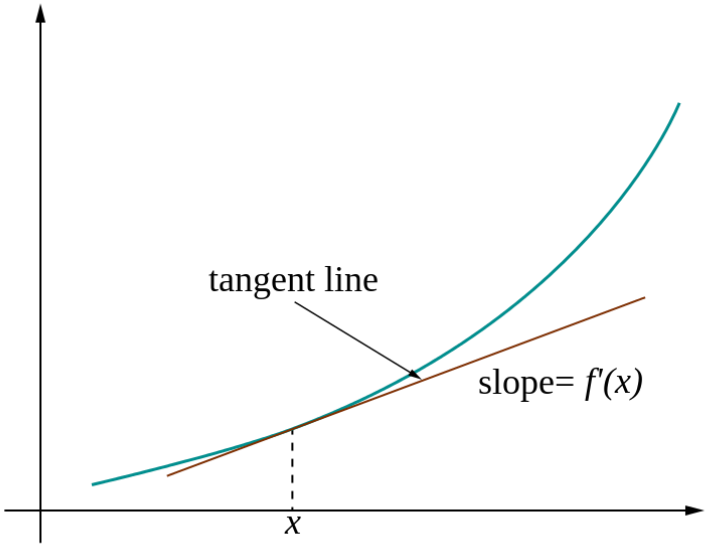
$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}.$$

The nabla symbol ∇, written as an upside-down triangle and pronounced "del", denotes the vector differential operator.

Adapted from Andrej Karpathy, definition/equation from https://en.wikipedia.org/wiki/Gradient

Loss gradients

- Different notations: $$\frac{\partial E}{\partial w_{ji}^{(1)}} \qquad \nabla_W L$$
- i.e. how does the loss change as a function of the weights
- We want to change weights in a way that makes the loss decrease as fast as possible



Gradient descent

- We'll update weights
- Move in direction opposite to gradient:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \overbrace{\nabla E(\mathbf{w}^{(\tau)})}$$

Time

Learning rate

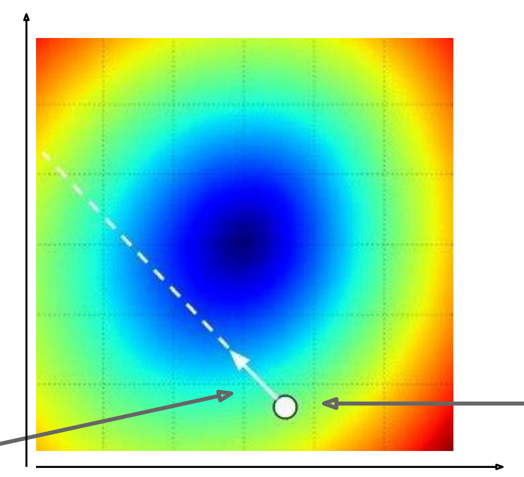


Figure from Andrej Karpathy

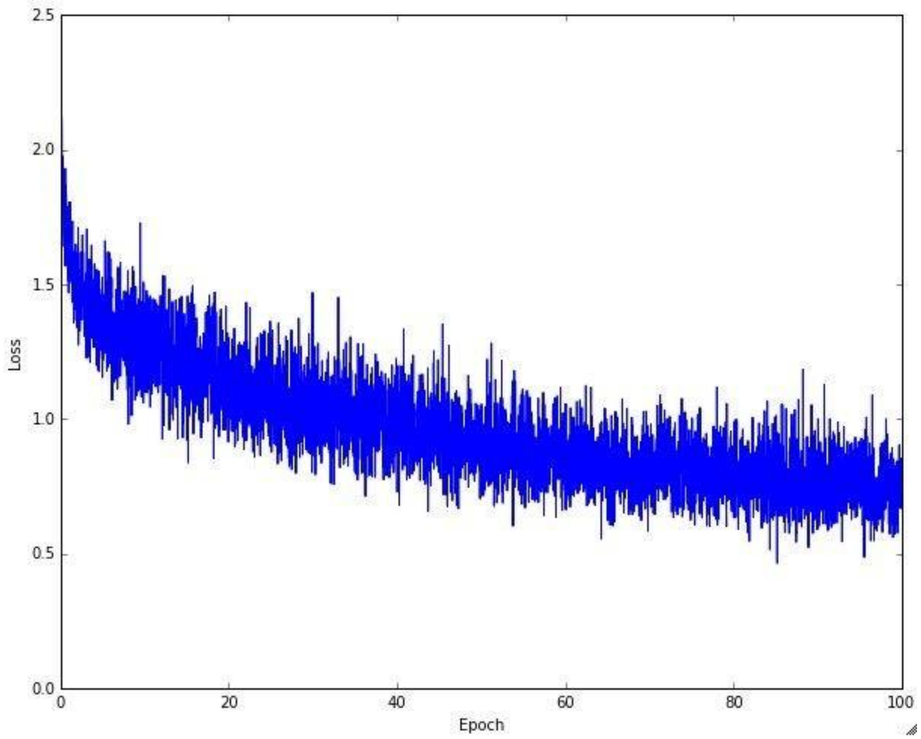
Gradient descent

- Iteratively *subtract* the gradient with respect to the model parameters (w)
- I.e. we're moving in a direction opposite to the gradient of the loss
- I.e. we're moving towards *smaller* loss

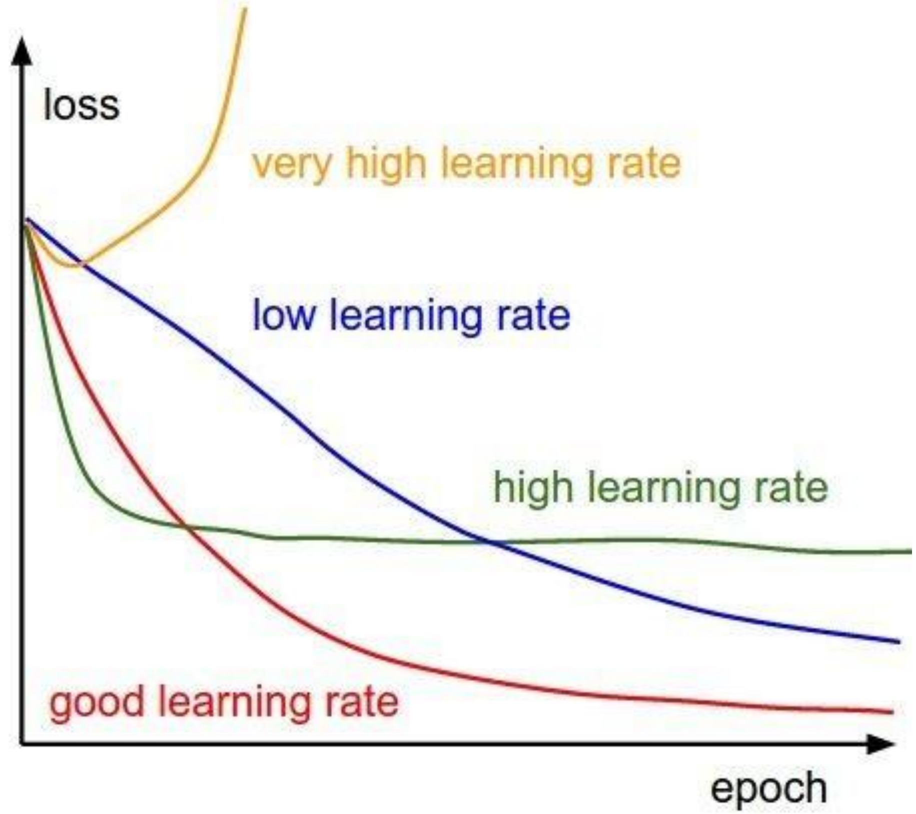
How to compute the loss/gradient?

- In classic gradient descent, we compute the gradient from the loss for all training examples
- Mini-batch gradient descent: Only use *some* of the data for each gradient update, cycle through training examples multiple times
  - Each time we’ve cycled through all of them once is called an ‘epoch’
  - Allows faster training (e.g. on GPUs), parallelization
  - Some benefits for learning due to randomness

Learning rate selection



The effects of step size (or “learning rate”)



https://www.deeplearning.ai/ai-notes/optimization/

Gradient descent in multi-layer nets

- We'll update weights
- Move in direction opposite to gradient:

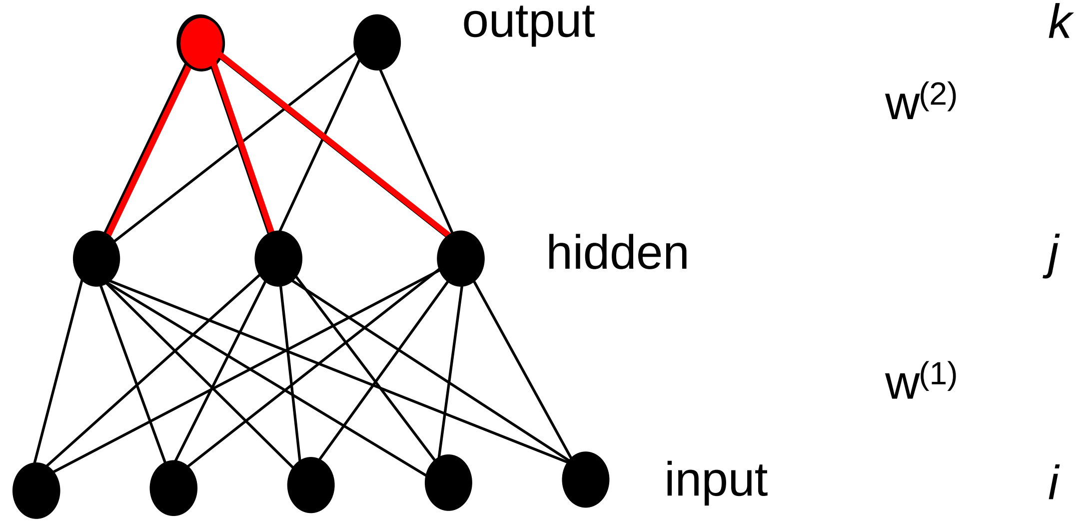
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

- How to update the weights at all layers?
- Answer: backpropagation of error from higher layers to lower layers

Backpropagation: Graphic example

First calculate error of output units and use this to change the top layer of weights.

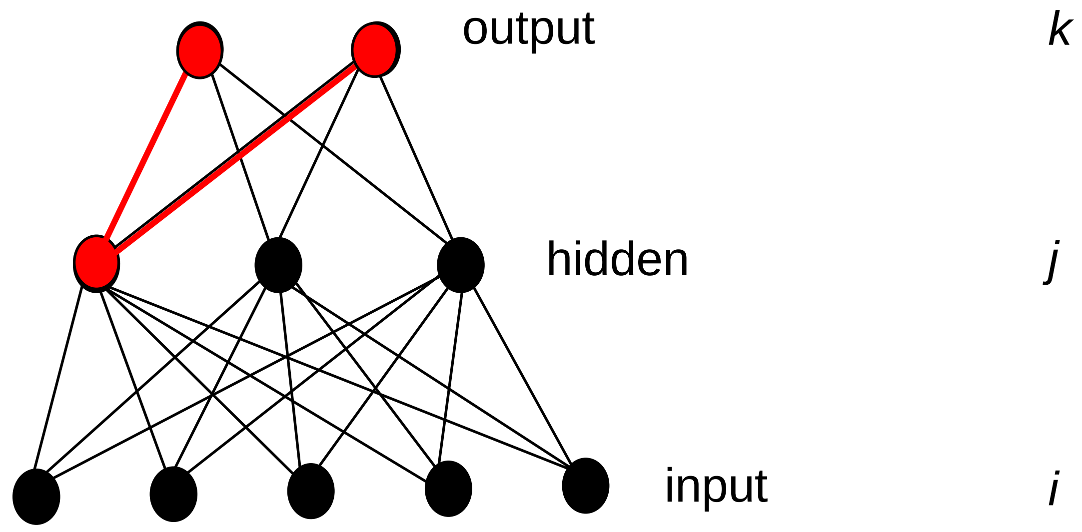
Calculate how to update weights into *j* (update at end of iter)



Adapted from Ray Mooney, equations from Chris Bishop

Backpropagation: Graphic example

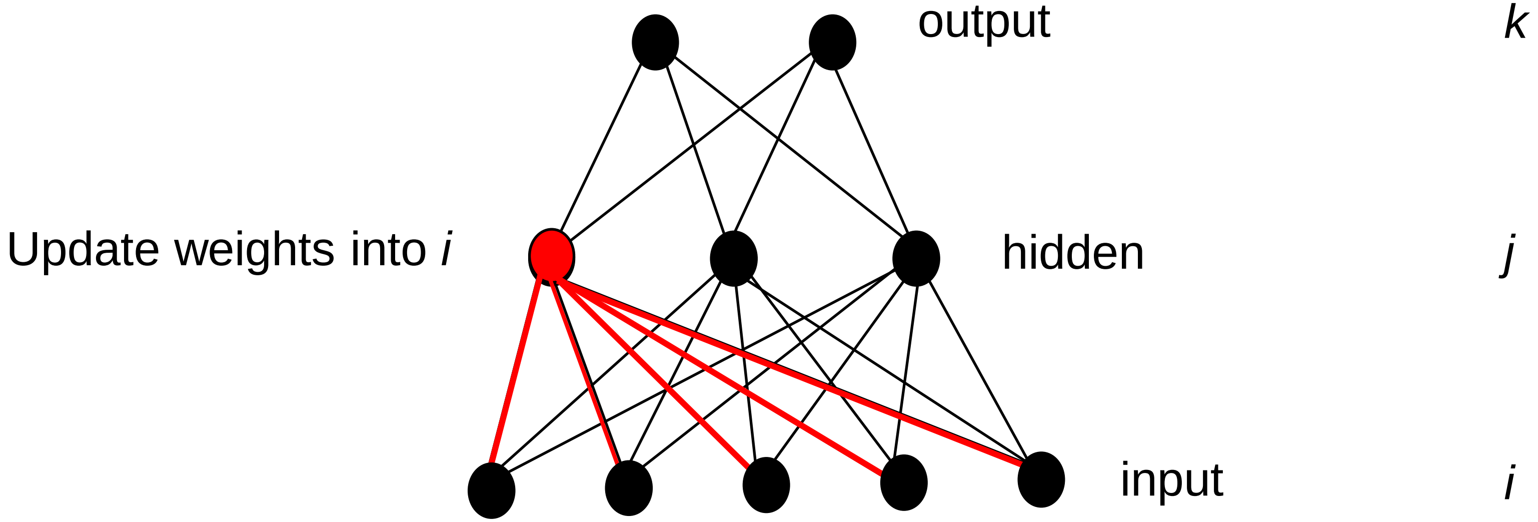
Next calculate error for hidden units based on errors on the output units it feeds into.



Adapted from Ray Mooney, equations from Chris Bishop

Backpropagation: Graphic example

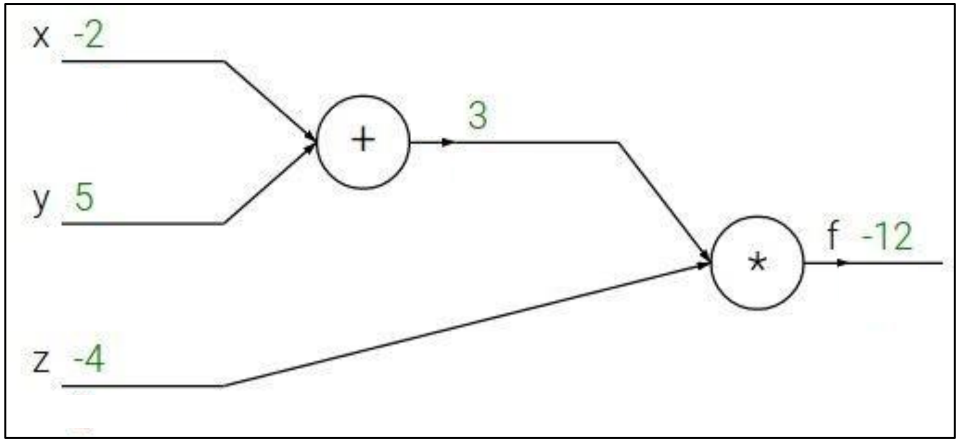
Finally update bottom layer of weights based on errors calculated for hidden units.



Adapted from Ray Mooney, equations from Chris Bishop

$$f(x, y, z) = (x + y)z$$

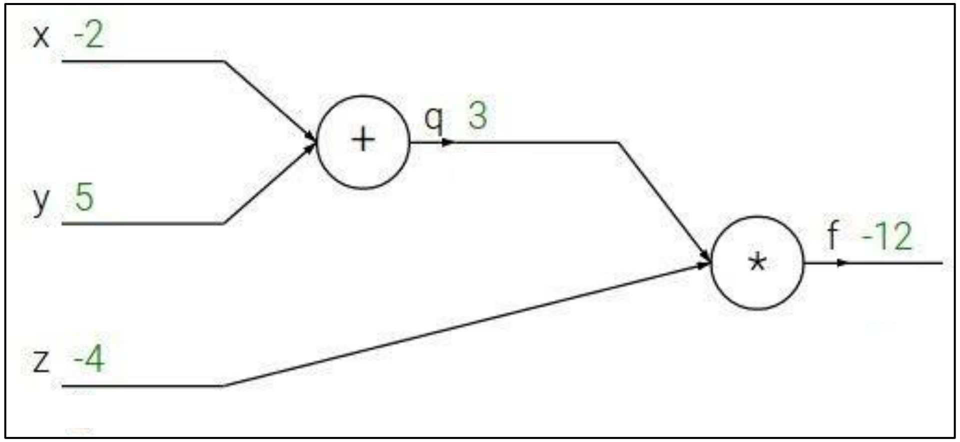
e.g. x = -2, y = 5, z = -4



Andrej Karpathy

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

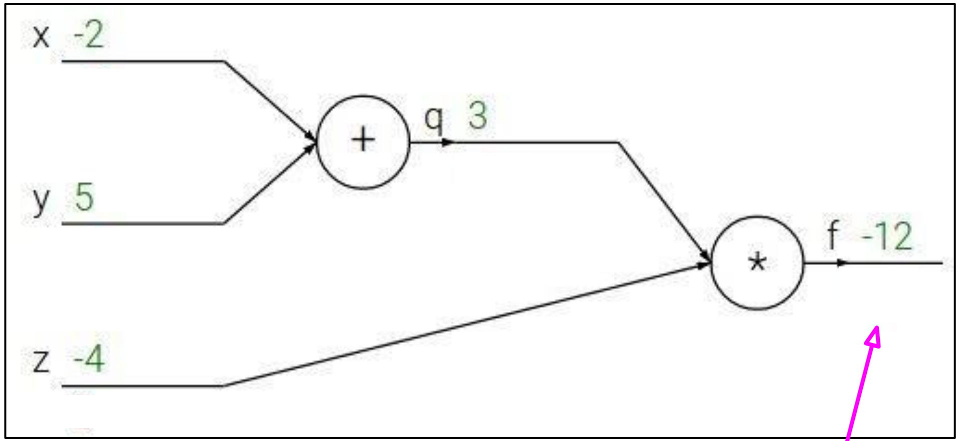
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



$$\frac{\partial f}{\partial f}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy

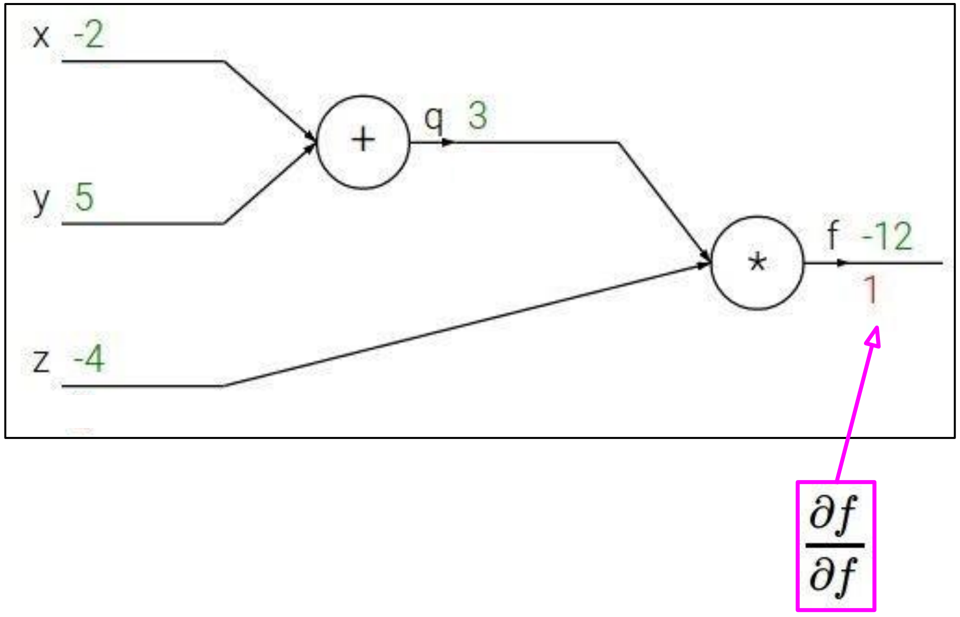
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Andrej Karpathy

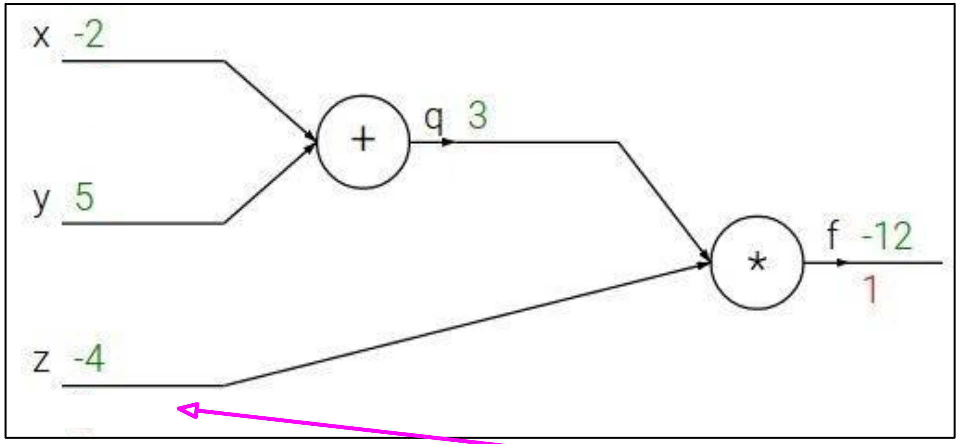
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

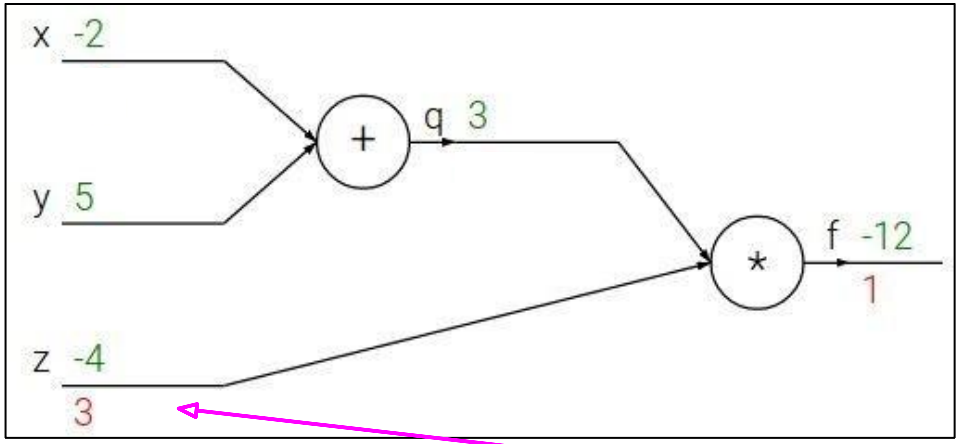
Andrej Karpathy

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

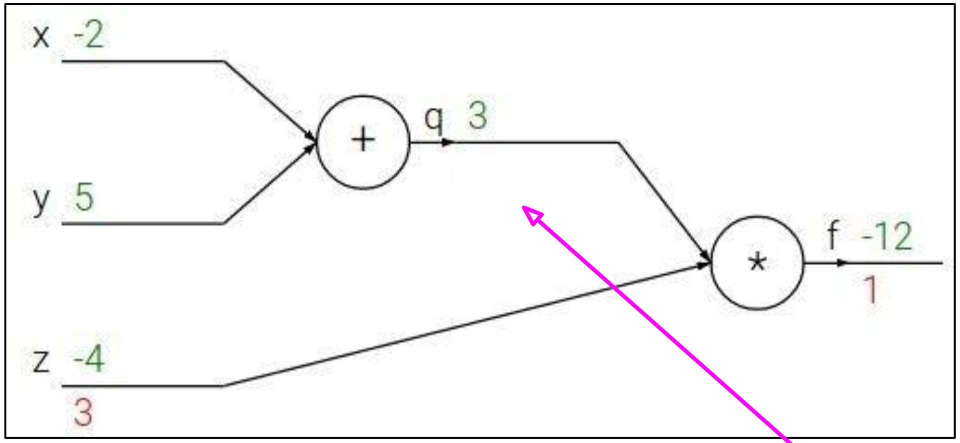


$$\frac{\partial f}{\partial z}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

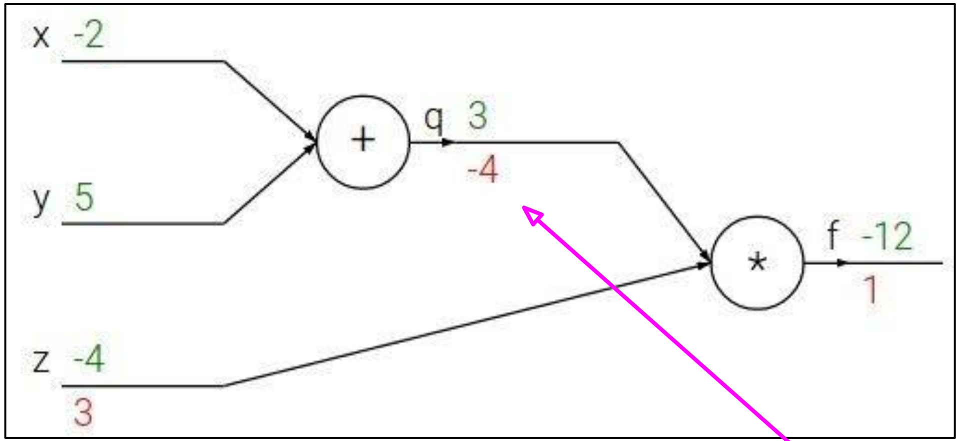
$$\boxed{\frac{\partial f}{\partial q}}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\boxed{\frac{\partial f}{\partial q}}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy

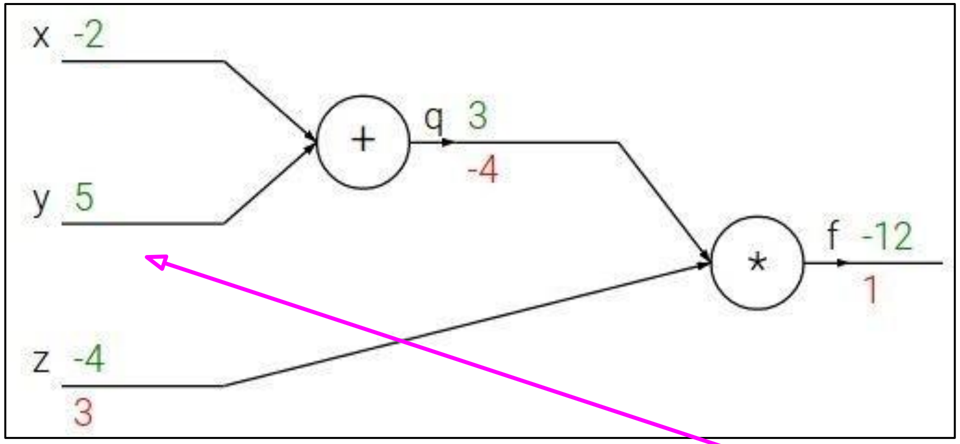
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\boxed{\frac{\partial f}{\partial y}}$

Andrej Karpathy

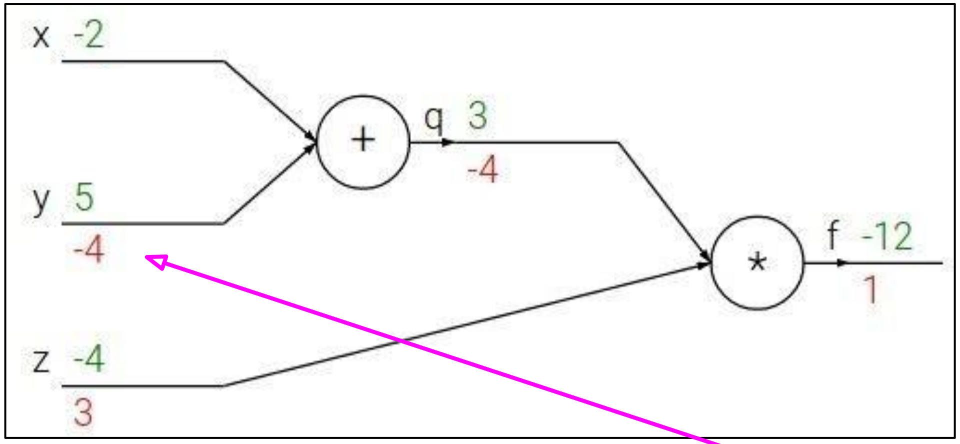
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Andrej Karpathy

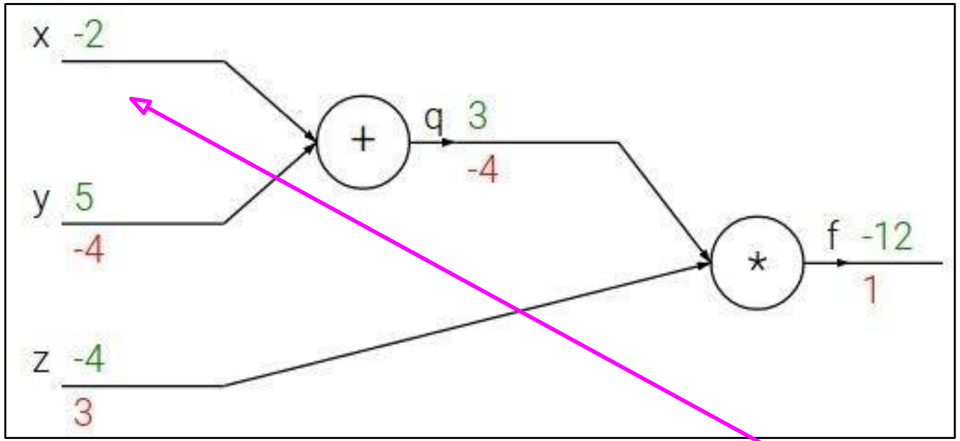
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial x}}$$

Andrej Karpathy

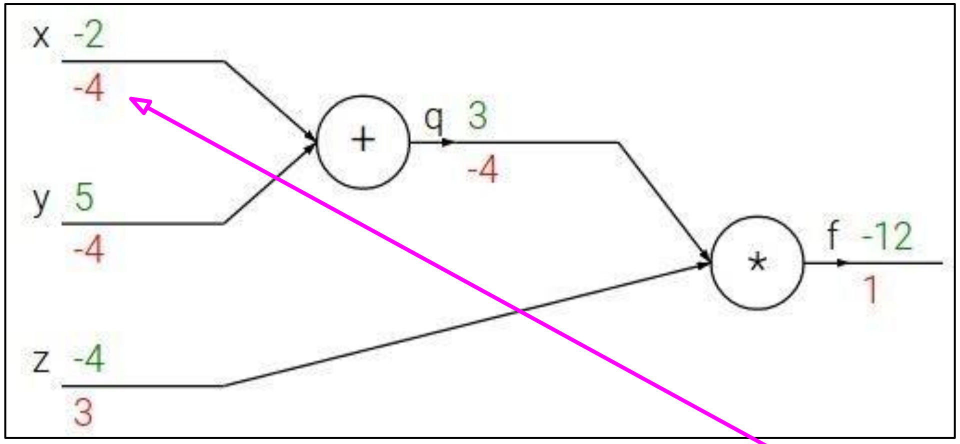
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

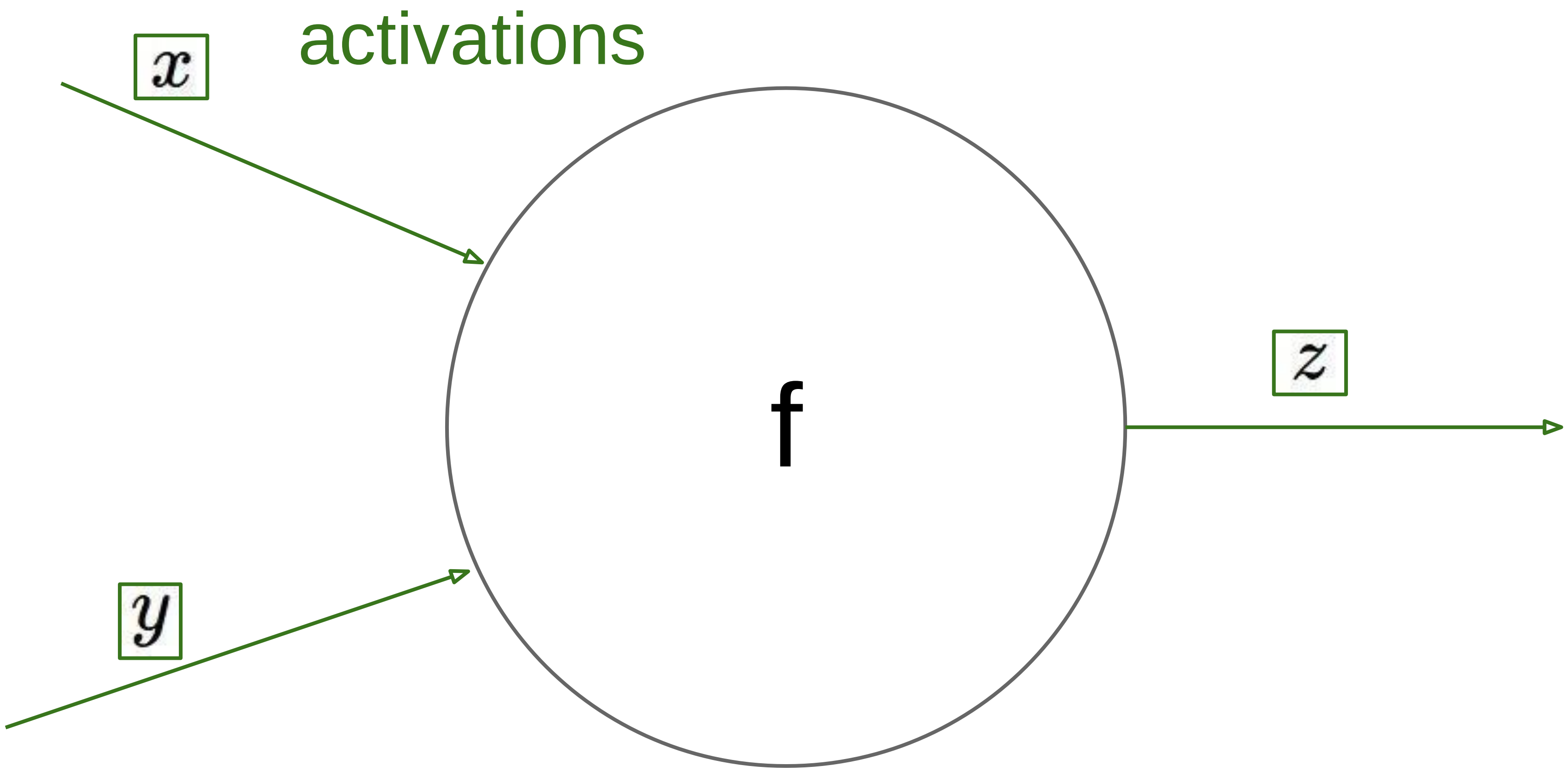
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



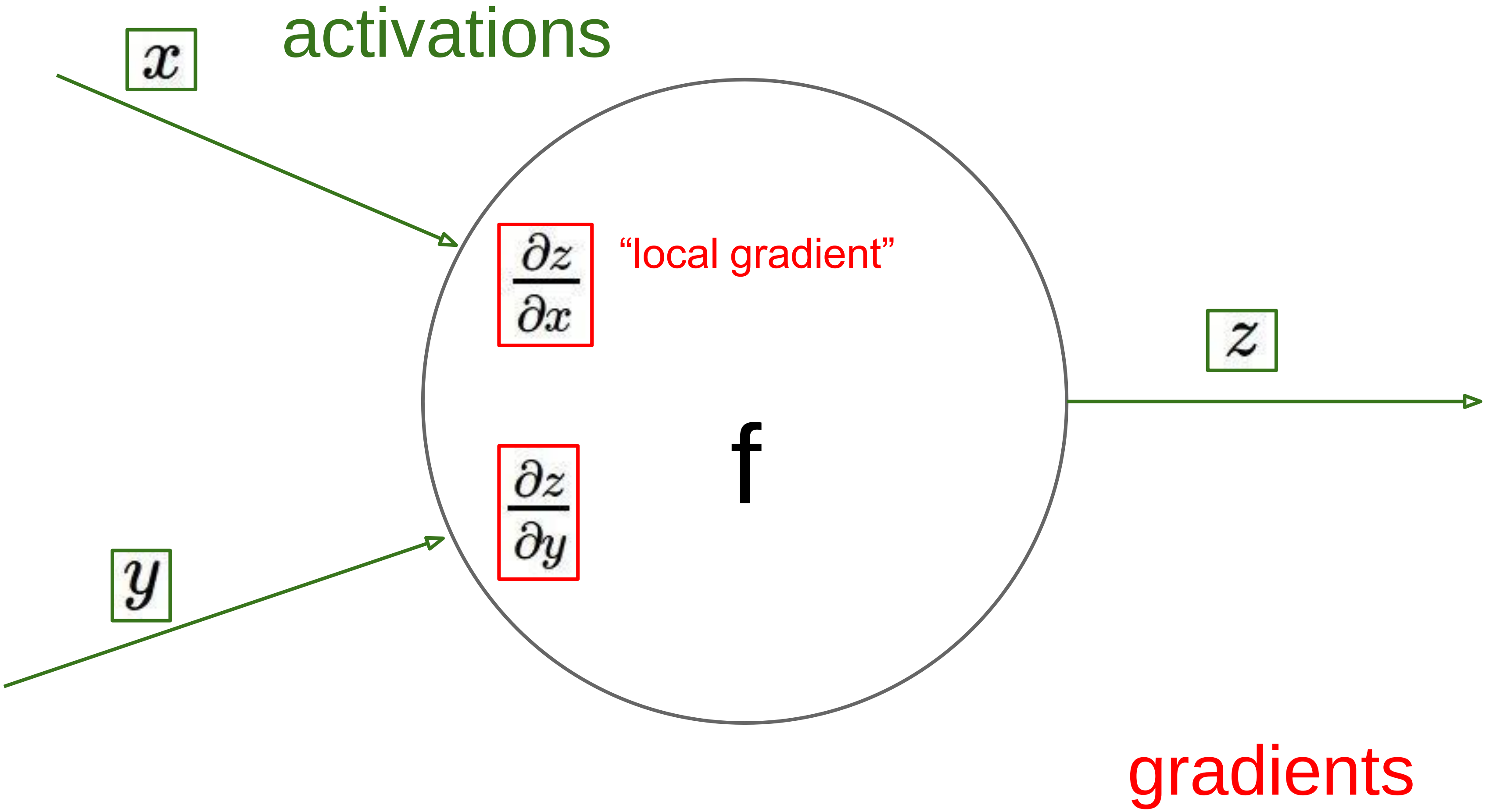
$$\boxed{\frac{\partial f}{\partial x}}$$

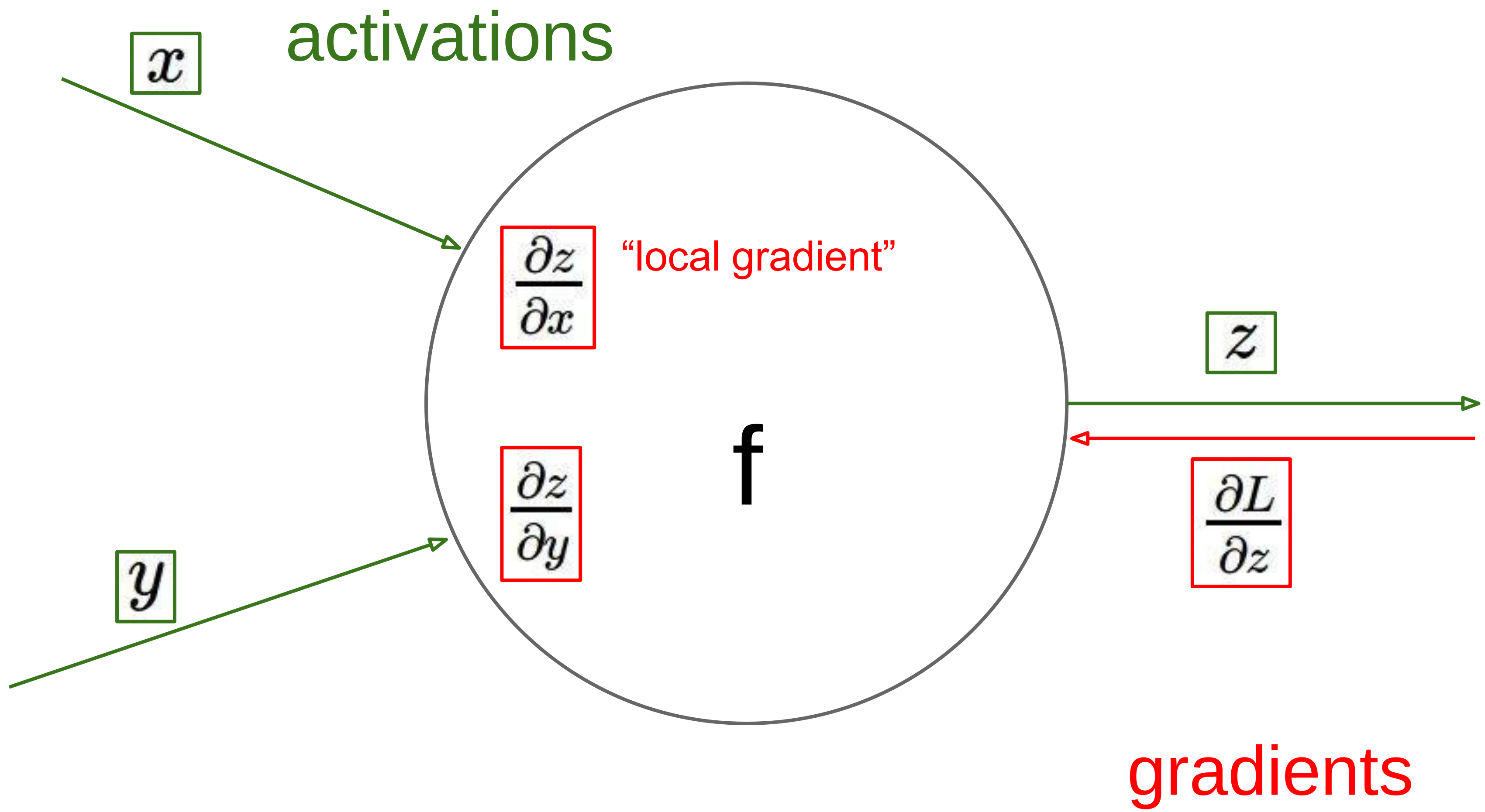
Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

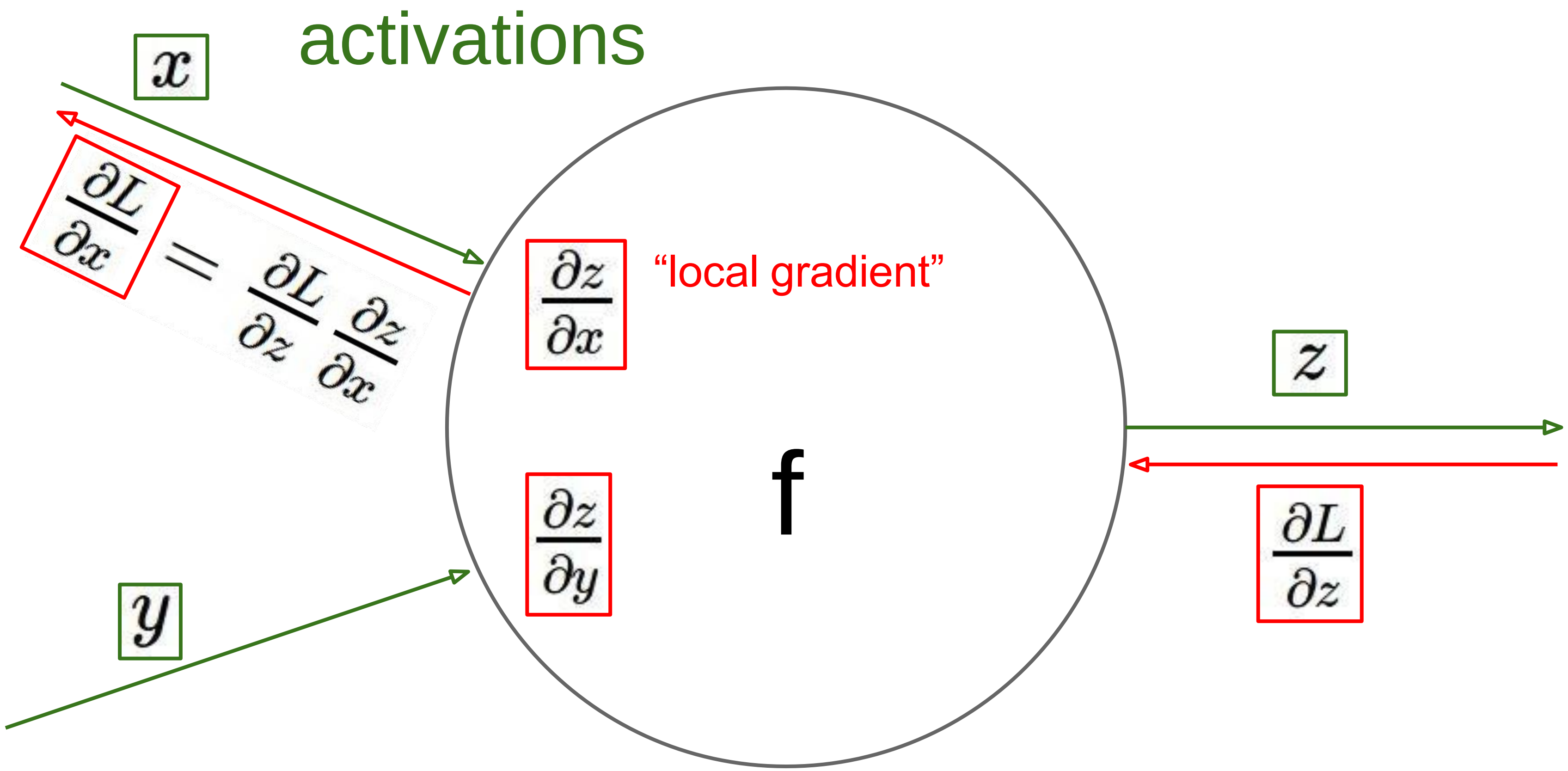


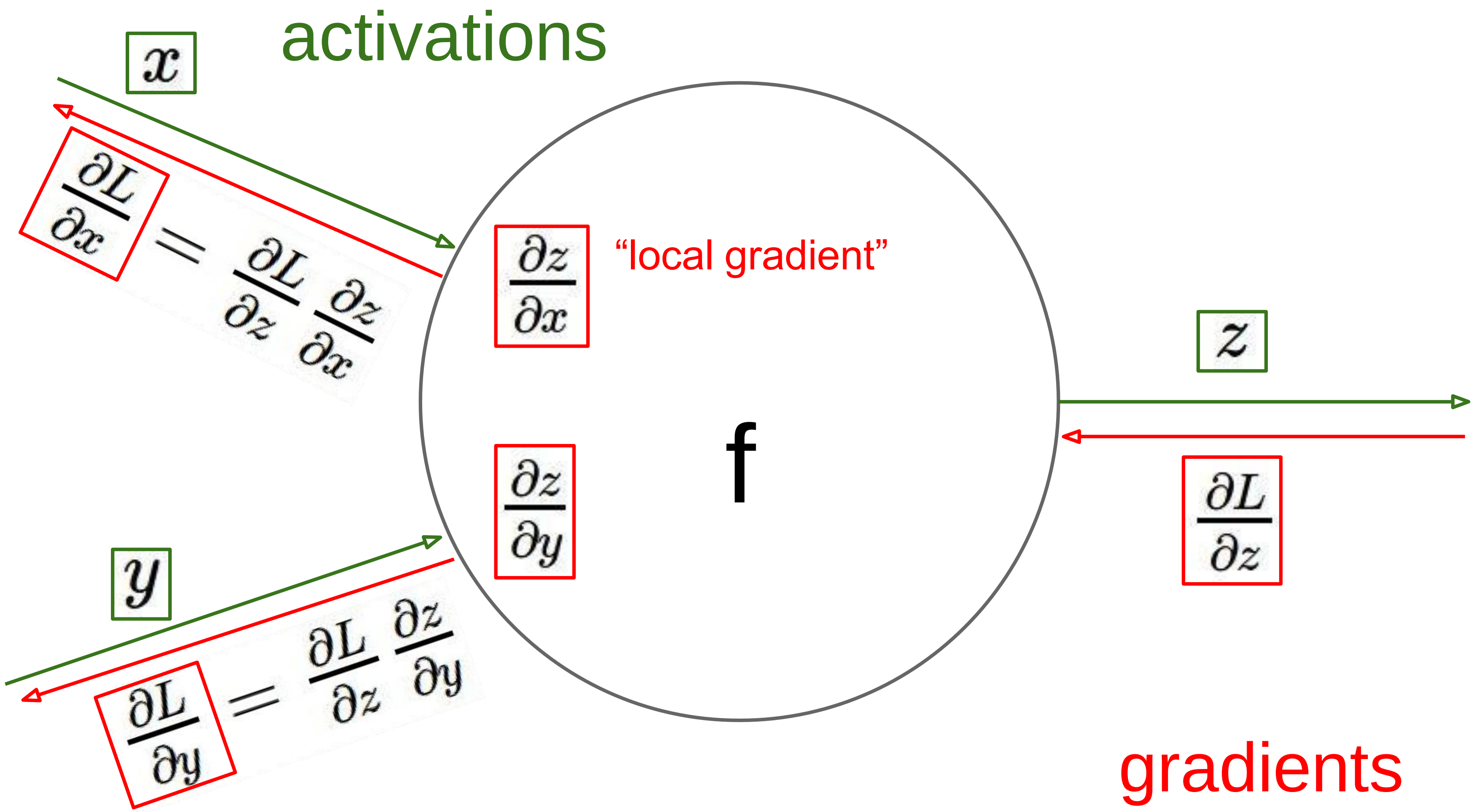
Andrej Karpathy


activations
x
$\frac{\partial z}{\partial x}$
"local gradient"
z
f
$\frac{\partial z}{\partial y}$
y
gradients



Andrej Karpathy

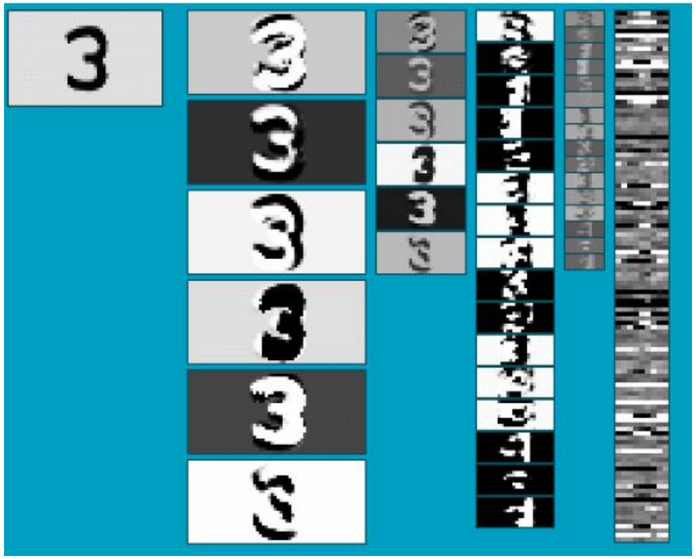

activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial y}$
f
z
y
$\frac{\partial L}{\partial z}$
gradients

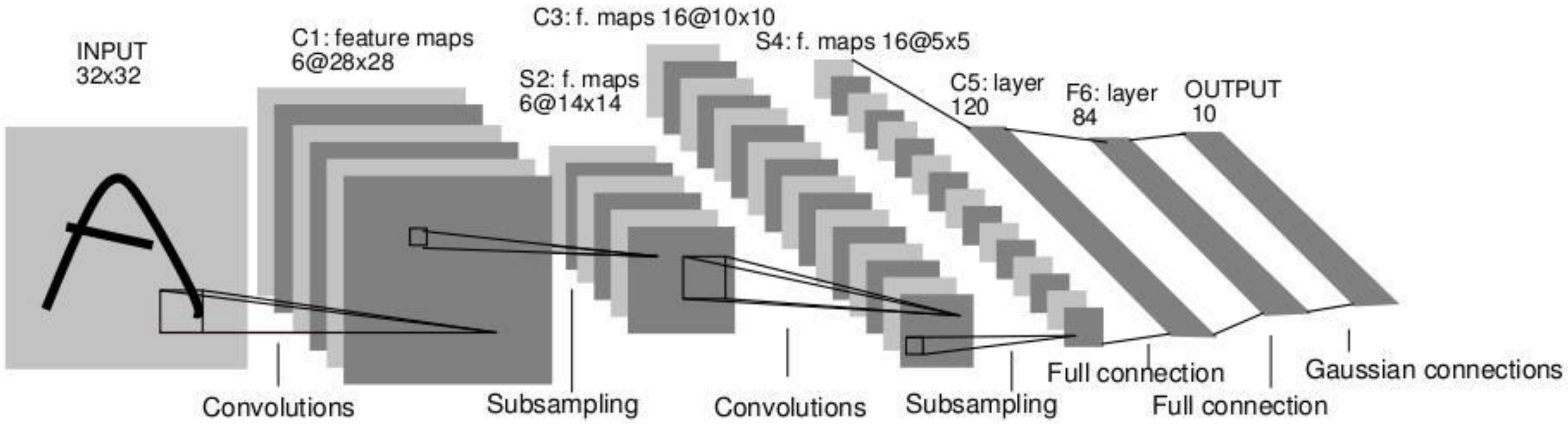


Andrej Karpathy

Convolutional Neural Networks (CNN)

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant, *more abstract* features
- Classification layer at the end





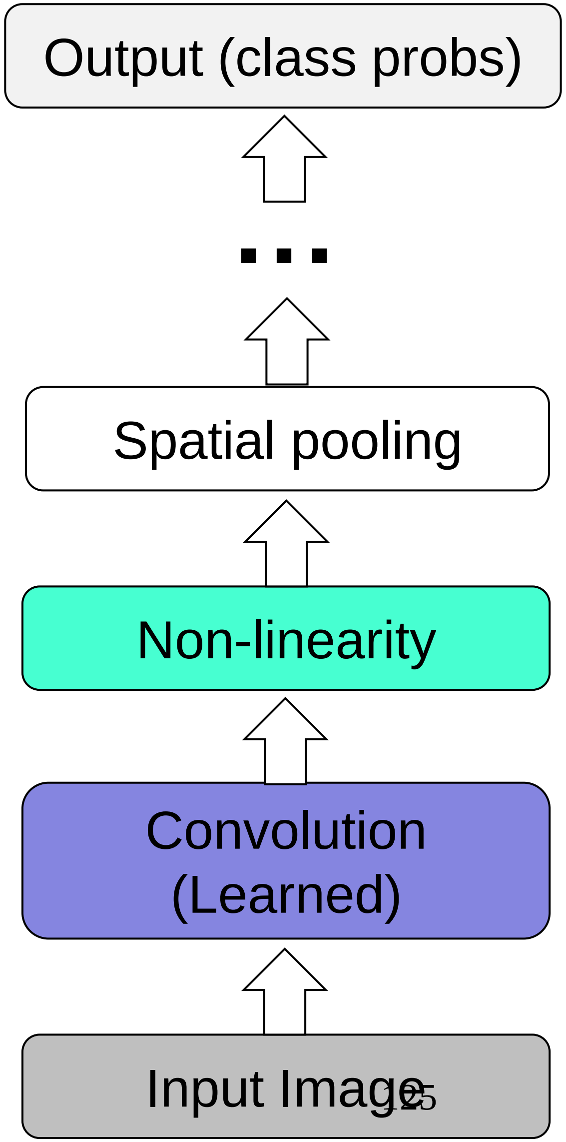
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, [Gradient-based learning applied to document recognition](), Proceedings of the IEEE 86(11): 2278–2324, 1998.

Adapted from Rob Fergus

Convolutional Neural Networks (CNN)

- Feed-forward feature extraction:
  1. Convolve input with learned filters
  2. Apply non-linearity
  3. Spatial pooling (downsample)

- Supervised training of convolutional filters by back-propagating classification error

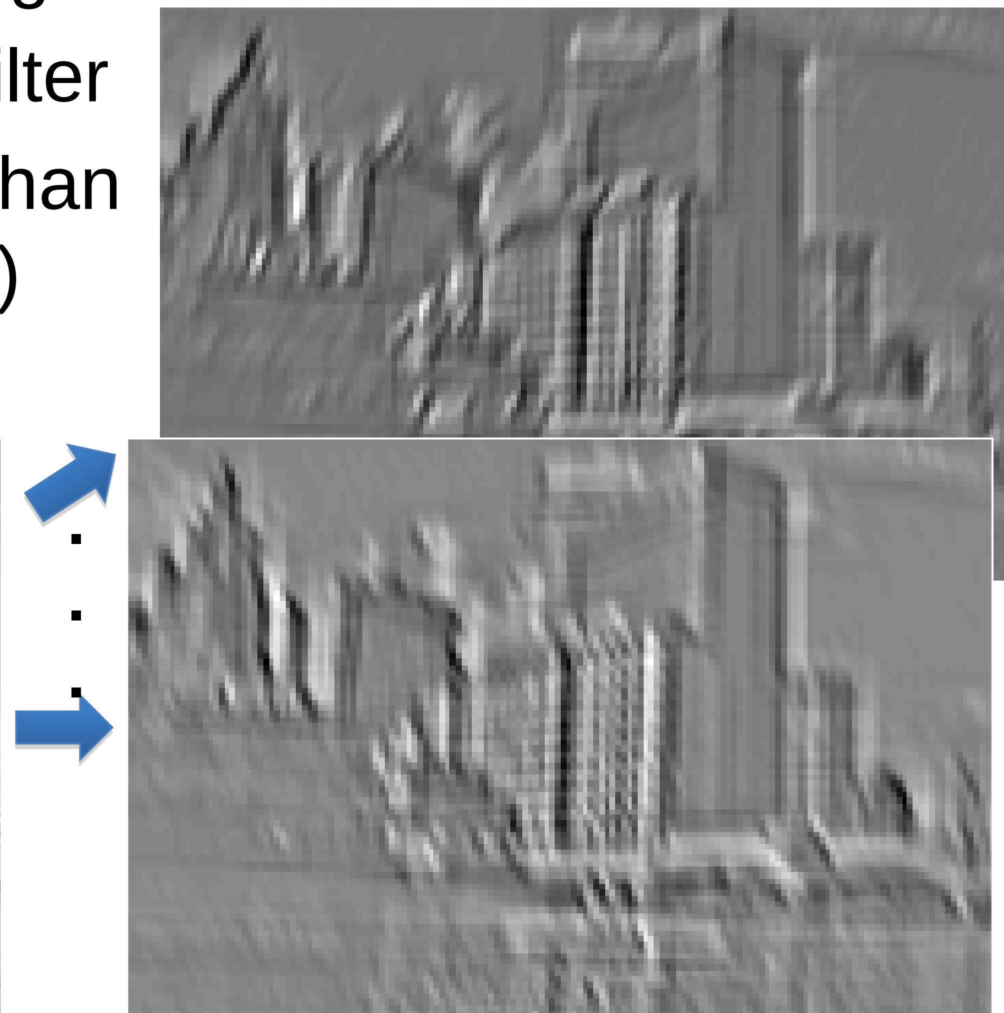


Adapted from Lana Lazebnik

1. Convolution

- Apply learned filter weights
- One feature map per filter
- Stride can be greater than 1 (faster, less memory)

Input

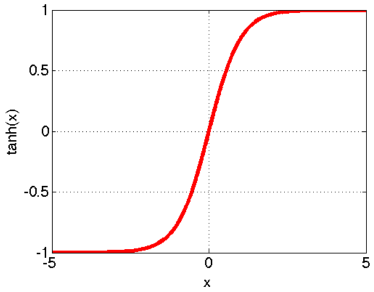


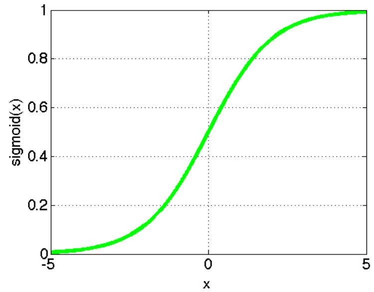
Feature Map

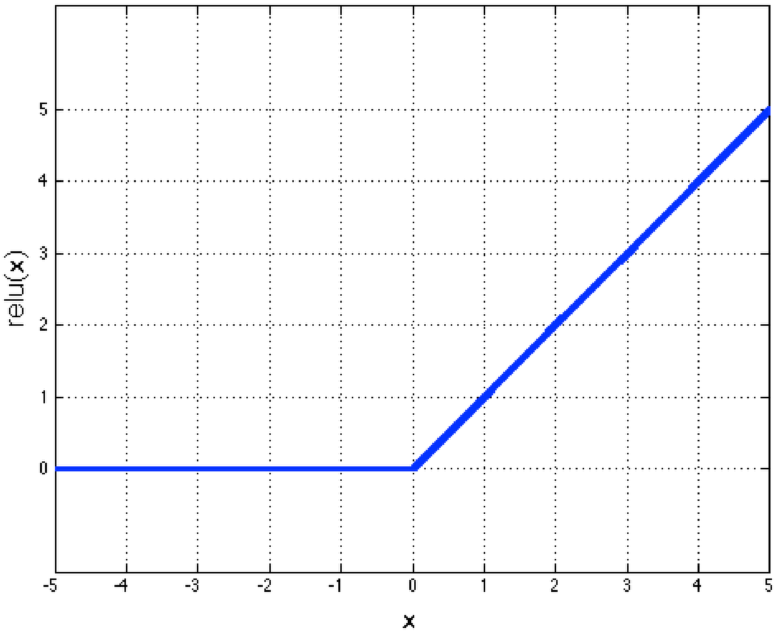
Adapted from Rob Fergus

2. Non-Linearity

- Per-element (independent)
- Some options:
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - Rectified linear unit (ReLU)
    - Avoids saturation issues



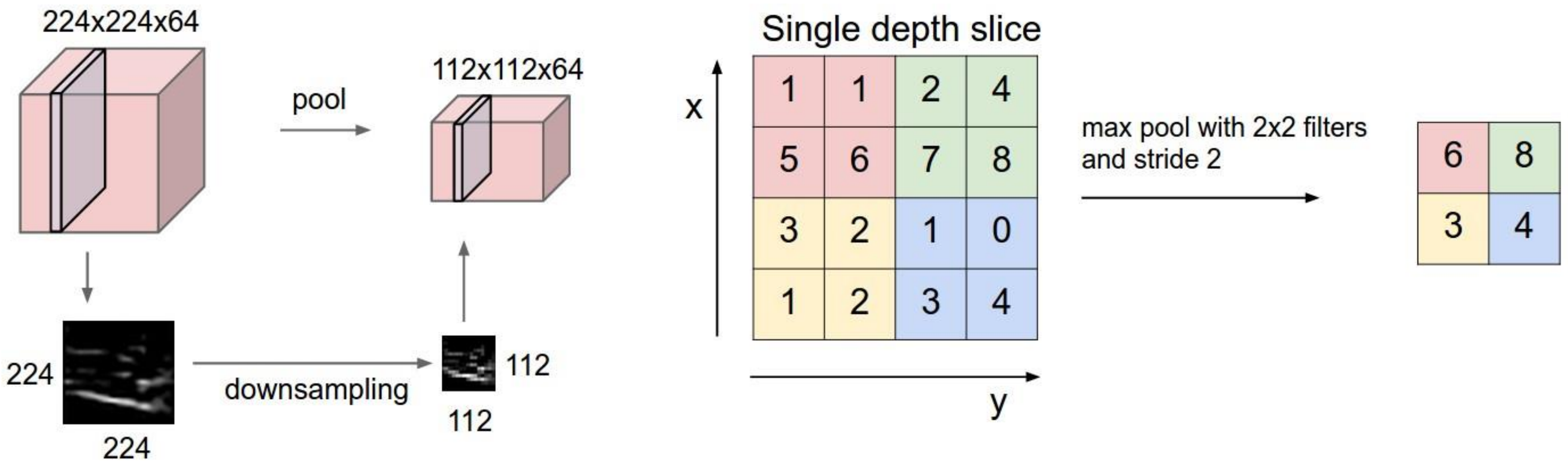




Adapted from Rob Fergus

3. Spatial Pooling

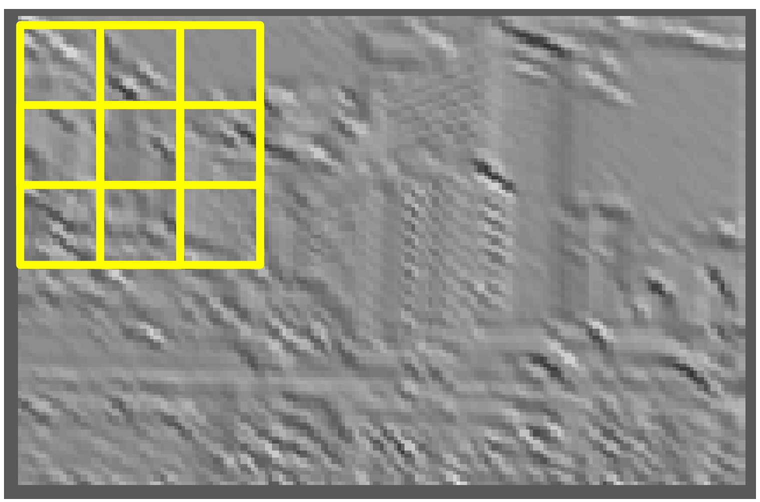
- Sum or max over non-overlapping / overlapping regions



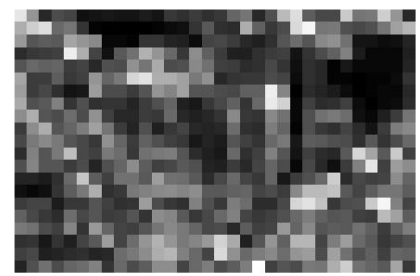
Rob Fergus, figure from Andrej Karpathy

3. Spatial Pooling

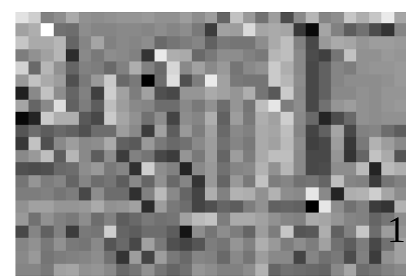
- Sum or max over non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (neurons see more of input)



Max



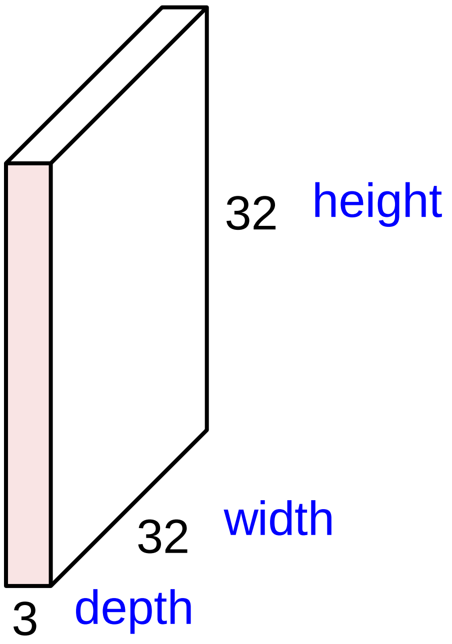
Sum



Adapted from Rob Fergus

Convolutions: More detail

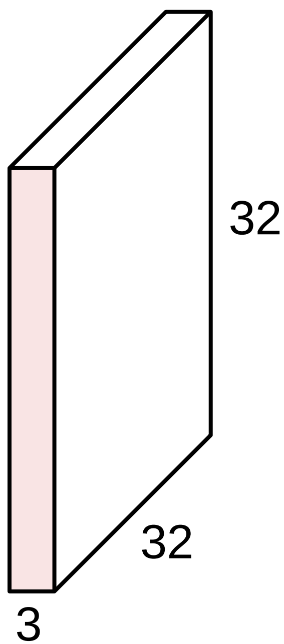
32x32x3 image



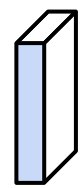
Andrej Karpathy

Convolutions: More detail

32x32x3 image



5x5x3 filter

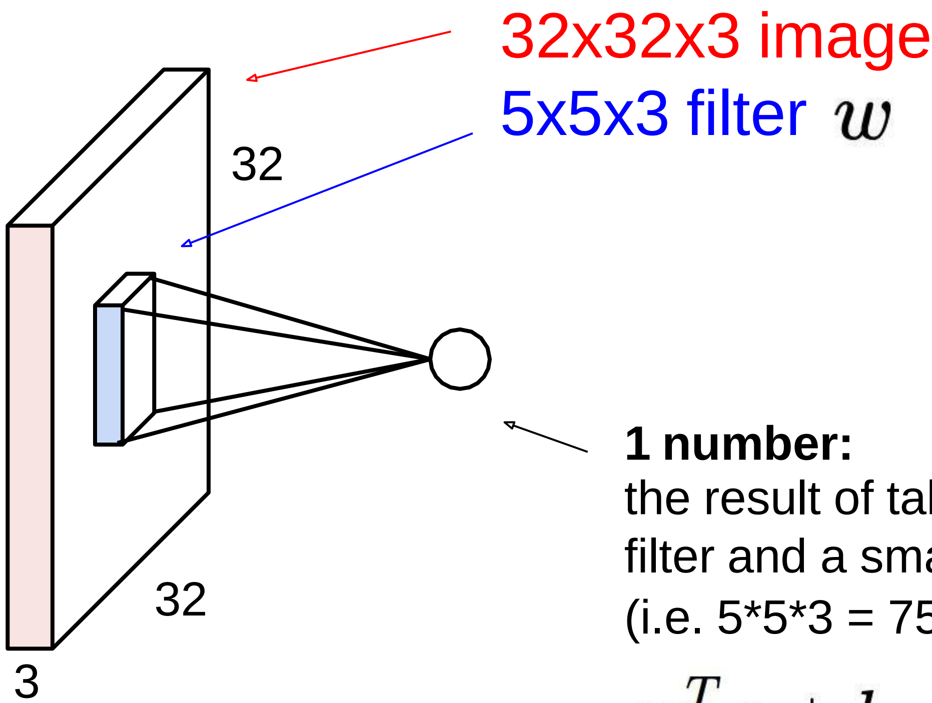


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Andrej Karpathy

Convolutions: More detail

Convolution Layer



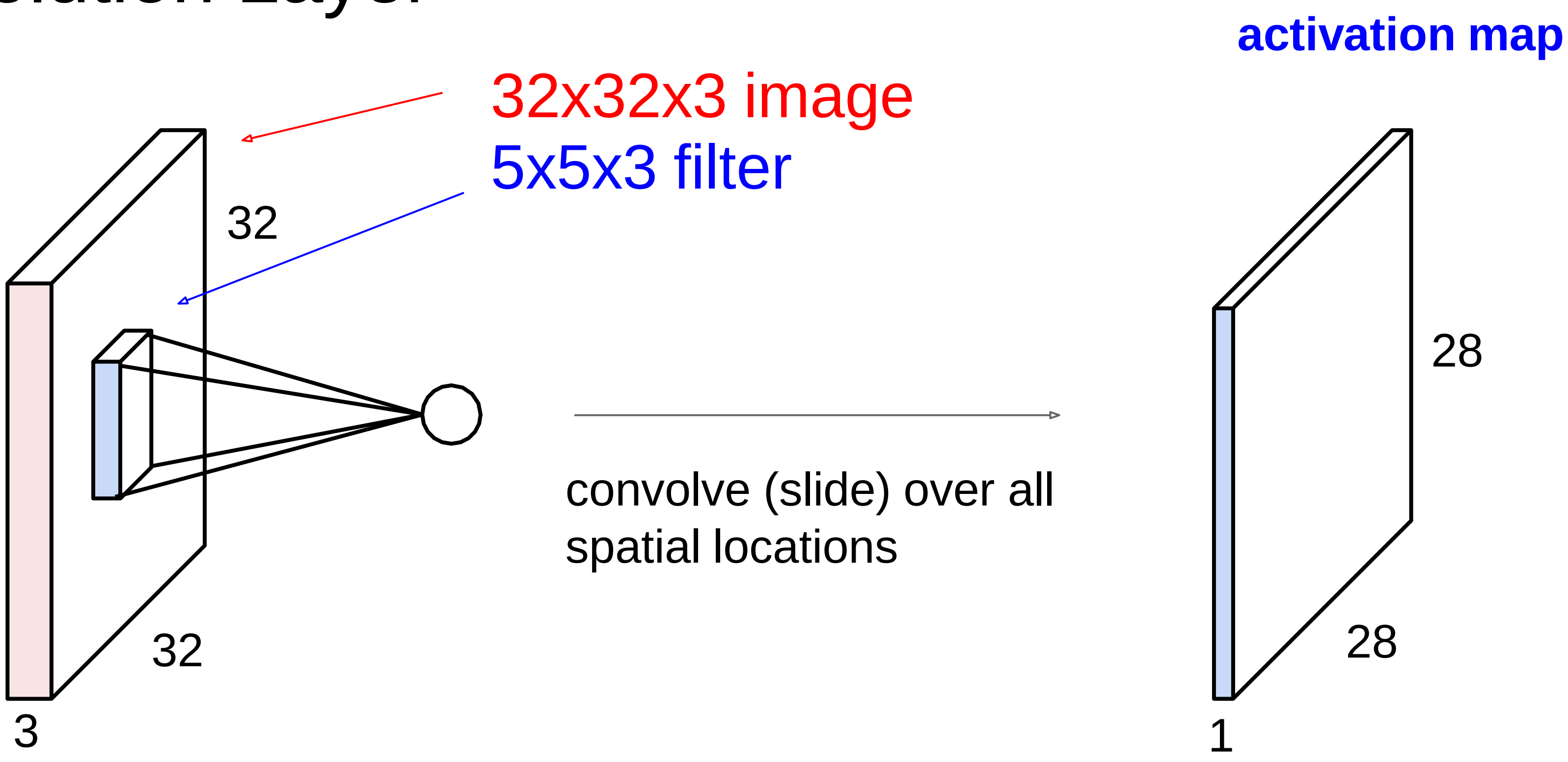
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Andrej Karpathy

Convolutions: More detail

Convolution Layer



Andrej Karpathy

Convolutions: More detail

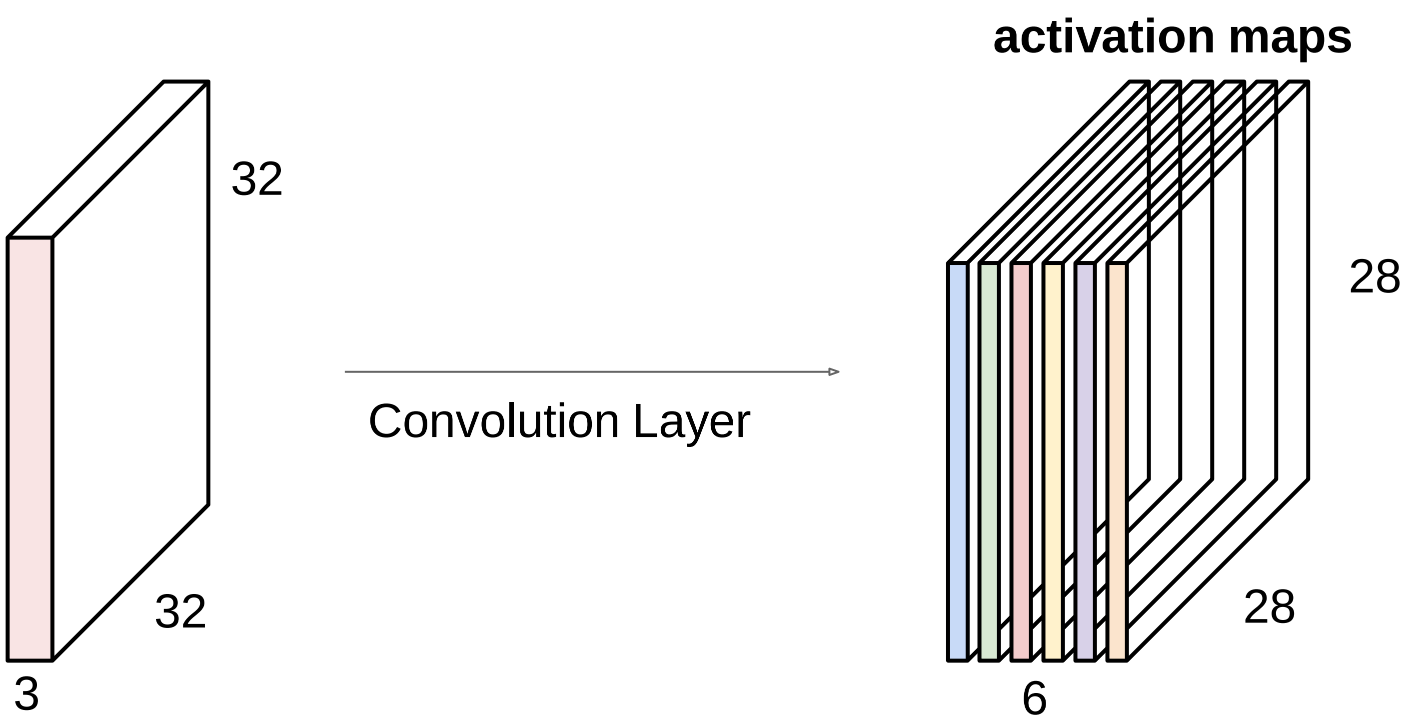
Convolution Layer

consider a second, green filter

Andrej Karpathy

Convolutions: More detail

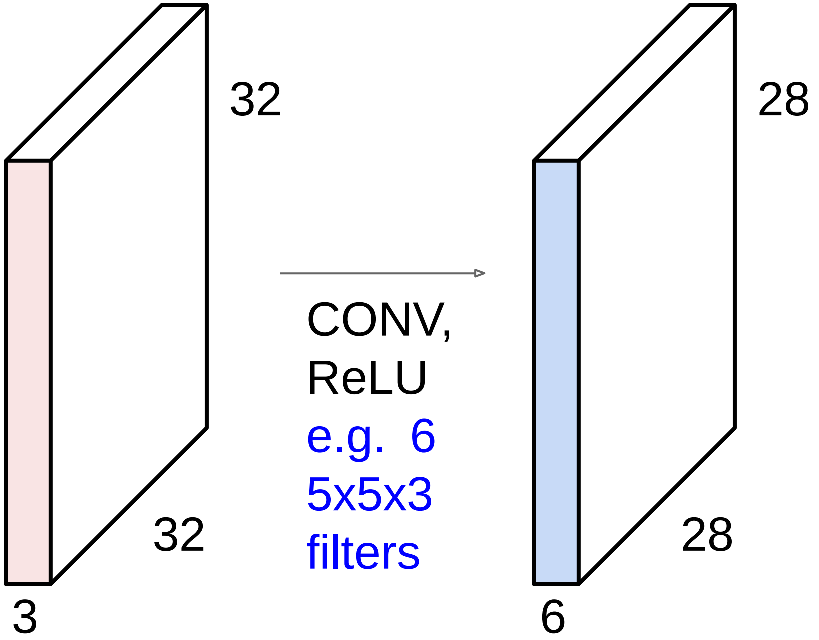
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

Convolutions: More detail

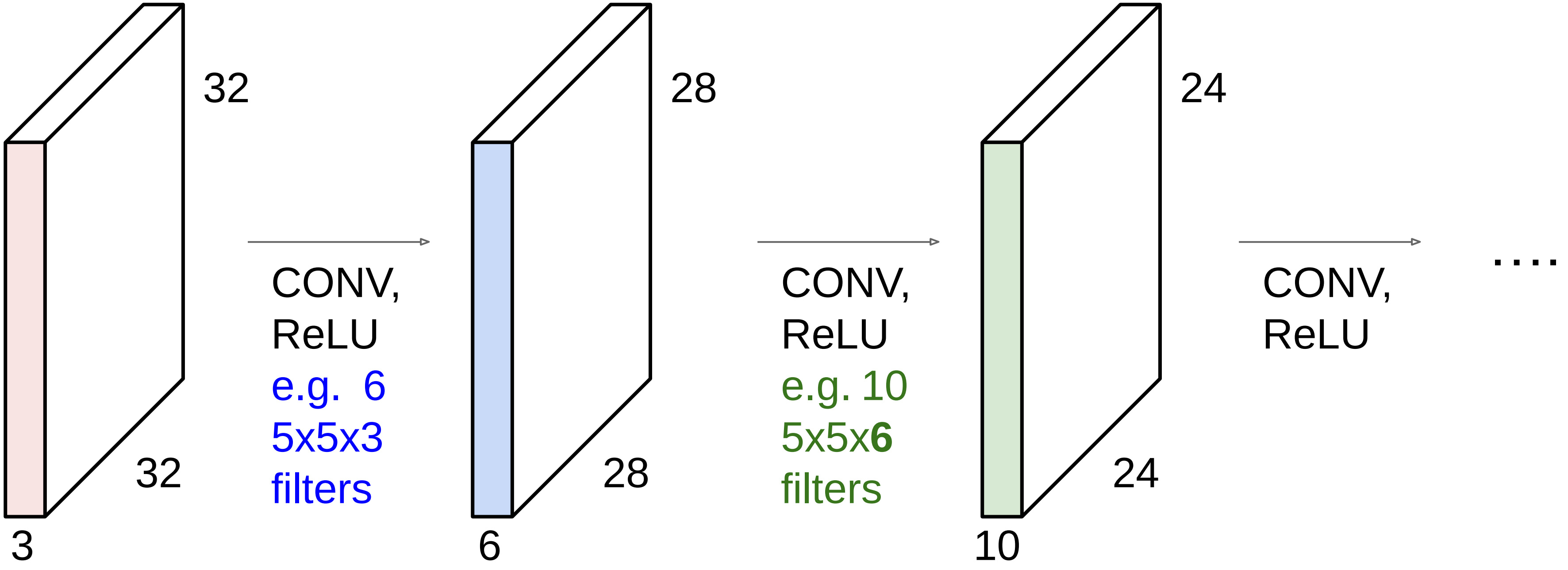
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Andrej Karpathy

Convolutions: More detail

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

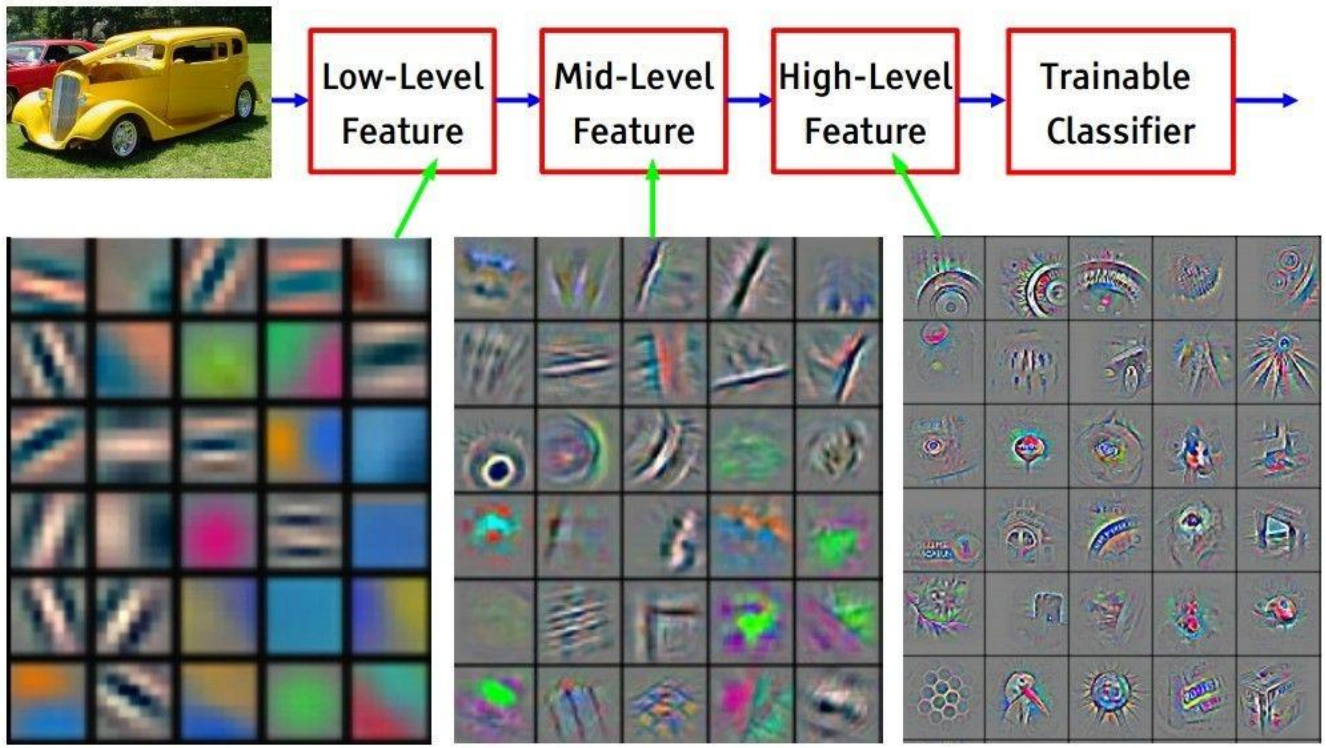


Andrej Karpathy

Convolutions: More detail

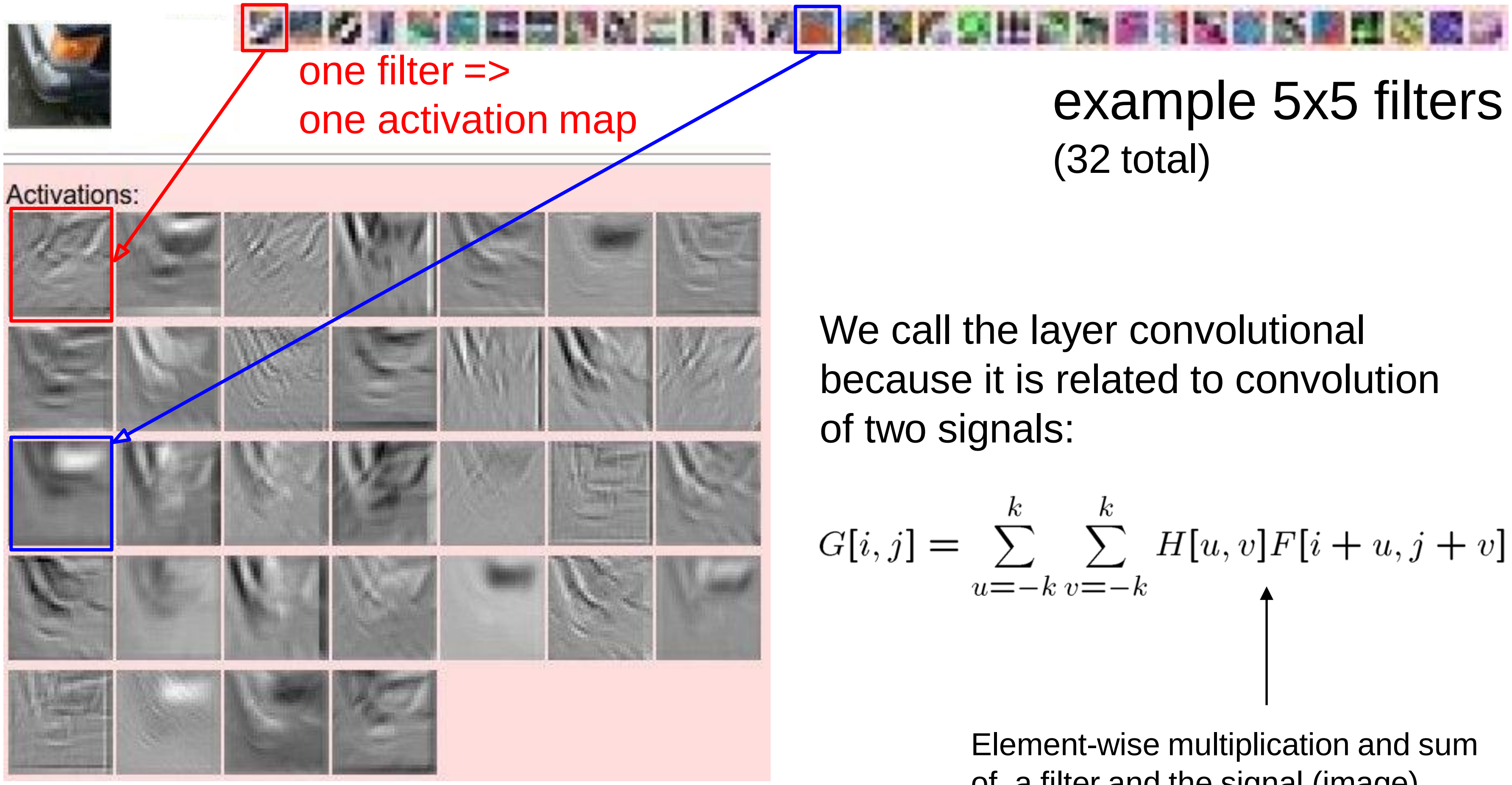
Preview

[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolutions: More detail



example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

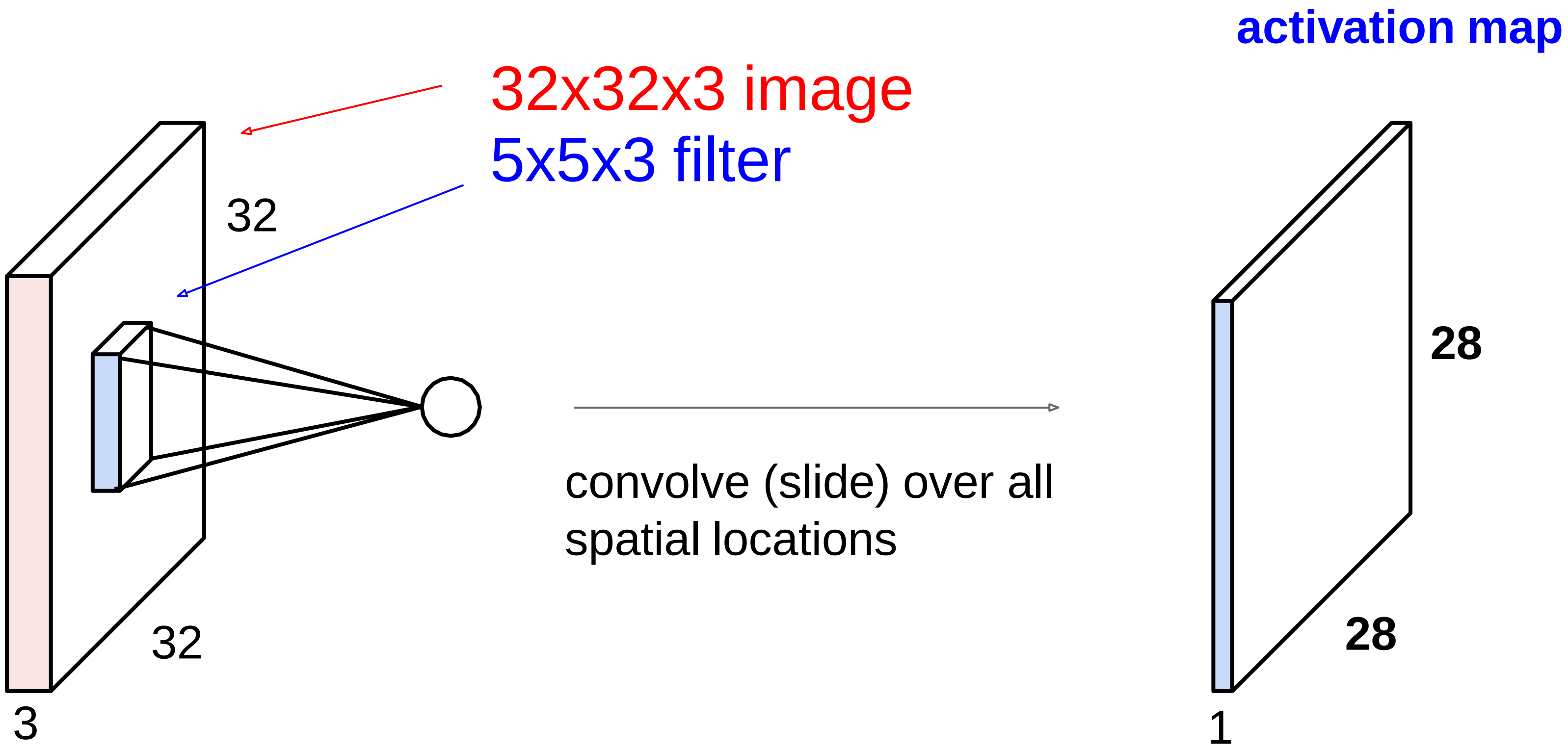
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

Element-wise multiplication and sum of a filter and the signal (image)

Adapted from Andrej Karpathy, Kristen Grauman

Convolutions: More detail

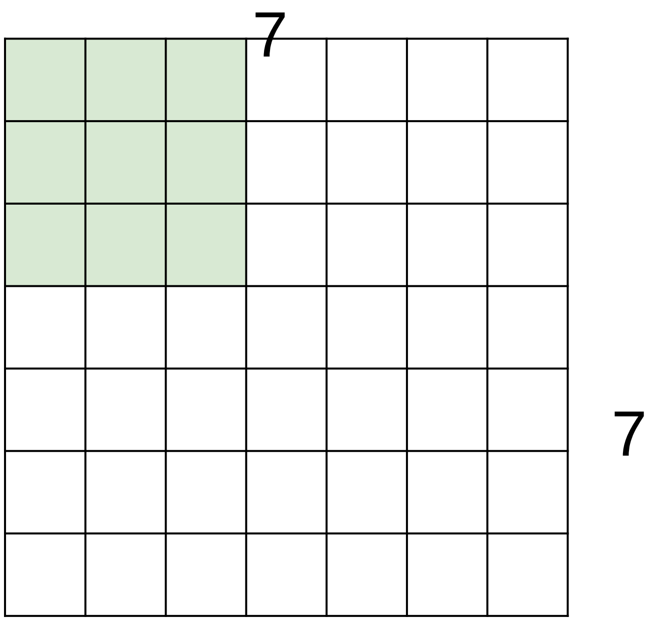
A closer look at spatial dimensions:



Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

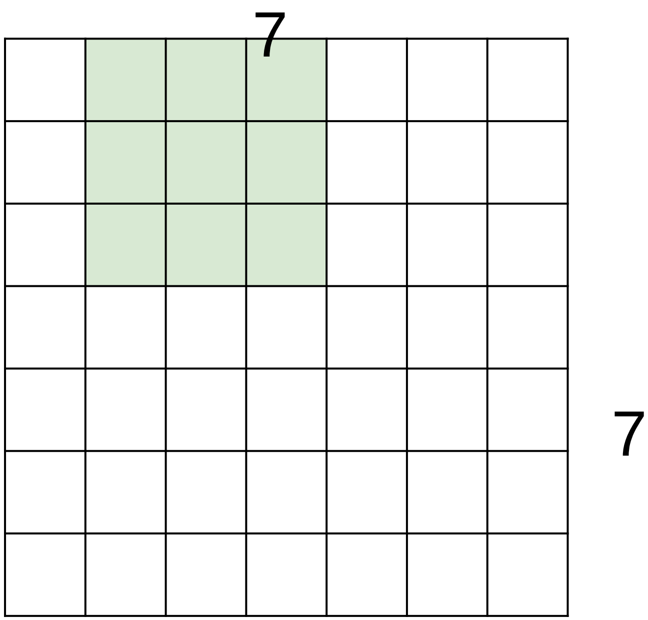


7x7 input (spatially)
assume 3x3 filter

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

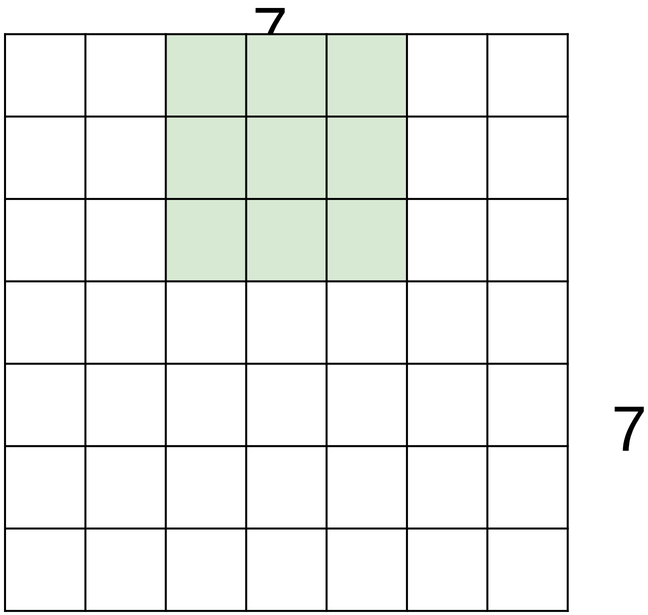


7x7 input (spatially)
assume 3x3 filter

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

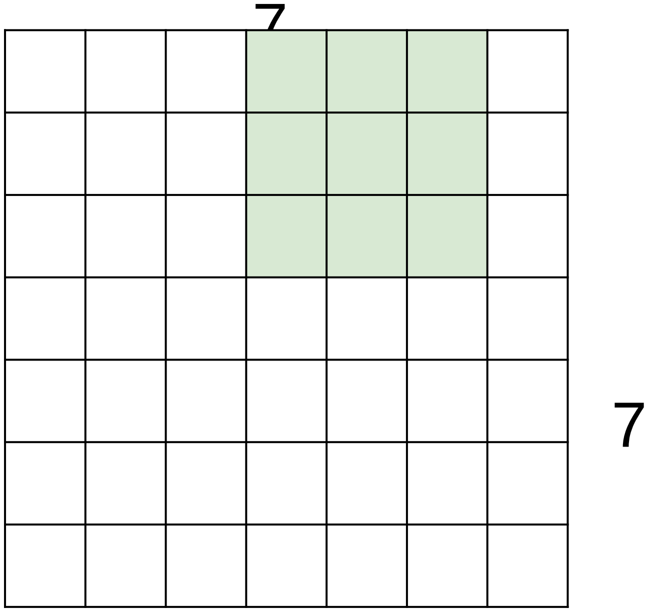


7x7 input (spatially)
assume 3x3 filter

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

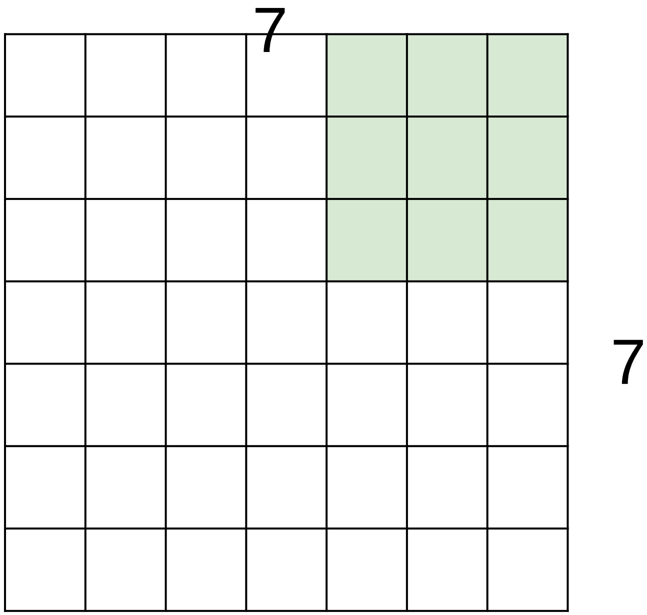


7x7 input (spatially)
assume 3x3 filter

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

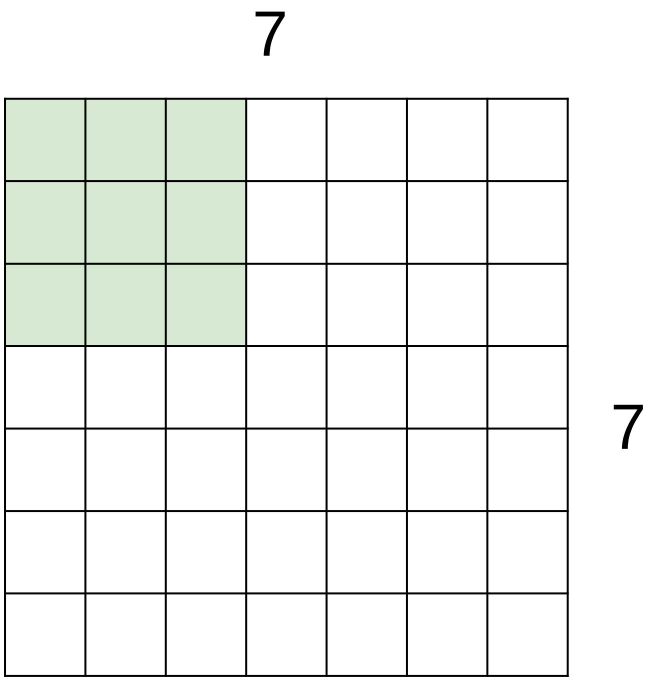


7x7 input (spatially)
assume 3x3 filter
=> 5x5 output

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

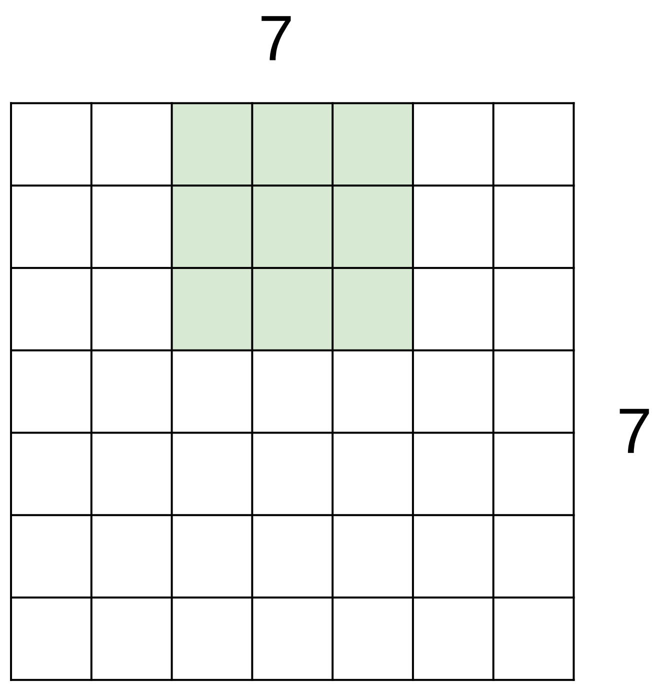


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

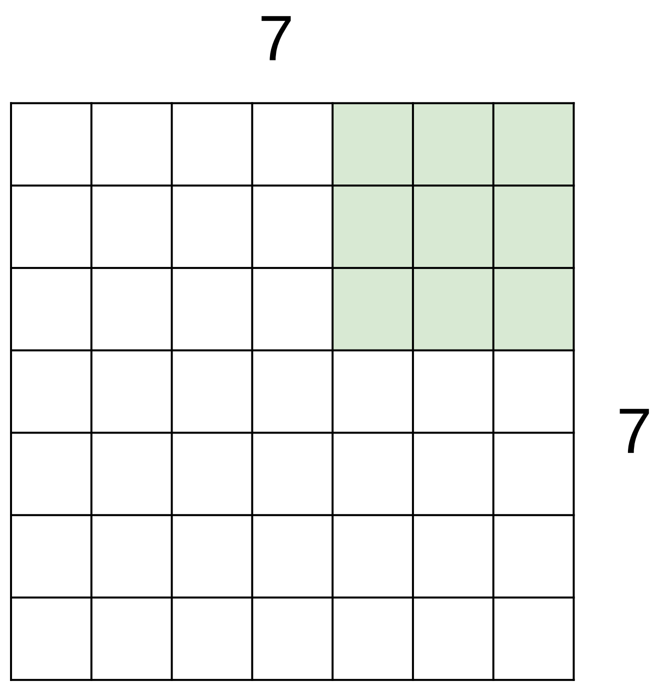


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

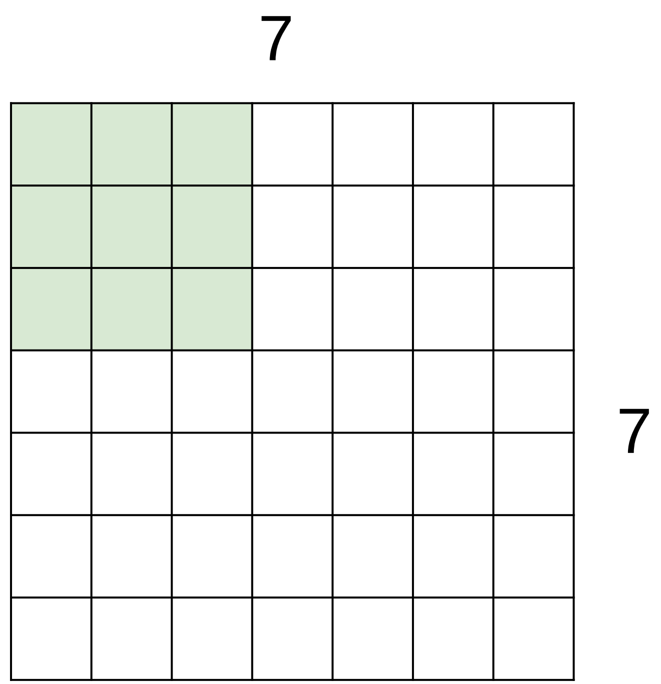


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

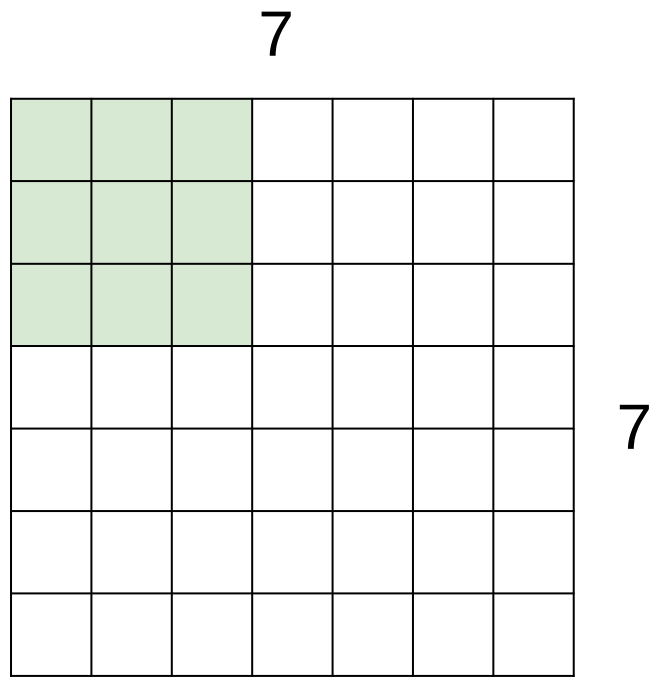


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

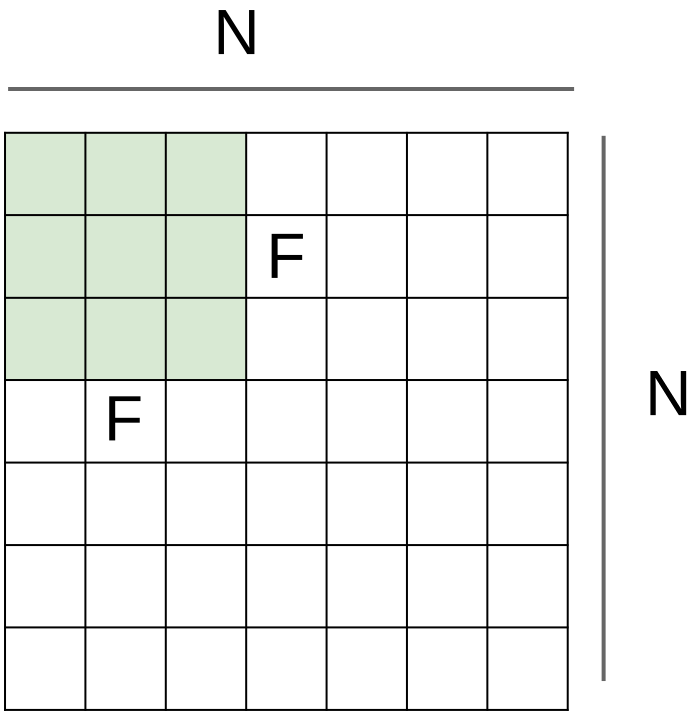


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Andrej Karpathy

Convolutions: More detail



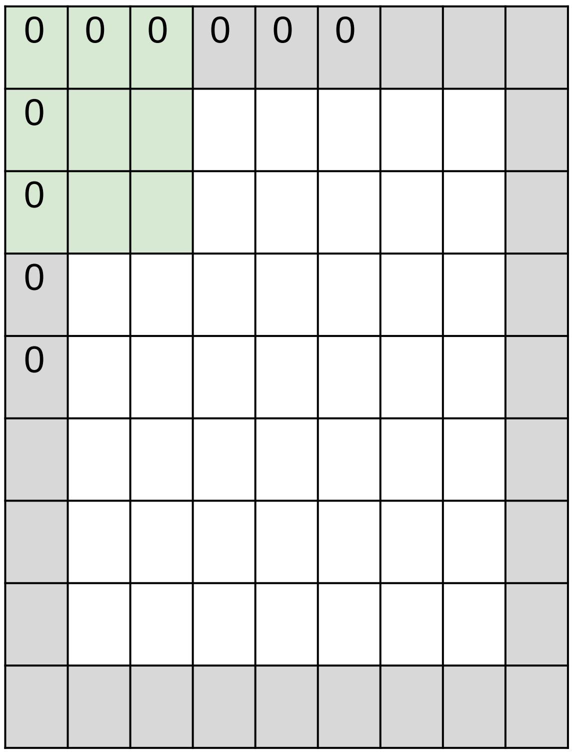
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Andrej Karpathy

Convolutions: More detail

In practice: Common to zero pad the border



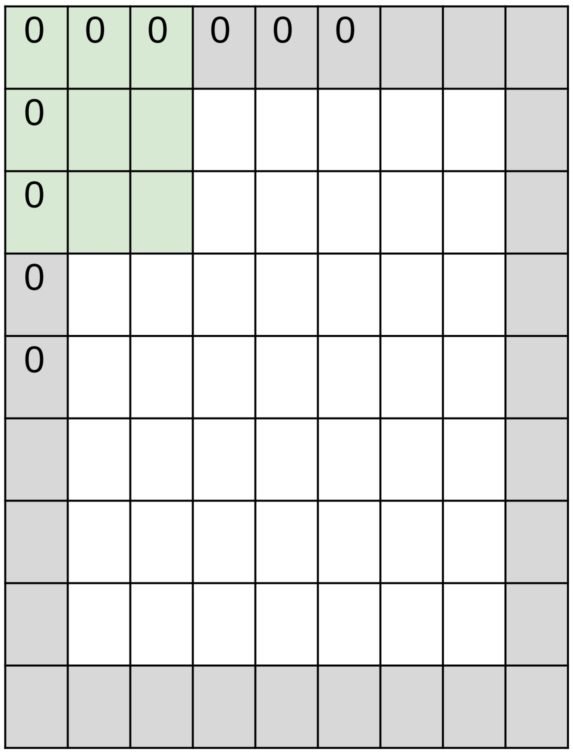
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

Andrej Karpathy

Convolutions: More detail

In practice: Common to zero pad the border



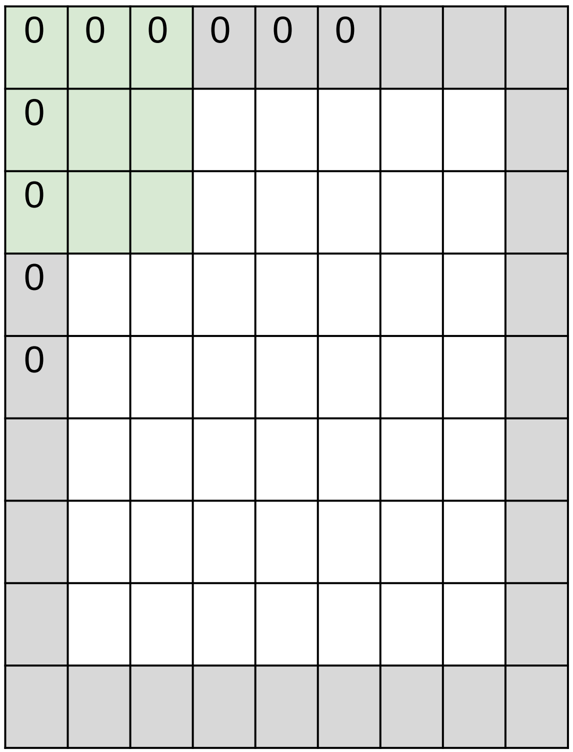
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Andrej Karpathy

Convolutions: More detail

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

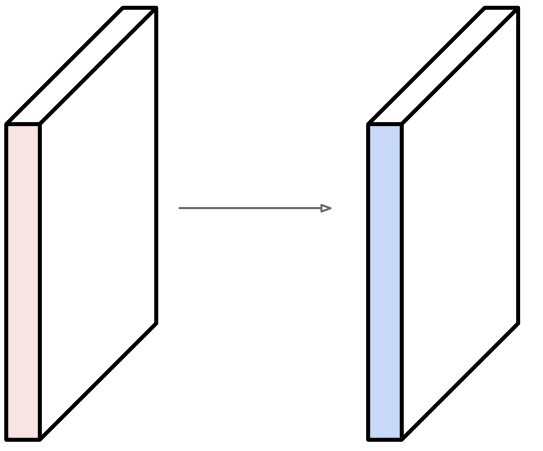
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

(N + 2*padding - F) / stride + 1

Andrej Karpathy

Convolutions: More detail

Examples time:



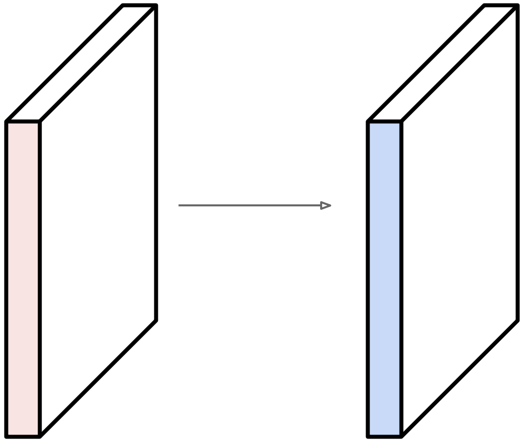
Input volume: **32x32x3**
10 5x5x3 filters with stride 1, pad 2

Output volume size: ?

Andrej Karpathy

Convolutions: More detail

Examples time:



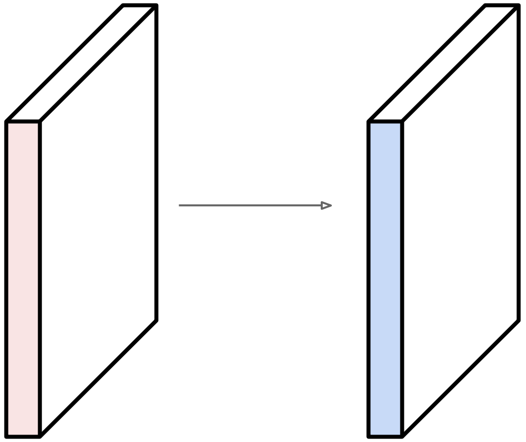
Input volume: **32x32x3**
10 5x5x3 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Andrej Karpathy

Convolutions: More detail

Examples time:



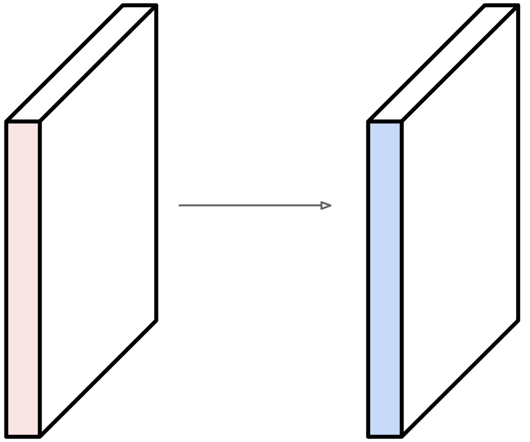
Input volume: **32x32x3**
10 5x5x3 filters with stride 1, pad 2

Number of parameters in this layer?

Andrej Karpathy

Convolutions: More detail

Examples time:



Input volume: **32x32x3**

10 5x5x3 filters with stride 1, pad 2

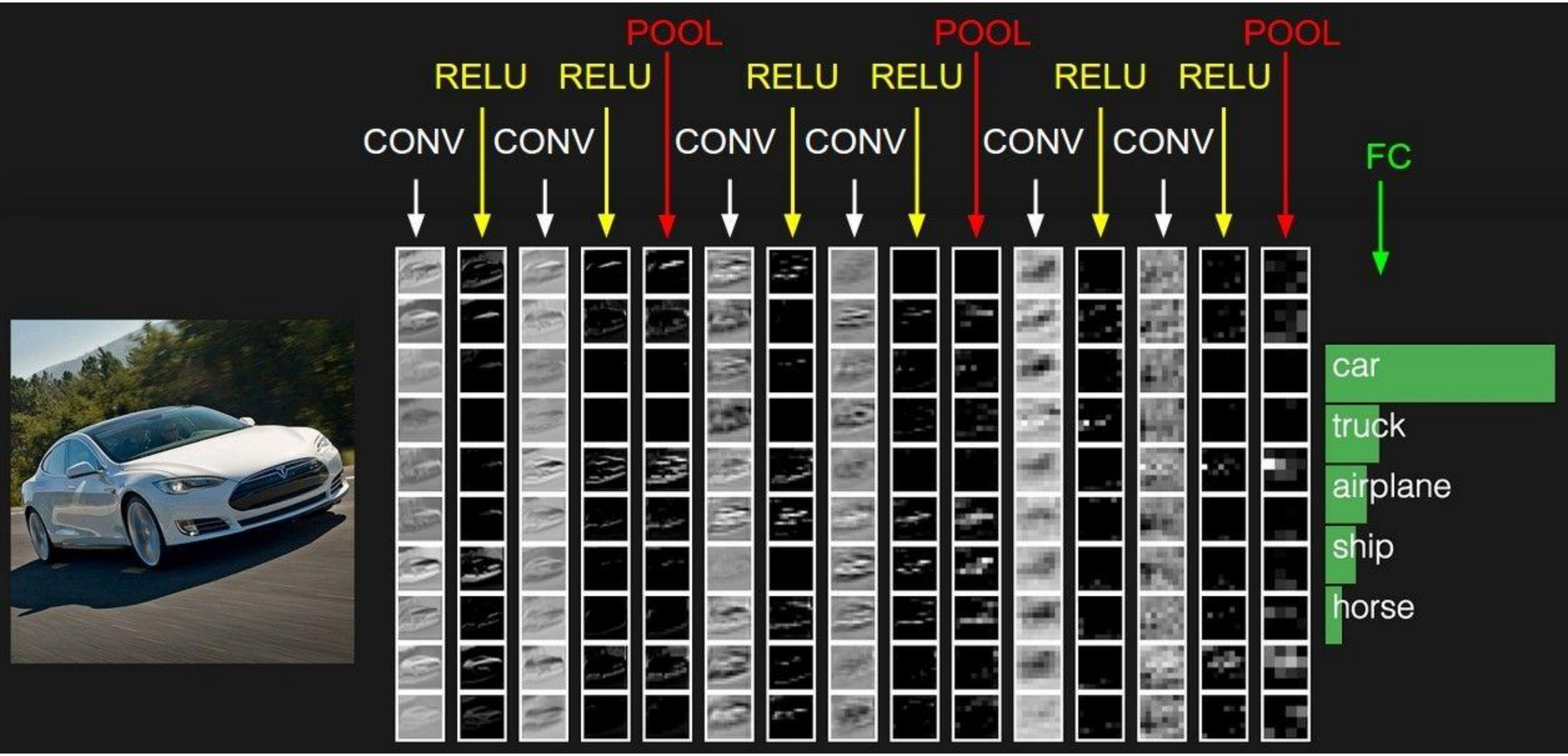
Number of parameters in this layer?

each filter has 5*5*3 + 1 = 76 params (+1 for bias)

=> 76*10 = **760**

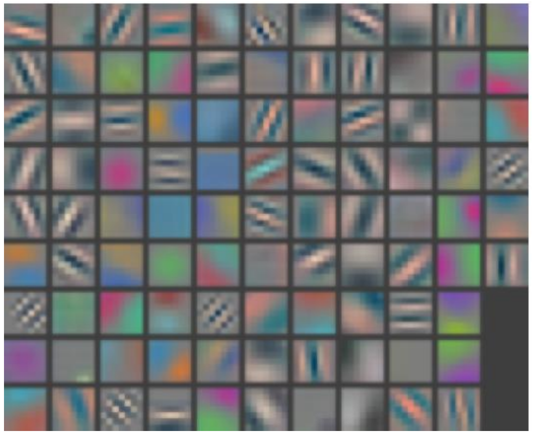
Andrej Karpathy

Putting it all together



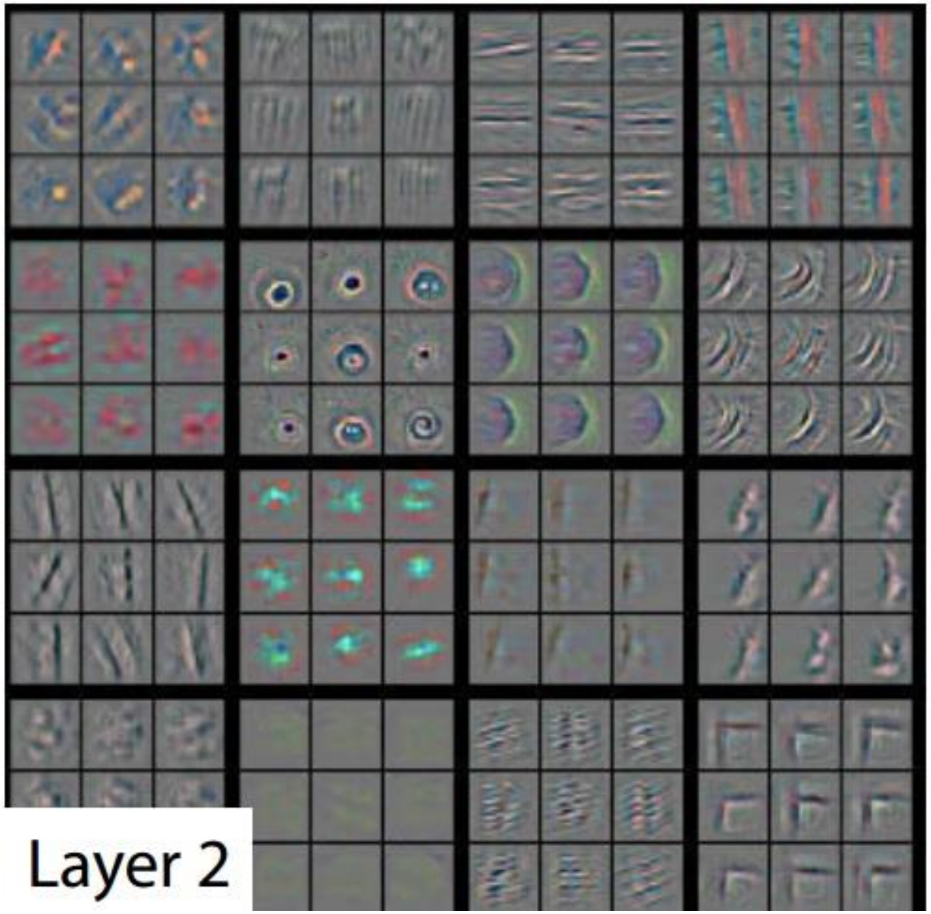
Andrej Karpathy

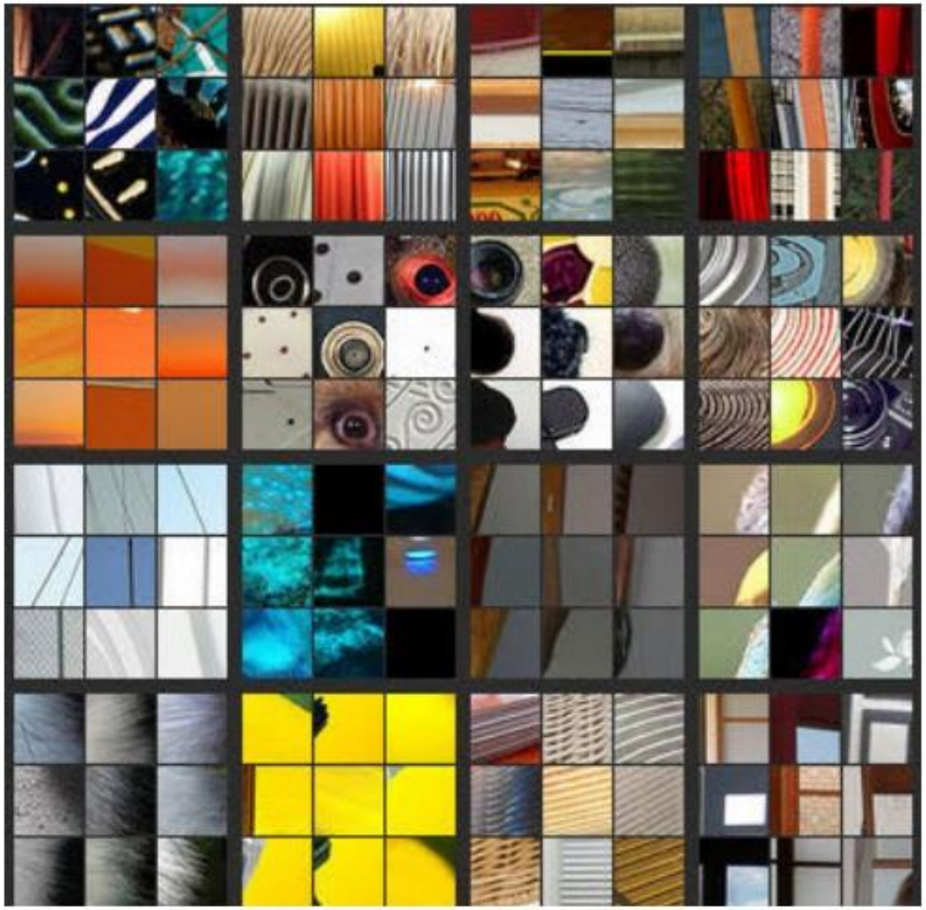
Layer 1



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 2

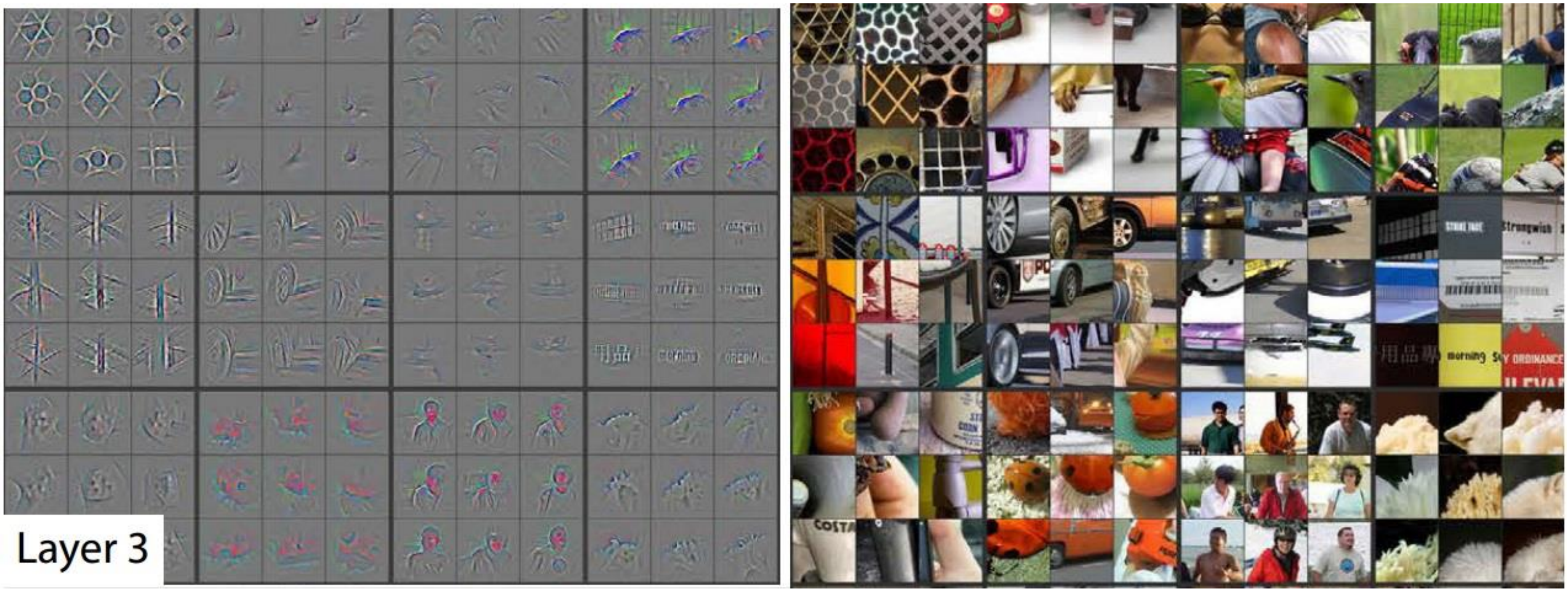




- Activations projected down to pixel level via decovolution
- Patches from validation images that give maximal activation of a given feature map

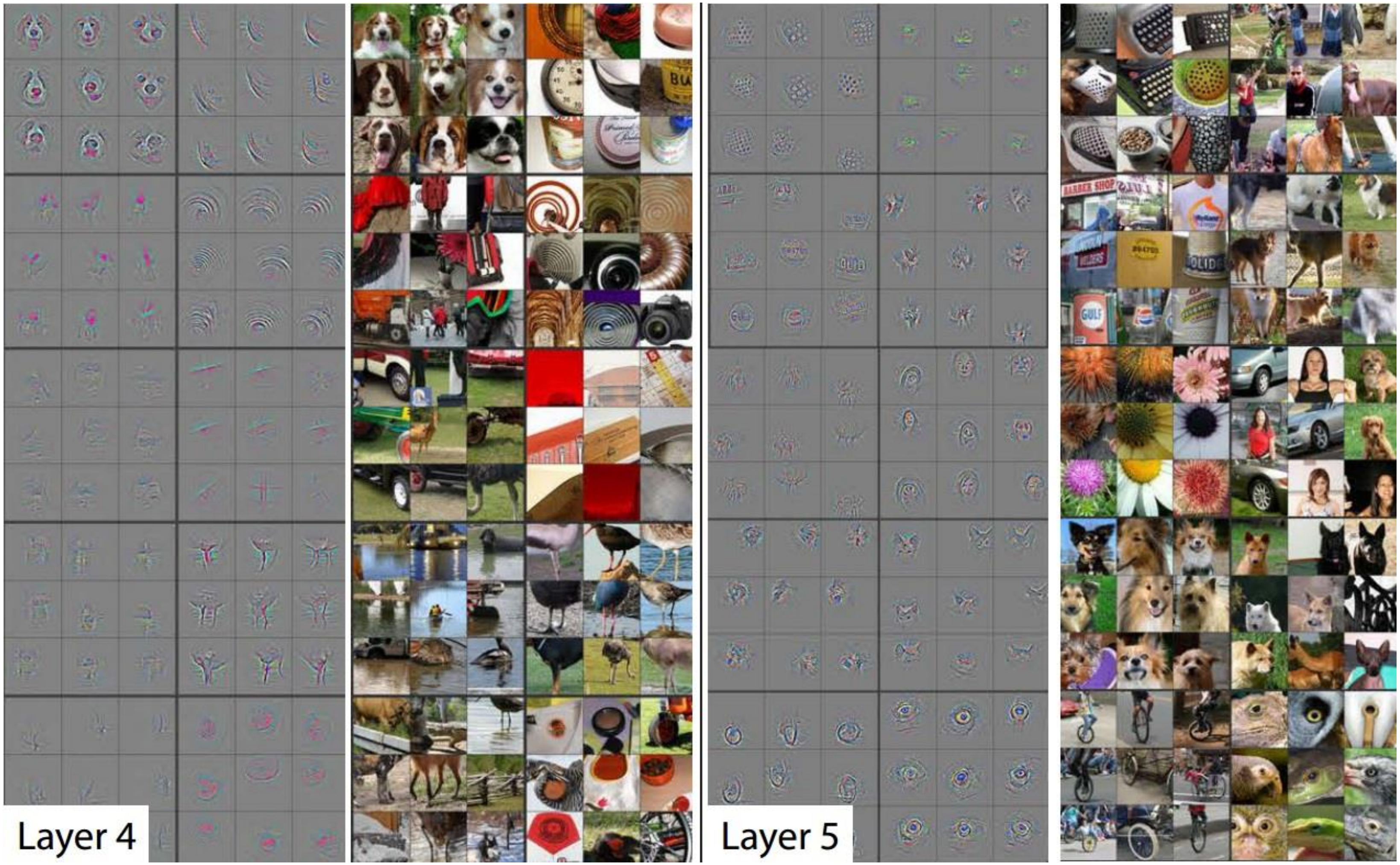
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 3



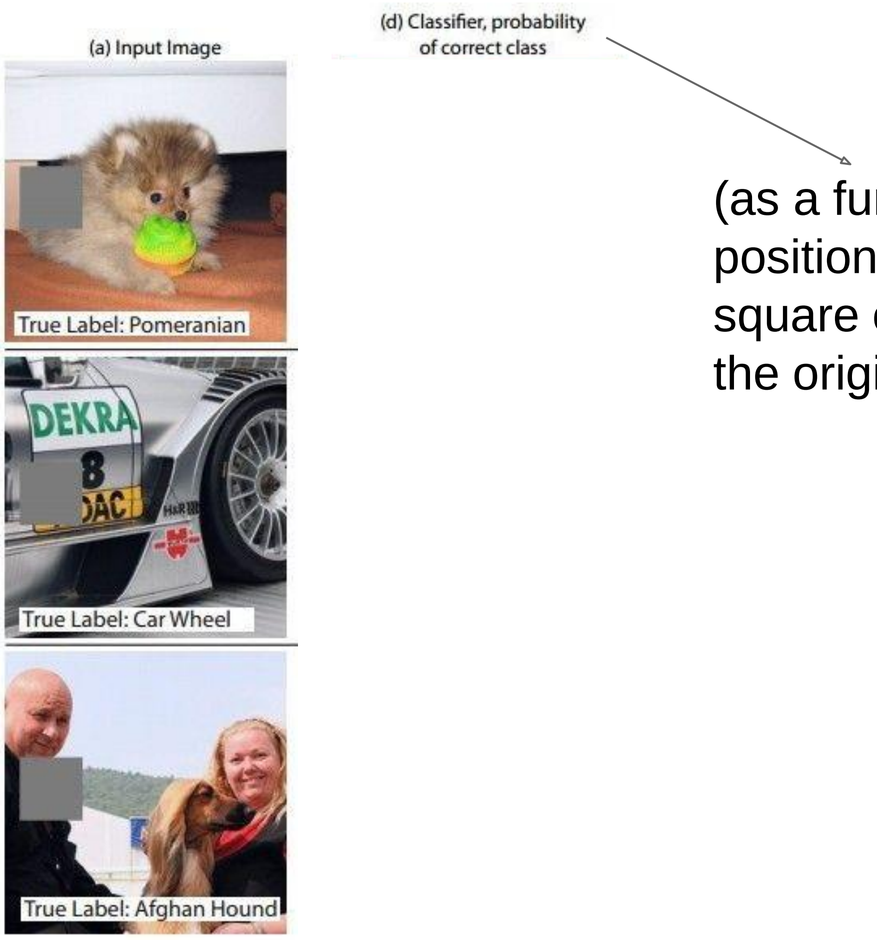
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 4 and 5



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

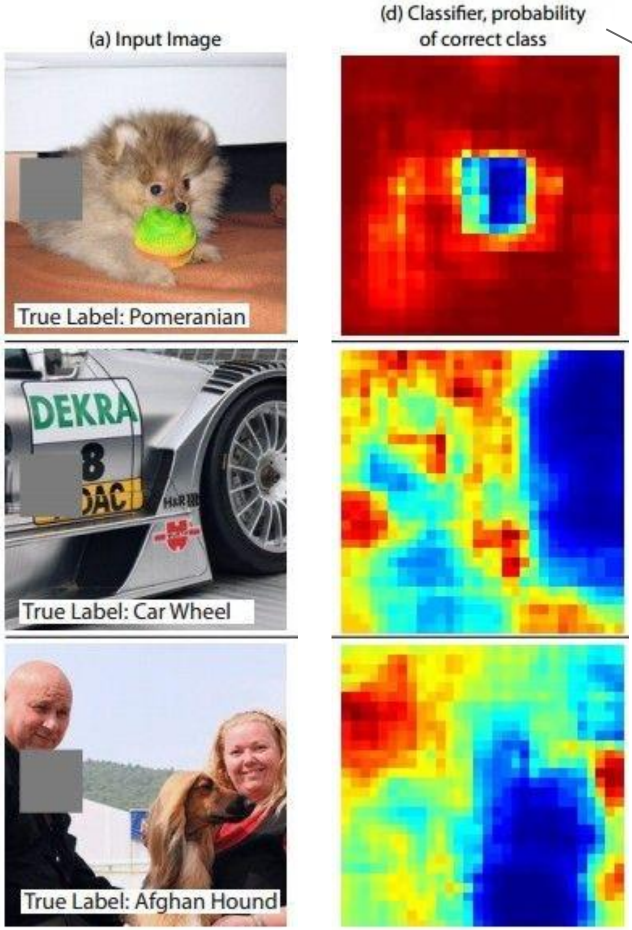
Occlusion experiments



(as a function of the position of the square of zeros in the original image)

[Zeiler & Fergus 2014]

Occlusion experiments

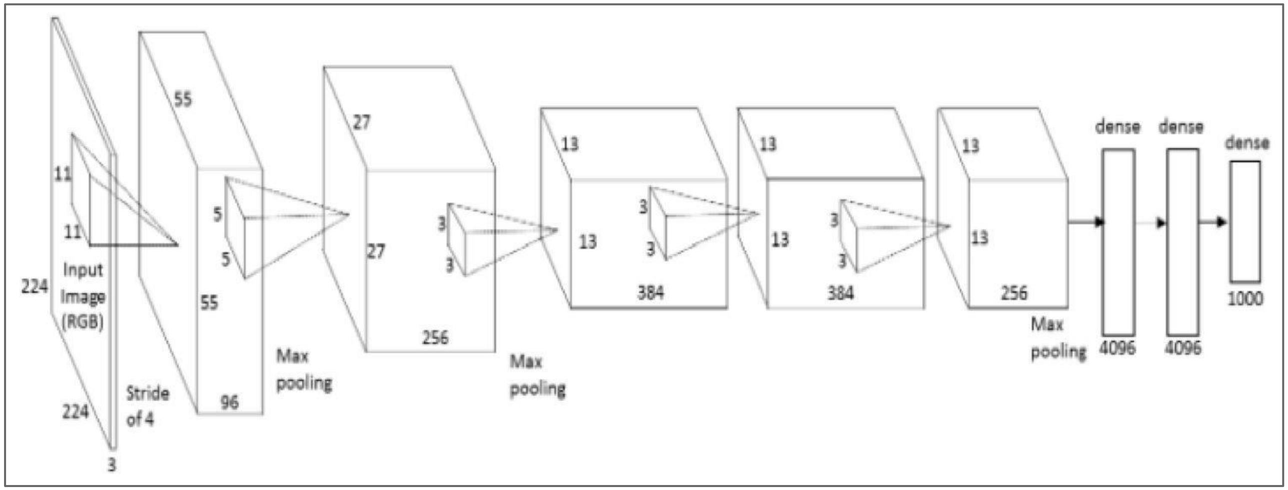


(as a function of the position of the square of zeros in the original image)

[Zeiler & Fergus 2014]

Andrej Karpathy

What image maximizes a class score?

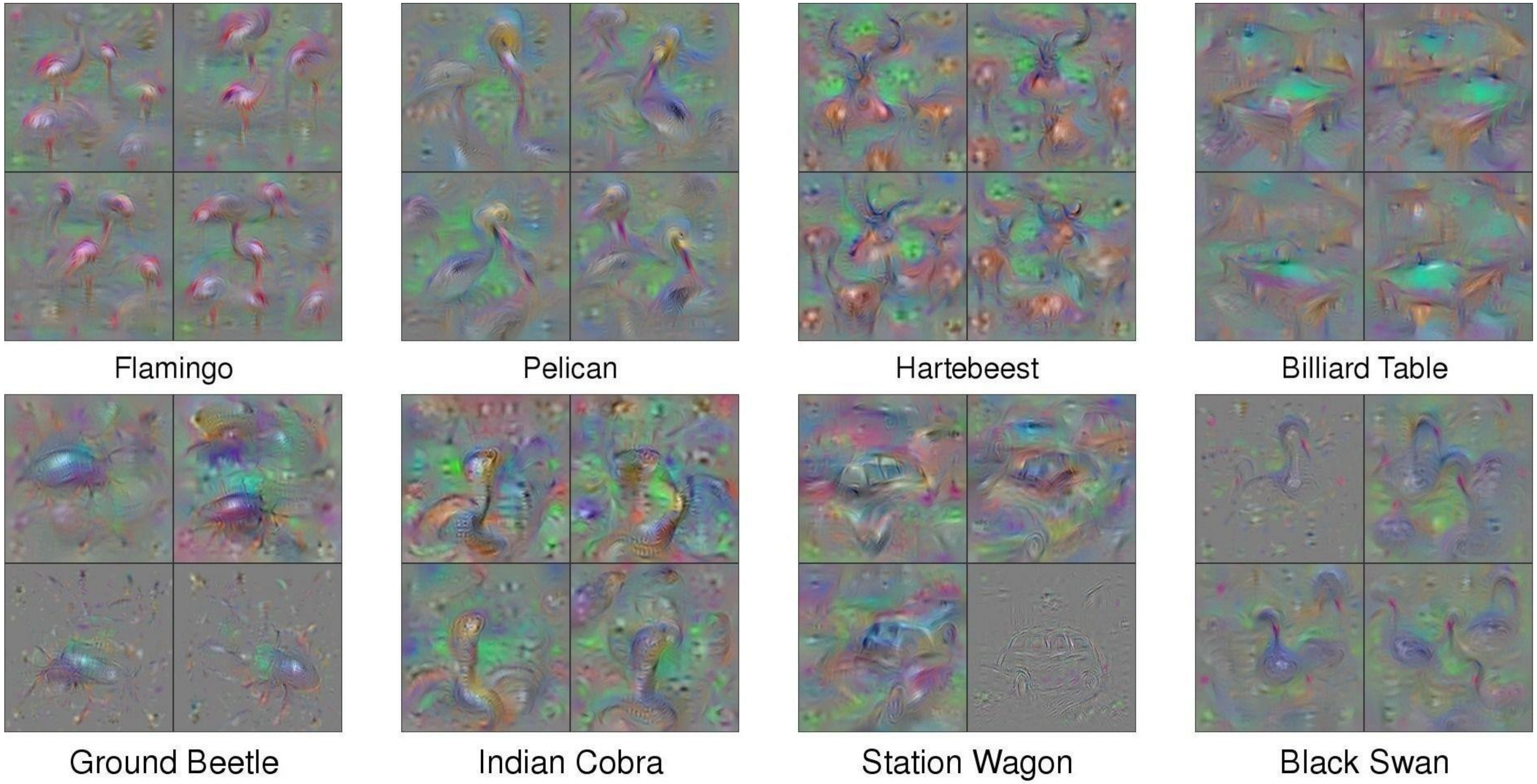


Repeat:

1. Forward an image
2. Set activations in layer of interest to all zero, except for a 1.0 for a neuron of interest
3. Backprop to image
4. Do an “image update”

Andrej Karpathy

What image maximizes a class score?



[Understanding Neural Networks Through Deep Visualization, Yosinski et al. , 2015]

http://yosinski.com/deepvis

Shape vs texture

(a) Texture image		(b) Content image		(c) Texture-shape cue conflict	
81.4%	**Indian elephant**	71.1%	**tabby cat**	63.9%	**Indian elephant**
10.3%	indri	17.3%	grey fox	26.4%	indri
8.2%	black swan	3.3%	Siamese cat	9.6%	black swan

Figure 1: Classification of a standard ResNet-50 of **(a)** a texture image (elephant skin: only texture cues); **(b)** a normal image of a cat (with both shape and texture cues), and **(c)** an image with a texture-shape cue conflict, generated by style transfer between the first two images.

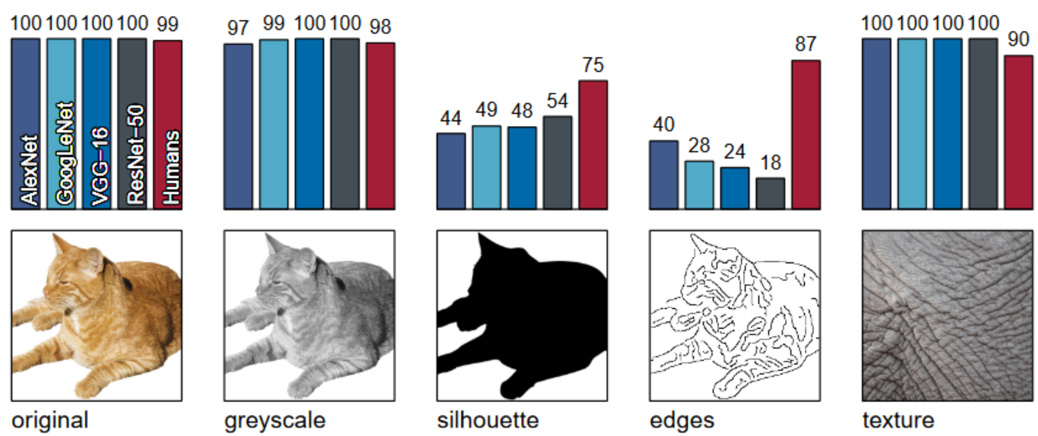


Figure 2: Accuracies and example stimuli for five different experiments without cue conflict.

ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness [Geirhos et al., ICLR 2019]

A Basic Architecture: AlexNet

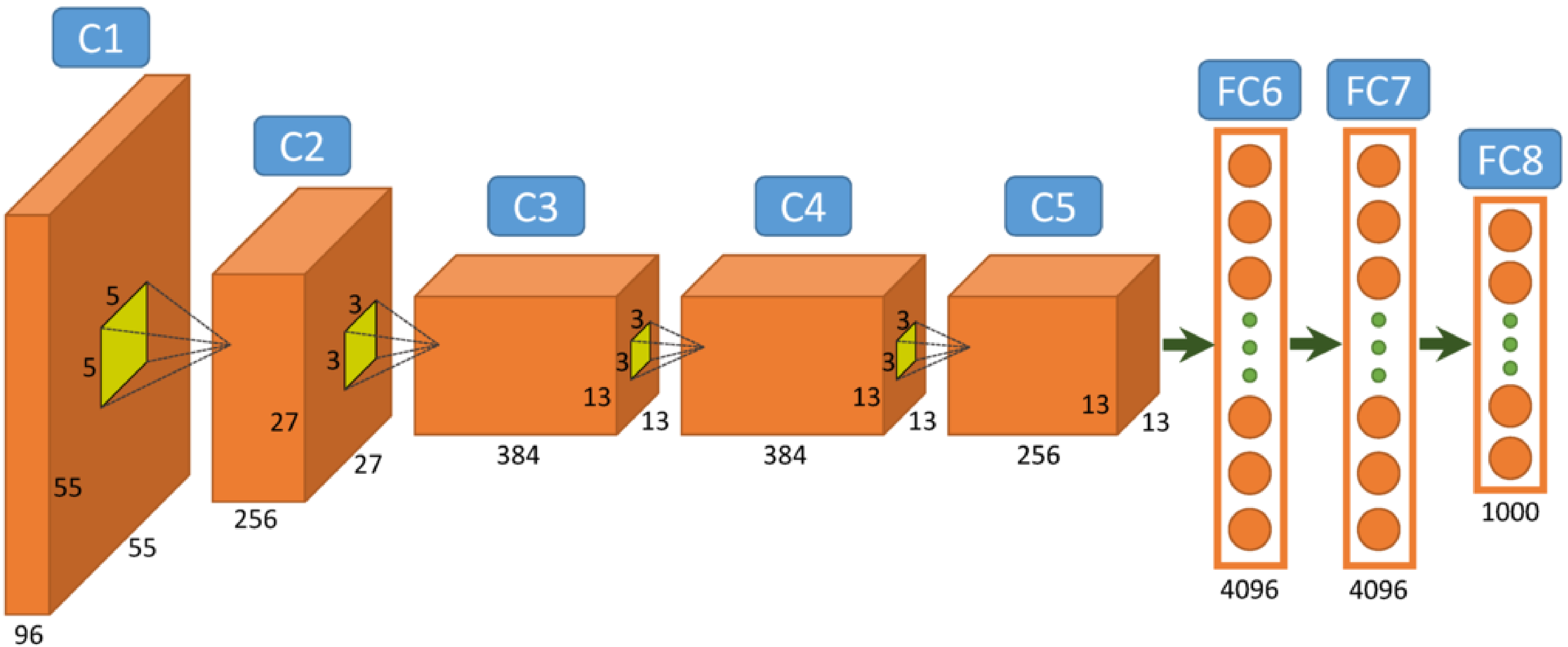
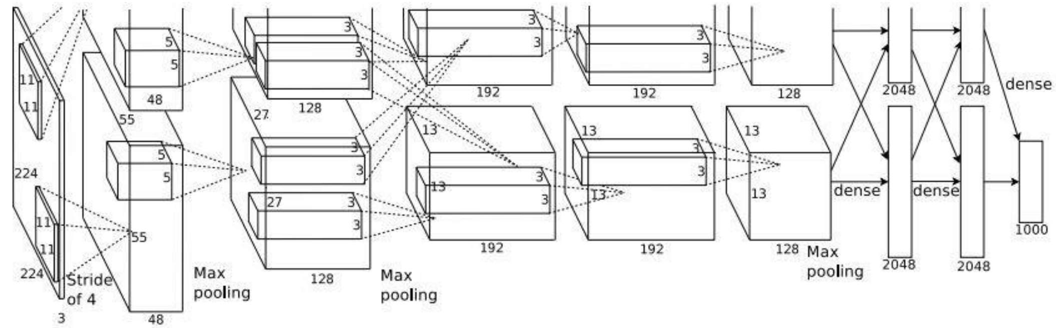


Figure from http://www.mdpi.com/2072-4292/7/11/14680/htm

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013
->
7.3% top 5 error

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

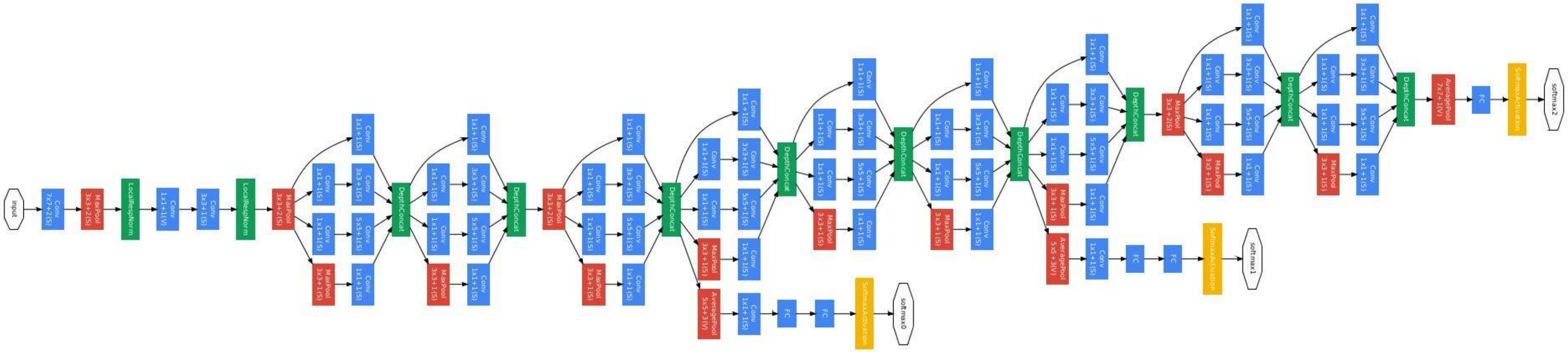
Table 2: **Number of parameters** (in millions).

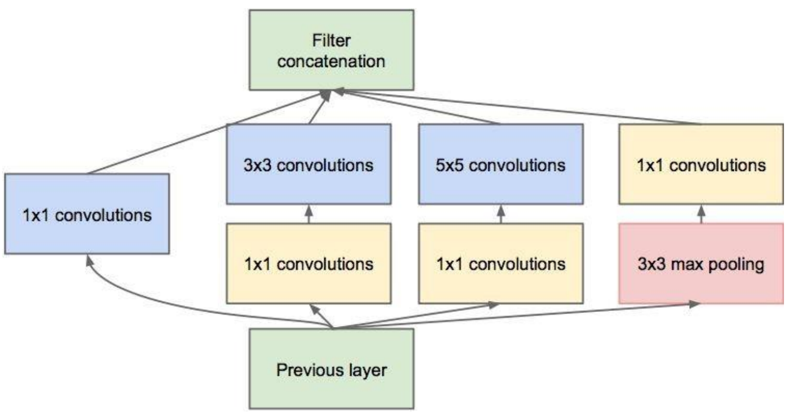
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Andrej Karpathy

Case Study: GoogLeNet

[Szegedy et al., 2014]





Inception module

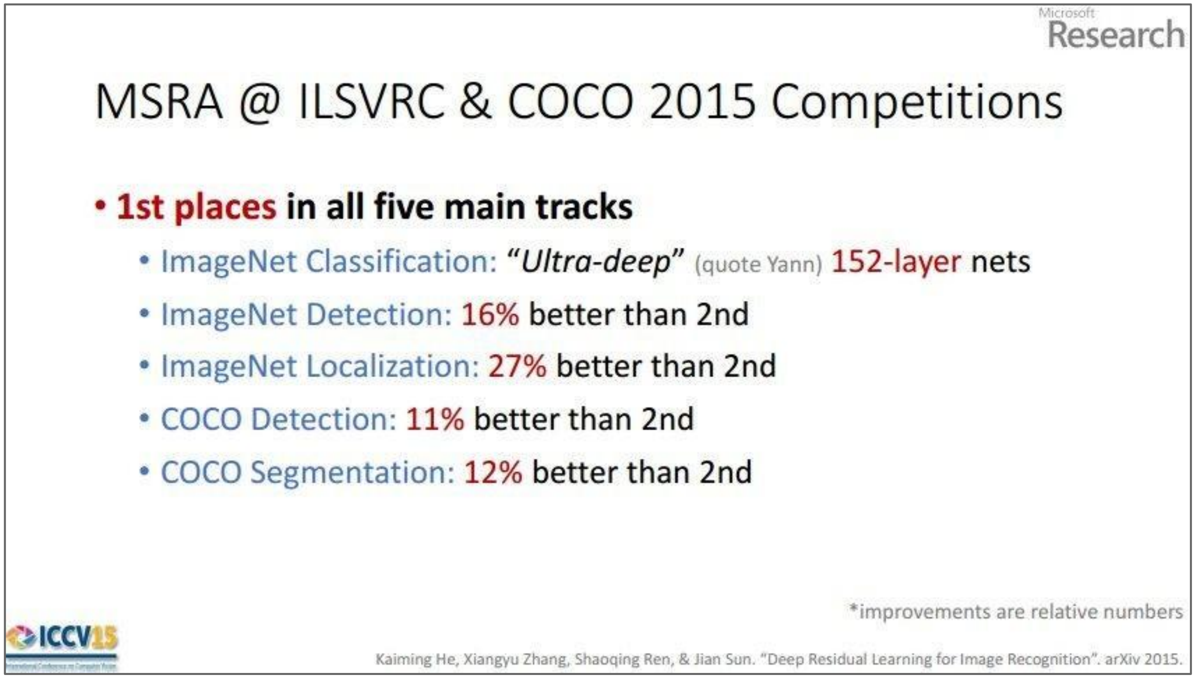
ILSVRC 2014 winner (6.7% top 5 error)

Andrej Karpathy

Case Study: ResNet

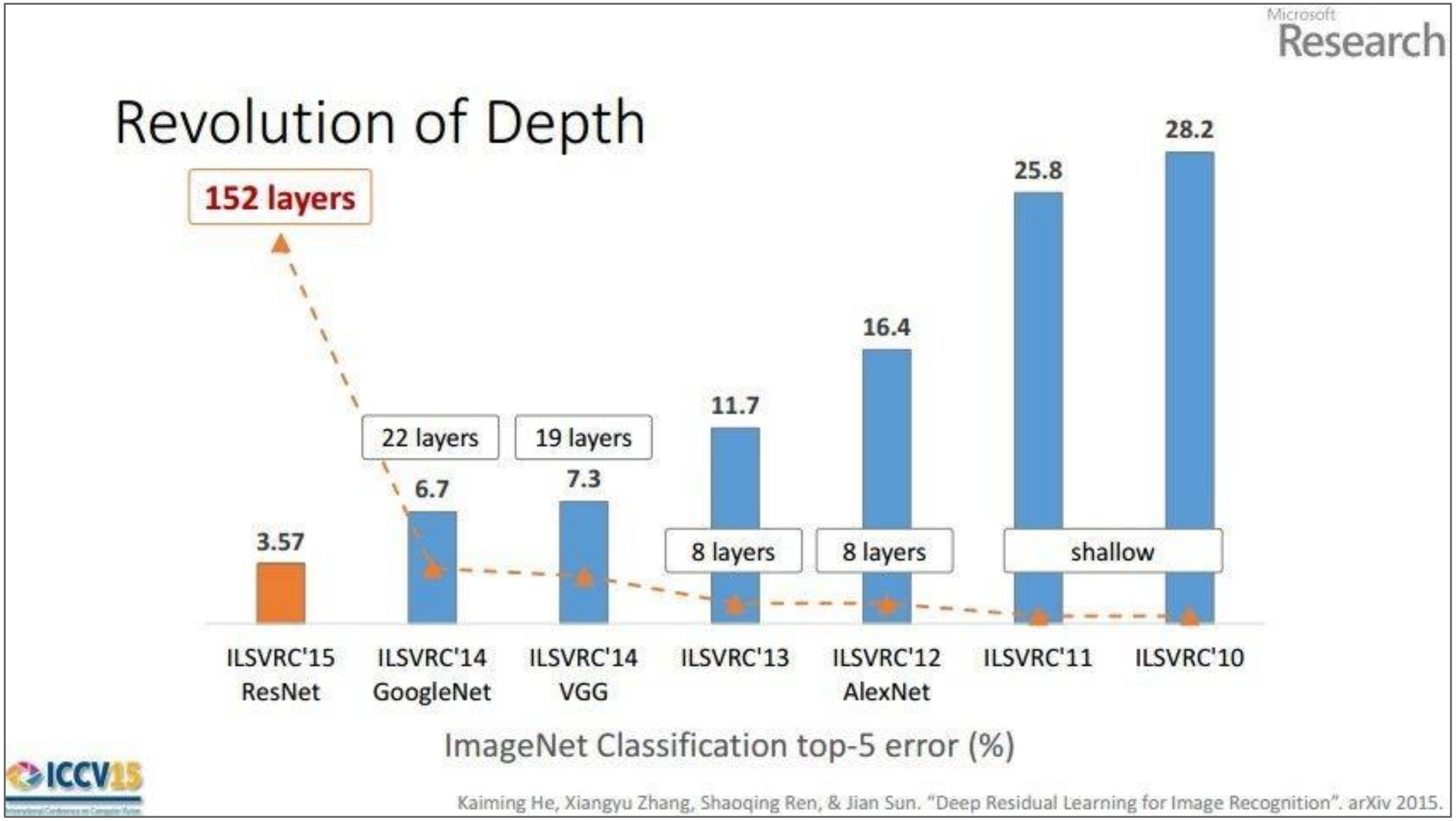
[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)



Slide from Kaiming He's presentation https://www.youtube.com/watch?v=1PGLj-uKT1w

Case Study: ResNet



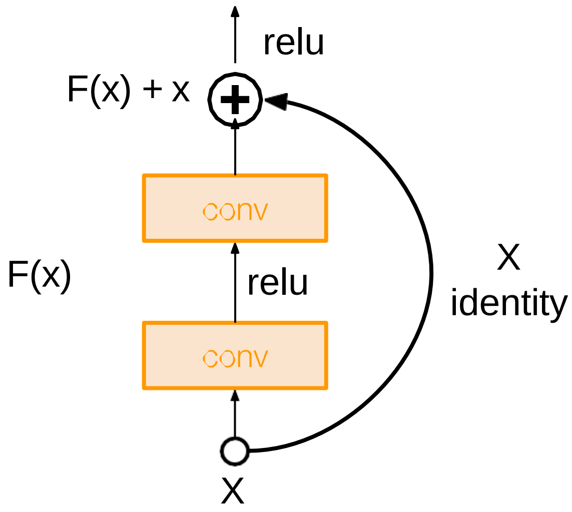
(slide from Kaiming He's presentation)

Case Study: ResNet

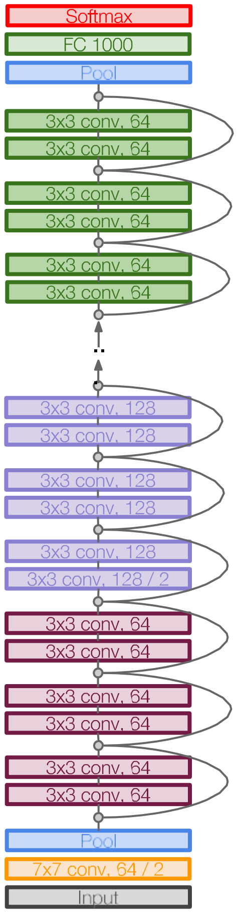
[He et al., 2016]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



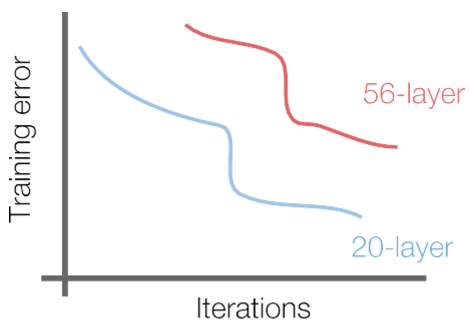
Residual block

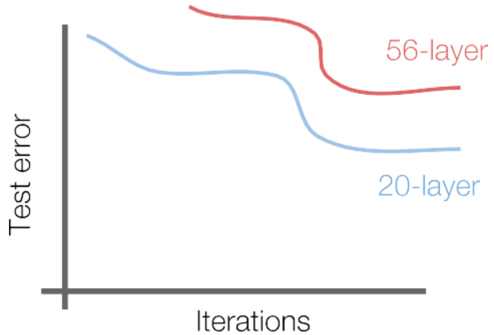


Case Study: ResNet

[He et al., 2016]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?





Q: What’s strange about these training and test curves?
[Hint: look at the order of the curves]

56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it’s not caused by overfitting!

Case Study: ResNet

[He et al., 2016]

Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

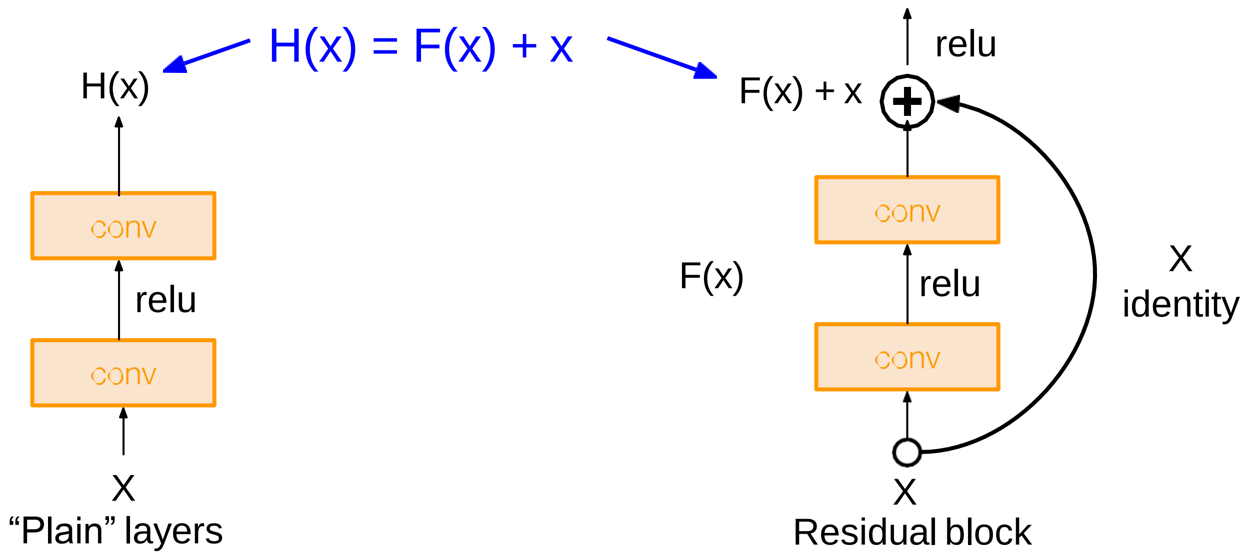
The deeper model should be able to perform at least as well as the shallower model.

A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

Case Study: ResNet

[He et al., 2016]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

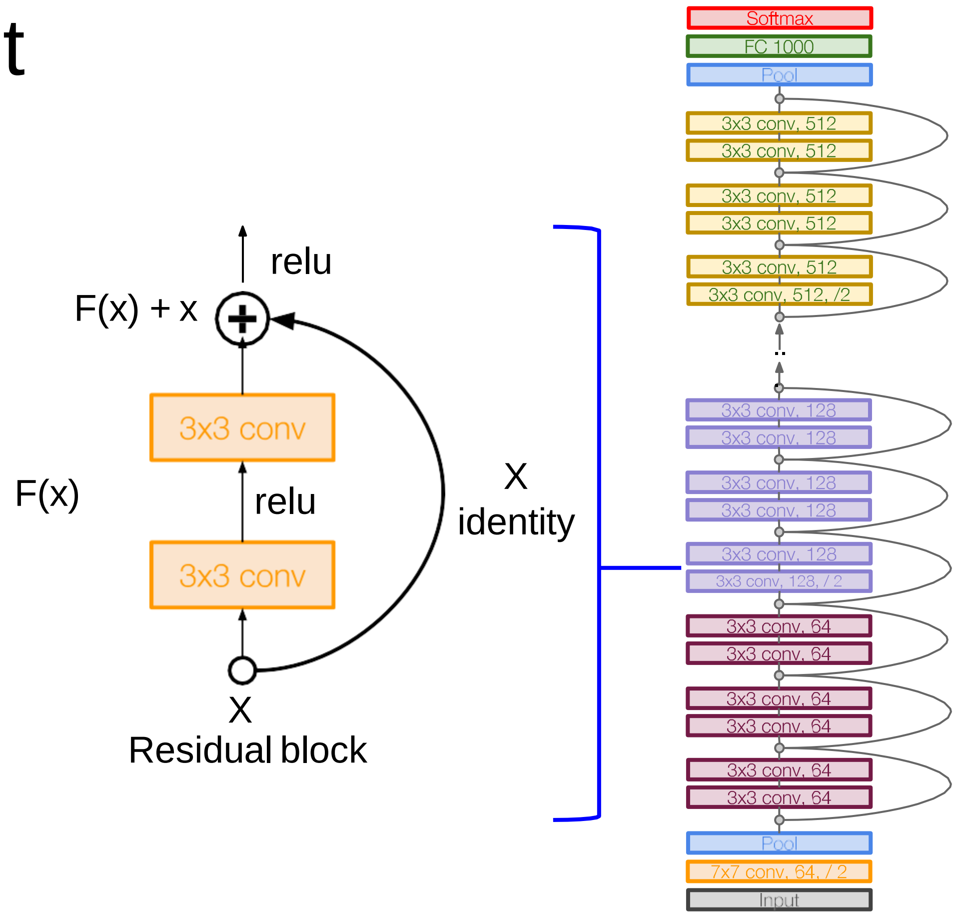


Use layers to fit residual F(x) = H(x) - x instead of H(x) directly

Case Study: ResNet

[He et al., 2016]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers

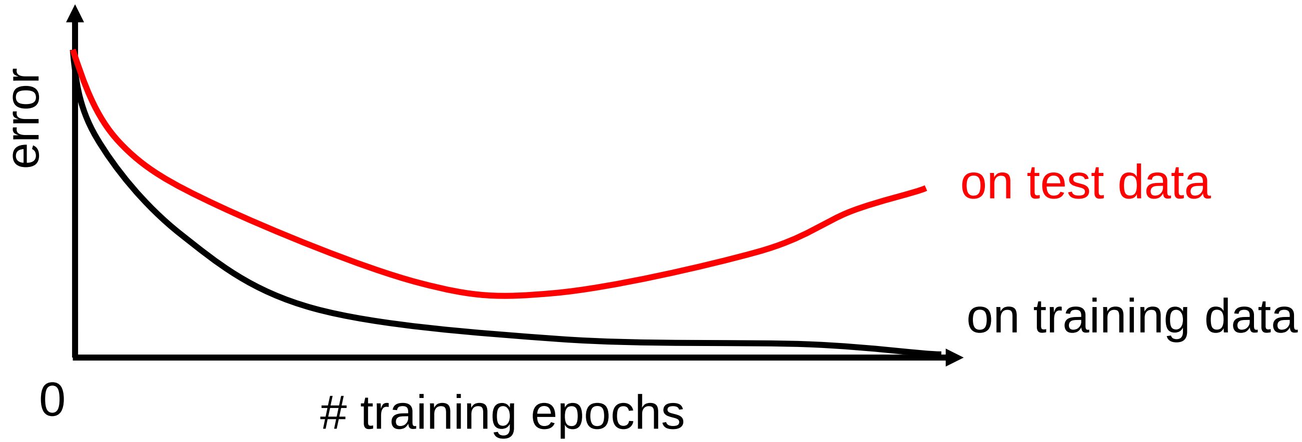


Training: Best practices

- Data
  - Center (subtract mean from) your data
  - Use data augmentation
  - Use mini-batch
- Weights/activations
  - To initialize weights, use “Xavier initialization”
  - Use regularization
  - Use RELU (most common), don’t use sigmoid
- Hyperparameters:
  - Learning rate: too high? Too low?
  - Use cross-validation to pick

Over-training prevention

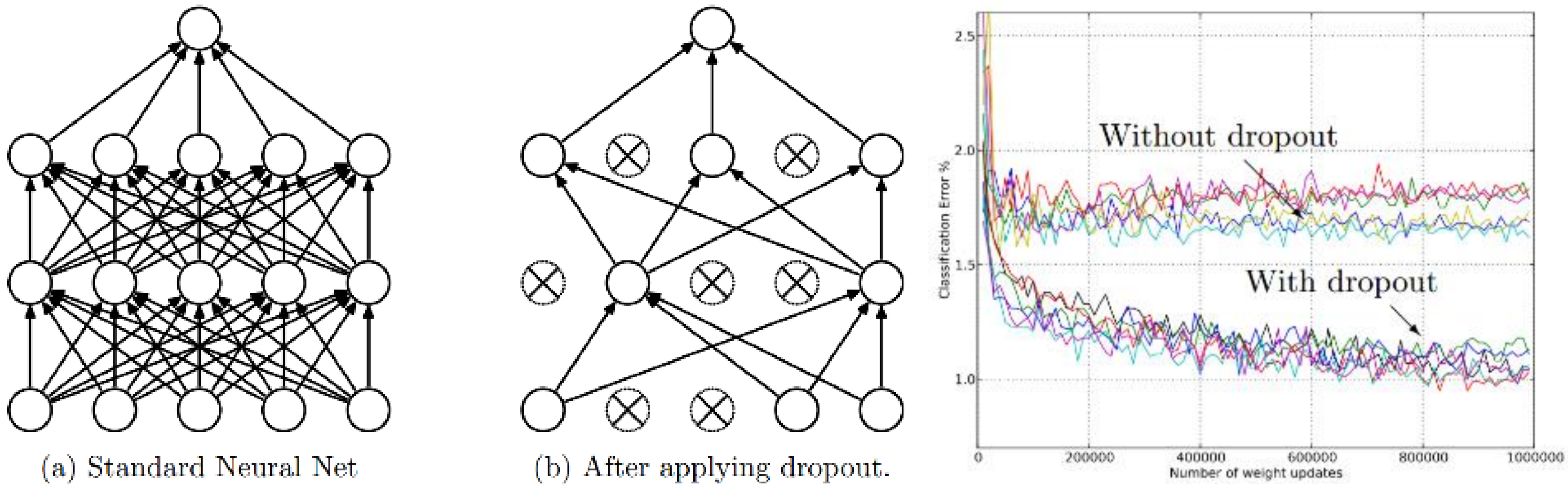
- Running too many epochs can result in over-fitting.



- Keep a hold-out validation set and test accuracy on it after every epoch. Stop training when additional epochs actually increase validation error.

Adapted from Ray Mooney

Regularization: Dropout



- Randomly turn off some neurons
- Allows individual neurons to independently be responsible for performance

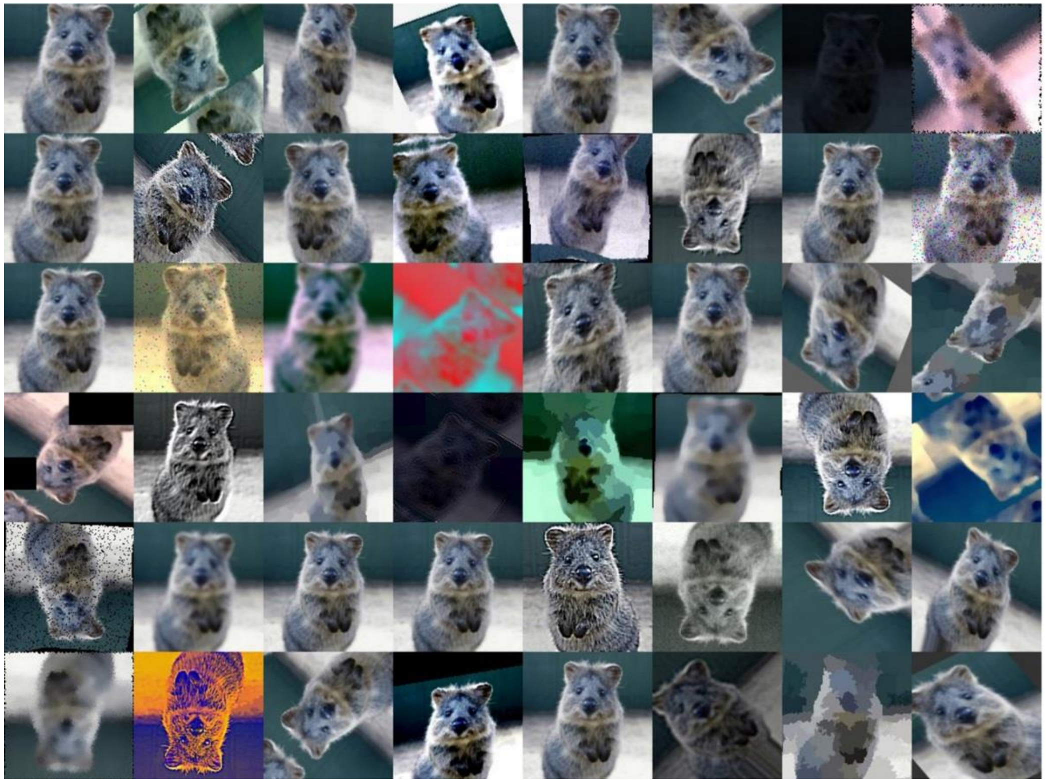
Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Adapted from Jia-bin Huang

Data Augmentation (Jittering)

Create *virtual* training samples

- Horizontal flip
- Random crop
- Color casting
- Geometric distortion



Jia-bin Huang, Image: https://github.com/aleju/imgaug

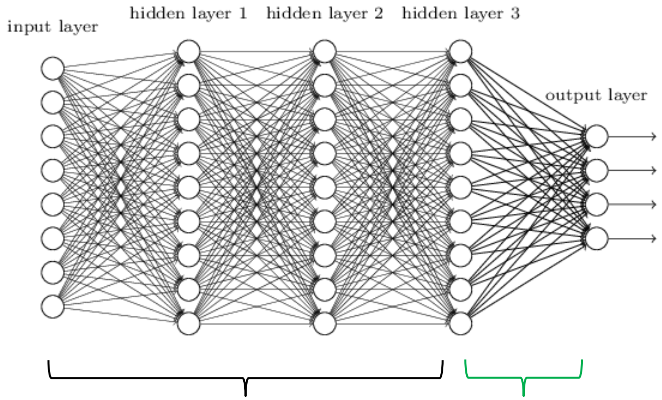
Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

BUSTED

Transfer Learning with CNNs

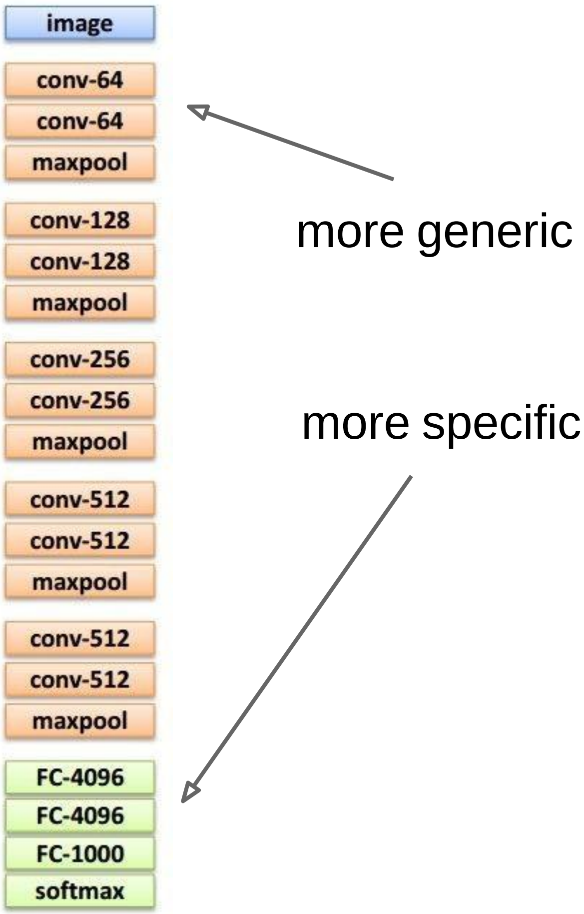
- The more weights you need to learn, the more data you need
- That's why with a deeper network, you need more data to train than for a shallower net
- One possible solution:



Set these to the already learned weights from another network

Learn these on your own task

Transfer Learning with CNNs

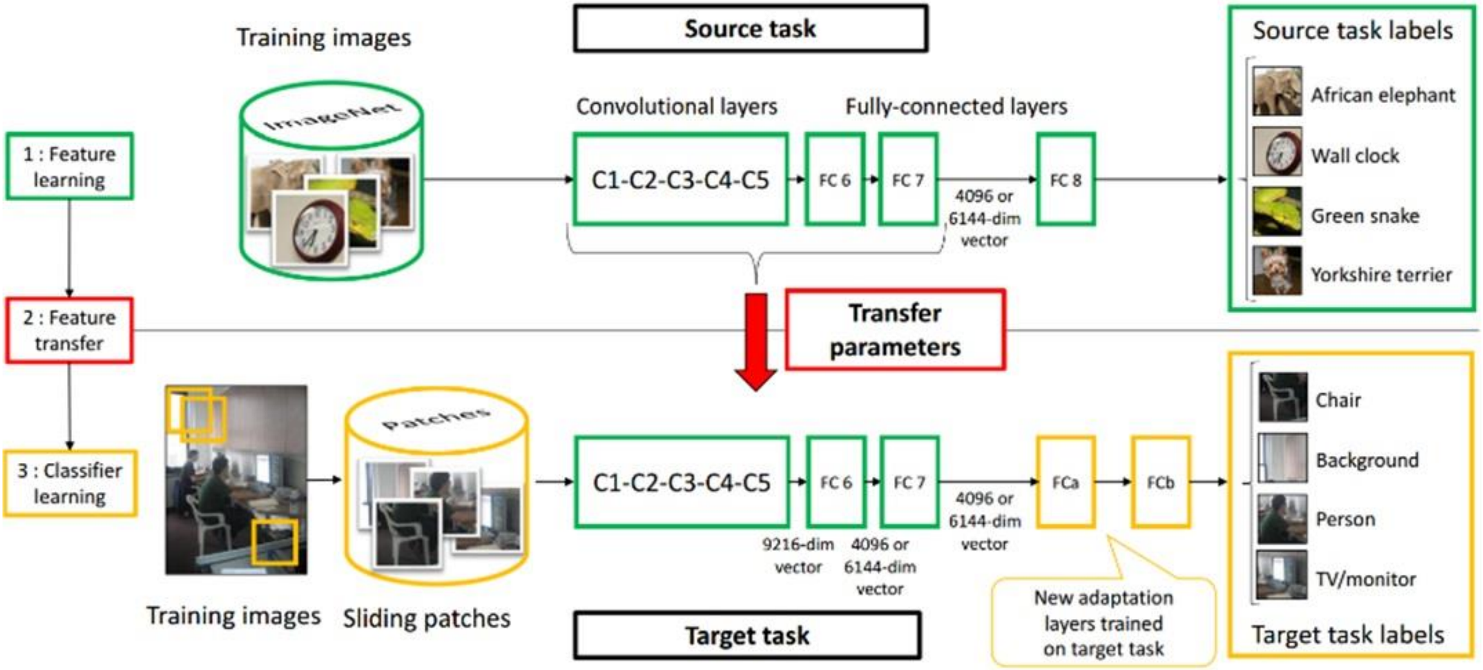


	very similar dataset	**very different dataset**
very little data	Use simple classifier from top layer	You’re in trouble… Try simple classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Adapted from Andrej Karpathy

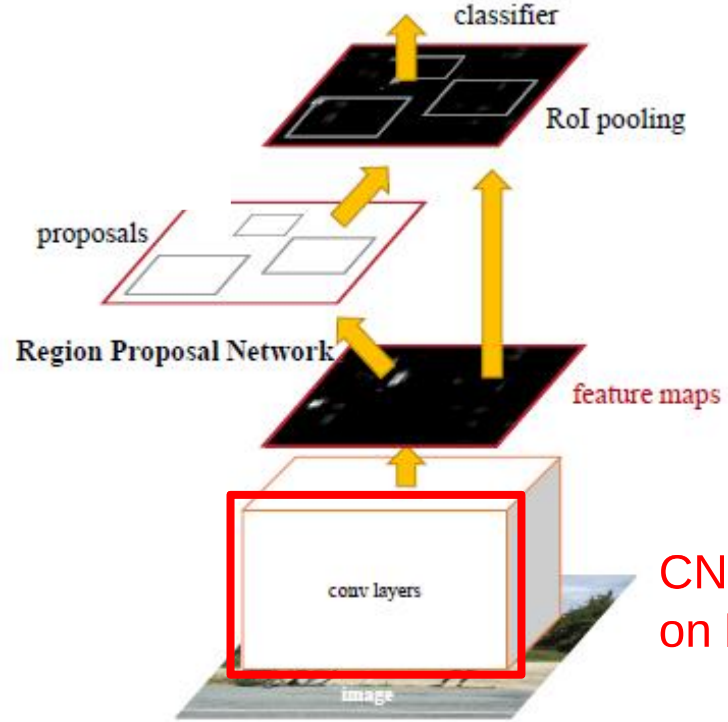
Pre-training on ImageNet

- Have a source domain and target domain
- Train a network to classify ImageNet classes
  - Coarse classes and ones with fine distinctions (dog breeds)
- Remove last layers and train layers to replace them, that predict target classes



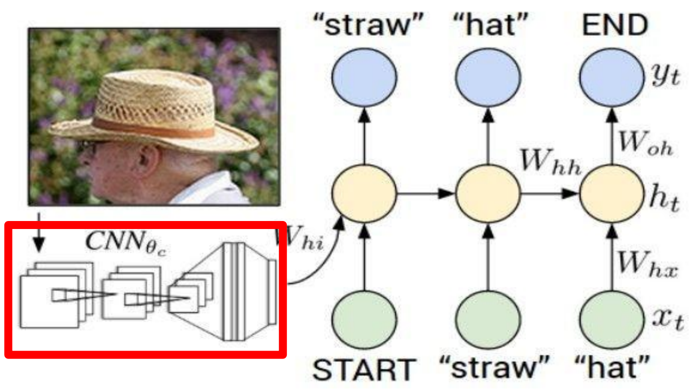
Oquab et al., "Learning and Transferring Mid-Level Image Representations…", CVPR 2014

Transfer learning with CNNs is pervasive…



CNN pretrained
on ImageNet

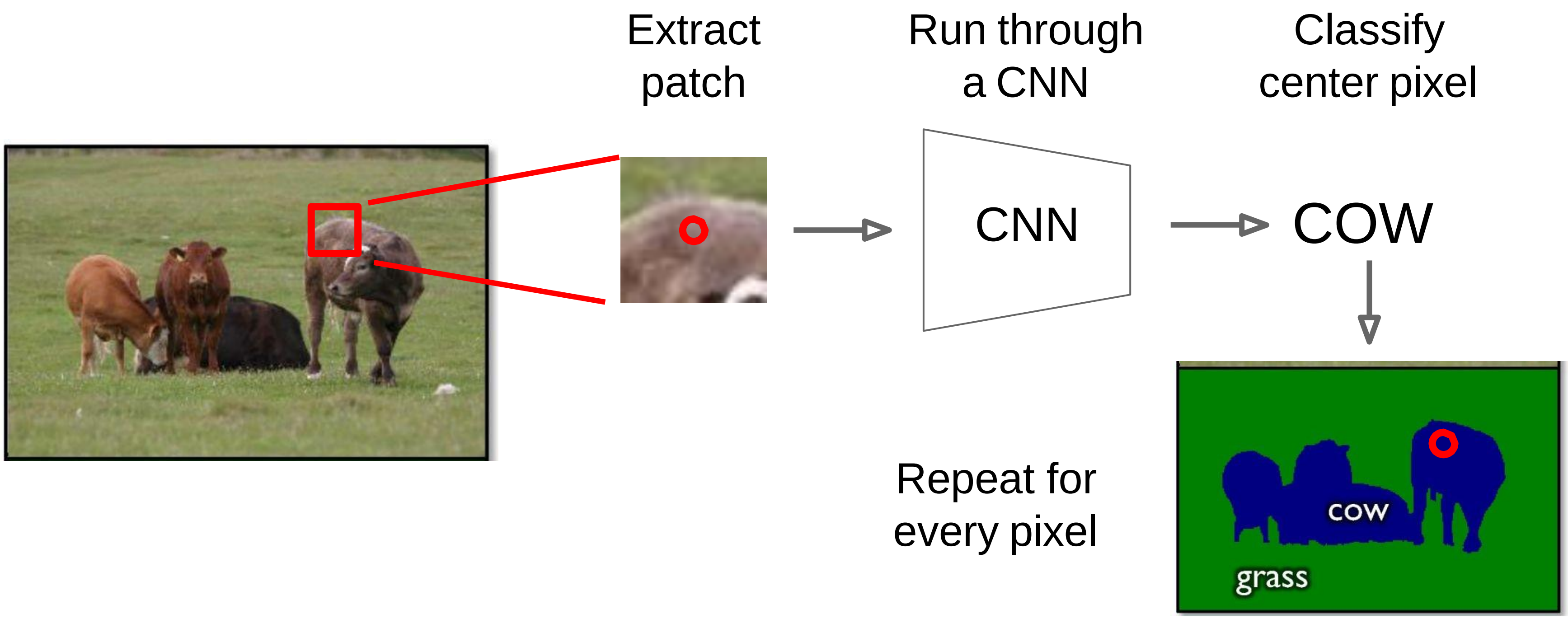
Image Captioning
Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015



Object Detection
Ren et al., "Faster R-CNN", NIPS 2015

Adapted from Andrej Karpathy

Semantic segmentation



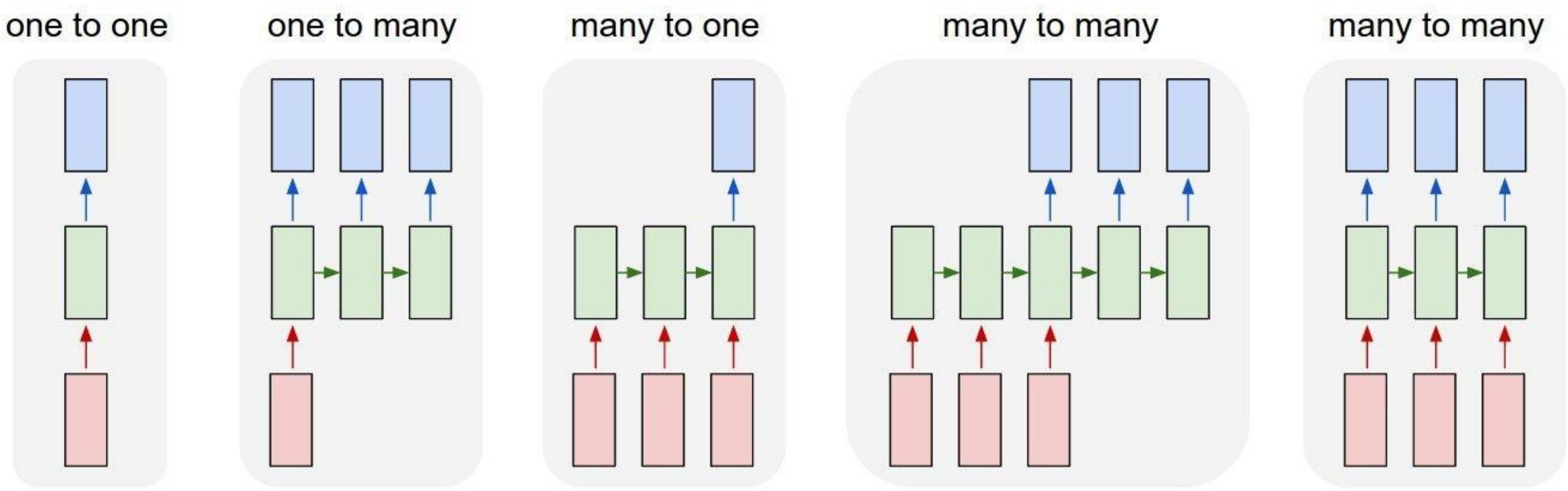
Andrej Karpathy

Analysis of pre-training on ImageNet

- Source:
  - distinguish 1000 ImageNet categories (incl. many dog breeds)
- Target tasks:
  - object detection and action recognition on PASCAL
  - scene recognition on SUN
- Pre-training with 500 images per class is about as good as pre-training with 1000
- Pre-training with 127 classes is about as good as pre-training with 1000
- Pre-training with (fewer classes, more images per class) > (more classes, fewer images)
- Small drop in if classes with fine-grained distinctions removed from pre-training set

Huh et al., "What makes ImageNet good for transfer learning?", arxiv 2016

Recurrent Networks offer a lot of flexibility:



e.g. **image captioning**
image -> sequence of words

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

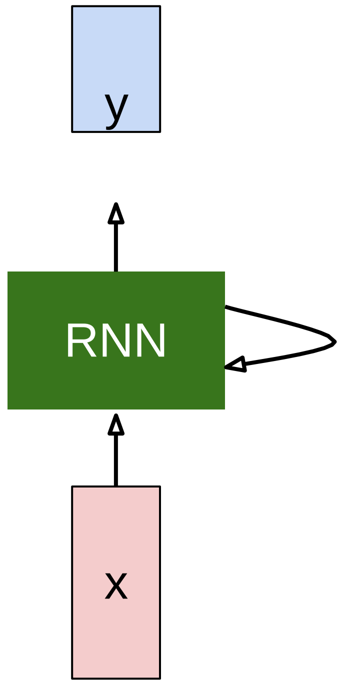
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W



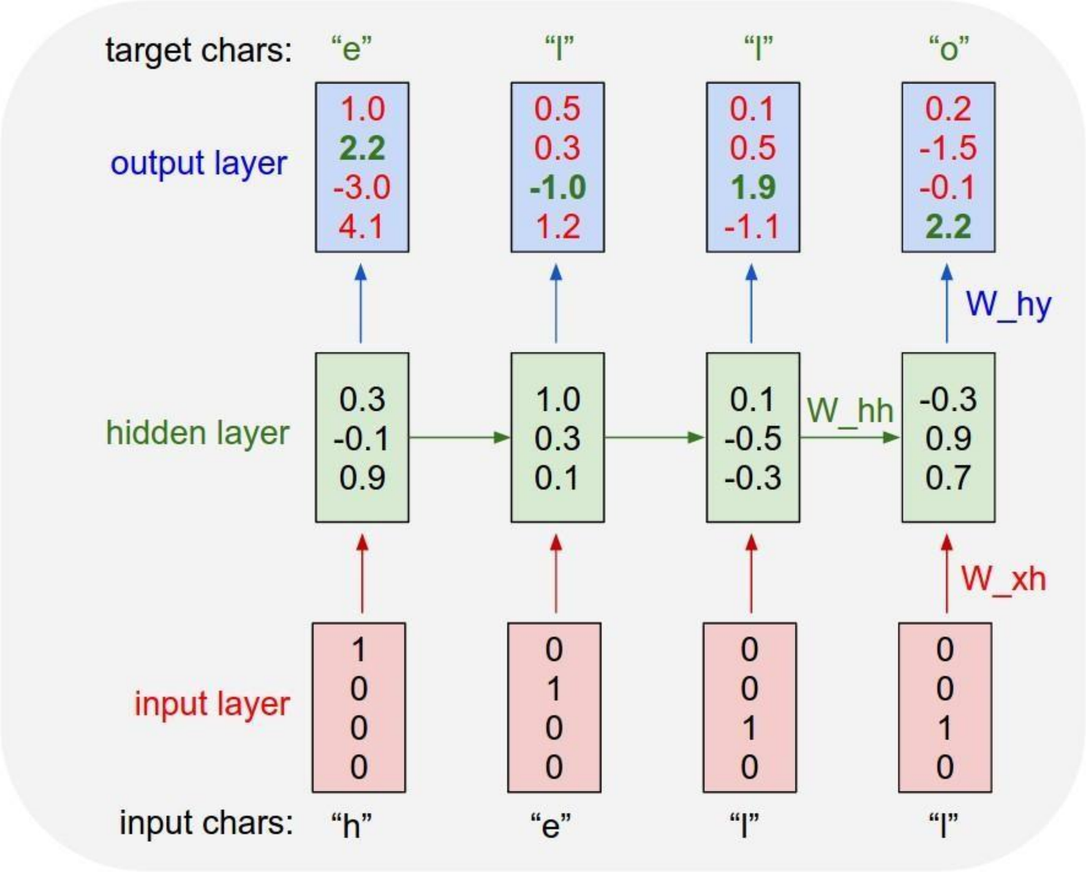
Andrej Karpathy

Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
"hello"

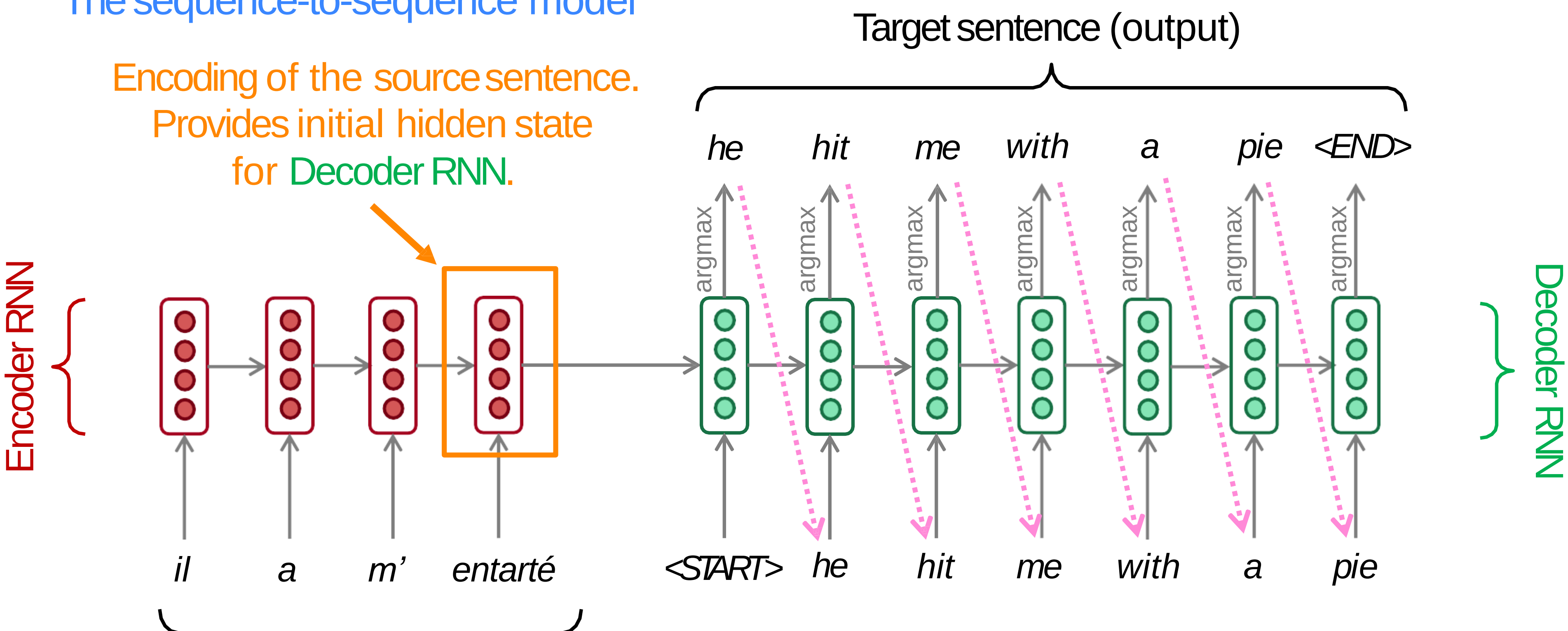


What do we still need to specify, for this to work?

What kind of loss can we formulate?

Neural Machine Translation (NMT)

The sequence-to-sequence model



Source sentence (input)

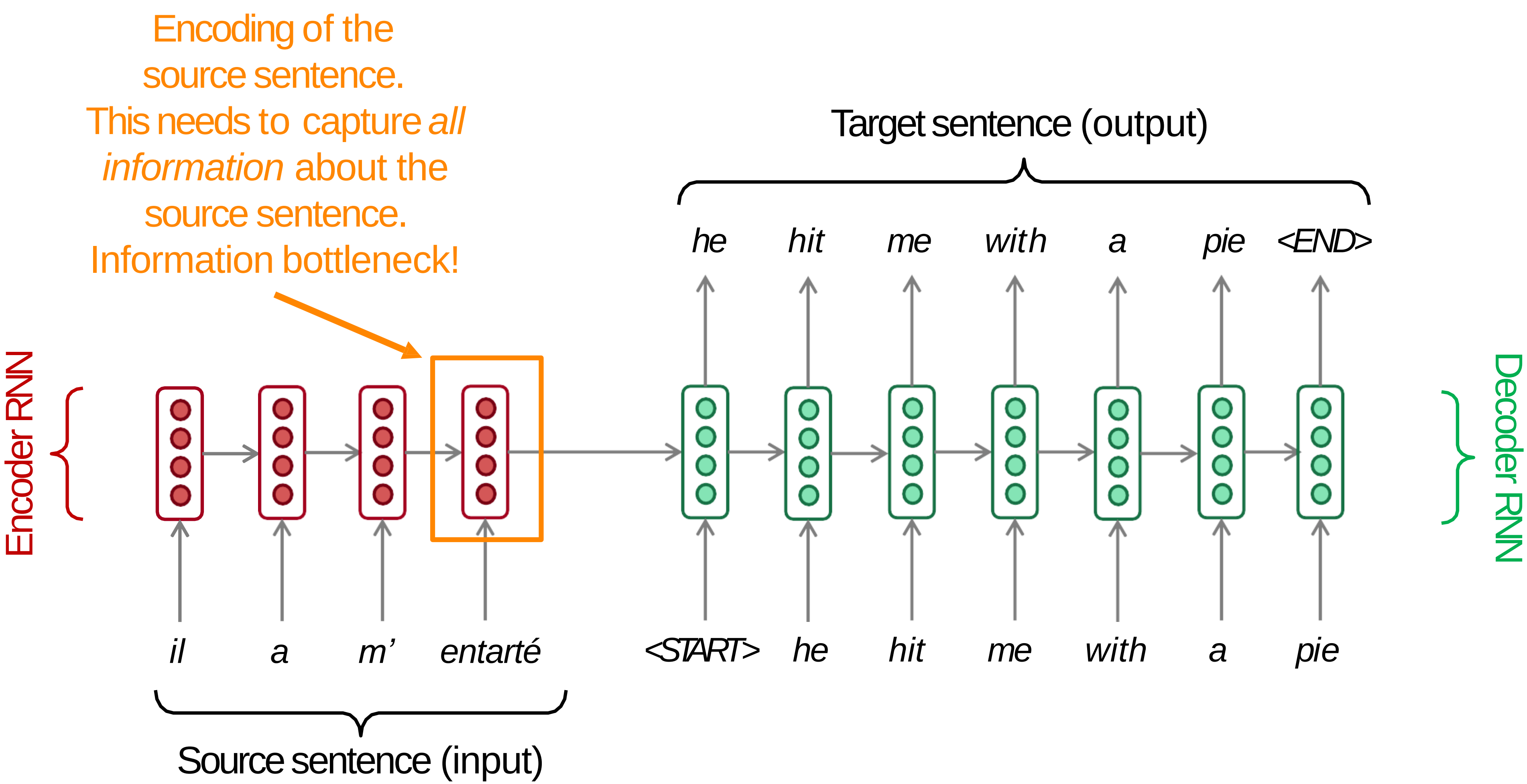
Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*.

Encoder RNN produces an encoding of the source sentence.

Note: This diagram shows **test time** behavior: decoder output is fed in ······> as next step's input

Abigail See

Sequence-to-sequence: the bottleneck problem



Abigail See

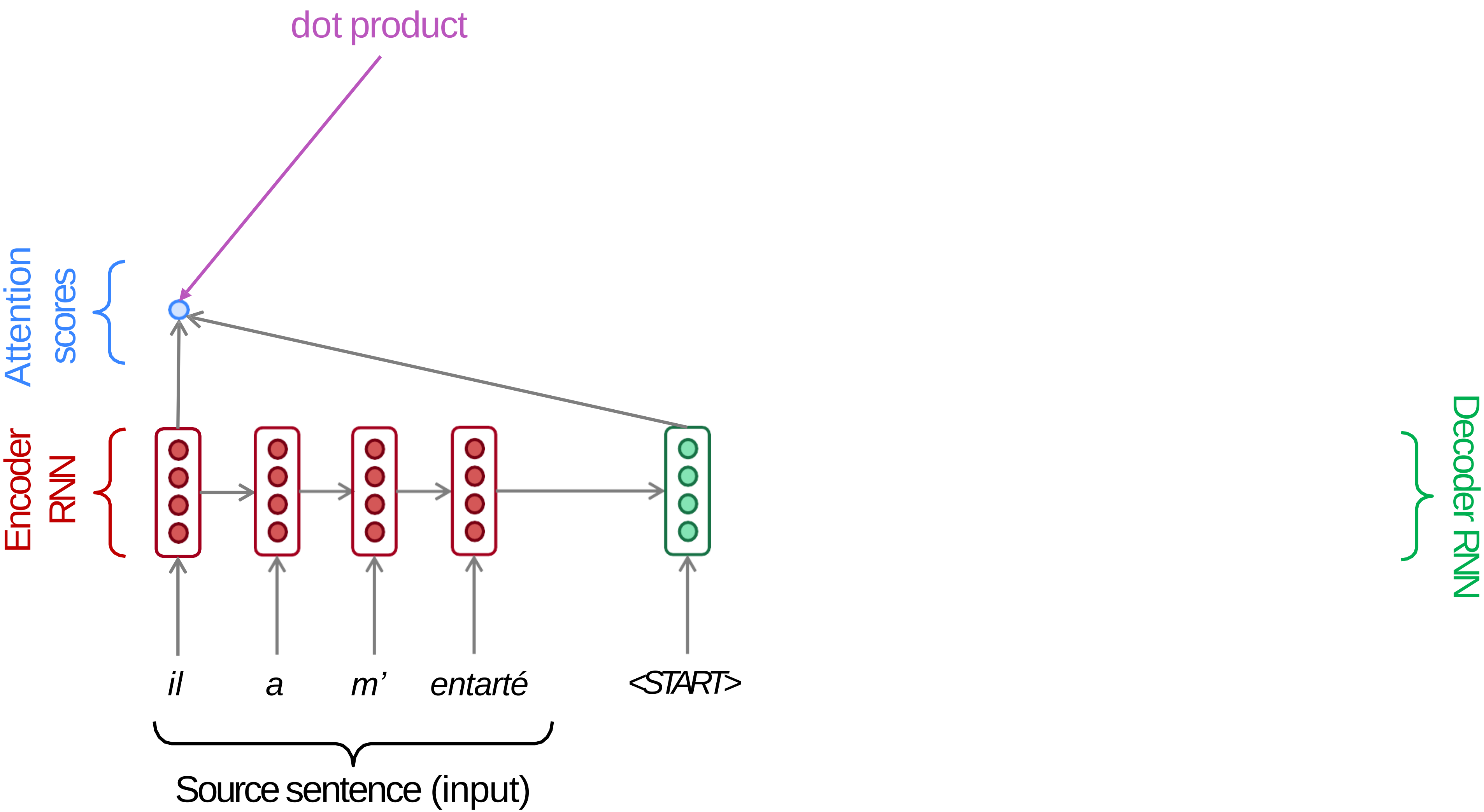
Attention

- **Attention** provides a solution to the bottleneck problem.

- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

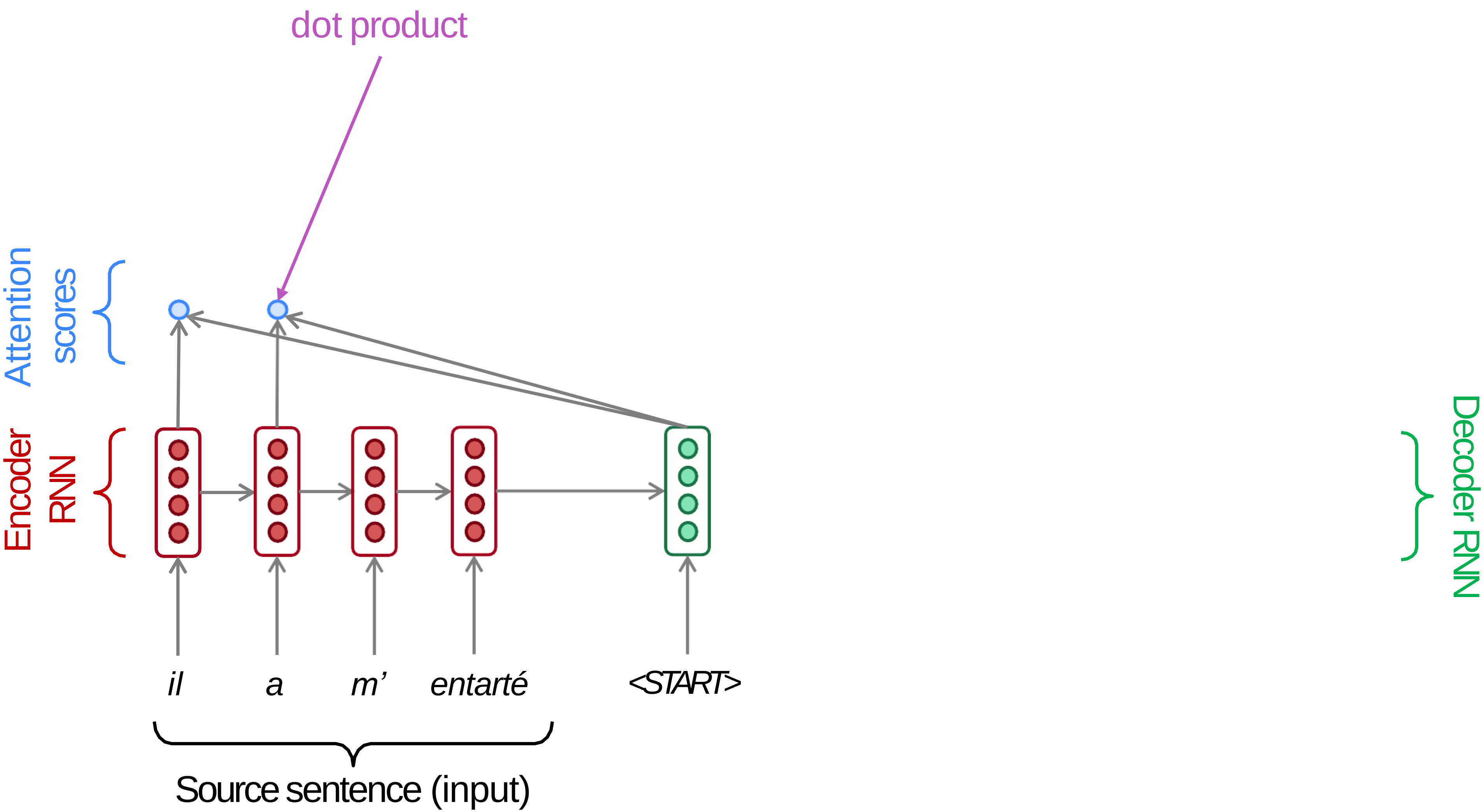
- First we will show via diagram (no equations), then we will show with equations

Sequence-to-sequence with attention



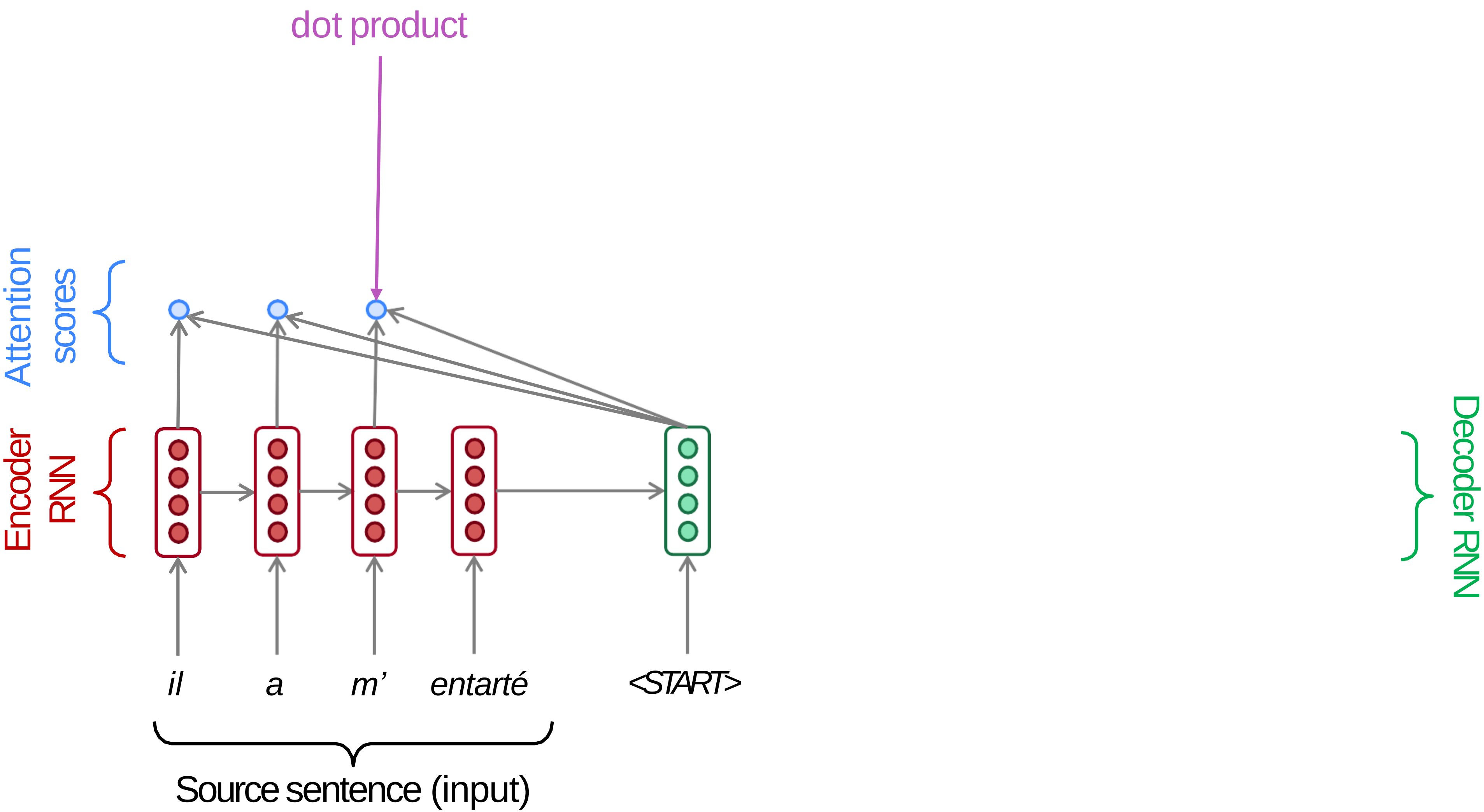
Abigail See

Sequence-to-sequence with attention



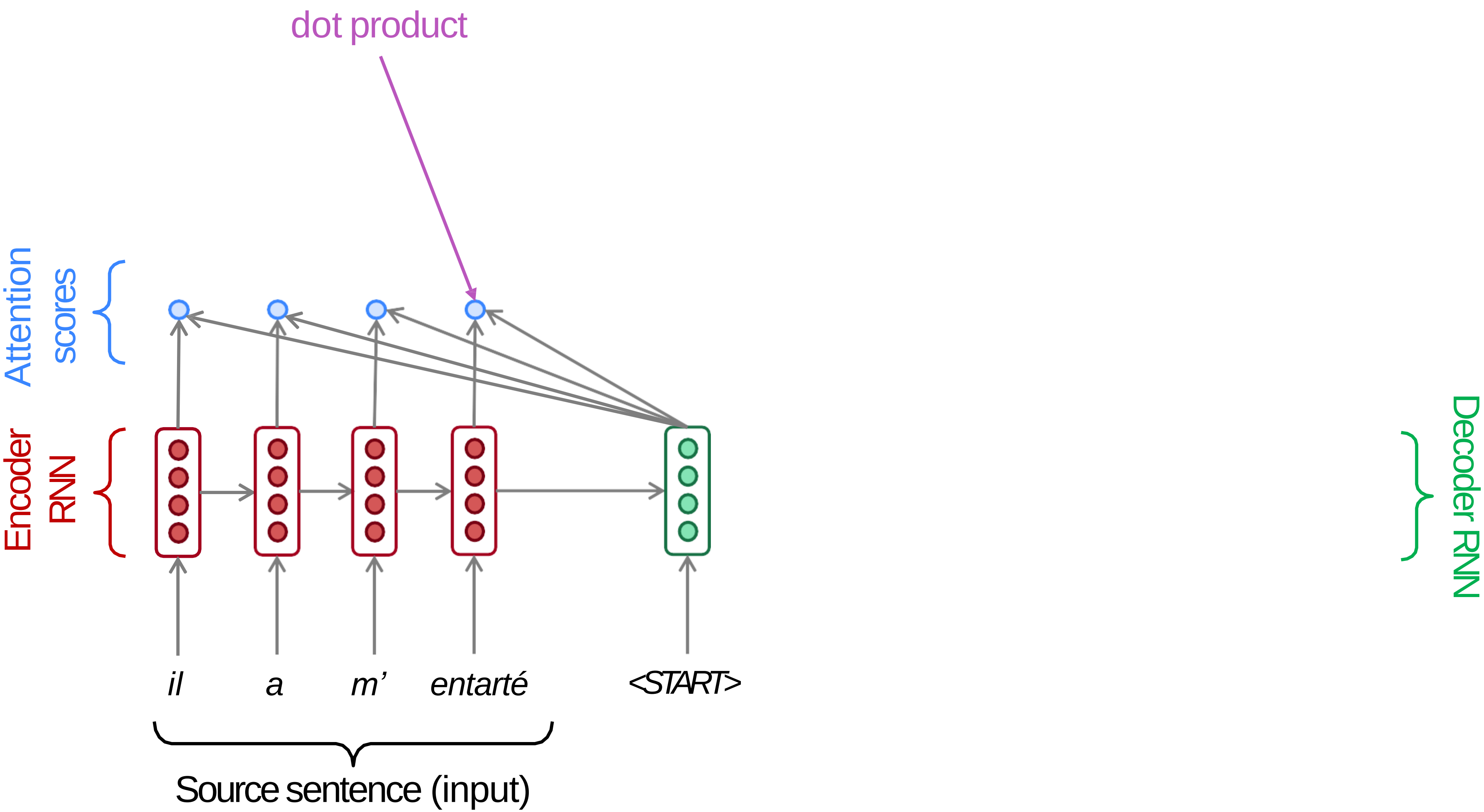
Abigail See

Sequence-to-sequence with attention



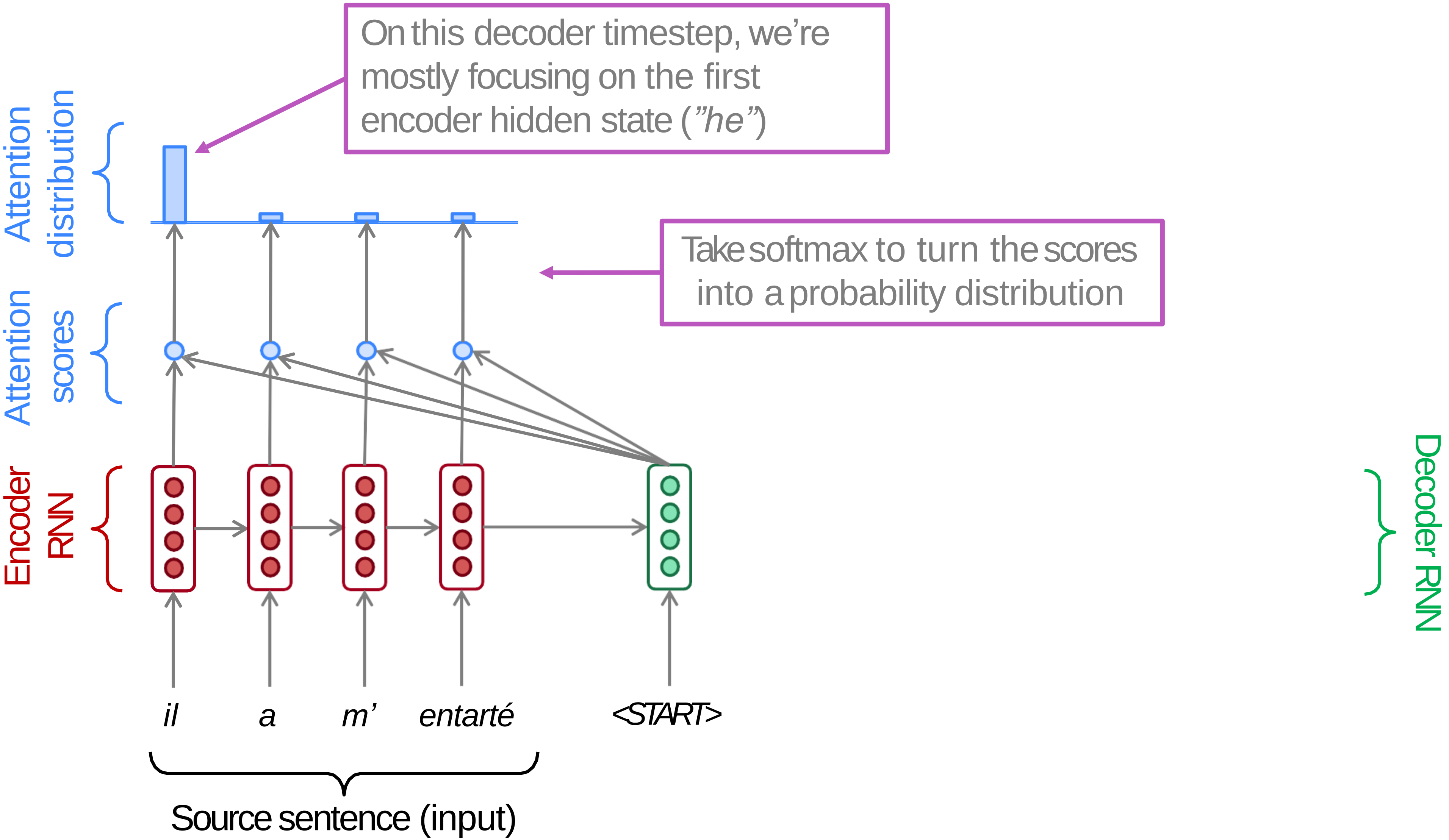
Abigail See

Sequence-to-sequence with attention



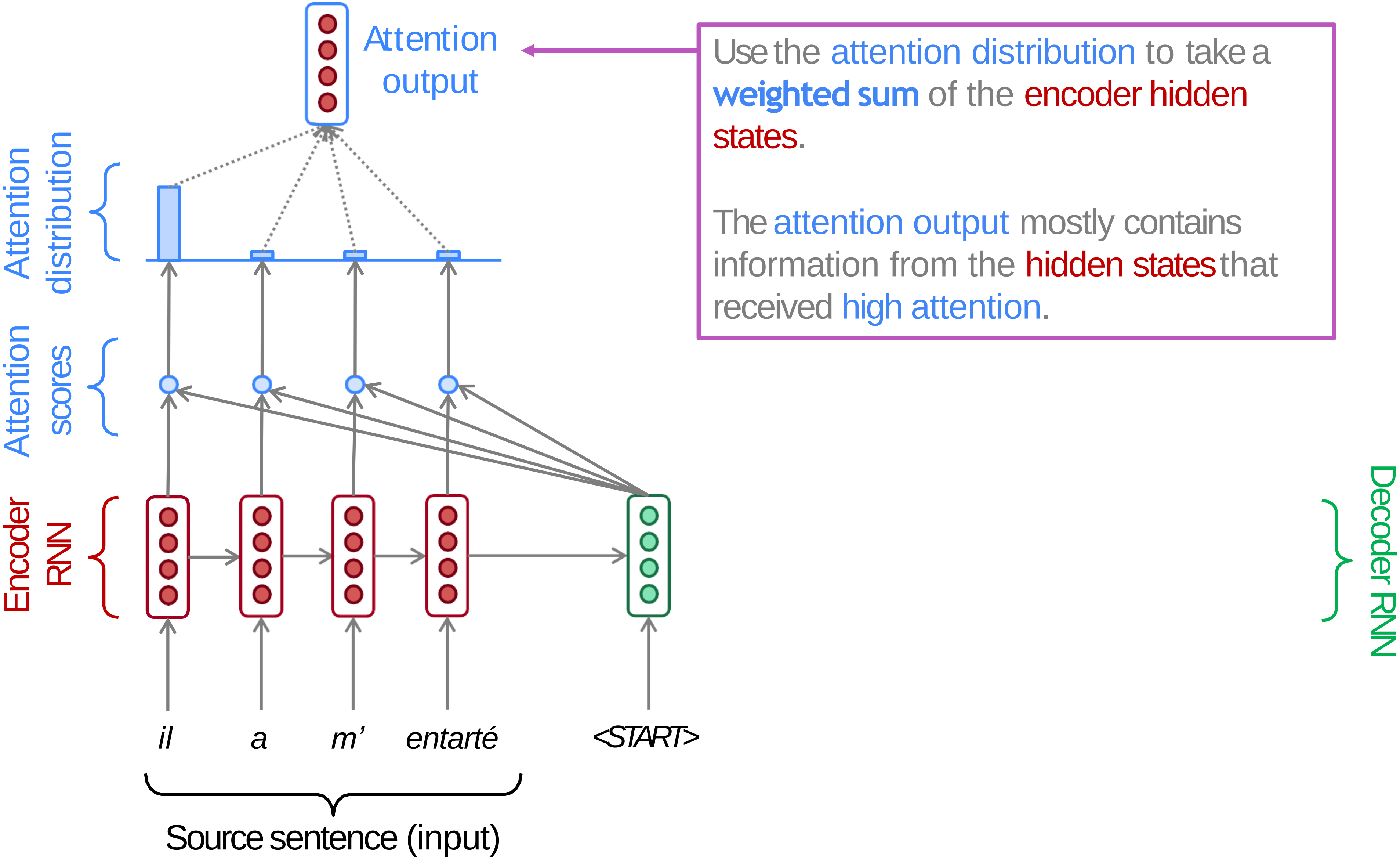
Abigail See

Sequence-to-sequence with attention



Abigail See

Sequence-to-sequence with attention

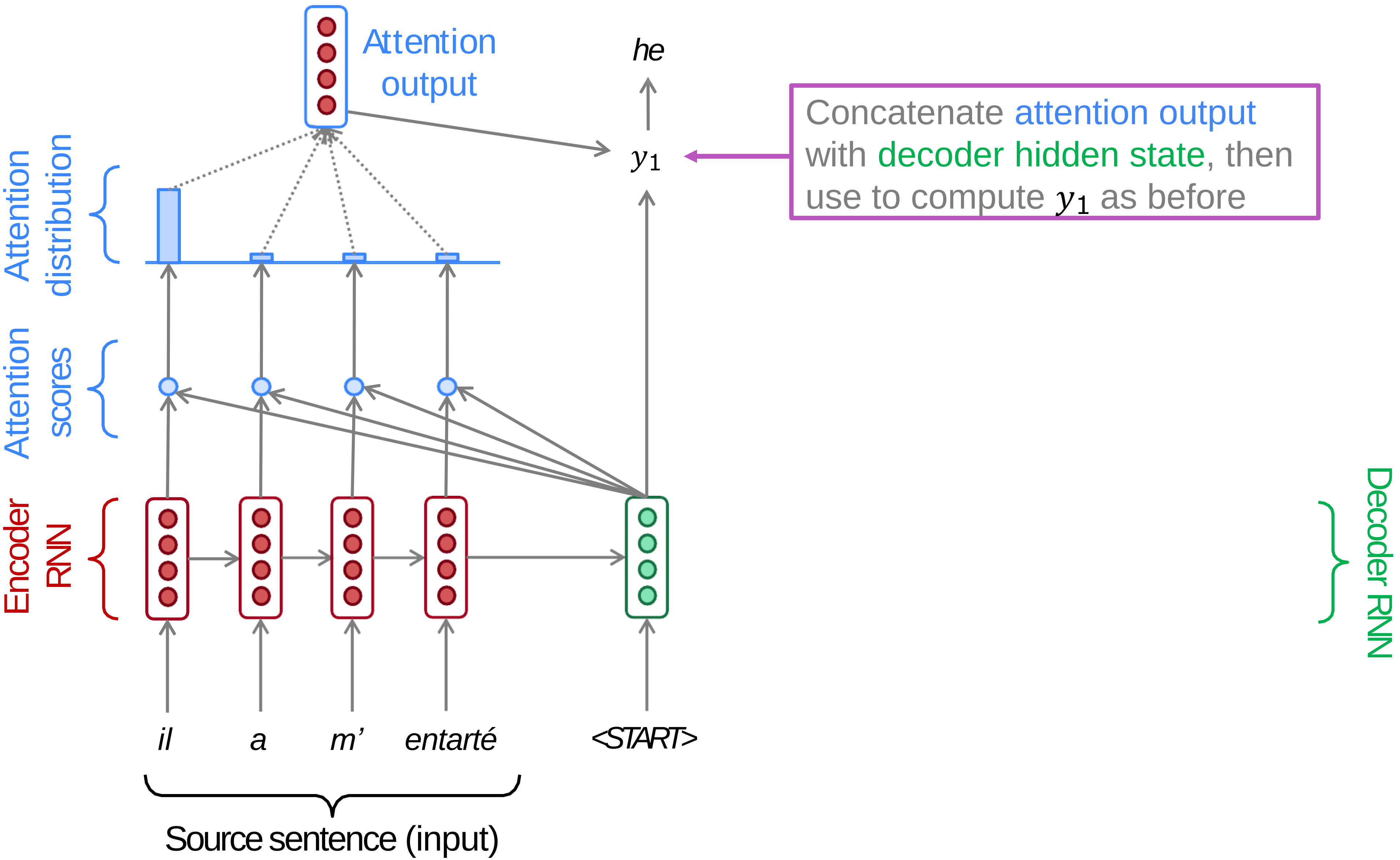


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

Abigail See

Sequence-to-sequence with attention



Concatenate attention output with decoder hidden state, then use to compute y_1 as before

Abigail See

Attention: in equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention score $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \mathrm{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

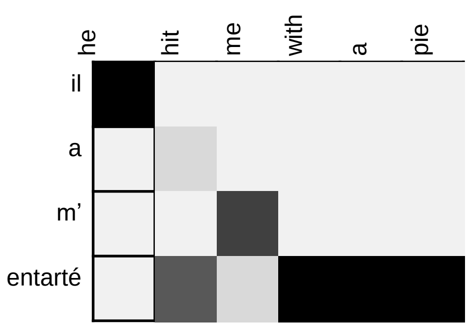
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is great

- Attention significantly improves NMT performance
  - It’s very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



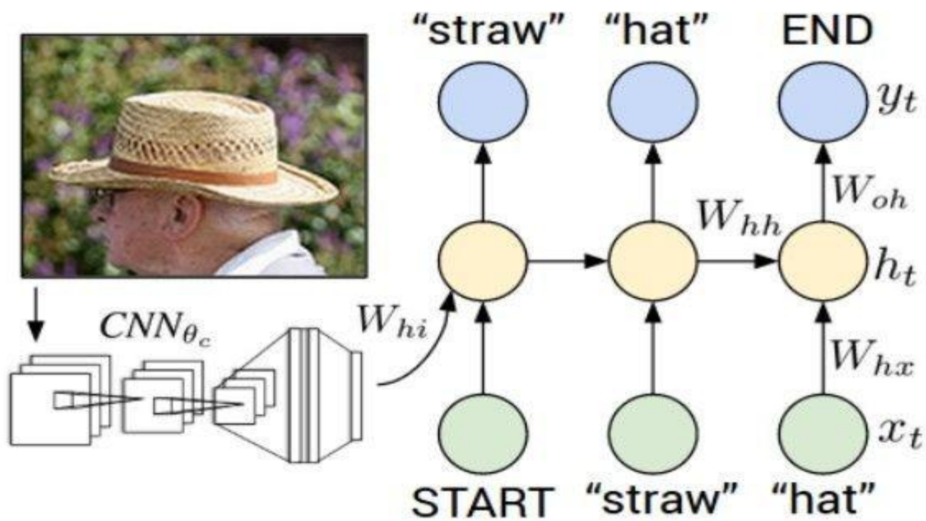
Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- <u>However</u>: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

- **<u>More general definition of attention</u>**:
  - Given a set of vector *values*, and a vector *query*, **<u>attention</u>** is a technique to compute a weighted sum of the values, dependent on the query.

- We sometimes say that the query *attends to* the values.
- For example, in seq2seq + attention model, each decoder hidden state (query) *attends to* all encoder hidden states (values).

Image Captioning

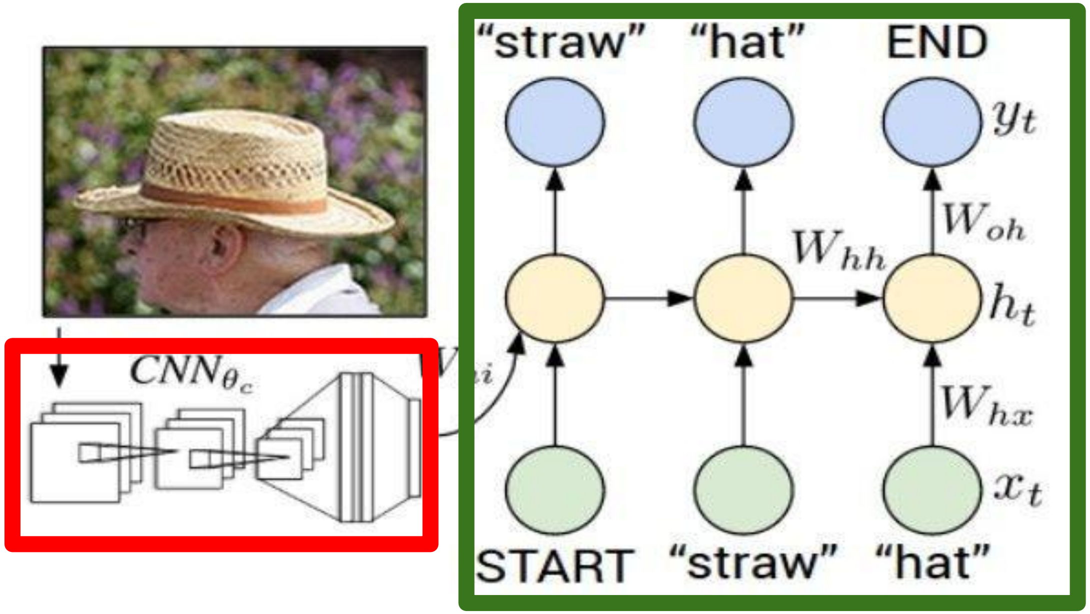


CVPR 2015:
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Adapted from Andrej Karpathy

Image Captioning

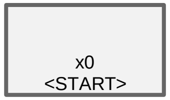
Recurrent Neural Network



Convolutional Neural Network

Image Captioning

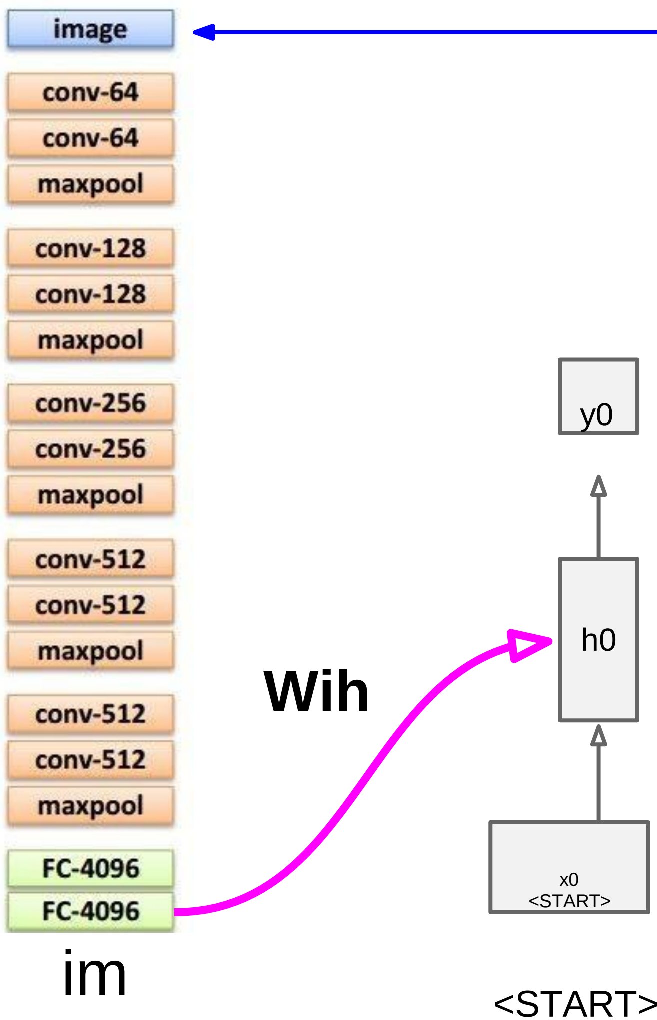
test image



<START>

Andrej Karpathy

Image Captioning



test image

before:

$h = \tanh(W_{xh} * x + W_{hh} * h)$

now:

$h = \tanh(W_{xh} * x + W_{hh} * h + \mathbf{W_{ih} * im})$

Andrej Karpathy

Image Captioning

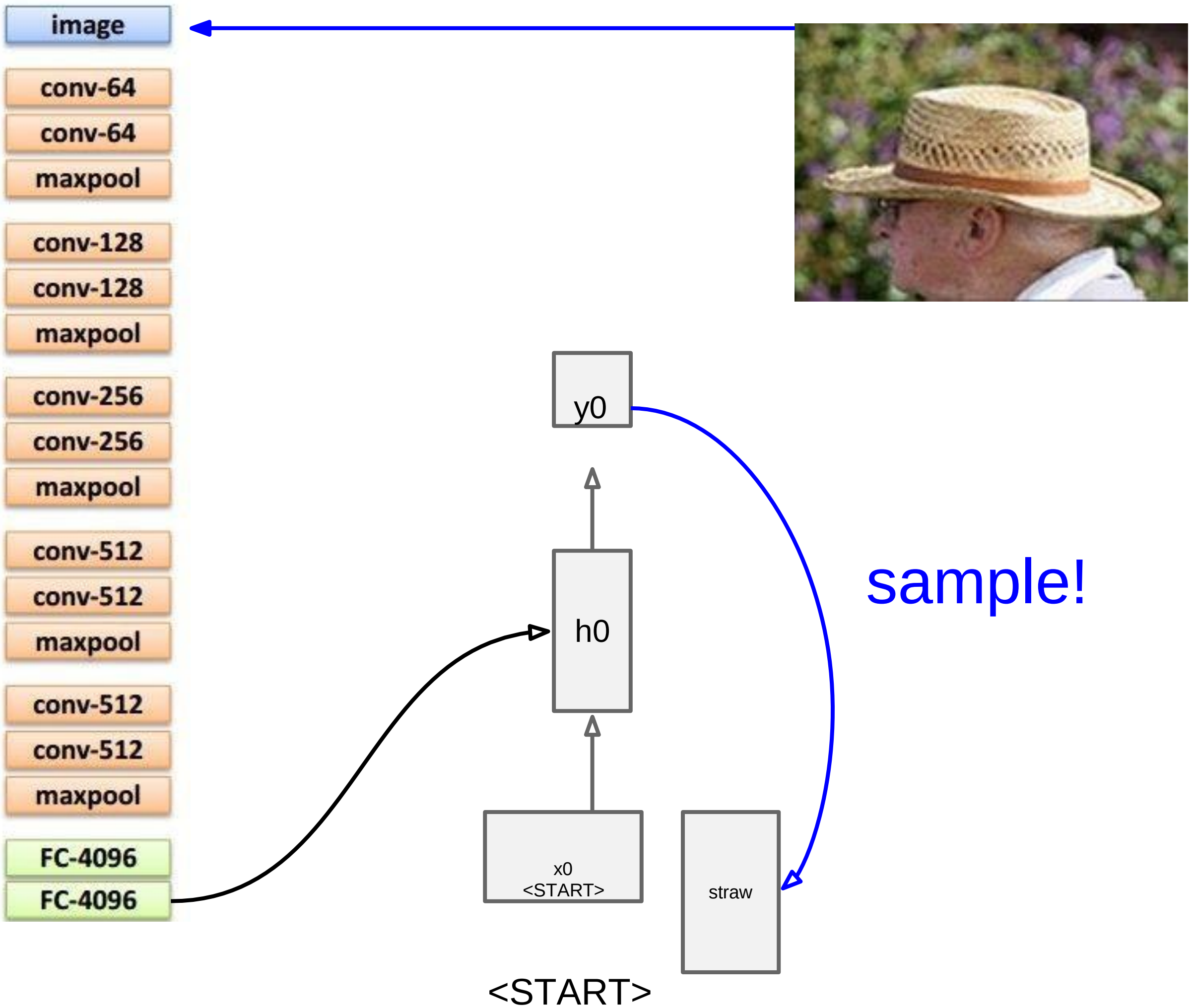
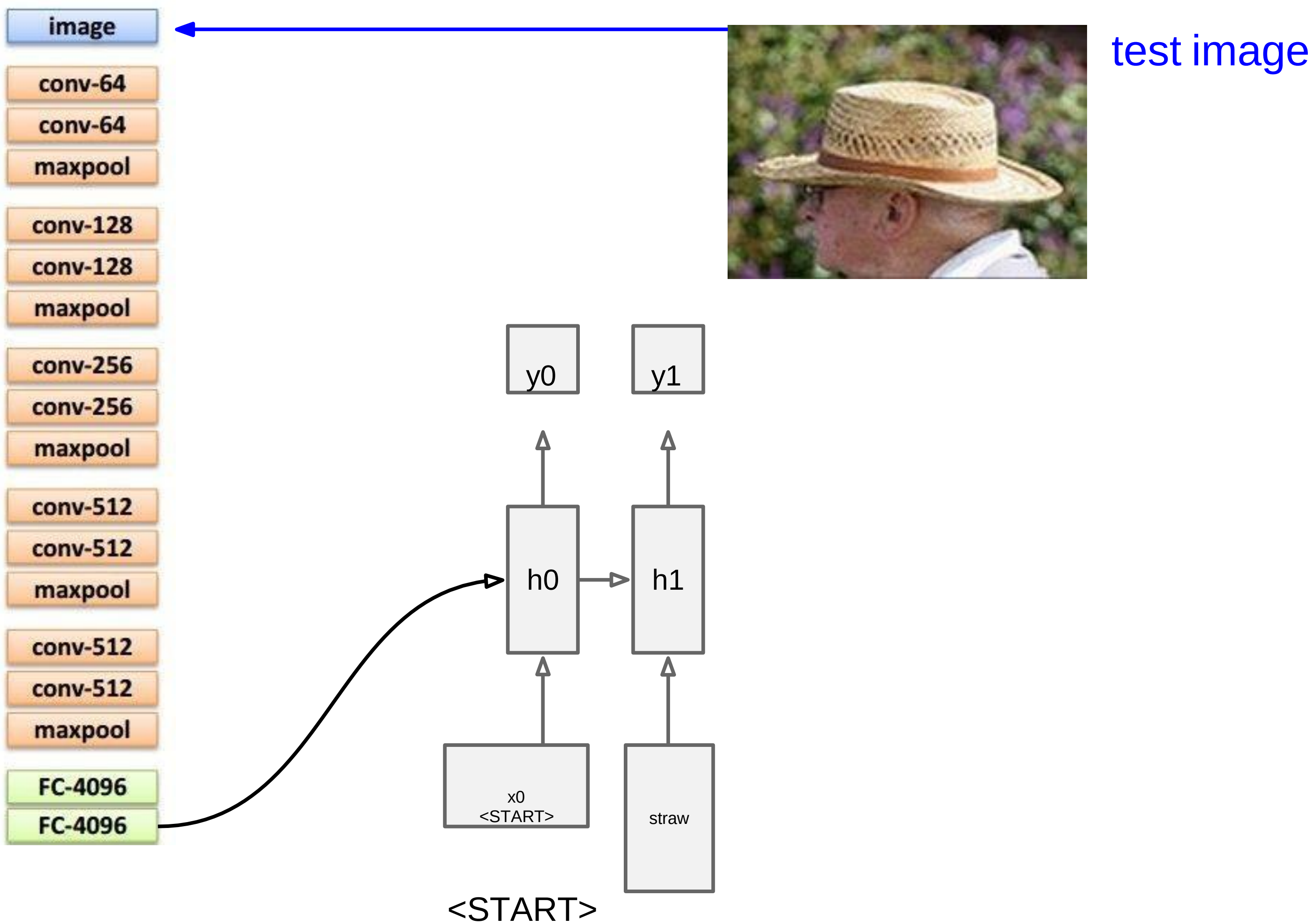
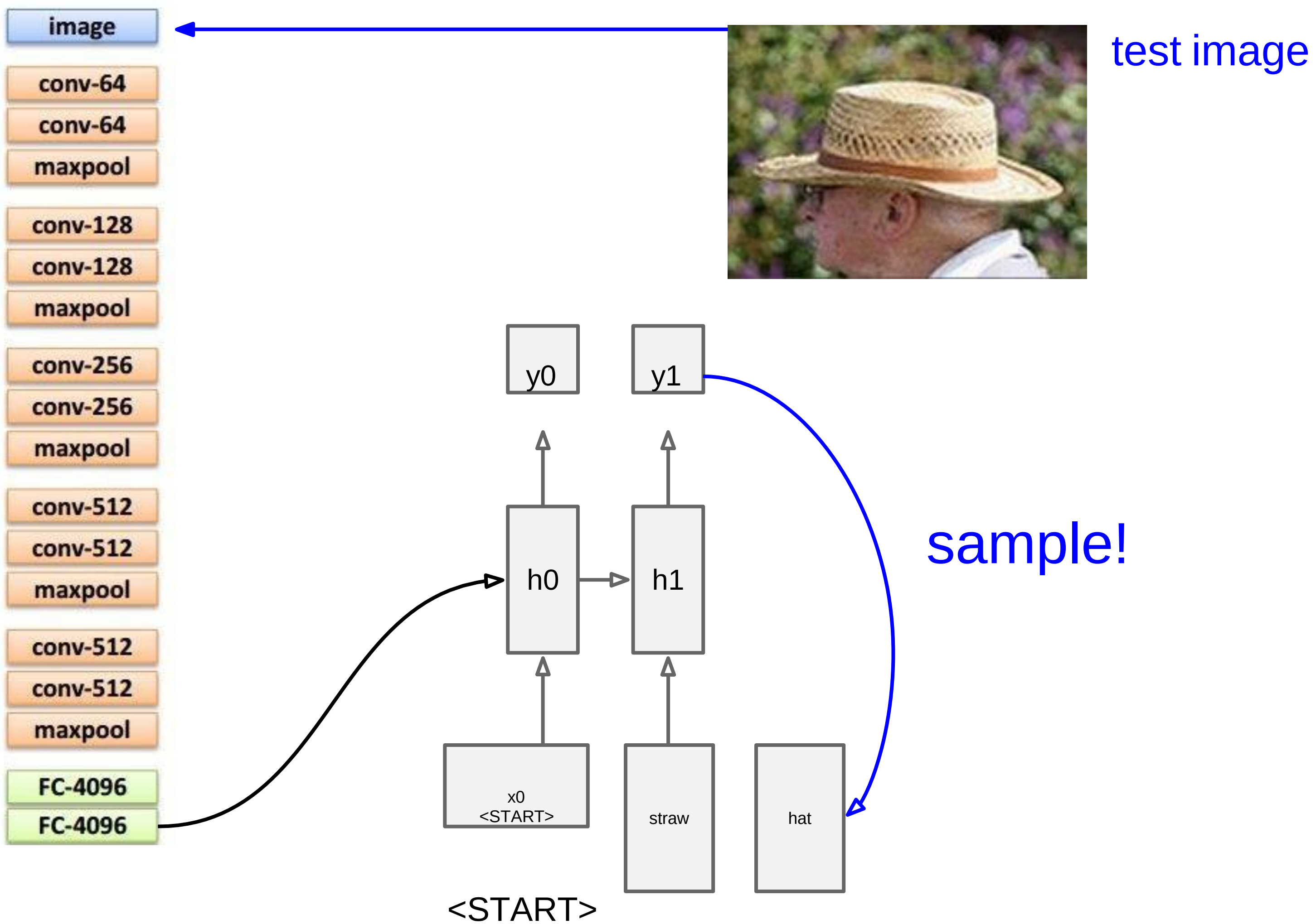


Image Captioning



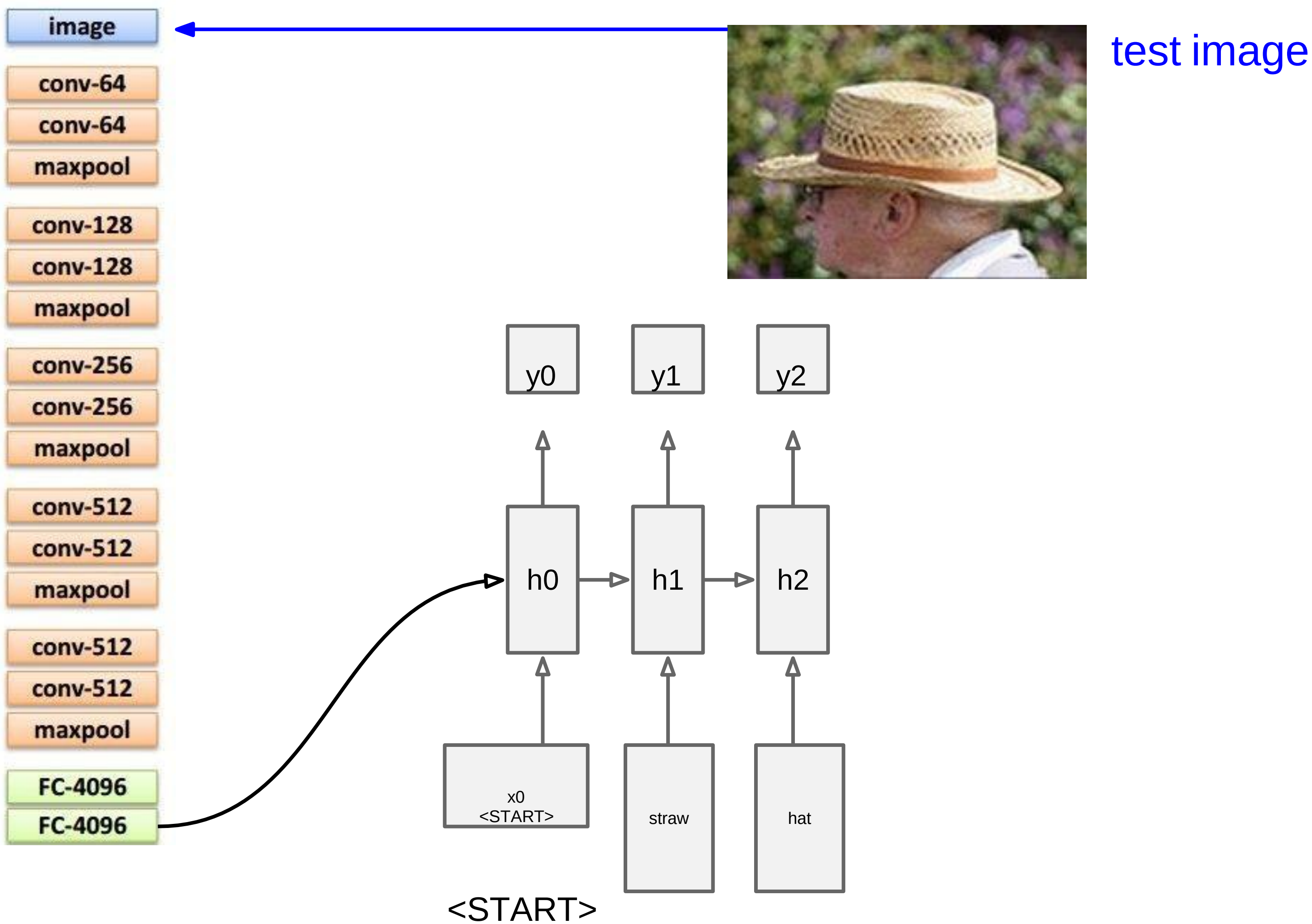
Andrej Karpathy

Image Captioning



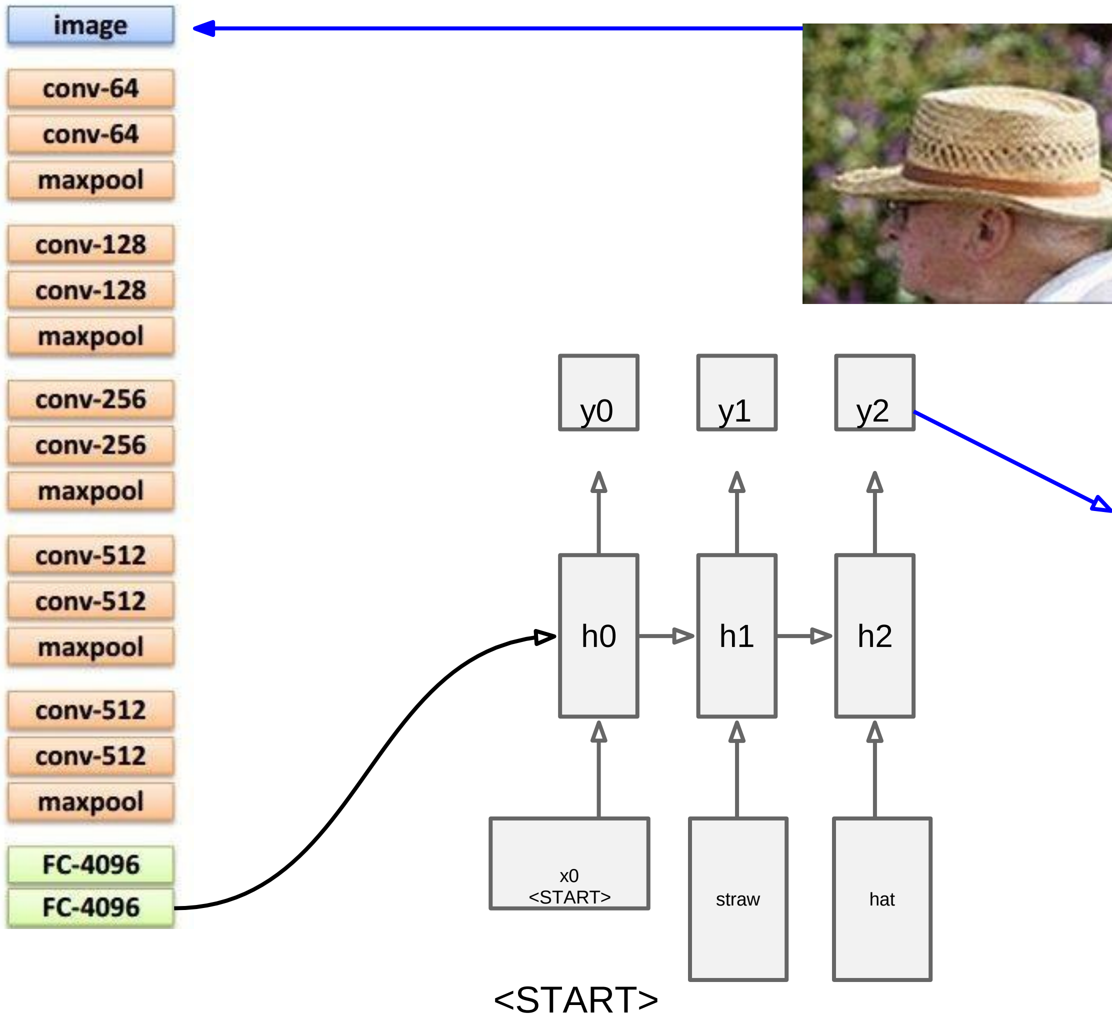
Andrej Karpathy

Image Captioning



Andrej Karpathy

Image Captioning



test image

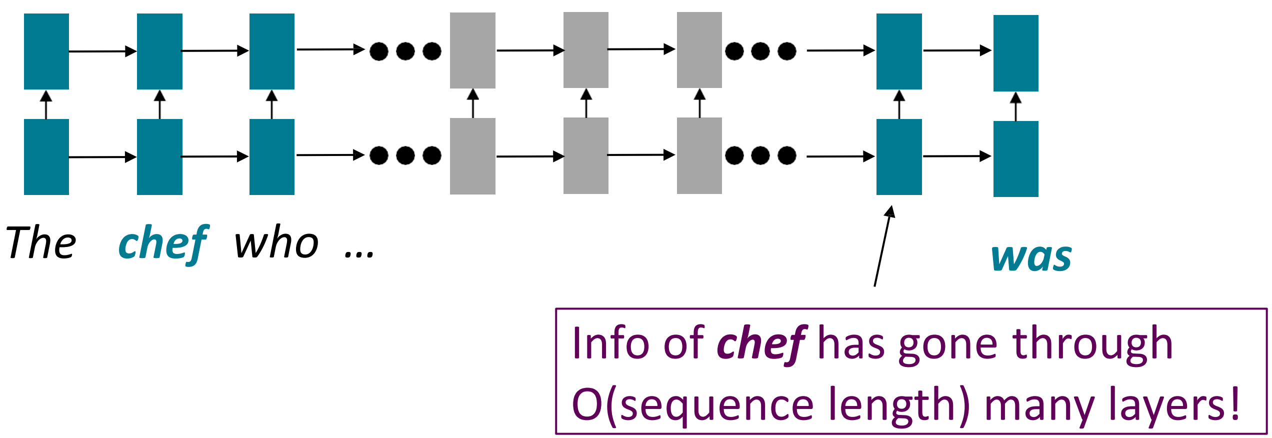
Caption generated:
“straw hat”

sample
<END> token
=> finish.

Adapted from Andrej Karpathy

Issues with recurrent models:
Linear interaction distance

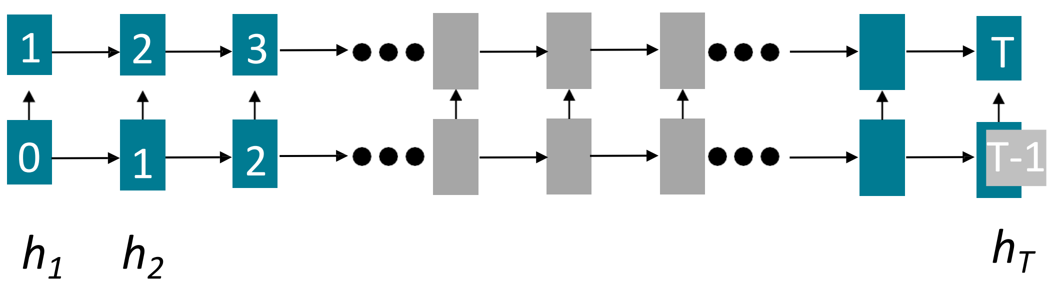
- **O(sequence length)** steps for distant word pairs to interact means:
  - Hard to learn long-distance dependencies (because gradient problems!)
  - Linear order of words is “baked in”; not necessarily the right way to think about sentences…



Adapted from John Hewitt

Issues with recurrent models:
Lack of parallelizability

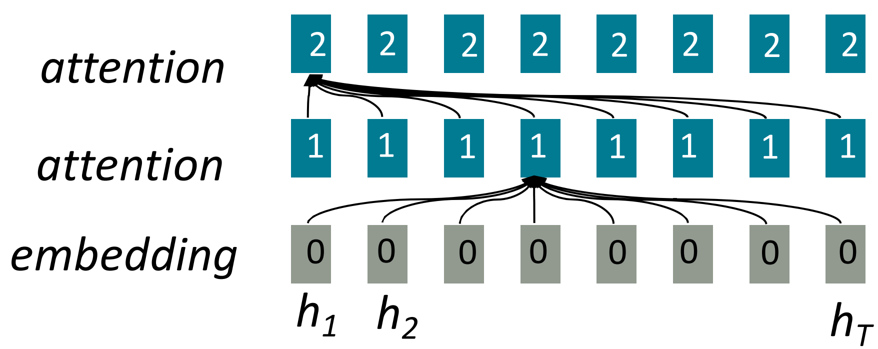
- Forward and backward passes have **O(sequence length)** unparallelizable operations
  - GPUs can perform a bunch of independent computations at once!
  - But future RNN hidden states can't be computed in full before past RNN hidden states have been computed
  - Inhibits training on very large datasets!



Numbers indicate min # of steps before a state can be computed

If not recurrence, then what?
How about attention?

- **Attention** treats each word's representation as a **query** to access and incorporate information from **a set of values.**
  - We saw attention from the **decoder** to the **encoder**; today we'll think about attention **within a single sentence**.
    - If **attention** gives us access to any state… maybe we can just use attention and don't need the RNN?
- Number of unparallelizable operations not tied to sequence length.
- All words interact at every layer!



All words attend to all words in previous layer; most arrows here are omitted

Adapted from John Hewitt

Self-Attention

- Attention operates on **queries**, **keys**, and **values.**
  - We have some **queries** $q_1, q_2, \ldots, q_T$. Each query is $q_i \in \mathbb{R}^d$
  - We have some **keys** $k_1, k_2, \ldots, k_T$. Each key is $k_i \in \mathbb{R}^d$
  - We have some **values** $v_1, v_2, \ldots, v_T$. Each value is $v_i \in \mathbb{R}^d$

The number of queries can differ from the number of keys and values in practice.

- In **self-attention**, the queries, keys, and values are drawn from the same source.
  - For example, if the output of the previous layer is $x_1, \ldots, x_T$, (one vec per word) we could let $v_i = k_i = q_i = x_i$ (that is, use the same vectors for all of them!)
- The (dot product) self-attention operation is as follows:

$$e_{ij} = q_i^\top k_j$$

Compute **key-query** affinities

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

Compute attention weights from affinities (softmax)

$$\text{output}_i = \sum_j \alpha_{ij} v_j$$

Compute outputs as weighted sum of **values**

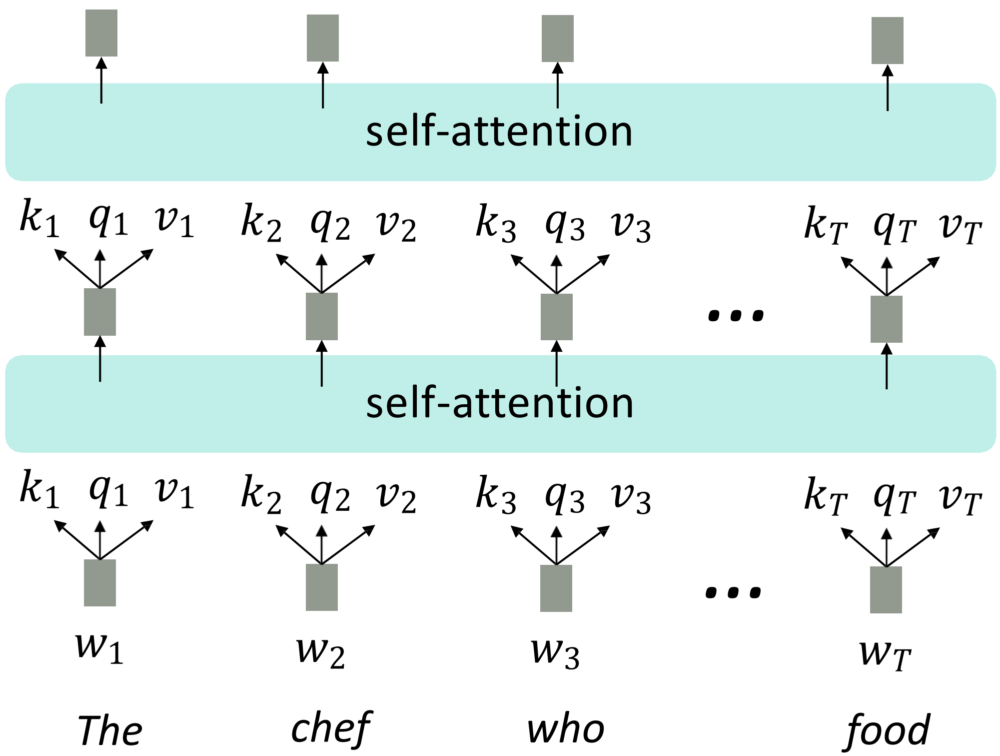
Self-Attention

- In the diagram at the right, we have stacked self-attention blocks, like we might stack LSTM layers.

- Can self-attention be a drop-in replacement for recurrence?

- No. It has a few issues, which we'll go through.

- First, self-attention is an operation on **sets**. It has no inherent notion of order.



Self-attention doesn't know the order of its inputs.

Barriers and solutions for Self-Attention as a building block

Barriers

- Doesn't have an inherent notion of order!
- No nonlinearities for deep learning magic! It's all just weighted averages
- Need to ensure we don't "look at the future" when predicting a sequence
  - Like in machine translation
  - Or language modeling

Solutions

- Add position representations to the inputs
- Easy fix: apply the same feedforward network to each self- attention output.
- Mask out the future by artificially setting attention weights to 0!

Fixing the first self-attention problem: **Sequence order**

- Since self-attention doesn't build in order information, we need to encode the order of the sentence in our keys, queries, and values.
- Consider representing each **sequence index** as a **vector**

$p_i \in \mathbb{R}^d$, for $i \in \{1,2,\dots,T\}$ are position vectors

- Don't worry about what the p_i are made of yet!
- Easy to incorporate this info into our self-attention block: just add the p_i to our inputs!
- Let v_i', k_i', q_i' be our old values, keys, and queries.

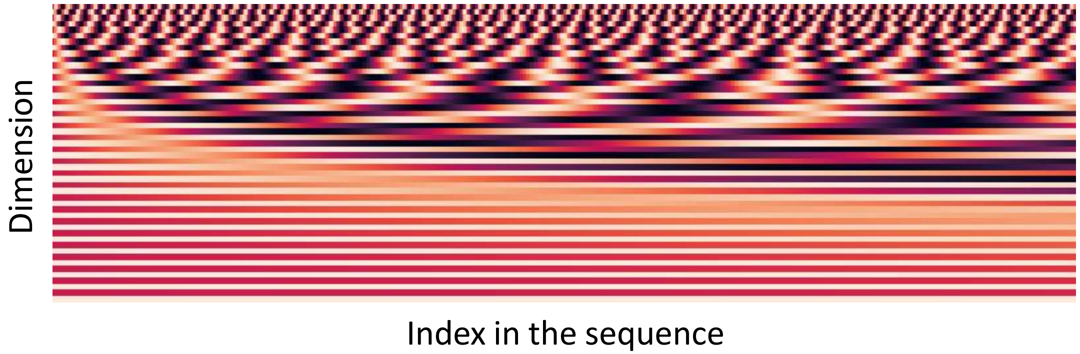
$$
\begin{aligned}
v_i &= v_i' + p_i \\
q_i &= q_i' + p_i \\
k_i &= k_i' + p_i
\end{aligned}
$$

In deep self-attention networks, we do this at the first layer! You could concatenate them as well, but people mostly just add…

Position representation vectors through sinusoids

- **Sinusoidal position representations:** concatenate sinusoidal functions of varying periods:

$$p_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$



- Pros:
  - Periodicity indicates that maybe “absolute position” isn’t as important
  - Maybe can extrapolate to longer sequences as periods restart!
- Cons:
  - Not learnable; also the extrapolation doesn’t really work!

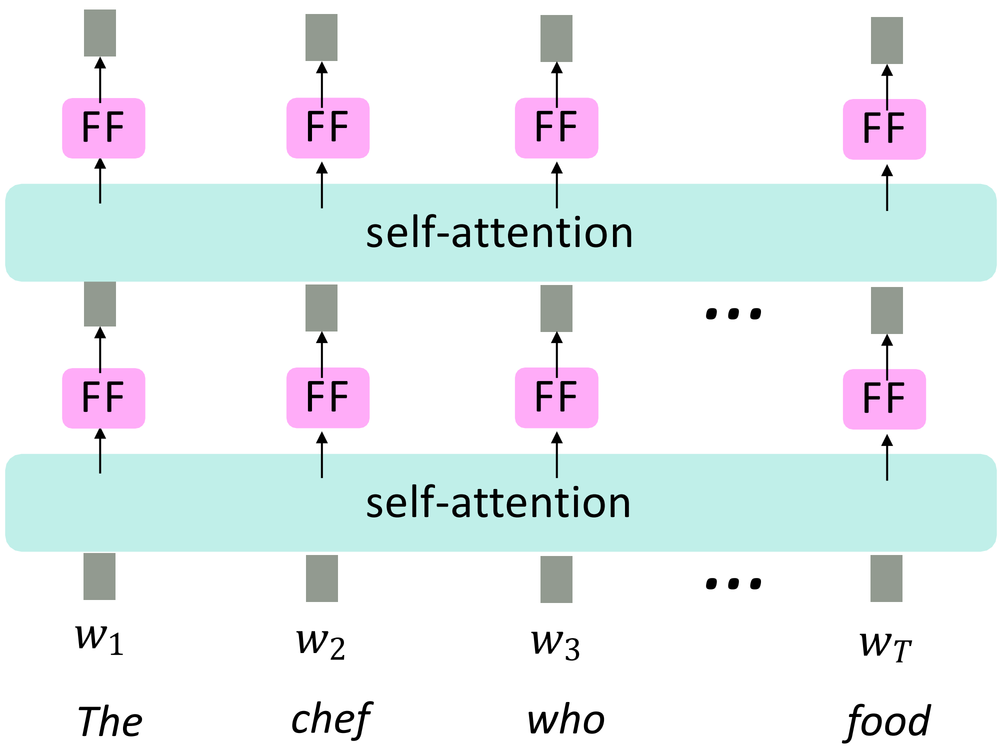
Image: https://timodenk.com/blog/linear-relationships-in-the-transformers-positional-encoding/

Adding nonlinearities in self-attention

- Note that there are no elementwise nonlinearities in self-attention; stacking more self-attention layers just re-averages **value** vectors

- Easy fix: add a **feed-forward network** to post-process each output vector.

$$
\begin{aligned}
m_i &= MLP(\text{output}_i) \\
&= W_2 * \text{ReLU}(W_1 \times \text{output}_i + b_1) + b_2
\end{aligned}
$$

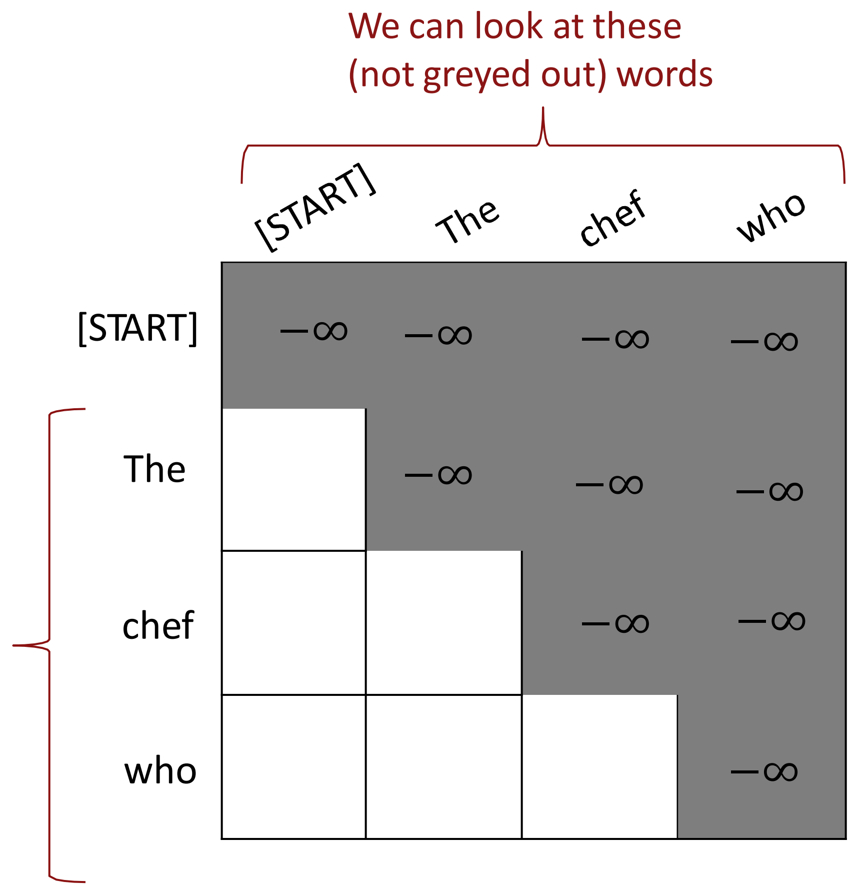


Intuition: the FF network processes the result of attention

Masking the future in self-attention

- To use self-attention in **decoders**, we need to ensure we can't peek at the future.
- At every timestep, we could change the set of **keys and queries** to include only past words. (Inefficient!)
- To enable parallelization, we **mask out attention** to future words by setting attention scores to $-\infty$.

$$e_{ij} = \begin{cases} q_i^{\top} k_j, j < i \\ -\infty, j \geq i \end{cases}$$



For encoding these words

Necessities for a self-attention building block:

- **Self-attention**:
  - The basis of the method.
- **Position representations**:
  - Specify the sequence order, since self-attention is an unordered function of its inputs.
- **Nonlinearities**:
  - At the output of the self-attention block
  - Frequently implemented as a simple feed-forward network.
- **Masking**:
  - In order to parallelize operations while not looking at the future.
  - Keeps information about the future from "leaking" to the past.

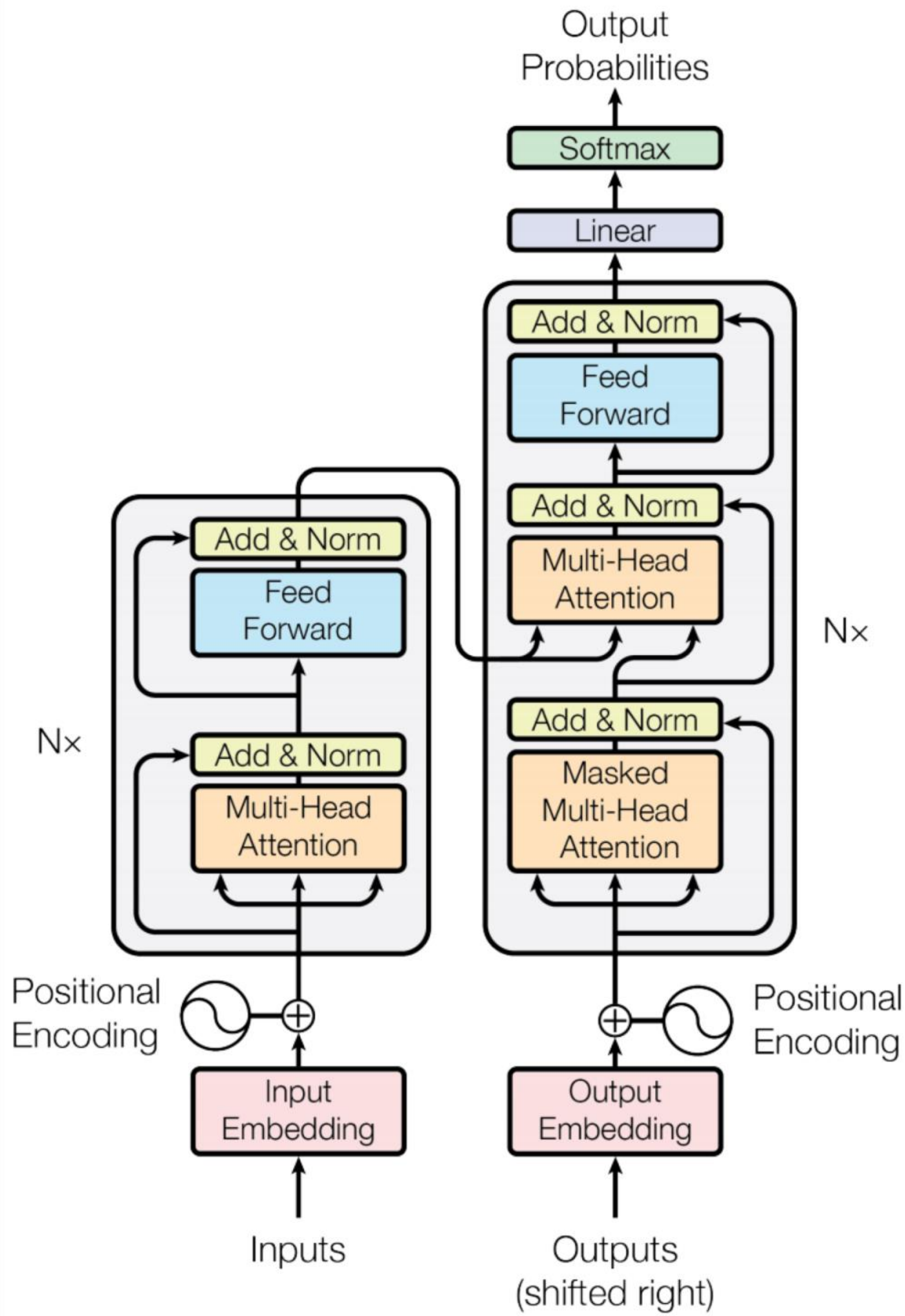
- That's it! But this is not the **Transformer** model we've been hearing about.

Transformer Overview

Attention is all you need. 2017. Aswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin
https://arxiv.org/pdf/1706.03762.pdf

- Non-recurrent sequence-to-sequence encoder-decoder model
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier

This and related figures from paper ⇑



The Transformer Encoder-Decoder [Vaswani et al., 2017]

Next, let's look at the Transformer Encoder and Decoder Blocks

What's left in a Transformer Encoder Block that we haven't covered?

1. **Key-query-value attention:** How do we get the k, q, v vectors from a single word embedding?
2. **Multi-headed attention**: Attend to multiple places in a single layer!
3. **Tricks to help with training!**
    1. Residual connections
    2. Layer normalization
    3. Scaling the dot product
    4. These tricks **don't improve** what the model is able to do; they help improve the training process

The Transformer Encoder:
Key-Query-Value Attention

- We saw that self-attention is when keys, queries, and values come from the same source. The Transformer does this in a particular way:
  - Let $x_1, \ldots, x_T$ be input vectors to the Transformer encoder; $x_i \in \mathbb{R}^d$

- Then keys, queries, values are:
  - $k_i = Kx_i$, where $K \in \mathbb{R}^{d \times d}$ is the key matrix.
  - $q_i = Qx_i$, where $\mathrm{Q} \in \mathbb{R}^{d \times d}$ is the query matrix.
  - $v_i = Vx_i$, where $\mathrm{V} \in \mathbb{R}^{d \times d}$ is the value matrix.

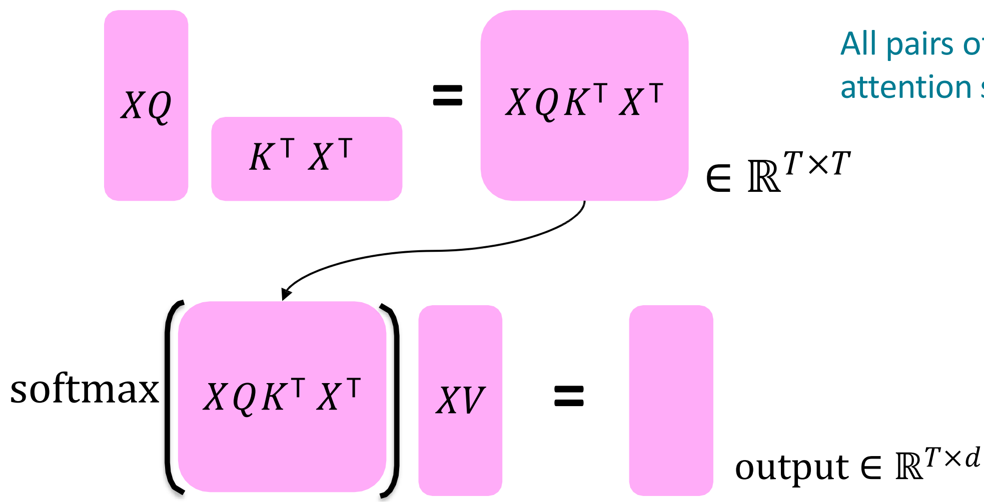
- These matrices allow *different aspects* of the x vectors to be used/emphasized in each of the three roles.

The Transformer Encoder: **Key-Query-Value Attention**

- Let's look at how key-query-value attention is computed, in matrices.
  - Let $X = [x_1; \dots; x_T] \in \mathbb{R}^{T\times d}$ be the concatenation of input vectors.
  - First, note that $XK \in \mathbb{R}^{T\times d}$, $XQ \in \mathbb{R}^{T\times d}$, $XV \in \mathbb{R}^{T\times d}$.
  - The output is defined as output $= \text{softmax}(XQ(XK)^\top) \times XV$.

First, take the query-key dot products in one matrix multiplication: $XQ(XK)$

Next, softmax, and compute the weighted average with another matrix multiplication.



All pairs of attention scores!

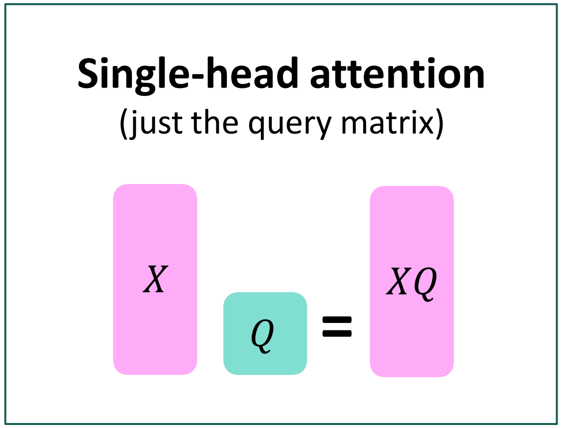
The Transformer Encoder:
Multi-headed attention

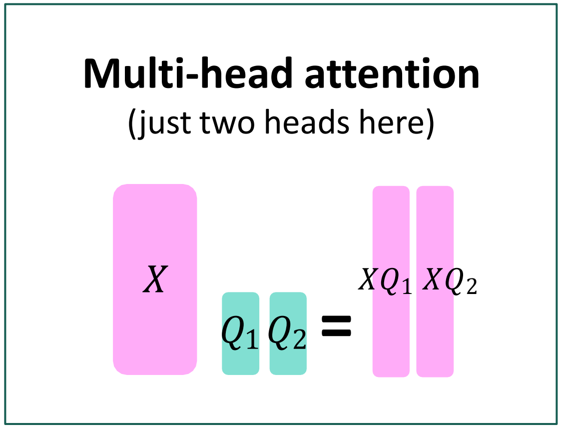
- What if we want to look in multiple places in the sentence at once?
  - For word i, self-attention "looks" where $x^\top Q^\top K x_j$ is high, but maybe we want to focus on different j for different reasons?
- We'll define **multiple attention "heads"** through multiple Q,K,V matrices
- Let, $Q_P, K_P, V_P \in \mathbb{R}^{d \times \frac{d}{h}}$, where h is the number of attention heads, and P ranges from 1 to h.
- Each attention head performs attention independently:
  - $\text{output}_P = \text{softmax}\left(XQ_P K_P^\top X^\top\right) * XV_P$, where $\text{output}_P \in \mathbb{R}^{d/h}$
- Then the outputs of all the heads are combined!
  - $\text{output} = Y[\text{output}_1; \ldots; \text{output}_h]$, where $Y \in \mathbb{R}^{d \times d}$

- Each head gets to "look" at different things, and construct value vectors differently.

The Transformer Encoder: **Multi-headed attention**

- What if we want to look in multiple places in the sentence at once?
  - For word i, self-attention "looks" where $x^\top Q^\top K x_j$ is high, but maybe we want to focus on different j for different reasons?
- We'll define **multiple attention "heads"** through multiple Q,K,V matrices
- Let, $Q_P, K_P, V_P \in \mathbb{R}^{d \times \frac{d}{h}}$, where h is the number of attention heads, and P ranges from 1 to h.

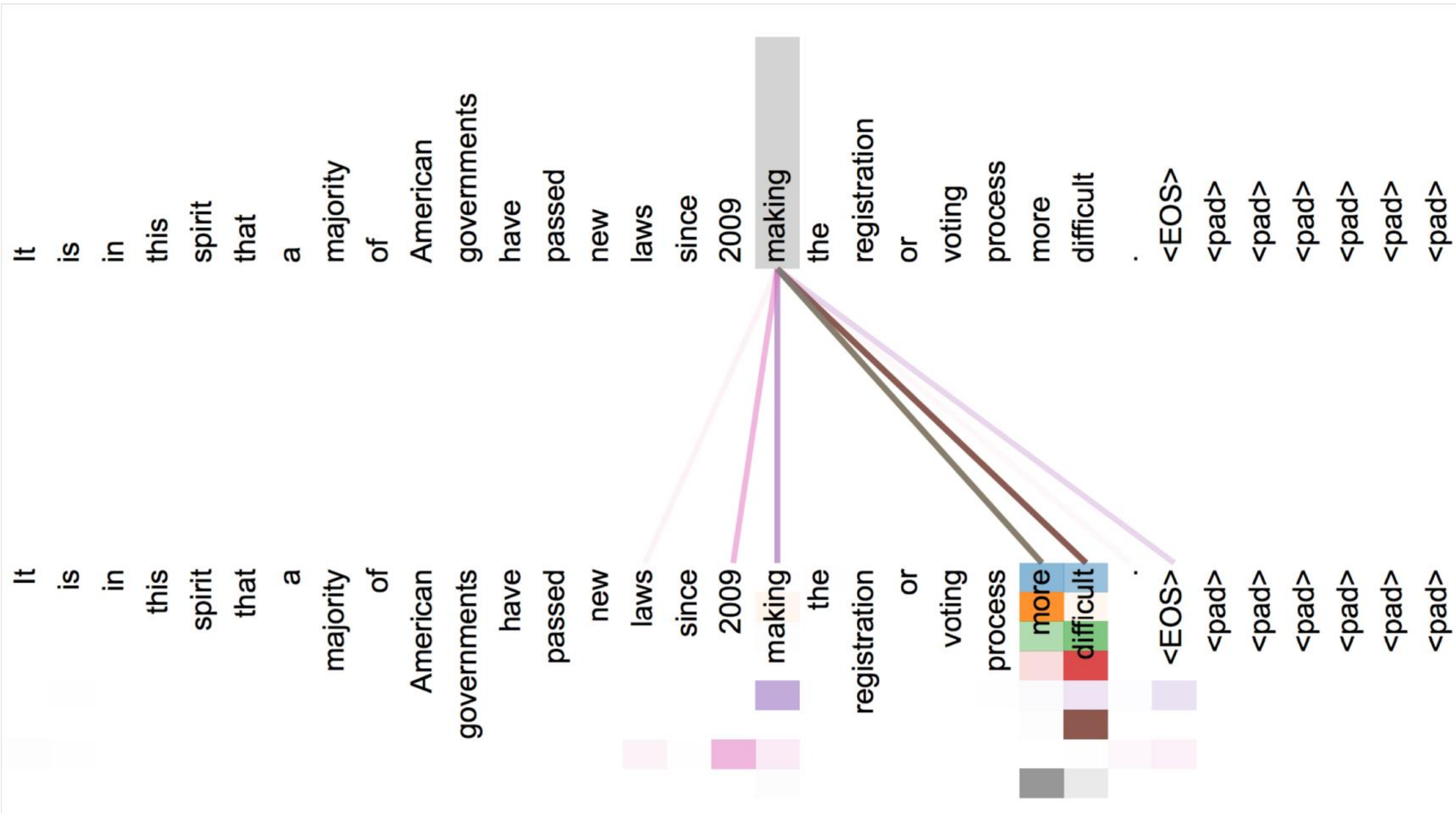




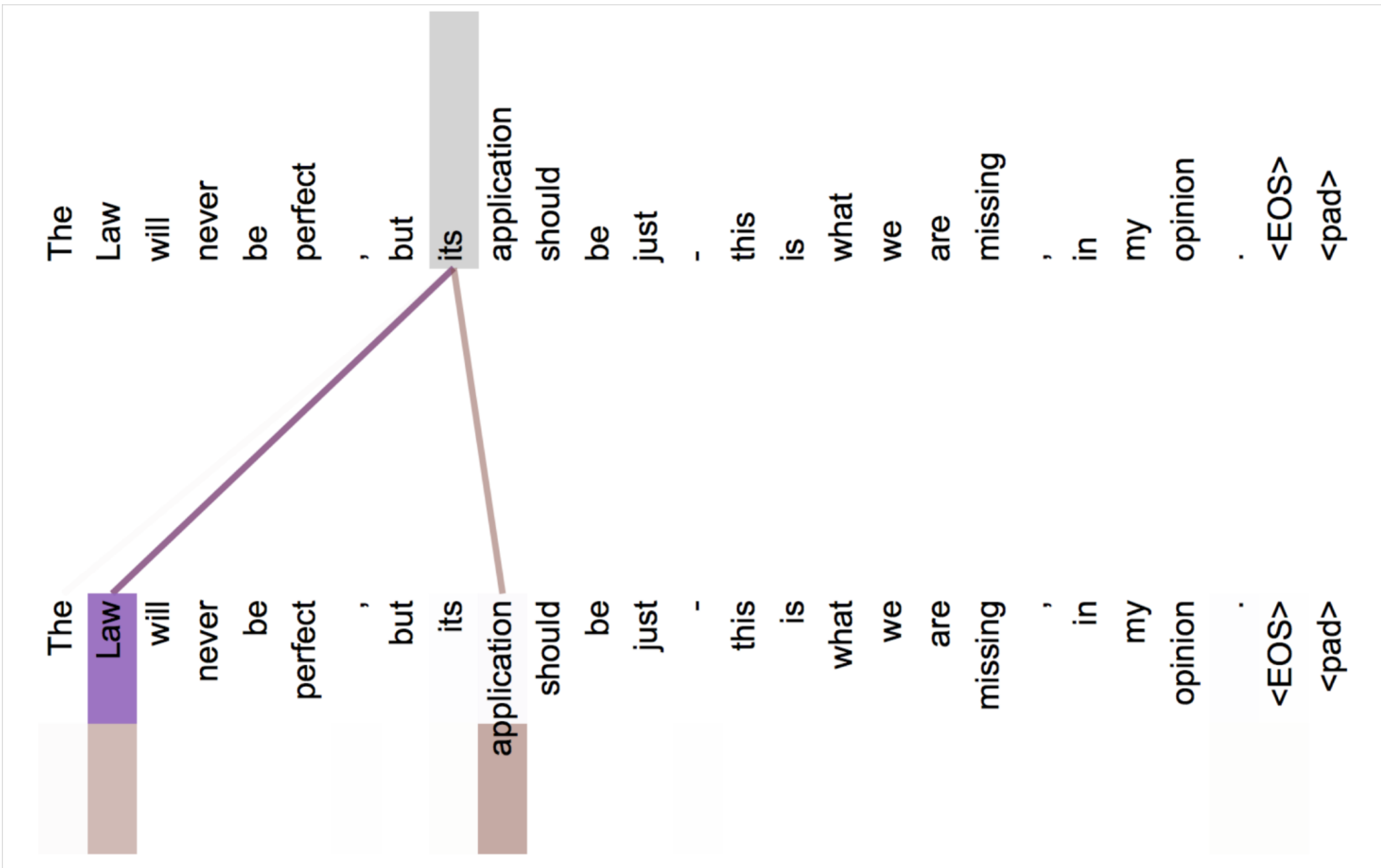
Same amount of computation as single-head self-attention!

Attention visualization in layer 5

- Words start to pay attention to other words in sensible ways

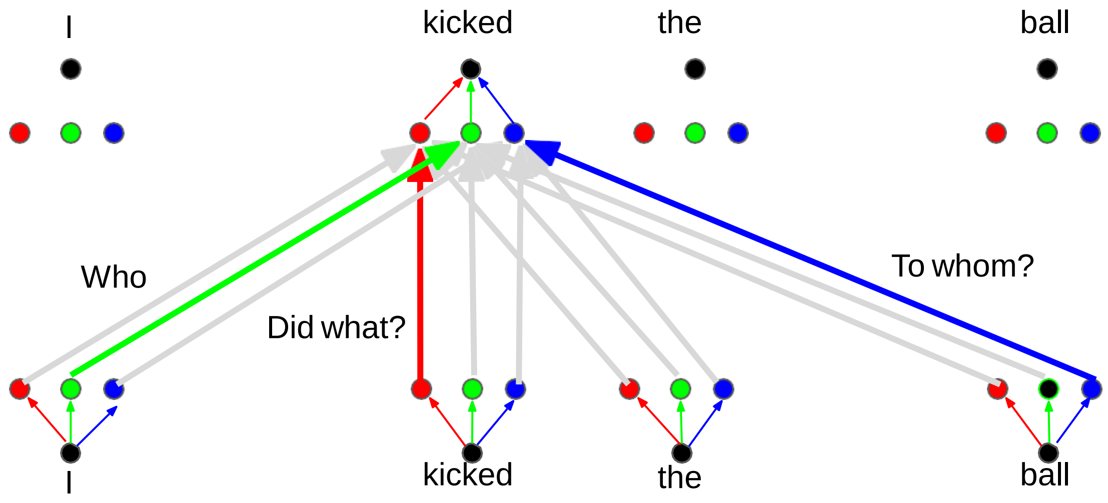


Attention visualization: Implicit anaphora resolution



In 5th layer. Isolated attentions from just the word ‘its’ for attention heads 5 and 6. Note that the attentions are very sharp for this word.

Parallel attention heads



Ashish Vaswani

Quadratic computation as function of seq. length

- One of the benefits of self-attention over recurrence was that it's highly parallelizable.
- However, its total number of operations grows as $O\,(T^2 d)$, where T is the sequence length, and d is the dimensionality.

$$XQ \quad K^\top X^\top = XQK^\top X^\top \in \mathbb{R}^{T\times T}$$

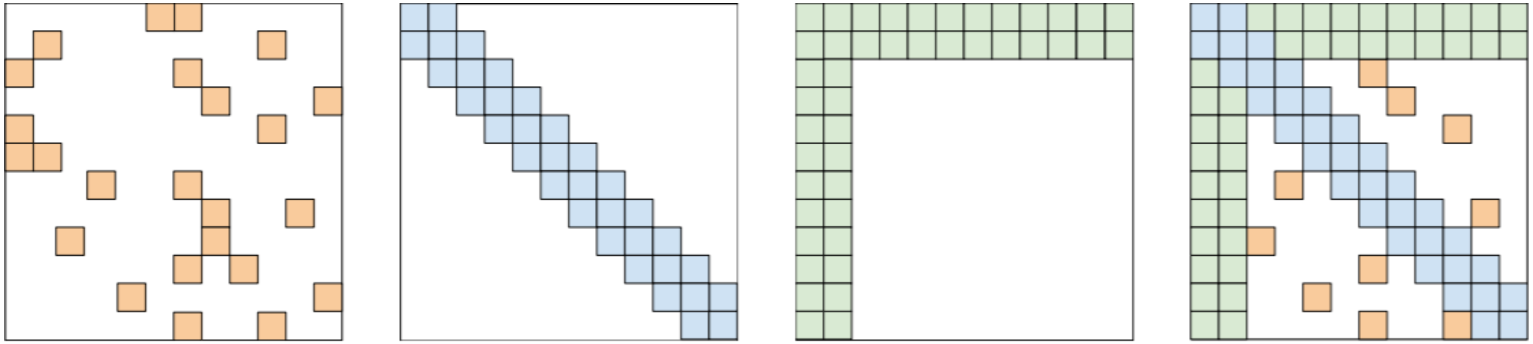
Need to compute all pairs of interactions!
$O(T^2 d)$

- Think of d as around $\mathbf{1,000}$.
  - So, for a single (shortish) sentence, $T \leq 30; T^2 \leq \mathbf{900.}$
  - In practice, we set a bound like $T = 512$.
  - **But what if we'd like $\boldsymbol{T \geq 10,000}$?** For example, to work on long documents?

Recent work on improving on quadratic self-attention cost

- Considerable recent work has gone into the question, *Can we build models like Transformers without paying the $O(T^2)$ all-pairs self-attention cost?*
- For example, **BigBird** [Zaheer et al., 2021]

Key idea: replace all-pairs interactions with a family of other interactions, **like local windows**, **looking at everything**, and **random interactions**.

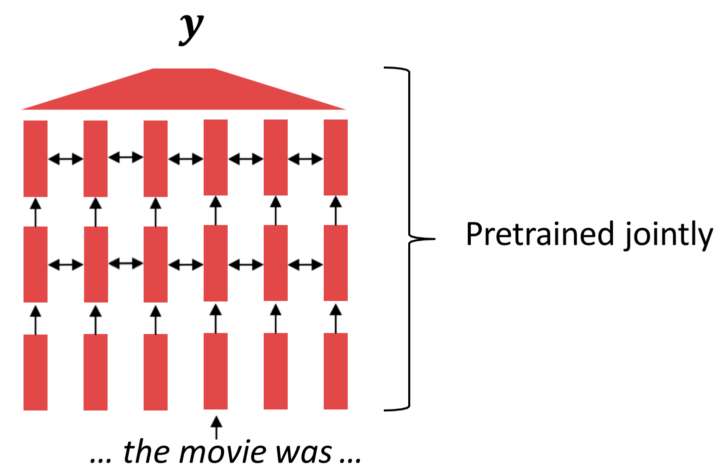


(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Pretraining models

In modern NLP:

- All (or almost all) parameters in NLP networks are initialized via **pretraining**.
- Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts.
- This has been exceptionally effective at building strong:
  - **representations of language**
  - **parameter initializations** for strong NLP models.

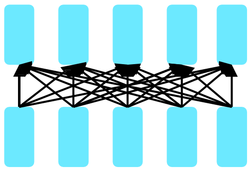


[This model has learned how to represent entire sentences through pretraining]

Adapted from John Hewitt

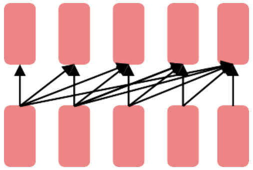
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



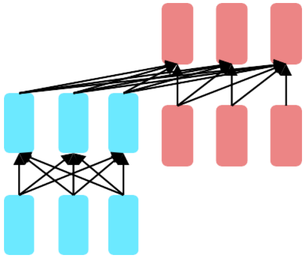
Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

Adapted from John Hewitt

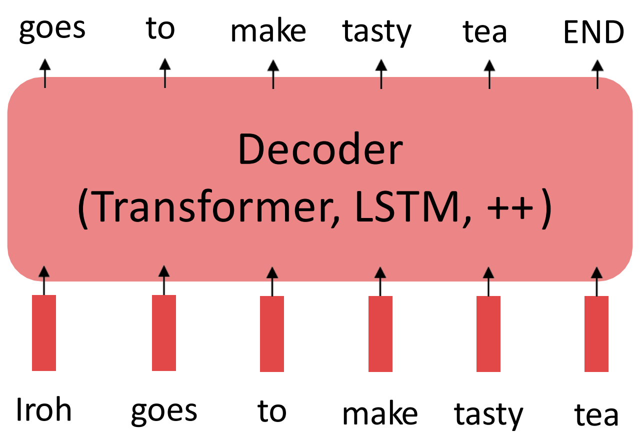
Pretraining through language modeling [Dai and Le, 2015]

Recall the **language modeling** task:

- Model $p_\theta(w_t | w_{1:t-1})$, the probability distribution over words given their past contexts.
- There's lots of data for this! (In English.)

Pretraining through language modeling:

- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.

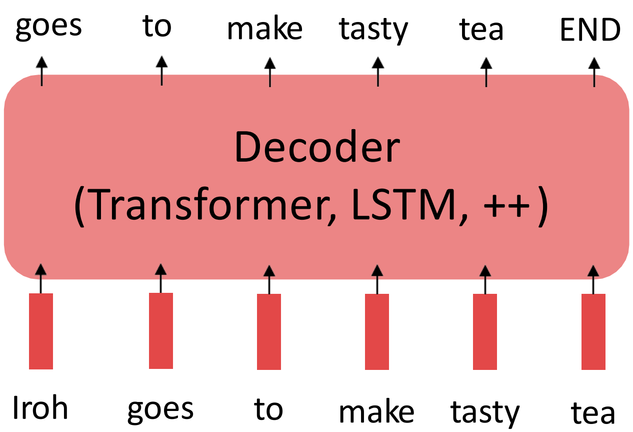


The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

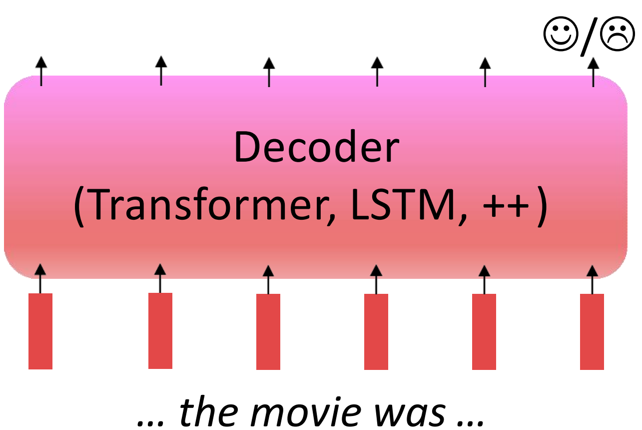
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!



Capturing meaning via context: What kinds of things does pretraining learn?

There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language:

- *Stanford University is located in__________, California.* [Trivia]
- *I put___fork down on the table.* [syntax]
- *The woman walked across the street, checking for traffic over___shoulder.* [coreference]
- *I went to the ocean to see the fish, turtles, seals, and_____.* [lexical semantics/topic]
- *Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was___.* [sentiment]
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the______. [some reasoning – this is harder]
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21,____[some basic arithmetic; they don't learn the Fibonnaci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.

Adapted from John Hewitt

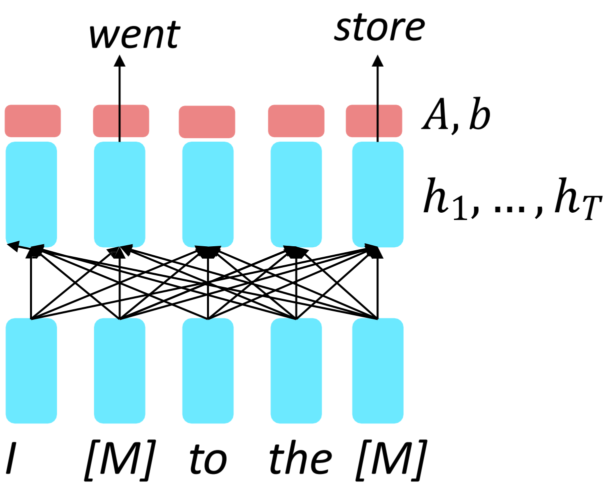
Pretraining encoders:
What pretraining objective to use?

So far, we've looked at language model pretraining. But **encoders get bidirectional context,** so we can't do language modeling!

Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

$$h_1, \dots, h_T = \text{Encoder}(w_1, \dots, w_T)$$
$$y_i \sim Aw_i + b$$

Only add loss terms from words that are "masked out." If x' is the masked version of x, we're learning $p_\theta(x|x')$. Called **Masked LM**.



[Devlin et al., 2018]

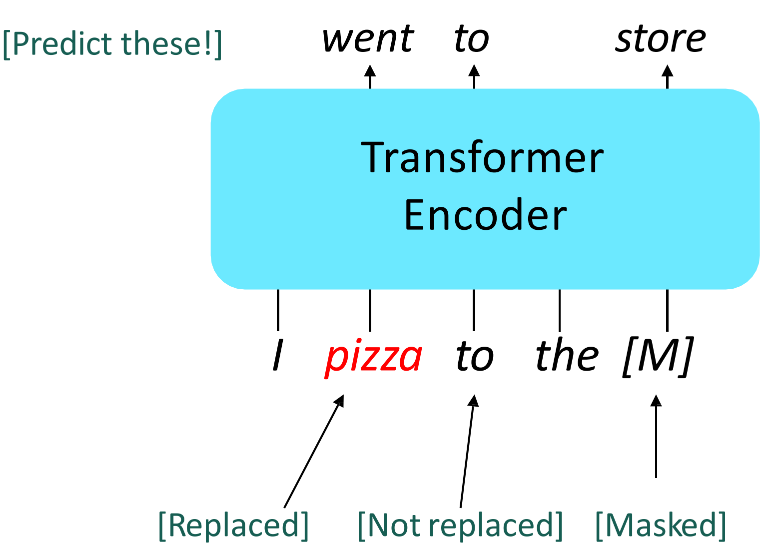
BERT: Bidirectional Encoder Representations from Tranformers

Devlin et al., 2018 proposed the "Masked LM" objective and **released the weights of a pretrained Transformer**, a model they labeled BERT.

Some more details about Masked LM for BERT:

- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)

- Too little masking: Too expensive to train
- Too much masking: Not enough context



[Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Tranformers

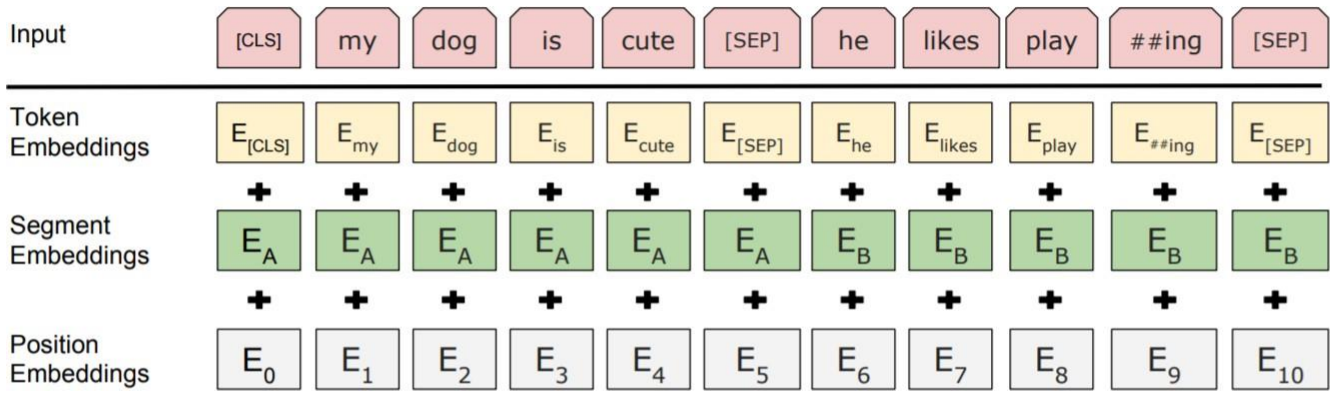
- Additional task: Next sentence prediction
- To learn *relationships* between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

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

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

BERT: Bidirectional Encoder Representations from Tranformers

- The pretraining input to BERT was two separate contiguous chunks of text:



- In addition to masked input reconstruction, BERT was trained to predict whether one chunk follows the other or is randomly sampled.
- Later work has argued this “next sentence prediction” is not necessary.

[Devlin et al., 2018, Liu et al., 2019]

Adapted from John Hewitt

BERT: Bidirectional Encoder Representations from Tranformers

Details about BERT

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
  - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
  - “Pretrain once, finetune many times.”

[Devlin et al., 2018]

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

2018's GPT was a big success in pretraining a decoder!

- Transformer decoder with 12 layers.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
- Trained on BooksCorpus: over 7000 unique books.
  - Contains long spans of contiguous text, for learning long-distance dependencies.
- The acronym "GPT" never showed up in the original paper; it could stand for "Generative PreTraining" or "Generative Pretrained Transformer"

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

How do we format inputs to our decoder for **finetuning tasks?**

Natural Language Inference: Label pairs of sentences as *entailing/contradictory/neutral*

Premise: *The man is in the doorway*
Hypothesis: *The person is near the door*

entailment

Radford et al., 2018 evaluate on natural language inference.
Here's roughly how the input was formatted, as a sequence of tokens for the decoder.

[START] *The man is in the doorway* [DELIM] *The person is near the door* [EXTRACT]

The linear classifier is applied to the representation of the [EXTRACT] token.

Increasingly convincing generations (GPT2) [Radford et al., 2018]

We mentioned how pretrained decoders can be used **in their capacities as language models.** **GPT-2,** a larger version of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Aside: Word structure and subword models

Let’s take a look at the assumptions we’ve made about a language’s vocabulary.

We assume a fixed vocab of tens of thousands of words, built from the training set.
All *novel* words seen at test time are mapped to a single UNK.

	word	vocab mapping	embedding
Common words	hat	pizza (index)	
	learn	tasty (index)	
Variations	taaaaasty	UNK (index)	
misspellings	laern	UNK (index)	
novel items	Transformerify	UNK (index)	

Aside: Word structure and subword models

Finite vocabulary assumptions make even *less* sense in many languages.

- Many languages exhibit complex **morphology**, or word structure.
  - The effect is more word types, each occurring fewer times.

Example: Swahili verbs can have hundreds of conjugations, each encoding a wide variety of information. (Tense, mood, definiteness, negation, information about the object, ++)

Here's a small fraction of the conjugations for *ambia* – to tell.

Conjugation of *-ambia* [less ▲]

Non-finite forms		
Form	Positive	Negative
Infinitive	kuambia	kutoambia
Simple finite forms		
Positive form	Singular	Plural
Imperative	ambia	ambieni
Habitual	huambia	

Complex finite forms																		
Polarity	Persons				Persons / Classes		Classes											
	1st		2nd		3rd / M-wa		M-mi		Ma		Ki-vi		N		U	Ku	Pa	Mu
	Sg.	Pl.	Sg.	Pl.	Sg. / 1	Pl. / 2	3	4	5	6	7	8	9	10	11 / 14	15 / 17	16	18
Past [less ▲]																		
Positive	niliambia naliambia	tuliambia twaliambia	uliambia waliambia	mliambia mwaliambia	aliambia	waliambia	uliambia	iliambia	liliambia	yaliambia	kiliambia	viliambia	iliambia	ziliambia	uliambia	kuliambia	paliambia	muliambia
Negative	sikuambia	hatukuambia	hukuambia	hamkuambia	hakuambia	hawakuambia	haukuambia	haikuambia	halikuambia	hayakuambia	hakikuambia	havikuambia	haikuambia	hazikuambia	haukuambia	hakukuambia	hapakuambia	hamukuambia
Present [less ▲]																		
Positive	ninaambia naambia	tunaambia	unaambia	mnaambia	anaambia	wanaambia	unaambia	inaambia	linaambia	yanaambia	kinaambia	vinaambia	inaambia	zinaambia	unaambia	kunaambia	panaambia	munaambia
Negative	siambii	hatuambii	huambii	hamambii	haambii	hawaambii	hauambii	haiambii	haliambii	hayaambii	hakiambii	haviambii	haiambii	haziambii	hauambii	hakuambii	hapaambii	hamuambii
Future [less ▲]																		
Positive	nitaambia	tutaambia	utaambia	mtaambia	ataambia	wataambia	utaambia	itaambia	litaambia	yataambia	kitaambia	vitaambia	itaambia	zitaambia	utaambia	kutaambia	pataambia	mutaambia
Negative	sitaambia	hatutaambia	hutaambia	hamtaambia	hataambia	hawataambia	hautaambia	haitaambia	halitaambia	hayataambia	hakitaambia	havitaambia	haitaambia	hazitaambia	hautaambia	hakutaambia	hapataambia	hamutaambia
Subjunctive [less ▲]																		
Positive	niambie	tuambie	uambie	mambie	aambie	waambie	uambie	iambie	liambie	yaambie	kiambie	viambie	iambie	ziambie	uambie	kuambie	paambie	muambie
Negative	nisiambie	tusiambie	usiambie	msiambie	asiambie	wasiambie	usiambie	isiambie	lisiambie	yasiambie	kisiambie	visiambie	isiambie	zisiambie	usiambie	kusiambie	pasiambie	musiambie
Present Conditional [less ▲]																		
Positive	ningeambia	tungeambia	ungeambia	mngeambia	angeambia	wangeambia	ungeambia	ingeambia	lingeambia	yangeambia	kingeambia	vingeambia	ingeambia	zingeambia	ungeambia	kungeambia	pangeambia	mungeambia
Negative	nisingeambia singeambia	tusingeambia hatungeambia	usingeambia hungeambia	msingeambia hamngeambia	asingeambia hangeambia	wasingeambia hawangeambia	usingeambia haungeambia	isingeambia haingeambia	lisingeambia halingeambia	yasingeambia hayangeambia	kisingeambia hakingeambia	visingeambia havingeambia	isingeambia haingeambia	zisingeambia hazingeambia	usingeambia haungeambia	kusingeambia hakungeambia	pasingeambia hapangeambia	musingeambia hamungeambia
Past Conditional [less ▲]																		
Positive	ningaliambia	tungaliambia	ungaliambia	mngaliambia	angaliambia	wangaliambia	ungaliambia	ingaliambia	lingaliambia	yangaliambia	kingaliambia	vingaliambia	ingaliambia	zingaliambia	ungaliambia	kungaliambia	pangaliambia	mungaliambia
Negative	nisingaliambia singaliambia	tusingaliambia hatungaliambia	usingaliambia hungaliambia	msingaliambia hamngaliambia	asingaliambia hangaliambia	wasingaliambia hawangaliambia	usingaliambia haungaliambia	isingaliambia haingaliambia	lisingaliambia halingaliambia	yasingaliambia hayangaliambia	kisingaliambia hakingaliambia	visingaliambia havingaliambia	isingaliambia haingaliambia	zisingaliambia hazingaliambia	usingaliambia haungaliambia	kusingaliambia hakungaliambia	pasingaliambia hapangaliambia	musingaliambia hamungaliambia
Conditional Contrary to Fact [less ▲]																		
Positive	ningeliambia	tungeliambia	ungeliambia	mngeliambia	angeliambia	wangeliambia	ungeliambia	ingeliambia	lingeliambia	yangeliambia	kingeliambia	vingeliambia	ingeliambia	zingeliambia	ungeliambia	kungeliambia	pangeliambia	mungeliambia
Gnomic [less ▲]																		
Positive	naambia	twaambia	waambia	mwaambia	aambia	waambia	waambia	yaambia	laambia	yaambia	chaambia	vyaambia	yaambia	zaambia	waambia	kwaambia	paambia	mwaambia
Perfect [less ▲]																		

[Wiktionary]

Aside: The byte-pair encoding algorithm

Subword modeling in NLP encompasses a wide range of methods for reasoning about structure below the word level. (Parts of words, characters, bytes.)

- The dominant modern paradigm is to learn a vocabulary of **parts of words (subword tokens).**
- At training and testing time, each word is split into a sequence of known subwords.

Byte-pair encoding is a simple, effective strategy for defining a subword vocabulary.

1. Start with a vocabulary containing only characters and an "end-of-word" symbol.
2. Using a corpus of text, find the most common adjacent characters "a,b"; add "ab" as a subword.
3. Replace instances of the character pair with the new subword; repeat until desired vocab size.

Originally used in NLP for machine translation; now a similar method (WordPiece) is used in pretrained models.

[Sennrich et al., 2016, Wu et al., 2016]

Aside: Word structure and subword models

Common words end up being a part of the subword vocabulary, while rarer words are split into (sometimes intuitive, sometimes not) components.

In the worst case, words are split into as many subwords as they have characters.

	word	vocab mapping	embedding
Common words	hat	hat	
	learn	learn	
Variations	taaaaasty	taa## aaa## sty	
misspellings	laern	la## ern##	
novel items	Transformerify	Transformer## ify	

GPT-3, in-context learning, very large models

So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- Fine-tune them on a task we care about, and take their predictions.

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters.

GPT-3 has 175 billion parameters.

GPT-3, in-context learning, very large models

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

The in-context examples seem to specify the task to be performed, and the conditional distribution mocks performing the task to a certain extent.

Input (prefix within a single Transformer decoder context):

“ thanks -> merci

hello -> bonjour

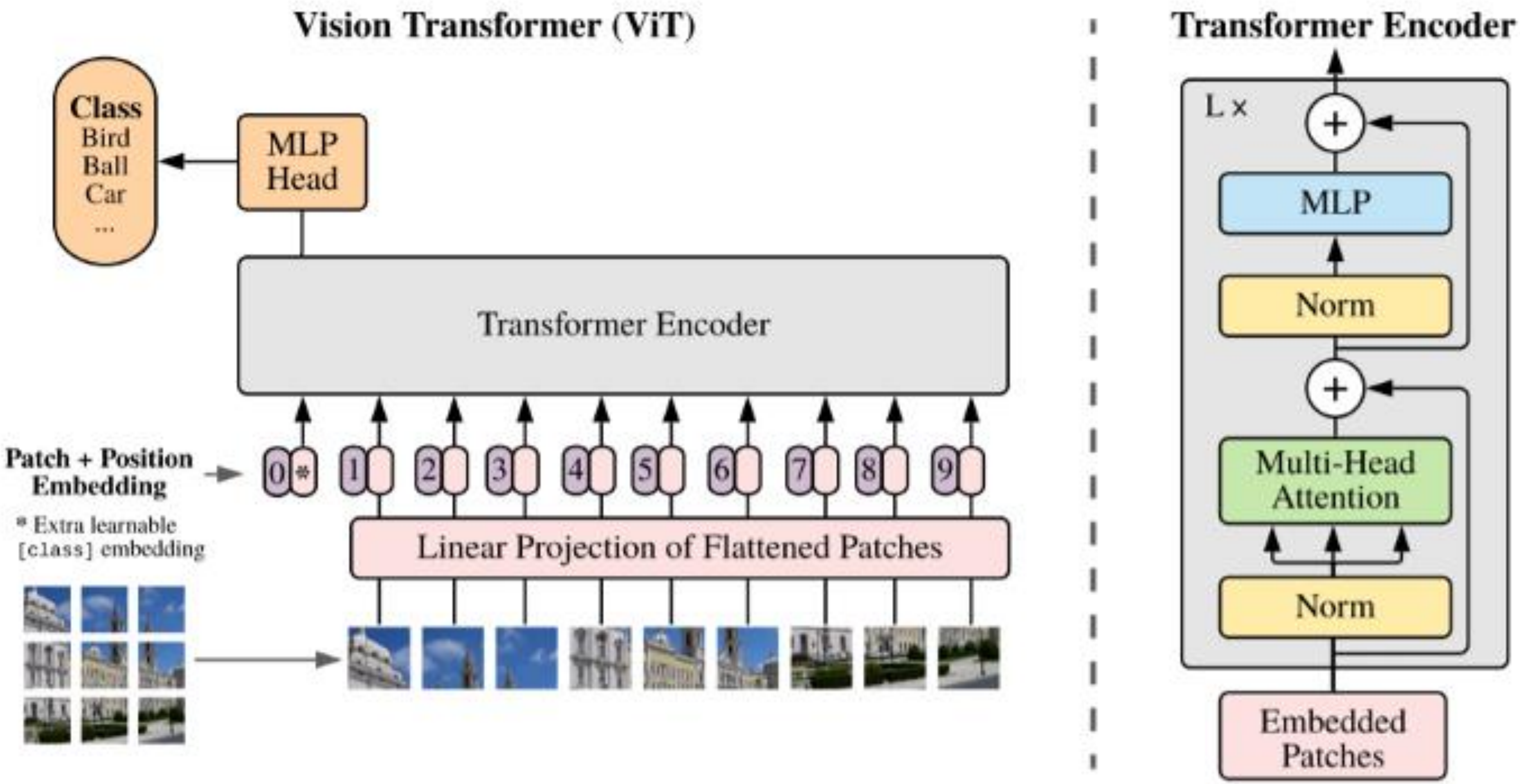
mint -> menthe

otter -> ”

Output (conditional generations):

loutre...”

Transformers in vision



Dosovitskiy, ICLR 2021, https://github.com/google-research/vision_transformer https://www.youtube.com/watch?v=TrdevFK_am4

Cross-modal transformers

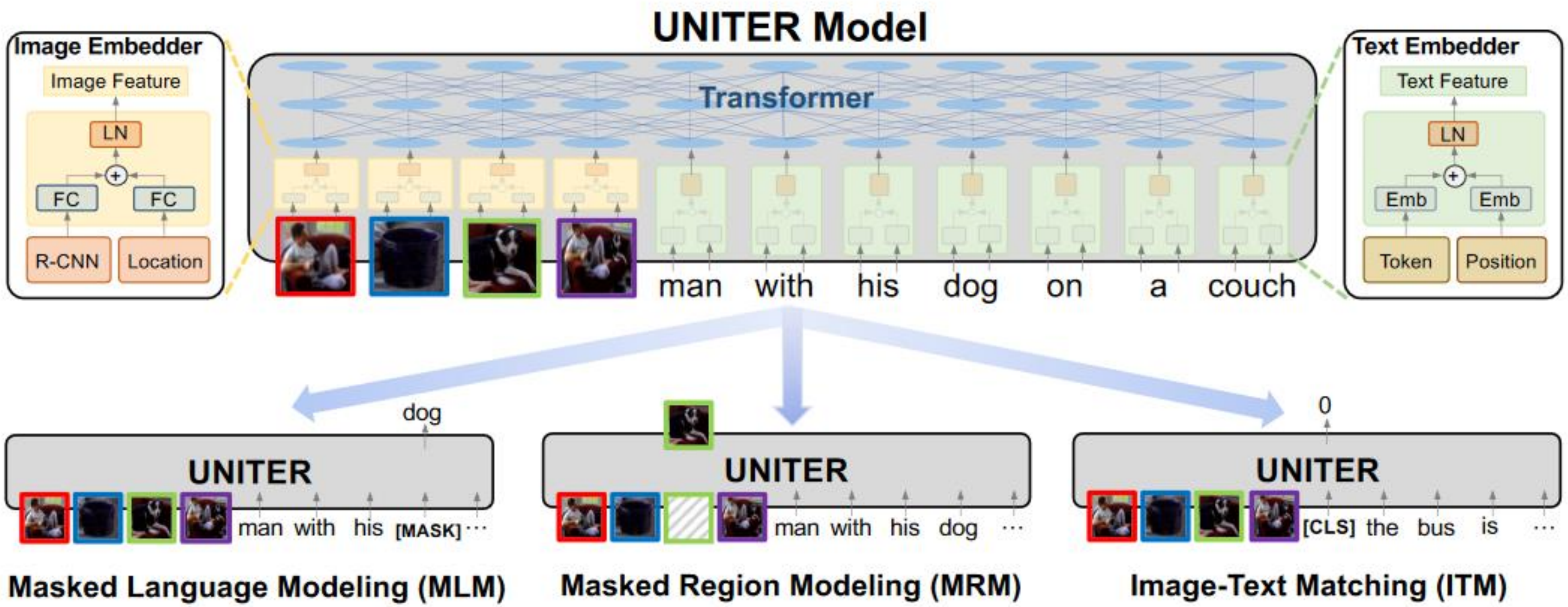


Figure 1: Overview of the proposed UNITER model (best viewed in color), consisting of an Image Embedder, a Text Embedder and a multi-layer self-attention Transformer, learned through three pre-training tasks.

Chen et al., "UNITER: Learning UNiversal Image-TExt Representations", ECCV 2020

Cross-modal transformers

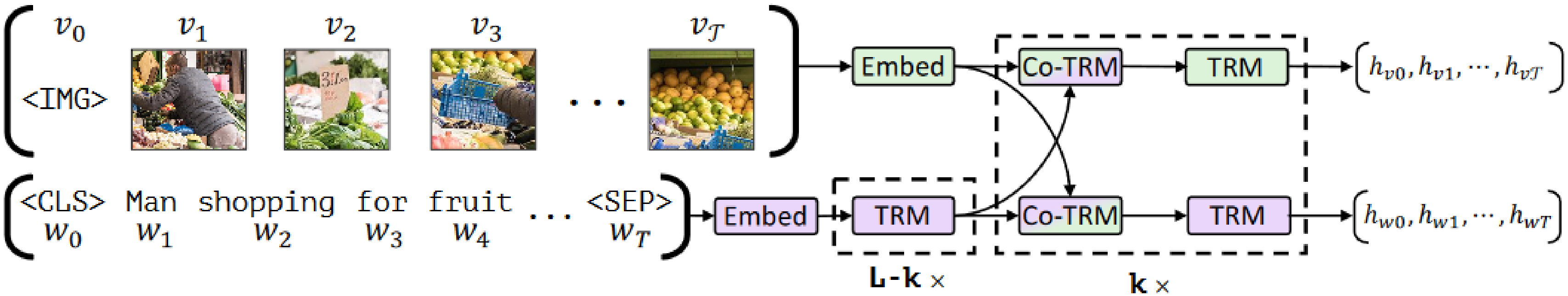


Figure 1: Our ViLBERT model consists of two parallel streams for visual (green) and linguistic (purple) processing that interact through novel co-attentional transformer layers. This structure allows for variable depths for each modality and enables sparse interaction through co-attention. Dashed boxes with multiplier subscripts denote repeated blocks of layers.

Lu et al., “ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks”, NeurIPS 2019

Cross-modal transformers

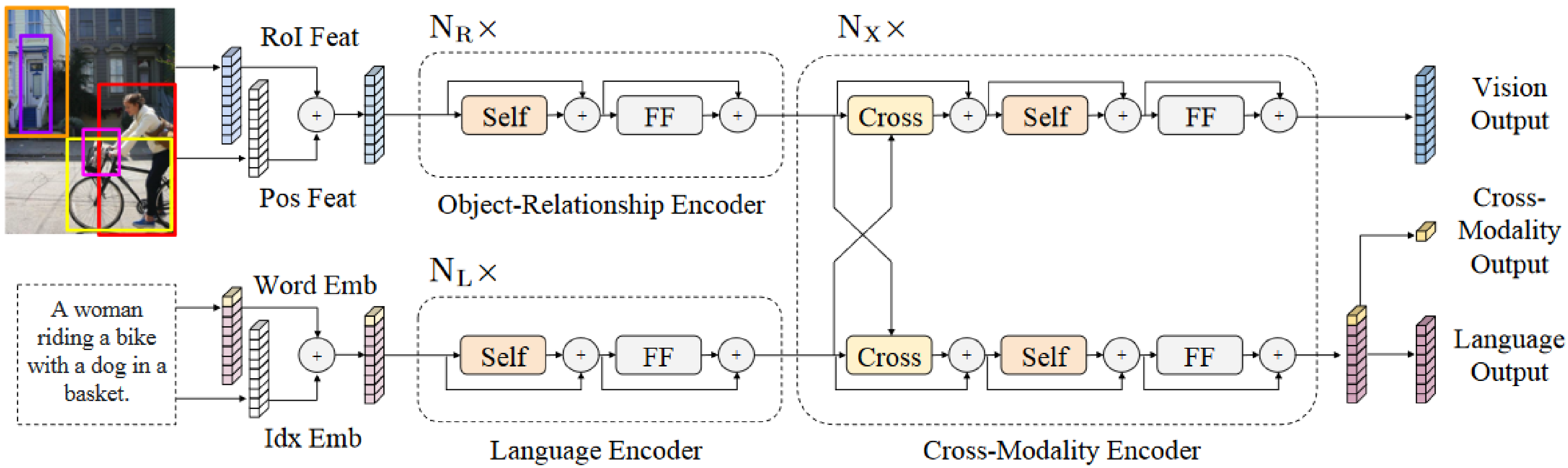


Figure 1: The LXMERT model for learning vision-and-language cross-modality representations. ‘Self’ and ‘Cross’ are abbreviations for self-attention sub-layers and cross-attention sub-layers, respectively. ‘FF’ denotes a feed-forward sub-layer.

Tan and Bansal, “LXMERT: Learning Cross-Modality Encoder Representationsfrom Transformers”, EMNLP 2019

Self-Supervised Learning

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

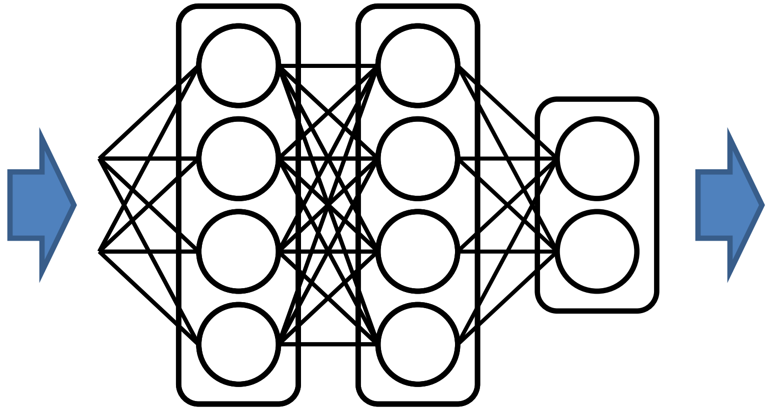
Jitendra Malik: "**Supervision** is the opium of the AI researcher"
Alyosha Efros: "The AI revolution will not be **supervised**"
Yann LeCun: “**Self-supervised** learning is the cake, **supervised** learning is the icing on the cake, **reinforcement learning** is the cherry on the cake"

Unsupervised Visual Representation Learning by Context Prediction

Carl Doersch, Alexei Efros and Abhinav Gupta

ICCV 2015

ImageNet + Deep Learning

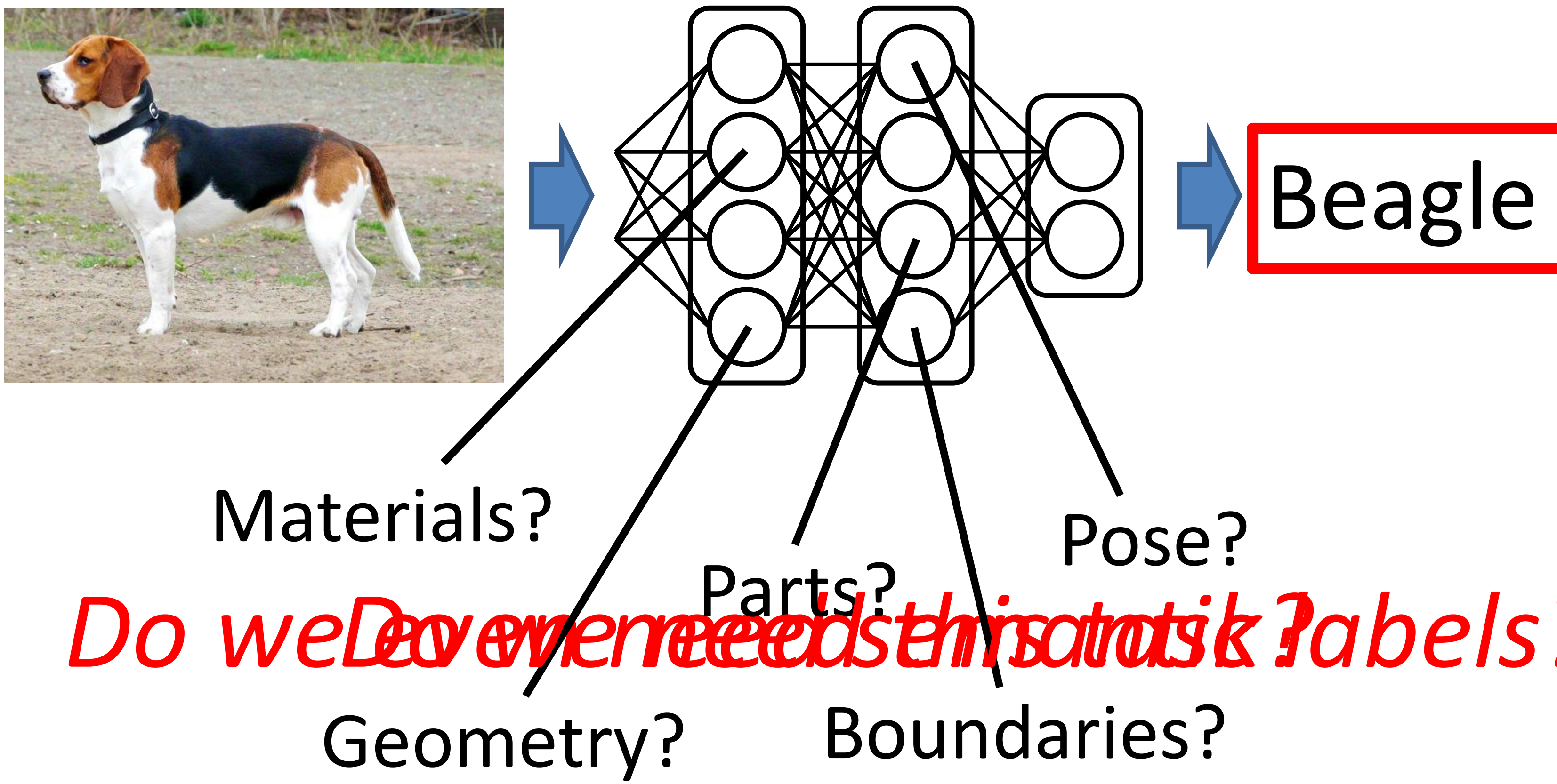


Beagle

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

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

ImageNet + Deep Learning



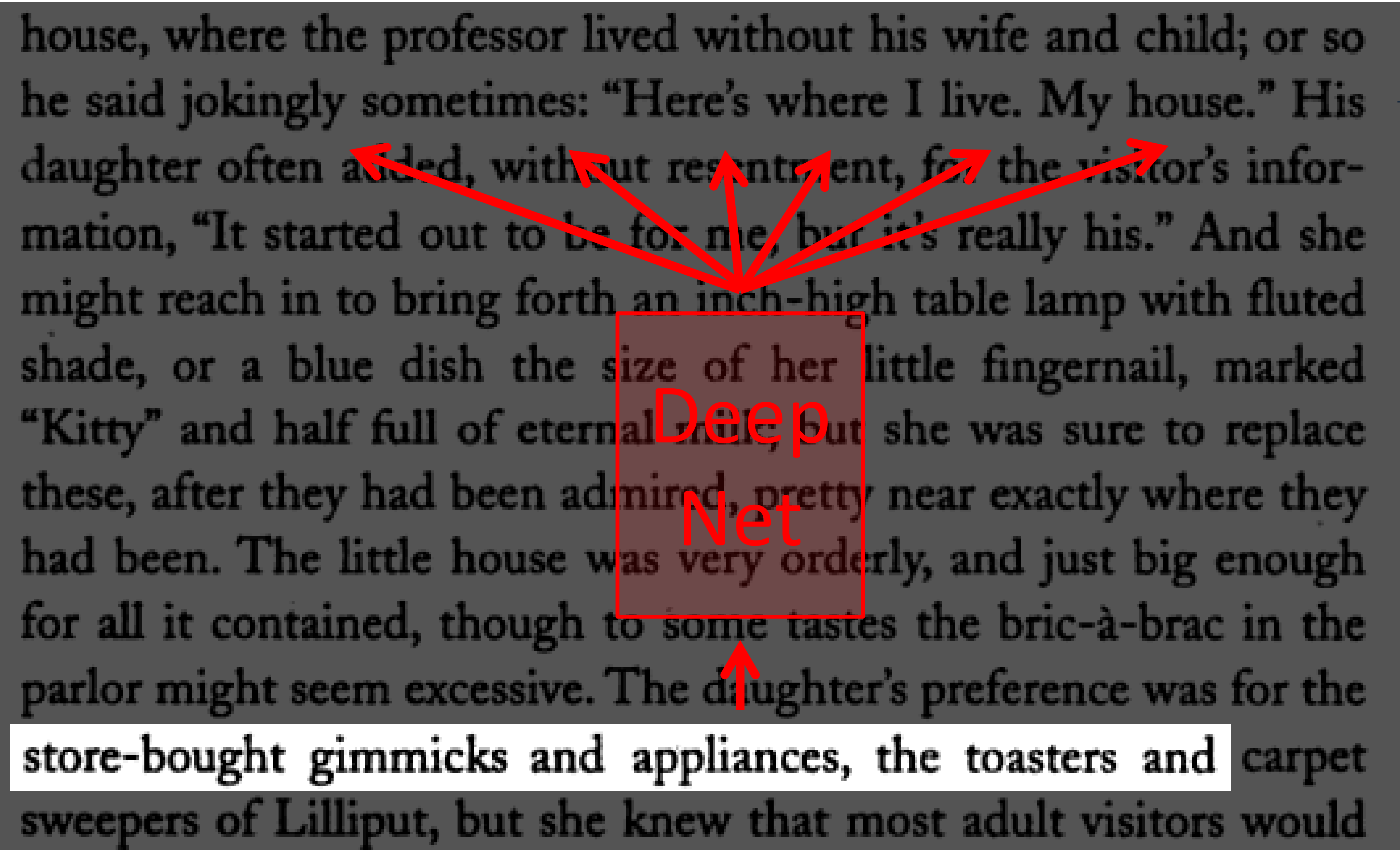
Do we need semantic labels?

Do we even need this task?

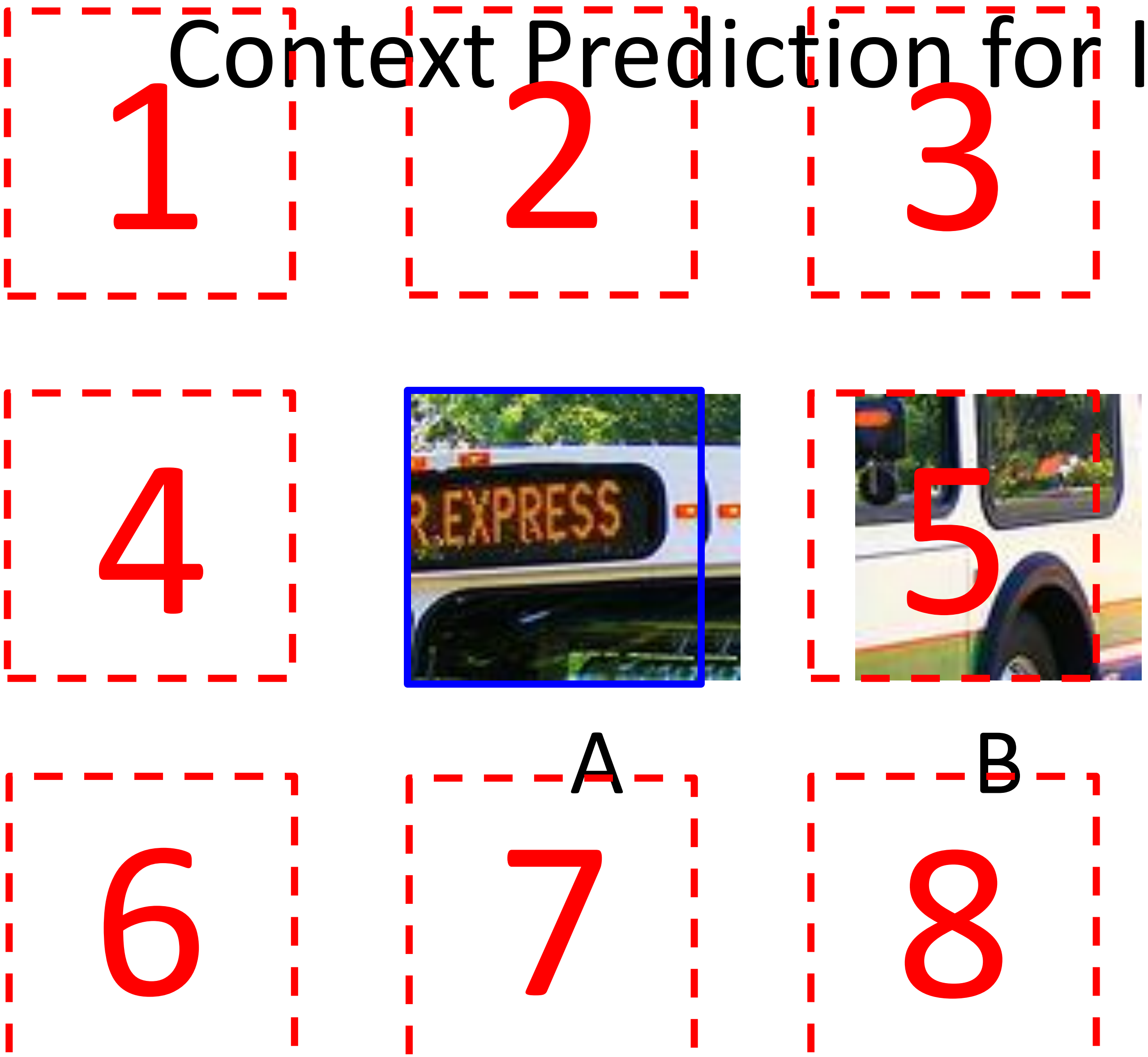
Doersch et al., "Unsupervised Visual Representation Learning by Context Prediction", ICCV 2015

Context as Supervision

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

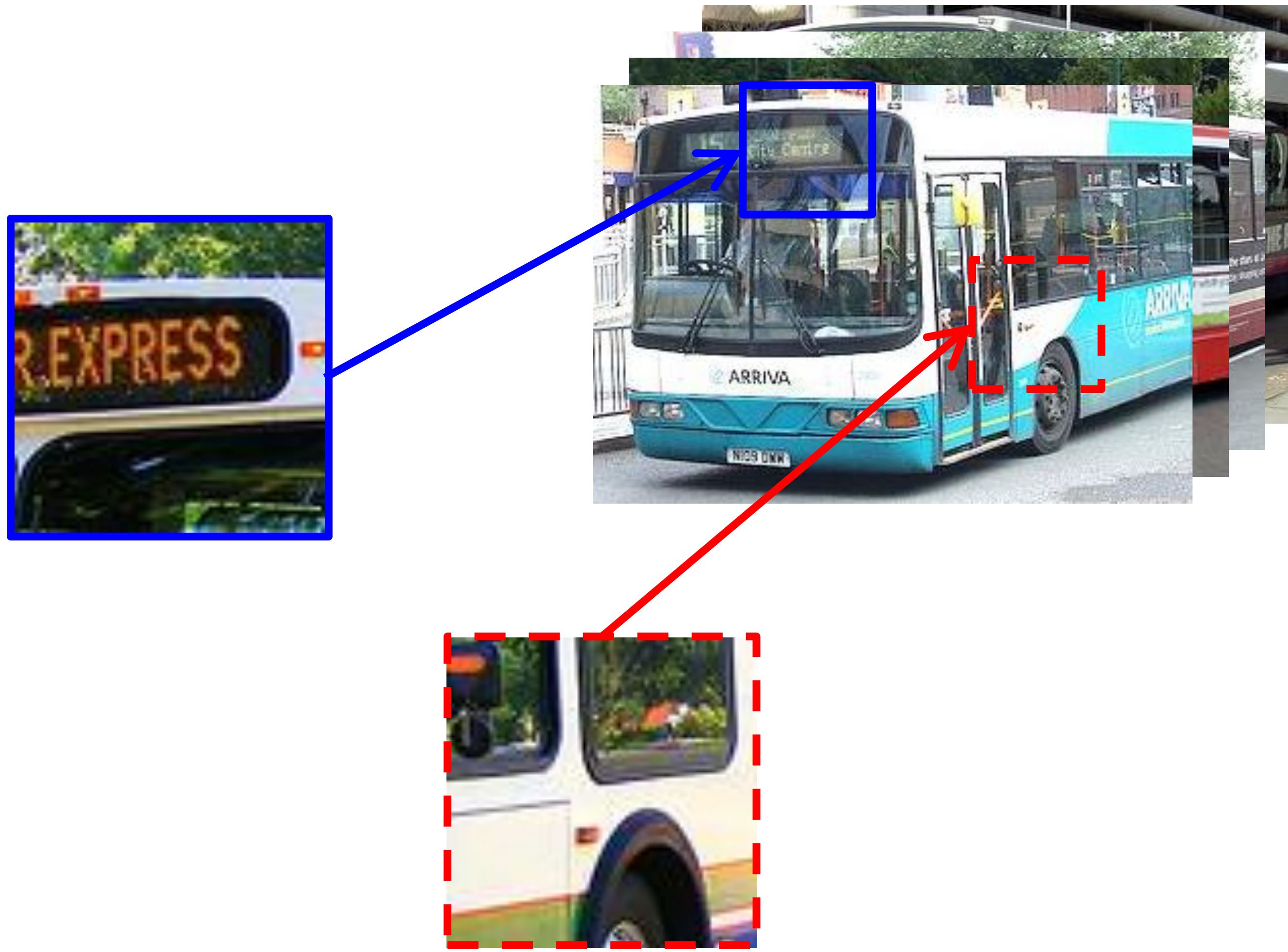

house, where the professor lived without his wife and child; or so
he said jokingly sometimes: “Here’s where I live. My house.” His
daughter often added, without resentment, for the visitor’s infor-
mation, “It started out to be for me, but it’s really his.” And she
might reach in to bring forth an inch-high table lamp with fluted
shade, or a blue dish the size of her little fingernail, marked
“Kitty” and half full of eternal milk, but she was sure to replace
these, after they had been admired, pretty near exactly where they
had been. The little house was very orderly, and just big enough
for all it contained, though to some tastes the bric-à-brac in the
parlor might seem excessive. The daughter’s preference was for the
store-bought gimmicks and appliances, the toasters and carpet
sweepers of Lilliput, but she knew that most adult visitors would

Context Prediction for Images



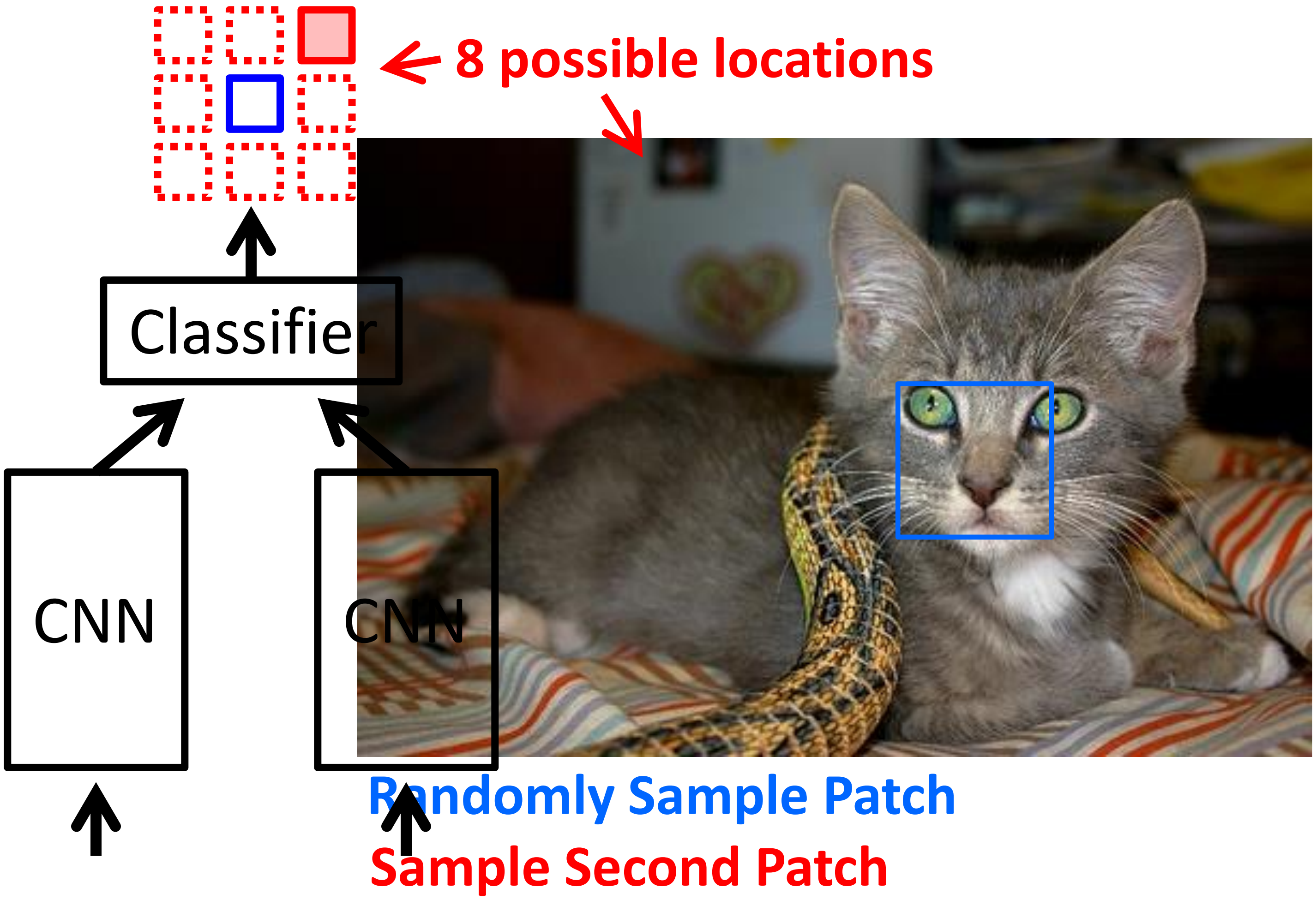
Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Semantics from a non-semantic task



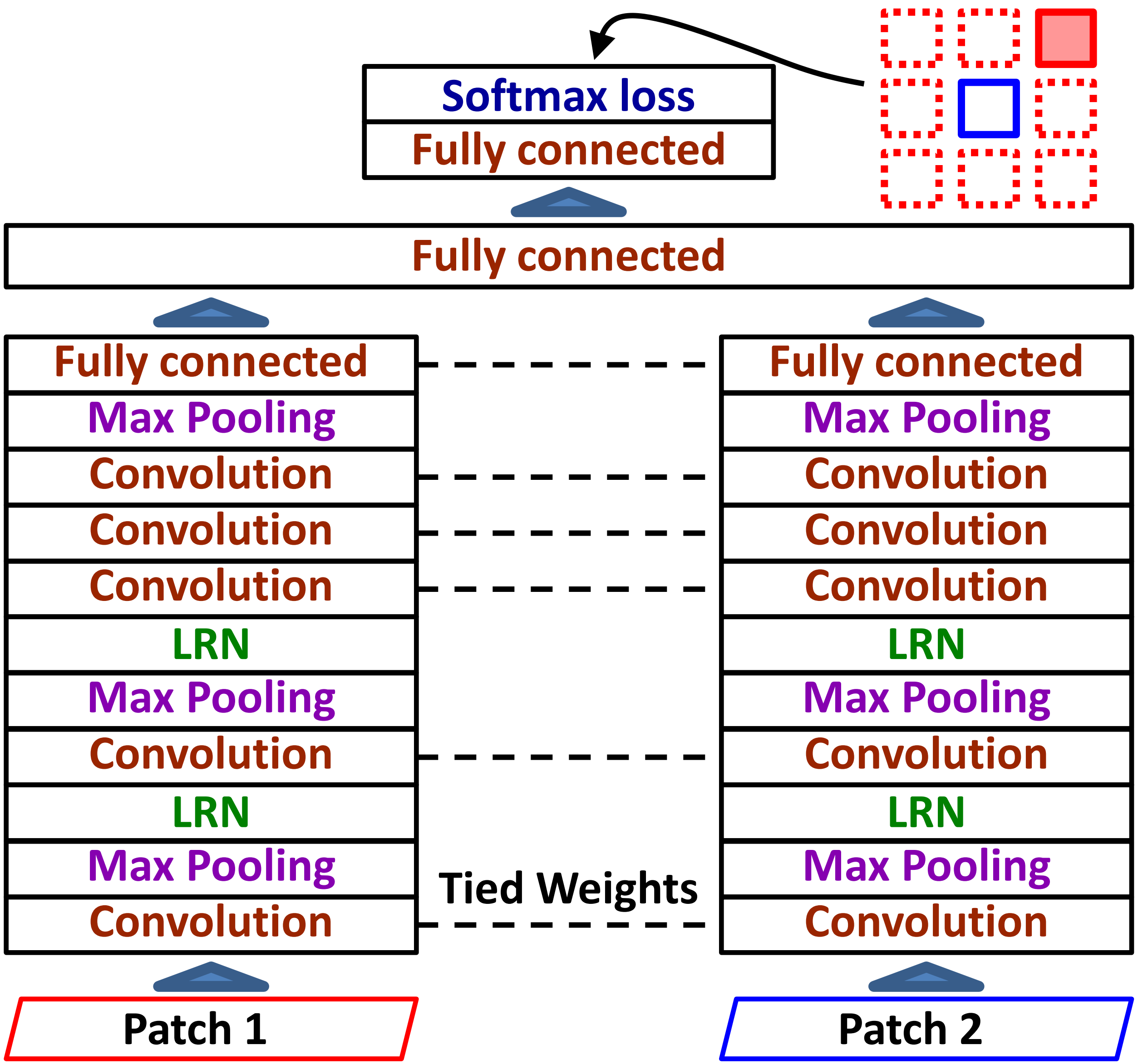
Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Relative Position Task

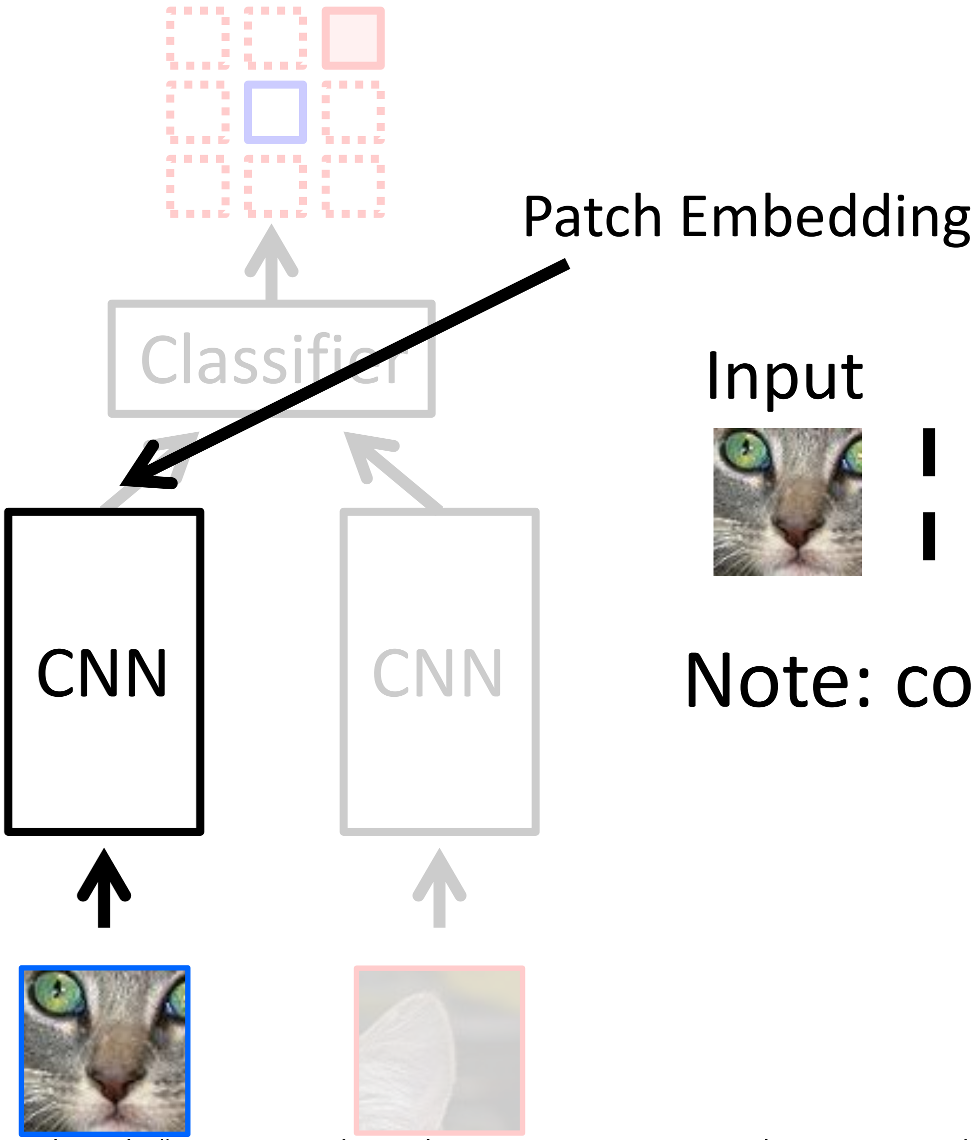


Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Architecture



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

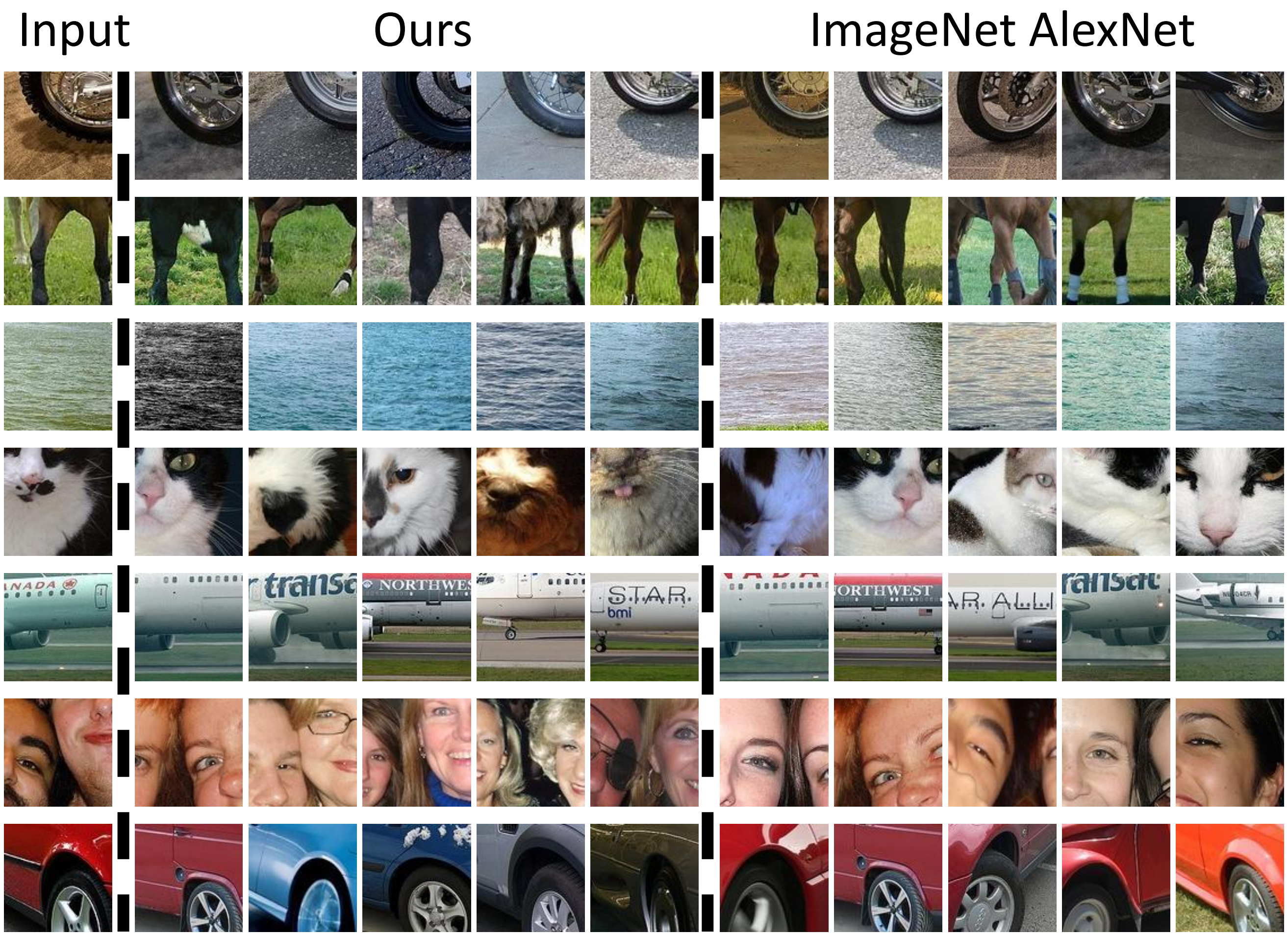


Input | Nearest Neighbors

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

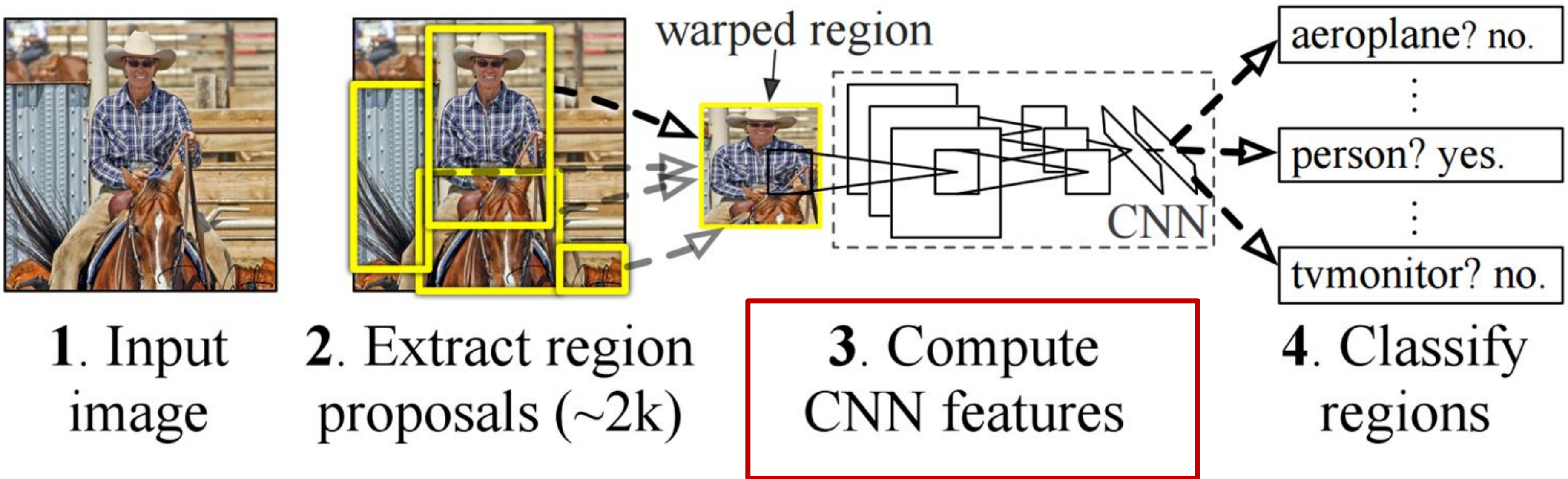
Doersch et al., "Unsupervised Visual Representation Learning by Context Prediction", ICCV 2015

What is learned?



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

Pre-Training for R-CNN

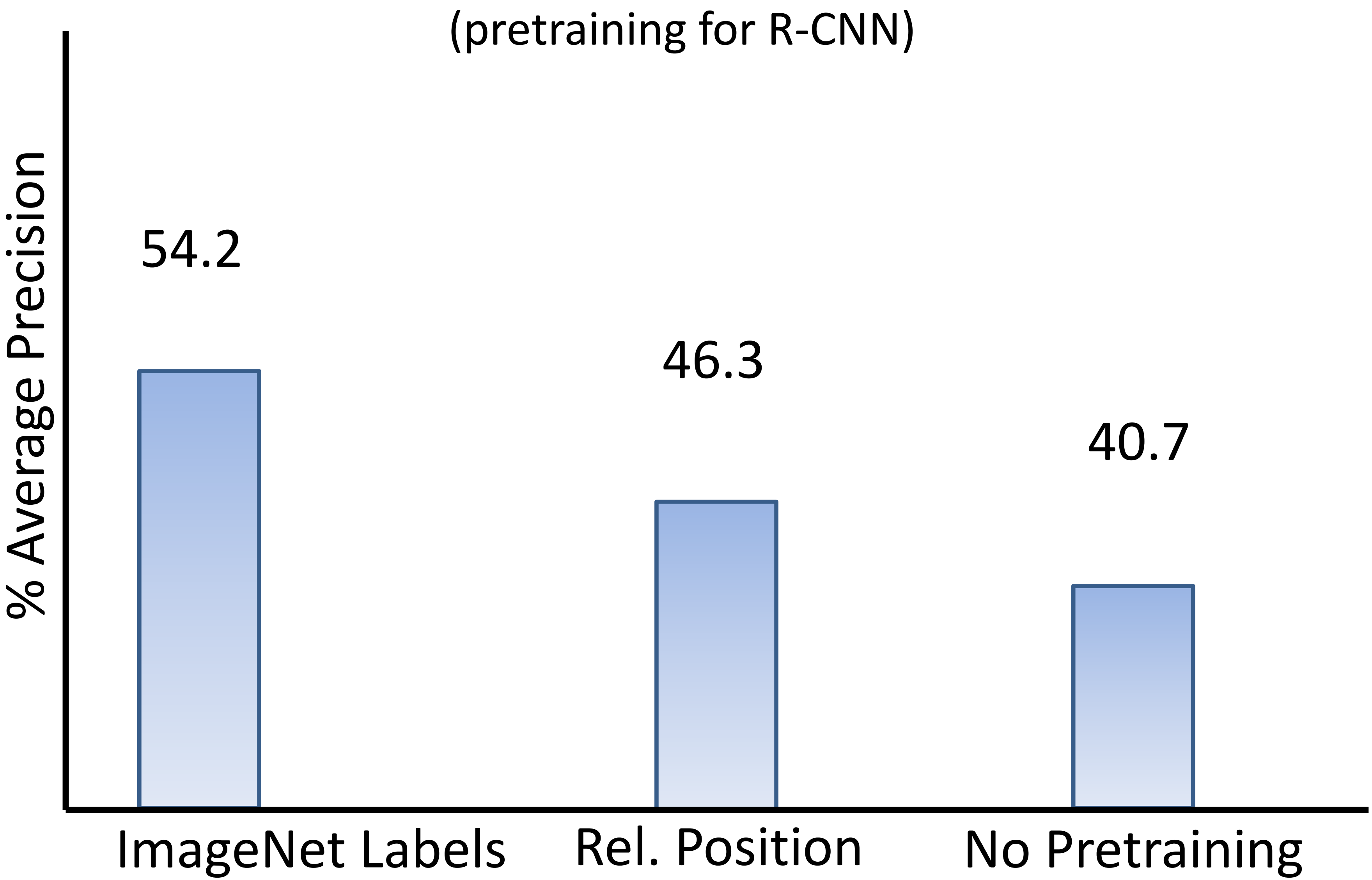


Pre-train on relative-position task, w/o labels

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

[Girshick et al. 2014]

VOC 2007 Performance



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

Which will be better?

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

Shuffle and Learn: Unsupervised Learning using Temporal Order Verification

Ishan Misra, C. Lawrence Zitnick, and Martial Hebert

ECCV 2016

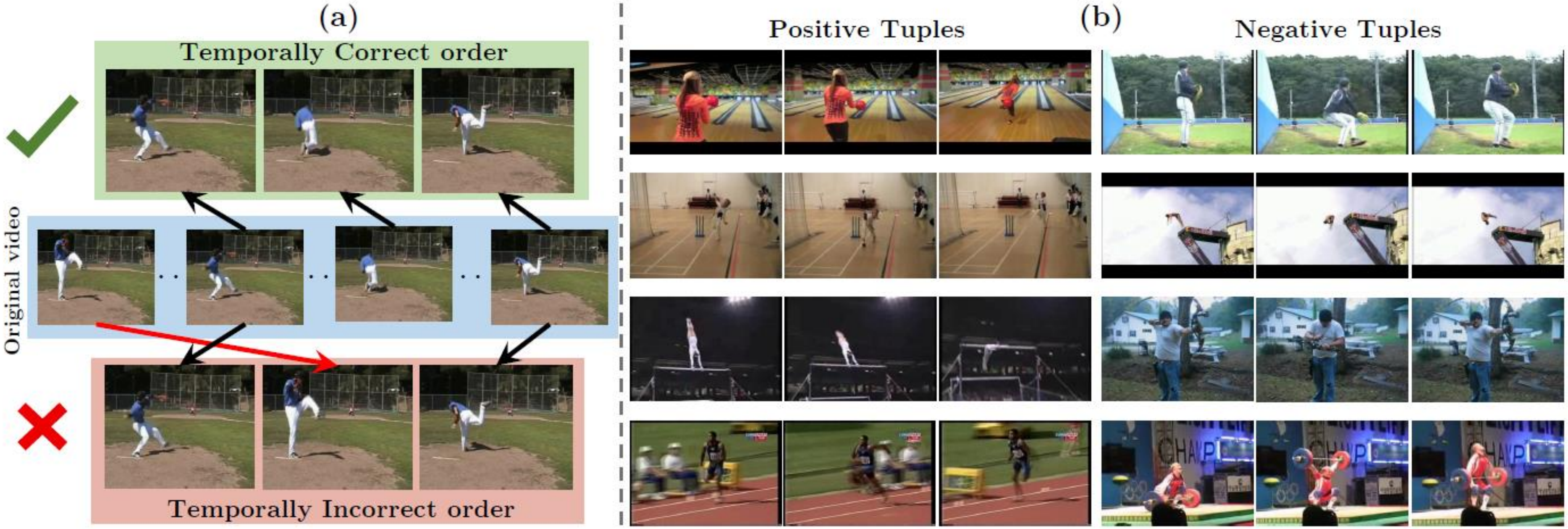


Fig. 1: **(a)** A video imposes a natural temporal structure for visual data. In many cases, one can easily verify whether frames are in the correct temporal order (shuffled or not). Such a simple sequential verification task captures important spatiotemporal signals in videos. We use this task for unsupervised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

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

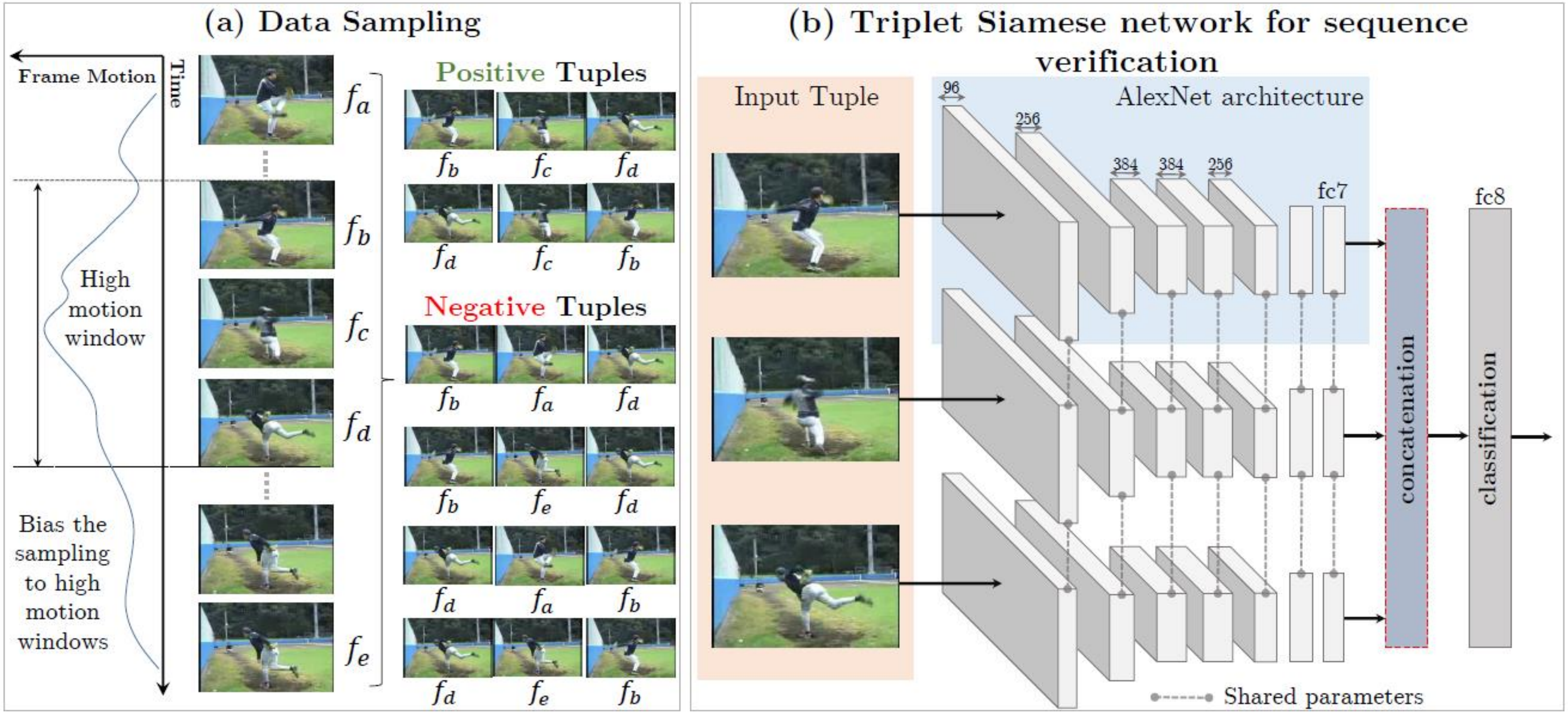


Fig. 2: **(a)** We sample tuples of frames from high motion windows in a video. We form positive and negative tuples based on whether the three input frames are in the correct temporal order. **(b)** Our triplet Siamese network architecture has three parallel network stacks with shared weights upto the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

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

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

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

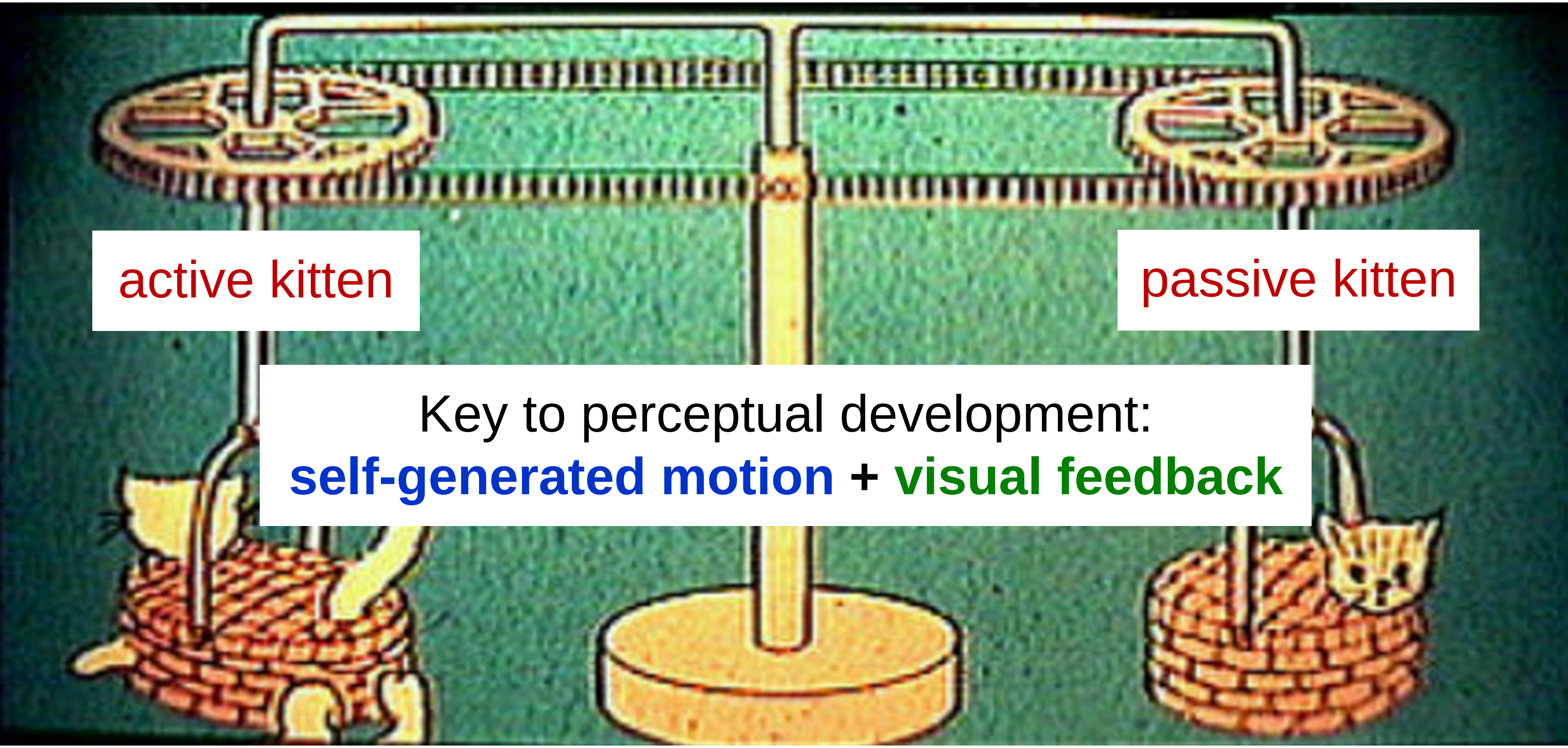
Misra et al., "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification", ECCV 2016

Learning image representations tied to ego-motion

Dinesh Jayaraman and Kristen Grauman

ICCV 2015

The kitten carousel experiment
[Held & Hein, 1963]



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

Problem with today's visual learning

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

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

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

Our idea: Ego-motion ↔ vision

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

+

Ego-motion motor signals **Unlabeled video**

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

Ego-motion ↔ vision: view prediction

After moving:

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

Ego-motion ↔ vision for recognition

Learning this connection requires:

- Depth, 3D geometry
- Semantics
- Context

Also key to recognition!

Can be learned without manual labels!

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

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

Approach idea: Ego-motion equivariance

Invariant features: unresponsive to some classes of transformations

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

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

"equivariance map"

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

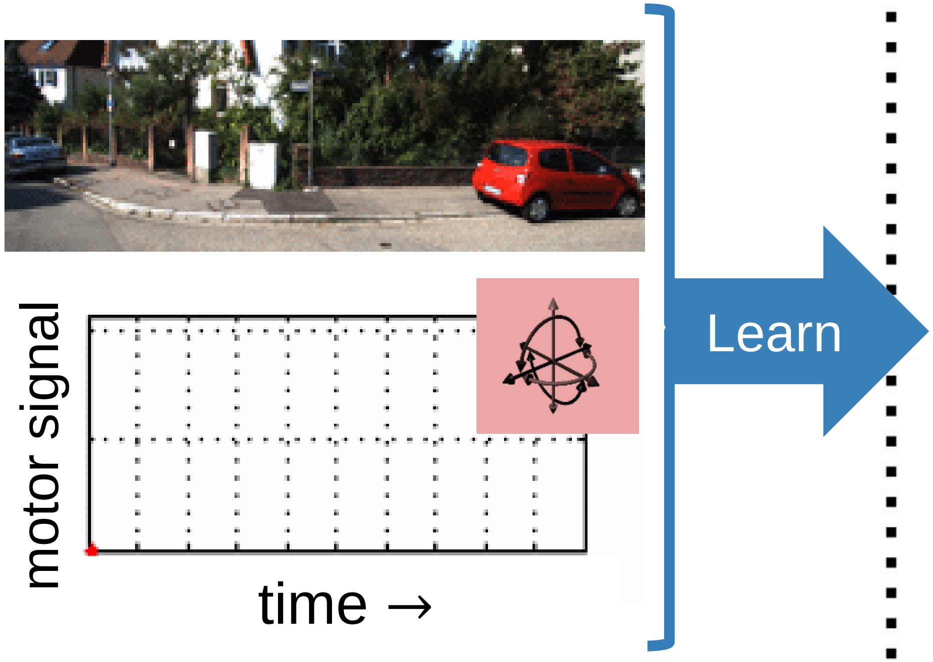
Invariance *discards* information;
equivariance *organizes* it.

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

Approach idea: Ego-motion equivariance

Training data

Unlabeled video +
motor signals



Equivariant embedding

organized by ego-motions

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

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

Approach overview

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

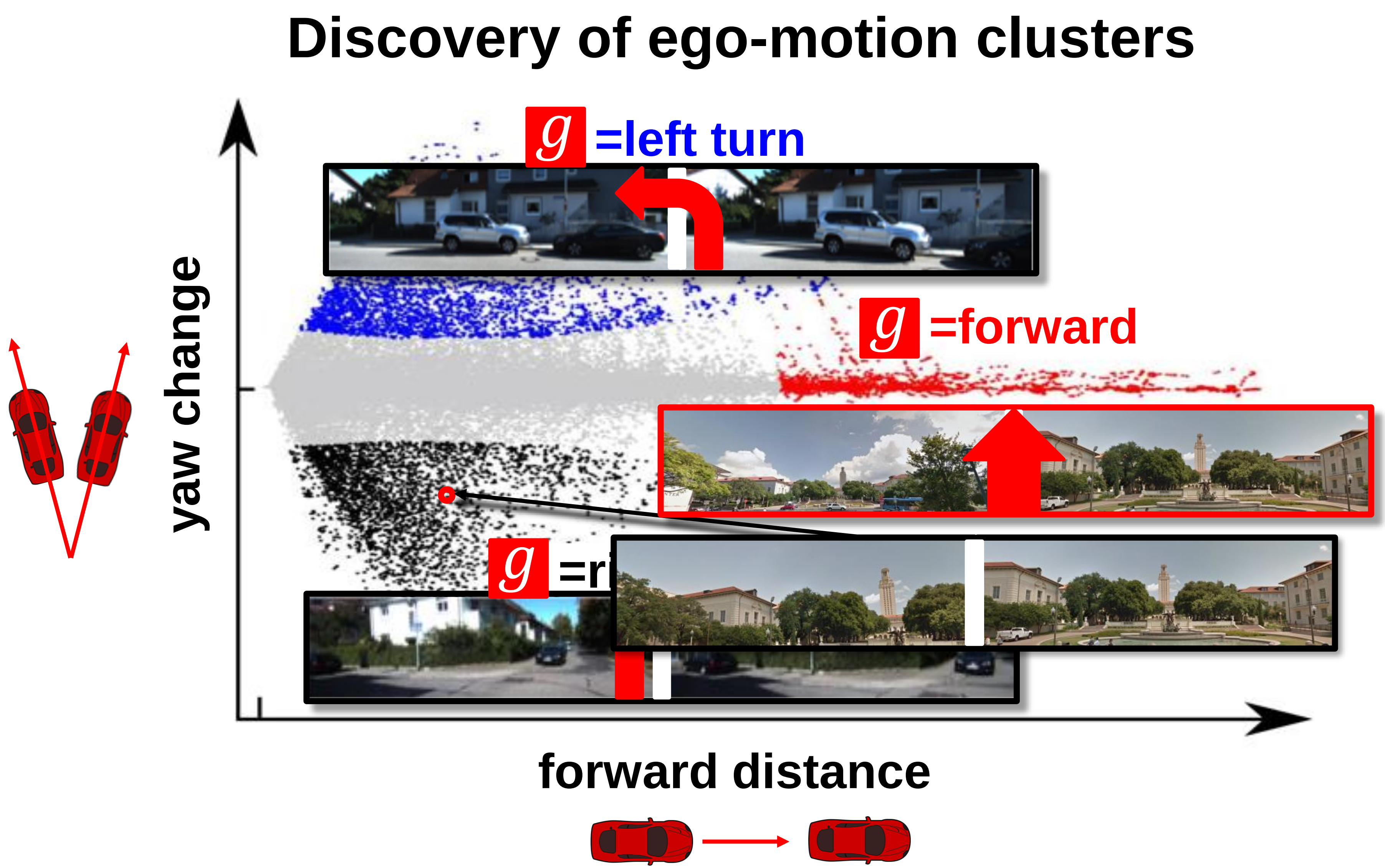
1. Extract training frame pairs from video

2. Learn ego-motion-equivariant image features

3. Train on target recognition task in parallel

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

Training frame pair mining

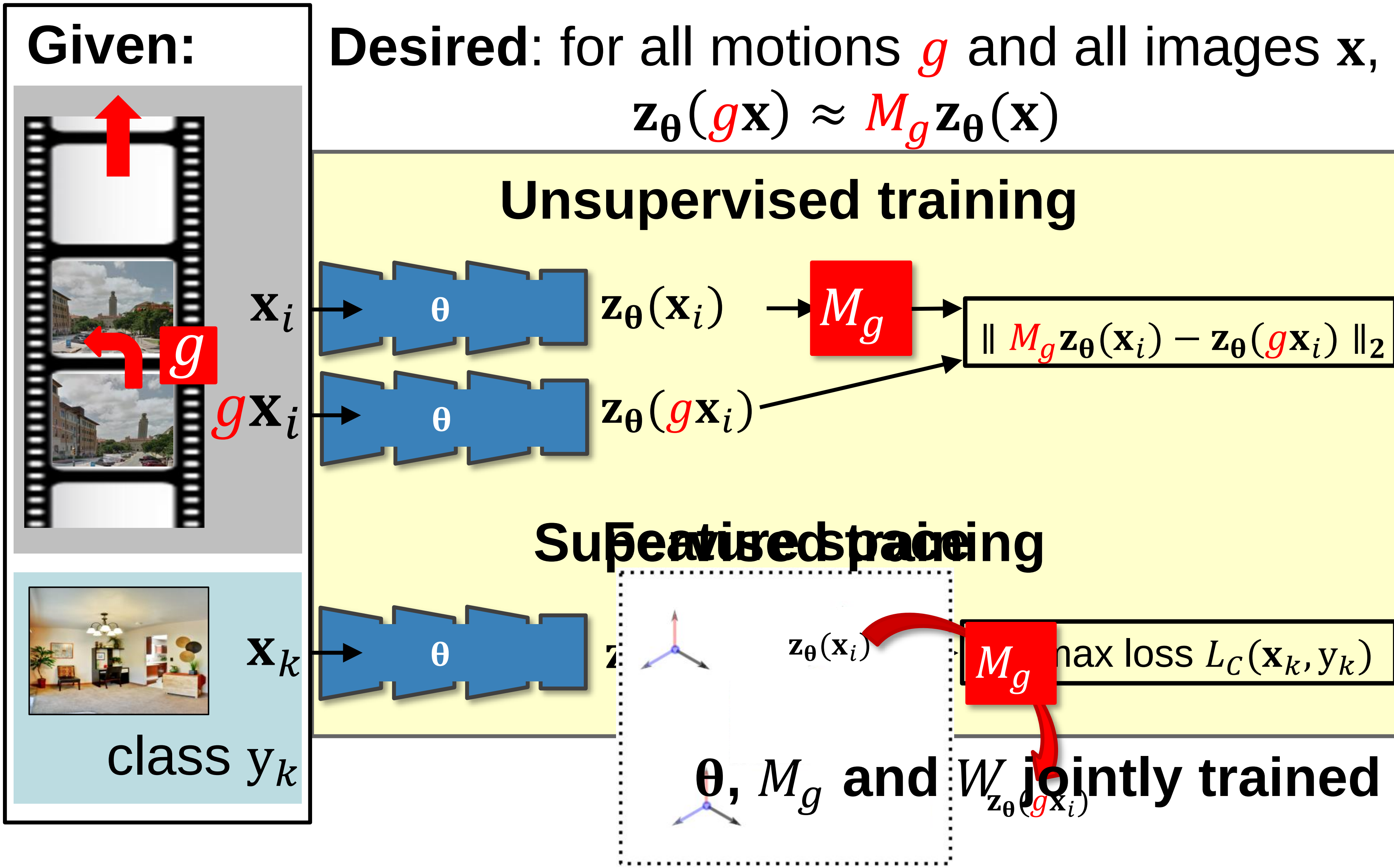


Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Ego-motion equivariant feature learning

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

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



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

Results: Recognition

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

Geiger et al, IJRR ’13

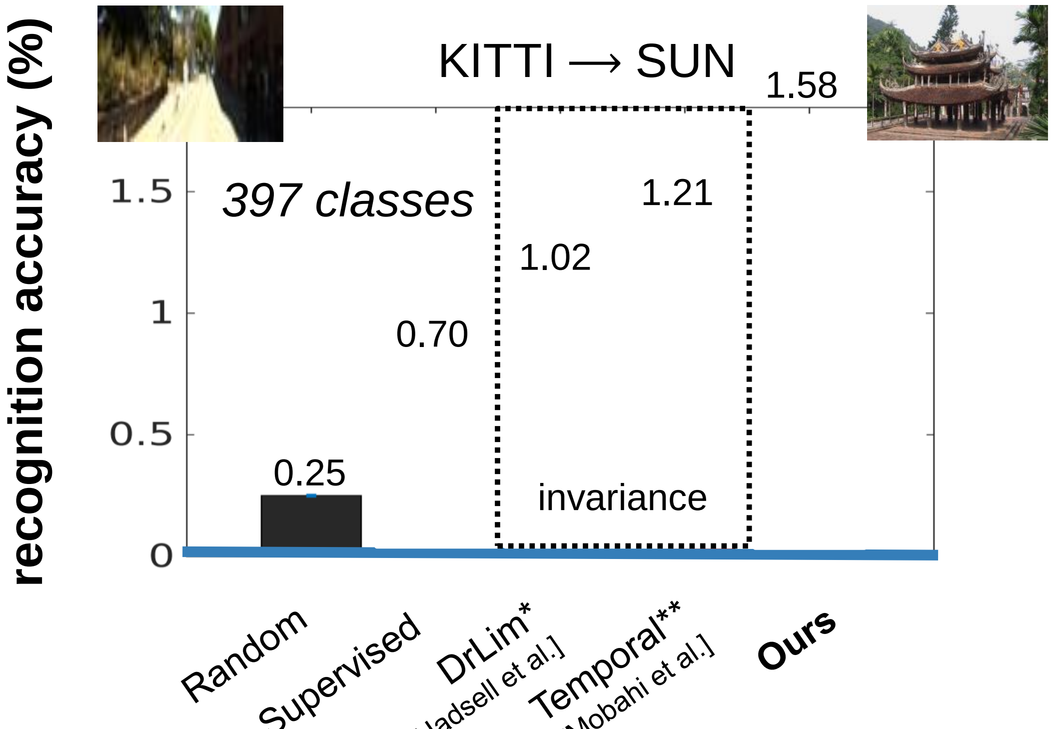
Exploit features for **static scene classification**
(SUN, 397 classes)

Xiao et al, CVPR ’10

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

Results: Recognition

Do ego-motion equivariant features improve recognition?



6 labeled training examples per class

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

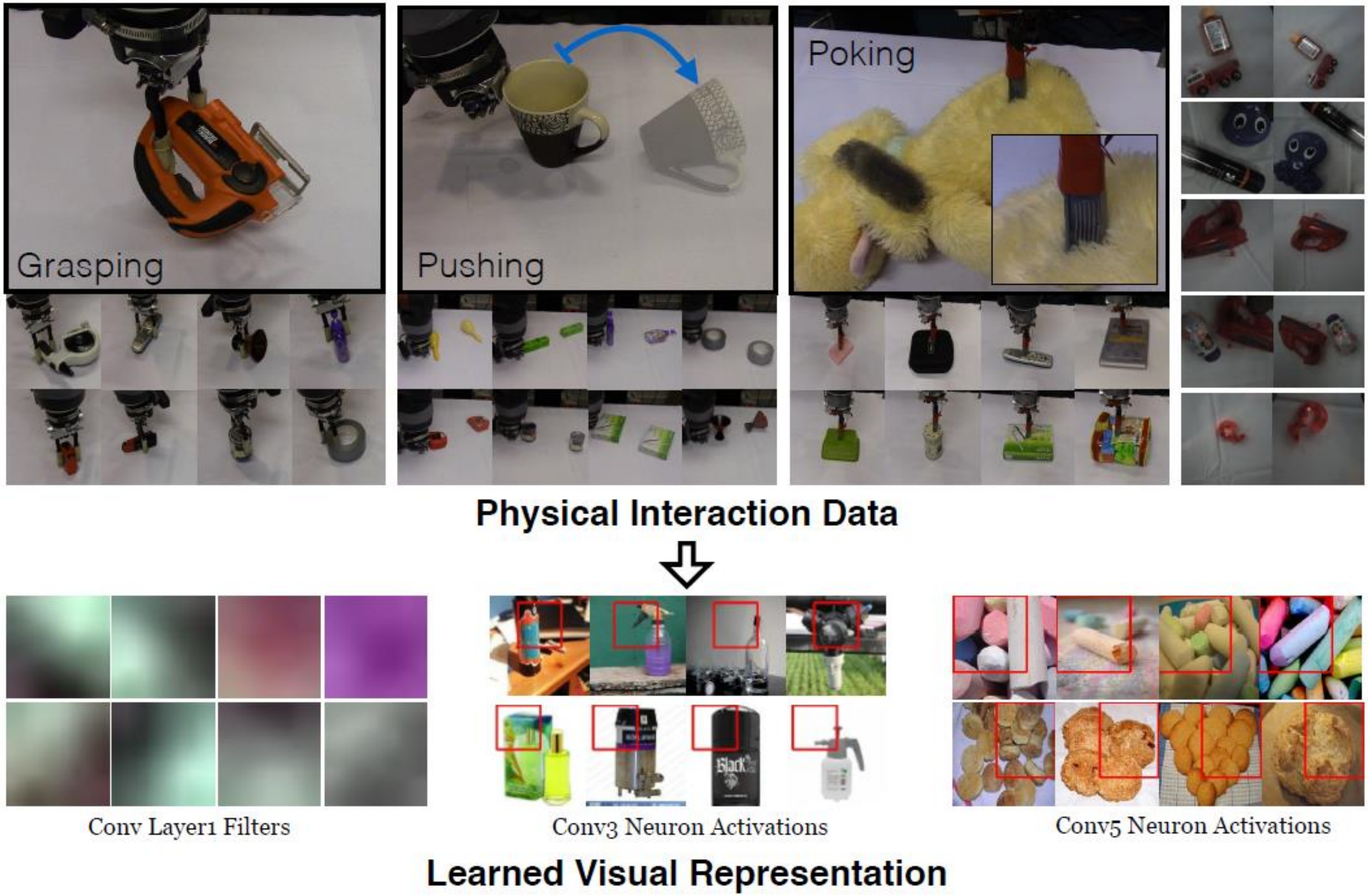
Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

The Curious Robot: Learning Visual Representations via Physical Interactions

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

ECCV 2016

Embodied representations



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

Grasping

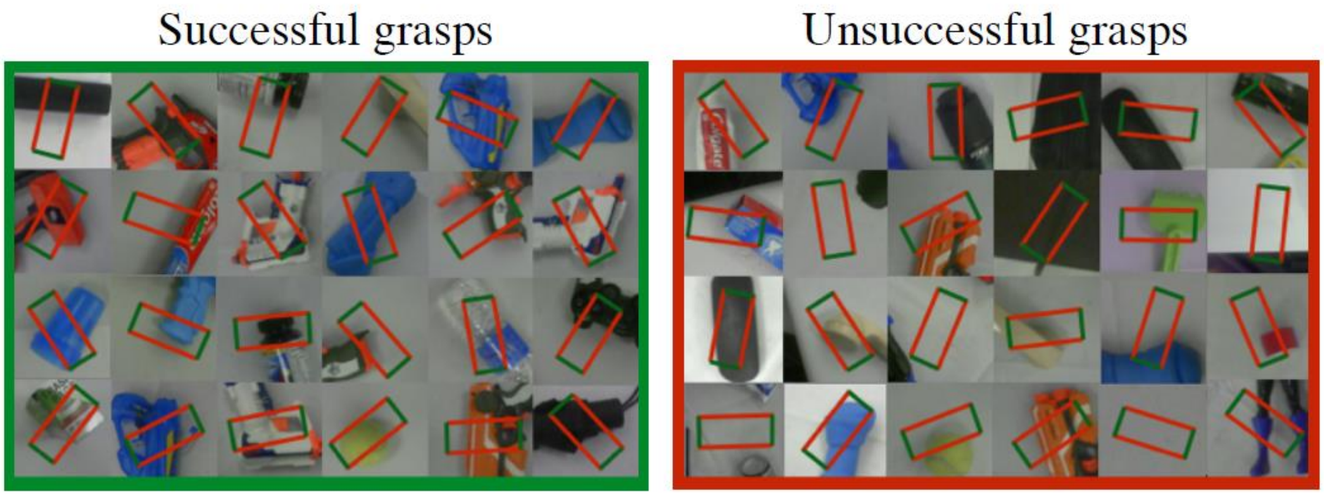


Fig. 2. Examples of successful (left) and unsuccesful grasps (right). We use a patch based representation: given an input patch we predict 18-dim vector which represents whether the center location of the patch is graspable at $0°$, $10°$, $\ldots 170°$.

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

Pushing

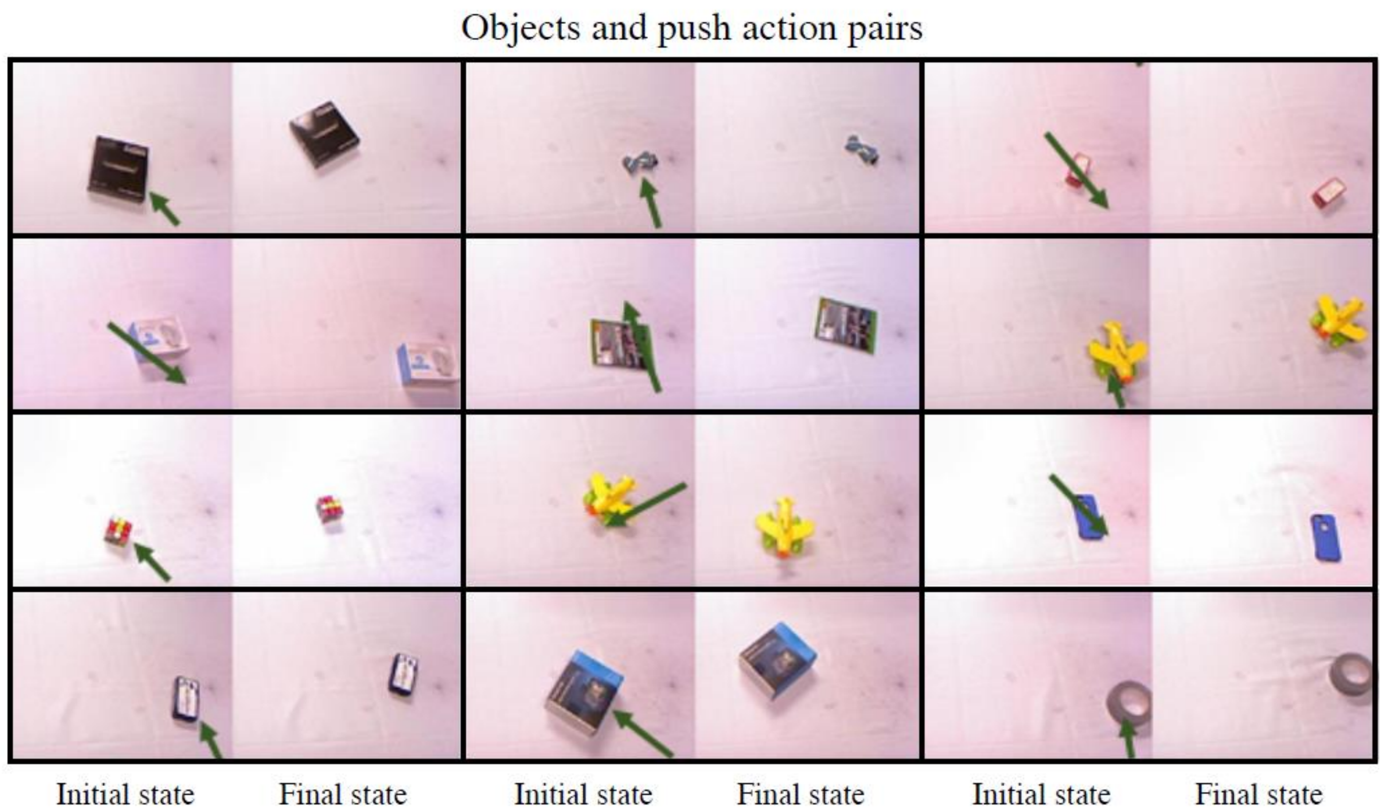


Fig. 4. Examples of initial state and final state images taken for the push action. The arrows demonstrate the direction and magnitude of the push action.

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

Poking

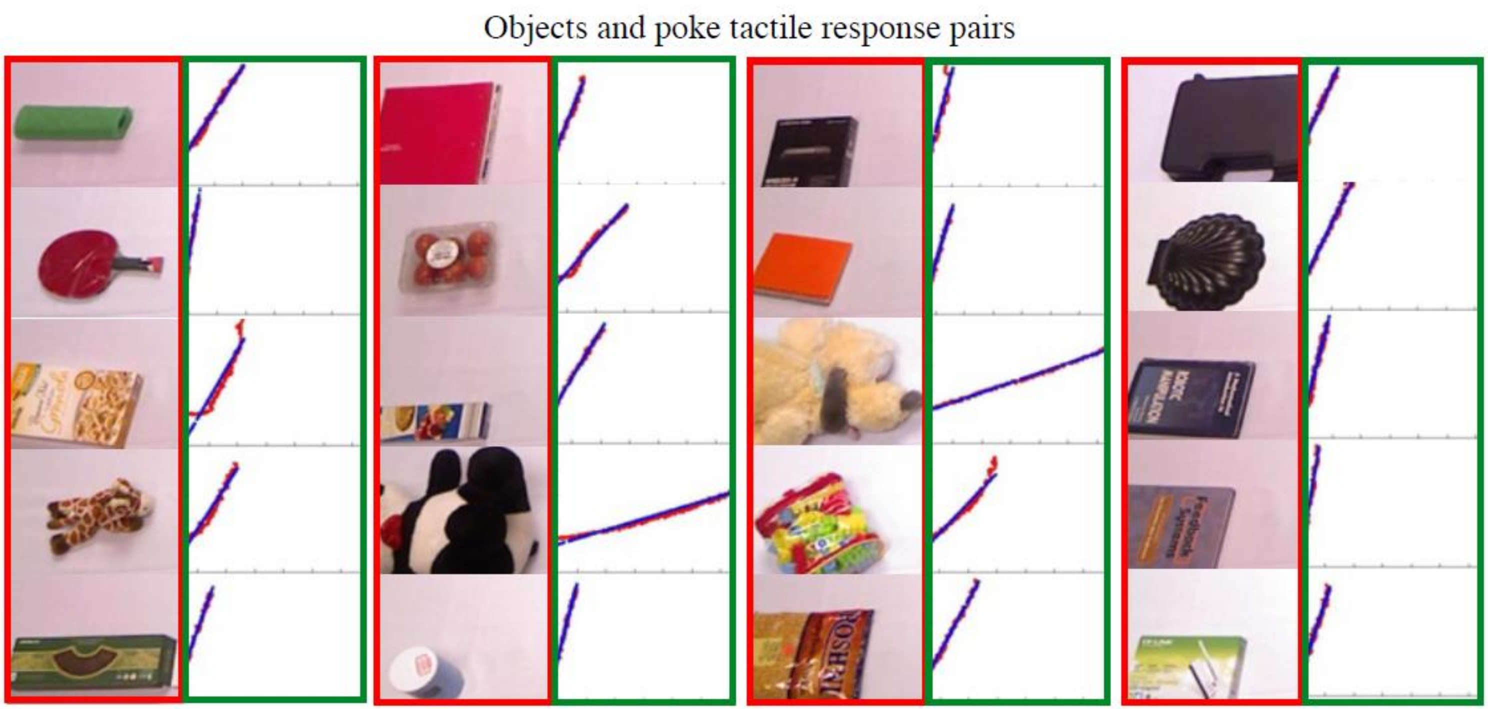


Fig. 6. Examples of the data collected by the poking action. On the left we show the object poked, and on the right we show force profiles as observed by the tactile sensor.

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

Representations from interactions

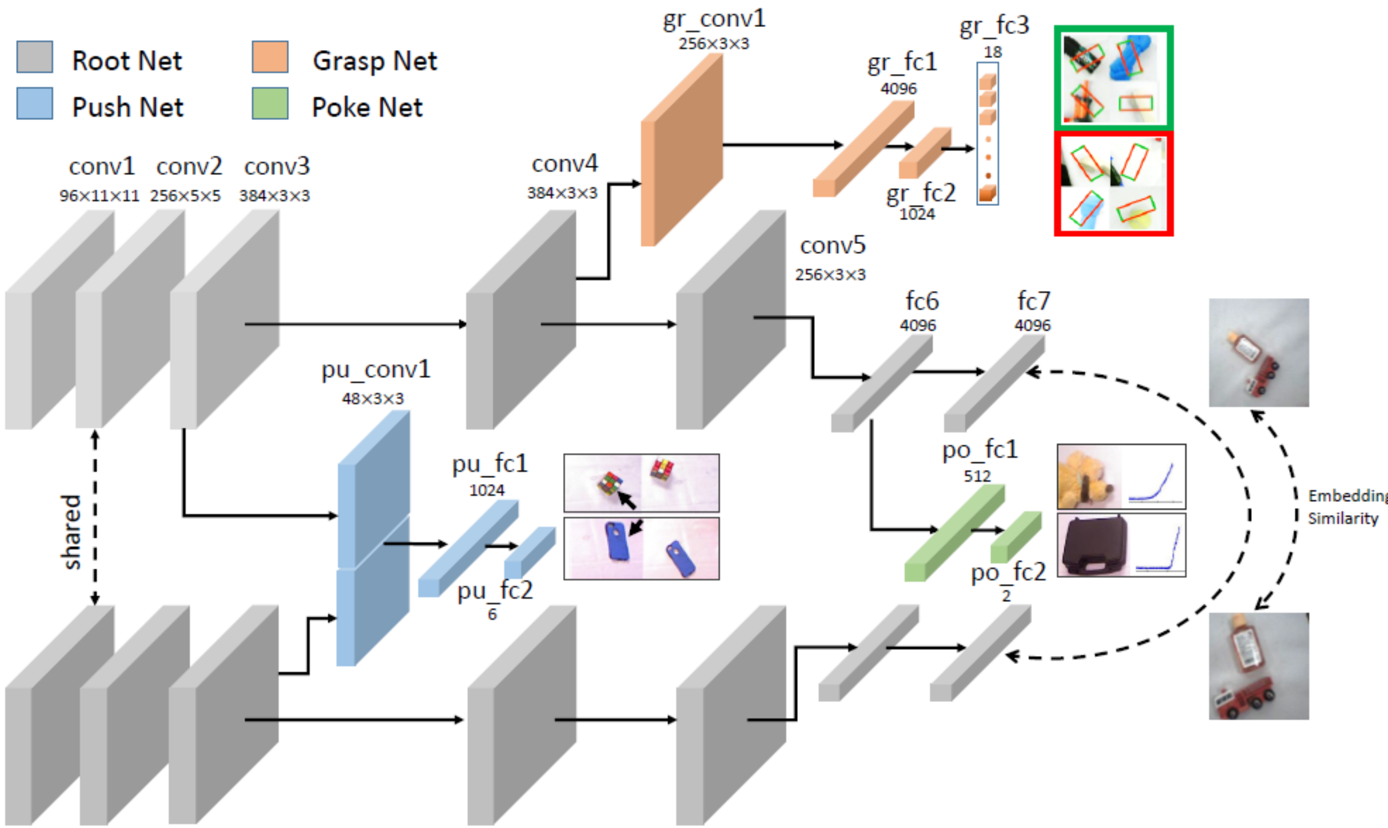


Fig. 8. Our shared convolutional architecture for four different tasks.

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

Classification/retrieval performance

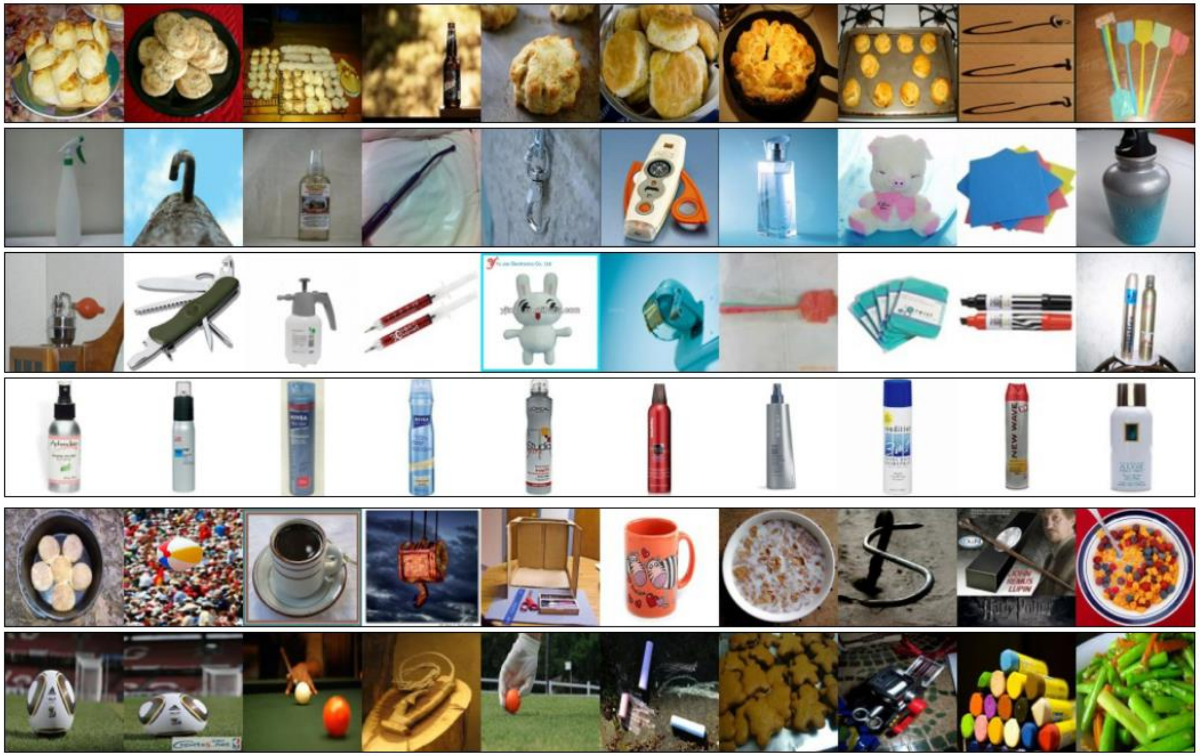


Fig. 10. The first column corresponds to query image and rest show the retrieval. Note how the network learns that cups and bowls are similar (row 5).

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

Classification/retrieval performance

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

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

Table 2. Image Retrieval with Recall@k metric

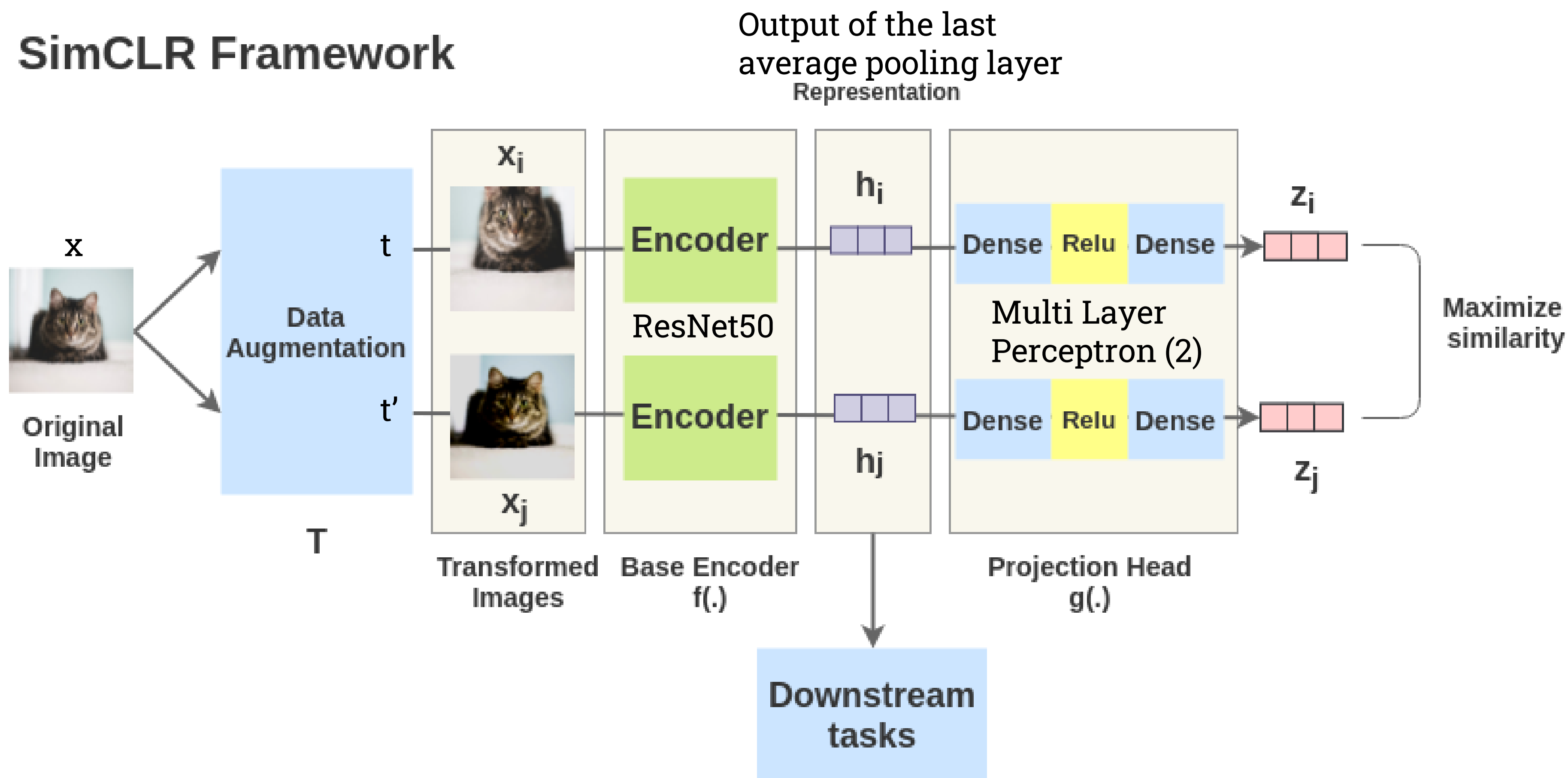
	Instance level				Category level			
	k=1	k=5	k=10	k=20	k=1	k=5	k=10	k=20
Random Network	0.062	0.219	0.331	0.475	0.150	0.466	0.652	0.800
Our Network	0.720	0.831	0.875	0.909	0.833	0.918	0.946	0.966
AlexNet	0.686	0.857	0.903	0.941	0.854	0.953	0.969	0.982

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

SimCLR - A Simple Framework for Contrastive Learning of Visual Representations

Chen et al., “SimCLR - A Simple Framework for Contrastive Learning of Visual Representations”, ICML 2020

SimCLR Framework



Chen et al., "SimCLR - A Simple Framework for Contrastive Learning of Visual Representations", ICML 2020

https://amitness.com/2020/03/illustrated-simclr/

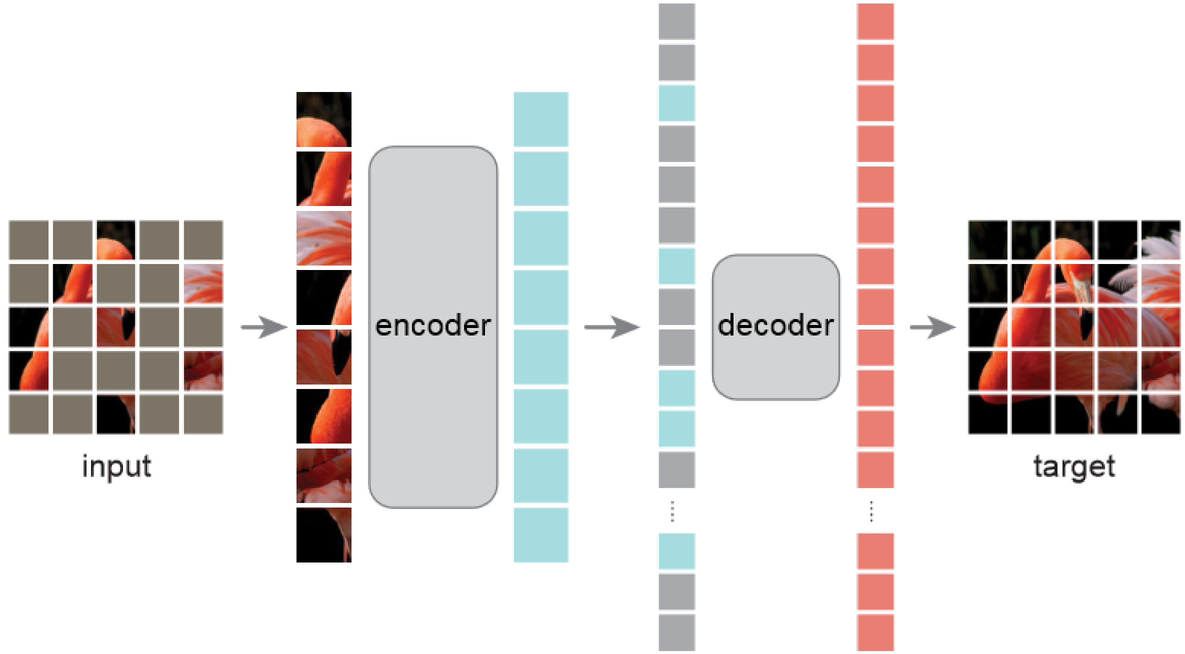


Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

He et al., “Masked Autoencoders Are Scalable Vision Learners”, CVPR 2022

Project Pitches