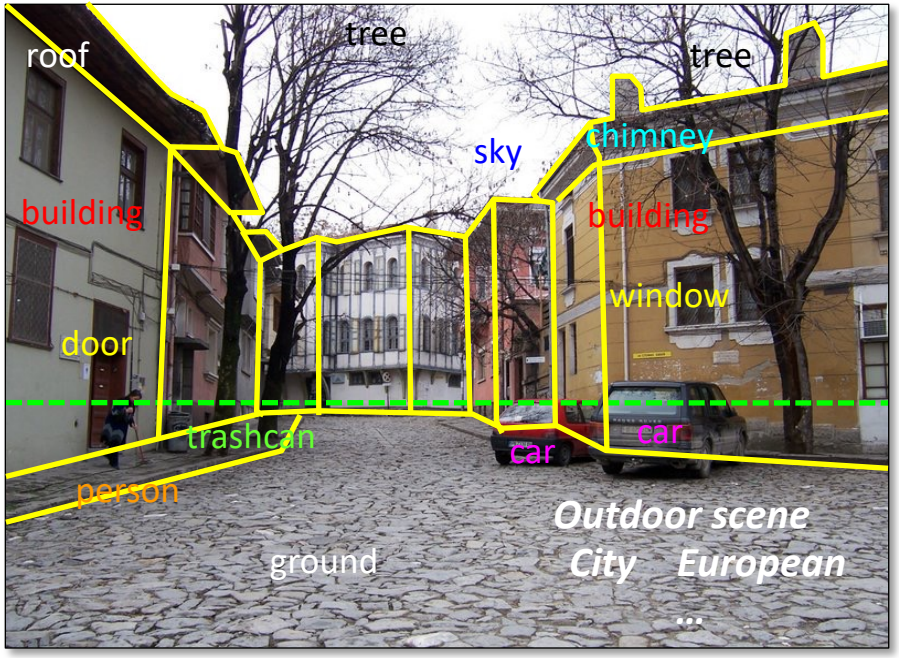
What kind of information can be extracted from an image?



Geometric information

Semantic information

Source: L. Lazebnik

Today's plan

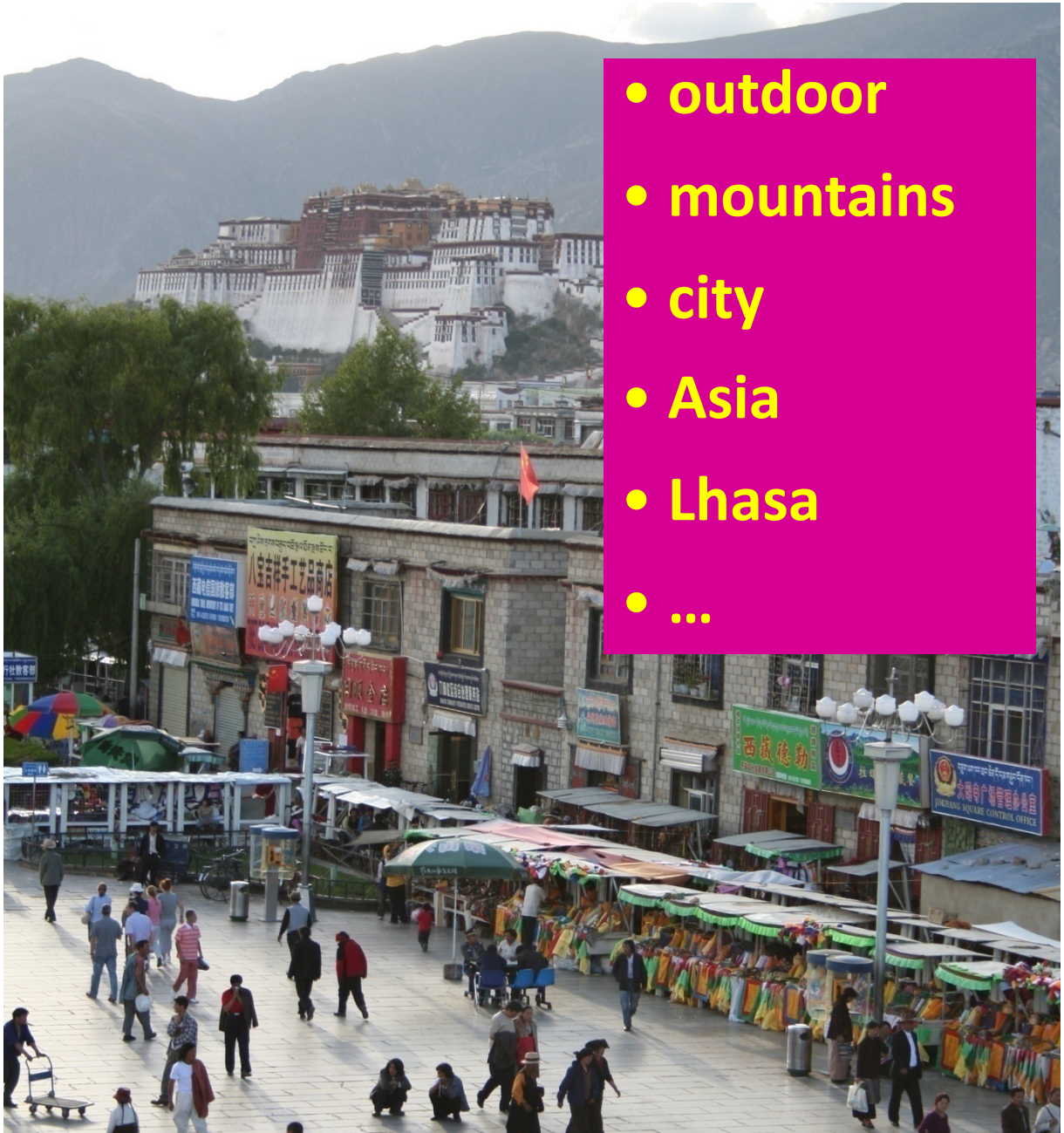
- Extracting semantic information from images
  - *The statistical learning framework*
  - *Neural networks and how to train them*
  - *Building blocks for designing neural networks for images*
  - *Extending neural networks for more complex tasks*
  - *Sequence modeling*

Common recognition tasks

Slide from L. Lazebnik.

Adapted from Fei-Fei Li

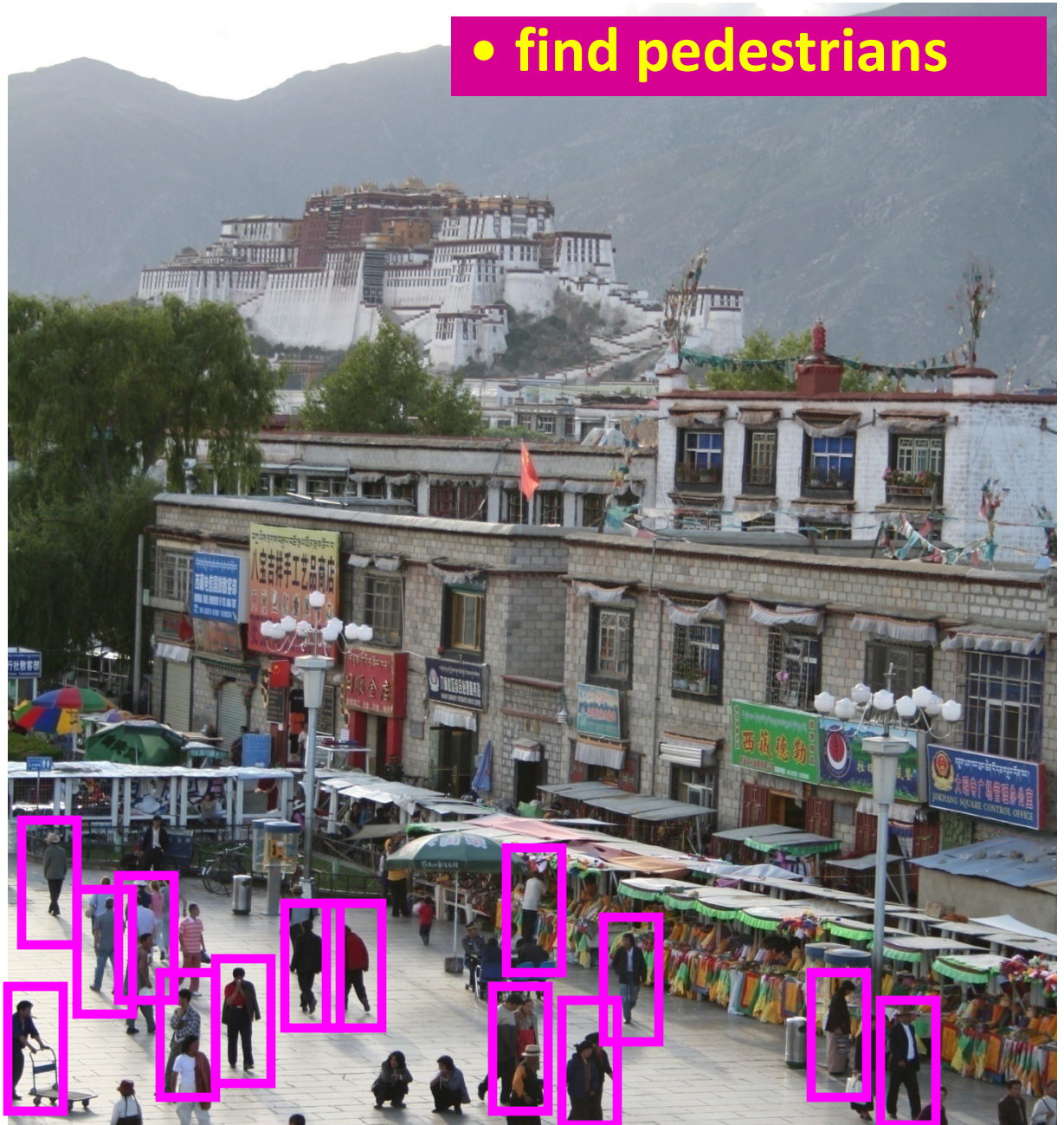
Image classification and tagging



Slide from L. Lazebnik.

Adapted from Fei-Fei Li

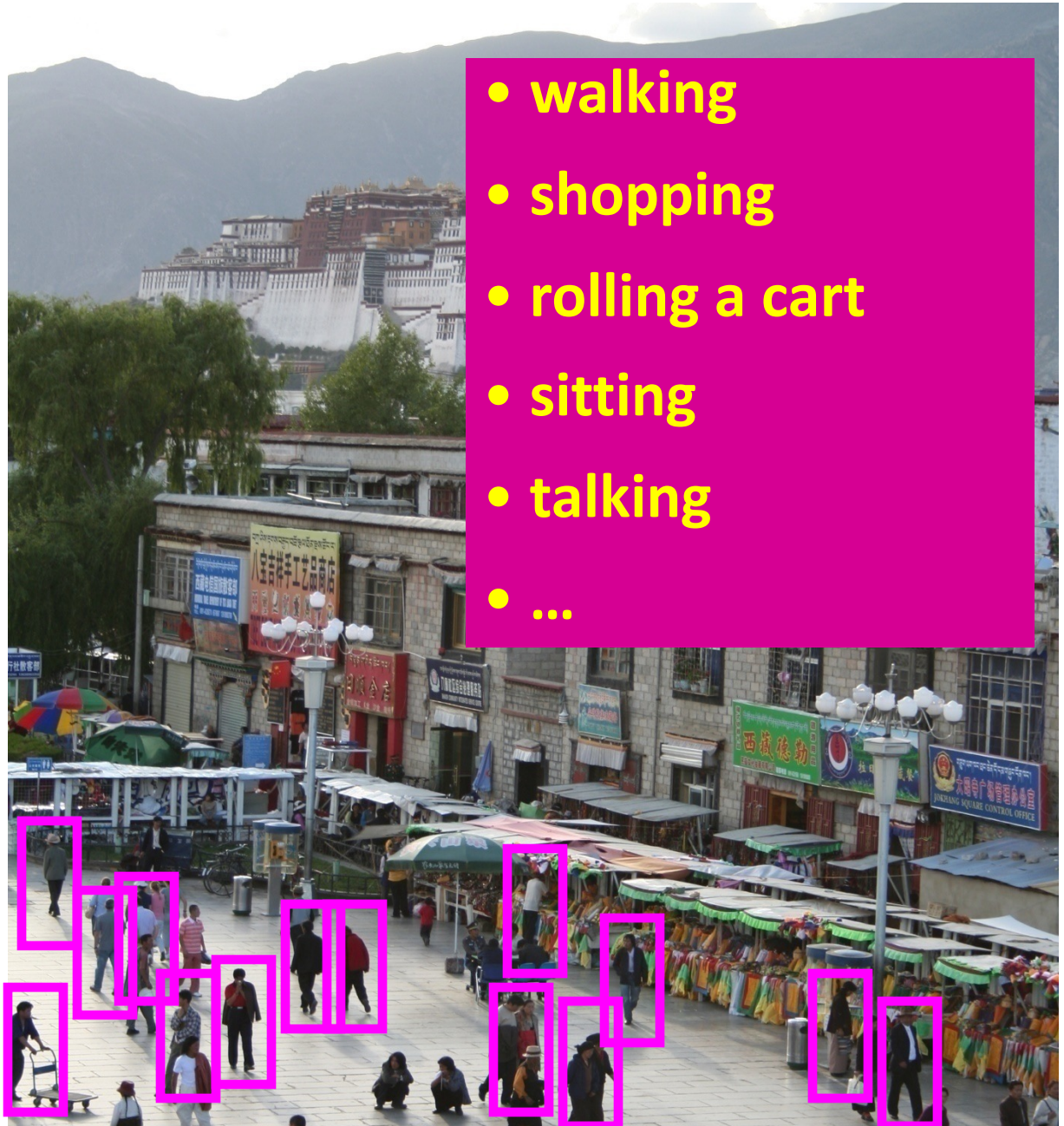
Object detection



Slide from L. Lazebnik.

Adapted from Fei-Fei Li

Activity recognition



Slide from L. Lazebnik.

Adapted from Fei-Fei Li

Semantic segmentation

Slide from L. Lazebnik.

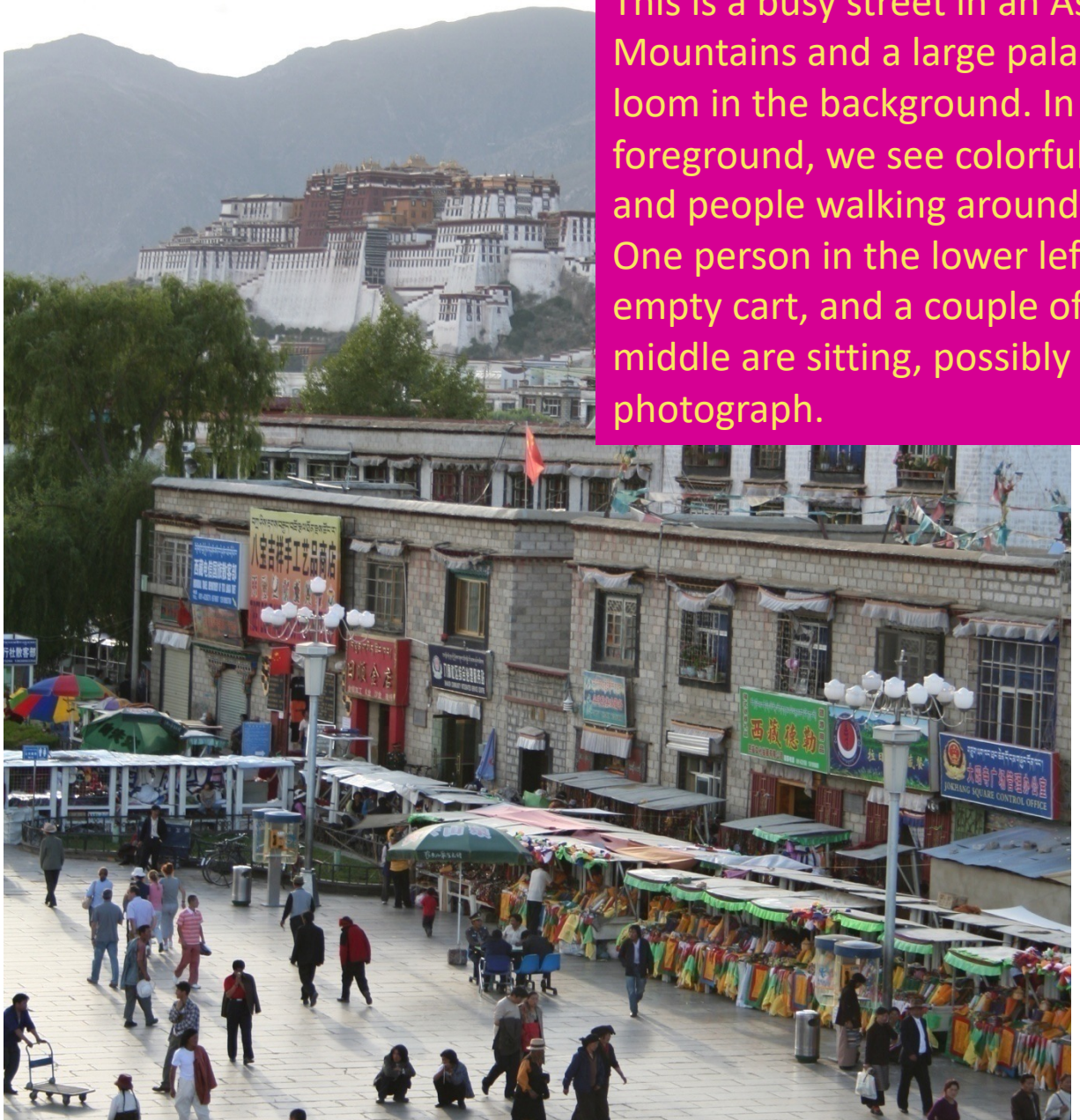
Adapted from Fei-Fei Li

Semantic segmentation

Slide from L. Lazebnik.

Adapted from Fei-Fei Li

Image description



This is a busy street in an Asian city. Mountains and a large palace or fortress loom in the background. In the foreground, we see colorful souvenir stalls and people walking around and shopping. One person in the lower left is pushing an empty cart, and a couple of people in the middle are sitting, possibly posing for a photograph.

Slide from L. Lazebnik.

Adapted from Fei-Fei Li

The statistical learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

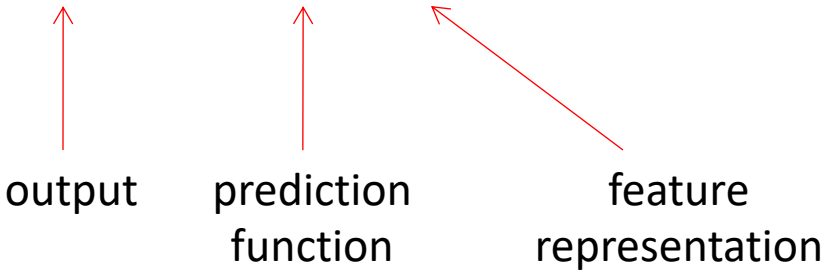
f() = “tomato”

f() = “cow”

Slide from L. Lazebnik.

The statistical learning framework

$$y = f(\mathbf{x})$$



- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide from L. Lazebnik.

Steps

Training

Training Images

Training Labels

Image Features

Training

Learned model

Testing

Test Image

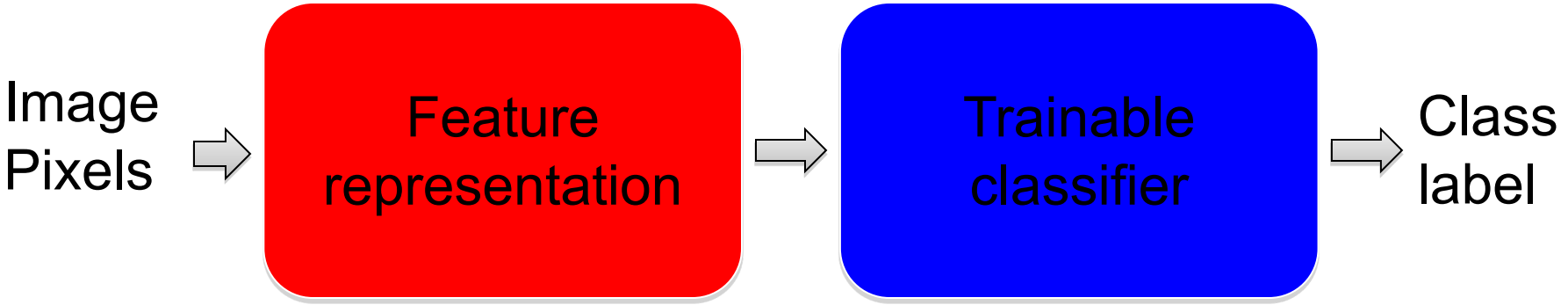
Image Features

Learned model

Prediction

Slide credit: D. Hoiem

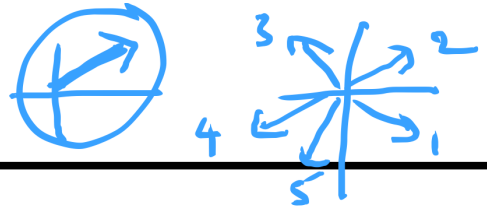
“Classic” recognition pipeline

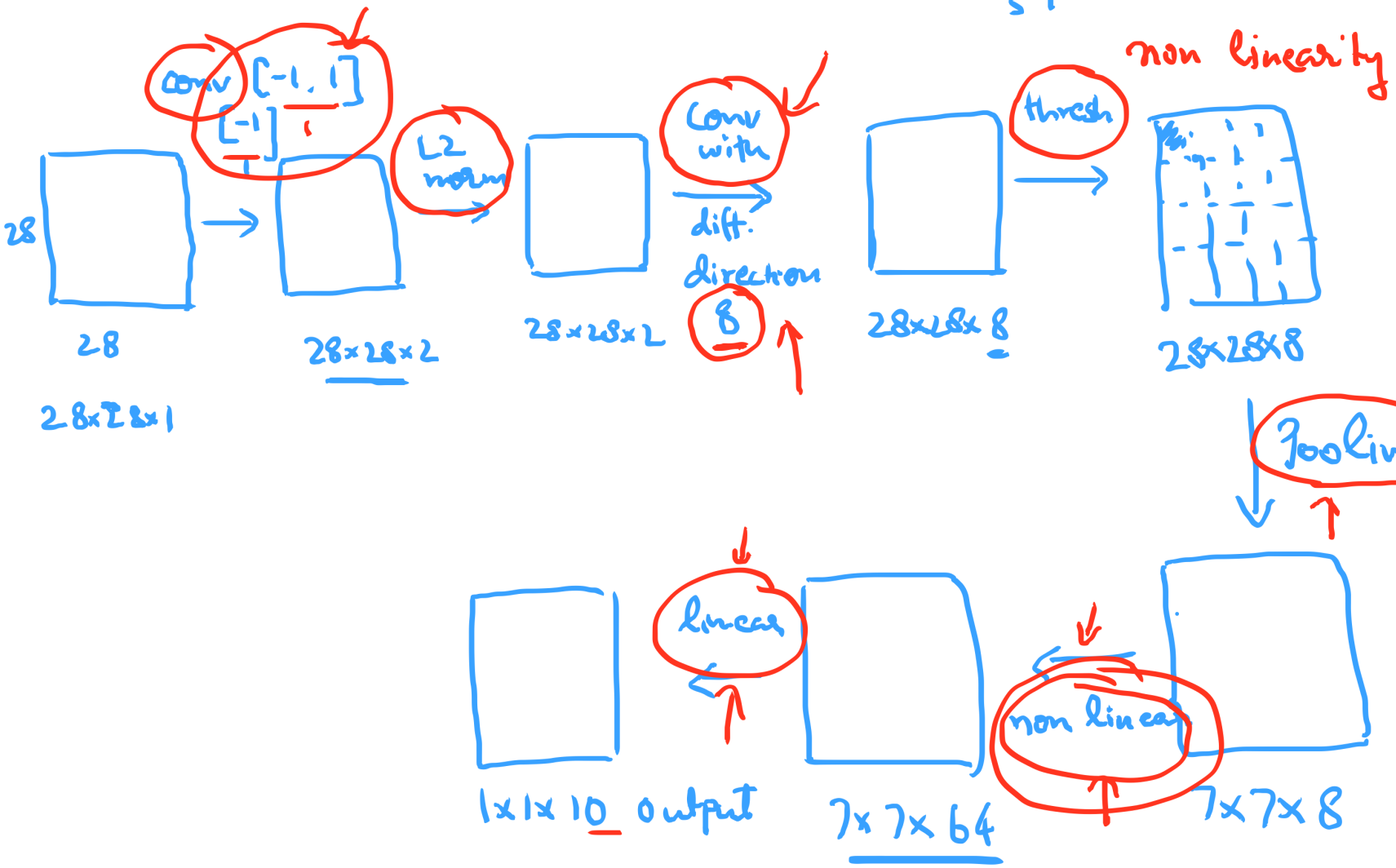


- Hand-crafted feature representation
- Off-the-shelf trainable classifier

Slide from L. Lazebnik.

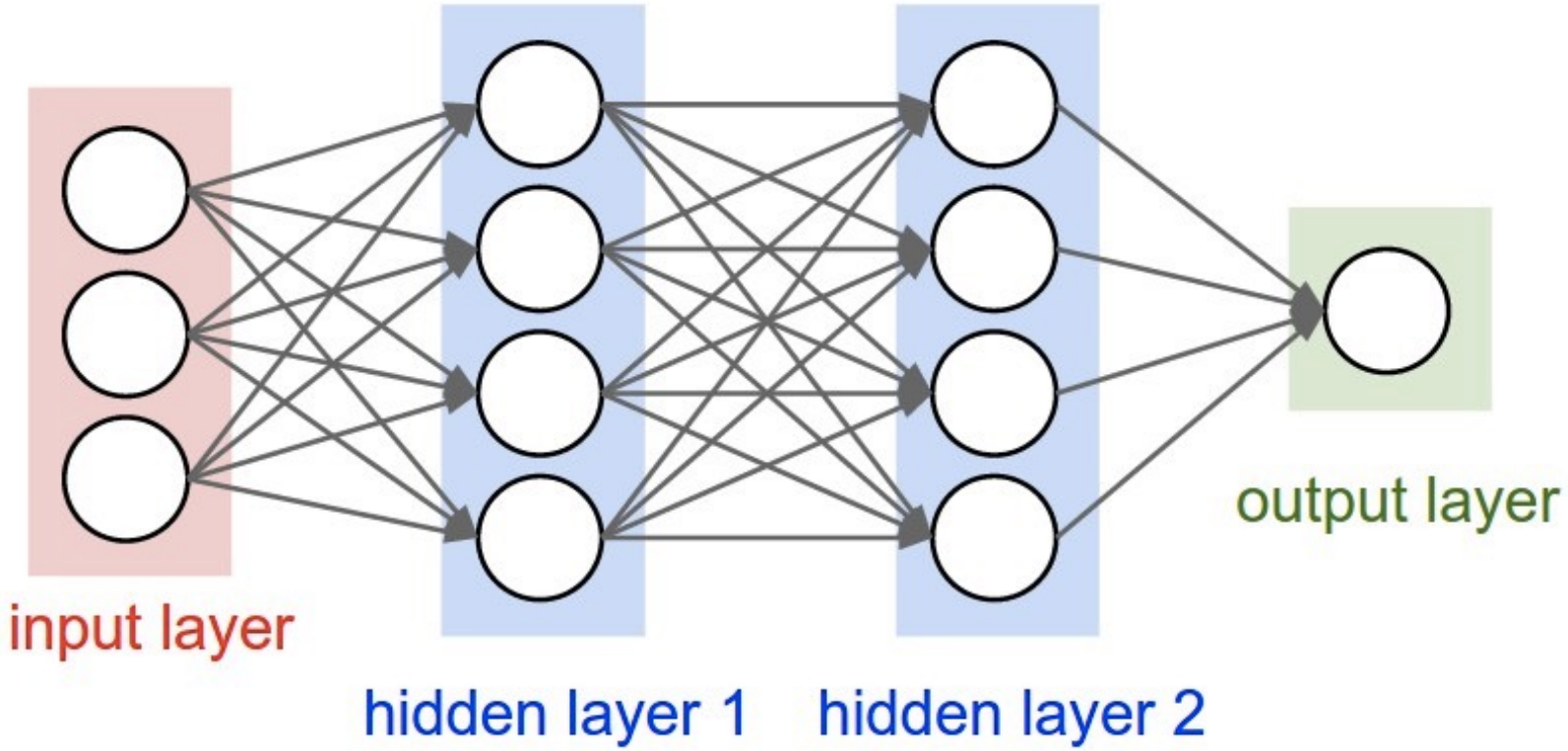
Digit classification





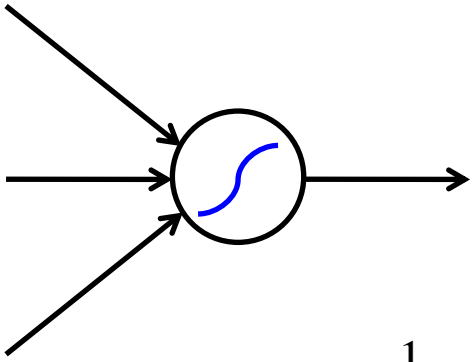
Neural networks or Multi-layer perceptrons

- Flexible way of defining non-linear functions
  - Linear layers interleaved with non-linearities

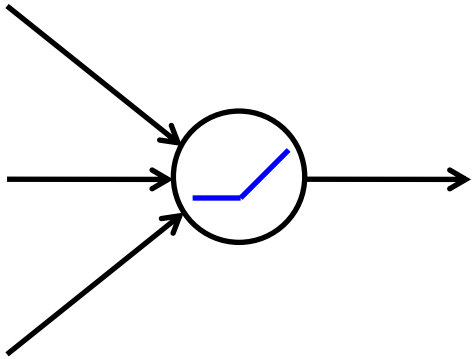


Neural networks or Multi-layer perceptrons

- Flexible way of defining non-linear functions
  - Linear layers interleaved with non-linearities
  - To be trainable, the nonlinearity should be *differentiable*



Sigmoid: $g(t) = \frac{1}{1+e^{-t}}$



Rectified linear unit (ReLU): $g(t) = \max(0,t)$

Training of multi-layer networks

- Find network weights to minimize the prediction loss between true and estimated labels of training examples:

$$E(\mathbf{w}) = \sum_i l(\mathbf{x}_i, y_i; \mathbf{w})$$

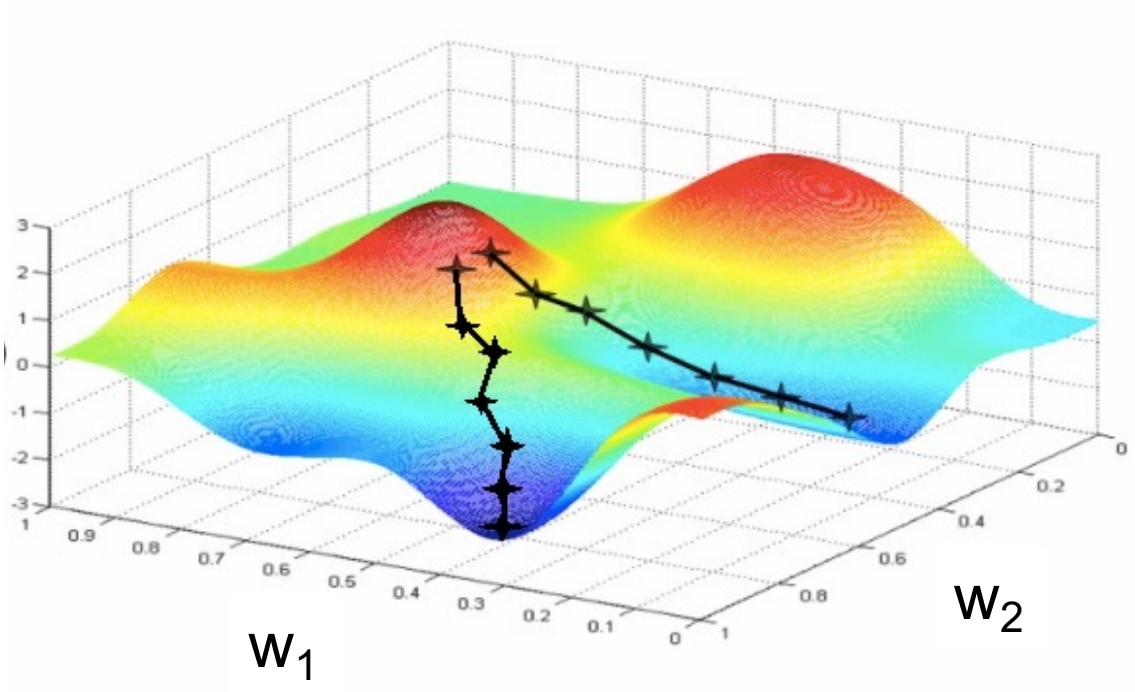
- Possible losses (for binary problems):
  - Quadratic loss: $l(\mathbf{x}_i, y_i; \mathbf{w}) = (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$
  - Log likelihood loss: $l(\mathbf{x}_i, y_i; \mathbf{w}) = -\log P_{\mathbf{w}}(y_i \mid \mathbf{x}_i)$
  - Hinge loss: $l(\mathbf{x}_i, y_i; \mathbf{w}) = \max(0, 1 - y_i f_{\mathbf{w}}(\mathbf{x}_i))$

Training of multi-layer networks

- Find network weights to minimize the prediction loss between true and estimated labels of training examples:

$$E(\mathbf{w}) = \sum_i l(\mathbf{x}_i, y_i; \mathbf{w})$$

- Update weights by **gradient descent:** $\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$



Training of multi-layer networks

- Find network weights to minimize the prediction loss between true and estimated labels of training examples:

$$E(\mathbf{w}) = \sum_i l(\mathbf{x}_i, y_i; \mathbf{w})$$

- Update weights by **gradient descent:** $\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$

- **Back-propagation:** gradients are computed in the direction from output to input layers and combined using chain rule
- **Stochastic gradient descent:** compute the weight update w.r.t. one training example (or a small batch of examples) at a time, cycle through training examples in random order in multiple epochs

Back-propagation

Back-propagation

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} L(h(x_i; \boldsymbol{\theta}), y_i)$$

Convolutional network

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \lambda \frac{1}{N} \sum_{i=1}^{N} \nabla L(h(x_i; \boldsymbol{\theta}), y_i)$$

Gradient descent update

Slide from B. Hariharan

Computing the gradient of the loss

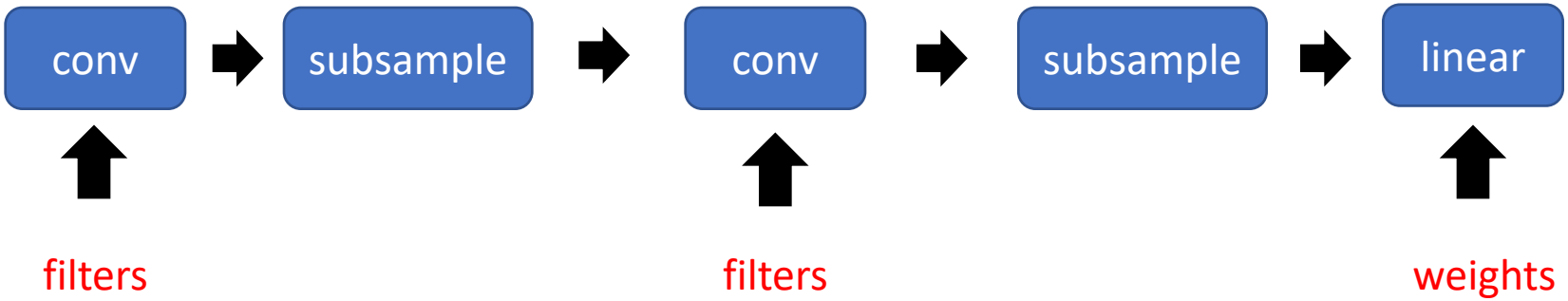
$$\nabla L(h(x;\boldsymbol{\theta}), y)$$

$$z = h(x;\boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}} L(z, y) = \frac{\partial L(z, y)}{\partial z} \frac{\partial z}{\partial \boldsymbol{\theta}}$$

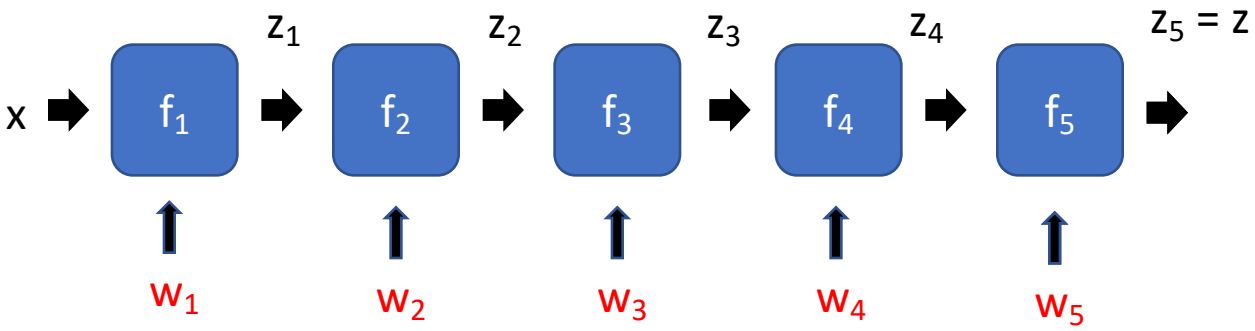
Slide from B. Hariharan

Convolutional networks



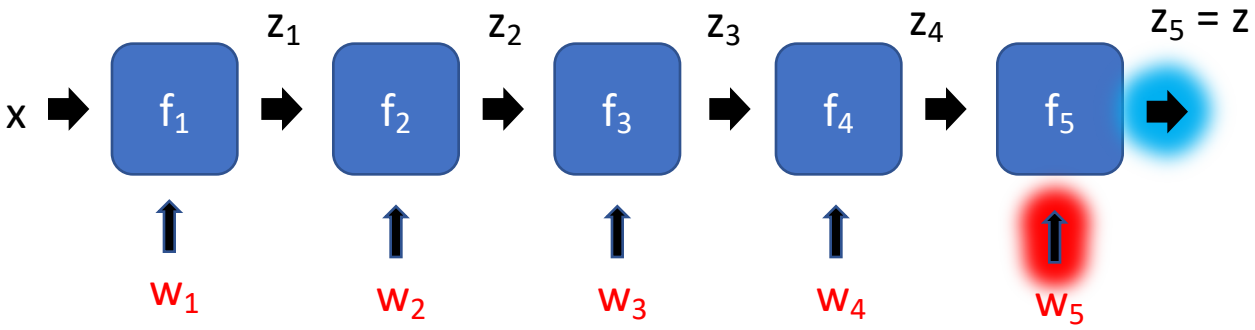
Slide from B. Hariharan

Gradient Computation



Slide from B. Hariharan

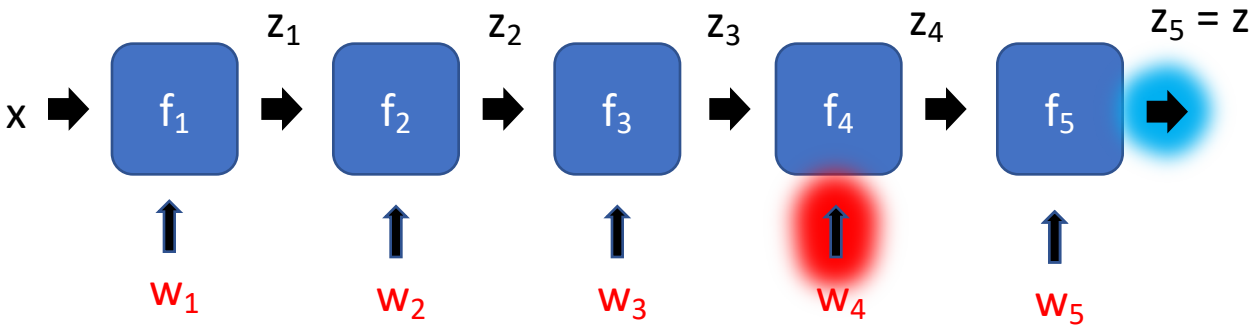
Gradient Computation



$$\frac{\partial z}{\partial w_5}$$

Slide from B. Hariharan

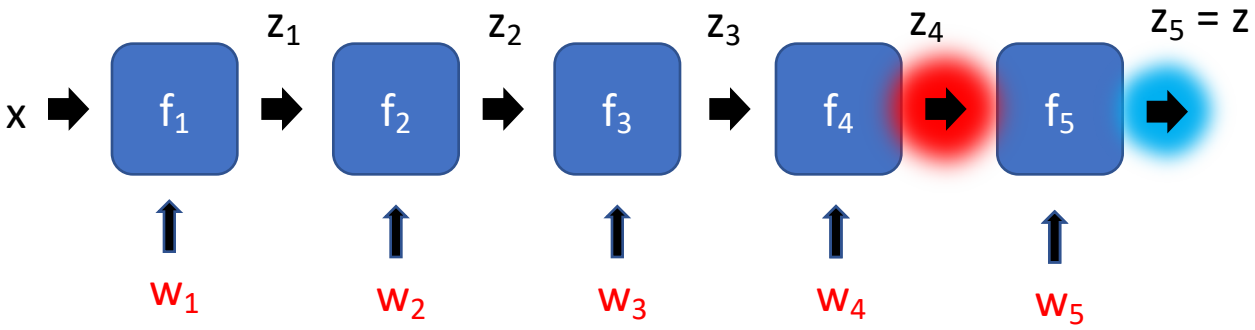
Gradient Computation



$$\frac{\partial z}{\partial w_4}$$

Slide from B. Hariharan

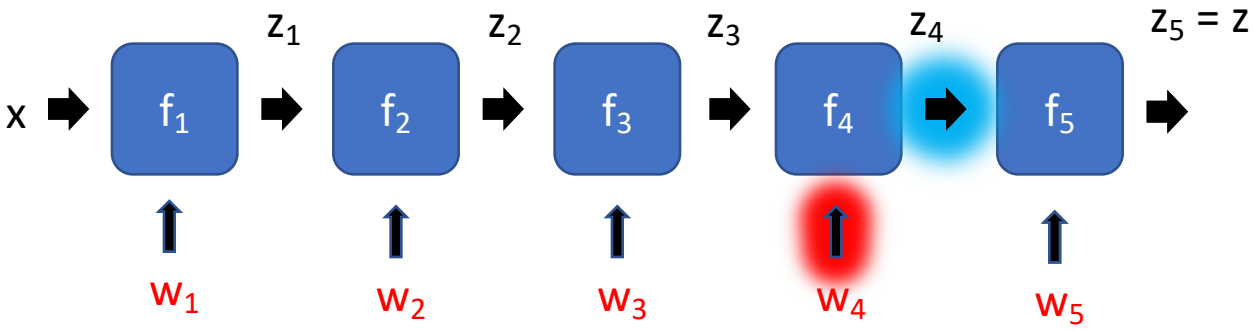
Gradient Computation



$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4}$$

Slide from B. Hariharan

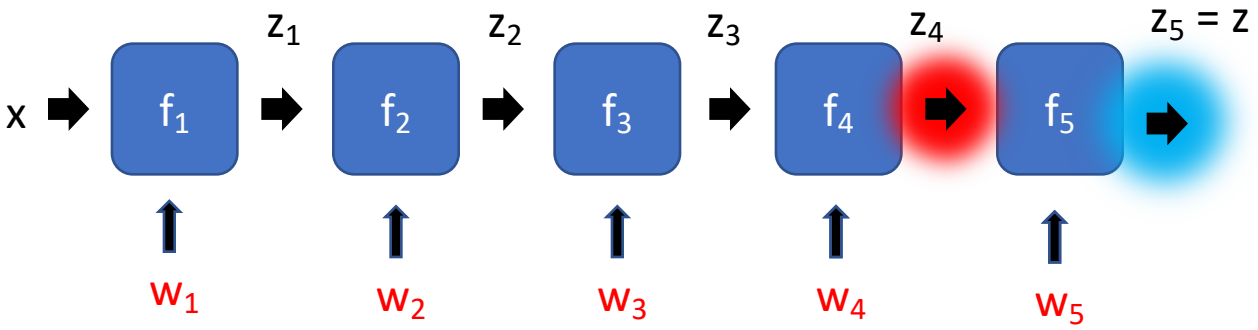
Gradient Computation



$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4}$$

Slide from B. Hariharan

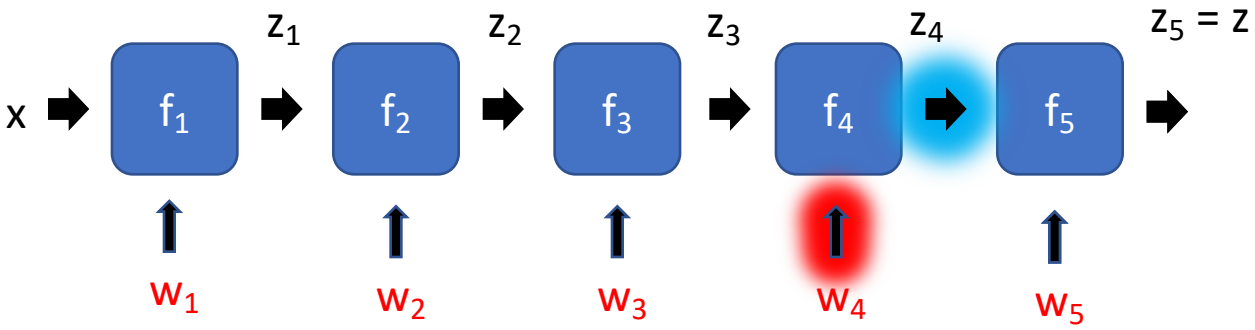
Gradient Computation



$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4} = \frac{\partial f_5(z_4, w_5)}{\partial z_4}\frac{\partial f_4(z_3, w_4)}{\partial w_4}$$

Slide from B. Hariharan

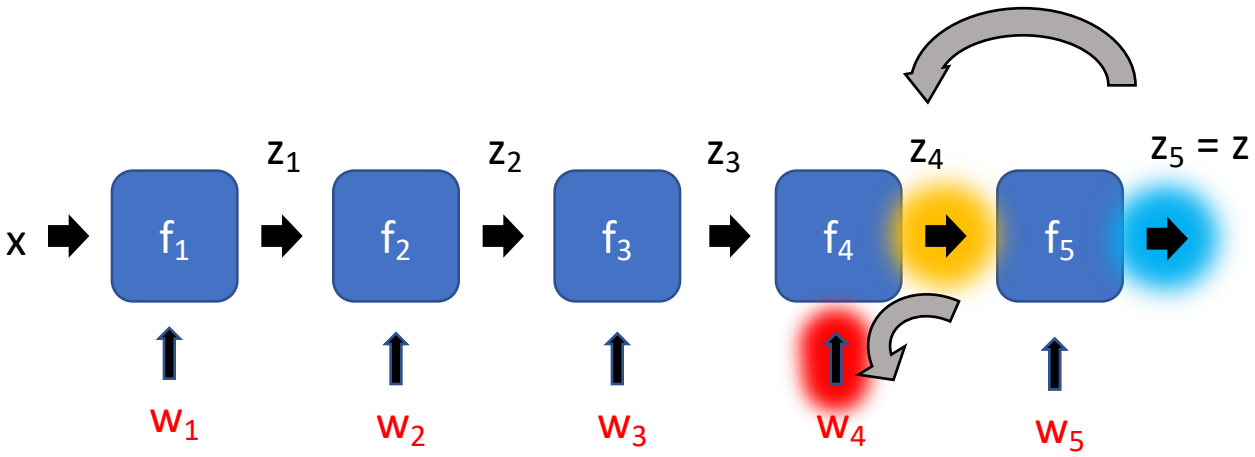
Gradient Computation



$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4} = \frac{\partial f_5(z_4, w_5)}{\partial z_4}\frac{\partial f_4(z_3, w_4)}{\partial w_4}$$

Slide from B. Hariharan

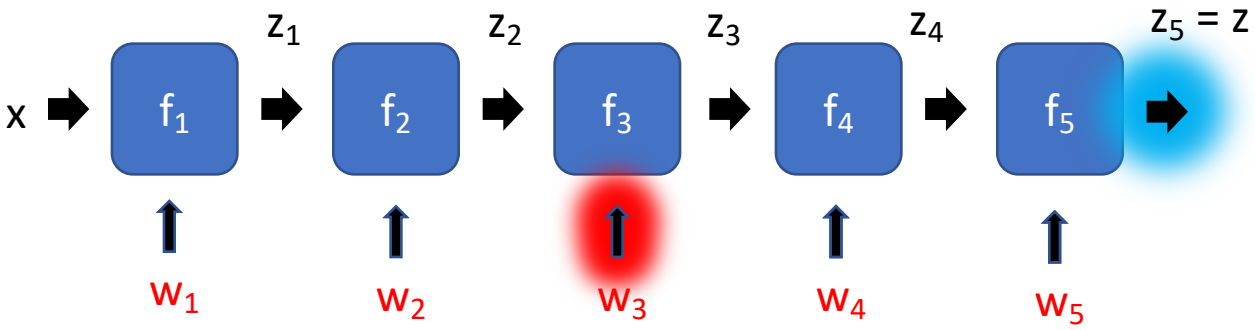
Gradient Computation



$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4} = \frac{\partial f_5(z_4, w_5)}{\partial z_4}\frac{\partial f_4(z_3, w_4)}{\partial w_4}$$

Slide from B. Hariharan

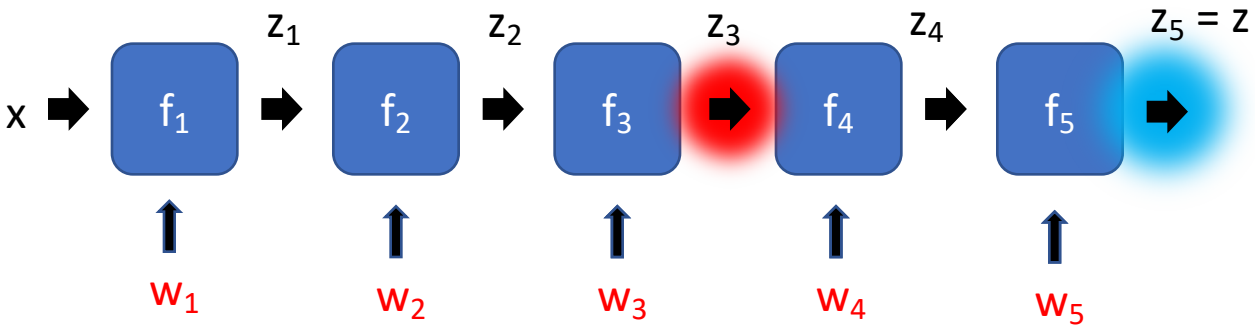
Gradient Computation



$$\frac{\partial z}{\partial w_3}$$

Slide from B. Hariharan

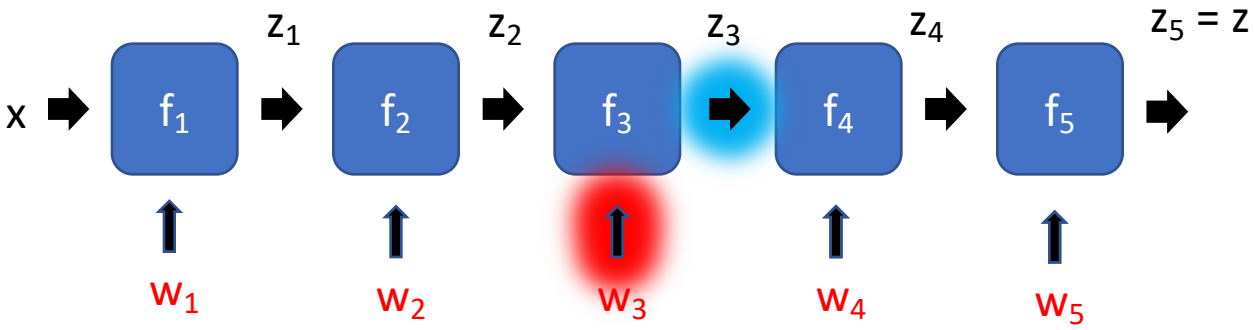
Gradient Computation



$$\frac{\partial z}{\partial w_3} = \frac{\partial z}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

Slide from B. Hariharan

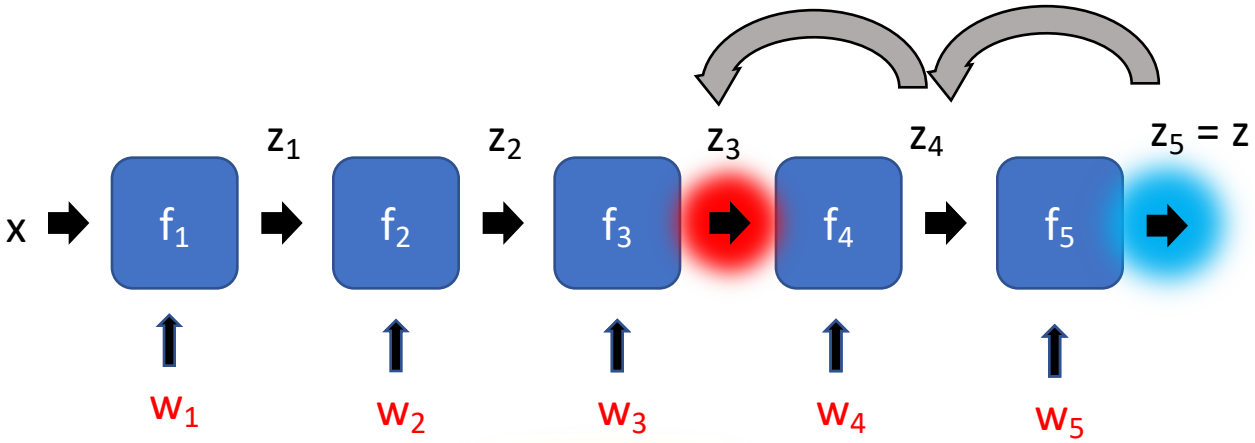
Gradient Computation



$$\frac{\partial z}{\partial w_3} = \frac{\partial z}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

Slide from B. Hariharan

Gradient Computation

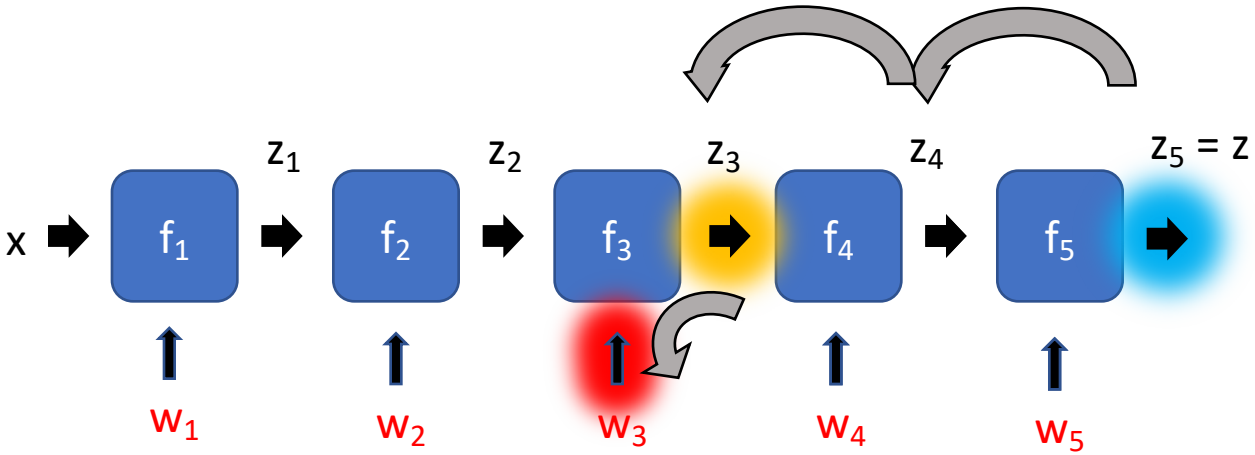


$$\frac{\partial z}{\partial z_3} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial z_3}$$

$$\frac{\partial z}{\partial w_3} = \frac{\partial z}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

Slide from B. Hariharan

Gradient Computation

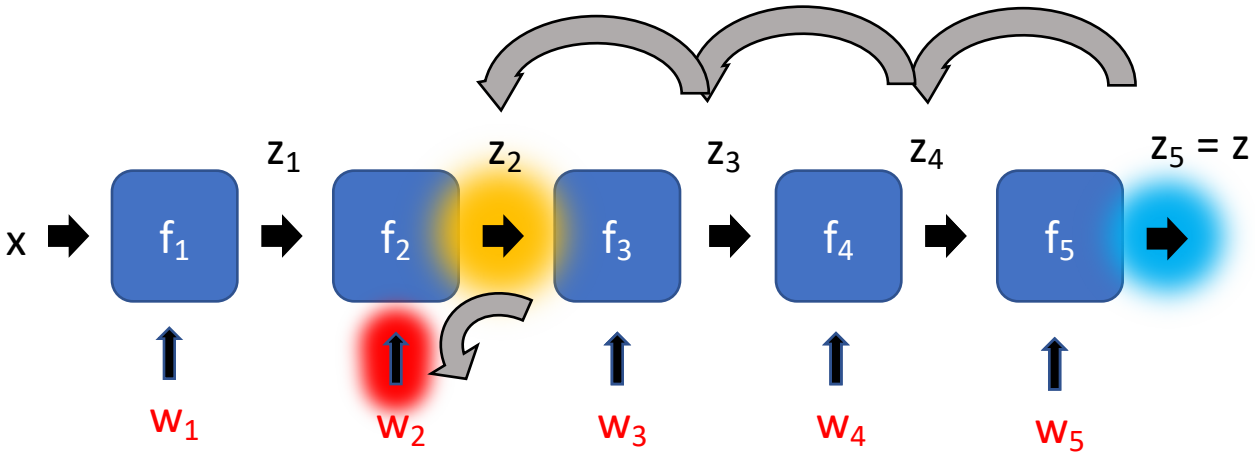


$$\frac{\partial z}{\partial z_3} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial z_3}$$

$$\frac{\partial z}{\partial w_3} = \frac{\partial z}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

Slide from B. Hariharan

Gradient Computation



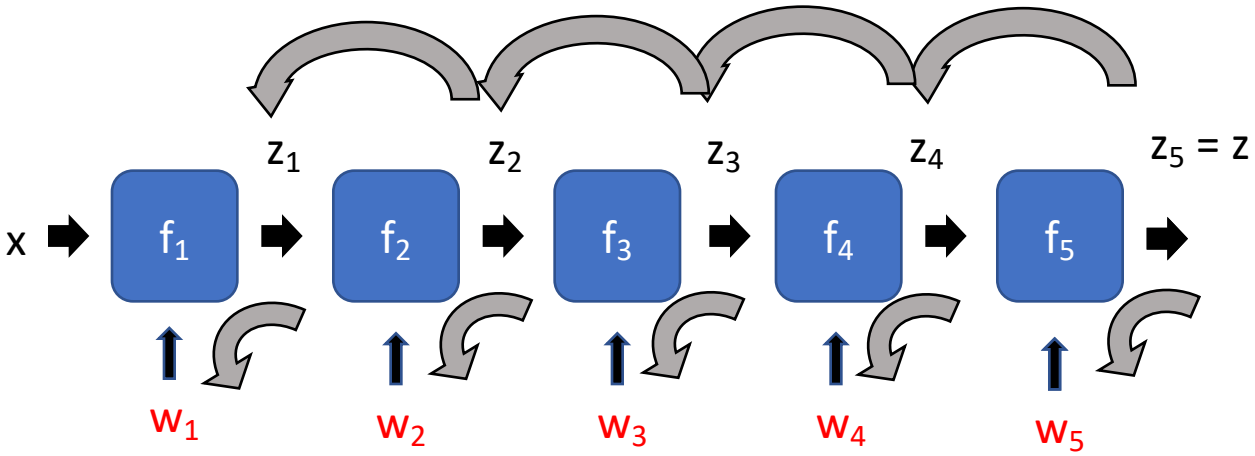
$$\frac{\partial z}{\partial z_2} = \frac{\partial z}{\partial z_3}\frac{\partial z_3}{\partial z_2}$$

$$\frac{\partial z}{\partial w_2} = \frac{\partial z}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$

Recurrence going backward!!

Slide from B. Hariharan

Gradient Computation



Back-Propagation

Slide from B. Hariharan

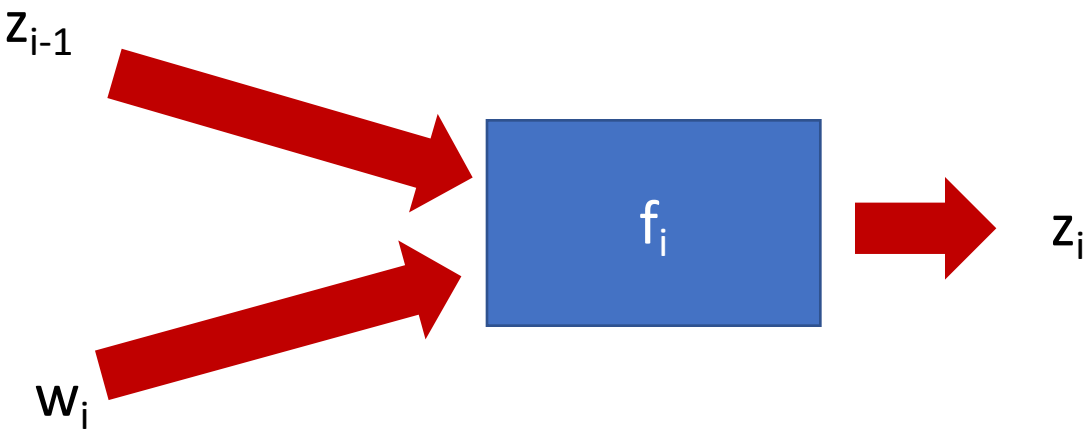
Backpropagation for a sequence of functions

- Each “function” has a “forward” and “backward” module
- Forward module for f_i
  - takes z_{i-1} and weight w_i as input
  - produces z_i as output
- Backward module for f_i
  - takes $g(z_i)$ as input
  - produces $g(z_{i-1})$ and $g(w_i)$ as output

$$g(z_{i-1}) = g(z_i)\frac{\partial z_i}{\partial z_{i-1}} \qquad g(w_i) = g(z_i)\frac{\partial z_i}{\partial w_i}$$

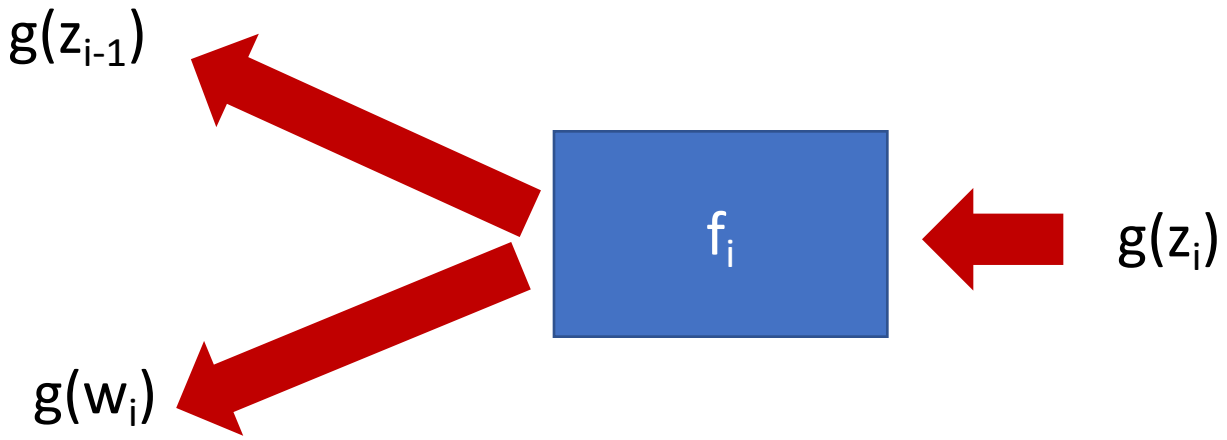
Slide from B. Hariharan

Backpropagation for a sequence of functions



Slide from B. Hariharan

Backpropagation for a sequence of functions



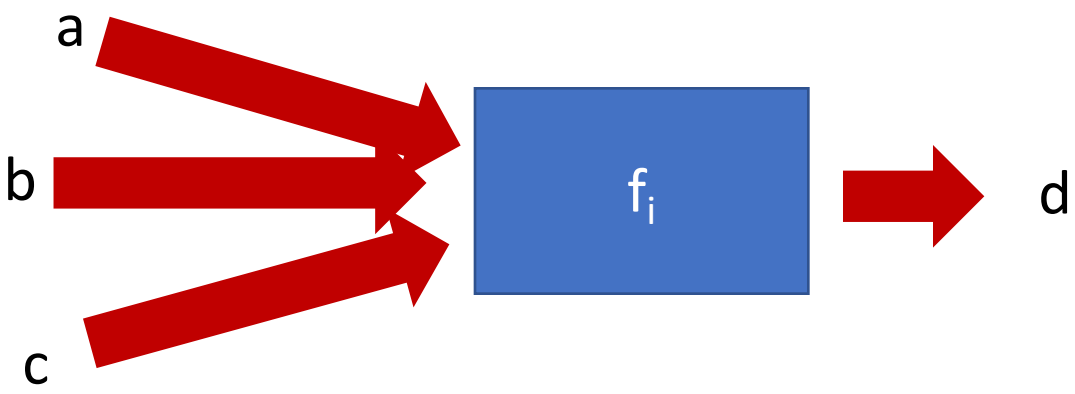
Slide from B. Hariharan

Computation graph - Functions

- Each node implements two functions
  - A “forward”
    - Computes output given input
  - A “backward”
    - Computes derivative of z w.r.t input, given derivative of z w.r.t output

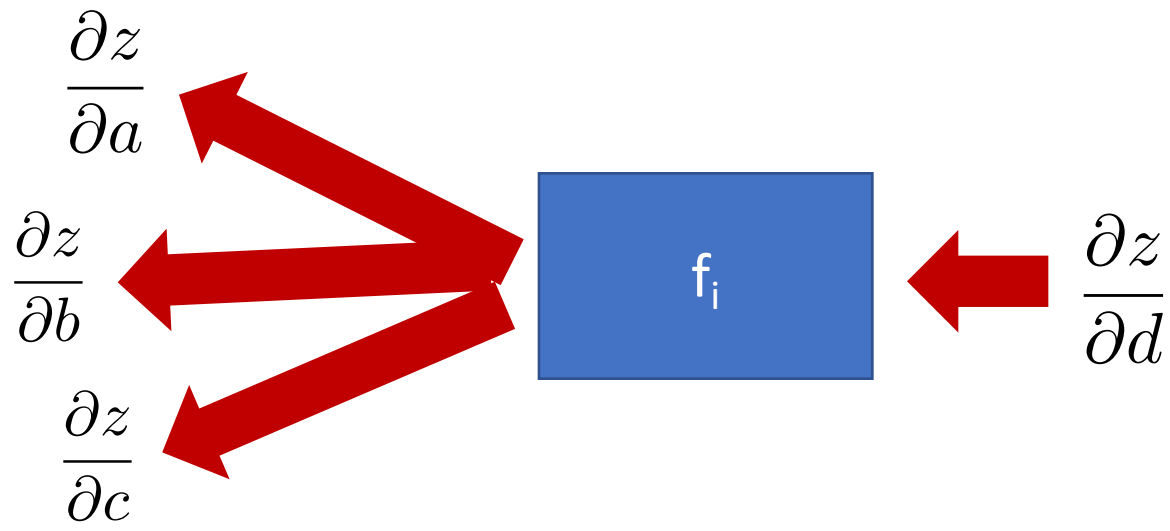
Slide from B. Hariharan

Computation graphs



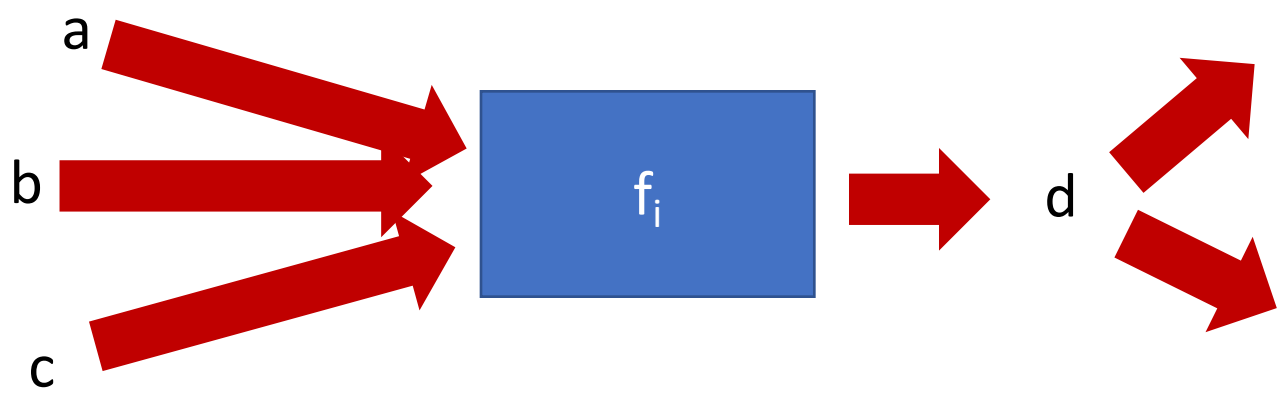
Slide from B. Hariharan

Computation graphs



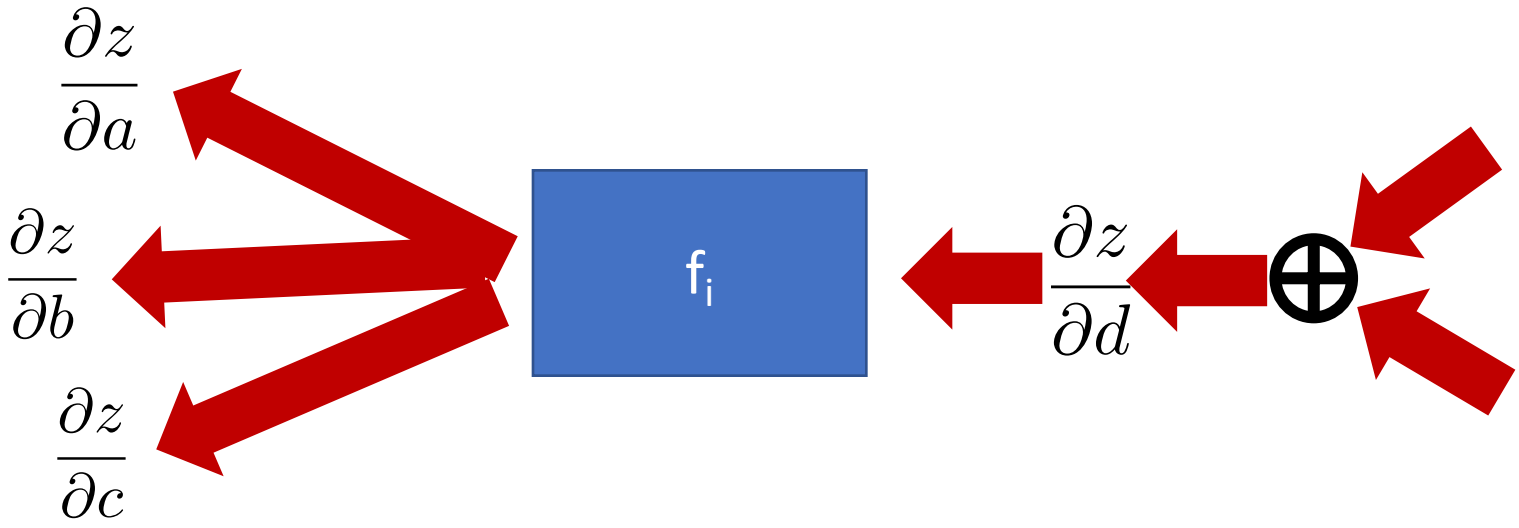
Slide from B. Hariharan

Computation graphs



Slide from B. Hariharan

Computation graphs



Slide from B. Hariharan

Neural network frameworks

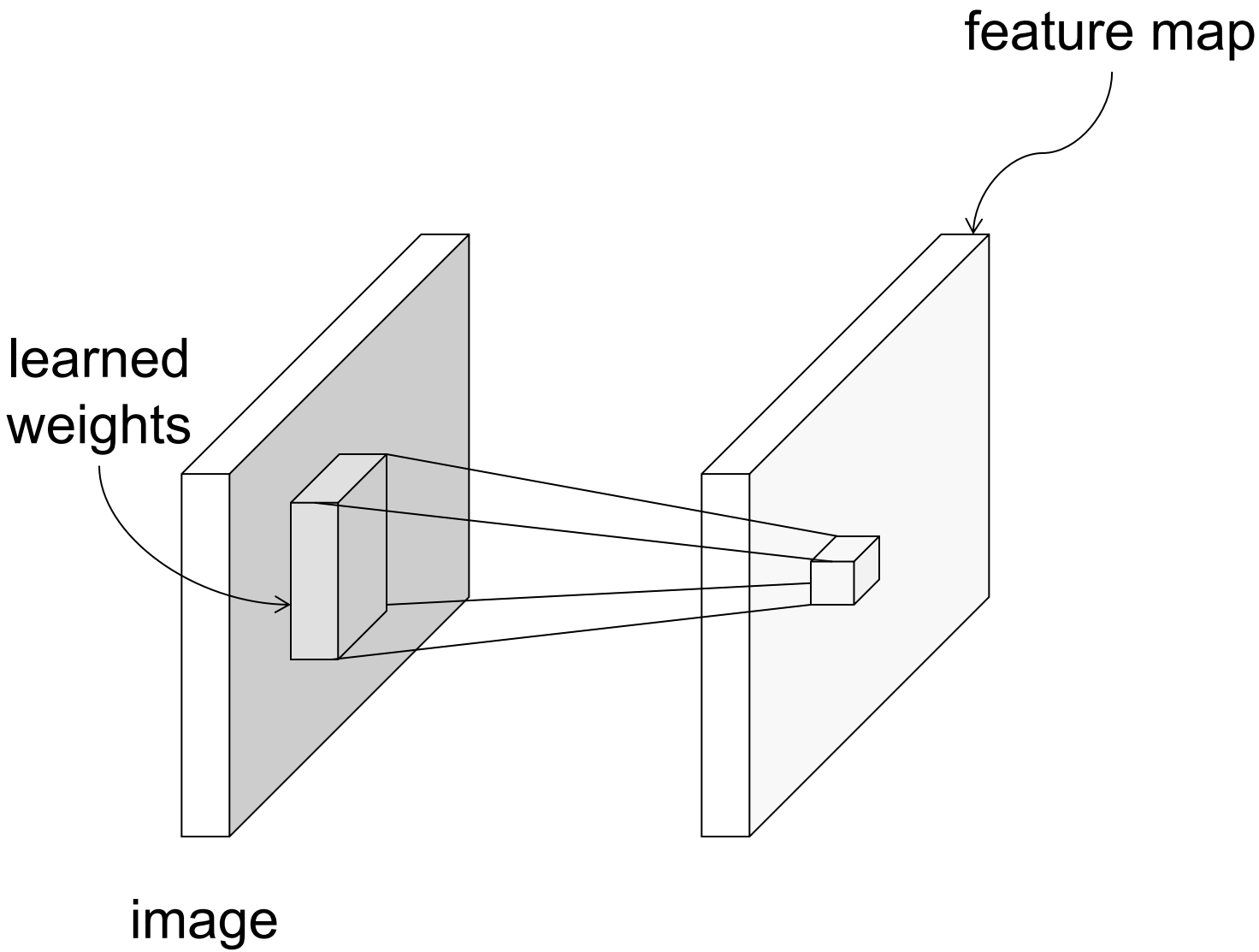
Slide from B. Hariharan

Building blocks

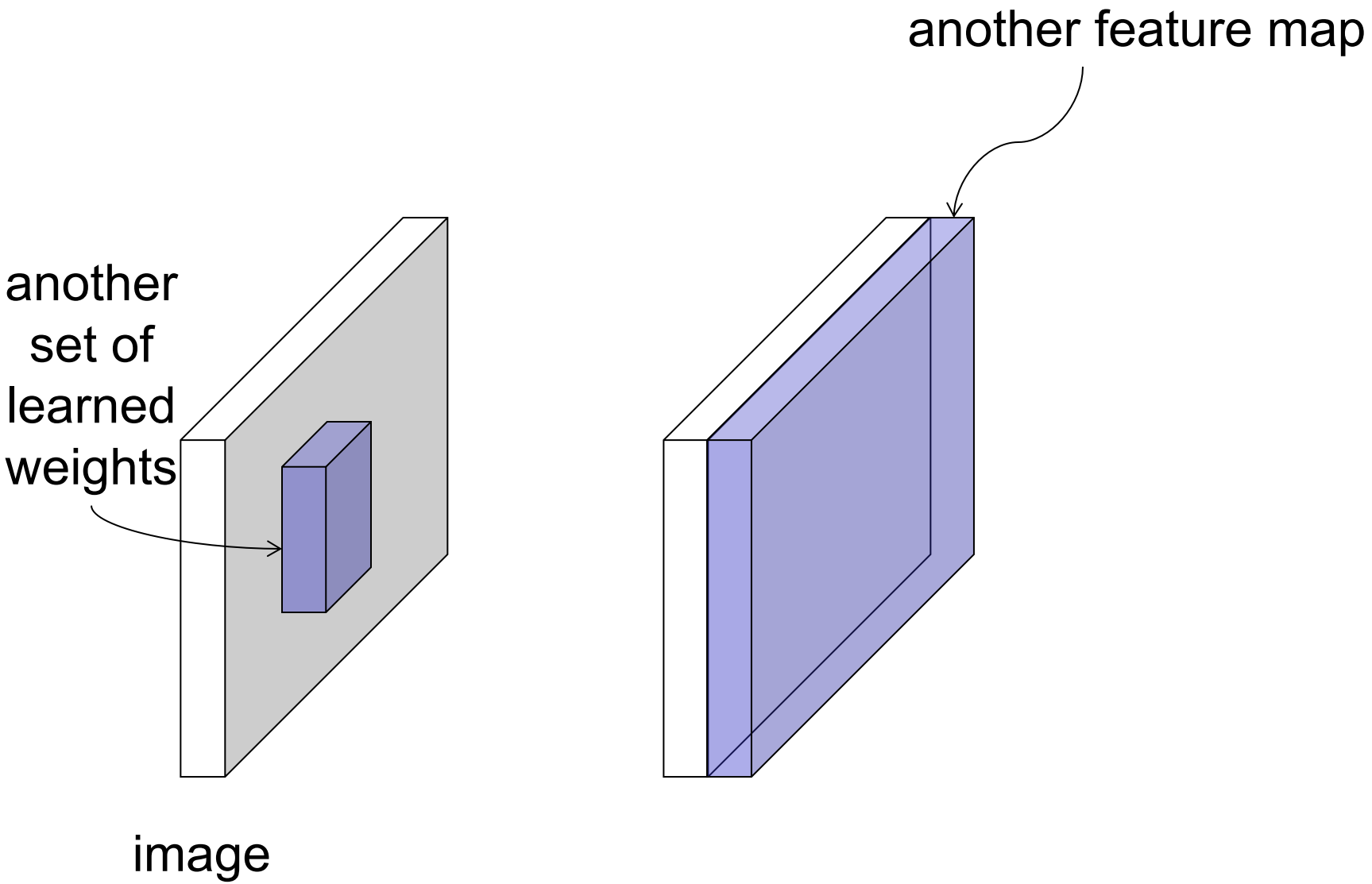
- Convolutional Layers
- Non-linearities
- Pooling
- Fully-connected Layers
- Attention Layers

Rationale?

Neural networks for images

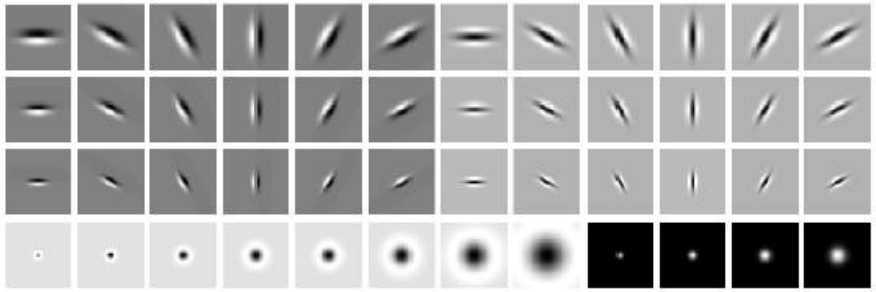


Neural networks for images



Convolution as feature extraction

bank of K filters

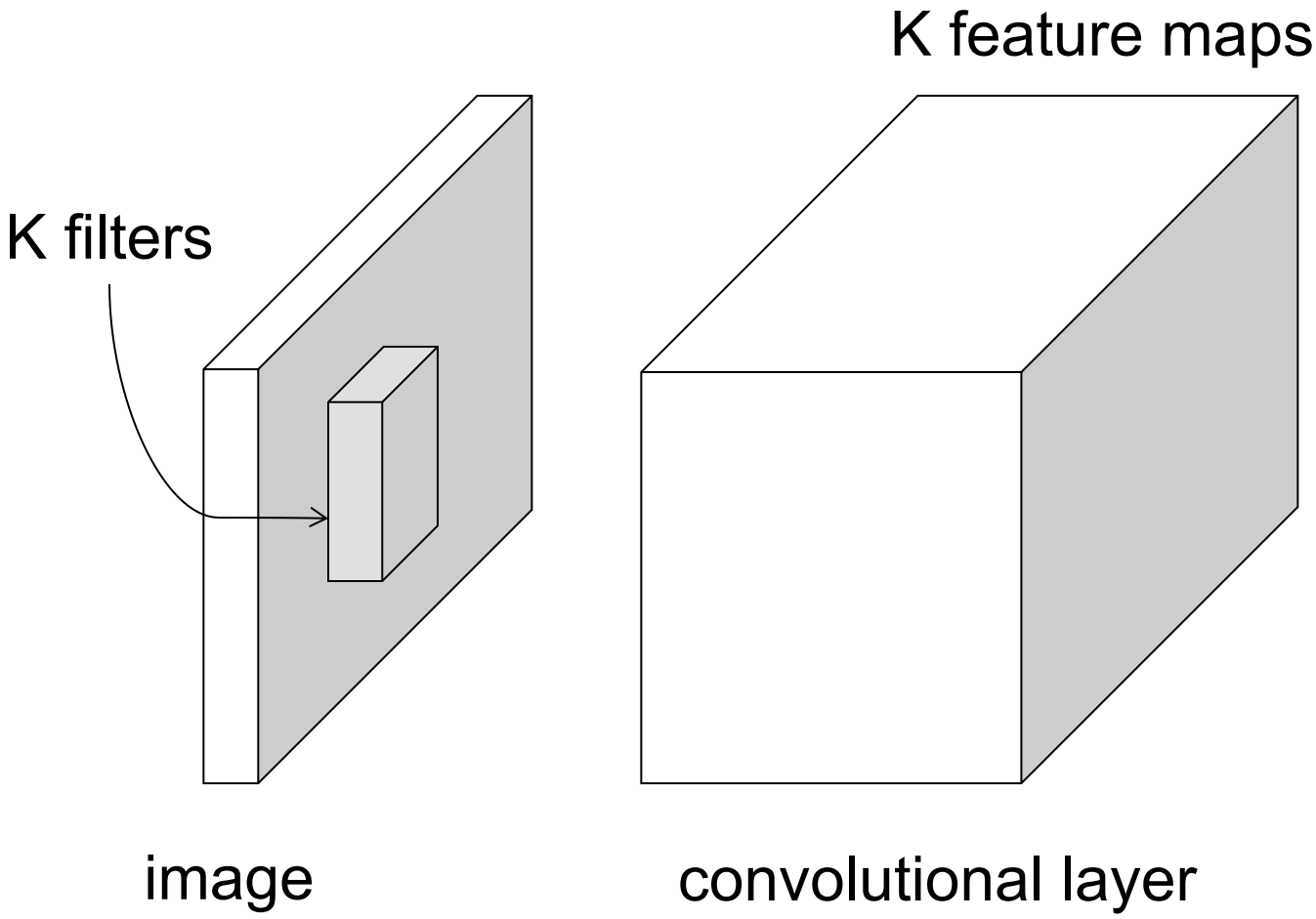


K feature maps

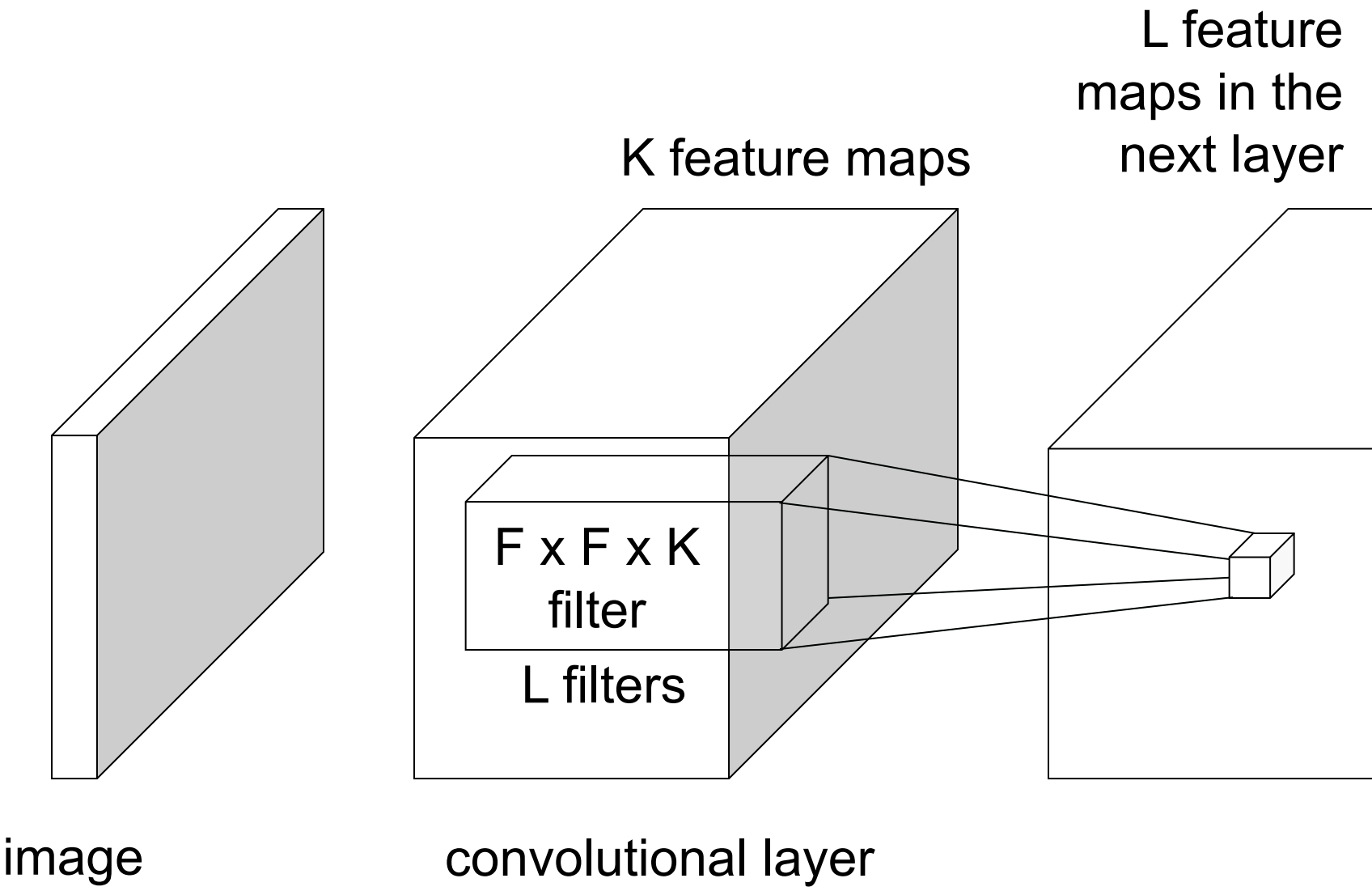
image

feature map

Convolutional layer



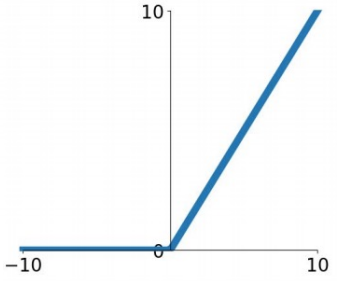
Convolutional layer



Non-Linearities

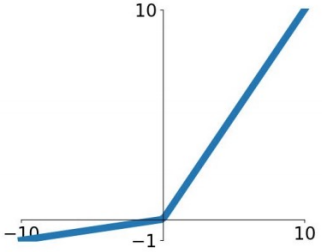
ReLU

$$\max(0, x)$$



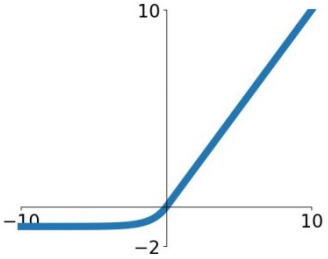
Leaky ReLU

$$\max(0.1x, x)$$



ELU

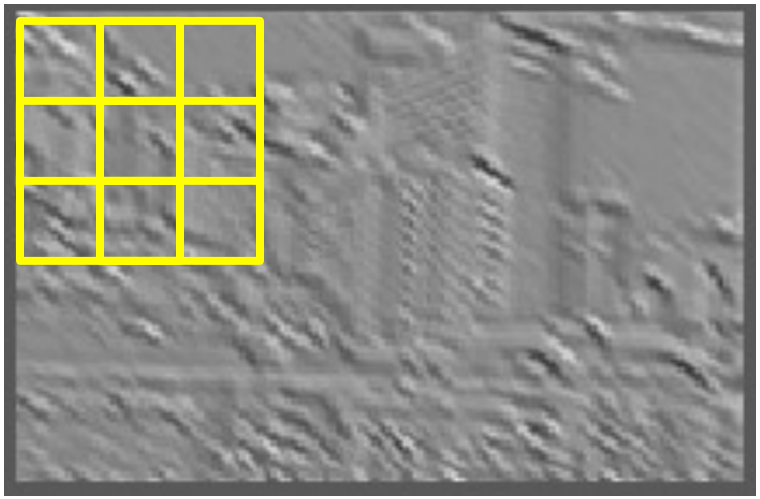
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



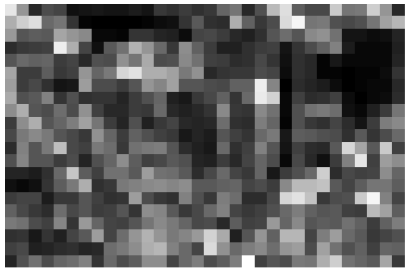
Source: R. Fergus, Y. LeCun

Source: Stanford 231n

Pooling Layers

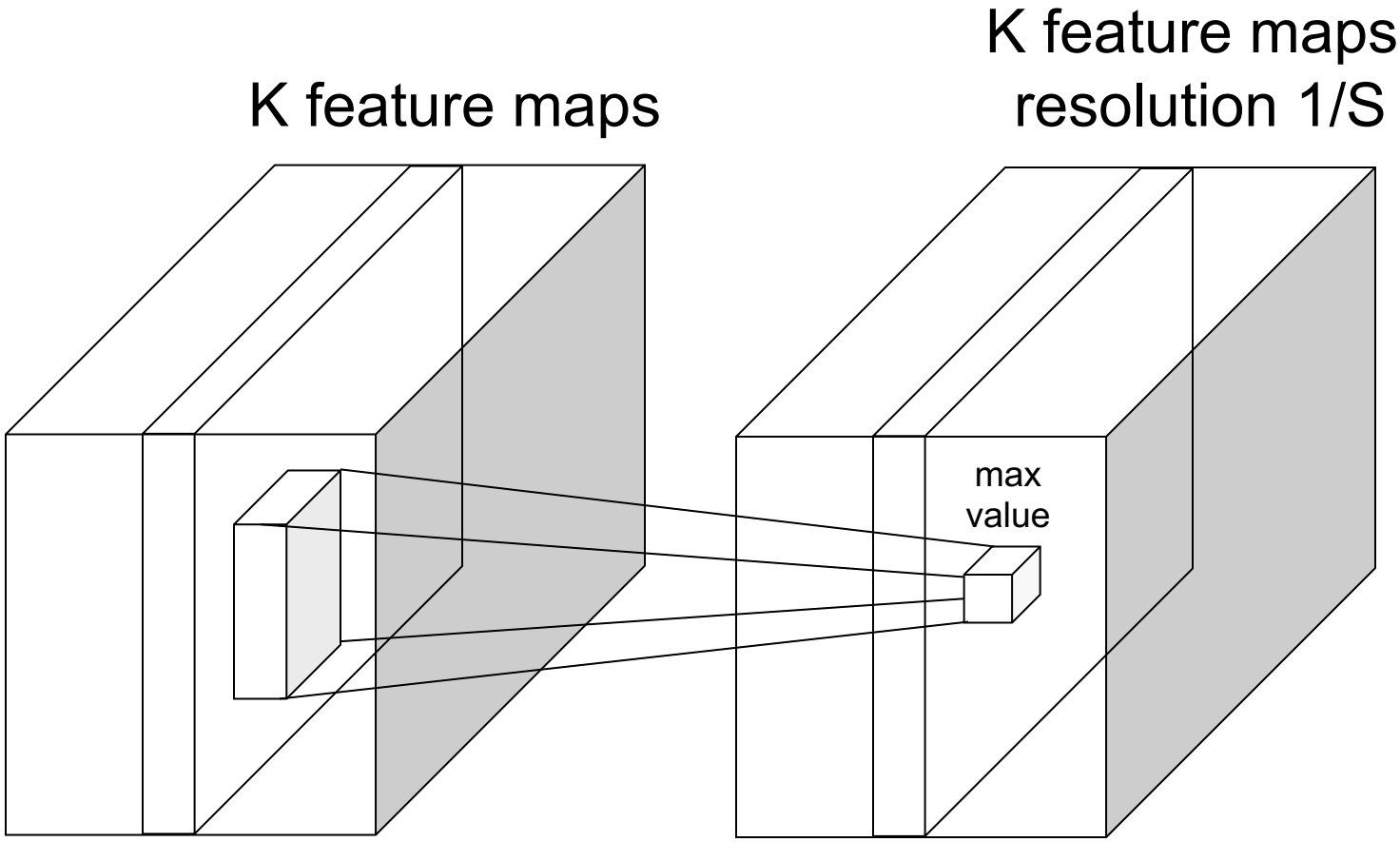


Max
(or Avg)



Source: R. Fergus, Y. LeCun

Pooling Layers

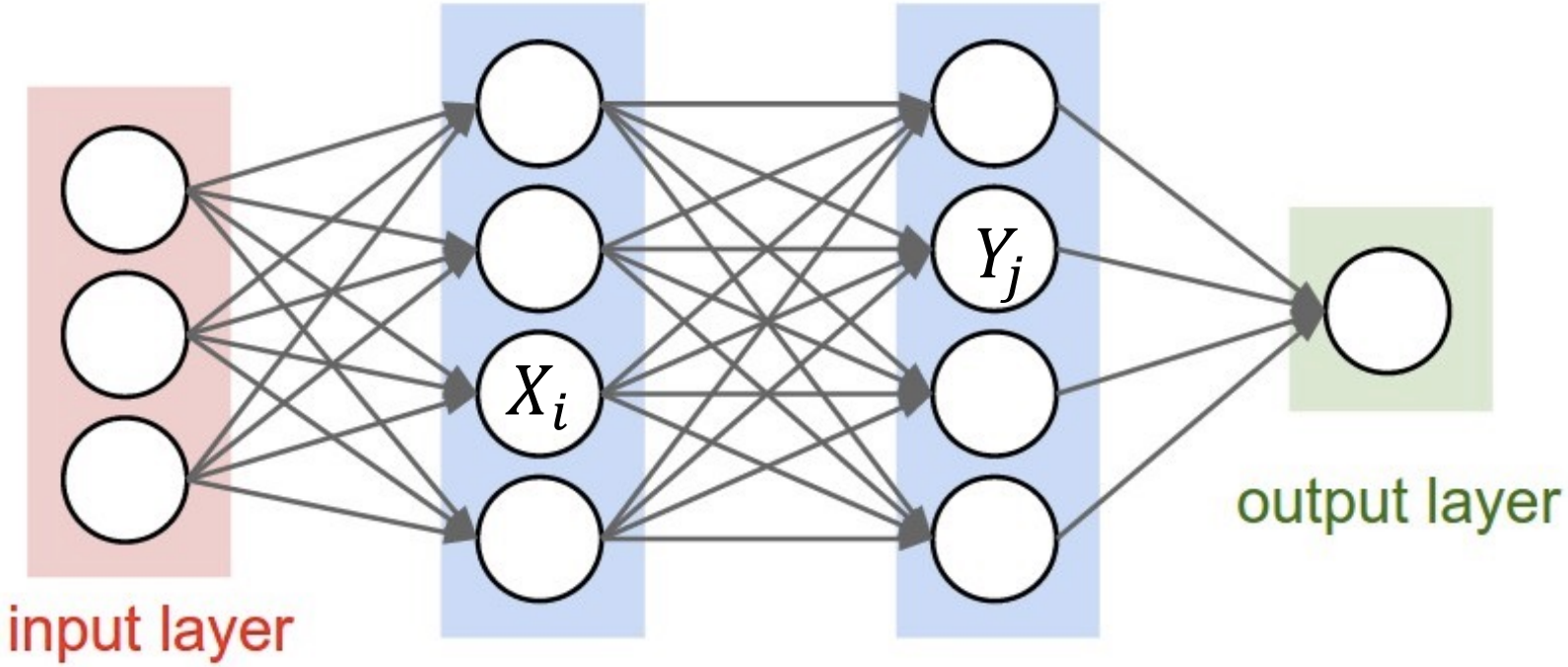


F x F pooling filter,
stride S

Usually: F=2 or 3, S=2

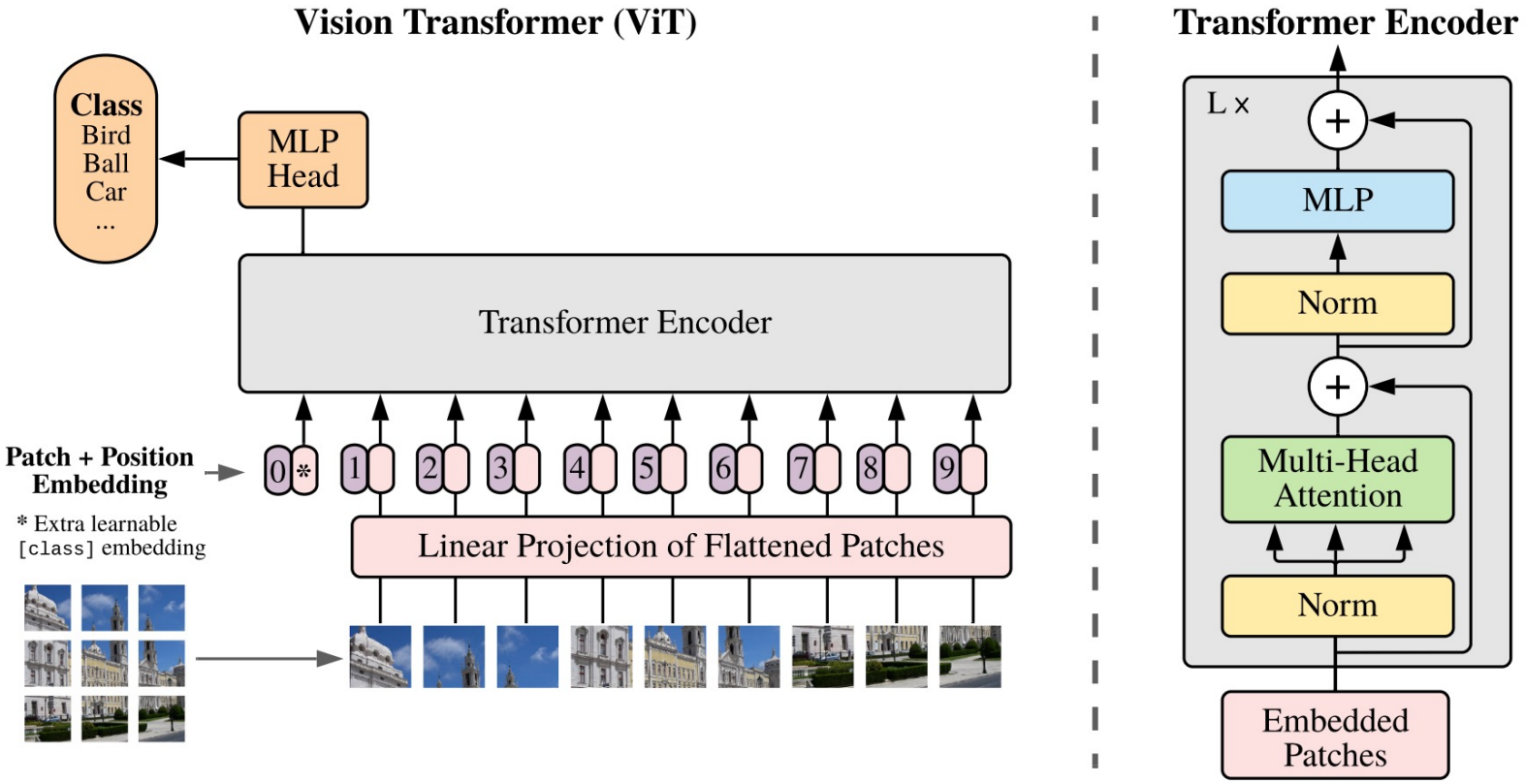
Fully-connected Layers

$$X \qquad Y = W^t X$$



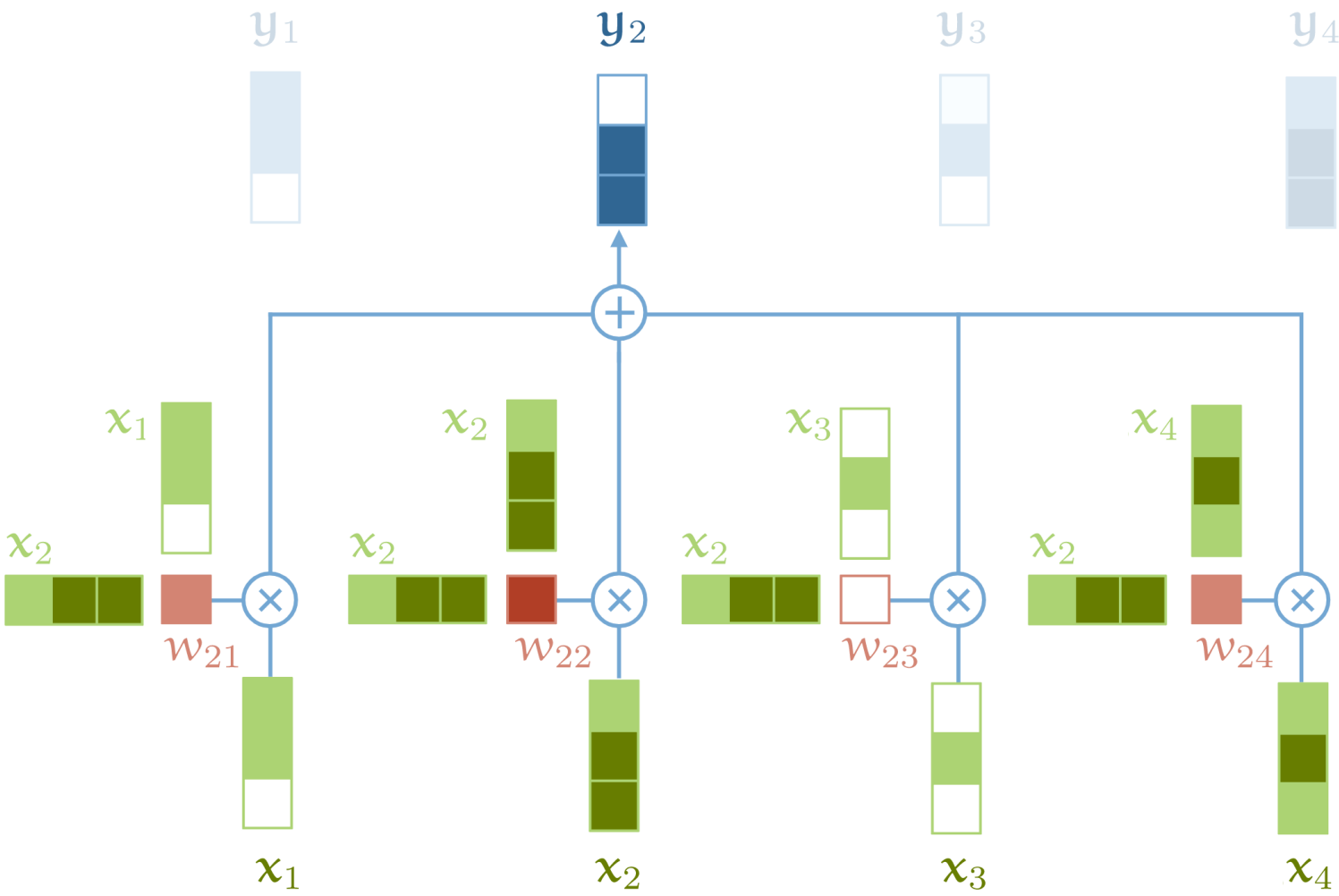
$$Y_j = \sum_i W_{ij} X_i$$

Attention (Vision Transformers)



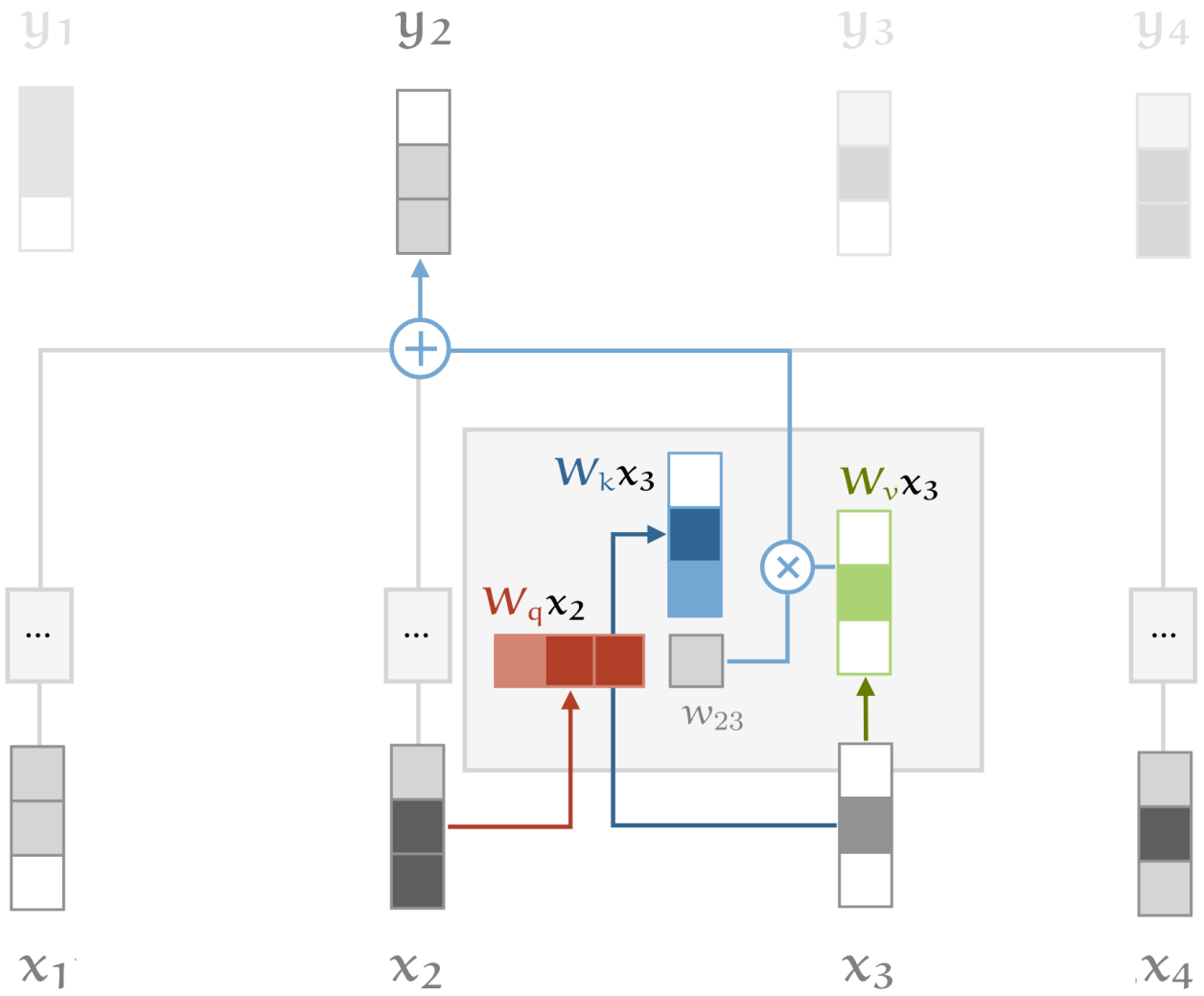
A. Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Attention



Source: http://peterbloem.nl/blog/transformers
See also: Attention is all you need

Attention (with key, query and value)



Source: http://peterbloem.nl/blog/transformers
See also: Attention is all you need

Representing Positions

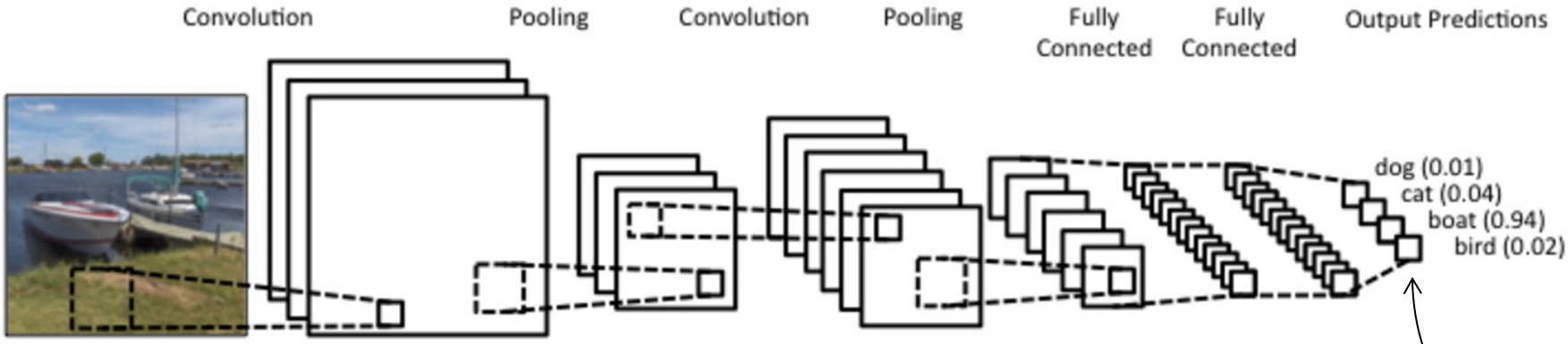
- Positional Embeddings
  - Learn embeddings for different positions
- Positional Encodings
  - Explicitly encode positions using sin, cos terms

See also: Attention is all you need

Components of a CNN architecture

- Convolutional Layers
- Non-linearities
- Pooling
- Fully-connected Layers
- Attention

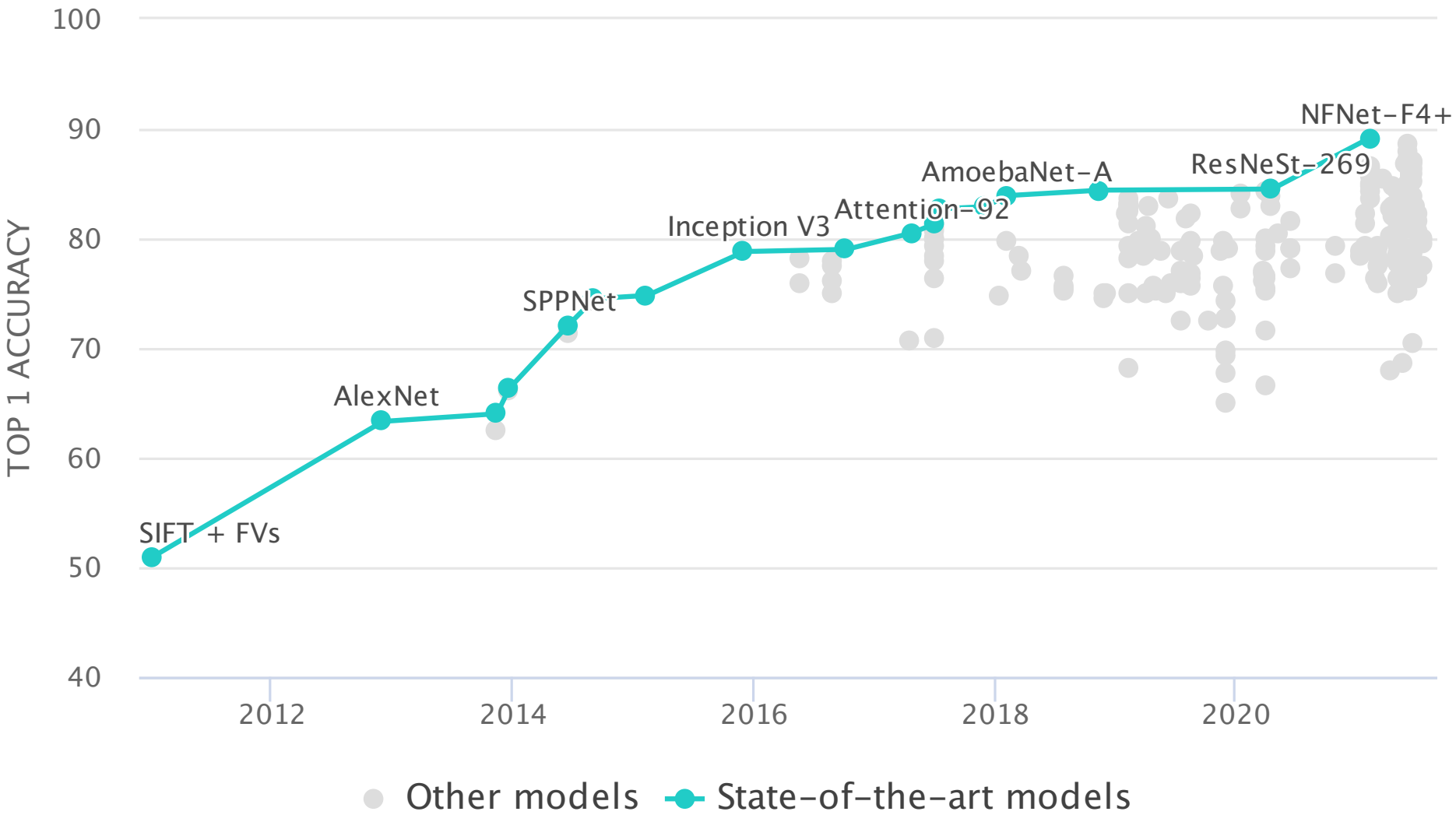
Putting it together



Softmax layer:

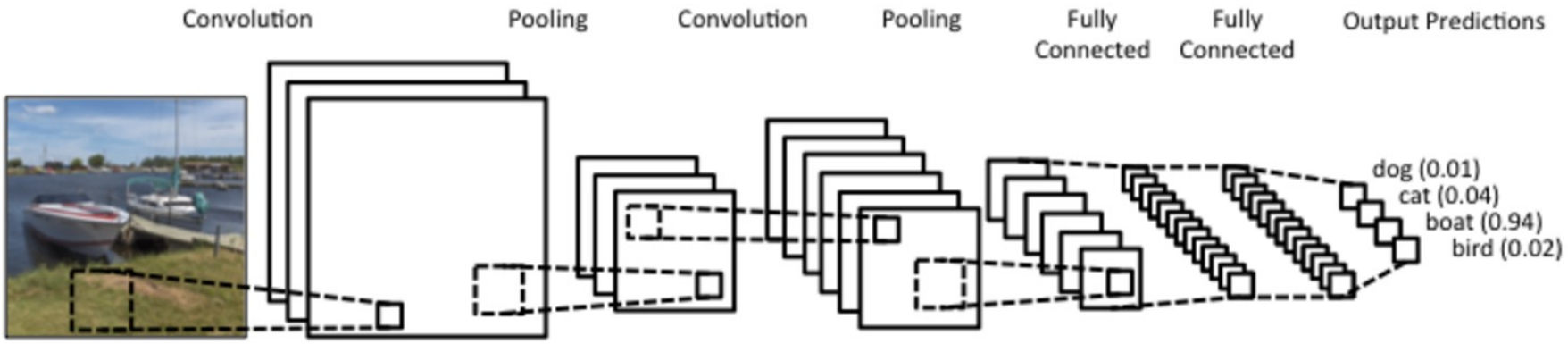
$$P(c \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_c \cdot \mathbf{x})}{\sum_{k=1}^{C} \exp(\mathbf{w}_k \cdot \mathbf{x})}$$

Many different models



Source: https://paperswithcode.com/

Learned Representations are Useful in General



1. Features extracted from CNNs trained on ImageNet were effective for many CV tasks.
2. Furthermore, learned network weights serve as an excellent starting point for other tasks.

J. Donahue, Y. Jia et al. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. ICML 2014

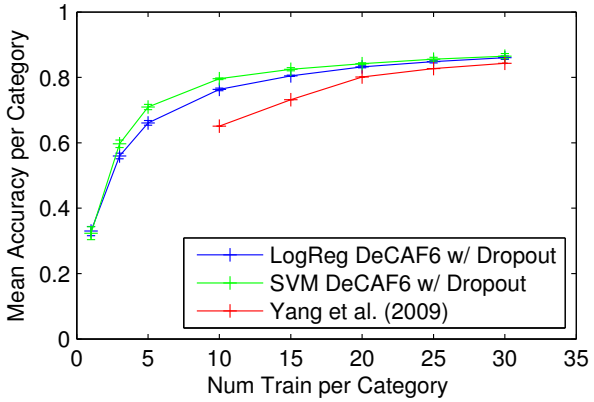
How to use a trained network for a new task?

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

Caltech 101

	Amazon → Webcam		
	SURF	$DeCAF_6$	$DeCAF_7$
Logistic Reg. (S)	9.63 ± 1.4	48.58 ± 1.3	53.56 ± 1.5
SVM (S)	11.05 ± 2.3	52.22 ± 1.7	53.90 ± 2.2
Logistic Reg. (T)	24.33 ± 2.1	72.56 ± 2.1	74.19 ± 2.8
SVM (T)	51.05 ± 2.0	78.26 ± 2.6	78.72 ± 2.3
Logistic Reg. (ST)	19.89 ± 1.7	75.30 ± 2.0	76.32 ± 2.0
SVM (ST)	23.19 ± 3.5	80.66 ± 2.3	79.12 ± 2.1
Daume III (2007)	40.26 ± 1.1	$\mathbf{82.14 \pm 1.9}$	81.65 ± 2.4
Hoffman et al. (2013)	37.66 ± 2.2	80.06 ± 2.7	80.37 ± 2.0
Gong et al. (2012)	39.80 ± 2.3	75.21 ± 1.2	77.55 ± 1.9
Chopra et al. (2013)		58.85	

Domain Adaptation



Caltech 101

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

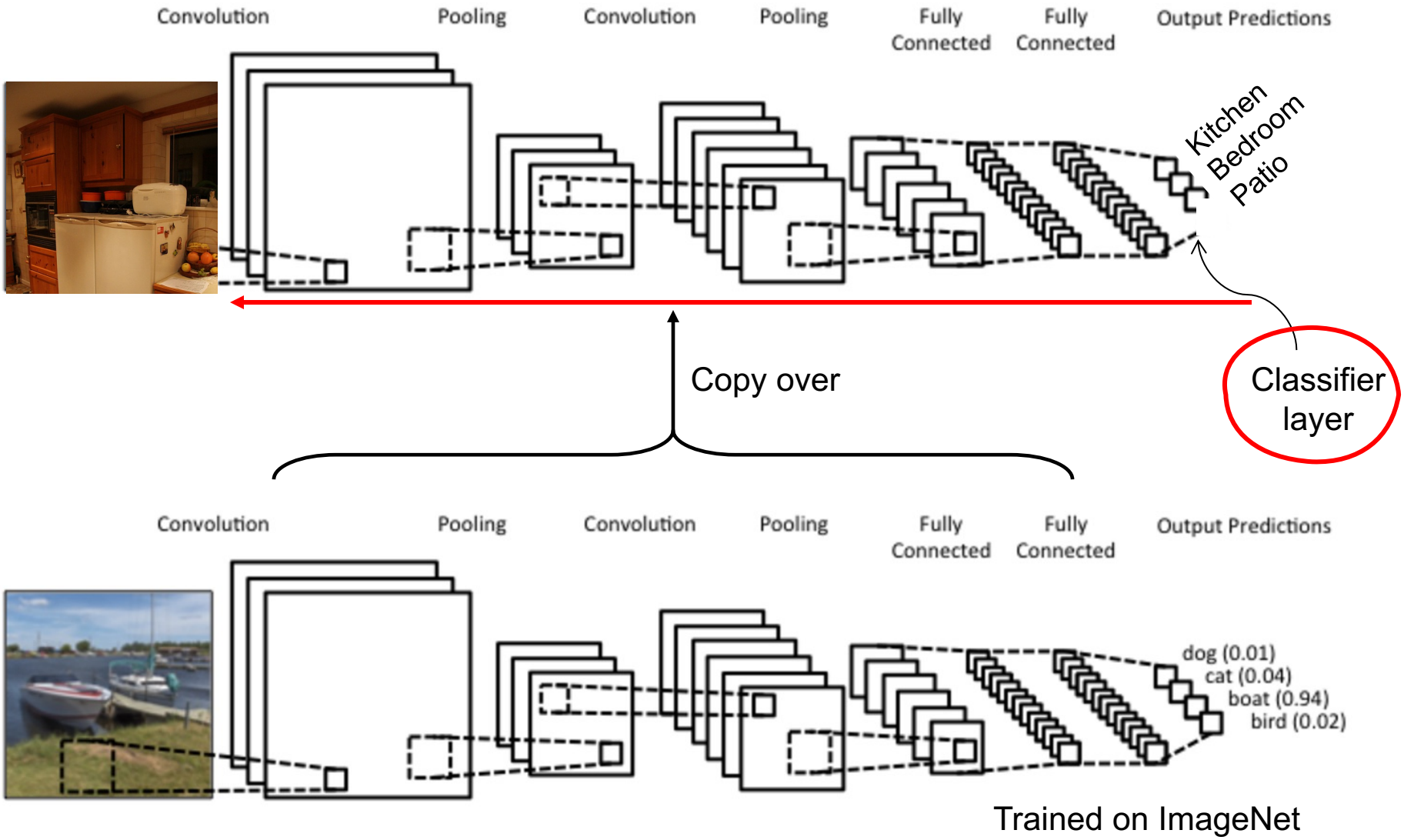
Fine-grained Classification

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

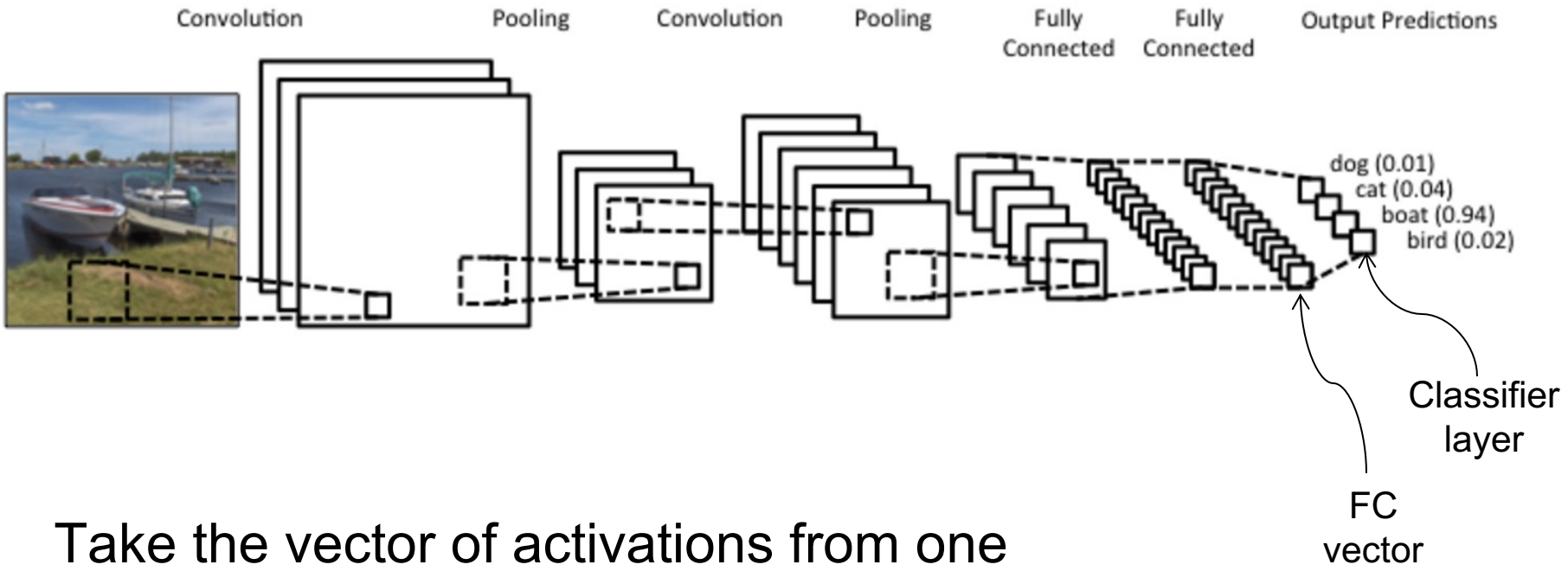
Scene Classification

J. Donahue, Y. Jia et al. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. ICML 2014

How to use a trained network for a new task?

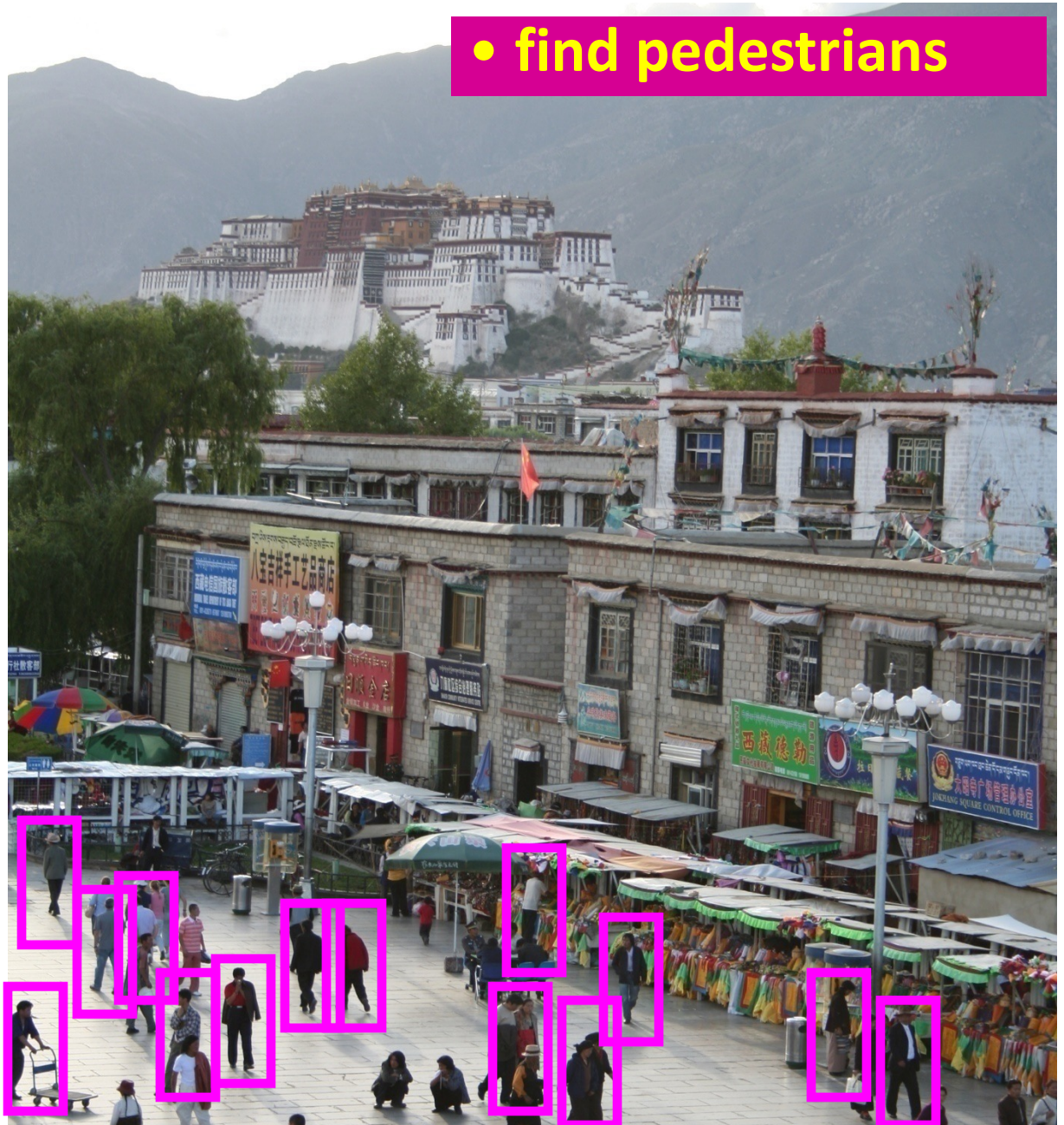


How to use a trained network for a new task?



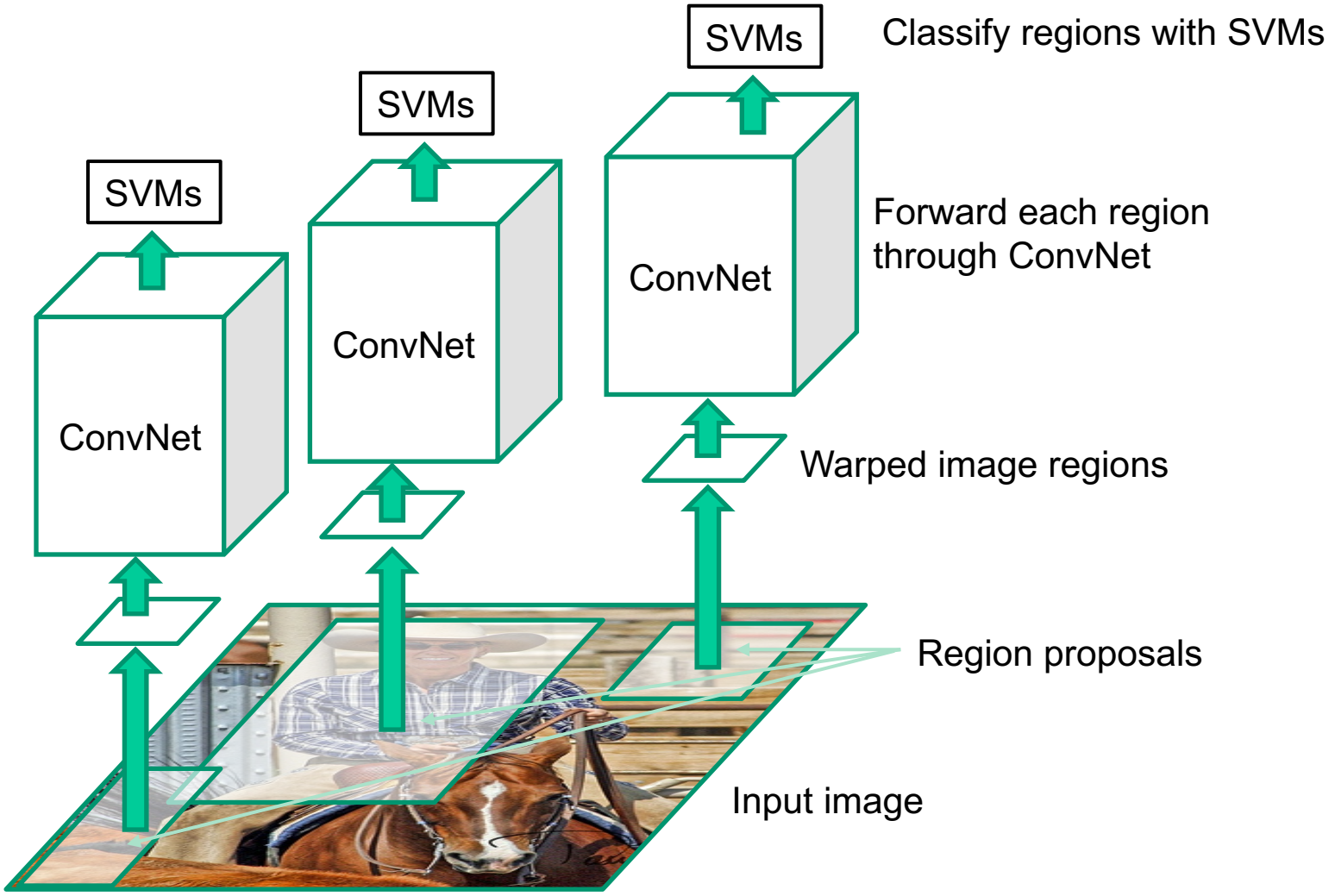
- Take the vector of activations from one of the fully connected (FC) layers and treat it as an off-the-shelf feature
  - Train a new classifier layer on top of the FC layer
- *Fine-tune* the whole network

Other tasks: Object detection



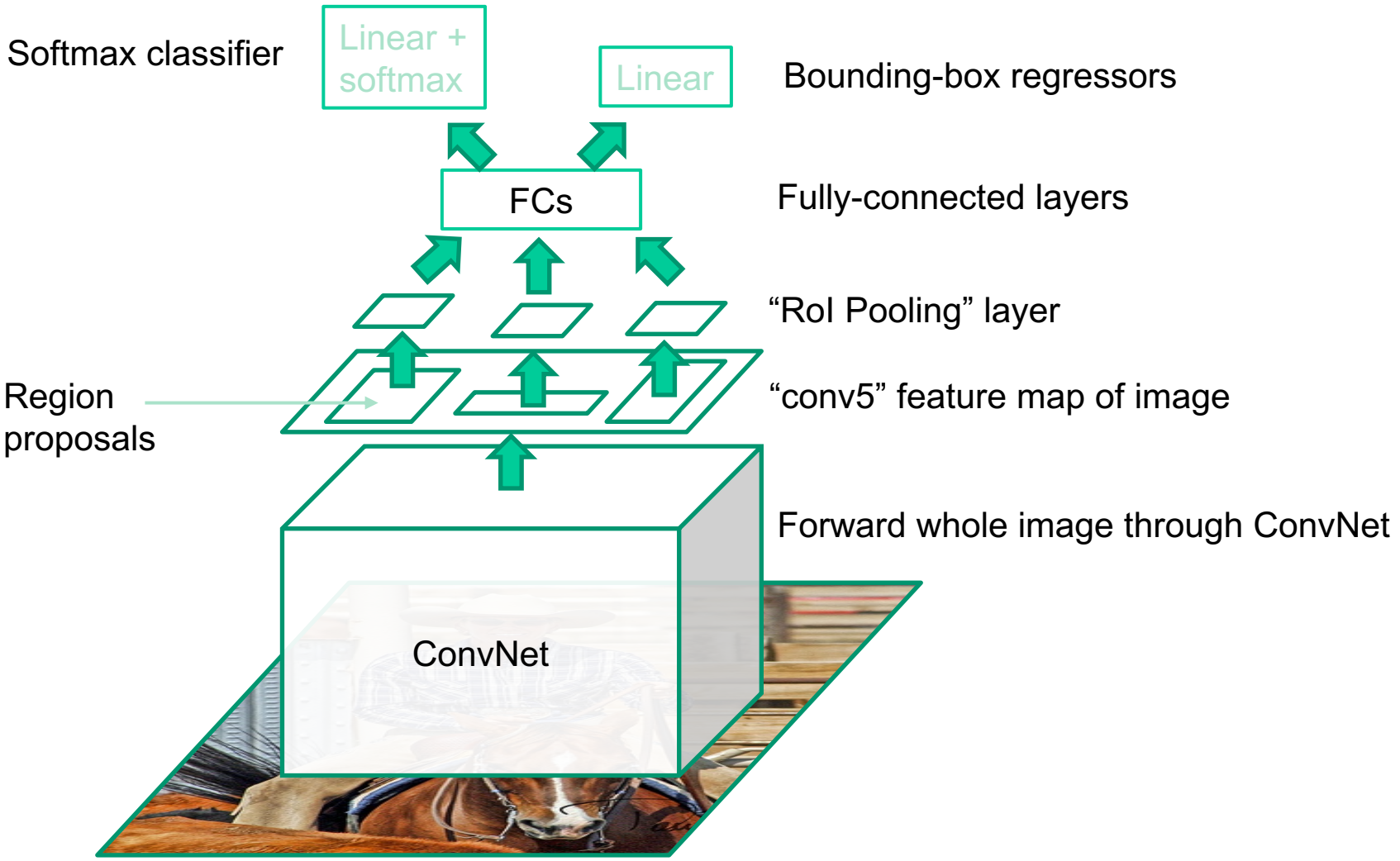
Adapted from Fei-Fei Li

R-CNN



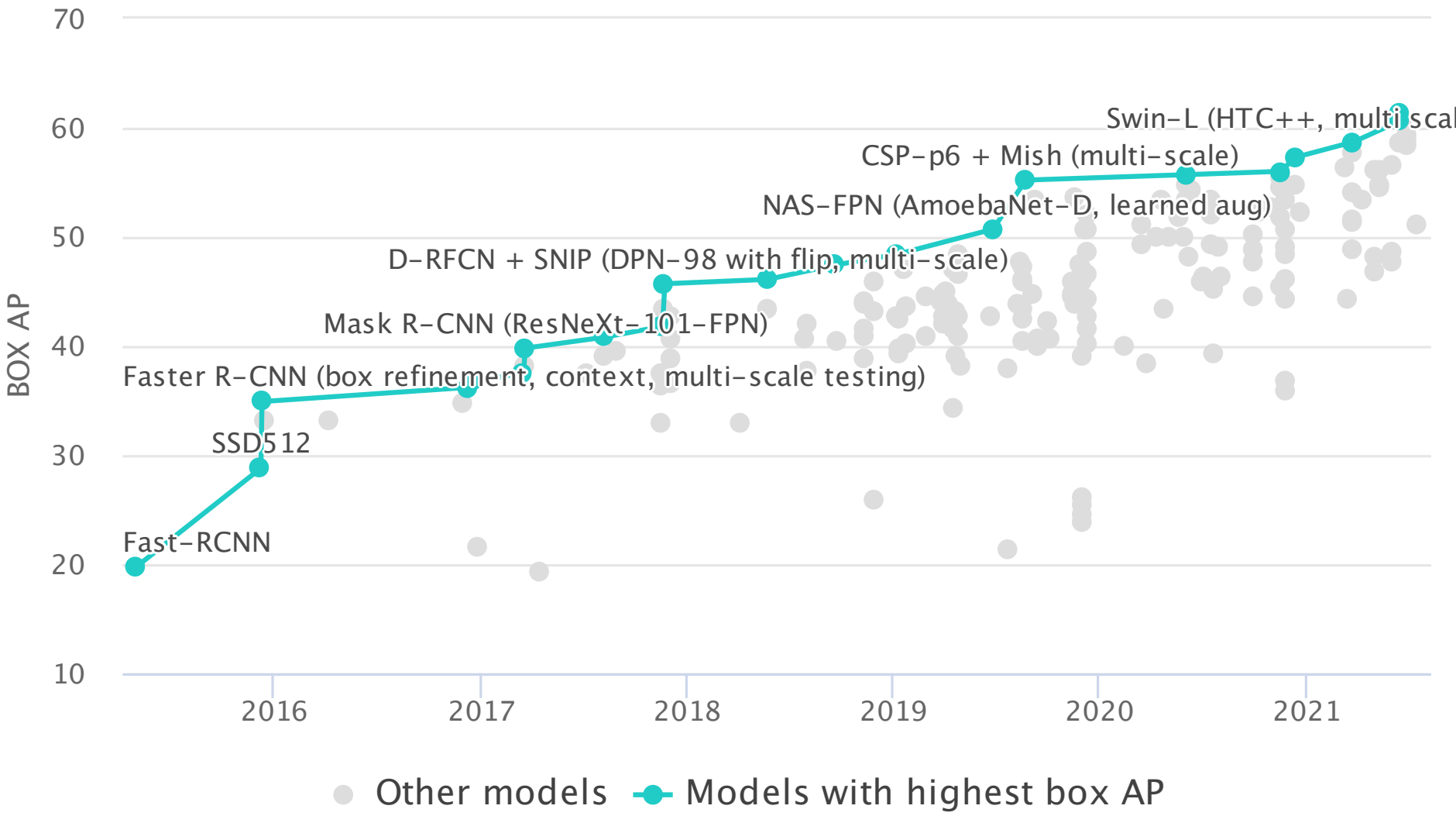
R. Girshick, J. Donahue, T. Darrell, and J. Malik, **Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation**, CVPR 2014.

Fast R-CNN



R. Girshick, Fast R-CNN, ICCV 2015

Steady Progress on Detection



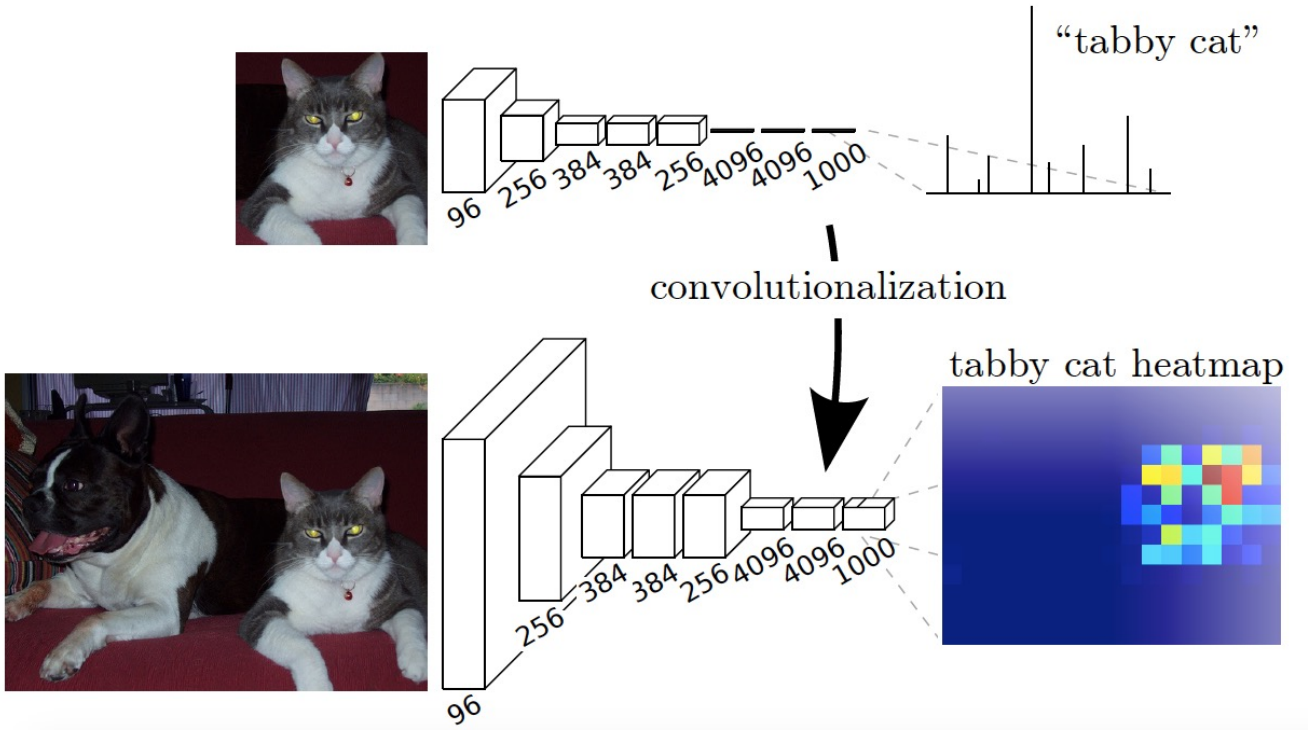
Source: https://paperswithcode.com/

Other Tasks: Semantic segmentation

Adapted from Fei-Fei Li

Convolutionalization

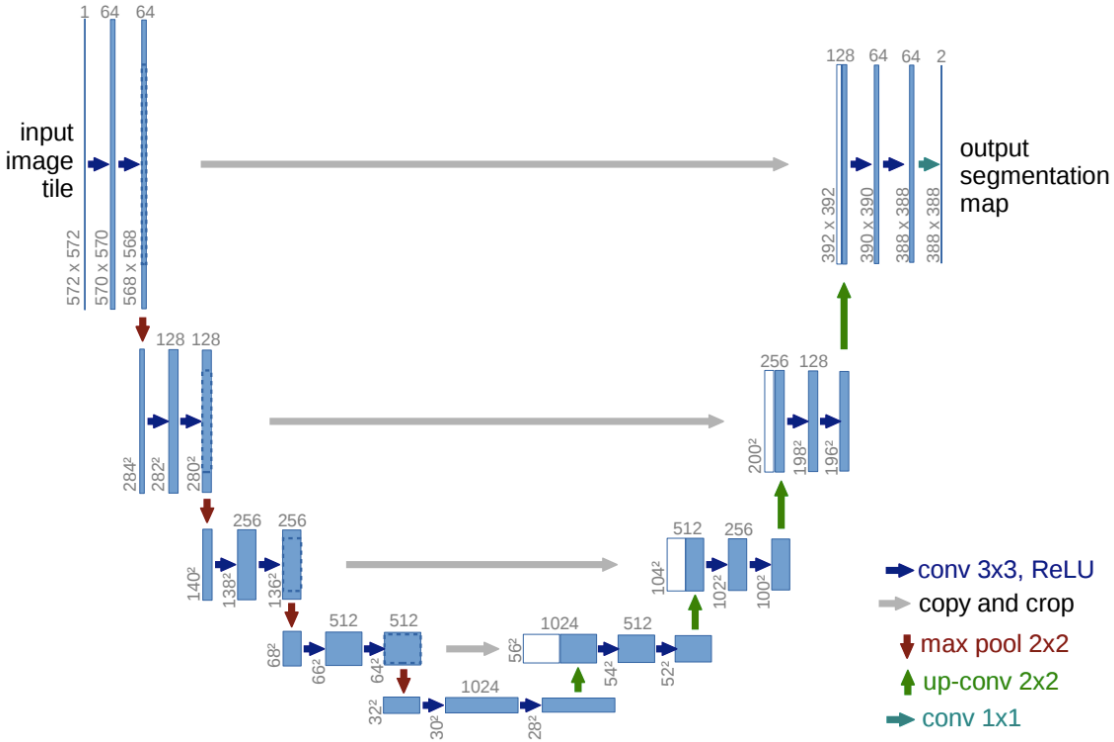
- Design a network with only convolutional layers, make predictions for all pixels at once



J. Long, E. Shelhamer, and T. Darrell, Fully Convolutional Networks for Semantic Segmentation, CVPR 2015

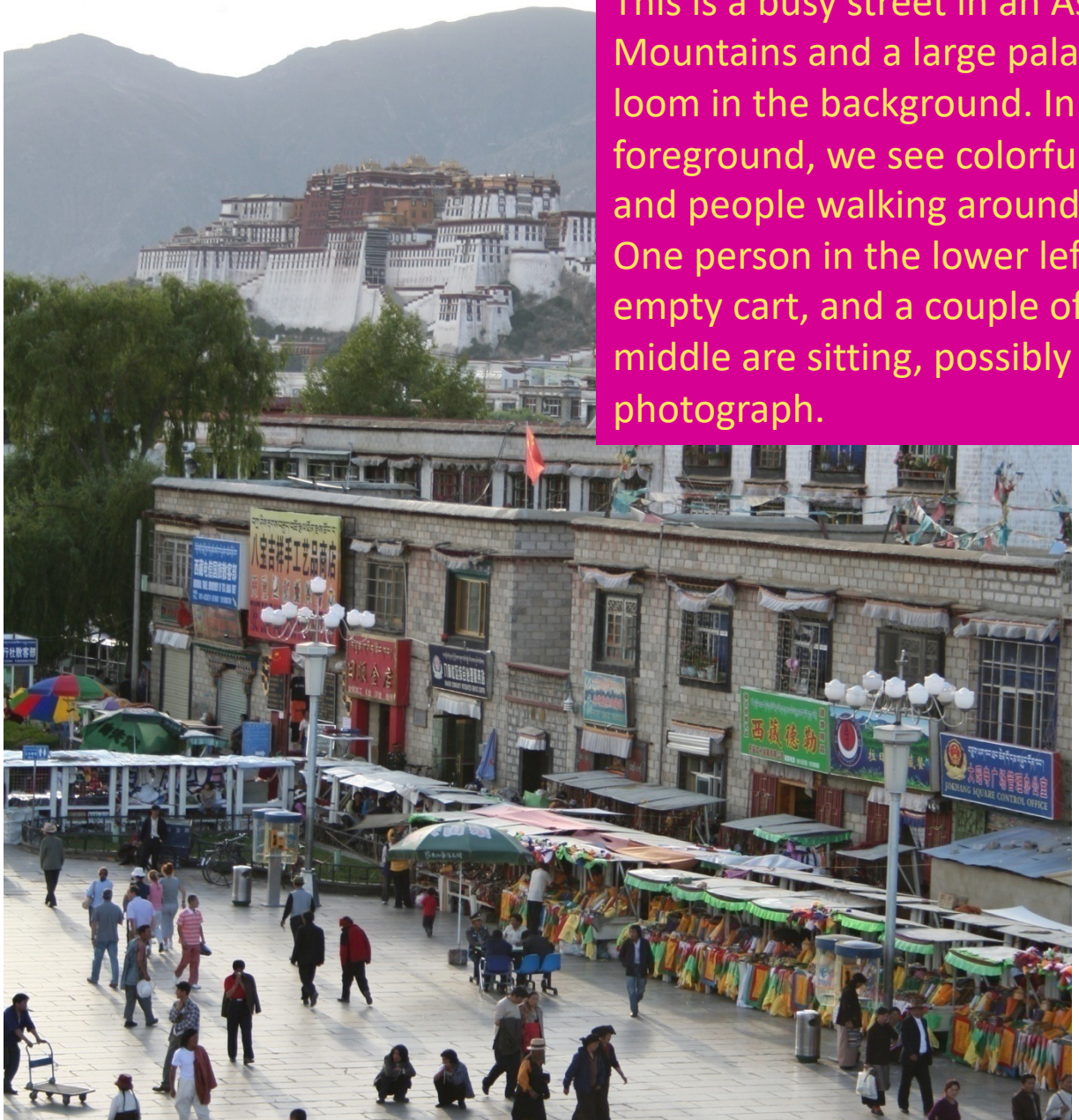
U-Net

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



O. Ronneberger, P. Fischer, T. Brox U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015

Image description

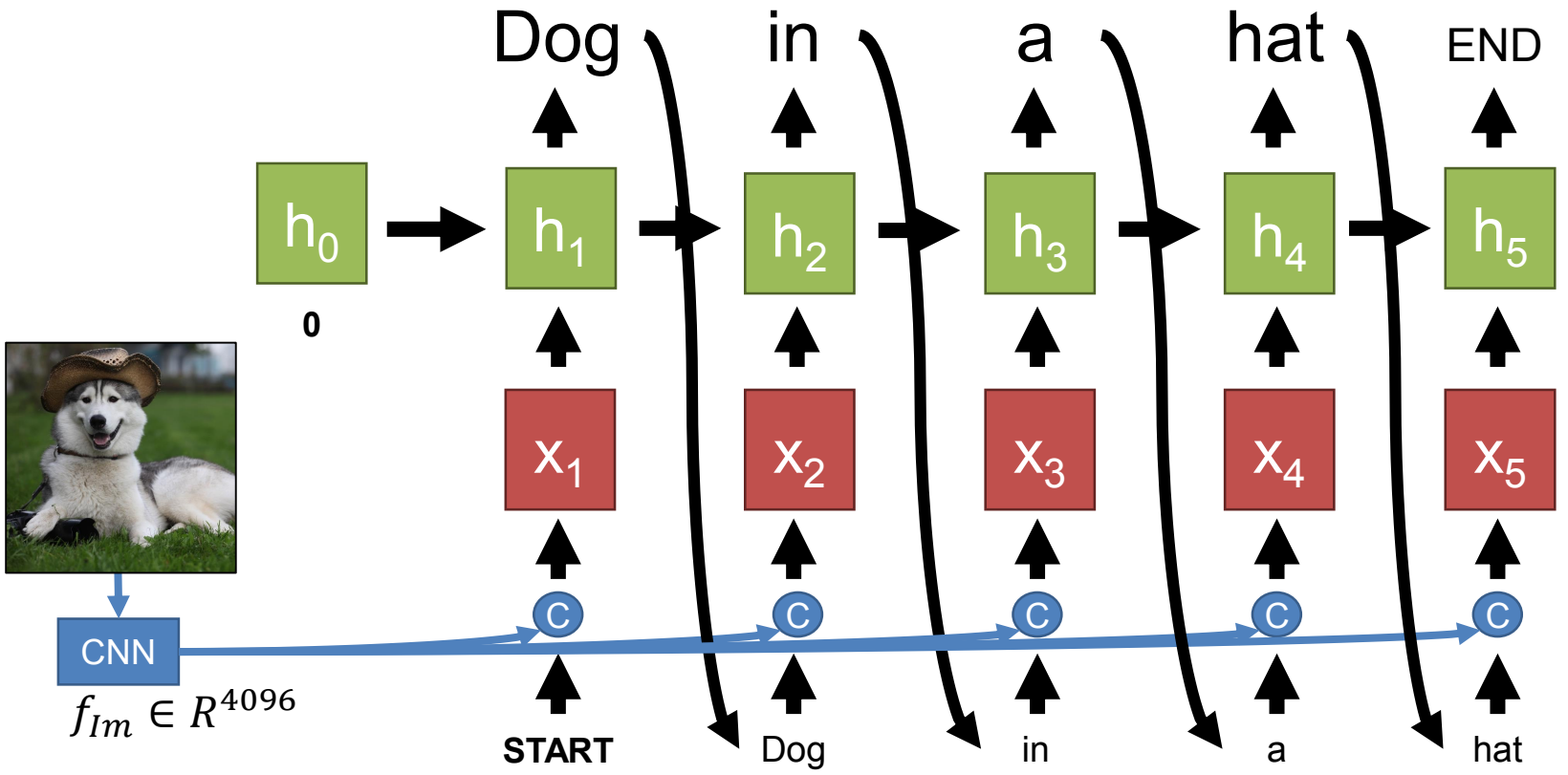


This is a busy street in an Asian city. Mountains and a large palace or fortress loom in the background. In the foreground, we see colorful souvenir stalls and people walking around and shopping. One person in the lower left is pushing an empty cart, and a couple of people in the middle are sitting, possibly posing for a photograph.

Adapted from Fei-Fei Li

Sequence Modeling

Captioning



Slide from D. Fouhey

See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Vision Review

- The vision problem
- Why is vision hard?
- 3D Reconstruction (basis from SLAM)
- Recognition

Additional Resources:

- See CS 543 / ECE 549 at UIUC: http://saurabhg.web.illinois.edu/teaching/ece549/sp2021
- See: CS 231n at Stanford: http://cs231n.stanford.edu/
- See Prof. A. Davison's Robotics course: https://www.doc.ic.ac.uk/~ajd/Robotics/
- Prof. Shenlong Wang's course on Robot Perception