https://tinyurl.com/mrx63mb6

Computer Vision 1

HC5b

Convolutional Neural Networks, Object Detection Architectures

Dr. Martin Oswald, Dr. Dimitris Tzionas, Dr. Arun Mukundan,

(m.r.oswald, d.tzionas, a.mukundan)@uva.nl

University of Amsterdam

computer vision lab

Image Understanding

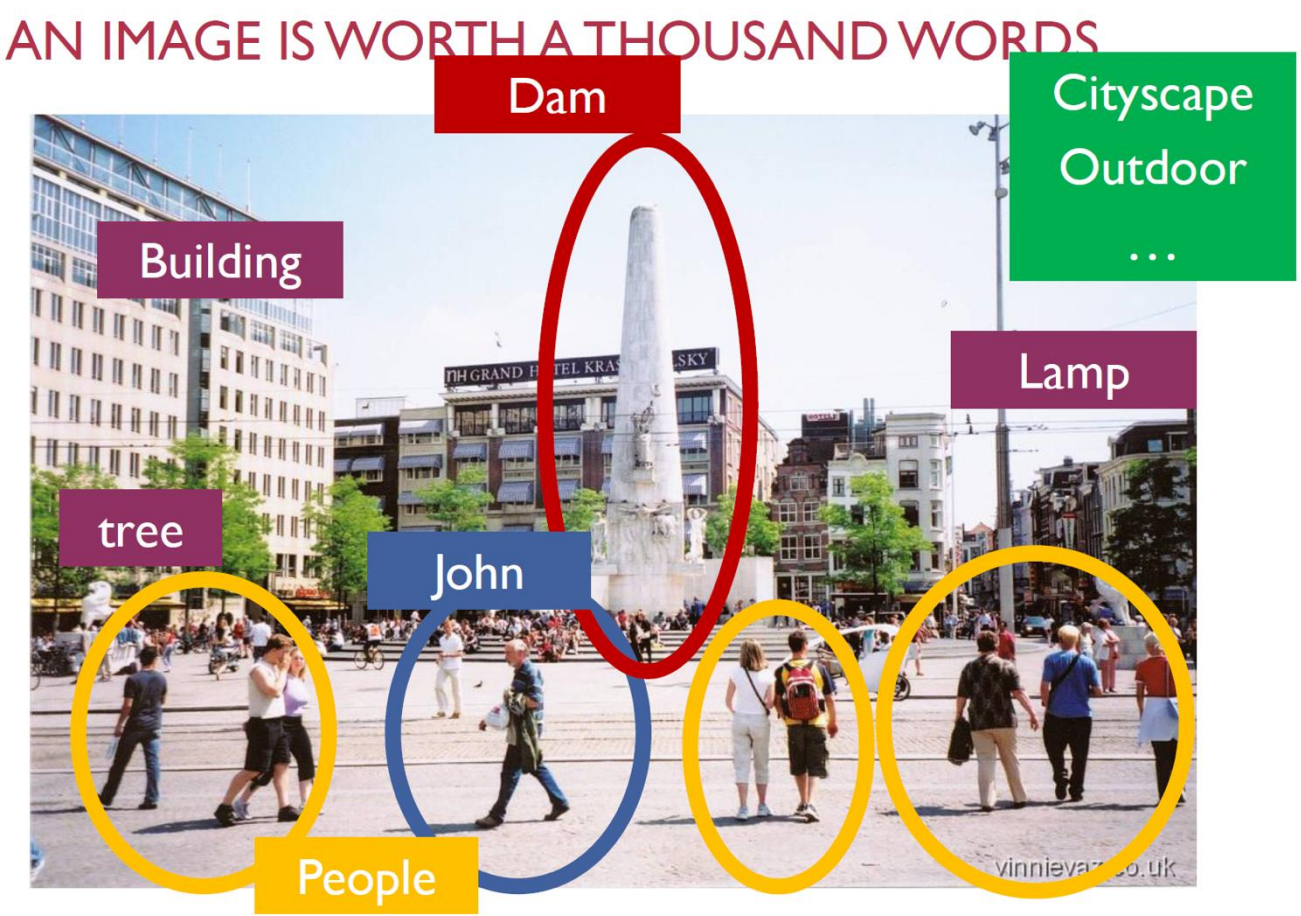
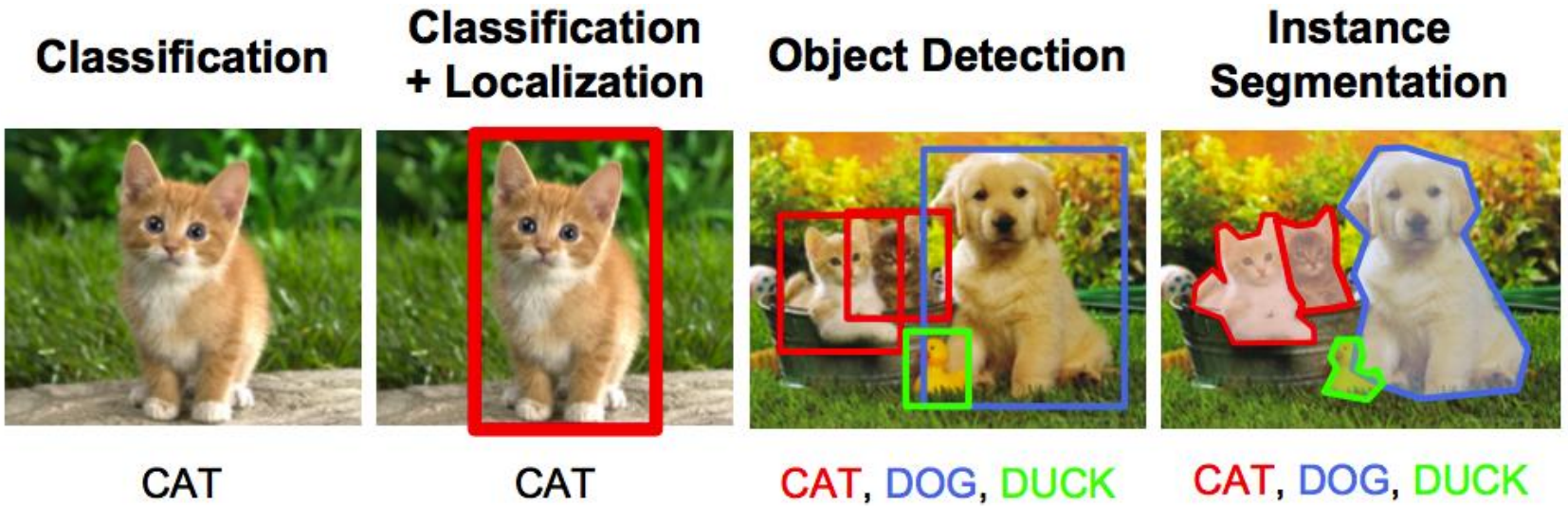


Image Understanding

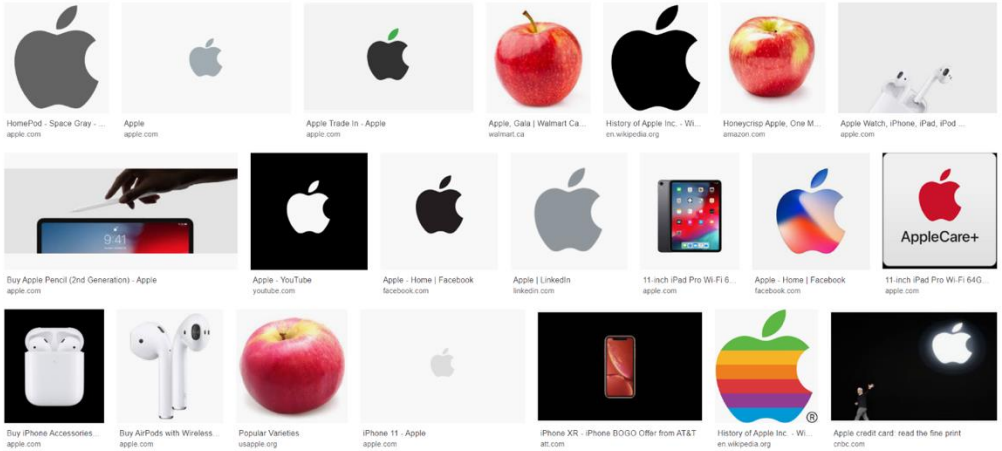


- Object Classification: classify the main object in an image if the image has one main object.
- Visual Image Retrieval: retrieve images from the library through a query image.
- Object Detection: identify whether there is an object in an image and recognize the object.
- Image Segmentation: segment an image into regions according the pixels the object belong to.

Recap

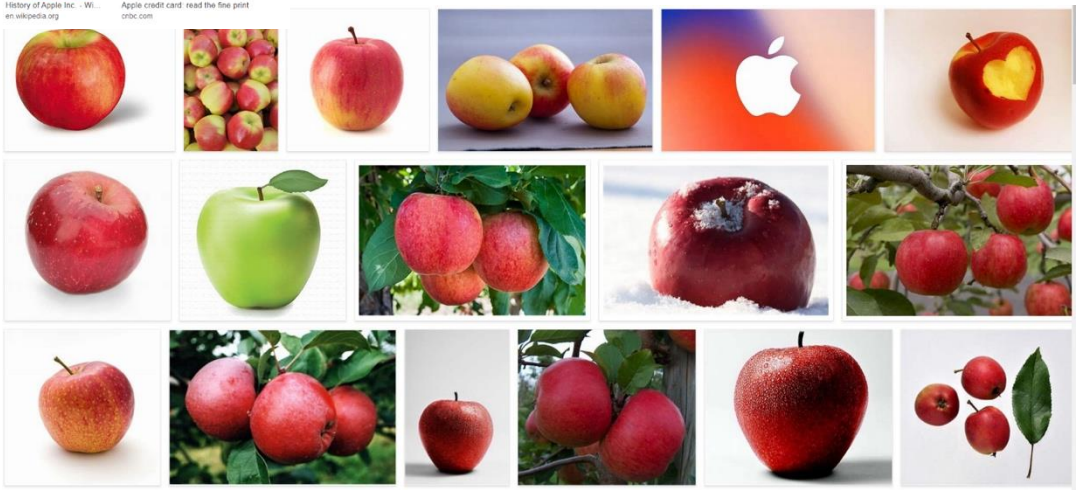
Image Retrieval

Search term: “appel”



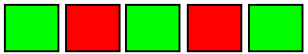
Google Search

Bing Search



Retrieval Evaluation

Average Precision@K

- Consider rank position of each relevant images
  - $K_1, K_2, \ldots, K_R$
- Compute Precision@K for each $K_1, K_2, \ldots, K_R$
- Average precision = average of P@K
- Example:  has average precision:

rel: 1 0 1 0 1

relevance: {0,1}

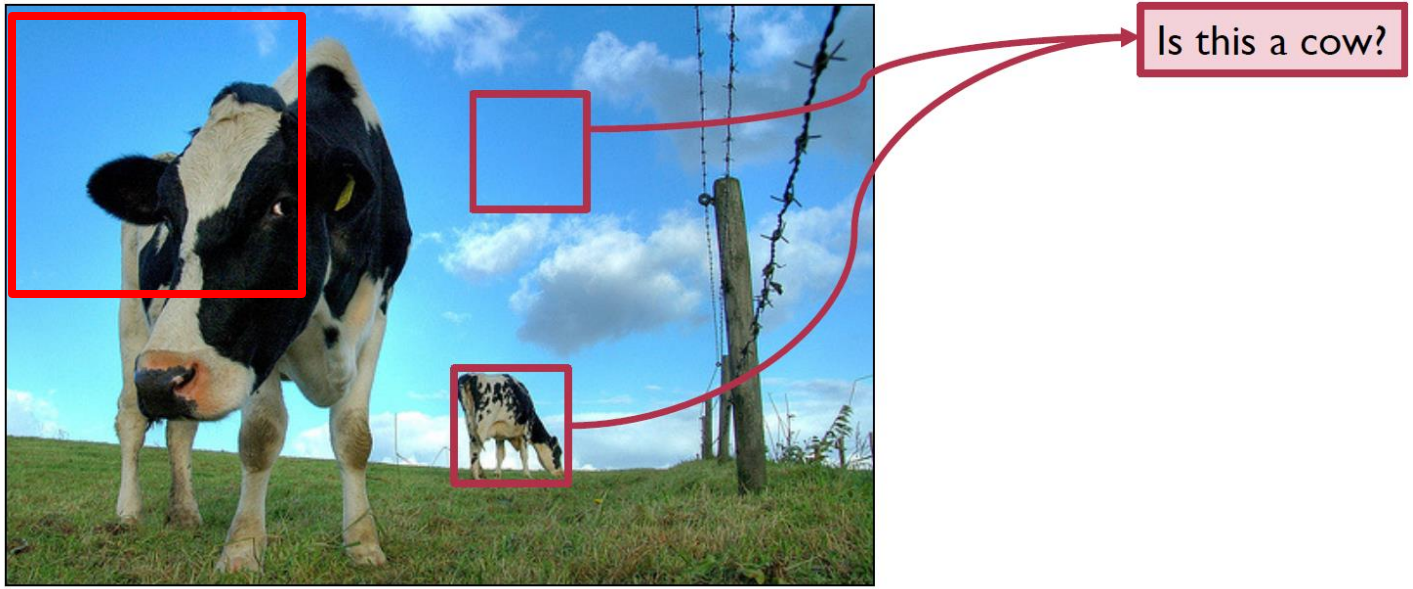
$$\textbf{AP@K} = \frac{\sum_{k=1}^{K}(P@k * relk)}{\#\ relevant\ results}$$

$$AP@K = \frac{1}{3} * \left({}^{1}/_{1} * 1 + {}^{1}/_{2} * 0 + {}^{2}/_{3} * 1 + {}^{2}/_{4} * 0 + {}^{3}/_{5} * 1\right) \approx 0.76$$

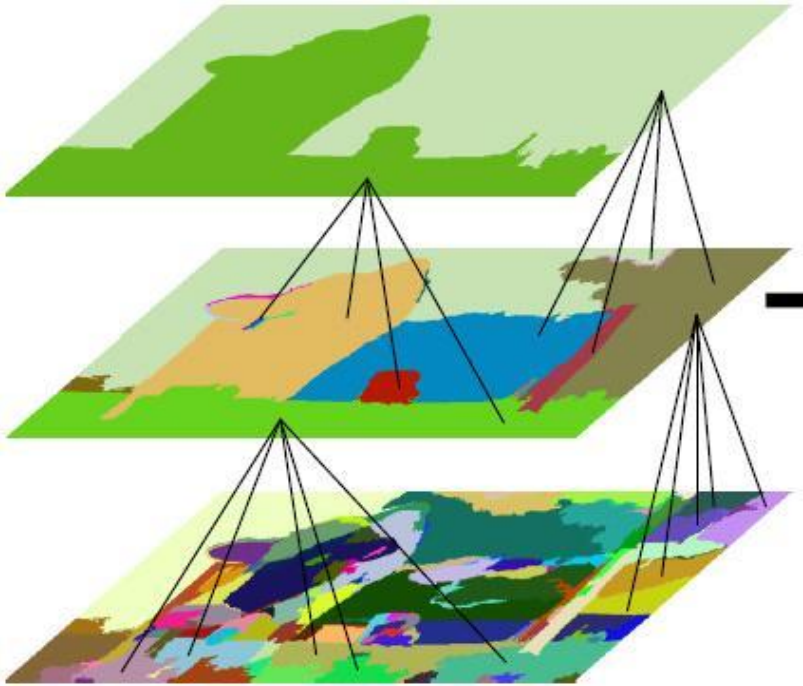
Object Detection

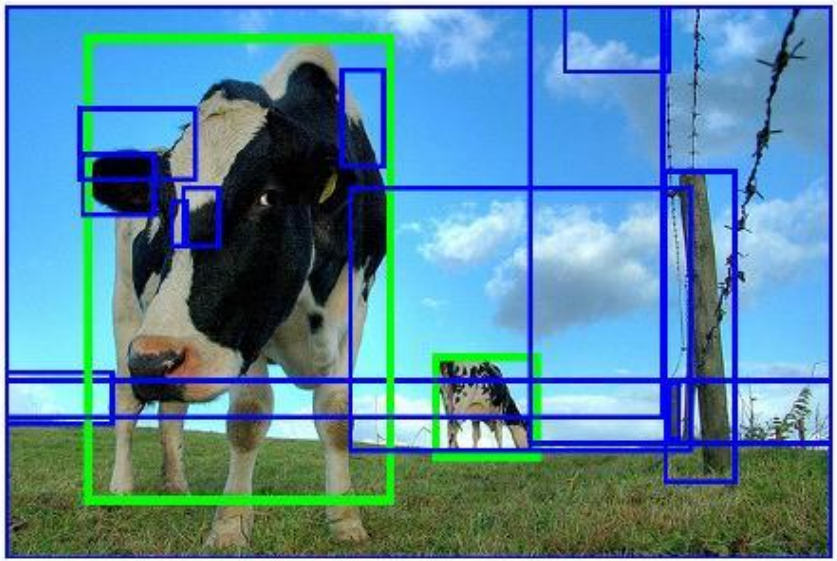
[https://youtu.be/7BgiFrPs-x8?si]

Region Proposals: Sliding Window



Region Proposals: Selective Search

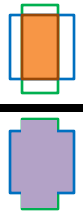




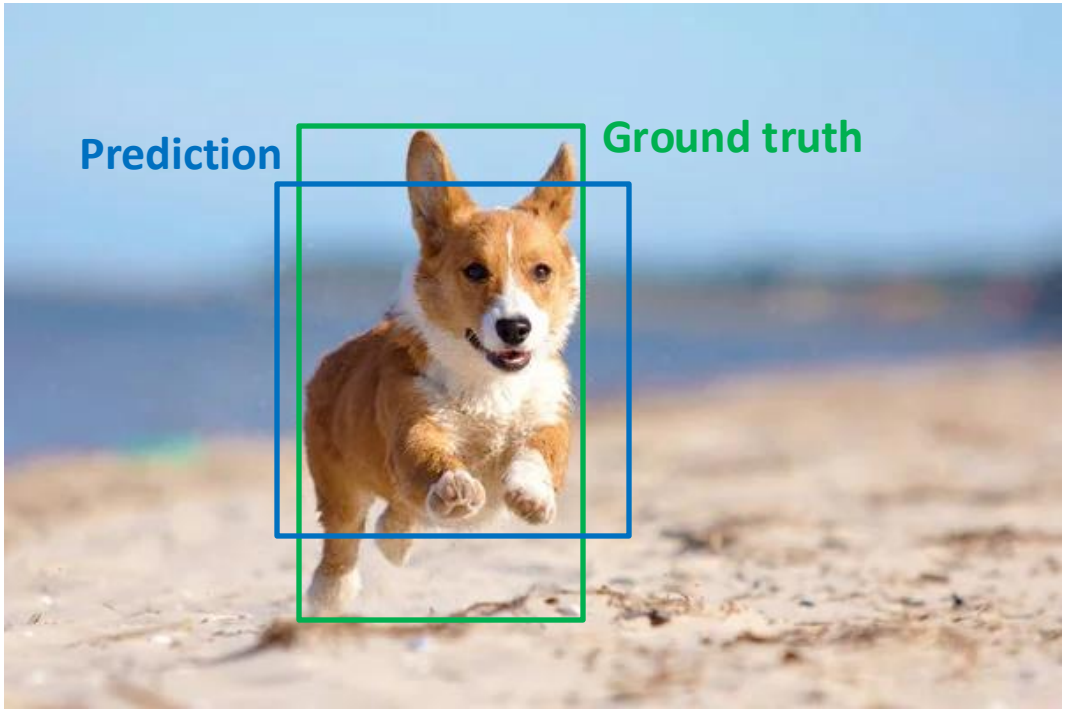
Comparing Bounding Boxes

- Intersection over Union (IoU)
  (a.k.a. “Jaccard similarity/index”)

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}}$$



- Typically choose a threshold e.g. IoU >= 0.5, or 0.7 between prediction and GT to decide about correct or incorrect predictions



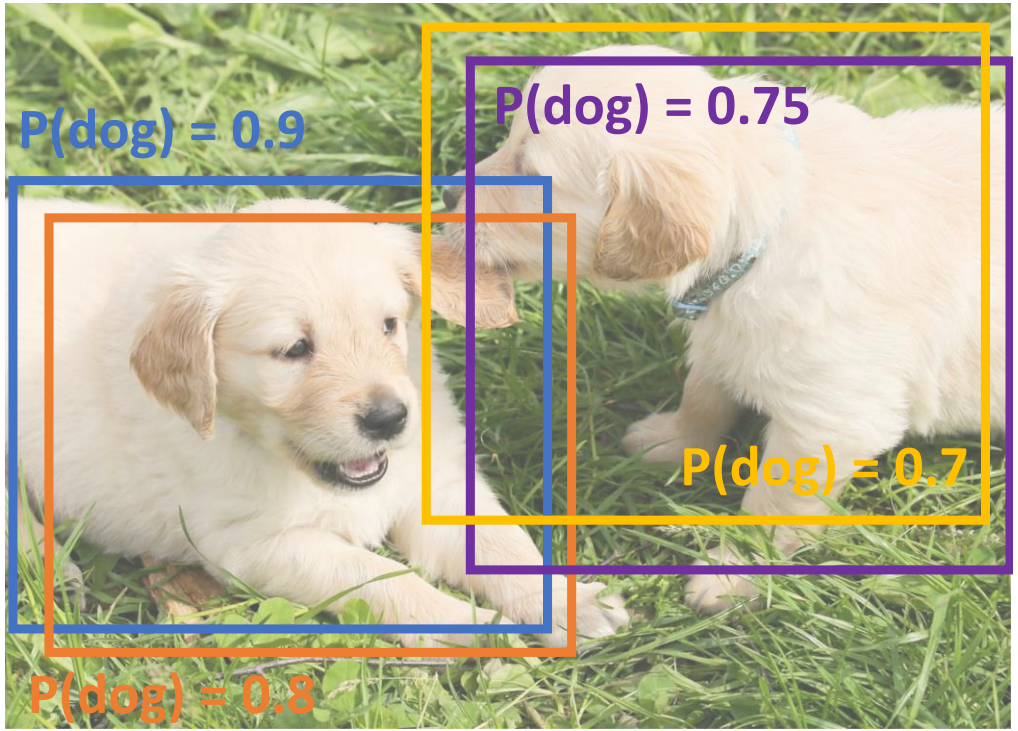
[https://depositphotos.com/photos/corgi-running.html]

Duplicate Filter: Non-Maximum Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using **Non-Max Suppression (NMS):**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



[Puppy image is CC0 Public Domain]

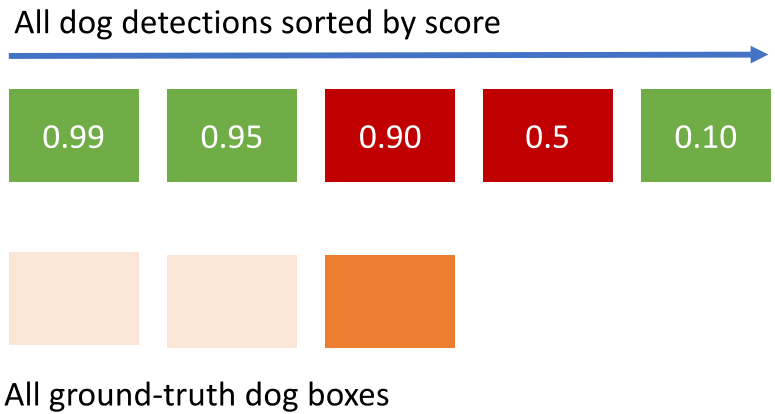
[Slide credit: Justin Johnson]

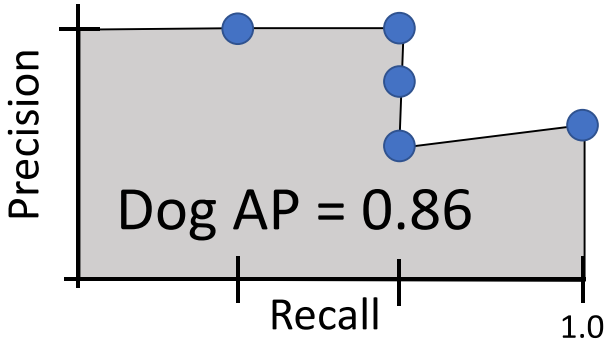
Evaluating Object Detectors

Mean Average Precision (mAP):

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

How to get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"



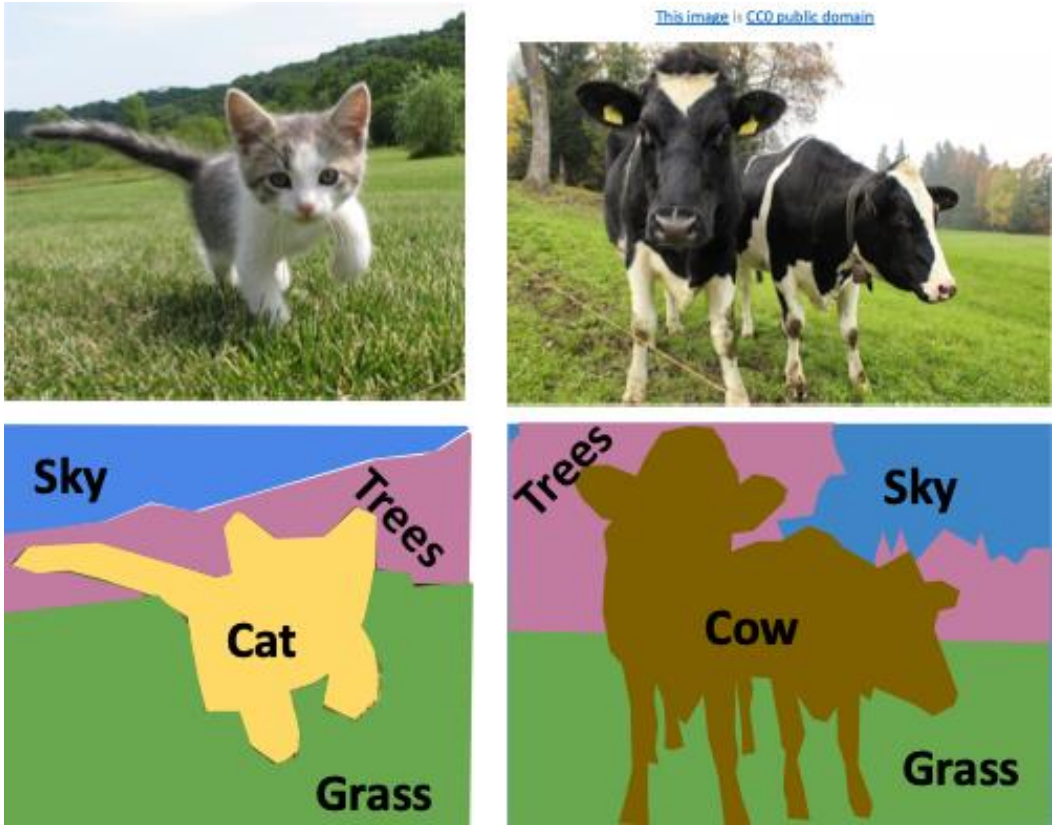


[Slide credit: Justin Johnson]

Image Segmentation

Goal: Partition the image into semantically-meaningful or perceptually-similar regions.
(In contrast to object detection: all pixels are labeled!)

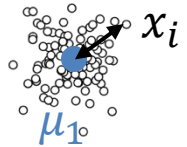
That is: Label each pixel in the image with a category label



K-Means

Goal: Minimize the distance between N given data points and K assigned cluster centers:

$$\text{minimize} \quad \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$



K-Means Algorithm

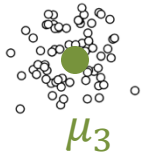
1. **Initialize** K cluster means $\mu_1, \dots, \mu_K$
2. **Reassign Points:** For i $= 1, \dots, N$ assign each point x_n to the closest cluster

   $$z_n = \text{argmin}_{k \in \{1,\dots,K\}} \|x_i - \mu_k\|^2$$

3. **Update Centroids:** Suppose $C_k = \{x_i : z_i = k\}$. Recompute the means as

   $$\mu_k = \text{mean}(C_k), \quad k = 1, \dots, K$$

4. **Repeat:** Go to step 2 if not yet converged



Today's Agenda

- Convolutional Neural Networks
- Object Detection with Region-based CNNs

Today's Agenda

- **Convolutional Neural Networks**
- Object Detection with Region-based CNNs

Historical Context

Traditional techniques till year 2000: Basis for neural networks already second half of last century.

1954 – Bag of Words (Harris)
1972 – Hough Transform (Hough, Duda)

1943 – Neuron (McCulloch/Pitts)
1958 – Perceptron (Rosenblatt)

First AI Winter (1974-1980)

1986 – HoG (Mc Connel)
1986 – Canny Edge Detection (Canny)
1988 – Harris Corner (Harris, Stephens)

1985 – BackPropagation (Werbos, Parker, Rumelhart, Hinton, Williams)

Second AI Winter (1987-1993)

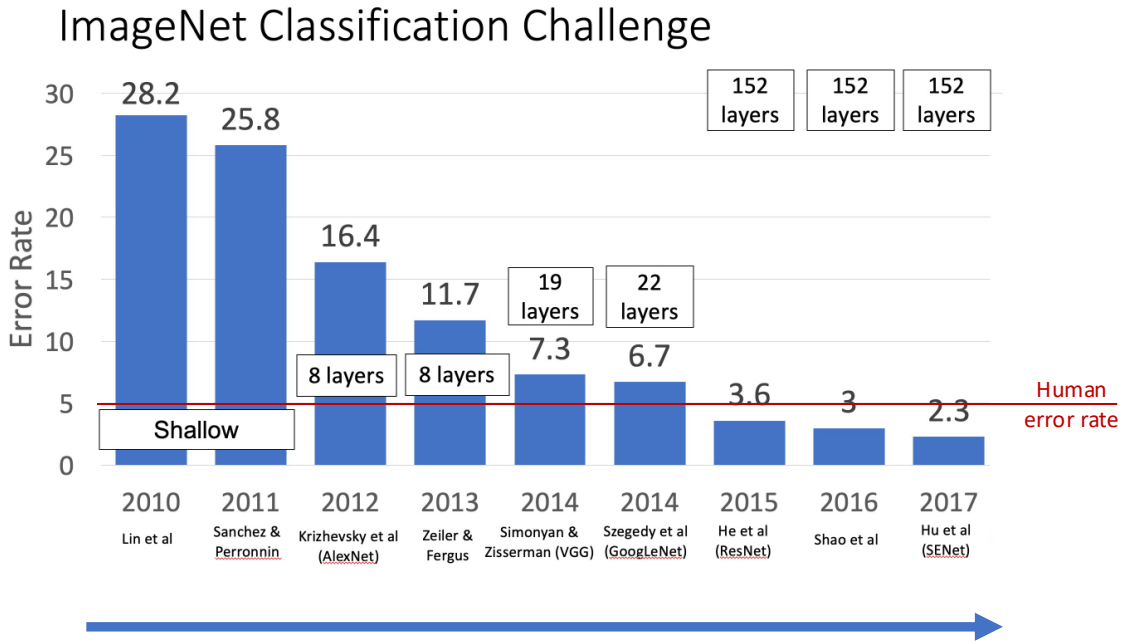
1999 – SIFT (Lowe)

1992 – 1995 SVM (Vapnik et al.)

[https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec]

Historical Context

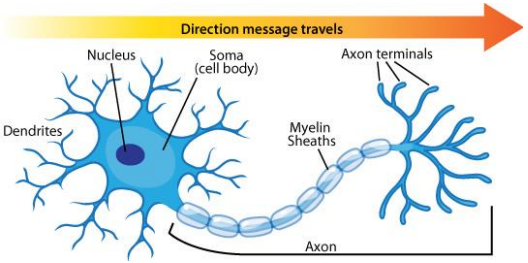
1989 - **CNN** (by LeCun) – handwritten digit recognition
1997 - **Long short-term memory (LSTM)**
1998 - **LeNet-5** (by LeCun)
2010 - **ImageNet** large scale visual recognition challenge (ILSVRC)
2012 - **AlexNet**
2013 - **R-CNN**
2014 - **VGG Net**
2014 - **GAN**
2015 - **ResNet**
2015 - **U-Net**
2015 - **Yolo - Darknet**
2016 - **R-CNN**
2017 - **Transformer**
2021 - **Vision Transformer (ViT)**
2022 - **Masked Autoencoder (MAE)**
2022 - **Stable Diffusion**



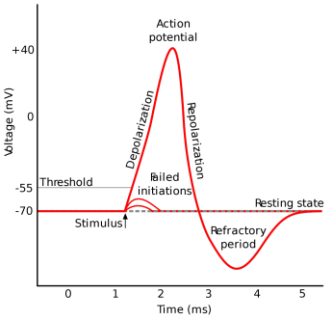
Artificial Neuron and Perceptron

- Biology inspired McCullock & Pitts (1943) to an artificial 'neuron'
- Rosenblatt added weights to the neuron model (1957)

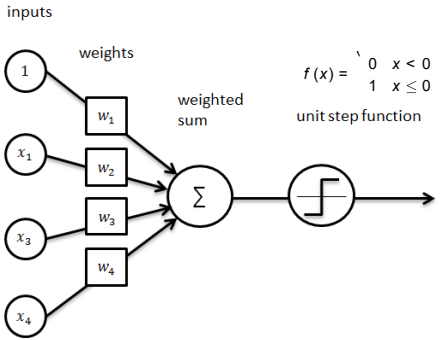
Biological model of a neuron



Stimulus behavior

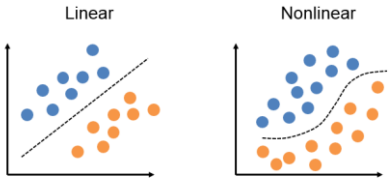


Rosenblatt's Perceptron



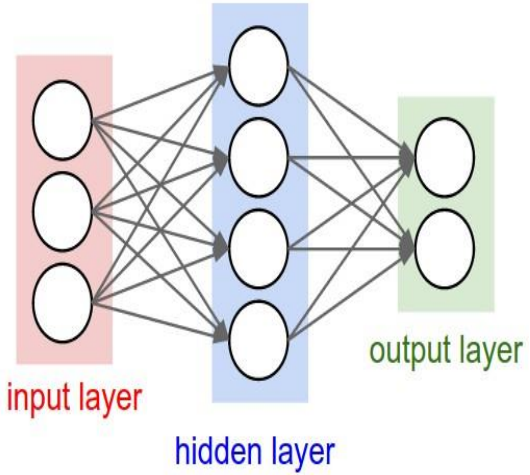
$$y = f\left(\sum_{i=1}^{n} x_i \cdot w_i\right)$$

- Artificial neuron
- Linear classifier, does only work for linearly separable data

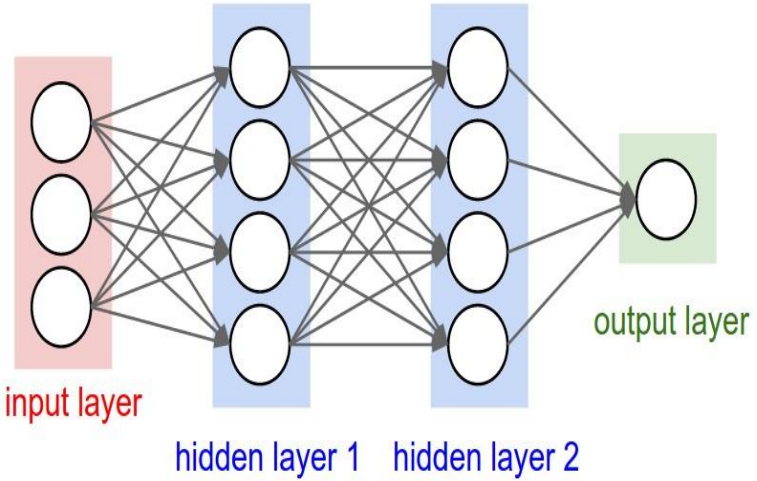


Multi-layer Perceptron (MLP)

- …is a '*fully connected*' neural network with non-linear activation functions
- …is a 'feed forward' neural network



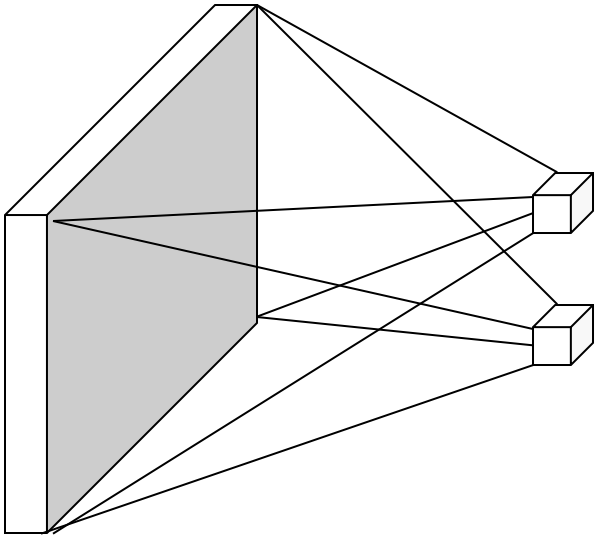
"2-layer Neural Net", or
"1-hidden-layer Neural Net"



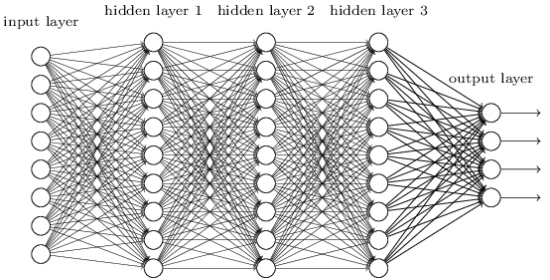
"3-layer Neural Net", or
"2-hidden-layer Neural Net"

[Andrej Karpathy, Fei-Fei Li, Justin Johnson]

A Neural Network for Images



image

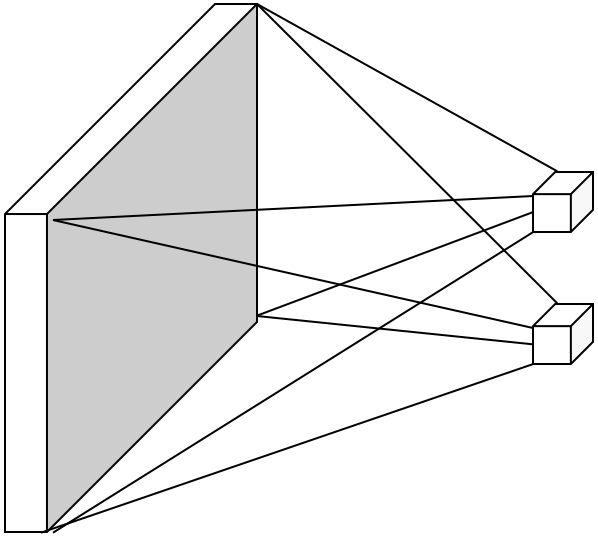


Fully connected layer

- This kind of design is known as *multi-layer perceptron* (MLP)

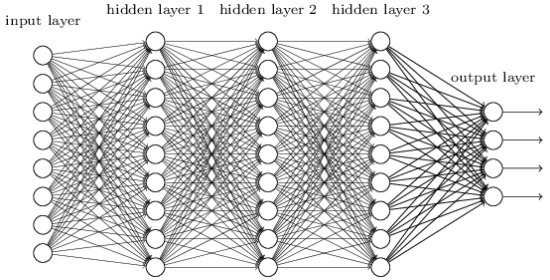
[Slide credit: S. Lazebnik]

A Neural Network for Images



image

Fully connected layer

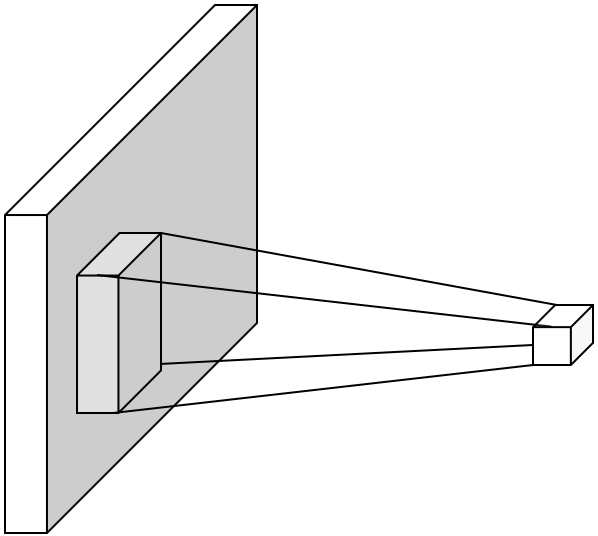


- This kind of design is known as *multi-layer perceptron* (MLP)
- What is wrong with this?

Recall: MLP as bank of whole-image templates

[Slide credit: S. Lazebnik]

Convolutional Architecture

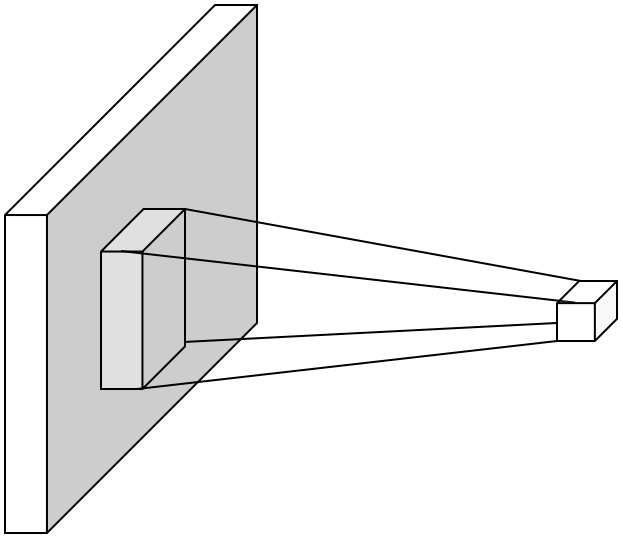


image

- Let's limit the *receptive fields* of units, tile them over the input image, and share their weights

[Slide credit: S. Lazebnik]

Convolutional Architecture

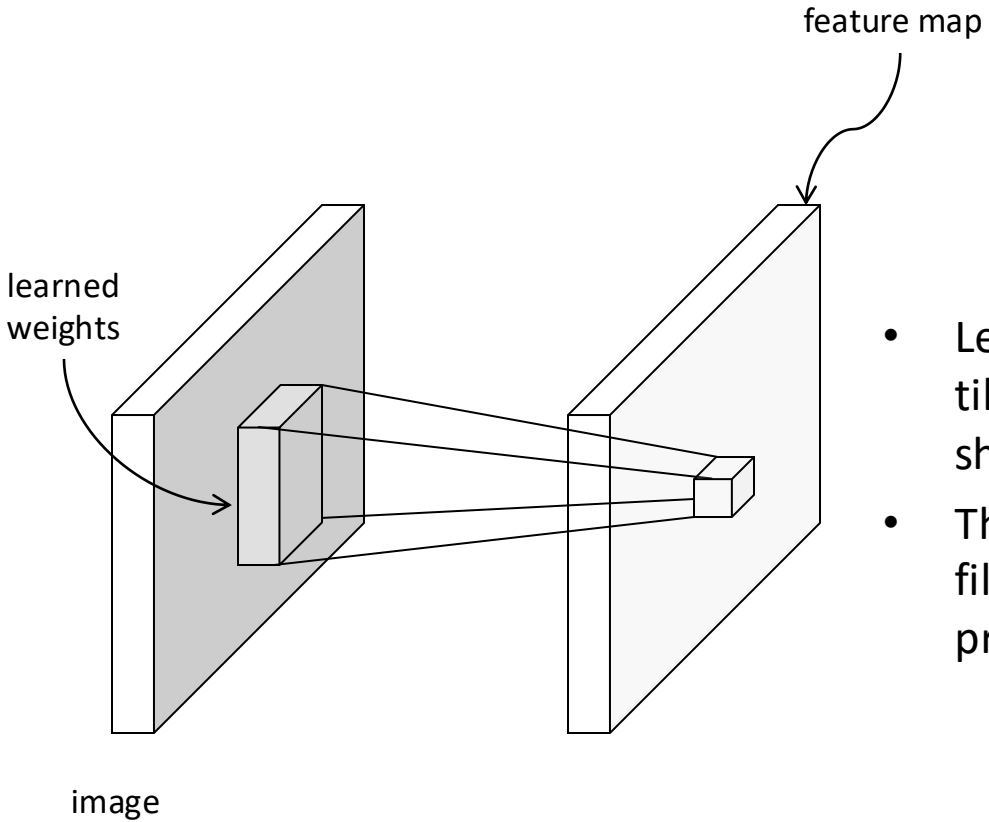


image

- Let's limit the *receptive fields* of units, tile them over the input image, and share their weights

[Slide credit: S. Lazebnik]

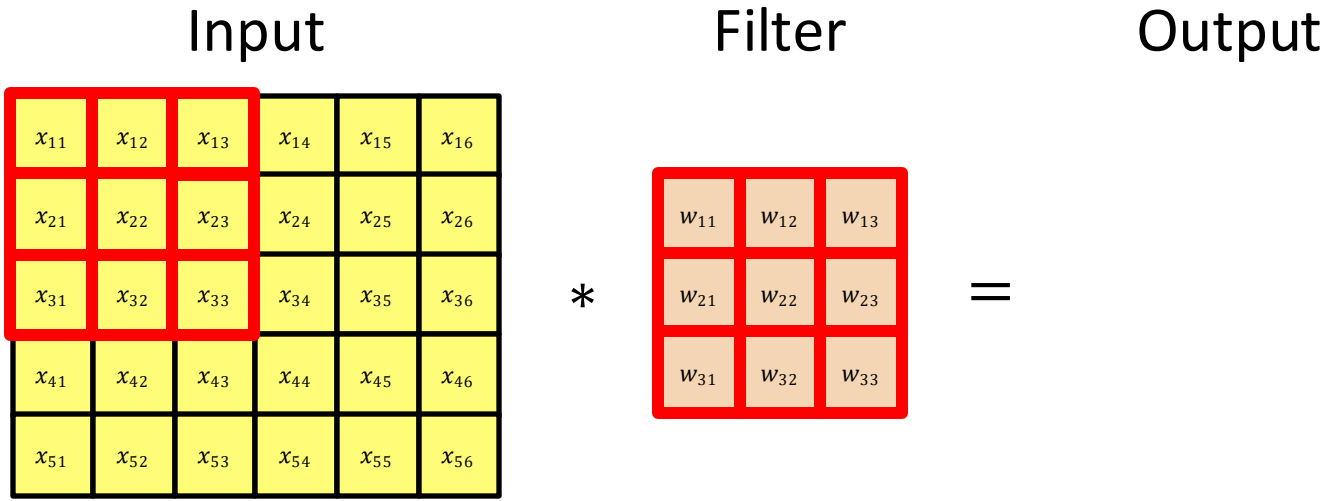
Convolutional Architecture



- Let's limit the *receptive fields* of units, tile them over the input image, and share their weights
- This is equivalent to sliding the learned filter over the image, computing dot products at every location

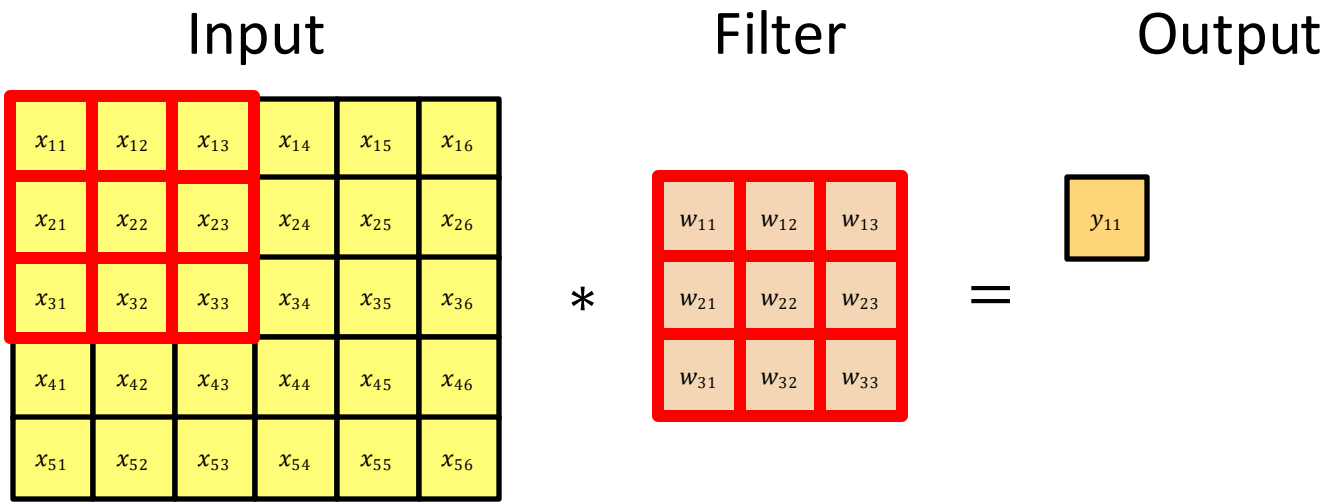
[Slide credit: S. Lazebnik]

Convolutional Architecture



[Slide credit: S. Lazebnik]

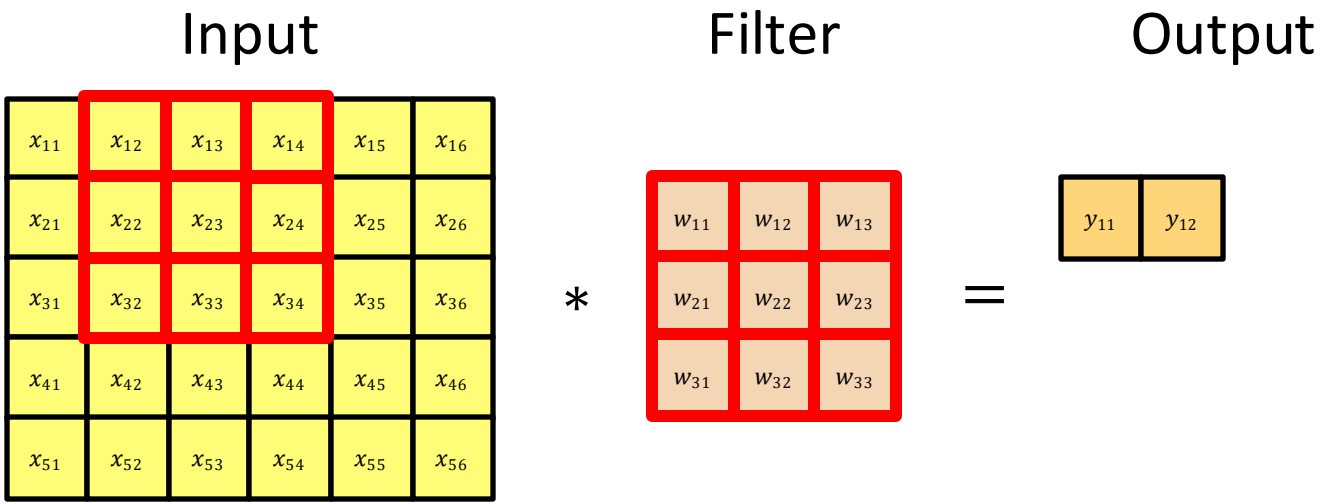
Convolutional Architecture



$$y_{11} = x_{11} \cdot w_{11} + x_{12} \cdot w_{12} + x_{13} \cdot w_{13} + \ldots + x_{33} \cdot w_{33}$$

[Slide credit: S. Lazebnik]

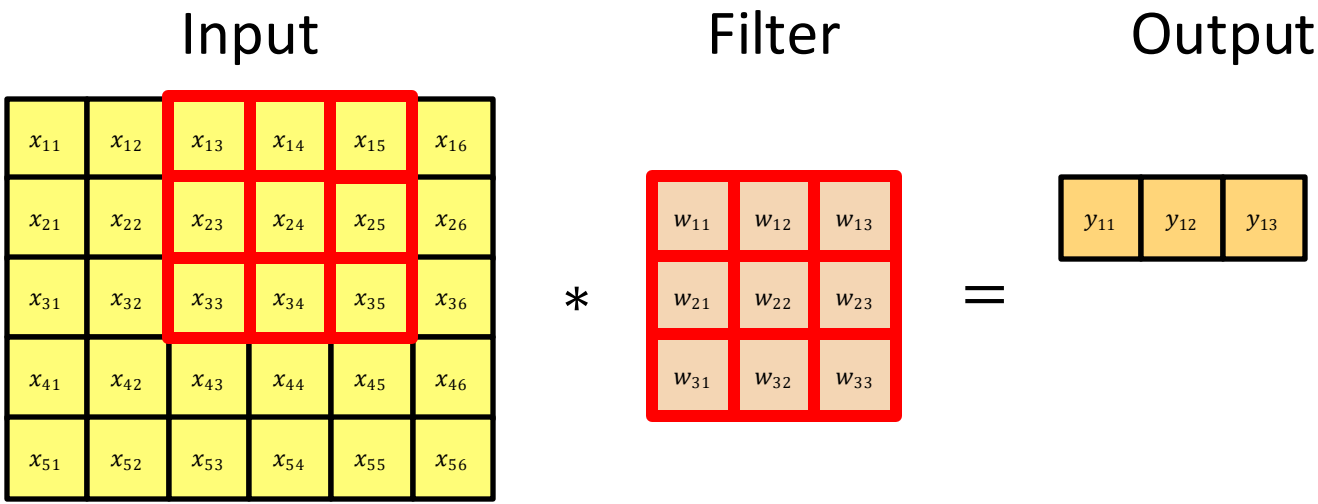
Convolutional Architecture



$$y_{12} = x_{12} \cdot w_{11} + x_{13} \cdot w_{12} + x_{14} \cdot w_{13} + \ldots + x_{34} \cdot w_{33}$$

[Slide credit: S. Lazebnik]

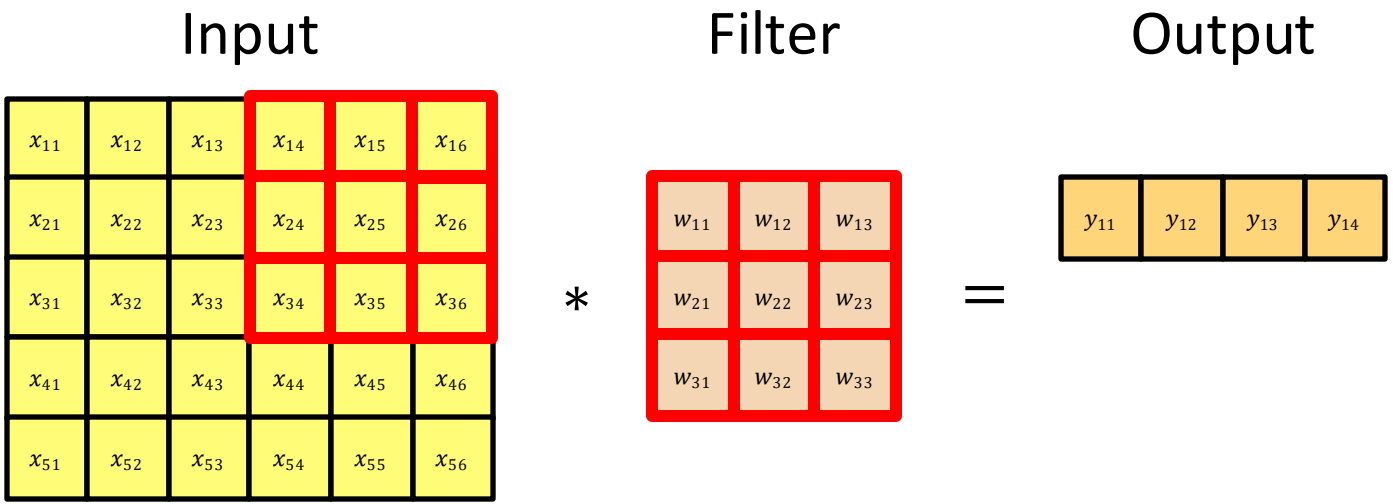
Convolutional Architecture



$$y_{13} = x_{13} \cdot w_{11} + x_{14} \cdot w_{12} + x_{15} \cdot w_{13} + \ldots + x_{35} \cdot w_{33}$$

[Slide credit: S. Lazebnik]

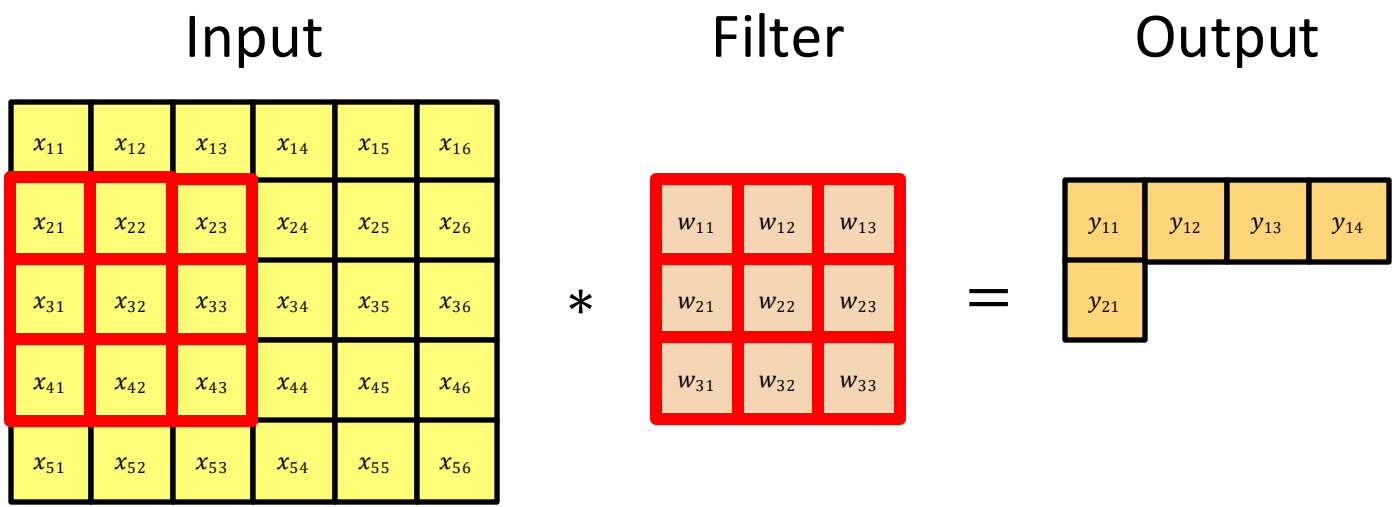
Convolutional Architecture



$$y_{14} = x_{14} \cdot w_{11} + x_{15} \cdot w_{12} + x_{16} \cdot w_{13} + \ldots + x_{36} \cdot w_{33}$$

[Slide credit: S. Lazebnik]

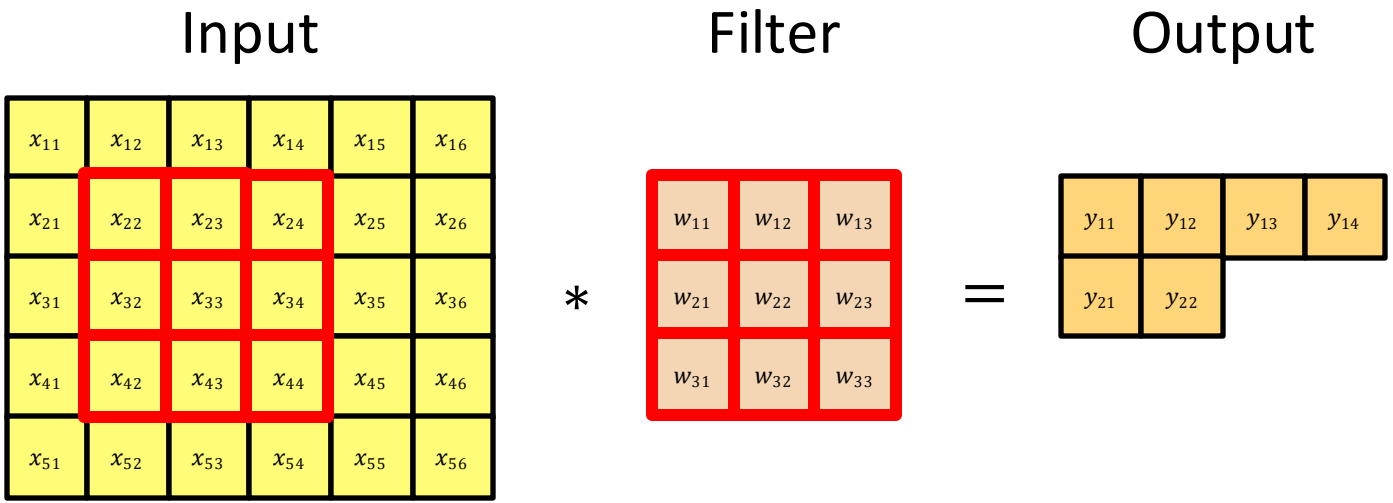
Convolutional Architecture



$$y_{21} = x_{21} \cdot w_{11} + x_{22} \cdot w_{12} + x_{23} \cdot w_{13} + \ldots + x_{43} \cdot w_{33}$$

[Slide credit: S. Lazebnik]

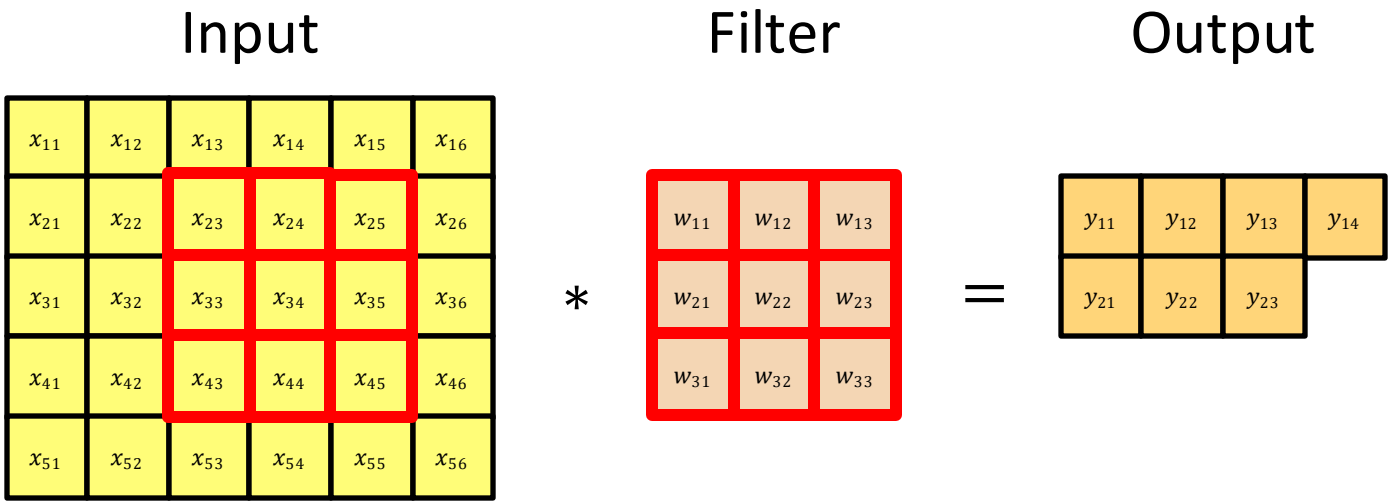
Convolutional Architecture



$$y_{22} = x_{22} \cdot w_{11} + x_{23} \cdot w_{12} + x_{24} \cdot w_{13} + \ldots + x_{44} \cdot w_{33}$$

[Slide credit: S. Lazebnik]

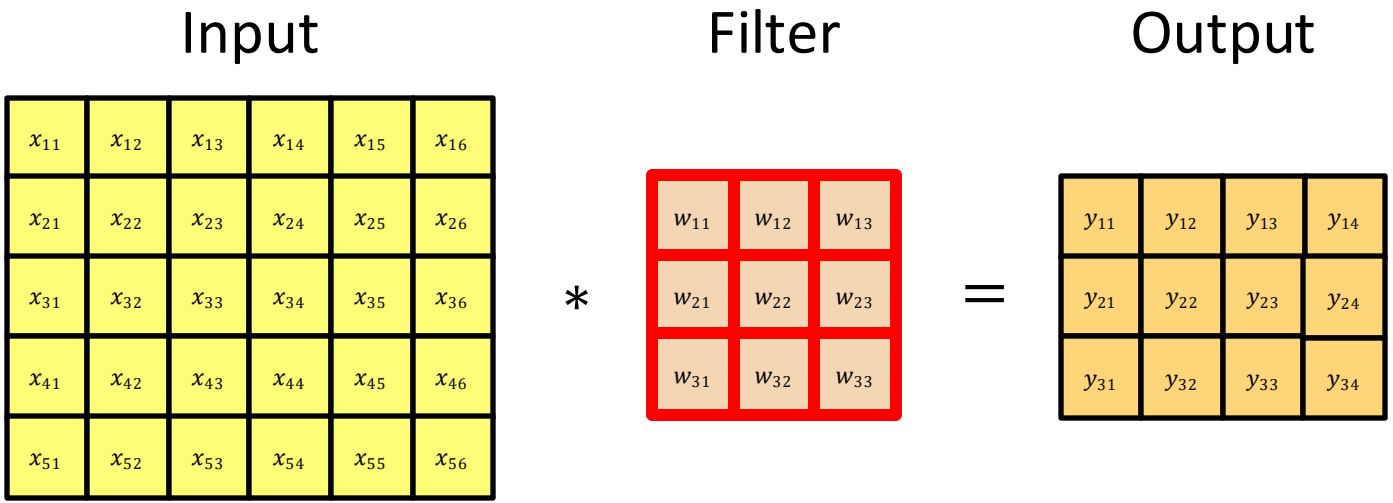
Convolutional Architecture



$$y_{23} = x_{23} \cdot w_{11} + x_{24} \cdot w_{12} + x_{25} \cdot w_{13} + \ldots + x_{45} \cdot w_{33}$$

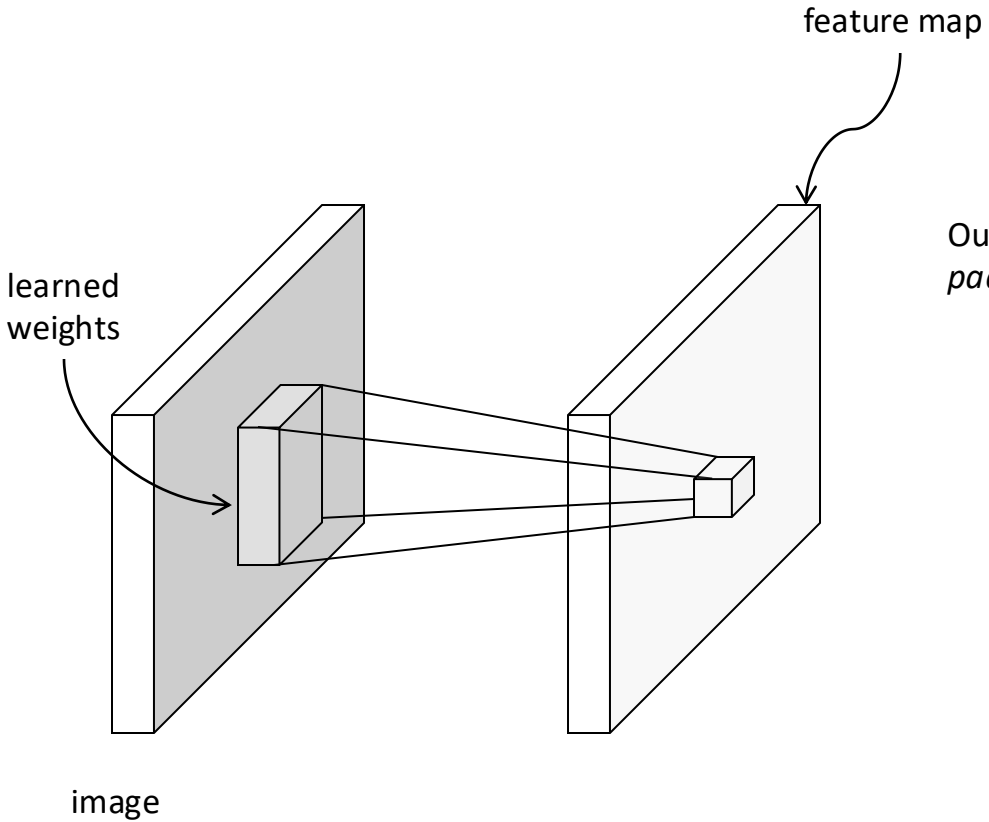
[Slide credit: S. Lazebnik]

Convolutional Architecture

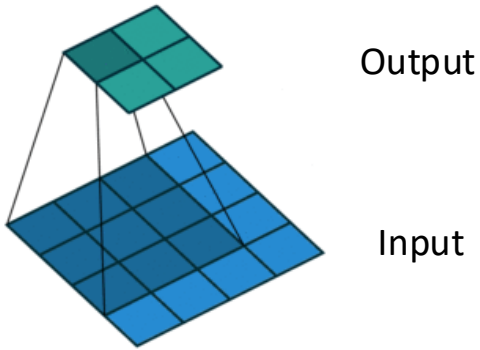


[Slide credit: S. Lazebnik]

Convolutional Architecture



Output feature map resolution depends on *padding* and *stride*

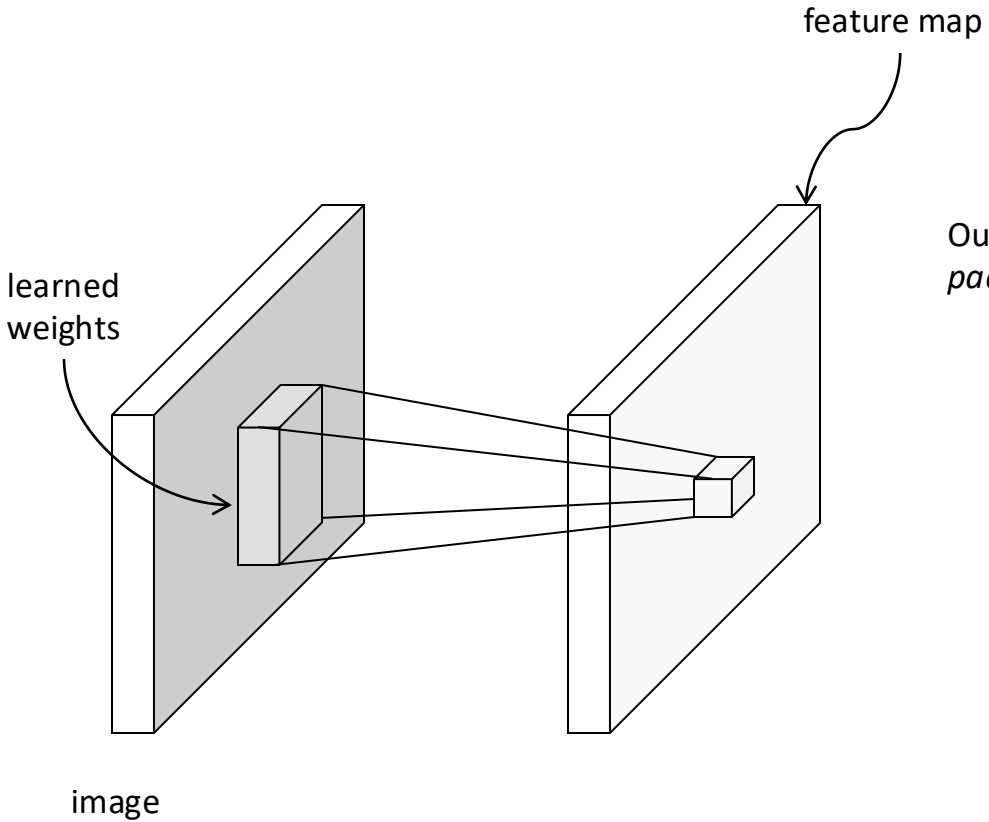


No padding, stride 1

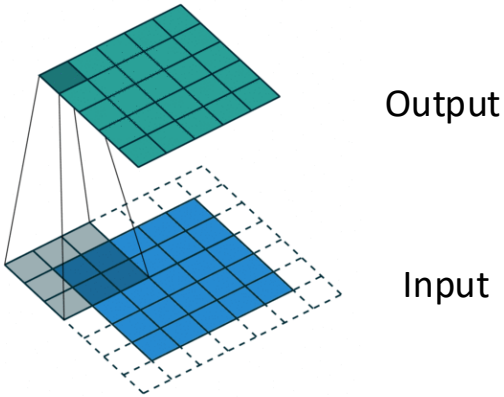
[Slide credit: S. Lazebnik]

[Animation source]

Convolutional Architecture



Output feature map resolution depends on *padding* and *stride*

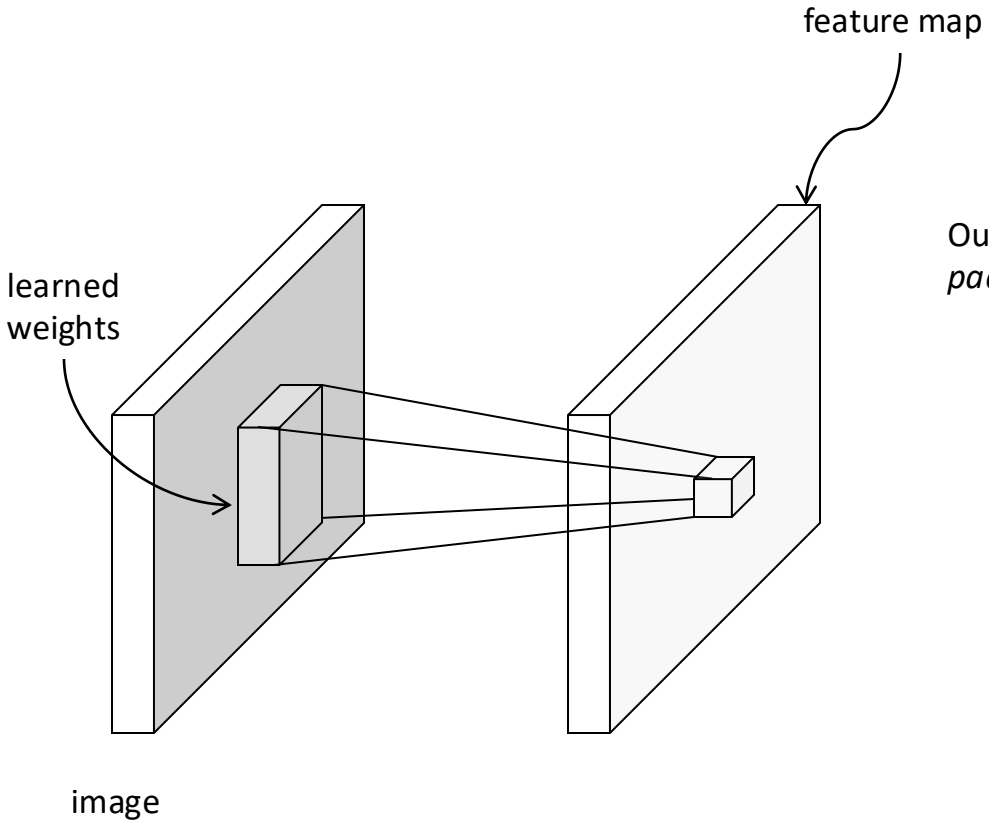


With padding, stride 1

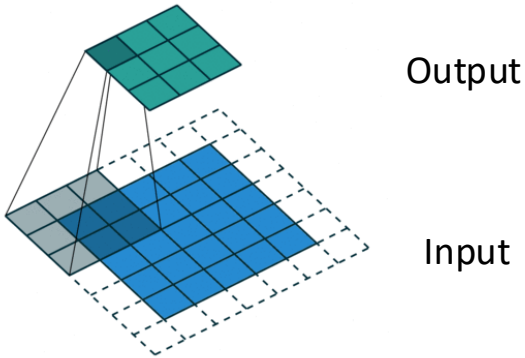
[Slide credit: S. Lazebnik]

[Animation source]

Convolutional Architecture



Output feature map resolution depends on *padding* and *stride*

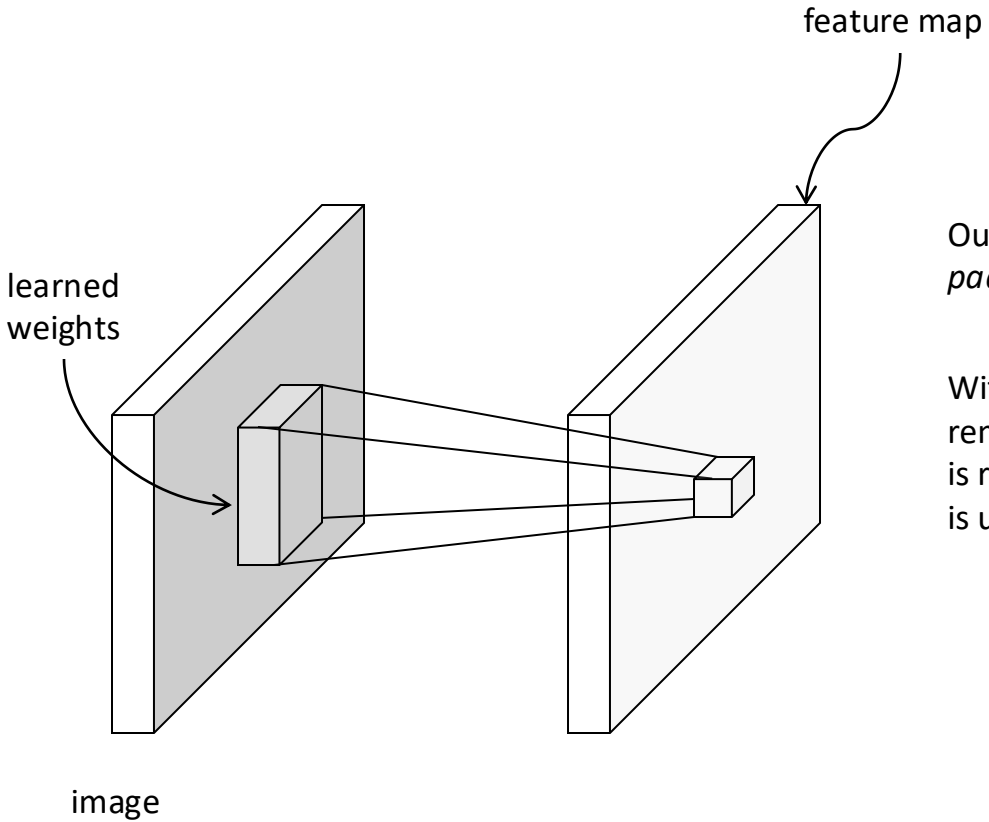


With padding, stride 2

[Slide credit: S. Lazebnik]

[Animation source]

Convolutional Architecture

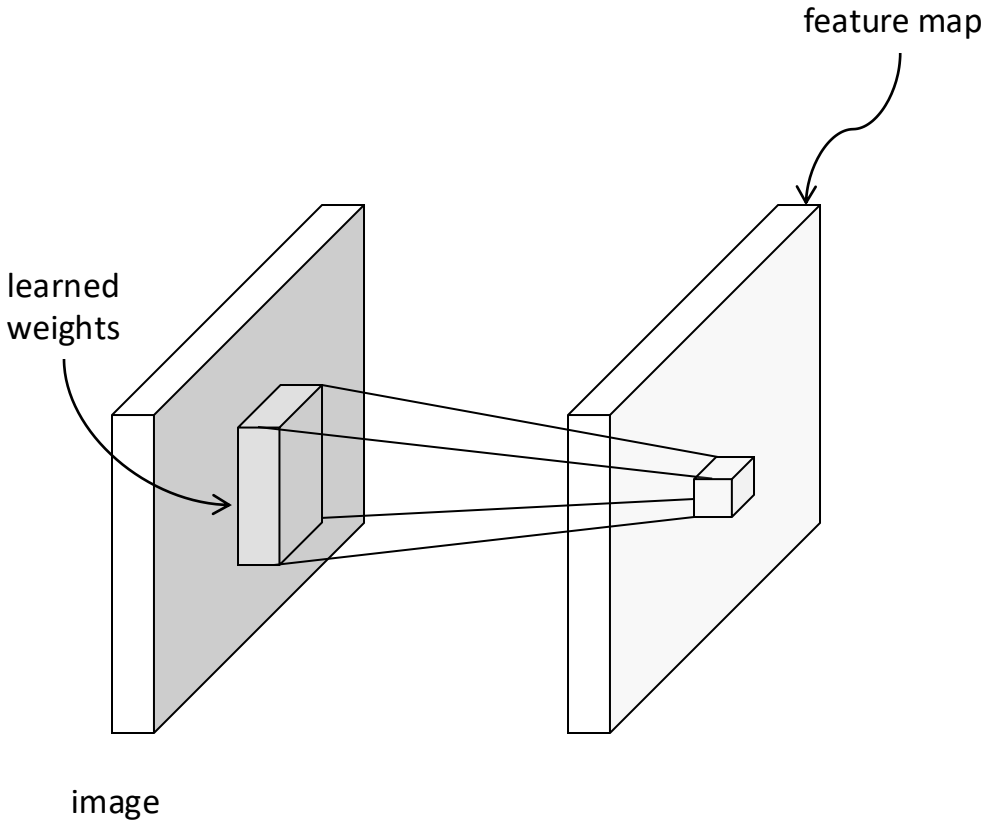


Output feature map resolution depends on *padding* and *stride*

With padding, spatial resolution remains the same if stride of 1 is used, is reduced by factor of $1/S$ if stride of S is used

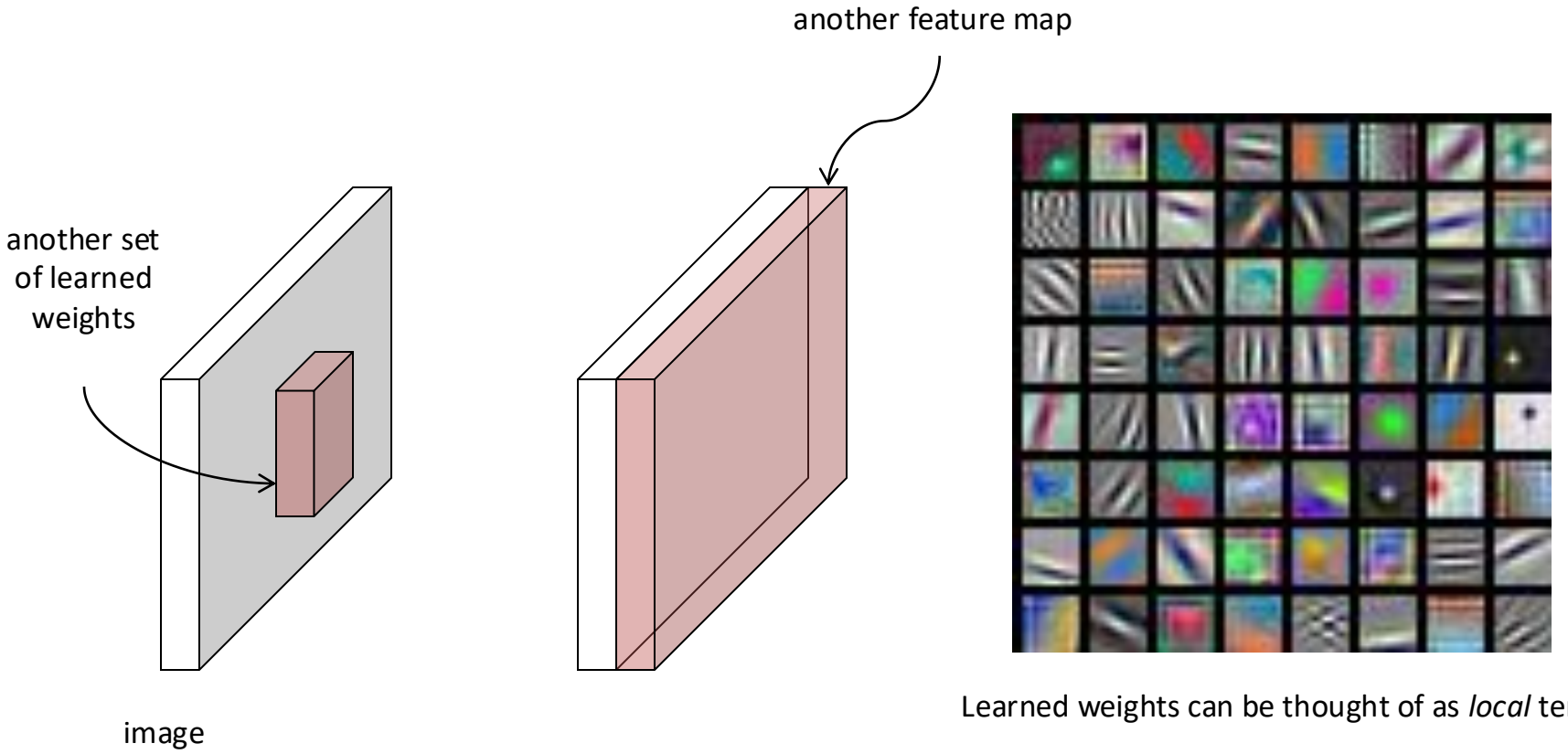
[Slide credit: S. Lazebnik]

Convolutional Architecture



[Slide credit: S. Lazebnik]

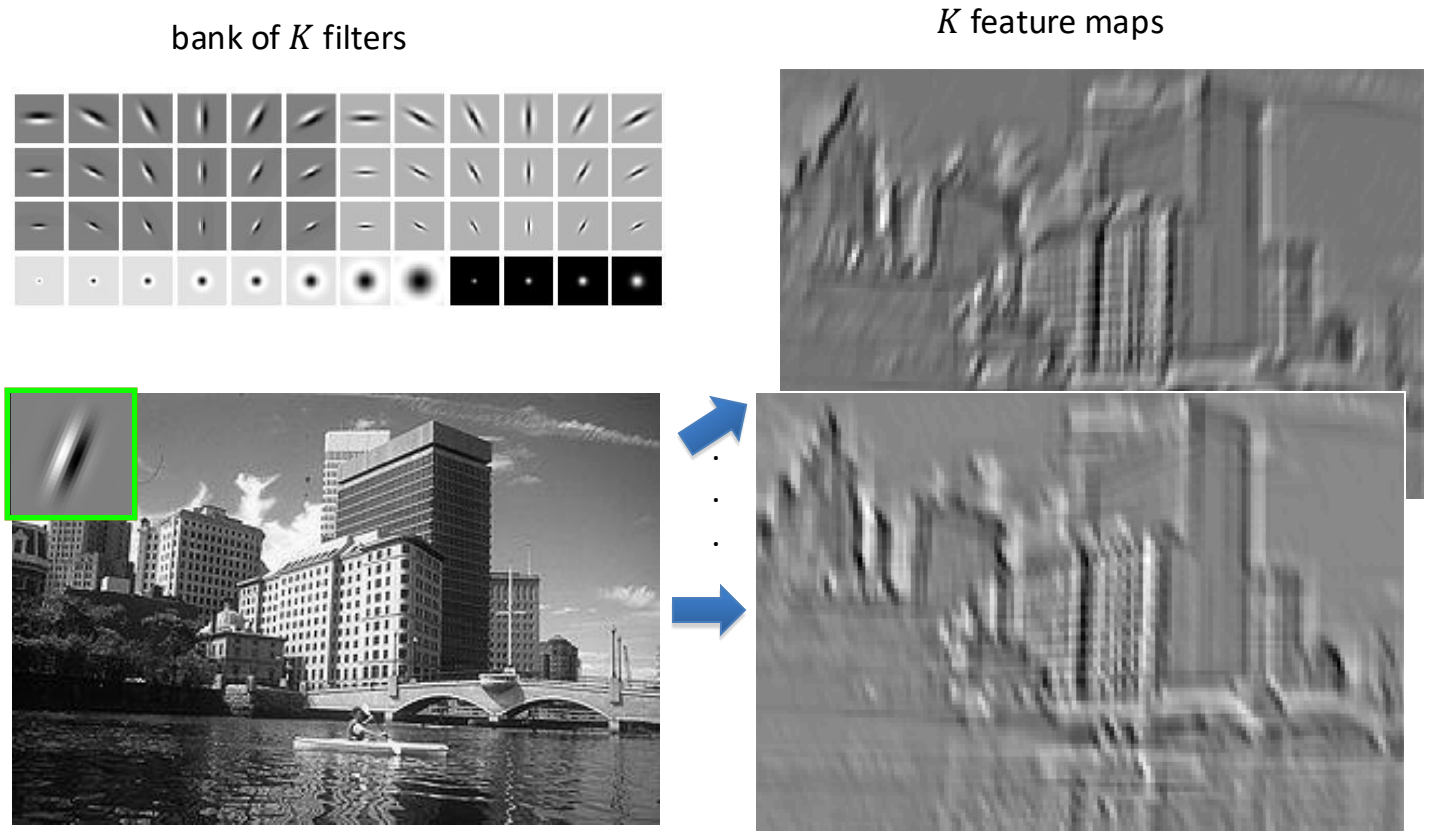
Convolutional Architecture



Learned weights can be thought of as *local* templates

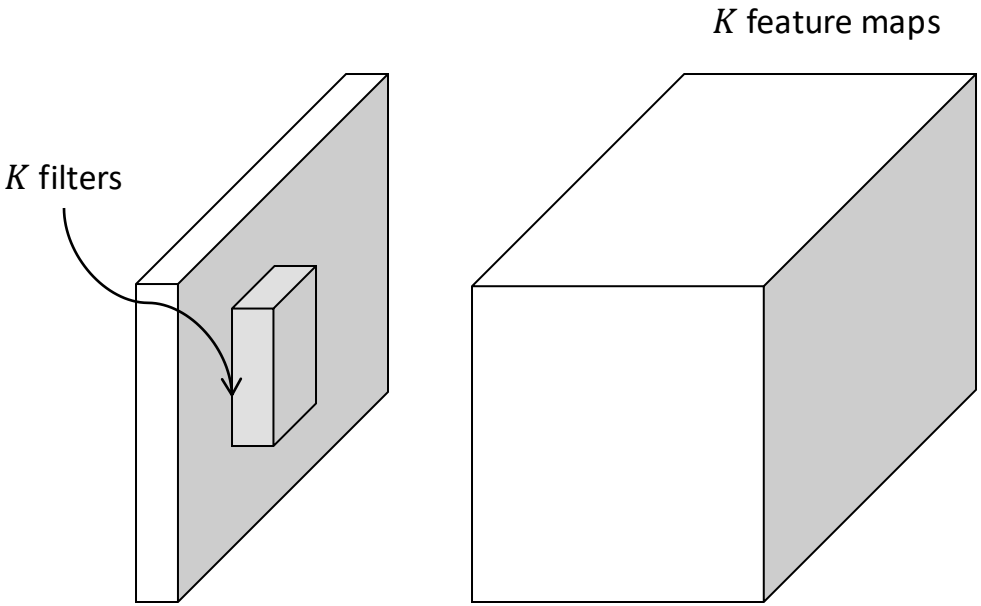
[Slide credit: S. Lazebnik]

Convolution and Traditional Feature Extraction



[Slide credit: S. Lazebnik]

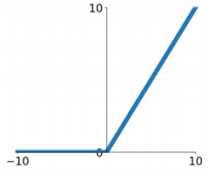
Elementwise Nonlinear Activation Function



Almost always directly followed by an activation function, e.g.:

Rectified Linear Unit (ReLU):

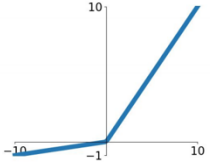
$$\max(0, x)$$



Some alternatives to ReLU:

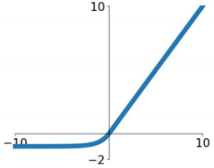
Leaky ReLU

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



ELU

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



[Slide credit: S. Lazebnik]

[Source: Stanford 231n]

Why Nonlinear?

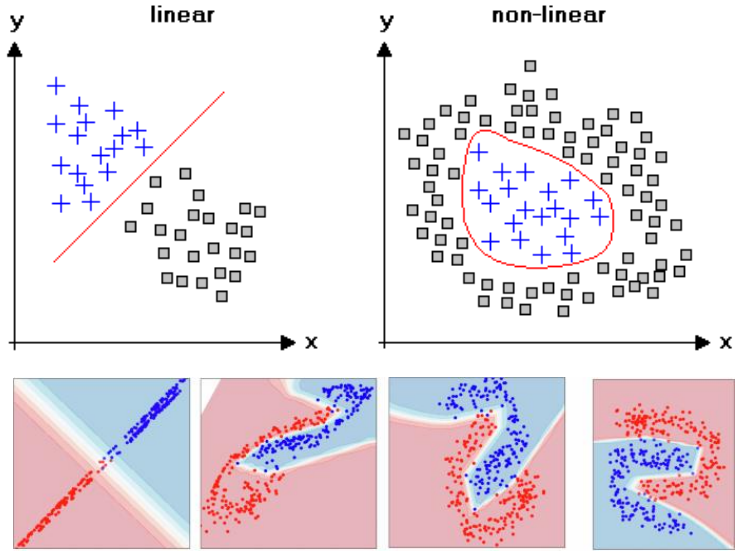
- Complex tasks require complex decision boundaries
- Stacking linear layers alone retains a **linear** decision boundary:

$$
\begin{aligned}
y &= L_n\left(\cdots L_2\left(L_1(x)\right)\right) \\
&= W_n \cdot \left(\ldots \cdot W_2 \cdot (W_1 \cdot x)\right) \\
&= \underbrace{W_n \ldots W_2 W_1} x \\
&= W_{\text{all}} x
\end{aligned}
$$

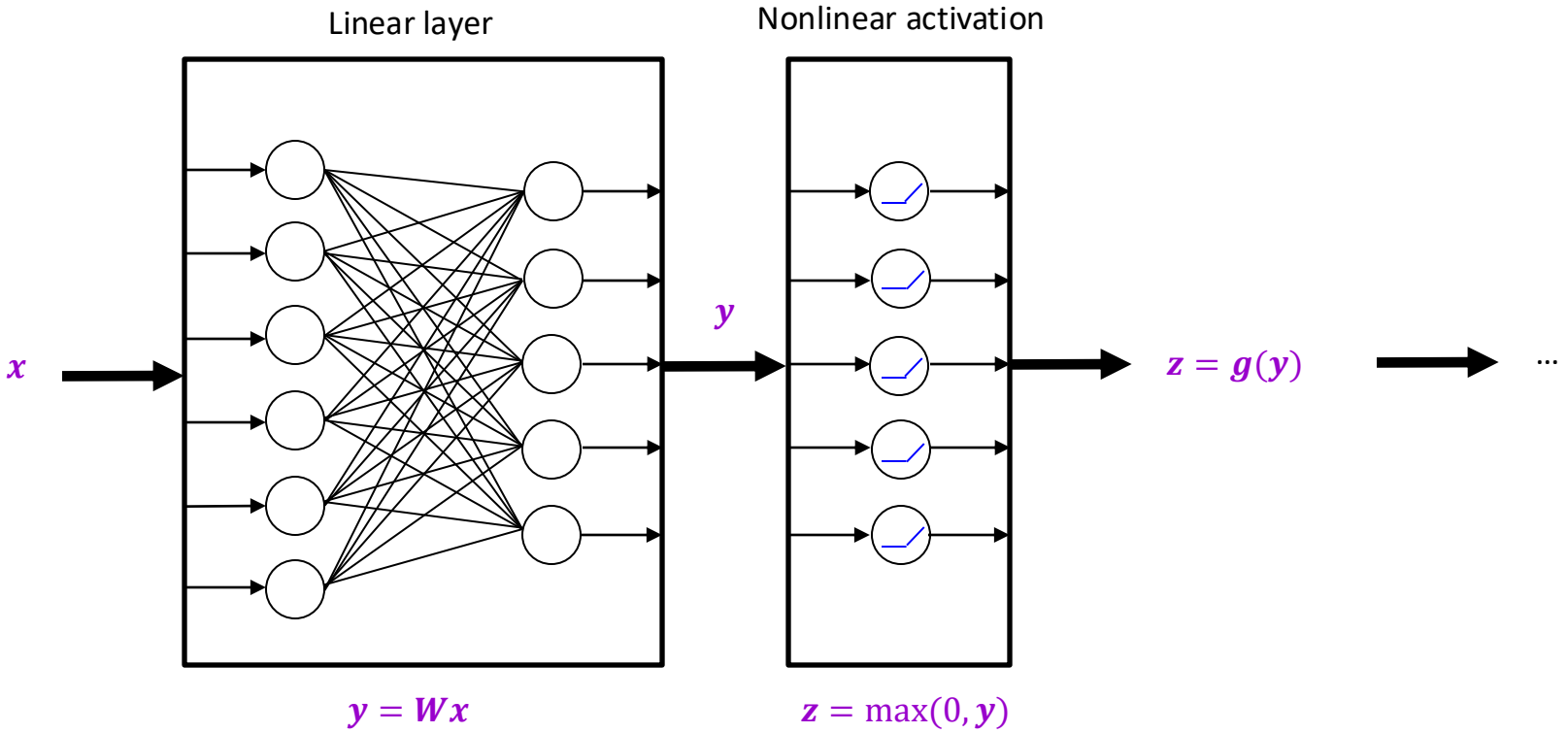
- Nonlinearities increase descriptive power and decision boundary complexity of neural networks

(Note: convolutions are linear layers)

Decision Boundaries



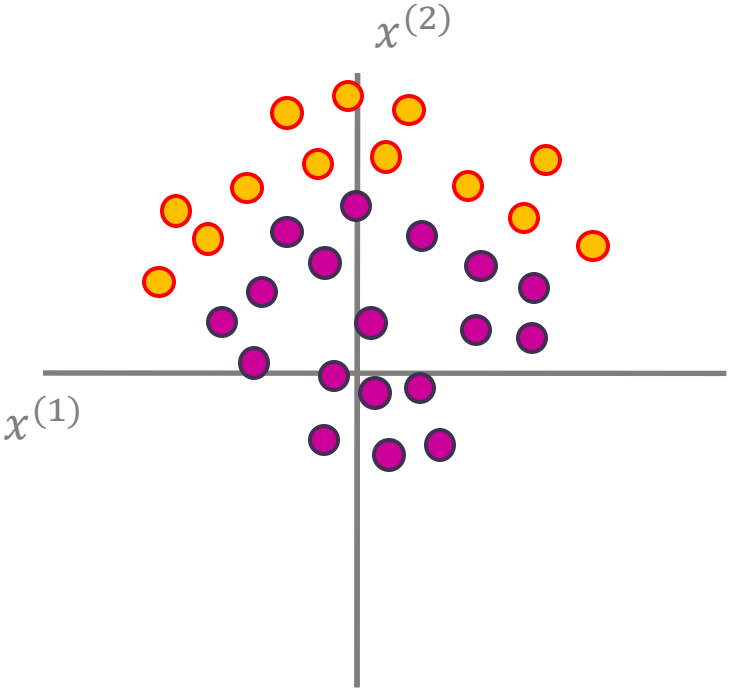
From linear classifiers to multi-layer networks



Stacking multiple layers of linear transformations and nonlinear activations increases the descriptive power of the network and decision boundary complexity with every layer.

The power of nonlinearities

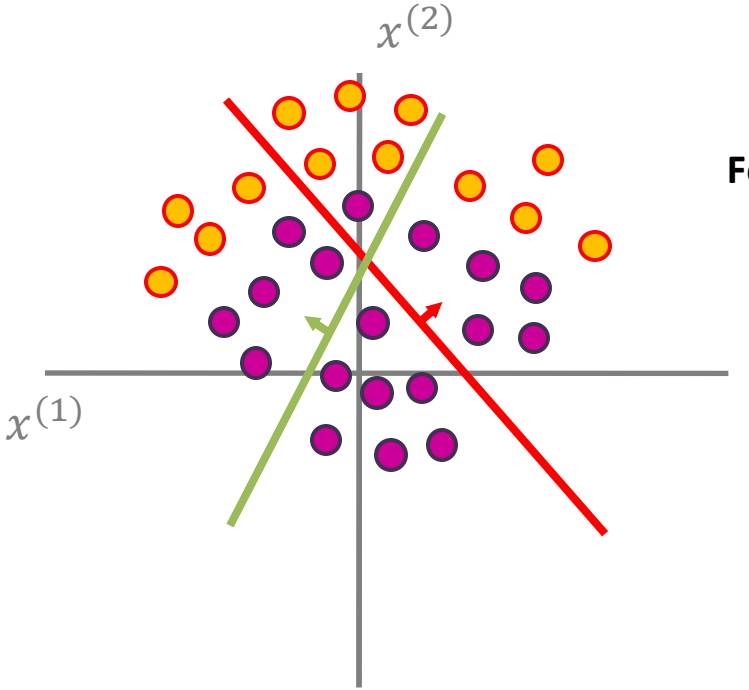
Points not linearly separable
in original space



[Slide credit: Justin Johnson]

The power of nonlinearities

Points not linearly separable in original space

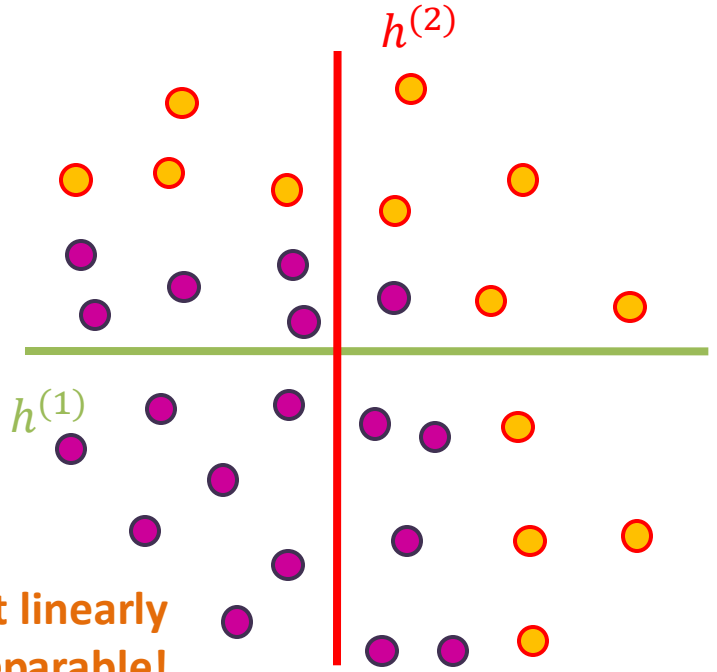


Consider a linear transform: $h = Wx + b$

Where x, h, b are 2-dimensional

Feature transform:

$h = Wx + b$



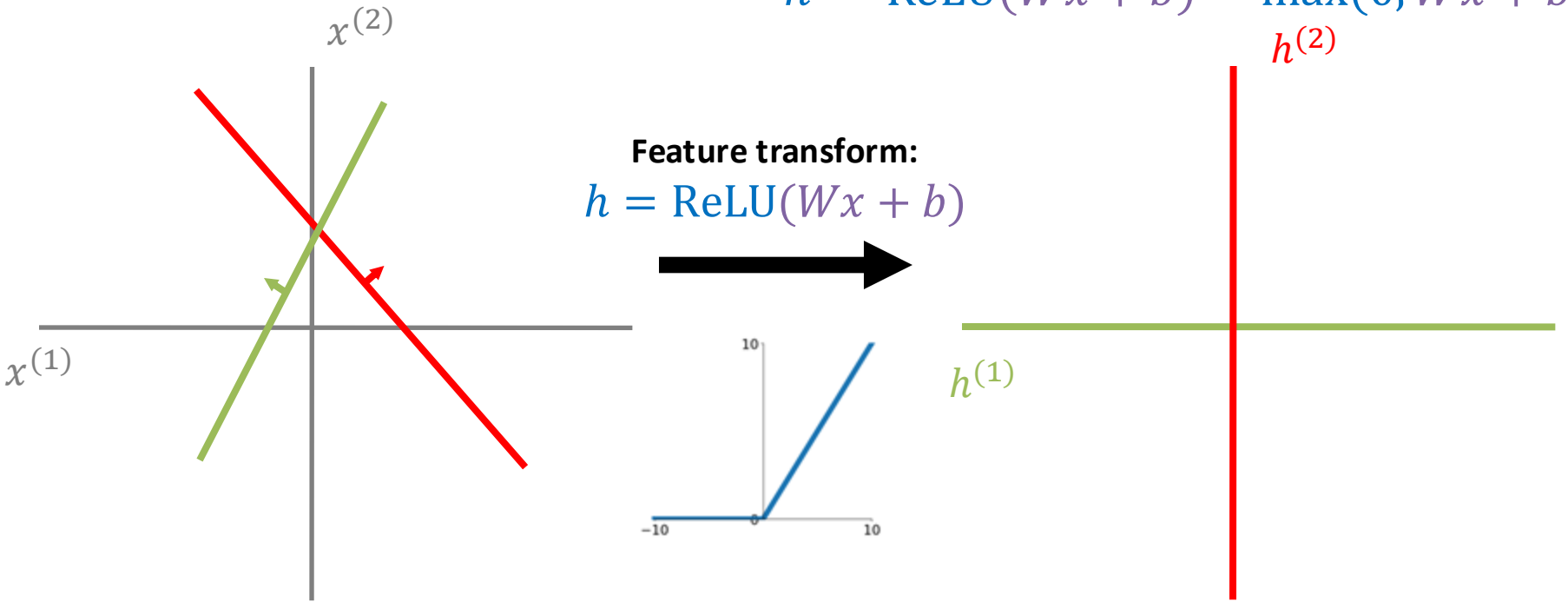
Still not linearly separable!

[Slide credit: Justin Johnson]

The power of nonlinearities

Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$

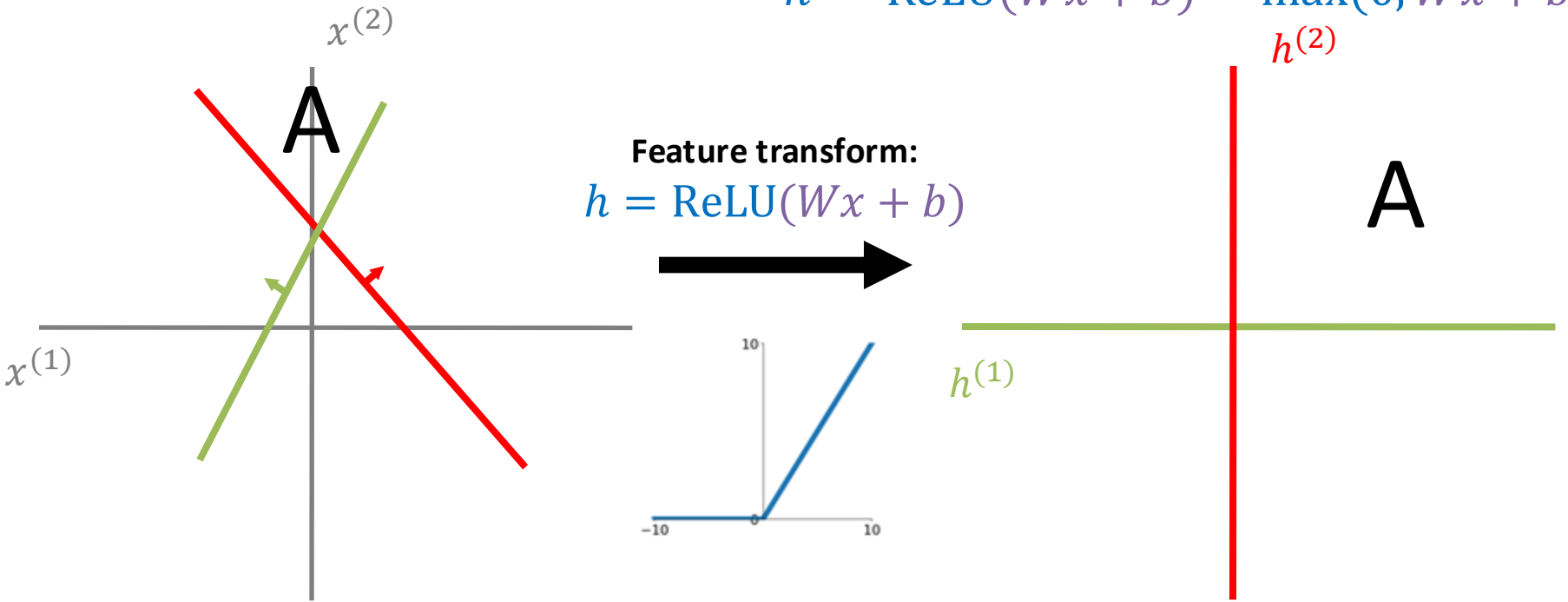


[Slide credit: Justin Johnson]

The power of nonlinearities

Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$

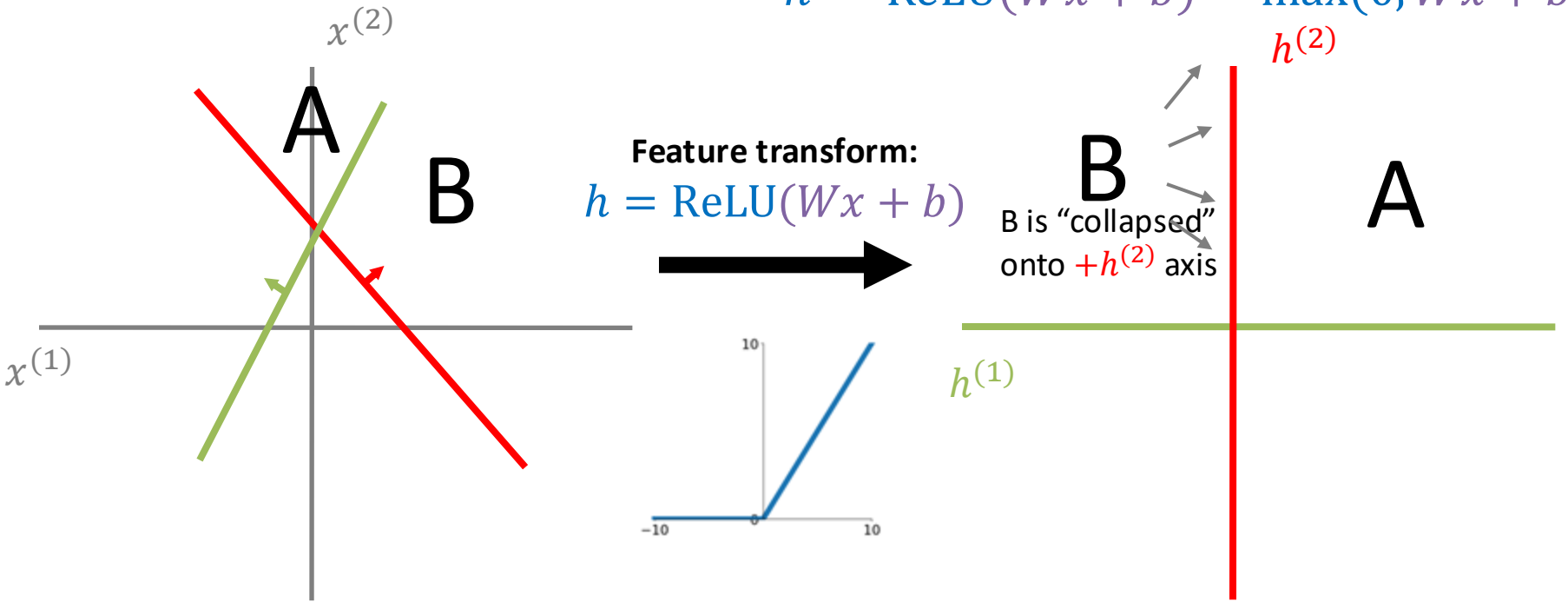


[Slide credit: Justin Johnson]

The power of nonlinearities

Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$

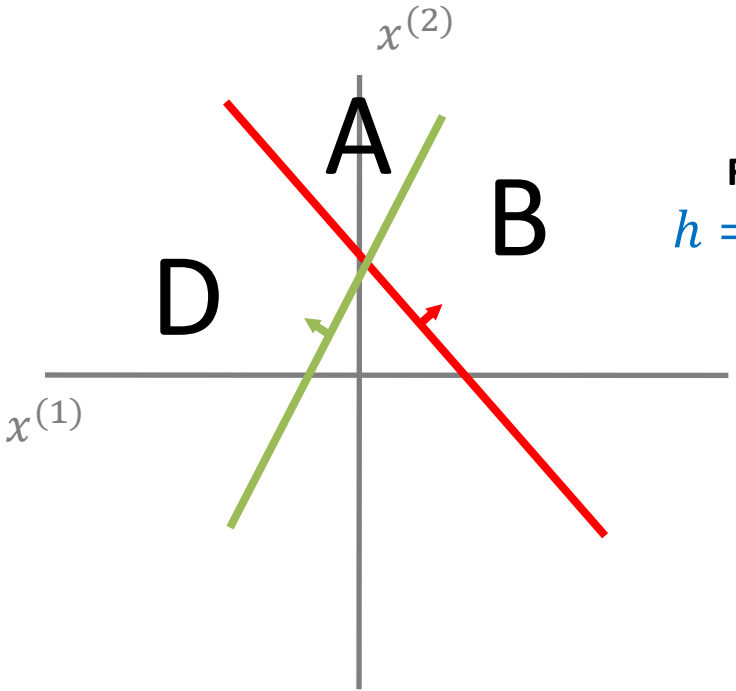


[Slide credit: Justin Johnson]

The power of nonlinearities

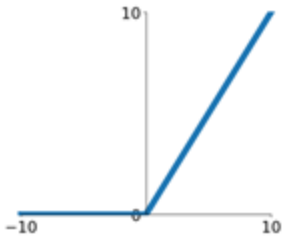
Let's add a nonlinearity:

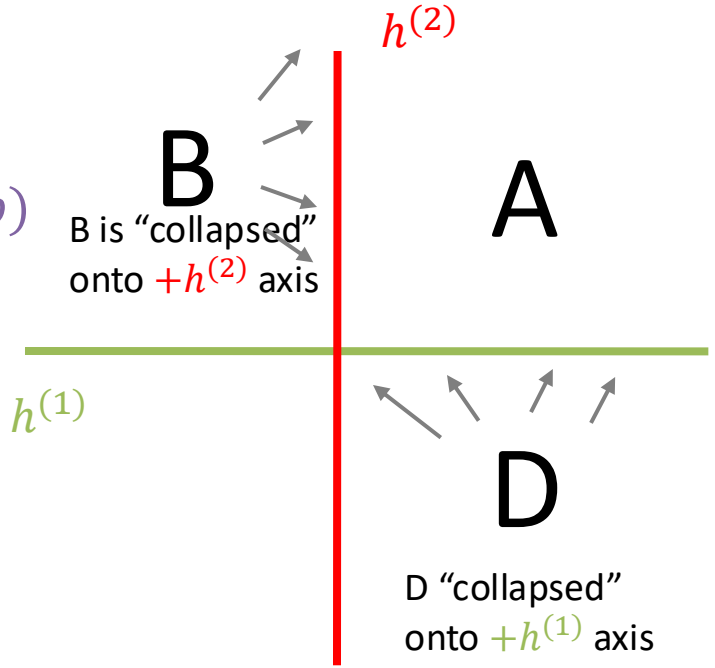
$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



Feature transform:

$$h = \text{ReLU}(Wx + b)$$



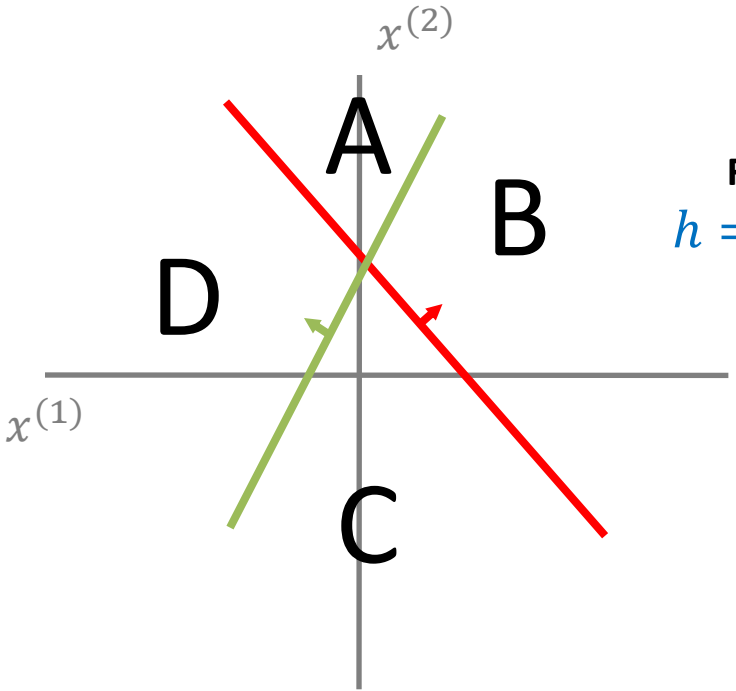


[Slide credit: Justin Johnson]

The power of nonlinearities

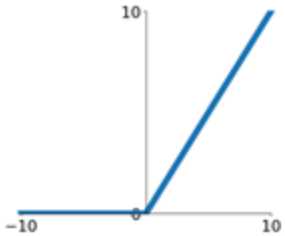
Let's add a nonlinearity:

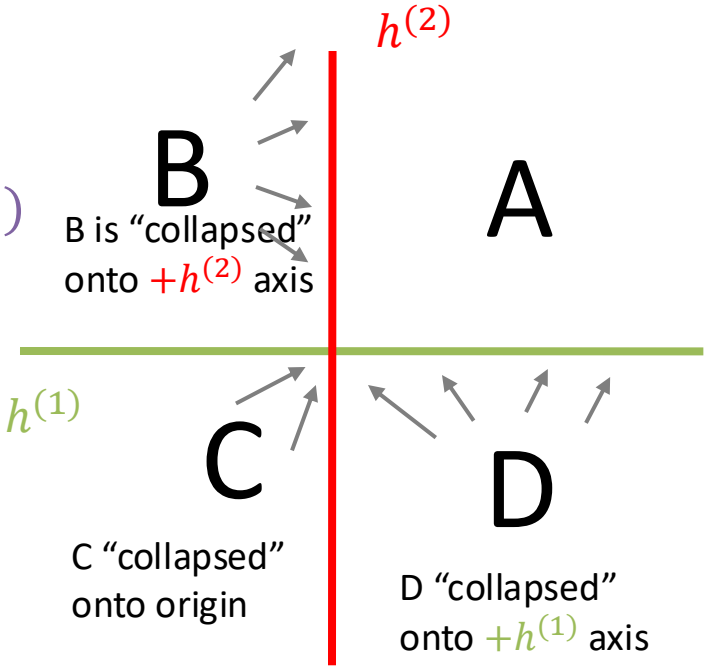
$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



Feature transform:

$$h = \text{ReLU}(Wx + b)$$

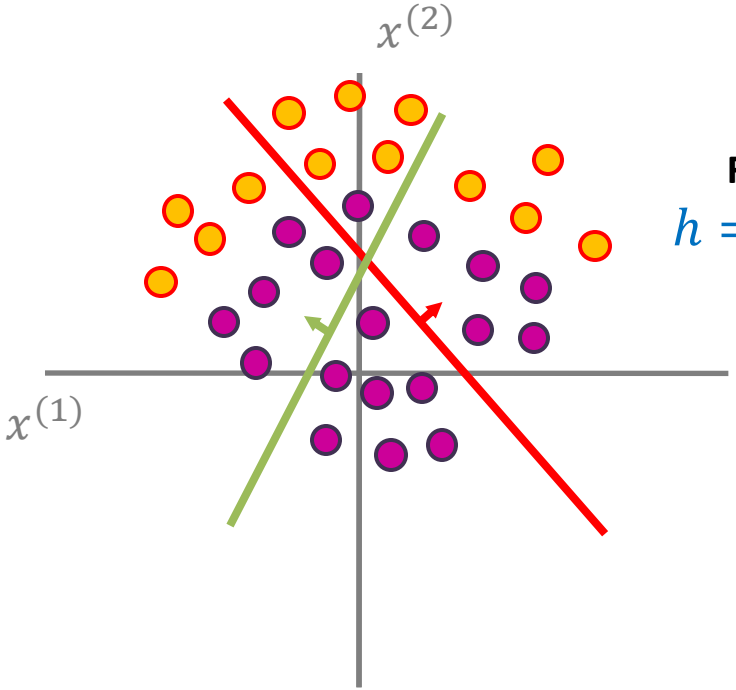




[Slide credit: Justin Johnson]

The power of nonlinearities

Points not linearly separable in original space

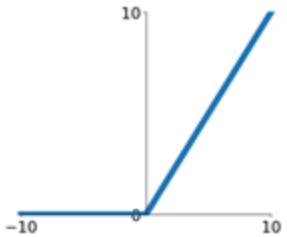


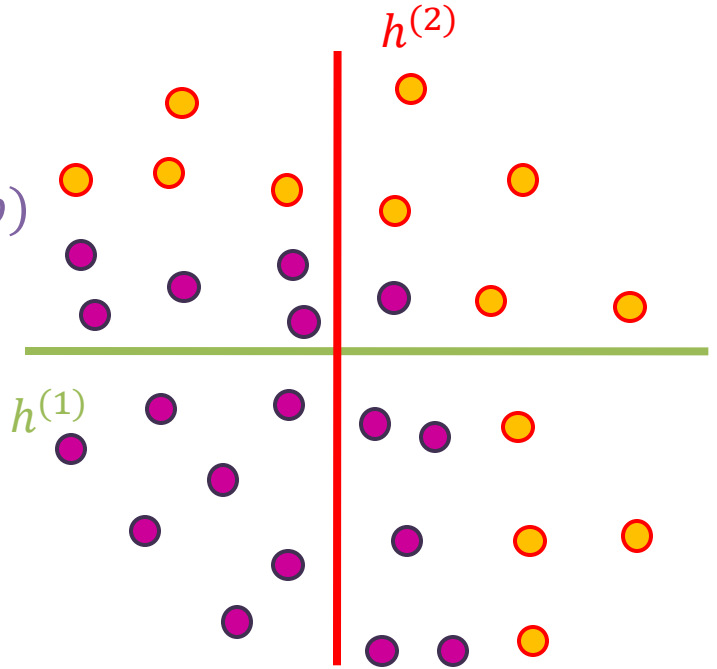
Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$

Feature transform:

$$h = \text{ReLU}(Wx + b)$$

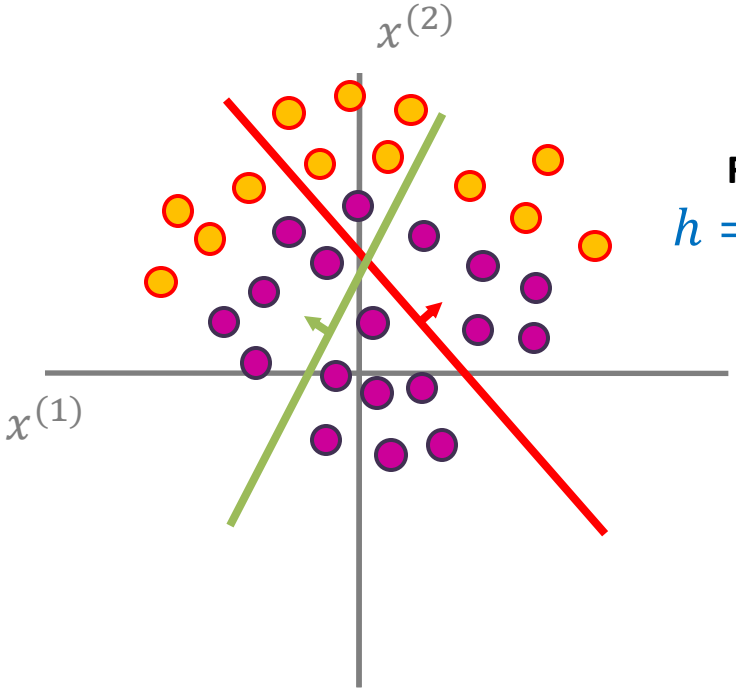




[Slide credit: Justin Johnson]

The power of nonlinearities

Points not linearly separable in original space

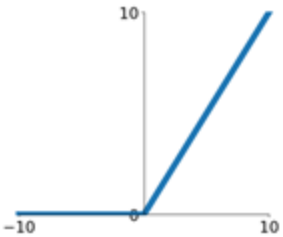


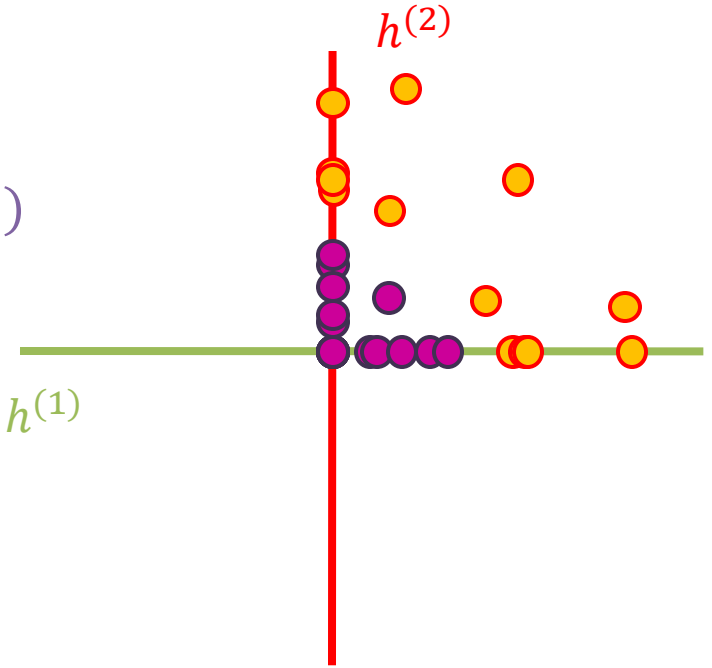
Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$

Feature transform:

$$h = \text{ReLU}(Wx + b)$$

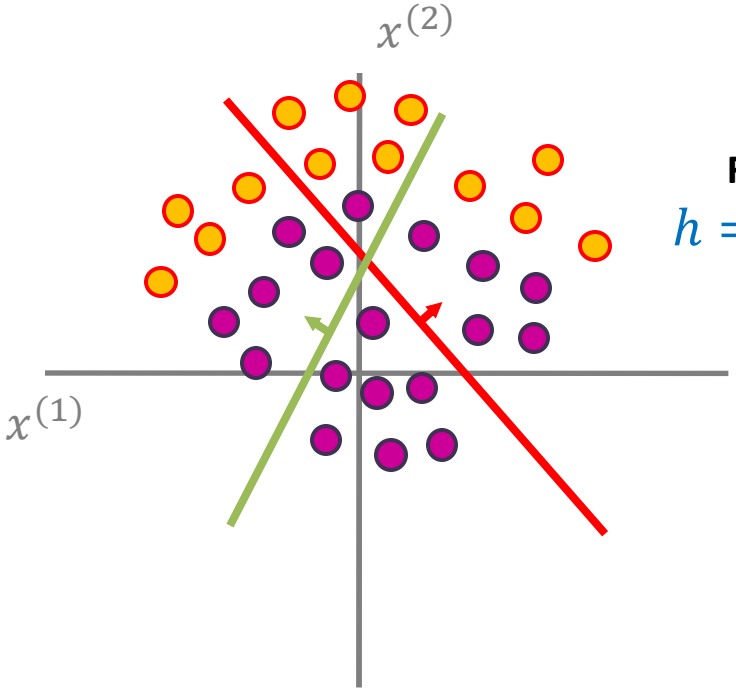




[Slide credit: Justin Johnson]

The power of nonlinearities

Points not linearly separable in original space

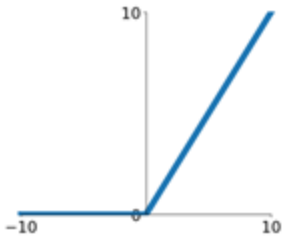


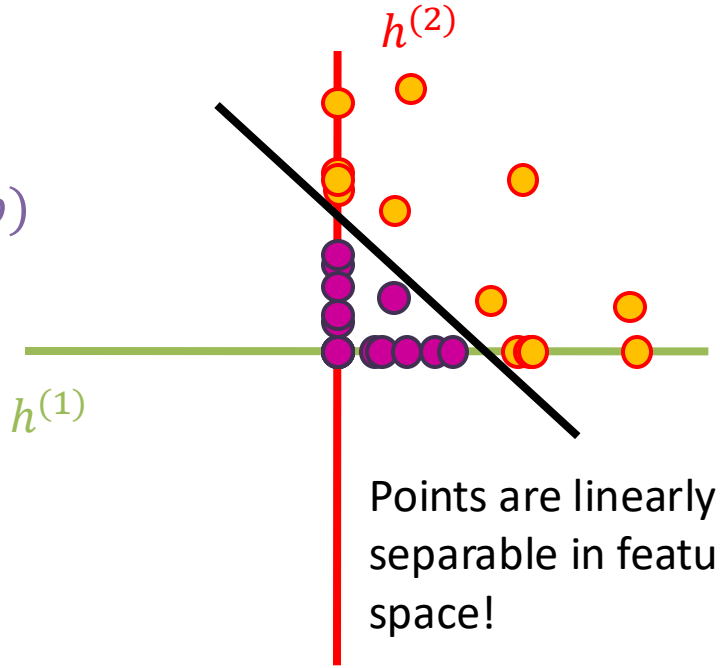
Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$

Feature transform:

$$h = \text{ReLU}(Wx + b)$$



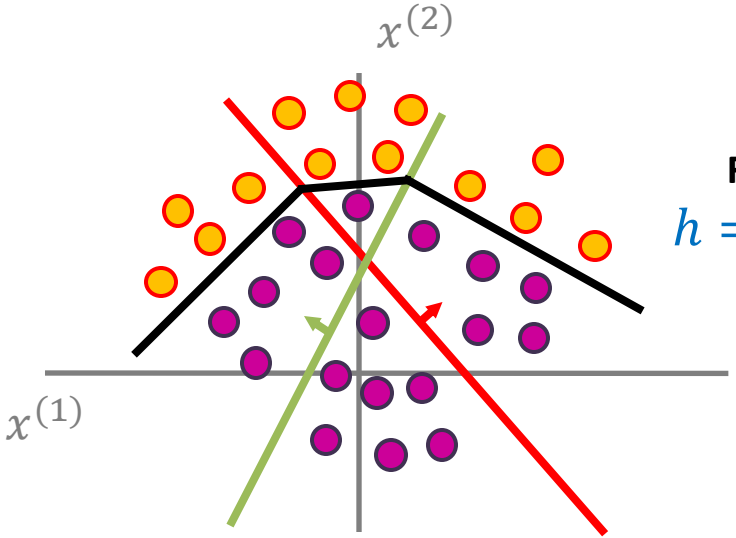


Points are linearly separable in feature space!

[Slide credit: Justin Johnson]

The power of nonlinearities

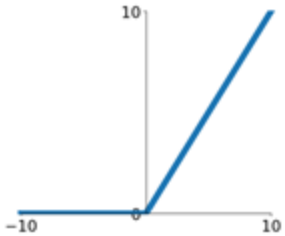
Points not linearly separable in original space



Linear classifier in feature space gives nonlinear classifier in original space

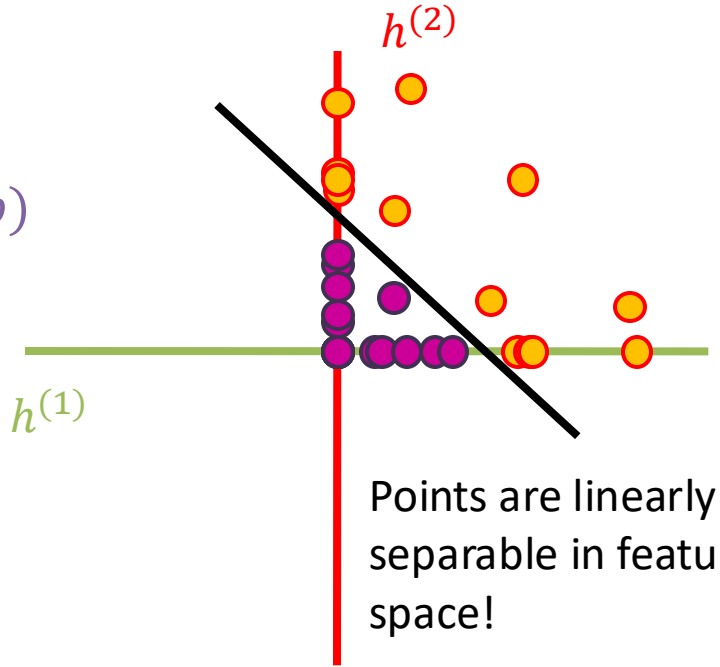
Feature transform:

$$h = \text{ReLU}(Wx + b)$$



Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



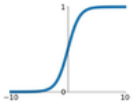
Points are linearly separable in feature space!

[Slide credit: Justin Johnson]

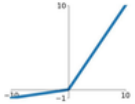
Common Activation Functions

Activation Functions (Most common)

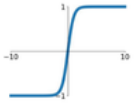
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



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

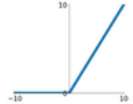


tanh
$\tanh(x)$

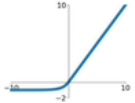


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

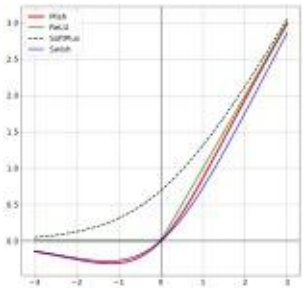
ReLU
$\max(0, x)$

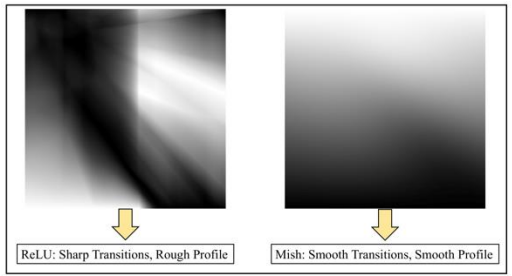


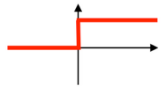
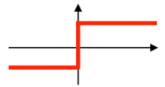
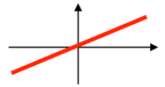
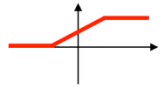
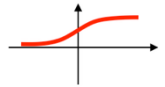
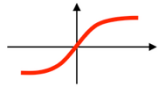
ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Mish
$x \tanh(\text{softplus}(x))$





Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1+e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer NN	

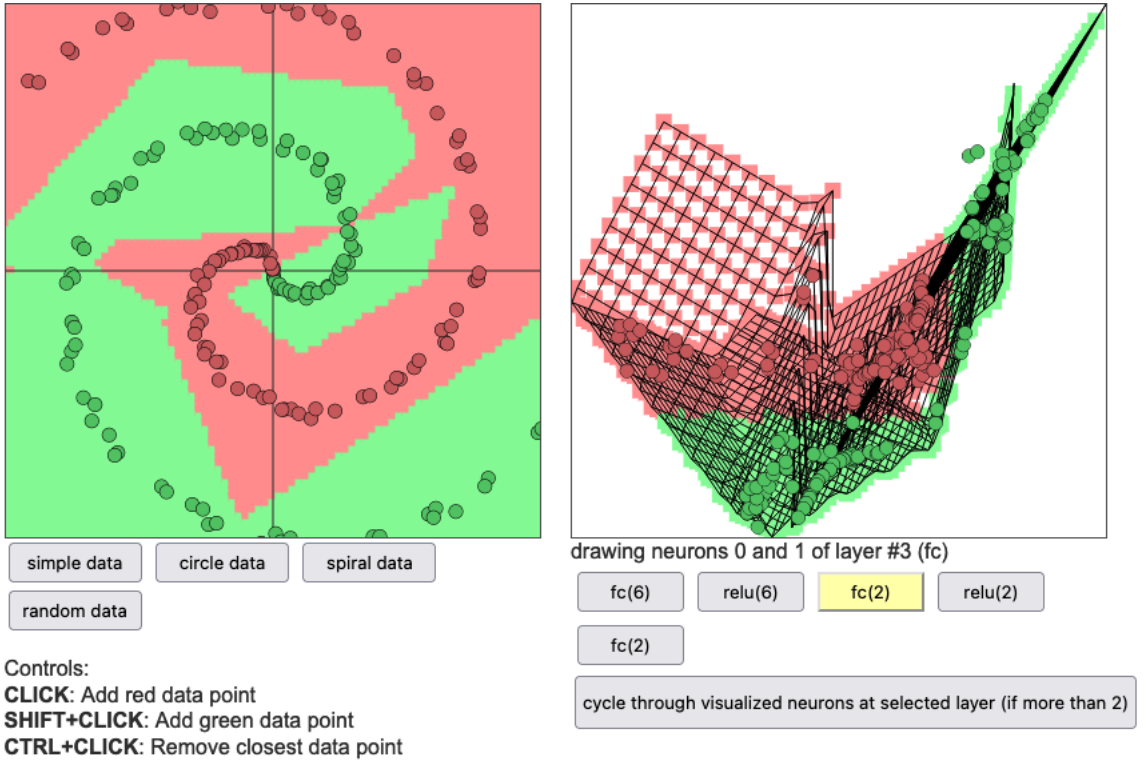
Loss landscape can be important for network training!

[Misra, Mish: Self Regularized Non-Monotonic Activation Function, BMVC 2020]

Common Activation Functions

- Activation functions influence the loss landscape and decision boundaries.

- Online Demo:
  by Andrej Karpathy

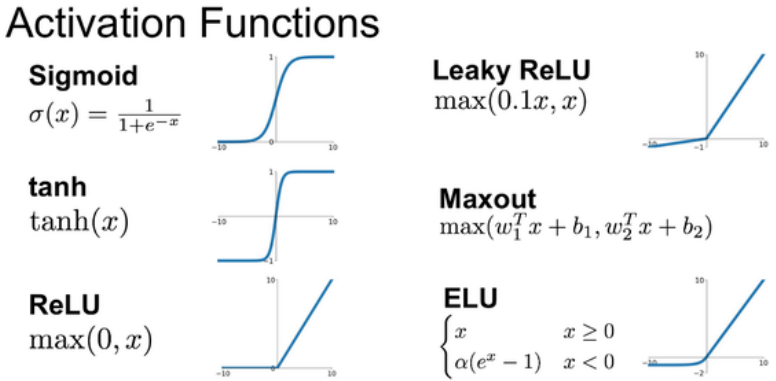


Normalization Layer

- Normalization avoids weights and internal values to converge towards infinity
- Normalization facilitates that the value range of network layers is in a suitable range for the activation function

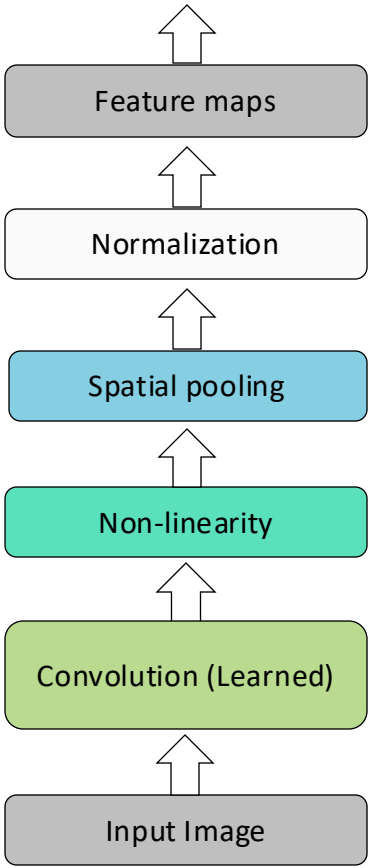
- In-Layer Normalization
  - Batch Normalization
  - Layer Normalization
  - Instance Normalization
  - Group Normalization

- Data Normalization (Feature Scaling)
  - Min-Max Normalization
  - Z-Score Standardization
  - Log Transformation



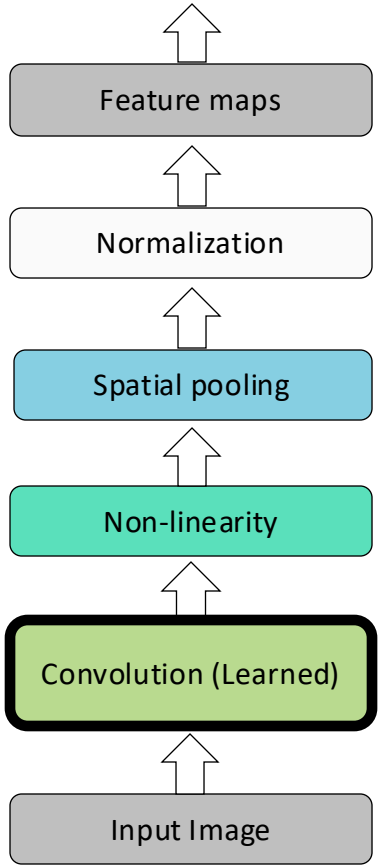
[Slide credit: Justin Johnson]

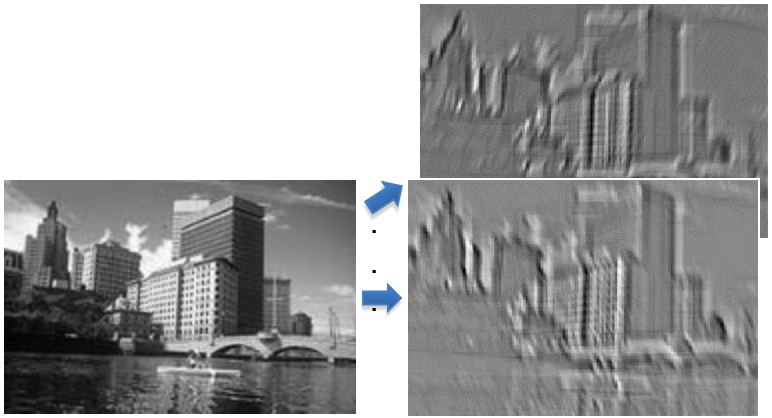
Convolutional Neural Networks



[Slide credit: S. Lazebnik]

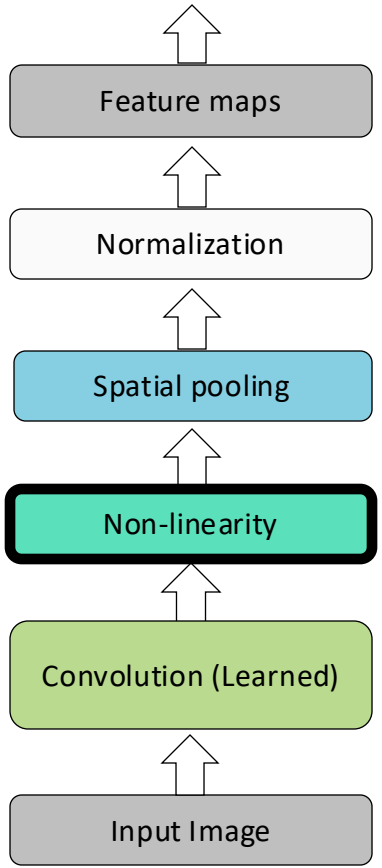
Convolutional Neural Networks



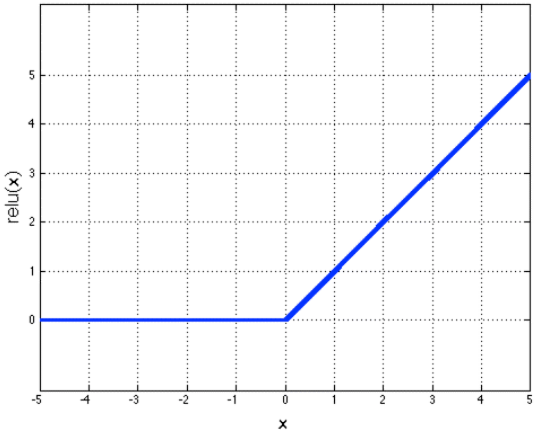


[Slide credit: S. Lazebnik]

Convolutional Neural Networks

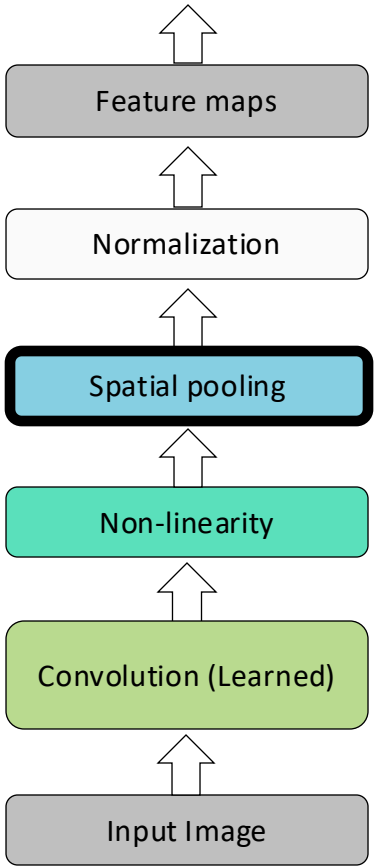


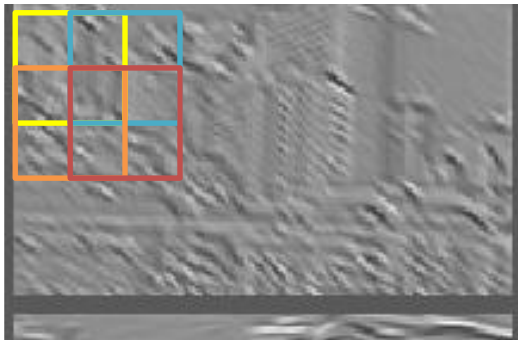
Rectified Linear Unit (ReLU)



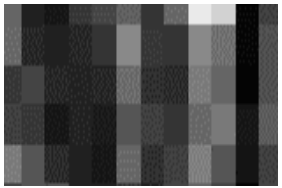
[Slide credit: S. Lazebnik]

Convolutional Neural Networks





Max pooling

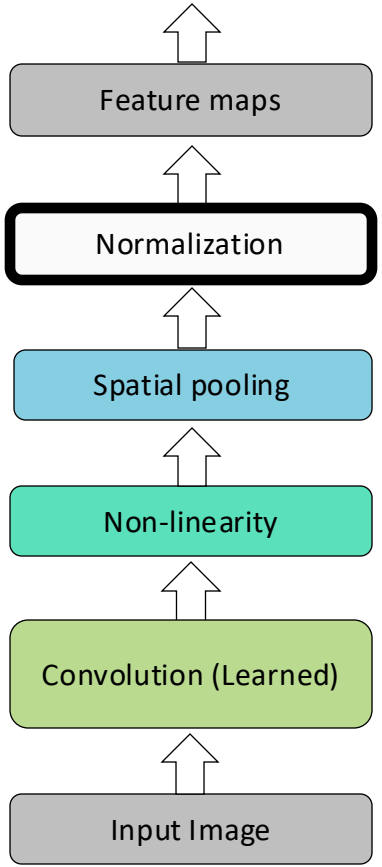


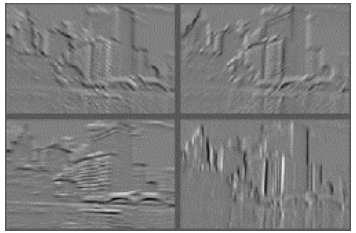
Max-pooling: a non-linear down-sampling

Provide *local invariance*

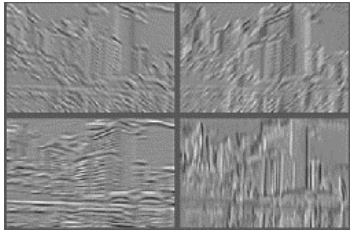
[Slide credit: S. Lazebnik]

Convolutional Neural Networks





Feature Maps

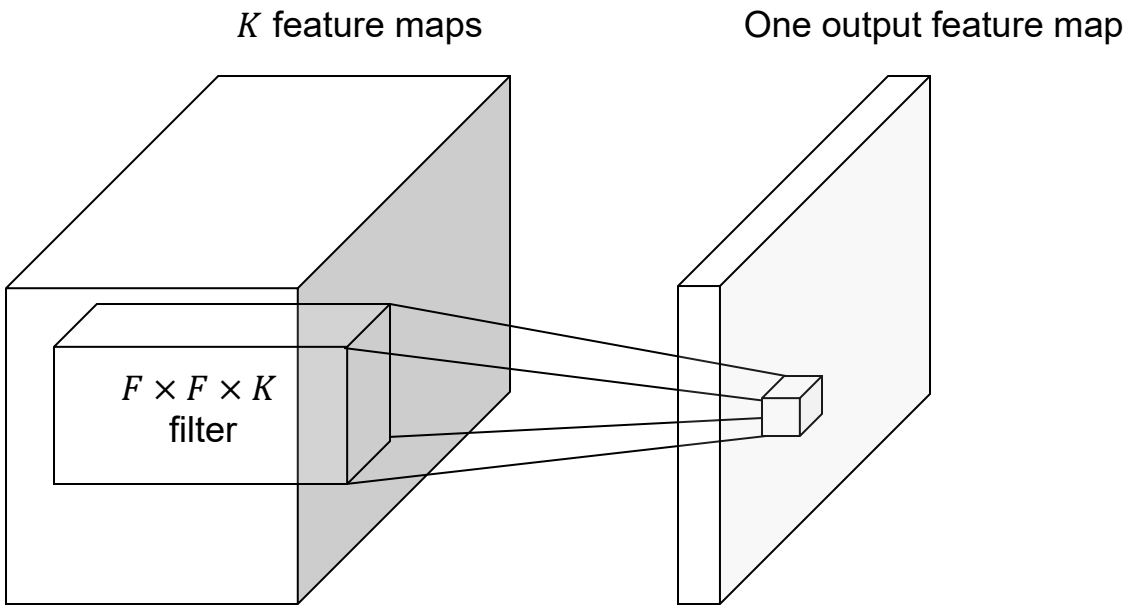


Feature Maps
After Contrast
Normalization

[Slide credit: S. Lazebnik]

Three-dimensional Convolutions

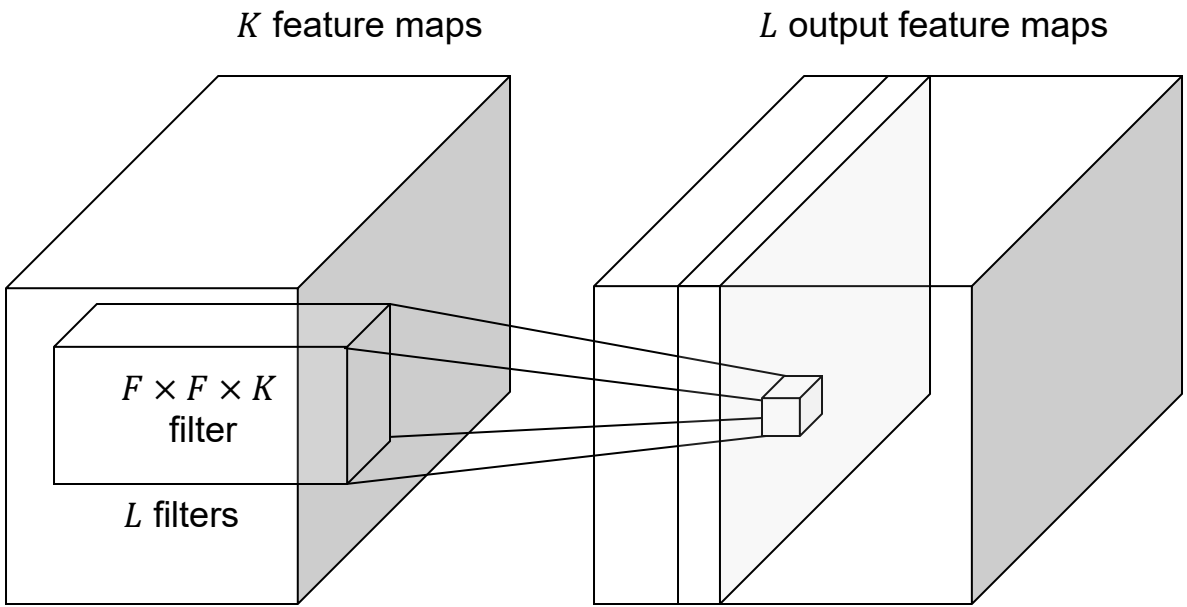
- What if the *input* to a convolutional layer is a stack of K feature maps?



[Slide credit: S. Lazebnik]

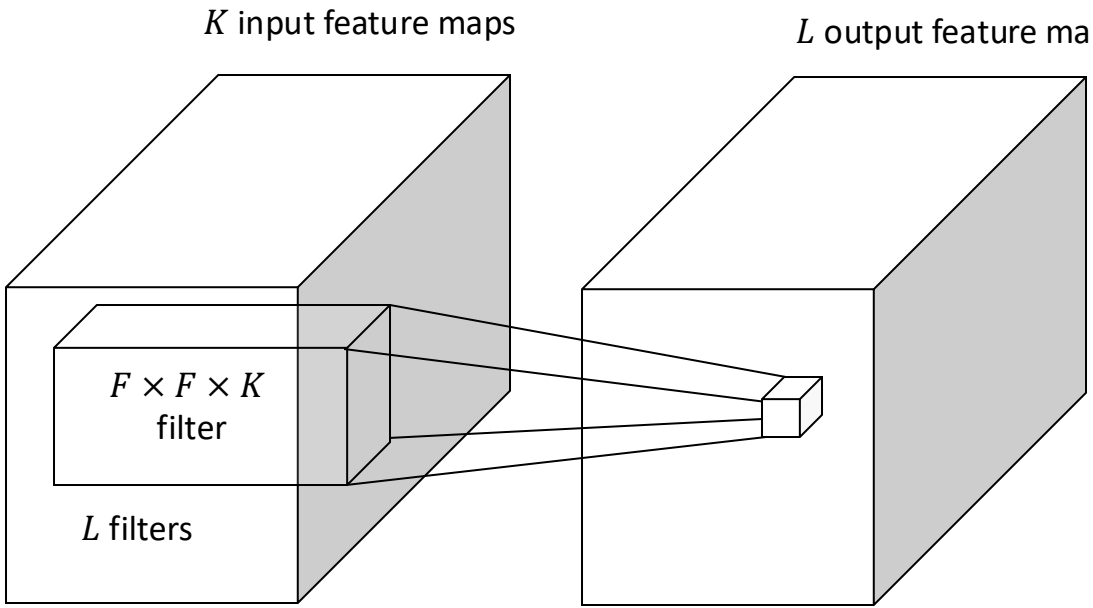
Three-dimensional Convolutions

- What if the *input* to a convolutional layer is a stack of K feature maps?



[Slide credit: S. Lazebnik]

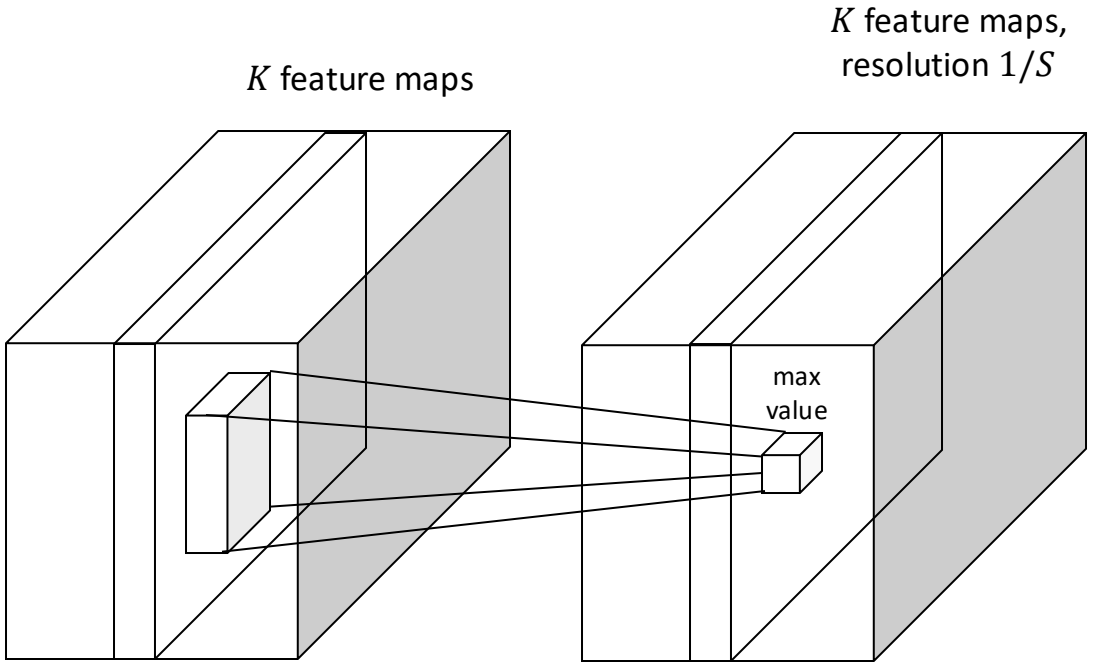
Convolutional Layer: Computational Cost



- Assuming the input feature maps have spatial resolution $H \times W$, how many operations are needed to compute the output feature volume?
  - F^2KLHW

[Slide credit: S. Lazebnik]

Convolutional Neural Networks



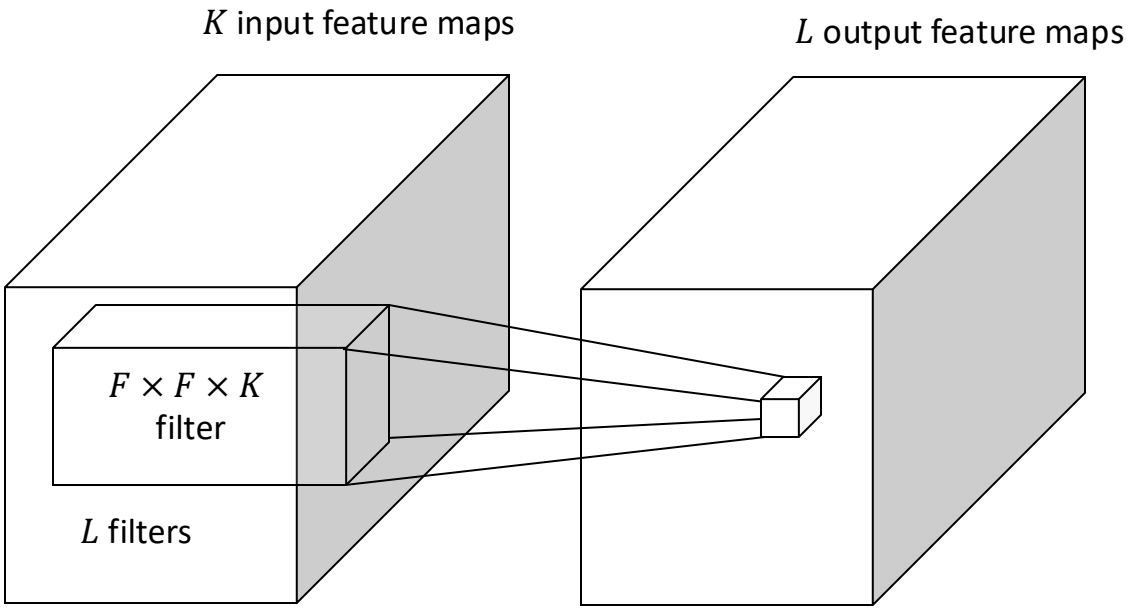
$F \times F$ pooling window, stride S

Backward pass: upstream gradient is passed back only to the unit with max value

Example: $F = 2$ or 3, $S = 2$

[Slide credit: S. Lazebnik]

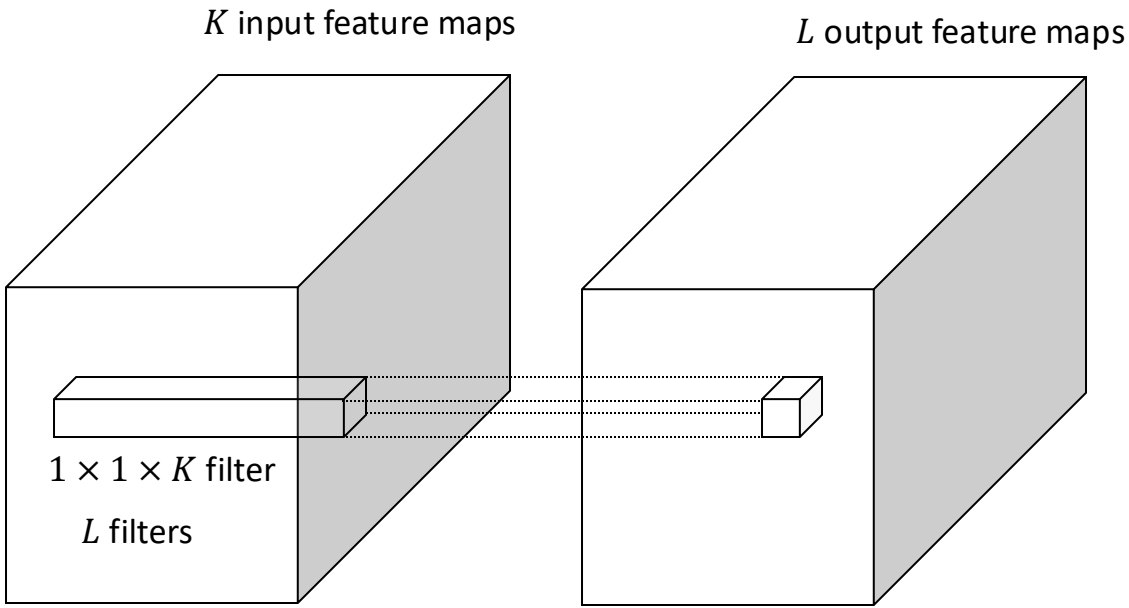
1x1 Convolutional Layer



What if we make $F = 1$?

[Slide credit: S. Lazebnik]

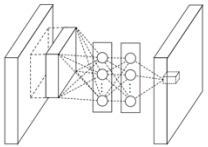
1x1 Convolutional Layer



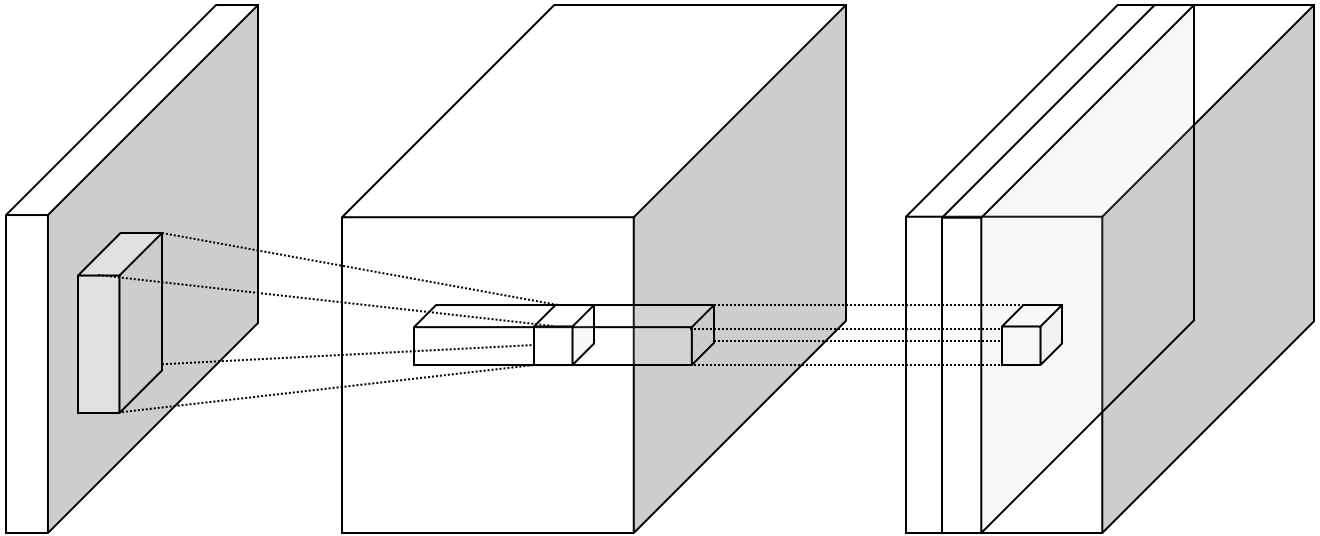
What if we make $F = 1$?

[Slide credit: S. Lazebnik]

1x1 Convolutional Layer



"network in network"

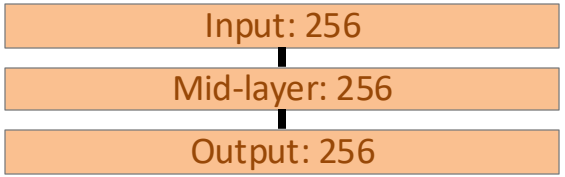


[Slide credit: S. Lazebnik]

[M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014]

Why 1x1 convolutions?

- **Option 1:** 3×3 conv layer with 256 channels at input and output
  - $256 \times 256 \times 3 \times 3$
  - Total $\approx$ $600{,}000$ weights & operations (per location)



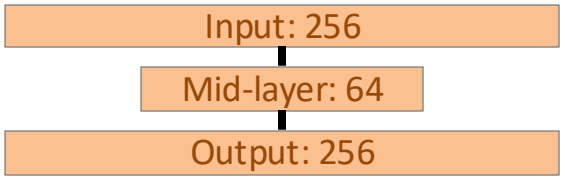
- **Option 2: "bottleneck module"**

  1×1 conv layer, $256 \rightarrow 64$ channels

  3×3 conv layer, $64 \rightarrow 64$ channels

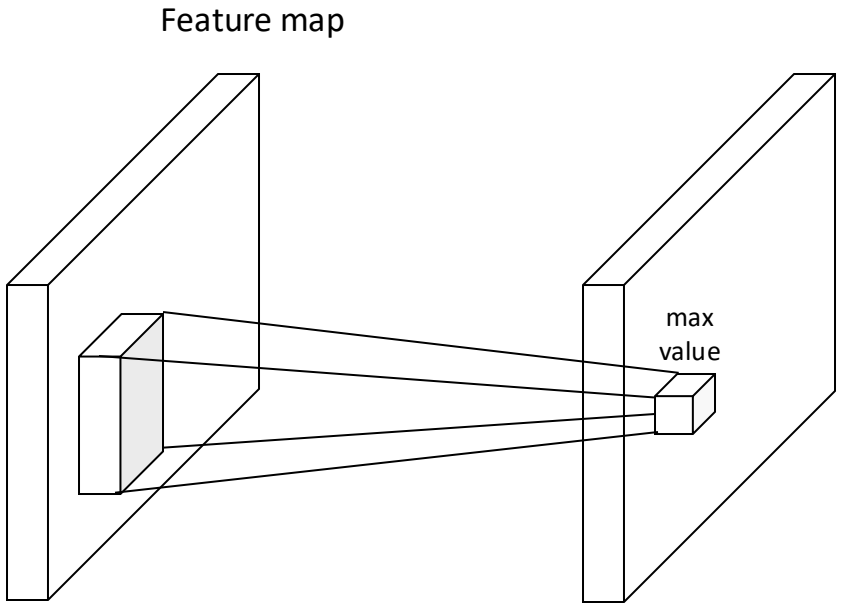
  1×1 conv layer, $64 \rightarrow 256$ channels

  - $256 \times 64 \times 1 \times 1 \approx 16{,}000$
  - $64 \times 64 \times 3 \times 3 \approx 36{,}000$
  - $64 \times 256 \times 1 \times 1 \approx 16{,}000$
  - Total $\approx$ $70{,}000$ weights & operations



[Slide credit: S. Lazebnik]

Max Pooling Layer



$F \times F$ pooling window,
stride S

Usually: $F = 2$ or 3, $S = 2$

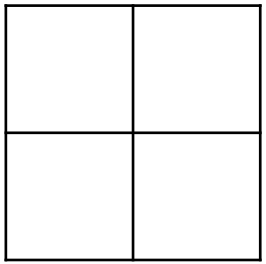
[Slide credit: S. Lazebnik]

Max Pooling: Example

Single channel

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Max pooling with 2×2 kernel size and stride 2



[Slide credit: S. Lazebnik]

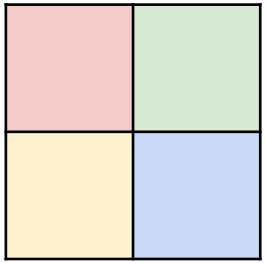
[Source: J. Johnson]

Max Pooling: Example

Single channel

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Max pooling with 2×2 kernel size and stride 2



[Slide credit: S. Lazebnik]

[Source: J. Johnson]

Max Pooling: Example

Single channel

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

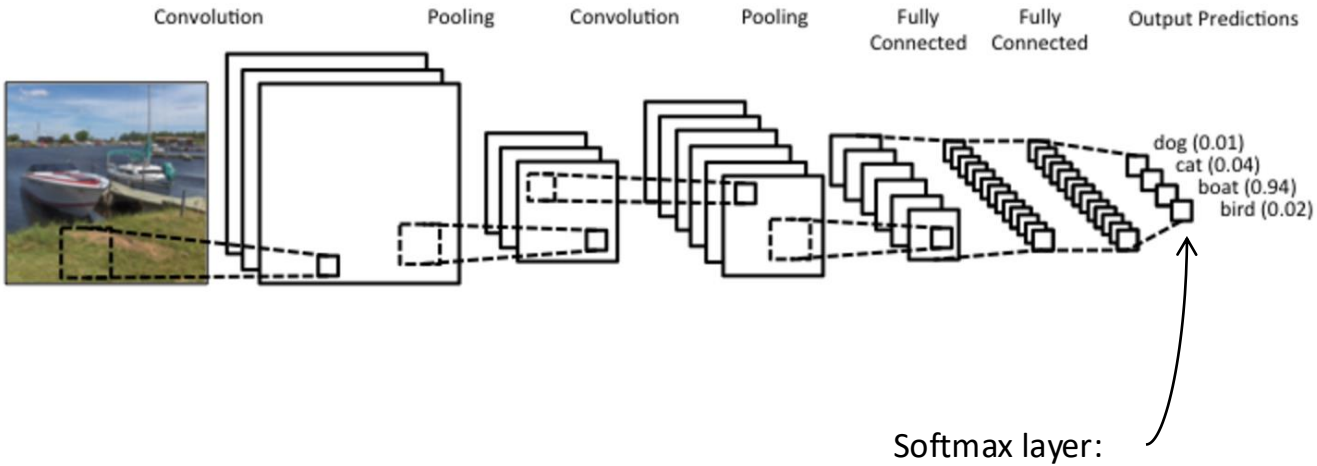
Max pooling with 2×2 kernel size and stride 2

6	8
3	4

[Slide credit: S. Lazebnik]

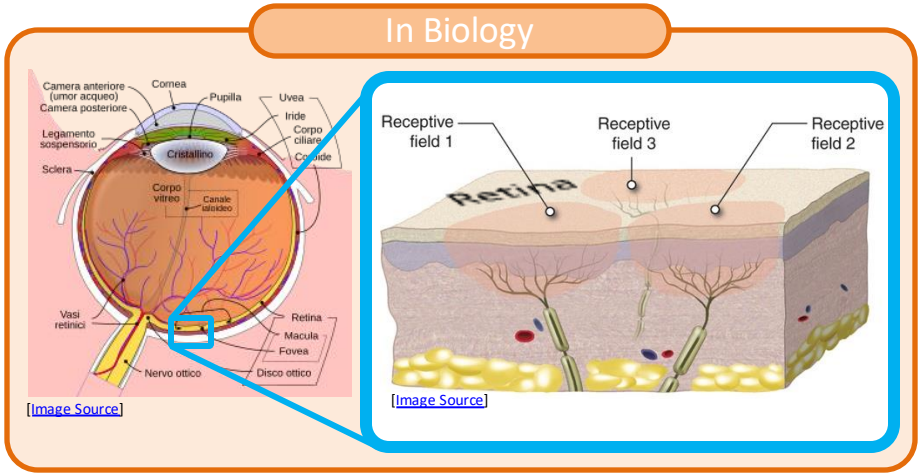
[Source: J. Johnson]

Simplified CNN Pipeline

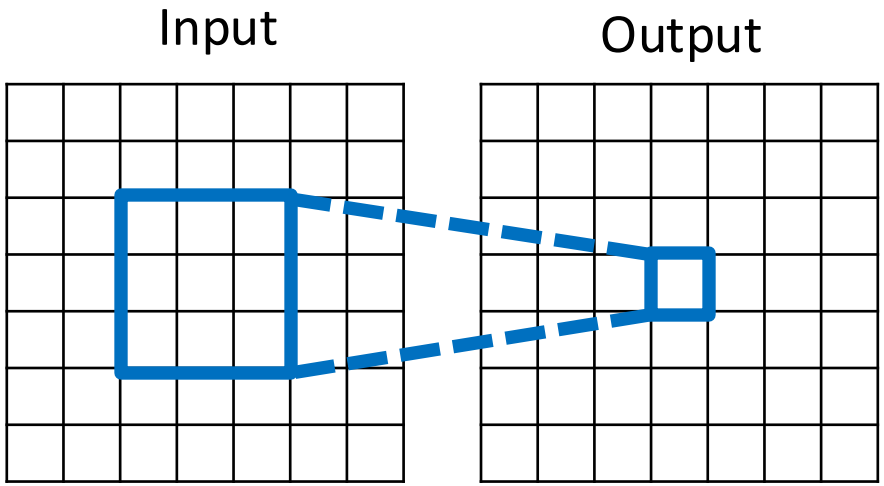


[Slide credit: S. Lazebnik]

Receptive Field



3x3 convolutions, stride 1



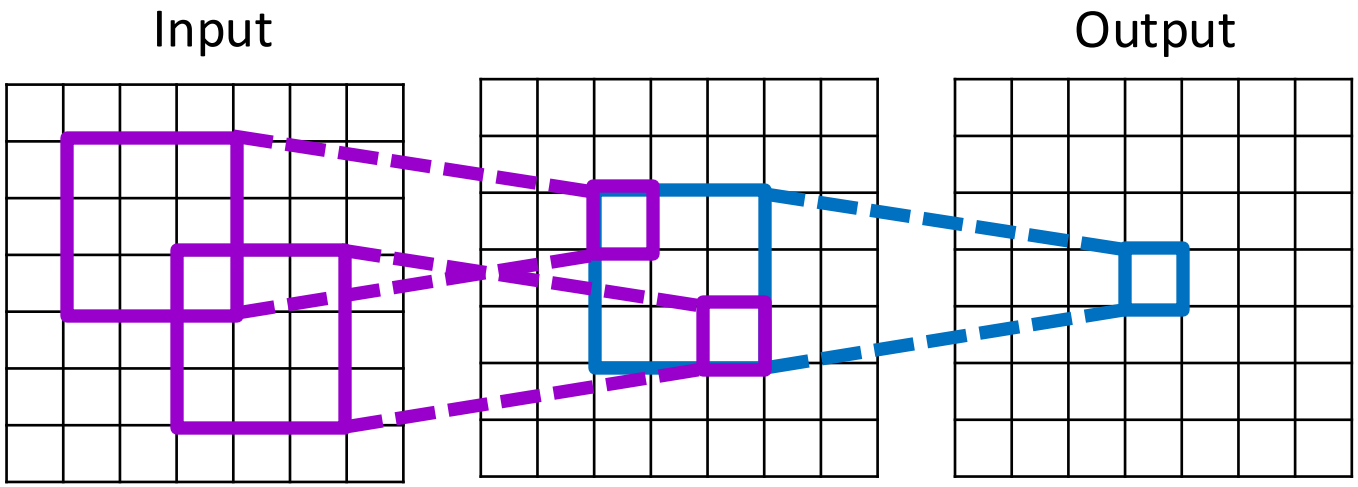
Receptive field size: 3x3

The *receptive field* of a unit is the region of the input feature map whose values contribute to the response of that unit (either in the previous layer or in the initial image).

[Slide credit: S. Lazebnik]

Receptive Field

3x3 convolutions, stride 1

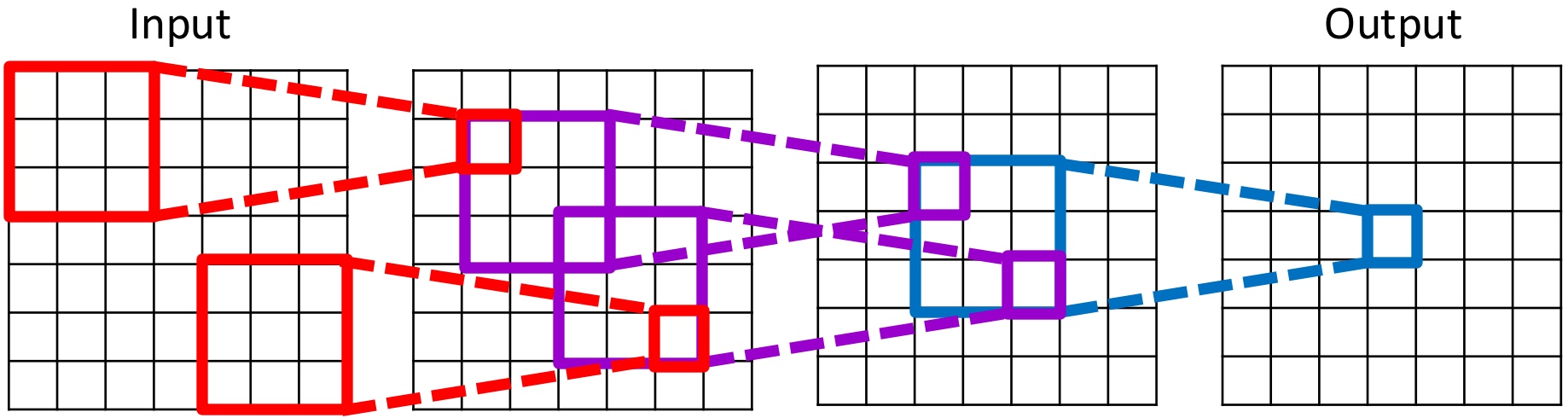


Receptive field size: 5x5

[Slide credit: S. Lazebnik]

Receptive Field

3x3 convolutions, stride 1



Receptive field size: 7x7

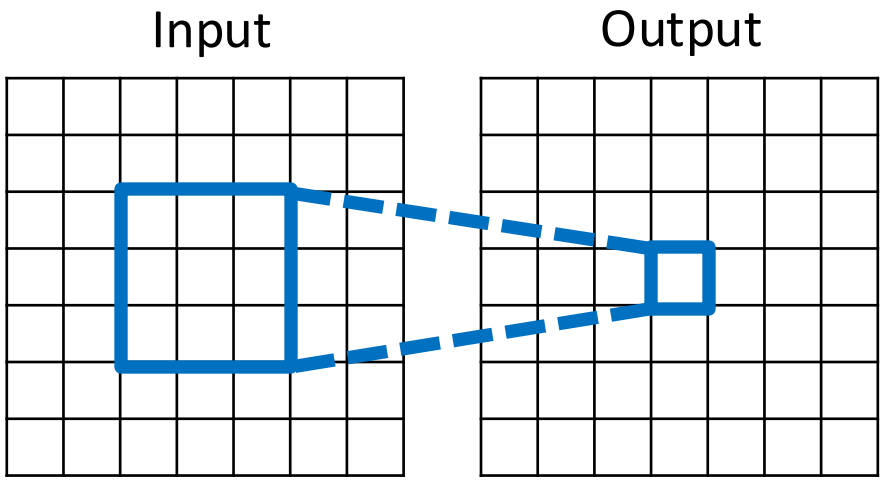
Each successive convolution adds $F - 1$ to the receptive field size

With L layers the receptive field size is $1 + L * (F - 1)$

[Slide credit: S. Lazebnik]

Receptive Field

3x3 convolutions, stride 2

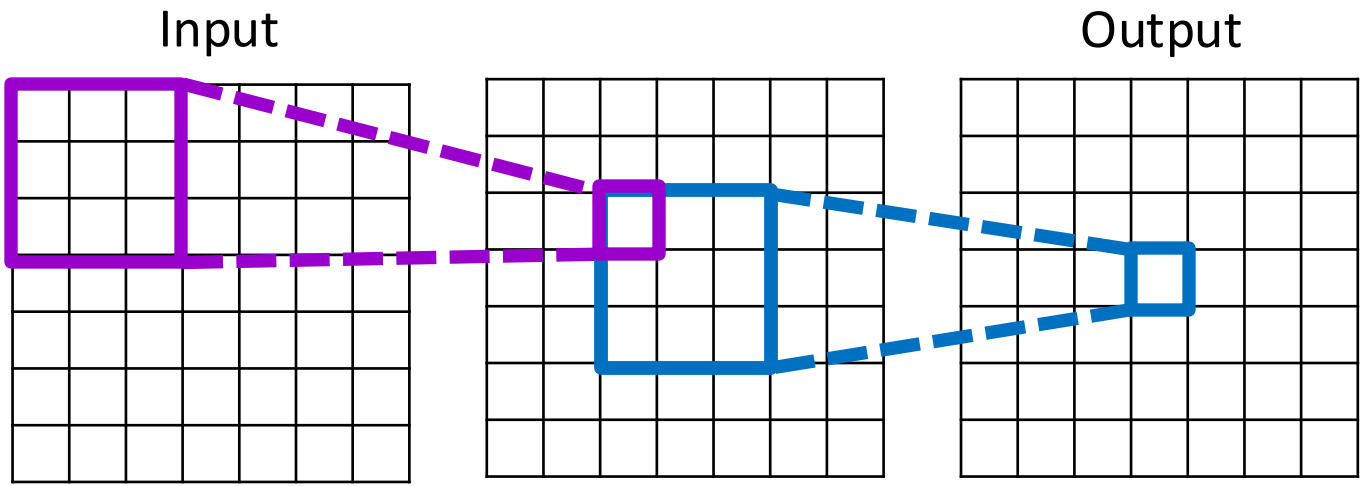


Receptive field size: 3x3

[Slide credit: S. Lazebnik]

Receptive Field

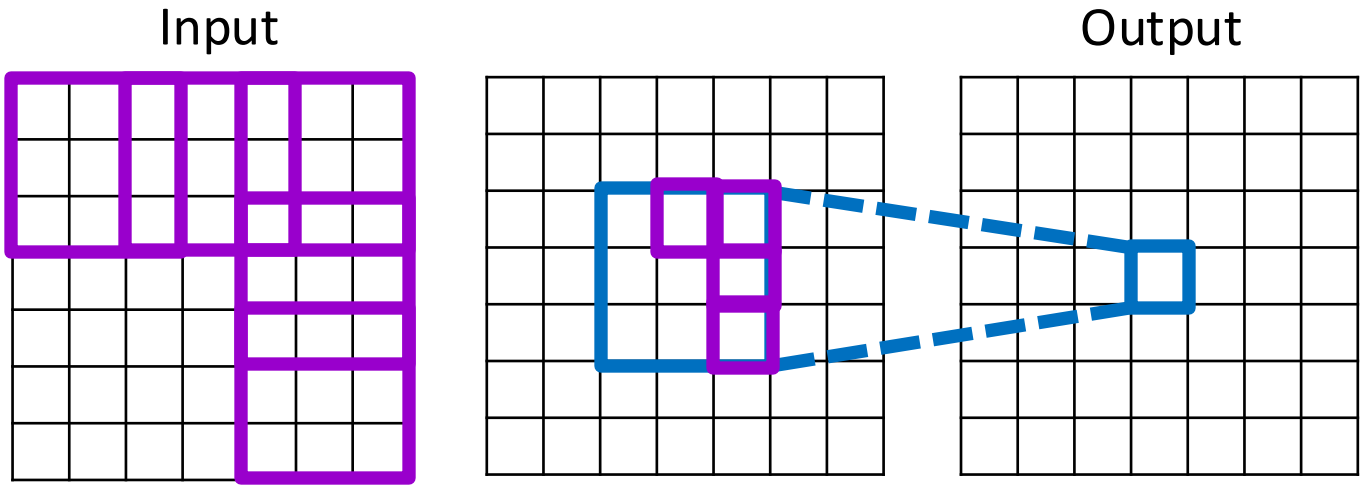
3x3 convolutions, stride 2



[Slide credit: S. Lazebnik]

Receptive Field

3x3 convolutions, stride 2



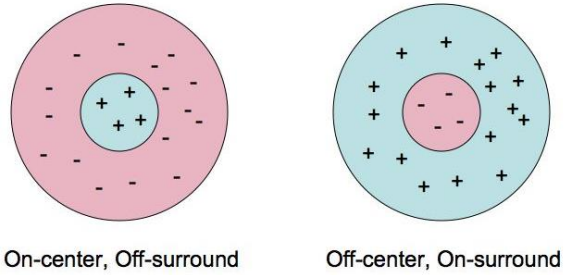
Receptive field size: 7x7

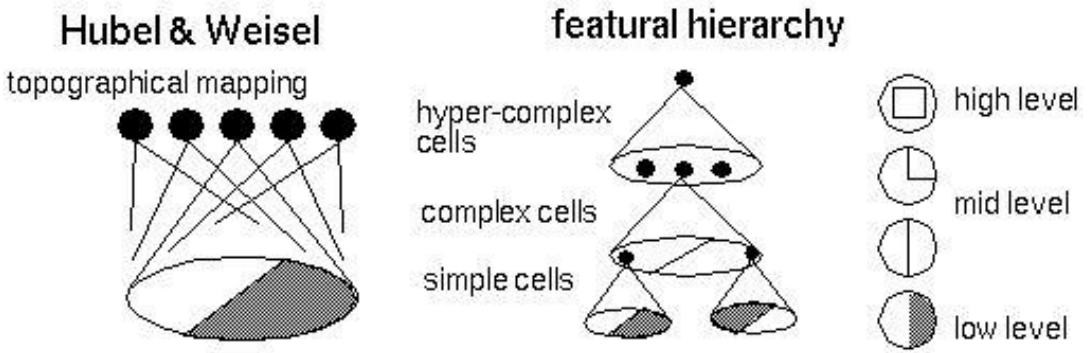
With a stride of 2, receptive field size is given by $2^{L+1} - 1$, i.e., it grows exponentially (though spatial resolution decreases exponentially)

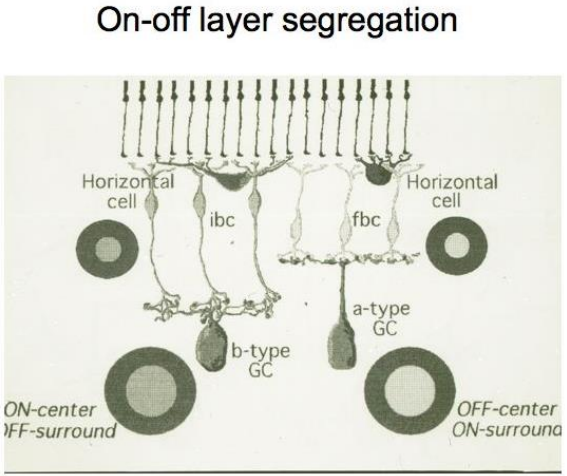
[Slide credit: S. Lazebnik]

Receptive Fields in Biology

- Biological inspiration: D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells







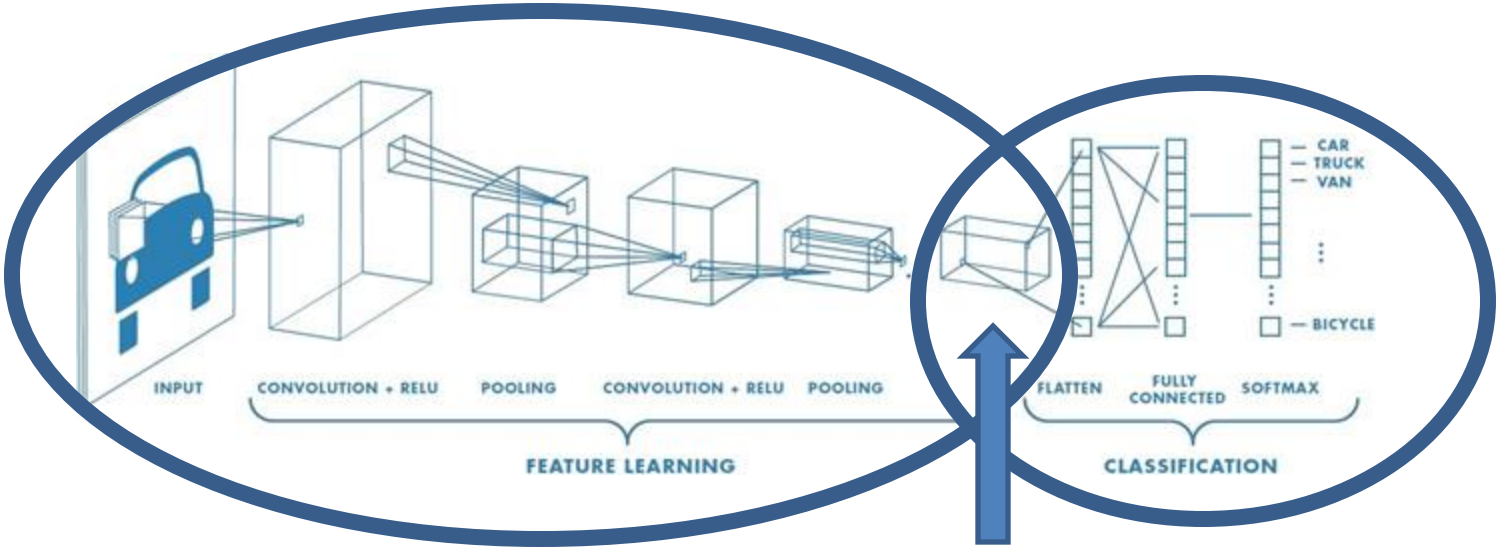
[Slide credit: S. Lazebnik]

[Source1, Source2]

Typical ConvNet Architecture

ConvNet architecture consists of sequence of convolutional blocks (convolution + relu + pooling + normalisation):

- First block convolves image into activations (depth = nr of filters)
- Consequent blocks convolve activations to new activations / feature maps



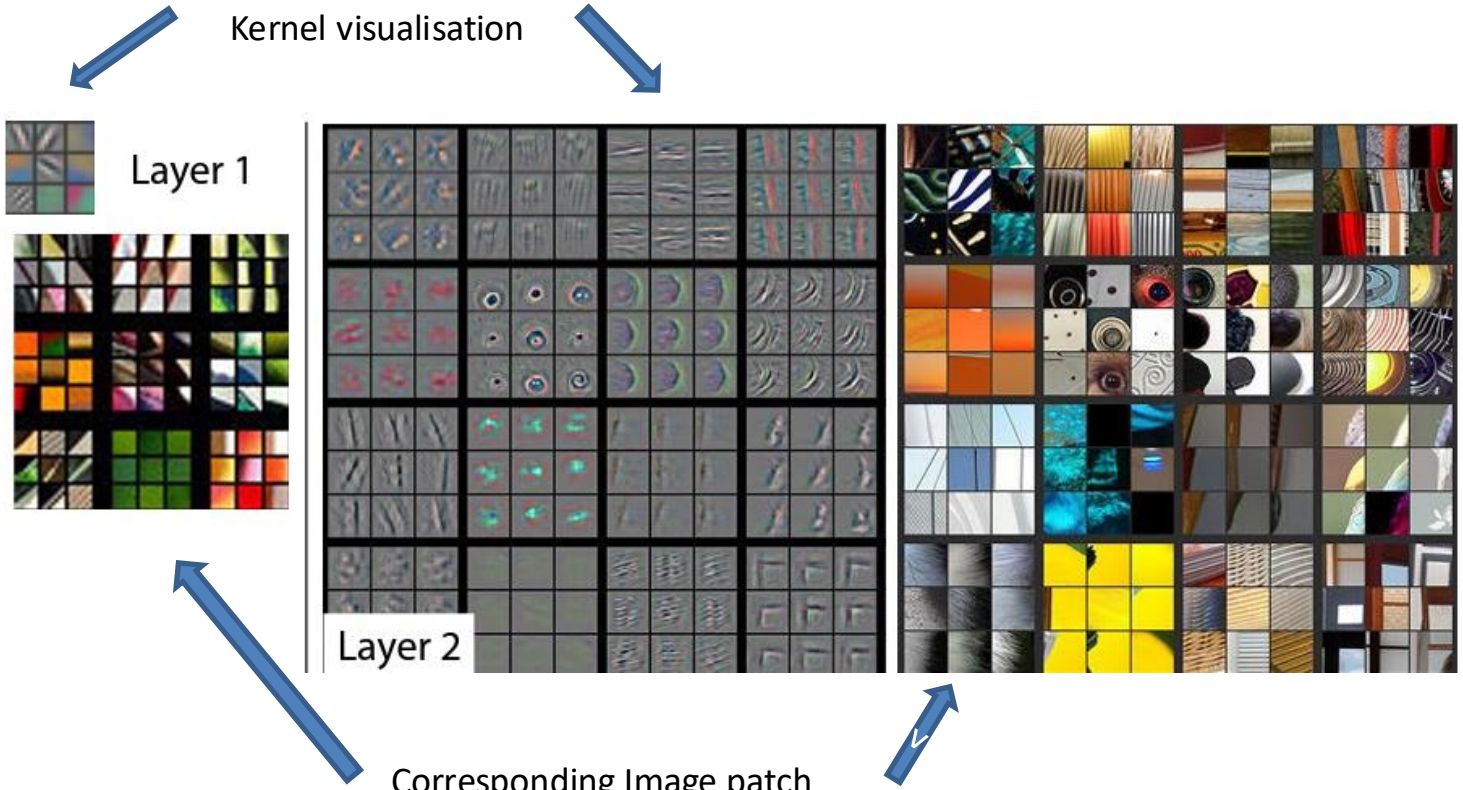
Final Feature Map

The final feature map is input for Classification

- Typical with fully connected layers

ConvNet Features

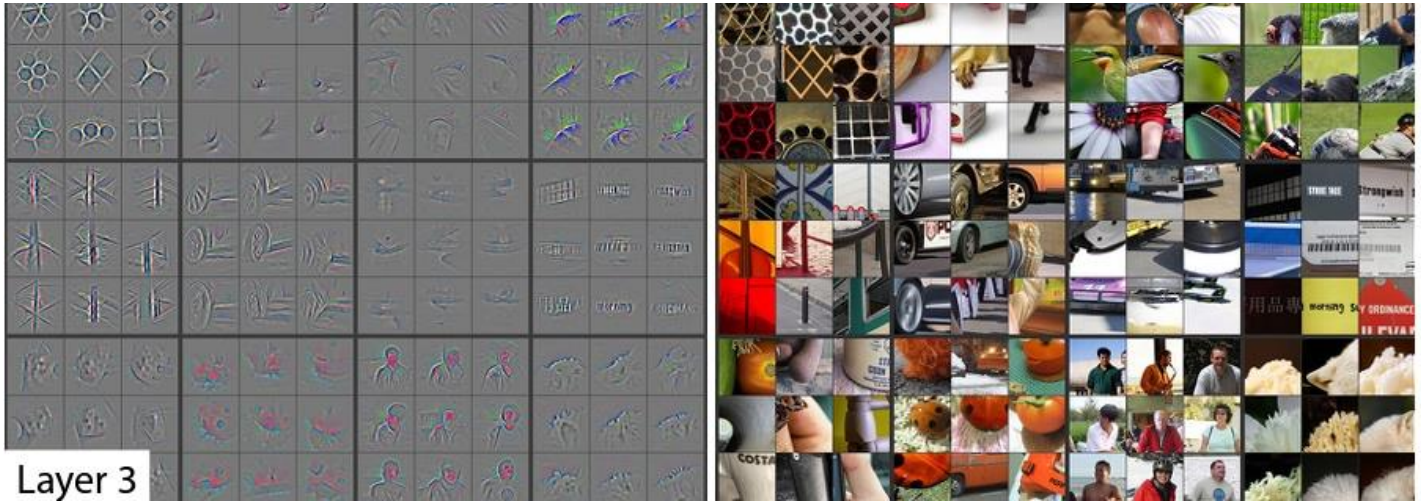
First layers learn basic features, later layers learn complex features



[Source: Zeiler, Fergus, 2013]

ConvNet Features

First layers learn basic features, later layers learn complex features

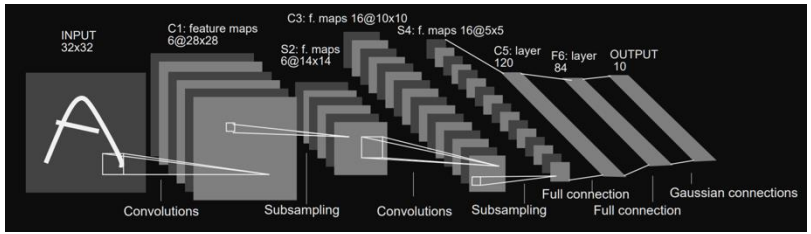


[Source: Zeiler, Fergus, 2013]

ConvNet Architectures

LeNet (Le Cun at al 1989, Bell lab) was first well known ConvNet.

Used back propagation.
Used for recognition of handwritten zipcode / checks (MNIST dataset)



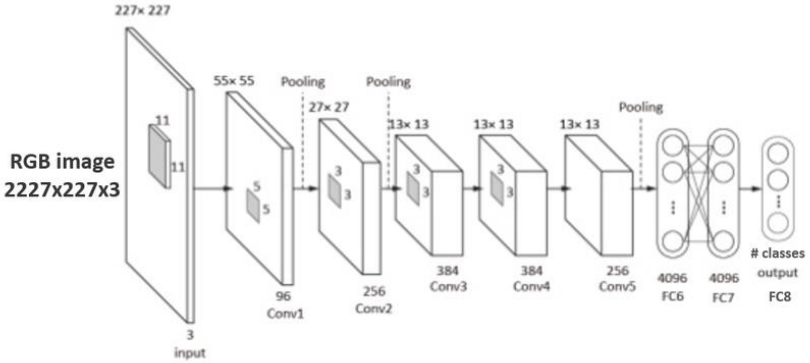
ConvNet Architecture with back propagation
- 2 convolutional layers
- reduces spacial dimensions: 28x28x1 (black & white image) → … → 5x5
- increases depth: 1 (black and white) → … → 16 (nr of filters)

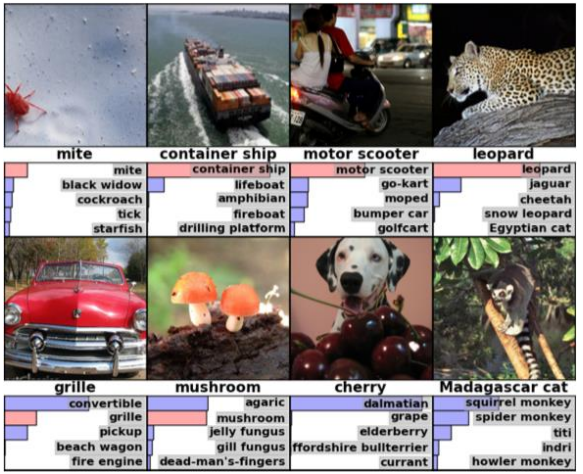
Issue: slow learning speed (CPU)

ConvNet Architectures

AlexNet (Alex Krishevsky, 2012)

2012 ImageNet Large Scale Visual Recognition Challenge: 15% top 5 error





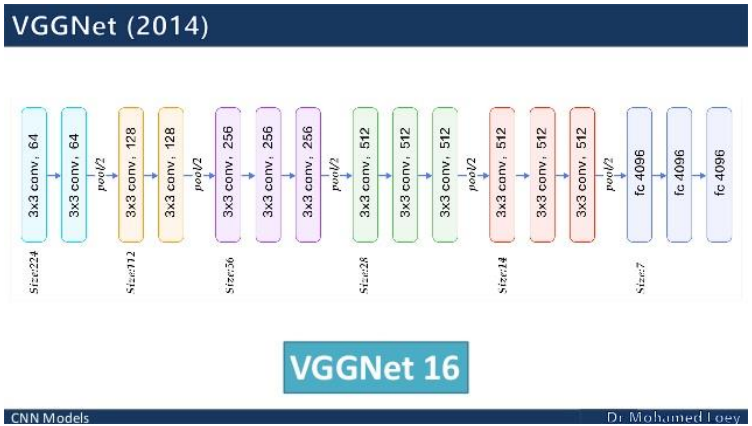
ConvNet Architecture:

- 5 convolutional layers and 3 fully connected layers (1000 classes)
- reduces spacial dimensions: 227x227 (image) → … → 13x13
- increases depth: 3 (image RGB depth) → … → 256 (nr of filters)
- use of Graphical Processing Unit (GPU)

ConvNet Architectures

VGGNet (Zimmerman, Simonyan, 2014)

The VGGNet achieved almost 92.7% top-5 test accuracy in 2014 ImageNet



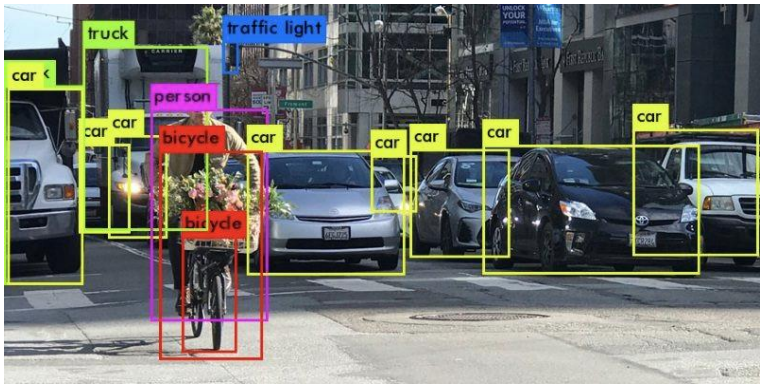
ConvNet Architecture:

- VGG16: 13 convolutional layers and 3 fully connected layers (1000 classes)
- reduces spacial dimensions: 224x224 (image) → … → 7x7
- increases depth: 3 (image RGB depth) → … → 512 (nr of filters)
- Large network. Even larger version: VGG19

ConvNet Architectures

Darknet-53 (2018) was a breakthrough in real-time object recognition: good recognition results in milliseconds

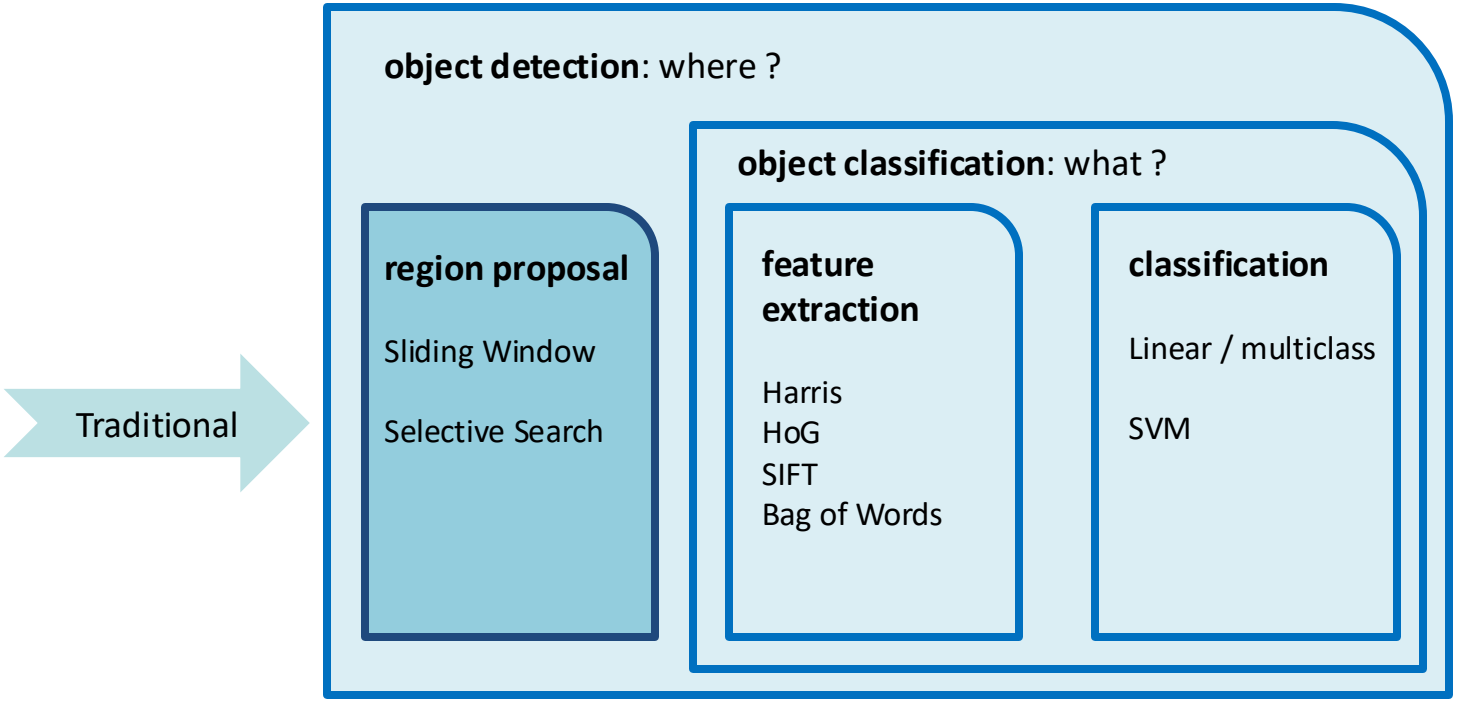
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			



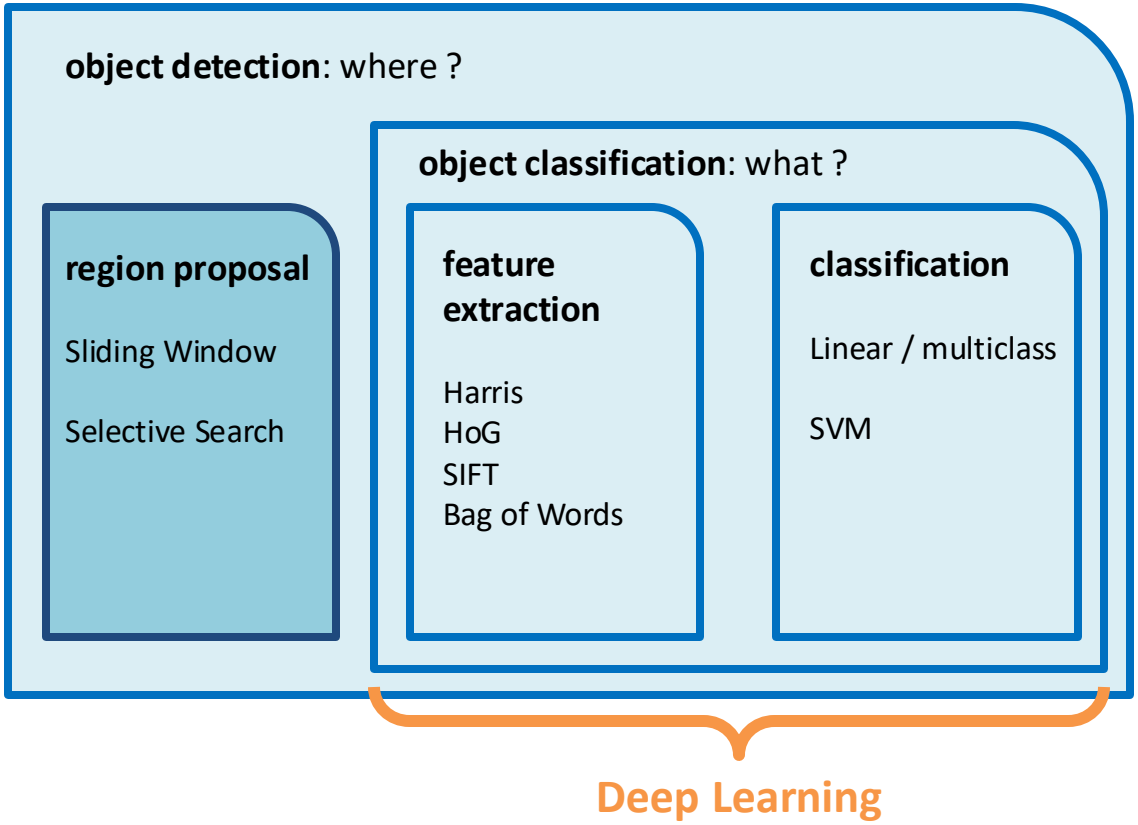
Today's Agenda

- Convolutional Neural Networks
- **Object Detection with Region-based CNNs**

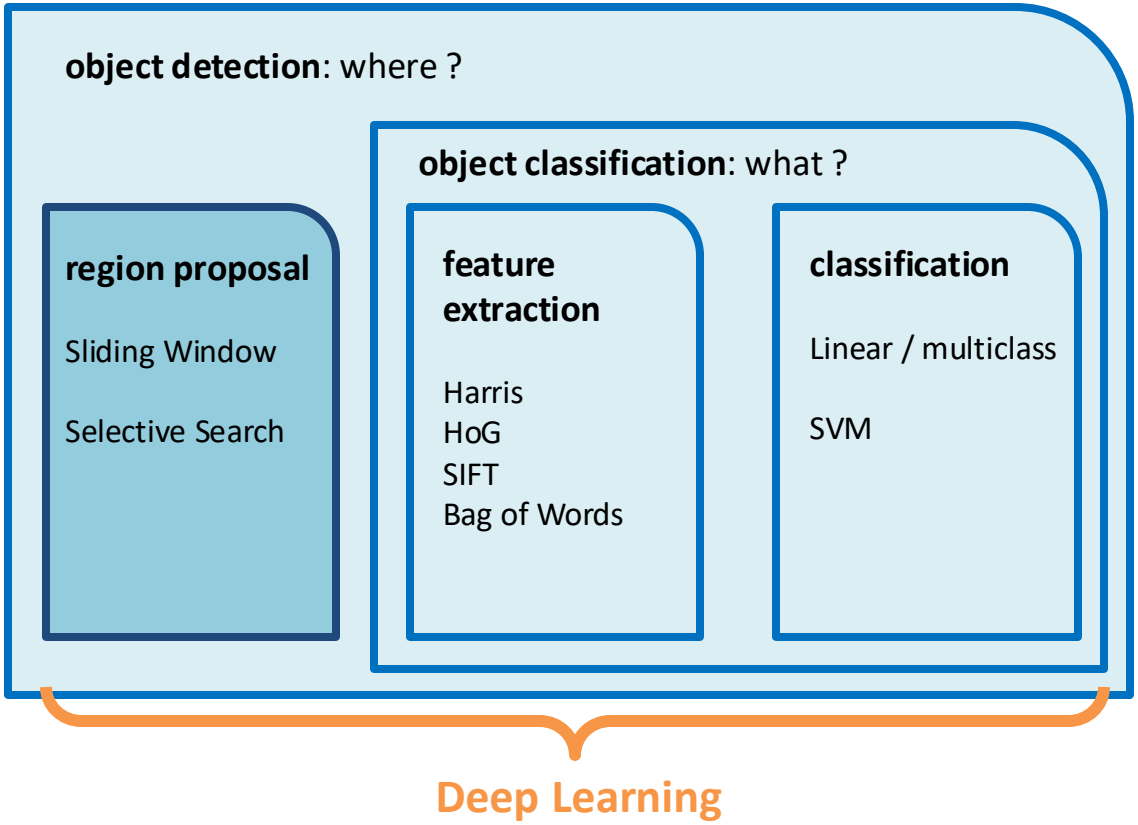
Object Detection



Object Detection

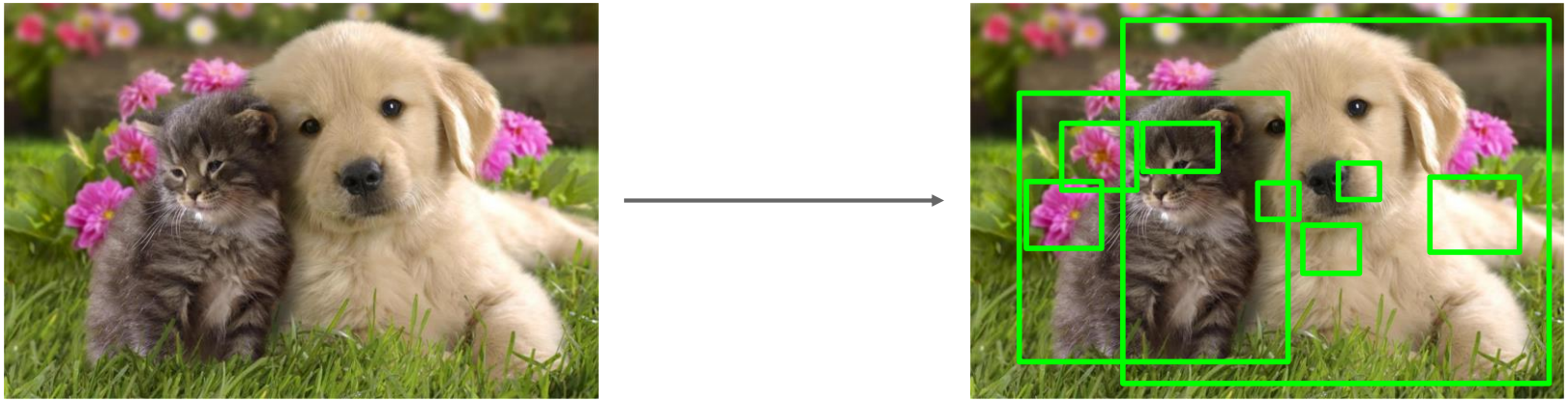


Object Detection



Region Proposals

- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region
- proposals in a few seconds on CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

[Slide credit: Justin Johnson]

Region-based CNN (R-CNN)

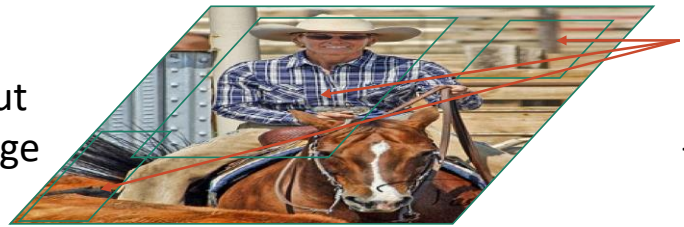
Input image

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

Region-based CNN (R-CNN)

Input image

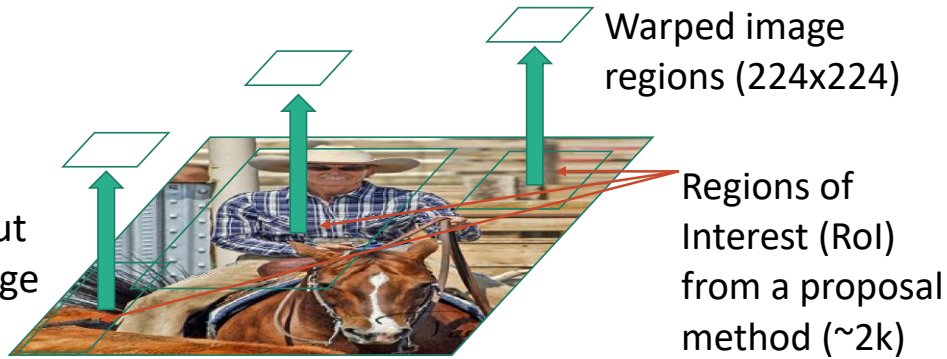


Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

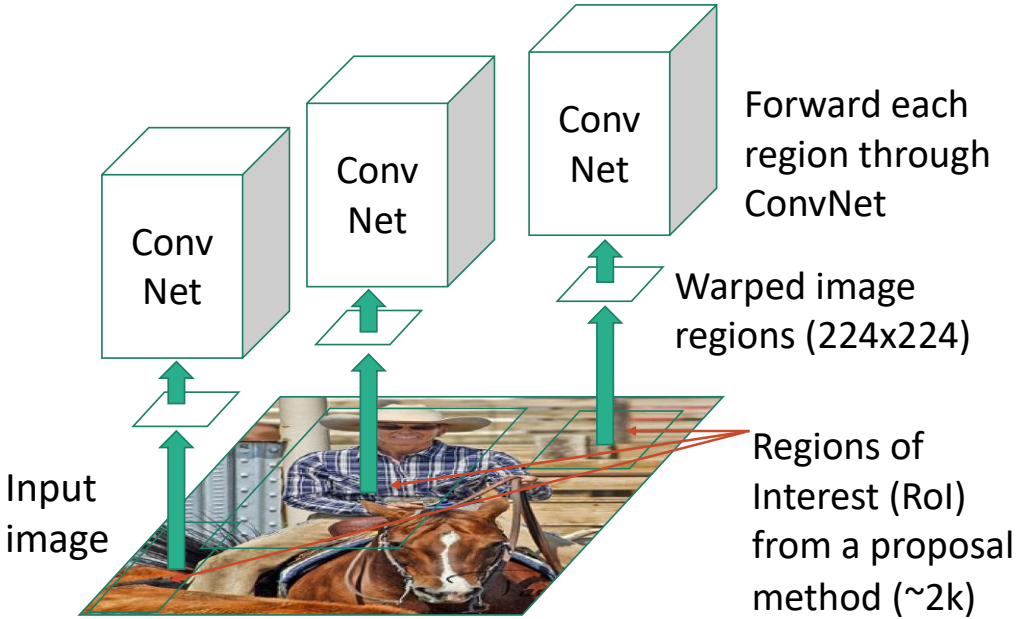
Region-based CNN (R-CNN)



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

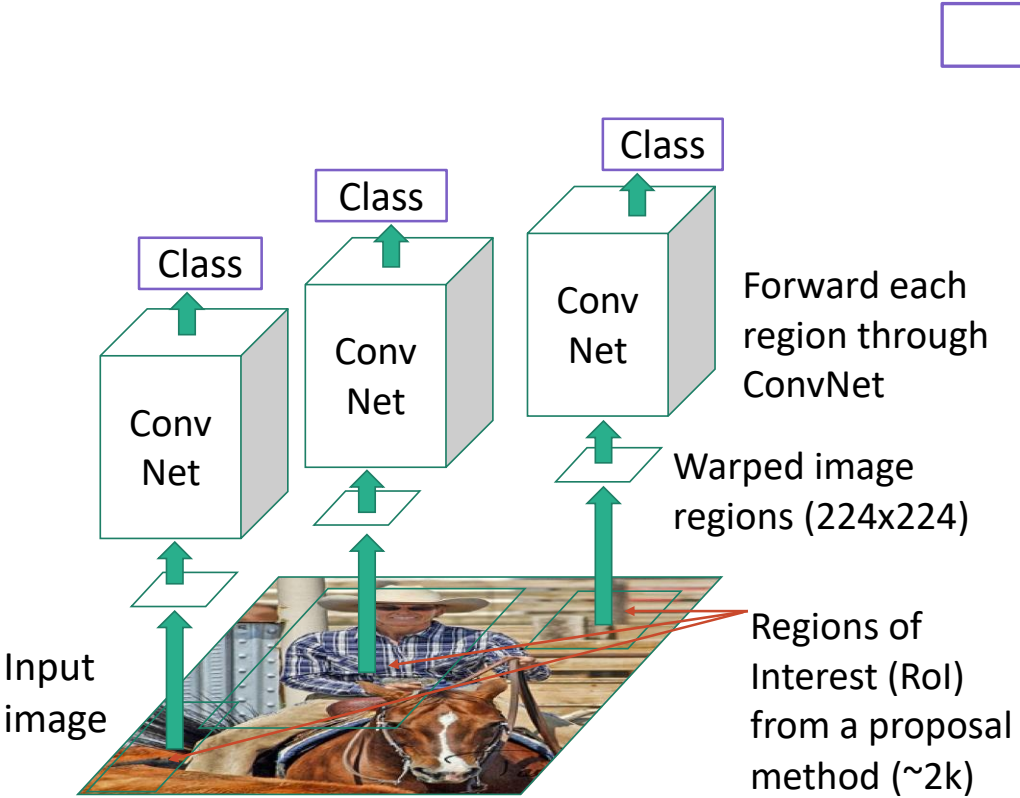
Region-based CNN (R-CNN)



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

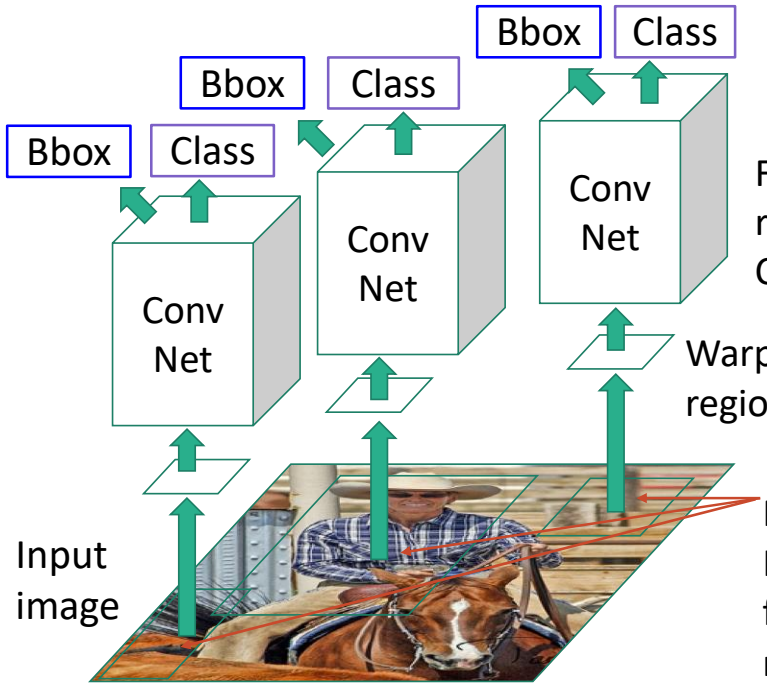
Region-based CNN (R-CNN)



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

Region-based CNN (R-CNN)



Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

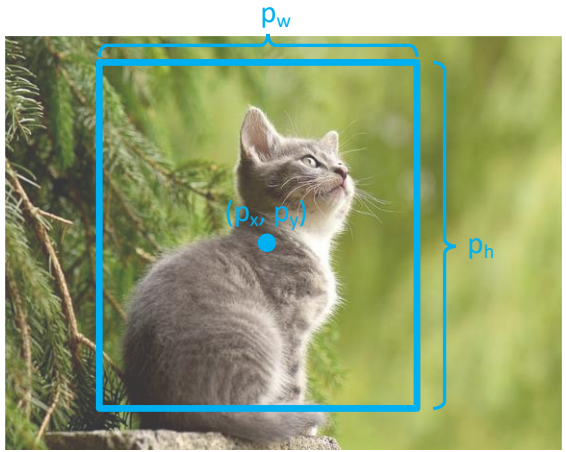
Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

R-CNN: Box Regression

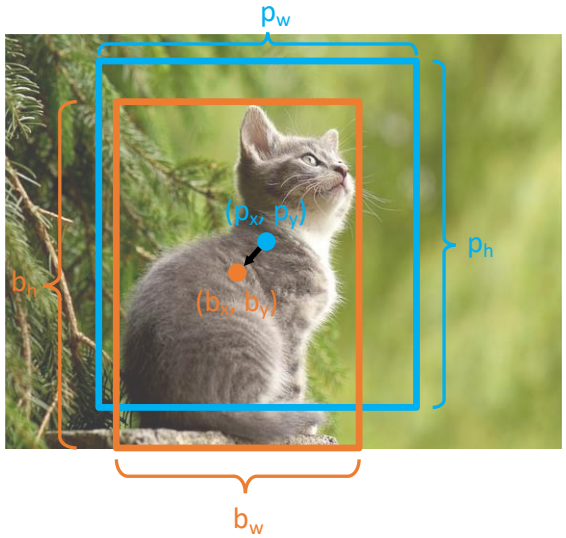


Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

[Slide credit: Justin Johnson]

R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is defined by:

$b_x = p_x + p_w t_x$
$b_y = p_y + p_h t_y$

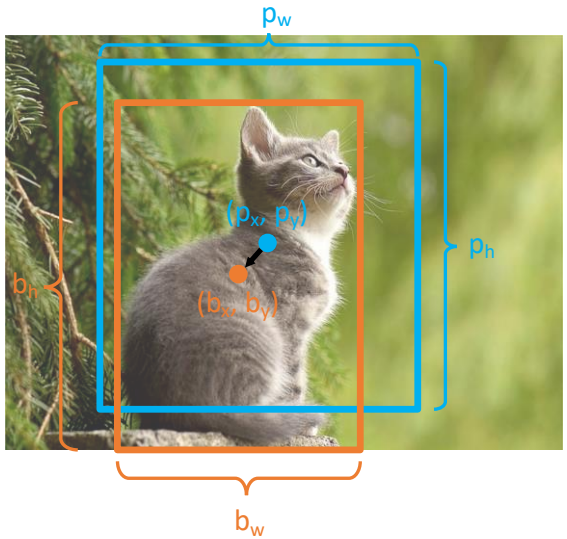
Shift center by amount relative to proposal size

$b_w = p_w \exp(t_w)$
$b_h = p_h \exp(t_h)$

Scale proposal; exp ensures that scaling factor is > 0

[Slide credit: Justin Johnson]

R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is:

$b_x = p_x + p_w t_x$

$b_y = p_y + p_h t_y$

$b_w = p_w \exp(t_w)$

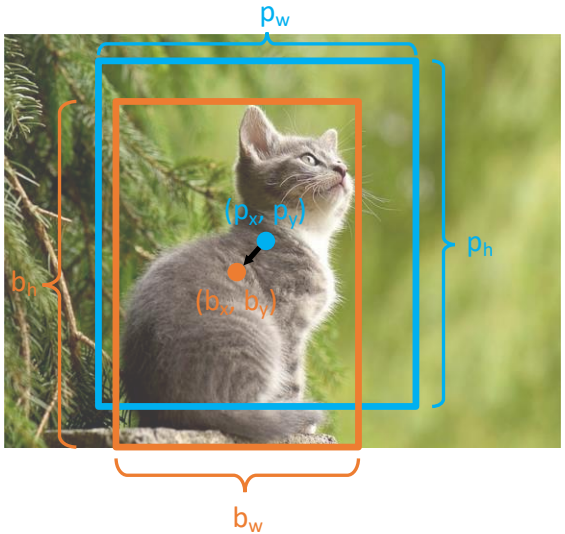
$b_h = p_h \exp(t_h)$

When transform is 0, output = proposal

L2 regularization encourages leaving proposal unchanged

[Slide credit: Justin Johnson]

R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is:

$b_x = p_x + p_w t_x$

$b_y = p_y + p_h t_y$

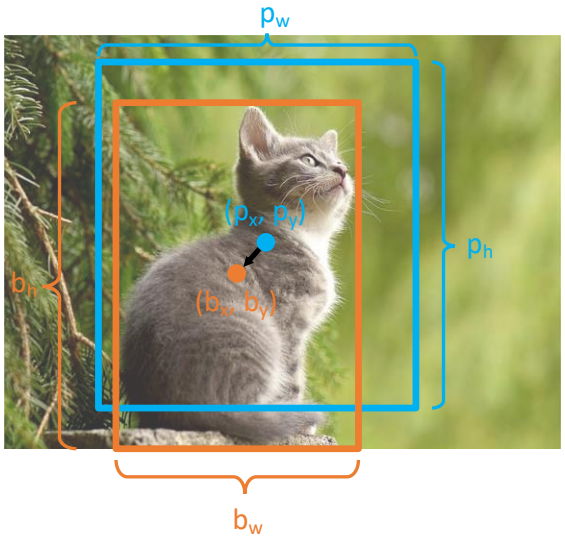
$b_w = p_w \exp(t_w)$

$b_h = p_h \exp(t_h)$

Scale / Translation invariance: Transform encodes *relative* difference between proposal and output; important since CNN doesn't see absolute size or position after cropping

[Slide credit: Justin Johnson]

R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is:

$b_x = p_x + p_w t_x$

$b_y = p_y + p_h t_y$

$b_w = p_w \exp(t_w)$

$b_h = p_h \exp(t_h)$

Given proposal and target output, we can solve for the transform the network should output:

$t_x = (b_x - p_x)/p_w$

$t_y = (b_y - p_y)/p_h$

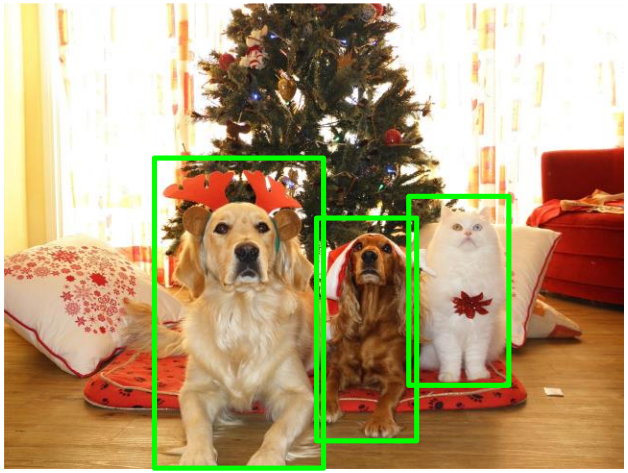
$t_w = \log(b_w/p_w)$

$t_h = \log(b_h/p_h)$

[Slide credit: Justin Johnson]

R-CNN Training

Input Image

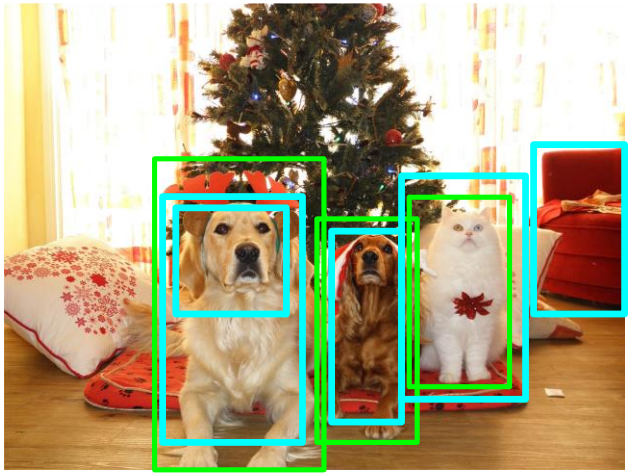


Ground-Truth boxes

[Slide credit: Justin Johnson]

R-CNN Training

Input Image

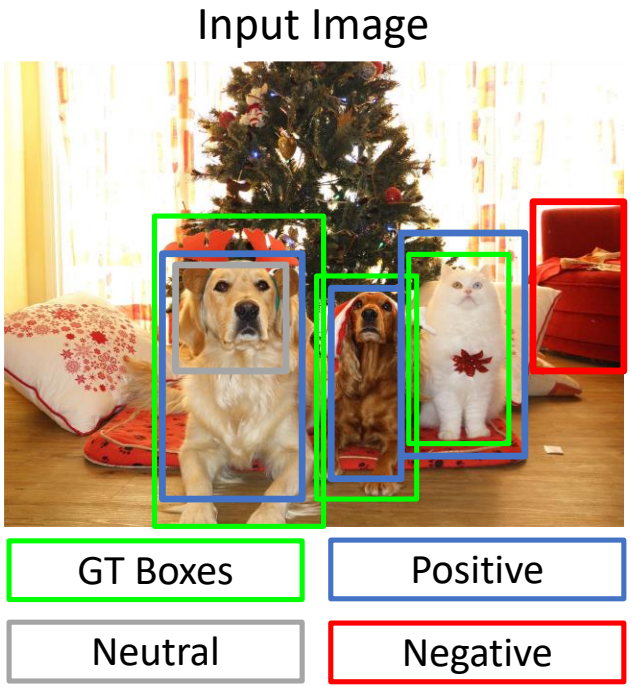


Ground-Truth boxes

Region Proposals

[Slide credit: Justin Johnson]

R-CNN Training

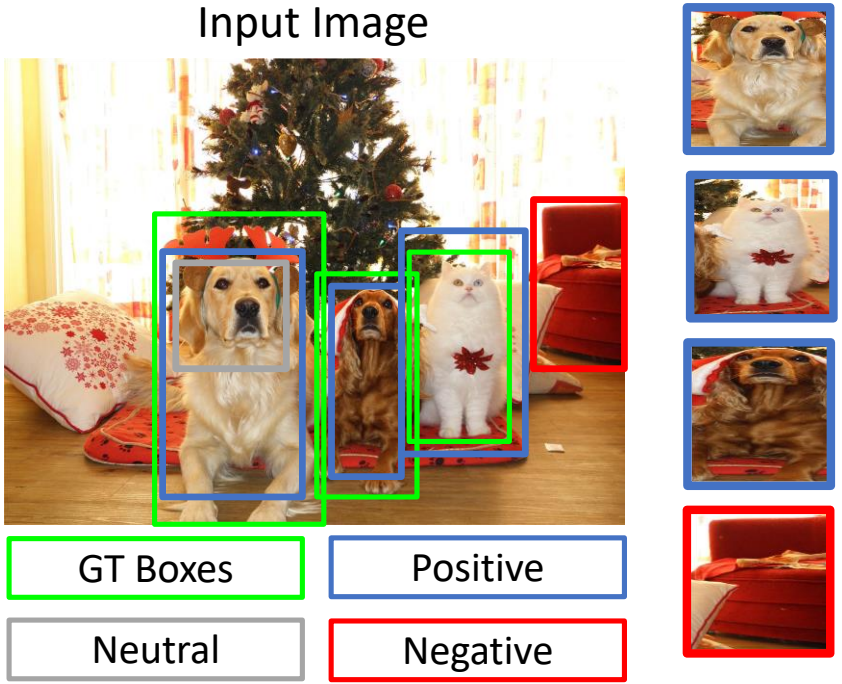


Categorize each region proposal as positive, negative, or neutral based on overlap with ground-truth boxes:

Positive: > 0.5 IoU with a GT box
Negative: < 0.3 IoU with all GT boxes
Neutral: between 0.3 and 0.5 IoU with GT boxes

[Slide credit: Justin Johnson]

R-CNN Training

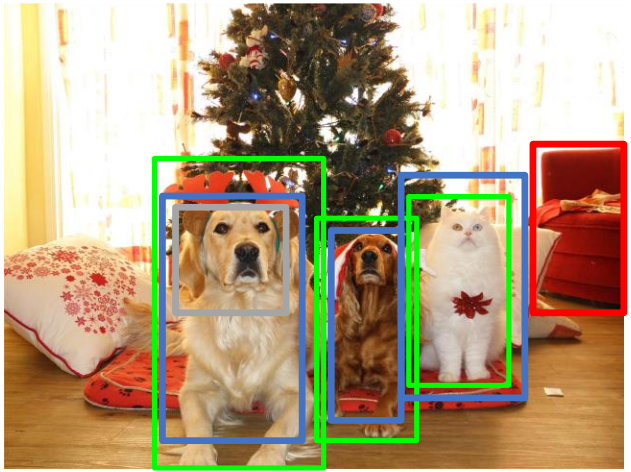


Crop pixels from each positive and negative proposal, resize to 224 x 224

[Slide credit: Justin Johnson]

R-CNN Training

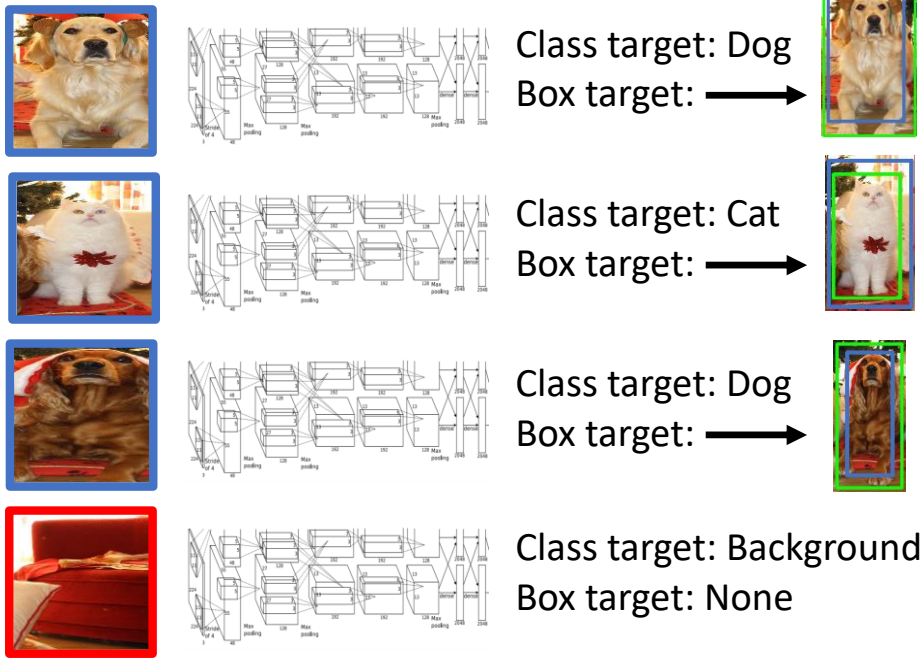
Input Image



GT Boxes | Positive

Neutral | Negative

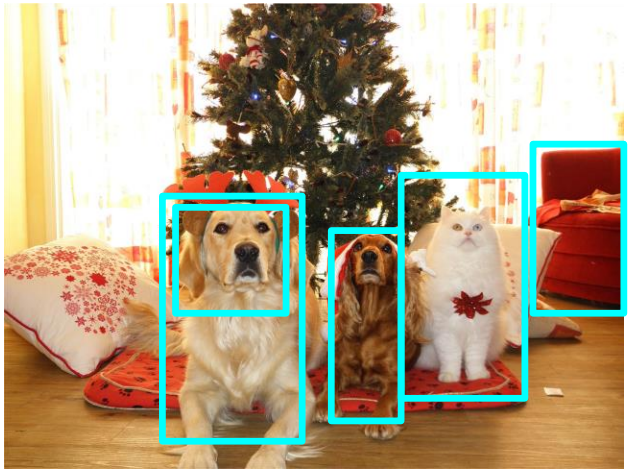
Run each region through CNN
Positive regions: predict class and transform
Negative regions: just predict class



[Slide credit: Justin Johnson]

R-CNN Test-Time

Input Image



Region Proposals

1. Run proposal method
2. Run CNN on each proposal to get class scores, transforms
3. Threshold class scores to get a set of detections

2 problems:
- CNN often outputs overlapping boxes
- How to set thresholds?

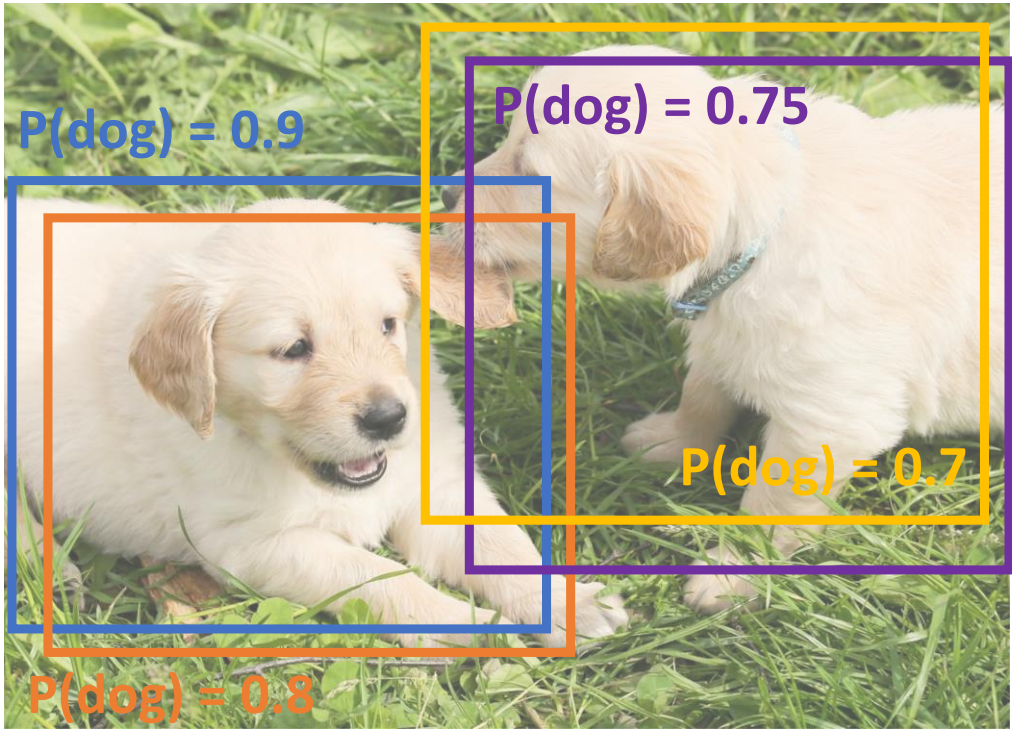
[Slide credit: Justin Johnson]

Filter Boxes: Non-Maximum Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using **Non-Max Suppression (NMS):**

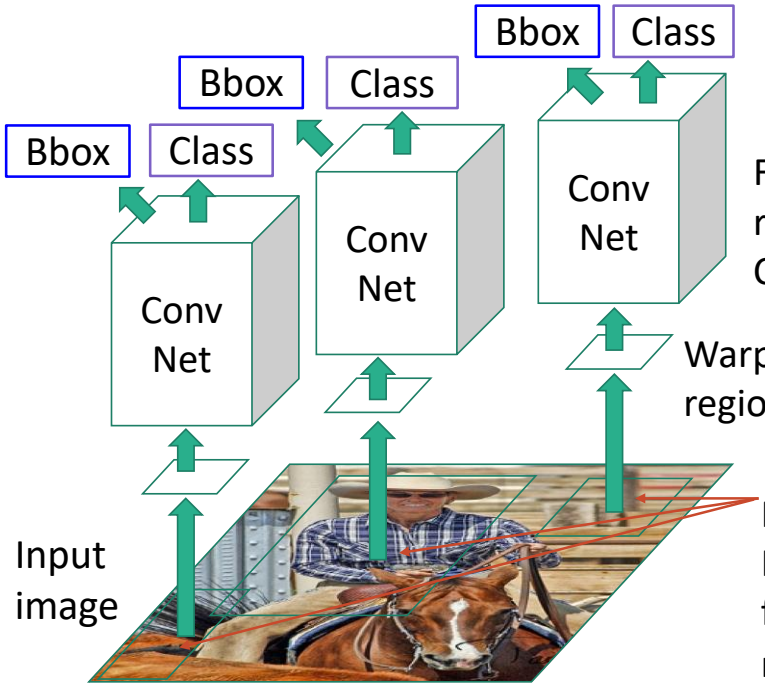
1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



[Puppy image is CC0 Public Domain]

[Slide credit: Justin Johnson]

R-CNN - Drawbacks



Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

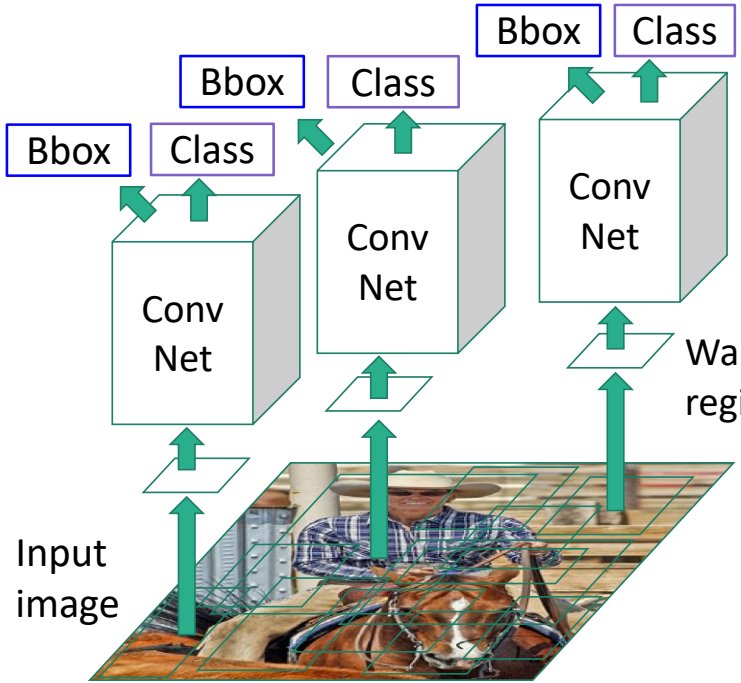
Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

R-CNN - Drawbacks



Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

Warped image regions (224x224)

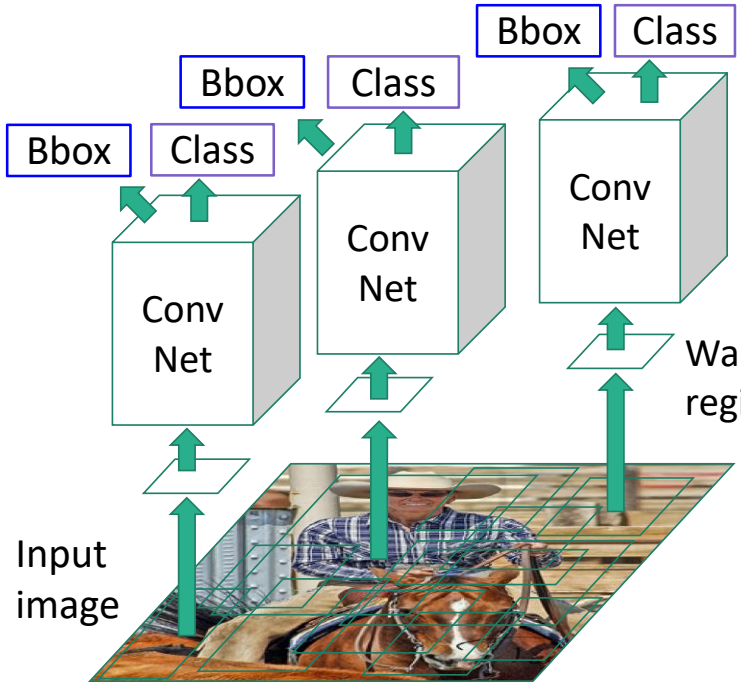
Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow! Need to do 2000 forward passes through CNN per image

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

R-CNN - Drawbacks



Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

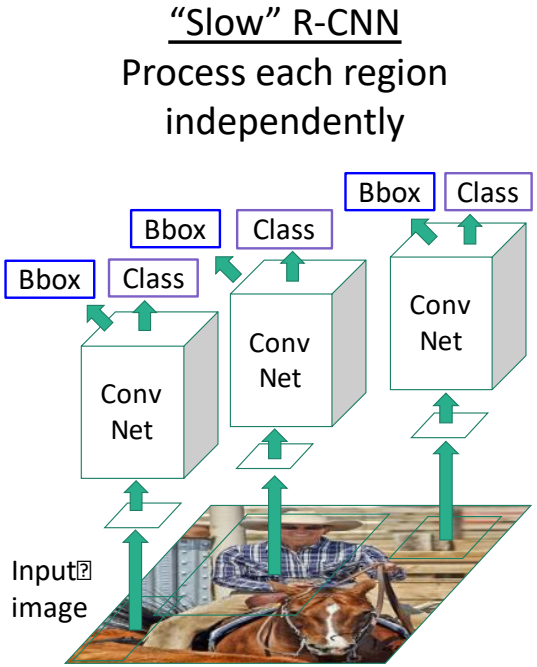
Problem: Very slow! Need to do 2000 forward passes through CNN per image

Idea: Overlapping proposals cause a lot of repeated work: same pixels processed many times. Can we avoid this?

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

[Slide credit: Justin Johnson]

R-CNN - Drawbacks

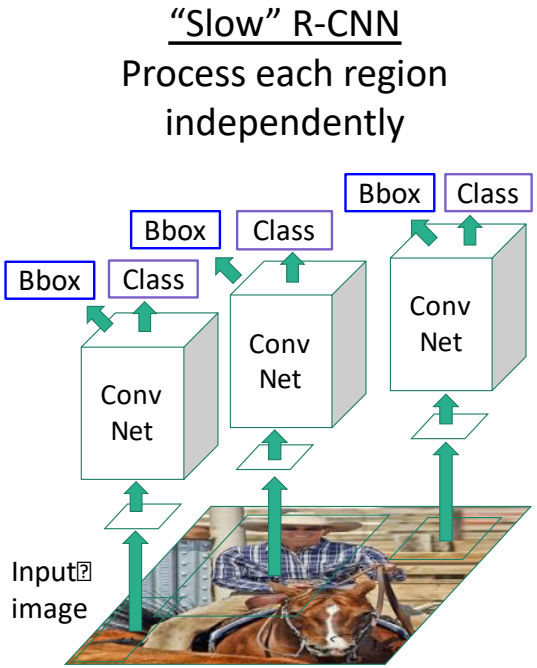


[Slide credit: Justin Johnson]

R-CNN - Drawbacks

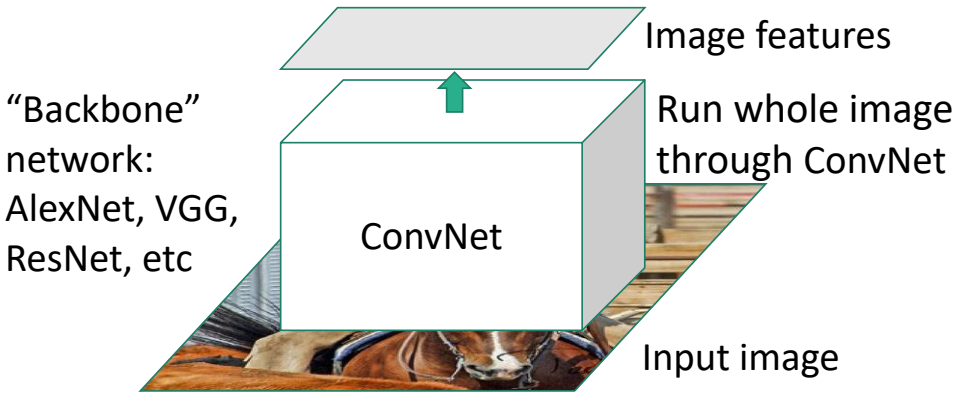
Input image

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



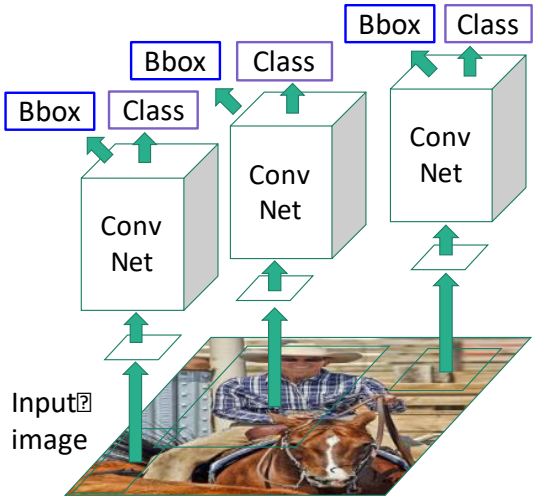
[Slide credit: Justin Johnson]

R-CNN - Drawbacks



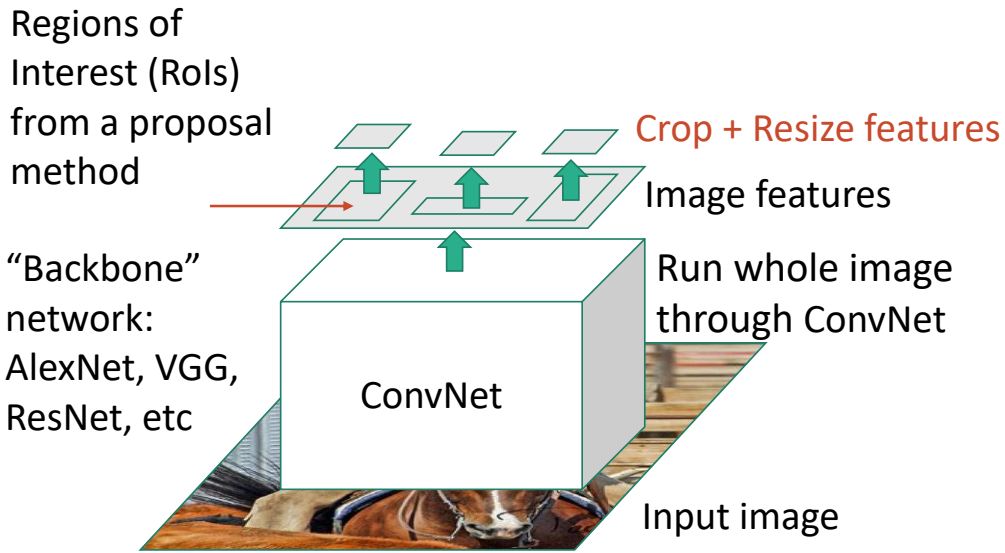
Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

“Slow” R-CNN
Process each region independently

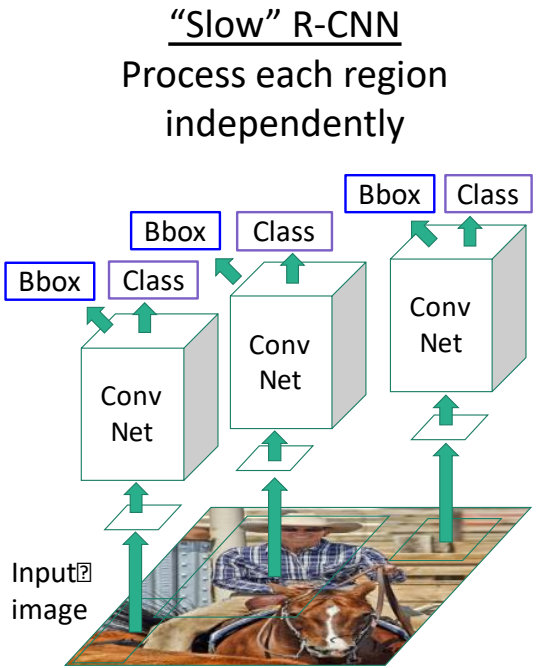


[Slide credit: Justin Johnson]

R-CNN - Drawbacks

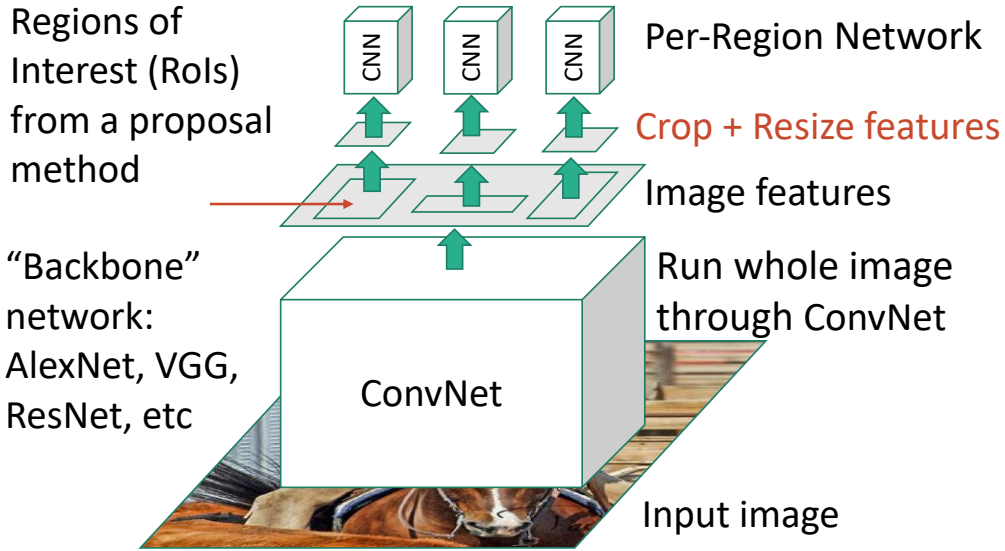


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

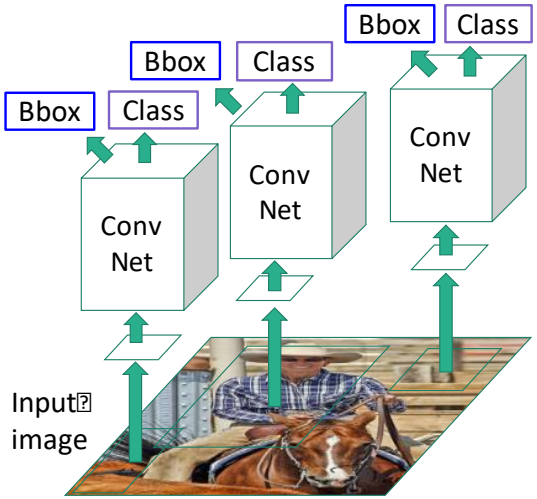


[Slide credit: Justin Johnson]

Fast R-CNN

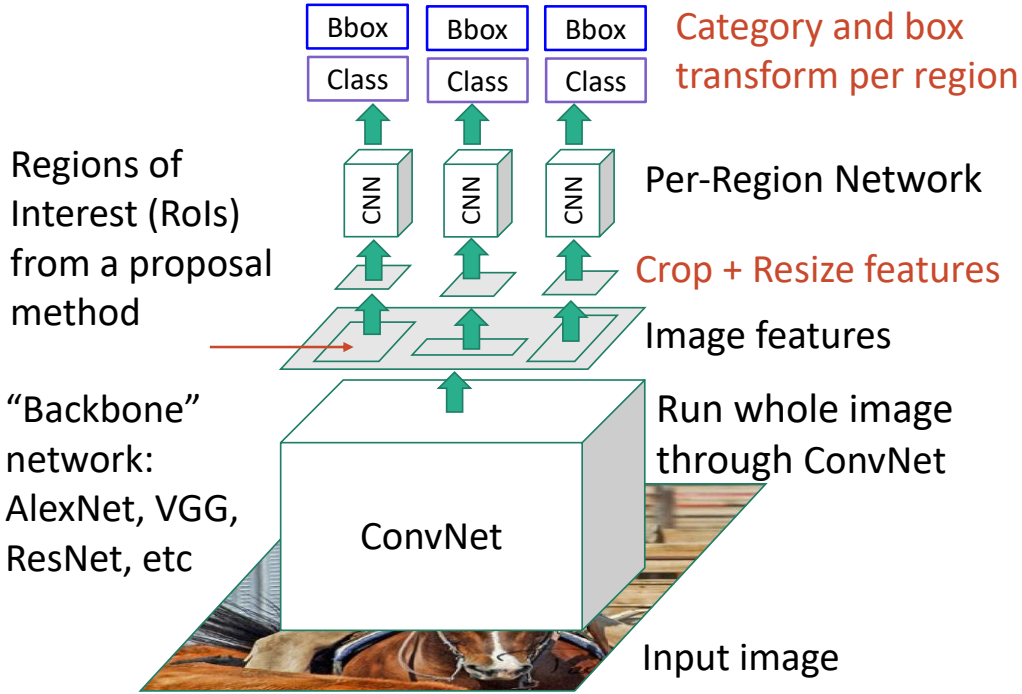


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

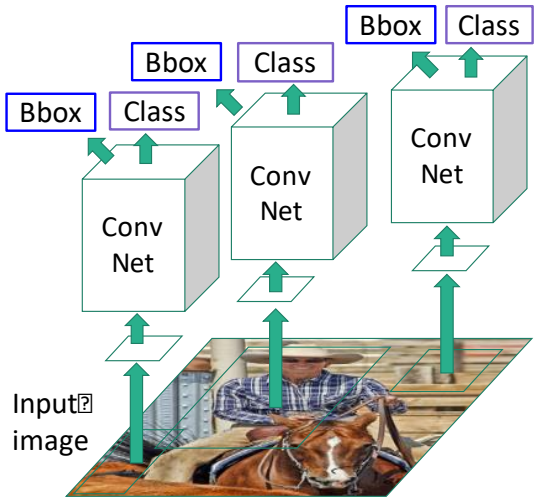


[Slide credit: Justin Johnson]

Fast R-CNN



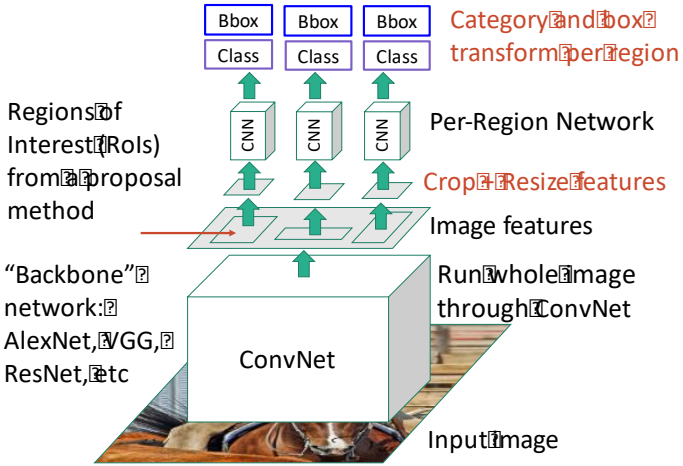
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



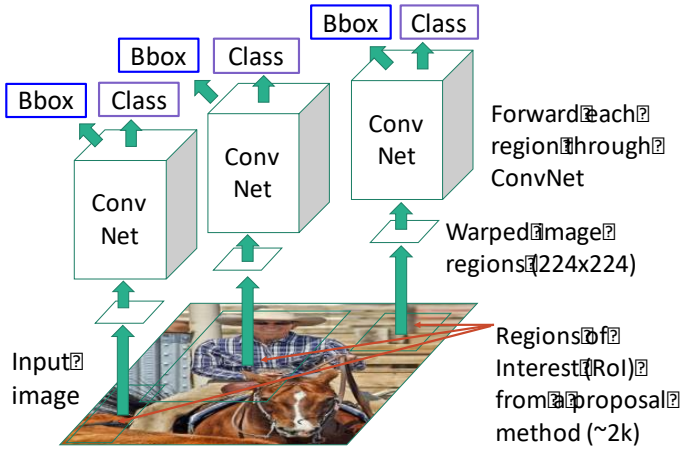
[Slide credit: Justin Johnson]

Fast R-CNN vs. (Slow) R-CNN

Fast R-CNN: Apply differentiable cropping to shared image features

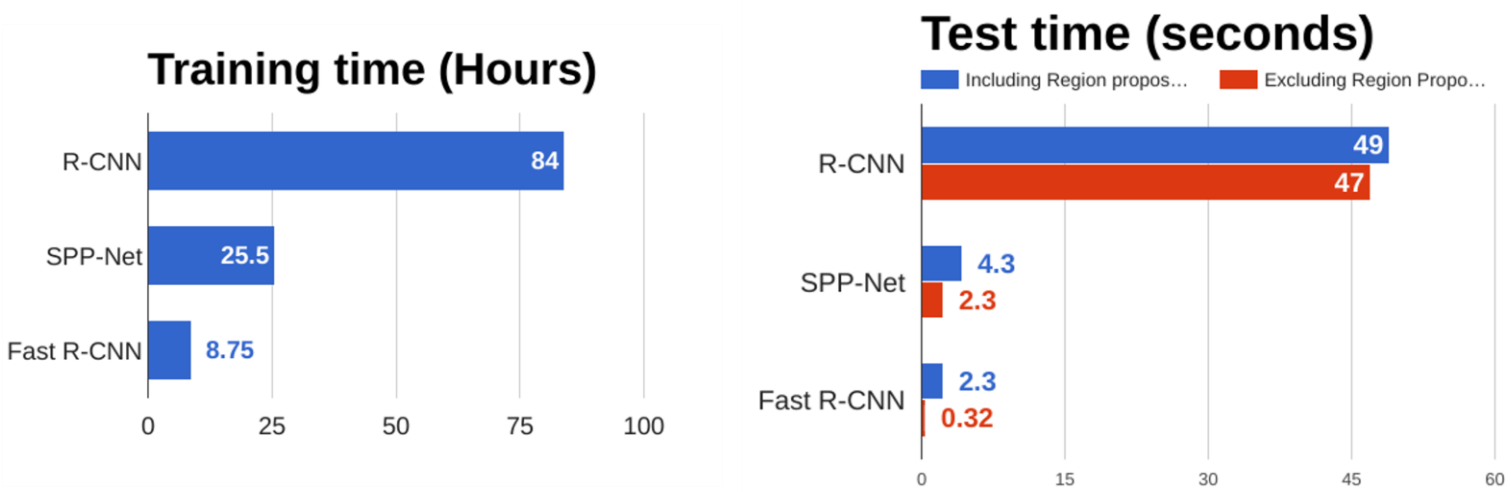


"Slow" R-CNN: Apply differentiable cropping to shared image features



[Slide credit: Justin Johnson]

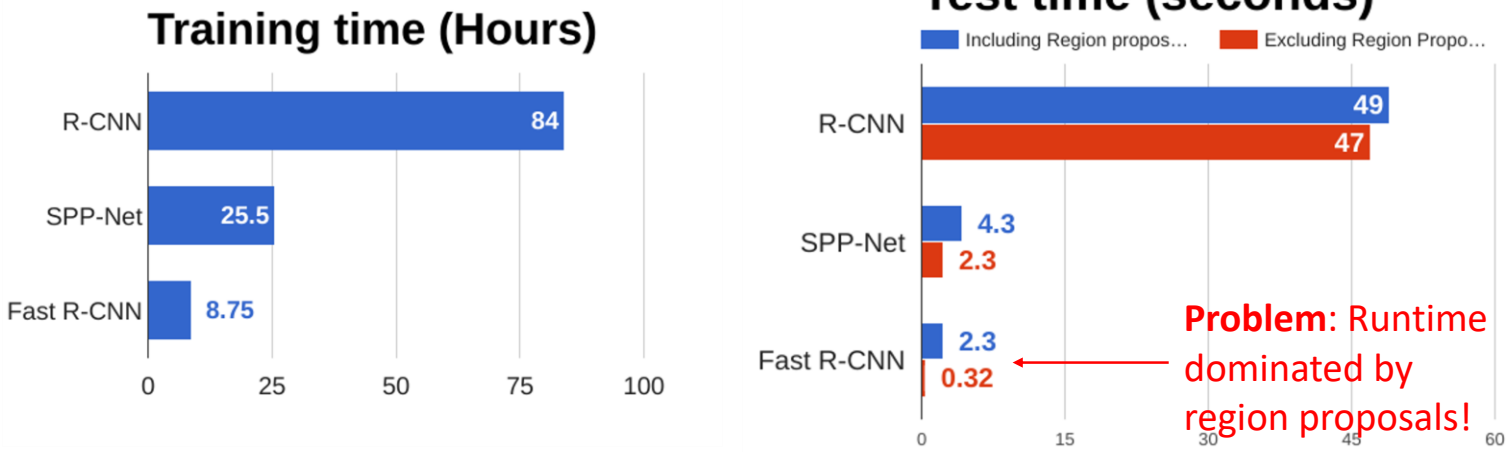
Fast R-CNN vs. (Slow) R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

[Slide credit: Justin Johnson]

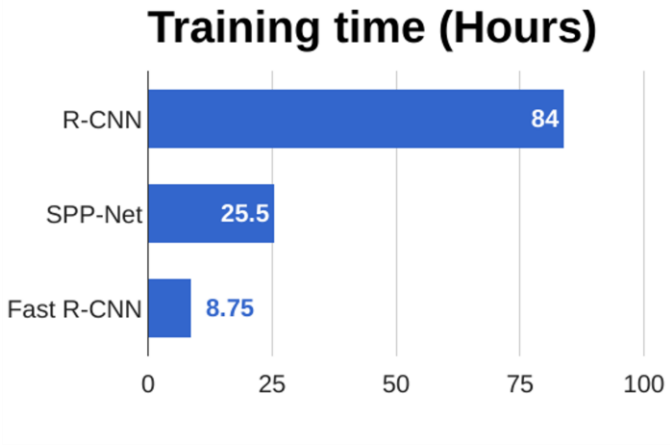
Fast R-CNN vs. (Slow) R-CNN

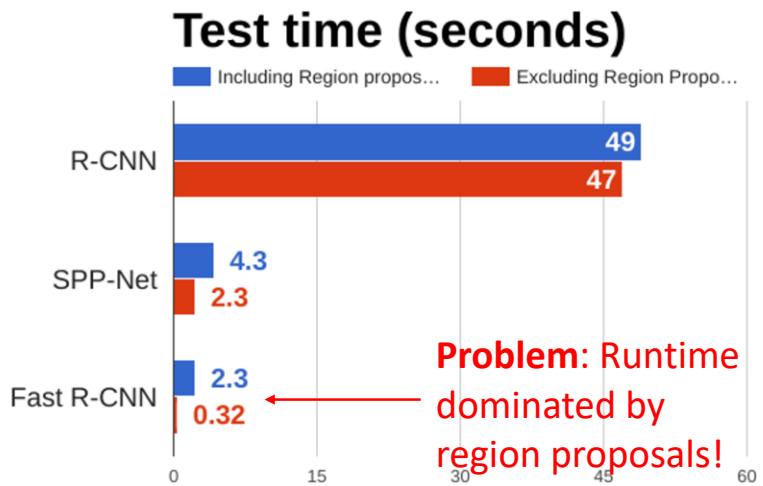


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

[Slide credit: Justin Johnson]

Fast R-CNN vs. (Slow) R-CNN





Problem: Runtime dominated by region proposals!

Recall: Region proposals computed by heuristic ”Selective Search” algorithm on CPU -- let’s learn them with a CNN instead!

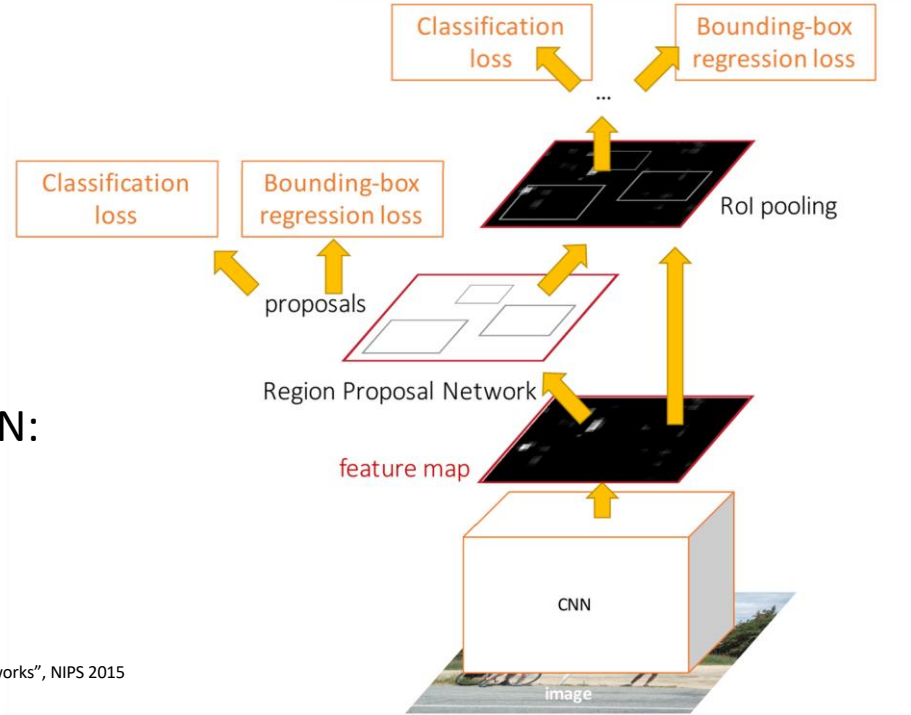
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

[Slide credit: Justin Johnson]

Faster R-CNN: Learnable Region Proposals

Insert **Region Proposal Network (RPN)** to predict proposals from features

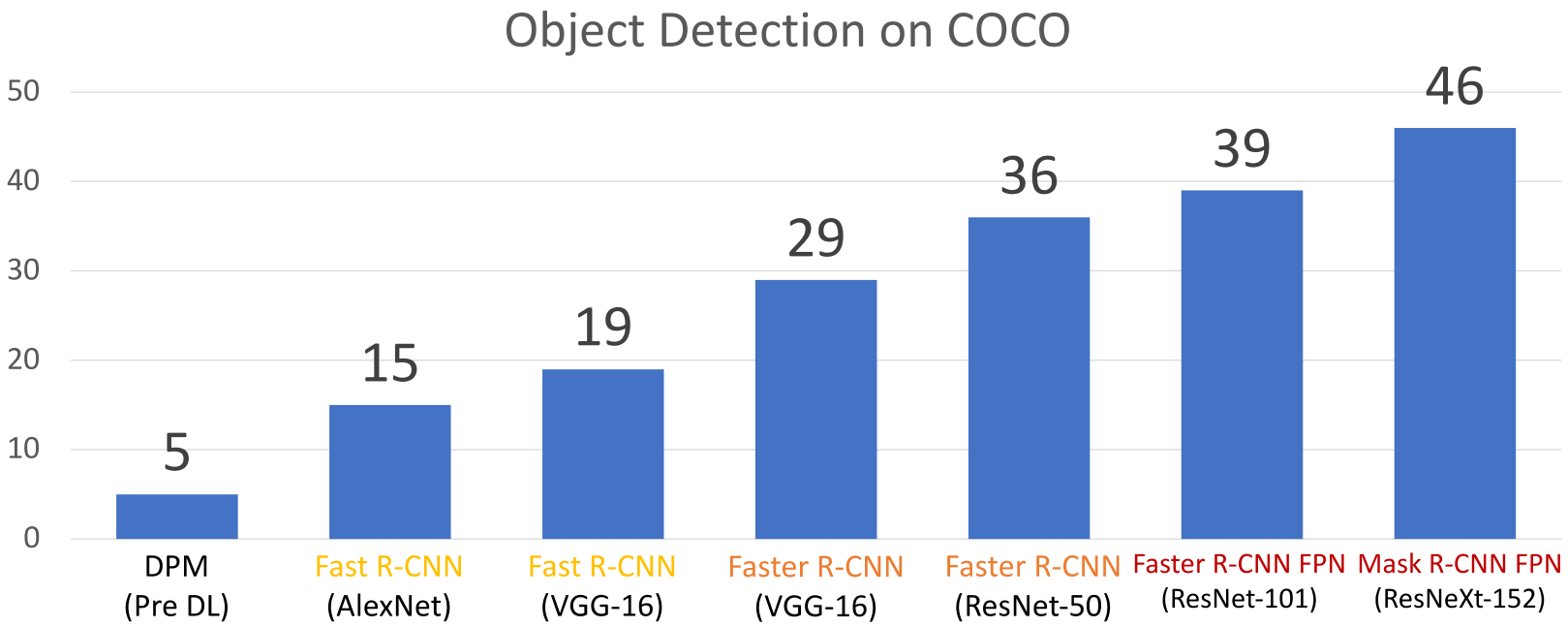
Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

[Slide credit: Justin Johnson]

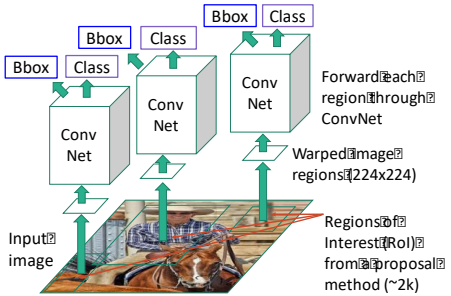
Overview: R-CNN Variants



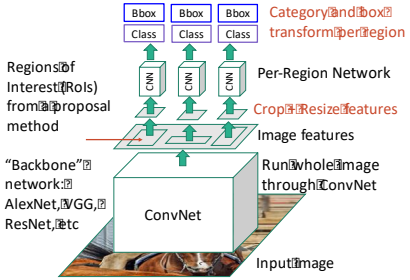
[Slide credit: Justin Johnson] [Ross Girshick, "The Generalized R-CNN Framework for Object Detection", ICCV 2017 Tutorial on Instance-Level Visual Recognition]

Overview: R-CNN Variants

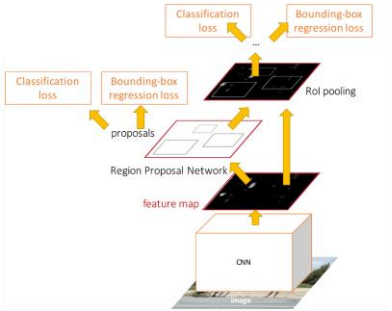
“Slow” R-CNN: Run CNN independently for each region



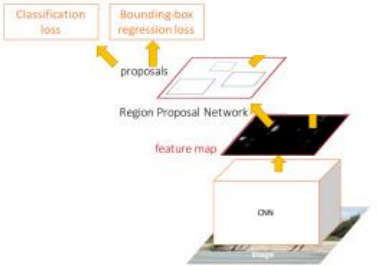
Fast R-CNN: Apply differentiable cropping to shared image features



Faster R-CNN: Compute proposals with CNN



Single-Stage: Fully convolutional detector



With anchors: RetinaNet
Anchor-Free: FCOS

[Slide credit: Justin Johnson]

Disclaimer

Many of the slides used here are obtained from online resources (including many open lecture materials) without appropriate acknowledgement. They are used here for the sole purpose of classroom teaching. All the credit and all the copyrights belong to the original authors. You should not copy it, redistribute it, put it online, or use it for any other purposes than for this course.