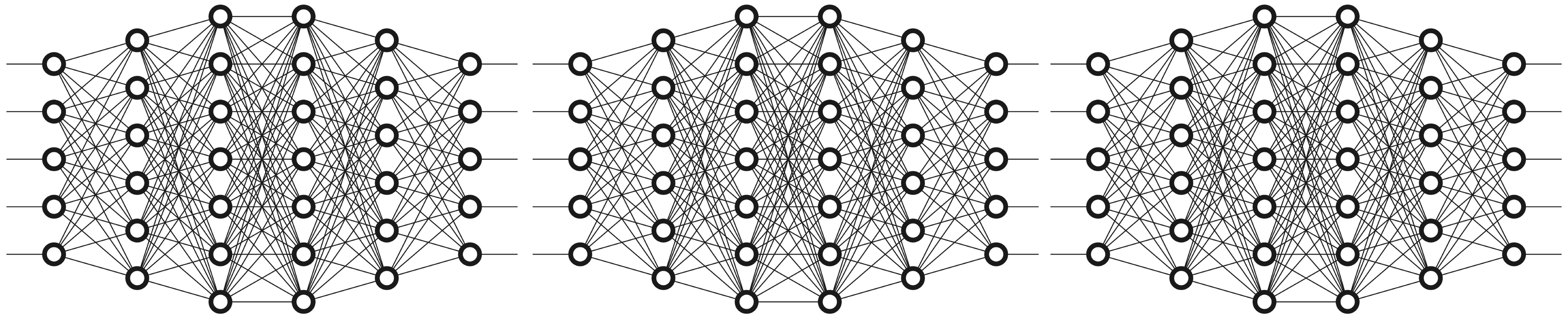
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 32: Applications of CNNs

Menu for Today (**November 27, 2020**)

Topics:

— Image classification with CNNs

— Object detection with CNNs

— Segmentation with CNNs

Redings:

— **Today's** Lecture: N/A

— **Next** Lecture: N/A

Reminders:

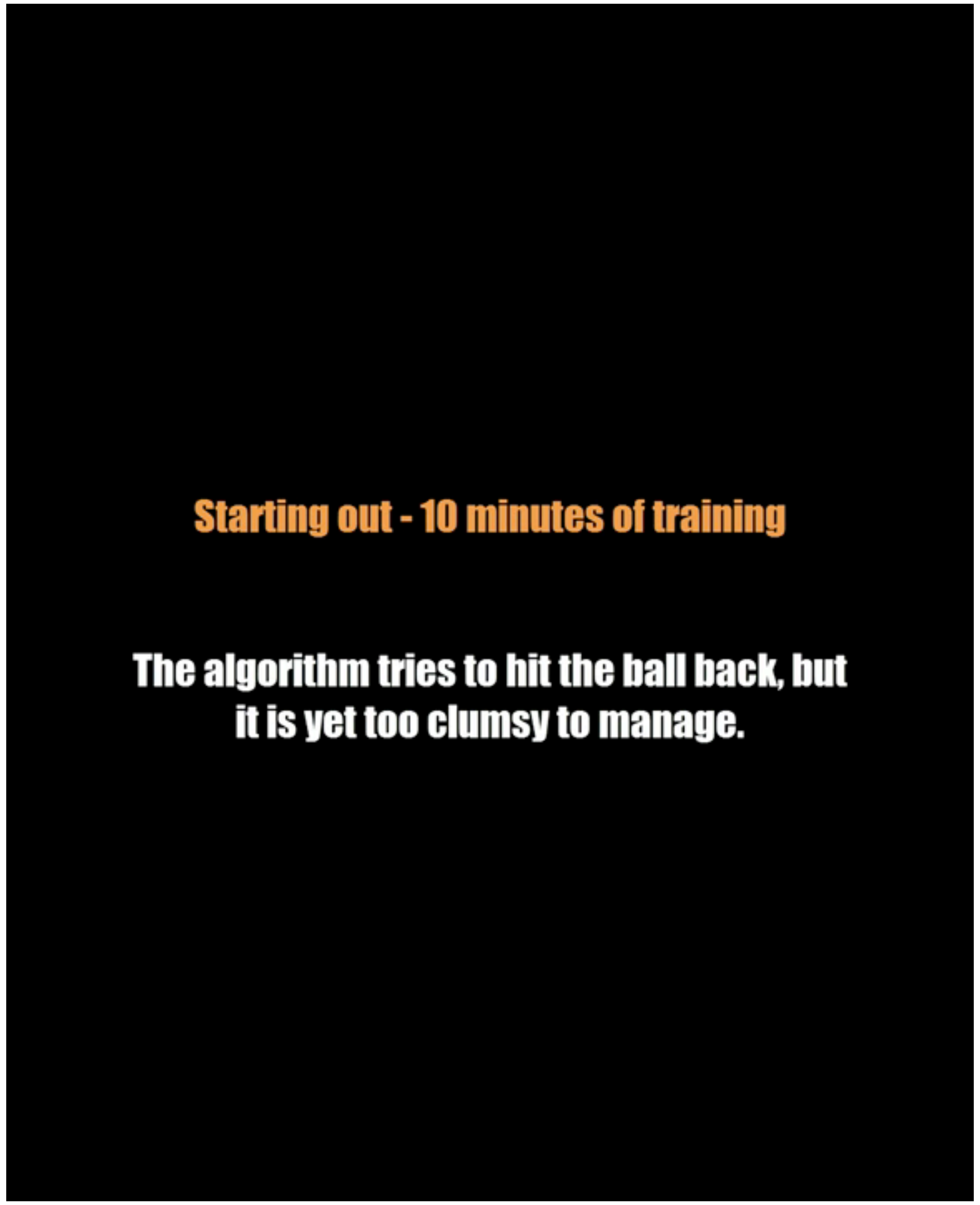
— **Assignment 6**: Deep Learning due **Wednsday, December 2nd**

— **Quiz 6** is **Monday**

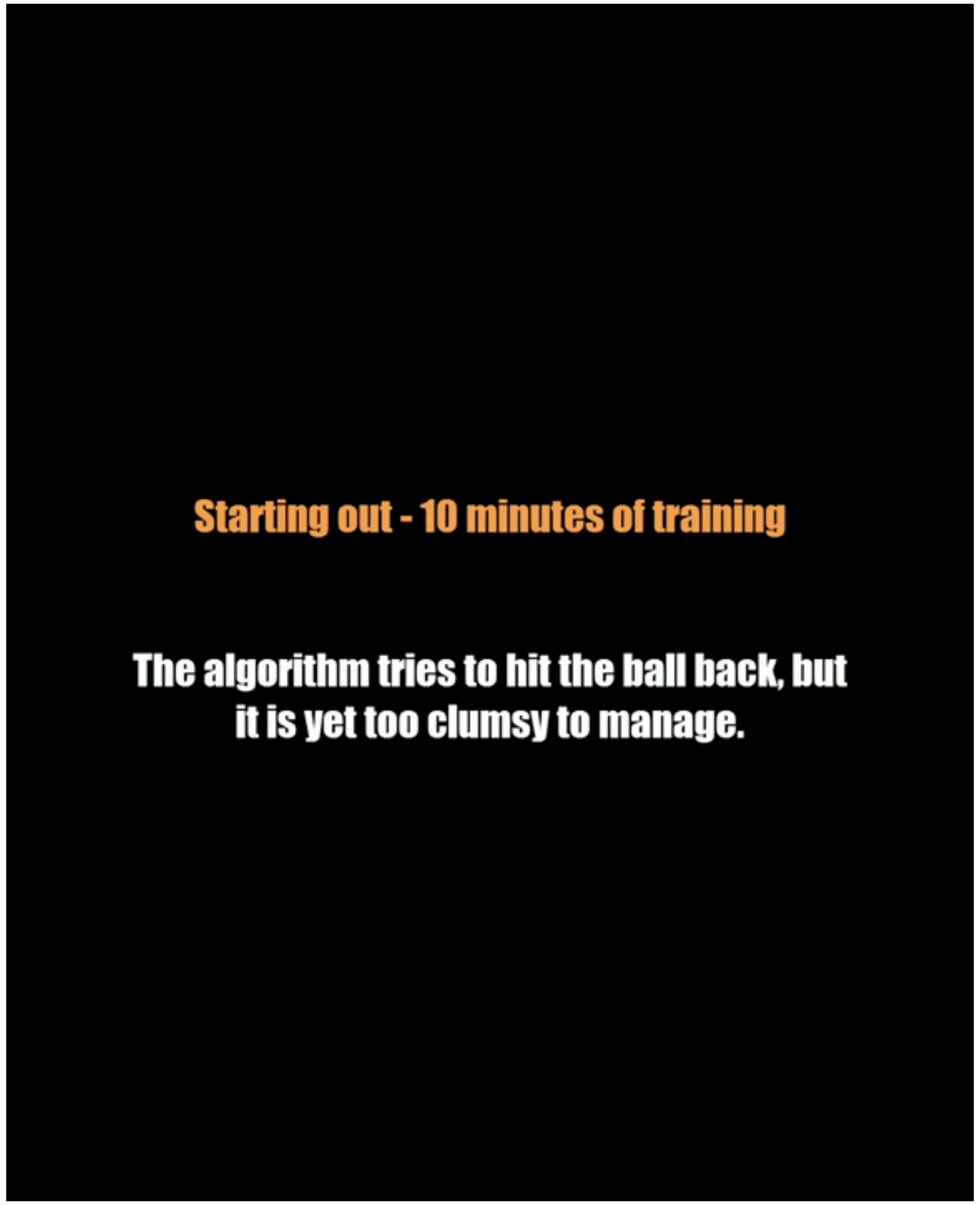
Today's "**fun**" Example: AlphaGo

Google DeepMind's AlphaGo

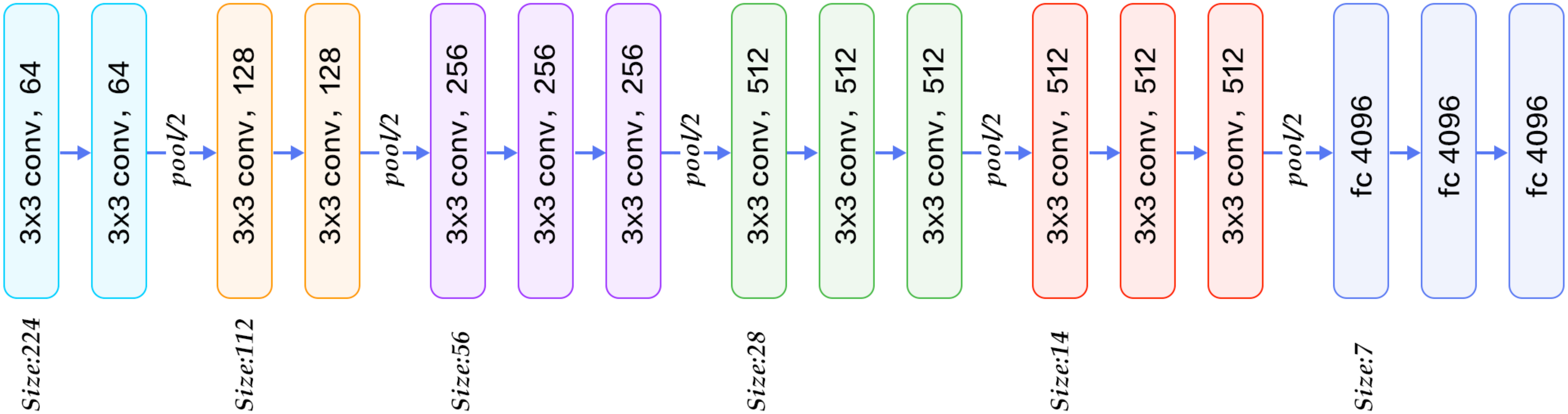
Today's "**fun**" Example: AlphaGo



Today’s “**fun**” Example: AlphaGo

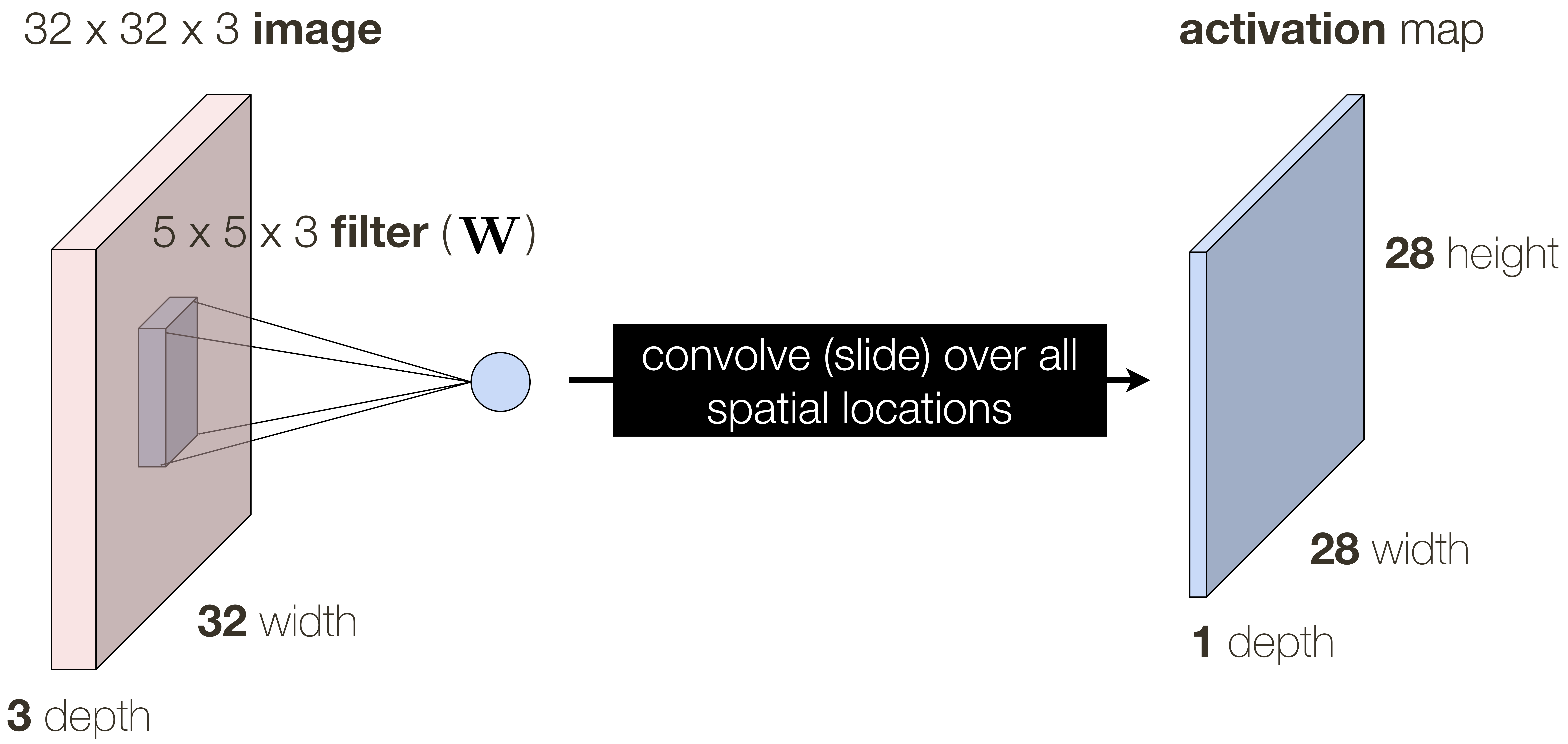


Convolutional Neural Networks



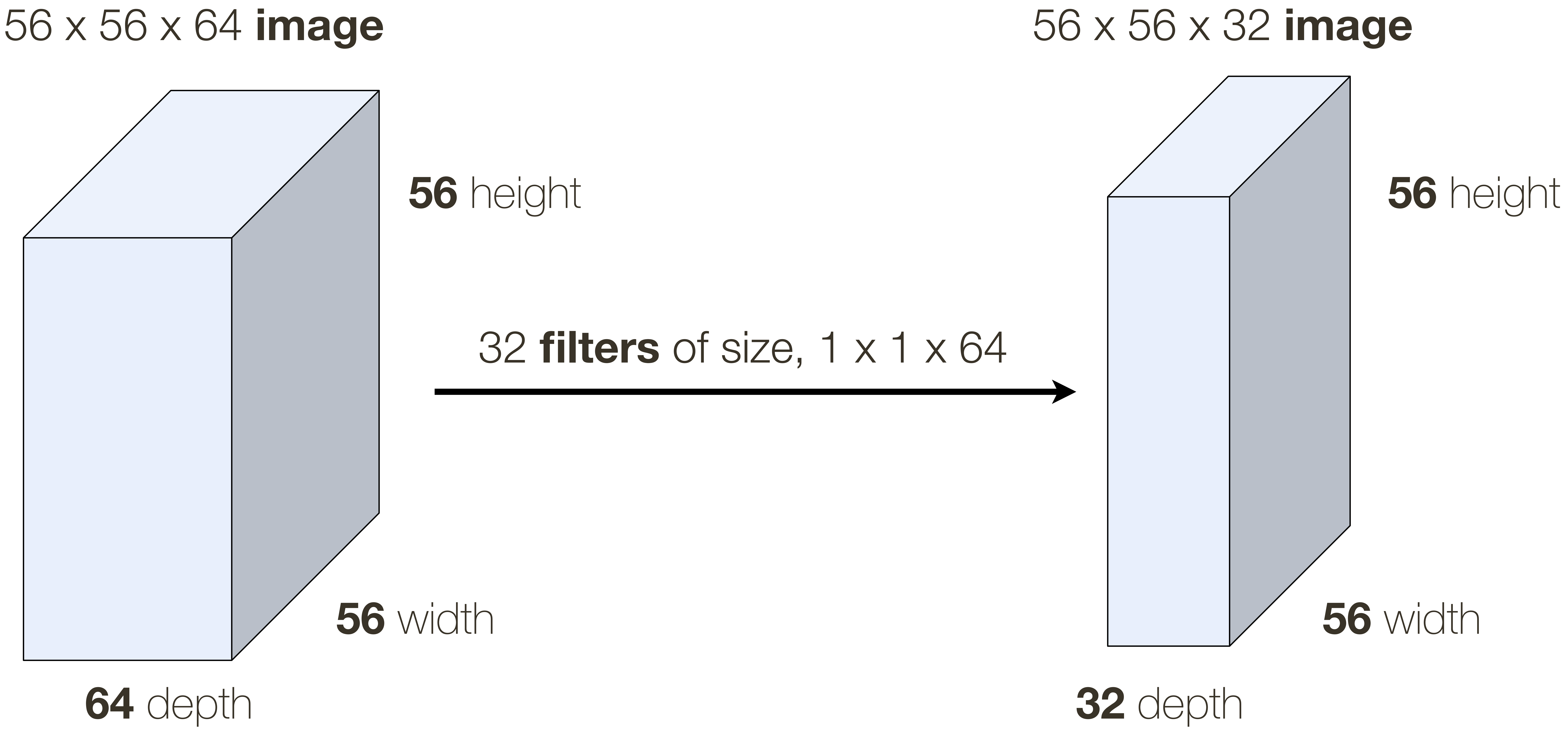
VGG-16 Network

Convolutional Layer: Closer Look at **Spatial Dimensions**



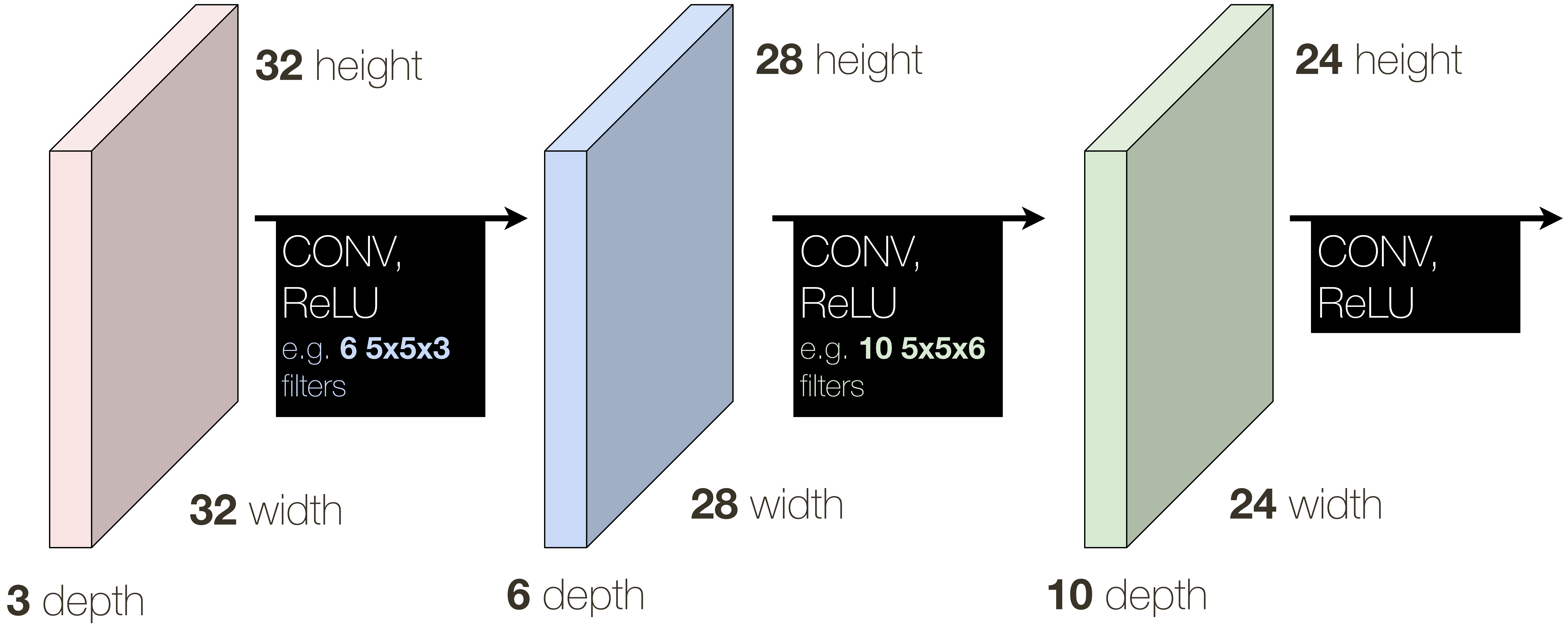
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer: **1x1 convolutions**



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$ (for mini-batch $N \times W_i \times H_i \times D_i$)

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$ (for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

— Number of filters: K (for typical networks $K \in \{32, 64, 128, 256, 512\}$)

— Spatial extent of filters: F (for a typical networks $F \in \{1, 3, 5, ...\}$)

— Stride of application: S (for a typical network $S \in \{1, 2\}$)

— Zero padding: P (for a typical network $P \in \{0, 1, 2\}$)

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$ (for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

- Number of filters: K (for typical networks $K \in \{32, 64, 128, 256, 512\}$)
- Spatial extent of filters: F (for a typical networks $F \in \{1, 3, 5, ...\}$)
- Stride of application: S (for a typical network $S \in \{1, 2\}$)
- Zero padding: P (for a typical network $P \in \{0, 1, 2\}$)

Produces a volume of size: $W_o \times H_o \times D_o$ (for mini-batch $N \times W_o \times H_o \times D_o$)

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$ (for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

— Number of filters: K (for typical networks $K \in \{32, 64, 128, 256, 512\}$)

— Spatial extent of filters: F (for a typical networks $F \in \{1, 3, 5, ...\}$)

— Stride of application: S (for a typical network $S \in \{1, 2\}$)

— Zero padding: P (for a typical network $P \in \{0, 1, 2\}$)

Produces a volume of size: $W_o \times H_o \times D_o$ (for mini-batch $N \times W_o \times H_o \times D_o$)

$$W_o = (W_i - F + 2P)/S + 1 \qquad H_o = (H_i - F + 2P)/S + 1 \qquad D_o = K$$

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$ (for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

— Number of filters: K (for typical networks $K \in \{32, 64, 128, 256, 512\}$)

— Spatial extent of filters: F (for a typical networks $F \in \{1, 3, 5, ...\}$)

— Stride of application: S (for a typical network $S \in \{1, 2\}$)

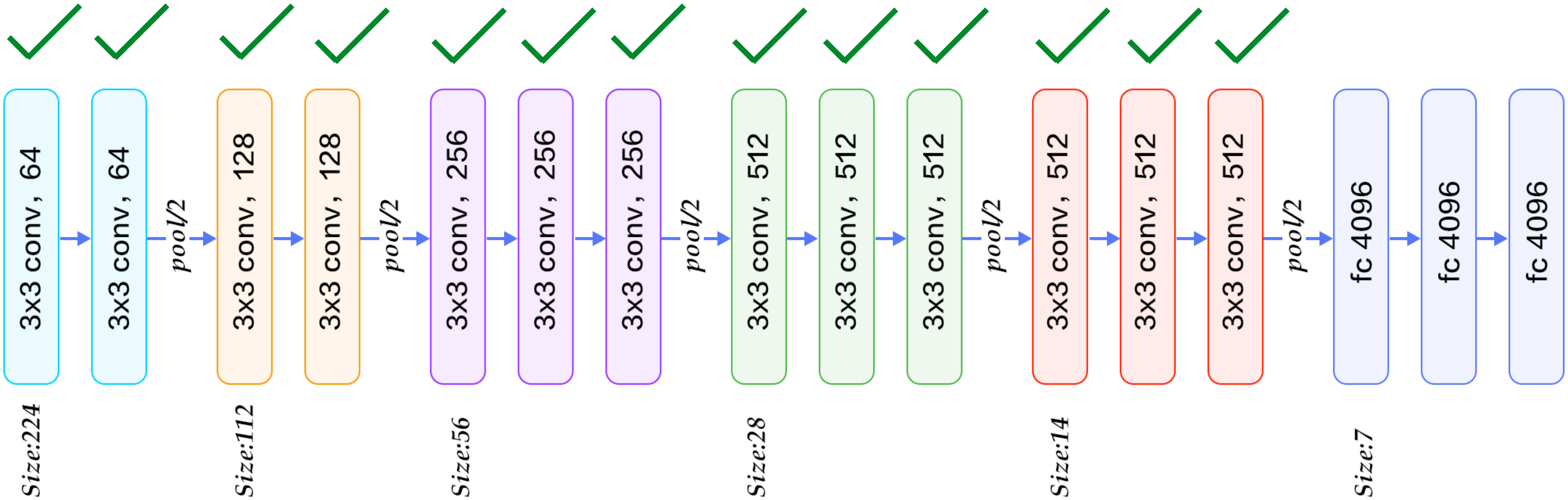
— Zero padding: P (for a typical network $P \in \{0, 1, 2\}$)

Produces a volume of size: $W_o \times H_o \times D_o$ (for mini-batch $N \times W_o \times H_o \times D_o$)

$$W_o = (W_i - F + 2P)/S + 1 \qquad H_o = (H_i - F + 2P)/S + 1 \qquad D_o = K$$

Number of total learnable parameters: $(F \times F \times D_i) \times K + K$

Convolutional Neural Networks



VGG-16 Network

CNNs: Reminder Fully Connected Layers

Input

Activation

$$\mathbf{W}^T\mathbf{x} + \mathbf{b}, \text{where } \mathbf{W} \in \mathbb{R}^{10\times 3072}$$

3072

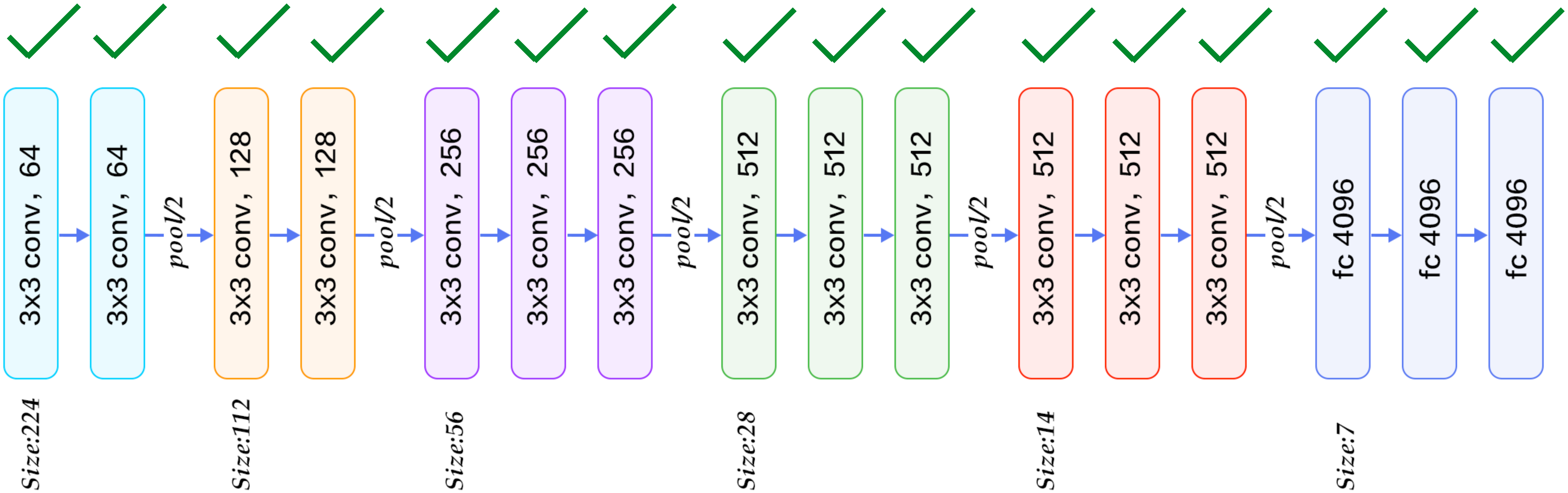
(32 x 32 x 3 image -> stretches to 3072 x 1)

each neuron looks at the full input volume

10

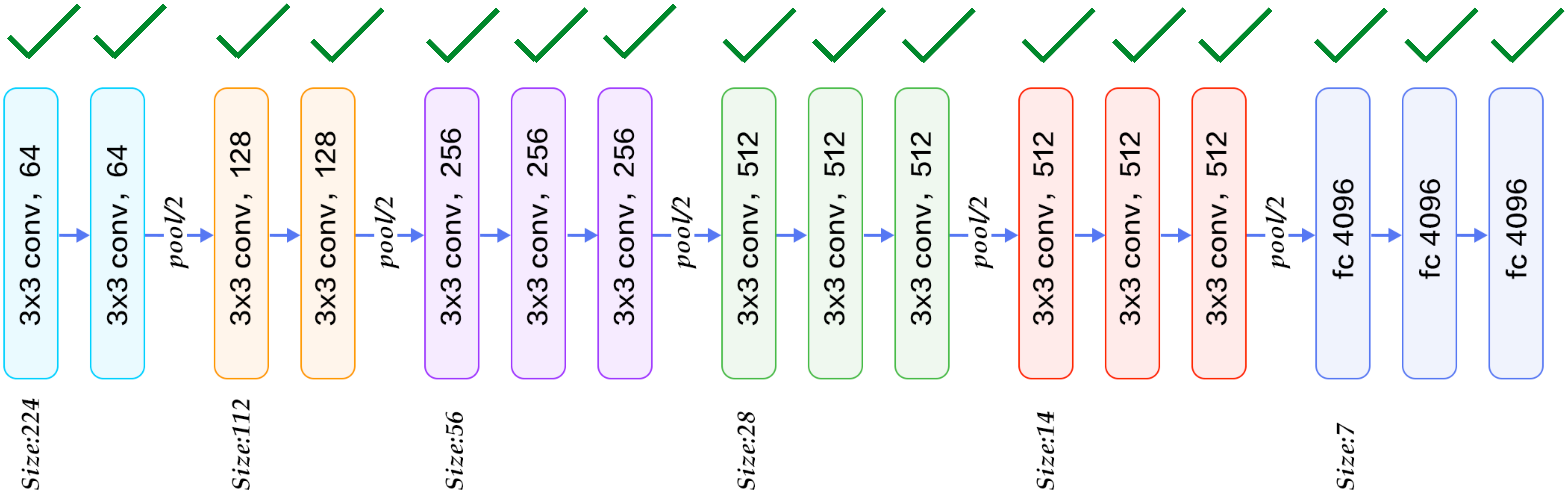
* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Networks



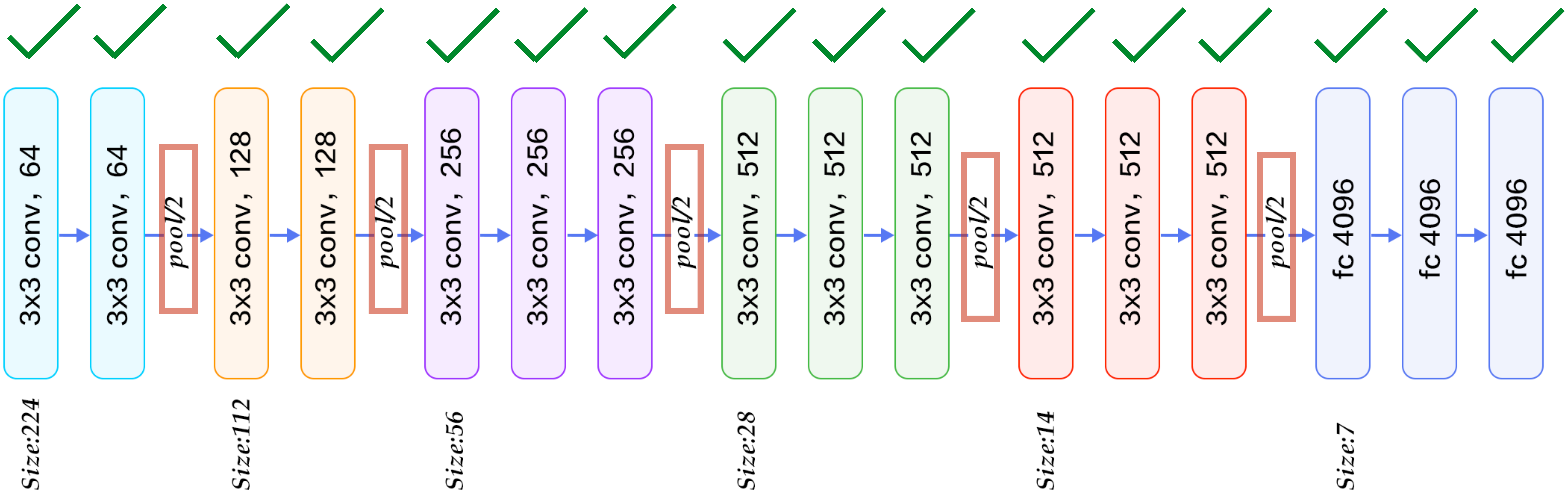
VGG-16 Network

Convolutional Neural Networks



VGG-16 Network

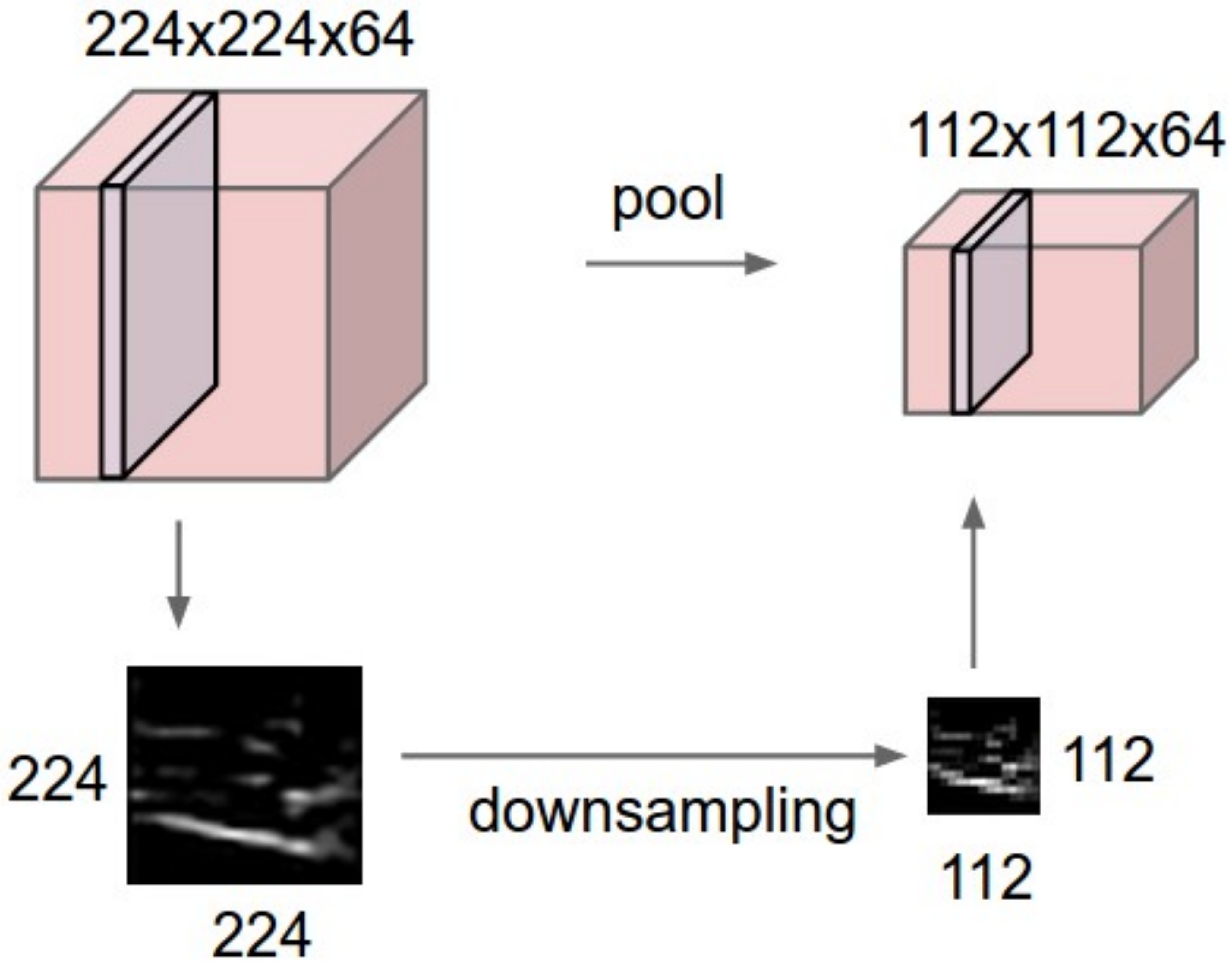
Convolutional Neural Networks



VGG-16 Network

Pooling Layer

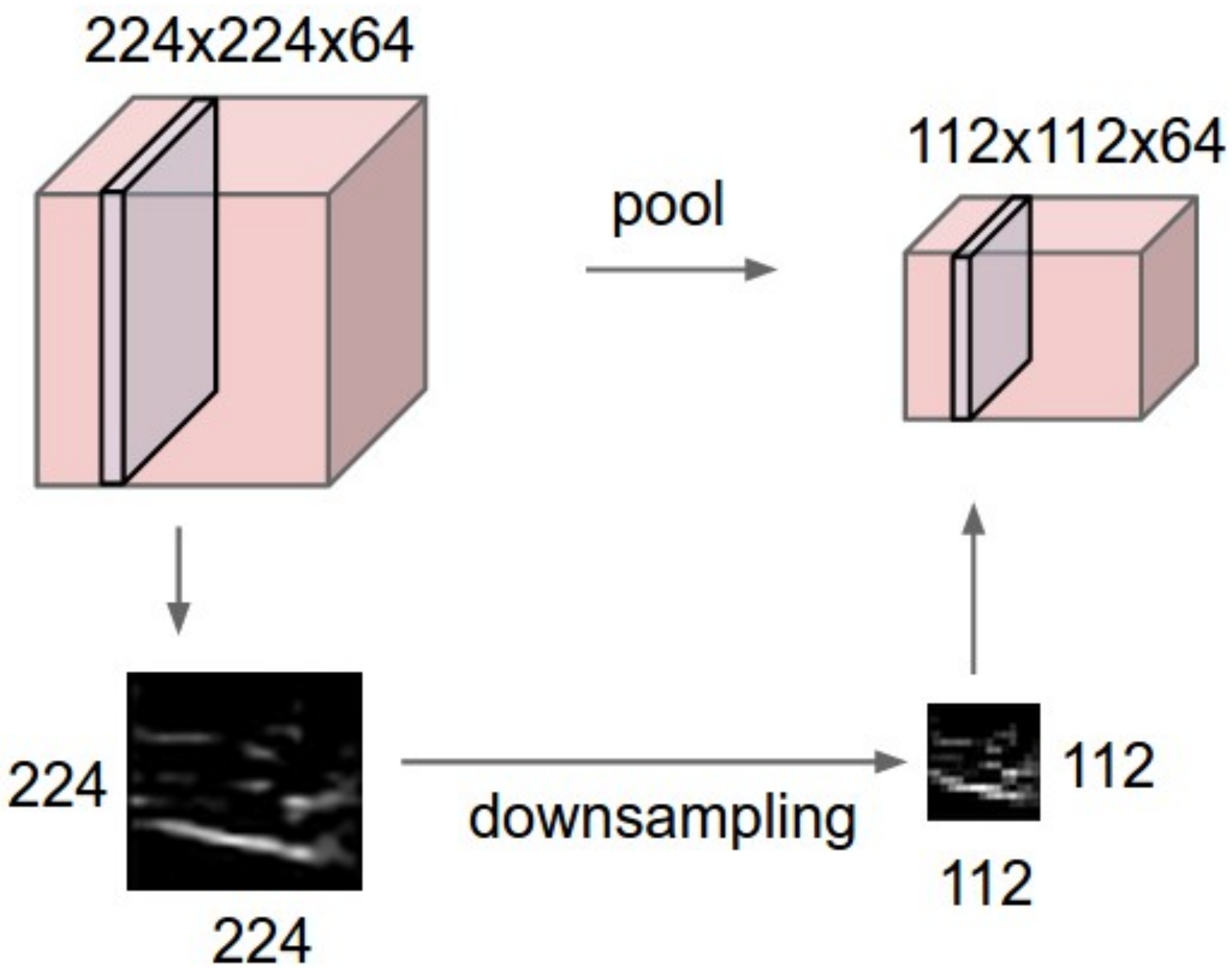
- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Pooling Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently

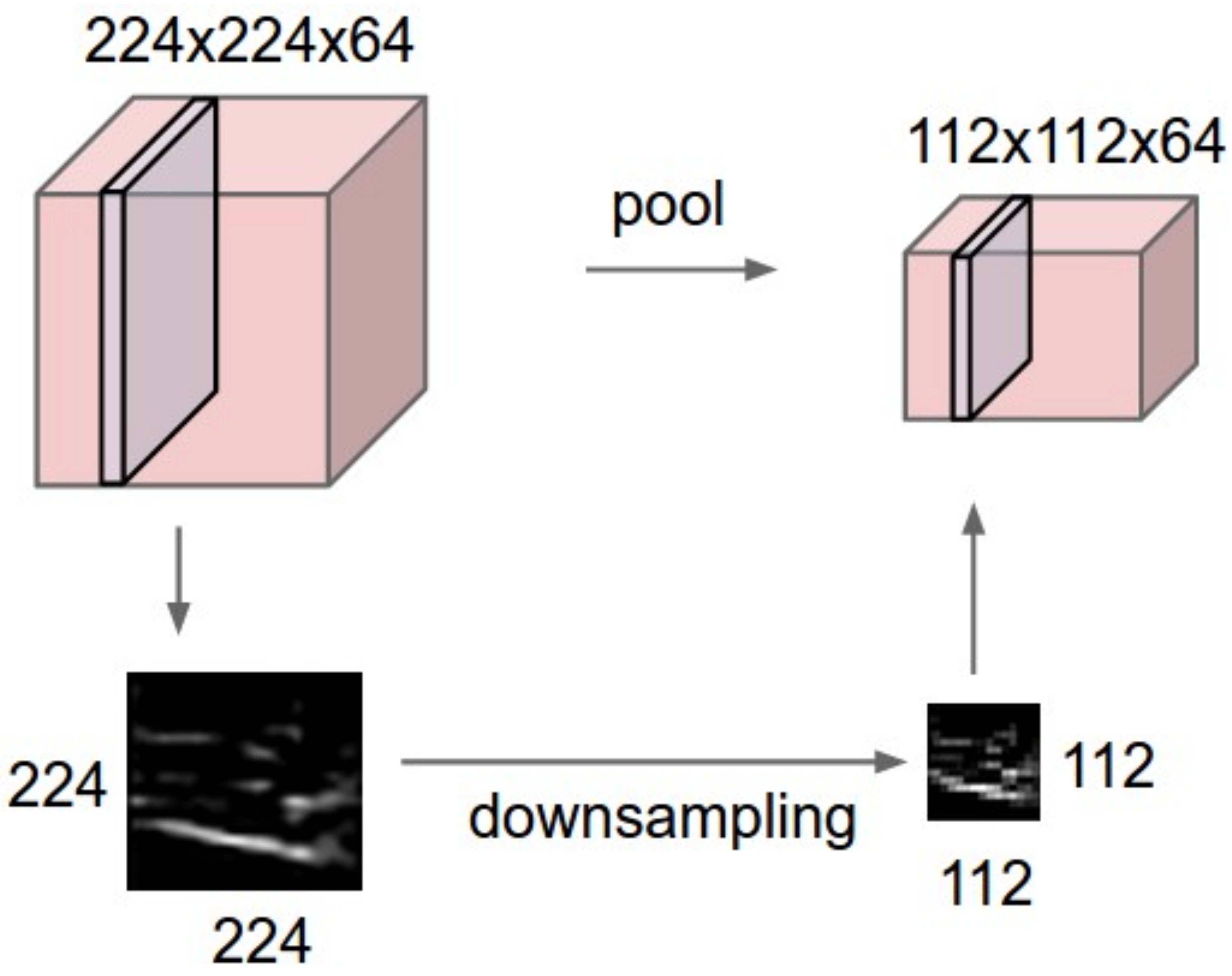


How many **parameters**?

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Pooling Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently



How many **parameters**?

None!

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Max **Pooling**

activation map

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

max pool with 2 x 2 filter
and stride of 2

6	8
3	4

Average **Pooling**

activation map

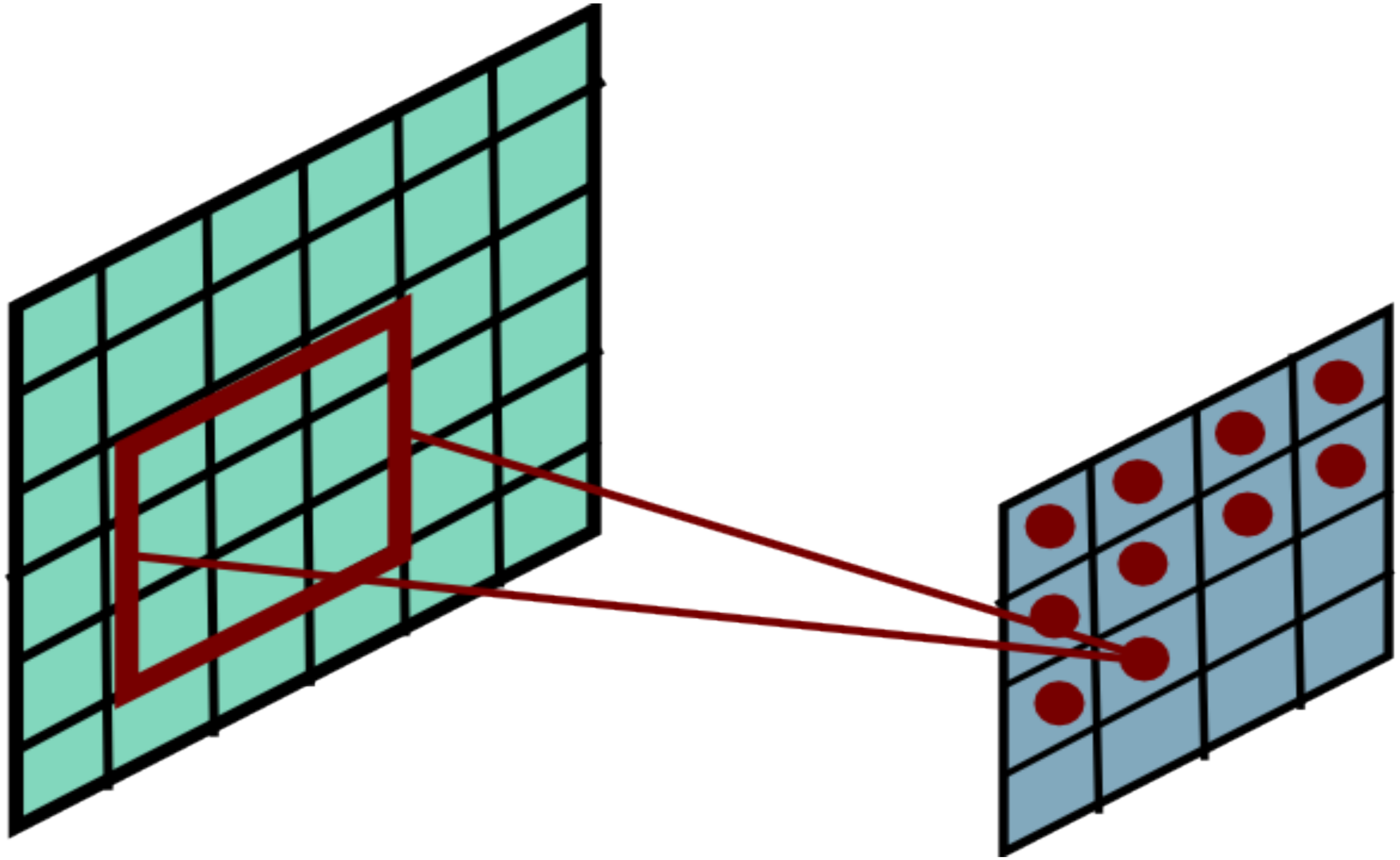
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

avg pool with 2 x 2 filter
and stride of 2

3.25	5.25
2	2

Pooling Layer **Receptive Field**

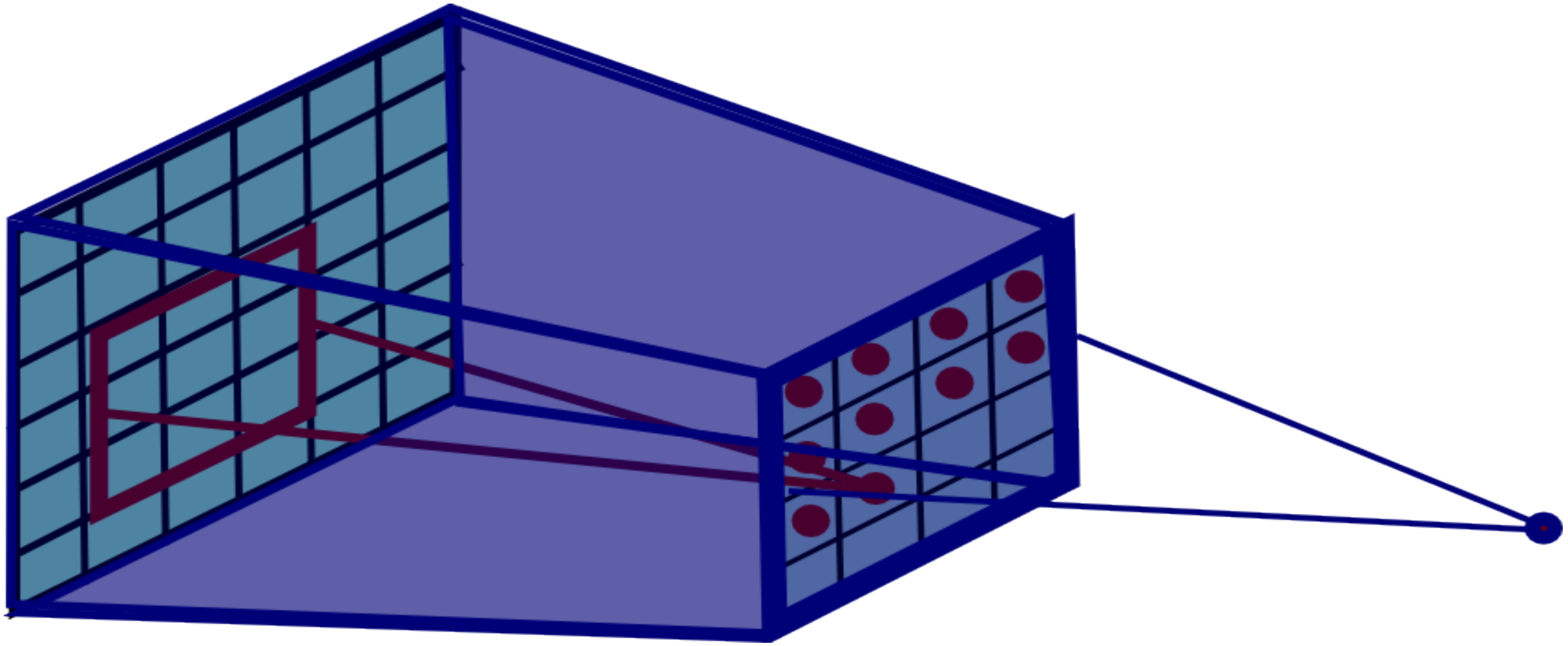
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: **(P+K-1)x(P+K-1)**



* slide from Marc'Aurelio Renzato

Pooling Layer **Receptive Field**

If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: **(P+K-1)x(P+K-1)**



* slide from Marc'Aurelio Renzato

Pooling Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$ (for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

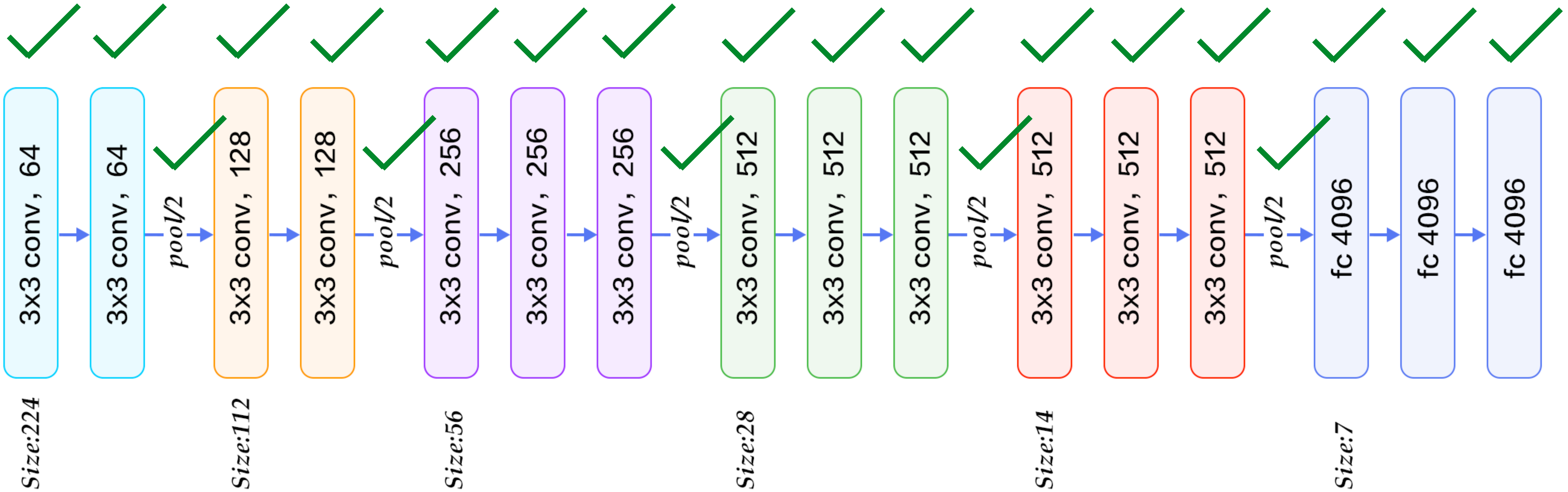
- Spatial extent of filters: K
- Stride of application: F

Produces a volume of size: $W_o \times H_o \times D_o$

$$W_o = (W_i - F)/S + 1 \qquad H_o = (H_i - F)/S + 1 \qquad D_o = D_i$$

Number of total learnable parameters: 0 (for mini-batch $N \times W_o \times H_o \times D_o$)

Convolutional Neural Networks



VGG-16 Network

Computer **Vision Problems**

Computer **Vision Problems**

Categorization

Computer **Vision Problems**

Categorization

Single-**label:** Horse
Church
Toothbrush
Person

IMAGENET

Computer **Vision Problems**

Categorization

Single-**label:** Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Computer **Vision Problems**

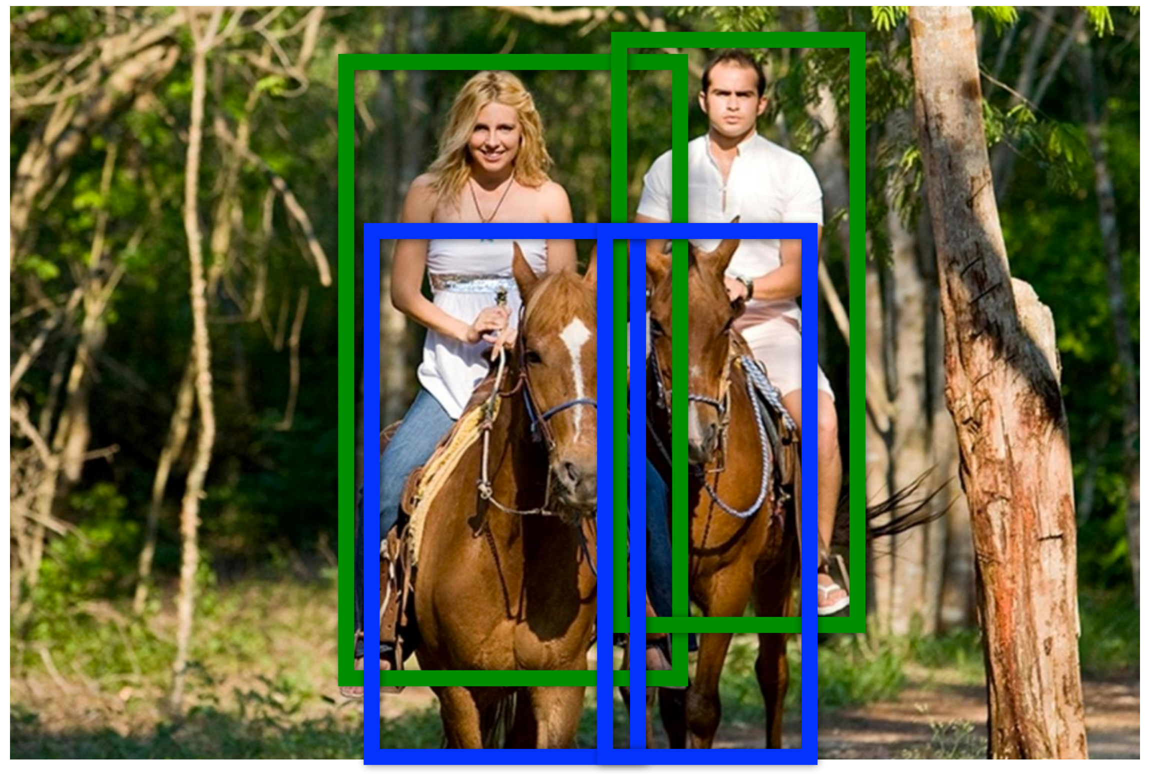
Categorization

Single-**label:** Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

Computer **Vision Problems**

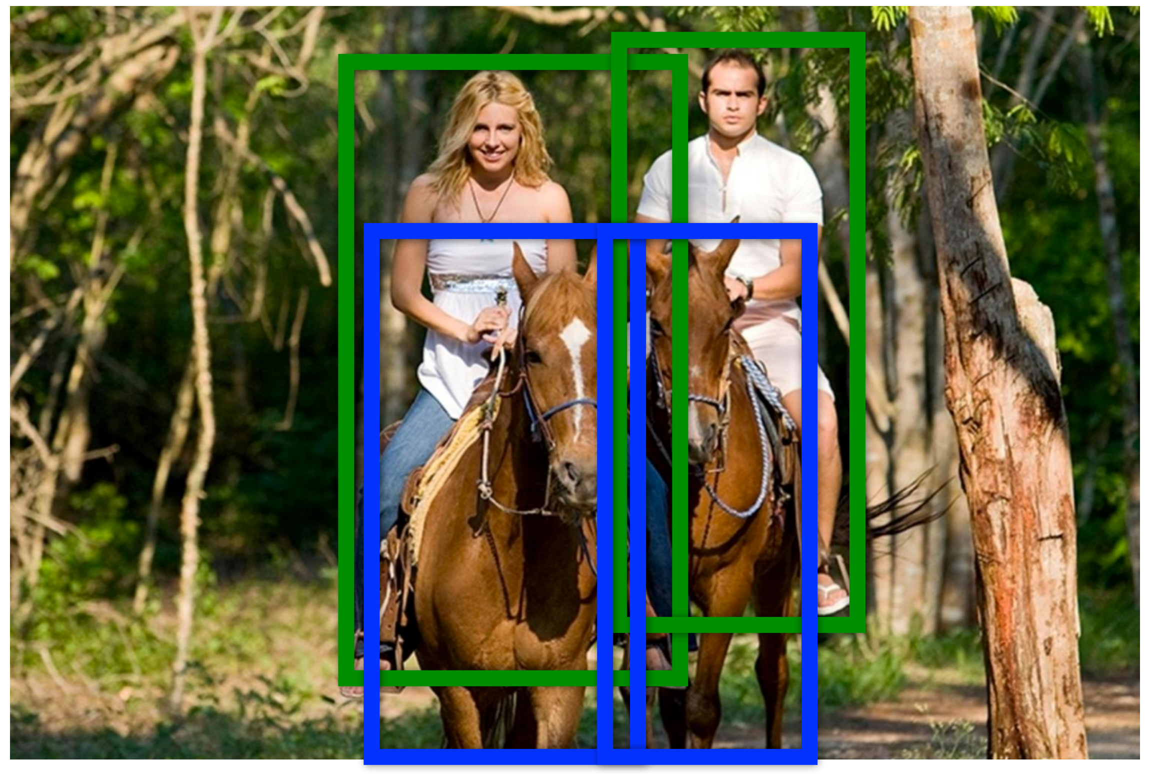
Categorization

Single-**label:** Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

Segmentation

Horse
Person

COCO
Common Objects in Context

Computer **Vision Problems**

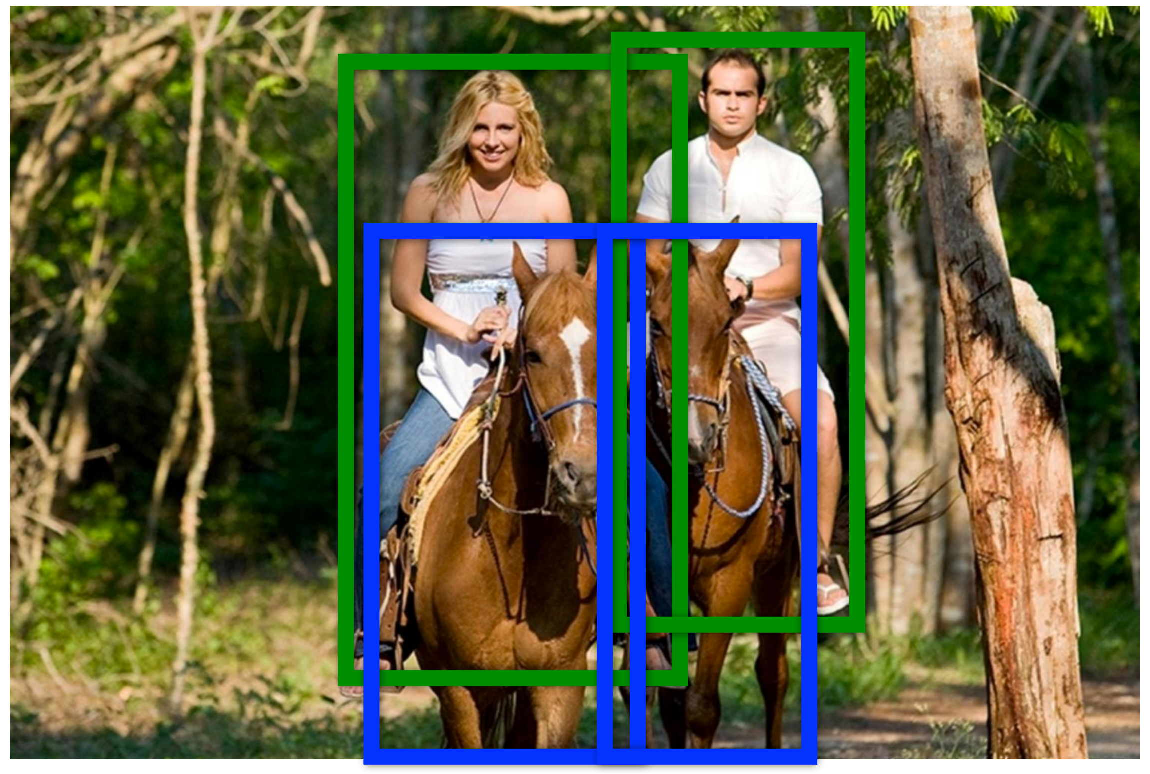
Categorization

Single-**label**: Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

Segmentation

Horse
Person

COCO
Common Objects in Context

Instance Segmentation

Horse1
Horse2
Person1
Person2

Computer **Vision Problems**

Categorization

Single-**label:** Horse
Church
Toothbrush
Person

IMAGENET

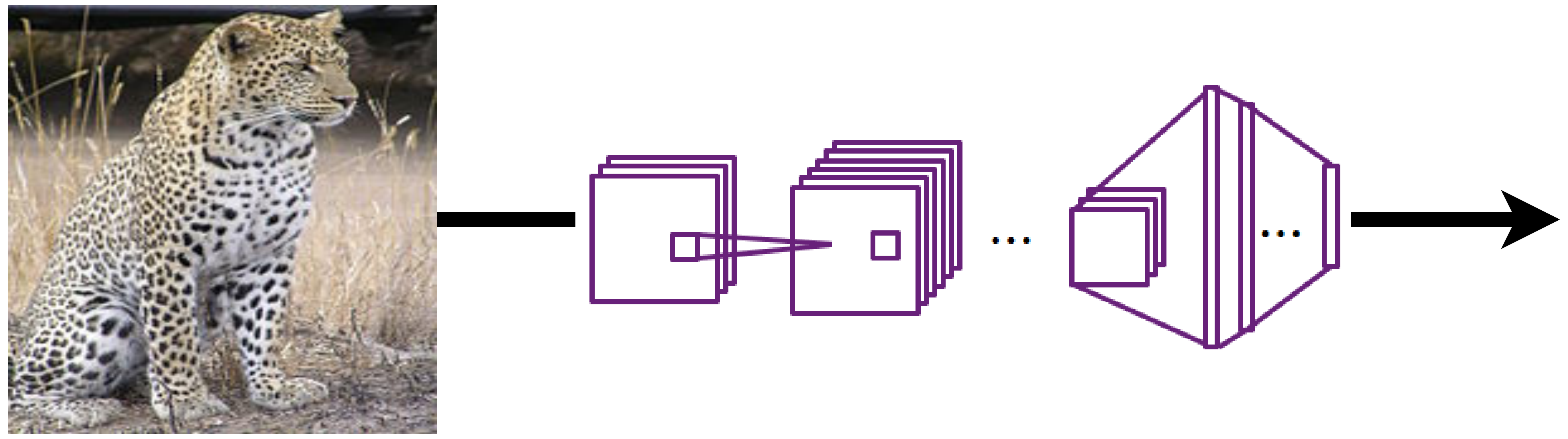
Multi-**label**: **Horse**
Church
Toothbrush
Person

Object **Classification**

Category	Prediction
Dog	No
Cat	No
Couch	No
Flowers	No
Leopard	**Yes**
…	…

Problem: For each image predict which category it belongs to out of a fixed set

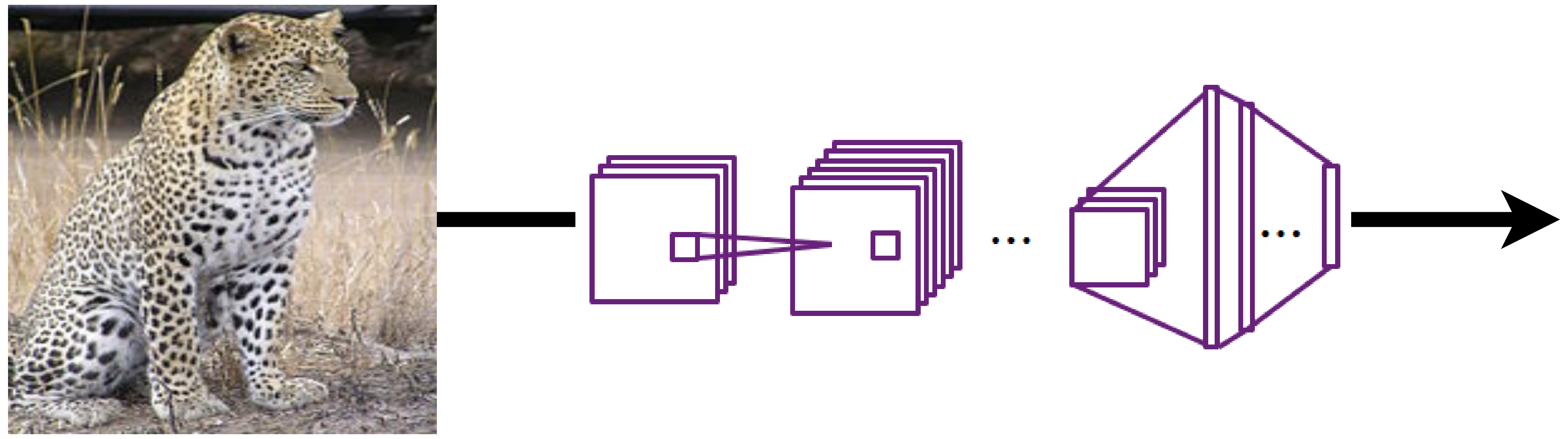
Object **Classification**

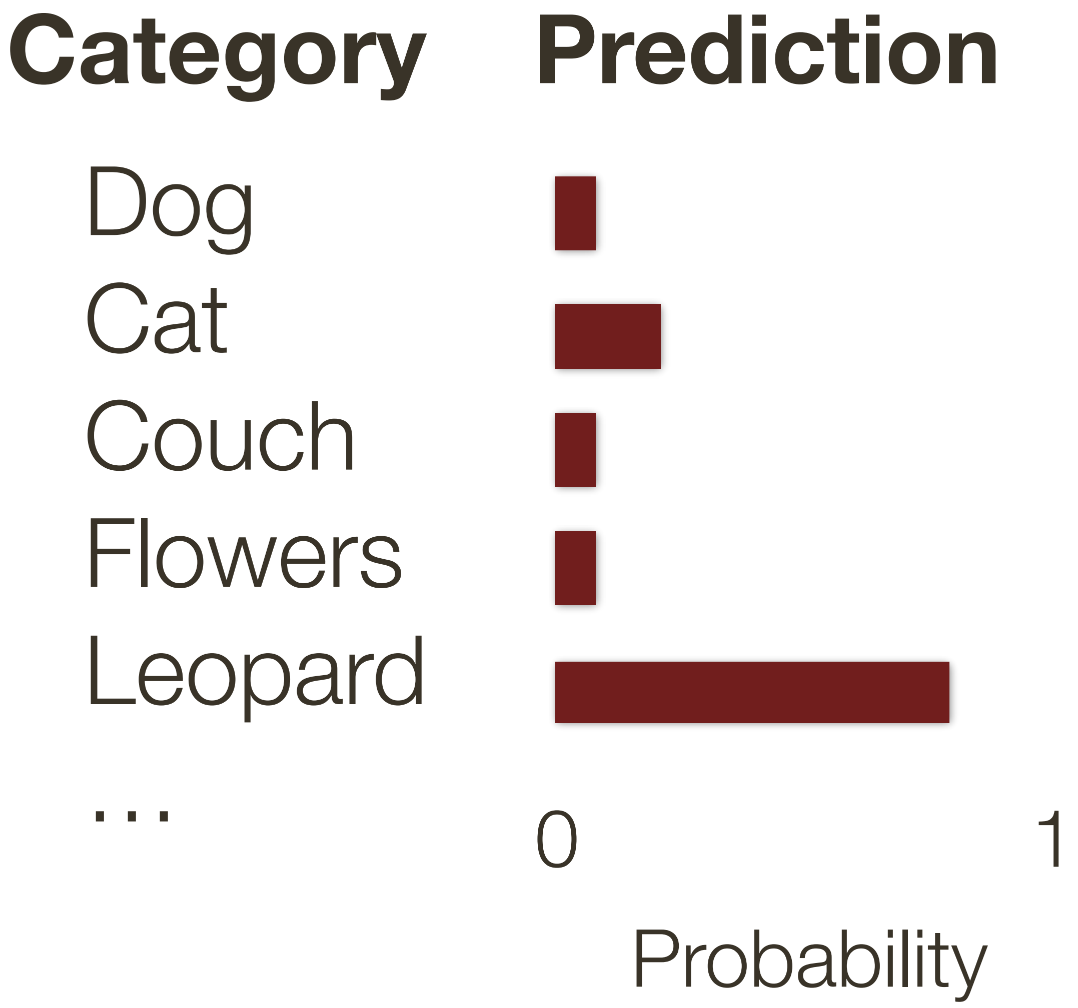


Category	Prediction
Dog	No
Cat	No
Couch	No
Flowers	No
Leopard	**Yes**
…	…

Problem: For each image predict which category it belongs to out of a fixed set

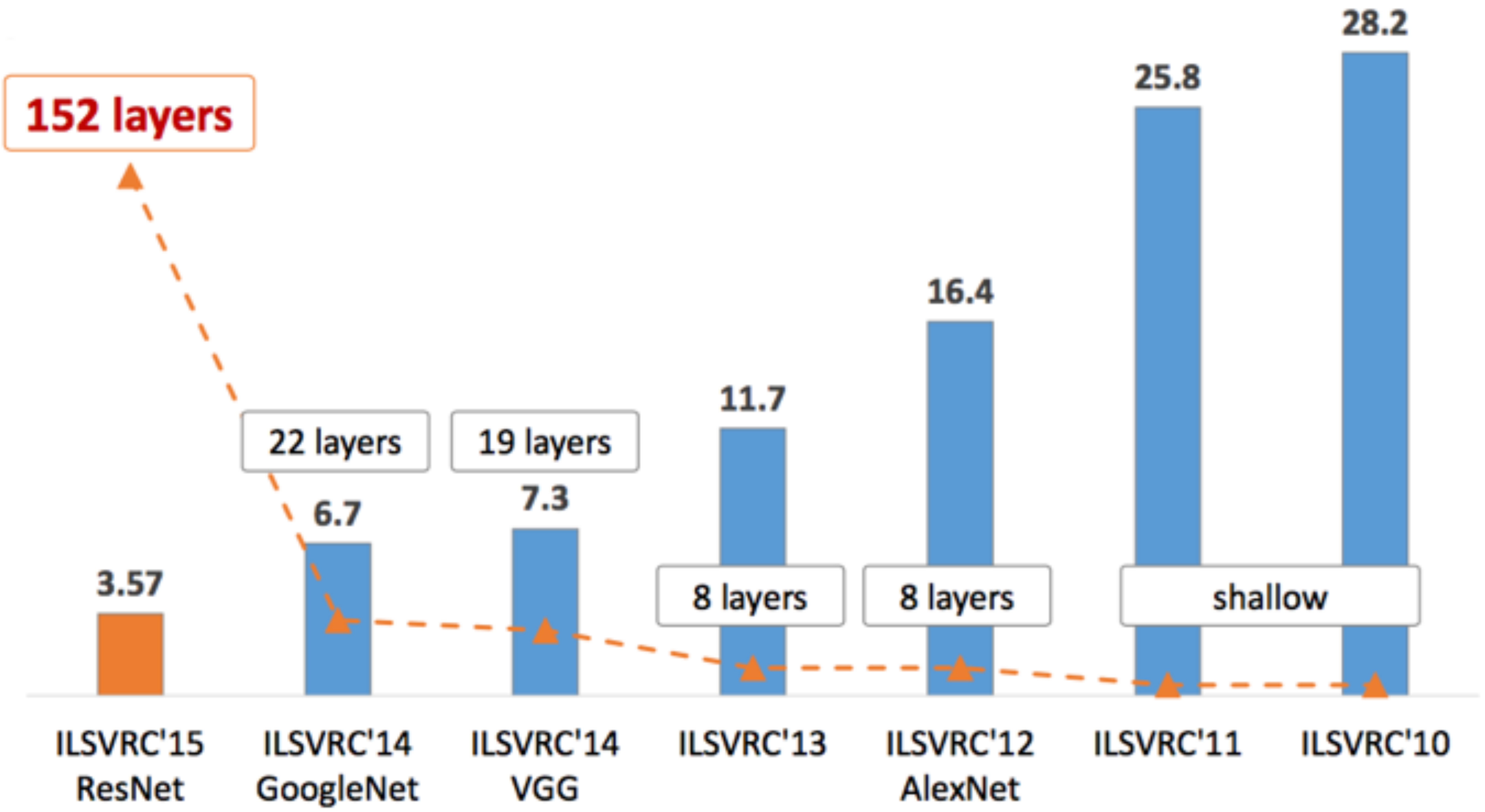
Object **Classification**



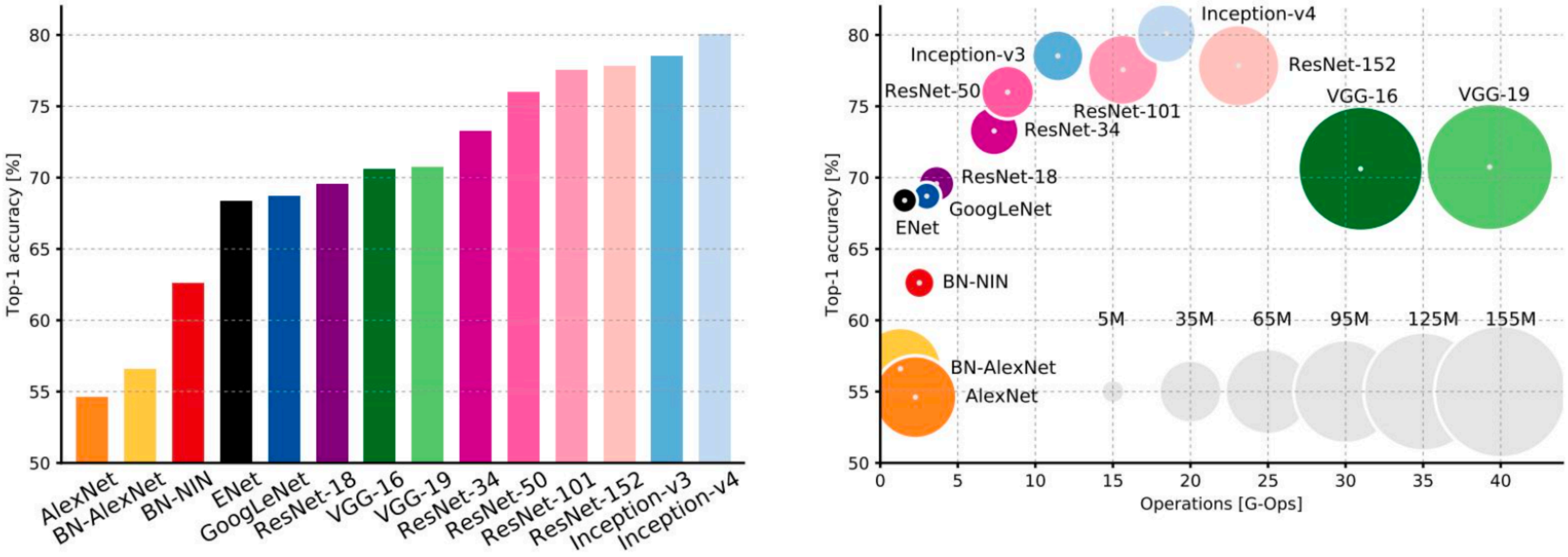


Problem: For each image predict which category it belongs to out of a fixed set

Object **Classification**



Comparing **Complexity**



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Computer **Vision Problems** (no language for now)

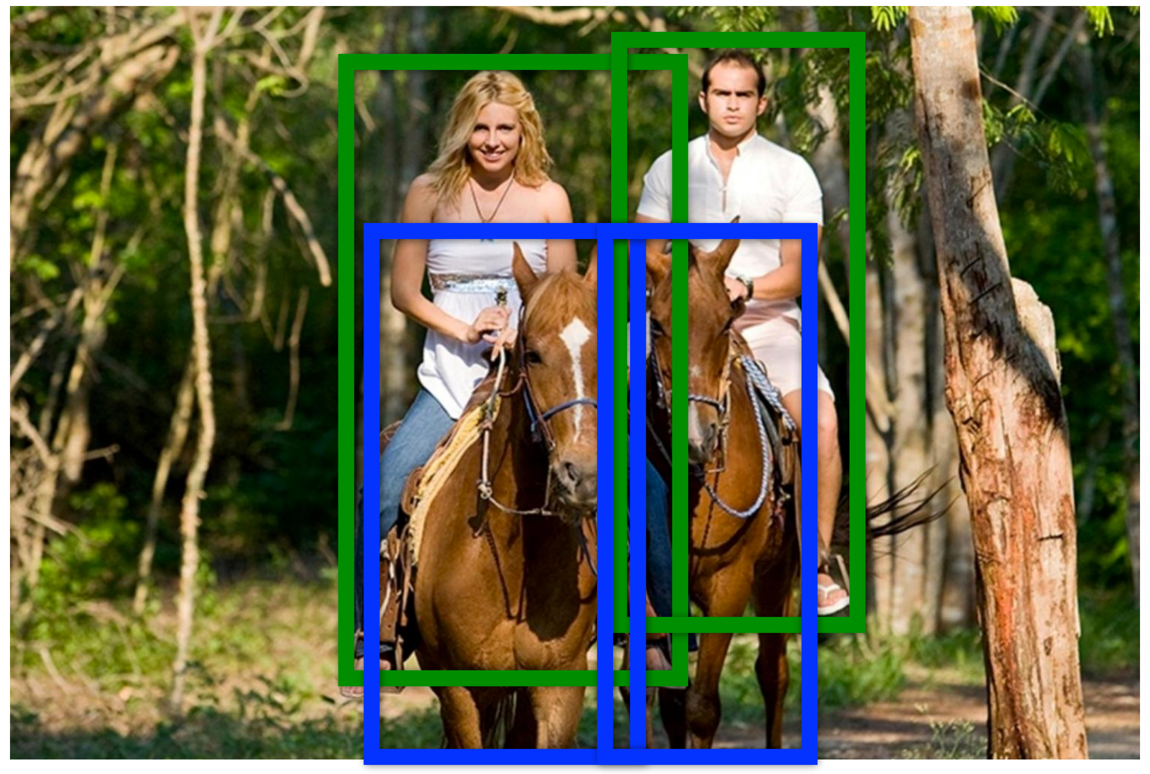
Categorization

Single-**label**: Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO Common Objects in Context

Segmentation

Horse
Person

COCO Common Objects in Context

Instance Segmentation

Horse1
Horse2
Person1
Person2

Computer **Vision Problems** (no language for now)

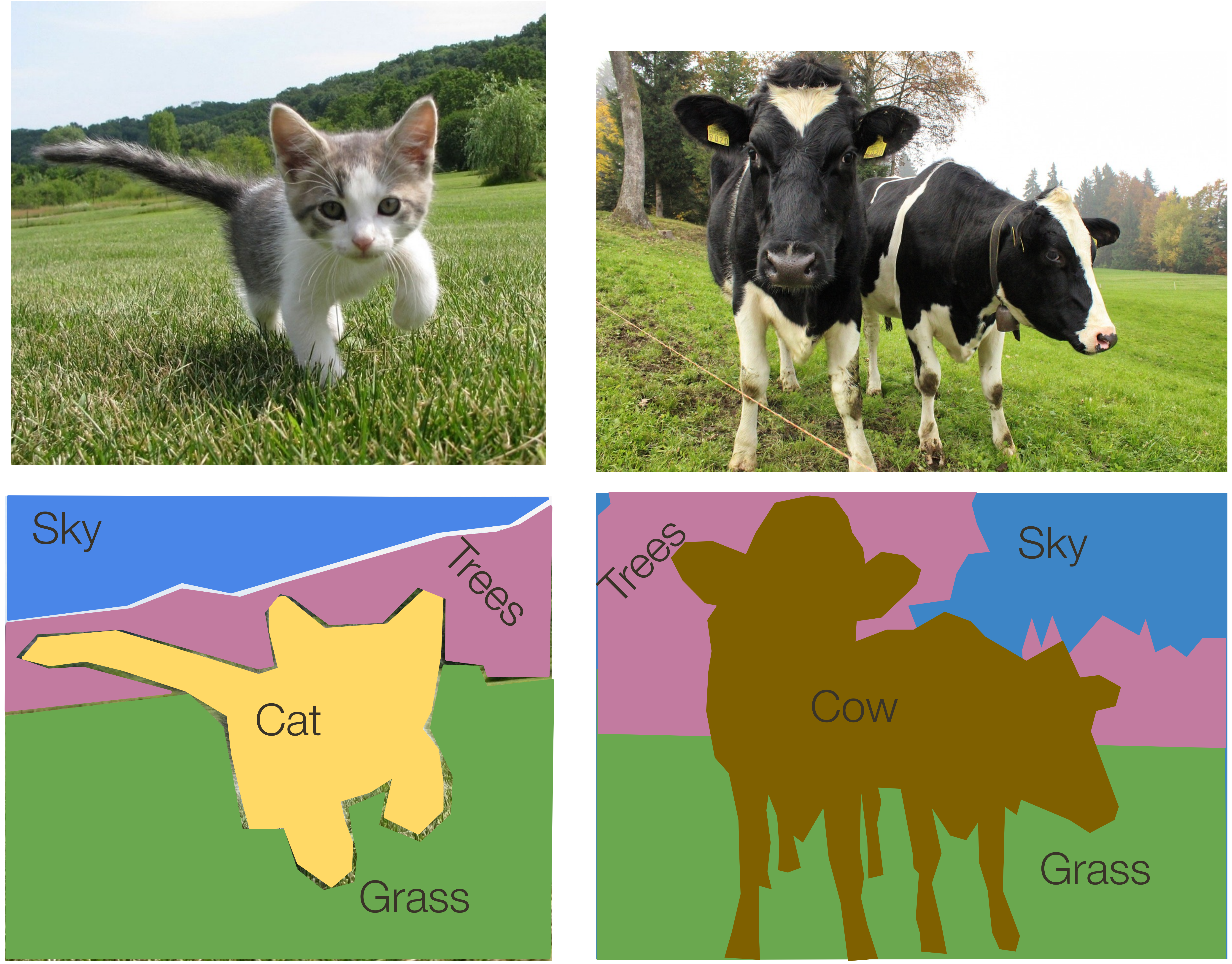
Segmentation

Horse
Person

COCO
Common Objects in Context

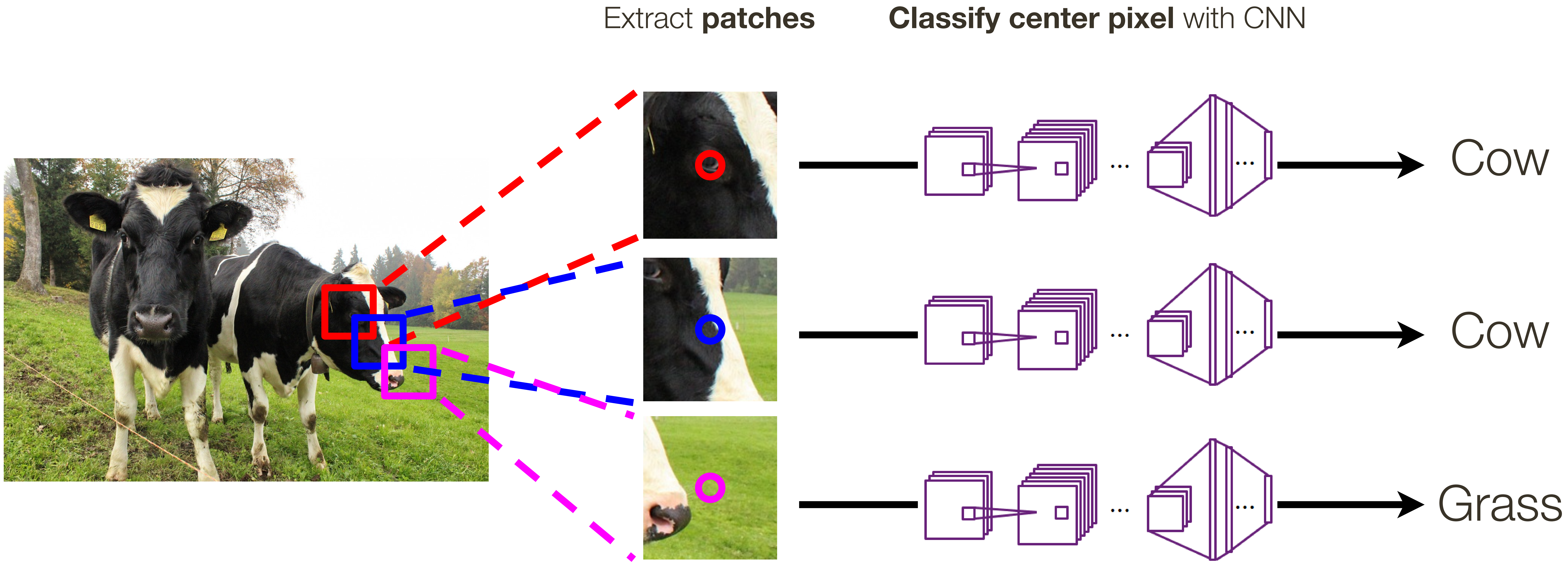
Semantic **Segmentation**

Label **every pixel** with a category label (without differentiating instances)



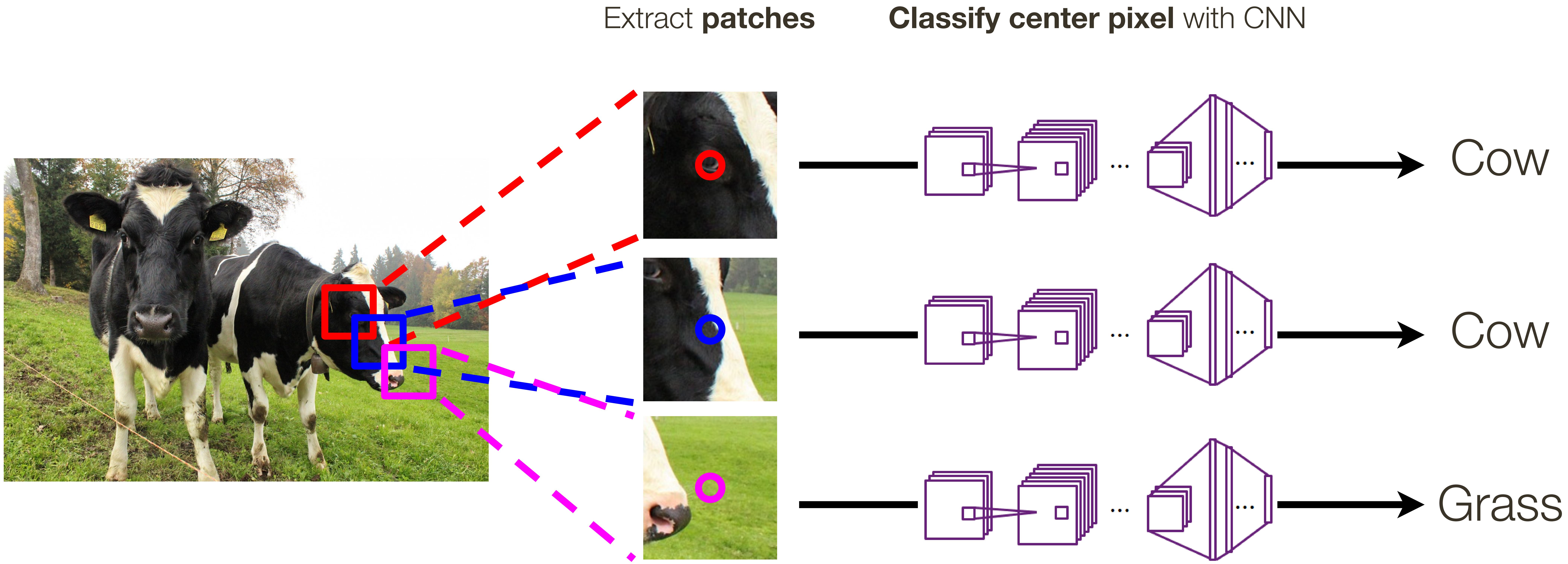
Semantic **Segmentation:** Sliding Window

[Farabet et al, TPAMI 2013]
[Pinheiro et al, ICML 2014]



Semantic **Segmentation:** Sliding Window

[Farabet et al, TPAMI 2013]
[Pinheiro et al, ICML 2014]

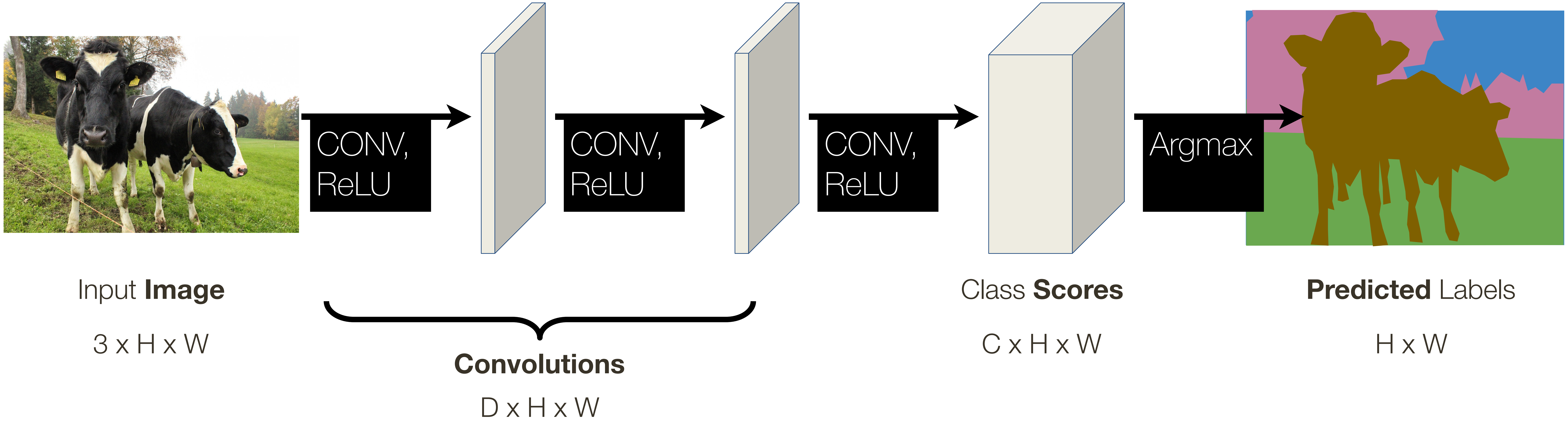


Problem: VERY inefficient, no reuse of computations for overlapping patches

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

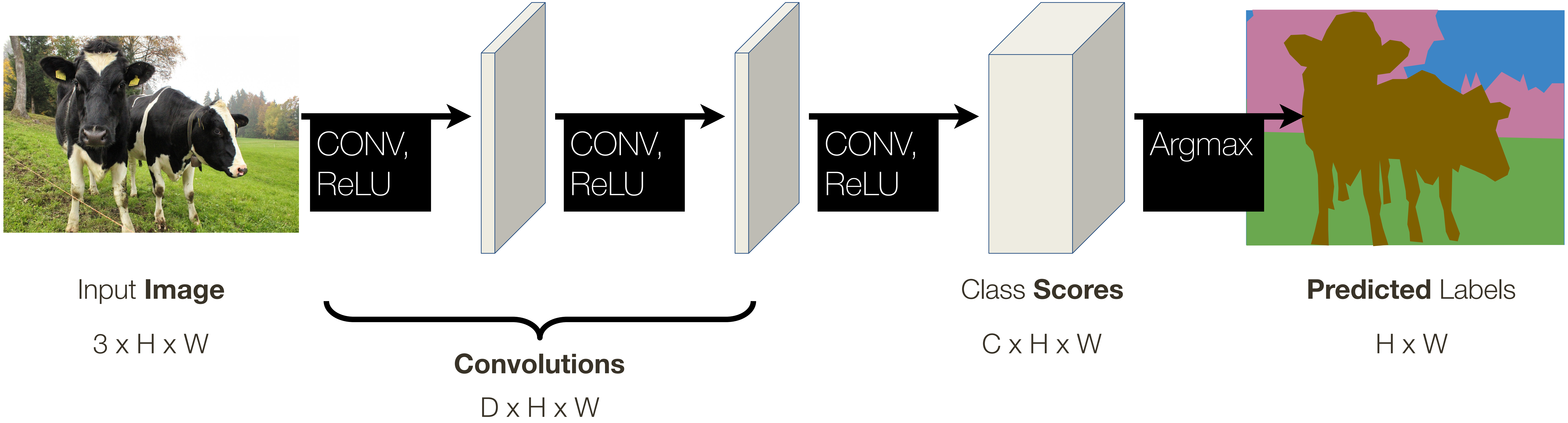
Design a network as a number of convolutional layers to make predictions for all pixels at once!



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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



Problem: Convolutions at the original image scale will be very expensive

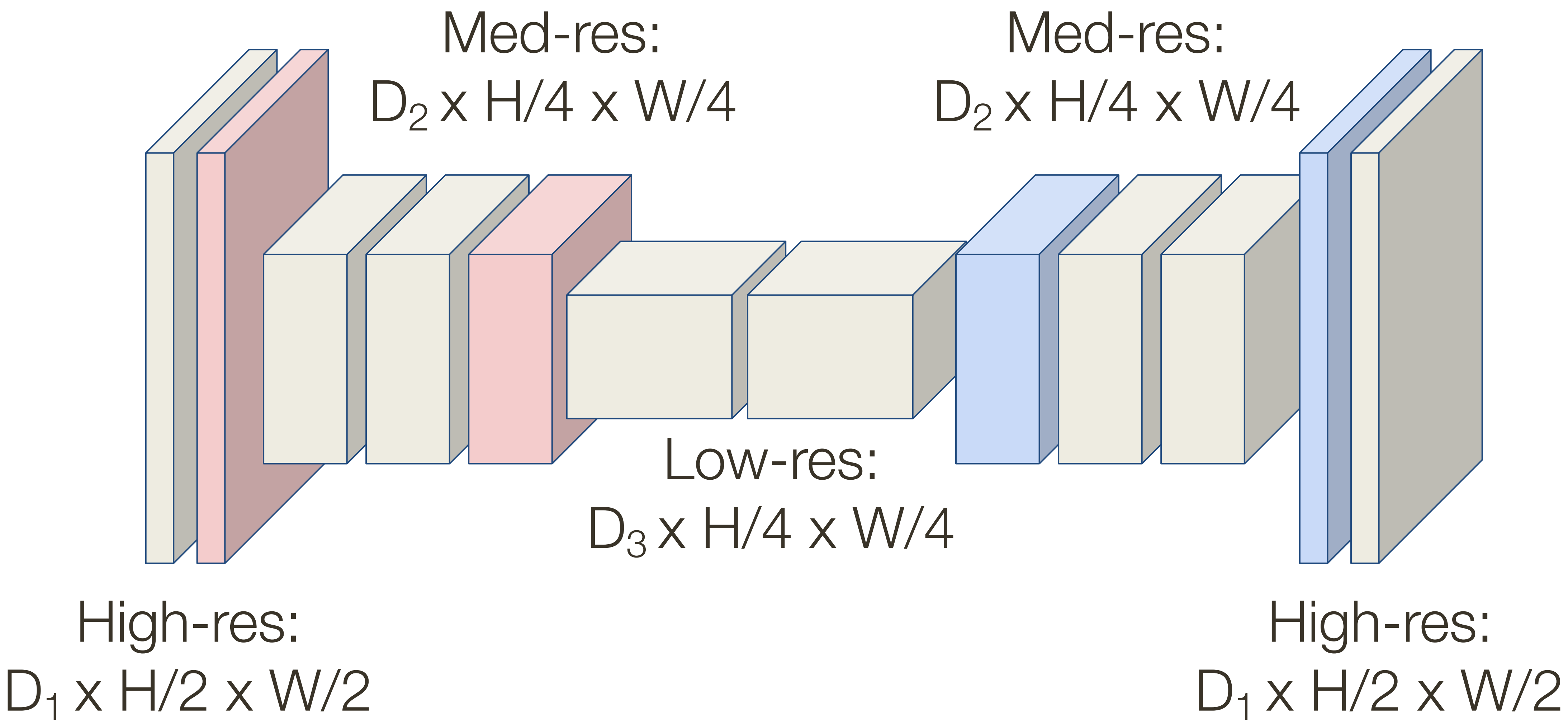
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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

Input **Image**

3 x H x W



Predicted Labels

H x W

[Long et al, CVPR 2015]
[Noh et al, ICCV 2015]

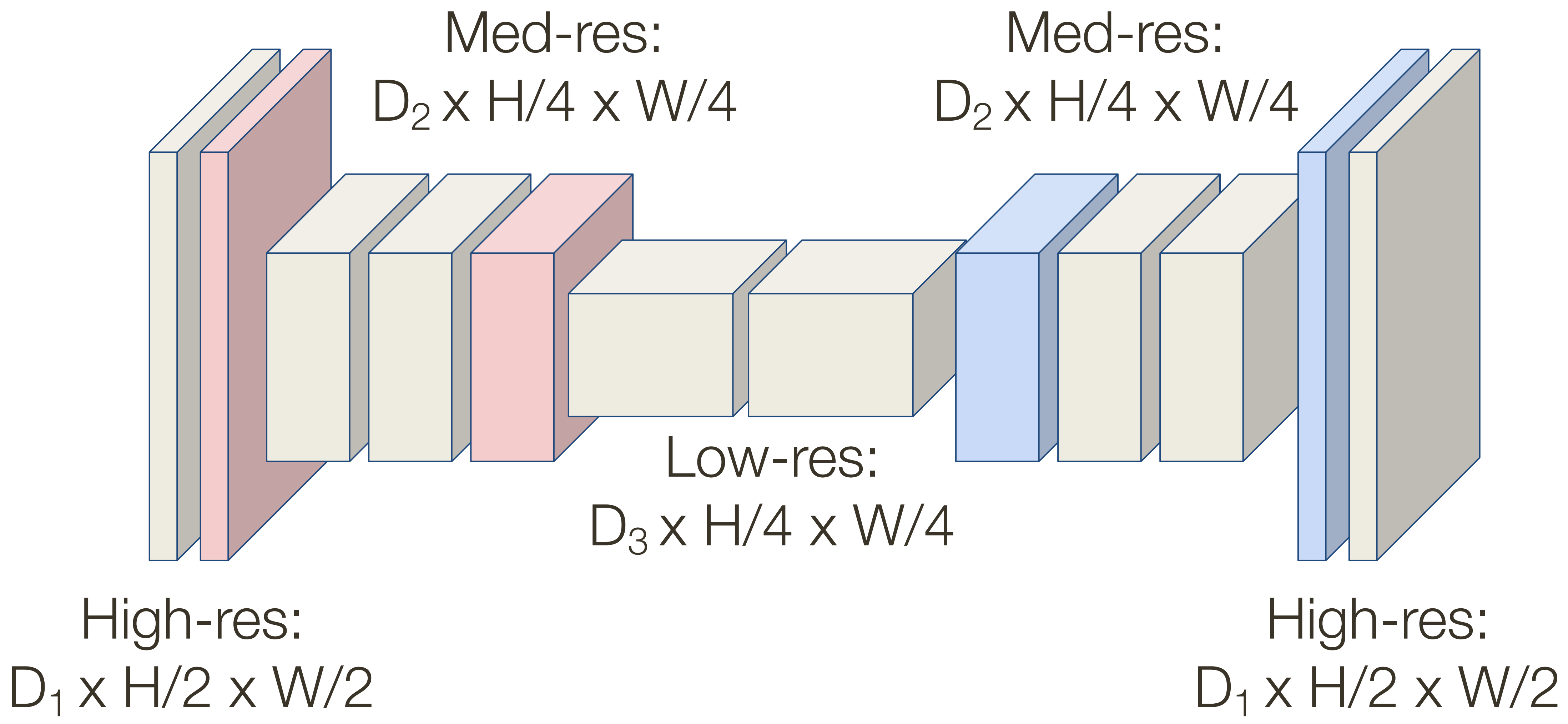
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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

Input **Image**

3 x H x W



Predicted Labels

H x W

Downsampling = Pooling

Upsampling = ???

[Long et al, CVPR 2015]
[Noh et al, ICCV 2015]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling** (a.k.a "Unpooling")

Nearest Neighbor

1	2
3	4

Input: 2 x 2

→

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

Output: 4 x 4

In-network **Up Sampling** (a.k.a "Unpooling")

Nearest Neighbor

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \left[\begin{array}{cc|cc} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ \hline 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{array}\right]$$

Input: 2 x 2 **Output:** 4 x 4

"Bed of Nails"

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \left[\begin{array}{cc|cc} 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 3 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

Input: 2 x 2 **Output:** 4 x 4

In-network **Up Sampling:** Max Unpooling

Max Pooling
Remember which element was max!

$$\begin{array}{cc|cc} 1 & 2 & 6 & 3 \\ 3 & 5 & 2 & 1 \\ \hline 1 & 2 & 2 & 1 \\ 7 & 3 & 4 & 8 \end{array} \longrightarrow \begin{array}{cc} 5 & 6 \\ 7 & 8 \end{array} \longrightarrow \cdots \longrightarrow \begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}$$

Rest of the network

Input: 4 x 4

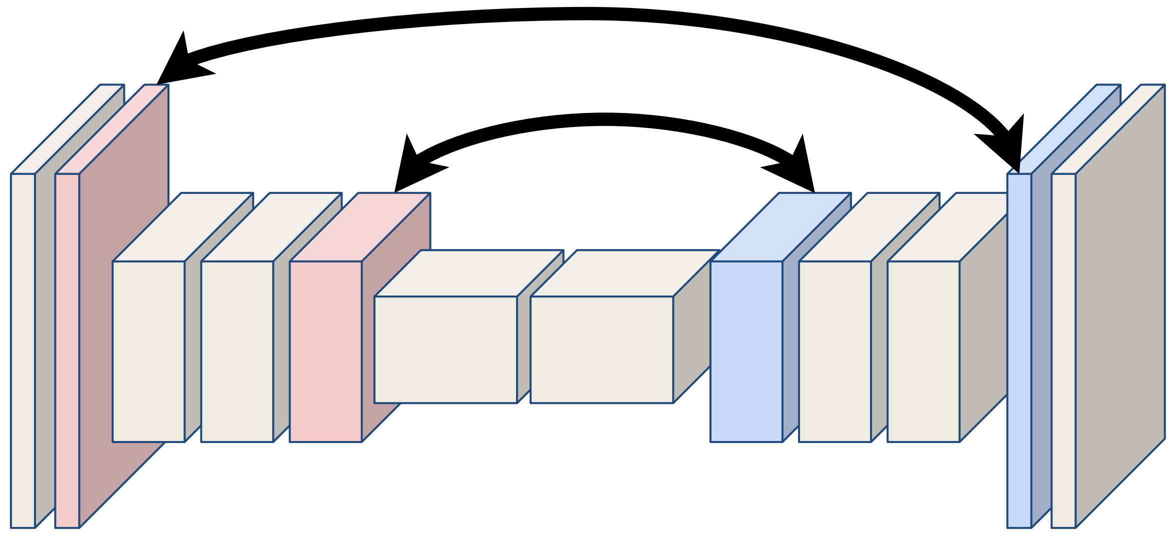
Output: 2 x 2

Max Unpooling
Use positions from pooling layer

Input: 2 x 2

$$\begin{array}{cc|cc} 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 4 \end{array}$$

Output: 4 x 4



Corresponding pairs of downsampling and upsampling layers

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Computer **Vision Problems** (no language for now)

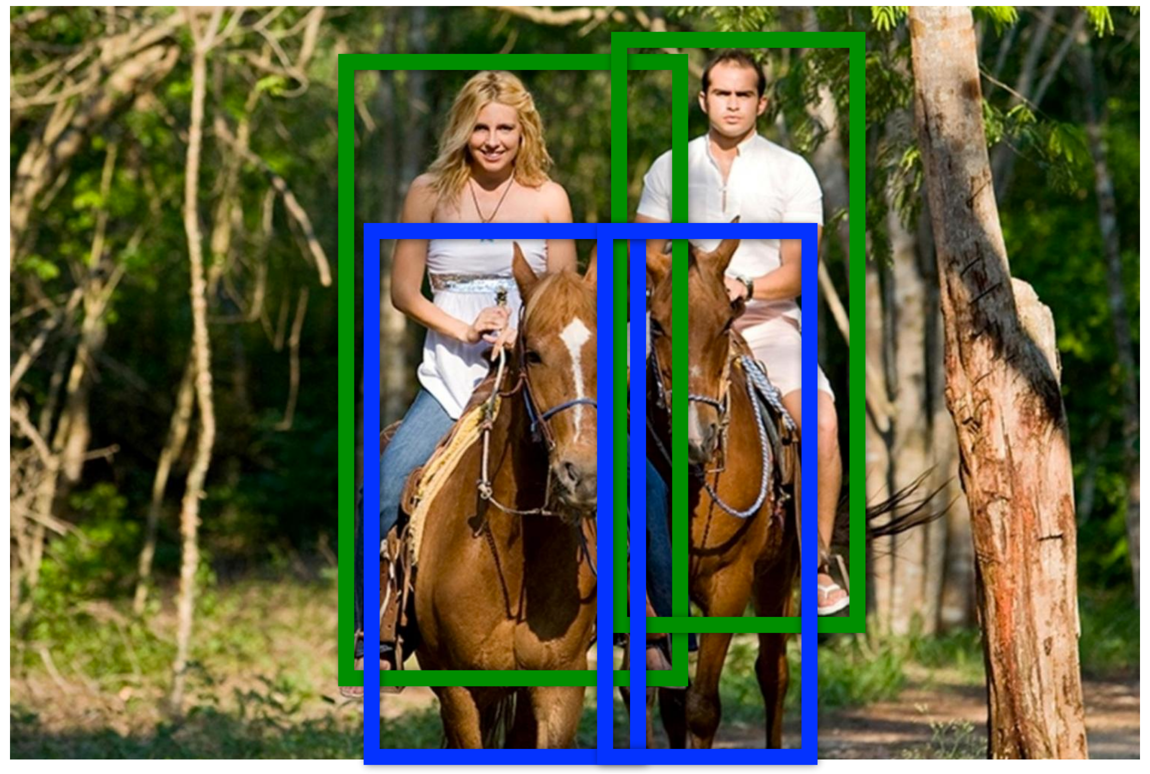
Categorization

Multi-**class:** Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

Segmentation

Horse
Person

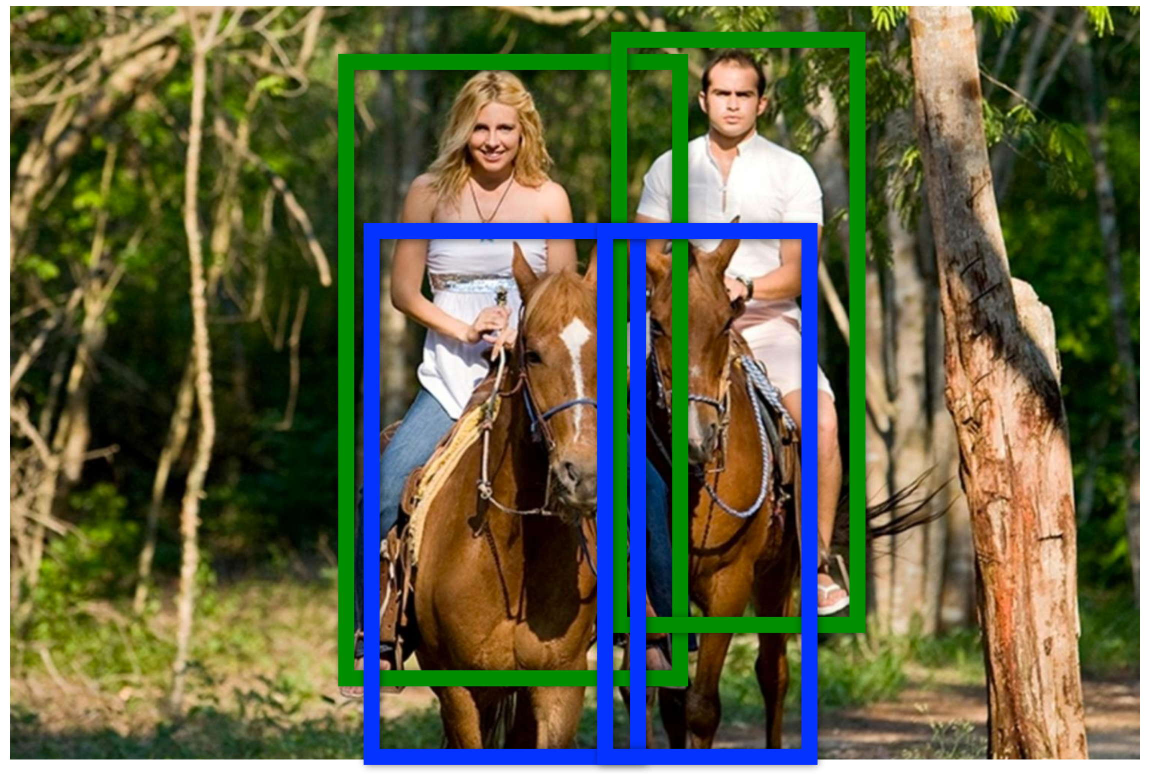
COCO
Common Objects in Context

Instance Segmentation

Horse1
Horse2
Person1
Person2

Computer **Vision Problems** (no language for now)

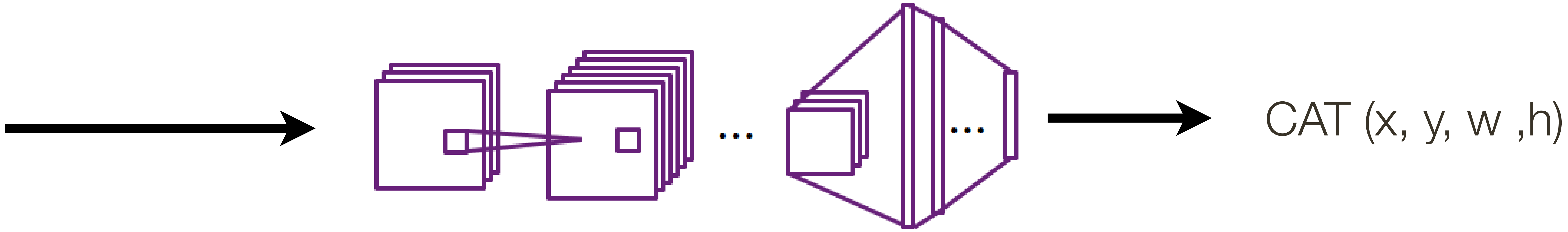
Detection



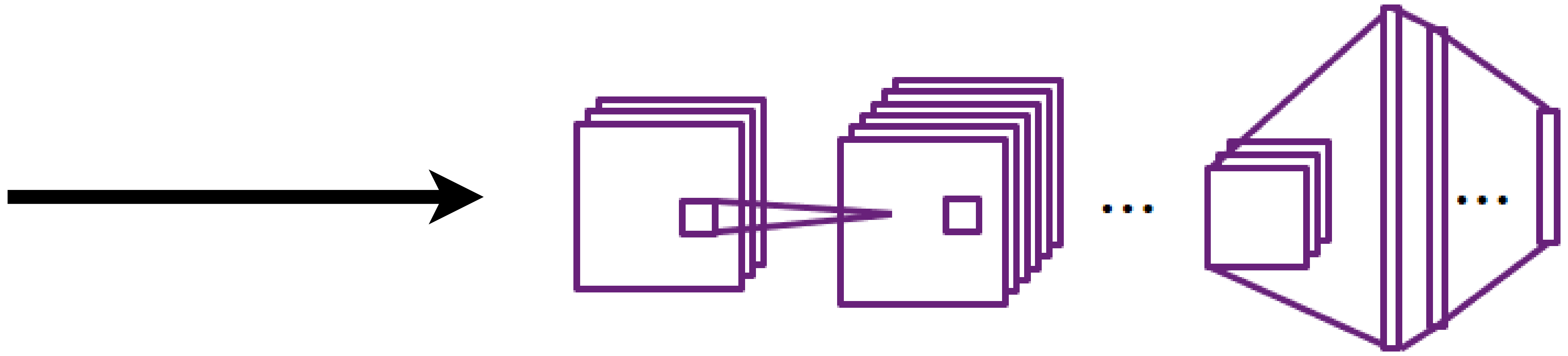
Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

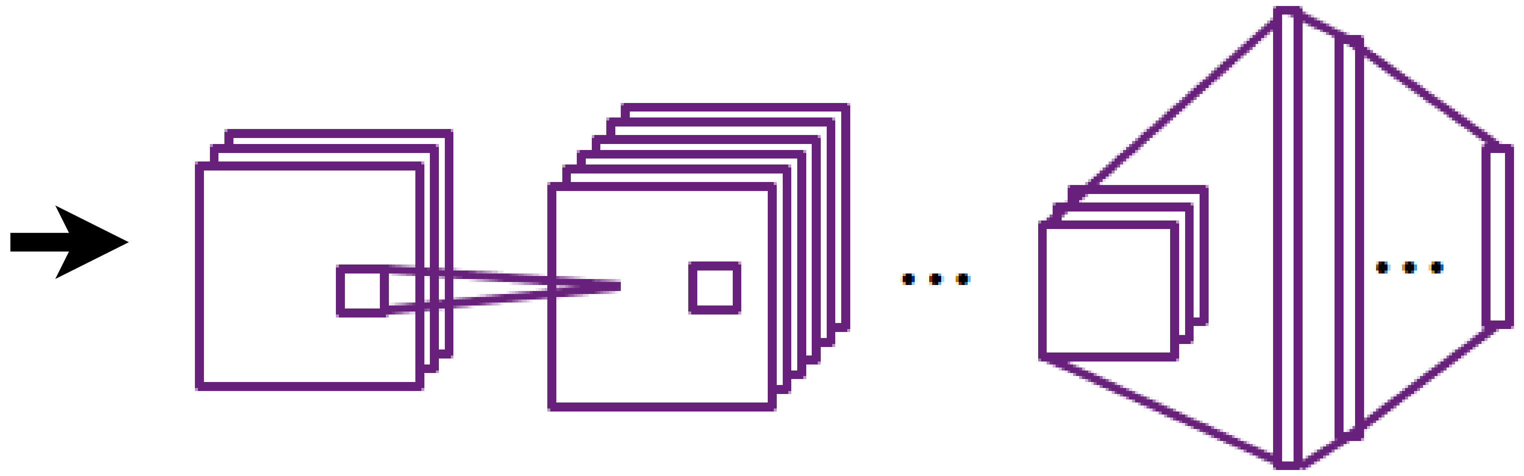
Object **Detection** as Regression Problem



Object **Detection** as Regression Problem



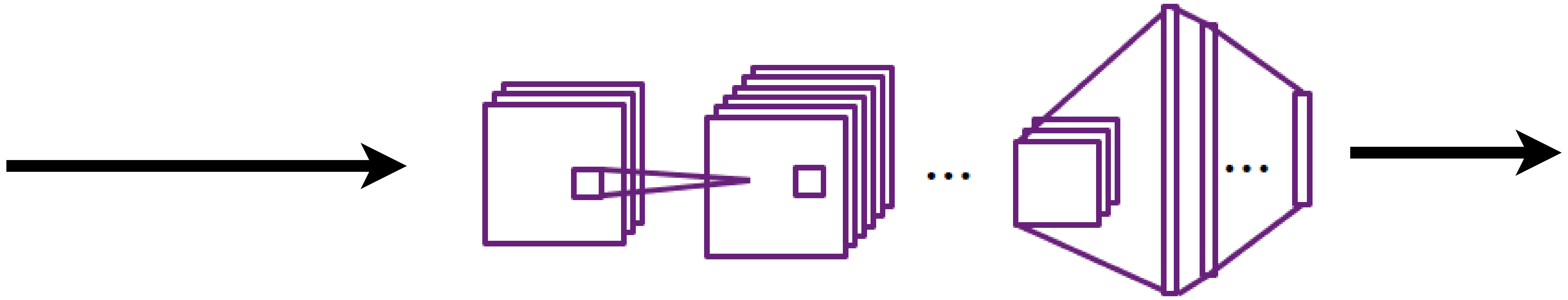
CAT (x, y, w ,h)



DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
…

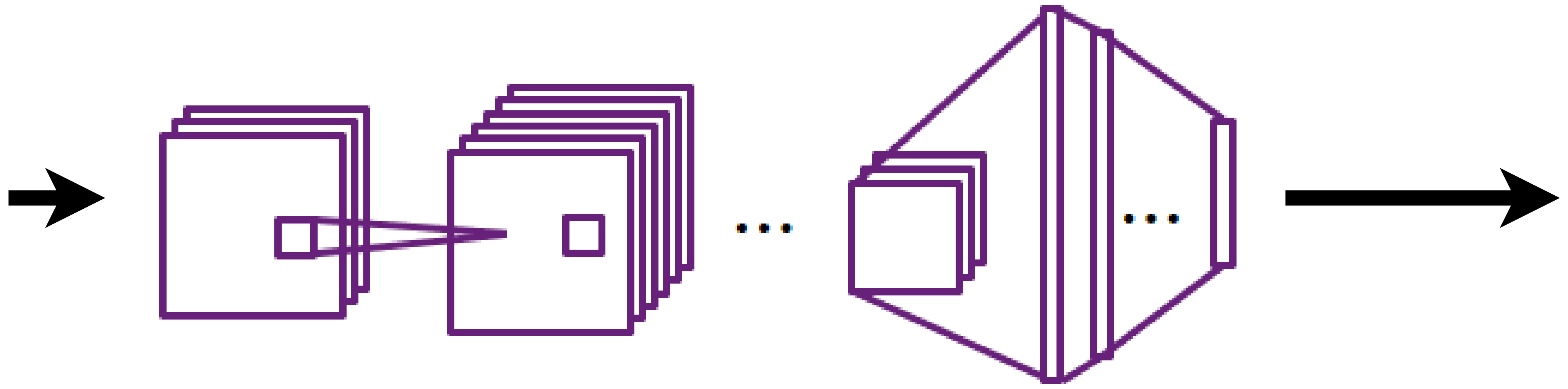
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Regression Problem



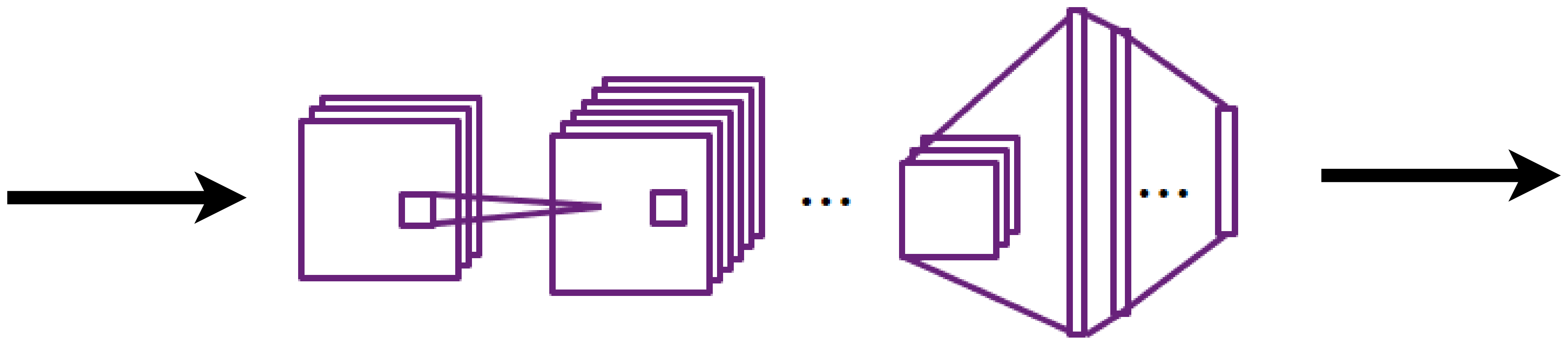
CAT (x, y, w ,h)

Problem: each image needs a different number of outputs



DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
…

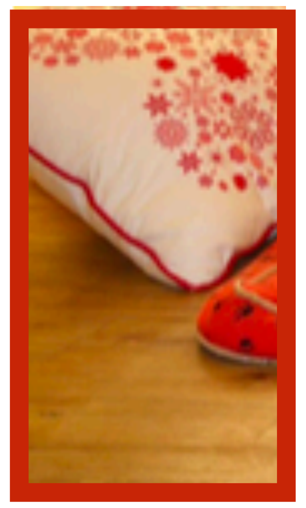
Object **Detection** as Classification Problem

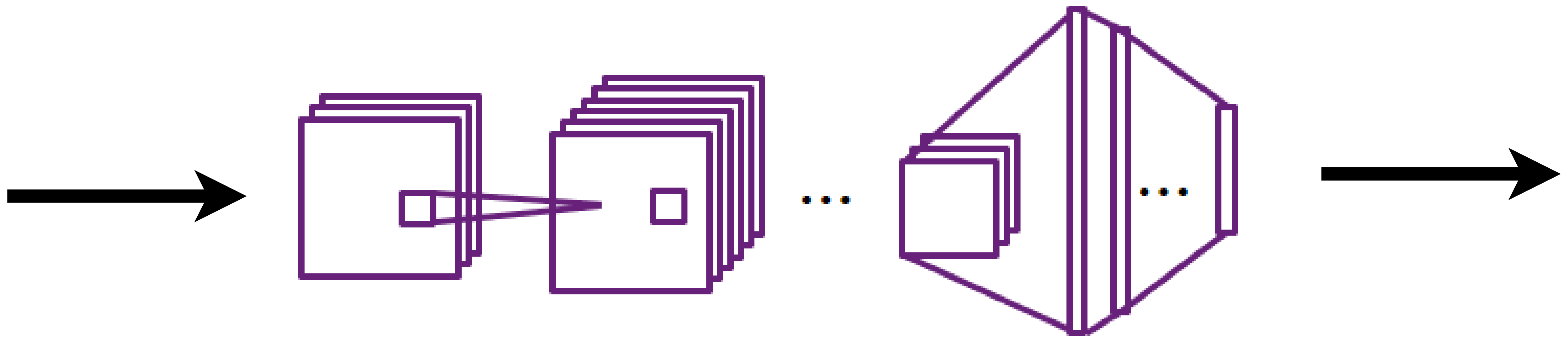


Category	**Prediction**
Dog	No
Cat	No
Couch	No
Flowers	No
Background	**Yes**
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

Object **Detection** as Classification Problem

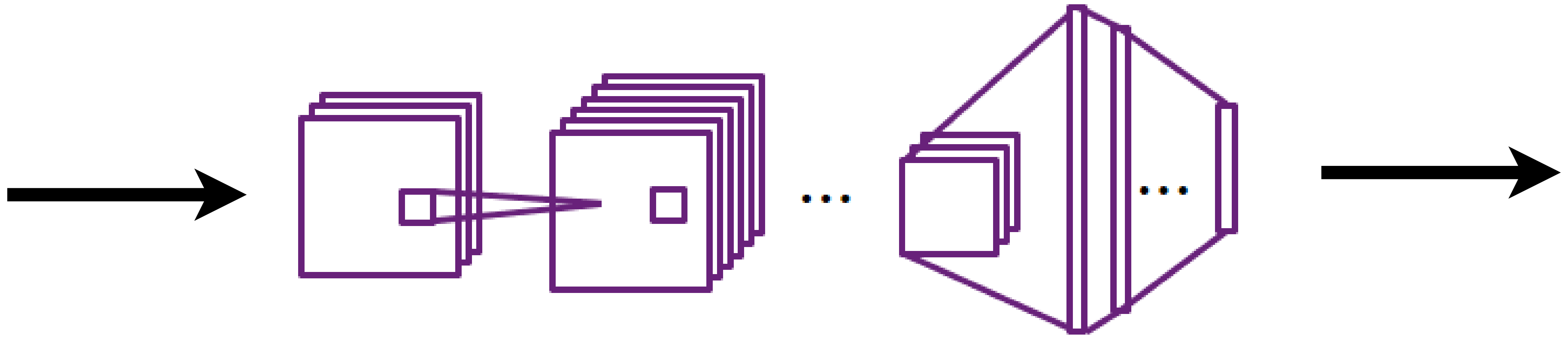




Category	**Prediction**
Dog	No
Cat	No
Couch	No
Flowers	No
Background	**Yes**
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

Object **Detection** as Classification Problem

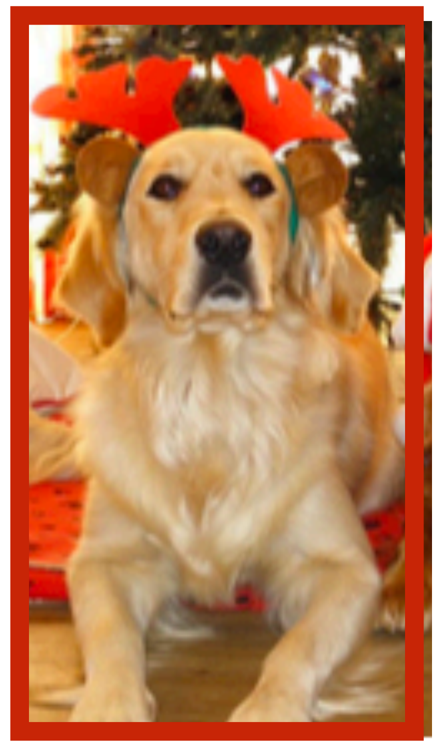


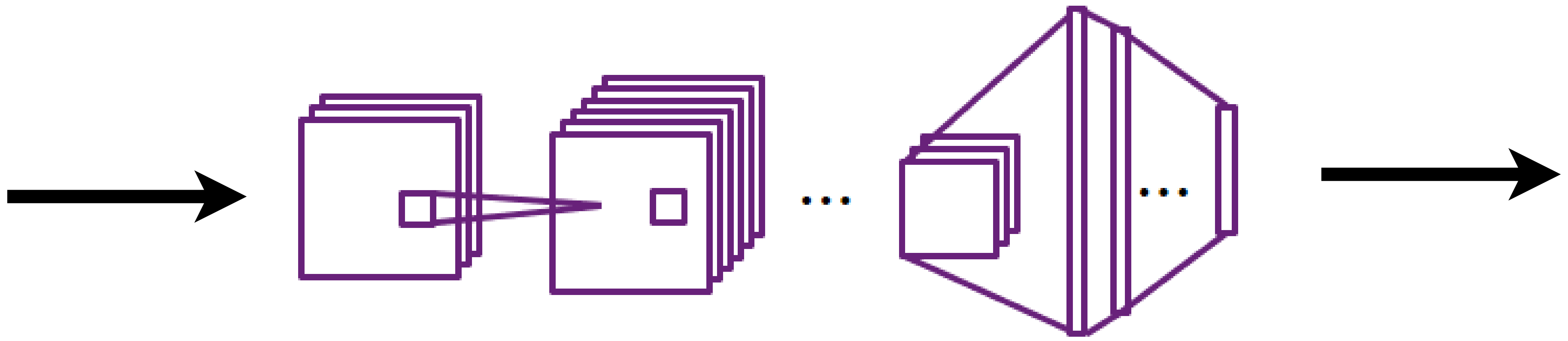
Category	**Prediction**
Dog	**Yes**
Cat	No
Couch	No
Flowers	No
Background	No
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Classification Problem



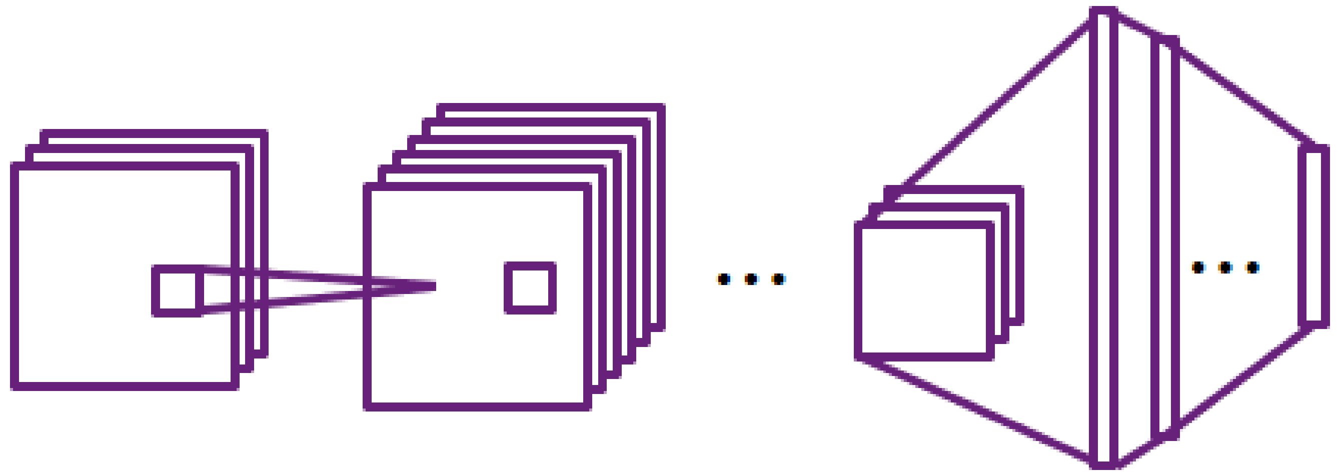


Category	**Prediction**
Dog	**Yes**
Cat	No
Couch	No
Flowers	No
Background	No
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Classification Problem

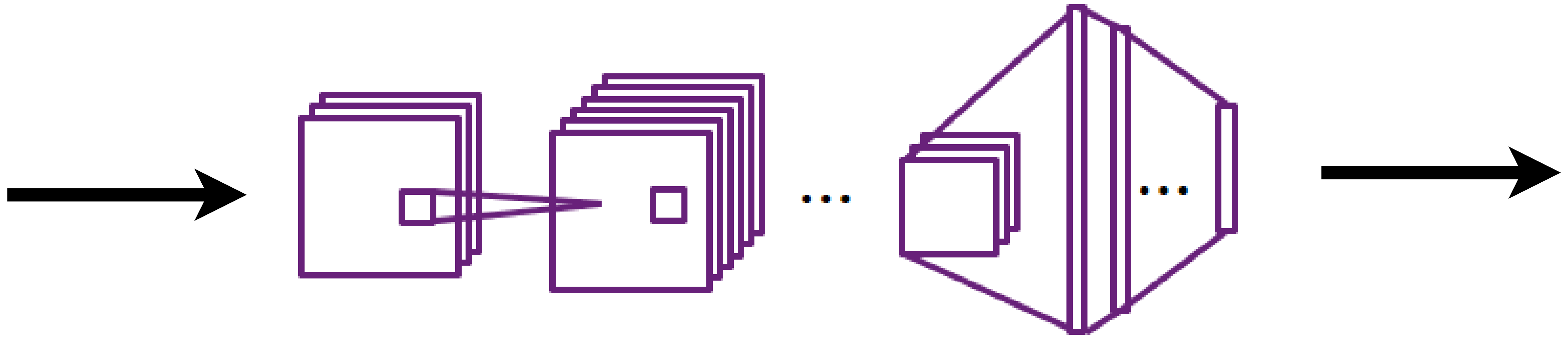


Category	**Prediction**
Dog	No
Cat	**Yes**
Couch	No
Flowers	No
Background	No
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

Object **Detection** as Classification Problem

Problem: Need to apply CNN to **many** patches in each image



Category	**Prediction**
Dog	No
Cat	**Yes**
Couch	No
Flowers	No
Background	No
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

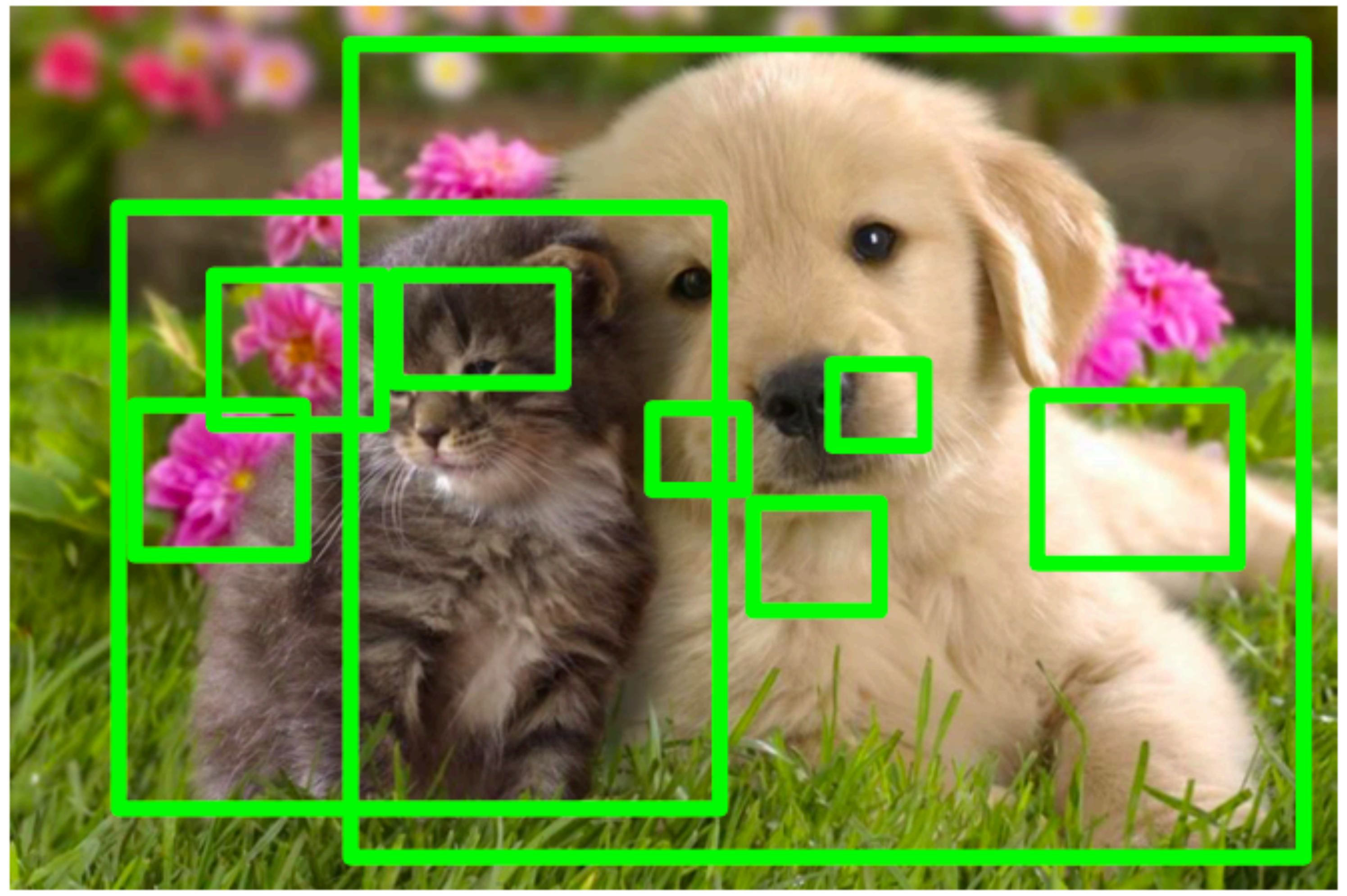
Region Proposals (older idea in vision)

[Alexe et al, TPAMI 2012]
[Uijkings et al, IJCV 2013]
[Cheng et al, CVPR 2014]
[Zitnick and Dollar, ECCV 2014]

Find image **regions that are likely contain objects** (any object at all)

- typically works by looking at histogram distributions, region aspect ratio, closed contours, coherent color

Relatively **fast to run** (Selective Search gives 1000 region proposals in a few seconds on a CPU)



Goal: Get "true" object regions to be in as few top K proposals as possible

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

R-CNN

[Girshick et al, CVPR 2014]

Input **Image**

* image from Ross Girshick

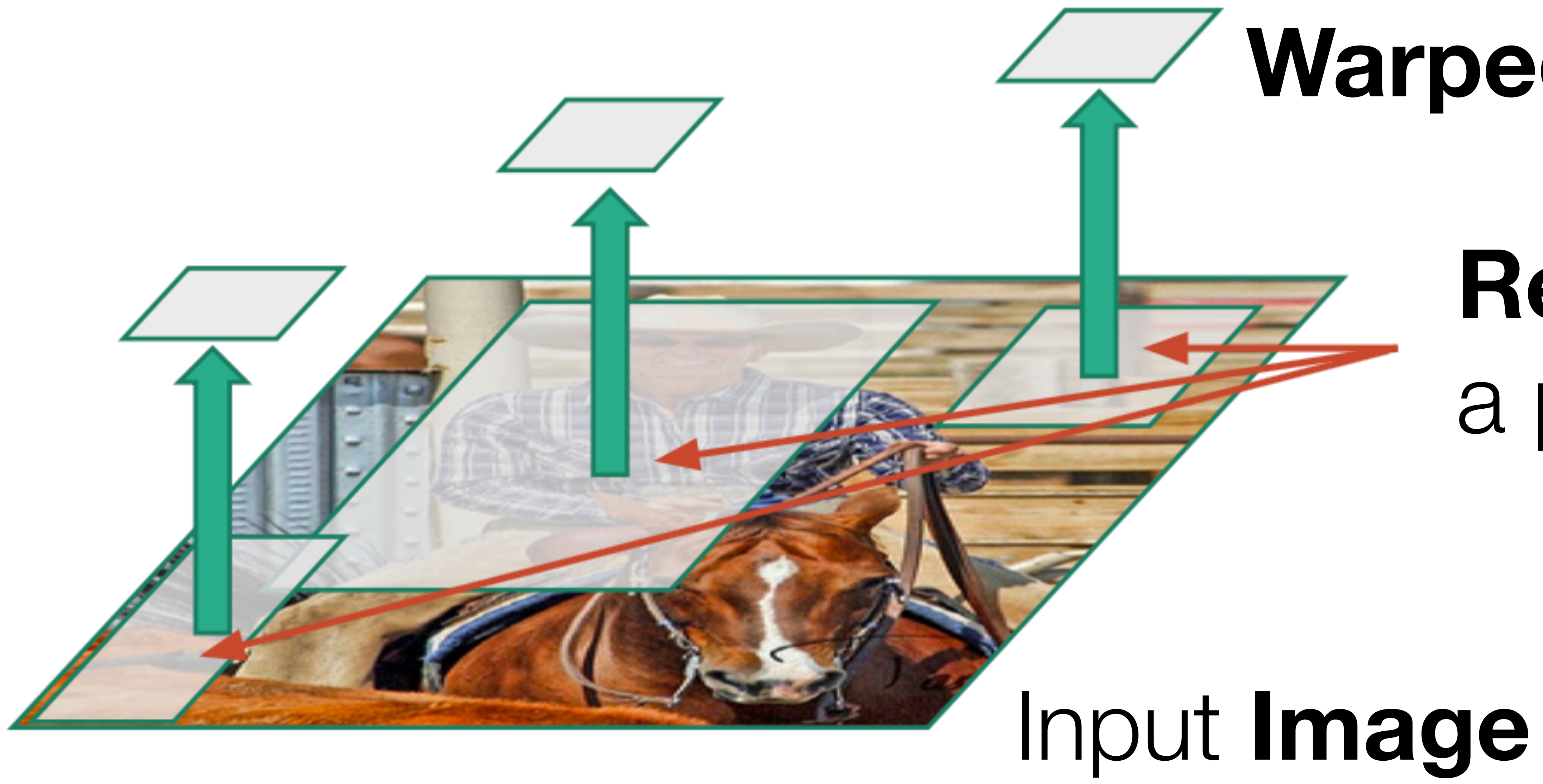
R-CNN

[Girshick et al, CVPR 2014]

* image from Ross Girshick

R-CNN

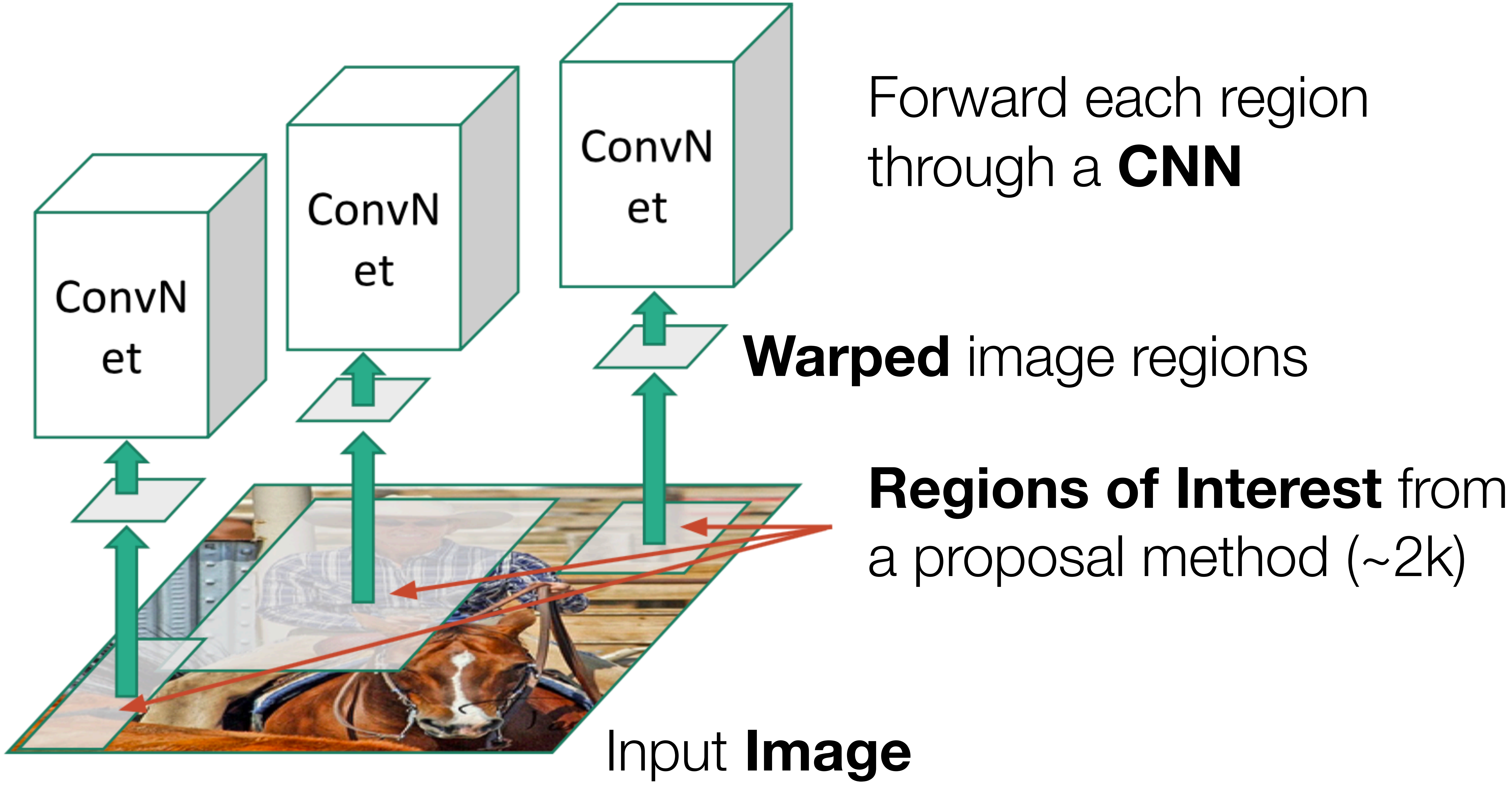
[Girshick et al, CVPR 2014]



* image from Ross Girshick

R-CNN

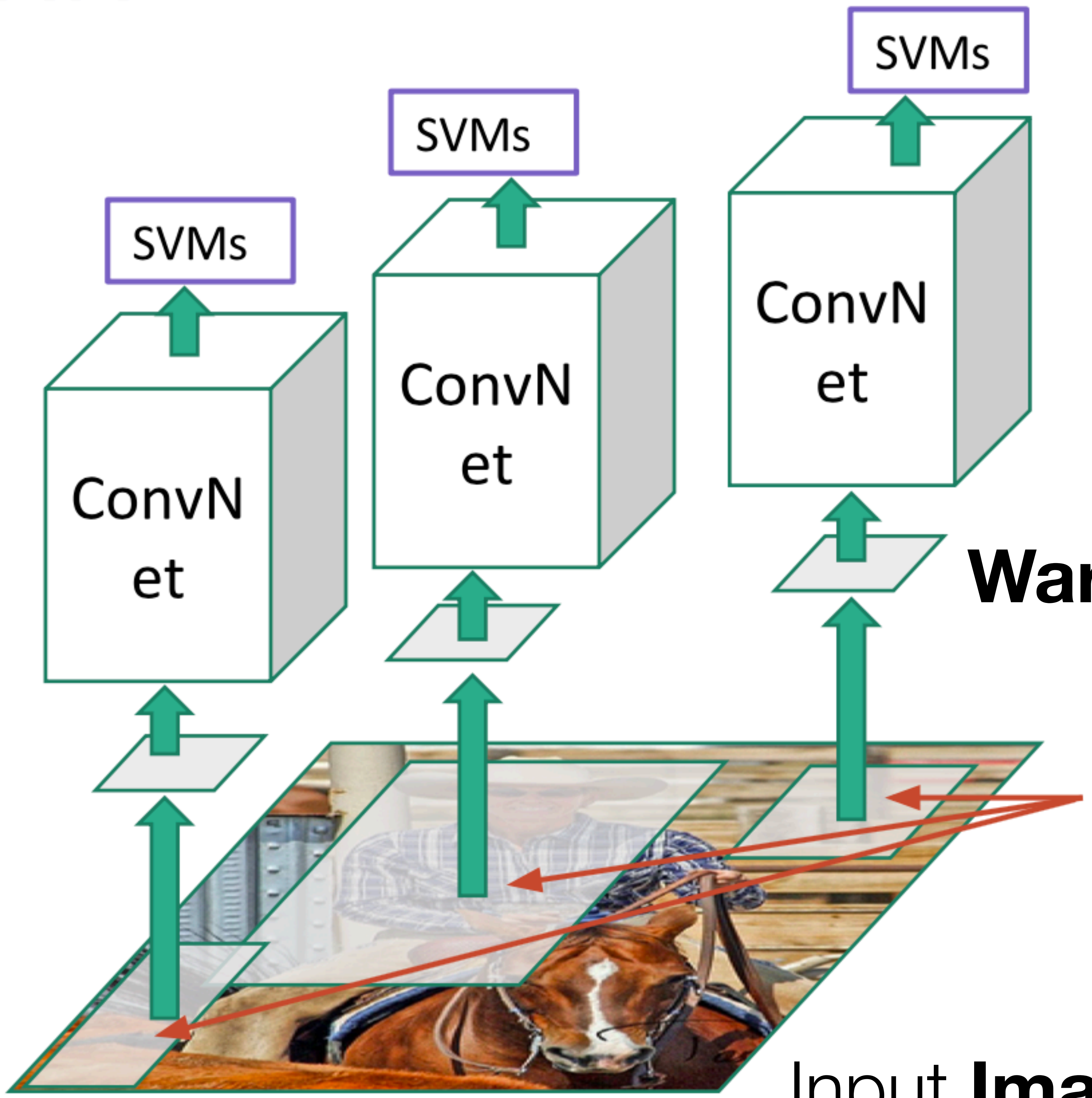
[Girshick et al, CVPR 2014]



* image from Ross Girshick

R-CNN

[Girshick et al, CVPR 2014]



Classify regions with SVM

Forward each region through a **CNN**

Warped image regions

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

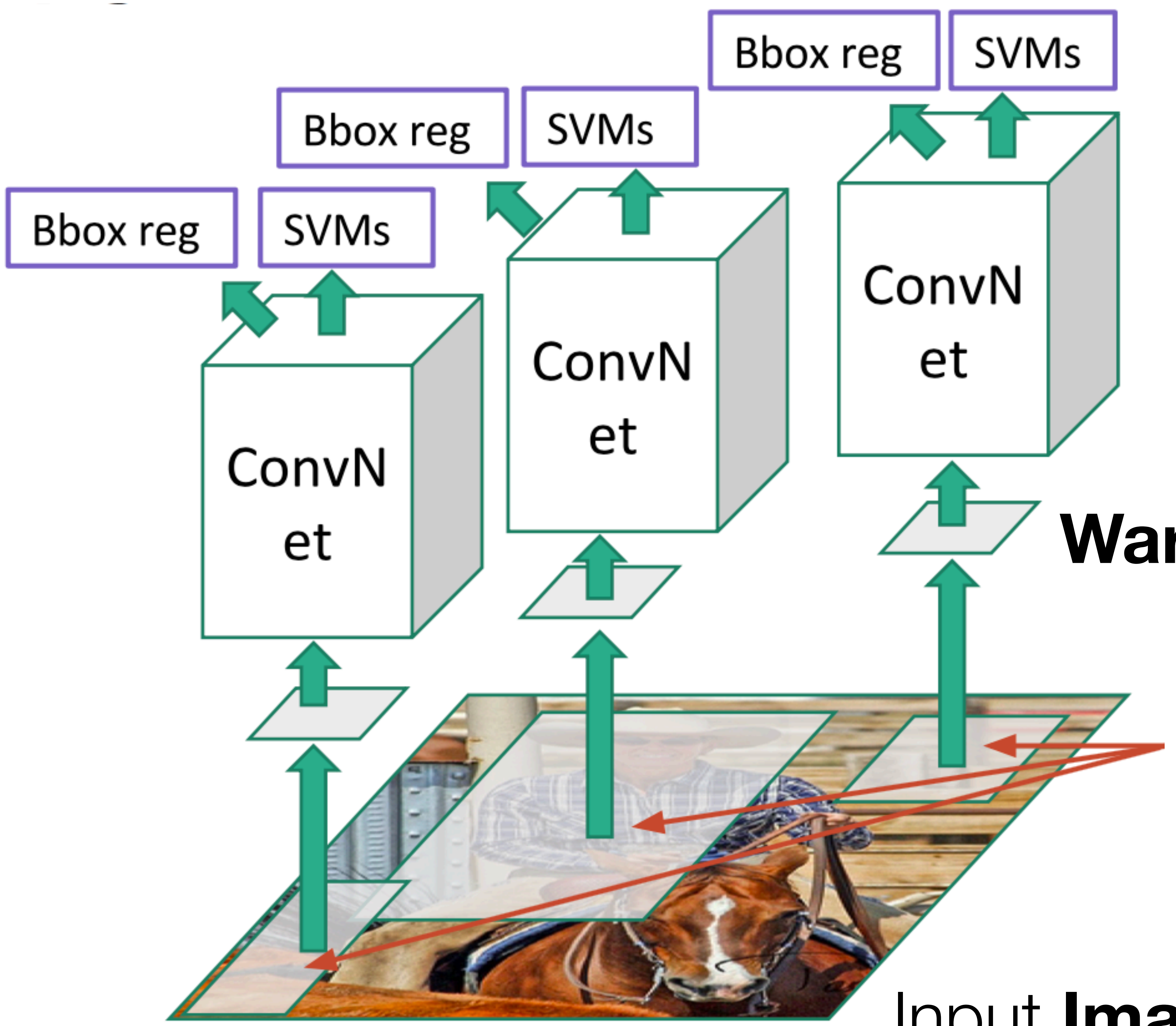
Input **Image**

* image from Ross Girshick

R-CNN

Linear Regression for bounding box offsets

[Girshick et al, CVPR 2014]



Classify regions with SVM

Forward each region through a **CNN**

Warped image regions

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

Input **Image**

* image from Ross Girshick

R-CNN

R-CNN (Regions with CNN features) algorithm:

— Extract promising candidate regions using an object proposals algorithm

— Resize each proposal window to the size of the input layer of a trained convolutional neural network

— Input each resized image patch to the convolutional neural network

Implementation detail: Instead of using the classification scores of the network directly, the output of the final fully-connected layer can be used as an input feature to a trained support vector machine (SVM)

Fast **R-CNN**

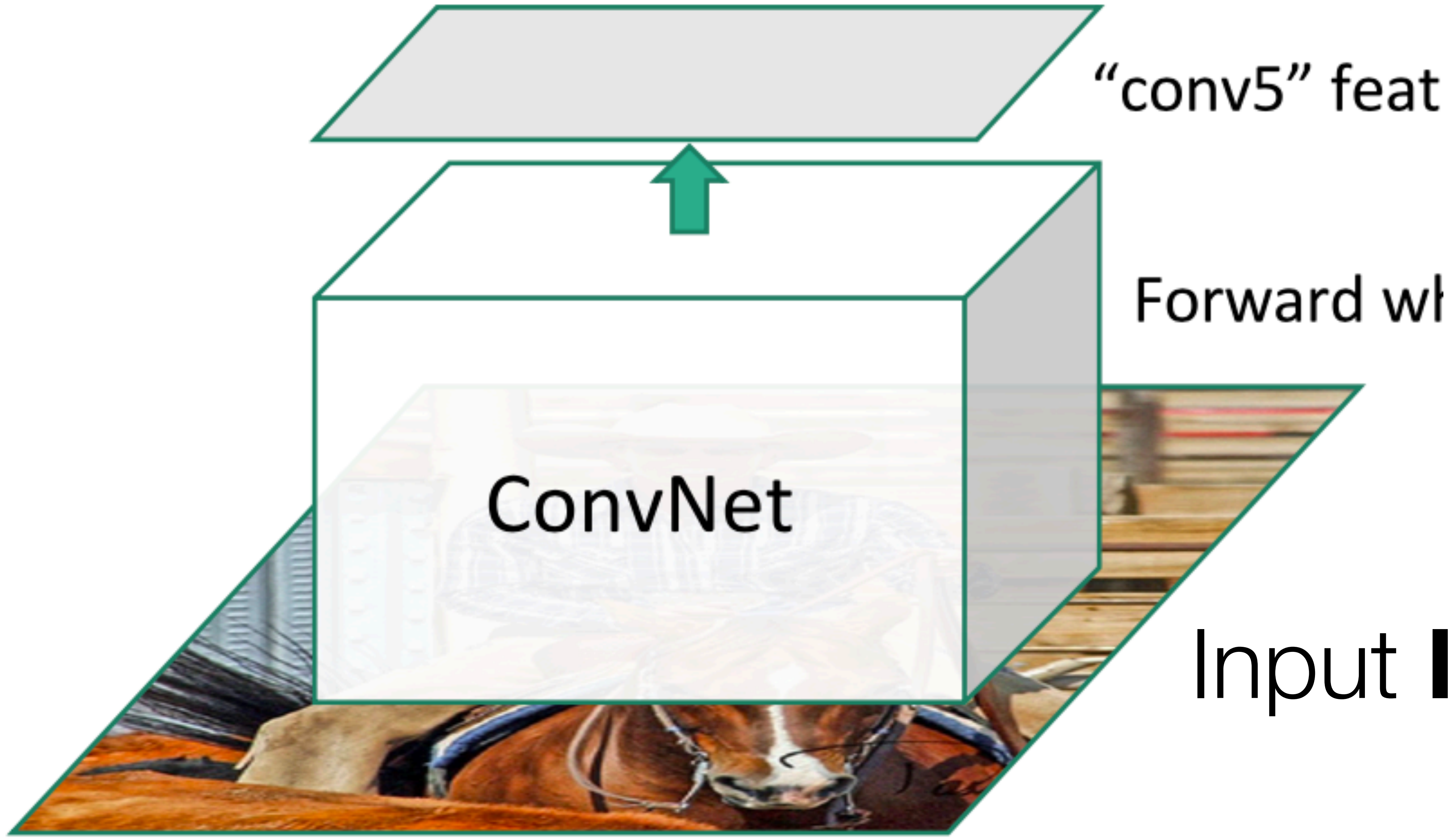
[Girshick et al, ICCV 2015]

Input **Image**

* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]

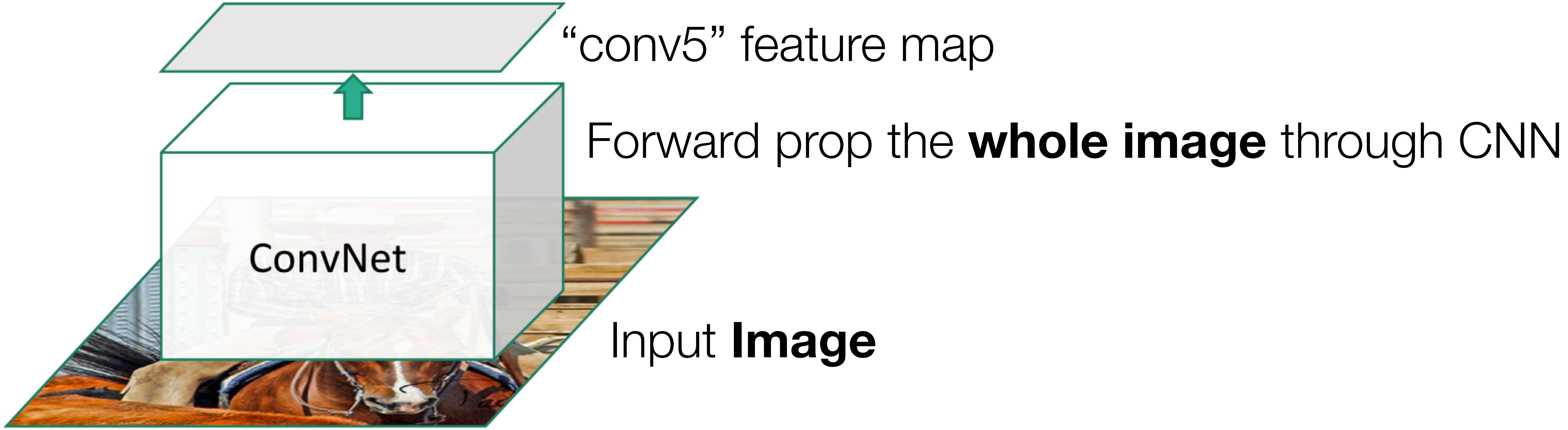


Input **Image**

* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]

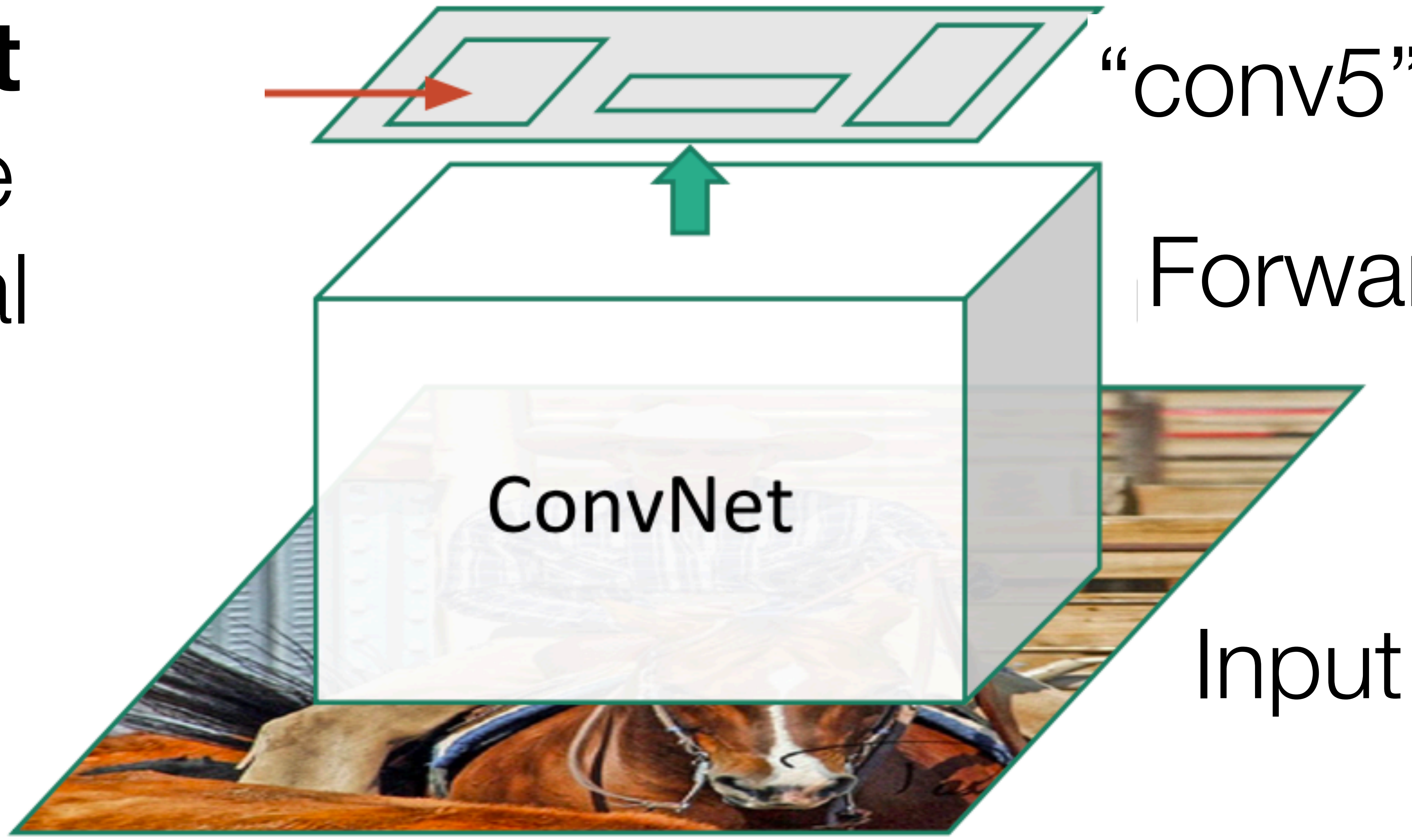


* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]

Regions of Interest from the proposal method



“conv5” feature map

Forward prop the **whole image** through CNN

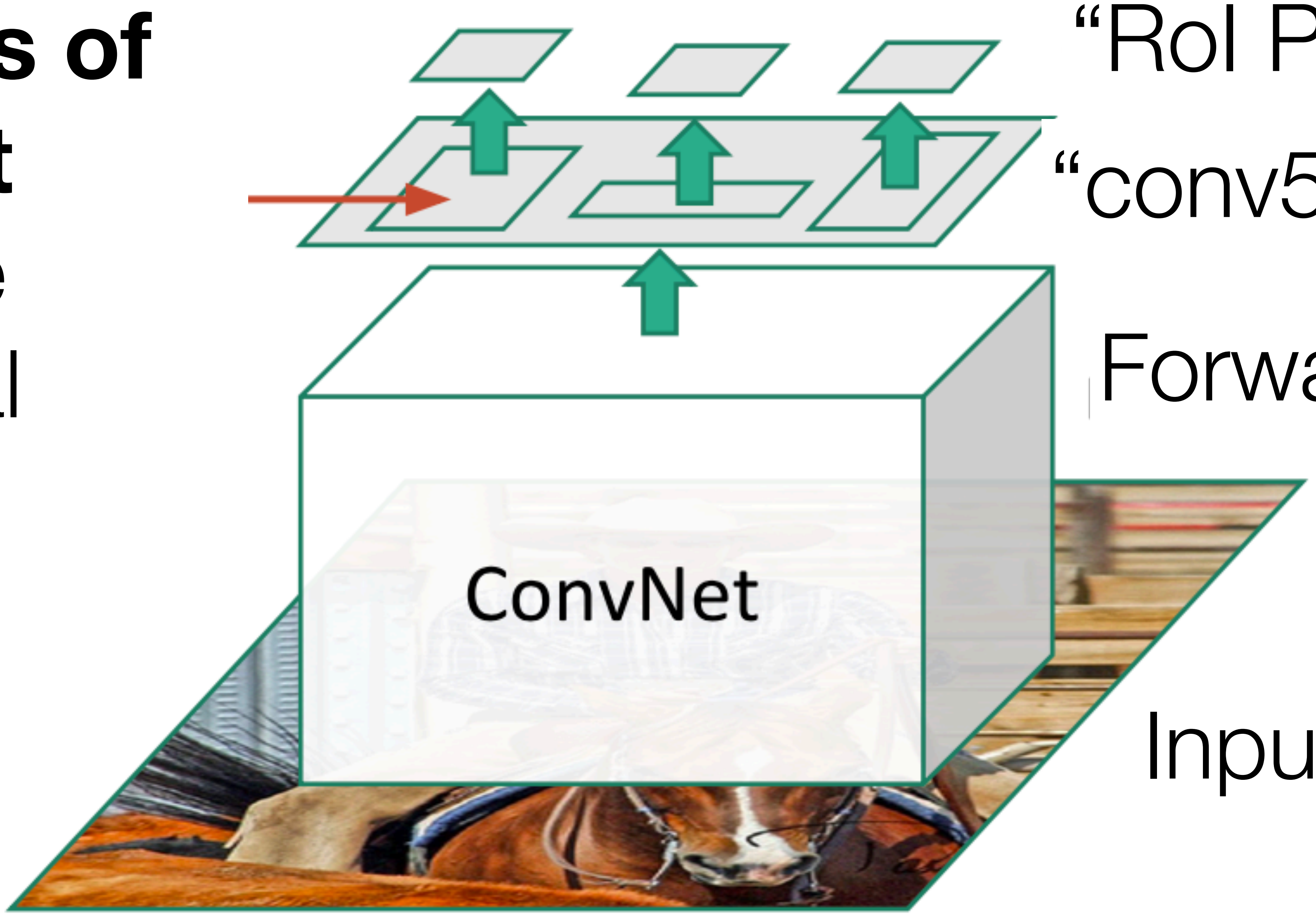
Input **Image**

* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]

Regions of Interest from the proposal method



“RoI Pooling” layer

“conv5” feature map

Forward prop the **whole image** through CNN

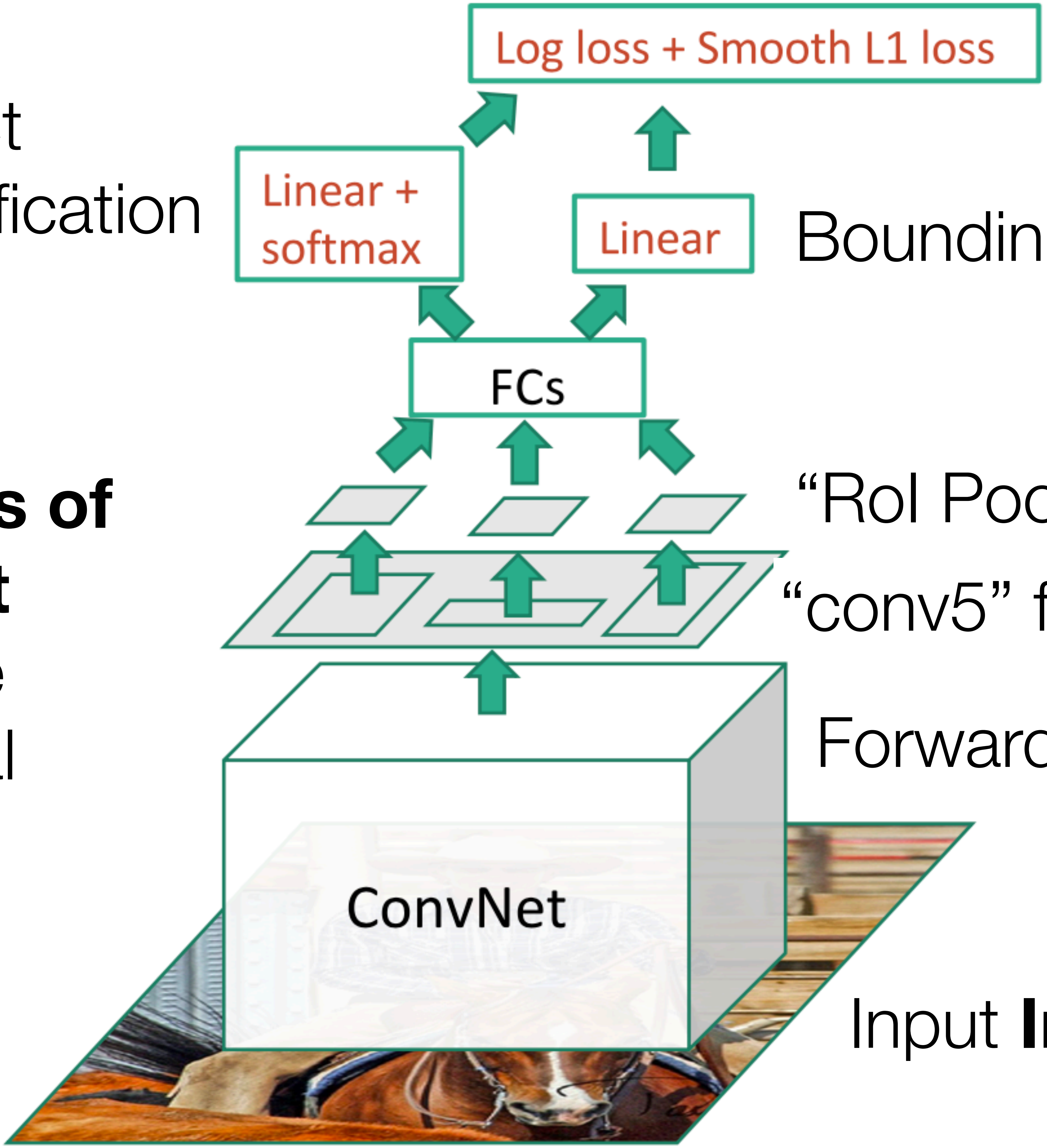
Input **Image**

Girshick, “Fast R-C
Figure copyright R

* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]



Multi-task loss

Object classification

Bounding box regression

“RoI Pooling” layer

Regions of Interest from the proposal method

“conv5” feature map

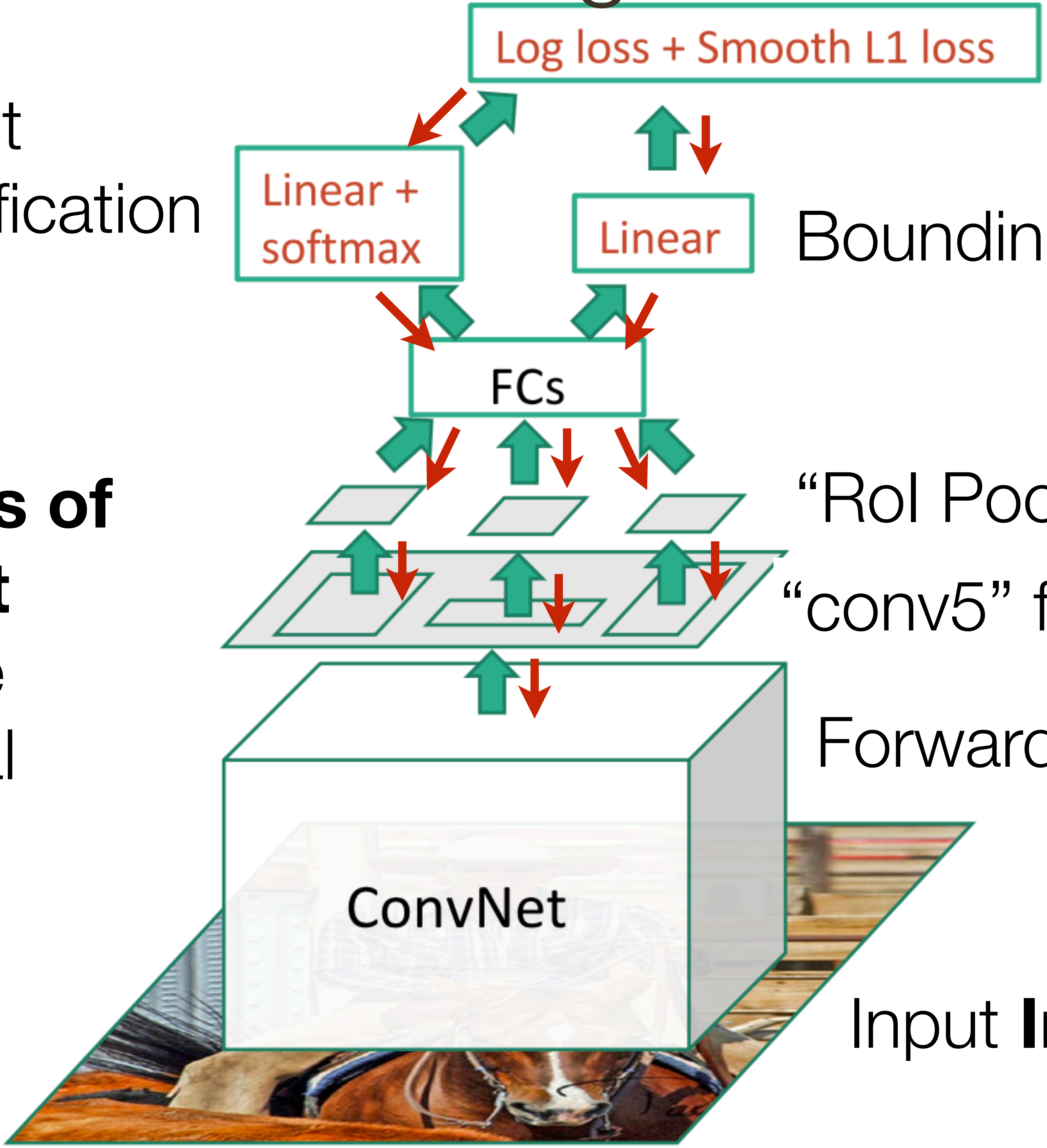
Forward prop the **whole image** through CNN

Input **Image**

* image from Ross Girshick

Fast **R-CNN:** Training

[Girshick et al, ICCV 2015]



Multi-task loss

Object classification

Bounding box regression

“RoI Pooling” layer

“conv5” feature map

Regions of Interest from the proposal method

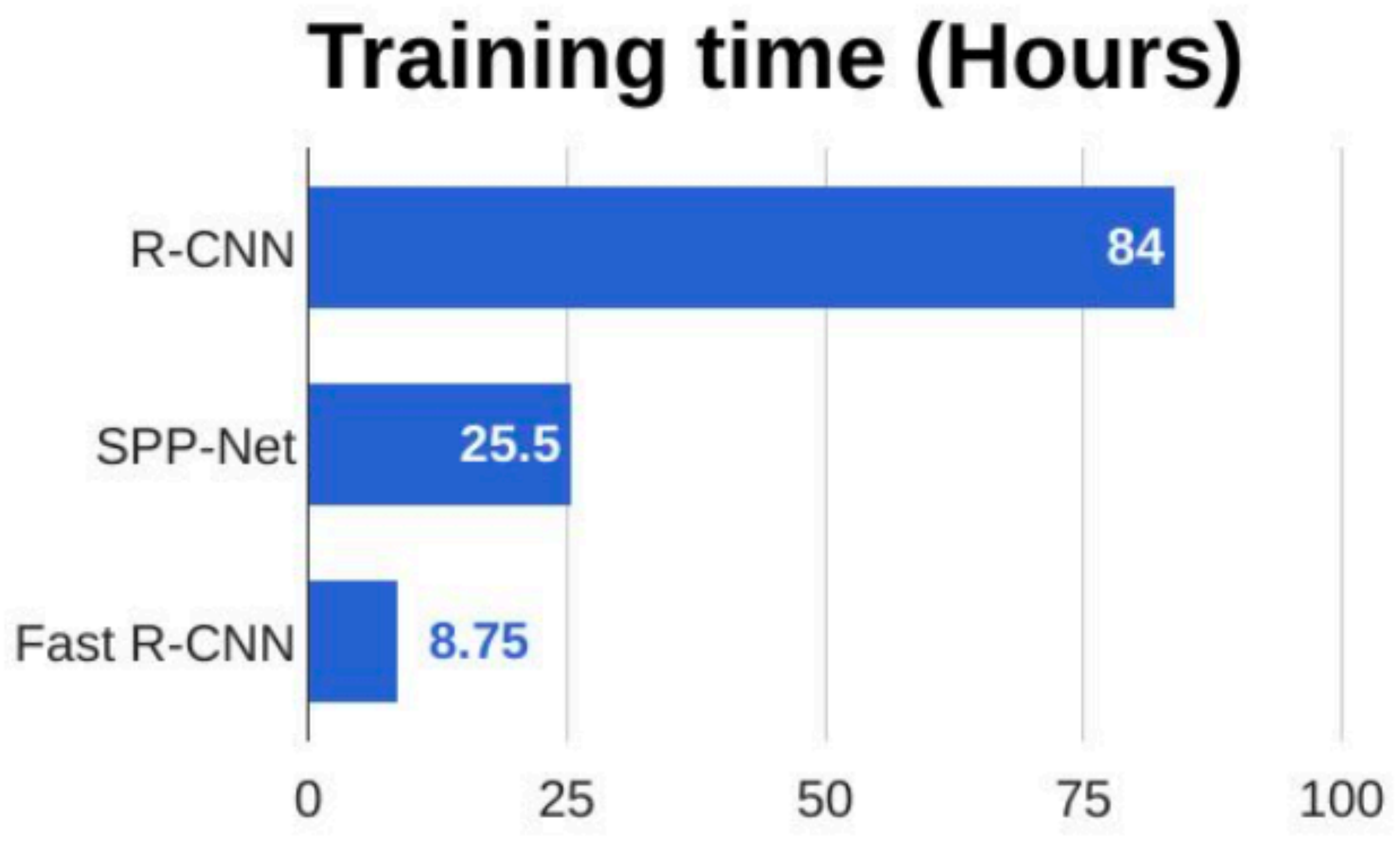
Forward prop the **whole image** through CNN

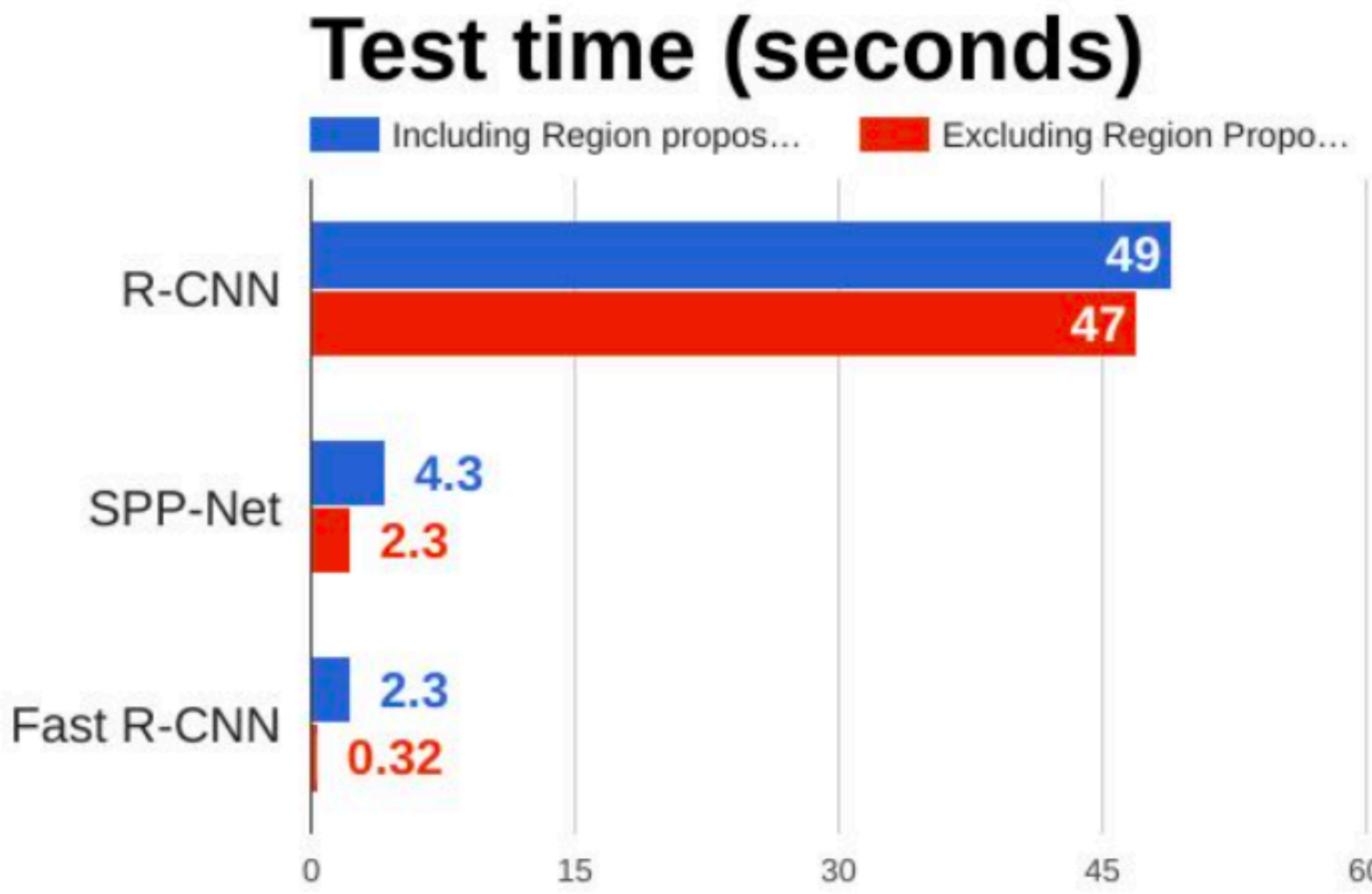
Input **Image**

* image from Ross Girshick

R-CNN vs. SPP vs. Fast R-CNN

[Girshick et al, CVPR 2014]
[Girshick et al, ICCV 2015]
[He et al, ECCV 2014]

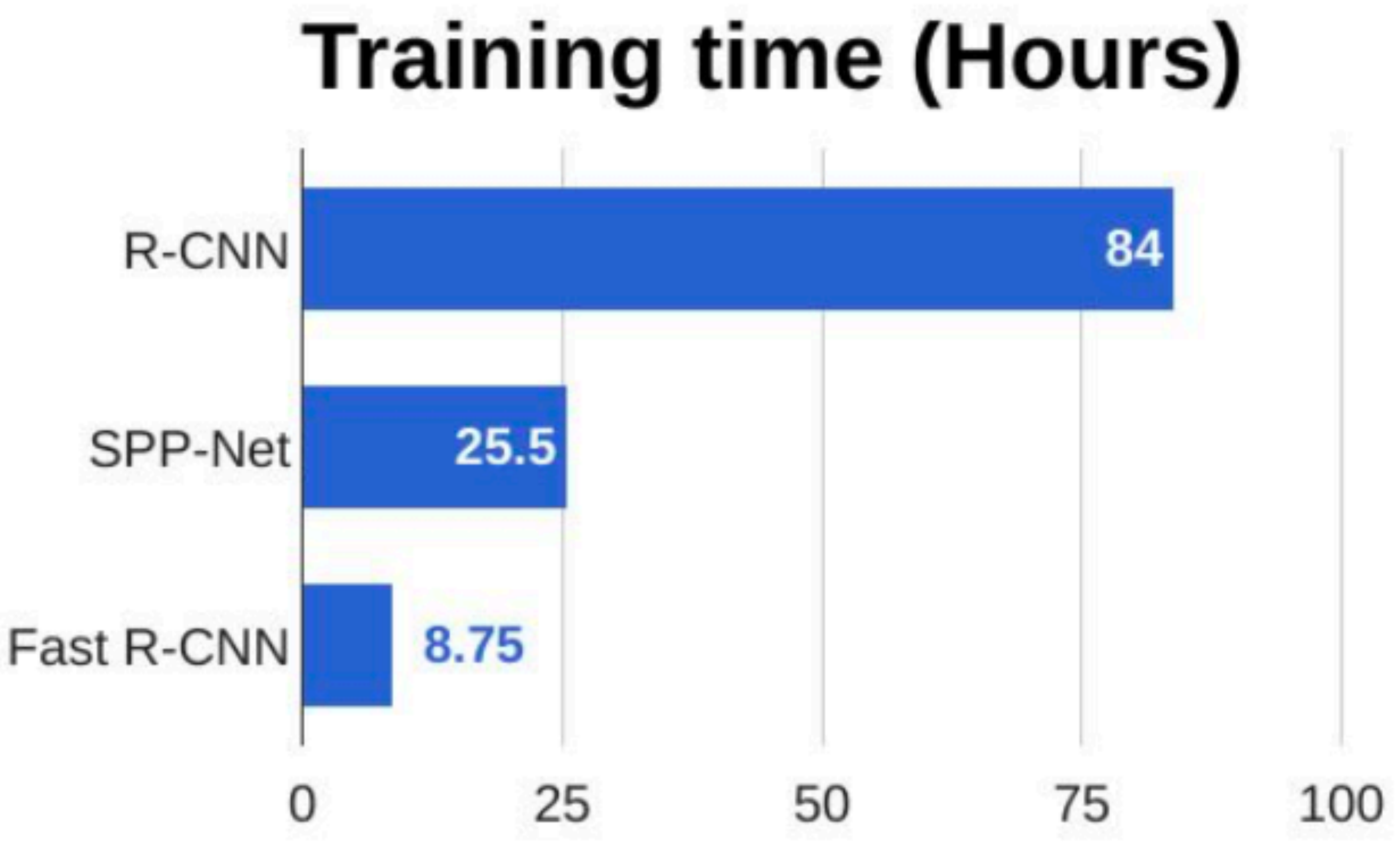


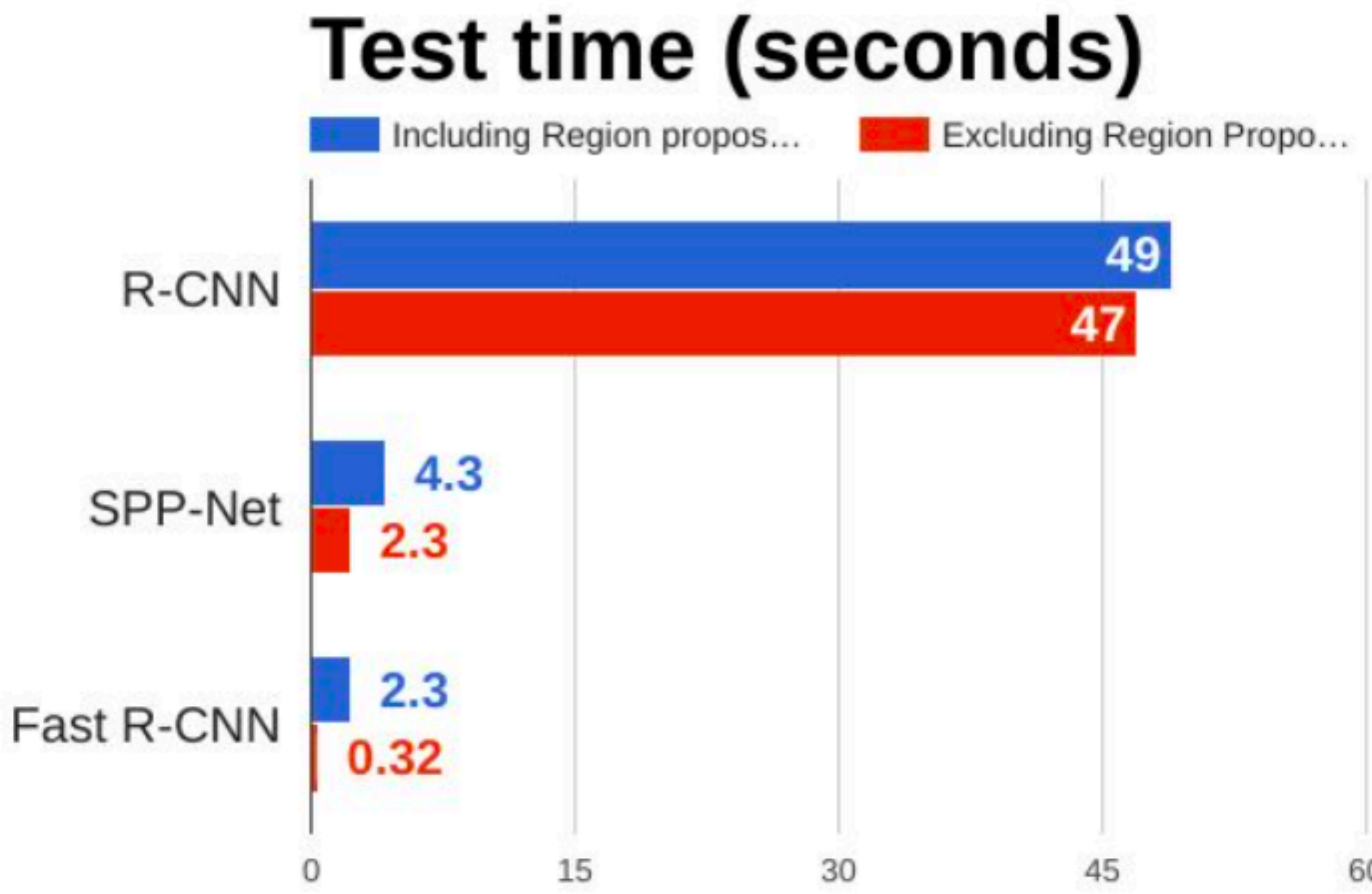


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

R-CNN vs. **SPP** vs. **Fast R-CNN**

[Girshick et al, CVPR 2014]
[Girshick et al, ICCV 2015]
[He et al, ECCV 2014]





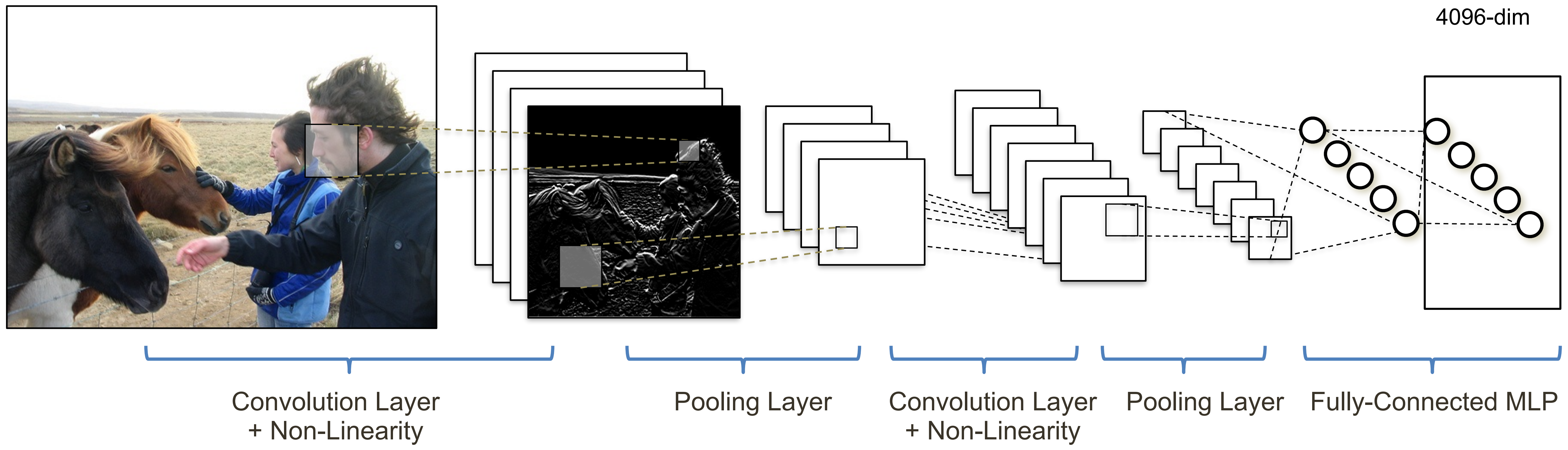
Observation: Performance dominated by the region proposals at this point!

Neural Image **Captioning**

* slide from Dhruv Batra

Neural Image **Captioning**

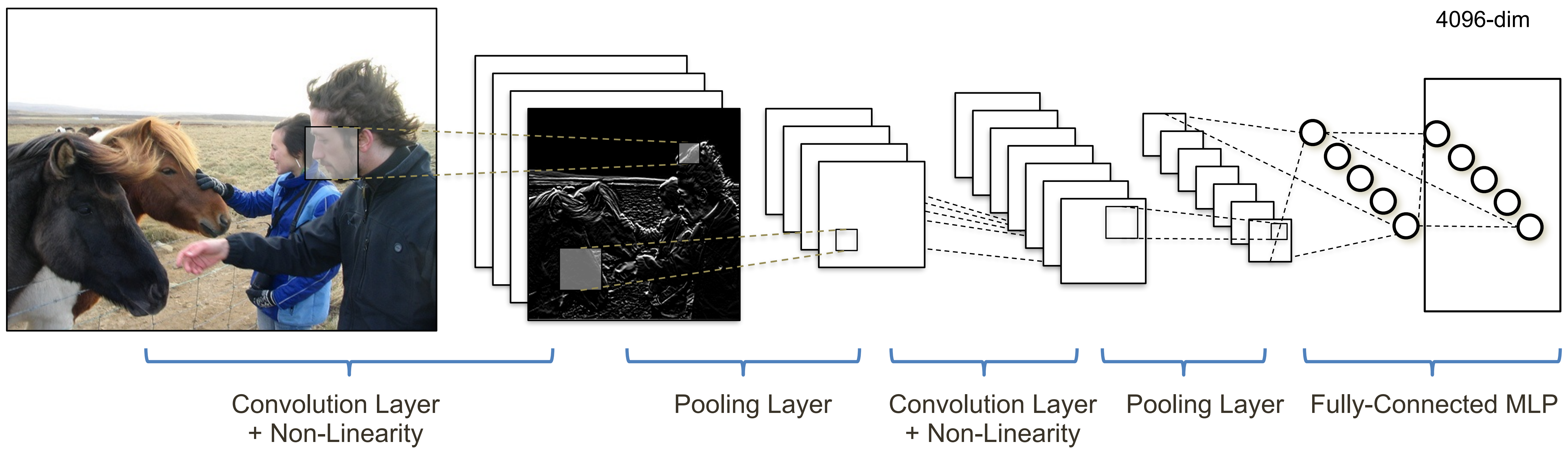
Image Embedding (VGGNet)



* slide from Dhruv Batra

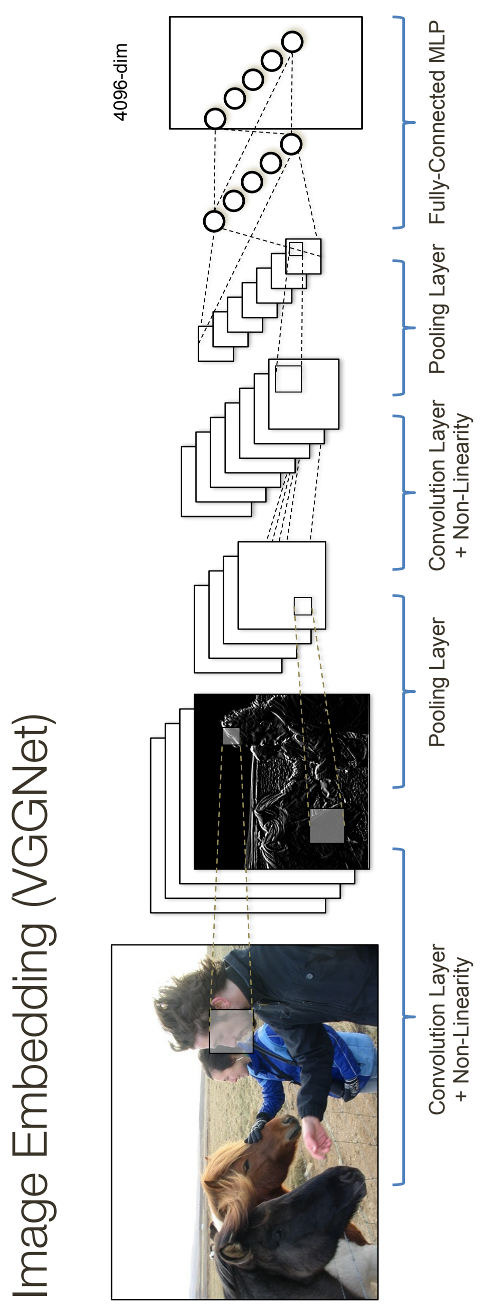
Neural Image **Captioning**

Image Embedding (VGGNet)

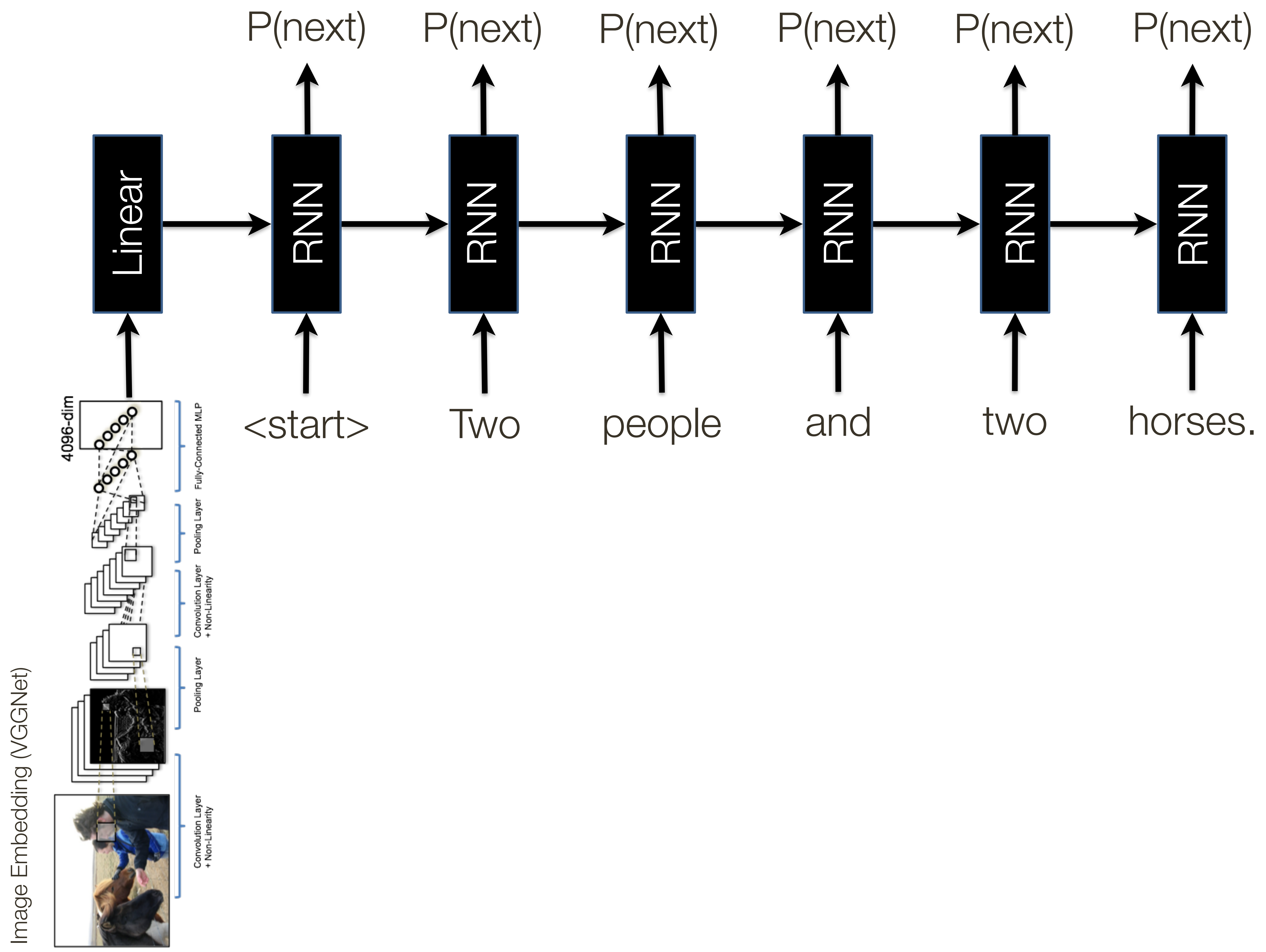


* slide from Dhruv Batra

Neural Image **Captioning**



Neural Image **Captioning**



* slide from Dhruv Batra

Neural Image **Captioning**

Good results

A cat sitting on a suitcase on the floor

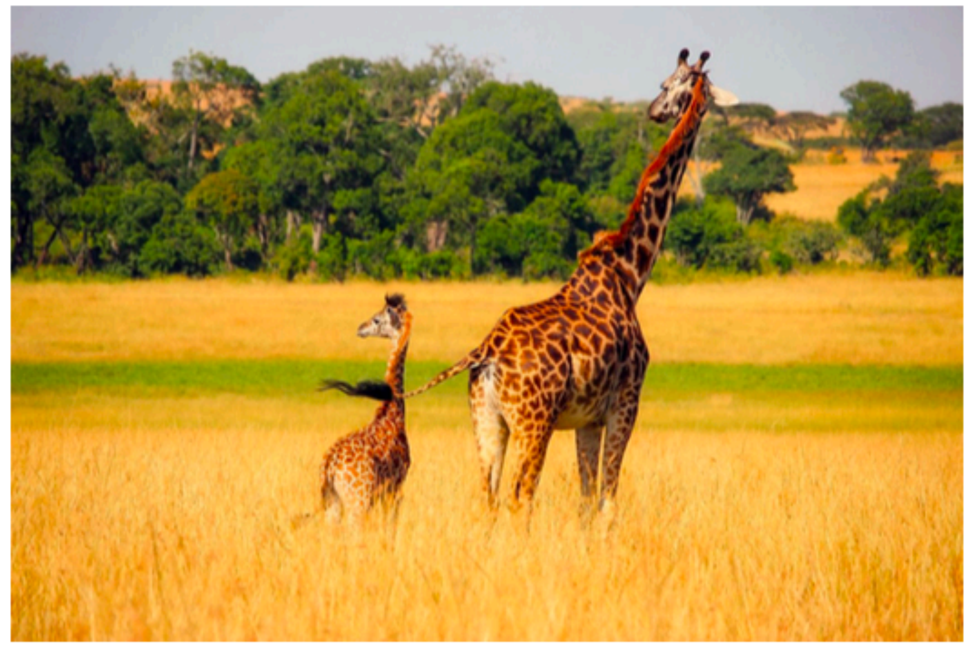
A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court



Two giraffes standing in a grassy field

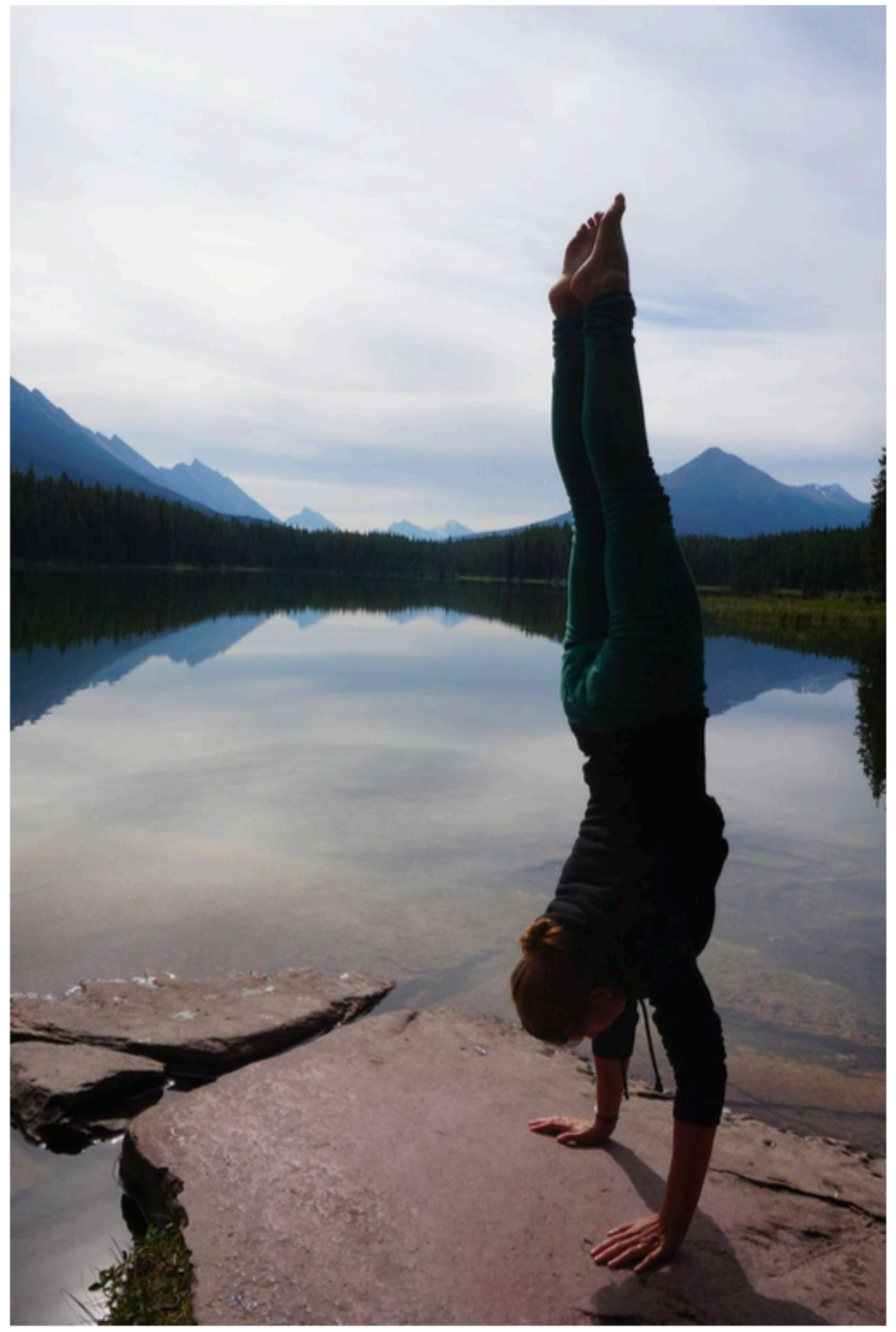
A man riding a dirt bike on a dirt track

Neural Image **Captioning**

Failure cases

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Summary

Common types of layers:

1. **Convolutional** Layer
   — Parameters define a set of learnable filters

2. **Pooling** Layer
   — Performs a downsampling along the spatial dimensions

3. **Fully-Connected** Layer
   — As in a regular neural network

Each layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function

Summary

The parameters of a neural network are learned using **backpropagation**, which computes gradients via recursive application of the chain rule

A **convolutional neural network** assumes inputs are images, and constrains the network architecture to reduce the number of parameters

A **convolutional layer** applies a set of learnable filters

A **pooling layer** performs spatial downsampling

A **fully-connected** layer is the same as in a regular neural network

Convolutional neural networks can be seen as learning a hierarchy of filters