Indian Institute of Information Technology, Allahabad

CVBL IIIT Allahabad

Convolutional Neural Network

By

Dr. Shiv Ram Dubey
Assistant Professor
Computer Vision And Biometrics Lab (CVBL)
Department Of Information Technology
Indian Institute Of Information Technology, Allahabad

Email: srdubey@iiita.ac.in Web: https://profile.iiita.ac.in/srdubey/

DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for Academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

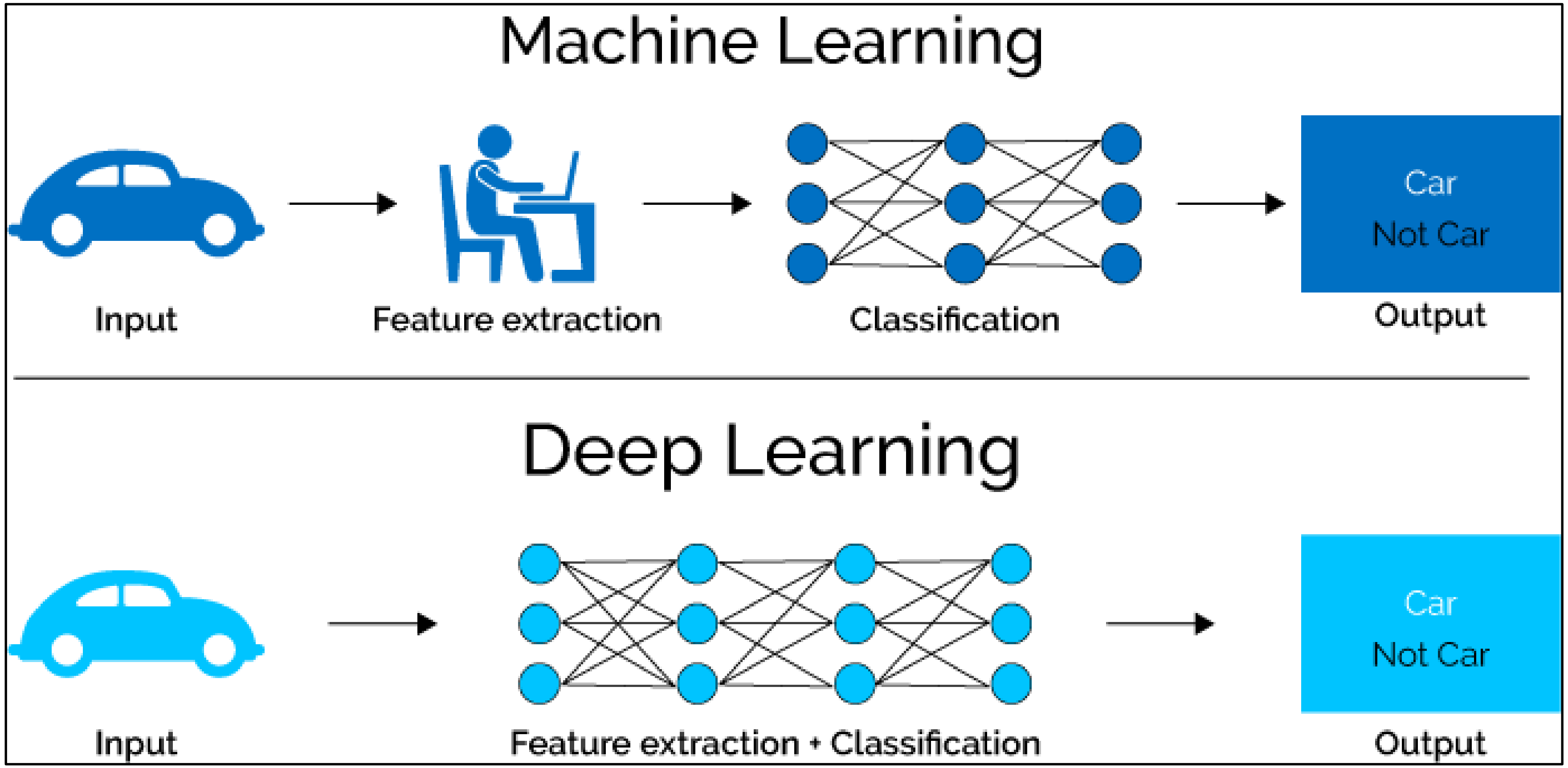
OUTLINE

- ❑ Convolutional Neural Networks (CNNs):
  - CNN architecture
  - Different layers
  - Progress
- ❑ CNN Architectures for Classification:
  - AlexNet, VGG, GoogleNet, ResNet
- ❑ CNN Architectures for Object Detection:
  - R-CNN, Fast R-CNN, Faster R-CNN, YOLO
- ❑ CNN Architectures for Segmentation:
  - U-Net, SegNet, Mask R-CNN
- ❑ CNN Architectures for Generation:
  - Pix2Pix, CycleGAN

Most material from CS231n CNN for Visual Recognition course at Stanford

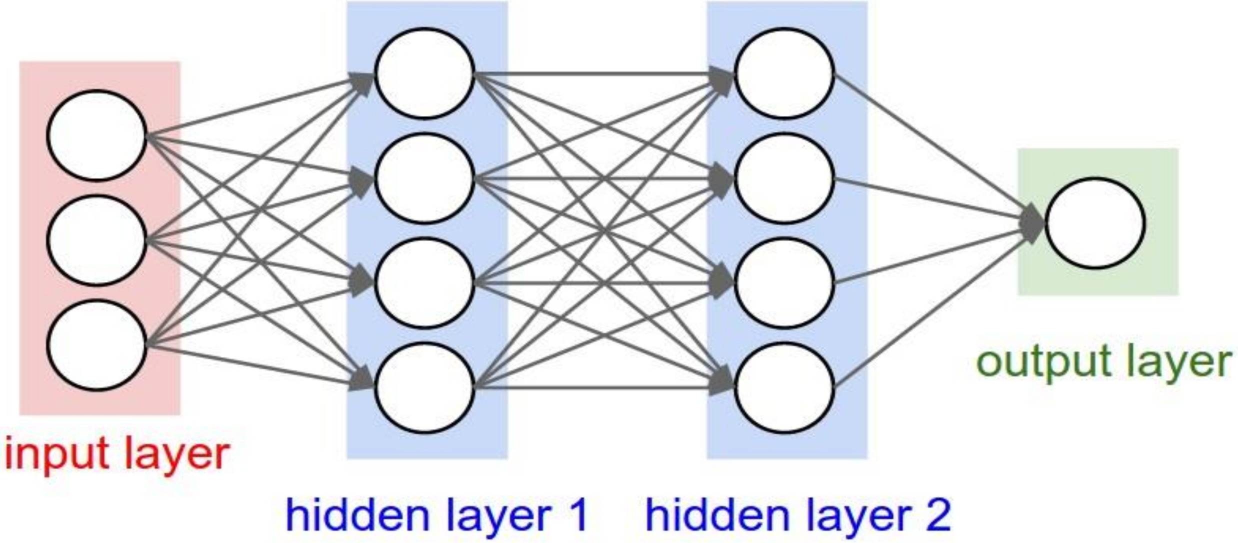
What is Deep Learning (DL) ?

- A machine learning subfield of learning **representations** of data. Exceptional effective at **learning patterns**.
- Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**
- If you provide the system **tons of information**, it begins to understand it and respond in useful ways.



https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

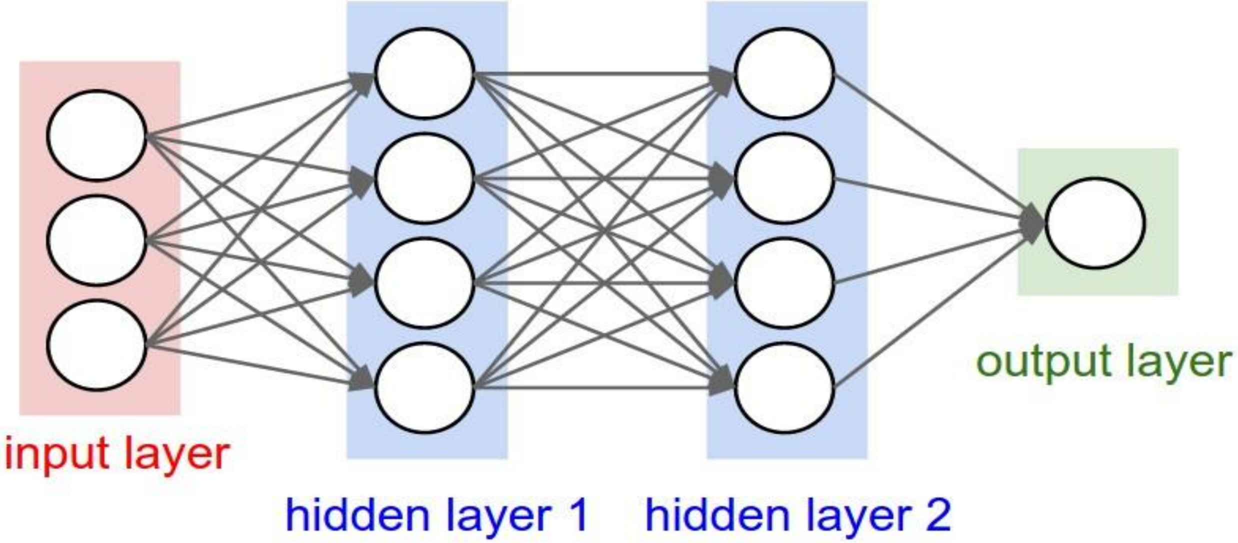
MULTI-LAYER NEURAL NETWORK & IMAGE



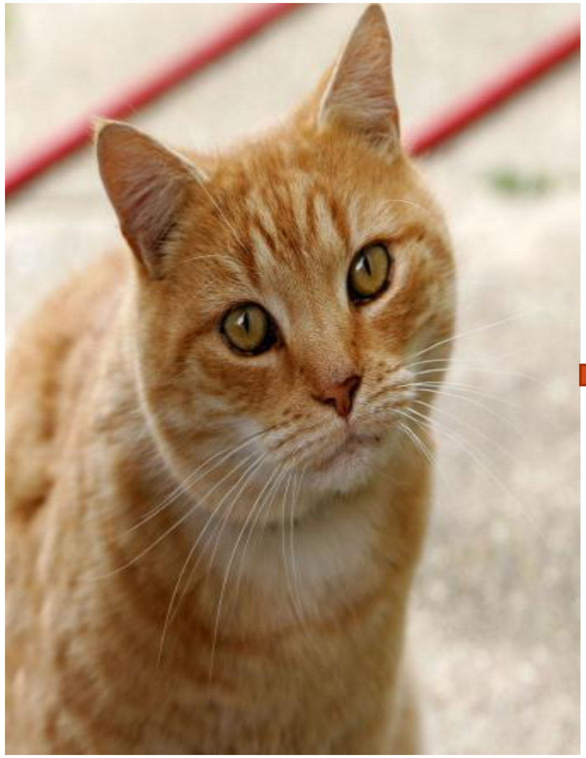
How to apply NN over Image?

MULTI-LAYER NEURAL NETWORK & IMAGE

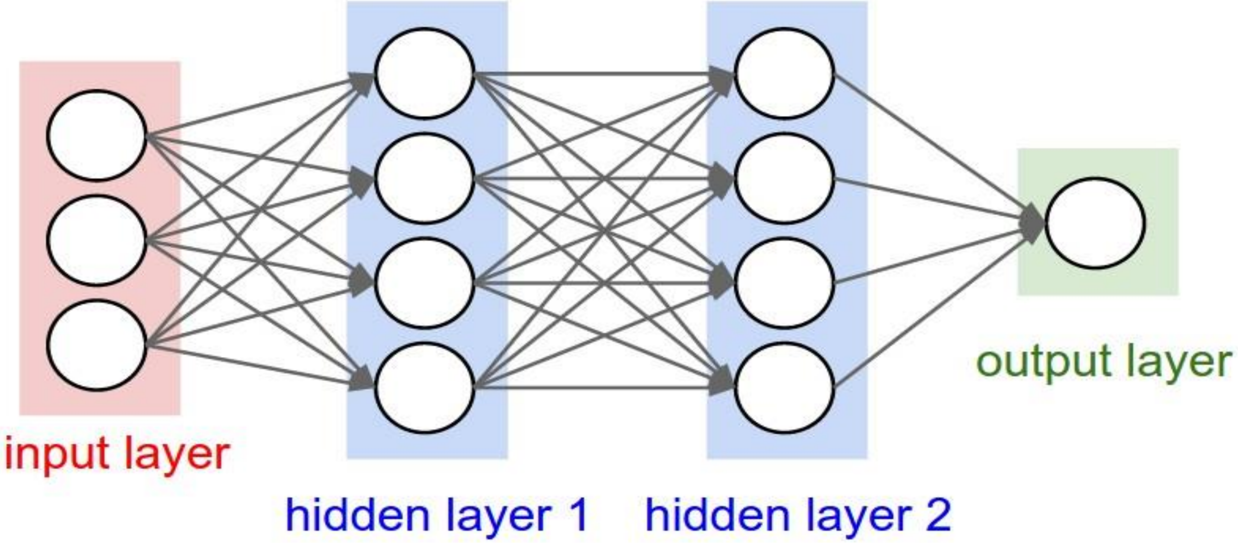
Stretch pixels in single column vector



MULTI-LAYER NEURAL NETWORK & IMAGE



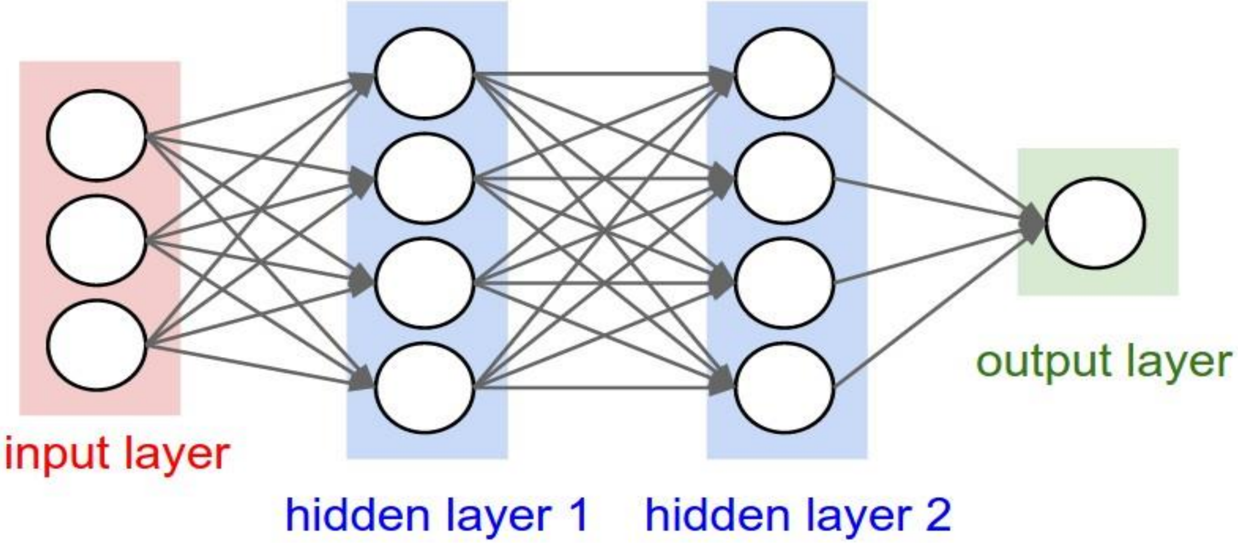
Stretch pixels in single column vector



Problems ?

MULTI-LAYER NEURAL NETWORK & IMAGE

Stretch pixels in single column vector



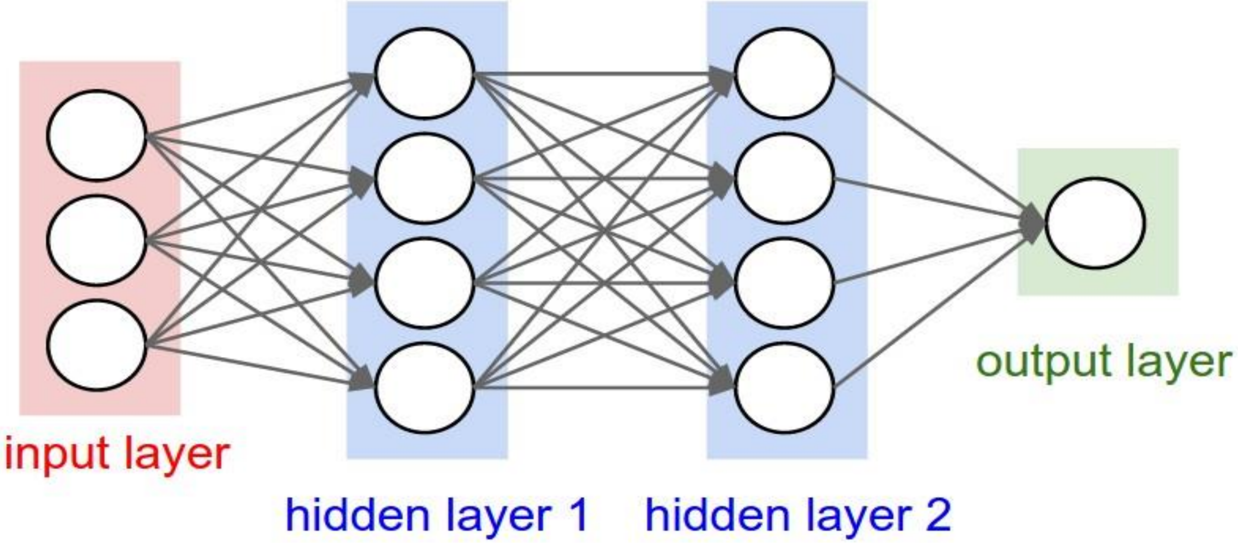
Problems:

High dimensionality

Local relationship

MULTI-LAYER NEURAL NETWORK & IMAGE

Stretch pixels in single column vector



Problems:

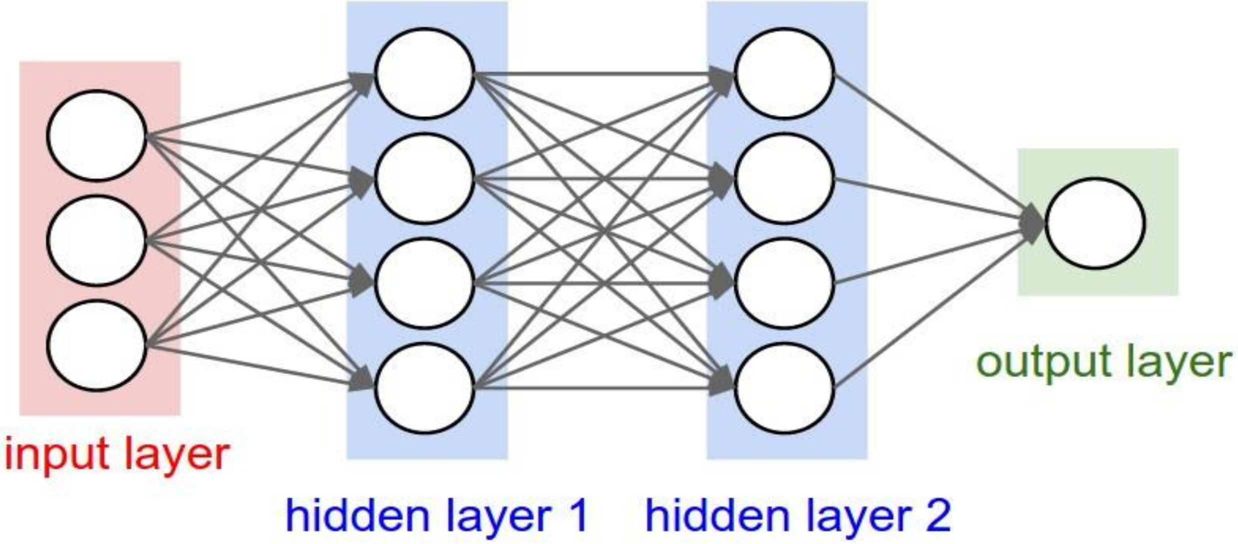
Solution ?

High dimensionality

Local relationship

MULTI-LAYER NEURAL NETWORK & IMAGE

Stretch pixels in single column vector



Problems:

High dimensionality

Local relationship

Solution:

Convolutional Neural Network

CONVOLUTIONAL NEURAL NETWORKS

- Also known as

  CNN,

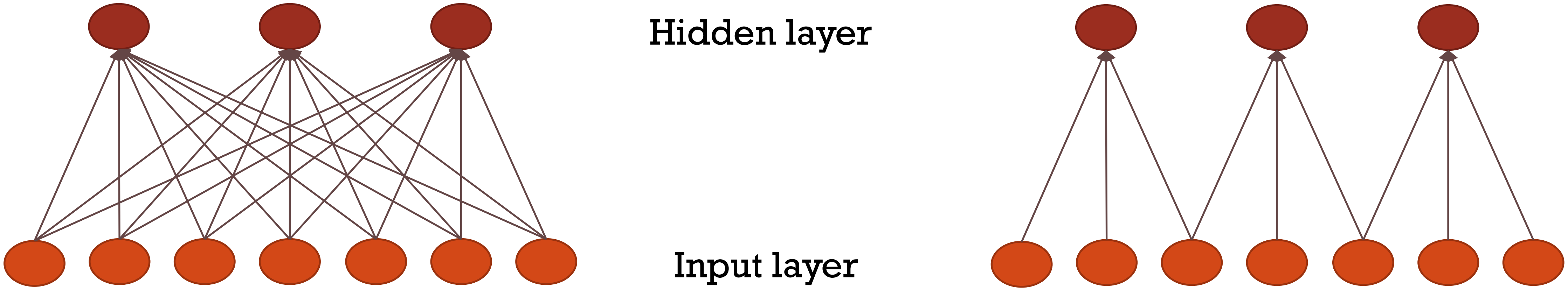
  ConvNet,

  DCN

- CNN = a multi-layer neural network with

  1. Local connectivity
  2. Weight sharing

CNN: LOCAL CONNECTIVITY

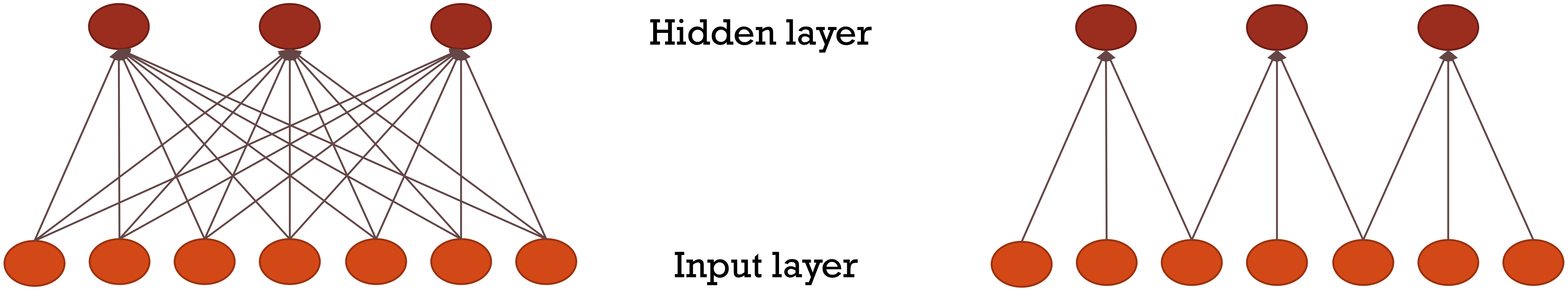


Global connectivity

Local connectivity

- # input units (neurons): 7
- # hidden units: 3

CNN: LOCAL CONNECTIVITY

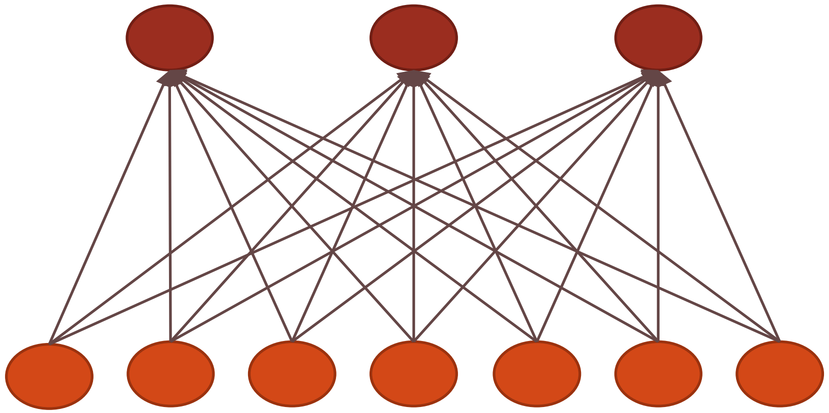


Global connectivity

Local connectivity

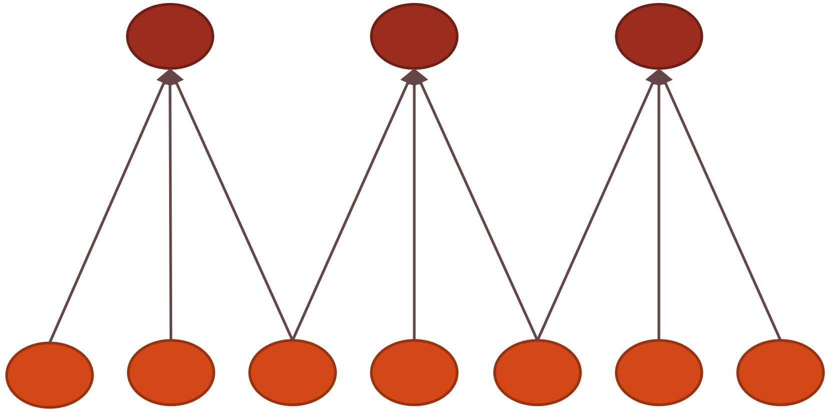
- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
  - Global connectivity: ?
  - Local connectivity: ?

CNN: LOCAL CONNECTIVITY



Hidden layer

Input layer

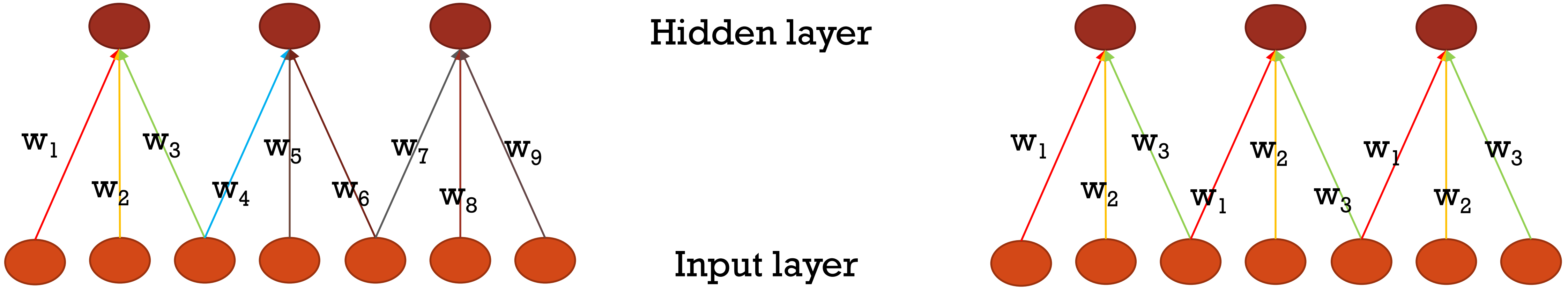


Global connectivity

Local connectivity

- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
  - Global connectivity: **3 x 7 = 21**
  - Local connectivity: **3 x 3 = 9**

CNN: WEIGHT SHARING

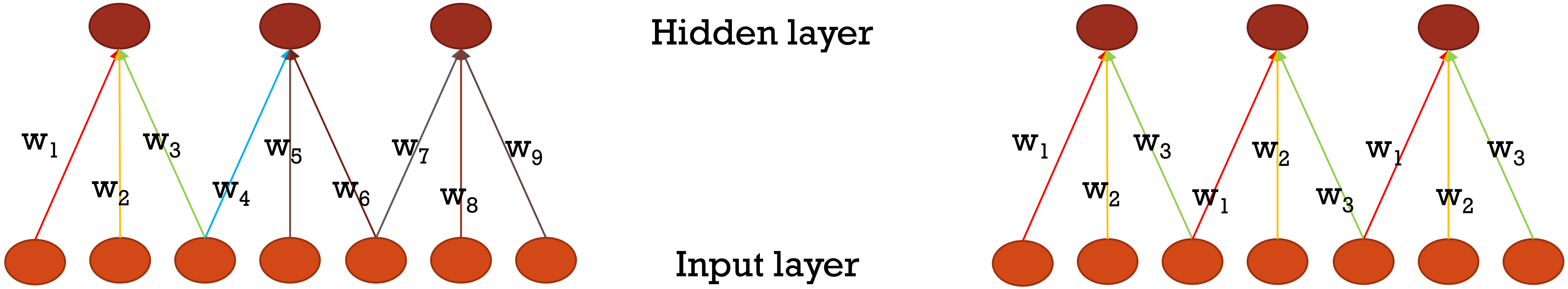


Without weight sharing

With weight sharing

- # input units (neurons): 7
- # hidden units: 3

CNN: WEIGHT SHARING

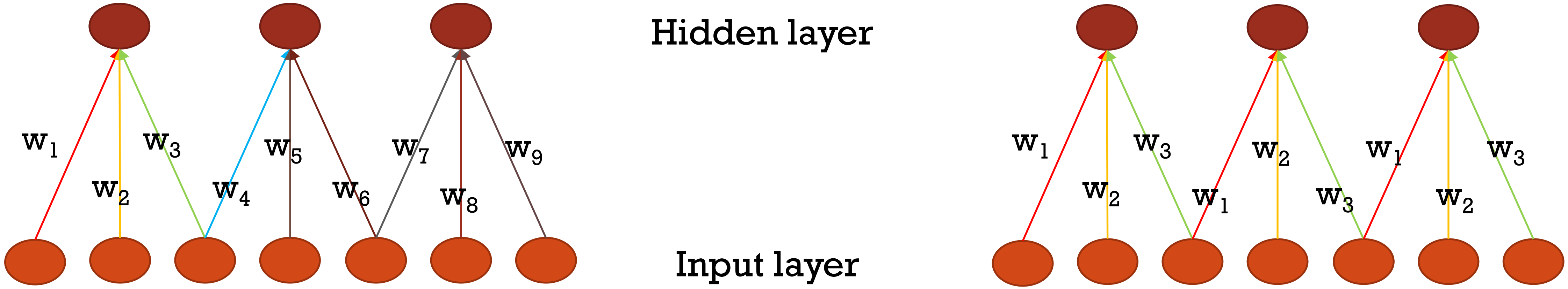


Without weight sharing

With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
  - Without weight sharing: ?
  - With weight sharing : ?

CNN: WEIGHT SHARING

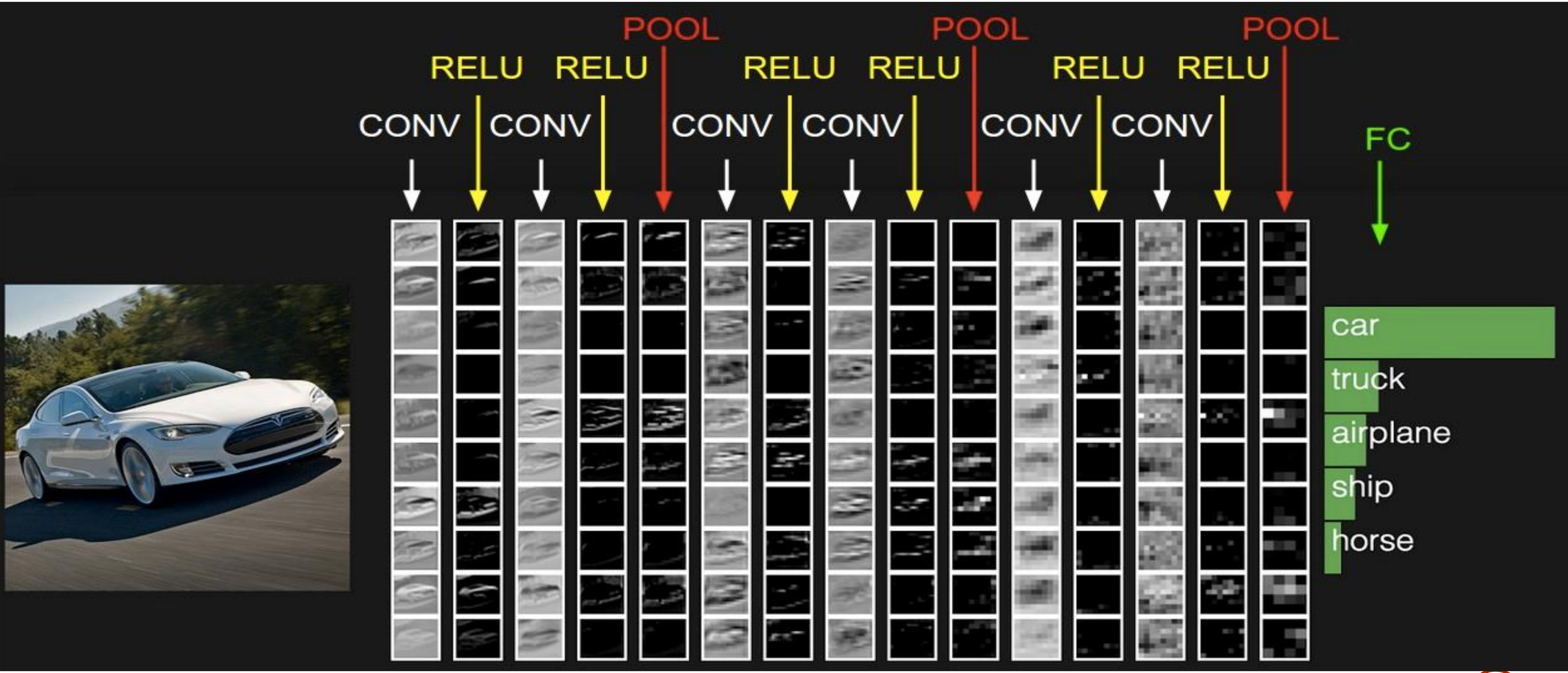


Without weight sharing

With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
  - Without weight sharing: **3 x 3 = 9**
  - With weight sharing : **3 x 1 = 3**

CONVOLUTIONAL NEURAL NETWORKS



Source: cs231n, Stanford University

LAYERS USED TO BUILD CONVNETS

Input Layer (Input image)

Convolutional Layer

Non-linearity Layer (such as Sigmoid, Tanh, ReLU, PReLU, ELU, Swish, etc.)

Pooling Layer (such as Max Pooling, Average Pooling, etc.)

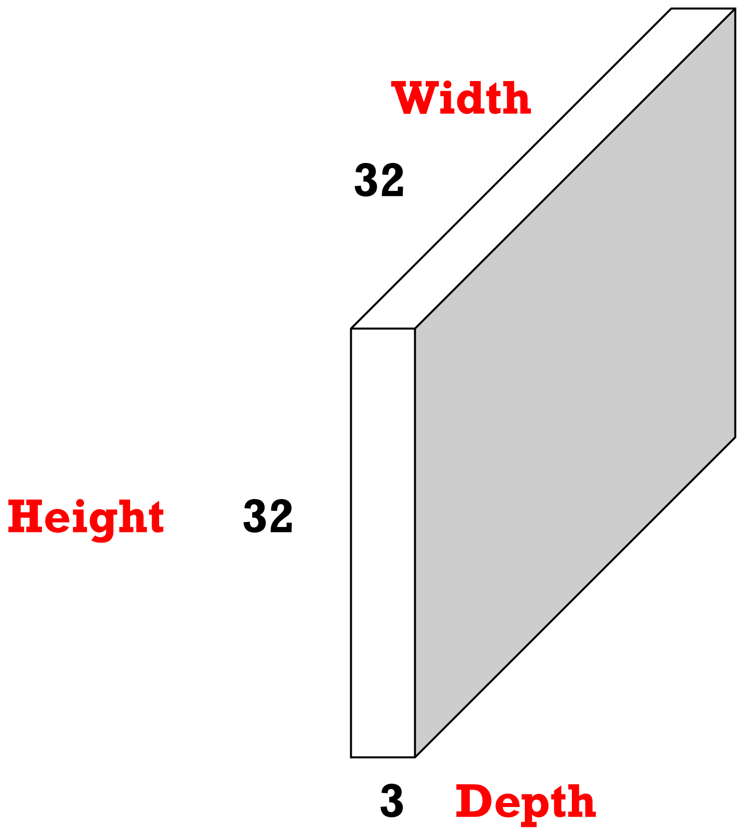
Fully-Connected Layer

Classification Layer (Softmax, etc.)

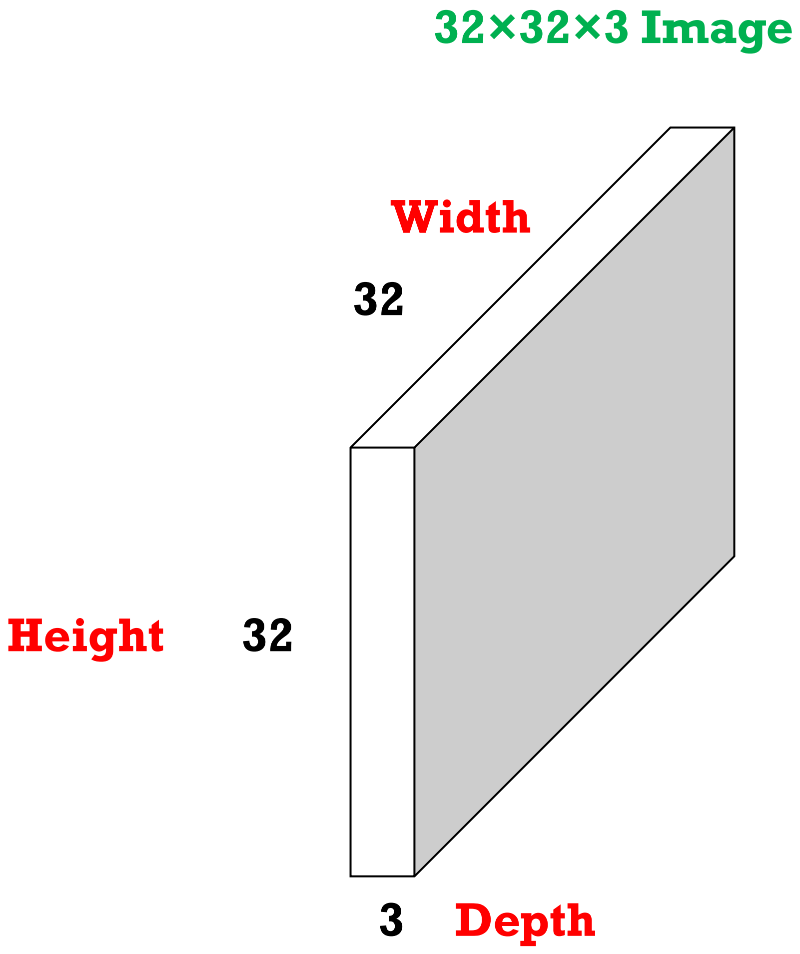
CONVOLUTIONAL LAYER

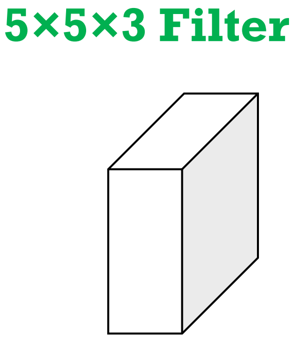
32×32×3 Image

-> preserve spatial structure



CONVOLUTIONAL LAYER





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

CONVOLUTIONAL LAYER

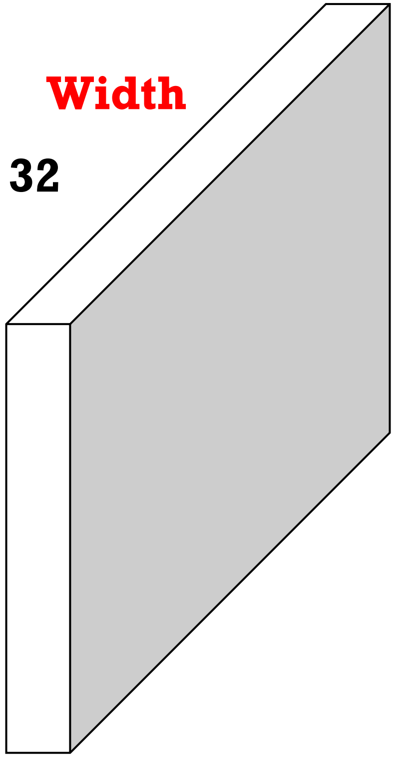
Handling multiple input channels

Filters always extend the full depth of the input volume

32×32×3 Image

Width

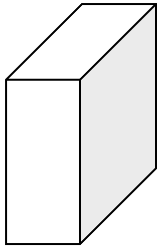
32



Height **32**

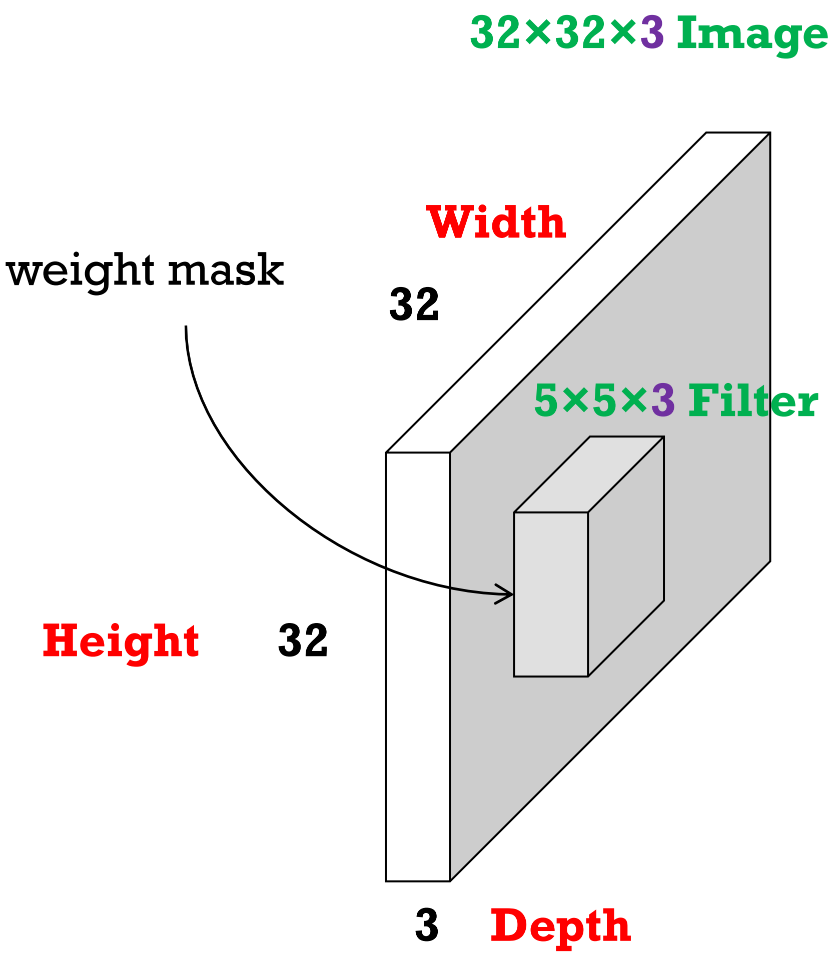
3 **Depth**

5×5×3 Filter

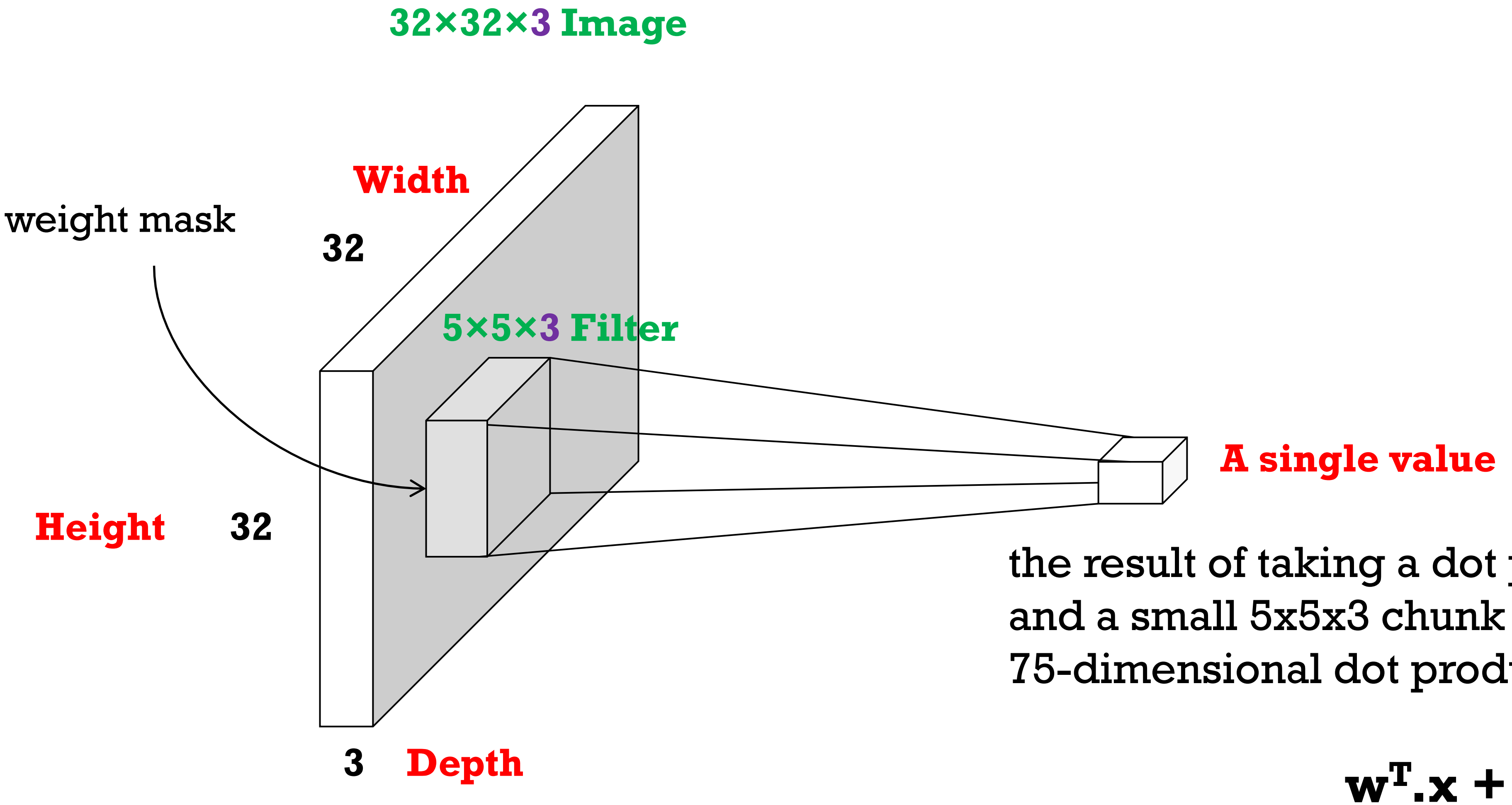


Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

CONVOLUTIONAL LAYER



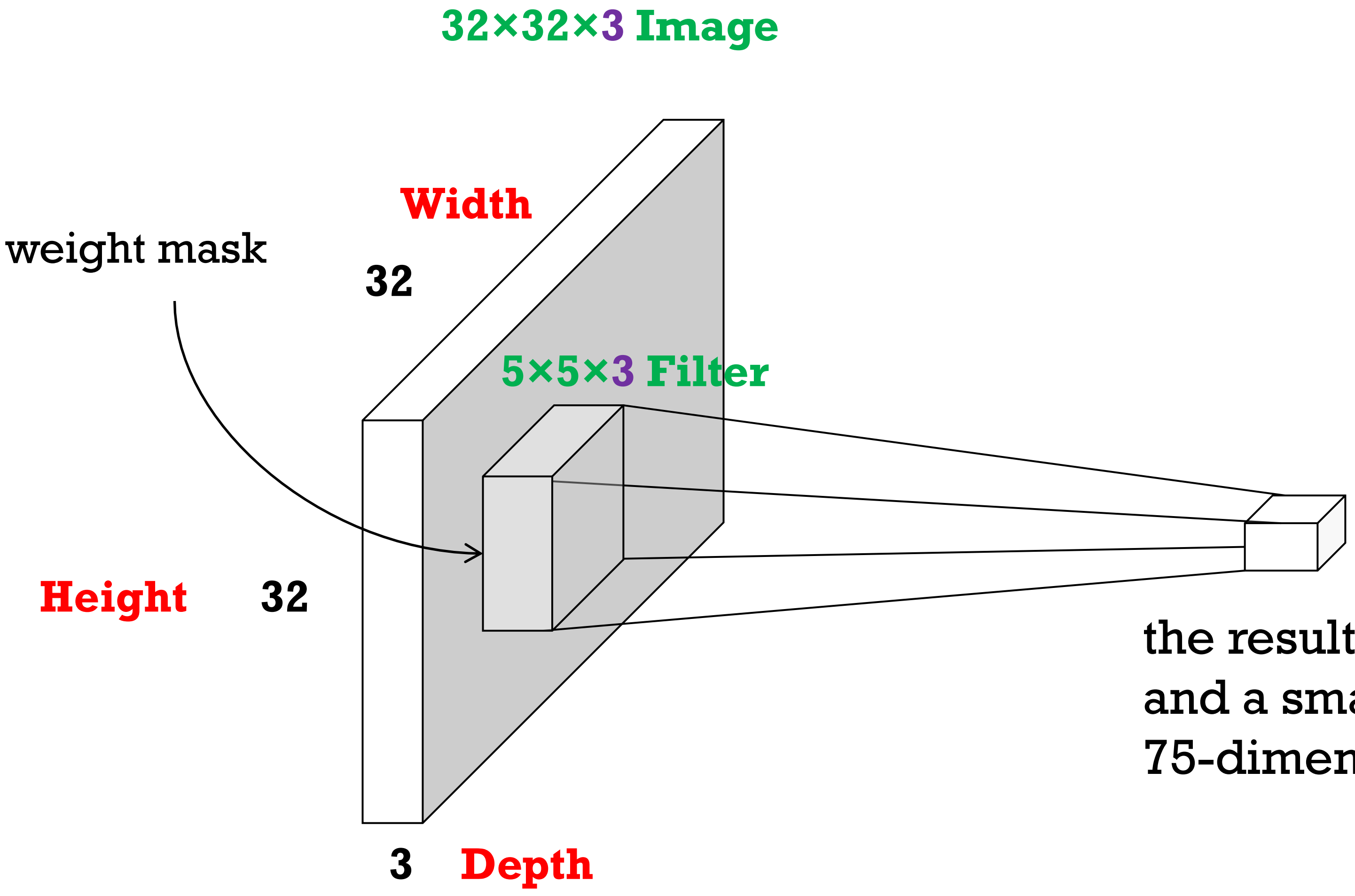
CONVOLUTIONAL LAYER

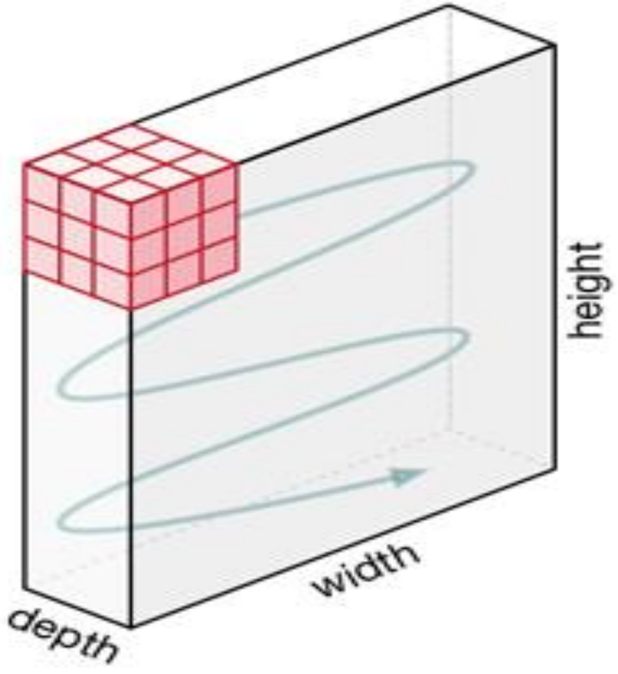


the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$\mathbf{w}^T.\mathbf{x} + \mathbf{b}$$

CONVOLUTIONAL LAYER

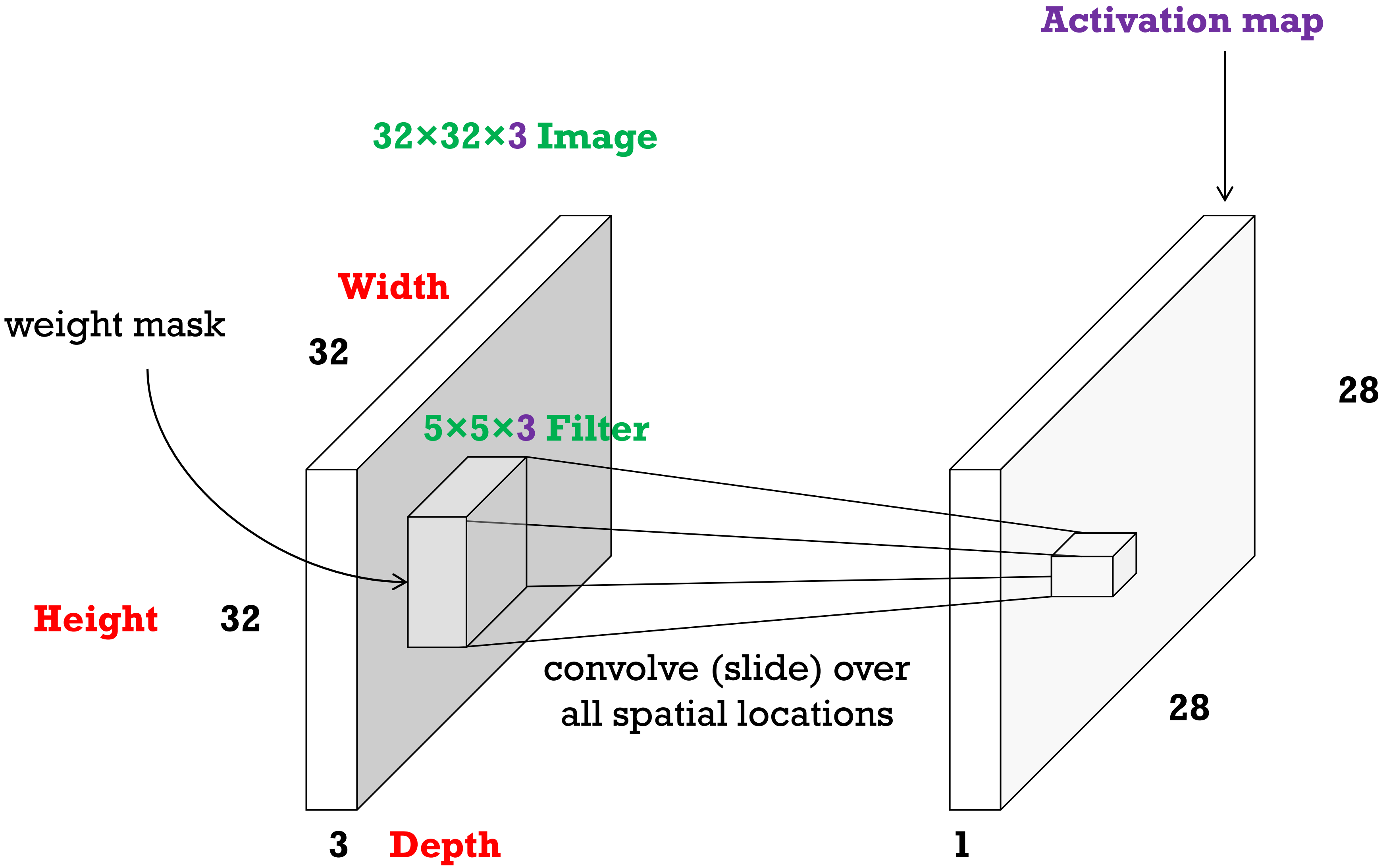


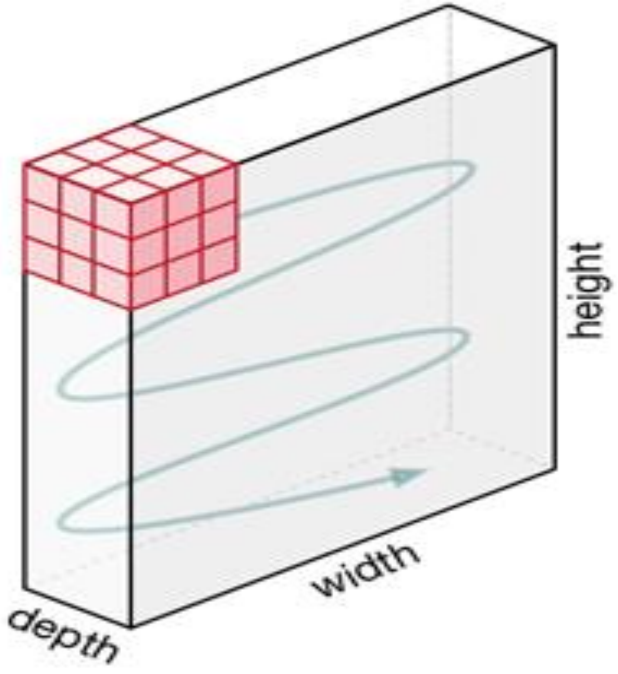


the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$\mathbf{w}^T.\mathbf{x} + \mathbf{b}$$

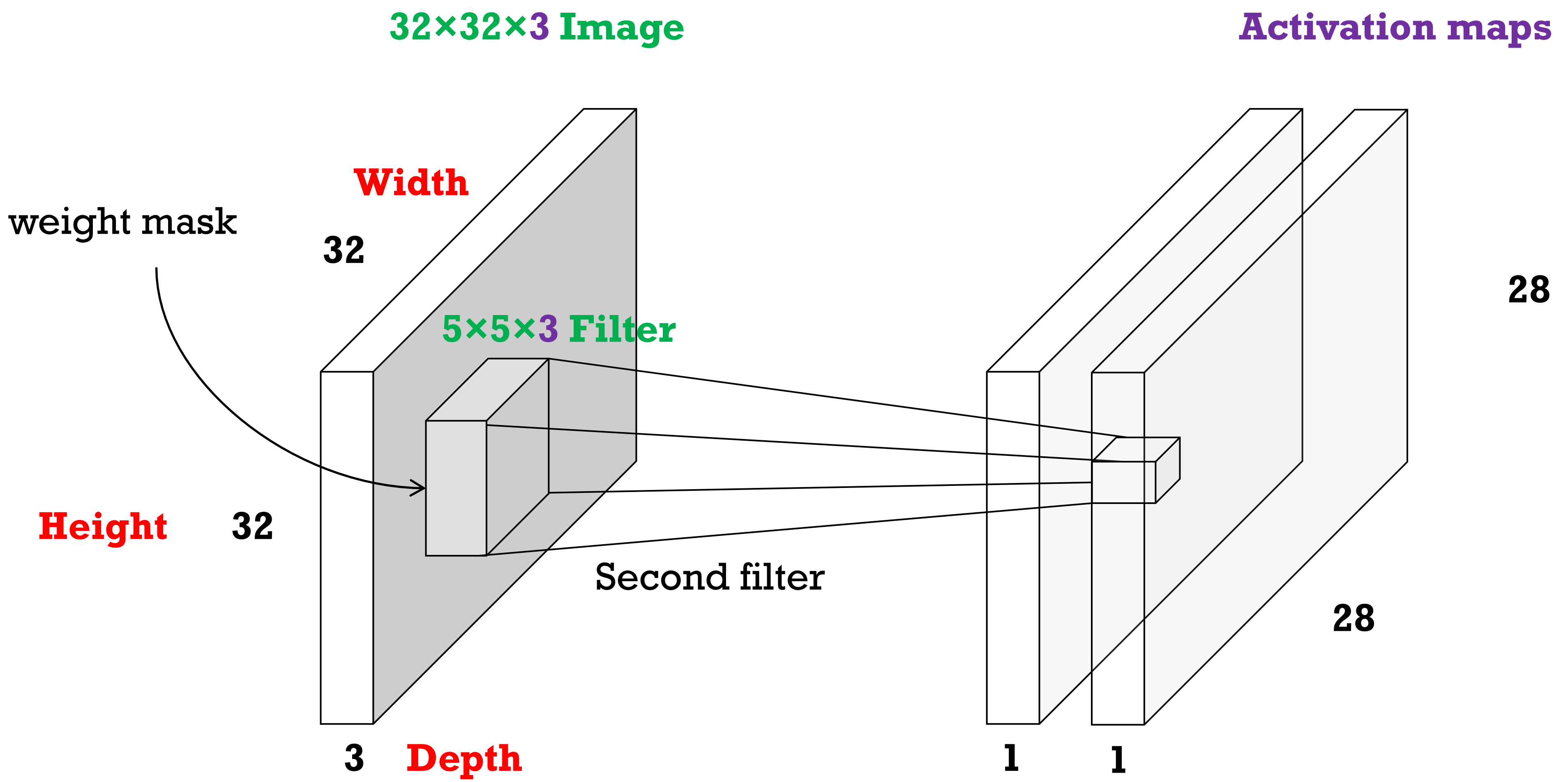
CONVOLUTIONAL LAYER





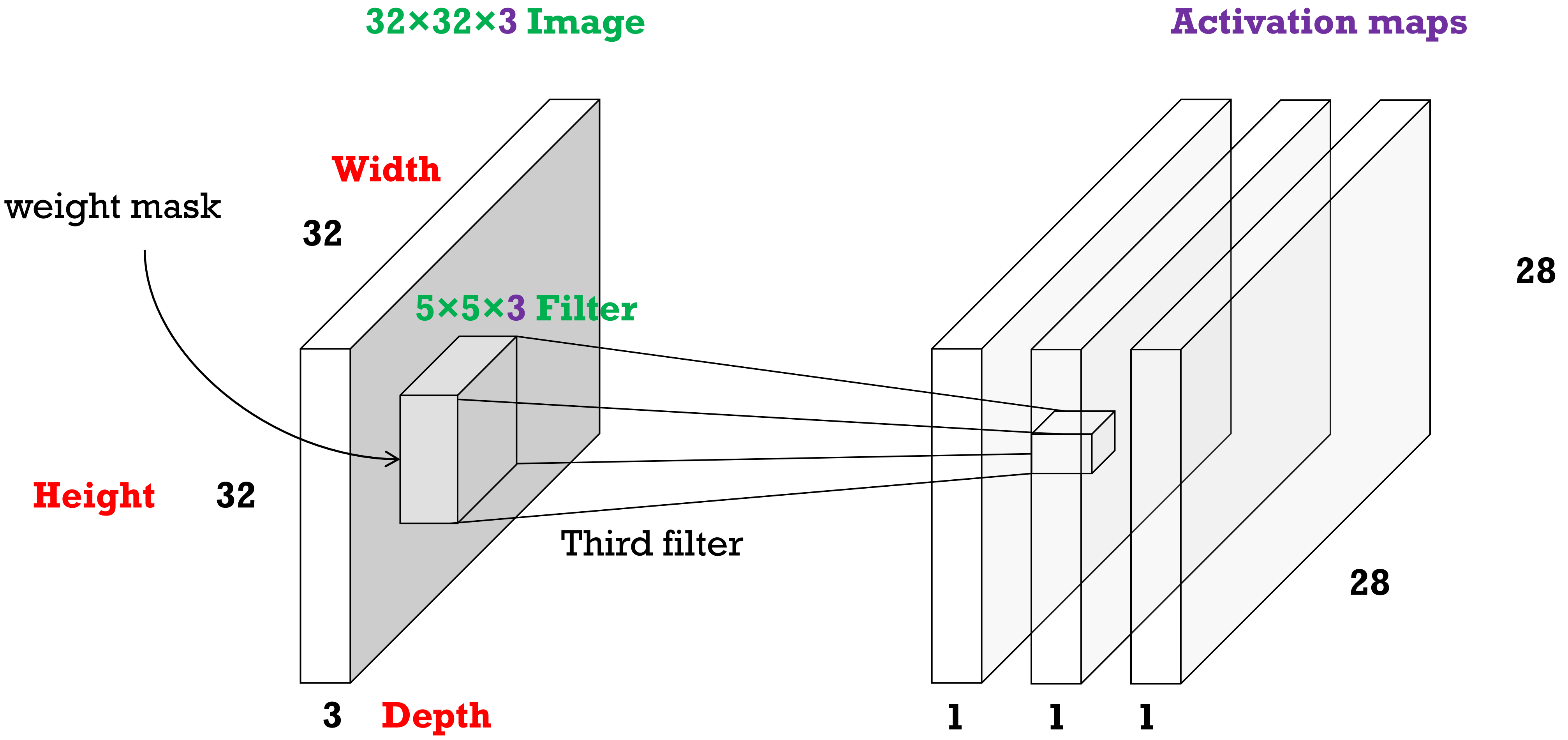
CONVOLUTIONAL LAYER

Handling multiple output maps



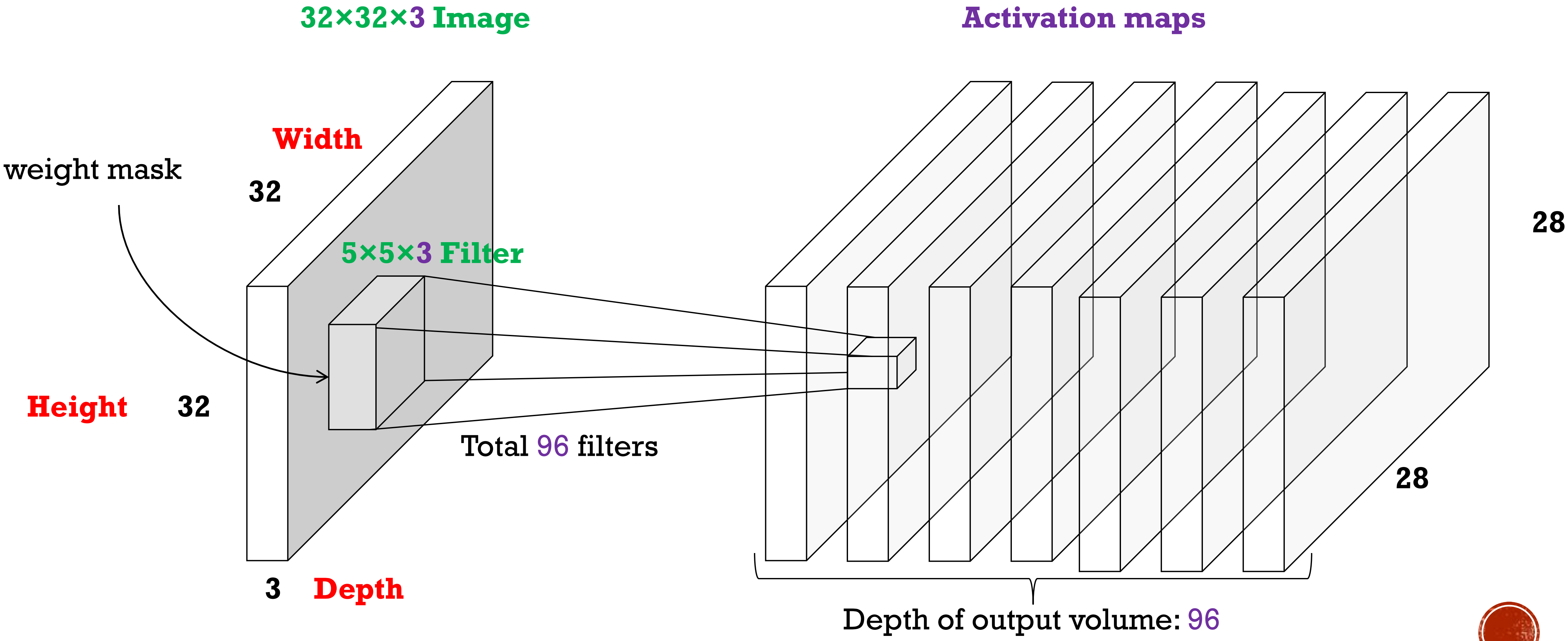
CONVOLUTIONAL LAYER

Handling multiple output maps



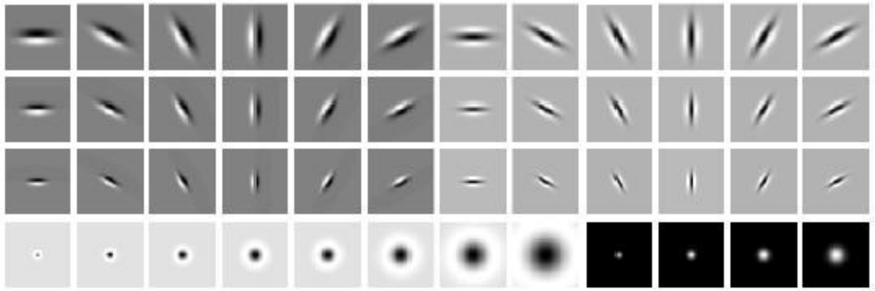
CONVOLUTIONAL LAYER

Handling multiple output maps

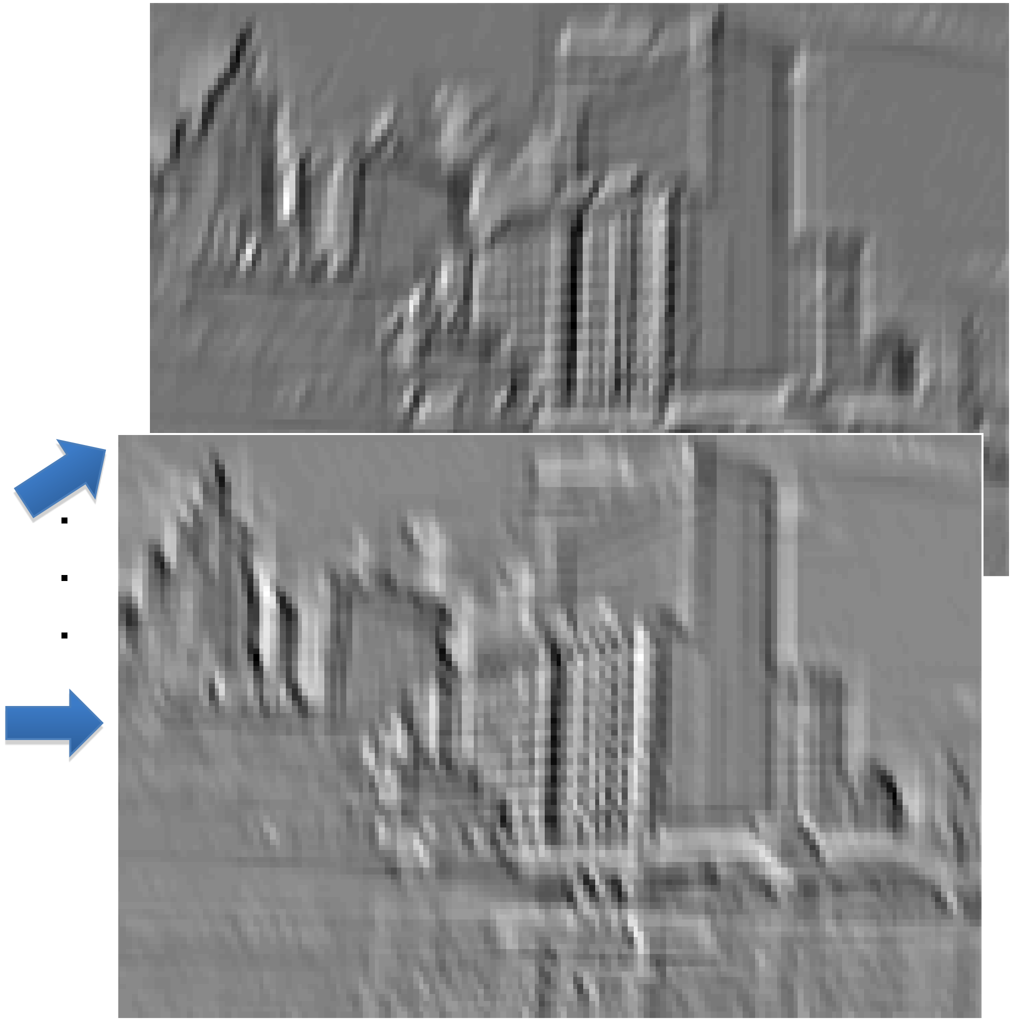


CONVOLUTION AND TRADITIONAL FEATURE EXTRACTION

bank of K filters



K feature maps



image

feature map

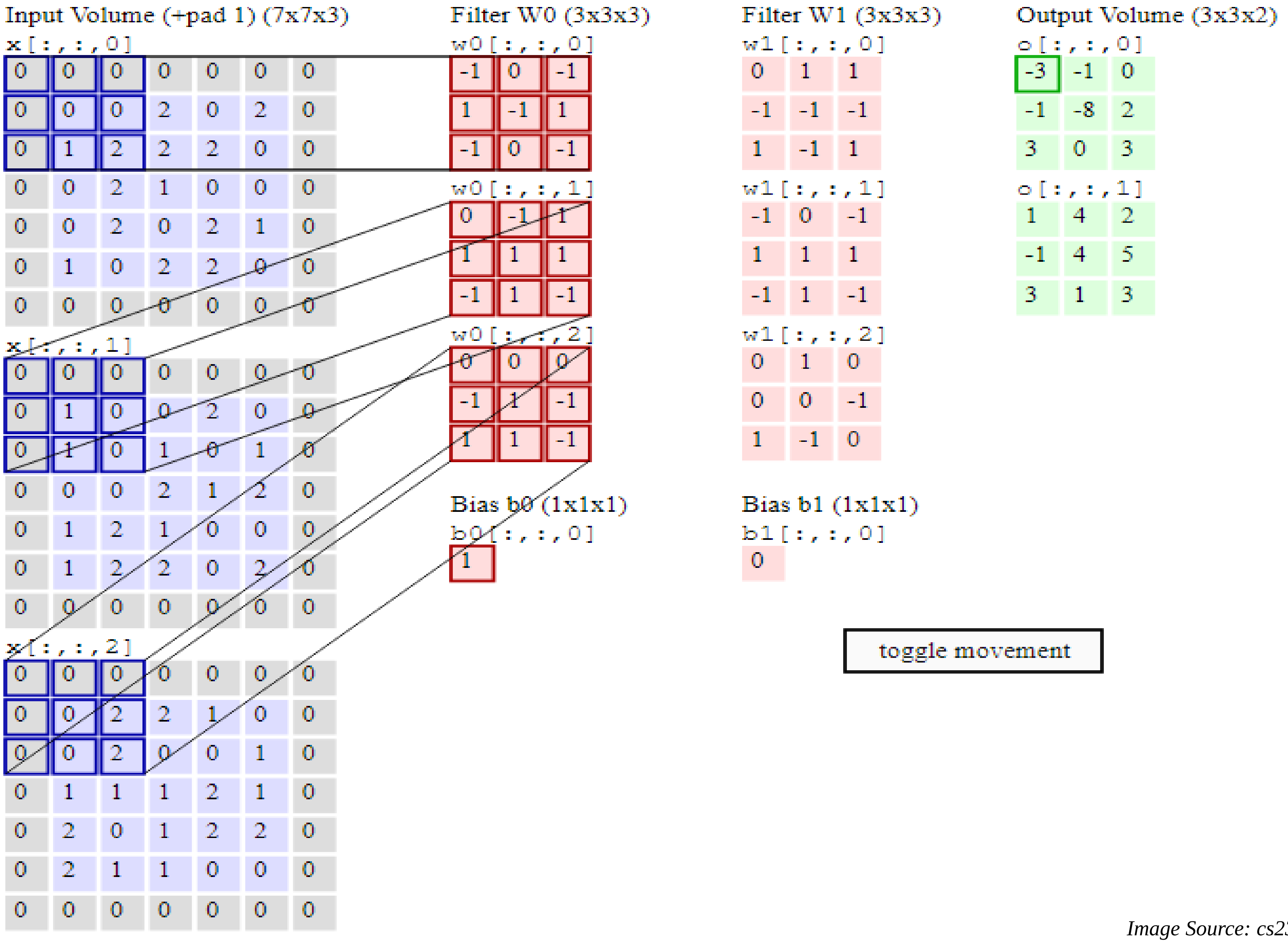


Image Source: cs231n, Oxford University

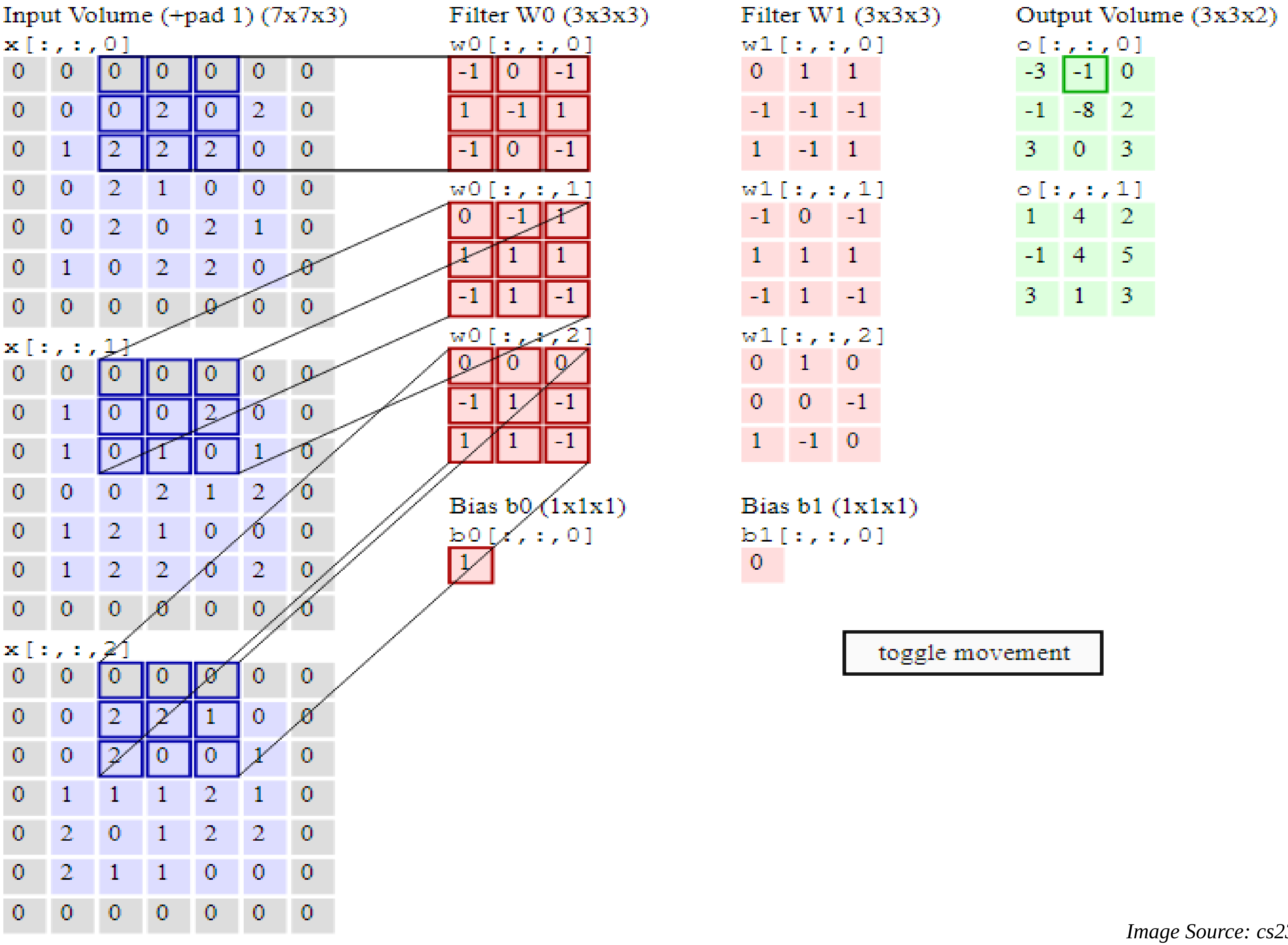


Image Source: cs231n, Oxford University

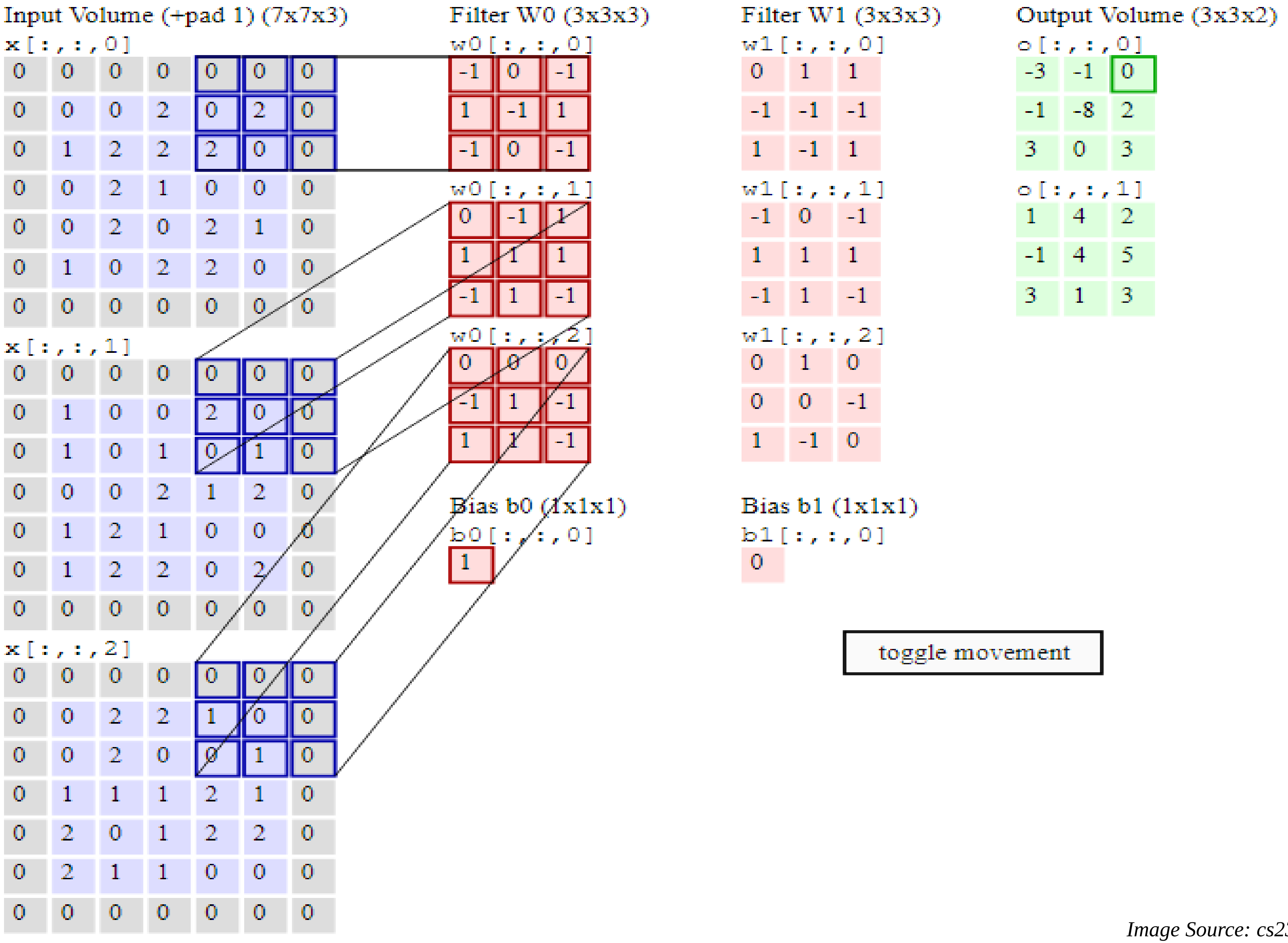
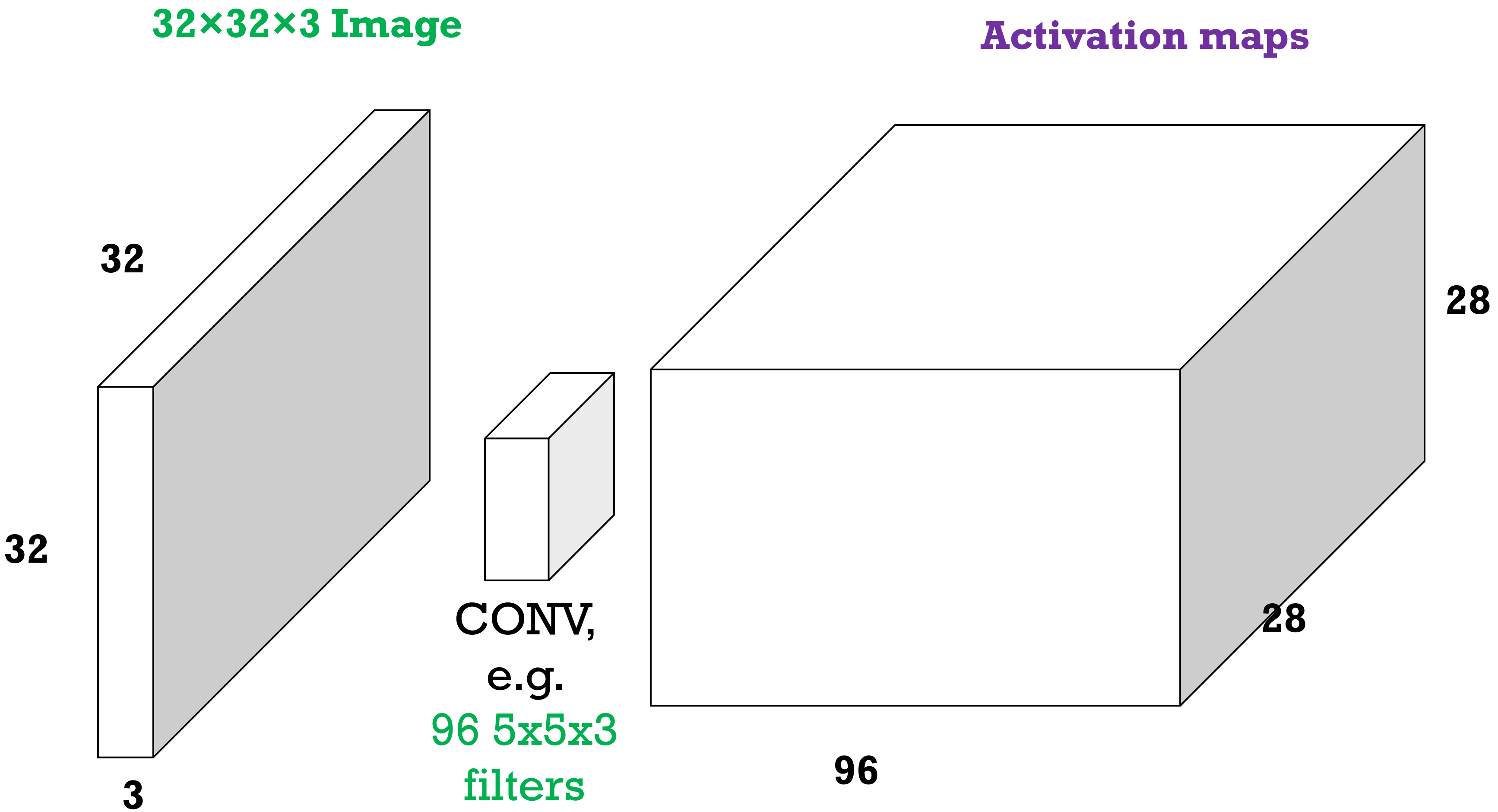


Image Source: cs231n, Oxford University

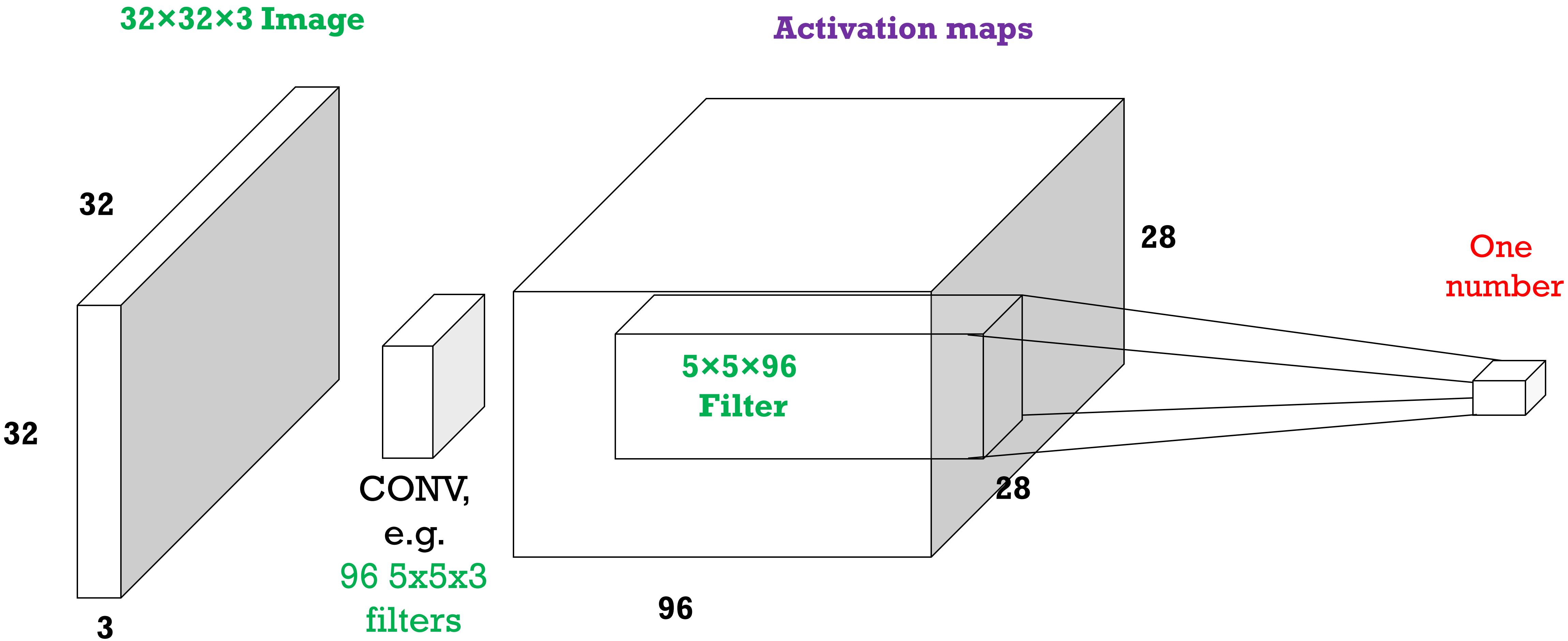
CONVOLUTIONAL LAYER

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



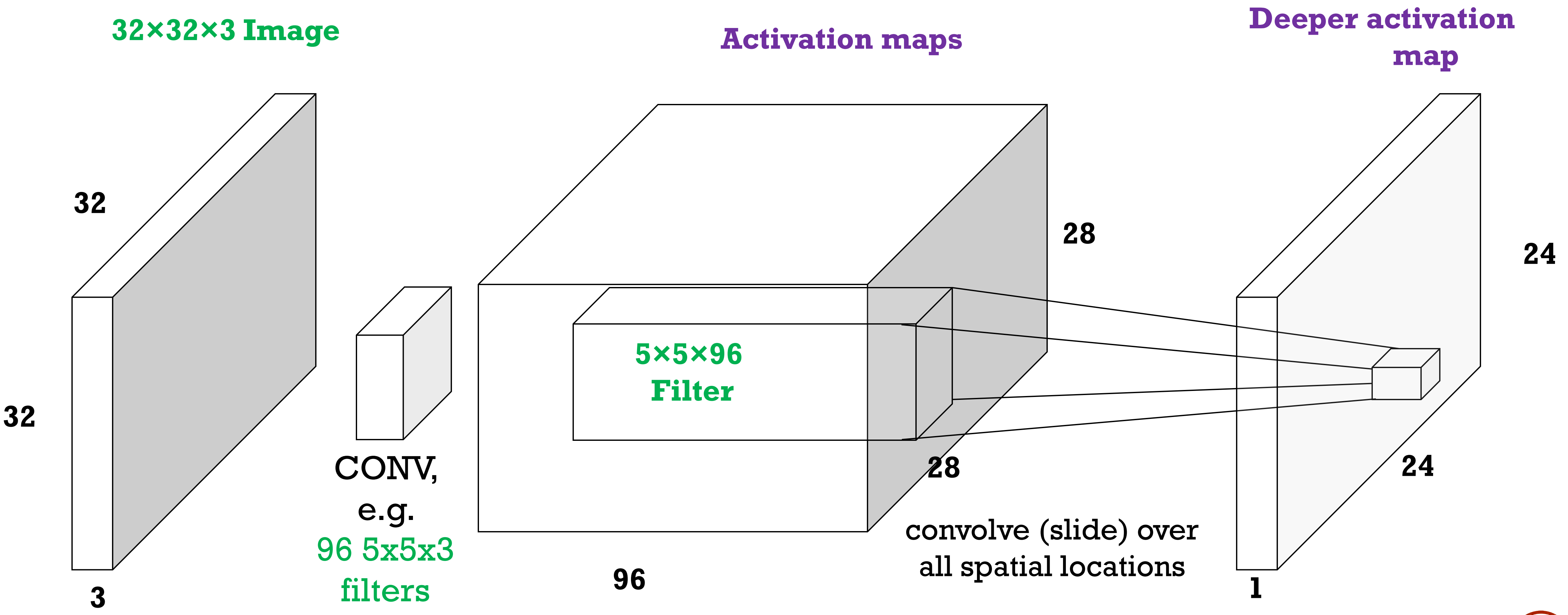
CONVOLUTIONAL LAYER

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



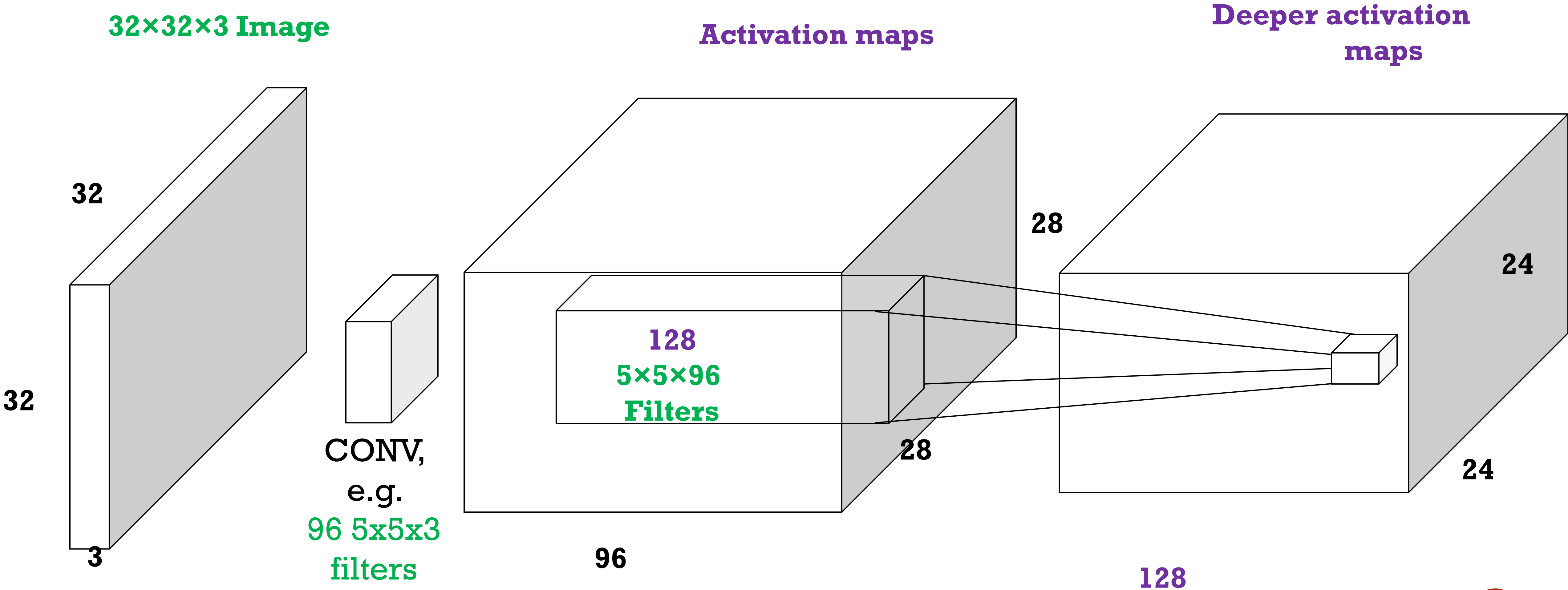
CONVOLUTIONAL LAYER

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



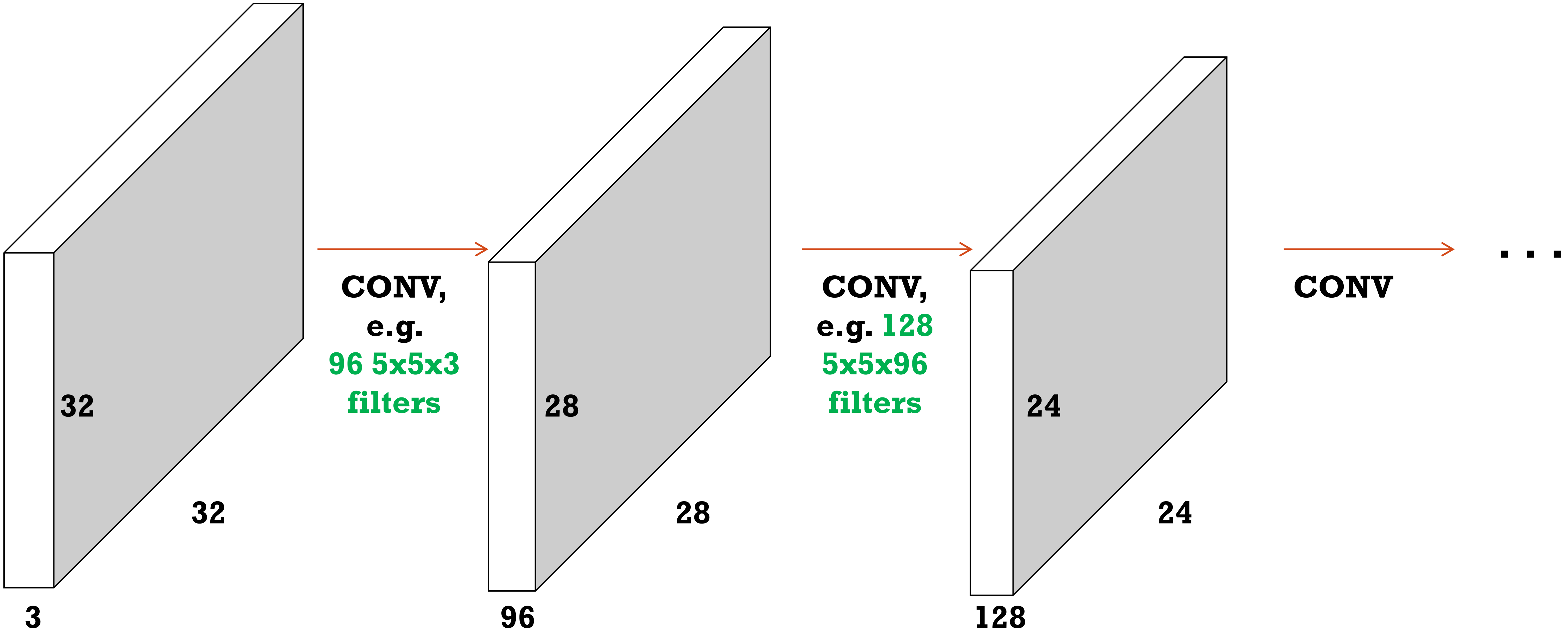
CONVOLUTIONAL LAYER

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



MULTILAYER CONVOLUTION

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



ANY CONVOLUTION LAYER

- Local connectivity
- Weight sharing
- Handling multiple input channels
- Handling multiple output maps

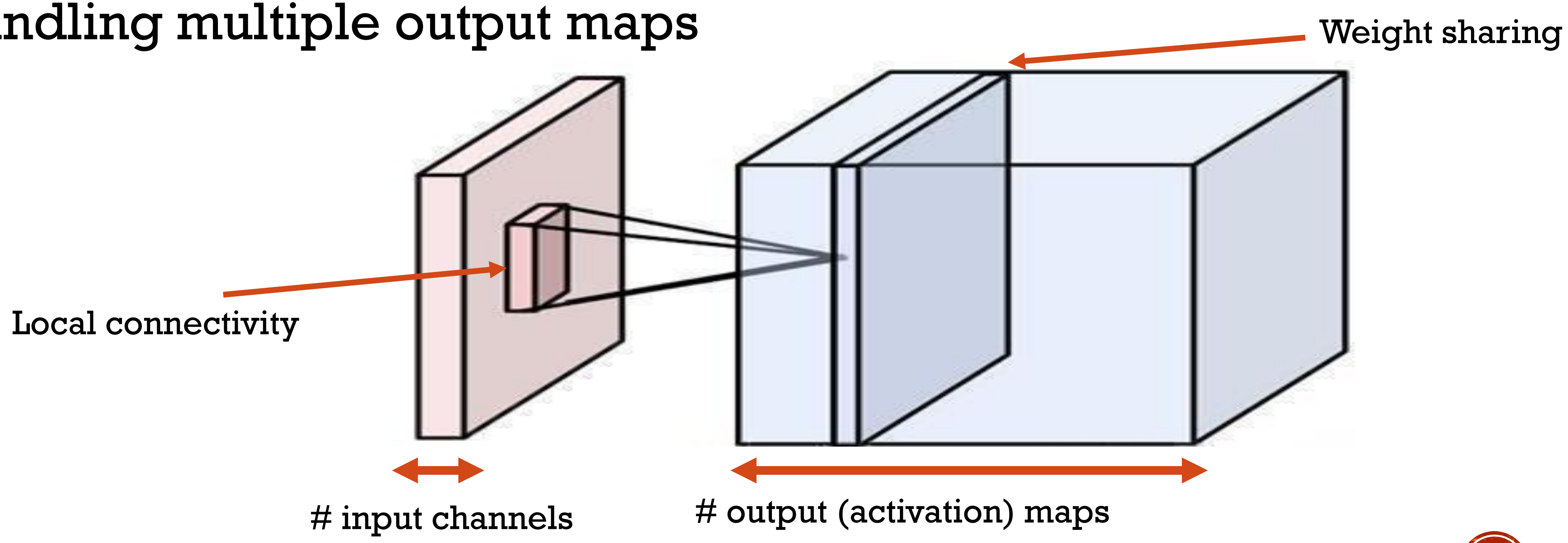
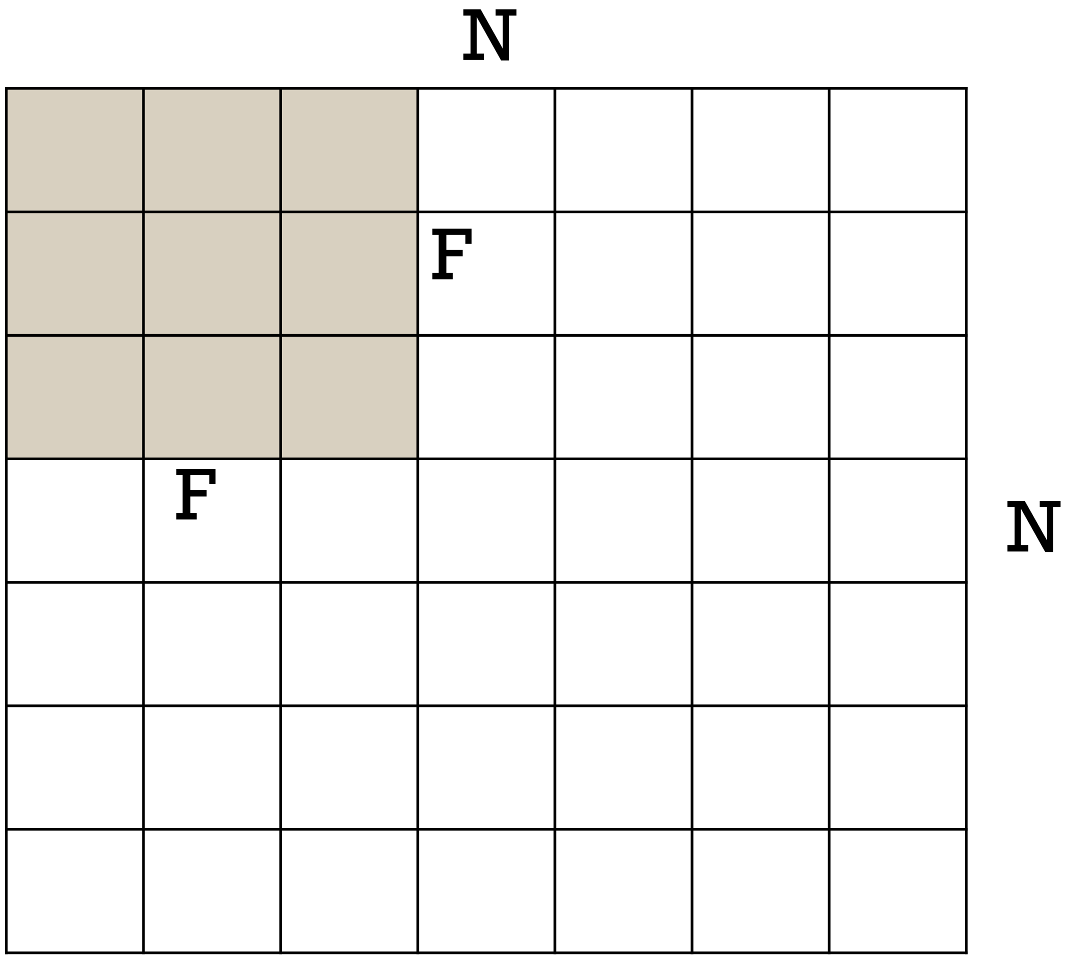


Image credit: A. Karpathy

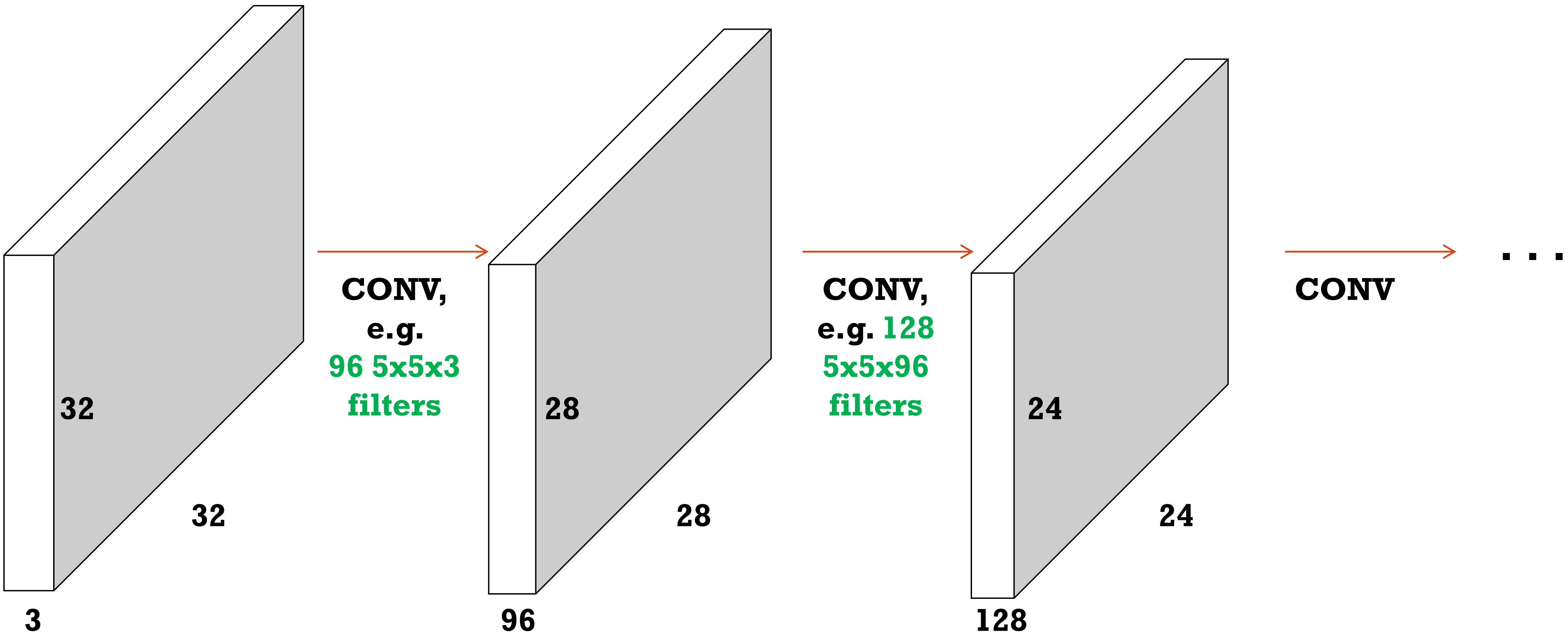
A CLOSER LOOK AT SPATIAL DIMENSIONS



Output size
(N - F) / stride + 1

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

A CLOSER LOOK AT SPATIAL DIMENSIONS



E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.

Source: cs231n, Stanford University

IN PRACTICE: COMMON TO ZERO PAD

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

e.g. input 7×7 (spatially)
3×3 filter, applied with stride 1
pad with 1 pixel border

7×7 Output

in general, common to see CONV layers with stride 1, filters of size F×F, and zero-padding with (F-1)/2. (will preserve size spatially)

e.g.
F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

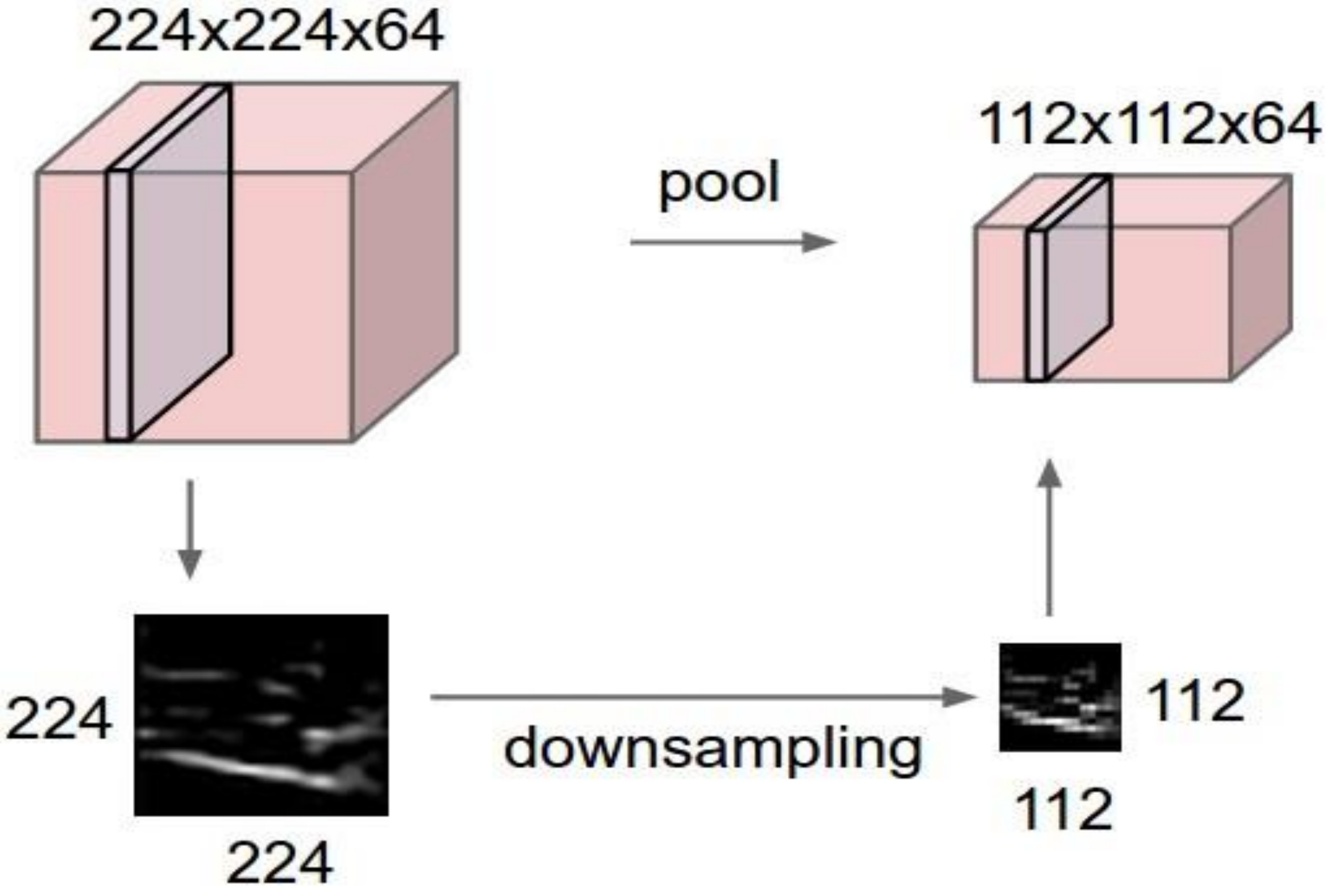
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.

Source: cs231n, Stanford University

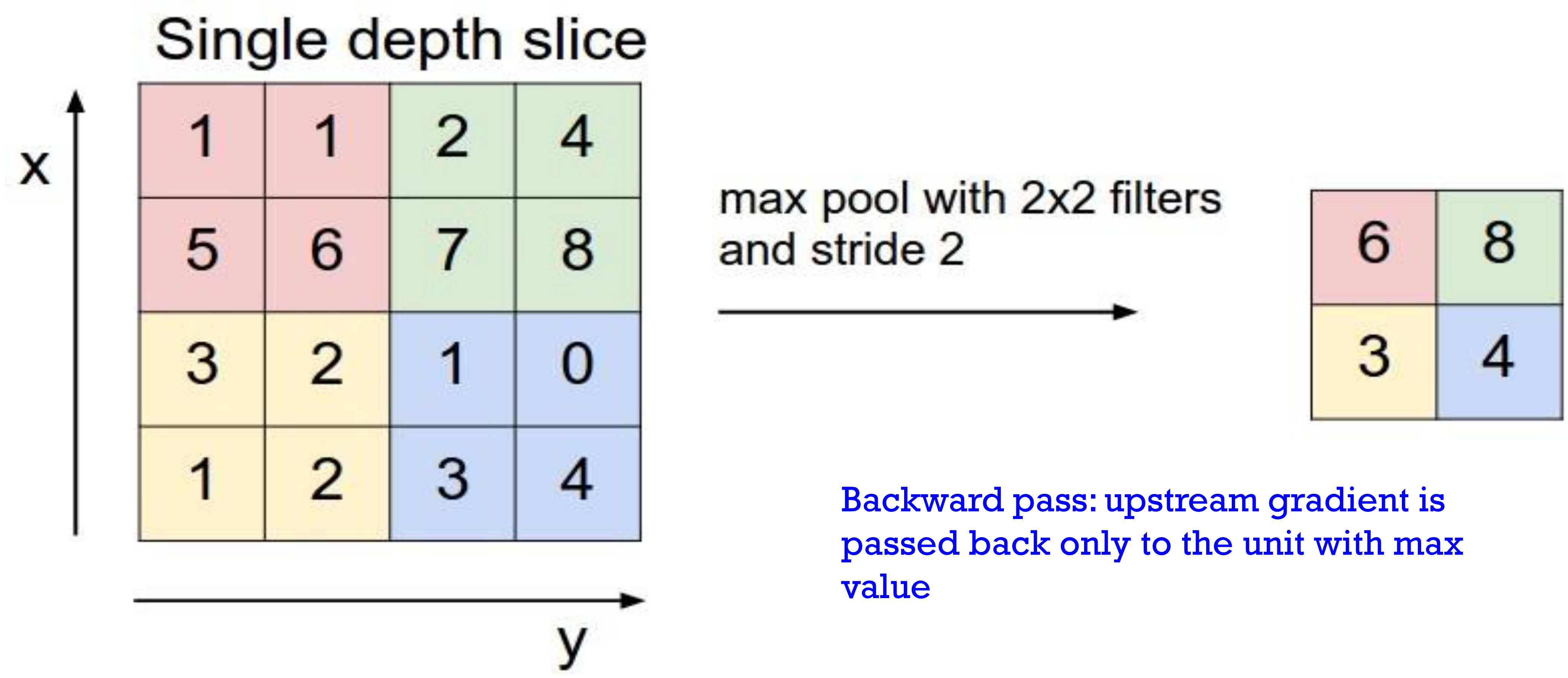
POOLING LAYER

- makes the representations smaller and more manageable
- operates over each activation map independently:



Source: cs231n, Stanford University

MAX POOLING



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

Source: cs231n, Stanford University

POOLING LAYER

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires two hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Source: cs231n, Stanford University

FULLY CONNECTED LAYER

- Connect every neuron in one layer to every neuron in another layer
- Same as the traditional multi-layer perceptron neural network

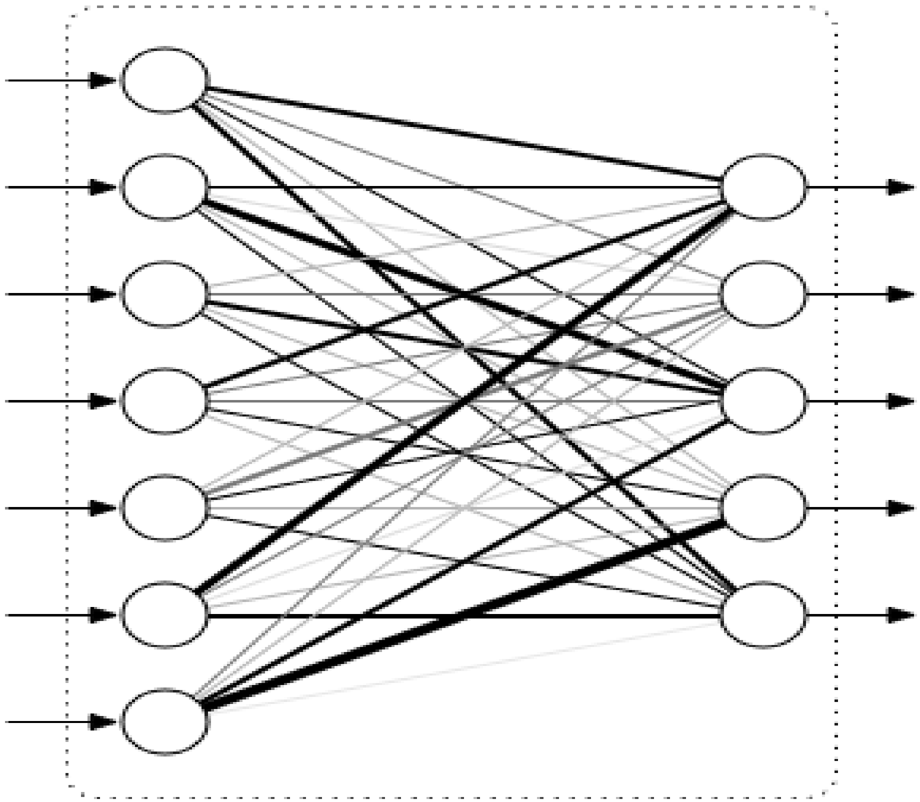


Image Source: machinethink.net

FULLY CONNECTED LAYER

- Connect every neuron in one layer to every neuron in another layer

- Same as the traditional multi-layer perceptron neural network

No. of Neurons (Last FC)
= No. of classes

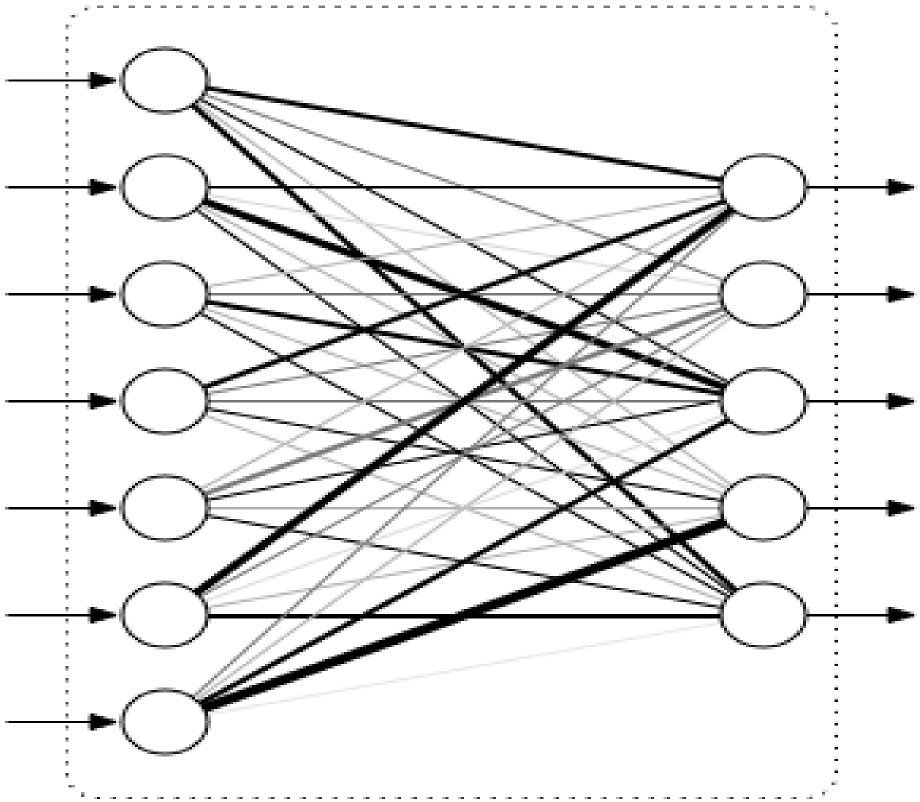


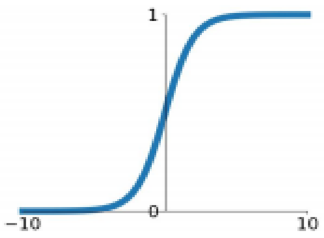
Image Source: machinethink.net

NON-LINEARITY LAYER

Activation Functions

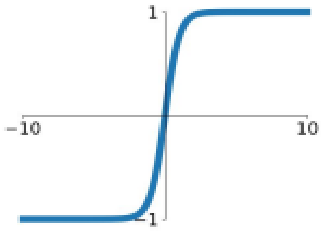
Sigmoid

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



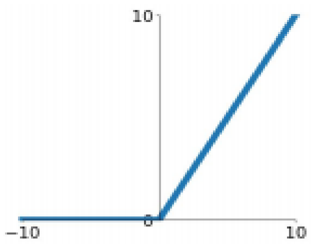
tanh

$\tanh(x)$



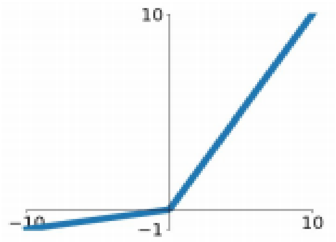
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

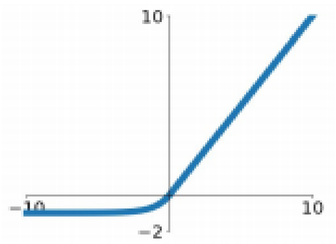


Maxout

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

ELU

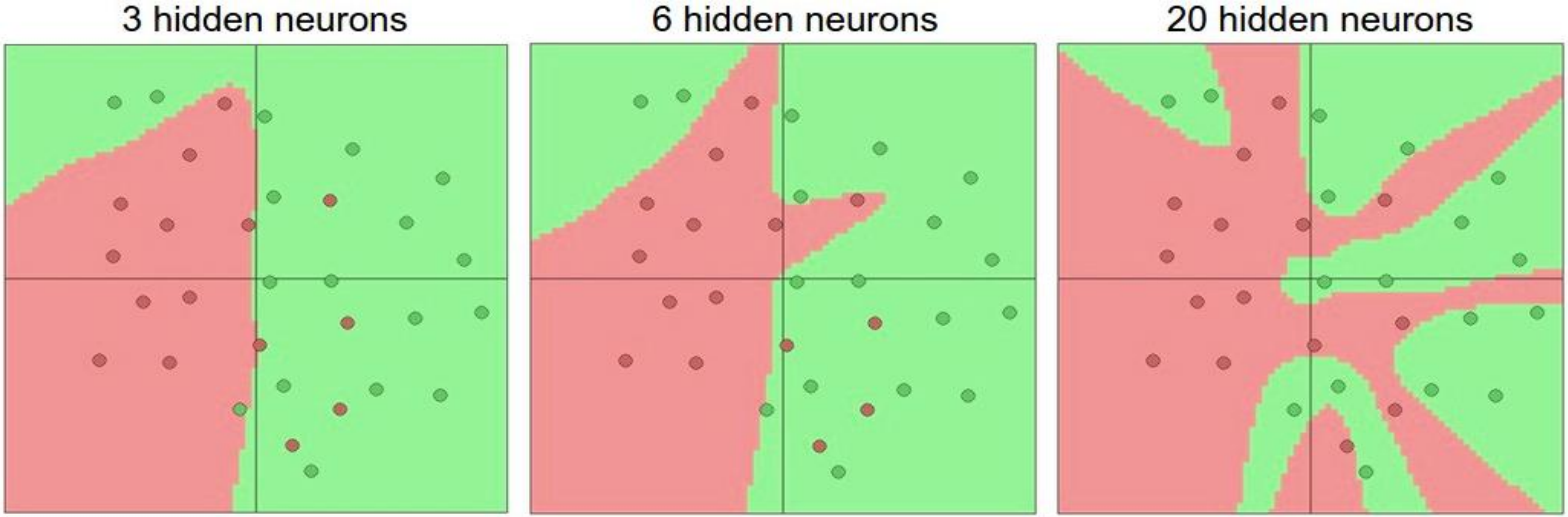
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Source: cs231n, Stanford University

Activation Functions

Non-linearities needed to learn complex (non-linear) representations of data, otherwise the NN would be just a linear function



More layers and neurons can approximate more complex functions

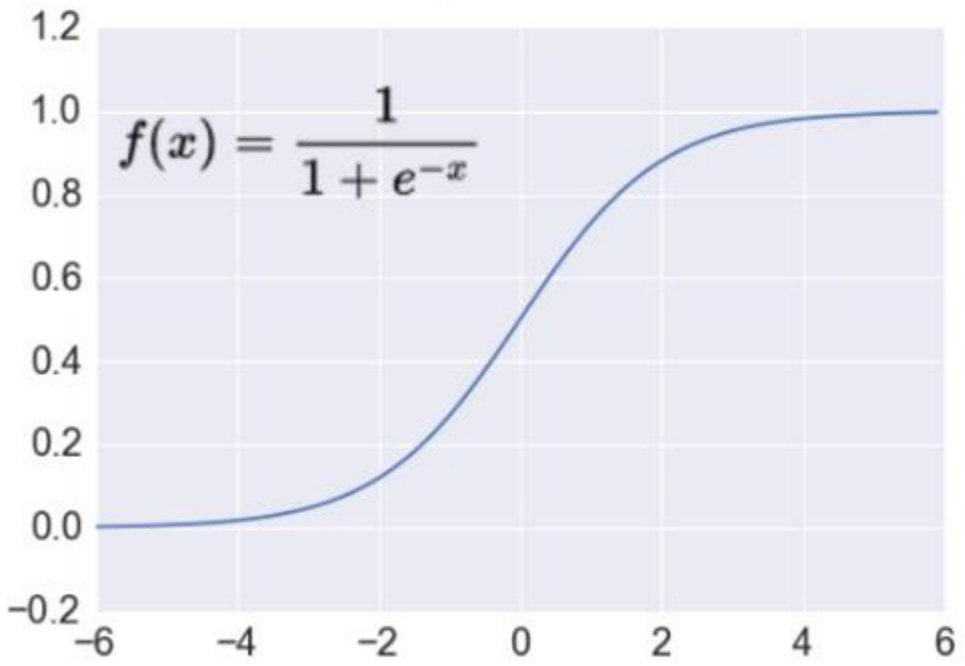
http://cs231n.github.io/assets/nn1/layer_sizes.jpeg

Full list: https://en.wikipedia.org/wiki/Activation_function

Activation Function: Sigmoid

Takes a real-valued number and "squashes" it into range between 0 and 1.

$$R^n \rightarrow [0,1]$$



\+ Nice interpretation as the **firing rate** of a neuron

- 0 = not firing at all
- 1 = fully firing

\- Sigmoid neurons **saturate** and **kill gradients**, thus NN will barely learn

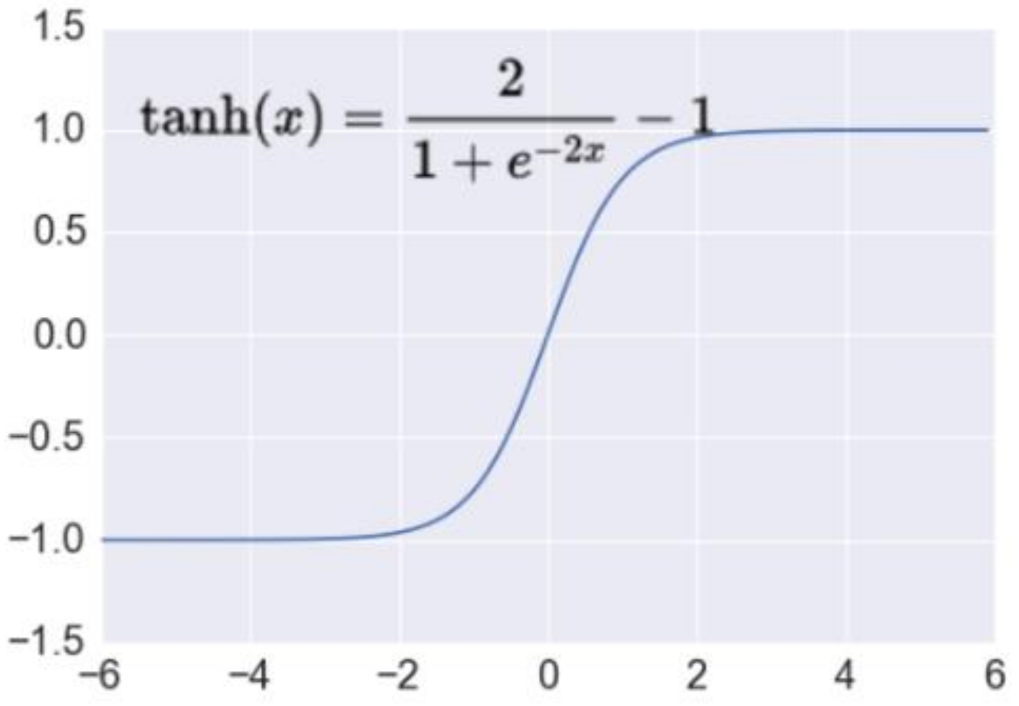
- when the neuron's activation are 0 or 1 (saturate)
  - 🙁 gradient at these regions almost zero
  - 🙁 almost no signal will flow to its weights
  - 🙁 if initial weights are too large then most neurons would saturate

http://adilmoujahid.com/images/activation.png

Activation Function: Tanh

Takes a real-valued number and "squashes" it into range between -1 and 1.

$$R^n \rightarrow [-1,1]$$



- Like sigmoid, tanh neurons **saturate**
- Unlike sigmoid, output is **zero-centered**
- Tanh is a **scaled sigmoid**: $\tanh(x) = 2sigm(2x) - 1$
- Drawbacks of Sigmoid with Tanh also.

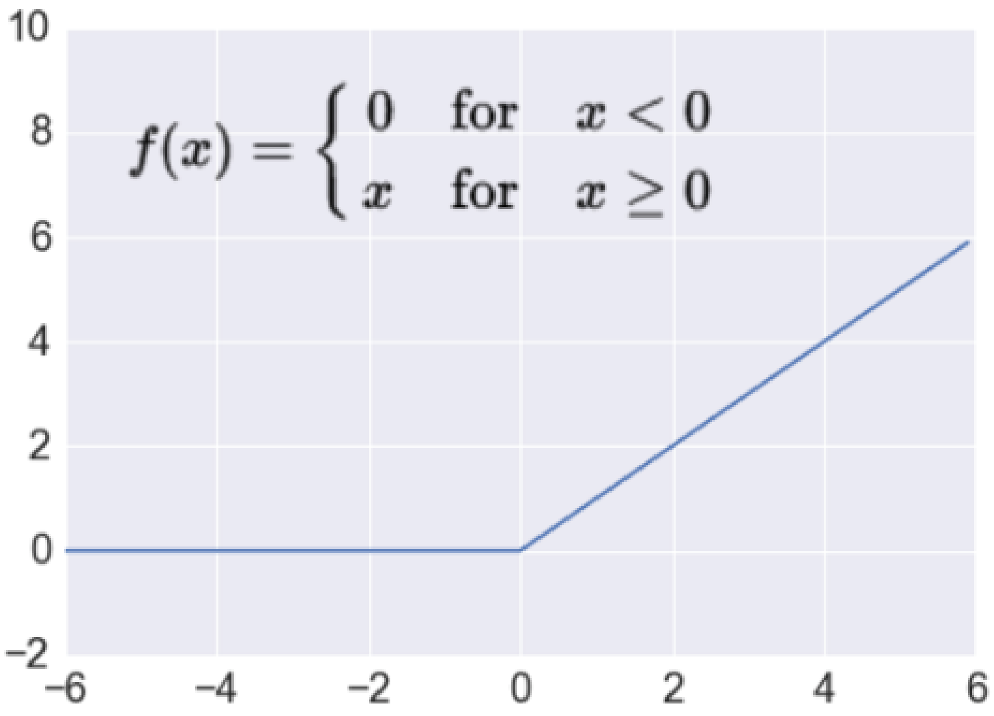
http://adilmoujahid.com/images/activation.png

Activation Function: ReLU

Takes a real-valued number and thresholds it at zero

$$f(x) = \max(0, x)$$

$$R^n \rightarrow R^n_+$$



Most Deep Networks use ReLU nowadays

🙂Trains much **faster**

- accelerates the convergence of SGD
- due to linear, non-saturating form

🙂Less expensive operations

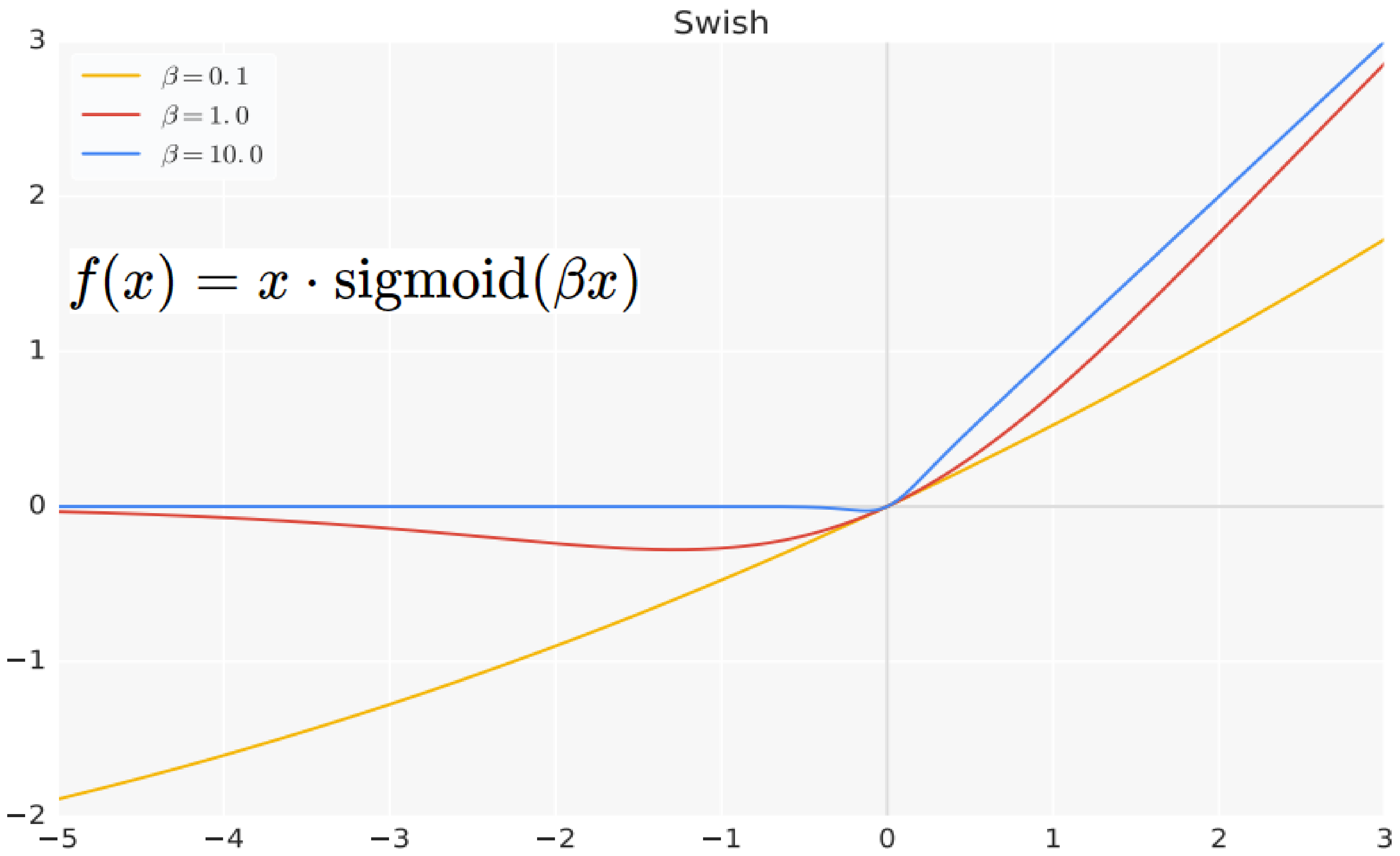
- compared to sigmoid/tanh (exponentials etc.)
- implemented by simply thresholding a matrix at zero

🙂More **expressive**

🙂Prevents the **gradient vanishing problem** for +ive inputs

http://adilmoujahid.com/images/activation.png

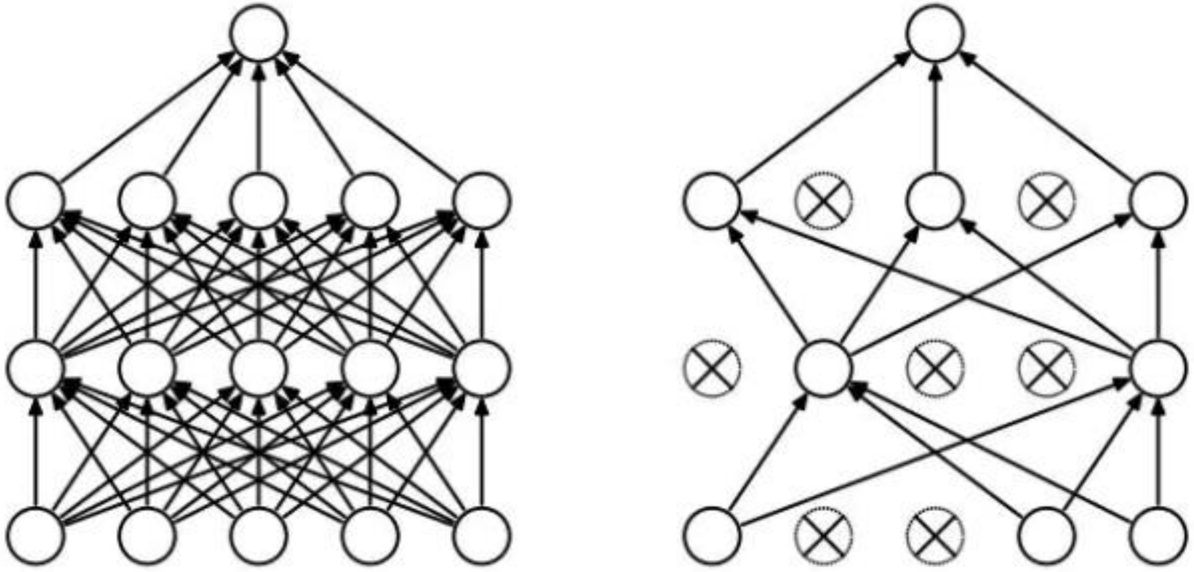
Activation Function: Swish



- ReLU is special case of Swish

Ramachandran et al. "Swish: a self-gated activation function." *ICLR Workshops*, 2018.

Regularization



Dropout

- Randomly drop units (along with their connections) during training
- Each unit retained with fixed probability p, independent of other units
- Hyper-parameter p to be chosen (tuned)

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research (2014)

Batch Normalization

"We want zero-mean unit-variance activations? lets make them so."

consider a batch of activations at some layer. To make each dimension zero-mean unit-variance, apply:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

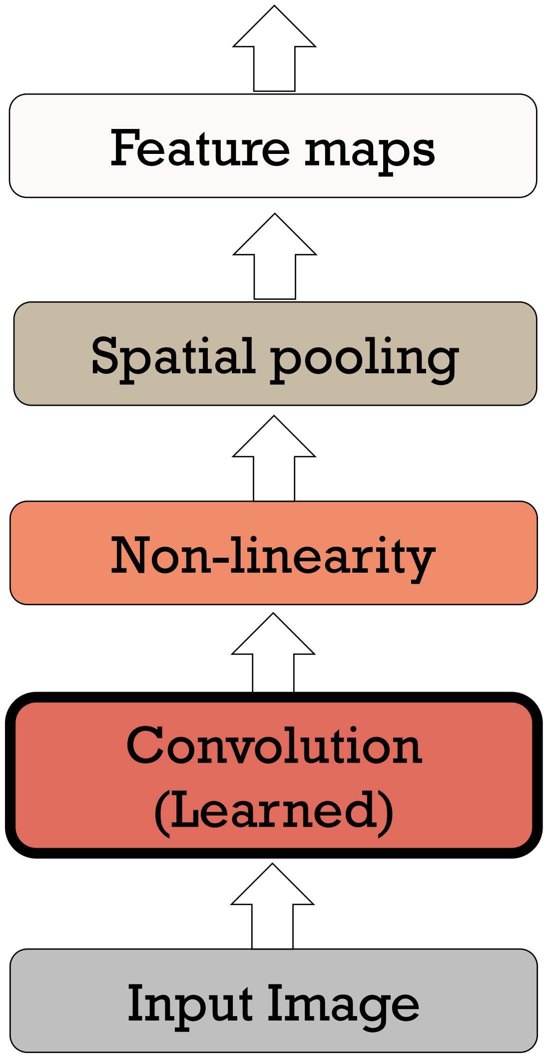
Source: cs231n

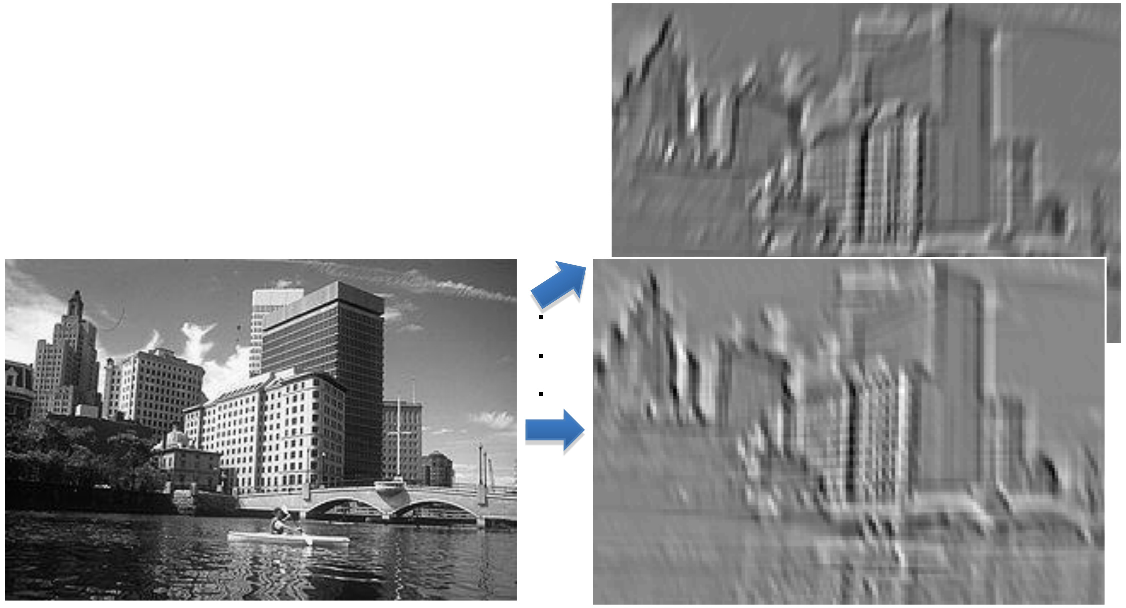
CLASSIFICATION/LOSS LAYER

- SVM Classifier
  - SVM Loss/Hinge Loss/Max-margin Loss

- Softmax Classifier
  - Softmax Loss/Cross-entropy Loss

SUMMARY: CNN PIPELINE



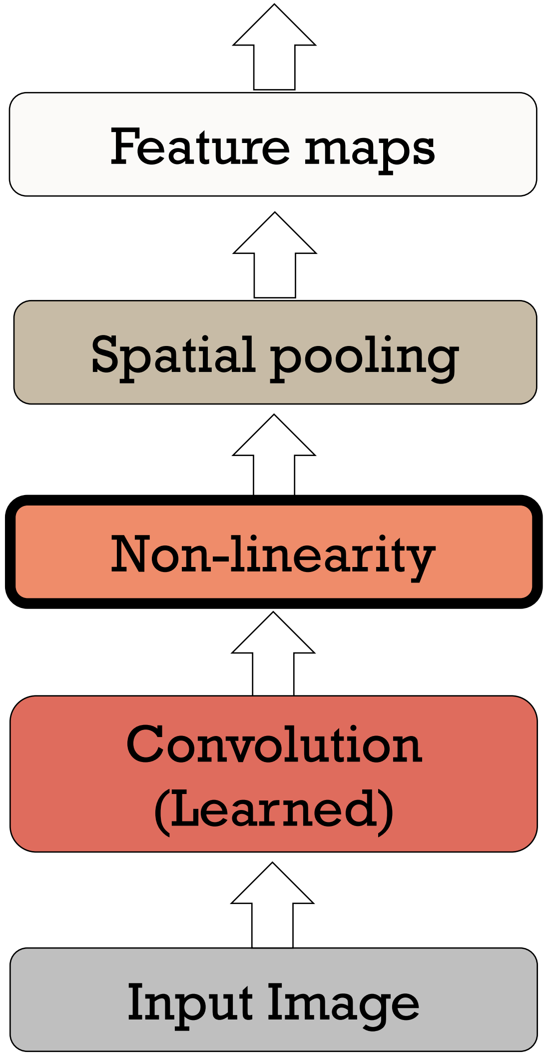


Input

Feature Map

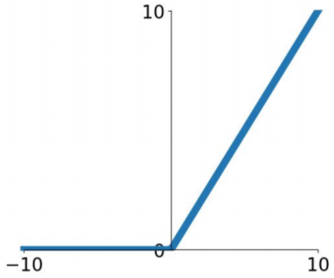
Source: R. Fergus, Y. LeCun

SUMMARY: CNN PIPELINE



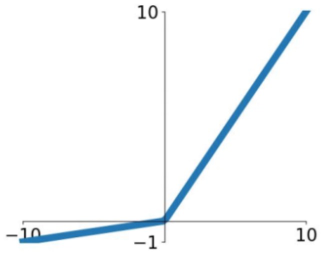
ReLU

$$\max(0, x)$$



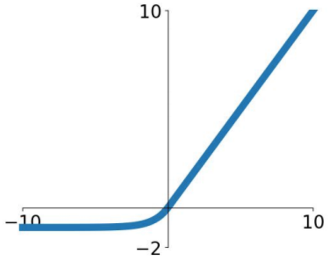
Leaky ReLU

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



ELU

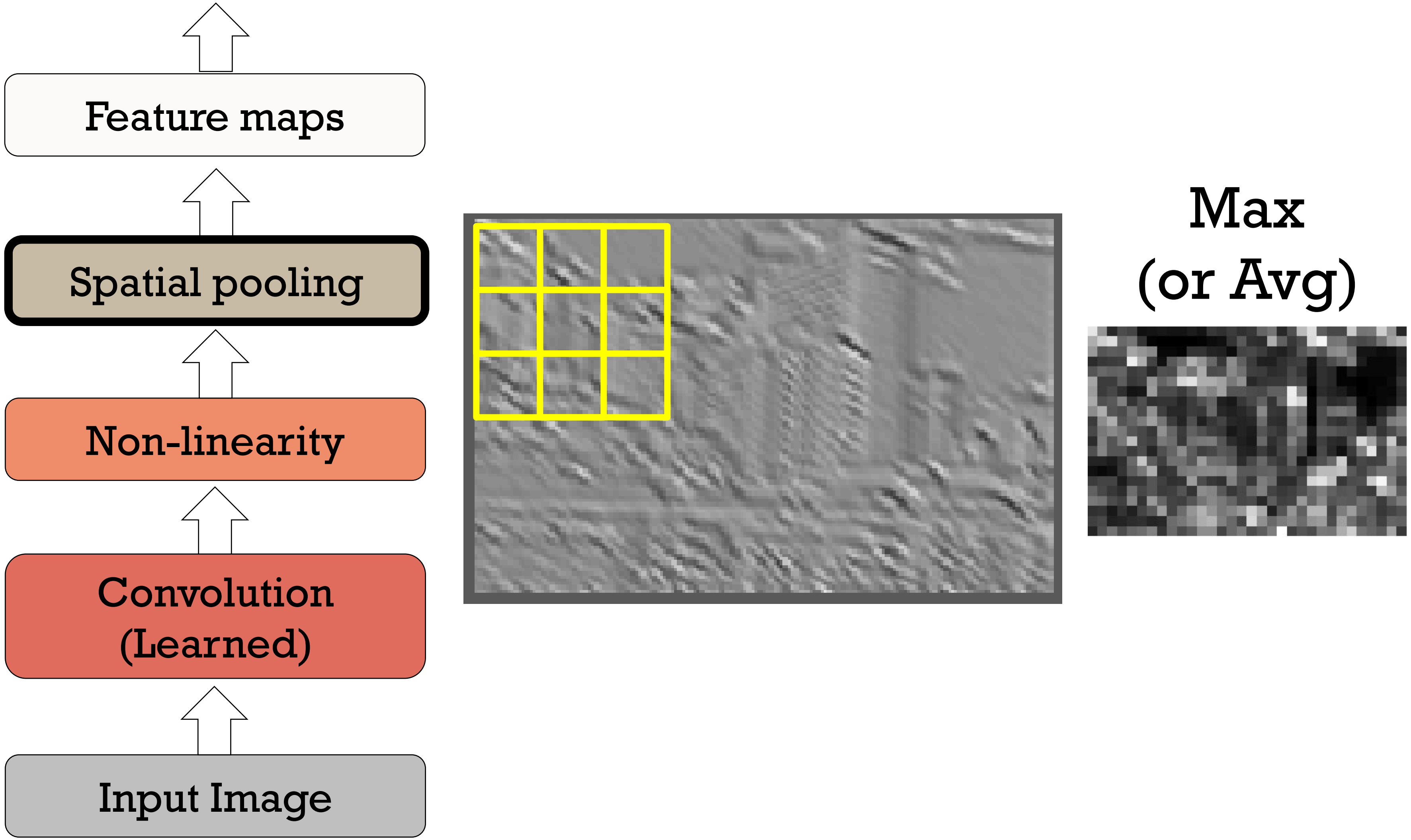
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Source: R. Fergus, Y. LeCun

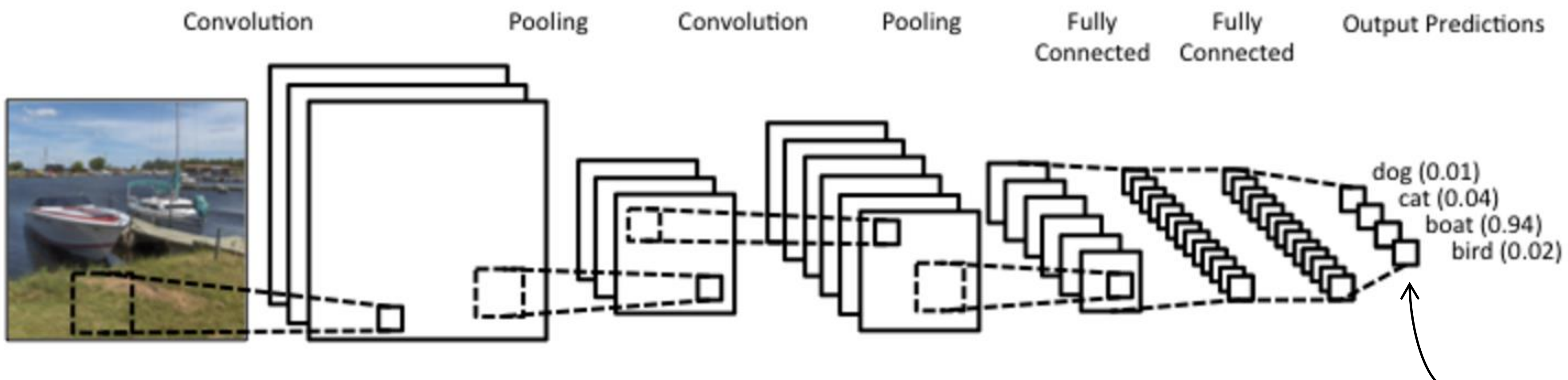
Source: Stanford 231n

SUMMARY: CNN PIPELINE

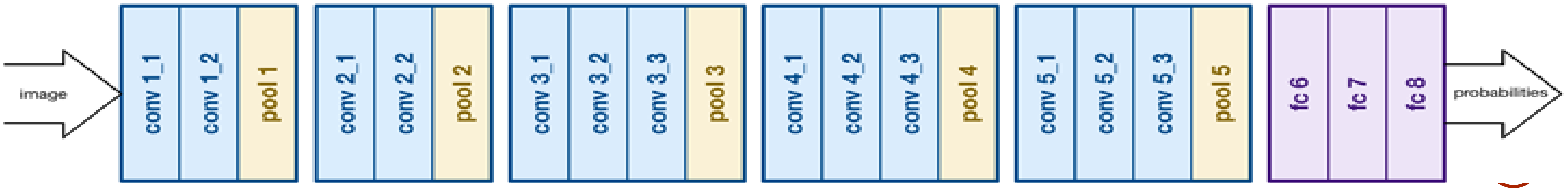


Source: R. Fergus, Y. LeCun

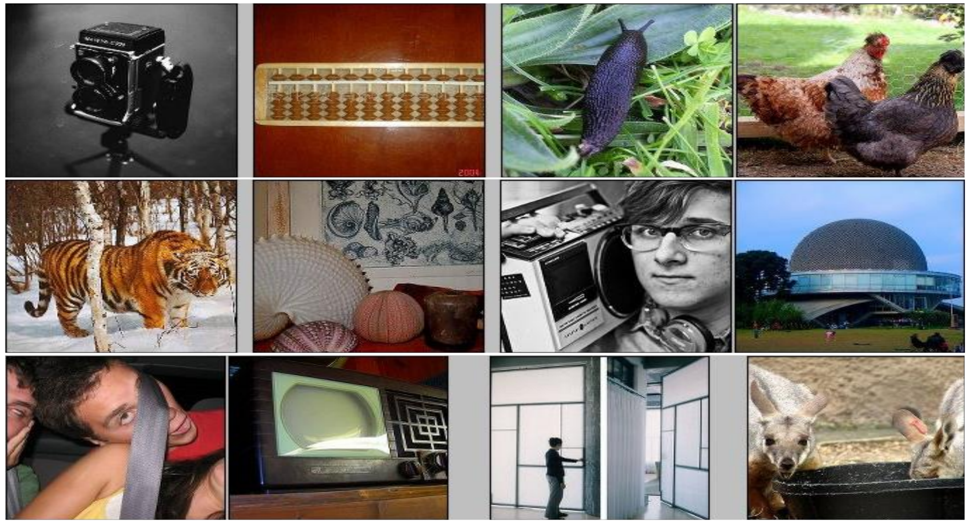
SUMMARY: CNN PIPELINE



Softmax layer:



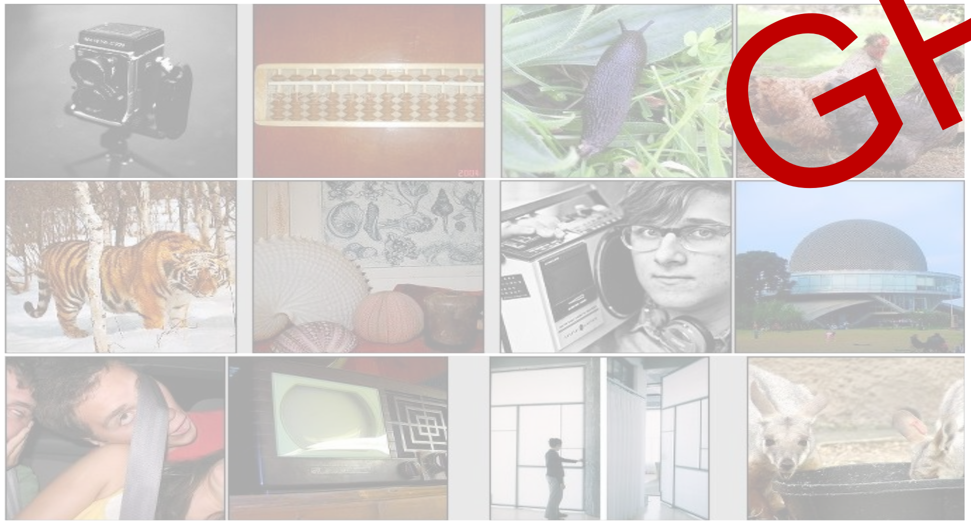
IMAGENET CHALLENGE



- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/

IMAGENET CHALLENGE

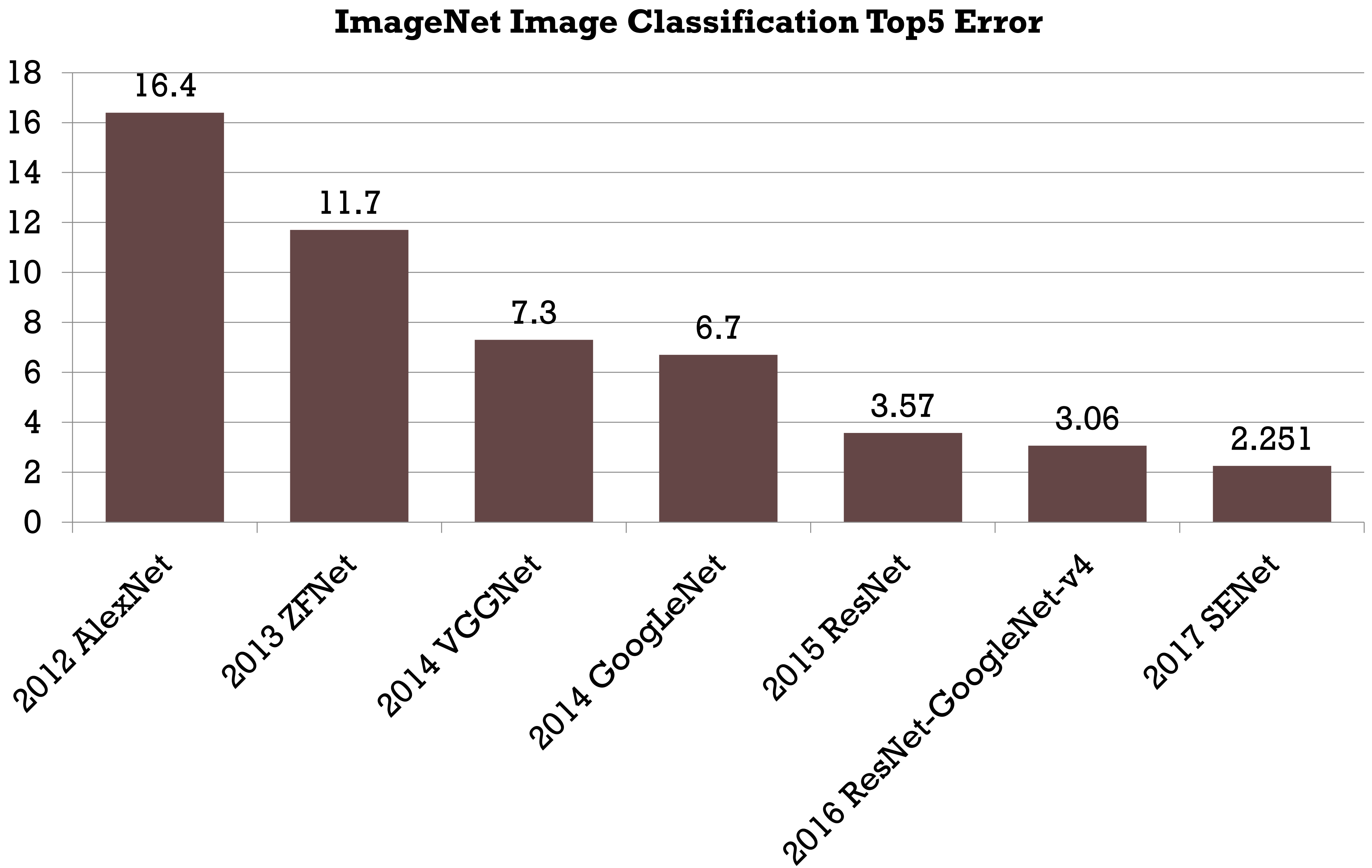


- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

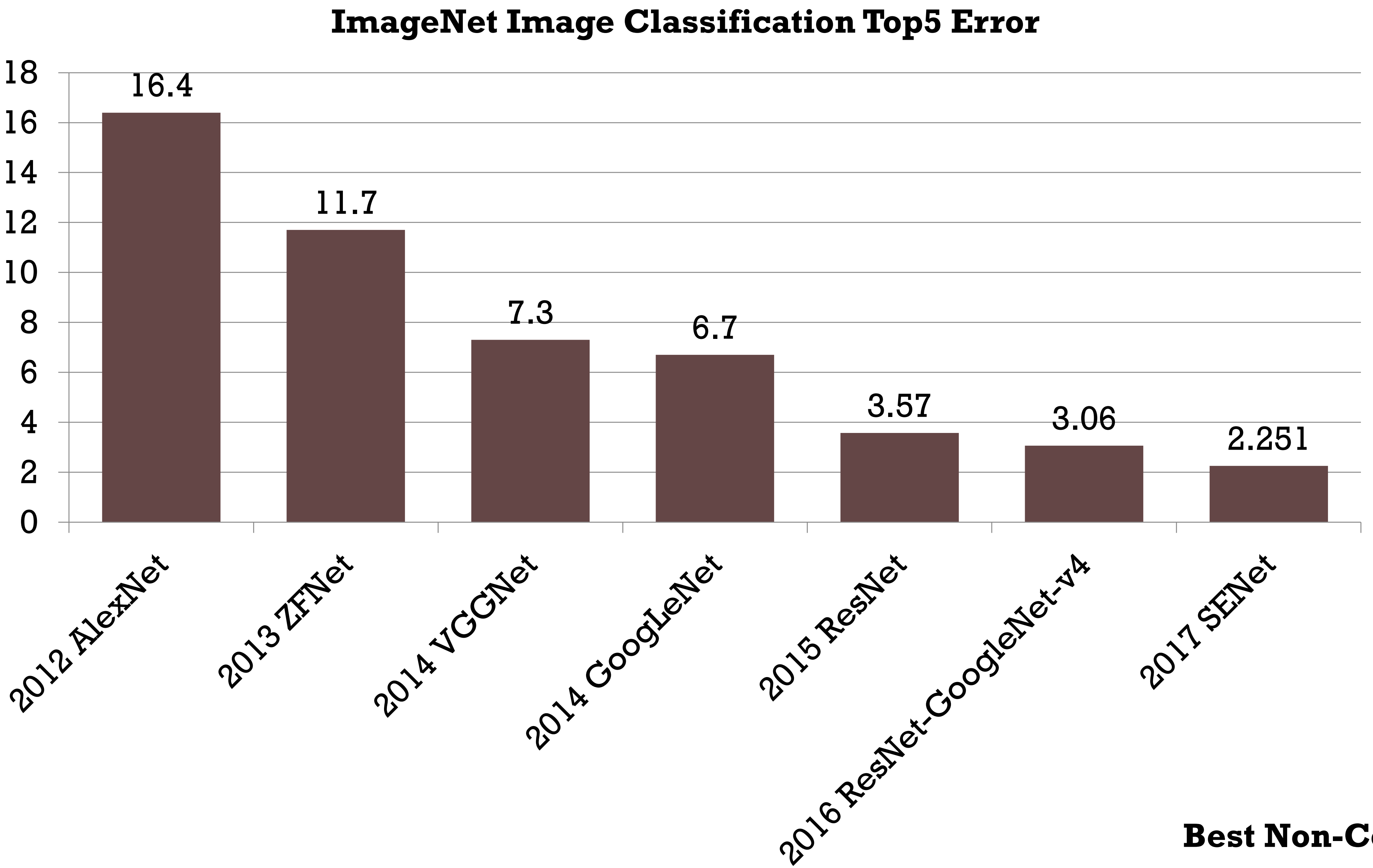
GPUs + Data

www.image-net.org/challenges/LSVRC/

PROGRESS ON IMAGENET CHALLENGE



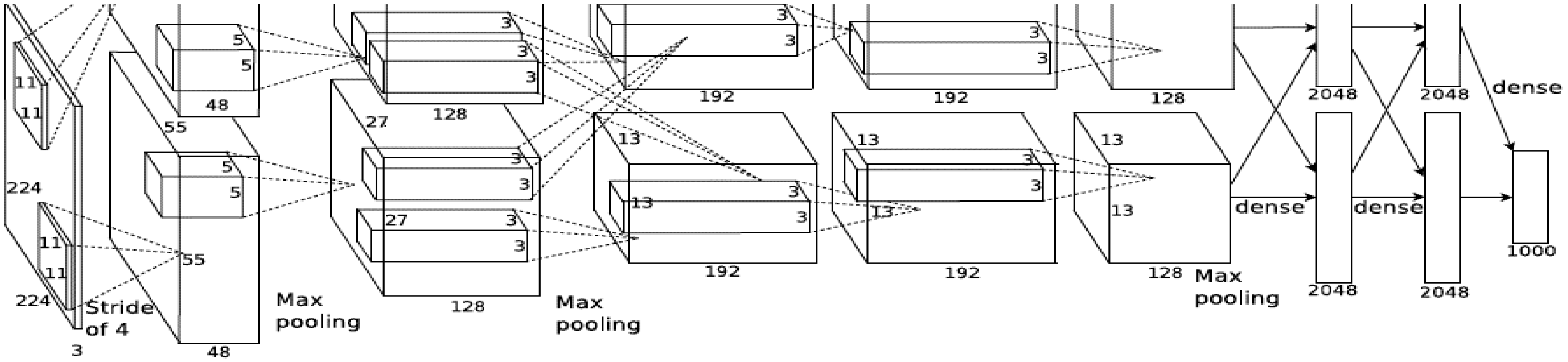
PROGRESS ON IMAGENET CHALLENGE



Best Non-ConvNet in 2012: 26.2%

CNN ARCHITECTURES FOR CLASSIFICATION

ALEXNET



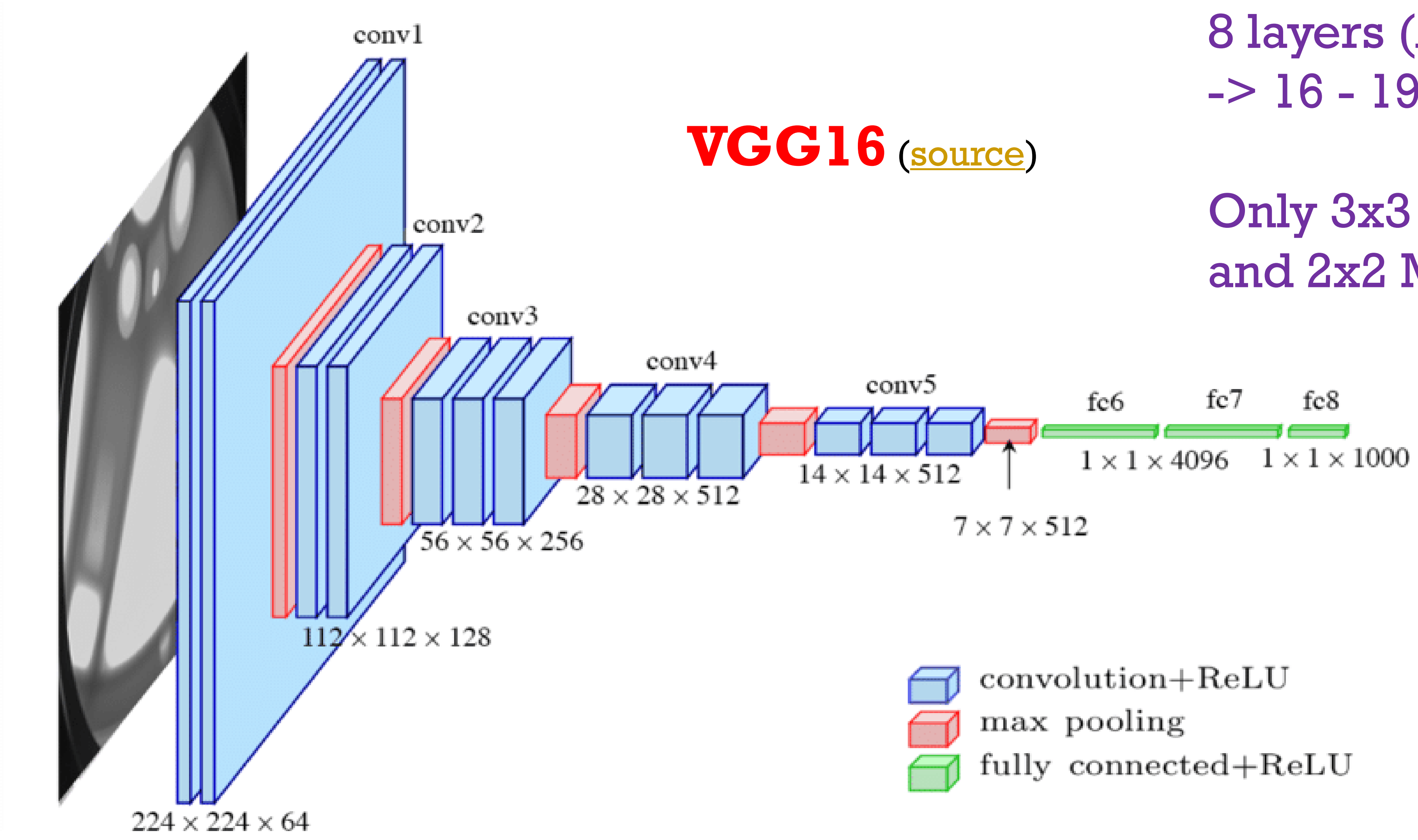
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256]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

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- batch size 128
- SGD Momentum 0.9
- Learning rate 0.01, reduced manually when val accuracy saturates

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

VGGNET



VGG16 (source)

Small filters, Deeper networks

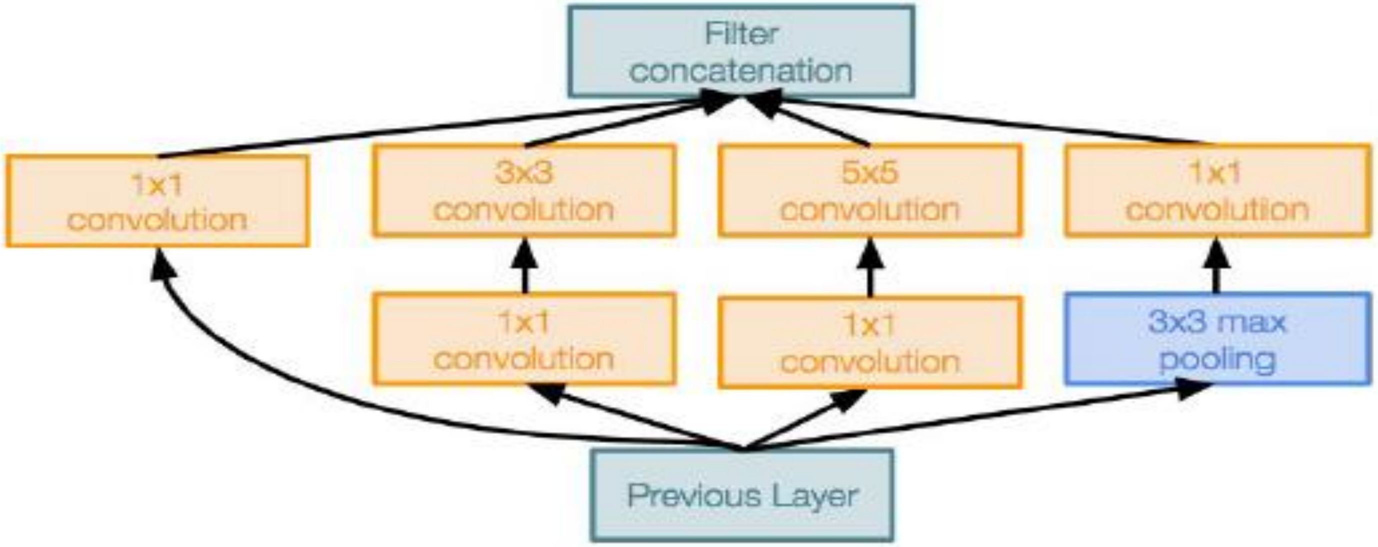
8 layers (AlexNet)
-> 16 - 19 layers (VGGNet)

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

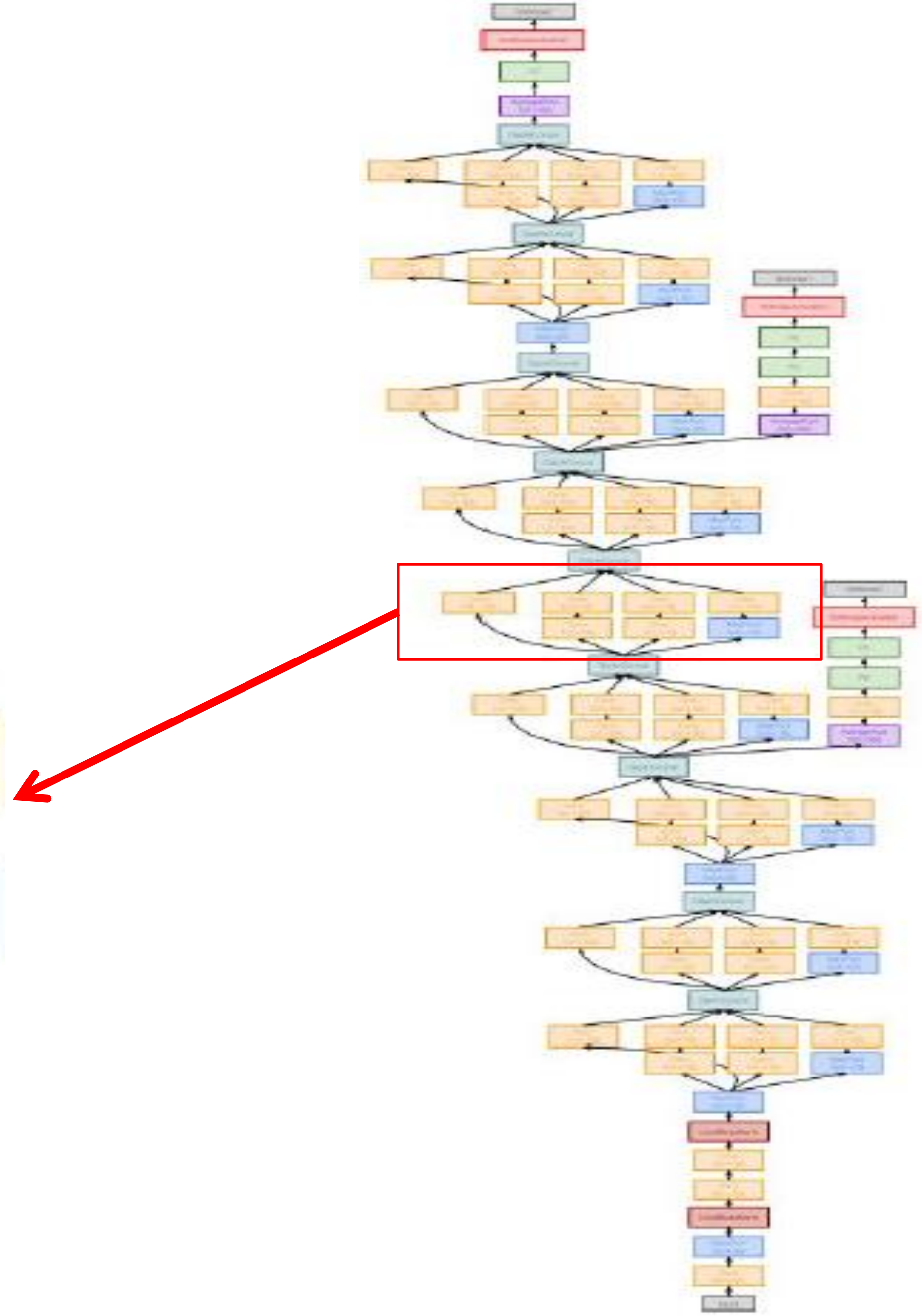
Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

GOOGLENET

“Inception module”:
design a good local network topology and then stack these modules on top of each other



Inception module



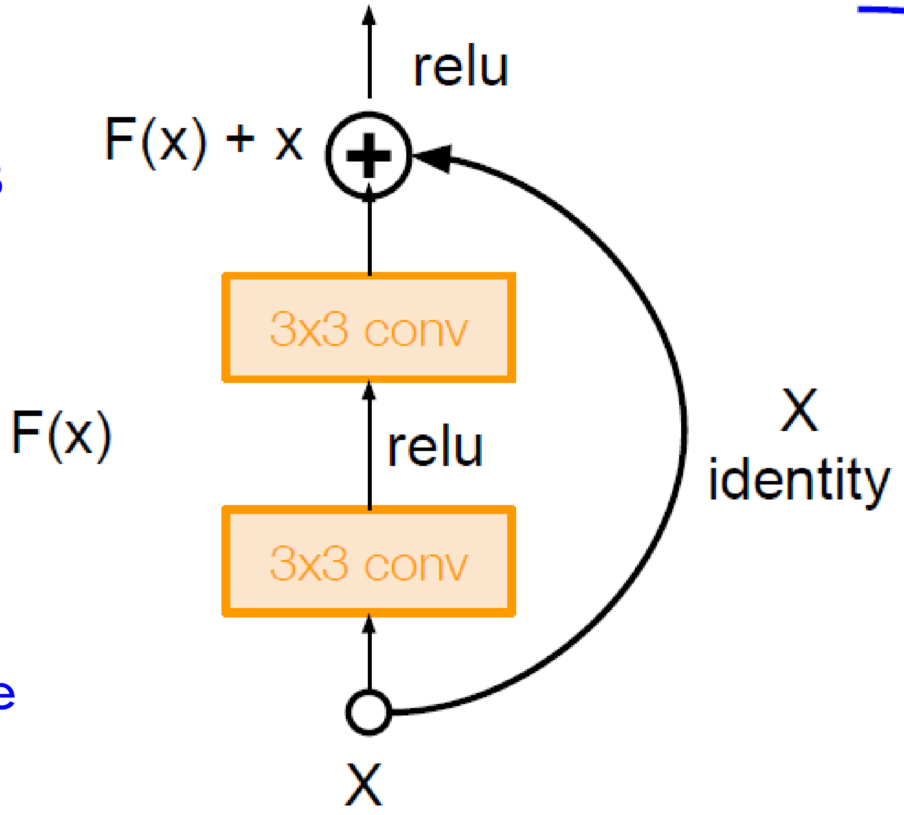
Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

Source: cs231n

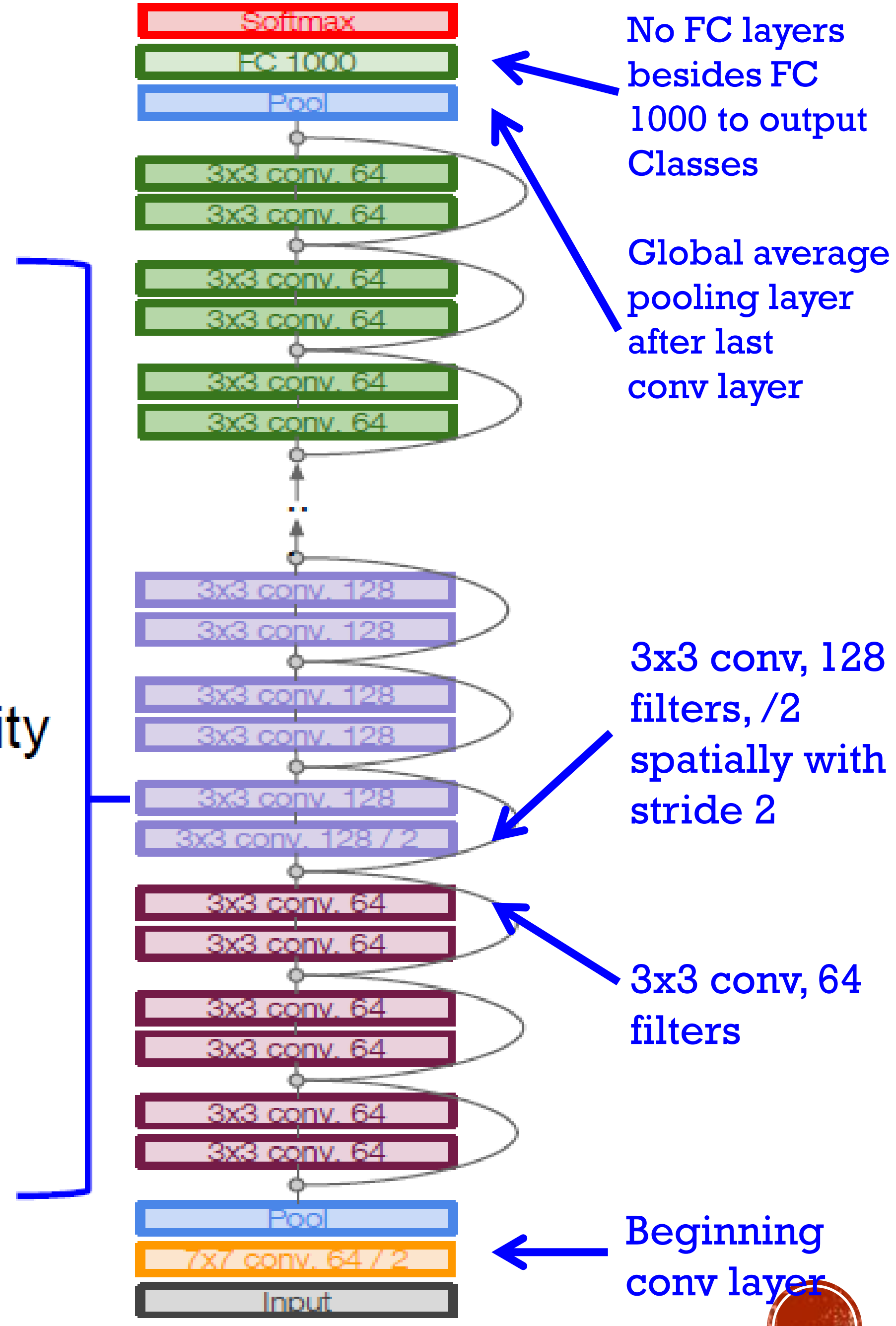
RESNET

Full ResNet architecture:

- Stack residual blocks
- Residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)



Residual block



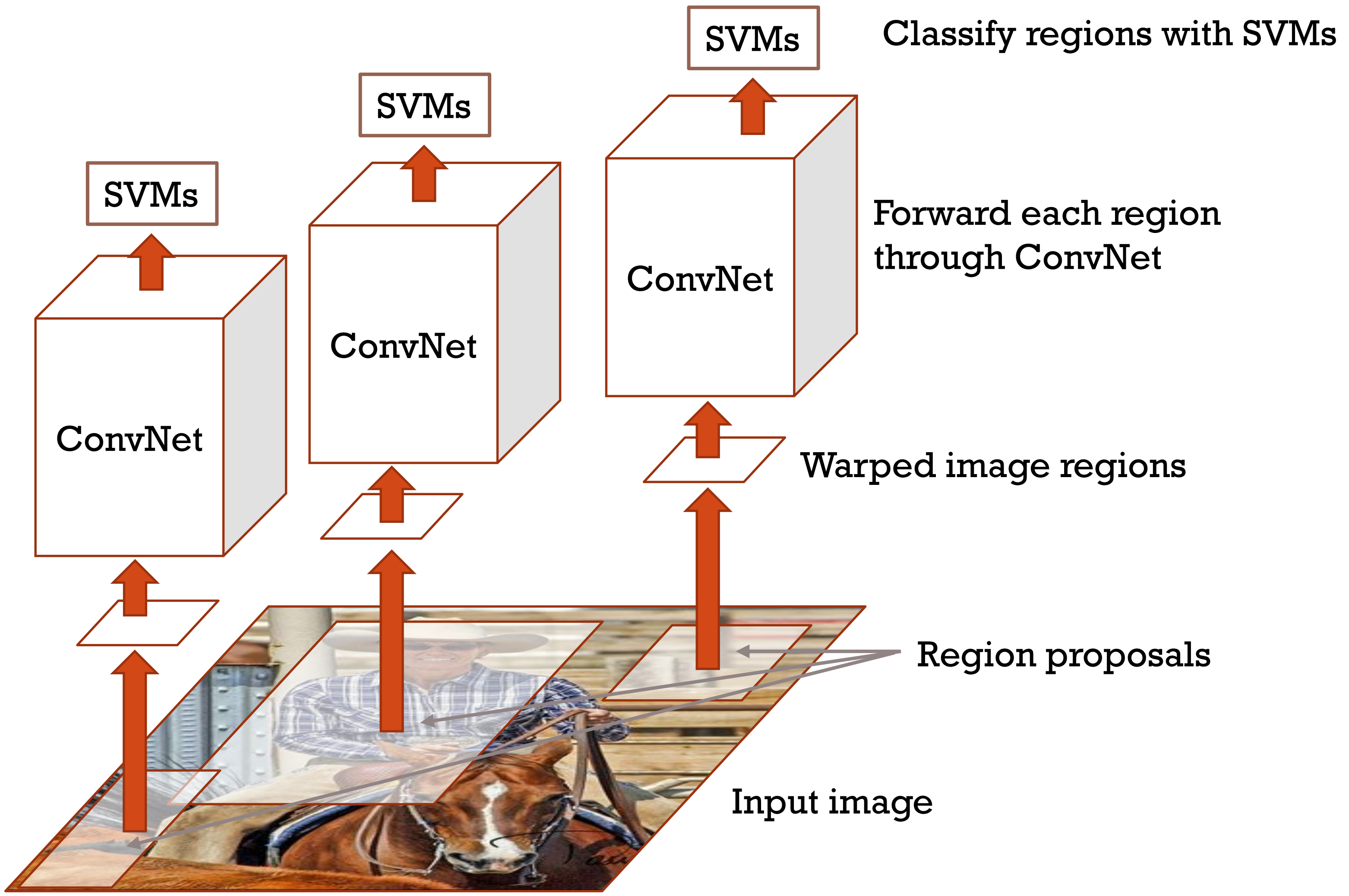
He et al. [Deep Residual Learning for Image Recognition](), IEEE CVPR 2016.

Source: cs231n

CNN ARCHITECTURES FOR OBJECT RECOGNITION

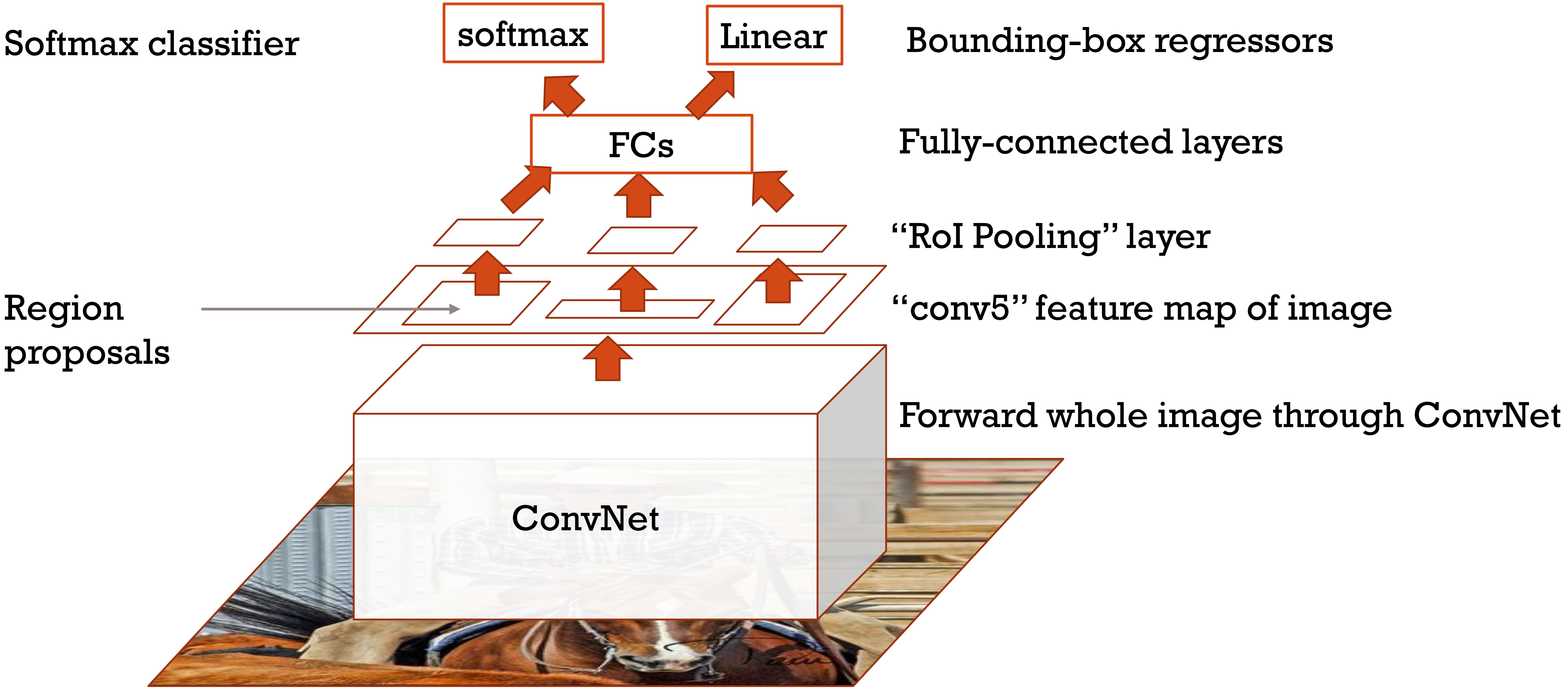
R-CNN: REGION PROPOSALS + CNN FEATURES

Source: R. Girshick



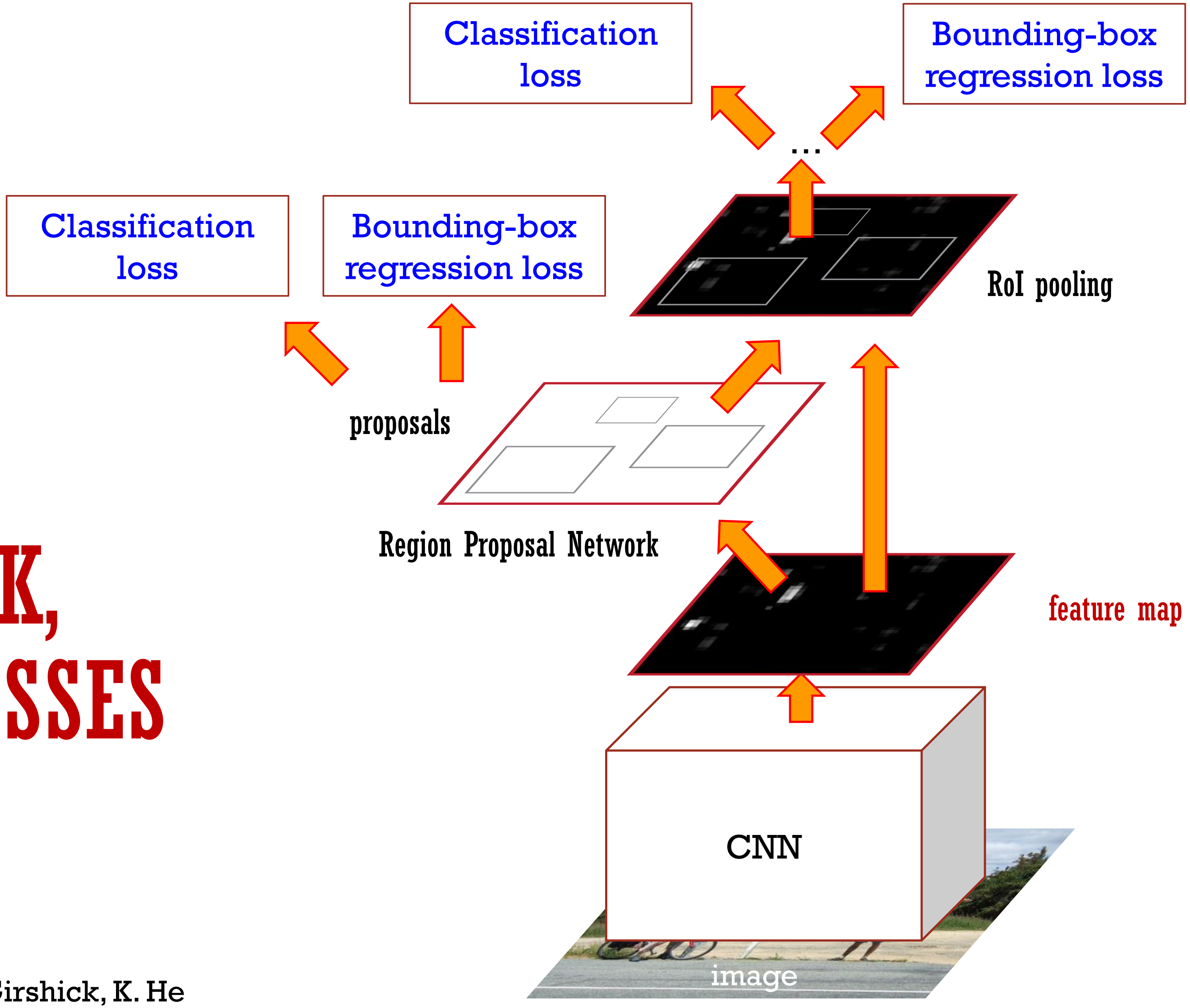
R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014

FAST R-CNN (GIRSHICK ICCV 2015)



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

FASTER R-CNN (REN ET AL. NIPS 2015)



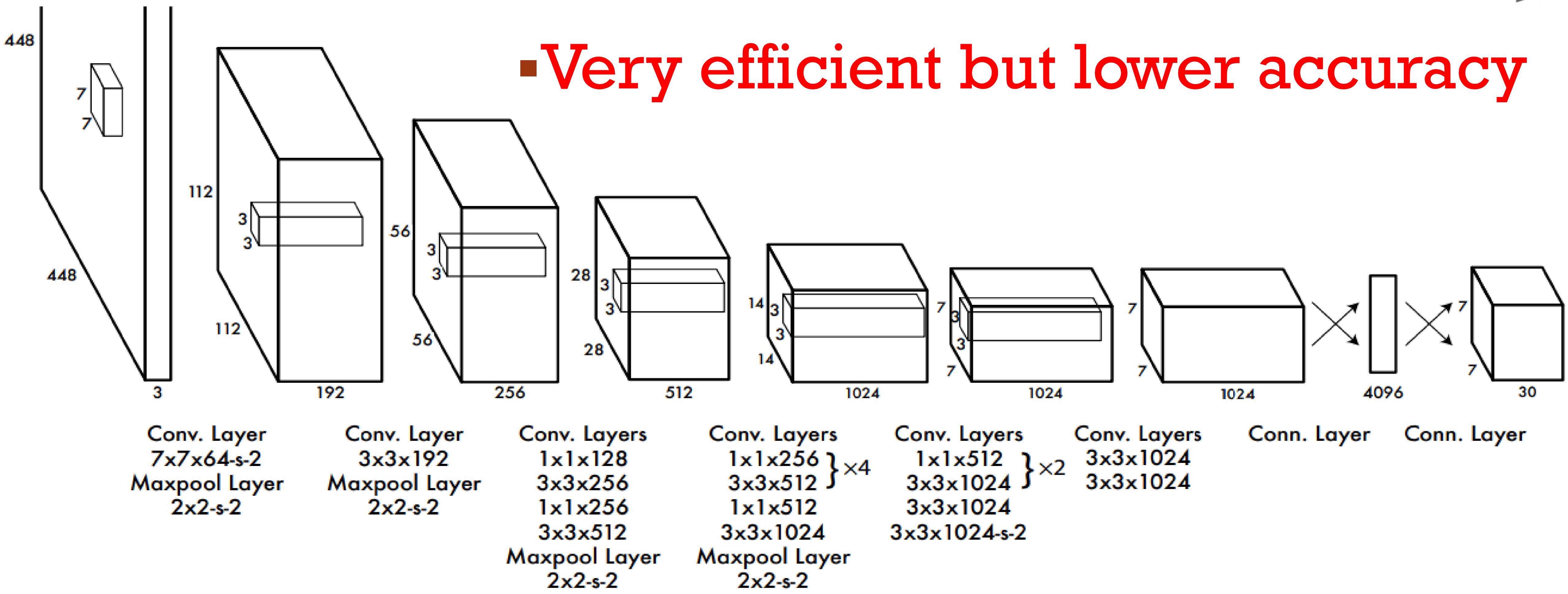
ONE NETWORK, FOUR LOSSES

Source: R. Girshick, K. He

YOU ONLY LOOK ONCE (YOLO)

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

- Very efficient but lower accuracy



Redmon et al. You only look once: Unified, real-time object detection. CVPR 2016.

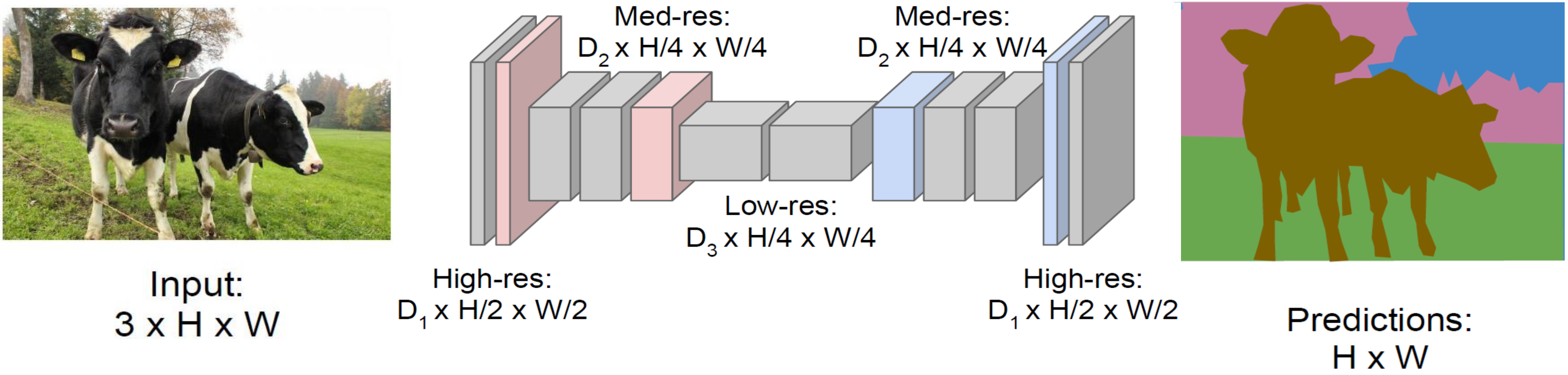
CNN ARCHITECTURES FOR SEGMENTATION

SEMANTIC SEGMENTATION: FULLY CONVOLUTIONAL

Downsampling:
Pooling, strided convolution

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

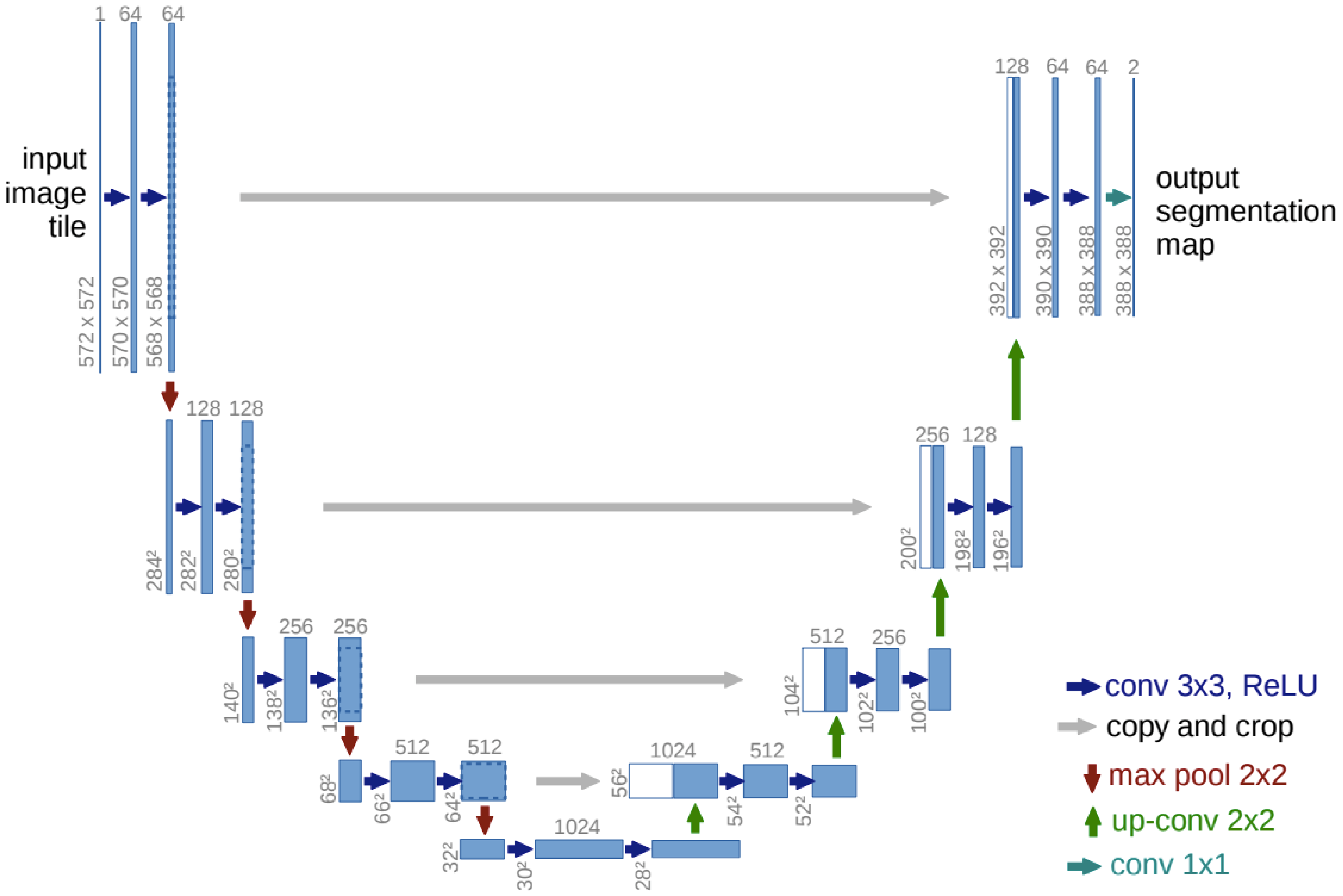
Upsampling:
Unpooling or strided transpose convolution



Long et al. Fully Convolutional Networks for Semantic Segmentation, CVPR 2015
Noh et al. Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

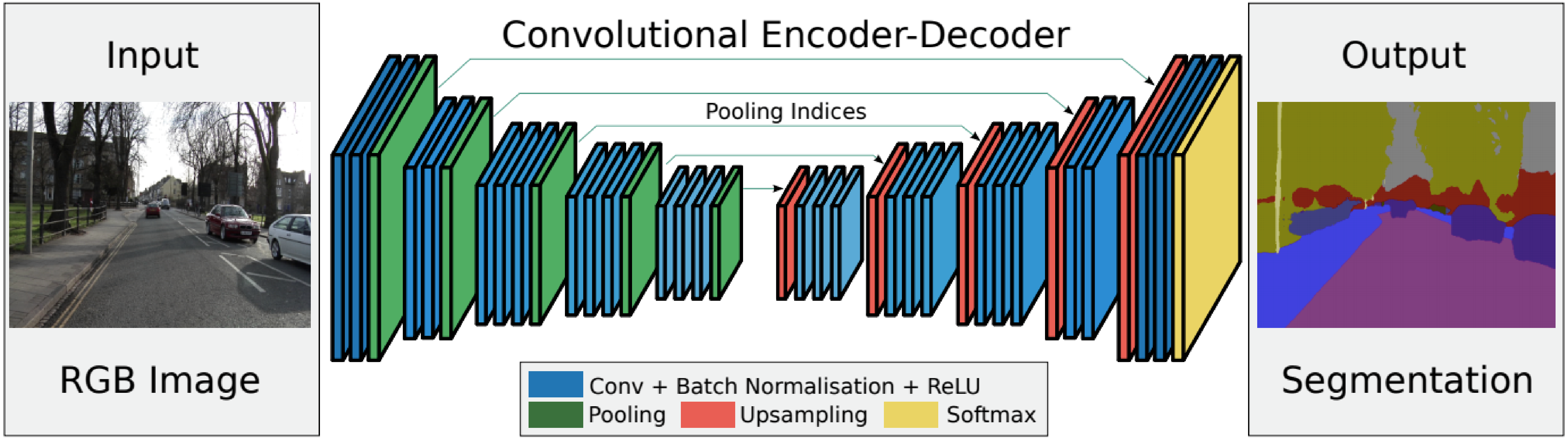
Slide credit: cs231n

U-NET



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

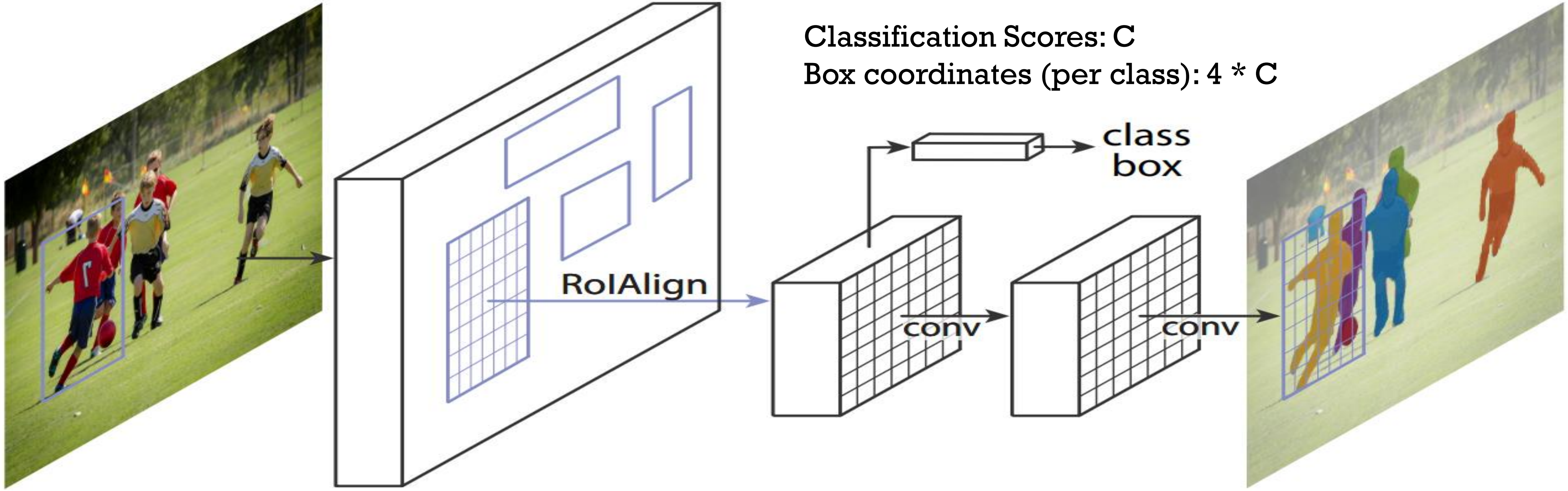
SEGNET



Drop the FC layers,
get better results

V. Badrinarayanan, A. Kendall and R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, PAMI 2017

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



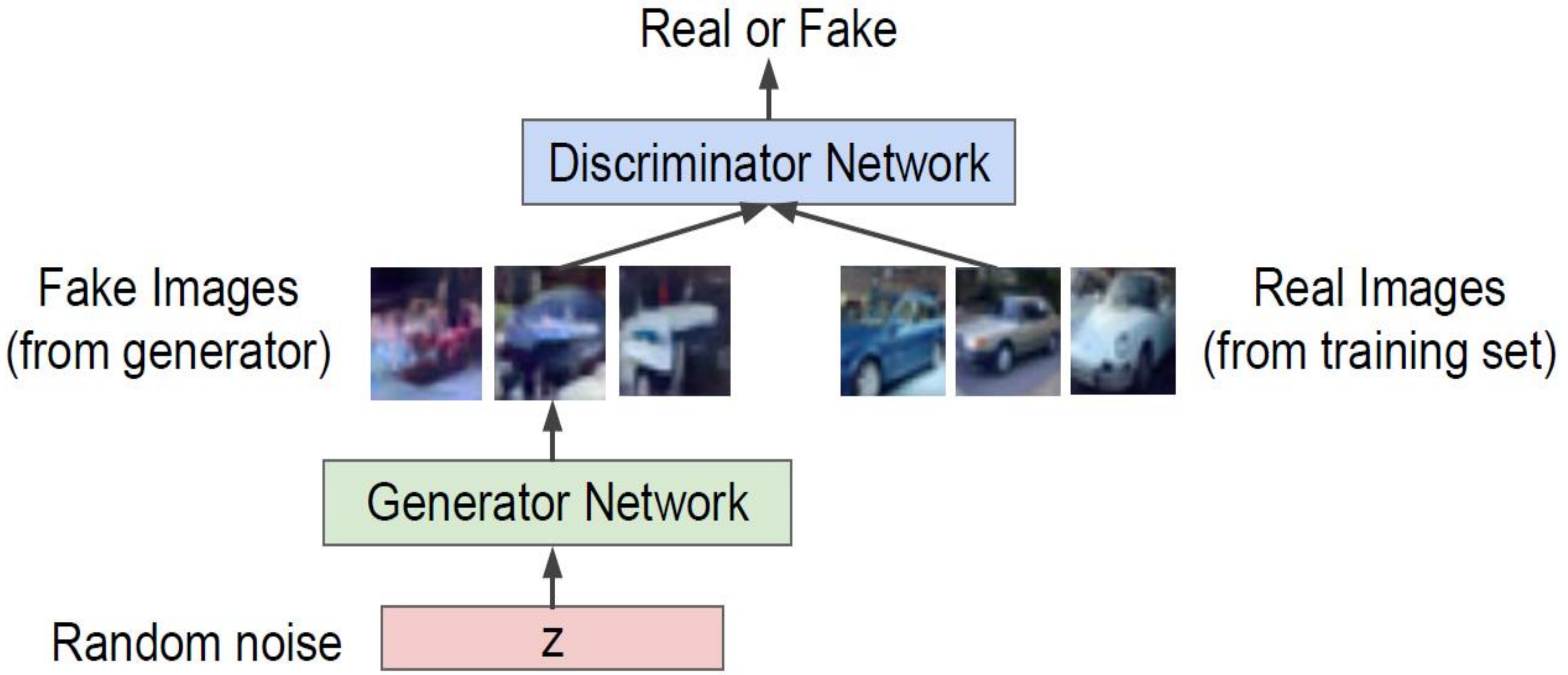
Mask R-CNN - extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

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

He et al. Mask R-CNN, ICCV 2017. https://arxiv.org/pdf/1703.06870.pdf

CNN ARCHITECTURES FOR GENERATIVE TASKS

GENERATIVE ADVERSARIAL NETWORKS



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Fake and real images copyright Emily Denton et al. 2015.

Credit: cs231n, Stanford

IMAGE-TO-IMAGE TRANSLATION: PIX2PIX

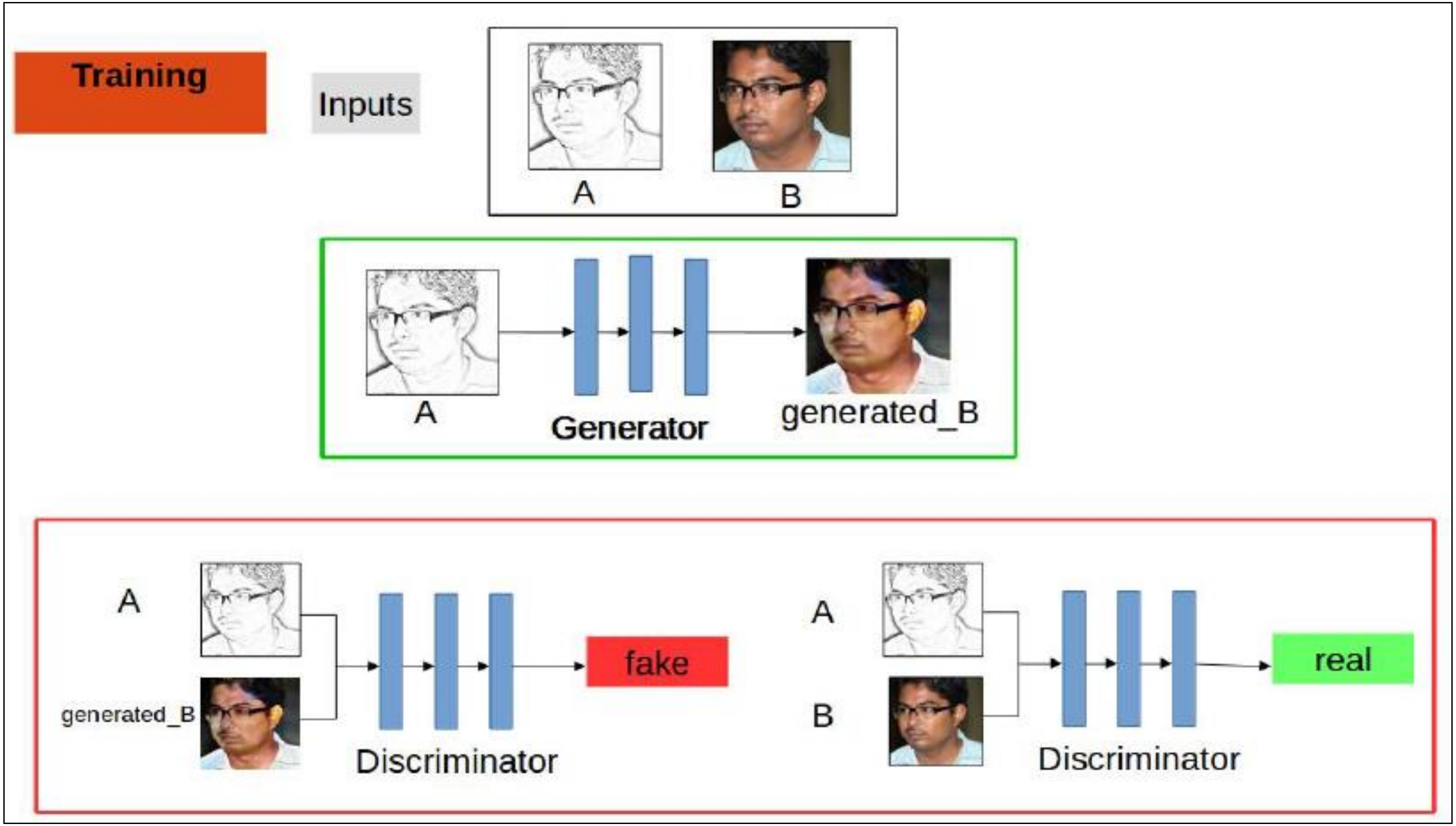
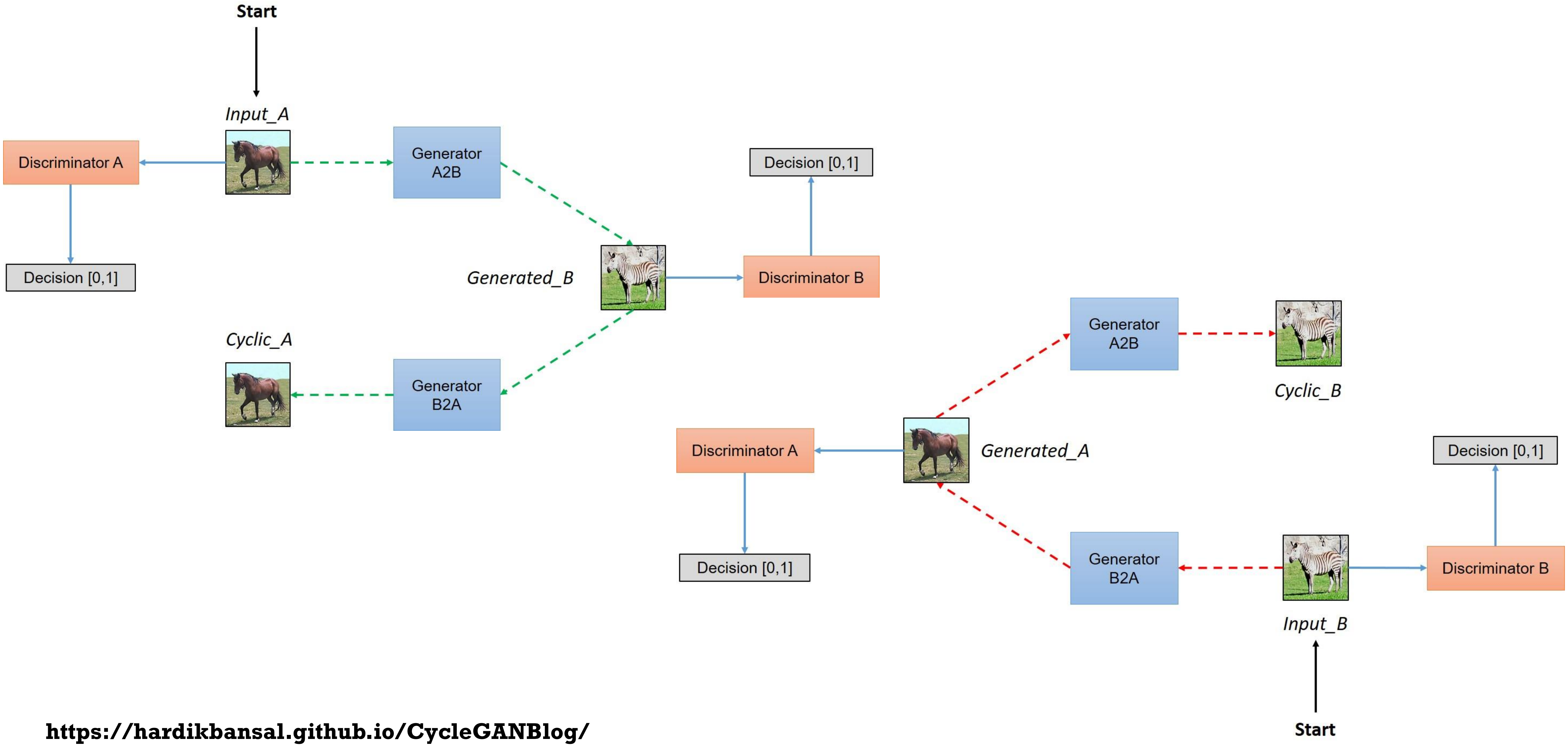


IMAGE-TO-IMAGE TRANSLATION: CYCLE GAN



https://hardikbansal.github.io/CycleGANBlog/

ACKNOWLEDGEMENT

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

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

THANK YOU

QUESTIONS?

Image Source: http://pluspng.com/best-of-luck-png-5072.html