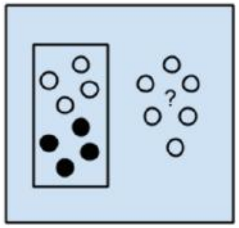
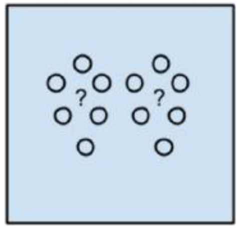
Deep Learning for Computer Vision

Lex Fridman

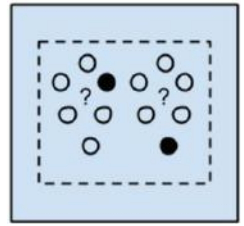
Computer Vision is Machine Learning



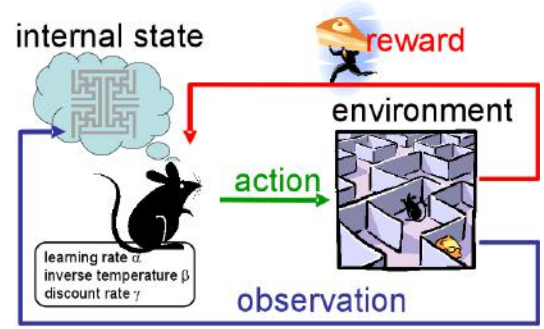
Supervised Learning



Unsupervised Learning

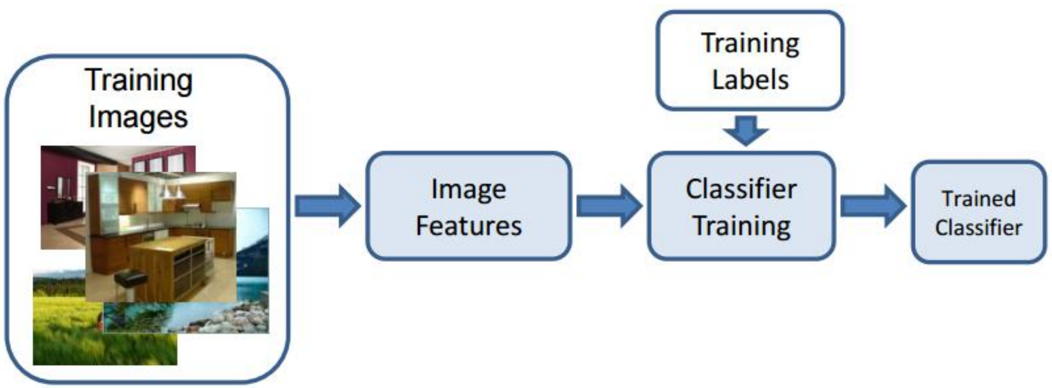


Semi-Supervised Learning

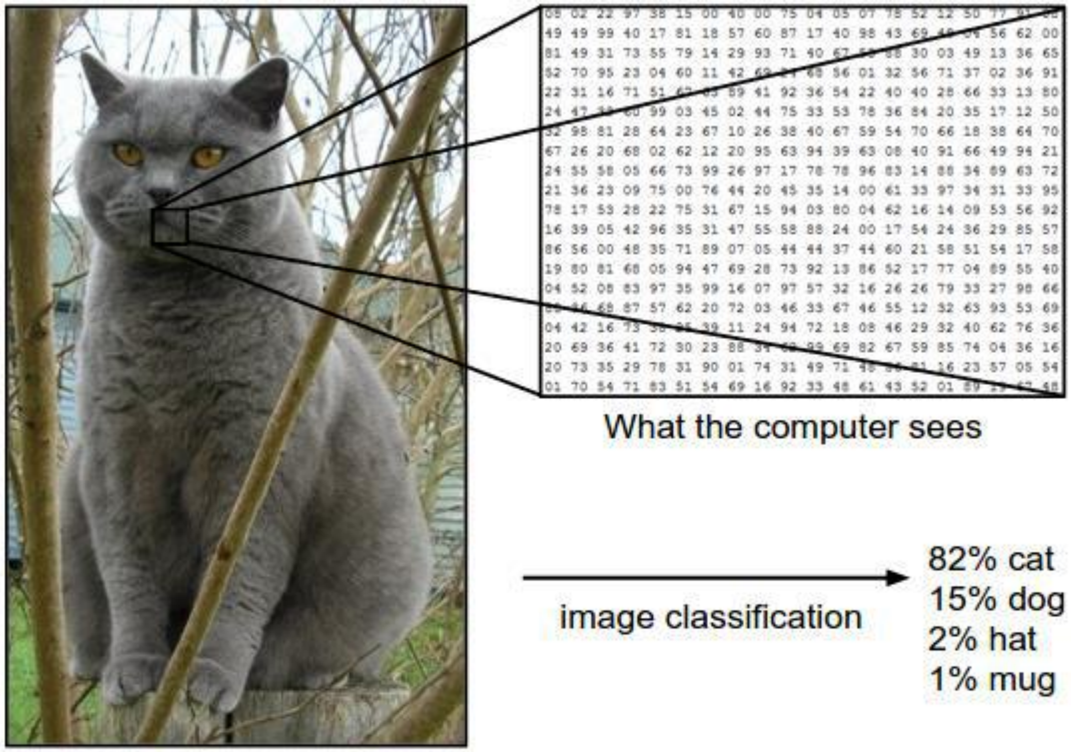


Reinforcement Learning

Computer Vision



Images are Numbers



- **Regression:** The output variable takes continuous values
- **Classification:** The output variable takes class labels
  - Underneath it may still produce continuous values such as probability of belonging to a particular class.

Human Vision Seems Easy **Why:** Data

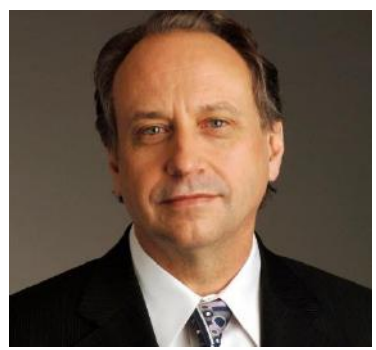
Visual perception: 540 millions years of data

Bipedal movement: 230+ million years of data

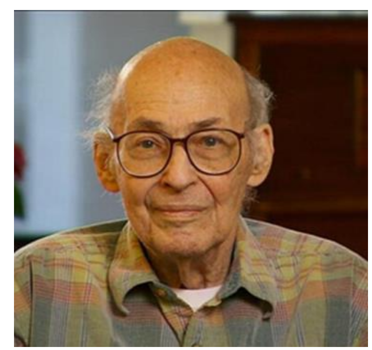
Abstract thought: 100 thousand years of data

"Encoded in the large, highly evolved sensory and motor portions of the human brain is a **billion years of experience** about the nature of the world and how to survive in it.... Abstract thought, though, is a new trick, perhaps less than **100 thousand years** old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it."
- *Hans Moravec, Mind Children (1988)*

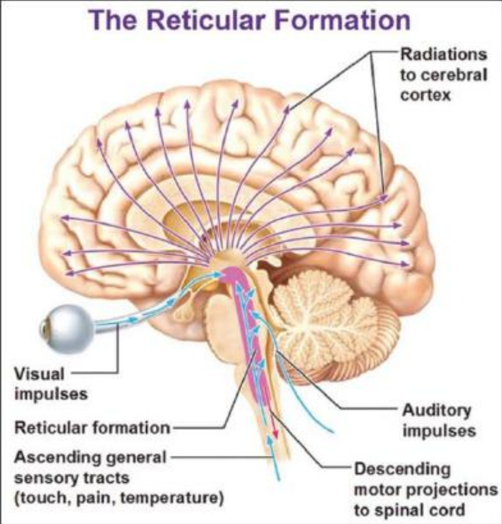
Hans Moravec (CMU)



Rodney Brooks (MIT)



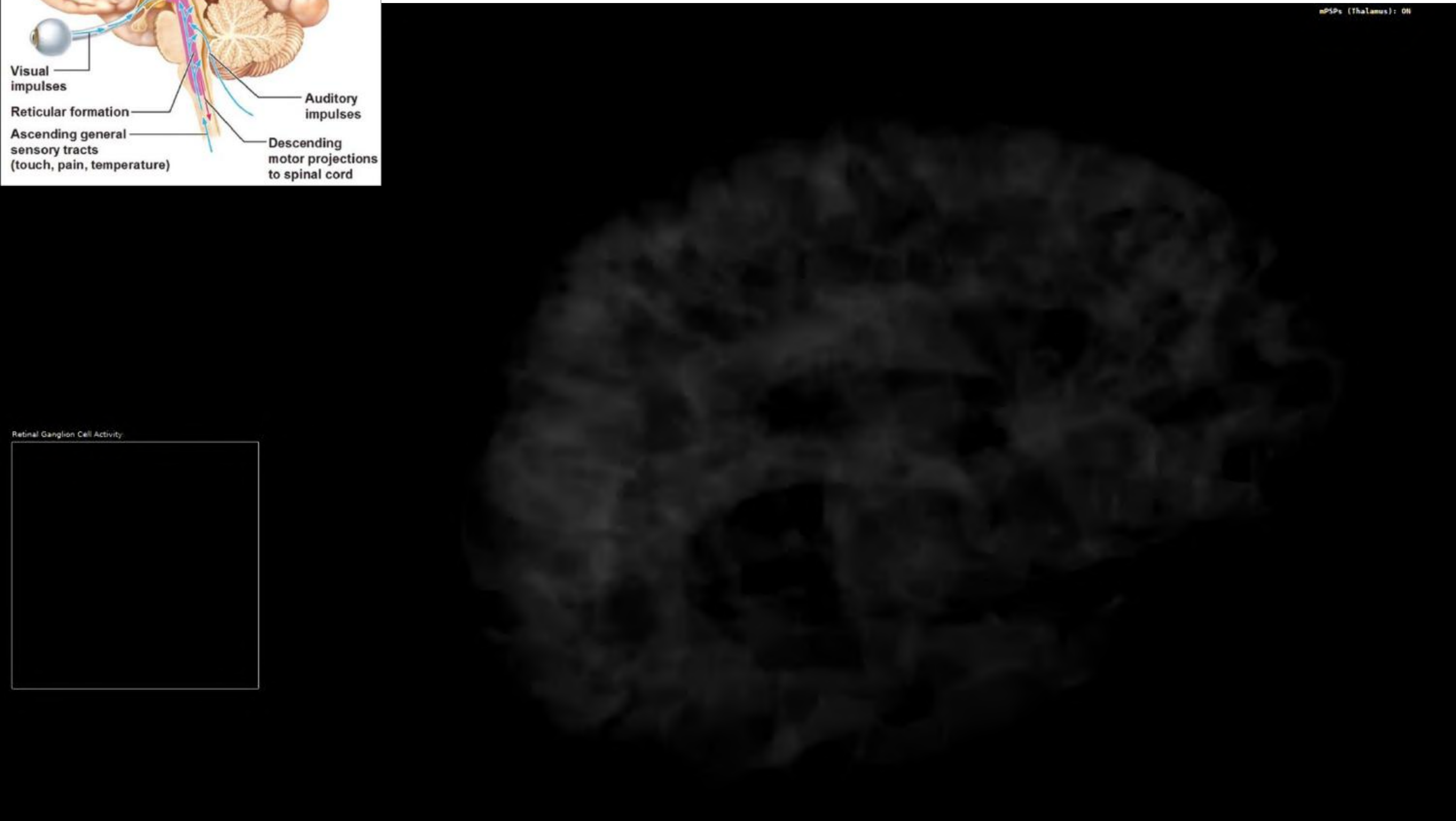
Marvin Minsky (MIT)



Human Vision

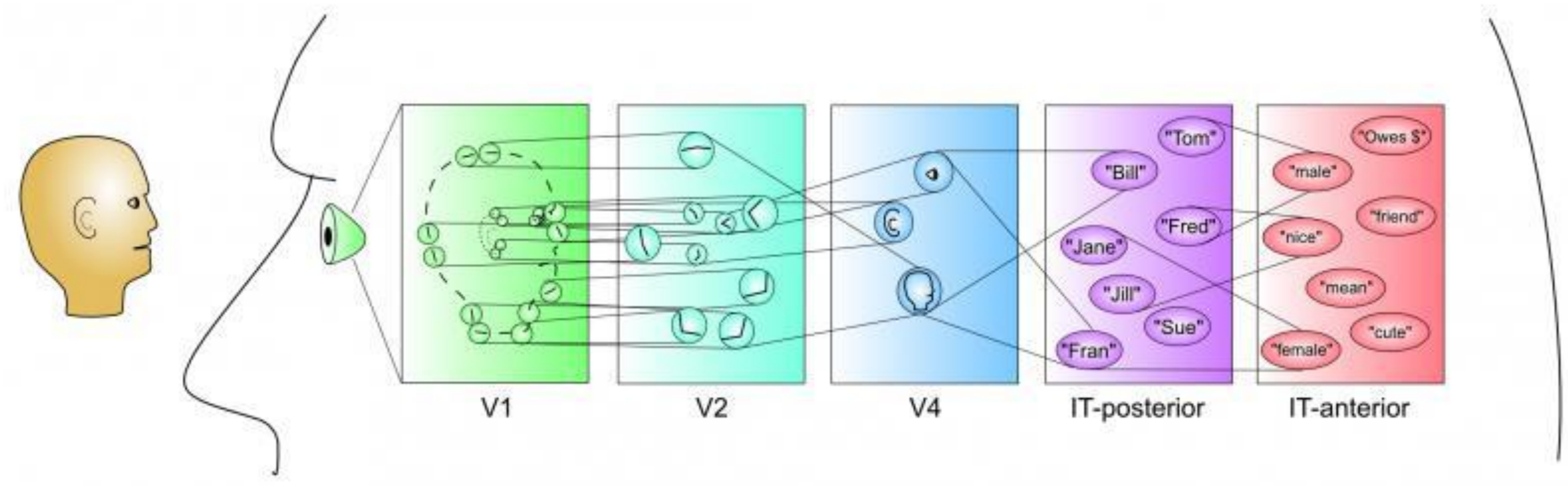
Its structure is instructive and inspiring!

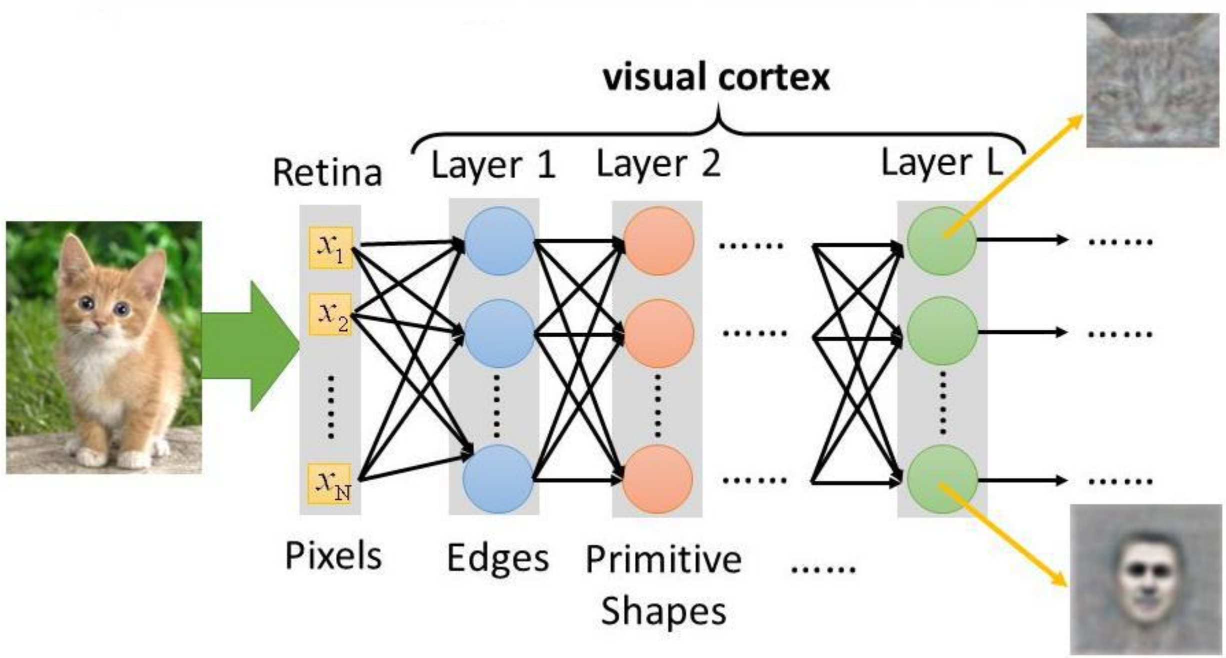
Thalamocortical System Simulation: 8 million cortical neurons + 2 billion synapses:



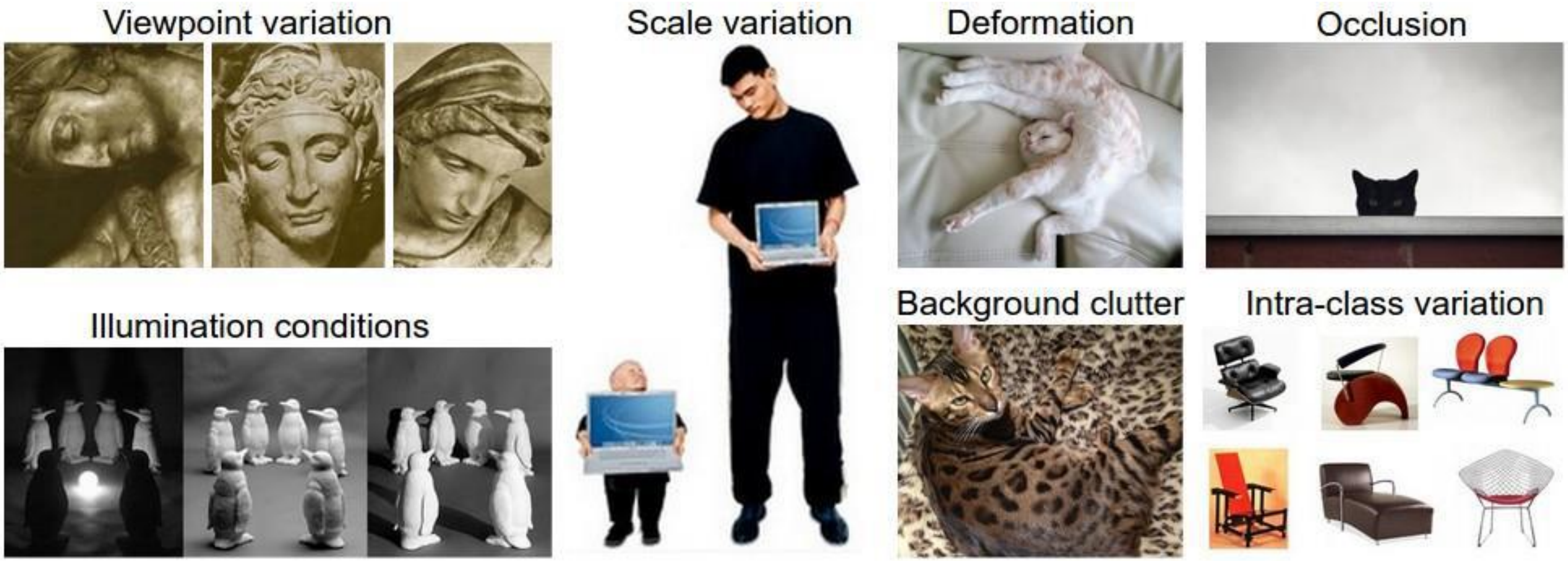
Visual Cortex

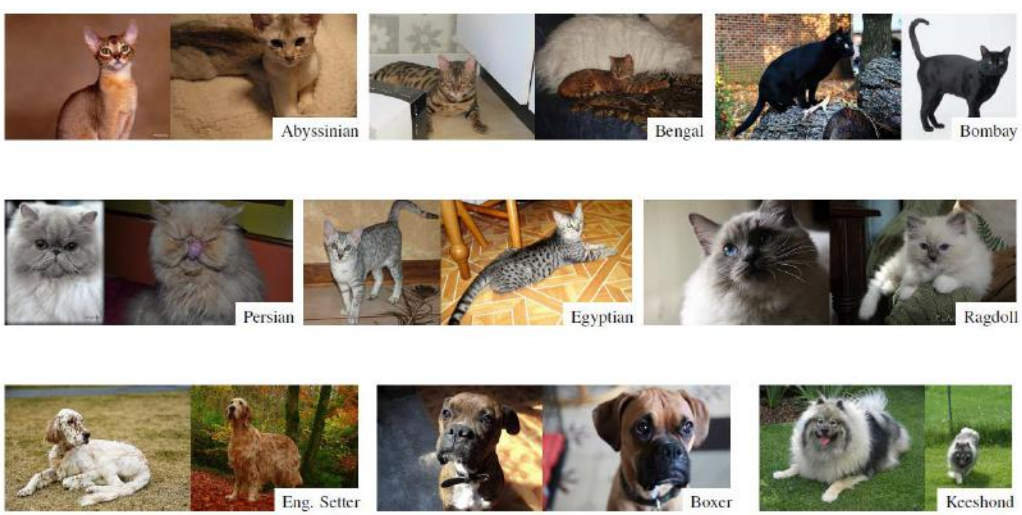
(Its Structure is Instructive and Inspiring)





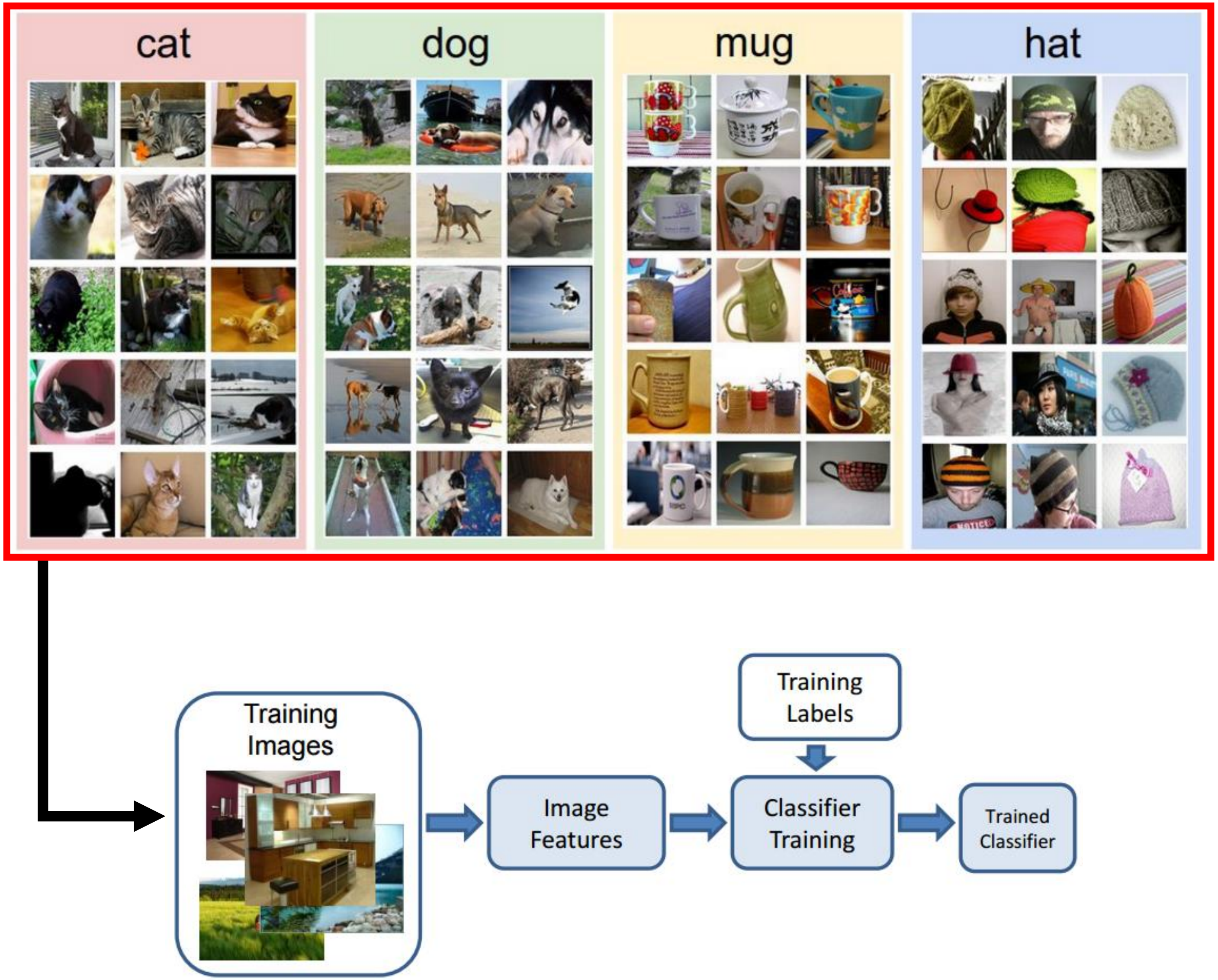
Computer Vision is Hard





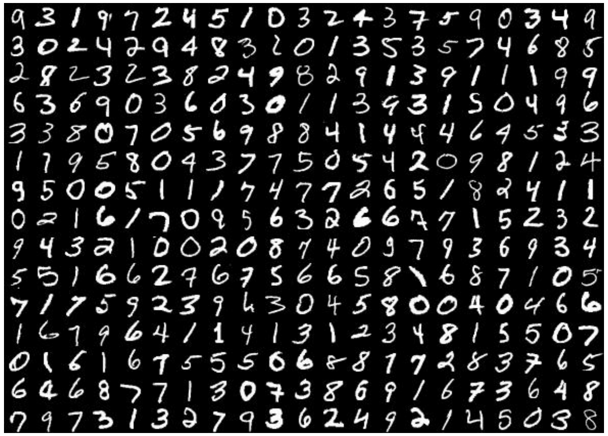
References: [66, 69, 89]

Image Classification Pipeline

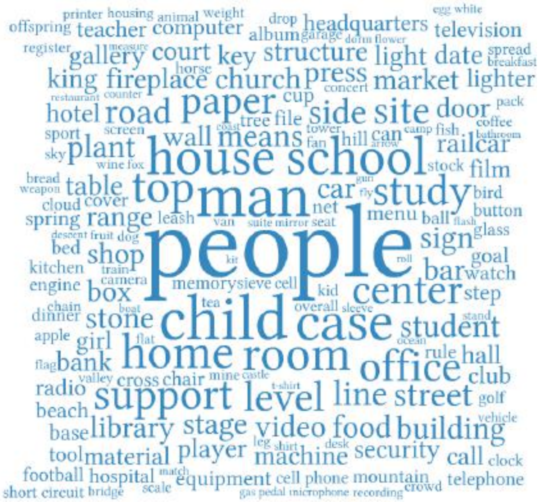


References: [81, 89]

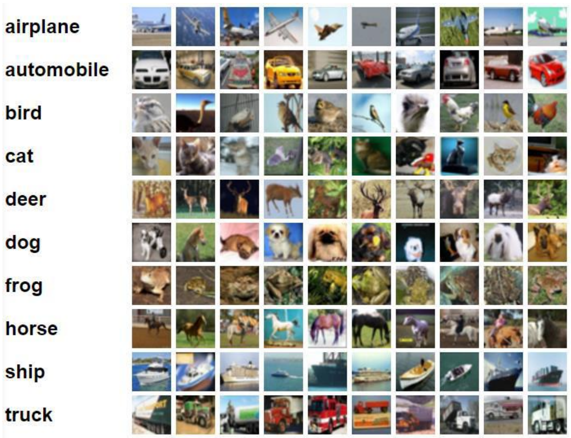
Famous Computer Vision Datasets



MNIST: handwritten digits



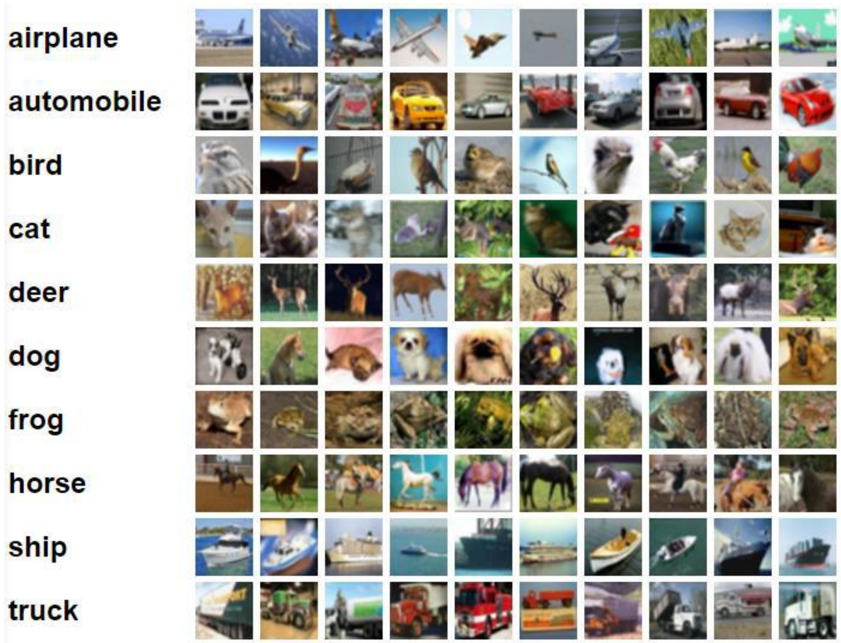
ImageNet: WordNet hierarchy



CIFAR-10(0): tiny images

Places: natural scenes

Let's Build an Image Classifier for CIFAR-10



test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

Let's Build an Image Classifier for CIFAR-10

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

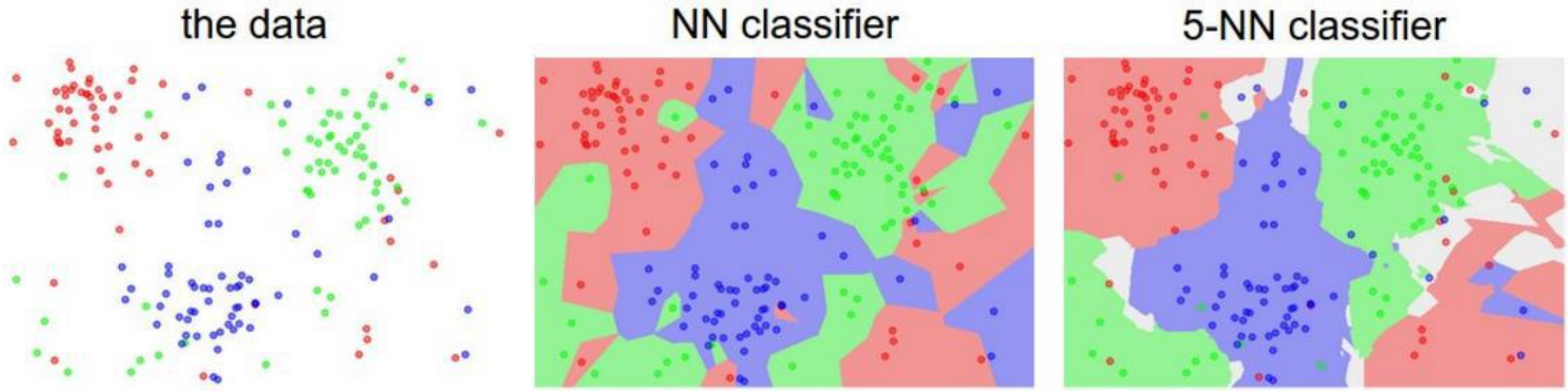
Accuracy

Random: **10%**

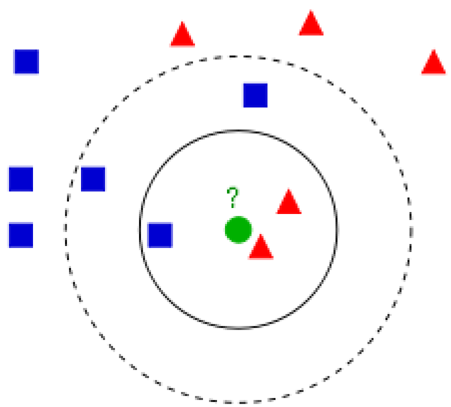
Our image-diff (with L1): **38.6%**

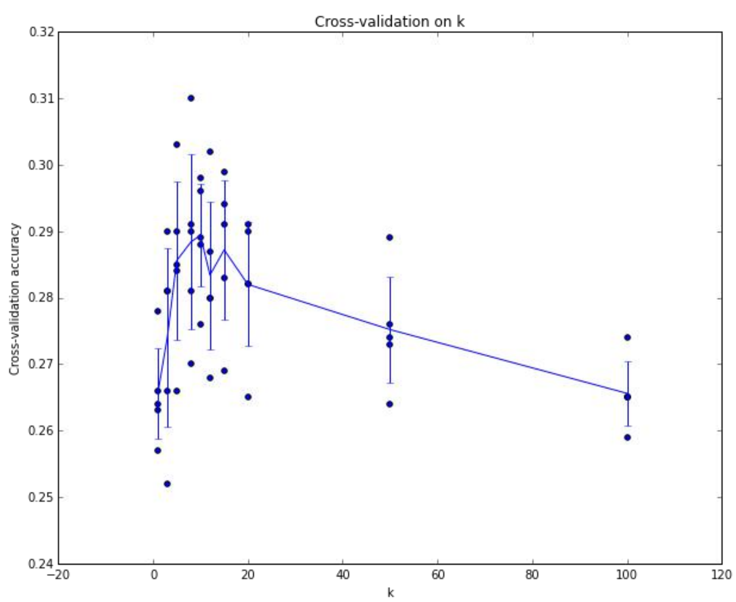
Our image-diff (with L2): **35.4%**

K-Nearest Neighbors: Generalizing the Image-Diff Classifier

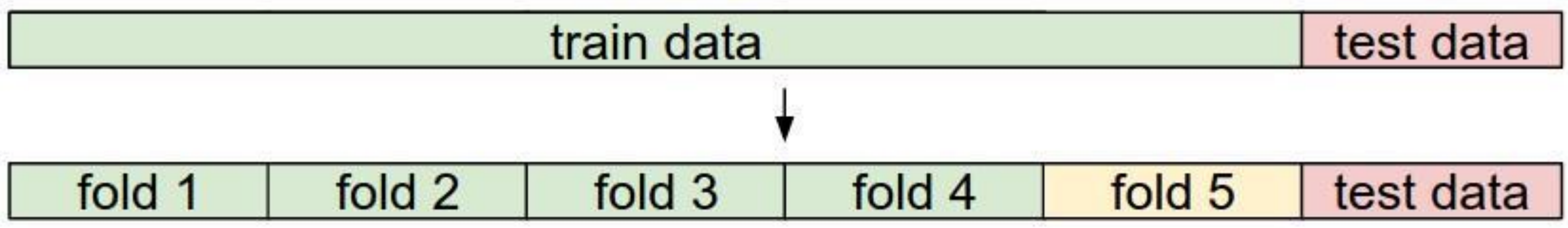


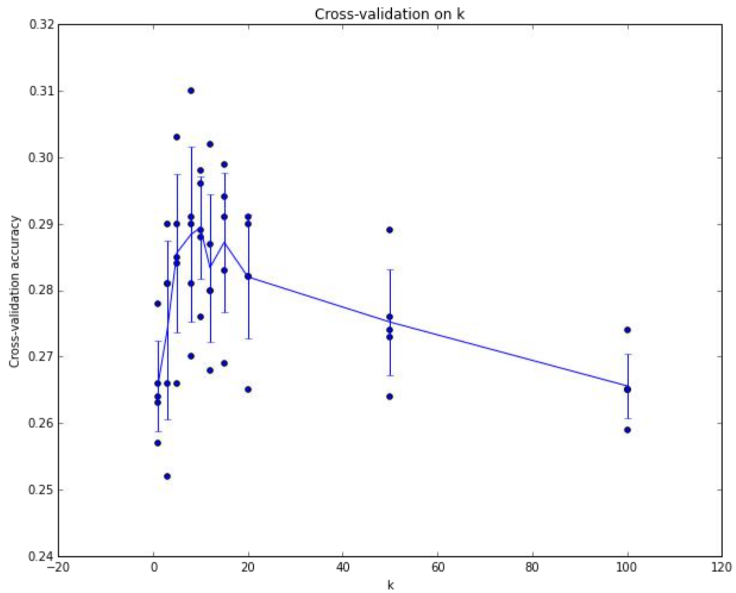
Tuning (hyper)parameters:





K-Nearest Neighbors: Generalizing the Image-Diff Classifier





Accuracy

Random: **10%**

Training and testing on the same data: **35.4%**

7-Nearest Neighbors: **~30%**

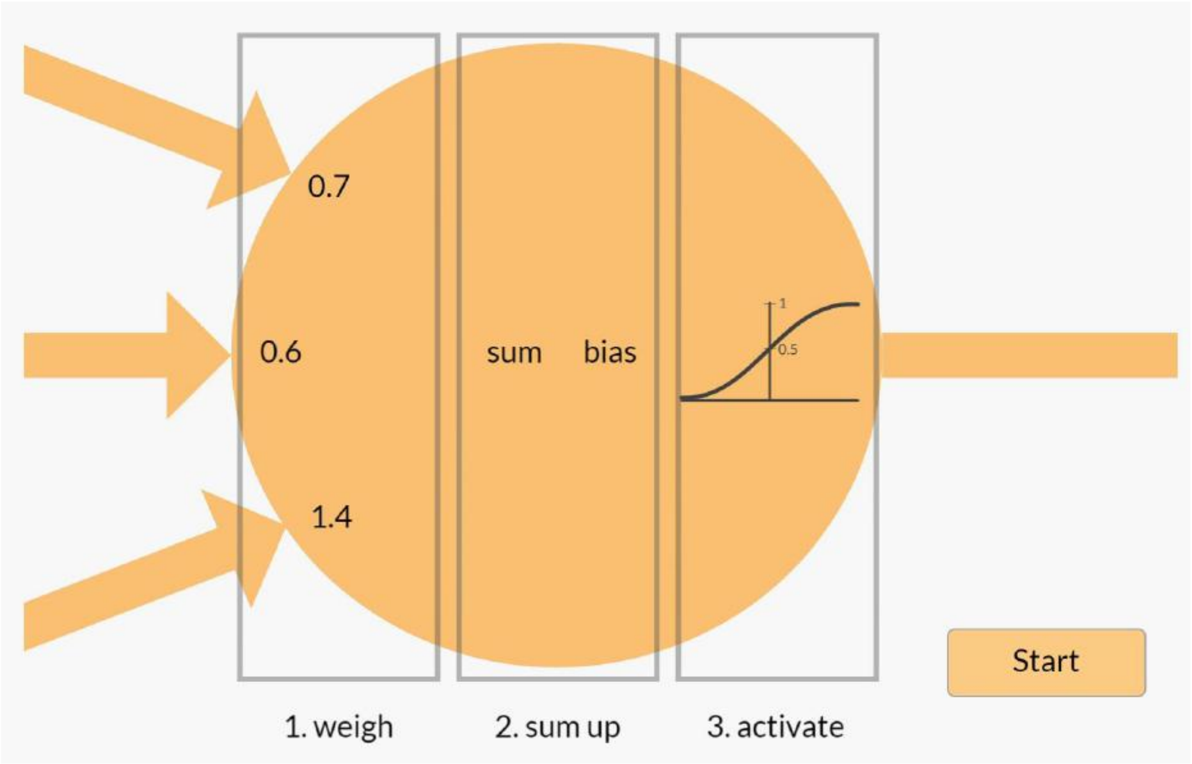
Human: **~94%**

...

Convolutional Neural Networks: **~95%**

Reminder: Weighing the Evidence

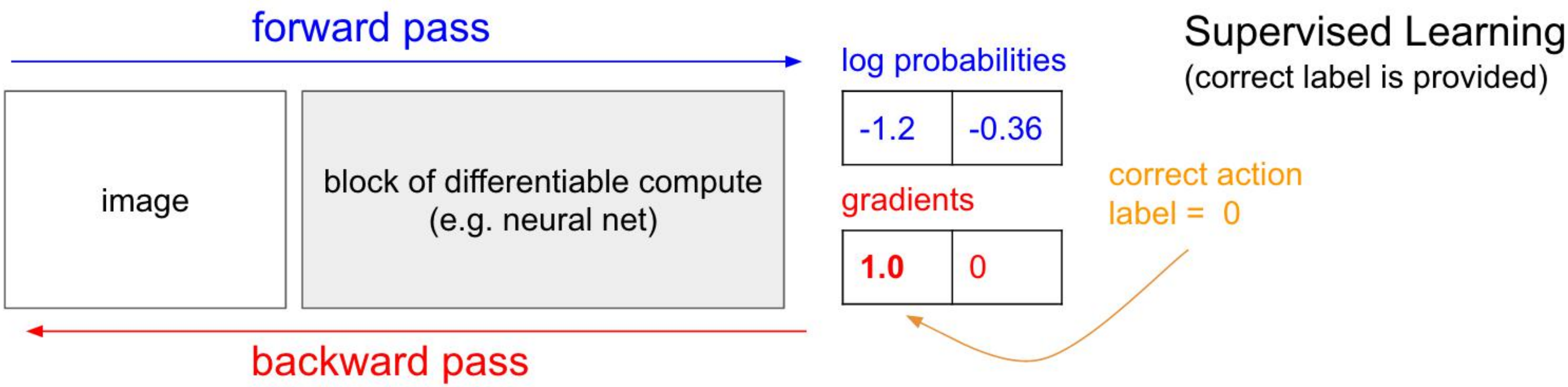
Evidence



Decisions

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Reminder: "Learning" is Optimization of a Function

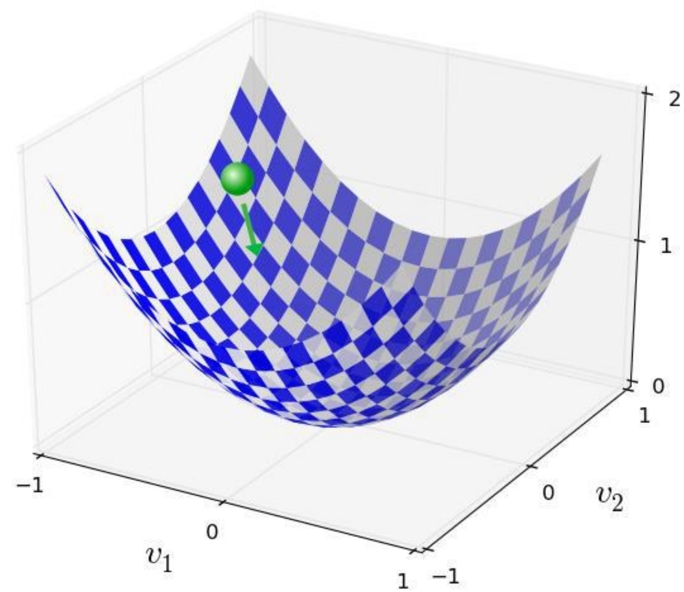


Ground truth for "6":

$$y(x) = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0)^T$$

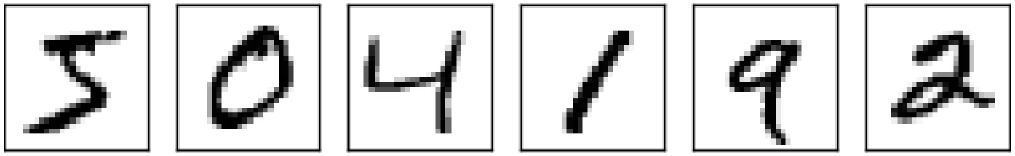
"Loss" function:

$$C(w, b) \equiv \frac{1}{2n} \sum_x \| y(x) - a \|^2$$

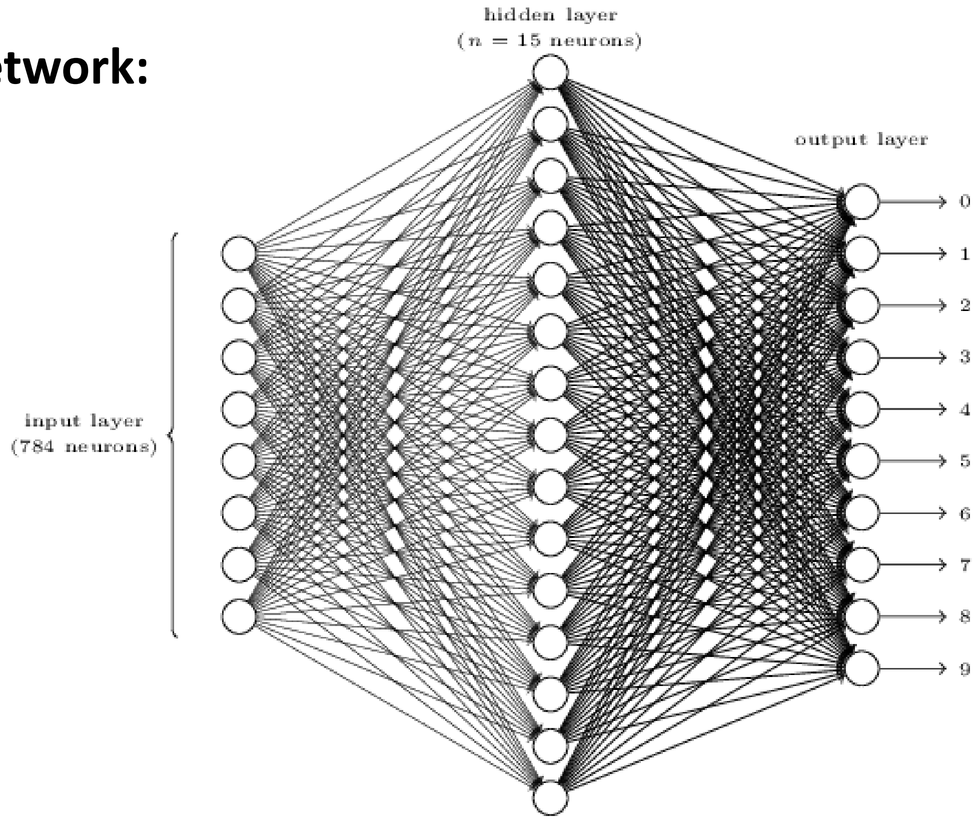


Classify and Image of a Number

Input:
(28x28)

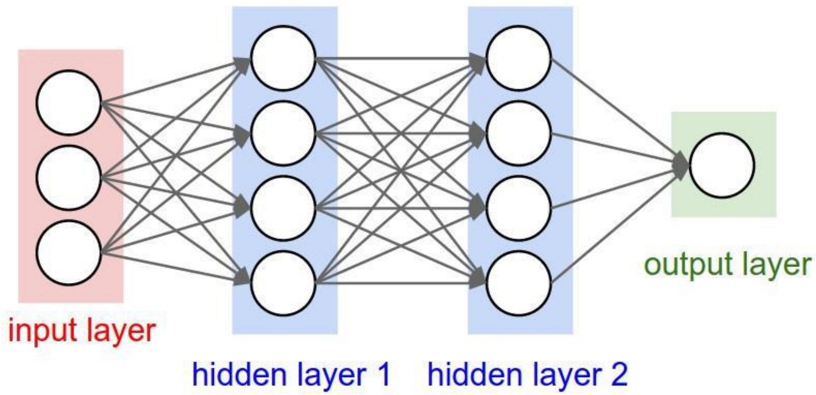


Network:

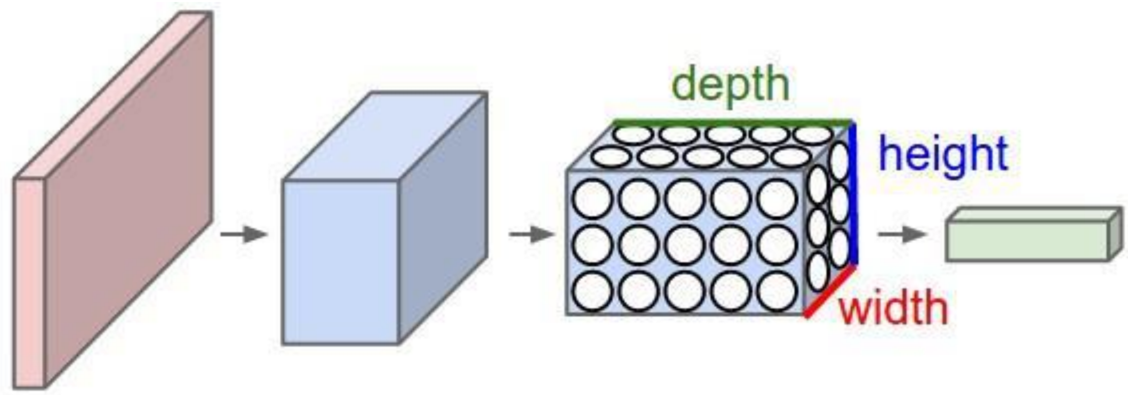


Convolutional Neural Networks

Regular neural network (fully connected):



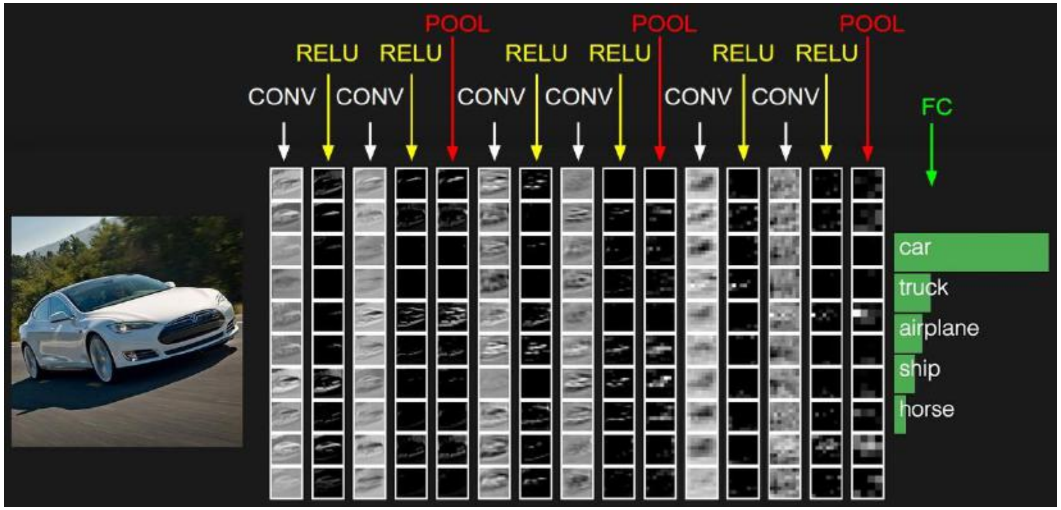
Convolutional neural network:



Each layer takes a 3d volume, produces 3d volume with some smooth function that may or may not have parameters.

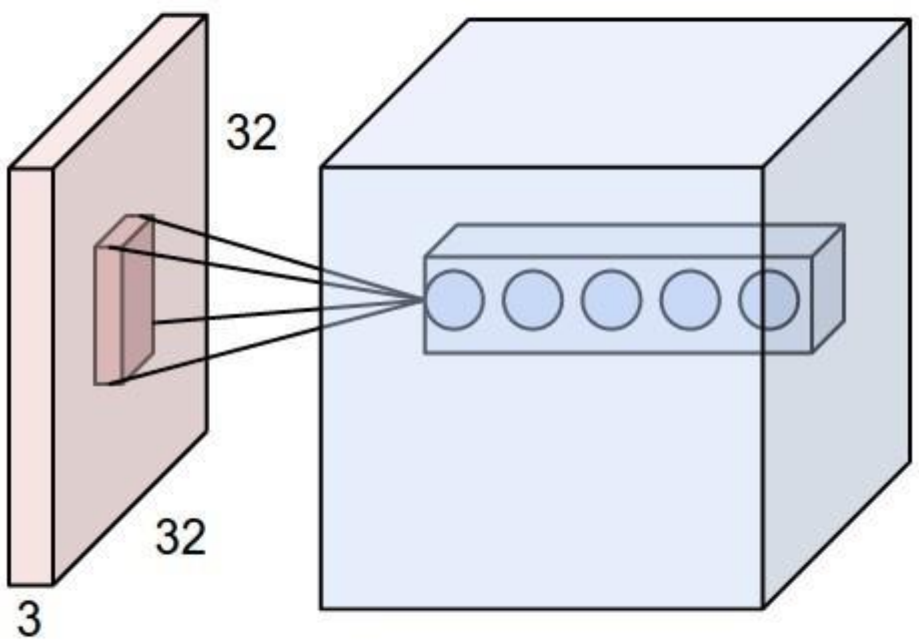
Convolutional Neural Networks: Layers

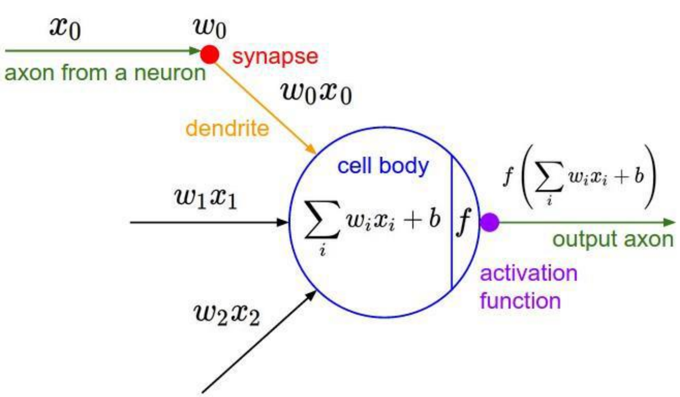
- **INPUT** [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- **CONV** layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- **RELU** layer will apply an elementwise activation function, such as the *max(0,x)* thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- **POOL** layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- **FC** (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.



Layers **highlighted in blue** have learnable parameters.

Dealing with Images: Local Connectivity



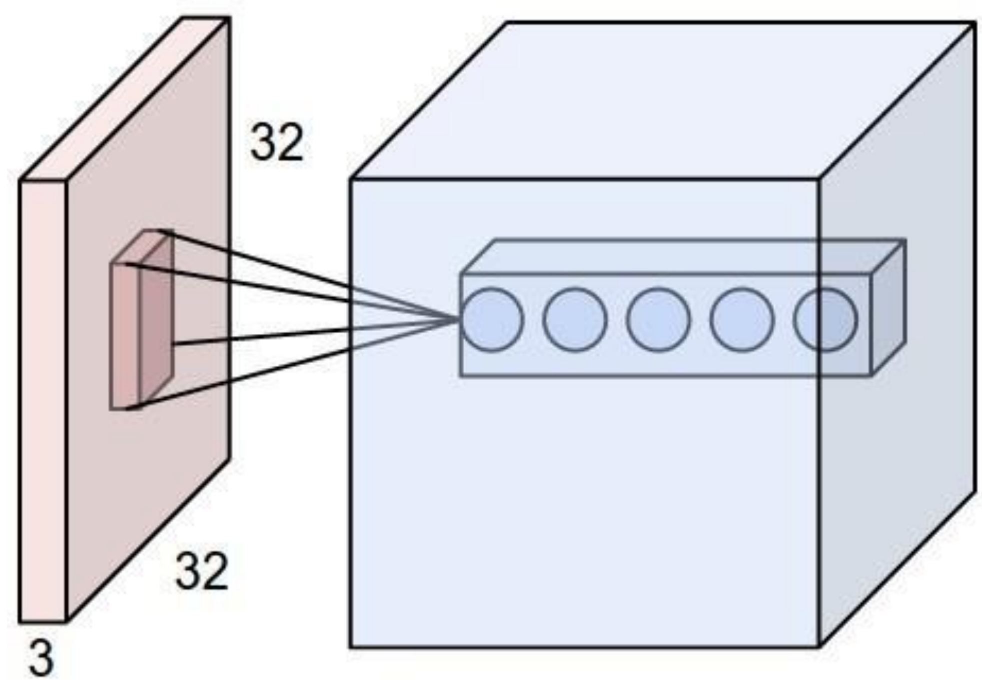


Same neuron. Just more focused (narrow "receptive field").

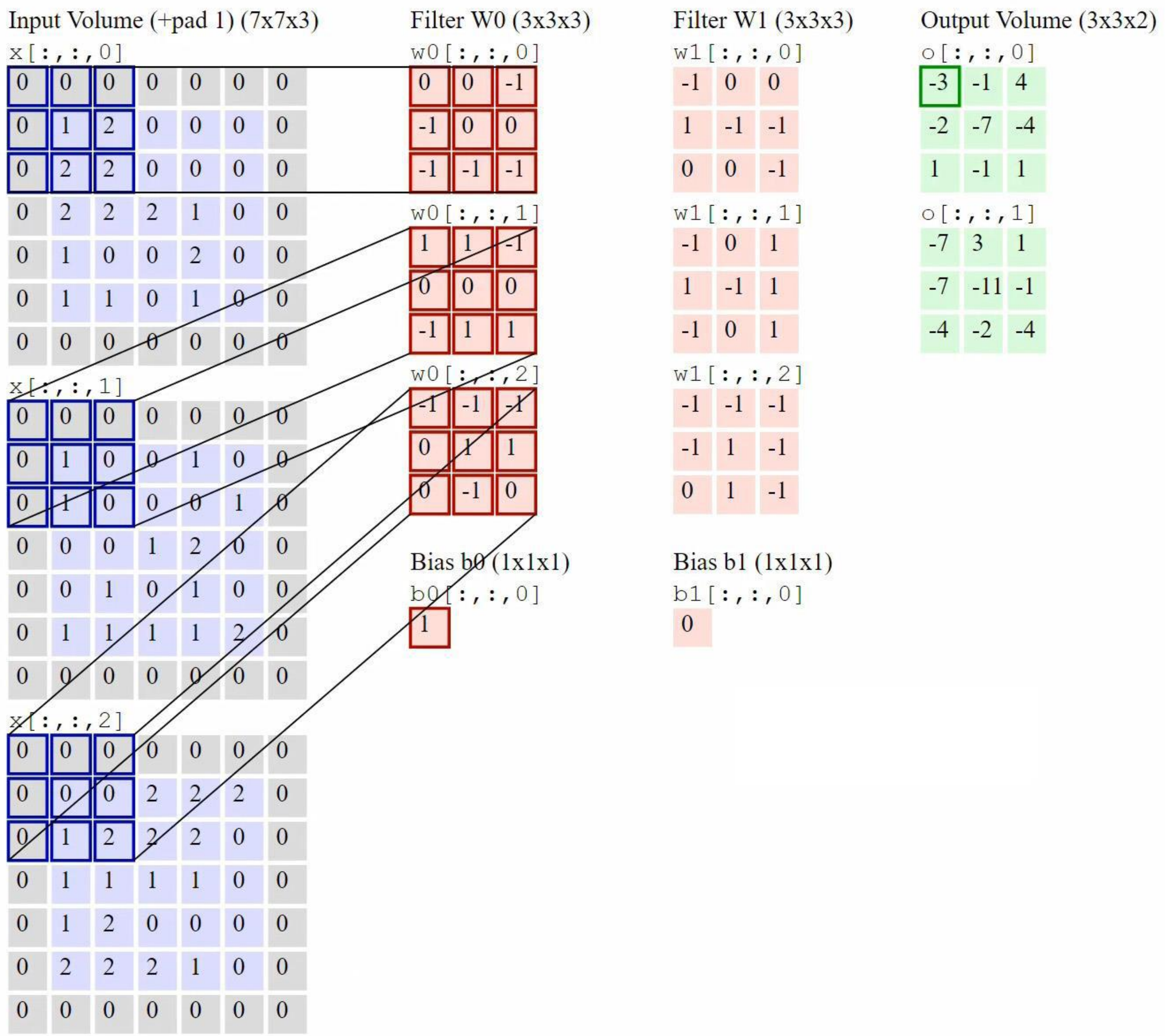
The parameters on a each filter are spatially "shared"

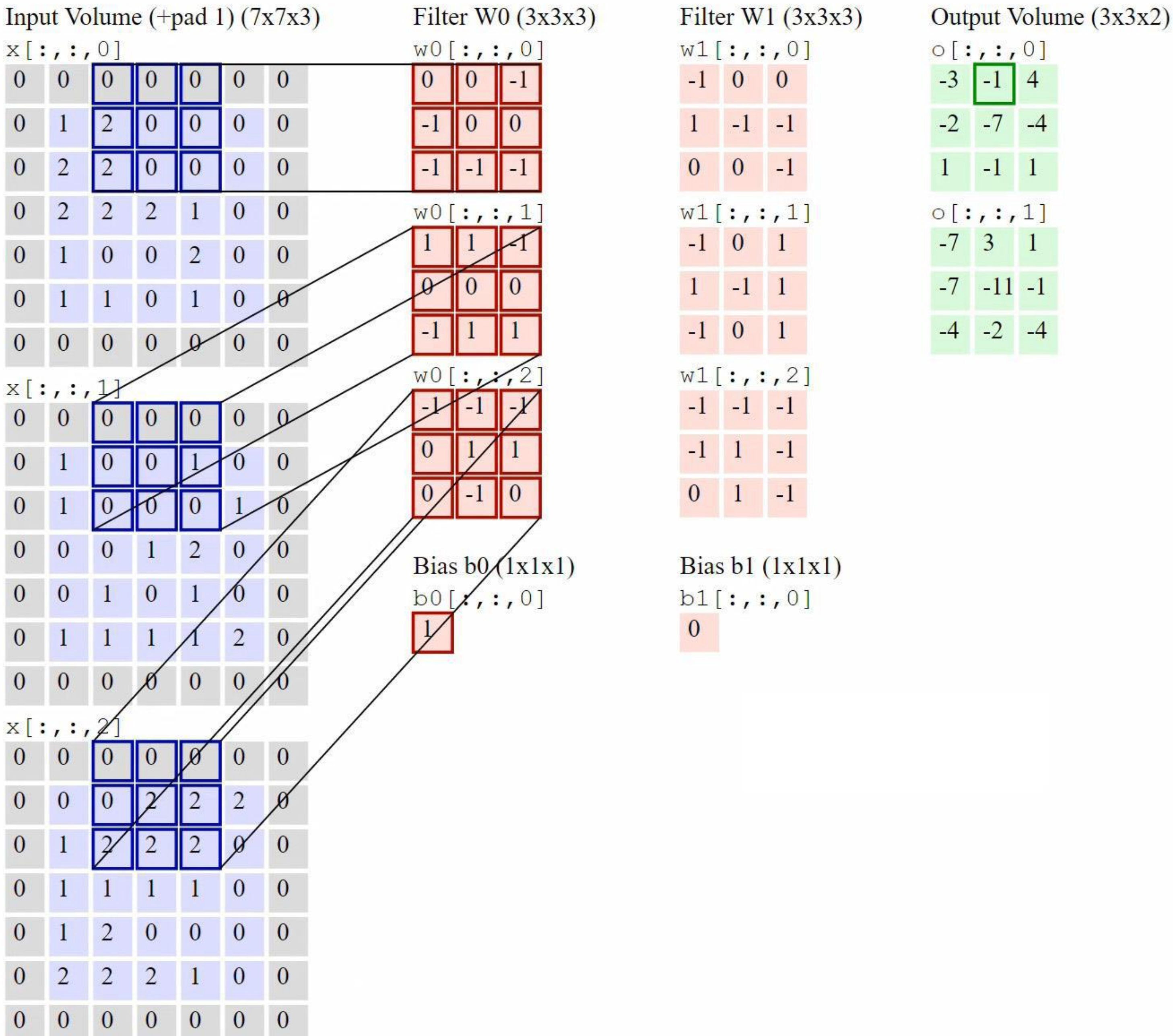
(if a feature is useful in one place, it's useful elsewhere)

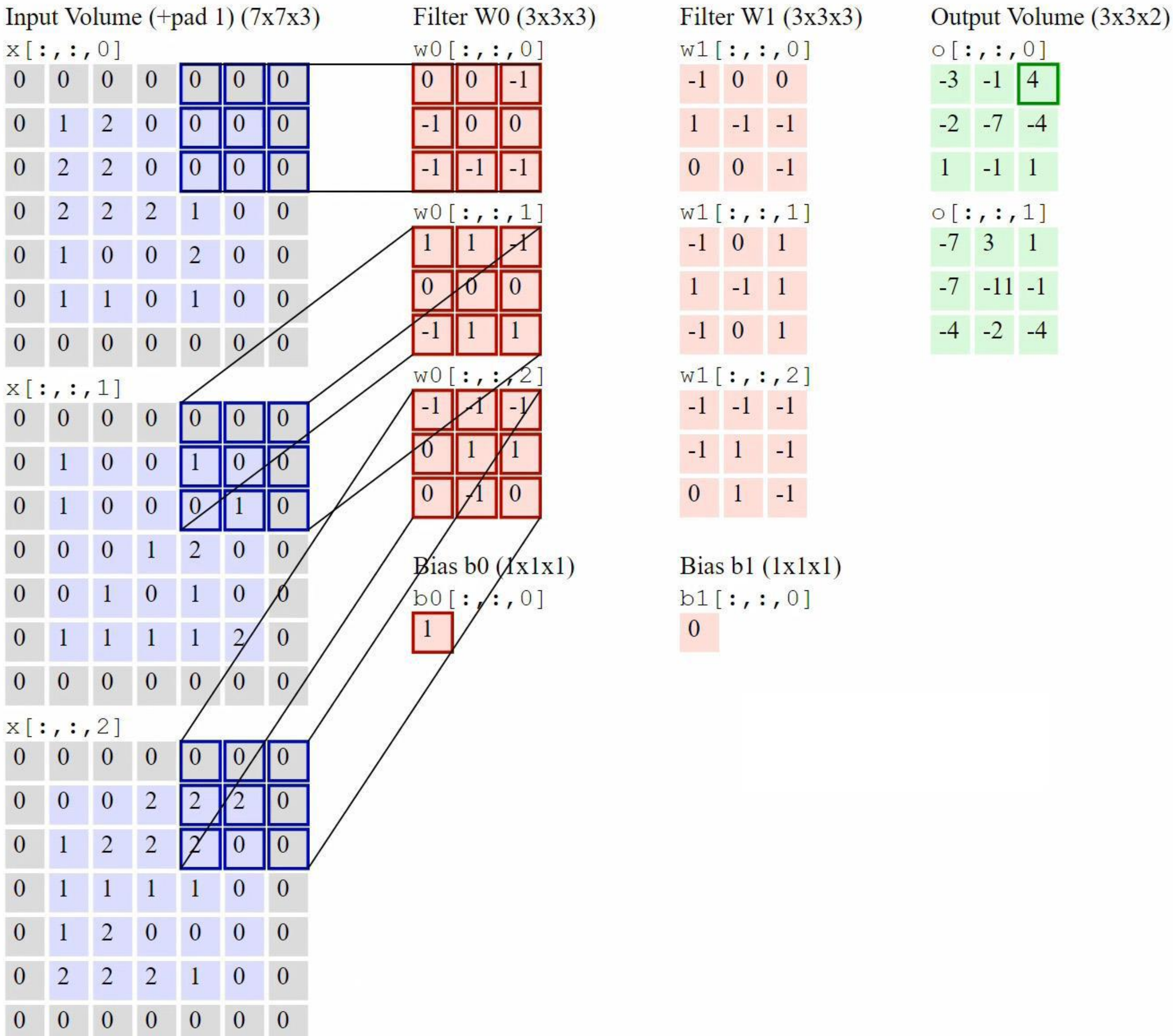
ConvNets: Spatial Arrangement of Output Volume



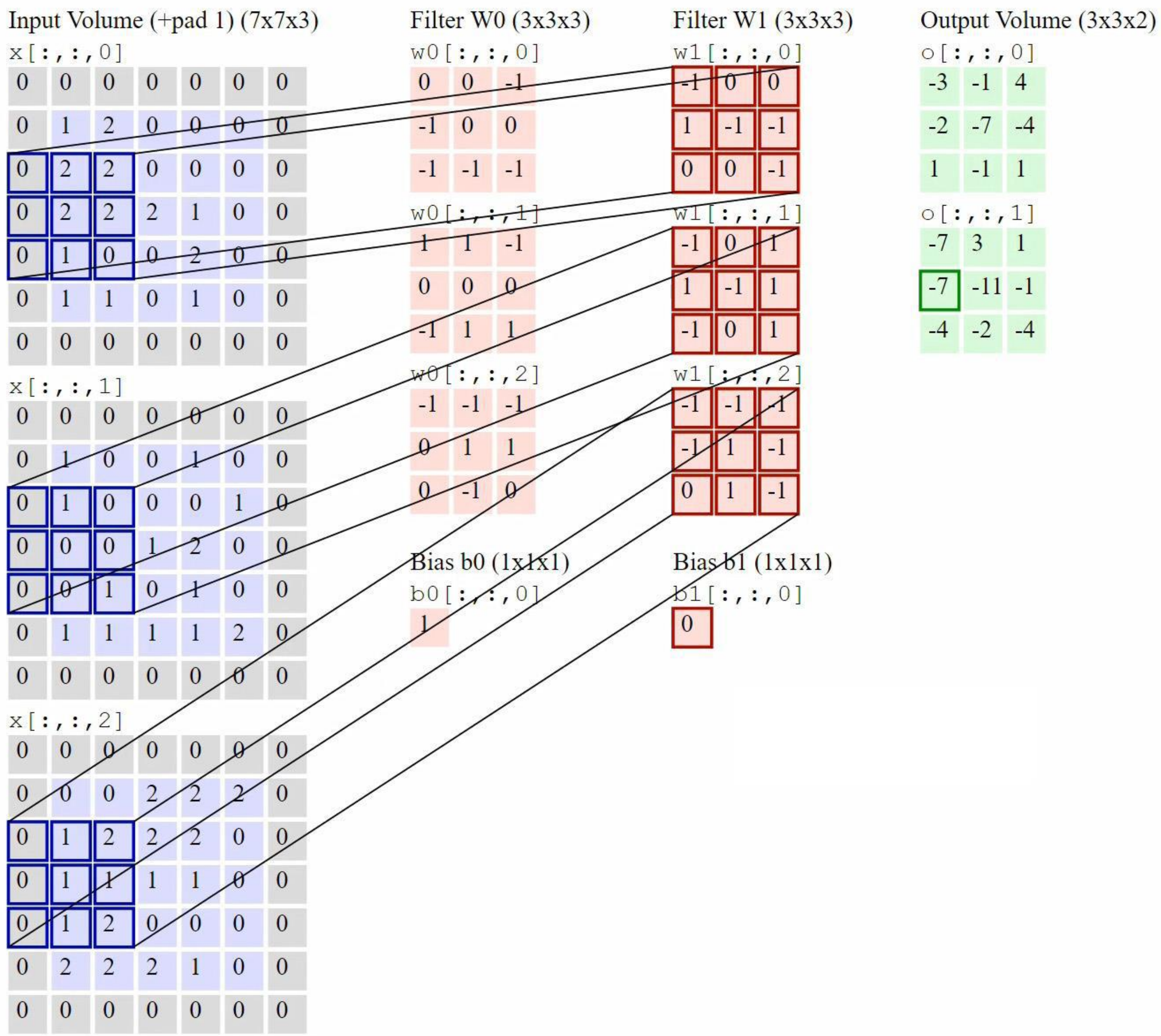
- **Depth:** number of filters
- **Stride:** filter step size (when we "slide" it)
- **Padding:** zero-pad the input


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

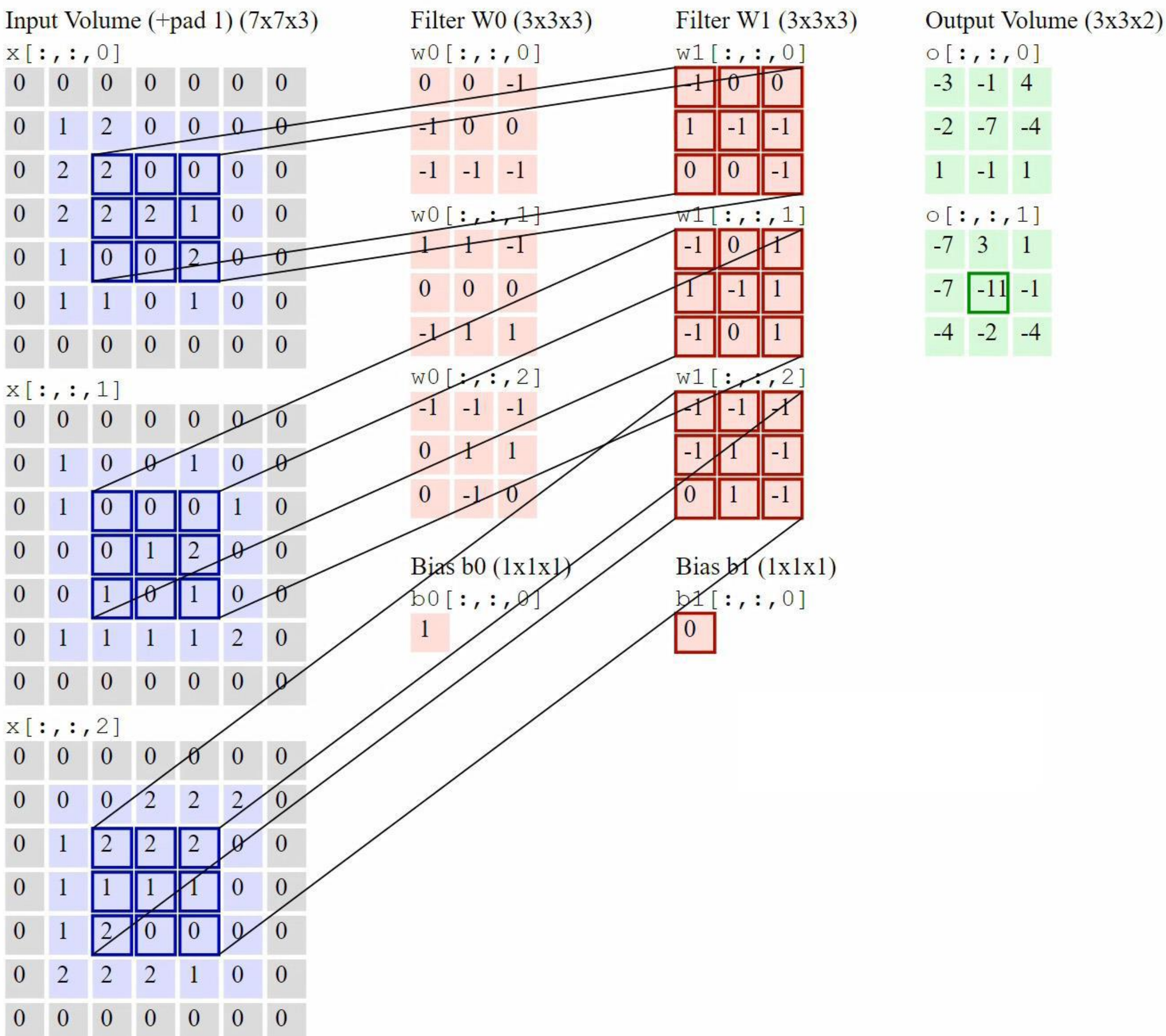

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

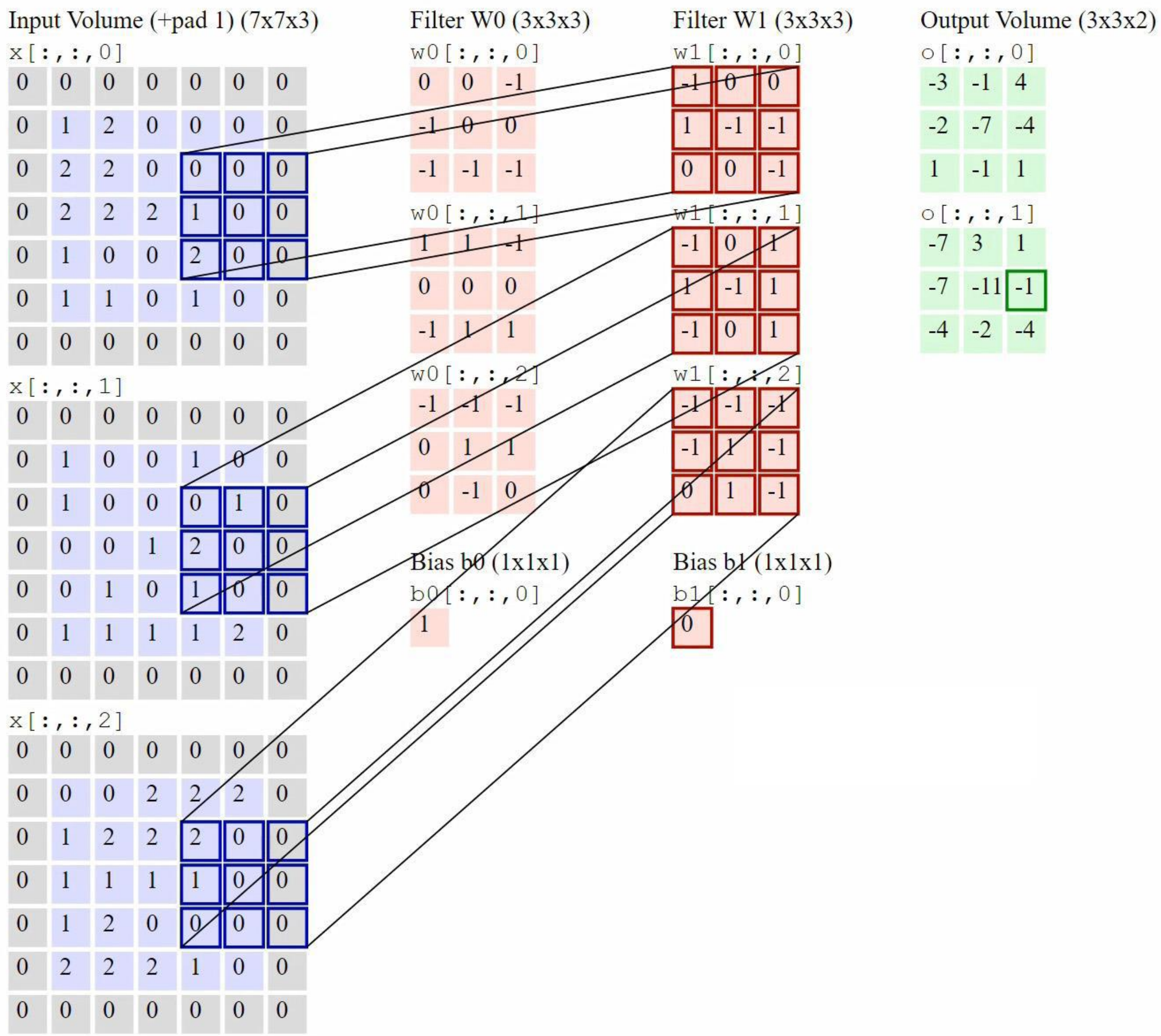


References: [95]

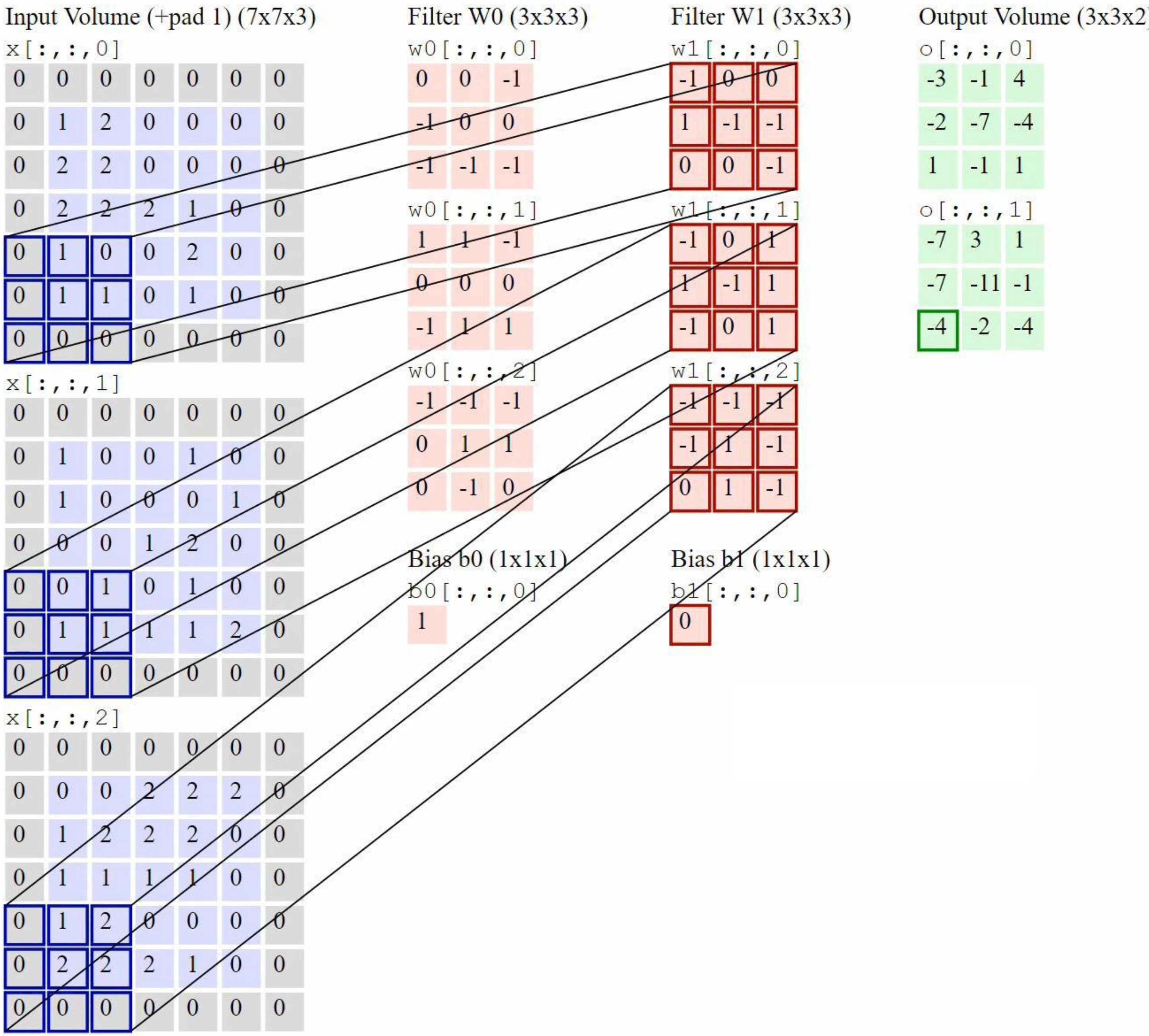


References: [95]

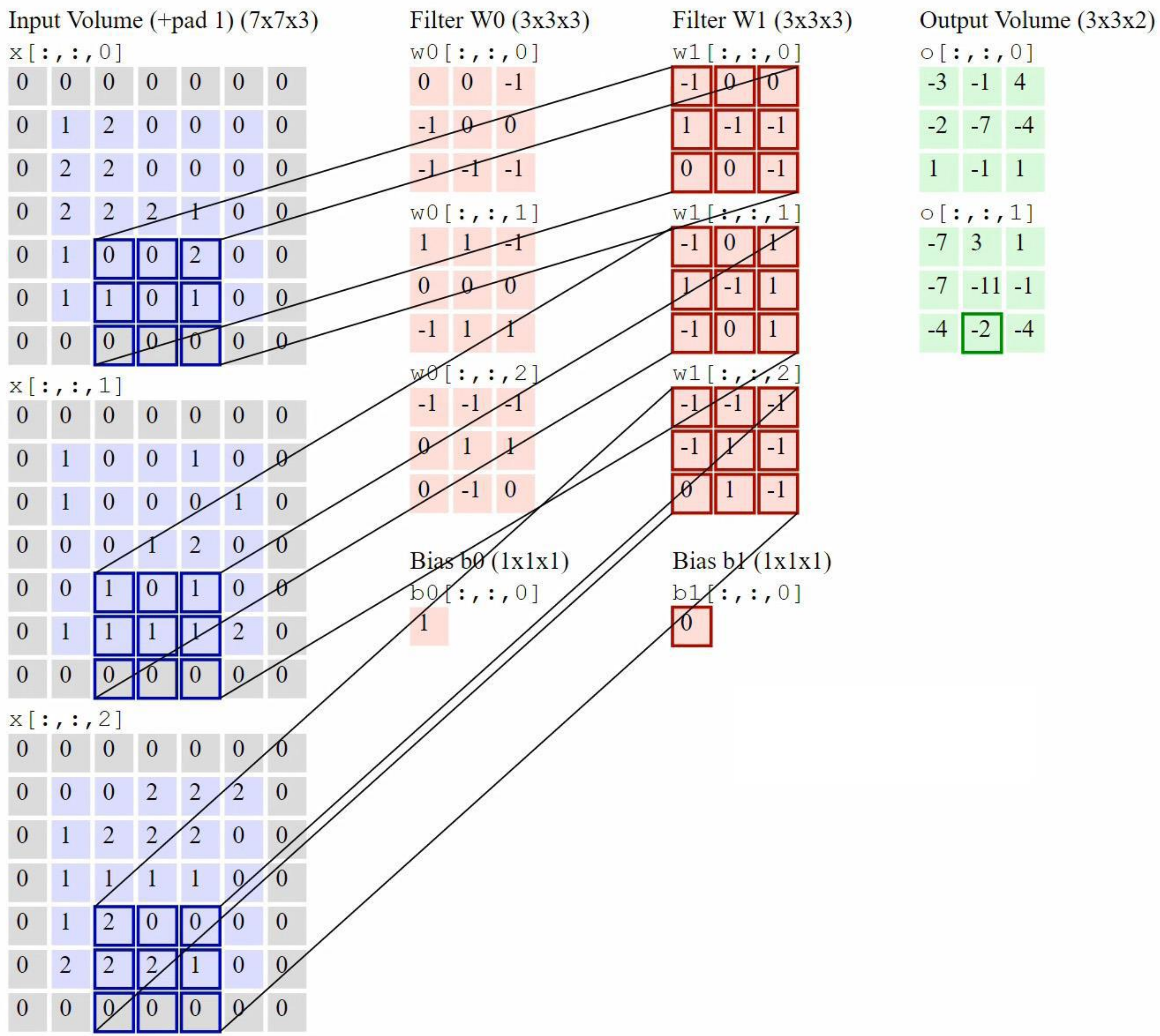

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4



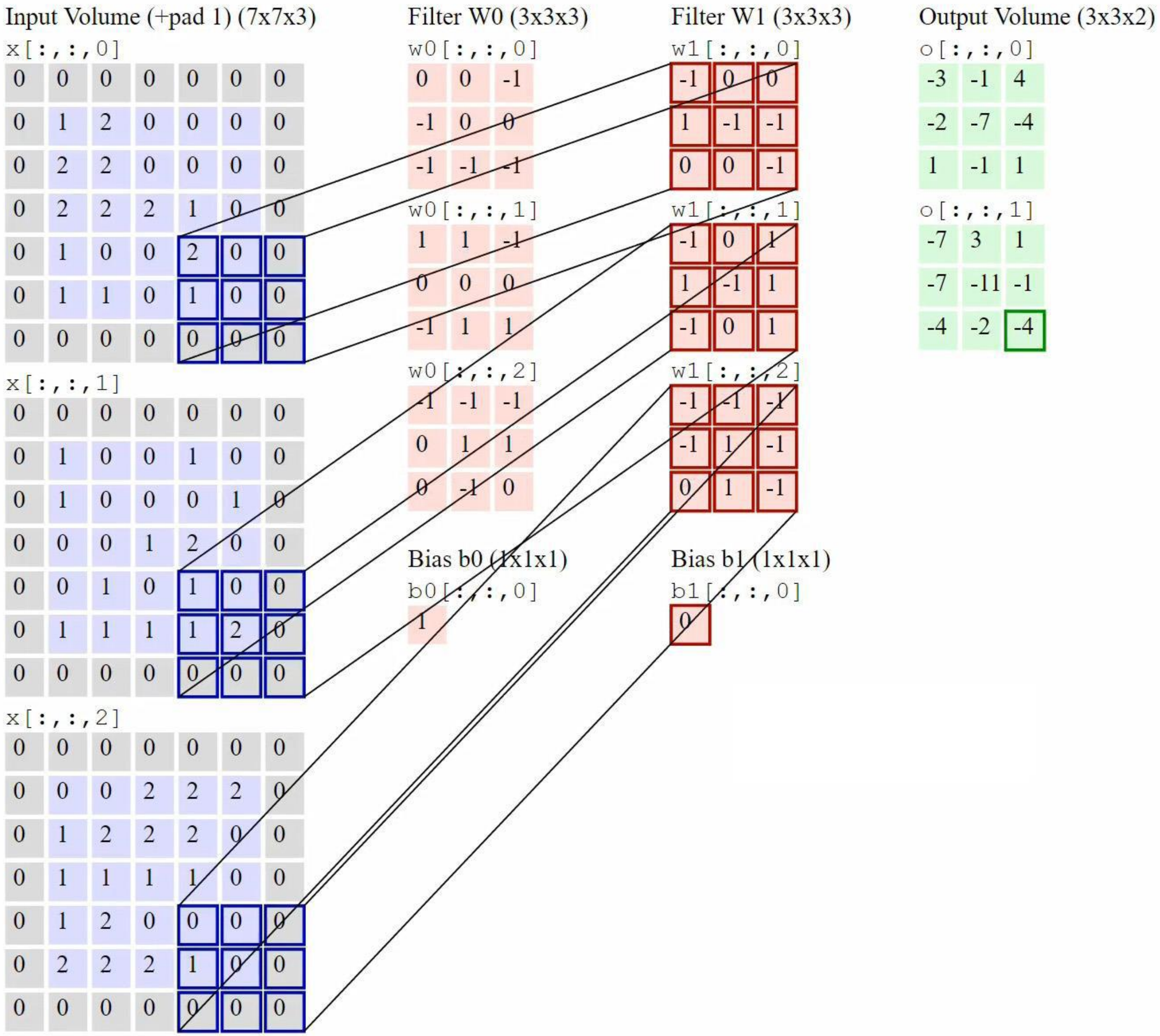
Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4



References: [95]



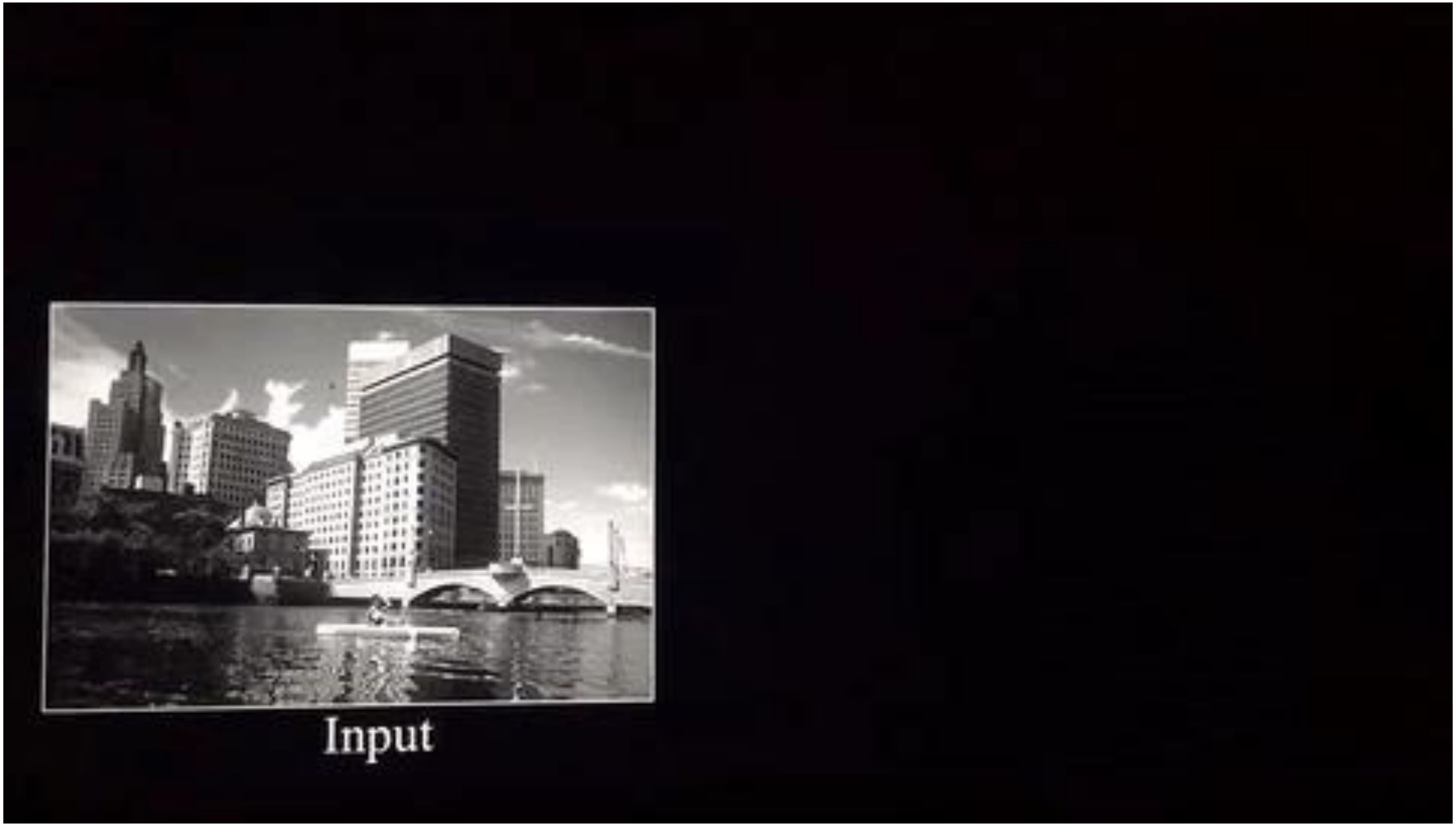
References: [95]


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

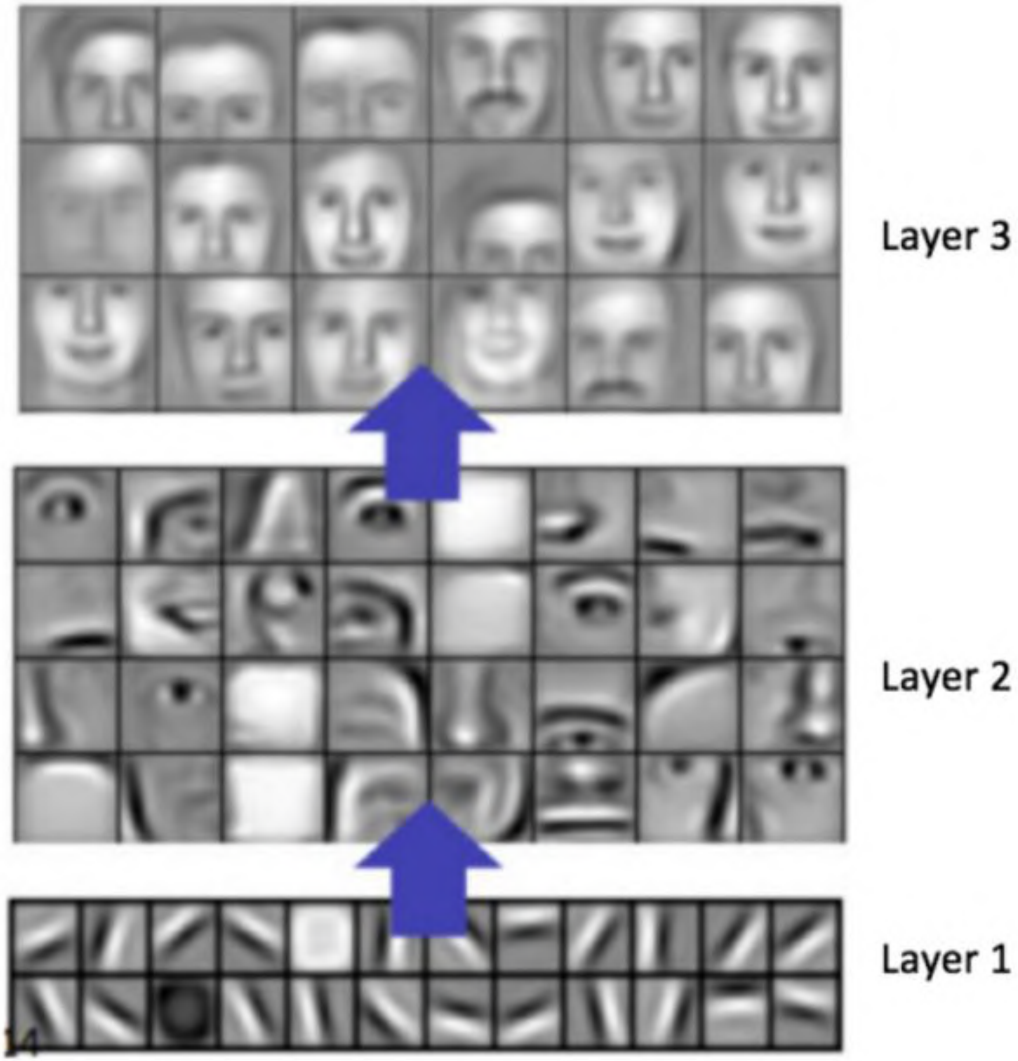
Convolution

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

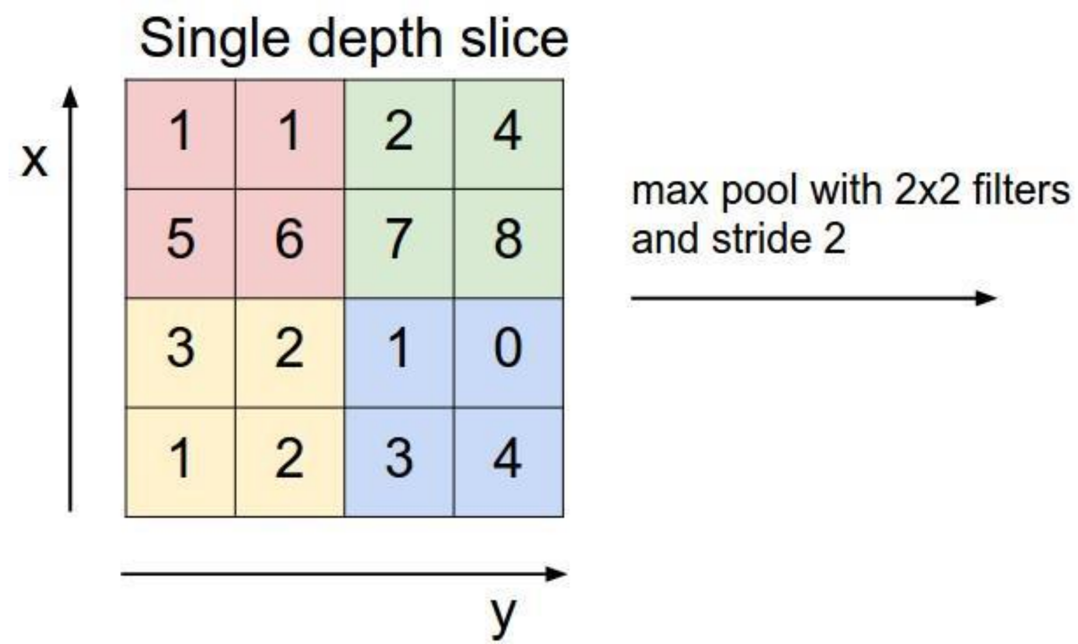
Convolution

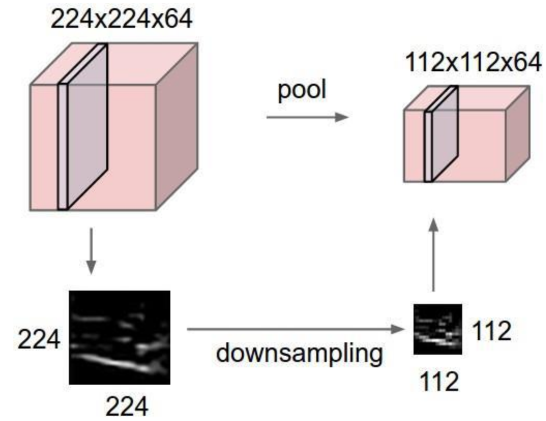


Convolution: Representation Learning

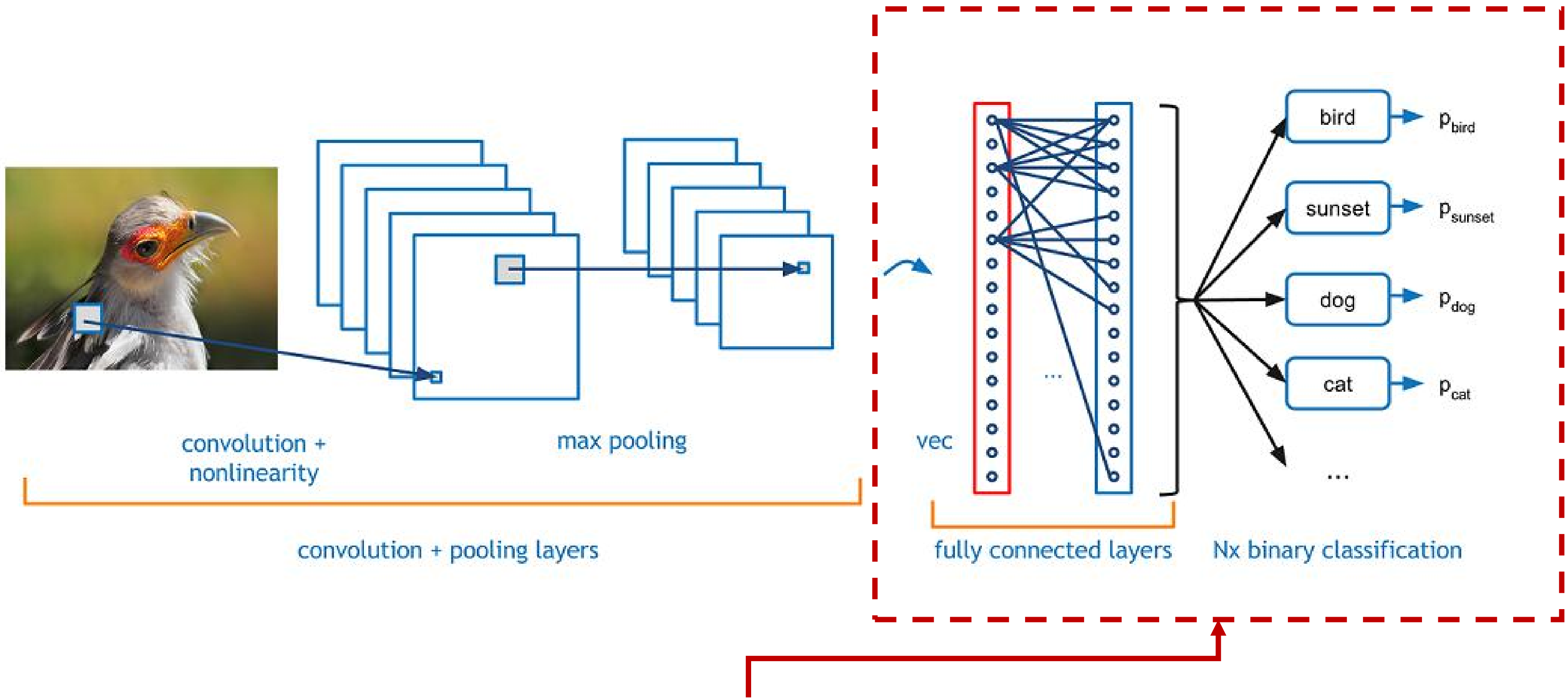


ConvNets: Pooling





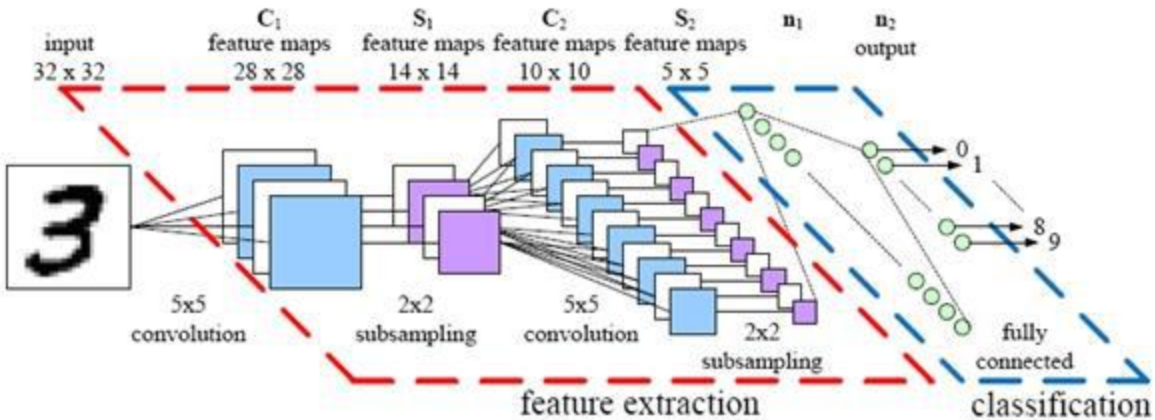
Same Architecture, Many Applications



This part might look different for:

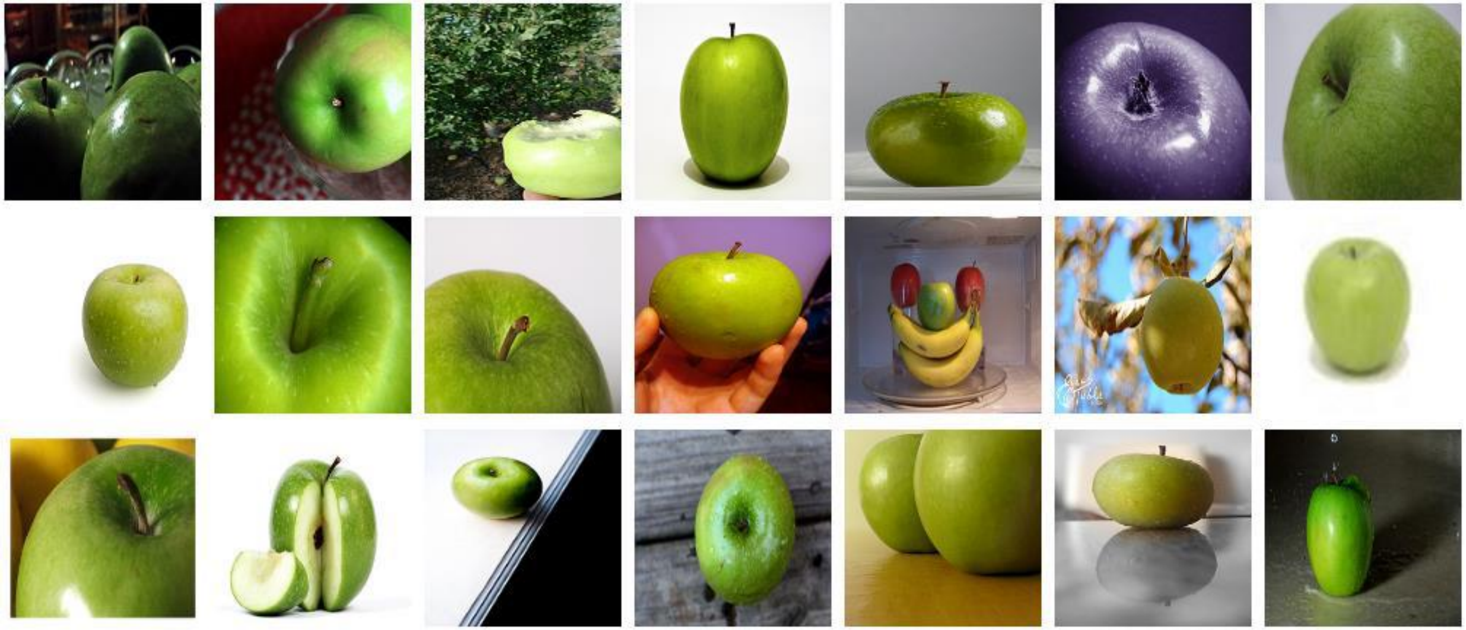
- Different image classification domains
- Image captioning with recurrent neural networks
- Image object localization with bounding box
- Image segmentation with fully convolutional networks
- Image segmentation with deconvolution layers

Object Recognition
Case Study: **ImageNet**



What is ImageNet?

- **ImageNet:** dataset of 14+ million images (21,841 categories)
  - Links to images not images
- Let's take the high level category of **fruit** as an example:
  - Total 188,000 images of fruit
  - There are 1206 Granny Smith apples:



What is ImageNet?

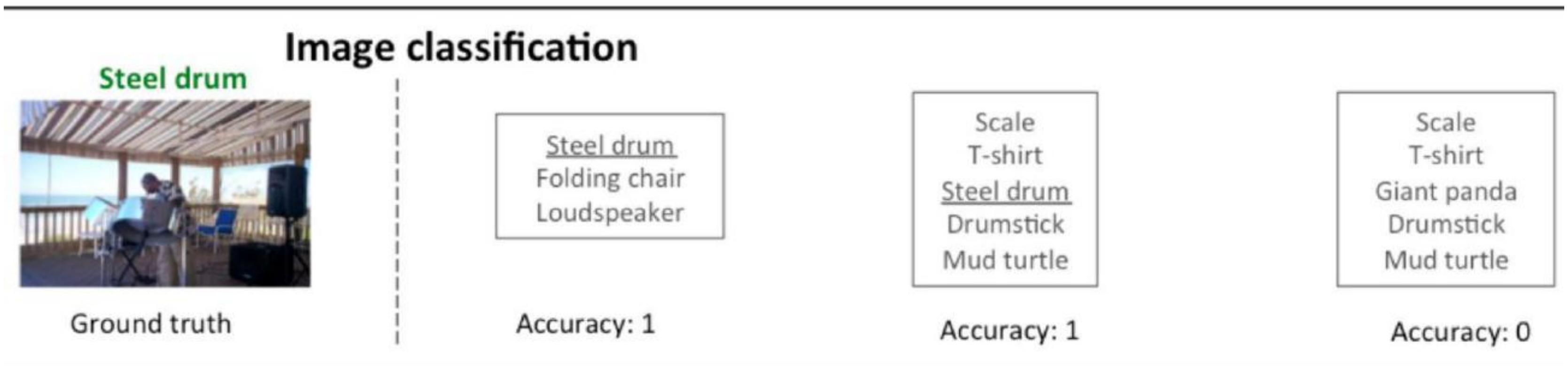
Dataset → • **ImageNet:** dataset of 14+ million images

Competition → • **ILSVRC:** ImageNet Large Scale Visual Recognition Challenge

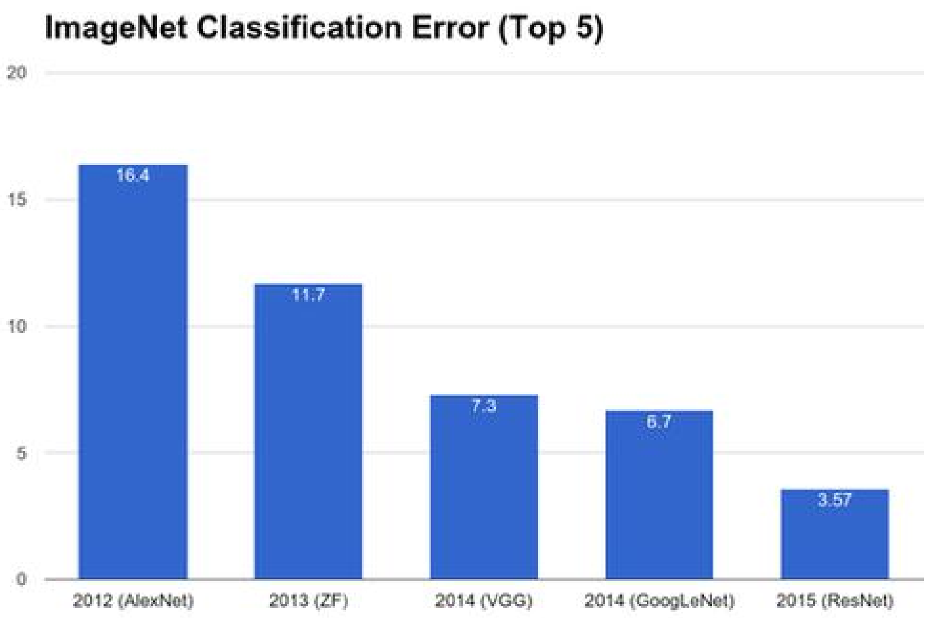
Networks → • AlexNet (2012)
• ZFNet (2013)
• VGGNet (2014)
• GoogLeNet (2014)
• ResNet (2015)
• CUImage (2016)

ILSVRC Challenge Evaluation for Classification

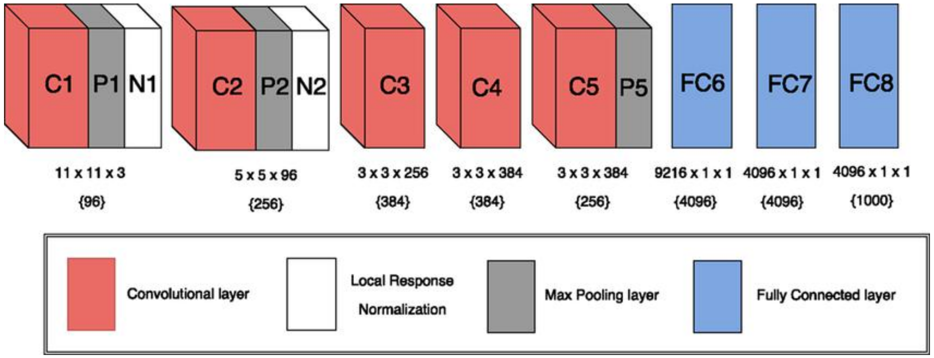
- Top 5 error rate:
  - You get 5 guesses to get the correct label



- ~20% reduction in accuracy for Top 1 vs Top 5
  - Example: In 2012 AlexNet achieved
- Human annotation is a binary task: “apple” or “not apple”



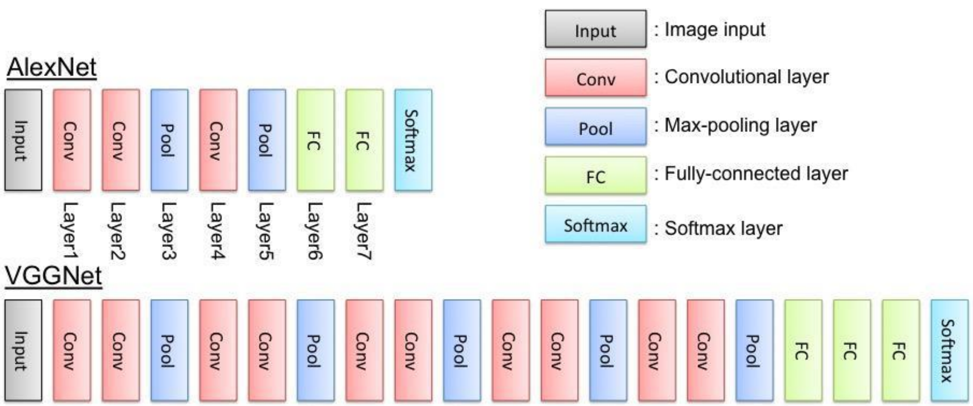
- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models



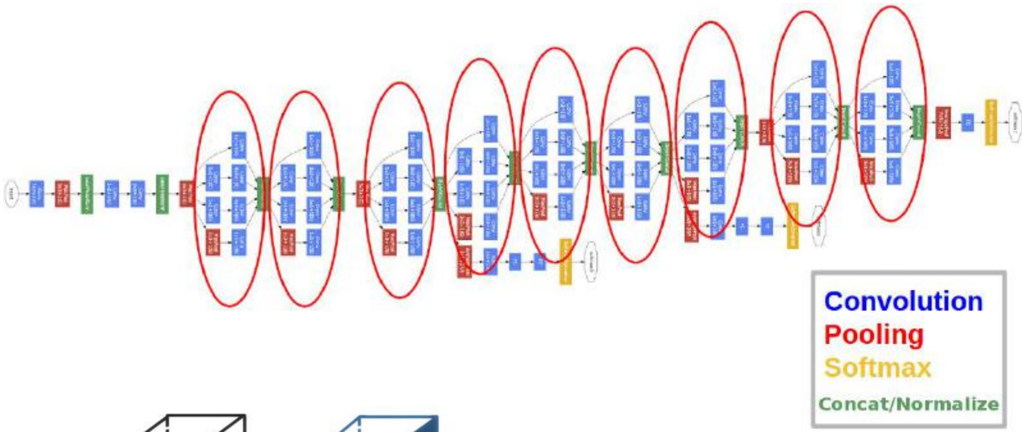
Krizhevsky et al. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

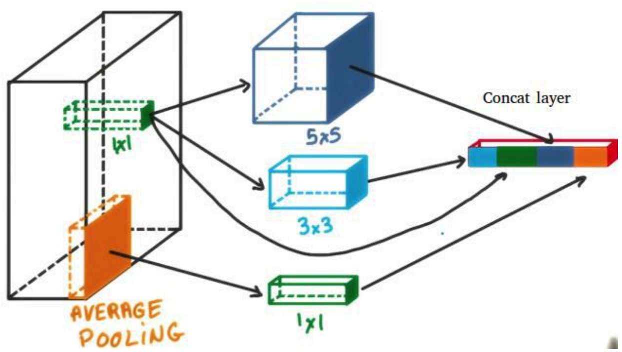
- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models

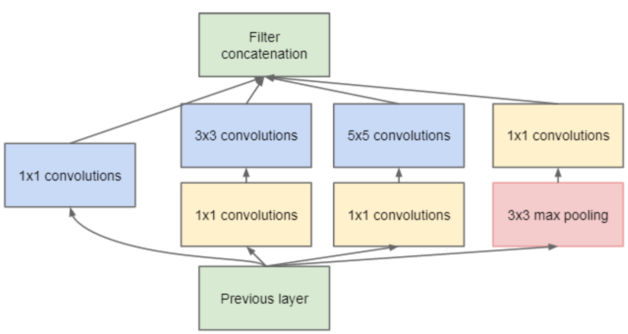
- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models



Simonyan et al. "Very deep convolutional networks for large-scale image recognition." 2014.

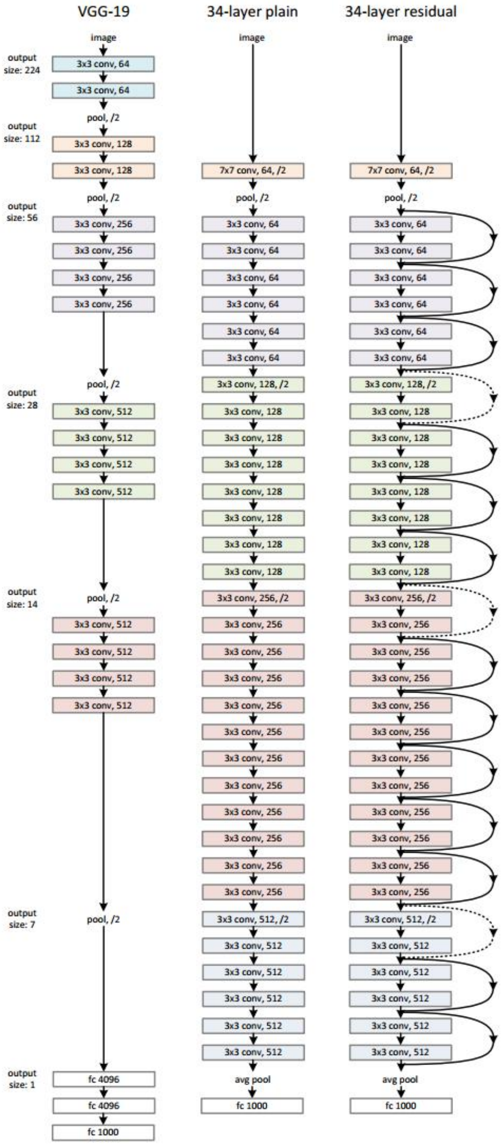


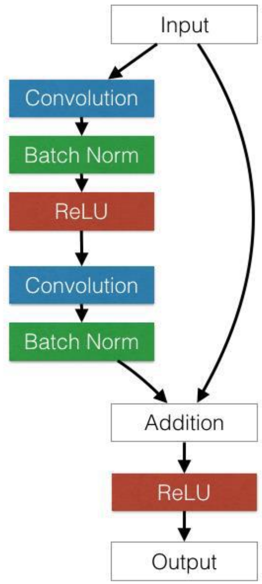




Szegedy et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform: 3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models

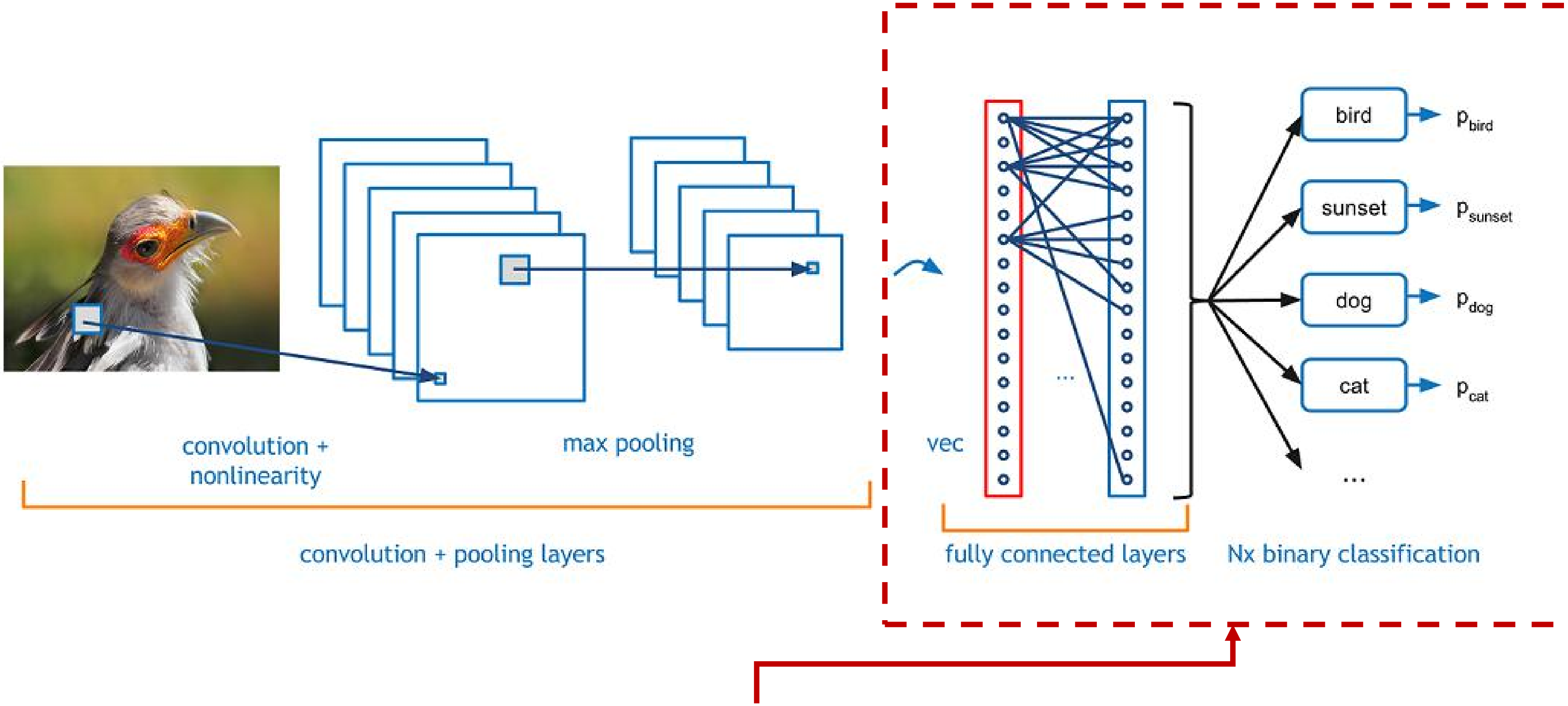




He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

- **AlexNet (2012): First CNN (15.4%)**
    - 8 layers
    - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
    - 8 layers
    - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
    - Beautifully uniform:
      3x3 conv, stride 1, pad 1, 2x2 max pool
    - 16 layers
    - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
    - Inception modules
    - 22 layers
    - 5 million parameters
      (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
    - More layers = better performance
    - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
    - Ensemble of 6 models

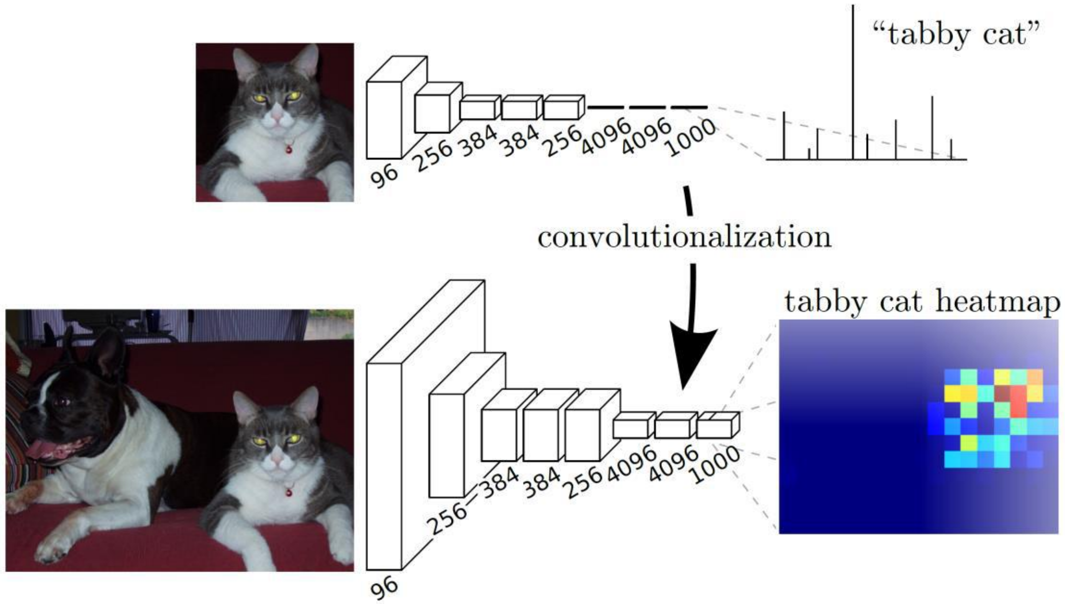
Same Architecture, Many Applications



This part might look different for:

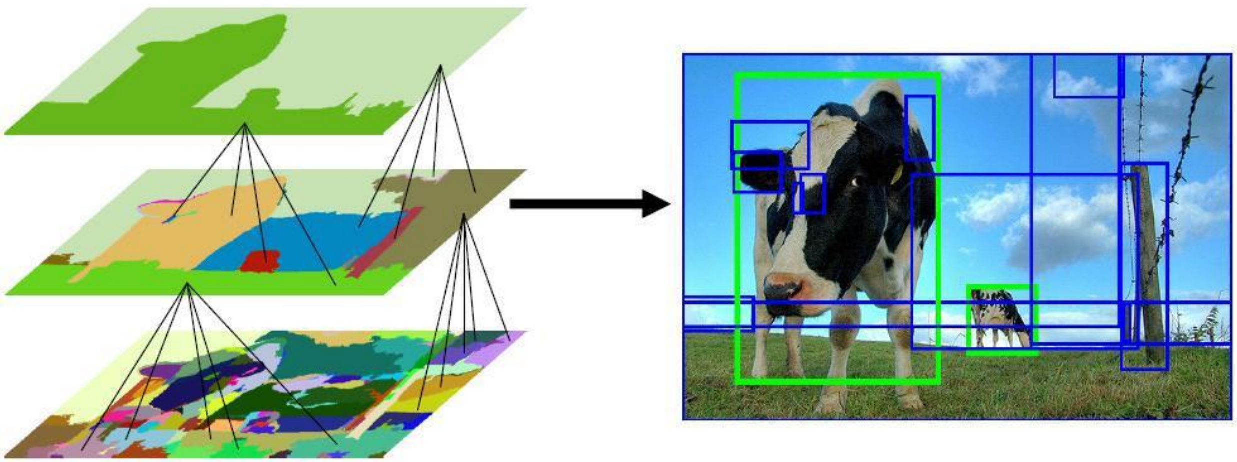
- Different image classification domains
- Image captioning with recurrent neural networks
- Image object localization with bounding box
- Image segmentation with fully convolutional networks
- Image segmentation with deconvolution layers

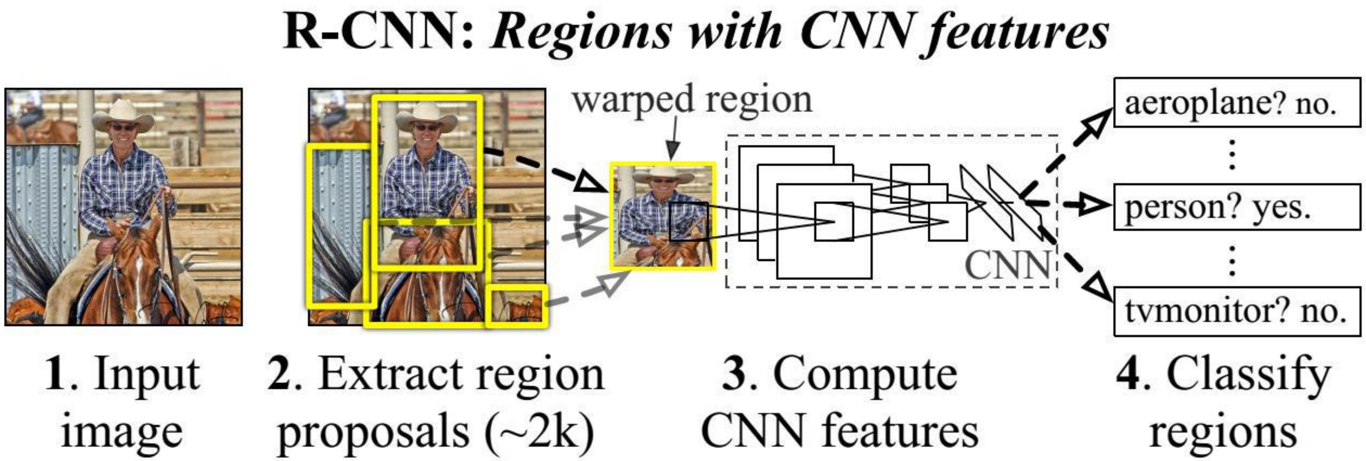
Segmentation



Original Ground Truth FCN-8

Object Detection

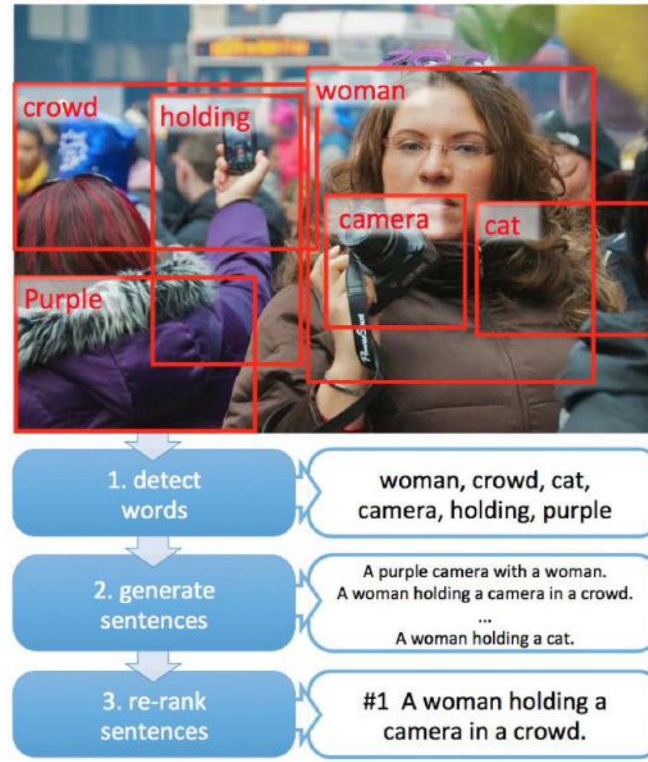




Applications: Image Caption Generation

a man sitting on a couch with a dog

a man sitting on a chair with a dog in his lap



dog (1.00) man (0.93) sitting (0.83) couch (0.66)

Applications: Image Question Answering

COCOQA 33827
What is the color of the cat?
Ground truth: black
IMG+BOW: black (0.55)
2-VIS+LSTM: black (0.73)
BOW: gray (0.40)

COCOQA 33827a
What is the color of the couch?
Ground truth: red
IMG+BOW: red (0.65)
2-VIS+LSTM: black (0.44)
BOW: red (0.39)

DAQUAR 1522
How many chairs are there?
Ground truth: two
IMG+BOW: four (0.24)
2-VIS+BLSTM: one (0.29)
LSTM: four (0.19)

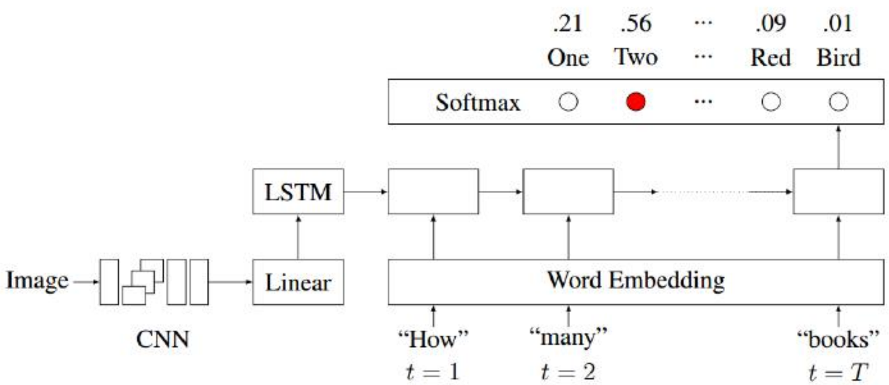
DAQUAR 1520
How many shelves are there?
Ground truth: three
IMG+BOW: three (0.25)
2-VIS+BLSTM: two (0.48)
LSTM: two (0.21)

COCOQA 14855
Where are the ripe bananas sitting?
Ground truth: basket
IMG+BOW: basket (0.97)
2-VIS+BLSTM: basket (0.58)
BOW: bowl (0.48)

COCOQA 14855a
What are in the basket?
Ground truth: bananas
IMG+BOW: bananas (0.98)
2-VIS+BLSTM: bananas (0.68)
BOW: bananas (0.14)

DAQUAR 585
What is the object on the chair?
Ground truth: pillow
IMG+BOW: clothes (0.37)
2-VIS+BLSTM: pillow (0.65)
LSTM: clothes (0.40)

DAQUAR 585a
Where is the pillow found?
Ground truth: chair
IMG+BOW: bed (0.13)
2-VIS+BLSTM: chair (0.17)
LSTM: cabinet (0.79)



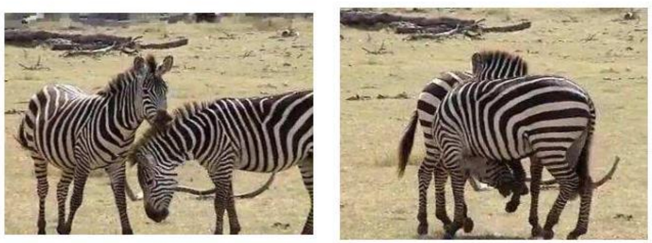
Ren et al. "Exploring models and data for image question answering." 2015.

Code: https://github.com/renmengye/imageqa-public

Applications: Video Description Generation

Correct descriptions.

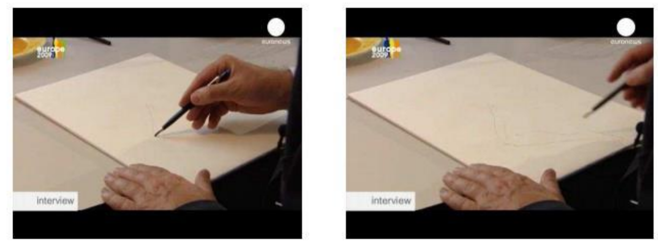
S2VT: A man is doing stunts on his bike.



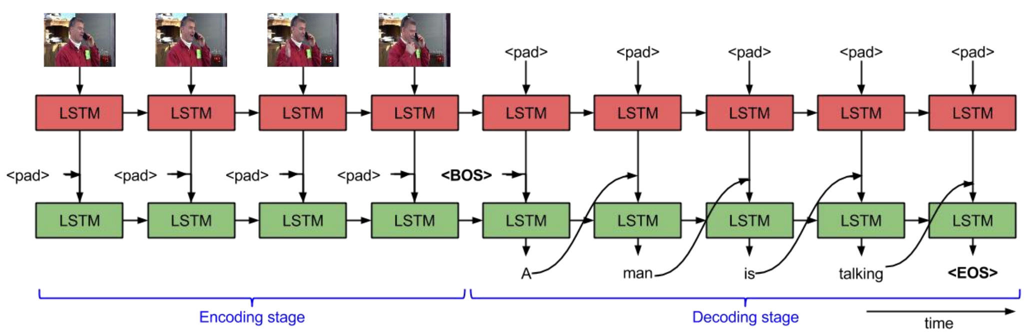
S2VT: A herd of zebras are walking in a field.

Relevant but incorrect descriptions.

S2VT: A small bus is running into a building.



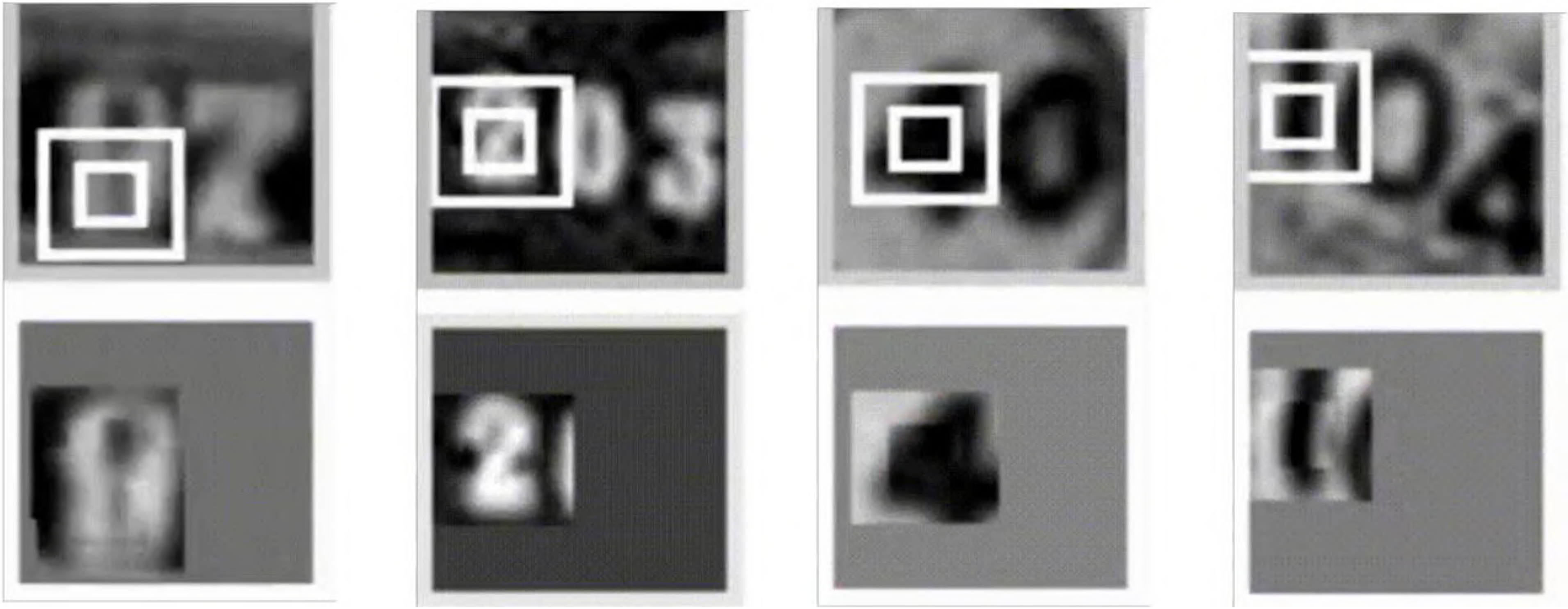
S2VT: A man is cutting a piece of a pair of a paper.

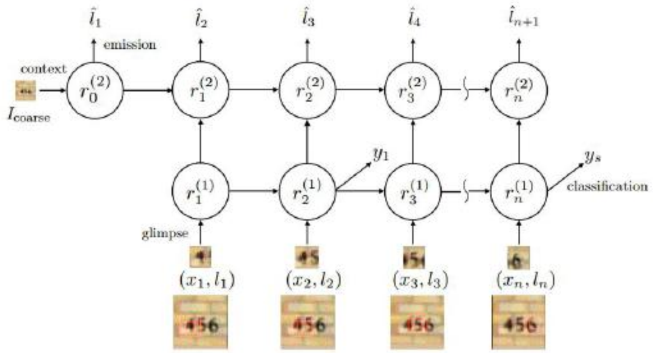


Venugopalan et al.
"Sequence to sequence-video to text." 2015.

Code: https://vsubhashini.github.io/s2vt.html

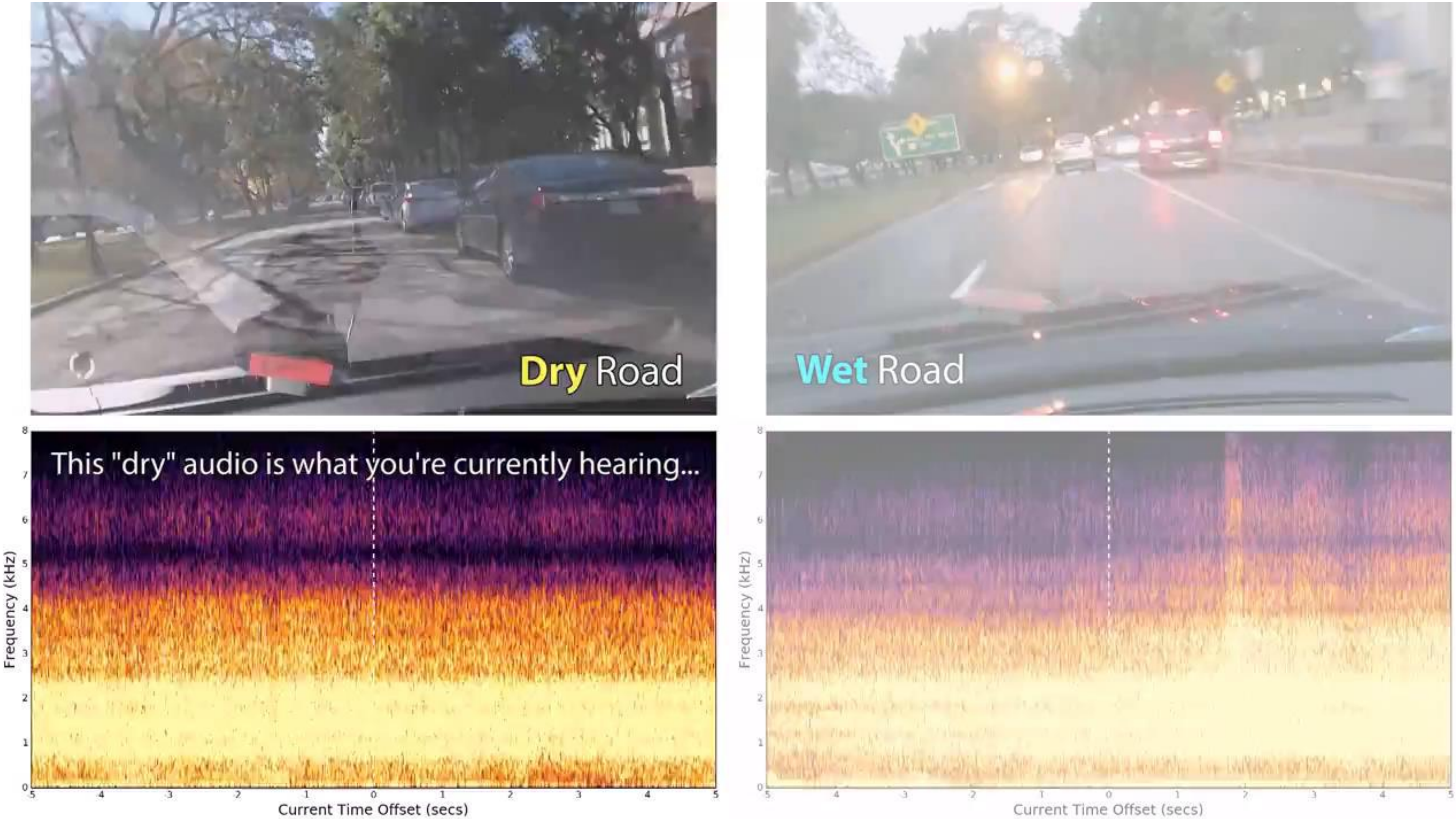
Applications: Modeling Attention Steering





Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. **"Multiple object recognition with visual attention."** (2014).

Application: Audio Classification



Driving Scene Segmentation

References: [127]

End-to-End Learning of the Driving Task

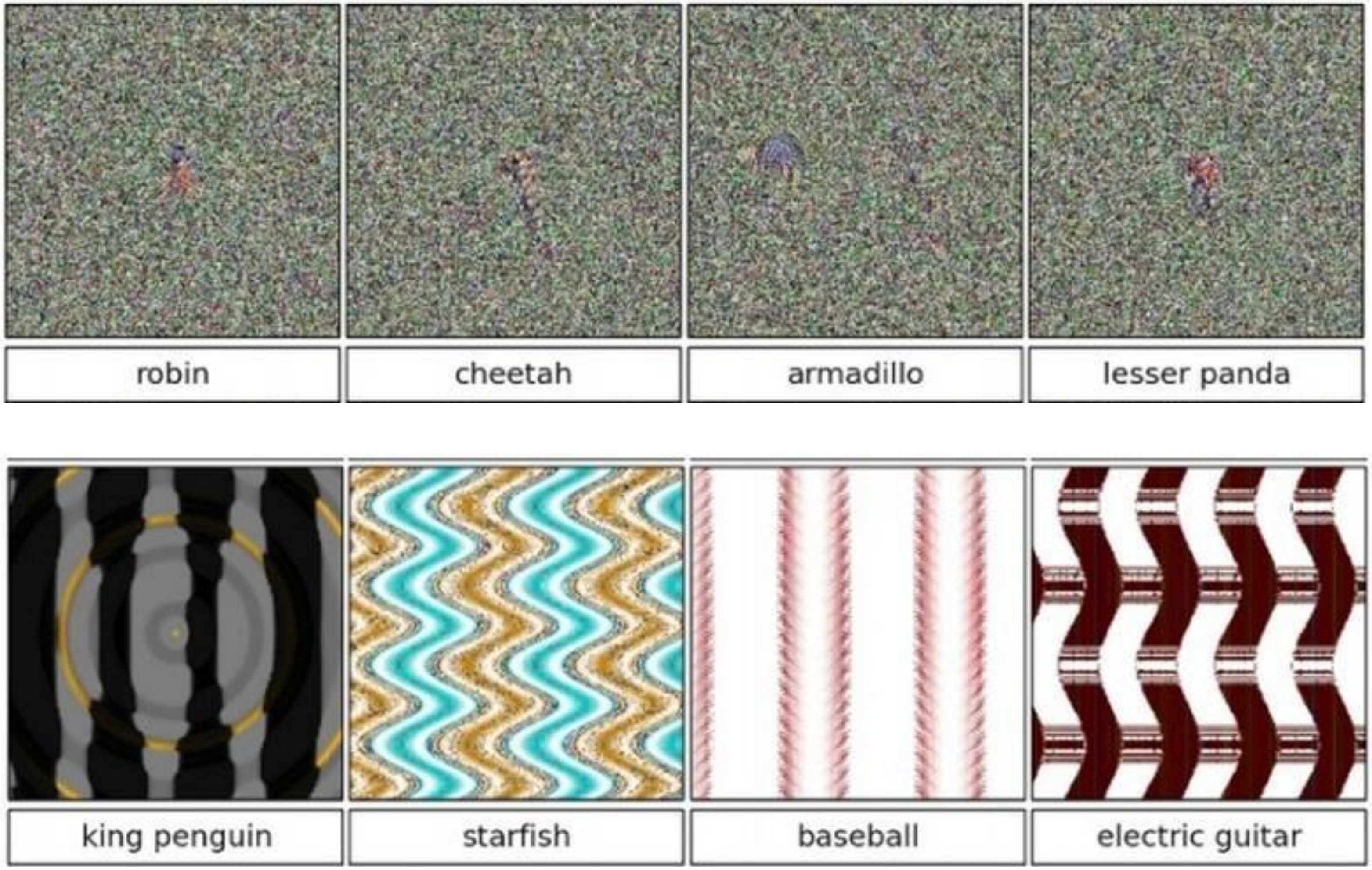
References: http://cars.mit.edu/deeptesla

Computer Vision for Intelligent Systems

3D Scene

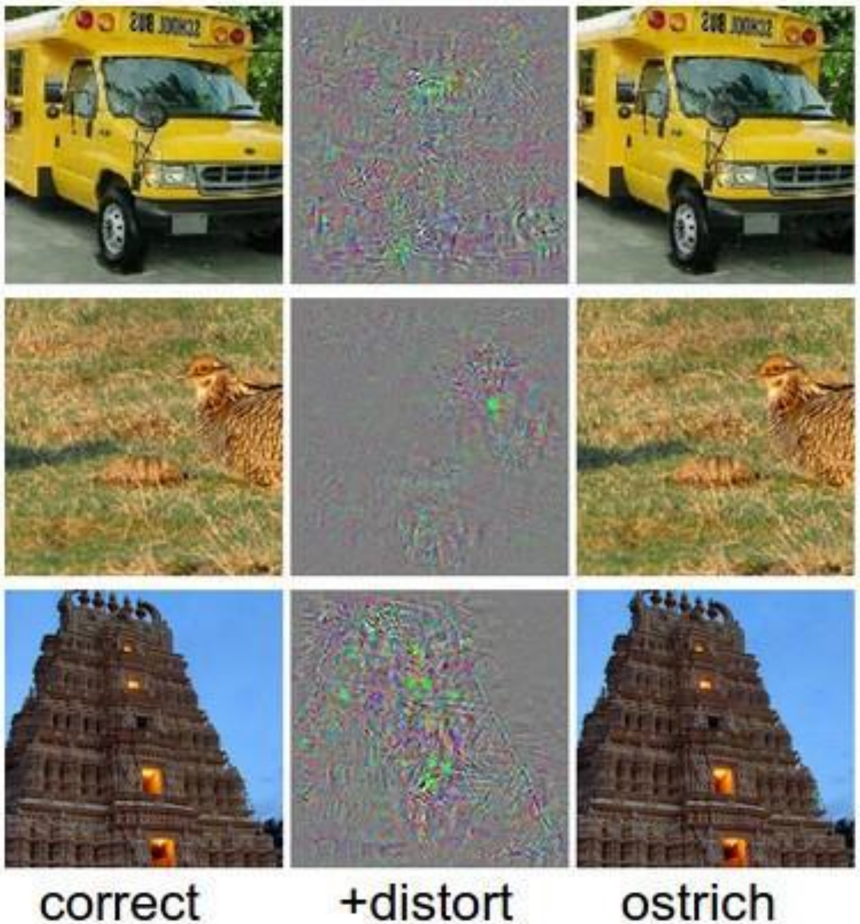
Feature Extraction	Texture	Color	Optical Flow	Stereo Disparity	
Grouping	Surfaces	Bits of objects	Sense of depth	Motion patterns	
Interpretation	Objects	Agents and goals	Shapes and properties	Open paths	Words
Action	Walk, touch, contemplate, smile, evade, read on, pick up, …				

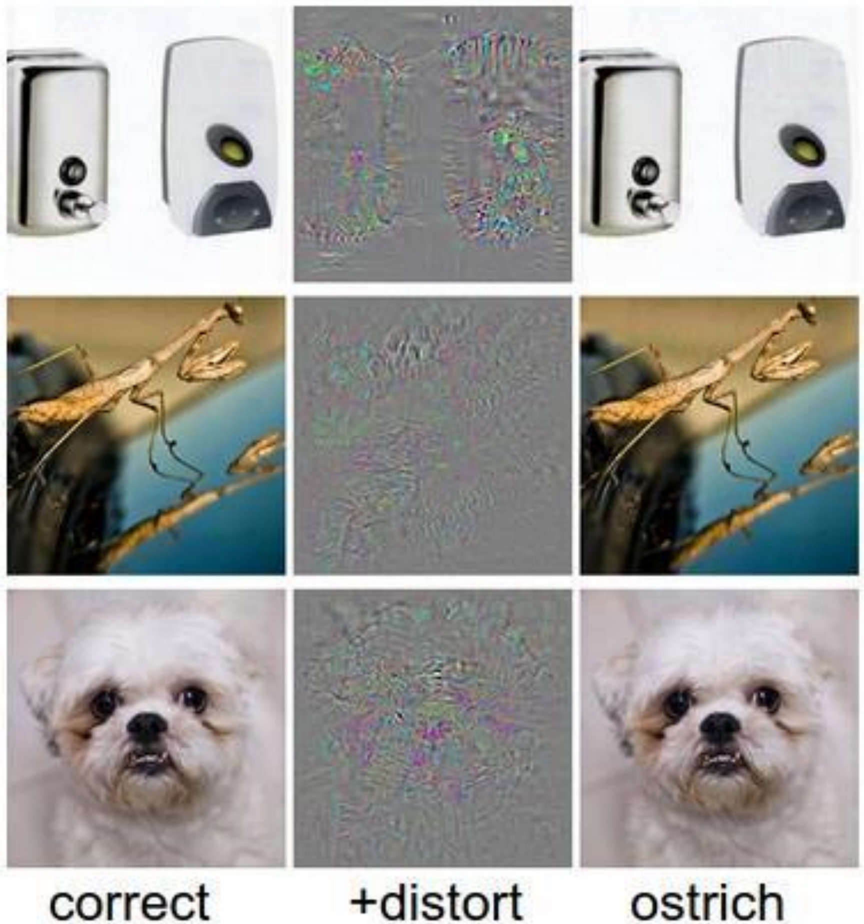
Open Problem: Robustness
>99.6% Confidence in the Wrong Answer



Nguyen et al. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." 2015.

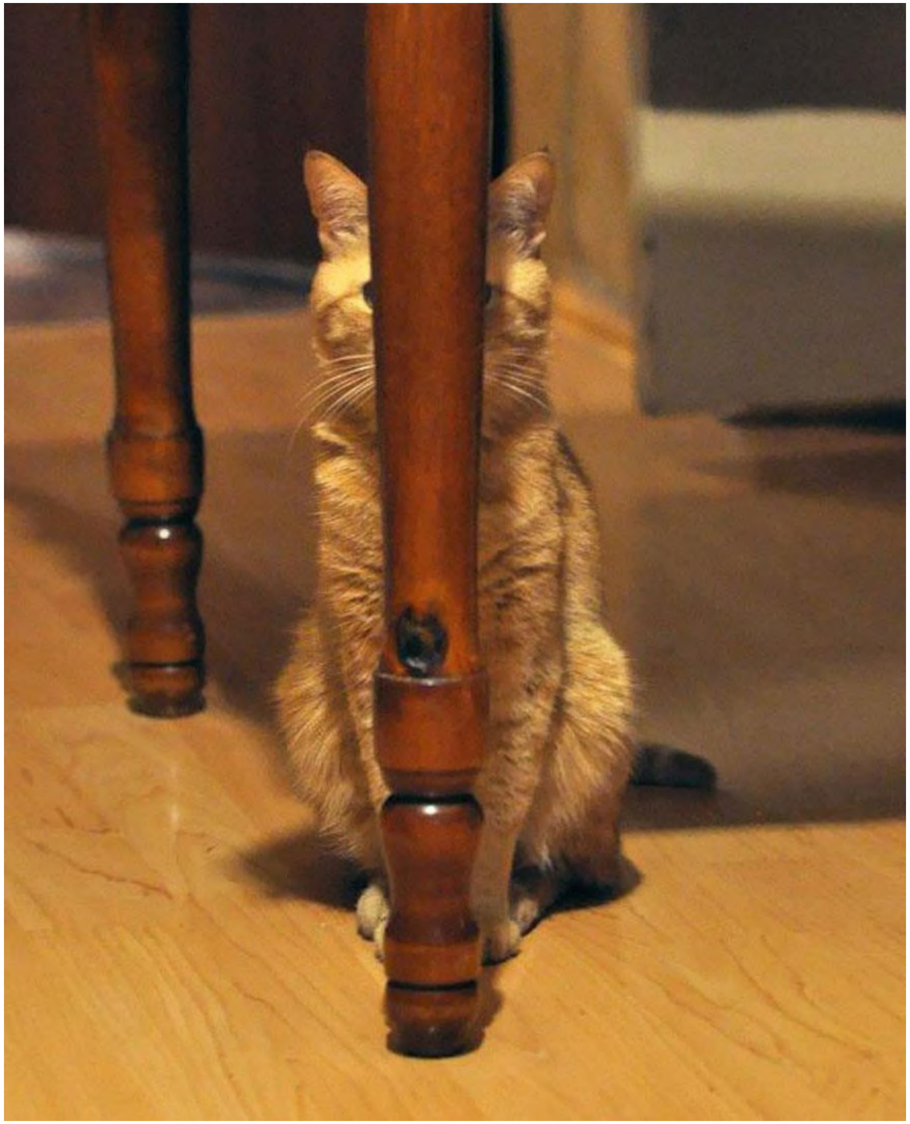
Open Problem: Robustness
Fooled by a Little Distortion





Szegedy et al. "Intriguing properties of neural networks." 2013.

Object **Cat**egory Recognition



Object **Cat**egory Recognition

Object **Cat**egory Recognition

Object Category Recognition

Object **Cat**egory Recognition

References

All references cited in this presentation are listed in the following Google Sheets file:

https://goo.gl/9Xhp2t