Convolutional Neural Networks

Zachary Lipton & Henry Chai

10701 — October 25th

email: zlipton@cmu.edu

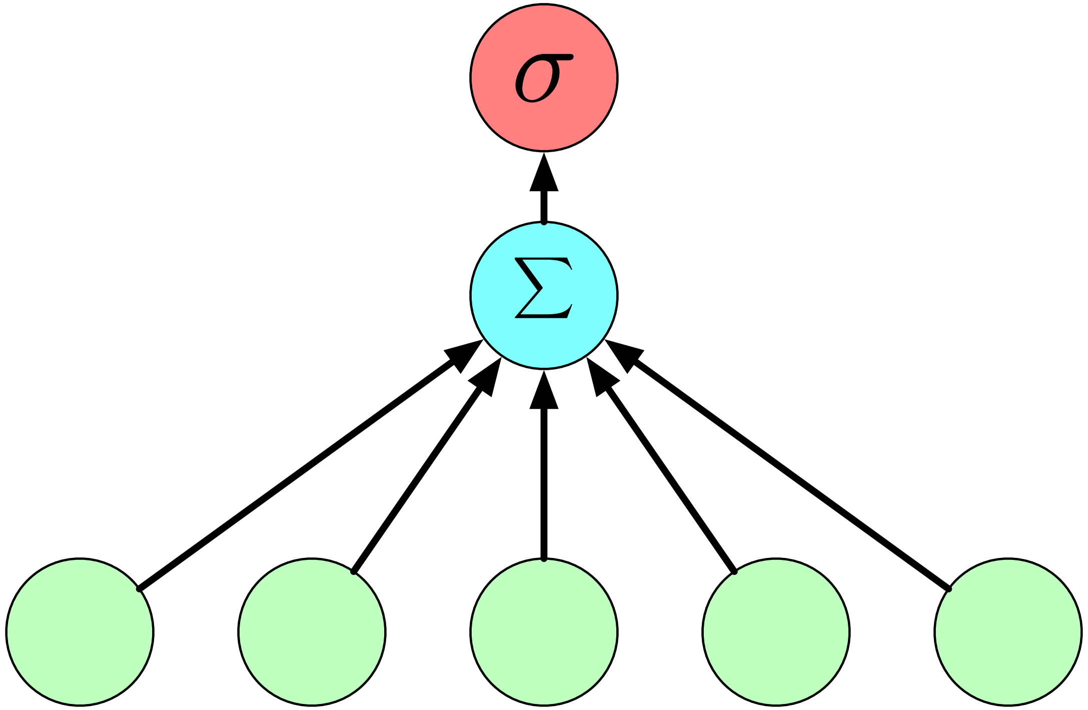
MACHINE LEARNING DEPARTMENT

Acknowledgments & Attributions

- Stanford 231n: Li, Karpathy, Johnson, Yeung
- MIT 6874: Manolis Kelis
- Dive into Deep Learning (Lipton, Zhang, Smola, Li) http://d2l.ai/
- Pretty pictures: OpenAI’s DALL-E 3 (accessed via ChatGPT(4))

Neural Networks Refresher

- Input features
- Architecture
  - Hidden layers
  - Pattern of connectivity
  - Activation functions
- Output layer
- Loss function
- Optimization algorithm
- Evaluation strategy



Kind of Task → Output Layer, Loss, Post-proc

- What choices would we make for binary classification?
- Multiclass classification?
- Multilabel classification?
- Scalar regression?
- Predicting x,y coordinates?
- Ranking?
- Matching?
- Predict a set?
- Classification with cost sensitivity?

Kind of Data → Representation, Architecture

- **Images:** Pixel Data → **CNNs**, Visual Transformers
- **Audio:** Raw wave form / STFTs → RNNs or Transformers
- **Natural Language:** Token Encodings / Embeddings → Transformers
- **Social Media or Molecular Data:** Graph Neural Networks


History of Vision

Cambrian Explosion
(530-545 Million Years Ago)

- Massive explosion in biodiversity
- Results in most major animal groups alive today
- Evolution of eye believed to have been a catalyst
    - Predators could suddenly locate and go after prey
    - Intense competition for prey
    - Intense competition to escape predators

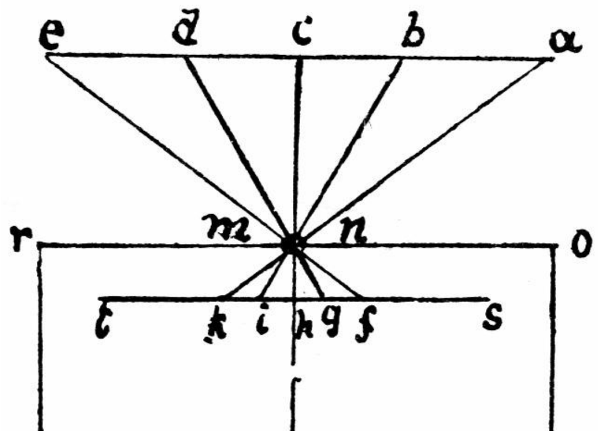
Source: British Natural History Museum, learned via Fei-Fei Li

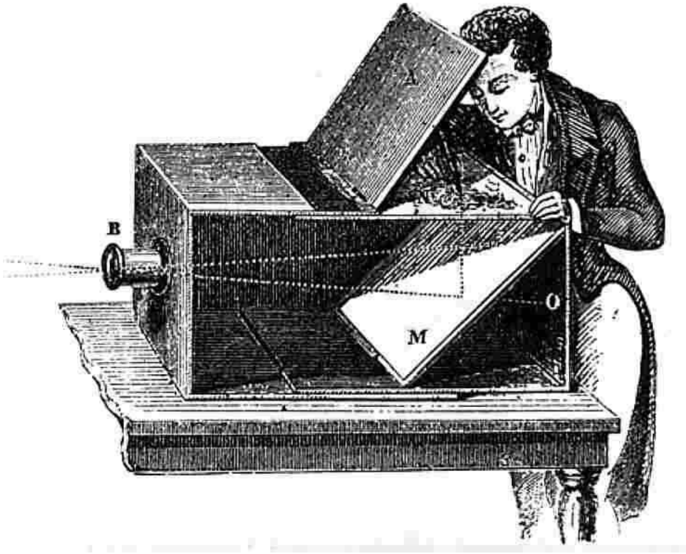
Primacy of Vision in Human Cognition

- Over 50% of neurons in neural cortex involved in visual processing
- Far the largest sensory system
- Cornea and lens shine (small) image onto retina
- Retina transduces image into electrical signals using:
  - Rods (night vision, more sensitive)
  - Cones (three varieties, responsible for color perception)
- Visual cortex (somewhat) hierarchically organized
- Optic nerve fibers → LGN → V1–V5 (occipital lobe)

Pre-Photography: Camera Obscura

- Pinhole camera—image projected thru small hole or lens onto a wall
- Possible inspiration for prehistoric art
- Described by Aristotle (322 BC), Euclid (in Optics)
- Described by Leonardo Da Vinci (1502)
- Used to study eclipses, sunspots
- Aid in drawing

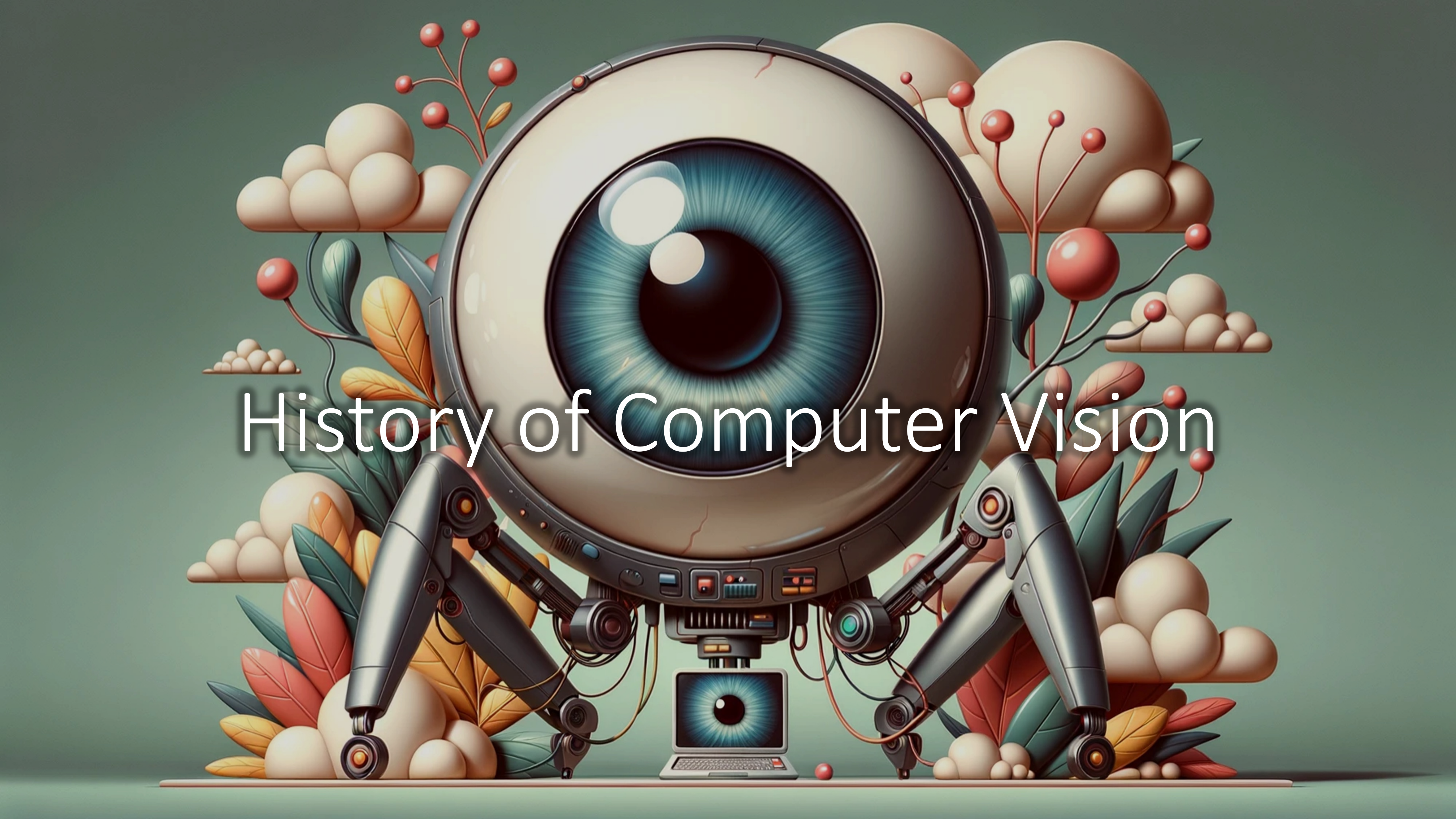




Photography

- 1826: Nicéphore Niépce captures an image (days of exposure)
- 1839: Metal-based daguerreotype process, birth of practical photos
- 1839: Paper-based negatives
- 1888: Kodal releases first hand-held camera, w preloaded film
- 1890s: First color photographs
- 1948: Polaroid introduces first instant camera
- 1990s: Commercial introduction of computer-based digital cameras
- 2023: More cameras than people, video = majority of bits of all data
  (the dark matter of the internet)

History of Computer Vision



Block World — Larry Roberts (1963)

- First PhD in computer vision
- Inspired by human ability to reconstruct 3D scenes from 2D images

(Roberts subsequently architected ARPANET)

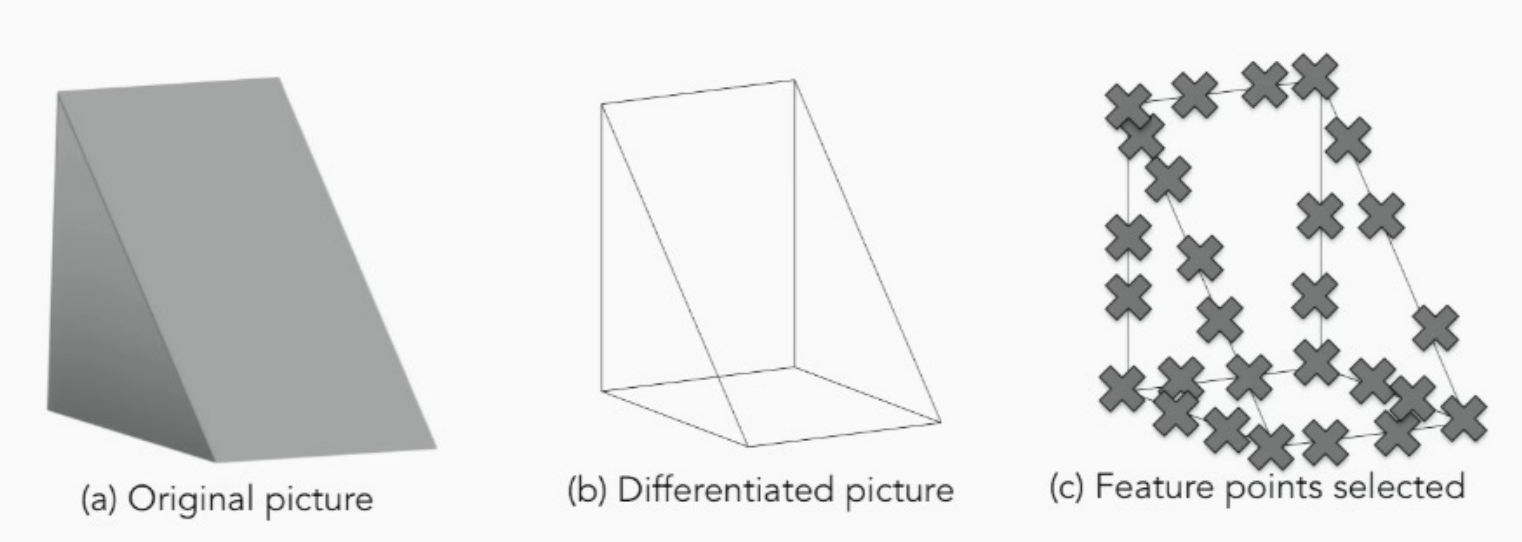


Image from http://cs231n.stanford.edu/schedule.html

Seymour Paper “Summer Vision Project”

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence Group July 7, 1966
Vision Memo. No. 100.

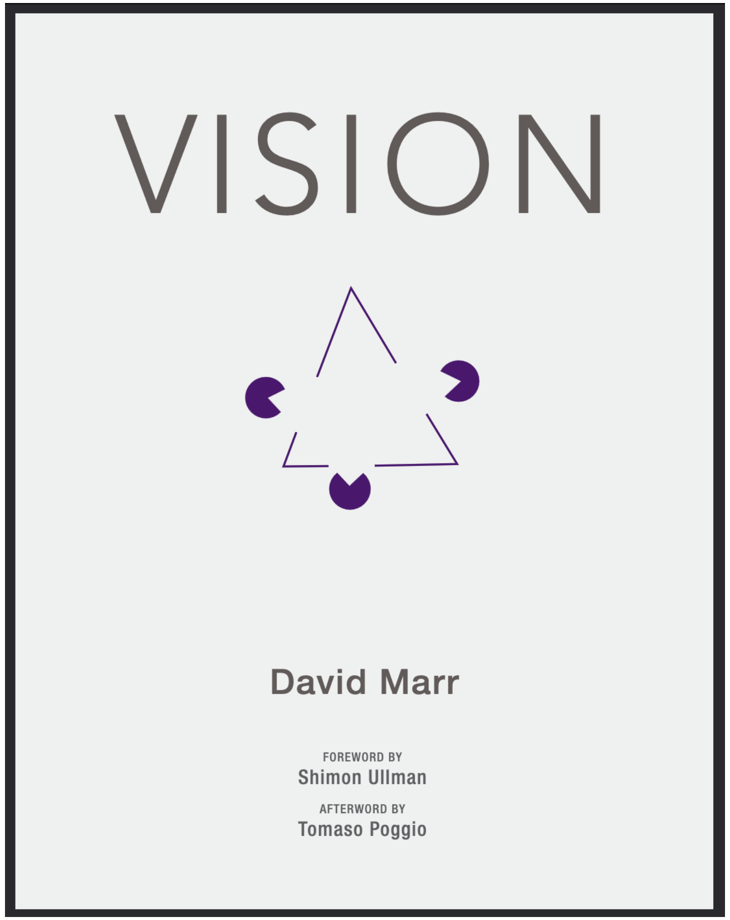
THE SUMMER VISION PROJECT

Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

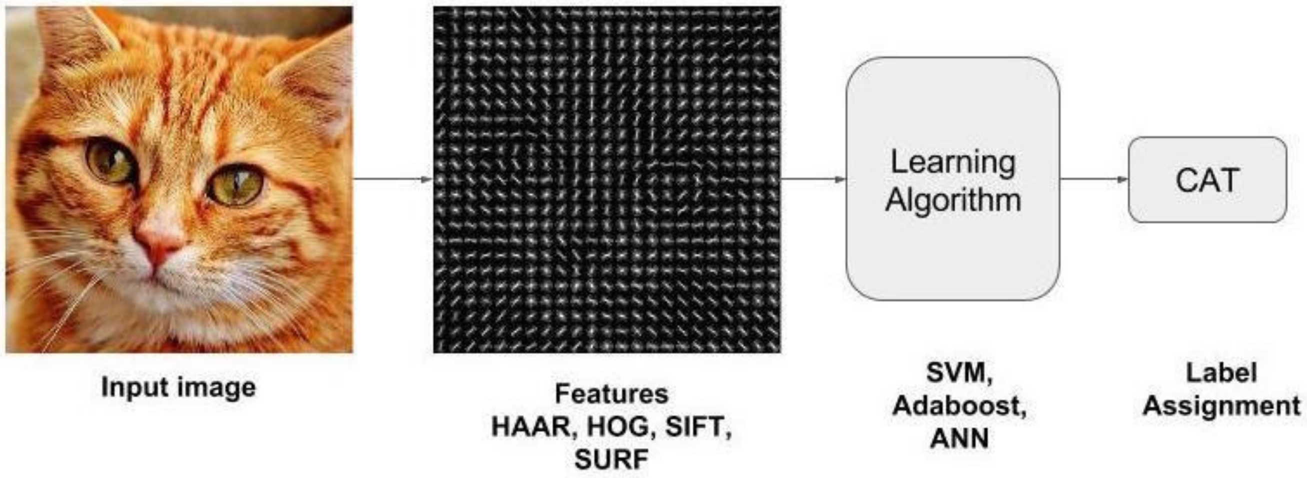
David Marr “Vision” 1982

- Input image
  (raw inputs)
- → Primal Sketch
  (blobs, edges, bars, lines, curves)
- → 2.5D Sketch
  (surface orientation, depth info, discontinuities)
- 3D Models
  (hierarchical model, volumetric primitives)

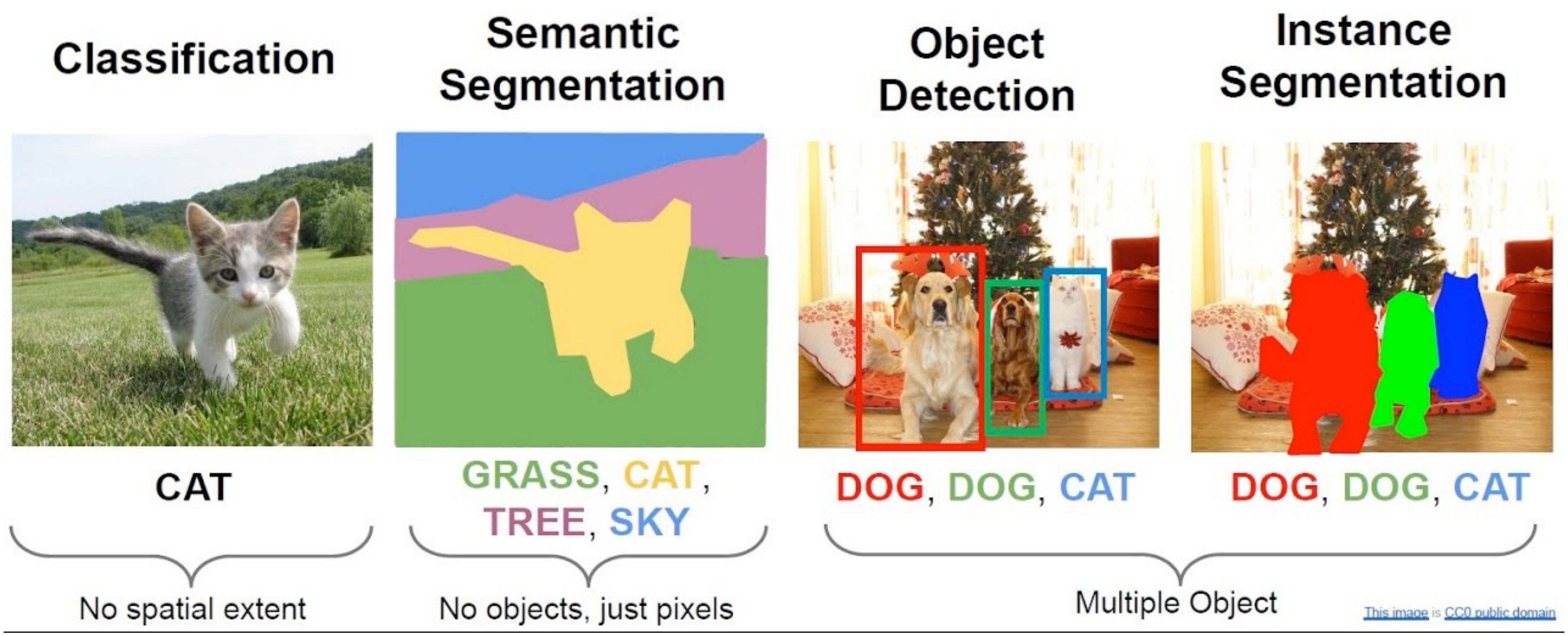


Early 2000s — emergence of ML-based vision

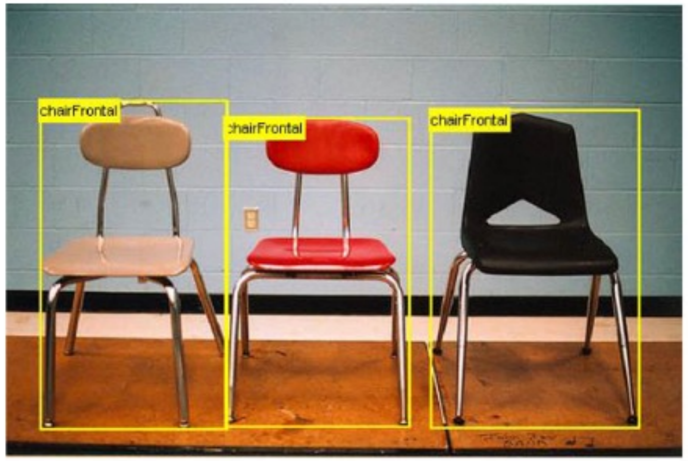
- SIFT features ("Scale-invariant feature transform")
  - Local and invariant to scale, rotation
  - Based on convolving images with Gaussian kernels
- Fed as input to ML classifiers
  - Popular choices: Adaboost & Support Vector Machines (SVMs)

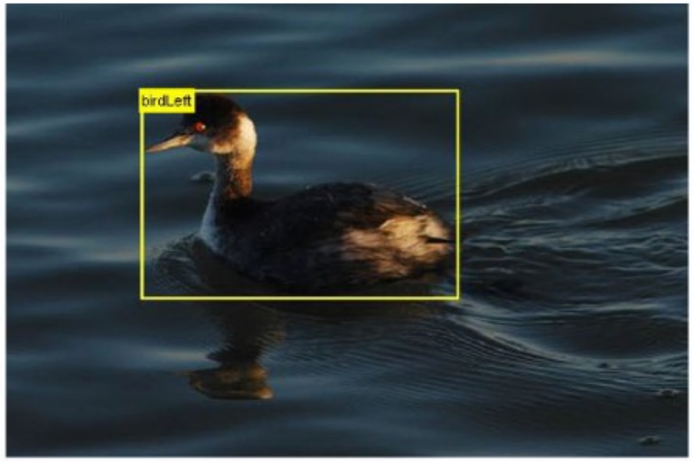


Canonical Image Tasks



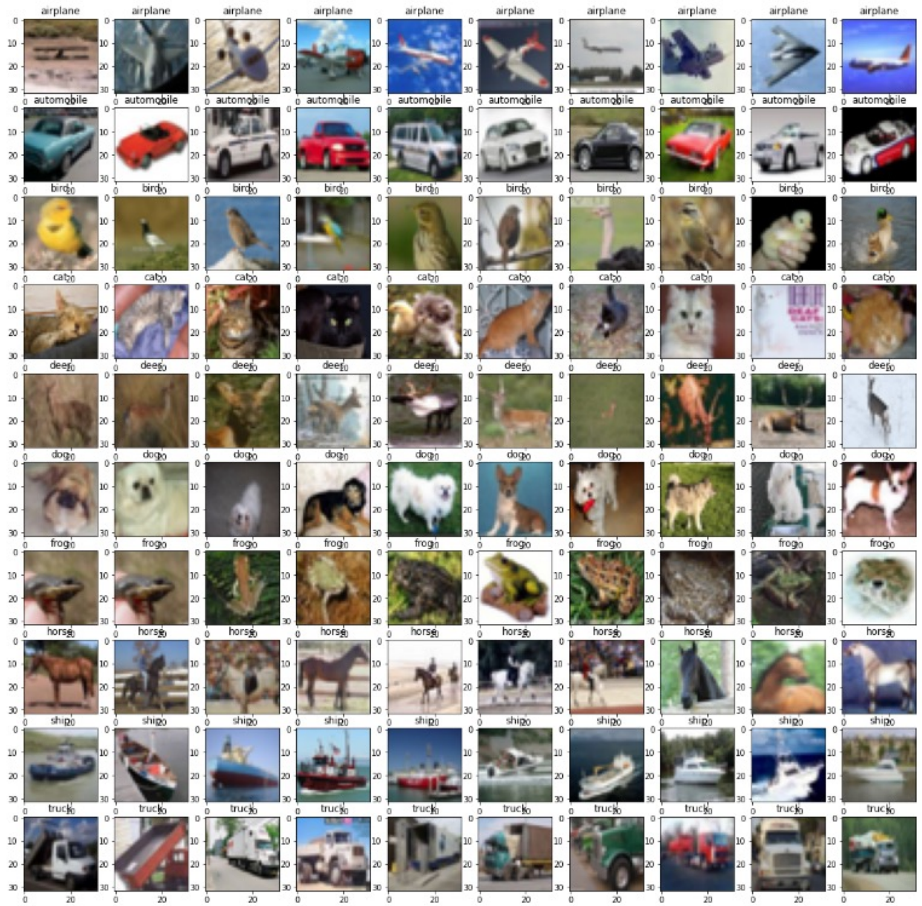
PASCAL Visual Object Classes (20 classes, 20k images)





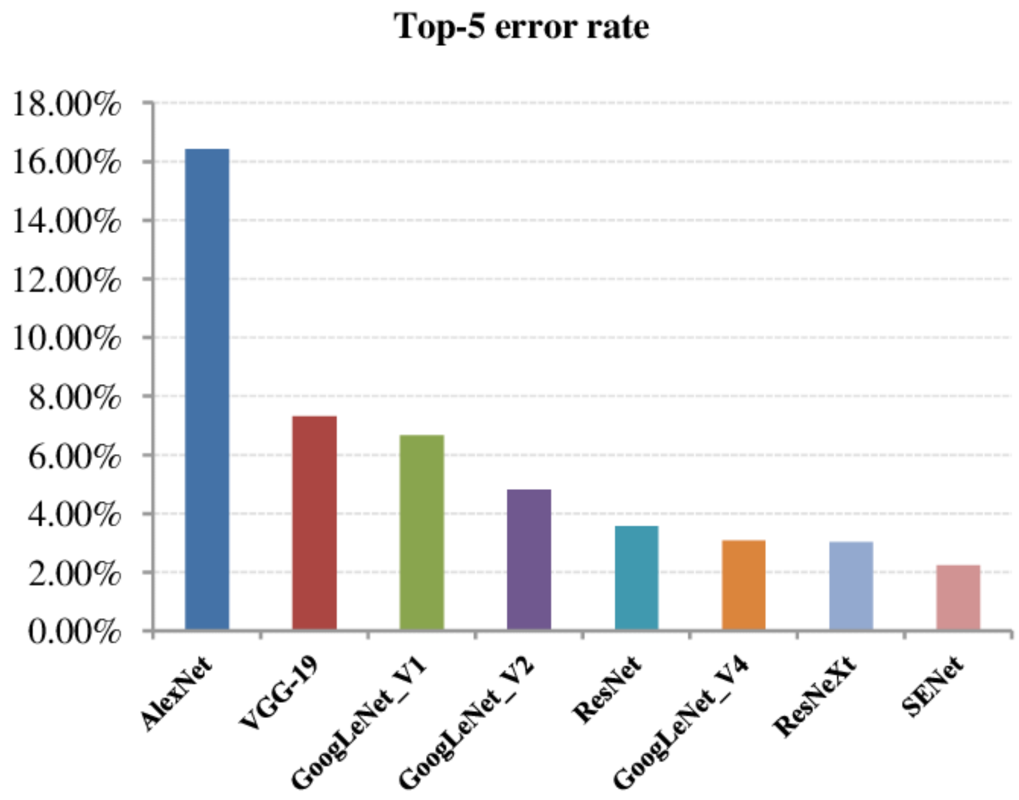
CIFAR 10 (& 100) Datasets

- CIFAR 10
  - 60k 32x32 color images
  - 10 classes (6k each)
  - 50k in train set, 10k in test set
- CIFAR100:
  - 60k 32x32 images
  - 1000 classes, (600 images each)
  - Grouped into 20 superclasses

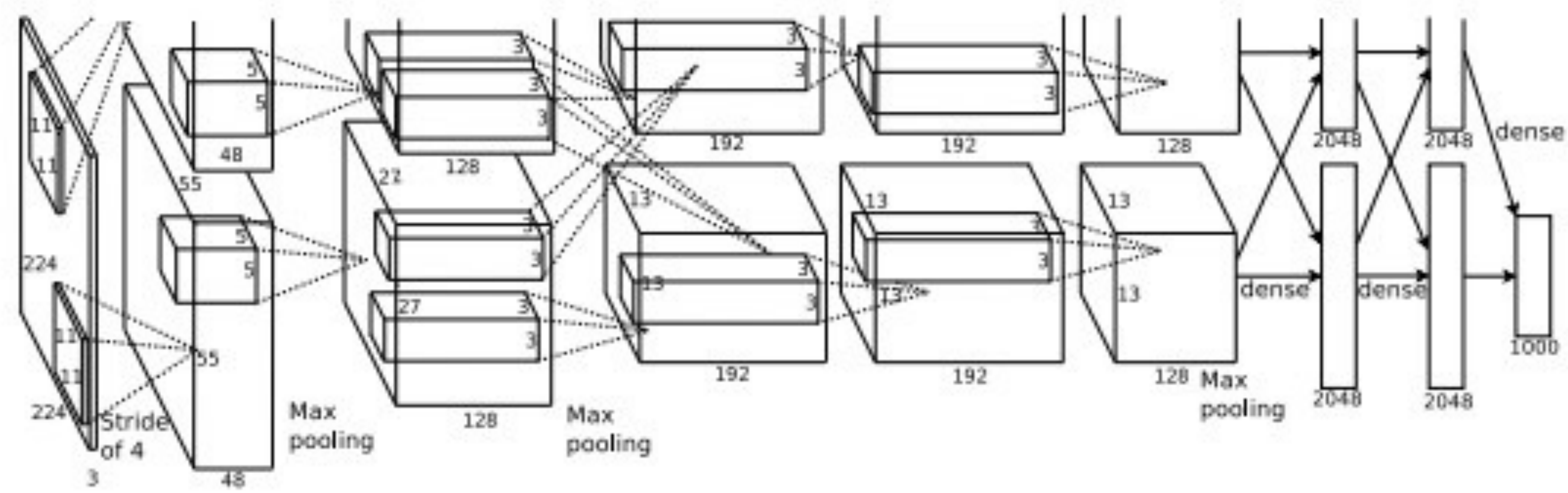


ImageNet Challenge

- Launched in 2009
- Collected images against WordNet hierarchy
- Sourced from Google, MSN, Yahoo!, Flickr
- Crowdsourcing to confirm labels
- 22k categories, 14M images



Convolutional Neural Network Architectures



[“AlexNet” — (Krizhevsky, Sutskever, Hinton 2012)](#)

ConvNetJS

- https://cs.stanford.edu/people/karpathy/convnetjs/

What do images look like to a computer?

54	42	48	36	7	78	42	21	44	35	15	28	7	80
97	33	60	38	96	15	2	90	13	7	93	45	87	85
81	48	67	66	88	22	79	99	87	83	73	40	66	96
31	49	58	85	80	31	51	99	36	5	57	81	57	75
21	55	65	17	59	15	20	19	88	74	0	27	26	35
55	75	37	13	46	70	42	35	13	98	35	78	92	27
52	60	81	38	56	56	79	89	6	43	71	67	24	66
33	22	71	12	56	15	0	79	46	17	87	17	15	88
11	31	33	78	54	78	70	43	55	24	84	49	89	76
52	66	93	53	9	33	23	51	23	90	27	98	74	82
17	7	24	25	96	31	3	67	78	61	96	86	99	12
86	55	81	70	7	61	48	39	13	64	38	37	40	93
84	24	70	29	21	34	41	82	9	43	77	74	58	91
69	17	38	15	32	46	9	60	66	21	7	58	25	97

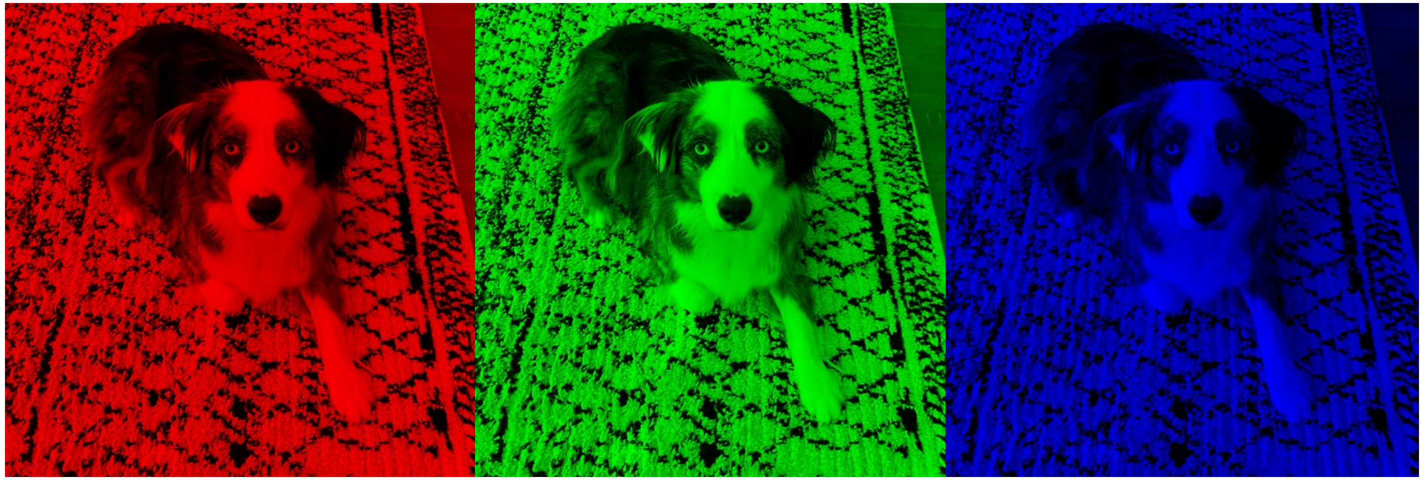
What do Color Images, to a Computer?

54	42	48	36	7	78	42	21	44	35	15	28	7	80
97	33	60	38	96	15	2	90	13	7	93	45	87	85
81	48	67	66	88	22	79	99	87	83	73	40	66	96
31	49	58	85	80	31	51	99	36	5	57	81	57	75
21	55	65	17	59	15	20	19	88	74	0	27	26	35
55	75	37	13	46	70	42	35	13	98	35	78	92	27
52	60	81	38	56	56	79	89	6	43	71	67	24	66
33	22	71	12	56	15	0	79	46	17	87	17	15	88
11	31	33	78	54	78	70	43	55	24	84	49	89	76
52	66	93	53	9	33	23	51	23	90	27	98	74	82
17	7	24	25	96	31	3	67	78	61	96	86	99	12
86	55	81	70	7	61	48	39	13	64	38	37	40	93
84	24	70	29	21	34	41	82	9	43	77	74	58	91
69	17	38	15	32	46	9	60	66	21	7	58	25	97

Multiple Input Channels

- Color images typically have three channels (RGB)
- Converting to grayscale loses information

=



Why not apply (k–)Nearest Neighbor?

- Where does the distance function come from?
- Shift an image X by two pixels to get X’
- The distance (Euclidean, Manhattan) $|X - X'|$ can be enormous!

Why not apply linear models?

- Nothing special about any pixel location
- Why should any weight be different than any other weight?
- An image and its inverse depict the same object!

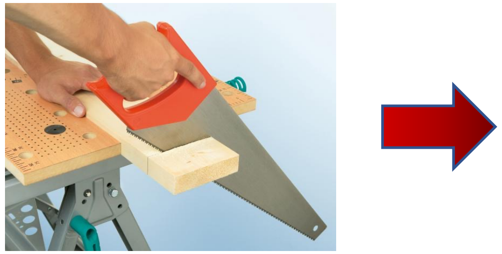
“The Semantic Gap”

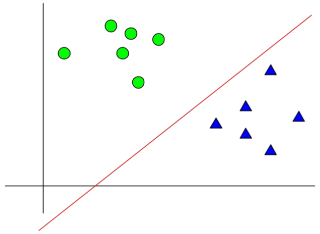
- Massive conceptual difference in abstraction between pixel and label
- Same object can come in different sizes, shapes, locations, colors, etc.
- Even the very same photograph could look wildly different at the pixel level (due to compression artifacts, filters, cropping)

Why Representation Learning?

Classical prediction pipeline

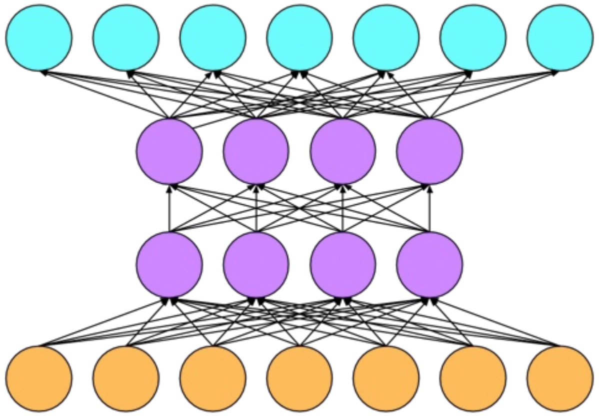
- **Hand-engineer features**
- Use **prior knowledge (or hacks)**
- Feed features to simple ML model





Deep learning pipeline

- **Learn the features** and the classifier **jointly**
- **Discover** interactions and non-linear relationships

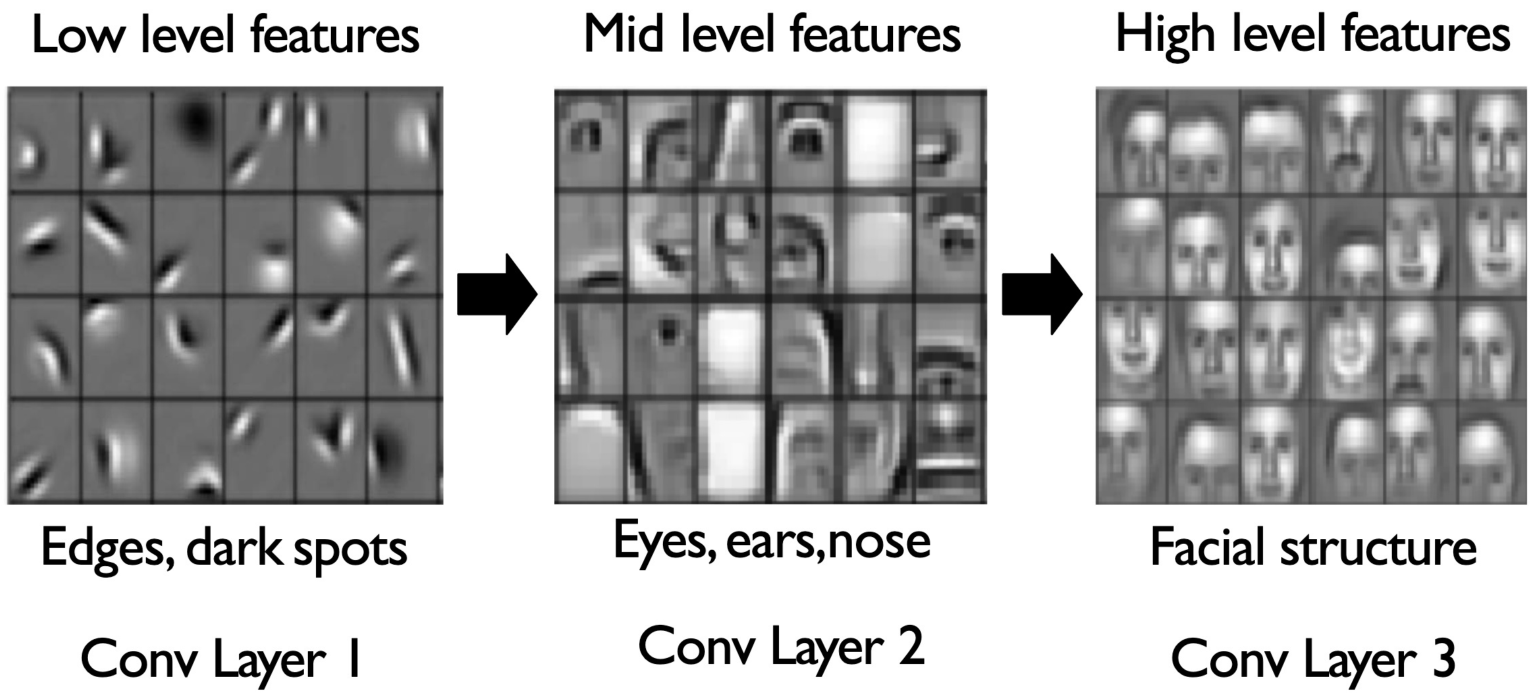


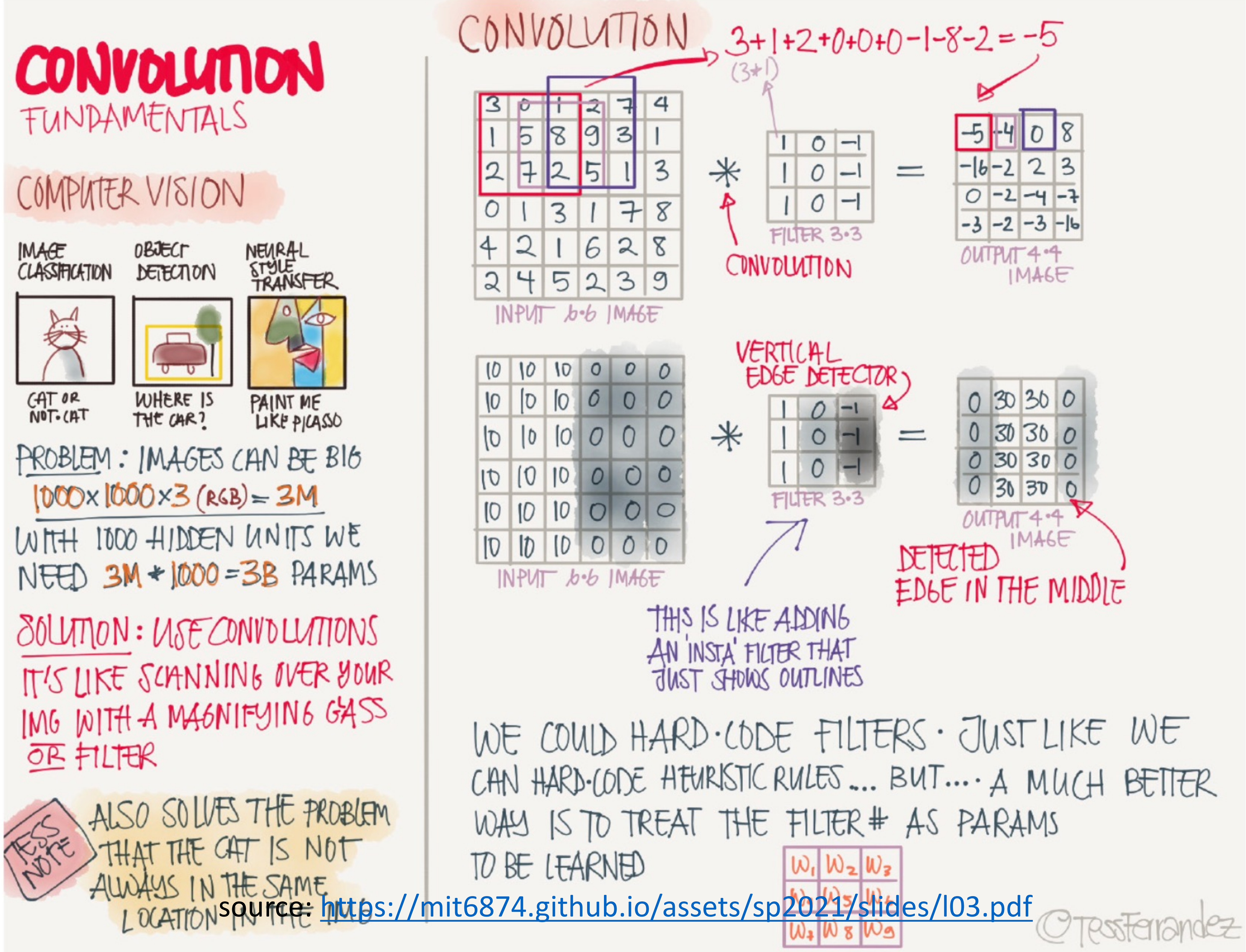
Why not classify images with MLPs?

- Suppose we wish capture 1000x1000 pixel color images
- How many input neurons would we need?
- Suppose we wish to preserve dimensionality in first hidden layer
- How many weights would we need?

Key Intuitions behind Convolutional Layers

- Our "internal representations" of preserve spatial structure
- Hierarchically arranged to bridge semantic gap, cartoon:





source: https://mit6874.github.io/assets/sp2021/slides/l03.pdf

Convolutions

Two Principles

- Translation Invariance
- Locality

From Dense Layers to Convolutional Layers

- Shape inputs and outputs as matrices (width, height)
- Shape weights as a giant 4D tensor (h,w) to (h',w')

$$h_{i,j} = \sum_{k,l} w_{i,j,k,l} x_{k,l} = \sum_{a,b} v_{i,j,a,b} x_{i+a,j+b}$$

V is re-indexes W such as that

$$v_{i,j,a,b} = w_{i,j,i+a,j+b}$$

Idea #1 - Translation Invariance

$$h_{i,j} = \sum_{a,b} v_{i,j,a,b} x_{i+a,j+b}$$

- A shift in *x* also leads to a shift in *h*
- *v* should not depend on (*i,j).* Fix via

$$v_{i,j,a,b} = v_{a,b}$$

$$h_{i,j} = \sum_{a,b} v_{a,b} x_{i+a,j+b}$$

That’s a cross-correlation

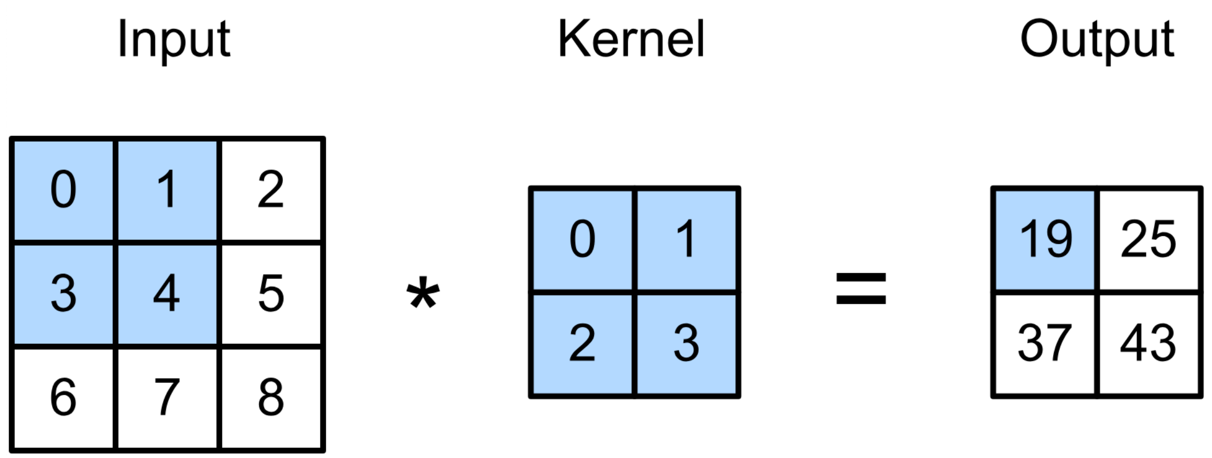
Idea #2 - Locality

$$h_{i,j} = \sum_{a,b} v_{a,b} x_{i+a,j+b}$$

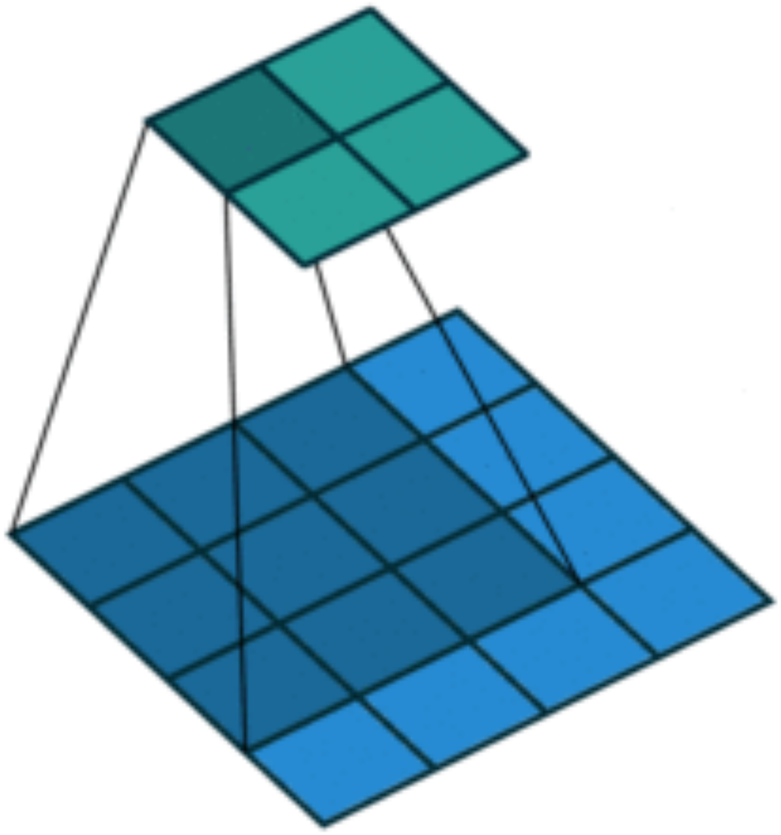
- We shouldn’t look very far from x(i,j) in order to assess what’s going on at h(i,j)
- Outside range $|a|,|b| > \Delta$ parameters vanish $v_{a,b} = 0$

$$h_{i,j} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} v_{a,b} x_{i+a,j+b}$$

2-D Cross Correlation



$$
\begin{aligned}
0\times0 + 1\times1 + 3\times2 + 4\times3 &= 19,\\
1\times0 + 2\times1 + 4\times2 + 5\times3 &= 25,\\
3\times0 + 4\times1 + 6\times2 + 7\times3 &= 37,\\
4\times0 + 5\times1 + 7\times2 + 8\times3 &= 43.
\end{aligned}
$$

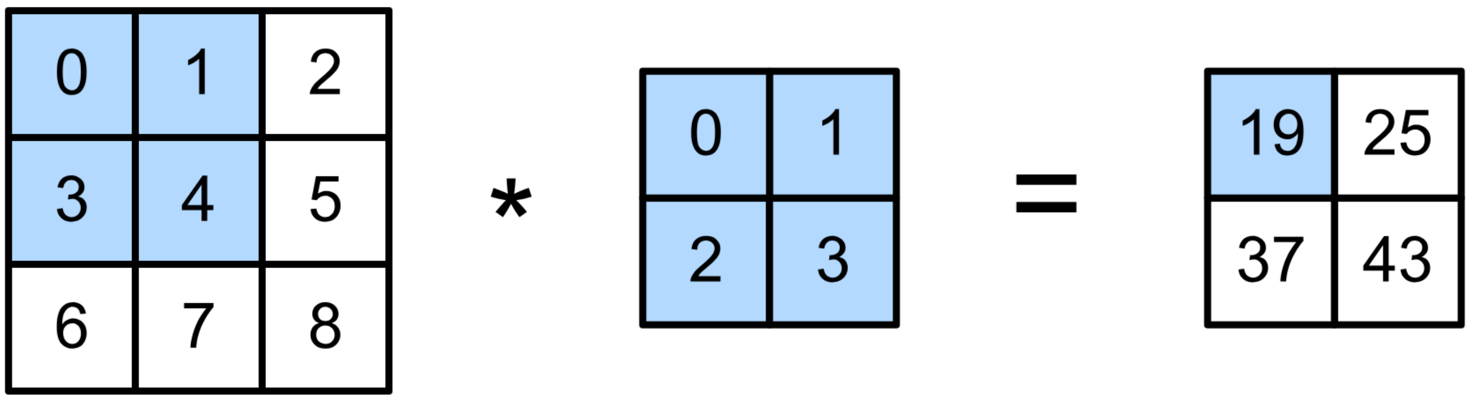


(vdumoulin@ Github)

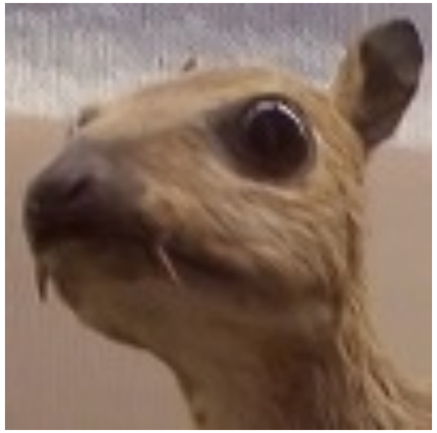
2-D Convolution Layer

- $\mathbf{X}$: $n_h \times n_w$ input matrix
- $\mathbf{W}$: $k_h \times k_w$ kernel matrix
- b: scalar bias
- $\mathbf{Y}$: $(n_h - k_h + 1) \times (n_w. - k_w + 1)$ output matrix
- **W** and *b* are learnable parameters

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + b$$



Examples



(wikipedia)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge Detection

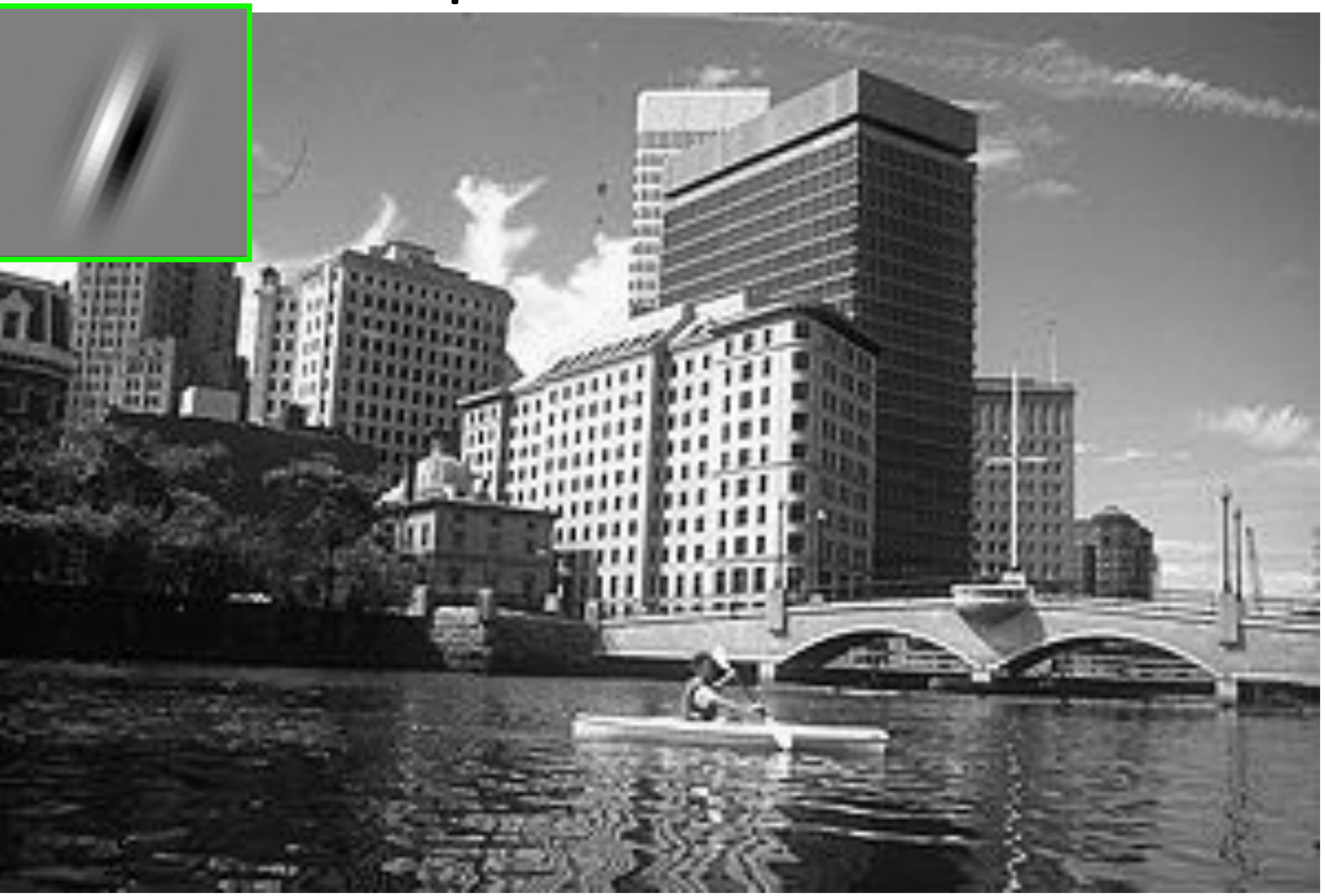
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen

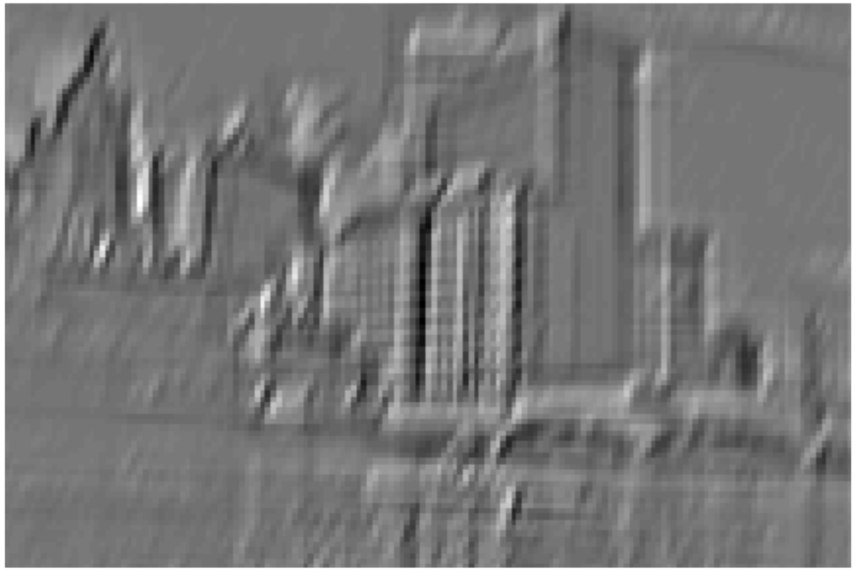
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian Blur

Examples



(Rob Fergus)



1-D and 3-D Cross Correlations

- 1-D

$$y_i = \sum_{a=1}^{h} w_a x_{i+a}$$

- 3-D

$$y_{i,j,k} = \sum_{a=1}^{h} \sum_{b=1}^{w} \sum_{c=1}^{d} w_{a,b,c} x_{i+a,j+b,k+c}$$

- Text
- Voice
- Time series

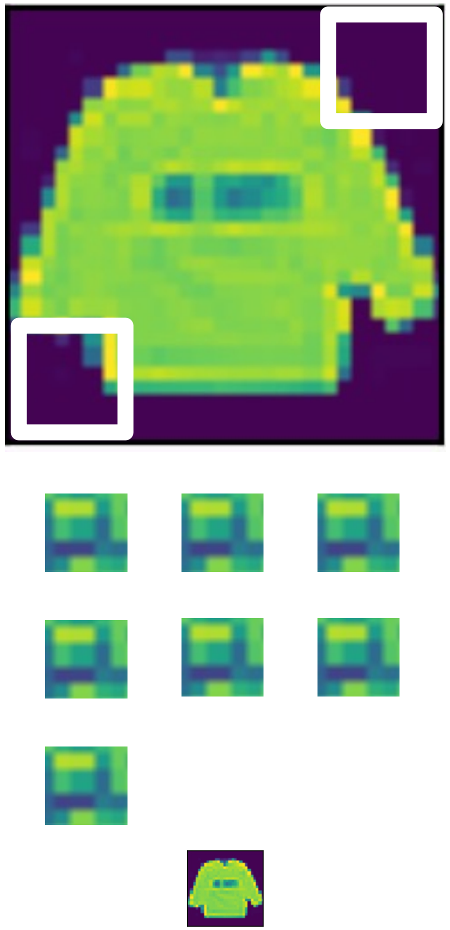
- Video
- Medical images

Padding and Stride

Padding

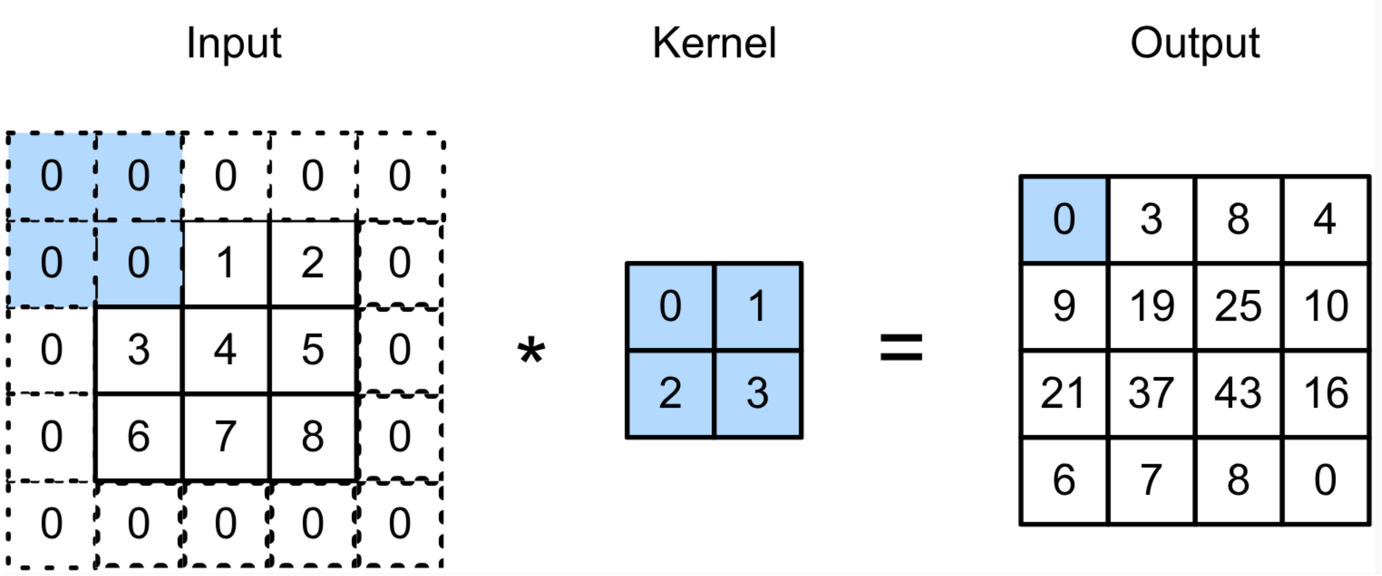
- Given a 32 x 32 input image
- Apply convolutional layer with 5 x 5 kernel
  - 28 x 28 output with 1 layer
  - 4 x 4 output with 7 layers
- Shape decreases faster with larger kernels
  - Shape reduces from $n_h \times n_w$

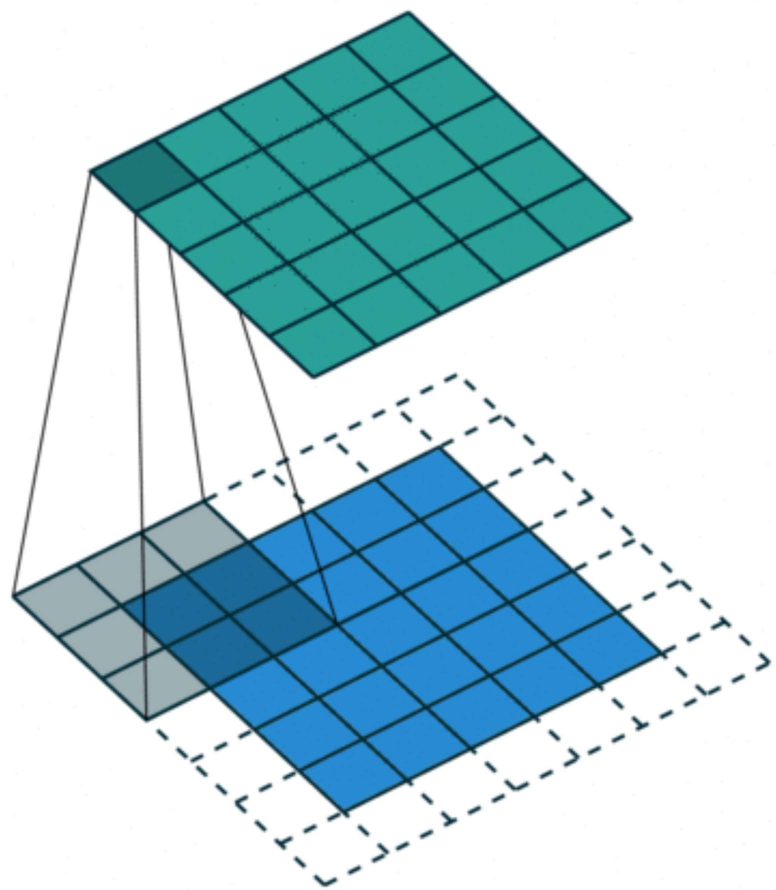
    to $(n_h - k_h + 1) \times (n_w - k_w + 1)$



Padding

Fills in rows/columns around input (with 0's)





$$0{\times}0 + 0{\times}1 + 0{\times}2 + 0{\times}3 = 0$$

Padding

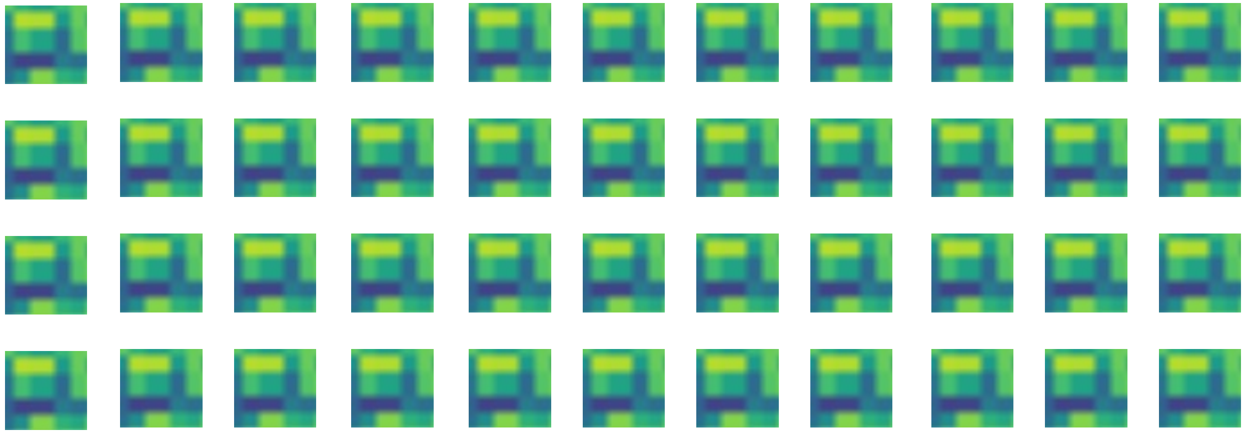
- Padding p_h rows and. p_w columns, output shape will be

$$(n_h - k_h + p_h + 1)\times(n_w - k_w + p_w + 1)$$

- A common choice is $p_h = k_h - 1$ and $p_w = k_w - 1$
  - Odd k_h pad $p_h/2$ on both sides
  - Even k_h pad $\lceil p_h/2 \rceil$ on top, $\lfloor p_h/2 \rfloor$ on bottom

Stride

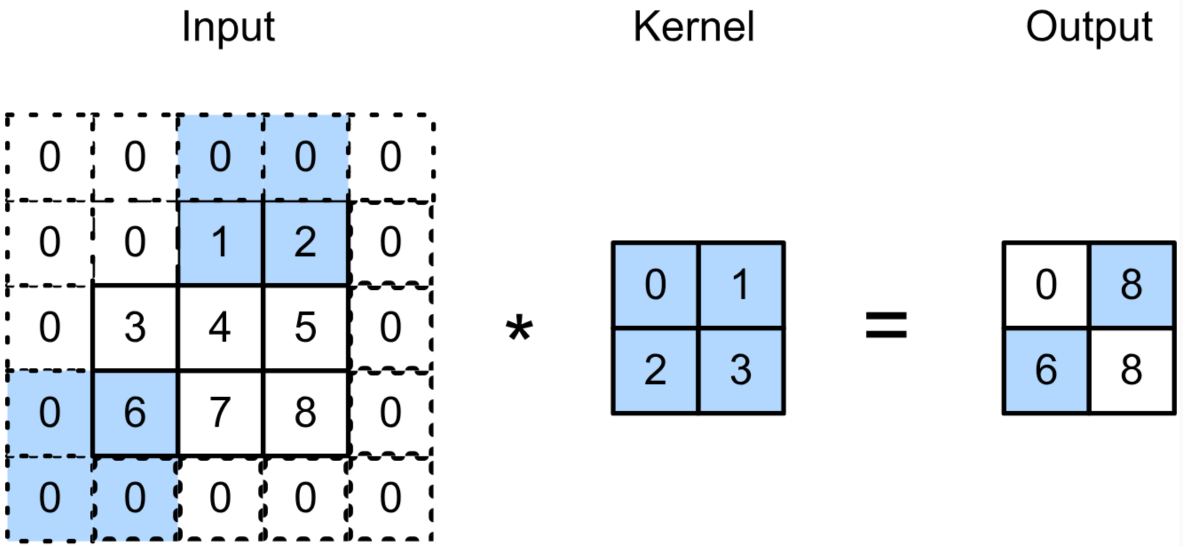
- Padding reduces shape linearly with #layers
  - Given a 224 x 224 input with a 5 x 5 kernel, needs 44 layers to reduce the shape to 4 x 4
  - Requires a large amount of computation

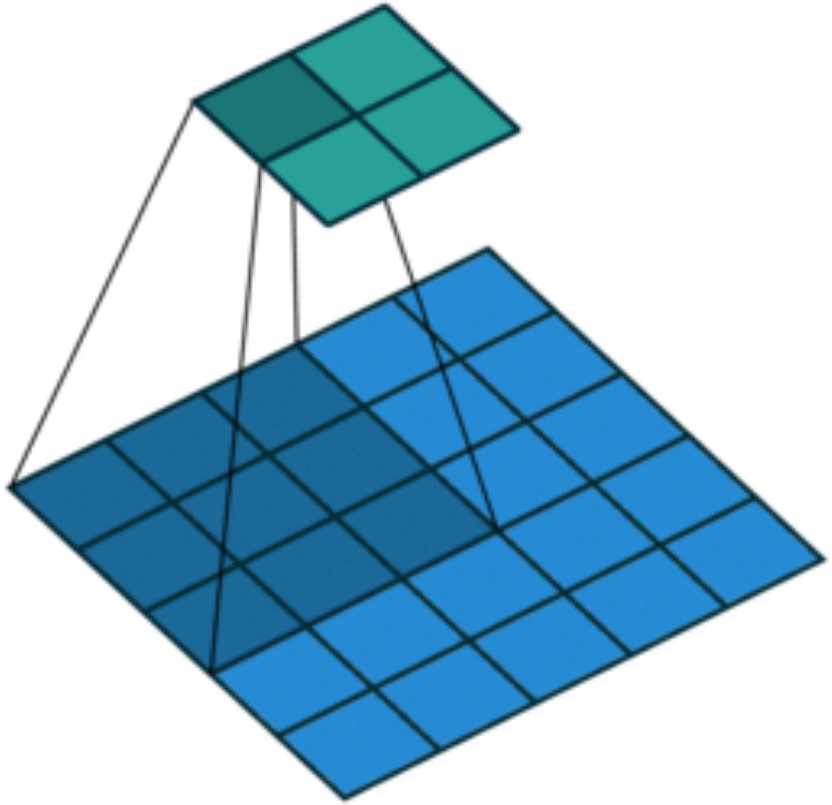


Stride

- Stride is the #rows/#columns per slide

  Strides of 3 and 2 for height and width





$$0{\times}0 + 0{\times}1 + 1{\times}2 + 2{\times}3 = 8$$
$$0{\times}0 + 6{\times}1 + 0{\times}2 + 0{\times}3 = 6$$

Stride

- Given stride s_h for the height and stride s_w for the width, the output shape is

$$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$

- With $p_h = k_h - 1$ and $p_w = k_w - 1$

$$\lfloor (n_h + s_h - 1)/s_h \rfloor \times \lfloor (n_w + s_w - 1)/s_w \rfloor$$

- If input height/width are divisible by strides

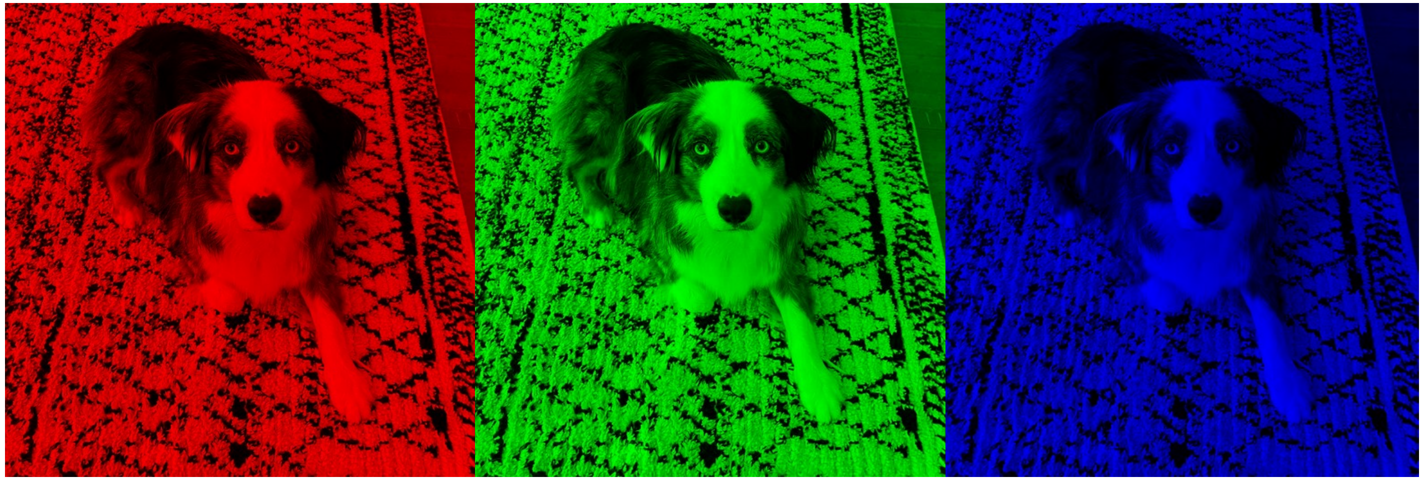
$$(n_h/s_h) \times (n_w/s_w)$$

Multiple Input and Output Channels

Multiple Input Channels

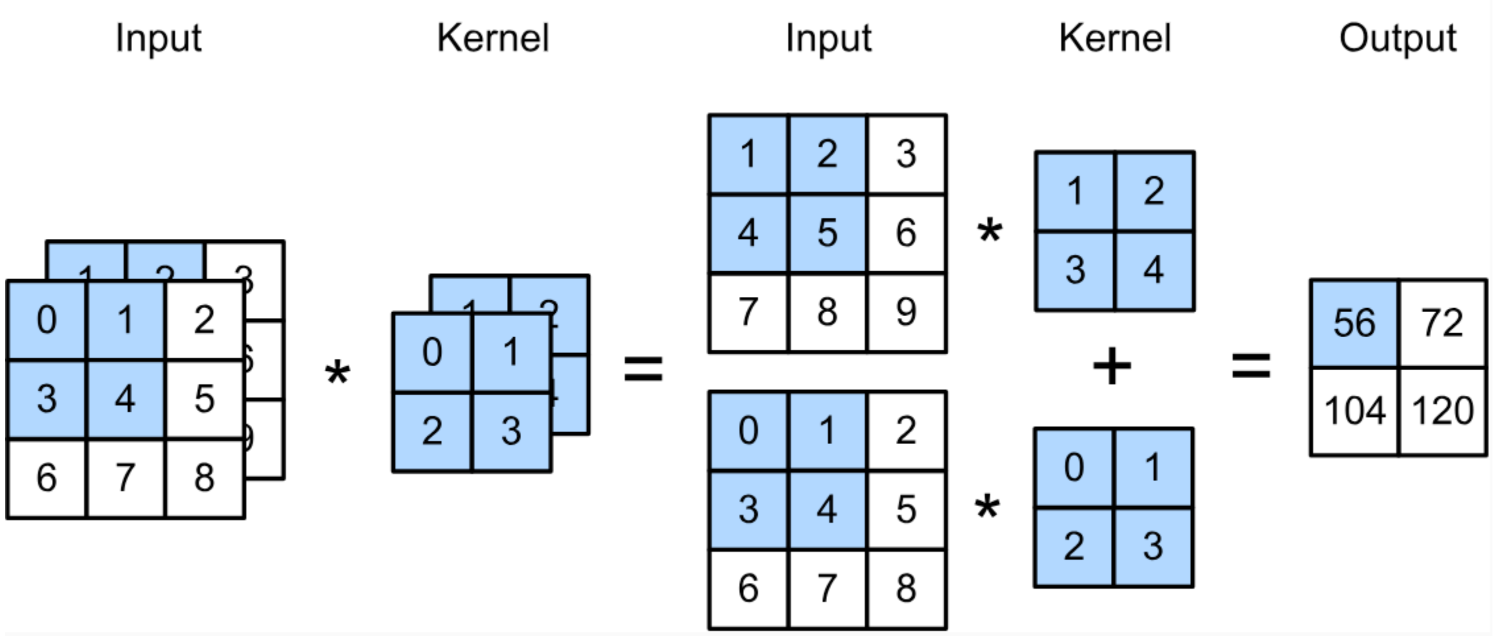
- Color images typically have three channels (RGB)
- Converting to grayscale loses information

=



Multiple Input Channels

- Allocate a separate kernel for each input channel, sum results over all channels to produce feature map



$$(1\times1 + 2\times2 + 4\times3 + 5\times4)$$
$$+(0\times0 + 1\times1 + 3\times2 + 4\times3)$$
$$= 56$$

Multiple Input Channels

- $\mathbf{X}$: $c_i \times n_h \times n_w$ input
- $\mathbf{W}$: $c_i \times k_h \times k_w$ kernel
- $\mathbf{Y}$: $m_h \times m_w$ output

$$\mathbf{Y} = \sum_{i=0}^{c_i} \mathbf{X}_{i,:,:} \star \mathbf{W}_{i,:,:}$$

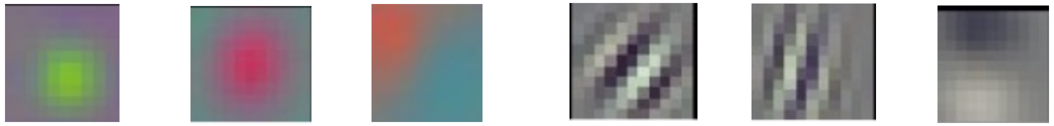
Multiple Output Channels

- With multiple kernels, each one generates an output channel
- Each channel is called a “feature map”
- Stacked together, we can think of this as a 4D parameter

- Input $\mathbf{X}: c_i \times n_h \times n_w$
- Kernel $\mathbf{W}: c_o \times c_i \times k_h \times k_w$
- Output $\mathbf{Y}: c_o \times m_h \times m_w$

Multiple Input/Output Channels

- Each output channel may recognize a particular pattern

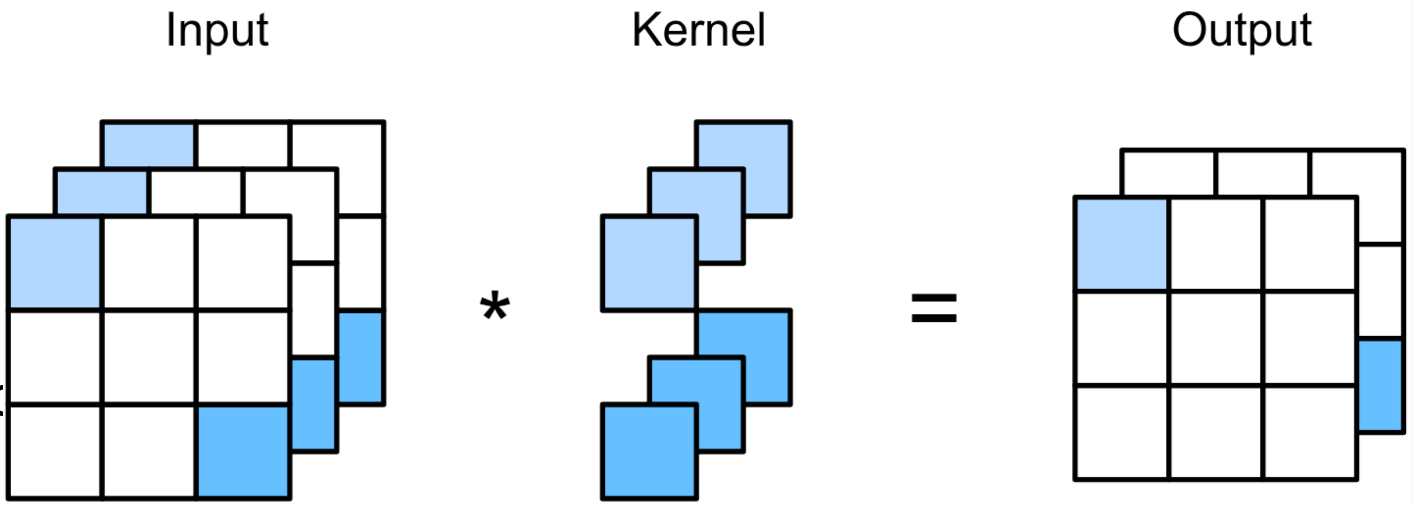



- Input channels kernels recognize and combines patterns in inputs

1 x 1 Convolutional Layer

$k_h = k_w = 1$

is a popular choice. It doesn't recognize spatial patterns, but fuse channels.



Equal to a d
weig

$n_h n_w \times c_i$

$c_o \times c_i$

Pooling Layers

Pooling

- Convolution is sensitive to position
  - Detect vertical edges

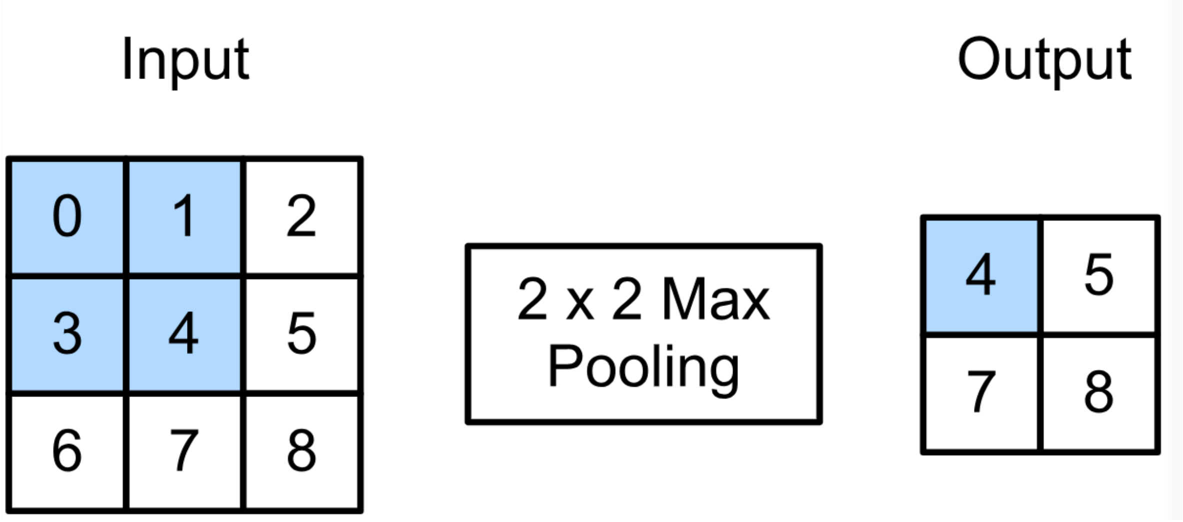
X
```
[[1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
```

Y
```
[[ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
```

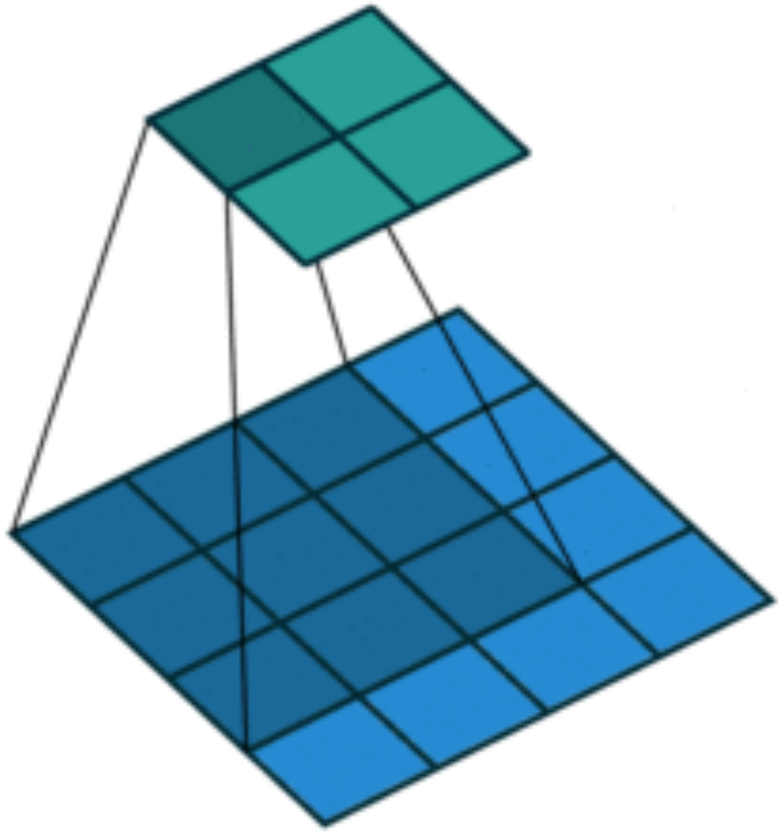
- We need some degree of invariance to translation
  - Lighting, object positions, scales, appearance vary among images

2-D Max Pooling

- Returns the maximal value in sliding wind



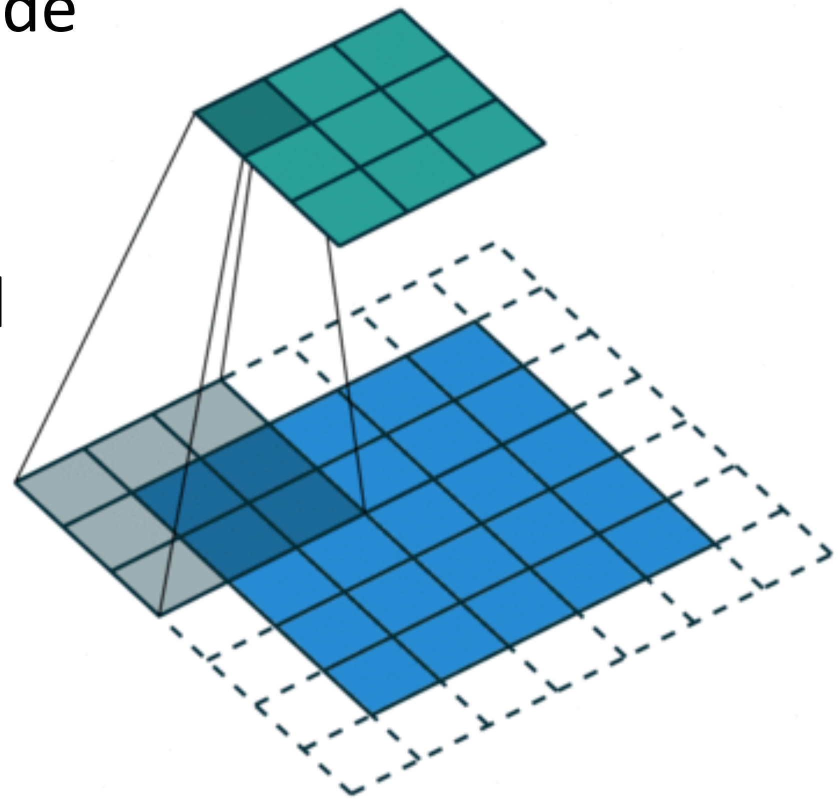
$$max(0,1,3,4) = 4$$



Padding, Stride, and Multiple Channels

- Pooling layers have similar padding and stride as convolutional layers
- No learnable parameters
- Pooling applied separately on each channel

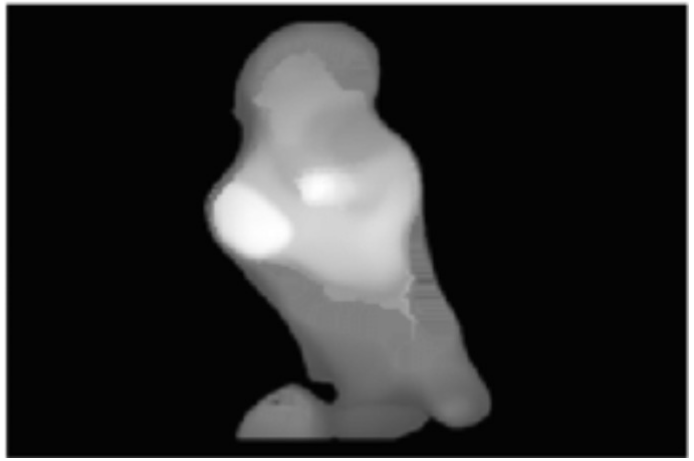
#output channels = #input channels



Max vs Mean

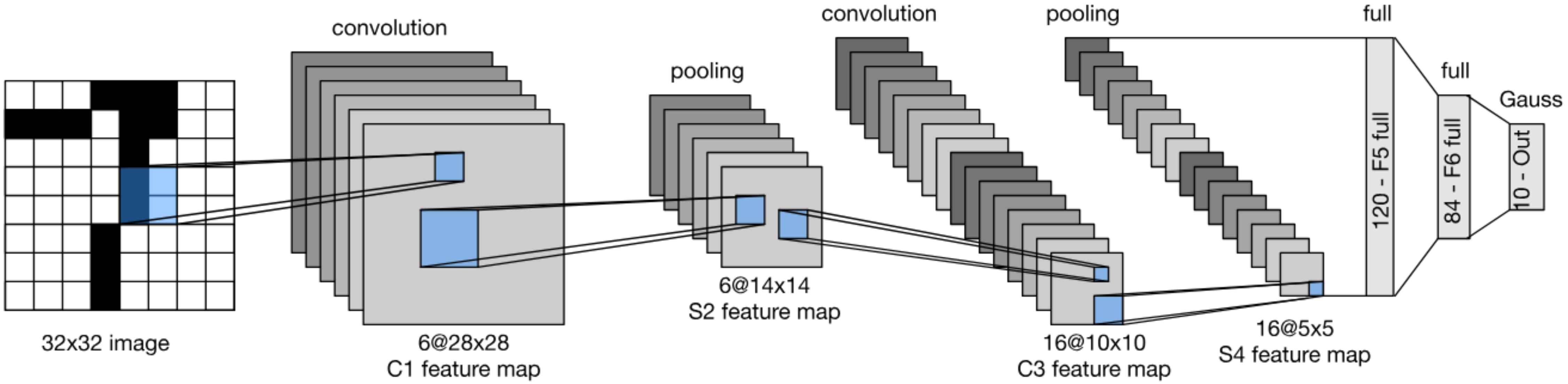
- Max pooling: the strongest pattern signal in a window, non-linear
- Average pooling: replace max with mean in max pooling, linear
  - The average signal in each window

Max pooling

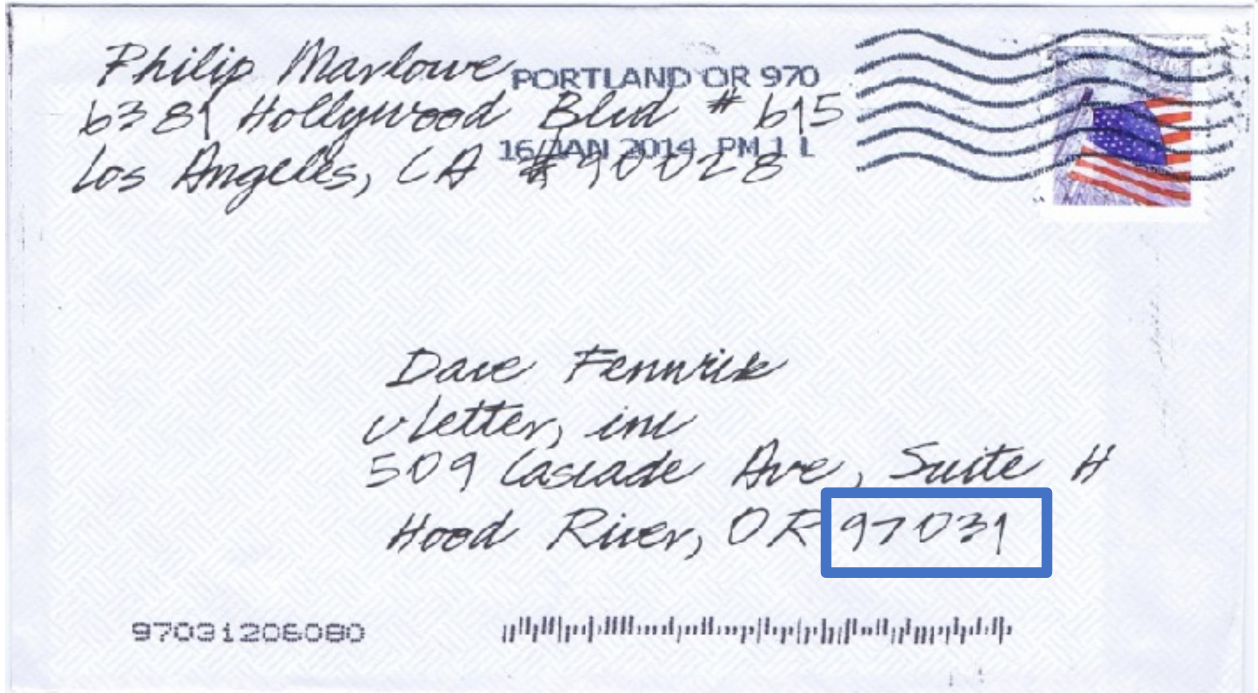


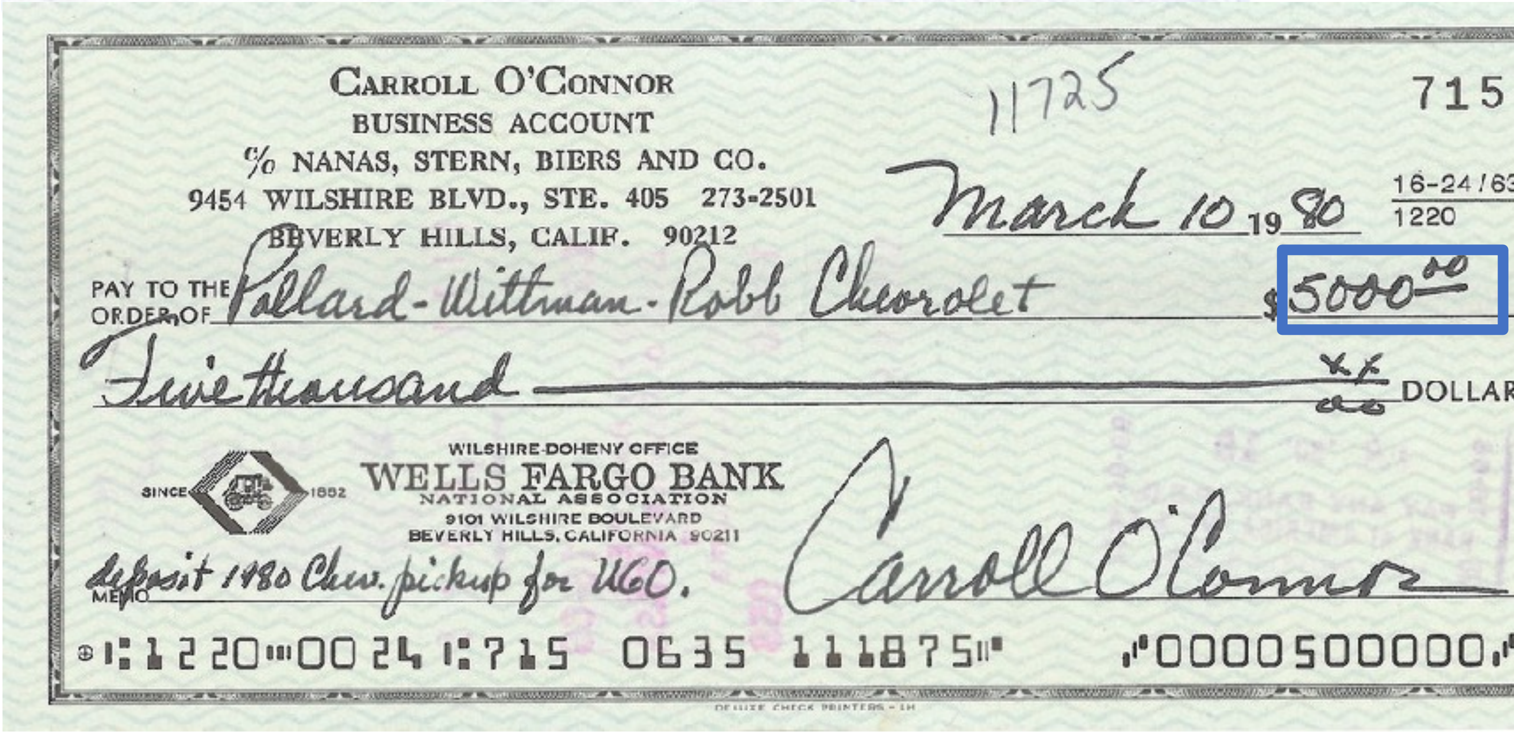
Average pooling

The LeNet Architecture



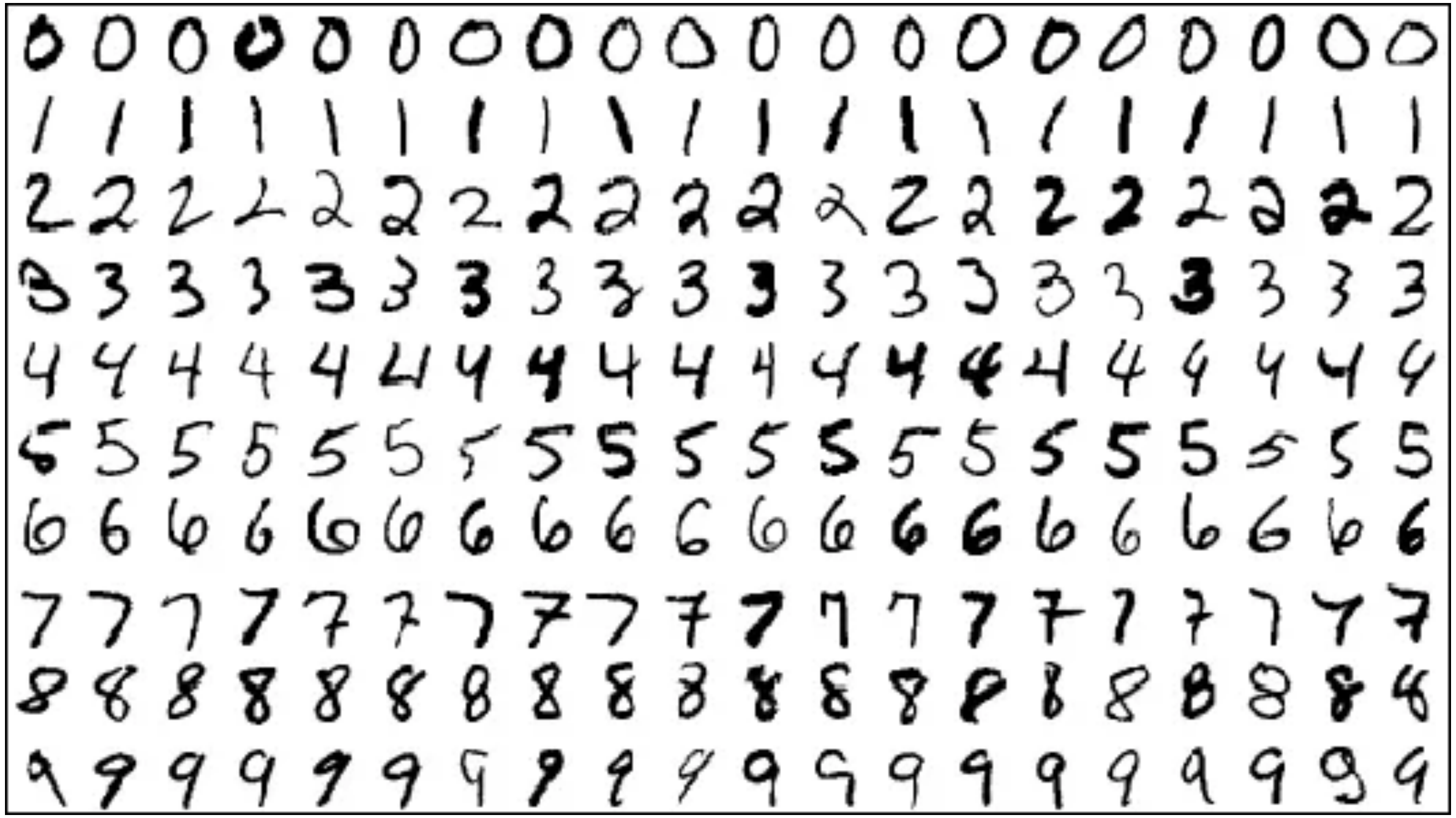
Handwritten Digit Recognition

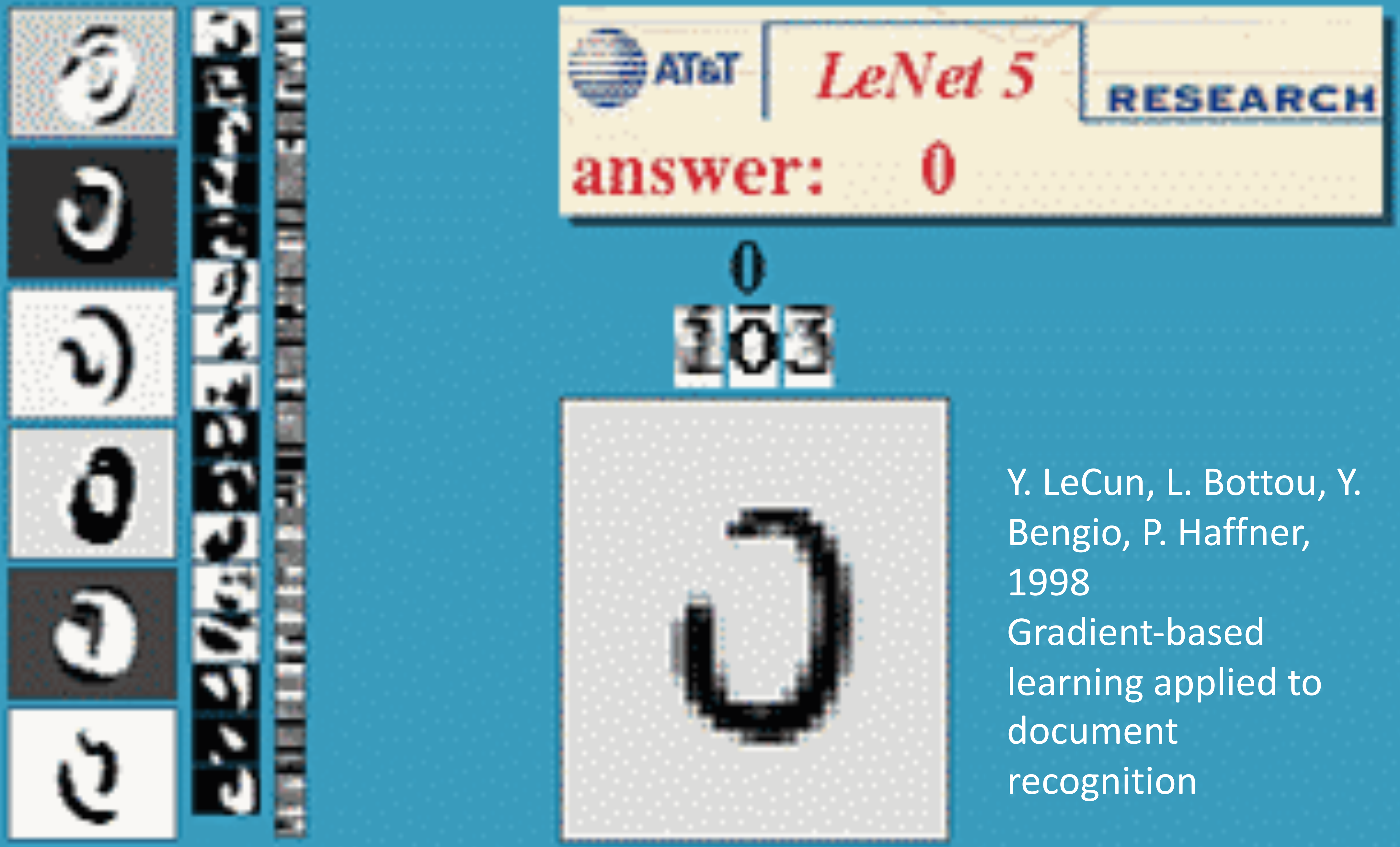




MNIST

- Centered and scaled
- 50,000 training data
- 10,000 test data
- 28 x 28 images
- 10 classes





Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998
Gradient-based learning applied to document recognition

LeNet in Pytorch

```python
class LeNet5(nn.Module):

    def __init__(self, n_classes):
        super(LeNet5, self).__init__()

        self.feature_extractor = nn.Sequential(
            nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5, stride=1),
            nn.Tanh(),
            nn.AvgPool2d(kernel_size=2),
            nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5, stride=1),
            nn.Tanh(),
            nn.AvgPool2d(kernel_size=2),
            nn.Conv2d(in_channels=16, out_channels=120, kernel_size=5, stride=1),
            nn.Tanh()
        )

        self.classifier = nn.Sequential(
            nn.Linear(in_features=120, out_features=84),
            nn.Tanh(),
            nn.Linear(in_features=84, out_features=n_classes),
        )

    def forward(self, x):
        x = self.feature_extractor(x)
        x = torch.flatten(x, 1)
        logits = self.classifier(x)
        probs = F.softmax(logits, dim=1)
        return logits, probs
```

https://towardsdatascience.com/implementing-yann-lecuns-lenet-5-in-pytorch-5e05a0911320

Summary

- Convolutional layer
  - Reduced model capacity compared to dense layer
  - Efficient at detecting spatial pattens
  - Enforces locality, spatial invariances
  - Computable in parallel
  - Control output shape via padding, strides and channels
- Max/Average Pooling layer
  - Provides some degree of invariance to translation
- Architecture Pattern
  - As network gets deeper, downsample on spatial axes, grow # of channels