CS 760: Machine Learning
Convolutional Neural Networks

Fred Sala

University of Wisconsin-Madison

October 19, 2021

Announcements

- **Logistics**:
  - HW 3 grades released, proposal feedback returned
  - Coming up: HW 4 due (Friday!), midterm review, midterm
- **Class roadmap**:

Tuesday, Oct. 19	Neural Networks IV
Thursday, Oct. 21	Neural Networks V
Tuesday, Oct. 26	Practical Aspects of Training + Review
Wed, Oct. 27	**Midterm**
Thursday, Oct. 28	Generative Models

NNs and More

Outline

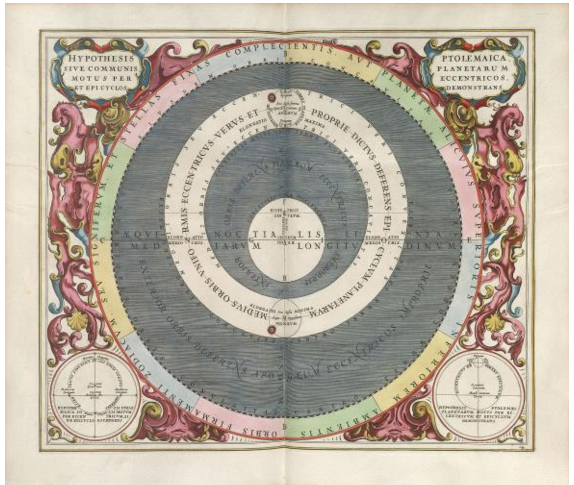
- **Review & Convolution Operator**
  - Experimental setup, convolution definition, vs. dense layers
- **CNN Components & Layers**
  - Padding, stride, channels, pooling layers
- **CNN Tasks & Architectures**
  - MNIST, ImageNet, LeNet, AlexNet, ResNets

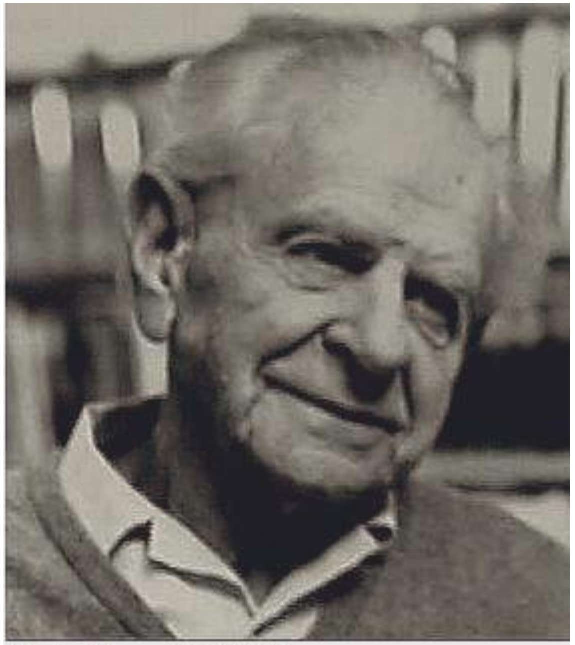
Outline

- **Review & Convolution Operator**
  - Experimental setup, convolution definition, vs. dense layers
- **CNN Components & Layers**
  - Padding, stride, channels, pooling layers
- **CNN Tasks & Architectures**
  - MNIST, ImageNet, LeNet, AlexNet, ResNets

Review: Experimental Setup

- **Hypothesis**
  - Needed for science of any sort (testable!)
  - “I will explore area Y”: not a hypothesis.
  - Details of experimental protocol are not part of hypothesis
- **Popper**: falsifiability





Sir Karl Popper (1902-1994)

Review: Experimental Setup Template

- Coffee Experiment (http://aberger.site/coffee/)
- Really great template for any paper’s **experimental setup**

Hypothesis

- Caffeine makes graduate students more productive.P

Proxy

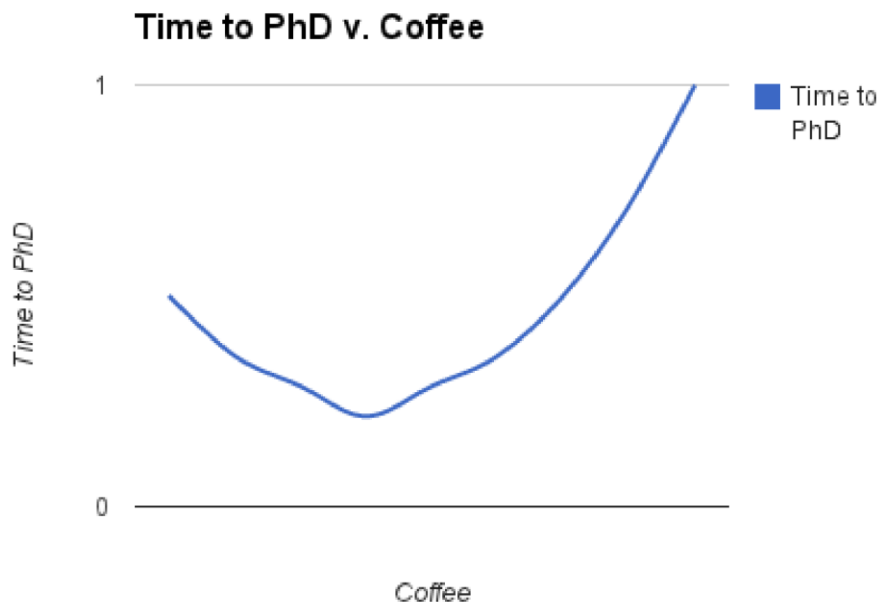
- Productivity: time it takes to complete their PhD
- Coffee consumption: # of cups of coffee a students drinks/day

Protocol:

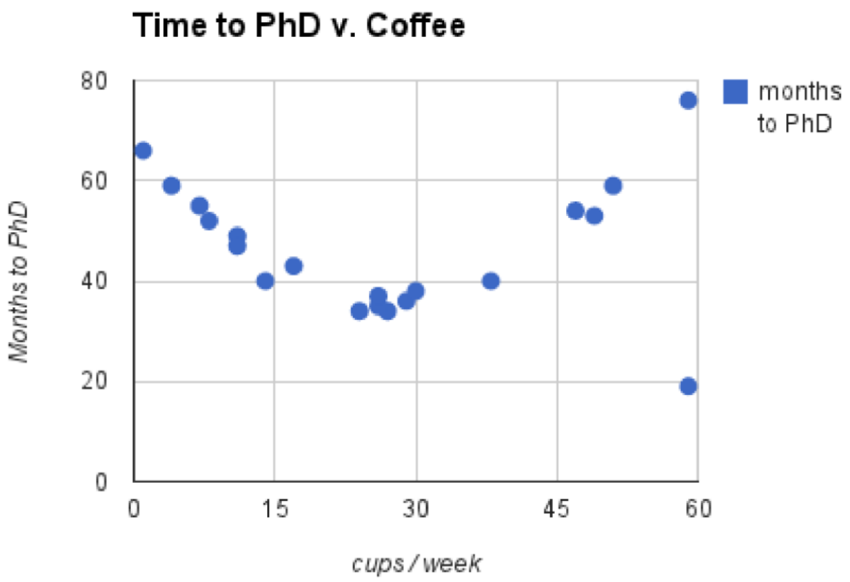
- Out of the 100 students in our school, have them report the mean cups of coffee they drink each week

Review: Coffee Experiment Continued

- **Expected Results**
  - No caffeine: slow.
  - Too much caffeine: caffeine tox.
  - Convex curve

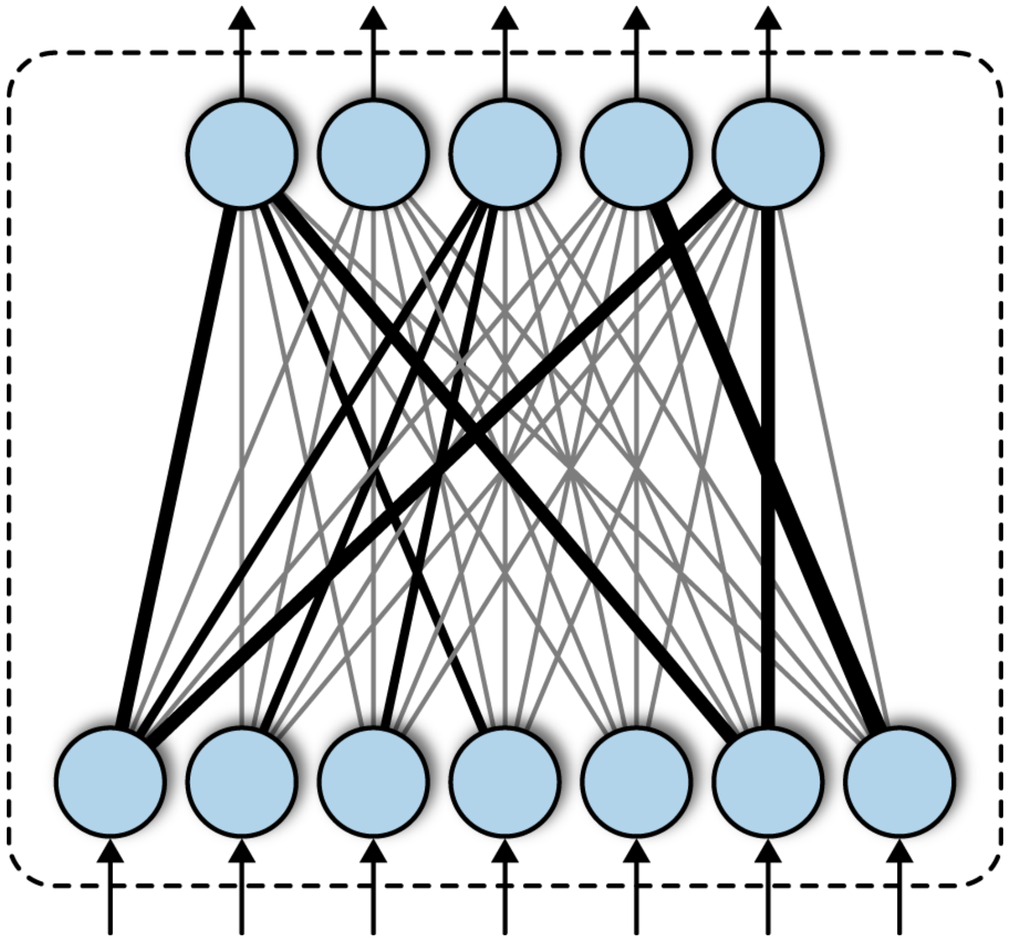


- **Results**
  - Match our expected results
  - Note outlier: further inquiry



Review: Fully-Connected Layers

- We used these in our MLPs:
- **Note**: lots of connections



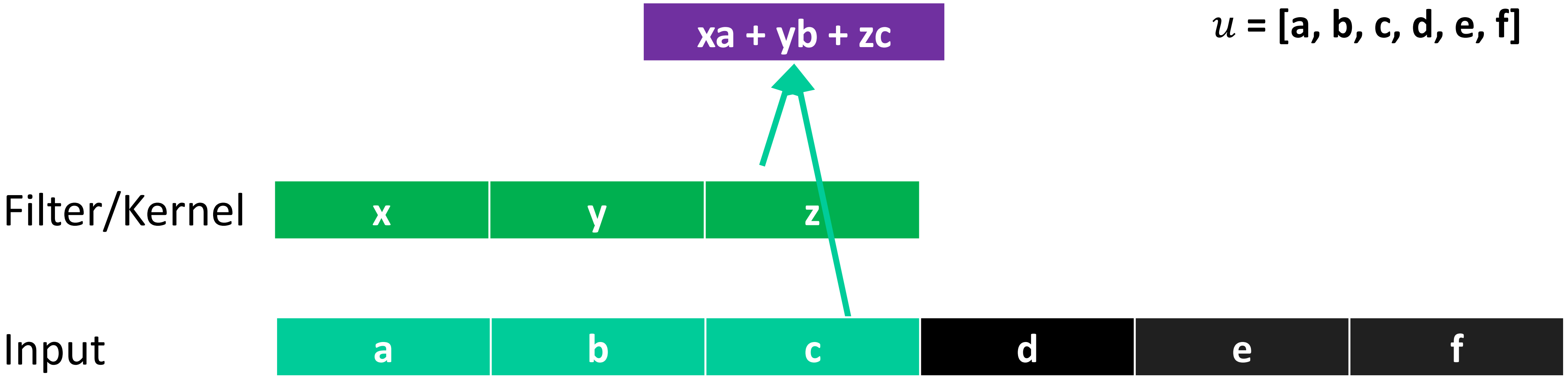
Review: Convolution Operator

- Basic formula: as $s = (u * w)$

$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a}$$

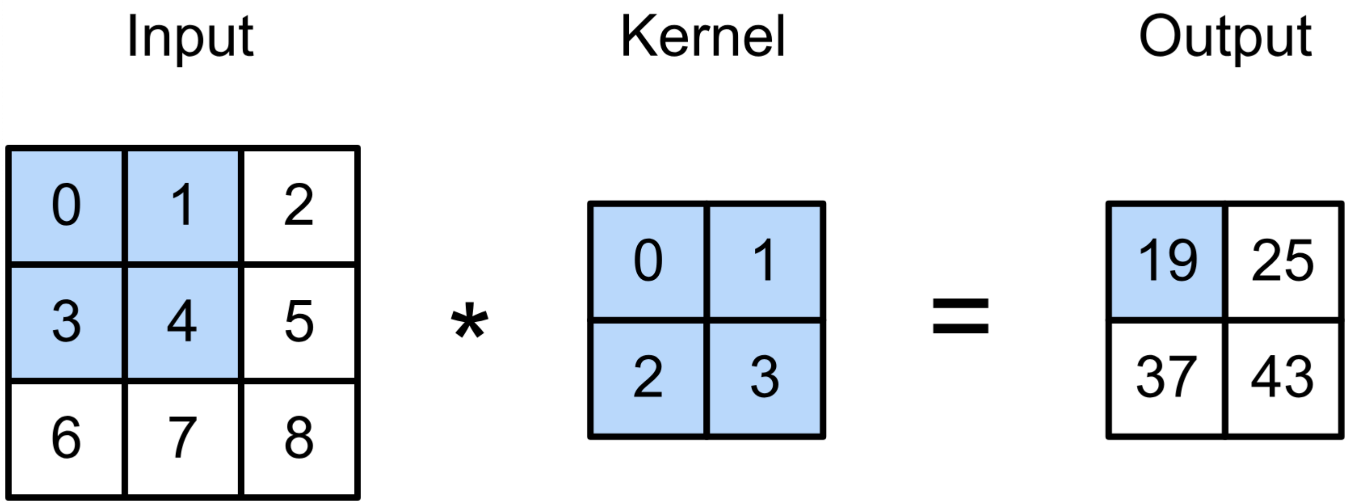
- Visual example:

w = [z, y, x]
u = [a, b, c, d, e, f]

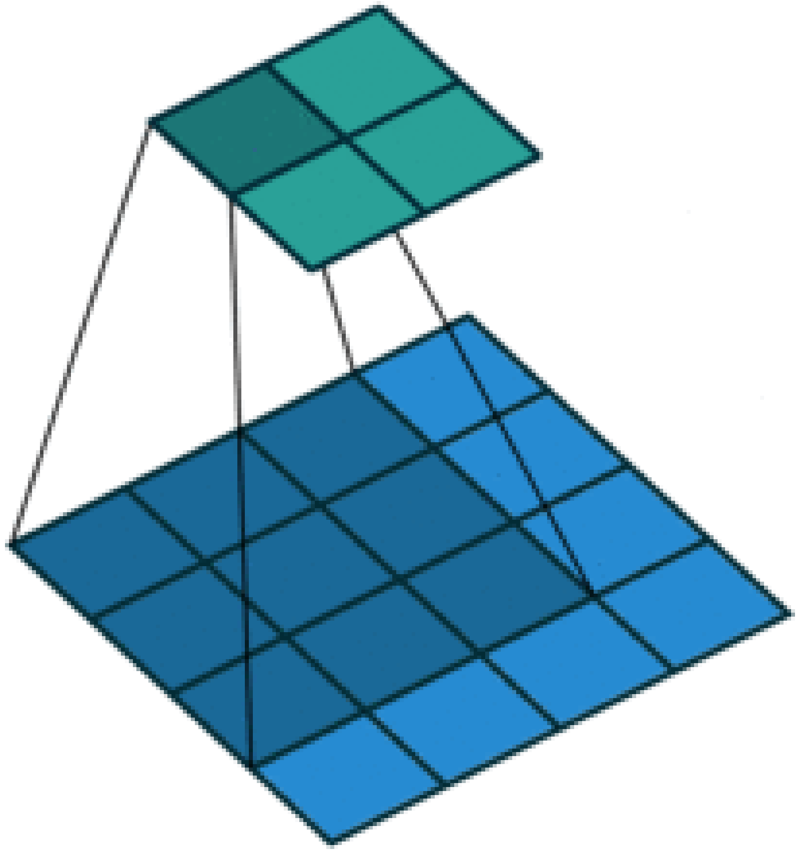


2-D Convolutions

- Example:



$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$



(vdumoulin@ Github)

Kernels: Examples

(wikipedia)

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

Edge Detection

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

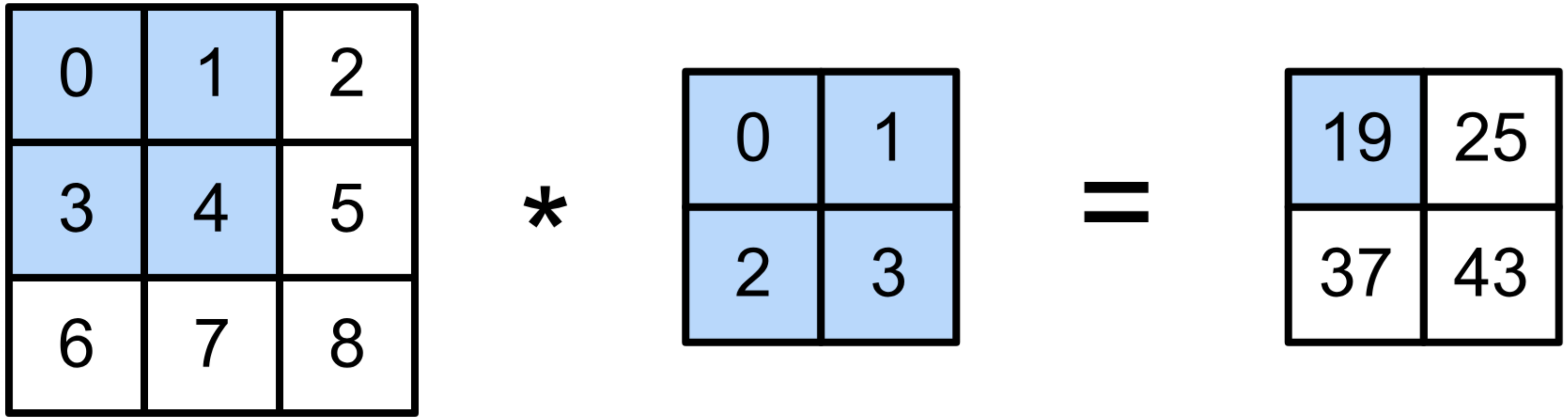
Sharpen

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

Gaussian Blur

Convolution Layers

- Notation:
  - n_h x n_w input matrix
  - k_h x k_w kernel matrix
  - b : bias (a scalar)
  - Y: () x () output matrix
- As usual W, b are learnable parameters

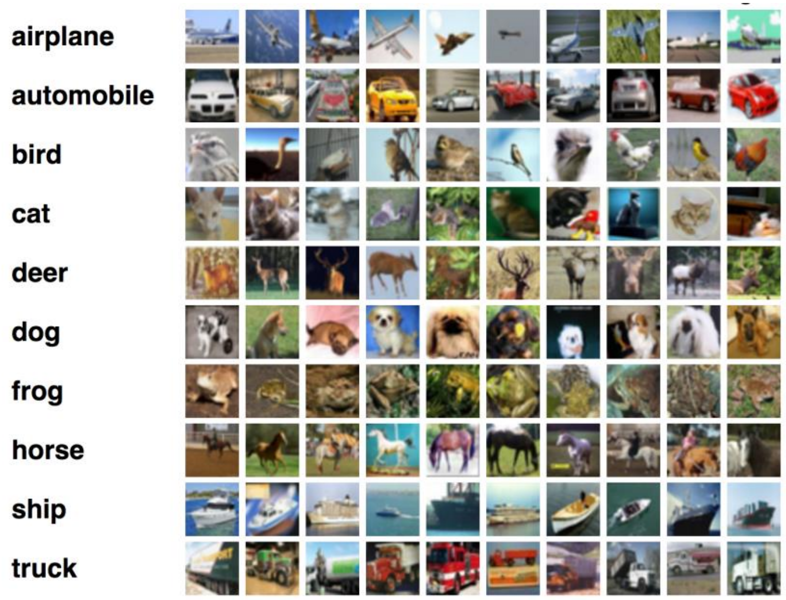


Convolutional Neural Networks

- Convolutional networks: neural networks that use **convolution** in place of general matrix multiplication in at least one of their layers

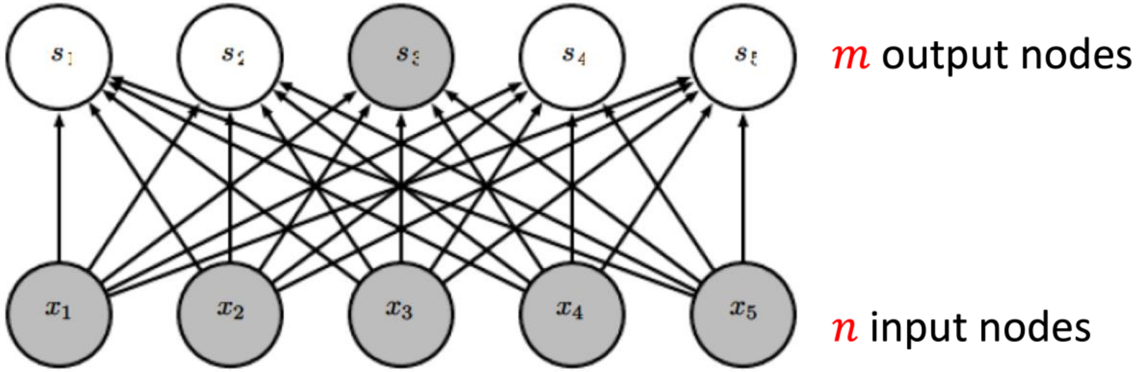
- Strong empirical application performance

- Standard for image tasks

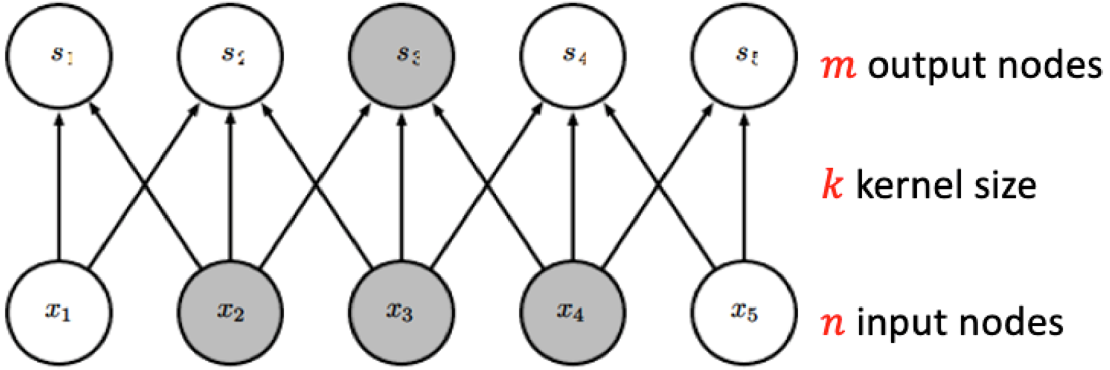


CNNs: Advantages

- Fully connected layer: m x n edges

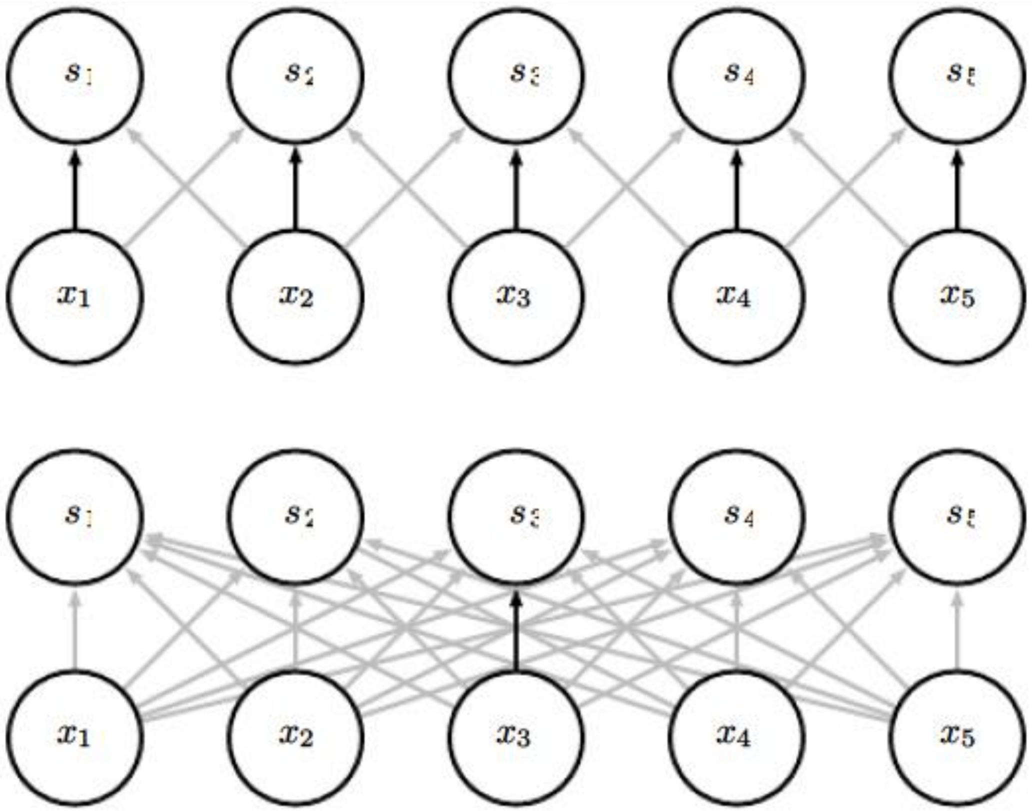


- Convolutional layer: ≤ m x k edges



CNNs: Advantages

- Convolutional layer: **same kernel used repeatedly**!



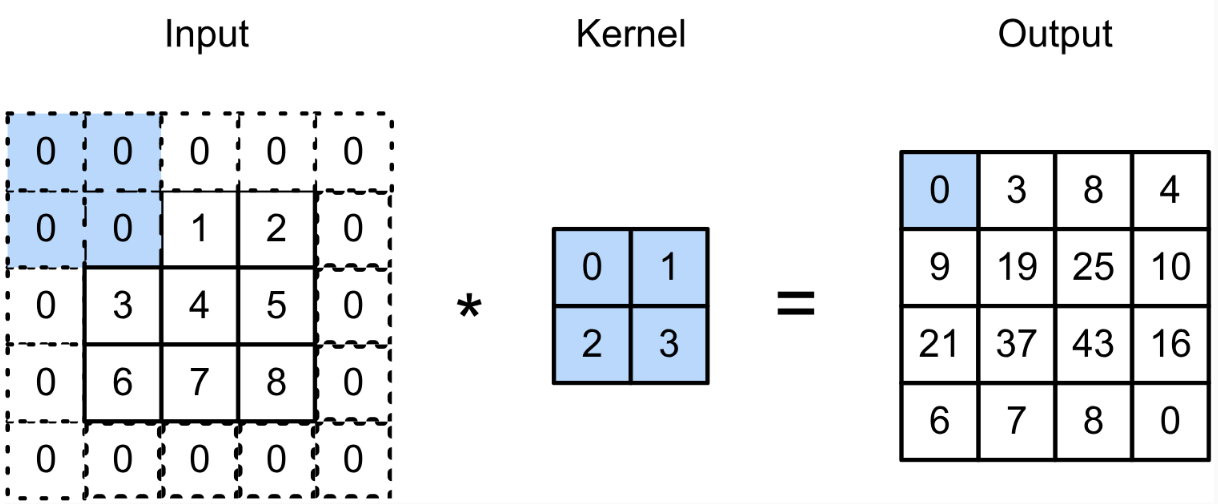
Break & Quiz

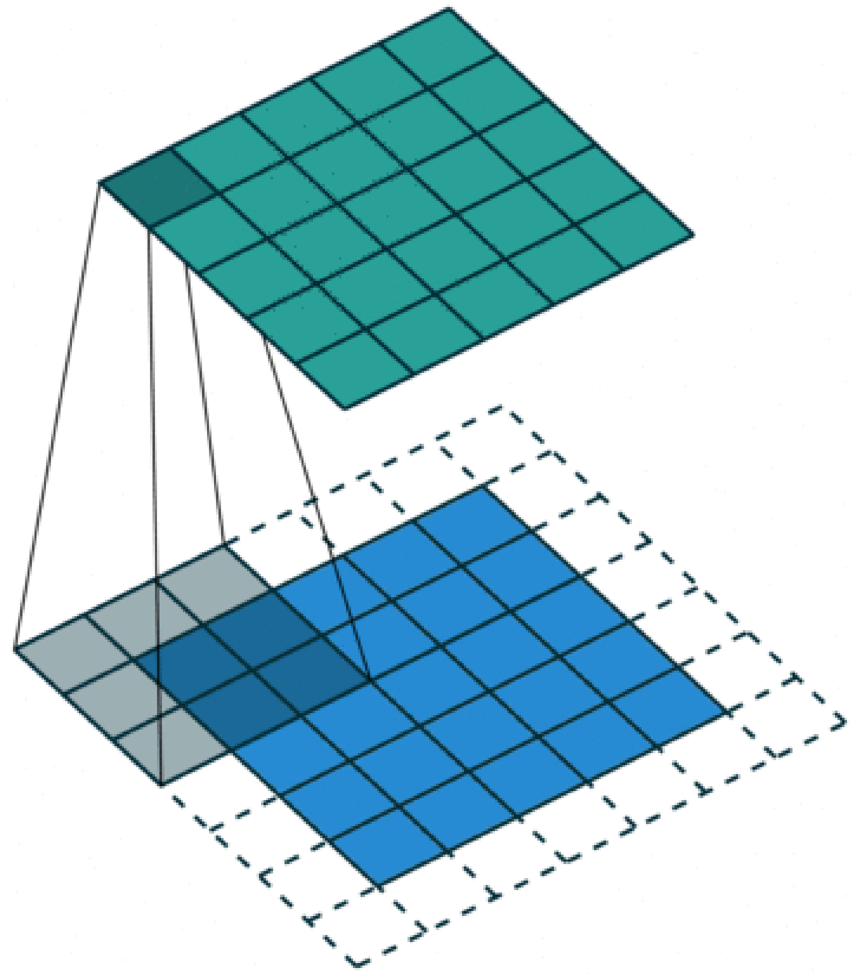
Outline

- **Review & Convolution Operator**
  - Experimental setup, convolution definition, vs. dense layers
- **CNN Components & Layers**
  - Padding, stride, channels, pooling layers
- **CNN Tasks & Architectures**
  - MNIST, ImageNet, LeNet, AlexNet, ResNets

Convolutional Layers: Padding

Padding adds rows/columns around input





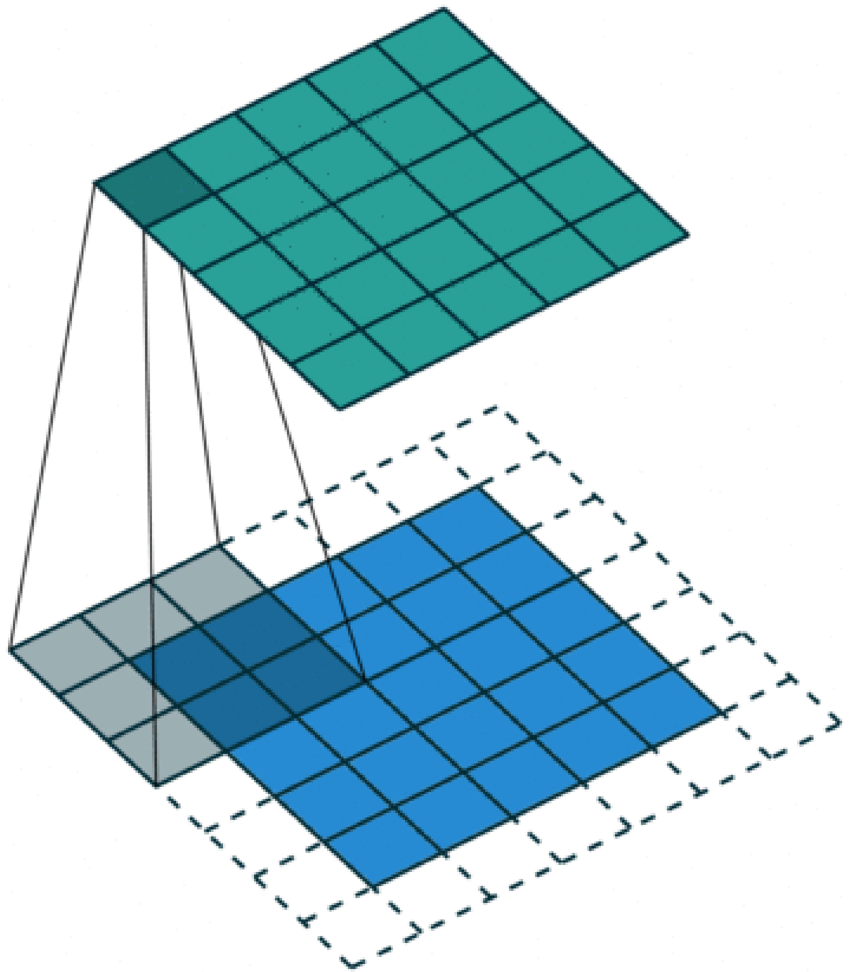
Convolutional Layers: Padding

Padding adds rows/columns around input

- Why?

1. Keeps **edge information**
2. Preserves sizes / allows deep networks
   - ie, for a 32x32 input image, 5x5 kernel, after 1 layer, get 28x28, after 7 layers, **only 4x4**

3. Can combine different filter sizes



Convolutional Layers: Padding

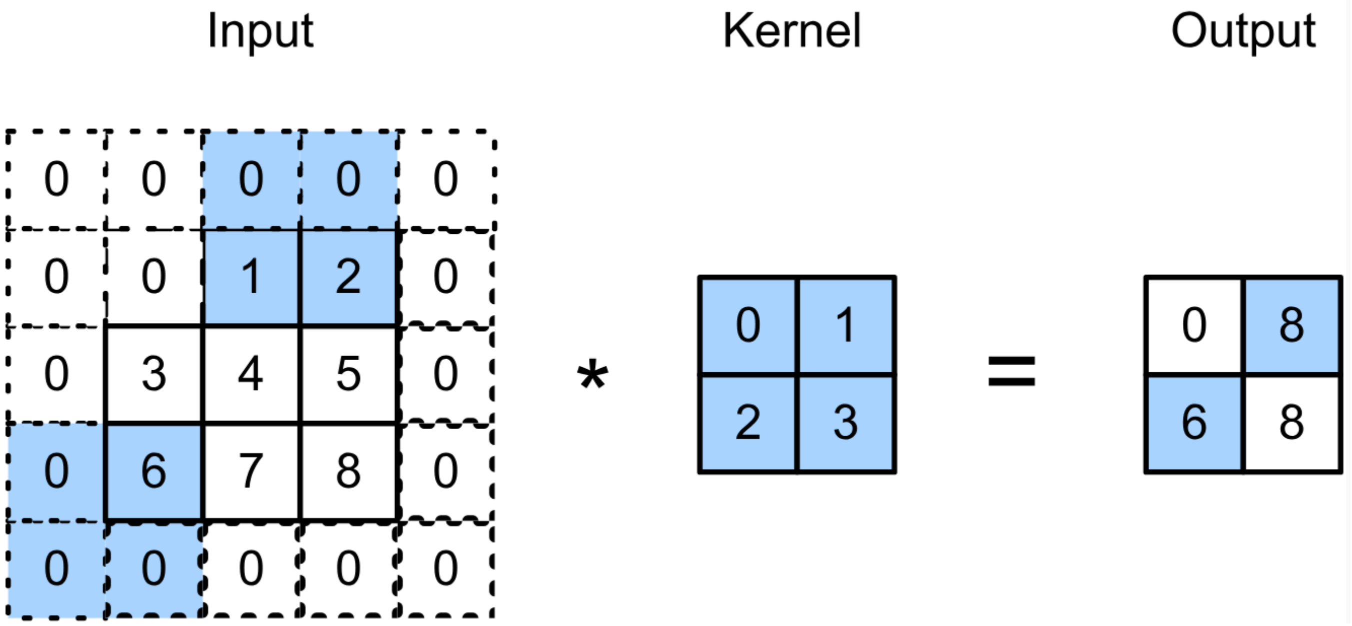
- Padding p_h rows and p_w columns, output shape is

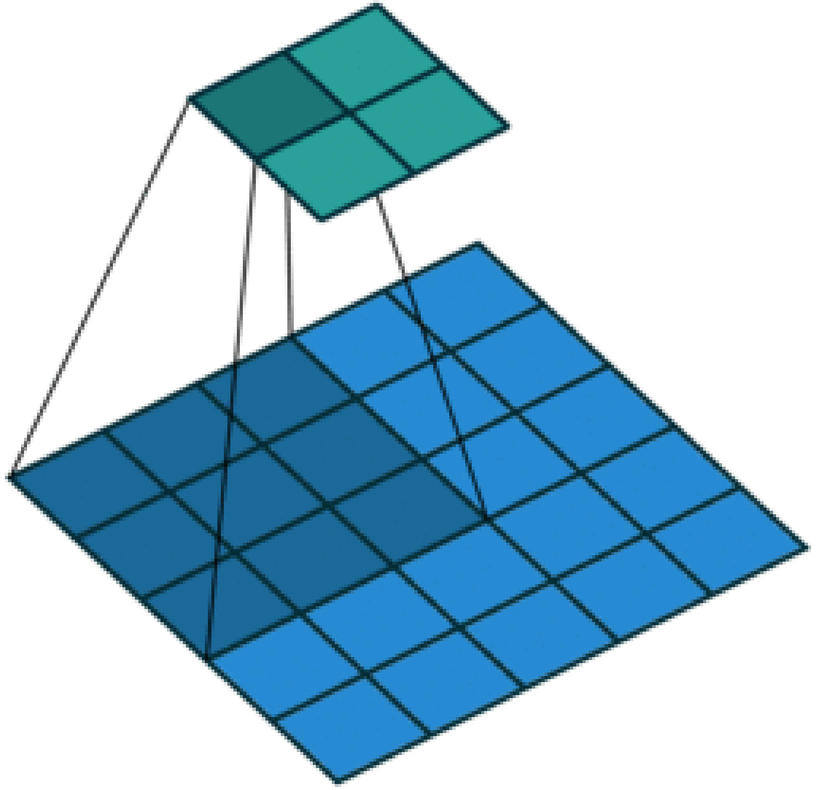
$$(n_h\text{-}k_h\text{+}p_h\text{+}1) \text{ x } (n_w\text{-}k_w\text{+}p_w\text{+}1)$$

- 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 ceil(p_h/2) on top, floor(p_h/2) on bottom

Convolutional Layers: Stride

- Stride: #rows/#columns per slide
- Example:





Convolutional Layers: Stride

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

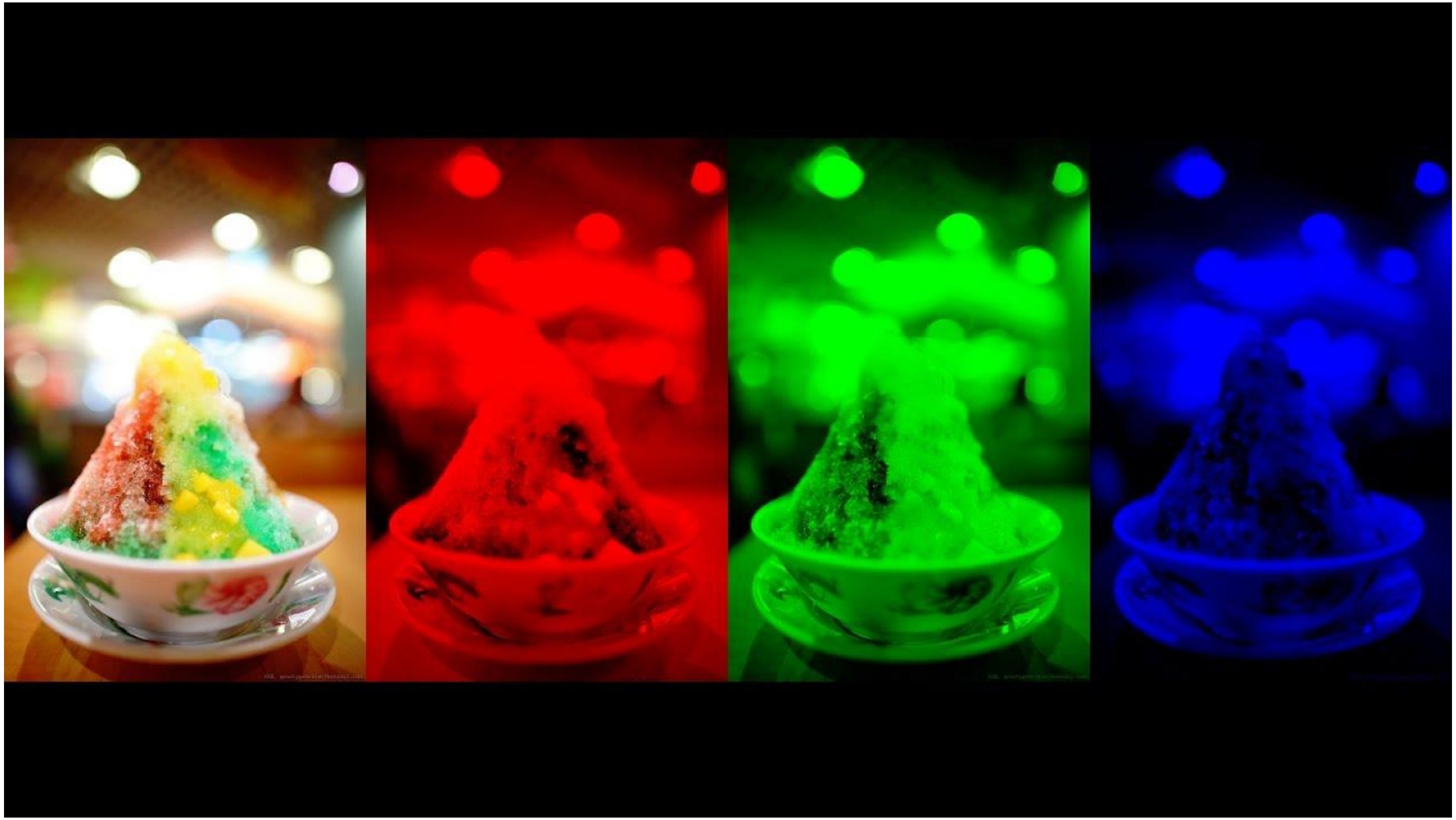
$$\lfloor(n_h-k_h+p_h+s_h)/s_h\rfloor \times \lfloor(n_w-k_w+p_w+s_w)/s_w\rfloor$$

- Set $p_h = k_h-1$, $p_w = k_w-1$, then get

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

Convolutional Layers: Channels

- Color images: three channels (RGB).



hyperCODEmia

Convolutional Layers: Channels

- Color images: three channels (RGB)
  - Note: contain different information
  - Just converting to one grayscale **image loses information**

wikipedia

Convolutional Layers: Channels

- How to integrate multiple channels?
  - Have a kernel for each channel, and then sum results over channels

$$\mathbf{X} : c_i \times n_h \times n_w$$

$$\mathbf{W} : c_i \times k_h \times k_w$$

$$\mathbf{Y} : m_h \times m_w$$

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

Convolutional Layers: Channels

- No matter how many inputs channels, so far we always get single output channel
- We can have **multiple 3-D kernels**, each one generates an output channel

$$\mathbf{X} : c_i \times n_h \times n_w$$

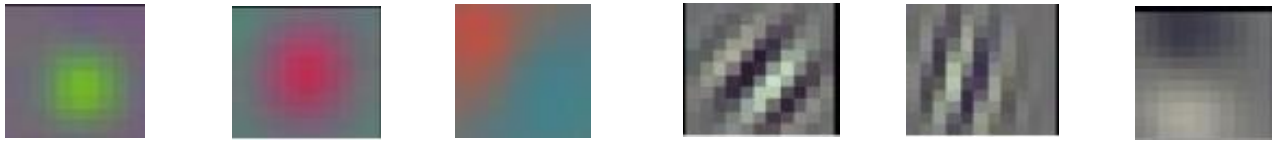
$$\mathbf{W} : c_o \times c_i \times k_h \times k_w$$

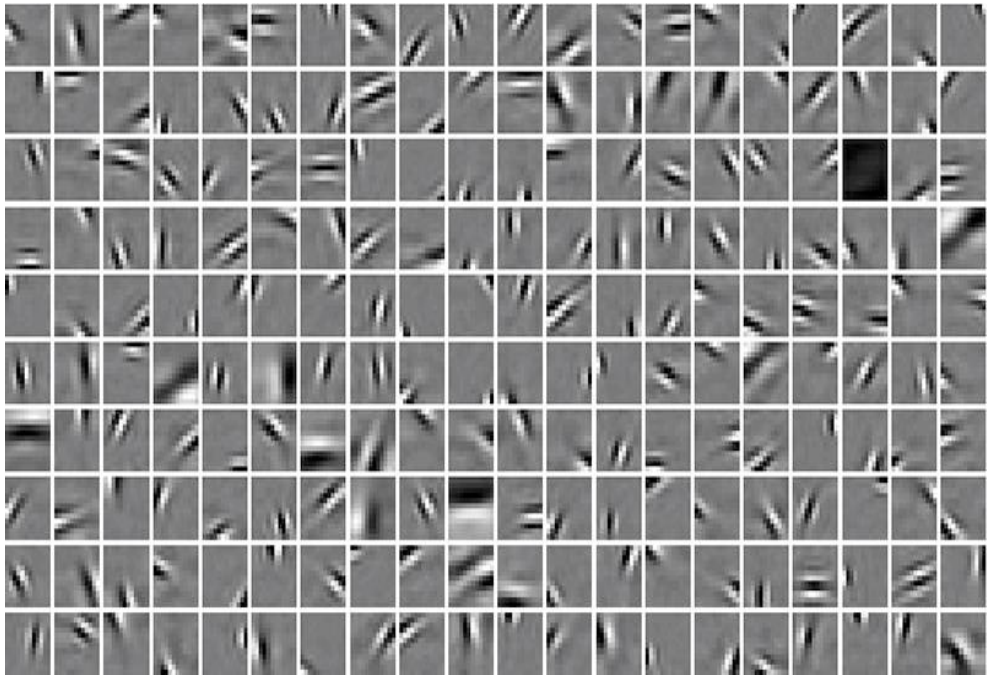
$$\mathbf{Y}_{i,:,:} = \mathbf{X} \star \mathbf{W}_{i,:,:,:}$$

$$\mathbf{Y} : c_o \times m_h \times m_w$$

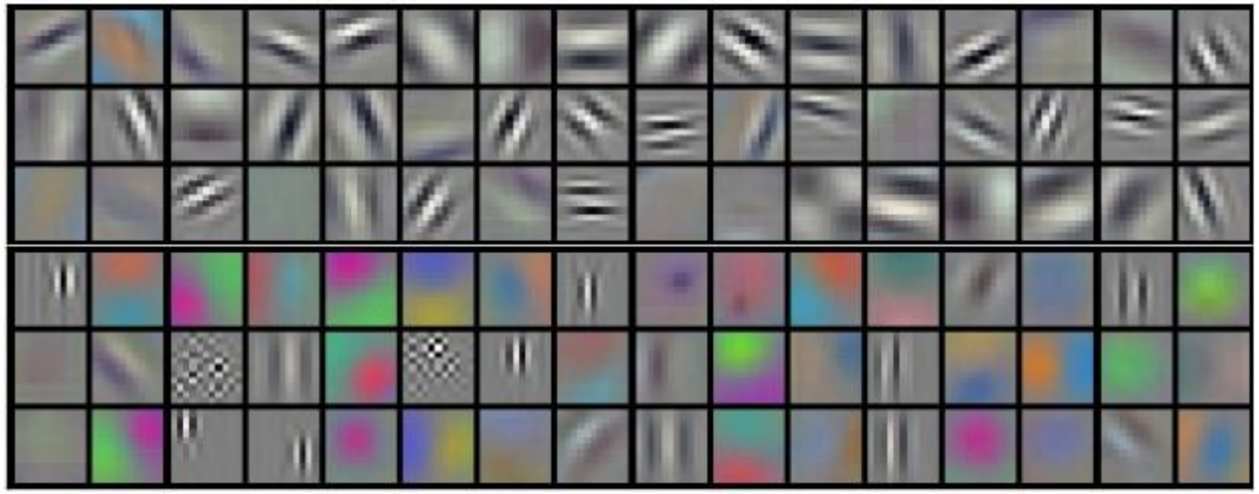
Convolutional Layers: Multiple Kernels

- Each 3-D kernel may recognize a particular pattern
  - Gabor filters





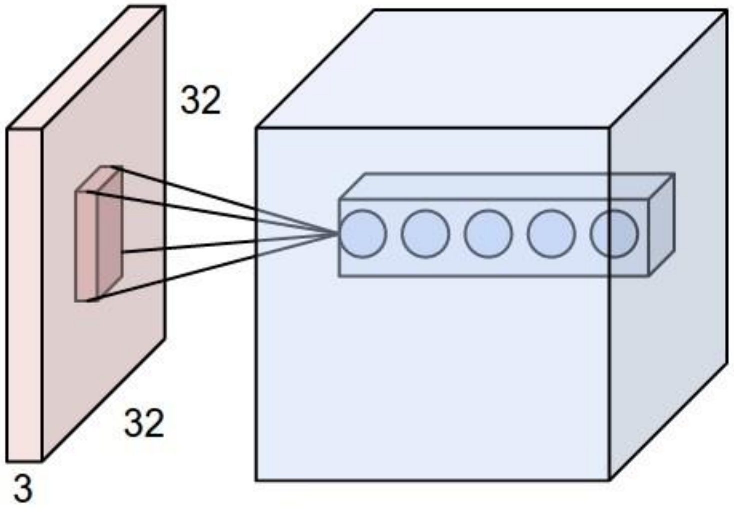
(Olshausen & Field, 1997)



Krizhevsky et al

Convolutional Layers: Summary

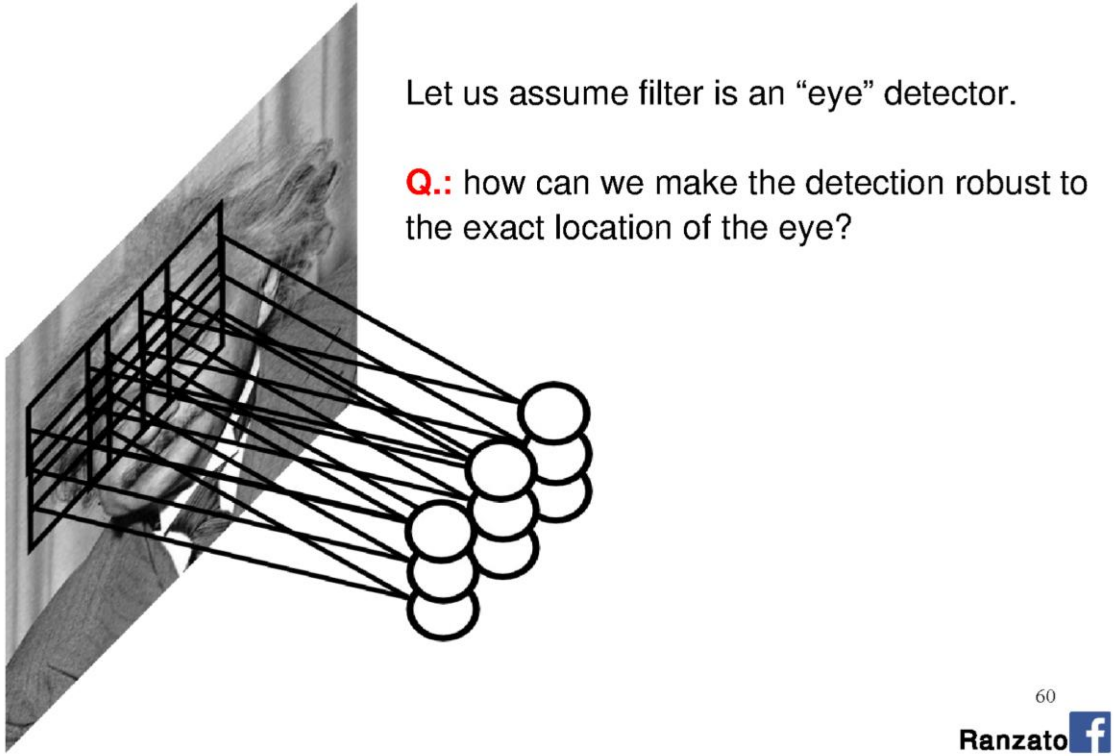
- Properties
  - Input: volume c_i x n_h x n_w (channels x height x width)
  - Hyperparameters: kernels/filters c_o, size k_h x k_w, stride s_h x s_w, zero padding p_h x p_w
  - Output: volume c_o x m_h x m_w (channels x height x width)
  - Parameters: k_h x k_w x c_i per filter, total $(k_h$ x k_w x $c_i)$ x c_o

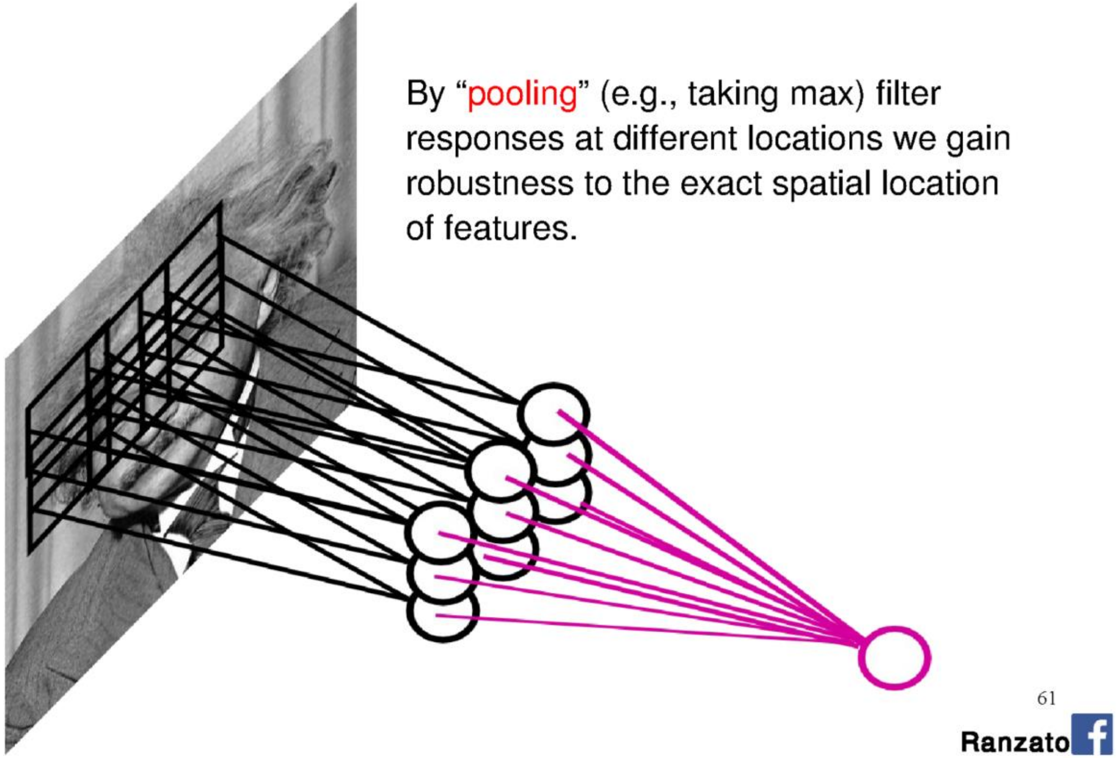


Stanford CS 231n

Other CNN Layers: Pooling

- Another type of layer

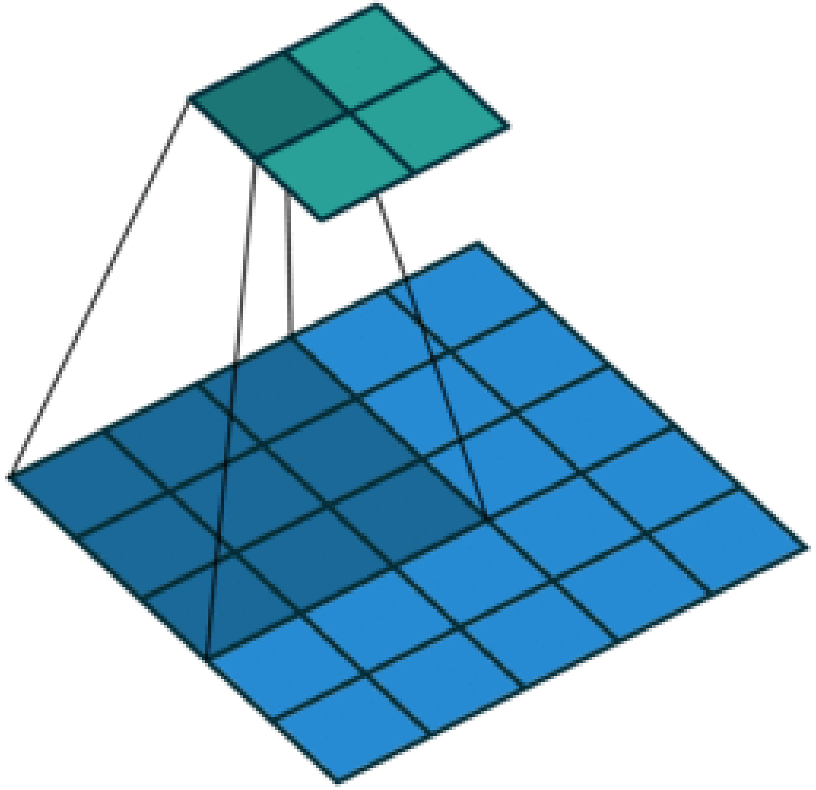




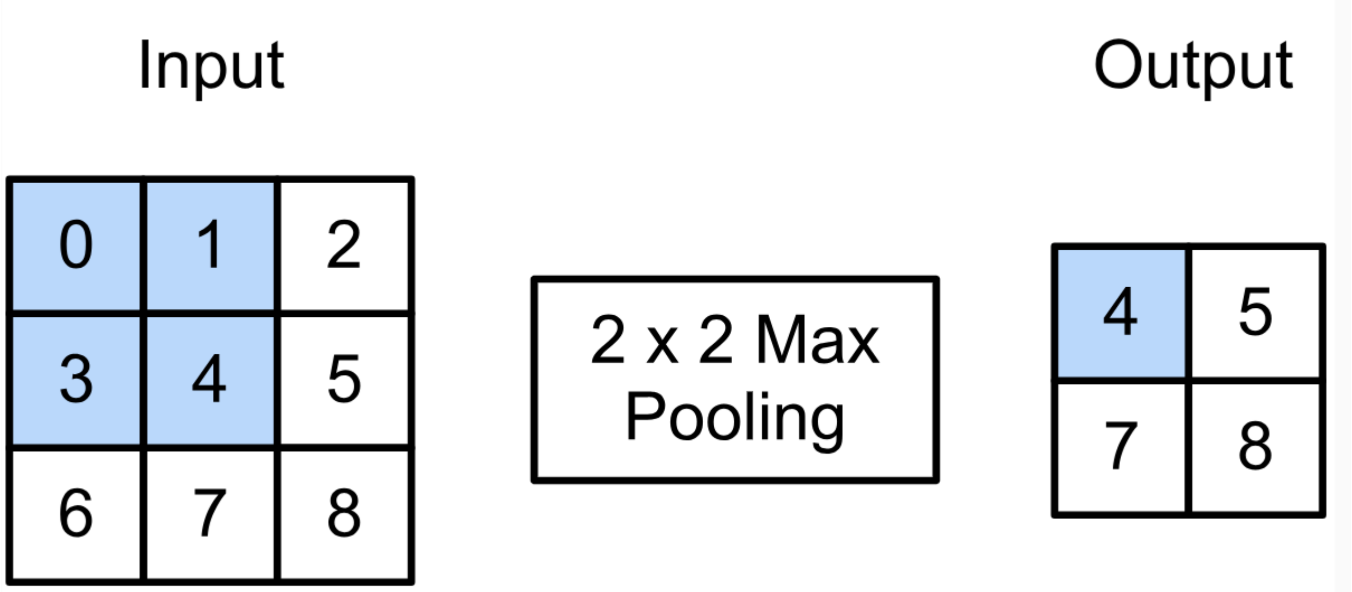
Credit: Marc'Aurelio Ranzato

Max Pooling

- Returns the maximal value in the sliding window

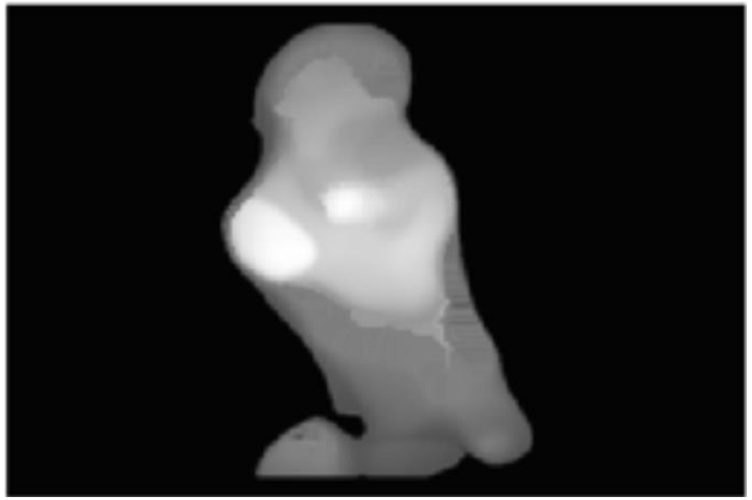


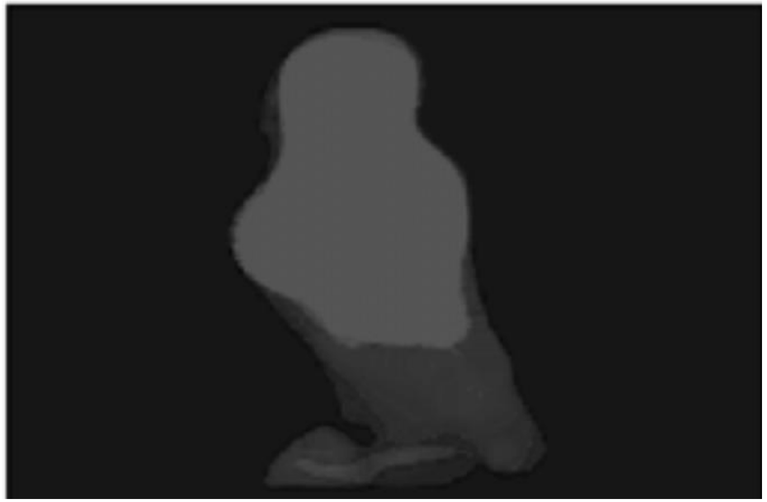
- Example:
  - max(0,1,3,4) = 4



Average Pooling

- Max pooling: the strongest pattern signal in a window
- Average pooling: replace max with mean in max pooling
  - The average signal strength in a window

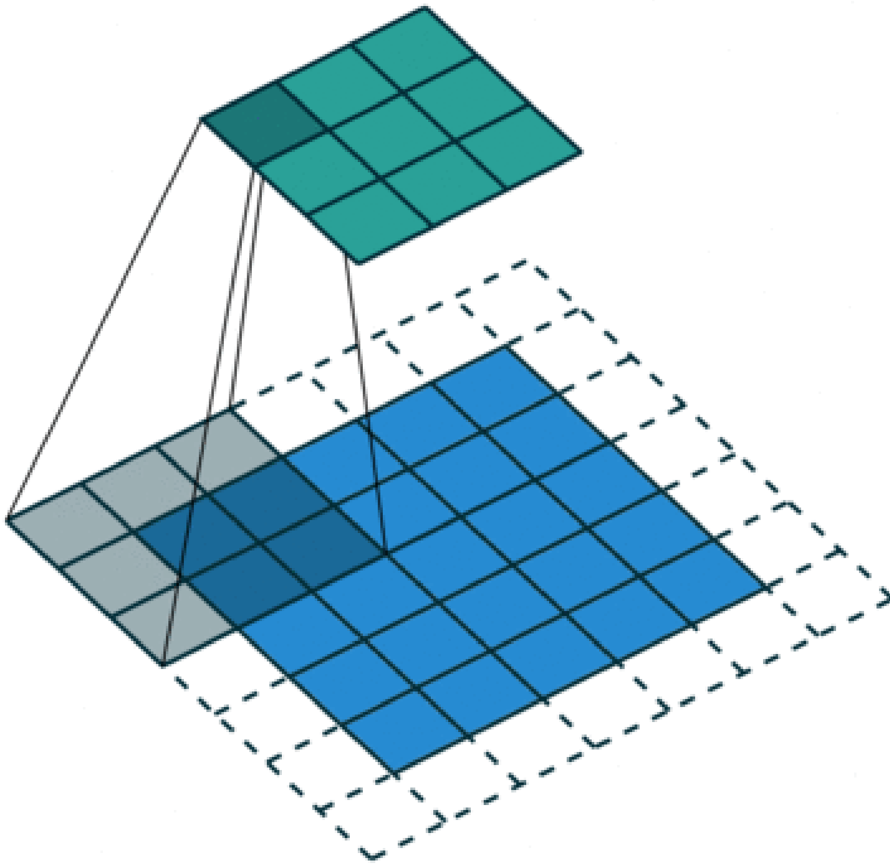




Other CNN Layers: Pooling

- Pooling layers have similar padding and stride as convolutional layers
- No learnable parameters
- Apply pooling for each input channel to obtain the corresponding output channe

#output channels = #input channels



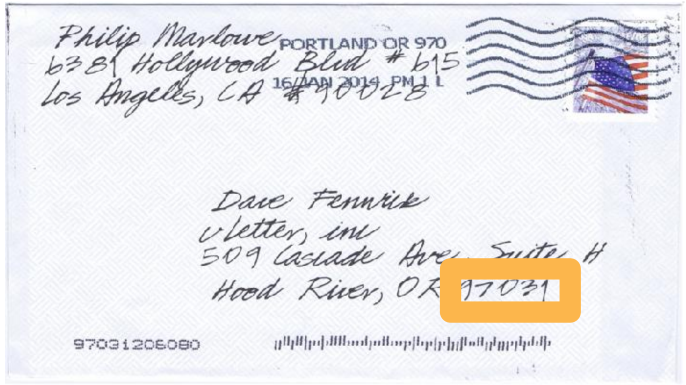
Break & Quiz

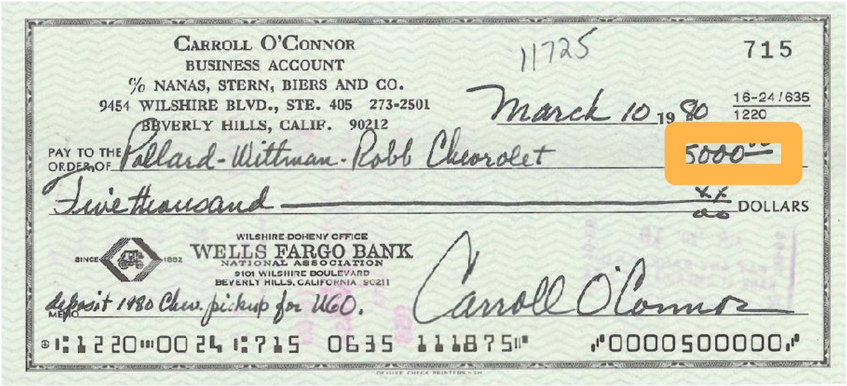
Outline

- **Review & Convolution Operator**
  - Experimental setup, convolution definition, vs. dense layers
- **CNN Components & Layers**
  - Padding, stride, channels, pooling layers
- **CNN Tasks & Architectures**
  - MNIST, ImageNet, LeNet, AlexNet, ResNets

CNN Tasks

- Traditional tasks: handwritten digit recognition
- Dates back to the '70s and '80s
  - Low-resolution images, 10 classes

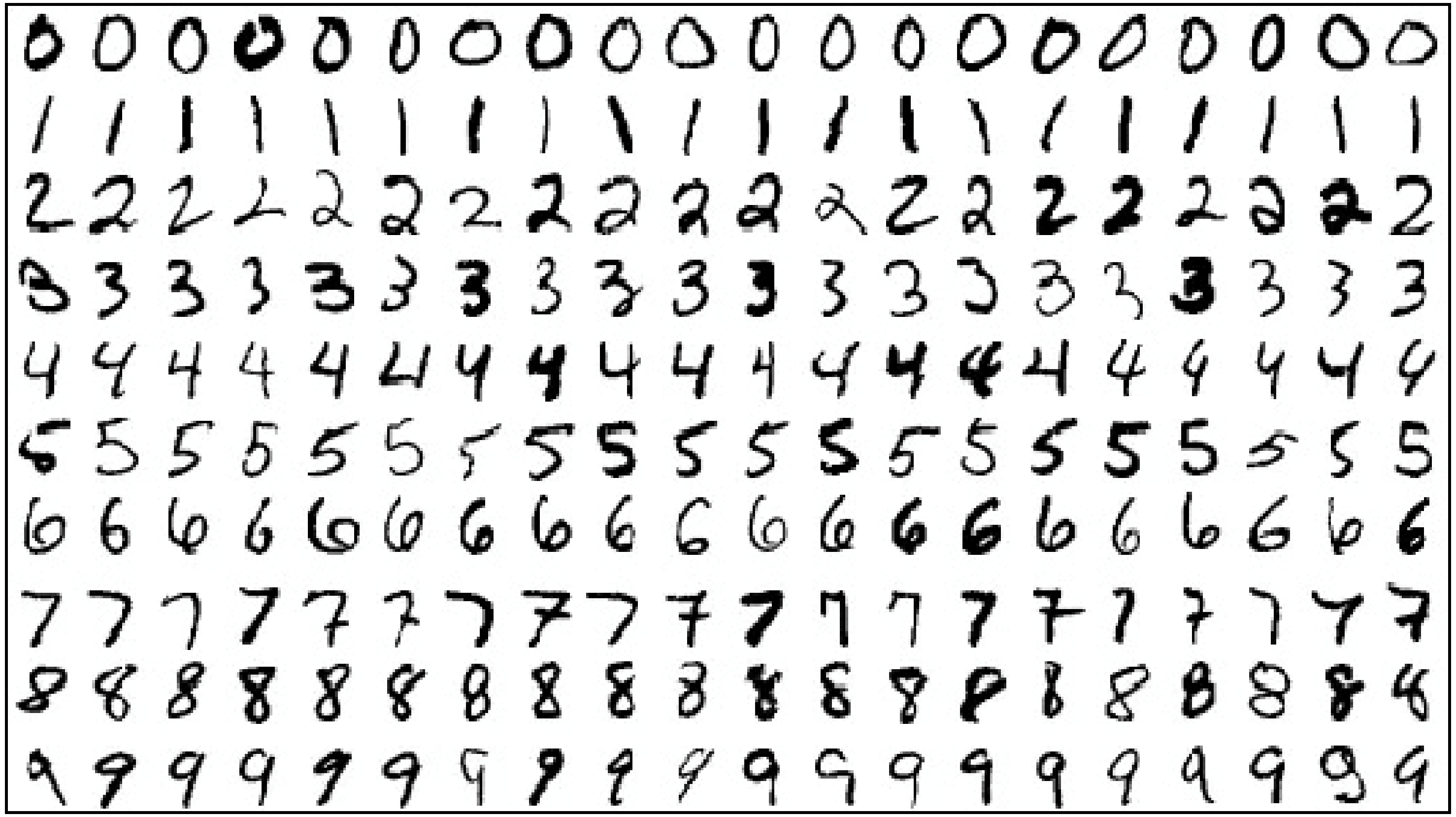




CNN Tasks

- Traditional tasks: handwritten digit recognition
- Classic dataset: MNIST

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

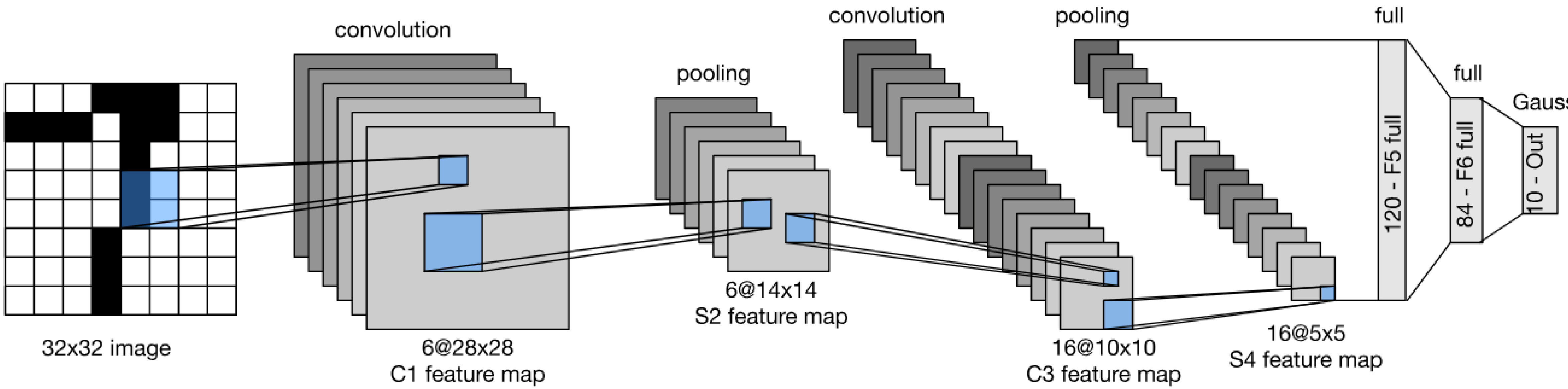


CNN Architectures

- Traditional tasks: handwritten digit recognition
- Classic dataset: MNIST
- 1989-1999: LeNet model

LeCun, Y et al. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation

LeCun, Y.; Bottou, L.; Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proc. IEEE



LeNet in PyTorch

- Pretty easy!
- Setup:

```
def __init__(self):
    super(LeNet5, self).__init__()
    # Convolution (In LeNet-5, 32x32 images are given as input. Hence padding of 2 is done below)
    self.conv1 = torch.nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5, stride=1, padding=2, bias=True)
    # Max-pooling
    self.max_pool_1 = torch.nn.MaxPool2d(kernel_size=2)
    # Convolution
    self.conv2 = torch.nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5, stride=1, padding=0, bias=True)
    # Max-pooling
    self.max_pool_2 = torch.nn.MaxPool2d(kernel_size=2)
    # Fully connected layer
    self.fc1 = torch.nn.Linear(16*5*5, 120)   # convert matrix with 16*5*5 (= 400) features to a matrix of 120 features (columns)
    self.fc2 = torch.nn.Linear(120, 84)       # convert matrix with 120 features to a matrix of 84 features (columns)
    self.fc3 = torch.nn.Linear(84, 10)        # convert matrix with 84 features to a matrix of 10 features (columns)
```

LeNet in PyTorch

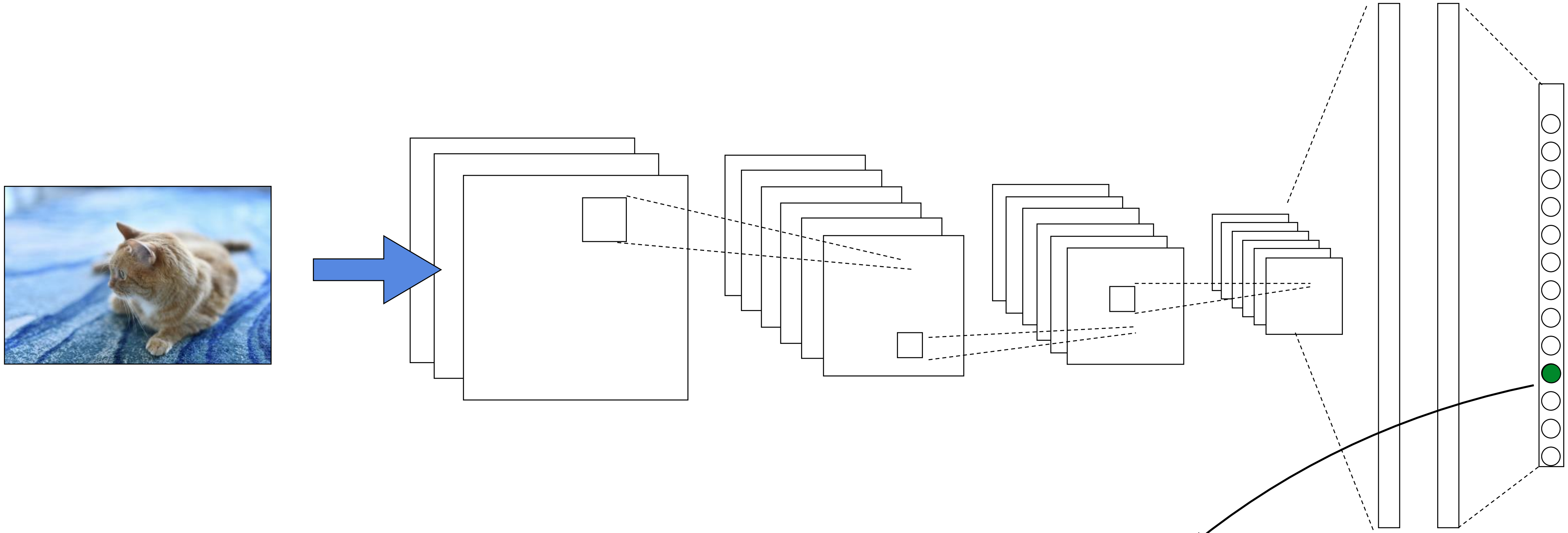
- Pretty easy!
- Forward pass:

```
def forward(self, x):
    # convolve, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.conv1(x))
    # max-pooling with 2x2 grid
    x = self.max_pool_1(x)
    # convolve, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.conv2(x))
    # max-pooling with 2x2 grid
    x = self.max_pool_2(x)
    # first flatten 'max_pool_2_out' to contain 16*5*5 columns
    # read through https://stackoverflow.com/a/42482819/7551231
    x = x.view(-1, 16*5*5)
    # FC-1, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.fc1(x))
    # FC-2, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.fc2(x))
    # FC-3
    x = self.fc3(x)

    return x
```

Training a CNN

- Q: so we have a bunch of layers. How do we train?
- A: same as before. Apply softmax at the end, use backprop.



$$p_i(\boldsymbol{x}) = \frac{\exp\left(f_i(\boldsymbol{x})\right)}{\sum_{j=1}^{N} \exp\left(f_j(\boldsymbol{x})\right)},$$

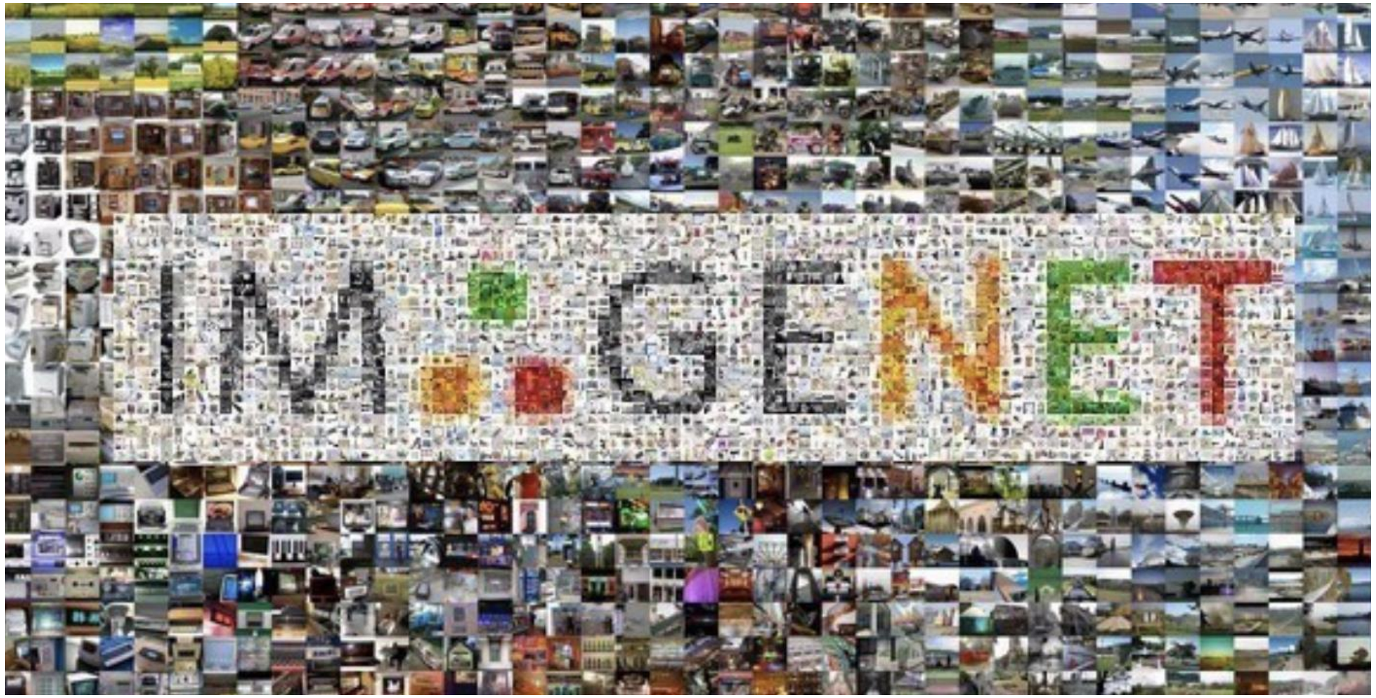
softmax

More CNN Architectures: ImageNet Task

- Next big task/dataset: image recognition on ImageNet
- Large Scale Visual Recognition Challenge (ILSVRC) 2012-2017

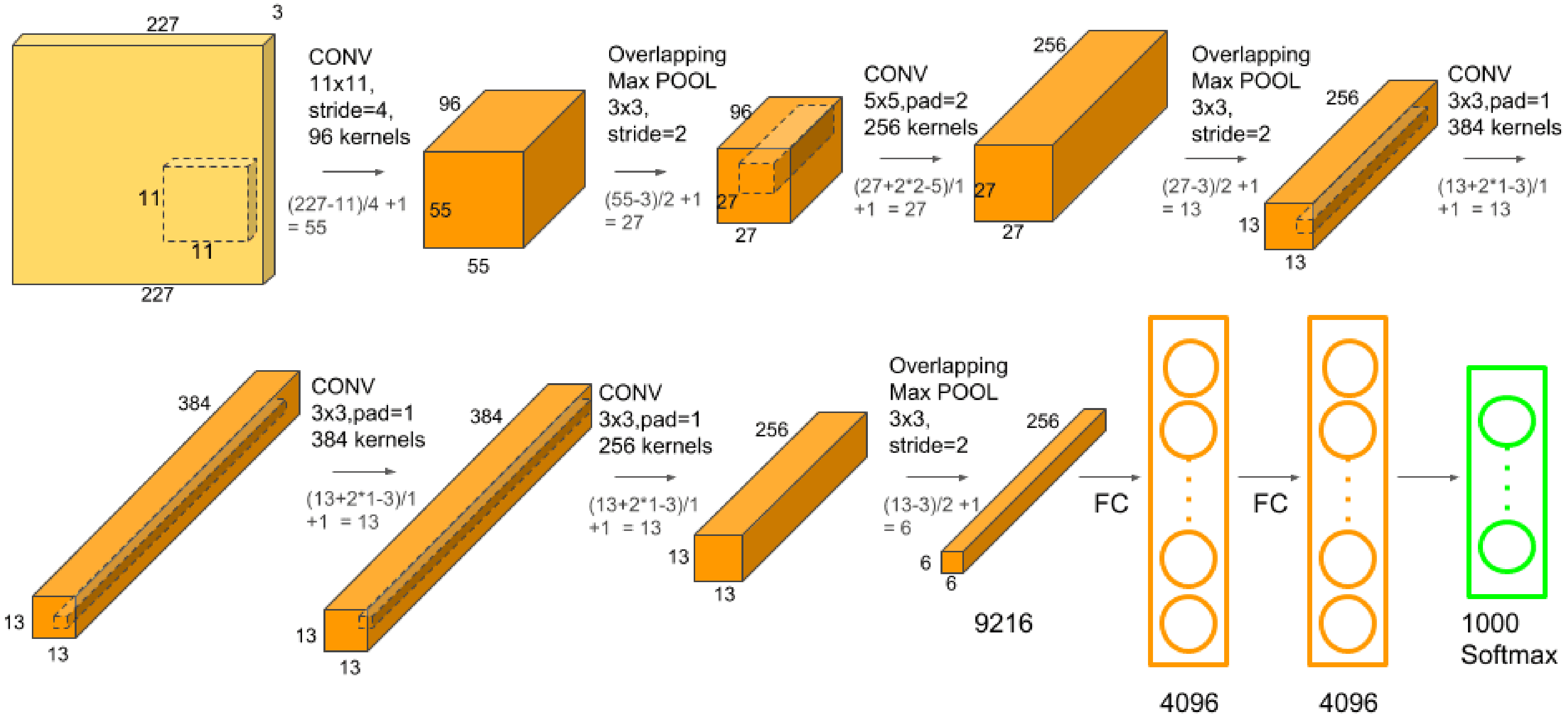
- Properties:
  - Thousands of classes
  - Full-resolution
  - 14,000,000 images

- Started 2009 (Deng et al)



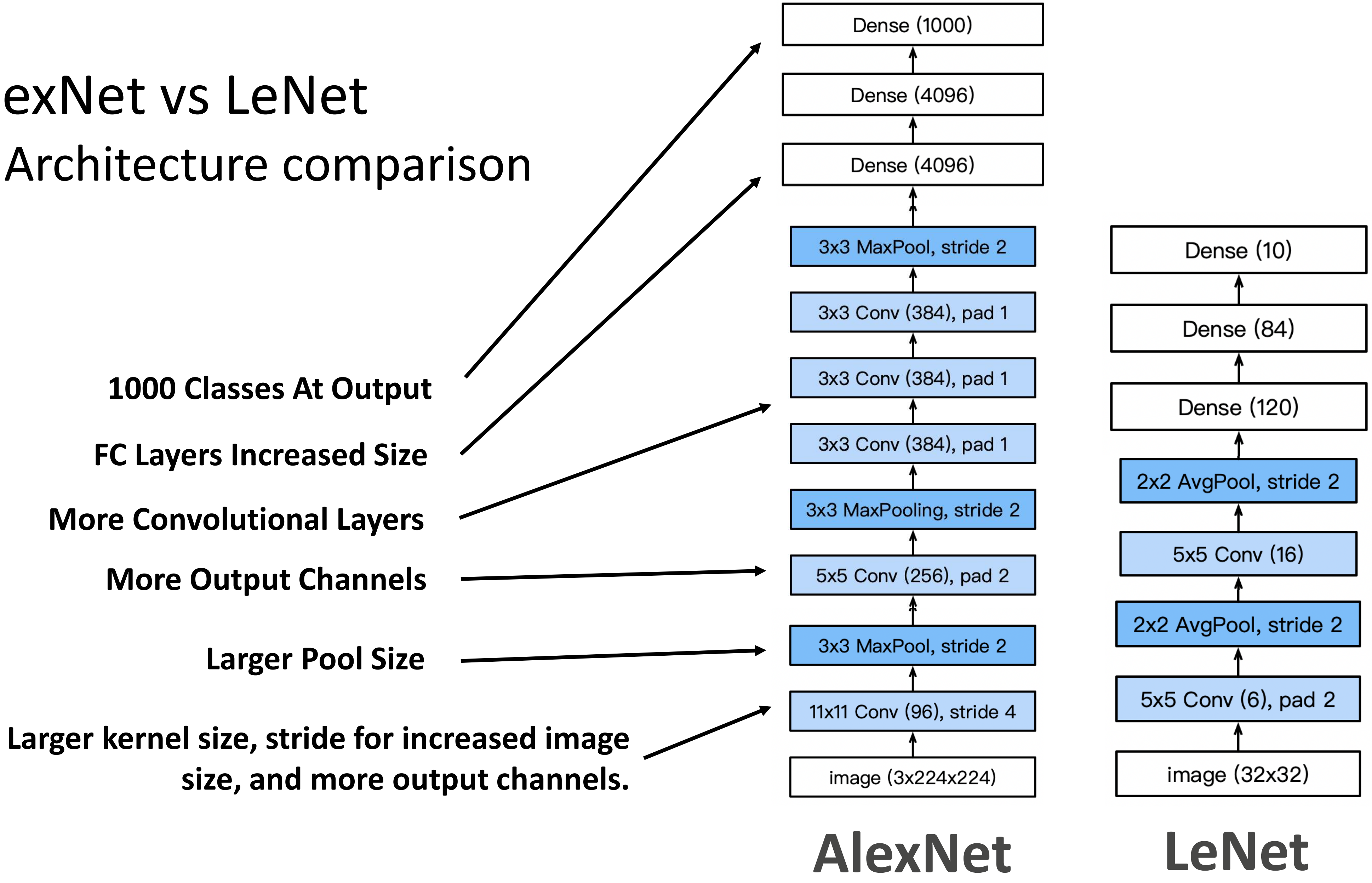
CNN Architectures: AlexNet

- First of the major advancements: AlexNet
- Wins 2012 ImageNet competition
- Major trends: deeper, bigger LeNet



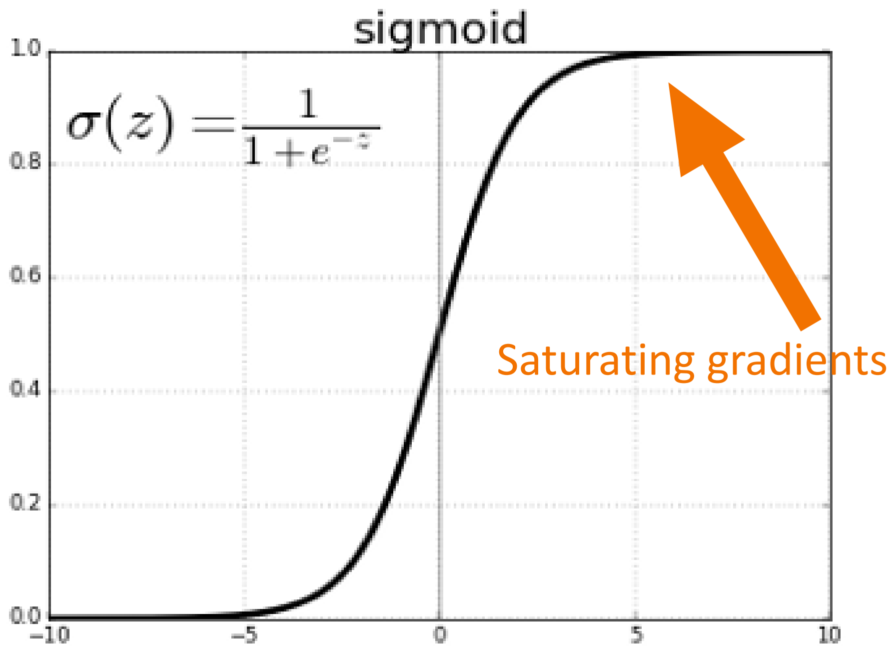
More CNN Architectures

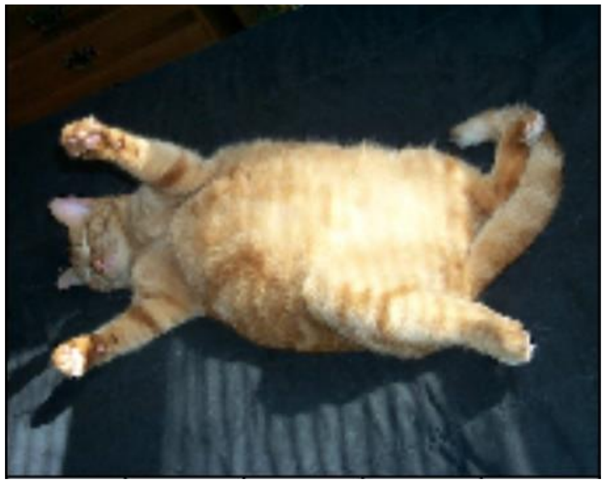
- AlexNet vs LeNet
  - Architecture comparison

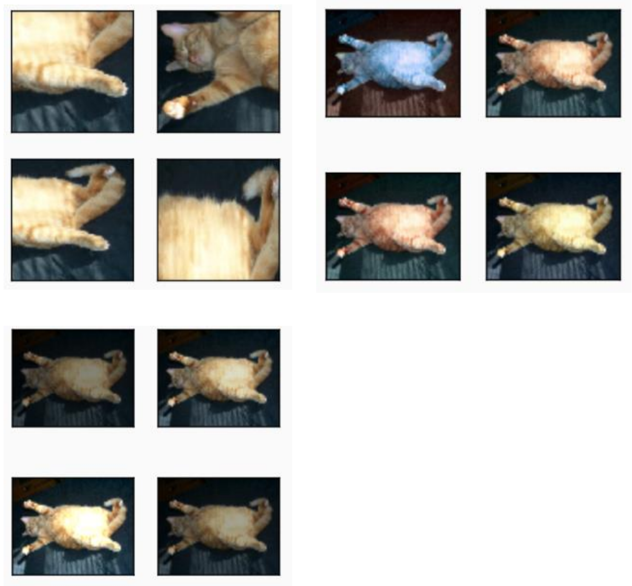


More Differences

- Activations: from sigmoid to ReLU
  - Deal with vanishing gradient issue
- Data Augmentation

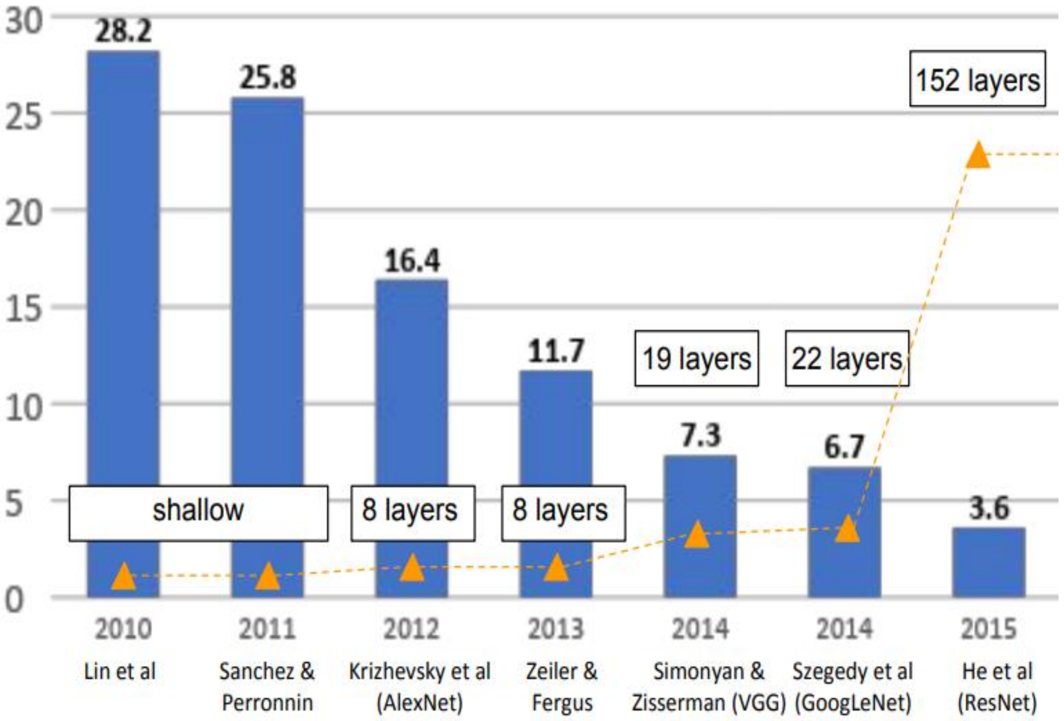






Going Further

- ImageNet error rate
  - Competition winners; note layer count on right.



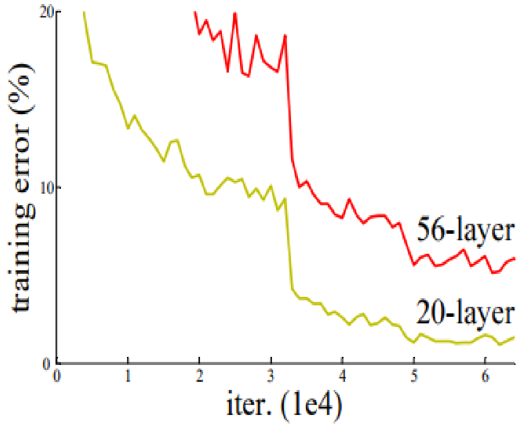
Credit: Stanford CS 231n

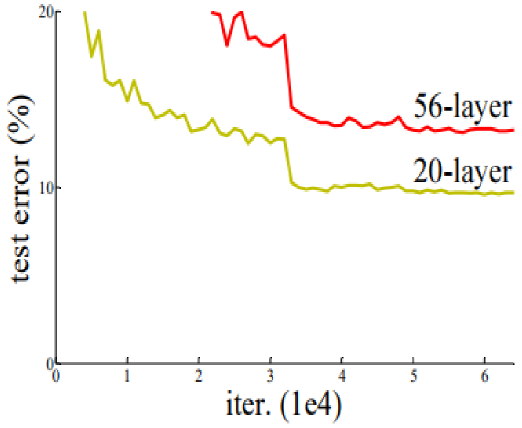
Add More Layers: Enough?

VGG: 19 layers. ResNet: 152 layers. **Add more layers**… sufficient?

- No! Some problems:
  - i) Vanishing gradients: more layers ➔ more likely
  - ii) Instability: can't guarantee we learn **identity** maps

Reflected in training error:



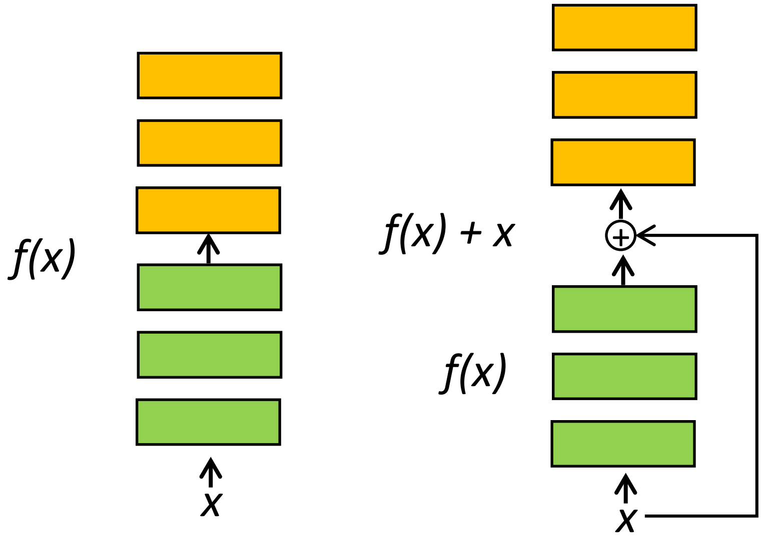


He et al: "Deep Residual Learning for Image Recognition"

Residual Connections

Idea: adding layers can't make worse if we can learn identity

- But, might be hard to learn identity
- Zero map is easy…
  - Make all the weights tiny, produces zero for output



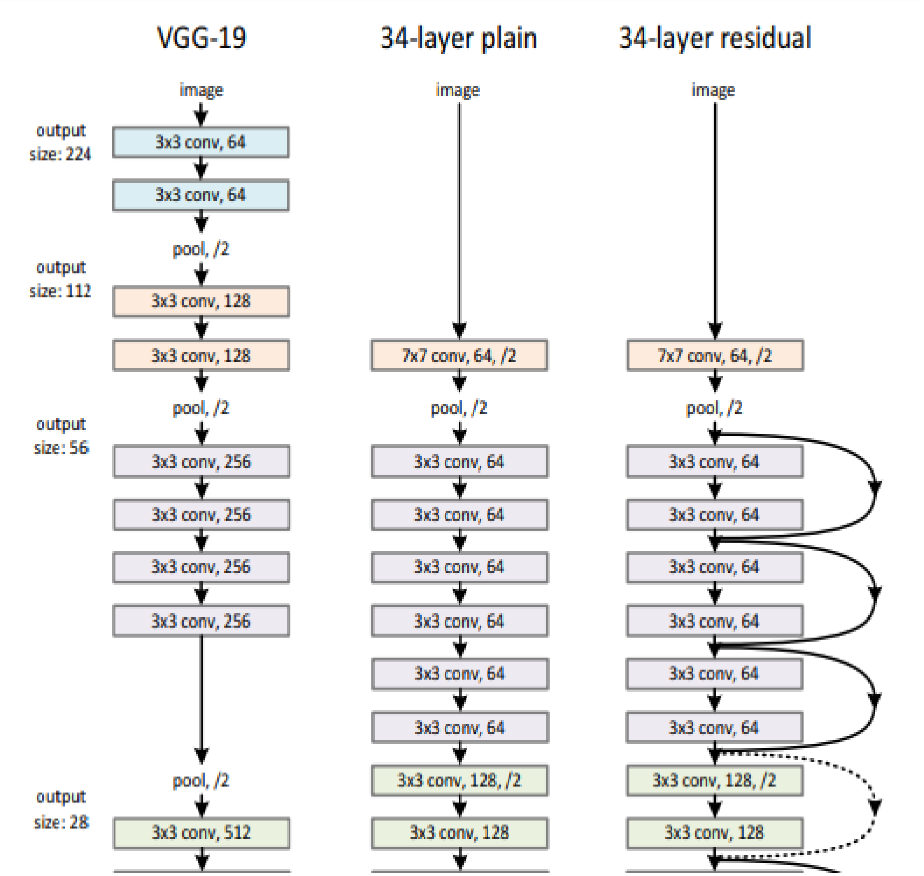
Left: Conventional layers block

Right: **Residual** layer block

To learn identity $f(x) = x$, layers now need to learn $f(x) = 0$ ➔ easier

ResNet Architecture

- **Idea**: Residual (skip) connections help make learning easier
- Example architecture:
- Note: residual connections
  - Every two layers for ResNet34
- Vastly better performance
  - No additional parameters!
  - Records on many benchmarks



He et al: “Deep Residual Learning for Image Recognition”

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov , Sharon Li