Introduction to Deep Learning

11. Convolutional and Pooling Layers

STAT 157, Spring 2019, UC Berkeley

Alex Smola and Mu Li

courses.d2l.ai/berkeley-stat-157

Logistics

Homework 4 winners

1. Andrew Peng (**0.10888**, ensemble of tree models), Andrew Tan (**0.10984**, single MLP), Farbod Nowzad, Ajay Shah
2. Linda Du, Lillian Du, Saurav Kadavath, Ryan Liu (**0.11257**)
3. Yoon Sung Hong, Eric Sung, Jemima Shi, Cheolho Jang (**0.11464**)

Homework 5 Corrections

- **We screwed up**
  Covariate shifted problem between the chosen classes is too easy
- **Fix**
  - Submit updated homework by 3/5/2019
  - Flip classes to make it harder
    'shirt + pullover' vs. 'sandals + sneaker'
    'sneaker + pullover' vs. 'shirt + sandals'

Project Presentations

	Group	Project	Member 1	Member 2	Member 3	Member 4	Member 5
Audio	11	Audio Recovery	Ryan Liu	Saurav Kadavath	Linda Du	Lillian Du	
Image	19	Adversarial examples from adversarial networks	Jesse Zhang	Benson Yuan	Jack Sullivan	Mohammad Arfeen	Mohammed Shaikh
	10	extract meaning information on agriculture produc	Shiyang Ni	Ling Xie	Sean Liu	Jiaqi Guo	
	15	What Are You Thinking: Image Generation By Decoı	Zhanyuan Zhang	Xiaoya Li	Pinshuo Ye	Ganyu (Bruce) Xu	Hsiao-Yu Chiang
	2	Multimodal Deep Learning	Ashley Chien	Shrishti Jeswani	Junseo Park		
NLP	17	Recognition and Translation of Medieval Literature	Anna Shang	Cher Hu	James Li	Haoxi Kuang	
	1	Ensuring Quality Conversations in Online Forums	Brandon Huang	Chen Luyu	Huang Liang	Xu Zhiming	Zeng Zhaorui
	3	Deep Learning with NLP	Hanmaro Song	Minjune Hwang	Kyle Nguyen	Joanne Chen	Kyle Cho
	6	Use machine learning algorithm to give better ranl	Christine Giang	Jobe Cowen	Mandi Zhao	Salim Damerdji	
	8	Yelp Commenting Robot Detection	Mutong Wu	Mike Chen??	Vincent Wu	Robbie Li	Claire Kou
Twitter	13	deep variational auto-encoders modeling distribut	David Nahm	Alice Lyu	Har'el Fishbein		
	14	Twitter Tweets Topic Classification	Yoon Sung Hong	Eric Sung	Jemima Shi	Cheol ho Jang	
Finance	4	Using News to Predict Stock Movements	Zabin Bashar	Jilin Cao	Mike Jin	Daniel Kim	
	9	Forecasting Market Behavior Using Multimodal Sen	Rajan Dutta	Eric Xu	David Chia	Jonathan Stuart	
	12	NLP Sentiment Analysis for Stock Market Predictior	Andrew Tunggal	Derek Tang	Mahir Jethanandani	Sophia Cheng	Stephanie Ortiz
	7	Non-local Neural Networks for Option Volatility Pre	Xinchen Xu	Jiazhong Mei	Liyang Zhao	Ziyang Zhou	Ziqi Pei
RL	16	Solving Open-face Chinese Poker Using Self-play Re	Andrew Tan	Andrew Peng	Farbod Nowzad	Ajay Shah	
Other	5	Explainability in NN Architectures	Ryan Roggenkemper	Hari Subbaraj	Julianna White	Mujahid Zaman	
	18	ACID	Alex Kassil	Catherine Cang	Inna Chernomorets	Derek Topper	
	20		Enrique Lopez	Shukre Ahmed	Frank Chen		

Slide Submission

- Email to **berkeley-stat-157@googlegroups.com**
  - Link to Google Drive slides
    (for template see e.g. https://tinyurl.com/yyv2z5sc)
  - Keynote slides
  - Powerpoint slides
  - PDF (if that's the only thing you can do)


Convolutions

Classifying Dogs and Cats in Images

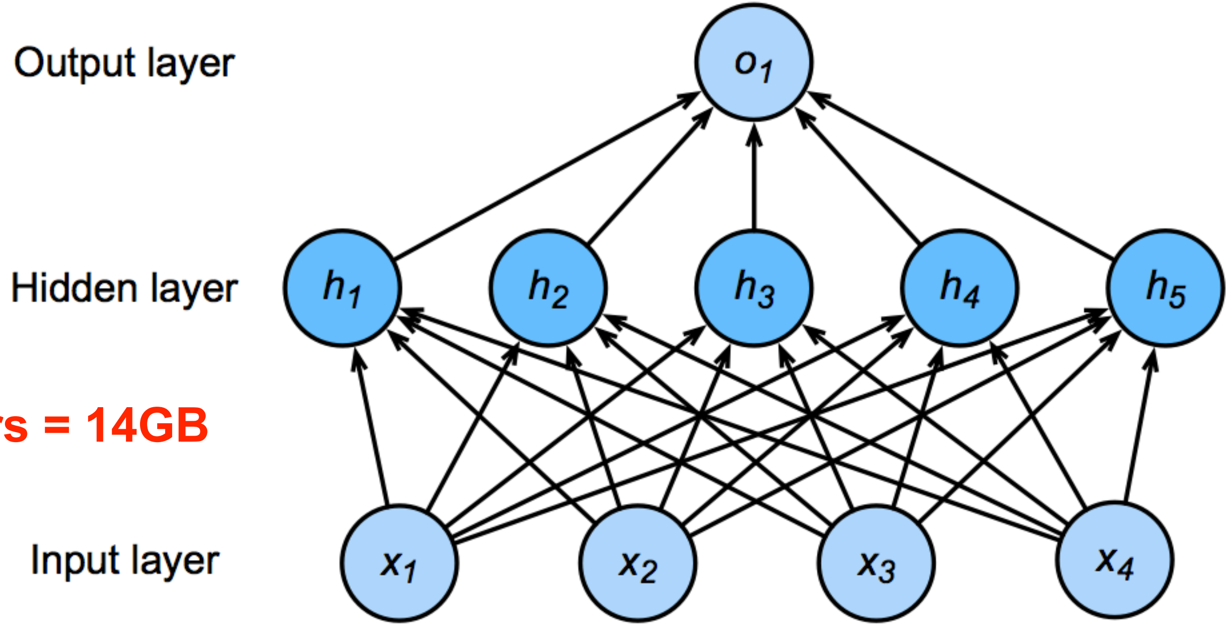
- Use a good camera
- RGB image has 36M elements
- The model size of a single hidden layer MLP with a 100 hidden size is 3.6 Billion parameters
- Exceeds the population of dogs and cats on earth
(900M dogs + 600M cats)

Flashback - Network with one hidden layer

100 neurons

3.6B parameters = 14GB

36M features



$$\mathbf{h} = \sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right)$$

Where is Waldo?

Two Principles

- Translation Invariance
- Locality

Rethinking Dense Layers

- Reshape inputs and output into matrix (width, height)
- Reshape weights into 4-D tensors (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 ~~2-D convolution~~
cross-correlation

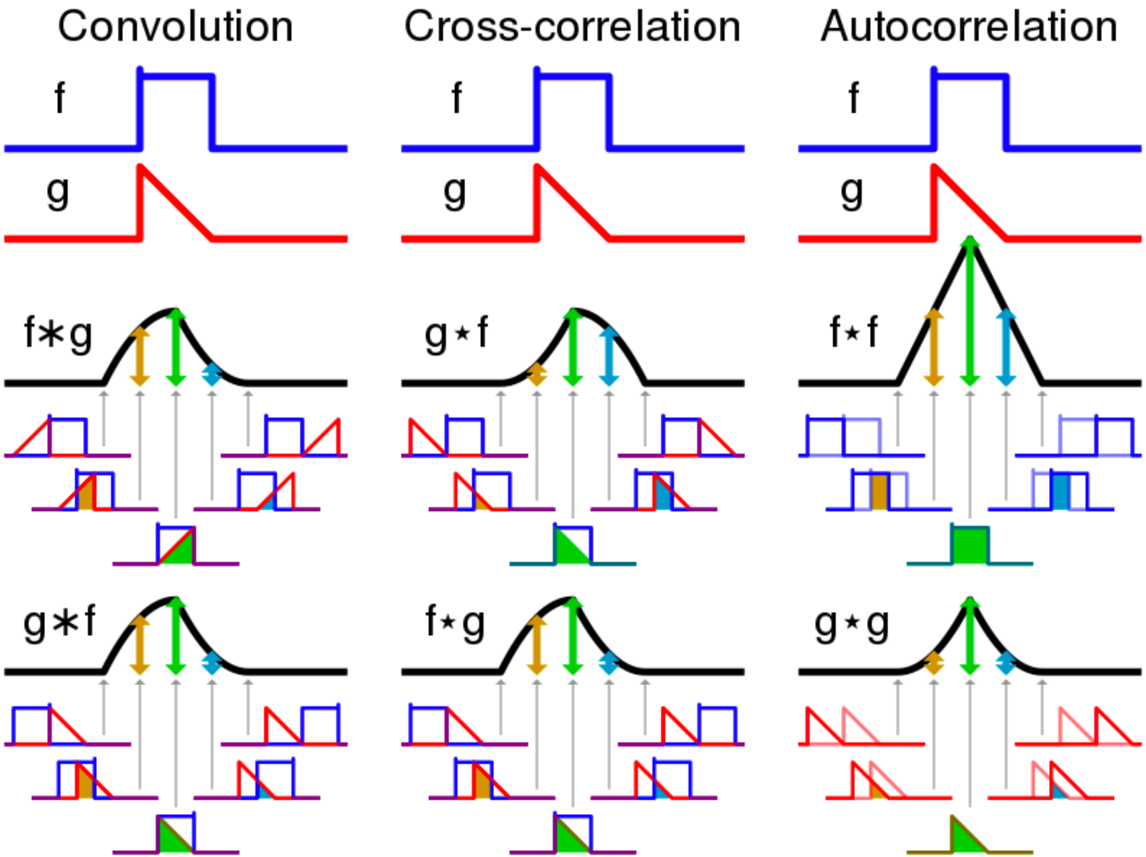
Idea #2 - Locality

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

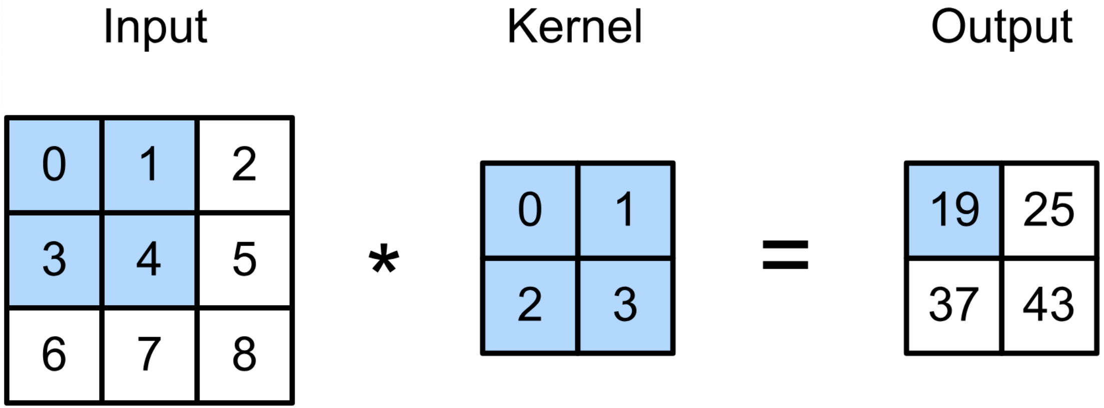
- We shouldn't look very far from x(i,j) in order to assess what's going on at h(i,j)
- Outside range $|a|,|b| > \Delta$ parameters vanish $v_{a,b} = 0$

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

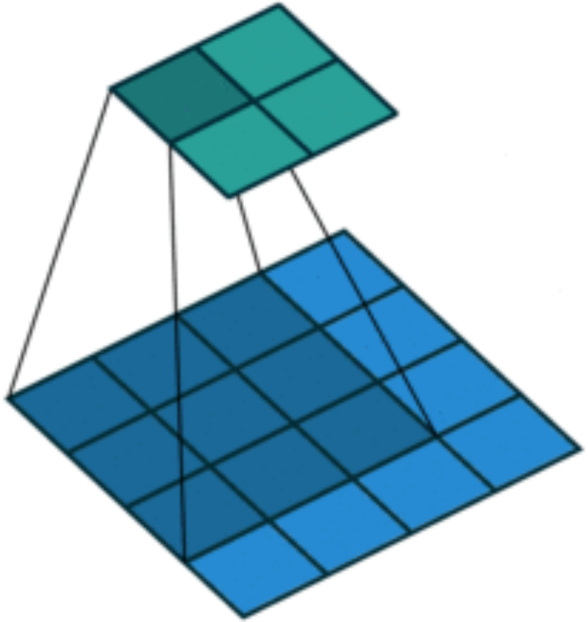
Convolution



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 Cross Correlation

Input

0	1	2
3	4	5
6	7	8

*

Kernel

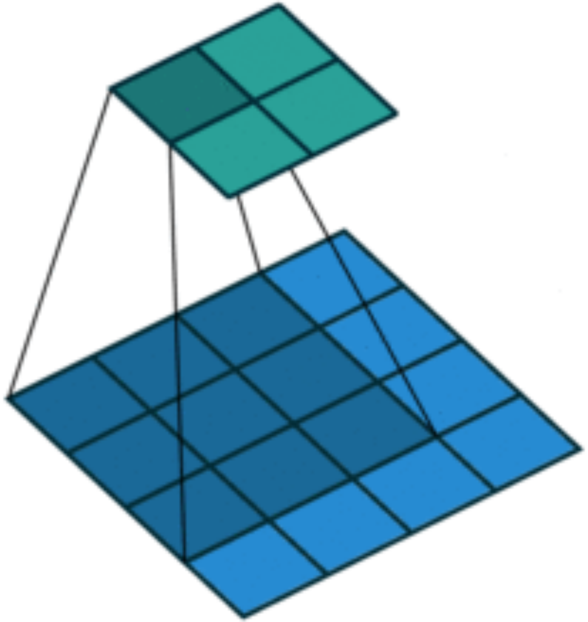
0	1
2	3

=

Output

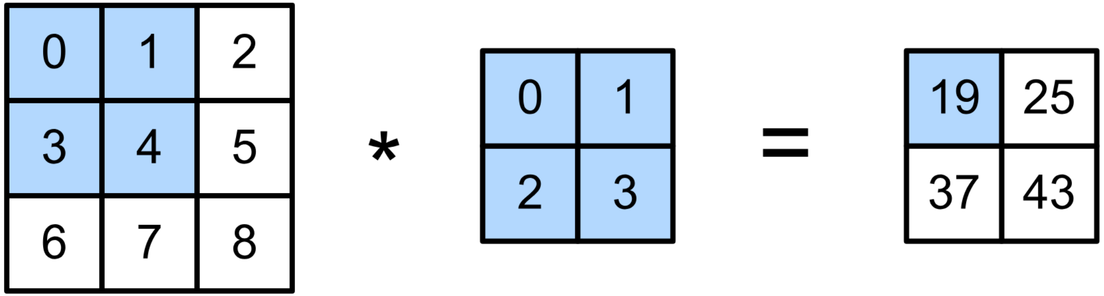
19	25
37	43

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



(vdumoulin@ Github)

2-D Convolution Layer

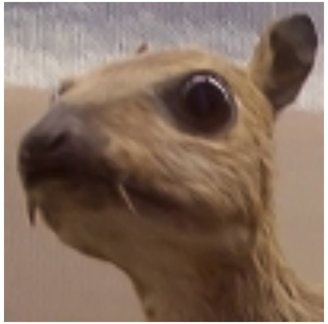


- $\mathbf{X} : n_h \times n_w$ input matrix
- $\mathbf{W} : k_h \times k_w$ kernel matrix
- b: scalar bias
- $\mathbf{Y} : (n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

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

- **W** and *b* are learnable parameters

Examples

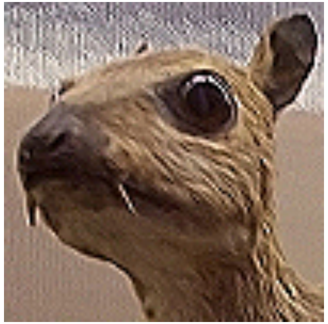


(wikipedia)

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

Edge Detection

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

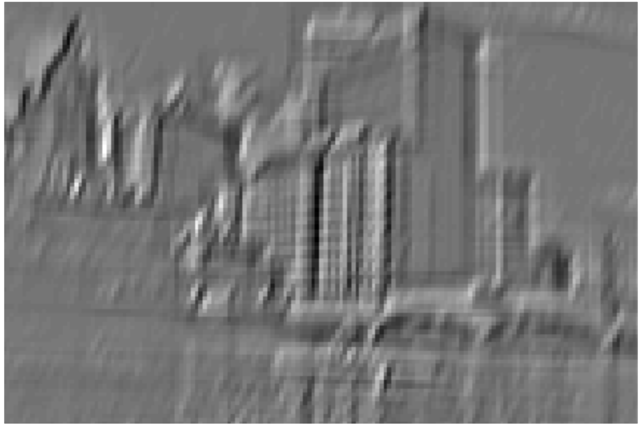


Sharpen

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

Gaussian Blur

Examples



(Rob Fergus)

Cross Correlation vs Convolution

- 2-D Cross Correlation

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

- 2-D Convolution

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

- No difference in practice during to symmetry

1-D and 3-D Cross Correlations

- 1-D

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

  - Text
  - Voice
  - Time series

- 3-D

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

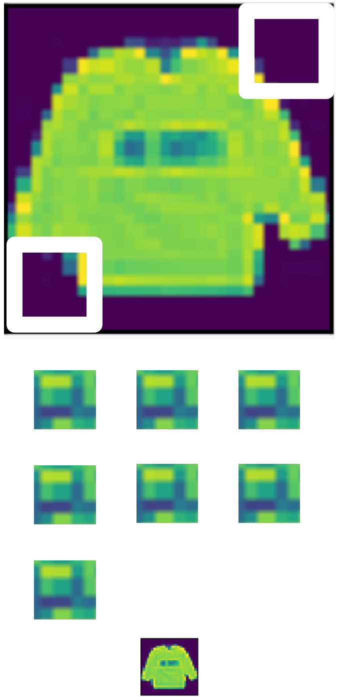
  - Video
  - Medical images


Padding and Stride

Padding

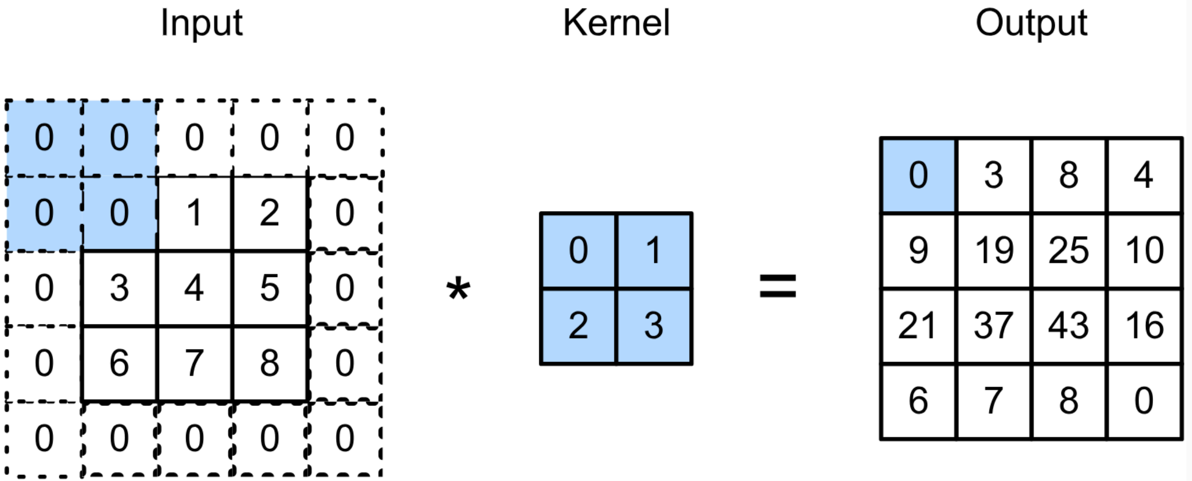
- Given a 32 x 32 input image
- Apply convolutional layer with 5 x 5 kernel
    - 28 x 28 output with 1 layer
    - 4 x 4 output with 7 layers
- Shape decreases faster with larger kernels
    - Shape reduces from $n_h \times n_w$ to

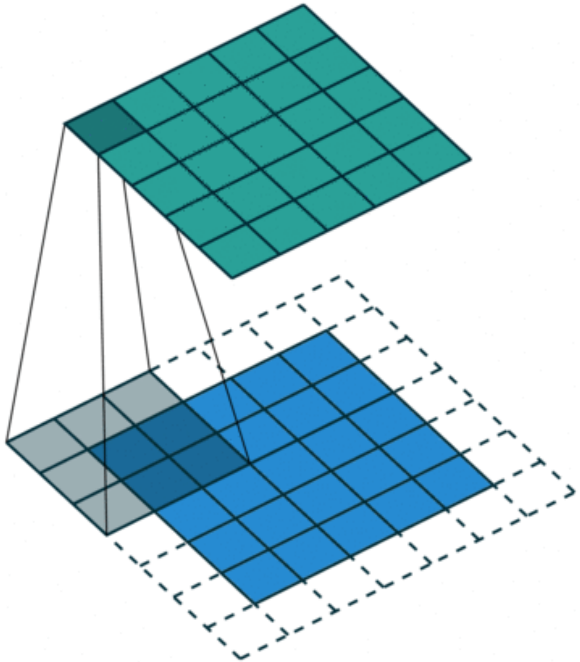
$$(n_h - k_h + 1) \times (n_w - k_w + 1)$$



Padding

Padding adds rows/columns around input

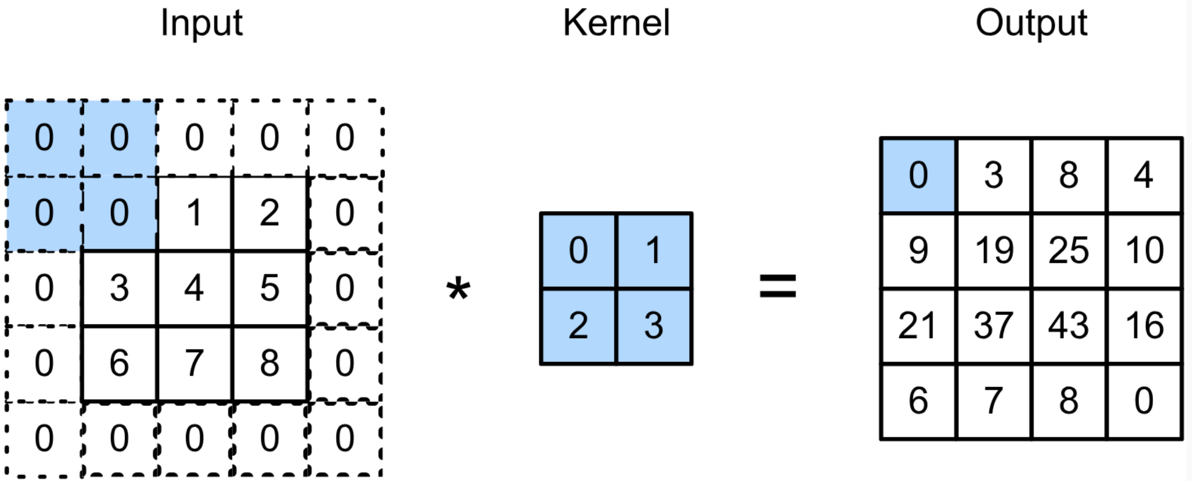


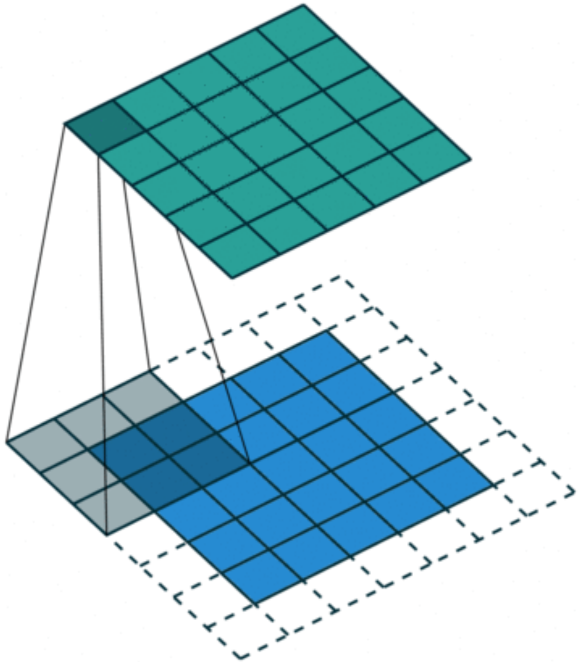


$$0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$$

Padding

Padding adds rows/columns around input





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

Padding

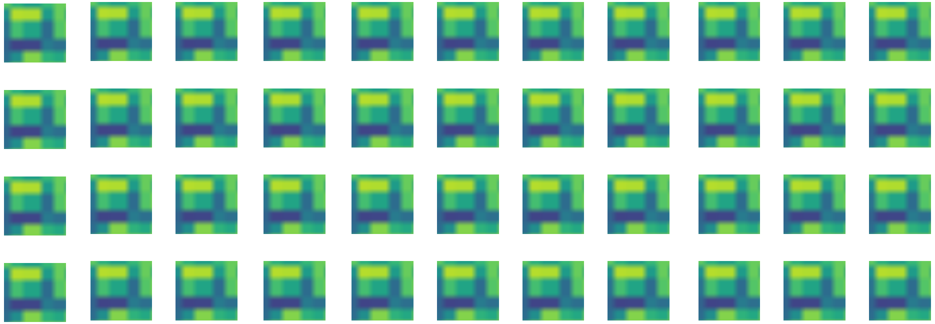
- Padding p_h rows and p_w columns, output shape will be

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

- 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

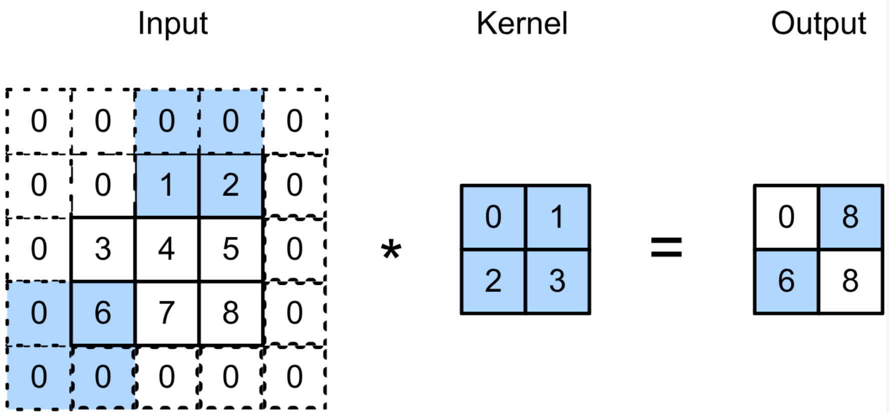
- Padding reduces shape linearly with #layers
  - Given a 224 x 224 input with a 5 x 5 kernel, needs 44 layers to reduce the shape to 4 x 4
  - Requires a large amount of computation



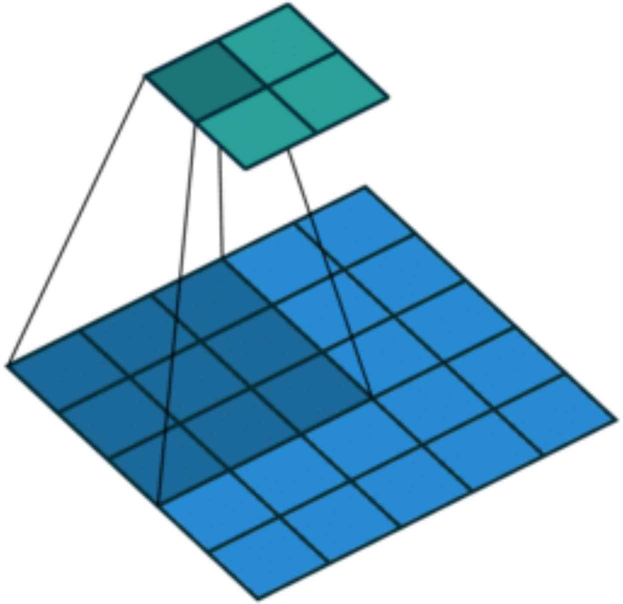
Stride

- Stride is the #rows/#columns per slide

Strides of 3 and 2 for height and width



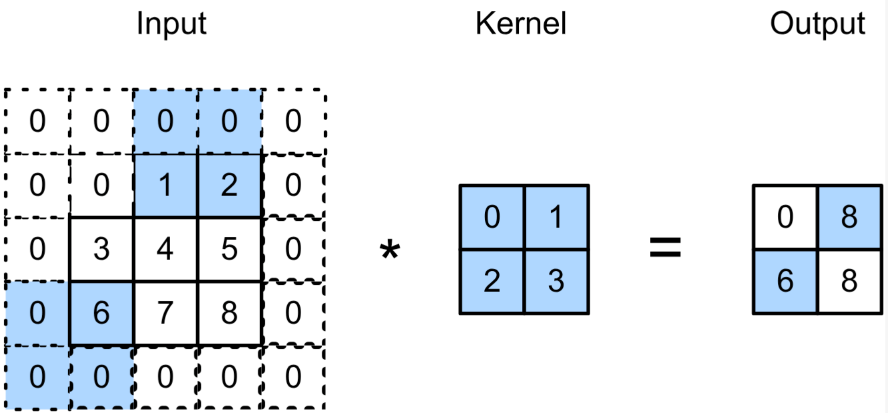
$$0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8$$
$$0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6$$



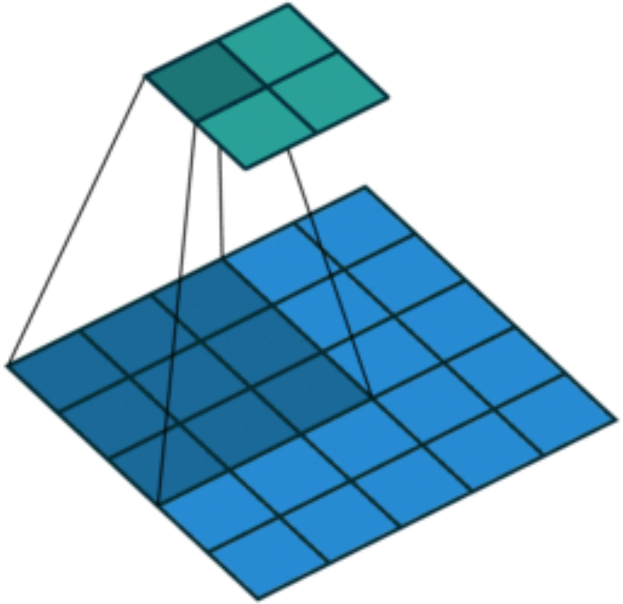
Stride

- 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

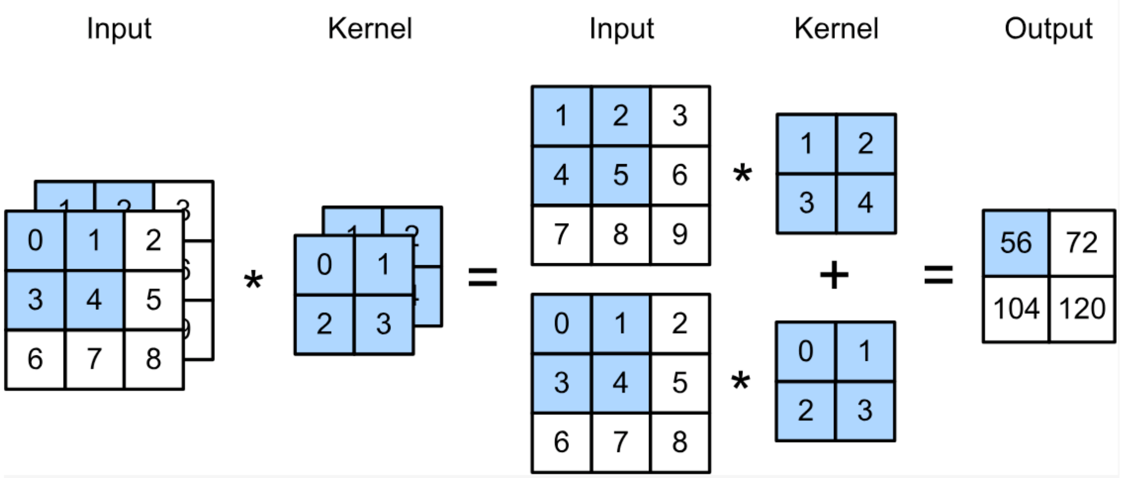
- Color image may have three RGB channels
- Converting to grayscale loses information

Multiple Input Channels

- Color image may have three RGB channels
- Converting to grayscale loses information

Multiple Input Channels

- Have a kernel for each channel, and then sum results over channels



$$(1\times 1 + 2\times 2 + 4\times 3 + 5\times 4)$$
$$+(0\times 0 + 1\times 1 + 3\times 2 + 4\times 3)$$
$$= 56$$

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

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

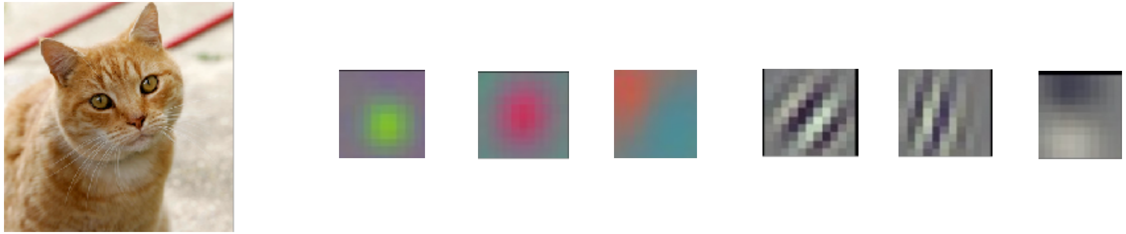
Multiple Output Channels

- No matter how many inputs channels, so far we always get single output channel
- We can have multiple 3-D kernels, each one generates a output channel
- 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 $\mathbf{Y} : c_o \times m_h \times m_w$

$$\mathbf{Y}_{i,:,:} = \mathbf{X} \star \mathbf{W}_{i,:,:,:}$$
$$\text{for } i = 1,\ldots,c_o$$

Multiple Input/Output Channels

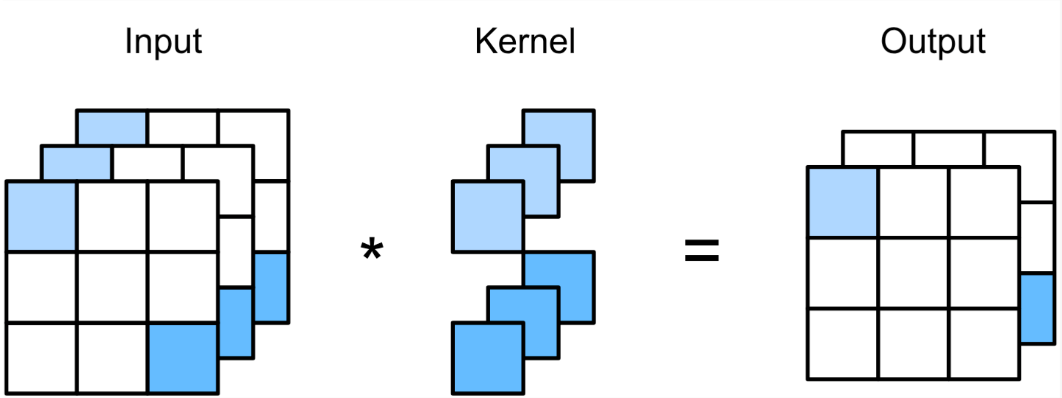
- Each output channel may recognize a particular pattern



- 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 dense layer with $n_h n_w \times c_i$ input and $c_o \times c_i$ weight.

2-D Convolution Layer Summary

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

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

- Complexity (number of floating point operations FLOP)

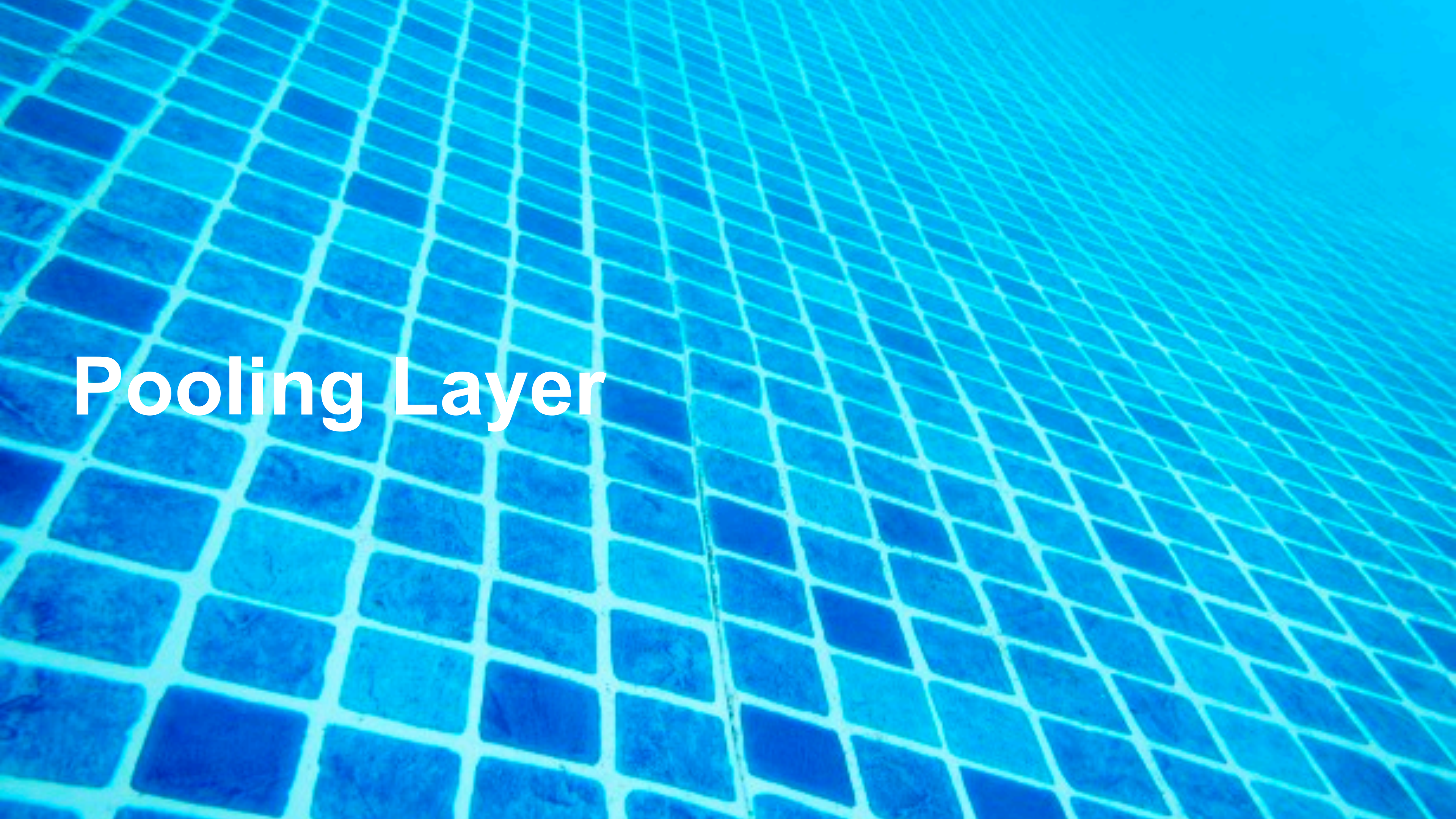
$$c_i = c_o = 100$$
$$k_h = h_w = 5$$
$$m_h = m_w = 64$$

$$O(c_i c_o k_h k_w m_h m_w)$$

1GFLOP

- 10 layers, 1M examples: 10PF
(CPU: 0.15 TF = 18h, GPU: 12 TF = 14min)

Pooling Layer



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

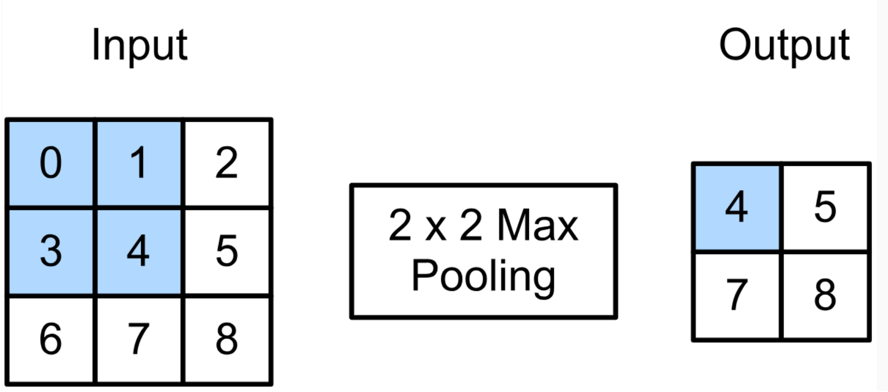
```
[[ 0.   1.   0.   0.
 [ 0.   1.   0.   0.
 [ 0.   1.   0.   0.
 [ 0.   1.   0.   0.
```

0 output with 1 pixel shift

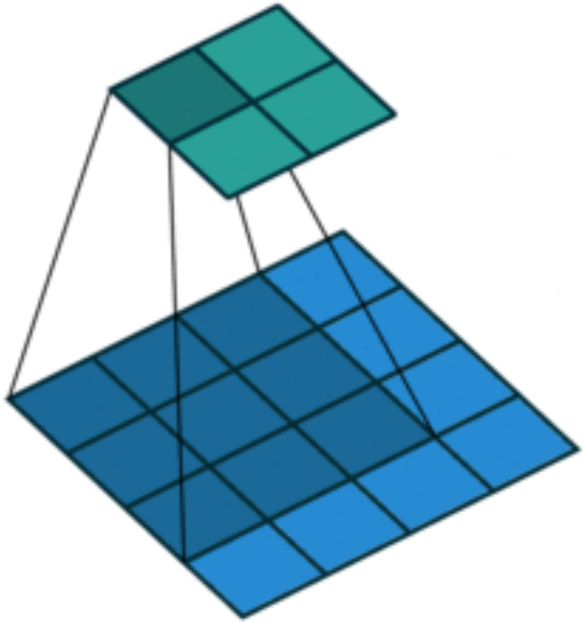
- We need some degree of invariance to translation
  - Lighting, object positions, scales, appearance vary among images

2-D Max Pooling

- Returns the maximal value in the sliding window

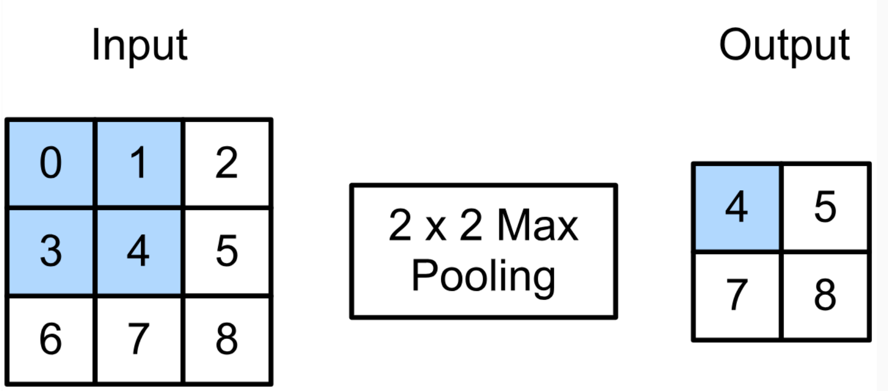


$$\max(0,1,3,4) = 4$$

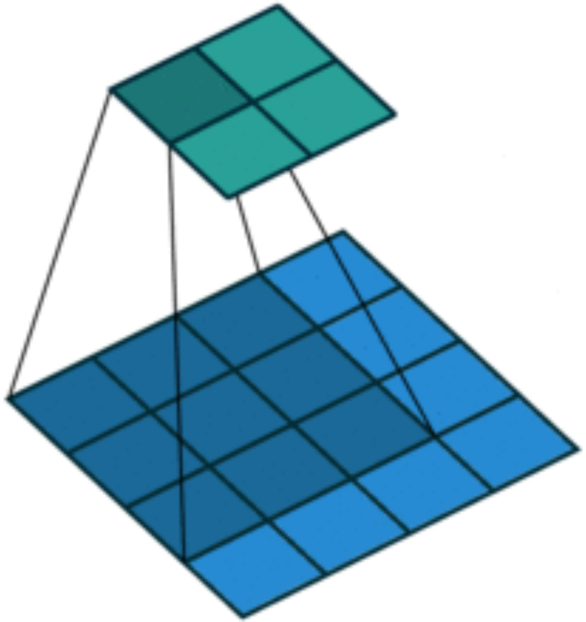


2-D Max Pooling

- Returns the maximal value in the sliding window



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



2-D Max Pooling

- Returns the maximal value in the sliding window

Vertical edge detection

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

Conv output

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

2 x 2 max pooling

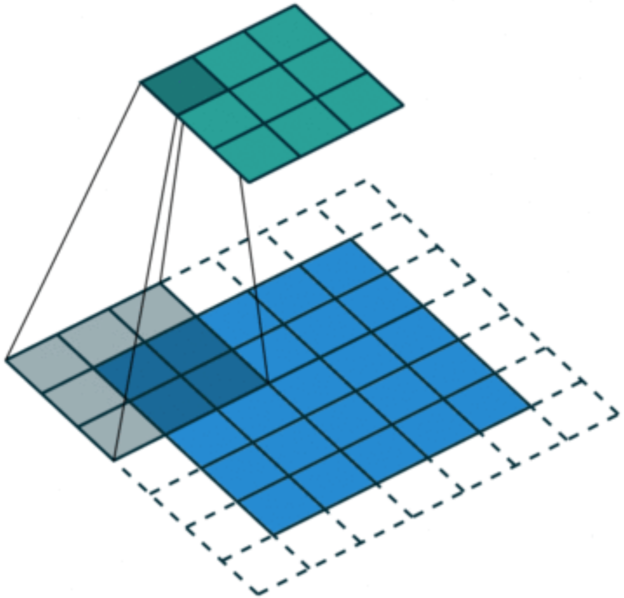
```
[[  1.   1.   1.   0.
 [  1.   1.   1.   0.
 [  1.   1.   1.   0.
 [  1.   1.   1.   0.
```

Tolerant to 1 pixel shift

Padding, Stride, and Multiple Channels

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

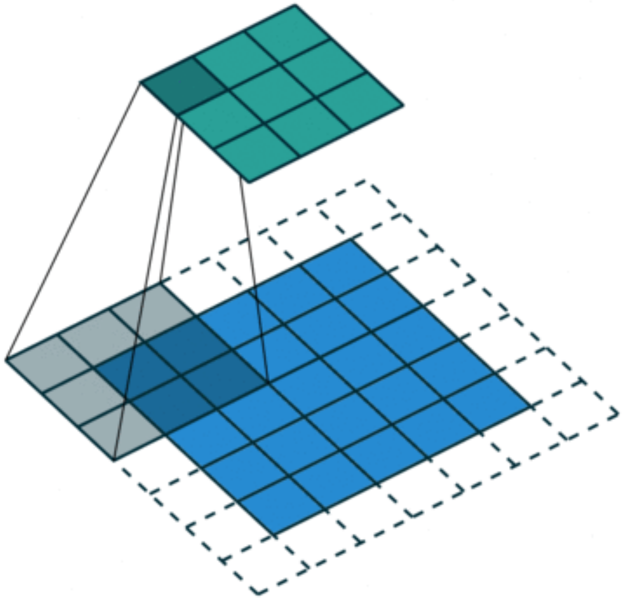
  #output channels = #input channels



Padding, Stride, and Multiple Channels

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

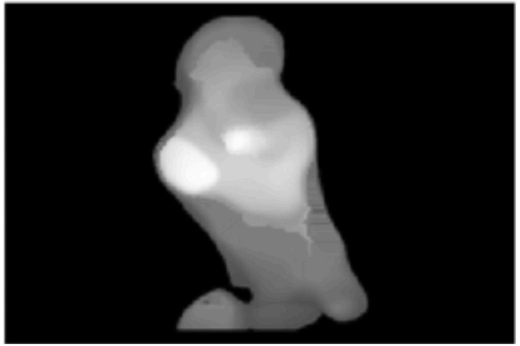
#output channels = #input channels



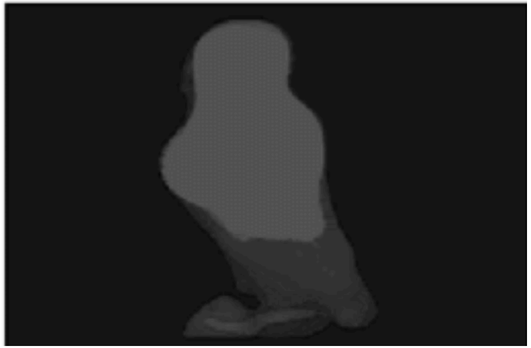
Average Pooling

- 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

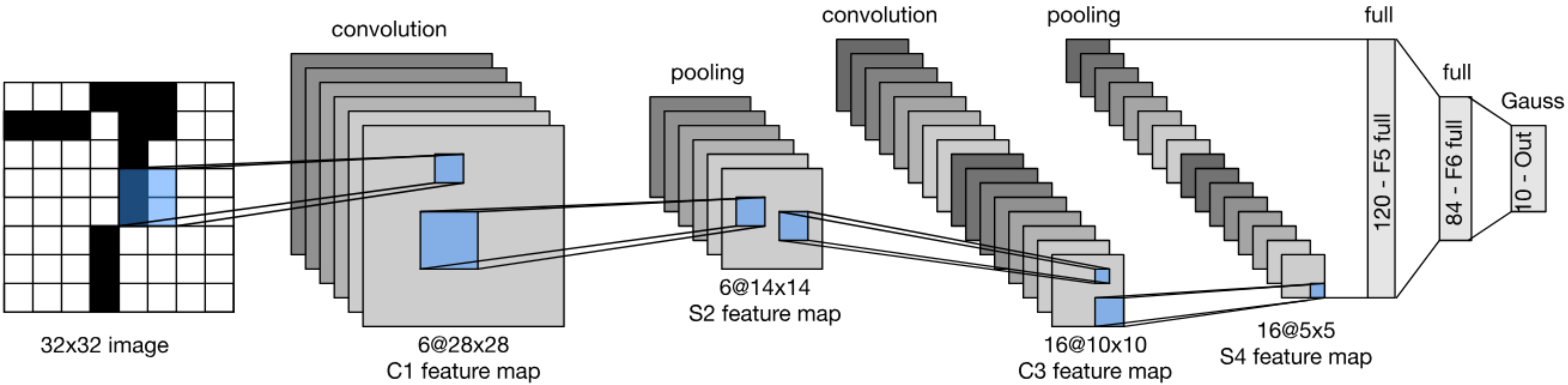
Max pooling



Average pooling

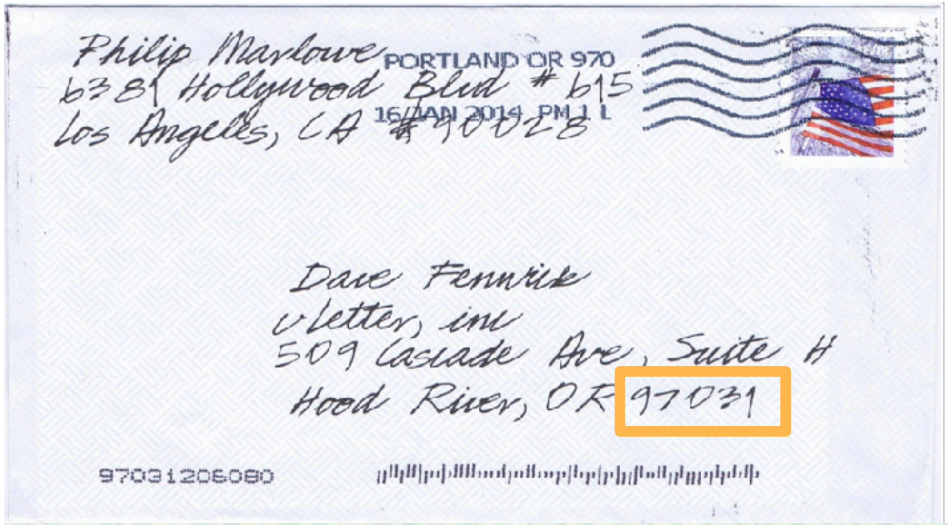


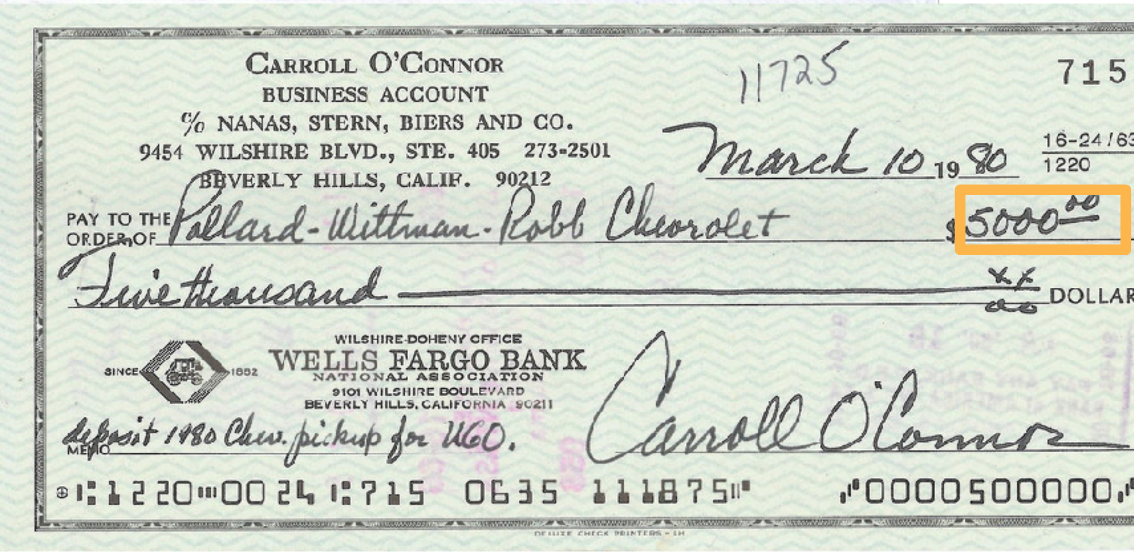
LeNet Architecture



gluon-cv.mxnet.io

Handwritten Digit Recognition

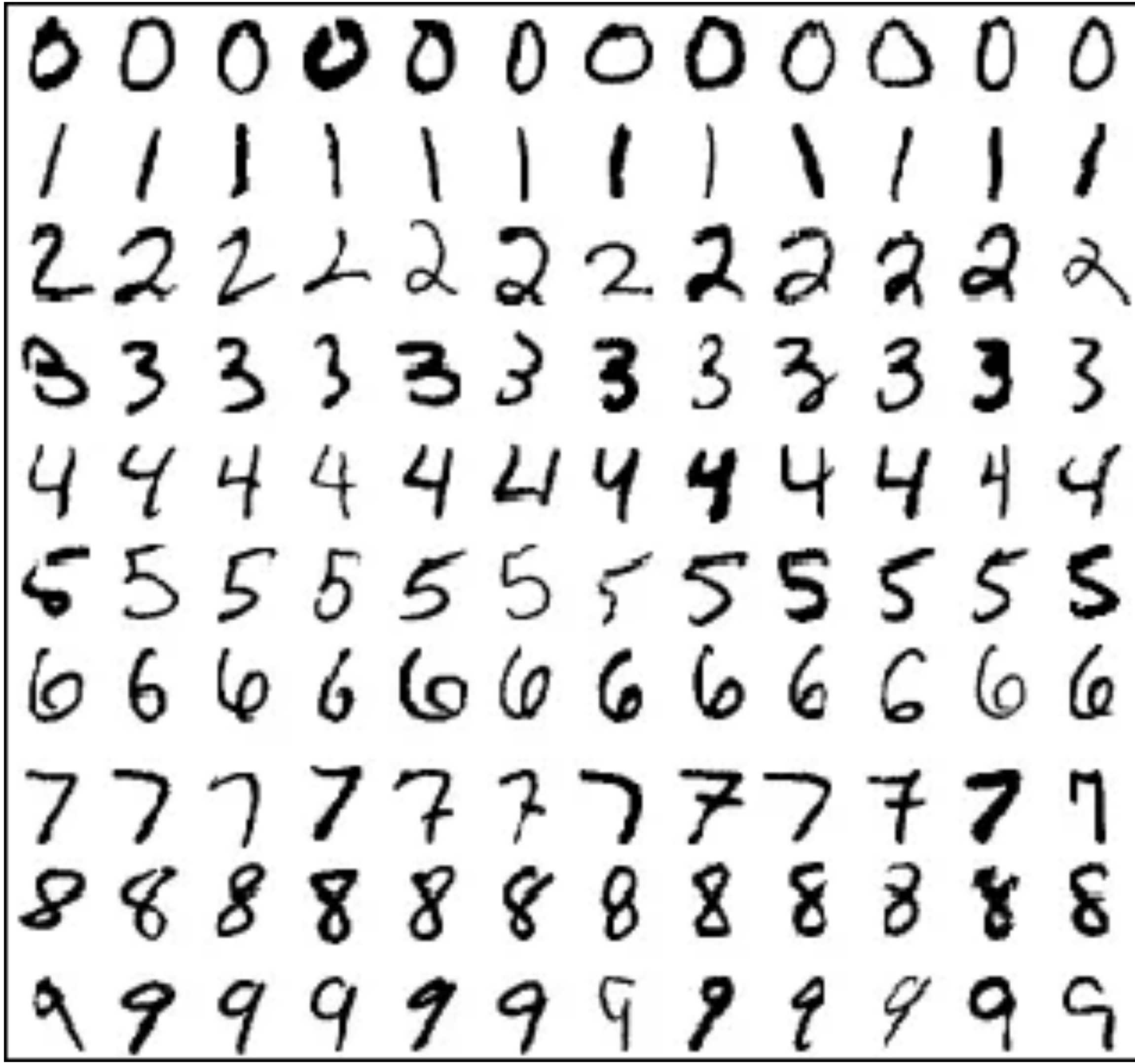


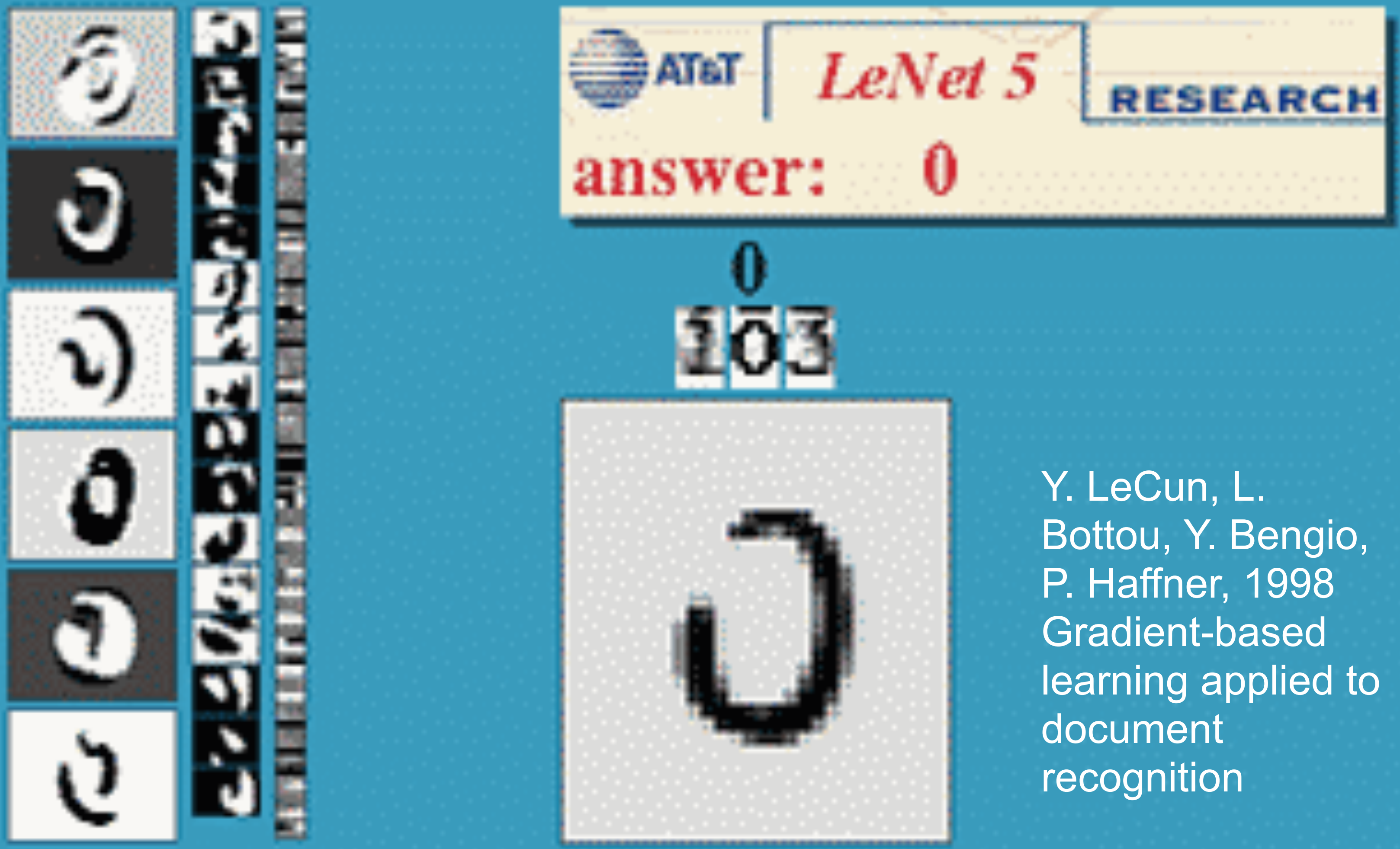


courses.d2l.ai/berkeley-stat-157

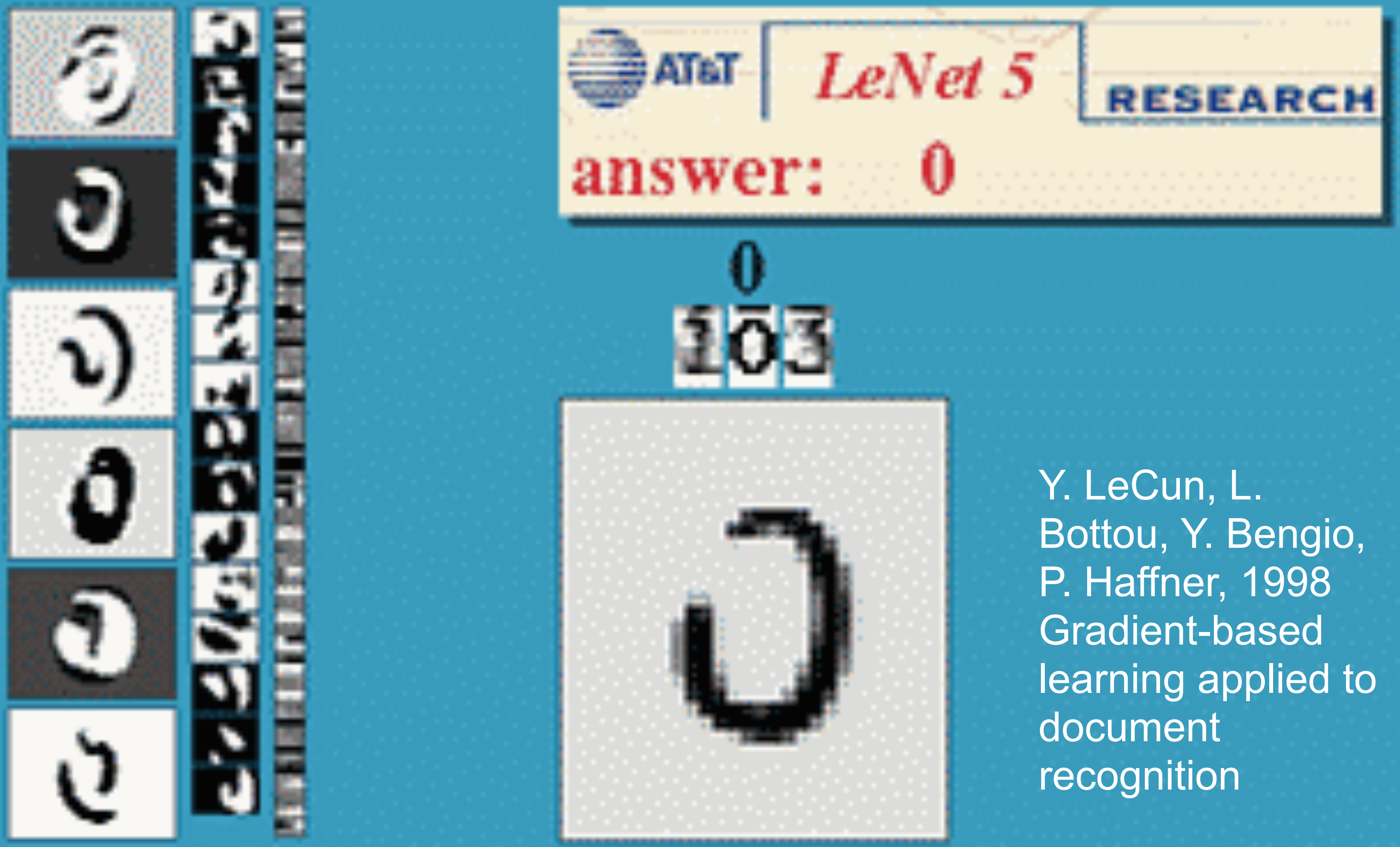
MNIST

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



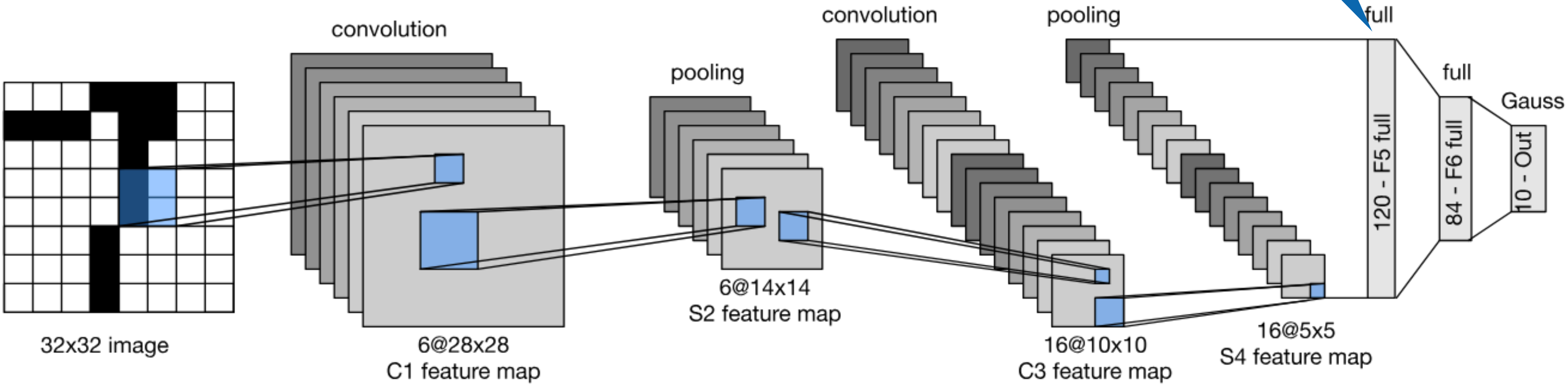


Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998 Gradient-based learning applied to document recognition



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

Expensive if we have many outputs



LeNet in MXNet

```
net = gluon.nn.Sequential()
with net.name_scope():
    net.add(gluon.nn.Conv2D(channels=20, kernel_size=5, activation='tanh'))
    net.add(gluon.nn.AvgPool2D(pool_size=2))
    net.add(gluon.nn.Conv2D(channels=50, kernel_size=5, activation='tanh'))
    net.add(gluon.nn.AvgPool2D(pool_size=2))
    net.add(gluon.nn.Flatten())
    net.add(gluon.nn.Dense(500, activation='tanh'))
    net.add(gluon.nn.Dense(10))

loss = gluon.loss.SoftmaxCrossEntropyLoss()

(size and shape inference is automatic)
```

Summary

- Convolutional layer
  - Reduced model capacity compared to dense layer
  - Efficient at detecting spatial pattens
  - High computation complexity
  - Control output shape via padding, strides and channels
- Max/Average Pooling layer
  - Provides some degree of invariance to translation