

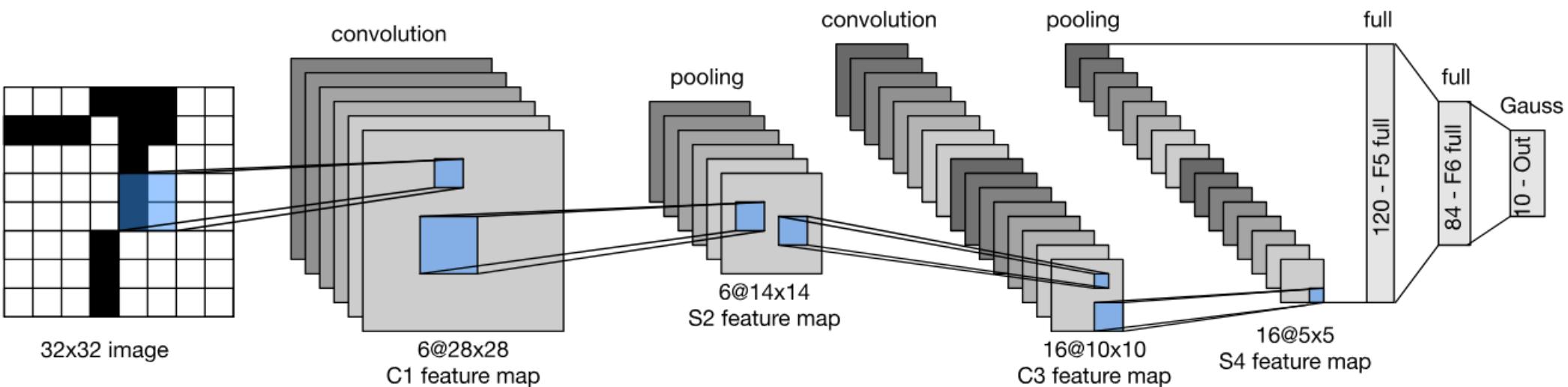
# **Introduction to Deep Learning**

## **Chapter 8 LeNet, AlexNet, VGG and NiN**

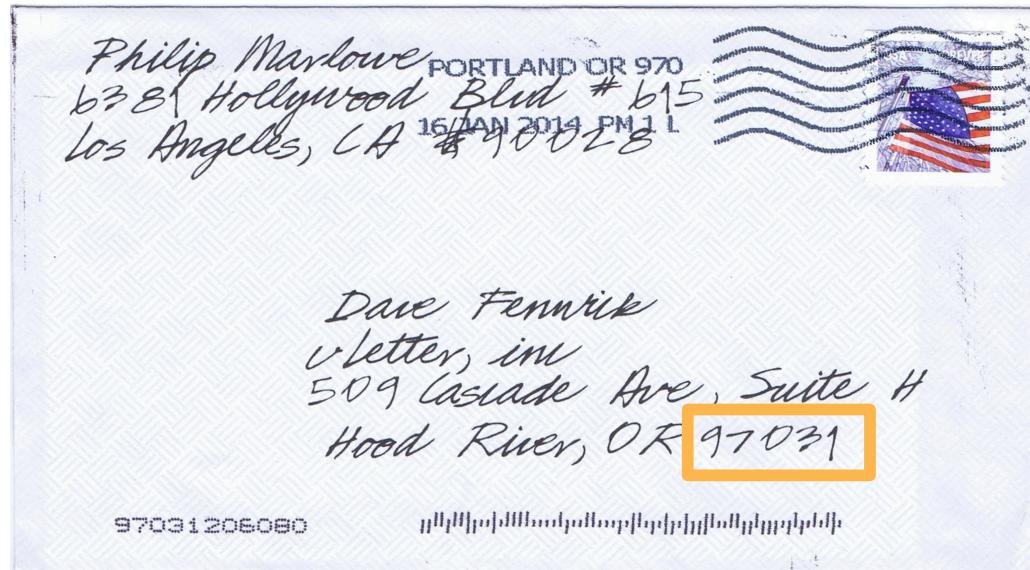
**Slides From Alex Smola and Mu Li**

**[courses.d2l.ai/berkeley-stat-157](https://courses.d2l.ai/berkeley-stat-157)**

# LeNet Architecture



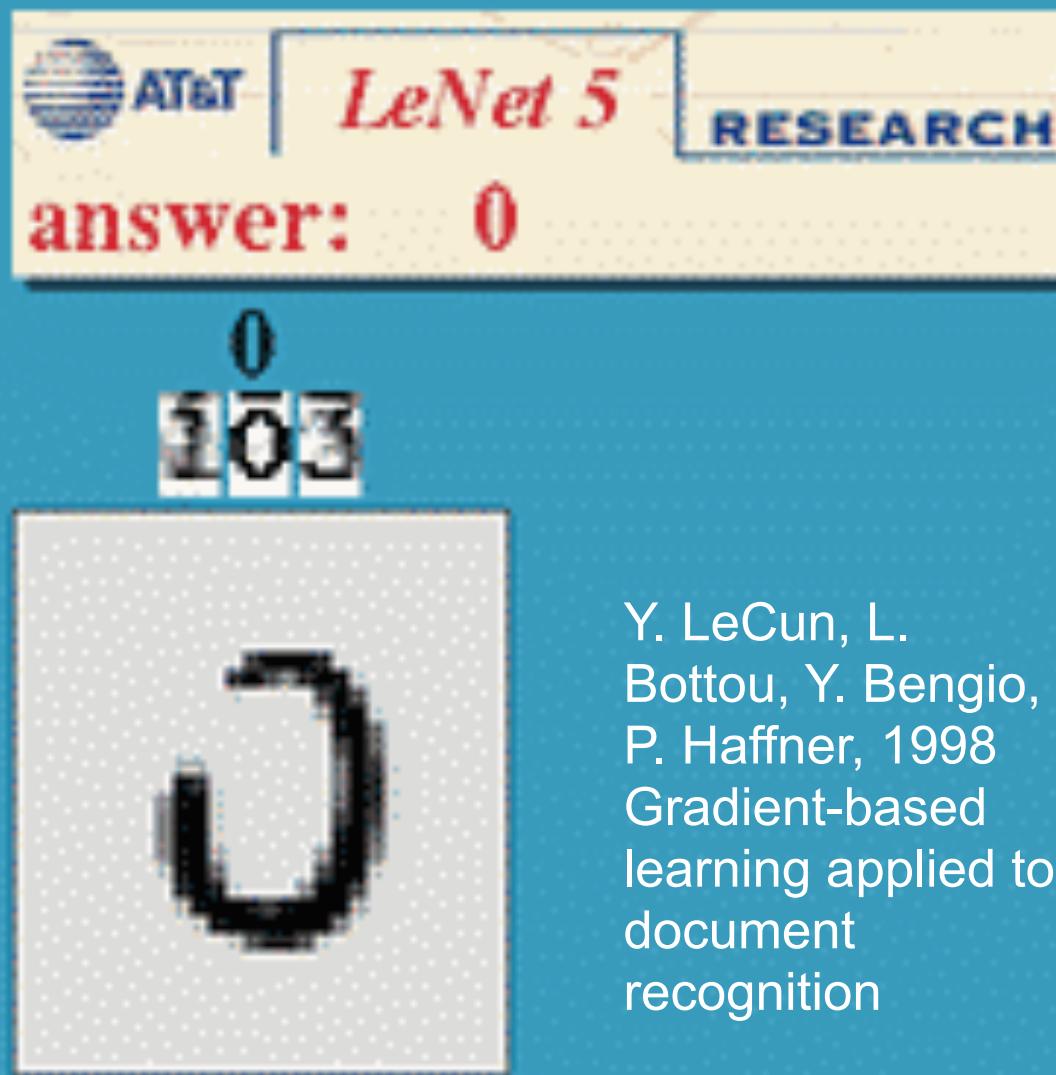
# Handwritten Digit Recognition



# MNIST

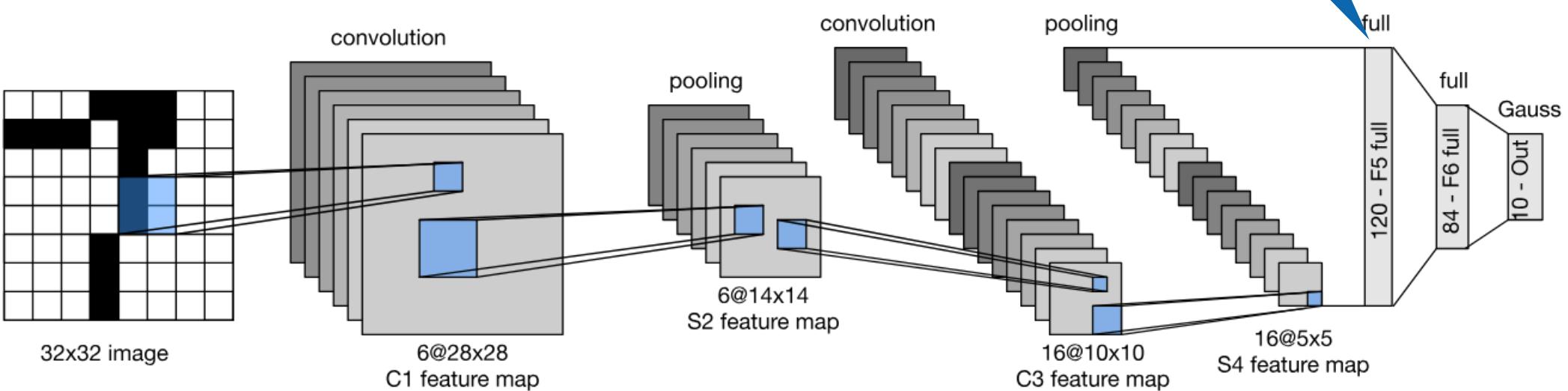
- Centered and scaled
- 60,000 training data
- 10,000 test data
- 28 x 28 images
- 10 classes





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

Expensive if we  
have many  
outputs



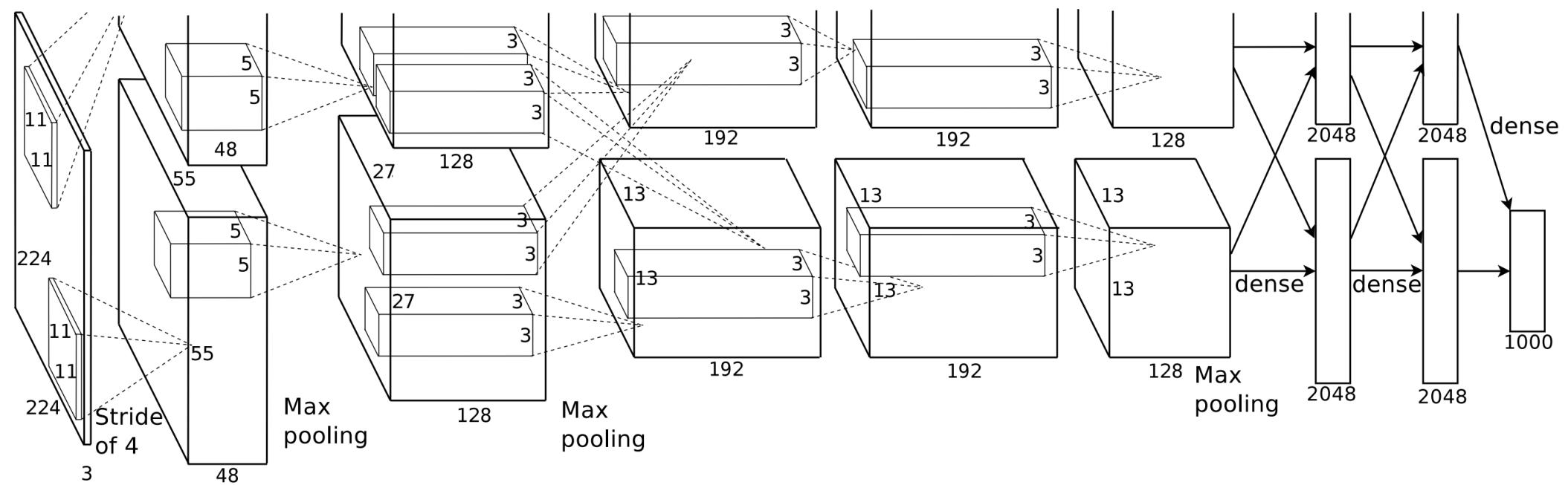
# LeNet in MXNet

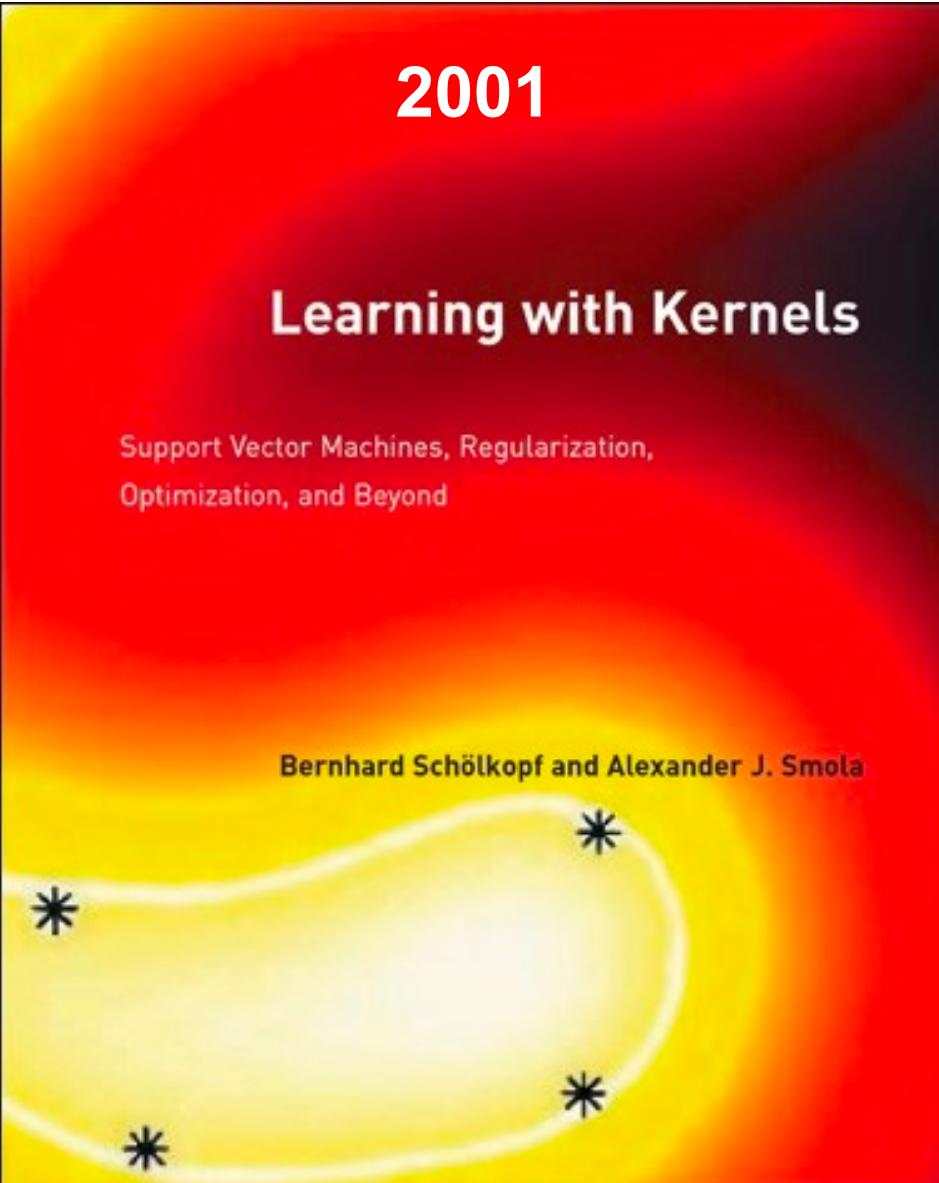
```
net = nn.Sequential()
net.add(nn.Conv2D(channels=6, kernel_size=5, activation='sigmoid'),
        nn.AvgPool2D(pool_size=2),
        nn.Conv2D(channels=16, kernel_size=5, activation='sigmoid'),
        nn.AvgPool2D(pool_size=2),
        nn.Dense(120, activation='sigmoid'),
        nn.Dense(84, activation='sigmoid'),
        nn.Dense(10))

loss = gluon.loss.SoftmaxCrossEntropyLoss()

(size and shape inference is automatic)
```

# AlexNet





2001

## Learning with Kernels

Support Vector Machines, Regularization,  
Optimization, and Beyond

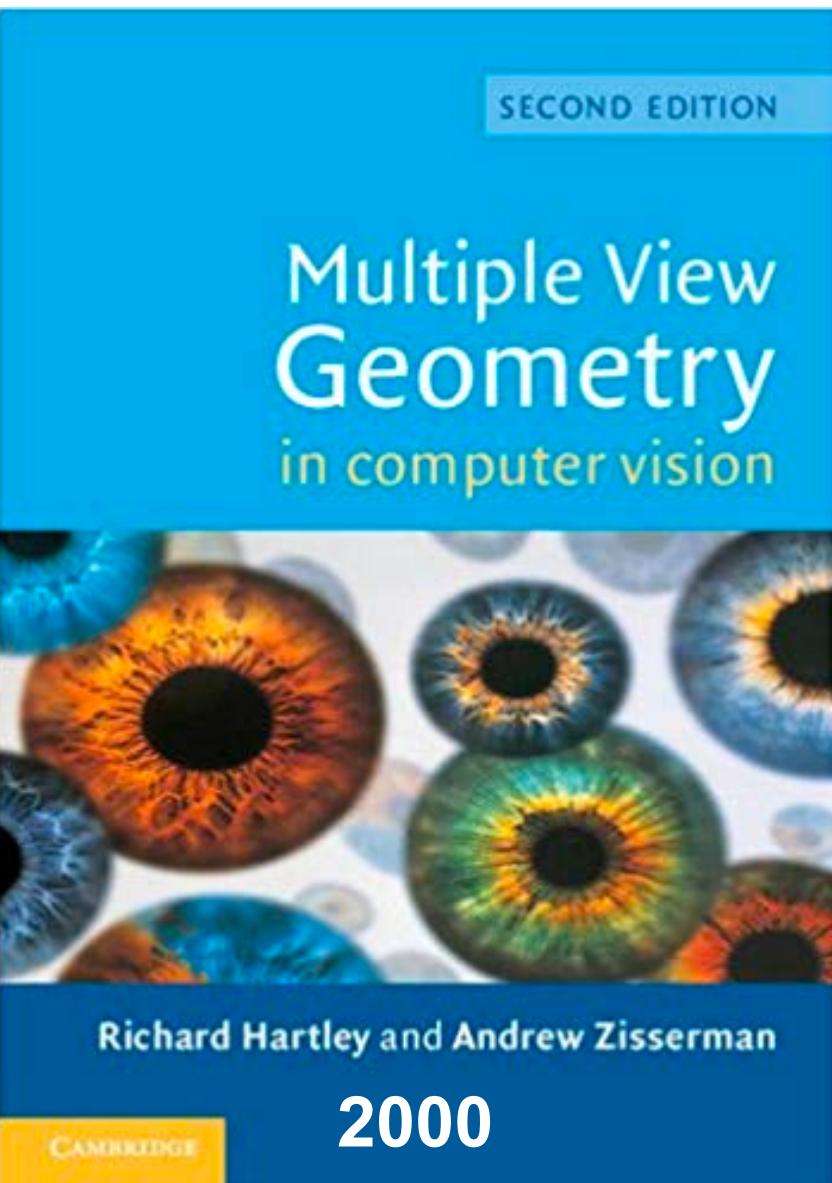
Bernhard Schölkopf and Alexander J. Smola



# Function Classes

In the 1990s, a new type of learning algorithm was developed, based on results from statistical learning theory: the Support Vector Machine (SVM). This gave rise to a new class of theoretically elegant learning machines that use a central concept of SVMs – kernels – for a number of

- Extract features
- Pick kernel for similarity
- **Convex** optimization problem
- Many beautiful theorems ...



# Geometry

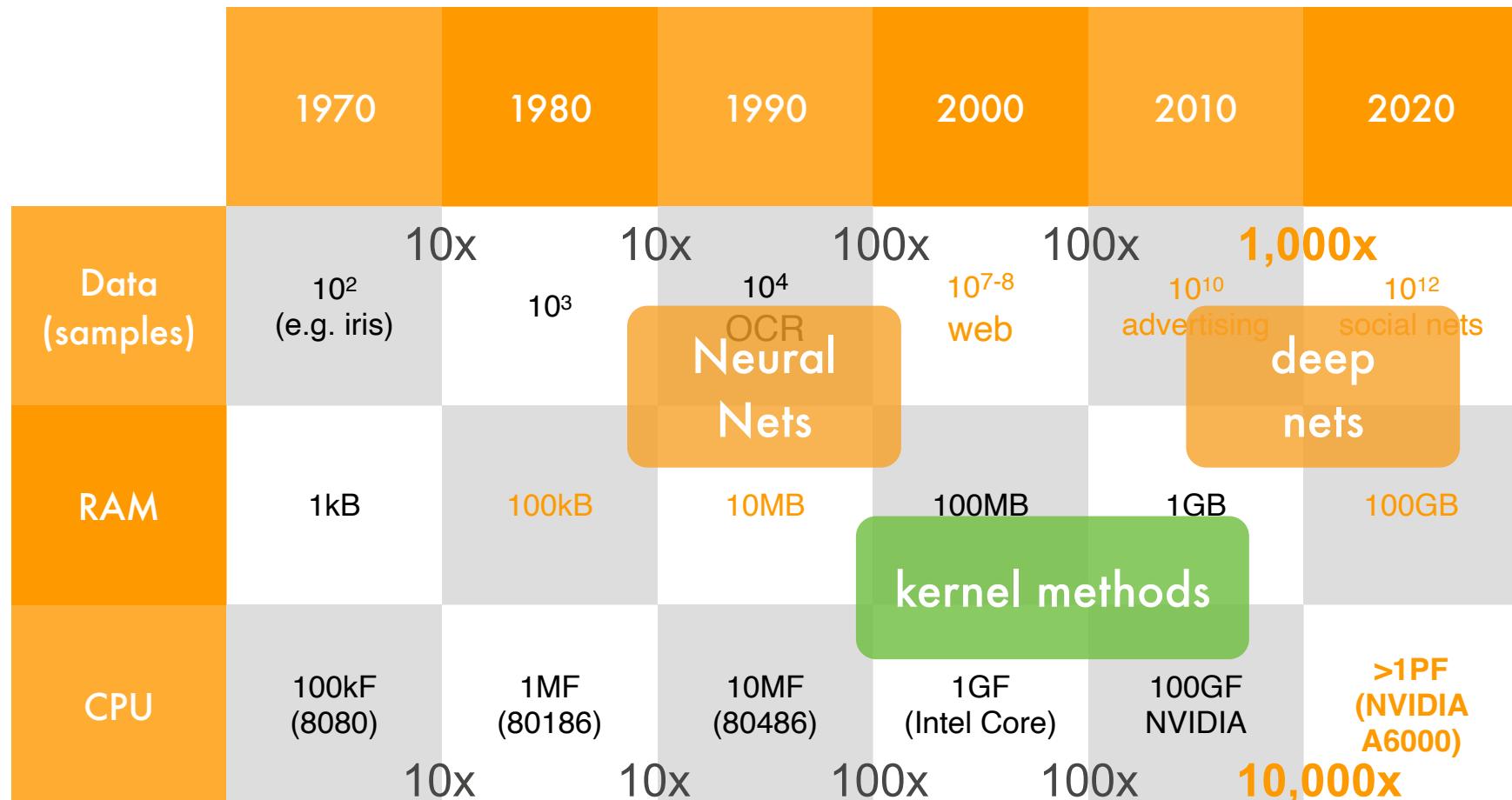
- Kernel ideas are used to solve PDE-heat equation ( $\frac{\partial u(x,t)}{\partial t} = -\Delta_M u(x,t) = -L_{xxx} u(x,t)$ )
- Extract features  $e^{-\lambda t}$
- Describe geometry (e.g. multiple cameras) analytically
- **(Non)Convex** optimization problems
- Many beautiful theorems ...
- Works very well in theory when the assumptions are satisfied

# Feature Engineering



- Feature engineering is crucial
- Feature descriptors, e.g. SIFT (Scale-invariant feature transform), SURF
- Bag of visual words (clustering)
- Then apply SVM ...

# Hardware



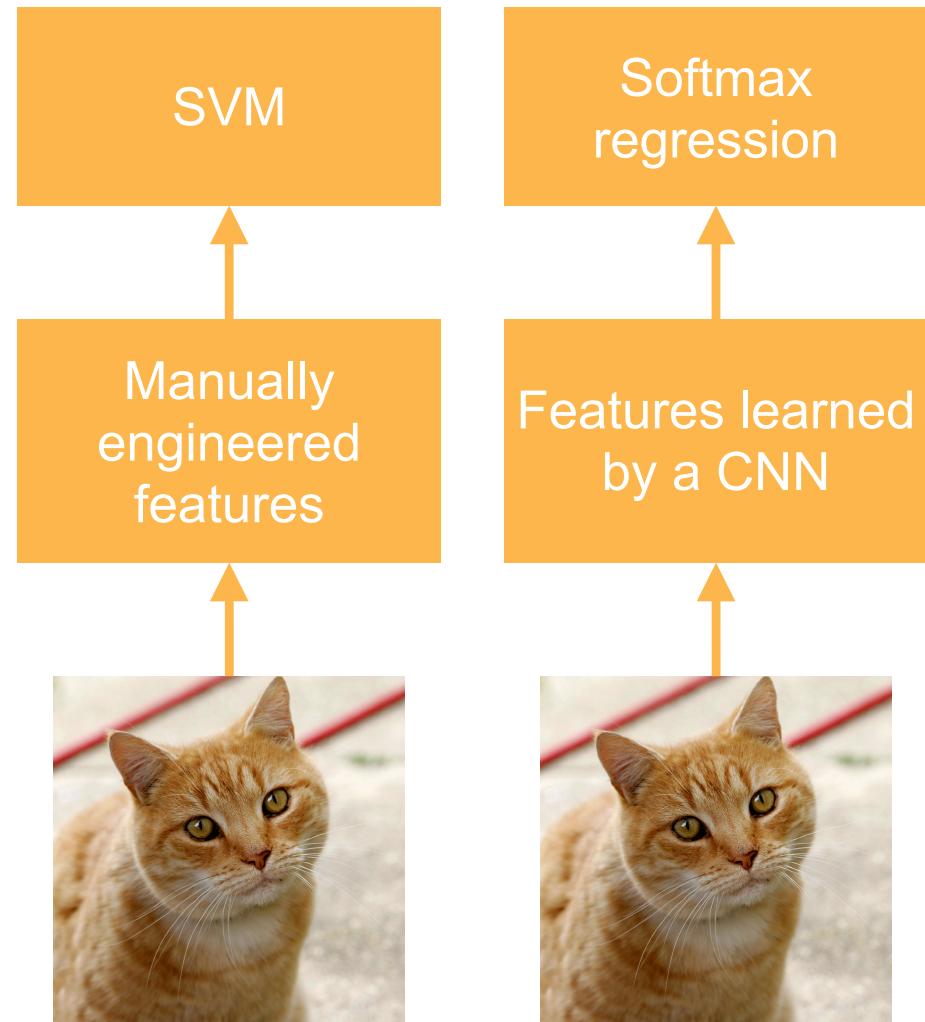
# ImageNet (2010)



<b>Images</b>	Color images with nature objects	Gray image for handwritten digits
<b>Size</b>	469 x 387	28 x 28
<b># examples</b>	1.2 M	60 K
<b># classes</b>	1,000	10

# AlexNet

- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Key modifications
  - Dropout (regularization)
  - ReLu (training)
  - MaxPooling
- Paradigm shift for computer vision

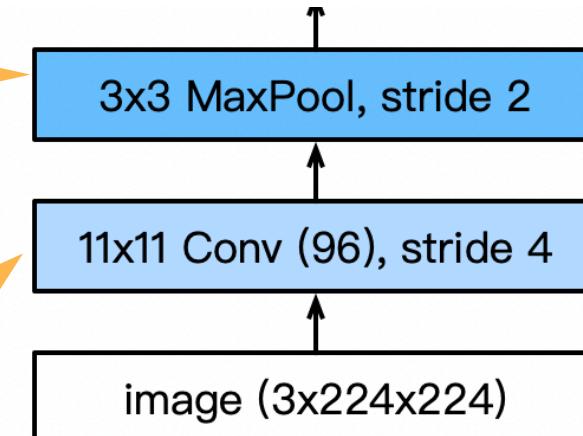


# AlexNet Architecture

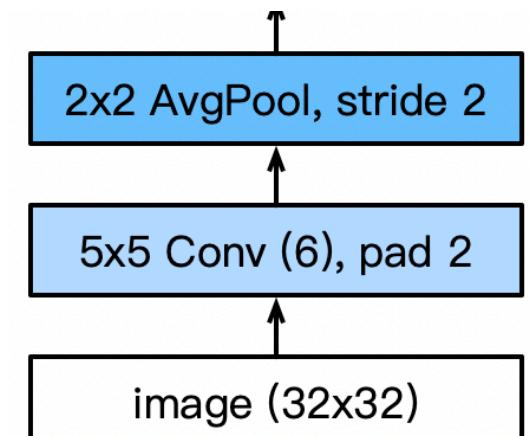
Larger pool size, change to max pooling

Larger kernel size, stride because of the increased image size, and more output channels.

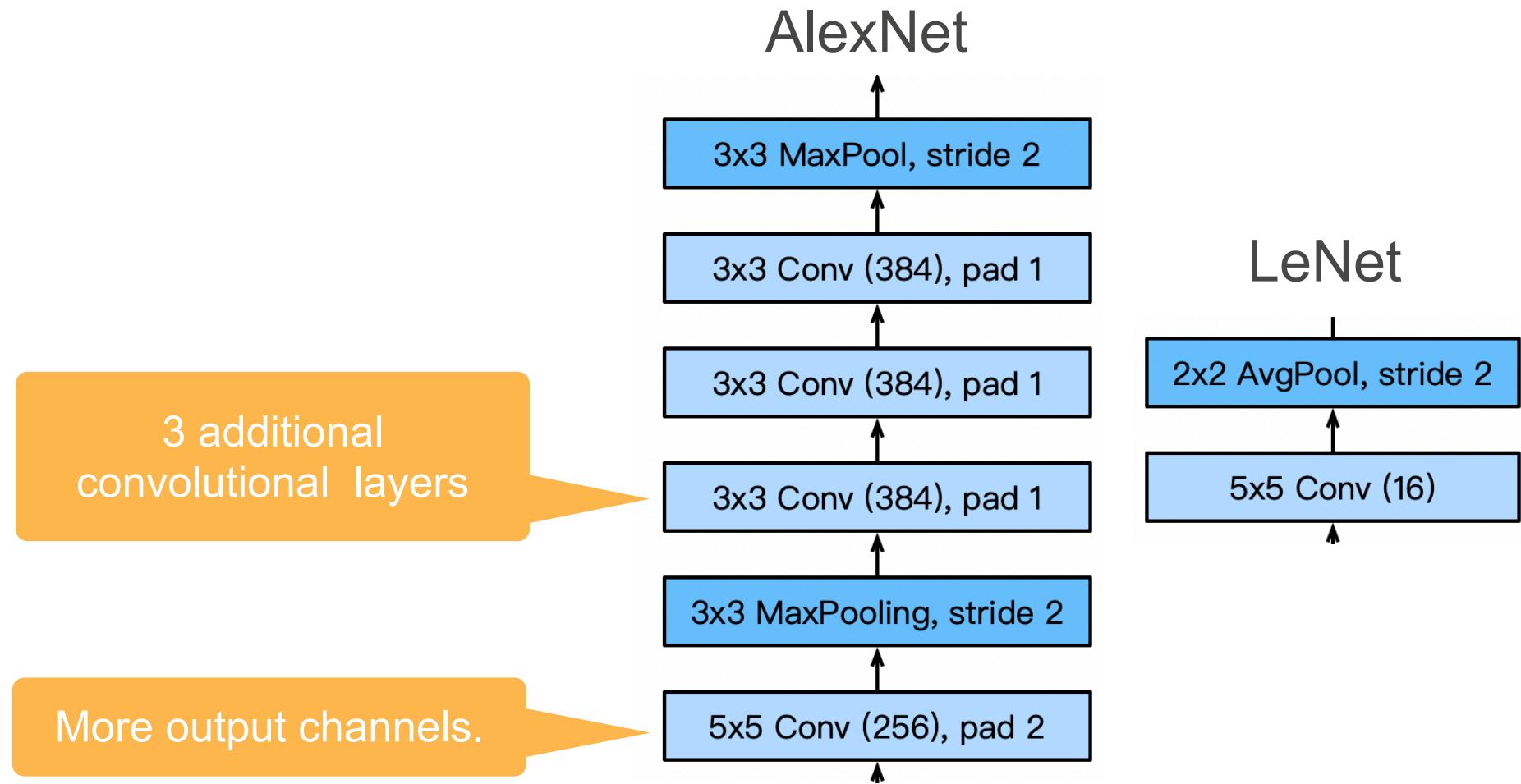
AlexNet



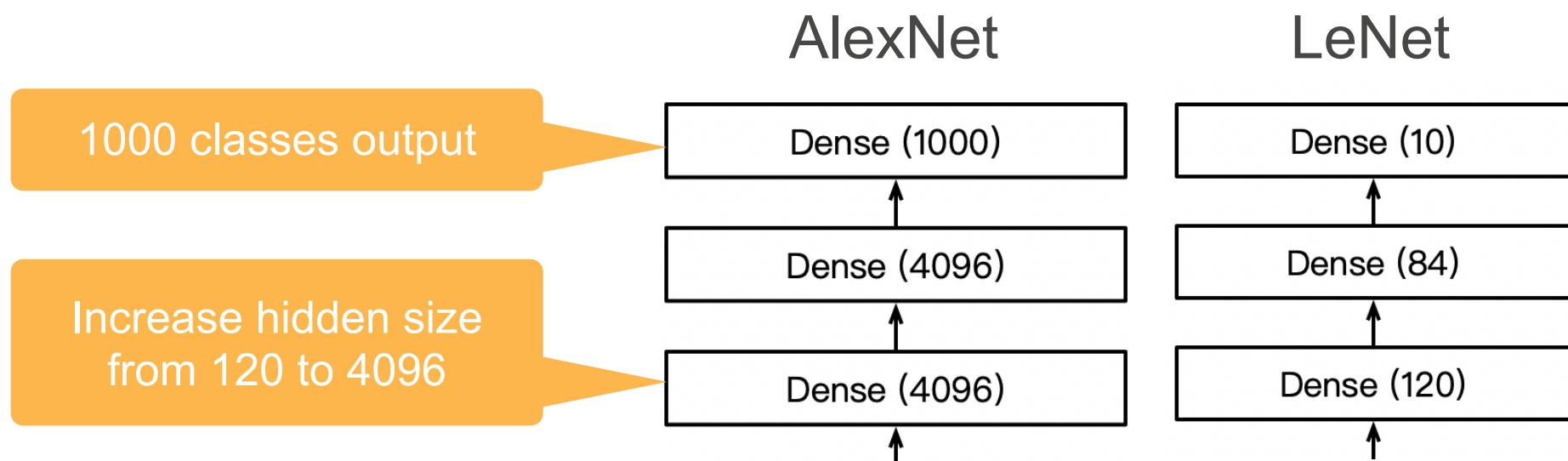
LeNet



# AlexNet Architecture

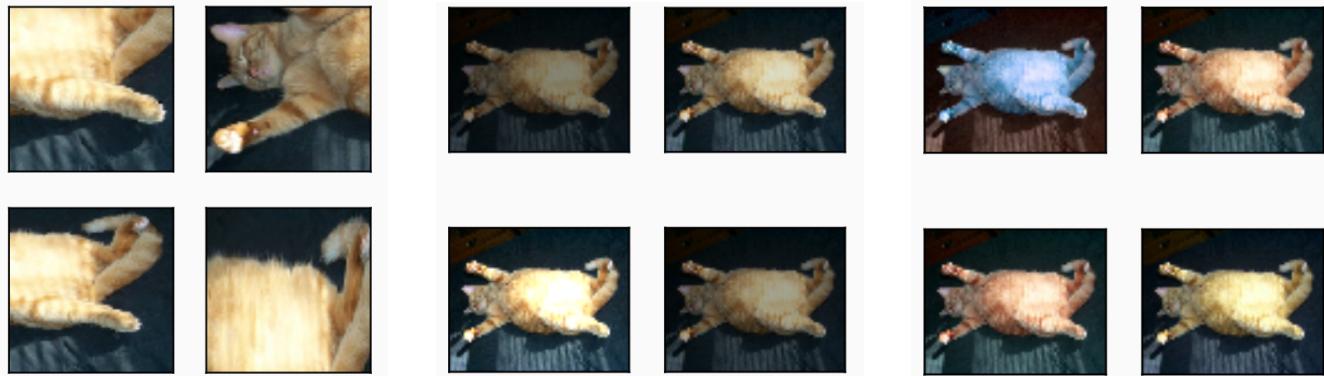
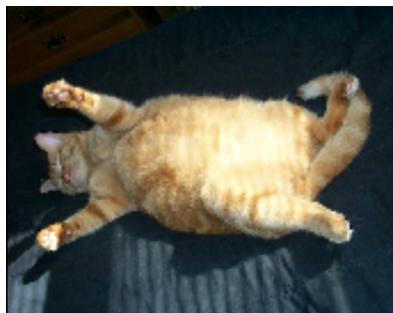


# AlexNet Architecture



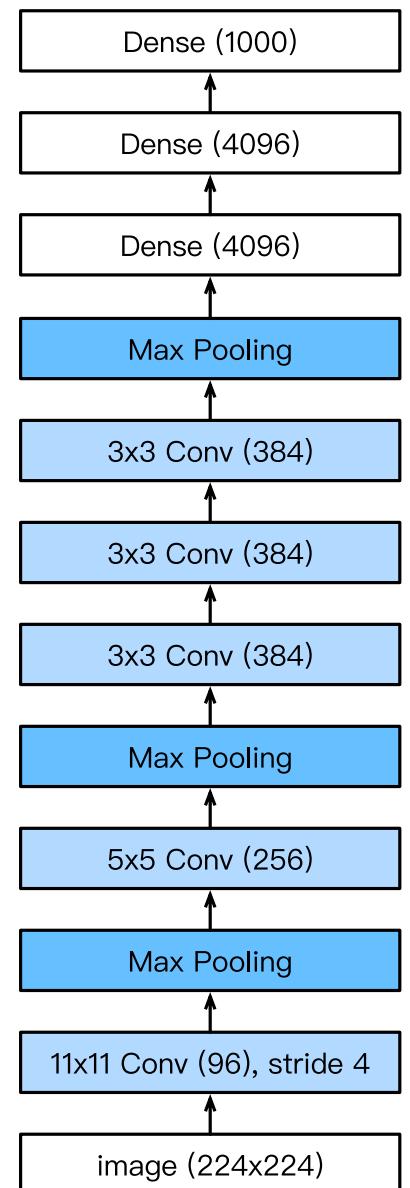
# More Tricks

- Change activation function from sigmoid to ReLu  
(no more vanishing gradient)
- Add a dropout layer after two hidden dense layers  
(better robustness / regularization)
- Data augmentation

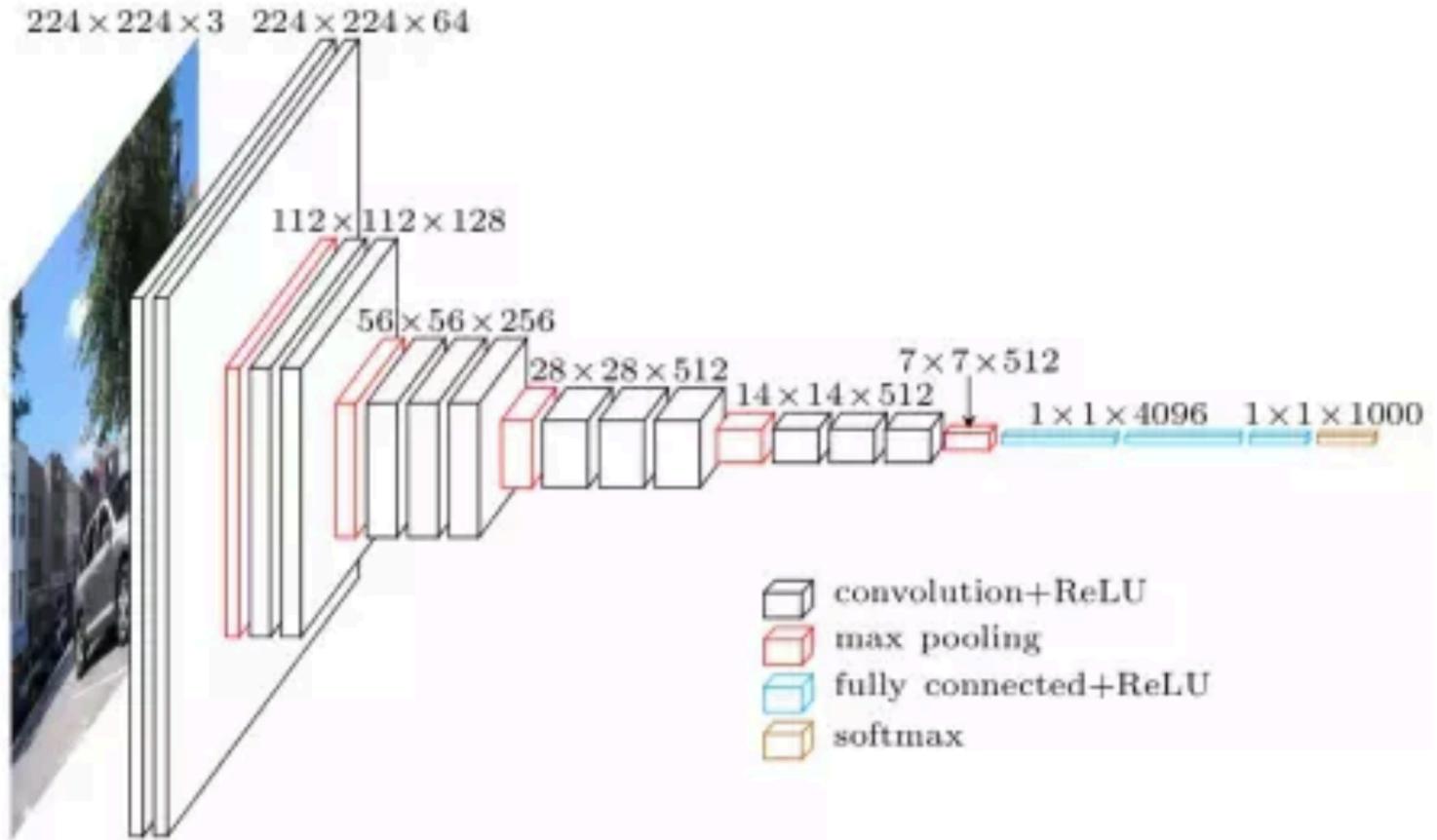


# Complexity

	#parameters		FLOP	
	AlexNet	LeNet	AlexNet	LeNet
<b>Conv1</b>	35K	150	101M	1.2M
<b>Conv2</b>	614K	2.4K	415M	2.4M
<b>Conv3-5</b>	3M		445M	
<b>Dense1</b>	26M	0.48M	26M	0.48M
<b>Dense2</b>	16M	0.1M	16M	0.1M
<b>Total</b>	46M	0.6M	1G	4M
<b>Increase</b>	11x	1x	250x	1x



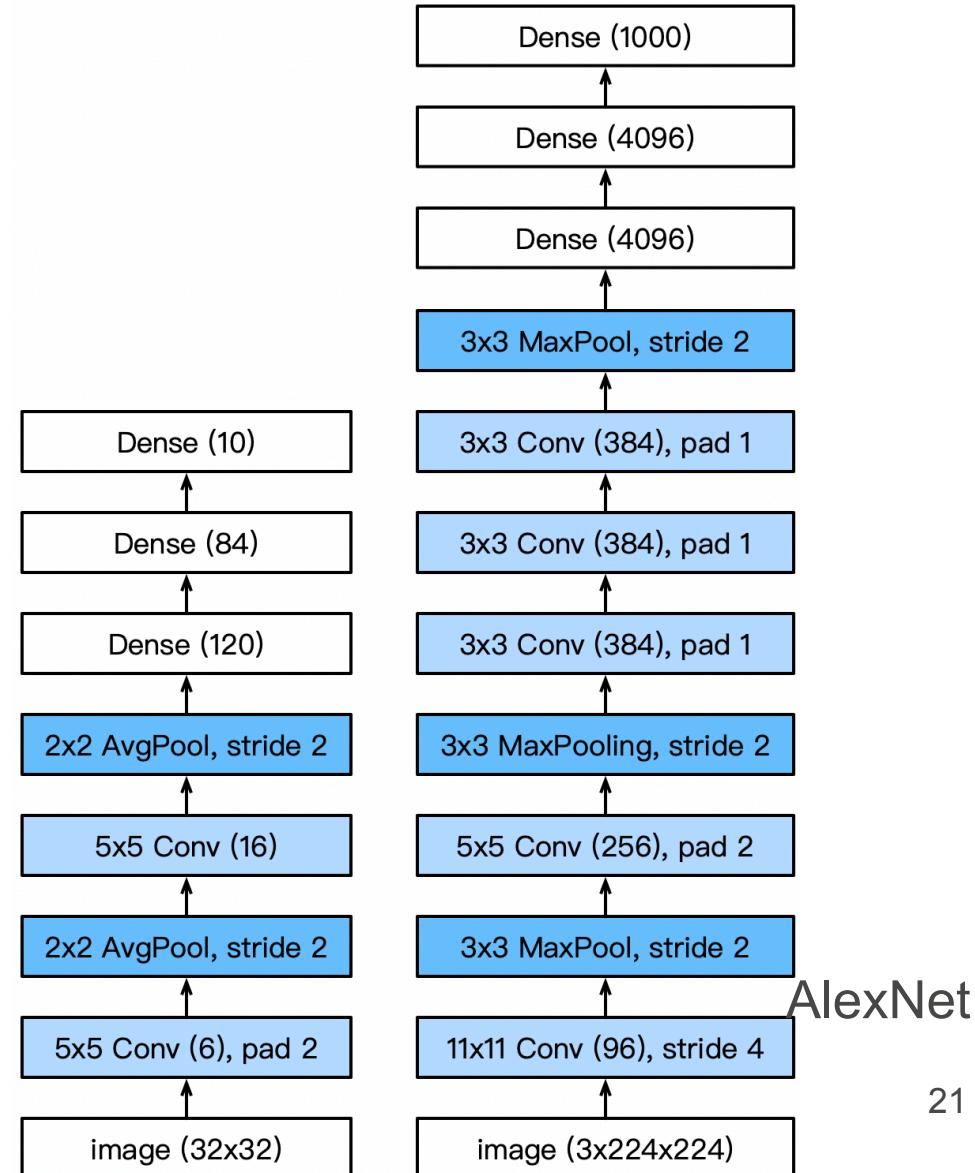
# VGG



Visual Geometry Group

# VGG

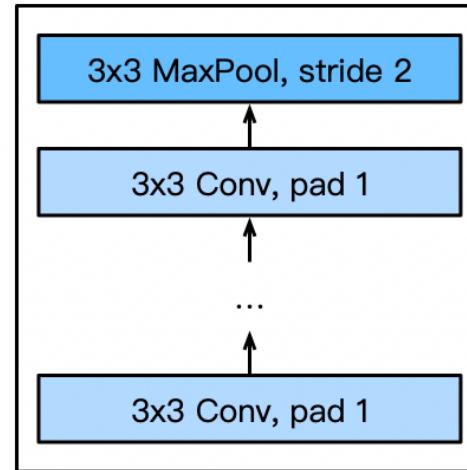
- AlexNet is deeper and bigger than LeNet to get performance
- Go even bigger & deeper?
- Options
  - More dense layers (too expensive)
  - **More convolutions**
  - Group into **blocks**



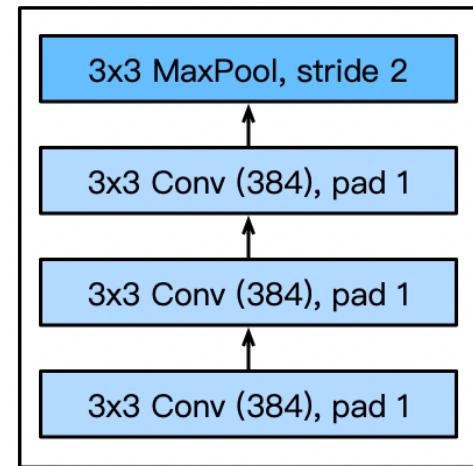
# VGG Blocks

- Deeper vs. wider?
  - 5x5 convolutions
  - 3x3 convolutions (more)
  - **Deep & narrow better**
- VGG block
  - 3x3 convolutions (pad 1)  
**(n layers, m channels)**
  - 2x2 max-pooling  
(stride 2)

VGG block

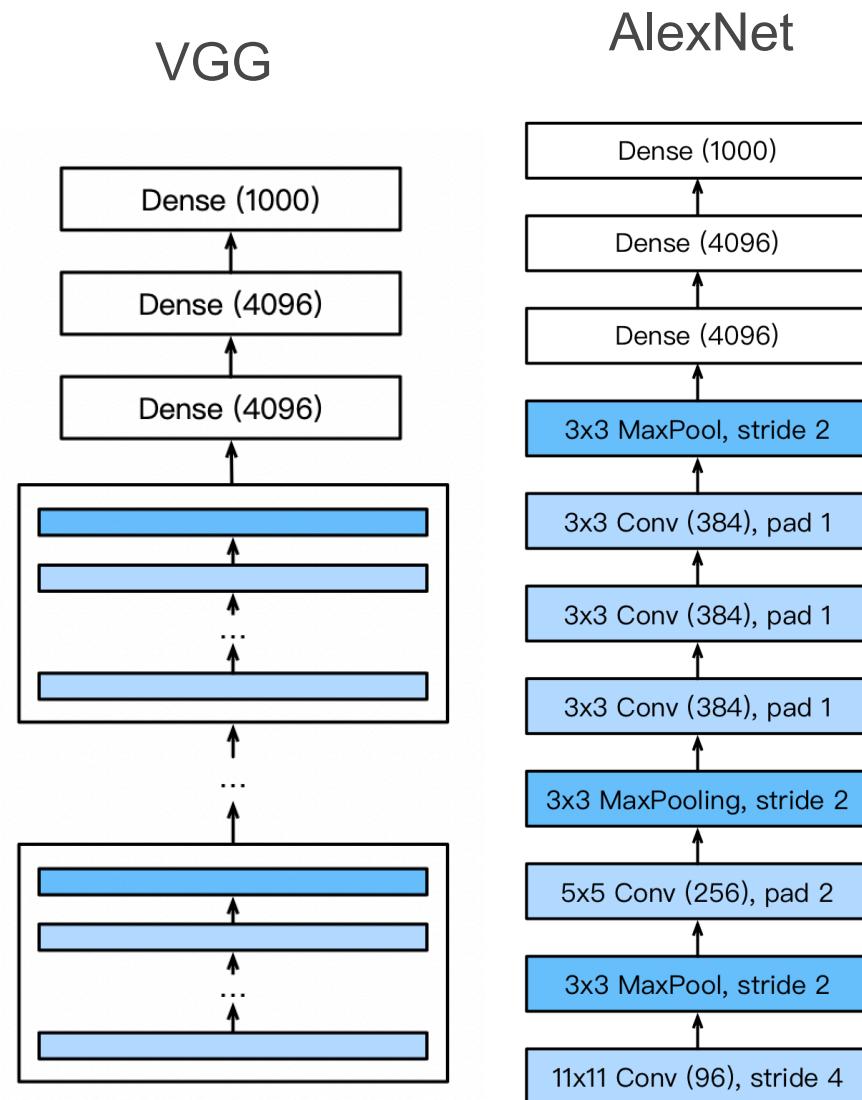


Part of AlexNet



# VGG Architecture

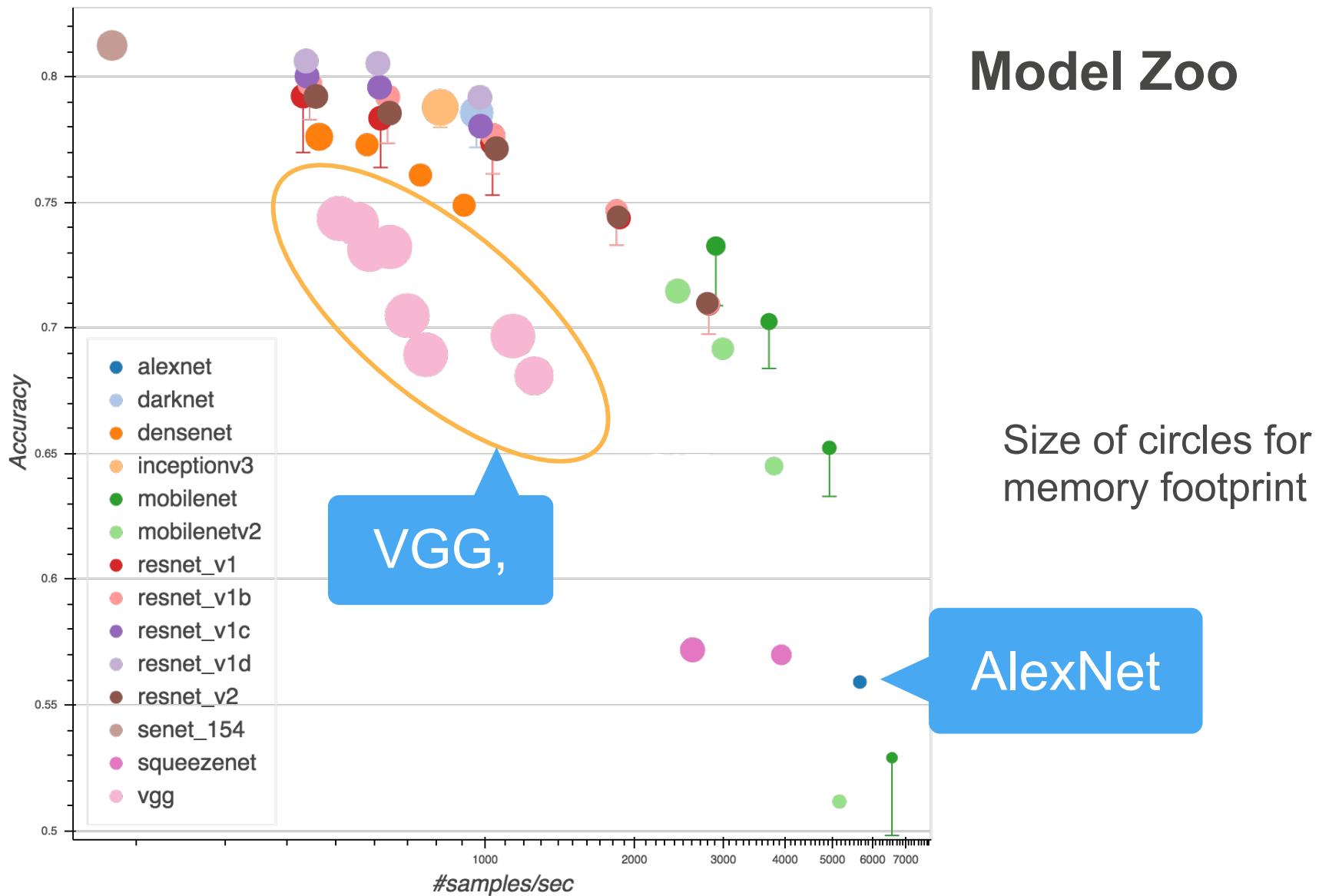
- Multiple VGG blocks followed by dense layers
- Vary the repeating number to get different architectures, such as VGG-16, VGG-19, ...



# Progress

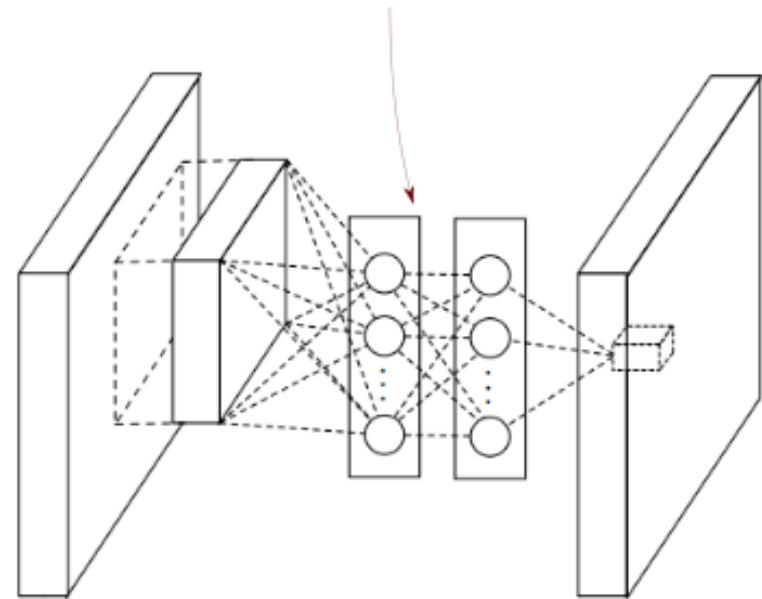
- LeNet (1995)
  - 2 convolution + pooling layers
  - 2 hidden dense layers
- AlexNet
  - Bigger and deeper LeNet
  - ReLu, Dropout, preprocessing
- VGG
  - Bigger and deeper AlexNet (repeated VGG blocks)

# Model Zoo

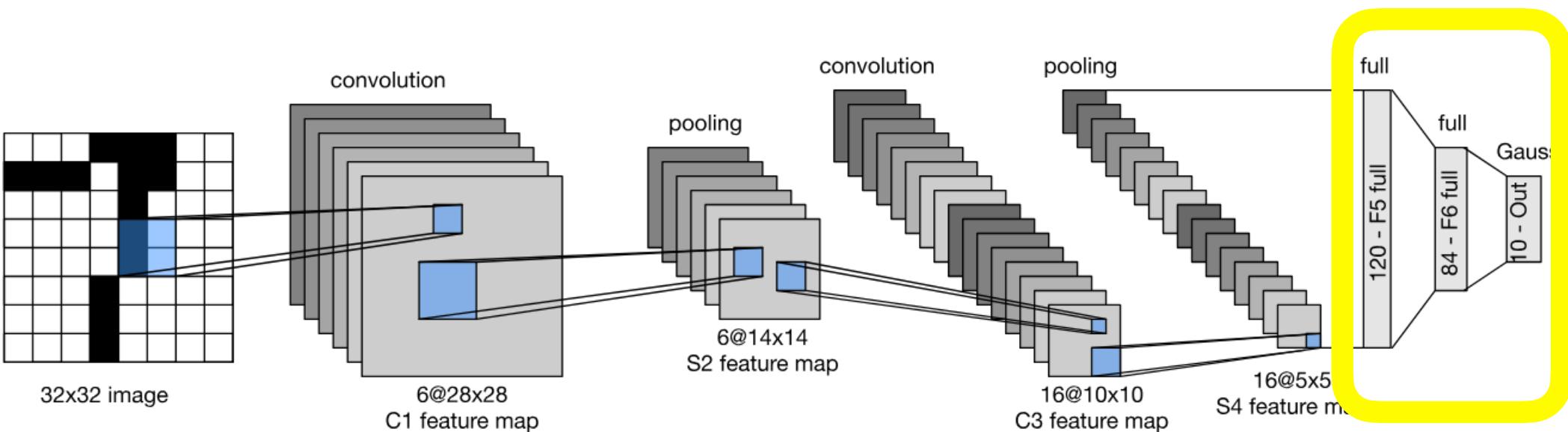


# Network in Network

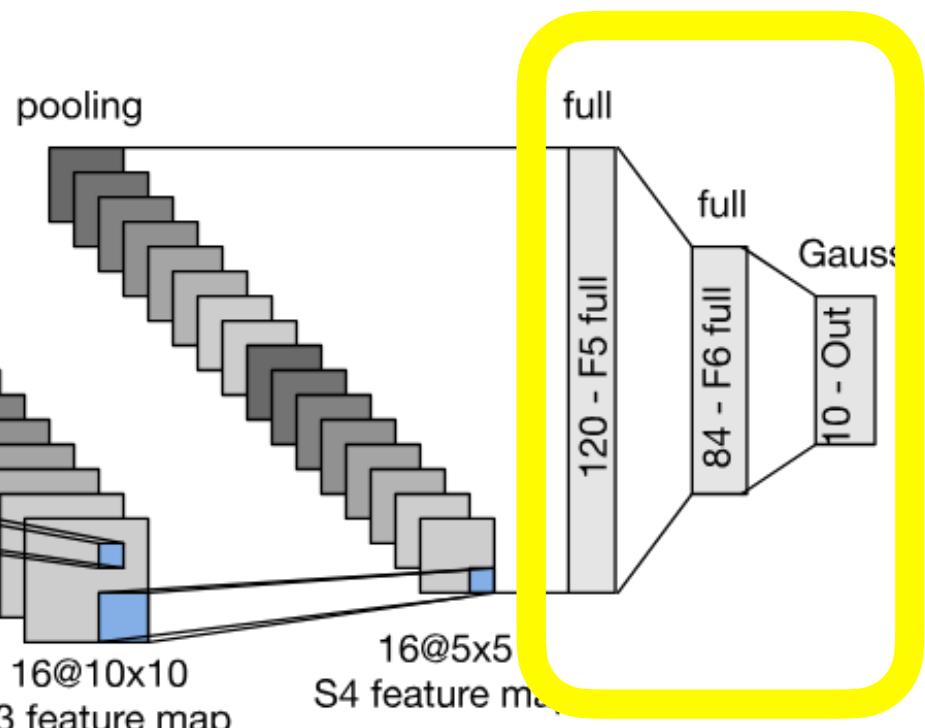
Non linear mapping introduced by mlpconv layer consisting of multiple fully connected layers with non linear activation function.



# The Curse of the Last Layer(s)



# The Last Layer(s). They are too large.



- Convolution layers need relatively few parameters

$$c_i \times c_o \times k^2$$

- Last layer needs many parameters for n classes

$$c \times m_w \times m_h \times n$$

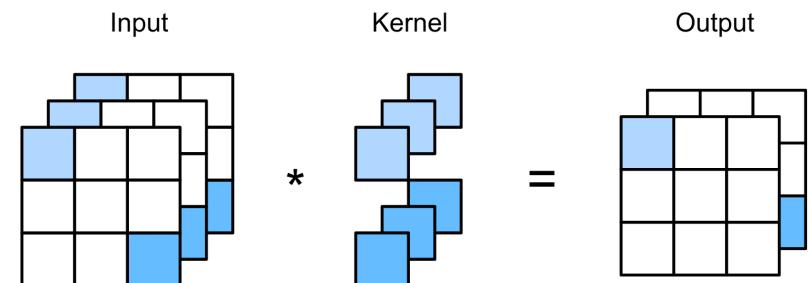
- LeNet  $16 \times 5 \times 5 \times 120 = 48k$
- AlexNet  $256 \times 5 \times 5 \times 4096 = 26M$
- VGG  $512 \times 7 \times 7 \times 4096 = 102M$

# VGG parameters

sequential1 output shape:	(1, 64, 112, 112)
sequential2 output shape:	(1, 128, 56, 56)
sequential3 output shape:	(1, 256, 28, 28)
sequential4 output shape:	(1, 512, 14, 14)
<b>sequential5 output shape:</b>	<b>(1, 512, 7, 7)</b>
<b>dense0 output shape:</b>	<b>(1, 4096)</b>
dropout0 output shape:	(1, 4096)
dense1 output shape:	(1, 4096)
dropout1 output shape:	(1, 4096)
dense2 output shape:	(1, 10)

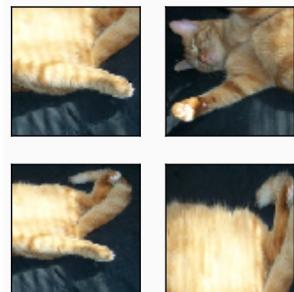
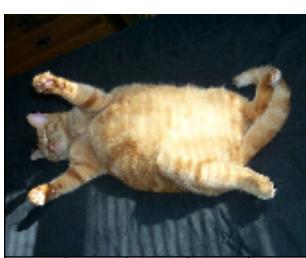
# Breaking the Curse of the Last Layer

- Key Idea
  - **Get rid of the fully connected last layer(s)**
  - Convolutions and pooling reduce resolution (e.g. stride of 2 reduces resolution 4x)
- Implementation details (MLP on pixel level)
  - Reduce resolution progressively
  - Increase number of channels
  - **Use  $1 \times 1$  convolutions**
    - (they only act per pixel)
- **Global average pooling in the end**

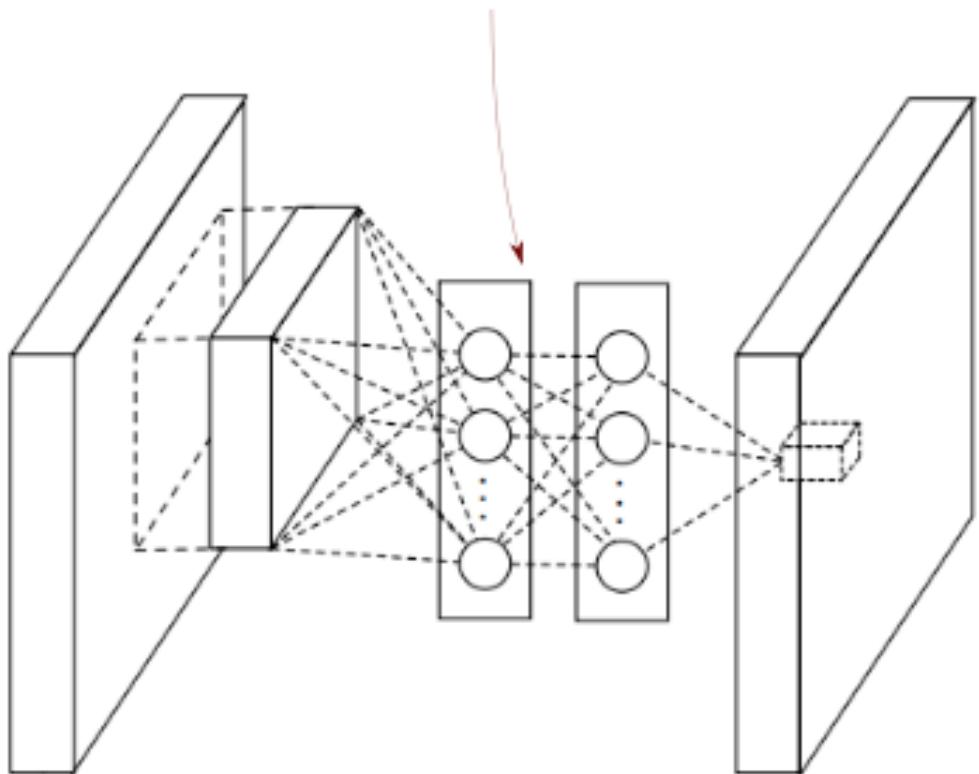


# What's a 1x1 convolution anyway?

- Extreme case  
1x1 image with n channels
- Equivalent to MLP
- Pooling allows for  
translation invariance of  
detection (e.g. 5x5)

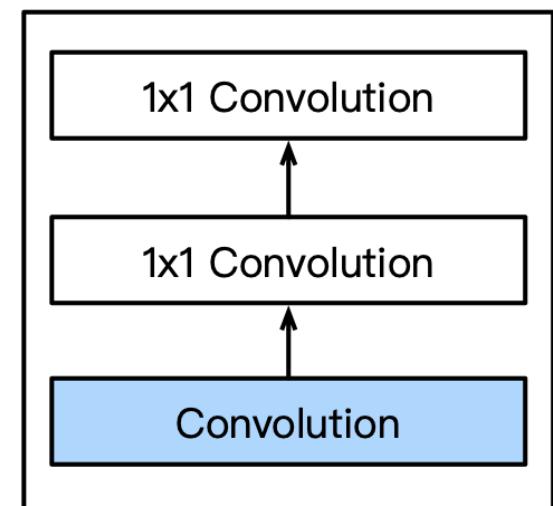


Non linear mapping introduced by mlpconv layer consisting of multiple fully connected layers with non linear activation function.

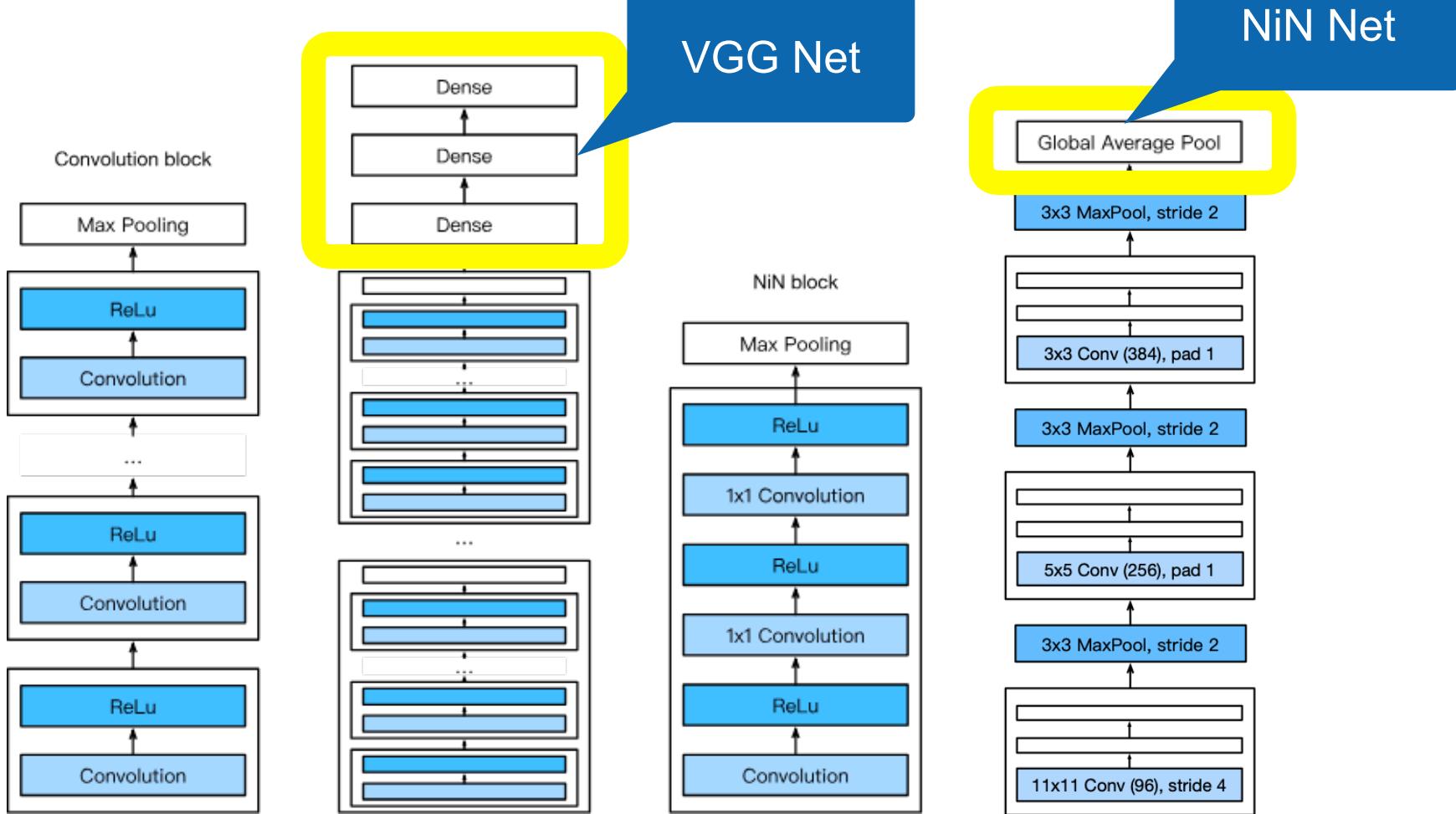


# NiN Block

- A convolutional layer
  - kernel size, stride, and padding are hyper-parameters
- Following by two 1x1 convolutions
  - 1 stride and no padding, share the same output channels as first layer
  - Act as dense layers

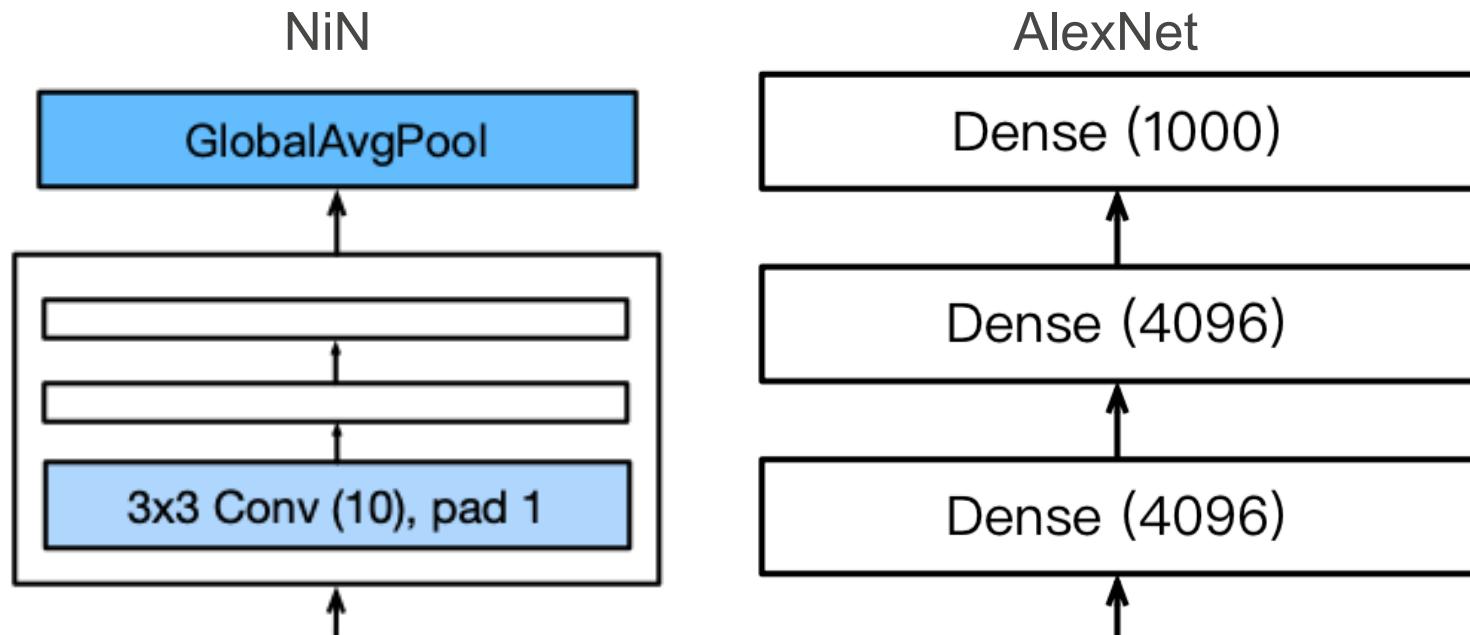


# NiN Networks



# NiN Last Layers

- Replaced AlexNet's dense layers with a NiN block
- Global average pooling layer to combine outputs



# Summary

- Reduce image resolution progressively
- Increase number of channels
- Global average pooling for given number of classes

```
sequential1 output shape: (96, 54, 54)
pool0 output shape: (96, 26, 26)
sequential2 output shape: (256, 26, 26)
pool1 output shape: (256, 12, 12)
sequential3 output shape: (384, 12, 12)
pool2 output shape: (384, 5, 5)
dropout0 output shape: (384, 5, 5)
sequential4 output shape: (10, 5, 5)
pool3 output shape: (10, 1, 1)
flatten0 output shape: (10)
```

NIN dimension reduction

# Summary

- **LeNet** (the first convolutional neural network)
- **AlexNet**
  - More of everything
  - ReLu, Dropout, Invariances
- **VGG**
  - Even more of everything (narrower and deeper)
  - Repeated blocks
- **NiN**
  - 1x1 convolutions + global pooling instead of dense