



Machine Learning

Lecture 13: Convolutional Neural Networks

Fall 2023

Instructor: Xiaodong Gu



Vision in the Wild



**“To know what is
where by looking.”**

The Rise and Impact of Computer Vision



Robotics



Autonomous driving



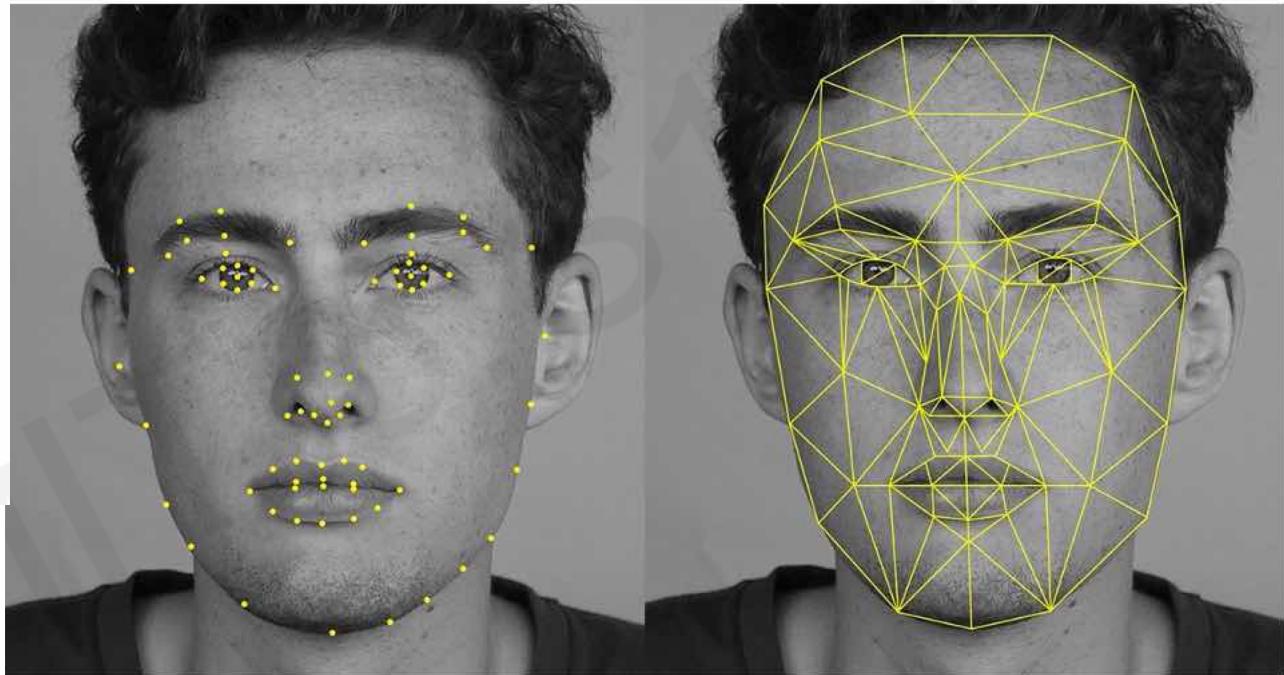
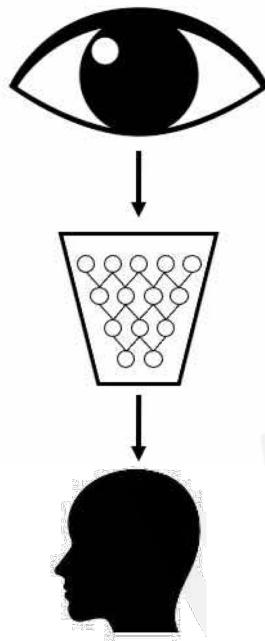
Military



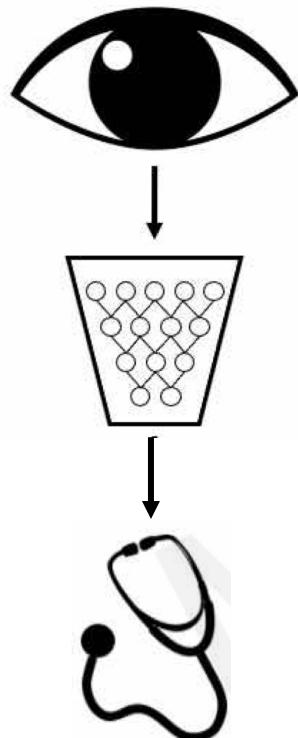
Biology & Medicine



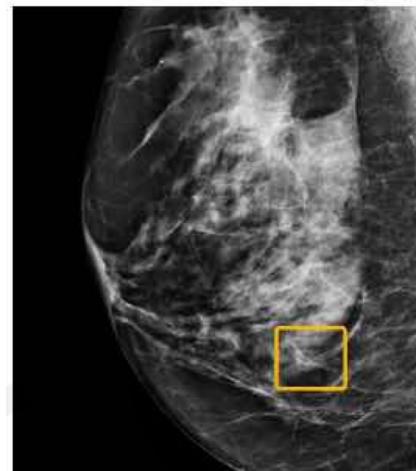
Impact: Facial Detection & Recognition



Impact: Medicine, Biology, Healthcare



Breast cancer



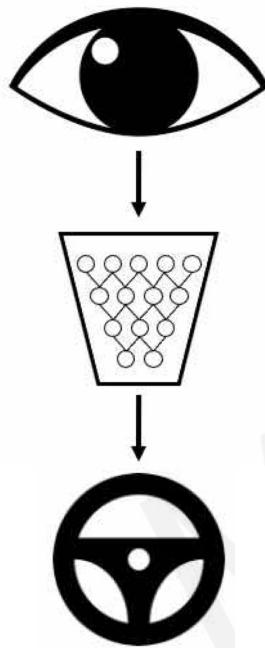
COVID-19



Skin cancer



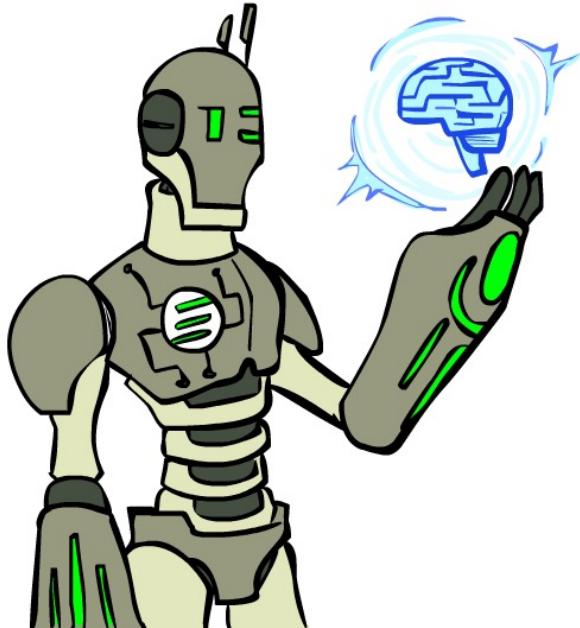
Impact: Self-Driving Cars



Today



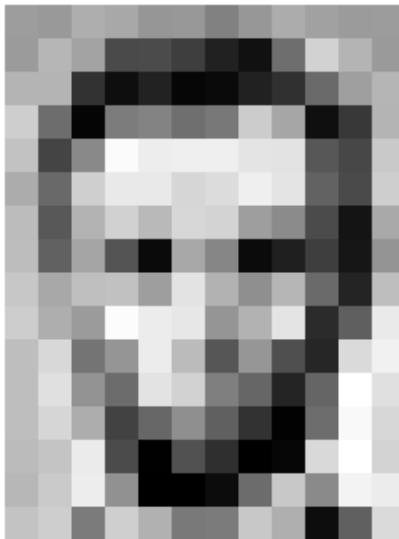
- Feedforward neural networks
- **Convolutional neural networks**
- Recurrent neural networks
- ...



What Computers See?



- Images are numbers



| | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 105 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 29 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 105 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 85 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 95 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 236 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 196 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

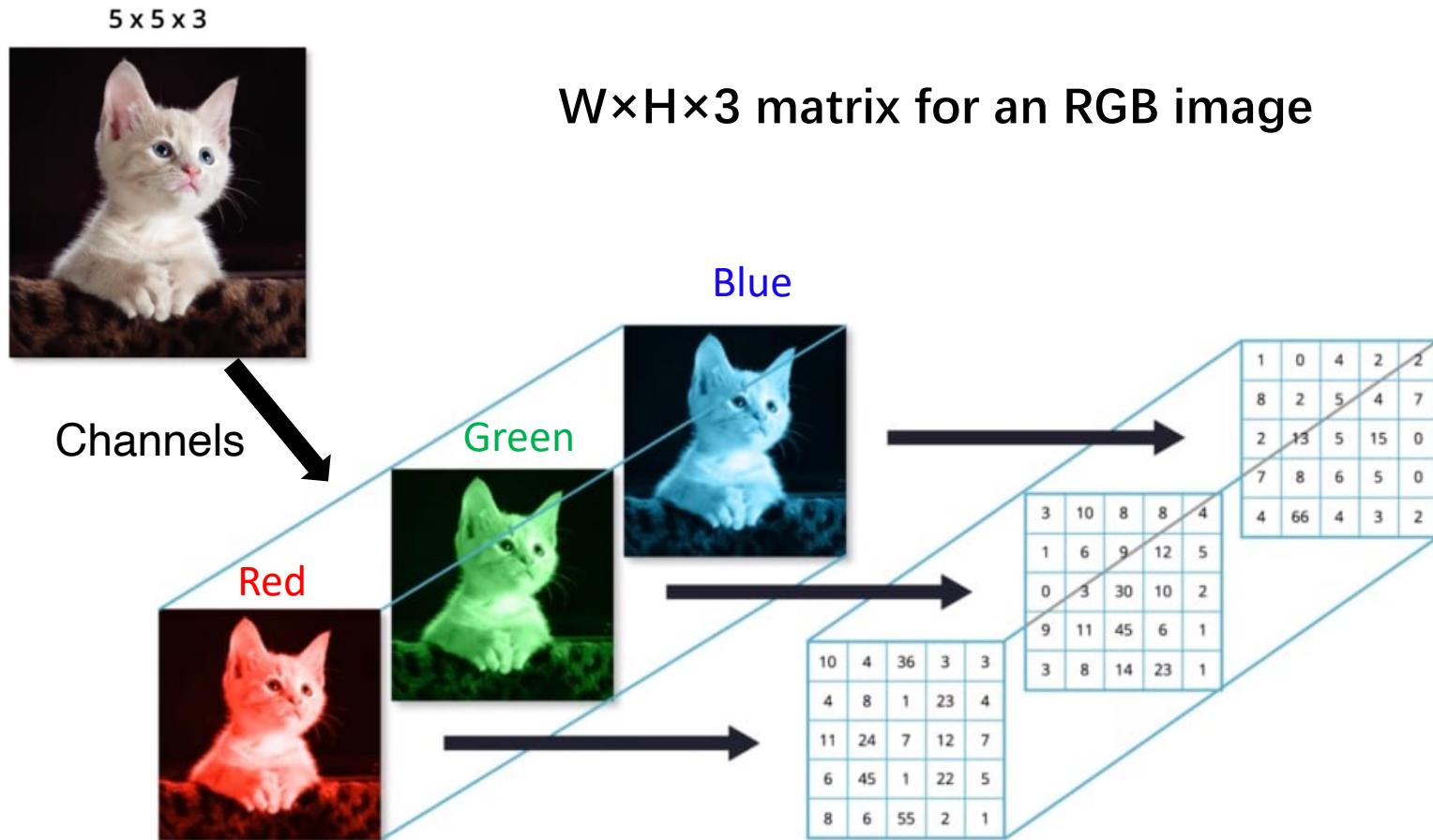
| | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 29 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 85 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 95 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 236 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 196 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

An image is just a matrix of numbers [0,255]

What Computers See?



- Images are numbers



How can computer **see** **understand** images?



- Idea #1: Pixels as features?



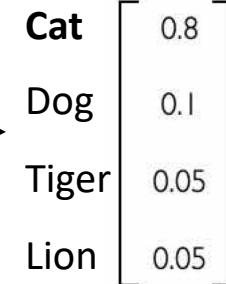
Input Image

| | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 156 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 259 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 216 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |

Pixel Representation

$$g(x) = \mathbf{w}^T x + w_0$$

Model



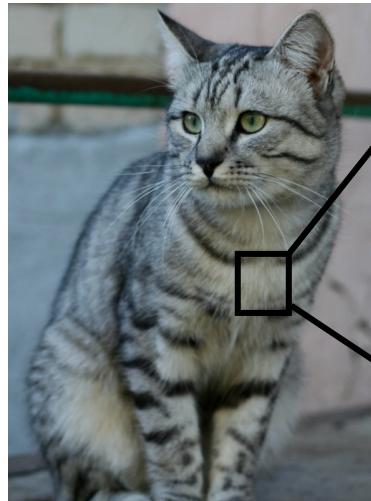
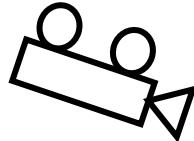
Classification



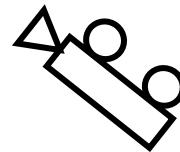
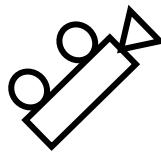
Challenges of Image Modeling



- Viewpoint Variation



| |
|--|
| [105 112 108 111 104 99 106 99 96 103 112 119 104 97 93 87] |
| [91 98 102 106 104 79 98 103 99 105 123 136 110 105 94 85] |
| [76 85 90 105 128 105 87 96 95 99 115 112 106 103 99 85] |
| [99 81 81 93 120 131 127 100 95 98 102 99 96 93 101 94] |
| [106 91 61 64 69 91 88 85 101 107 109 98 75 84 96 95] |
| [114 108 85 55 55 69 64 54 64 87 112 129 98 74 84 91] |
| [133 137 147 103 65 81 80 65 52 54 74 84 102 93 85 82] |
| [128 137 144 140 109 95 86 70 62 65 63 63 60 73 86 101] |
| [125 133 148 137 119 121 117 94 65 79 80 65 54 64 72 98] |
| [127 125 131 147 133 127 126 131 111 96 89 75 61 64 72 84] |
| [115 114 108 123 150 148 131 118 113 109 100 92 74 65 72 78] |
| [89 93 90 97 108 147 131 118 113 114 113 109 106 95 77 80] |
| [63 77 86 81 77 79 102 123 117 115 117 125 125 130 115 87] |
| [62 65 82 89 78 71 80 101 124 126 119 101 107 114 131 119] |
| [63 65 75 88 89 71 62 81 120 138 135 105 81 98 110 118] |
| [87 65 71 87 106 95 69 45 76 130 126 107 92 94 105 112] |
| [118 97 82 86 117 123 116 66 41 51 95 93 89 95 102 107] |
| [164 146 112 80 82 120 124 104 76 48 45 66 88 101 102 109] |
| [157 170 157 120 93 86 114 132 112 97 69 55 70 82 99 94] |
| [130 128 134 161 139 100 109 118 121 134 114 87 65 53 69 86] |
| [128 112 96 117 150 144 120 115 104 107 102 93 87 81 72 79] |
| [123 107 96 86 83 112 153 149 122 109 104 75 80 107 112 99] |
| [122 121 102 80 82 86 94 117 145 148 153 102 58 78 92 107] |
| [122 164 148 103 71 56 78 83 93 103 119 139 102 61 69 84]] |



All pixels change when
the camera moves!

Challenges of Image Modeling



Illumination



Occlusion



Deformation



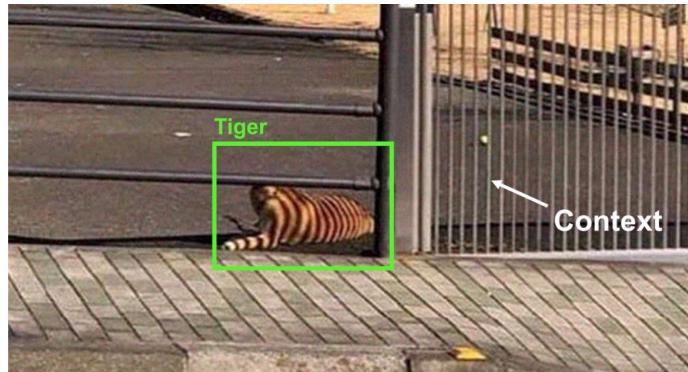
Background Clutter



Challenges of Image Modeling



Context



Intra-class variation





Manual Features?

- Idea #2: Manual features?



Nose,
Eyes,
Mouth



Wheels,
License Plate,
Headlights



Door,
Windows,
Step
...



Nose,
Eyes,
Mouth,
Hair,
...



Manual Features

$$g(x) = \mathbf{w}^T x + w_0$$

Model

→ “cat”

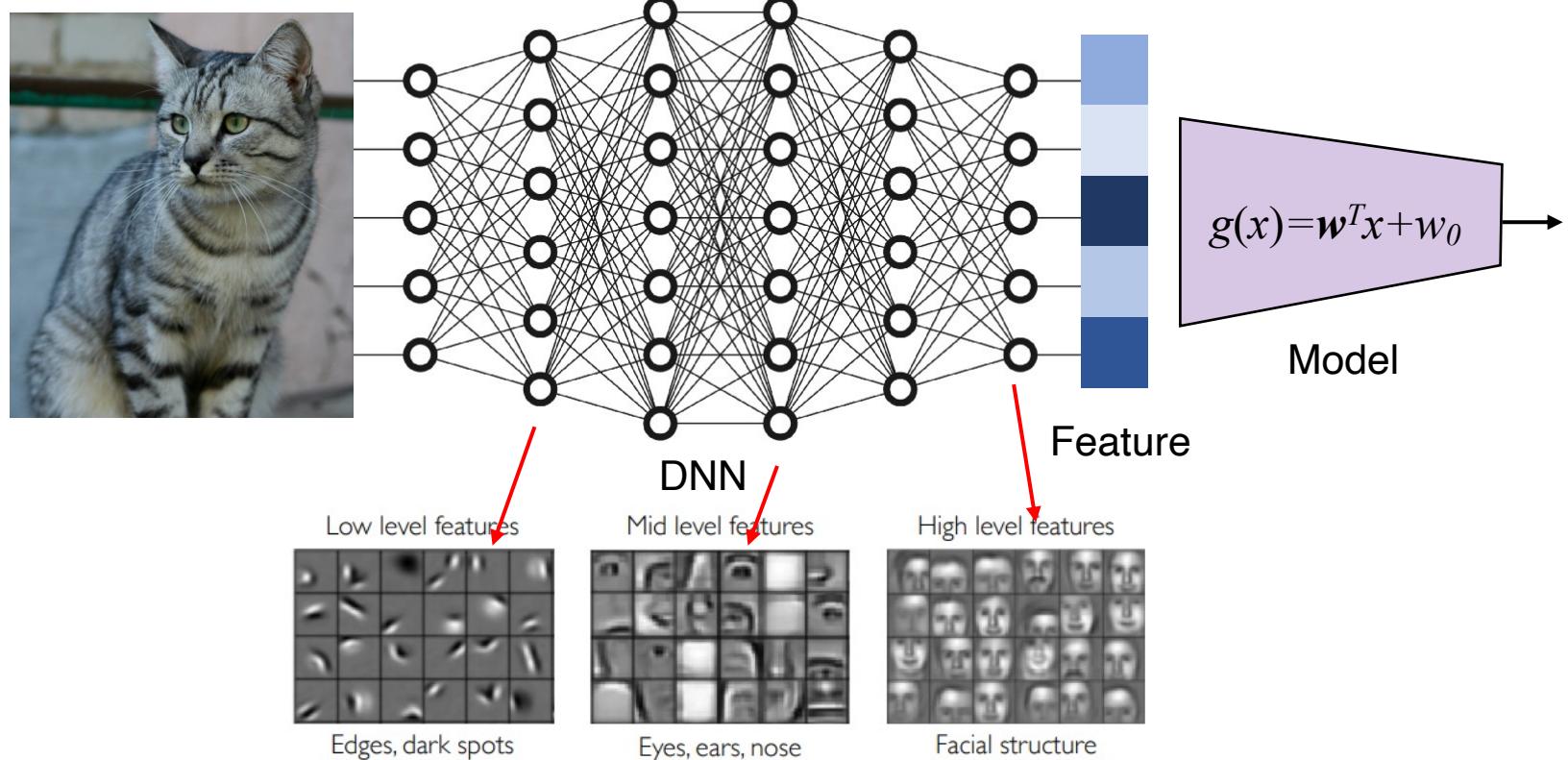
Any Problems?

Learning Feature Representations

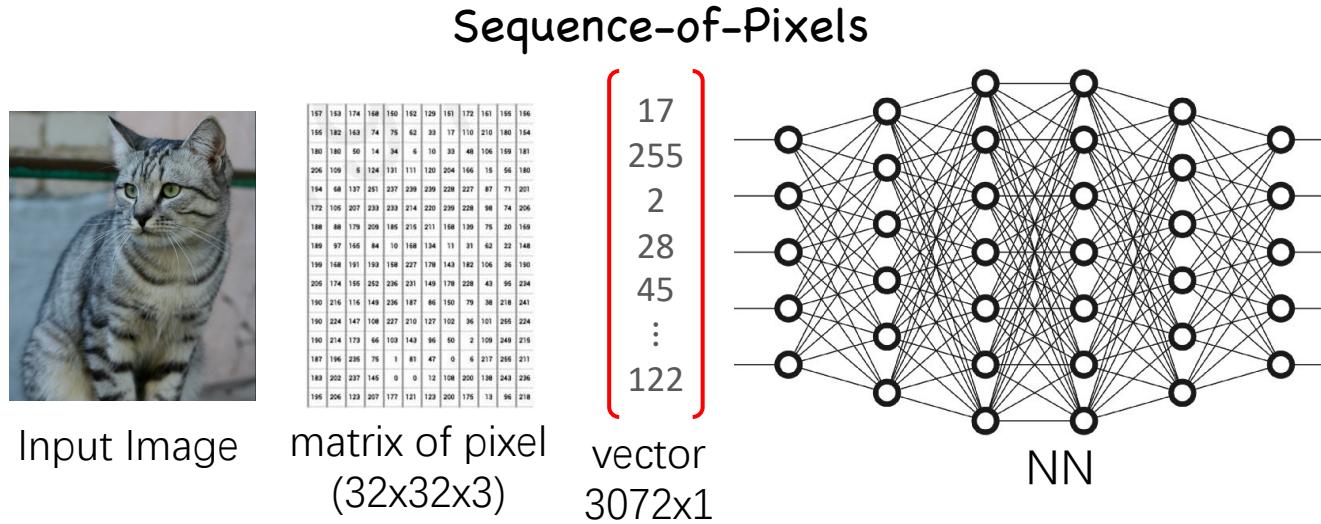


Idea #3: Deep Learning?

Can we automatically learn features from the data?



Learning Features Using Neural Networks



A Big Problem

All input pixels should be stretched into a 1-d vector

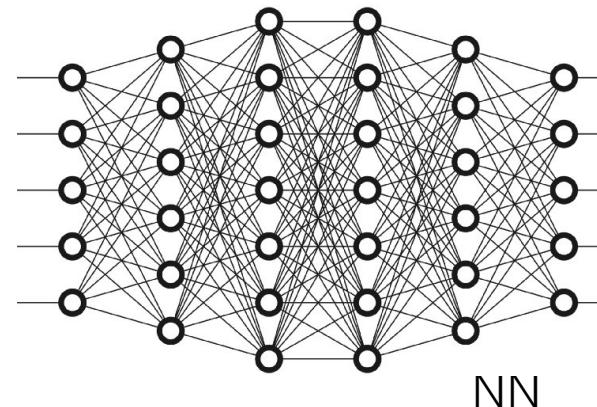
Learning Features Using Neural Networks



Other problems ?



Input Image



- Always concern the entire image
- Sensitive to scale, position,...

Problems with Standard NNs



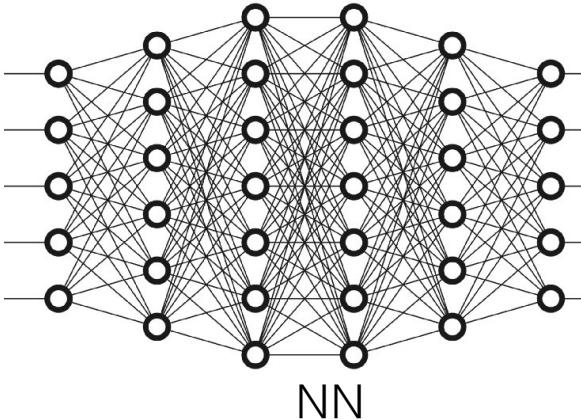
Input Image

| | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 156 | 154 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 4 | 10 | 33 | 48 | 106 | 159 | 181 |
| 205 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 154 | 68 | 137 | 231 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 106 | 207 | 233 | 233 | 214 | 220 | 239 | 238 | 98 | 74 | 206 |
| 186 | 88 | 178 | 209 | 185 | 215 | 211 | 158 | 130 | 76 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 159 | 168 | 191 | 193 | 158 | 227 | 179 | 143 | 182 | 106 | 36 | 150 |
| 205 | 174 | 186 | 232 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 195 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 160 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 256 | 224 |
| 159 | 214 | 172 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 248 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 256 | 211 |
| 183 | 202 | 237 | 145 | 0 | 9 | 12 | 108 | 200 | 138 | 243 | 236 |
| 150 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 178 | 13 | 96 | 218 |

matrix of pixel
(32x32x3)

$$\begin{bmatrix} 17 \\ 255 \\ 2 \\ 28 \\ 45 \\ \vdots \\ 122 \end{bmatrix}$$

vector
3072x1



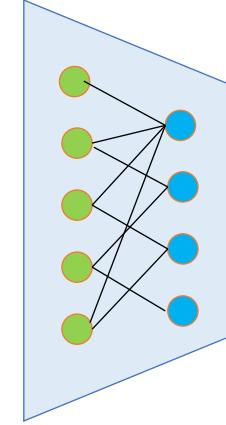
- **Seq-of-Pixels:** The spatial structure of images is destroyed
- **Locality insensitive:** concern the entire picture instead of local features.
- **Translation variant:** sensitive to the global position of a feature
- **Too many parameters:** fully connected layers yield a huge volume of parameters.



Modeling Images: Design Criteria

- To model images, we need to:
 1. Keep the spatial structure
 2. Sensitive to **locality**
 3. Handle **variants** in input images
 4. Share **parameters** across the spatial regions

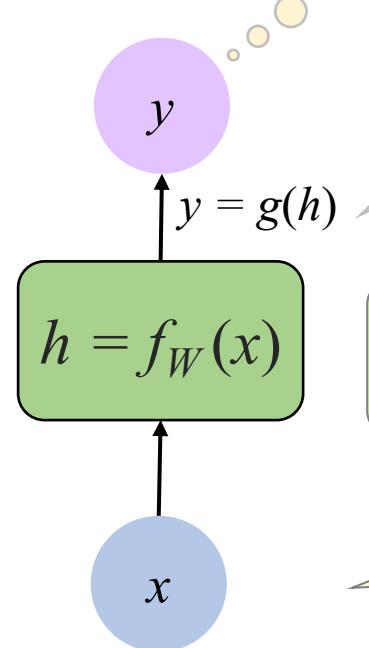
| | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 157 | 153 | 174 | 168 | 160 | 182 | 129 | 151 | 172 | 16 |
| 156 | 182 | 163 | 74 | 76 | 62 | 33 | 17 | 110 | 21 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 10 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 1 |
| 154 | 68 | 137 | 291 | 237 | 238 | 239 | 238 | 237 | 8 |
| 172 | 105 | 207 | 233 | 239 | 215 | 220 | 236 | 228 | 9 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 7 |
| 189 | 97 | 165 | 84 | 70 | 168 | 134 | 11 | 31 | 6 |
| 158 | 168 | 191 | 193 | 168 | 227 | 179 | 143 | 182 | 10 |
| 205 | 174 | 195 | 232 | 236 | 231 | 149 | 178 | 228 | 4 |
| 190 | 216 | 116 | 149 | 236 | 187 | 66 | 190 | 79 | 3 |
| 193 | 224 | 147 | 198 | 227 | 213 | 127 | 102 | 36 | 10 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 10 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 21 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 106 | 200 | 13 |
| 195 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 1 |





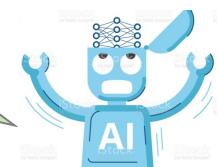
From MLP to CNN

I process all pixels at once
without being aware of their
spatial, scale...



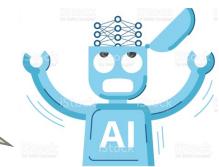
How can I “distinguish” the spatial
of different objects?

Why not just focus on small, local
regions and **scan** for interested objects?



But, **how** can I “focus”?

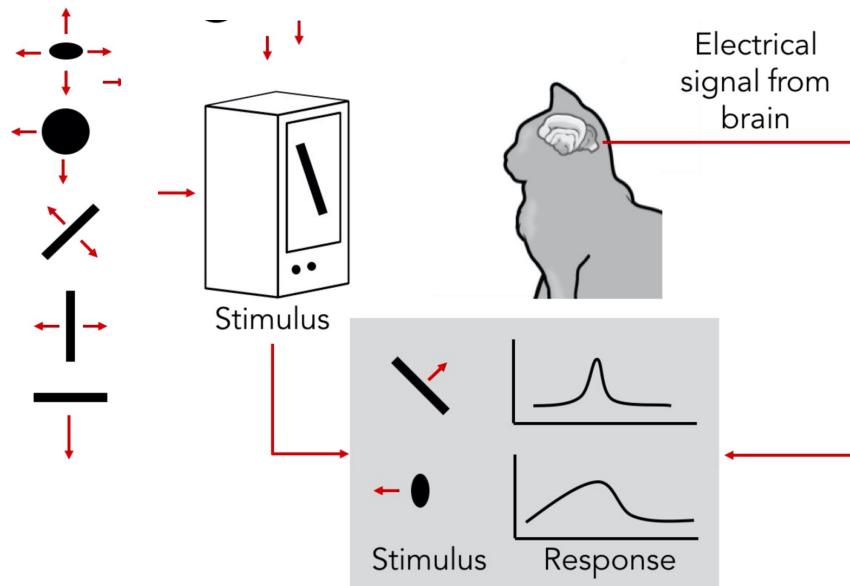
Let's get some inspirations from
human brains ..



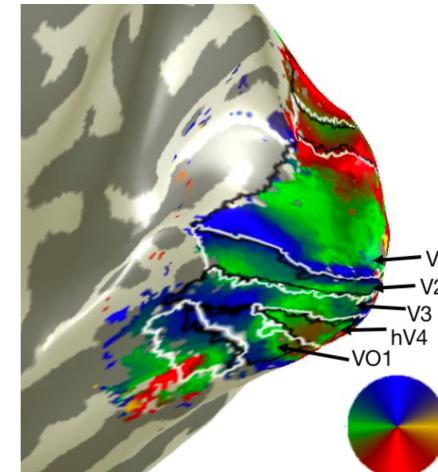
Inspired by Biology



- Each neuron in the cat's striate cortex (大脑皮层) has a **receptive field** of a specific shape.
[Hubel & Wiesel, 1959]



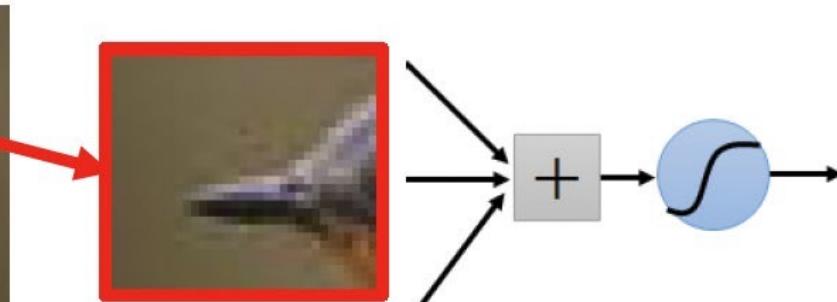
And nearby cells in cortex represent nearby regions in the visual field



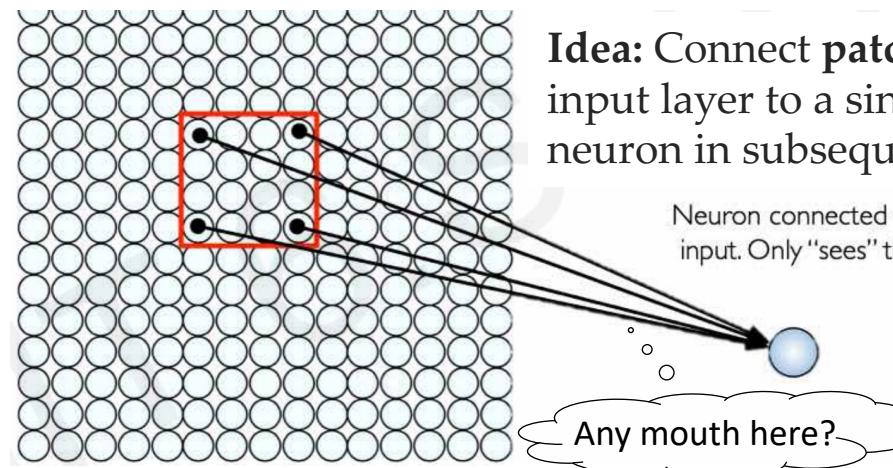
Convolution: The Idea



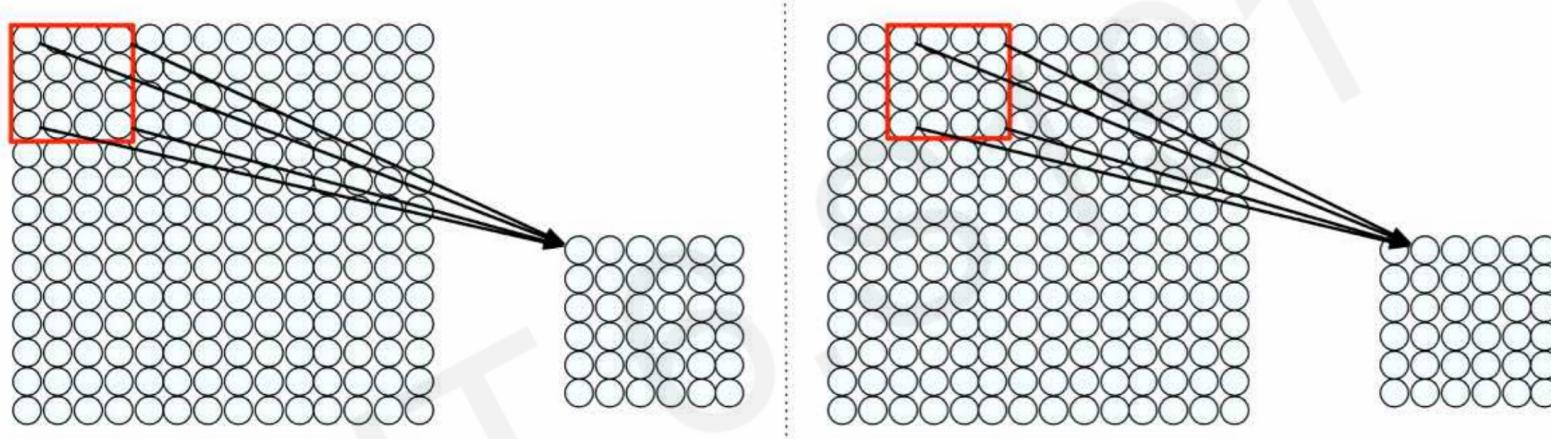
- This inspires us to **scan** over the image for a specific feature through weighing and activating.



Input: 2D image.
Array of pixel values



Convolution: The Idea



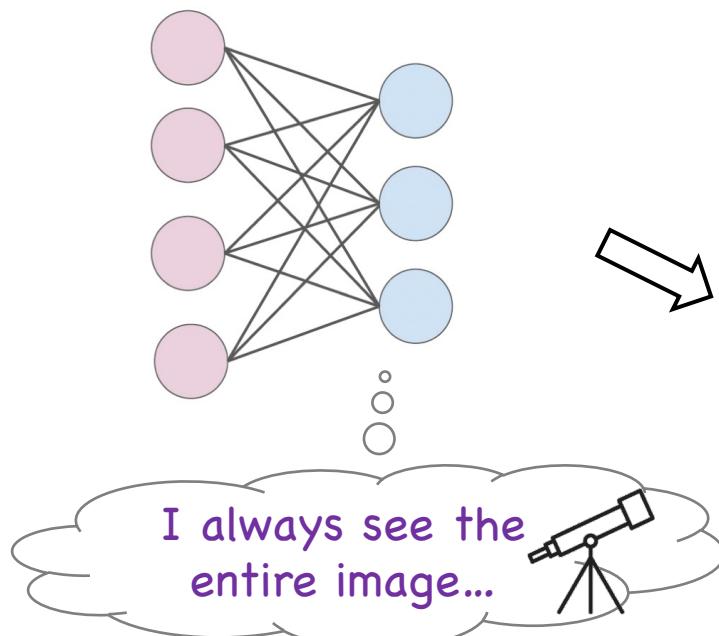
- **Slide** over the image spatially, computing feature transformation (how?)
- During sliding, the weights for all neuron must be **shared** (why?)

MLP vs CNN



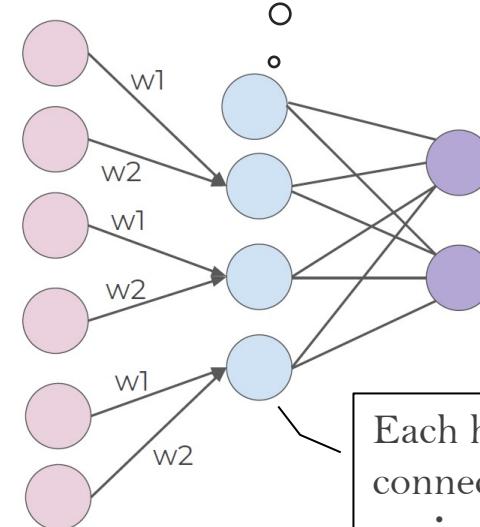
- Restrict the scope of hidden units to focus on **local** features.
- **Sharing weights** among hidden units

Fully-connected NN



I only see small regions for what I am interested.

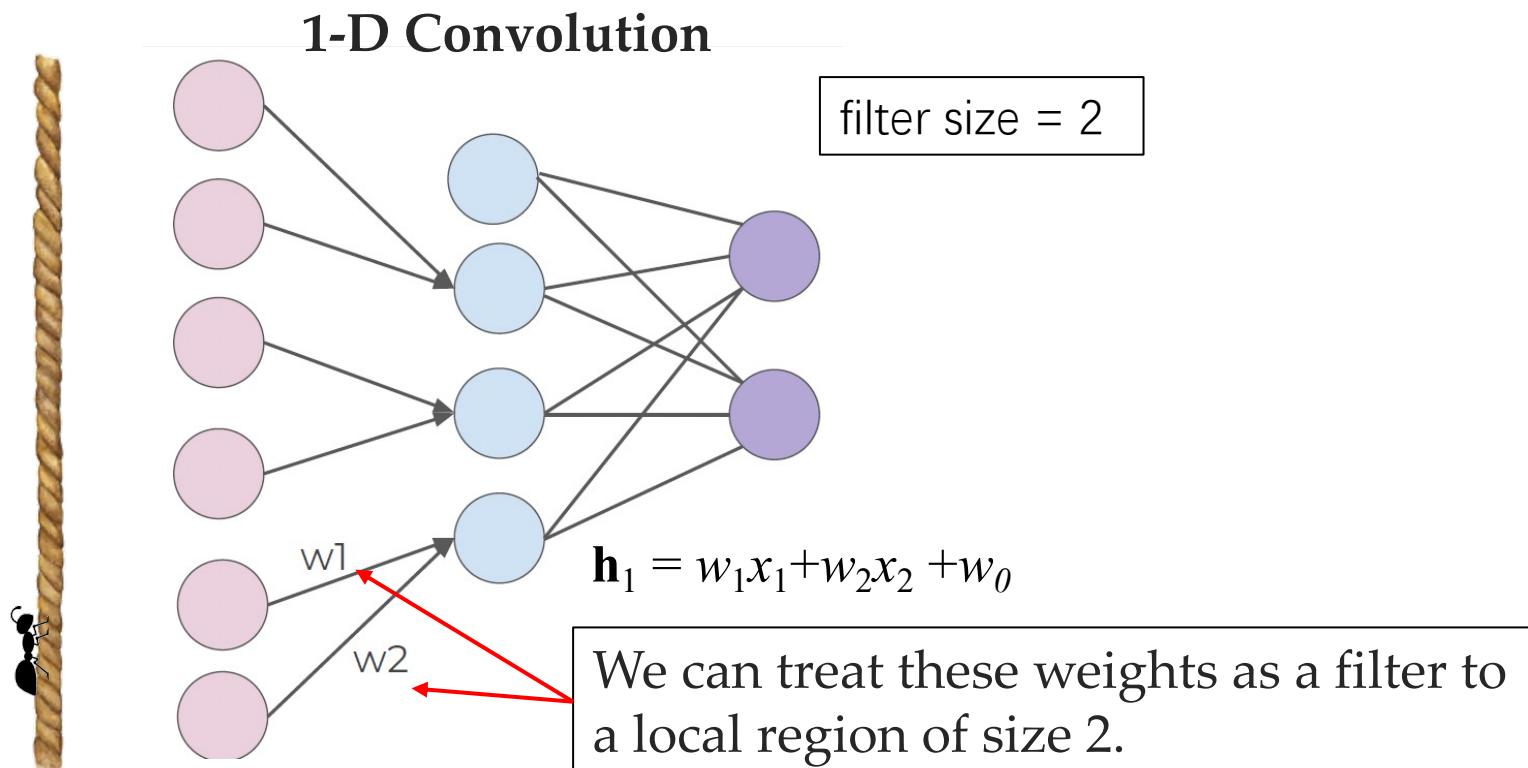
Locally-connected NN



Convolutions and Filters (1-D)



- Filter (a.k.a., convolutional kernel): weights assigned to the input region by a hidden unit.

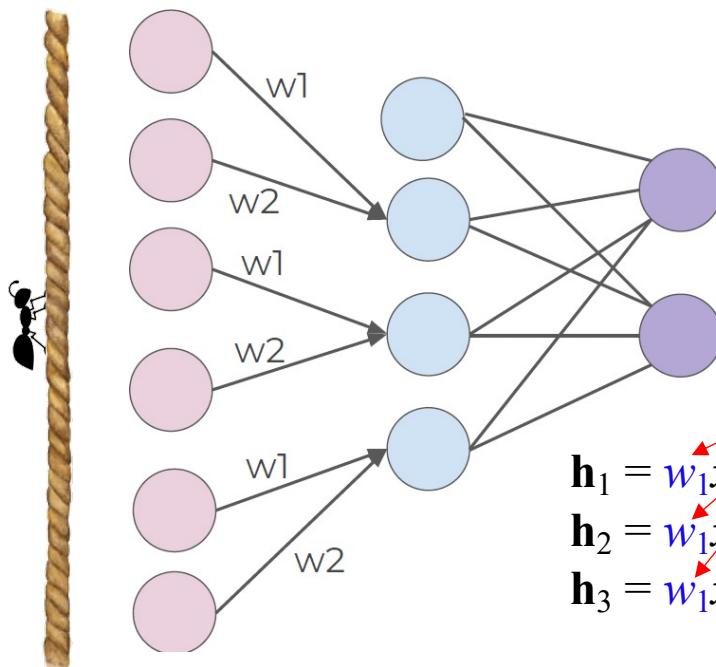


Convolutions and Filters (1-D)



- We now apply the **same** set of weights over the **whole** picture.

1-D Convolution



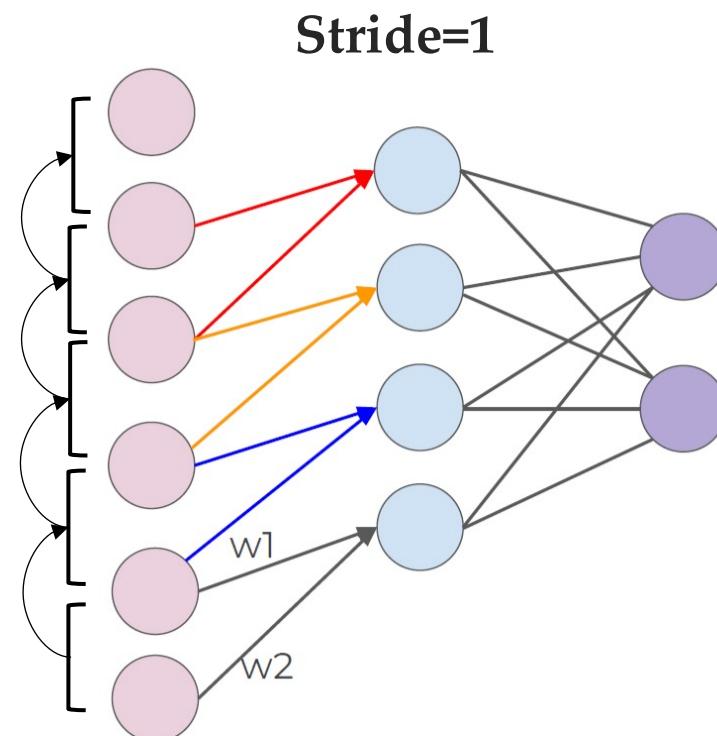
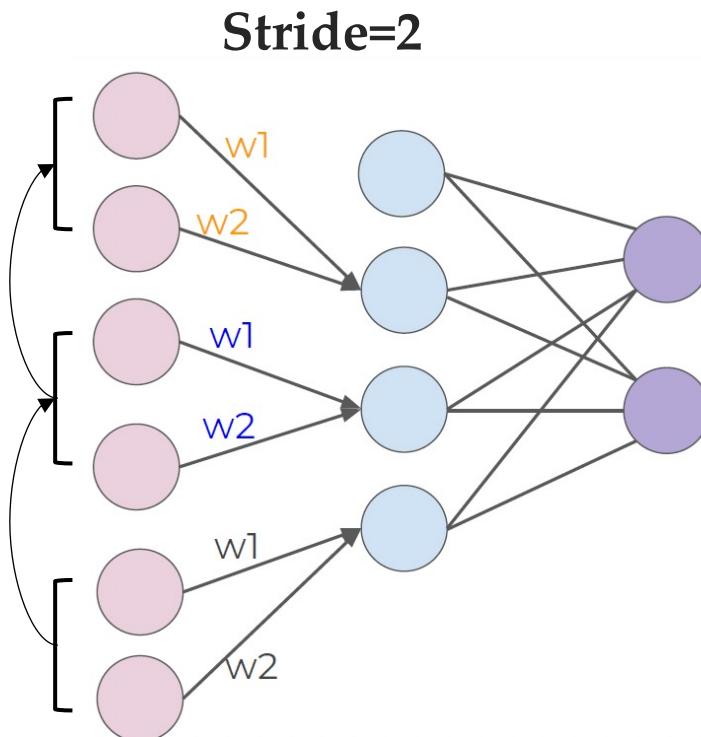
Parameters are **shared** over the whole picture for one filter.

$$\begin{aligned}h_1 &= w_1x_1 + w_2x_2 + w_0 \\h_2 &= w_1x_3 + w_2x_4 + w_0 \\h_3 &= w_1x_5 + w_2x_6 + w_0\end{aligned}$$

Convolutions and Filters



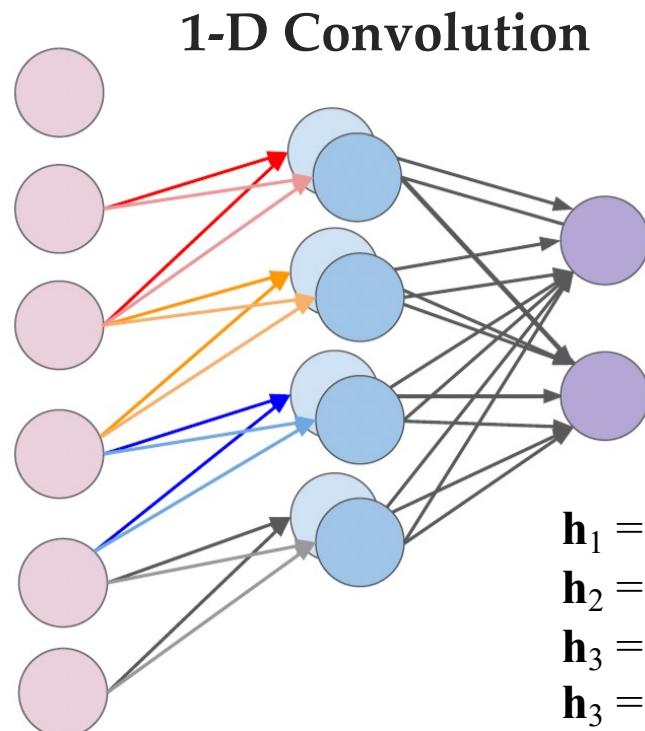
- Note that the filter scans every 2 pixels in this example.
- We can control the filter to scan every N pixels ($N=\text{stride}$).



Convolution and Filters



- We can then expand this idea to **multiple filters**.
- Each filter is detecting a different feature



filters = 2
filter size = 2
stride = 1

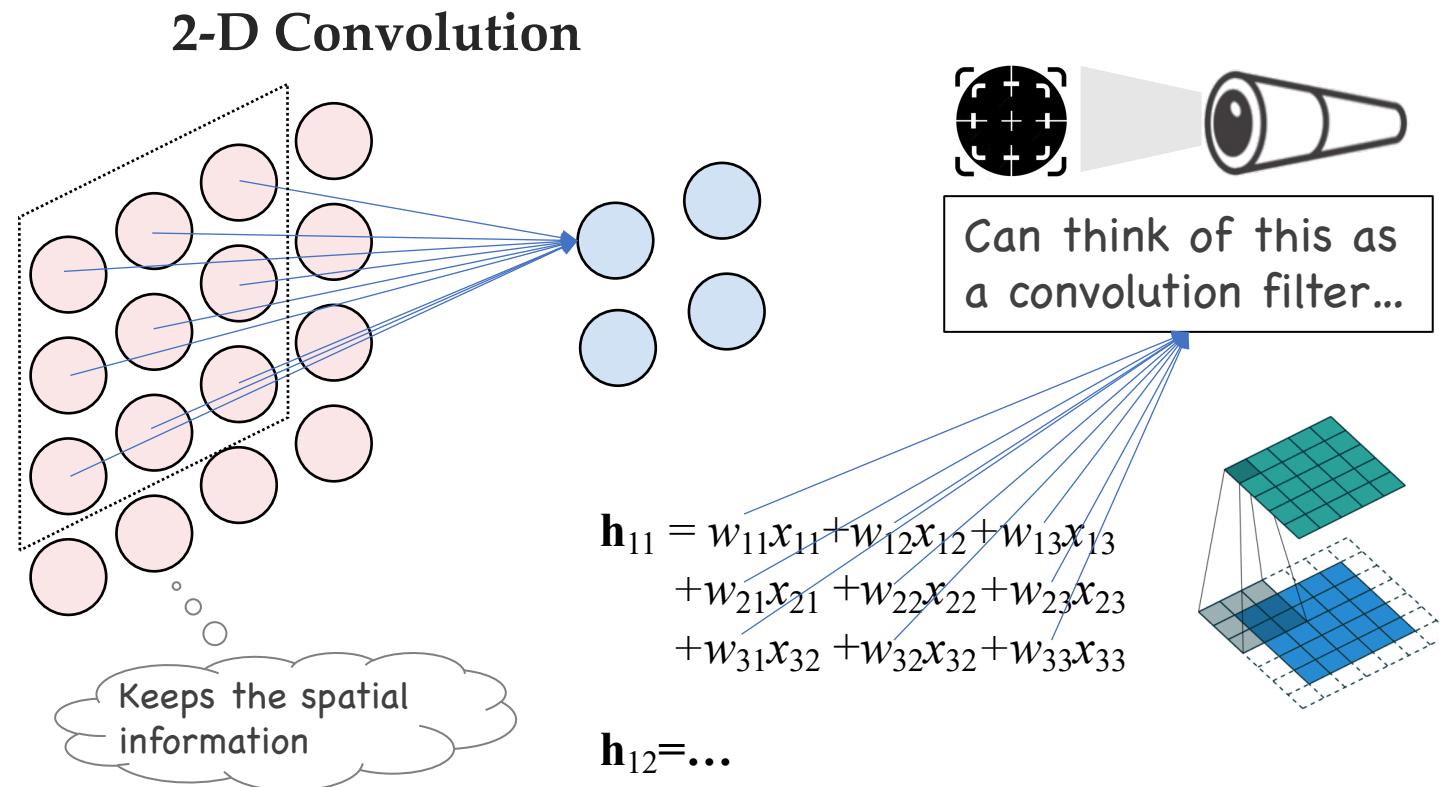
$$\begin{aligned}\mathbf{h}_1 &= w_1x_1 + w_2x_2 + w_0 \\ \mathbf{h}_2 &= w_1x_2 + w_2x_3 + w_0 \\ \mathbf{h}_3 &= w_1x_3 + w_2x_4 + w_0 \\ \mathbf{h}_3 &= w_1x_4 + w_2x_5 + w_0\end{aligned}$$

$$\begin{aligned}\mathbf{h}_1 &= w'_1x_1 + w'_2x_2 + w'_0 \\ \mathbf{h}_2 &= w'_1x_2 + w'_2x_3 + w'_0 \\ \mathbf{h}_3 &= w'_1x_3 + w'_2x_4 + w'_0 \\ \mathbf{h}_3 &= w'_1x_4 + w'_2x_5 + w'_0\end{aligned}$$

2-D Convolutions



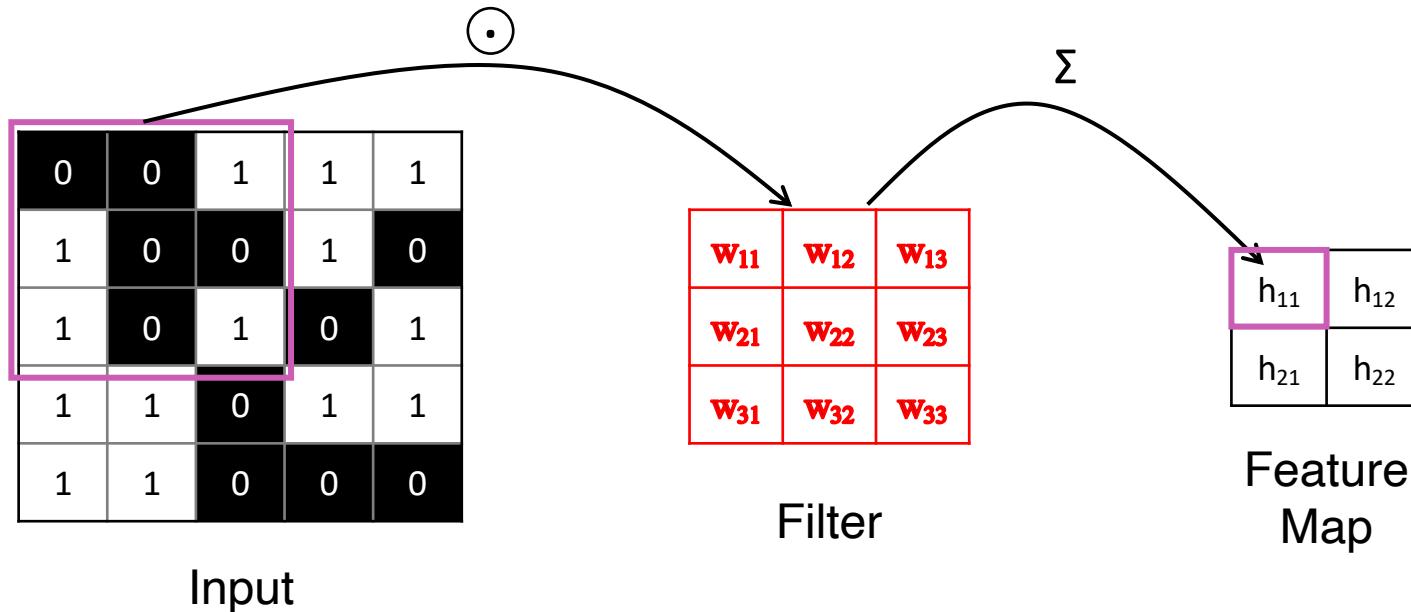
- Now let's expand these concepts to **2-D Convolution**, since we'll mainly be dealing with images.



2-D Convolutions

- Convolutional Operation

elementwise multiplication and sum of a filter and the image



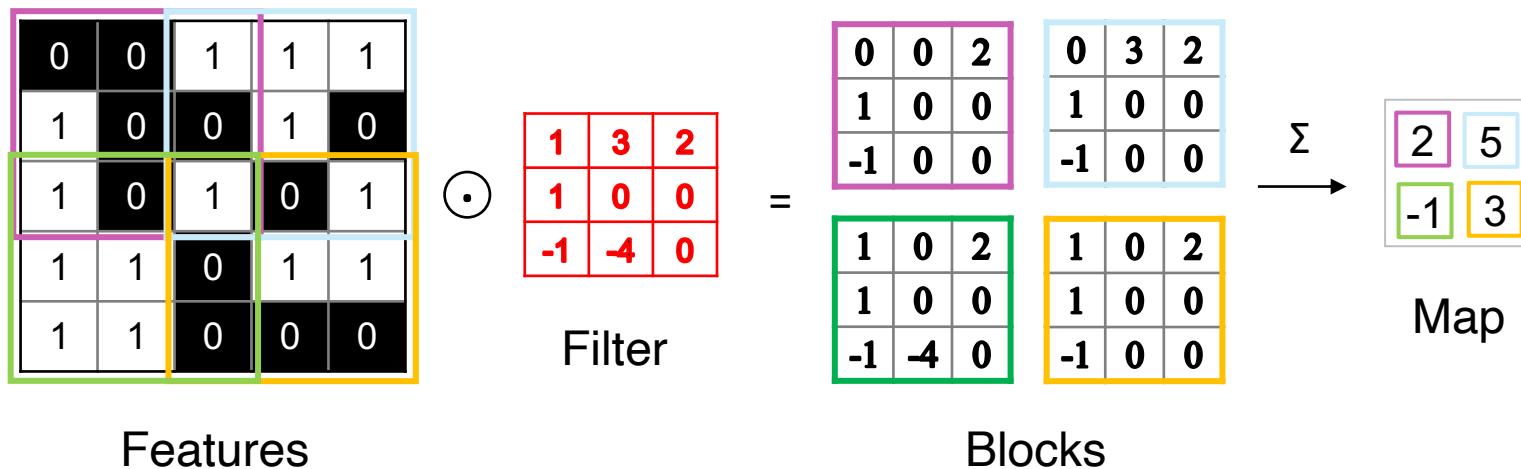
* We call it convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2]$$

2-D Convolutions

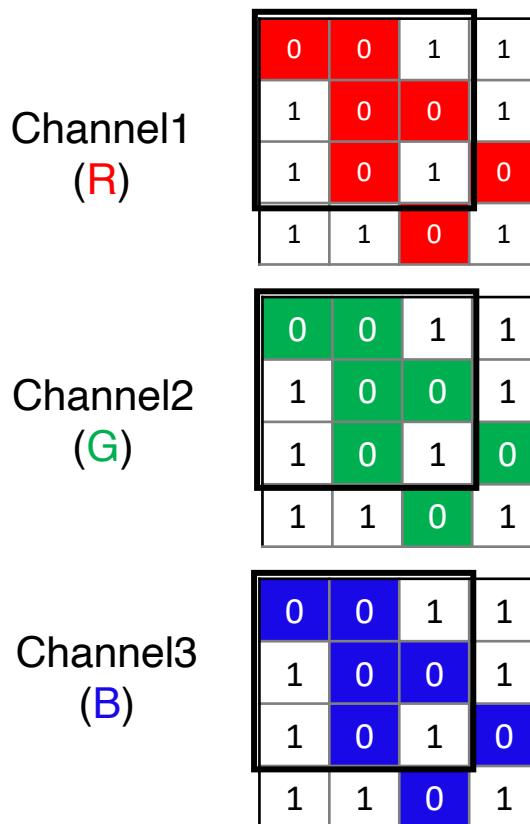


- Example:

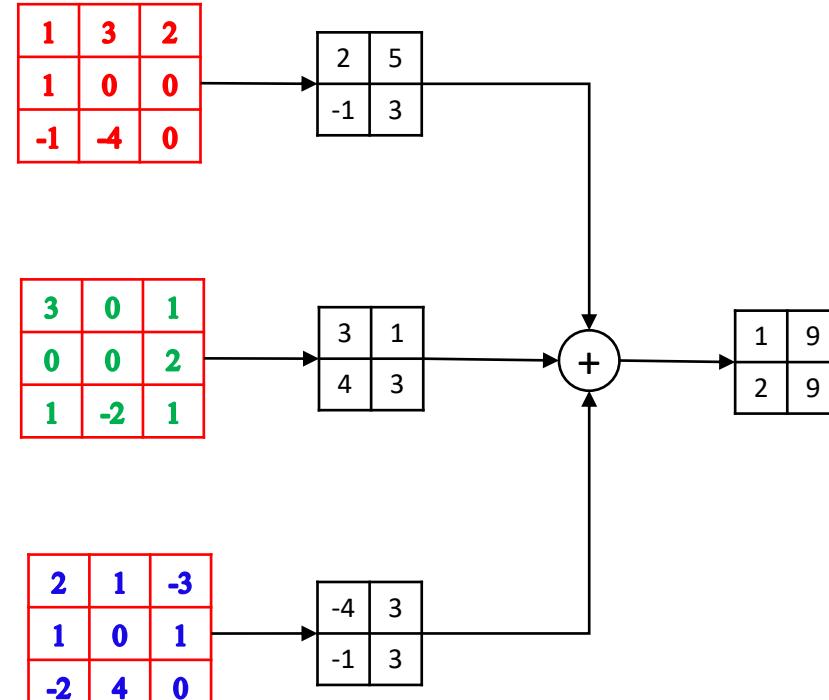


Multiple Input Channels

- What if we have color?



So it is actually 3-D convolution...

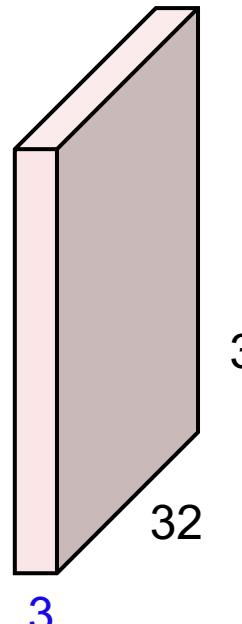


“3-D” Convolution



- 3-D Image and Filter

32x32x3 image



Filters always extend the full depth of the input volume

5x5x3 filter

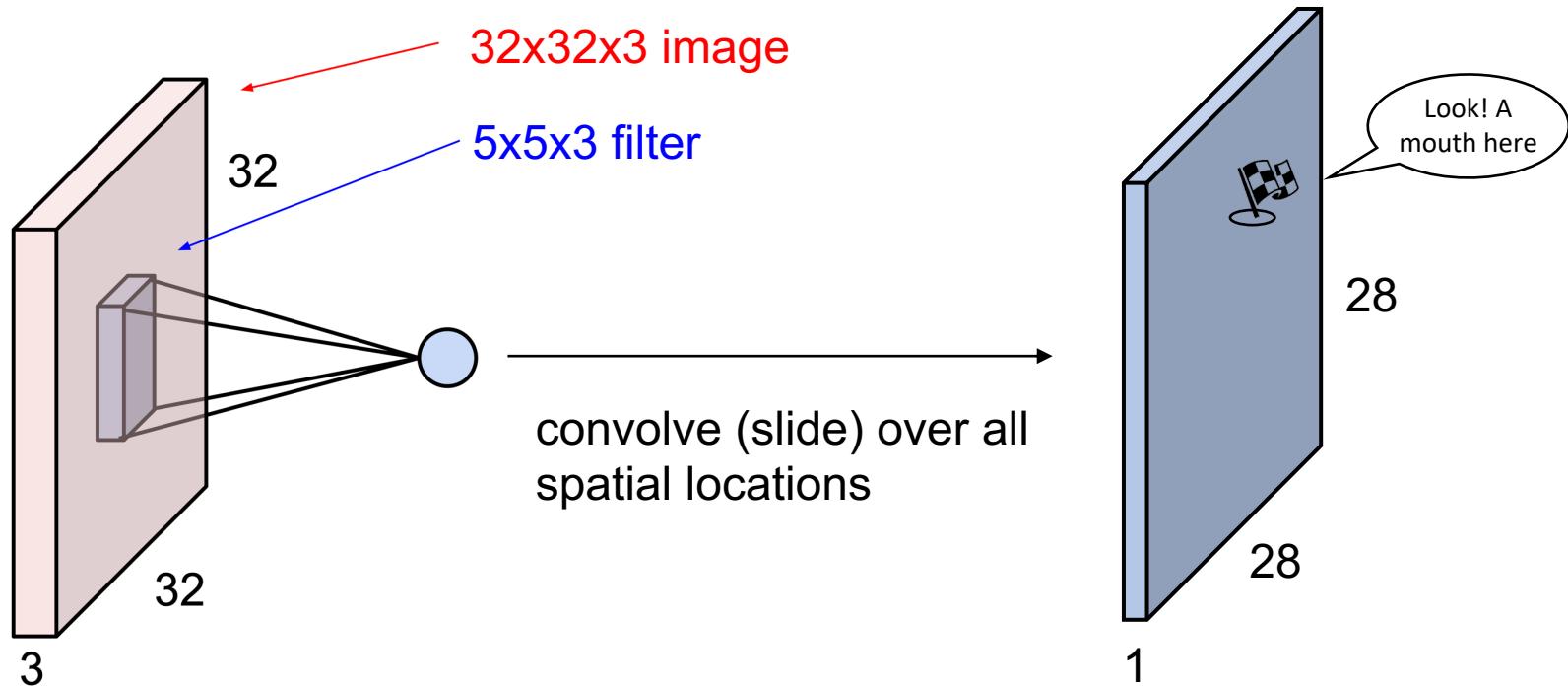


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Feature Map

- A map that stores the locations of a specific feature activated by a filter.

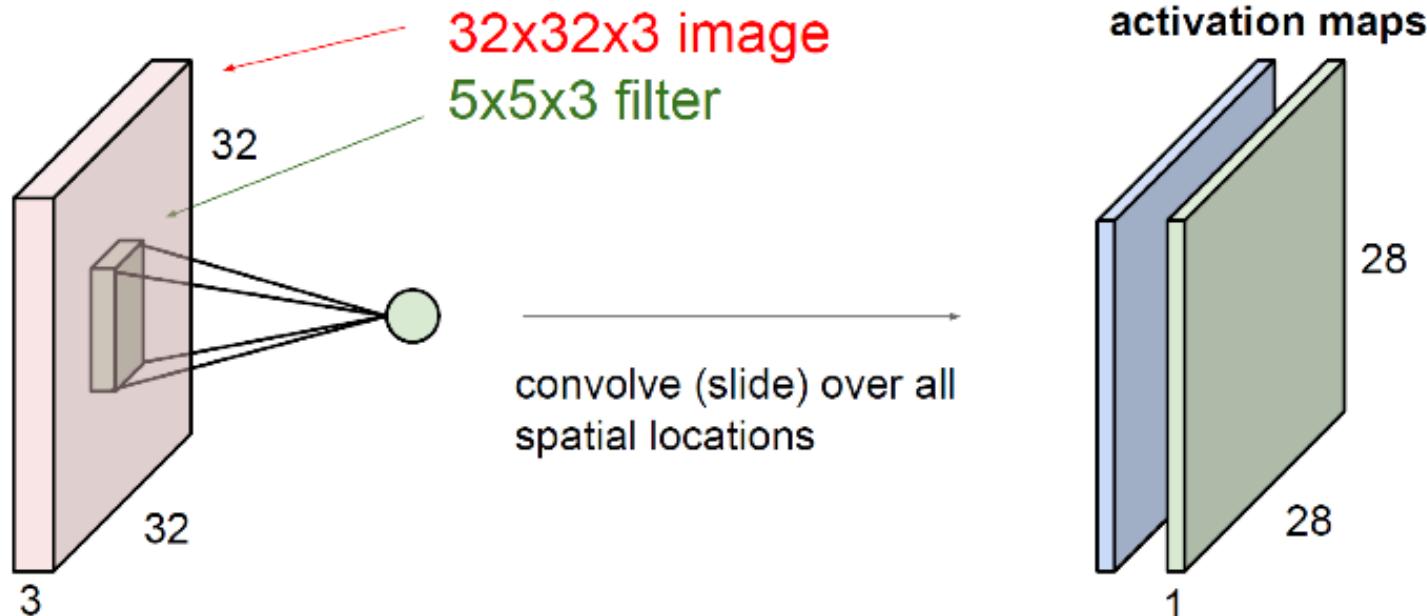
feature (activation) map



Multiple Filters

- Using multiple filters to scan multiple features.

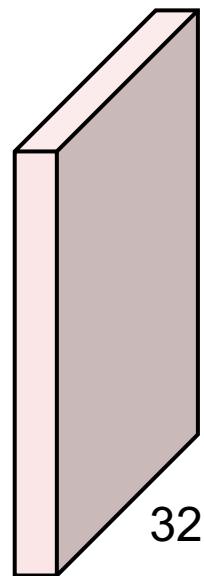
consider a second filter



Multiple Feature Maps

- For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

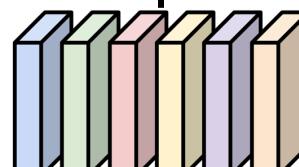
3x32x32 image



Input
channels

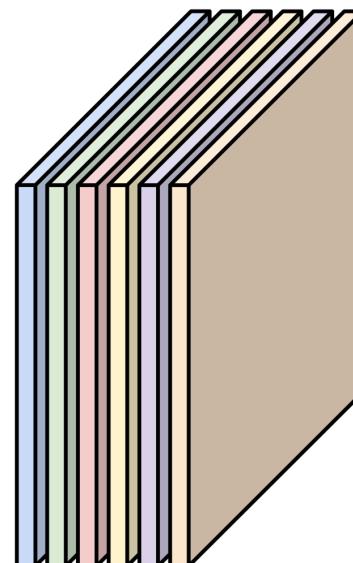
6 filters, each 5x5

Convolution
Layer



6x3x5x5 filters

6 activation maps,
each 1x28x28



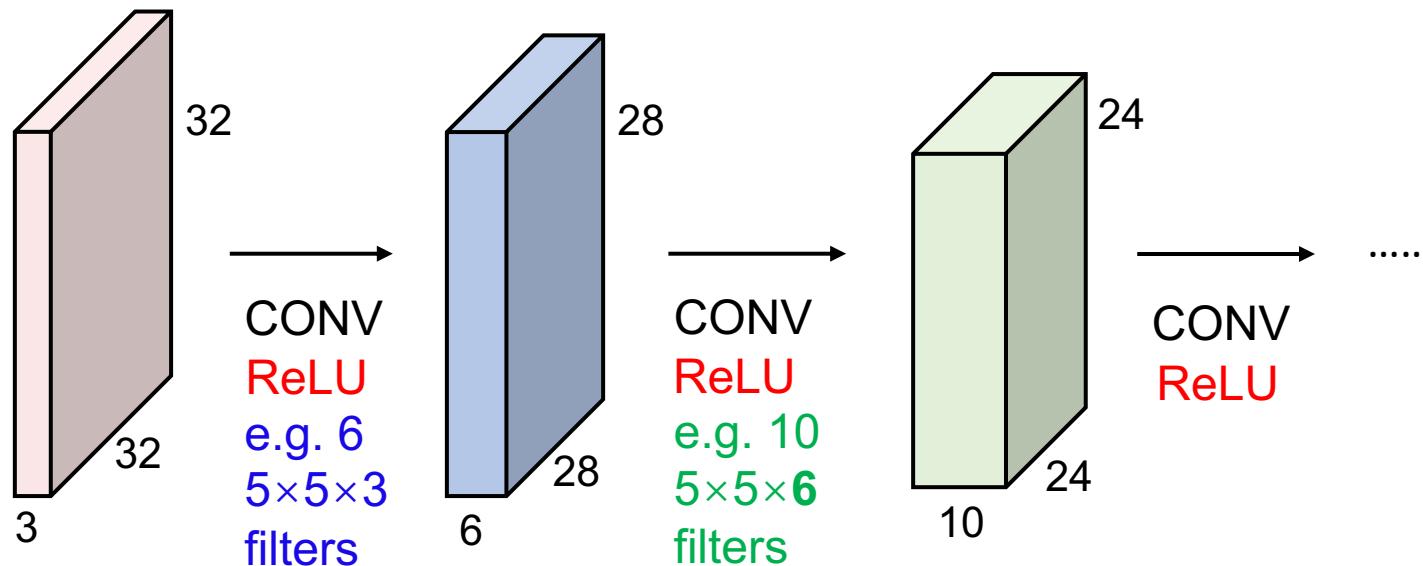
Stack activations to get a
6x28x28 output image!

Output
channels



Multiple Layers

- A sequence of convolutional layers, interspersed with activation functions (e.g., ReLU).

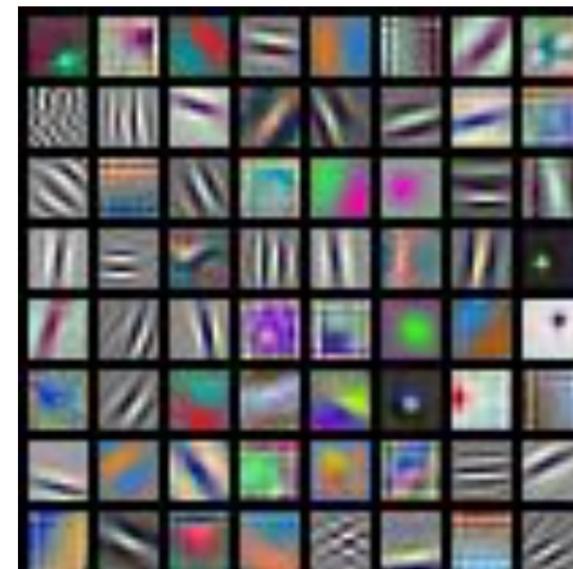
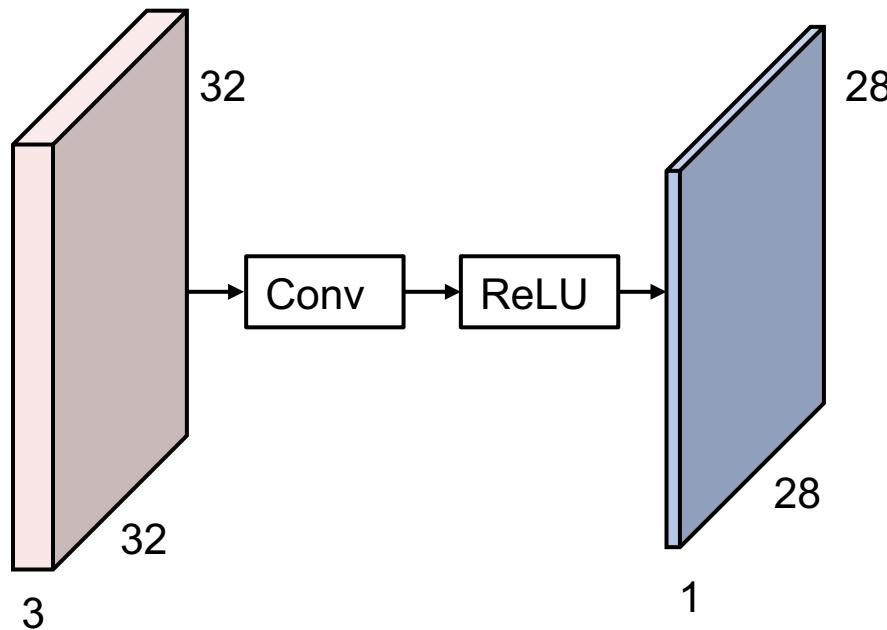




Multiple Layers

- What do convolutional filters learn?

First-layer conv filters: local image templates
(Often learns oriented edges, opposing colors)



AlexNet: 64 filters, each 3x11x11



Multiple Layers

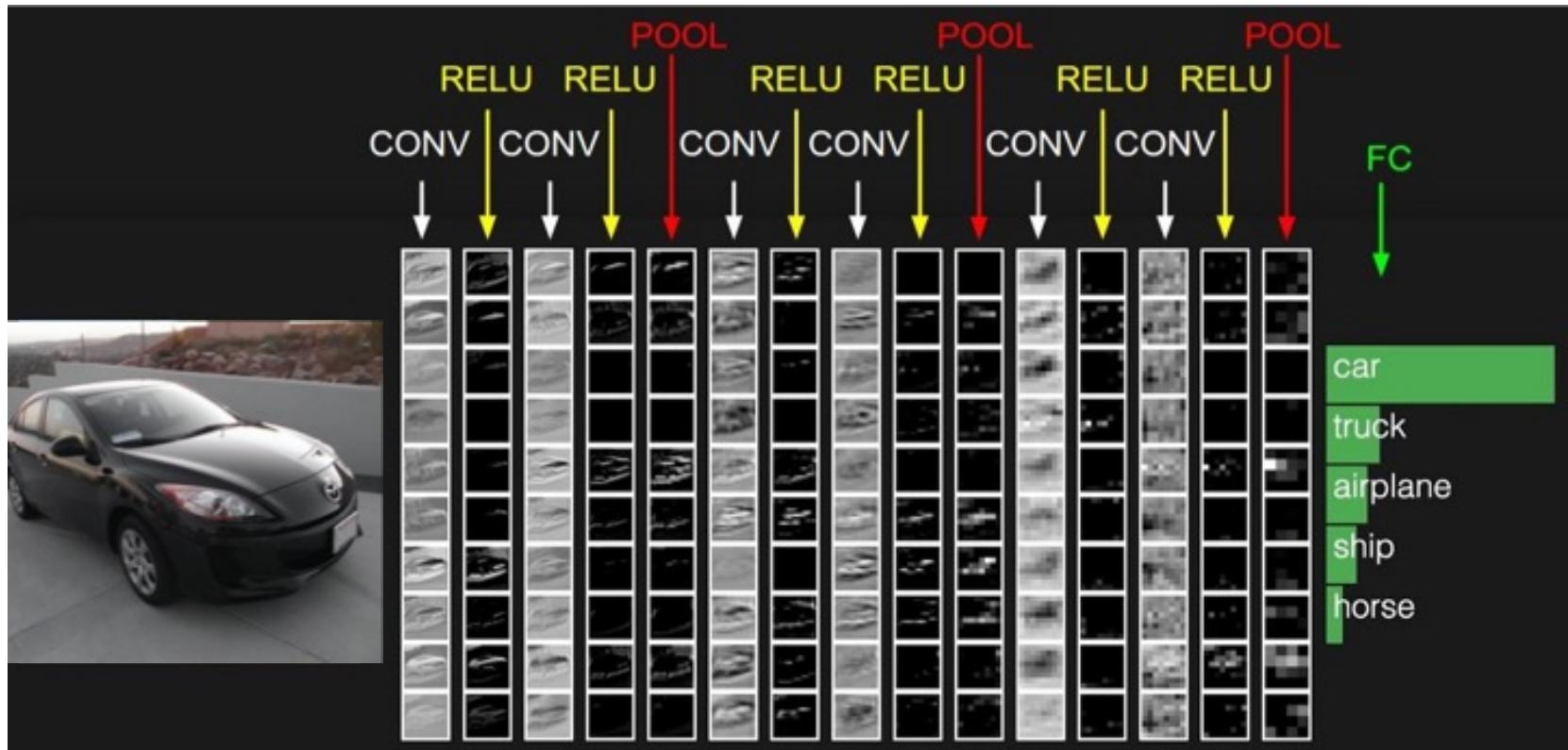
- What does each convolutional filter learn?

Example: 32 5x5 filters one filter \Rightarrow one feature map



Multiple Layers

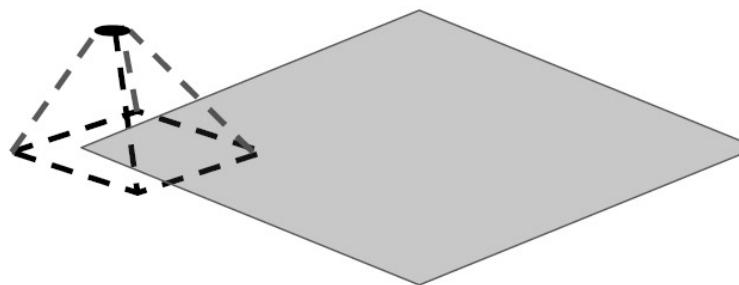
- What does each layer learn?



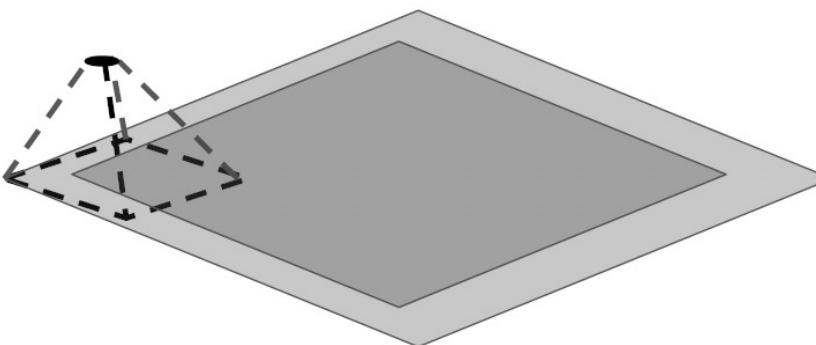
Padding



- We run into a possible issue for edge neurons! There may not be an input there for them.



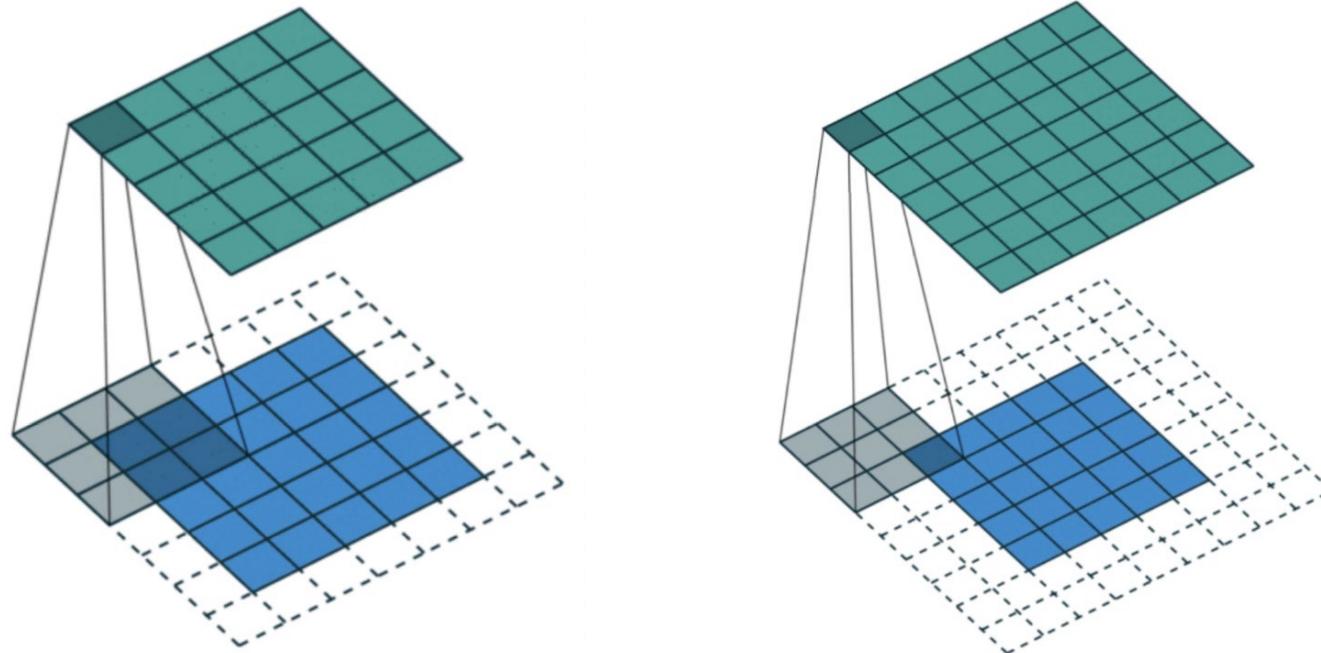
- We can fix this by adding a “**padding**” of zeros around the image.



Padding

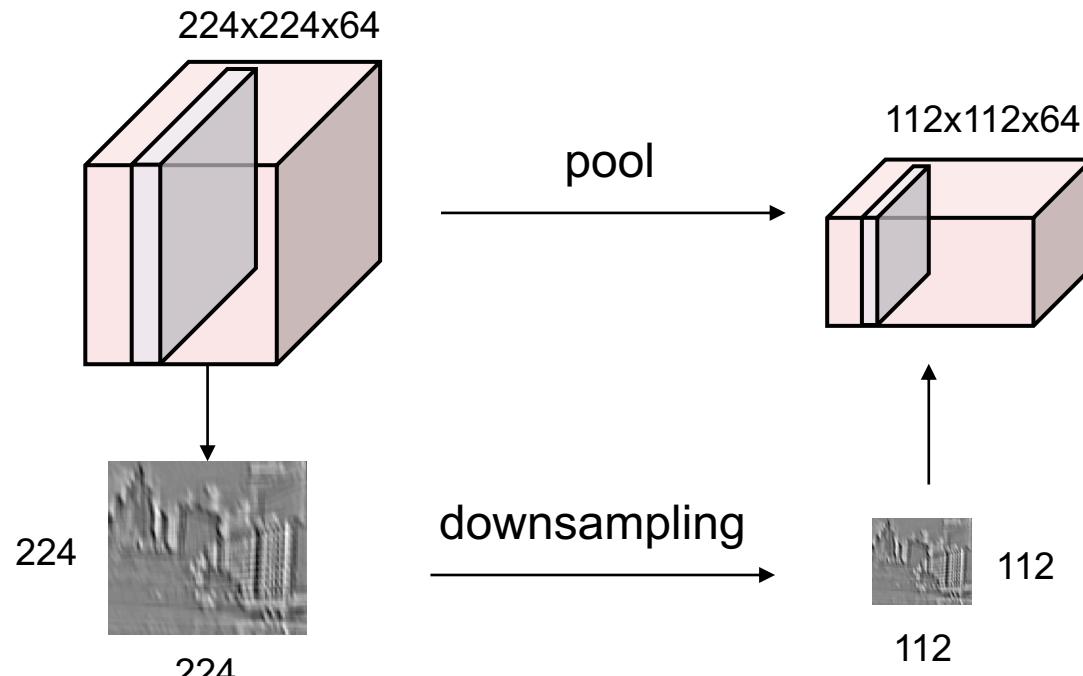


- **(Zero-)Padding:** symmetrically adding zeroes to the input matrix.
- allows the size of the input to be adjusted to our requirement.



Pooling

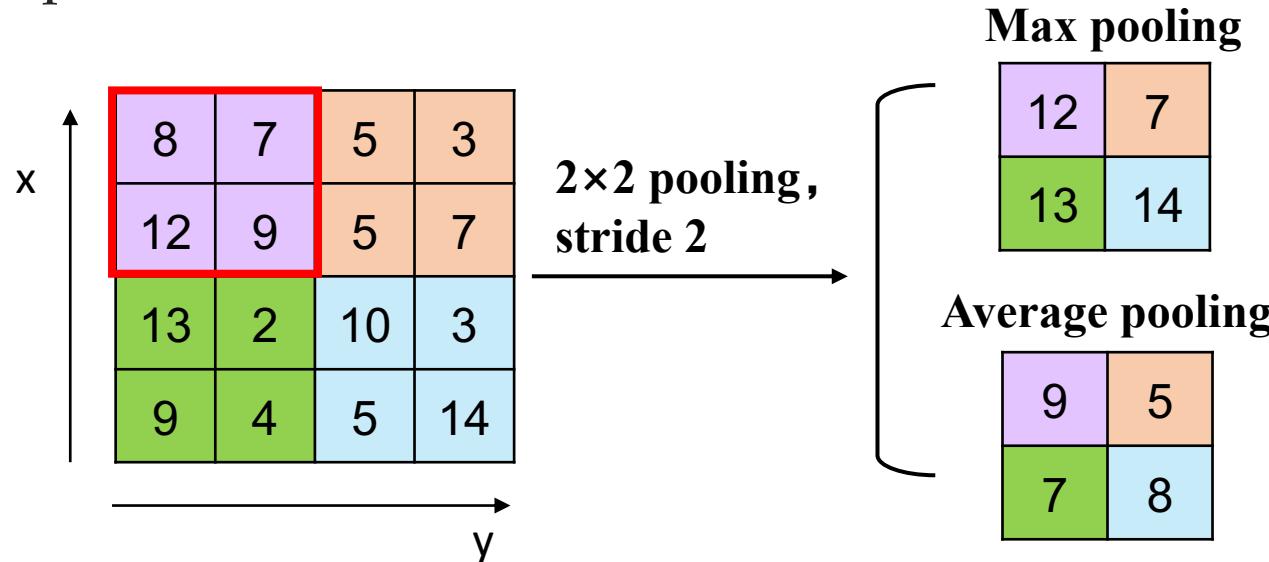
- **Downsample** the feature map to reduces the memory use.
- Makes the representations smaller and more manageable



Pooling does not change the object

Pooling

- **Max pooling:** The maximum pixel value of the patch is selected.
- **Average pooling:** The average value of all the pixels in the patch is selected.

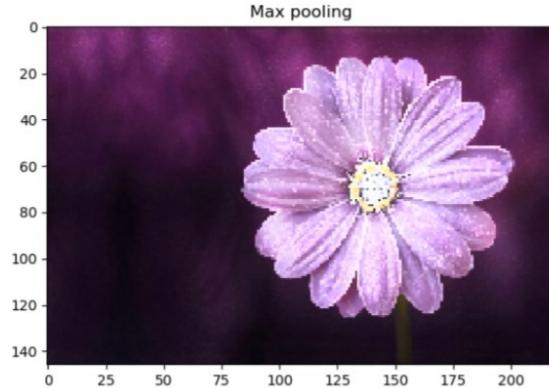


No learnable parameters
Introduces spatial invariance

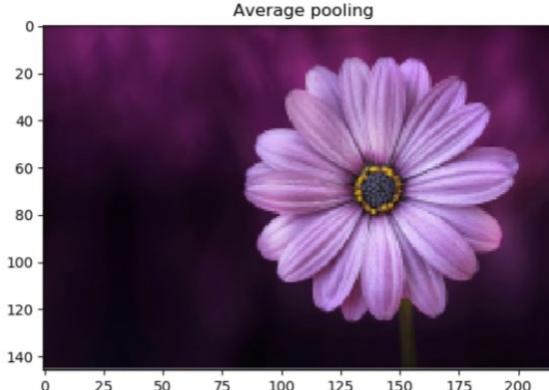
Pooling



- In the following example, a filter of 9×9 is chosen.

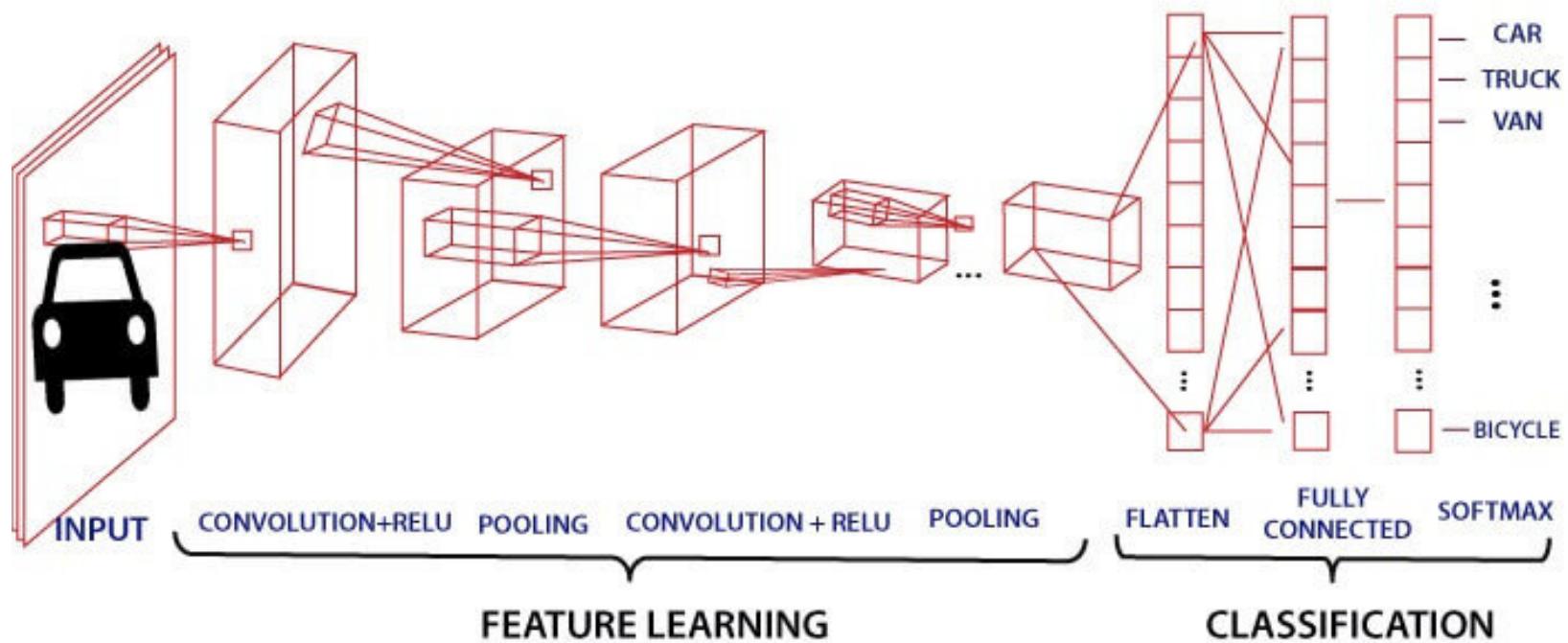


Max pooling selects the **brighter** pixels from the image.

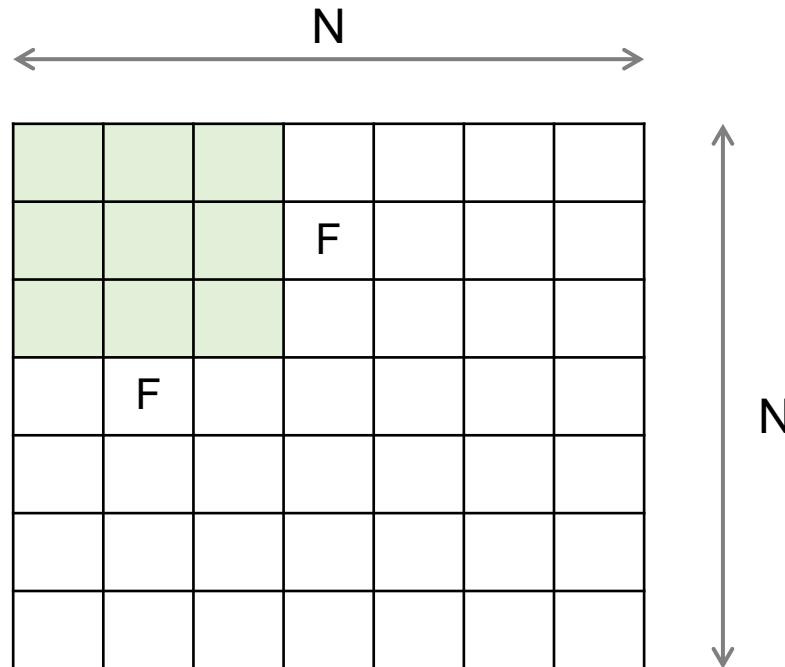


Average pooling method **smooths** out the image and hence the sharp features may not be identified.

The Big Picture



A Closer Look at Spatial Dimensions



Input size: **NxN**
Filter size: **FxF**

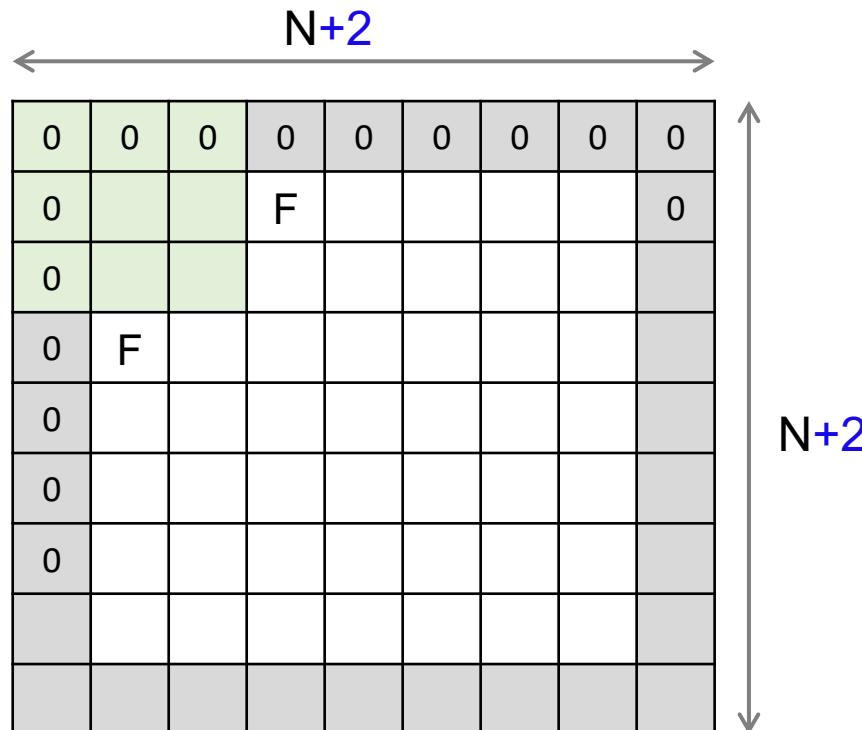
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => $(7 - 3)/1 + 1 = 5$
stride 2 => $(7 - 3)/2 + 1 = 3$
stride 3 => $(7 - 3)/3 + 1 = 2.33 :\backslash$

A Closer Look at Spatial Dimensions



In practice: Common to zero pad the border



Pad with P pixel border

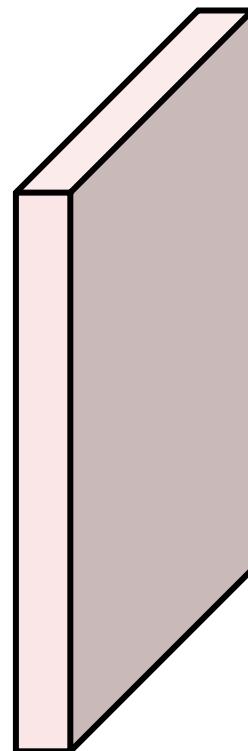
Output size:
 $(N + 2P - F) / \text{stride} + 1$

e.g. $N = 7$, $F = 3$:
stride 1 $\Rightarrow (7 + 2 - 3)/1 + 1 = 7$
stride 2 $\Rightarrow (7 + 2 - 3)/2 + 1 = 4$
stride 3 $\Rightarrow (7 + 2 - 3)/3 + 1 = 3$

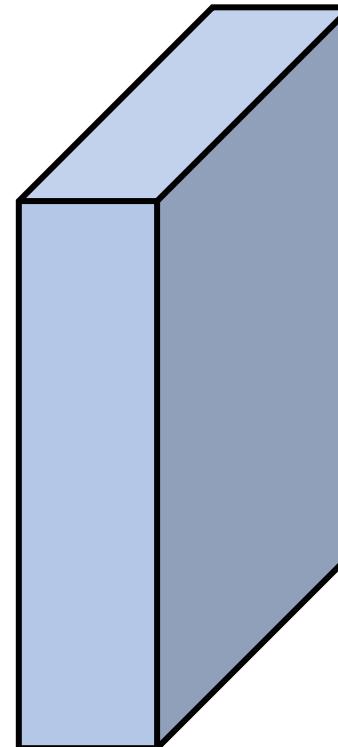
Quiz



- What is the output size ? Hint: $(N+2P-F)/\text{stride}+1$



10 5x5 filters
with stride 1,
pad 2



Input volume:
 $32 \times 32 \times 3$

Output volume size = ?

Thinking..



- Can we let CNN act as MLP? How?
- Can we apply CNN for texts?

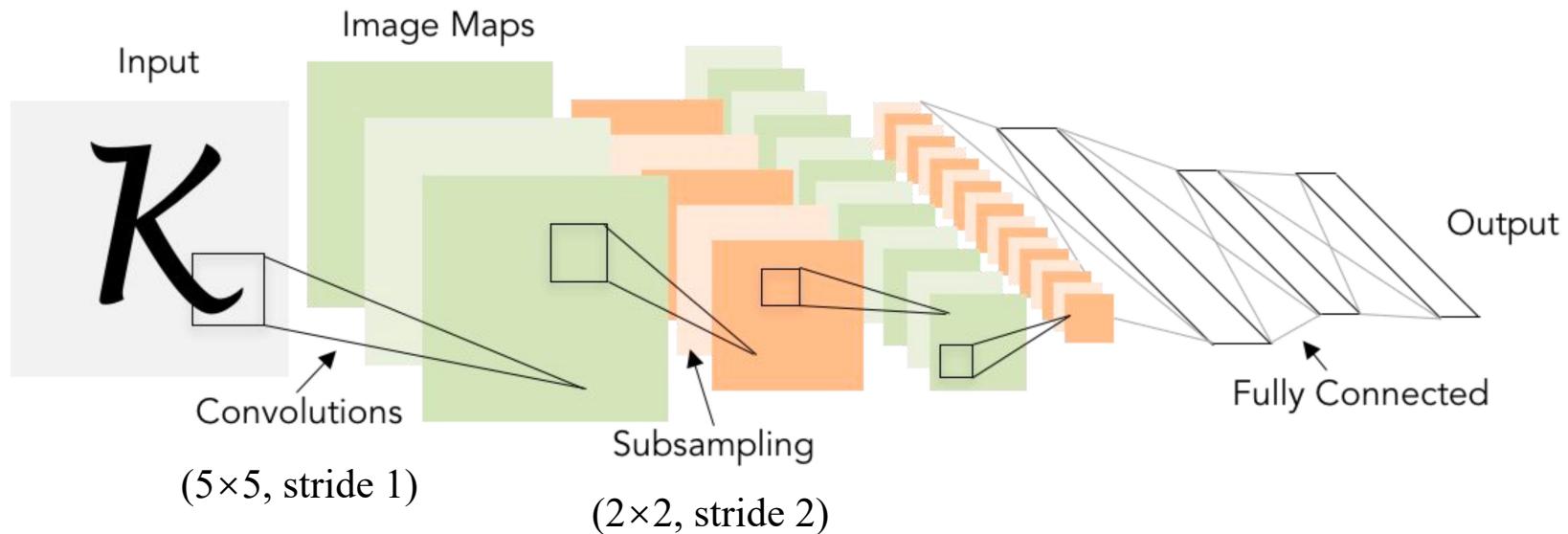




Some CNNs

LeNet-5

[LeCun et al., 1998]



Architecture: [CONV1-POOL1-CONV2-POOL2-FC1-FC2]

Conv filters were 5×5 , applied at stride 1

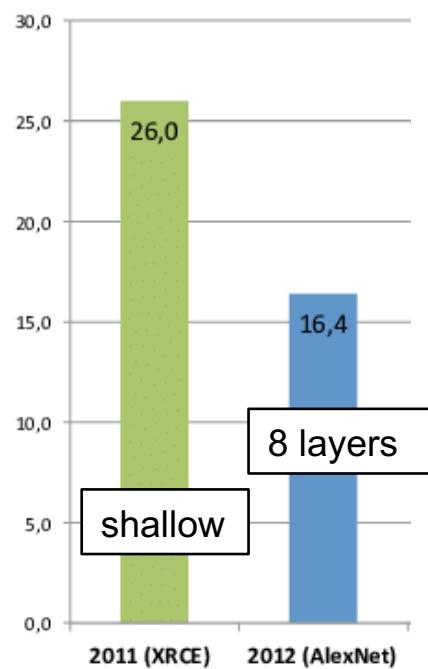
Subsampling (Pooling) layers were 2×2 applied at stride 2

AlexNet

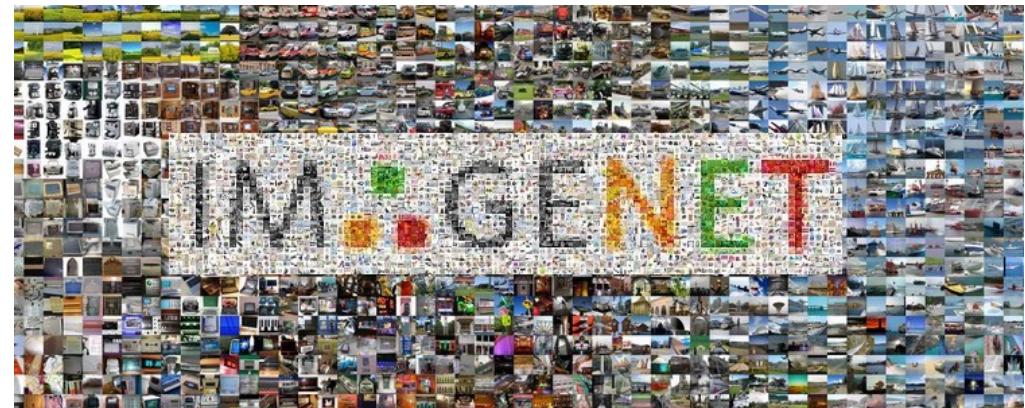


- The first deep learning winner

ImageNet Classification Error (%)



The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



ImageNet is a large scale image database organized according to the [WordNet](#) hierarchy, in which each node is depicted by thousands of images. **ImageNet** contains 14,197,122 images within more than 20k categories.

AlexNet [Krizhevsky et al. NIPS 2012]



- **Architecture:**

CONV1: 96 11×11 filters at stride 4, pad 0

MAX POOL1 : 3×3 filters at stride 2

NORM1 : Normalization layer

CONV2 : 256 5×5 filters at stride 1, pad 2

MAX POOL2 : 3×3 filters at stride 2

NORM2 : Normalization layer

CONV3 : 384 3×3 filters at stride 1, pad 1

CONV4 : 384 3×3 filters at stride 1, pad 1

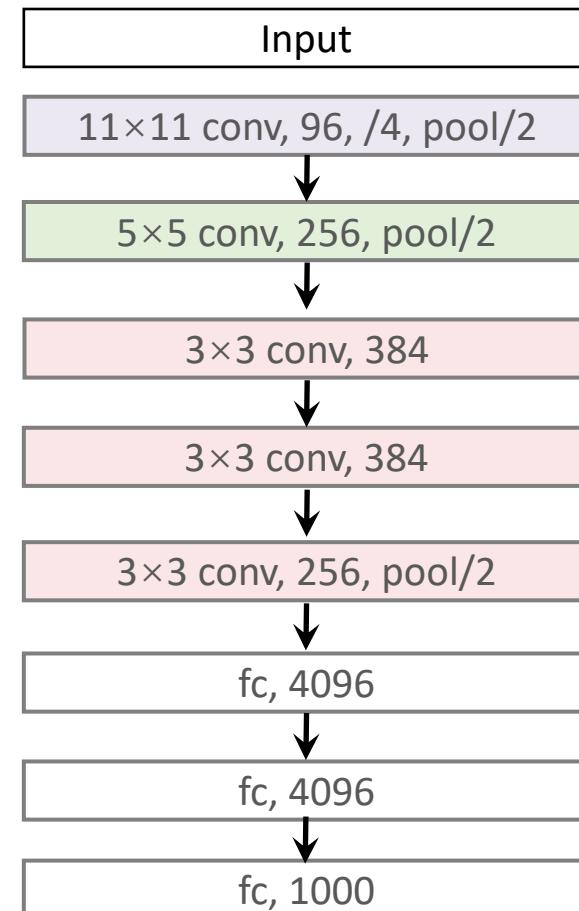
CONV5 : 256 3×3 filters at stride 1, pad 1

MAX POOL3 : 3×3 filters at stride 2

FC6 : 4096 neurons

FC7 : 4096 neurons

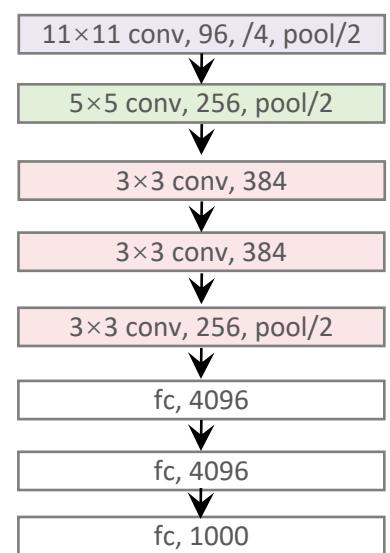
FC8 : 1000 neurons (class scores)



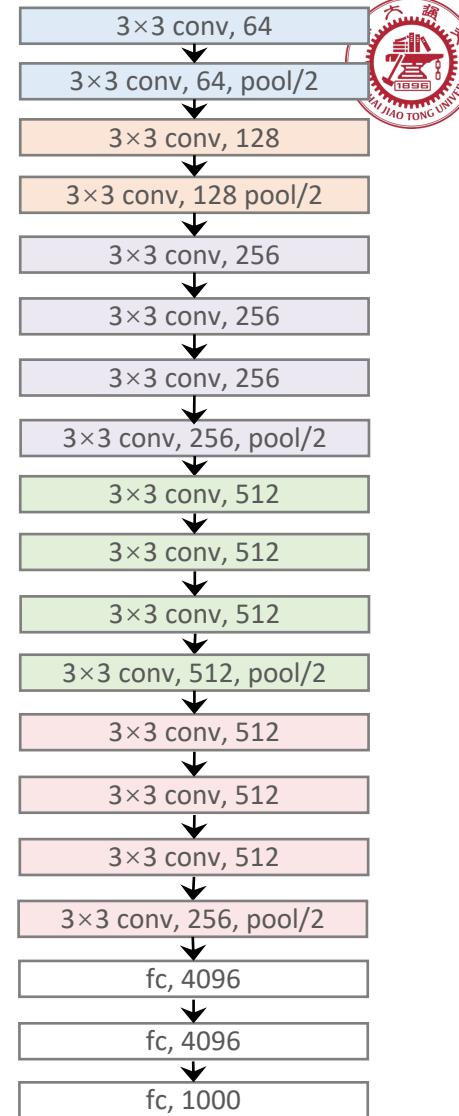
Krizhevsky et al: ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

VGGNet [Simonyan & Zisserman, 2015]

- Small filters:
 - ▷ only 3×3 , stride 1 and 2×2 max pool stride 2
- Deeper networks
 - ▷ $8 \rightarrow 16/19$ layers



AlexNet

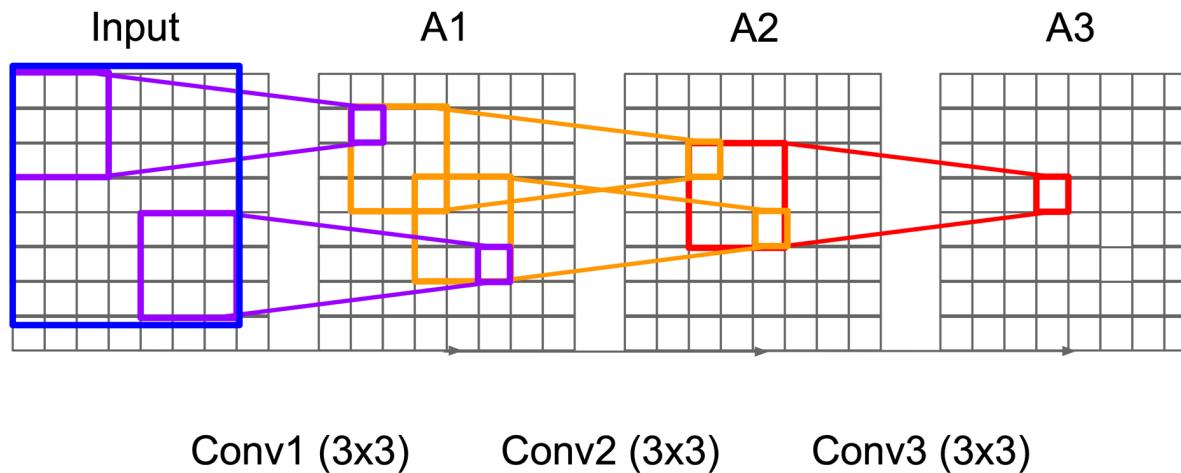


VGG19



- Why use smaller filters? (3×3 conv)

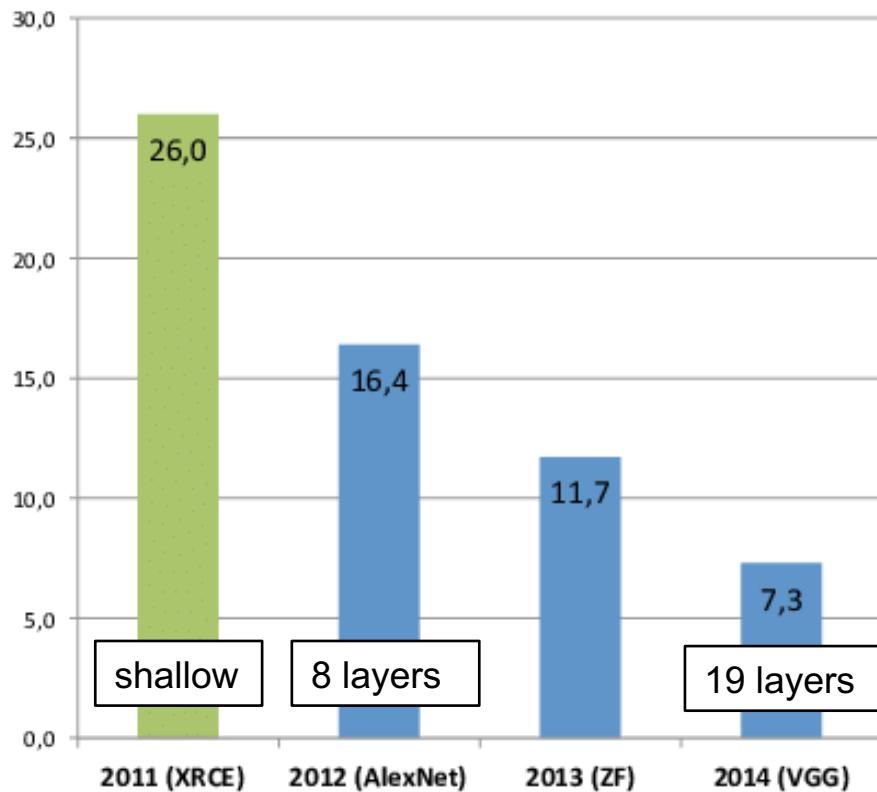
- ▷ Stack of three 3×3 conv (stride 1) layers has same effective receptive field as one 7×7 conv layer.
- ▷ But deeper, more non-linearities
- ▷ And fewer parameters: $3 \times (3^2 C^2)$ vs. $7^2 C^2$ for C channels per layer



VGGNet



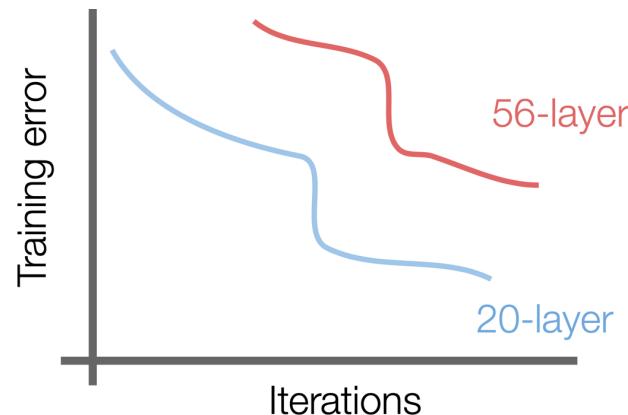
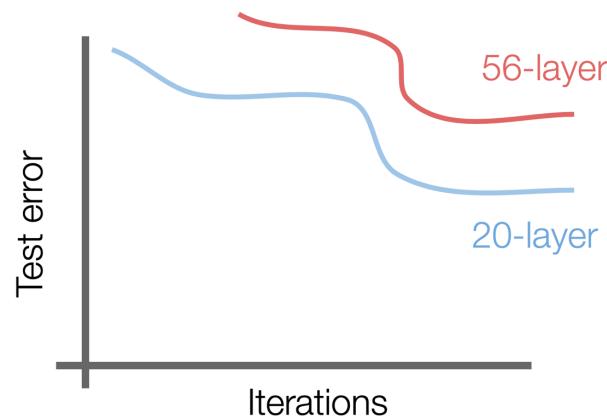
ImageNet Classification Error (Top 5)



The deeper, the better?



- What happens when we continue stacking deeper layers on a “plain” convolutional neural network?



The deeper model performs worse, but it's **not caused by overfitting!**

Hypothesis: the problem is an *optimization* problem, **deeper models are harder to optimize**

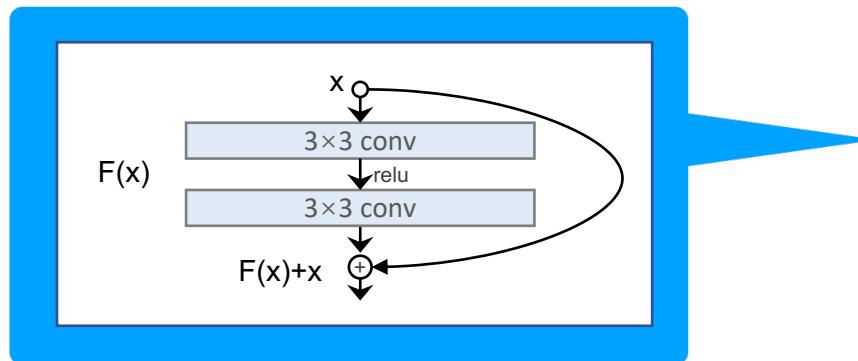
ResNet

[He et al. 2015]



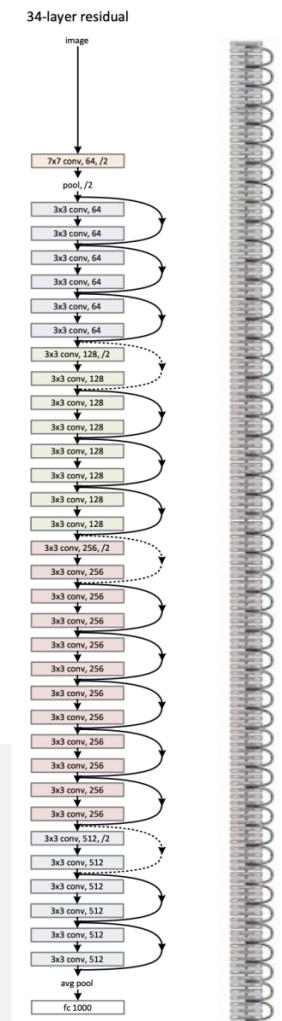
- Very deep CNNs using residual connections

- Stack of many **residual blocks**
- Every residual block has two 3x3 conv layers
- ...



Why use residual connection?

1. gradients can propagate faster through a highway (recall LSTM)
2. within each block, only small residuals have to be learned
3. let deeper model to perform as good as shallower model by copying the learned layers from the shallower model and setting additional layers to identity mapping.

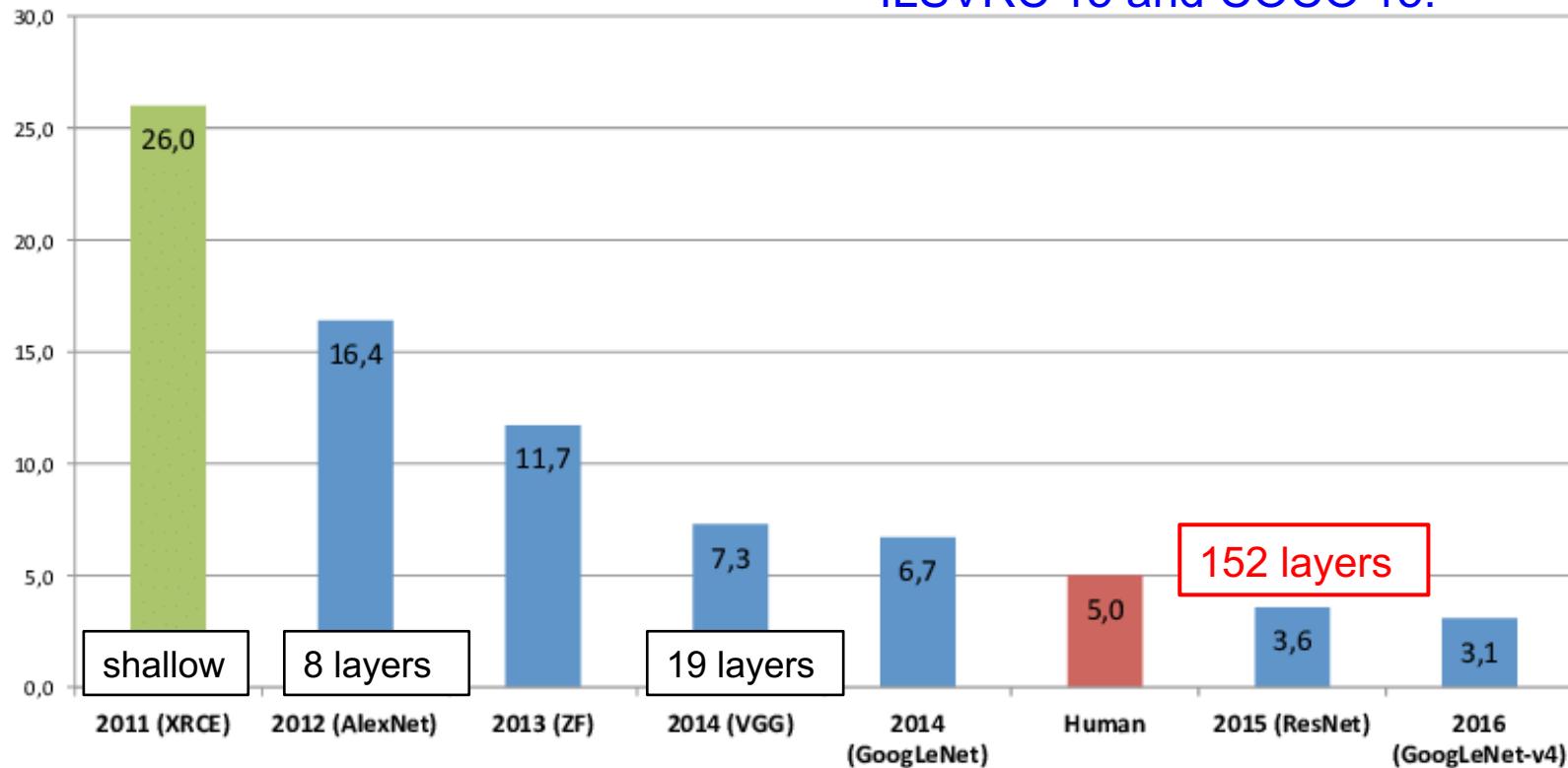


ResNet



- “The Revolution of Depth”

ImageNet Classification Error (Top 5)

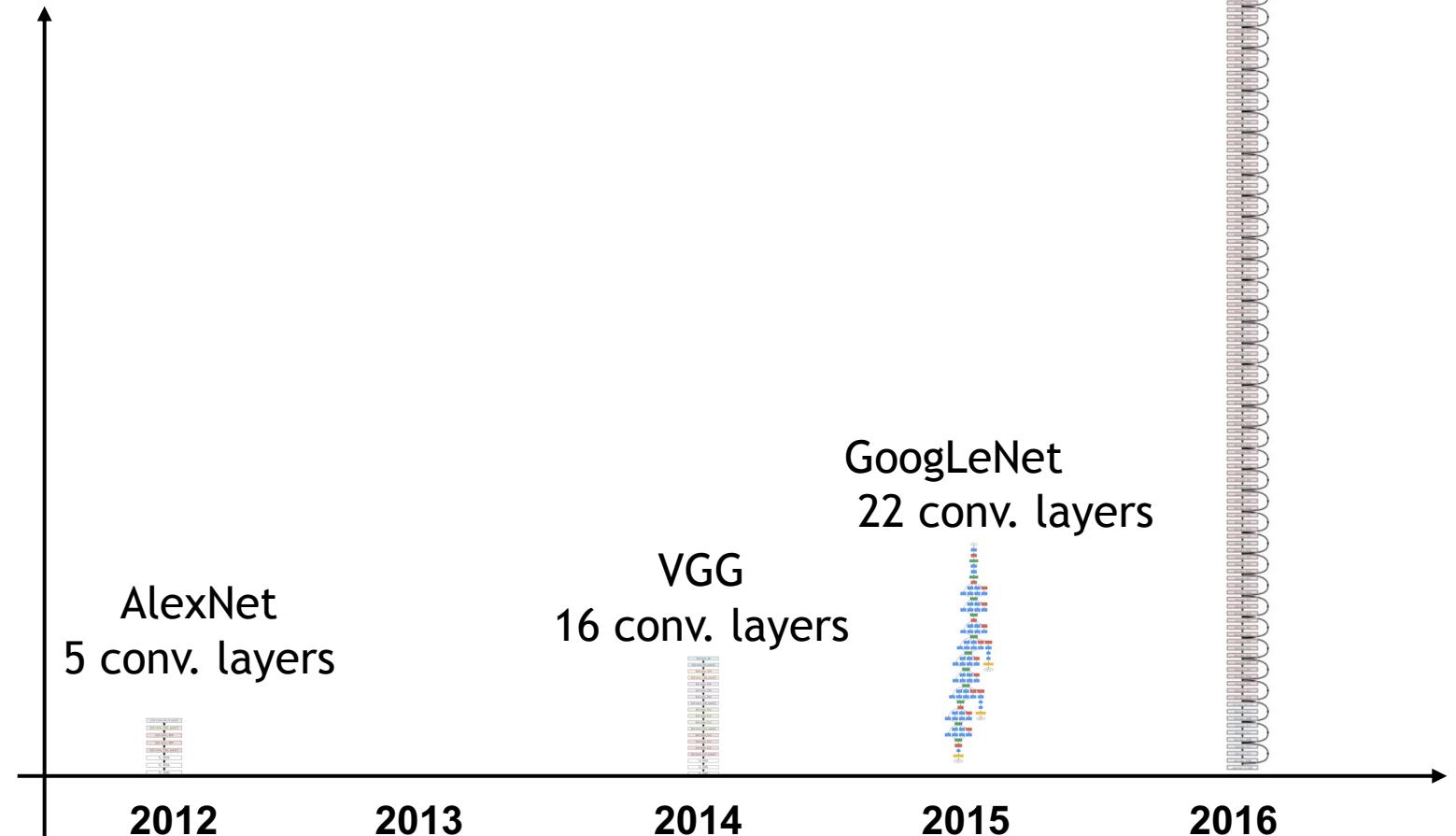


He et al. Deep Residual Learning for Image Recognition. CVPR 2016.

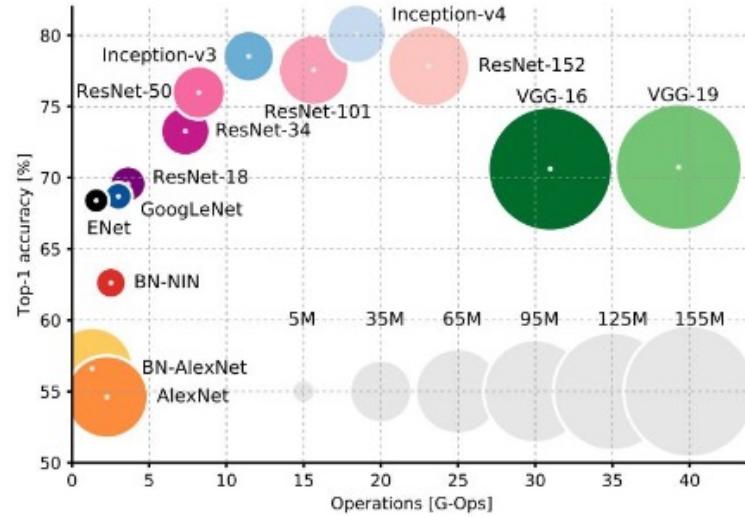
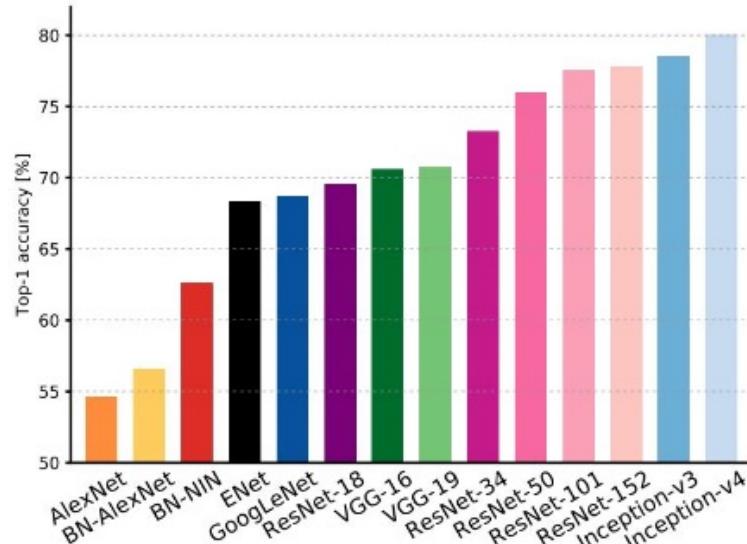
Make Them Bigger



ResNet
>100 conv. layers



Comparing Complexity..



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

TIME for Coding



Tutorial: MNIST image recognition using PyTorch

- <https://www.kaggle.com/code/turksoyomer/pytorch-tutorial-on-mnist-dataset>



What's Next?



The World of Computer Vision

Classification



cat
dog

Semantic Segmentation



Object Detection

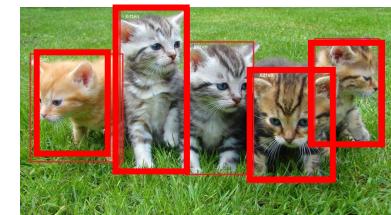


Image Captioning



A cat standing
on the grass

Image Generation

