

Introduction to Deep Learning

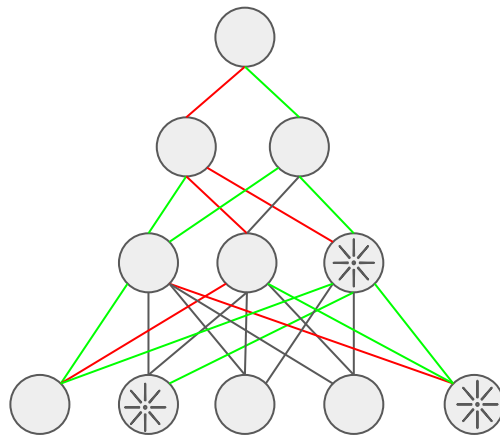
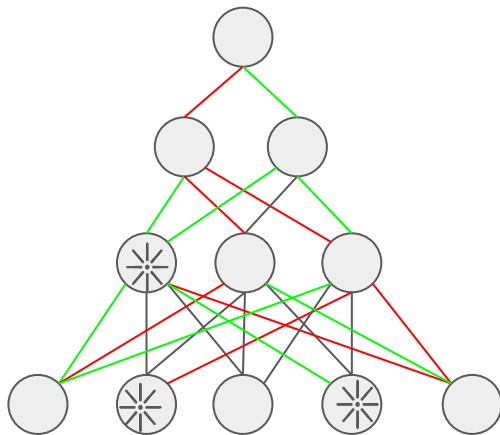
Fall 2018

Convolutional Neural Networks

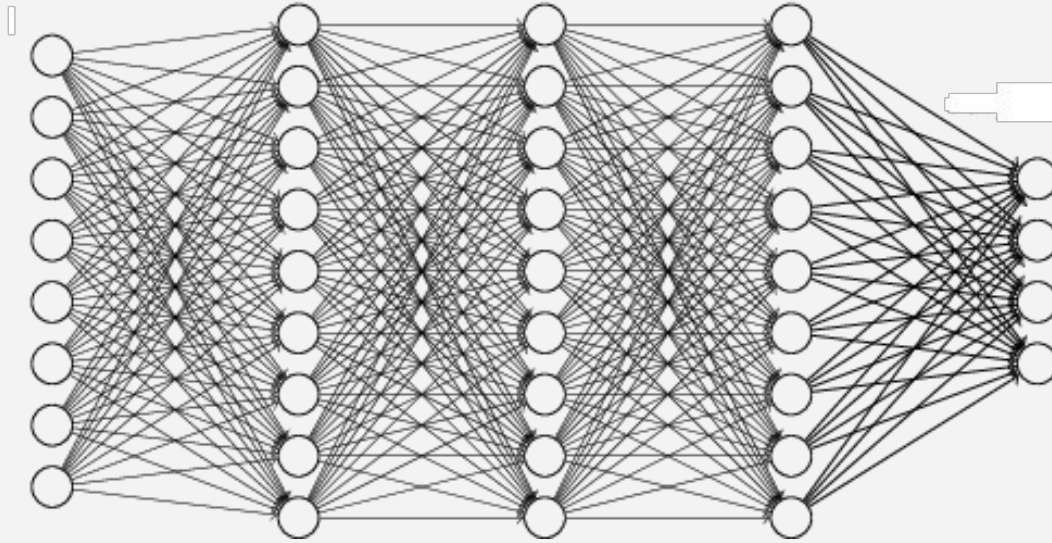


Feature Extraction in the Hidden Layers

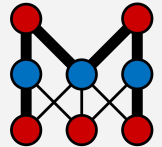
- The goal of each layer of an ANNs is to represent features/patterns present in the previous layer
- These are the *dimensions* of that layer
- They are *learned*



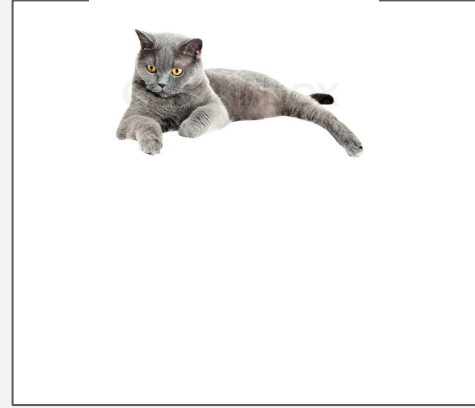
Fully connected multi-layer network



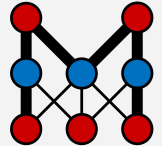
Large number of parameters
Computationally expensive to train



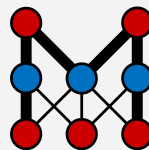
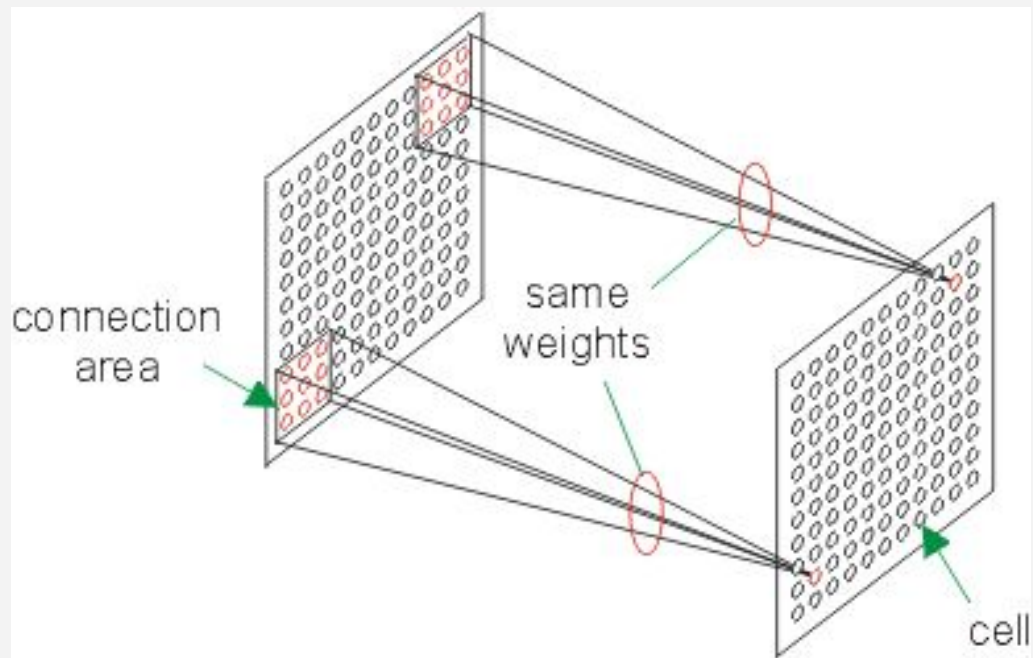
Structure in images is (often) local and repeated



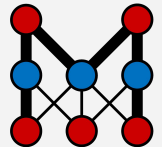
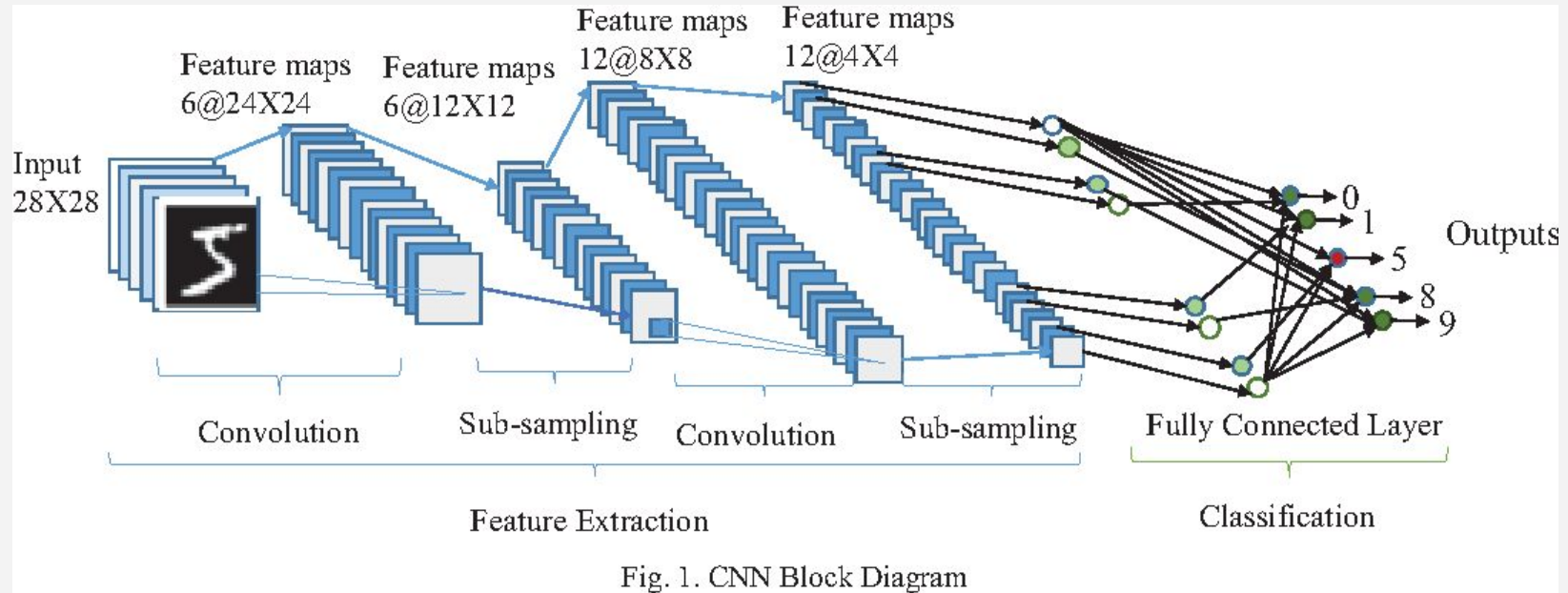
Fully connected network would have to learn each of these independently



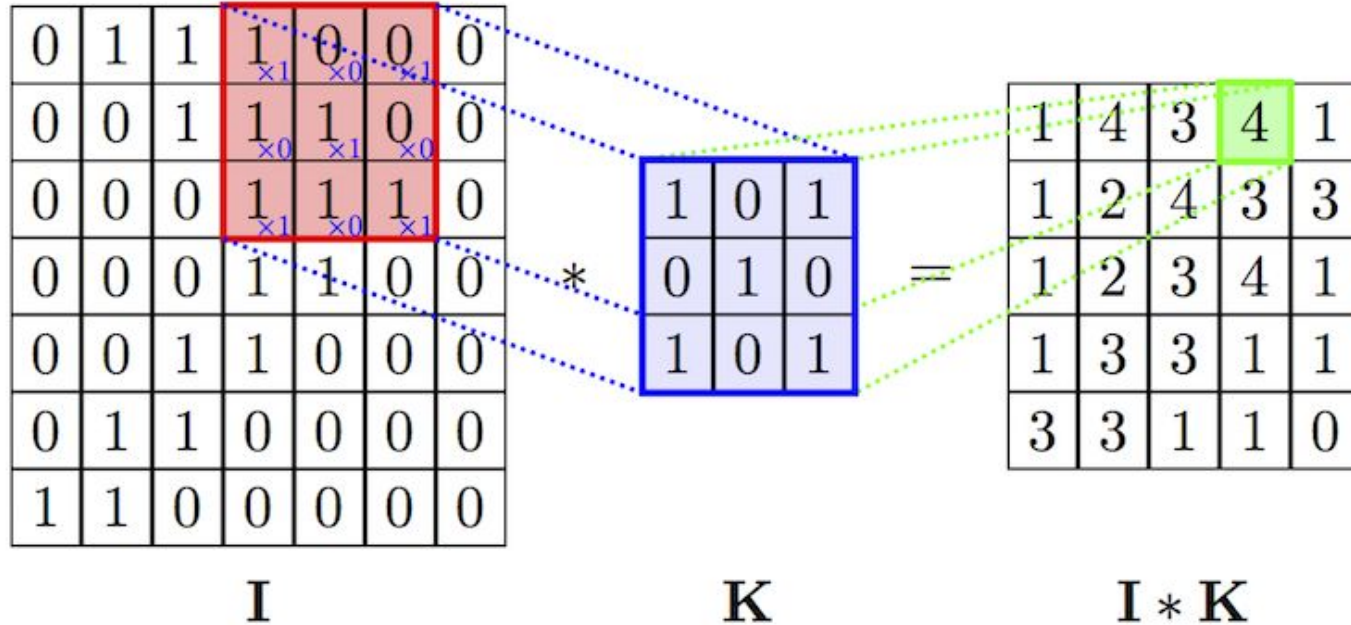
Weight Sharing



Convolutional Neural Networks



Convolutions: Multiplication and Summation



7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

*

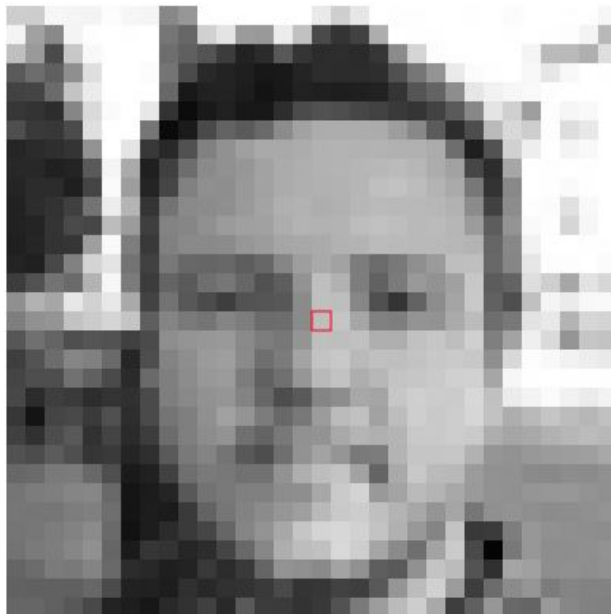
1	0	-1
1	0	-1
1	0	-1

=

6		

$$\begin{aligned}
 &7 \times 1 + 4 \times 1 + 3 \times 1 + \\
 &2 \times 0 + 5 \times 0 + 3 \times 0 + \\
 &3 \times -1 + 3 \times -1 + 2 \times -1 \\
 &= 6
 \end{aligned}$$

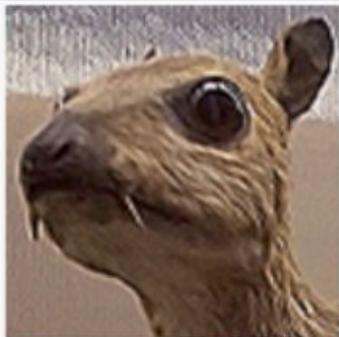
206 205 247 245 244 253 247 245 136 131 255 255 255 255 255 254 207 231 255 254 254 255 255 254 255 252 255 255 254 255 247
 244 161 137 244 254 255 254 255 118 103 209 228 155 153 236 193 74 52 65 173 255 254 254 255 255 255 254 255 254 184
 180 154 75 200 249 255 255 255 110 96 84 61 35 44 89 53 44 45 43 54 140 213 253 255 255 255 255 245 187 186 176 223
 90 109 96 143 223 255 255 255 252 117 75 41 35 31 24 25 36 45 44 44 46 81 118 148 254 252 254 255 248 231 248 255 254
 67 89 107 156 236 255 255 255 104 20 34 35 39 20 20 34 32 30 32 34 53 85 100 142 231 242 247 249 255 255 255 255
 55 51 45 134 218 251 255 232 51 12 35 33 24 24 46 75 82 78 71 69 58 53 67 90 136 238 258 158 253 246 249 255
 79 98 56 75 224 255 255 118 11 27 74 99 91 106 140 162 173 173 173 158 137 92 46 78 187 217 206 254 232 233 255
 38 43 47 52 147 255 229 56 41 81 129 145 180 169 169 172 178 179 178 179 177 177 172 110 31 82 209 238 255 244 249 255
 40 40 33 36 90 245 171 32 65 110 139 145 151 162 171 174 178 179 182 184 187 183 173 162 71 45 167 255 254 255 254 255
 37 44 44 31 69 250 158 36 70 129 143 142 153 162 171 175 177 178 182 191 194 188 180 170 120 51 137 255 254 250 254 255
 34 45 51 64 116 237 181 53 116 138 140 143 124 164 176 178 174 177 183 185 185 185 183 178 149 68 141 254 252 225 249 255
 34 36 52 74 71 188 156 63 131 134 144 155 180 161 173 179 178 179 189 193 190 185 187 182 156 93 148 250 254 214 247 255
 32 38 52 54 159 250 126 57 129 138 138 140 151 156 188 188 171 178 180 187 186 185 185 183 180 152 136 242 255 255 254 254
 36 32 72 129 212 228 115 65 121 104 102 104 94 103 134 158 170 162 125 108 121 143 155 190 191 104 134 230 253 253 255 251
 61 82 116 107 179 247 124 60 101 90 111 119 103 81 94 147 191 178 126 98 123 153 147 161 200 92 100 222 207 167 227 215
 144 178 167 231 210 232 170 67 115 88 76 62 83 85 88 139 152 190 135 88 53 59 141 165 201 97 79 192 245 235 248 248
 127 145 149 185 204 213 197 95 133 122 117 133 126 108 110 137 191 197 167 129 127 148 147 171 188 110 121 228 233 180 215 212
 67 112 100 79 85 62 65 75 142 148 151 153 138 125 120 149 191 190 193 175 174 193 198 190 258 127 183 239 219 149 158 195
 83 108 108 134 129 106 39 78 132 142 135 159 139 111 124 164 195 200 186 192 191 195 200 202 200 143 217 253 249 242 238 234
 69 78 78 113 97 74 43 106 127 140 152 155 125 97 112 150 185 194 174 183 196 198 202 258 209 186 247 254 255 254 254 254
 72 44 63 59 48 52 49 74 127 137 146 149 132 103 78 93 134 141 168 165 199 207 204 203 215 193 236 244 251 242 238 243
 55 20 69 73 59 88 46 74 117 127 144 161 148 124 105 120 156 187 193 162 189 206 201 255 214 194 174 185 197 188 163 193
 65 49 77 89 50 68 43 61 109 127 141 147 113 100 121 145 148 189 181 176 181 201 201 205 202 174 188 189 178 183 188 184
 82 76 92 79 54 58 37 47 90 121 132 156 88 78 111 146 163 149 122 124 180 197 197 198 178 149 146 152 152 157 159 168
 104 107 122 123 105 79 27 33 66 111 122 120 114 114 147 175 190 196 163 101 179 200 187 185 156 146 145 139 137 141 140 145
 117 124 127 133 135 105 21 28 37 88 115 121 126 128 141 142 168 202 212 153 154 186 180 168 154 146 144 149 151 151 147 144
 119 118 118 125 128 111 21 29 38 58 100 118 131 140 151 159 186 201 205 192 180 168 149 166 119 144 147 143 140 141 144 148
 117 119 125 130 139 106 18 29 44 58 70 102 133 147 168 197 212 215 210 195 177 132 133 195 57 59 126 151 145 143 142 141
 113 125 126 134 145 102 27 54 52 38 46 89 105 135 175 189 193 216 206 186 139 111 164 203 74 5 121 151 142 142 143 146
 101 108 123 121 132 105 44 40 31 29 57 44 58 101 147 144 138 163 145 94 90 145 196 167 84 48 165 160 142 144 143 145
 98 97 97 96 104 78 34 33 30 48 41 49 51 58 74 53 55 69 63 89 150 188 209 156 62 108 140 149 125 133 131 131
 102 102 97 88 73 35 30 23 42 50 65 41 90 60 59 31 57 82 123 157 167 205 169 62 96 151 105 101 154 135 130 129



<http://setosa.io/ev/image-kernels/>

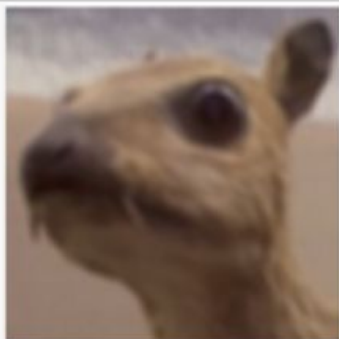
Sharpen

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



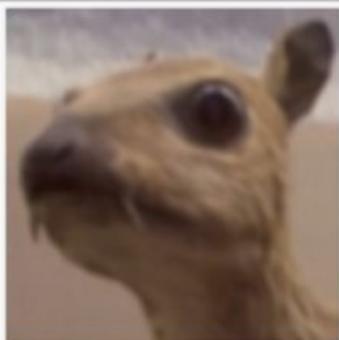
Box blur
(normalized)

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



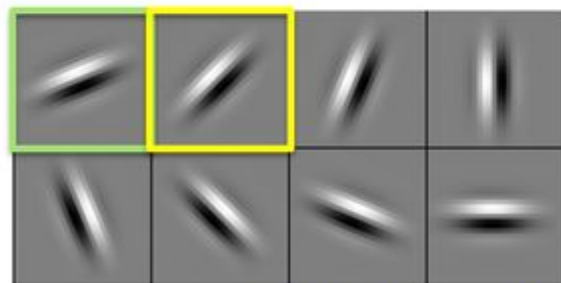
Gaussian blur
(approximation)

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

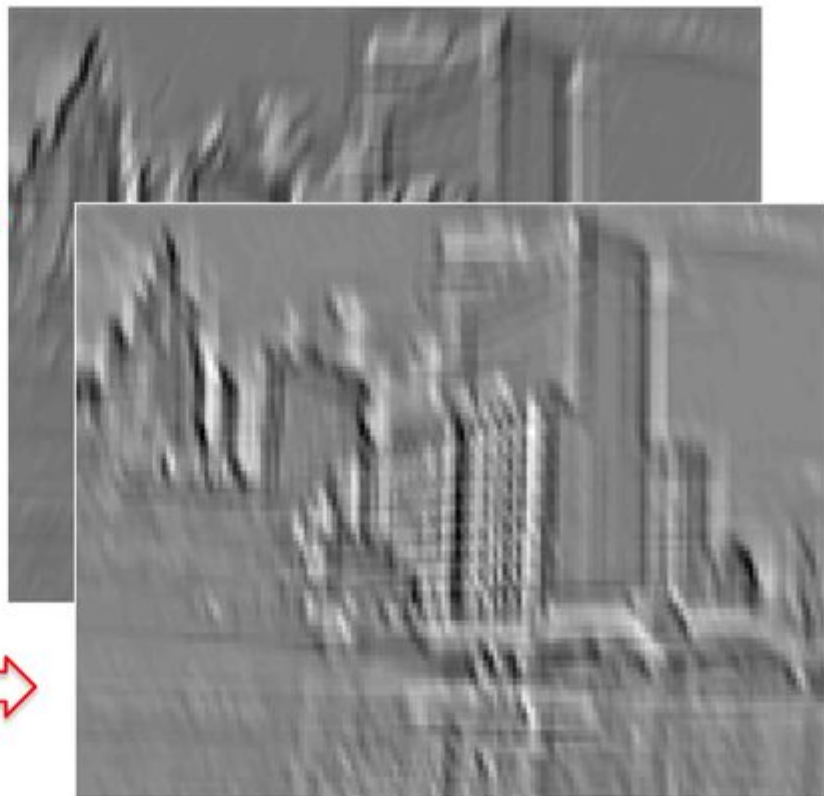
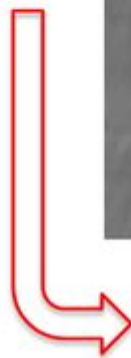




Input image



Filter bank (to be learned)



Feature maps



Input

Image Size: 5 X 5

Padding Size: 1

Kernel Size: 3 X 3

Stride: 2

Feature Map: 3X3



Image Size: 5 X 5

Padding Size: 1

Kernel Size: 3 X 3

Stride: 1

Feature Map: 5 X 5

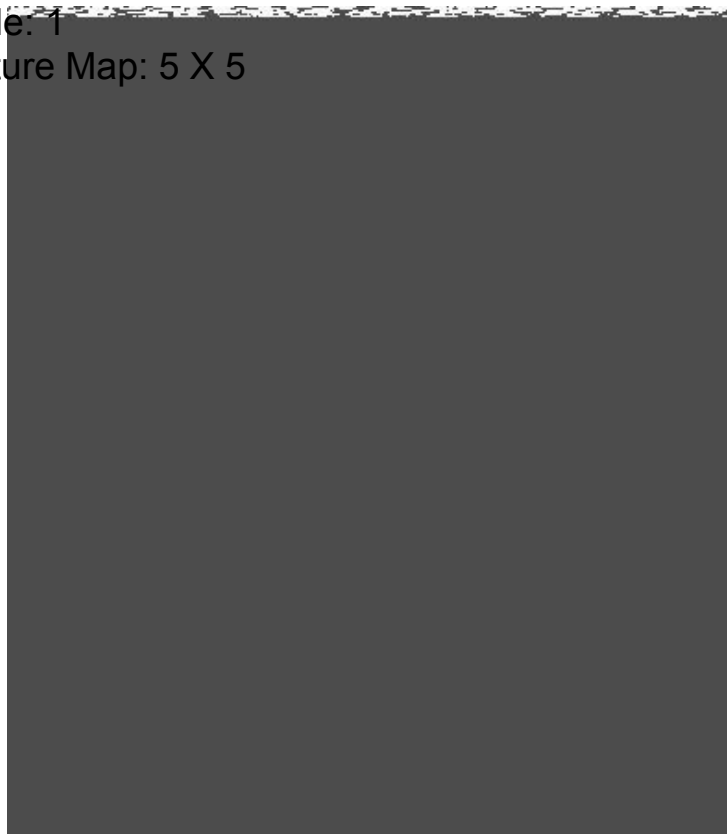


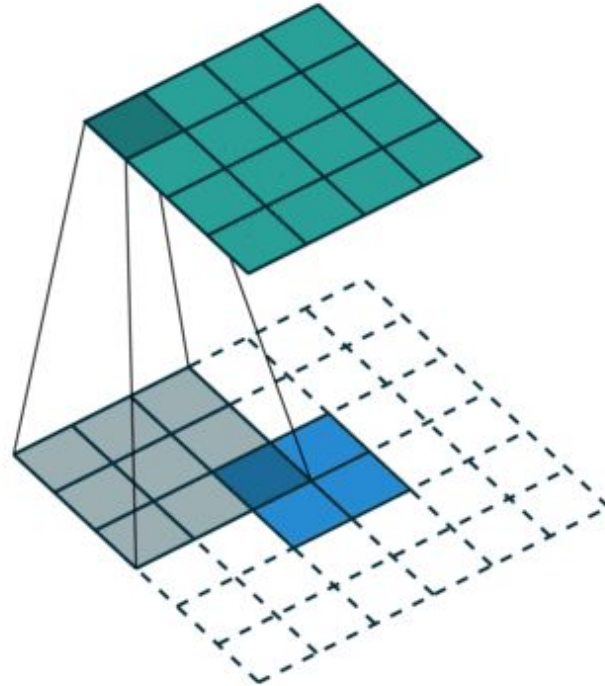
Image Size: 2 X 2

Padding Size: 2

Kernel Size: 3 X 3

Stride: 2

Feature Map: 4X4



0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix		
0	-1	0
-1	5	-1
0	-1	0

320				

Output Matrix

$$\begin{aligned}
 &0 * 0 + 0 * -1 + 0 * 0 \\
 &+ 0 * -1 + 105 * 5 + 102 * -1 \\
 &+ 0 * 0 + 103 * -1 + 99 * 0 = 320
 \end{aligned}$$

**Convolution with horizontal and
vertical strides = 1**

3d (RGB) Input \longrightarrow 1D output

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



308

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-498

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



164

+

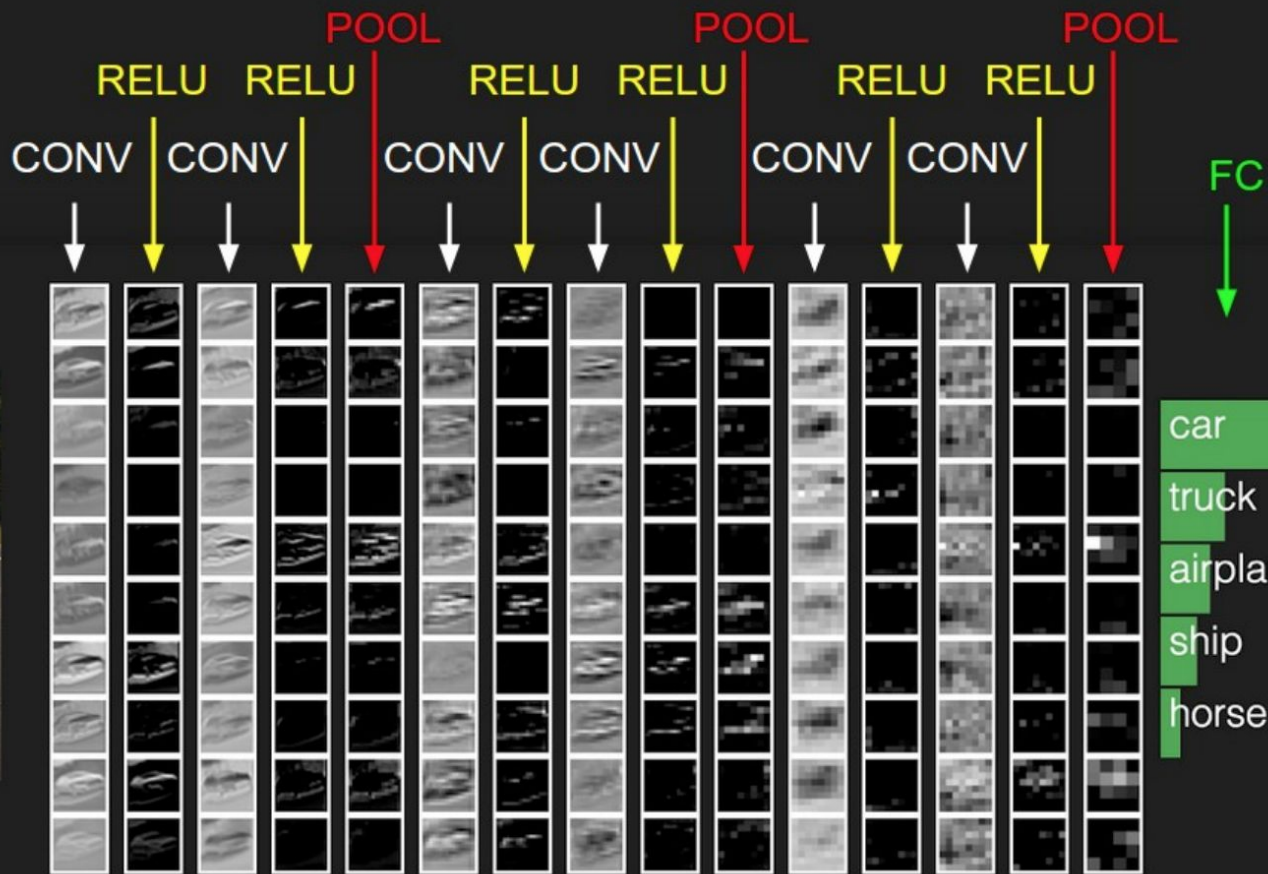
+

Bias = 1

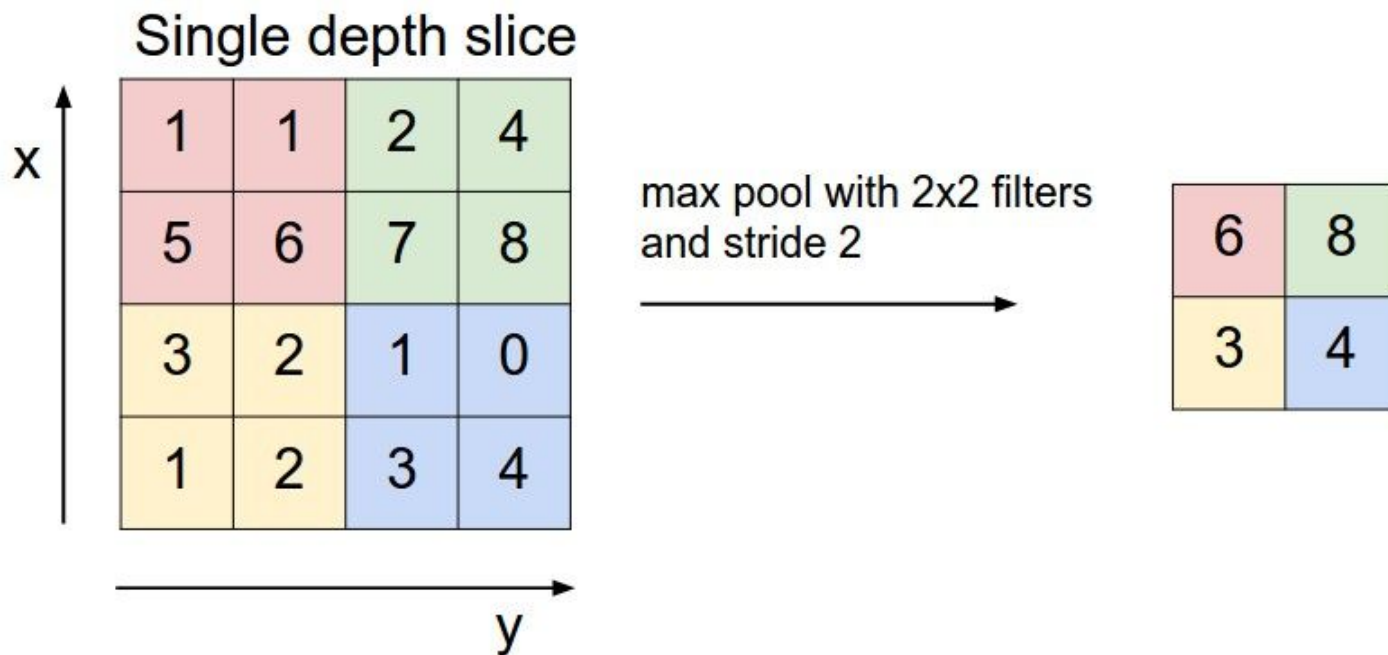
+ 1 = -25

Output

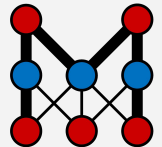
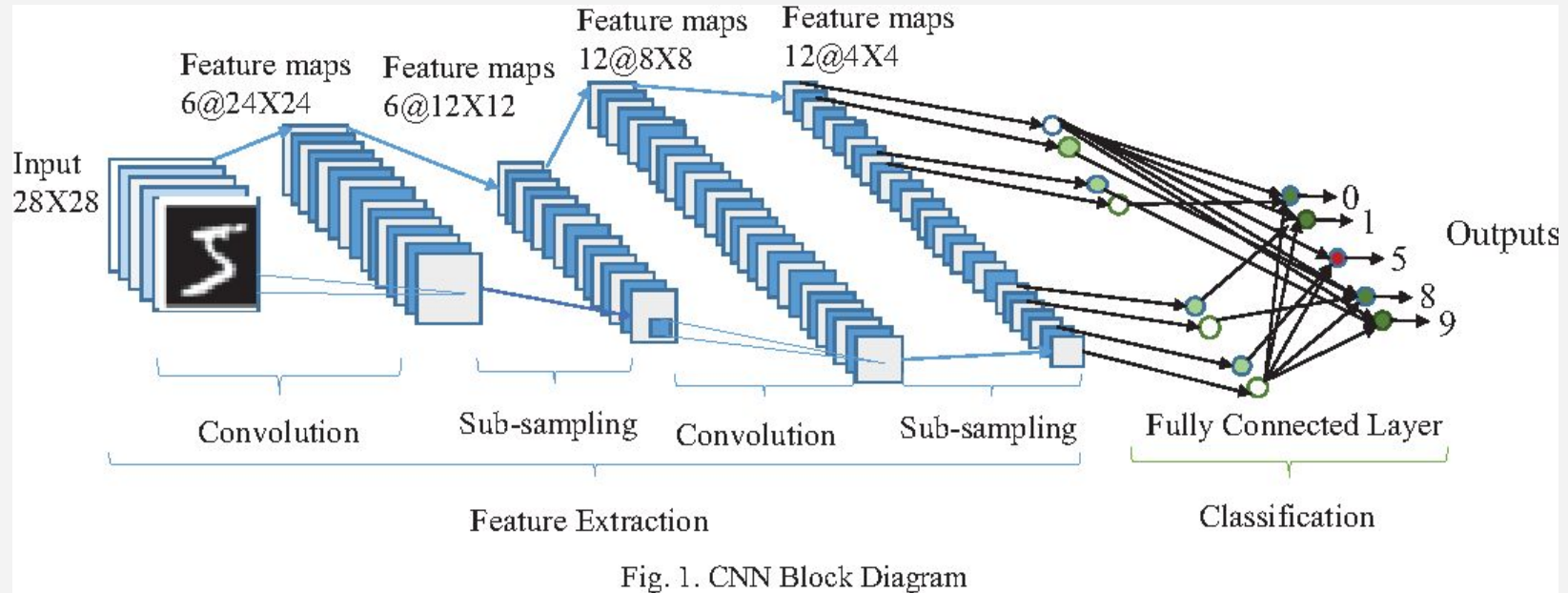
-25				...
				...
				...
				...
...

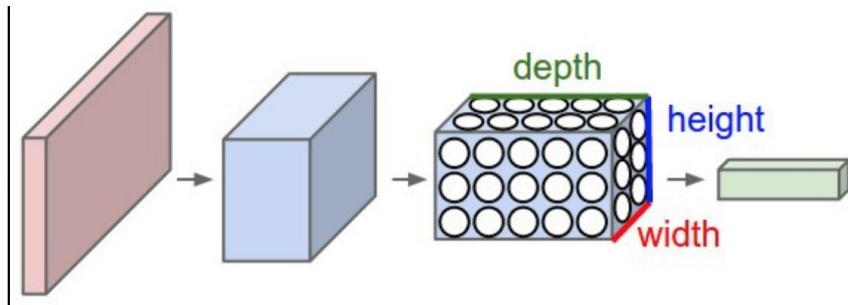
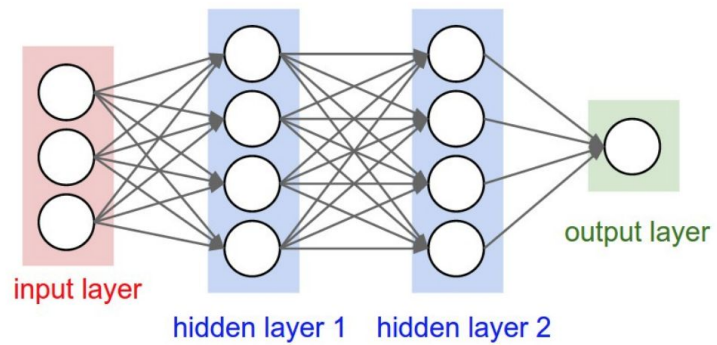


Pooling to reduce dimensionality



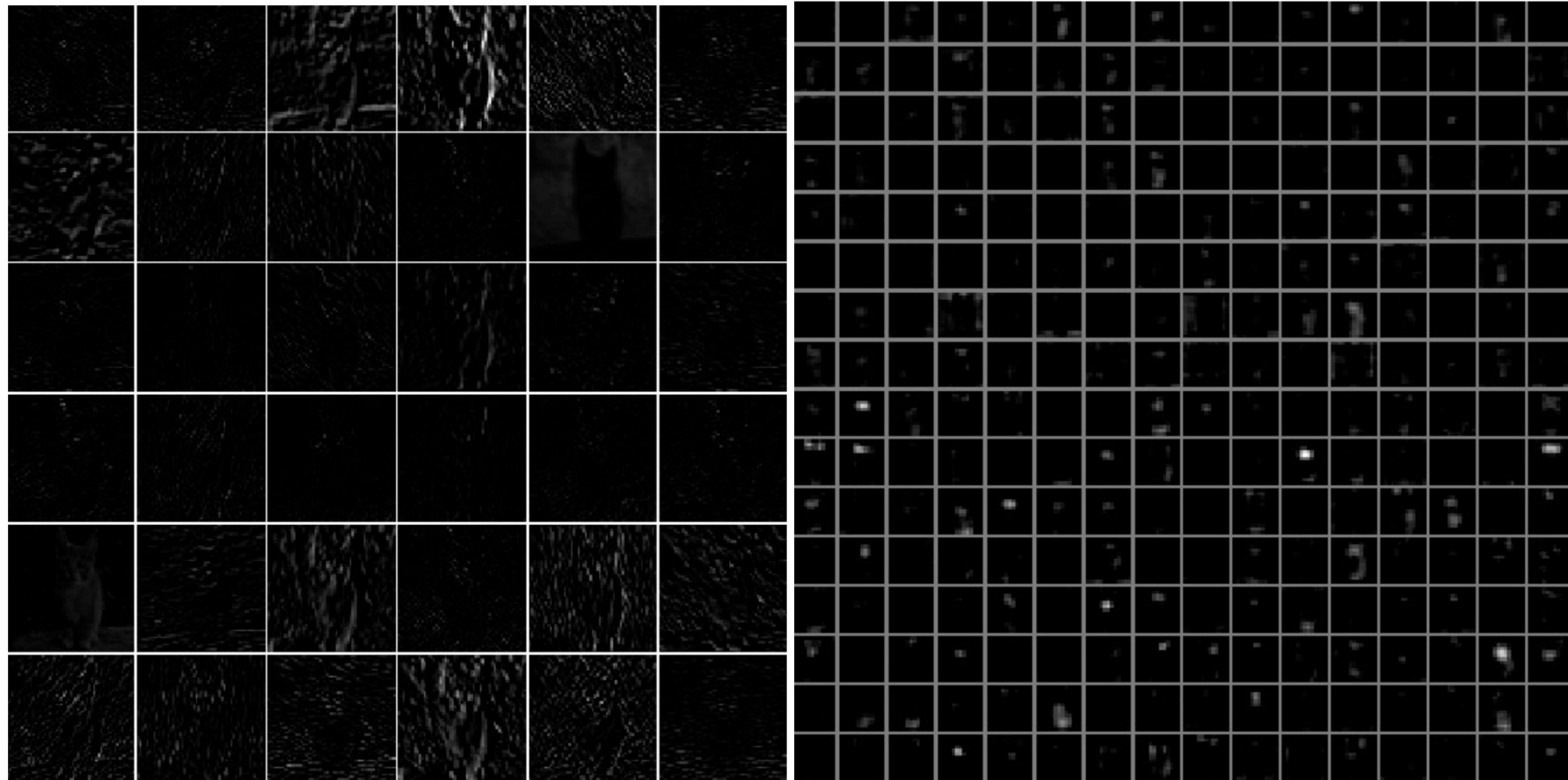
Convolutional Neural Networks











Typical-looking activations on the first CONV layer (left), and the 5th CONV layer (right) of a trained AlexNet looking at a picture of a cat. Every box shows an activation map corresponding to some filter. Notice that the activations are sparse (most values are zero, in this visualization shown in black) and mostly local.