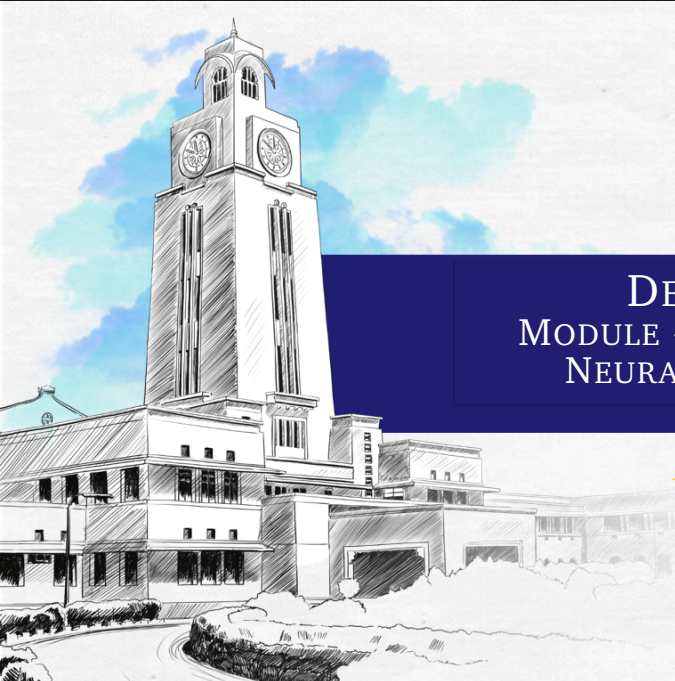


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Deep Learning
Module # 5 : Convolutional Neural Network [CNN]

Seetha Parameswaran
Asst Prof, BITS Pilani

The instructor is gratefully acknowledging the authors who made their course materials freely available online.

In This Segment

Image Classification

64x64 x3

Cat? (0/1)

Cat? (0/1)

Classify into 3 classes

Face Detection

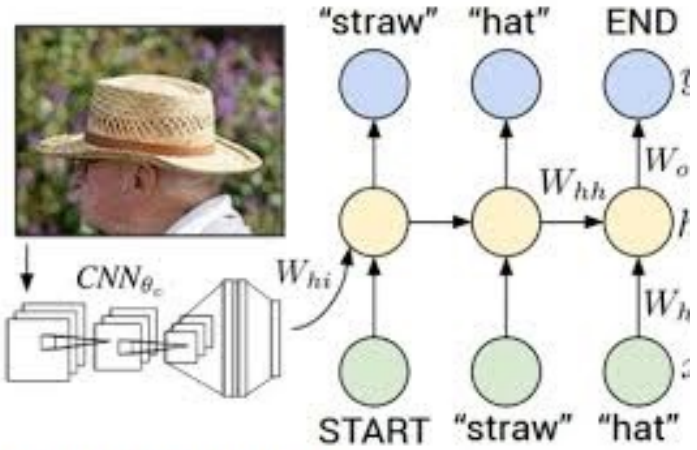
Object Recognition

"SIFT" & Object Recognition, David Lowe, 1999

Image is public domain

Image is CC BY-SA 2.0

Image Captioning

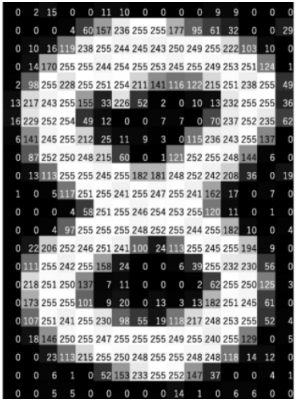


Andrej Karpathy, Fei-Fei Li

In This Segment

Digital Grayscale Image

Pixel intensities = $I(x, y)$



```
  0   2  15   0   0  11  10   0   0   0   0   9   9   0   0   0
  0   0   0   4  60 157 236 255 255 177  95  61  32   0   0  29
  0  10  16 119 238 255 244 245 243 250 249 255 222 103  10   0
  0  14 170 255 255 244 254 255 253 245 255 249 253 251 124   1
  2  98 255 228 255 251 254 211 141 116 122 215 251 238 255  49
 13 217 243 255 155  33 226  52   2   0  10  13 232 255 255  36
 16 229 252 254  49  12   0   0   7   7   0  70 237 252 235  62
  6 141 245 255 212  25  11   9   3   0 115 236 243 255 137   0
  0  87 252 250 248 215  60   0   1 121 252 255 248 144   6   0
  0  13 115 255 255 245 255 182 181 248 252 242 208  36   0  19
  1   0   5 117 251 255 241 255 247 255 241 162  17   0   7   0
  0   0   0   4  58 251 255 246 254 253 255 120  11   0   1   0
  0   0   4  97 255 255 255 248 252 255 244 255 182  10   0   4
  0  22 206 252 246 251 241 100  24 113 255 245 255 194   9   0
  0 111 255 242 255 158  24   0   0   6  39 255 232 230  56   0
  0 218 251 250 137   7  11   0   0   0   2  62 255 250 125   3
  0 173 255 255 101   9  20   0  13   3  13 182 251 245  61   0
  0 107 251 241 255 230  98  55  19 118 217 248 253 255  52   4
  0  18 146 250 255 247 255 255 255 249 255 240 255 129   0   5
  0   0  23 113 215 255 250 248 255 255 248 248 118  14  12   0
  0   0   6   1   0  52 153 233 255 252 147  37   0   0   4   1
  0   0   5   5   0   0   0   0   0  14   1   0   6   6   0   0
```

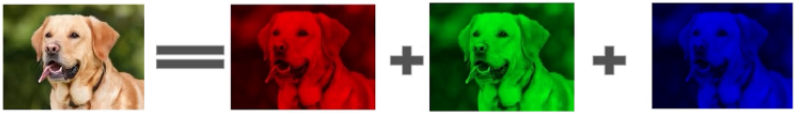

```
  0   2  15   0   0  11  10   0   0   0   0   9   9   0   0   0
  0   0   0   4  60 157 236 255 255 177  95  61  32   0   0  29
  0  10  16 119 238 255 244 245 243 250 249 255 222 103  10   0
  0  14 170 255 255 244 254 255 253 245 255 249 253 251 124   1
  2  98 255 228 255 251 254 211 141 116 122 215 251 238 255  49
 13 217 243 255 155  33 226  52   2   0  10  13 232 255 255  36
 16 229 252 254  49  12   0   0   7   7   0  70 237 252 235  62
  6 141 245 255 212  25  11   9   3   0 115 236 243 255 137   0
  0  87 252 250 248 215  60   0   1 121 252 255 248 144   6   0
  0  13 115 255 255 245 255 182 181 248 252 242 208  36   0  19
  1   0   5 117 251 255 241 255 247 255 241 162  17   0   7   0
  0   0   0   4  58 251 255 246 254 253 255 120  11   0   1   0
  0   0   4  97 255 255 255 248 252 255 244 255 182  10   0   4
  0  22 206 252 246 251 241 100  24 113 255 245 255 194   9   0
  0 111 255 242 255 158  24   0   0   6  39 255 232 230  56   0
  0 218 251 250 137   7  11   0   0   0   2  62 255 250 125   3
  0 173 255 255 101   9  20   0  13   3  13 182 251 245  61   0
  0 107 251 241 255 230  98  55  19 118 217 248 253 255  52   4
  0  18 146 250 255 247 255 255 255 249 255 240 255 129   0   5
  0   0  23 113 215 255 250 248 255 255 248 248 118  14  12   0
  0   0   6   1   0  52 153 233 255 252 147  37   0   0   4   1
  0   0   5   5   0   0   0   0   0  14   1   0   6   6   0   0
```


https://mozanunal.com/2019/11/img2sh/

Digital Color Image

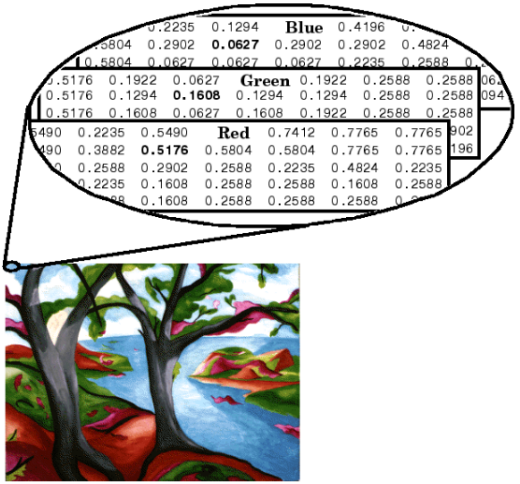
Colour Image

This image is composed of many colors and almost all colors can be generated from the three primary colors- **Red, Green, and Blue.** We can say that each colored image is composed of these three colors or 3 channels- Red, Green, and Blue-



https://www.analyticsvidhya.com/blog/2021/03/grayscale-and-rgb-format-for-storing-images/

Digital Color Image



https://www.mathworks.com/help/matlab/creating_plots/image-types.html

In This Segment

Image Classification - Challenges

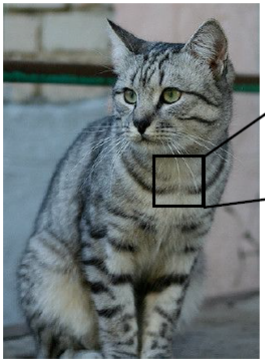
Image Classification: A core task in Computer Vision

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

Image Classification - Challenges

The Problem: Semantic Gap




```
[[105 112 108 111 104  93 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 118 105  94  85]
 [ 76  85  96 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  92 101  94]
 [105  91  61  64  65  91  88  85 101 107 105  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  88  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  68  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]
 [115 114 109 123 150 148 131 118 113 109 100  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  80  78  71  88 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 130 135 105  81  98 110 110]
 [ 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  55  95  93  89  95 102 107]
 [104 146 112  80  82 120 124 104  76  48  45  66  88 101 102 100]
 [157 170 157 120  93  86 114 132 112  97  69  55  78  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 128 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  88 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  76  92 107]
 [122 164 148 103  71  56  78  83  93 103 110 130 102  61  69  84]]
```

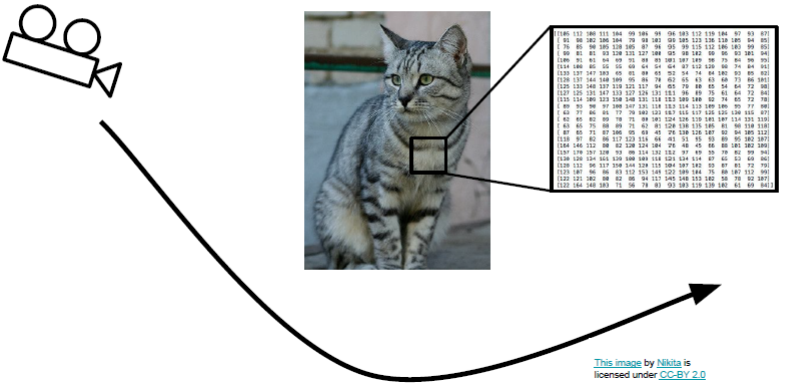
What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Image Classification - Challenges

Challenges: Viewpoint variation

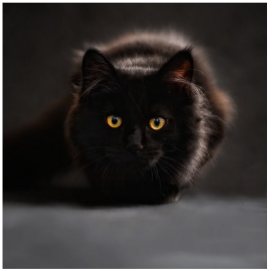


All pixels change when the camera moves!

Image Classification - Challenges

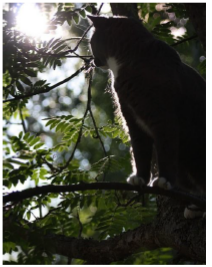
Challenges: Illumination

This image is CC0 1.0 public domain



This image is CC0 1.0 public domain

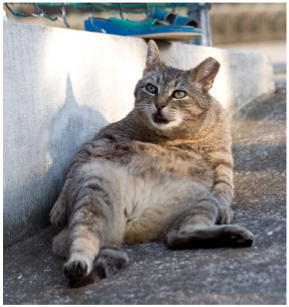
This image is CC0 1.0 public domain



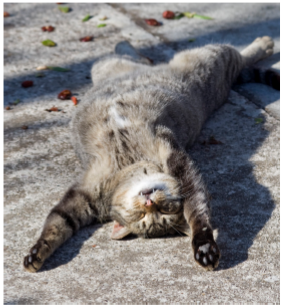
This image is CC0 1.0 public domain

Image Classification - Challenges

Challenges: Deformation



This image by Umberto Salvagnin is licensed under CC-BY 2.0



This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0

This image by Tom Thai is licensed under CC-BY 2.0

Challenges: Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Image Classification - Challenges

Challenges: Intraclass variation

Image Classification - Challenges

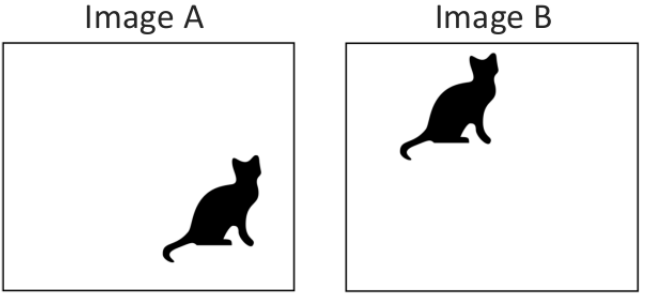
Pixelated Images

Larger Images

- A 50x50 Image requires 2500 input dimensions.
- A HD image 1920x1080 is about 2 million inputs. (2 Mega Pixels)

Image Invariance - Challenges

Object can appear anywhere in the image.

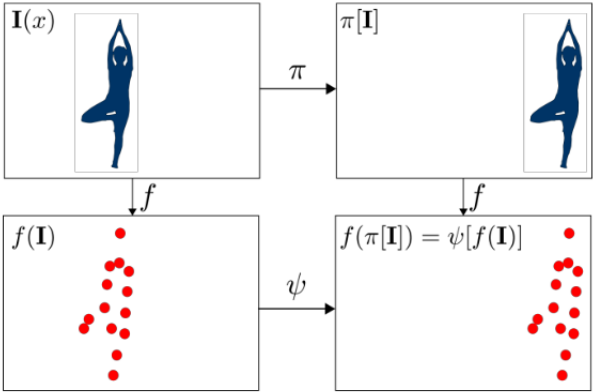


Model predicts “cat” for both images!

Source: https://penkovsky.com/neural-networks/day5/

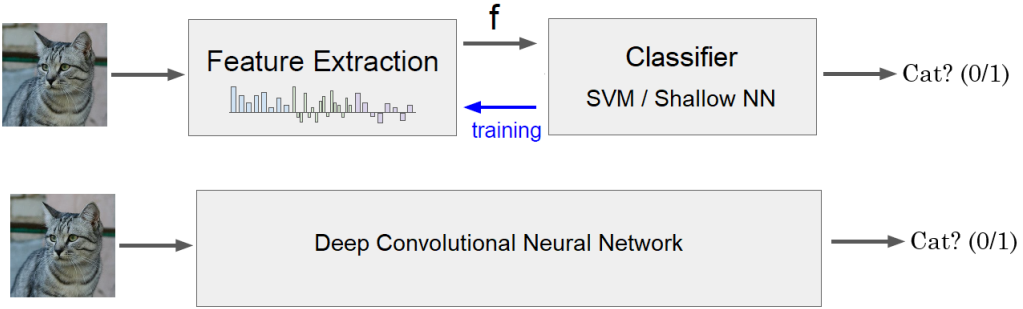
Image Equivariance - Challenges

Changes to the input predictably (equivalently) should not affect the output.



Source: https://towardsdatascience.com/translational-invariance-vs-translational-equivariance-f9fbc8fca63a

Image Classification - Solution



In This Segment

Convolutional Neural Networks (CNN)

- Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
- Special kind of neural network for processing data that has a known, grid-like topology.
- Eg: Time-series data – 1D grid taking samples at regular time intervals.
- Eg: Image data – 2D grid of pixels.
- Network employs a mathematical operation called **convolution**.

Example of CNN – LeNet-5

- Handwritten Character recognition
- 10-categories classification problem

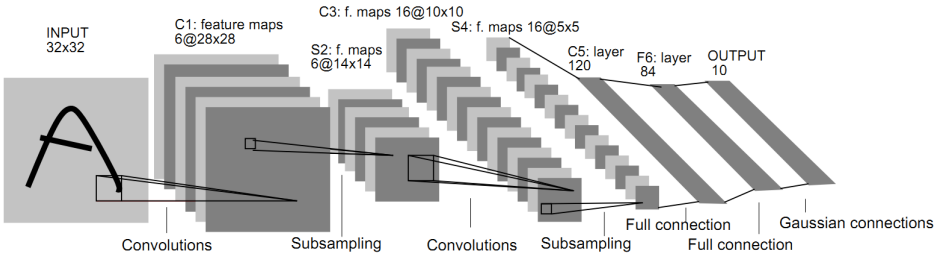


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Courtesy: Yann LeCun

Why CNN for Images

- Some patterns are much smaller than the whole image.

Courtesy: Akka Zemmari

Why CNN for Images

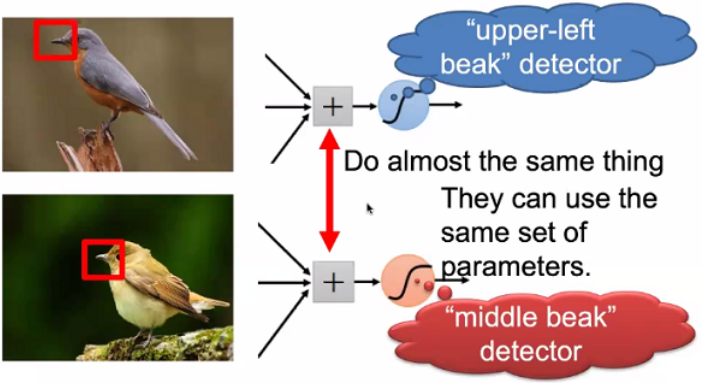
- Some patterns are much smaller than the whole image.

A neuron does not have to see the whole image to discover the pattern.

Courtesy: Akka Zemmari

Why CNN for Images

- Same pattern may appear in different regions.



Courtesy: Akka Zemmari

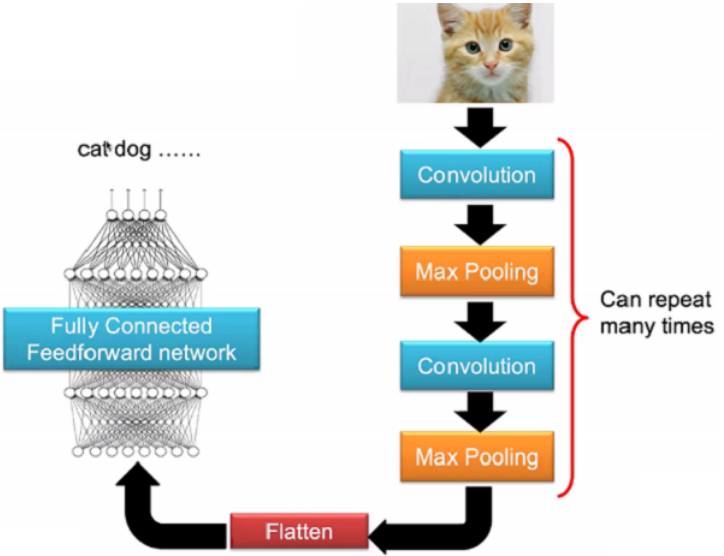
Why CNN for Images

- Sub-sampling the pixels will not change the object.

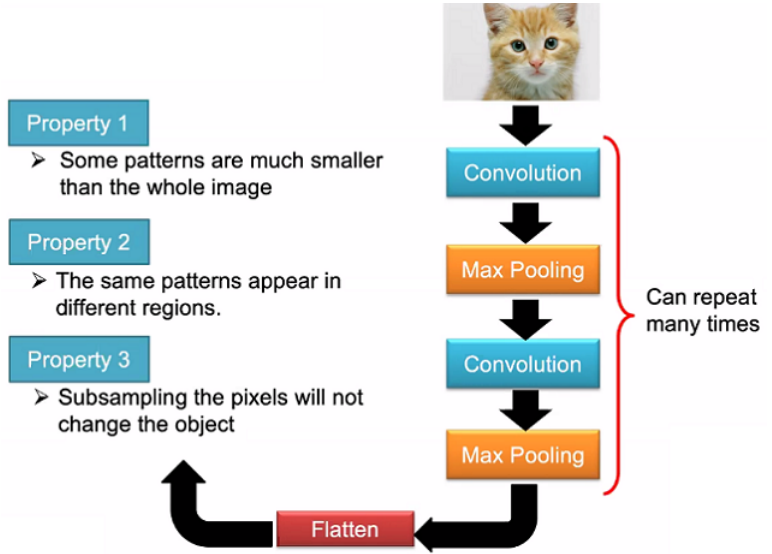
- We can sub-sample the image to make it smaller.
- Less parameters for the network to process the image.

Courtesy: Akka Zemmari

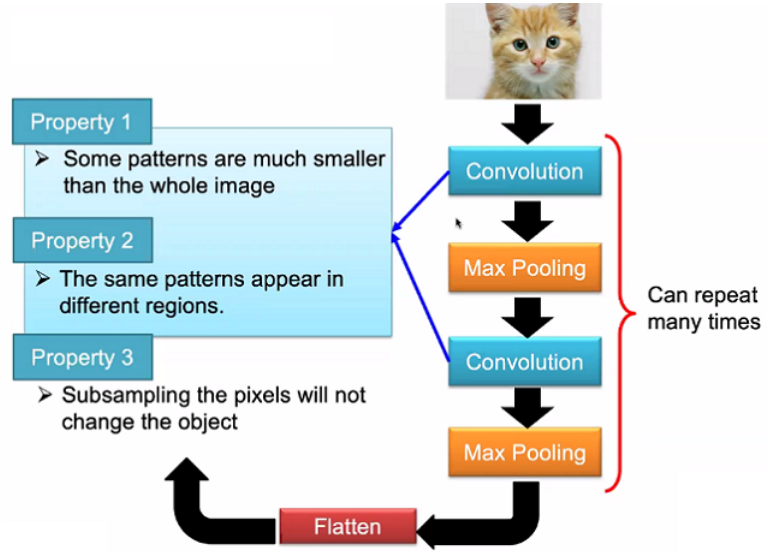
The Whole CNN



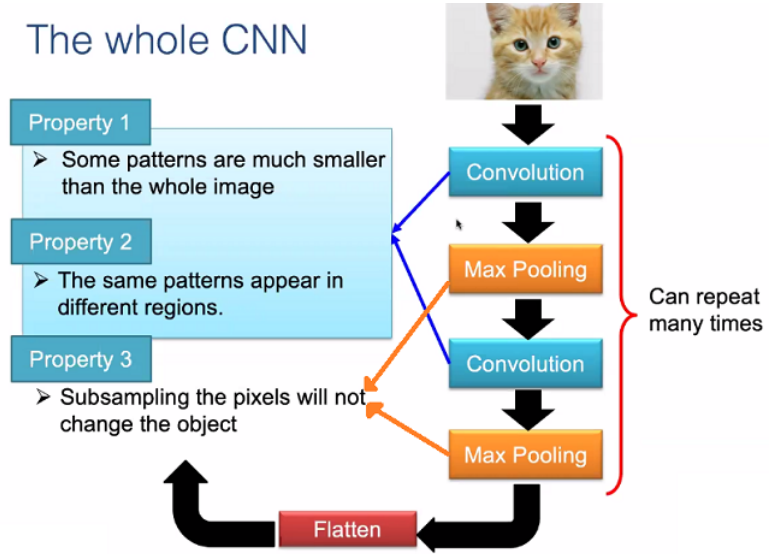
The whole CNN



The Whole CNN



The Whole CNN



In This Segment

Convolution Layer

- Perform Convolution operation.
- In mathematics, convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other. [Wikipedia]
- The term convolution refers to both the result function and to the process of computing it. [Wikipedia]
- We will assume that the kernel is square matrix.
- We will assume that the kernel is centered on the pixel of interest. (One reason why kernel is odd matrix.)
- Convolution operation

$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a,j-b} K_{\frac{m}{2}+a,\frac{n}{2}+b}$$

1-D Convolution

- Suppose we track position of an object using a sensor or GPS device at discrete time intervals.
- Assume that the sensor is noisy.
- To obtain a less noisy estimate, we would like to take a weighted average.
- We would only sum over a small window or a weight array (filter) (w).

$$s_t = \sum_{a=0}^{k} x_{t-a} w - a$$

- Slide the filter over the input and convolve.
- The input and the kernel are one dimensional.

1-D Convolution

$s = x_5 w_0 + x_4 w_1 + x_3 w_2 + x_2 w_3 + x_1 w_4 + x_0 w_5$

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50
W	0.01	0.02	0.03	0.04	0.4	0.5					
S						1.80					
W		0.01	0.02	0.03	0.04	0.4	0.5				
S						1.80	1.96				
W			0.01	0.02	0.03	0.04	0.4	0.5			
S						1.80	1.96	2.11			

2D Convolution Operation

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input Image 6×6

These values inside the matrix are to be learned by the network. We call these as parameters or weights.

1	-1	-1
-1	1	-1
-1	-1	1

Kernel / Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Kernel / Filter 2

Property 1: Each filter detects a small pattern (3×3)

Convolution Operation

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input Image 6×6

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Start at top left corner.
Perform element wise product and add all the values.

Output

3

Convolution Operation

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input Image 6×6

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Slide by one pixel towards right. Perform element wise product and add all the values.

Output

3 -1

Convolution Operation

stride $s = 1$

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input Image 6×6

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Slide by one pixel towards right. Perform element wise product and add all the values.

Output

3 -1 -3

Convolution Operation

stride $s = 1$

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input Image 6×6

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Slide by one pixel towards right. Perform element wise product and add all the values.

Output

3 -1 -3 -1

Convolution Operation

stride $s = 1$

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

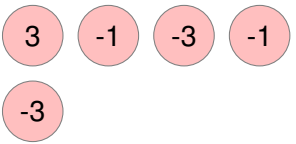
Input Image 6×6

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Jump to next row, left most pixel.
The filter has to completely superimpose the pixels of input.
NOT PARTIALLY.

Output



Convolution Operation

stride $s = 1$

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input Image 6×6

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

After sliding through the entire image.

3	-1	-3	-1
-3	-1	0	-3
-3	-2	0	1
3	-2	-2	-1

Convolution Operation

stride $s = 1$

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input Image 6×6

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

3	-1	-3	-1
-3	-1	0	-3
-3	-2	0	1
3	-2	-2	-1

Property 2 is satisfied.

Convolution Operation

stride $s = 1$

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

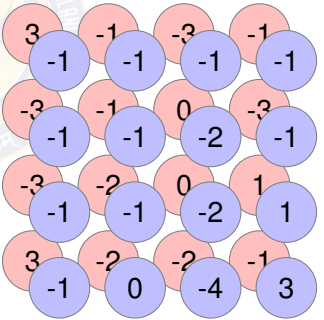
Input Image 6×6

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1



Feature Maps

Convolution Operation

stride $s = 3$

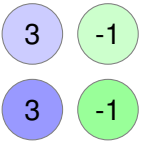
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input Image 6×6

Filter1

1	-1	-1
-1	1	-1
-1	-1	1

Output



Convolution with Striding

- Stride means skip by how many pixels.

$$\begin{aligned} \text{Input Size} &: n \times n \\ \text{Stride} &: s \\ \text{Kernel Size} &: f \times f \\ \text{Output Size} &: \left\lfloor \frac{n-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n-f}{s} + 1 \right\rfloor \end{aligned}$$

- The filter must lie completely inside the input.

Padding

- When filter is convolved with the input image
  - Image shrinks – 6×6 image became a 4×4 image.
  - Pixels in the corners are used only once, when compared the pixels in the middle. The data in the corners are thrown away.
- To resolve, apply padding to the input image before convolution.
- Padding means append zeros around the boundary of the input.

Convolution with Padding

Input Image with padding 8×8

0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	0
0	0	0	1	1	0	0	0
0	1	0	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	0	1	0	1	0	0
0	0	0	0	0	0	0	0

*

Filter1

1	-1	-1
-1	1	-1
-1	-1	1

Padding

- Two kinds of padding
  - Valid Padding
    - Padding is not applied.

$$p = 0$$

    - Input size $\neq$ output size.
    - Output shrinks when compared to the input.
  - Same Padding
    - Padding is applied.

$$p = \frac{f-1}{2}$$

    - Input size $=$ output size.
    - We maintain the size of input and output as the same.

Padding

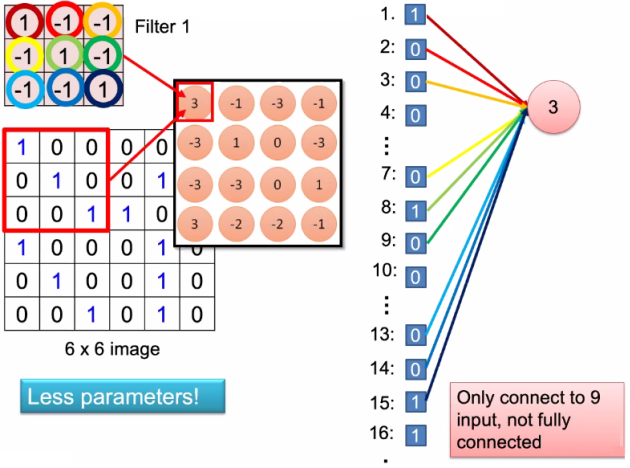
- How much to pad?

$$\text{Padding} : p = \frac{f-1}{2}$$
$$\text{Kernel } 3 \times 3 : p = 1$$
$$\text{Kernel } 5 \times 5 : p = 2$$
$$\text{Kernel } 7 \times 7 : p = 3$$

- Padding preserves the input size.

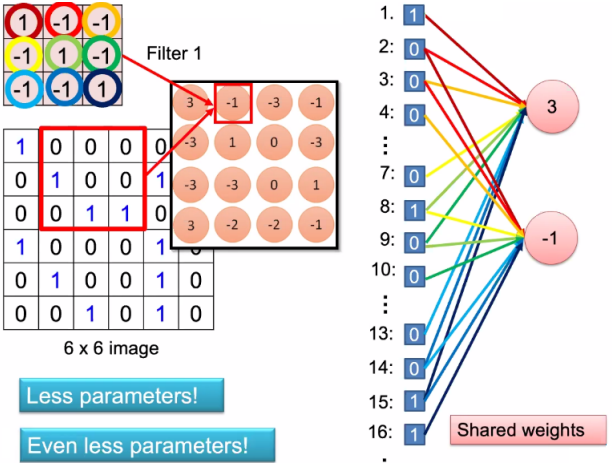
$$\text{Input Size} : n \times n$$
$$\text{Padding} : p$$
$$\text{Kernel Size} : f \times f$$
$$\text{Output Size} : (n + 2p - f + 1) \times (n + 2p - f + 1)$$

Convolutional Layer



Courtesy: Akka Zemmari

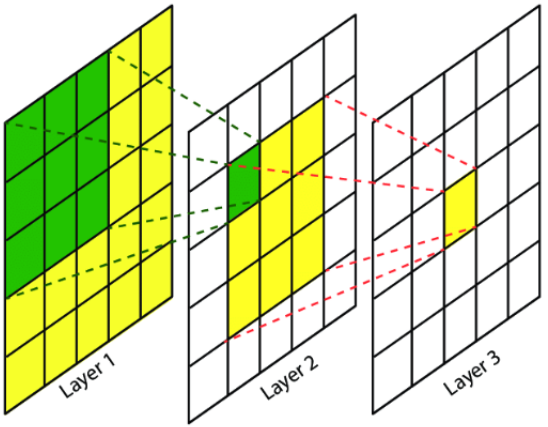
Convolutional Layer



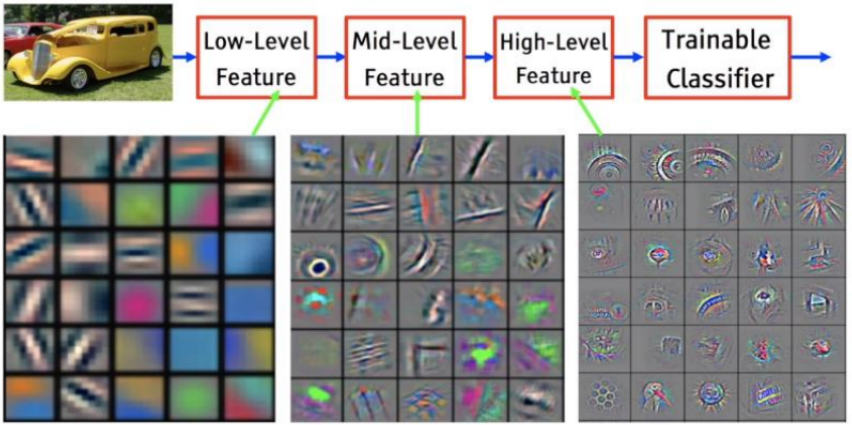
Courtesy: Akka Zemmari

Activations or Receptive Fields

- Effective area "seen" by a neuron
- What is the effective number of pixels seen at each layer?
  - Input image x
  - $h_1 = Conv2d(x, k = 3)$
  - $h_2 = Conv2d(h_1, k = 3)$
  - $h_3 = Conv2d(h_2, k = 3)$
  - ...
  - $h_n = Conv2d(h_{n-1}, k = 3)$
- Homework: What happens if the kernel size was 5?



Local to Global



Yann LeCun's presentations on ConvNets

Parameter Size in Convolution Layer

- Input image: $B \times W_{in} \times H_{in} \times C_{in}$
  - Input data has B samples
  - Input or Image size is $W_{in} \times H_{in} \times C_{in}$
- Convolution filter
  - Weight parameters: $C_{in} \times C_{out} \times K_W \times K_H$
  - Bias parameters: C_{out}
- Output activation: $B \times W_{out} \times H_{out} \times C_{out}$
  - C_{out} filters of shape $C_{in} \times K_W \times K_H$ operate on each sample

Parameter Size in Convolution Layer

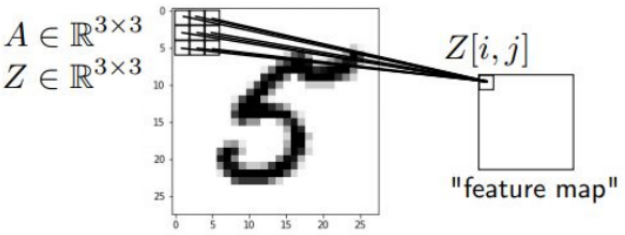
- Each filter gives one 2D output.
- K filter will give K such 2D outputs.
- The number of parameters depend on the number of kernels and the kernel size.

$$\text{Number of kernels in layer } (l-1) : K_{l-1}$$
$$\text{Kernel size} : f \times f$$
$$\text{Number of kernels in layer } (l) : K_l$$
$$\text{Number of Parameters in layer } (l) : K_{l-1} * f * f * K_l$$

Cross-Correlation vs Convolution

- Convolution in Deep Learning is actually cross-correlation.
- Cross-correlation is sliding dot product over the images.
- Cross-correlation

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{-k}^{k} K[u,v] A[i+u, j+v]$$



Cross-Correlation vs Convolution

Cross-Correlation

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{-k}^{k} K[u,v]A[i+u, j+v]$$

(Note the looping direction given in red.)

1) -1,-1	2) -1,0	3) -1,1
4) 0,-1	5) 0,0	6) 0,1
7) 1,-1	8) 1,0	9) 1,1

Convolution

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{-k}^{k} K[u,v]A[i-u, j-v]$$

(Flipping the kernel.)

9) -1,-1	8) -1,0	7) -1,1
6) 0,-1	5) 0,0	4) 0,1
3) 1,-1	2) 1,0	1) 1,1

Cross-Correlation vs Convolution

- Convolution in Deep Learning is actually cross-correlation.
- Convolution (as in Signal Processing) has nice association property.

$$(A * B) * C = A * (B * C)$$

- In DL, we usually don't care about that (as opposed to traditional computer vision and signal processing applications.)
- Cross-correlation is easier to implement.
- Maybe the term "convolution" for cross-correlation because popular, because " Cross-correlation Neural Network" sounds weird.

Transposed Convolution

Regular Convolution:

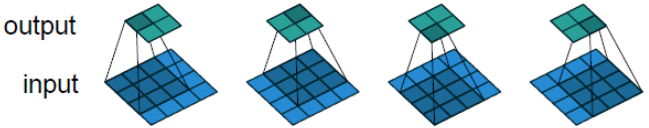
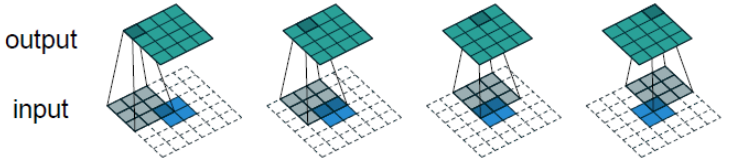


Figure 2.1: (No padding, unit strides) Convolving a $3 * 3$ kernel over a $4 * 4$ input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

Transposed Convolution (emulated with direct convolution):



Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Dilated Convolution

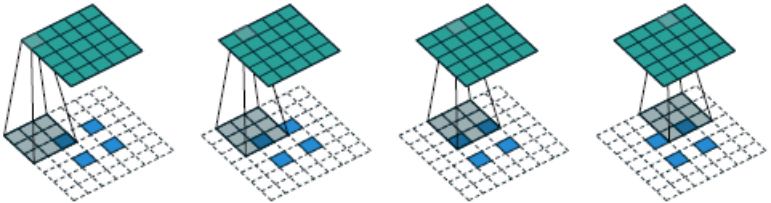
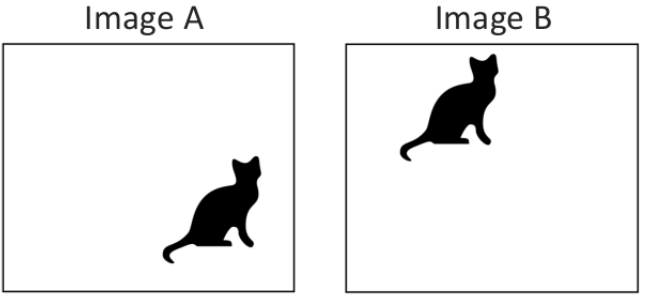


Figure 4.5: The transpose of convolving a $3 * 3$ kernel over a $5 * 5$ input using $2 * 2$ strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 0$). It is equivalent to convolving a $3 * 3$ kernel over a $2 * 2$ input (with 1 zero inserted between inputs) padded with a $2 * 2$ border of zeros using unit strides

Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Remember Image Invariance Challenge

Object can appear anywhere in the image.



Model predicts “cat” for both images!

Source: https://penkovsky.com/neural-networks/day5/

Pooling (Pool) Layer

- Helps to achieve image invariance. Objects can appear anywhere in the Image and still get detected.
- Used for sub-sampling
- Reduce the size of representation
- Speed up computation

Pooling

- Apply a filter of size f and stride s. Padding is not applied.
- The output dimensions will be

$$\left\lfloor \frac{n-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n-f}{s} + 1 \right\rfloor$$

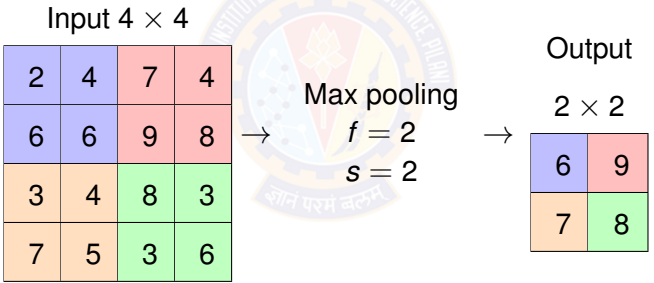
- Fixed computation
- Gradient descent is not applied as there are not parameters to be learned.
- Pooling is applied on each of the channels.

Pooling Layer

- Two types
  1. Max pooling
  2. Average pooling

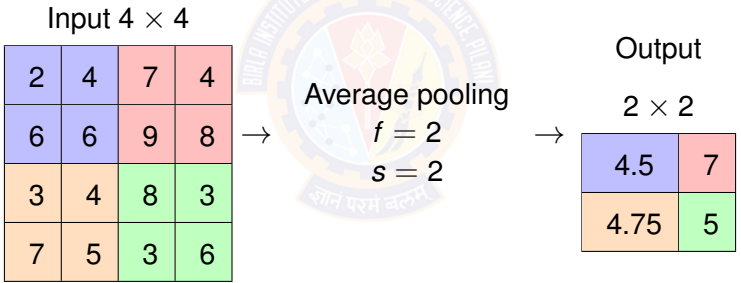
Max Pooling

- Take the maximum of the sub region that is considered.



Average Pooling

- Compute the average of the sub region that is considered.



Convolution + Max Pooling

Input Image 6×6

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1

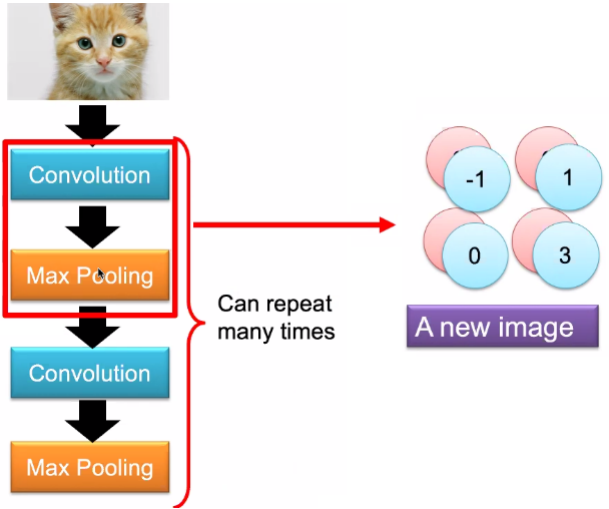
Convolve s=1

3	-1	-3	-1
-3	-1	0	-3
-3	-2	0	1
3	-2	-2	-1
-1	-1	-1	-1
-1	-1	-2	-1
-1	-1	-2	1
-1	0	-4	3

Maxpool f=2, s=2

3	0
3	1
-1	-1
0	3

Convolution + Max Pooling



Flatten

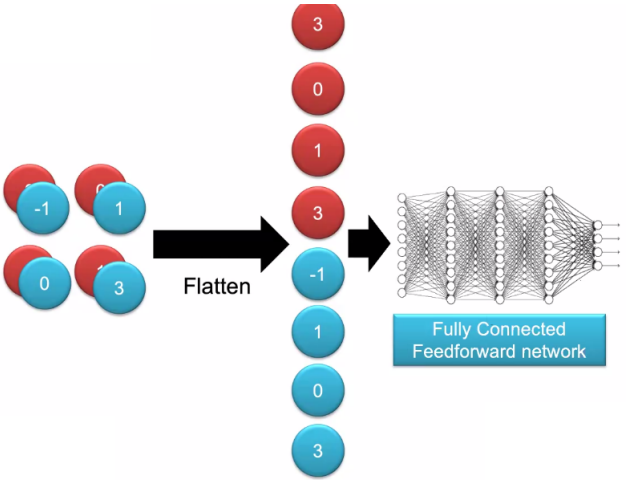
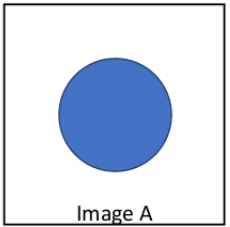
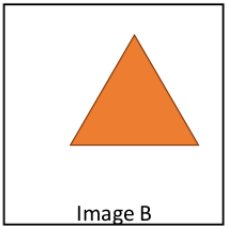
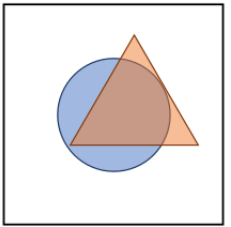
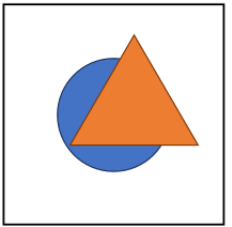

-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

Image is Highly Non-Linear



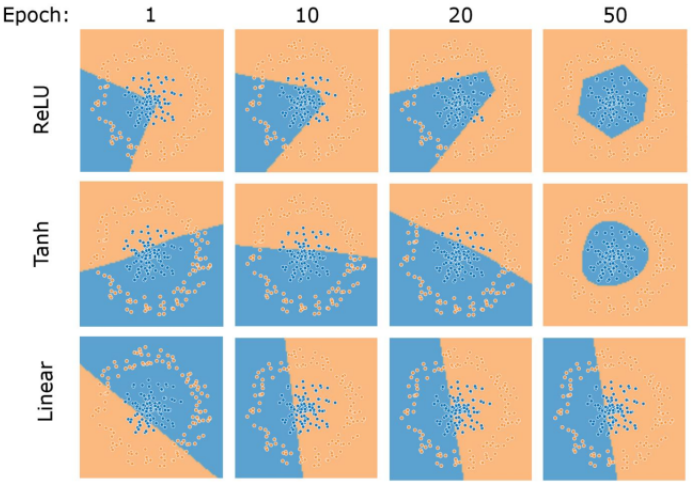






Rectified Linear Unit (ReLU)

- Very fast
- Gradients
- Piece-wise linear approximation



ReLU in Images

Jon Barron's talk at the MIT Vision and Graphics Seminar 2021 on Neural Radiance Fields

Loss for Training

- Mean Squared Error
- Binary Cross Entropy
- Multi class Cross Entropy
- Class-weighted cross-entropy losses (as a way to counter imbalance)
- L1 (MAE), L2 (MSE) losses
- Focal, Dice, Huber loss
- Margin ranking losses
- Generalized Intersection-over-Union (GIoU) loss
- Connectionist Temporal Classification (CTC) loss

Regularisation and Optimization

- Drop out
- Batch Normalisation
- RMSProp, Adam or any other optimizer can be used.

Classification

- $p \in [0, 1]$ - use Sigmoid activation function
- k classes - use softmax activation function

In This Segment

Putting Everything Together

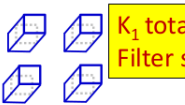
1. Identify the task as classification or regression. This helps in identify the output size.
2. Collect the N images. This helps in identifying input size as $W \times H \times C$.
3. Feature extraction
   - Decide on the number of Convolution layers. This can be derived by looking at how large or small pattern we are interested in.
   - Decide on the kernel size and number of kernels in each layer. Smaller patterns in images require smaller kernels. Number of kernels can be derived from the non-linearity.
   - Use ReLU activation function for better approximation.
4. Classification
   - Use Fully connected layers for classification.
   - The number of (dense) layers depends on the non-linearity.
   - The last (output) layer activation is decided based on Step 1.
5. Train and Test on the available dataset.

Convolutional Neural Network

- Input: 1 or 3 images
  - Grey scale or color
  - Will assume color to be generic

Convolutional Neural Network

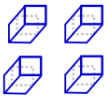
- Input is convolved with a set of K_1 filters or kernels.
  - Typically, K_1 is a power of 2 e.g 2,4,8,16,32,...
  - Filters are typically 5x5x3, 3x3x3 or 1x1x3 (better notation).
  - Typical stride is 1 or 2.



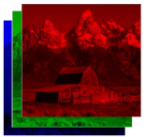
$I \times I$ *image*

Convolutional Neural Network

- Input is convolved with a set of K_1 filters or kernels.
  - Typically, K_1 is a power of 2 e.g 2,4,8,16,32,...
  - Filters are typically 5x5, 3x3 or 1x1.
  - Filters should be small enough to capture fine features, particularly important for scaled down images.



K_1 total filters
Filter size: $L \times L \times 3$



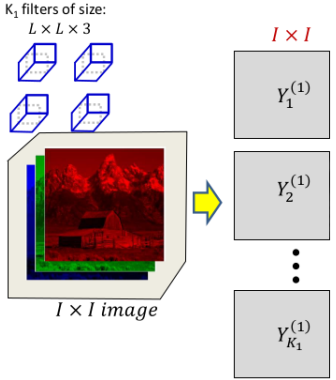
$I \times I$ *image*

Parameters to choose: K_1, L and S
1. Number of filters K_1
2. Size of filters $L \times L \times 3 + bias$
3. *Stride* of convolution S

Total number of parameters: $K_1(3L^2 + 1)$

Convolutional Neural Network

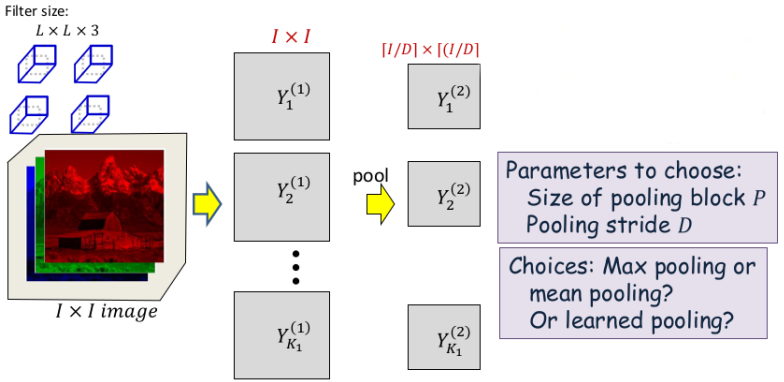
- First Convolutional layer:
  - Several convolutional filers, which are 3D.
  - Convolution is followed typically by a ReLU activation.
  - Each filter creates a singly 2D feature map.



The layer includes a convolution operation followed by an activation (typically RELU)

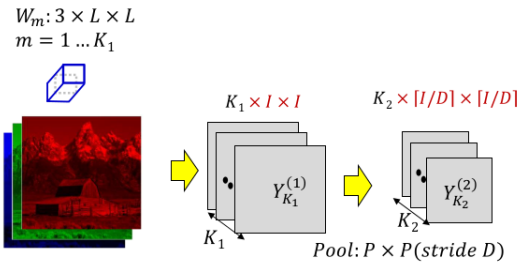
Convolutional Neural Network

- First Pooling layer:
  - For each $P \times P$ block of each map, pool down to a single value.



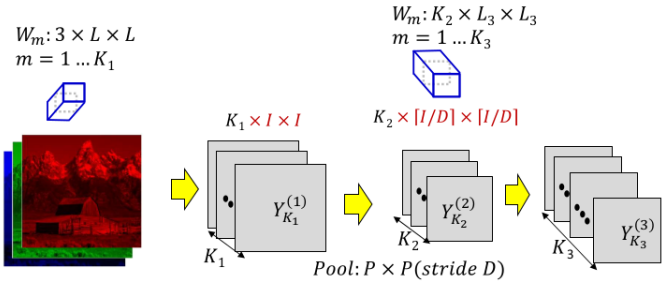
Convolutional Neural Network

- Better drawing



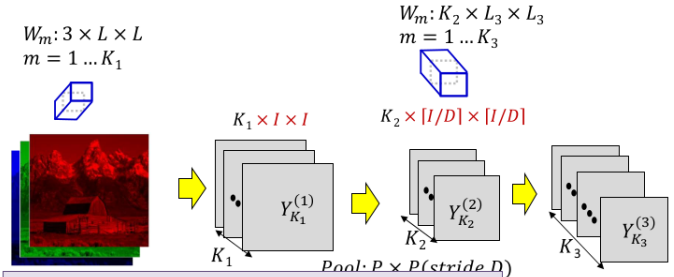
Convolutional Neural Network

- Second Convolutional layer" K_2 3D filters resulting in K_2 feature maps.



Convolutional Neural Network

- Second Convolutional layer



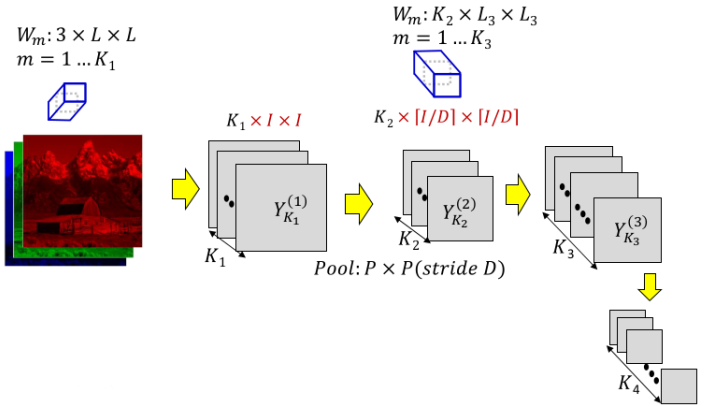
Parameters to choose: K_3, L_3 and S_3

1. Number of filters K_3
2. Size of filters $L_3 \times L_3 \times K_2 + bias$
3. *Stride* of convolution S_3

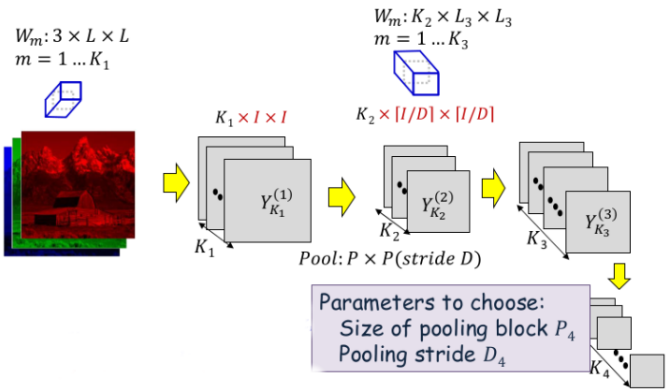
Total number of parameters: $K_3(K_2 L_3^2 + 1)$
All these parameters must be learned

Convolutional Neural Network

- Second Pooling layer: K_2 pooling operations results in K_2 reduced 2D maps.

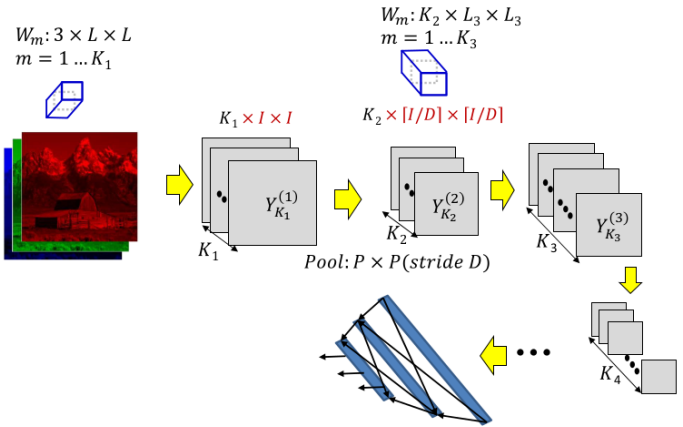


- Second Pooling layer



Convolutional Neural Network

- Fully connected layer: This continues for several layers until the final convolved feature maps are fed to MLP.



Convolutional Neural Network

Size of Layers

- Each convolution layer with stride 1 typically maintains the size of the image.
  - With appropriate zero padding.
  - If performed without zero padding it will decrease the size of the input.
- Each convolution layer will generally increase the number of maps from the previous layer.
  - Increasing layers reduces the amount of information lost by subsequent downsampling.
- Each pooling layer with stride D decreases the size of the maps by a factor of D.
- Filters within a layer must all be the same size, but sizes may vary with layer. Similarly for pooling D may vary with layer.
- In general the number of convolutional filters increases with layers.

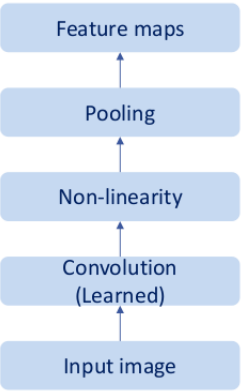
Convolutional Neural Network

Parameters to choose (design choices)

- Number of convolutional and downsampling layers
  - And arrangement (order in which they follow one another)
- For each convolution layer:
  - Number of filters K_i
  - Spatial extent of filter $L_i \times L_i$
    - ⋆ The "depth" of the filter is fixed by the number of filters in the previous layer K_{i-1}.
  - The stride S_i
- For each downsampling/pooling layer:
  - Spatial extent of filter $P_i \times P_i$
  - The stride D_i
- For the final MLP:
  - Number of layers
  - Number of neurons in each layer

CNN-Recap

- Neural network with specialized connectivity structure
- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised training
- Train convolutional filters by back-propagating error
- Convolution over time



References

1. Deep Learning with Python by Francois Chollet.
   `https://livebook.manning.com/book/deep-learning-with-python/`
2. Deep Learning by Ian Goodfellow, YoshuaBengio, Aaron Courville
   `https://www.deeplearningbook.org/`

Thank You!