Today's Agenda

1) Convolutional Neural Networks (CNNs)

ANN / MLP ✓✓

Visual Computing
Computer Vision

Images / Video

Applications

1) Image Classification — Dog / Cat → Object
2) Object Detection — GANs
3) Image Segmentation — 1.5 - 2 months
4) Face Recognition — Practicals
5) Object Tracking
6) Anomaly Detection — CV → 2012
7) OCR — NLP → 2018-19

Number Plate

Object Detection + OCR

Colour Images

Channels

- RGB → Printing
- CMYK
- RGBA
- BGR → HUE HSI
- GRAY
- RGB IOR

Photoshop

Channels → Feature Maps

Kodak — BGR → RGB
Fujifilm

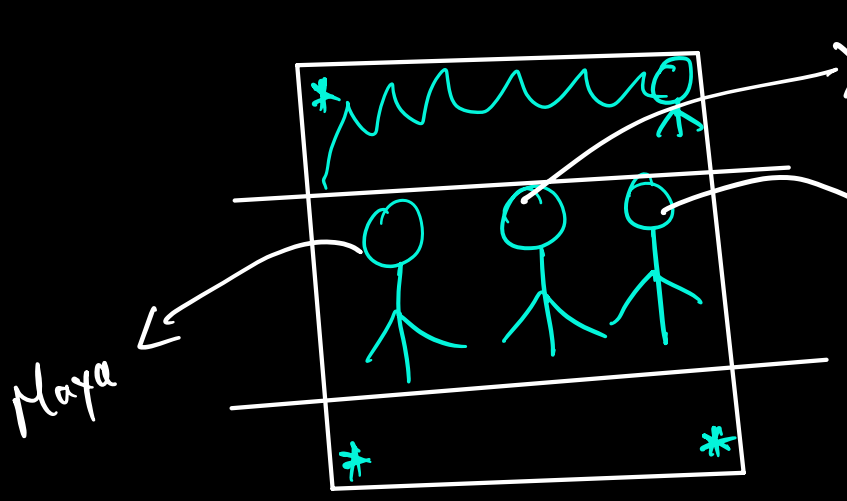
OpenCV → BGR

Imaging Libs
1) openCV
2) PIL

R → Red
G → Green
B → Blue

Combine: → 7 million Colours

Multiple Colour Combinations



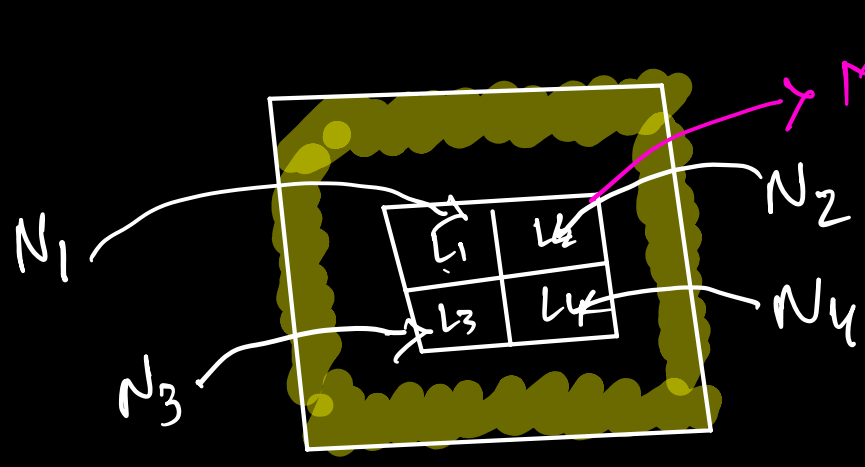
60% information

Local Regions

1) Mid Part of the image

Distant Regions

1) Edges of the image



$N_1 \rightarrow L_1$

$N_2 \rightarrow L_2$

Bound by Coordinates

CNN Components

1) Image
2) Kernel → Filters / Feature Extractors / Edge Detectors

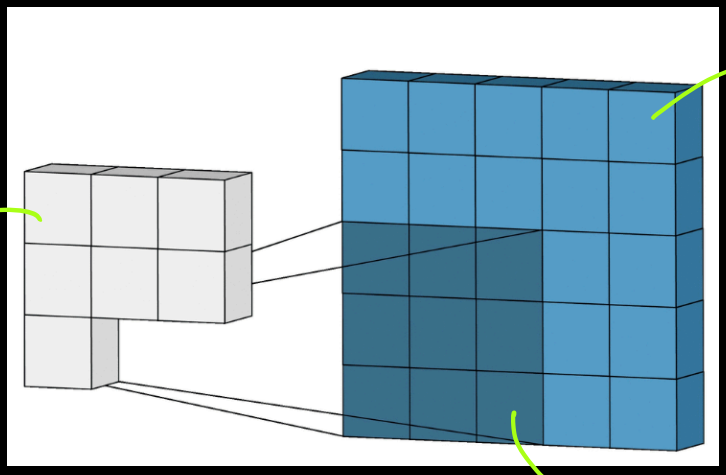


Image (10x10)

Output

Kernel → convolving on image (3x3)

Principle:

5 x 5 → (3x3) → 3 x 3

Input Image size — Kernel size — Output feature size

5 x 5 → 3 x 3 (Reduced Size)

2 pixels at the top and 2 pixels in bottom

2 x 2 = 4 pixels

Image Matrix

Kernel Matrix

Output Matrix

Image • Kernel → Output

(3x3)

A ∪ B

[[R], [G], [B]]

Kernel will be of 1×1 to any no

Practice :: Even Kernels ✗

Odd Kernels ✓

Input image

CNN (No of Kernels, kernel size)

$2^{n=4}$, 3 × 3

16 kernels, 3×3

In every layer in CNN, each Kernel will extract a unique feature.

Kernel 3×3 Weights

$$\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}$$ → updated during Backprop

x = float 64

Model → best weights

Part of the image $[M_1]_{3\times3} \cdot [M_2] = \text{Output Val}$

$[M_2]$ → 3×3 Kernel

3×3 = 9 pixels

Receptive Field

3×3 = 9

11×11 = 121

3×3 (1) → 5×5 (2) → 7×7 (3) → 9×9 (4) → 11×11 (5)

5 (3×3) ⇒ 11×11

45 ⇒ 121

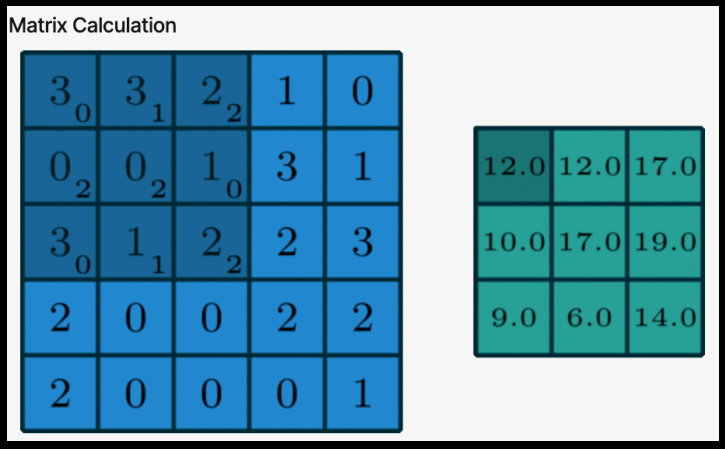
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow$$ Kernel values / weights /

↳ Backprop

Weight Initializer

Kernel Size, No of Kernels, Input image size

All are hyperparameters



$= (3\times0) + (3\times1) + (2\times2) +$
$(0\times2) + (0\times2) + (1\times0) +$
$(3\times0) + (1\times1) + (2\times2)$

$= 3 + 4 + 1 + 4$

$= 12$

Feature Maps

Image → Kernel → Output

h x w x c

↳ channels

feature maps

5×5 → 3×3 → 3×3×16

16 Kernels

Feature Accumulation

Single Kernel output = 3×3

16 Kernel output = 3×3×16

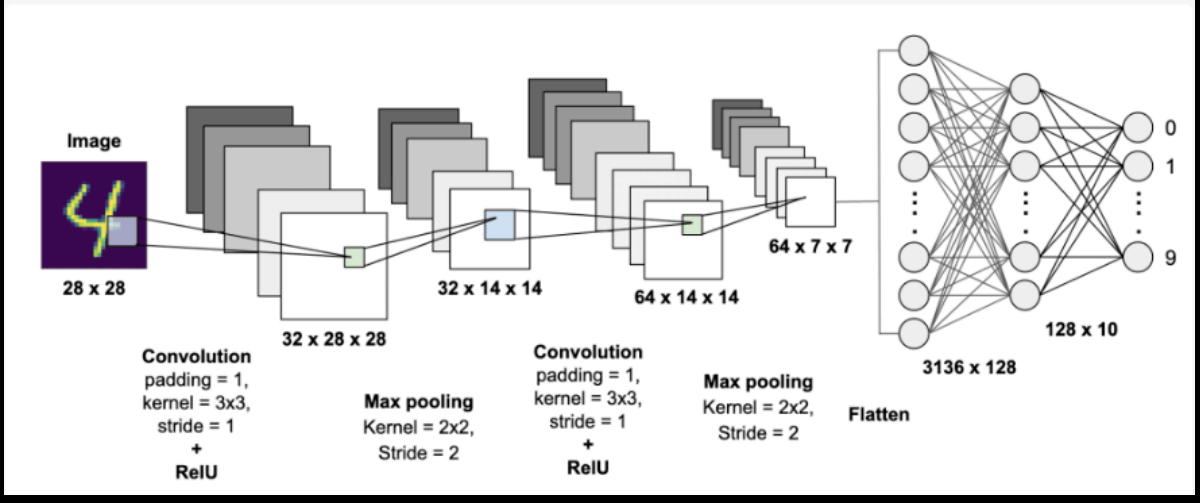
R G B

[ ] [ ] [ ]

16 Kernels

Image → RGB → (224, 224, 3) → 3×3 / 16 kernels → 222×222×16

Batch Norm is applied on channel / feature info



Conv 1

Input

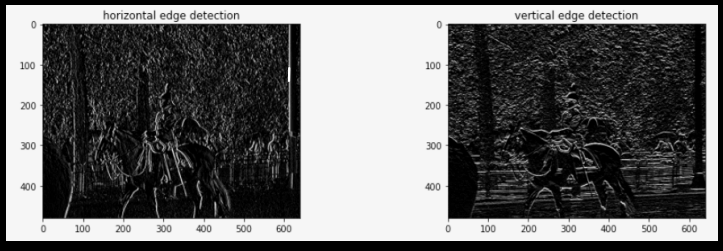
28 x 28 x 1

Conv Op

padding 1 → 28 x 28 x 32

padding 0 → 26 x 26 x 32

Filters




```
sobel_y = np.array([[ -1, -2, -1]
                    [ 0, 0, 0],
                    [ 1, 2, 1]])
# vertical edge detection
sobel_x = np.array([[-1, 0, 1],
                    [-2, 0, 2],
                    [-1, 0, 1]])
```

Horizontal, Vertical, Diagonal (2 types)

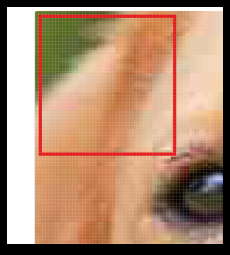
Levelwise Feature Extraction

Features → Low, Mid, High

model = Sequential ( )

- Cnn layer 1
- Cnn layer 2
- Cnn layer 3
- Max Pooling
- Cnn layer 4
- Max Pooling
- Cnn layer 5

Extract



Low level → Mid level → High

Low level features require less kernels

Keep on increasing no of kernels as we go down / close the network.

32, 64, 128, 256, 512 →

32, 32, 64, 128
MP
32, 64, 128

$\begin{bmatrix} 8 \\ 16 \\ 32 \end{bmatrix}$ Block (1)

Max Pooling = $h \times w \times 32$

$\begin{bmatrix} 16 \\ 32 \\ 64 \end{bmatrix}$

Edges / Gradients → Patterns → Textures → Parts of Objects → Object → Scene