## What is a Convolution

- · Operation that involves adding & multiplying
- · Input image \* Filter = Output Image

  for olve

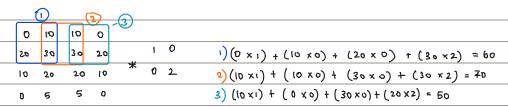
  operation

· useful for performing edge detection

- · convolution is like image modifier
  - L feature transformation
- · Different filters will result in different outputs
  - L Gaussian filter = Blurs image

    Edge detection filter = Edge detection

# Convolution Operation



# + convolution is commutative

output matrix

### Convolution output

if input length = N then output length = (N - K) + 1kernel length = K

- \* Images are typically not square & some neural nets use square for convenience
- \* Filters are almost always square

#### Convolution Equation

- \* Filters in CNNs are learned using gradient descent
  - Ll. Filters assigned with random values
  - 2. Input convolve filters to produce feature maps
  - 3. Loss calculated & compared with labels
  - 4. Backprop to calculate gradient
  - 5. Filters updated using gradient descent

### Padding

- Pad input array with 0s to make convolved output same size as input
  - "valid": output = N-K+1 (filtur can only bouch valid input)
  - "same": output = N (output same size as input, done with padding)
  - "full", output = N+K-1 (ensure non-zero output) (Not typically used)

Dot Product & correlation Angle blw 926 . a.b = 2 92 bi = 191161 cos Pab · Also called cosine similarity / cosine distance

· cos(0) = 1 & same dire

cos(90) = 0 1 porthogonal

cos(180)=-1 2 opp dirg.

 $\cos \theta_{ab} = \frac{q \cdot b}{\tan (b)}$  : how similar are a 2 b.

magnitude of vector  $|a| = \frac{2}{2}q_i^2$ 

0: orthogonal

Pearson correlation: (3: -a) (6: -b) = very similar to det

| \frac{2}{\mathbb{Z}(9\diversity)^2} \frac{2}{\mathbb{Z}(6\diversity)^2} \fra

Dot product is like a correlation measure of 2 vectors. L related to filter in CNN finding correlated features

# Matrix operation & convolution

$$a = \begin{bmatrix} a_1 \\ a_2 \\ q_3 \end{bmatrix} \qquad W = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} \qquad \alpha_1 * W = \begin{bmatrix} G_1W_1 + \alpha_2U_2 \\ G_2W_1 + \alpha_3U_2 \\ Q_3W_1 + G_4W_2 \end{bmatrix} = \begin{bmatrix} W_1 & W_2 & 0 & 0 \\ 0 & W_1 & W_2 & 0 \\ 0 & W_1 & W_2 & 0 \end{bmatrix} \begin{pmatrix} g_1 \\ G_2 \\ G_3 \\ G_4 \\ Q_4 \\ Q_4 \\ Q_4 \\ Q_4 \\ Q_5 \\ Q_6 \\$$

### Consider opposite case:

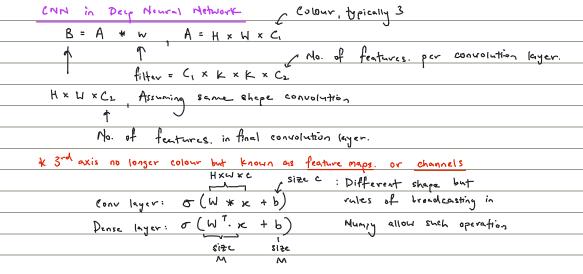
- · using convolutions to replace matrix multiplications.
- · main idea behind parameter / weight sharing

#### Translation invariance

- · convolution operation recognize patterns / features regardless of location within input space
- · filter slides over input data, ensuring same filter applied across entire input

### CNN Input vs output

- · For colour Images , shape = H x W x 3
- · Kernel size = KxK
- · output = H x W for same size operation.
- \* Discrepancy: loput = 30, output = 20. Need some shape to stack convolution layers. Solz: For each feature output, stack it.



### cost savings (conv vs matrix multiplication)

Convolution:

1944 : 32 × 32 × 3

kernel: 3 x 5 x 5 x 64 = 4800 params.

output: 28 x 28 x 64

(32-5+1)

Matrix Multiplication; (flatten)

Input = 32x32x3 = 3072

output = 28 x 28 x 64 = 50176

Weight: 3072 x 50176 = 154 000 000 params.

L 32000 x more than conv operation.

Suffers from translation variance.

Head to learn every possible location