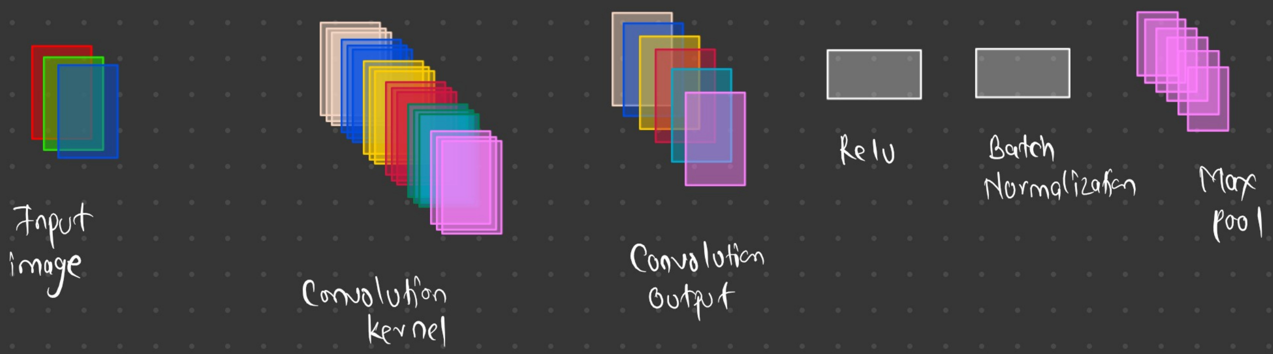


$$\text{Relu} \xrightarrow{max} (0, x)$$

$$x \to max(0, 16) \to \underline{\underline{16}}$$

$$x \to 16$$

Conv2D
- kernel – 3x3
- Stride – 2
- padding – 0
- kernel dimension – ?
  - Input – dimension = 3
    - Kernel = 3x3x3
- Total kernel = 6

<u>8 – 12 – 2024</u>

→ CNN flow
  → BN
  → relu
  → channels

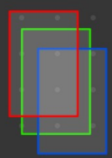
→ Forward & Backward propagation

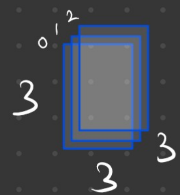
Input Image

$100 \times 100 \times 3$

Convolution operation

$k=3,\ p=0,\ s=1,\ N.k = 6$



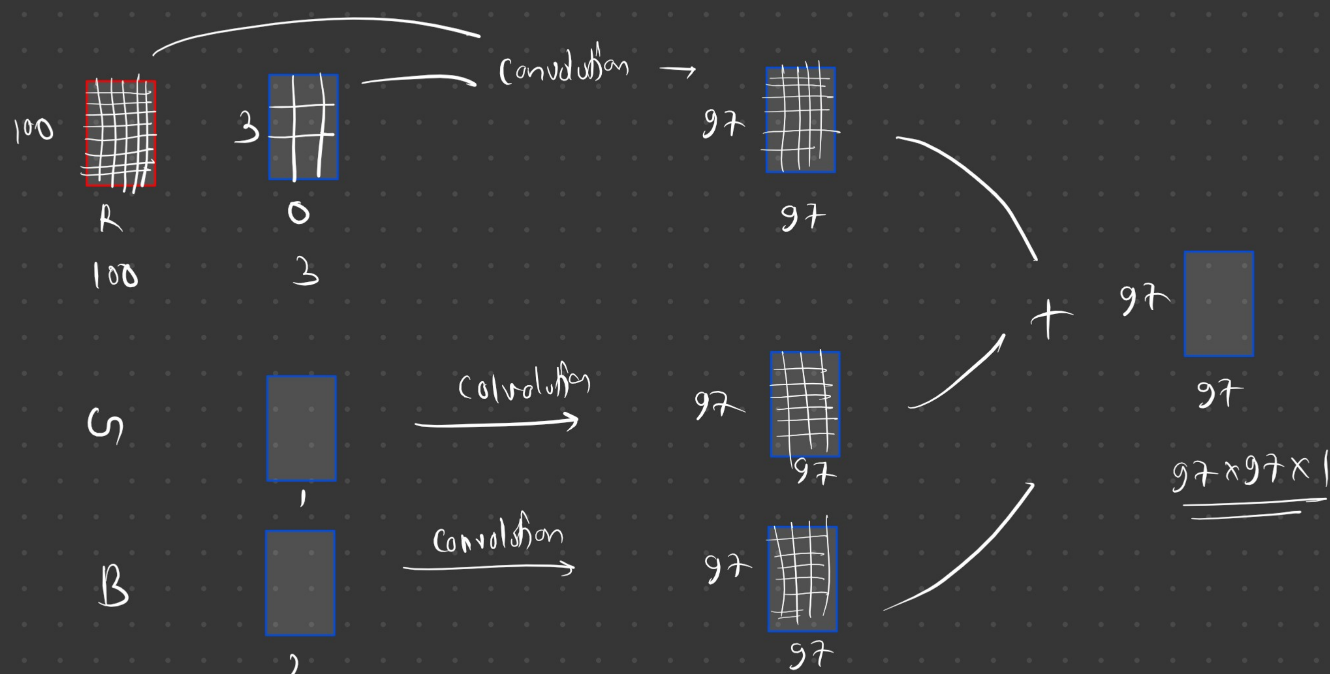


K. N-1

$6 \rightarrow 97 \times 97 \times 1$

$\boxed{97 \times 97 \times 6}$

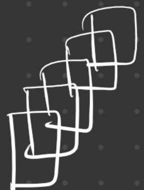
Each
Dimension of kernel → Same as input



Conv

$k-3,\ s-1,\ p-0$
$N.k = 64$

Output → ? → 64
no of channel?



$16 \times 16 \times 32$

$\boxed{14 \times 14 \times 64}$

single kernel size dim → ?

$\boxed{3 \times 3 \times 32}$ ✓

$14 \times 14 \times 1$

Batch normalization

1 image → SGD

100 → 10 batch →

- 10 img — 1
- 10 img — 2
- 10 img — 3
- ⋮
- 10 img 10

taking batch no. 1 for training

image
10 × 100 × 100 × 3

k-3, S-1, p-0, 32

→ image dimension →

img 1 → 97 × 97 × 32

⋮

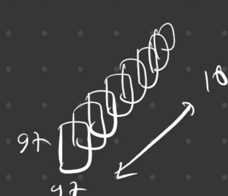
img 10 → 97 × 97 × 32

→ 10 × 97 × 97 × 32

↓

channel

→ 1/32 channel → 10 images →



⋮

$x \rightarrow \dfrac{x - \mu}{\sigma}$

→ μ & σ

$\mu \downarrow$ val1

$\sigma \downarrow$ val2

→ 32nd/32 → 10 image → μ & σ → normalise each pixel

→ $\boxed{\text{scale}}$ & $\boxed{\text{shift}}$ $\boxed{+1 \;\; +1}$

0 – 1

scale → 2

0.1 → 2   0.112   0.3 → 3

$\boxed{0.2}$ $\boxed{0.6}$

filters , $\boxed{\gamma, \beta}$

Tools → Convolution → weighted sum → loss function

→ stride , Padding → neuron → BN

→ pooling → Activation function

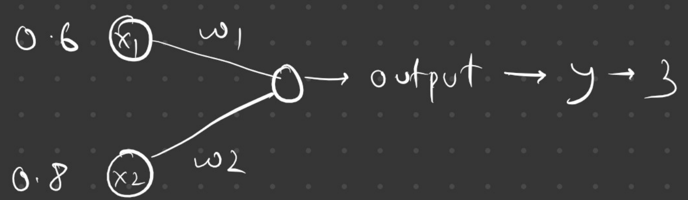
→ FC layer → kernel → weight & biases

→ Learning ✓✓

X_1	X_2	y
0.6	0.8	3

→

w_1	w_2
0.5	0.1
0.7	0.1



→ $(0.6 \times 0.5) + (0.8 \times 0.1) \rightarrow 0.38 \rightarrow \hat{y}$

$(0.7 \times 0.5) + (0.8 \times 0.1) \rightarrow 0.43$

$y_{predicted}$

(1) Forward pass

(2) loss → $(\hat{y} - y)^2 \rightarrow (0.38 - 3)^2 = 6.864$

→ $(0.43 - 3)^2 = 6.604$

(3) we calculate gradient of our learnable parameter.

w_1 = value

w_2 = value

> A value that indicates how much a NN parameter should change to reduce error

L = 0.84   if i change w_1 with a small unit how much does it affect loss

0.5 → 0.2 +ve → 0.7 →   +ve how much amount

→ -0.2 → 0.3 →   +ve

→ Back propagation → gradients → +ve, -ve, how much?

$w_1 = 0.5$ $w_1 = \frac{\partial loss}{\partial w_1} = 0.3$

→ Optimiser → Gradient descent

→ updates the learnable parameters based on their calculated gradients.

learning rate η — it influences how much of gradient you want.

$$w_{1new} = w_{1old} - \eta \, w_{1.G}$$

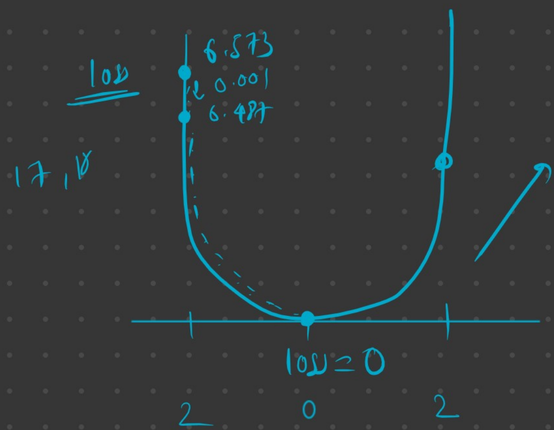
$$= 0.5 - 1 \cdot 0.3$$

$$= 0.5 - 0.3 \rightarrow 0.2$$

$$= 0.5 - 0.01(0.3)$$

$$= 0.497$$

→ loss ↑ ↓



1.9 → 2

1.973 → 2

8 → 2

Adam

RMSprop

local minima →

global minima →