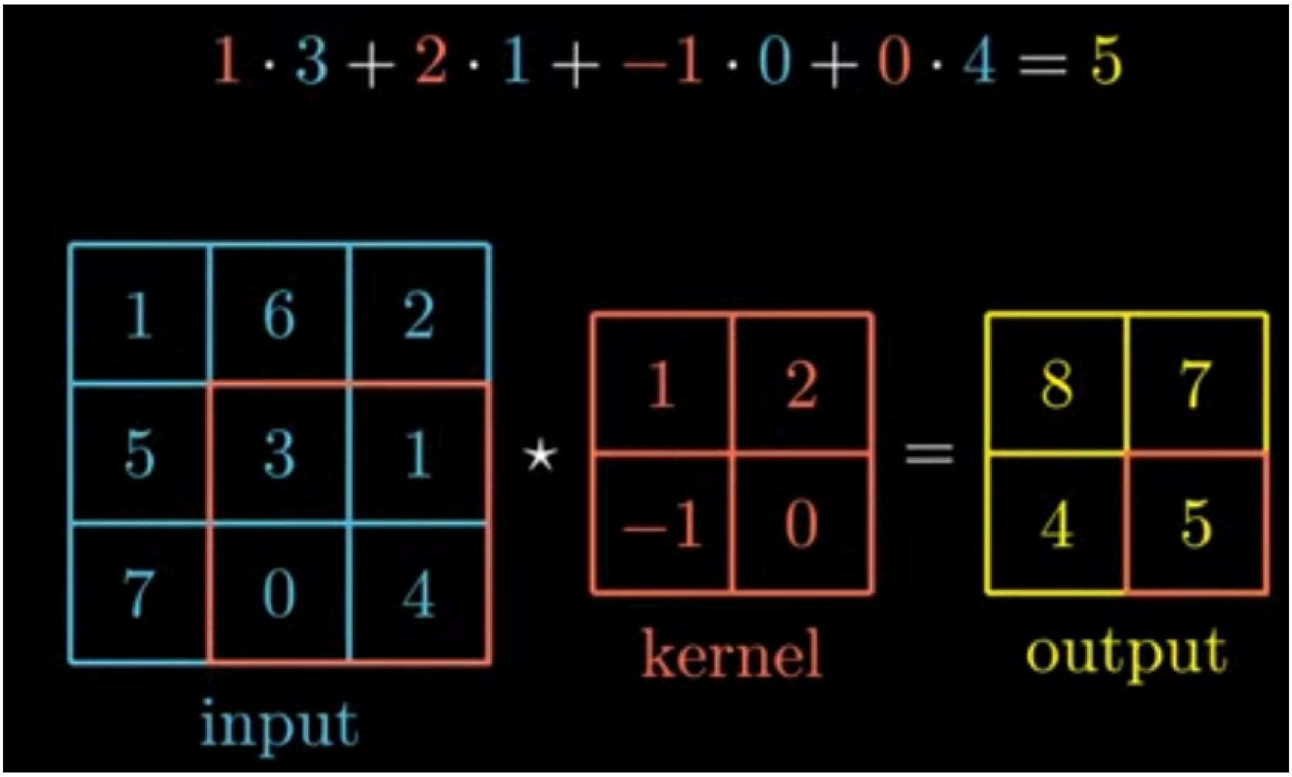
Convolutional Neural Network

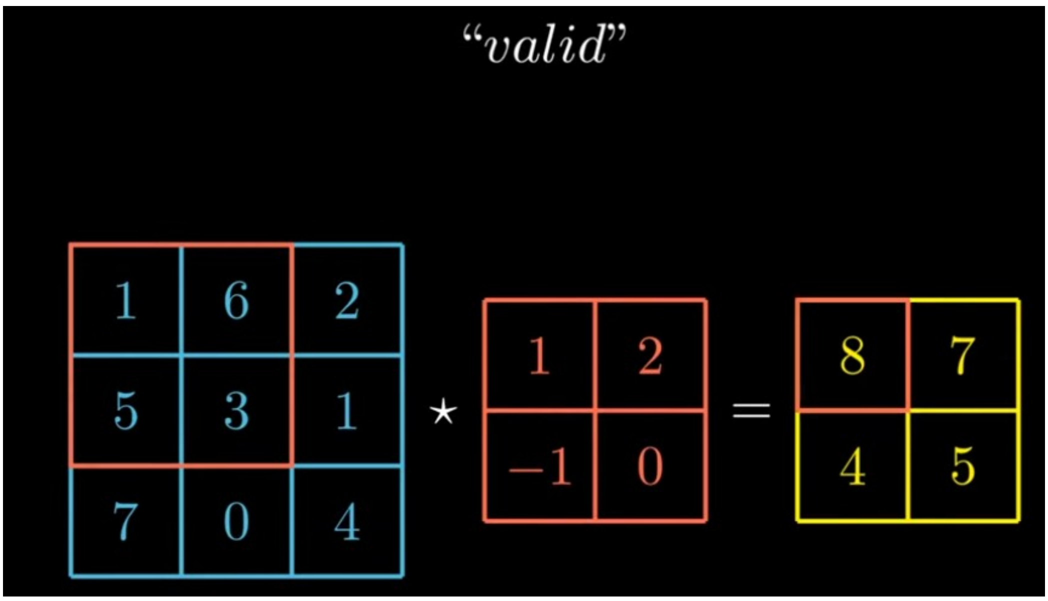
Cross-Correlation:



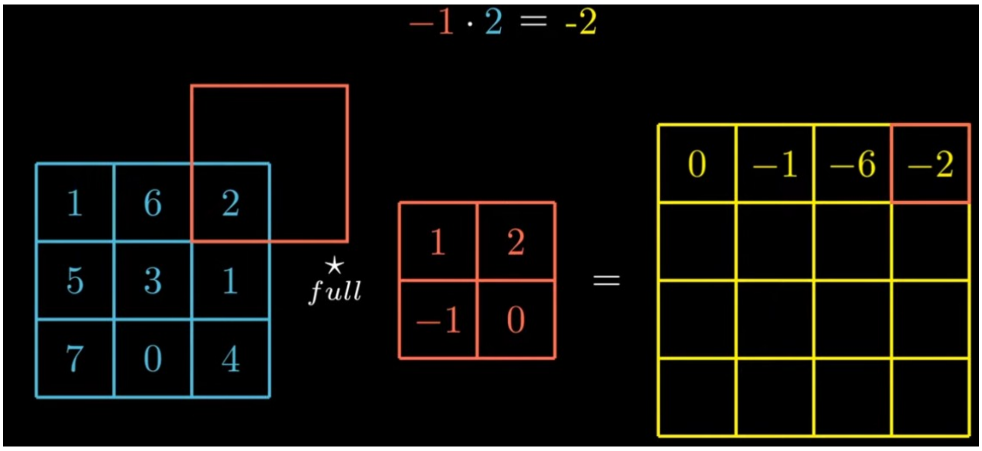
$$Y = I - K + 1$$

size of output Y = I – (K + 1) ?

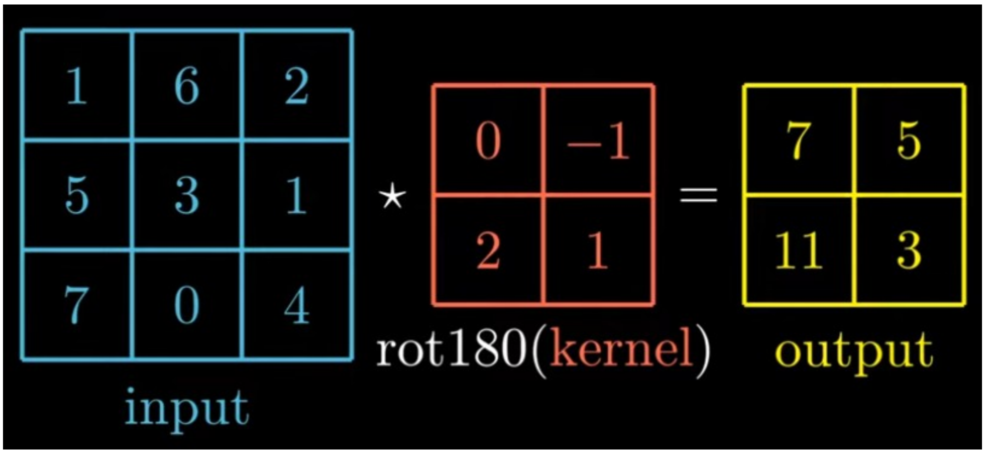
Valid Cross-Correlation

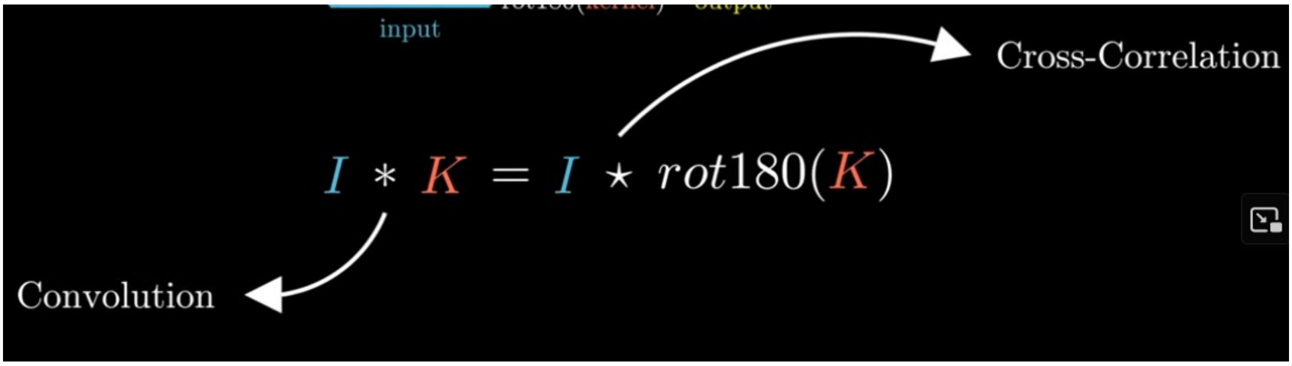


Full Cross-Correlation:

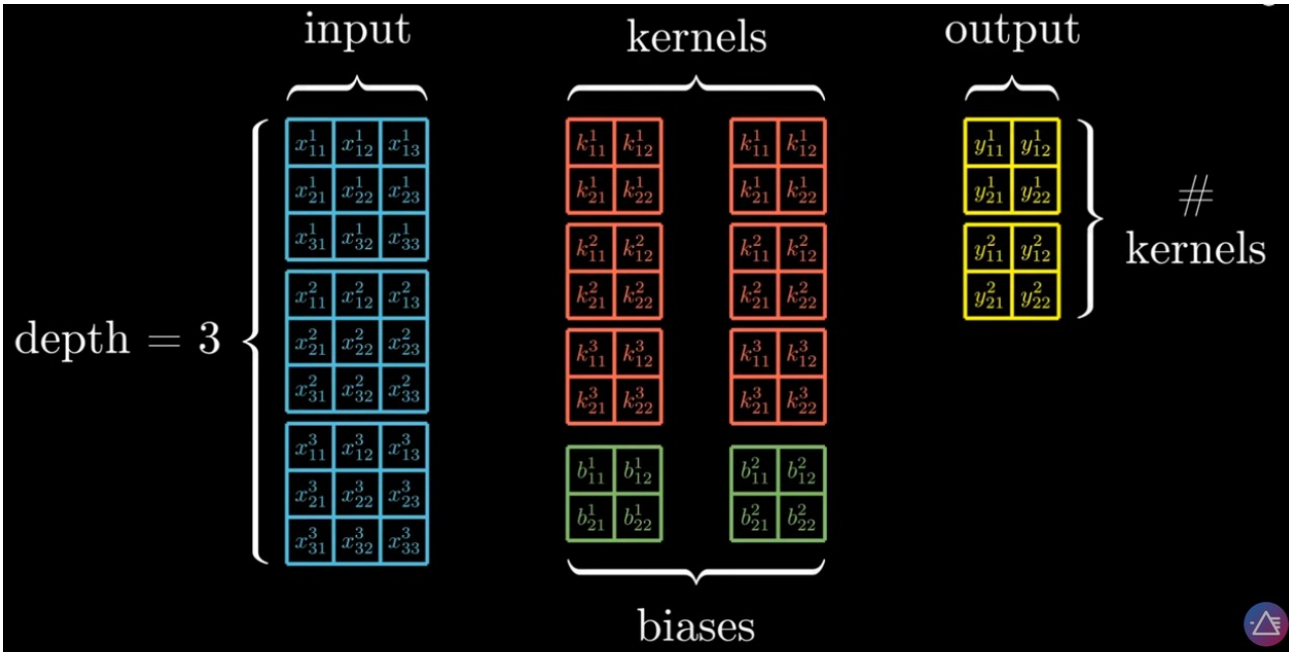


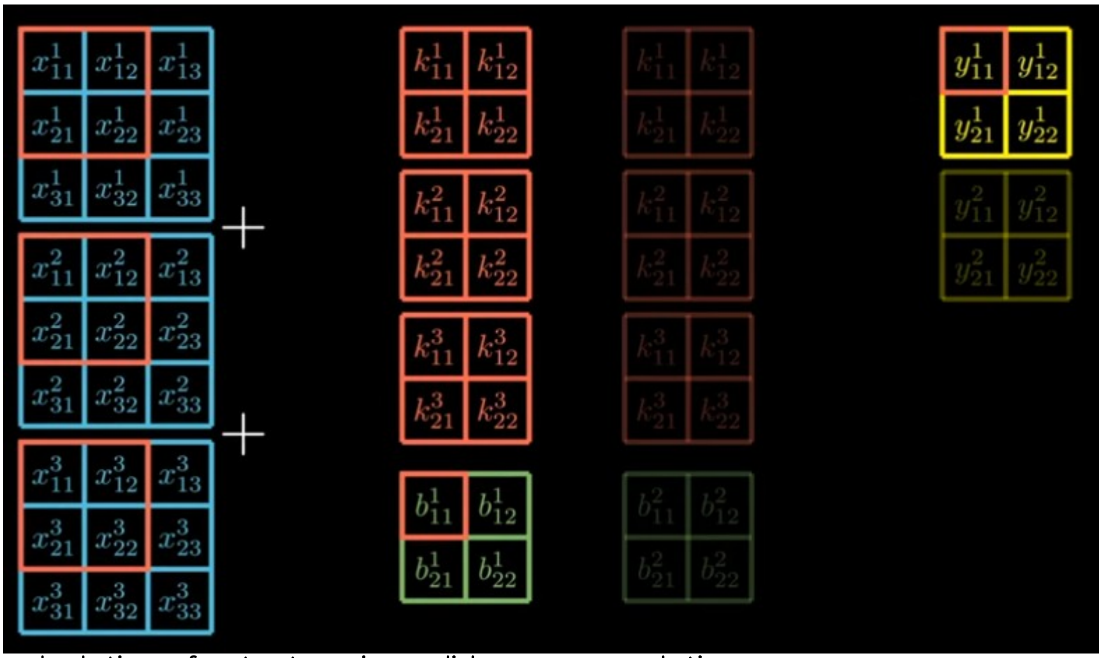
Convolution:



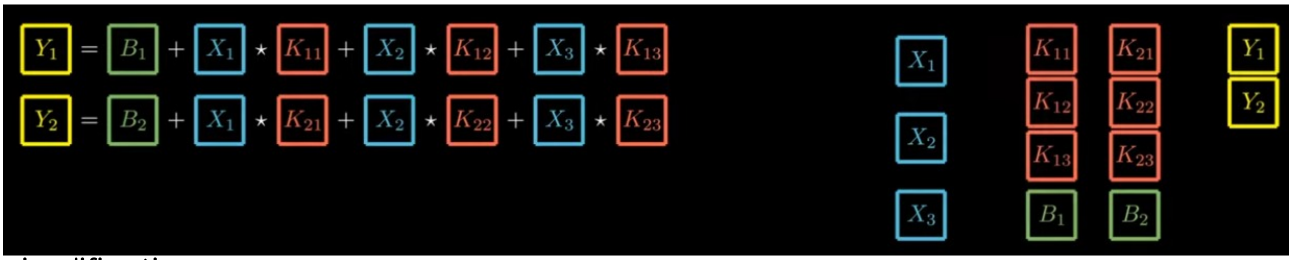


Convolutional Layer:





calculation of outputs using valid cross-correlation



simplification

$$
\begin{aligned}
Y_1 &= B_1 + X_1 \star K_{11} + \cdots + X_n \star K_{1n} \\
Y_2 &= B_2 + X_1 \star K_{21} + \cdots + X_n \star K_{2n} \\
&\vdots \\
Y_d &= B_d + X_1 \star K_{d1} + \cdots + X_n \star K_{dn}
\end{aligned}
$$

generalization with the **d** for the depth of the layer or the depth of the output (corresponding also to the number or kernels) and **n** with the size/depth of the input

Forward Propagation:

$$
Y_i = B_i + \sum_{j=1}^{n} X_j \star K_{ij}, \quad i = 1 \ldots d
$$

```
def forward(self, input_nn):
    self.input_nn = input_nn
    self.output = np.copy(self.biases)
    for i in range(self.depth):
        for j in range(self.input_depth):
            self.output[i] += signal.correlate2d(self.input_nn[j], self.kernels[i, j], mode: "valid")
    return self.output
```

Backward Propagation:

$$\frac{\partial E}{\partial Y_i} \rightarrow \frac{\partial E}{\partial K_{ij}}, \frac{\partial E}{\partial B_i}$$

$$\frac{\partial E}{\partial Y_i} \rightarrow \frac{\partial E}{\partial X_j}$$

gradients

$$Y_i = B_i + \sum_{j=1}^{n} X_j \star K_{ij}$$

Forward propagation

$$Y_i = B_i + X_1 \star K_{i1} + \cdots + X_n \star K_{in}$$

expansion

$$Y_i = B_i + X_1 \star K_{i1}$$

cropping expansion for simplification

Kernel Gradient

$$\begin{cases} y_{11} = b_{11} + k_{11}x_{11} + k_{12}x_{12} + k_{21}x_{21} + k_{22}x_{22} \\ y_{12} = b_{12} + k_{11}x_{12} + k_{12}x_{13} + k_{21}x_{22} + k_{22}x_{23} \\ y_{21} = b_{21} + k_{11}x_{21} + k_{12}x_{22} + k_{21}x_{31} + k_{22}x_{32} \\ y_{22} = b_{22} + k_{11}x_{22} + k_{12}x_{23} + k_{21}x_{32} + k_{22}x_{33} \end{cases}$$

$$\frac{\partial E}{\partial Y} = \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} \\ \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{\partial E}{\partial k_{11}} & \frac{\partial E}{\partial k_{12}} \\ \frac{\partial E}{\partial k_{21}} & \frac{\partial E}{\partial k_{22}} \end{bmatrix}$$

$$\frac{\partial E}{\partial k_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial k_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial k_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial k_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial k_{11}}$$

chain rule

$$\frac{\partial E}{\partial k_{11}} = \frac{\partial E}{\partial y_{11}}\underbrace{\frac{\partial y_{11}}{\partial k_{11}}}_{x_{11}} + \frac{\partial E}{\partial y_{12}}\underbrace{\frac{\partial y_{12}}{\partial k_{11}}}_{x_{12}} + \frac{\partial E}{\partial y_{21}}\underbrace{\frac{\partial y_{21}}{\partial k_{11}}}_{x_{21}} + \frac{\partial E}{\partial y_{22}}\underbrace{\frac{\partial y_{22}}{\partial k_{11}}}_{x_{22}}$$

using 4wd propagation simplification as reference

$$\frac{\partial E}{\partial k_{11}} = \frac{\partial E}{\partial y_{11}}x_{11} + \frac{\partial E}{\partial y_{12}}x_{12} + \frac{\partial E}{\partial y_{21}}x_{21} + \frac{\partial E}{\partial y_{22}}x_{22}$$

$$\frac{\partial E}{\partial k_{12}} = \frac{\partial E}{\partial y_{11}}x_{12} + \frac{\partial E}{\partial y_{12}}x_{13} + \frac{\partial E}{\partial y_{21}}x_{22} + \frac{\partial E}{\partial y_{22}}x_{23}$$

$$\frac{\partial E}{\partial k_{21}} = \frac{\partial E}{\partial y_{11}}x_{21} + \frac{\partial E}{\partial y_{12}}x_{22} + \frac{\partial E}{\partial y_{21}}x_{31} + \frac{\partial E}{\partial y_{22}}x_{32}$$

$$\frac{\partial E}{\partial k_{22}} = \frac{\partial E}{\partial y_{11}}x_{22} + \frac{\partial E}{\partial y_{12}}x_{23} + \frac{\partial E}{\partial y_{21}}x_{32} + \frac{\partial E}{\partial y_{22}}x_{33}$$

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \star \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} \\ \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix}$$

$$\frac{\partial E}{\partial K} = X \star \frac{\partial E}{\partial Y}$$

simplification

$$\boxed{Y = B + X \star K \Rightarrow \frac{\partial E}{\partial K} = X \star \frac{\partial E}{\partial Y}}$$

$$Y_i = B_i + \sum_{j=1}^{n} X_j \star K_{ij}, \quad i = 1 \dots d$$

$$\begin{cases} Y_1 = B_1 + X_1 \star K_{11} + \cdots + X_n \star K_{1n} \\ Y_2 = B_2 + X_1 \star K_{21} + \cdots + X_n \star K_{2n} \\ \vdots \\ Y_d = B_d + X_1 \star K_{d1} + \cdots + X_n \star K_{dn} \end{cases}$$

relation between 4wd propagation and gradient simplification

$$\frac{\partial E}{\partial K_{21}} = X_1 \star \frac{\partial E}{\partial Y_2} \quad \rightarrow \quad \frac{\partial E}{\partial K_{ij}} = X_j \star \frac{\partial E}{\partial Y_i}$$

no chain rule can be applied here, instead the helper (gradient simplification) can be used. Applying the example for K21, it can be seen that only in Y2 the kernel K21 is available. Using this the Kernel gradient can be generalized.

Bias Gradient

$$Y_i = B_i + \sum_{j=1}^{n} X_j \star K_{ij}$$ Forward propagation

$$Y_i = B_i + X_1 \star K_{i1} + \cdots + X_n \star K_{in}$$ expansion

$$Y_i = B_i + X_1 \star K_{i1}$$ cropping expansion for simplification

$$\frac{\partial E}{\partial Y} = \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} \\ \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{\partial E}{\partial b_{11}} & \frac{\partial E}{\partial b_{12}} \\ \frac{\partial E}{\partial b_{21}} & \frac{\partial E}{\partial b_{22}} \end{bmatrix}$$

$$\begin{cases} y_{11} = b_{11} + k_{11}x_{11} + k_{12}x_{12} + k_{21}x_{21} + k_{22}x_{22} \\ y_{12} = b_{12} + k_{11}x_{12} + k_{12}x_{13} + k_{21}x_{22} + k_{22}x_{23} \\ y_{21} = b_{21} + k_{11}x_{21} + k_{12}x_{22} + k_{21}x_{31} + k_{22}x_{32} \\ y_{22} = b_{22} + k_{11}x_{22} + k_{12}x_{23} + k_{21}x_{32} + k_{22}x_{33} \end{cases}$$

$$\frac{\partial E}{\partial b_{11}} = \frac{\partial E}{\partial y_{11}}\boxed{\frac{\partial y_{11}}{\partial b_{11}}} + \frac{\partial E}{\partial y_{12}}\boxed{\frac{\partial y_{12}}{\partial b_{11}}} + \frac{\partial E}{\partial y_{21}}\boxed{\frac{\partial y_{21}}{\partial b_{11}}} + \frac{\partial E}{\partial y_{22}}\boxed{\frac{\partial y_{22}}{\partial b_{11}}}$$

chain rule for simplification

$$\frac{\partial E}{\partial b_{11}} = \frac{\partial E}{\partial y_{11}}$$

$$\frac{\partial E}{\partial b_{12}} = \frac{\partial E}{\partial y_{12}}$$

$$\frac{\partial E}{\partial b_{21}} = \frac{\partial E}{\partial y_{21}}$$

$$\frac{\partial E}{\partial b_{22}} = \frac{\partial E}{\partial y_{22}}$$

$$\frac{\partial E}{\partial B} = \frac{\partial E}{\partial Y}$$

$$\boxed{Y = B + X \star K \Rightarrow \frac{\partial E}{\partial B} = \frac{\partial E}{\partial Y}}$$

simplified bias gradient based upon 4wd propagation

$$Y_i = B_i + \sum_{j=1}^{n} X_j \star K_{ij}, \quad i = 1 \dots d$$

$$\begin{cases} Y_1 = B_1 + X_1 \star K_{11} + \cdots + X_n \star K_{1n} \\ Y_2 = B_2 + X_1 \star K_{21} + \cdots + X_n \star K_{2n} \\ \vdots \\ Y_d = B_d + X_1 \star K_{d1} + \cdots + X_n \star K_{dn} \end{cases}$$

actual equation for 4wd propagation

$$\boxed{\frac{\partial E}{\partial B_1} = \frac{\partial E}{\partial Y_1} \quad \longrightarrow \quad \frac{\partial E}{\partial B_i} = \frac{\partial E}{\partial Y_i}}$$

bias gradient calculated based upon generalization

Input Gradient

$$Y_i = B_i + \sum_{j=1}^{n} X_j \star K_{ij}$$

Forward propagation

$$Y_i = B_i + X_1 \star K_{i1} + \cdots + X_n \star K_{in}$$

expansion

$$Y_i = B_i + X_1 \star K_{i1}$$

cropping expansion for simplification

$$\begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} + \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \star \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}$$

$$y_{11} = b_{11} + k_{11}x_{11} + k_{12}x_{12} + k_{21}x_{21} + k_{22}x_{22}$$
$$y_{12} = b_{12} + k_{11}x_{12} + k_{12}x_{13} + k_{21}x_{22} + k_{22}x_{23}$$
$$y_{21} = b_{21} + k_{11}x_{21} + k_{12}x_{22} + k_{21}x_{31} + k_{22}x_{32}$$
$$y_{22} = b_{22} + k_{11}x_{22} + k_{12}x_{23} + k_{21}x_{32} + k_{22}x_{33}$$

$$\frac{\partial E}{\partial Y} = \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} \\ \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{\partial E}{\partial x_{11}} & \frac{\partial E}{\partial x_{12}} & \frac{\partial E}{\partial x_{13}} \\ \frac{\partial E}{\partial x_{21}} & \frac{\partial E}{\partial x_{22}} & \frac{\partial E}{\partial x_{23}} \\ \frac{\partial E}{\partial x_{31}} & \frac{\partial E}{\partial x_{32}} & \frac{\partial E}{\partial x_{33}} \end{bmatrix}$$

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}} \underbrace{\frac{\partial y_{11}}{\partial x_{11}}}_{k_{11}} + \frac{\partial E}{\partial y_{12}} \underbrace{\frac{\partial y_{12}}{\partial x_{11}}}_{0} + \frac{\partial E}{\partial y_{21}} \underbrace{\frac{\partial y_{21}}{\partial x_{11}}}_{0} + \frac{\partial E}{\partial y_{22}} \underbrace{\frac{\partial y_{22}}{\partial x_{11}}}_{0} \quad \begin{cases} y_{11} = b_{11} + k_{11}x_{11} + k_{12}x_{12} + k_{21}x_{21} + k_{22}x_{22} \\ y_{12} = b_{12} + k_{11}x_{12} + k_{12}x_{13} + k_{21}x_{22} + k_{22}x_{23} \\ y_{21} = b_{21} + k_{11}x_{21} + k_{12}x_{22} + k_{21}x_{31} + k_{22}x_{32} \\ y_{22} = b_{22} + k_{11}x_{22} + k_{12}x_{23} + k_{21}x_{32} + k_{22}x_{33} \end{cases}$$

starting for instance with x11 and using the chain rule

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}} k_{11}$$

$$\frac{\partial E}{\partial x_{12}} = \frac{\partial E}{\partial y_{11}} k_{12} + \frac{\partial E}{\partial y_{12}} k_{11}$$

$$\frac{\partial E}{\partial x_{13}} = \frac{\partial E}{\partial y_{12}} k_{12}$$

$$\frac{\partial E}{\partial x_{21}} = \frac{\partial E}{\partial y_{11}} k_{21} + \frac{\partial E}{\partial y_{21}} k_{11}$$

$$\frac{\partial E}{\partial x_{22}} = \frac{\partial E}{\partial y_{11}} k_{22} + \frac{\partial E}{\partial y_{12}} k_{21} + \frac{\partial E}{\partial y_{21}} k_{12} + \frac{\partial E}{\partial y_{22}} k_{11}$$

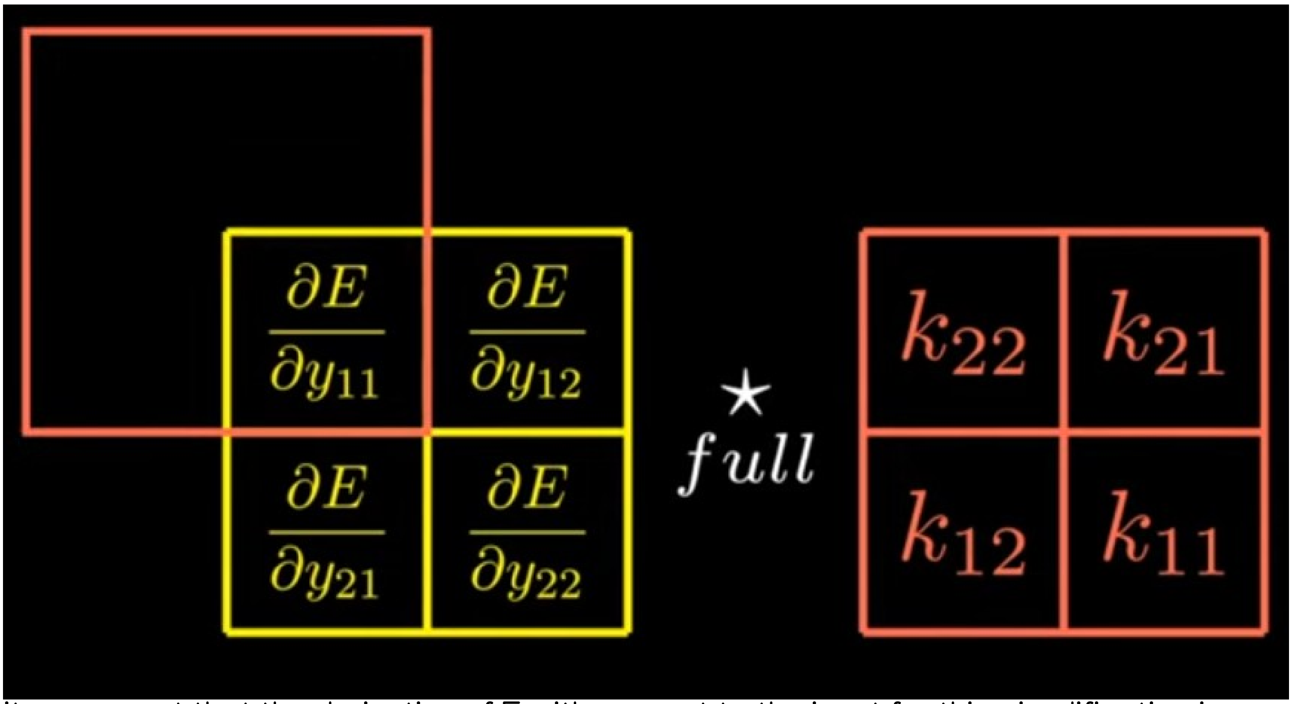
$$\frac{\partial E}{\partial x_{23}} = \frac{\partial E}{\partial y_{12}} k_{22} + \frac{\partial E}{\partial y_{22}} k_{12}$$

$$\frac{\partial E}{\partial x_{31}} = \frac{\partial E}{\partial y_{21}} k_{21}$$

$$\frac{\partial E}{\partial x_{32}} = \frac{\partial E}{\partial y_{21}} k_{22} + \frac{\partial E}{\partial y_{22}} k_{21}$$

$$\frac{\partial E}{\partial x_{33}} = \frac{\partial E}{\partial y_{22}} k_{22}$$

expanding it to all x values of this simplification



it comes out that the derivative of E with respect to the input for this simplification is equal to the derivative of E with respect to the output fully cross-correlated with the kernel rotated by 180°

$$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} \overset{\star}{full}\; rot180(K)$$

$$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} \overset{*}{full}\; K$$

the input gradient is equal to the output gradient fully convolved with the kernel

$$\boxed{Y = B + X \star K \Rightarrow \frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} \overset{*}{full}\; K}$$

the simplified version

$$Y_i = B_i + \sum_{j=1}^{n} X_j \star K_{ij}, \quad i = 1 \dots d$$

$$\begin{cases} Y_1 = B_1 + X_1 \star K_{11} + \cdots + X_n \star K_{1n} \\ Y_2 = B_2 + X_1 \star K_{21} + \cdots + X_n \star K_{2n} \\ \vdots \\ Y_d = B_d + X_1 \star K_{d1} + \cdots + X_n \star K_{dn} \end{cases}$$

actual equation for 4wd propagation

$$\frac{\partial E}{\partial X_1} = \frac{\partial E}{\partial Y_1} \overset{*}{full} K_{11} + \cdots + \frac{\partial E}{\partial Y_d} \overset{*}{full} K_{d1}$$

$$\frac{\partial E}{\partial X_j} = \frac{\partial E}{\partial Y_1} \overset{*}{full} K_{1j} + \cdots + \frac{\partial E}{\partial Y_d} \overset{*}{full} K_{dj}$$

$$\boxed{\frac{\partial E}{\partial X_j} = \sum_{i=1}^{d} \frac{\partial E}{\partial Y_i} \overset{*}{full} K_{ij}, \quad j = 1 \dots n}$$

$$\frac{\partial E}{\partial K_{ij}} = X_j \star \frac{\partial E}{\partial Y_i}$$

$$\frac{\partial E}{\partial B_i} = \frac{\partial E}{\partial Y_i}$$

$$\frac{\partial E}{\partial X_j} = \sum_{i=1}^{n} \frac{\partial E}{\partial Y_i} \underset{full}{*} K_{ij}$$

summary of all gradients

```python
def backward(self, output_gradient, learning_rate):
    kernels_gradient = np.zeros(self.kernels_shape)
    input_gradient = np.zeros(self.input_shape)

    for i in range(self.depth):
        for j in range(self.input_depth):
            kernels_gradient[i, j] = signal.correlate2d(self.input_nn[j], output_gradient[i], mode: "valid")
            input_gradient[j] += signal.convolve2d(output_gradient[i], self.kernels[i, j], mode: "full")

    self.kernels -= learning_rate * kernels_gradient  # gradient descent
    self.biases -= learning_rate * output_gradient  # gradient descent
    return input_gradient
```

Reshape Layer:

This layer is needed because the output of the convolutional layer is a 3d block. Typically dense layers are used at the output of a neural network. Dense layers use column vectors as inputs

```
1 usage (1 dynamic)
def forward(self, input_nn):
    return np.reshape(input_nn, self.output_shape)

1 usage (1 dynamic)
def backward(self, output_gradient, learning_rate):
    return np.reshape(output_gradient, self.input_shape)
```

Binary Cross Entropy Loss:

This is needed because the classification will be performed using MNIST database for 2 digits (0 and 1) for the sake of computational effort → Binary Classification

$$Y^* = \begin{bmatrix} y_1^* \\ y_2^* \\ \vdots \\ y_i^* \end{bmatrix}$$

$$y_i^* \in \{0, 1\}$$

desired output vector

$$Y^* = \begin{bmatrix} y_1^* \\ y_2^* \\ \vdots \\ y_i^* \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{bmatrix}$$

$$E = -\frac{1}{n}\sum_{i=1}^{n} y_i^* log(y_i) + (1 - y_i^*)log(1 - y_i)$$

binary entropy loss considering desired output (y*) and actual output (y)

```
def binary_cross_entropy(y_true, y_pred):
    return np.mean(-y_true * np.log(y_pred) - (1 - y_true) * np.log(1 - y_pred))
```

$$E = -\frac{1}{n}\sum_{i=1}^{n} y_i^* log(y_i) + (1 - y_i^*) log(1 - y_i)$$

$$\frac{\partial E}{\partial Y} = \begin{bmatrix} \frac{\partial E}{\partial y_1} \\ \frac{\partial E}{\partial y_2} \\ \vdots \\ \frac{\partial E}{\partial y_i} \end{bmatrix}$$

$$\begin{aligned}
\frac{\partial E}{\partial y_1} &= \frac{\partial}{\partial y_1}\left(-\frac{1}{n}\sum_{i=1}^{n} y_i^* log(y_i) + (1 - y_i^*) log(1 - y_i)\right) \\
&= \frac{\partial}{\partial y_1}\left(-\frac{1}{n}(y_1^* log(y_1) + (1 - y_1^*) log(1 - y_1)) - \cdots - \frac{1}{n}(y_n^* log(y_n) + (1 - y_n^*) log(1 - y_n))\right) \\
&= \frac{\partial}{\partial y_1}\left(-\frac{1}{n}(y_1^* log(y_1) + (1 - y_1^*) log(1 - y_1))\right)
\end{aligned}$$

Given:

$$f(y_{\text{actual}}) = -\frac{1}{n}\left(y_{\text{desired}} \cdot \log(y_{\text{actual}}) + (1 - y_{\text{desired}}) \cdot \log(1 - y_{\text{actual}})\right)$$

We want to find:

$$\frac{df}{dy_{\text{actual}}}$$

Using the chain rule and the derivative of $\log(x)$:

$$f'(y_{\text{actual}}) = -\frac{1}{n}\left(y_{\text{desired}} \cdot \frac{1}{y_{\text{actual}}} + (1 - y_{\text{desired}}) \cdot \frac{-1}{1 - y_{\text{actual}}}\right)$$

$$= -\frac{1}{n}\left(\frac{y_{\text{desired}}}{y_{\text{actual}}} - \frac{1 - y_{\text{desired}}}{1 - y_{\text{actual}}}\right)$$

calculating the output gradient using binary cross entropy for instance for y_1

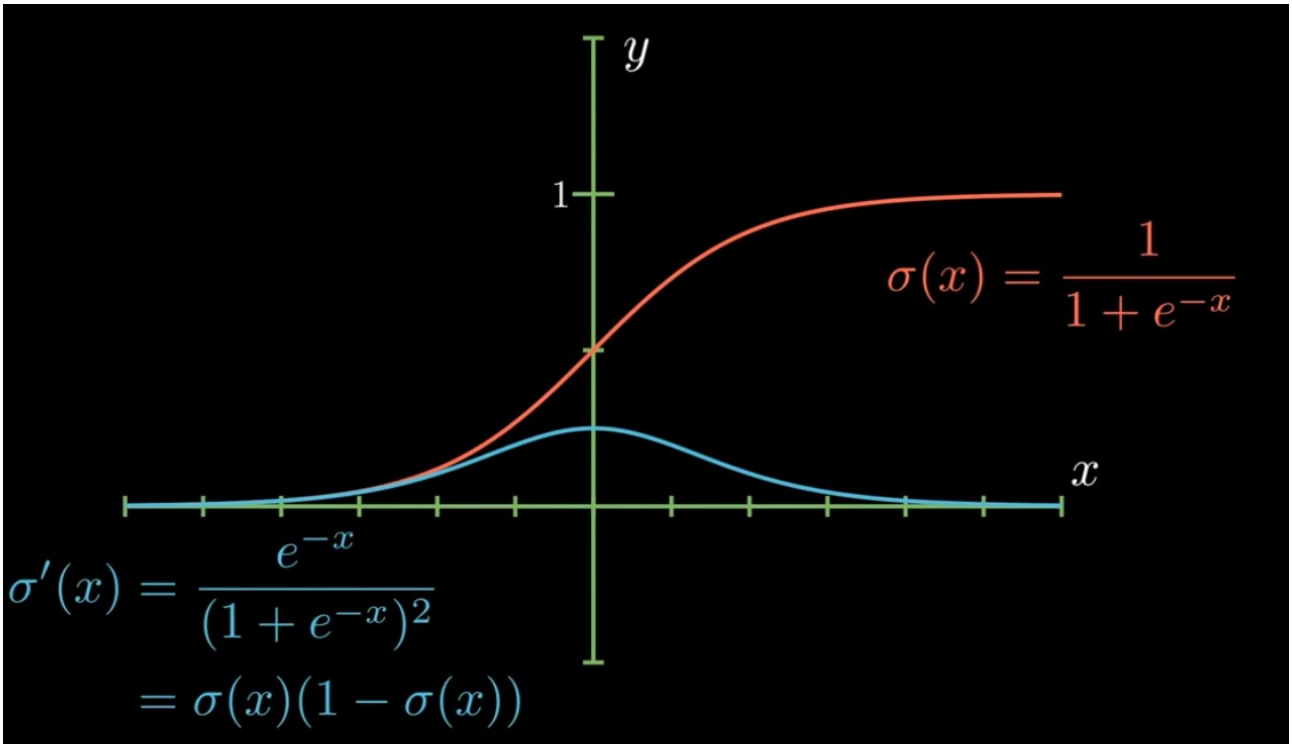
$$\frac{\partial E}{\partial y_1} = \frac{1}{n}\left(\frac{1 - y_1^*}{1 - y_1} - \frac{y_1^*}{y_1}\right) \quad \rightarrow \quad \frac{\partial E}{\partial y_i} = \frac{1}{n}\left(\frac{1 - y_i^*}{1 - y_i} - \frac{y_i^*}{y_i}\right)$$

generalization for y_i

```python
def binary_cross_entropy_prime(y_true, y_pred):
    return ((1 - y_true) / (1 - y_pred) - y_true / y_pred) / np.size(y_true)
```

Sigmoid Activation:

The binary cross entropy uses the logarithmic function. It does not take negative inputs. The output of a sigmoid function is bounded between 0 and 1.




```python
class Sigmoid(Activation):
    def __init__(self):
        def sigmoid(x):
            return 1 / (1 + np.exp(-x))

        def sigmoid_prime(x):
            s = sigmoid(x)
            return s * (1 - s)

        super().__init__(sigmoid, sigmoid_prime)
```