Open in app ↗ Sign up Sign In

Search Medium

Pavithra Solai Follow

Mar 19, 2018 · 8 min read · Listen

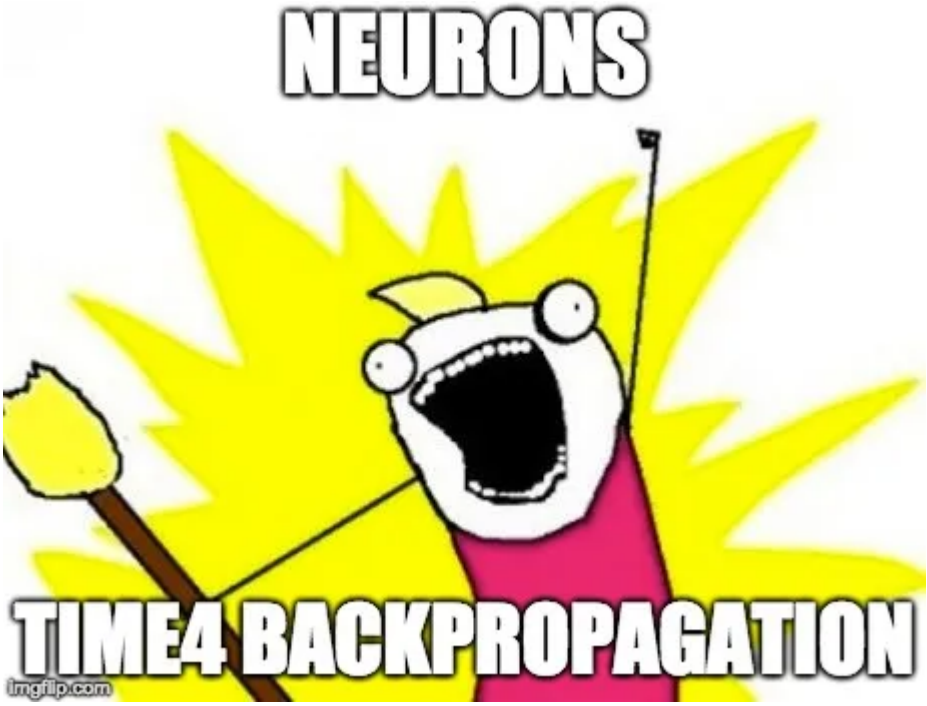
Save

Convolutions and Backpropagations

Ever since AlexNet won the ImageNet competition in 2012, Convolutional Neural Networks (CNNs) have become ubiquitous. Starting from the humble LeNet to ResNets to DenseNets, CNNs are everywhere.

But have you ever wondered what happens in a Backward pass of a CNN, especially how Backpropagation works in a CNN. If you have read about Backpropagation, you would have seen how it is implemented in a simple Neural Network with Fully Connected layers. (*Andrew Ng's course on Coursera does a great job of explaining it*). But, for the life of me, I couldn't wrap my head around how Backpropagation works with Convolutional layers.

4.8K 38



The more I dug through the articles related to CNNs and Backpropagation, the more confused I got. Explanations were mired in complex derivations and notations and they needed an extra-mathematical muscle to understand it. And I was getting nowhere.

> *I know, you don't have to know the mathematical intricacies of a Backpropagation to implement CNNs. You don't have to implement them by hand. And hence, most of the Deep Learning Books don't cover it either.*

So when I finally figured it out, I decided to write this article. To simplify and demystify it. Of course, it would be great if you understand the basics of Backpropagation to follow this article.

STATUTORY WARNING:

It would be better if you have read about Backpropagations and the use of Derivatives in it

The most important thing about this article is to show you this:

> We all know the forward pass of a Convolutional layer uses Convolutions. But, the backward pass during Backpropagation also uses Convolutions!

So, let us dig in and start with understanding the intuition behind Backpropagation. (*And for this, we are going to rely on Andrej Karpathy's amazing CS231n lecture — https://www.youtube.com/watch?v=i94OvYb6noo*).

But if you are already aware of the chain rule in Backpropagation, then you can skip to the next section.

Understanding Chain Rule in Backpropagation:

Consider this equation

$$f(x,y,z) = (x + y)z$$

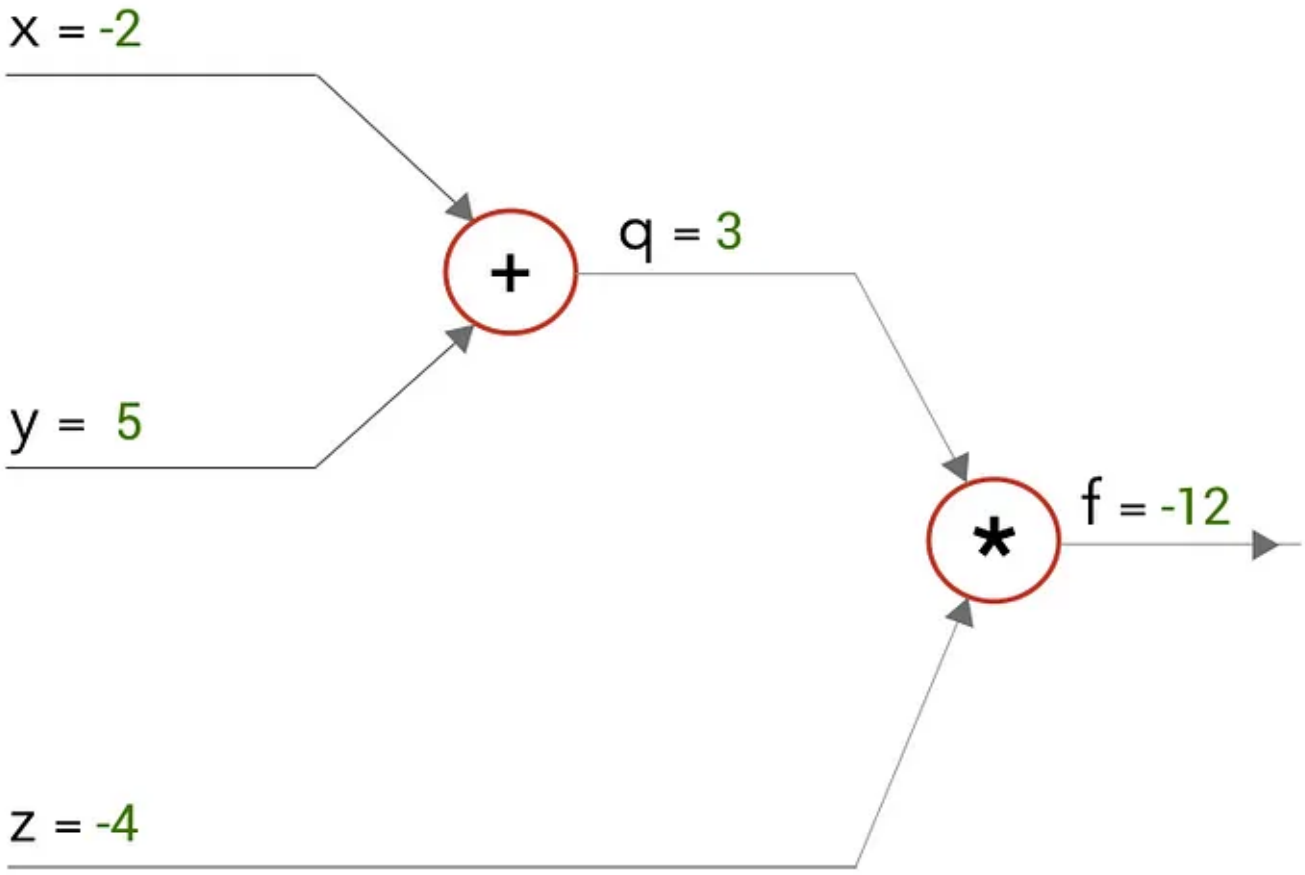
To make it simpler, let us split it into two equations.

$$f(x,y,z) = (x+y)z$$

$$q = x + y$$

$$f = q * z$$

Now, let us draw a computational graph for it with values of x, y, z as **x = -2, y = 5, z = 4.**

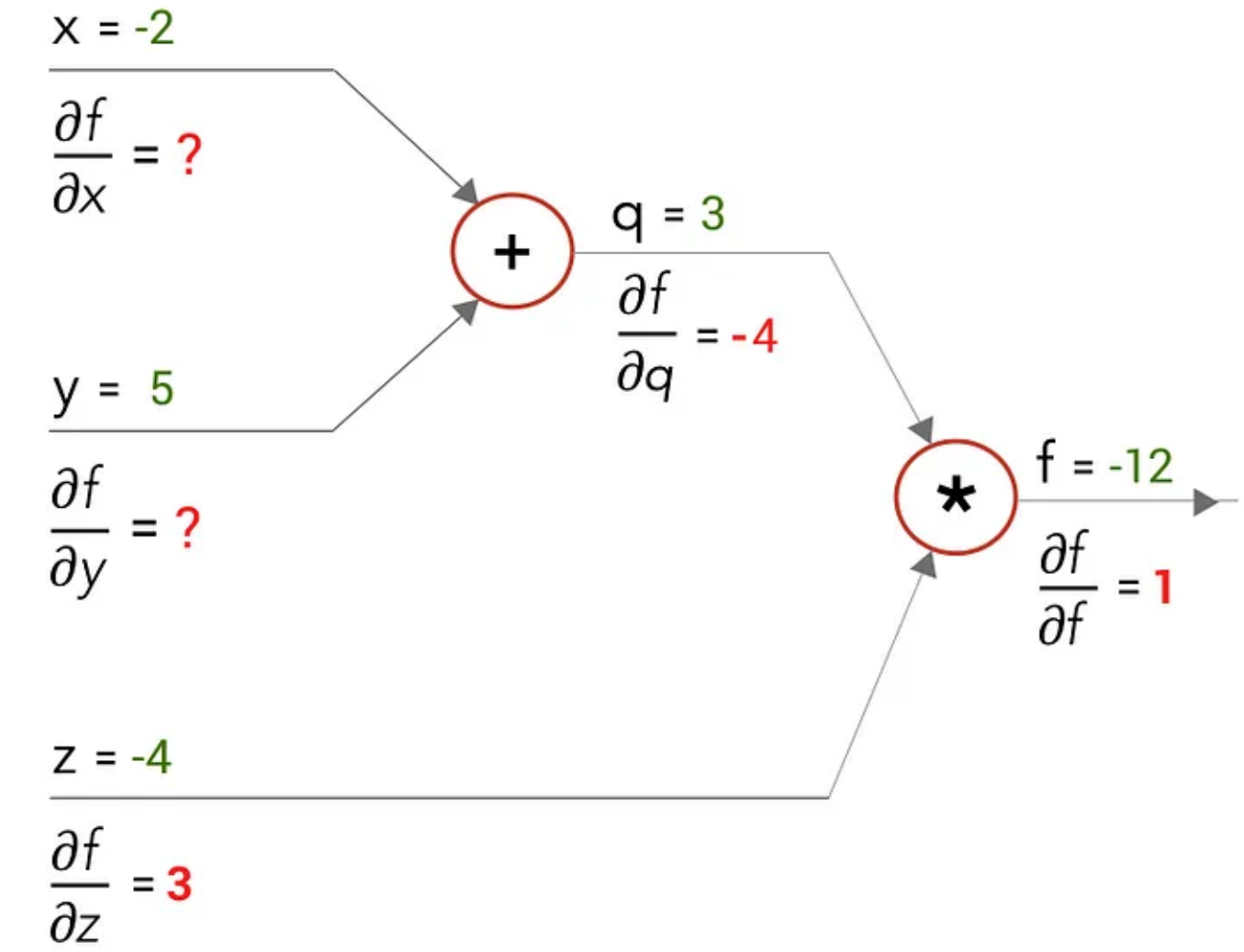


Computational Graph of f = q*z where q = x + y

When we solve for the equations, as we move from left to right, ('the forward pass'), we get an output of **f = -12**

Now let us do the backward pass. Say, just like in Backpropagations, we derive the gradients moving from right to left at each stage. So, at the end, we have to get the values of the gradients of our inputs x,y and z — **∂f/∂x** *and* **∂f/∂y** *and* **∂f/∂z** (*differentiating function f in terms of x,y and z)*

Working from right to left, at the multiply gate we can differentiate *f* to get the gradients at *q* and *z* — **∂f/∂q** and **∂f/∂z .** And at the add gate, we can differentiate *q* to get the gradients at *x* and *y* — **∂q/∂x** and **∂q/∂y.**



$$f = q * z$$

$$\frac{\partial f}{\partial q} = z \mid z = -4$$

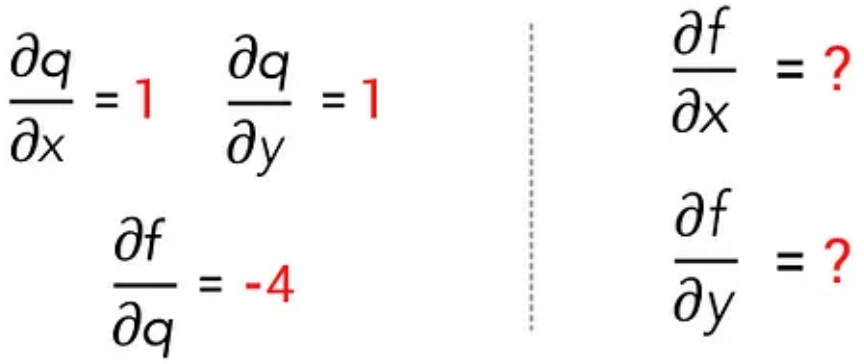
$$\frac{\partial f}{\partial z} = q \mid q = 3$$

$$q = x + y$$

$$\frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

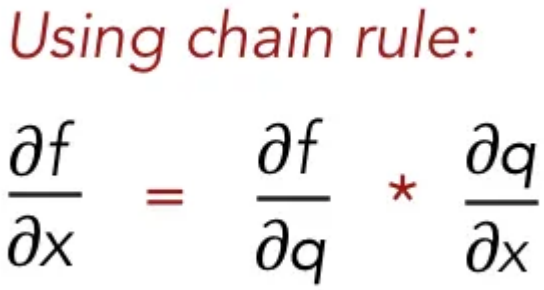
Calculating gradients and their values in the computational graph

We have to find **∂f/∂x** and **∂f/∂y** but we only have got the values of **∂q/∂x** and **∂q/∂y.** So, how do we go about it?



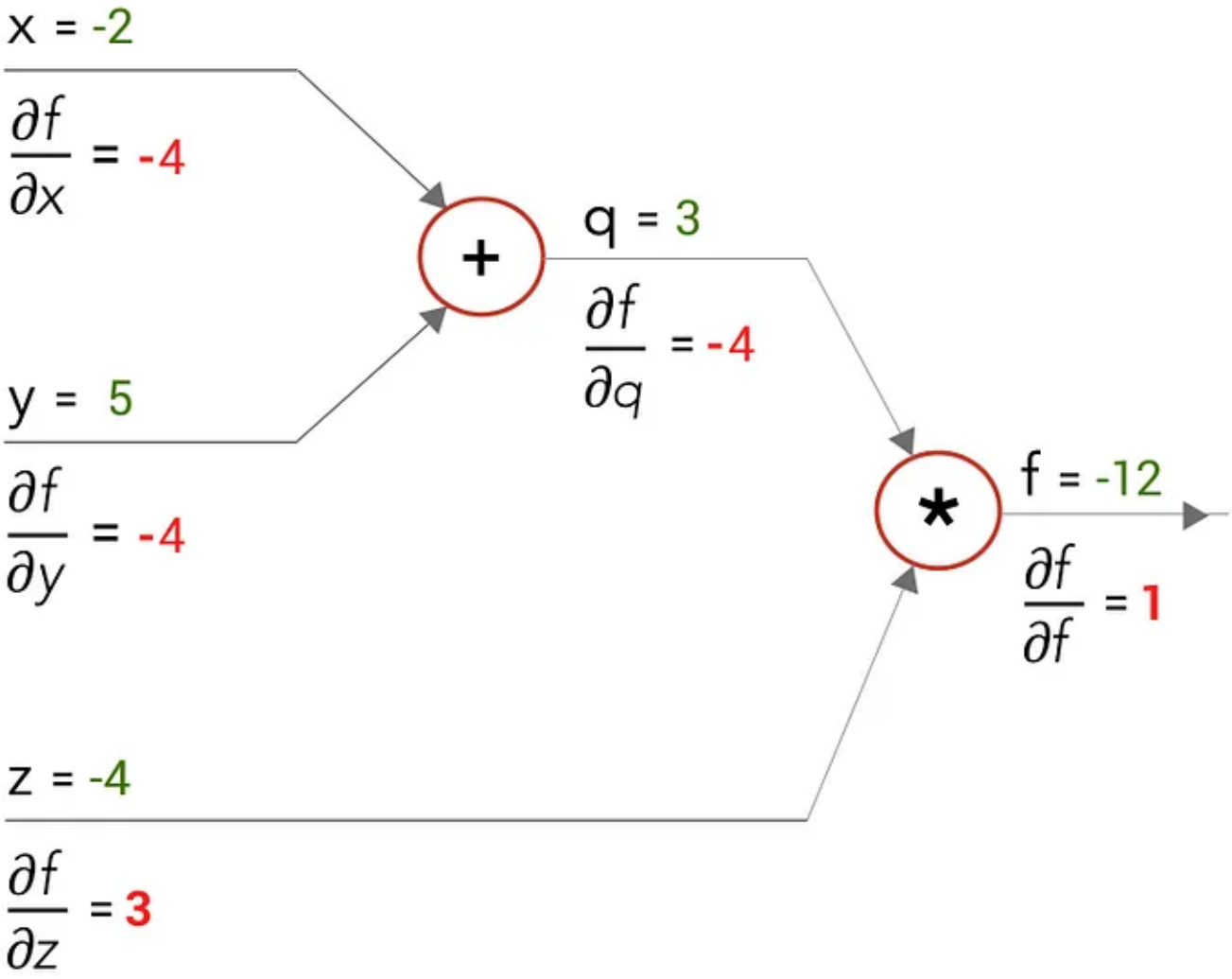
How do we find **∂f/∂x** and **∂f/∂y**

This can be done using the chain rule of differentiation. By the chain rule, we can find **∂f/∂x** as



Chain rule of Differentiation

And we can calculate **∂f/∂x** and **∂f/∂y** as:



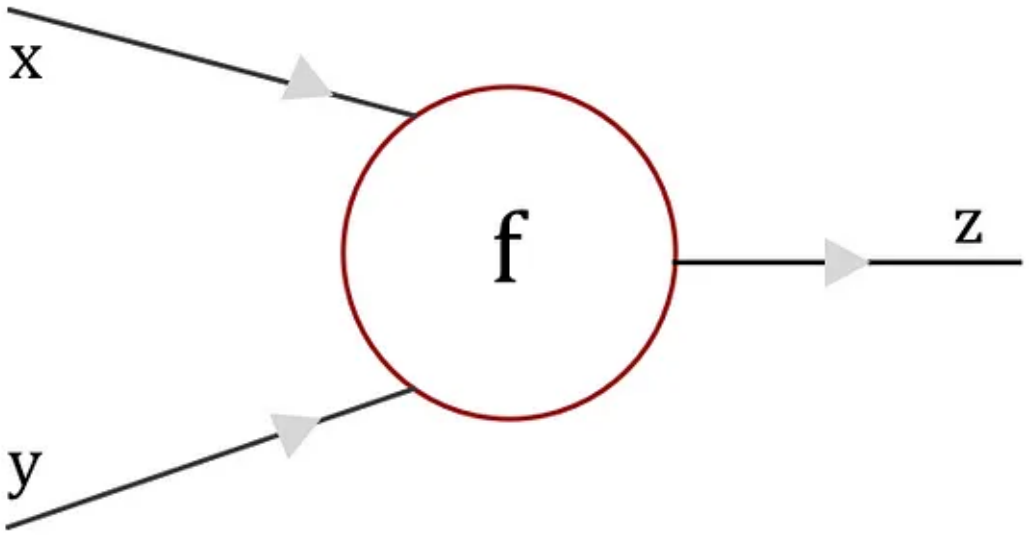
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial x} = -4 * 1 = -4$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial y} = -4 * 1 = -4$$

Backward pass of the Computational graph with all the gradients

Chain Rule in a Convolutional Layer

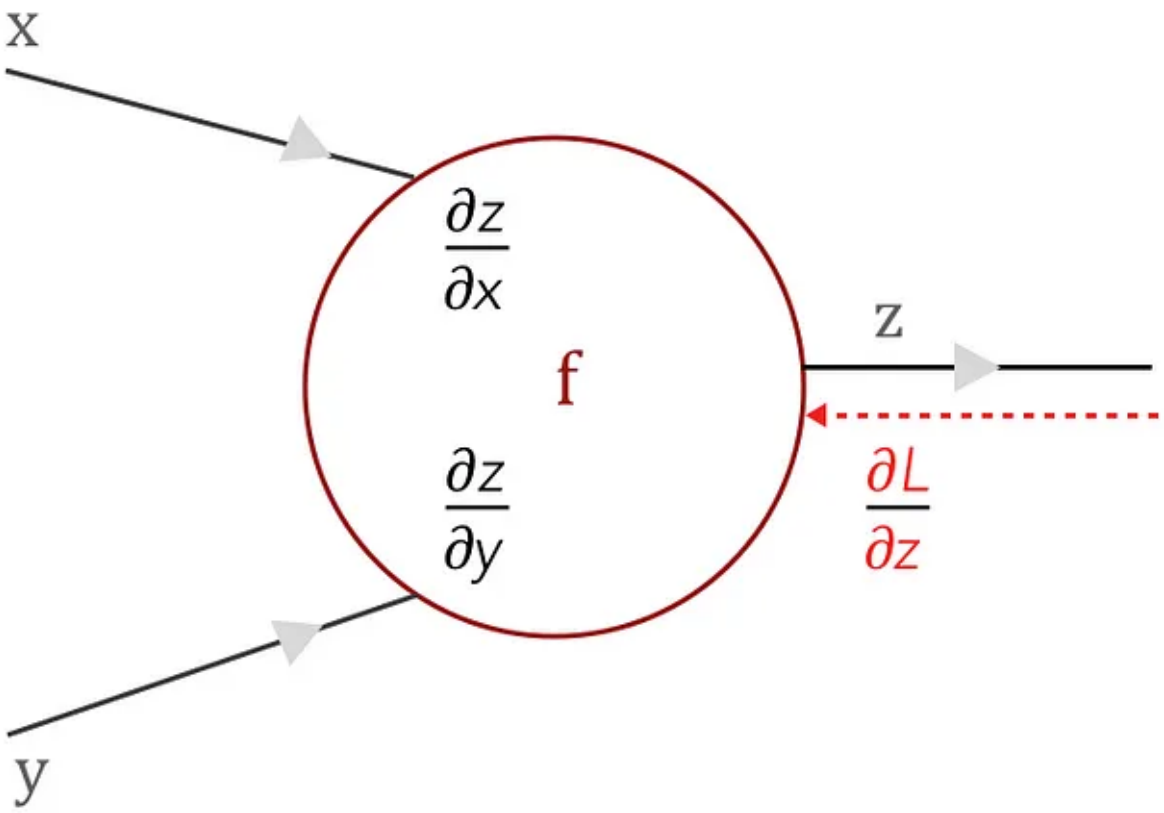
Now that we have worked through a simple computational graph, we can imagine a CNN as a massive computational graph. Let us say we have a gate ***f*** in that computational graph *with inputs* ***x and y*** *which outputs* ***z.***



A simple function f which takes x and y as inputs and outputs z

We can easily compute the *local gradients — differentiating z with respect to x and y* as **∂z/∂x** and **∂z/∂y**

For the forward pass, we move across the CNN, moving through its layers and at the end obtain the loss, using the loss function. And when we start to work the loss backwards, layer across layer, we get the gradient of the loss from the previous layer as **∂L/∂z.** In order for the loss to be propagated to the other gates, we need to find **∂L/∂x** and **∂L/∂y**.

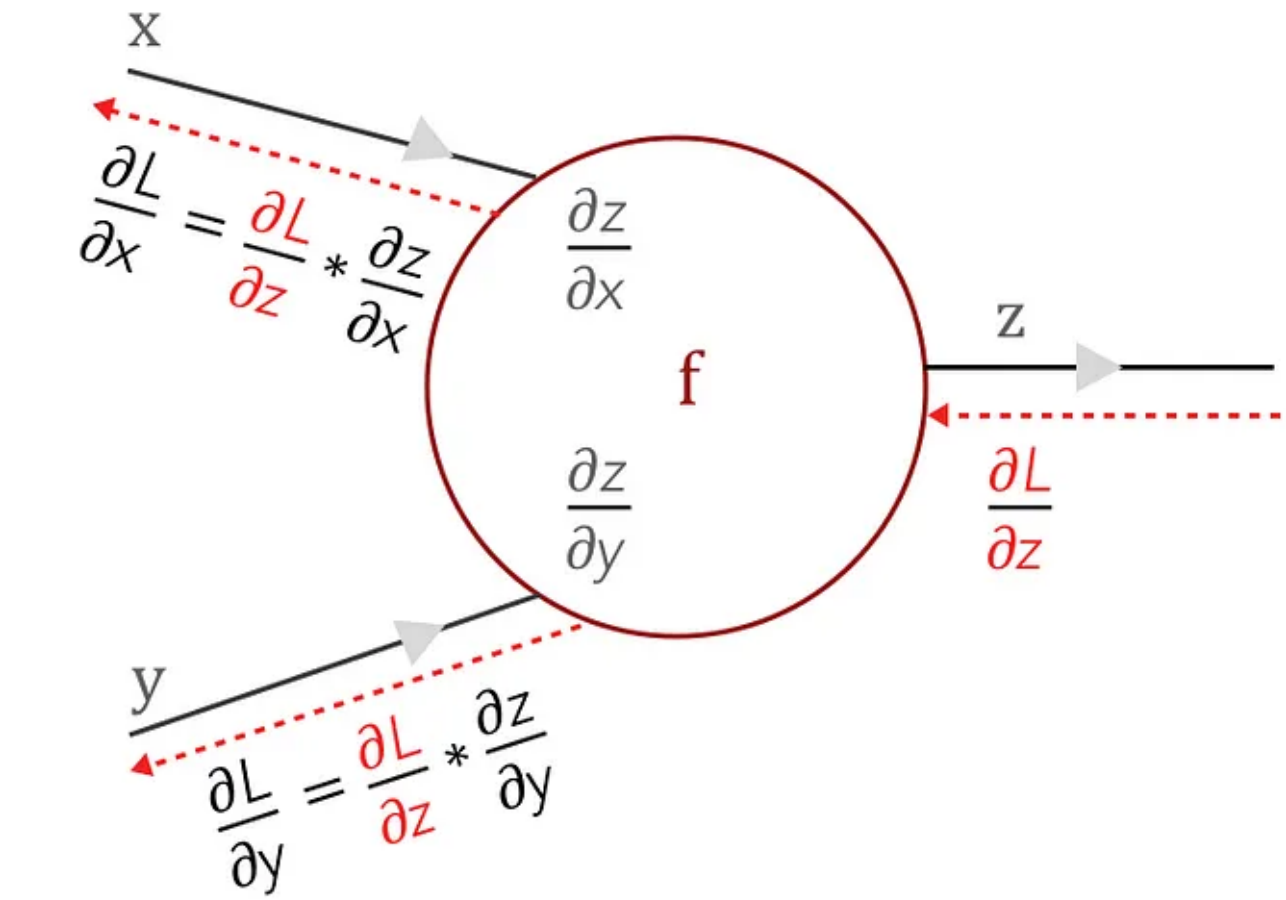


$\frac{\partial z}{\partial x}$ & $\frac{\partial z}{\partial y}$ *are local gradients*

$\frac{\partial L}{\partial z}$ *is the loss from the previous layer which has to be backpropagated to other layers*

Local gradients can be computed using the function f. Now, we need to find **∂L/∂x** and **∂L/∂y,** as it needs to be propagated to other layers.

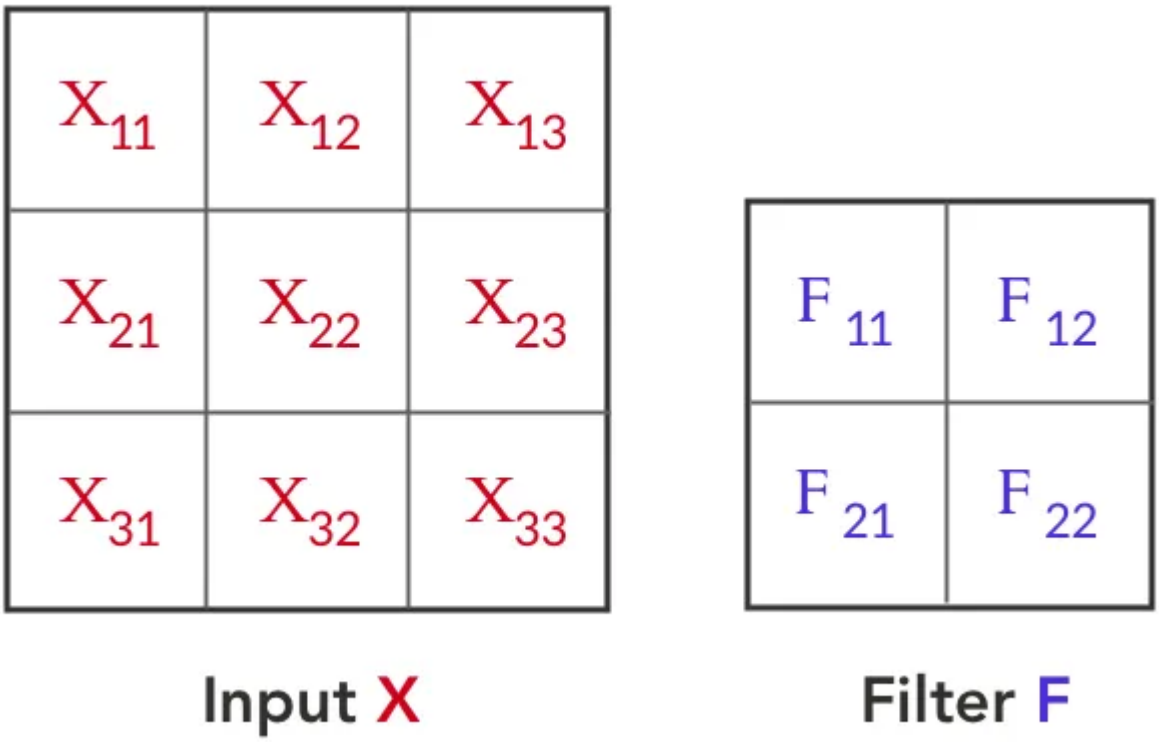
The chain rule comes to our help. Using the chain rule we can calculate **∂L/∂x** and **∂L/∂y,** which would feed the other gates in the extended computational graph



Finding the loss gradients for x and y

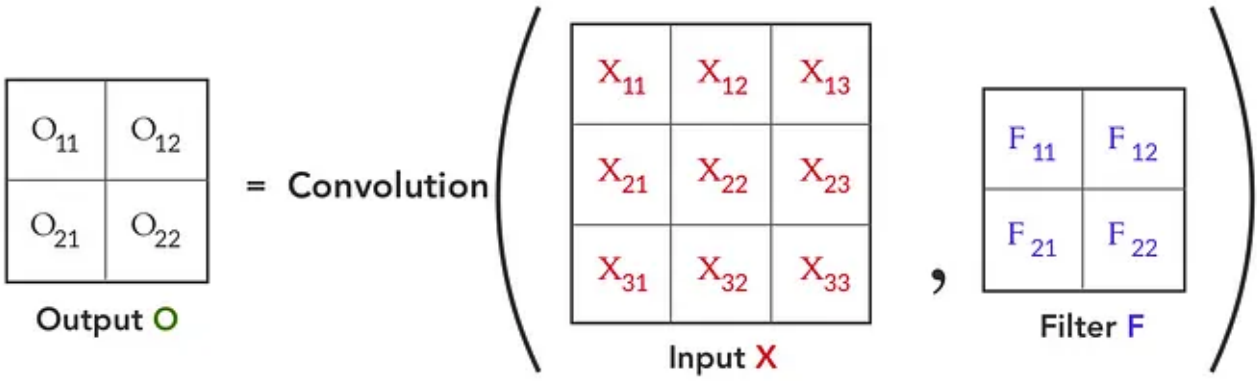
So, what has this got to do with Backpropagation in the Convolutional layer of a CNN?

Now, lets assume the function ***f is a convolution*** *between* **Input X and a Filter F.** Input X is a 3x3 matrix and Filter F is a 2x2 matrix, as shown below:

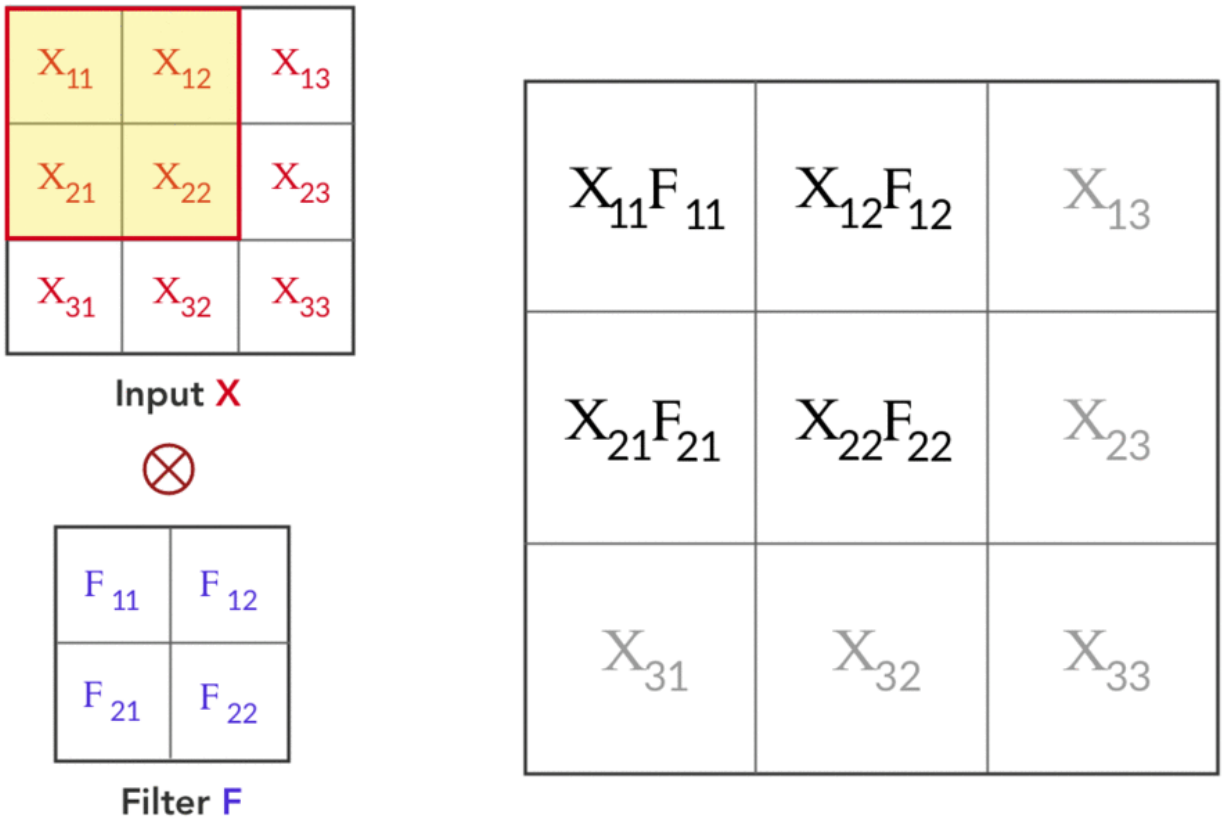


A simple Convolutional Layer example with Input X and Filter F

Convolution between Input X and Filter F, gives us an output O. This can be represented as:



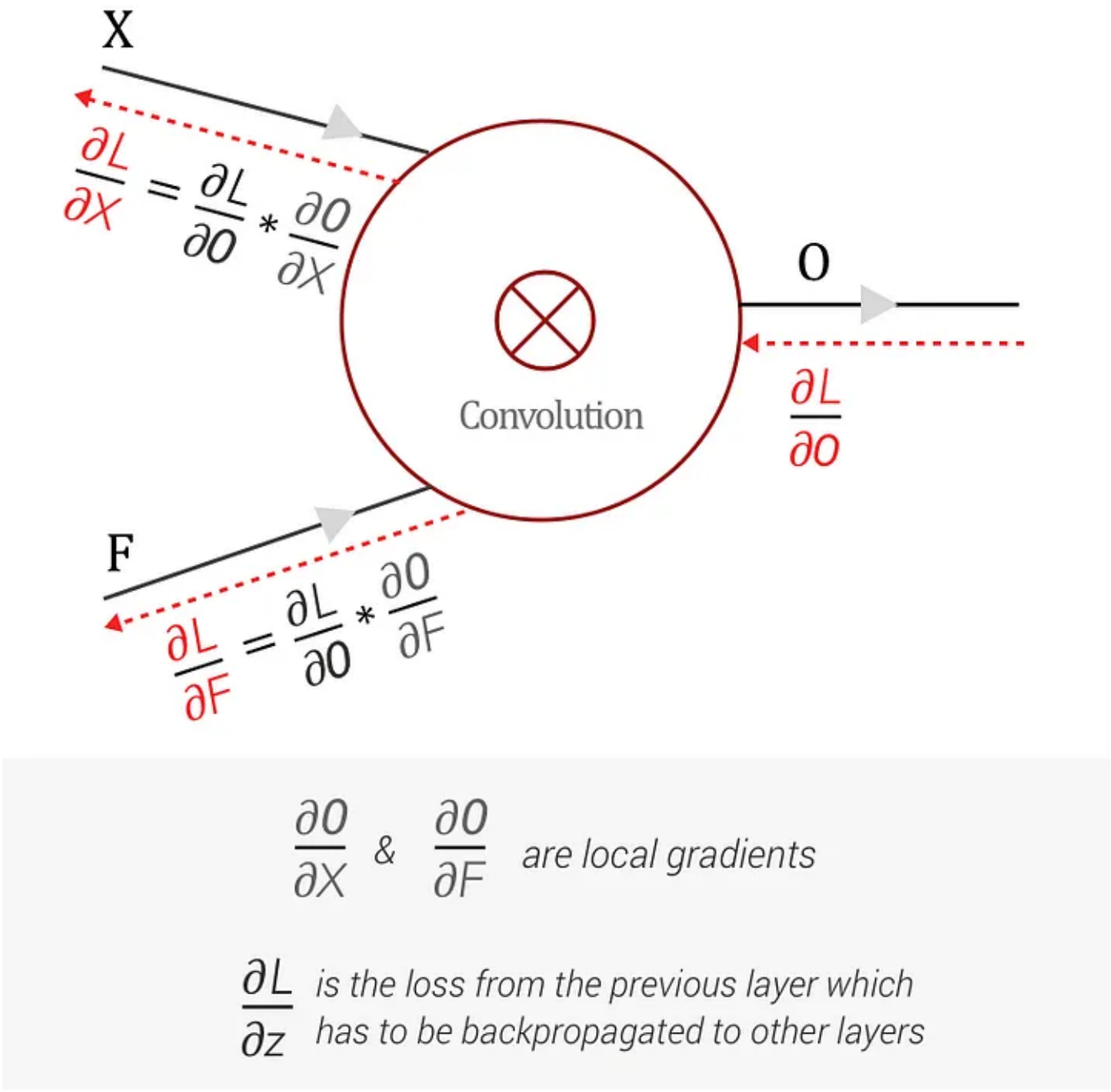
Convolution Function between X and F, gives Output O



$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

Convolution operation giving us values of the Output O

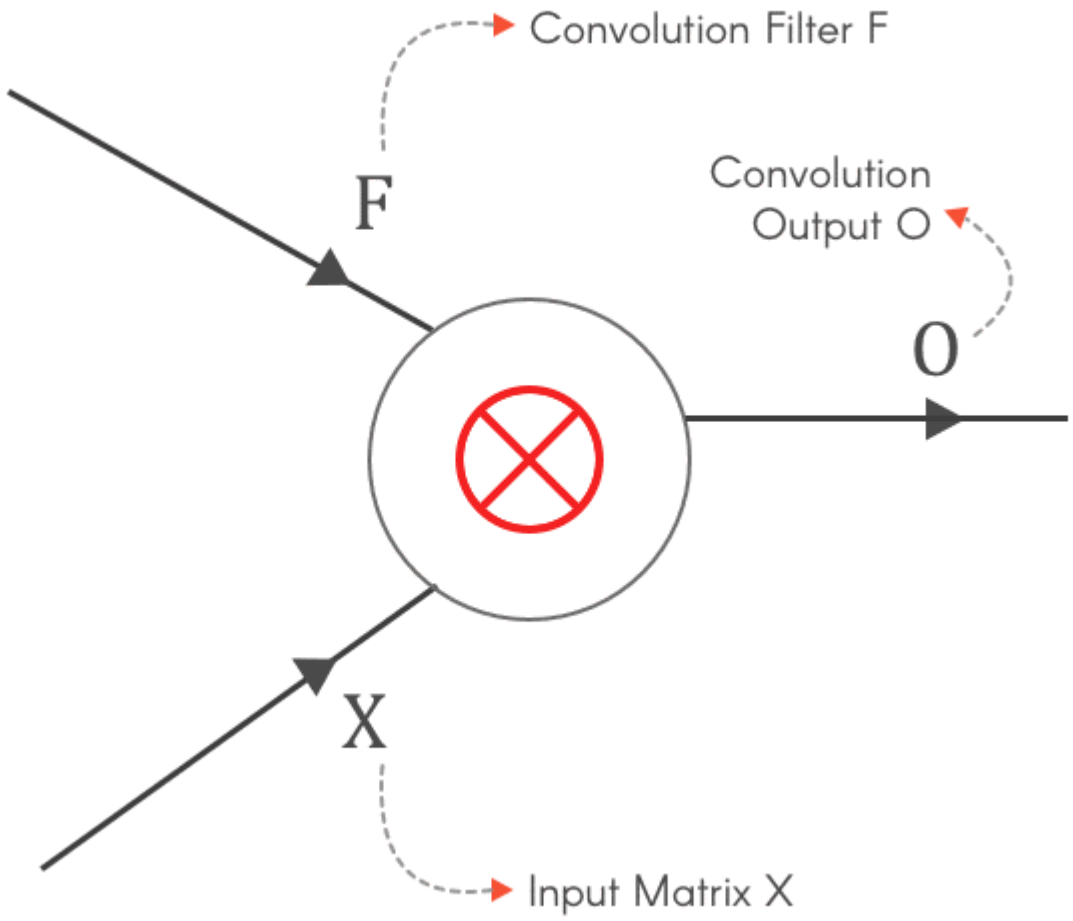
This gives us the forward pass! Let's get to the Backward pass. As mentioned earlier, we get the loss gradient with respect to the Output O from the next layer as $\partial L/\partial O$, during Backward pass. And combining with our previous knowledge using Chain rule and Backpropagation we get:



Function f during Backward pass

As seen above, we can find the local gradients **∂O/∂X** and **∂O/∂F** with respect to Output O. And with loss gradient from previous layers — **∂L/∂O** and using chain rule, we can calculate **∂L/∂X** and **∂L/∂F**.

Well, but why do we need to find ∂L/∂X and ∂L/∂F?



Why do we need to find ∂L/∂X and ∂L/∂F

So let's find the gradients for X and F — ∂L/∂X and ∂L/∂F

Finding ∂L/∂F

This has two steps as we have done earlier.

- Find the local gradient **∂O/∂F**
- Find **∂L/∂F** using chain rule

*Step 1: Finding the local gradient — **∂O/∂F:***

This means we have to differentiate Output Matrix O with Filter F. From our convolution operation, we know the values. So let us start differentiating the first element of O- O^{11} with respect to the elements of F — F^{11} , F^{12}, F^{21} and F^{22}

Local Gradients → (A)

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

Finding derivatives with respect to F_{11} , F_{12} , F_{21} *and* F_{22}

$$\frac{\partial O_{11}}{\partial F_{11}} = X_{11} \quad \frac{\partial O_{11}}{\partial F_{12}} = X_{12} \quad \frac{\partial O_{11}}{\partial F_{21}} = X_{21} \quad \frac{\partial O_{11}}{\partial F_{22}} = X_{22}$$

Similarly, we can find the local gradients for O_{12} , O_{21} *and* O_{22}

Step 2: Using the Chain rule:

As described in our previous examples, we need to find **∂L/∂F** as:

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial O} * \frac{\partial O}{\partial F}$$

Gradient to update Filter F

Loss Gradient from previous layer

Local Gradients

O and **F** are matrices. And **∂O/∂F** will be a partial derivative of a matrix **O** with respect to a matrix **F**! On top of it we have to use the chain rule. This does look complicated but thankfully we can use the formula below to expand it.

$$\text{For every element of } F$$

$$\frac{\partial L}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial F_i}$$

Formula to derive a partial derivative of a matrix with respect to a matrix, using the chain rule

Expanding, we get..

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{11}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{11}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{11}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{11}}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{12}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{12}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{12}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{12}}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{21}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{21}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{21}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{21}}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{22}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{22}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{22}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{22}}$$

Derivatives of **∂L/∂F**

Substituting the values of the local gradient — **∂O/∂F from Equation A**, we get

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * X_{11} + \frac{\partial L}{\partial O_{12}} * X_{12} + \frac{\partial L}{\partial O_{21}} * X_{21} + \frac{\partial L}{\partial O_{22}} * X_{22}$$

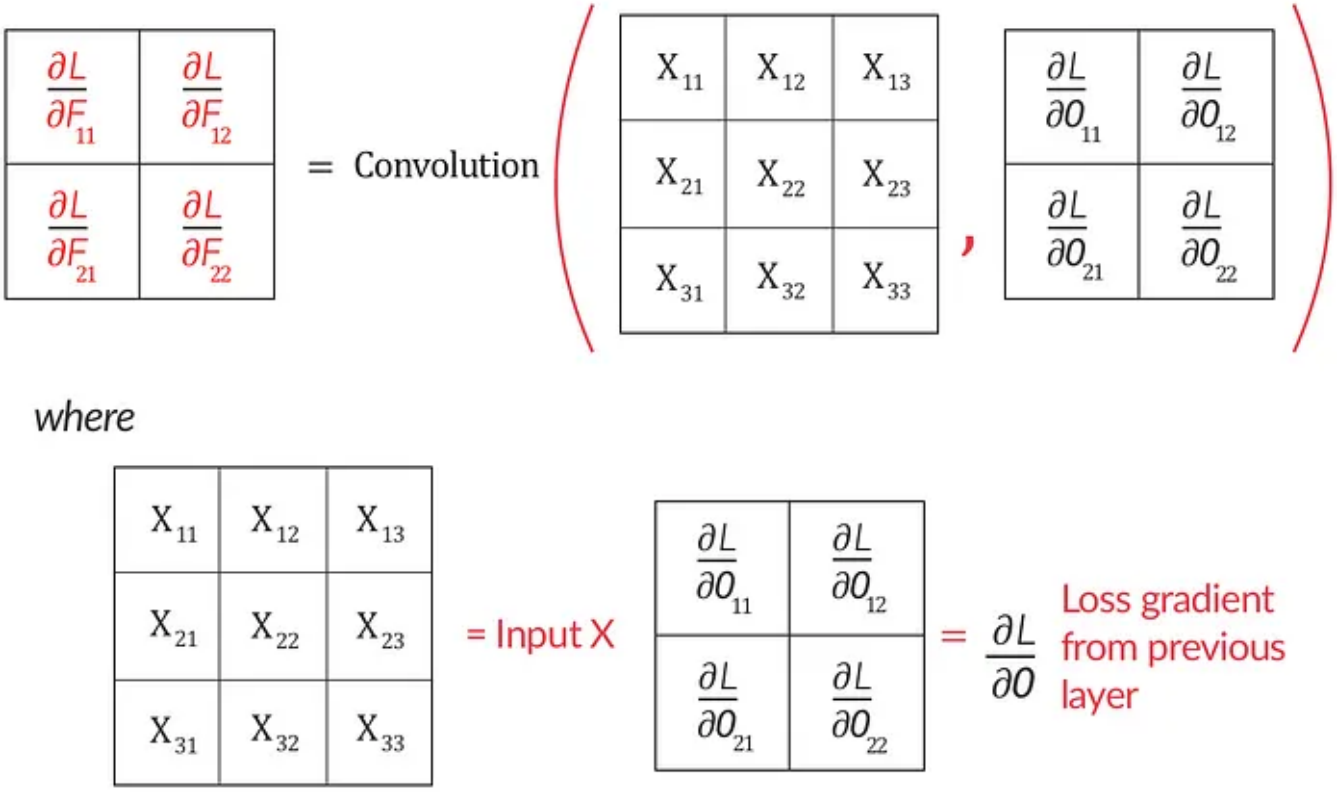
$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * X_{12} + \frac{\partial L}{\partial O_{12}} * X_{13} + \frac{\partial L}{\partial O_{21}} * X_{22} + \frac{\partial L}{\partial O_{22}} * X_{23}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * X_{21} + \frac{\partial L}{\partial O_{12}} * X_{22} + \frac{\partial L}{\partial O_{21}} * X_{31} + \frac{\partial L}{\partial O_{22}} * X_{32}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * X_{22} + \frac{\partial L}{\partial O_{12}} * X_{23} + \frac{\partial L}{\partial O_{21}} * X_{32} + \frac{\partial L}{\partial O_{22}} * X_{33}$$

Using local gradients values from Equation A

If you closely look at it, this represents an operation we are quite familiar with. We can represent it as a **convolution operation between input X** and loss gradient **∂L/∂O as shown below:**



∂L/∂F = Convolution of input matrix X and loss gradient ∂L/∂O

∂L/∂F is nothing but the convolution between Input X and Loss Gradient from the next layer ∂L/∂O

Finding ∂L/∂X:

Step 1: Finding the local gradient — ∂O/∂X:

Similar to how we found the local gradients earlier, we can find **∂O/∂X** as:

Local Gradients: → (B)

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

Differentiating with respect to X_{11}, X_{12}, X_{21} *and* X_{22}

$$\frac{\partial O_{11}}{\partial X_{11}} = F_{11} \quad \frac{\partial O_{11}}{\partial X_{12}} = F_{12} \quad \frac{\partial O_{11}}{\partial X_{21}} = F_{21} \quad \frac{\partial O_{11}}{\partial X_{22}} = F_{22}$$

Similarly, we can find local gradients for O_{12}, O_{21} *and* O_{22}

Local gradients **∂O/∂X**

Step 2: Using the Chain rule:

For every element of X_i

$$\frac{\partial L}{\partial X_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial X_i}$$

Expanding this and substituting from Equation B, we get

$$\frac{\partial L}{\partial X_{11}} = \frac{\partial L}{\partial O_{11}} * F_{11}$$

$$\frac{\partial L}{\partial X_{12}} = \frac{\partial L}{\partial O_{11}} * F_{12} + \frac{\partial L}{\partial O_{12}} * F_{11}$$

$$\frac{\partial L}{\partial X_{13}} = \frac{\partial L}{\partial O_{12}} * F_{12}$$

$$\frac{\partial L}{\partial X_{21}} = \frac{\partial L}{\partial O_{11}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{11}$$

$$\frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial O_{11}} * F_{22} + \frac{\partial L}{\partial O_{12}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{12} + \frac{\partial L}{\partial O_{22}} * F_{11}$$

$$\frac{\partial L}{\partial X_{23}} = \frac{\partial L}{\partial O_{12}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{12}$$

$$\frac{\partial L}{\partial X_{31}} = \frac{\partial L}{\partial O_{21}} * F_{21}$$

$$\frac{\partial L}{\partial X_{32}} = \frac{\partial L}{\partial O_{21}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{21}$$

$$\frac{\partial L}{\partial X_{33}} = \frac{\partial L}{\partial O_{22}} * F_{22}$$

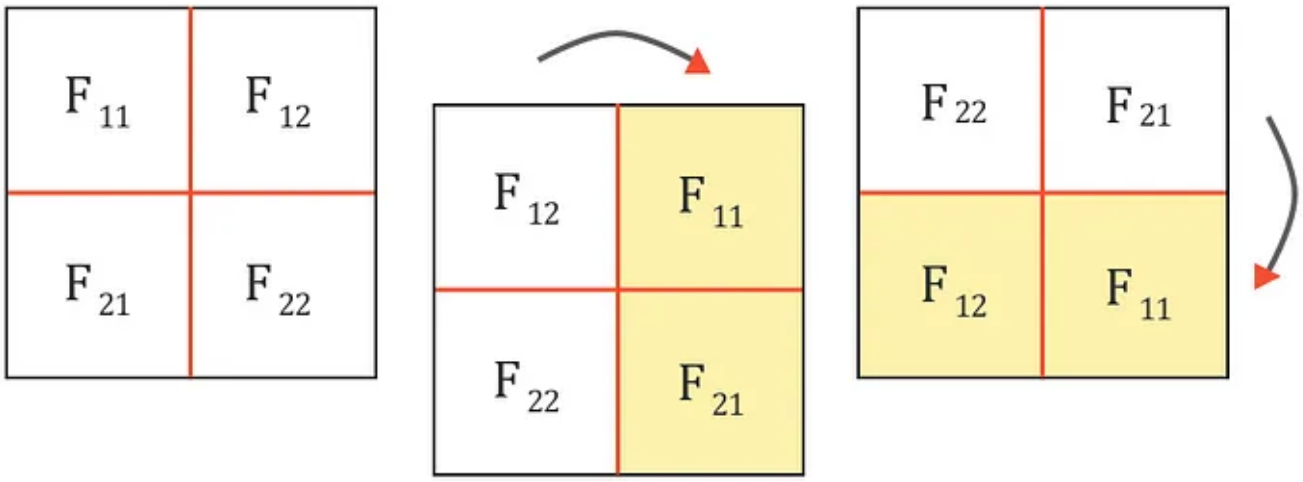
Derivatives of **∂L/∂X** using local gradients from Equation

Ok. Now we have the values of **∂L/∂X.**

Believe it or not, even this can be represented as a convolution operation.

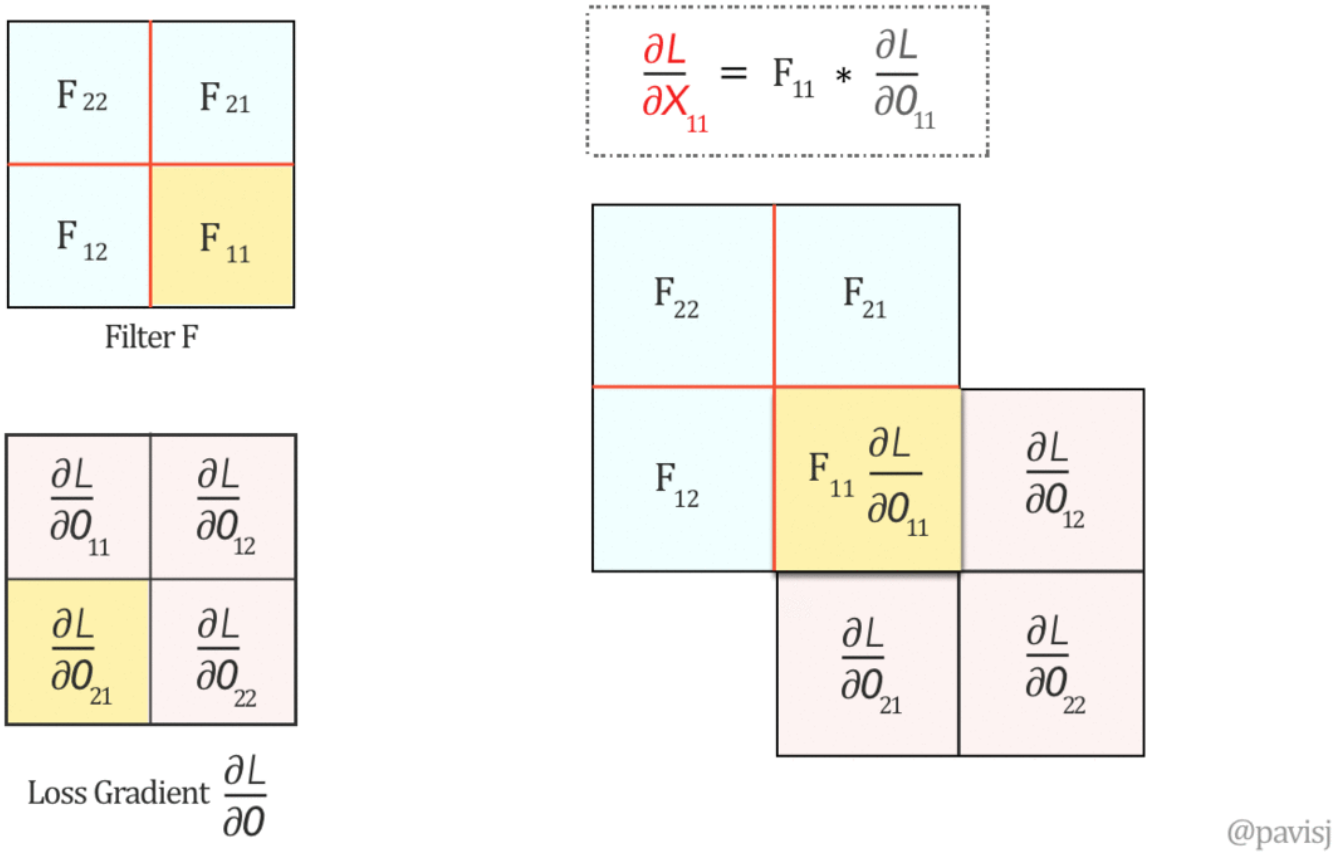
∂L/∂X can be represented as 'full' convolution between a 180-degree rotated Filter F and loss gradient ∂L/∂O

First, let us rotate the Filter F by 180 degrees. This is done by flipping it first vertically and then horizontally.



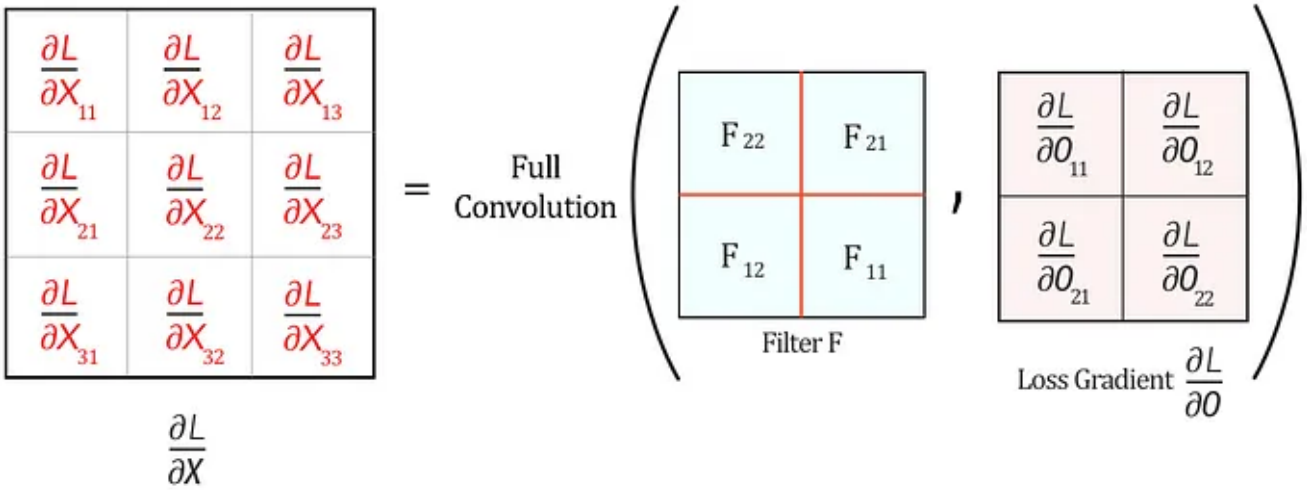
Flipping Filter F by 180 degrees — flipping it vertically and horizontally

Now, let us do a 'full' convolution between this flipped Filter F and ∂L/∂O, which can be visualized as below: (*It is like sliding one matrix over another from right to left, bottom to top)*



Full Convolution operation visualized between 180-degree flipped Filter F and loss gradient ∂L/∂O

The full convolution above generates the values of ∂L/∂X and hence we can represent ∂L/∂X as

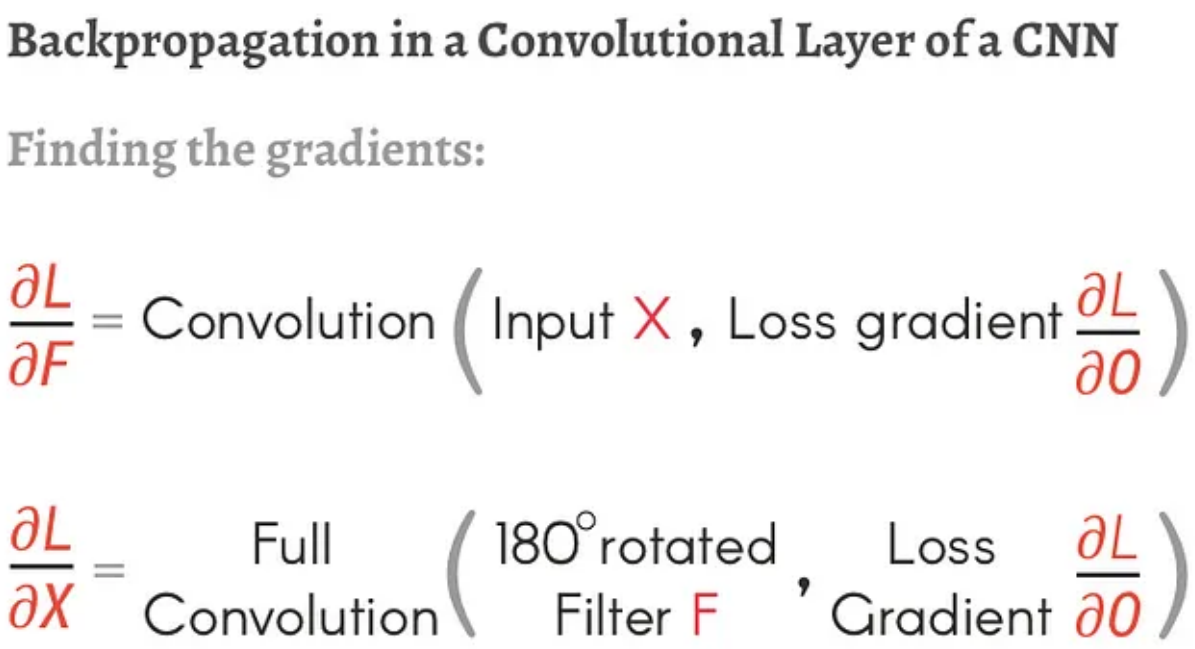


∂L/∂X can be represented as 'full' convolution between a 180-degree rotated Filter F and loss gradient ∂L/∂O

*Well, now that we have found **∂L/∂X and ∂L/∂F**, we can now come to this conclusion*

> Both the Forward pass and the Backpropagation of a Convolutional layer are Convolutions

Summing it up:



How to calculate **∂L/∂X and ∂L/∂F**

Hope this helped to explain how Backpropagation works in a Convolutional layer of a CNN.

If you want to read more about it, do look at these links below. And do show some love by clapping for this article. Adios! :)

Backpropagation In Convolutional Neural Networks

Convolutional neural networks (CNNs) are a biologically-inspired variation of the multilayer perceptrons (MLPs)...

www.jefkine.com

Back Propagation in Convolutional Neural Networks — Intuition and