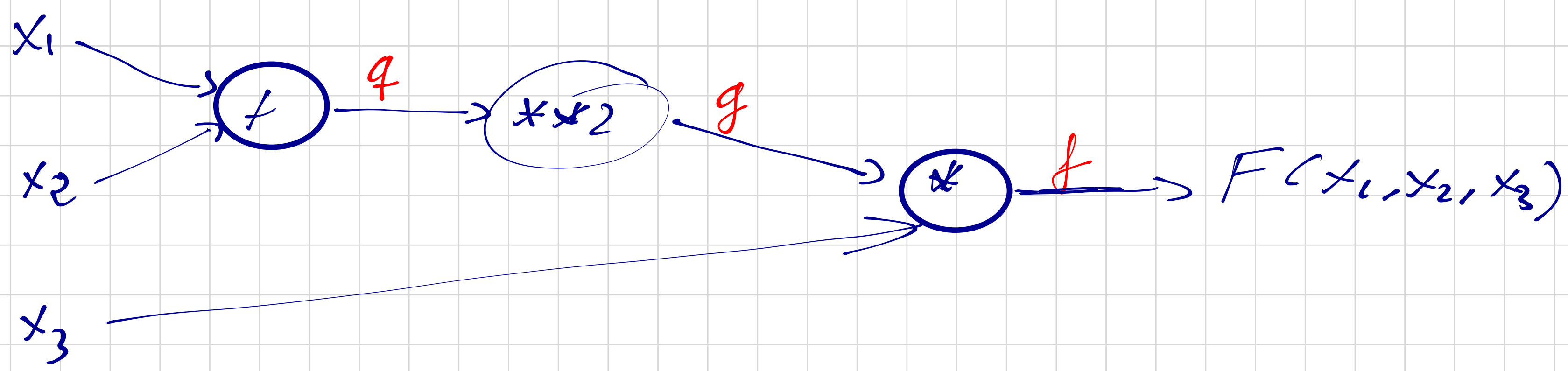
BACKPROPAGATION

- Notice until now we have computed the ∇F by hand. This was easy because the pixels only depends on his neighbours, but if we have a very complex cost function to minimize (e.g. a deep neural network), the task is very hard and prone to error.

~~Solution~~ Back propagation and Chain Rule.

Take as example this function $F(x_1, x_2, x_3) = (x_1 + x_2)^2 \cdot x_3$

Decompose the function into small pieces (elementary functions)



Small pieces

$q = x_1 + x_2$

$g = q^2$

$f = g \cdot x_3$

compute by hand the derivative of this elementary functions $\Rightarrow$

Derivatives

$\frac{\partial q}{\partial x_1} = 1 \qquad \frac{\partial q}{\partial x_2} = 1$

$\frac{\partial g}{q} = 2q$

$\frac{\partial f}{\partial g} = x_3 \qquad \frac{\partial f}{\partial x_3} = g$

For a given point (x_1, x_2, x_3) we want to compute the gradient of F at that specific point.

1st step: compute $F(x_1, x_2, x_3)$

2nd step: From $F(x_1, x_2, x_3)$ go back applying the Chain Rule

$$\frac{\partial F(G(x))}{\partial x} = \frac{\partial G(x)}{\partial x} \cdot \frac{\partial F}{\partial G}$$

Example of the chain Rule: Compute the derivative of $\left(\underbrace{(x^2+y)}_{G}\right)^2 \rightarrow F$

$$\frac{\partial}{\partial x}(x^2+y)^2 = \left[\frac{\partial}{\partial x}(x^2+y)\right]\cdot\frac{\partial}{\partial G}G^2 =$$

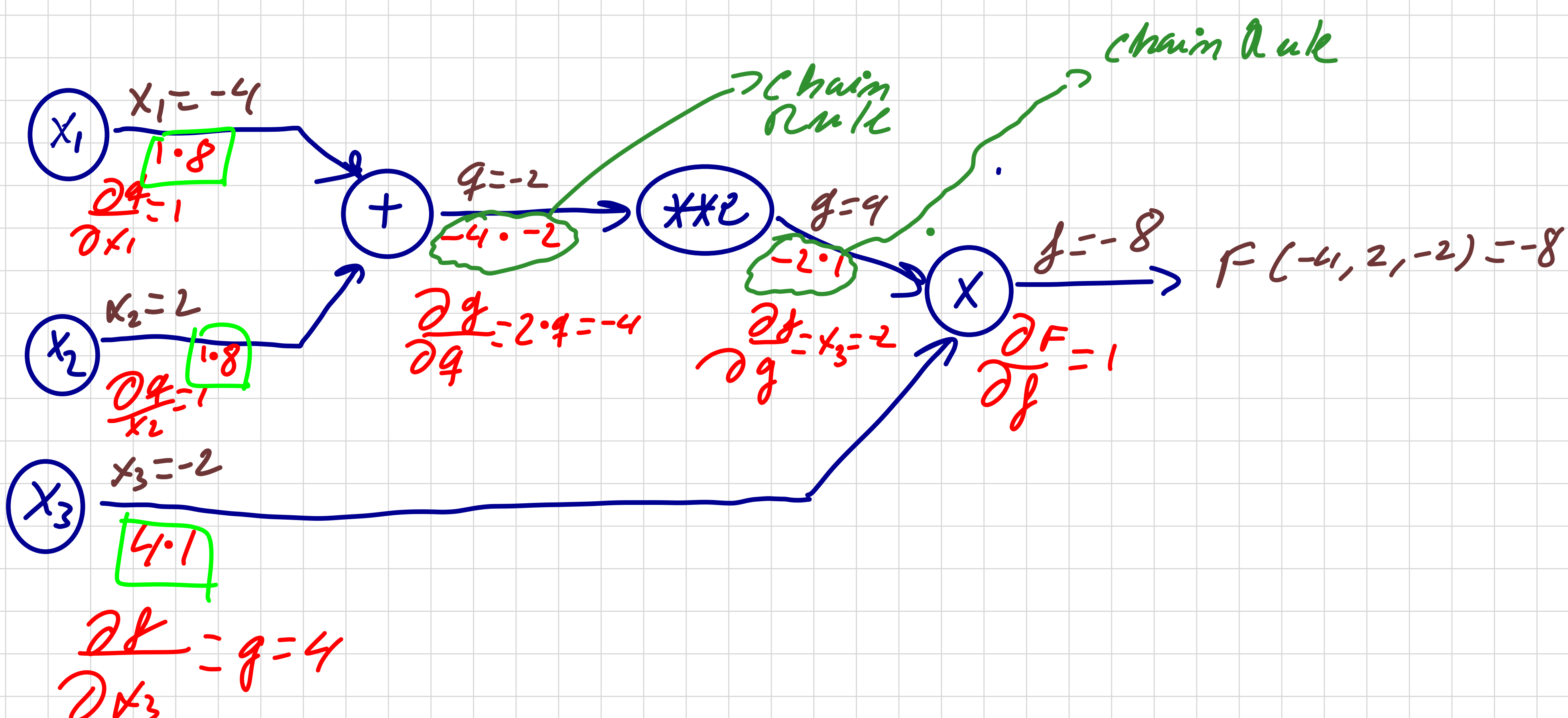
$$= [2x]\cdot 2(x^2+y)$$

In our example $F(x_1,x_2,x_3)=(x_1+x_2)^2\cdot x_3$:

- Compute ∇F at point $\begin{cases} x_1=-4 \\ x_2=2 \\ x_3=-2 \end{cases}$

Forward step

Backward step



$$\nabla F(-4,2,-2)=(8,8,4)^T$$

The result matches if you compute by hand the gradient

$$\nabla F(x_1,x_2,x_3)=\left(2x_3(x_1+x_2),\ 2x_3(x_1+x_2),\ (x_1+x_2)^2\right)^T$$

$$\nabla F(-4,2,-2)=\left(2\cdot(-2)(-4+2),\ 2(-2)(-4+2),\ (-4+2)^2\right)^T=(8,8,4)^T$$