This commutative diagram is all you need to know to train convnets (if you already know multilayer perceptron and convolution).

x is an image squished into a vector.
$sqr()$ reshapes such vector back into a square matrix.
$K * S$ stands for a convolution operation of a kernel K with a matrix S.
$vec()$ turns a matrix into a vector $vec(sqr(x)) = x$.
$M \cdot x$ is a dot product.

It can be somewhat tricky to figure out how to get M from K and vice versa. Took me a day to code it, but most of the time has been wasted on searching for a bug that ended up being caused by numpy's automatic datatype conversion.

$$\begin{tikzcd}
x \arrow[r, "sqr(x)"] \arrow[d, "M\cdot x"] & S \arrow[d, "K*S"] \\
y & Z \arrow[l, "vec(Z)"]
\end{tikzcd}$$

Training

1. Compute y from x and some K by going to the right-down-left way.
2 Find an equivalent $M \cdot x$ operation.
3. Differentiate it as you would normally do in a multilayer perceptron.
4. Do gradient descent on M.
5. Get a new K from an updated M, go to step 1.

How to find M from K and vice versa?

It involves some linear algebra which I had to google how to solve. I wrote all of it without internet but then got stuck at solving a not super nice system of linear equations. Fortunately numpy has a lot of great features built in. A good intuition is that both dot products and convolutions involve summs of products of elements. All code will be available as soon as it's pretty.