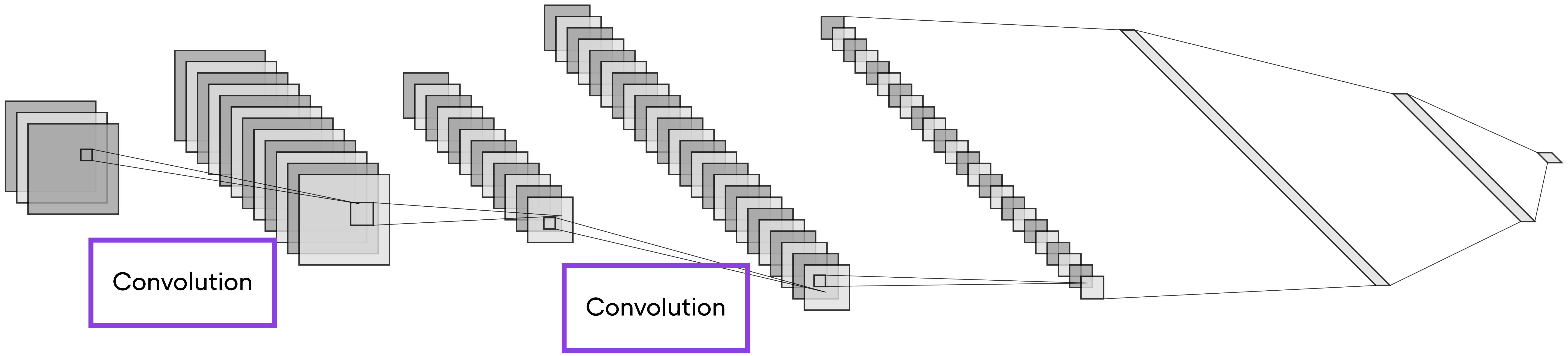
7.2

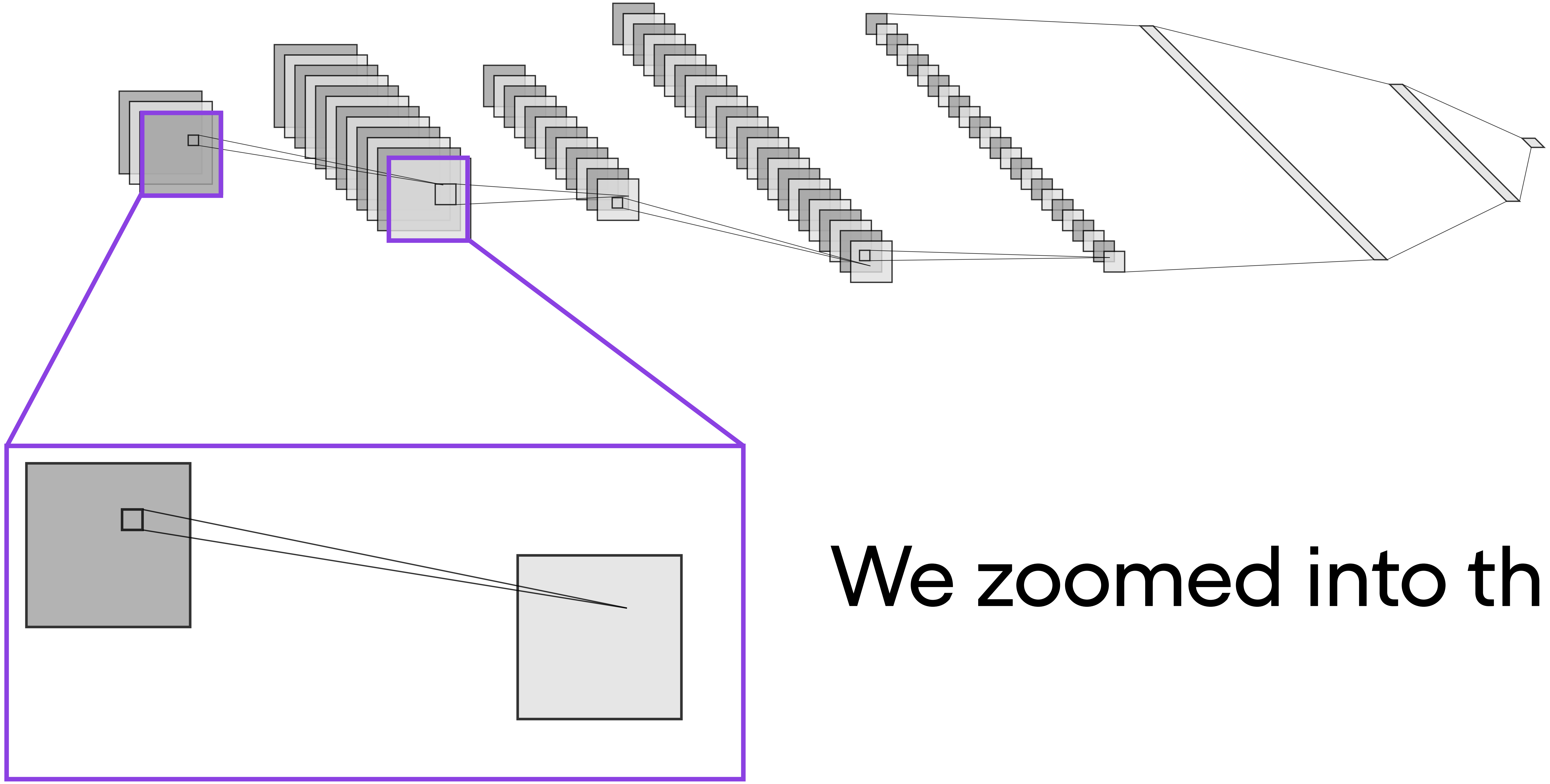
How Convolutional Neural Networks Work

Part 3: Convolutions With Multiple Channels

Sebastian Raschka and the Lightning AI Team

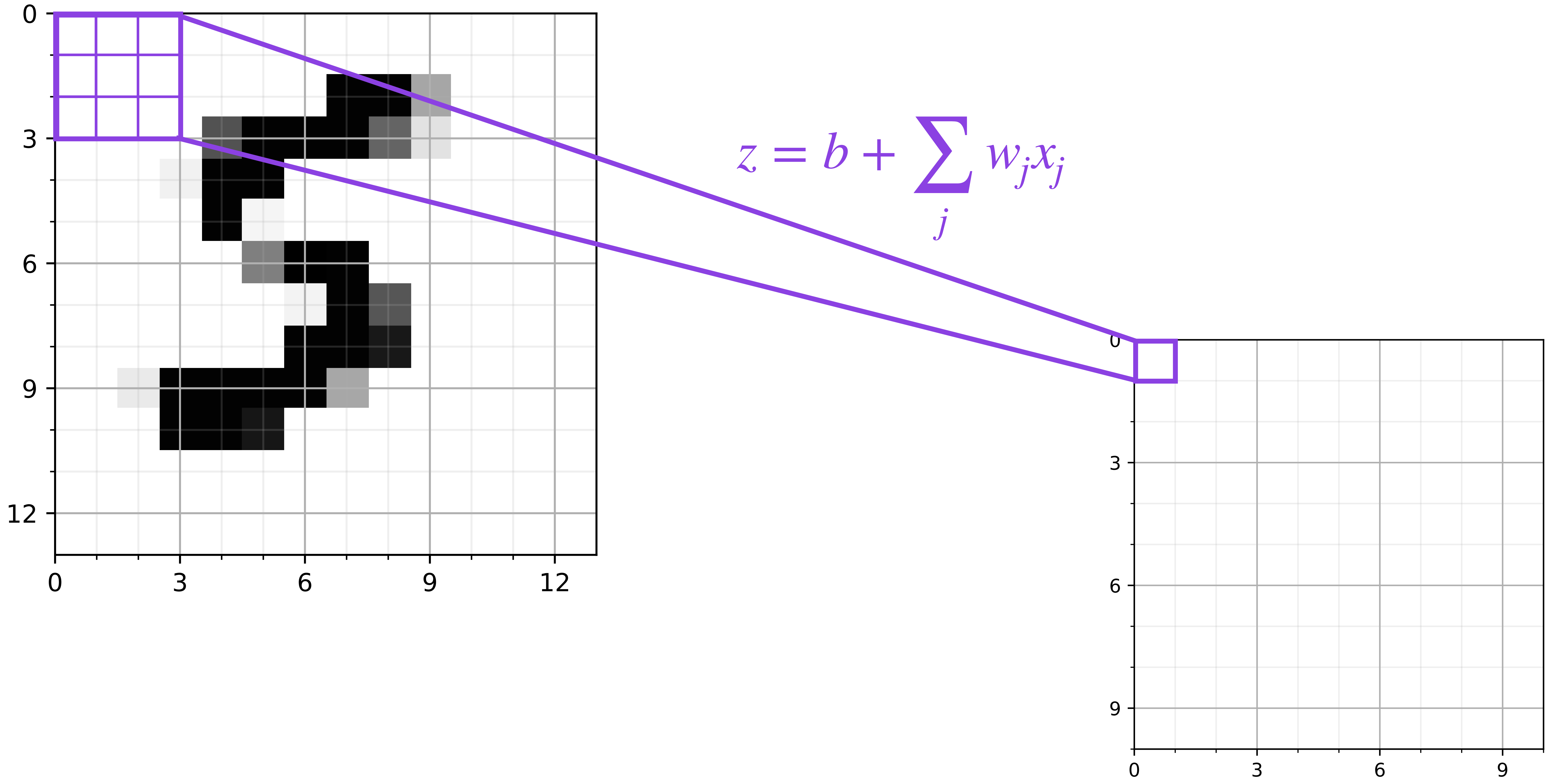
We looked at convolutional layers in more detail





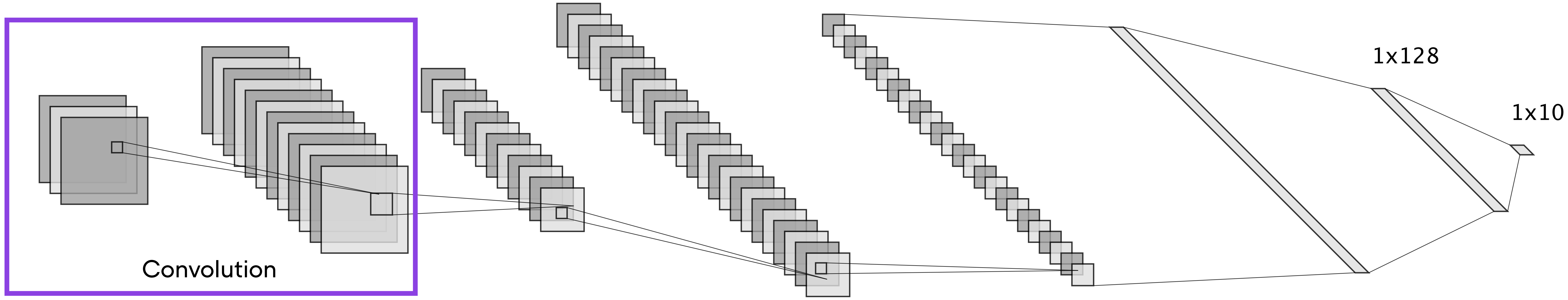
We zoomed into this part

1 input channel, 1 output channel



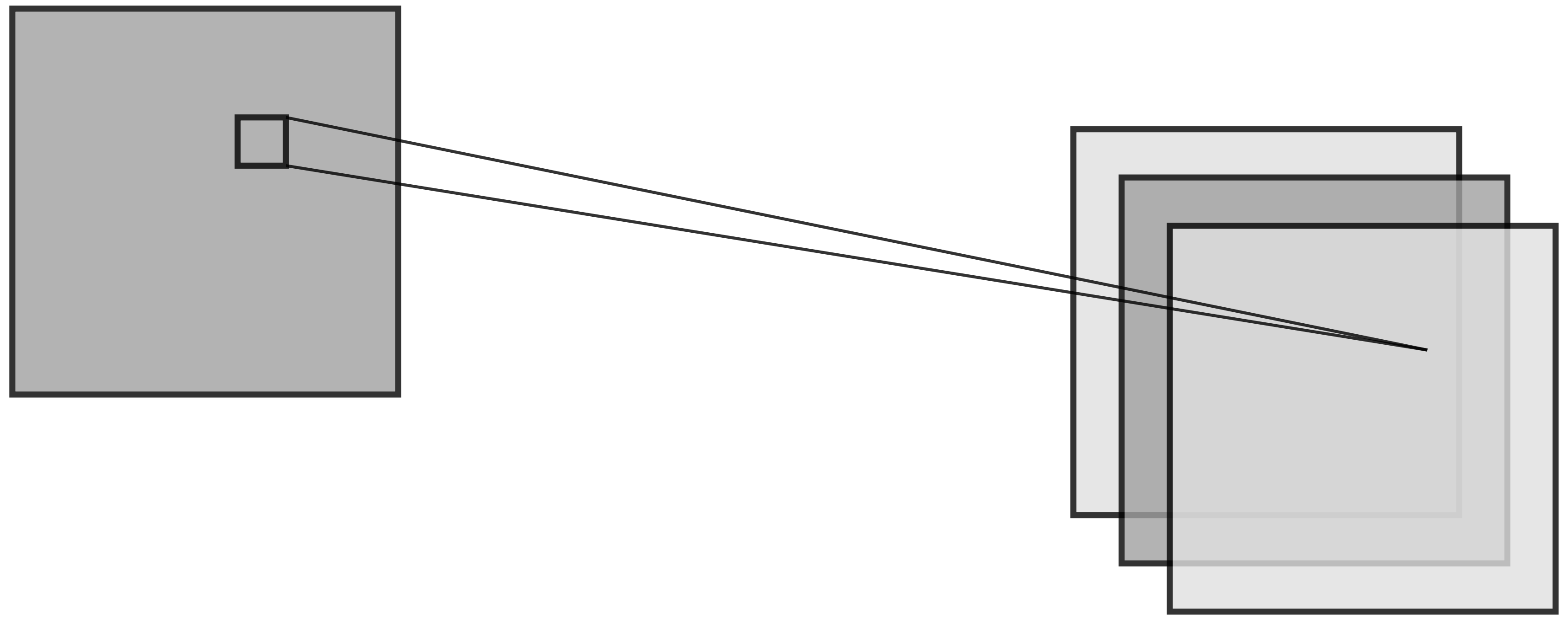
$$z = b + \sum_j w_j x_j$$

Now, let's learn about convolutions with multiple channels



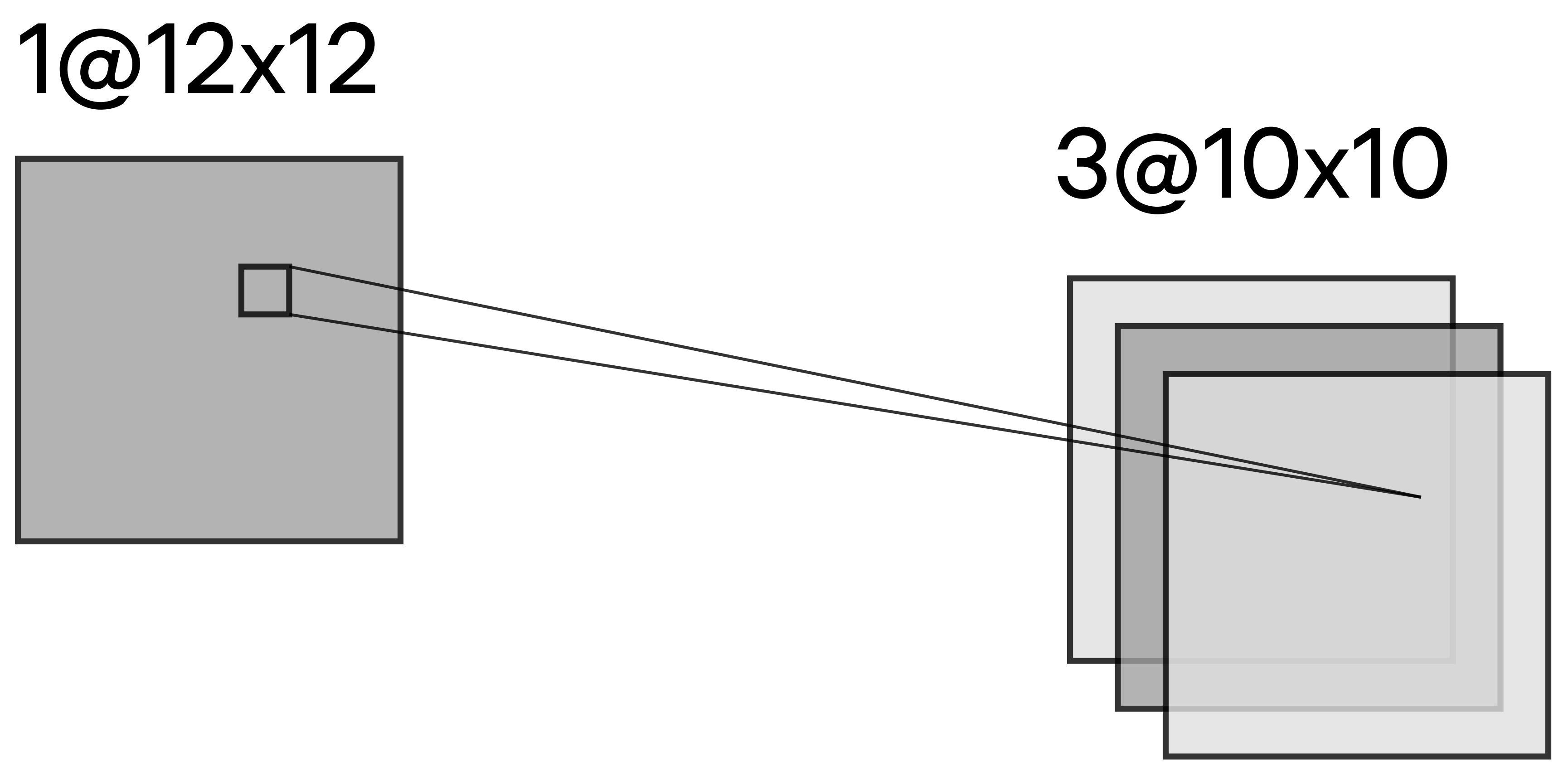
Let's start with multiple output channels

1 input channel, 3 output channels

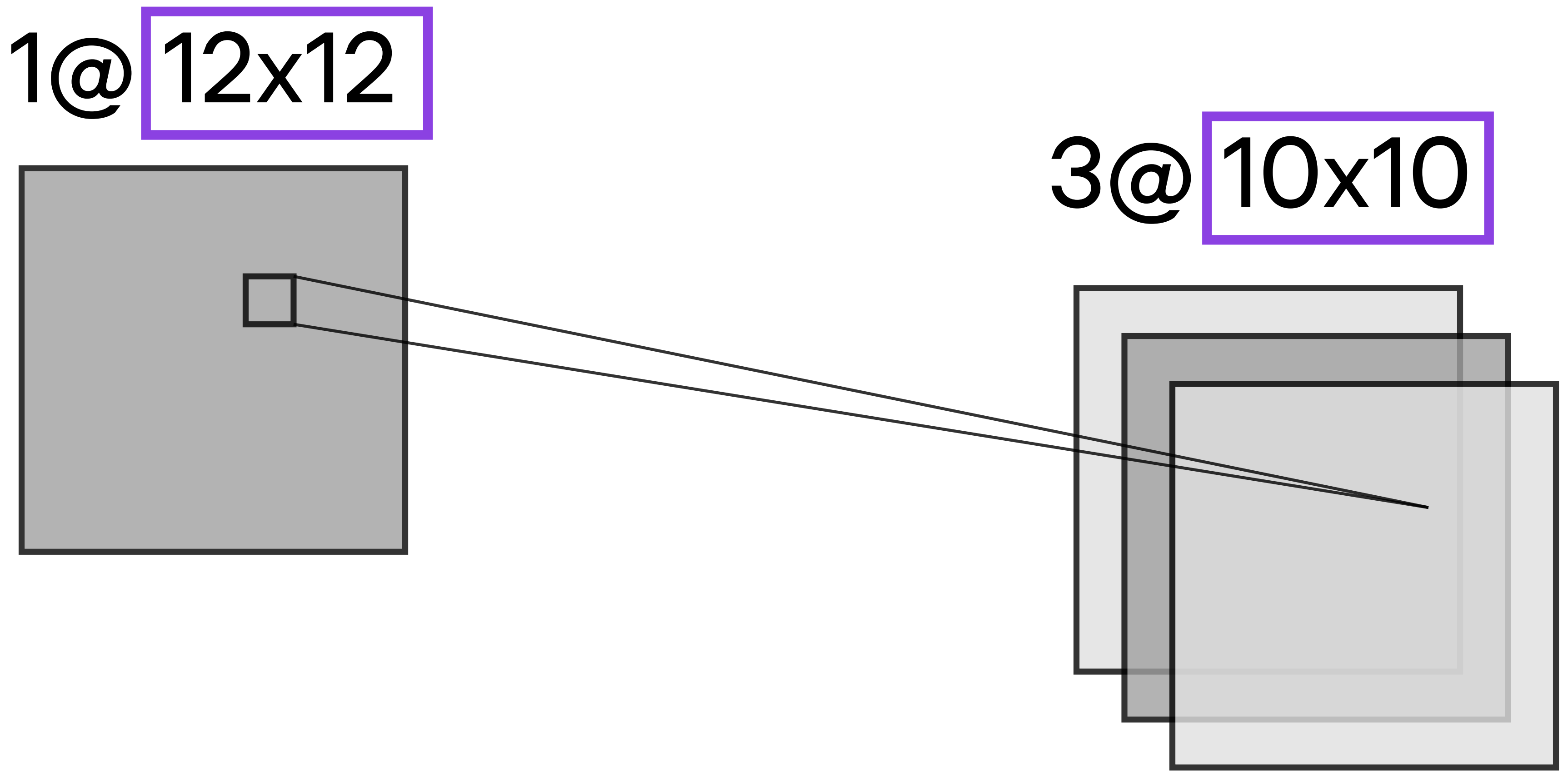


1 input channel, 3 output channels

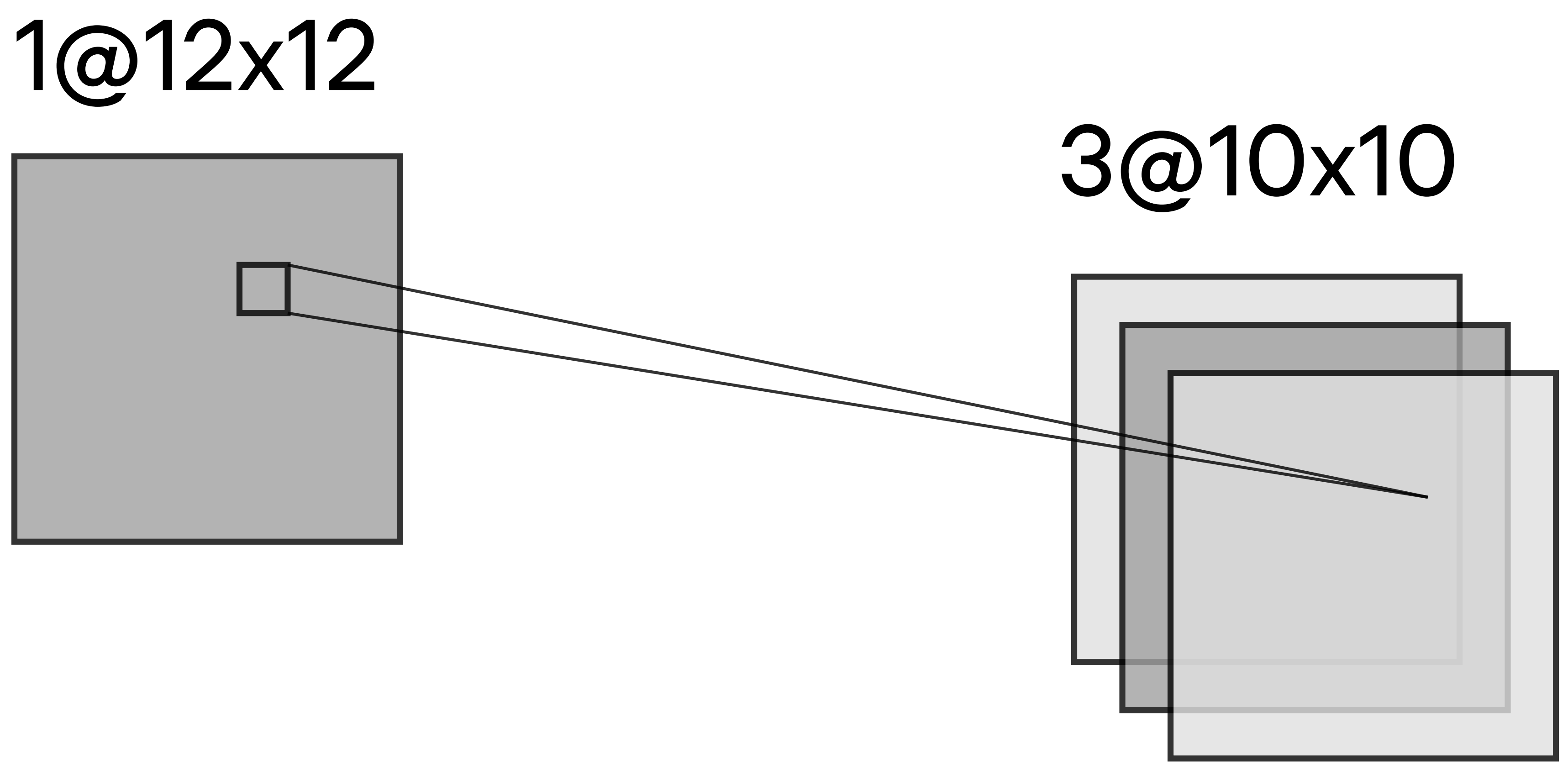
Adding a bit more annotation ...



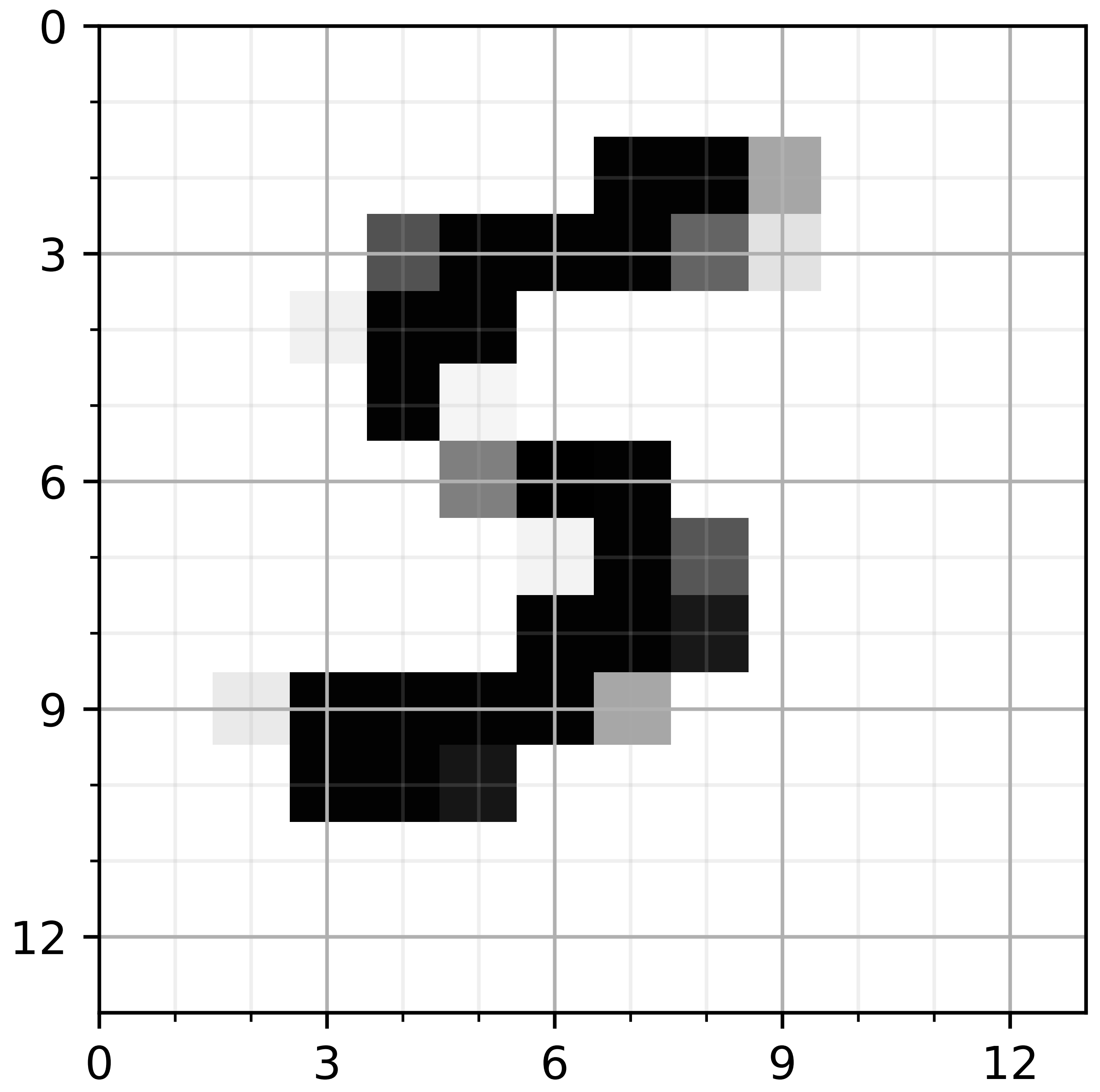
We will discuss how to control the output size in a later video

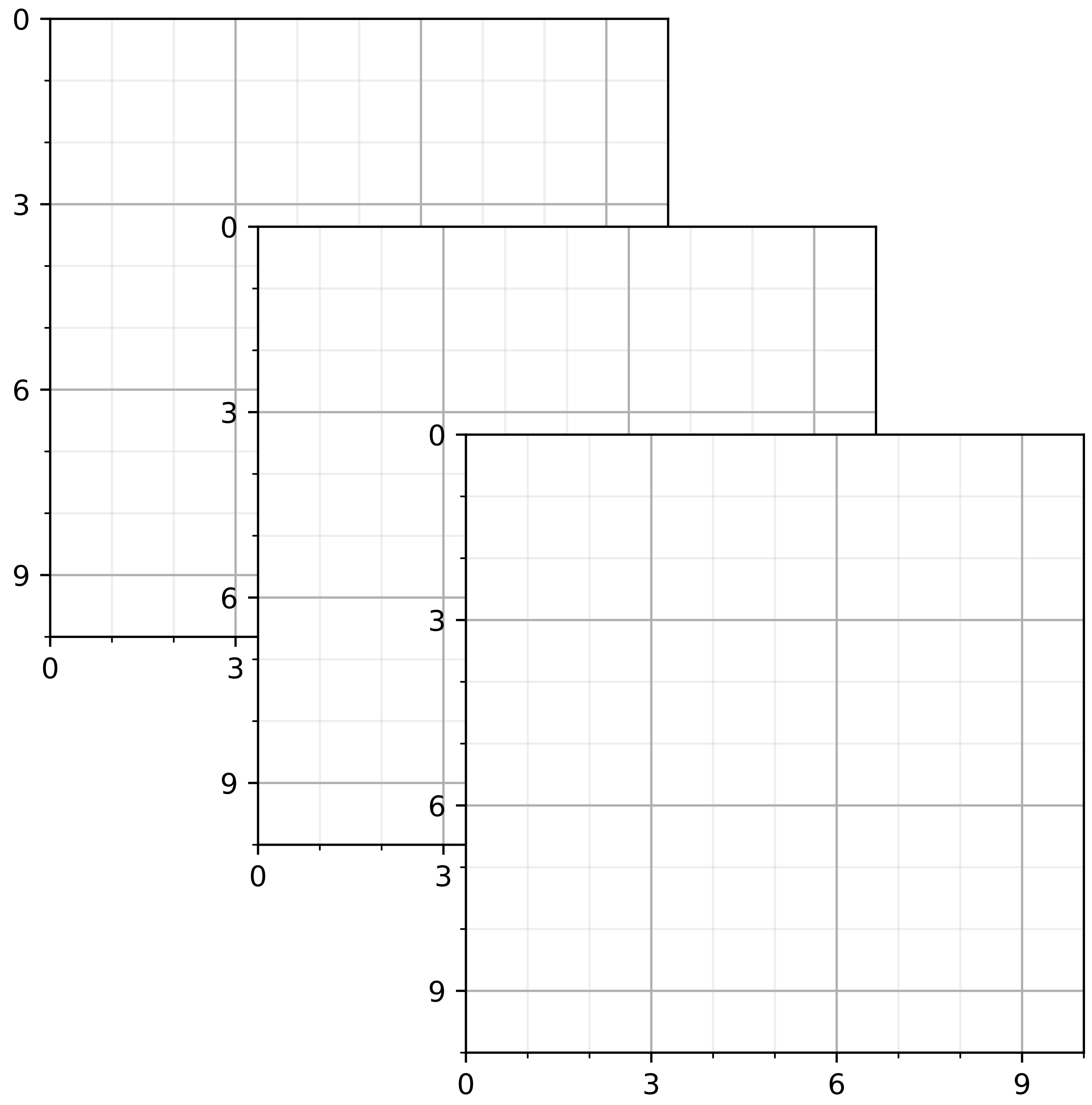


1 input channel, 3 output channels



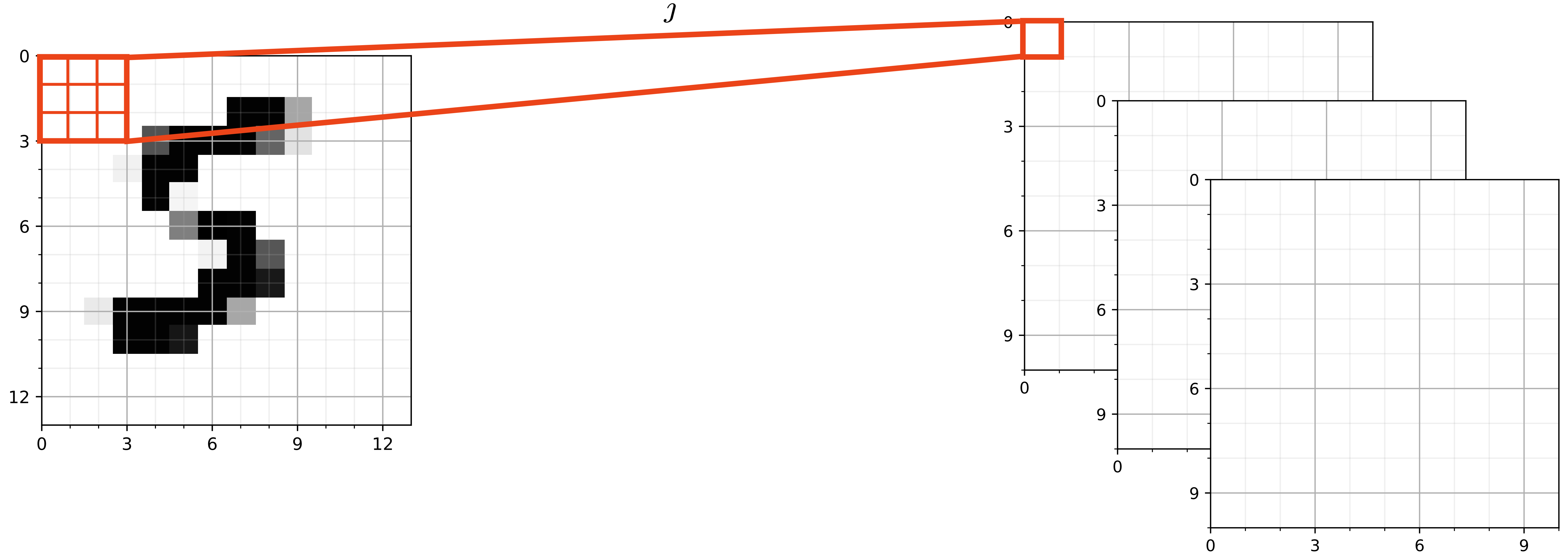
1 input channel, 3 output channels



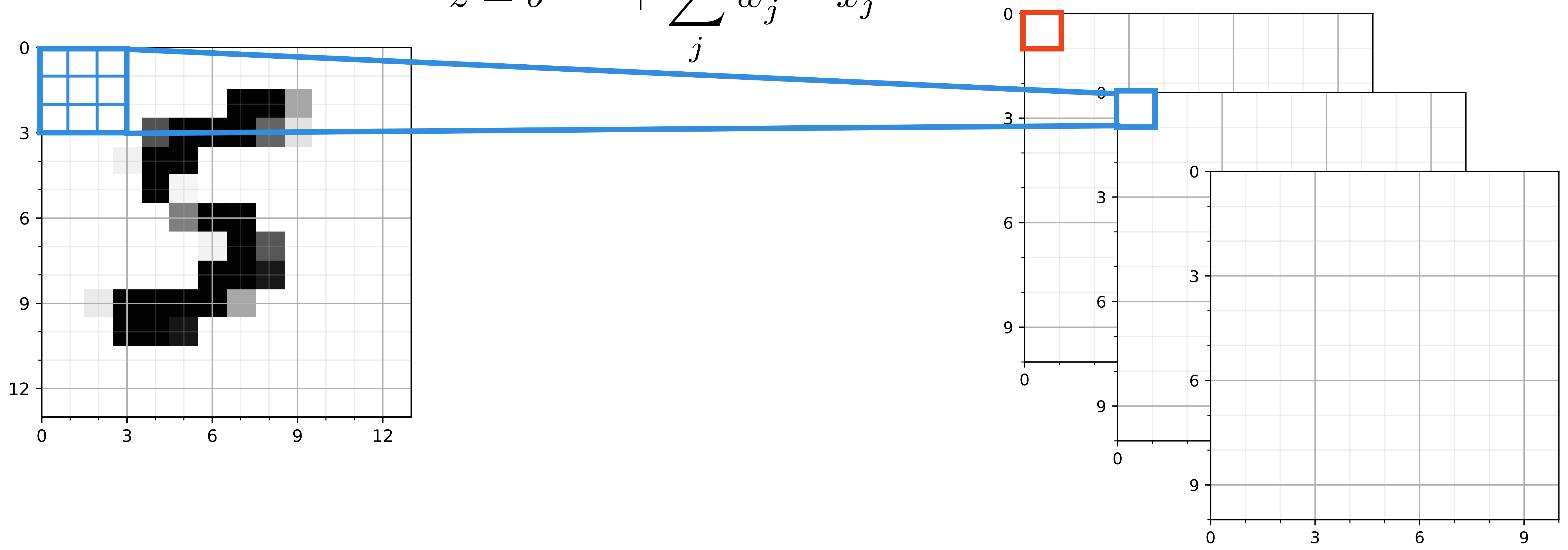


1 input channel, 3 output channels

$$z = b^{(@1)} + \sum_j w_j^{(@1)} x_j$$

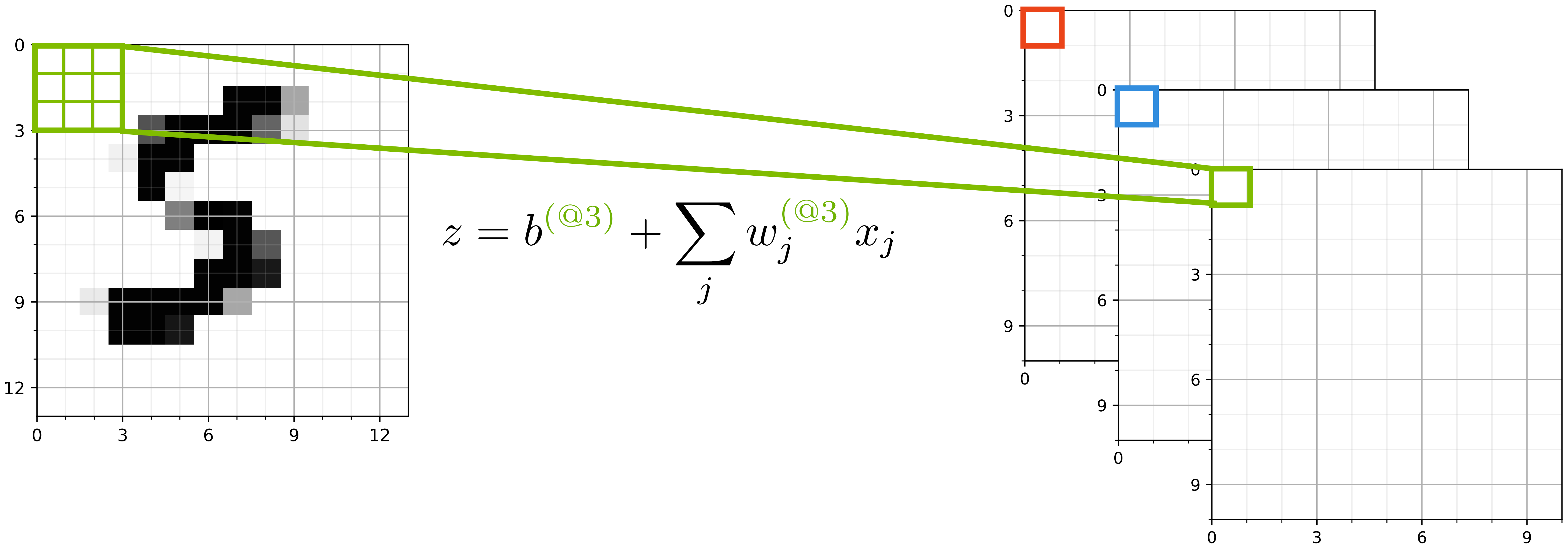


1 input channel, 3 output channels



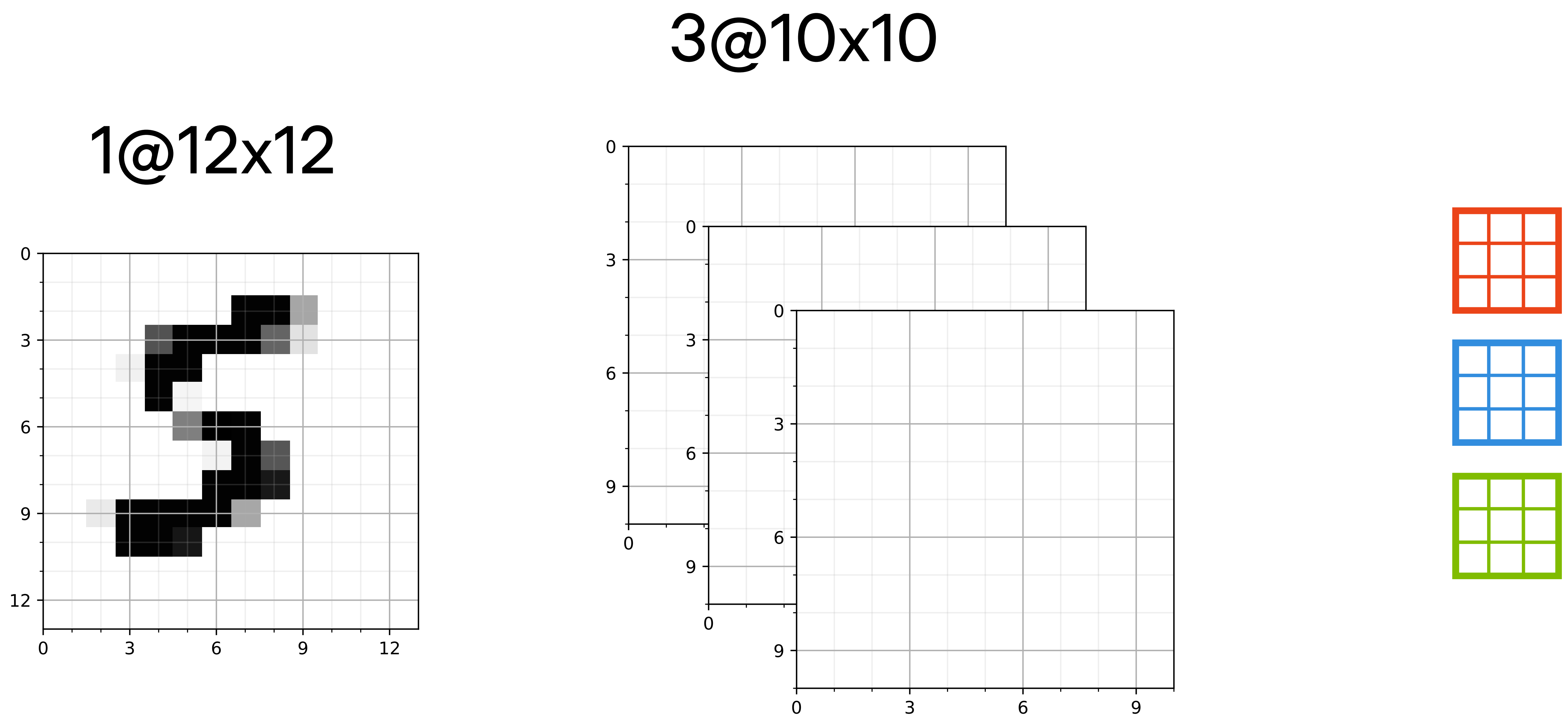
$$z = b^{(@2)} + \sum_j w_j^{(@2)} x_j$$

1 input channel, 3 output channels



$$z = b^{(@3)} + \sum_j w_j^{(@3)} x_j$$

1 input channel, 3 output channels



Multiple "feature detectors" (kernels) are used to create multiple feature maps

What about multiple input channels?

RGB Image

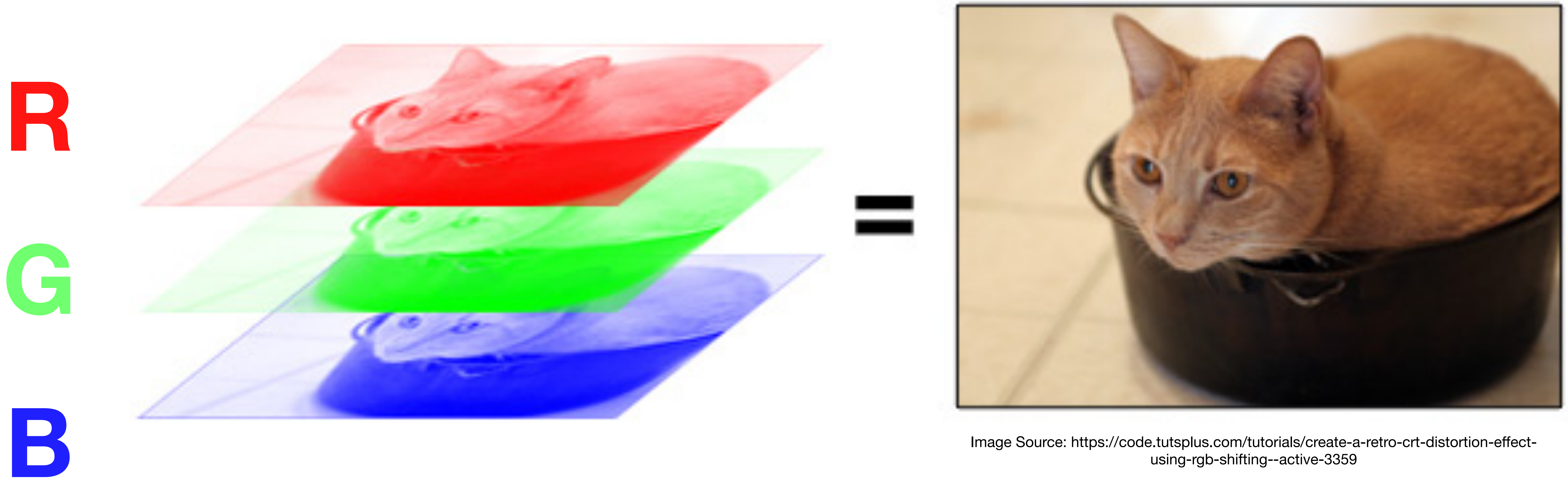
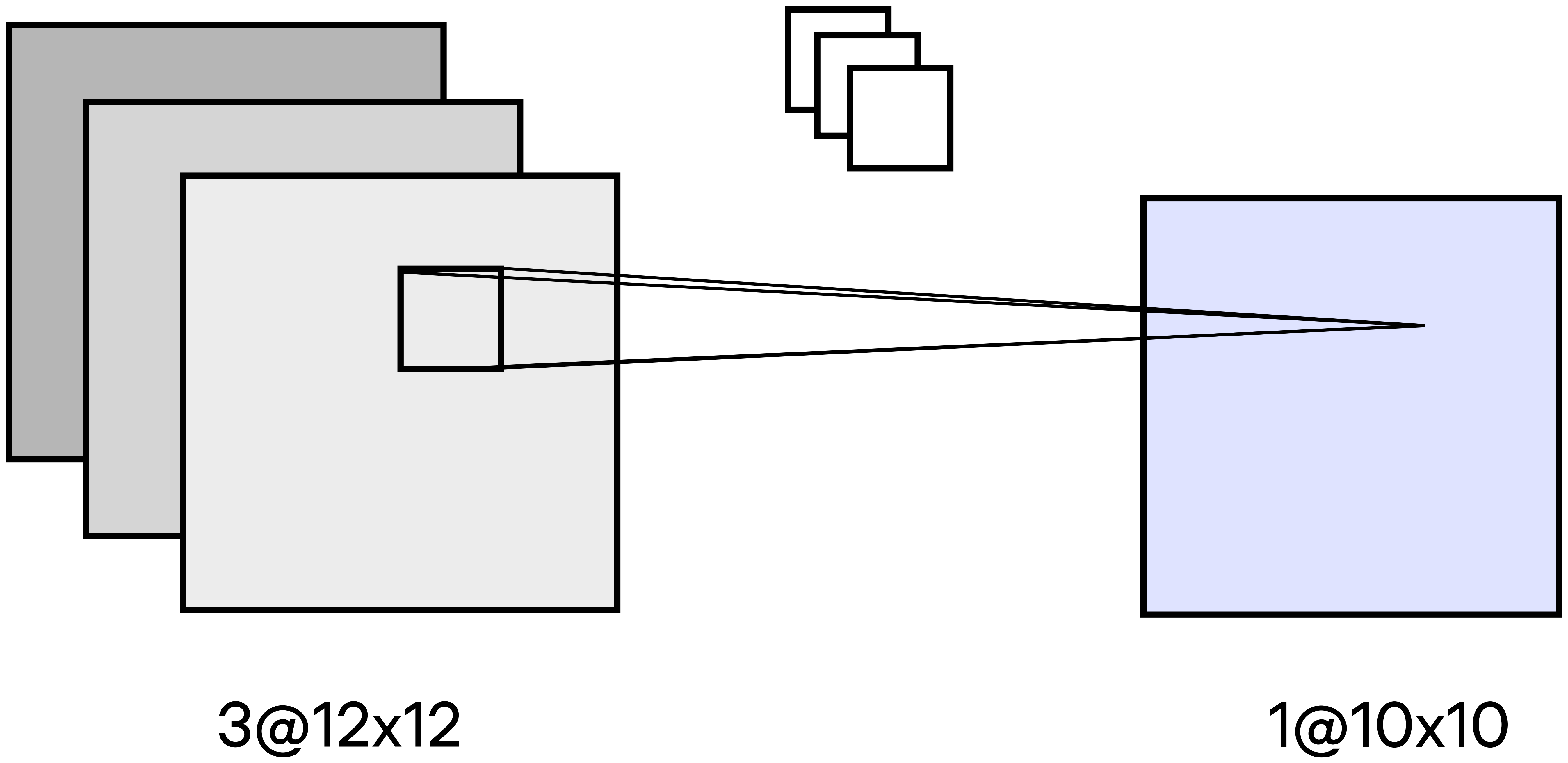


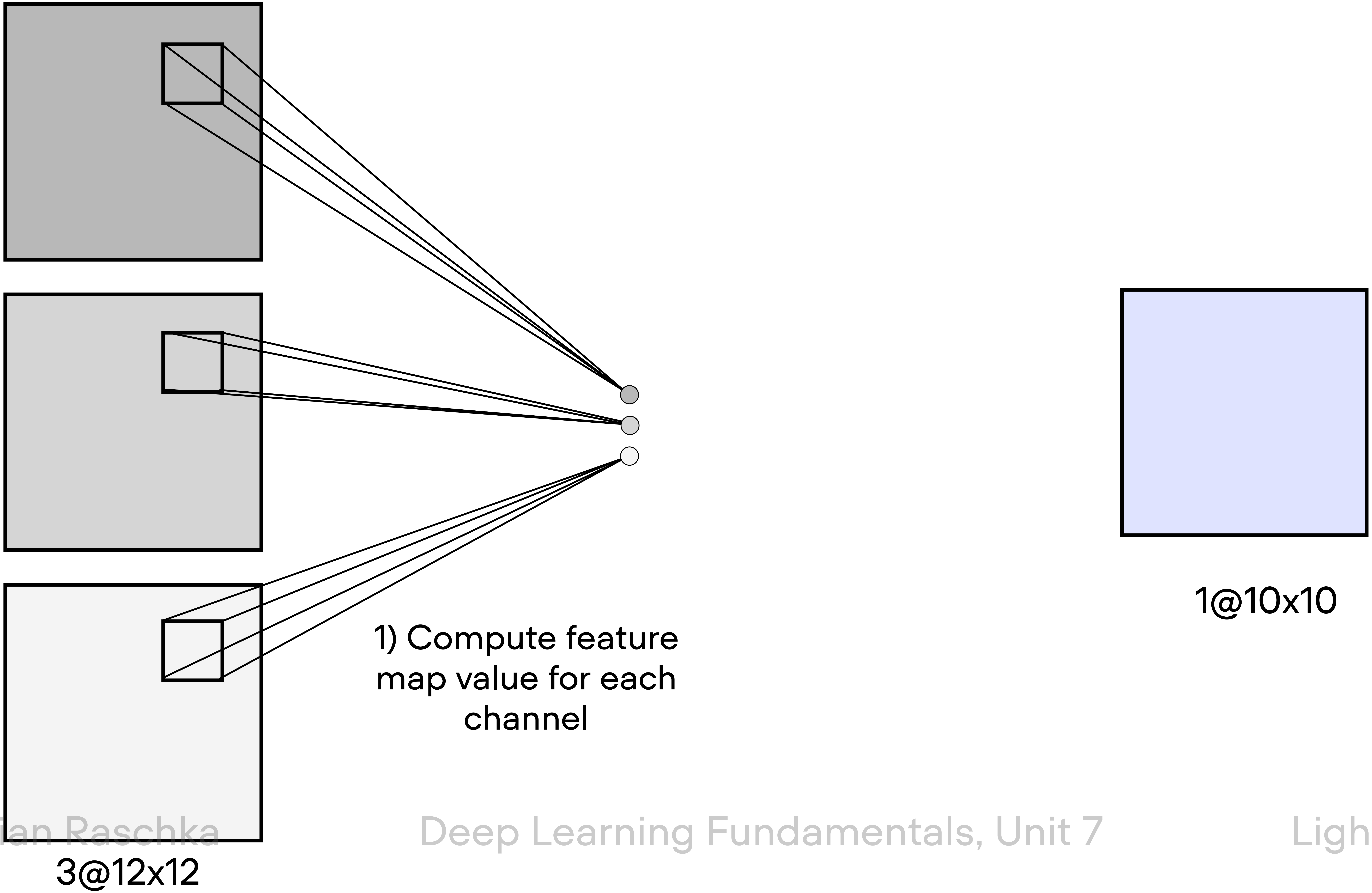
Image Source: https://code.tutsplus.com/tutorials/create-a-retro-crt-distortion-effect-using-rgb-shifting--active-3359

Color image as a stack of matrices

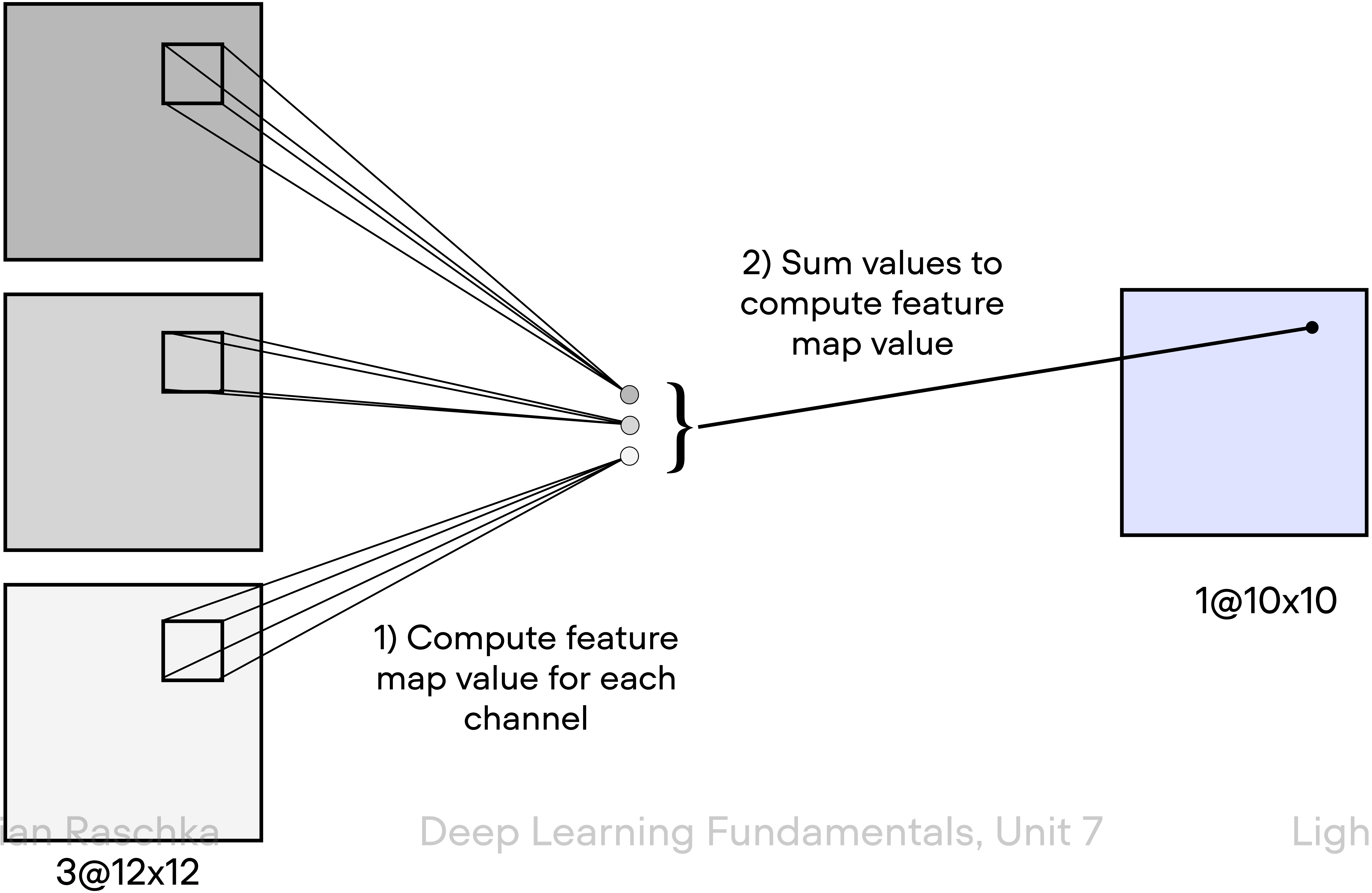
3 input channels, 1 output channel

kernel has 3 channels



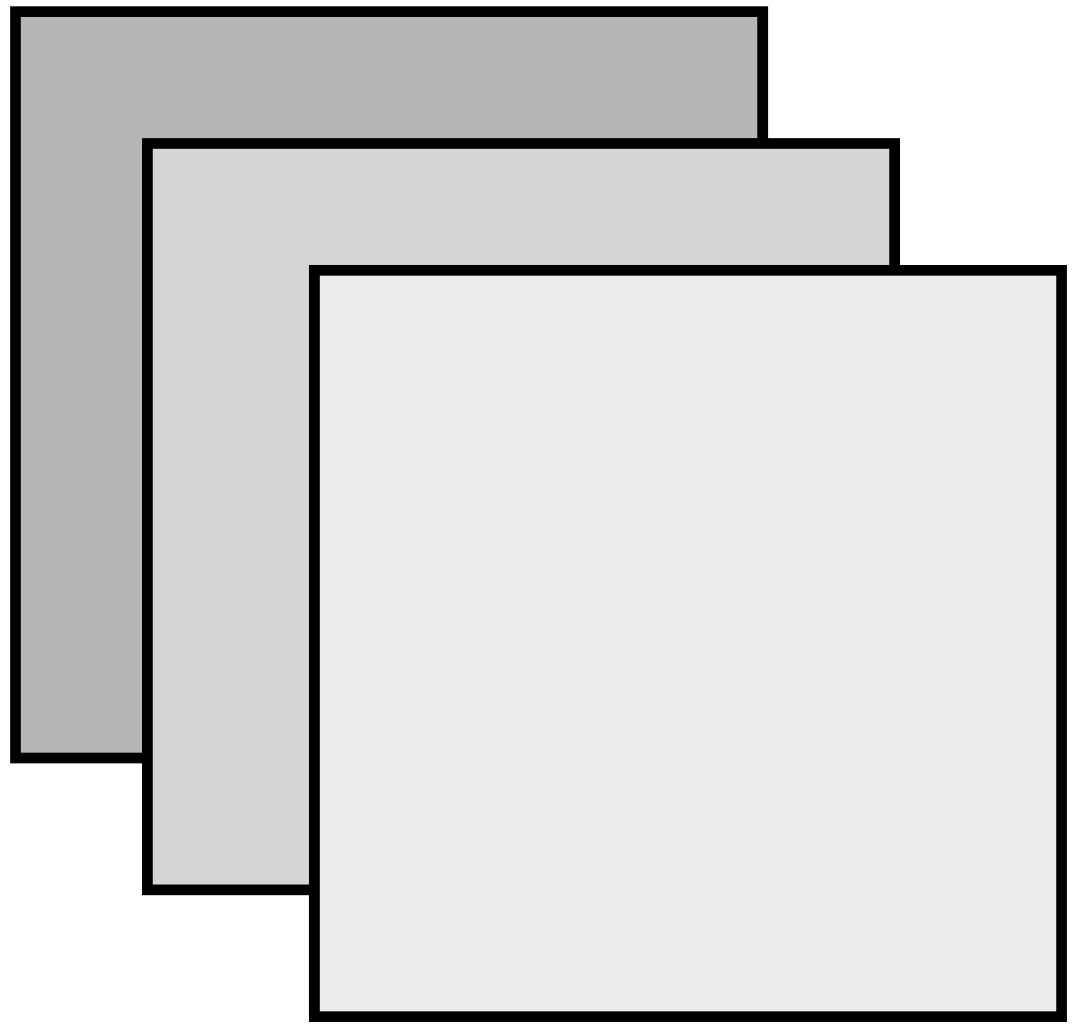


1) Compute feature map value for each channel


2) Sum values to compute feature map value
1@10x10
1) Compute feature map value for each channel
3@12x12

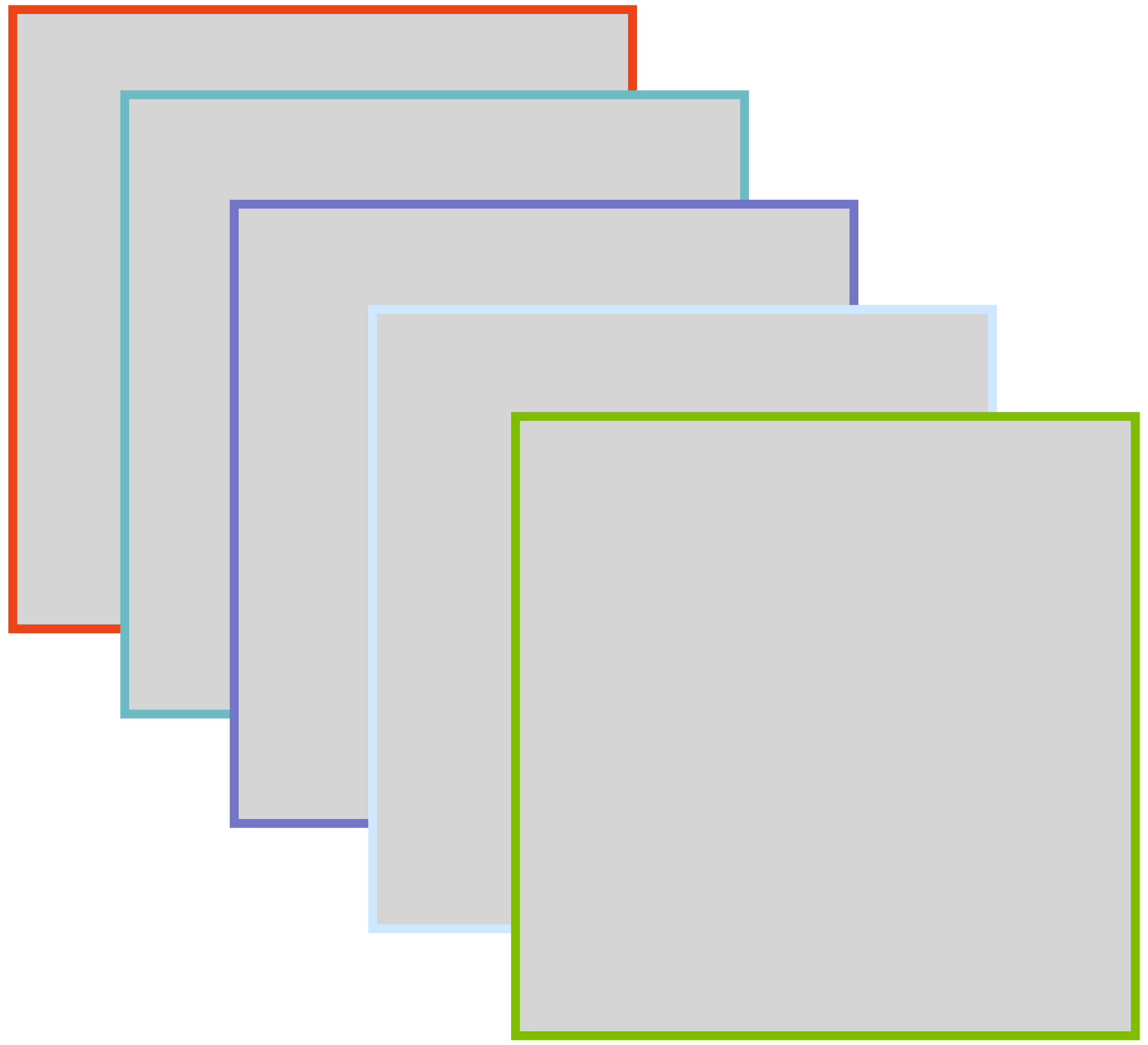
What about multiple input AND output channels?

3 input channels



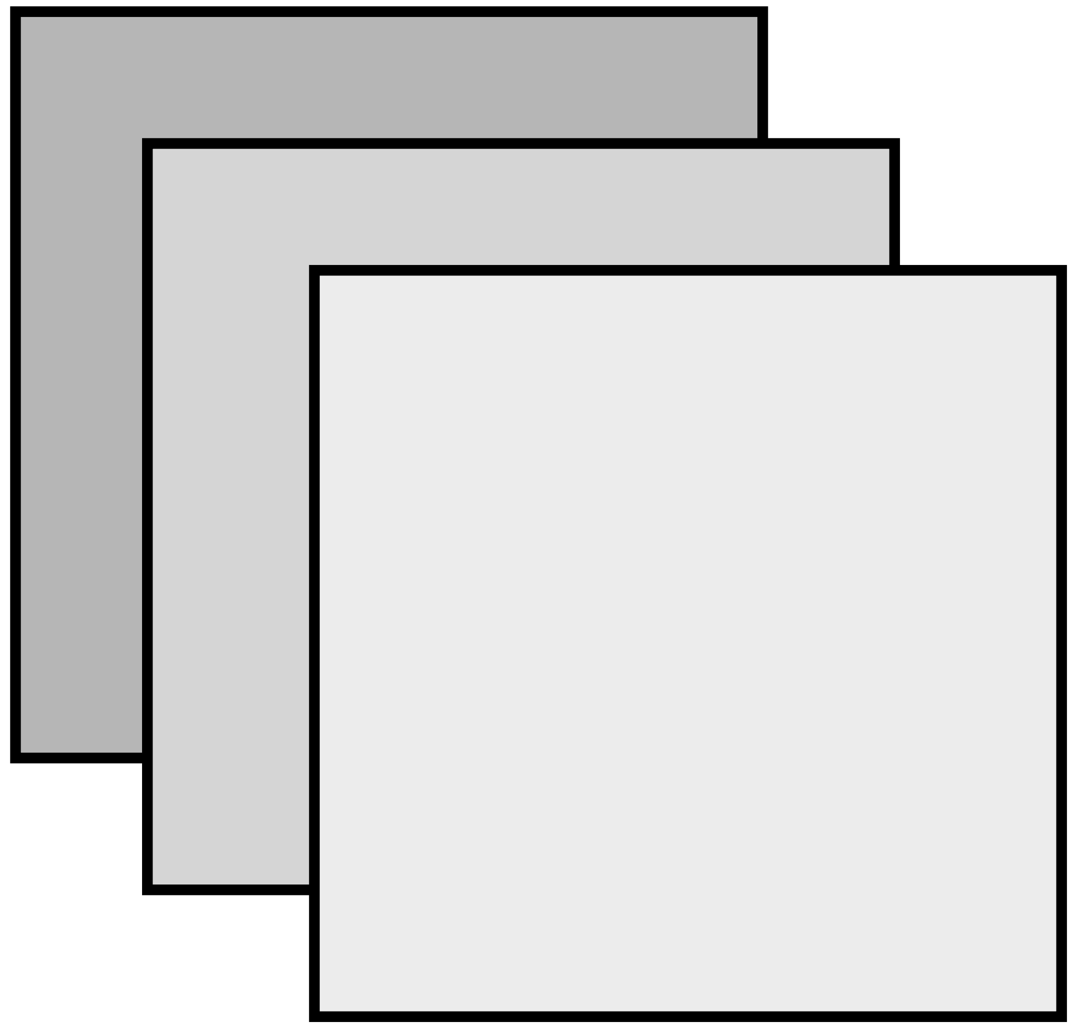
3@64x64

5 output channels



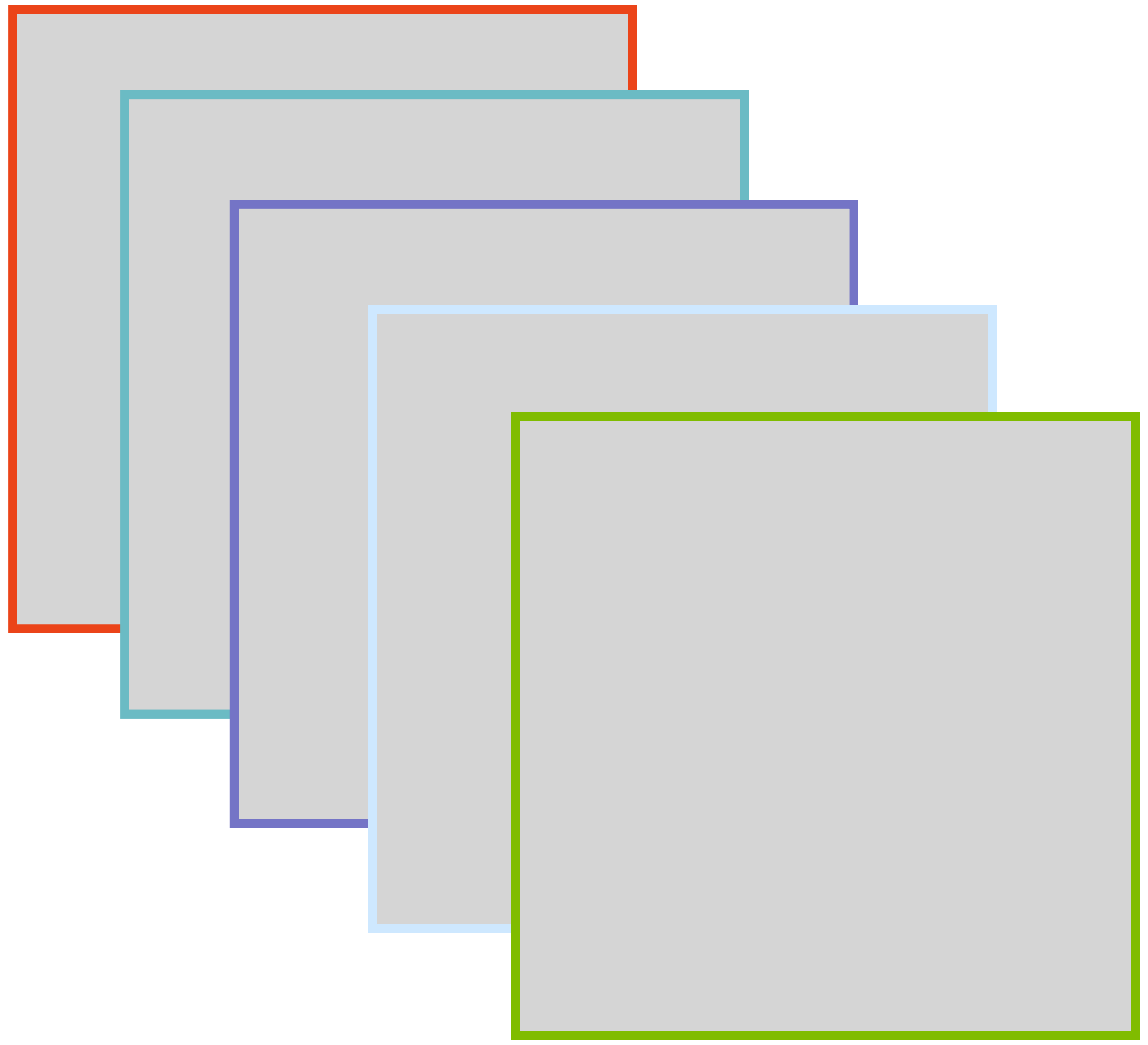
5@64x64

3 input channels



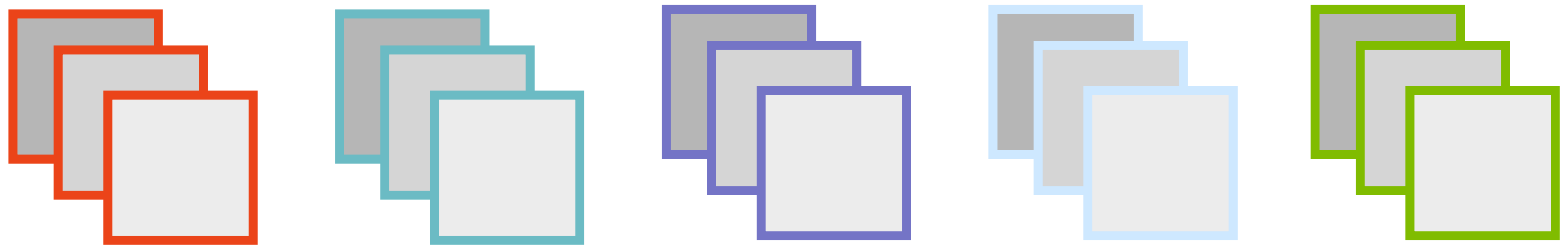
3@64x64

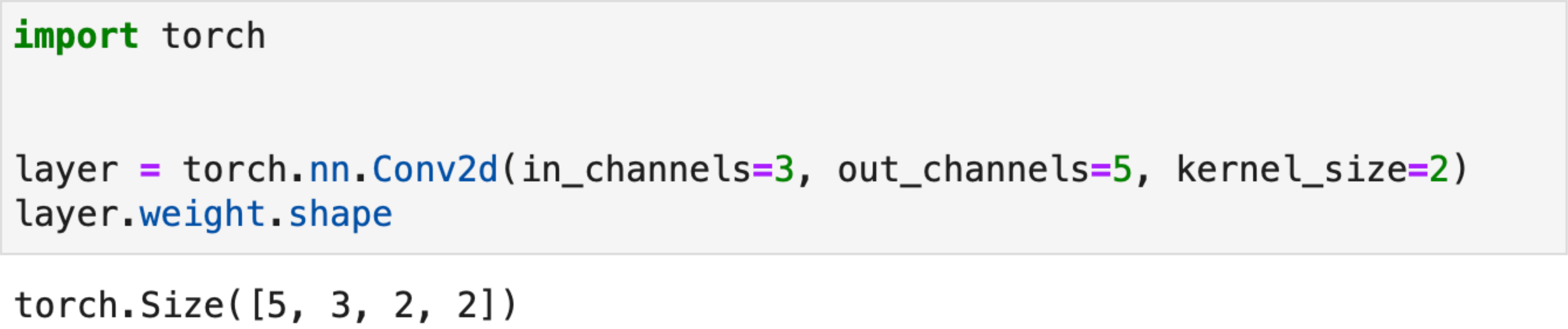
5 output channels



5@64x64

5 kernels with
3 channels each

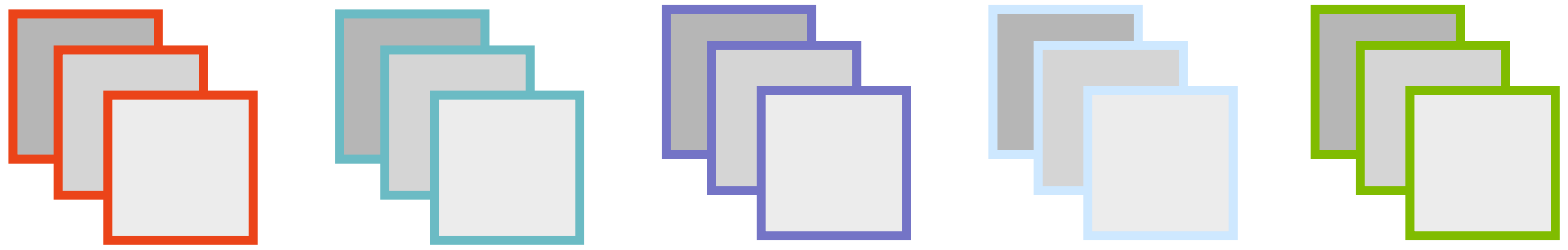




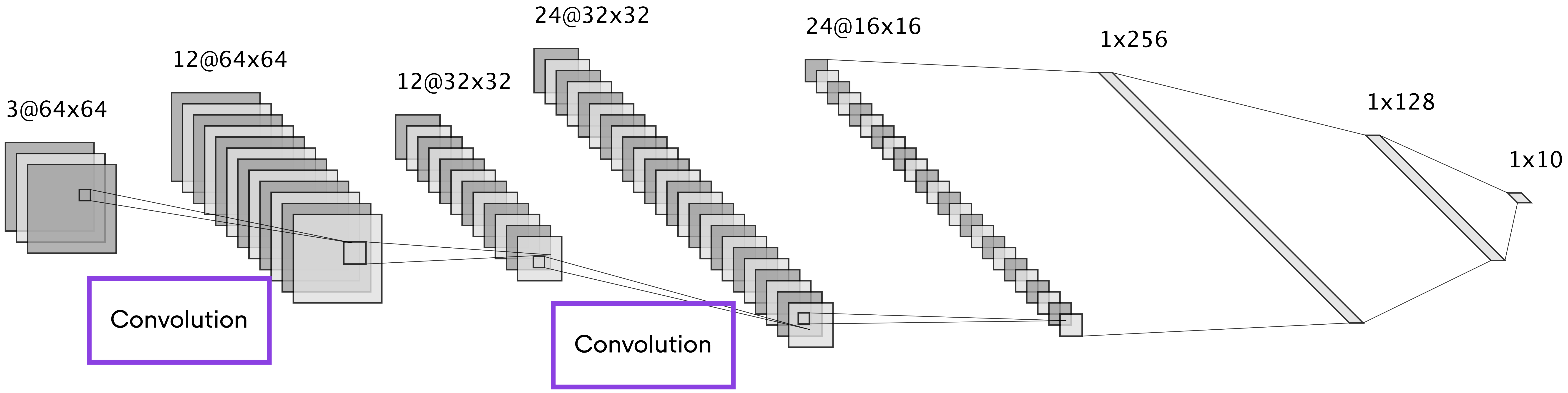
```python
import torch


layer = torch.nn.Conv2d(in_channels=3, out_channels=5, kernel_size=2)
layer.weight.shape
```

```
torch.Size([5, 3, 2, 2])
```

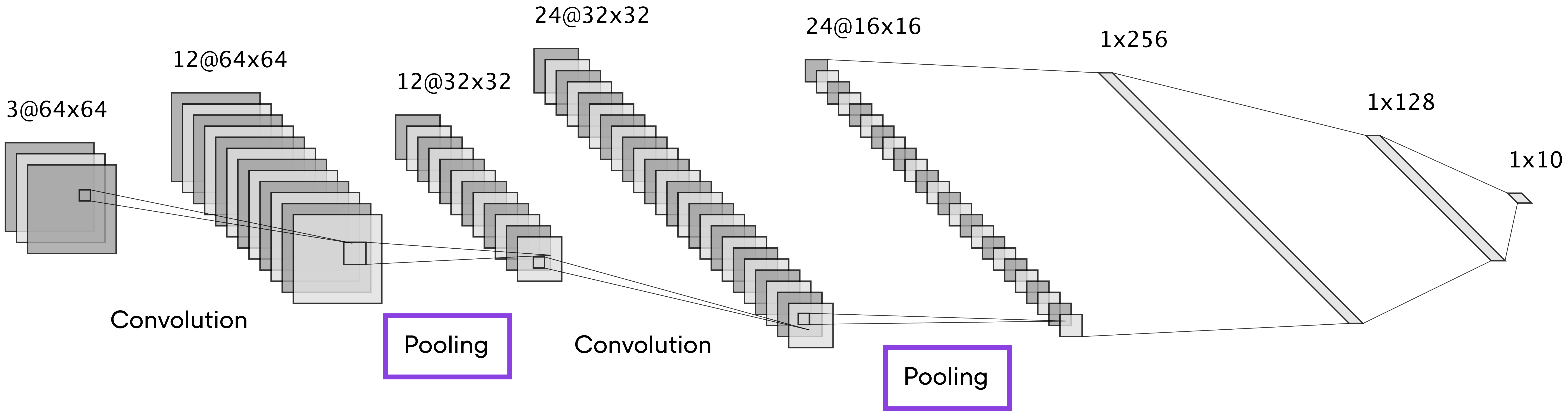
5 kernels with
3 channels each



We covered convolutional layers with multiple channels



Next: What are these pooling layers?


3@64x64
12@64x64
12@32x32
24@32x32
24@16x16
1x256
1x128
1x10
Convolution
Pooling
Convolution
Pooling