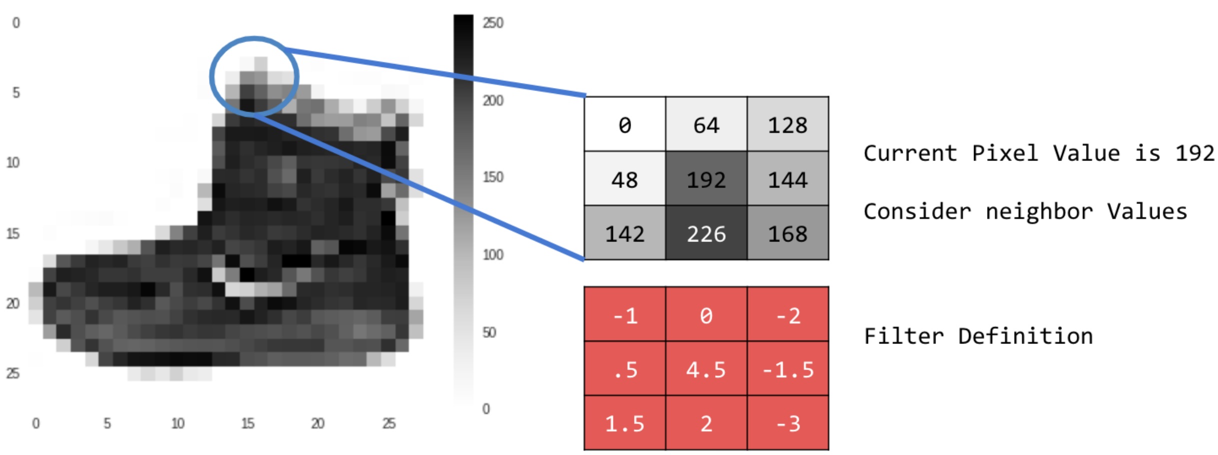
Convolution: Part 1

A **convolution** is a filter that passes over an image, processes it, and extracts features that show a commonolatity in the image.

Generating convolutions is very simple -- we simply scan every pixel in the image and then look at its neighboring pixels. We multiply out the values of these pixels by the equivalent weights in a filter.

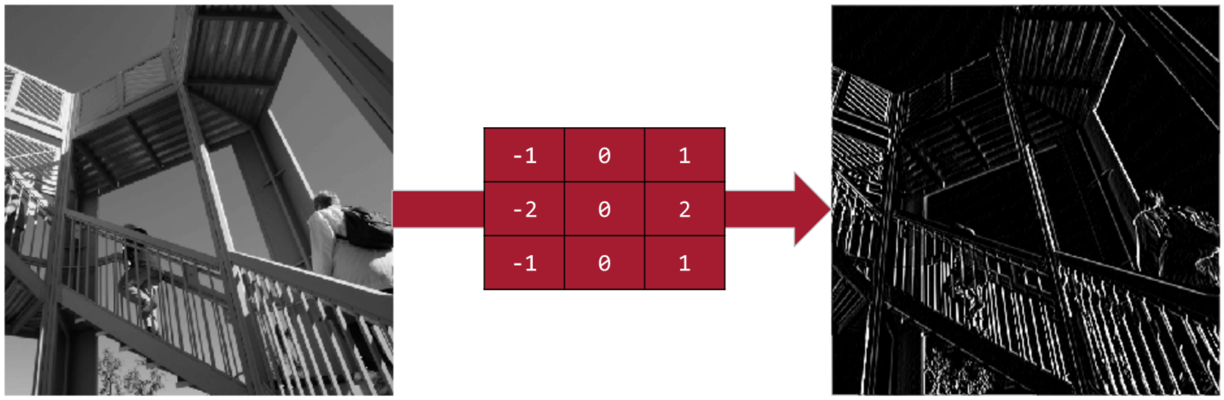
So, for example, consider a 3x3 Convolution:



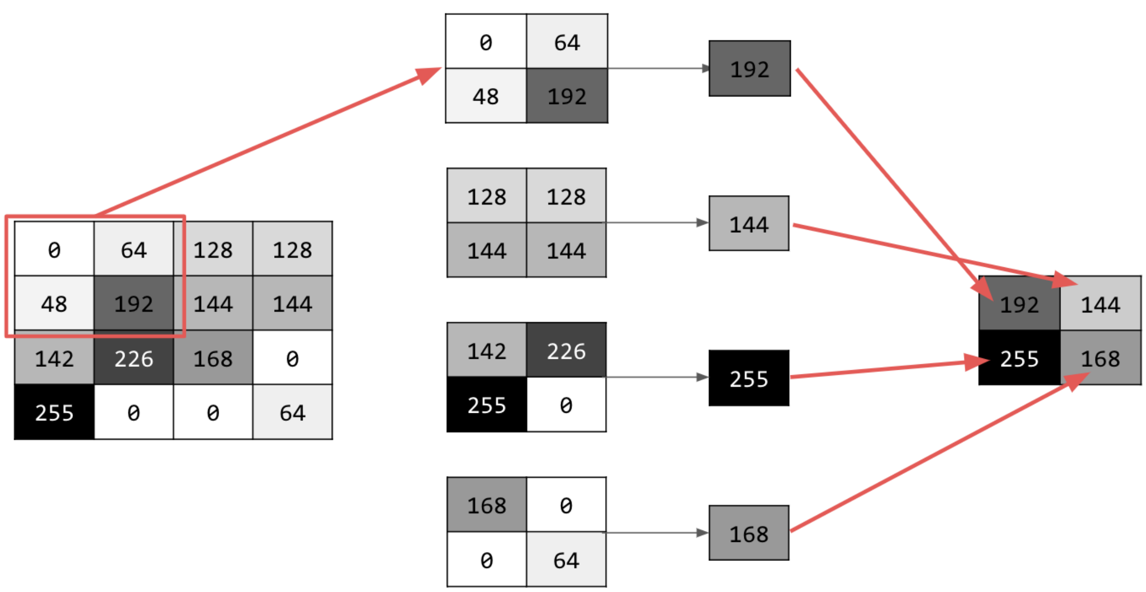

```
CURRENT_PIXEL_VALUE = 192
NEW_PIXEL_VALUE = (-1 * 0) + (0 * 64) + (-2 * 128) +
                  (.5 * 48) + (4.5 * 192) + (-1.5 * 144) +
                  (1.5 * 42) + (2 * 226) + (-3 * 168)
```

The current pixel value is 192. We calculate the new pixel value by multiplying each pixel in the 3x3 neighborhood of *pixel 192* by corresponding values in the 3x3 filter, and then summing all the individual products.

Convolutions can be used to extract certain features from an image. For example, in the below figure we see a basic filter that removes most of the information from the image except for vertical lines:



Another important concept - **pooling** also helps in detecting features. The goal of pooling is to reduce the overall amount of information in an image, while maintaining the features that are detected as present. There are different types of pooling, the most popular being MAX pooling. The idea here is to iterate over the image, and look at the pixel and its immediate neighbors to the right, below, and right-below. Take the largest (hence, the name MAX pooling) value out of them and load it into the new image. Thus, the new image will be 1/4 the size of the old image -- with the dimensions on X and Y being halved by this process. You'll see that the features get maintained despite this compression!



Therefore, when convolution is combined with pooling, an image like the one above (two images above) could be compressed *and* could have the features enhanced.