Filters as pattern detectors

D.A. Forsyth,

University of Illinois at Urbana Champaign

Some slides adapted from
Svetlana Lazebnik, who adapted from Alyosha Efros, Derek Hoiem

Filters are dot products

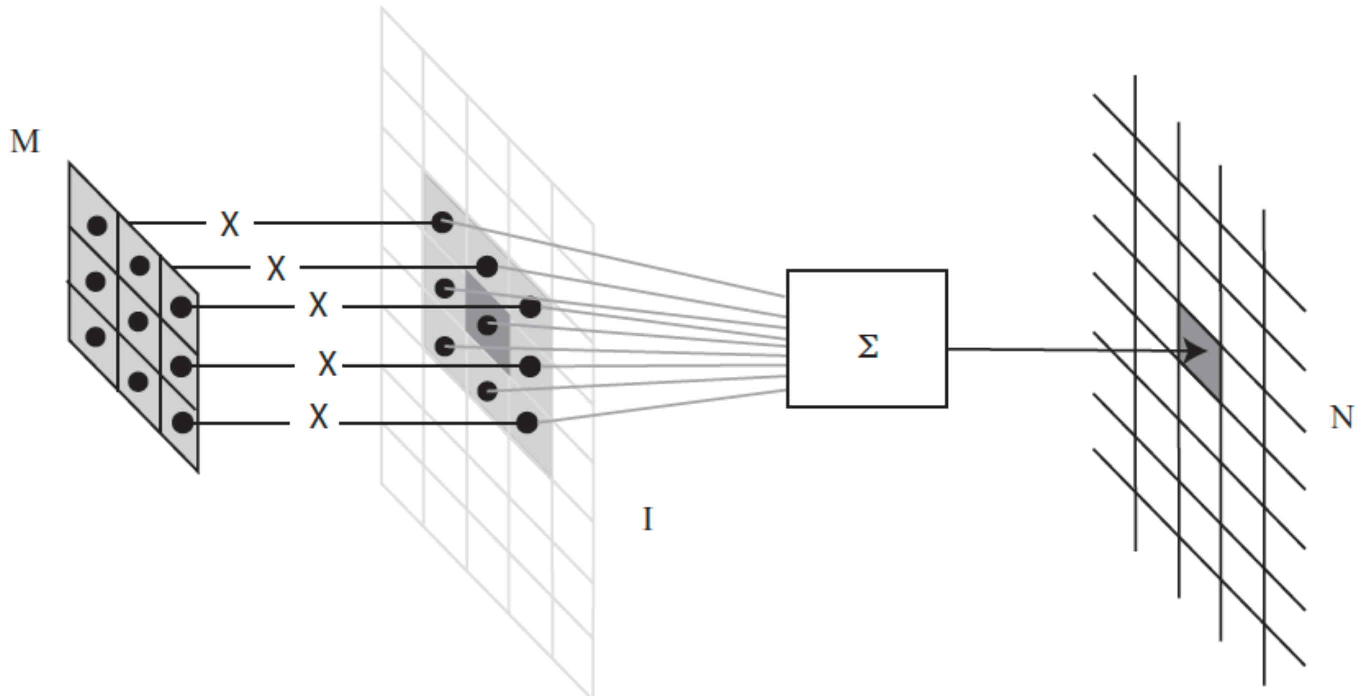


FIGURE 3.1: *To compute the value of $\mathcal{N}$ at some location, you shift a copy of $\mathcal{M}$ (the flipped version of $\mathcal{W}$) to lie over that location in $\mathcal{I}$; you multiply together the non-zero elements of $\mathcal{M}$ and $\mathcal{I}$ that lie on top of one another; and you sum the results.*

Filters as pattern detectors

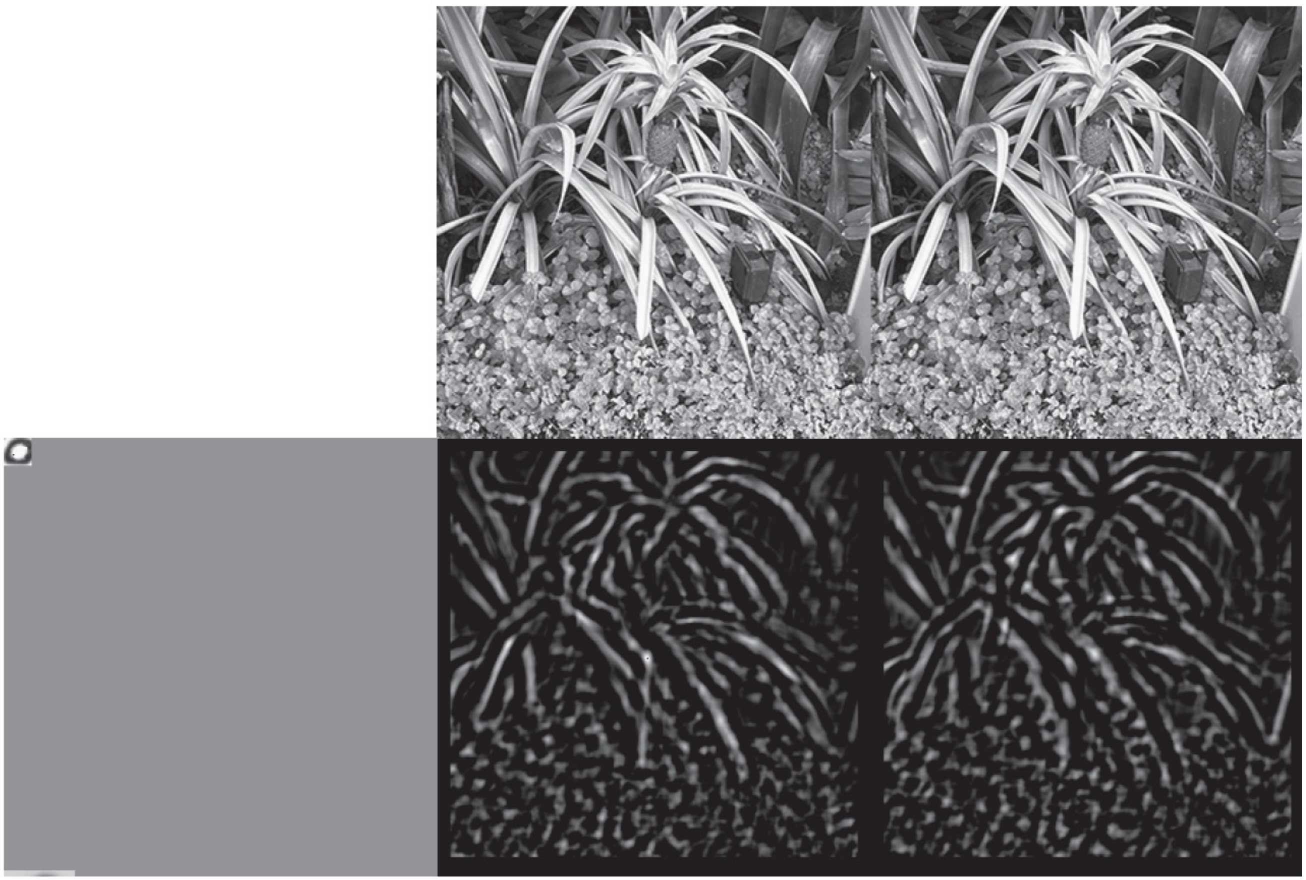
6.1.1 Pattern Detection by Convolution

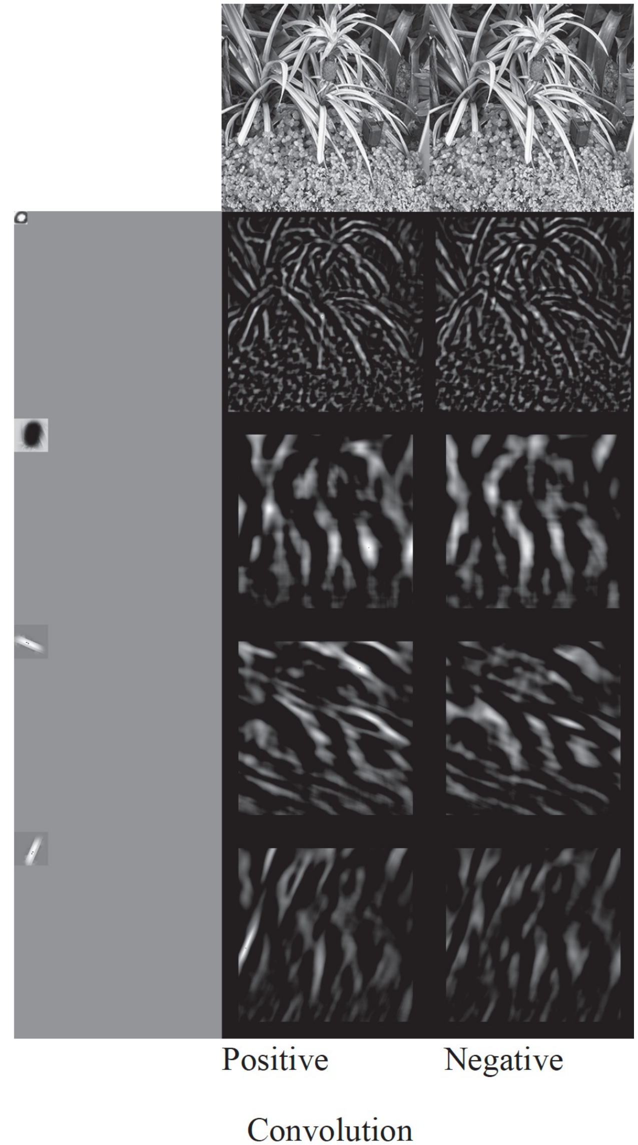
The properties of a dot product explain why a convolution is interesting: it can be used as a very simple pattern detector. Recall that if $\mathbf{u}$ and $\mathbf{v}$ are unit vectors, then $\mathbf{u}^T\mathbf{v}$ is maximized when $\mathbf{u} = \mathbf{v}$ and minimized when $\mathbf{u} = -\mathbf{v}$. Interpreting $\mathbf{u}$ as a vector of kernel weights and $\mathbf{v}$ as a vector of image values suggests the rough rule of thumb: filters respond most strongly to image patterns that look like the filter kernel.

The mean of $\mathbf{v}$ presents an issue. Write $\mathbf{1}$ for a vector of ones. Then $\mathbf{u}^T(\mathbf{v} + c\mathbf{1}) = \mathbf{u}^T\mathbf{v} + c\mathbf{u}^T\mathbf{1}$, so you can increase or decrease the response of the filter by adding a constant to the image window *unless* $\mathbf{u}^T\mathbf{1} = 0$. This suggests that the best pattern detection is obtained by using a filter with zero mean. If $\mathbf{u}^T\mathbf{1} = 0$, the magnitude of $\mathbf{u}$ just changes the scale of the response to the filter. The local maxima (or minima) of the response are what is important – these signal where a pattern is present – and so the magnitude of $\mathbf{u}$ doesn't really matter.

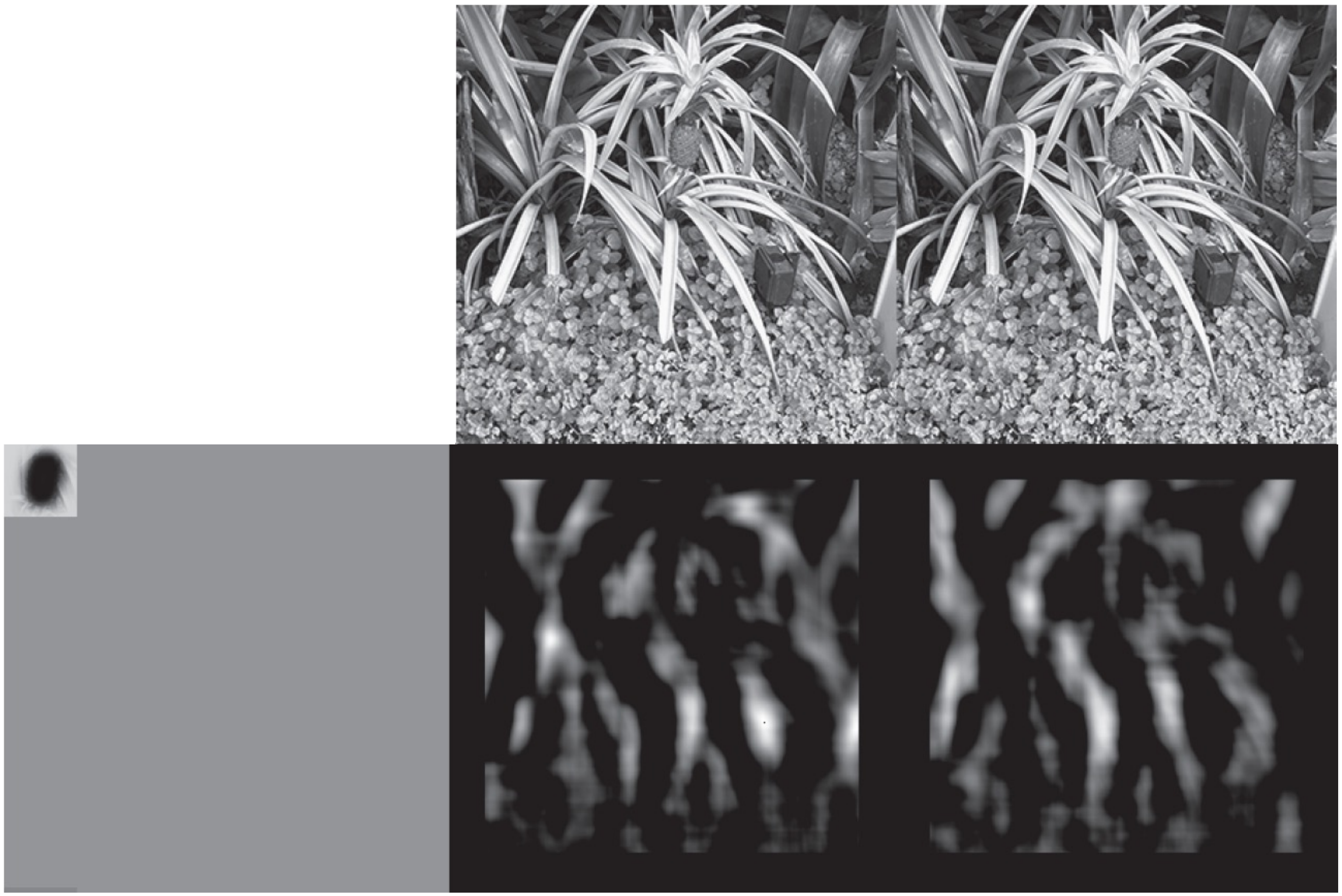
Useful Fact: *A zero-mean filter is a pattern detector that responds positively to image patches that look like it, and negatively to patches that look like it with a contrast reversal*

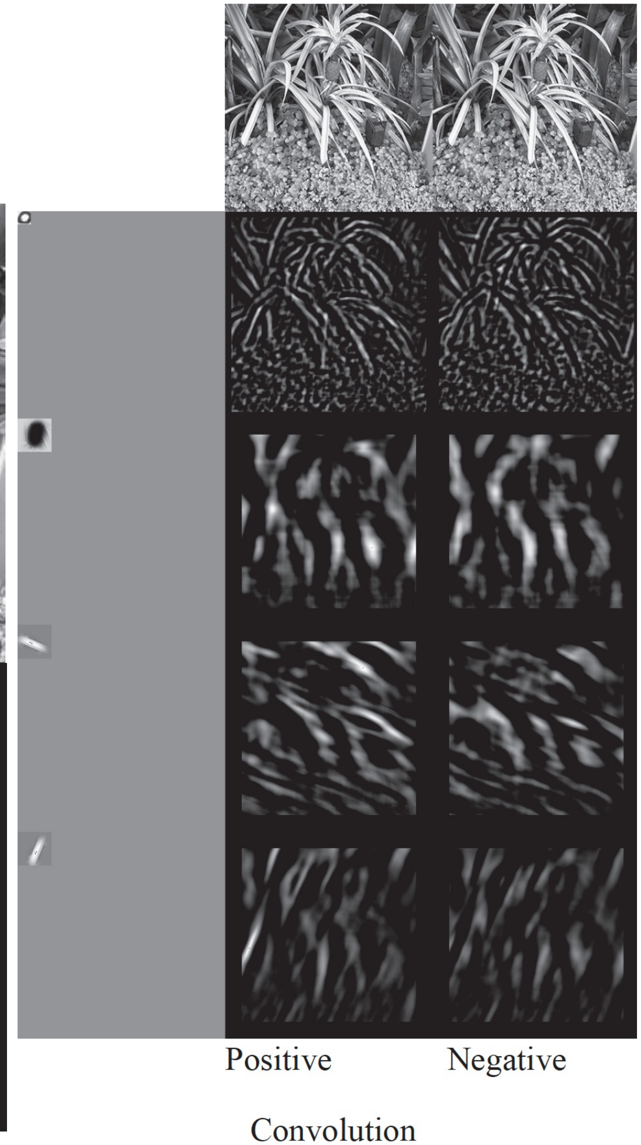
Filters detect patterns





Filters detect patterns



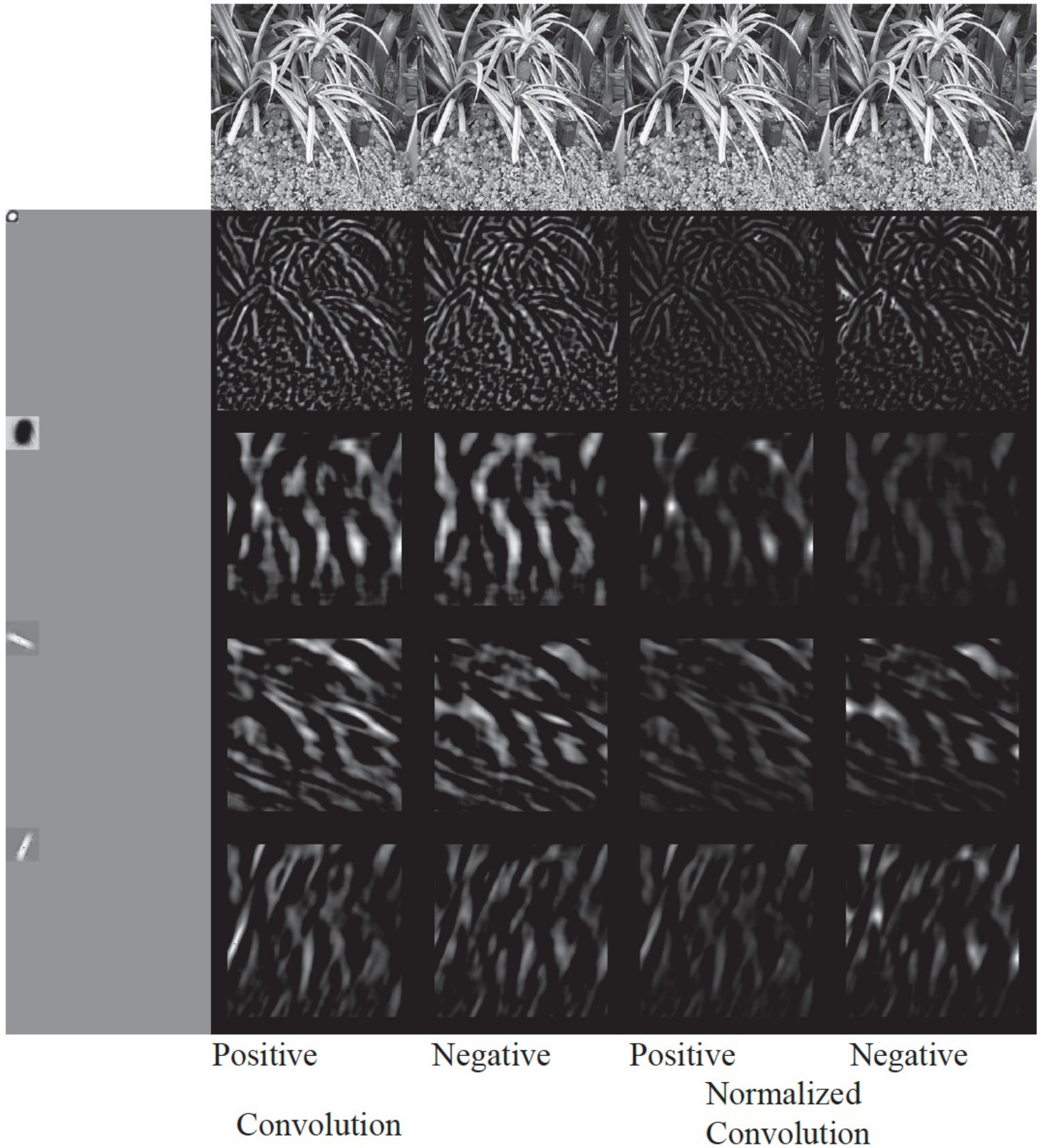


Normalized convolution

If the mean of the kernel is zero, scaling the image will scale the value of the convolution. If you test the convolution value against a threshold to find a pattern, you will find more instances when the image gets brighter, and fewer when it gets darker, which is usually inconvenient. One strategy to build a somewhat better pattern detector is to normalize the result of the convolution to obtain a value that is unaffected by scaling the image. For example, smooth the image with a Gaussian to obtain $\mathcal{G}$, then form

$$\mathcal{C}_{ij} = \frac{\mathcal{N}_{ij}}{\mathcal{G}_{ij} + \epsilon}$$

(remember, $\mathcal{N}_{ij}$ was obtained by convolving the image with some zero-mean kernel). Here $\mathcal{G}$ is an estimate of how bright the image is. Most images have all positive pixels (a zero pixel value is usually a sign of camera problems) so using ϵ to avoid dividing by zero isn't essential. But note that $\epsilon > 0$ causes the score to saturate if the image is very dark. This makes sense because a group of very dark pixels is more likely to have a pattern present through thermal noise. The process that produces $\mathcal{C}$ is known as *normalized convolution*, and produces an improvement in the detector.



Positive
Negative
Positive
Negative
Convolution
Normalized
Convolution

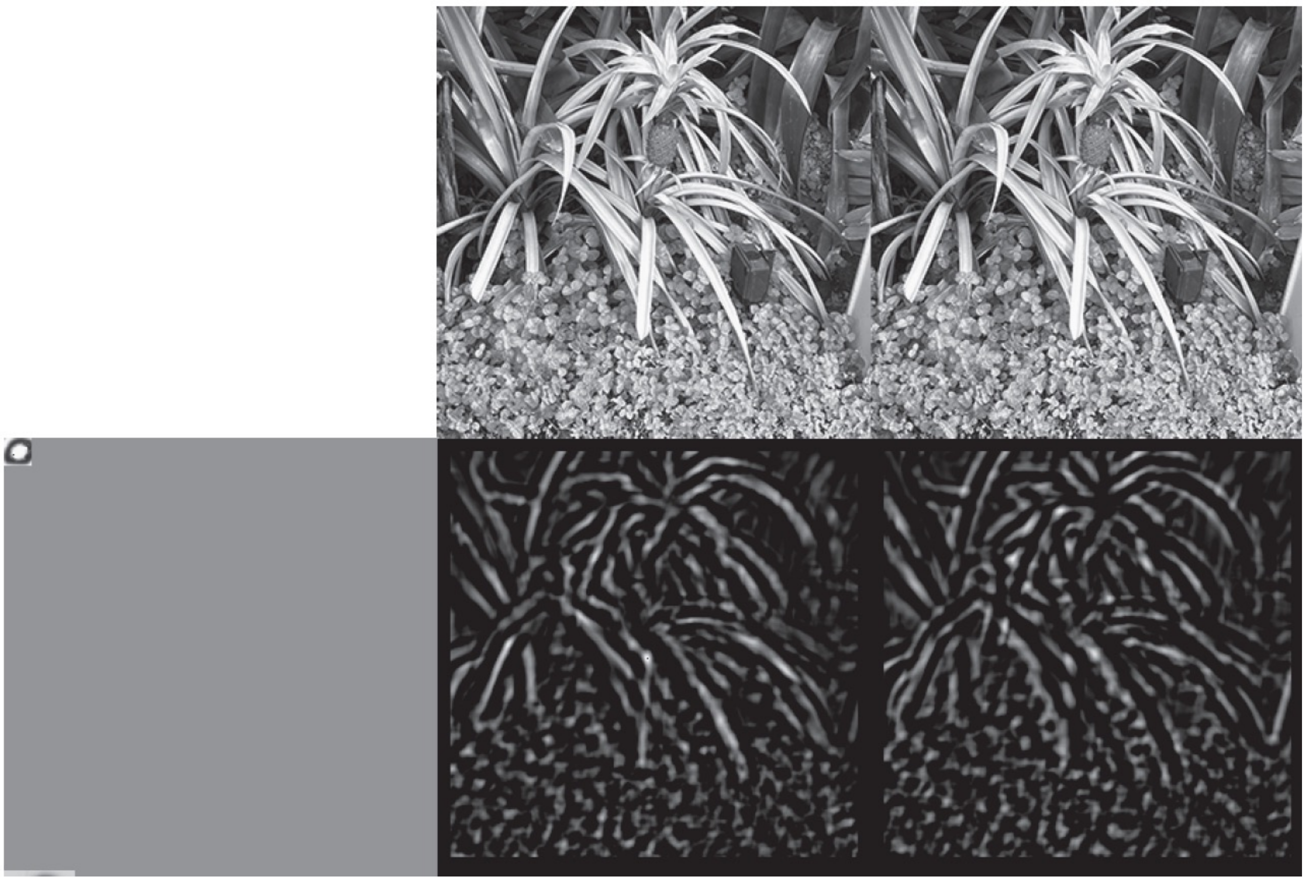
ReLUs

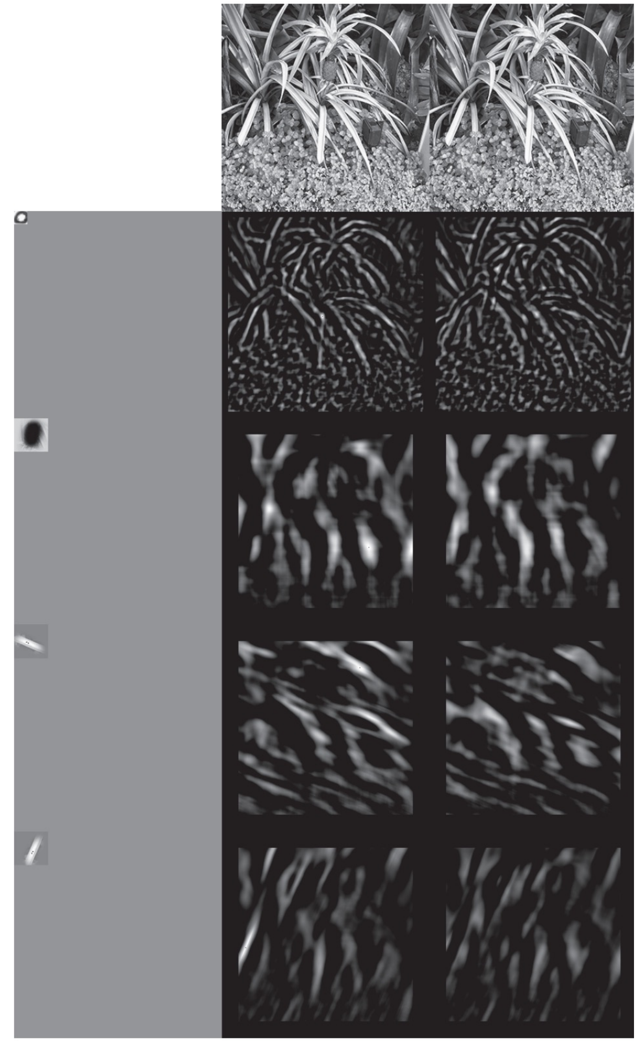
Write $\mathcal{W}$ for a kernel representing some pattern you wish to find. Assume that $\mathcal{W}$ has zero mean, so that the filter gives zero response to a constant image. Notice that $\mathcal{N} = \mathcal{W} * \mathcal{I}$ is strongly positive at locations where $\mathcal{I}$ looks like $\mathcal{W}$, and strongly negative when $\mathcal{I}$ looks like a contrast reversed (so dark goes to light and light goes to dark) version of $\mathcal{W}$. Usually, you would want to distinguish between (say) a light dot on a dark background and a dark dot on a light background. Write

$$\texttt{relu}(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

(often called a *Rectified Linear Unit* or more usually *ReLU*). Then $\texttt{relu}(\mathcal{W} * \mathcal{I})$ is a measure of how well $\mathcal{W}$ matches $\mathcal{I}$ at each pixel, and $\texttt{relu}(-\mathcal{W} * \mathcal{I})$ is a measure of how well $\mathcal{W}$ matches a contrast reversed $\mathcal{I}$ at each pixel. The ReLU will appear again.

Filters detect patterns





Multi-channel convolution

The description of convolution anticipates monochrome images, and Figure 4.3 shows filters applied to a monochrome image. Color images are naturally 3D objects with two spatial dimensions (up-down, left-right) and a third dimension that chooses a slice or *channel* (R, G or B for a color image). Color images are sometimes called *multi-channel images*. Multi-channel images offer a natural for representations of image patterns, too — two dimensions that tell you where the pattern is and one that tells you what it is. For example, the results in Figure 4.3 can be interpreted as a block consisting of eight channels (four patterns, original contrast and contrast reversed). Each slice is the response of a pattern detector *for a fixed pattern*, where there is one response for each spatial location in the block, and so are often called *feature maps* (it is entirely fair, but not usual, to think of an RGB image as a rather uninteresting feature map).

Multi-channel Convolution

For a color image $\mathcal{I}$, write $\mathcal{I}_{k,ij}$ for the k'th color channel at the i, j'th location, and $\mathcal{K}$ for a color kernel – one that has three channels. Then interpret $\mathcal{N} = \mathcal{I} * \mathcal{K}$ as

$$\mathcal{N}_{ij} = \sum_{kuv} \mathcal{I}_{k,i-u,j-v} \mathcal{K}_{kuv}$$

which is an image with a single channel. This $\mathcal{N}$ is a single channel image that encodes the response to a single pattern detector. Much more interesting is an encoding of responses to multiple pattern detectors, and for that you must use multiple kernels (often known as a *filter bank*). Write $\mathcal{K}^{(l)}$ for the l'th kernel, and obtain a feature map

$$\mathcal{N}_{l,ij} = \sum_{kuv} \mathcal{I}_{k,i-u,j-v} \mathcal{K}^{(l)}_{kuv}.$$

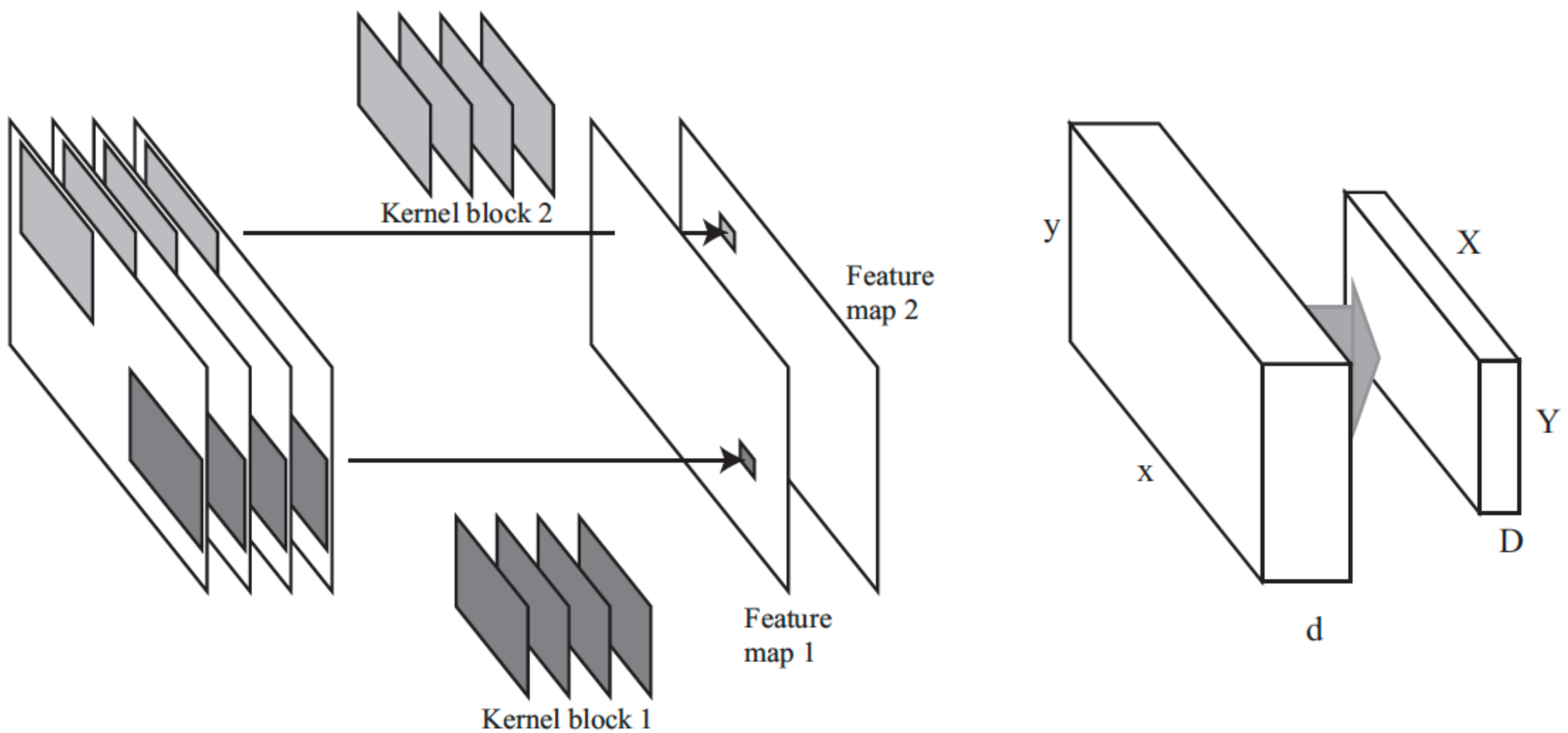
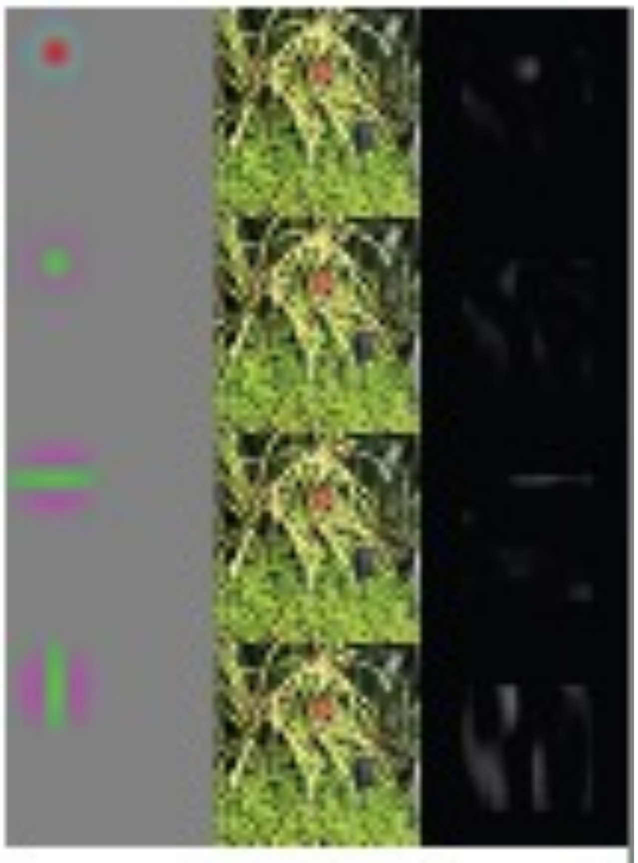


FIGURE 4.4: *On the* **left**, *two kernels (now 3D, as in the text) applied to a set of feature maps produce one new feature map per kernel, using the procedure of the text (the bias term isn't shown). Abstract this as a process that takes an* $x \times y \times d$ *block to an* $X \times Y \times D$ *block (as on the* **right**).

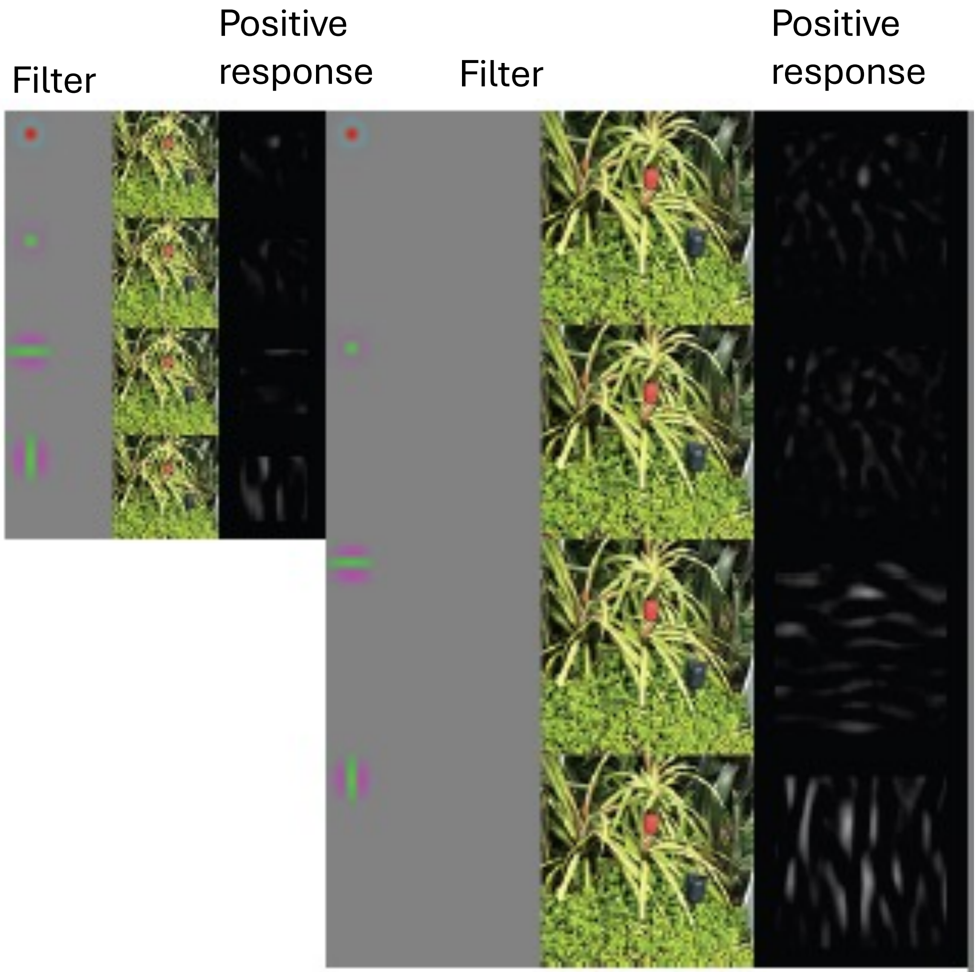
Representing Images with Filter Banks

Filter

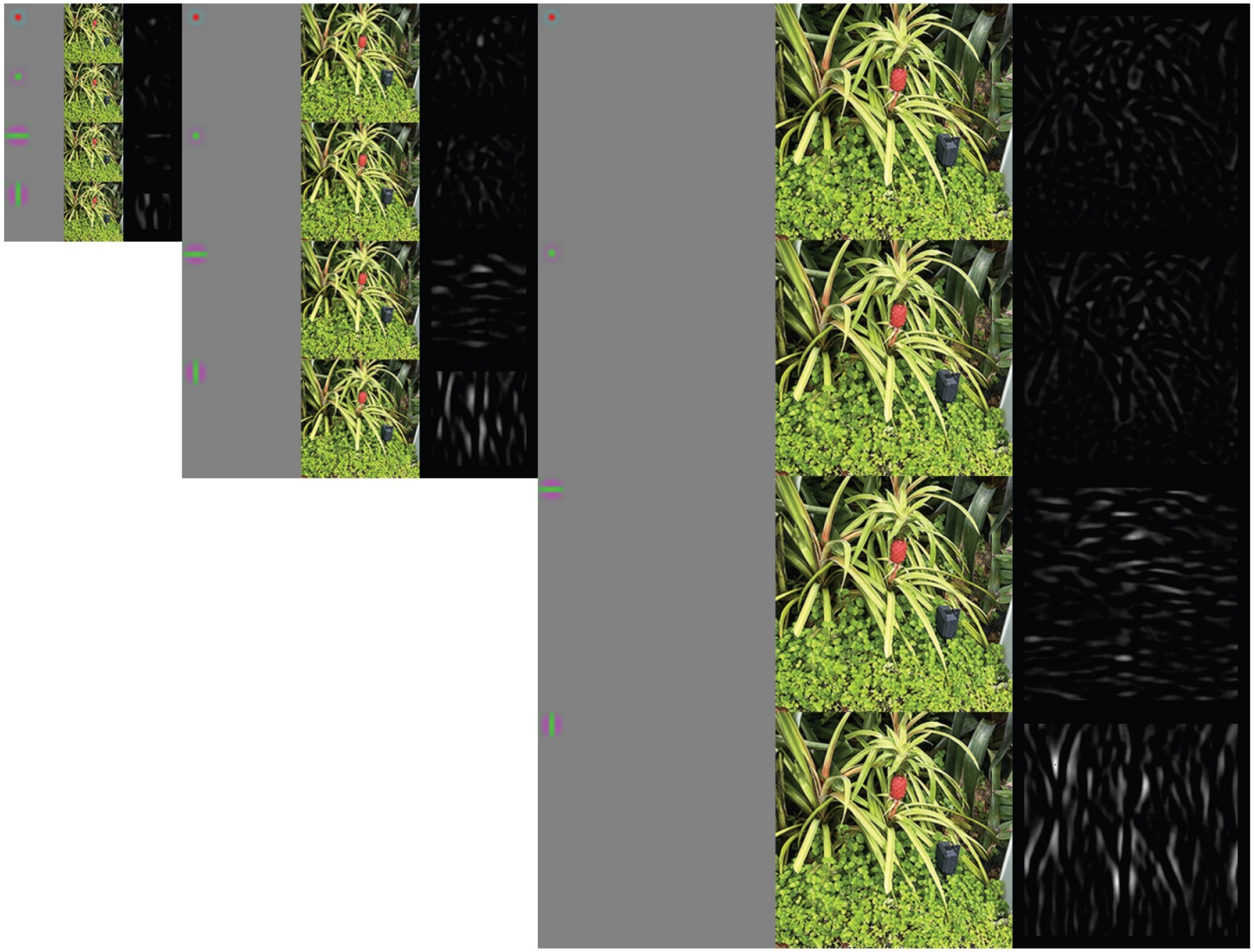
Positive response



Representing Images with Filter Banks



Representing Images with Filter Banks



But which filters should I use?

- Up till about 2012:
  - choose some, mostly spots and bars
- After 2012:
  - lots; choose ones that work well in your application using an optimization procedure

Think about this…

6.2. Why is a non zero-mean filter a poor choice of pattern detector?
6.3. Why is a normalized convolution useful?
6.4. Why is a normalized convolution useful?
6.5. Why does “subtracting a small constant from the response before applying the ReLU” help suppress small responses to a pattern detector?
6.6. Why does “subtracting a small constant from the response before applying the ReLU” (Section 6.1.3) help suppress small responses to a pattern detector?
6.7. Is normalized convolution linear in the convolution kernel?
6.8. Is normalized convolution linear in the image?
6.9. Is multichannel convolution linear in the convolution kernel?
6.10. Is multichannel convolution linear in the image?
6.11. Can you normalize multichannel convolution?
6.12. Can you construct a zero-mean kernel for multichannel convolution?