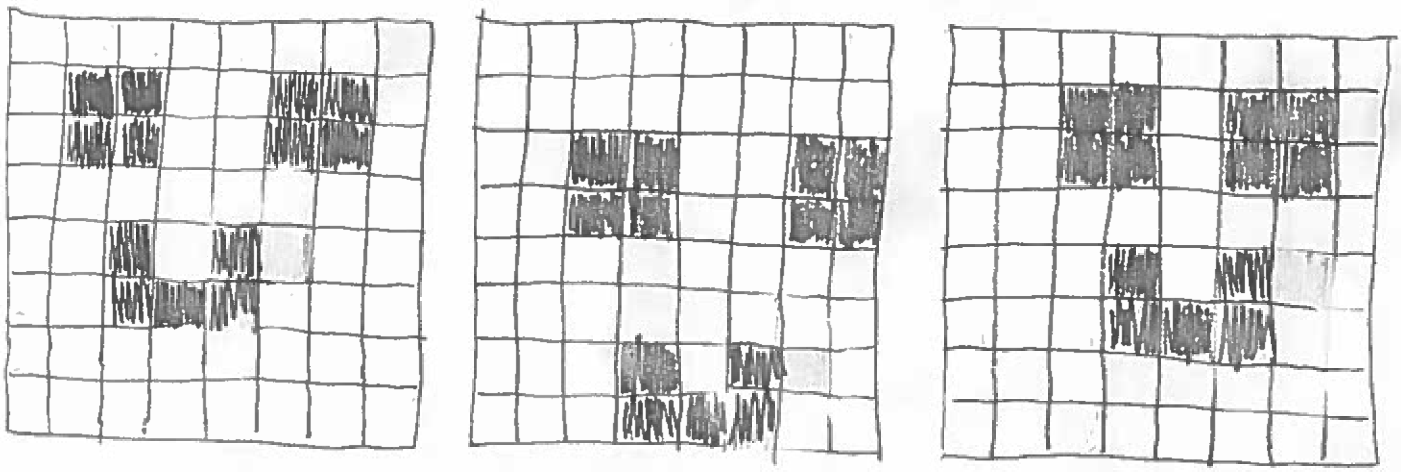
CNNs for Images

① Imagine we scanned in various drawings of happy faces (at a resolution of 8 pixels by 8 pixels):



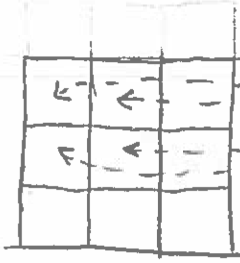
② To develop a CNN for happy face detection, we need to develop two convolution kernels: an "eye detector" and a "smile detector." We'll apply these kernels in the following way:

$$\underbrace{a}_{\substack{\text{ReLU}\\ \text{activation}\\ \text{function}}}\left(\text{sum}\left(\underbrace{\begin{bmatrix} \ & \ & \ \\ \ & \ & \ \\ \ & \ & \ \end{bmatrix}}_{\substack{\text{image}\\ \text{window}}} \overset{\substack{\text{Hadamard (elementwise)}\\ \text{product}}}{\oplus} \underbrace{\begin{bmatrix} \ & \ & \ \\ \ & \ & \ \\ \ & \ & \ \end{bmatrix}}_{\substack{\text{convolution}\\ \text{kernel}}}\right) - \text{offset}\right)$$

CNNS FOR IMAGES

③ For instance, to detect an eye:

we want the other pixels to equal 0 (i.e. white)



we want these pixels to equal 1 (i.e. black)

④ So we can build an eye detector as follows:

$$a\left(\text{sum}\left(\begin{bmatrix} \ & \ & \ \\ \ & \ & \ \\ \ & \ & \ \end{bmatrix} \oplus \begin{bmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix}\right) - 3\right)$$

⑤ For an eye:

$$a\left(\text{sum}\left(\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix}\right) - 3\right)$$

$$= a\left(\text{sum}\left(\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) - 3\right)$$

$$= a(4-3)$$

$$= a(1)$$

$$= 1$$ ← which is positive!

CNNs for Images

⑥ Non-eyes evaluate to zero:

$$a\left(\text{sum}\left(\begin{bmatrix}0&1&0\\1&1&0\\0&1&0\end{bmatrix} \oplus \begin{bmatrix}1&1&-1\\1&1&-1\\-1&-1&-1\end{bmatrix}\right)-3\right)$$

$$= a\left(\text{sum}\left(\begin{bmatrix}0&1&0\\1&1&0\\0&-1&0\end{bmatrix}\right)-3\right)$$

$$= a(2-3)$$

$$= a(-1)$$

$$= 0$$

⑦ We can build a mouth detector similarly:

$$a\left(\text{sum}\left(\begin{bmatrix}\ &\ &\ \\\ &\ &\ \\\ &\ &\ \end{bmatrix} \oplus \begin{bmatrix}-1&-1&-1\\1&-1&1\\1&1&1\end{bmatrix}\right)-3\right)$$

CNNs for Images

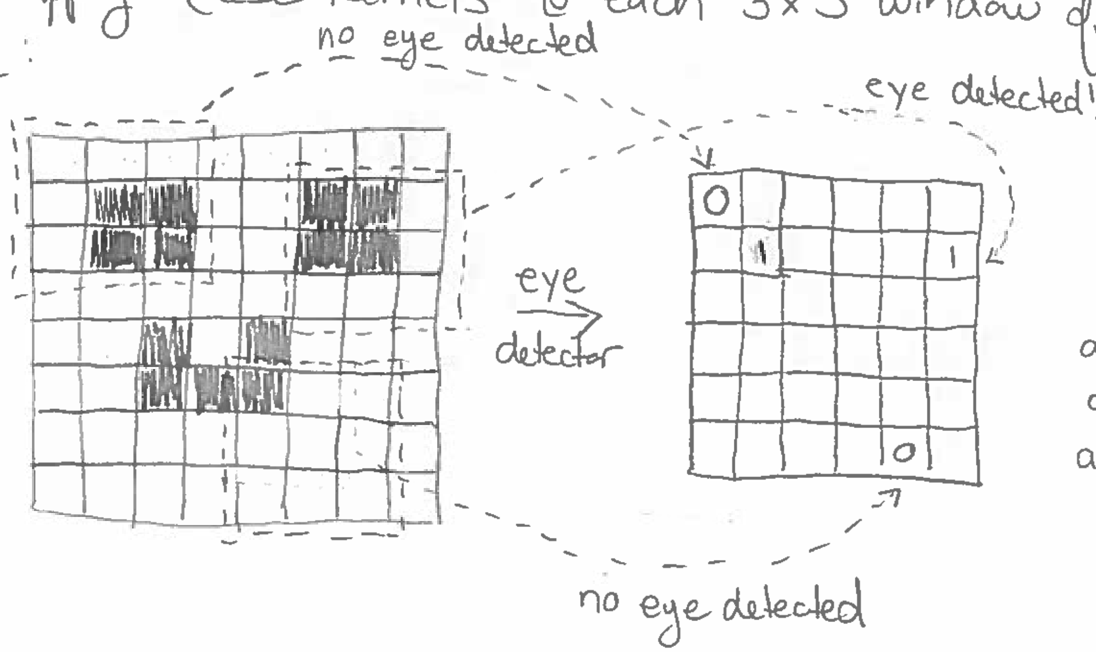
⑧ So we have two convolutional kernels, each with shape 3×3:

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

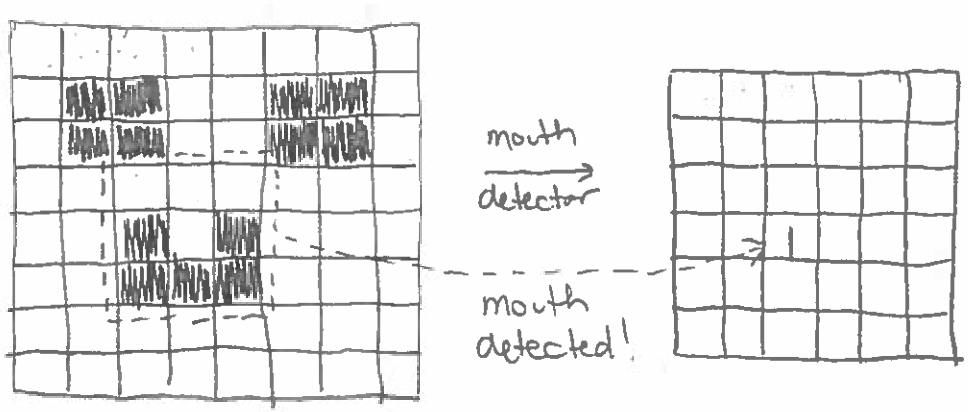
"eye detector" "mouth detector"

---

⑨ If we apply these kernels to each 3×3 window of the image:

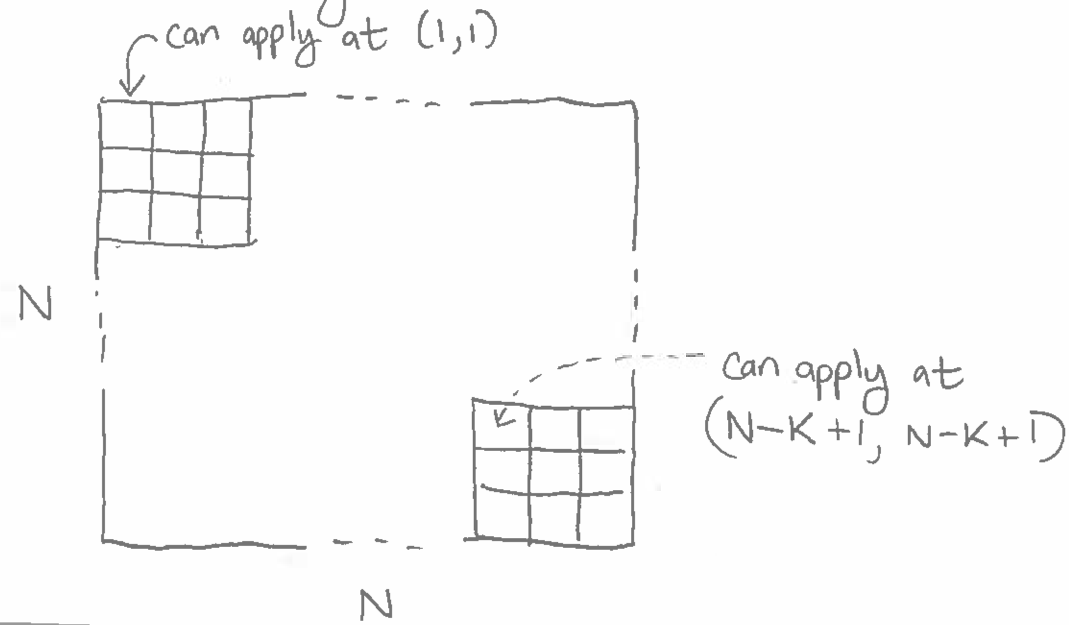


all the rest of the entries are zero

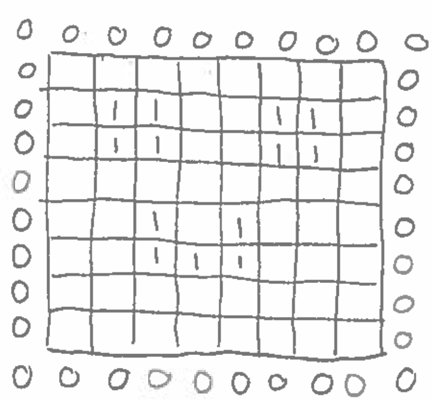


all the rest of the entries are zero

(10) You'll notice that by applying the 3x3 convolutional kernel to the 8x8 image, we end up with a 6x6 array. In general, if we apply a $K \times K$ convolutional kernel to an $N \times N$ image, we end up with an $N-K+1 \times N-K+1$ array:



(11) If we want to apply the kernels without reducing the image dimensions, we can apply a padding of zeroes around the border:



Note that a padding of width 1 turns our 8x8 image to a 10x10 image

CNNS FOR IMAGES

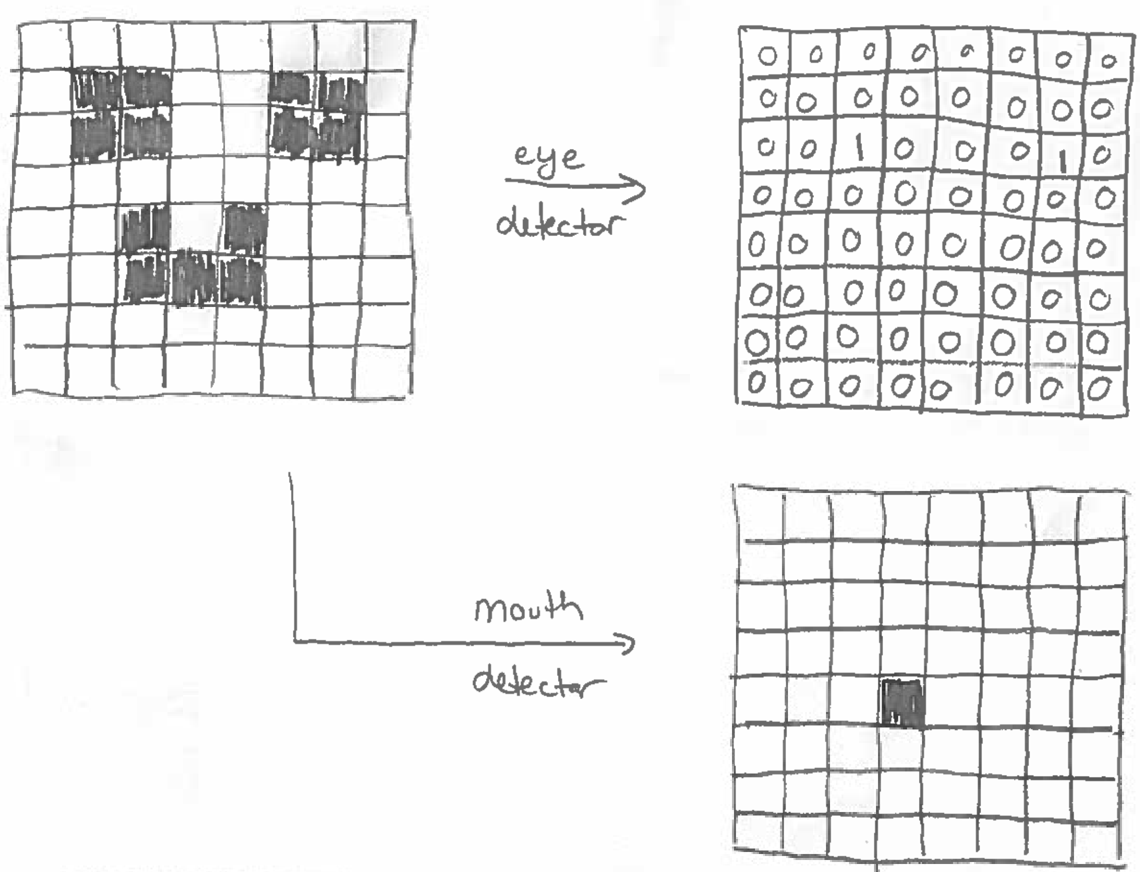
(12) Applying a 3×3 kernel to the resulting 10×10 array yields an array of size $10-3+1 \times 10-3+1$, i.e. 8×8, which is the same size as our original image.

In general, for a $K \times K$ convolutional kernel where K is odd, applying the kernel to an $N \times N$ image with padding width $\left\lfloor \frac{K}{2} \right\rfloor$ yields an array of size $N \times N$.

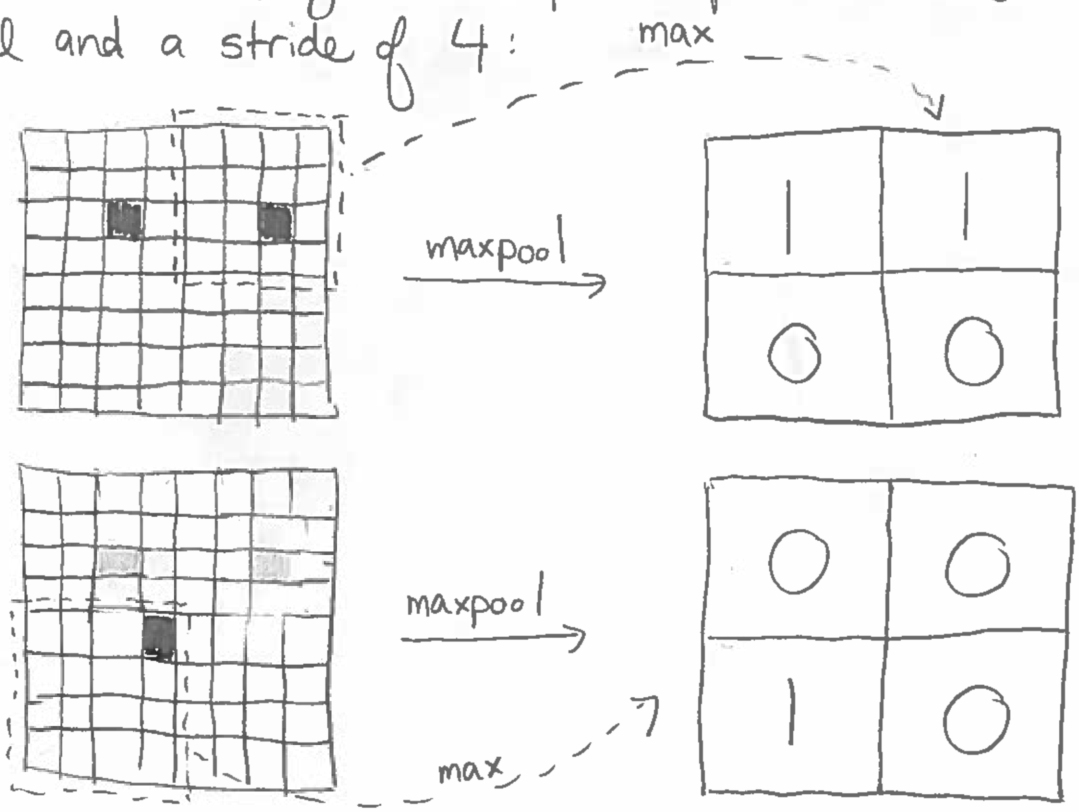
(13) Exercise: Prove the final statement from (12).

CNNS FOR IMAGES

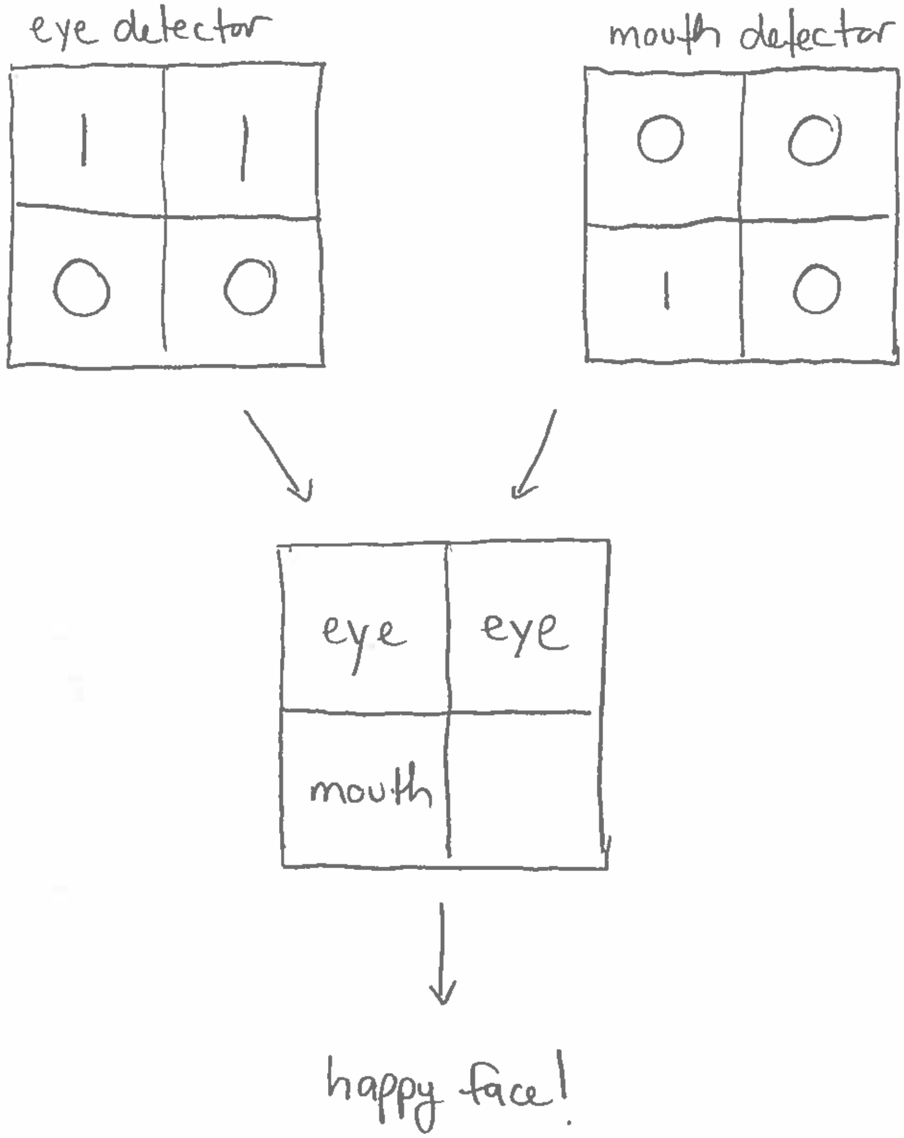
(14) If we apply our two kernels to the padded happy face:



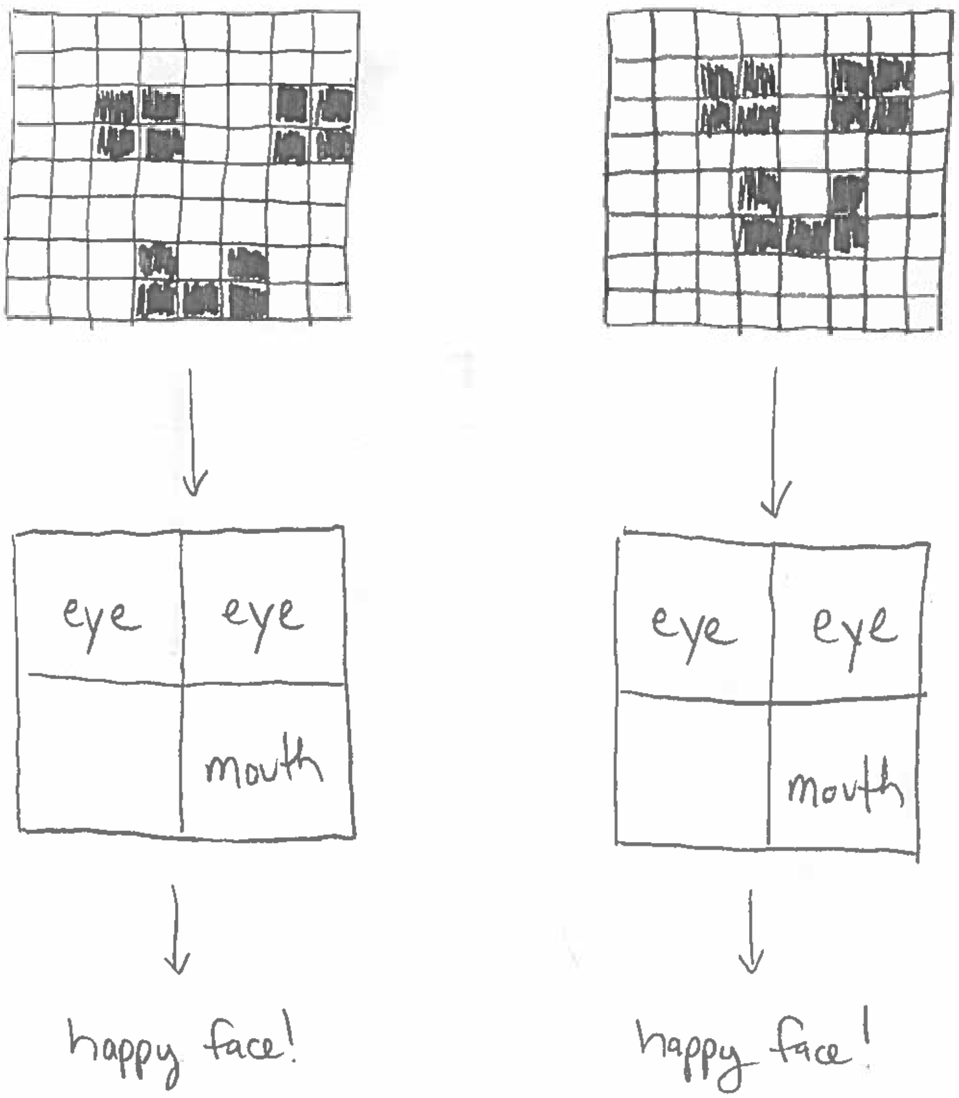
(15) Then we can apply a maxpool operation with a 4x4 kernel and a stride of 4:



(16) Our final layer then knows:



(17) Because of the pooling, the resulting CNN can detect any number of variants, as long as the eyes are in the upper left and right quadrants, and the mouth is in one of the lower quadrants:



CNNS FOR IMAGES

(18) Image CNNs get a bit arduous to draw, but we can fairly easily describe them with our established CNN terminology. The example can be described as:

- a convolutional layer with two 3×3 kernels of stride 1 and padding 1
- a maxpool layer with a 4×4 kernel of stride 4 and padding 0

(19) We can draw it at a high level of granularity: