COMP90086 Computer Vision
Week 2

Image Filtering

Lecture Notes summarized by Neo

Semester 2 2021

1 Convolution

1.1 Pixel Operator (one to one pixel mapping)

Computes an output value at each pixel location, based on the **input pixel value**.

$$g(i,j) = h(f(i,j)), \text{ where}$$

- (i,j) is the pixels of the image,
- g is the output image,
- f is the input image and
- h is the filtering function.

Example

- $g(i,j) = 0.5(f(i,j))$
  降低图片亮度，每个 Pixel 数值减半（Pixel 数越高越亮）。
- Gamma Correction 伽马矫正
  Pixel Operator 的一种，使用更为复杂的**非线性**方程。

1.2 Local Operator (many to one pixel mapping)

Computes an output value at each pixel location, based on **a neighbourhood of pixels around the input pixel**.

e.g. Sharpening （锐化）：
Finding the average color of the pixels around each pixel in a specified radius, and then contrasting that pixel from that average color.

2 Linear Filtering (used in local operator)

Output pixel’ s value is a weighted sum of a neighbourhood around the input pixel.

2.1 Cross-Correlation 互相关

$$g(i,j) = h(u,v) \oplus f(i,j), \text{ where}$$

- (i,j) is the pixels of the image,
- g is the output image,
- f is the input image,
- h is the kernel and
- $\oplus$ is the cross-correlation operator.

$$g(i,j) = \sum_{u,v} f(i+u, j+v)h(u,v)$$

2.2 Convolution 卷积

$$g(i,j) = h(u,v) * f(i,j), \text{ where}$$

- (i,j) is the pixels of the image,
- g is the output image,
- f is the input image,
- h is the kernel and
- $*$ is the convolution operator.

$$g(i,j) = \sum_{u,v} f(i-u, j-v)h(u,v)$$

Cross-Correlation Example

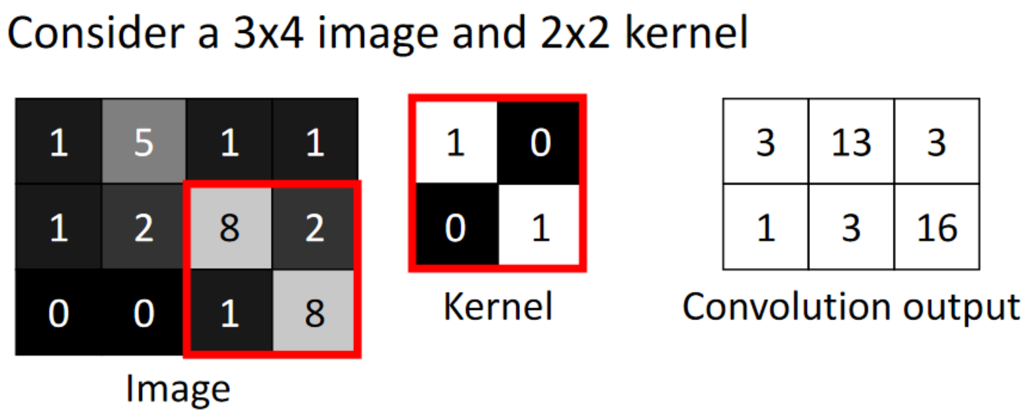


Figure 1: Cross-Correlation of 3*4 image with 2*2 kernel output 2*3 image

由于使用 Kenel 会导致原图像的边框无法转换，最后的图像会比原图像小。
右下角 Pixel：$8 * 1 + 2 * 0 + 1 * 0 + 8 * 1 = 16$

2.3 Convolution with Colour Images

在处理 RGB（或其他格式）彩色图片时，把**原图分成 3 个不同颜色的 Channel**（RGB）并分别应用不同（或相同）的 Kernel，最后把结果合并。

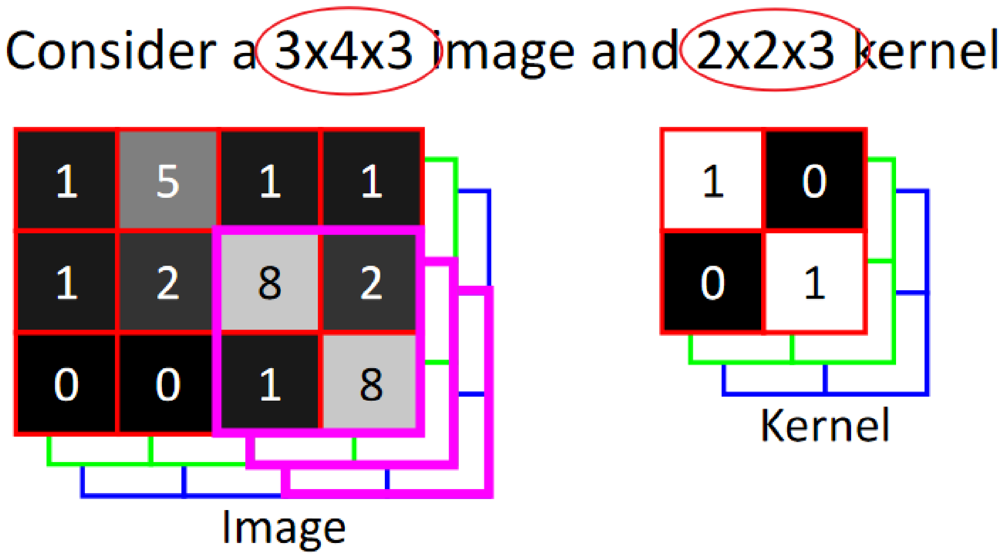


Figure 2: Separate RGB image into three channels

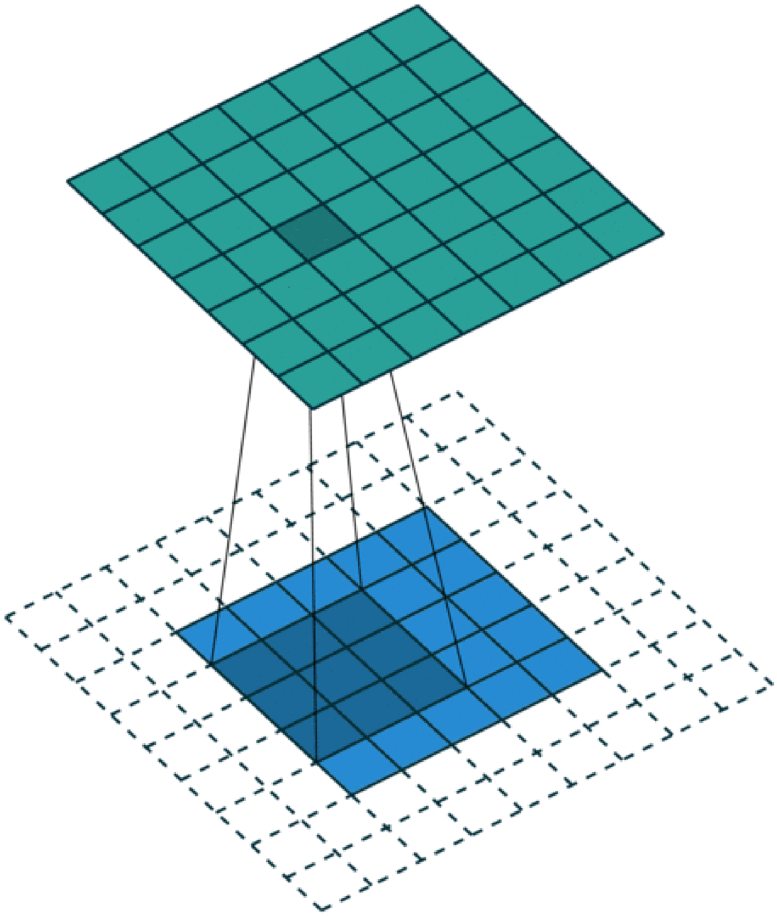


Figure 3: Example with the green grid being the original image and the blue one is the result

2.4 Cross-correlation v.s. Convolution

Overlay filter on image for Corss-correlation and **flip filter** horizontally and vertically for Convolution.
注：使用 **Convolution** 时要把 Kernel **上下前后颠倒**。

In Deep Learning, the filters in convolution are not reversed. Rigorously speaking, it' s cross-correlation. We essentially perform element-wise multiplication and addition. But it' s a convention to just call it convolution in deep learning. It is fine because the weights of filters are learned during training. If the reversed function g in the example above is the right function, then after training the learned filter would look like the reversed function g. Thus, there is no need to reverse the filter first before training as in true convolution.

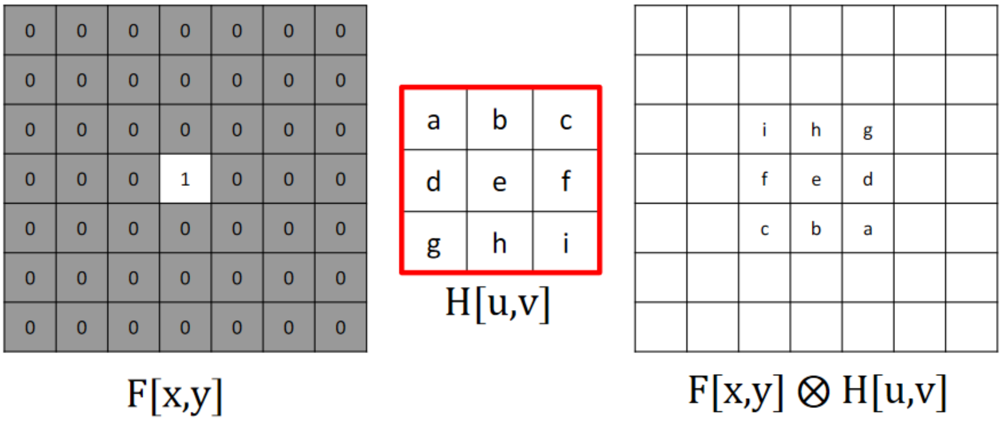


Figure 4: Cross-correlation

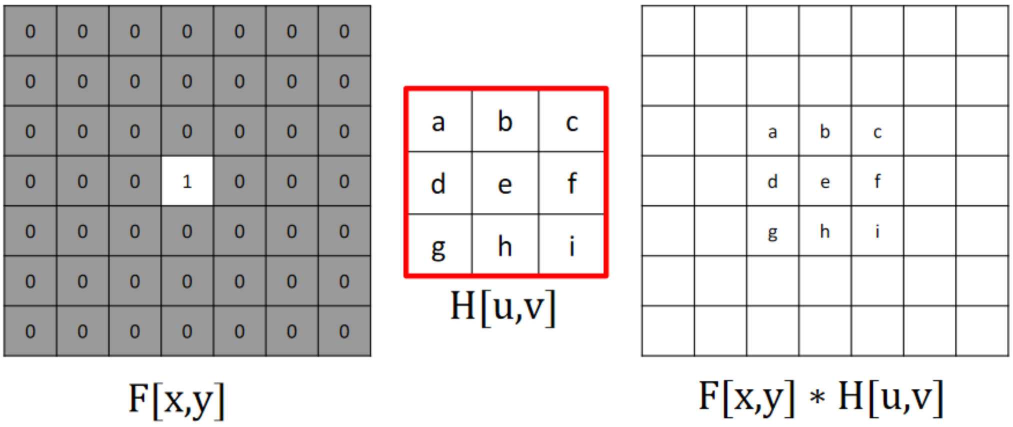


Figure 5: Convolution

Cross-correlation

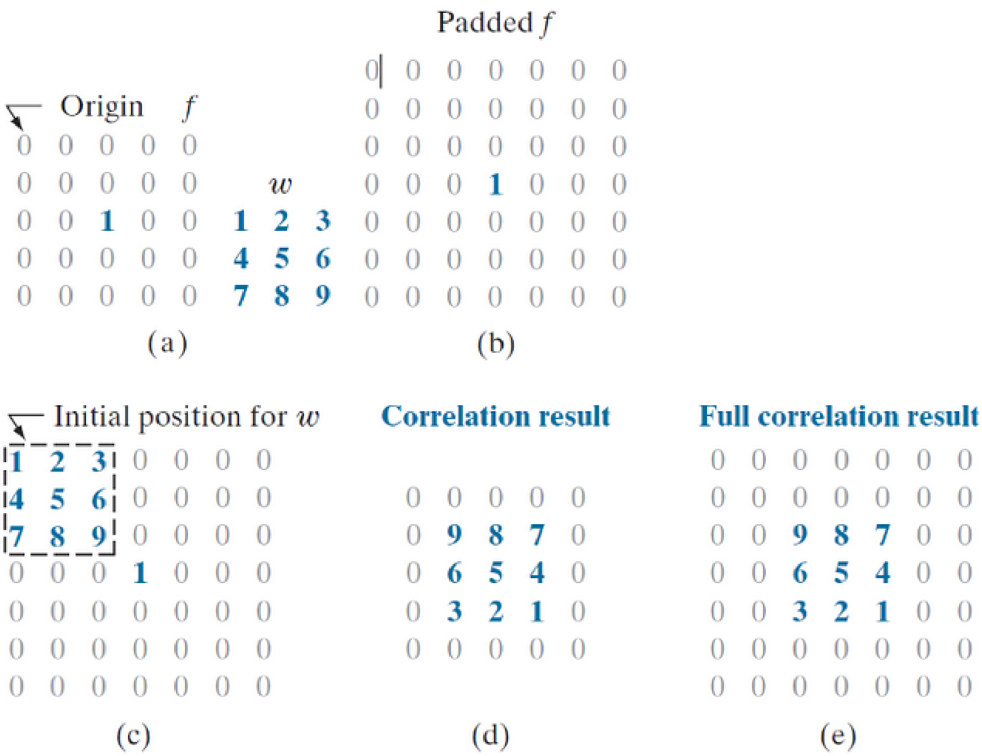


Figure 6: Cross-correlation

Convolution

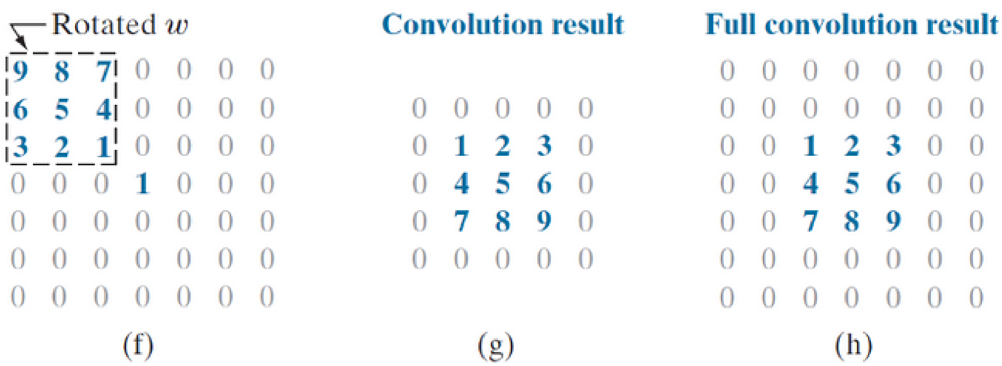


Figure 7: Convolution

3 Common Filters 常见的 Filter，与原图像做卷积

3.1 Original (No effect)

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

3.2 Shift Left

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

3.3 Sharpening

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

3.4 Gaussian Blur 高斯模糊

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

图像的高斯模糊过程就是**图像与正态分布做卷积**。

3.5 Sobel Operator 索伯算子

使用以下两种 Kernel 与图像做卷积可以**探测边界**。

- Detect Horizontal Edges

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Detect Vertical Edges

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

3.6 Some other filters

- Average/blur filters: average pixel values, blur the image
- Sharpening filters: subtract pixel from surround, increase fine detail
- Edge filters: compute difference between pixels, detect oriented edges in image

4 Properties of Linear Filtering

- Commutative: $f * h = h * f$
- Associative: $(f * h1) * h2 = f * (h1 * h2)$
- Distributive over addition: $f * (h1 + h2) = (f * h1) + (f * h2)$
- Multiplication cancels out: $kf * h = f * kh = k(f * h)$

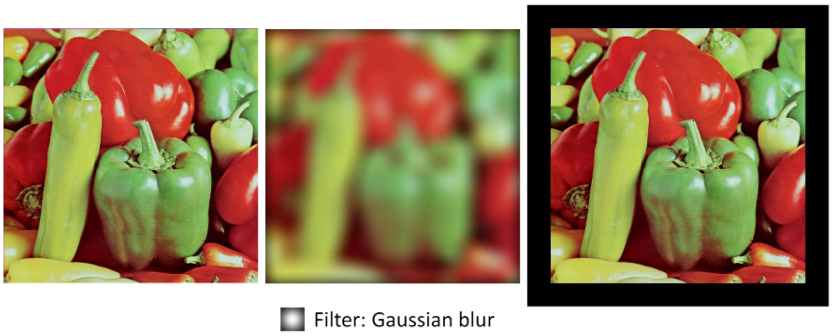
5 Efficient Filtering

通常使用一个复杂的 Filter 的效率要比连续使用简单的 Filter 效率要高。

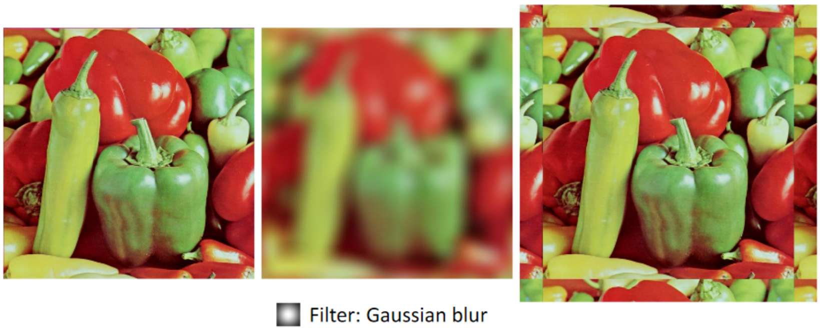
6 Border Handling

因为使用卷积会导致最外一圈的 Pixel 消失，需要一些手段来“恢复”边界。

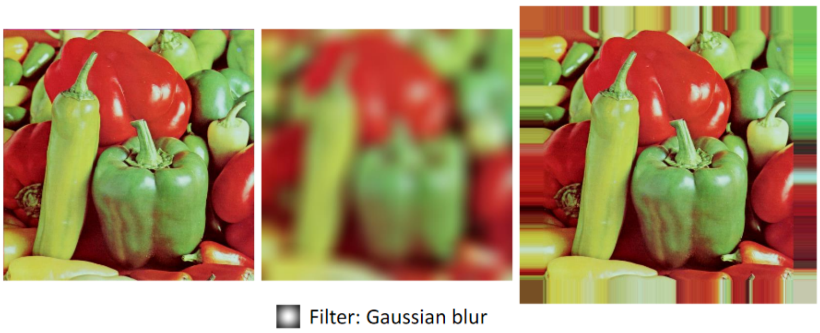
- Pad with constant value



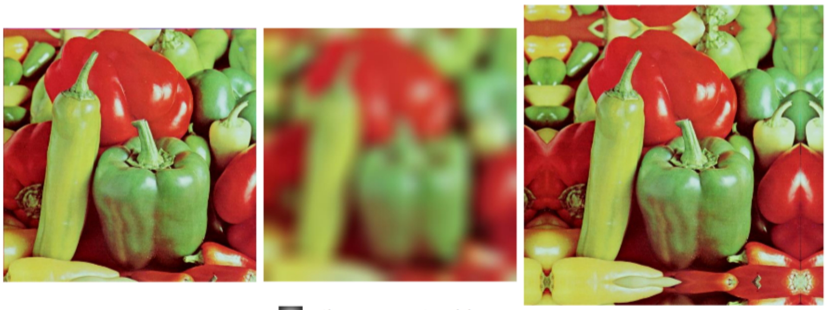
- Wrap image



- Clamp / replicate the border value



- Reflect image



Filter: Gaussian blur