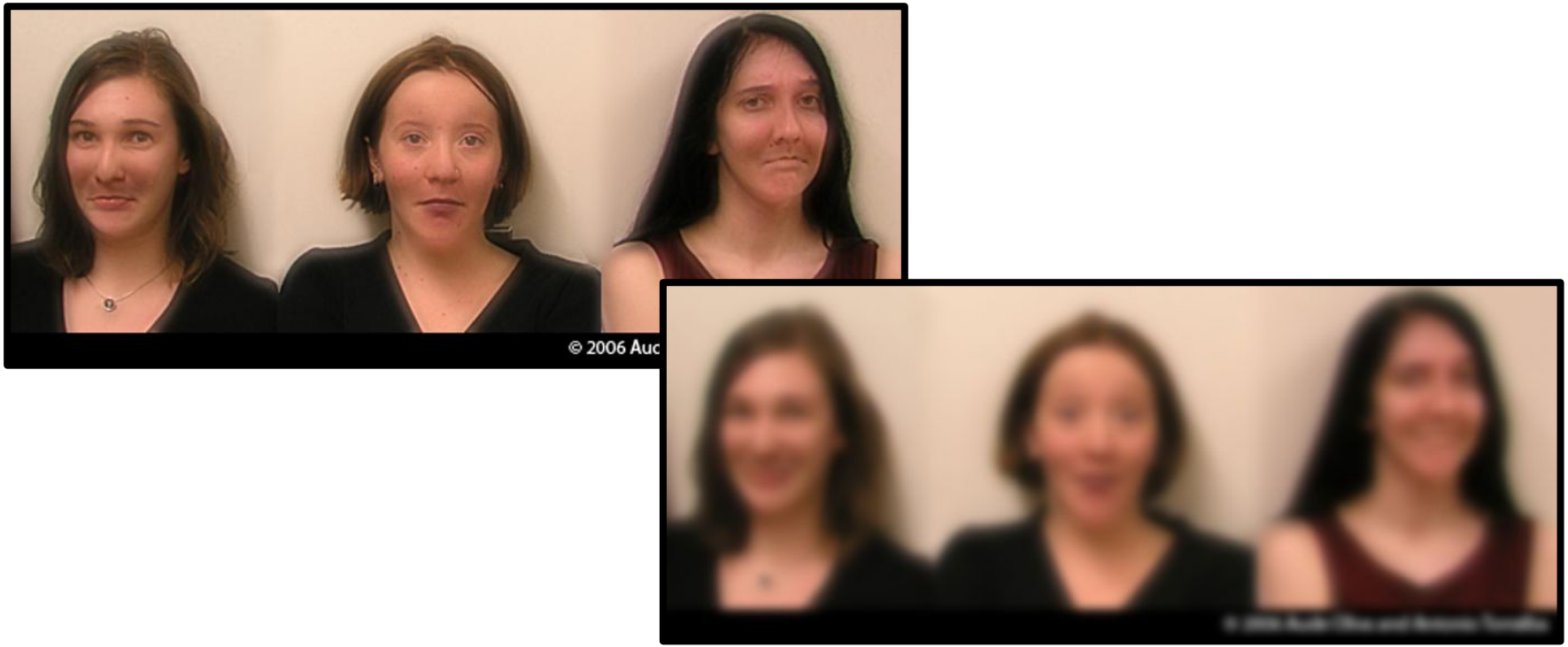
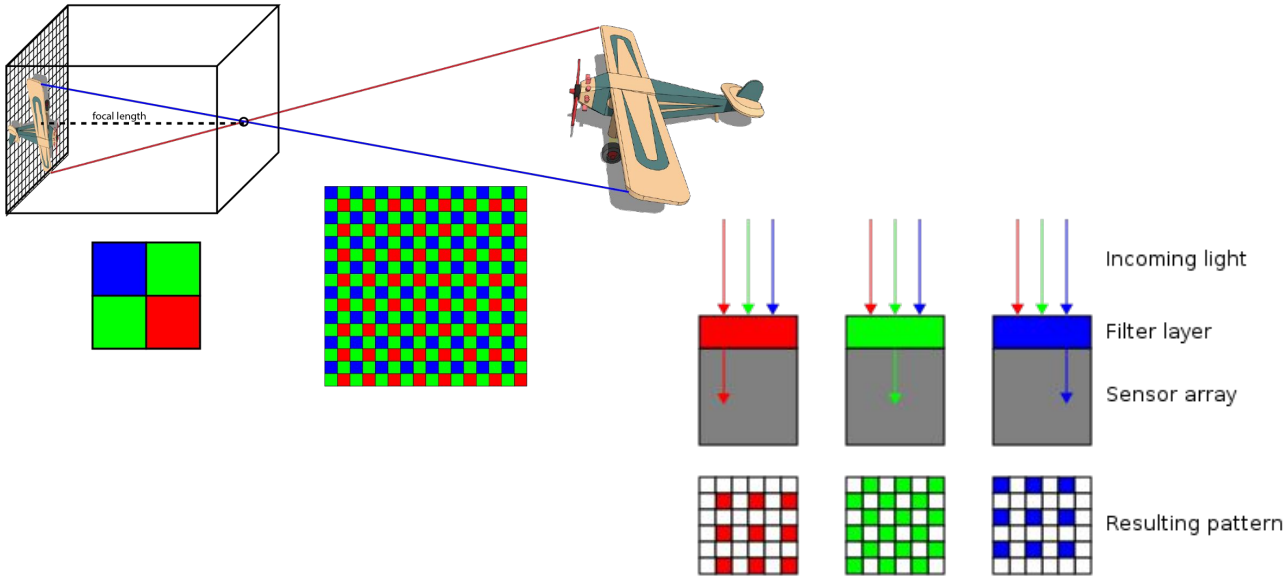
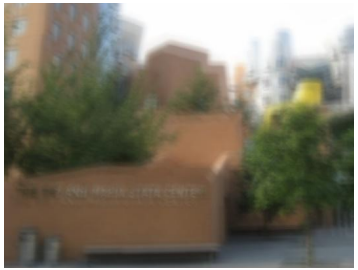
# Filtering




focal length
Incoming light
Filter layer
Sensor array
Resulting pattern

# “Optical” convolution

Camera shake

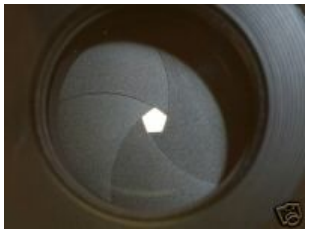


=


*

Source: Fergus, *et al.* “Removing Camera Shake from a Single Photograph”, SIGGRAPH 2006

**Bokeh**: Blur in out-of-focus regions of an image.



Source: https://www.diyphotography.net/diy_create_your_own_bokeh/

# Types of Images

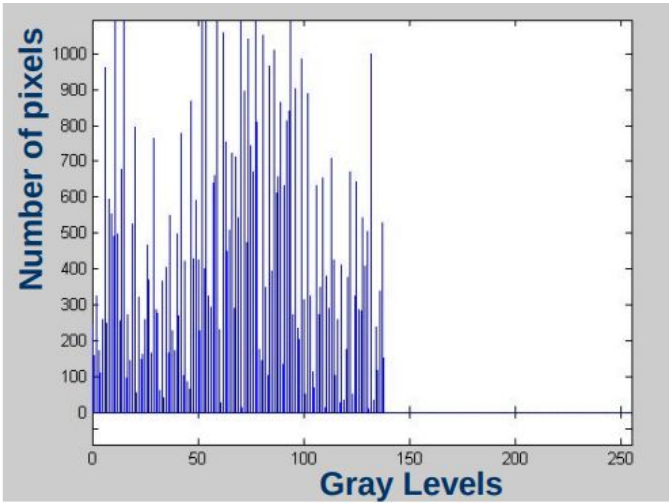
Binary

Gray Scale

Color

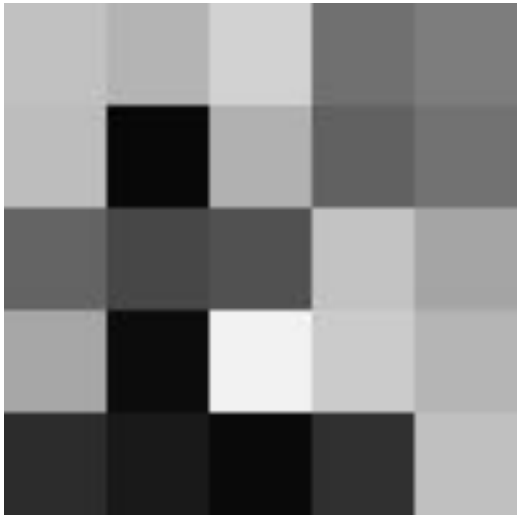
# Histogram

- Histogram captures the distribution of gray levels in the image.
- How frequently each gray level occurs in the image

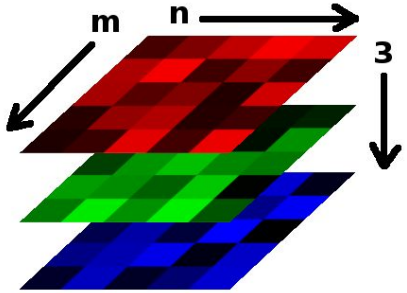


Images are matrix.

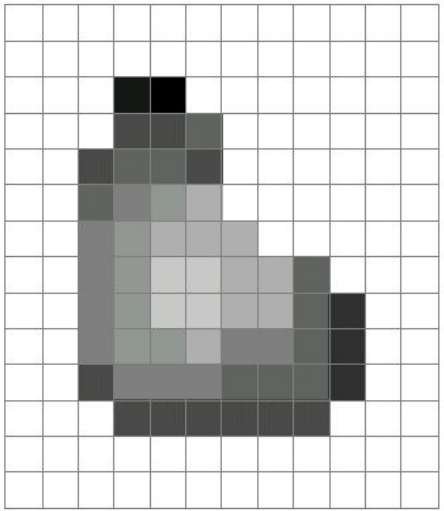
Values in matrix = how much light



$$\begin{bmatrix} 193 & 180 & 210 & 112 & 125 \\ 189 & 8 & 177 & 97 & 114 \\ 100 & 71 & 81 & 195 & 165 \\ 167 & 12 & 242 & 203 & 181 \\ 44 & 25 & 9 & 48 & 192 \end{bmatrix}$$




```
Image = cv2.imread(FileName, Flag)
```



=

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 20 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 75 | 75 | 75 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 75 | 95 | 95 | 75 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 96 | 127 | 145 | 175 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 127 | 145 | 175 | 175 | 175 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 127 | 145 | 200 | 200 | 175 | 175 | 95 | 255 | 255 | 255 |
| 255 | 255 | 127 | 145 | 200 | 200 | 175 | 175 | 95 | 47 | 255 | 255 |
| 255 | 255 | 127 | 145 | 145 | 175 | 127 | 127 | 95 | 47 | 255 | 255 |
| 255 | 255 | 74 | 127 | 127 | 127 | 95 | 95 | 95 | 47 | 255 | 255 |
| 255 | 255 | 255 | 74 | 74 | 74 | 74 | 74 | 74 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |

# Image as Discrete Functions

- Images are usually **digital** (**discrete**):
  - **Sample** the 2D space on a regular grid
- Represented as a matrix of integer values

Cartesian coordinates

$$f[n,m] = \begin{bmatrix} \ddots & & \vdots & & \\ & f[-1,-1] & f[0,-1] & f[1,-1] & \\ \cdots & f[-1,0] & f[0,0] & f[1,0] & \cdots \\ & f[-1,1] & f[0,1] & f[1,1] & \\ & & \vdots & & \ddots \end{bmatrix}$$

Notation for discrete functions

$j$

$i$

pixel

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 62 | 79 | 23 | 119 | 120 | 105 | 4 | 0 |
| 10 | 10 | 9 | 62 | 12 | 78 | 34 | 0 |
| 10 | 58 | 197 | 46 | 46 | 0 | 0 | 48 |
| 176 | 135 | 5 | 188 | 191 | 68 | 0 | 49 |
| 2 | 1 | 1 | 29 | 26 | 37 | 0 | 77 |
| 0 | 89 | 144 | 147 | 187 | 102 | 62 | 208 |
| 255 | 252 | 0 | 166 | 123 | 62 | 0 | 31 |
| 166 | 63 | 127 | 17 | 1 | 0 | 99 | 30 |

# Cont'd

**An Image** as a function $f$ from $R^2$ to $R^M$:

$f(x,y)$ gives the **intensity** at position $(x,y)$

Defined over a rectangle, with a finite range:

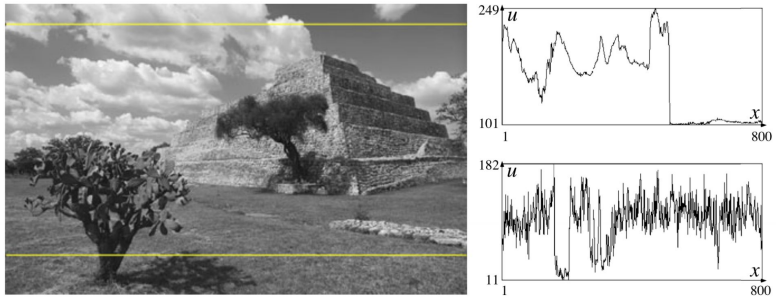
$f: [a,b] \times [c,d] \rightarrow [0,255]$

Domain support

range

Color images

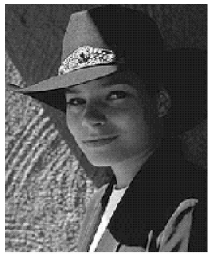
$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$




Histograms are a type of image function

# 2D discrete-space systems (filters)

Salt and pepper noise



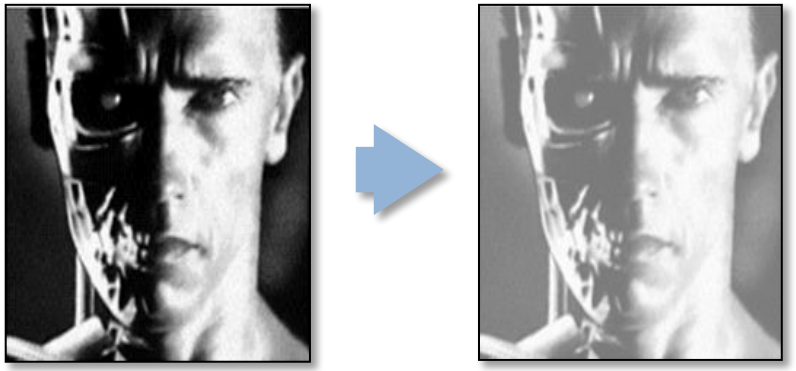
$$f[n,m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n,m]$$

$$g = \mathcal{S}[f], \quad g[n,m] = \mathcal{S}\{f[n,m]\}$$

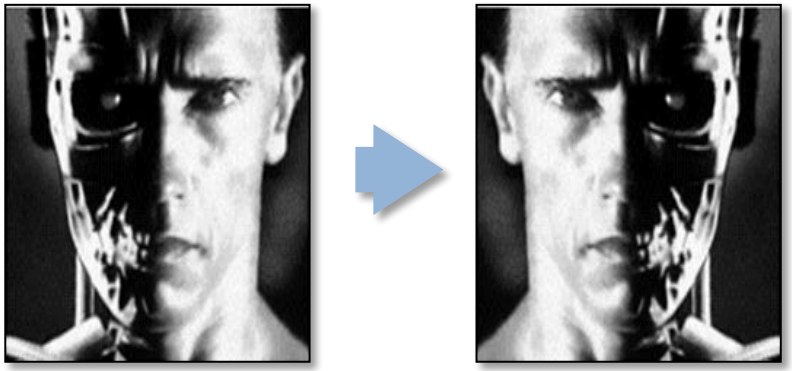
$$f[n,m] \xrightarrow{\mathcal{S}} g[n,m]$$

S : system operator

# 2D discrete-space systems (filters)



$$g(x,y) = f(x,y) + 20$$



$$g(x,y) = f(-x,y)$$

# Canonical Image Processing problems

- Image Restoration
  - **denoising**
  - deblurring
- Image Compression
  - JPEG, JPEG2000, MPEG..
- Computing Field Properties
  - optical flow
  - disparity
- Locating Structural Features
  - corners
  - edges

# Systems and Filters

**Filtering:**

- Forming a new image whose pixel values are transformed from original pixel values

**Goals:**

- Goal is to extract **useful information** from images, or transform images **into another domain** where we can modify/enhance image properties
  - Enhance images

    Denoise, resize, increase contrast, etc.
  - Extract information from images

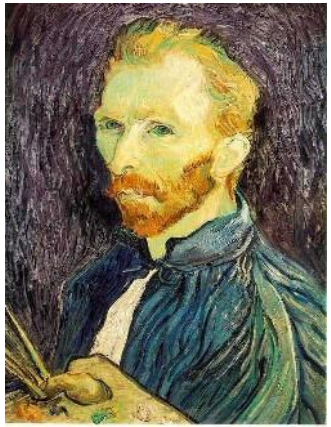
    Texture, edges, distinctive points, etc.
  - Detect patterns

    Template matching

## De-noising

Salt and pepper noise

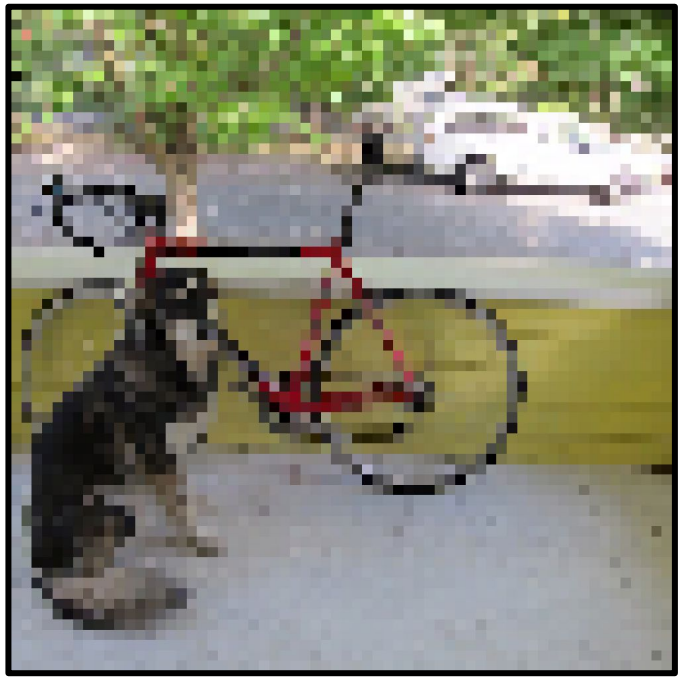
## Super-resolution



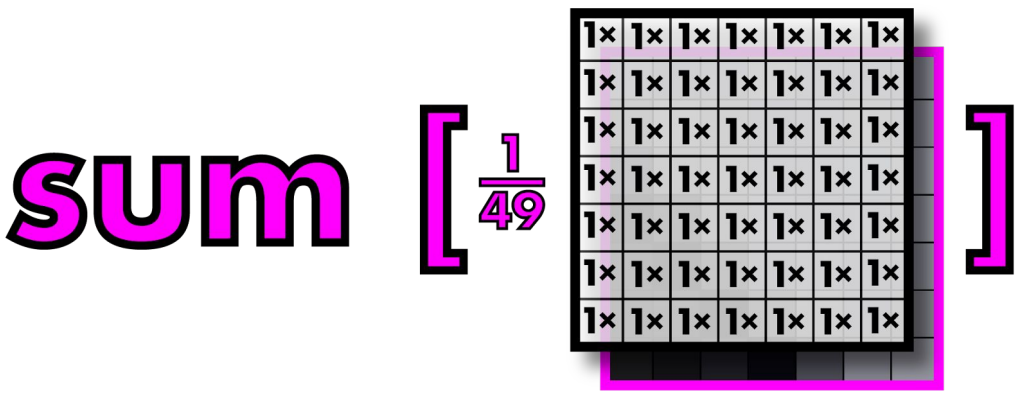
## In-painting


Bertamio et al

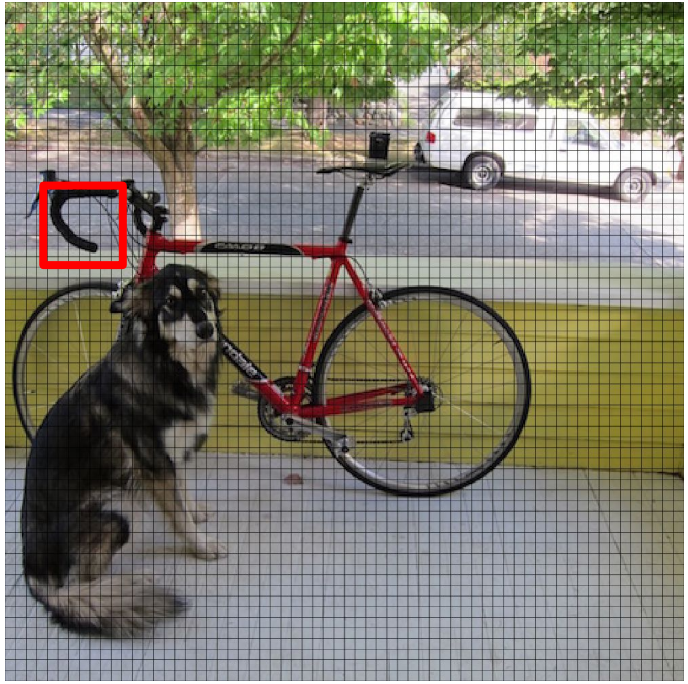
# Filter example : Moving Average

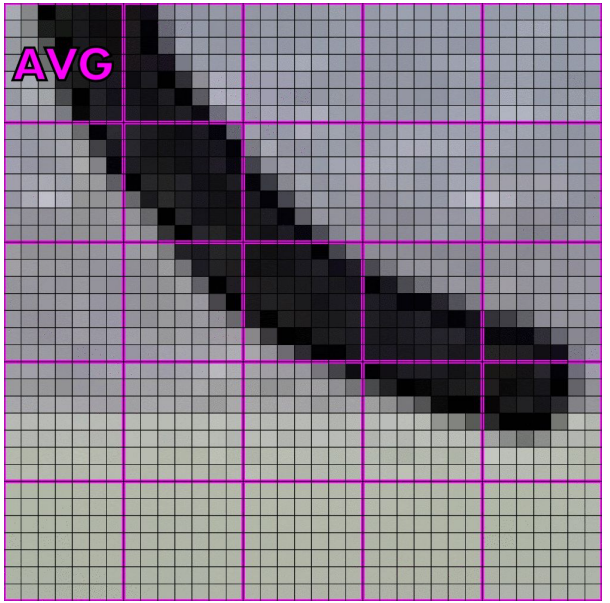


# Cont’d : Moving Average

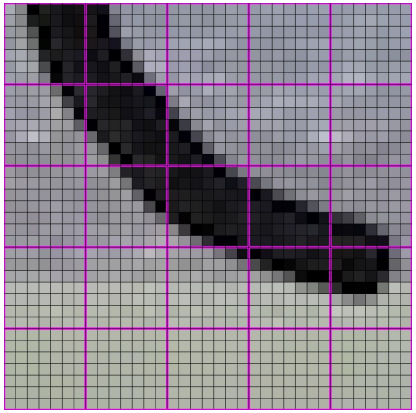


# How do? Averaging!

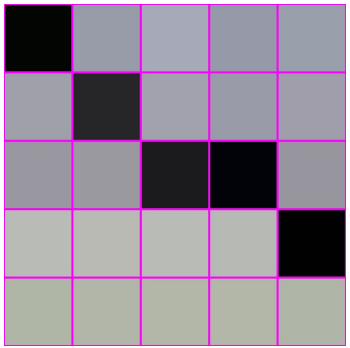


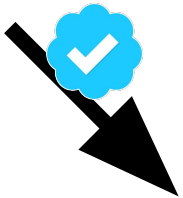


# How do? Averaging!

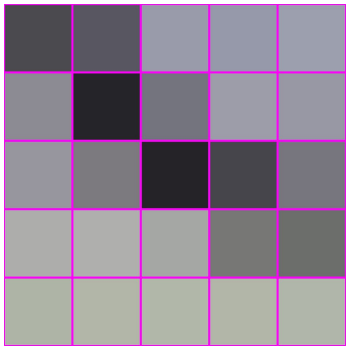


Resize :
“interpolation”

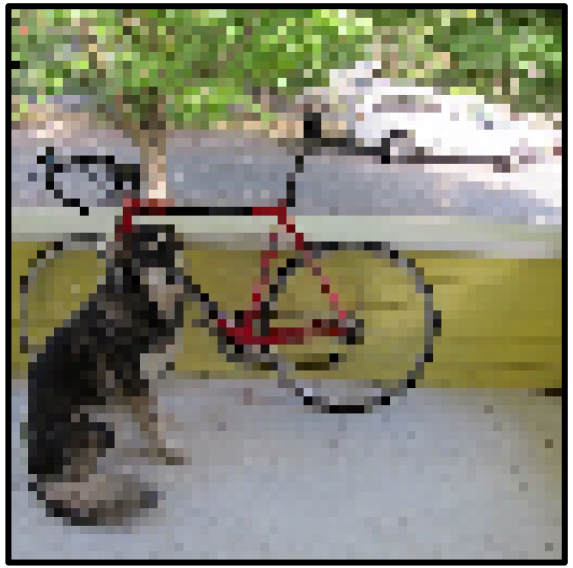




averaging



# Before and After !



# LOOK AT HOW MUCH BETTER

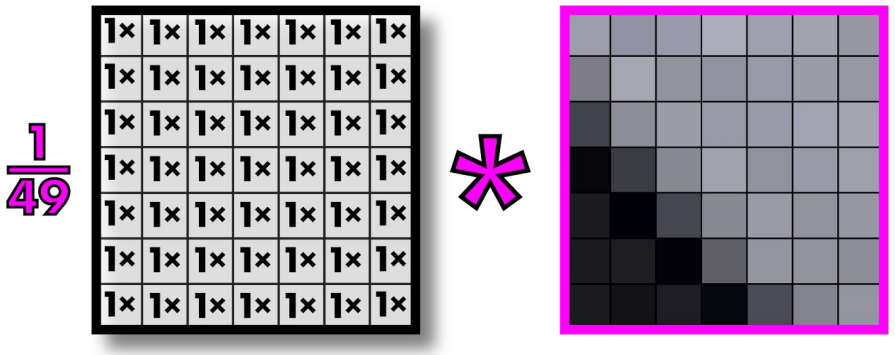
# Cont'd

In summary:

- This filter creates a new pixel with an average of its neighborhood.

- Achieve smoothing effect (remove sharp features)

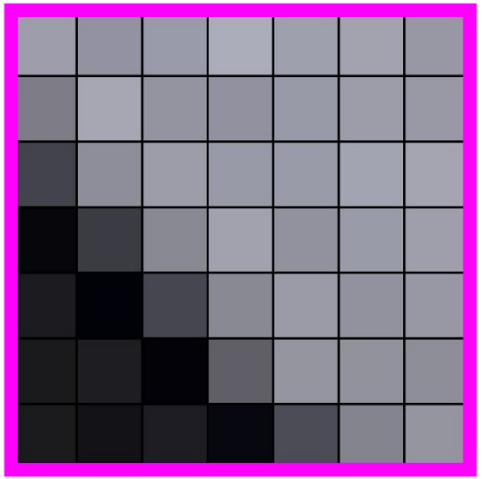
$h[\cdot,\cdot]$ *Filter* or *kernel*




$$f * h = \sum_k \sum_l f(k,l)h(-k,-l)$$

# Call this operation “*convolution*”

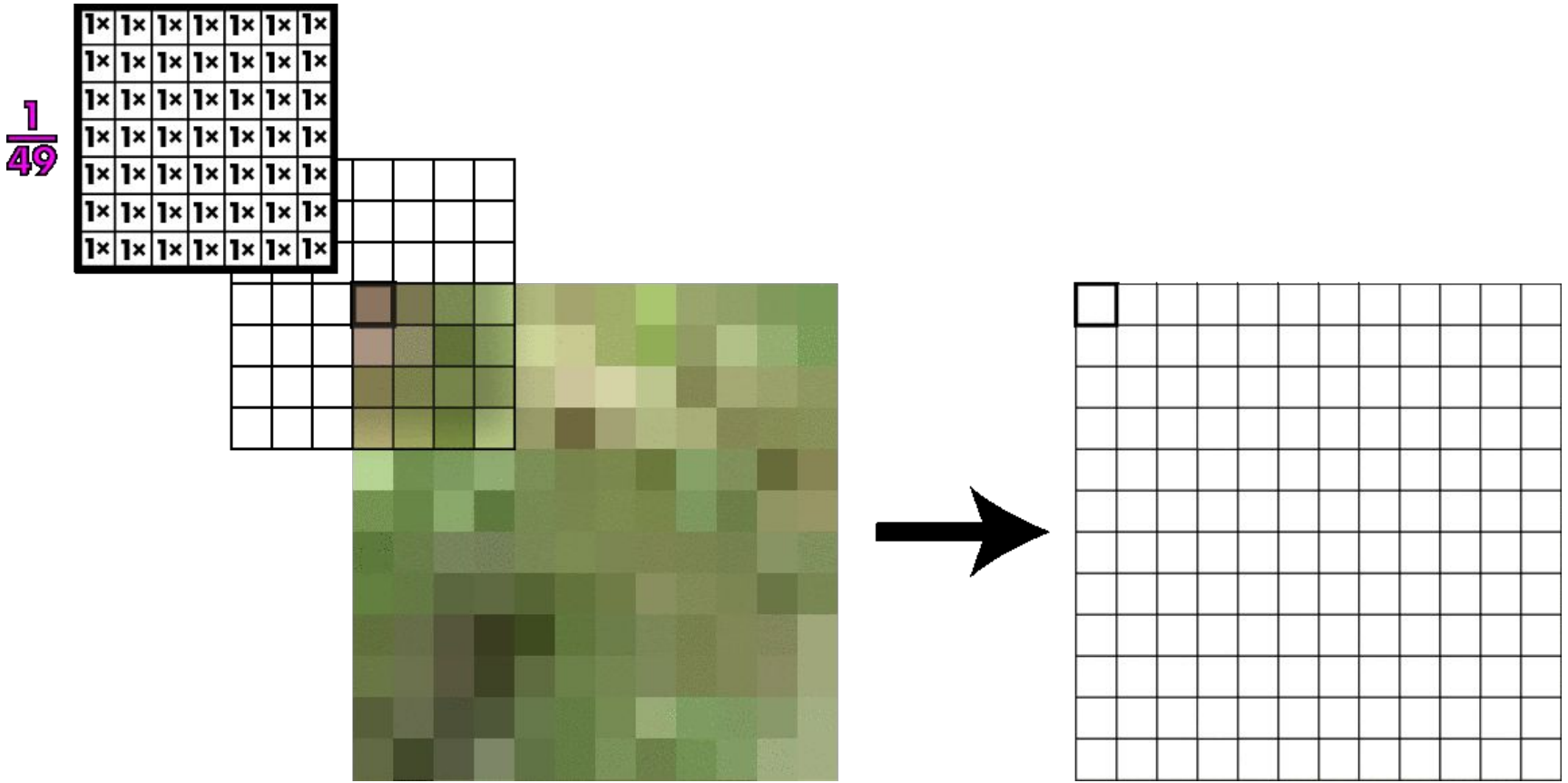
*Filter* or *kernel*

$$\frac{1}{49} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \quad \text{  }$$

# Convolutions on larger images

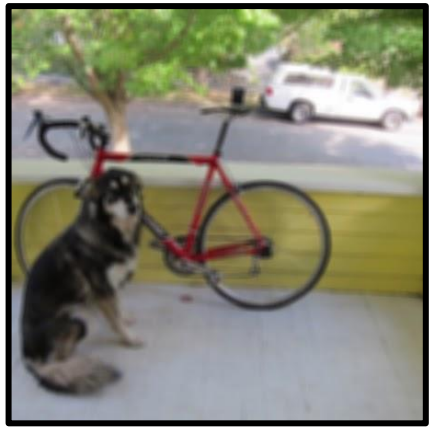
$$\frac{1}{49} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \quad 

# Kernel slides across image

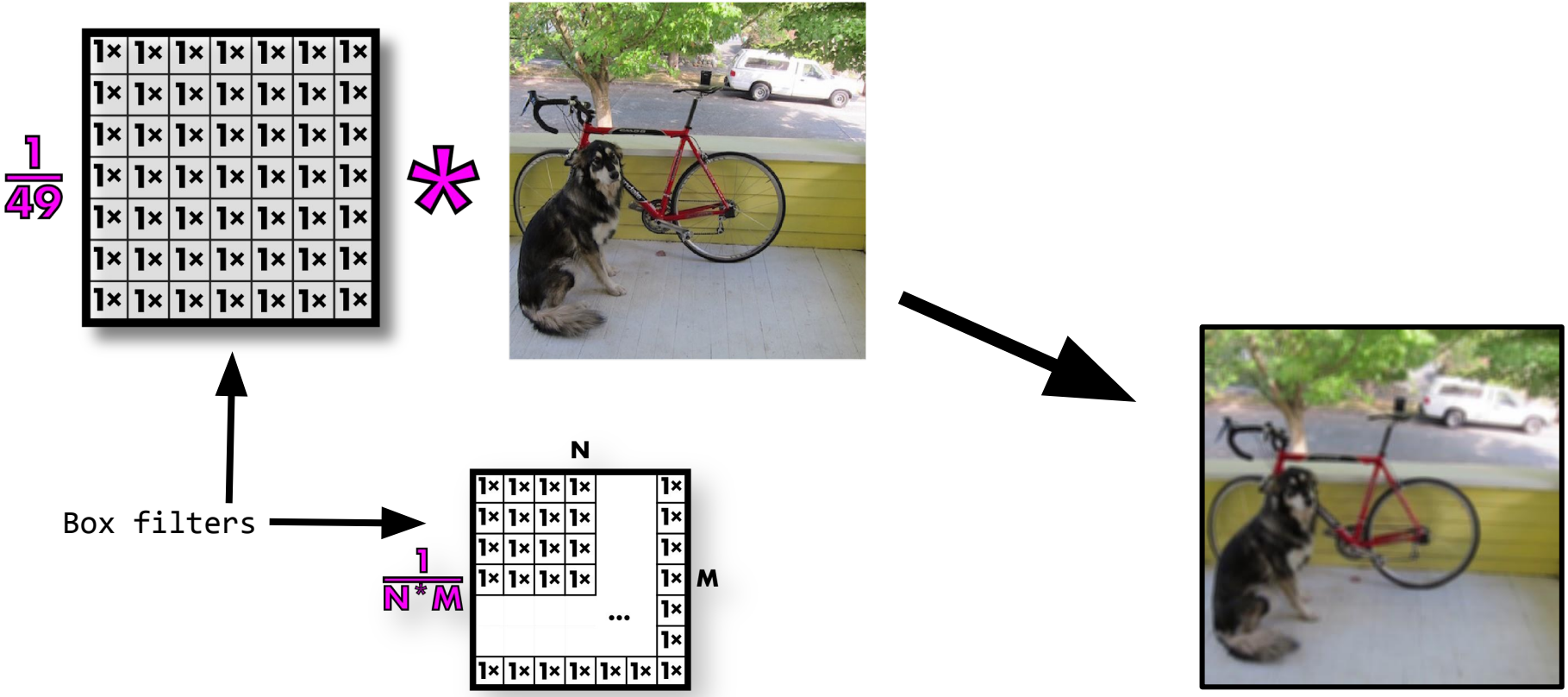


# Convolutions on larger images

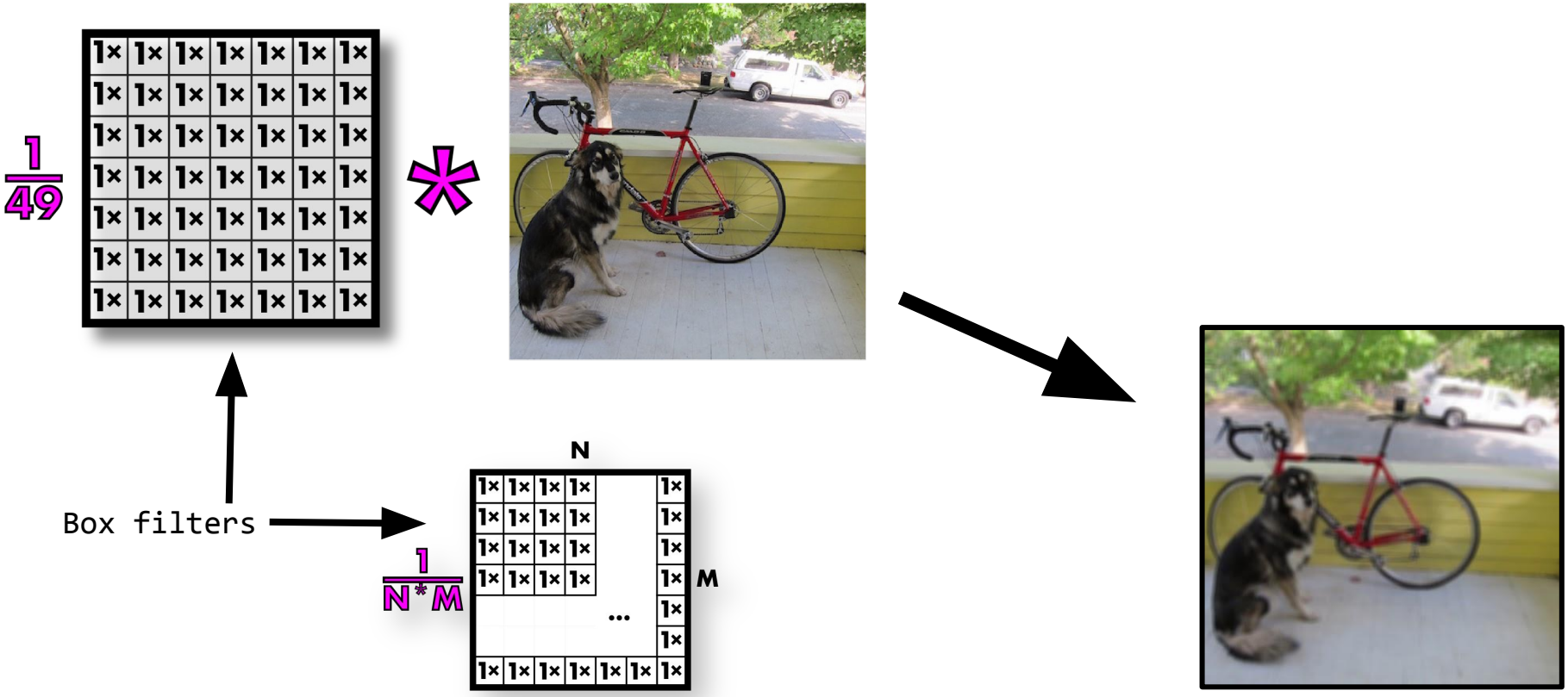
$$\frac{1}{49}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \quad \rightarrow$$



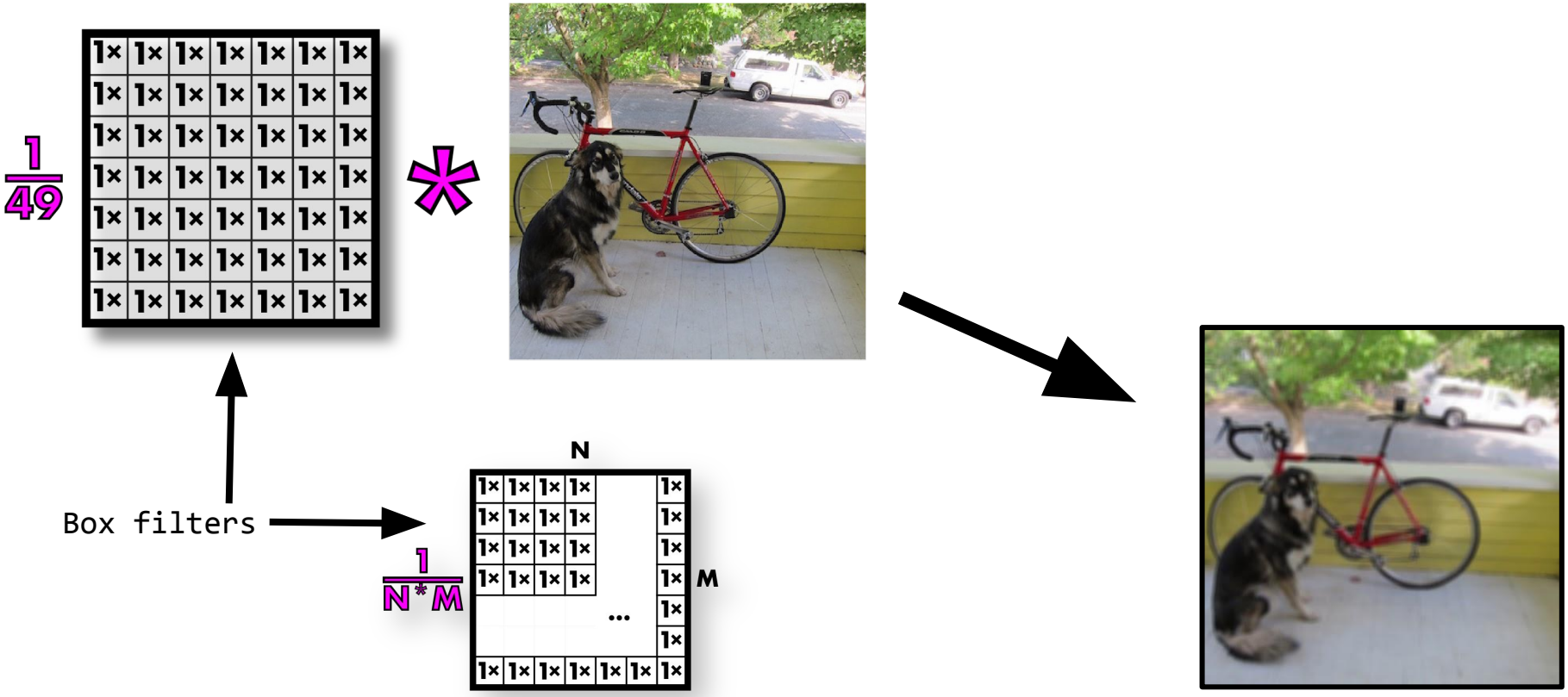
# This is called box filter



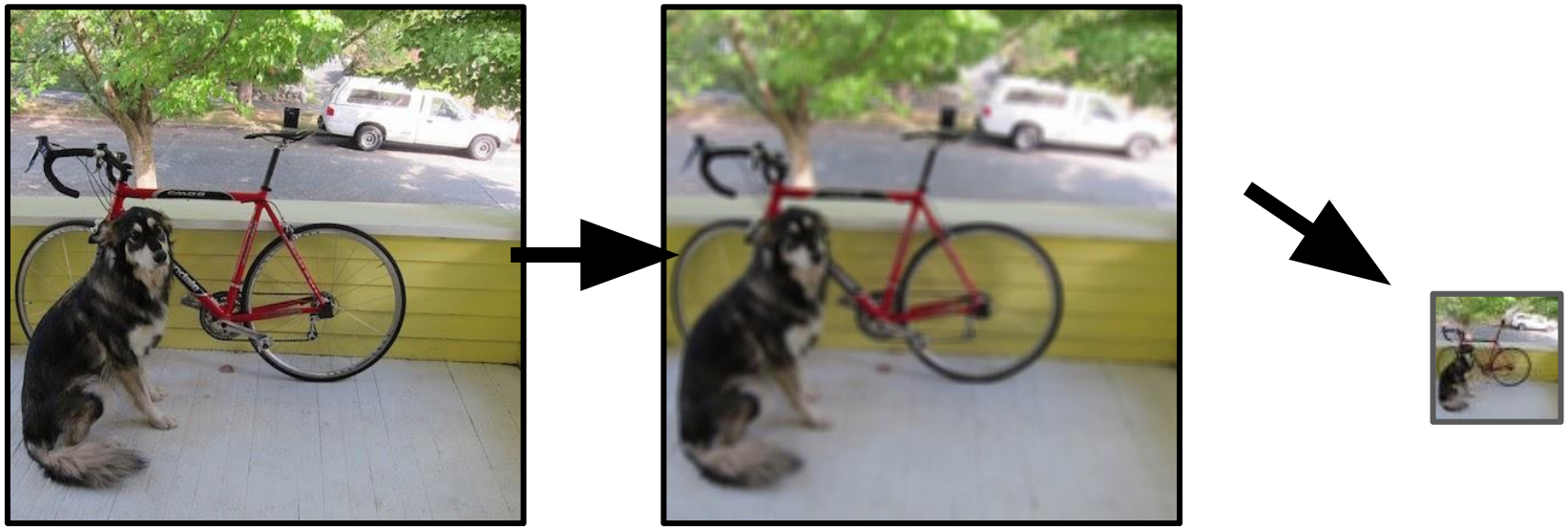
# Box filters smooth image



# Box filters smooth image



# Now we resize our smoothed image

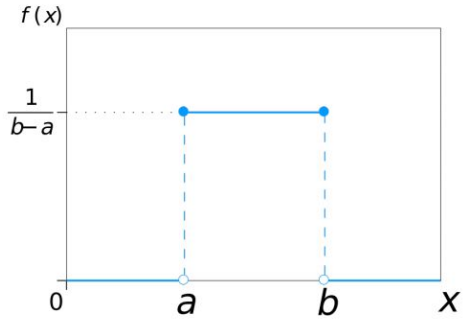


# Image noise

- Light Variations
- Camera Electronics
- Surface Reflectance
- Lens
- Noise is random,
  - with some probability
- It has a distribution

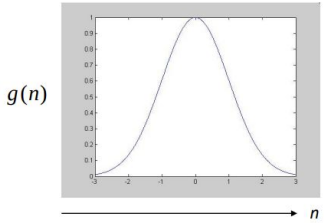
- I(x,y) – true pixel value at (x,y)
- n(x,y) - noise at (x,y)
- I^(x,y) = I(x,y) + n(x,y) additive noise
- I^(x,y) = I(x,y) * n(x,y) multiplicative noise

# Salt and Pepper Noise



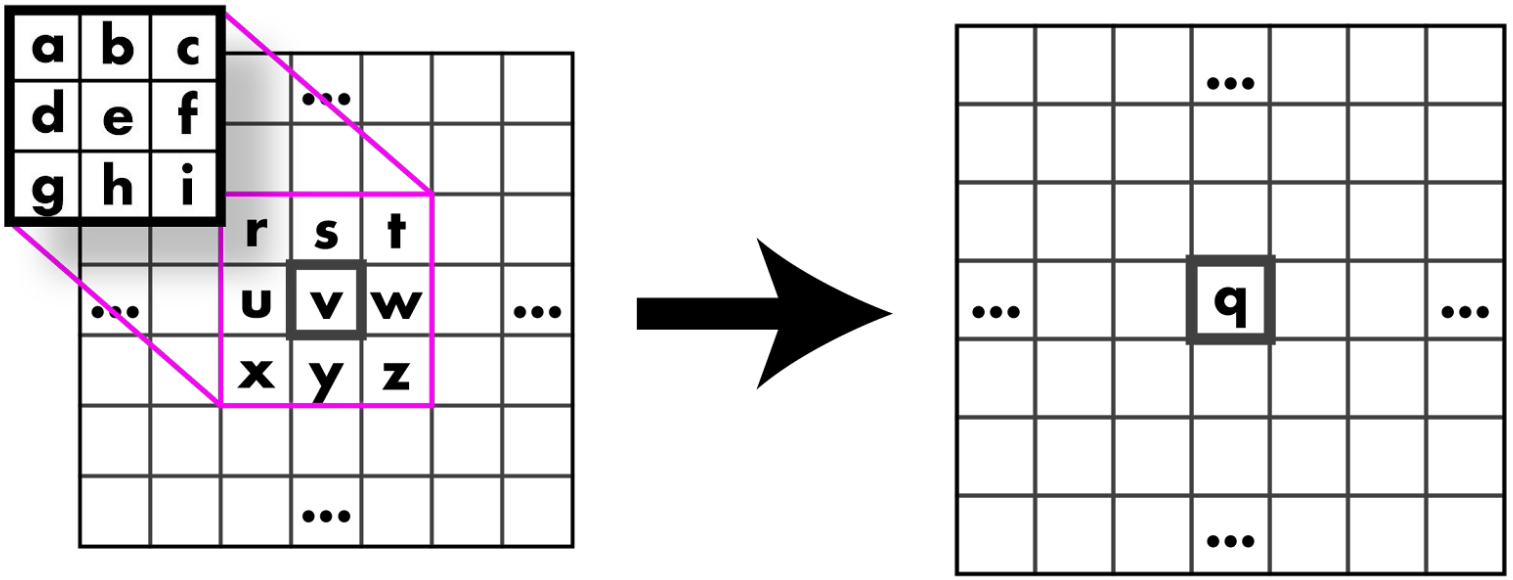

# Gaussian Noise

$$n(x,y) \approx g(n) = e^{\frac{-n^2}{2\sigma^2}}$$



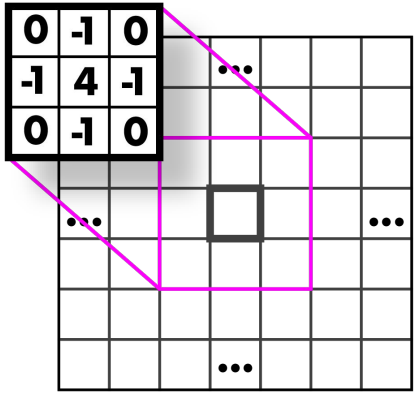

Probability Distribution
*n* is a random variable

# How convolution works ?

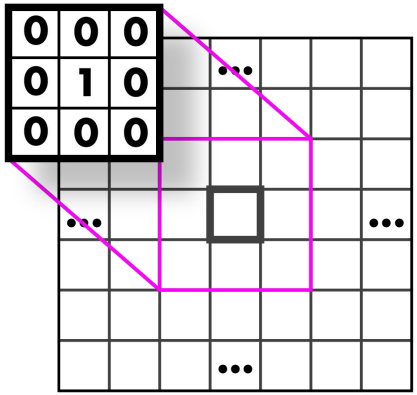



$$q = a \times r + b \times s + c \times t + d \times u + e \times v + f \times w + g \times x + h \times y + i \times z$$

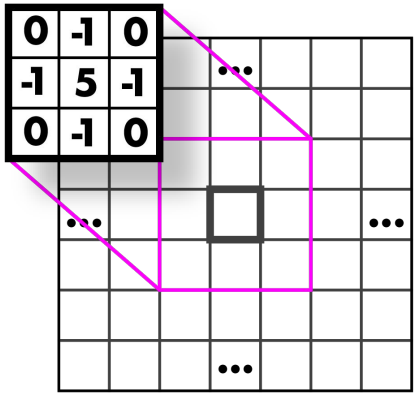
# High-pass Kernel: finds edges




# Identity Kernel: Does nothing!

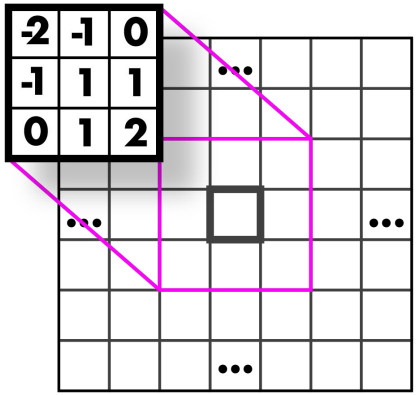



# Sharpen Kernel: sharpens!




Note: sharpen = highpass + identity!

# Emboss Kernel: stylin’




# Q & A

Thank You !