
Want to make image smaller



448x448 -> 64x64



448x448 -> 64x64



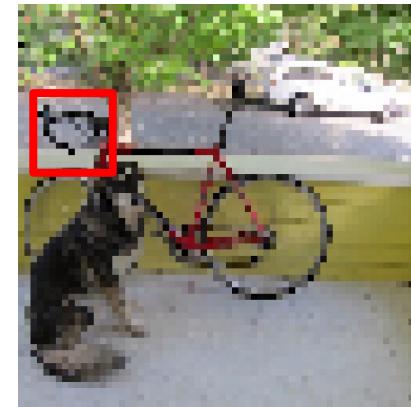
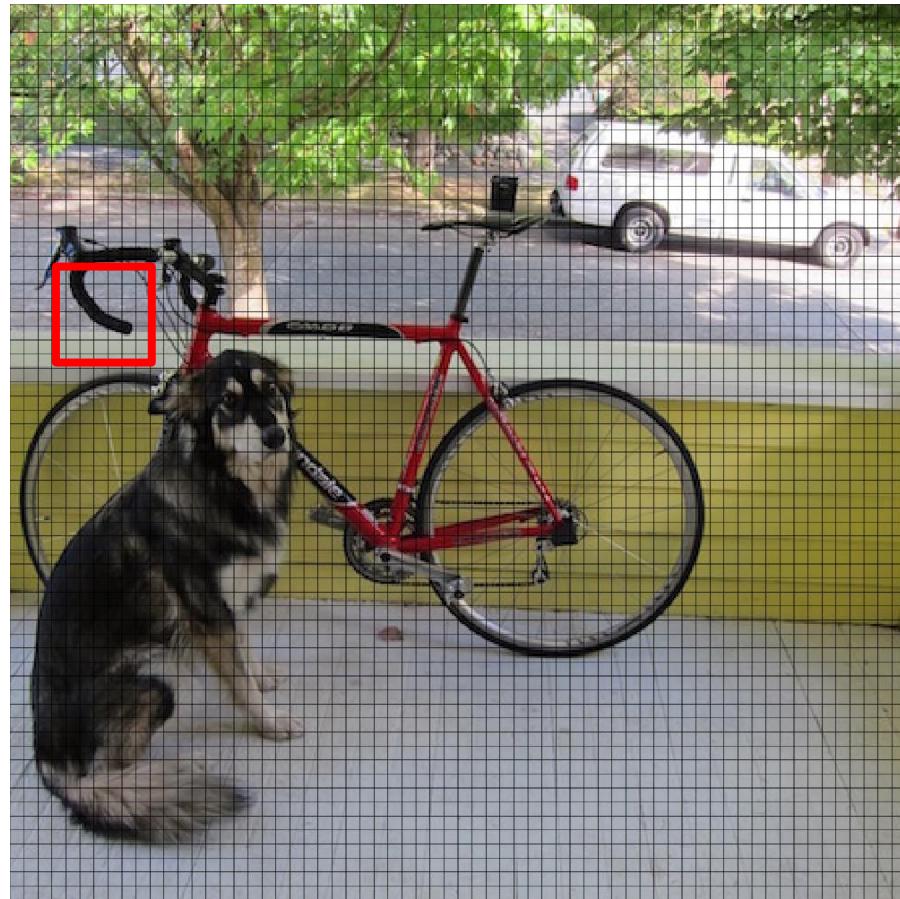
448x448 -> 64x64



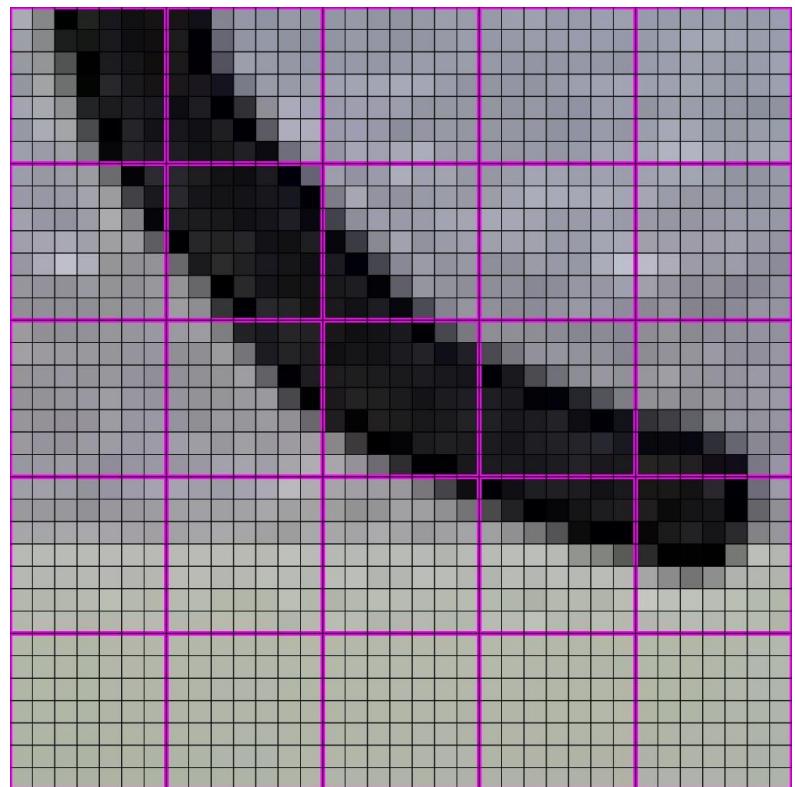
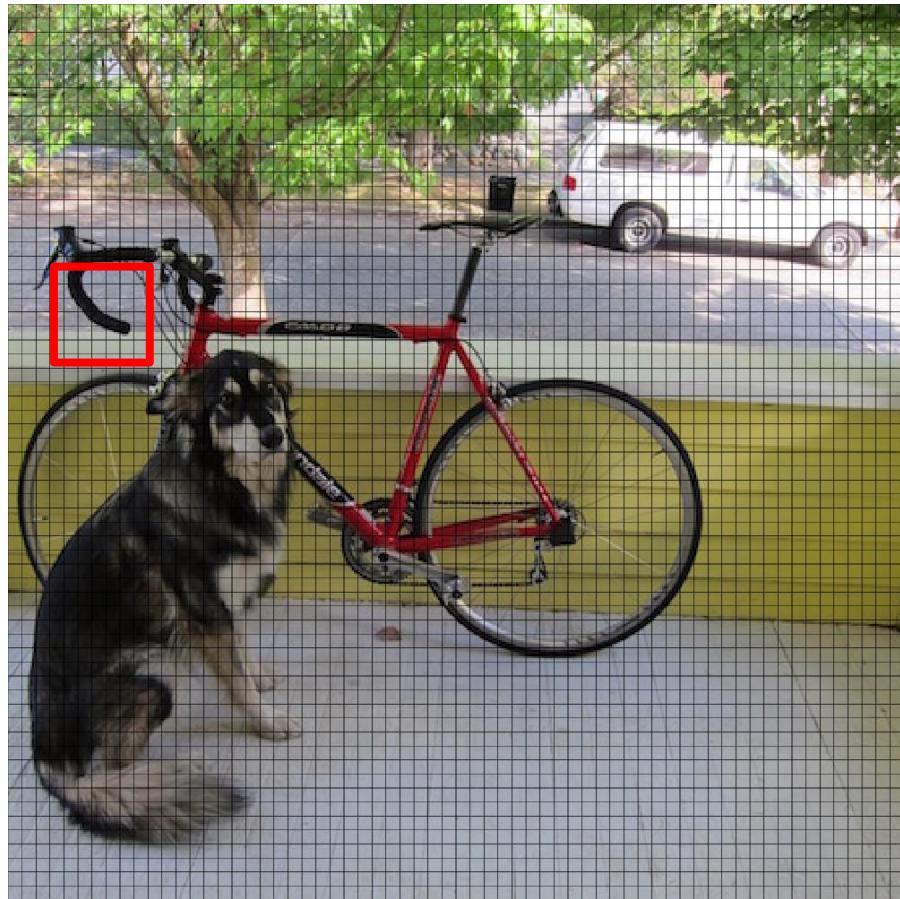
448x448 -> 64x64



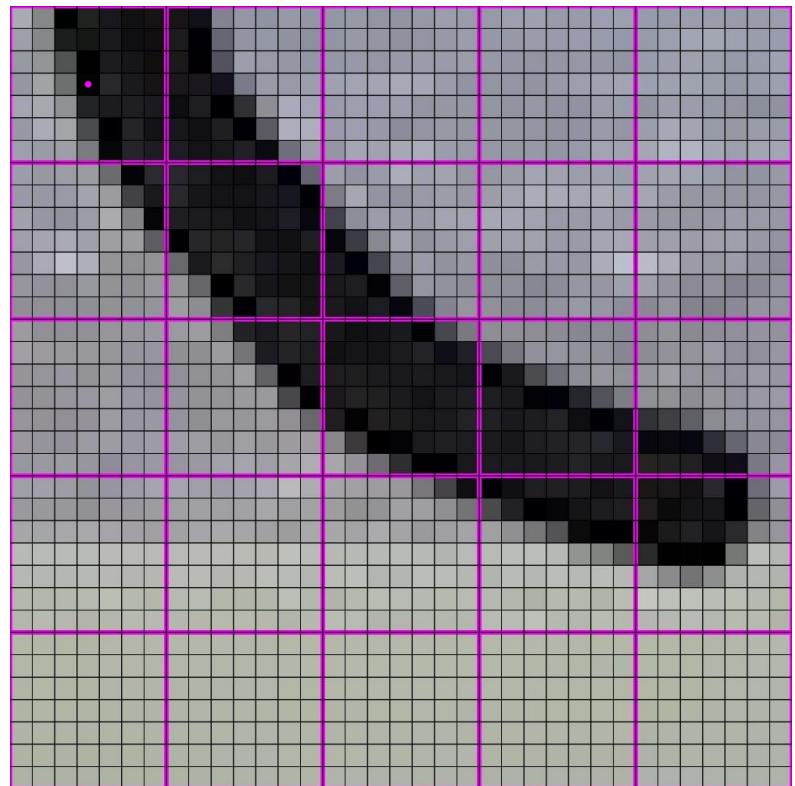
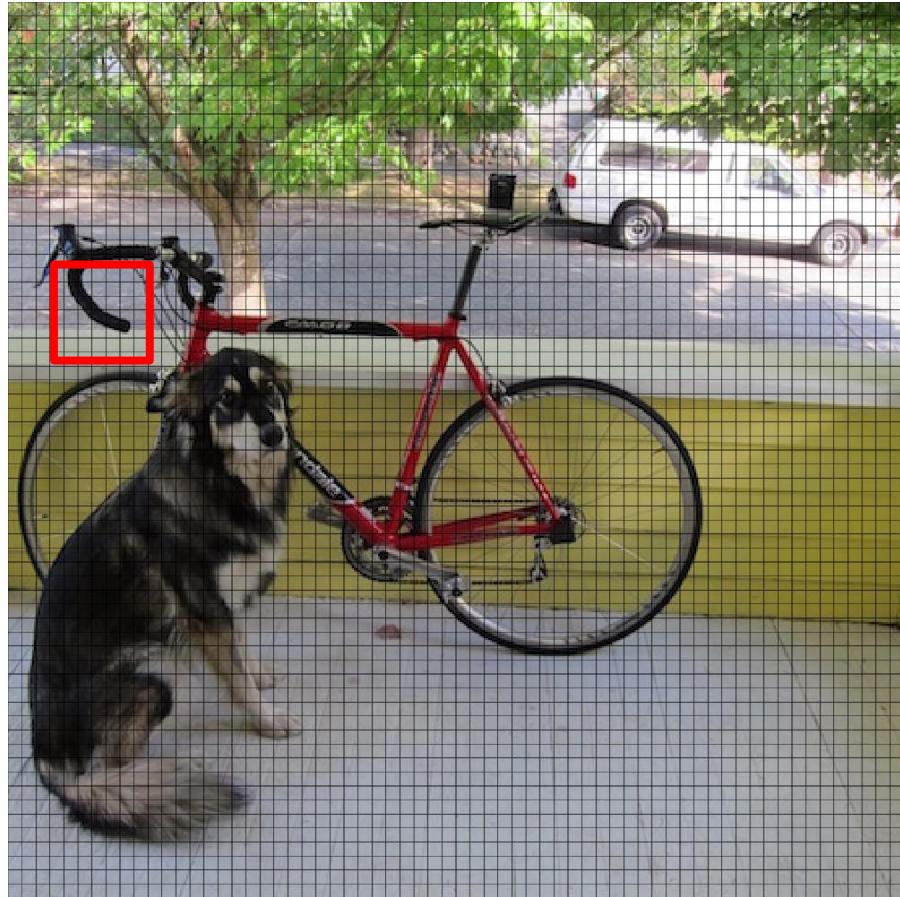
448x448 -> 64x64



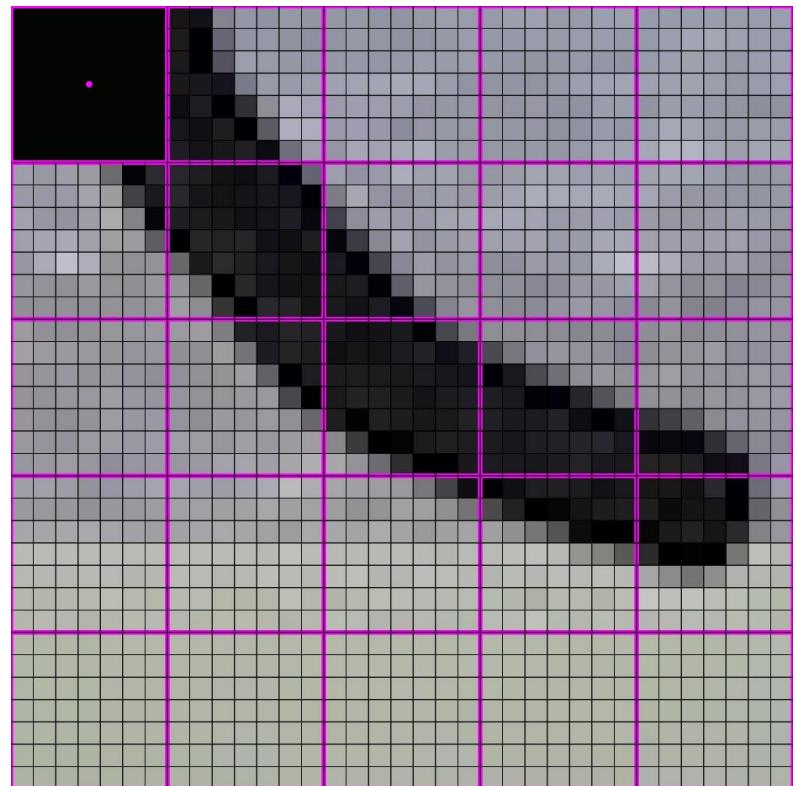
448x448 -> 64x64



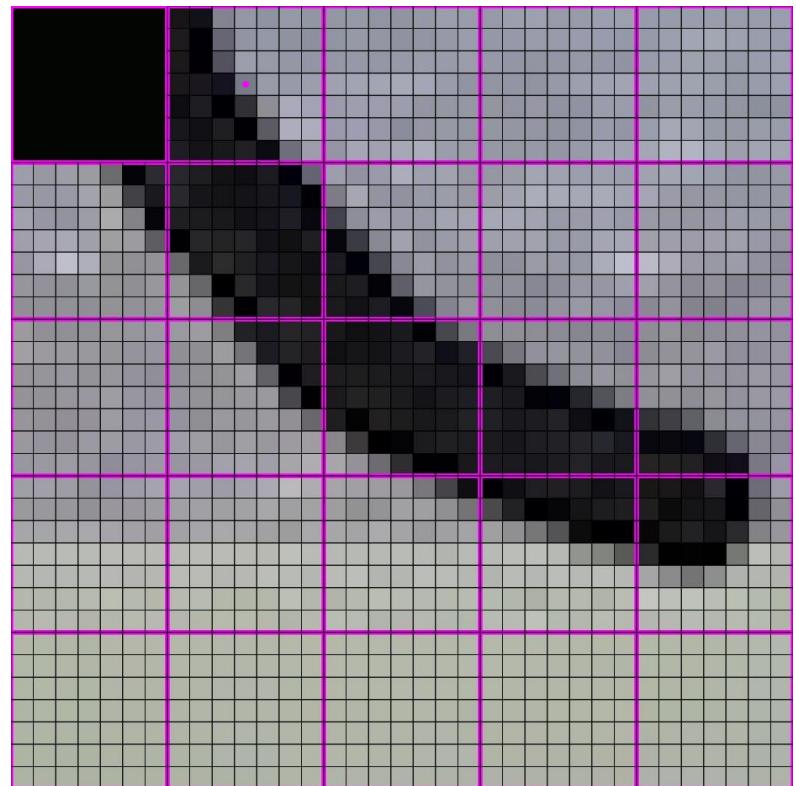
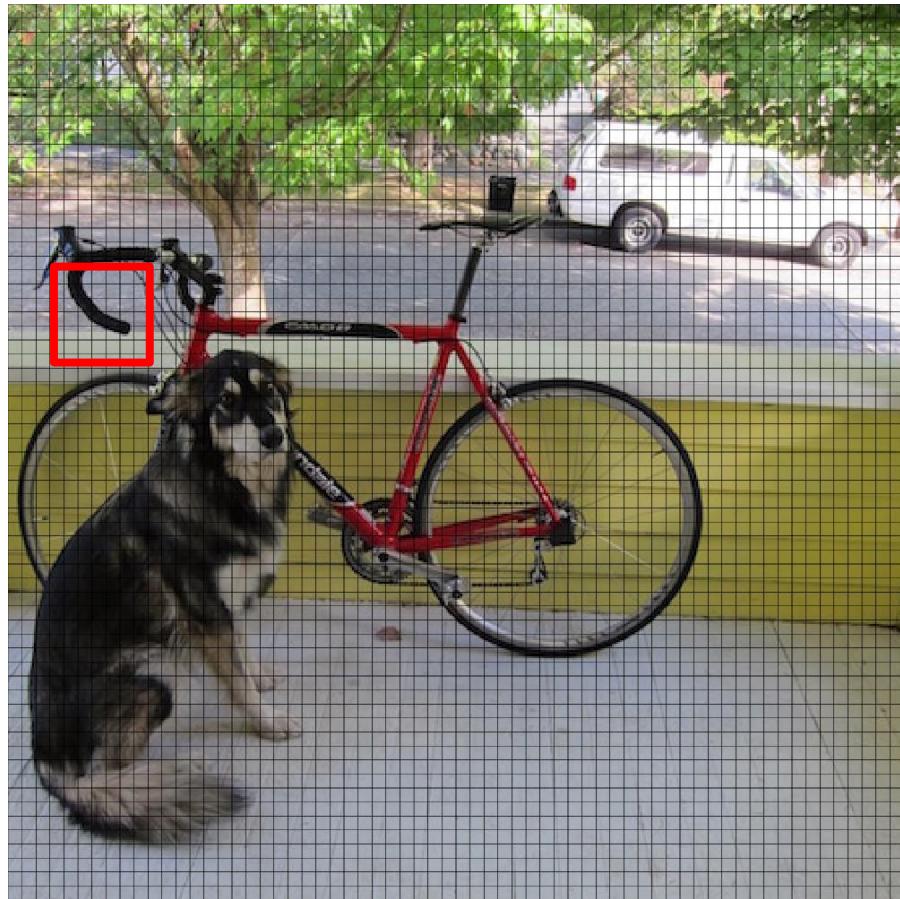
448x448 -> 64x64



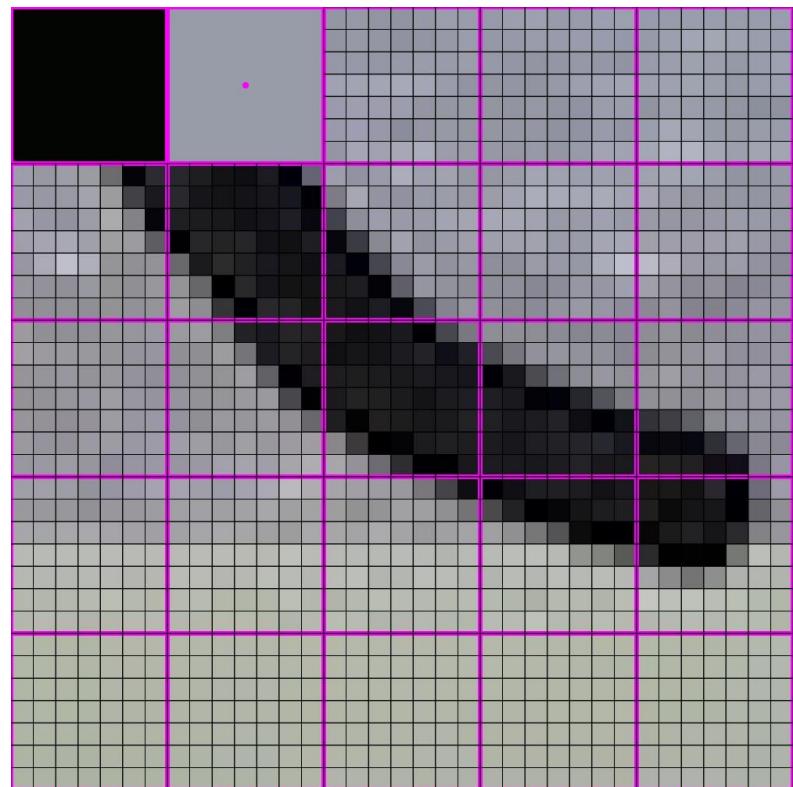
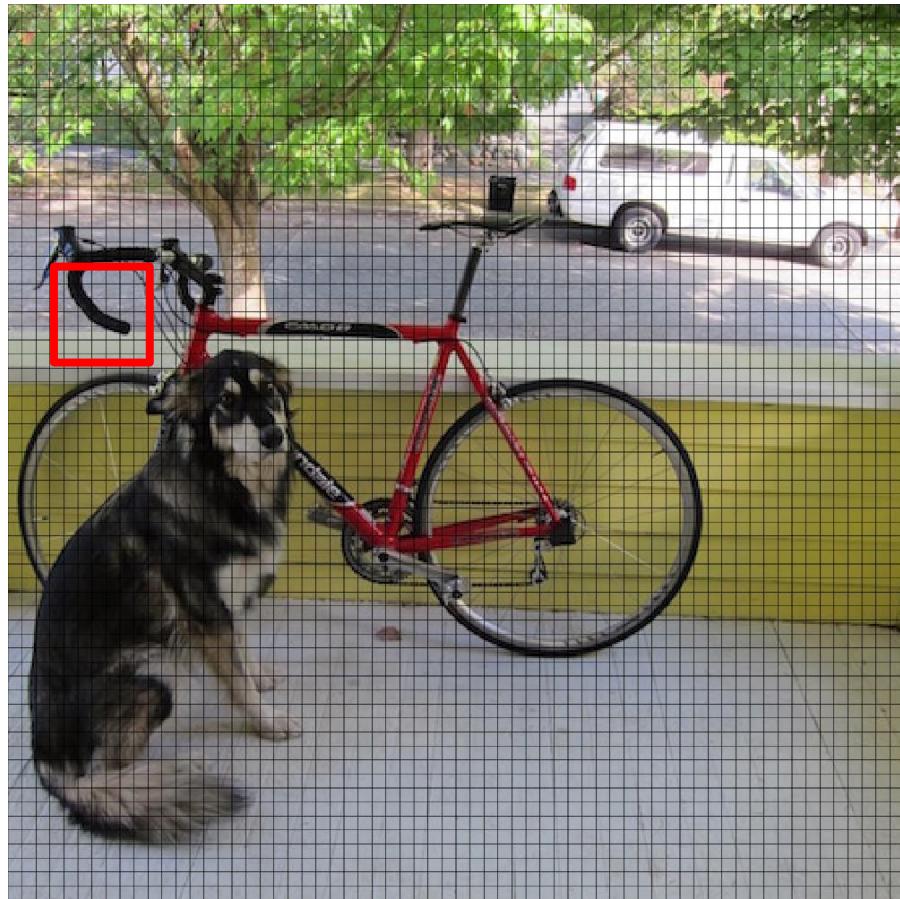
448x448 -> 64x64



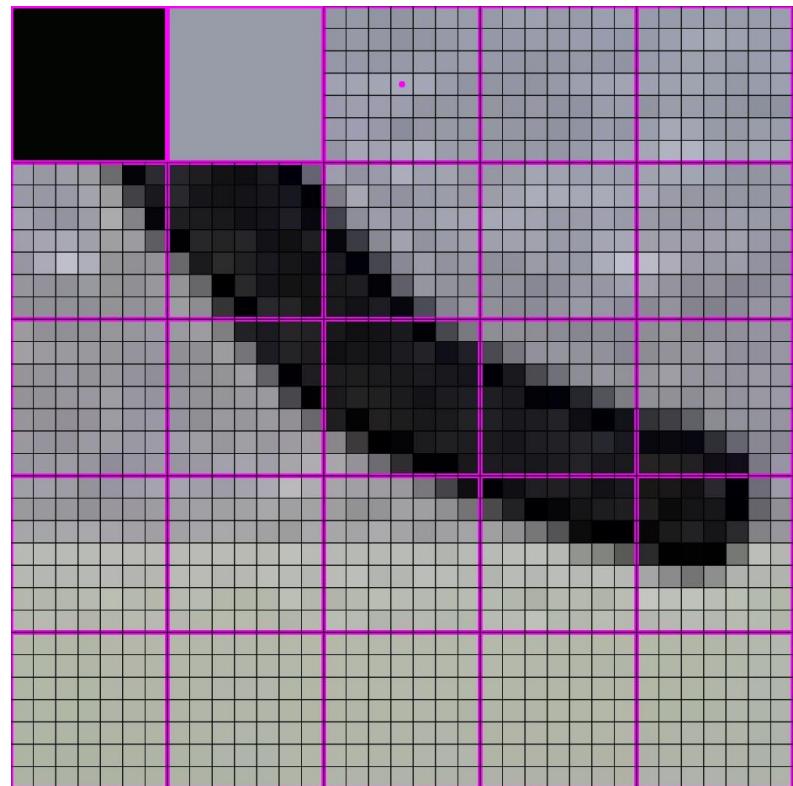
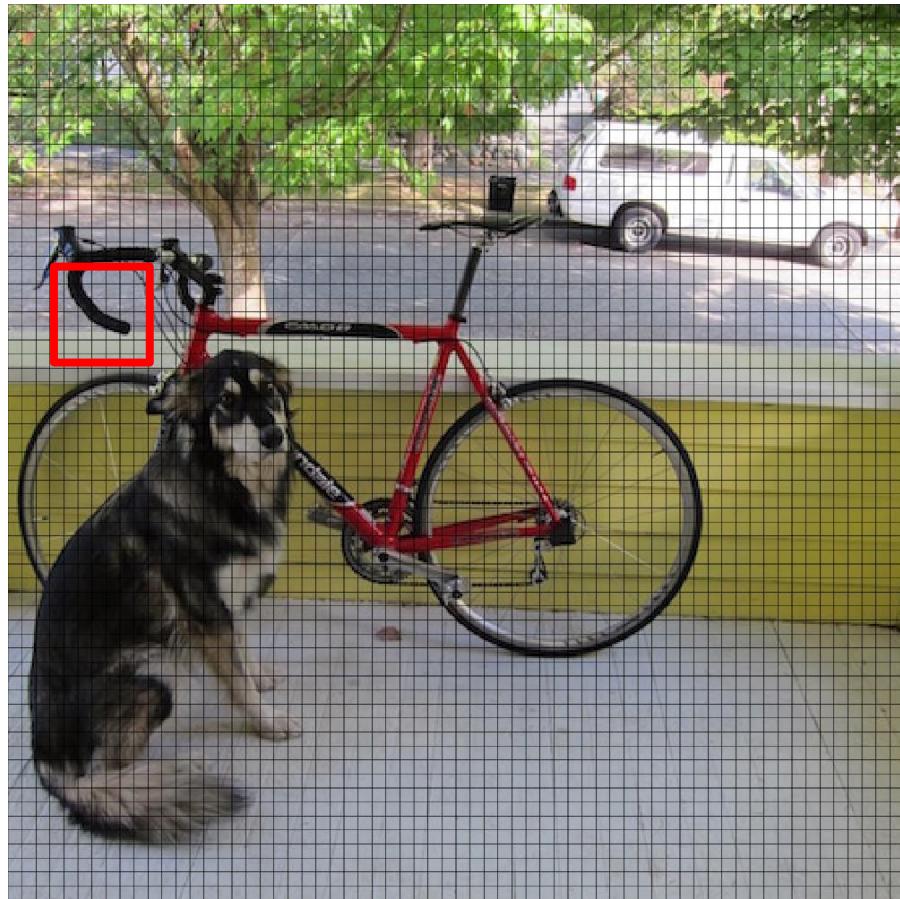
448x448 -> 64x64



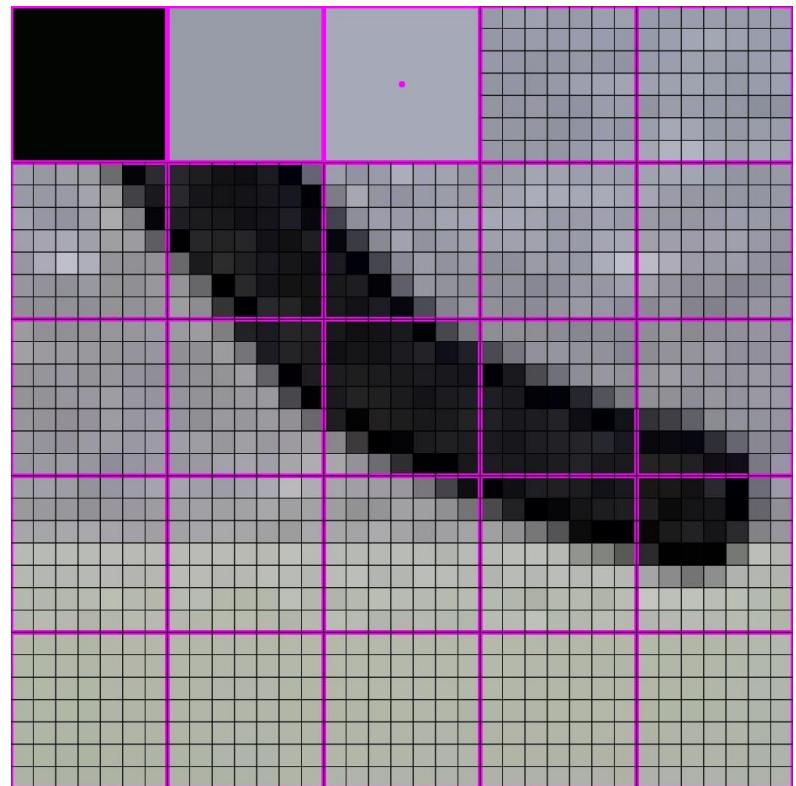
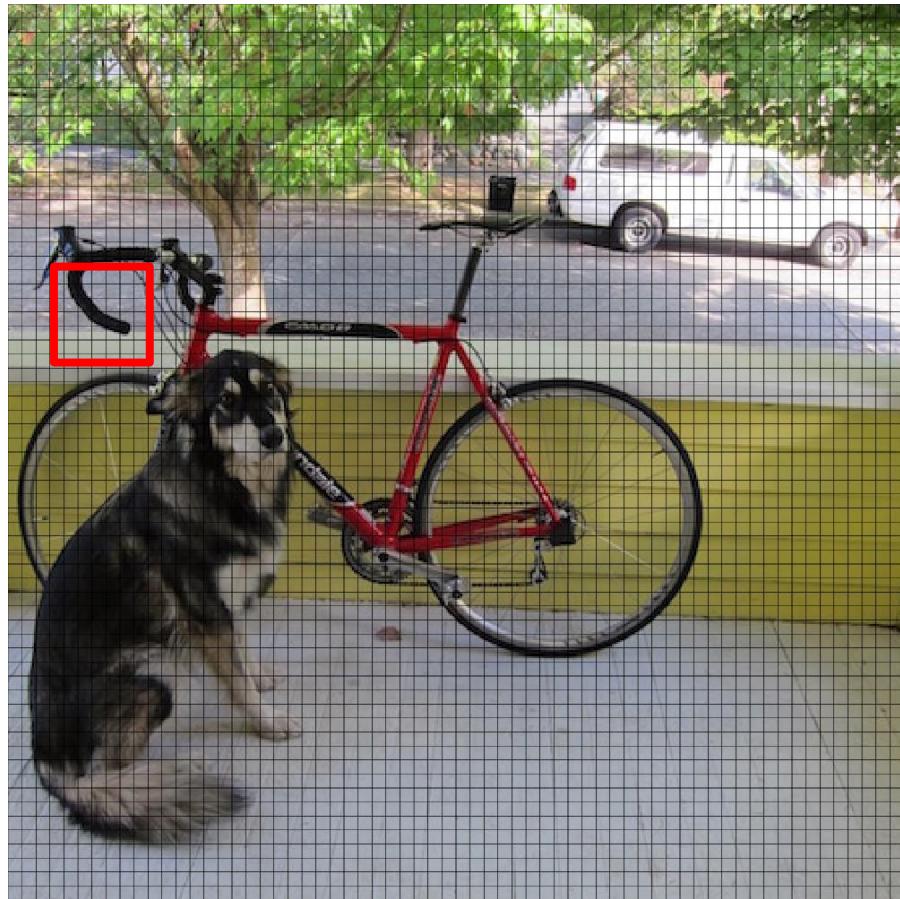
448x448 -> 64x64



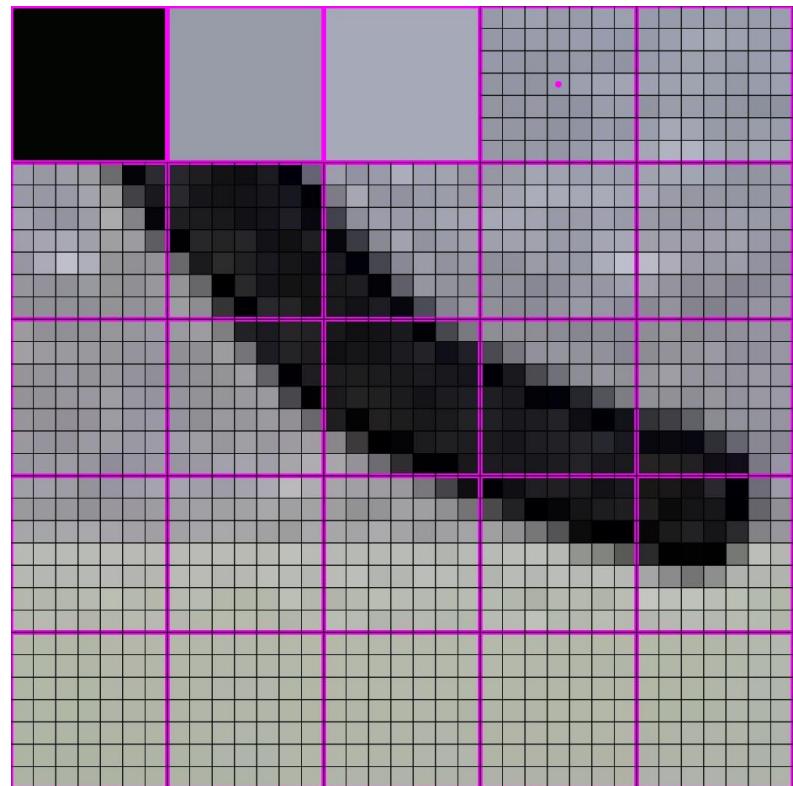
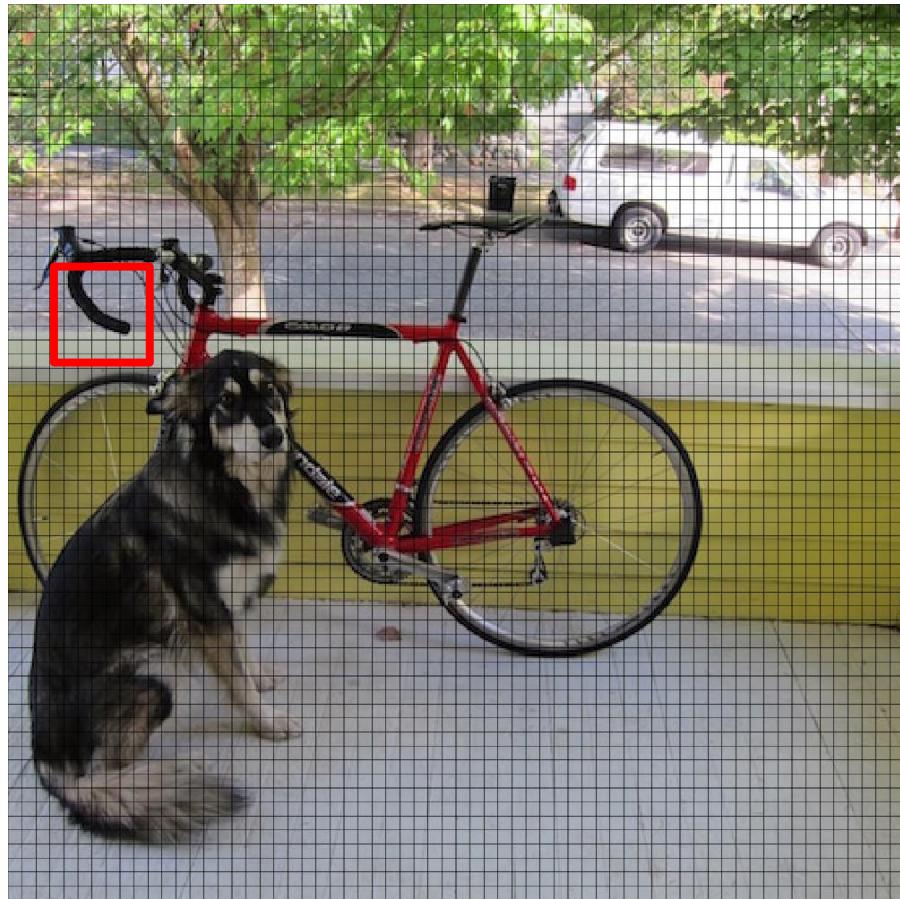
448x448 -> 64x64



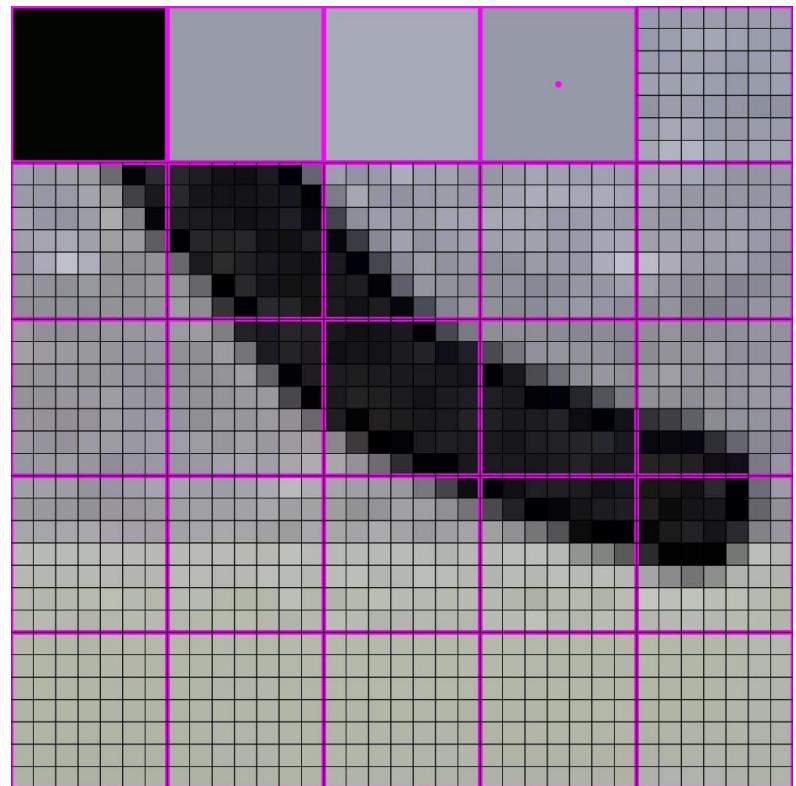
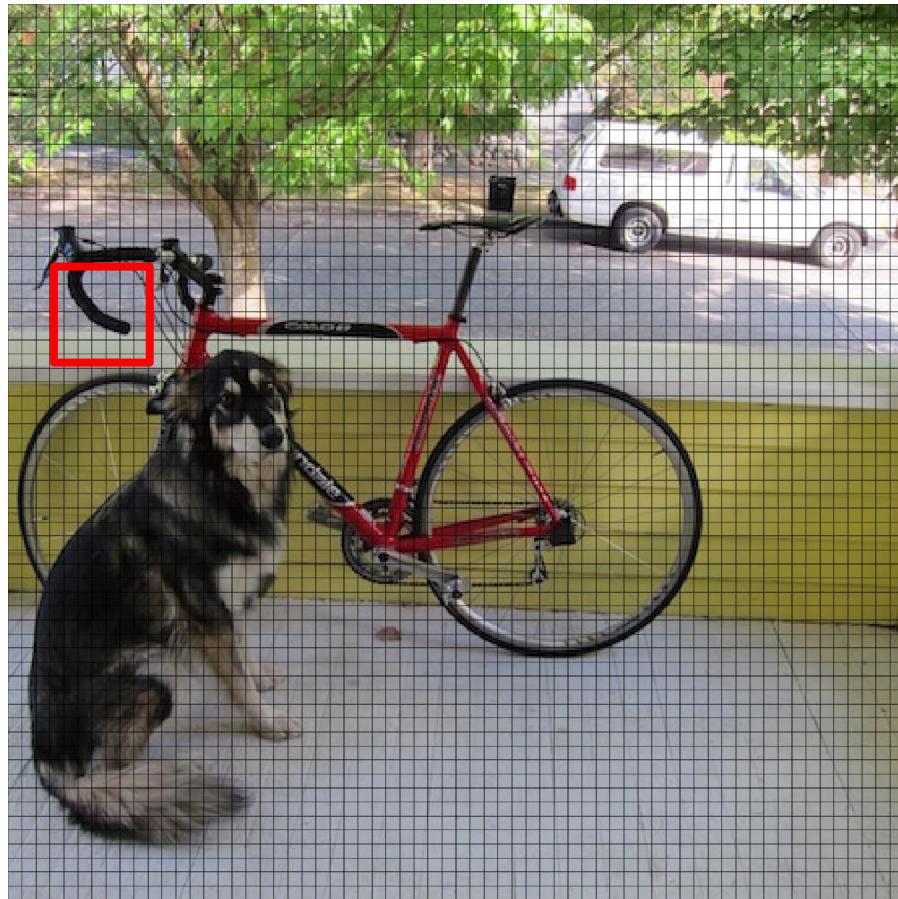
448x448 -> 64x64



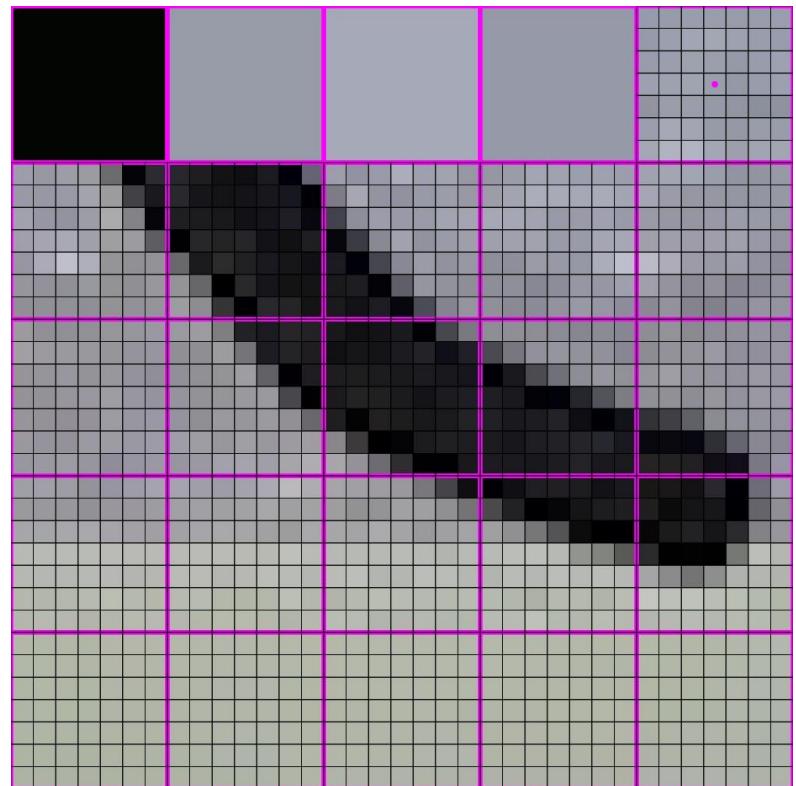
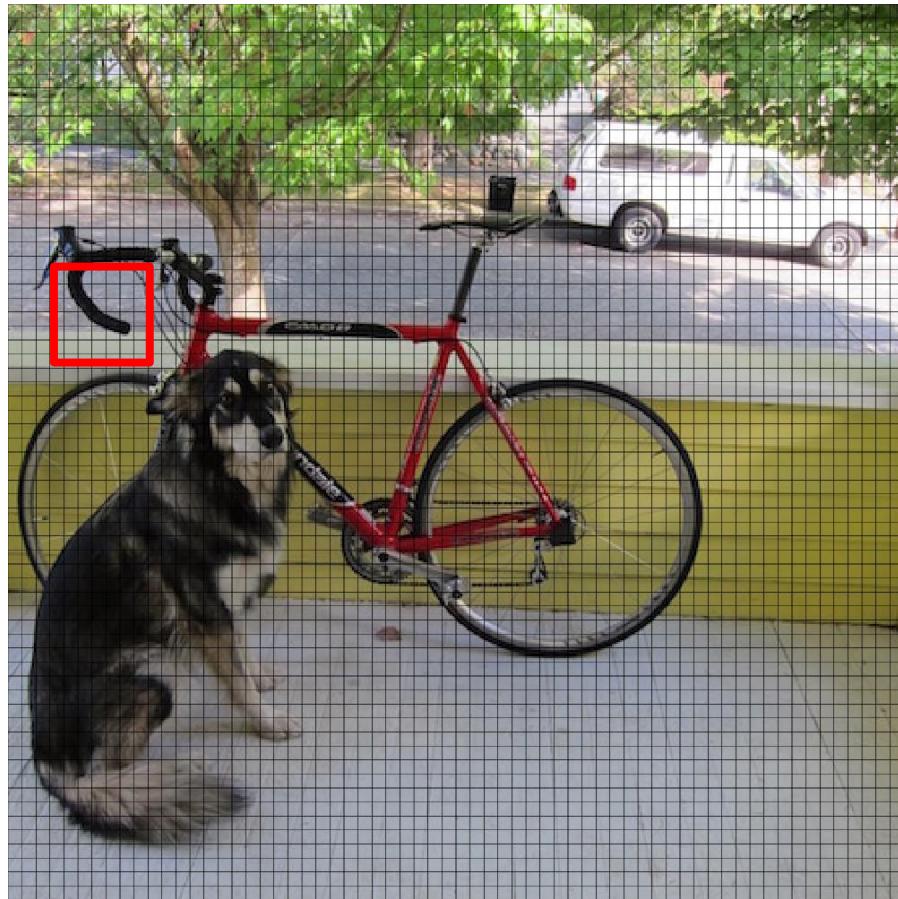
448x448 -> 64x64



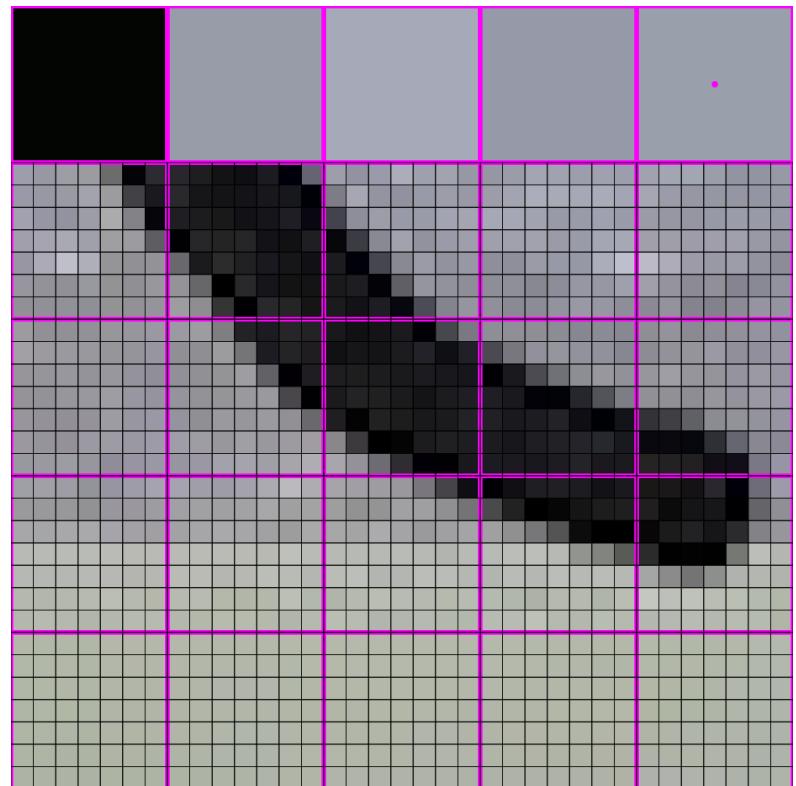
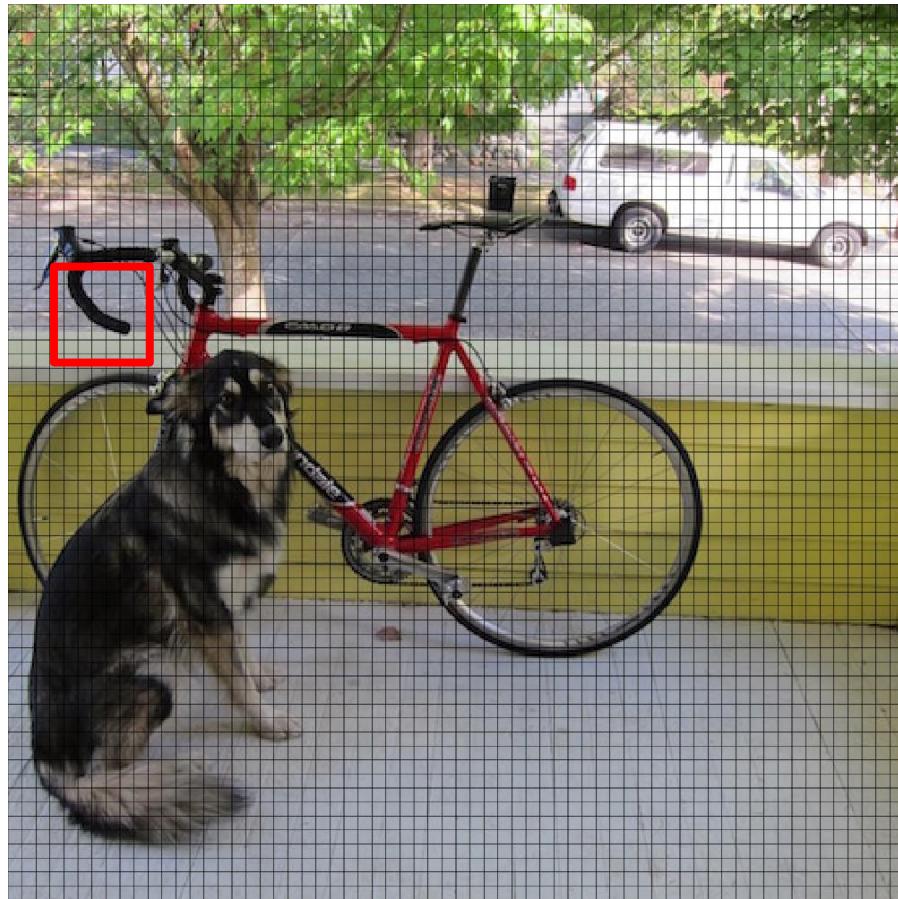
448x448 -> 64x64



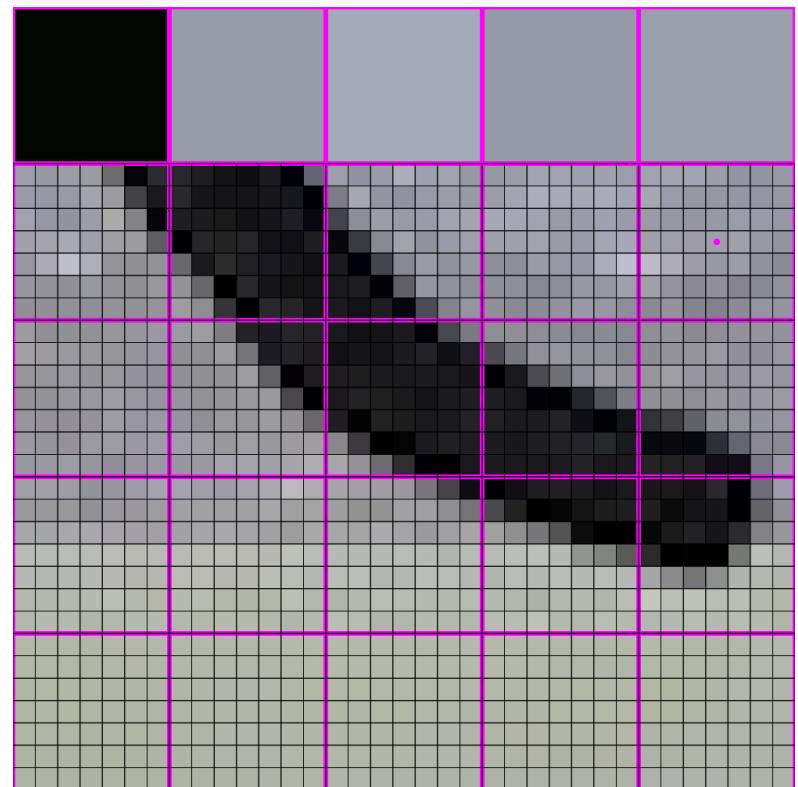
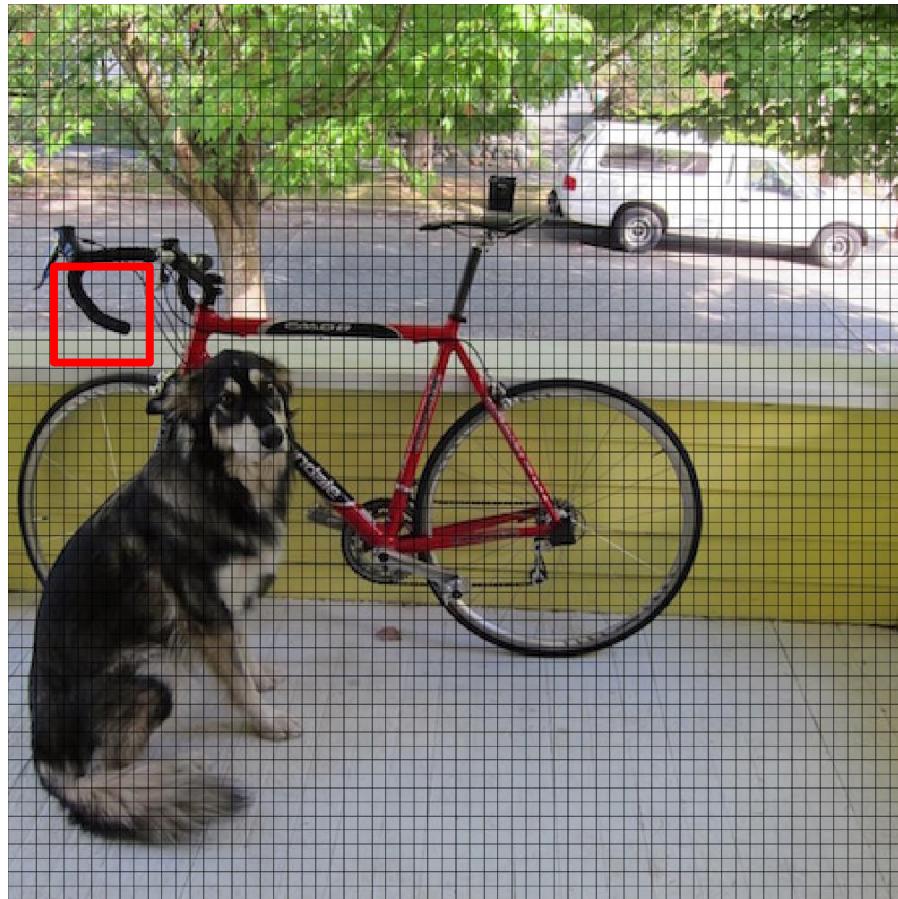
448x448 -> 64x64



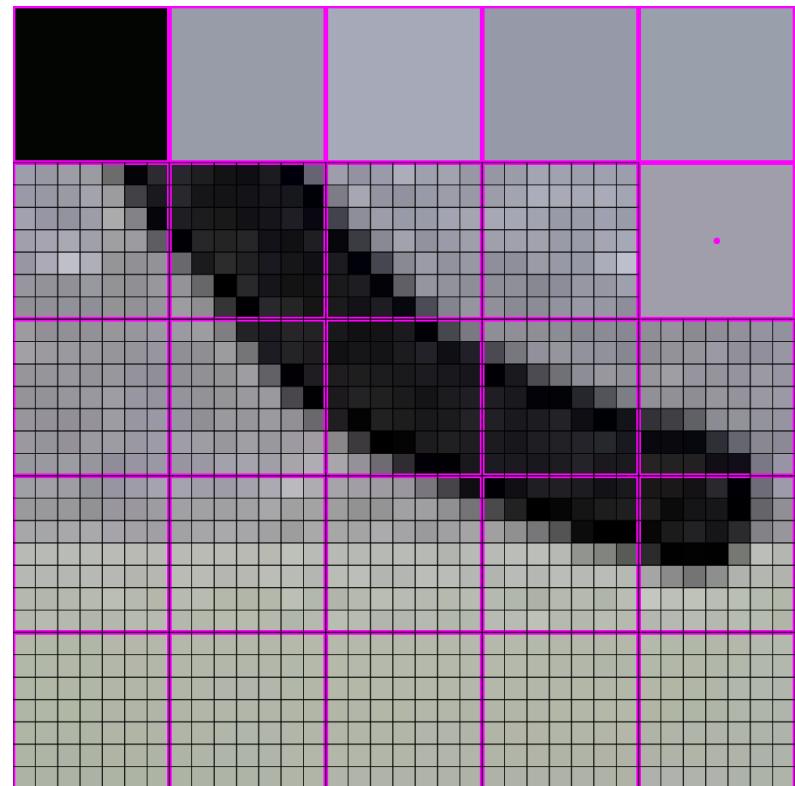
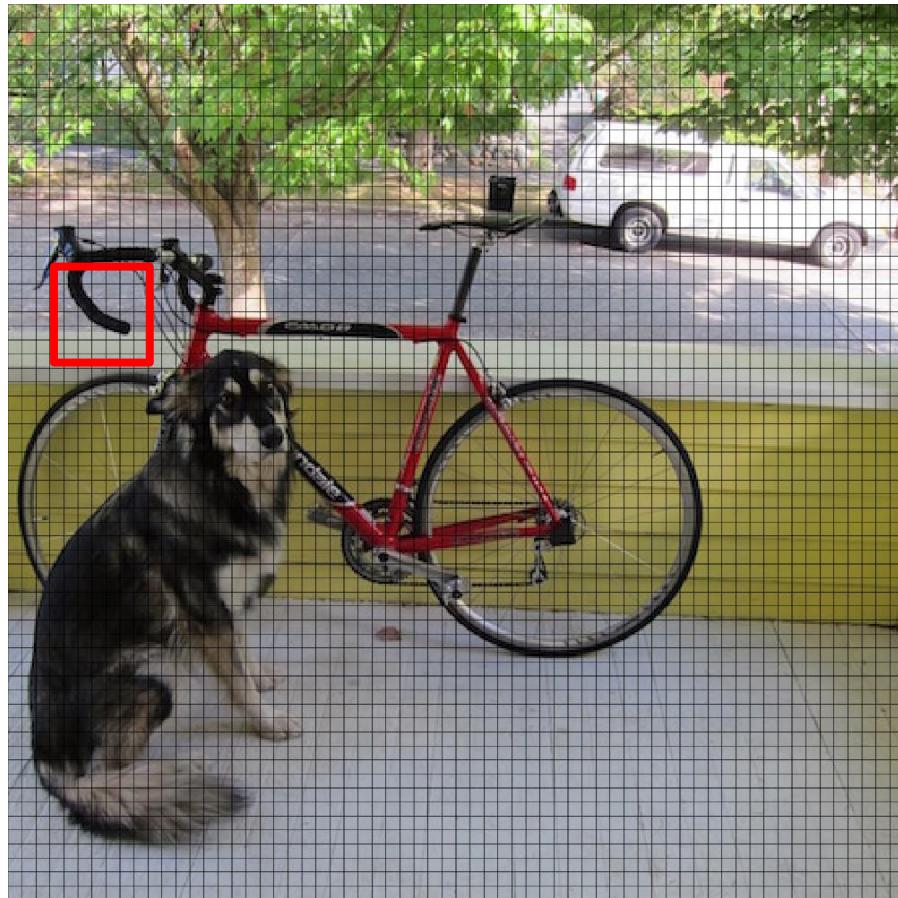
448x448 -> 64x64



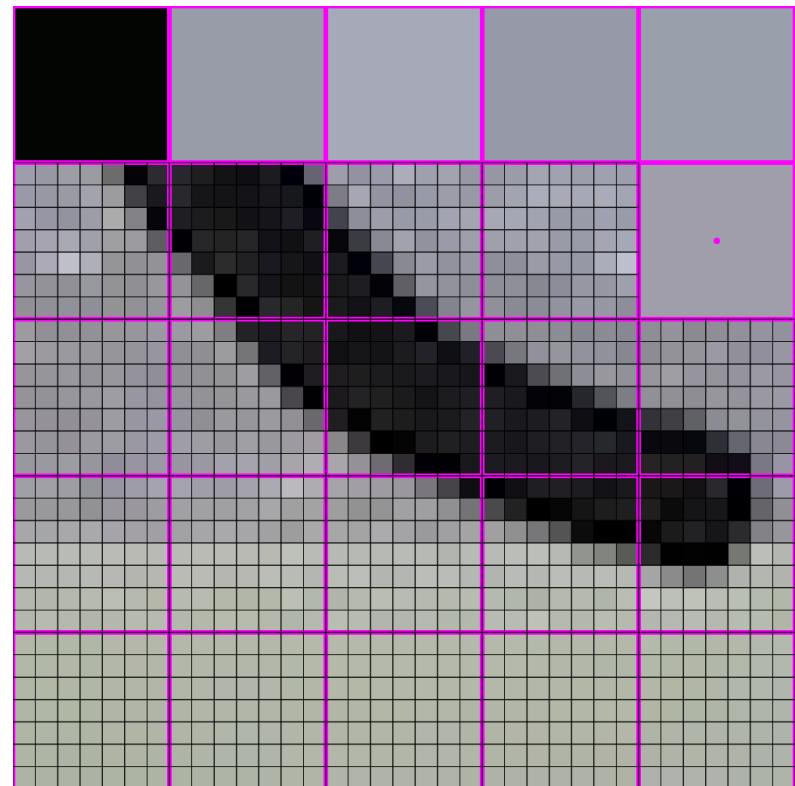
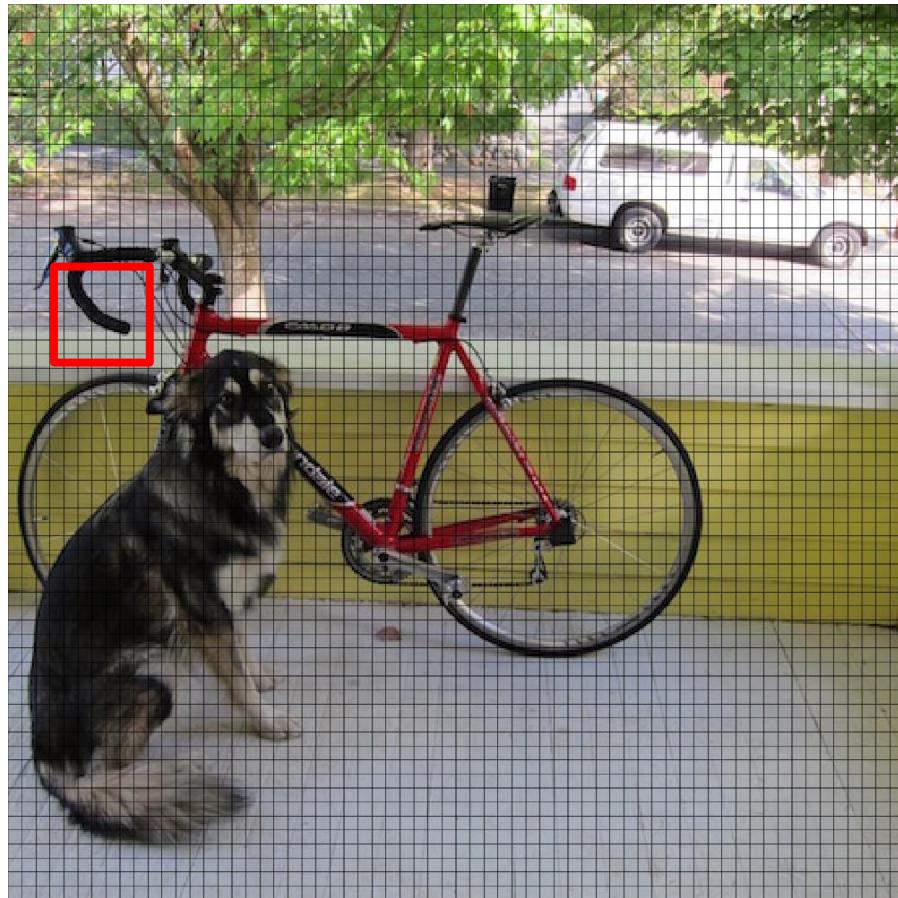
448x448 -> 64x64



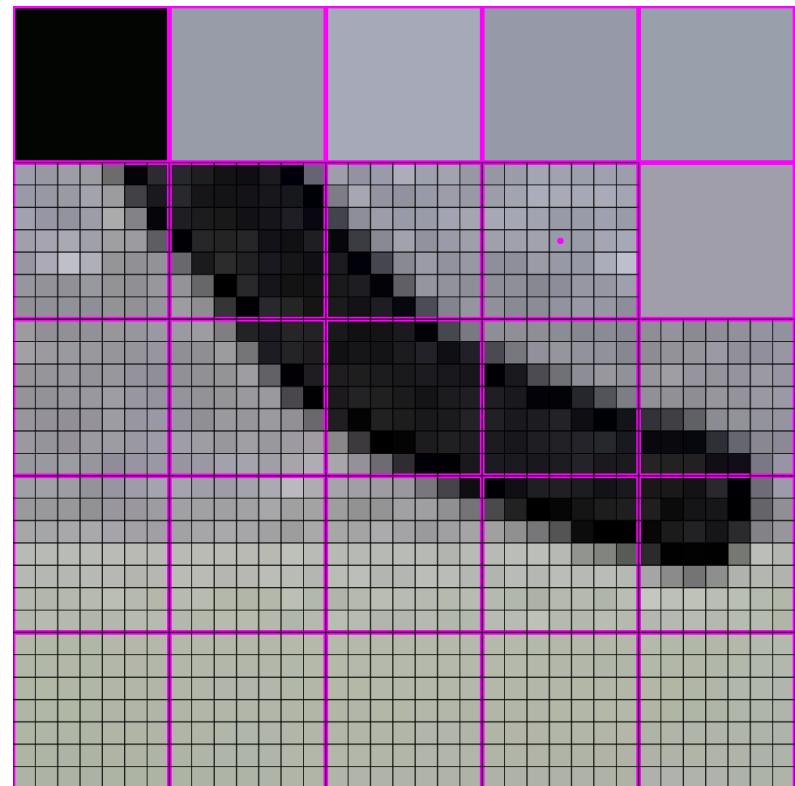
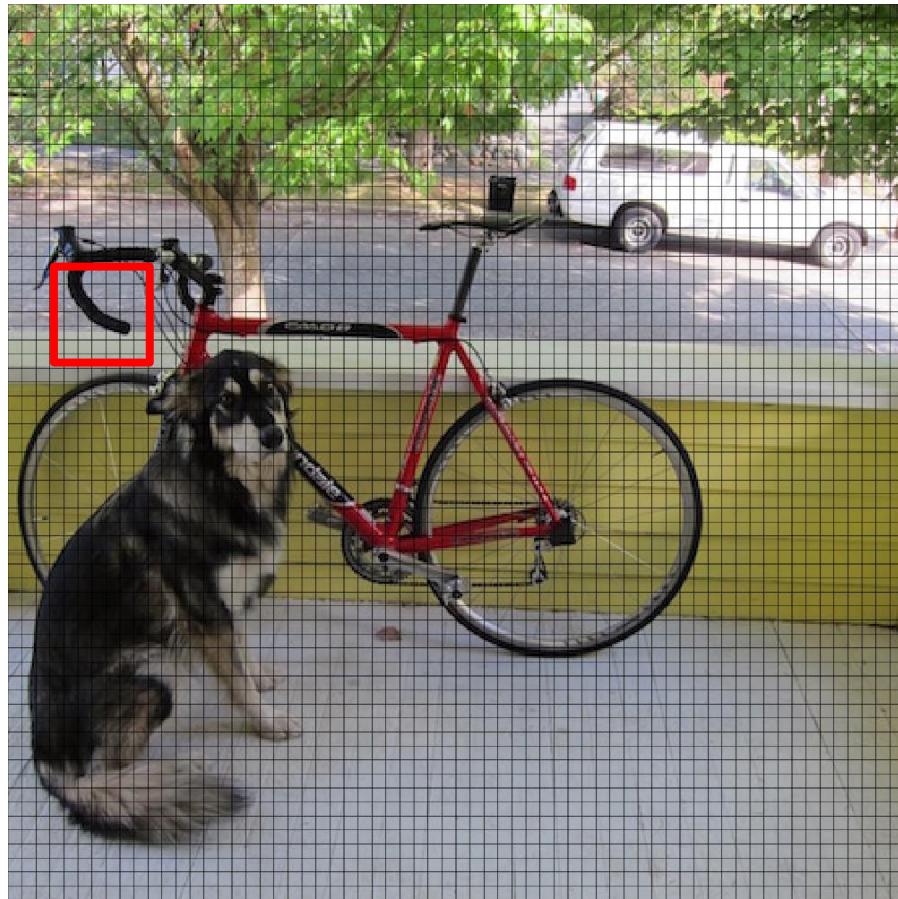
448x448 -> 64x64



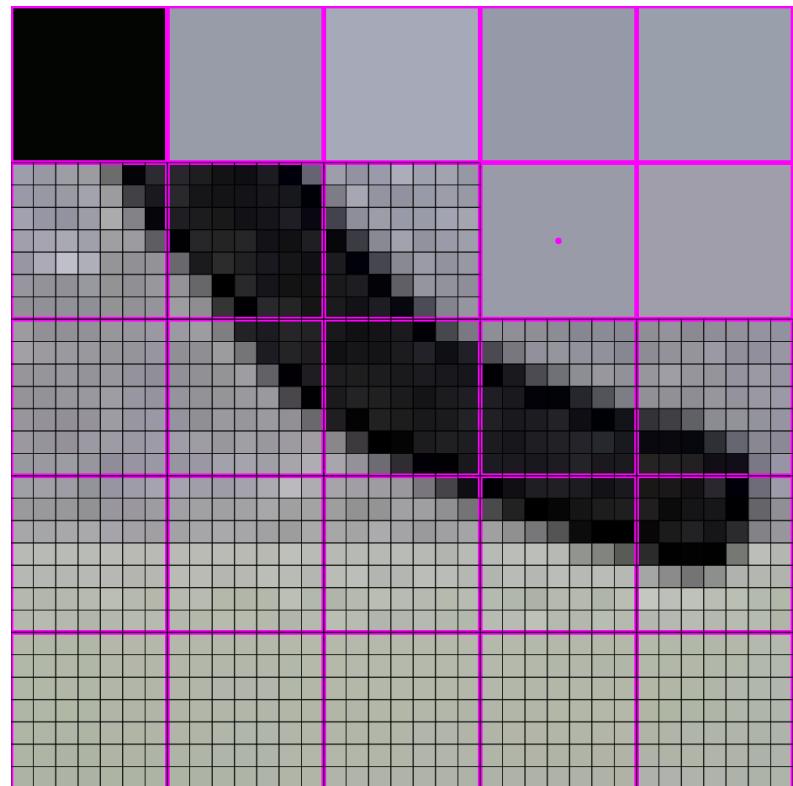
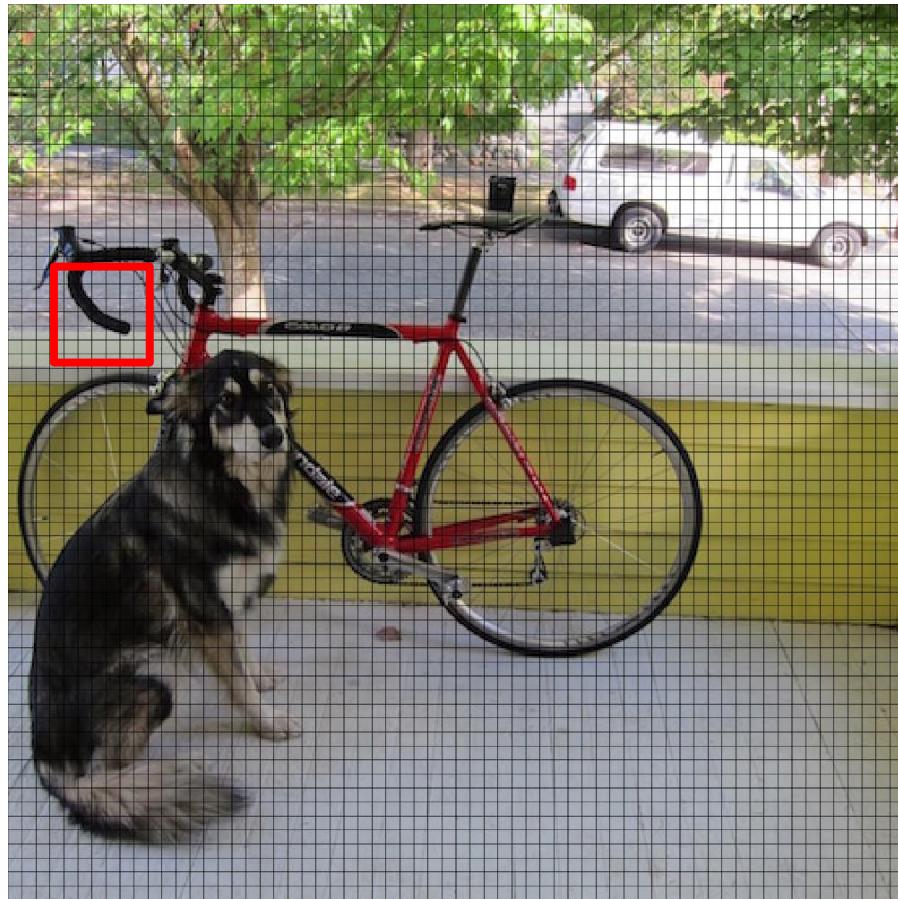
448x448 -> 64x64



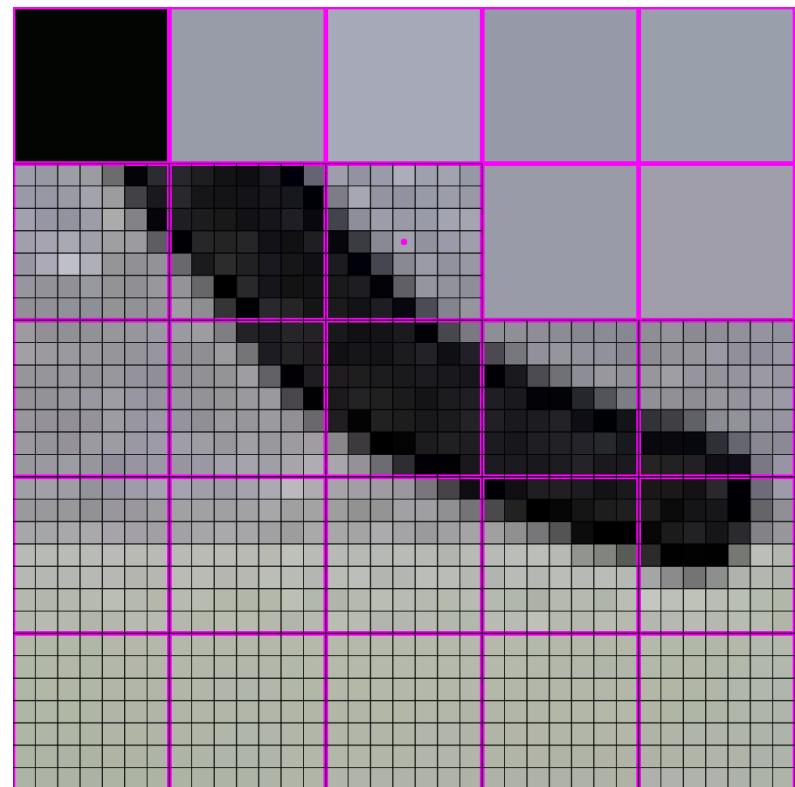
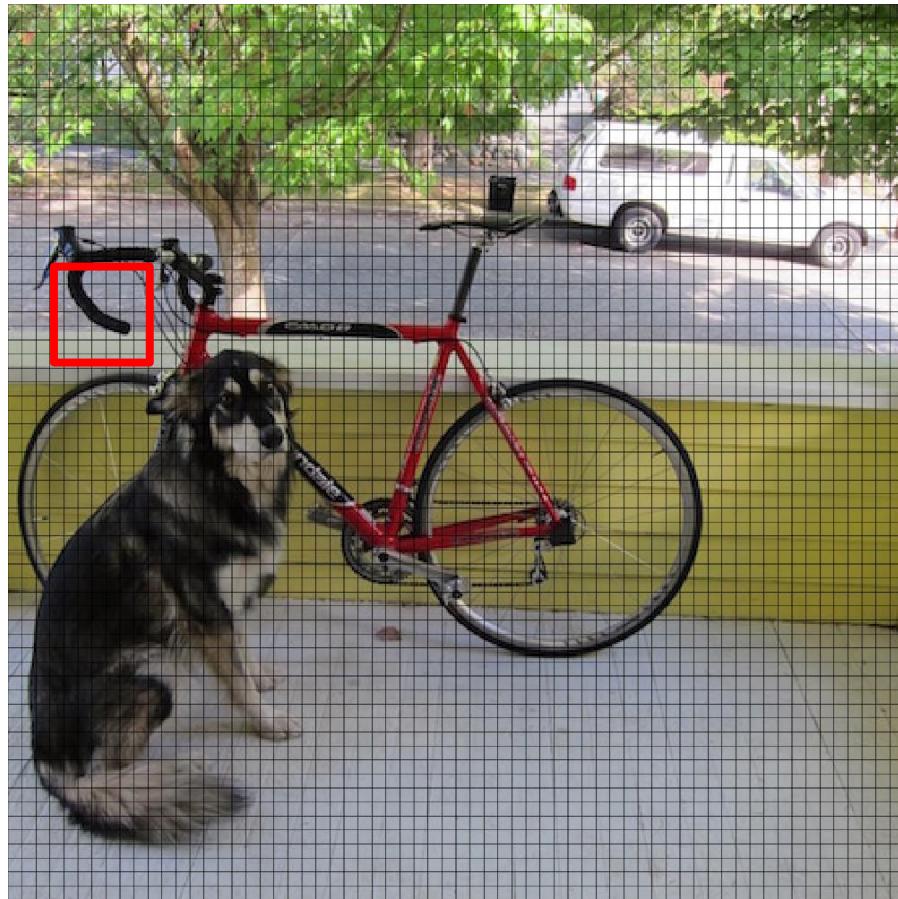
448x448 -> 64x64



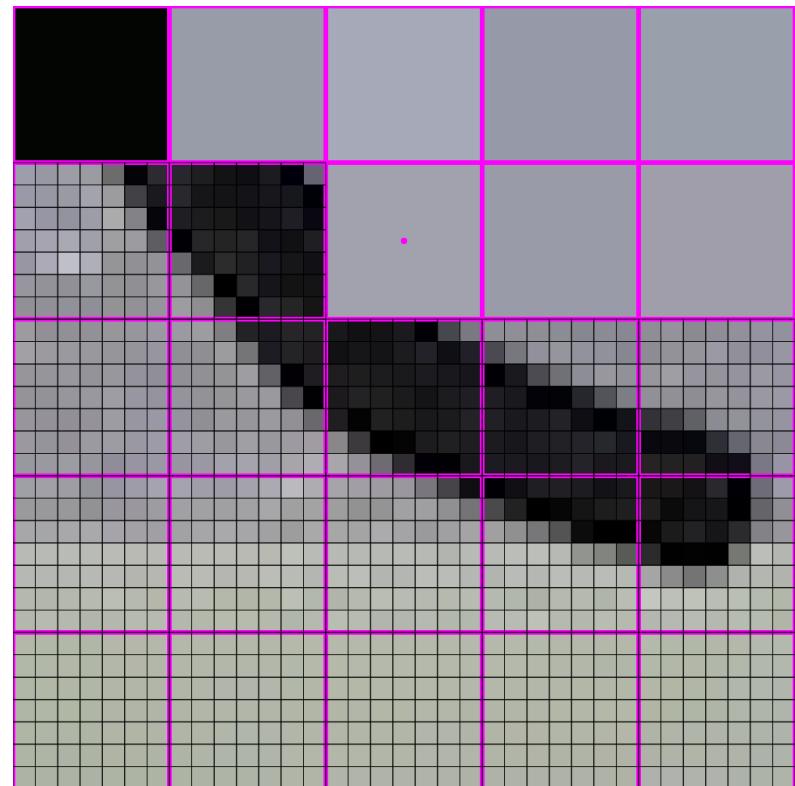
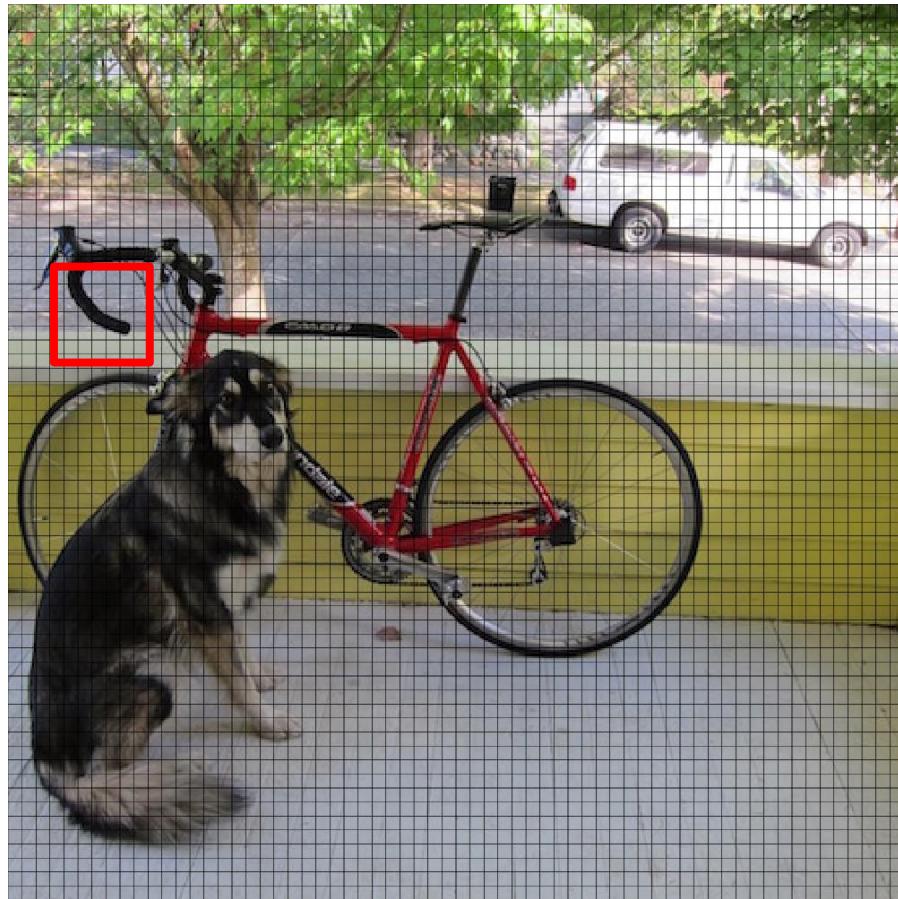
448x448 -> 64x64



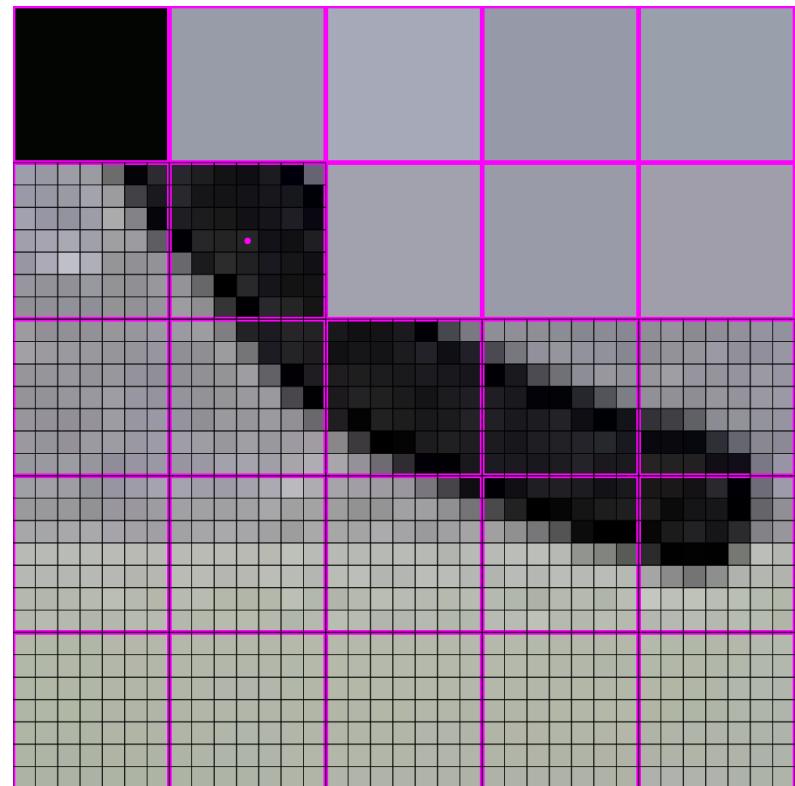
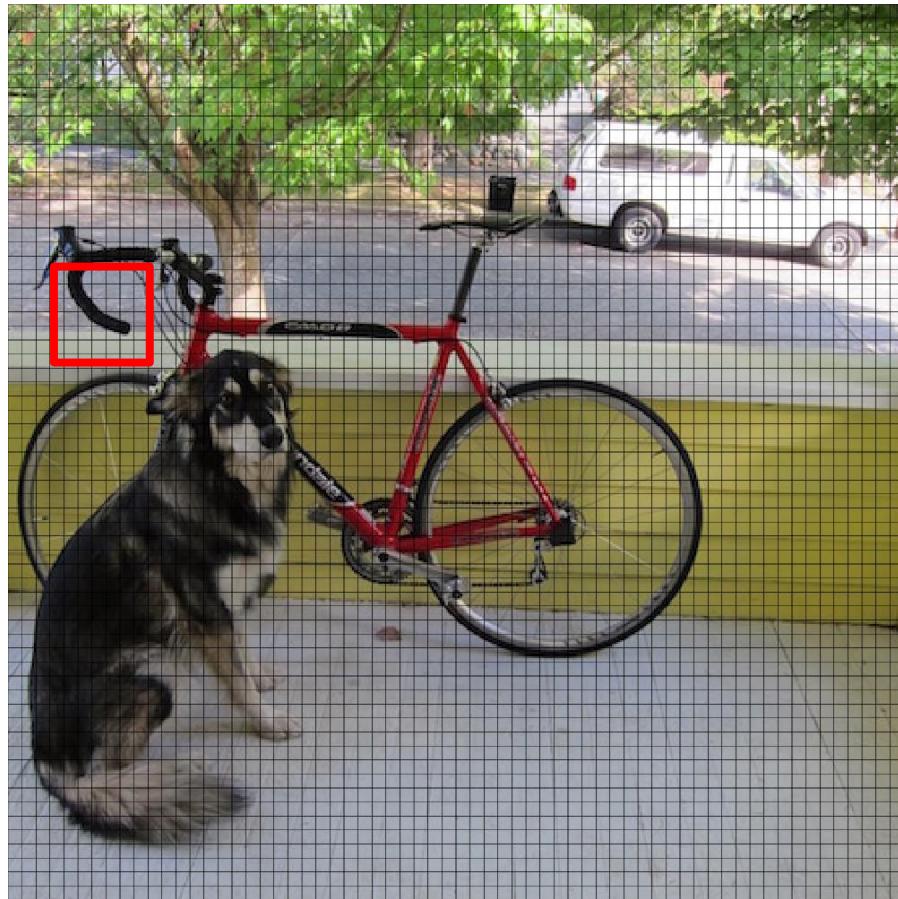
448x448 -> 64x64



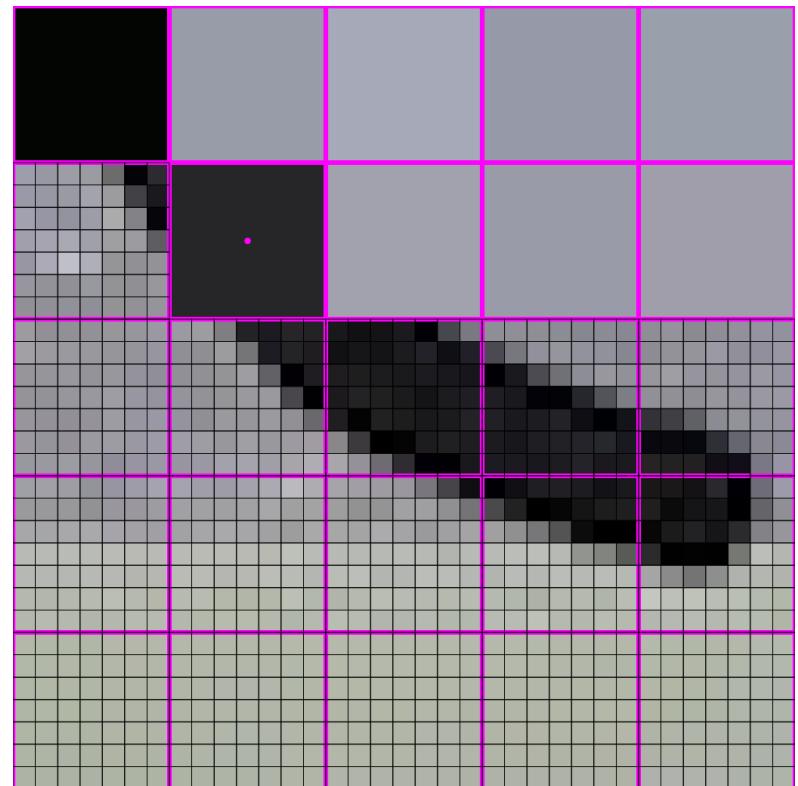
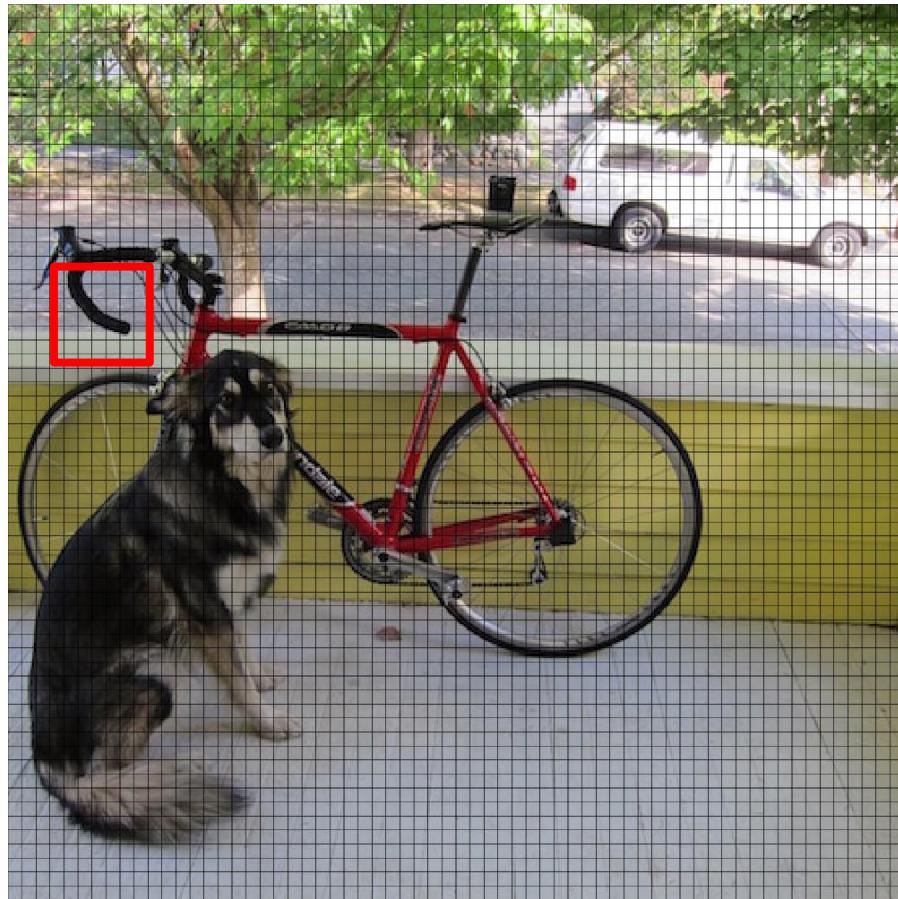
448x448 -> 64x64



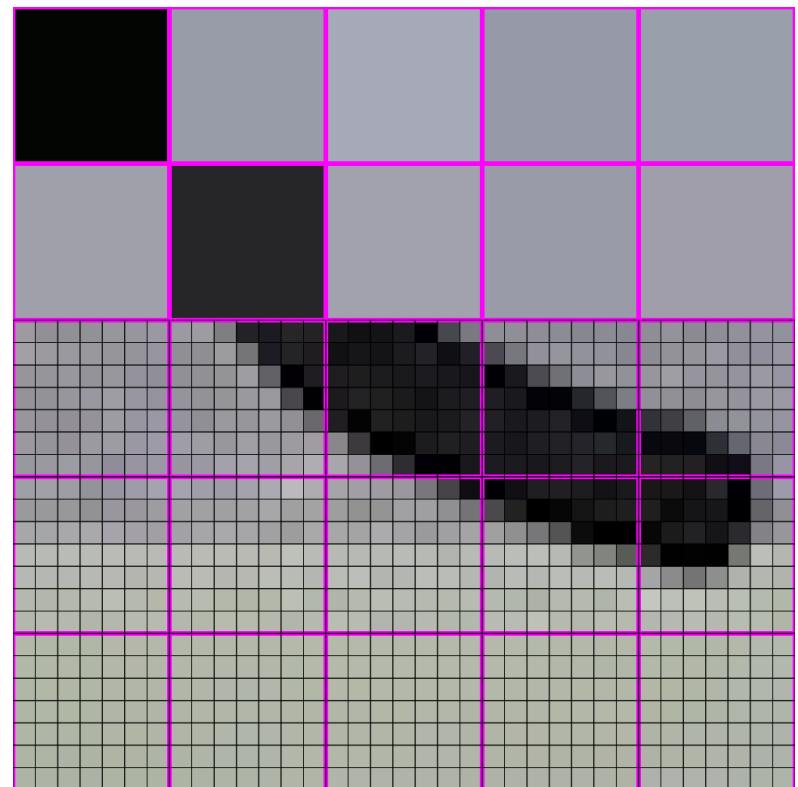
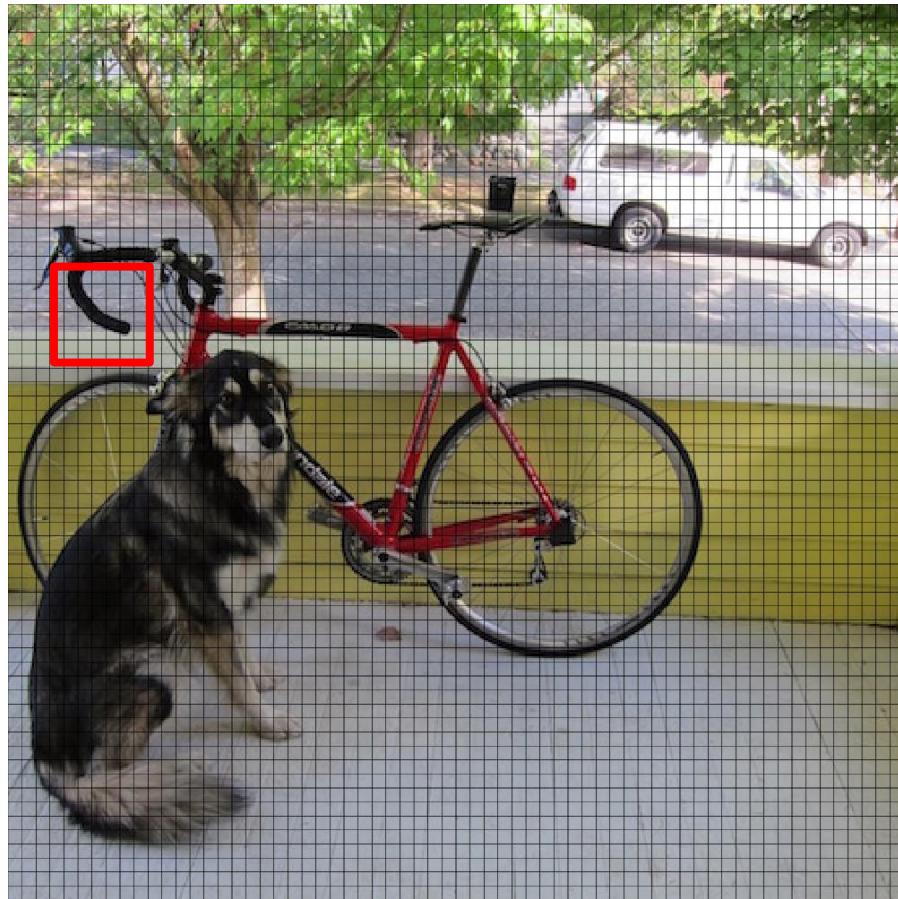
448x448 -> 64x64



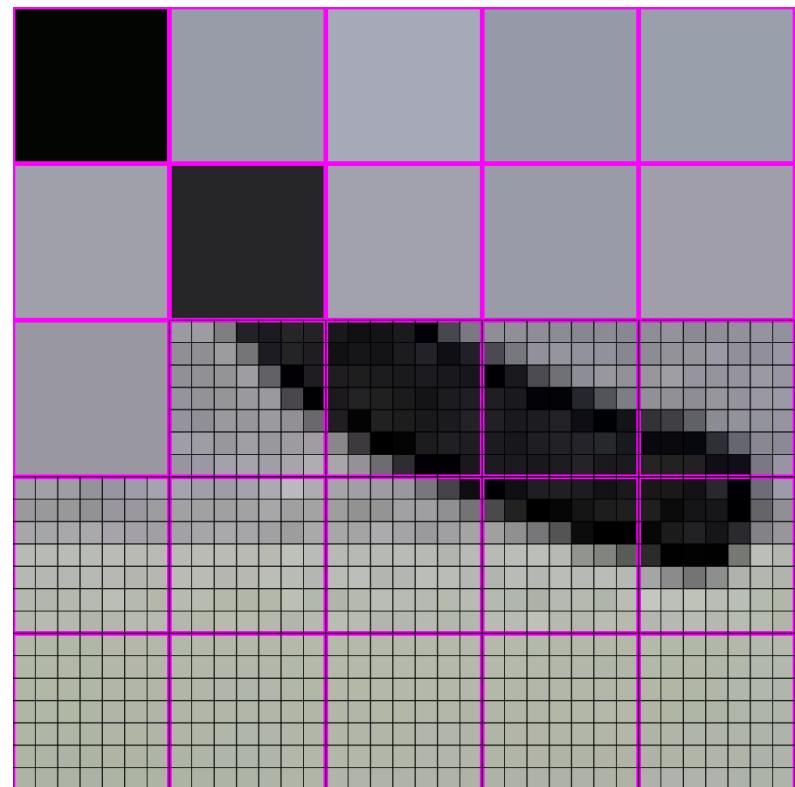
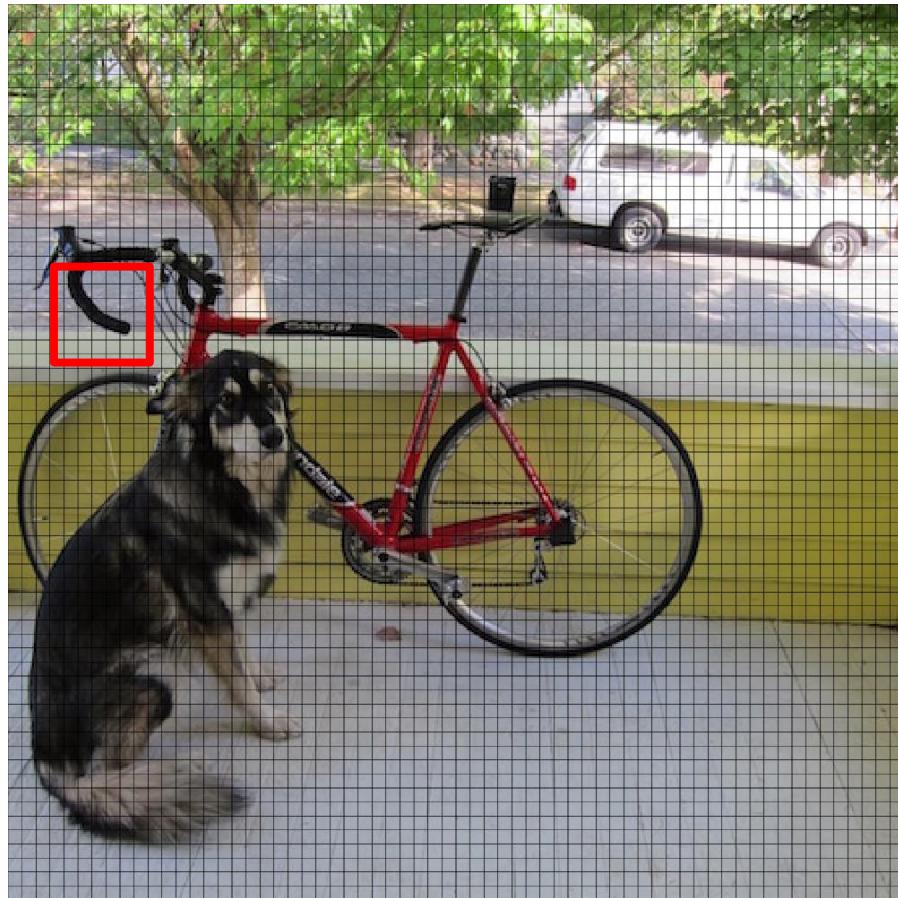
448x448 -> 64x64



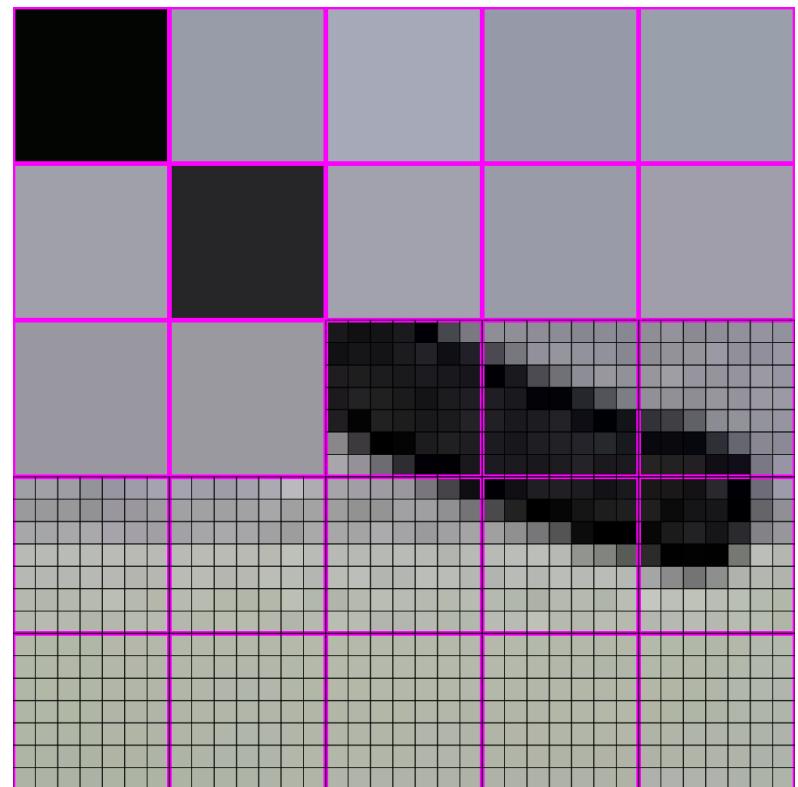
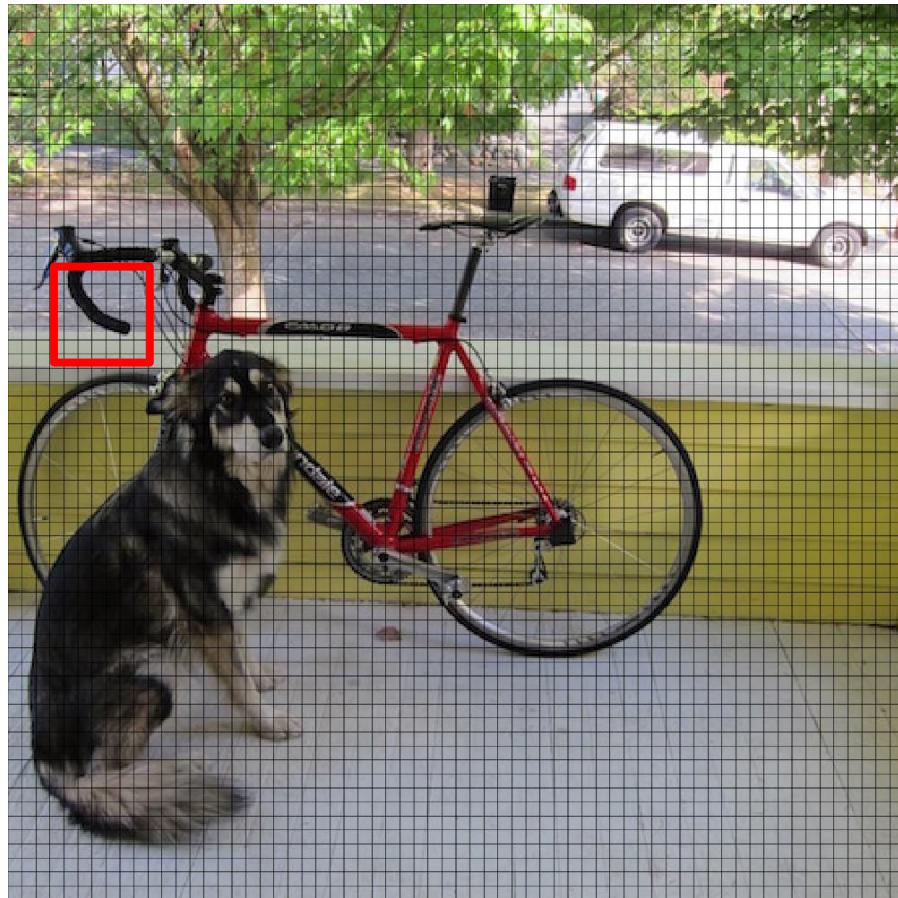
448x448 -> 64x64



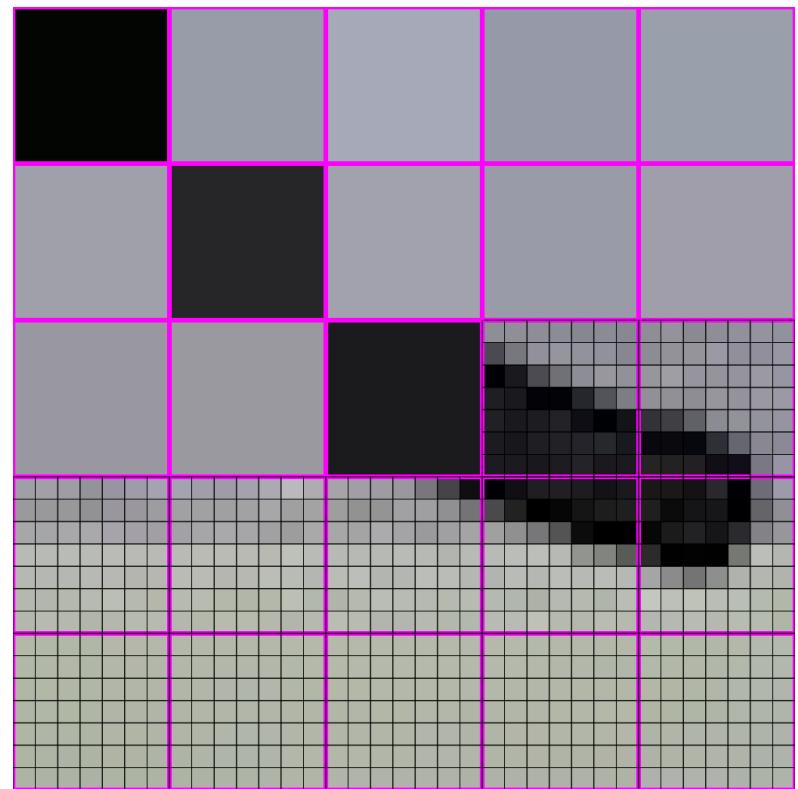
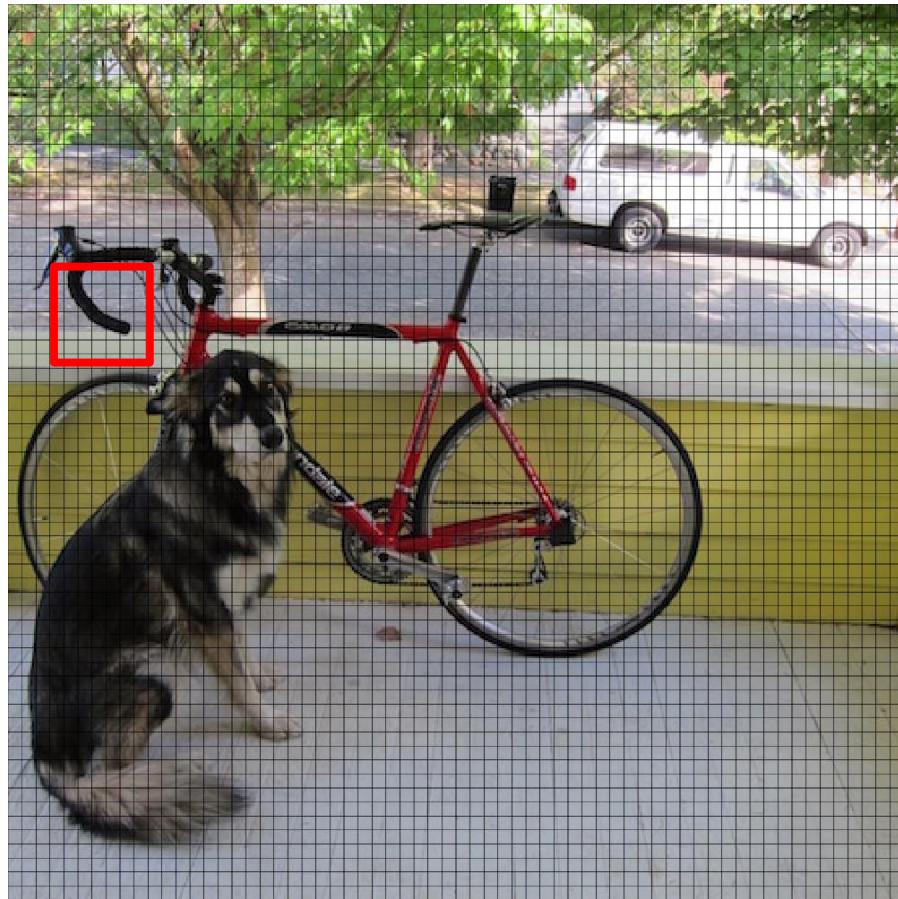
448x448 -> 64x64



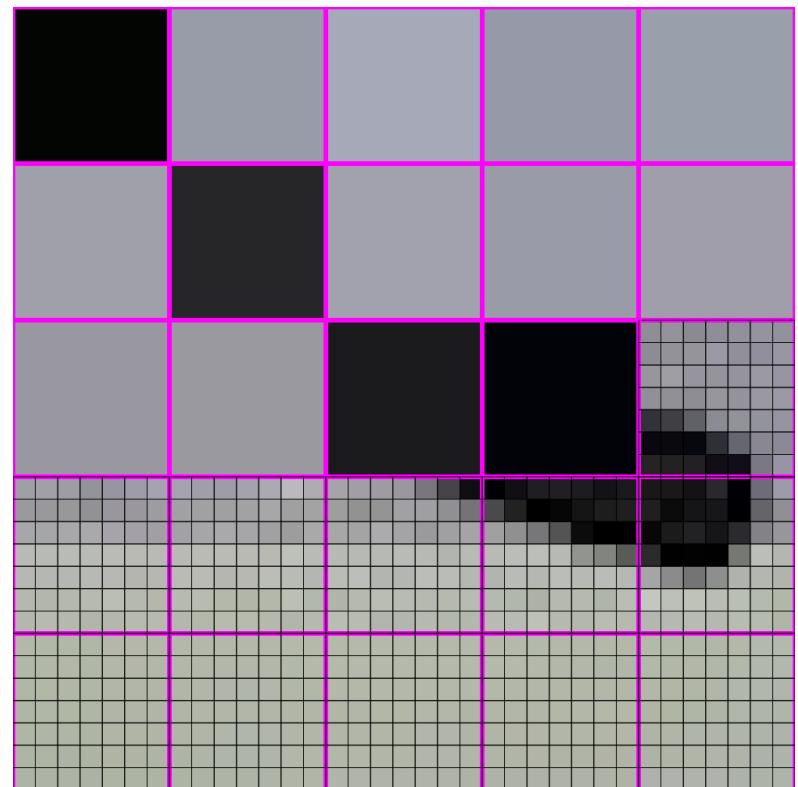
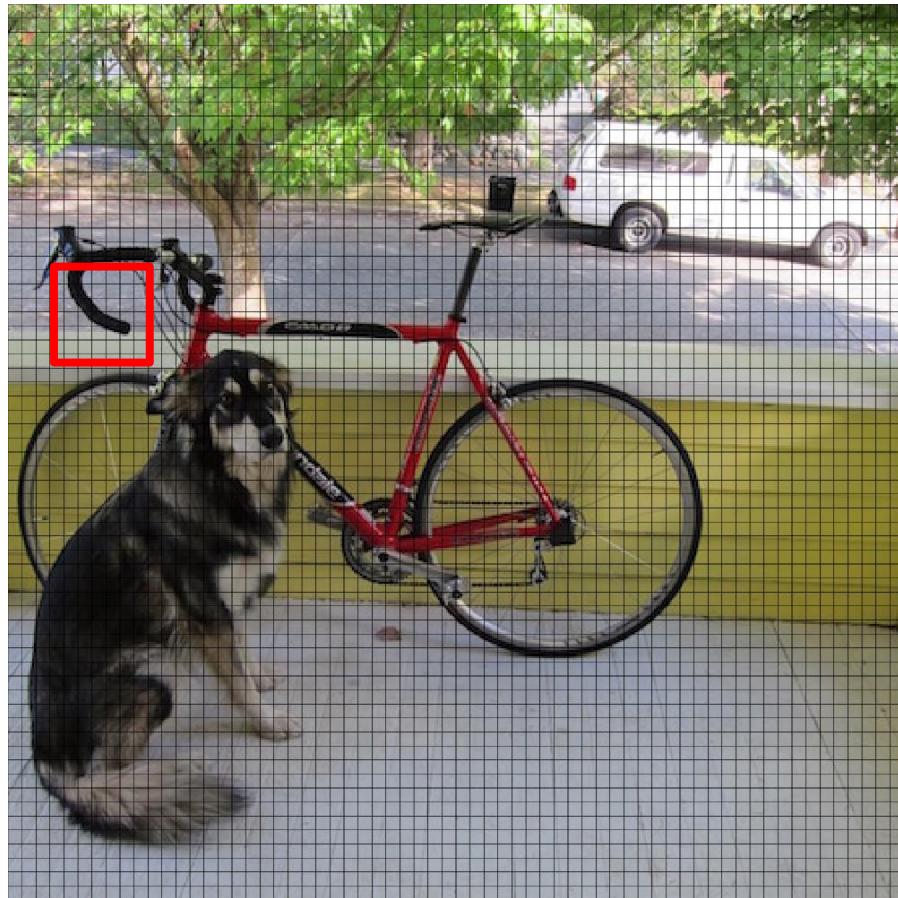
448x448 -> 64x64



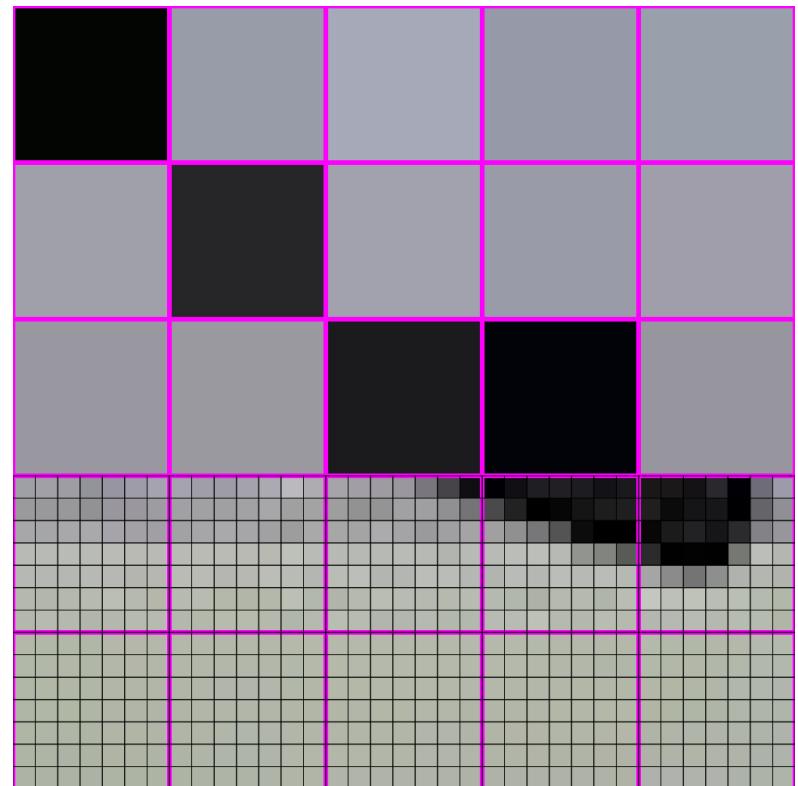
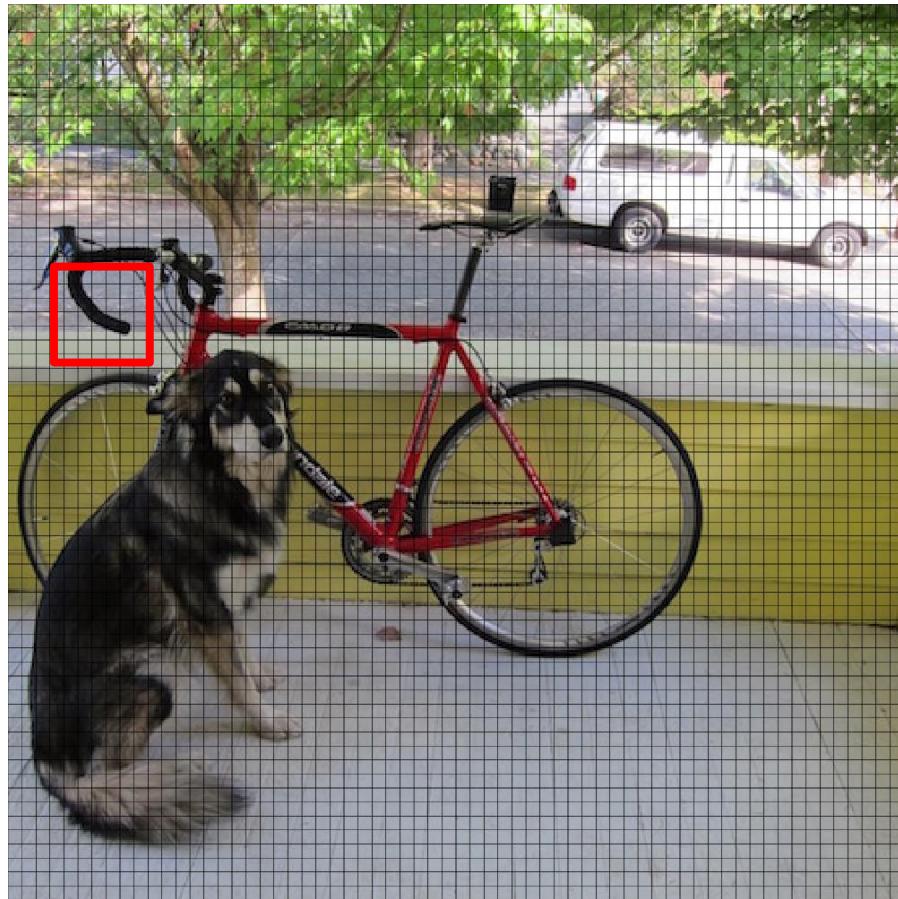
448x448 -> 64x64



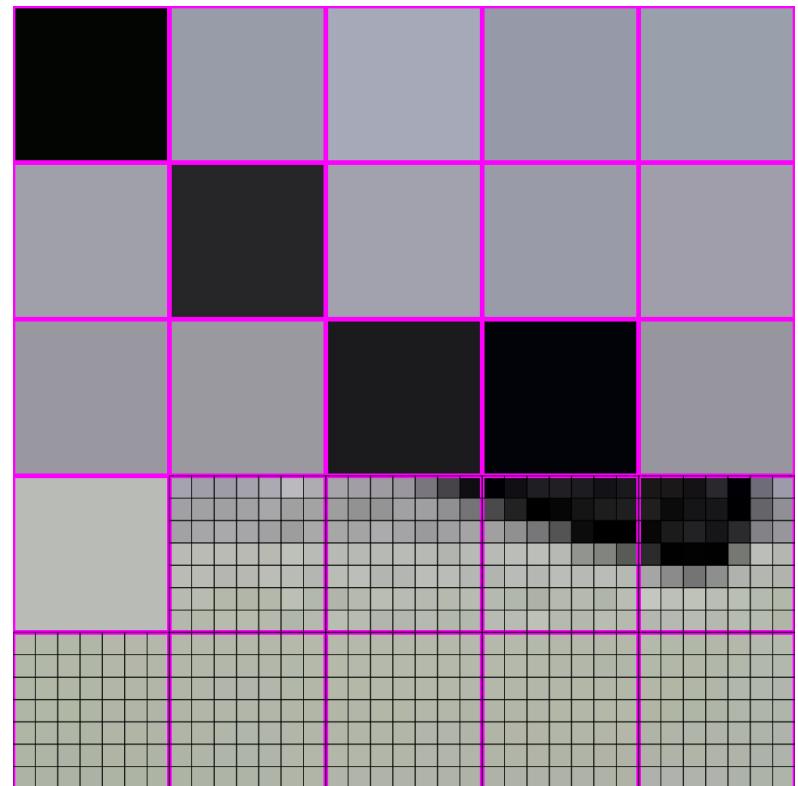
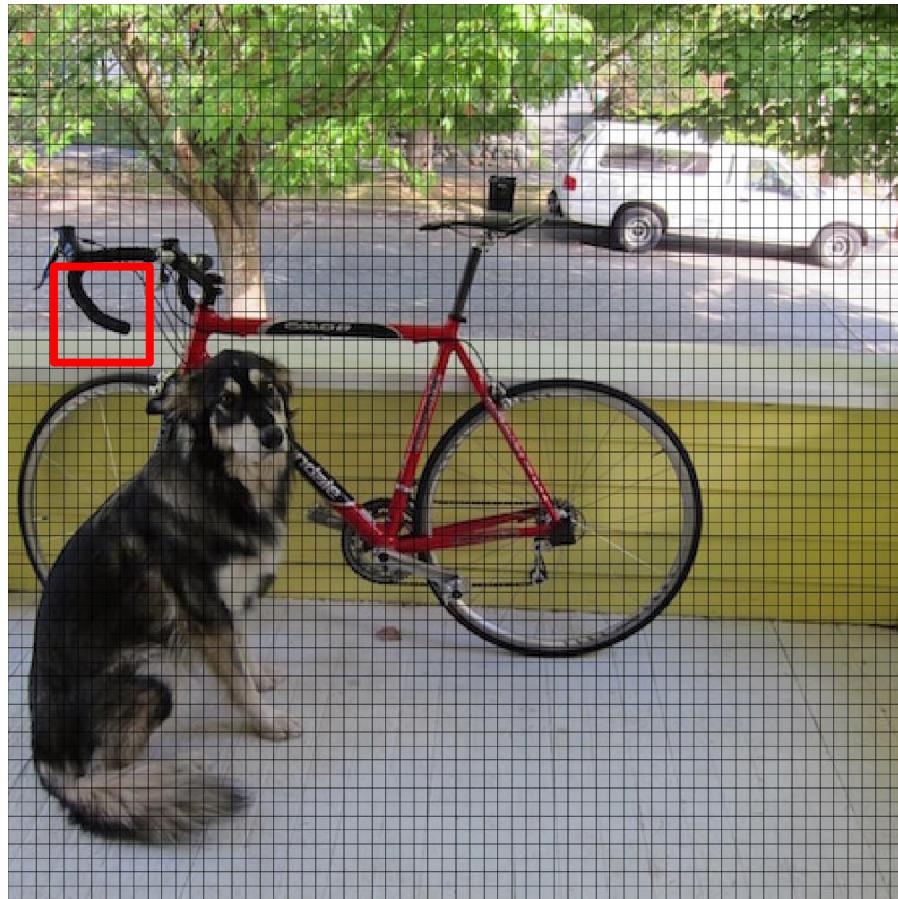
448x448 -> 64x64



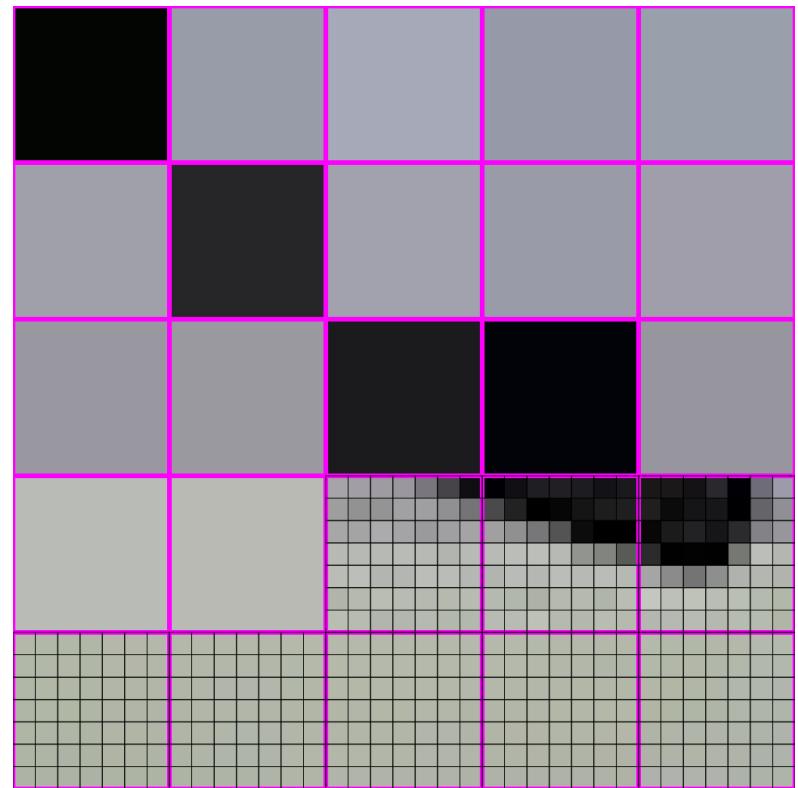
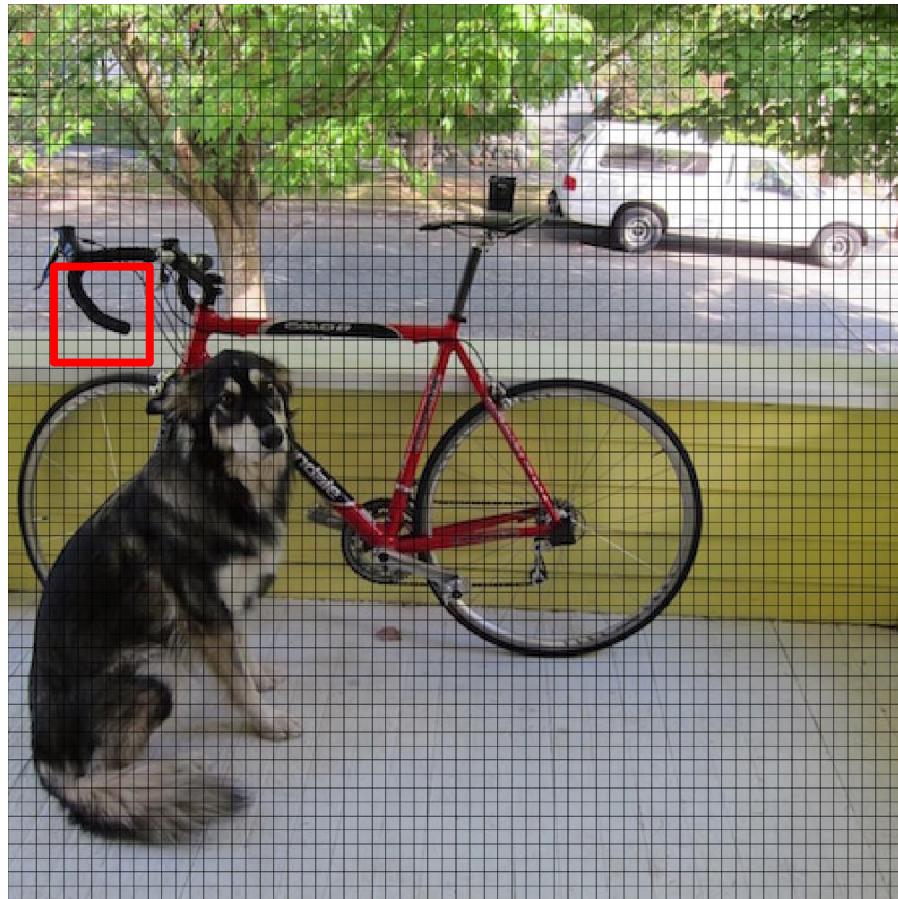
448x448 -> 64x64



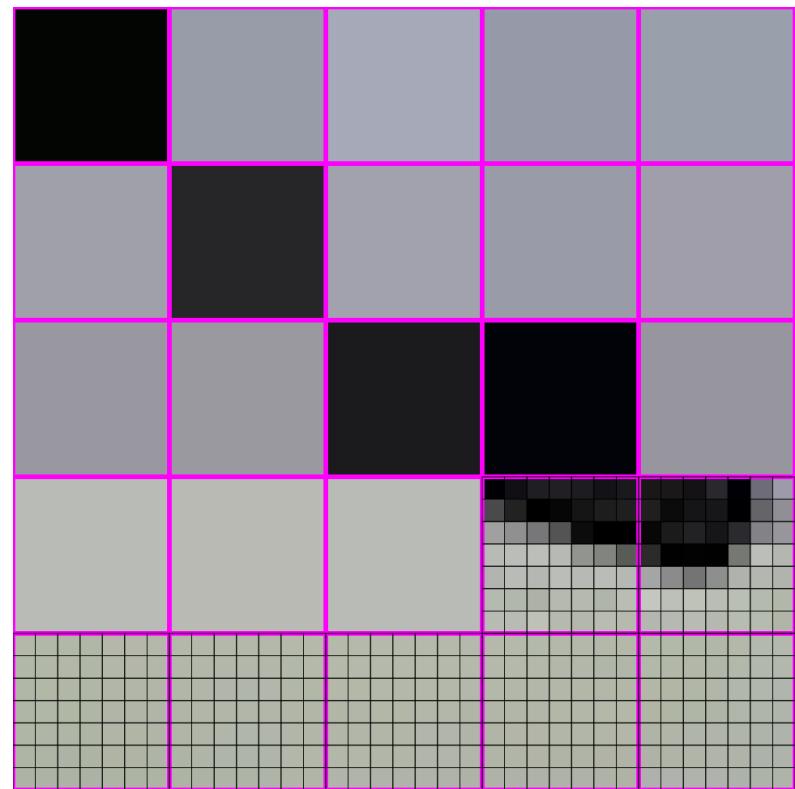
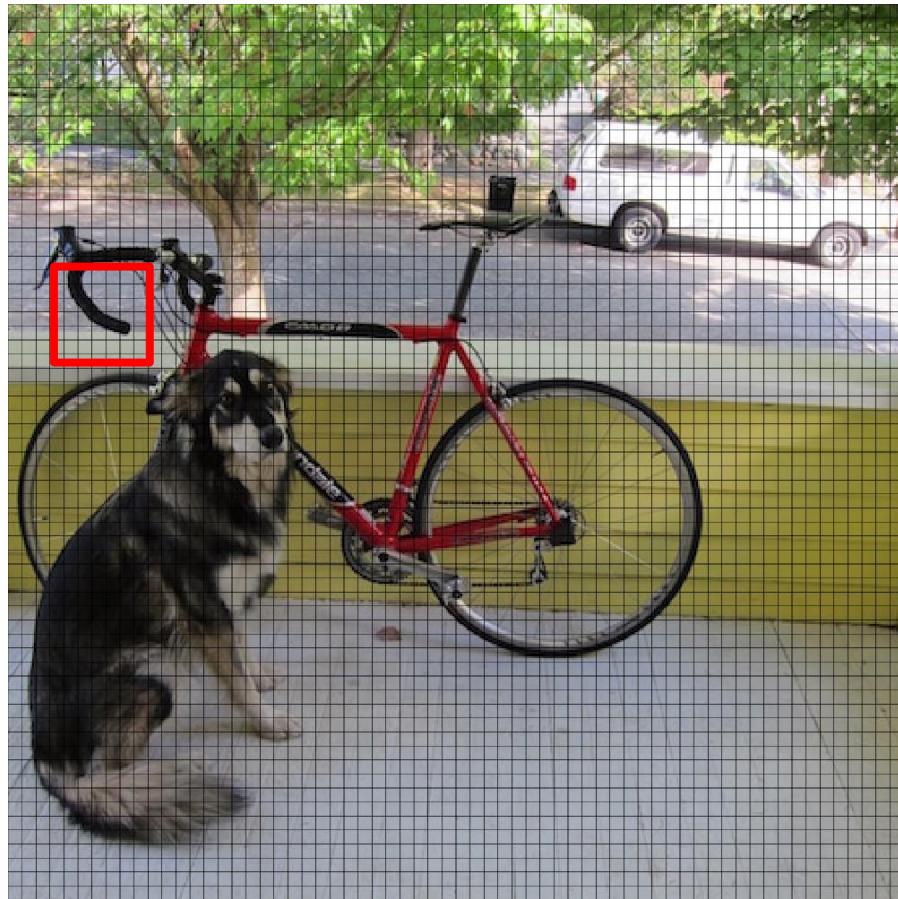
448x448 -> 64x64



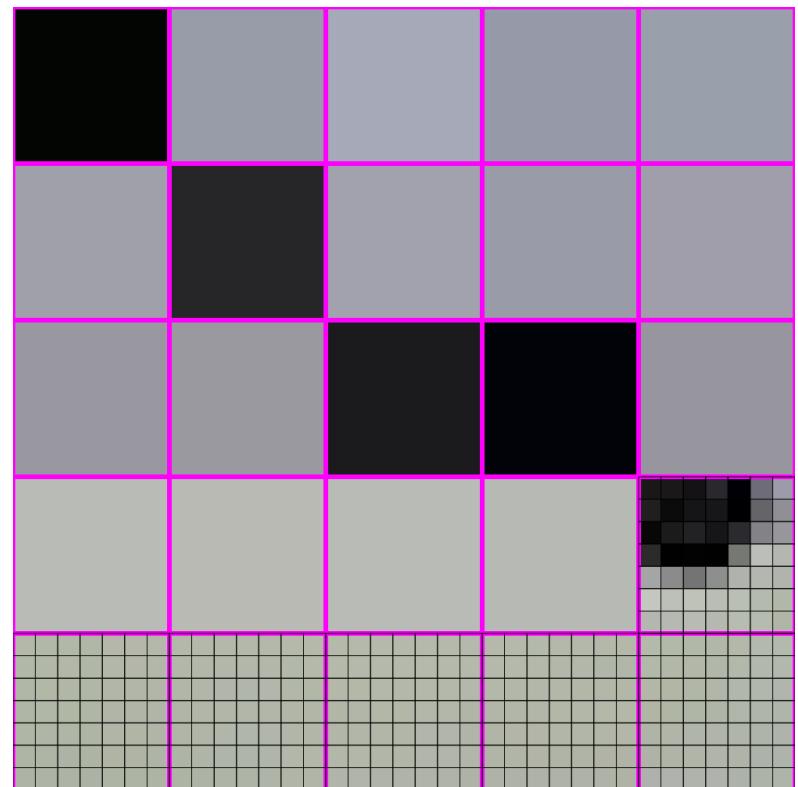
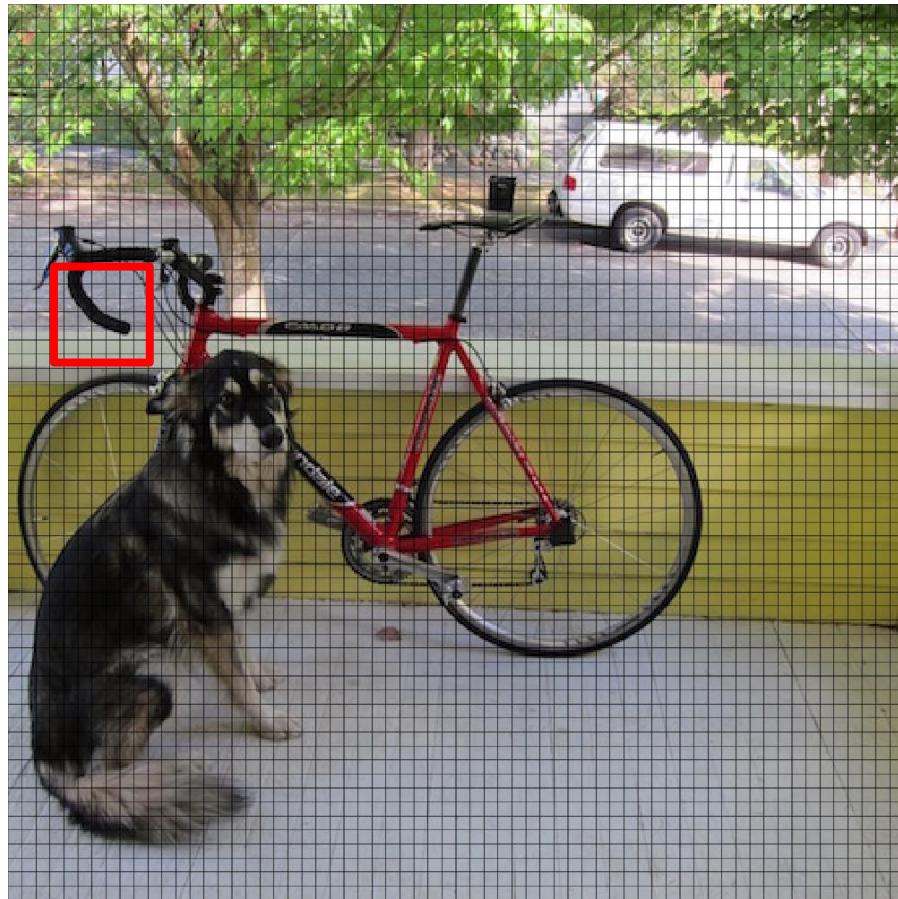
448x448 -> 64x64



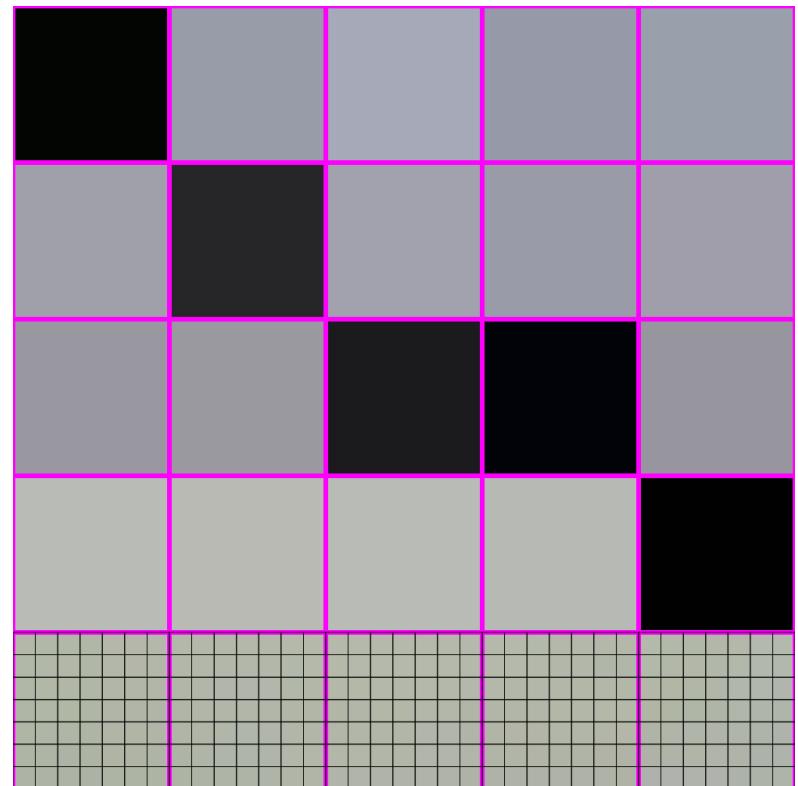
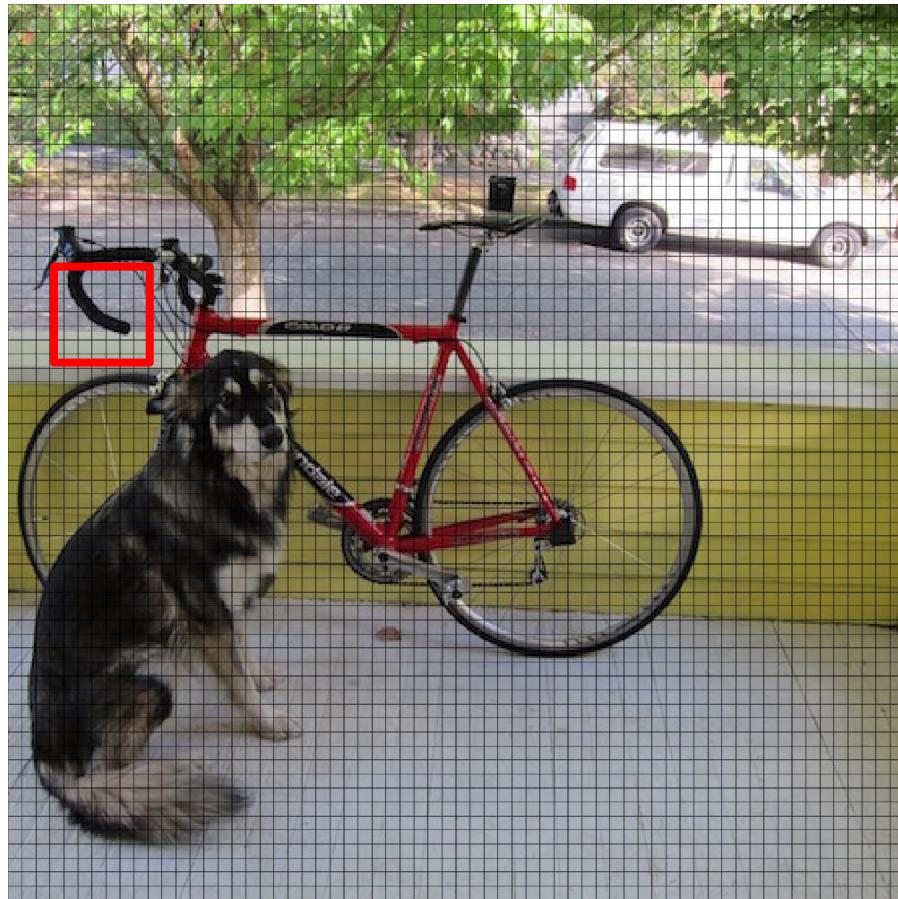
448x448 -> 64x64



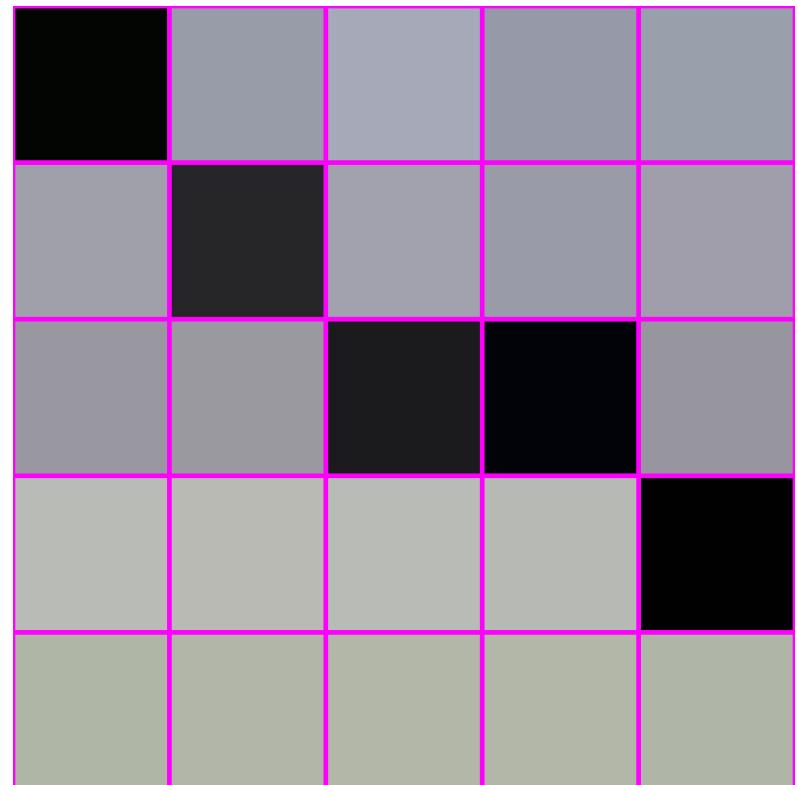
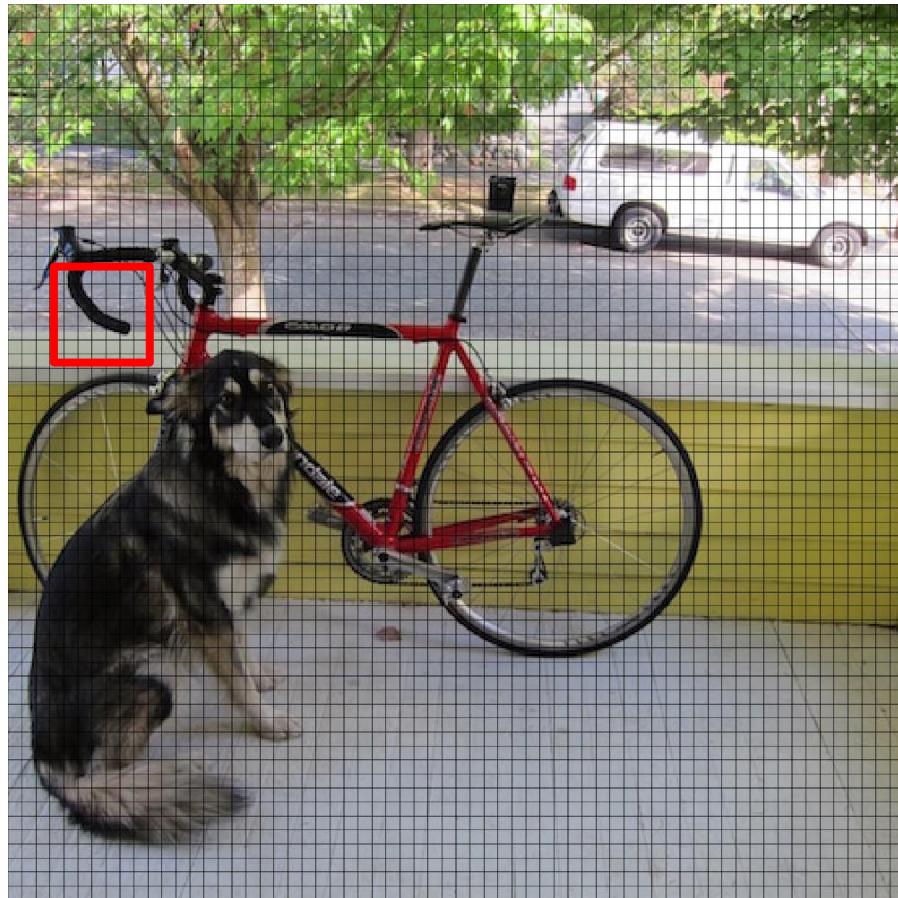
448x448 -> 64x64



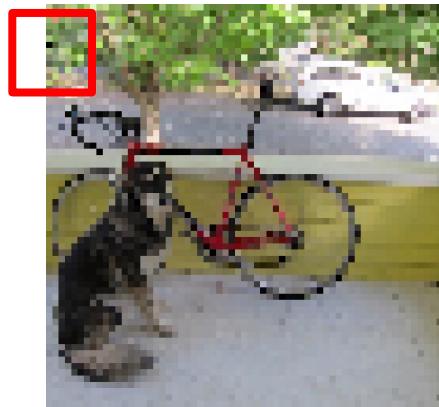
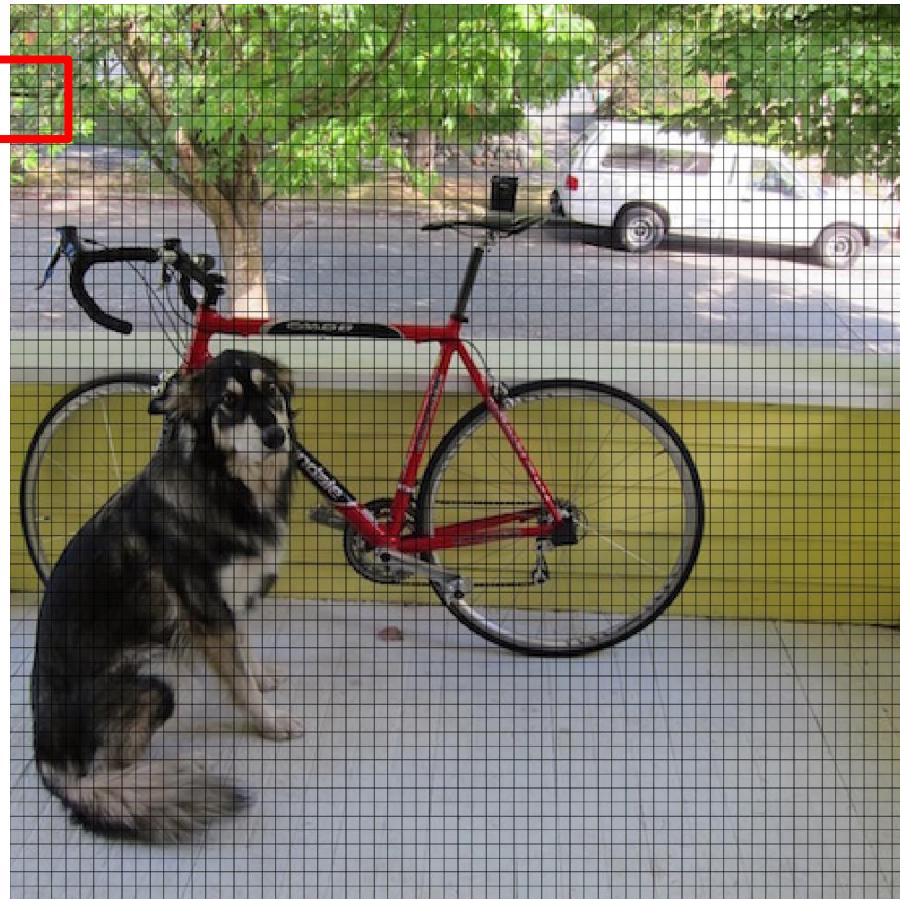
448x448 -> 64x64



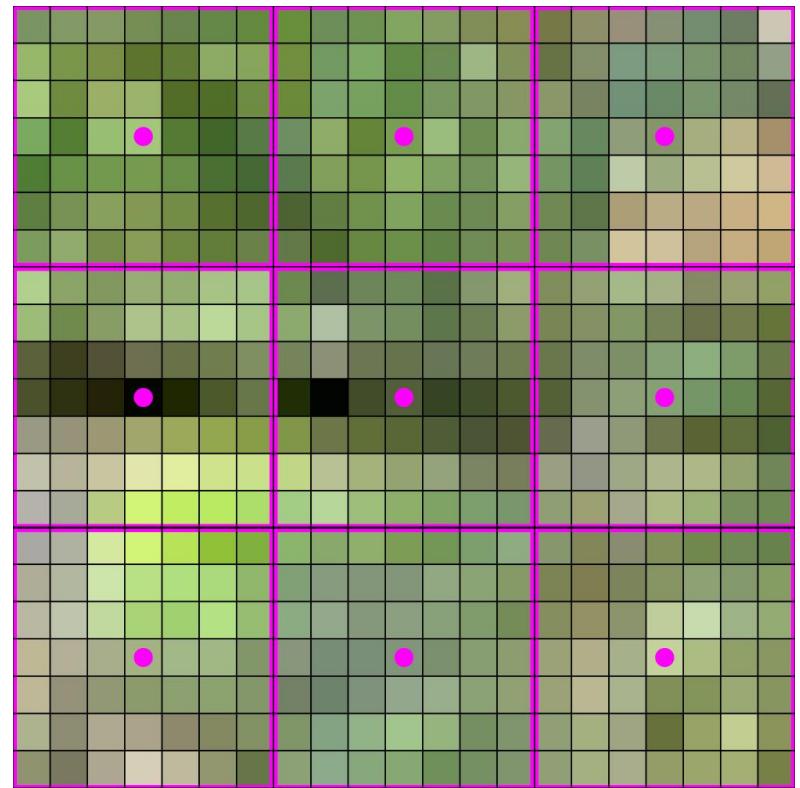
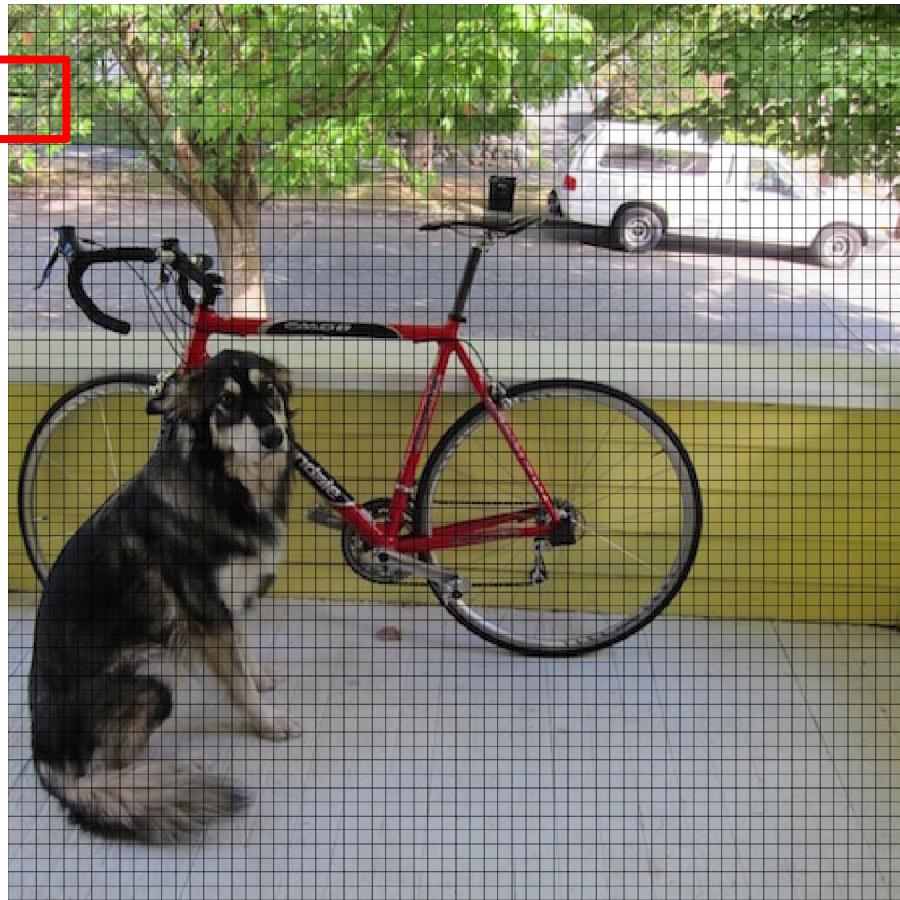
448x448 -> 64x64



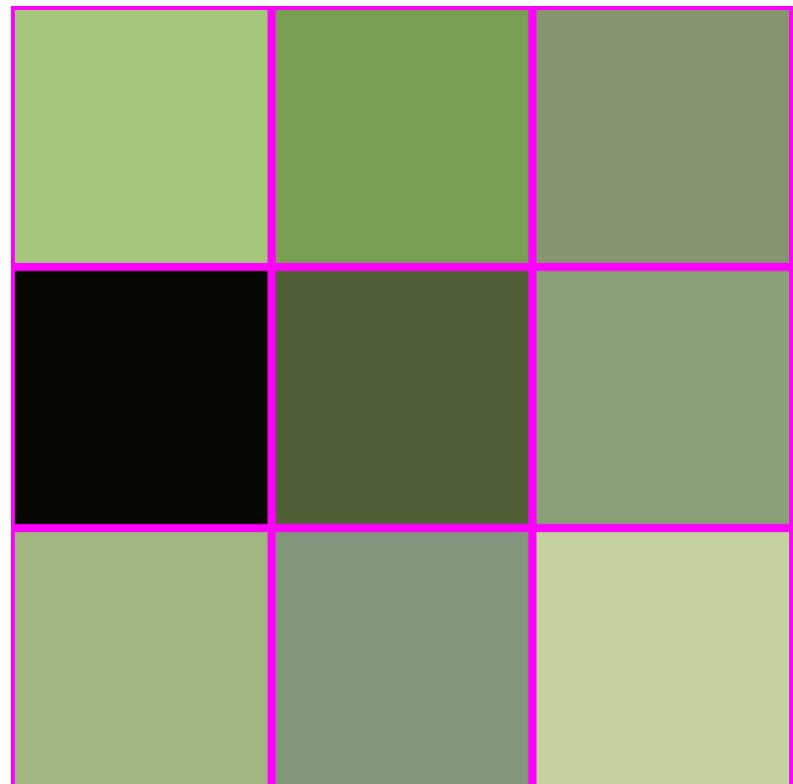
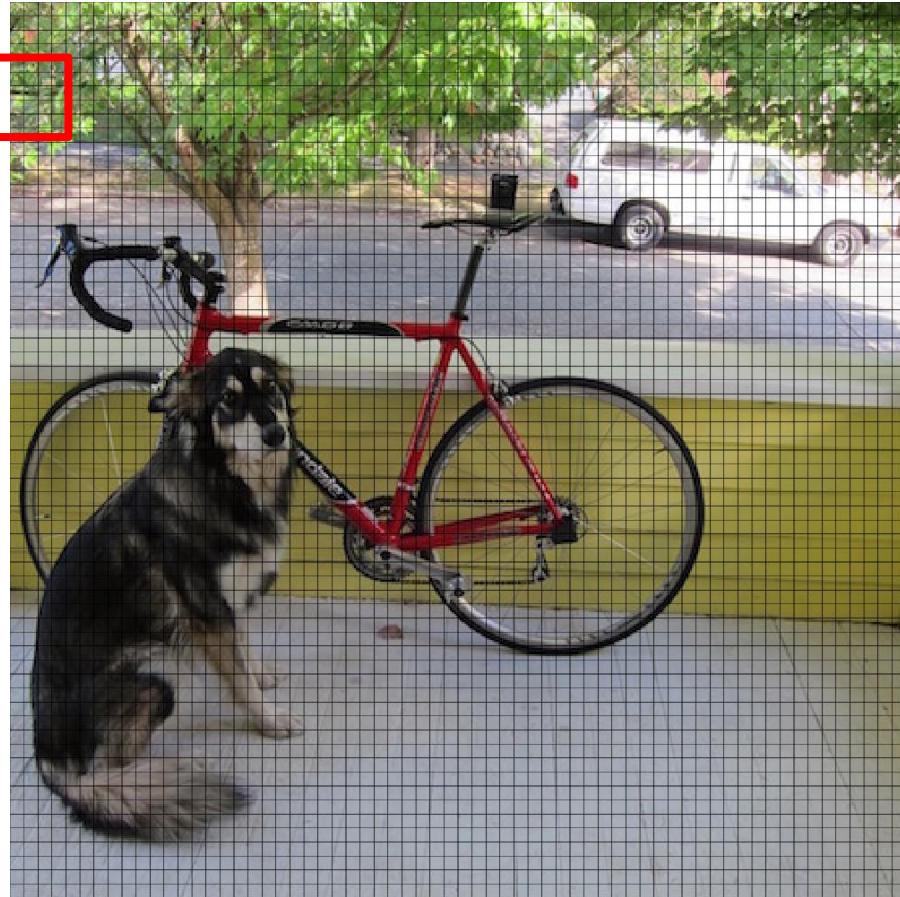
448x448 -> 64x64



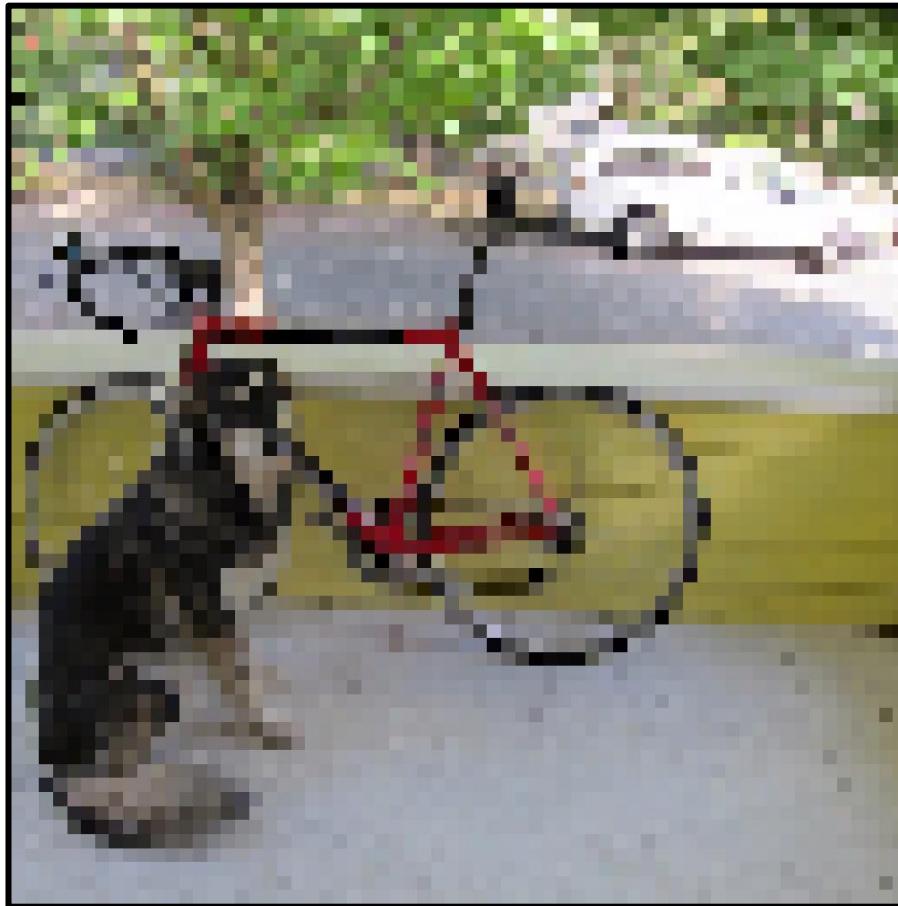
448x448 -> 64x64



448x448 -> 64x64



IS THIS ALL THERE IS??



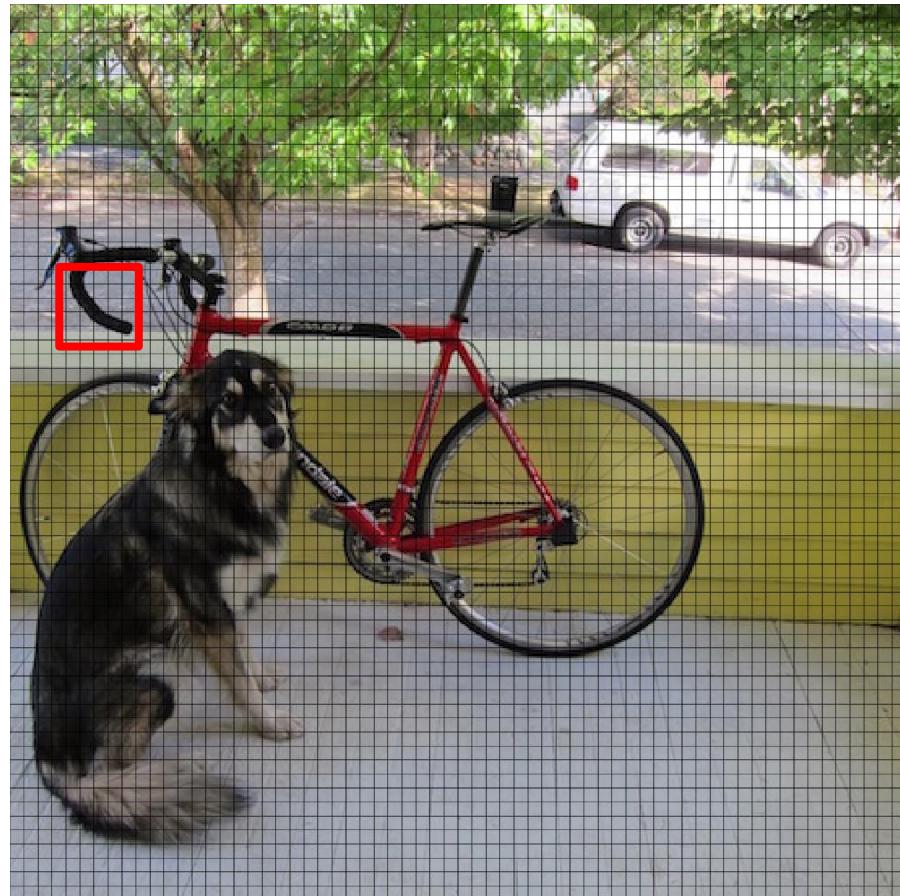
THERE IS A BETTER WAY!



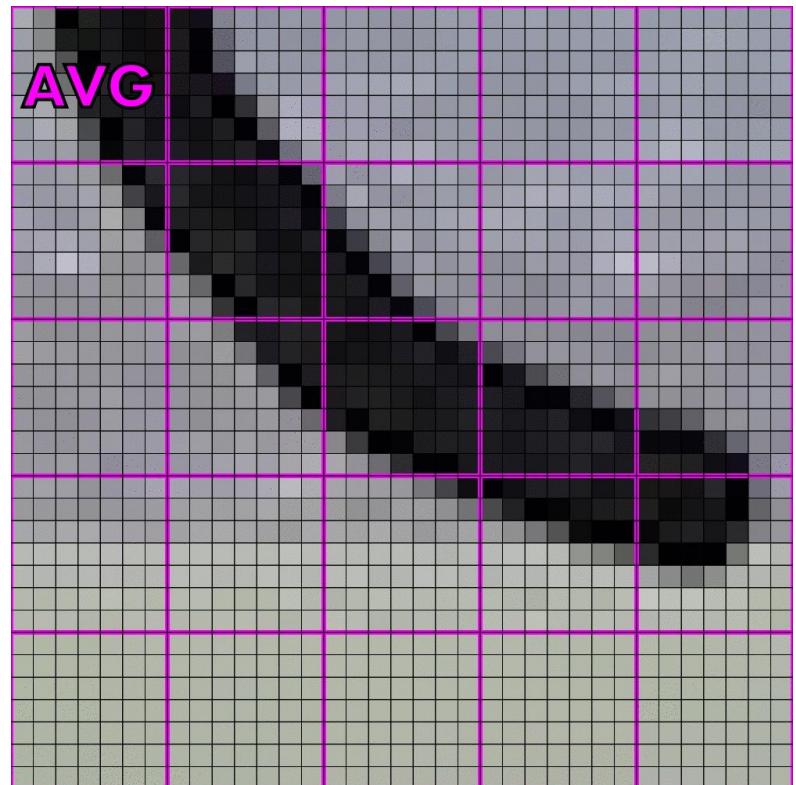
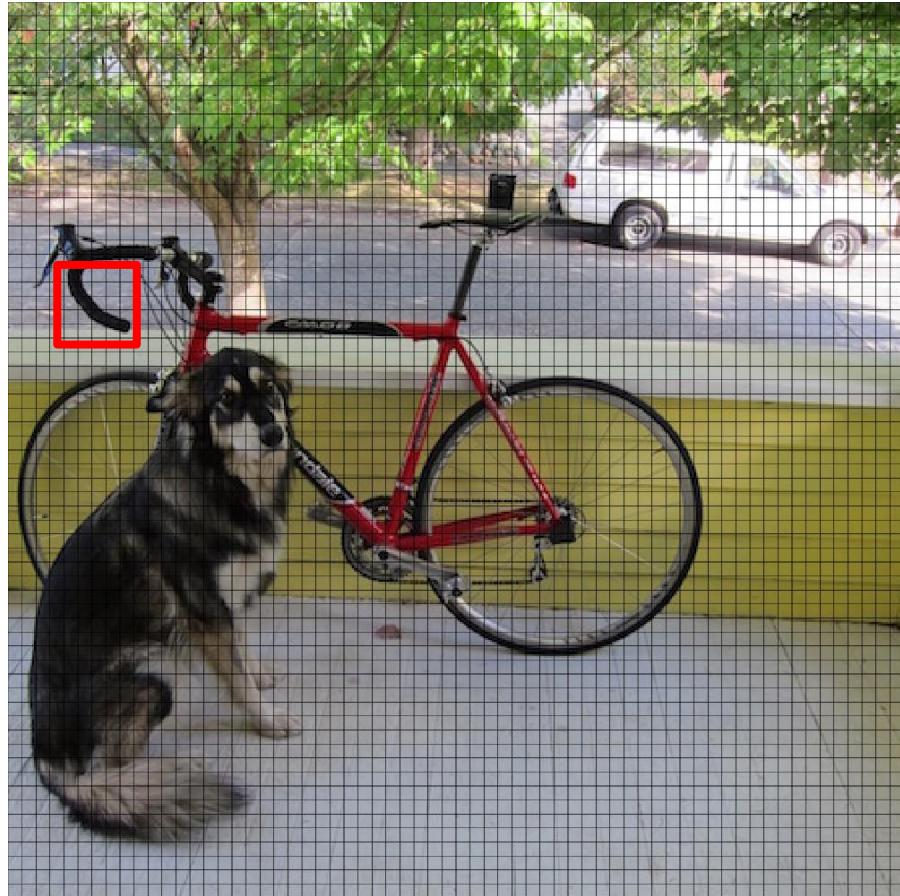
LOOK AT HOW MUCH BETTER



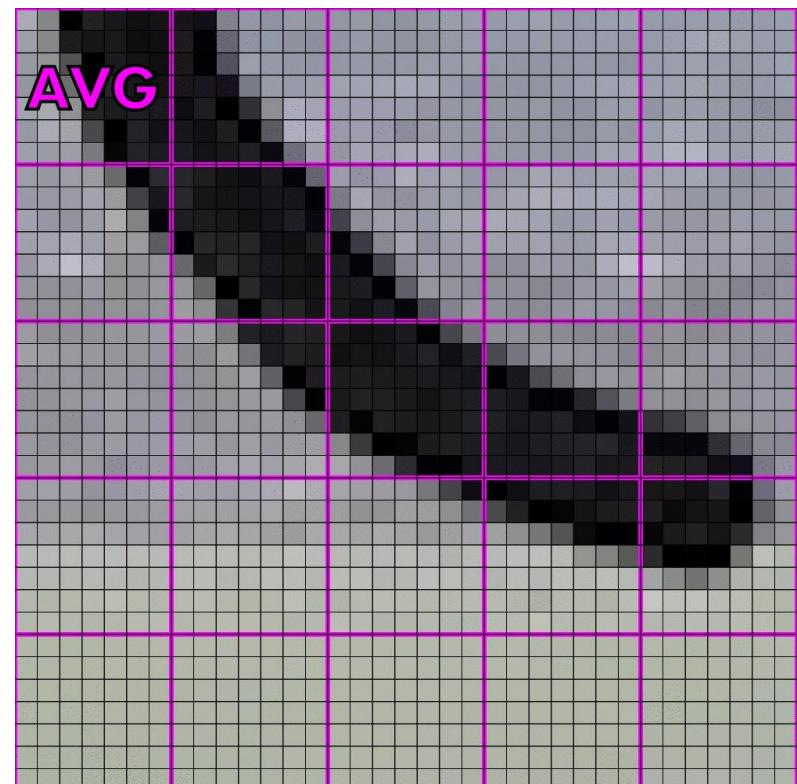
How do?



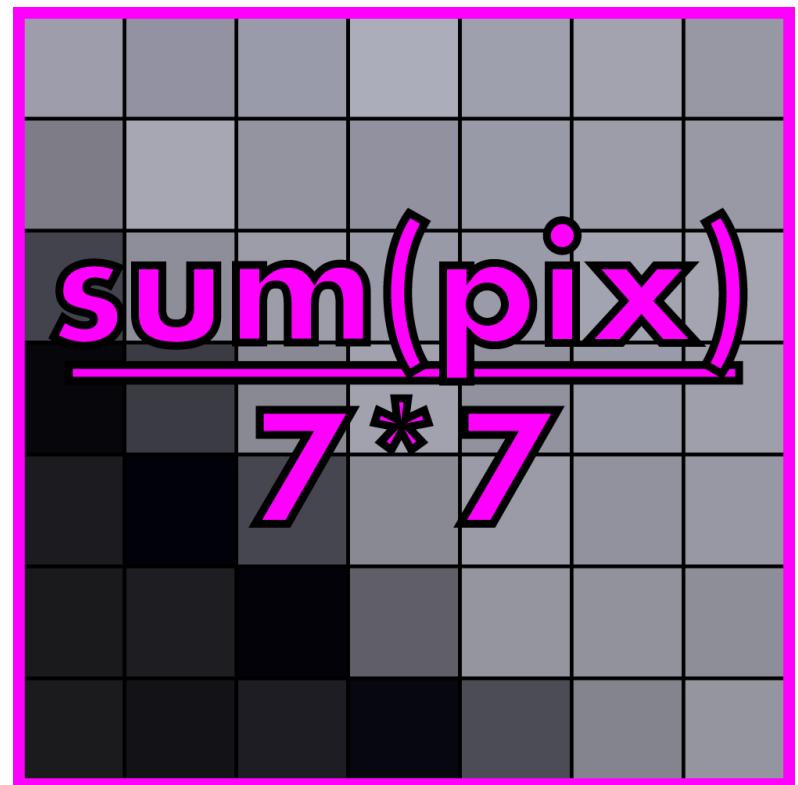
How do? Averaging!



What is averaging?

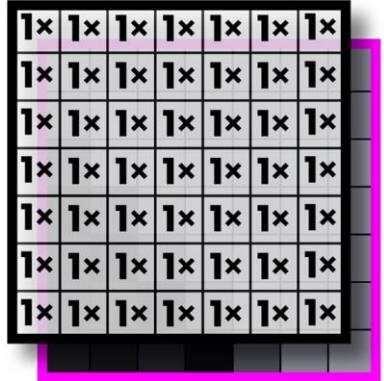


What is averaging? A weighted sum



What is averaging? A weighted sum

sum $\left[\frac{1}{49} \right]$



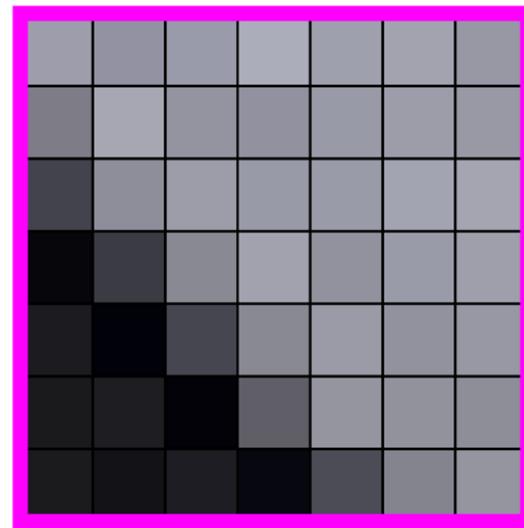
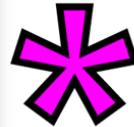
A 7x7 grid of 49 small squares, each containing the text "1x". The grid is enclosed in a pink rectangular frame. To the left of the grid, the word "sum" is written in large, bold, black letters, followed by a pink bracket containing the fraction $\frac{1}{49}$.

Call this operation “convolution”

Filter or kernel

$\frac{1}{49}$

1x						
1x						
1x						
1x						
1x						
1x						
1x						

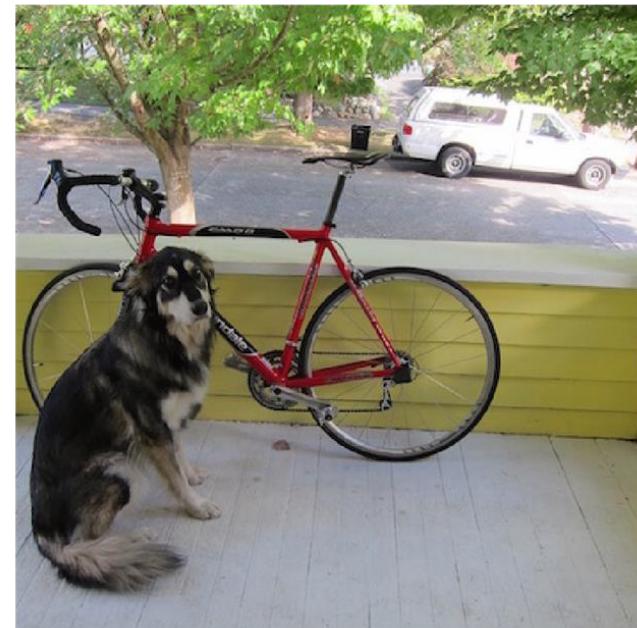
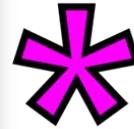


Note: multiplying an image section by a filter is actually called “correlation” and convolution involves inverting the filter first, but since our filters are generally symmetric, we call everything convolution. This is what all computer vision people do.

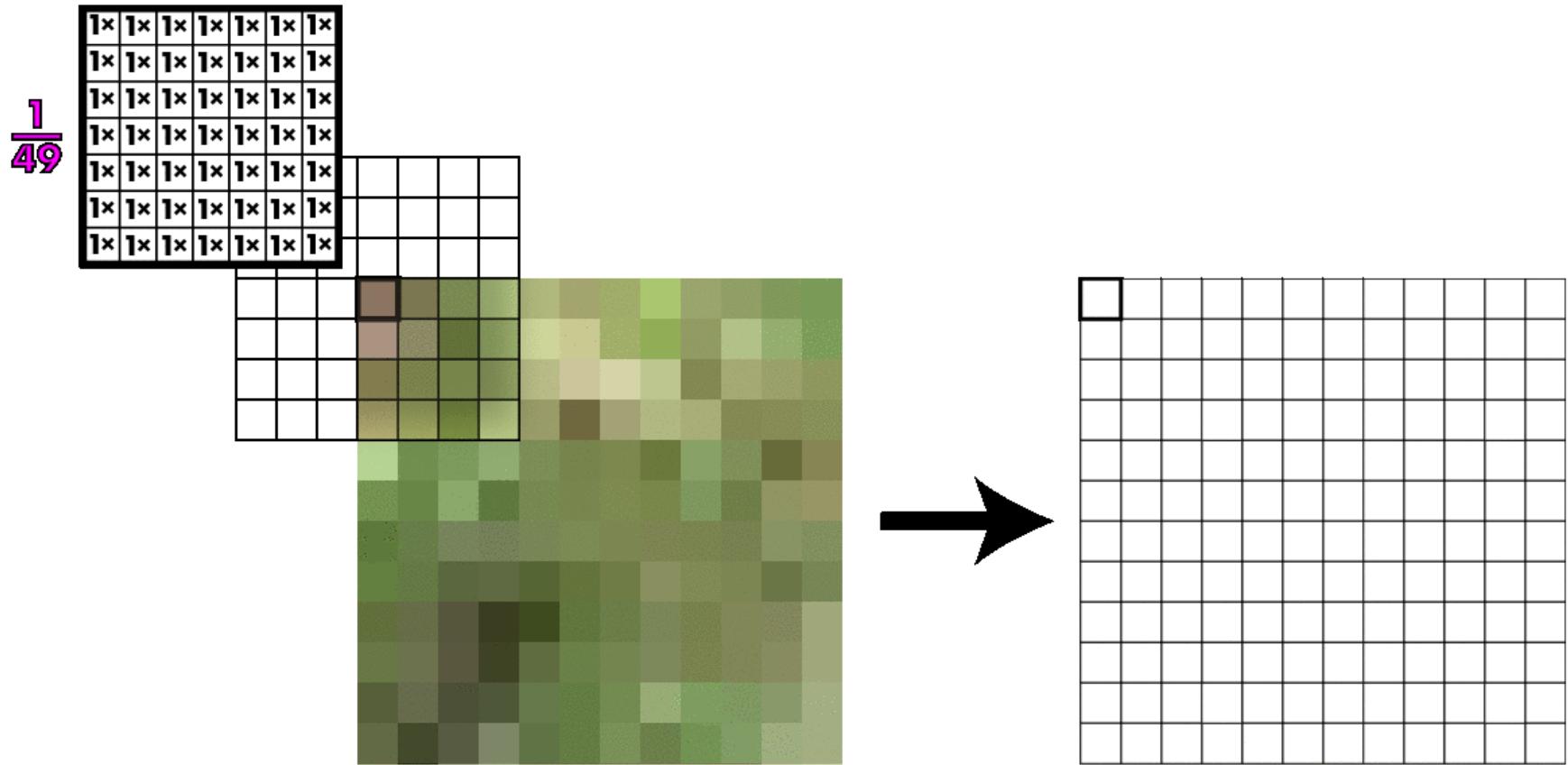
Convolutions on larger images

$\frac{1}{49}$

$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							



Kernel slides across image



Kernel slides across image

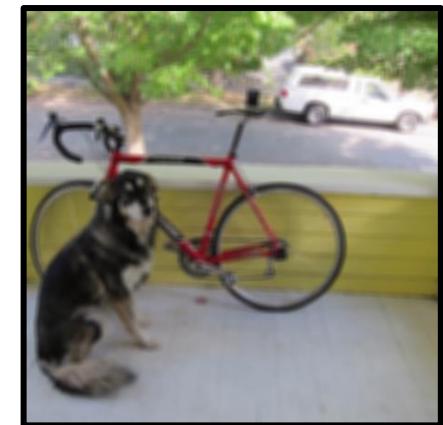
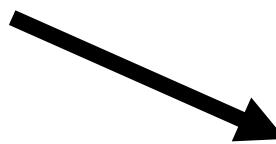
What happens at the edges of the images?

1. Zero padding: easiest but can give bad results
2. Duplicate the outermost row or column
3. Use wraparound

Convolutions on larger images

$\frac{1}{49}$

1x							
1x							
1x							
1x							
1x							
1x							
1x							



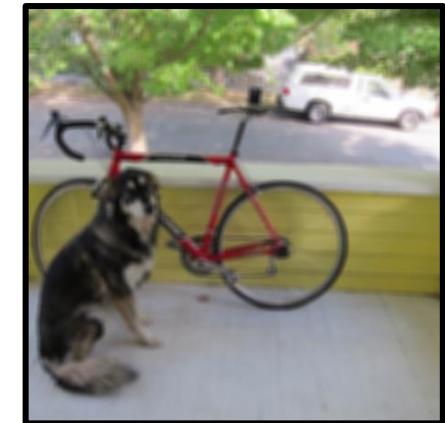
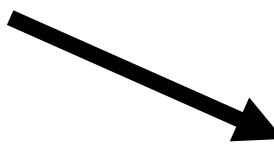
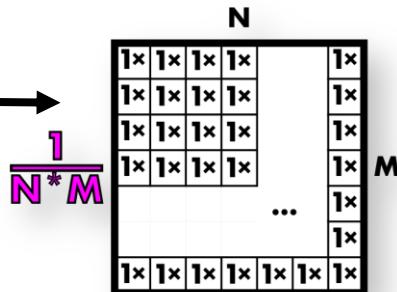
This is called box filter

$\frac{1}{49}$

1x							
1x							
1x							
1x							
1x							
1x							
1x							



Box filters



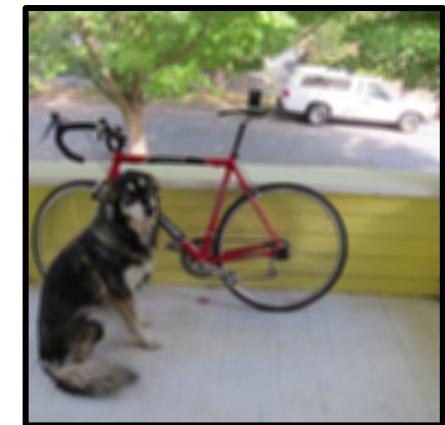
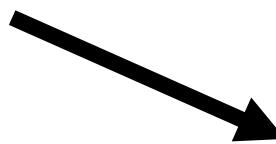
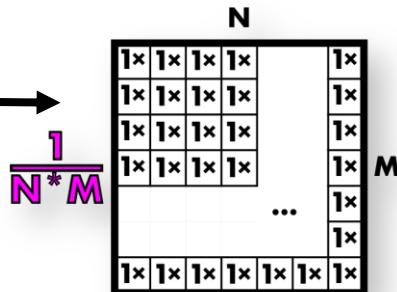
Box filters smooth image

$\frac{1}{49}$

1x							
1x							
1x							
1x							
1x							
1x							
1x							



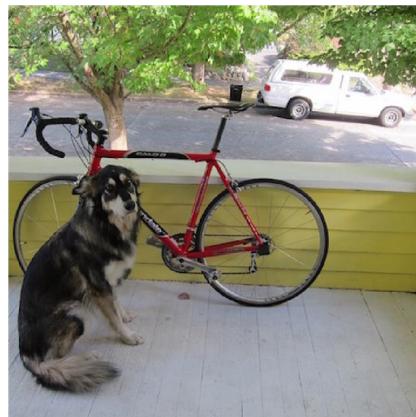
Box filters



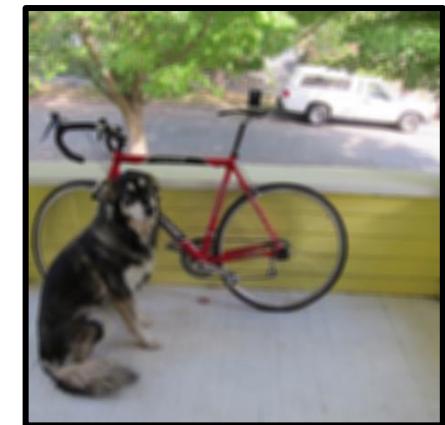
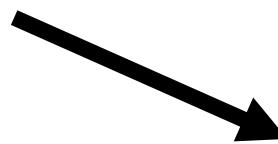
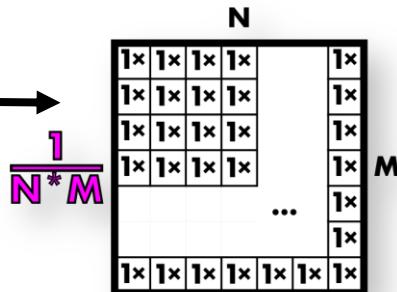
Box filters smooth image

$\frac{1}{49}$

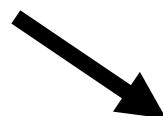
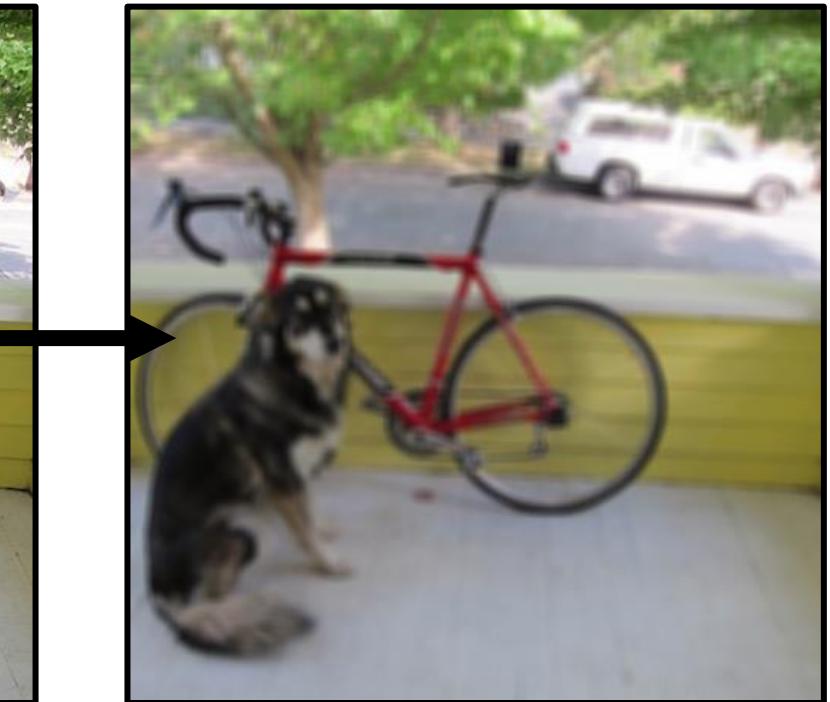
1x							
1x							
1x							
1x							
1x							
1x							
1x							



Box filters



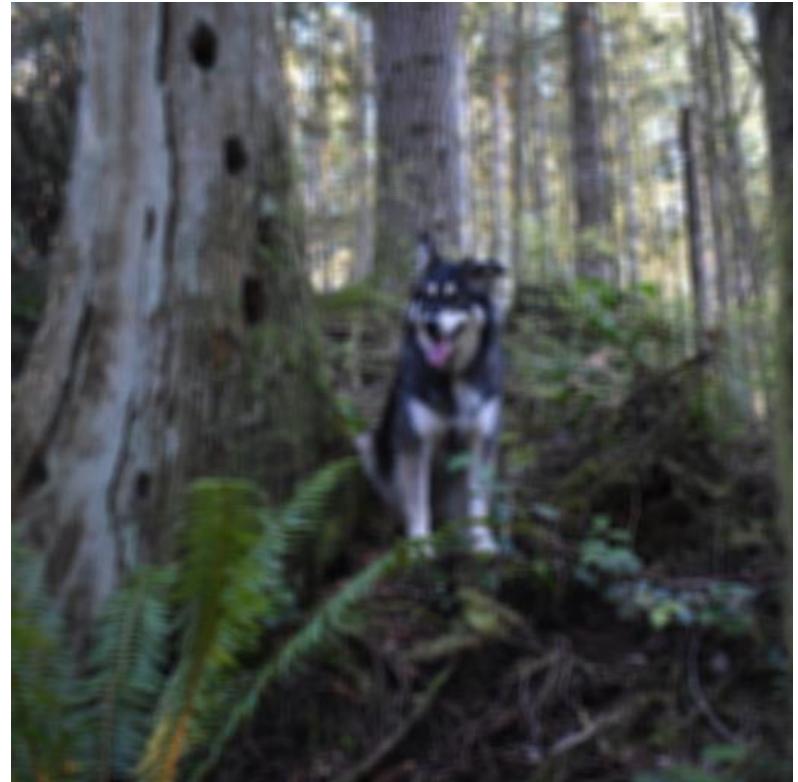
Now we resize our smoothed image



So much better!



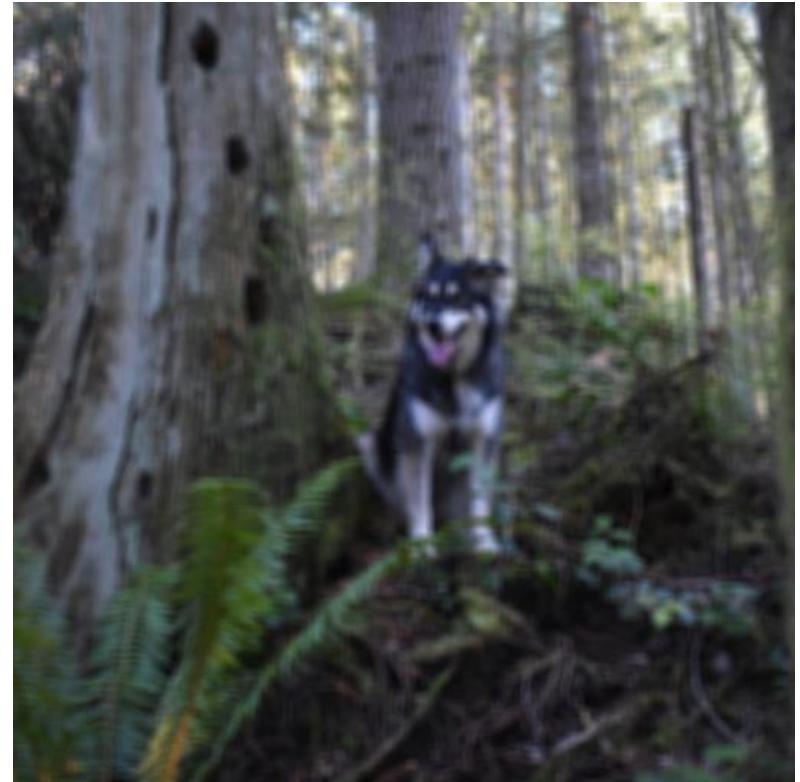
Box filters have artifacts



Box filters have artifacts

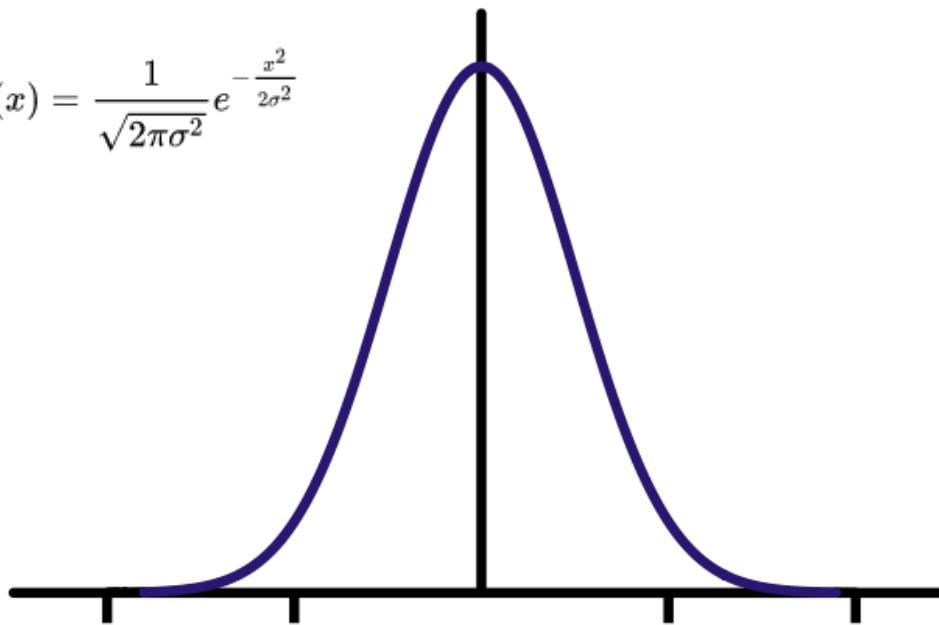


We want a smoothly weighted kernel



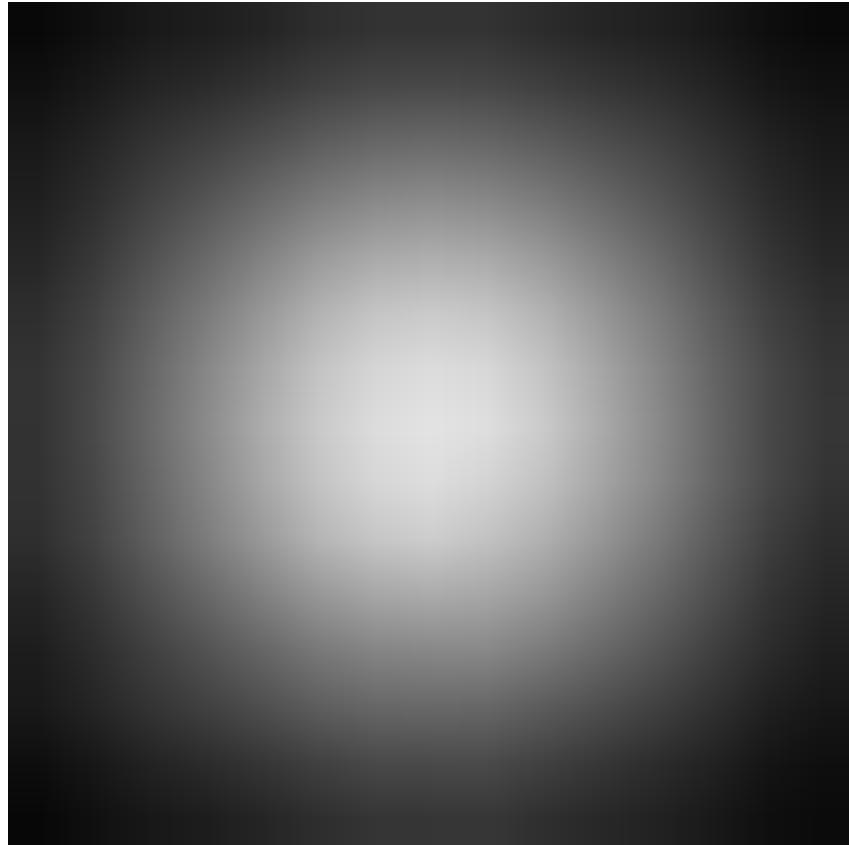
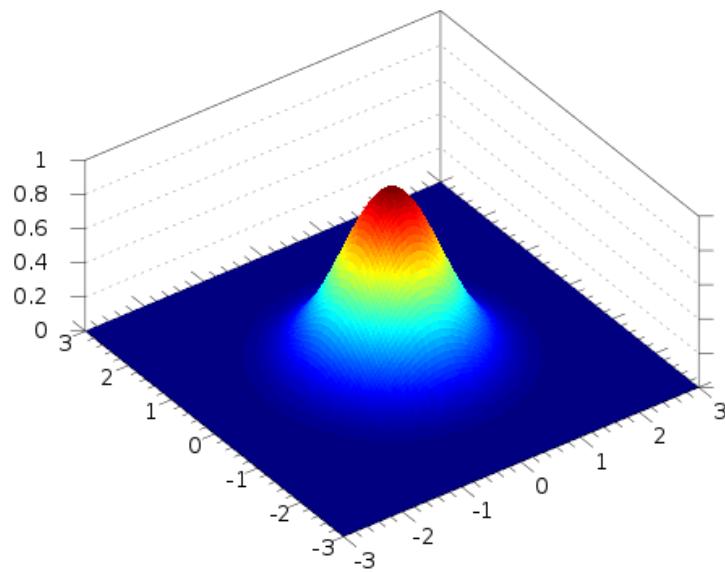
Gaussians

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

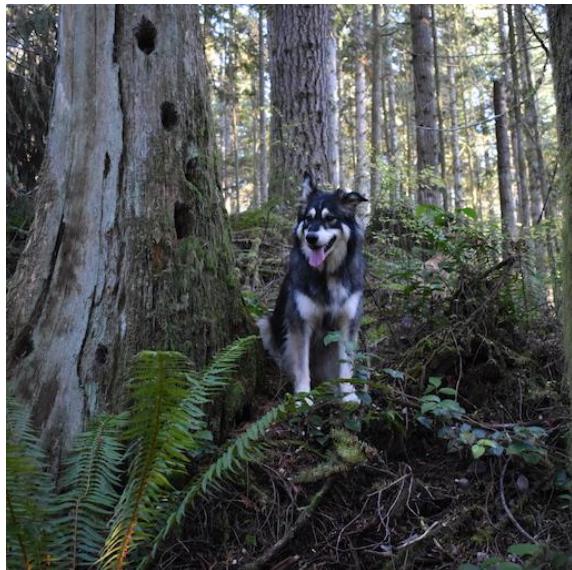
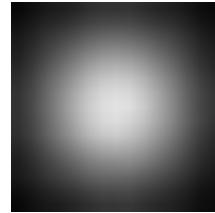
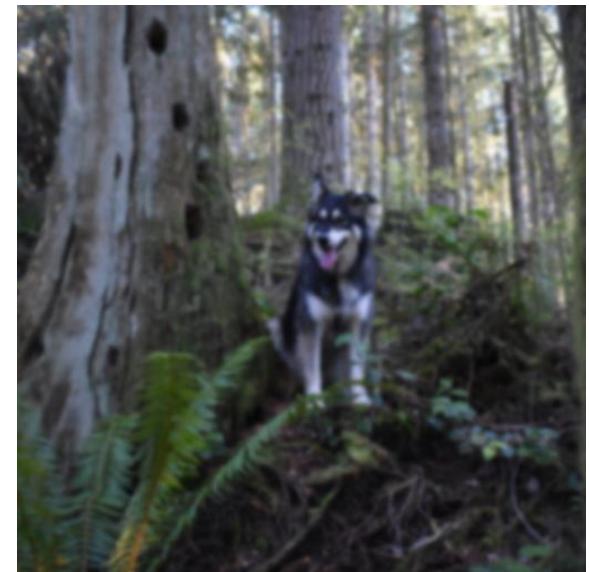


2d Gaussian

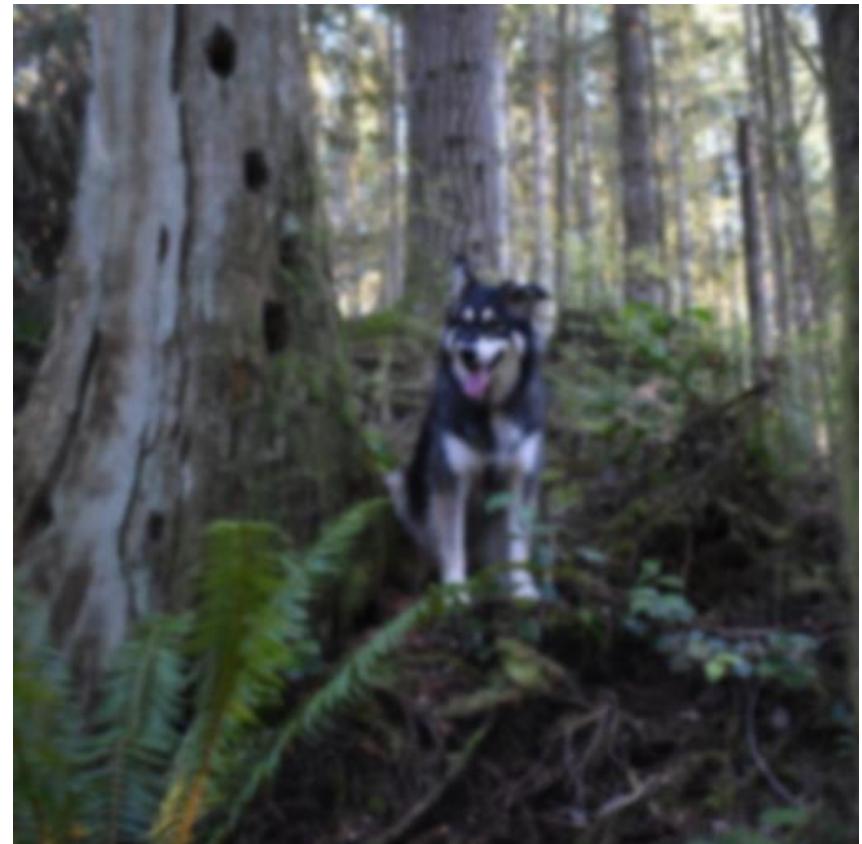
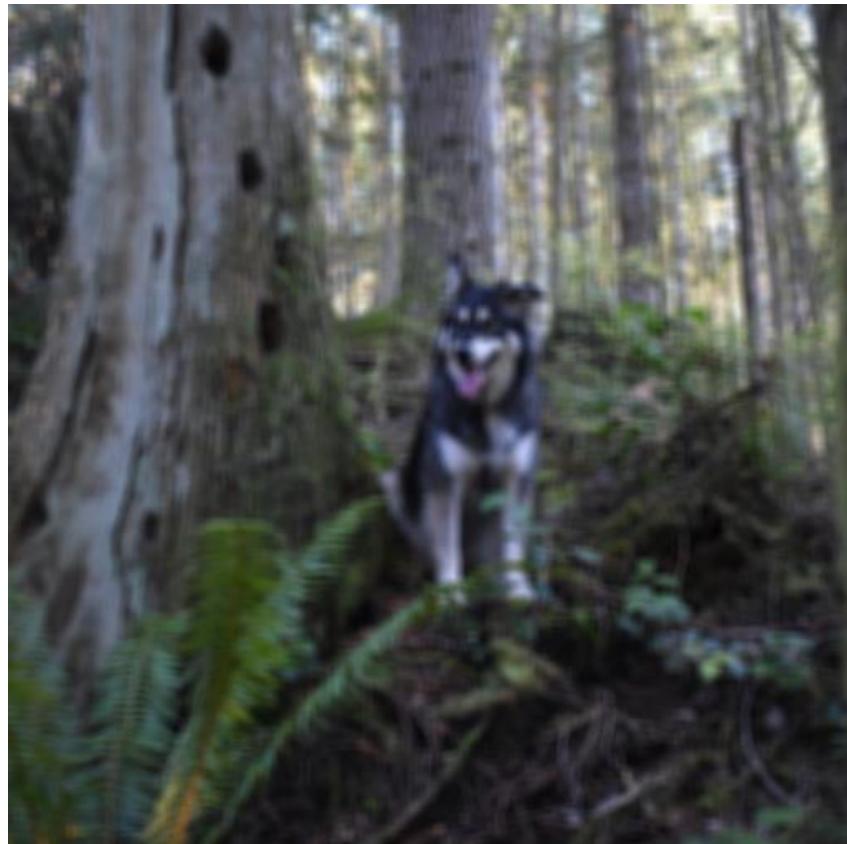
$$g(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$



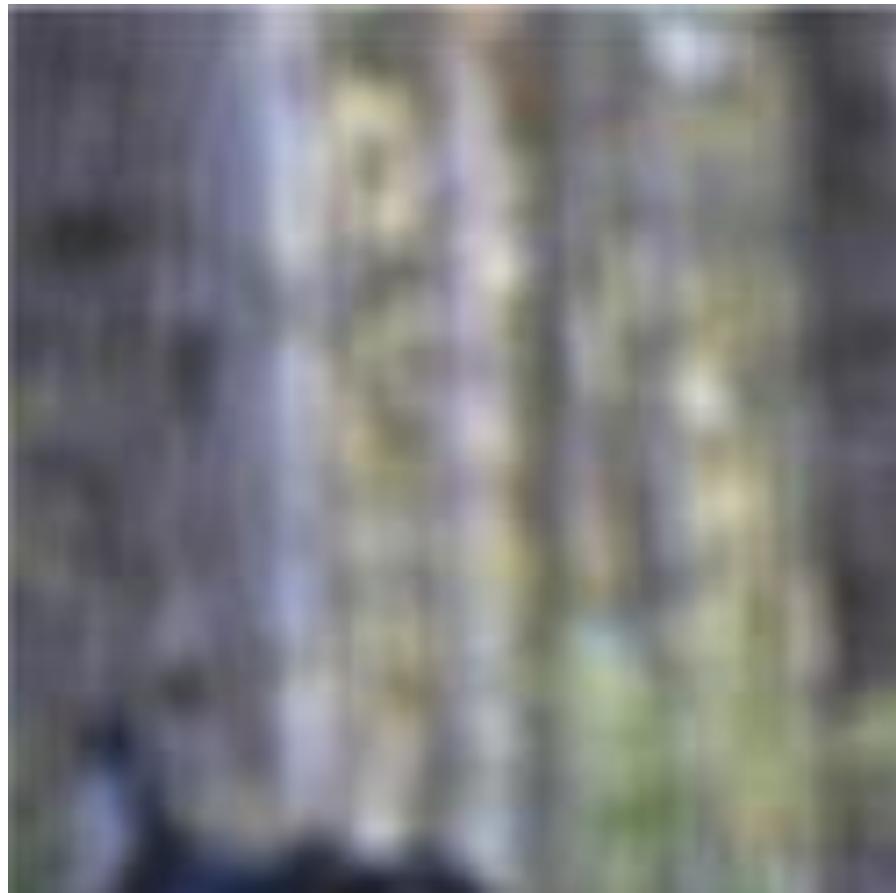
Better smoothing with Gaussians

 $*$  $=$ 

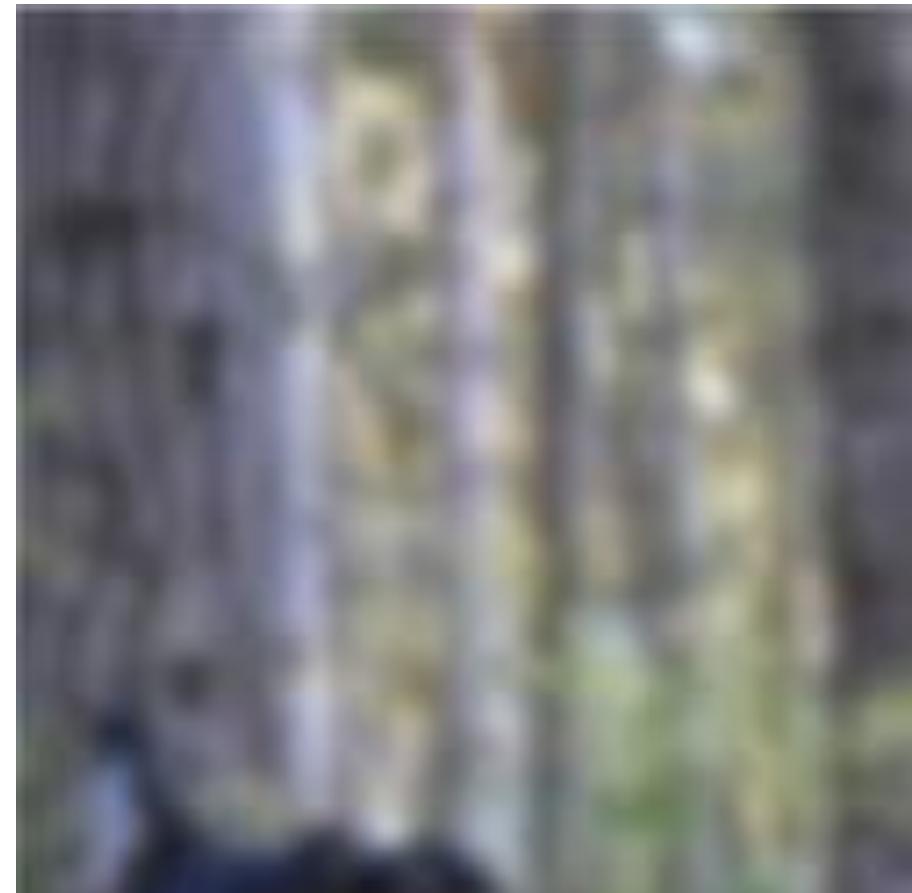
Better smoothing with Gaussians



Better smoothing with Gaussians

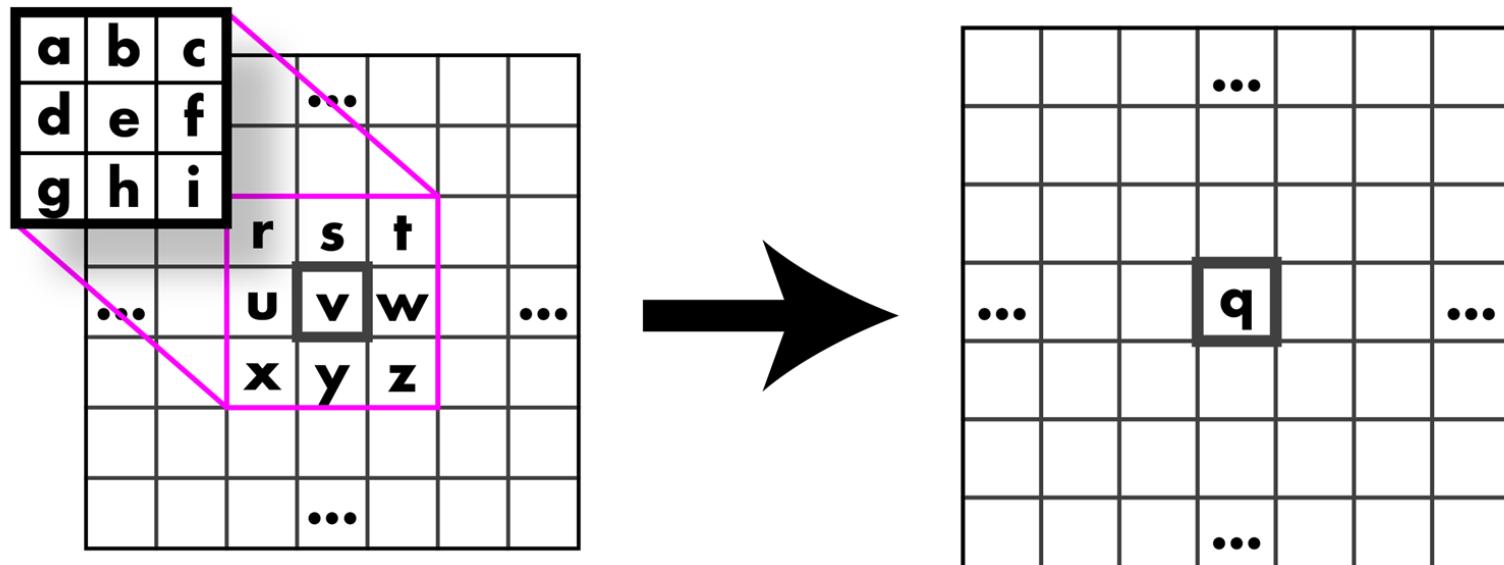


Box Filtered



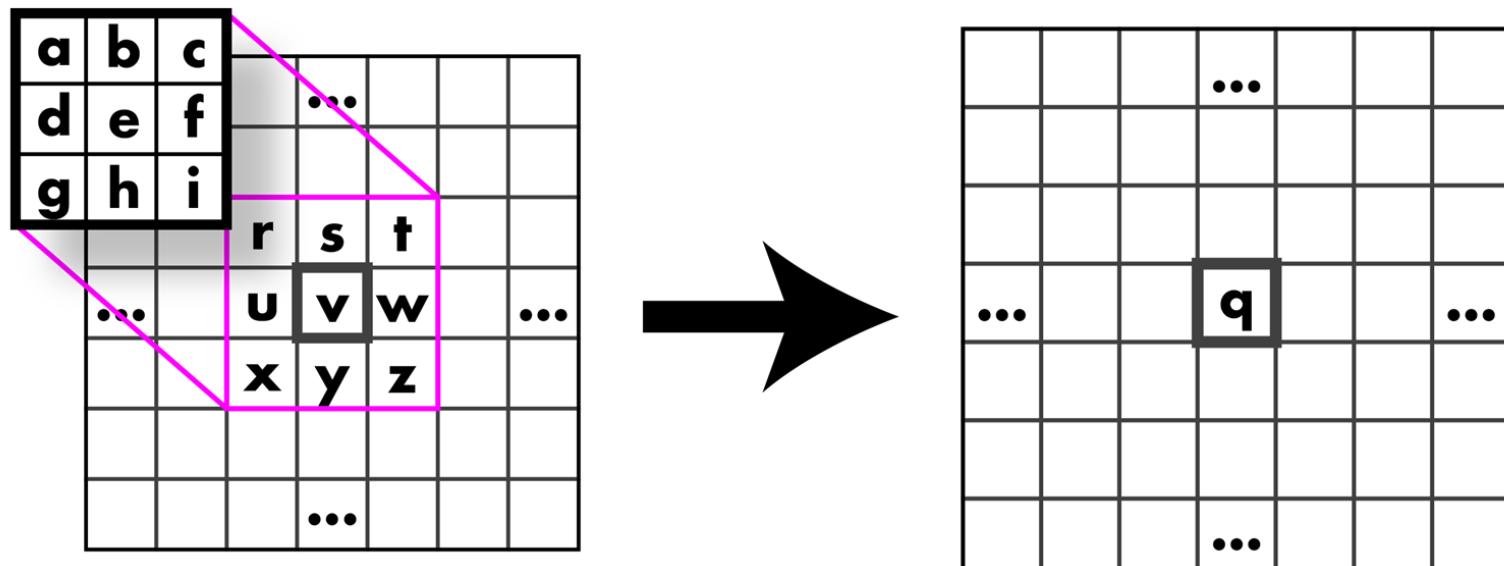
Gaussian Filtered

Wow, so what was that convolution thing??



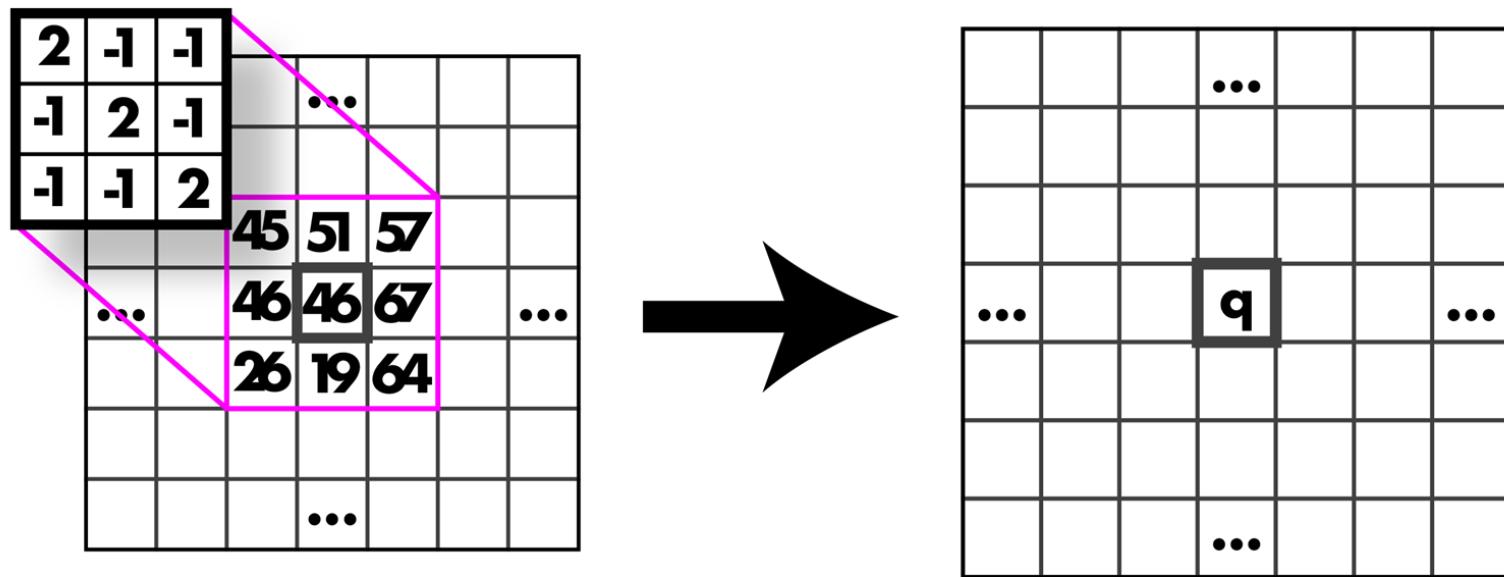
$$q = a \times r + b \times s + c \times t + d \times u + e \times v + f \times w + g \times x + h \times y + i \times z$$

Wow, so what was that convolution thing??

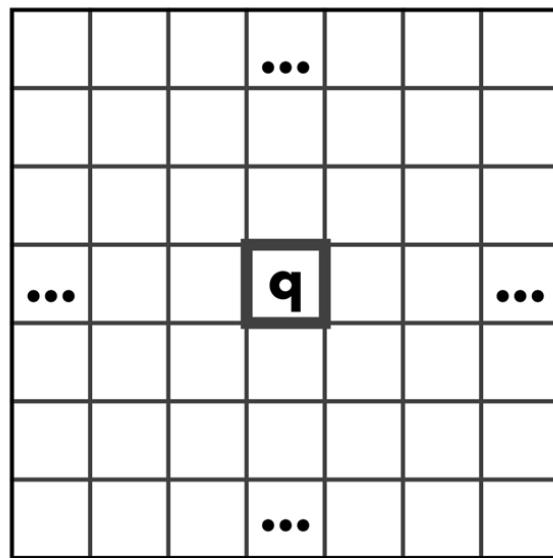
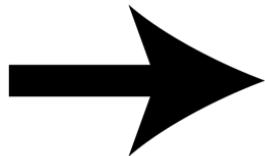
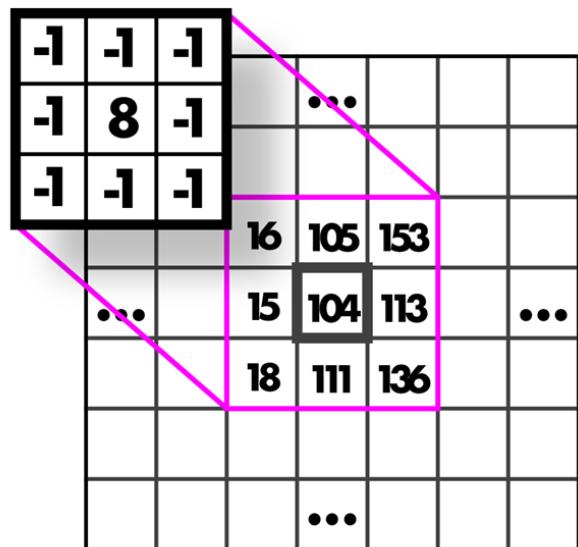


$$q = a \times r + b \times s + c \times t + d \times u + e \times v + f \times w + g \times x + h \times y + i \times z$$

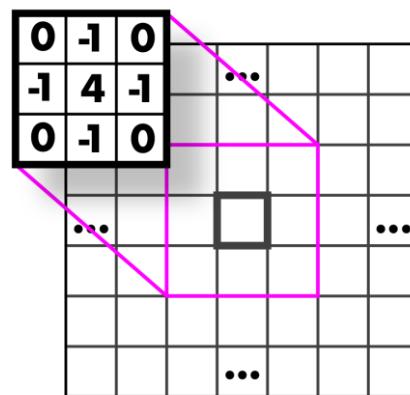
Calculate it, go!



Calculate it, go!

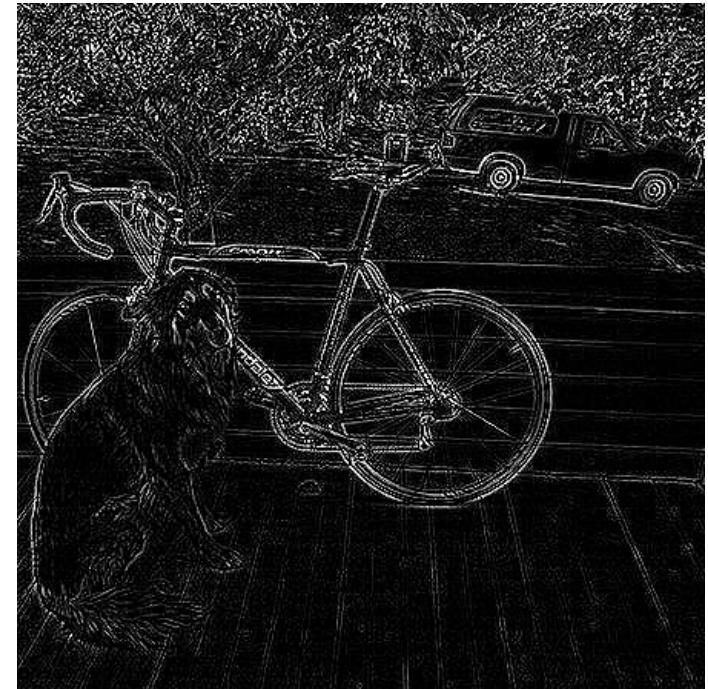
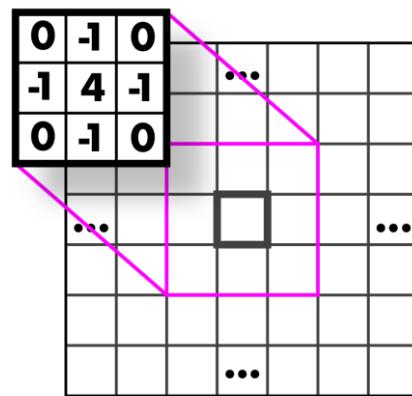


Guess that kernel!

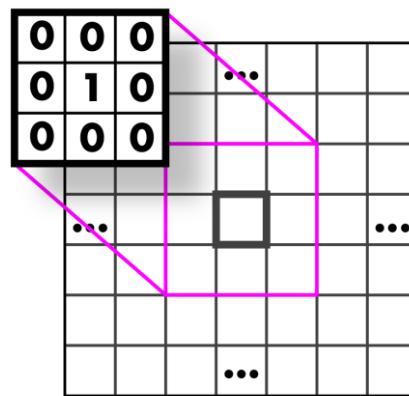


Highpass Kernel: finds edges

(applied to the graytone image!)



Guess that kernel!



Identity Kernel: Does nothing!

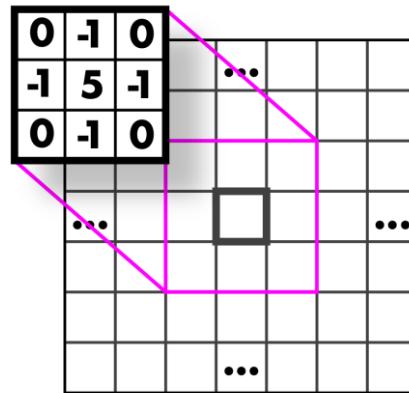


$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

A diagram illustrating a convolution operation with a 3x3 identity kernel. To the left is a 3x3 input matrix. To the right is a 5x5 output grid. A pink cross highlights the central element of the kernel as it slides across the input. The output value at the center of the kernel's receptive field is highlighted with a black square.

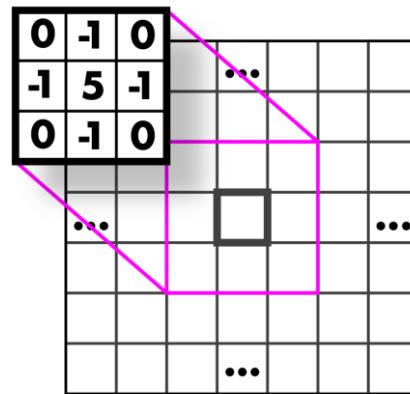


Guess that kernel!



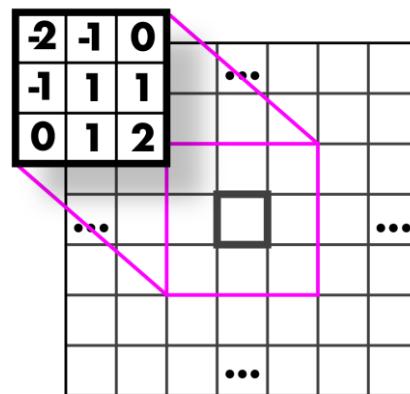
Sharpen Kernel: sharpens!

(applied to all three bands)

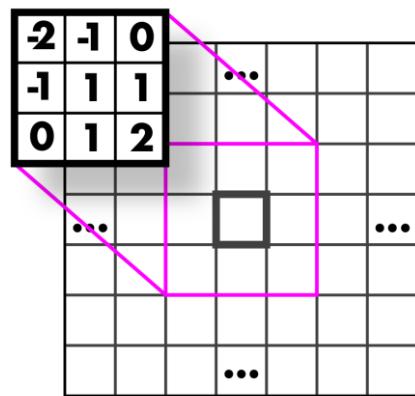


Note: sharpen = highpass + identity!

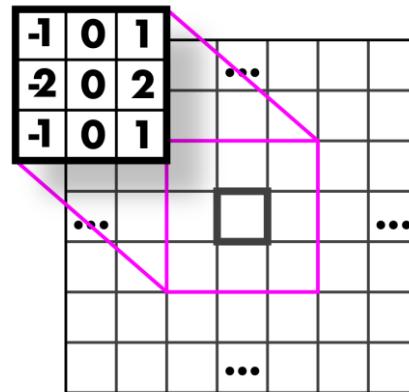
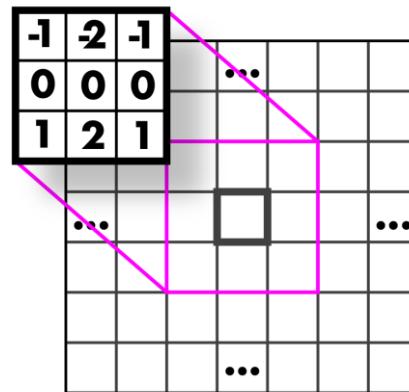
Guess that kernel!



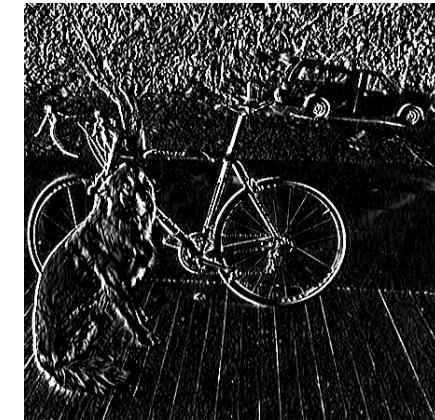
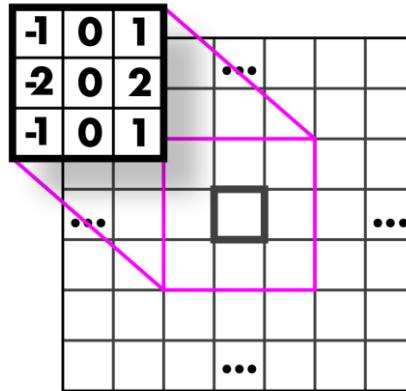
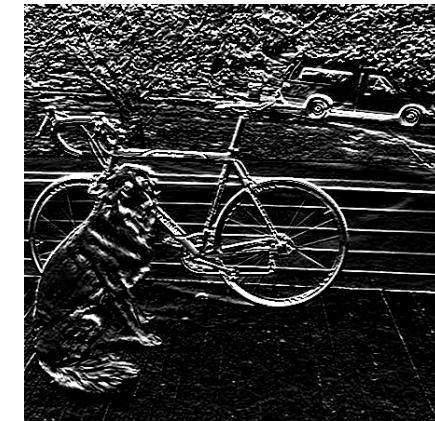
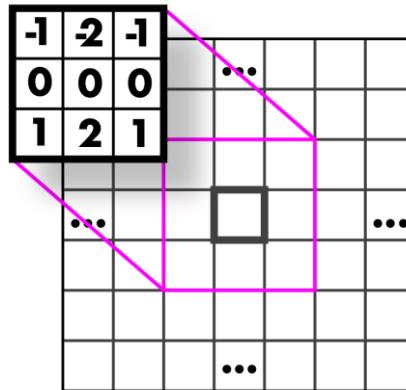
Emboss Kernel: stylin' (applied to all three bands)



Guess those kernels!



Sobel Kernels: edges (applied to a graytone image and thresholded)



Sobel Kernels: edges and gradient!



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

A diagram illustrating the application of a Sobel kernel to a 3x3 grid. The kernel is represented by the matrix:

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

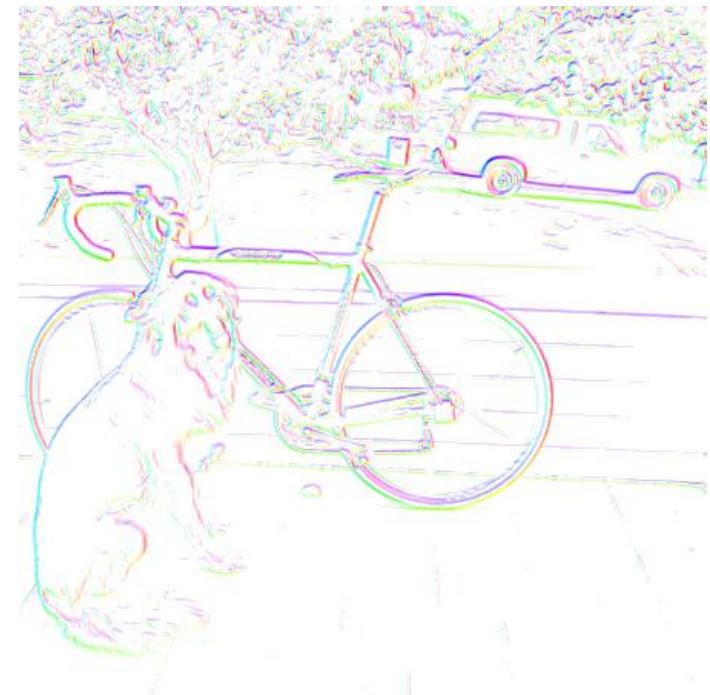
The grid has a central square highlighted in black. The result of the convolution is shown in the bottom-right square of the grid.

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

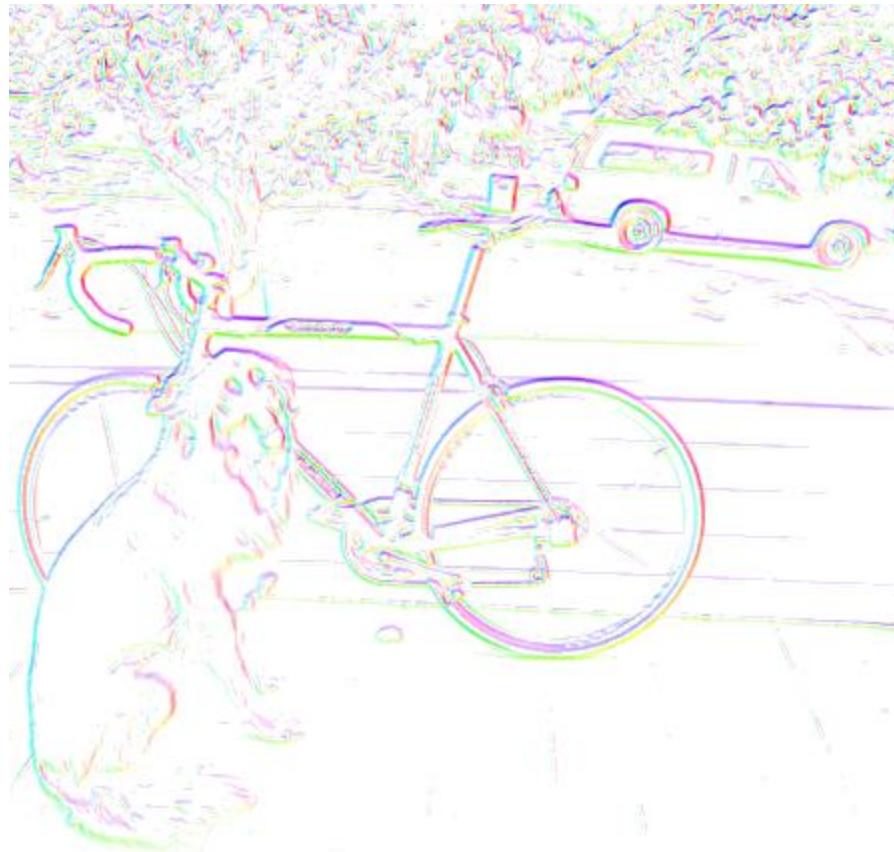
A diagram illustrating the application of a Sobel kernel to a 3x3 grid. The kernel is represented by the matrix:

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

The grid has a central square highlighted in black. The result of the convolution is shown in the bottom-right square of the grid.



Sobel Kernels: edges and gradient!



This visualization is showing the magnitude and direction of the gradient. We will talk further about this when we discuss edges.