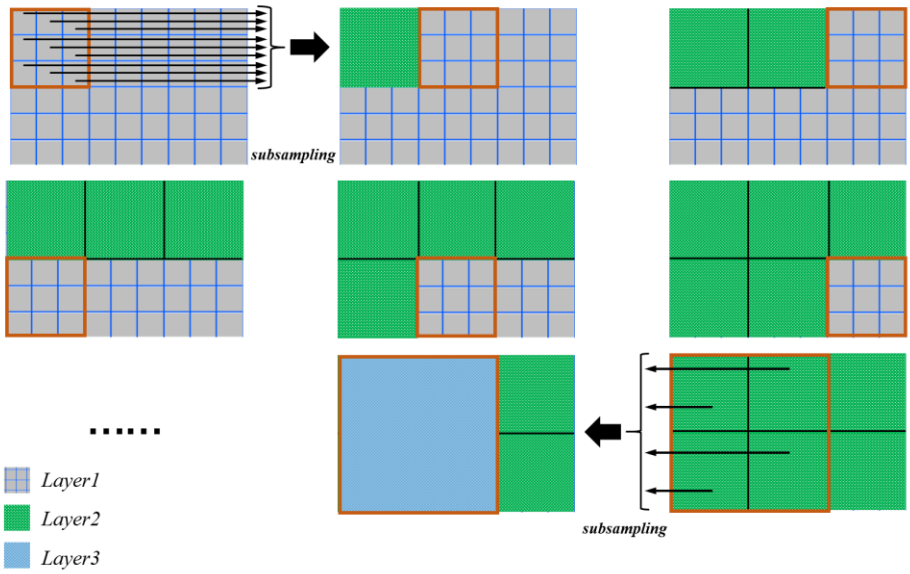
SUBSAMPLING:

相当于缩小图片，即缩小图片尺寸。当图片缩小时，图片区域内的像素数量也会减少。这时，将把原图像中一个区域内的若干个像素点的信息取平均，得到一个像素点上的信息，然后将此信息赋给图片缩小后对应的像素点即可。



UPSAMPLING:

相当于放大图片，即放大图片尺寸，然后在原有像素中插值，将像素信息存入由于放大而新产生的像素中。插值的方法有很多种，比如：邻插值（新像素点使用最近的旧像素点中的信息）、双线性二次插值（放大之前，4 个旧像素点构成最小的矩形，对于落在矩形内的新像素点，使用 4 个旧像素点的信息做内插）、高阶插值等。

需要注意的是，位图(Google 关键词 raster 和 vector 了解位图和矢量图的区别)中存在灰度值突变的像素，通常存在于图片中的对象的轮廓上，上述的插值方法都不能在灰度突变的像素上插出像素信息连续变化的新像素点，即上述插值方法无法处理边缘。

对边缘的处理需要单独的插值方法，根据处理边缘的时间将插值方法分为两种类型：

1. 放大图片之前处理边缘：先检测图片中的边缘，然后检测边缘上的像素。平坦区域（即颜色变化梯度小的区域）内的像素插值将使用传统的插值方式；对于非平坦区域使用对应设计出的插值方式，从而在创建新像素点时保持原来的边缘细节。
2. 放大图片之后处理边缘：先用传统插值方式放大图片，再检测图像内对象的边缘。对放大后的边缘上的像素使用对应设计出的插值方式，同样可以保持新像素点的边缘细节。

以上的插值方法会对图像区域统一处理。当然可以在图像中的不同区域使用不同的插值方式，只要在插值前先将图片划分为不同的区域，即可在不同的区域上使用不同的插值方式。

BIG O NOTATION

Notation	Name	Example
$O(1)$	constant	Determining if a binary number is even or odd; Calculating $(-1)^n$; Using a constant-size lookup table
$O(\log \log n)$	double logarithmic	Number of comparisons spent finding an item using interpolation search in a sorted array of uniformly distributed values
$O(\log n)$	logarithmic	Finding an item in a sorted array with a binary search or a balanced search tree as well as all operations in a Binomial heap
$O((\log n)^c)$ $c>1$	polylogarithmic	Matrix chain ordering can be solved in polylogarithmic time on a parallel random-access machine.
$O(n^c)$ $0<c<1$	fractional power	Searching in a k-d tree
$O(n)$	linear	Finding an item in an unsorted list or in an unsorted array; adding two n-bit integers by ripple carry
$O(n \log^* n)$	n log-star n	Performing triangulation of a simple polygon using Seidel's algorithm, or the union-find algorithm. Note that $\log^*(n) = \begin{cases} 0, & \text{if } n \leq 1 \\ 1 + \log^*(\log n), & \text{if } n > 1 \end{cases}$
$O(n \log n) = O(\log n!)$	linearithmic, loglinear, or quasilinear	Performing a fast Fourier transform; Fastest possible comparison sort; heapsort and merge sort
$O(n^2)$	quadratic	Multiplying two n-digit numbers by a simple algorithm; simple sorting algorithms, such as bubble sort, selection sort and insertion sort; (worst case) bound on some usually faster sorting algorithms such as quicksort, Shellsort, and tree sort
$O(n^c)$	polynomial or algebraic	Tree-adjoining grammar parsing; maximum matching for bipartite graphs; finding the determinant with LU decomposition
$L_n[\alpha, c] = e^{(c+o(1))(\ln n)^\alpha (\ln \ln n)^{1-\alpha}}$ $0<\alpha<1$	L-notation or sub-exponential	Factoring a number using the quadratic sieve or number field sieve
$O(c^n)$ $c>1$	exponential	Finding the (exact) solution to the travelling salesman problem using dynamic programming; determining if two logical statements are equivalent using brute-force search
$O(n!)$	factorial	Solving the travelling salesman problem via brute-force search; generating all unrestricted permutations of a poset; finding the determinant with Laplace expansion; enumerating all partitions of a set

参考：https://en.wikipedia.org/wiki/Big_O_notation