Bit Depth in Images

Intensity

Bits

Bytes

Binary

8 bit

16 bit

Byte Swapping

Little Endian

Big Endian

NaN

Computer data is stored as 1's and 0's.

The smallest chunk of information is a **bit**, which can be set to 1 or 0 (+ or -).

So, 1 bit can store either of 2 states (2^1):
"1" or "0".

2 bits can store any one of 4 states (2^2):
00 01 11 10

Image storage size is in bits and bytes.

3 bits can store any one of 8 states (2^3):

000	001	010	100
011	101	110	111

4 bits 24=16

5 bits 25=32

6 bits 26=64

7 bits 27=128

8 bits= 1 **byte** …which can represent any one of 256 values (2^8)

16 bits= 2 **bytes** …which can represent any one of 65,536 values (2^{16})

All else being equal, the more pixels an image has, the more bytes it takes to store it in the computer.

However,

images with the same resolution and file type can differ in storage size based on the number of bits needed to store each pixel.

Why do some pixels require more storage bits than others?

Because image intensity

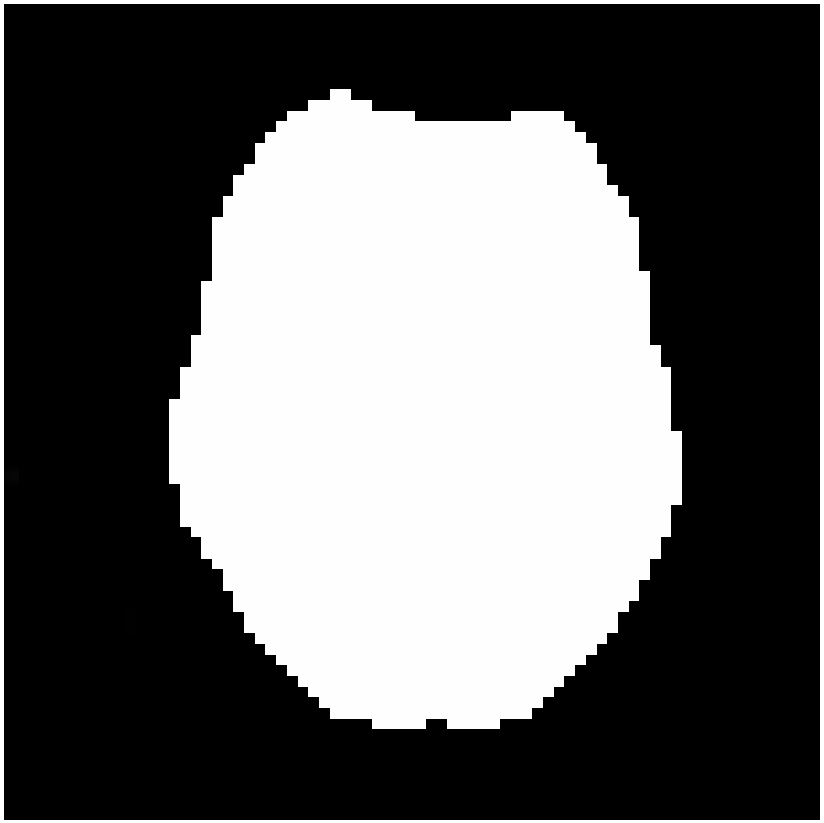
represented as shades of grey (**greyscale**)

can be quantized at different levels,

typically, 1 bit, 8 bits or 16 bits.

White is the most intense (highest value) and black is the least intense (lowest value).

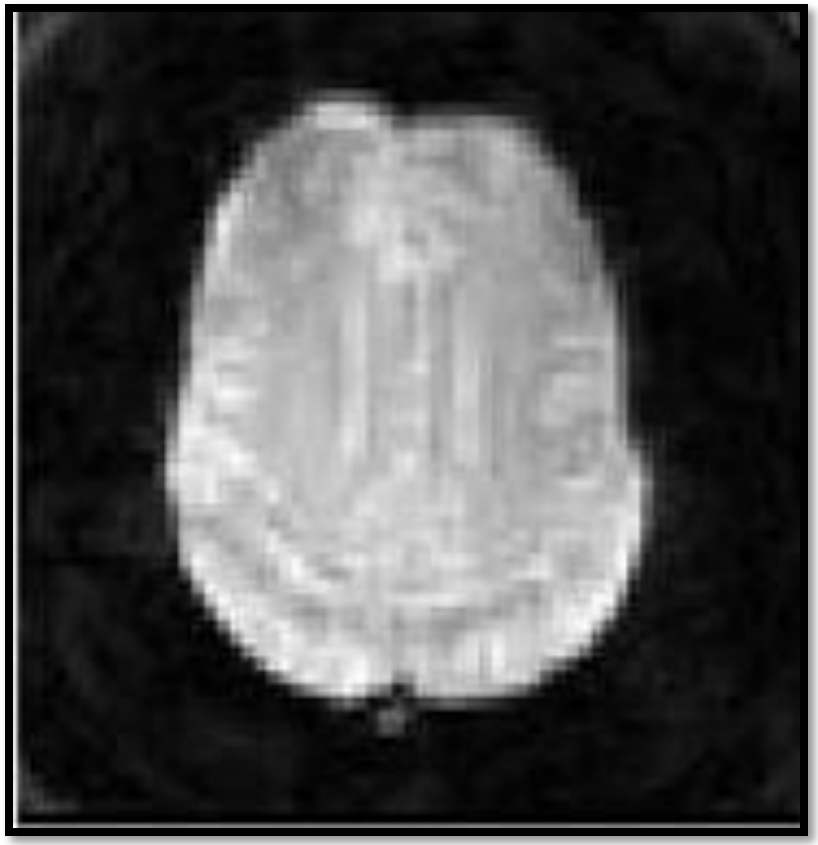
1 bit: Binary



In a binary image, there are only 2 values: 0=Black; 1=White.

So each pixel requires only 1 bit to store.

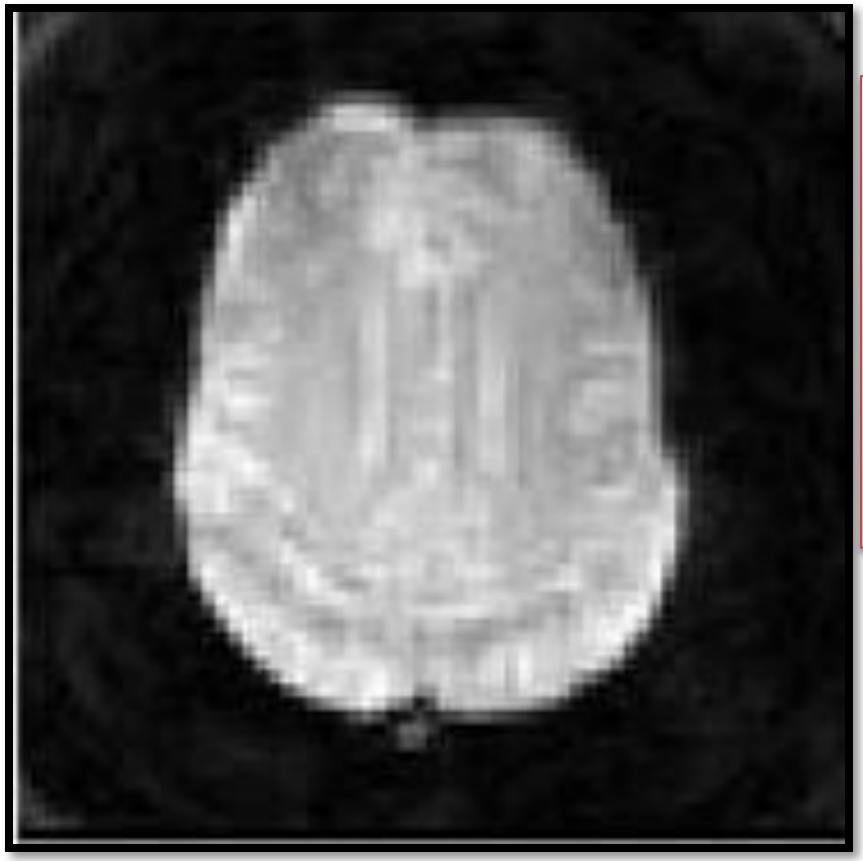
8 bit (1 byte)



8 bit (1 byte) quantization= 256 levels of grey, which is about the limit of what the human eye can distinguish.

Each pixel requires 8 bits to store.

16 bit (2 bytes)



65,536 levels of grey: We can't see it, but the computer can, so our MR scanners capture 16 bit grayscale.

Each pixel requires 16 bits to store.

Said another way,

It takes 1 byte to store each pixel/voxel in an 8 bit image,

but 2 bytes to store each pixel/voxel in a 16 bit image.

16 bit images are vulnerable to a problem called byte swapping.

If a voxel holds 16 bits (2 bytes), then it turns out there are 2 competing standards for ordering those bytes.

Intel based machines (PCs, Linux, newer macs) use the "Little Endian" order: least significant byte first:

Little portion
0 to 255

Big portion
256 to 65535

SGI, older MACs and Suns use "Big Endian" order, most significant byte first:

Big portion
256 to 65535

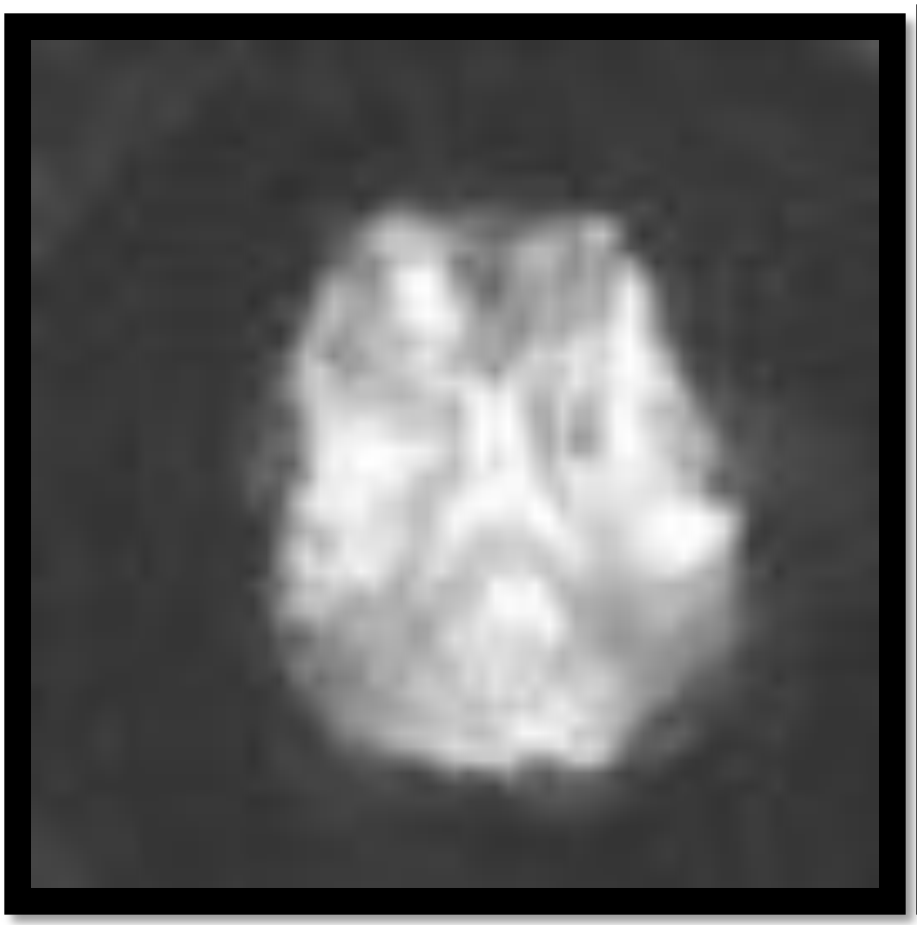
Little portion
0 to 255

If you prepare an image on a big endian machine and move it to a little endian machine (or vice-versa)

You may encounter an image displayed in the wrong byte order for the displaying machine.

Newer operating systems and programs usually handle the problem for you, but you may see it some day.

Byte Swapping



Right

Wrong

Afni provides a program called 2swap to swap the 2 bytes in a 16 bit image, if they look funny.

Afni also provides 4swap, to swap the bytes in a 4 byte (32 bit) image, if it looks wrong.

NaNs

NaN=Not a Number

Result from nonsensical operations; e.g., (∞ * 0).

NaN cannot be used in any mathematical calculations (they are not numbers).

Occasionally, NaNs in images voxels create difficulties, and must be replaced with 0's.

Summary

To understand image size, you must know the units of storage in the computer: bits and bytes.

In medical imaging, you will typically deal with 1 bit (binary mask) images, and 16 bit images.

You might use 8 bit grayscale images for papers and presentations, because they look just as good, but take less storage space.

Bits

Bytes

Binary

8 bit

16 bit

Byte Swapping

Little Endian

Big Endian

NaN