Set 7. Lossy Data Compression

Skill 7.01: Describe the purpose of lossy data compression

Skill 7.01 Concepts

Lossy compression algorithms are techniques that reduce file size by discarding the less important information.

Nobody likes losing information, but some types of files are so large that there's just not enough space to keep all the original data, plus we didn't need all that data in the first place. That's often the case with photos, videos, and audio recordings; files that try to capture the beautiful complexity of the world we live in.

Computers can capture an incredible amount of detail in a photo—but how much of that detail can humans actually perceive? As it turns out, there's a lot of detail that we can remove. Lossy compression algorithms are all about figuring out clever ways to remove detail without humans noticing (too much).

For example, here's a photo of a cat in a hat:

Here's that same photo, but compressed to *half the size*:

Can you tell the difference? At a quick glance, I can't— and even if I could with more effort, it'd still be worth saving half the disk space.

Let's explore some of the clever ways that lossy compression algorithms remove detail to reduce file size.

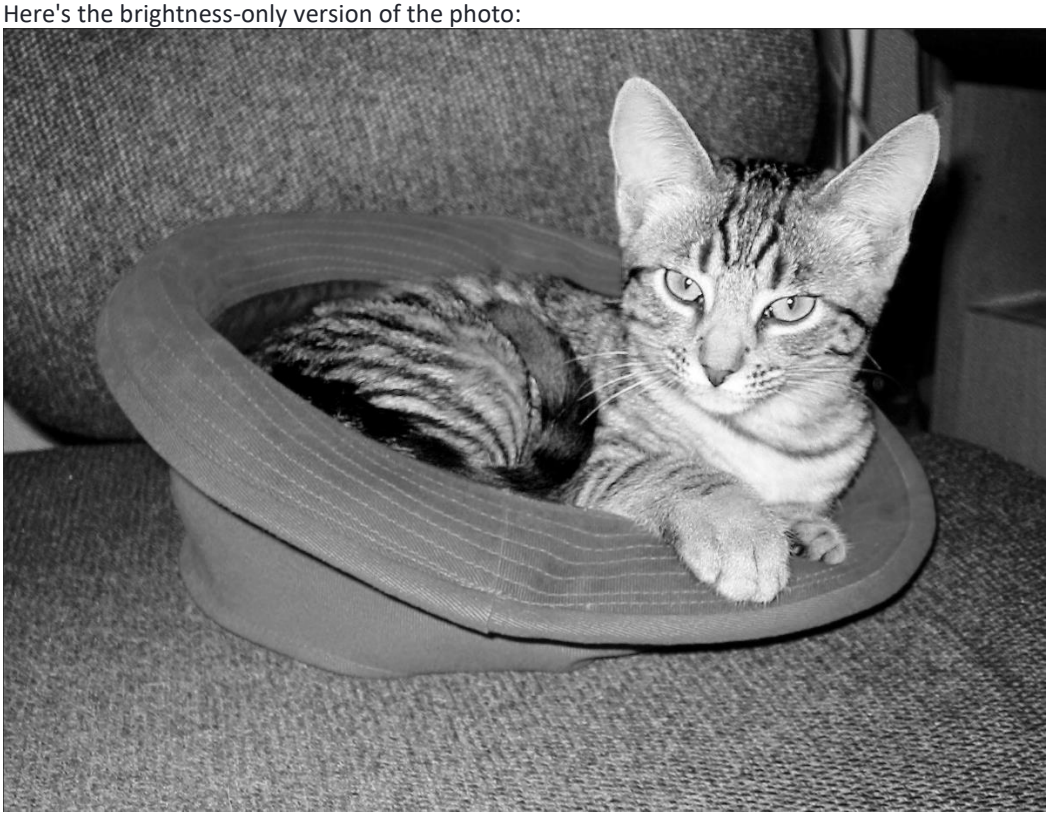
Skill 7.02: Describe lossy image compression techniques

Skill 7.02 Concepts

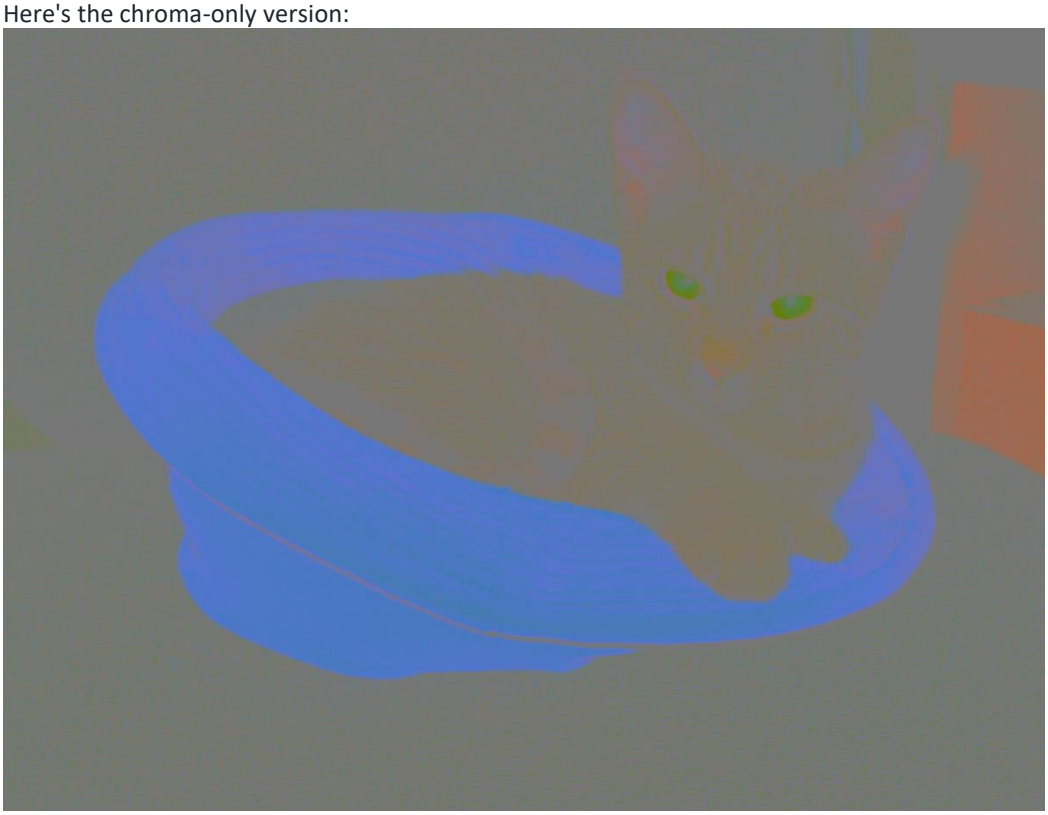
The human eye is better at perceiving differences in brightness than differences in color. A compression algorithm can take advantage of that fact by keeping the brightness while reducing the amount of color information, a process known as **chroma subsampling**.

Let's try it on the photo of the cat in the hat. The first step is to separate the brightness information from the chroma (color).

Here's the brightness-only version of the photo:



Here's the chroma-only version:



This photo is interesting because the cat and the chair are both fairly colorless- the blue of the hat is what stands out in the chroma version. (Perhaps we should all own black and white cats, to save disk space? Hmm...)

Let's zoom into an 8x2 block of chroma from the left eye:

Each of those pixels are slightly different chroma values, so there are 16 values total. We can average each 2x2 block and

set its color to the average, so that there are only 4 values total:

The result looks quite similar to the original, but it has a quarter of the original color information. If we apply that to the entire image, we can save a lot of space without affecting perception much.

Chroma subsampling is a process used in many compression algorithms that deal with images, including the popular file formats JPEG and MPEG.

Those algorithms also use a process called discrete cosine transform (DCT) to simplify the details in the brightness layer. If you'd like to learn more, https://csfieldguide.org.nz/en/chapters/coding-compression/image-compression-using-jpeg/

Skill 7.02 Exercise 1

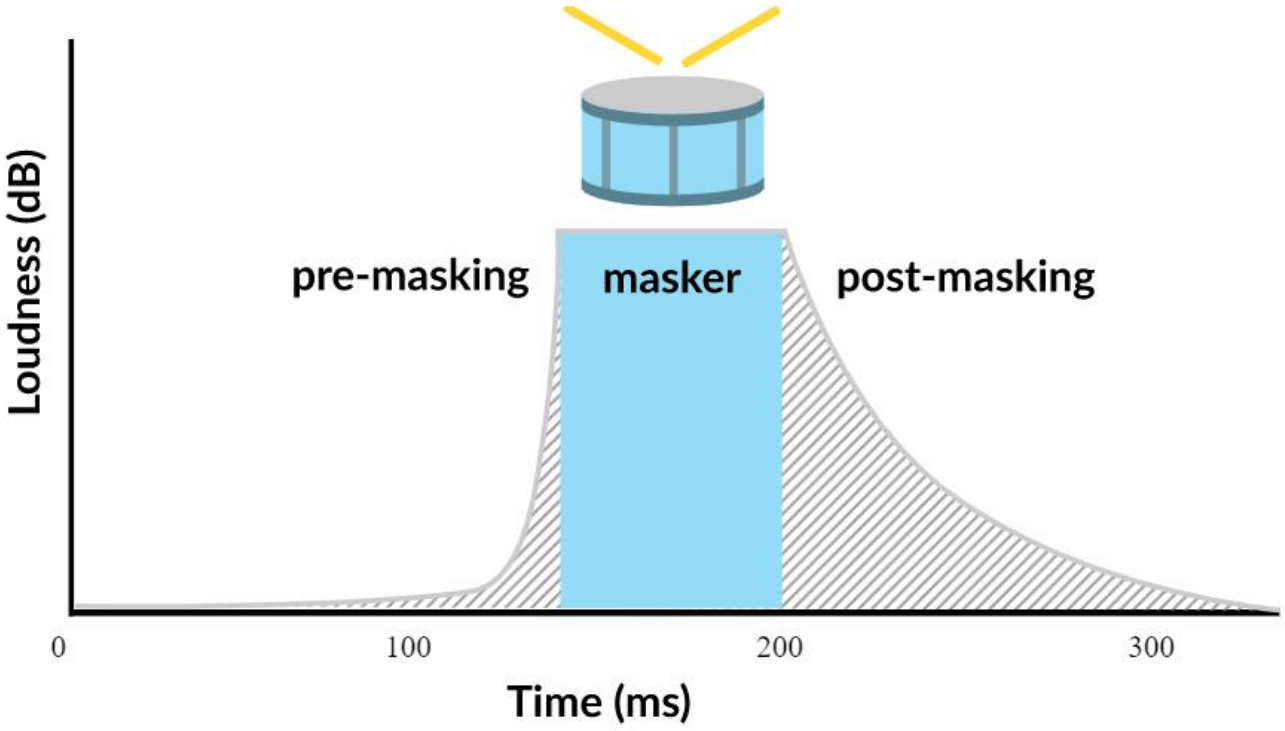
Skill 7.03: Describe lossy audio compression techniques

Skill 7.03 Concepts

The human ear has limitations to what it can hear. Audio compression algorithms can analyze an audio file and discard the sounds that are outside our hearing capacity, a process known as **perceptual audio coding**.

One interesting limitation of our ears is known as **temporal masking**. That's when a sudden sound can mask (hide) other sounds for a period after it occurs—and even a bit before!

For example, imagine a song with a loud drum beat. The beat masks sounds for about 20 milliseconds before it happens, and for up to 200 milliseconds after it happens. This graph illustrates the masking effect:



The computer sees those hidden sounds in the recorded audio file, but our ears can't hear them, so audio compression algorithms can safely discard that information or represent it with fewer bits.

Compression algorithms can also use other limitations of our natural auditory process, like the high frequency limit and simultaneous masking. If you'd like to learn more, you can research the fascinating field of **psychoacoustics**.

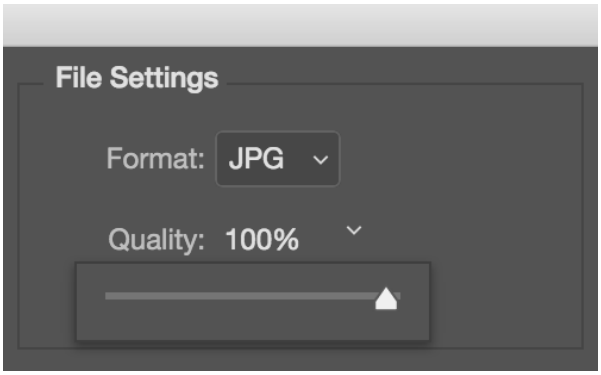
Skill 7.03 Exercise 1

Skill 7.04: Identify situations best suited for lossy or lossless compression

Skill 7.04 Concepts

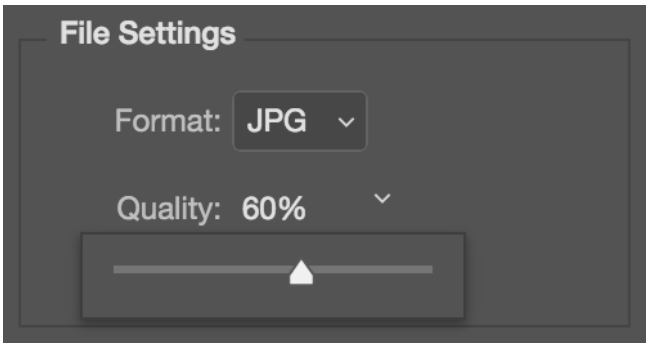
When we use a lossless compression algorithms, we can always reconstruct 100% of the original data. With lossy compression algorithms, we're losing some % of the original data; maybe 5%, maybe 10%, maybe 70%. How much do we lose? We can decide that based on our use case for the data.

Consider the earlier photo of the cat in the hat. If the plan is to put that photo in a presentation about cat fashion and project it on a large screen, we probably want to keep as much detail as possible. In that case, we can either use a lossless compression algorithm (like PNG) or we can use a lossy compression algorithm (like JPEG) and specify a high quality (like 100%). Photo editing applications often give you those options.



At 100% quality, the 400x300 photo takes up 169 KB of space:

What if we want to use that photo in a website, and our target users for the website are in another country on low-bandwidth connections? The smaller the file, the faster the download. We want them to see the photo, but for this use case, high quality isn't as important as download speed. We'll definitely want to use a lossy compression algorithm, and we can specify a lower quality when exporting. The algorithm can tweak its internal dials to simplify details even more.



At 60% quality, the photo takes up 48 KB:

That still looks pretty good - definitely usable for a low-bandwidth website. How low can we go? Here's the photo at 1% quality:

It's definitely not perfect—the green of the eyes seems to be smearing into the fur, and there are artifacts where the hat meets the chair. But it's also *only 12 KB*, less than a tenth of the original size. It's impressive how much information we can lose but still convey so much detail.

We can also specify quality for lossy compression algorithms that transform audio and video files. You've probably seen low quality, highly compressed videos around the Web; videos are the most likely to get compressed since they're so large to begin with.

Whenever we use lossy compression, we're always making a trade-off between quality and size, and it's up to us to find the settings that work best for our use case.

Skill 7.03 Exercise 1