Revisiting images formats for the web

Every so often I see comparisons between image formats that say one format is better than others or that one format is better for a given task but I've always wondered where the numbers came from and what testing criteria was used. Rather than take things at face value, I want to make sure that whatever decision I make it is backed up with data. I've put all the files and scripts on a Github repository for you to run the same tests and see if the results match mine. Beware that the TIFF images are very large and may take a large chunk of your data plan if you download them on mobile.

This post does not cover HEIC/HEIF and AVIF image formats. To cover those two formats well, I need more time to compile and test the tools and want to make sure that I don't mix tool research with image quality. I will post the results of any research on those two formats in a separate post.

Before we jump into looking at the quality of compressed images, let's take a look at what's available and what's coming down the pipeline as far as image formats for the web.

Format	Initial Release	Open?	Type	Available Encoder	Encoder to use	Notes
GIF	1987	No	Lossless	Yes	ImageMagick	
JPG	1992	Yes	Lossy	Yes	ImageMagick	According to Wikipedia patents for JPE technologies expired
PNG	1996	Yes	Lossless	Yes	ImageMagick	Also Provides animation support. Chec [caniuse.com](https://caniuse.com/#feat for supported browsers
WebP	2010	Yes	Lossless and Lossy	Yes	ImageMagick or directly with cwebp	Based of WebM video compression

The process

As Kornel Lesinski writes in How to compare images fairly:

> Absolutely the worst way to compare images is to convert one lossy format to another and conclude you “can’t see the difference”.
>
> Why is it bad? Save a photo as a couple JPEGs at quality=98 and quality=92. It will be hard to tell them apart, but their file sizes will differ by nearly 40%! Does it prove that JPEG is 40% better than… itself? No, it shows that “quality appears the same, but the file much is smaller!” can easily be nonsense that proves nothing.
>
> To make a fair comparison you really have to pay meticulous attention to encoder settings, normalizing quality, and ensuring that compared images are in fact comparable.
>
> It’s really hard to make a fair comparison.

Before we even start encoding the images we have to do a few things:

- Find a lossless, high-quality image to use as the source for the exercises
- Decide what tools we will use to encode the images. Sometimes this may be decided for you as there may not be many tools for the newer formats
- Decide what criteria you will use for your testing and how will you measure it
- Will you measure objective quality using tools like DSSIM?
- Will you compare file size for a given quality?
- **How will you decide which metric is more important?**
- Make sure that the tools produce similar output. For example, all formats should use chroma subsampling or none should
- **Figure out what are the equivalent settings for the formats that you’re testing. Q=80 for a JPEG image may not be the same as Q=80 for other formats**
- Test all formats at the same or equivalent quality
- Make sure that all format encoders can work from the same source
- If the format offer lossless and lossy compression use lossy to match what JPEG does

I chose to work with different images in TIFF format. Information about the specific images is listed below:

- Images from Los Lajones Estate, downloaded from their website

- Image of the USS California from Wikimedia Commons is in the public domain
- Images from Hubblesite taken from bat shadow and NGC 6302: The "Butterfly Nebula" are in the public domain as stated in the hubblesite copyright page

We'll use different encoders for different formats, below is the list of formats with their associated image encoders. All the binaries were reinstalled to make sure I have the latest versions available via Homebrew as of this writing:

- PNG: ImageMagick
- JPG: ImageMagick
- WebP: cwebp

First Test: Equal Quality to measure file size

The first test I wanted to run is what happens if we encode a TIFF source image to all image formats that we can test with the same quality value, in this case, 80. It is important to note that a JPEG image encoded at 80 quality is not the same as a lossy WebP image encoded at the same quality. We do it this way because it's the easiest way to test and it's what I would do in Photoshop of when running image compression with tools like imagemin

The questions that I want to answer with this test:

Keeping quality constant, what loseless format provides the smaller file size?

Rather than type the command every time that I run the test, and to make the results reproducible, I created the Bash script below

```
#! /usr/bin/env bash

# Variable holding name of source image.
SOURCE_IMAGE='STSCI-H-p2022a-f-4398x3982'

'STSCI-H-p2022a-f-4398x3982'# Variables holding names of
```

```
# encoders's binaries
IMAGE_MAGICKbp'
HEIC_ENCODER='heif-enc'

ech'
HEIC_ENCODER='HEIC_ENCODER='heif-enc' ${IMAGE_MAGICK} 2>/dev/null; then
  echo encoding to PNG
  ${IMAGE_MAGICK} ${SOURCE_IMAGE}.tif \
  -quality 80 ${SOURCE_IMAGE}.png
  echo encoding to JPG
  ${IMAGE_MAGICK} ${SOURCE_IMAGE}.tif \
  -quality 80 ${SOURCE_IMAGE}.jpg
else
  echo cannot convert to PNG or JPG${IMAGE_MAGICK}WEBP_ENCODER} 2>/dev/nu
  echo encoding to lossy WebP
  ${WEBP_ENCODER} -q 80 \
  ${SOURCE_IMAGE}.tif \
  -o ${SOURCE_IMAGE}.webp
else
  echo cannot convert to WEBP
fi

if hash ${HEIC_ENCODER} 2>/dev/null; then
  echo encoding to lossy HEIC
  ${HEIC_ENCODER} --quality 80 \
  ${SOURCE_IMAGE}.png
else
  echo could not encode to HEIC
fi
```

My Results with images encoded from TIF high quality sources and JPG where TIF was not an option:

Format	File Size
TIFF (base)	15MB
PNG	13.9MB
JPG	855KB

Format	File Size
WebP	266KB

There is a lot of research and tweaking to obtain optimal results.

So in the naïve, all quality is the same test, WebP wins by a lot. remember that image quality is not a straight equivalency across formats as explained earlier.

Finding the optimal quality

I know that optimal quality depends on the type of image and the screens we're working with, but an initial step on determining our optimal quality may be to establish what are the best compression settings for each format. We're likely to be serving at least two with our sources or srcset images. To answer this question we'll do a two-step process:

- We create a set of WebP and a set of JPG images with quality ranging from 50 to 100
- We'll use SSIM to provide an objective metric to use in comparing the images.

We then analyze the SSIM results to decide which of the compressed images gives us the best combination of quality measured by SSIM against a 100 quality compressed PNG image (for some reason ImageMagick's compare command will not work against a TIFF source image) and file size.

The Scripts

Each step of the process uses its own script.

The first scripts generates the multiple images. It is essentially the same script that we used for the previous evaluation, except that we have to pass the name of the image, without extension, as a parameter when we invoke the script.

```
if hash ${WEBP_ENCODER} 2>/dev/null; then
  for i in {50..100..10}
    do
      echo encoding to lossy WebP at ${i} quality
```

```
      ${WEBP_ENCODER} -q ${i} \
    ${i}SOURCE_IMAGE}.tif \
      -o ${SOURCE_IMAGE}-${i}.webp
  done
else
  echo cannot convert to WEBP
fi
```

The second script does the comparison. I would much rather use a direct SSIM encoder rather than using ImageMagick's compare command but none of the available encoders works as well as I would like.

So we use `magick compare` as our comparison tool and we run it against every WebP and JPG image we created using the 100 quality PNG as our standard to compare against.

I'd rather compare against the TIFF image but IM errors out with that comparison so, in my opinion, PNG is the next best available testing option.

This code will only work on Bash shells version 4 and higher.

```
SOURCE_IMAGE=$1
IMAGE_MAGICK_COMPARE='magick compare'

echo starting to w'magick compare'mparison

if [ -f ${SOURCE_IMAGE}-100.png ]; then
  echo ${SOURCE_IMAGE}-100.png exits
  for ${SOURCE_IMAGE}i0}
    do
      echo "running comparisons for webp at ${i} quality"
      ${IMAGE_MAGICK_COMPARE} -metric ssim \
      ${SOURCE_IMAGE}-100.png \
      ${SOURCE_IMAGE}-${i}.webp \
      null:
  done

  for i in {50..100.${i}"running comparisons for webp at ${i} quality"i at
```

```
      ${IMAGE_MAGI${i}"running comparisons for jpg at ${i} quality"
      ${IMAGE_MAGICK_COMPARE} -metric ssim \
      ${SOURCE_IMAGE}-100.png \
      ${SOURCE_IMAGE}-${i}.jpg \
      null:
  done
else
  echo can\'t run the WebP comparison
fi
```

The Results

The first table uses STSCI-H-p2022a-f-4398x3982 from the Nasa image library... The image is 14MB and may use a significant portion of your data plan in mobile.

The file sizes are generally what I expected and the differences between the SSIM values is similar enough to make file size becomes the primary consideration.

Quality	WebP File Size	WebP SSIM Value	JPG File Size	JPG SSIM
100	2.1MB	0.986584	10.9M	0.992733
90	639KB	0.981029	3.6MB	0.985442
80	266KB	0.975904	2.1MB	0.982087
70	183KB	0.973957	1.5MB	0.978859
60	153KB	0.973044	1.1MB	0.974777
50	128KB	0.972133	864KB	0.972219

The second table uses geisha-high-resas the image to test. This is a much brighter and deep contrast color image. The image is 11.9MB and may use a significant portion of your data plan in mobile.

Quality	WebP File Size	WebP SSIM Value	JPG File Size	JPG SSIM
100	2.6MB	0.980442	5.2MB	0.992432
90	757KB	0.961136	1.4MB	0.966931
80	300KB	0.947507	788KB	0.956424

Quality	WebP File Size	WebP SSIM Value	JPG File Size	JPG SSIM
70	217KB	0.943466	535KB	0.949292
60	188KB	0.941836	407KB	0.942214
50	166KB	0.939598	332KB	0.937974

The final example, the USS California image presents some interesting variance for analysis. The image is 12.8MB and may use a significant portion of your data plan in mobile.

I ran the compression and the SSIM comparison in separate steps, so I made the incorrect assumption that grayscale WebP images would exhibit the same behavior as color ones where the WebP files scored better in the SSIM metric across the board instead of fluctuating as they did. Need to do more research, particularly if it has to do with encoding settings on the WebP size and whether the `-jpeg_like` and `-shap_yuv` flags would change the results in any way.

Quality	WebP File Size	WebP SSIM Value	JPG File Size	JPG SSIM
100	10.1MB	1	11.2MB	0.998955
90	3MB	0.956338	3.6MB	0.951193
80	1.1MB	0.909765	2.1MB	0.930837
70	669KB	0.891505	1.5MB	0.91801
60	530KB	0.88458	1.1MB	0.907007
50	426KB	0.877016	864MB	0.898054

What’s missing

There are two additional image formats that should be in the encoding tests but are not.

HEIC is an image format based on the HEVC video code. The only encoder I found for the format produced significantly larger sizes than any other formats. I need to run additional tests to make sure I’m encoding it as it should be and that the larger sizes are not a result of my doing it wrong.

AVIF is the image format based on the open-source AV1 video format. The

same encoder that creates HEIC files (lib-heif) will, supposedly, create AVIF files when running with a flag. I have yet to get it to work.

I was able to install the libavif reference implementation but I'll take a little time to document the process and then set up another test to see how heic and avif compare to WebP, JPG, and PNG.

Conclusions

There is additional work that needs to happen regarding format-specific configuration and encoding flags and test them to see if the formats are affected by the flags you use when doing the compression.

Depending on the type of images you use on your site or project, evaluating the file sizes of a representative number of images is always a good idea.