WebP: Balancing Efficiency and Quality

Katherine Qin
EE-569 Image and video coding
Taught by Professor Ebrahimi

What is WebP?

- Developer: Google (2010)
- Goal: Replace older formats with a single, highly efficient format for the web.
- Advantage: Smaller file sizes at comparable quality.

Technology in 2010: Social networks, Nexus 1, and iPad/Smartphone wars

[1][2]

WebP Architecture

- VP8 (Lossy)
  - Based on VP8 video intra-frame compression
  - Predictive block coding
- VP8L (Lossless)
  - Reconstruction with previously seen pixel patterns

[2][3][4][5]

VP8L Lossless Encoder

- Input: Image in RGBA
- Output: Compressed VP8L bitstream
- Advanced spatial prediction and entropy coding

Pixel-level spatial prediction

Color cache (dictionary-like)

Back-reference coding

Multi-level entropy coding (Huffman)

[6]

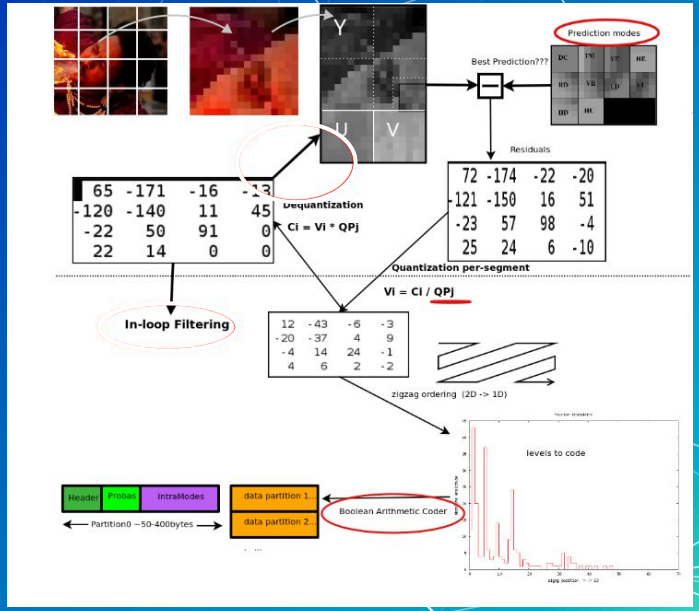
Lossy Encoder

- Input: image in RGB
- Output: compressed VP8 bitstream
- Mandatory processing:
  RGB → YUV 4:2:0

Intra Prediction modes

Adaptive Block Quantization (QP)

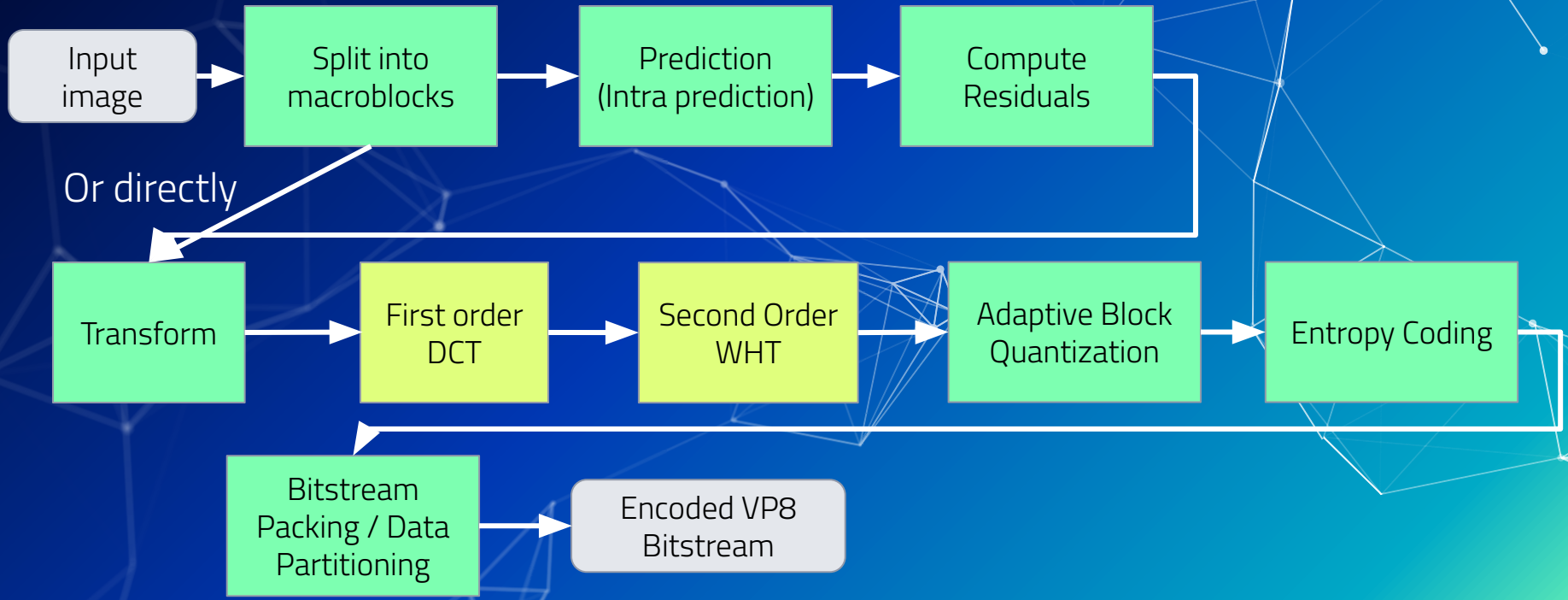
Arithmetic Coding



[7][8]

Lossy Encoder

16x16 luma + 8x8 chroma



Implementation Details

- Input: PNG, JPEG, TIFF, or WebP
- Output: WebP
- Implementation: libwebp 1.6.0

```
cwebp input.png -q 80 -o output.webp
```

Parameters:

Lossless: -lossless

- Encode image losslessly, default=off

Lossy: -q, -m 0

- -q: Quality (0-100)
- -m: Compression methods (0=fast, 6=slowest), default=4

[9]

Additional topic: Near Lossless

- Problem: a ceiling for maximum lossy bitrate
- -near_lossless option
  - Hypothesis: simplified lossy mode
- -near_lossless is lossy but uses lossless (VP8L) encoder

-near_lossless

use near-lossless image preprocessing, default=100 The range is 0 (maximum preprocessing) to 100 (no preprocessing, the default). [9][10]

Experiment Workflow

Choose Images → Lossless Coding → Choose Thresholds → Lossy Coding

Code chosen images using lossless coding. No additional tags

Lossy code all images at the 4 bit rate thresholds. No pre or post processing

15 Images from
JPEG AIC-3 Dataset (5)
CLIC 2022 (10)
All 8-bit images
Variation in color, objects, etc.

4 bit-rate thresholds for lossy coding.
Lowest - JPEG at 60% quality
Highest - 90% of bpp of best lossy compression codec
Linear interpolation between points for additional 2 points

Visual Quality Metrics

- PSNR
- MS-SSIM
- VIF
- BRISQUE

PyTorch Image Quality (PIQ), Python 3.13

[11]

Basics of SSIM

- Structural Similarity Index Measure
- Developed by Wang et al. (2004)
- HVS is highly adapted to extract structural information

$$SSIM(\mathrm{Im}_1,\mathrm{Im}_2) = \left[l(\mathrm{Im}_1,\mathrm{Im}_2)\right]^{\alpha}\left[c(\mathrm{Im}_1,\mathrm{Im}_2)\right]^{\beta}\left[s(\mathrm{Im}_1,\mathrm{Im}_2)\right]^{\gamma} \quad (\alpha>0,\beta>0,\gamma>0)$$

- **Luminance comparison function:** $l(\mathrm{Im}_1,\mathrm{Im}_2) = \frac{2\mu_1\mu_2 + C_1}{\mu_1^2 + \mu_2^2 + C_1}$ (C_1=constant)
- **Contrast comparison function:** $c(\mathrm{Im}_1,\mathrm{Im}_2) = \frac{2\sigma_1\sigma_2 + C_2}{\sigma_1^2 + \sigma_2^2 + C_2}$ (C_2=constant)
- **Structure comparison function:** $s(\mathrm{Im}_1,\mathrm{Im}_2) = \frac{\sigma_{1,2} + C_3}{\sigma_1\sigma_2 + C_3}$ (C_3=constant)

[12][13]

MS-SSIM

- Multiscale SSIM (Wang et al.)
- Output range [0, 1]
  - 1 highest quality
  - 0 poor quality.

$$\text{MS-SSIM} = [l_m(\mathbf{x}, \mathbf{y})]^{\alpha M} \cdot \prod_{j=1}^{M} [c_j(\mathbf{x}, \mathbf{y})]^{\beta_j} [s_j(\mathbf{x}, \mathbf{y})]^{\gamma_j}.$$

[14]

Legend and Note

00002_853x945.png → 02

00003_945x840.png → 03

00007_1600x1200.png → 07

00009_2048x1536.png → 09

00010_2592x1946.png → 10

Apple_tree_1365_2048.png → apple

Bridge_1848_1224.png → bridge

Celebration_2048_1365.png → cele

Crosswalk_backlight_2048_1360.png → cross

neon_2048_1365.png → neon

night_event_1463_2048.png → night

portrait_veil_1391_2048.png → port

rapeseed_field_2048_1365.png → rapeseed

street_dusk_2048_1135.png → street

bridge_1848_1224.png → bridge

video_game_2048_1152.png → video

Lossless Comparison

	Lossless bpp				
Images	**HEIF**	**JPEG2000**	**JPEG XL**	**WebP**	**AVIF**
02	10.77	7.55	6.43	7.35	9.25
03	9.39	7.60	6.55	7.24	8.49
07	10.68	8.65	7.07	8.49	10.54
09	11.67	7.43	6.37	7.40	11.04
10	11.04	7.54	6.25	7.60	10.30
apple	7.91	6.48	5.67	6.53	7.34
bridge	10.33	7.45	5.89	7.04	9.75
cele	9.24	6.48	5.48	6.80	8.87
cross	16.42	11.51	10.48	11.84	16.78
neon	15.55	13.06	11.13	11.54	15.91
night	14.97	11.54	10.40	11.47	15.62
port	13.10	8.08	6.80	8.28	12.94
rapeseed	12.48	9.39	8.89	8.96	12.49
street	14.51	10.82	9.47	10.00	14.56
video	8.10	7.47	5.83	6.85	8.32

Red - worst
Green - best
Purple - second best

Lossless Comparison - Photos

10

Apple

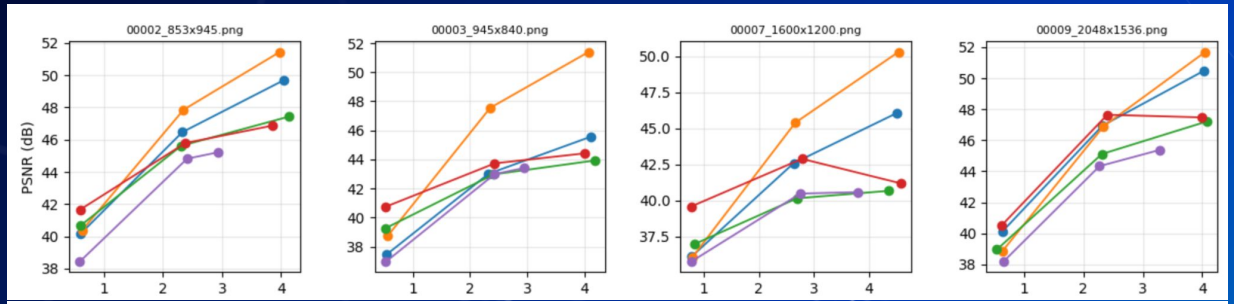
Cele

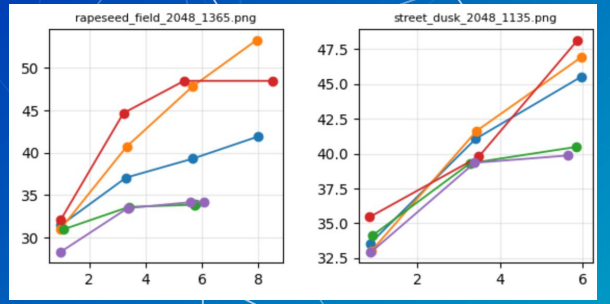
Port

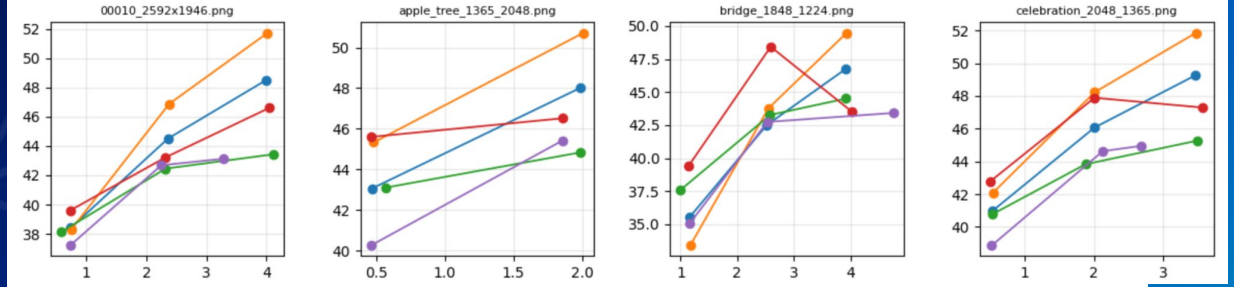
Cross

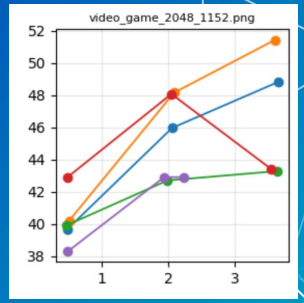
Lossy Results

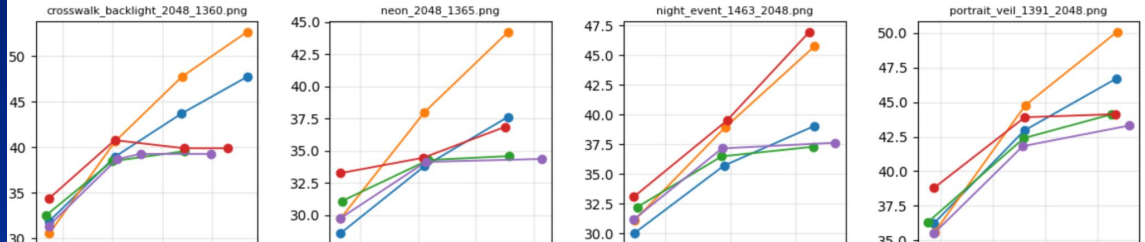
PSNR vs Bitrate



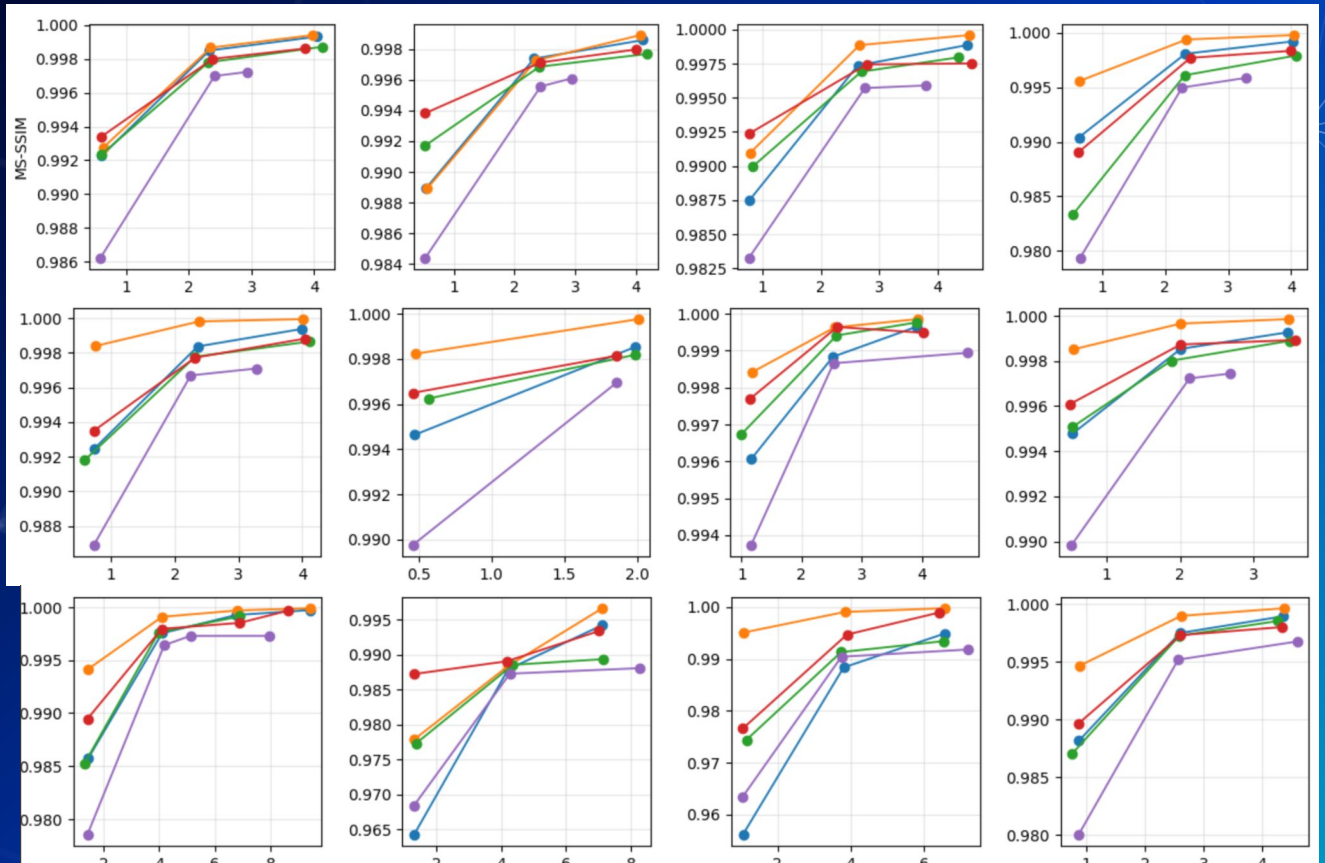




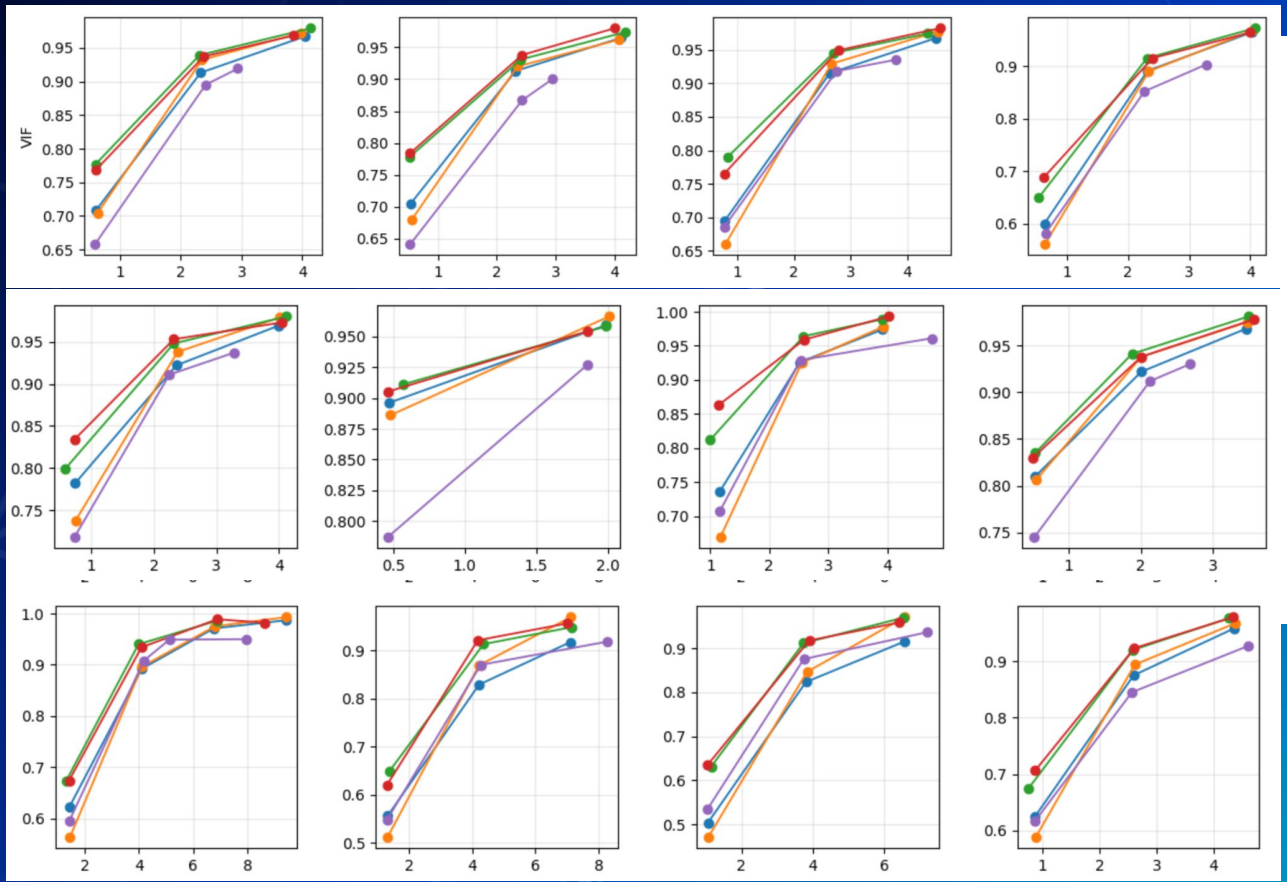




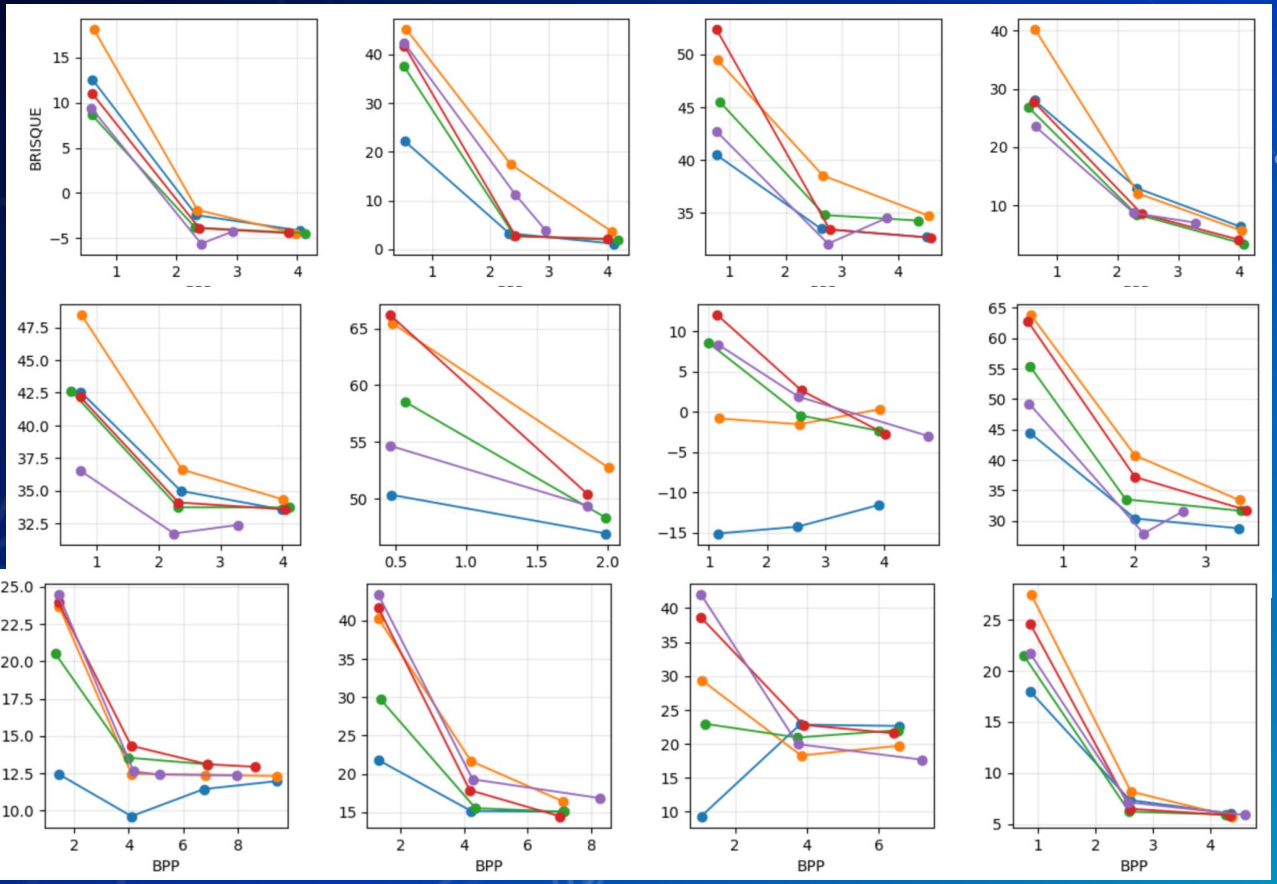
MS-SSIM vs Bitrate



VIF vs Bitrate



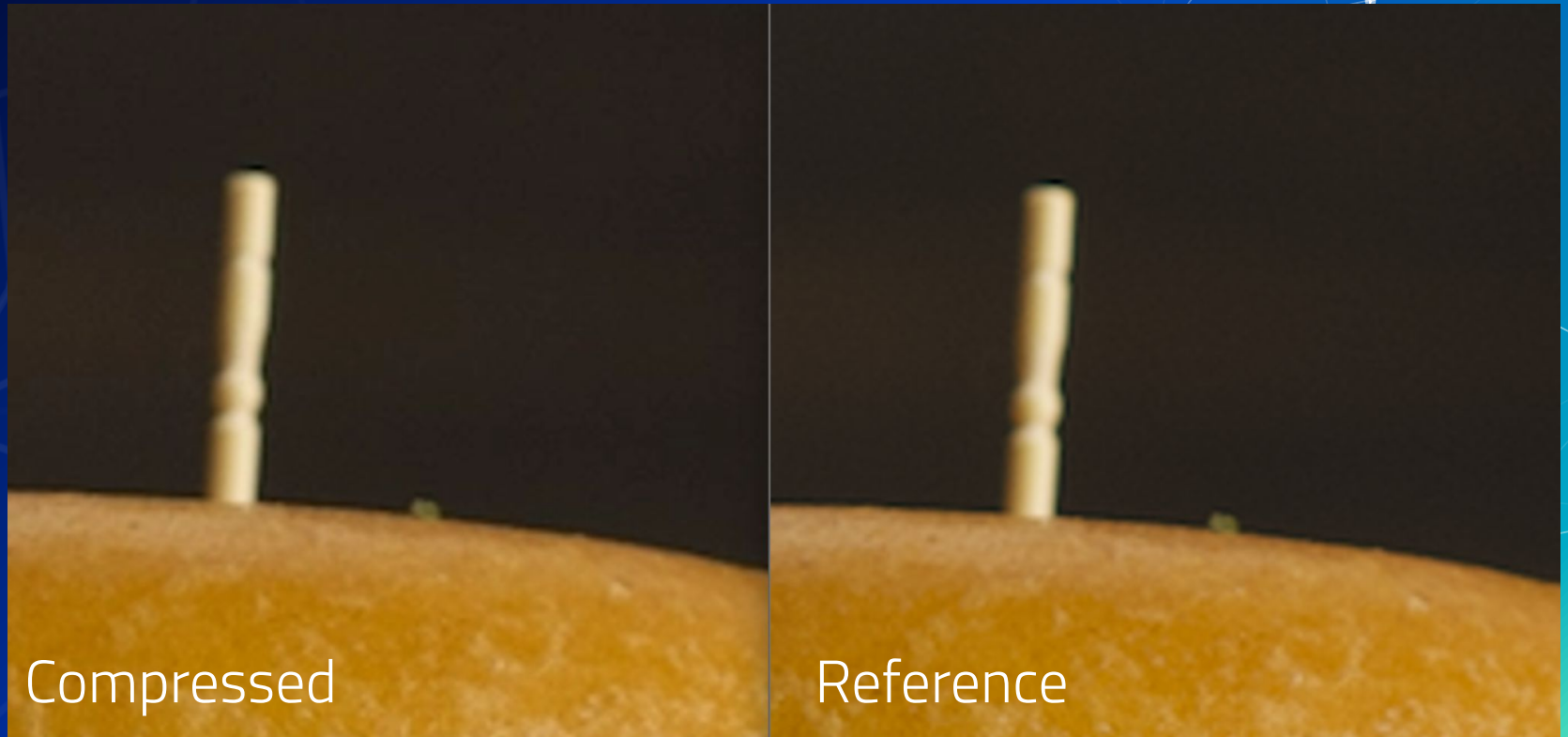
BRISQUE vs Bitrate



Example Images - Blocking Artifact

Example Image - Colour

Example Image - Colour and Background Pattern



Key Takeaways

- Outperformed by newer formats (lossy)
- Performs reasonably well (lossless)
- Observed artifacts

Next steps: more bitrate points, more diverse image content, more visual quality metrics

Chlubna, T., Zemčík, P. Comparative survey of image compression methods across different pixel formats and bit depths. SIViP 19, 981 (2025). https://doi.org/10.1007/s11760-025-04579-6
Johnston N. et al. Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Network. CVFR. Google research (2017) https://doi.org/10.48550/arXiv.1703.10114
Dai Y, Xue C, and Zhou L. Visual saliency guided perceptual adaptive quantization based on HEVC intra-coding for planetary images. PLOS one. (2022) https://doi.org/10.1371/journal.pone.0263729

References

[1] https://developers.google.com/speed/webp

[2] https://developers.googleblog.com/en/webp-a-new-image-format-for-the-web/

[3] http://blog.webmproject.org/2010/05/introducing-webm-open-web-media-project.html

[4] https://en.wikipedia.org/wiki/Resource_Interchange_File_Format

[5] VP8 https://datatracker.ietf.org/doc/rfc6386/

[6] https://developers.google.com/speed/webp/docs/compression#lossless_webp

[7] https://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/37073.pdf

[8] https://developers.google.com/speed/webp/docs/compression

[9] https://github.com/webmproject/libwebp/blob/main/doc/tools.md

[10] https://groups.google.com/a/webmproject.org/g/webp-discuss/c/0GmxDmlexek/m/3ggyYsaYdFEJ

[11] https://piq.readthedocs.io/en/latest/

[12] https://ece.uwaterloo.ca/~z70wang/research/ssim/

[13] https://www.imatest.com/docs/ssim/

[14] https://ieeexplore.ieee.org/document/1292216

Appendix

Here are some supporting slides

RIFF (Resource Interchange File Format)

- Introduced: 1991 by Microsoft & IBM
- Type: Open, free container format
- Purpose: Store data in structured chunks (metadata + content)
- Design:
  - Little-endian format (x86 systems)
  - Minimal overhead (~20 bytes)
  - Extensible for metadata

WebM and VP8

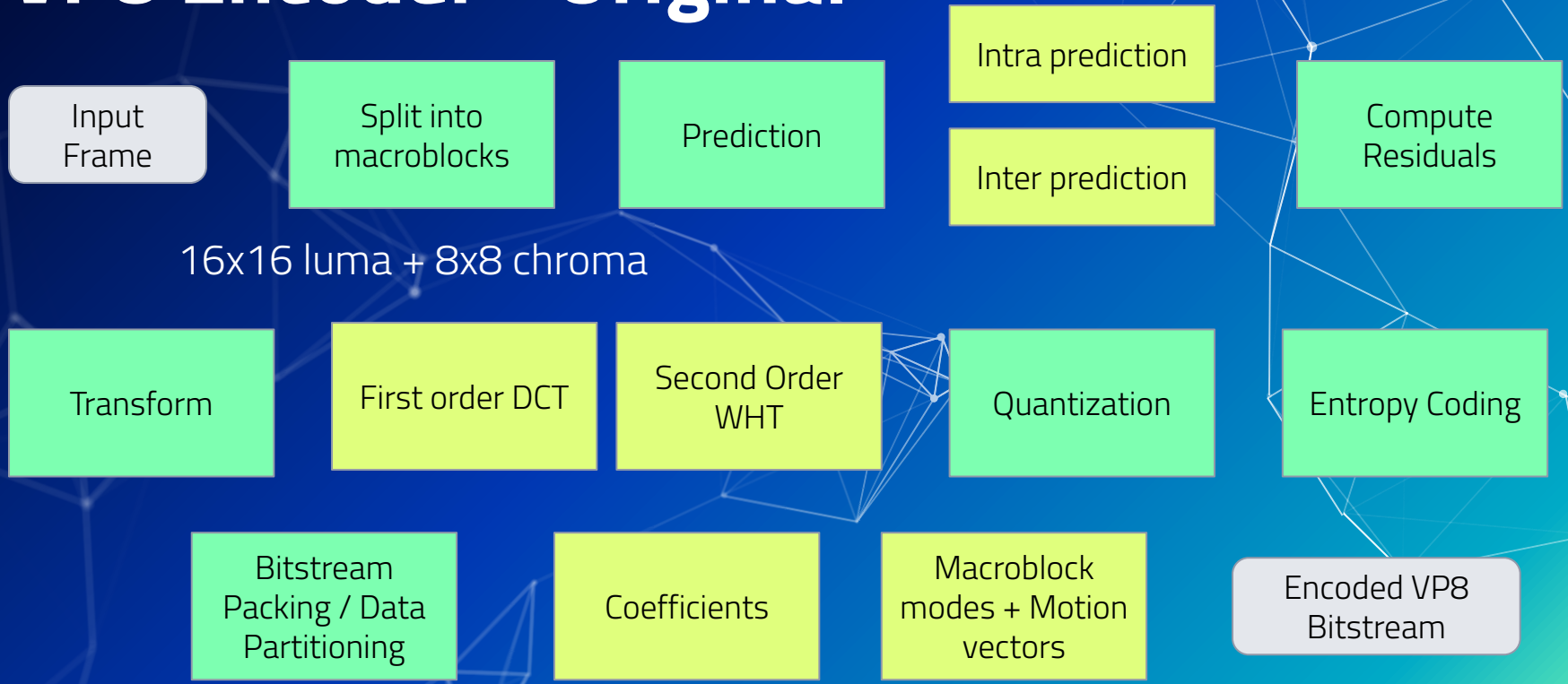
- WebM project: open, freely implementable media format for the web,
- Addresses: lack of an open alternative to proprietary video codecs.
- WebM combines the VP8 video codec, the Vorbis audio codec, and a container based on Matroska. V
- VP8 delivers high-quality video while efficiently adapting to varying bandwidth and device constraints, making it well suited for web and mobile video delivery.

More about VP8

- Open source video codec for web video, part of Google's WebM project.
- Designed for low bandwidth and wide range of client devices.
- Uses hybrid block-based transforms: 4x4 DCT + 4x4 Walsh-Hadamard on DC coefficients.
- Flexible reference frames (Last, Golden, Alternate) and intra/inter prediction (including SPLITMV and TM_PRED).
- Boolean arithmetic coding with frame-level adaptive context; supports parallel decoding for multi-core processors.

https://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/37073.pdf

VP8 Encoder - Original



Walsh-Hadamard Transform (WHT)

- Purpose: Further decorrelates DC coefficients in a 16×16 luma macroblock.
- Type: 4×4 second-order transform applied after 4×4 DCT.
- Benefit: Reduces redundancy among DC values, improving compression efficiency.
- Chosen for its fast, simple, and effective decorrelation

https://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/37073.pdf

LibWebP Links

- See https://developers.google.com/speed/webp for details on the image format.
- The latest source tree is available at https://chromium.googlesource.com/webm/libwebp
- It is released under the same license as the WebM project. See https://www.webmproject.org/license/software/ or the "COPYING" file for details.

WebP codec is a library to encode and decode images in WebP format. This package contains the library that can be used in other programs to add WebP support, as well as the command line tools 'cwebp' and 'dwebp' to compress and decompress images respectively.

libwebp File Structure

```
## Files

*   bin/cwebp : encoding tool
*   bin/dwebp : decoding tool
*   bin/gif2webp : gif conversion tool
*   bin/img2webp : animation creation tool
*   bin/vwebp : webp visualization tool
*   bin/webpinfo : webp analysis tool
*   bin/webpmux : webp muxing tool
*   bin/anim\_diff : webp file comparison tool
*   bin/anim\_dump : tool for dumping animation frames
*   bin/get\_disto : tool for calculating file distortion
*   bin/webp\_quality : webp quality estimation tool
*   doc/ : manual in HTML and text formats
*   lib/ : static libraries
*   include/webp : headers
```

libwebp Encoding tools - P1

The easiest use should look like:

```shell
cwebp input.png -q 80 -o output.webp
```

which will convert the input file to a WebP file using a quality factor of 80 on
a 0->100 scale (0 being the lowest quality, 100 being the best. Default value is
75).

You might want to try the `-lossless` flag too, which will compress the source
(in RGBA format) without any loss. The `-q` quality parameter will in this case
control the amount of processing time spent trying to make the output file as
small as possible.

If input size (-s) for an image is not specified, it is assumed to be a PNG,
JPEG, TIFF or WebP file. Note: Animated PNG and WebP files are not supported.

Options:

```
-h / -help ..... short help
-H / -longhelp ..... long help
-q <float> ..... quality factor (0:small..100:big), default=75
-alpha_q <int> ..... transparency-compression quality (0..100),
                     default=100
-preset <string> ..... preset setting, one of:
                       default, photo, picture,
                       drawing, icon, text
   -preset must come first, as it overwrites other parameters
-z <int> ..... activates lossless preset with given
               level in [0:fast, ..., 9:slowest]
```

```
-m <int> ..... compression method (0=fast, 6=slowest), default=4
-segments <int> ..... number of segments to use (1..4), default=4![alt text](image.png)
-size <int> ..... target size (in bytes)
-psnr <float> ..... target PSNR (in dB. typically: 42)
```

libwebp Encoding tools - P2

```
-s <int> <int> ......... input size (width x height) for YUV
-sns <int> ............. spatial noise shaping (0:off, 100:max),
default=50
-f <int> ............... filter strength (0=off..100), default=60
-sharpness <int> ....... filter sharpness (0:most .. 7:least
sharp), default=0
-strong ................ use strong filter instead of simple
(default)
-nostrong .............. use simple filter instead of strong
-sharp_yuv ............. use sharper (and slower) RGB->YUV
conversion
-partition_limit <int> . limit quality to fit the 512k limit on
                         the first partition (0=no
                         degradation ... 100=full)
-pass <int> ............ analysis pass number (1..10)
-qrange <min> <max> .... specifies the permissible quality range
                         (default: 0 100)
-crop <x> <y> <w> <h> .. crop picture with the given rectangle
-resize <w> <h> ........ resize picture (*after* any cropping)
-resize_mode <string> .. one of: up_only, down_only, always
(default)
-mt ..................... use multi-threading if available
-low_memory ............ reduce memory usage (slower encoding)
-map <int> ............. print map of extra info
-print_psnr ............ prints averaged PSNR distortion
-print_ssim ............ prints averaged SSIM distortion
-print_lsim ............ prints local-similarity distortion
-d <file.pgm> .......... dump the compressed output (PGM file)
-alpha_method <int> .... transparency-compression method (0..1),
default=1
```

```
-alpha_filter <string> . predictive filtering for alpha plane,
                         one of: none, fast (default) or best
-exact ................. preserve RGB values in transparent area,
default=off
-blend_alpha <hex> ..... blend colors against background color
                         expressed as RGB values written in
                         hexadecimal, e.g. 0xc0e0d0 for red=0xc0
                         green=0xe0 and blue=0xd0
-noalpha ............... discard any transparency information
-lossless .............. encode image losslessly, default=off
-near_lossless <int> ... use near-lossless image preprocessing
                         (0..100=off), default=100
-hint <string> ......... specify image characteristics hint,
                         one of: photo, picture or graph

-metadata <string> ..... comma separated list of metadata to
                         copy from the input to the output if
                         present.
                         Valid values: all, none (default), exif,
                         icc, xmp

-short ................. condense printed message
-quiet ................. don't print anything
-version ............... print version number and exit
-noasm ................. disable all assembly optimizations
-v ..................... verbose, e.g. print encoding/decoding
times
-progress .............. report encoding progress
```

Suggestions for visual quality tuning

```
The main options you might want to try in order to further tune
the visual
quality are:

-preset -sns -f -m

Namely:

*   `preset` will set up a default encoding configuration
targeting a particular
    type of input. It should appear first in the list of options,
    so that
    subsequent options can take effect on top of this preset.
    Default value is
    'default'.
*   `sns` will progressively turn on (when going from 0 to 100)
some additional
    visual optimizations (like: segmentation map re-enforcement).
    This option
    will balance the bit allocation differently. It tries to take
    bits from the
    "easy" parts of the picture and use them in the "difficult"
    ones instead.
    Usually, raising the sns value (at fixed -q value) leads to
    larger files,
    but with better quality. Typical value is around '75'.
```

```
*   `f` option directly links to the filtering strength used by
the codec's
    in-loop processing. The higher the value, the smoother the
    highly-compressed
    area will look. This is particularly useful when aiming at
    very small files.
    Typical values are around 20-30. Note that using the option
    -strong/-nostrong will change the type of filtering. Use "-f
    0" to turn
    filtering off.
*   `m` controls the trade-off between encoding speed and
quality. Default is 4.
    You can try -m 5 or -m 6 to explore more (time-consuming)
    encoding
    possibilities. A lower value will result in faster encoding
    at the expense
    of quality.
```

Other Options

-f 0 ⟶ Filter Strength. This is the main control for the deblocking filter. A higher value means the filter smooths the image more aggressively to hide artifacts. Setting it to 0 is for maximum bitrate within lossy mode.

-sns 0 ⟶ Spatial Noise Shaping. This is a highly effective pre-quantization process. It determines which areas (spatially) should receive more or fewer bits. A high value (100) means the encoder is very aggressive at shifting bits away from low-perceptibility areas (lower final file size). 0 disables it, maximizing the bitrate.

Other Options

https://developers.google.com/speed/webp/docs/cwebp

-sharp_yuv → Sharper YUV Conversion. This controls the quality of the mandatory RGB to YUV conversion (the chroma subsampling step). Using this flag gives you a more mathematically accurate (and visually sharper) conversion, but it takes longer.

-sharpness 0 → Specify the sharpness of the filtering (if used). Range is 0 (sharpest) to 7 (least sharp). Default is 0.

-size → Specify a target size (in bytes) to try and reach for the compressed output. The compressor will make several passes of partial encoding in order to get as close as possible to this target.

Other Options

-f → post processing

-sns → encoder

-sharp_yuv → preprocessing

-sharpness → postprocessing

Other Options

Images	-q 100	-f 0	-sns 0	-sharp_yuv	-sharpness 0	-m 0	None - Ref
09	2.72	0.44	0.45	0.44	0.44	0.51	0.44
cross	5.02	1.59	1.63	1.66	1.59	1.78	1.59
video	1.82	0.32	0.33	0.34	0.32	0.45	0.32

	-q 100				
Images	-f 0	-sns 0	-sharp_yuv	-sharpness 0	-m 0
09	2.72	2.72	2.84	2.72	3.28
cross	5.02	5.02	5.28	5.02	7.94
video	1.82	1.82	1.99	1.82	2.24

	-q 100 and -m 0			
Images	-f 0	-sns 0	-sharp_yuv	-sharpness 0
09	3.28	3.28	3.48	3.28
cross	7.94	7.94	8.5	7.94
video	2.24	2.24	2.52	2.24

-near_lossless

Specify the level of near-lossless image preprocessing. This option adjusts pixel values to help compressibility, but has minimal impact on the visual quality. It triggers lossless compression mode automatically. The range is 0 (maximum preprocessing) to 100 (no preprocessing, the default). The typical value is around 60. Note that lossy with -q 100 can at times yield better results.

https://groups.google.com/a/webmproject.org/g/webp-discuss/c/0GmxDmlexek/m/3ggyYsaYdFEJ

Literature - Ceiling

Papers:

https://www.sciencedirect.com/science/article/pii/S0923596512000203

https://infoscience.epfl.ch/server/api/core/bitstreams/4f881b93-33db-4d5c-8e20-c3400a1d3936/content

https://link.springer.com/article/10.1007/s11760-025-04579-6

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0263729

https://ieeexplore.ieee.org/document/7026134

https://developers.google.com/speed/webp/docs/webp_study

https://openaccess.thecvf.com/content_cvpr_2018/papers/Johnston_Improved_Lossy_Image_CVPR_2018_paper.pdf

https://arxiv.org/pdf/2304.04518

Other implementations - Ceiling

GIMP - Reaches under the determined max level as seen in images

FreeConvert.com - Does not reach max at -q 99, but when -q 100 is triggered, the -near_lossless tag is triggered.

Cloudconvert - Is the exact same as -q 99. When -q 100 is triggered, the near_lossless tag is triggered.

Pixelied - Uses only lossy when -q 100

Pickflow -q 100 has the near_lossless tag triggered with the same additional tags

ToWebP -q 100 near_lossless tag is triggered again.

EzGif - also -near_lossless triggered.

SSIM

l = 1 similar brightness
l < 1 one image is darker or brighter

c = 1 similar high contrast
c < 1 one is blurry/flat

s = 1 perfect alignment
s < 1 edges shifted

$$SSIM(\mathrm{Im}_1,\mathrm{Im}_2) = \left[l(\mathrm{Im}_1,\mathrm{Im}_2)\right]^{\alpha}\left[c(\mathrm{Im}_1,\mathrm{Im}_2)\right]^{\beta}\left[s(\mathrm{Im}_1,\mathrm{Im}_2)\right]^{\gamma} \quad (\alpha>0,\beta>0,\gamma>0)$$

- **Luminance comparison function:** (C_1=constant)
$$l(\mathrm{Im}_1,\mathrm{Im}_2) = \frac{2\mu_1\mu_2 + C_1}{\mu_1^2 + \mu_2^2 + C_1}$$
- **Contrast comparison function:** (C_2=constant)
$$c(\mathrm{Im}_1,\mathrm{Im}_2) = \frac{2\sigma_1\sigma_2 + C_2}{\sigma_1^2 + \sigma_2^2 + C_2}$$
- **Structure comparison function:**
$$s(\mathrm{Im}_1,\mathrm{Im}_2) = \frac{\sigma_{1,2} + C_3}{\sigma_1\sigma_2 + C_3} \quad (C_3\text{=constant})$$

Note: MS-SSIM “Scale” means to LPF and downsample by 2

Lossless Comparison