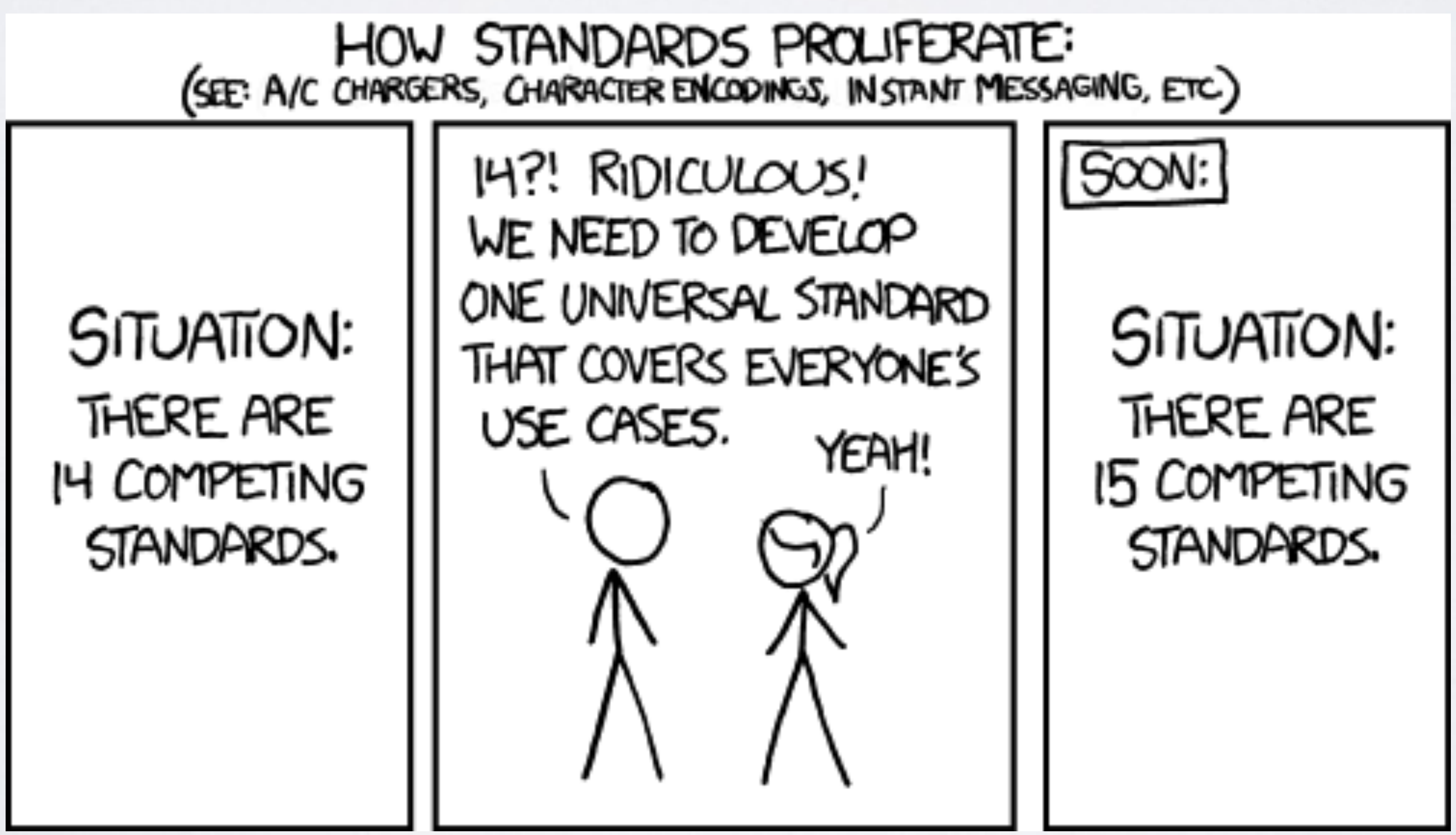
FREE LOSSLESS IMAGE FORMAT

Jon Sneyers and Pieter Wuille
jon@cloudinary.com
Cloudinary
pieter.wuille@gmail.com
Blockstream

ICIP 2016, September 26th

DON'T WE HAVE ENOUGH IMAGE FORMATS ALREADY?

- JPEG, PNG, GIF, WebP, JPEG 2000, JPEG XR, JPEG-LS, JBIG(2), APNG, MNG, BPG, TIFF, BMP, TGA, PCX, PBM/PGM/PPM, PAM, …
- Obligatory XKCD comic:



YES, BUT…

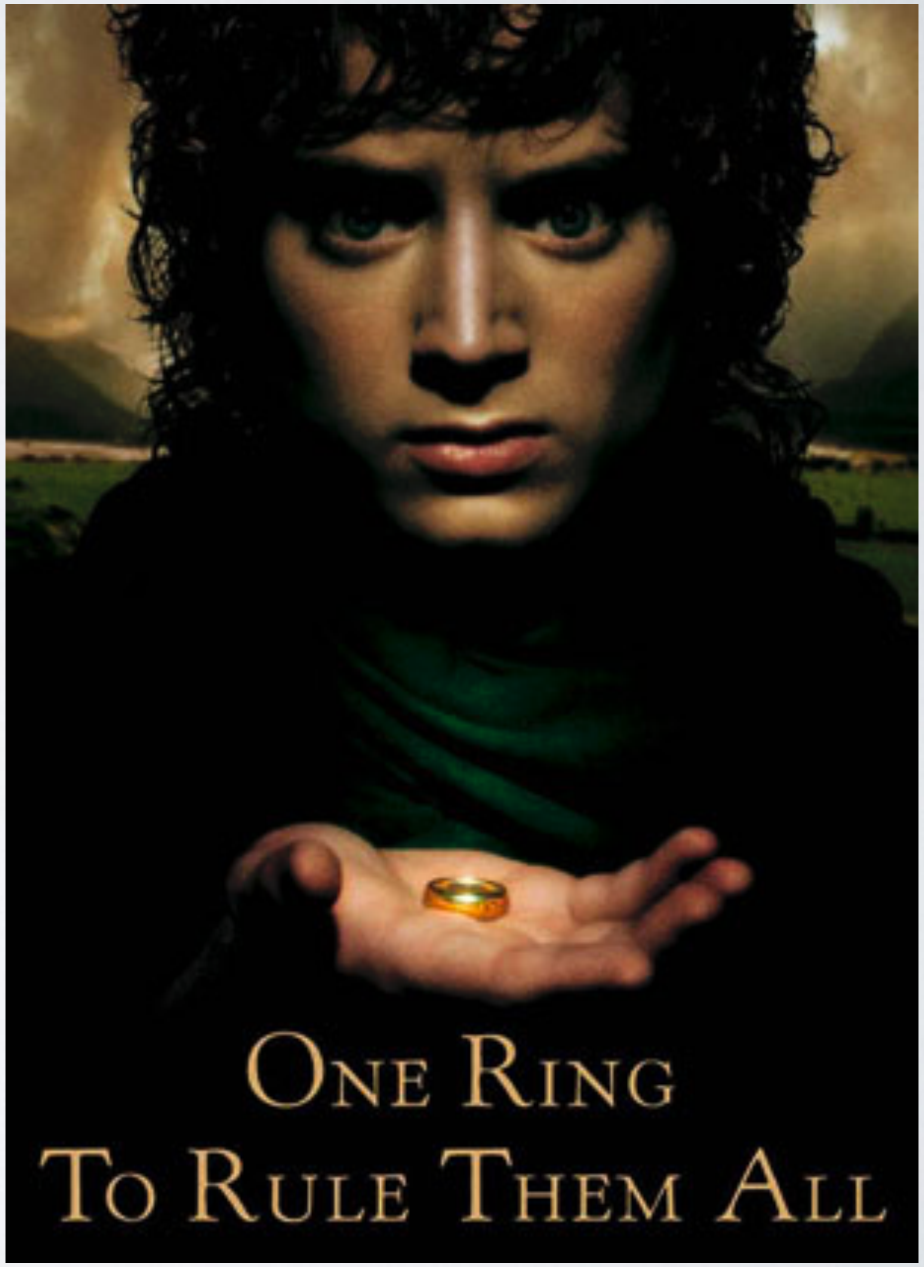
- There are many kinds of images: photographs, medical images, diagrams, plots, maps, line art, paintings, comics, logos, game graphics, textures, rendered scenes, scanned documents, screenshots, …


L
BRACE
BIT
XFCE
the FreeType Project
Groot-Bijgaarden
Bekkerzeel
Zellik
Heysel - Heizel
Laeken - Laken
Jette
Ganshoren
Berchem-Sainte-Agathe
- Sint-Agatha-Berchem
Sint-Martens-Bodegem
Molenbeek-Saint-Jean
- Sint-Jans-Molenbeek
Dilbeek
Bruxelles - Brussel
Anderlecht
Sint-Anna-Pede
Saint-Gilles
- Sint-Gillis
Forest - Vorst
Vlezenbeek
Gaasbeek
Uccle - Ukkel
Zuun
Ruisbroek

EVERYTHING SUCKS AT SOMETHING

- None of the existing formats works well on *all* kinds of images.
  - JPEG / JP2 / JXR is great for photographs, but…
  - PNG / GIF is great for line art, but…
  - WebP: basically two totally different formats
    - Lossy WebP: somewhat better than (moz)JPEG
    - Lossless WebP: somewhat better than PNG
    - They are both .webp, but *you* still have to pick the format

GOAL: ONE FORMAT
THAT COMPRESSES ALL IMAGES WELL



EXPERIMENTAL RESULTS

Corpus		(bit depth)	Lossless formats									JPEG*	
			FLIF	FLIF*	WebP	BPG	PNG	PNG*	JP2*	JXR	JLS	100%	90%
Natural (photo)	[4]	8	1.002	**1.000**	1.234	1.318	1.480	2.108	1.253	1.676	1.242	1.054	0.302
	[4]	16	1.017	**1.000**	/	/	1.414	1.502	1.012	2.011	1.111	/	/
	[5]	8	1.032	**1.000**	1.099	1.163	1.429	1.664	1.097	1.248	1.500	1.017	0.302
	[6]	8	1.003	**1.000**	1.040	1.081	1.282	1.441	1.074	1.168	1.225	0.980	0.263
	[7]	8	1.032	**1.000**	1.098	1.178	1.388	1.680	1.117	1.267	1.305	1.023	0.275
	[8]	8	1.001	**1.000**	1.059	1.159	1.139	1.368	1.078	1.294	1.064	1.152	0.382
	[8]	12	1.009	**1.000**	/	1.854	2.053	2.378	2.895	5.023	2.954	/	/
Artificial	[9]	8	1.039	**1.000**	1.212	1.145	1.403	1.609	1.436	1.803	1.220	1.193	0.233
	[10]	8	**1.000**	1.095	1.371	1.649	1.880	2.478	4.191	7.619	3.572	5.058	2.322
	[11]	8	**1.000**	1.037	1.982	4.408	2.619	2.972	10.31	33.28	33.12	14.87	9.170
	[12]	8	1.106	1.184	**1.000**	2.184	1.298	1.674	3.144	3.886	2.995	3.186	1.155
	[13]	8	**1.000**	1.049	1.676	1.734	2.203	2.769	4.578	10.35	4.371	5.787	2.987

* : Format supports progressive decoding (interlacing).
/ : Unsupported bit depth.
Numbers are scaled so the best (smallest) lossless format corresponds to 1.

Fig. 4. Compressed corpus sizes using various image formats.

HOW DOES IT WORK?

- General outline: pretty traditional
  - Color transform
  - Spatial domain (no DCT/DWT transform)
  - Interlacing
  - Prediction
  - Entropy coding: MANIAC

COLOR TRANSFORM

- RGBA **channel compaction** to reduce effective bit depth if only a subset of the 2^8 or 2^16 possible values effectively occur in the image
- (compacted) **RGB**A to **YCoCg**A
  - $P_{urple} = (R+B)/2$, $Y = (P+G)/2$, $Co = R-B$, $Cg = G-P$
    Note: one extra bit for Co/Cg (signed values)
  - YCoCg is lossless and optional, can also use (permuted / green-subtracted) RGB
- If very sparse colors: **palette** (just like PNG/GIF), arbitrary palette size
- If relatively sparse colors: **color buckets**, a generalization of palette with 'discrete' and 'continuous' buckets to reduce the range of Y/Co/Cg given the value of nothing/Y/Y+Co

INTERVAL COLOR RANGES

- Channel order: A, Y, Co, Cg
- To encode any color value, first compute the interval of 'valid' values based on known constraints
- E.g. if `Y=0`, then we know that `-3 ≤ Co ≤ 3`
- Intervals are derived from YCoCg definition, color buckets, explicitly stored bounds

INTERLACING: ADAM∞

1								2					
3								3					

INTERLACING: ADAM∞

1				4				2				4	
3				4				3				4	

INTERLACING: ADAM∞

1				4				2				4	
5				5				5				5	
3				4				3				4	

INTERLACING: ADAM∞

1		6		4		6		2		6		4	
5		6		5		6		5		6		5	
3		6		4		6		3		6		4	

INTERLACING: ADAM∞

1		6		4		6		2		6		4	
7		7		7		7		7		7		7	
5		6		5		6		5		6		5	
7		7		7		7		7		7		7	
3		6		4		6		3		6		4	

INTERLACING: ADAM∞

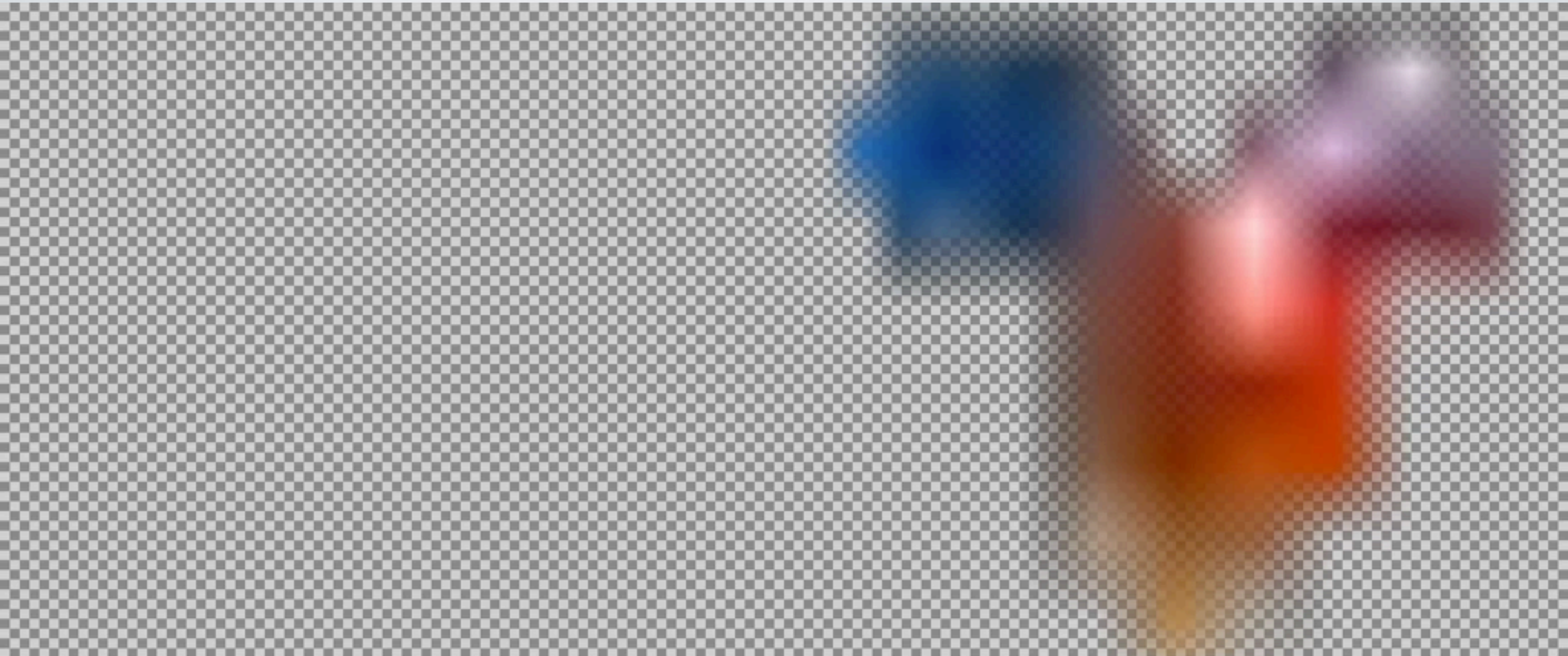
1	8	6	8	4	8	6	8	2	8	6	8	4	8
7	8	7	8	7	8	7	8	7	8	7	8	7	8
5	8	6	8	5	8	6	8	5	8	6	8	5	8
7	8	7	8	7	8	7	8	7	8	7	8	7	8
3	8	6	8	4	8	6	8	3	8	6	8	4	8

INTERLACING: ADAM∞

1	8	6	8	4	8	6	8	2	8	6	8	4	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9
7	8	7	8	7	8	7	8	7	8	7	8	7	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9
5	8	6	8	5	8	6	8	5	8	6	8	5	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9
7	8	7	8	7	8	7	8	7	8	7	8	7	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9
3	8	6	8	4	8	6	8	3	8	6	8	4	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9

ADAM7 vs ADAM∞

or rather: plain RGB vs prioritized YCoCg



PNG (Adam7)
289924 bytes
(.06%)

partial file
200 bytes

FLIF
158184 bytes
(.12%)

PREDICTION

- Key difference with Adam7-PNG: interlacing is taken into account in the prediction/filtering

PNG (ADAM7) PREDICTION

1	8	6	8	4	8	6	8	2	8	6	8	4	8
7	8	7	8	7	8	7	8	7	8	7	8	7	8
5	8	6	8	5	8	6	?	5		6		5	
7		7		7		7		7		7		7	
3		6		4		6		3		6		4	

FLIF PREDICTION

1	8	6	8	4	8	6	8	2	8	6	8	4	8
7	8	7	8	7	8	7	8	7	8	7	8	7	8
5	8	6	8	5	8	6	?	5		6		5	
7		7		7		7		7		7		7	
3		6		4		6		3		6		4	

MANIAC ENTROPY CODING

The main "new thing" in FLIF

Meta-**A**daptive **N**ear-zero **I**nteger **A**rithmetic **C**oding

MANIAC ENTROPY CODING

- **M**eta-**A**daptive **N**ear-zero **I**nteger **A**rithmetic **C**oding
- Base idea: CABAC (context-adaptive binary AC)
- Contexts are not static (i.e. one big fixed array) but dynamic (a tree which grows branches during encode/decode)
  - The tree structure is learned at encode time, encoded in the bitstream
  - Context model *itself* is specific to the image, not fixed by the format (so it is *meta*-adaptive)

CONTEXT MODEL

- Problem: how many contexts?
  - Too few: cannot really capture the actual ‘context’ (contexts that behave differently get lumped together)
  - Too many: too few symbols per context (similar contexts get updated separately)

CABAC

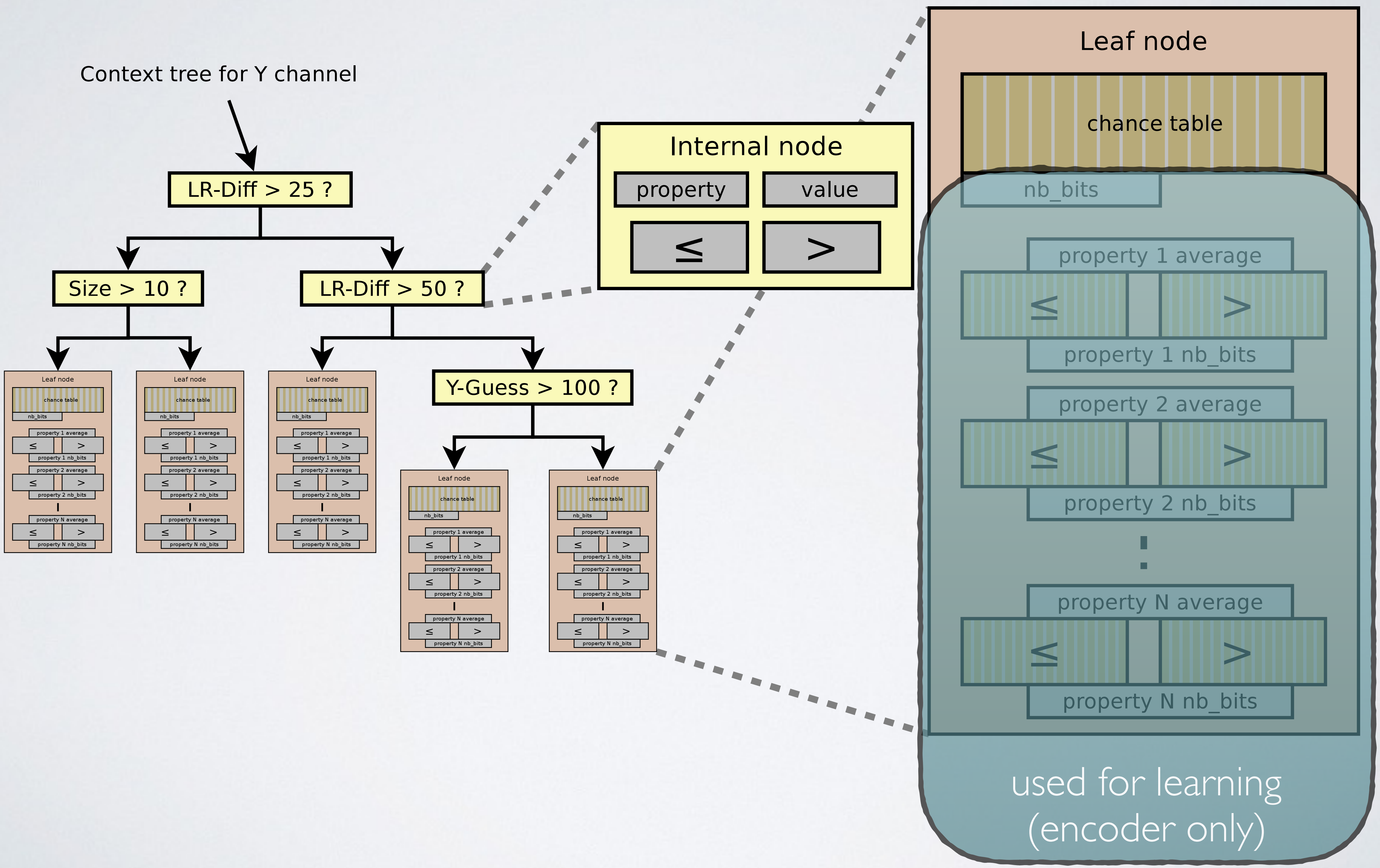
- Example context model: FFV1, “large model”
  - up to **5** properties: (TT-T), (LL-L), (L-TL), (TL-T), (T-TR)
  - Properties are **quantized**, and used to determine the AC context
  - Context are organized in an array (i.e. `context[11][11][5][5][5]`)
  - **Fixed** number of contexts
    - 666 in the “small model”
    - 7563 in the “large model”

MANIAC

- Example context model: FLIF
  - up to **11** properties: e.g. (TT-T), (LL-L), (L-(TL+BL)/2), (T-(TL+TR)/2), (B-(BL+BR)/2), (T-B), the predictor: e.g. median((T+B)/2, T+L-TL, L+B-BL), the median-index, the value of A, the value of Y, the "luma prediction miss": (Y - (YT+YB)/2)
  - Properties are **not quantized**, and used to determine the AC context
  - Contexts are organized in a dynamic structure ("MANIAC tree")
  - **No fixed** number of contexts

MANIAC TREE

MANIAC TREE



Context tree for Y channel
LR-Diff > 25 ?
Size > 10 ?
LR-Diff > 50 ?
Y-Guess > 100 ?
Internal node
property
value
≤
>
Leaf node
chance table
nb_bits
property 1 average
≤
>
property 1 nb_bits
property 2 average
property 2 nb_bits
property N average
property N nb_bits
used for learning
(encoder only)

KEY INSIGHT

- Compression = Machine Learning
  - If you can (probabilistically) predict/classify, then you can compress
- Every ML technique is a potential entropy coder
  - MANIAC: decision trees

ENTROPY CODING

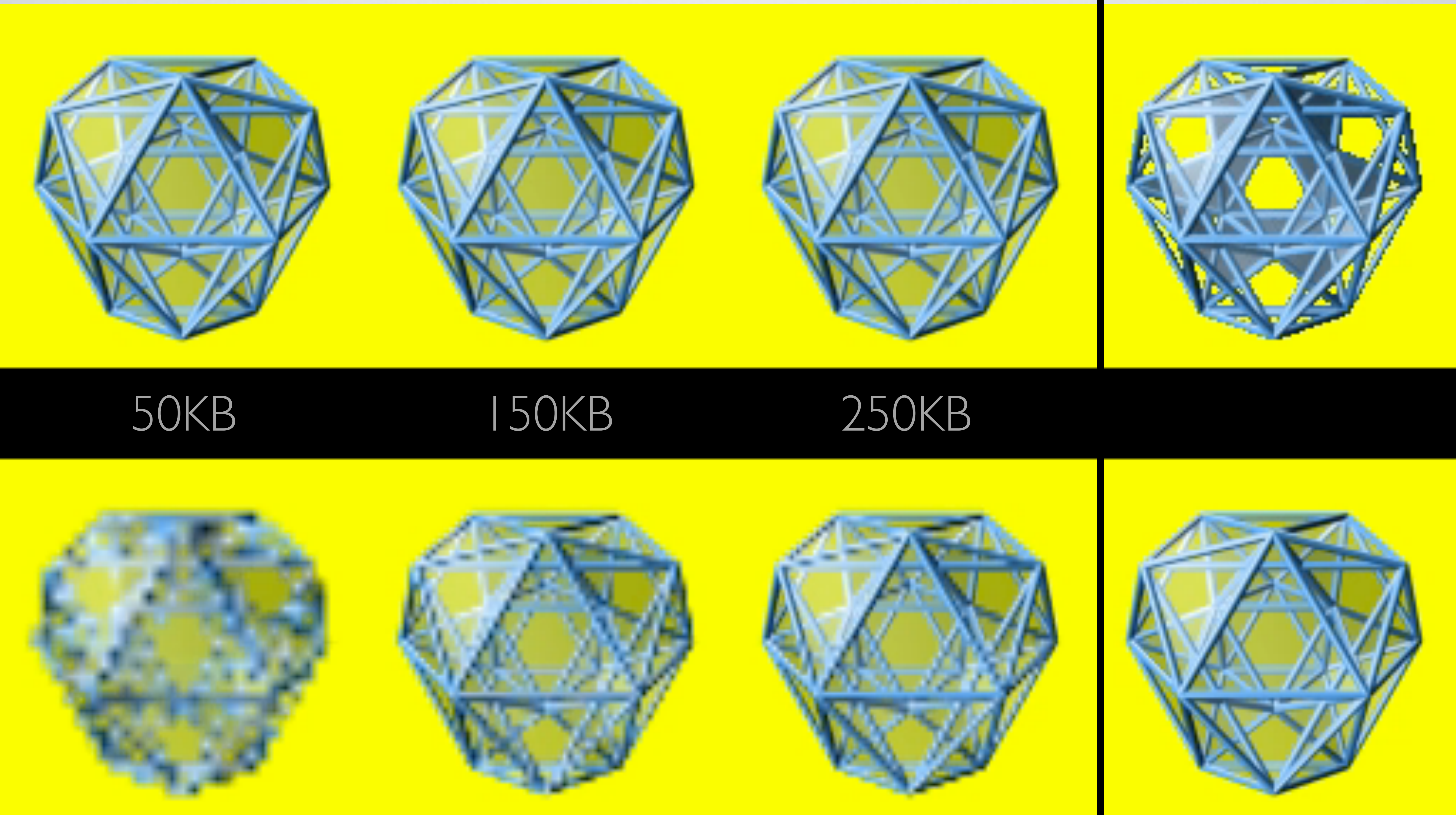
	Huffman	LZW	DEFLATE (LZ + Huffman)	AC (pre-CABAC)	CABAC	MANIAC
Used in	JPEG	GIF	PNG, lossless WebP	JPEG-AC, JPEG 2000, VP8 (WebP)	H.264, FFV1, HEVC (BPG), VP9	FLIF
Global adaptive (initial chances can be tuned)	✅	❌	✅	✅	✅	✅
Local adaptive (chances can be updated)	❌	✅	✅	✅	✅	✅
Context-adaptive (chances per context)	❌	❌	❌	❌	✅	✅
Meta-adaptive (context model can be tuned)	❌	❌	❌ (lossless WebP: somewhat)	❌	❌	✅

FLIF FEATURES

- Up to 16-bit RGBA, lossless (like PNG)
  A=0 pixels can have ***undefined*** RGB values (values not encoded), this is optional
- Interlaced (default) or non-interlaced
- Animation (with some inter-frame features: FrameShape, Lookback)
- Can store metadata (ICC color profile, Exif/XMP metadata)
- Rudimentary support for camera raw RGGB
- Poly-FLIF: javascript polyfill decoder

APNG: 962KB

GIF: 436KB
(256 colors, no full alpha)



FLIF: 526KB

Fully decoded
APNG or FLIF

LOSSY FLIF?

- Encoder can optionally modify the input pixels in such a way that the image compresses better
- This works surprisingly well!
  - Other lossless formats (PNG, lossless WebP) can also be used in a lossy way, but they typically don't even get anywhere *near* the lossy formats
- Plus: there's room for future improvement

MOZJPEG

262,800 BYTES
DSSIM: 0.00134261
PSNR: 33.5447

VS

PNG8

264,653 BYTES
DSSIM: 0.00639207
PSNR: 31.9077

MOZJPEG

VS

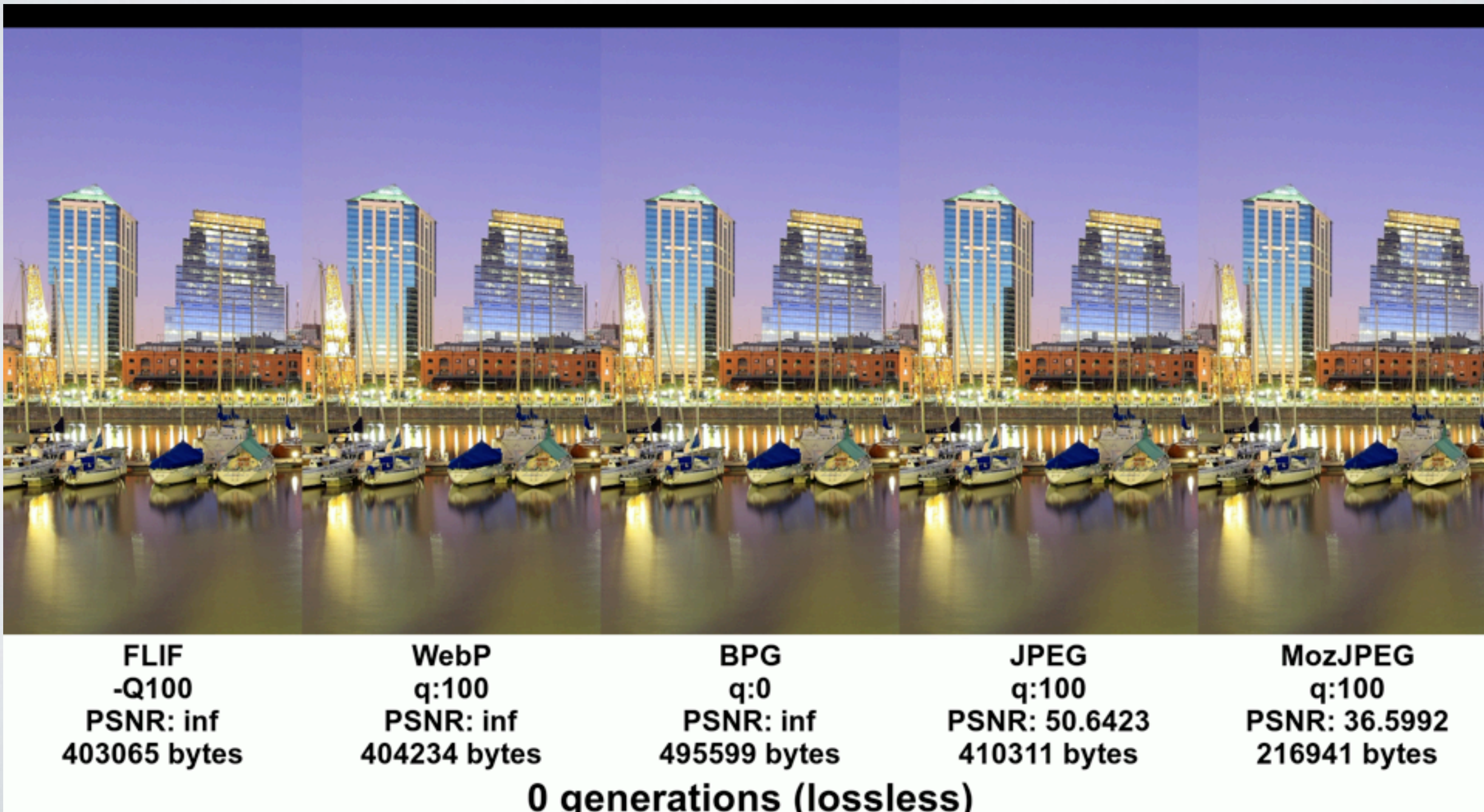
FLIF

262,800 BYTES
DSSIM: 0.00134261
PSNR: 33.5447

248,225 BYTES
DSSIM: 0.00106984
PSNR: 37.2284

DO WE STILL NEED LOSSY?

- Maybe we don't need (inherently) lossy **formats** anymore?
  - Lossy is still useful, but maybe lossy encoding to lossless target formats is good enough?



FUTURE DIRECTIONS

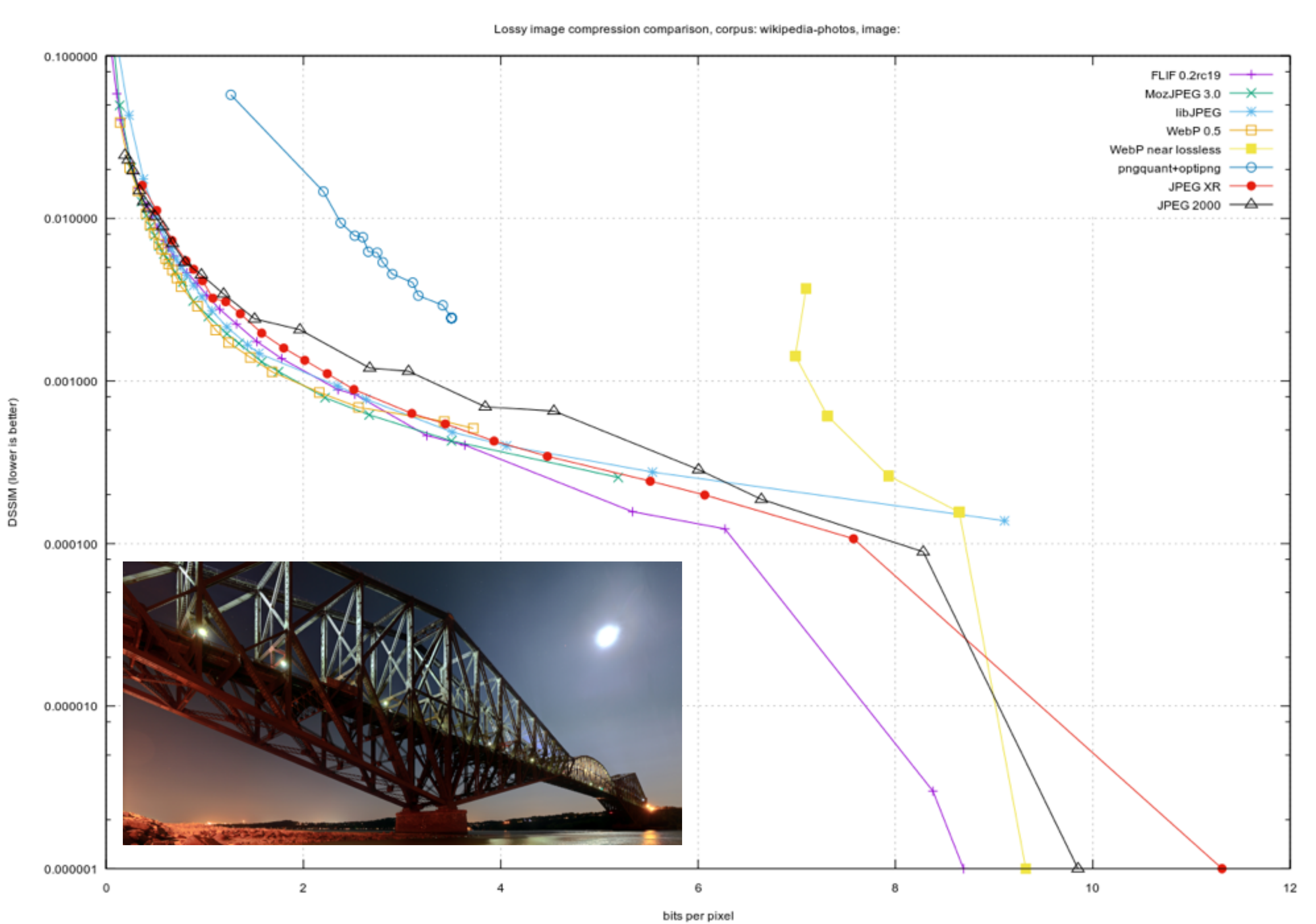
- Apply MANIAC to other formats / general-purpose compression
- Try MANIAC-style entropy coding based on other ML techniques (Neural nets, SVM, etc etc)
- Improve (decoding) performance
- Improve (lossless/lossy) compression

QUESTIONS?

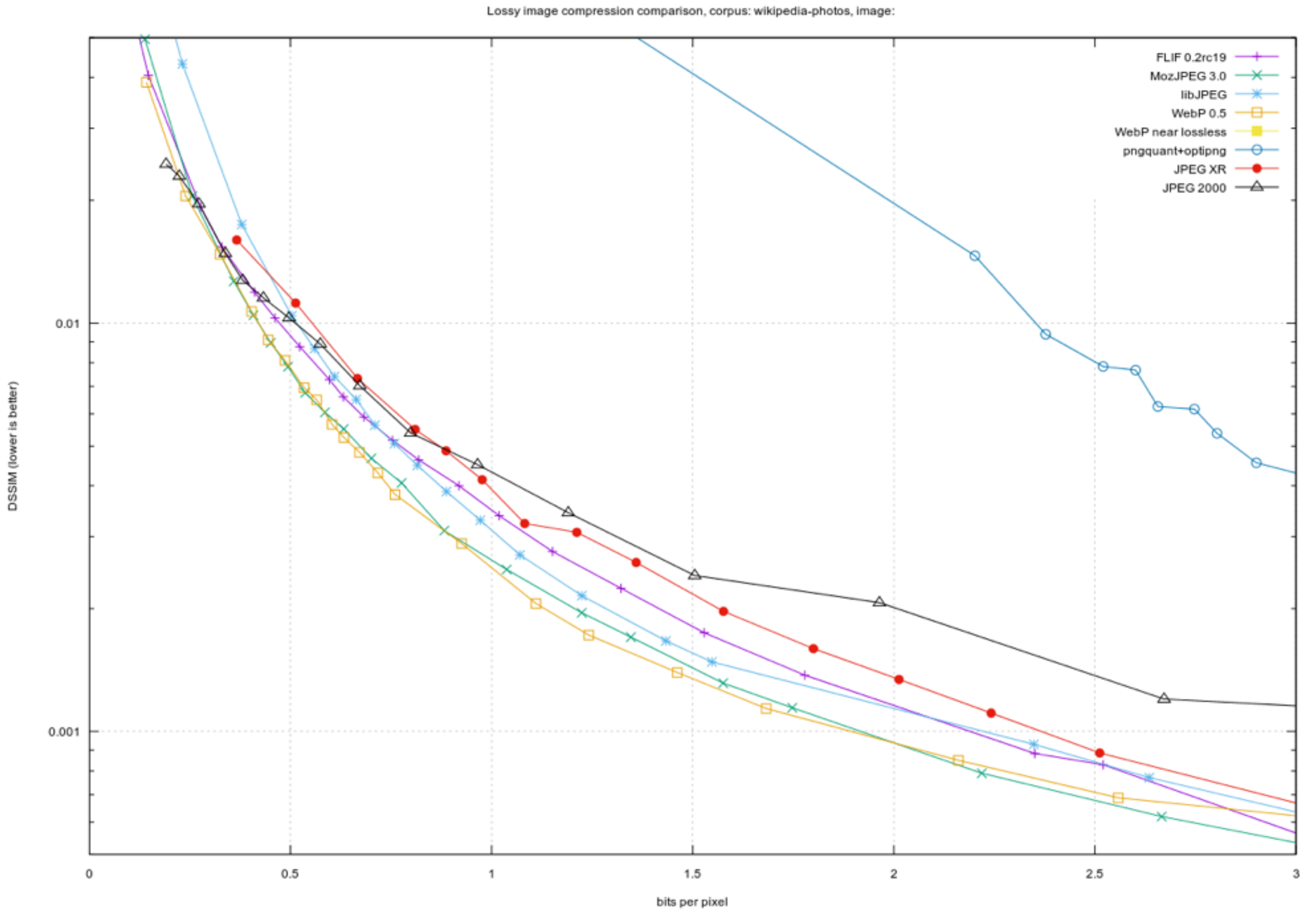
- Reference implementation of FLIF: https://github.com/FLIF-hub/FLIF
- FLIF home page: http://flif.info/
- Decoder license: Apache 2.0

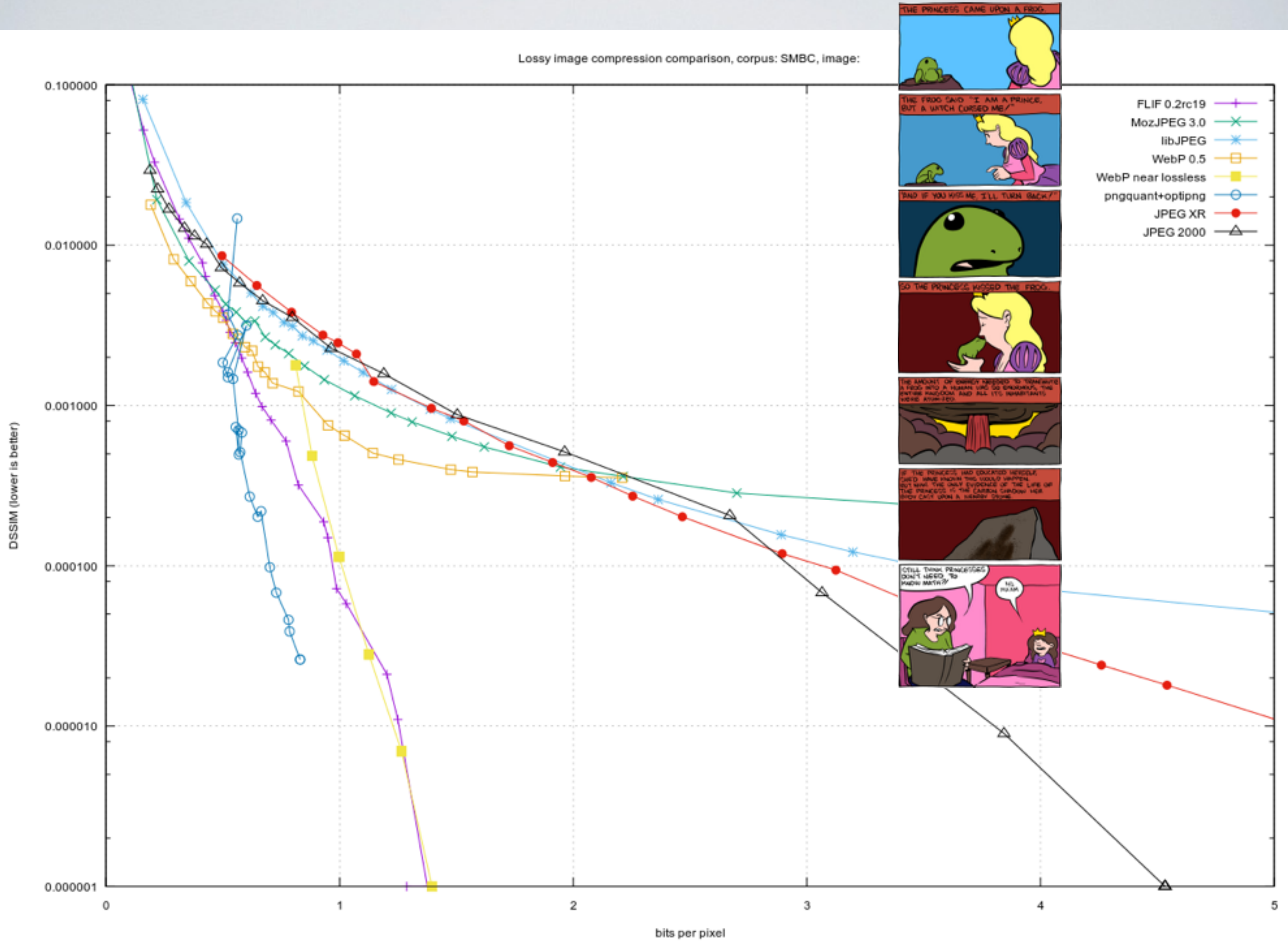
Encoder license: LGPLv3

jon@cloudinary.com



Lossy image compression comparison, corpus: wikipedia-photos, image:
FLIF 0.2rc19
MozJPEG 3.0
libJPEG
WebP 0.5
WebP near lossless
pngquant+optipng
JPEG XR
JPEG 2000
DSSIM (lower is better)
bits per pixel
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
0
2
4
6
8
10
12


Lossy image compression comparison, corpus: wikipedia-photos, image:
FLIF 0.2rc19
MozJPEG 3.0
libJPEG
WebP 0.5
WebP near lossless
pngquant+optipng
JPEG XR
JPEG 2000
0.01
0.001
DSSIM (lower is better)
0
0.5
1
1.5
2
2.5
3
bits per pixel


Lossy image compression comparison, corpus: SMBC, image:
DSSIM (lower is better)
bits per pixel
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
0
1
2
3
4
5
FLIF 0.2rc19
MozJPEG 3.0
libJPEG
WebP 0.5
WebP near lossless
pngquant+optipng
JPEG XR
JPEG 2000
THE PRINCESS CAME UPON A FROG.
THE FROG SAID "I AM A PRINCE, BUT A WITCH CURSED ME!"
"AND IF YOU KISS ME, I'LL TURN BACK!"
SO THE PRINCESS KISSED THE FROG.
STILL THINK PRINCESSES DON'T NEED TO KNOW MATH?
NO MA'AM