



Benchmarking AI Image Codec against Conventional Image Codecs

Ryan Lei, Media Foundation, Meta

VQEG Fall Meeting, Dec 15-19 2025

Motivation

- Evaluate the compression efficiency and compute complexity of different image codecs
- Understand the status of the NN based image codecs
- Benchmarking framework and script for image codec evaluation

Encoders Under Evaluation

VVC (VTM 23.11)	Reference software for Rec. ITU-T H.266 ISO/IEC 23090-3 Versatile Video Coding (VVC)
AV2 v10.0	Next gen codec currently under development from AOM
SVT-AV1 v-3.1.0	AV1-compliant software encoder library from Intel SVT
Libaom 3.12.0	AV1 reference software encoder library from AOM
JPEGGLI v0.11	JPEG encoder/decoder compatible with libjpeg62
NNVC-v14	The NNVC (Neural Network-based Video Coding), developed by the Joint Video Experts Team (JVET) based on the Versatile Video Coding (VVC) reference software, VTM.
JPEGAI	Reference software for Rec. ITU-T T.840.1 ISO/IEC 6048-1 JPEG AI learning-based image coding system (JPEG-AI)
DCVC-RT	Developed by Microsoft; presented at CVPR 2025

Image Data Set

Dataset	Number of Images	Color range	Comments
AV2-CTC	58	Full range Originally in y4m file 720p to 8K	From test video sequences used in the Common Test Condition (CTC) for the AV2 video codec standard development. We will use class F1 and F2 which contain medium to high resolution still images in y4m files (image info)
CLIC2025	32	Limited range Originally in PNG file ~2K resolution	From the Challenge on Learned Image Compression (CLIC), 1.8 ~ 3.3M Pixels (image info)
JPEGAJ	50	Limited range Originally in PNG file 720p to 2K	Variable sizes from 1M ~ 50M Pixels (image info)
mucped23-originals	49	Limited range Originally in PNG file 512x512	test dataset used by the jpegli evaluation (image info)

Encoding Configurations

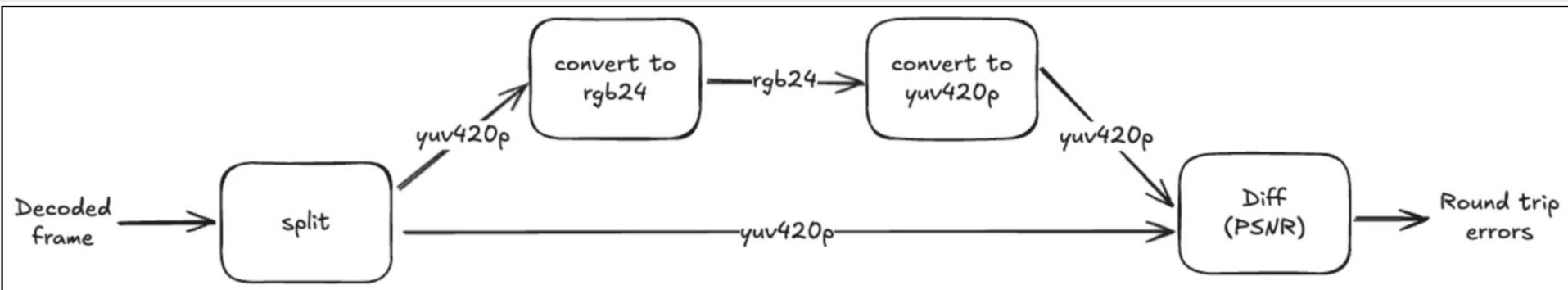
Encoder	Compression Level (QP / CRF / Qindex, etc)	Input File Format
VVC	[17, 22, 27, 32, 37, 42, 47, 52, 57, 62]	YUV / Y4M, width and height has to be multiple of 8
AV2	[60, 85, 110, 135, 160, 185, 210, 235]	YUV / Y4M
SVT-AV1, Preset [0, 7, 10]	[7, 15, 23, 31, 39, 47, 55, 63]	YUV / Y4M
Libaom Preset [0, 7]	[15, 23, 31, 39, 47, 55, 63]	YUV / Y4M
JPEGLI	[45, 55, 65, 75, 85, 95] (high value means better quality)	PNG
NNVC	[17, 22, 27, 32, 37, 42, 47, 52, 57, 62]	YUV / Y4M, width and height has to be multiple of 8
JPEG-AI	[12, 25, 50, 75, 100] (bpp x 100)	PNG
DCVC-RT	[15, 23, 31, 39, 47, 55]	YUV

Images are carefully converted to YUV and PNG file to make sure color range and color formats are preserved.

Benchmarking framework and script:

https://gitlab.com/AOMediaCodec/SVT-AV1/-/tree/master/test/benchmarking?ref_type=heads

YUV to RGB Conversion Deep Dive



Command	Round Trip Error *
<code>ffmpeg -i <input.mp4> -filter_complex "[0:v] split [v0][v1]; [v0] format=rgb24 [v0rgb]; [v0rgb] format=pix_fmts=yuv420p:color_ranges=tv [v0yuv]; [v0yuv][v1] psnr=stats_file=/tmp/test.txt" -f null -</code>	mse_avg:0.87,mse_y:1.29, mse_u:0.05, mse_v:0.02 psnr_avg:48.71, psnr_y:47.01, psnr_u:60.83, psnr_v:65.73
<code>ffmpeg -i <input.mp4> -filter_complex "sws_flags=lanczos+accurate_rnd+full_chroma_int:sws_dither=none:param0=5; [0:v] split [v0][v1]; [v0] format=rgb24 [v0rgb]; [v0rgb] format=pix_fmts=yuv420p:color_ranges=tv [v0yuv]; [v0yuv][v1] psnr=stats_file=/tmp/test.txt" -f null -</code>	mse_avg:0.00 mse_y:0.00 mse_u:0.00 mse_v:0.00 psnr_avg:75.36 psnr_y:75.11 psnr_u:81.82 psnr_v:73.47
<code>ffmpeg -y -i <input.mp4> -filter_complex "sws_flags=in_color_matrix=bt709:in_range=tv:out_color_matrix=bt709:out_range=tv:flags=lanczos+accurate_rnd+full_chroma_int:sws_dither=none:param0=5; [0:v] split [v0][v1]; [v0] format=rgb24 [v0rgb]; [v0rgb] format=pix_fmts=yuv420p:color_ranges=tv [v0yuv]; [v0yuv][v1] psnr=stats_file=test3tv.txt" -f null -</code>	mse_avg:0.00 mse_y:0.00 mse_u:0.00 mse_v:0.00 psnr_avg:inf psnr_y:inf psnr_u:inf psnr_v:inf

*Tested with lady.y4m image in AV2 CTC dataset

Average BDRATE against JPEGLI

**BD-Rate Summary Table
(vs jpegli speed 0)**

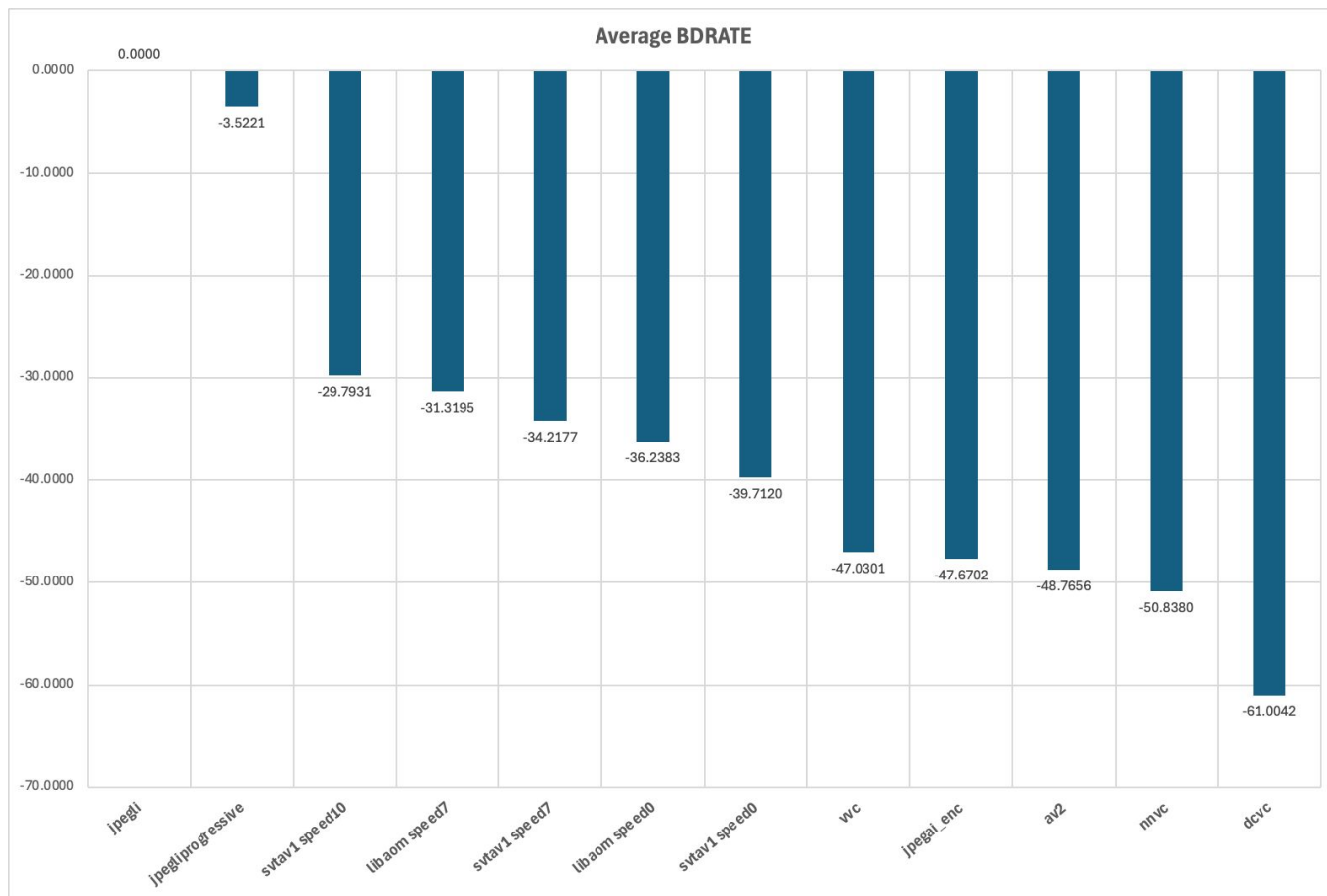
Encoder/Speed	SSIMULACRA2	ms_ssim	psnr	psnr_cb	psnr_cr	ssim	vmaf	vmaf_neg	avg
av2 (speed=0)	-47.28%	-46.62%	-63.18%	-77.41%	-65.57%	-37.12%	-50.19%	-48.19%	-48.77%
dcvc (speed=0)	N/A	-60.20%	-70.56%	-87.19%	-76.23%	-61.42%	-57.21%	-55.63%	-61.00%
jpegai_enc (speed=0)	-46.10%	-49.84%	-47.53%	-61.07%	-44.88%	-43.04%	-54.23%	-45.28%	-47.67%
jpegli (speed=0)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
pegliprogressive (speed=0)	-3.52%	-3.56%	-3.57%	-3.25%	-3.24%	-3.55%	-3.47%	-3.46%	-3.52%
libaom (speed=0)	-32.30%	-34.64%	-52.30%	-67.64%	-47.72%	-22.51%	-39.09%	-36.58%	-36.24%
libaom (speed=7)	-27.75%	-30.32%	-48.77%	-66.14%	-45.79%	-17.32%	-33.03%	-30.72%	-31.32%
nnvc (speed=0)	-46.78%	-49.33%	-64.94%	-74.58%	-58.82%	-42.05%	-51.86%	-50.06%	-50.84%
svtav1 (speed=0)	-35.83%	-37.53%	-55.83%	-63.70%	-45.40%	-27.92%	-41.66%	-39.50%	-39.71%
svtav1 (speed=7)	-29.63%	-32.63%	-51.67%	-59.15%	-38.05%	-22.02%	-35.76%	-33.60%	-34.22%
svtav1 (speed=10)	-24.93%	-28.56%	-48.00%	-55.19%	-32.47%	-16.99%	-31.15%	-29.12%	-29.79%
vvc (speed=0)	-43.64%	-45.49%	-61.66%	-70.13%	-53.66%	-37.82%	-47.71%	-45.87%	-47.03%

* Avg column is calculated as the average across PSNR, SSIM, MS-SSIM, VMAG, VMAF-NEG, SSIMULACRA for Y plane only.

** DCVC result only includes mupced23 dataset with 512x512 images

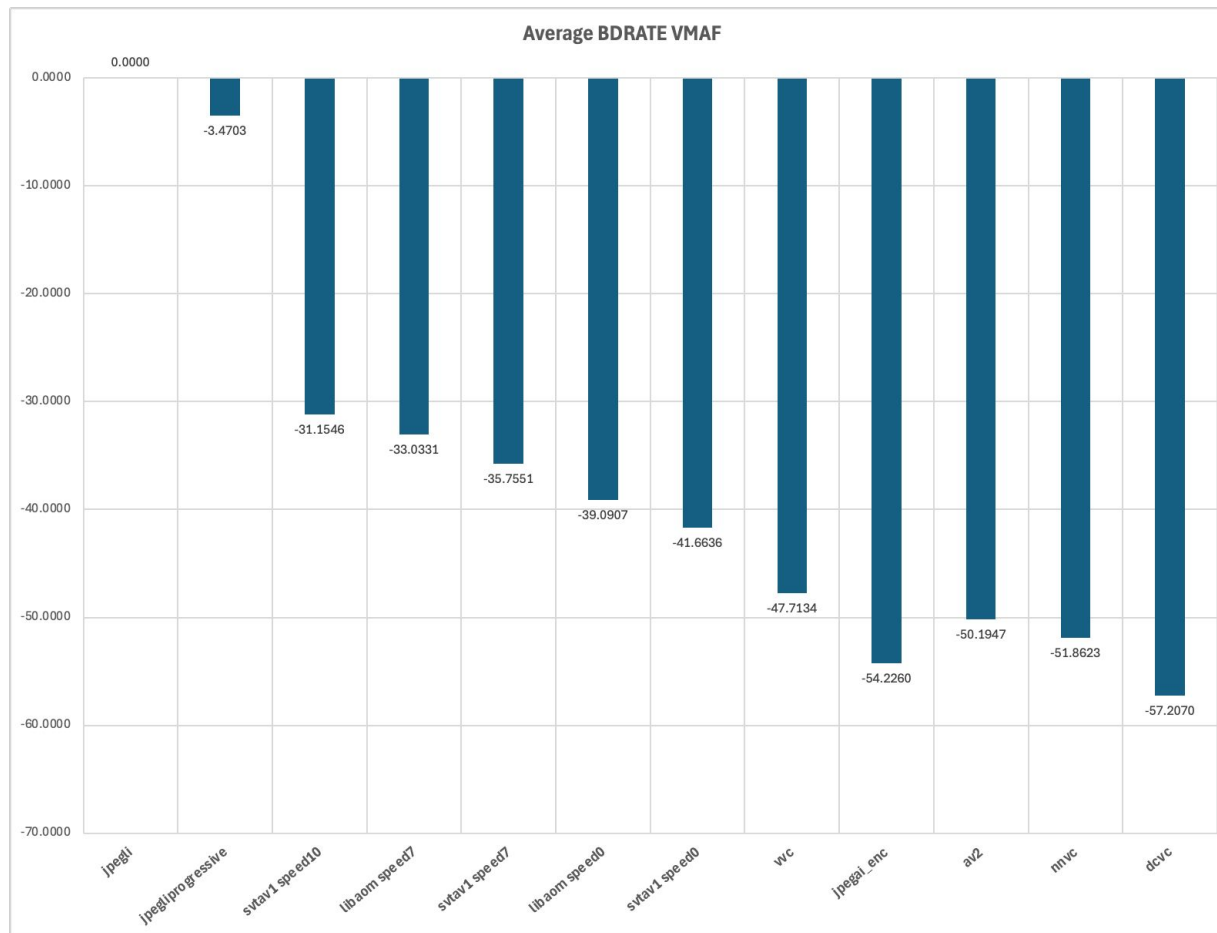
*** libvmaf is used for PSNR, SSIM, VMAF calculation, <https://github.com/cloudinary/ssimulacra2> is used for ssimulacra

Compression Efficiency Result



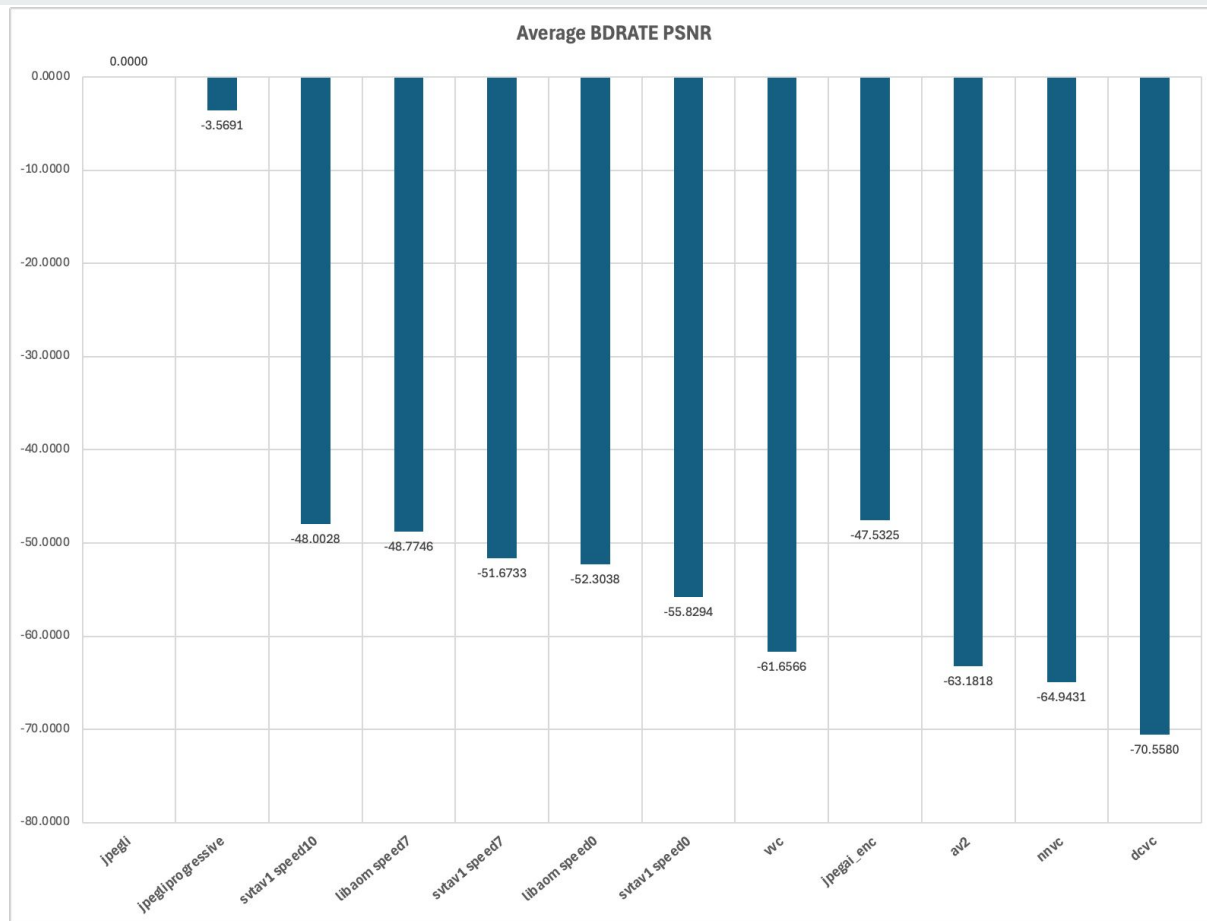
* DCVC result only includes mucped23 dataset with 512x512 images

Compression Efficiency Result



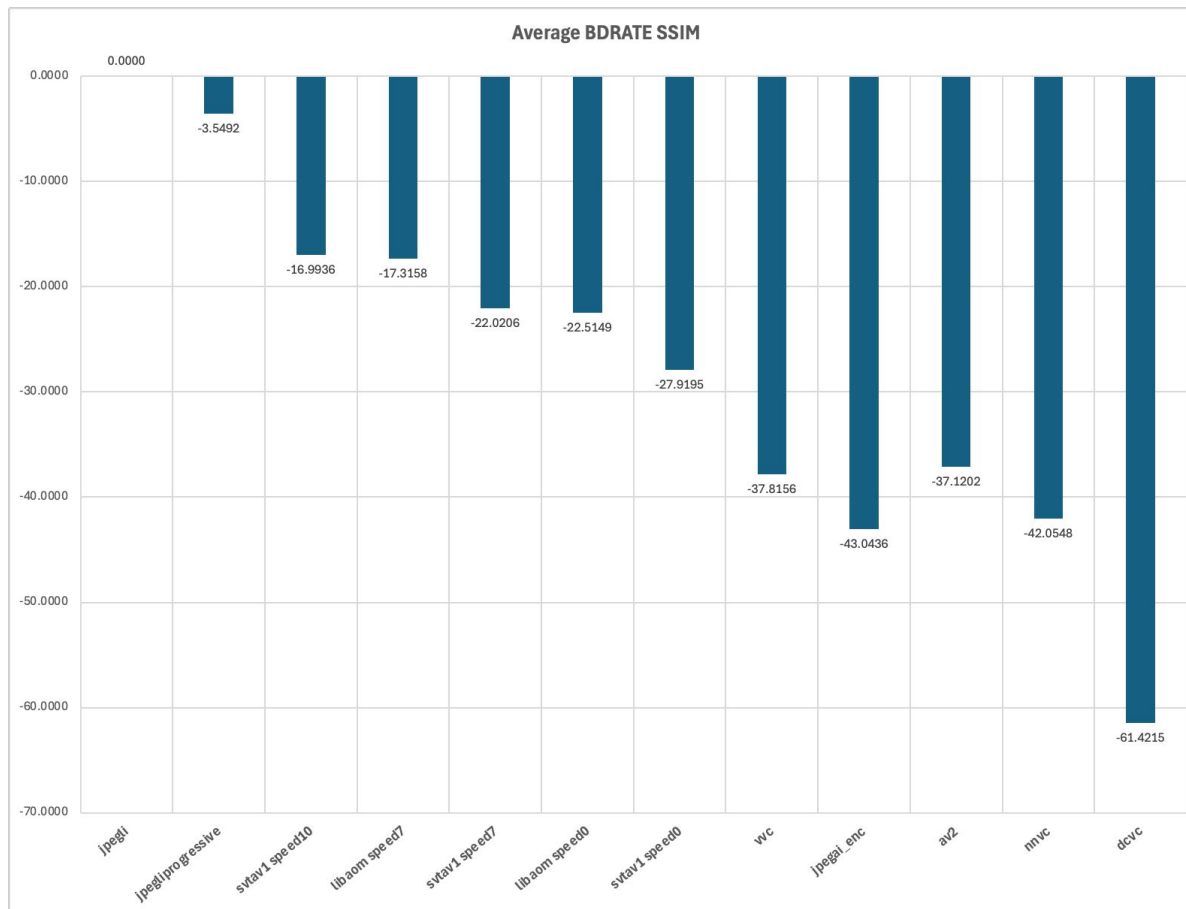
* DCVC result only includes mucped23 dataset with 512x512 images

Compression Efficiency Result



* DCVC result only includes muced23 dataset with 512x512 images

Compression Efficiency Result



* DCVC result only includes muced23 dataset with 512x512 images

Issues with SSIM implementation

**BD-Rate Summary Table
(vs jpegli speed 0)**

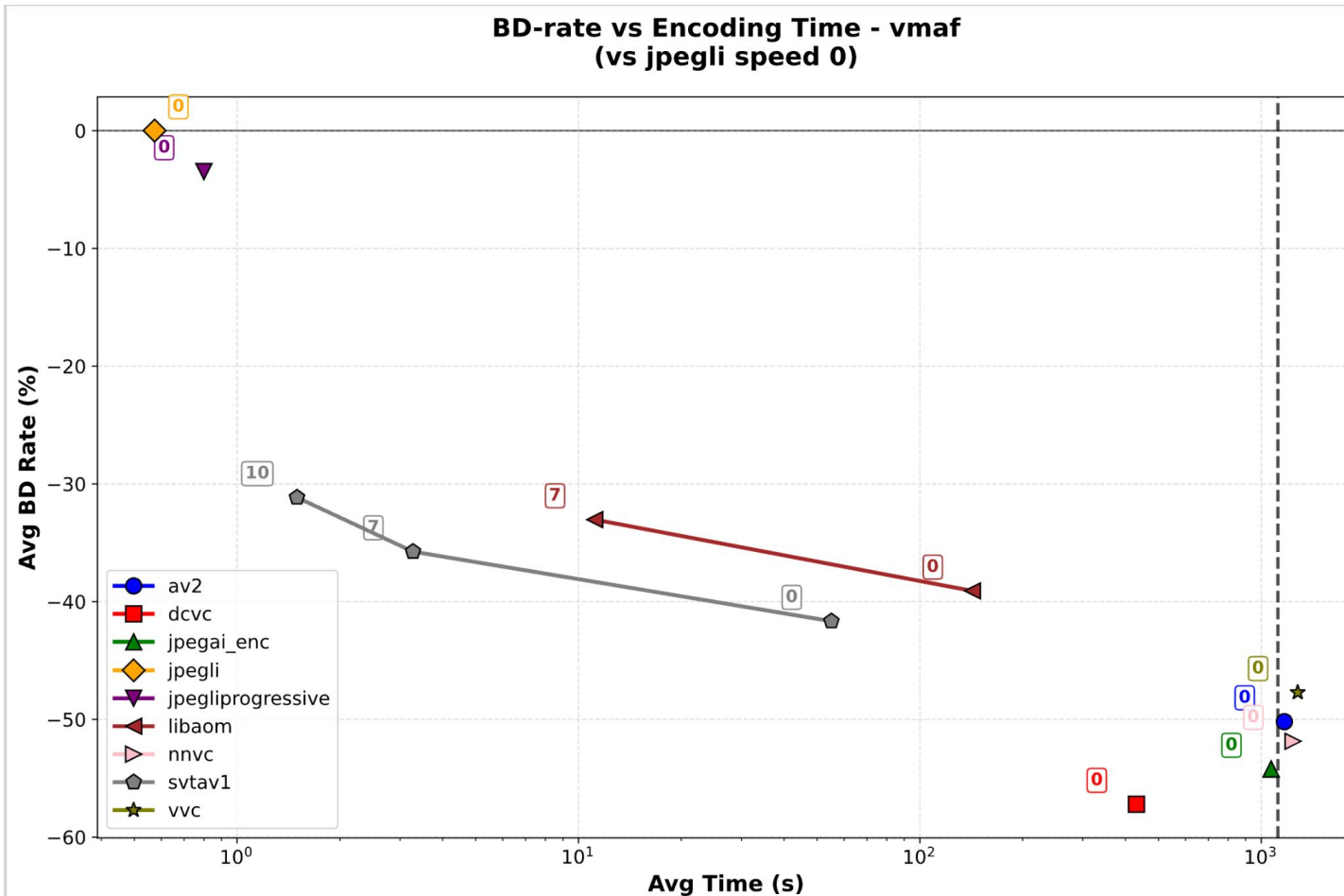
Encoder/Speed	SSIMULACRA2	ms_ssim	psnr	psnr_cb	psnr_cr	ssim	vmaf	vmaf_neg	avg
av2 (speed=0)	-47.28%	-46.62%	-63.18%	-77.41%	-65.57%	-37.12%	-50.19%	-48.19%	-48.77%
dcvc (speed=0)	N/A	-60.20%	-70.56%	-87.19%	-76.23%	-61.42%	-57.21%	-55.63%	-61.00%
jpegai_enc (speed=0)	-46.10%	-49.84%	-47.53%	-61.07%	-44.88%	-43.04%	-54.23%	-45.28%	-47.67%
jpegli (speed=0)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
pegliprogressive (speed=0)	-3.52%	-3.56%	-3.57%	-3.25%	-3.24%	-3.55%	-3.47%	-3.46%	-3.52%
libaom (speed=0)	-32.30%	-34.64%	-52.30%	-67.64%	-47.72%	-22.51%	-39.09%	-36.58%	-36.24%
libaom (speed=7)	-27.75%	-30.32%	-48.77%	-66.14%	-45.79%	-17.32%	-33.03%	-30.72%	-31.32%
nnvc (speed=0)	-46.78%	-49.33%	-64.94%	-74.58%	-58.82%	-42.05%	-51.86%	-50.06%	-50.84%
svtav1 (speed=0)	-35.83%	-37.53%	-55.83%	-63.70%	-45.40%	-27.92%	-41.66%	-39.50%	-39.71%
svtav1 (speed=7)	-29.63%	-32.63%	-51.67%	-59.15%	-38.05%	-22.02%	-35.76%	-33.60%	-34.22%
svtav1 (speed=10)	-24.93%	-28.56%	-48.00%	-55.19%	-32.47%	-16.99%	-31.15%	-29.12%	-29.79%
vvc (speed=0)	-43.64%	-45.49%	-61.66%	-70.13%	-53.66%	-37.82%	-47.71%	-45.87%	-47.03%

~10% difference between BDRATE-SSIM and BDRATE-MS-SSIM for AV2, libaom, NNVC, VVC, SVT-AV1
BDRATE-SSIM and BDRATE-MS-SSIM is very close for DCVC, JPEGLI is very close

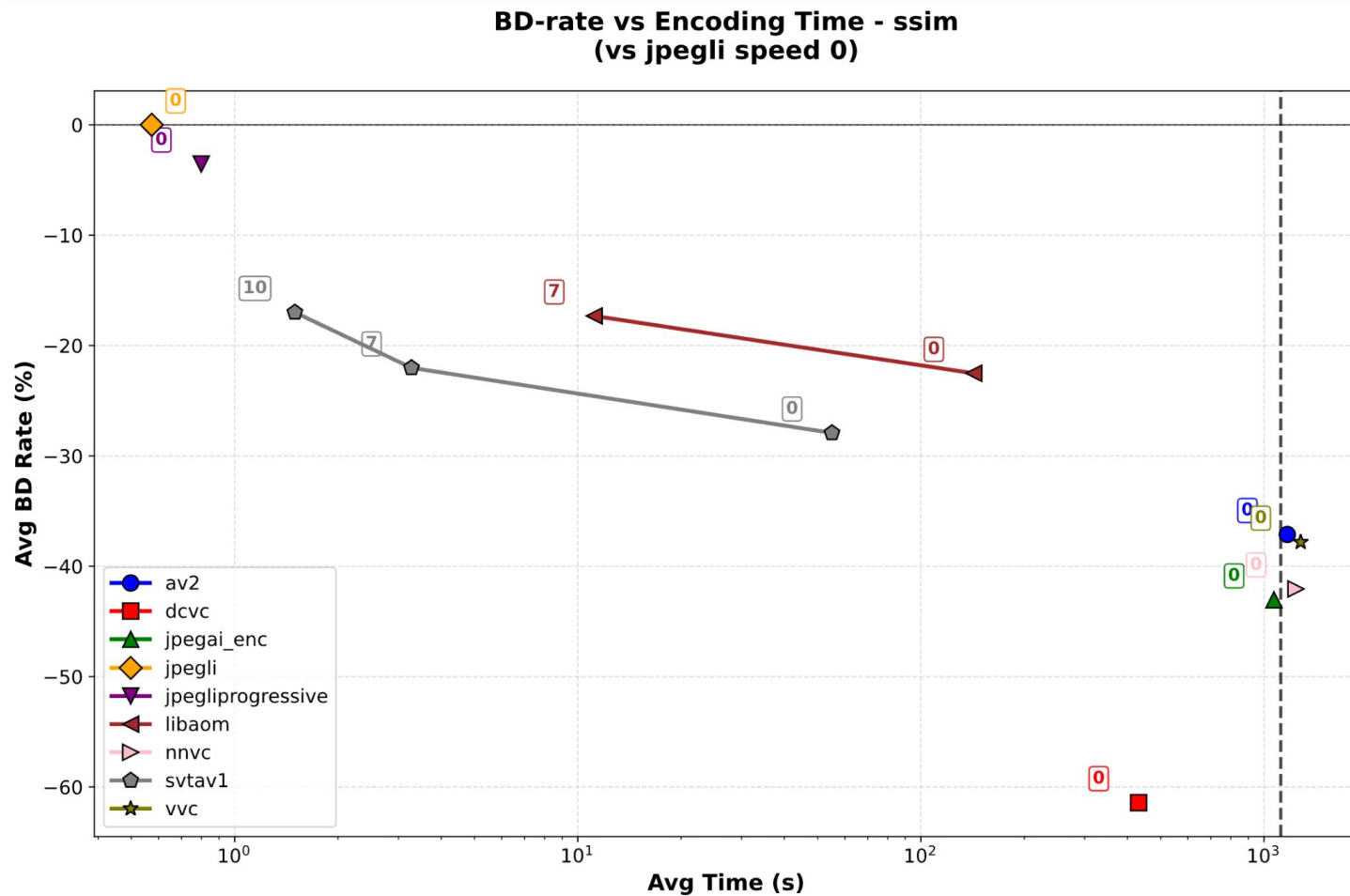
Issues with SSIM and MS-SSIM implementation

- libvmaf implementation of SSIM followed the “suggested usage” outlined in <https://ece.uwaterloo.ca/~z70wang/research/ssim/>, in which encoded image is downsampled to 256 - 384 resolution and then calculating SSIM
- So high frequency details are filtered out and codecs with higher coding gain at high resolution image are penalized.

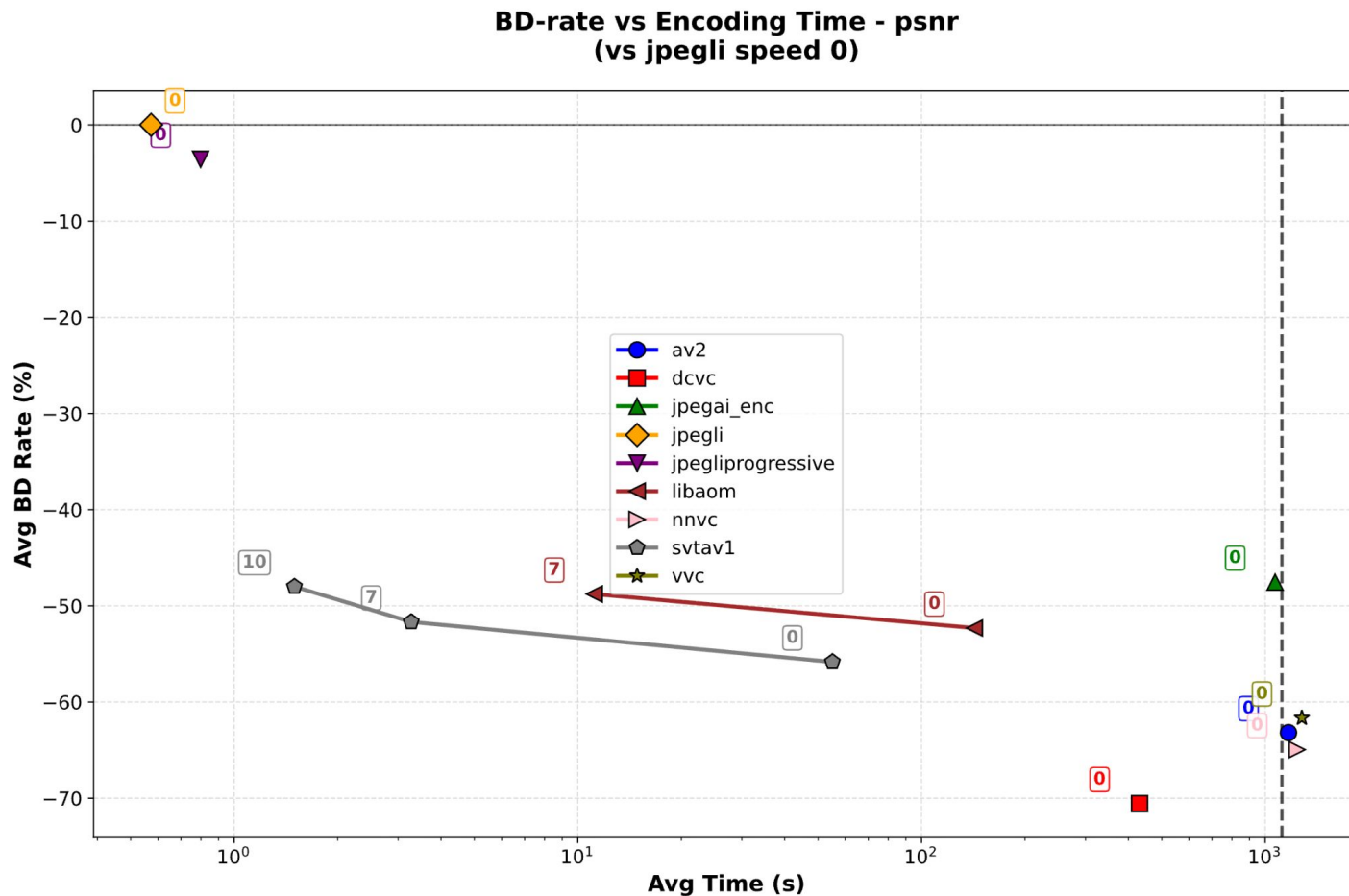
Compression Efficiency vs Complexity Result



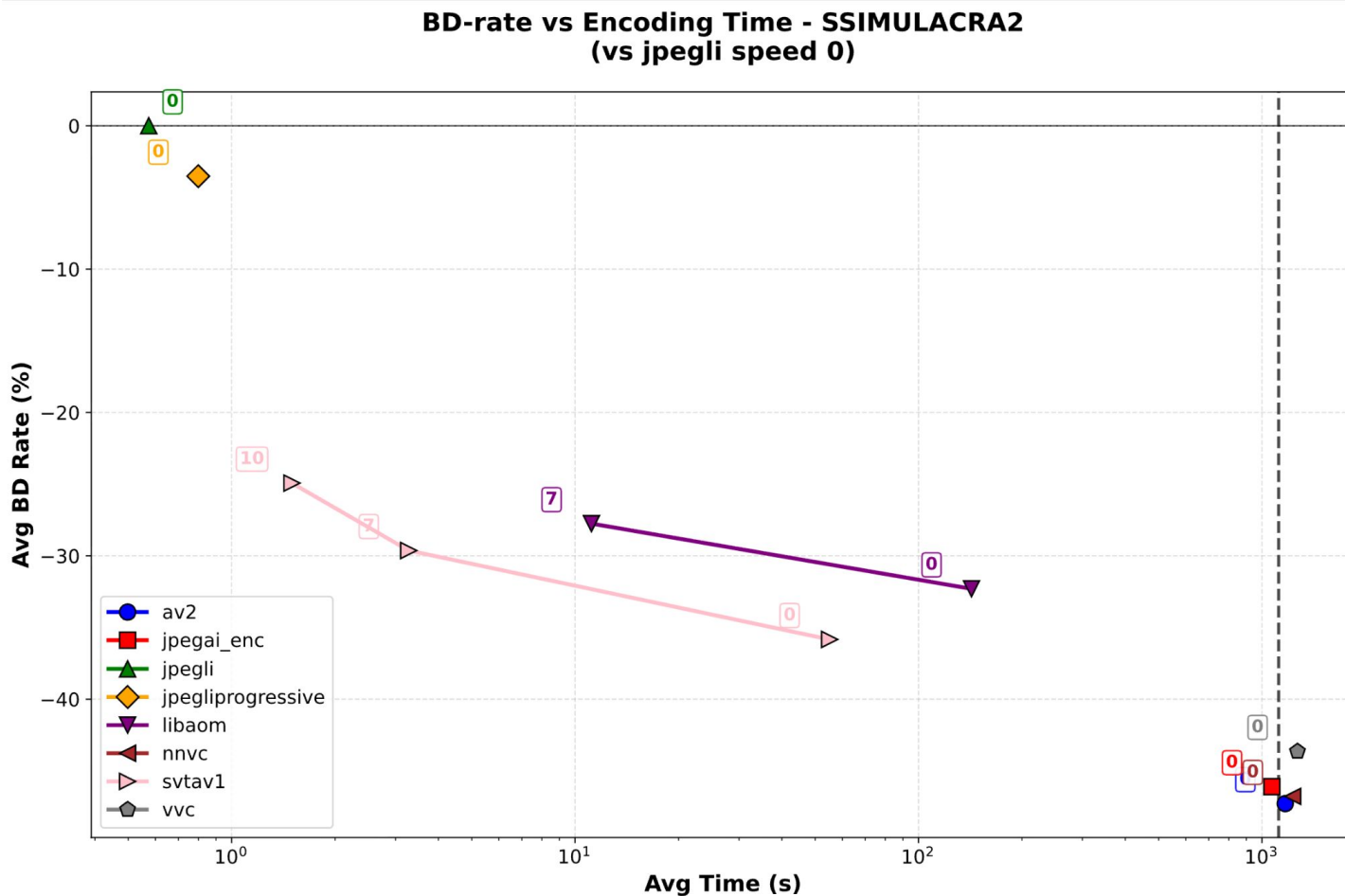
Compression Efficiency vs Complexity Result



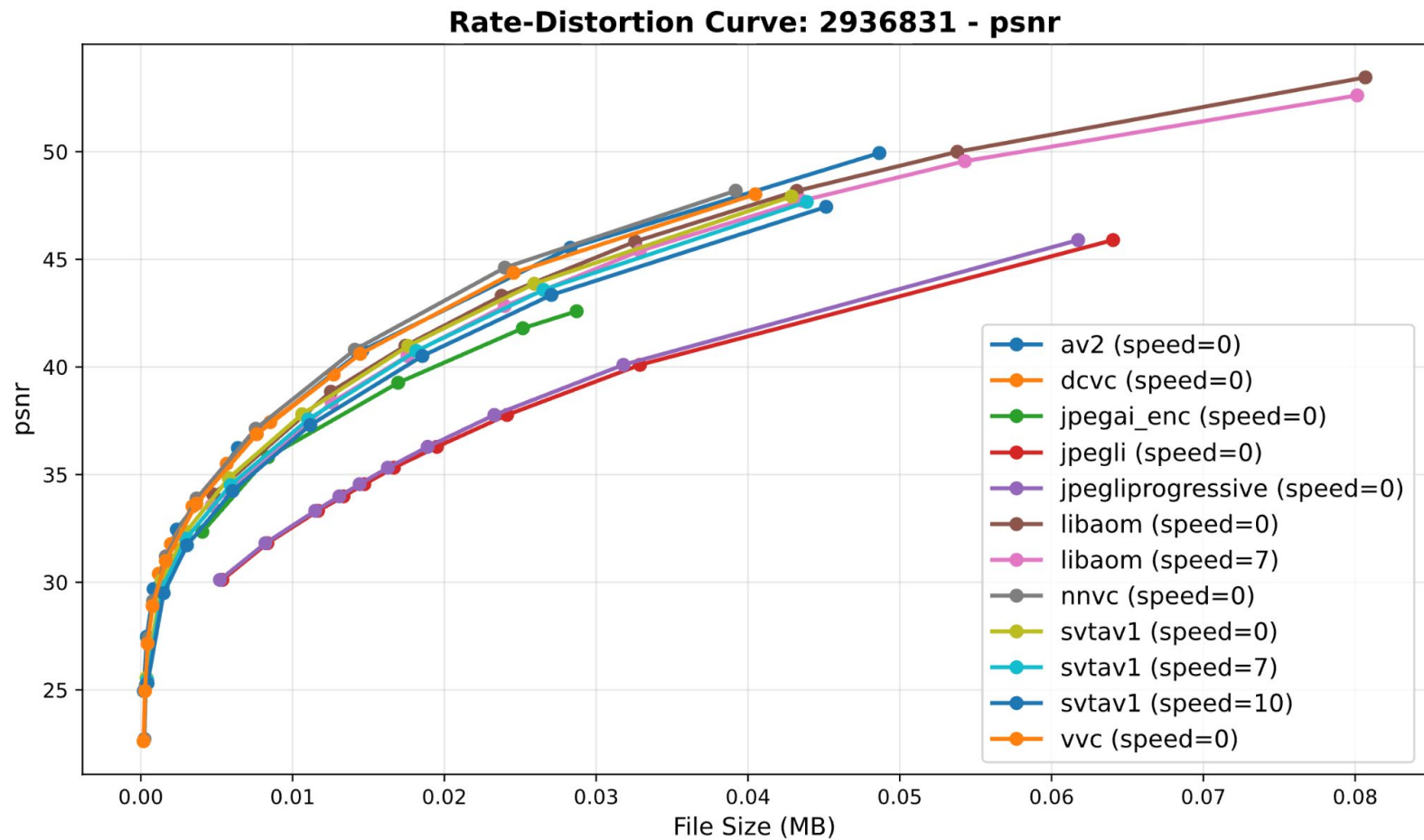
Compression Efficiency vs Complexity Result



Compression Efficiency vs Complexity Result



Example Image RD Curve



Example Images



original, 512x512



av2-speed0-qp185, psnr=29.69dB, filesize=888 Bytes

Example Images



svt-av1-speed0-qp55, psnr=30.08dB, filesize= 1443 Bytes



dcvc-q15, psnr=30.39dB, filesize=1242 Bytes

Example Images



vvc-qp42, psnr=30.99dB, filesize=1730 Bytes



nnvc-qp42, psnr=31.20dB, filesize=1727 Bytes

Example Images



svt-av1-q23, psnr=40.97dB, filesize=18473 Bytes



dcvc-q55, psnr=39.65dB, filesize=13322 Bytes

Example Images



av2-qp110, psnr=40.7633dB, filesize=15306 Bytes



nnvc-qp27, psnr=40.7993dB, filesize=14781 Bytes

Issues with DCVC

- Encoding and decoding has to be done on the same platform to avoid artifacts.
- Model seems does not work for some of the high resolution images

Issues with DCVC

Original Source

Decoded Result

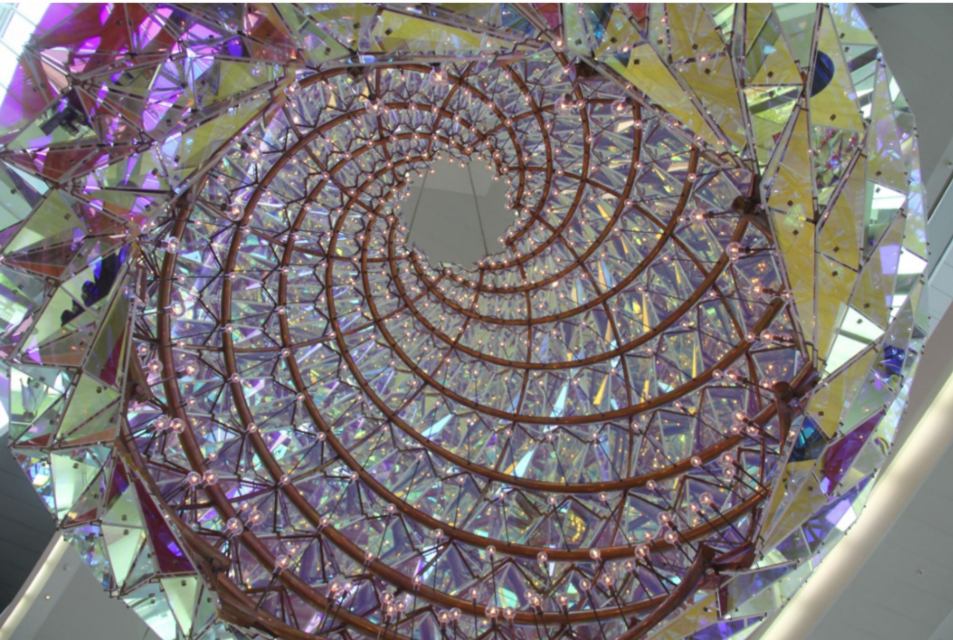


buildings_02 (4032x3024)
dcvc | QP15

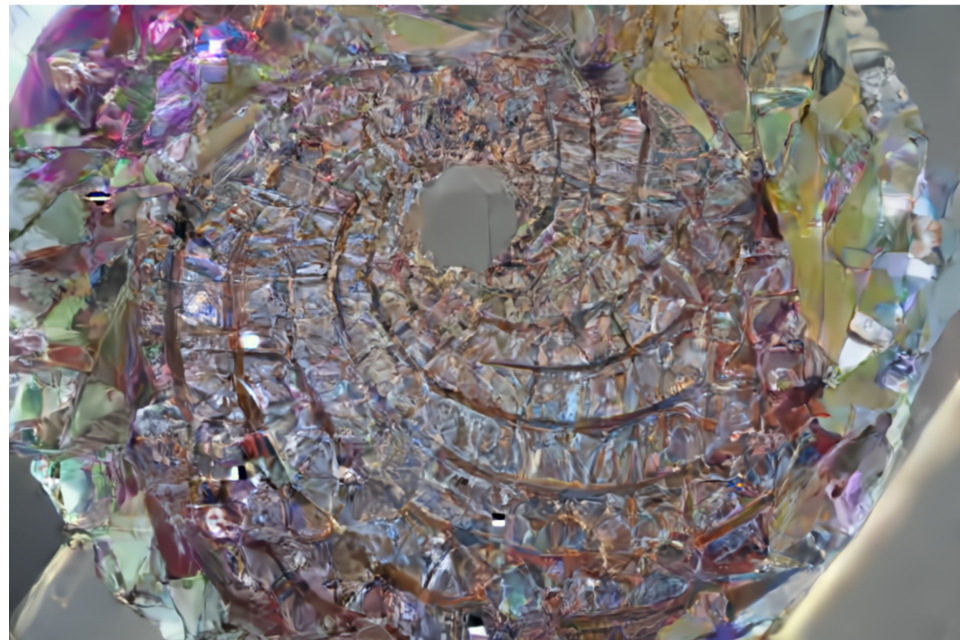
SSIM2: -64.51 | PSNR-Y: 15.22 | VMAF: 5.52

Issues with DCVC

Original Source



Decoded Result



OperaLamps (1296x864)
dcvc | QP23

SSIM2: -283.32 | PSNR-Y: 17.29 | VMAF: 4.06

Issues with DCVC

Original Source



Decoded Result



buildings_03 (4032x3024)
dcvc | QP15

SSIM2: -408.36 | PSNR-Y: 12.97 | VMAF: 6.70

Issues with DCVC

Delta (Difference)

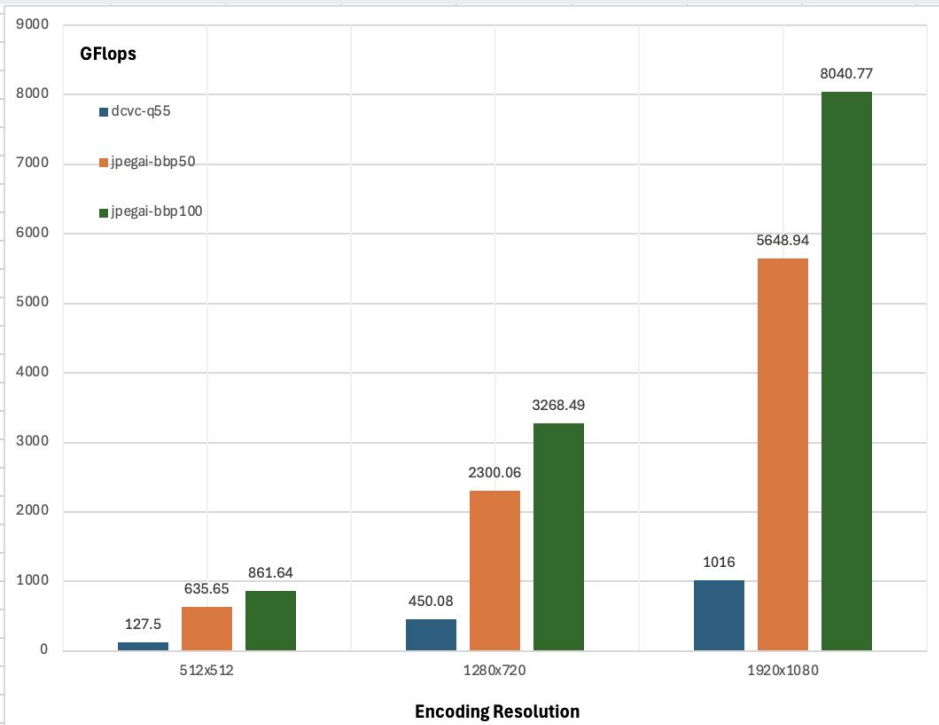
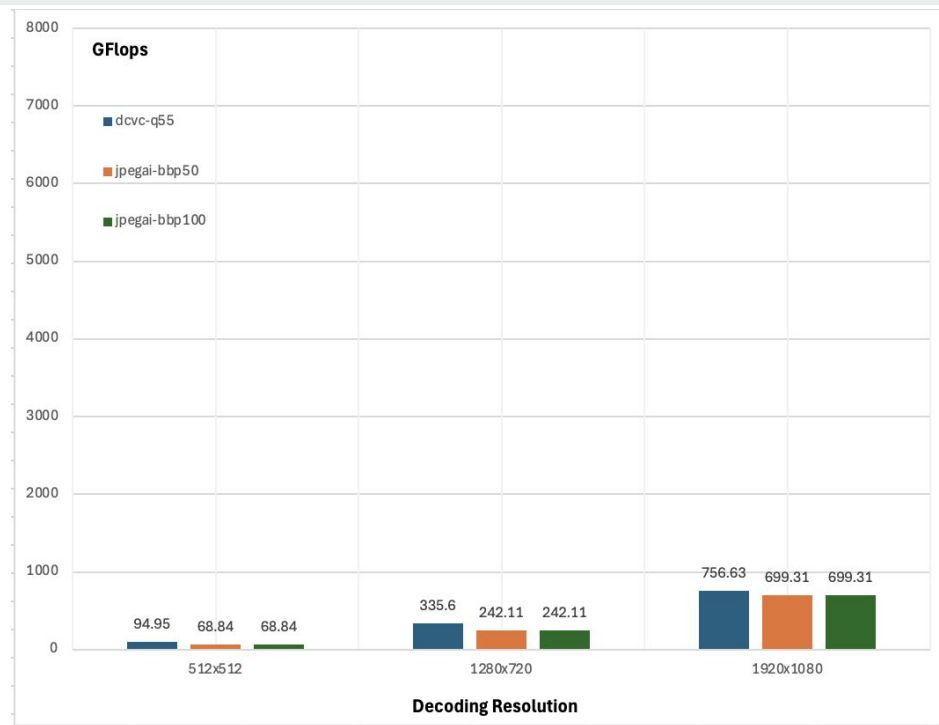


buildings_03 (4032x3024)

dcvc | QP15

SSIM2: -408.36 | PSNR-Y: 12.97 | VMAF: 6.70

Performance of JPEGAI and DCVC



- Encoding and decoding complexity of DCVC is almost the same.
- Encoding complexity of jpegai is significantly higher than DCVC
- Decoding complexity of jpegai is slightly cheaper than DCVC
- It should be noted that jpegai encoder adapts based on the complexity of the image, the numbers shown are captured using an image of random (gaussian) noise; using actual images these numbers seem to be increasing.

Conclusions

- End to end learning based image codecs, such as JPEGAI, DCVC, shows promising compression efficiency improvement (~5-10%) over SOTA conventional codes such as VVC, AV2.
- Compute complexity, especially decoding complexity, of end to end learning based image codecs is still quite high for actual production.
- Besides compression efficiency and compute cost, there are still many remaining issues, such as deterministics across platforms and issues with high resolution images, need to be resolved for learning based image codecs.
- Conventional quality metrics used for benchmarking codecs do not work well for AI based codecs.