Validation Report: Deep Render Codec Testing and Evaluation Methodology

Author:

Jan Ozer, Streaming Learning Center

Publisher:

Streaming Learning Center
www.streaminglearningcenter.com
May, 2025

Introduction and Overview

I've been following the work and progress of the AI-based Deep Render codec, with multiple interviews and product demonstrations. In these meetings, Deep Render made aggressive claims regarding encoding and decoding performance and comparative quality. This includes 22 fps 1080p30 encoding and 69 fps 1080p30 decoding on an M4-based Apple Mini, and a 45% BD-Rate compared to SVT-AV1 (all in this video).

Recently, Deep Render provided me with access to its codec so I could evaluate the accuracy of these claims. This included access to a version of FFmpeg configured with the Deep Render encoder and decoder, and a version of VLC Player that incorporated Deep Render playback, and source and files encoding using the Deep Render, SVT-AV1, x265, and VvenC codecs.

The TL/DL is this. The Deep Render codec is currently configured only for low-latency real-time use cases. If you configure the SVT-AV1 for this use case, based upon subjective testing performed by Vittorio Baroncini from VABTech UK, and encodes that I confirmed, Deep Render delivers the claimed 45% BD-Rate advantage over SVT-AV1. It also delivers the promised encoding and decoding performance.

Looking solely at VMAF and adding x265 and VvenC to the mix using the real-time low latency configuration, I also confirmed that the Deep Render codec delivers a positive BD-Rate advantage over SVT-AV1 and x265, while trailing the efficiency of VVenC by about 14%.

Beyond this, Deep Render has achieved remarkable, multi-faceted integration for a new codec of any kind, much less an AI-based codec. Not only does it work seamlessly in FFmpeg and VLC Player, it also encodes and decodes quite efficiently on a very common computing platform.

Deep Render is still a work in process, and the tested use case is limited. Deep Render needs to progress with VOD and other live use cases, and there's much testing to be done to explore how its NPU-based decoding will perform in the wild. There's no doubt that Deep Render has made very substantial progress to date, and—kudos to the management team—that they substantially delivered on the early claims made about quality and performance.

Work Performed

In this engagement I functioned more as an auditor than a test designer and implementor. Specifically, I validated subjective and objective metric-based quality testing performed by Deep Render and VABTech UK. This means confirming that the command strings, encoding parameters, and resulting files used in their evaluation were consistent, appropriate, and reproducible, in the context of low-latency, real-time applications.

Deep Render provided:

- All encoded files for all tested codecs that were used to create the subjective and objective reports and the command strings used to create them.
- VMAF scores.
- VMAF-based BD-Rate computations and RD-Curves developed for internal Deep Render testing.
- Executables for x265 (4.1.54-fa2770934), SVT-AV1 (v2.3.0), VVenC, using vvencFFapp, build 1.12.1, and the Deep Render codec (FFMPEG v1.0.0).
- A report entitled; Report of the formal subjective assessment test conducted on the Deep Render encoder by Vittorio Baroncini (VABTech UK). You can download the report here.
- Other supportive documentation.

My analysis incorporated the following steps:

1. Assessed and documented the command strings used across all codecs.
2. Applied those settings to a random sampling of test clips to confirm the outputs used in the VMAF comparisons and subjective assessment.
3. Re-computed the VMAF metric on encoded files to confirm Deep Render's results.
4. Verified that the metric results were used correctly to compute RD-curves and BD-Rate deltas.
5. Performed spot checks comparing encoded files to source, visually and via metrics, to confirm scoring trends and validate subjective results.

This approach allowed us to verify that the data used to support Deep Render's claims was produced under consistent, repeatable conditions across all codecs.

About this Report

Deep Render commissioned this validation report. All testing, analysis, and conclusions were independently conducted and authored by Jan Ozer of the Streaming Learning Center.

Test System: Apple Mac Mini (2024, M4)

We performed all encoding and decoding on a 2024 Apple Mac mini equipped with the M4 system-on-chip (SoC). The M4 integrates:

- **CPU**: 10-core configuration with four performance cores and six efficiency cores.
- **GPU**: Integrated 10-core GPU supporting hardware-accelerated ray tracing.
- **Neural Engine (NPU)**: 16-core Neural Engine capable of up to thirty-eight trillion operations per second.
- **Memory**: 16 GB of unified LPDDR5X memory with 120 GB/s bandwidth.
- **Storage**: 256 GB SSD.

Test Clips

Deep Render used a group of test clips called the Diverse Dataset that includes twenty curated 1080p, 8-bit, 24-60 fps video clips spanning five content categories:

- **Social media**: Drawn from the YouTube SFV+HDR dataset, representing popular categories like Animals, Cooking, Dance, Gaming, Health, and Sports.
- **Webcam**: Sourced from Microsoft's VCD dataset, containing talking-head sequences typical of video conferencing.
- **Gaming**: Captured in-house from 4K YouTube gameplay and trailer content, down sampled to 1080p.
- **Animation**: Also curated from YouTube 4K trailers, including titles like *Monsters Inc*, down sampled to 1080p.
- **Cinematic**: From the MCL-JCV dataset, featuring structured sequences such as *Kimono1*, *PeopleOnStreet*, and *ToddlerFountain*.

This mix provides a representative challenge set for evaluating real-time, low latency encoding across both synthetic and natural content types. More information about the clips are available upon request (and you can view encoded versions using the comparison app discussed in Section 3.

1. Command Line Analysis - Encoder Alignment

The first task was to ensure that Deep Render deployed a consistent and fair set of encoding strings for the tested codecs for both subjective and VMAF-based evaluations over the Diverse Dataset.

Subjective comparisons included only Deep Render and SVT-AV1, while metric comparisons also included x265 and VVenC.

As discussed above, Deep Render compared the codecs using a real-time, low latency encoding mode. With all codecs, this meant configuring settings that minimize latency, such as enabling low-latency modes while disabling lookahead, B-frames, and other potential sources of latency. Deep Render also rigorously standardized encoding configurations to ensure an apples-to-apples comparison. For this reason, Deep Render configured all encodes to:

- Maintain 1920x1080 resolution at the source frame rate.
- Disable B-frames and adaptive GOP structures.
- Use fixed QP or CQP-based rate control.
- Disable adaptive quantization and psycho-visual tuning where possible and enable "tuning" for metrics where available for metrics testing.
- Match keyframe behavior with long GOPs or `gop=0` equivalents, though with files all under 10 seconds, this made little difference.

The goal of our analysis was to assess whether the encoding configurations were consistently applied and to verify that the encoding configurations deployed produced the files used for comparative testing.

Deep Render

We verified all Deep Render encodes using a Deep Render-supplied version of FFmpeg using clip-specific variations of the following command string:

```
ffmpeg-dr -i input.y4m -vcodec drcodec -qp 43 -tune psnr -pix_fmt
yuv420p -g 0 -y output.mp4
```

Low Latency: `-g 0` disables periodic keyframes, reducing GOP-induced latency.

Tuning: The `-tune psnr` flag disables psycho-visual optimizations and prioritizes metric performance. Deep Render also produced a set of clips with this disabled to optimize for subjective comparisons against SVT-AV1.

Preset: Notably, the Deep Render codec only has a single preset that is enabled by default.

Performance: The Deep Render codec leverages the M4's NPU for encoding and decoding, and was the only codec accelerated by hardware in these tests. Testing at 1080p30, the Deep Render codec rendered approximately 22 fps, or 0.738 real-time (Figure 1). Testing at 720p30 yielded 40 fps, achieving faster than real-time performance. Note that none of the codecs output 1080p30 in real time using the tested configuration.

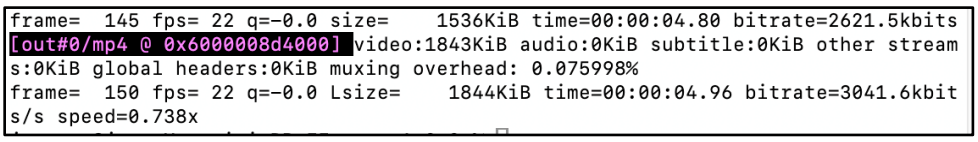

Figure 1. Encoding performance with a 30 fps YUV input file.

This command string produced an .MP4 that we could play in a Deep Render supplied version of VLC player. To compute metrics, we converted the .MP4 file to Y4M using the same Deep Render supplied FFmpeg version.

Decoding Performance

To evaluate decoding performance, we converted the MP4 file produced by the Deep Render encoding into a Y4M file stored into RAM, which minimized the impact of storing the large Y4M file to disk. In these tests, the Deep Render codec achieved 2.58x, or around 77.4 fps, well over the claimed 69 fps.

Not surprisingly, all tested videos also played at full frame rate when tested in the Deep Render capable VLC Player (Figure 2),

Figure 2. Deep Render achieved seamless playback in the VLC Player.

SVT-AV1

Deep Render produced the SVT-AV1 encodes using the open-source SvtAv1EncApp that they also supplied to us. Deep Render configured the SVT-AV1 encodes as follows:

```
SvtAv1EncApp --rc 0 --preset 7 --qp 35 -i input.yuv --width 1920 --
height 1080 --fps-num 30 --fps-denom 1 --keyint -1 --pred-struct 1 --
lp 1 --fast-decode 2 --tune 1 --aq-mode 0 --passes 1 -b output.ivf --
frames 150 -o output.yuv
```

Low-Latency Match: Deep Render configured SVT-AV1 for low-latency operation by disabling periodic keyframes (`--keyint -1`), enforcing a flat prediction structure (`--pred-struct 1`), disabling scene-change detection, and limiting thread usage (`--lp 1`). Fast-decode features were enabled (`--fast-decode 2`) to approximate the decoder optimizations present in Deep Render's mode.

Tuning and Metric Consistency: Both Deep Render and SVT-AV1 were primarily tuned for PSNR to produce metric-driven outputs. In SVT-AV1, this was set with `--tune 1`. Rate control was disabled (`--rc 0`) and encoding was conducted at a fixed QP with no adaptive quantization (`--aq-mode 0`) for consistency.

Preset 7 provides a middle-of-the-road quality and complexity tradeoff. While lower (slower) presets might have delivered higher quality, preset 7 encodes produced approximately 9 frames per second on the test system, significantly slower than real-time for 30 fps video (See Figure 3).

Arguably, we could have recommended deploying a faster, lower-quality preset to match Deep Render's faster, NPU-based encoding speed.

```
Encoding
SUMMARY --------------------------------- Channel 1  ---------------------------------
Total Frames            Frame Rate              Byte Count              Bitrate
         150            30.00 fps                  5265106             8424.17 kbps


Channel 1
Average Speed:          8.965 fps
Total Encoding Time:    16732 ms
Total Execution Time:   16747 ms
Average Latency:        104 ms
Max Latency:            290 ms
janozer@jans-Mac-mini SVT-AV1 %
```

Figure 3. SVT-AV1 encoding performance using Preset 7.

Limitations on Perceptual Comparison: While Deep Render provided both PSNR-tuned and perceptually tuned encodes for subjective testing, SVT-AV1 could not be tuned for perceptual optimization (`--tune 0`) in low-delay mode (see Figure 4). As a result, SVT-AV1 encodes were evaluated only in PSNR-tuned mode. This represents a limitation when comparing subjective quality results between Deep Render and SVT-AV1.

```
Svt[warn]: Instance 1: Tune 0 is not applicable for low-delay, tune will be forced to 1.
Svt[warn]: TPL is disabled for aq_mode 0
Svt[info]: Level of Parallelism: 1
Svt[info]: Number of PPCS 3
Svt[info]: [asm level on system : up to neon_i8mm]
Svt[info]: [asm level selected : up to neon_i8mm]
```

Figure 4. Tuning for perceptual quality was not available in low-delay mode.

To be clear, we tested with both `--tune 0` and no `–tune` switch, and the files were exactly the same as the file encoded using `--tune 1` Note that this behavior is not surprising given that the default setting for tuning is `1`, so in effect, the absence of the `--tune` switch is the same as `--tune 1`.

x265

Deep Render produced the x265 encodes using the open source x265 executable and the following command string.

```
x265 --input-res 1920x1080 --input-depth 8 --fps 30 --qp 35 --bframes
0 --scenecut 0 --keyint 6000 --ref 1 --no-wpp --frame-threads 1 --
preset medium -o output.bin --input input.yuv
```

Low Latency: Deep Render achieved low-latency encoding by manually disabling B-frames, scene cuts, and wavefront parallel processing (WPP), and by setting frame threading to 1 (`--frame-threads 1`). These changes mirror what x265's --tune zerolatency preset would have done, but Deep Render applied them individually to leave tuning available for other purposes like PSNR optimization.

We confirmed that Deep Render's manual approach produced functionally equivalent output to --tune zerolatency and preserved it for consistency. Other common zero-latency optimizations (`--b-adapt 0, --rc-lookahead 0, --no-cutree`) were also tested independently but showed no impact on output quality.

Tuning: By default, x265 tunes for perceptual quality. To benchmark using objective metrics, the x265 documentation recommends tuning for PSNR (`--tune psnr`). On a subset of five test files, enabling PSNR tuning and adjusting QP to match the original file sizes increased VMAF by an average of 1.53%. If this improvement were consistent across the full dataset, the BD-Rate advantage reported for Deep Render over x265 would likely shrink by 3–5 percentage points. While not enough to affect the relative rankings, it suggests the original DR vs. x265 delta may slightly overstate Deep Render's advantage.

Encoding Performance: Deep Render encoded x265 files using the Medium preset with full CPU-based software encoding. On the Apple M4 Mac Mini testbed, x265 encoding throughput was approximately 11.7 frames per second (fps), about half the frame rate achieved by the NPU-accelerated Deep Render codec.

```
encoded 150 frames in 12.84s (11.68 fps), 2398.48 kb/s, Avg QP:34.98
janozer@jans-Mac-mini Codec Testing %
```

Figure 5. Using the Medium preset, x264 encoded at 11.68 frames per second.

VVenC

Deep Render produced the VVenC files using the open source vvencFFapp and supplied the same for our testing. Deep Render used the following command:

```
./vvencFFapp -v 1 -c vvc_faster_8bit.cfg -i input.yuv -s 1920x1080 -b
input.bin --fps=30 --preset=faster --passes=1 --poc0idr=1 --qp=26 --
qpa=0 --threads=8 -o output.yuv
```

Note that VVenC can accept commands both in the command line and config file, `vvc_fast_8bit.cfg,` shown in the command string. Those commands discussed below that aren't shown in the command string are enabled in the config file.

Low Latency: GOP size was set to 1 in the config file (`GOPSize: 1`) and `IntraPeriod: -1` ensured no periodic I-frames. Setting `--poc0idr=1` forced the first frame to be an IDR to ensure decoder compatibility. Temporal filtering (MCTF) was disabled in ultra-low latency mode to minimize encoder delay.

Tuning: VVenC does not offer an explicit PSNR tuning mode but setting `--qpa=0` disables perceptual QP adaptation, which improves objective metric scores like PSNR and VMAF. This is functionally similar to PSNR tuning in other codecs. No other psycho-visual optimizations were enabled.

Threads: VVenC was run with `--threads=8` to match the Deep Render configuration and avoid unnecessary throttling. While this setting does introduce some latency, it was a practical compromise, as VVenC's architecture requires multi-threading for any meaningful throughput.

Preset: Deep Render tested using the `faster` preset, which is the fastest available (and lowest quality). This was appropriate, given that encoding using the `faster` preset and `--threads=8` achieved a throughput of approximately 10.7 fps.

```
./vvencFFapp -v 1 -c vvc_faster_8bit.cfg -i 2_videoSRC14.yuv -s 1920x1080 -b   7
3.21s user 2.82s system 552% cpu 13.748 total
janozer@jans-Mac-mini Codec Testing % START=$(date +%s); ./vvencFFapp -v 1 -c vv
c_faster_8bit.cfg -i 2_videoSRC14.yuv -s 1920x1080 -b 2_Jan_videoSRC14_vvenc_QP3
5.bin --fps=30 --preset=faster --passes=1 --poc0idr=1 --qp=35 --qpa=0 --threads=
8; END=$(date +%s); DURATION=$((END - START)); echo "Encoding FPS: $(echo "scale
=2; 150 / $DURATION" | bc)"

Encoding FPS: 10.71
janozer@jans-Mac-mini Codec Testing %
```

Figure 6. Using VVenC's faster preset (the fastest available) produced 10.71 fps.

Testing Methodology

As described, to verify the encodes and metric calculations, we spot checked each codec's encoding string using five files, one from each genre. This meant:

- Producing the output files on the M4 testbed.
- Comparing the bitrate and VMAF score of that file to the equivalent encoded file provided by Deep Render.
- Comparing the computed VMAF score to that deployed by Deep Render.

In all cases, the output files and metric computations matched those reported by Deep Render to about +/- 1%. In many cases, the results were identical, both in the output file and the computed metric score. It's clear that the command strings shown above were used to produce the files Deep Render used in the report, and that the bitrates and VMAF scores used to compute BD-Rate comparisons are accurate.

2. BD-Rate Validation Summary

Objective: Confirm the VMAF BD-Rate findings supplied by Deep Render (Table 1) by independently verifying the underlying encoding and metric computations.

Baseline vs.	Deep Render	VVC (VVenC)	AV1 (SVT-AV1)	HEVC (x265)
Deep Render	baseline	+14.32%	-3.06%	-18.04%
VVC (VVenC)	-	baseline	-15.20%	-28.30%
AV1 (SVT-AV1)	-	-	baseline	-15.45%
HEVC (x265)	-	-	-	baseline

Table 1. BD-Rate computations provided by Deep Render.

Approach: To do this, we:

- **Spot-checked the encoded outputs** for all codecs to confirm that settings were appropriate, tuning was fair, and command lines matched the intended configurations (as described above).
- **Verified the average VMAF scores** used in the BD-Rate calculations by confirming that the input scores corresponded to the actual decoded files.
- **Independently reproduced the BD-Rate values** using Excel and the verified VMAF data, confirming curve fitting, integration, and delta calculations.

Results: Our independently computed BD-Rate results matched Deep Render's original values within ±1% across all codecs and comparisons. These minor differences fall within the expected margin for curve fitting tools and have no material impact on codec performance rankings or

overall conclusions. Aside from the discrepancy regarding not tuning x265 for PSNR, as discussed above, we consider the BD-Rate findings in the Deep Render report to be accurate and credible.

3. Subjective Evaluation Spot Check

The formal subjective assessment was conducted by Vittorio Baroncini, a respected expert in video quality evaluation. His process followed well-established international standards (ITU BT.500 and ITU-T P.910) using a DSIS (double stimulus impairment scale) protocol, with twenty-four naïve subjects across multiple test labs. The procedures, scoring, and lab setup all meet the rigor expected for professional-grade subjective evaluation.

file name	*BD*		*file name*	*BD*
Adventure_SRC09-DR_v25_FLOAT	0,67		SRC023_DR_v25_FLOAT	0,34
Adventure-SRC01_DR_v25_FLOAT	0,55		SRC0246-DR_v25_FLOAT	0,58
CSGO_Part1-DR_v25_FLOAT	0,71		SRC055-DR_v25_FLOAT	0,61
CSGO_Part2-DR_v25_FLOAT	0,66		SRC080_DR_v25_FLOAT	0,68
Franken-DR_v25_FLOAT	0,59		SRC123_DR_v25_FLOAT	0,39
Invincible-DR_v25_FLOAT	0,35		TOM-Jerry-DR_v25_FLOAT	0,40
Monster_DR_v25_FLOAT	0,67		videoSRC04_DR_v25_FLOAT	0,39
SCR2143-DR_v25_FLOAT	0,50		videoSRC11_DR_v25_FLOAT	0,71
SDR_Health_sowp_DR_v25_FLOAT	0,68		videoSRC14_DR_v25_FLOAT	0,68
SRC019_DR_v25_FLOAT	0,54		videoSRC17_DR_v25_FLOAT	0,69

Table 2. BD-Rate comparisons provided by VABTech UK.

For the record, these values average out to a Harmonic Mean of 0.5357, which translates to a BD-Rate savings of 46.43%, about 1.5 percentage points higher than claimed by Deep Render. Numbers vary slightly if you average using geometric or arithmetic mean.

To independently assess the validity of the reported results, we performed a targeted spot check of five clips, one from each content category. These included Deep Render and SVT-AV1 encodes at comparable bitrates. We reviewed the files in real-time and inspected frame-by-frame using decoded `y4m` versions. While this type of analysis has inherent limitations, including the lack of motion perception, blind evaluation, and controlled viewing environment, it still offers a useful sanity check. If something looked clearly off or the reported rankings were implausible, it would be evident.

What we found was consistent with the subjective results. Clips encoded using the two codecs were generally very close in quality, though when stressed, visible macroblock artifacts and coarse quantization began to appear in three of the five tested SVT-AV1 encodes. These artifacts were subtle and often only visible at high magnification but likely registered as distracting noise in real-world viewing. In contrast, when Deep Render started to degrade, it did so by gently smoothing detail rather than introducing visible structural errors. This approach appears to have helped maintain subjective scores.

In one illustrative example (videoSRC14 - Figure 7), blocks were visible around the subject's nose in the SVT-AV1 encode at moderate bitrate, while the Deep Render version preserved cleaner detail in that same region. We found similar blocky regions in three of the five test clips reviewed.

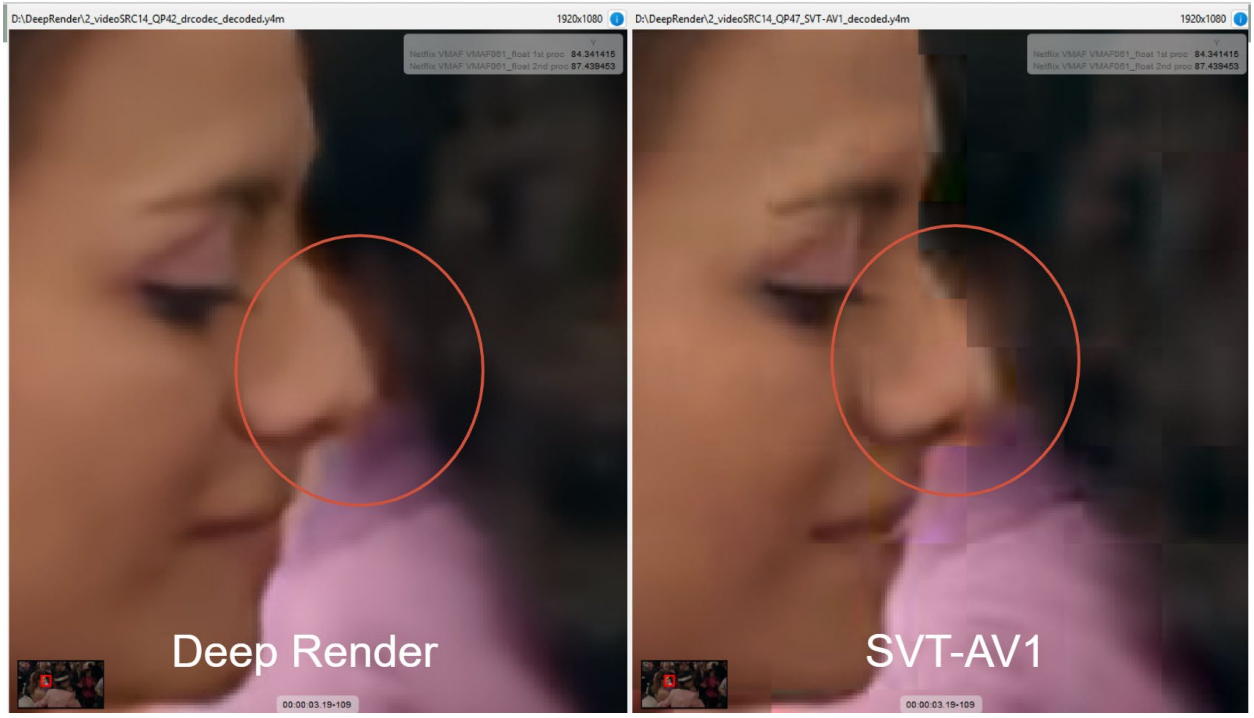

Figure 7. Visible blocks in the SVT-AV1 clip; slight blurring in the Deep Render clip.

Across all clips tested, Deep Render never appeared subjectively worse than SVT-AV1, and in many cases delivered perceptible improvements in texture, motion, or edge handling. These were much more evident in real world clips than either animation or gaming.

Deep Render Comparison App

Note that Deep Render has created a web resource that allows viewers to compare the Deep Render codec to libvpx-vp9, SVT-AV1, VvenC, and x265 using the same source clips used by VABTech UK. Located at https://eval.deeprender.ai/ (Figure 8), you can load the clips on the left.

The Deep Render encodes always appear on the right side, and you can toggle through the other codecs to display on the left side using the toggle on the upper left, choosing low, medium and high-quality representations for any codec on either side. Once loaded, you can play the video, or stop on a frame, while dragging a slider to hide one side and reveal the other, a useful way to visualize the qualitative differences between the two encodes with actually encoding yourself.

To convert the disparate codecs and formats into a sharable interface, Deep Render transcoded all clips to H.264 using CRF 17. You can right click and download clips originally encoded with the Deep Render codec, so I downloaded several to check the bitrate, which topped out at about 20 Mbps for 1080p30. I compared the H.264 version of the file to the Y4M file converted directly from the Deep Render encode, and saw very little difference, and none that was meaningful. So, the tool should give anyone interested the ability to compare Deep Render quality against the other codecs using a wide range of clips and encoding configurations.

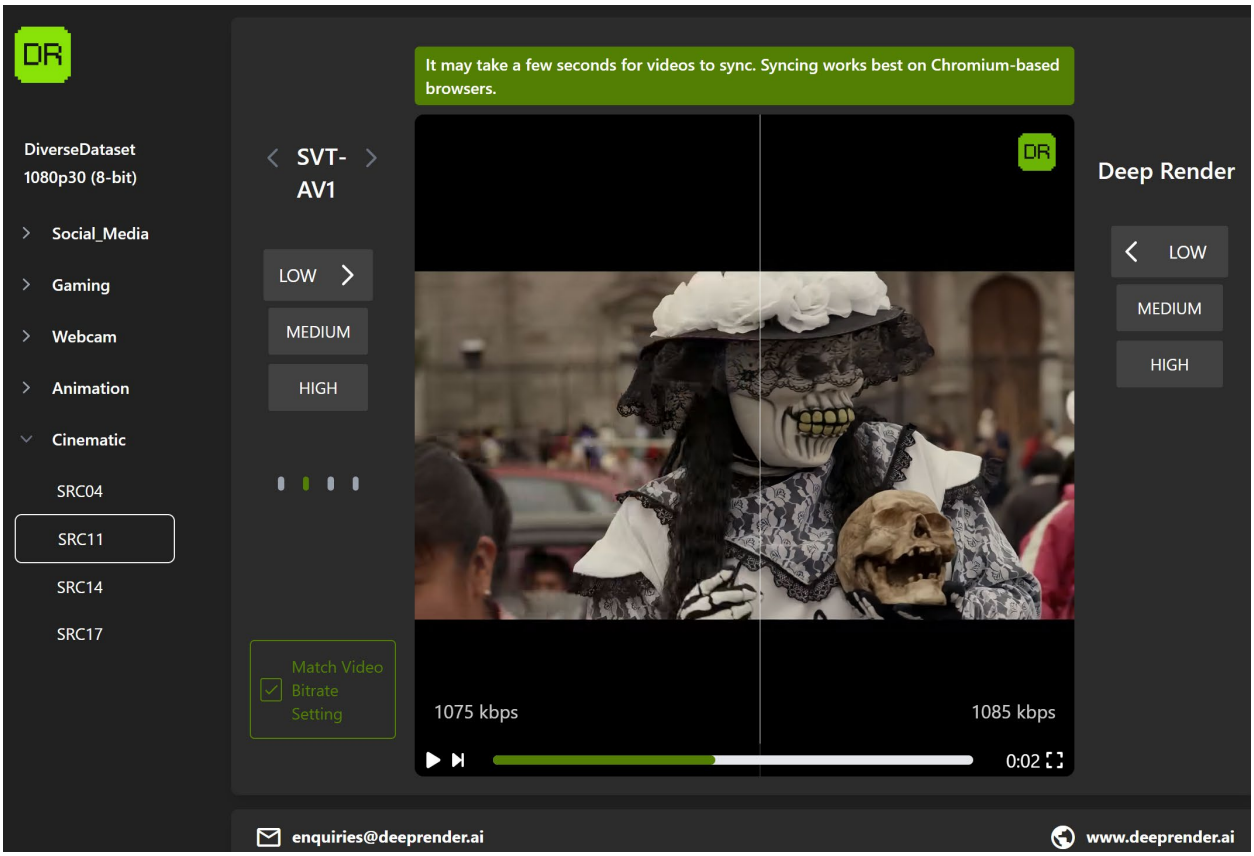

Figure 8. The Deep Render comparison app lets you visualize all the clip included in the subjective as encoded with the Deep Render and other codecs.

4. Conclusion

Our independent analysis confirms that Deep Render's testing was conducted using consistent, appropriate, and reproducible methods. Encoding parameters were fair across all codecs, and the resulting outputs were verified as functionally equivalent to the files created by Deep Render for the subjective and objective analysis. Metric computations were correct, and BD-Rate results matched our independently derived values within an acceptable ±1% margin.

The only material discrepancy we found was the omission of PSNR tuning in the original x265 tests, which, if corrected, would reduce but not eliminate the reported advantage for Deep Render.

Subjectively, while differences were often subtle, visual comparisons supported the reported MOS rankings. Deep Render typically avoided the visible macroblocking and quantization artifacts that emerged in SVT-AV1 encodes under pressure, instead exhibiting graceful degradation through minor detail loss. In no case did Deep Render appear perceptually worse.

Taken together, these findings support the integrity of Deep Render's reported results and confirm that the performance advantages claimed in their report are credible, repeatable, and grounded in methodologically sound testing.

What about the discrepancy between the 3% VMAF-based BD-Rate advantage Deep Render enjoyed over SVT-AV1 and the 45%+ subjective BD-Rate advantage? While we trust VMAF as a reliable metric for assessing encoding performance, it becomes less definitive when comparing different codecs. This is particularly true for AI codecs that introduce techniques not reflected in VMAF's training data. In cases like this, when objective and subjective results vary in degree, subjective evaluation should be considered the gold standard.

However, it's essential to note that these conclusions are specific to the context of real-time, low-latency encoding—Deep Render's current operating mode. They should not be generalized to traditional VOD encoding workflows or even non-latency-constrained live transcoding, where different tuning options and encoder features may significantly affect the comparative performance of all tested codecs.