Video codec comparison using the dynamic optimizer framework

Ioannis Katsavounidis, Liwei Guo

Ioannis Katsavounidis, Liwei Guo, "Video codec comparison using the dynamic optimizer framework," Proc. SPIE 10752, Applications of Digital Image Processing XLI, 107520Q (17 September 2018); doi: 10.1117/12.2322118

SPIE.

Event: SPIE Optical Engineering + Applications, 2018, San Diego, California, United States

Video codec comparison using the dynamic optimizer framework

Ioannis Katsavounidis and Liwei Guo

Video Algorithms - Netflix, Los Gatos, CA 95032, USA

ABSTRACT

We present a new methodology that allows for more objective comparison of video codecs, using the recently published Dynamic Optimizer framework. We show how this methodology is relevant primarily to non-real time encoding for adaptive streaming applications and can be applied to any existing and future video codecs. By using VMAF, Netflix's open-source perceptual video quality metric, in the dynamic optimizer, we offer the possibility to do visual perceptual optimization of any video codec and thus produce optimal results in terms of PSNR and VMAF. We focus our testing using full-length titles from the Netflix catalog. We include results from practical encoder implementations of AVC, HEVC and VP9. Our results show the advantages and disadvantages of different encoders for different bitrate/quality ranges and for a variety of content.

Keywords: Video encoding, perceptual video quality, video codec, dynamic optimizer.

1. INTRODUCTION

Video encoding is a complex process that involves a number of processing steps, performed on each frame of a video sequence. These steps typically involve block-based intra- or inter-prediction, followed by 2-D transform of residuals between original pixel values and their corresponding prediction, quantization of transform coefficients and entropy coding of quantized values. There are more steps involved in modern video codecs, such as deblocking filtering - essentially, an adaptive low-pass filter - of reconstructed frames before they are being used by subsequent frames as predictors. Another improvement introduced over time is the use of variable block partitioning of input frames in order to capture individual objects in a video scene. All these incremental steps over the basic video encoding structure, introduced more than 20 years ago, has allowed modern video codecs to be increasingly efficient. A high-level system diagram of a modern video encoder, showing the individual blocks discussed earlier, is presented in Fig. 1.

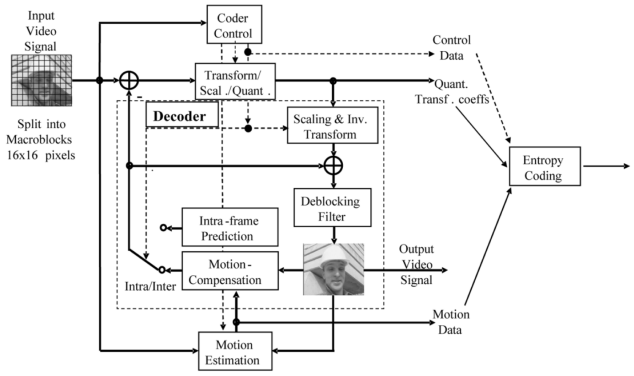


Figure 1: AVC/H.264 video encoder. Reproduced from[1]

Further author information: (Send correspondence to Ioannis Katsavounidis)
Ioannis Katsavounidis: E-mail: ikatsavounidis@netflix.com; Liwei Guo: E-mail: lguo@netflix.com

Applications of Digital Image Processing XLI, edited by Andrew G. Tescher, Proc. of SPIE Vol. 10752, 107520Q · © 2018 SPIE · CCC code: 0277-786X/18/$18 · doi: 10.1117/12.2322118

The video coding research community has produced new and more efficient encoders, with a refresh cycle of about 7 years. The most popular video codecs today are H.264/AVC,[2] H.265/HEVC[3] and VP9.[4] During the course of video codec standardization, video researchers have been using a software implementation of what is called "reference encoder". The corresponding reference encoders for the 3 video coding standards mentioned earlier (AVC, HEVC and VP9) are called "JM",[5] "HM"[6] and "Libvpx",[7] respectively.

In addition to these reference encoders, the open-source community, as well as commercial entities, have been actively working to productize video encoding by offering practical alternatives as software packages. Such practical implementations of video encoders are being used by the vast majority of companies that need video encoding - especially those that don't have real-time constraints.

Testing methodology and associated test vectors play an important role in codec standardization, especially to evaluate the advantages of certain coding tools. These are usually referred to as "common test conditions" and they allow groups of researchers to reproduce each others' results, in order to develop appropriate tools that improve performance of previous video codecs. For consistency, the same testing methodology and test vectors have also been used by practically every researcher in the field of video coding, when publishing their results in this field.

Although different from one codec to another, the testing methodology used to compare codecs or encoder configurations can be described in the following general steps:

- Each test sequence is encoded at the input source frame resolution only
- A set of fixed - independent of the test vector - quality parameter (QP) values are chosen and together with the source resolution are used to produce encodes for each test vector
- The quality of each encode is calculated with respect to the reference input video sequence using the pixel-based Peak-Signal-to-Noise-Ratio (PSNR) fidelity metric
- Together with the associated bitrate of each encode, a set of (R,Q) points are produced for each test vector
- The corresponding sets from two different encoders or encoder configurations are then plotted on the (R,Q) plane and the average difference in bitrate at the same quality is reported in terms of percentage, with negative values meaning bitrate gain and positive values indicating bitrate loss
- These average bitrate percentage differences are also called "BD-rate" figures, short for Bjontegaard-delta rate[8]

When it comes to test vectors, they are typically chosen at the beginning of standardization efforts and are meant to represent the typical usage of a video codec, thus they include natural video of different types (slow motion, high detail, etc.) as well as synthetic video, representative of computer animation or computer games. These are oftentimes called "standard sequences" and they are rather well known among the video compression community. For practical reasons, all these standard sequences are rather short (at most 10 sec.) and they tend to include a single shot of visual content; an exception comes from some of the sequences contributed to the MPEG standardization body by Netflix from the "El Fuente"[9] and "Chimera"[10]open-source video sequences, which include 3-4 shots.

It is typically expected to obtain different results when one applies a certain video codec into their own application; both differences in content and operating parameters, as well as the usage of a practical - instead of the reference - encoder implementation, are sources of discrepancies between results obtained during standardization and those reported by practical uses of these codecs.

Another factor that complicates running and reporting codec comparisons has to do with the exact metric chosen to report BD-rates. Even for the case of PSNR, there is some disagreement as to which aggregation method should be used across different color planes (Y, Cb, Cr), but also temporally. The evolution of more perceptually-tuned video quality objective measures, such as SSIM,[11] VIF[12] and VMAF[13] - only to name a few - adds more metrics for video researchers to use in order to measure and report BD-rates.

There have been numerous publications on this topic of video codec comparison for different applications. In particular, the work by De Cock *et al.*[14] used VMAF for the first time as one of the key metrics to quantify encoder gains. It addition, it used clips from actual content from the Netflix catalog, as opposed to the standard sequences that most researchers had been using up to that point. It compared VP9, AVC and HEVC, using VP9's reference encoder implementation, libvpx.[7] Building upon that work, Guo *et al.*[15] extended codec comparisons to include the almost finalized - at that time - AV1 reference encoder, aomenc.[16] Most importantly, this latest work also used the dynamic optimizer framework, which made comparison among different encoder implementations more fair.

In this work, we attempt to provide first and foremost a better codec comparison testing methodology, as it relates to non-realtime encoding of entertainment video for adaptive streaming. Key components of this methodology are:

1. Allowing the use of multiple resolutions to represent each video sequence. An example would be to use 1280×720, 960×540 and 640×360 encoding resolutions, on top of the input resolution of 1920×1080, used currently, to encode a certain HD source sequence.
2. Using *scaled* quality metrics, where video sequences of lower resolution are first decoded and then up-scaled to match the original input sequence resolution. In this way, encodes of different resolution can be compared.
3. For single-shot video clips, such as the majority of standard sequences, we propose the construction of the corresponding convex hull of (R,Q) points coming from various QP and encoding resolution combinations. These convex hulls can then be used to compare encoders in the "relevant" range of QPs for each encoding resolution.
4. For videos with multiple shots, such as full titles from the Netflix catalog, we propose the use of the Dynamic Optimizer to pick the optimal path of encoding each of the shots in a longer sequence.

We hope to show that this methodology, coupled with the use and reporting of results for both PSNR and VMAF quality metrics can provide more meaningful and practical answers to the question "how much better is encoder A vs. B". Application developers can then make choices based on whether they believe PSNR or VMAF results is most reflective of what the users of their application perceive as video quality, and the type of content they intend to encode.

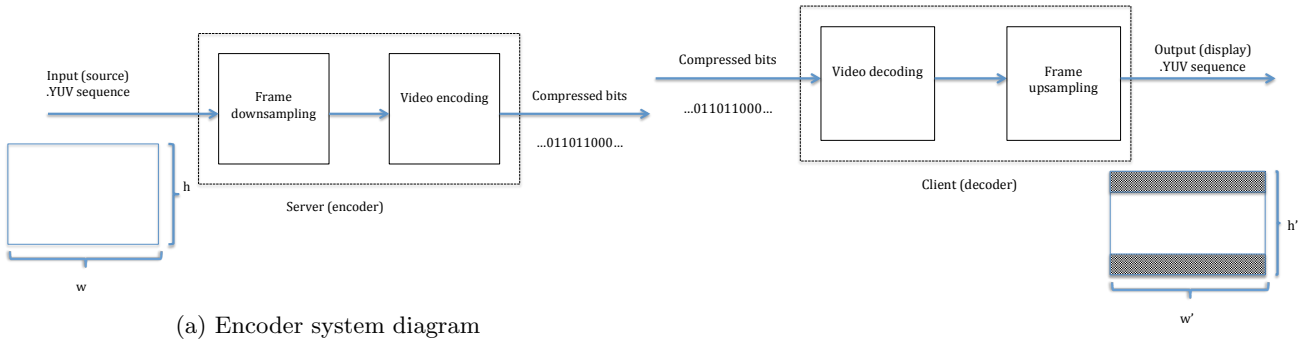
Another important contribution of this work is the introduction and usage of practical encoders in dual configuration - one geared towards PSNR-optimality and the other towards VMAF/perceptual optimality. By offering both alternatives to the dynamic optimizer, we can now reliably and with less concern quote results for both PSNR and VMAF, as mentioned earlier.

Finally, by testing with full-title sequences, we hope to make our results more representative of real-life savings when using a certain codec over another in adaptive video streaming today. Although we don't claim that our limited selection of 8 full titles from the Netflix library is representative of the entire catalog, we at least hope to give some indication about the performance of codecs relative to each other.

The rest of the work is structured as follows. In section 2, we make a brief presentation of VMAF and the Dynamic Optimizer framework. In section 3, we show details about the 3 encoder implementations we chose, followed by the testing methodology in section 4. We present our results in section 5 and conclude with a comprehensive analysis and discussion in section 6.

2. VMAF AND THE DYNAMIC OPTIMIZER

Over the past several years, video encoding has been shifting its focus to the new era of video entertainment: HTTP adaptive streaming (HAS). The ability to utilize the secure, peer-to-peer TCP/IP packet-based communication protocol opened tremendous opportunities for the deployment of video services that consumers embraced and continue to adapt all over the world. There are some fundamental differences of HAS over broadcast TV, namely:



(a) Encoder system diagram

(b) Decoder system diagram

Figure 2: Server - client video communication system. Source (w, h) and display (w', h') resolutions are typically different; they are also different from encoded video resolution.

- There is no need to deal with the difficult problem of missing bits due to transmission errors in broadcast TV; instead, one needs to devise strategies to deal with the issue of missing or reduced internet connectivity
- There is no need for consumers to store or archive video content, since they can access the content they want to watch anytime, anywhere - even though "download-and-play" is also a use-case offered by many streaming providers, such as Netflix
- The ability to dynamically switch bitrates and qualities during a streaming session allows for a nearly-optimal Quality of Service (QoS) to be attained without much interference with or help from network operators and their networks; clients can make this type of decisions independently and without much impact to other people watching the same or a different show
- The non real-time nature of over-the-top (OTT) HAS allows for a much longer processing time in preparation of the encoded bitstreams, which translates to a significant opportunity to produce higher quality streams than those offered by traditional TV broadcasters
- Broadcast TV offers a single quality to all clients, which also translates to a given, fixed choice of video codec and encoding parameters. HAS, on the other hand, offers multiple representations of the same video content, which means it can offer different codecs to different clients, based on their decoding capabilities, different resolutions and/or different bitrates depending on many parameters such as network bandwidth, client display size and service differentiation.

We show the high-level structure of an adaptive streaming server-client communication system in Fig. 2.

2.1 VMAF

Taking all the above into consideration, Netflix, together with external research collaborators, developed and eventually open-sourced a new perceptual video quality metrics called Video Multimethod Assessment Fusion (VMAF). VMAF is a machine-learning approach to fuse existing, highly efficient objective video quality metrics, called elementary metrics. In the case of VMAF, the first set of elementary metrics chosen is a multi-scale version of the image quality metric VIF, where instead of aggregating all terms coming from the 5 different analysis scales used in VIF into a single figure, they are instead kept separate, essentially providing 5 VIF quality scores, one at each resolution scale. Another feature chosen is the detail loss measure (DLM),[17] also an image quality metric which quantifies the loss of structure mostly due to blurring and compression. The final feature is called "motion" and is essentially the same as the Temporal Indicator (TI)[18] index expressing how much pixels of consecutive frames are different.

These features are fused through a Support Vector Machine (SVM) regressor[19] and a final score is produced, expressing video quality on a scale of $[0, 100]$, with 0 indicating really poor and 100 nearly perfect video quality.

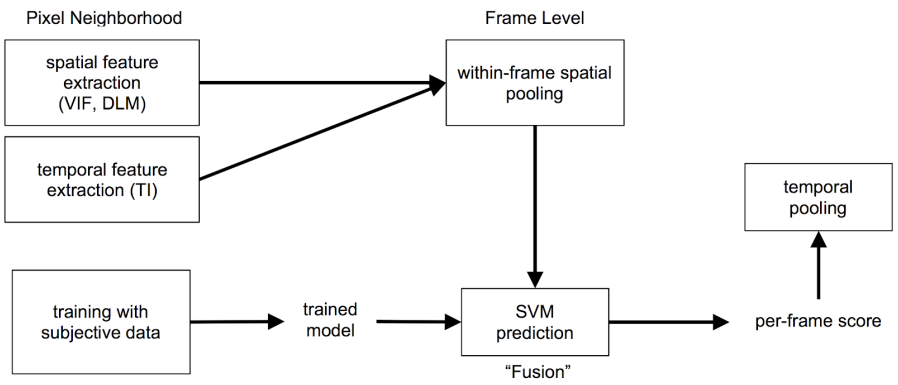


Figure 3: VMAF system diagram

The actual parameters used in the SVM regressor of VMAF have been obtained through collection of subjective testing from viewers who were asked to score the quality of distorted and reference video clips presented to them on a video monitor. It's important to note and always keep in mind the following facts about VMAF.

- VMAF is a full-reference quality metric, i.e. it requires the presence of both the original source and the distorted version in order to produce a quality score
- Both in the generation of video sequences used for subjective testing, and in subsequent application of VMAF, only two types of distortions are taken into consideration - scaling and compression
- When two encoded versions of the same original source are obtained using different encoding resolutions, one needs to consider the *scaled* VMAF value of each one of them. In this case, a pair of filters is used to first down-sample the original source to the desired encoding resolution and, following decoding of the encoded clip, to upsample it back to the original source resolution.

A high-level system diagram of VMAF is shown in Fig. 3

Although far from perfect, it has been shown by both internal and external studies that VMAF correlates really well with human perception of video quality, in particular within the context of adaptive streaming, where the only two types of distortion present are scaling and compression artifacts. VMAF is an open-source project and researchers are encouraged to use it but also to contribute back to it, as part of the continuous effort to improve it and cover new use-cases. We refer interested readers to the Netflix tech-blog describing VMAF.[20]

2.2 Dynamic Optimizer

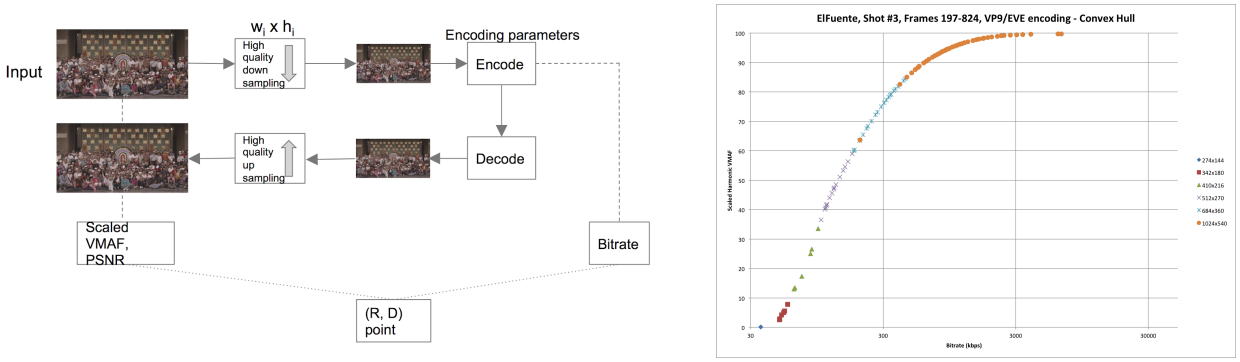
Returning to the original considerations regarding HAS, one can immediately understand that the problem faced by adaptive streaming providers is not how to do *one* encode of each titles, but rather how to do an *ensemble* of them, that can cover the entire range of qualities/bitrates that users of their service may need, depending on the exact conditions of their connection. This, together with a number of key observations, which can be summarized in the following points, led to the invention of a new video encoding optimization framework, called "Dynamic Optimizer", described in one of Netflix's techblogs.[21] Netflix further explained the benefits brought by the dynamic optimizer in another techblog,[22] where it was announced that this framework is currently being used to produce streams.

- Most visual content used in commercial services, is comprised of concatenated *shots*, each one about 2–10 sec long. These shots have little - if any - correlation and thus can be used as the basic atoms of independently coded units in a video encoder.

- At the shot level, visual content is rather uniform and homogeneous, which means that the coding mode that fits naturally in coding each shot individually can be approximated by the "fixed quality" mode. This mode is referred to as "fixed QP" and is the default mode chosen in reference encoder implementations or by terms such as "Constant Rate Factor" (CRF) in popular practical encoders, such as x264 and x265.
- The visual disruption occurring at shot boundaries - typically, an abrupt change of camera point of view or setting - is the dominant factor by the human visual system; as such, any change of coding parameters at the shot boundary has the highest chance of being unnoticed. In other words, shots offer the perfect opportunity to adjust encoding resolution and/or quantization parameter
- Cloud computing allows multiple encodings of the same set of frames a rather straightforward task using software encoders.
- The proliferation of perceptual video quality, such as SSIM[11] and VMAF metrics allows using them, instead of the traditional Peak-Signal-to-Noise-Ratio (PSNR) metric, to improve encoder decisions - in particular those that have more system-level impact, such as the optimal resolution and quantization parameter to use for each shot in a long video sequence.

The steps followed by the Dynamic Optimizer framework can be summarized as follows

1. A long video sequence is first split in shots, with an optional maximum time duration of each coding unit. For example, one may require that each coded unit must be less than 10 sec., in order to allow practical system offer seek points and/or stream adaptation during streaming of content to a client using HAS.
2. Each shot is pre-processed by downsampling it to a number of target encoding resolutions. For example, a video sequence that is provided in HD resolution (1920×1080) can be downsampled to 1280×720, 960×540 and 640×360. Please note that this step is common in almost every practical on-demand streaming service, since this allows clients to quickly adapt to changing network conditions.
3. Each shot gets encoded using a multitude of quantization parameters/CRF values, as allowed by the video encoder at hand. For example, using the HM reference encoder to produce HEVC streams, one can provide QP values in the range 0-51. Similarly, the x265 practical HEVC encoder allows for the specification of a CRF parameter, also taking values in the same range (0-51). These shot-encodes are also referred to as *elemental* encodes.
4. Each elemental encode is fully decoded and upsampled back to the same resolution of the original video sequence; in the previous example, that corresponds to (1920×1080). Then, a set of video quality metrics is calculated, including, PSNR and VMAF values. We called these metrics *scaled* PSNR and *scaled* VMAF, correspondingly.
5. At this point, one picks one of the available quality metrics - for example, scaled VMAF, and provides the set of (R,Q) values to the optimization framework, effectively providing a two-dimensional array (matrix) of such points. The first dimension in this array represents shot index, i.e. time, and the second index is an enumerator for the possible resolution and quality factor values used to produce it. One can convert the (R,Q) points into their corresponding (R,D) points, by converting quality to distortion.
6. The matrix of (R,D) points are then processed first independently for each shot, where the vector of (R,Q) points corresponding to a certain shot is used to construct the lower convex-hull of them, thus keeping a subset of them. It should be clear by now that this convex hull may have a different number of elemental encodes for each shot. For example, a shot with all-black frames may have just 1 or 2 points, while a highly complex, full of texture or large motion content may have many more.
7. These convex hulls are then used to jointly optimize to produce the coding path that minimizes distortion for a given target average bitrate or - equivalently - minimize bitrate for a given target average distortion. See Ortega and Ramchandran[23] for more details.



(a) Dynamic Optimizer main computational block

(b) Convex hull of a single shot

Figure 4: Dynamic optimizer operation

In essence, the dynamic optimizer uses the empirically produced convex hull to characterize each shot. It then finds the best trade-off between quality and bitrate jointly among all shots. One can view this method as a "multi-pass" encoding rate control method, with the difference being that, due to the exponentially growing number of possibilities when considering long sequences of multiple shots, it allows what is truly an upper bound on how well long-term rate control can perform. It has been shown in the techblog that first discussed the dynamic optimizer that, compared to fixed-QP encoding of a long sequence, it can provide approximately 25% bitrate savings, when using either VMAF or PSNR as the objective metric.

We show a few of the steps in dynamic optimizer in Fig. 4.

One of the largest advantages offered by the dynamic optimizer is that it is agnostic to the actual video encoder being used to produce elemental encodes. At the same, by swapping one quality metric for another - for example, using scaled PSNR instead of scaled VMAF - one can obtain results for multiple metrics, while still using the same elemental encodes as building blocks. These two advantages led us using it as a form of "equalizer" when comparing different codecs through their corresponding encoder implementations. This is the motivation to include it in this work, since we are presenting comparison results over long video sequences.

It needs to be also understood by the arguments discussed previously that the methodology we are proposing is suitable for off-line (non realtime) video encoding of entertainment-type premium video. This covers nicely the adaptive streaming application area, but we need to be aware that video consumption is much more than that, since it includes online video gaming, screen-content sharing, two-way video conferencing-style realtime communication, just to name a few. In these applications, the results attained vary, depending on the type of content, the real-time nature and the different display devices used.

3. VIDEO ENCODERS USED FOR TESTING

We decided to focus on 3 state-of-the-art video coding standards: H.264/AVC,[2] H.265/HEVC[3] and VP9.[4] For each one of these standards, we opted to use a practical implementation, tuned towards applications that build products around them.

3.1 H.264/AVC

Finalized in 2003 by ITU.T and ISO/MPEG, and first used by the new (at that time) Blu-ray optical disks, H.264/Advanced Video Coding (AVC)[2] is perhaps the most popular video codec used by the world's most important TV broadcasting and over-the-top (OTT) streaming service providers.

x264[24] is an open source AVC encoder project, maintained by the VideoLan (VLC) community that has been evolving over the past decade or so. It reached a very mature state about 5 years ago, and it is available both as a standalone executable that one can build in a number of operating systems and HW configurations, but also as a library that can be called from the popular "ffmpeg"[25] media processing framework. The version used for this study is the one tagged as "x264-snapshot-20180718-2245-stable". x264 allows tradeoff between speed

and quality by setting a number of parameters. In addition, it offers two different general modes of operation - selecting quality or selecting target bitrate. The traditional bitrate mode allows the user to select desired bitrate and the afforded level of variability - constant bitrate (CBR) or variable bitrate (VBR) - as well as the number of encoding passes desired (one or two). The quality mode is referred to as "constant rate factor" (CRF) mode, and allows the user to specify a value that is analogous to the fixed QP value that other AVC encoder implementations offer. The key difference between CRF and fixed QP is that the former attempts to adjust QP values used in each frame appropriately, such that the perceived video quality remains constant, unlike QP values that need to have a separate determination among I, P and B frames. In addition to the CRF mode, another two of its key settings that can affect quality of an encode is the choice of "aq-mode" and "psy-rd" values. The first of these parameters (aq-mode) controls if and to what extend x264 changes quantization parameter values spatially within a frame. The idea here is that by allowing lower QP values in areas of a frame where distortion is more visible (such as flat regions), one can improve visual quality - sometimes, at the expense of a lower PSNR value. The second parameter (psy-rd) controls how much mode decisions are penalized for the appearance of coding error, and not only by its energy. In this way, again, one attempts to minimize the visual appearance of errors.

We opted to use the slowest mode offered in x264, in single-threaded configuration, in order to maximize coding efficiency. We also produced two sets of encodes for each shot in a video sequence: one with the "psy-rd" parameter on and another by turning it off.

3.2 H.265/HEVC

The successor to AVC, developed by the same standardization bodies and finalized in 2013, H.265/High-Efficiency Video Coding (HEVC) has been proposed as a way to reduce bitrate usage by about 50% over its predecessor, offering thus the possibility to transmit high quality video for the emerging new capabilities in video applications: Ultra High Definition (UHD, a synonym for video resolution 3840×2160), high frame rates and high-dynamic-range are just a few of the possibilities offered by HEVC. .

x265[26] is an open source HEVC encoder project, maintained by the MultiCoreWare company, which evolved from the x264 codebase and offers similar features and parameters. It is also integrated within the ffmpeg framework. Similar to x264, it offers a "CRF" mode which we used in this study. There are 3 major parameters that allow for perceptual tuning in x265: aq-mode, psy-rd and psy-rdoq. The first two are very similar in nature to their counterparts in x264. The last one, according to the documentation of x265 "–psy-rdoq will adjust the distortion cost used in rate-distortion optimized quantization (RDO quant), enabled by –rdoq-level 1 or 2, favoring the preservation of energy in the reconstructed image". We used x265 version 2.8.0 (released May 2018) for our testing.

Similar to x264, we used the slowest mode offered in x265, in single-threaded configuration. We again produced two sets of encodes per shot; one with and another without enabling the "psy-rd" and "psy-rdoq" parameters.

3.3 VP9

VP9 is a royalty-free video codec offered by Google, similar in structure to the MPEG family of codecs. The specification of VP9 was frozen in 2013 and has been widely adopted by YouTube, the popular video-sharing platform offered by Google.

EVE-VP9 is one of the very few commercial VP9 encoders provided by a company called "Two Orioles".[27] This company offers a web interface that academic institutions and researchers in the field can use to test the quality of EVE-VP9 by uploading their test sequences and then downloading the VP9-encoded streams when encoding is finished. EVE also offers a general speed setting, where the value 0 achieves the highest quality, at the expense of compute time. EVE-VP9 offers a general tuning parameter that can take the value "psnr" or "visual"; the intention is to produce encodes that have higher PSNR values vs. encodes that are more visually pleasing. We used release EVE-1.2.5 (July 2018) for our tests.

We opted for the second-slowest speed setting of EVE-VP9 ("speed=1" - slowest is "speed=0") after quick experimentation showed very little gains by the slowest mode. Once again, we produce both flavors of encodes per shot - one with "psnr" and another with "visual" tuning.

4. TESTING METHODOLOGY

The dynamic optimizer allows for any subset of encoding and QPs, not necessarily the same, to be used for different shots in a video sequence. In order to reduce the number of encodes performed for each shot, a greedy binary-search-like iterative algorithm was developed and validated at Netflix, prior to this work. This algorithm has been shown with a number of video sequences, video codecs, coding parameters and quality metrics to produce virtually identical results compared to encoding of all resolutions and QP parameters. We call this version "iterative dynamic optimizer" (IDO).

A key difference between the iterative variant of dynamic optimizer (DO) and the full version of DO is that the former requires specification of the desired bitrates and/or qualities where this greedy algorithm refines at each iteration. The full version, on the other hand, provides the "total solution" to the video encoding range, i.e. a complete, dense, R-Q convex hull for the entire sequence.

In terms of quality metrics - and their corresponding distortion measures - we used 4; 2 based on PSNR and the other 2 based on VMAF. We call the PSNR-based metrics "True PSNR" and "Classic PSNR", and are defined as follows.

Peak-signal-to-noise-ratio (PSNR) of two images X and Y, both sized $K \times L$ in pixels, is a traditional image metric that expresses mean-squared-error (MSE) of each image component (Y/Cb/Cr) in a logarithmic scale (dB), as follows.

$$PSNR = 10 \log_{10}\left(\frac{255^2}{\frac{1}{KL}\sum_{i=0}^{K-1}\sum_{j=0}^{L-1}(X(i,j)-Y(i,j))^2}\right) \tag{1}$$

Let N be the number of frames in a video sequence and $PSNRY_i$, $PSNRCb_i$ and $PSNRCr_i$ be the PSNR values for each one of the 3 color components of frame i. Classic PSNR - this is the arithmetic average of PSNR values for the Y-component of each frame:

$$PSNR_{classic} = \frac{1}{N}\sum_{n=0}^{N-1} PSNRY_n \tag{2}$$

True PSNR is the arithmetic average of MSE values for the Y/Cb/Cr component of each frame, properly weighted by the number of pixels in each component, expressed in dB; for the case of 8-bit YUV4:2:0 video is the following:

$$PSNR_{true} = 10 \log_{10}\left(\frac{1}{N}\sum_{n=0}^{N-1} 10^{-PSNRY_n} + 0.25\frac{1}{N}\sum_{n=0}^{N-1} 10^{-PSNRCb_n} + 0.25\frac{1}{N}\sum_{n=0}^{N-1} 10^{-PSNRCr_n}\right) \tag{3}$$

One can easily show that Classic PSNR can also be calculated by taking the geometric mean of MSE values for the Y component first, and then expressing that mean in dB. As a result, and focusing on the Y-component for a moment, one can prove (inequalities among harmonic, geometric and arithmetic means of sequences) that the Classic-PSNR value is always higher from the corresponding True-PSNR value. In practice, since Cb- and Cr-PSNR values are typically higher than Y-PSNR for a given frame, there is no mathematically guarantee on such inequality. Moreover, we can show that large differences in MSE values among different frames in a sequence result in a higher discrepancy between True PSNR and Classic PSNR values. Given that there is a number of publications that prefers one over the other, we decided to calculate and use both these metrics for encoded shots. We hope that, through subjective validation, the video compression community can settle on a single PSNR metric that can be used consistently for reporting research results.

When it comes to VMAF, one should always remember that it is the result of fusing various image-based quality metrics, together with a simple temporal feature. Thus, VMAF is calculated and reported on a per-frame basis, similar to the way PSNR works. Therefore, temporal aggregation of these individual frame VMAF values

is an open research question. We have experimented with a number of temporal aggregation methods, and we found that two approaches that produce consistently visually pleasing results, and also correlate nicely with subjective experiments, are arithmetic-mean averaging, called Linear VMAF, and harmonic-mean averaging, called Harmonic VMAF, as defined below:

$$VMAF_{linear} = \frac{1}{N}\sum_{n=0}^{N-1} VMAF_n \qquad VMAF_{harmonic} = \frac{N}{\sum_{n=0}^{N-1}\frac{1}{1+VMAF_n}} - 1 \tag{4}$$

In the following, and for brevity, we will use the terms CPSNR, TPSNR, LVMAF and HVMAF to indicate the 4 different video quality metrics discussed previously - Classic PSNR, True PSNR, Linear VMAF and Harmonic VMAF, correspondingly.

Given the use of iterative DO as the main block to successively optimize the quality of encodes for each codec, there was a need to choose the appropriate quality/bitrate points for the iterative DO to operate on. Since the variability of content, we focused on setting specific quality targets, instead of fixed bitrate targets. In fact, as shown through a previous "Per-title encoding optimization" Netflix tech-blog,[28] choosing the appropriate bitrates that better fit a given source video sequence can significantly improve the performance of a given video codec. Moreover, given the superiority of VMAF in correlating with human subjective scores, as well as its consistency, we opted to set a fixed set of HVMAF targets as the basic input to the iterative DO algorithm. In order to cover the entire range of qualities, motivated from recent studies around the just-noticeable-difference (JND) model in video quality,[29] which show that the average JND step corresponds to a VMAF difference of 6 units, we chose the following set of 12 VMAF quality targets:

$$[30, 36, 42, 48, 54, 60, 66, 72, 78, 84, 90, 96]$$

One can notice the spacing of 6 VMAF units among these targets and the range from 30 ("poor") to 96 - which should be close to perceptually lossless quality.

Encoding resolution were chosen to densely cover the entire range from very low (256×144) up to the source, i.e. HD, resolution. These are the 10 encoding resolutions chosen for this study:

$$[1920 \times 1080, 1536 \times 864, 1216 \times 684, 960 \times 540, 768 \times 432, 608 \times 342, 480 \times 270, 384 \times 216, 320 \times 180, 256 \times 144]$$

One can notice a ratio of close to 1.5 for the number of pixels between consecutive steps in the encoding resolution ladder.

In summary, the following strategy has been followed, for each sequence and each codec independently:

- Shot encodes were produced at various resolutions and QPs
- Dynamic optimization was run, with HVMAF as the objective function and 12 HVMAF target values, as described earlier.
- The resulting bitrates $b_i, i \in 1, \ldots, 12$, one for each step in the quality ladder were calculated
- Dynamic optimization was run 3 more times; each time using one of the other 3 quality metrics (LVMAF, TPSNR and CPSNR) as objective function, after setting the previously obtained bitrates b_i as targets

In effect, the proposed methodology achieves 12 sets of encodes for a given title, each set representing increasing qualities/bitrates of that title, i.e. 12 steps in a typical adaptive streaming quality ladder. Each set contains 4 encodes with the same average bitrate, the difference among them being that they represent the optimal encode one could get by using CPSNR, TPSNR, LVMAF and HVMAF as the objective function within the dynamic optimizer framework.

The previous steps were run iteratively, performing additional encodes for each shot, thus achieving increasing quality encodes at each iteration. Convergence and thus termination of iterative DO is signaled when there is no improvement between consecutive iterations. That is the final convergence point for the iterative DO algorithm, from which the resulting (R,Q) pairs were then taken and reported in the following tables and figures.

5. RESULTS

Before going to the actual figures, we believe it's important to explain the parameters chosen for encoders and testing material

5.1 General encoder settings - source material preprocessing

We restricted our analysis to YUV420, 8-bit content. We chose High Profile for AVC, Main Profile for HEVC and Profile 0 for VP9. We used only source material of High Definition (1920×1080) or higher resolution, with frame rates between 24 and 30fps - also known as standard frame rate (SFR). Content available in higher formats was down-converted to HD, SFR, YUV420/8-bit prior to processing.

For the temporal subsampling of high frame-rate (HFR) sources, the odd-numbered frames (counting from 0) were kept. For spatial down-scaling of higher than HD sources, we used the Lanczos filter with parameter $\alpha = 5$, a linear filter with very high quality. To convert 10-bit sources to 8-bit digital signals, the 2 least-significant bits of each sample were discarded, with rounding. The same high-quality Lanczos $\alpha = 5$ filter was used to upsample decoded sequences to the same input source resolution, prior to calculating PSNR and VMAF quality metrics.

Where present in the source material, black lines or columns were removed at the preprocessing step and didn't contribute to the coding cost, nor to the quality of reconstructed frames.

Another important preprocessing step was the partition of long sequences in shots. To do that, a shot-change detection algorithm, described in,[30] based on multi-scale frame pixel differences was used. Although not perfect (especially for cross-fades and other non-cut shots), the shot partition was kept uniform for all codecs under testing, so as to equalize the number of Intra/Key frames. It should be understood that, in the presence of a more sophisticated shot-detection algorithm, the efficiency of the whole system is increased, for all codecs tested.

5.2 Specific encoder settings

In general, we tried to use those parameters that result in the most exhaustive search space for each encoder. As such, we disabled multi-threading or the use of tiles, for those encoders that allow it. Moreover, we used a fairly large search window for motion estimation, setting it to be equal to 1/8 of the frame width, which corresponds to 240 pixels at HD resolutions.

Detailed settings of different encoders can be found in Table 1. As mentioned before, for each encoder, we produce two sets of results, with different emphasis of tuning (perceptual vs PSNR). The corresponding encoder options (alternative to each other) are listed in the last two rows of the table.

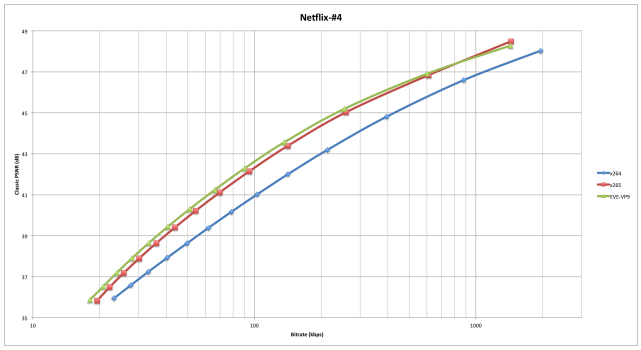
Table 1: Encoder settings of x264, x265 and EVE-VP9

settings	x264	x265	EVE-VP9
profile	high	main	profile 0
preset	placebo	placebo	speed 1
number of titles	n/a	1	1
multi-threading	off	off	off
pass	1	1	1
PSNR-tuning	psy-rd=0	psy-rd=0 psy-rdoq=0	tune=psnr
perceptual-tuning	psy-rd=1.00	psy-rd=1.00 psy-rdoq=1.00	tune=visual

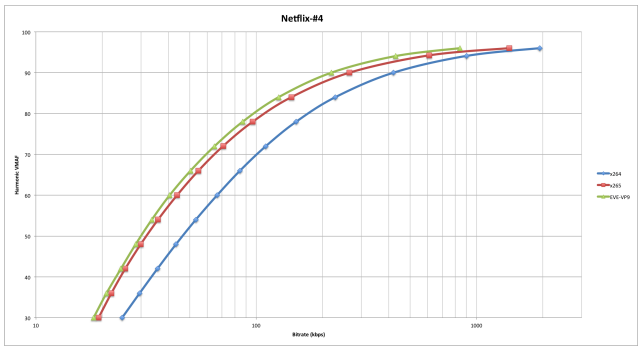
5.3 Test video material

We decided to test this methodology on 8 full-titles from the Netflix catalog; 7 contain natural video with a wide variety of visual content, typical of film and TV-drama and 1 animation (synthetic video) content. All 8 titles were available in high quality sources.

We understand that the use of proprietary material is an impediment to those researchers who wish to reproduce our results. Yet, we believe that if researchers attempt to use the proposed framework and methodology on the content that Netflix has produced and made public, such as "El Fuente"[9] and "Chimera",[10] their results should be similar to the ones we obtained on the 8 full Netflix titles.



(a) Convex hull using CPSNR metric



(b) Convex hull using HVMAF metric

Figure 5: Example Rate-Quality convex hulls produced by the dynamic optimizer using different metrics.

5.4 Main results

We first present a typical set of convex hulls obtained for one of the Netflix titles (Netflix-#4), for the 3 tested encoders. Fig. 5a shows the dynamic optimizer result using CPSNR as the objective metric, while fig. 5b is the corresponding result in terms of HVMAF.

One can immediately notice that different quality metrics can change the ranking of these codecs. It can also be observed from the following summary table that there is different behavior of codecs for natural vs. synthetic video content. For natural video, in terms of HVMAF, the better codec is EVE-VP9, with a smaller advantage in the low quality range, which becomes more significant in the high quality range. In fact, at the upper bound of high quality range (HVMAF value 96) of the specific sequence shown in Fig. 5b, there is a difference of almost 40% in favor of EVE-VP9. For synthetic video, x265 offers better performance in the low quality range, while EVE-VP9 is better in the high quality range, in particular when using VMAF metric.

The behavior seen on these two figures is typical of what we observed on the majority of natural video sequences. The different behavior in low and high quality ranges led us to further split and report performance in 3 different quality ranges - low quality (LQ), high quality (HQ) and full quality (FQ) - when reporting BD-rates.

This is the method we used to determine the portion of the Convex hull that corresponds to low and high quality, individually for each test sequence:

- Start with the convex hull using the HVMAF quality metric for the baseline encoder (x264)
- Calculate the bitrate points corresponding to HVMAF quality of 30 (b_{min}), 63 (b_{mid}) and 96 (b_{max}) to mark the lowest, middle and highest quality ranges. b_{mid} is used as the threshold between low and high quality ranges
- Subsequently, use the 3 calculated bitrates points ($b_{min}, b_{mid}, b_{max}$) and the convex hulls of the 3 quality metrics (TPSNR, CPSNR and LVMAF) for the baseline encoder (x264) to obtain the map into corresponding qualities for all 4 quality metrics through their convex hulls.
- Let $Q_{min}^{metric}, Q_{max}^{metric} and Q_{max}^{metric}$ be the set of 3 relevant qualities for each one of the 4 quality metrics. By construction, $Q_{min}^{HVMAF} = 30, Q_{mid}^{HVMAF} = 63$ and $Q_{max}^{HVMAF} = 96$, while for the other 3 metrics, the values are content-dependent; but we guarantee that the corresponding bitrate ranges remain the same across all 4 metrics.

This method of defining low and high quality ranges is consistent with the way we used iterative dynamic optimizer to obtain 12 steps in the range $[30, 96]$ of HVMAF quality ladder. In this way, we make sure that half of this dozen of target points map to the low quality range, while the other half map to the high quality range. This method was proven to be robust against all different kinds of content, natural and synthetic, and

Table 2: BD-rate (%) performance of 3 video encoders on 8 test video sequences; using PSNR and VMAF metrics

Metric	Sequence name	Quality Range	x264	x265	EVE-VP9
TPSNR	AVERAGE (Netflix titles)	LQ	0.0	-25.4	-26.8
TPSNR	AVERAGE (Netflix natural video)	LQ	0.0	-25.2	-28.7
TPSNR	AVERAGE (Netflix animation)	LQ	0.0	-27.1	-14.1
TPSNR	AVERAGE (Netflix titles)	HQ	0.0	-32.4	-32.9
TPSNR	AVERAGE (Netflix natural video)	HQ	0.0	-32.8	-34.0
TPSNR	AVERAGE (Netflix animation)	HQ	0.0	-29.1	-25.0
TPSNR	AVERAGE (Netflix titles)	FQ	0.0	-20.7	-25.1
TPSNR	AVERAGE (Netflix natural video)	FQ	0.0	-20.2	-25.8
TPSNR	AVERAGE (Netflix animation)	FQ	0.0	-23.9	-20.3
CPSNR	AVERAGE (Netflix titles)	LQ	0.0	-30.8	-30.2
CPSNR	AVERAGE (Netflix natural video)	LQ	0.0	-30.3	-30.8
CPSNR	AVERAGE (Netflix animation)	LQ	0.0	-34.0	-25.8
CPSNR	AVERAGE (Netflix titles)	HQ	0.0	-43.9	-39.7
CPSNR	AVERAGE (Netflix natural video)	HQ	0.0	-45.1	-41.0
CPSNR	AVERAGE (Netflix animation)	HQ	0.0	-35.5	-30.4
CPSNR	AVERAGE (Netflix titles)	FQ	0.0	-29.6	-30.1
CPSNR	AVERAGE (Netflix natural video)	FQ	0.0	-29.1	-30.5
CPSNR	AVERAGE (Netflix animation)	FQ	0.0	-32.8	-27.3
LVMAF	AVERAGE (Netflix titles)	LQ	0.0	-30.4	-34.8
LVMAF	AVERAGE (Netflix natural video)	LQ	0.0	-31.1	-36.1
LVMAF	AVERAGE (Netflix animation)	LQ	0.0	-26.2	-25.8
LVMAF	AVERAGE (Netflix titles)	HQ	0.0	-30.6	-39.8
LVMAF	AVERAGE (Netflix natural video)	HQ	0.0	-31.1	-41.1
LVMAF	AVERAGE (Netflix animation)	HQ	0.0	-27.4	-31.1
LVMAF	AVERAGE (Netflix titles)	FQ	0.0	-23.2	-30.6
LVMAF	AVERAGE (Netflix natural video)	FQ	0.0	-23.8	-32.5
LVMAF	AVERAGE (Netflix animation)	FQ	0.0	-19.0	-17.3
HVMAF	AVERAGE (Netflix titles)	LQ	0.0	-28.8	-32.6
HVMAF	AVERAGE (Netflix natural video)	LQ	0.0	-29.3	-33.8
HVMAF	AVERAGE (Netflix animation)	LQ	0.0	-25.7	-24.3
HVMAF	AVERAGE (Netflix titles)	HQ	0.0	-30.1	-38.8
HVMAF	AVERAGE (Netflix natural video)	HQ	0.0	-30.5	-40.0
HVMAF	AVERAGE (Netflix animation)	HQ	0.0	-27.6	-30.5
HVMAF	AVERAGE (Netflix titles)	FQ	0.0	-25.1	-31.0
HVMAF	AVERAGE (Netflix natural video)	FQ	0.0	-25.4	-32.2
HVMAF	AVERAGE (Netflix animation)	FQ	0.0	-23.1	-22.4

has the advantage of identifying portions of the convex hull that are more likely to be used in adaptive streaming applications.

We present the average BD-rate % gains for the 3 video encoders we studied on Table 2.

We draw the readers' attention to the fact that the full-quality range is extending beyond the union of low-quality and high-quality ranges. For example, in the case of HVMAF, full quality covers the entire scale of $[0, 100]$, even though low quality covers the range $[30, 63]$ and high quality covers the range $[63, 96]$. As a result, in some cases, the full-quality BD-rate is different from the average of the corresponding BD-rates for low and high qualities. This can be explained by the fact that the very-low or ultra-high quality regions can contribute unequally to the full quality result.

Table 3: Compute and memory complexity of video encoders, using middle QP to encode clip of 53 frames at 2048 × 1080 resolution

Video encoder	x264	x265	EVE-VP9
Time (sec).	47.5	922	215
Frames-per-sec	1.116	0.057	0.247
Relative complexity	1×	19.4×	4.5×
Video encoder	x264	x265	EVE-VP9
Memory (kbytes)	833,080	720,644	677,396
Number of frame buffers	257	222	209
Relative complexity	1.23×	1.06×	1×

5.5 Computational and memory complexity

All encoding was performed using Netflix's cloud computing infrastructure. For our experiments we used virtualized computing instances hosted on server computers running Ubuntu Linux OS over "Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz", with peak frequency scaling of 2.70GHz.

The compute and memory requirements of each video encoder were benchmarked using a short, 53-frame clip from "El Fuente", encoded at 2048 × 1080 resolution, using the middle QP value of each encoder. Since each encoder was configured to run in single-threaded mode, the execution times are representative of the compute complexity of each task.

To collect these figures, the GNU "time" utility was used and the sum of User time and System time is reported. Encoding time figures are also normalized into encoded frames-per-sec and peak memory usage is translated into equivalent number of input frames, an indication of how many frames are internally stored by each encoder for its operation.

The results collected are presented in Table 3.

The total size of the experiments, for all 3 video encoders - in dual configuration - and all sequences, resulted into over 10 million encoded clips, each clip representing a certain encode produced by an encoder operating on a shot with a given encoding resolution and QP. The integral of compute times in order to produce all these encodes was over 1 million compute hours (a compute hour is equivalent to one core of a compute instance being occupied for one hour at 100% system loading).

A special note is due to the Netflix Media Cloud Engineering - Compute and Storage Infrastructure (MCE-CSI) team who made possible to run this massive scale of testing. The internally developed tool that allows scaling of an encoding task to many thousands of compute instances is called "Archer". Without the Archer framework, these results couldn't be obtained in a relatively short (3 week) timeframe.

Archer was introduced by Netflix's MCE-CSI team in 2016 as simple map-reduce style media processing platform that allows developers to package their code quickly and efficiently in containers and then use virtualized compute instances around existing cloud computing resources. Archer leverages the abstraction of virtualized containers, cloud storage and distributed computing through message passing. It allows developers to bring to their applications any OS tool of choice. More details about Archer can be found in[31]

6. DISCUSSION

One can draw from the previous figures and tables a few important conclusions.

When considering all 8 titles for the full quality range:

- Both x265 and EVE-VP9 are more efficient than x264 by approximately 22.9% in terms of TPSNR, 29.8% in terms of CPSNR, 26.9% in terms of LVMAF and 28.0% in terms of HVMAF
- EVE-VP9 is more efficient than x265 on all metrics: 4.4% in terms of TPSNR, 0.5% in terms of CPSNR, 7.4% using LVMAF and 5.9% using HVMAF. In fact, the VMAF advantage of EVE-VP9 over x265 is rather significant.

When splitting content between natural video and animation for the full range:

- EVE-VP9 has higher efficiency on natural video content compared to animation; there is a difference in performance for the four metrics of 5.5% (TPSNR), 3.2% (CPSNR), 15.2% (LVMAF) and 9.8%(HVMAF)
- x265 exhibits very small difference in gains between natural video and animation; favoring animation in terms of PSNR (TPSNR - 3.7%, CPSNR - 3.7%) and favoring natural video in terms of VMAF (LVMAF - 4.8%, HVMAF - 2.3%)
- EVE-VP9 is better than x265 for natural video content - offering gains that depend on the metric, as follows: 5.6% (TPSNR), 1.4% (CPSNR), 8.7%(LVMAF) and 6.8%(HVMAF)
- x265 is better than EVE-VP9 for animation content - gains are 3.6% (TPSNR), 5.5% (CPSNR), 1.7% (LVMAF) and 0.7% (HVMAF) for the different metrics.

When comparing performance between low and high quality ranges for all titles, both encoders are more efficient in the high quality range, with gains that vary depending on the metric:

- EVE-VP9 has higher efficiency on the high quality over the low quality range by 6.1-9.5% in terms of PSNR metrics and 5.0-6.2% in terms of VMAF metrics
- x265 has higher efficiency on the high quality range compared to the low quality range by 7-13.1% in terms of PSNR metrics, and 0.2-1.3% in terms of VMAF metrics.

ACKNOWLEDGMENTS

The authors are grateful to Anne Aaron, Rich Gerber, Megha Manohara and David Ronca from the Encoding Technologies team at NETFLIX for providing useful feedback on earlier drafts of this work. Naveen Mareddy from the MCE-CSI team at Netflix provided crucial help with the necessary compute instances offered by Archer - we would like to thank him for that.

REFERENCES

[1] Wiegand, T., Sullivan, G. J., Bjøntegaard, G., and Luthra, A., "Overview of the h.264/avc video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on* **13**(7), 560–576 (2003).
[2] "Advanced video coding for generic audiovisual services," (2003). ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC).
[3] "High efficiency video coding," (2013). ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC).
[4] Grange, A., de Rivaz, P., and Hunt, J., "VP9 bitstream and decoding process specification," (2016).
[5] "JM, code repository - reference AVC software." link: http://iphome.hhi.de/suehring/tml/download.
[6] "HM, code repository - reference HEVC software." link: https://hevc.hhi.fraunhofer.de.
[7] "libvpx, code repository - reference VP9 encoder/decoder software." link: https://github.com/webmproject/libvpx.
[8] Bjontegaard, G., "Calculation of average psnr differences between rd curves," *ITU-T Q.6/SG16 VCEG 13th meeting* (2001).
[9] "Consumer Digital Video Library - El Fuente." link: http://www.cdvl.org/documents/ElFuente_summary.pdf.
[10] "Consumer Digital Video Library - Chimera." link: https://www.cdvl.org/documents/NETFLIX_Chimera_4096x2160_Download_Instructions.pdf.
[11] Wang, Z., Bovik, A., H.Sheikh, and Simoncelli, E., "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Processing* **13**(4), 600–612 (2004).
[12] Sheikh, H. R. and A.C.Bovik, "Image information and visual quality," *IEEE Trans. on Image Processing* **15**(2), 430–444 (2006).
[13] "VMAF, code repository." link: https://github.com/Netflix/vmaf.

[14] Cock, J. D., Mavlankar, A., Moorthy, A., and Aaron, A., "A large-scale video codec comparison of x264, x265 and libvpx for practical vod applications," *Proc.SPIE* **9971**, 9971 – 9971 – 17 (2016).
[15] Guo, L., Cock, J. D., and Aaron, A., "Compression performance comparison of x264, x265, libvpx and aomenc for on-demand adaptive streaming applications," in [*Picture Coding Symposium (PCS-2018)*], (2018).
[16] "aomenc, code repository - open-source AV1 encoder software." link: https://aomedia.googlesource.com/aom/.
[17] Li, S., Ma, L., and Ngan, K., "Full-reference video quality assessment by decoupling detail losses and additive impairments," *Circuits and Systems for Video Technology, IEEE Transactions on* **22**(7), 1100–1112 (2012).
[18] Winkler, S., "Analysis of public image and video databases for quality assessment," *IEEE Journal of Selected Topics in Signal Processing* **6**(6), 616–625 (2012).
[19] C.Cortes and V.Vapnik, "Support-vector networks," *Machine Learning* **20**, 273–297 (1995).
[20] Li, Z., Aaron, A., Katsavounidis, I., Moorthy, A., and Manohara, M., "The NETFLIX tech blog: Toward a practical perceptual video quality metric," (Mar. 2018). link: http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html.
[21] Katsavounidis, I., "The NETFLIX tech blog: Dynamic optimizer – a perceptual video encoding optimization framework," (Mar. 2018). link: https://medium.com/netflix-techblog/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f.
[22] Manohara, M., Moorthy, A., Cock, J. D., Katsavounidis, I., and Aaron, A., "The NETFLIX tech blog: Optimized shot-based encodes: Now streaming!," (Mar. 2018). link: https://medium.com/netflix-techblog/optimized-shot-based-encodes-now-streaming-4b9464204830.
[23] Ortega, A. and Ramchandran, K., "Rate-distortion methods for image and video compression: An overview," *IEEE Signal Processing Magazine* **15**(6), 23–50 (1998).
[24] "x264, code repository - open-source AVC encoder software." link: https://github.com/mirror/x264.
[25] "ffmpeg, code repository." link: https://github.com/FFmpeg/FFmpeg.
[26] "x265, code repository - open-source HEVC encoder software." link: https://github.com/videolan/x265.
[27] "Two orioles." link: https://www.twoorioles.com/eve-for-vp9.
[28] "The NETFLIX tech blog: Per-title encode optimization." link: http://techblog.netflix.com/2015/12/per-title-encode-optimization.html.
[29] Wang, H., Katsavounidis, I., Zhou, J., Park, J., Lei, S., Zhou, X., Pun, M., Jin, X., Wang, R., Wang, X., Huang, J., Kwong, S., and Kuo, C.-C. J., "Videoset: A large-scale compressed video quality dataset based on jnd measurement," *Journal of Visual Communication and Image Representation* **46**, 292–302 (2017).
[30] Katsavounidis, I., Aaron, A., and Ronca, D., "Native resolution detection of video sequences," in [*SMPTE 2015 Annual Technical Conference and Exhibition*], 1–20, SMPTE (2015).
[31] Mareddy, N., Miguel, F. S., Wong, R., Prabhu, M., and Johansson, O., "The NETFLIX tech blog: Simplifying media innovation at netflix with archer," (Jul. 2018). link: https://medium.com/netflix-techblog/simplifying-media-innovation-at-netflix-with-archer-3f8cbb0e2bcb.