Encoding Parameters Prediction for Convex Hull Video Encoding

Ping-Hao Wu*, Volodymyr Kondratenko †, Gaurang Chaudhari ‡, and Ioannis Katsavounidis ‡
Facebook Inc., 1 Hacker Way, Menlo Park, CA 94025
Email: npw@fb.com*, vkondratenko@fb.com†, gaurangc@fb.com‡, ikatsavounidis@fb.com§

***Abstract*—Fast encoding parameter selection technique have been proposed in the past. Leveraging the power of convex hull video encoding framework, an encoder with a faster speed setting, or faster encoder such as hardware encoder, can be used to determine the optimal encoding parameters. It has been shown that one can speed up 3 to 5 times, while still achieve 20-40% BD-rate savings, compared to the traditional fixed-QP encodings.**

Such approach presents two problems. First, although the speedup is impressive, there is still $\sim$3% loss in BD-rate. Secondly, the previous approach only works with encoders implementing standards that have the same quantization scheme and QP range, such as between VP9 and AV1, and not in the scenario where one might want to use a much faster H.264 encoder, to determine and predict the encoding parameters for VP9 or AV1 encoder.

In this work we propose and present new additions to address these issues. First, we show that the predictive model can be used to gain back the quality loss incurred from using a faster encoder. Secondly, we also demonstrate that, by introducing a mapping layer, it is possible to predict between two arbitrary encoders and reduce the computational complexity even further. Such degree of complexity reduction is made possible by the fact that different generations of video codecs are usually orders of magnitude apart in terms of complexity and thus one can use an encoder from an earlier generation to predict the encoding parameters for an encoder from later generations.

***Index Terms*—Video encoding, video codec, per-shot encoding, convex hull**

I. Introduction

In the age of internet video, adaptive bitrate streaming (ABR) has been the dominant form of distribution. In ABR streaming, as opposed to one single encoding, several encodings with different resolutions and/or quality levels are typically produced for a single video source. This allows client applications to quickly switch to an appropriate version depending on the bandwidth and device limitations. To address the paradigm shift in practical applications of video codecs, Dynamic Optimizer and the convex hull video encoding framework were proposed to provide a way to compare codecs fairly [1]. It has also been shown that a massive 20-40% saving in BD-rate can be achieved under this framework.

However, such impressive saving comes with a heavy overhead in computational complexity, which is typically $6-10\times$ compared to the traditional fixed-QP encoding. A fast encoding parameter selection for the convex hull video encoding framework was proposed in [2] to significantly reduce the computational cost to roughly 120-150% of traditional fixed-QP encoding, with no more than 3 percentage points (p.p.) BD-rate loss. It uses a faster encoder or the same encoder with faster speed settings to perform analysis on the video sequence first, to determine the optimal encoding parameters to be used with a slower encoder with better compression efficiency. Although this fast selection technique can achieve similar savings with a fraction of the computations, limitations are still present. First of all, the few percent loss in BD-rate is not negligible in the case of higher-quality contents, where a much higher target quality and/or target bitrate is often used. Secondly, the quantization scheme of the fast encoder, used for analysis, and that of the slow encoder, used for producing the actual deliverable encodings, need to be the same in order for the technique to work.

To address the limitations, we experimented different features with our internal implementation of VP9 [3], and proposed a predictive approach work on top of the fast selection technique presented in [2]. With this new approach, an arbitrary pair of encoders can be used in the “analysis” and the actual “encode” steps. Furthermore, it is also shown that more than half of the loss in BD-rate can be recovered.

In Sec. II, the prior work on fast encoding parameter selection technique for convex hull video encoding, comprising of 3 steps, “preprocess”, “analysis”, and “encode”, will be briefly reviewed. In Sec. III, the effect of encoder features will first be presented. Then the new “prediction” step, on top of the prior fast encoding parameter selection technique, will be introduced to address the aforementioned issues. Sec. IV is dedicated to presenting and discussing the experiment setup and results. Followed by Sec. V with summary, remarks, and future works.

II. Encoding Parameters Selection and Convex Hull Encoding

Convex hull video encoding and the Dynamic Optimizer framework first appeared in a NETFLIX tech blog [4], and the subsequent work of using it to fairly compare video codecs [1]. The process of efficiently selecting optimal encoding parameters based on this framework is briefly described below, with more details in [2].

1) **Preprocess**: Perform shot detection and split a video sequence into multiple shots.
2) **Analysis**:
   a) Downscale and encode: Downsample and encode each shot using a faster encoder or a faster setting, at M different resolutions and N QP values.

    b) Decode and upsample: Decode and upsample each encode back to the original resolution of the video sequence for metrics calculation.
    c) Dynamic Optimizer: Use Dynamic Optimizer to determine the optimal selection of encoding parameters (resolution and QP) for all shots, at a desired average quality level.
3) **Encode**: Encode using a slower encoder or encoding setting, by directly applying the optimal encoding parameters obtained in the previous step.

Similar approach that analyzes the video by pre-encode has also been proposed in [5]. In this work, we focus on using faster encoders to perform the analysis.

III. Predicting Encoding Parameters

In this section, the effect of different encoding features is first explored. Then a new prediction step is introduced on top of the approach presented above, illustrated in Fig. 1. A predictive modeling approach within the same standard is first presented to improve the bitrate saving without additional computations. It recovers the BD-rate loss when switching from a slower encoder to a faster one for the "analysis" step.

Extending the same concept that inspired our prior work, a cross-codec prediction approach is then presented, using a quantization parameter map across two codecs that have different quantization schemes. This allows one to use the fastest encoder from an earlier generation codec, to predict the encoding parameters for an encoder from a later generation. With the orders of magnitude difference in computations across codec generations, this enables us to massively improve BD-rate performance, with minimal computation overhead.

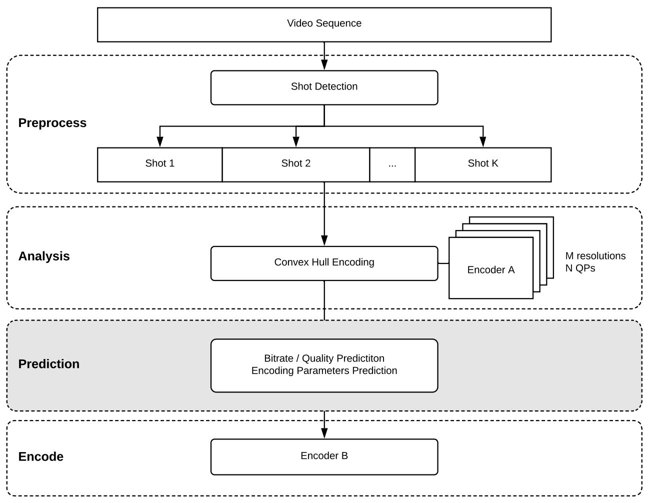

Fig. 1. Illustration of predicting encoding parameters for convex hull encoding.

A. Effects of Encoder Features

An encoder implementation usually has a number of tools and features that optimizes for bitrate-quality trade-off. One of the key, and the most compute intensive, components in block-based video encoding is mode decision (MD), which determines the appropriate mode and partition for a given block. A straightforward way is to evaluate the entire search space, i.e., compute the cost of all possible combinations and choose the best one, which is obviously the most expensive way. To reduce the complexity, non-normative techniques, which often include early terminations or threshold based pruning of the search space, are employed. One of the main factors in the different speed settings of an encoder implementation is based on these smart reductions of search space. Typically, a faster encoding setting would have more smart reductions on the search space, while the slowest one would have none or very few of such reduction.

To get further compute reduction in the "analysis" step, the mode decision search space is analyzed and a static reduction of search space is done, rather than depending on the dynamic early termination or pruning. The analysis is done with the VP9 internal experimental implementation in [3]. All early terminations or threshold-based pruning were disabled, and thus created a brute-force search configuration referred to as the full configuration. In our experiment, the following lower compute configurations were used as the faster encode setting in the "analysis" step.

1) **Feature 1** (F1) : Disable 4×4 + disable compound mode: disable the 4×4 TU mode search and the compound modes in inter frame in the mode decision
2) **Feature 2** (F2) : Luma based MD: Disable mode decision process for chroma
3) **Lowest compute**: Combination of the above (F1 + F2)

The performance model of the experimental encoder was mapped to estimate the number of cycles spent in finishing one entire frame of different frame types. The performance was estimated by modeling the memory access, bandwidth, and latency of data movement across the entire encoding process. All blocks are modeled in a pipelined manner to match the throughput of one superblock. If the full configuration is normalized to use 100% of the compute, Table I shows the compute and the corresponding BD-rate loss when the lower compute setting is used at the "analysis" step.

TABLE I
Low compute for Analysis + Full configuration for Encode

		BD-rate loss		
Analysis step configuration	**Compute %**	**PSNR**	**FB-MOS**	**VMAF**
Full configuration	100%	0%	0%	0%
F1	66.91%	-0.04 p.p.	-0.01 p.p.	0.00 p.p.
F2	61.02%	2.72 p.p.	3.06 p.p.	1.20 p.p.
lowest compute = F1 + F2	50.94%	2.71 p.p.	3.09 p.p.	1.20 p.p.

The result is averaged across 10 publicly available high quality videos. In Table I, positive numbers for BD-rate loss mean worse than the baseline, where the full configuration is used for both the "analysis" and the "encode" steps. Lower compute configurations were used for the "analysis" and the full configuration for the "encode" step. F2 and the lowest compute configuration are almost the same in terms of BD-rate while the difference in compute is around 11 p.p. This is because that while the features that account for the operations in the mode decision block were disabled, it does not necessarily translate to the winner decision for the final encoding.

Many of the early terminations and pruning in faster speed settings are targeted towards sub8x8 and inter joint compound search thresholds. This provided the motivation for disabling 4x4 or compound mode features. The lowest compute configuration used for the "analysis" step gives almost 50% compute reduction with around 3.1 p.p. loss in BD-rate for FB-MOS and even lower 1.2 p.p. loss for VMAF. We would later demonstrate such loss can be recovered with a model based approach in Sec. III-B.

Similar to [2], to examine the effect of these low compute features and their relationship with resolution and QP further, we perform a statistical analysis of the qualities and bitrates from different configurations. Quality metrics and bitrates are averaged over all the shots for each (resolution, QP), and across different (resolution, QP). The aggregate changes of bitrate and quality metrics with different compute configurations are then examined. Table II shows the difference of bits, quality metrics for each configuration, compared with the full configuration. The quality metrics stay roughly the same, while bits spent increase noticeably with lower compute.

TABLE II
AVERAGE BITRATE AND QUALITY DIFFERENCES WITH SAME (RESOLUTION, QP), COMPARED TO FULL CONFIGURATION

Encoding configuration	Δ bitrate	Δ PSNR	Δ FB-MOS	Δ VMAF
Full configuration	0.00%	0.00	0.00	0.00
F1	0.28%	-0.01	-0.04	0.01
F2	11.18%	-0.29	-2.53	-1.07
lowest compute = F1 + F2	11.50%	-0.30	-2.57	-1.06

B. Predicting within The Same Coding Standard

Based on the observation in Sec. III-A, a predictive modeling approach that uses Machine Learning is presented, to be able to predict the bitrates and quality metrics produced by a slower encoder from that by a faster one.

When considering the model architecture, the bitrate and quality interaction needs to be taken into account. The multi-target regression model was chosen in order to look at bitrate and quality as a combined target. We built a model factory conducting a comparison analysis of 35 machine learning regression algorithms. Multivariate random forest regressor was selected as the best fit for this purpose. It provides an additional bonus of having a built-in cross validation step assuring balanced data representation. The data is split into train (80%) and test (20%) sets warranting good generalization potential of the model. Below is the model description.

- **Dataset structure**: Each row of the dataset is a unique combination of video shot, QP, and resolution.
- **Feature set**: The features used for the training are QP, width, height, frame count, and the corresponding bitrate and quality metrics collected from the faster encoder.
- **Label**: The output label consists of the pair of bitrate and quality metric, which were produced by a slow encoder for the same video shot and parameters as the feature set.
- **Model evaluation**: The model is evaluated in two phases. First RMSE and MAPE metrics are used for the model evaluation on per row basis. Then, BD-rate loss between the convex hull of the slow encoder and that based on the prediction model is used to select the best fitting model out of the most performing ones from the first phase.

Once the model training phase completes, encoder B parameters can be obtained from encoder A with the modified "analysis" step, realized this way:

1) **Preprocess**
2) **Analysis**:
    a) Downsample and encode
    b) Decode and upsample
    c) **Prediction**: From the bitrate and quality pair of each encoding, (r_A, q_A), predict the would-be $(\hat{r_B}, \hat{q_B})$ for encoder B.
    d) Determine the optimal selection of encoding parameters (resolution and QP) using the predicted $(\hat{r_B}, \hat{q_B})$.
3) **Encode**

C. Predicting Across Different Coding Standards

The approach in the previous section works within the same coding standard, because that even if with different implementations, how QP affects the distortion is similar and thus possible to perform the one-to-one prediction.

On the other hand, different quantization schemes do exist for different standards. Two of the most popular ones, H.264/AVC and VP9, have completely different designs in quantization. The same QP would not result in the same quality level. Furthermore, increasing QP by a certain scale also would not result in the same proportional decrease in quality.

The key insight from [2] is that between a slower and a faster speed setting, the distortion stays roughly the same with the same QP value, while the bitrate changes proportionally to the content complexity. Extending on this concept, when looking at two different coding standards, if one can find certain QP mapping between them that can produce encodings with similar distortions, the bitrate differences should also be proportional to the content complexity.

Based on this, the encoding parameters prediction across different coding standards is proposed. Without loss of generality, in the following, encoder A denotes an encoder implementing an earlier generation of video coding standard X, and encoder B, a later generation, Y.

First, a map between the different QP values between the two coding standards X and Y would be constructed in the following fashion.

1) Gather video shots from a wide variety of contents.
2) Downsample and encode each shot using both encoder A and B, at M different resolutions and N QP values.

3) Decode and upsample each encode back to the original resolution for metrics calculation.
4) Aggregate the quality metrics for all shots that were encoded at the same resolution and QP value.
5) Build a QP map by matching the aggregated quality.

The last step, in our experiment, is simply done with the nearest neighbor approach. That is, for a QP value with encoder A, find the QP value for encoder B that would produce the closest quality, at the same resolution. With such map, one can then predict the encoding parameters needed to be used with encoder B, from the convex hull of encoder A. It can be realized this way:

1) **Preprocess**
2) **Analysis**:
   a) Downsample and encode
   b) Decode and upsample
   c) Dynamic Optimizer
3) **Prediction**: From the list of optimal parameters for encoder A at a target quality level, look up the corresponding QP value to be used with encoder B.
4) **Encode**

One of the key benefits the presented approach is that it can work with any two given video codecs and leverage the huge complexity differences between early codecs such as H.264, and the modern codecs such as VVC or AV1. However, one can argue that the most important benefit is that, given a single video sequence, the convex hull analysis would only need to be done once with a relatively cheap encoder. Results can be stored, and later used to predict encoding parameters for any new encoder or future video coding standards.

IV. Experiment Results

A. Setup

We focused on two state-of-the-art video coding standards that have popular open-sourced software implementations: H.264/AVC, and VP9. For H.264/AVC, we used x264 with single-threaded, CRF mode, and tune PSNR, along with two different presets: "veryslow" and "veryfast". For VP9, we used libvpx with single-threaded, 2-pass encoding, and constant quality mode, along with two different speed settings: 0 (slowest) and 4. In addition, we used our HW video encoder [6] to provide HW-encoded H.264 streams, as well as an internal experimental implementation of VP9 [3].

Ten real-world internet videos were used, covering a wide variety of contents, ranging from professionally edited to amateur-produced videos. They were encoded at 8 different resolutions: $1080p$, $720p$, $540p$, $432p$, $360p$, $288p$, $216p$, $144p$, and 6 different QP/CRF values: $[23, 31, 35, 37, 39, 43]$ for H.264 and $[28, 33, 38, 43, 48, 53]$ for VP9. We used Lanczos with alpha 3 to downsample and upsample the frames.

Two perceptual video quality metrics were used: FB-MOS [7], a metric developed by Facebook, and the well-known VMAF [8].

B. Predicting within the Same Codec

For predicting within the same codec, we opted to predict from HW encoders that are typically much faster, to software encoders. For each video shot, QP, and resolution, the HW video encoder would produce a (bitrate, quality) pair, used as the feature set of the model. For the same range of (video, QP, resolution), the (bitrate, quality) pairs produced by x264 "veryslow" are used as the label.

TABLE III
BD-RATE COMPARED TO X264 "VERYSLOW" WITH FIXED CRF 27

Quality metric	Technique in [2]	Predictive Modeling	Δ
FB-MOS	-13.9%	-16.0%	2.1 p.p.
VMAF	-28.08%	-29.34%	1.26 p.p.

Table III shows the results of the BD-rate for each of the quality metrics averaged across 10 videos, compared to x264 "veryslow" with fixed CRF 27. The approach presented in [2] is shown as well as the results based on predictive modeling. By introducing the prediction step, the predictive modeling can achieve a BD-rate gain of 16.0% on FB-MOS, and 29.34% on VMAF, as opposed to 13.9% and 28.08%, respectively, without prediction.

The same experiments were conducted on VP9 encoders as well, with the internal lowest compute experimental implementation of VP9, and libvpx. We can observe that without prediction, there is a gain of 16.96% in FB-MOS and 35.97% in VMAF, over fixed-QP encodings, while with predictive modeling, the gain becomes 19.22% and 36.8%. Considering the approach in [2] introduced less than 3 p.p. BD-rate loss, the results here show that the predictive modeling can recover majority of the loss.

TABLE IV
BD-RATE COMPARED TO LIBVPX "CPU=0" WITH FIXED CQ-LEVEL 38

Quality metric	Technique in [2]	Predictive Modeling	Δ
FB-MOS	-16.96%	-19.22%	2.26 p.p.
VMAF	-35.97%	-36.8%	0.84 p.p.

C. Predicting Across Different Codecs

The QP map is constructed based on the encoding statistics of an entirely different video sequence, shown in Fig. 2. As can be seen, it is fairly consistent between x264 CRF and libvpx CQ-level across resolutions. It also maxes out fairly quickly as CRF increases, which is consistent with what most researchers have noticed, which is that libvpx cannot reach as low bitrate as x264.

Table. V shows the results of cross-codec prediction. The baseline is libvpx cpu=0 with fixed CQ-level 38. Subsequent rows are the BD-rate, on FB-MOS and VMAF, obtained by "analysis" and "encode" done with different encoders or different speed settings. The row "libvpx cpu=0 $\rightarrow$ libvpx cpu=0 represents the ground truth, which is obtained by

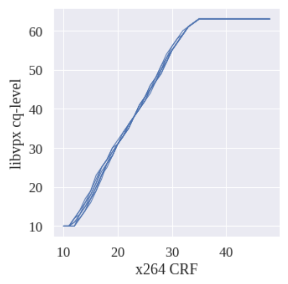

Fig. 2. x264 CRF to libvpx CQ-level mapping based on VMAF

TABLE V
BD-RATE COMPARED WITH LIBVPX "CPU=0" FIXED CQ-LEVEL 38

Configuration	FB-MOS	VMAF	Analysis Time	Total Time
libvpx cpu=0 (fixed 38)	0%	0%	-	100%
libvpx cpu=0 → libvpx cpu=0	-22.58%	-44.56%	100%	600%
libvpx cpu=4 → libvpx cpu=0	-22.02%	-44.15%	19.7%	218.4%
x264 veryfast → libvpx cpu=0	-19.35%	-40.41%	1.93%	111.6%

applying the slowest speed setting for both the analysis and the final encode step. The following rows are the results of analyzing with libvpx speed 4, and with x264 "veryfast", along with the prediction step, respectively.

Same as demonstrated in the prior work, analyzing with a faster speed setting of the same encoder, the BD-rate loss is minimal, in this case, less than 1 p.p. if staying within libvpx. On the other hand, using x264 to predict the encoding parameters of libvpx does have a larger BD-rate drop, 3.23 p.p. in FB-MOS and 4.15 p.p. in VMAF.

However, since x264 is orders of magnitude faster than libvpx, the compute overhead incurred in the analysis step is minimal. As shown in Table. V, if the analysis time with libvpx cpu=0 is normalized as 100%, the corresponding time needed for analysis with x264 "veryfast" is only 1.93%. As for the total time needed to do both the analysis and the encode, it would be 600% of the traditional fixed-QP encoding if analyzed with libvpx cpu=0, assuming 6 QPs used for analysis. With the cross-codec prediction technique, it would take only 111.6%. In other words, a massive 20-40% BD-rate improvement can still be achieved with roughly 11% more compute, compared to the traditional fixed-QP encoding.

V. Conclusion and Future Works

With faster encoder in the same codec family, it is possible to significantly reduce the complexity needed for per-shot convex hull optimization, with the trade-off of 3 p.p. BD-rate loss, as shown in our prior work. A new prediction step is introduced in this work, building on top of the fast encoding parameter selection technique, to predict across any encoder pair, and recover the majority of the BD-rate loss incurred by analyzing with a faster encoder.

First, a prediction model is introduced to predict the bitrate and the quality of the slower encoder, from that of a faster encoder. Majority of the BD-rate loss from using a faster encoder, can be recovered this way. Secondly, a cross-codec prediction approach was presented, by constructing a quantization parameter map between encoders based on perceptual quality metrics. An encoder from an earlier generation can thus be used to predict the encoding parameters for one from a later generation. This further reduces the time needed to perform analysis. By predicting from x264 "veryfast" to libvpx speed 0, a simialr 20-40% BD-rate improvement over the traditional fixed-QP encoding can still be achieved, with only 3-4 p.p. loss. However, the analysis time is only 2% of what would be needed if libvpx speed 0 is used for analysis, and compared to the traditional fixed-QP encoding, only 11% additional time is needed to have such huge BD-rate improvement.

The prediction approach opens up various possibilities for determining optimal encoding parameters. We plan to continue exploring techniques to further improve and reduce the 3-4 p.p. BD-rate loss in the cross-codec prediction, while keeping the complexity as low as possible.

Acknowledgement

We would like to thank our colleagues within Facebook's Video Infrastructure, as well as RTP and CEA teams for their help in running all these experiments.

References

[1] I. Katsavounidis and L. Guo, "Video codec comparison using the dynamic optimizer framework," in *Applications of Digital Image Processing XLI*, A. G. Tescher, Ed., vol. 10752, International Society for Optics and Photonics. SPIE, 2018, pp. 266 – 281. [Online]. Available: https://doi.org/10.1117/12.2322118

[2] P.-H. Wu, V. Kondratenko, and I. Katsavounidis, "Fast encoding parameter selection for convex hull video encoding," in *Applications of Digital Image Processing XLIII*, A. G. Tescher and T. Ebrahimi, Eds., vol. 11510, International Society for Optics and Photonics. SPIE, 2020, pp. 181 – 194. [Online]. Available: https://doi.org/10.1117/12.2567502

[3] G. Chaudhari, H. Lalgudi, and H. Reddy, "Cross-codec encoding optimizations for video transcoding," in *Applications of Digital Image Processing XLIII*, A. G. Tescher and T. Ebrahimi, Eds., vol. 11510, International Society for Optics and Photonics. SPIE, 2020, pp. 60 – 68. [Online]. Available: https://doi.org/10.1117/12.2569294

[4] I. Katsavounidis, "The NETFLIX tech blog: Dynamic optimizer — a perceptual video encoding optimization framework," https://netflixtechblog.com/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f, Mar. 2018.

[5] M. Takeuchi, S. Saika, Y. Sakamoto, T. Nagashima, Z. Cheng, K. Kanai, J. Katto, K. Wei, J. Zengwei, and X. Wei, "Perceptual quality driven adaptive video coding using jnd estimation," in *2018 Picture Coding Symposium (PCS)*, 2018, pp. 179–183.

[6] K. Lee and V. Rao, "Accelerating facebook's infrastructure with application-specific hardware," https://engineering.fb.com/data-center-engineering/accelerating-infrastructure/, 2019.

[7] S. L. Regunathan, H. Wang, Y. Zhang, Y. R. Liu, D. Wolstencroft, S. Reddy, C. Stejerean, S. Gandhi, M. Chen, P. Sethi, A. Puntambekar, M. Coward, and I. Katsavounidis, "Efficient measurement of quality at scale in Facebook video ecosystem," in *Applications of Digital Image Processing XLIII*, A. G. Tescher and T. Ebrahimi, Eds., vol. 11510, International Society for Optics and Photonics. SPIE, 2020, pp. 69 – 80. [Online]. Available: https://doi.org/10.1117/12.2569920

[8] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "The NETFLIX tech blog: Toward a practical perceptual video quality metric," link: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652, 2016.