FAUST: Fast Per-Scene Encoding Using Entropy-Based Scene Detection and Machine Learning

Anatoliy Zabrovskiy*§, Prateek Agrawal†, Christian Timmerer*‡, and Radu Prodan*
* University of Klagenfurt, Austria
† Lovely Professional University, Punjab, India
‡ Bitmovin, Klagenfurt, Austria
§ Petrozavodsk State University, Petrozavodsk, Russia
Email: *§anatoliy.zabrovskiy@aau.at, †prateek.agrawal@aau.at
*‡christian.timmerer@aau.at, *radu.prodan@aau.at

***Abstract*—HTTP adaptive video streaming is a widespread and sought-after technology on the Internet that allows clients to dynamically switch between different stream qualities presented in the bitrate ladder to optimize overall received video quality. Currently, there exist several approaches of different complexity for building such a ladder. The simplest method is to use a static bitrate ladder, and the more complex one is to compute a per-title encoding ladder. The main drawback of these approaches is that they do not provide bitrate ladders for scenes with different visual complexity within the video. Moreover, most modern methods require additional computationally-intensive test encodings of the entire video to construct the convex hull, used to calculate the bitrate ladder. This paper proposes a new fast per-scene encoding approach called FAUST based on 1) quick entropy-based scene detection and 2) prediction of optimized bitrate ladder for each scene using an artificial neural network. The results show that our model reduces the mean absolute error to 0.15, the mean square error to 0.08, and the bitrate to 13.5% while increasing the difference in video multimethod assessment fusion to 5.6 points.**

I. Introduction

The number of adaptive video streaming applications and platforms is constantly increasing. It becomes more demanding to encode, distribute, share, and consume video streams anywhere, anytime, and on any device. The main goal of such streaming platforms is to provide the optimized streaming quality to the client, considering the client's preferences and technical constraints [1], [2]. To achieve this goal, streaming providers and services use various modern technologies and standards, such as HTTP Adaptive Streaming (HAS) [3], video codecs [4], and calculating the optimized bitrate ladders for video files [5]. In HAS, a video stream consists of multiple representations split into short video segments typically between 2 s to 10 s, adaptively transmitted to a client device depending on many influencing features, such as network bandwidth or display resolution [3], [6], [7]. The HAS technology allows compensating network speed fluctuations by switching between available representations [8]. In some implementations, such as MPEG-DASH [6], HAS is independent of video codecs and can utilize various video compression formats. This advantage allows choosing among several video codecs, such as Advanced Video Coding (AVC) [9], High Efficiency Video Coding (HEVC) [10], VP9 [11], AOMedia Video 1 (AV1) [12], and Versatile Video Coding (VVC) [13]. In addition, the methods used to calculate a suitable bitrate ladder play a vital role in HAS.

The most general and straightforward approach for selecting bitrate-resolution pairs is using a 'one-size-fits-all' concept (or fixed/static/classic bitrate ladder) [5]. Such a static bitrate ladder has a pre-defined and fixed set of representations with various bitrate-resolution pairs covering multiple visual qualities. Table I presents one example of a static bitrate ladder from Apple's video encoding documentation [14], widely used because of its simplicity and ease of deployment. However, encoding the entire video using a static bitrate-resolution pair results in different visual qualities of the encoded segments [15]. This happens because limiting the static bitrate prevents the video encoder from increasing the value for more complex video scenes and vice-versa.

Besides using a simple static bitrate ladder, more sophisticated methods use video complexity analysis, test (or trial) encodings, and machine learning techniques to compute the optimized bitrate ladder for a video sequence [16]. Netflix introduced an approach called per-title encoding [5] to improve the video quality, such as PSNR [17] and VMAF [18], by choosing the most appropriate bitrates for videos depending on their complexity. The per-title bitrate ladder outperforms the static bitrate ladder in the overall objective quality and reduced storage [5]. Several variants of the per-title encoding, also known as per-title, per-shot, per-scene, and context-aware, include proprietary industry [19]–[21] and research solutions [22], [23]. Unfortunately, these solutions still rely on multiple test video encodings to calculate the bitrate-resolution ladder for a video [15], [24]. This typically results in more intensive use of computing units, usually located in the cloud, and increases the computation time and cost. Moreover, most methods calculate the bitrate ladder for the entire video while a single video can contain many video scenes and even segments of completely different visual complexity [25], [26]. Thus, it is advisable to split the video into scenes of similar visual complexity and analyze them separately. Again, splitting high-definition video into scenes of different complexity requires

ISSN 2305-7254

TABLE I. APPLE STATIC BITRATE LADDER (BITRATE-RESOLUTION PAIRS).

No.	*Bitrate [kbps]*	*Resolution [width × hight]*
1	145	416 x 234
2	365	640 x 360
3	730	768 x 432
4	1100	768 x 432
5	2000	960 x 540
6	3000	1280 x 720
7	4500	1280 x 720
8	6000	1920 x 1080
9	7800	1920 x 1080

significant computational resources and requires analyzing video frames.

To decrease the scenes detection time and optimized bitrate ladder calculation, we propose a novel and accurate method called *fast per-scene encoding using entropy-based scene detection and machine learning (FAUST)*, which consist of two phases: 1) entropy-based scene detection, and 2) optimized per-scene bitrate ladders calculation. FAUST is a fast method that calculates a temporal information metric and its entropy for video sequences encoded at low bitrate and resolution, which significantly saves computation time. Our approach follows previous research [25], which showed a high correlation between the temporal information of original videos and its low bitrate and resolution encodings. We use FAUST for per-scene encoding optimization and adaptive streaming video with multiple media presentation descriptions where adaptation at the video scene level is possible [27].

The main contributions of FAUST are:

- We proposed a fast entropy-based video scene detection approach for the x264 video codec that uses *temporal information (TI)* [28] of the video encoded at low resolution and bitrate to split videos into scenes.
- We introduced a neural network for predicting the quality metric for video segments with different sets of input features, and minimized the mean absolute error to 0.15 and the mean square error (MSE) to 0.08.
- We introduced a fast per-scene encoding that does not require multiple test encodings of the entire video, which allowed us to reduce the bitrate to 13.5% on average while increasing the difference in video multimethod assessment fusion (VMAF) quality to 5.6 points.

The paper has six sections. Section II highlights the related work. Section III explains the FAUST methodology and Section IV describes its implementation. Section V evaluates the experimental results, concluded in Section VI with an outlook for future work.

II. RELATED WORK

Recent research [12], [20], [21], [29]–[34] presented different proprietary solutions dedicated to generating optimized bitrate ladders for adaptive streaming.

Katsavounidis *et al.* [35] presented a perceptual video encoding optimization that splits the video into shots with similar visual complexity frames. They performed multiple trial encodings of the video shots using different pre-defined parameters, such as resolutions and qualities and calculated a convex hull to generate an optimized bitrate ladder.

Silhavy *et al.* [22] applied random forest regression (RFR), multilayer perceptron, and support vector regression (SVR) to generate predicted per-title bitrate ladder using the VMAF video quality metric. Similar to our approach, they avoided test encodings of the videos. The results showed that RFR has the lowest Root Mean Square Error (RMSE) for selected bitrates and scores a VMAF prediction with a value of 3.22.

Cock *et al.* [23] presented a cloud method that enhances the per-title ladder by identifying the best-fit bitrate for all video segments (or chunks) depending on their complexity using a Constant rate factor (CRF) based multi-pass video encoding. They showed that the per-chunk encoding improved the video quality and outperformed fixed bitrate ladder encoding. They obtained the gains decrease for higher resolutions (1080p encodes) to about 0.3 dB for PSNR metric and 0.45 for VMAF.

Takeuchi *et al.* [15] to optimize the bitrates proposed an encoding solution based on SVR, test encodings, and perceptual video quality. They generated multiple bitrate-resolution pairs (encoding recipe), keeping a constant just-noticeable difference [36] gap. The solution had better storage and bitrate reduction than the conventional static bitrate ladders. For most videos, their method achieves smaller storage sizes compared to the static bitrate ladder. The average BD-Rate [37] is -34.3%.

Bhat *et al.* [38] proposed a real-time adaptive resolution prediction for the HEVC codec and low bitrate scenarios. They analyzed the first few frames of a video sample and use binary classification to find suitable resolutions for video encoding. The results indicated an average bitrate saving from 2.3% to 12.6%, depending on the video resolution.

Covell *et al.* [39] estimated an optimal constant rate factor for a specific bitrate of a video segment using an artificial neural network (ANN). They analyzed the relationship between the encoding parameters and the video complexity and achieved an accuracy of 20% bitrate error.

Angeliki *et al.* [40] proposed a content-agnostic HEVC codec that uses machine learning to predict the bitrate ranges for selected resolutions based on original temporal and spatial video characteristics. The method reduced the number of test encodings to determine the values for a bitrate ladder and reached a mean Bjontegaard Delta-Rate RSNR (BDRSNR) [37] of $0.01\,\mathrm{dB}$ compared to the ground truth.

Kumar *et al.* [24] proposed a per-scene optimization framework for video on demand HAS that determines the maximum quality or minimum bitrate for various encoded representations. To detect video scenes, the authors used a threshold-based algorithm implemented in an intelligent scene cut detection and video splitting tool called PySceneDetect [41]. The framework adjusted the bitrate ladder depending on the

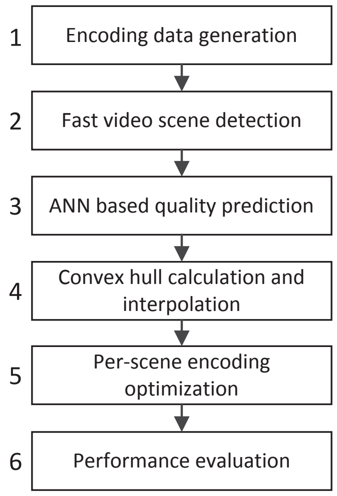
1 Encoding data generation
2 Fast video scene detection
3 ANN based quality prediction
4 Convex hull calculation and interpolation
5 Per-scene encoding optimization
6 Performance evaluation

Fig. 1. Process flow of proposed methodology

complexity of video content and reduced the bitrate of video streams by up to 10% while preserving the video quality.

Most of the existing methods calculate optimized bitrate ladders for a given video sequence using multiple test encodings [23], [24], [35]. In addition, scene detection on the original high resolution videos [42] leads to more intensive use of computational units, and increases the time and cost of the encoding process [43]. Advanced methods use machine learning techniques to predict the video quality and avoid test encodings [22]. However, the disadvantage of such methods is their low accuracy and speed. In general, these methods calculate the optimized bitrate ladder for the entire video sequence rather than for individual video scenes [22].

III. Methodology

This section presents our methodology comprising six sequential phases, shown in Fig. 1.

A. Encoding data generation

This phase selects videos of varying complexity of content, which should contain sequences of video frames with different spatial and temporal information [28].

The *spatial information (SI)* is a measure of the spatial video complexity, calculated as follows:

$$SI = \max_{\forall F_n}\{\sigma\left[Sobel\left(F_n\right)\right]\}, \tag{1}$$

where F_n is a video frame at time n, and σ is the standard deviation across all the pixels in the *Sobel* filter of its luminance component. The luminance component is a measure of the intensity (or brightness) of a video signal. The $\max$ function then selects the maximum standard deviation across all the frames presented in the video sequence.

The *temporal information (TI)* reflects the amount of motion in a video, defined based on a *motion difference* function

Algorithm 1: Algorithm for detecting video scenes. It calculates the timestamps of starting points of video scenes.

```
Input: $Video\ duration\ in\ sec.\ (duration)$
Input: $Threshold\ value\ (threshold)$
Output: $Array\ of\ timestamps\ for\ starting\ points$
        $of\ video\ scenes$
// Indexing of arrays duration[] and
   timestamps[] starts from one
1  $avg\_ti[\,] \leftarrow \{'val1','val2','val3',...'valP'\}$
2  $avg\_entropy[\,] \leftarrow \{'val1','val2','val3',...'valN'\}$
3  $timestamps[\,] \leftarrow null$
4
5  Function
     calc_timestamps($avg\_ti, avg\_entropy$):
6      $output[\,] \leftarrow null$
7      for $i$ = $0$ to $duration$ do
8          if $avg\_ti[i] \geq threshold$ then
9              if $avg\_entropy[i] \geq threshold$ then
10                 $output \leftarrow i$
11     end
12     return $output$
13 End Function
14 $timestamps \leftarrow$
     $calc\_timestamps(avg\_ti, avg\_entropy)$
```

M_n between the luminance components for identically spaced pixels in two sequential frames F_n and F_{n-1}:

$$M_n(i,j) = F_n(i,j) - F_{n-1}(i,j), \tag{2}$$

where $F_n(i,j)$ is the frame pixel located at row i and column j at time n in the sequence. The TI metric is the maximum standard deviation of $M_n(i,j)$ calculated for all the pixels:

$$TI = \max\{\sigma\left[M_n(i,j)\right]\}. \tag{3}$$

Since adaptive and dynamic switching between representations usually starts at segment boundaries [44], the video segment length becomes an important feature in adaptive video streaming. Thus, the next step is therefore to create segments from the selected video sequences between $2\,\mathrm{s}$ to $4\,\mathrm{s}$ in length, as commonly used in industry and research [44], [45]. The $4\,\mathrm{s}$ segments show a good trade-off between encoding efficiency and video streaming performance [45], and $2\,\mathrm{s}$ segments are typically used for low-latency streaming [44]. We then extract the important features collected from the encoded video segments for the dataset, such as segment name, input segment size, resolution, etc.

B. Fast video scene detection

This phase detects video scenes using an entropy-based approach that analyzes the average TI and TI entropy every second and defines the boundaries between scenes. In general, entropy is a term from information theory that shows the uncertainty or disorder of the variable associated with an event [46]. It is a measure of the average information required to describe the random variable. In other words, the entropy value reflects the predictability of a variable, with high entropy values for less predictable events. Let us assume a random variable X representing the TI of a video sequence and x_i one possible values of X. We model the entropy of the TI using the following equation:

$$H(X) = -\sum_{i=1}^{n} P\left(x_i\right) \cdot \log \mathcal{P}\left(x_i\right), \tag{4}$$

where $\mathcal{P}(x_i)$ is the appearance probability of event x_i. Based on information theory and Eq. 4, an event with low probability $\mathcal{P}\left(x_i\right)$ is more informative and has high entropy. In our case the mean TI and entropy of TI values are normalized between 0 and 1. A new video scene is detected if both average TI and entropy TI reach a fixed threshold. By changing value of *threshold* the sensitivity of the scene detection can be modified. A lower threshold usually detects more scenes. In our work, we used a threshold value of 0.5, which means that if the mean TI and entropy TI for two consecutive seconds are greater than or equal to 50%, then a new scene is detected. We proposed an Algorithm 1 for the video scene detection. The result of this algorithm is an array of timestamps for the starting points of the video scenes. We merged the short scene with the next longer video scene in a video because the duration of the video scene should be kept at least the length of the video segment. Finally, the output of this phase includes information about the location, number, and duration of scenes in the video sequence.

C. ANN based quality prediction

This phase prepares training and testing data and predicts a full-reference metric called Luminance Peak Signal-to-Noise Ratio (YPSNR) of video segments using ANN. We predict the YPSNR video quality, as it is commonly used for an objective assessment of video quality. [4]. Before training the ANN, we pre-process the dataset generated in the first phase and reduce the number of records by calculating the minimum YPSNR for all combinations of segment name, width, height, and encoding preset. Then, we split the dataset in training and testing subsets.

We designed the ANN [47] by choosing a sequential model where only the first input layer receives the actual variables, and the hidden layers automatically detect the input shapes. Next, we selected the appropriate activation functions [48] for training it. We used a linear activation function for the output layer as it does not limit the output of ANN to any range, and a rectified linear unit (ReLU) non-linear activation function for the hidden layers. The ReLU function is half-rectified and converts the value to zero for all negative inputs. We used the MAE that represents the absolute model prediction error in units of the variable, and the MSE the squared average difference between the actual and the predicted values to

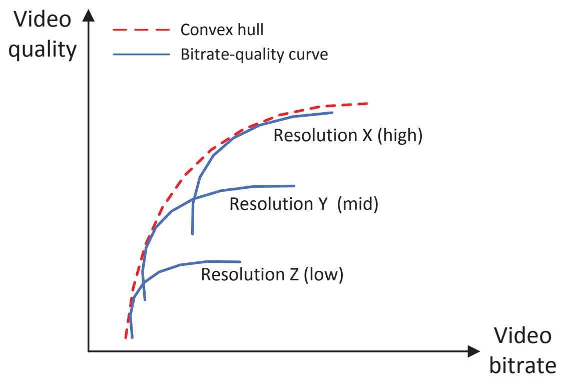

Fig. 2. Example of convex hull. The figure adapted from [5]

evaluate the ANN output results,

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \tag{5}$$

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 \tag{6}$$

where, n is a number of predicted qualities, y_j is the predicted video quality and $\hat{y}_j$ is the actual video quality. We tune and update the training features of the ANN using MAE and MSE, and repeat this process until we obtain consistency and accuracy. Finally, we use the developed ANN model to predict the YPSNR for all input instances of the dataset.

D. Convex hull calculation and interpolation

This phase calculates and interpolates the convex hulls for the middle segments of all detected scenes. Fig. 2 presents one schematic example of a convex hull for a single video sequence. The convex hull is a red boundary line representing the relationship between encoding bitrate and video quality for different video resolutions. Typically, the convex hull line points achieve Pareto efficiency and ideally have to be used for an optimized bitrate ladder construction. To calculate the convex hull, different approaches can be utilized. The most popular method is making several test encodings of a video using different encoding features [5]. In turn, our system does not require any trial encodings because ANN provides all information for convex hull construction. In our FAUST approach, we assume that the complexity of the video content of each scene is reflected by one middle segment. Thus, we predict and calculate the convex hull for all video scenes using middle segments and the ANN. If the video scene has an odd number of segments, say 7, the middle segment will be number 4. In turn, if the video scene has an even number of segments, say 6, then we consider segment 4 as the middle segment. The FAUST approach using the ANN can automatically predict the YPSNR quality for selected bitrate-resolution pairs, taking into account the visual content complexity and x264 codec encoding features. It then applies cubic spline interpolation [49]

Algorithm 2: Algorithm for selecting video bitrate from the classic bitrate ladder for comparison with the optimized bitrate ladder.

```
Input: $Classic\ ladder\ bitrates\ (c\_bitrates)$
Input: $Optimized\ ladder\ bitrates\ (o\_bitrates)$
Output: $Selected\ bitrates\ array\ (selected\_bitrates)$
// Indexing of arrays c_bitrates[]
   and o_bitrates[] starts from zero
$c\_bitrates[\,] \leftarrow \{'val1','val2','val3',...'valP'\}$
$o\_bitrates[\,] \leftarrow \{'val1','val2','val3',...'valN'\}$
$number\_classic \leftarrow lenght(c\_bitrates[\,])$
$number\_optimized \leftarrow lenght(o\_bitrates[\,])$
$selected\_bitrates[\,] \leftarrow null$

Function select($c\_bitrates, o\_bitrates$):
    $output[\,] \leftarrow null$
    for $i$ = 0 to number_optimized do
        for $j$ = 0 to number_classic do
            if $c\_bitrates[j] \geq o\_bitrates[i]$ then
                $output \leftarrow c\_bitrates[j]$
        end
    end
    return output
End Function
$selected\_bitrates \leftarrow select(c\_bitrates, o\_bitrates)$
```

to the convex hull points predicted by the ANN to estimate unknown values that fall between the known values.

E. Per-scene encoding optimization

In this phase, we calculate the individual bitrate ladders for all video scenes using predicted and interpolated convex hulls. For each video scene, we select all points on the interpolated convex hull that belong to the specific YPSNR quality range. We use the selected video quality spacing *e.g.*, 1, 1.5, or 2 quality points to calculate the number of bitrate-resolution pairs and values of quality targets for each scene. For the selected quality points (or targets), our program finds the appropriate bitrates and resolutions using an interpolated convex hull. The FAUST approach output includes optimized bitrate ladders for all scenes of an input video file.

F. Performance evaluation

This phase compares our per-scene encoding approach with the classic bitrate ladder using three key metrics: bitrate reduction, storage reduction, and quality difference [50]. We calculate the *bitrate reduction* B_r metric by comparing the bitrate values used at different resolutions along the bitrate ladder.

We proposed an Algorithm 2 to select the bitrate values from the classic bitrate ladder. The algorithm selects the equal or the next highest bitrate value from the classic bitrate ladder for all bitrates from the optimized bitrate ladders. It then compares the sum of all bitrates of optimized per-scene bitrate ladders with the sum of selected bitrates from the classic bitrate ladder and calculates the deviation percentage (bitrate reduction) as shown in Eq. 7.

$$B_r = \frac{\sum b_{classic} - \sum b_{optimized}}{\sum b_{classic}} \cdot 100 \qquad (7)$$

Similar to the equation Eq. 7, we calculate the *storage reduction* metric as a deviation percentage for the sums of the encoded file sizes. To compare video quality using both approaches, we calculate the *quality difference* between the averages of video quality metrics (*i.e.*, YPSNR [4], VMAF [18], and XPSNR [51]) measured for the encoded video scenes using classical and optimized bitrate ladders.

IV. Implementation

This Section describes the implementation of our proposed FAUST methodology. We run the experiments using Intel i7-3720QM processor, FFmpeg software, and Python 3.6 to calculate SI, TI, and YPSNR quality metric.

A. Encoding data generation

We first selected ten videos (see Table II) from the public dataset [17]. Table II shows the main characteristics of original video sequences. The Fig. 3 presents the SI and TI metrics of the selected video sequences. The average TI and SI metric values confirm the varying complexity of the video content. Thus, we can argue that we used video sequences that represent a wide range of possible use cases and visual differences. Using the FFmpeg [52] program v4.1.3, we divided all videos into 240 video segments of 2 s and 4 s duration length. All video segments are in raw Y4M video format.

After creating the raw Y4M segments, we encoded each segment using the FFmpeg x264 codec implementation with the *veryslow* encoding preset to maintain the highest quality compared to the original video. The x264 video codec contains the set of encoding presets as follows: *ultrafast, superfast, veryfast, faster, fast, medium (default preset), slow, slower, veryslow, placebo*; such that, for the same video sequence and encoding bitrate with a slower x264 preset, typically there will be a slower encoding speed and better video quality. According to the FFmpeg technical manual [53], we did not use the *placebo* preset as it does not significantly improve quality over *veryslow* preset. Further in this paper, we consider all of these prepared video segments as source files and use them for the encoding.

Our work is primarily about *HTTP Adaptive Streaming* and thus, we utilized the *bitrate ladder* as presented in Table III. The selected bitrate ladder covers a wide range of encoding bitrates and resolutions. This selection is based on publicly available datasets proposed in the literature [54], industry use cases, and video encoding recommendations [55], [56]. We created the raw ANN training/testing dataset with 451440 encoding tasks (*240 segments * 19 bitrates * 11 resolutions * 9 encoding presets*) for the segments of 2 s and 4 s length. Creating a dataset with all possible combinations of encoding features typically contributes to better training and accuracy

TABLE II. ORIGINAL VIDEO FILE CHARACTERISTICS

Video description	Video category	Frames per second	Duration (in sec)
BBB	Animation	30	60
Beauty	Moving head	30	20
DrivingPOV	Moving cars	60	20
HoneyBee	Nature (flying bee)	30	20
Jockey	Sports (running jockey)	30	20
Sintel	Animation	24	60
TOS	Animation and real	24	60
WindAndNature	Rotating wind vanes	60	20
ReadySetGo	Sports (horse racing)	30	20
YachtRide	Moving yacht	30	20

TABLE III. BITRATE LADDER (BITRATE-RESOLUTION PAIRS). BITRATE VALUES ARE IN KBPS

#	Bitrate	Resolution	#	Bitrate	Resolution
1	100	256x144	11	4300	1920x1080
2	200	320x180	12	5800	1920x1080
3	240	384x216	13	6500	2560x1440
4	375	384x216	14	7000	2560x1440
5	550	512x288	15	7500	2560x1440
6	750	640x360	16	8000	3840x2160
7	1000	768x432	17	12000	3840x2160
8	1500	1024x576	18	17000	3840x2160
9	2300	1280x720	19	20000	3840x2160
10	3000	1280x720			

of a neural network model therefore we used all 11 unique resolution values from the same bitrate ladder table for each encoding bitrate as presented in Table III. In addition, we calculated SI and TI metrics for all segments encoded at low resolution (144p) and low bitrate (100 kbps) using the *ultrafast* encoding preset. Calculating TI and SI metrics for the original high-resolution video sequences is a very time-consuming operation. Therefore, we used a fast approach introduced in [25] which computes SI and TI metrics more than ten times faster for video segments with low resolution. The Pearson's [57] correlation coefficient between encoded segments (144p) and original video segments (2160p) for TI and SI metrics indicates a positively strong and highly correlated relationship, respectively [25]. Finally, each entry in our raw encoding dataset contains the following fields: *segment_name, input_segment_size, segment_size_144p, encoding_bitrate, si_144p, ti_144p, segment_duration, width, height, encoding_preset, fps, and ypsnr_quality*.

B. Fast video scene detection

We implemented a proposed scene detection algorithm (see Algorithm 1) to find and split video sequences into visually independent scenes. Our approach analyzes the average value and entropy of the TI metric for each second of the video sequence and detects a new scene if both average TI and entropy TI reach a certain threshold. The scene detection algorithm uses the TI values for 144p encoded videos which are presented in our raw encoding dataset.

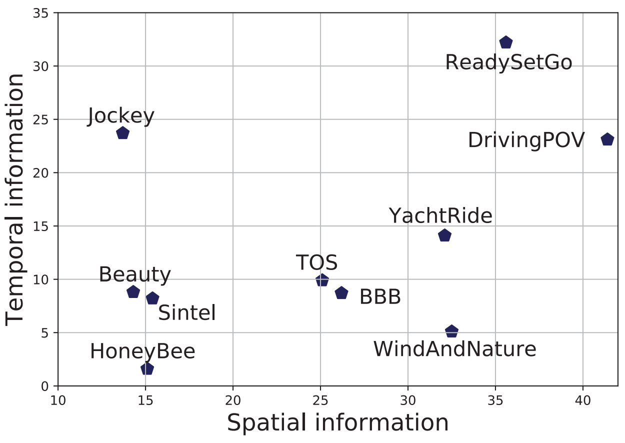

Fig. 3. Average spatial information (SI) and temporal information (TI) for video sequences.

TABLE IV. SELECTED INPUT SETS FOR THE ANN MODEL

ANN model	*No. of input set*						
input features	*1*	*2*	*3*	*4*	*5*	*6*	*7*
$height$	x	x	x	x	x	x	x
$width$	x	x	x	x	x	x	x
$number\_of\_pixels$	x	-	x	x	x	x	x
$encoding\_preset$	x	x	x	x	x	x	x
$encoding\_bitrate$	x	x	x	x	x	x	x
si_144p	x	x	-	x	-	x	-
ti_144p	x	x	x	-	-	x	-
$input\_segment\_size$	x	x	x	x	x	-	-
$segment\_size\_144p$	x	x	x	x	x	-	-
$segment\_duration$	x	x	x	x	x	x	x
fps	x	x	x	x	x	x	x

C. ANN based quality prediction

After identifying multiple video segment features and detecting all video scenes for three animation video sequences (BBB, Sintel, and TOS), we selected and grouped the most significant features from the raw encoding dataset to the ANN input sets. Note, we only used animated video sequences because there are multiple video scenes in them. We also derived a new feature for the ANN model called $number\_of\_pixels$ by multiplying video segment $width$ and $height$. We used an input set with the selected features for training and testing of neural network individually. In total, we selected seven sets of input data, as shown in the Table IV. For example, the first set contains all possible input features from *height* to *fps*, marked as *x*.

We pre-processed our raw encoding dataset. To do this, we first sorted the dataset in ascending order by two features *ypsnr_quality* and *encoding_bitrate* and then reduced the number of entries by taking only the first YPSNR quality

TABLE V. CHARACTERISTICS OF ANN MODELS

Characteristic	***Optimized value***
No. of neurons in input layer	from 7 to 11
No. of hidden layers	7
No. of neurons in hidden layers	300/128/128/128/128/300
No. of neurons in output layer	1
Learning rate	default value
No. of epochs passed	400
Training / testing / validation data	70%/20%/10%
Hidden layer activation function	ReLU
Output layer activation function	Linear
Optimizer	Adadelta
Loss function	MAE
Metric	MAE, MSE
Batch size	64
Initializing weights	truncated normal

TABLE VI. ANN MODELS INPUT FEATURES

Input feature	***Description***
$height$	video height (in pixels) from the Table III
$width$	video width (in pixels) from the Table III
$number\_of\_pixels$	video height * video width
$encoding\_preset$	ultrafast, superfast, veryfast, faster, fast, medium, slow, slower, veryslow
$encoding\_bitrate$	video bitrates (in kbps) from the Table III
si_144p	SI metric of 144p video
ti_144p	TI metric of 144p video
$input\_segment\_size$	original segment size in bytes
$segment\_size\_144p$	144p segment size in bytes
$segment\_duration$	2s and 4s segment length
fps	24, 30 and 60 from the Table II

for all possible combinations of [*segment_name, width, height, and encoding_preset*]. This means that the transformed dataset contains the minimum YPSNR quality values that can be achieved with the selected encoding features. Finally, we rounded up all YPSNR quality values to one decimal place and created a new dataset for training and testing the ANN with 436 973 records.

To predict the YPSNR quality for middle segments of detected video scenes, we used the Keras Python library [58] to implement the ANN model for all input sets (see Table IV) independently. The ANN models consist of seven to eleven input neurons, seven hidden layers, and one output neuron. We considered *Adagrad*, *Adadelta*, and *RMsprop* [59] optimization algorithms for the compilation of the ANN model and finally selected Adadelta after evaluating its performance with others. Adadelta optimization algorithm is an advanced optimizer of Adagrad, which adapts learning rate hyperparameter based on a moving window of gradient updates rather than collecting all past gradients' information. This specific property of Adadelta makes it better than other optimization algorithms to learn and adjust the learning parameters by default, even with multiple updates. Table V shows the ANN model characteristics and Table VI explains ANN input features in detail. We received predicted *YPSNR* quality as an ANN output and further used

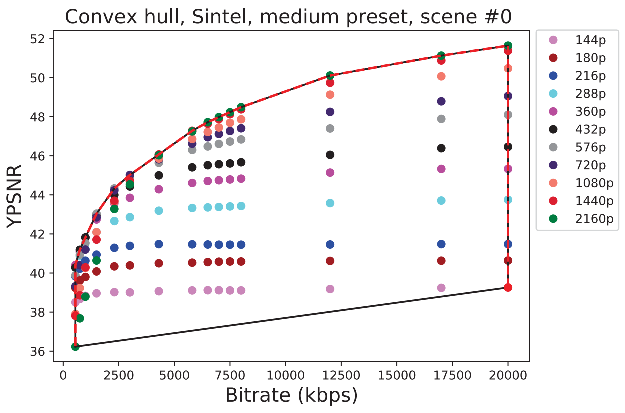

Fig. 4. The convex hull of the Sintel video sequence. Scene #0

it to calculate the convex hull. To assess the output results of ANN models, we used MAE and MSE metrics. We analysed the model performance in Section V in detail.

D. Convex hull calculation and interpolation

After developing and deploying the ANN model in the proposed system, we predicted YPSNR quality metrics for the middle video segment of each video scene for all possible combinations of bitrate and resolution presented in the Table III. In total, we predicted 209 *(19 bitrates * 11 resolutions)* quality values for each middle segment. For each series of predicted quality points, our proposed FAUST approach calculated the convex hull curve [60]. Fig. 4 shows a convex hull example of Sintel video for scene number zero. The same color points belong to the same resolution. For example, the blue color represents the video resolution of 216p, where 216 is the video height (see Table III). We applied the cubic spline interpolation function [49] to find and estimate the unknown values of the convex hull that fall between known values. We further used the predicted and interpolated convex hulls to select the appropriate bitrate-resolution pairs for the optimized bitrate ladders for each video sequence.

E. Per-scene encoding optimization

To build an optimized bitrate ladder for each video sequence scene, we used interpolated convex hulls of the middle segments. Our approach selects all points that belong to the interpolated convex hull range from 30 to 45 dB. Typically, the viewer does not perceive the YPSNR quality above 45 dB while values below 35 usually indicate visible artifacts [5]. We used a 30 dB lower bound for YPSNR metric in order to have lower bitrate(s) in optimized bitrate ladders to match the bandwidth constants of more clients. We used a 1.5-point YPSNR quality interval for each scene to calculate the number of bitrate-resolution pairs and quality targets for them. The higher the value of the selected quality interval between the two video qualities results, the smaller the number of bitrate in the resulting bitrate ladder. The resulting number of bitrate-resolution pairs varies for different scenes and depends directly

TABLE VII. THE TIME REQUIRED TO DETECT VIDEO SCENES FOR BBB VIDEO SEQUENCE USING THREE METHODS

Operation (in sec)	*FAUST approach*	*FFmpeg filter [61]*	*PySceneDetect tool [41]*
encoding to 144p	8.9	n/a	n/a
calculating TI	0.9	n/a	n/a
scene detection	0.1	27.7	34.0
Total time	**9.9**	**27.7**	**34.0**

TABLE VIII. THE TIME REQUIRED TO DETECT VIDEO SCENES FOR SINTEL VIDEO SEQUENCE USING THREE METHODS

Operation (in sec)	*FAUST approach*	*FFmpeg filter [61]*	*PySceneDetect tool [41]*
encoding to 144p	6.4	n/a	n/a
calculating TI	1.2	n/a	n/a
scene detection	0.1	21.7	25.5
Total time	**7.7**	**21.7**	**25.5**

on their visual complexity. For the selected quality targets, we find the bitrates and resolutions using an interpolated convex hull. Finally, we calculate a set of optimized bitrate ladders for all video scenes of the input video file.

V. RESULTS AND ANALYSIS

In this section, we present the results to analyze the performance and to examine the advantages of using the proposed FAUST approach for per-scene encoding optimization.

a) Fast scene detection: On real encoding platforms, various algorithms and tools are available to detect scenes in a video sequence. We compared our FAUST approach with two popular and optimized scene detection tools: *(i)* FFmpeg [61] and *(ii)* PySceneDetect [41].

- The FFmpeg filter analyzes the difference between video frames using a pre-defined threshold,
- PySceneDetect uses the *detect-threshold* method to find scene changes and split the video into individual scenes [24].

Table VII shows the performance comparison among three methods in terms of scene detection time of BBB video sequence. The results show that our proposed FAUST approach takes 8.9 s to encode BBB video for low resolution (144p) and bitrate (100 kbps). In addition, it takes 0.9 s to compute TI metric for low resolution (144p) video. However, the scene detection time is just 0.1 s, very short compared to the FFmpeg and PySceneDetect tools, since we analysed and detected the scenes for low resolution videos. The total scene detection time (see Table VII, last row) shows that the FAUST method detects scenes almost three times faster than FFmpeg and more than three times faster than PySceneDetect tool. Tables VIII and IX show the performance comparison for Sintel and TOS videos, respectively. We see that for Sintel and TOS videos for all three methods, the scene detection time is lower than for the BBB video sequence. This is because the frame rate for Sintel and TOS is 24 and for the BBB it is 30. Fewer

TABLE IX. THE TIME REQUIRED TO DETECT VIDEO SCENES FOR TOS VIDEO SEQUENCE USING THREE METHODS

Operation (in sec)	*FAUST approach*	*FFmpeg filter [61]*	*PySceneDetect tool [41]*
encoding to 144p	6.6	n/a	n/a
calculating TI	0.9	n/a	n/a
scene detection	0.1	19.9	25.4
Total time	**7.6**	**19.9**	**25.4**

TABLE X. THE NUMBER OF SCENES DETECTED FOR THE DIFFERENT THRESHOLDS

Threshold	*BBB*	*Sintel*	*TOS*
0.45	14 / 10	9 / 7	7 / 6
0.50	11 / 8	6 / 4	7 / 6

frames in video faster processing time. The number of scenes detected for Sintel, BBB and TOS videos, for two different thresholds are shown in Table X. The results include two values for each video sequence and threshold, for example 9/7 for Sintel video with a threshold of 0.45. This means that the proposed approach initially detects nine scenes, out of which two are very short (less than 2 s) and thus combined with others, resulting in a total of seven scenes.

b) Convex hull prediction: We evaluated the ANN model on various input sets and tuned the model characteristics (see Table V) to find the most appropriate input features that minimize the MAE and MSE accuracy metrics. Table IV presents each input feature set and Table XI shows the MAE and MSE results for all seven input sets. The results show that ANN model with 2^{nd} set of input features predicts the optimized video quality by minimizing MAE and MSE up to 0.15 and 0.08, respectively. The 2^{nd} set includes all default input features except for *number_of_pixels* but it takes video *width* and *height*. *number_of_pixels* is a multiplication of video *width* and *height* so indirectly, it takes all the ANN input features for the video quality prediction. The 6^{th} input features set (without files size) achieves second highest performance with 0.15 MAE and 0.09 MSE and compromises slightly with 2^{nd} set performance in terms of MSE value. Interestingly, the results for the ANN model with the 7^{th} set of input features (without TI, SI and file size information) produce the worst performance with 2.98 MAE and 14.1 MSE. It shows that the excluded features in the 7^{th} set have the maximum impact on the ANN model performance. Thus, we see that overall 2^{nd} input set achieves maximum performance followed by 6^{th} set, 4^{th} set, 1^{st} set, 3^{rd} set, 5^{th} set and 7^{th} set.

Finally, we selected the 2^{nd} input features set for ANN model to predict the quality of the video segment to construct further and interpolate the convex hull. Fig. 5 shows the predicted and actual interpolated convex hulls for the *medium* encoding preset and scene number 3 of the Sintel video sequence. The convex hull using predicted data is very similar to a convex hull using real data.

TABLE XI. MAE AND MSE METRICS FOR DIFFERENT SETS OF INPUT FEATURES OF THE ANN MODEL

ANN model input set	*MAE*	*MSE*	*Input set description*
1	0.16	0.10	default (all features)
2	**0.15**	**0.08**	w/o number of pixels
3	0.17	0.09	w/o SI
4	0.16	0.08	w/o TI
5	0.23	0.17	w/o SI and TI
6	0.15	0.09	w/o file size
7	2.98	14.1	w/o TI, SI, file size

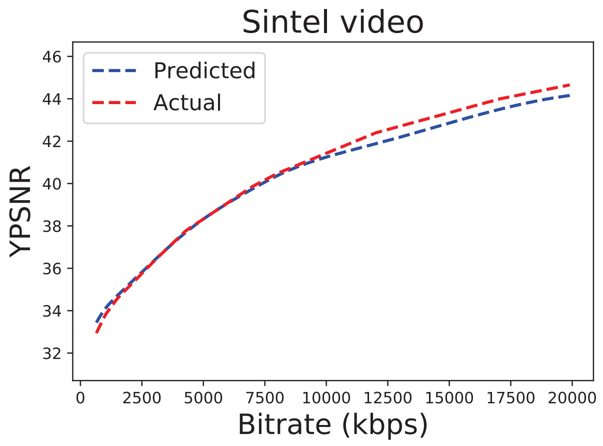

Fig. 5. Convex hull on predicted data (blue color) and actual data (red color) for Sintel video sequence, scene #3. Medium encoding preset

c) Performance analysis with classic bitrate ladder: Table XII shows the comparison between the proposed FAUST approach and the classic bitrate ladder on three key metrics: *bitrate reduction*, *storage reduction*, and *quality difference*. We described these key metrics in detail in the Section III. We selected the equal or next highest bitrate value from the classic bitrate ladder to compare with all bitrates of optimized bitrate ladders generated by the proposed FAUST approach. Then, we compared the sum of all the bitrates of the FAUST bitrate ladders for each scene with the sum of the selected bitrates from the classic bitrate ladder to calculate the overall bitrate reduction. The bitrate reduction depends on the streaming video use case and increases in multiples with the number of real remote viewers (or clients) and thus becomes a crucial performance metric.

Table XII shows the key metric results for the BBB, Sintel and TOS video sequence. The FAUST approach reduces the bitrate while maintaining a higher VMAF video quality for all tested video sequences compared to the classic bitrate ladder. The largest bitrate reduction is 15.6% for BBB video sequence, and the smallest bitrate reduction is 12.3% for TOS video. The average bitrate reduction and VMAF difference calculated for three testing video sequences are 13.5% and 5.6 points, respectively. The average difference values for VMAF, YPSNR, and XPSNR quality for all three videos are positive, which shows that the proposed FAUST approach improves both the bitrate reduction and the overall video quality.

TABLE XII. COMPARISON OF THE FAUST APPROACH VERSUS THE CLASSIC BITRATE LADDER FOR BBB, SINTEL AND TOS VIDEO SEQUENCES

Metric	*BBB*	*Sintel*	*TOS*	***Average value***
Bitrate reduction (in %)	15.6	12.5	12.3	**13.5**
Storage reduction (in %)	10.5	12.4	11.7	**11.5**
VMAF difference	4.7	3.7	8.3	**5.6**
YPSNR difference	1.2	0.3	1.5	**0.6**
XPSNR difference	-0.1	0.0	0.7	**0.2**

TABLE XIII. PERFORMANCE COMPARISON BETWEEN FAUST AND MIPSO [24]

	MiPSO approach	*FAUST approach*
Number of video sequences	3	3
Scene detection time for BBB	34.0 sec	9.9 sec
Convex hull calculation	slow	fast
Minimum bitrate reduction	10%	12.3%
VMAF quality difference	6.2-7.7	3.7-8.3

d) State-of-the-art comparison: We analyzed and compared the performance of the proposed FAUST approach with a state-of-the-art framework called MiPSO (*Multi-Period Per-Scene Optimization For HTTP Adaptive Streaming*) presented by Kumar *et al.* [24]. For this, we investigated and compared their scene detection method with our entropy-based scene detection approach. The MiPSO approach utilizes the threshold-based mode of PySceneDetect [41] tool for video scenes detection. The threshold-based mode analyzes the original high resolution video sequence for changes in the average frame brightness and requires more computing resources and processing time than our FAUST approach. Table. XIII shows the detailed comparison of both approaches for the BBB video sequence in terms of different features and performance parameters. We see that the bitrate reduction and VMAF quality difference are very similar for both approaches, but the video scene detection time using our FAUST approach (9.9 s) is more than three times faster than the MiPSO framework algorithm, i.e. 34 s. The major reason behind the performance similarity is because both the approaches use an interpolated convex hull to calculate the bitrate ladder for each scene, and the test video sequences are also identical. The major advantage of our proposed FAUST approach is that it can detect video scenes very quickly compared to the MiPSO framework. Moreover, there is no need to run multiple tests (or trial) encoding to build a convex hull to calculate per-scene bitrate ladders with the FAUST approach. Instead of numerous test encodings, the proposed FAUST approach constructs the convex hull using the ANN predicted video quality, that significantly saves time and computational resources. For example, the FAUST approach for the Sintel video sequence takes 0.4 seconds to make all the quality predictions for the convex hull calculation on an Intel Core i5 processor while with MiPSO based approach, this will take much longer, as test video encodings are required.

e) Summary: The results show that the proposed FAUST approach requires relatively little time to calculate the optimized per-scene bitrate ladders for a video. Additionally, the proposed FAUST approach predicts the video quality metric for the selected video segments by minimizing MAE and MSE to 0.15 and 0.08, respectively. Finally, we can summarise that the FAUST approach has a higher bitrate reduction than the state-of-the-art approach.

VI. Conclusions and Future Work

In this research, we propose a novel approach called FAUST to quickly and accurately calculate optimized bitrate ladder for each video scene. The FAUST approach includes fast entropy-based scene detection and segment quality prediction using the ANN model. We performed scene detection on low bitrate and resolution versions of video sequences, which greatly speeds up the processing time. We developed the ANN model based on encoding dataset generated using x264 video codec, multiple encodings with different bitrates, presets and resolutions. Our results show that the selected set of ANN model input features gives an improved prediction of video quality. The proposed ANN model predicts the YPSNR quality metric for the segments, minimizing MAE and MSE to 0.15 and 0.08, respectively. In particular, the results show that our FAUST approach, with fast scene detection and ANN based video quality prediction, can reduce bitrate to 13.5% while increasing the difference in VMAF quality to 5.6 points.

Future work includes experiments on new emerging video codecs and adding more sophisticated approaches for multiple quality metrics prediction, such as intelligently selecting and analyzing the content complexity of a few very short samples of a video scene to make prediction about the required bitrate ladder for the entire scene. Additionally, we will investigate using the data of the previously optimized per-scene bitrate ladders for different scenes to predict the convex hull for a new video scene with a similar video content complexity.

Acknowledgment

This work has been supported by the Austrian Research Promotion Agency (FFG) under the *APOLLO* project.

References

[1] D. Rayburn, “Streaming quality survey report: determining what impacts end user video quality the most,” 2020, [Edgeware Report, https://www.edgeware.tv/, accessed 23-Aug-2021].

[2] A. Erfanian, F. Tashtarian, A. Zabrovskiy, C. Timmerer, and H. Hellwagner, “OSCAR: On Optimizing Resource Utilization in Live Video Streaming,” *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[3] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, “A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 562–585, Firstquarter 2019.

[4] A. Zabrovskiy, C. Feldmann, and C. Timmerer, “A Practical Evaluation of Video Codecs for Large-Scale HTTP Adaptive Streaming Services,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 998–1002.

[5] Anne Aaron and others, “Per-Title Encode Optimization,” https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2, 2015, [Online; accessed 23-August-2021].

[6] I. Sodagar, “The MPEG-DASH Standard for Multimedia Streaming Over the Internet,” *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1109/MMUL.2011.71

[7] A. Zabrovskiy, E. Petrov, E. Kuzmin, and C. Timmerer, “Evaluation of the performance of adaptive HTTP streaming systems,” *CoRR*, vol. abs/1710.02459, 2017. [Online]. Available: http://arxiv.org/abs/1710.02459

[8] A. Zabrovskiy, E. Kuzmin, E. Petrov, C. Timmerer, and C. Mueller, “AdViSE: Adaptive Video Streaming Evaluation Framework for the Automated Testing of Media Players,” in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017, pp. 217–220.

[9] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC Video Coding Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[10] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[11] D. Mukherjee, J. Han, J. Bankoski, R. Bultje, A. Grange, J. Koleszar, P. Wilkins, and Y. Xu, “A Technical Overview of VP9: The Latest Open-Source Video Codec,” in *SMPTE 2013 Annual Technical Conference Exhibition*, Oct 2013, pp. 1–17.

[12] C. Chen, Y. Lin, S. Benting, and A. Kokaram, “Optimized transcoding for large scale adaptive streaming using playback statistics,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 3269–3273.

[13] B. Bross, J. Chen, and S. Liu, “Versatile Video Coding (Draft 7),” JVET-P2001, Geneva, CH, Document, Oct. 2019.

[14] Apple, “HLS Authoring Specification for Apple Devices,” https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices, 2020, [Online; accessed 23-August-2021].

[15] 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.

[16] A. V. Katsenou, J. Sole, and D. R. Bull, “Efficient bitrate ladder construction for content-optimized adaptive video streaming,” 2021.

[17] A. Zabrovskiy, C. Feldmann, and C. Timmerer, “Multi-Codec DASH Dataset,” in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 438–443. [Online]. Available: https://doi.org/10.1145/3204949.3208140

[18] “Video Multi-Method Assessment Fusion,” 2020, [https://github.com/Netflix/vmaf, accessed 23-Aug-2021].

[19] J. Ozer, “Choosing the Best Per-Title Encoding Technology,” 2019, [https://go.bitmovin.com/choosing-per-title-encoding-technology, accessed 23-Aug-2021].

[20] Fraunhofer, “FAMIUM Deep Encode,” https://www.fokus.fraunhofer.de/en/fame/deep-encode, 2020, [Online; accessed 23-August-2021].

[21] Mux, “Instant Per-Title Encoding, Better quality through machine learning.” https://mux.com/per-title-encoding, 2020, [Online; accessed 23-August-2021].

[22] D. Silhavy *et al.*, “Machine learning for per-title encoding,” in *NAB Broadcast Engineering and Information Technology (BEIT)*, 2020.

[23] J. De Cock, Z. Li, M. Manohara, and A. Aaron, “Complexity-based consistent-quality encoding in the cloud,” in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 1484–1488.

[24] V. P. K. M, C. Timmerer, and H. Hellwagner, “MiPSO: Multi-Period Per-Scene Optimization For Http Adaptive Streaming,” in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.

[25] A. Zabrovskiy, P. Agrawal, R. Mathá, C. Timmerer, and R. Prodan, “ComplexCTTP: Complexity Class Based Transcoding Time Prediction for Video Sequences Using Artificial Neural Network,” in *2020 IEEE*

Sixth International Conference on Multimedia Big Data (BigMM), 2020, pp. 316–325.

[26] P. Agrawal, A. Zabrovskiy, A. Ilangovan, C. Timmerer, and R. Prodan, "FastTTPS: fast approach for video transcoding time prediction and scheduling for HTTP adaptive streaming videos," *Cluster Computing*, Nov 2020. [Online]. Available: https://doi.org/10.1007/s10586-020-03207-x

[27] N. Kim and B. Lee, "Analysis and Improvement of MPEG-DASH-based Internet Live Broadcasting Services in Real-world Environments," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 5, pp. 2544–2557, 2019. [Online]. Available: https://doi.org/10.3837/tiis.2019.05.017

[28] ITU-T, "Subjective video quality assessment methods for multimedia applications," International Telecommunication Union, Geneva, Recommendation ITU-T P.910, Apr. 2008.

[29] Y. A. Reznik, K. O. Lillevold, A. Jagannath, J. Greer, and J. Corley, "Optimal Design of Encoding Profiles for ABR Streaming," in *Proceedings of the 23rd Packet Video Workshop*, ser. PV '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 43–47. [Online]. Available: https://doi.org/10.1145/3210424.3210436

[30] Bitmovin, "Encoding Software and Per-Title Encoding," https://bitmovin.com/per-title-encoding, 2020, [Online; accessed 23-August-2021].

[31] C. Systems, "Cambria FTC Source-Adaptive Bitrate Ladder based on the encoding complexity of the source video," https://capellasystems.net/products/transcoding/cambria-ftc/features, 2020, [Online; accessed 23-August-2021].

[32] Beamr, "Content-adaptive bitrate (CABR) technology," https://beamr.com, 2020, [Online; accessed 23-August-2021].

[33] Brightcove, "Context Aware Encoding," https://www.brightcove.com/en/context-aware-encoding, 2020, [Online; accessed 23-August-2021].

[34] E. Labs, "QoE Smart Encoding and Stream Optimizer," https://www.epiclabs.io/product-lightflow-optimize-video-stream, 2020, [Online; accessed 23-August-2021].

[35] I. Katsavounidis, "Dynamic optimizer — a perceptual video encoding optimization framework," 2018, [Netflix Technology Blog, accessed 23-Aug-2021].

[36] ITU-T, "Objective perceptual video quality measurement using a jnd-based full reference technique," International Telecommunication Union, Geneva, Recommendation ITU-T J.144, Apr. 2004.

[37] G. BJONTEGARD, "Calculation of average psnr differences between rd-curves," *ITU-T VCEG-M33*, 2001. [Online]. Available: https://ci.nii.ac.jp/naid/10016799583/en/

[38] M. Bhat, J. M. Thiesse, and P. Le Callet, "A case study of machine learning classifiers for real-time adaptive resolution prediction in video coding," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.

[39] M. Covell, M. Arjovsky, Y. Lin, and A. C. Kokaram, "Optimizing transcoder quality targets using a neural network with an embedded bitrate model," in *Visual Information Processing and Communication VII, San Francisco, California, USA, February 14-18, 2016*, O. G. Guleryuz, A. Said, and R. L. Stevenson, Eds. Ingenta, 2016, pp. 1–7. [Online]. Available: http://ist.publisher.ingentaconnect.com/contentone/ist/ei/2016/00002016/00000002/art00016

[40] A. V. Katsenou, J. Sole, and D. R. Bull, "Content-gnostic bitrate ladder prediction for adaptive video streaming," in *2019 Picture Coding Symposium (PCS)*, 2019, pp. 1–5.

[41] "Intelligent scene cut detection and video splitting tool," https://pyscenedetect.readthedocs.io/en/latest/, 2020, [Online; accessed 23-August-2021].

[45] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 89–94. [Online]. Available: http://doi.acm.org/10.1145/2155555.2155570

[42] D. Rotman, D. Porat, and G. Ashour, "Robust and efficient video scene detection using optimal sequential grouping," in *2016 IEEE International Symposium on Multimedia (ISM)*, 2016, pp. 275–280.

[43] D. Vatolin, D. Kulikov, E. Sklyarov, S. Zvezdakov, and A. Antsiferova, "Video Transcoding Clouds Comparison 2019," Moscow State University, Tech. Rep., 11 2019.

[44] N. Bouzakaria, C. Concolato, and J. Le Feuvre, "Overhead and performance of low latency live streaming using MPEG-DASH," in *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, July 2014, pp. 92–97.

[46] S. Rossi and L. Toni, "Understanding user navigation in immersive experience: An information-theoretic analysis," in *Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems*, ser. MMVE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 19–24. [Online]. Available: https://doi.org/10.1145/3386293.3397115

[47] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad", "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, Nov 2018. [Online]. Available: https://doi.org/10.1016/j.heliyon.2018.e00938

[48] C. Nwankpa, W. Ijomah, and A. G. S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *ArXiv*, vol. abs/1811.03378, 2018.

[49] "A cubic-spline interpolation," https://docs.scipy.org/doc/scipy/reference/tutorial/interpolate.html, 2020, [Online; accessed 23-August-2021].

[50] L. Kröpfl, "Per-Title Bitrate Ladder Benchmark Tool," https://bitmovin.com/per-title-bitrate-ladder-benchmark-tool/, 2018, [Online; accessed 23-August-2021].

[51] C. R. Helmrich, M. Siekmann, S. Becker, S. Bosse, D. Marpe, and T. Wiegand, "XPSNR: A Low-Complexity Extension of The Perceptually Weighted Peak Signal-To-Noise Ratio For High-Resolution Video Quality Assessment," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 2727–2731.

[52] "FFprobe Documentation," https://ffmpeg.org/ffprobe.html, 2020, [Online; accessed 23-August-2021].

[53] "H.264 Video Encoding Guide," https://trac.ffmpeg.org/wiki/Encode/H.264, 2020, [Online; accessed 23-August-2021].

[54] J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Datasets for AVC (H.264) and HEVC (H.265) Evaluation of Dynamic Adaptive Streaming over HTTP (DASH)," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 51:1–51:6. [Online]. Available: http://doi.acm.org/10.1145/2910017.2910625

[55] Twitch, "Broadcasting Guidelines," https://stream.twitch.tv/encoding/, 2019, [Online; accessed 23-August-2021].

[56] YouTube, "Live encoder settings, bitrates, and resolutions," https://support.google.com/youtube/answer/2853702, 2019, [Online; accessed 23-August-2021].

[57] L. Sheugh and S. H. Alizadeh, "A note on pearson correlation coefficient as a metric of similarity in recommender system," in *AI Robotics (IRANOPEN)*, April 2015, pp. 1–6.

[58] "Keras: The Python Deep Learning library," https://keras.io/, 2019, [Online; accessed 23-August-2021].

[59] M. D. Zeiler, "Adadelta: An adaptive learningrate method," 2012, computing Research Repository (CoRR).

[60] "Convex hull calculation using SciPy," https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.ConvexHull.html, 2020, [Online; accessed 23-August-2021].

[61] "FFmpeg Filters Documentation," http://www.ffmpeg.org/ffmpeg-filters.html#select_002c-aselect, 2020, [Online; accessed 23-August-2021].