IEEE Open Journal of Signal Processing

Received 25 January 2021; revised 6 April 2021; accepted 4 May 2021. Date of publication 10 May 2021; date of current version 22 September 2021. The review of this article was arranged by Associate Editor Dr. M. Mrak.

*Digital Object Identifier 10.1109/OJSP.2021.3078657*

# Fast Multi-Resolution and Multi-Rate Encoding for HTTP Adaptive Streaming Using Machine Learning

**EKREM ÇETINKAYA [1] (Graduate Student Member, IEEE), HADI AMIRPOUR [1] (Student Member, IEEE), CHRISTIAN TIMMERER [1] (Senior Member, IEEE), AND MOHAMMAD GHANBARI [1,2] (Life Fellow, IEEE)**

[1] Christian Doppler Laboratory ATHENA, Alpen-Adria-Universität Klagenfurt, 9020 Klagenfurt, Austria
[2] School of Computer Science and Electronic Engineering, University of Essex, CO4 3SQ Colchester, U.K.

CORRESPONDING AUTHOR: EKREM ÇETINKAYA (e-mail: ekrem@itec.aau.at)

This work was supported by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and Christian Doppler Research Association, is gratefully acknowledged. Christian Doppler Laboratory ATHENA.

**ABSTRACT** HTTP Adaptive Streaming (HAS) is the de-facto solution for video delivery over the Internet. In HAS, each video is encoded at multiple quality levels and resolutions (i.e., representations) to enable adaptation of the streaming session to viewing and network conditions of the client. This requirement brings encoding challenges along with it, e.g., a video source should be encoded efficiently at multiple bitrates and resolutions. Fast multi-rate encoding approaches aim to address this challenge of encoding multiple representations from a single video by re-using information from already encoded representations. In this paper, a convolutional neural network is used to speed up both multi-rate and multi-resolution encoding for HAS. For multi-rate encoding, the lowest bitrate representation is chosen as the reference. For multi-resolution encoding, the highest bitrate from the lowest resolution representation is chosen as the reference. Pixel values from the target resolution and encoding information from the reference representation are used to predict Coding Tree Unit (CTU) split decisions in High-Efficiency Video Coding (HEVC) for dependent representations. Experimental results show that the proposed method for multi-rate encoding can reduce the overall encoding time by 15.08% and parallel encoding time by 41.26%, with a 0.89% bitrate increase compared to the HEVC reference software. Simultaneously, the proposed method for multi-resolution encoding can reduce the encoding time by 46.27% for the overall encoding and 27.71% for the parallel encoding on average with a 2.05% bitrate increase.

**INDEX TERMS** HTTP adaptive streaming, HEVC, multirate encoding, machine learning.

## I. INTRODUCTION

Video streaming is a vital part of today's Internet and accounts for the majority of today's global Internet traffic. Its share is expected to rise in the near future [1].

*HTTP Adaptive Streaming* (HAS) is the de-facto solution for delivering video content over the Internet and *Dynamic Adaptive Streaming over HTTP* (DASH) [2] is the standard solution introduced by MPEG for HAS. In HAS, videos are provided at different qualities and resolutions (*i.e.*, representations) in plain HTTP servers and the suitable representation is requested by the client based on the underlying network conditions. An example video storage schema for HAS is shown in Fig. 1.

The need to encode multiple representations of the same video content for HAS and the increasing complexity of video codecs are among the main problems for multi-rate encoding. To address these problems, fast multi-rate encoding methods were introduced. The main idea behind fast multi-rate encoding is to exploit the redundancy that is introduced while encoding the same video content at different quality levels. Therefore, the video is encoded at one bitrate (*i.e.*, *reference*

[Online]. Available: https://athena.itec.aau.at/

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

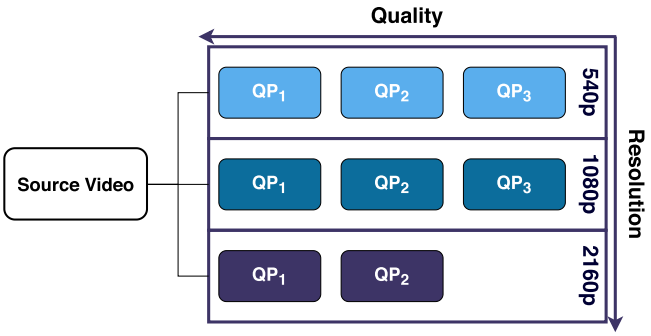


**FIGURE 1.** Example video storage schema for HAS. Multiple representations with different quality (*e.g.*, resolution, bitrate) are encoded from a single source video.

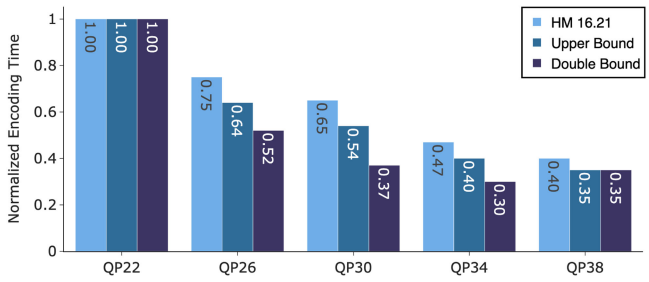


**FIGURE 2.** Normalized average encoding time for HEVC reference software (HM 16.21), Upper Bound, and Double Bound approaches.

*representation*) and the encoding information obtained here is re-used to speed up the encoding of the remaining representations (*i.e.*, *dependent representations*).

Increasing complexity of video contents in recent years (*i.e.*, higher resolutions, frame rate, *etc.*) [3] brought the need for more efficient video codecs. High Efficiency Video Coding (HEVC) [4] was standardized in 2013 to address the increasing complexity of video content and increased usage thereof is reported recently [3]. Although HEVC offers up to about 50% bitrate saving compared to its predecessor, *i.e.*, Advanced Video Coding (AVC) [5], this bitrate saving is achieved at the cost of a significantly increased time-complexity.

One of the most important and time-consuming tools introduced in HEVC is the Coding Tree Unit (CTU) structure used for block partitioning [6]. In HEVC, each frame is partitioned into $64 \times 64$ pixels sized square regions (CTUs) and then each CTU can be further divided recursively into smaller square regions (*i.e.*, Coding Unit (CU)) up to three times with the smallest block size being $8 \times 8$ pixels. Each split increases the depth value of the CU by 1 (*i.e.*, depth 0 for $64 \times 64$ pixels and depth 3 for $8 \times 8$ pixels). This block partitioning scheme allows HEVC to achieve a more precise motion compensation, but since typically the HEVC reference software uses a brute force approach to determine the optimal CTU partitioning, it also increases the time-complexity of the encodings [6]. In fast multi-rate encoding approaches, when a CTU is encoded, its information can be shared with other representations to avoid unnecessary search process.

There have been many attempts in the literature to propose a more efficient multi-rate or/and multi-resolution encoding scheme [7]–[13]. The most common approach is to use the highest quality as the *reference representation* [7], [9] while some approaches use the lowest quality as the *reference representation* [10], [12]. While several approaches achieve improvements for multi-rate encoding in terms of encoding time-complexity using the highest quality representation as the reference [7], [8], the overall time-complexity of parallel encoding scenarios are still not improved. In parallel encoding, the encoding of the video representations are conducted at the same time in different CPU or GPU cores and since encoding of *dependent representations* needs information from the *reference representation*, encoding time of the highest quality representation is a bottleneck in that scenario for the aforementioned approaches.

An example of the encoding time using the aforementioned approaches is given in Fig. 2. Here the *upper bound* means the highest depth value to be searched for the given CTU is the depth level of the co-located CTU in the highest quality representation (*i.e.*, QP22). At the same time, *double-bound* also uses the depth level of the co-located CTU in the lowest quality (*i.e.*, QP37) representation to limit the minimum depth level to be searched as well. Since both of these approaches require the encoding information from the highest quality representation to encode the dependent representations, they do not change the encoding time for the highest quality representation. Thus, the highest quality representation is still a bottleneck for parallel encoding scenario in this case. There have been some attempts in the literature to address this problem [11], [12], however, the multi-resolution scenario is not included in those studies.

This paper is an extension to our previous work [12] where we introduced a machine learning-based fast multi-rate encoding approach to improve the parallel encoding performance using the information from the lowest quality representation. This paper extends the method mentioned above to address multi-resolution encoding scenarios by introducing an enhanced Convolutional Neural Network (CNN) based encoding method. The proposed method can decrease the encoding time-complexity for both serial and parallel encoding scenarios.

The remainder of the paper is organized as follows. In Section II, a brief summary of the related work is given. Section III provides an overview about multi-rate and multi-resolution encoding. Section IV introduces the proposed method for multi-rate encoding and Section V describes the method for multi-resolution encoding. Experimental results for both approaches are given in Section VI. The paper is concluded with Section VII.

## II. RELATED WORK

Fast multi-rate encoding has become a popular topic due to the increasing usage of HAS for video delivery. The main goal in multi-rate encoding is to utilize information from

multiple versions of a single video to speed up encoding decisions for the remaining representations.

Representation with the highest resolution and quality is chosen as the reference representation in [7]. The maximum depth level of a CTU in the reference representation is chosen as the upper bound for CTU depth level searches (*i.e.*, depth levels that are higher than the reference CTU are skipped) while encoding the remaining representations. CTU structure, prediction mode, intra mode and motion vector information [4] are re-used to speed up encoding the remaining representations.

In [8], a double-bound approach is proposed for limiting CTU partitioning search options for multi-rate encoding. Information from both the highest and the lowest quality representations are used to speed up the CTU partitioning process for the remaining representations. Depth levels that are higher than the reference CTU in the highest quality and lower than the reference CTU in the lowest quality are skipped while searching for CU levels in the dependent representations. Multi-resolution scenarios are not mentioned in this study.

A random forest (RF) is used to predict the split decision for the given CU using the information from the reference representation (*i.e.*, the highest quality and resolution) in [9]. RF uses the transform coefficients, motion vectors, and CTU block structure as features to make the decision.

Although they improve the performance of individual encodings, the aforementioned studies do not improve the time complexity of the parallel encoding scenarios since they are using the highest quality representation as the reference, thus limiting the overall time complexity for parallel-encoding by the encoding time of the reference representation. There have been several studies that utilize encoding information from a lower quality representation which can be used in parallel encoding scenarios.

Encoding the lowest quality and resolution first and using it as the reference representation is proposed in [10]. Several refinement techniques are applied to information from the lower resolutions before using them to encode the higher resolution.

A multi-resolution framework for x265 that uses encoding information from the lower resolution representations is used to encode higher resolution representations in [14]. Different encoding information is shared depending on whether the resolutions are dyadic (*i.e.*, both width and height are multiple factor of 2 such as 540p and 1080p) or not. The encoding information reuse is limited by general information such as slice-type or scene-cut decisions for non-dyadic resolutions. In contrast, the encoding information in the block level is re-used between dyadic resolutions.

Amirpour *et al.* [11], analyzed the effects of choosing different quality levels as the reference representation. Based on the findings, the middle-quality representation is selected as the reference. An upper bound or a lower bound for CTU searches is applied for dependent representations based on their quality levels.

A CNN based approach for fast multi-rate encoding is proposed in [12]. A set of videos are encoded with HEVC and encoding information such as motion vectors, prediction unit (PU) modes, and RD costs are obtained to construct the training dataset. Then, separate CNNs for each quality level and depth level are trained using these aforementioned encoding information and YUV values from the raw videos. Finally, these CNNs are used for depth 0 and 1 decisions for the highest quality ($QP_{22}$) and depth 0 decisions for the second-highest quality ($QP_{26}$) to speed up the encoding with the specific focus on the parallel encoding performance.

Moreover, multiple studies utilized machine learning for video transrating/transcoding. A random forest based classifier is used to predict the CTU upper bound for HEVC transrating in [13]. Encoding information from both higher and lower bitrates are used to predict the CTU upper bound for the given quality level. The Long Short Term Memory (LSTM) model is used to predict CU split decisions for AVC to HEVC transcoding in [15]. Encoding information from the AVC bitrstream are fed to the LSTM and the CU split decisions are given for each depth level.

In this paper, a machine learning-based approach for both multi-rate and multi-resolution encoding is proposed which aims to reduce the encoding time for both serial and parallel encoding scenarios.

## III. MULTI-RATE AND MULTI-RESOLUTION ENCODING

The adaptive aspect of HAS enables a client to have the best possible content quality at a given time. An adaptive bitrate (ABR) algorithm at the client is responsible for constant monitoring of the network condition and deciding on what is the best possible quality level at a given time for requesting the next video segment. To make this adaptive streaming scheme work, there need to be multiple representations of the same video in the content server so that the ABR can have enough options for a variety of network and client viewing conditions. Therefore, the same video content needs to be encoded at different qualities and resolutions to achieve this variety of representations.

Encoding the same video into multiple representations results in significantly increased encoding time-complexity. Fast multi-rate and multi-resolution encoding approaches try to capitalize on the similarity between these representations to reduce the encoding time by choosing a reference representation and re-using its information to reduce encoding decisions for the remaining, *i.e.*, dependent representations.

One important aspect that should be considered in a multi-resolution case is that the spatial area that is covered by a single CTU in the lower resolution needs to be covered by multiple CTUs in the higher resolution due to the increased spatial resolution. An example of this difference is shown in Fig. 3. As can be seen in the figure, if it is required to use the encoding information from one resolution to speed up the encoding decision for another resolution, a mapping between CTUs is required to transfer the information efficiently.

It should be noted that width and height differences between two resolutions do not necessarily be as factors of 2. There are many scenarios where the resolution ratios are not

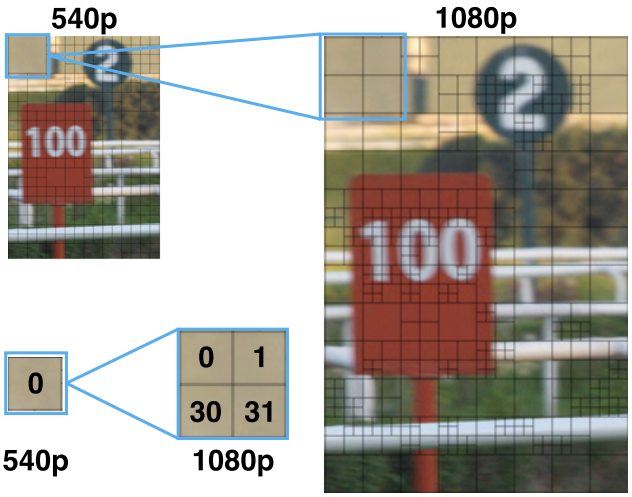


**FIGURE 3.** Number of CTUs needed to cover the same spatial area between different resolutions. The numbers in the boxes correspond to the CTU indexes. Here, the indexes are given in a row-major order, *i.e.*, it starts from top-left, increments by one while moving from left to right and moves to the left-most CTU in the next row when the CTUs in the row are finished.

integer values (*e.g.*, 1080p (1920 × 1080) and 720p (1280 × 720). It is important to determine a proper mapping for CTUs between different resolutions to use the encoding information efficiently for those cases. An example of this can be seen in [7] for PU mode decision where a *candidate list* system is used to utilize information from multiple high-resolution blocks to encode a low-resolution block.

## IV. MULTI-RATE ENCODING: FAME-ML

In this section, we introduce our previous work, a machine learning-based fast multi-rate approach for HAS with a specific focus on the parallel encoding performance, **Fa**st multi-rate **E**ncoding using **M**achine **L**earning (FaME-ML), in detail [12]. The encoding information used in FaME-ML is explained and then the training dataset is introduced. Afterwards, the convolutional neural network (CNN) used in FaME-ML is presented in detail, and the section is concluded with the overall methodology.

In FaME-ML, the lowest quality representation (*i.e.*, $QP_{38}$) is used as the reference representation and its encoding information is re-used to speed up the remaining representations in the same resolution. For each depth level and QP combination, a different CNN is trained.

### A. ENCODING INFORMATION

For FaME-ML we used the following encoding information as features that have been determined experimentally:

1) RD cost (required bits to encode the co-located CTU and its four sub-CUs for depth 0 and depth 1 values), $F_{RD}$, 5 elements.
2) Variance of pixel values (calculated for the co-located CTU and its four sub-CUs in the raw video), $F_V$, 5 elements.

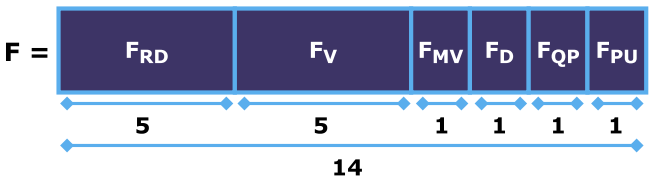


**FIGURE 4.** Feature vector structure in FaME-ML. Numbers indicate how many elements are stored for each feature.

3) Motion vectors (the average magnitude of motion vectors inside the reference CTU), $F_{MV}$, 1 element.
4) Depth split decision of the co-located CTU for the given depth level, $F_D$, 1 value.
5) Frame level QP, $F_{QP}$, 1 element.
6) PU decision of the co-located CTU, $F_{PU}$, 1 element.

.

Each feature is min-max normalized globally. The final feature vector for each CTU consists of 14 elements as shown in Fig. 4.

### B. DATASET

The network is trained over 12 sequences which have been introduced in the HEVC Common Test Conditions [16]. Each sequence is encoded with HM 16.21 [17] at five QP levels (22, 26, 30, 34, 38) and the aforementioned encoding information in the CTU level is saved to be used in the training phase. 90% of frames are chosen as training set and the remaining 10% are used as the validation set.

### C. FAME-ML CNN

A CNN is used as the depth split decision classifier in FaME-ML. Y, U, and V values from the raw video and the feature vector from the reference representation are fed into the CNN to obtain the split decision for a given CTU. The overall structure of the CNN is depicted in Fig. 5.

Y, U, and V values are passed through convolution block separately as the first step. Single max-pooling is applied to the Y channel so that the spatial size of the Y channel can match U and V channels. Resulting feature maps from this first part are concatenated and fed into the next part of the CNN which is the main texture processing part. The output from this part of the network has two dimensions ($[P_{non-split}, P_{split}]$) where each value represents the probability to split or not-split the given CTU/CU. Finally, the encoding feature vector is appended to this output and the final split decision is obtained after the fully-connected layers.

Batch normalization [18] and dropout [19] method are used to regulate the network. Rectified linear unit (ReLU) [20] is used as the activation function and Adam [21] is used as optimizer with *a learning rate* of $10^{-4}$.

### D. OVERALL METHOD

FaME-ML is designed with a specific focus on the parallel encoding time, thus, the CNN is used to predict the CTU decisions for only those representations that can be a bottleneck

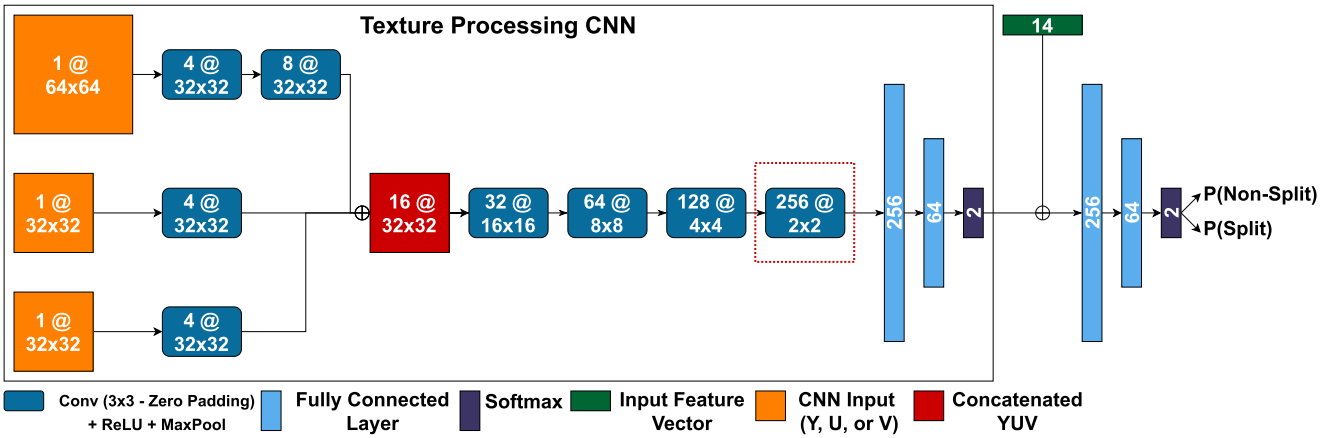


**FIGURE 5.** CNN architecture of depth 0 classifier used in FaME-ML. Values inside the boxes are given in the following format from left to right: *Channel count @ Width × Height of the channel* for convolution layers, *output size* for fully connected and softmax layers, and *input size* for the feature vector. In depth 1 classifier, section marked with red-dotted rectangle is removed and all layers but final two fully connected layers inside the *texture processing CNN* have their sizes halved.

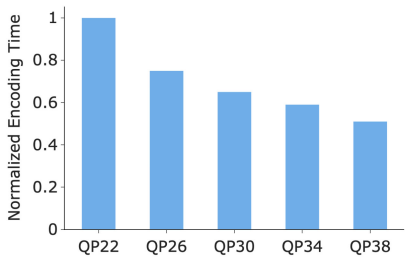


**FIGURE 6.** Normalized encoding time of test sequences with HM 16.21.

in the parallel encoding scenario. The normalized encoding time of test sequences with HM 16.21 can be seen in Fig. 6. It can be seen that if the encoding times of $QP_{22}$ and $QP_{26}$ can be reduced to a similar level to $QP_{30}$, there will not be an obvious bottleneck in that scenario. Thus, the CNN is only applied for $QP_{22}$ and $QP_{26}$ in FaME-ML. Moreover, since more time reduction is needed for $QP_{22}$ than $QP_{26}$, the CNN is used for both depth 0 and depth 1 decisions while it is only used for depth 0 decision for $QP_{26}$.

## V. MULTI-RESOLUTION ENCODING: FARES-ML

Despite achieving promising results for multi-rate encoding, FaME-ML does not address the multi-resolution encoding scenarios. Thus, we propose **Fa**st multi-**Res**olution encoding using **M**achine **L**earning (FaRes-ML) as an extension of FaME-ML to address multi-resolution scenarios. This section describes the details of FaRes-ML following the same structure as for FaME-ML.

Different networks are trained for different target resolutions and quality levels. There are three resolutions (540p, 1080p, and 2160p) and four quality levels for each resolution in the proposed setup. In total, there are 11 CNNs in the proposed framework. For FaRes-ML, the highest quality of the lowest resolution (*i.e.*, QP22 of 540p) is selected as the reference representation.

### A. ENCODING INFORMATION

Since the multi-resolution scenario is more challenging than just predicting the decisions in the same resolution, the feature set used in FaME-ML is extended for multi-resolution. Our aim was to use the maximum available coding decisions given by the HEVC reference encoder. After collecting that information, we experimentally conducted different pieces of training using a different set of features to decide on the best feature set to be used for this approach. The feature processing part of the FaRes-ML is also modified to exploit the encoding decision correlation among different representations more precisely.

Following encoding information is used in FaReS-ML:

1) RD costs in CU level (*CUBits, CUCosts*).
2) Motion vectors (*MVX, MVY*).
3) PU modes (*PU*).
4) CTU block partitioning structure (*Depth*).
5) Reference frame selection (*Reference*).
6) Frame level QP (*QP*).
7) **Intra luma and chroma directions**. These values correspond to the directions of luma and chroma components.
8) **Prediction modes**. These values correspond to the inter or intra prediction mode decisions within the reference CTU.

The last two encoding information were not used in FaME-ML, and the remaining features were used in a different format.In total, 11 features are used for FaRes-ML. Instead of processing the features and reducing their representations to a single dimension as in FaME-ML, each feature is directly saved in a matrix with size $8 \times 8$ where each value in the matrix corresponds to $8 \times 8$ pixel area, which is the minimum spatial size for a given CU. Note that these matrices consist of encoding decisions made by the HEVC reference software (HM 16.21) [17] while encoding the reference representation.

Like FaME-ML, min-max normalization is applied in video level for each feature (*i.e.*, for each feature, minimum and maximum value in the whole video is found, and all values are

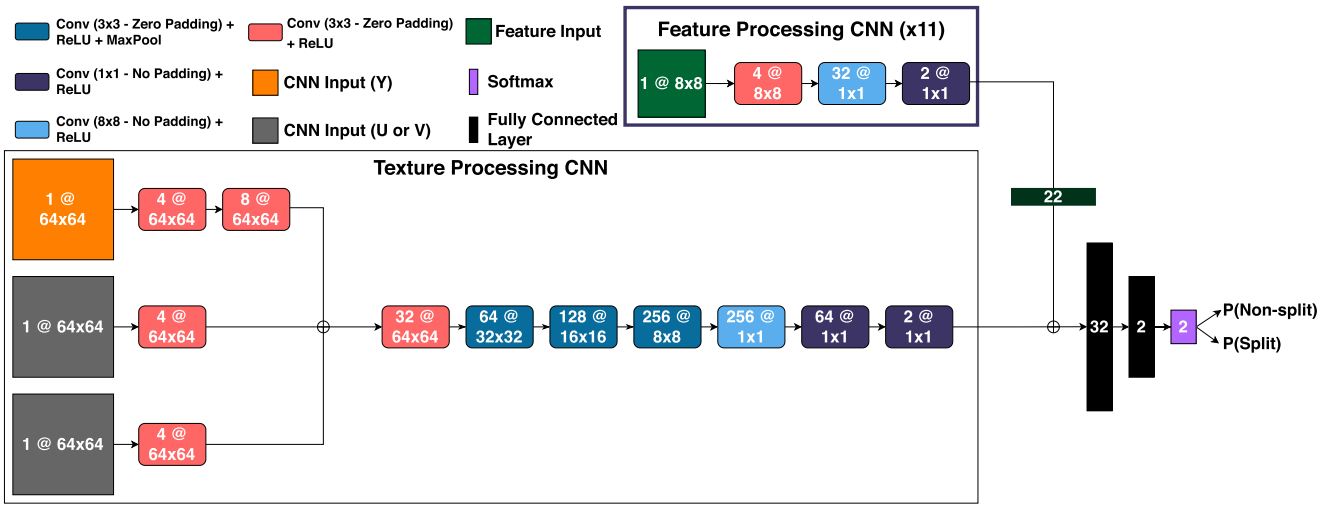


**FIGURE 7.** Structure of the CNN used in FaRes-ML.

normalized using these min-max values). Normalized features are saved and used in the *feature processing CNN* part of FaRes-ML.

Applying the min-max normalization resulted in an 0.5% increase in F1-scores on average in our experiments. It should also be noted that the possible value range for features is determined in the HEVC encoder except for the *CUBits* and *CUCosts* features. The effect of skipping the normalization for these two features was negligible in our experiments (*i.e.*, 0.1%).

### B. DATASET

We used 15 sequences from SJTU dataset [22] as our training data. All sequences were encoded with HEVC reference software (HM 16.21) [17] and encoding information at CTU level were extracted. All sequences are 30 fps and the first four seconds of each sequence (*i.e.*, 120 frames of each sequence) are used in the dataset. 90% of frames in each sequence are chosen as the training data and the remaining 10% are chosen as the validation data.

### C. FARES-ML CNN

The overall structure of the CNN used in FaRes-ML is depicted in Fig. 7. Y, U, and V values from the raw video with the target resolution and the encoding information from the reference representation are fed into the CNN to get the output for the split decision for the current CTU.

Y, U, and V values are processed in the first part of the network (*i.e.*, *texture processing CNN*). This part of the network consists of two convolutional blocks for the Y channel, one convolutional block for the U channel, and one convolutional block for the V channel. The reasoning behind using more convolutional blocks for the Y channel is that the Y channel contains more relevant information in most cases when 4 : 2 : 0 chroma subsampling is used. Each convolutional block in this part consists of a convolution operation followed by a batch normalization [18] followed by a ReLU [20] activation. Outputs of these convolutional blocks are concatenated channel-wise at the end. In this part of the network, each input has $64 \times 64$ sizes and the size of the outputs from these layers is not changed, thus the final output from the first part of the *texture processing CNN* has 16 channels with size $64 \times 64$.

Following the concatenation, the intermediate output is passed through four convolutional blocks with max-pooling applied at the end of each block, which reduces the size of the input to $8 \times 8$ at the end with 256 channels. Finally, it is passed through one convolution layer with kernel size $8 \times 8$ and two convolution layers with kernel size $1 \times 1$ to obtain the output with 2 dimensions that is not activated.

The second part of the network (i.e., *feature processing CNN*) is used for processing the encoding information. In this part, each encoding information is passed through a single convolution layer followed by a ReLU activation. Then, similar to the final part of the *texture processing* part, convolution with kernel size $8 \times 8$ is applied followed by another convolution with kernel size $1 \times 1$. These layers produce output with 2 dimensions, similar to the output of the *texture processing* part, for each encoding feature. This part of the network is used for each feature individually (*i.e.*, 11 times) and the outputs are concatenated together.

Finally, the output from the *texture processing* and the *encoding feature processing* parts are concatenated and passed through a fully-connected layer with a softmax function at the end to obtain the probability distribution for the non-split or the split class for the given CTU. The class with the higher probability is chosen to obtain the final split decision.

Adam [21] optimizer is used to train the network with a learning rate of $8 \times 10^{-4}$ and the learning rate is reduced by a factor of 10 (*i.e.*, learning rate is multiplied by 0.1) if the training loss is stuck for 10 epochs. ReLU [20] was used as the activation function. Batch normalization [19] and dropout [19] are applied in specific parts of the network to regulate.

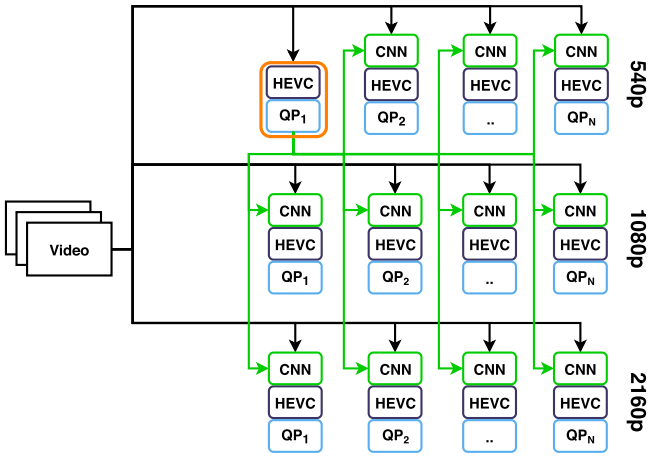


**FIGURE 8.** Flowchart of the FaRes-ML.

### D. OVERALL METHODOLOGY

The flowchart of the FaRes-ML is shown in Fig. 8 and works as follows:

1) Use HEVC to encode the reference representation (*i.e.*, 540p at QP1 in our example) and store the encoding information that will be further re-used while encoding the remaining (*i.e.*, dependent) representations.
2) Feed the available information to the CNN trained for the target resolution and QP combinations. That is, provide the Y, U, and V values from the raw video in the target resolution to the first part of the CNN (*i.e.*, texture processing CNN) and encoding information from the reference representation to the second part of the CNN (*i.e.*, feature processing CNN).
3) Use the decision of the CNN to speed up the encoding of the dependent representations.

## VI. EXPERIMENTAL RESULTS

Experimental results for both approaches are given in this section. First, the common evaluation setup for both methods is introduced. Second, results and ablation studies for FaME-ML are presented. Finally, the experimental results and ablation studies for FaRes-ML conclude this section.

### A. COMMON EVALUATION SETUP

Bjotengaard delta rates using PSNR and VMAF are used to evaluate the performance [23], [24]. Pytorch is used as the machine learning framework [25] and all experiments are conducted on the server with an Intel Xeon Gold 5218 @ 2.30 GHz, NVIDIA Quadro GV100, 384 GB memory, and Ubuntu Linux 18.04 as the OS. Please note that the predictions run on the GPU while the actual video encodings run on the CPU in parallel. GPU predictions were a multitude faster than the CPU encodings. Thus, the predictions did not cause an extra delay. For example, the time it takes to predict CTU decisions for the *Basketball* sequence was around 350 secs for 1080p resolution, while encoding the same sequence took around 2500 secs for $QP_{22}$ and 800 secs for $QP_{37}$.

**TABLE 1.** ROC-AUC Scores of the CNN Used in FaME-ML for Different QP Targets and Depth Levels

| Target QP | Depth 0 | Depth 1 |
|---|---|---|
| $QP_{22}$ | 0.79 | 0.77 |
| $QP_{26}$ | 0.81 | 0.75 |

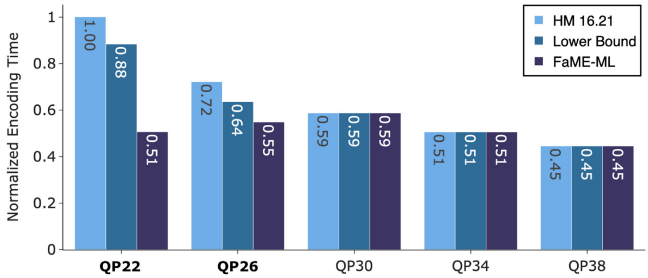


**FIGURE 9.** Normalized encoding time-complexities of different methods and FaME-ML.

### B. RESULTS: FAME-ML

To evaluate the performance of FaME-ML, 8 sequences from different datasets [26], [27] are used as the test set. The encoding information for the lowest quality representation (*i.e.*, $QP_{38}$) is saved and re-used to speed up the remaining representations (*i.e.*, $QP_{34}$, $QP_{30}$, $QP_{26}$, $QP_{22}$). ROC-AUC [28] scores of the trained networks on the test set are given in Table 1.

FaME-ML is compared with unmodified HEVC reference software (*HM 16.21*) [17] and the *Lower Bound* approach. The *Lower Bound* approach is a modified version of [8]. In the lower bound approach, the lowest quality representation is used as the reference encoding and the CU depth searches for the remaining representations are bounded by the depth level of the co-located CU in the reference representation, *i.e.*, depth levels that are lower than the co-located CU are not searched.

Normalized encoding time of different methods are given in Fig. 9. It can be seen that FaME-ML eliminates the obvious bottleneck problem by bringing down the encoding time of two highest quality representations to a similar level to the rest of the representations. The overall time saving for the parallel encoding scenario is 41.26% on average.

Encoding performance for individual sequences compared to the HM 16.21 are given in Table 2. BD-Rate (BDR) results using both PSNR ($BDR_P$) and VMAF ($BDR_V$) metrics for the test sequences are given along with the difference between the maximum time-complexity of each method compared to the maximum time-complexity of the reference software ($\Delta T_P$). The average BD-PSNR & BD-VMAF values are $-0.0152$ & $-0.0283$ and $-0.0243$ & $-0.0393$ for the lower bound approach and FaME-ML, respectively. Moreover, FaME-ML can reduce the overall encoding time by 15.08% on average.

**TABLE 2. Encoding Results for Test Sequences Using Lower Bound and FaME-ML. Metrics are Calculated Over All Five QP Levels. $\Delta T_P$ Represents the Encoding Time Difference Between the Highest Complexity Representation for Each Method**

| | Lower Bound | | | | | FaME-ML | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Sequence** | $\Delta T_P$ | **BDR**$_P$ | **BDR**$_P$ / $\Delta T_P$ | **BDR**$_V$ | **BDR**$_V$ / $\Delta T_P$ | $\Delta T$ | **BDR**$_P$ | **BDR**$_P$ / $\Delta T_P$ | **BDR**$_V$ | **BDR**$_V$ / $\Delta T_P$ |
| DucksTakeOff | 9.84 % | 0.346 % | 3.51 | 0.092 % | 0.93 | 36.42 % | 0.305 % | 0.84 | 0.119 % | 0.32 |
| InToTree | 3.11 % | 0.368 % | 11.83 | 0.688 % | 22.12 | 54.59 % | 1.325 % | 2.42 | 0.511 % | 0.93 |
| OldTownCross | 4.17 % | 0.457 % | 10.95 | 0.191 % | 4.58 | 52.89 % | 0.955 % | 1.80 | 0.077 % | 0.14 |
| ParkJoy | 21.23 % | 0.404 % | 1.90 | 0.083 % | 0.39 | 36.04 % | 0.920 % | 2.55 | 0.250 % | 0.69 |
| RedKayak | 12.72 % | 0.764 % | 6.01 | 0.282 % | 2.21 | 22.98 % | 0.525 % | 2.28 | 0.184 % | 0.81 |
| RushFieldCuts | 17.90 % | 0.471 % | 2.63 | 0.101 % | 0.56 | 40.60 % | 1.214 % | 2.99 | 0.456 % | 1.12 |
| ControlledBurn | 2.30 % | 0.703 % | 30.56 | 0.146 % | 6.34 | 46.91 % | 0.679 % | 1.47 | 0.493 % | 1.05 |
| ParkRunning3 | 16.81 % | 0.475 % | 2.82 | 0.086 % | 0.51 | 39.67 % | 1.178 % | 2.97 | 0.507 % | 1.27 |
| **Average** | **11.01 %** | **0.498 %** | **8.77** | **0.208 %** | **4.70** | **41.26 %** | **0.887 %** | **2.16** | **0.324 %** | **0.79** |

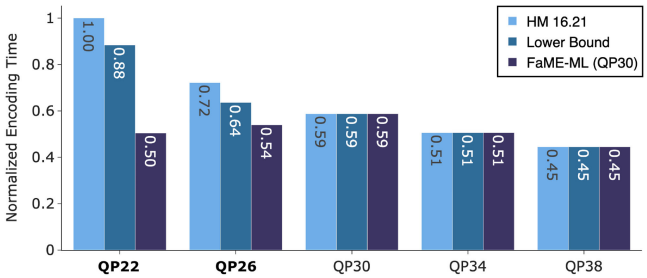


**FIGURE 10. Normalized encoding time-complexities of different methods and FaME-ML with $QP_{30}$ as the reference.**

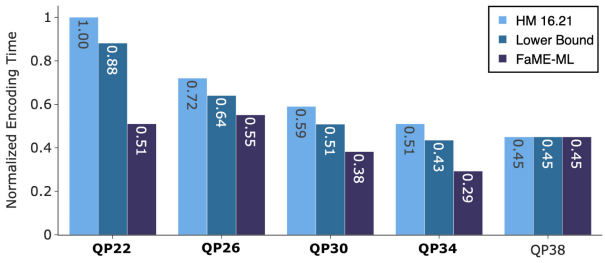


**FIGURE 11. Normalized encoding time-complexities of different methods and FaME-ML when the FaME-ML is applied to all dependent representations.**

### C. ABLATION STUDIES

#### 1) USING MIDDLE-QUALITY AS THE REFERENCE

It can be seen in the resulting encoding time graph that the middle-quality representation has the highest encoding time. Thus, one might ask why the middle-quality was not used as the reference instead of the lowest quality. To answer this question, the resulting encoding time graph with $QP_{30}$ as the reference representation is given in Fig. 10. It can be seen that the difference is almost negligible, and the same applies for BD-Rate as changing reference representation from $QP_{38}$ to $QP_{30}$ results with only 0.04% decrease in BD-Rate. Therefore, the lowest quality representation $QP_{38}$ is chosen as the reference representation as it is the ideal reference for speeding up the parallel encoding scenario since it has the lowest encoding time complexity.

#### 2) APPLYING CNN FOR ALL DEPENDENT REPRESENTATIONS

Another study was about the encoding performance when the CNN is applied to all dependent representations instead of the highest two quality ones. In particular, the lowest quality representation ($QP_{38}$) is used as the reference and Depth 0 decisions by CNNs along with the Depth 1 decision for $QP_{22}$ are applied to encode the representations. Moreover, the lower bound approach is applied to all dependent representations.

In this encoding scheme, the BD-Rate is increased to 2.1% compared to 0.88% in the proposed scheme. The resulting time graph is given in Fig. 11. It can be seen that, if the CNN is applied to speed up the encoding of $QP_{30}$ and $QP_{34}$, the parallel encoding time is now bounded by the $QP_{26}$ and the time saving for parallel encoding is increased to 45% compared to 41% in the proposed scheme.

### D. RESULTS: FARES-ML

We used 6 sequences from the MCML dataset [30] as test sequences. All videos were encoded at three resolutions (540p, 1080p, and 2160p) and at four different quality levels ($QP_{22}$, $QP_{27}$, $QP_{32}$, $QP_{37}$) for each resolution. Similar to the FaME-ML, the performance of FaRes-ML is compared with HEVC reference software (HM 16.21) [17] and the lower bound approach [8].

Average f1 scores of CTU decisions in the test set for different target quality and resolution levels using networks trained with $QP_{37}$ and $QP_{22}$ from the 540$p$ resolution as the reference representation are given in Table 3. It can be seen that using the highest quality representation of 540$p$ resolution gives more accurate results, thus it was used as the final reference representation in the proposed method. F1 scores using $QP_{37}$ as the reference is missing for 2160$p$ since it was observed that using the $QP_{22}$ as the reference performs better for the 1080$p$ resolution. Thus the training using $QP_{37}$ as the reference was not done for 2160$p$ due to high training time requirements.

It is essential to understand the reasoning behind relatively low f1 scores for depth predictions. We believe this is caused by the large differences between characteristics of different videos, which makes the CTU split decision problem a hard one for learning-based solutions since it is difficult to generalize properly. Fig. 12 illustrates the two input vectors at the same resolution and same QP level used in our studies. The first row is a feature vector that belongs to a CTU from *Bund Nightscape* video in the training set and the second row is a feature vector that belongs to a CTU from *Lake* video in the test set. It can be seen that despite minor differences in several features (*i.e.*, *Y, U, V CTUs*, *CUBits*, and *CUCosts*), most of

**TABLE 3.** F1-Scores of Different Networks Used in FaRes-ML

| Reference | 540p | | | | 1080p | | | | 2160p | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QP37 | QP32 | QP27 | QP22 | QP37 | QP32 | QP27 | QP22 | QP37 | QP32 | QP27 | QP22 |
| 540p - QP37 | - | 81.53 % | 88.03 % | 87.85 % | 59.54 % | 62.22 % | 73.28 % | 80.79 % | - | - | - | - |
| 540p - QP22 | 78.95 % | 85.49 % | 93.44 % | - | 67.66 % | 71.81 % | 79.53 % | 84.37 % | 64.81 % | 67.35 % | 69.14 % | 76.02 % |

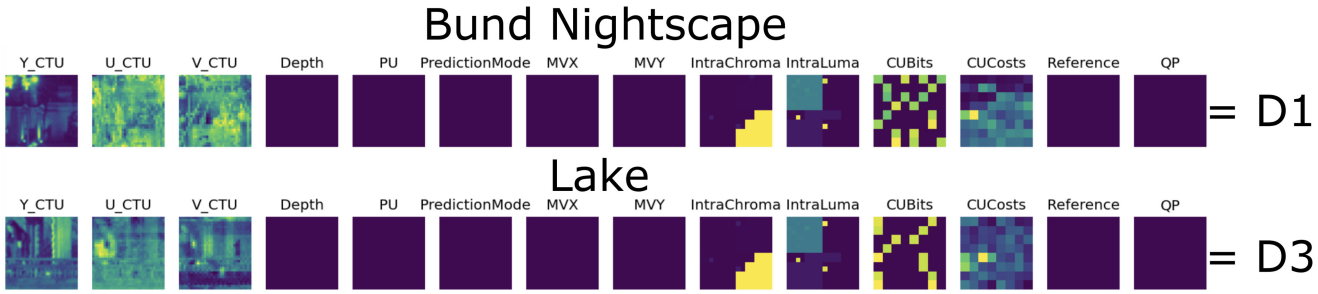


**FIGURE 12.** Illustration of two different feature vectors in the dataset. The first row is a feature vector from the *Bund Nightscape* sequence from the training set and the second one belongs to *Lake* sequence from the test set. Both vectors are obtained at the same resolution with the same QP. It can be seen that despite the very minor differences between features, depth decisions given by HEVC are totally different.

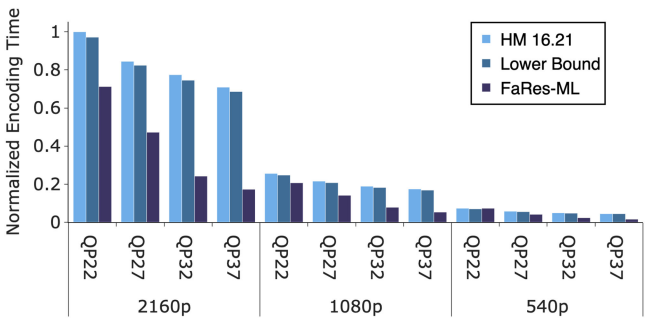


**FIGURE 13.** Average normalized encoding time for test sequences using HEVC reference software (HM 16.21), lower bound approach, and the FaRes-ML.

the features are very similar if not the same. However, the depth decision by HEVC is Depth 1 for the first feature vector and Depth 3 for the second feature vector. This example shows the CTU split decision problem is difficult to generalize.

Encoding results for test sequences are given in Table 4. Here we can see that the FaRes-ML can decrease the encoding time-complexity for 2160$p$ and 1080$p$ resolutions by 52.53% and 49.63% on average, respectively, while the time reduction is lower for 540$p$ sequences (*i.e.*, 36.65%) compared to higher resolution versions. This is expected since there is no encoding time saving for the reference representation ($QP_{22}$), which has the highest time-complexity in the 540$p$ resolution. In fact, the average time saving for the remaining representations (*i.e.*, $QP_{27}$, $QP_{32}$, $QP_{37}$) in the 540$p$ resolution is 49.12%.

To illustrate the parallel encoding performance better, the average normalized encoding times for test sequences in various quality levels and resolutions are shown in Fig. 13. It can be seen that using FaRes-ML can help with the bottleneck problem in parallel encoding scenarios by reducing the complexity in the higher quality and resolution representations. The encoding-time reduction is around (27.71%) in the highest quality ($QP_{22}$ and 2160$p$), which is much higher compared to the *lower bound approach* (2.84%) but not as high as the performance of FaME-ML in the multi-rate encoding. The reasoning behind this is that even if the FaRes-ML can predict depth decisions efficiently, the correlation between different resolutions of the same video is lower than the correlation between the same resolution but different quality level representations.

Example RD curves for the *Basketball* sequence in three resolutions are given in Fig. 14. Sample CTUs when the wrong depth prediction is given by FaRes-ML are shown in Fig. 15.

Finally, to see the encoding performance of FaME-ML in the multi-resolution scenario, *Basketball* sequence is encoded in all resolutions and QP levels using the $QP_{37}$ of 540$p$ representation as the reference (cf. Table 5). As expected, FaRes-ML outperforms FaME-ML in all cases thanks to its improved structure. The only case where FaME-ML performs better in terms of BD-Rate is the 540p version since it is a multi-rate encoding in this case; however, encoding time reduction is still better for FaRes-ML.

### *E. ABLATION STUDIES*

#### 1) EFFECTS OF FEATURES

To better understand the effect of each feature, another experiment is conducted. All feature vector values are set to zero except a single feature, and the prediction results are obtained. This procedure is applied for all features individually and resulting F1 scores are used to interpret the effect of each feature on the depth decision. The information factor is calculated by applying min-max normalization to average F1 scores. Visualization of the feature effects is given in Fig. 16.

It can be seen that the CTU partitioning decision in the reference encoding (*Depth*) has the highest information factor. This is expected since the depth decision given by the

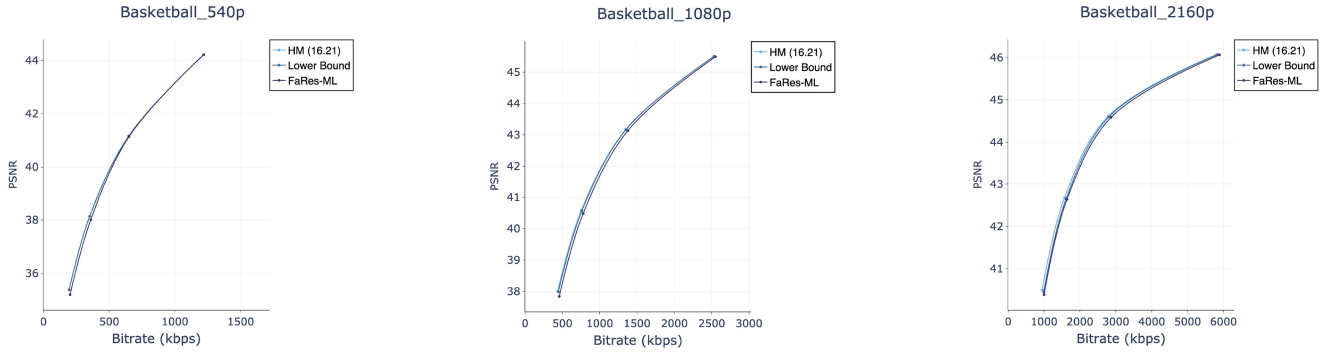


**FIGURE 14.** RD-Curves for the *Basketball* sequence in three resolutions.

**TABLE 4.** Comparison Between the Lower Bound Approach and the FaRes-ML With Respect to the Unmodified HM Reference Software (16.21)

| Sequence | | Lower Bound | | | | | FaRes-ML | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\Delta T$ | $BDR_P$ | $BDR_V$ | $BD_{PSNR}$ | $BD_{VMAF}$ | $\Delta T$ | $BDR_P$ | $BDR_V$ | $BD_{PSNR}$ | $BD_{VMAF}$ |
| 3840x2160 | Basketball | 4.42 % | 2.85 % | 3.35 % | -0.084 | -0.412 | 43.47 % | 2.18 % | 2.70 % | -0.068 | -0.312 |
| | Bunny | 3.64 % | 1.44 % | 1.10 % | -0.051 | -0.165 | 66.64 % | 3.91 % | 4.15 % | -0.139 | -0.520 |
| | Characters | 2.62 % | 6.92 % | 4.84 % | -0.193 | -0.095 | 53.76 % | 2.76 % | 1.44 % | -0.071 | -0.079 |
| | Contsruction | 1.67 % | 1.70 % | 2.08 % | -0.047 | -0.181 | 55.57 % | 4.08 % | 5.03 % | -0.095 | -0.357 |
| | Dolls | 3.24 % | 2.48 % | 2.60 % | -0.061 | -0.323 | 53.28 % | 3.45 % | 4.09 % | -0.081 | -0.443 |
| | Lake | 2.74 % | 0.14 % | 0.04 % | -0.006 | -0.011 | 42.46 % | 1.76 % | 1.57 % | -0.074 | -0.274 |
| **Average** | | **3.06 %** | **2.59 %** | **2.34 %** | **-0.074** | **-0.198** | **52.53 %** | **3.02 %** | **3.16 %** | **-0.088** | **-0.331** |
| 1920x1080 | Basketball | 5.12 % | 0.98 % | 0.90 % | -0.041 | -0.175 | 45.27 % | 3.47 % | 3.14 % | -0.150 | -0.538 |
| | Bunny | 3.15 % | 0.50 % | 0.49 % | -0.020 | -0.031 | 60.52 % | 2.57 % | 2.96 % | -0.100 | -0.324 |
| | Characters | 3.65 % | 2.18 % | 1.01 % | -0.076 | -0.018 | 49.73 % | 0.34 % | 0 % | -0.011 | -0.024 |
| | Construction | 2.12 % | 0.94 % | 1.03 % | -0.034 | -0.156 | 55.23 % | 2.09 % | 2.41 % | -0.070 | -0.192 |
| | Dolls | 4.42 % | 1.10 % | 1.57 % | -0.034 | -0.236 | 49.67 % | 3.93 % | 4.73 % | -0.120 | -0.580 |
| | Lake | 2.76 % | 0.12 % | 0.04 % | -0.004 | -0.004 | 37.35 % | 1.39 % | 1.49 % | -0.054 | -0.253 |
| **Average** | | **3.53 %** | **0.97 %** | **0.84 %** | **-0.035** | **-0.103** | **49.63 %** | **2.30 %** | **2.46 %** | **-0.084** | **-0.318** |
| 960x540 | Basketball | 4.55 % | 0.06 % | 0.21 % | -0.003 | -0.027 | 32.28 % | 2.11 % | 3.03 % | -0.102 | -0.433 |
| | Bunny | 2.53 % | 0.13 % | 0.26 % | -0.006 | -0.044 | 45.21 % | 1.18 % | 2.11 % | -0.052 | -0.229 |
| | Characters | 2.84 % | 0.31 % | 0.75 % | -0.007 | -0.013 | 39.44 % | 0.01 % | 0 % | -0.003 | -0.038 |
| | Construction | 1.64 % | 0.11 % | 0.60 % | -0.004 | -0.059 | 43.42 % | 0.21 % | 0.71 % | -0.007 | -0.056 |
| | Dolls | 3.65 % | 0.14 % | 0.51 % | -0.005 | -0.018 | 30.90 % | 0.57 % | 0.66 % | -0.020 | -0.070 |
| | Lake | 2.50 % | 0.02 % | 0.29 % | -0.001 | -0.036 | 28.66 % | 0.89 % | 2.35 % | -0.034 | -0.317 |
| **Average** | | **2.95 %** | **0.12 %** | **0.43 %** | **-0.004** | **-0.032** | **36.65 %** | **0.83 %** | **1.48 %** | **-0.036** | **-0.191** |
| **Total Average** | | **3.18 %** | **1.22 %** | **1.20 %** | **-0.037** | **-0.111** | **46.27 %** | **2.05 %** | **2.36 %** | **-0.069** | **-0.280** |

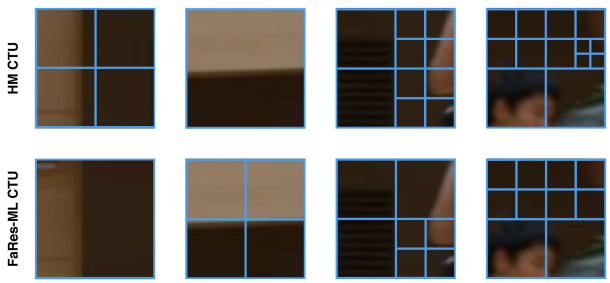


**FIGURE 15.** Example CTUs from the encoded *Basketball* sequence when the FaRes-ML predicts a wrong depth decision.

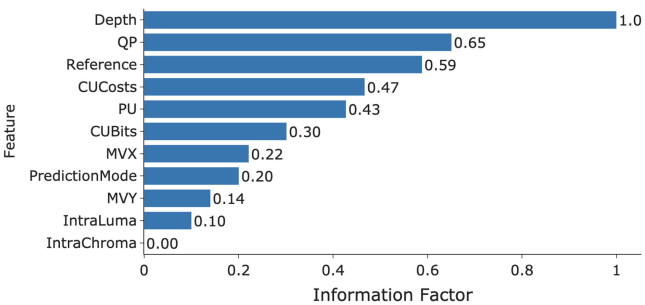


**FIGURE 16.** Information factor of features.

HEVC during encoding is a clear indicator of the encoding complexity of the given CTU and it should be well correlated with the CTU in the dependent representation.

### 2) MINIMUM DEPTH PREDICTION NETWORK

In this study, the network is modified to predict the minimum depth level inside a given CTU instead of a binary split decision for CTU split decision, thus both the intermediate and the final outputs have four dimensions now. Here, each dimension represents the probability of the CTU belonging to a depth level (*i.e.*, 0, 1, 2, and 3) given the input information. The modified CNN architecture can be seen in Fig. 17.

In this method, the CNN is trained with the minimum depth level inside the given CTU as the target during the training phase. The decision of CNN is used to skip lower-level depth searches based on the predicted depth value to improve the *lower bound* approach. This can be seen as an improvement

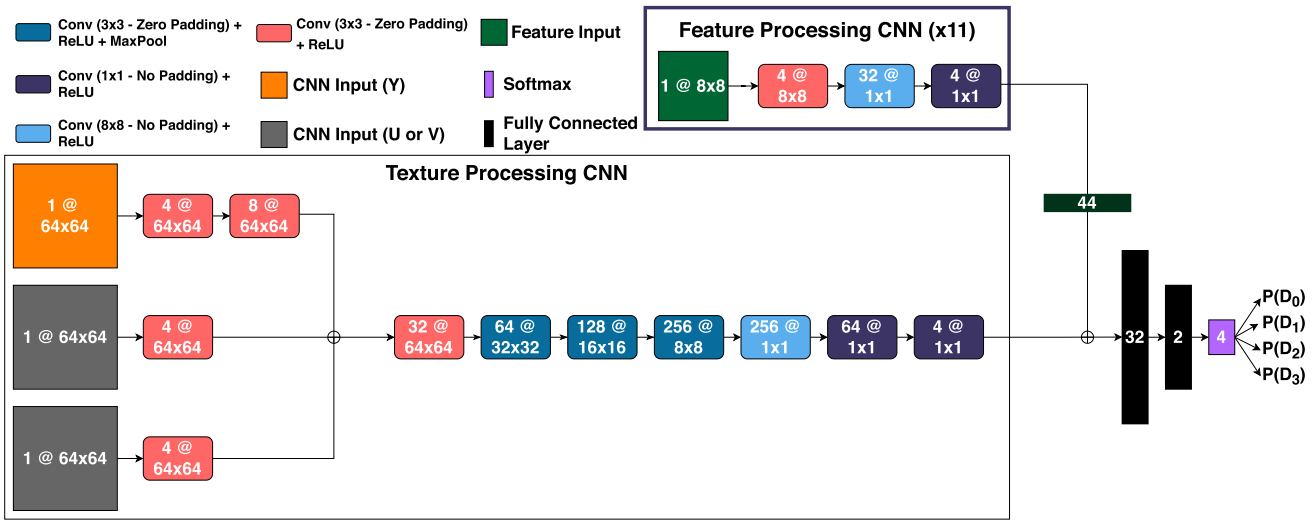


**FIGURE 17.** Modified FaRes-ML CNN architecture for minimum depth prediction. The intermediate output vectors now have 4 dimensions, one for each depth level.

**TABLE 5.** Encoding Performances of the FaME-ML and FaRes-ML for the *Basketball* Sequence

| Method | | $\Delta T$ | $BDR_P$ | $BD_{PSNR}$ |
|---|---|---|---|---|
| 2160p | FaME-ML | 18.12 % | 4.30 % | -0.128 |
| | FaRes-ML | 43.42 % | 2.18 % | -0.068 |
| 1080p | FaME-ML | 21.40 % | 7.81 % | -0.299 |
| | FaRes-ML | 45.27 % | 3.47 % | -0.150 |
| 540p | FaME-ML | 9.44 % | 1.98 % | -0.109 |
| | FaRes-ML | 32.28 % | 2.11 % | -0.102 |

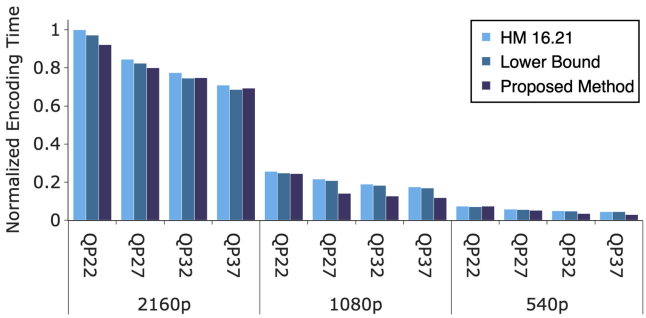


**FIGURE 18.** Average normalized encoding time for test sequences using HEVC reference software (HM 16.21), lower bound approach, and the proposed minimum depth prediction approach.

for the lower bound approach using machine learning. The encoding time graph is given in Fig. 18. While the proposed approach can improve the lower bound approach, the improvement is barely noticeable; thus, it is not used in the final approach. With the minimum depth prediction network, the classification problem becomes a multi-class classification instead of the binary classification, which poses an extra challenge and reduces the prediction accuracy. For example, the accuracy of the minimum depth prediction network for $1080p/QP22$ is 53.08% while it is 84.37% for the network used in the FaRes-ML.

## VII. CONCLUSION

In this paper, a machine learning-based approach is proposed for fast multi-resolution and multi-rate encoding for HTTP adaptive streaming. First, a fast multi-rate encoding approach (FaME-ML) is proposed, and then it is extended to address multi-resolution scenarios (FaRes-ML). The lowest quality representation is used as the reference for FaME-ML, while the highest quality representation from the lowest resolution is chosen as the reference for the extended FaRes-ML approach. Encoding information (*e.g.*, RD cost, intra directions, motion vectors, prediction modes, *etc.*) from the reference representation in addition to luma (*i.e.*, Y), chroma (*i.e.*, U, and V) values from the target resolution for a given CTU are passed to the CNN to predict the split decision. A CTU split decision dataset is constructed to train separate CNNs for each target QP and resolution combination. The trained CNN is then applied during the encoding of dependent representations. Experimental results show that the FaME-ML can achieve 15.08% time saving for serial encoding and 41.26% for parallel encoding while causing 0.89% bitrate increase. On the other hand, FaRes-ML can achieve 46.27% encoding time saving for serial encoding and 27.71% time saving for parallel encoding while causing a 2.05% bitrate increase.

As future work, the proposed approach can be applied for other encoding decisions (*e.g.*, PU decision, reference frame selection, etc.) to improve the encoding performance better. Moreover, the extension of the proposed approach for VVC can also be an interesting study due to the increased encoding time complexity of VVC compared to HEVC.

## REFERENCES

[1] Cisco, "Cisco annual internet report (2018–2023) white paper," 2017–2022, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, Apr. 2011.

[3] Bitmovin Inc., "Bitmovin video developer report 2020," Accessed: Nov. 03, 2020. [Online]. Available: https://go.bitmovin.com/video-developer-report-2020

[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[5] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[6] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec. 2012.

[7] D. Schroeder, A. Ilangovan, M. Reisslein, and E. Steinbach, "Efficient multi-rate video encoding for HEVC-based adaptive HTTP streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 1, pp. 143–157, Jan. 2018.

[8] H. Amirpour, E. Çetinkaya, C. Timmerer, and M. Ghanbari, "Fast multi-rate encoding for adaptive HTTP streaming," in *Proc. Data Compression Conf.*, 2020, pp. 358–358.

[9] J. De Praeter *et al.*, "Fast simultaneous video encoder for adaptive streaming," in *Proc. IEEE 17th Int. Workshop Multimedia Signal Process.*, 2015, pp. 1–6.

[10] K. Goswami *et al.*, "Adaptive multi-resolution encoding for ABR streaming," in *Proc. 25th IEEE Int. Conf. Image Process.*, 2018, pp. 1008–1012.

[11] H. Amirpour, E. Çetinkaya, C. Timmerer, and M. Ghanbari, "Towards optimal multirate encoding for HTTP adaptive streaming," in *Proc. Int. Conf. Multimedia Model.*, 2021, pp. 469–480.

[12] E. Çetinkaya, H. Amirpour, C. Timmerer, and M. Ghanbari, "FaME-ML: Fast multirate encoding for HTTP adaptive streaming using machine learning," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process.*, 2020, pp. 87–90.

[13] M. Grellert, L. A. da Silva Cruz, B. Zatt, and S. Bampi, "Coding mode decision algorithm for fast HEVC transrating using heuristics and machine learning," *J. Real-Time Image Process.*, pp. 1–16, 2021.

[14] A. Mathesawaran *et al.*, "Open source framework for reduced-complexity multi-rate HEVC encoding," in *Proc. Appl. Digit. Image Process. XLIII. Int. Soc. Opt. Photon.*, vol. 11510, 2020, Art. no. 115101Y.

[15] Y. Wei, Z. Wang, M. Xu, and S. Qiao, "An LSTM method for predicting CU splitting in H.264 to HEVC transcoding," in *Proc. IEEE Vis. Commun. Image Process.*, 2017, pp. 1–4.

[16] F. Bossen *et al.*, "Common test conditions and software reference configurations," in *JCTVC-L1100 12th JCT-VC Meeting*, Jan. 2012, p. 7.

[17] JCT-VC, "HEVC Reference Software HM 16. 21," Accessed: Oct. 10, 2020. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jct-vc/HM

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 448–456.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[20] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations (ICLR)*, 2015.

[22] L. Song, X. Tang, W. Zhang, X. Yang, and P. Xia, "The SJTU 4 K video sequence dataset," in *Proc. 5th IEEE Int. Workshop Qual. Multimedia Exp.*, 2013, pp. 34–35.

[23] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," in *VCEG-M33 13th VCEG Meeting*, Austin, TX, USA, Apr. 2001.

[24] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," Accessed: May 29, 2020. [Online]. Available: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652, 2016

[25] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

[26] L. Haglund, "The SVT high definition multi format test set," Swedish Television Stockholm, version 1.0, Sveriges Telev. AB (SVT), Tech. Rep., 2006.

[27] K. Suehring and X. Li, "JVET common test conditions and software reference configurations," in *JVET-B1010 2nd JVET Meeting*, San Diego, CA, USA, Feb. 2016.

[28] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[29] M. Cheon and J.-S. Lee, "Subjective and objective quality assessment of compressed 4 K UHD videos for immersive experience," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 7, pp. 1467–1480, Jul. 2018.




**EKREM ÇETINKAYA** (Graduate Student Member, IEEE) was born in Istanbul, Turkey in 1995. He received the B.Sc. and M.Sc. degrees in computer science from Ozyegin University, Istanbul, Turkey, in 2018 and 2019, respectively. He is currently working toward the Ph.D. degree with Alpen-Adria-Universität, Klagenfurt, Austria. He is working on the content provisioning part of the ATHENA project and he contributed to several publications. His current research focuses on video encoding enhancements with machine learning.



**HADI AMIRPOUR** (Student Member, IEEE) received the two B.Sc. degrees in electrical and biomedical engineering and the M.Sc. degree in electrical engineering from the K. N. Toosi University of Technology, Tehran, Iran. He was involved in the project EmergIMG, a Portuguese consortium on emerging imaging technologies, funded by the Portuguese funding agency and H2020. He is currently with ATHENA. His research interests include video streaming, image and video compression, quality of experience, emerging 3D imaging technology, and medical image analysis.



**CHRISTIAN TIMMERER** (Senior Member, IEEE) is currently an Associate Professor with the Institute of Information Technology and is the Director of the Christian Doppler Laboratory ATHENA, Klagenfurt, Austria. He has coauthored seven patents and more than 200 articles in his research field, which include immersive multimedia communication, streaming, adaptation, and quality of experience. He was the General Chair of the WIAMIS 2008, QoMEX 2013, MMSys 2016, and PV 2018 and has participated in several EC-funded projects, notably DANAE, ENTHRONE, P2P-Next, ALICANTE, SocialSensor, COST IC1003 QUALINET, and ICoSOLE. He also participated in ISO/MPEG work for several years, notably in the area of MPEG-21, MPEG-M, MPEG-V, and MPEG-DASH, where he also the Standard Editor. In 2013, he Co-Founded Bitmovin to provide professional services around MPEG-DASH, where he holds the position of the Chief Innovation Officer — Head of Research and Standardization.



**MOHAMMAD GHANBARI** (Life Fellow, IEEE) is an Emeritus Professor with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom. He is currently involved in the Athena Project with the Universitat Klagenfurt, Klagenfurt, Austria. He is internationally best known for the pioneering work on layered video coding, which earned him IEEE Fellowship in 2001 and he was also promoted IEEE Life Fellow in 2014. He has registered for thirteen international patents and authored or coauthored more than 700 technical papers on various aspects of video networking, many of which have had fundamental influences in this field, which include video or image compression, layered or scalable video coding, video transcoding, motion estimation, and video quality metrics. He is the author and coauthor of eight books, and his book *Video Coding: An Introduction to Standard Codecs*, published by IET press in 1999, was the recipient of the Rayleigh Prize as the Best Book of Year 2000 by IET.