Performance Analysis of Machine Learning for Arbitrary Downsizing of Pre-Encoded HEVC Video

Luong Pham Van, Johan De Praeter, Glenn Van Wallendael, Jan De Cock, and Rik Van de Walle

**Abstract — ***Nowadays, broadcasters deliver ultra-high resolution video to their consumers. This live video is sent to a set-top box for display on a television. However, if one or more users in the home want to view the same video on their personal mobile devices with a lower display resolution and limited processing power, decoding the original ultra-high resolution video would result in stuttering and quickly drain the battery life on these devices. To enable a satisfactory consumer experience, the resolution of the video stream should be adapted to the target mobile device at the set-top box. The aim of this paper is to investigate the performance of different machine learning strategies to arbitrary downsize video pre-encoded with the high efficiency video coding standard (HEVC). These machine learning techniques exploit correlation between input and output coding information to predict the splitting behavior of HEVC coding units. Several machine learning algorithms are optimized. Additionally, both online and offline training strategies are tested. Of the tested algorithms, online-trained random forests achieve the best compression-efficiency with a bit rate increase of 5.4% and an average complexity reduction of 70%*[1].

Index Terms — Video adaptation, arbitrary downsizing, high efficiency video coding, machine learning.

I. Introduction

Ultra-high definition displays have become commonplace in the current consumer electronics market, leading to an increased demand for ultra-high definition content. When a consumer watches a live broadcast of such content over satellite, the video is received by a set-top box and decoded for display on a TV. However, if one or more users want to follow the live broadcast on their mobile devices when leaving the room, the video would have to be transmitted to the mobile device at full resolution. Due to the high decoding complexity of ultra-high resolution video, the mobile device would have insufficient processing power to deliver a fluid viewing experience, and the limited battery life would drain quickly.

To enable this kind of second-screen viewing for the consumer without video stuttering and while maximizing the battery life, the decoding complexity of the video should be minimized. Therefore, the video arriving at the mobile device should have the same resolution as the target display. To achieve this, the video resolution can be adapted to the target mobile device at the set-top box by using transcoding techniques [1].

Transcoding is an adaptation technique which modifies the properties of a video stream. Depending on the type of modified properties, transcoding can be classified into three main categories including bit rate reduction [2], [3], spatial resolution reduction [4], and frame rate reduction [5]. Bit rate reducing techniques are recommended for small reductions in bit rate. When more drastic reductions are needed, spatial downsizing is advised.

This paper extends previous work [6] to optimize downsizing of videos pre-encoded with HEVC. As the main contribution to the state-of-the-art, the performance of different machine learning strategies is investigated. To determine the optimal strategy, optimized and non-optimized versions of different algorithms are compared. Additionally, the benefits of content-adaptive feature selection are examined and both online and offline training strategies are tested.

The rest of this paper is organized as follows. First, an overview of the related works in transcoding is given in Section II. The proposed arbitrary downsizing architecture is described in Section III. Then, the machine learning model for predicting coding unit splits is proposed in Section IV. In this section, the optimization of machine learning is elaborated on. Thereafter, a transcoding complexity control mechanism is presented in Section V. The performance of the proposed techniques is then evaluated in Section VI. Finally, the conclusion is drawn in Section VII.

II. Related Works

A typical downsizing transcoder consists of a decoder-encoder cascade. Such a transcoder decodes the input video, resizes it, and then re-encodes the result. However, the re-encoding step has a high computational complexity. Several techniques have been proposed to reduce the complexity of

[1] The research activities that have been described in this paper were funded by Ghent University, iMinds, Ph.D. and post-doctoral fellow grants of the Agency for Innovation by Science and Technology (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union. Furthermore, this work was carried out using the Stevin Supercomputer Infrastructure at Ghent University. The work is performed as part of the iMinds V-FORCE (Video 4K Composition and Efficient streaming) project (under IWT grant agreement no. 130655).

Luong Pham Van, Johan De Praeter, Glenn Van Wallendael, Jan De Cock, and Rik Van de Walle are with the Department of Electronics and Information Systems – Multimedia Lab, Ghent University – iMinds, Ghent, Belgium (e-mail: luong.phamvan@ugent.be, johan.depraeter@ugent.be, glenn.vanwallendael@ugent.be, jan.decock@ugent.be, rik.vandewalle@ugent.be).

Contributed Paper
Manuscript received 09/01/15
Current version published 12/29/15
Electronic version published 12/29/15.

the re-encoding step by predicting coding information. Doing video downsizing for the advanced video coding standard (H.264/AVC) [7], the motion of the output video is derived from the input motion vectors using a weighted median filter. By predicting the motion vectors, the motion estimation process in the encoder can be accelerated. However, this method leads to a high subjective quality loss.

Alternative fast transcoding techniques predict encoding information by using machine learning to exploit the correlation between the coding information of the input and output video [8]-[13]. Different machine learning algorithms have been used in these transcoding techniques. Support vector machines (SVM [14]) have been applied for transcoding H.264/AVC bitstreams [8]. Decision trees (DT [15]) are also widely used for transcoding [9]-[12]. More recently, the linear discriminant function has been used for transcoding a video from H.264/AVC to the high efficiency video coding (HEVC) standard [13], [16].

In the existing literature, most machine learning techniques for transcoding are based on offline training [8]-[12]. This means that the prediction model is trained on a set of videos (the training set) and evaluated on a separate test set. In this case, the prediction model only has to be trained once. However, such a model is not content-adaptive. This problem is solved by using an online training approach [13]. The prediction model is made content-adaptive by training on the first *N* frames of the sequence to predict the decisions for the following frames. A disadvantage of this approach is that retraining might be needed after changes in the video content.

In the above machine learning methods for transcoding, a fixed feature set is used. This feature set is often determined through observations by selecting features from a larger set, without an automatic mechanism. The reduced feature set is then used for training. However, the features providing the most information might be different depending on the sequence. Therefore, there is a need to adaptively determine the optimal features to use for given training data.

Since the accuracy of the machine learning model might be low, some existing techniques modify the generated model. For example, several coding modes (skip and 2Nx2N modes) are always tested even when they are not recommended by the machine learning model [13]. On the other hand, only the highest levels in the decision tree are used in order to avoid overtraining on the training set [12]. Another method to handle the low accuracy of the machine learning model is to take the confidence of a prediction into account [10]. In that case, only predictions with a confidence above a certain threshold will be used to skip decisions in the encoder.

The existing transcoding techniques demonstrate that the re-encoding complexity can be significantly reduced by using machine learning. However, the following problems should be considered to achieve the best performance of transcoding. First, only a single machine learning algorithm is applied in existing techniques. However, it is unknown what the effects would be if a different algorithm were applied. Second, although attempts have been made to make a model adaptive by using online training, an adaptive feature selection has not been taken into account. Finally, the optimization of machine learning parameters has not been focused on.

III. Arbitrary Downsizing Architecture

Many downsizing transcoding techniques have been proposed for previous video coding standards (e.g. H.264/AVC). However, these transcoding techniques cannot be directly applied to the HEVC standard due to the differences in block structure, motion estimation and residual coding. Therefore, to apply HEVC for a diverse set of applications, a novel efficient downsizing technique is needed for this new standard.

This paper proposes several techniques to accelerate the downsizing process of HEVC video using machine learning. An overview of the splitting process of coding units (CUs) in HEVC is presented first. Then, the proposed system for fast arbitrary downsizing of HEVC video is given.

A. Overview of the HEVC CU Splitting Process

The goal of the HEVC CU splitting process is to obtain the optimal block structure for coding a video frame. The largest block in the structure, i.e. the coding tree unit (CTU), which is typically 64x64 pixels, is recursively split into CUs from depth 0 (64x64 pixel CU) to depth 3 (8x8 pixel CU) [16]. For each CU, three prediction modes (i.e. skip, inter, and intra modes) are supported. Each CU is the root for further evaluation of prediction units (PU), and transform units (TUs). Depending on the mode, eight different PU sizes can be chosen. To obtain the most efficient mode for a CU at depth *d*, all PU partitions and all Residual Quad-Tree (RQT) configurations are evaluated during the rate distortion optimization (RDO) process. This RDO process is recursively performed on the four sub-CUs at depth *d + 1*. After evaluating the rate distortion (RD) cost of a CU at depth d and the combined RD-costs of its sub-CUs, the splitting of the CU is decided and signaled by a split-flag.

The HEVC encoder is very complex due to the flexibility of CU splitting. This complexity can be reduced by limiting the number of RD-evaluations. Several techniques for fast CU size decisions have been proposed [17]-[19]. The complexity of CU partitioning is the motivation for designing transcoding models that predict the splitting behavior of CUs.

B. Arbitrary Downsizing Architecture

The proposed transcoding system is depicted in Fig. 1. First, the input high resolution HEVC video is decoded followed by the extraction of coding information and raw video features of the reconstructed video. Based on the network bandwidth constraints and/or the screen resolution of playback devices, the downsizing scaling factor can be chosen. Using a non-normative downsampling filter [20], the decoded video is then resized by dividing its width and height by this scaling factor. Meanwhile, the splitting behaviour of the CUs in the output video stream is predicted using machine learning models. Finally, the resized video is re-encoded using this predicted information. The details of building and optimizing the prediction model are elaborated on in Section IV.

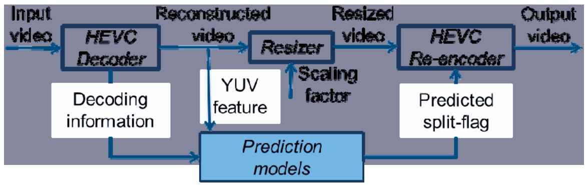

Fig. 1. Proposed arbitrary downsizing architecture for HEVC video.

It should be noted that the arbitrary scaling factor may lead to misalignment between the co-located area of the output CU and the co-located input CUs (Fig. 2). In this case, the existing mode mapping algorithms [21] cannot be used. However, this problem can be solved by the proposed machine learning method. The correlation between the block size of the output CU and the coding information of partially co-located and fully co-located CUs is exploited by machine learning.

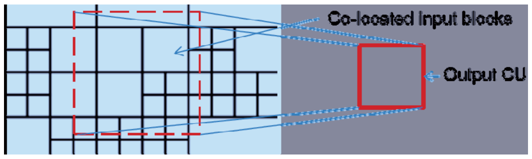

Fig. 2. The misalignment of co-located input CUs.

IV. Fast CU Splitting Prediction Model using Machine Learning

The CU splitting prediction models are constructed using machine learning algorithms to predict the CU structure of the output video. There are several challenges that should be considered when building the prediction models in order to improve the transcoding performance. Firstly, the accuracy of the prediction should be improved as much as possible. If the prediction is more accurate, the quality loss will be smaller. Furthermore, the complexity of the prediction step needs to be small. Finally, the model should be adaptive. In other words, it should be possible to construct a model and optimize it regardless of transcoding parameters such as the scaling factor of the video resolution.

This paper proposes several techniques to handle the above challenges of transcoding based on machine learning. Since either an online or offline training mechanism can be applied, these mechanisms are described in the following subsection. To achieve an optimal accuracy level of prediction, several machine learning algorithms are then investigated. The optimal parameters of these algorithms are adaptively chosen depending on the training data as described thereafter. As the last part of this section, a description is given about how the best features are selected during training by using an adaptive feature selection mechanism. This mechanism aims to lower the overall complexity of the machine learning system.

A. Training Mechanism

Three prediction models, which respectively predict the splitting behavior of CUs at depth 0, 1, and 2, are constructed with machine learning algorithms. For these algorithms, either an online or offline training strategy can be applied.

For online training, a set of N frames is first transcoded without acceleration. Then, the coding information from the input bit stream, the features of the decoded video and the CU depths in the resized video are extracted from these N frames. Based on this training data, the machine learning parameters and the important features are selected. The three prediction models are then built using these optimal settings. Using these models, the CU structure of the following frames can be predicted to reduce the complexity of the re-encoding step.

The advantage of this online training mechanism is that the parameters of the machine learning models are adapted to the content of the input sequence. Additionally, this method does not depend on the coding configuration, e.g. it is independent of the scaling factor used during transcoding. However, if the properties of the video content change significantly, the prediction accuracy of the model might be reduced and retraining might be necessary.

An alternative training strategy is offline training. With this technique, the first K frames of selected sequences are used as a training set to build the three prediction models. The other sequences are then encoded using these models. One of the challenges of offline training is to find a set of training sequences that sufficiently represents the complete test set. To achieve this, the training sequences are selected by using a cross-validation technique. The prediction models are first constructed using only a single training sequence with input resolutions varying from 832x480 up to 1920x1080 pixels [22]. The number of training frames is based on the resolution of the sequence. The 1920x1080, 1280x720, and 832x480 sequences respectively use 50, 75, 100 frames for training. After a prediction model is built using a single sequence, all other sequences are transcoded using this model. The three sequences that result in the highest compression efficiency for the test set are then combined as the final training set. The offline prediction models are trained on this combined set.

The advantage of offline training is that the model only needs to be trained once for a single scaling factor. However, this also means that for each possible scaling factor, a different model is needed. This might make offline training less fitting for arbitrary downscaling than for dyadic downscaling or transcoding between different video standards. Moreover, the performance of the models might be negatively affected if the videos in the test set differ from the training set.

B. Overview of Selected Machine Learning Algorithms

Several supervised machine learning algorithms, which label training data and predict the correct label for an input sample, can be used for classifying a set of input data. One of the goals of this paper is to investigate if there is a significant difference between machine learning techniques. Therefore, this paper elaborates on the performance of four commonly used algorithms: decision trees (DT) [15], random forests (RF) [23], adaptive boosting (AdaBoost) [24], and support vector machines (SVM) [14]. An overview of these algorithms will be provided next.

The DT algorithm is a technique for classification based on simple decision rules. The model consists of a root, internal nodes, leaf nodes, and branches. At the root and each internal node, the input sample is evaluated using the decision rule of that node. Depending on the outcome of that rule, the input sample follows one of the branches originating from the node. When the input sample reaches a leaf node, the DT model returns the prediction given by that leaf node. Implementation-wise, a tree consists of many if-else statements, which results in a low complexity for generating predictions. However, a disadvantage of this algorithm is that a tree might become overly complex, resulting in overfitting, which negatively impacts the performance of the DT model on test sets. A decision tree is also highly sensitive to small variations in the data set, meaning that it can produce different results when some samples are removed or added.

The RF and AdaBoost algorithms try to overcome the disadvantages of the DT algorithm by combining several trees. RF constructs decision trees by randomly selecting a subset of features from all available features to determine the decision rule at each internal node. The decision rule is based on a single feature from this random subset. On the other hand, AdaBoost improves the performance of a weak classifier such as DT by using an iterative approach. At each iteration, training samples are assigned weights, and incorrectly classified training samples will gain a larger weight. The cost of overcoming the disadvantages of the DT algorithm is that multiple trees need to be trained, which increases the complexity of the algorithms.

While the previous three algorithms use rule-based decision tree classification, SVM is more memory-based. In SVM, the dataset is mapped in a high-dimensional space with the goal of constructing a hyper-plane that maximizes the distance between samples belonging to different classes. As a result, SVM has higher storage and computing requirements than DT-based algorithms. Moreover, the complexity of the algorithm greatly increases with the number of features and samples.

C. Parameter Selection for Machine Learning

The classification performance depends not only on the learning algorithm, but also on the parameters of this algorithm. For a given algorithm, the classification accuracy varies widely when the parameter settings change. Therefore, a parameter selection method has to be deployed to select proper parameters for a given data set. The meaning of these parameters is summarized in Table I. In the tree-based algorithms, *max_depth* and *min_samples_leaf* of a tree are the most important parameters. In SVM, the radial basis function (rbf) kernel is used. Two parameters of the kernel, *C* and *γ* significantly affect the prediction performance. Additionally, the number of trees (*ntree*) in a random forest affects not only the coding performance but also the prediction time. A high *ntree* increases the accuracy of the prediction. However, it leads to a higher prediction complexity since the number of trees that need to be evaluated increases linearly with *ntree* [25]. The parameter selection for tree-based algorithms and SVM is presented first. Then, the proposed method for *ntree* selection is given.

TABLE I
THE NON-OPTIMIZED MACHINE LEARNING PARAMETERS

Algorithm	Parameter	Meaning	Default
SVM	*C*	Parameter *C* of the error term	1
	γ	Kernel coefficient for the kernel	0
Adaboost, DT, RF	*max_depth*	The maximum depth of the tree	None
	min_samples_leaf	The minimum number of samples required to be at a leaf node	1
RF	*ntree*	Number of trees in the forest	10

1) Optimization of General Parameters

General parameters of machine learning algorithms can be derived by using a 'grid-search' with cross-validation [26]. This approach tests all possible combinations for a set of parameters. The combination with the best cross-validation accuracy is chosen. In this paper, *max_depth* is selected from {3, 6, 9, 12} whereas *min_samples_leaf* is {1%, 2%, 3%, 4%, 5%, 6%} of the total number of samples in training data. *C* is in the range of {1, 10, 50, 100, 500, 1000} whereas *γ* is {10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}}.

2) Proposed ntree Selection Mechanism

Selecting *ntree* in RF should achieve a trade-off between prediction accuracy and prediction time. If a 'grid-search' is applied, a high *ntree* is often selected from the given set, resulting in a high prediction complexity. Hence, this approach is not an optimal method for selecting *ntree*.

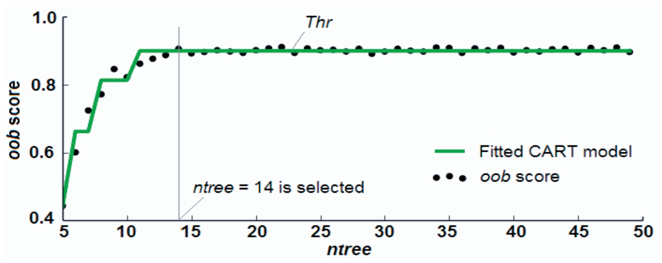

Fig. 3. *ntree* selection for RF, with block size = 32. The input bit stream was encoded using QP = 32 and is downsized by a factor of 2.

This paper proposes an efficient *ntree* selection mechanism based on the Out-of-Bag (*oob*) score. The *oob* score is a parameter to estimate the prediction accuracy of a model for an unknown data set [25]. When *ntree* increases, the score increases accordingly. However, after a certain number of trees, this score stabilizes around a threshold *Thr*. The threshold is derived by fitting a curve using a classification and regression trees (CART) [27] model as follows. First, *oob* scores are obtained for values of *ntree* from 5 to 50. A CART model is then generated to fit the *oob* scores (Fig. 3). *Thr* is set as the maximum *oob* score of the CART model, since this is the value to which the *oob* score converges. Then, the optimal *ntree* is selected by using (1), with oob_n being the oob score with n trees:

$$ntree = \underset{n\in\{5:50\}}{\arg min}\left\{oob_n \geq Thr\right\} \quad (1)$$

Since this *ntree* is the minimal value for which the *oob* score is equal to or greater than the chosen threshold, this number of trees is assumed to be the optimal trade-off between prediction accuracy and the complexity of the model.

D. *Content-Adaptive Feature Selection*

The splitting behavior of a CU in the output video is highly dependent on not only the features of the decoded video, but also on the coding information of the co-located CUs in the input video. Hence, a number of metrics describing the relevant characteristics of the content of the decoded video and the coding information of the input video have been acquired to get an accurate prediction model of the splitting behavior. However, using all of the features might not be optimal since irrelevant features may introduce noise. On the other hand, using only a small set of features results in a larger generalization error. Therefore, it is necessary to select the most important features [28]. First, an overview of the features is given. Then, a feature elimination mechanism is provided.

1) *Overview of all Features*

A total of 52 features were extracted from the decoded video and the input bit stream. These features were used as the initial feature set for training.

Features from the decoded video: A set of 36 features [29] from the decoded video were considered. They are based on Sobel filtering on frame pixel values, consecutive frame comparison, pixel value variations, and various combinations and variations of the aforementioned. The calculations were performed on the luminance component of the region of the picture that is co-located with the block for which the splitting behavior needs to be determined. These features mostly describe spatial and temporal activity in the picture. Examples include temporal and spatial indexes.

Features from the input bit stream: 16 features are based on coding information of the co-located input blocks. These features are the following:

- ✓ At the CU level, the *mean*, *variance*, *maximum*, and *minimum* of the input CU depths are included.
- ✓ At the TU level, the *mean*, *variance*, *maximum*, and *minimum* of input TU depths are also used.
- ✓ At the PU level, the depth of a PU is defined as the CU depth if the PU is not split. Otherwise; the PU depth equals the CU depth plus 1. The *mean*, variance, *maximum*, and *minimum* of the input PU depths are included.
- ✓ Additionally, the *variance* of the input motion vectors is taken into account.
- ✓ The last two features are the *variance* and the *mean* of the transform coefficient variance.

2) *Feature Elimination using Random Forests*

Selection of effective and relevant features is crucial for classification. A good feature selector could help reduce training time, prediction time, as well as reduce memory requirements. In addition, this eliminates irrelevant or noisy features, which can result in an increase of the prediction accuracy. Feature selection approaches can be classified into three categories including filter, wrapper, and embedded [30]:

- ✓ The filter approach uses general characteristics of the training data to select interesting features. Since the relationships between features are not considered, this method tends to select redundant features. Thanks to the computational efficiency, the filter method is usually chosen as a preprocessing method.
- ✓ The wrapper approach performs a preprocessing step. However, it uses a machine learning algorithm as a part of the feature selection process. For example, the RF algorithm inherently allows the calculation of feature importances [31]. The features with the highest importance can then be retained. This method provides a superior performance compared to the filter approach. However, its complexity depends on the training complexity of the machine learning algorithm.
- ✓ The embedded approach performs feature selection automatically as part of the machine learning algorithm.

This paper applies a content-adaptive feature selection algorithm based on the wrapper approach. The machine learning algorithm used in the selection process is RF, which provides high prediction performance and low training complexity. The selection process consists of the following three steps:

Feature ranking: The importance of each feature is determined for 50 runs of RF training (Fig. 4). This importance is calculated during each run as the expected fraction of samples that the feature will contribute to. The features are then sorted in descending order based on the average value over the 50 runs.

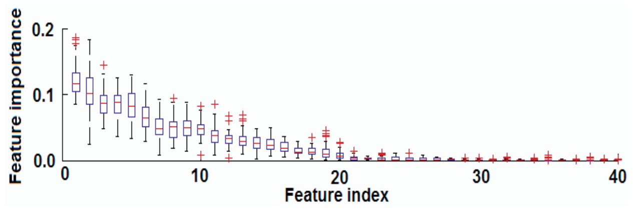

Fig. 4. Feature importance for 50 runs. The variance of the feature importance becomes smaller as the average importance reaches zero.

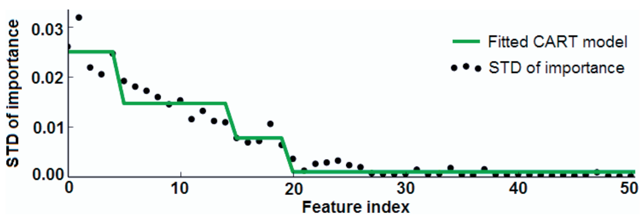

Fig. 5. Feature selection for transcoding the FourPeople sequence. The input video (QP = 27), is downsized by a scaling factor of 1.5. The standard deviation of important features is larger than for the noisy features, which have a standard deviation close to zero. The threshold, which is the minimum of the CART model, results in selecting the 29 most important features.

Determining the threshold: The standard deviation (STD) of the feature importance of each feature is calculated. These results are plotted using the same order generated by the feature ranking step. A CART model is then fitted to this data (Fig. 5). The threshold for feature selection is set as the minimum prediction of the CART model. This threshold is a measure for the minimal noise of the feature importance. If a feature has an importance below this threshold, the importance can therefore be considered as 0.

Feature elimination: Only features with an average feature score greater than the threshold are retained.

V. Transcoding Complexity Control

Reducing the transcoding complexity usually results in a decrease in coding performance. However, for some use cases, a high coding performance might be necessary. Therefore, an optimal trade-off should be made between transcoding complexity and coding performance.

In order to achieve a trade-off between complexity and coding performance, some machine learning algorithms such as decision trees and random forests can be modified to output probabilistic values for each prediction. This means that the splitting behavior of a CU is predicted with a confidence c. To achieve higher coding performance, only decisions with a higher c should be allowed. To control this trade-off between coding performance and transcoding complexity, a threshold T_c is defined. If c is larger than T_c, the predicted split-flag directly controls the splitting behavior of CUs. Otherwise, the CU is fully evaluated for both split and not split. With a high T_c, the number of full evaluations increases, resulting in a higher transcoding complexity while improving coding performance. Consequently, the transcoding complexity can be controlled by adjusting T_c to achieve a trade-off between complexity and coding performance.

VI. Experimental Results

The proposed downsizing transcoder was tested with various machine learning algorithms. First, the methodology of experiments is presented. Then, the machine learning algorithms are compared to each other, and they are also compared to a trivial method. The performance of online and offline training strategies and the effects of optimizations are considered as well. Thereafter, the influence of the complexity control mechanism is shown.

A. Methodology of the Experiments

In the following experiments, the original video is encoded using an HEVC encoder following a low delay prediction structure. This structure is characterized by an IPPP prediction order using four reference frames. The QP is set to {22, 27, 32, 37}. Version 12 of the HEVC reference software [32] is used. Search mode "TZSearch" and "FEN" (fast encoder decision) are enabled. All sequences (18 in total) have input resolutions varying from 832x480 up to 1920x1080 pixels except two sequences Traffic (3840x2048) and PeopleOnStreet (3840x2160) [22] which are bigger.

After the initial encoding step, the HEVC bit stream is decoded and downsized by a scaling factor of $\sigma \in \{1.33, 2.00, 4.00, 1.50\}$ using a non-normative downsampling filter [20]. Although the algorithm can handle any possible scaling factor, these factors have been selected as a representative set. They respectively reduce the dimensions of the picture to 3/4, 2/4, 1/4 and 2/3 of the original dimensions. A maximum reduction of 1/4 is used since this decreases the total number of pixels with 1/16. Additionally, the value of 2/3 was tested since this occurs in a scenario where 1080p video is downsized to 720p.

Following the downsizing step, the resulting video is re-encoded using the proposed machine learning prediction models with the number of training frames N set to 10. Since only the impact of σ on the transcoding performance is evaluated, the other coding conditions of the output video (profile, QP) are left unchanged as in the input video.

The performance of the proposed transcoding technique is evaluated by comparing it to a non-optimized decoder-encoder cascade (*Ref*) in terms of Bjøntegaard Delta Bit Rate (BDBR) [33] and time saving (TS). The downsized version of the original video is used as a reference for calculating the Peak Signal-to-Noise Ratio (PSNR). Bjøntegaard Delta PSNR (BDPSNR) has been evaluated as well. The experimental results show that the BDPSNR loss and BDBR have the same behavior when comparing different transcoding strategies. Therefore, only BDBR is mentioned in this paper. Time saving is given by (2) in which T_{Pro} is the total transcoding time using the proposed method and T_{Ref} is the total transcoding time using *Ref*. For the online training strategy, training time is included in the transcoding time.

$$TS(\%) = \frac{T_{Ref}(ms) - T_{Pro}(ms)}{T_{Ref}(ms)} \times 100 \qquad (2)$$

This paper proposes a trivial method for comparison with the machine learning based transcoding techniques. The trivial method predicts the splitting behavior of an output CU using the mean depth of the co-located input CUs. If the mean depth is higher than the current depth of the output CU, the output CU is split immediately. Else, the output CU is not split further and is encoded using the current depth. The motivation of using the mean depth of the input co-located CUs is that this feature and the output splitting behavior have a high correlation, as the importance of this feature has been observed to consistently rank high compared to other features.

B. Evaluation of the Performance of Machine Learning Algorithms for Downsizing

The performance of different machine learning algorithms without optimizing the parameters is presented first. In these tests, the online training mechanism was used. Then, the results using parameter optimization are elaborated on. The offline training mechanism is also evaluated. Finally, the results of feature selection are given.

In a first experiment, non-optimized parameters are used. The chosen values for the parameters with a high impact on the prediction accuracy are given in Table I [25]. Note that setting *max_depth* to 'None' and *min_samples_leaf* to 1 means that the complete tree will be generated, i.e. the model will likely be overfitted to the training data.

The experimental result of using non-optimized machine learning parameters is shown in Table II. It is seen that all machine learning methods achieve the same complexity reduction as the trivial method (about 71% on average). These results are similar since the CU structure is always completely predicted, which means that the encoder can skip the complete CU partitioning. Any slight differences in complexity reduction are due to the fact that less blocks need to be evaluated if the predicted structure contains more large blocks. In the other tables (Table III, IV, and V) the complexity reduction is similar for the same reasons.

As is also seen in Table II, contrary to the TS, the performance of the methods differs in terms of BDBR. Using AdaBoost and DT leads to high bit rate increases (around 11.7% on average for each). These two methods perform worse than the trivial method, which has a BDBR increase of 8.44%. The performance of RF and SVM is better with an increase of 7.42% and 8.85%, respectively. Compared to the trivial method, while the performance of RF is slightly better, the performance of other machine learning algorithms is worse. In other words, non-optimized machine learning models produce a bad fit for the given data set. Therefore, they should always be optimized if machine learning is used for transcoding. In terms of BDPSNR, quality losses of 0.50 dB, 0.51 dB, 0.38 dB, 0.33 dB, and 0.37 dB are measured for DT, AdaBoost, SVM, RF, and Trivial, respectively.

The transcoding performance with optimized machine learning algorithms as described in Section IV is presented in Table III, where RF_{200} is the result of the RF algorithm with a fixed *ntree* of 200. These experiments use the complete set of 52 features. As can be seen from these results, the coding performance significantly improves compared to using non-optimized parameters. With a slightly different complexity reduction, the bit rate penalty of using optimized parameters is clearly reduced. The SVM, AdaBoost, and DT algorithms demonstrate a similar performance with bit rate increases of 7.16%, 7.15%, and 7.38%, respectively. The RF algorithm is better than the others with a 5.41% bit rate penalty. By using the proposed *ntree* selection, the number of trees varies from 10 to 30. Although the number of trees is very low compared to 200, the performance of RF remains the same whereas the prediction time is significantly reduced, indicating that the proposed *ntree* selection indeed selects an optimal number of trees.

The results of offline training are shown in Table IV. Online training performs better than offline training even when cross-validation is used to select the training sequences for offline training. For example, the online model has an average BDBR of 5.41% for RF, whereas the offline model has a BDBR of 7.07%. The offline model for SVM has a higher BDBR than the trivial model since optimal machine learning parameters of an offline model may not be optimal for every sequence. Moreover, to apply the offline strategy to other downsizing scaling factors, the model would have to be retrained. Consequently, to achieve the best performance, online training should be used whenever possible. A drawback of online training compared to offline training is training complexity. The time saving of the online training strategy is slightly lower than offline training (around 2%) since the 10 first frames are fully encoded followed by a training phase.

Finally, the performance of the feature elimination algorithm has also been analyzed. The result of transcoding with optimized parameters and feature selection is shown in Table V. In general, the feature elimination algorithm succeeds in reducing noisy features while retaining a similar coding performance. A comparison of the results in Table V and Table III, where the complete feature set was used, shows that the BDBR remains similar after feature selection. The influence of the feature selection is higher with a BDBR increase of 5.50%, 7.33%, and 7.59% for RF, SVM, and DT, respectively. On the other hand, the feature selection mechanism decreases the BDBR of AdaBoost to 7.10%.

C. Transcoding Complexity Control Scheme

The transcoding complexity can be controlled by adjusting the threshold of the confidence of prediction T_c. To investigate the effect of the threshold on the transcoding complexity, this threshold is varied from 0.5 to 0.9 with a step of 0.1. The machine learning algorithm used in this experiment is RF, which offers the best transcoding performance among the investigated algorithms. The experimental results with different thresholds are depicted in Fig. 6.

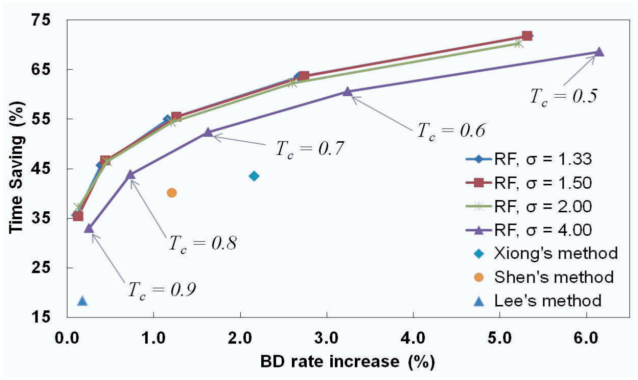

Fig. 6. The transcoding complexity using RF can be controlled by adjusting the threshold of the prediction confidence.

When the threshold is increased, the bit rate penalty and the complexity reduction reduce accordingly. In particularly, when T_c is set to 0.5, the complexity reduction is around 70% with a bit rate increase of around 6% for σ = 4.0 and 5% for the other cases. In contrast, when T_c is 0.9, the complexity reduction drops to about 33% with a negligible bit rate penalty. The performance of RF is compared with three state-of-the-art fast CU size decision algorithms including Lee's algorithm [17], Shen's algorithm [18], and Xiong's algorithm [19], which are applied to encode the down-sized version of the reconstructed video. The average performance of each algorithm is depicted in Fig. 6. The proposed method outperforms these algorithms when changing the threshold T_c of the prediction confidence. When T_c is 0.9, the proposed method results in the same bit rate increase (0.15%) compared to Lee's method. However, the proposed method achieves 35% time saving while the time saving for Lee's method is only 19%. When T_c is 0.8, the proposed method, Shen's method, and Xiong's method achieve a similar complexity reduction (around 45%). However, the proposed method results in only 0.51% BDBR while the other techniques have higher bit rate penalties (1.21% for Shen's method and 2.16% for Xiong's method).

TABLE V
RESULTS FOR AN ONLINE MACHINE LEARNING STRATEGY, USING NON-OPTIMIZED PARAMETERS

σ	BDBR (%)					TS (%)				
	Trivial	RF	SVM	Adaboost	DT	Trivial	RF	SVM	Adaboost	DT
1.33	9.20	7.37	9.27	12.44	12.69	71.65	72.00	71.41	72.01	72.03
1.50	8.26	7.14	8.87	11.60	11.79	72.23	71.99	71.24	72.00	72.01
2.00	7.17	7.13	7.98	10.87	10.96	71.16	70.57	70.34	70.23	70.49
4.00	9.12	8.03	9.27	11.88	11.40	69.49	69.01	69.05	68.54	69.12
Average	8.44	7.42	8.85	11.70	11.71	71.13	70.89	70.51	70.70	70.91

TABLE V
RESULTS FOR AN ONLINE MACHINE LEARNING STRATEGY, USING OPTIMIZED PARAMETERS, WITHOUT FEATURE SELECTION

σ	BDBR (%)					TS (%)				
	RF	RF_{200}	SVM	Adaboost	DT	RF	RF_{200}	SVM	Adaboost	DT
1.33	5.29	5.27	6.67	7.25	7.29	71.01	69.94	72.14	71.19	71.93
1.50	5.27	5.25	6.98	7.03	6.89	71.15	71.01	72.02	71.05	71.50
2.00	5.15	5.11	7.04	6.75	6.87	70.00	69.46	70.11	70.12	70.58
4.00	5.95	5.91	7.94	7.60	8.49	68.25	68.03	67.54	68.27	69.02
Average	5.41	5.38	7.16	7.15	7.38	70.10	69.61	70.44	70.16	70.76

TABLE V
RESULTS FOR AN OFFLINE MACHINE LEARNING STRATEGY, USING OPTIMIZED PARAMETERS, WITHOUT FEATURE SELECTION

σ	BDBR (%)					TS (%)				
	Trivial	RF	SVM	Adaboost	DT	Trivial	RF	SVM	Adaboost	DT
1.33	9.20	6.75	8.56	7.31	7.42	71.65	73.71	74.47	73.93	74.25
1.50	8.26	6.86	8.47	7.25	8.24	72.23	73.61	73.65	73.94	74.08
2.00	7.17	6.82	11.12	7.37	7.51	71.16	72.23	73.00	72.38	72.27
4.00	9.12	7.85	9.82	8.43	9.48	69.49	70.44	70.95	70.82	71.05
Average	8.44	7.07	9.49	7.68	8.16	71.13	72.50	73.02	72.77	73.04

TABLE V
RESULTS FOR AN ONLINE MACHINE LEARNING STRATEGY, USING OPTIMIZED PARAMETERS, WITH FEATURE SELECTION

σ	BDBR (%)					TS (%)				
	Trivial	RF	SVM	Adaboost	DT	Trivial	RF	SVM	Adaboost	DT
1.33	9.20	5.33	6.89	7.21	7.36	71.65	71.06	72.11	71.28	71.91
1.50	8.26	5.32	7.07	7.01	7.00	72.23	71.21	71.42	72.14	71.98
2.00	7.17	5.22	6.75	6.78	6.92	71.16	70.28	69.36	70.02	70.17
4.00	9.12	6.15	8.61	7.41	8.83	69.49	68.03	67.26	67.51	68.22
Average	8.44	5.50	7.33	7.10	7.59	71.13	70.15	70.04	70.24	70.57

VII. CONCLUSION

In this paper, different machine learning strategies for downsizing HEVC video are investigated. The pre-encoded HEVC video stream is decoded. The reconstructed video is then downsized using an arbitrary factor, which allows adapting the video to the network and/or device constraints. Afterwards, the resized video is re-encoded. The machine learning models utilize the correlation between coding information of the input and output coding units to accelerate the re-encoding process. Different optimized and non-optimized machine learning algorithms have been tested with both online and offline training strategies. The effect of an adaptive feature selection algorithm has also been investigated.

Experimental results have shown that machine learning algorithms should only be used when optimized, since otherwise a trivial method might perform better.

If machine learning methods are used with the proposed optimizations, an online training strategy is preferred over an offline training strategy. This makes the models more adaptive to the content and results in a higher coding efficiency. Additionally, the transcoding complexity can be controlled to achieve a trade-off between transcoding complexity and coding performance. Among the investigated machine learning algorithms, the random forests resulted in the best transcoding performance by reducing 70% complexity on average with a bit rate increase of 5.4%. With a negligible bit rate increase, this method can reduce the transcoding complexity with 35%.

References

[1] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," *IEEE Trans. Consum. Electron.*, vol. 44, no. 1, pp. 88-98, Feb. 1998.

[2] J. De Cock, S. Notebaert, P. Lambert, and R. Van de Walle, "Requantization transcoding for H.264/AVC video coding," *Signal Processing: Image Communication*, vol. 25, no. 4, pp. 235-254, Apr. 2010.

[3] L. Pham Van, *et al.*, "Fast motion estimation for closed-loop HEVC transrating," *in Proc. IEEE International Conference on Image Processing*, Paris, France, pp. 2492-2496, Oct. 2014.

[4] M.-J. Chen, M.-C. Chu, and S.-Y. Lo, "Motion vector composition algorithm for spatial scalability in compressed video," *IEEE Trans. Consum. Electron.*, vol. 47, no. 3, pp. 319-325, Aug. 2001.

[5] C. Zhang, S. Zheng, C. Yuan, and F. Wang, "A novel low-complexity and high-performance frame-skipping transcoder in DCT domain," *IEEE Trans. Consum. Electron.*, vol. 51, no. 4, pp. 1306-1312, Nov. 2005.

[6] L. Pham Van, J. De Praeter, G. Van Wallendael, J. De Cock, and R. Van de Walle, "Machine learning for arbitrary downsizing of pre-encoded video in HEVC," *in Proc. IEEE International Conference on Consumer Electronics*, Las Vegas, USA, pp. 431-432, Jan. 2015.

[7] Y.-P. Tam and H. Sun, "Fast motion re-estimation for arbitrary downsizing video transcoding using H.264/AVC standard," *IEEE Trans. Consum. Electron.*, vol. 50, no. 3, pp. 887-894, Aug. 2004.

[8] Z.-G. Liu, Y. Yang, and X.-H. Ji, "Fast macroblock mode decision for H.264/AVC baseline profile video transcoder based on support vector machines," *Multimedia Syst.*, vol. 18, no. 5, pp. 359-372, Dec. 2011.

[9] G. Fernández-Escribano, *et al.*, "Low-complexity heterogeneous video transcoding using data mining," *IEEE Trans. Multimedia*, vol. 10, no. 2, pp. 286-299, Feb. 2008.

[10] L. Pham Van, *et al.*, "Fast transrating for high efficiency video coding based on machine learning," *in Proc. IEEE International Conference on Image Processing*, Melbourne, Australia, pp. 1573-1577, Sep. 2013.

[11] J. Martínez, G. Fernández-Escribano, H. Kalva, W. Fernando, and P. Cuenca, "Wyner-Ziv to H.264 video transcoder for low cost video encoding," *IEEE Trans. Consum. Electron.*, vol. 55, no. 3, pp. 1453-1461, Aug. 2009.

[12] J. De Praeter, *et al.*, "Efficient transcoding for spatial misaligned compositions for HEVC," *in Proc. IEEE International Conference on Image Processing*, Paris, France, pp. 2487-2491, Oct. 2014.

[13] E. Peixoto, T. Shanableh, and E. Izquierdo, "H.264/AVC to HEVC video transcoder based on dynamic thresholding and content modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 99-112, Jan. 2014.

[14] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 237-297, Sep. 1995.

[15] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theory and Applications*, New York: World Scientific, 2008.

[16] G. Sullivan, J. 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.

[17] H. S. Lee, K. Y. Kim, T. R. Kim, and G. H. Park, "Fast encoding algorithm based on depth of coding-unit for high efficiency video coding," *Opt. Eng.*, vol. 51, no. 6, pp. 1-11, Jun. 2012.

[18] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang, "An effective CU size decision method for HEVC encoders," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 465-470, Feb. 2013.

[19] J. Xiong, H. Li, Q. Wu, and F. Meng, "A fast HEVC inter CU selection method based on pyramid motion divergence," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 559-564, Feb. 2014.

[20] S. Sun and J. Reichel, "AHG Report on Spatial Scalability Resampling, Joint Video Team," JVT-R006, Bangkok, Thailand, Jan. 2006.

[21] J. De Cock, S. Notebaert, K. Vermeirsch, P. Lambert, and R. Van de Walle, "Dyadic spatial resolution reduction transcoding for H.264/AVC," *Multimedia Syst.*, vol. 16, no. 2, pp. 139-149, Jan. 2010.

[22] F. Bossen, "Common test conditions and software reference configurations," JCTVC-J1100, Geneva, CH, May 2012.

[23] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5-32, Oct. 2001.

[24] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119-139, Aug. 1997.

[25] F. Pedregosa, *et al.*, "Scikit-learn: machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825-2830, Nov. 2011.

[26] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Mach. Learn.*, vol. 46, no. 1-3, pp. 131-159, Feb. 2002.

[27] L. Barber, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*, Chapman and Hall/CRC, New York, 1984.

[28] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78-87, Oct. 2012.

[29] S. Jeannin and A. Divakaran, "MPEG-7 visual motion descriptors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 6, pp. 720-724, Jun. 2001.

[30] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157-1182, Mar. 2003.

[31] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern Recogn. Lett.*, vol. 31, no. 14, pp. 2225-2236, Oct. 2010.

[32] K. McCann, *et al.*, "High efficiency video coding (HEVC) test model 12 (HM12) encoder description," JCTVC-N1002, Vienna, Austria, Jul. 2013.

[33] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," document VCEG-M33 of ITU-T Video Coding Experts Group (VCEG), Apr. 2001.

Biographies

Luong Pham Van obtained the M. Sc. degree in Electrical Engineering from Sungkyunkwan University, Korea in 2011. Currently, he is with Multimedia Lab, Ghent University, where he is working towards a Ph.D. His main research interests include video compression, adaptation of video streams, transcoding, next generation video compression and machine learning.

Johan De Praeter received the M.Sc. degree in Computer Science Engineering from Ghent University, Belgium, in 2013. Since then he joined Multimedia Lab, Ghent University. At Multimedia Lab he is working towards a Ph.D. His research interests include video compression, High Efficiency Video Coding, machine learning, and transcoding.

Glenn Van Wallendael obtained the M.Sc. degree in Applied Engineering from the University College of Antwerp, Belgium, in 2006 and the M.Sc. degree in Engineering from Ghent University, Belgium in 2008. Afterwards, he worked towards a Ph.D. at Multimedia Lab, Ghent University, with the financial support of the Agency for Innovation by Science and Technology (IWT). Currently, he continues working in the same group as a post-doctoral researcher. His main topics of interest are video compression including scalable video compression and transcoding.

Jan De Cock obtained the M.S. and Ph.D. degrees in Engineering from Ghent University, Belgium, in 2004 and 2009, respectively. Since 2004 he has been working at Multimedia Lab, Ghent University, iMinds. In 2010, he obtained a postdoctoral research fellowship from the Flemish Agency for Innovation by Science and Technology (IWT) and in 2012, a postdoctoral research fellowship from the Research Foundation Flanders (FWO). His research interests include high-efficiency video coding and transcoding, scalable video coding, and multimedia applications.

Rik Van de Walle received his M.Sc. and PhD degrees in Engineering from Ghent University, Belgium in 1994 and 1998 respectively. After a visiting scholarship at the University of Arizona (Tucson, USA), he returned to Ghent University, where he became professor of multimedia systems and applications, and head of the Multimedia Lab. His current research interests include multimedia content delivery, presentation and archiving, coding and description of multimedia data, content adaptation, and interactive (mobile) multimedia applications.