Deep Learning-based Video Coding: Benchmarking and One Step Forward

Helena Oliveira

Instituto Superior Técnico, Universidade de Lisboa - Instituto de Telecomunicações, Lisboa, Portugal

Abstract— **Video coding is the process that takes raw, original, digital video data and codes it into a binary representation, targeting efficient compression by exploiting the spatial, temporal, statistical and perceptual redundancies. Until recently, video coding was exclusively performed with traditional video coding methods that target the exploitation of these redundancies with handcrafted algorithms. Lately, however, deep learning methods, and namely attention models, have shown great results for tasks related to computer vision and image processing, so it is only natural that this novel technology is also applied to other signal processing challenges like image and video coding. This paper reviews several state-of-the-art deep learning-based video coding solutions and provides a solid and meaningful benchmarking for some of the reviewed solutions with publicly available software against the most recent traditional video codec (VVC), to provide awareness into the current performance of these video codecs. Afterwards, the second objective of this paper is to improve a deep learning-based solution. As such, the training of the chosen codec (RLVC) will be replicated to guarantee the ability to replicate results, so that later this solution can be extended with an attention model. While the results achieved with the final extended codec do not yet show consistent improvement overall, since the integration of attention models into video coding solution is still a novel research path, promising developments are expected in the future.**

Keywords— ***video coding, compression efficiency, deep learning, neural networks, attention models***

I. INTRODUCTION

Visual communications have a fundamental role in Human societies. In the digital era, this has led to the explosion of digital image and video-based applications and services, notably following the democratization of image and video acquisition, storage, and streaming. For this explosion to happen, video coding standards have been fundamental and have been evolving over the years to offer increasing compression efficiency, notably the more recent Advanced Video Coding (H.264/AVC), High Efficiency Video Coding (HEVC) and Versatile Video Coding (VVC) standards. The maturity and high performance of these video coding standards and the associated ecosystems have been fundamental during the recent COVID-19 pandemic, as video-based services and communications had an essential role on the mitigation of its negative impacts. These video coding standards include some key coding modules, notably motion estimation, temporal and spatial prediction, transform, quantization, and entropy coding, which have been carefully handcrafted, targeting the highest compression efficiency for the target range of rates/qualities in a widespread set of applications and services.

Recently, the multimedia arena has been shaken by the impact of deep learning (DL)-based technologies, notably for computer vision tasks, e.g., classification, detection, and recognition, with above human performance levels often achieved. In this context, it was just a question of time for DL-based tools to enter the image and video coding arenas since it is impossible to ignore its potential benefits regarding conventional coding approaches. DL-based coding solutions create the so-called *latent representation*, containing the most important learned content features to describe the input data, following a training process where a loss function controls the DL-based model optimization. The training process is at the heart of the DL-based media representation paradigm, especially when the goal is to have a single compressed representation, which is efficient both for fidelity decoding as well as computer vision tasks, e.g., classification and recognition, since these goals are both important for an increasing number of application scenarios.

One of the most recent developments in DL is related to attention models, which are techniques for neural networks (NN) that enable them to process complex inputs and focus on specific aspects of these inputs [1], which have already been exploited to solve some problems with great success, e.g., Neural Machine Translation. The first DL-based attention models were designed to be integrated into architectures based on Recurrent Neural Networks (RNN), with the goal of improving the performance of Natural Language Processing (NLP) tasks [2], which highly depend on temporal correlations, and have since proven to be a great addition; so much so that solutions based only on attention models, like the Transformer [3], have shown remarkable results. Recently, attention models have been applied with large success to computer vision tasks [4] [5], notably video classification [6]. Thus, since attention models have been created to deal with temporal data and have already been applied successfully to image and video classification tasks, this paper speculates that they might come to also have positive impact on DL-based video coding solutions.

Recent works have shown that besides the competitive image and video compression performances, DL-based coding solutions allow extending the utility of the compressed representations by offering three key advantages: i) a single efficient (compressed) representation for both humans and machines, e.g. autonomous vehicles; ii) reduction of the complexity resources associated to computer vision tasks as already starting from compressed domain features, thus at least partly skipping feature extraction (from decoded content); and iii) better analysis accuracy by allowing the computer vision tasks to use the compressed domain features extracted from the original image/video data instead of extracting them from the lossy decoded image/video as for conventional coding solutions where feature extraction happens after full decoding.

In this context, the first objective of this paper is to perform a solid, meaningful, and extensive benchmarking of the compression performance of some recent DL-based video coding solutions regarding the most powerful and recent conventional video coding solution, the VVC standard [7]. This comparison will be performed under the largely adopted JVET defined Common Test Conditions (CTC) and evaluation procedures for NN-based video coding technology [8]. This type of novel performance comparison is critical to know in a reliable way how far some recent DL-based video coding solutions are from the well-established conventional video coding solutions in terms of compression performance. The second objective of this paper is to improve a deep learning-based video coding solution from the literature, so-called Recurrent Learned Video Compression (RLVC), by extending it with an attention model and analyzing the preliminary compression results.

To achieve its purposes, this paper is organized as follows: Section II offers a brief description of the assessed DL-based video coding solutions. Next, Section III describes the adopted test conditions and evaluation procedures, Section IV presents and analyzes the performance benchmarking results. Section V describes the training process and then analyses the replication of results achieved with the new obtained RLVC models, and Section VI will report on the preliminary results of the integration of an attention model into the RLVC. Finally, Section VII concludes the paper.

II. Selected Learning-based Video Codecs

This section describes the selected DL-based video codecs which will be benchmarked in this paper. This set of codecs was determined by the publicly available software as only results available in published papers would not be enough to perform a solid comparison under common test conditions. This means all performance results later presented for the JVET conditions have been generated in the context of this paper, which represents a major, novel contribution.

A. Deep Video Compression (DVC)

The DVC [9] DL-based video coding solution adopts a classical, hybrid video coding architecture, where each module is replaced by a DL-based tool, thus originating a one-to-one correspondence between the classical coding modules and the DL-based models, see Figure 1.

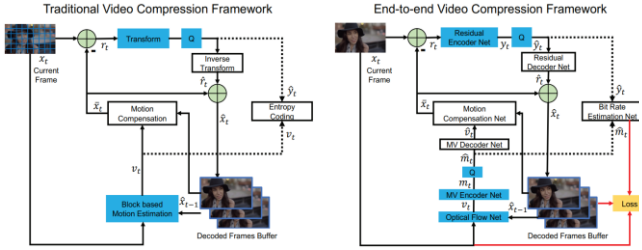

Figure 1: Traditional video coding architecture (left); DVC architecture (right) [9].

In the DVC, motion estimation is performed using a Convolutional Neural Network (CNN) to extract the optical flow, always from the previously decoded frame. The resulting residual information is coded using a CNN-based encoder-decoder network and the motion-associated latents are quantized to save rate while still reaching a good enough motion representation. The decoded optical flow is used to warp the previous decoded frame to estimate the motion compensated prediction for the current frame. However, since the motion compensated frame has artifacts, it is processed by a CNN, together with the previous decoded frame and the decoded optical flow, to obtain a refined prediction frame. Lastly, the prediction residual is compressed into a latent representation using a non-linear NN [10], which is quantized to obtain several Rate-Distortion (RD) points. The whole DVC architecture is trained with a RD loss function, using as end-to-end distortion metric the Mean Square Error (MSE) between the original and decoded frames, while the bitrate is estimated using a rate estimation module.

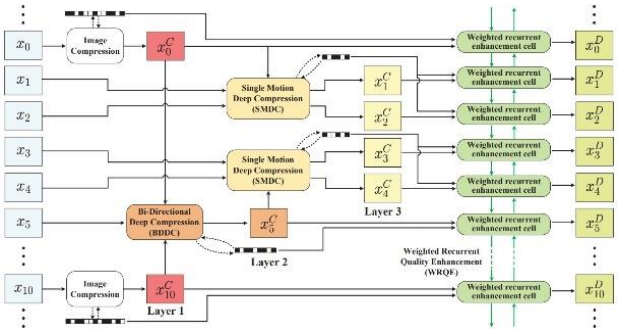

Figure 2: HLVC architecture, illustration for the first GOP [13].

The OpenDVC implementation of this coding solution made available by Yang *et al.* is used in Section IV for performance assessment [11][12]. The OpenDVC implementation offers two different DL-based models: one trained using MSE as the distortion metric, and another trained using (1 – MS-SSIM) as the distortion metric. The several DVC DL-based models are trained in a progressive manner, starting with the motion estimation network, which is trained using a distortion-only loss function; after, the motion compression, motion compensation and the full end-to-end networks are trained using a RD loss function.

B. Hierarchical Learned Video Compression (HLVC)

The HLVC [13] DL-based video coding solution adopts a hierarchical coding architecture inside a Group of Pictures (GOP). At the decoder, the frames are decoded and enhanced using a recurrent network to increase the decoded frames' quality, see Figure 2.

The first HLVC layer includes the first and last frames of the GOP, which are coded with a state-of-the-art image coding solution, at the highest quality. The second HLVC layer codes the middle frame of the GOP with medium quality, using a NN for bi-directional prediction, and the frames at the edge of the GOP as anchors. The third and last HLVC layer codes the remaining frames of the GOP with lower quality, using the closest already decoded frame as anchor (from the first or second layers); only uni-directional (temporal) prediction and a single motion map are used to code two adjacent frames. At the decoder side, after each frame is decoded by the corresponding layer, all frames are processed by an enhancement RNN which leverages on the frames with better quality to improve the quality of the remaining frames.

Each NN in HLVC is trained separately. The networks at the encoder and decoder sides use an RD loss function, while the recurrent enhancement network is trained with a distortion only loss function; since it is applied only at the decoder, rate is not an issue. The HLVC networks were trained with both MSE and (1 – MS-SSIM) distortion, to obtain two different coding models. The HLVC software implementation is publicly available [14], and can be used in the so-called *fast* and *slow modes*. While the fast mode has predefined networks for the coding process, the slow mode tries different network models to encode each frame and selects the one with the best result to achieve the best possible RD performance at the cost

of increasing computational complexity; this is the coding mode used in Section IV for performance assessment.

C. Recurrent Learned Video Compression (RLVC)

The RLVC [15] DL-based video coding solution includes two recurrent auto-encoders (RAE) and two recurrent probability models (RPM), two of each for dealing with motion and residue information, see Figure 3. Due to its recurrent nature, the RAE allows using temporal information from several frames at a time; in this context, previous inputs are given as cell and hidden states and used by the ConvLSTM layer to help generate the latent representation for the current frame, instead of choosing only specific frames as anchors like most DL-based video codecs. The RPM networks model the probability mass function of the obtained latent representations, based on previous iterations; this information is then applied to adaptive arithmetic coding.

In the RLVC codec, the first frame of each GOP is coded with a state-of-the-art image coding solution, either DL-based or conventional. For the remaining frames, motion estimation is performed using a pyramid optical flow network [16]. The RAE codes the estimated motion, considering the cell and hidden states passed on, which express inputs from previous frames; the decoded motion estimation is applied to create a motion compensated prediction. Next, the residual between the original and motion compensated frames is obtained and coded using another RAE. The RPM is used to recurrently predict the temporally conditional probability mass function, to reduce the rate when entropy coding the latent representations.

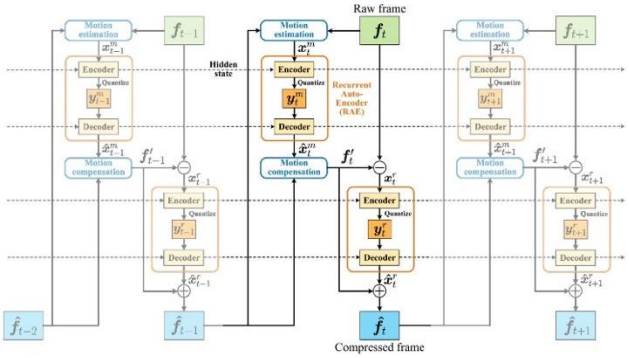

Figure 3: RLVC architeture [15].

The RLVC codec is trained in a progressive manner. First, the motion estimation network is trained with a distortion only loss function; after, the RAE and motion compensation networks are trained using a RD loss function and, finally, the full network is trained end-to-end with a RD loss function. The RLVC is trained with two different distortion metrics, notably MSE and (1 – MS-SSIM), thus obtaining two different coding models. The RLVC software has been made publicly available by the authors and is used in Section IV for performance assessment [17].

III. Test Conditions and Evaluation Procedures

This section will describe the test conditions and evaluation procedures adopted for the performance benchmarking exercise targeted in this paper, which are largely those specified by JVET in the document "JVET common test conditions and evaluation procedures for neural network-based video coding technology", called JVET NN CTC in the following [8].

A. Test Material and Coding Rates

The selected video test materials correspond to the key classes recommended in the JVET NN CTC [8], as shown in Table 1. All original videos are in the YUV color space with 4:2:0 subsampling; there are sequences at 8 and 10-bit per sample, which has required some software adaptations for the learning-based codecs. For the DL-based coding solutions, the various rates are obtained with four models trained for several bitrates using MSE-RGB in the loss function (since the JVET NN CTC uses PSNR as quality metric), while for VVC the quantization steps specified in the JVET NN CTC [8] are used.

Table 1: Test video sequences and characteristics.

Class	Sequence Name	Nº of Frames	Frame Rate	Bit-depth	Spatial Resolution
B	*MarketPlace*	600	60	10	1920×1080
	RitualDance	600	60	10	1920×1080
	Cactus	500	50	8	1920×1080
	BasketballDrive	500	50	8	1920×1080
	BQTerrace	600	60	8	1920×1080
C	*RaceHorses*	300	30	8	832×480
	BQMall	600	60	8	832×480
	PartyScene	500	50	8	832×480
	BasketballDrill	500	50	8	832×480
D	*RaceHorses*	300	30	8	416×240
	BQSquare	600	60	8	416×240
	BlowingBubbles	500	50	8	416×240
	BasketballPass	500	50	8	416×240

B. Coding Benchmarks and Pipelines

The benchmark for this performance assessment is, naturally, the Versatile Video Coding (VVC) standard [7], the most recent and efficient representative of a succession of video coding standards based on the so-called *hybrid coding architecture*. This is naturally a very challenging benchmarking for the DL-based coding solutions, which are just emerging and did not have yet time to mature; however, this is also the most relevant benchmark to assess the performance gap between the technologies under comparison. In this context, three VVC coding configurations will be used: i) VVC Random Access (RA); ii) VVC Low Delay P (only P frames); and iii) VVC Intra. The VVC reference software version 11 [18] has been used for VVC coding using the JVET provided configurations [8]. According to JVET NN CTC [8], VVC coding must be performed at 10-bit depth even if the content is available at 8-bit since this improves the VVC compression performance. This implies that the 8-bit videos had to be converted to 10-bit videos, in this case by adding two bits as '10', i.e., in the central bin position.

To perform a fair benchmarking, the same had to happen with the DL-based codecs which were not prepared to code 10-bit videos; in this context, the software for the three selected DL-based codecs had to be upgraded to code videos at 10-bit instead of the previous 8-bit depth used in the original implementation; in this process, it was confirmed that the compression performance was improved for the selected quality metrics. Therefore, since the DL-based codecs need RGB input, the YUV, 4:2:0 8-bit depth video sequences were first converted to 10-bit depth as described above, and after the RGB frames (16-bit PNG format) were obtained using ffmpeg following the BT.709 recommendation. After DL-based coding, the decoded 10-bit RGB frames were converted to 10-bit YUV frames as recommended by the JVET NN CTC for quality assessment.

C. Performance Metrics

For the test quality metrics it was decided to adopt PSNR-Y as recommended by the JVET NN CTC [8] and VMAF [19]

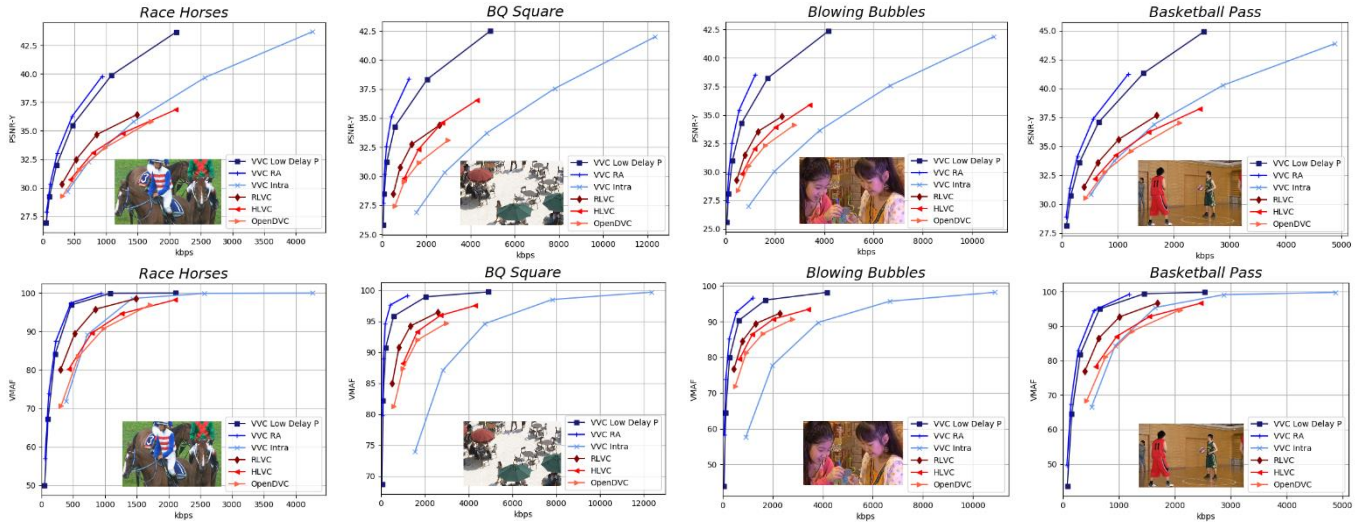

Figure 4: PSNR-Y (top) and VMAF (bottom) RD performance for the Class D video sequences.

due to its recent popularity for video quality assessment. The distortion metrics used for training are those referred in the previous section and selected by the respective authors since no retraining has been performed. As usual, the rate is measured in bits per second. For both metrics, BD-Rate values will be presented to express the rate savings/losses regarding the selected reference codec, in this case the VVC Low Delay P configuration; this choice was motivated by the need to have a good overlapping of rates and qualities among the assessed codecs to obtain reliable BD-Rate values.

As requested by the JVET NN CTC, all quality metrics computation happened at 10-bit and thus all video sequences originally at 8-bit depth were converted to 10-bit depth for quality metrics computation; remind that all coding is performed at 10-bit depth.

IV. Performance Results and Analysis

This section presents the RD performance results and BD-Rates for the test conditions defined in the previous section.

A. RD Performance by Codec and Sequence

For a better visualization of the codecs' RD performance, this section includes the RD performance charts for the Class D test sequences, both for the PSNR-Y and VMAF quality metrics, see Figure 4. These charts allow to make some first qualitative observations, notably: i) VVC Inter codecs (excluding VVC Intra) are clearly still the best in RD performance; ii) the VVC Inter to DL-based codecs quality gap seems to be shorter for the VMAF metric; and iii) the quality growth with rate is much faster for the VMAF metric. The RD performance gap for VMAF is much shorter than for PSNR-Y what is a great sign considering the better VMAF correlation with perceptual quality.

B. Overall BD-Rate Performance

Table 2 and Table 3 include the PSNR-Y and VMAF BD-Rates respectively, using as anchor the VVC Low Delay P configuration. It is important to note that the RLVC BD-Rate results in [15] are rather different from those presented here for three main reasons: i) the quality metric in [15] is an uncommon PSNR-RGB based on the pixel-level average MSE for the R, G and B components and not the common PSNR-Y or PSNR-YUV metrics; ii) in [15], RLVC is compared with theHEVC standard with the coding performed with the x.265 software (and not with the VVC standard using the VVC reference software as in this paper); and iii) the x.265 software is run at its 'LDP VERY FAST' configuration what may bring some performance penalty to reduce the complexity. These differences clearly demonstrate why a solid benchmarking needs clear, well-defined conditions, metrics, and pipelines as in this paper. The RD performance results in the tables allow to derive the following conclusions:

- The best performing DL-based codec is the RLVC, followed by the HLVC and the OpenDVC. The RLVC modelling of temporal dependencies using recurrent networks (e.g., ConvLSTM) brings performance benefits, due to its capability of making efficient predictions and tracking long-term dependencies, i.e., not only considering data of the past decoded frame.
- All DL-based codecs perform much better than VVC Intra but much worse than VVC RA. The best performing DL-based codec, RLVC, is outperformed by the VVC Low Delay P configuration performance (remind they both use Low Delay P settings). This means that DL-based tools exploiting the temporal correlation in a more efficient way are still needed.
- For the DL-based codecs, the VMAF BD-Rate is smaller than the corresponding PSNR-Y BD-Rate. Since VMAF is better correlated with human perception than the PSNR-Y mathematical fidelity measure, it means lower rate losses can be achieved if perception is considered. This is expected as DL-based codecs usually perform better for perceptual quality metrics where mathematical fidelity is not the target.

In summary, while the DL-based codecs do not yet perform at the same level of VVC, their RD performance is very promising considering these codecs are just emerging, while VVC has had decades of research efforts behind it.

Table 2: PSNR-Y BD-Rate (%): VVC Low Delay P as reference.

		VVC Intra	VVC RA	RLVC	HLVC	Open DVC
Class B	*MarketPlace*	370,81	-31,41	230,73	310,29	384,78
	RitualDance	245,68	-21,72	186,73	258,01	295,17
	Cactus	571,07	-22,57	333,68	467,64	546,12
	BasketballDrive	166,29	-24,83	258,84	320,54	425,50
	BQTerrace	493,17	-23,91	533,63	549,76	856,76
	Average	**369,40**	**-24,89**	**308,72**	**381,25**	**501,66**
Class C	*RaceHorses*	136,6	-18,2	224,4	297,8	348,3
	BQMall	447,2	-28,7	189,1	302,4	365,1
	PartyScene	509,2	-25,8	212,3	367,5	400,7
	BasketballDrill	413,4	-17,2	230,3	312,1	398,2
	Average	**376,6**	**-22,5**	**214,0**	**320,0**	**378,0**
Cla	*Basketball Pass*	160,6	-21,1	110,5	175,4	206,1
	BQSquare	762,0	-37,6	345,4	401,9	725,2

s s D	*Blowing Bubbles*	515,8	-36,5	153,0	232,9	300,2
	RaceHorses	170,9	-36,7	120,9	198,2	212,6
	Average	**402,3**	**-33,0**	**182,5**	**252,1**	**361,0**

Table 3: VMAF BD-Rate (%): VVC Low Delay P as reference.

		VVC Intra	VVC RA	RLVC	HLVC	Open DVC
Class B	*MarketPlace*	-	-28,42	197,33	259,56	383,42
	RitualDance	-	-19,63	163,85	259,56	285,45
	Cactus	-	-14,79	460,26	639,97	659,85
	Basketball Drive	-	-26,83	218,39	317,82	401,50
	BQTerrace	-	22,38	744,90	704,39	1 111,81
	Average	**-**	**-13,46**	**356,95**	**436,26**	**568,41**
Class C	*RaceHorses*	1 767,9	-21.12	170,1	301,4	318,5
	BQMall	1 114,4	-18,7	135,3	246,3	287,1
	PartyScene	794,3	-18,0	176,2	299,0	331,7
	Basketball Drill	458,8	-6,9	186,4	287,9	318,3
	Average	**1 033,9**	**-14,5**	**167,0**	**283,6**	**313,9**
Class D	*Basketball Pass*	213,8	-19,5	78,2	156,9	184,6
	BQSquare	1 603,5	-35,3	340,1	452,3	674,1
	Blowing Bubbles	699,2	-32,9	131,3	184,0	263,0
	RaceHorses	519,7	-14,0	93,0	185,8	210,9
	Average	**759,0**	**-25,4**	**160,7**	**244,8**	**333,2**

V. RLVC Training and Performance Validation

Considering the achieved results in Section IV, the DL-based video codec chosen to be improved was the RLVC, due to being the codec that achieved the best RD performance during the extensive benchmarking. Thus, this section will present the methodology used to train the RLVC models [15] (reviewed in Section II) with the key target to replicate, with the new trained RLVC models, the performance results obtained with the trained models made available by the RLVC authors. These models will be referred as 'Newly Trained RLVC' models, and the associated performance replication and validation of these models is essential to guarantee that the RLVC can be trained as intended, to then proceed with its improvement with these obtained results as a foundation for comparison. Given the complexity and time constraints, the RLVC training and performance replication have only been performed for the PSNR-RGB metric.

A. Training Conditions and Dataset

As mentioned earlier, the RLVC has six main modules that require training: 1) Motion estimation network [16]; 2) Motion RAE; 3) Motion compensation network, which has two main modules: the warping network, followed by the artifact smoothing network; 4) Residue RAE; 5) Motion RPM; and 6) Residue RPM.

The dataset used for training is the Vimeo-90k septuplet [20], which consists of 91701 video sequences, each with seven frames, with a 448×256 spatial resolution. Each time the frames are loaded for training, they are randomly cropped down to a 256×256 spatial resolution. Since MSE-RGB will be the metric used for distortion, the first frame of every training sequence will be Intra coded with the HEVC-based BPG codec, so that the training process can use the already decoded Intra frame as anchor, in the same way as when performing the coding process. The pre-trained models for the motion estimation network are used, but these weights will continue to be updated during the end-to-end training.

B. RLVC Training Phases

As mentioned in Section II, the RLVC model is trained progressively with several different loss functions and neural networks. The RLVC training strategy proposed by the authors includes three phases as follows:

Phase 1 – Frame by Frame Training

The first training phase employs three different loss functions and a progressive involvement of all the RLVC NN modules with the exception of the RPMs (for entropy coding). The distinct characteristic of this first training phase is that its loss functions only use sequences of two frames (frame pairs) to train the networks, thus the naming "Frame by Frame"; however, the ConvLSTM cell and hidden states are passed from one training iteration to the next, so the recurrency is still used in this training stage.

Every loss function in this phase is a rate-distortion loss function with the form $\lambda D + R$, where the λ dictates the final quality and rate trade-off achieved for the video frames. The first loss function considers only the MSE-RGB between the original frame and the frame obtained after the warping network and the bitrate produced with the motion RAE. The second loss function considers the MSE-RGB between the original frame and the frame obtained after the artifact smoothing network, and the bitrate produced by the motion RAE. The third and last loss function considers the MSE-RGB between the original frame and the final decoded frame (obtained after the addition of the residual processed by the residue RAE), and the bitrate produced by the motion and residue RAEs.

This first training phase has 700 000 iterations, and the first loss function is used until iteration 20 000, the second loss function from iteration 20 000 to 40 000, and the third loss functions from iteration 40 000 until the end of training. At each training iteration, a pair of frames is processed, and the second frame is encoded and decoded with the RLVC architecture, based on the previous decoded frame and the ConvLSTM states, if there are any (for the first pair of frames from a sequence the ConvLSTM states are null); the distortion and bitrate obtained for the second frame are used in the loss function. For the first 100 000 iterations, only the first pair of frames from four random training sequences is loaded into a batch, and as such, the ConvLSTM states are re-initialized at every iteration, and a new batch is also loaded at every iteration. After iteration 100 000, each batch consists of four 7-frame sequences, and as such a new batch and the ConvLSTM states are only re-initialized after 6 iterations, since each iteration processes a pair of frames from the training sequences and then the next iteration processes the next pair.

Phase 2 – Frame Sequence Training

Since all the neural networks involved in the generation of the latent representations (both RAEs, motion estimation, and motion compensation networks) are initialized and pre-trained in Phase 1, Phase 2 performs the fine-tuning of these models using all seven frames of the training sequences. The loss function considers the cumulative MSE-RGB between the original frame and decoded frame for all frames in the training sequence (except the first Intra-coded frame), and the cumulative bitrate obtained for the coding of the whole sequence.

This training phase only has 150 000 iterations, but at each iteration the whole training sequence (7 frames) which is loaded in the batch is processed, and the cumulative distortion and bitrate are used in the loss function. The ConvLSTM states are initialized with each new training iteration, and a new batch of sequences is also loaded.

Phase 3 – RPM Training:

Lastly, the RPM networks are trained, namely the motion and residue RPMs. The RPM training uses the already established models for the remaining networks (obtained at the end of Phase 2) and trains only the RPMs, maintaining all the other models fixed. The two RPMs are trained separately in this phase to allow for faster and less complex training for the rest of the neural network modules in Phases 1 and 2. Since the remaining network weights are not updated in this phase, the RPM training is rather quick.

The RPM training consists of only 20 000 iterations, and the loss functions are only rate-based functions, and consider only the bitrate produced by the motion RPM or the residue RPM, since they are trained separately.

C. Training Analysis

This subsection will analyze the three training phases defined earlier. All models were trained with a NVIDIA Tesla V100 GPU, with 32GB of memory.

Phase 1 – Frame by Frame Training

With the used GPU, this training phase takes approximately 150 hours (6 days and 6 hours) to train each model. The evolution of the last loss function used in Phase 1 can be observed in Figure 5.

As previously described, Phase 1 of training uses three different loss functions in succession, therefore until iteration 40 000 the training is not being optimized for the loss function displayed in Figure 5. From iteration 40 000, a sharp drop can be seen, as expected; however, after iteration 100 000, there is a slight deterioration of the loss function values, which may be explained by the change in the number of frames loaded into each batch, which after iteration 100 000 starts using the whole 7-frame sequences, instead of only the first pair from each sequence, therefore, the RLVC starts leveraging the use of the ConvLSTM layer in this part of the training.

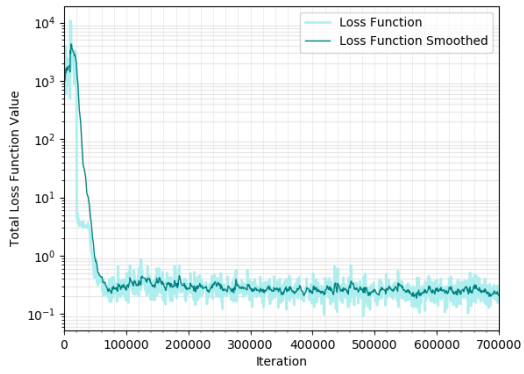

Figure 5: Evolution of the third loss function used in Phase 1 of training.

Phase 2 – Frame Sequence Training

With the used GPU, the second training phase takes roughly 112 hours (4 days and 16 hours) to train each model. Figure 6 shows the evolution of the loss function used in Phase 2 of training.

This phase of training is rather different from Phase 1, since it performs fine-tuning and, as such, the progress to be made is smaller and it is harder to optimize the loss function. In this case, the smoothed curve of the loss function provides a more straightforward insight into the training trend while it makes it easier to perceive the clear downward trend in the loss function. The values of this loss function have large variations, as observed in Figure 6, but it is important to note that this loss function begins with a much smaller value than that of Phase 1, so every variation is more noticeable than in the previous

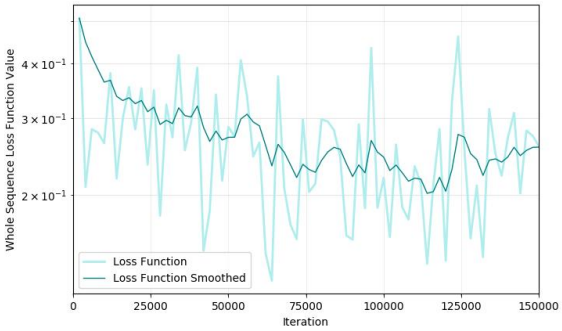

Figure 6: Evolution of the loss function used in Phase 2 of training.

training phase. Also, since this loss function considers the cumulative distortion and rate, any variation in the network weights that causes a negative progress will have much more impact in the loss function of Phase 2 than it would in the loss function of Phase 1, which considers only the distortion and rate for one frame at a time.

Phase 3 – RPM Training

The last training phase is rather fast, relatively to the previous training phases, since the models being trained in this phase, the RPMs, are smaller than the set of networks trained in the earlier phases. This phase of training takes grossly 24 hours, for each RPM of each model. Figure 7 and Figure 8 show the evolution of both loss functions used for the training of each of the RPMs (motion and residue, respectively). This training phase progresses rather quickly and with a clear loss function trend, given that it only considers the bitrate, and the only weights to update are those for the RPMs, which means there are less weights to set, and therefore less elements to provoke instability.

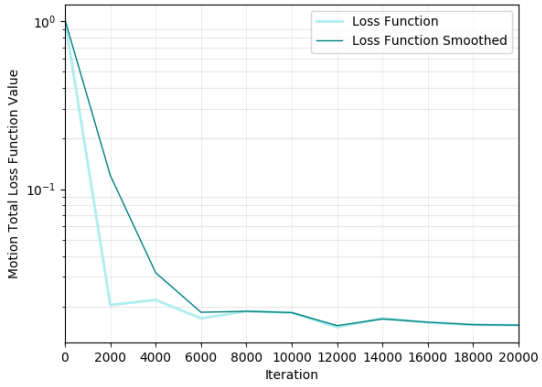

Figure 7: Evolution of Phase 3 loss function to train the motion RPM.

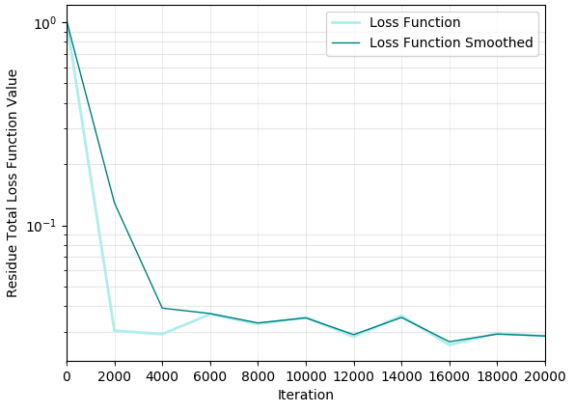

Figure 8: Evolution of Phase 3 loss function to train the residue RPM.

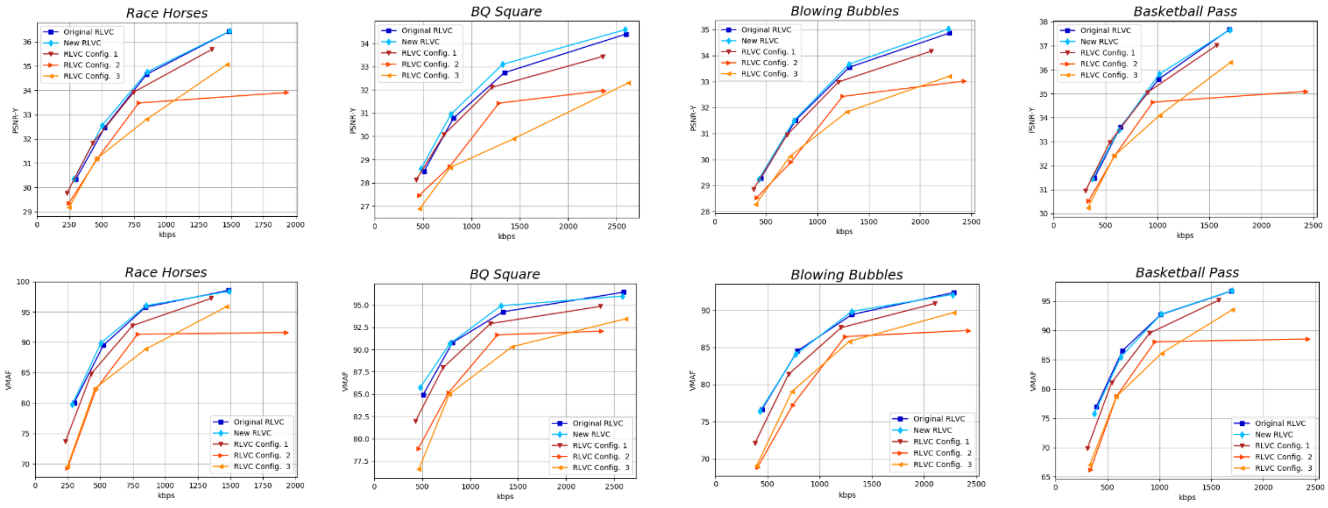

Figure 9: PSNR-Y (top) and VMAF (bottom) RD performance for the Class D video sequences, with the Newly Trained RLVC models.

D. Performance Validation

After completing all the training phases described previously, the Newly Trained RLVC models were used to meet the key objective of this section: to replicate the RD performance obtained with the pre-trained RLVC models provided by the RLVC authors and, thus, to validate the Newly Trained RLVC models as a sound starting point for the improvement of the RLVC framework (and software). The Newly Trained RLVC models were used to obtain RD performance results for the RLVC codec, notably for the Class C and D JVET video sequences, under the same testing conditions as used for the RD performance results obtained with the original, pre-trained RLVC models, which have been reported in Section IV. Additionally, other RLVC configurations for the training phases were studied, with the purpose of determining if any time costly training phase could be skipped or shortened to less iterations with the goal to reduce the overall (large) training time.

Besides the 'Original RVLC Models', the RLVC training configurations studied were:

- **Newly Trained RLVC:** Trained under the same precise conditions as suggested by the RLVC authors [15] to very closely replicate the original RD performance results.
- **RLVC Configuration 1 – Phase 1 RAEs:** For this configuration, Phase 2 is completely skipped. All neural networks (except the RPMs) are only trained with Phase 1, and then the RPMs are trained from the networks obtained at the end of Phase 1.
- **RLVC Configuration 2 – Phase 1 RAEs Shortened:** This configuration is similar to Configuration 1, but the training is cut short at iteration 300 000 and the RPMs are trained from the models obtained at this iteration.
- **RLVC Configuration 2 – Phase 1 RAEs Shortened:** This configuration is similar to Configuration 2, but the training decreases even more, and only trains until iteration 200 000 and the RPMs are trained from the models obtained at this iteration.

Figure 9 shows the PSNR-Y and VMAF RD performance for the Class D JVET videos, for the four target qualities using the PSNR-RGB metric as the training quality metric for the following configurations: i) Original RLVC Models (in dark blue); ii) Newly Trained RLVC Models (in light blue); iii) RLVC Configuration 1 (in dark red); iv) RLVC Configuration 2 (in orange); and v) RLVC Configuration 3 (in yellow). From the results in Figure 9, it is possible to observe:

- **Newly Trained RLVC models versus Original RLVC models:** The Newly Trained RLVC models can achieve a similar, and even better, RD performance as the Original RLVC models provided by the authors. Even though the training followed the recommendations by the RLVC authors, there are several aspects that may account for the slight differences in performance obtained. Since the neural networks weights are initialized randomly, the starting point for the training of the Newly Trained RLVC models may be different than the starting point of the pre-trained RLVC models; therefore, both models can be optimizing towards rather different minima, which may lead to rather different models and, therefore, slightly different, and in this case better, RD performance.
- **PSNR-Y versus VMAF:** For the VMAF metric, there is a clear decrease in quality when Phase 2 of training is skipped, which is not so evident with the PSNR-Y metric. For the VMAF metric, there is also a smaller gap in quality between the results obtained with the Newly Trained RLVC models and the RLVC Configuration 3 than there is for the PSNR-Y quality metric.
- **New RLVC models versus Configuration 1 models:** Comparing the Newly Trained RLVC models with Configuration 1 models, a slight RD performance variation may be noted. With the VMAF quality metric, all sequences show a faintly worse RD performance when Phase 2 is skipped. For the PSNR-Y metric, the results show that the Configuration 1 RD performance is similar to the Newly Trained RLVC models for the lower qualities, but worse for the higher qualities. For the lower qualities, the Configuration 1 results have worse quality but smaller bitrate compared with the Newly Trained RLVC models, which ultimately makes the curves align in this part of the chart; however, for the highest qualities of the RLVC Configuration 1, the PSNR-Y starts to worsen, thus creating a gap between the two curves.
- **Configuration 1 models versus Configuration 2 and 3 models:** Both training Configurations 2 and 3 are proven to always be worse than the remaining configurations, which implies that the training should, at least, not be stopped too early in Phase 1. From Configuration 2 to Configuration 3, a slight decrease in RD performance can be noted, which is

due to the fewer 100 000 iterations that Configuration 3 performs in Phase 1.

In summary, it may be concluded that it is possible to appropriately train the RLVC models, i.e., the 'Newly Trained RLVC Models', and thus to 'control' the RLVC framework in training and testing. This implies that a reliable starting point and benchmark for the next potential improvements to the RLVC architecture is available. Moreover, the study of simpler training configurations for the RLVC model allows concluding that skipping and/or shortening the original training phases is not advisable, as the RLVC RD performance would be penalized.

VI. RLVC WITH ATTENTION MODEL: A PRELIMINARY STUDY

After careful thought on the best way to extend the RLVC architecture to improve its RD performance, the tool that seemed most worthwhile were the attention models.

Attention models were firstly devised and used to improve RNN-based architectures [2], which deal with temporal data, specifically containing gated units similar to LSTM cells; thus it is known that attention models and RNNs work well together, and the RLVC is an RNN-based architecture. Attention models have also been incorporated into architectures with RNNs to solve computer vision tasks and have shown improvements [21], thus indicating that attention models may also be of use for tasks involving data such as images and video. As such, the idea was integrating an attention model into the RLVC, since it is an RNN-based model, which is where the attention models first started being used. Since, overtime, several different types of attention models have been designed and implemented, the attention model chosen to extend the RLVC architecture will be motivated and described.

This section will firstly describe the attention model used to extend the RLVC architecture [15], with the goal of improving the codec's RD performance. Then, the incorporation of the attention model in the RLVC architecture and the associated training process will be presented, and lastly the preliminary RD performance of the extended RLVC (X-RLVC) architecture will be examined and compared to the Newly Trained RLVC model RD performance achieved in Section V. Because DL-based video coding research is very computationally intensive, all experiments, especially the training, take a significant amount of time. This has prevented to go in this paper beyond a preliminary study of RVLC with attention models.

A. Selected Attention Model

The incorporated attention model was the so-called Convolutional Block Attention Module (CBAM) [22]. This attention model is simple but effective and was designed to be easily integrated into any CNN architecture. The CBAM was created and tested to improve architectures that perform image classification, and has shown to offer competitive results [22]. These capabilities may be also important in aiding the RLVC framework to learn which inputs it should dedicate more attention in order to achieve a better RD performance. In practice, an attention model is applied to an input to identify which of its parts should be given more emphasis, i.e., attention, which translates into multiplying those inputs by higher weights, and the less important inputs by lower weights, so their values can stand out if they should be given more attention, or be drowned out if they are less significant.

Figure 10 shows the architecture of the selected CBAM model. This attention model considers two different types of attention: *channel* wise attention and *spatial* wise attention. The CBAM takes the input tensor, which is three-dimensional (an image in RBG channels, or a feature map of an image already passed through some convolutional layers, which will add depth), and processes the set of input features along the various tensor channels (channel attention module) first and then processes the features along the tensor channel axis (spatial attention module). Both attention modules are characterized by a set of weights that are applied to the input, by multiplication, thus embedding into each part of the input its associated significance.

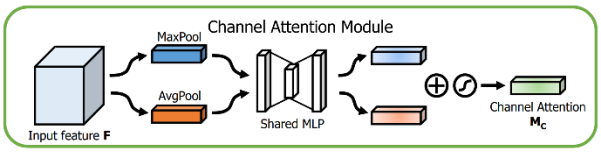

Figure 10: CBAM architecture [22].

B. RLVC Extended Architecture

For the modification of the RLVC architecture, the CBAM module was placed in the motion and residue RAEs. The RAEs perform the most important operations in the codec, since they encode and decode the motion and residue information, defining the final quality, while taking into consideration information from past frames in the video sequence. Thus, by inserting the attention model in the RAEs, the encoder-decoder architecture should receive additional assistance in detecting which information should be considered most important, either to encode, or to pass along to the next pair of processed frames, through the ConvLSTM layer states. To study the different impacts of the CBAM module in the RLVC codec, two different positions for the model, within the RAEs, were studied. Other positions could also have been studied if time had allowed.

- **Position 1 – Before Quantization:** In this case, the CBAM module is placed immediately before the quantization of the latent representations, i.e., after all the RAE layers, see Figure 11.
- **Position 2 – Before ConvLSTM:** In this case, the CBAM module is placed before the ConvLSTM layer at the encoder side of the RAE, see Figure 12.

Figure 11: Modified RAE architecture with CBAM in Position 1.

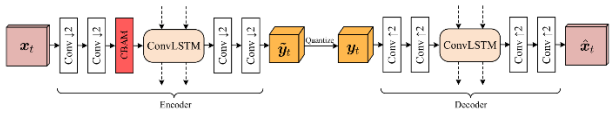

Figure 12: Modified RAE architecture with CBAM in Position 2.

The remaining RLVC architecture, notably the motion estimation and motion estimation networks, and both RPM networks, was not changed at all.

C. Training Process

The training of the X-RLVC model will basically include the same phases described in Section V for the original RLVC model, with the eventual addition of some new phases associated to the CBAM training. Since the Newly Trained RLVC models from Section V are available, they will be used in some training configurations as starting models for the

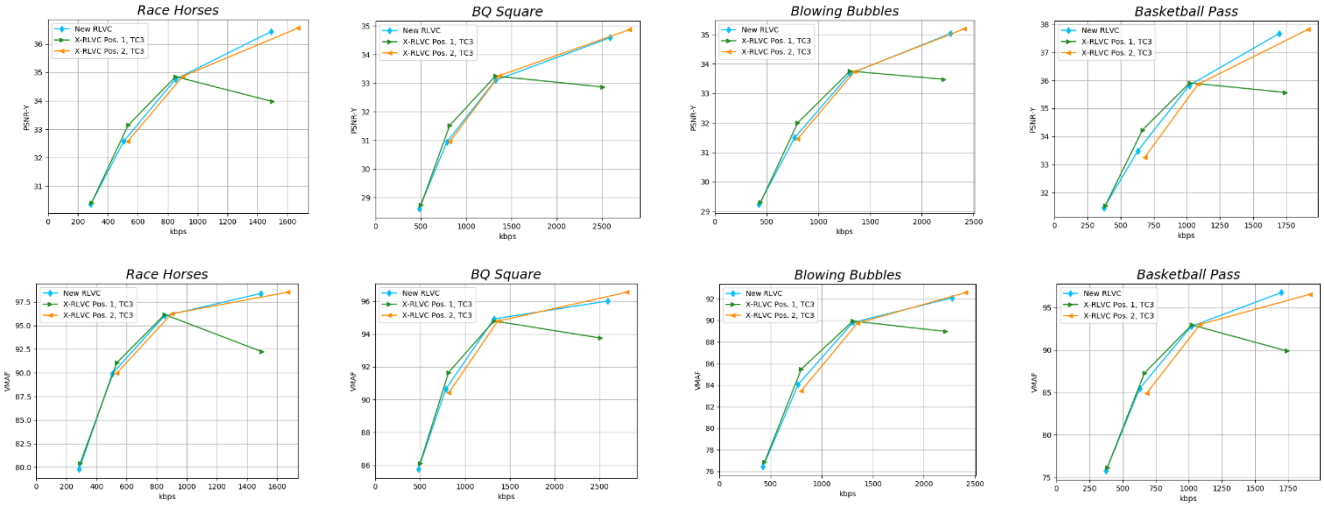

Figure 13: PSNR-Y (top) and VMAF (bottom) RD performance for the Class D video sequences, with the Newly Trained RLVC models and the X-RLVC models.

additional training of the X-RLVC with CBAM; this means in practice that the X-RLVC models may refine and extend the previously available RLVC models to avoid starting the training from the beginning. While there are many interesting training configurations, the initially chosen configurations to train both positions of the CBAM were:

- **Training Configuration 1 (TC1) – Square One with Phase 1:** The X-RLVC is trained from the start, following exactly the same training as described in Section V.
- **Training Configuration 2 (TC2) – Phase 1 Refinement:** The X-RLVC performs Phase 1 training, but only using the third loss function, and the weights obtained at the end of Phase 1 training from Section V are used to initialize all models of the X-RLVC at the beginning of training, except the CBAM which is initialized randomly. The RPM networks are trained from the models obtained at the end of this refinement.
- **Training Configuration 3 (TC3) – Phase 2 Refinement:** In this configuration the X-RLVC performs Phase 2 training, and the X-RLVC models are initialized from the RLVC models obtained at the end of Phase 2 in Section V, except for the CBAM which is initialized randomly. The RPM networks are trained from the models obtained after the refinement.

D. Performance Results and Analysis

Figure 13 shows the obtained preliminary RD performance results for the Class D test videos, for both PSNR-Y and VMAF quality metrics. From the preliminary results shown in Figure 13, it is possible to observe that there is a quality deterioration problem for the highest quality model (λ=2048), when the CBAM is in Position 1 of the X-RLVC, which sometimes achieves even worse quality than that obtained with the model for λ=512. When the loss function produced by the training of this model is examined, it can be observed that it did not progress as expected, and that the loss function evolution is quite erratic and unstable, even more so than the training example shown in Section V. To avoid this quality deterioration problem, several options were considered, and some implemented:

- **Smaller Learning Rate:** The learning rate used for the training of the highest quality model was reduced by 10, compared to the learning rate used for the remaining qualities, with the objective of making the network training take smaller steps, and thus, oppose the erratic evolution of the loss function.
- **Residual Connection:** Similarly to what is performed in [23], a residual connection is applied around the CBAM. The residual connection should allow the gradients to flow backwards through the network while skipping the CBAM layers, and thus enable deeper networks to be trained without the risk of vanishing or exploding gradients.
- **Initialization from λ=1024 model, with CBAM:** Instead of initiating the training from the RLVC model for λ=2048, obtained in Section V, training can be initiated from the model obtained after training the X-RLVC for λ=1024, i.e., the previous RD point, where the quality deterioration problem does not exist. This strategy was not implemented.

Despite the strategies listed above, the training of X-RLVC with the CBAM in Position 1 and forhyperparameter λ=2048 was not successful, and the quality degradation can still be observed. Examining the remaining preliminary RD performance curves presented in Figure 13, the following conclusions are reached:

- **RLVC versus X-RLVC:** Generally, neither version of the X-RLVC models with any training configuration provide better RD performance results over the Newly Trained RLVC models in a consistent way. The X-RLVC Position 1 shows improvement for some rate points, mainly λ=512, and in some cases for λ=1024, but shows worse RD performance for the highest quality model. The X-RLVC Position 2 has a consistently worse RD performance than the Newly Trained RLVC models.
- **X-RLVC Position 1 versus X-RLVC Position 2:** The X-RLVC Position 1 shows more promising RD performance improvements, but the X-RLVC Position 2 results are more consistent. X-RLVC in Position 1 shows consistent, even if minor, improvement over the New RLVC models for the λ values 512 and 1024. Also for this case, the X-RLVC model with λ=256 proves to have a similar performance as the New RLVC model for the same quality. However, the gains for these RD points do not make up for the poor performance achieved with the λ=2048 model (highest rate). For X-RLVC Position 2, the lowest quality model (λ=256) is not available, since during training it suffered from the exploding gradient problem, and there were no computing resources to initiate training again. The

remaining quality models achieved for the X-RLVC Position 2 do not show overall improvements over the New RLVC models, but for some sequences demonstrate a similar RD performance for the higher quality models. Overall, the X-RLVC Position 1 would be more promising if the quality degradation problem for the highest rate could be solved.

- **X-RLVC Position 1 Top Quality Problems:** It is difficult to obtain good results for the X-RLVC model with the hyperparameter λ=2048, when the CBAM is situated in Position 1, even after the strategies mentioned earlier were applied. The larger values of the loss function make the training of this model more unstable and, coupled with an attention model, the combination is prone to erratic and unpredictable training, which can lead to lower performance, as observed for the X-RLVC with the CBAM in Position 1, trained in Configuration 1.

From the obtained preliminary RD performance results with the X-RLVC models, it can be concluded that the insertion of the selected attention model in the RLVC architecture in the previously described positions and its subsequent training does not overall improve the RLVC performance. While some improvements happen for specific rate points, the gains are not consistent to claim an overall performance improvement.

VII. Final Remarks

DL-based tools are coming to the multimedia coding arena, notably targeting image and video coding. This paper offers the first solid performance study on DL-based video coding, using the JVET NN CTC, and including the most recent conventional coding benchmark, VVC. The main conclusion is that the RD performance gap is currently still large (although lower for perceptual quality metrics) and thus future improvements are still needed to reduce this gap as it occurred for DL-based image coding.

Finally, this paper offers some preliminary performance results for an extended RLVC solution, notably for several CBAM positions within the RLVC architecture, and for different training strategies. These preliminary experiments did not yet allow obtaining clear RD performance gains with the extended RLVC since many ideas and associated experiments could not be tested due to the time and computational constraints. However, attention model biased coding approaches are likely to succeed in the future and will be a hot topic of work in the associated research community in the near future since they have great potential to provide performance improvements.

In summary, the main objectives of this paper were to perform the benchmarking of the current DL-based video coding models against conventional coding technology, provided in Section IV, and to design, implement and train a DL-based video codec extended with an attention model, based on an already existing architecture from the literature, which would hopefully provide some RD performance improvement. While this second objective was not fully achieved, significant steps have been made in that direction.

References

[1] DeepAI, "Attention Models." [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/attention-models. [Accessed: 19-Oct-2021].

[2] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *3rd International Conference on Learning Representations*, San Diego, CA, USA, 2015.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 5998–6008.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, Virtual, 2021.

[5] S. Atito, M. Awais, and J. Kittler, "SiT: Self-supervised vIsion Transformer," *ArXiv Prepr. arXiv2104.03602*, 2021.

[6] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "ViViT: A Video Vision Transformer," *ArXiv Prepr. arXiv2103.15691*, 2021.

[7] B. Bross, J. Chen, J. R. Ohm, G. J. Sullivan, and Y. K. Wang, "Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC)," *Proc. IEEE*, vol. 109, no. 9, pp. 1463–1493, 2021.

[8] JVET, "Common Test Conditions and Evaluation Procedures for Neural Network-based Video Coding Technology," *Doc. JVET-U2016-r1*. 2021.

[9] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An End-to-End Deep Video Compression Framework," in *IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 11006–11015.

[10] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational Image Compression With a Scale Hyperprior," in *6th International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.

[11] R. Yang, L. Van Gool, and R. Timofte, "OpenDVC - An Open Source Implementation of the DVC Video Compression Method," *ArXiv Prepr. arXiv2006.15862*, 2020.

[12] R. Yang, L. Van Gool, and R. Timofte, "OpenDVC Software Implementation." [Online]. Available: https://github.com/RenYang-home/OpenDVC. [Accessed: 01-Mar-2021].

[13] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, "Learning for Video Compression With Hierarchical Quality and Recurrent Enhancement," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020, pp. 6627–6636.

[14] R. Yang, L. Van Gool, and R. Timofte, "HLVC Software Implementation." [Online]. Available: https://github.com/RenYang-home/HLVC. [Accessed: 01-Mar-2021].

[15] R. Yang, F. Menzter, L. Van Gool, and R. Timofte, "Learning for Video Compression With Recurrent Auto-Encoder and Recurrent Probability Model," *IEEE J. Sel. Top. Signal Process.*, vol. 15, no. 2, pp. 388–401, 2021.

[16] A. Ranjan and M. J. Black, "Optical Flow Estimation Using a Spatial Pyramid Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, USA, 2017, pp. 4161–4170.

[17] R. Yang, F. Menzter, L. Van Gool, and R. Timofte, "RLVC Software Implementation." [Online]. Available: https://github.com/RenYang-home/RLVC. [Accessed: 01-Mar-2021].

[18] JVET, "VVCSoftware_VTM." [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/releases#VTM-11.0. [Accessed: 01-Mar-2021].

[19] Z. Li, C. Bampis, J. Novak, A. Aaron, K. Swanson, A. Moorthy, and J. Cock, "VMAF: The Journey Continues," *Netflix Technol. Blog*, vol. 25, 2018.

[20] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video Enhancement With Task-oriented Flow," *Int. J. Comput. Vis.*, vol. 127, no. 8, pp. 1106–1125, 2019.

[21] F. Wang and D. M. J. Tax, "Survey on the Attention Based RNN Model and Its Applications in Computer Vision," *ArXiv Prepr. arXiv1601.06823*, vol. arXiv:1601, 2016.

[22] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," in *European Conference on Computer Vision*, Munich, Germany, 2018, pp. 3–19.

[23] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778.