Learning-based End-to-End Video Compression Using Predictive Coding

Matheus C. de Oliveira*, Luiz G. R. Martins*, Henrique Costa Jung*,
Nilson Donizete Guerin Jr*, Renam Castro da Silva†, Eduardo Peixoto*, Bruno Macchiavello*,
Edson M. Hung*, Vanessa Testoni†, and Pedro Garcia Freitas†
*University of Brasília, †Samsung R&D Brazil,

***Abstract*—Driven by the growing demand for video applications, deep learning techniques have become alternatives for implementing end-to-end encoders to achieve applicable compression rates. Conventional video codecs exploit both spatial and temporal correlation. However, due to some restrictions (e.g. computational complexity), they are commonly limited to linear transformations and translational motion estimation. Autoencoder models open up the way for exploiting predictive end-to-end video codecs without such limitations. This paper presents an entire learning-based video codec that exploits spatial and temporal correlations. The presented codec extends the idea of P-frame prediction presented in our previous work. The architecture adopted for I-frame coding is defined by a variational autoencoder with non-parametric entropy modeling. Besides an entropy model parameterized by a hyperprior, the inter-frame encoder architecture has two other independent networks, responsible for motion estimation and residue prediction. Experimental results indicate that some improvements still have to be incorporated into our codec to overcome the all-intra coding set up regarding the traditional algorithms High Efficiency Video Coding (HEVC) and Versatile Video Coding (VVC).**

I. Introduction

Video data takes over about 82% of the data transferred on worldwide networks and that proportion has been firmly growing further, as indicated in the last Cisco's annual Visual Networking Index [1]. Moreover, due to the pandemic, the world is currently experiencing rapid worldwide growth of private, public, and government entities that require employees to work from home. This increases significantly the use of video conference services [2]. Meanwhile, the introduction of novel video communication technologies such as Wider Color Gamut (WCG), High Dynamic Range (HDR), High Frame-Rate (HFR), Ultra-High Definition (UHD), and the forthcoming immersive video services have increased the challenge of transmitting video information. Therefore, efficient video compression has a fundamental role to enable these technologies, being always pressing and urgent.

Since the first proposal of hybrid coding of pictorial data [3], several video compression standards have been proposed. For instance, H.264 [4] (the most widely used video codec), H.265 [5], and H.266 [6] (the last video compression standard). All these codecs follow a hybrid coding framework that includes transform-based coding, residual-based coding, motion compensation (i.e., inter-prediction), and intra-prediction techniques. The algorithms that implement these techniques vary from one standard to another although they all rely on hand-crafted coding tools.

Over the last few decades, the development of these hand-crafted techniques has led the video coding performance to be improved around 50% every 10 years under the cost of additional computational complexity, memory, and energy consumption. For example, Seidel [7] has shown that there is a trend in the complexity increase: each novel video coding standard is about 10 times more complex than its predecessor while its coding efficiency increases only 2 times. Now, the video compression community has confronted novel challenges to further enhance the coding efficiency to attend to the strong requirements of future video communication applications. So, there exists a demand for new video coding schemes to explore new video compression directions, especially considering deep learning-based methods.

The study of deep learning-based video coding by leveraging the state-of-the-art video coding standard HEVC [5] has been an active area of research in recent years. Almost all the modules of HEVC (intra-prediction, inter-prediction, quantization, entropy coding, and loop filtering) have been explored and improved by incorporating various deep learning techniques. Reviews of some representative works about trained deep networks as tools within traditional coding schemes or together with traditional coding tools can be found in the field literature [8], [9]. From some evidence that deep learning can significantly improve modules of classical video compression frameworks [9], recent works [10]–[12] propose end-to-end solutions for video coding, meaning that all operations performed are derivable and the training of the different modules can be done jointly. Those solutions are designed for high compression efficiency and achieve this by exploiting spatial and temporal redundancies within and across video frames.

There are two central challenges to build an end-to-end video coding solution. First, it is very complex to design a single neural network that replaces the whole video compression system. The existing end-to-end learning-based approaches cannot exploit the entire information redundancy with the same efficiency of the existing hybrid coding scheme adopted in traditional video codecs [13]. For that reason, it is very desired to combine the advantages of both learning-based and hybrid frameworks in traditional video coding. Second, there is a need to create a strategy for generating and compressing the motion information that is intrinsic to video information.

Video coding methods thickly depend on motion information to decrease temporal redundancy. A straightforward solution is to employ a learning-based optical-flow algorithm to produce motion information.

In this paper, we propose an end-to-end deep video compression solution that combines the advantages of both neural networks and predictive coding. The proposed model is based on predictive coding as conventional video codecs. The proposed video codec is composed of a type of artificial neural network known as 'autoencoder' [14], an unsupervised artificial neural network that learns how to efficiently reduce the input and then learns how to reconstruct the input data from the reduced representation. The autoencoder is responsible for encoding the anchors (I-frames) and also the predicted frames (P-frames). The structure of the codec and the basis of its transformations are presented in Section III. Section IV covers the training procedures, while results are presented in Section V. Finally, conclusions are made in section VI.

II. A Brief Review on Learning-Based Coding

End-to-end image coding has risen for less than five years, inaugurating a novel field for lossy compression. The vast majority of the end-to-end learning-based methods follow an autoencoder-like [15] approach. Different variations of this approach were proposed by Toderici *et al.* [16]–[18] and Gregor *et al.* [19] to obtain a compact and discrete representation of images by applying quantization to the bottleneck layer of autoencoders. To control and enable variable bitrate, these models progressively analyze (encode) and synthesize (decode) residual errors with multiple autoencoders. Progressive codes are indispensable to Rate-Distortion Optimization (RDO) since better visual quality can be achieved with the increase of bitrate. Baig *et al.* [20] proposed an inpainting approach to enable a progressive analyzer that exploits spatial correlation presented by neighboring blocks to reduce redundancy in the image. Balle *et al.* [21] proposed an image compression framework consisting of a nonlinear analysis transformation (encoder), a uniform quantizer, and a nonlinear synthesis transformation (decoder) that, unlike the preceding works and under certain conditions, achieved a dramatic improvement in visual quality, exhibiting better rate-distortion performance than the JPEG and JPEG 2000 standards. Similarly, Theis *et al.* [22] changed the latent/bottleneck quantization with a smooth estimate with an incremental training strategy. The learning-based approach is still a hot topic in the image coding community and recent trends basically include the development of components of autoencoder architectures [23]–[25].

Unlike images, videos contain highly temporal redundancy between consecutive frames. This redundancy is exploited using frame interpolation [26] and frame extrapolation [27], [28]. Moreover, some works estimate the optical flow between frames [29]–[31]. On the other hand, Spatial Transformer Networks (STN) apply parametric transformations (i.e., zoom, rotate, and skew) to blocks of feature maps to spatially transform the feature maps to capture temporal correlation [32], [33]. Inspired by the aforementioned efforts and based on the successful research in learning-based image compression, we further explore a learning-based framework for video compression.

III. Proposed Low-delay AI based video codec

We adopted a Group of Pictures (GOP) organization for the proposed codec. Each GOP comprises a group of N consecutive frames. While it is usual that each GOP has the same number of frames, this is not a requirement in our codec – each GOP can have any number of frames. Also, each GOP is completely independent of previous GOPs – there is no dependency neither from reference pictures nor from entropy encoders. As such, each GOP can be encoded or decoded in parallel without any degradation. This creates a versatile structure that can be applied in several real applications. The video segmentation into GOP sets is depicted in Fig. 1.

As commonly defined in video coding, in the proposed codec, the first frame of a GOP is an intra frame. All other $N-1$ frames are encoded using inter-frame prediction (P-frames). At this point, we are using exclusively the previous frame as the reference. The arithmetic encoder is initialized at the start of each GOP and finalized at the end of the GOP. This GOP structure is depicted in Fig. 2.

The frames are encoded using a YUV $4{:}4{:}4$ format represented as floating-point variables in the range $[0, 1]$ per channel. Regardless of the input format used, the video is first transformed into this representation. After decoding (or when computing the reconstructed video in the encoder) the video is then transformed back to its original format.

A. Intra-Frame Encoding

The main architecture adopted for intra-frame coding is based on the work by Ballé et al. [21] shown in Fig. 3. The codec is optimized to minimize the following objective function:

$$L[g_a, g_s, p_{\tilde{\mathbf{y}}}] = -\mathbb{E}[\log_2 p_{\tilde{\mathbf{y}}}] + \lambda \mathbb{E}[d(\mathbf{x}, \hat{\mathbf{x}})]. \quad (1)$$

The first component of the above loss is the expectation of the information content, i.e., the entropy, whereas the second term is the expectation of the distortion, which is related to the reconstruction quality. The symbol λ controls the relevance of the reconstruction in the optimization – it works as the Lagrange multiplier of common rate-distortion optimization.

In the nomenclature used by the authors, and well-established in the literature, analysis transform is the encoder component g_a, which takes the input frame $\mathbf{x}$ into the reduced dimensional latent space $\mathbf{y}$. Synthesis transform is the decoder component g_s, which aims to produce a distribution from which the reconstruction $\hat{\mathbf{x}}$ can be gathered.

Kingma et al. [34] show that the formulated rate-distortion loss is equivalent to the Bayesian variational autoencoder loss function. From this result, the distribution of the intra-frame reconstruction, which is equivalent to the likelihood of the generative model, is represented by a Gaussian, whose mean is

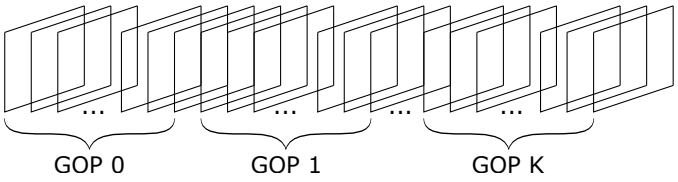


Fig. 1: Frame separation in GOP sets.

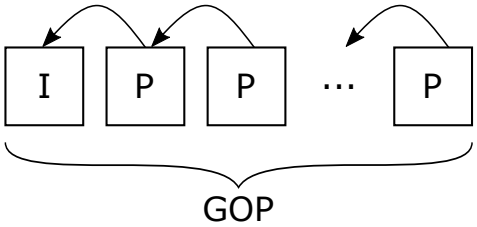


Fig. 2: GOP Structure. The arrows point to the reference frame.

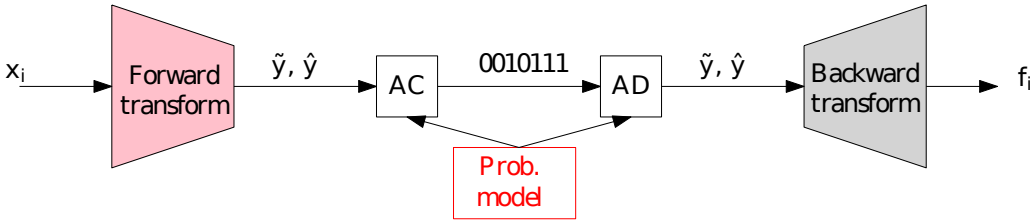


Fig. 3: Intra frame encoder architecture.

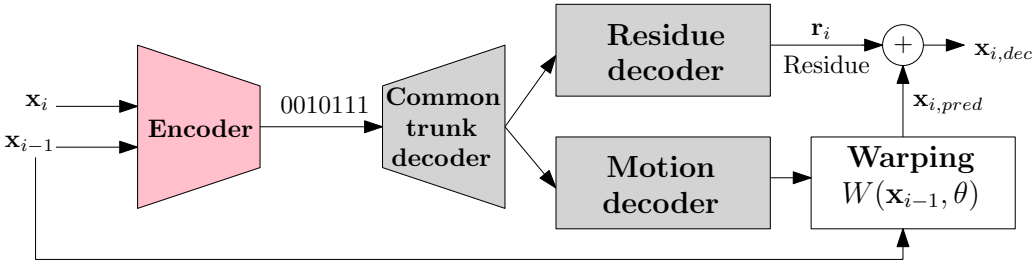


Fig. 4: Overall model architecture

defined by the synthesis transform, and the scale is controlled by the λ hyperparameter.

$$p_{\mathbf{x}|\tilde{\mathbf{y}}}(\mathbf{x}|\tilde{\mathbf{y}}) = \mathcal{N}(\mathbf{x}|g_s(\tilde{\mathbf{y}}), (2\lambda)^{-1}\mathbb{1}), \quad (2)$$

Also, the code (latent) distribution is modeled with a non-parametric model, with independent marginal components:

$$p_{\mathbf{y}|\boldsymbol{\psi}}(\psi^{(0)}, \psi^{(1)}, ...) = \prod_i p_{y_i|\psi^{(i)}}(\psi^{(i)}), \text{with } \mathbf{y} = g_a(\mathbf{x}) \quad (3)$$

with $\psi^{(i)}$ representing the parameters used to approximate the distribution. Latent quantization consists of a non-differentiable operation: $\hat{y}_i = round(y_i)$. In training time, this procedure is replaced by additive uniform noise. This approach yields the definition of the relaxed probability model of the code, denoted here as $\tilde{\mathbf{y}}$.

$$p_{\tilde{\mathbf{y}}|\boldsymbol{\psi}}(\tilde{\mathbf{y}}|\boldsymbol{\psi}) = \prod_i \left(p_{y_i|\psi^{(i)}}(\psi^{(i)}) * \mathcal{U}\left(-1/2, 1/2\right)\right)(\tilde{y}_i) \quad (4)$$

As the network converges based on Eq. 1, the authors in [21], [35] conclude that the continuous probability density presented in Eq. 4 results in an approximation for the quantization with unitary bin size, which is actually performed at test time. The fully-factorized density model adopted for the prior $p_{\tilde{\mathbf{y}}}$ supports the *non-parametric* nomenclature for the network used in intra encoder. Its architecture is detailed in Table I.

TABLE I: Encoder and decoder networks for the intra-frame codec. In our notation, C-9 $\times$ 9, $\downarrow$2, GDN, 256 indicates a convolutional layer with 256 filters, each with spatial support 9 $\times$ 9, stride 2, and the Generalized Divisive Normalization (GDN) as the activation function. Similarly, TC refers to transposed convolutional layer and IGDN to inverse GDN.

Encoder	**Decoder**
Forward transform	**Backward transform**
C-9 $\times$ 9, $\downarrow$2, GDN, 256	TC-9 $\times$ 9, $\uparrow$2, IGDN, 256
C-5 $\times$ 5, $\downarrow$2, GDN, 256	TC-5 $\times$ 5, $\uparrow$2, IGDN, 256
C-5 $\times$ 5, $\downarrow$2, 256	TC-5 $\times$ 5, $\uparrow$2, 256

The quantized coefficients are encoded using a simple arithmetic encoder with fixed tables for each coefficient. The tables are gathered and stored during training, and then used at runtime to encode the coefficients in a lossless fashion.

B. Inter-Frame Encoding

The architecture of our inter-frame encoder was initially presented in our previous work [36] and is shown in Fig. 4. It is comprised of stacked convolutional layers as detailed in Tables II and III. We have used 256 filters in the convolutional layers. As observed in Fig. 4, the autoencoder composed by the *Encoder* and the *Common Trunk Decoder* is the first transformation to which the joint input is submitted. Its architecture, proposed by Minnen et al. [37] as an extension

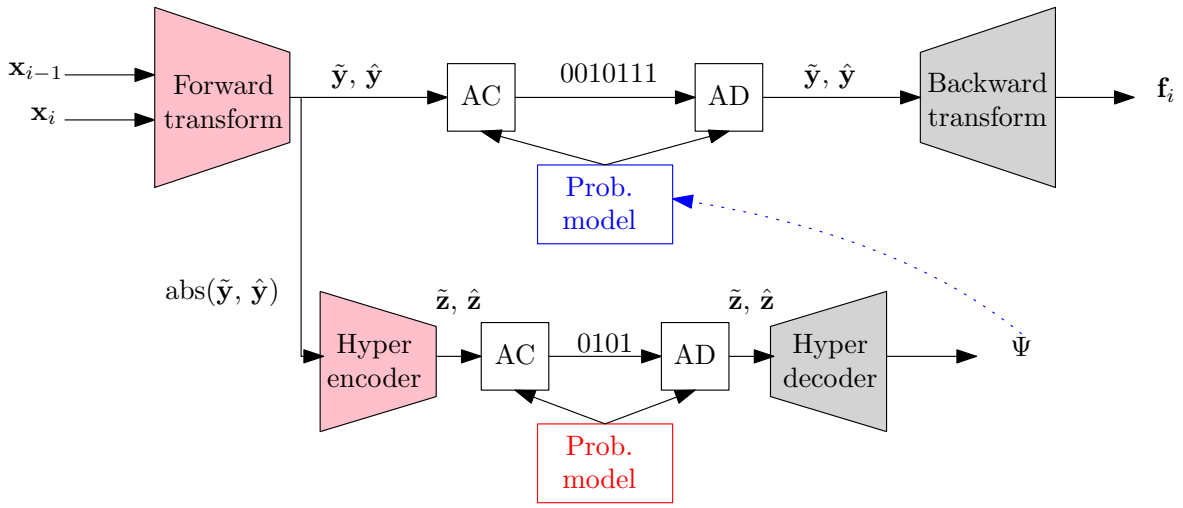


Fig. 5: Encoder/Common trunk decoder architecture. AC and AD stand for Arithmetic encoder and decoder, respectively.

TABLE II: Encoder and decoder networks for the inter-frame codec. The notation follows the same established in Table I. ReLU refers to the rectified linear unit activation function.

Encoder	**Common Trunk Decoder**
Forward transform	**Backward transform**
C-5 × 5, ↓2, GDN, 256	TC-5 × 5, ↑2, IGDN, 256
C-5 × 5, ↓2, GDN, 256	TC-5 × 5, ↑2, IGDN, 256
C-5 × 5, ↓2, GDN, 256	TC-5 × 5, ↑2, IGDN, 256
C-5 × 5, ↓2, 256	TC-5 × 5, ↑2, 256
Hyper encoder	**Hyper decoder**
C-3 × 3, ReLU, 256	TC-5 × 5, ↑2, ReLU, 256
C-5 × 5, ↓2, ReLU, 256	TC-5 × 5, ↑2, ReLU, 256
C-5 × 5, ↓2, 256	TC-3 × 3, 256

TABLE III: Motion decoder and residue decoder networks.

Motion decoder	**Residue decoder**
C-5 × 5, ReLU, 32	C-5 × 5, ReLU, 32
C-1 × 1, 2	C-1 × 1, 3

of the model introduced by Balle et al. [35], is illustrated in Fig. 5 and detailed in Table II. It is important to mention that our proposal uses image warping that is not limited to discrete spatial locations to perform motion compensation in inter-frame prediction.

In this modeling, it is assumed that each dimension $\tilde{y}_i$ of the main latent space follows a Gaussian with mean μ_i and standard deviation σ_i and is independent of the remaining ones. These parameters are obtained from a hyperprior modeled by the auxiliary variational autoencoder. For the sake of notation, the components of this additional autoencoder are represented by h. Therefore, the associated latent is denoted by $\mathbf{z} = h_a(\mathbf{y})$ and the hyper-synthesis is referred to as h_s.

As there is no prior beliefs, the modeling of $p_{\tilde{\mathbf{z}}}$ follows a non-parametric approach similar to our intra-encoder and equivalent to the one illustrated in Eq. 4. Since the hyperprior autoencoder is optimized to produce parameters $[\boldsymbol{\mu}, \boldsymbol{\sigma}] = h_s(\tilde{\mathbf{z}})$, the posterior $p_{\tilde{\mathbf{y}}|\tilde{\mathbf{z}}}$ in training time is given by Eq. 5.

$$p_{\tilde{y}|\tilde{z}}(\tilde{y}|\tilde{z}) = \prod_i \left(\mathcal{N}(\mu_i, \sigma_i^2) * \mathcal{U}\left(-1/2, 1/2\right)\right)(\tilde{y}_i). \quad (5)$$

Notice that by stacking optimized transformations according to the modeling presented in Eq. 4 (adapted for $\tilde{z}$) and 5, a single joint factorized posterior under variational analysis is obtained. The hyper-latent $\mathbf{z}$ is considered as side information and should also be compressed. Therefore, the loss espoused for inter-frame encoding is presented in Eq. 6 and it is an expanded version of the one shown in Eq. 1.

$$L[g_a, g_s, h_a, h_s, p_{\tilde{y}}, p_{\tilde{z}}] = -\mathbb{E}_1 - \mathbb{E}_2 + \lambda \mathbb{E}[d(z, \hat{z})], \quad (6)$$

where $\mathbb{E}_1 = \mathbb{E}[\log_2 p_{\tilde{\mathbf{y}}}]$ and $\mathbb{E}_2 = \mathbb{E}[\log_2 p_{\tilde{\mathbf{z}}}]$. It can be seen in Fig. 4 that the feature maps produced by the *Common Trunk Decoder* are directly submitted to the *Motion* and *Residue Decoders*. Therefore, the *Encoder* is jointly optimized to generate a latent which also contains information for motion estimation and for obtaining residue between frames.

The purpose of the Motion Decoder is to produce the parameters $\boldsymbol{\theta} = \{t_x, t_y\} \in \mathbb{R}^{2 \cdot w \cdot h}$ for a warping transformation $W(\cdot; \boldsymbol{\theta})$ over the reference frame $\mathbf{x}_{i-1}$, previously reconstructed by the non-parametric model, described in Section III-A. This transformation is performed by the Spatial Transform Network [32] and has been restricted to a bidirectional translation, which is replicated in all channels, resulting in $\mathbf{x}_{i,pred}$. Following [32], each pixel in the warped output $\mathbf{x}_{i,pred}$ is computed by applying a sampling kernel centered at a particular location (x_{i-1}, y_{i-1}) in the reference frame $\mathbf{x}_{i-1}$.

$$\begin{pmatrix} x_{i-1} \\ y_{i-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{pmatrix} \begin{pmatrix} x_i & y_i & 1 \end{pmatrix}^T \quad (7)$$

The *Residue Decoder* captures the details present in the $\mathbf{x}_i$ frame that could not be extracted from $\mathbf{x}_{i,pred}$. It is important

to point out that the unique input of the network dedicated to the residuals prediction is the *Common Trunk Decoder* feature maps. In this context, the block generates the novelties of the $\mathbf{x}_i$ frame with respect to $\mathbf{x}_{i,pred}$ without previously knowing the parameters $\boldsymbol{\theta}$ that had defined the warping transformation.

For the inter-frame coding, both the hyperprior and main latent quantized coefficients are encoded using arithmetic coding. The hyperprior coefficients are encoded in the same fashion as those obtained in intra-frame coding, using fixed tables for each coefficient derived during the training stage. The quantized coefficients of the main latent are encoded using a probability table derived from the variance estimated by the hyperprior, which strategy was suggested by Balle et al. [35]. Since this hyperprior is also transmitted to the receiver, the decoder can derive this probability table to perfectly decode the bitstream (i.e., the unique losses in these coefficients arise from the quantization process).

IV. Training Procedure

As described in Section III, the proposed codec is composed of two main networks: one to encode a single frame (*intra frame encoder*) and another one to encode a single frame using a given reference frame (*inter frame encoder*). Therefore, they are not trained together but in two stages, of which the first is dedicated to the intra encoder network and the second to the inter-encoder network. This training was performed on GPU using NVidia GeForce RTX 2080 Ti graphics cards.

1) Training the Intra Model: We create a new training dataset by combining the data available in the following datasets to train our intra-frame encoder: (i) CLIC Professional dataset [38] (628 images) ; (ii) CLIC Mobile dataset [38] (1111 images); (iii) DIV2K dataset [39] (902 images); (iv) Ultra-Eye Ultra HD dataset [40] (41 images); (v) MCL-JCI [41], [42] (51 images); and (vi) FLICKR2K dataset [43] (2650 images). The images in these datasets were converted to YUV 4:4:4 to unify the input format. For each image, we extract patches that consist of non-overlapping blocks of size 256×256 pixels. These patches were taken from a grid that is randomly displaced by an offset of eight pixels in each direction. This small random displacement is applied to avoid prior compression artifacts appear in the same positions in every patch. The full dataset consists of 88,529 image patches.

2) Training the Inter Model: To train the inter-frame codec, we need a pair of frames, of which one will be used as a reference and the other will actually be encoded. Naturally, the network is trained to optimize the reconstruction of this second frame only (denoted current or target frame). To train the network using a scenario closer to the testing one, the reference frame is first encoded using the already trained intra-frame codec and this decoded frame is concatenated with the following to feed the inter-frame network.

In order to have a dataset composed of sequential frames, we have gathered videos from the UGC Dataset [44], a large set made available from YouTube videos of different content. We have used videos coming from the following subsets: (i) Animation (22 videos); (ii) CoverSong (22 videos); (iii) Gaming (39 videos); (iv) HowTo (17 videos); (v) Lecture (24 videos); (vi) LiveMusic (21 videos); (vii) Lyrics (2 videos); (viii) MusicVideo (26 videos); (ix) NewsClip (17 videos); (x) Sports (33 videos); (xi) VerticalVideo (20 videos); and (xii) Vlog (32 videos). Only videos originally in 1920×1080 resolution (1080P) were used. The contents are very diverse with natural and synthetic (games and animation) images. Some natural videos have low movement content, such as News clips, as well as very high movement content, such as Sports videos. Unexpectedly, many videos from the Animation class had very little movement. In total, 275 videos were used. Patches with 256×256 dimensions are randomly cropped from a region in the video, and the same offset is used for all frames (i.e., the reference and target images are always co-located). Most of this content is in YUV 420 format, so we perform a color space transformation to YUV 444 and normalize it to the range $[0, 1]$.

V. Rate-Distortion Evaluation

We evaluated the rate-distortion relation using the JVET data set, specifically class D and F. The encoding parameters follow the recommendation from the draft test conditions defined for DNNVC ISO/IEC JTC1/SC29/WG11 activity. The videos were encoded with a 10 bit depth and the Intra Period (i.e., GOP size) is constant within a sequence and depends on the frame rate of the source. The video sequences used in our tests were:

- RaceHorses (300 frames, 30 fps, 416×240 pixels)
- BQSquare (600 frames, 60 fps, 416×240 pixels) with GOP size equals 64.
- BlowingBubbles (500 frames, 50 fps, 416×240 pixels) with GOP size equals 48.
- BasketballPass (500 frames, 50 fps, 416×240 pixels)
- BasketballDrillText (500 frames, 50 fps, 832×480 pixels) with GOP size equals 48.

Fig. 6 depicts the results for Y-PSNR, Y-MSSSIM, and Y-SSIM for the abovementioned sequences. These plots show the RD curves of our codec, HEVC, and VVC codecs, all in low-delay mode, using the same intra period. The results of the three codecs in all intra configurations were also included. From the frames-per-second rate required by the sequences, the abscissa axis has the unit *kbps* instead of the unit *bpp*, which the latter is adopted to evaluate the image encoders.

From these plots, we can notice that VVC and HEVC in low-delay (with inter configuration) consistently outperformed our codec. Our codec has a performance more competitive when compared with the HEVC and VVC both in all-intra profile, especially when evaluated by subjective metrics. Fig. 7 and 8 provide a visual comparison between frames in their original forms, encoded by a traditional algorithm and encoded by one of our proposed codecs.

Some features can be observed when analyzing RD curves referring to neural codecs. For example, the non-monotonicity of a curve derives from the fact that each of the points is composed of two independently and stochastically optimized

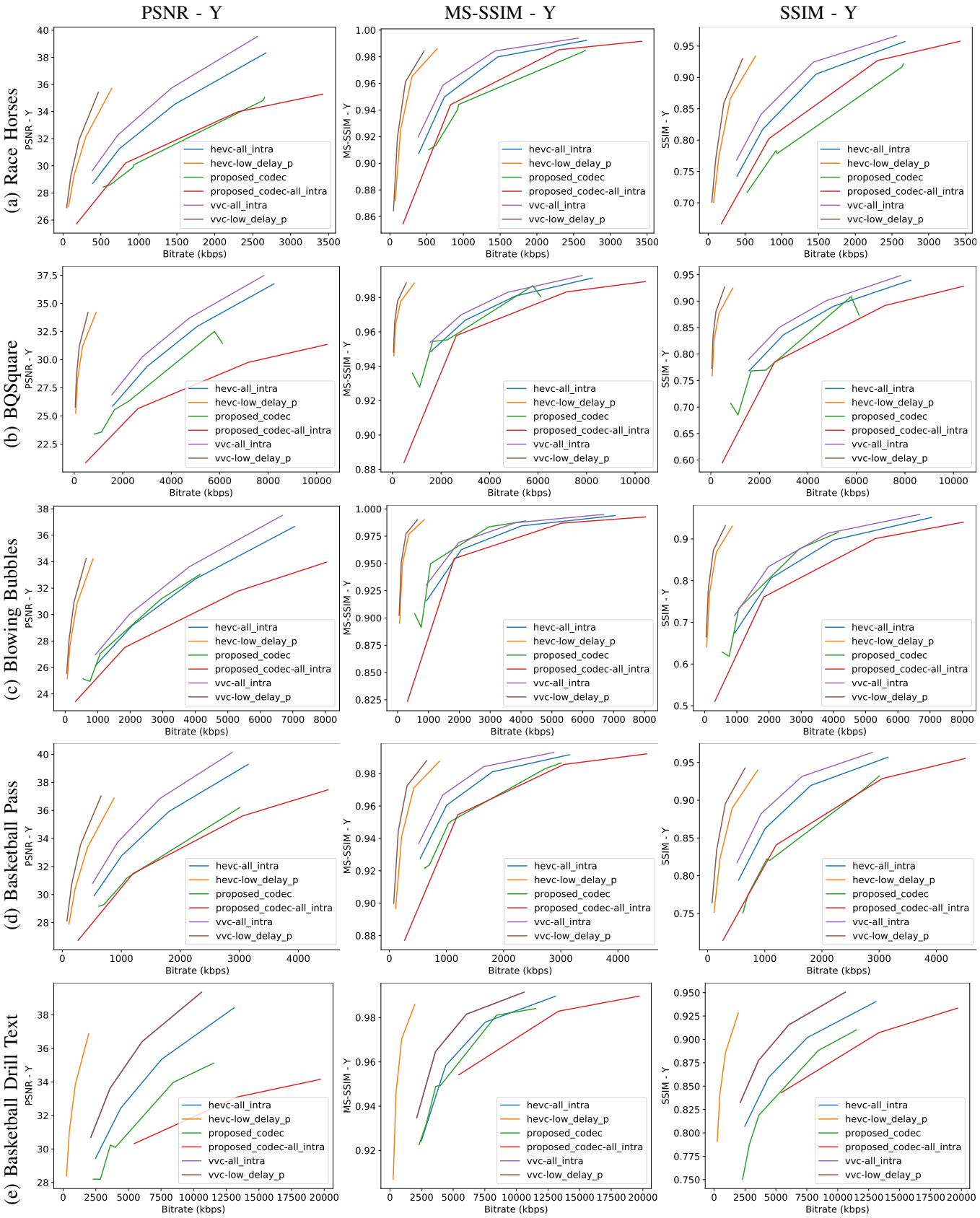
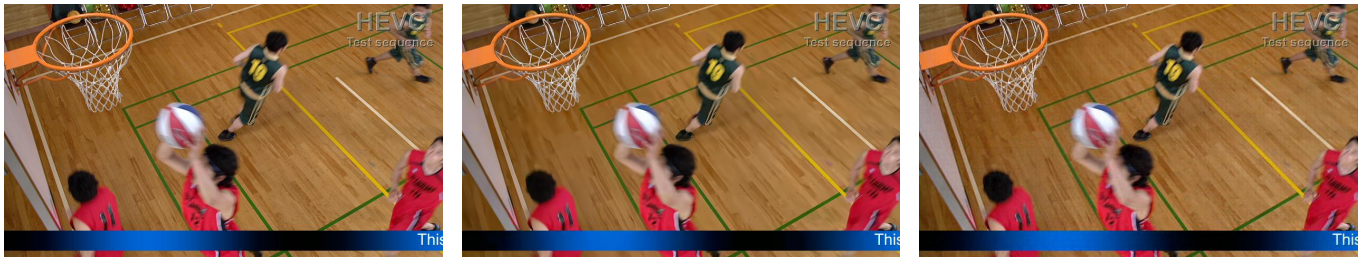


Fig. 6: RD results for different video sequences according to Y-PSNR, Y-MSSIM, and Y-SSIM metrics.

(a) Original Frame (b) Frame Encoded by HEVC - All Intra (c) Frame Encoded By Our All Intra Codec

Fig. 7: Comparisons for a frame from the sequence Blowing Bubbles.



(a) Original Frame (b) Frame Encoded by VVC - Low Delay P (c) Frame Encoded By Our Proposed Codec

Fig. 8: Comparisons for a frame from the sequence Basketball Drill.

models. Because of this, points with close hyper-parameters may imply the reported behavior.

A conventional codec uses both intra and inter prediction during P-frame encoding, while our model limits inter prediction to temporal correlation and accuracy of the warping process. Nevertheless, the proposed inter-encoding outperforms the neural architecture, which uses only intra frames. That is, the adoption of P-frames is shown to be superior to a sequence encoded only by I-frames. Moreover, we presented a fully operational end-to-end video codec based on autoencoder, that uses a completely independent GOP structure that allows for parallel encoding/decoding and uses a motion model that is not limited to discrete spatial locations. While several improvements can be made, at this point we already have an encoder that can be comparable with HEVC (all-intra) in some cases.

VI. Conclusions

In this paper, we extend our previous work [36] to implement a learning-based video codec with low-delay and all-intra configurations, with an independent GOP structure. Some future work may be devoted to improving the optical flow predictions by decomposing the reference frames into feature maps or performing transformations with learned kernels. Furthermore, entropy models can become more robust by adopting, for example, mixtures models. Additionally, some intra prediction processes can also be incorporated into P-frame encoding. Inter prediction can be refined by proposing a network that merges the assignments of motion and residue prediction, allowing sharing of weights and joint optimization. Finally, a joint training strategy for the entire architecture can significantly improve the RD performance since inter-frame coding is highly dependent on the initial intra-frame reference.

Acknowledgment

Part of the results presented in this work was obtained through the *Deep Codec* project funded by Samsung Eletrônica da Amazônia Ltda under the Brazillian Informatics Law 8.248/91. BM thanks CNPq PQ 308548/2018-3.

References

[1] G. M. D. T. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, 2019.

[2] E. Koeze and N. Popper, "The virus changed the way we internet," Apr 2020. [Online]. Available: https://www.nytimes.com/interactive/2020/04/07/technology/coronavirus-internet-use.html

[3] A. Habibi, "Hybrid coding of pictorial data," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 614–624, 1974.

[4] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 13, no. 7, pp. 560–576, 2003.

[5] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[6] G. Sullivan, "Versatile video coding (vvc) arrives," in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, 2020, pp. 1–1.

[7] I. Seidel, "Exploiting satd properties to reduce energy in video coding," Ph.D. dissertation, The Federal University of Santa Catarina, https://repositorio.ufsc.br/handle/123456789/216224, 2020.

[8] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wanga, "Image and video compression with neural networks: A review," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[9] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, "Deep learning-based video coding: a review and a case study," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–35, 2020.

[10] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, "Video compression through image interpolation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 416–431.

[11] O. Rippel, S. Nair, C. Lew, S. Branson, A. G. Anderson, and L. Bourdev, "Learned video compression," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[12] A. Habibian, T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen, "Video compression with rate-distortion autoencoders," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7033–7042.

[13] G. Lu, X. Zhang, W. Ouyang, L. Chen, Z. Gao, and D. Xu, "An end-to-end learning framework for video compression," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[14] A. R. T. Dumas and C. Guillemot, "Autoencoder based image compression: Can the learning be quantization independent?" in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018.

[15] D. H. Ballard, "Modular learning in neural networks." in *AAAI*, vol. 647, 1987, pp. 279–284.

[16] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5306–5314, preprint available at http://arxiv.org/abs/1608.05148.

[17] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," in *International Conference on Learning Representations (ICLR)*, April 2016.

[18] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, and G. Toderici, "Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks," Cornell University Library, Tech. Rep., mar 2017. [Online]. Available: http://arxiv.org/abs/1703.10114

[19] K. Gregor, F. Besse, D. Jimenez Rezende, I. Danihelka, and D. Wierstra, "Towards conceptual compression," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3549–3557. [Online]. Available: http://papers.nips.cc/paper/6542-towards-conceptual-compression.pdf

[20] M. H. Baig, V. Koltun, and L. Torresani, "Learning to inpaint for image compression," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1246–1255.

[21] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *International Conference on Learning Representations (ICLR)*, April 2017, pp. 1–27.

[22] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," Tech. Rep., 2017. [Online]. Available: https://openreview.net/pdf?id=rJiNwv9gg

[23] F. Mentzer, L. V. Gool, and M. Tschannen, "Learning better lossless compression using lossy compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6638–6647.

[24] J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear transform coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 339–353, 2020.

[25] S. Singh, S. Abu-El-Haija, N. Johnston, J. Ballé, A. Shrivastava, and G. Toderici, "End-to-end learning of compressible features," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 3349–3353.

[26] S. Santurkar, D. Budden, and N. Shavit, "Generative compression," in *2018 Picture Coding Symposium (PCS)*. IEEE, 2018, pp. 258–262.

[27] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," Jan. 2016, 4th International Conference on Learning Representations, ICLR 2016 ; Conference date: 02-05-2016 Through 04-05-2016.

[28] X. Jin, Z. Chen, S. Liu, and W. Zhou, "Augmented coarse-to-fine video frame synthesis with semantic loss," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2018, pp. 439–452.

[29] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[30] D. Sun, X. Yang, M. Liu, and J. Kautz, "Models matter, so does training: An empirical study of cnns for optical flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[31] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, "Unsupervised deep learning for optical flow estimation," in *AAAI*, 2017, pp. 1495–1501. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14388

[32] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," 2016, computer Vision and Pattern Recognition.

[33] C.-H. Lin and S. Lucey, "Inverse compositional spatial transformer networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[35] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018, international Conference on Learning Representations.

[36] R. C. d. Silva, N. D. G. Jr., P. Sanches, H. C. Jung, E. Peixoto, B. Macchiavello, E. M. Hung, V. Testoni, and P. G. Freitas, "Joint motion and residual information latent representation for p-frame coding," in *3rd Challenge on Learned Image Compression*, 2020.

[37] D. Minnen, J. Ballé, and G. Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression," in *NIPS*, sep 2018. [Online]. Available: http://arxiv.org/abs/1809.02736

[38] "Dataset of the CVPR workshop and challenge on learned image compression (CLIC)," online, http://www.compression.cc.

[39] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017, DIV2K dataset: DIVerse 2K resolution high quality images as used for the challenges at NTIRE (CVPR 2017 and CVPR 2018) and at PIRM (ECCV 2018), available from https://data.vision.ee.ethz.ch/cvl/DIV2K/.

[40] H. Nemoto, P. Hanhart, P. Korshunov, and T. Ebrahimi, "Ultra-Eye: UHD and HD images eye tracking dataset," in *Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, Singapore, September 2014, available from http://mmspg.epfl.ch/ultra-eye. [Online]. Available: http://infoscience.epfl.ch/record/200190

[41] L. Jin, J. Y. Lin, S. Hu, H. Wang, P. Wang, I. Katsavounidis, A. Aaron, and C.-C. J. Kuo, "Mcl-jci dataset," online, http://mcl.usc.edu/mcl-jci-dataset/.

[42] ——, "Statistical Study on Perceived JPEG Image Quality via MCL-JCI Dataset Construction and Analysis," in *Electronic Imaging (2016), the Society for Imaging Science and Technology (IS&T)*, 2016.

[43] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[44] "Youtube dataset for video compression research," online, https://media.withyoutube.com/.