DATA-DRIVEN INTRA-PREDICTION MODES IN THE DEVELOPMENT OF THE VERSATILE VIDEO CODING STANDARD

Jonathan Pfaff[1], Philipp Helle[1], Philipp Merkle[1], Michael Schäfer[1], Björn Stallenberger[1], Tobias Hinz[1], Heiko Schwarz[1, 2], Detlev Marpe[1], Thomas Wiegand[1, 3]
[1]Fraunhofer HHI, Berlin, Germany, [2]FU Berlin, Berlin, Germany, [3]TU Berlin, Berlin, Germany

Abstract – *In this paper, intra-prediction modes for video coding that were designed using data-driven methods are presented. These predictors were incorporated into a test model of the emerging versatile video coding (VVC) standard and yield compression benefit over state-of-the-art intra-prediction tools. However, most of the use cases for video coding require severe complexity and memory restrictions, in particular at the decoder side. As data-driven methods typically result in predictors that are described by a large set of parameters and operations, satisfying these constraints turned out to be a difficult task. The purpose of this paper is to outline key steps in the complexity reduction of the trained intra-prediction modes that were discussed in the VVC standardization activity. These simplifications finally led to matrix-based intra-prediction (MIP) which is part of the current VVC draft international standard.*

Keywords – Video Coding, intra-prediction.

1. INTRODUCTION

In recent years, the demand for broadcasting, streaming and storing of video content has significantly increased, but memory and transmission capacities are limited resources. As a consequence, in 2017, a Call for Proposals (CfP) for new video coding technologies with increased compression capabilities compared to state-of-the-art codecs was issued by the Joint Video Experts Team (JVET), [27].

One of the responses given to that call was a video codec submitted by Fraunhofer HHI, [2, 20]. This codec has a hybrid block based design and includes several advanced tools. Some of these advanced concepts were contained in the Joint Exploration Model (JEM) developed by the JVET [11], while others were newly proposed. Among these newly proposed tools were intra-prediction modes that were designed as the outcome of a training experiment based on a large set of training data. These intra-prediction modes provide significant coding gains over state-of-the-art video coding technologies. They are represented by fully connected neural-networks with several layers.

After results of the CfP were received, experts of the JVET collaboratively initiated a standardization process for a new video coding standard called versatile video coding (VVC), [19]. Here, the development of a standard which enables substantial compression benefits compared to existing technologies, in particular within the emerging scenario of coding UHD or HDR-content, was targeted. In the VVC standardization activity, individual coding tools with promising compression performance were investigated by the JVET within so-called core experiments. Among these tools were the aforementioned data-driven intra-prediction modes.

In the course of their investigation, several modifications of the initially proposed intra-prediction modes which mainly target a complexity reduction were developed. The final variant, called matrix-based intra-prediction (MIP) represents a low complexity version. MIP has a small memory requirement and does not increase the number of multiplications in comparison to existing intra-prediction modes. It was included into the working draft 5 of the VVC standard at the 14th JVET-meeting in Geneva in March 2018, [9].

Recently, several interesting machine-learning based approaches to image compression have been developed. Without aiming at completeness, we mention the work of Ballé et al., [3], [4], Agustsson, Mentzer et al., [1], [18], Minnen et al. [17], Rippel et al. [24], Theis et al. [30] and Toderici et al. [31]. In these approaches, image compression systems were designed which do not use a block-based approach and which do not use intra-prediction in a traditional sense. Rather, they extract several features from the input image via a convolutional neural-network. These features are quantized into symbols and then transmitted in the bitstream. The decoder reconstructs the image by a deconvolutional neural-network which is applied to the dequantized symbols. Parts of this network might also be used in an arithmetic coding engine to model conditional probabilities of coded symbols. The parameters of the neural-networks are obtained on a large set of training data.

In our work, we used machine-learning techniques to develop a compression tool which still fits into a hybrid block-based architecture. Such an architecture is used in many existing video codecs like advanced video coding (AVC) [12, 33] or high efficiency video coding (HEVC) [13, 29] and also forms the basis of the emerging VVC [8]. Within this architecture, our intra predition modes simply replace or complement the classical intra-prediction

modes which are already used in traditional video codecs. Other components of the surrounding video codec like block-partitioning or transform and residual-coding are not altered by our method.

This paper is organized as follows. In section 2, we describe the general setup for designing data-driven intra-prediction modes. In section 3, we depict their realization by fully connected neural-networks. In section 4, a simplification of the neural-networks via prediction into the transform domain is outlined. MIP is described in section 5. In the final section 6, some conclusions shall be considered.

2. DATA-DRIVEN DESIGN OF INTRA-PREDICTION MODES

In typical block-based hybrid video codecs, predictive coding is used. Thus, when a receiver of a video signal wants to reconstruct the content of a transmitted video on a given block, out of information that is already available, it generates a prediction signal. This prediction signal serves as a first approximation of the video signal to be reconstructed. In a second step, a prediction residual is added to generate the reconstructed video signal. This prediction residual needs to be transmitted in the bitstream and thus the quality of the prediction signal greatly influences the compression efficiency.

There are two methods to generate a prediction signal: Inter- and intra-picture prediction. In the case of inter-picture prediction, the prediction signal is generated by motion-compensated prediction where already decoded video frames which are different from the current frame serve as the input.

Conversely, in the case of intra-prediction, the prediction signal is generated out of already reconstructed sample values that belong to the same frame and are typically spatially adjacent to the current block. Thus, as shown in Fig. 1, input for intra-prediction are the reconstructed samples r above and left of a block of samples to be predicted.

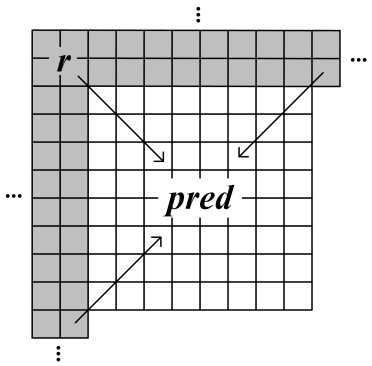

Fig. 1 – intra-prediction on a single block. In principle, all reconstructed samples are available.

In conventional video codecs like HEVC and also in the JEM, the intra-prediction signal is generated either by angular prediction or by the DC and planar modes. The angular prediction modes copy the already reconstructed sample values on the lines left and above of the block along a specific direction that is parametrized by an angular parameter. Here, for fractional angular positions, an interpolation filtering is applied to the reference samples. The DC mode generates a constant prediction signal that corresponds to the mean sample value of the adjacent samples, while the planar mode interpolates between a prediction along the horizontal and the vertical direction. In the JEM, an additional post-filtering step, called position dependent prediction combination, PDPC [25], is optionally applied to the intra-prediction signal.

In our approach to intra-prediction, we tried to design n more general intra-prediction modes using data-driven methods. A priori, it was only assumed that the i-th intra-prediction mode should generate the prediction signal $pred_i$ as

$$pred_i = F_i(r; \theta_i); \qquad (1)$$

see Fig. 2 . Here, the function F_i is a predefined function which, however, depends on parameters θ_i that are determined in a training algorithm using a large set of training data. Note that when the prediction is used in the final codec, the parameters θ_i are fixed. For their determination, we developed a training algorithm that tries to simulate several aspects of modern video codecs. When executing it, we applied recent machine learning techniques like [15]. Key parts of our training algorithm are independent from the specific form of the prediction function F_i.

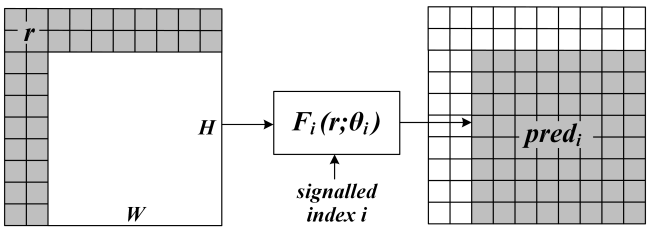

Fig. 2 – Design of intra-prediction modes with fixed function F_i and its trained parameters θ_i. The index i is transmitted.

A central problem one faces in the above design of more flexible intra-prediction modes is their complexity in comparison to traditional intra-prediction techniques described above. The reason is that since the optimal form of the intra-prediction modes in *(1)* is unkown, a rather large capacity of the neural-networks is assumed by which a larger set of functions can be approximated. In the VVC standardization process, the complexity of the prediction modes was assessed in two ways. First, the complexity to execute the function F_i was taken into account. This complexity can be measured for example in number of multiplications per sample or in terms of decoder runtime. Second, the memory requirement, i.e. the size of the parameters θ_i which need to be stored, turned out to be a very important aspect for a complexity evaluation of the method. In the sequel, intra-prediction modes

based on several variants of the function F_i that represent different degrees of complexity will be discussed.

3. NEURAL-NETWORK-BASED INTRA PREDICTORS

In our CfP response, each function F_i from *(1)* was given by a fully connected neural-network with three hidden layers, [21, 22]. For each rectangular block of width W and height H with W and H being integer powers of two between 4 and 32, n prediction modes were supported. The number n is equal to 35 for $max(W, H) < 32$ and is equal to 11, otherwise.

The neural-network-based intra-prediction modes are illustrated in Figures 3 and 4. Input for the prediction are the $d = 2(W + H + 2)$ reconstructed samples r on the two lines left and above the block, as well as the 2×2 corner on the top left. The dimension of the three hidden layers is equal to d. In order to improve the training and to reduce the number of parameters needed, these layers are shared by all prediction modes F_i. Their output can be interpreted as a set of features $ftr \in \mathbb{R}^d$ of the surrounding samples. In the last layer, these features are affine-linearly combined where this combination depends on the prediction mode i.

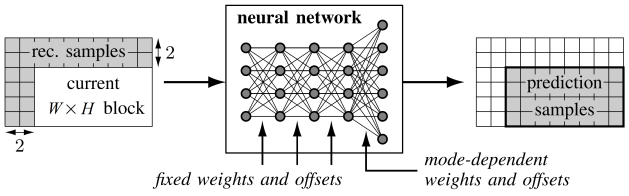

Fig. 3 – Overview of NN-based intra-prediction

For the signalization of the mode index $i \in \{0, \ldots, n-1\}$, we used a second neural-network whose input is the same vector of reconstructed samples r as above and whose output is a conditional probability mass function p over the n modes, given the reconstructed samples r. When one of the intra-prediction modes is to be applied at the decoder, a number $index \in \{0, \ldots, n-1\}$ needs to be parsed and the probability mass function p needs to be computed. Then the $index$-th most probable mode with respect to p has to be used; see Fig. 4. Here, the binarization of $index$ is such that small values of $index$ require less bins.

Our signalling approach shares similarities with some of the machine-learning based image compression approaches mentioned in the introduction. In [4], [18], [31], an arithmetic coding engine with conditional probabilities that are computed on the fly by a neural-network out of reconstructed symbols is used. In our approach, however, the conditional probability p is not directly invoked into the arithmetic coding engine in order to avoid a parsing dependency.

The parameters θ_i of the prediction modes, which correspond to the matrix coefficients and the offset-vector entries of the neural-network, were determined by attempting to minimize a specific loss function over a large set of training data. This loss function was defined as a linear combination of the ℓ_1 norm of the DCT-II-transform coefficients of the prediction residual and of a sigmoid term on these coefficients. The sigmoid term has a steep slope in some neighborhood of zero and its slope becomes smaller the farther away from zero. In this way, during training by gradient descent, the prediction modes are steered towards modes for which the energy of the prediction residual is concentrated in very few transform coefficients while most of the transform coefficients will be quantized to zero. This reflects the well-known fact that in the transform coding design of modern hybrid video codecs, it is highly beneficial in terms of rate-saving if a transform-coefficient can be quantized to zero; see for example [28]. In the training algorithm, all prediction modes over all block shapes were trained jointly. The parameters of the neural-network used in the mode signalization were also determined in that algorithm. In the optimization, a stochastic gradient descent approach with Adam-optimizer [15] was applied. For more details on the training algorithm, we refer to [14].

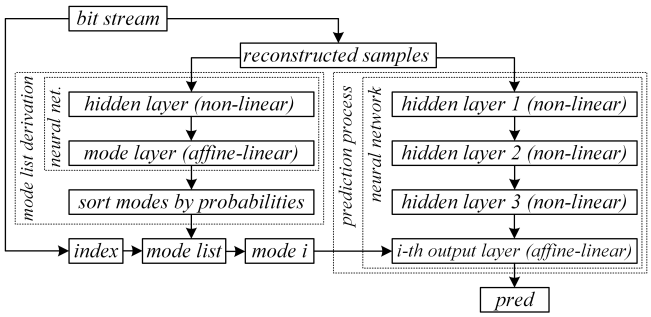

Fig. 4 – Construction of the prediction signal $pred$ at the decoder using neural-networks. A hidden layer maps an input vector x to $\sigma(A_{hid}x + b_{hid})$ with σ being the exponential linear unit function [22]. The mode layer maps its input x to $A_{mode}x + b_{mode}$ which represents, up to normalization, the logarithms of the discrete probability distribution of the modes [22]. The output of hidden layer 3 on the right is the feature vector $ftr \in \mathbb{R}^d$. The i-th output layer maps its input x to $A_i x + b_i$ which represents the $W \times H$-dimensional prediction signal.

The neural-network-based intra-prediction modes were integrated in a software that was equivalent to the HEVC reference software anchor with the extension that it also supported non-square partitions, [22]. They were added as complementary to the HEVC intra-prediction modes. In the all-intra configuration, they gave a compression benefit of -3.01%; see [22, Table 1]. Here and in the sequel, all objective results report luma Bjøntegaard delta (BD) rates according to [5], [6]. Moreover, the standard QP-parameters 22, 27, 32 and 37 are used and the simulations are performed following JVET common test conditions, [7]. For the test sequences of [22], the neural-network prediction modes were used for approximately 50% of all intra blocks.

As reported in [22], the measured average decoding time was 248%. Throughout the paper, a conventional CPU-

based cluster was used for measuring decoder runtimes. No SIMD- or GPU-based optimization was applied. According to the architecture of the neural-networks used, the total number of parameters needed for the prediction modes described in this section is about 5.5 million. Since all parameters were stored in 16-bit-precision, this corresponds to a memory requirement of about 11 Megabyte.
Our method should be compared to the method of [16], where also intra-prediction modes based on fully connected layers are trained and integrated into HEVC. While the compression benefit reported in [16] is similar to ours, its decoder runtime is significantly higher; see Table III of [16].

4. PREDICTION INTO THE TRANSFORM DOMAIN

The complexity of the neural-network-based intra-prediction modes from the previous section increases with the block-sizes W and H. This is particularly true for the last layer of the prediction modes, where for each output sample of the final prediction, $2 \cdot (W + H + 2)$ many multiplications have to be carried out and a $(W \cdot H) \times (2 \cdot (W + H + 2))$-matrix has to be stored for each prediction mode.
Thus, instead of predicting into the sample domain, in subsequent work [21, 14] we transformed our predictors such that they predict into the frequency domain of the discrete cosine transform DCT-II. Thus, if T is the matrix representing the DCT-II, the i-th neural-network predictor from the previous section predicts a signal $pred_{i,tr}$ such that the final prediction signal is given as

$$pred_i = T^{-1} \cdot pred_{i,tr}.$$

The key point is that each prediction mode has to follow a fixed sparsity pattern: For a lot of frequency components, $pred_{i,tr}$ is constrained to zero in that component, independent of the input. In other words, if $A_{i,tr}$ is the matrix used in the last layer for the generation of $pred_{i,tr}$, then for each such frequency component, the row of the matrix $A_{i,tr}$ corresponding to that component consists only of zeros. Thus, the entries of that row do not need to be stored and no multiplications need to be carried out in the matrix vector product $A_{i,tr}i \cdot ftr$ for that row. The whole process of predicting into the frequency domain is illustrated in Figure 5.

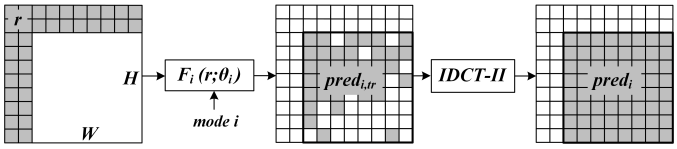

Fig. 5 – intra-prediction into the DCT-II domain. The white samples in the output $pred_{i,tr}$ denote the DCT-coefficients which are constrained to zero. The pattern depends on the mode i.

In the underlying codec, the inverse transform T^{-1} is already applied to the transform coefficients c of the prediction residual res. Thus, at the decoder, one can replace the computation of $T^{-1}(c)$ by the computation of $T^{-1}(c + pred_{i,tr})$. Consequently, as long as the prediction residual is non-zero, no extra inverse transform needs to be executed when passing from $pred_{i,tr}$ to $pred_i$.
The weights θ_i of the involved neural-networks were obtained in two steps. First, the same training algorithm as in the previous section was applied and the predictors were transformed to predict into the frequency domain. Then, using again a large set of training data, for each predictor it was determined which of its frequency components could be set to zero without significantly changing its quality on natural image content. For more details, we refer to [14].
As a further development, for the signalization of conventional intra-prediction modes, a mapping from neural-network-based intra-prediction modes to conventional intra-prediction modes was implemented. Via this mapping, whenever a conventional intra-prediction mode is used on a given block, neighboring blocks which use the neural-network-based prediction mode can be used for the generation of the list of most probable modes on the given block. For further details, we refer to [14].
In an experimental setup similar to the one of the previous section, the intra-prediction modes of the present section gave a compression benefit of -3.76% luma-BD-rate gain; see [14, Table 2]. Compared to the results of the previous section, these results should be interpreted as saying that the prediction into the transform-domain with the associated reduction of the last layer does not yield any significant coding loss and that the mapping from neural-network-based intra-prediction modes to conventional intra-prediction modes additionally improves the compression efficiency. As reported in [14], the measured decoder runtime overhead is 147%, the measured encoder runtime overhead is 284%. Thus, from a decoder perspective, the complexity of the method has been significantly reduced. Also, the memory requirement of the method was reduced significantly. In the architecture from Figure 5, approximately 1 Megabyte of weights need to be stored.

5. MATRIX-BASED INTRA-PREDICTION MODES

In the further course of the standardization, the data-driven intra-prediction modes were again simplified leading to matrix-based intra-prediction (MIP) modes, [23, 26]. These modes were adopted into the VVC-standard at the 14-th JVET-meeting in Geneva [9]. The complexity of the MIP modes can be described as follows. First, the number of multiplications per sample required by each MIP-prediction mode is at most four and thus not higher than for the conventional intra-prediction modes which require four multiplications per sample either due to the four-tap interpolation filter for fractional angle positions or due to PDPC. Second, the memory requirement of the method is strongly reduced. Namely, the memory to store

all MIP-prediction modes is equal to 8 Kilobyte. This corresponds to a memory reduction by a factor 1000 and by a factor 100 in comparison to the methods of section 3 and section 4, respectively. The key idea to achieve the aforementioned two complexity constraints is to use downsampling and up-sampling operations in the domain of the prediction input and output.

For predicting the samples of a $W \times H$-block, W and H integer powers of two between 4 and 64, MIP takes one line of H and W reconstructed neighboring boundary samples left and above the block as input. Then, the prediction signal is generated using the following three steps which are also summarized in Figure 6:

1. From the boundary samples, four samples in the case $W = H = 4$ and eight samples, else, are extracted by averaging.

2. A matrix-vector multiplication, followed by addition of an offset, is carried out with the averaged samples as an input. The result is a reduced prediction signal on a subsampled set of samples in the original block.

3. The prediction signal at the remaining positions is generated from the prediction signal on the subsampled set by linear interpolation.

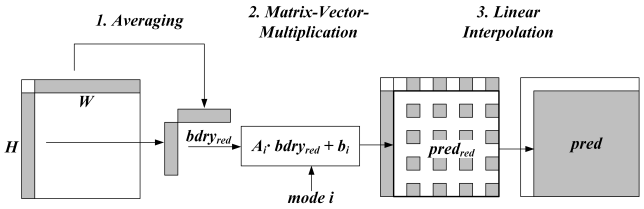

Fig. 6 – The flow chart of matrix-based intra-prediction for $W \times H$-block.

The averaging step on the boundary, which is performed for all MIP-modes, could be interpreted as a low complexity version of the joint feature extraction that was part of the neural-network-based intra-prediction; see section 3. Moreover, one could rephrase the linear interpolation step by saying that each MIP-mode predicts into the transform domain of the $(5, 3)$-wavelet transform, where only low subbands are predicted to be non-zero. Thus, conceptionally, this part of MIP-prediction is similar to the prediction into the DCT-domain described in the previous section 3. However, note that for the predictors predicting into the DCT-domain, not all high frequency components of the prediction signal were set to zero but rather a more flexible sparsity pattern was used whereas the MIP-predictors are constrained to generate only low-pass signals.

We now describe each of the three steps in the MIP prediction in more detail. In the first step, the left and top input boundaries $bdry^{top}$ and $bdry^{left}$ are reduced to smaller boundaries $bdry^{top}_{red}$ and $bdry^{left}_{red}$. Here, $bdry^{top}_{red}$ and $bdry^{left}_{red}$ both consists of 2 samples in the case of a 4×4-block and both consist of 4 samples in all other cases.

In the case of a 4×4-block, for $0 \leq i < 2$, one defines

$$bdry^{top}_{red}[i] = bdry^{top}[2i] + bdry^{top}[2i + 1].$$

In all other cases, if the block-width W is given as $W = 4 \cdot 2^n$, for $0 \leq i < 4$ one defines

$$bdry^{top}_{red}[i] = \frac{1}{2^n} \sum_{j=0}^{2^n-1} bdry^{top}[2^n \cdot i + j].$$

The reduced left boundary $bdry^{left}_{red}$ is defined analogously. The two boundaries $bdry^{top}_{red}$ and $bdry^{left}_{red}$ are concatenated to form the reduced boundary

$$bdry_{red} = [bdry^{left}_{red}, bdry^{top}_{red}]; \qquad (2)$$

see Fig. 7. It has size 4 for 4×4 blocks and size 8, elsewhere.

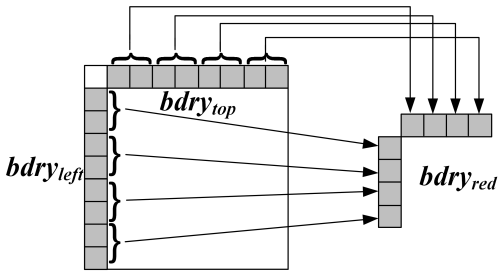

Fig. 7 – The averaging step for an 8×8-block. This results in four samples (two in the case of 4×4-blocks) along each axis.

In the second step, out of the reduced input vector $bdry_{red}$ one generates a reduced prediction signal $pred_{red}$. The latter signal is a signal on the downsampled block of width W_{red} and height H_{red}. Here, W_{red} and H_{red} are defined as:

$$W_{red} = 4,\ H_{red} = 4;\ \text{if}\ W = H = 4$$
$$W_{red} = \min(W, 8),\ H_{red} = \min(H, 8);\ \text{elsewhere.}$$

The reduced prediction signal $pred_{red}$ of the i-th prediction mode is computed by calculating a matrix vector-product and adding an offset:

$$pred_{red} = A_i \cdot bdry_{red} + b_i. \qquad (3)$$

Here, A_i is a matrix that has $W_{red} \cdot H_{red}$ rows and 4 columns if $W = H = 4$ and 8 columns in all other cases. Moreover, b is a vector of size $W_{red} \cdot H_{red}$.

The matrices and offset vectors needed to generate the prediction signal are taken from three sets S_0, S_1, S_2. The set S_0 consists of 18 matrices each of which has 16 rows and 4 columns and 18 offset vectors of size 16. Matrices and offset vectors of that set are used for blocks of size 4×4. The set S_1 consists of 10 matrices , each of which has 16 rows and 8 columns and 10 offset vectors of size 16. Matrices and offset vectors of that set are used for blocks of sizes 4×8, 8×4 and 8×8. Finally, the set S_2 consists of 6 matrices , each of which has 64 rows and 8 columns and

of 6 offset vectors of size 64. Matrices and offset vectors of that set or parts of these matrices and offset vectors are used for all other block shapes.

In the third step, at the sample positions that were left out in the generation of $pred_{red}$, the final prediction signal arises by linear interpolation from $pred_{red}$. This linear interpolation is not needed if $W = H = 4$. To describe it, assume without loss of generality that $W \geq H$. One extends the prediction signal to the top by the reconstructed values and writes $pred_{red}[x][-1]$ for the first line. Then the signal $pred_{red}^{ups,ver}$ on a block of width W_{red} and height $2 * H_{red}$ is given as

$$\begin{aligned} pred_{red}^{ups,ver}[x][2y+1] &= pred_{red}[x][y] \\ pred_{red}^{ups,ver}[x][2y] &= \frac{1}{2}(pred_{red}[x][y-1] \\ &\quad + pred_{red}[x][y]) \end{aligned}$$

The latter process is carried out k times until $2^k \cdot H_{red} = H$. Next, a horizontal up-sampling operation is applied to the result of the vertical up-sampling. The latter up-sampling operation uses the full boundary left of the prediction signal; see Fig. 8.

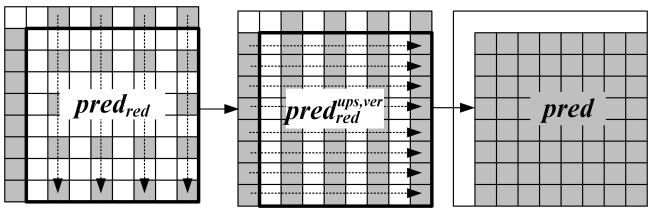

Fig. 8 – The final interpolation step for an 8×8-block. The second up-sampling operation uses the full boundary.

For each prediction mode generated by A_i, $b_i \in S_{0/1/2}$ with $i > 0$, also the transposed prediction mode is supported. This means that one interchanges $bdry_{red}^{top}$ and $bdry_{red}^{left}$, computes the matrix vector product and the offset addition as before and then interchanges the x and the y coordinate in the resulting reduced prediction signal. The up-sampling step is then carried out as before. As a consequence, for blocks of size 4×4, a total number of 35 MIP modes is supported. For blocks of size 8×4, 4×8 and 8×8, a total number of 19 MIP modes is supported. For all other block shapes, a total number of 11 MIP modes is supported.

The MIP-prediction mode was signalled using a most probable mode scheme that is based on intra-prediction modes of neighboring blocks, similar to the well-known signalization of conventional intra-prediction modes. The neural-network that predicts the conditional probability of an intra-prediction mode out of neighboring reconstructed samples was removed for complexity reasons.

In order to determine the matrices of the MIP-prediction modes, a training algorithm similar to the algorithm outlined in section 3 was used. Here, the constraints given by the input down-sampling, the output up-sampling and the sharing of the predictors across different block shapes were incorporated into the training algorithm.

The MIP-tool gave a compression benefit of -0.79% luma BD-rate gain, [23, Table 1]. The measured decoder runtime was 99% which means that MIP did not cause any decoder runtime overhead. The measured encoder runtime overhead was 138%. As for the other variants of our data-driven intra-prediction modes, different trade-offs between compression performance and encoder runtime overhead are possible and were developed subsequently. In this paper, the complexity issue is mainly considered from a decoder perspective. A software reference for the current version of MIP can be found in the document [32].

After its adoption into VVC, several further modifications were performed for the final design of MIP in the current VVC draft international standard [10]. Most importantly, all matrix coefficients of the involved matrices are represented by 8-bit integers and the offset vectors b_i from *(3)* are set to zero. For an efficient 8-bit implementation, the matrix-vector multiplication $A_i \cdot bdry_{red}$ from *(3)* is replaced by the matrix-vector multiplication

$$\widetilde{A}_i \cdot y_{red} + bdry_{red}[0] \cdot 1, \qquad (4)$$

where the vector y_{red} is defined by

$$\begin{aligned} y_{red}[0] &= bdry_{red}[0] - 2^{B-1}, \\ y_{red}[i] &= bdry_{red}[i] - bdry_{red}[0], i > 0. \end{aligned}$$

Here, 1 denotes the vector of ones and B denotes the bit-depth. Since the entries of y_{red} are typically smaller than the entries of $bdry_{red}$, this modification of the matrix-vector multiplication leads to a smaller impact of the approximation error that arises when one passes from the trained floating point matrices to the 8-bit integer matrices. The result of the matrix-vector multiplication *(4)* is right-shifted by 6 to generate the final prediction signal. The constant right-shift 6 was achieved by smoothly restricting the dynamic range of the matrix-entries already during the training process. Also, several non-normative encoder-speedups for MIP were included into the reference software.

6. CONCLUSION

In this paper, several variants of data-driven intra-prediction modes were presented. Such modes can improve the compression efficiency of state-of-the-art video codecs. However, a standard like the emerging versatile video coding is targeted to both enable high compression rates and to be implementable on multiple types of consumer devices at moderate complexity and costs. The latter requirement forms a particular challenge for the presented approach since, a priori, the resulting intra-prediction modes are much less structured than conventional ones and thus require a lot of parameters to be stored. As a consequence, architectural constraints that reflect some well-known image processing methods were invoked into the training and design of the predictors. In particular, sparsification in the transform domain and

subband decomposition were employed. In this way, one could significantly decrease the complexity of the predictors and finally make them suitable for a broad application scenario like versatile video coding.

REFERENCES

[1] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, *Soft-to-hard vector quantization for end-to-end learning compressible representations*, in *Advances in Neural Information Processing Systems*, 2017

[2] M. Albrecht et al., *Description of SDR, HDR and* 360° *video coding technology proposal by Fraunhofer HHI*, Doc. JVET-J0014, San Diego, 2018

[3] J. Ballé, V. Laparra, E. P. Simoncelli, *End-to-end optimized image compression*, in *Int. Conf. on Learning Representations*, 2017

[4] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, *Variational image compression with a scale hyperprior*, in *Int. Conf. on Learning Representations*, 2018

[5] G. Bjøntegaard, *Calculation of average PSNR differences between RD-curves*, Doc. VCEG-M33, 2001

[6] G. Bjøntegaard, *Improvement of BD-PSNR Model*, Doc. VCEG-AI11, 2008

[7] F. Bossen, J. Boyce, X. Li, V. Seregin, K. Sühring, *JVET common test conditions and software reference configurations for SDR video*, Doc. JVET-N1010, Geneva, 2019

[8] B. Bross, *Versatile Video Coding (Draft 1)*, Doc. JVET-J1001, San Diego, 2018

[9] B. Bross, J. Chen, S. Liu, *Versatile Video Coding (Draft 5)*, Doc. JVET-N1001, Geneva, 2019

[10] B. Bross, J. Chen, S. Liu, *Versatile Video Coding (Draft 8)*, Doc. JVET-Q2001, Brussels, 2020

[11] J. Chen, M. Karczewicz, Y.-W. Huang, K. Choi, J.-R. Ohm, G. J. Sullivan, *The Joint Exploration Model (JEM) for Video Compression with Capability beyond HEVC*, to appear in *IEEE Trans. Circuits and Systems for Video Technol.*

[12] ITU-T and ISO/IEC, *Advanced Video Coding for Generic Audiovisual Services, H.264 and ISO/IEC 14496-10, vers. 1*, 2003

[13] ITU-T and ISO/IEC, *High Efficiency Video Coding, H.265 and ISO/IEC 23008-2, vers. 1*, 2013

[14] P. Helle, J. Pfaff, M. Schäfer, R. Rischke, H. Schwarz, D. Marpe, T. Wiegand, *Intra Picture Prediction for Video Coding with Neural Networks*, in *Proc. IEEE Data Compression Conf. (DCC), Snowbird*, 2019

[15] D. Kingma, J. Ba (2015), *Adam: A Method for Stochastic Optimization*, in *Int. Conf. on Learning Representations*, 2014

[16] J. Li, B. Li, J. Xu, R. Xiong, W. Gao, *Fully Connected Network-Based Intra prediction for Image Coding*, in *IEEE Trans. Image Process*, vol. 27, no. 7, 2018, pp. 3236-3247

[17] D. Minnen, J. Ballé, G. Toderici, *Joint autoregressive and hierarchical priors for learned image compression*, in *Advances in Neural Information Processing Systems*, 2018

[18] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, *Conditional probability models for deep image compression*, in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[19] G. J. Sullivan, J.-R. Ohm, *Meeting Report of the 10th JVET Meeting*, Doc. JVET-J1000, San Diego, 2018

[20] J. Pfaff, H. Schwarz, D. Marpe, B. Bross, S. De-Luxán-Hernández, P. Helle, C. R. Helmrich, T. Hinz, W.-Q. Lim, J. Ma, T. Nguyen, J. Rasch, M. Schäfer, M. Siekmann, G. Venugopal, A. Wieckowski, M. Winken, T. Wiegand *Video Compression Using Generalized Binary Partitioning, Trellis Coded Quantization, Perceptually Optimized Encoding, and Advanced Prediction and Transform Coding* , to appear in *IEEE Trans. Circuits and Systems for Video Technol.*

[21] J. Pfaff, P. Helle, D. Maniry and S. Kaltenstadler, B. Stallenberger, P. Merkle, M. Siekmann, H. Schwarz, D. Marpe, T. Wiegand, *Intra prediction modes based on Neural Networks*, Doc. JVET-J0037, San Diego, 2018

[22] J. Pfaff, P. Helle, D. Maniry, S. Kaltenstadler, W. Samek, H. Schwarz, D. Marpe, T. Wiegand, *Neural network based intra prediction for video coding*, in *Proc. SPIE Applic. of Digital Image Process. XLI*, volume 10752, 2018

[23] J. Pfaff, B. Stallenberger, M. Schäfer, P. Merkle, P. Helle, T. Hinz, H. Schwarz, D. Marpe, T. Wiegand, *CE3: Affine linear weighted intra prediction (CE3-4.1, CE3-4.2)*, Doc. JVET-N0217, Geneva, 2019

[24] O. Rippel, L. Bourdev, *Real-time adaptive image compression*, in *Proc. of Machine Learning Research*, vol. 70, 2017, pp. 2922–2930

[25] A. Said, X. Zhao, M. Karczewicz, J. Chen, *Position dependent prediction combination for intra-frame video coding*, in *IEEE International Conference on Image Processing (ICIP)*, 2016

[26] M. Schäfer, B. Stallenberger, J. Pfaff, P. Helle, H. Schwarz, D. Marpe, T. Wiegand *An Affine-Linear intra prediction With Complexity Constraints* , in *IEEE International Conference on Image Processing (ICIP)*, 2019

[27] A. Segall, V. Baroncini, J. Boyce, J. Chen, T. Suzuki, *Joint Call for Proposals on Video Compression with Capability beyond HEVC*, Doc. JVET-H1002, Macau, 2017

[28] J. Sole, R. Joshi, N. Nguyen, T. Ji, M. Karczewicz, G. Clar, F. Henry, A. Dueñas, *Transform Coefficient Coding in HEV*, in *IEEE Trans. Circuits and Systems for Video Technol.*, vol. 22, no. 12, 2012, pp. 1765-1777

[29] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, *Overview of the High Efficiency Video Coding (HEVC) Standard*, in *IEEE Trans. Circuits and Systems for Video Technol.*, vol. 22, no. 12, 2012, pp. 1649-1668

[30] L.Theis, W. Shi, A. Cunningham, F. Huszár, *Lossy image compression with compressive autoencoders*, in *Int. Conf. on Learning Representations*, 2017

[31] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, *Full resolution image compression with recurrent neural networks*, in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017

[32] G. Van der Auwera, L. Li, A. Filippov, *CE3: Summary report on intra prediction and mode coding*, Doc. JVET-N0023, Geneva, 2019

[33] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, A. Luthra, *Overview of the H.264/AVC Video Coding Standard*, in *IEEE Trans. Circuits and Systems for Video Technol.*, vol. 13, no. 7, 2003, pp. 560-576