Intra-Frame Coding Using a Conditional Autoencoder

Fabian Brand, *Graduate Student Member, IEEE*, Jürgen Seiler, *Senior Member, IEEE*, and André Kaup, *Fellow, IEEE*

***Abstract*—Exploiting spatial redundancy in images is responsible for a large gain in the performance of image and video compression. The main tool to achieve this is called intra-frame prediction. In most state-of-the-art video coders, intra prediction is applied in a block-wise fashion. Up to now angular prediction was dominant, providing a low-complexity method covering a large variety of content. With deep learning, however, it is possible to create prediction methods covering a wider range of content, being able to predict structures which traditional modes can not predict accurately. Using the conditional autoencoder structure, we are able to train a single artificial neural network which is able to perform multi-mode prediction. In this paper, we derive the approach from the general formulation of the intra-prediction problem and introduce two extensions for spatial mode prediction and for chroma prediction support. Moreover, we propose a novel latent-space-based cross component prediction. We show the power of our prediction scheme with visual examples and report average gains of 1.13% in Bjøntegaard delta rate in the luma component and 1.21% in the chroma component compared to VTM using only traditional modes.**

***Index Terms*—Video coder, intra prediction, conditional autoencoder, deep learning.**

I. Introduction

NATURAL images contain a large amount of spatial redundancy. To compress an image to a small size, the encoder typically exploits the spatial correlation of neighboring pixels and reduce this redundancy. This is very often done with a suitable prediction scheme. Since most current codecs, like High Efficiency Video Coding (HEVC) [1] or Versatile Video Coding (VVC) [2], [3], are block-based, the task at hand is to predict the content of one block from a causal neighborhood. The causal neighborhood usually consists of pixels above and to the left of the current block. Currently, angular intra prediction is dominant in HEVC and VVC. Angular prediction is particularly useful to predict blocks which mainly consist of one sharp edge. In such prediction schemes, the content from the neighborhood is extended in a certain angle into the block. The angle is subject to rate-distortion-optimization (RDO) at the encoder and has to be transmitted to the decoder. It is common that intra prediction uses additionally signaled side-information. Additionally to angular prediction modes, HEVC and VVC define a DC and a planar mode which are used for blocks that do not consist of a single edge. A novel feature in VVC is Matrix Intra Prediction (MIP) [4], [5] which is the first learning-based intra prediction to be incorporated in a standard. MIP interprets the support area as vector and performs a matrix multiplication to obtain the prediction signal. The matrix is picked out of several options which were learned in advance using a large training set and are explicitly saved in the software.

Another improvement in intra prediction, specifically addressing chroma coding, is the Cross-Component Linear Model (CCLM) [6]. With CCLM, the chroma component can be predicted from the luma component using a simple linear model. The parameters of this model can be estimated from the support area of the chroma and luma components and therefore do not have to be transmitted. Furthermore, VTM allows using different reference lines for intra prediction with a distance of up to four pixels from the block to be predicted.

Apart from novel methods for intra prediction, the performance of VVC is due to a number of other new features. A large gain was achieved by using a new tree partitioning by extending the Quadtree which was used in HEVC with a nested Multi Type Tree (MTT), thus allowing for rectangular blocks also in intra prediction. In coding the chroma component, a so called Chroma Separate Tree (CST) is used, which creates two separate trees for luma and chroma content. Both methods increase the performance significantly at the cost of complexity. Furthermore, there were improvements in in-loop-filtering by introducing the Adaptive Loop Filter (ALF) [7], which trains several filters during the encoding process which are explicitly signaled. Beside the mentioned tools, there are several others which individually sometimes only have a small improvement, but altogether give a gain of 24.2% rate savings over HEVC in all intra configuration for version 6 [8].

The remainder of the paper is organized as follows: Section 2 will give an overview of state-of-the-art methods for learning-based coding tools as well as relevant machine learning techniques. In Section 3, we will first introduce the concept of the conditional autoencoder for intra prediction with a spatially correlated latent space. Afterwards, as a new contribution, we will introduce several methods how the components which have been introduced for the luma component can be applied for predicting the chroma component. In Section 4, we will evaluate all methods. At first, we will give visual examples demonstrating the flexibility of the proposed prediction method compared to state-of-the-art methods. Afterwards, we will evaluate the

Manuscript received June 18, 2020; revised September 28, 2020; accepted October 15, 2020. Date of publication October 29, 2020; date of current version February 22, 2021. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Michele Covell. *(Corresponding author: Fabian Brand.)*

The authors are with the Multimedia Communications and Signal Processing Friedrich-Alexander-Universit at Erlangen-Nurnberg, BY 91058 Erlangen, Germany (e-mail: fabian.brand@fau.de; juergen.seiler@fau.de; hongbojiang2004@gmail.com).

Digital Object Identifier 10.1109/JSTSP.2020.3034768

rate-distortion behavior of the methods, showing that both the prediction in the luma component as well as in the chroma component result in gains compared to state-of-the-art methods.

The main contributions of this paper are three-fold: First we propose an extension to the CAE model enforcing a spatially correlated latent space. This extension allows us to develop a spatial mode prediction scheme. Afterwards, we propose several chroma-prediction methods that extend our previous work. Finally, using the newly proposed models, we demonstrate that we are able to build a fully functional intra coding system giving gains over the state-of-the-art prediction methods in all three components.

II. Related Work

With the recent developments in the area of artificial neural networks and larger computational capabilities both for training and on end-user-devices for inference, there is a focus on using neural networks for image and video coding. Multiple methods have been proposed to use neural networks as coding tools in classical hybrid video coding.

First to mention is Matrix Intra Prediction (MIP), which is a complexity-reduced version of a neural-network-based intra prediction method [9]. Pfaff *et al.* train several neural networks to serve as prediction modes for intra prediction. Furthermore, the authors propose a second network to predict the mode itself, which can be used to generate a Most Probable Mode (MPM) list as used in HEVC and VVC. This model was improved and simplified in [5] and finally was integrated in the VVC standard [4].

There are several other approaches to intra prediction using neural networks. In [10], [11], the authors use neural networks to train one or two additional modes to the classical prediction. For training, they split the training set in blocks which would be predicted with angular prediction and those which would be predicted with DC or planar mode. That way, the authors were able to train two networks covering different signal characteristics.

In [12], Schiopu *et al.* propose several CNN-based modes specifically trained for lossless coding with HEVC. Different to other approaches, they use a large surrounding area as input instead of just one or a few lines. This enables the use of convolutional neural networks, while most other approaches use fully connected layers. The authors propose to use only their modes, dismissing the traditional modes. In a similar approach in [13], Meyer *et al.* use convolutional neural networks with fully connected layers at the end to propose new intra modes for both luma and chroma prediction, also including cross-component prediction

All of the above methods train multiple networks for multiple modes. Since the modes have to cover different signal characteristics, special attention has to be paid to the training procedure. On the one hand each network has to be specific enough to perform accurate prediction for the specific content. On the other hand, all networks together have to be broad enough to cover a large variety of content. Possible solutions for this problem are splitting the training set according to some criterion as in [11], or innovative training methods as in [5], training multiple networks jointly. In [14], we have proposed the conditional autoencoder for intra prediction, a novel method to generate an arbitrary number of modes using just one network. With this approach, instead of a prediction mode, we transmit a latent space representation which, in a very abstract way, contains instructions how to predict the block from its neighborhood. The network follows principles of an autoencoder [15], extended by additional side-information. Similar networks, using autoencoders with additional information added in various stages [16] were used in different research areas such as segmentation [17] or hand writing recognition [18].

Besides intra prediction, other coding tools can be improved by neural networks as well. In [19], Ding *et al.* proposed to use a convolutional neural network (CNN) for in loop filtering. The network structure the authors used is built upon the very deep super resolution (VDSR) network [20]. The trained network is applied on the decoded image in the loop filter stage. The authors report gains for a low bitrate scenario, in which the coding artifacts are particularly strong and can be removed by the trained filters.

In another instance [21], Laude *et al.* proposed to generate artificial reference pictures generated by neural networks for inter prediction. The artificial image replaces one reference frame for inter prediction.

Aside from improving coding tools within existing codecs, much research goes into end-to-end image compression. This was achieved by Ballé *et al.* in [22]. Here, the whole image was transformed into a latent space using a compressive autoencoder. An autoencoder is a two stage network consisting of an encoder and decoder network which are concatenated. Essentially, the network is trained to approximate the identity function, i.e., the output should be equal to the input. However, the network contains a so-called bottleneck, which is the layer between encoder and decoder network that has a lower dimensionality than the input and output. This layer therefore contains a dimensionality reduced representation of the input, called the latent space representation. With additional constraints to the latent space, compressibility of the latent space can be enforced and then be transmitted over the channel. In a subsequent publication [23], the authors have proposed a second stage to the autoencoder which allows them to compress and transmit priors for the latent space representation which are used in the arithmetic coding to further enhance the coding performance.

III. Conditional Autoencoder for Intra Prediction

A. General Concept

In intra prediction, the goal is to generate a good prediction signal $\hat{\boldsymbol{y}}$ for the content $\boldsymbol{y}$ of a block from the reference area $\boldsymbol{x}$, which is a causal spatial neighborhood of $\boldsymbol{y}$. Fig. 1 shows an example for a four pixel wide causal reference area. This problem has similarities to image inpainting, however with two major differences. First, the problem is harder since there is only data from one side available for inpainting. However, since the encoder knows the true content, the second difference is that we can use side-information to compensate the previous disadvantage by multi-mode prediction. Multi-mode prediction

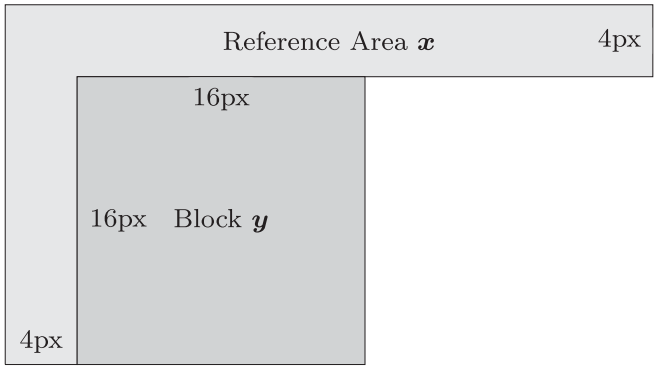

Fig. 1. Schematic showing a four pixel wide reference area $\boldsymbol{x}$ of the block $\boldsymbol{y}$.

allows us to use M different prediction functions $f_m(\cdot)$, where $m \in \{1, 2, \ldots, M\}$ denotes the index of the prediction mode. The coder tests different functions as predictor and signals the best to the decoder. Let us now assume, the mode m^* was picked as optimal and transmitted as side-information, we can formulate the prediction as follows:

$$\hat{\boldsymbol{y}} = f_{m^*}(\boldsymbol{x}) \tag{1}$$

Our goal is now to convert this equation into a form where we have only one trainable prediction function which still performs multi-mode prediction. First, we can rewrite this equation by using m^* as additional argument to the function instead of an index, yielding $f(m^*, \boldsymbol{x})$. This way, formally, we have one function. This function has an integer value m^* as input, so we can not compute gradients for this variable. Hence, instead of using a scalar value for the mode, we use the continuous vector $\boldsymbol{p}$. We can now write:

$$\hat{\boldsymbol{y}} = f(\boldsymbol{p}, \boldsymbol{x})\,. \tag{2}$$

The vector $\boldsymbol{p}$ contains information how to predict $\boldsymbol{y}$ from $\boldsymbol{x}$. We can therefore assume it depends only on the original block $\boldsymbol{y}$ and the reference area $\boldsymbol{x}$, as it describes a connection between both of them. Therefore we can define a function $g(\cdot)$ which computes $\boldsymbol{p}$:

$$\boldsymbol{p} = g(\boldsymbol{y}, \boldsymbol{x})\,. \tag{3}$$

Now, $g(\cdot)$ computes a vector describing how to perform the prediction from $\boldsymbol{x}$ to $\boldsymbol{y}$, before $f(\cdot)$ uses this information to perform the prediction. So far, both functions were only described in an abstract way.

Neural networks are known to be universal approximators [24], [25], meaning they can approximate any function arbitrarily close. We therefore can use them to approximate the functions $g(\cdot)$ and $f(\cdot)$. We model each function with a neural network consisting of seven fully connected layers. Since they have shown an efficient convergence behavior, we mainly use leaky rectified linear units (LReLU) [26] as activation functions ϕ:

$$\phi(v) = \begin{cases} 0.1 \cdot v & \text{if } v < 0 \\ v & \text{else} \end{cases} \tag{4}$$

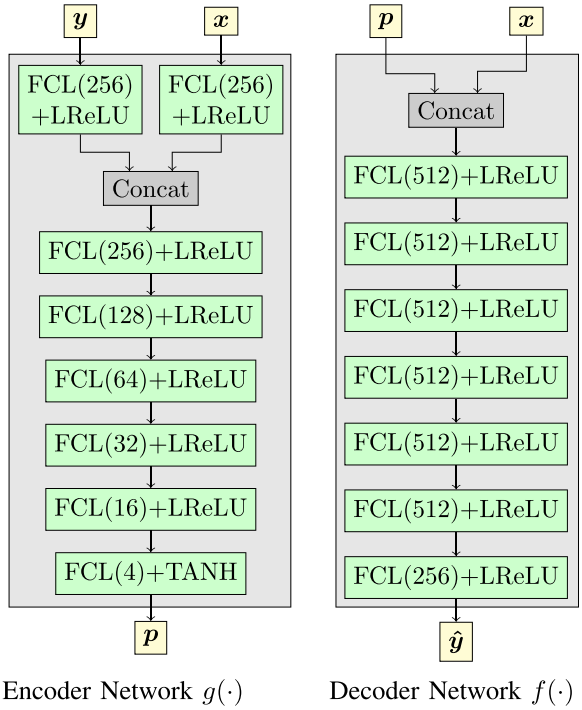

Fig. 2. Networks we use to model $g(\cdot)$ (left) and $f(\cdot)$ (right). FCL denotes a fully connected layer. The number of neurons is noted in braces. LReLU and TANH denote the leaky rectified linear unit and hyperbolic tangent activation function, respectively.

Only the last layer of $g(\cdot)$ has a hyperbolic tangent activation function to limit the elements of $\boldsymbol{p}$ between -1 and 1.

By modeling both functions as neural networks and concatenating the networks appropriately, we can train them jointly end-to-end. In Fig. 2, we show the networks we use to model both functions. In Fig. 3, the red underlaid area shows the concatenation of both networks. We see that the structure resembles the autoencoder structure. A signal $\boldsymbol{y}$ is compressed to a low-dimensional representation $\boldsymbol{p}$, before the second part of the network reconstructs the signal, yielding $\hat{\boldsymbol{y}}$. The crucial difference to a classical compressive autoencoder is that both the encoder and the decoder part know the spatial neighborhood $\boldsymbol{x}$. The compression and decompression is therefore performed under the condition of the known spatial neighborhood, hence the name *conditional autoencoder* (CAE).

We train the network with a loss function according to [5]. The function models the bitrate needed to transmit the residual. Using $\sigma(\cdot)$ as the sigmoid function, we can write the model as:

$$L_1 = \min_{t=1\ldots5} \sum_{m,n} |c_{m,n}^{(t)}| + \alpha_1 \sigma\Big(\alpha_2 |c_{m,n}^{(t)}| - \alpha_3\Big)\,, \tag{5}$$

where $c_{m,n}^{(t)}$ denotes the frequency coefficient at positions (m, n) of the residual block transformed with the t^{th} transform as used in VVC. The function tests all available transforms and picks one with the least estimated number of bits. As in VVC, we use DCT Type 2 in both directions as one possible transform and the four combinations of DCT Type 8 and DST Type 7 as the remaining four transforms. By this means we get as close to

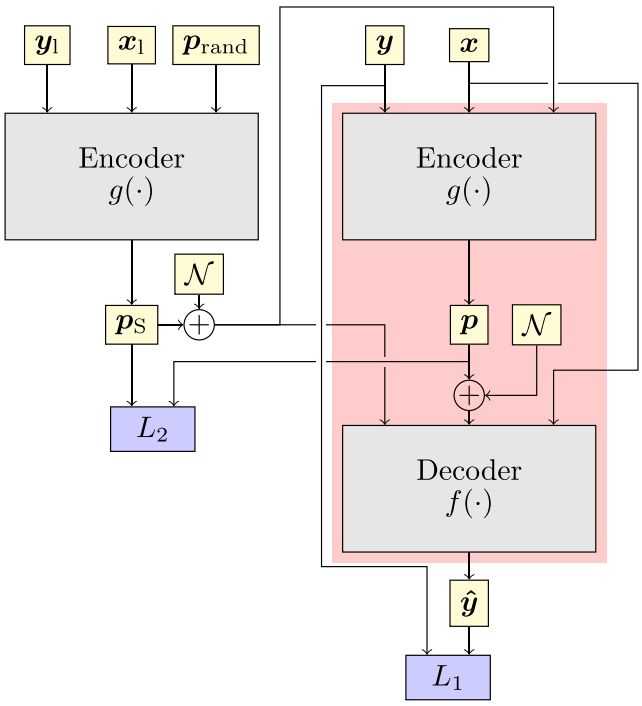

Fig. 3. Training setup for joint training of encoder and decoder. The red underlaid area shows the base network, the remainder extends the network with a spatially correlated latent space. For this schematic we use only one spatial neighbor which has the content $\boldsymbol{y}_\mathrm{l}$ and the support area $\boldsymbol{x}_\mathrm{l}$. In training both encoder networks share their weights to assure the same behavior. The input $\boldsymbol{p}_\mathrm{rand}$ denotes a randomly generated vector in the latent space and $\mathcal{N}$ denotes zero-mean white Gaussian noise with variance 0.1.

the coding procedure in VVC as possible. Here, $\alpha_{1,2,3}$ denote model parameters which are chosen such that the function is approximately proportional to the actual bitrate of a residual signal that is coded within VVC as measured on a separate training set.

The vector $\boldsymbol{p}$ has to be transmitted as side information similar to the prediction mode in classical prediction approaches. For transmission, we have to quantize this vector. We choose a trained Linde-Buzo-Gray vector quantizer [27] for that purpose. In the following, we denote the quantized latent space representation as $\hat{\boldsymbol{p}}$. For good performance, we need to take this into account during training. Since it is not possible to efficiently model an adaptively trained vector quantizer in a neural network, we approximate $\hat{\boldsymbol{p}}$ for the training by adding white Gaussian noise $\mathcal{N}(0, 0.1)$ with variance 0.1:

$$\hat{\boldsymbol{p}} \approx \boldsymbol{p} + \mathcal{N}(0, 0.1) \tag{6}$$

The number of quantization levels corresponds to the new number of modes. It is easy to see that this number can be chosen arbitrarily without changing the number of networks or the main training, solely by changing the vector quantizer. The strength of the noise was determined empirically. We need a certain amount of noise to simulate the large quantization we perform afterwards. However, if we add noise with too high variance, the initial convergence is inhibited. We therefore choose a variance of 0.1 as a compromise.

In [14], we furthermore have shown that the prediction quality can be increased by performing a second training, which refines only the decoder network and accustoms it to vector-quantized inputs. That way, we can compensate the lower variance of the simulated quantization noise. Optimally, this training has to be performed for each number of modes. This kind of training is possible since we only have a small amount of possible values. In the following we set the number of quantization levels $q = 64$. The quantization also determines the size of the latent space, i.e. the dimensionality of the vector $\boldsymbol{p}$. On the one hand, a larger latent space yields a better representation of the block and allows more flexibility in training. On the other hand, a higher dimensionality also increases the error made during quantization. In preliminary experiments we found a latent space size of 4 suitable. In the context of CAE, the term mode will from now on represent a quantized latent space representation.

B. Spatial Mode Prediction

Intra prediction exploits also larger scale correlation by spatially predicting the intra mode itself. This is possible since the classical intra-prediction modes have very generic properties, describing a direction of the block content. The modes generated by the CAE do not have that property, they are therefore difficult to predict over different blocks. As further described in [28], to make mode prediction possible, we train the network only for 16×16 blocks and rescale the other blocks accordingly, such that all block sizes are predicted with the same model. That way, we do not have to take different models for each block size into account, which greatly simplifies mode prediction across different block sizes.

As a second step, we define a spatial neighborhood in the latent space $\boldsymbol{p}_\mathrm{S} = [\boldsymbol{p}_{\mathrm{s},1} \dots \boldsymbol{p}_{\mathrm{s},n} \dots \boldsymbol{p}_{\mathrm{s},N}]$, consisting of the latent spaces $\boldsymbol{p}_{\mathrm{s},n}$ of N spatial, causal neighboring blocks. If one or more blocks are not available due to boundary effects or coder constraints, they are copied from those which are available. If none is available, all neighbors are set to the representation with the highest a-priori probability. We now want to use $\boldsymbol{p}_\mathrm{S}$ to construct a most probable mode list. However, predicting the latent space of the current block from the latent space of its spatial neighbors is only possible if neighboring blocks yield similar latent spaces. We therefore add another loss function:

$$L_2 = \sum_{n=1}^{N} \left\| \boldsymbol{p} - \boldsymbol{p}_{\mathrm{s},n} \right\|_2 , \tag{7}$$

with the ℓ_2 norm $\| \cdot \|_2$. This loss function enforces the latent space of adjacent blocks to be similar. However, this is a strong additional constraint on the network and decreases the performance. We therefore use $\hat{\boldsymbol{p}}_\mathrm{S}$ as additional input to the encoder and decoder network, conveying additional information. We hence arrive at the following equations:

$$\boldsymbol{p} = g(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{p}_\mathrm{S}) \tag{8}$$

$$\hat{\boldsymbol{y}} = f(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{p}_\mathrm{S}) . \tag{9}$$

We do not need to signal this information since it is known at the decoder. Fig. 3 shows the setup for the training. We train

the network using weight-sharing between both instances of the encoder network. For sake of simplicity, in the figure, we set $N = 1$ and only use the block to the left as reference.

Having trained an autoencoder with such correlated latent space, we can define an elegant prediction scheme, similar to the most-probable-mode (MPM) list, already in use in VVC [3]. We define an MPM list with six elements according to the following distance measure $d(\hat{\boldsymbol{p}})$ representing the inverse probability:

$$d(\hat{\boldsymbol{p}}) = \min_{n=1\ldots N} \left[\left\| \hat{\boldsymbol{p}} - \hat{\boldsymbol{p}}_{\mathrm{s},n} \right\|_2 \right] \tag{10}$$

A latent space representation that is close, in the sense of euclidean distance, to one of the neighboring representations is therefore considered likely. Note that this definition sets the reference modes themselves to the maximal probability. The six elements with smallest distance are picked for the MPM list. If two modes have the same distance measure, the one with the smaller index is preferred. We order the modes, such that modes with higher prior probability—measured on the training set—have a smaller index.

The algorithms from the two previous sections can now be integrated in the coding software. For that purpose, we replaced Matrix Intra Prediction (MIP) with CAE-based prediction functions. The proposed CAE-based prediction modes therefore exists in parallel to classical prediction. Since we can only transmit a quantized latent space, we need to make sure, that the best quantization point close to the non-quantized value is transmitted. For that purpose, we use rate-distortion-optimization to choose between the five quantization points which are closest to the ideal $\boldsymbol{p}$. As for MIP, first a flag is transmitted, whether classical prediction or CAE-based prediction is used. Afterwards, the mode is transmitted using the MPM list. In [28], this method is described in more detail and examined extensively.

C. Prediction of the Chroma Components

The algorithms in the previous sections were all concerned with predicting the luma component. However, the prediction of the chroma components is also important for the coding performance. The chroma component also strongly affects visual quality, since color artifacts can cause irritable effects. Since the chroma components are transmitted after the luma component, and both components are locally correlated to some degree, it is possible and sensible to use the reconstructed luma component for prediction. To integrate a chroma component prediction into the CAE-based approach, we need to extend our model.

A straightforward approach would be to use both an encoding and a decoding network on the chroma components and transmit a new latent space representation. We could train a new CAE in a similar way on the chroma component for that purpose. However, VVC only uses very little rate to signal the chroma prediction mode, since it is picked from a short list. In order to be compatible, we therefore would have to quantize the latent space very coarsely, leading to bad results.

We therefore choose to use the same latent space representation as in the luma component and train an additional network to predict the chroma components. In the following, let $\boldsymbol{y}_{\mathrm{u}}$ and $\boldsymbol{y}_{\mathrm{v}}$ denote the original block in the u and v component. Furthermore,

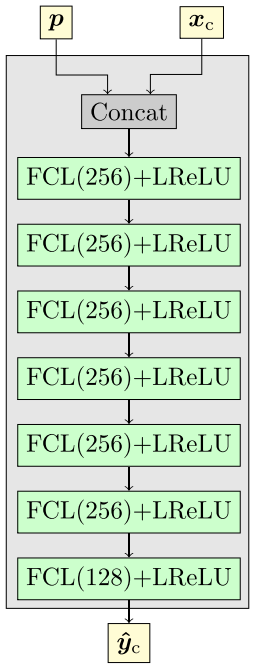

Fig. 4. Network to model $f_{\mathrm{c}}(\cdot)$. The notation is equivalent to Fig. 2.

since we perform chroma component prediction jointly for both components, let $\boldsymbol{y}_{\mathrm{c}} = [\boldsymbol{y}_{\mathrm{u}}, \boldsymbol{y}_{\mathrm{v}}]$ denote the concatenation of both components. For $\hat{\boldsymbol{y}}$ and $\boldsymbol{x}$, we use an equivalent notation. We can now define the chroma prediction function f_{c} as follows:

$$\hat{\boldsymbol{y}}_{\mathrm{c}} = f_{\mathrm{c}}\left(\boldsymbol{p}, \boldsymbol{x}_{\mathrm{c}}\right), \tag{11}$$

with the latent space representation $\boldsymbol{p}$ from the luma component. Again, we model $f_{\mathrm{c}}(\cdot)$ as a neural network comprising fully connected layers. Fig. 4 shows the exact structure. There are two possibilities to train the system, now. Since $\boldsymbol{p}$ is shared, we can train luma and chroma component jointly by adding the chroma decoder to the setup in Fig. 3. However, we found that this degrades the prediction quality of the luma component. Instead, we compute $\boldsymbol{p}$ for each luma block in the training set and train $f_{\mathrm{c}}(\cdot)$ by itself. This has the advantage that we can directly train on quantized latent space representations. We train the network on a loss function $L_3 = L_3^{(\mathrm{u})} + L_3^{(\mathrm{v})}$ which is similar to the luma loss function L_1 but only takes the coefficients $c_{m,n}^{(\mathrm{u/v})}$ of the DCT Type 2 transform into account, since VVC only uses this transform for chroma residual coding [3]:

$$L_3^{(\mathrm{u})} = \sum_{m,n} |c_{m,n}^{(\mathrm{u})}| + \beta_1 \sigma\Big(\beta_2 |c_{m,n}^{(\mathrm{u})}| - \beta_3\Big). \tag{12}$$

Here, $c_{m,n}^{(\mathrm{u})}$ is the DCT Type 2 coefficient of the residual U-component at position (m, n). We compute $L_3^{(\mathrm{v})}$ in the same way and train on the sum of both functions. Both functions share the model parameters $\beta_{1,2,3}$, which are chosen for the model to match the number of bits needed for transmitting the residual. We optimize them on a separate training set. We train the network only on a block size of 8×8 and rescale the inputs and outputs accordingly. We choose a smaller size than for the

luma component to take the typical downsampling of the chroma components into account.

Since prediction in the chroma component does not allow using as much side information as needed to transmit the latent space, we have to derive $\boldsymbol{p}$ from the luma component. We test two possibilities here. First, we use the mechanism already in place which derives a possible chroma prediction mode from the co-located luma block. In a straightforward approach, if CAE was used for luma prediction, we use the same quantized latent space representation in the chroma component as used in the luma component. However, a CAE-based mode describes the block content in a very specific way, so this principle only works if chroma and luma block are at matching positions. However, one new feature in VVC is the Chroma Separate Tree (CST), which allows to generate separate block partitioning trees for luma and chroma. Using this feature—different to HEVC—co-located chroma and luma blocks may not be of the same size (taking chroma subsampling into account) and can therefore not share a latent space representation. We therefore apply this method only to the chroma blocks for which a luma block of same size and position exists. For all other cases, if the derived mode is a CAE-based mode, we take the planar mode instead. For this method we do not need to change anything in the mode signaling, as the CAE-based chroma prediction exists as derived mode. However, first experiments showed that this method does not perform well, mainly due to the limitation caused by CST. We call this model CAE with derived chroma prediction.

Since the main issue with the previous method is that there are not suitable latent space representations available for many blocks, we can use the reconstructed luma signal $\tilde{\boldsymbol{y}}$ to generate a latent space representation $\tilde{\boldsymbol{p}}$ of the corresponding area. We then use this vector as input to the chroma prediction function:

$$\hat{\boldsymbol{y}}_{\mathrm{c}} = f_{\mathrm{c}}\left(\tilde{\boldsymbol{p}}, \boldsymbol{x}_{\mathrm{c}}\right), \tag{13}$$

with

$$\tilde{\boldsymbol{p}} = g(\tilde{\boldsymbol{y}}, \boldsymbol{x}). \tag{14}$$

The decoder can execute both parts of the network here, since the encoder network now gets the reconstructed luma block as input. We do not need to transmit any latent space representation. This has the advantage that we can use $\tilde{\boldsymbol{p}}$ directly without having to deal with the quantization error. On the other hand, $\tilde{\boldsymbol{p}}$ is less accurate than $\boldsymbol{p}$, since it is based on the reconstructed signal instead of the original signal.

In the coder, we implement this strategy as additional mode for chroma prediction, which is signaled as separate flag, after cross component linear model (CCLM) and the derived prediction mode. Conceptually, this prediction has some similarities to CCLM, since it bases the prediction of the chroma component on the reconstructed signal in the luma component. However, instead of applying a linear model, we extract a small dimensional vector as kind of blueprint, in which way prediction has to be performed. Different to CCLM, the actual prediction function then does not use the reconstructed signal. This is useful, if the linear model does not hold for the whole block, as we will see later on. We now can allow chroma CAE prediction for all blocks, independently of the chosen mode in the luma component. In the following evaluation, we will refer to this chroma prediction mode as cross component conditional autoencoder (CC-CAE).

In both methods, we have to deal with a degradation of the latent space. If we use the first method, we use a quantized latent space representation which was computed on the original signal. On the other hand, we do not need to quantize the latent space in the second method but the latent space describes the reconstructed and therefore distorted signal instead of the original.

IV. Performance Evaluation

A. Training and Setup

As recommended in [29], we use the DIV2K training set for training all models. Since in real applications, the reference area is taken only from the reconstructed signal, we use coded images to extract all reference areas and uncoded images for the target patches. We code each image with a random QP using HEVC and extract blocks of size 16×16. To take into account the possibility of dealing with down-sampled patches, we use each image several times in downscaled versions.

We implement the network in the caffe [30] environment and train the networks using the Adam optimizer [31] with standard parameters and learning rate 10^{-4}. For training the luma predictors, we combine L_1 and L_2 to the overall loss function L by

$$L = L_1 + \lambda L_2, \tag{15}$$

with $\lambda = 10$. We picked the optimal value for λ and the variance of the simulated quantization noise based on preliminary tests on the DIV2K validation set.

First experiments training the networks showed that a pre-trained network yields a better performance. We therefore perform a training where we jointly optimize on all different transforms instead of just the best one. This means that we replace the minimum operator in (5) by a sum. We use the obtained weights to initialize the network. This is only necessary for the luma network. Since the chroma network only uses one transform, this step is skipped.

For the experiments, we implemented the prediction scheme into the VVC test model (VTM) version 6.2. We use the same method as MIP to signal the use of CAE-based modes. In our experiments, CAE and MIP are exclusive, so MIP is not used together with CAE. We perform our tests on 10 frames of each sequence from the JVET test set [32] with the All Intra (AI) profile. In accordance with the JVET common test conditions [32], we only code every eighth frame to gain a larger diversity of tested content.

B. Visual Examples

Before demonstrating the performance of our prediction scheme in terms of rate-distortion behavior, we first visually show the prediction performance in different scenarios. For this purpose, we configured the coder to use only blocks of size 16×16. For the following examples we compare the traditional modes, MIP and our proposed CAE. We only show blocks,

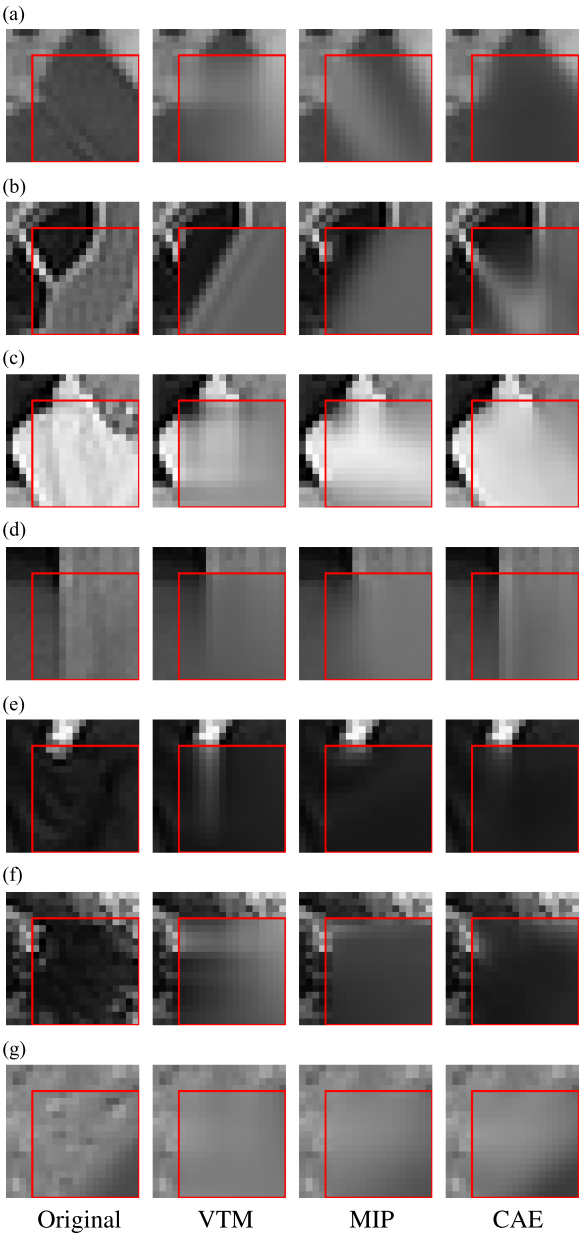

Fig. 5. Examples of blocks predicted with different methods. The left column shows the original block. The second column, labeled VTM, shows the best traditional mode. The third column, MIP, shows the prediction with MIP and the right column shows the predicted signal of CAE. The red frame shows the predicted block. Outside of the frame, we show the reconstructed signal from which the reference signal originates.

where MIP and CAE were selected in the respective coders. Consequently, in each shown example, MIP and CAE both perform better than the traditional modes, which are denoted with VTM in the figure. At first we analyze the luma signal more closely. To that end, we selected blocks, in which the difference between the predictors are more visible.

In many cases, it is noticeable that the CAE is able to predict in multiple directions at the same time. We show examples of this property in Fig. 5(a)–(c). In Fig. 5(a), we see that the upper left edge should be predicted from top right to bottom left and the upper right edge in the other direction. Since this is not possible with traditional prediction, VTM chooses to use the planar mode. MIP is performing something similar to an angular prediction, causing a large error in the center of the block. CAE, on the other hand, is able to accurately estimate both edges while keeping the remainder of the block clean.

Similarly, in Fig. 5(b), VTM and MIP both ignore the thin line on the left side by performing prediction mainly from one direction. CAE is able to predict both edges, however, the result is less sharp than for the traditional modes. We also see that the initially vertical direction of the upper edge is kept. In Fig. 5(c), the chosen planar mode is not able to perform a good prediction, while MIP has errors on the bottom of the block. CAE produces sharp and correct edges except for the upper right corner, where the edge becomes blurry.

Fig. 5(d) shows that CAE is able to preserve sharp edges better than MIP while remaining more flexible than the traditional modes. We see that VTM chose planar mode here instead of the vertical mode. This is probably due to the fact that the vertical prediction mode would fill the left part of the block with black due to the black pixels in the reference area. With the planar mode this problem does not occur, however, the edge gets less sharp. Here, MIP and the planar mode perform similarly. We see that CAE is able to obtain the sharp edge while still choosing the correct color in the left part, accurately following the gradient.

Fig. 5(e) shows an advantage of MIP and CAE compared to traditional prediction when it comes to structures that stop at a certain point in the block to be predicted. We see that VTM chooses the planar mode again, yielding a white structure reaching far in the block. MIP and CAE manage to stop the white structure from progressing too far into the block while maintaining the black background. In Fig. 5(f), we see a similar effect, however, here MIP has problems keeping the background dark, probably due to the larger amount of bright pixels in the reference area, while CAE is able to maintain the dark background. None of the prediction algorithms is able to predict the brighter area in the bottom right corner.

In Fig. 5(g), we see that, in a limited way, CAE is able to predict content that can not be seen in the reference area, such as the black area at the lower right corner. This content is less sharp than other predictions. However, in this block, we see that the prediction follows the original structure.

With the examples from Fig. 5 as described above, we see that CAE-based prediction has a great flexibility and can predict a wide variety of content, even though we only use 64 quantization points in the latent space, which is very similar to the number of traditional modes.

It is more difficult to find visual examples for the performance in the chroma domain since the content carries much less information. Therefore the differences in the prediction signal are often only very subtle. For that reason, we increased the contrast of the following examples to improve the visual impression.

Fig. 6 shows selected blocks where the difference in the signals is visible. For each block and chroma component, we show the original block, and the signal coded by standard VTM and our new method CC-CAE. Additionally, we show the co-located reconstructed luma component which is used in CCLM for each block.

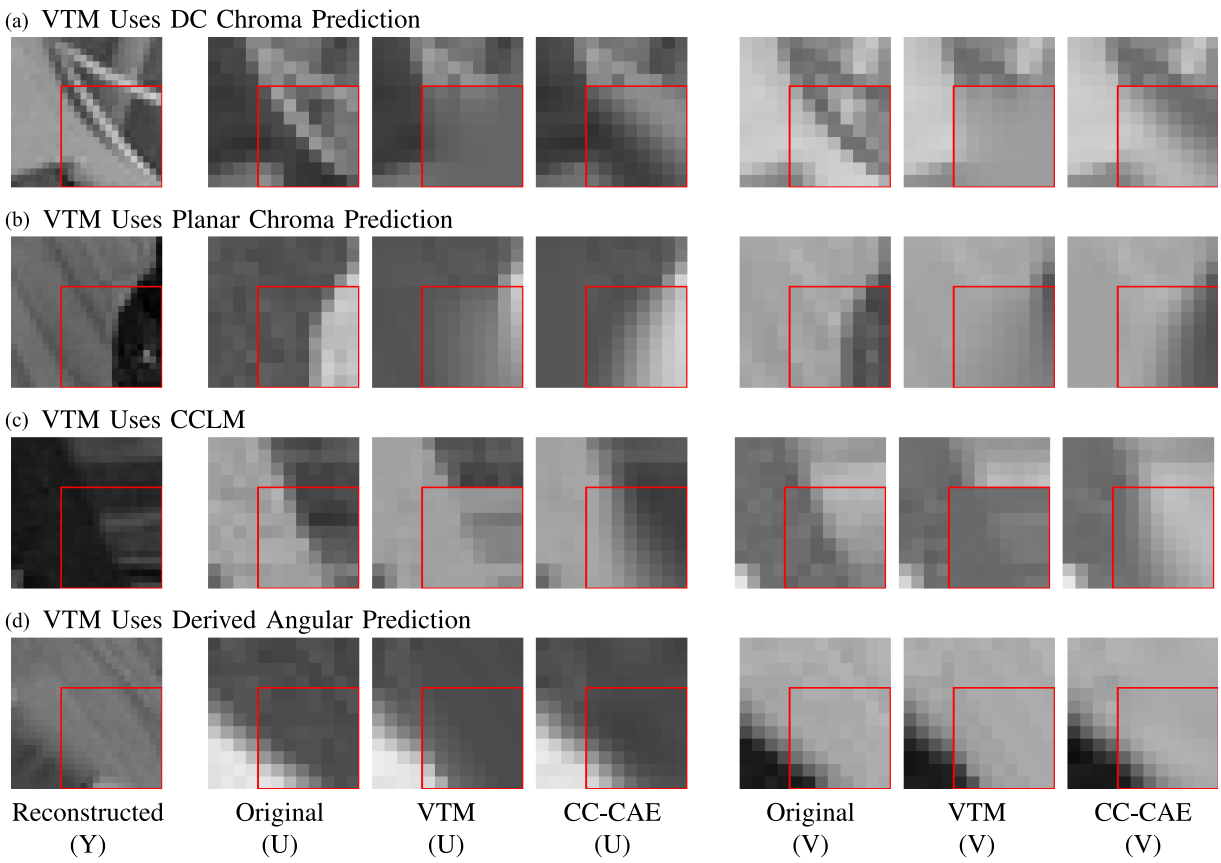

Fig. 6. Examples of chroma blocks which were predicted with traditional methods in VTM and with CC-CAE. We show the original and predicted signals for both chroma components, as well as the co-located reconstructed luma signal which serves as reference for CCLM. The mode which was used by the original VTM to predict the chroma component is given in the sub captions.

In Fig. 6(a), we see a block in which CCLM would fail, since both parts of the block follow a different linear model. VTM therefore chooses the DC mode for prediction which does not capture the diagonal structure. Note that a diagonal mode would probably work well in this case, however since chroma modes are only selected from a very small list, this mode is not available here. We see that CC-CAE generates the diagonal structure in the upper right triangle however the signal looses in sharpness. Also, we see that in the V-component, the edge is slightly shifted. However in both components, the lower left corner is predicted accurately.

In Fig. 6(b), the block consists of a curved structure which traditional prediction can not predict. As in the luma component, CC-CAE is able to follow the structure to some extend, here the V-component shows a better result than the U-component, following the structure more accurately.

In Fig. 6(c), we compare our method with a block that was predicted with CCLM in VTM. Here we see that the bright structures in the luma block cause small disturbances in the prediction signal of the chroma component which are not present in the original. Furthermore, we can see that the linear model is not valid for the right part and therefore generates wrong prediction values. On the other hand, CC-CAE uses more abstract information from the luma component. We only transfer the knowledge, how to predict the block from the reference area. CC-CAE therefore continues the upper right part correctly into the block. Similarly to before, angular prediction would have given better results than CCLM but was not available.

Fig. 6(d) shows that CC-CAE can also improve angular prediction. This example was predicted by VTM using a derived angular mode. However, the used mode yields a prediction signal that is too steep to match the original. CC-CAE produces a signal with the correct angle.

Altogether, we see that CC-CAE is also very flexible in the chroma component. However, compared to the luma component, the prediction signal looses sharpness. This may be due to the fact that the latent space is produced from the luma component only so some uncertainty remains in the model. However, we see that also visually CC-CAE performs better than other available prediction modes.

C. Results

After showing that CAE is able to predict many different kinds of content, we now want to demonstrate the performance of CAE in the coding software. In Table I, we show the results for different learning-based approaches. We report the Bjøntegaard delta rate (BD-rate) [33] for each component separately. We report gains compared to VTM 6.2 using only traditional modes with MIP turned off.

TABLE I
RESULTS COMPARING DIFFERENT LEARNING-BASED METHODS REGARDING THEIR RATE-DISTORTION PERFORMANCE. ALL RESULTS ARE BD-RATES IN %. THE REFERENCE IS VTM 6.2 WITH STANDARD CONFIGURATION AND MIP TURNED OFF

Luma Prediction Chroma Prediction	MIP [4] Planar			CAE Planar			CAE Derived			CAE CC-CAE		
Channel	Y	U	V	Y	U	V	Y	U	V	Y	U	V
Tango2	-1.08	-0.44	-0.51	-2.08	-0.68	-1.34	-2.07	-1.26	-1.41	**-2.20**	**-3.13**	**-1.92**
FoodMarket4	-1.62	-0.78	-0.83	-3.56	-2.11	-2.10	-3.52	-2.54	-2.34	**-3.62**	**-2.84**	**-2.70**
Campfire	-0.56	-0.20	-0.11	**-0.83**	-0.40	+0.05	-0.83	**-0.44**	-0.03	-0.82	-0.33	**-0.26**
Class A1	-1.09	-0.47	-0.48	-2.16	-1.06	-1.13	-2.14	-1.42	-1.26	**-2.22**	**-2.10**	**-1.63**
CatRobot1	-0.57	-0.08	-0.20	-1.09	-0.63	-0.60	-1.13	-0.81	-0.67	**-1.20**	**-1.65**	**-1.24**
DaylightRoad2	-0.37	+0.03	-0.00	-0.80	-0.17	+0.02	**-0.80**	-0.62	-0.19	-0.80	**-1.63**	**-0.84**
ParkRunning3	-0.50	-0.18	-0.17	-0.99	-0.60	-0.64	-1.00	-0.71	-0.70	**-1.05**	**-0.94**	**-1.01**
Class A2	-0.48	-0.08	-0.12	-0.96	-0.47	-0.41	-0.98	-0.71	-0.52	**-1.02**	**-1.41**	**-1.03**
MarketPlace	-0.67	-0.30	-0.47	-1.19	-0.82	-0.86	-1.18	-0.58	-1.08	**-1.22**	**-0.93**	**-1.37**
RitualDance	-0.42	+0.20	+0.16	**-1.25**	-0.55	-0.96	-1.25	-0.84	-1.10	-1.21	**-0.92**	**-1.15**
Cactus	-0.48	-0.02	+0.09	-0.85	-0.47	-0.29	-0.87	-0.61	-0.38	**-0.96**	**-1.06**	**-0.98**
BasketballDrive	-0.39	-0.52	+0.15	-0.82	-0.44	-0.10	-0.89	-0.48	-0.17	**-0.91**	**-1.16**	**-1.08**
Kimono	-1.11	-0.88	-0.84	-1.91	-2.19	**-2.12**	-1.89	-2.14	-2.00	**-1.93**	**-2.57**	-2.04
BQTerrace	-0.29	-0.03	+0.26	-0.68	-0.22	+0.12	-0.67	-0.11	-0.11	**-0.70**	**-0.50**	**-0.49**
Class B	-0.56	-0.26	-0.11	-1.12	-0.78	-0.70	-1.12	-0.79	-0.81	**-1.16**	**-1.19**	**-1.18**
RaceHorses	-0.44	+0.08	-0.33	-0.78	-0.32	-0.29	-0.78	-0.45	-0.51	**-0.87**	**-0.95**	**-0.64**
BQMall	-0.50	-0.27	-0.50	-0.73	-0.40	-0.64	-0.74	-0.31	**-1.04**	**-0.79**	**-0.96**	-1.00
PartyScene	-0.57	+0.02	-0.27	-0.62	-0.07	-0.18	-0.57	-0.28	**-0.58**	**-0.62**	**-0.56**	-0.51
BasketballDrill	-0.42	-0.03	+0.17	-0.60	-0.26	-0.35	-0.62	+0.02	-0.34	**-0.78**	**-0.54**	**-0.84**
Class C	-0.48	-0.05	-0.23	-0.68	-0.26	-0.37	-0.68	-0.25	-0.62	**-0.77**	**-0.75**	**-0.75**
RaceHorses	-0.48	-0.12	+0.20	-0.92	-0.53	-0.58	-0.92	-0.55	-0.82	**-0.98**	**-1.21**	**-0.85**
BQSquare	**-0.74**	-0.22	+0.22	-0.52	-0.30	-0.03	-0.54	-0.13	+0.01	-0.52	**-0.75**	**-0.16**
BlowingBubbles	-0.48	-0.28	-0.68	**-0.73**	-1.01	-0.58	-0.68	**-1.30**	-0.66	-0.71	-1.03	**-1.14**
BasketballPass	-0.29	-0.55	-0.51	-0.39	-0.01	-1.17	**-0.55**	-0.18	-0.49	-0.37	**-1.04**	**-2.06**
Class D	-0.50	-0.29	-0.19	-0.64	-0.46	-0.59	**-0.67**	-0.54	-0.49	-0.65	**-1.01**	**-1.05**
FourPeople	-0.43	-0.10	-0.18	**-1.18**	-0.64	-0.50	-1.16	-0.90	-0.70	-1.15	**-1.16**	**-1.46**
Johnny	-0.53	-0.68	-0.95	-1.41	-1.13	-0.81	-1.42	-1.19	-1.61	**-1.45**	**-1.56**	**-1.75**
KristenAndSara	-0.46	-0.84	+0.59	-1.10	-0.75	+0.19	**-1.10**	-0.68	-0.11	-1.07	**-1.65**	**-1.12**
Class E	-0.47	-0.54	-0.18	-1.23	-0.84	-0.37	**-1.23**	-0.92	-0.81	-1.22	**-1.46**	**-1.44**
Average	-0.58	-0.27	-0.20	-1.09	-0.64	-0.60	-1.10	-0.74	-0.74	**-1.13**	**-1.26**	**-1.16**

At first, we only evaluate the performance on the luma component. Turning on MIP yields rate savings of 0.58%. We see that all versions of CAE-based prediction outperform MIP by an average of at least 0.51%. In the best case, using CC-CAE for chroma prediction, we save 1.13% rate over the baseline, which is 0.55% more than MIP. We observe that all three chroma prediction schemes show very similar behavior in the Y-component. This is no surprising result, since the chroma prediction does not directly influence the Y-component. CAE performs particularly well for classes with high resolution. For class A1, we measure a gain of 1.19% against MIP, saving 2.22% compared to the baseline. We see that only for the the BQSquare sequence, MIP outperforms all CAE-based methods. Interestingly, this due to both a good performance of MIP (the best of all low resolution sequences) and a bad performance of CAE (the second worst of all sequences). For all other sequences, CAE outperforms MIP. With our method, we almost double the rate-savings of MIP. The gains we achieve over MIP are considerable and comparable to the gains of other intra-coding tools newly introduced in VVC, like multi-reference line prediction or intra sub-partitioning [34]. Furthermore, we found that the gain is present at all evaluated rate points.

To analyze the results further, we show the relative occurrence of certain groups of prediction modes in Fig. 7. Here, we distinguish between planar and DC mode and group all angular and all CAE-based modes together. We see that on average, CAE-based modes occur in more than 40% of the area and more often than all angular modes combined. Individual sequences like Kimono consist to almost 75% of CAE blocks. In Fig. 7(b), we show how the mode distribution changes when CAE is turned on compared to VTM only using traditional prediction modes. We see that mainly blocks which previously were predicted with the planar mode are selected for CAE-based prediction. This is expected, since the planar mode is often chosen when the angular models are not sufficient, which is the intended use case for CAE-based prediction. In most frames, also some angular modes are replaced by CAE-based modes, showing again the flexibility of CAE-based modes. At this point, we want to emphasize that we achieve this flexibility without explicitly training the network for certain structures.

If we compare the BD-rates of the chroma components, we observe larger differences between the CAE chroma prediction methods. First, we note that a planar chroma prediction is sufficient to achieve better compression in the chroma components compared to MIP. However, in some sequences, such as Campfire, the results are actually worse than the baseline, having a positive BD-rate. A possible explanation lies in the way the chroma prediction mode is chosen and signaled in VVC.

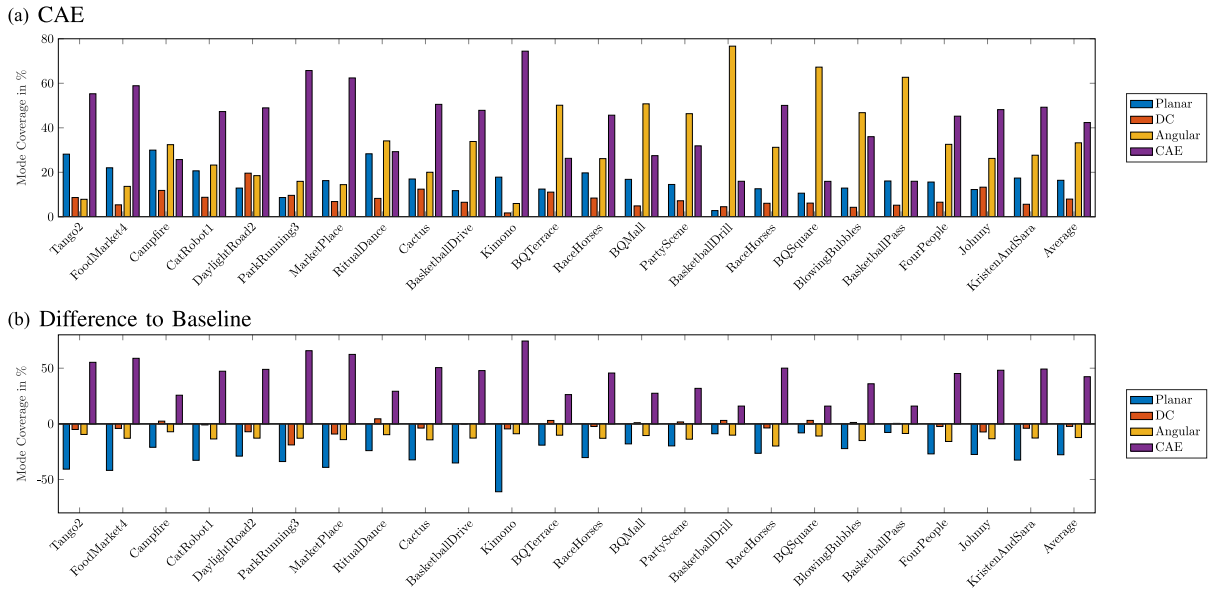

Fig. 7. Percentage of the area predicted with the indicated group of modes for the luma component. We separate between the individual modes planar and DC and all angular as well as CAE-based modes together. (a) shows the coverage for CC-CAE while (b) shows the difference in coverage compared to the baseline.

In VVC, there are eight modes which are tested, three cross component prediction modes, four fixed traditional modes, horizontal, vertical, planar, and DC and the derived mode, which is usually the same as the co-located luma mode. Since CAE yields a very flexible and powerful prediction, occupying between 30% and 50% of the image, many blocks which were predicted with an angular mode are now predicted with CAE. Some of these blocks have very strong edges, that can not be properly predicted with DC or planar mode. Since the derived mode is now set to planar instead of the appropriate angular mode, it is no longer possible to predict the chroma block by this angular mode and the remaining modes are not sufficient.

It is therefore important to allow CAE modes also in the chroma component. As discussed before, we have two possibilities. First we can use derived CAE prediction, i.e., we use the same latent space as in the luma component. We see that this approach does not yield good results. We gain only about 0.1% compared to the previous method using planar prediction. A major reason is the limited possibility for derived CAE-modes. As discussed above, a derived CAE-mode is only sensible if the chroma block is exactly aligned with the co-located luma block. As another possibility, we can use a latent space representation derived from the reconstructed luma signal (CC-CAE). Table I shows that this chroma prediction methods gives large gains over MIP of almost 1% on average in the chroma components. We see that in the majority of cases, CC-CAE yields the best result among all tested algorithms. Even in the sequence DaylightRoad2, which suffered losses in the V-component with planar chroma prediction, we are now able to save 0.84% in BD-rate. Similarly to the luma component, all rate points profited from the use of CAE-based chroma-prediction.

Similarly as above, Fig. 8 shows the distribution of chroma prediction modes. We grouped the modes into the derived mode from luma, all cross component linear model (CCLM) modes, the default modes (DC, Planar and selected angular modes) and the proposed CC-CAE mode. We see that the new chroma mode is chosen for about 20% of the total area which is less than for the luma component. We see that CCLM is picked in most cases, often providing a stronger prediction scheme to compete against than in the luma component. Also due to the picked signaling method, signaling CC-CAE costs two bit per block more than CCLM. However, we see in Fig. 8(b) that the amount of blocks predicted with CCLM is reduced for almost all frames. Furthermore, the test shows that the gains of CC-CAE arise from replacing all three groups of modes in a similar amount. This is different from the luma component, where the preferred mode to be replaced by CAE was planar.

TABLE II
ENCODER RUNTIME OF THE CURRENT IMPLEMENTATION COMPARED TO VTM WITH MIP TURNED OFF

Luma Prediction Chroma Prediction	MIP [4] Planar	CAE Planar	CAE Derived	CAE CC-CAE
Class A1	114%	780%	783%	820%
Class A2	112%	807%	810%	881%
Class B	114%	857%	861%	907%
Class C	116%	822%	827%	903%
Class D	114%	836%	848%	903%
Class F	115%	913%	894%	921%
Average	114%	838%	840%	893%

D. Complexity

Our method has a larger complexity than MIP. Our experiments show that CAE-based methods take about eight to nine times the time of VTM without MIP. Table II shows the runtime of our coder relative to VTM with MIP disabled. The experiments were performed on a Intel Xeon 3.30 GHz processor

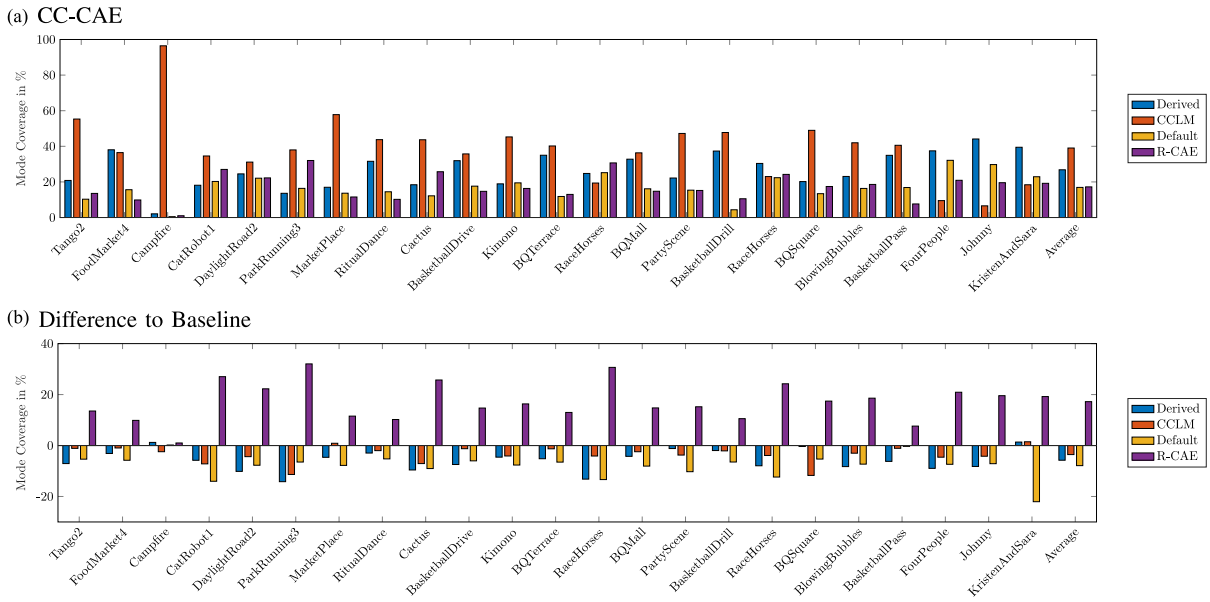

Fig. 8. Percentage of the area predicted with the indicated group of modes for the luma component. We separate between the individual modes planar and DC and all angular as well as CAE-based modes together. (a) shows the coverage for CC-CAE while (b) shows the difference in coverage compared to the baseline.

and a GeForce GTX 1060 GPU. However, these measurements include the time transferring the data to the GPU and back. Empirical measurements indicate that in our case, since we use a small network, the data transfer is responsible for about 80% of total runtime of the prediction. By using different architectures that require less overhead there is an immense potential for reducing the runtime. Also the network itself has not been optimized for inference runtime yet. There are several well-known methods to reduce a the inference time for neural networks, for example network pruning [35] or quantizing the network coefficients to integer values [36].

V. Conclusion

In this paper we introduced novel concepts for intra prediction for all components. We showed that the concept of the conditional autoencoder for intra prediction can be extended to the chroma components in several ways. It is possible to treat the CAE as a derived prediction mode and use the same latent space representation as for the luma component. This however comes with the drawback that due to Chroma Separate Tree (CST) the derived mode can not be used for all blocks. We show that the better alternative is to treat CAE as cross component prediction, using latent space representations which were computed from the reconstructed luma component.

Furthermore, with this publication we propose a new prediction system for all components building upon our previous work. Using a spatially correlated latent space and the cross-component CAE chroma extensions, we save more than 0.5% rate in the luma and about 1% in the chroma components compared to current technology. In this paper we proved, that the conditional autoencoder for intra prediction, which we proposed in [14] is able to form a fully functional intra prediction system for all components, outperforming state-of-the-art methods in terms of Bjøntegaard delta rate savings.

In this paper, we demonstrated how the concept of a conditional autoencoder can be employed in the classical hybrid video coder in different ways. However, the task of transmitting information given relevant known signals is very common in video coding problems. We therefore see great potential of applying the concept on further points in hybrid video coders, as well as in possible future end-to-end optimized video coding technologies.

References

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

[2] B. Bross, J. Chen, and S. Liu, "Versatile video coding (Draft 6), JVET-O2001-v14," in *Proc. 15th Meeting Joint Video Explor. Team*, Jul. 2019, pp. 1–441.

[3] J. Chen, Y. Ye, and S. Kim, "Algorithm description for versatile video coding and test model 6 (VTM 6), JVET-O2002-v2," in *Proc. 15th Meeting Joint Video Explor. Team*, Jul. 2019, pp. 1–92.

[4] J. Pfaf *et al.*, "CE3: Affine linear weighted intra prediction (CE3-4.1, CE3-4.2) , JVET-N0217-v1," in *Proc. 14th Meeting Joint Video Explor. Team*, Mar. 2019, pp. 1–18.

[5] P. Helle, J. Pfaff, M. Schäfer, R. Rischke, H. Schwarz, D. Marpe, and T. Wiegand, "Intra picture prediction for video coding with neural networks," in *Proc. Data Compression Conf*, Mar. 2019, pp. 448–457.

[6] K. Zhang, J. Chen, L. Zhang, X. Li, and M. Karczewicz, "Enhanced cross-component linear model for chroma intra-prediction in video coding," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3983–3997, Aug. 2018.

[7] C.-Y. Tsai *et al.*, "Adaptive loop filtering for video coding," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 934–945, Dec. 2013.

[8] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, "AHG report: Test model software development (AHG3)," in *Proc. 14th Meeting Joint Video Explor. Team*, Oct. 2019, pp. 1–6.

[9] J. Pfaff *et al.*,"Neural network based intra prediction for video coding," in *Proc. SPIE*, vol. 10752, 2018, pp. 13–1–8.

[10] J. Li, B. Li, J. Xu, and R. Xiong, "Intra prediction using fully connected network for video coding," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2017, pp. 1–5.

[11] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao, "Fully connected network-based intra prediction for image coding," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3236–3247, Jul. 2018.
[12] I. Schiopu, H. Huang, and A. Munteanu, "CNN-based intra-prediction for lossless HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 1816–1828, Jul. 2020.
[13] M. Meyer, J. Wiesner, J. Schneider, and C. Rohlfing, "Convolutional neural networks for video intra prediction using cross-component adaptation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2019, pp. 1607–1611.
[14] F. Brand, J. Seiler, and A. Kaup, "Intra frame prediction for video coding using a conditional autoencoder approach," in *Proc. Picture Coding Symp.*, Nov. 2019, pp. 1–5.
[15] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval," in *Proc. Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 2011, pp. 489–494.
[16] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst. 28*, 2015, pp. 3483–3491. [Online]. Available: http://papers.nips.cc/paper/5775-learning-structured-output-representat ion-using-deep-conditional-generative-models.pdf
[17] C. Bif *et al.*, "3D high-resolution cardiac segmentation reconstruction from 2D views using conditional variational autoencoders," in *Proc. IEEE 16th Int. Symp. Biomed. Imag.*, Apr. 2019, pp. 1643–1646.
[18] M. Hosoe, T. Yamada, K. Kato, and K. Yamamoto, "Offline text-independent writer identification based on writer-independent model using conditional autoencoder," in *Proc. 16th Int. Conf. Front. Handwriting Recognit.*, Aug. 2018, pp. 441–446.
[19] D. Ding, G. Chen, D. Mukherjee, U. Joshi, and Y. Chen, "A CNN-based in-loop filtering approach for AV1 video codec," in *Proc. Picture Coding Symp.*, 2019, pp. 1–5.
[20] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.
[21] T. Laude, F. Haub, and J. Ostermann, "HEVC inter coding using deep recurrent neural networks and artificial reference pictures," in *Proc. Picture Coding Symp.*, Mar. 2019, pp. 1–5.
[22] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proc. Int. Conf. Learn. Representations*, Toulon, France, Apr. 2017, pp. 1–27.
[23] J. Ball, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–47.
[24] G. Cybenke, "Approximation by superpositions of a sigmoidial function," *Math. Control, Signals, Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.
[25] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Netw.*, vol. 6, no. 6, pp. 861–867, 1993.
[26] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML Workshop Deep Learn. Audio, Speech Lang. Process.*, 2013, pp. 675–678.
[27] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, no. 1, pp. 84–95, Jan. 1980.
[28] F. Brand, J. Seiler, and A. Kaup, "Introducing latent space correlation to conditional autoencoders for intra prediction," in *Proc. IEEE Vis. Commun. Image Process.*, Dec. 2020, pp. 1–4.
[29] Y. Li, S. Liu, and K. Kawamura, "Methodology and reporting template for neural network coding tool testing JVET-M1006," in *Proc. 13th Meeting Joint Video Explor. Team*, Marrakech, 2019, pp. 1–4.
[30] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, Nov. 2014, pp. 675–678.
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations* , May 2015, pp. 1–15.
[32] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, "JVET common test conditions and software reference configurations for SDR video, JVET-N1010-v1," in *Proc. 14th Meeting Joint Video Explor. Team*, Mar. 2019, pp. 1–6.
[33] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves, VCEG-M33," in *Proc. 13th Meeting Video Coding Experts Group*, Jan. 2001, pp. 1–5.
[34] W.-J. Chien *et al.*, "JVET AHG report: Tool reporting procedure (AHG13), JVET-P0013-v2," in *Proc. 16th Meeting Joint Video Explor. Team*, Oct. 2019, pp. 1–12.
[35] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 11264–11272.
[36] H. Xie, Y. Song, L. Cai, and M. Li, "Overflow aware quantization: Accelerating neural network inference by low-bit multiply-accumulate operations," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jan. 2020, pp. 868–875.

Fabian Brand (Graduate Student Member, IEEE) received the master's degree in electrical engineering from Friedrich-Alexander University Erlangen-Nürnberg (FAU), Germany, in 2018. While working towards the bachelor's degree, he worked on methods for frame-rate-conversion of video sequences, and during the master's degree, he researched automated harmonic analysis of classical music and style classification. Since 2019, he has been a Researcher with the Chair of Multimedia Communications and Signal Processing, Friedrich-Alexander University Erlangen-Nürnberg (FAU), where he conducts research on methods for video compression and deep learning. Mr. Brand received, the Best Paper Award of the Picture Coding Symposium (PCS) 2019.

Jürgen Seiler (Senior Member, IEEE) received the Habilitation degree in 2018, the Doctoral degree in 2011, and the Diploma degree in electrical engineering, electronics, and information technology in 2006. He is a Senior Scientist and Lecturer with the Chair of Multimedia Communications and Signal Processing, at the Friedrich-Alexander Universität Erlangen-Nürnberg, Germany. He has authored or coauthored more than 90 technical publications. His research interests include image and video signal processing, signal reconstruction and coding, signal transforms, and linear system theory. Dr. Seiler was the recipient of the dissertation award of the Information Technology Society of the German Electrical Engineering Association as well as the dissertation award of the Staedtler-Foundation, both in 2012. In 2007, he received diploma awards from the Institute of Electrical Engineering, Electronics and Information Technology, Erlangen, as well as from the German Electrical Engineering Association. He also received scholarships from the German National Academic Foundation and the Lucent Technologies Foundation. He is the corecipient of four Best Paper Awards.

André Kaup (Fellow, IEEE) received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from RWTH Aachen University, Aachen, Germany, in 1989 and 1995, respectively. He joined Siemens Corporate Technology, Munich, Germany, in 1995 and became Head of the Mobile Applications and Services Group in 1999. Since 2001, he has been a Full Professor and the Head of the Chair of Multimedia Communications and Signal Processing at Friedrich-Alexander University Erlangen-Nürnberg (FAU), Germany. From 1997 to 2001, he was the Head of the German MPEG delegation. From 2005 to 2007, he was a Vice Speaker of the DFG Collaborative Research Center 603. From 2015 to 2017, he was Head of the Department of Electrical Engineering and Vice Dean of the Faculty of Engineering with FAU. He has authored around 350 journal and conference papers and has over 120 patents, granted or pending. His research interests include image and video signal processing and coding, and multimedia communication. Dr. Kaup is a member of the IEEE Multimedia Signal Processing Technical Committee and a member of the scientific advisory board of the German VDE/ITG. He was an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and was a Guest Editor for IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING. He was a Siemens Inventor of the Year 1998 and obtained the 1999 ITG Award. He was the recipient of several Best Paper Awards including the Paul Dan Cristea Special Award in 2013, and his group won the Grand Video Compression Challenge at the Picture Coding Symposium 2013. The Faculty of Engineering with FAU honored him with the Teaching Award in 2015. In 2018, he was elected full member of the Bavarian Academy of Sciences.