Context-adaptive neural network based prediction for image compression

Thierry Dumas, Aline Roumy, Christine Guillemot

Context-adaptive neural network based prediction for image compression

Thierry Dumas, Aline Roumy, Christine Guillemot

***Abstract*— This paper describes a set of neural network architectures, called Prediction Neural Networks Set (PNNS), based on both fully-connected and convolutional neural networks, for intra image prediction. The choice of neural network for predicting a given image block depends on the block size, hence does not need to be signalled to the decoder. It is shown that, while fully-connected neural networks give good performance for small block sizes, convolutional neural networks provide better predictions in large blocks with complex textures. Thanks to the use of masks of random sizes during training, the neural networks of PNNS well adapt to the available context that may vary, depending on the position of the image block to be predicted. When integrating PNNS into a H.265 codec, PSNR-rate performance gains going from 1.46% to 5.20% are obtained. These gains are on average 0.99% larger than those of prior neural network based methods. Unlike the H.265 intra prediction modes, which are each specialized in predicting a specific texture, the proposed PNNS can model a large set of complex textures.**

***Index Terms*—Image compression, intra prediction, neural networks.**

I. Introduction

INTRA prediction is a key component of image and video compression algorithms and in particular of recent coding standards such as H.265 [1]. The goal of intra prediction is to infer a block of pixels from the previously encoded and decoded neighborhood. The predicted block is subtracted from the original block to yield a residue which is then encoded. Intra prediction modes used in practice rely on very simple models of dependencies between the block to be predicted and its neighborhood. This is the case of the H.265 standard which selects according to a rate-distortion criterion one mode among 35 fixed and simple prediction functions. The H.265 prediction functions consist in simply propagating the pixel values along specified directions [2]. This approach is suitable in the presence of contours, hence in small regions containing oriented edges [3], [4], [5]. However, it fails in large areas usually containing more complex textures [6], [7], [8]. Instead of simply propagating pixels in the causal neighborhood, the authors in [9] look for the best predictor within the image by searching for the best match with the so-called template of the block to be predicted. The authors in [10] further exploit self-similarities within the image with more complex models defined as linear combinations of k-nearest patches in the neighborhood.

Thierry Dumas is currently with Interdigital, 35576 Cesson-Sévigné, France (e-mail: thierry.dumas@interdigital.com). Aline Roumy and Christine Guillemot are with INRIA Rennes, 35042 Rennes, France (e-mail: aline.roumy@inria.fr, christine.guillemot@inria.fr).

This work has been supported by the French Defense Procurement Agency (DGA).

In this paper, we consider the problem of designing an intra prediction function that can predict both simple textures in small image blocks, as well as complex textures in larger ones. To create an optimal intra prediction function, the probabilistic model of natural images is needed. Let us consider a pixel, denoted by the random variable X, to be predicted from its neighboring decoded pixels. These neighboring decoded pixels are represented as a set $\mathcal{B}$ of observed random variables. The authors in [11] demonstrate that the optimal prediction $\hat{X}^*$ of X, i.e. the prediction that minimizes the mean squared prediction error, is the conditional expectation $\mathbb{E}\left[X|\mathcal{B}\right]$. Yet, no existing model of natural images gives a reliable $\mathbb{E}\left[X|\mathcal{B}\right]$.

However, neural networks have proved capable of learning a reliable model of the probability of image pixels for prediction. For example, in [12], [13], recurrent neural networks sequentially update their internal representation of the dependencies between the pixels in the known region of an image and then generate the next pixel in the unknown region of the image.

In this paper, we consider the problem of learning, with the help of neural networks, a reliable model of dependencies between a block, possibly containing a complex texture, and its neighborhood that we refer to as its context. Note that neural networks have already been considered in [14] for intra block prediction. However, the authors in [14] only take into consideration blocks of sizes 4×4, 8×8, 16×16, and 32×32 pixels and use fully-connected neural networks. Here, we consider both fully-connected and convolutional neural networks. We show that, while fully-connected neural networks give good performance for small block sizes, convolutional neural networks are more appropriate, both in terms of prediction PSNR and PSNR-rate performance gains, for large block sizes. The choice of neural network is block size dependent, hence does not need to be signalled to the decoder. This set of neural networks, called Prediction Neural Networks Set (PNNS), has been integrated into a H.265 codec, showing PSNR-rate performance gains from 1.46% to 5.20%.

In summary, the contributions of this paper are as follows:

- We propose a set of neural network architectures, including both fully-connected and convolutional neural networks, for intra image prediction.
- We show that, in the case of large block sizes, convolutional neural networks yield more accurate predictions compared with fully-connected ones.
- Thanks to the use of masks of random sizes during training, the neural networks of PNNS well adapt to the available context that may vary. E.g. in H.265, the available context, hence the number of known pixels in the neighborhood, depends on the position of the

considered prediction unit within the coding unit and within the coding tree unit.

- Unlike the H.265 intra prediction modes, which are each specialized in predicting a specific texture, the proposed PNNS, trained on a large unconstrained set of images, is able to model a large set of complex textures.
- We prove experimentally a surprising property of the neural networks for intra prediction: they do not need to be trained on distorted contexts, meaning that the neural networks trained on undistorted contexts generalize well on distorted contexts, even for severe distortions.

The code to reproduce our numerical results and train the neural networks is available online [1].

II. Conditions for efficient neural network based intra prediction

Prediction is a key method in rate distortion theory, when complexity is an issue. Indeed, the complexity of vector quantization is prohibitive, and scalar quantization is rather used. But, scalar quantization cannot exploit the statistical correlations between data samples. This task can be done via prediction [15]. Prediction can only be made from data samples available at the decoder, i.e. causal and distorted data samples. By distorted causal data samples we mean previously encoded and decoded pixels above and on the left side of the image block to be predicted. This set of pixels is often referred to as the context of the block to be predicted.

Optimal prediction, i.e. conditional expectation [11], requires knowing the conditional distribution of the image block to be predicted given causal and distorted data samples. Estimating such a conditional distribution is difficult. The use of the predictor by the decoder would in addition require sending the distribution parameters. Classical approaches in predictive coding consist in proposing a set of predefined functions and choosing the best of them in a rate-distortion sense. Thus, the number of possible functions is limited. On the other hand, neural networks can approximate many functions, in particular complex predictive functions such as the generation of future video frames given an input sequence of frames [16], [17].

But, the use of neural networks for intra prediction within an image coding scheme raises several questions that we address in this paper. What neural network architecture provides enough power of representation to map causal and distorted data samples to an accurate prediction of a given image block? What context size should be used? Section III looks for a neural network architecture and the optimal number of causal and distorted data samples for predicting a given image block. Moreover, the amount of causal and distorted data samples available at the decoder varies. It depends on the partitioning of the image and the position of the block to be predicted within the image. Section IV trains the neural networks so that they adapt to the variable context size. Finally, can neural networks compensate for the quantization noise in its input and be efficient in a rate-distortion sense? Sections V and VI answer these two questions with experimental evidence.

[1] https://github.com/thierrydumas/context_adaptive_neural_network_based_prediction

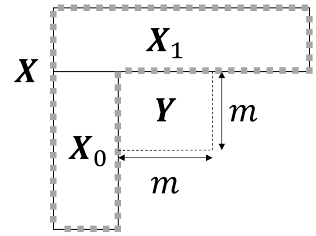


Fig. 1: Illustration of the relative positions of $\mathbf{X}$, $\mathbf{X}_0$, $\mathbf{X}_1$, and $\mathbf{Y}$.

III. Proposed neural network based intra prediction

Unlike standard intra prediction in which the encoder chooses the best mode in a rate-distortion sense among several pre-defined modes, only one neural network among a set of neural networks does the prediction here. Unlike [14], our set contains both fully-connected and convolutional neural networks. This section first presents our set of neural networks. Then, it explains how one neural network is selected for predicting a given image block and the context is defined according to the block size. Finally, the specificities of the integration of our neural networks into H.265 are detailed.

A. Fully-connected and convolutional neural networks

Let $\mathbf{X}$ be a context containing decoded pixels above and on the left side of a square image block $\mathbf{Y}$ of width $m \in \mathbb{N}^*$ to be predicted (see Figure 1). The transformation of $\mathbf{X}$ into a prediction $\hat{\mathbf{Y}}$ of $\mathbf{Y}$ via either a fully-connected neural network f_m, parametrized by $\boldsymbol{\theta}_m$, or a convolutional neural network g_m, parametrized by $\boldsymbol{\phi}_m$, is described in (1). The corresponding architectures are depicted in Figures 2 and 3.

$$\begin{aligned} \mathbf{X}_c &= \mathbf{X} - \alpha \\ \hat{\mathbf{Y}}_c &= \begin{cases} f_m\left(\mathbf{X}_c; \boldsymbol{\theta}_m\right) \\ g_m\left(\mathbf{X}_c; \boldsymbol{\phi}_m\right) \end{cases} \\ \hat{\mathbf{Y}} &= \max\left(\min\left(\hat{\mathbf{Y}}_c + \alpha, 255\right), 0\right) \end{aligned} \tag{1}$$

During optimization, each input variable to a neural network must be approximatively zero-centered over the training set to accelerate convergence [18]. Besides, since the pixel space corresponds to both the input and output spaces in intra prediction, it makes sense to normalize the input and output similarly. One could subtract from each image block to be predicted and its context their respective mean during training. But, this entails sending the mean of a block to be predicted to the decoder during the test phase. Instead, the mean pixel intensity α is first computed over all the training images. Then, α is subtracted from each image block to be predicted and its context during training. During the test phase, α is subtracted from the context (see (1) where the subscript c stands for centered).

This preprocessing implies a postprocessing of the neural network output. More precisely, the learned mean pixels

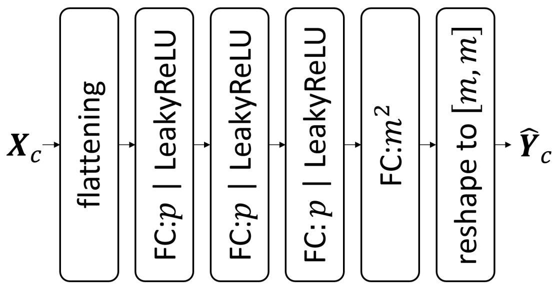


Fig. 2: Illustration of the fully-connected architecture f_m. FC:p|LeakyReLU denotes the fully-connected layer with p output neurons and LeakyReLU with slope 0.1 as non-linear activation. Unlike FC:p|LeakyReLU, FC:p has no non-linear activation. $\boldsymbol{\theta}_m$ gathers the weights and biases of the 4 fully-connected layers of f_m.

intensity is added to the output and the result is clipped to $[0, 255]$ (see (1)).

The first operation for both architectures consists in formatting the context to ease the computations in the neural network. In the case of a fully-connected neural network, all elements in the context are connected such that there is no need to keep the 2D structure of the context [19]. Therefore, the context is first vectorized, and fast vector-matrix arithmetics can be used (see Figure 2). However, in the case of a convolutional neural network, fast computation of 2D filtering requires to keep the 2D structure of the context. Moreover, again for fast computation, the shape of the input to the convolution has to be rectangular. That is why the context is split into two rectangles $\mathbf{X}_0$ and $\mathbf{X}_1$ (see Figures 1 and 3). $\mathbf{X}_0$ and $\mathbf{X}_1$ are then processed by distinct convolutions.

The proposed fully-connected architecture is composed of 4 fully-connected layers. The first layer computes an overcomplete representation of the context to reach $p \in \mathbb{N}^*$ output coefficients. Overcompleteness is chosen as it is observed empirically that overcomplete representations in early layers boost the performance of neural networks [20], [21], [22]. The next two layers keep the number of coefficients unchanged, while the last layer reduces it to provide the predicted image block. The first three layers have LeakyReLU [23] with slope 0.1 as non-linear activation. The last layer has no non-linear activation. This is because the postprocessing discussed earlier contains already a non-linearity, which consists in first adding the learned mean pixels intensity to the output and clipping the result to $[0, 255]$.

The first task of the convolutional architecture is the computation of features characterizing the dependencies between the elements in $\mathbf{X}_0$. $\mathbf{X}_0$ is thus fed into a stack of convolutional layers. This yields a stack $\mathbf{Z}_0$ of $l \in \mathbb{N}^*$ feature maps (see Figure 3). Similarly, $\mathbf{X}_1$ is fed into another stack of convolutional layers. This yields a stack $\mathbf{Z}_1$ of l feature maps.

All the elements in the context can be relevant for predicting any image block pixel. This implies that the information associated to all spatial locations in the context has to be merged. That is why the next layer in the convolutional architecture merges spatially $\mathbf{Z}_0$ and $\mathbf{Z}_1$ (see Figure 3). More precisely, for $i \in [|1, l|]$, all the coefficients of the i^{th} feature map of $\mathbf{Z}_0$ and of the i^{th} feature map of $\mathbf{Z}_1$ are merged through affine combinations. Then, LeakyReLU with slope 0.1 is applied, yielding the merged stack $\overline{\mathbf{Z}}$ of feature maps. Note that this layer bears similarities with the "channelwise fully-connected layer" [24]. But, unlike the "channelwise fully-connected layer", it merges two stacks of feature maps of different height and width. Its advantage over a fully-connected layer is that it contains l times less weights.

The last task of the convolutional architecture is to merge the information of the different feature maps of $\overline{\mathbf{Z}}$. $\overline{\mathbf{Z}}$ is thus fed into a stack of transposed convolutional layers [25], [26]. This yields the predicted image block (see Figure 3). Note that all convolutional layers and transposed convolutional layers, apart from the last transposed convolutional layer, have LeakyReLU with slope 0.1 as non-linear activation. The last transposed convolutional layer has no non-linear activation due to the postprocessing discussed earlier.

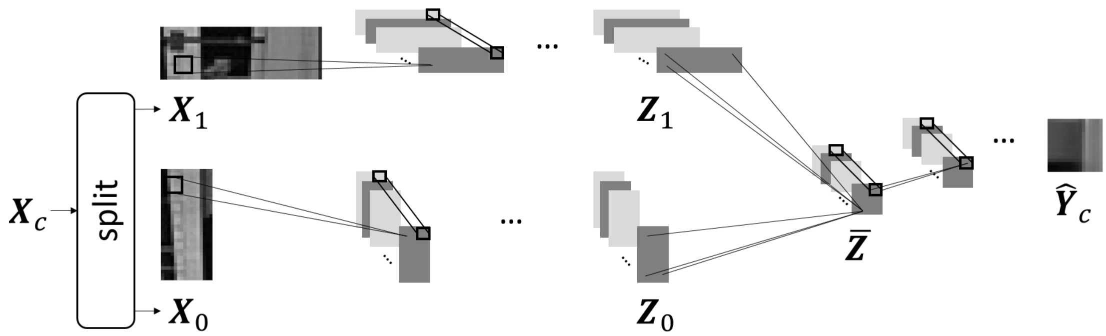
B. Growth rate of the context size with the block size

Now that the architectures and the shape of the context are defined, the size of the context remains to be optimized. The causal neighborhood of the image block to be predicted used by the H.265 intra prediction modes is limited to one row of $2m+1$ decoded pixels above the block and one column of $2m$ decoded pixels on the left side of the block. However, a context of such a small size is not sufficient for neural networks as a neural network relies on the spatial distribution of the decoded pixels intensity in the context to predict complex textures. Therefore, the context size has to be larger than $4m + 1$.

But, an issue arises when the size of the context grows too much. Indeed, if the image block to be predicted is close to either the top edge of the decoded image $\mathbf{D}$ or its left edge, a large context goes out of the bounds of the decoded image. The neural network prediction is impossible. There is thus a tradeoff to find a suitable size for the context.

Let us look at decoded pixels above and on the left side of $\mathbf{Y}$ to develop intuitions regarding this tradeoff. When m is small, the long range spatial dependencies between these decoded pixels are not relevant for predicting $\mathbf{Y}$ (see Figure 4). In this case, the size of the context should be small so that the above-mentioned issue is limited. However, when m is large, such long range spatial dependencies are informative for predicting $\mathbf{Y}$. The size of the context should now be large, despite the issue. Therefore, the context size should be a function of $m^q, q \geq 1$.

From there, we ran several preliminary experiments in which $q \in \{1, 2, 3\}$ and the PSNR between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ was measured. The conclusion is that a neural network yields the best PSNRs when the size of the context grows with m^2, i.e. the ratio between the size of the context and the size of the image block to be predicted is constant. This makes sense as, in the most common regression tasks involving neural networks, such as super-resolution [28], [29], segmentation [30], [31] or video interpolation [32], [33], [34], the ratio between the input image dimension and the output image dimension also remains constant while the height and width of the output image increase. Given the previous conclusions,



$$Z_i = g_m^c\left(X_i; \phi_m^{c,i}\right), \quad \bar{Z} = \bar{g}_m(Z_0, Z_1; \bar{\phi}_m), \quad \hat{Y}_c = g_m^t(\bar{Z}; \phi_m^t)$$

Fig. 3: Illustration of the convolutional architecture g_m in which g_m^c, $\overline{g}_m$, and g_m^t denote respectively a stack of convolutional layers, the merger, and the stack of transposed convolutional layers. ϕ_m^c, $\overline{\phi}_m$, and ϕ_m^t gather the weights and biases of respectively g_m^c, $\overline{g}_m$, and g_m^t. $\phi_m = \left\{\phi_m^{c,0}, \phi_m^{c,1}, \overline{\phi}_m, \phi_m^t\right\}$.

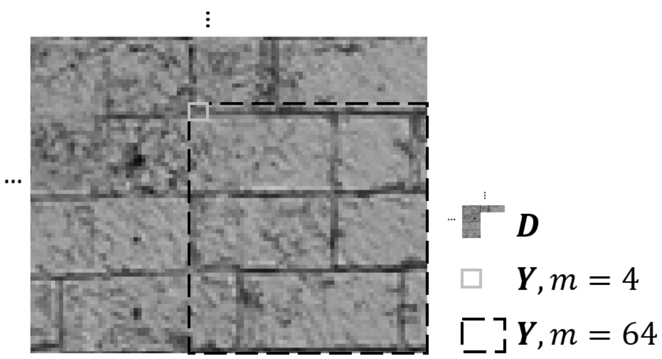


Fig. 4: Dependencies between $\mathbf{D}$ and $\mathbf{Y}$. The luminance channel of the first image in the Kodak suite [27] is being encoded via H.265 with Quantization Parameter QP $= 17$.

$\mathbf{X}_0$ is a rectangle of height $2m$ and width m. $\mathbf{X}_1$ is a rectangle of height m and width $3m$.

C. Criterion for selecting a proposed neural network

Now that the context is defined with respect to m, all that remains is to choose between a fully-connected neural network and a convolutional one according to m.

Convolutional neural networks allow better extracting 2D local image structures with fewer parameters compared to fully-connected neural networks [19]. This is verified by the better prediction performances of the convolutional neural networks for $m > 8$ (see Sections IV-E and IV-F). However, this property becomes less critical for small block sizes. We indeed experimentally show that, when $m \leq 8$, for the prediction problem, fully-connected architectures outperform convolutional architectures.

Therefore, when integrating the neural network based intra predictor into a codec like H.265, the criterion is to predict a block of width $m > 8$ via a proposed convolutional neural network, and a block of width $m \leq 8$ via a fully-connected neural network.

D. Integration of the neural network based intra prediction into H.265

A specificity of H.265 is the quadtree structure partitioning, which determines the range of values for m and the number of available decoded pixels in the context.

In H.265 [1], an image is partitioned into Coding Tree Units (CTUs). A CTU is composed of one luminance Coding Tree Block (CTB), two chrominance CTBs, and syntax elements. For simplicity, let us focus on a single CTB, e.g. the luminance CTB. The CTB size is a designed parameter but the commonly used CTB size is 64×64 pixels. A CTB can be directly used as Coding Block (CB) or can be split into 4 32×32 CBs. Then, each 32×32 CB can be iteratively split until the size of a resulting CB reaches a minimum size. The minimum size is a designed parameter. It can be as small as 8×8 pixels, and is set to this value in most configurations. A Prediction Block (PB) is a block on which the prediction is applied. If the size of a CB is not the minimum size, this CB is identical to its PB. Otherwise, in the case of intra prediction, this CB can be split into 4 4×4 PBs. More splittings of this CB into PBs exist for inter prediction [1]. A recursive rate-distortion optimization finds the optimal splitting of each CTB.

Due to this partioning, $m \in \{4, 8, 16, 32, 64\}$. For each $m \in \{4, 8\}$, a fully-connected neural network is constructed with internal size $p = 1200$. Similarly, one convolutional neural network is constructed per block width $m \in \{16, 32, 64\}$. The convolutional architecture for each $m \in \{16, 32, 64\}$ is detailed in Appendix A.

Another consequence of this partitioning is that the number of available decoded pixels in the context depends on m and the position of image block to be predicted in the current CTB. For instance, if the block is located at the bottom of the current CTB, the bottommost m^2 pixels in the context are not decoded yet. More generally, it might happen that a group of $n_0 \times m$ pixels, $n_0 \in \{0, 4, ..., m\}$, located at the bottom of the context, is not decoded yet. Similarly, a group of $m \times n_1$ pixels, $n_1 \in \{0, 4, ..., m\}$, located furthest to the right in the context, may not have been decoded yet (see Figure 5). When pixels are not decoded yet, the solution in H.265

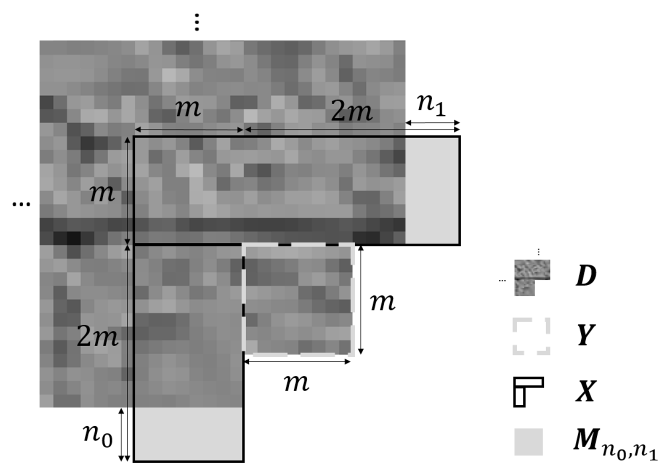


Fig. 5: Illustration of the masking of the undecoded pixels in $\mathbf{X}$ for H.265. The luminance channel of the first image in the Kodak suite is being encoded via H.265 with QP $= 37$. Here, $m = 8$ and $n_0 = n_1 = 4$.

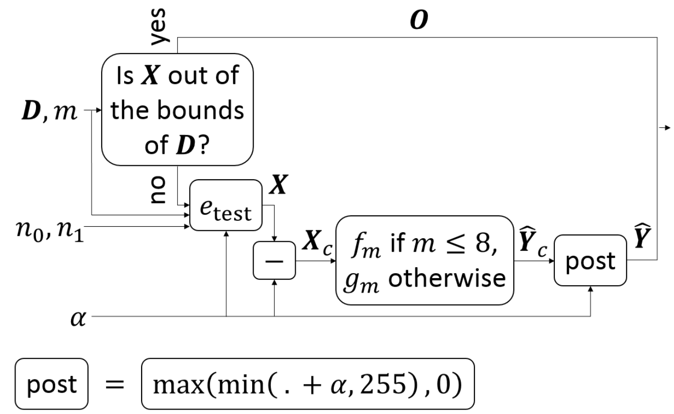


Fig. 6: Illustration of the neural network based intra prediction scheme inside H.265.

is to copy a decoded pixel into its neighboring undecoded pixels. But, this copy process cannot be re-used here. Indeed, it would fool the neural network and make it believe that, in an undecoded group of pixels and its surroundings, the spatial distribution of pixels intensity follows the regularity induced by the copy process. Alternatively, it is possible to indicate to a neural network that the two undecoded groups are unknown by masking them. The mask is set to the learned mean pixels intensity over the training set so that, after subtracting it from the context, the average of each input variable to a neural network over the training set is still near 0. More precisely regarding the masking, the first undecoded group in the context is fully covered by an α-mask of height n_0 and width m. The second undecoded group in the context is fully covered by an α-mask of height m and width n_1 (see Figure 5). The two α-masks together are denoted $\mathbf{M}_{n_0,n_1}$. In Section IV-A, the neural networks will be trained so that they adapt to this variable number of available decoded pixels in the context.

Figure 6 summarizes the integration of the neural network based intra prediction scheme into H.265. The last issue to address is when no context is available. This occurs when the context goes out of the bounds of the decoded image, i.e. the pixel at the top-left of the context is not inside the decoded image. In this case, no neural network is used, and a zero prediction $\mathbf{O}$ of $\mathbf{Y}$ is returned. In Figure 6, e_{test} extracts $\mathbf{X}$ from $\mathbf{D}$ with respect to the position of $\mathbf{Y}$ while applying $\mathbf{M}_{n_0,n_1}$ to $\mathbf{X}$, following Figure 5.

IV. Neural networks training

This section explains how our neural networks are trained. Notably, an adaptation to the changing number of available decoded pixels in the input context is proposed. Then, an experiment shows the effectiveness of this approach. Moreover, this experiment compares the predictions of convolutional neural networks and those of fully-connected neural networks in terms of prediction PSNR in the case of large image blocks to be predicted.

A. Adaptation of the neural networks to the variable number of available decoded pixels via random context masking

The fact that n_0 and n_1 vary during the test phase, e.g. in H.265, has to be considered during the training phase. It would be unpractical to train one set of neural networks for each possible pair $\{n_0, n_1\}$. Instead, we propose to train the neural networks while feeding them with contexts containing a variable number of known pixels. More precisely, during the training phase, n_0 and n_1 are sampled uniformly from the set $\{0, 4, ..., m\}$. This way, the amount of available information in a training context is viewed as a random process the neural networks have to cope with.

B. Objective function to be minimized

The goal of the prediction is to minimize the Euclidean distance between the image block to be predicted and its estimate, or in other words to minimize the variance of the difference between the block and its prediction [11], also called the residue. The choice of the L2 norm is a consequence of the L2 norm chosen to measure the distortion between an image and its reconstruction. So, this Euclidean distance is minimized to learn the parameters $\boldsymbol{\theta}_m$. Moreover, regularization by L2 norm of the weights (not the biases), denoted $[\boldsymbol{\theta}_m]_W$, is applied [35] (see (2)).

$$\min_{\boldsymbol{\theta}_m} \mathbb{E}\left[\left\|\mathbf{Y}_c - f_m\left(\mathbf{X}_c; \boldsymbol{\theta}_m\right)\right\|_2\right] + \lambda \left\|[\boldsymbol{\theta}_m]_W\right\|_2^2 \quad (2)$$
$$\mathbf{Y}_c = \mathbf{Y} - \alpha$$

The expectation $\mathbb{E}[.]$ is approximated by averaging over a training set of image blocks to be predicted, each paired with its context. For learning the parameters $\boldsymbol{\phi}_m$, (2) is used, replacing $\boldsymbol{\theta}_m$ with $\boldsymbol{\phi}_m$ and f_m with g_m.

The optimization algorithm is ADAM [36] with mini-batches of size 100. The learning rate is 0.0001 for a fully-connected neural network and 0.0004 for a convolutional one. The number of iterations is 800000. The learning is divided by 10 after 400000, 600000, and 700000 iterations. Regarding the other hyperparameters of ADAM, the recommended values [36] are used. The learning rates and $\lambda = 0.0005$ were found via an exhaustive search. For a given m, a neural network is trained for each combination of learning rate and λ value in $\{0.007, 0.004, 0.001, 0.0007, 0.0004, 0.0001, 0.00007, 0.00004\}$ $\times$ $\{0.001, 0.0005, 0.0001\}$. We select the pair of parameter values giving the largest mean prediction PSNR on a validation set.

For both fully-connected and convolutional neural networks, the weights of all layers excluding the first one are initialized via Xavier's initialization [37]. Xavier's initialization allows to accelerate the training of several consecutive layers by making the variance of the input to all layers equivalent. The problem concerning the first layer is that the pixel values in the input contexts, belonging to $[-\alpha, 255-\alpha]$, have a large variance, which leads to instabilities during training. That is why, in the first layer, the variance of the output is reduced with respect to that of the input by initializing the weights from a Gaussian distribution with mean 0 and standard deviation 0.01. The biases of all layers are initialized to 0.

C. Training data

The experiments in Sections IV-D and VI involve luminance images. That is why, for training, image blocks to be predicted, each paired with its context, are extracted from luminance images.

One 320×320 luminance crop $\mathbf{I}$ is, if possible, extracted from each RGB image in the ILSVRC2012 training set [38]. This yields a set $\boldsymbol{\Gamma} = \left\{\mathbf{I}^{(i)}\right\}_{i=1\ldots1048717}$.

The choice of the training set of pairs of contexts and image blocks to be predicted is a critical issue. Moreover, it needs to be handled differently for fully-connected and convolutional neural networks. Indeed, a convolutional neural network predicts large image blocks with complex textures, hence its need for high power of representation (see Appendix A). As a consequence, it overfits during training if no training data augmentation [39], [40], [41], [42] is used. In constrast, a fully-connected neural network predicts small blocks with simple textures, hence its need for relatively low power of representation. Thus, it does not overfit during training without training data augmentation. Moreover, we have noticed that a training data augmentation scheme creates a bottleneck in training time for a fully-connected neural network. Therefore, training data augmentation is used for a convolutional neural network exclusively. The dependencies between a block to be predicted and its context should not be altered during the training data augmentation. Therefore, in our training data augmentation scheme, the luminance crops in $\boldsymbol{\Gamma}$ are exclusively randomly rotated and flipped. Precisely, for each step of ADAM, the scheme is Algorithm 1. s_{rotation} rotates its input image by angle $\psi \in \{0, \pi \,/\, 2, \pi, 3\pi \,/\, 2\}$ radians. s_{flipping} flips its input image horizontally with probability 0.5. e_{train} is the same function as e_{test}, except that e_{train} extracts $\left\{\mathbf{X}^{(i)}, \mathbf{Y}^{(i)}\right\}$ from potentially rotated and flipped $\mathbf{I}$ instead of extracting $\mathbf{X}$ from $\mathbf{D}$ and the position of the extraction is random instead of being defined by the order of decoding. For training a fully-connected neural network, $\left\{\mathbf{X}_c^{(i)}, \mathbf{Y}_c^{(i)}\right\}_{i=1\ldots10000000}$ is generated offline from $\boldsymbol{\Gamma}$, i.e. before the training starts.

The issue regarding this generation of training data is that the training contexts have no quantization noise whereas, during the test phase in a coding scheme, a context has quantization noise. This will be discussed during several experiments in Section VI-D.

Algorithm 1 Training data augmentation for the convolutional neural networks

Inputs: $\boldsymbol{\Gamma}$, m, α.

$$
\begin{aligned}
&\forall i \in [|1,100|],\\
&\quad \bar{i} \sim \mathcal{U}\,[1, 1048717]\\
&\quad \psi \sim \mathcal{U}\left\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}\\
&\quad n_0, n_1 \sim \mathcal{U}\,\{0, 4, ..., m\}\\
&\quad \mathbf{J} = s_{\text{flipping}}\left(s_{\text{rotation}}\left(\mathbf{I}^{(\bar{i})}, \psi\right)\right)\\
&\quad \left\{\mathbf{X}^{(i)}, \mathbf{Y}^{(i)}\right\} = e_{\text{train}}\left(\mathbf{J}, m, n_0, n_1, \alpha\right)\\
&\quad \mathbf{X}_c^{(i)} = \mathbf{X}^{(i)} - \alpha\\
&\quad \mathbf{Y}_c^{(i)} = \mathbf{Y}^{(i)} - \alpha
\end{aligned}
$$

Output: $\left\{\mathbf{X}_c^{(i)}, \mathbf{Y}_c^{(i)}\right\}_{i=1\ldots100}$.

D. Effectiveness of the random context masking

A natural question is whether the random context masking applied during training to adapt the neural networks to the variable number of known pixels in the context degrades the prediction performance. To address this question, a neural network trained with random context masking is compared to a set of neural networks, each trained with a fixed mask size. The experiments are performed using fully-connected neural networks for block sizes 4×4 and 8×8 pixels (f_4 and f_8), and convolutional neural networks for block sizes 16×16, 32×32, and 64×64 pixels (g_{16}, g_{32}, and g_{64}).

The experiments are carried out using the 24 RGB images in the Kodak suite [27], converted into luminance. 960 image blocks to be predicted, each paired with its context, are extracted from these luminance images. Table I shows the average PSNR between the image block to be predicted and its prediction via PNNS over the 960 blocks, denoted $\text{PSNR}_{\text{PNNS},m}$, for each block width m and each test pair $\{n_0, n_1\}$. We see that a neural network trained with a fixed mask size has performance in terms of PSNR that significantly degrades when the mask size during the training phase and the test phase differ. By contrast, a neural network trained with random context masking allows to get the best (bold) or the second best (italic) performance in terms of PSNR for all the possible mask sizes during the test phase. Moreover, when the second best PSNR performance is achieved, the second best PSNR is very close to the best one.

A second set of experiments is related to the success rate $\mu_{\text{PNNS},m}$ of the neural network based prediction, i.e. the rate at which PNNS provides a better prediction in terms of PSNR than any other prediction of H.265. Again, the neural network trained with random context masking achieves the best (bold) or the second best rate (italic), where the second best rate is very close to the best rate (see Table II).

TABLE I: Comparison of (a) $\mathrm{PNSR}_{\mathrm{PNNS},4}$, (b) $\mathrm{PSNR}_{\mathrm{PNNS},8}$, (c) $\mathrm{PSNR}_{\mathrm{PNNS},16}$, (d) $\mathrm{PSNR}_{\mathrm{PNNS},32}$, and (e) $\mathrm{PSNR}_{\mathrm{PNNS},64}$ for different pairs $\{n_0, n_1\}$ during the training and test phases. $\mathcal{U}\{0, 4, ..., m\}$ denotes the fact that, during the training phase, n_0 and n_1 are uniformly drawn from the set $\{0, 4, 8, 12, ..., m\}$.

Test $\{n_0, n_1\}$	Training f_4 with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 4\}$	$\{4, 0\}$	$\{4, 4\}$	$\mathcal{U}\{0, 4\}$
$\{0, 0\}$	**34.63**	34.39	34.44	34.23	*34.57*
$\{0, 4\}$	32.90	*34.39*	32.82	34.23	**34.42**
$\{4, 0\}$	32.79	32.68	**34.44**	34.23	*34.39*
$\{4, 4\}$	30.93	32.68	32.82	**34.23**	*34.20*

(a)

Test $\{n_0, n_1\}$	Training f_8 with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 8\}$	$\{8, 0\}$	$\{8, 8\}$	$\mathcal{U}\{0, 4, 8\}$
$\{0, 0\}$	**31.85**	31.73	31.68	31.62	*31.79*
$\{0, 8\}$	30.93	**31.73**	30.83	31.62	*31.66*
$\{8, 0\}$	30.70	30.75	**31.68**	31.62	**31.68**
$\{8, 8\}$	29.58	30.75	30.83	**31.62**	*31.56*

(b)

Test $\{n_0, n_1\}$	Training g_{16} with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 16\}$	$\{16, 0\}$	$\{16, 16\}$	$\mathcal{U}\{0, 4, ..., 16\}$
$\{0, 0\}$	*29.23*	22.69	24.85	20.76	**29.25**
$\{0, 16\}$	28.65	**29.12**	24.66	23.99	*29.11*
$\{16, 0\}$	28.43	22.60	*29.06*	22.99	**29.12**
$\{16, 16\}$	27.87	28.37	28.35	**28.98**	*28.97*

(c)

Test $\{n_0, n_1\}$	Training g_{32} with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 32\}$	$\{32, 0\}$	$\{32, 32\}$	$\mathcal{U}\{0, 4, ..., 32\}$
$\{0, 0\}$	**25.93**	22.13	22.29	18.85	*25.92*
$\{0, 32\}$	25.39	**25.79**	22.18	21.75	*25.76*
$\{32, 0\}$	25.41	22.02	**25.82**	20.38	*25.80*
$\{32, 32\}$	25.00	25.31	25.32	**25.63**	*25.62*

(d)

Test $\{n_0, n_1\}$	Training g_{64} with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 64\}$	$\{64, 0\}$	$\{64, 64\}$	$\mathcal{U}\{0, 4, ..., 64\}$
$\{0, 0\}$	*21.46*	19.41	19.78	18.17	**21.47**
$\{0, 64\}$	21.27	*21.35*	19.68	19.95	**21.38**
$\{64, 0\}$	21.11	19.18	**21.34**	19.06	**21.34**
$\{64, 64\}$	20.94	21.08	21.16	**21.27**	**21.27**

(e)

TABLE II: Comparison of success rates in percentage (a) $\mu_{\mathrm{PNNS},4}$, (b) $\mu_{\mathrm{PNNS},8}$, (c) $\mu_{\mathrm{PNNS},16}$, (d) $\mu_{\mathrm{PNNS},32}$, and (e) $\mu_{\mathrm{PNNS},64}$ for different pairs $\{n_0, n_1\}$ during the training and test phases. $\mathcal{U}\{0, 4, ..., m\}$ denotes the fact that, during the training phase, n_0 and n_1 are uniformly drawn from the set $\{0, 4, 8, 12, ..., m\}$.

Test $\{n_0, n_1\}$	Training f_4 with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 4\}$	$\{4, 0\}$	$\{4, 4\}$	$\mathcal{U}\{0, 4\}$
$\{0, 0\}$	**22%**	17%	19%	16%	*19%*
$\{0, 4\}$	15%	**18%**	13%	15%	*17%*
$\{4, 0\}$	11%	11%	**20%**	17%	*19%*
$\{4, 4\}$	11%	12%	13%	**16%**	*15%*

(a)

Test $\{n_0, n_1\}$	Training f_8 with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 8\}$	$\{8, 0\}$	$\{8, 8\}$	$\mathcal{U}\{0, 4, 8\}$
$\{0, 0\}$	**33%**	30%	30%	30%	*32%*
$\{0, 8\}$	21%	**31%**	20%	**31%**	29%
$\{8, 0\}$	21%	20%	**34%**	31%	*32%*
$\{8, 8\}$	16%	20%	20%	**31%**	*30%*

(b)

Test $\{n_0, n_1\}$	Training g_{16} with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 16\}$	$\{16, 0\}$	$\{16, 16\}$	$\mathcal{U}\{0, 4, ..., 16\}$
$\{0, 0\}$	**55%**	18%	20%	9%	*54%*
$\{0, 16\}$	42%	**51%**	18%	18%	*50%*
$\{16, 0\}$	40%	15%	*51%*	17%	**53%**
$\{16, 16\}$	33%	36%	40%	**52%**	*49%*

(c)

Test $\{n_0, n_1\}$	Training g_{32} with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 32\}$	$\{32, 0\}$	$\{32, 32\}$	$\mathcal{U}\{0, 4, ..., 32\}$
$\{0, 0\}$	**63%**	30%	30%	14%	**63%**
$\{0, 32\}$	54%	**61%**	27%	27%	**61%**
$\{32, 0\}$	54%	28%	**63%**	22%	*62%*
$\{32, 32\}$	46%	53%	52%	**60%**	**60%**

(d)

Test $\{n_0, n_1\}$	Training g_{64} with $\{n_0, n_1\}$				
	$\{0, 0\}$	$\{0, 64\}$	$\{64, 0\}$	$\{64, 64\}$	$\mathcal{U}\{0, 4, ..., 64\}$
$\{0, 0\}$	**68%**	39%	43%	27%	*67%*
$\{0, 64\}$	63%	*64%*	41%	44%	**65%**
$\{64, 0\}$	62%	36%	*66%*	38%	**68%**
$\{64, 64\}$	57%	61%	63%	*64%*	**66%**

(e)

Therefore, we conclude that random context masking does not alter the performance of the trained neural networks. Besides, random context masking is an effective way of dealing with the changing number of known pixels in the context. Section VI will discuss rate-distortion performance. Note that, in order to fix n_0 and n_1 during the test phase, the previous experiments have been carried out outside H.265.

E. Relevance of convolutional neural networks for predicting large image blocks

Let us have a look at the overall efficiency of our convolutional neural networks for predicting large image blocks before comparing convolutional neural networks and fully-connected neural networks in this case. Figures 7, 8, 9, 10, 11, and 12 each compares the prediction of an image block provided by the best H.265 intra prediction mode in terms of prediction PSNR and the prediction provided by PNNS. Note that, the neural networks of PNNS yielding these predictions are trained via random context masking. Note also that $n_0 = n_1 = 0$ during the test phase. In Figure 11, the image block to be predicted contains the frame of a motorbike. There is no better H.265 intra prediction mode in terms of prediction PSNR than the DC mode in this case. In contrast, PNNS can predict a coarse version of the frame of the motorbike. In Figure 12, the block to be predicted contains lines of various directions. PNNS predicts a combination of diagonal lines, vertical lines and horizontal lines, which is not feasible for a H.265 intra prediction mode. However, Figures 9 and 10 show failure cases for the convolutional neural network. The quality of the prediction provided by the convolutional neural network looks worse than that provided by the best H.265 mode. Indeed, the prediction PSNRs are 27.09 dB and 23.85 dB for the H.265 mode of index 25 and the convolutional neural network respectively in Figure 9. They are 29.35 dB and 27.37 dB for the H.265 mode of index 29 and the convolutional neural

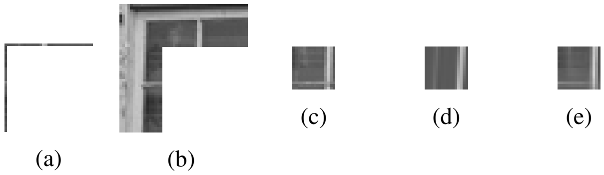


Fig. 7: Prediction of a block of size 16×16 pixels via the best H.265 mode in terms of PSNR and PNNS: (a) H.265 causal neighborhood, (b) PNNS context, (c) block to be predicted, (d) predicted block via the best H.265 mode (of index 27) in terms of PSNR, and (e) predicted block via PNNS.

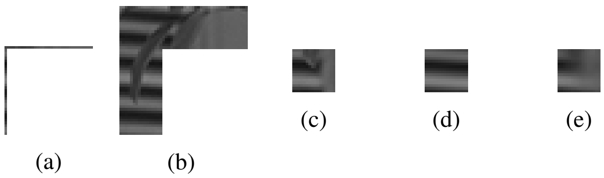


Fig. 8: Prediction of a block of size 16×16 pixels via the best H.265 mode in terms of PSNR and PNNS: (a) H.265 causal neighborhood, (b) PNNS context, (c) block to be predicted, (d) predicted block via the best H.265 mode (of index 11) in terms of PSNR, and (e) predicted block via PNNS.

network respectively in Figure 10. Therefore, the convolutional neural networks of PNNS is often able to model a large set of complex textures found in large image blocks.

Table III compares our set of neural networks $\{f_4, f_8, g_{16}, g_{32}, g_{64}\}$ with the set of 4 fully-connected neural networks in [14], called Intra Prediction Fully-Connected Networks Single (IPFCN-S), in terms of prediction PSNR. "Single" refers to the single set of 4 fully-connected neural networks. This differs from the two sets of fully-connected neural networks mentioned later on in Section VI-C. The 4 fully-connected neural networks in [14] predict image blocks of sizes 4×4, 8×8, 16×16, and 32×32 pixels respectively. Let $\text{PSNR}_{\text{IPFCN-S},m}$ be the PSNR between the image block and its prediction via IPFCN-S, averaged over the 960 blocks. We thank the authors in [14] for sharing the trained model of each fully-connected neural network of IPFCN-S. Table III reveals that the performance of PNNS in terms of prediction PSNR is better than that of IPFCN-S for all sizes of image blocks to be predicted. More interestingly, when looking at the success rate $\mu_{\text{IPFCN-S},m}$ of IPFCN-S,

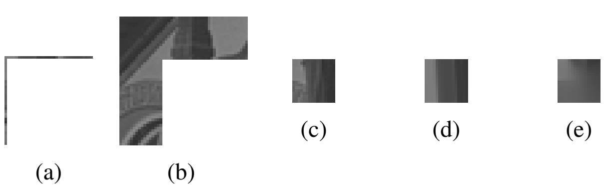


Fig. 9: Prediction of a block of size 16×16 pixels via the best H.265 mode in terms of PSNR and PNNS: (a) H.265 causal neighborhood, (b) PNNS context, (c) block to be predicted, (d) predicted block via the best H.265 mode (of index 25) in terms of PSNR, and (e) predicted block via PNNS.

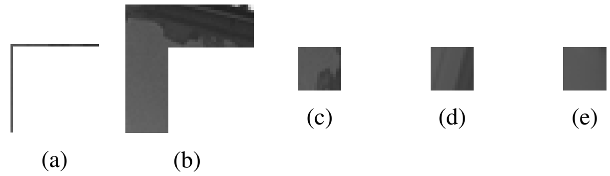


Fig. 10: Prediction of a block of size 16×16 pixels via the best H.265 mode in terms of PSNR and PNNS: (a) H.265 causal neighborhood, (b) PNNS context, (c) block to be predicted, (d) predicted block via the best H.265 mode (of index 29) in terms of PSNR, and (e) predicted block via PNNS.

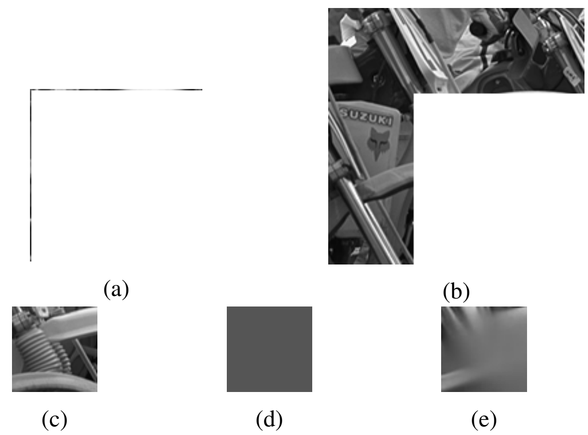


Fig. 11: Prediction of a block of size 64×64 pixels via the best H.265 mode in terms of PSNR and PNNS: (a) H.265 causal neighborhood, (b) PNNS context, (c) block to be predicted, (d) predicted block via the best H.265 mode (DC) in terms of PSNR, and (e) predicted block via PNNS.

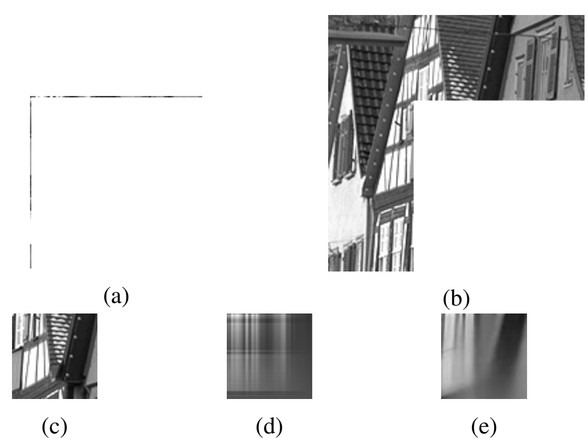


Fig. 12: Prediction of a block of size 64×64 pixels via the best H.265 mode in terms of PSNR and PNNS: (a) H.265 causal neighborhood, (b) PNNS context, (c) block to be predicted, (d) predicted block via the best H.265 mode (planar) in terms of PSNR, and (e) predicted block via PNNS.

TABLE III: Comparison of $\text{PSNR}_{\text{PNNS},m}$ and $\text{PSNR}_{\text{IPFCN-S},m}$ [14] for $m \in \{4, 8, 16, 32, 64\}$. f_4, f_8, g_{16}, g_{32}, and g_{64} are trained via random context masking. During the test phase, $n_0 = n_1 = 0$.

m	$\text{PSNR}_{\text{PNNS},m}$	$\text{PSNR}_{\text{IPFCN-S},m}$ [14]
4	**34.57**	33.70
8	**32.01**	31.44
16	**29.25**	28.71
32	**25.52**	24.96
64	21.47	–

TABLE IV: Comparison of success rates in percentage $\mu_{\text{PNNS},m}$ and $\mu_{\text{IPFCN-S},m}$ [14], for $m \in \{4, 8, 16, 32, 64\}$. f_4, f_8, g_{16}, g_{32}, and g_{64} are trained via random context masking. During the test phase, $n_0 = n_1 = 0$.

m	$\mu_{\text{PNNS},m}$	$\mu_{\text{IPFCN-S},m}$ [14]
4	**19**%	14%
8	**31**%	26%
16	**54**%	35%
32	**60**%	41%
64	67%	–

i.e. the rate at which IPFCN-S provides a better prediction in terms of PSNR than any other prediction of H.265, the difference $\mu_{\text{PNNS},m} - \mu_{\text{IPFCN-S},m}$ increases with m (see Table IV). This shows that convolutional neural networks are more appropriate than fully-connected ones in terms of prediction PSNR for large block sizes. Note that, in Tables III and IV, there is no comparison for block width 64 pixels as this block width is not considered in [14].

F. Justification of fully-connected neural networks for predicting small image blocks

Now that the benefit of using convolutional neural networks rather than fully-connected ones for predicting large image blocks is shown, the choice of fully-connected neural networks for predicting small blocks must be justified. Table V displays the average PSNR between the image block to be predicted and its prediction over the 960 blocks for both a fully-connected architecture and a convolutional one for each block width $m \in \{4, 8, 16\}$. The two fully-connected architectures f_4 and f_8 are already presented in Figure 2. The fully-connected architecture f_{16} is also that in Figure 2. The three convolutional architectures g_4, g_8, and g_{16} are shown in Table XIII in Appendix A. We see that the convolutional neural network provides better prediction PSNRs on average than its fully-connected counterpart only when $m > 8$. This justifies why f_4 and f_8 are selected for predicting blocks of width 4 and 8 pixels respectively.

G. Different objective functions

In the objective function to be minimized over the neural network parameters, the distortion term involves the $L2$ norm of the difference between the image block to be predicted and its estimate (see (2)). We also tried an alternative. This alternative consisted in replacing the $L2$ norm by the $L1$ norm. The choice of the $L1$ norm could be justified by the fact

TABLE V: Comparison of the average prediction PSNR given by the fully-connected architecture and that given by the convolutional one for different pairs $\{n_0, n_1\}$ during the test phase: (a) $\text{PSNR}_{\text{PNNS},4}$ given by f_4 and g_4, (b) $\text{PSNR}_{\text{PNNS},8}$ given by f_8 and g_8, and (c) $\text{PSNR}_{\text{PNNS},16}$ given by f_{16} and g_{16}. All neural networks are trained by drawing n_0 and n_1 uniformly from the set $\mathcal{U}\{0, 4, 8, 12, ..., m\}$.

Test $\{n_0, n_1\}$	f_4	g_4
$\{0, 0\}$	34.57	33.82
$\{0, 4\}$	34.42	33.66
$\{4, 0\}$	34.39	33.79
$\{4, 4\}$	34.20	33.54

(a)

Test $\{n_0, n_1\}$	f_8	g_8
$\{0, 0\}$	31.79	31.40
$\{0, 8\}$	31.66	31.41
$\{8, 0\}$	31.68	31.32
$\{8, 8\}$	31.56	31.32

(b)

Test $\{n_0, n_1\}$	f_{16}	g_{16}
$\{0, 0\}$	29.07	29.25
$\{0, 16\}$	28.95	29.11
$\{16, 0\}$	28.95	29.12
$\{16, 16\}$	28.86	28.97

(c)

that, in H.265, for a given image block to be predicted, the criterion for selecting several intra prediction modes among all modes involves the sum of absolute differences between the image block to be predicted and its estimate[3]. But, this alternative distortion term did not yield any increase of the mean prediction PSNRs compared to those shown in Table I.

A common alternative to the regularization by $L2$ norm of the weights is the regularization by $L1$ norm of the weights [35]. Both approaches have been tested and it was observed that the regularization by $L2$ norm of the weights slightly reduces the mean prediction error on a validation set at the end of the training.

V. Signalling of the Prediction Modes in H.265

Before moving on to the experiments in Section VI where PNNS is integrated into a H.265 codec, the last issue is the signalling of the prediction modes inside H.265. Indeed, the integration of PNNS into H.265 requires to revisit the way all modes are signalled. Section V describes two ways of signalling PNNS into H.265. The purpose of setting up these two ways is to identify later on which signalling yields the largest PSNR-rate performance gains and discuss why a difference between them exists. The first signalling is the substitution of a H.265 intra prediction mode with PNNS. The second signalling is a switch between PNNS and the H.265 intra prediction modes.

A. Substitution of a H.265 intra prediction mode with PNNS

Section V-A first describes how H.265 selects the best of its 35 intra prediction modes for predicting a given image block. Based on this, a criterion for finding the mode to be replaced with PNNS is built.

To select the best of its 35 intra prediction modes for predicting a given image block, H.265 proceeds in two steps. During the first step, the 35 modes compete with one another. Each mode takes the causal neighborhood of the block to compute a different prediction of the block. The sum of absolute differences between the input block and its prediction is linearly combined with the mode signalling cost, yielding the mode “fast” cost [2]. The modes associated to a given number of lowest “fast” costs are put into a “fast” list [3]. During the second step, only the modes in the “fast” list compete with one another. The mode with the lowest rate-distortion cost is the best mode.

Knowing this, it seems natural to replace the mode that achieves the least frequently the lowest rate-distortion cost. Therefore, the frequency of interest $\nu_{\overline{m}} \in [0, 1], \overline{m} \in \{4, 8, 16, 32, 64\}$, is the number of times a mode has the lowest rate-distortion cost when $m = \overline{m}$ divided by the number of times the above-mentioned selection process is run when $m = \overline{m}$. To be generic, $\nu_{\overline{m}}$ should not be associated to luminance images of a specific type. That is why 100 380×480 luminance crops extracted from the BSDS300 dataset [43] and 100 1200×1600 luminance crops extracted from the INRIA Holidays dataset [44] are encoded with H.265 to compute $\nu_{\overline{m}}$. In this case, the mode of index 18 has on average the lowest $\nu_{\overline{m}}$ when $\overline{m} \in \{4, 16, 32, 64\}$ (see Figure 13). Note that this conclusion is verified with QP $\in \{22, 27, 32, 37, 42\}$. Note also that statistics about the frequency of selection of each H.265 intra prediction mode have already been analyzed [45], [46]. But, they are incomplete for our case as they take into account few videos and do not differentiate each value of $\overline{m}$. Thus, PNNS replaces the H.265 mode of index 18.

As explained thereafter, the signalling cost the H.265 intra prediction mode of index 18 is variable and can be relatively large. When substituting the H.265 intra prediction mode of index 18 with PNNS, this variable cost transfers to PNNS. In contrast, the purpose of the second signalling of PNNS inside H.265 is to induce a fixed and relatively low signalling cost of PNNS.

B. Switch between PNNS and the H.265 intra prediction modes

The authors in [14] propose to signal the neural network mode with a single bit. This leads to the signalling of the modes summarized in Table VI. In addition, we modify the process of selecting the 3 Most Probable Modes (MPMs) [47] of the current PB to make the signalling of the modes even more efficient. More precisely, if the neural network mode is the mode selected for predicting the PB above the current PB or the PB on the left side of the current PB, then the neural network mode belongs to the MPMs of the current PB. As a result, redundancy appears as the neural network mode has not only the codeword 1 but also the codeword of a MPM. That is why, in the case where PNNS belongs to the MPMs of the current PB, we substitute each MPM being PNNS with either planar, DC or the vertical mode of index 26 such that the 3 MPMs of the current PB are different from each other. Besides, planar takes priority over DC, DC having priority over the vertical mode of index 26. See the code[1] for further details regarding this choice.

[2]“fast” stresses that the cost computation is relatively low.

[3]See “TEncSearch::estIntraPredLumaQT” at https://hevc.hhi.fraunhofer.de/trac/hevc/browser/trunk/source/Lib/TLibEncoder/TEncSearch.cpp

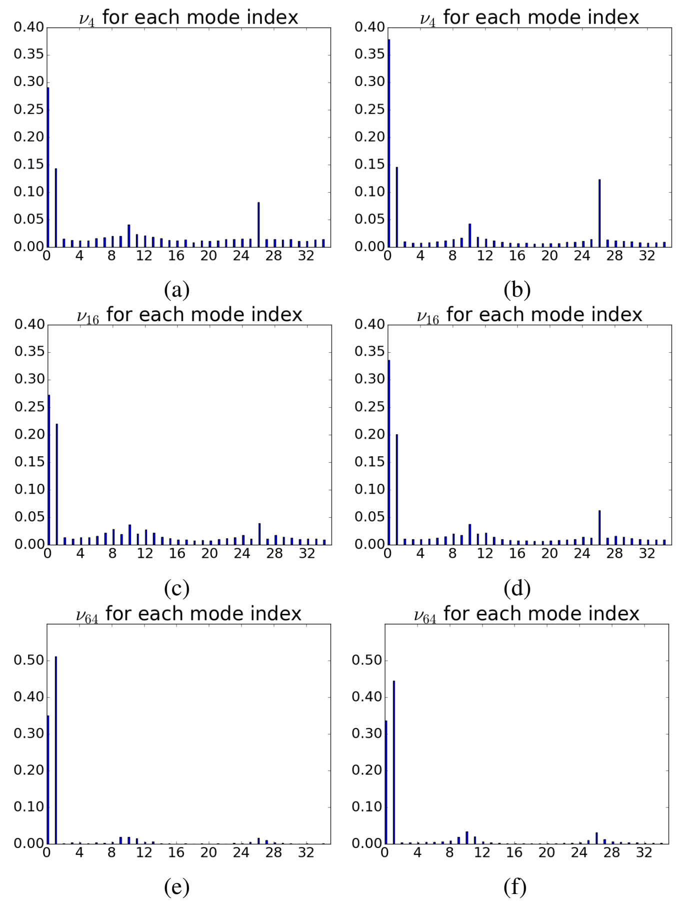


Fig. 13: Analysis of (a, b) ν_4, (c, d) ν_{16}, and (e, f) ν_{64} for each mode. 100 luminance crops from either (a, c, e) the BSDS300 dataset or (b, d, f) the INRIA Holidays dataset are encoded via H.265 with QP = 32.

TABLE VI: Signalling of the modes described in [14].

Mode	codeword
Neural network mode	1
First MPM	010
Second MPM	0110
Third MPM	0111
Non-MPM and non-neural network mode	00 {5bits}

The key remark concerning Section V is that there is a tradeoff between the signalling cost of PNNS versus the signalling cost of the H.265 intra prediction modes. Indeed, the substitution (see Section V-A) keeps the signalling cost of the H.265 intra prediction modes constant but the signalling cost of PNNS can be up to 6 bits. Instead, the switch decreases the signalling cost of PNNS and raises that of each H.265 intra prediction mode.

VI. Experiments

Now that two ways of signalling PNNS inside H.265 are specified, these two ways can be compared in terms of PSNR-rate performance gains. Moreover, PNNS integrated into H.265 can be compared to IPFCN-S integrated into H.265 in terms of PSNR-rate performance gains.

TABLE VII: PSNR-rate performance gains compared with H.265 of "PNNS substitution" and "PNNS switch" for the first test set.

Image name	PSNR-rate performance gain			
	"PNNS substitution"			"PNNS switch"
	Low rate	High rate	All rates	All rates
Barbara	2.47%	1.31%	1.79%	2.77%
Lena	1.68%	2.11%	2.05%	3.78%
Mandrill	0.77%	0.58%	0.67%	1.46%
Peppers	1.61%	1.50%	1.71%	3.63%

A. Experimental settings

The H.265 HM16.15 software is used in all the following experiments. The configuration is all-intra main. Note that the following settings only mention the integration of PNNS into H.265 via the substitution of the H.265 intra prediction mode of index 18 with PNNS, so called "PNNS substitution". But, the same settings apply to the integration of PNNS into H.265 via the switch between PNNS and the H.265 intra prediction modes, so called "PNNS switch". The PSNR-rate performance of "PNNS substitution" with respect to H.265 is computed using the Bjontegaard metric [48], which is the average saving in bitrate of the rate-distortion curve of "PNNS substitution" with respect to the rate-distortion curve of H.265. It is interesting to analyze whether there exists a range of bitrates for which "PNNS substitution" is more beneficial. That is why 3 different ranges of bitrates are presented. The first range, called "low rate", refers to QP $\in \{32, 34, 37, 39, 42\}$. The second range, called "high rate", corresponds to QP $\in \{17, 19, 22, 24, 27\}$. The third range, called "all rates", computes the Bjontegaard metric with the complete set of QP values from 17 to 42. The H.265 common test condition [49] recommends $\{22, 27, 32, 37\}$ as QPs setting. Yet, we add several QPs to the recommended setting. This is because, to compute the Bjontegaard metric for "low rate" for instance, a polynom of degree 3 is fitted to rate-distortion points, and at least 4 rate-distortion points, i.e. 4 different QPs, are required to get a good fit.

Four test sets are used to cover a wide variety of images. The first test set contains the luminance channels of respectively Barbara, Lena, Mandrill and Peppers. The second test set contains the 24 RGB images in the Kodak suite, converted into luminance. The third test set gathers the 13 videos sequences of the classes B, C, and D of the H.265 CTC, converted into luminance. The fourth test set contains 6 widely used videos sequences [4], converted into luminance. Our work is dedicated to image coding. That is why only the first frame of each video sequence in the third and fourth test sets are considered. It is important to note that the training in Section IV, the extraction of the frequency of selection of each H.265 intra prediction mode in Section V, and the current experiments involve 7 distinct sets of luminance images. This way, PNNS is not tuned for any specific test luminance image.

[4]ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences/

TABLE VIII: PSNR-rate performance gains compared with H.265 of "PNNS substitution" and "PNNS switch" for the second test set.

Kodak image index	PSNR-rate performance gain			
	"PNNS substitution"			"PNNS switch"
	Low rate	High rate	All rates	All rates
1	1.20%	0.74%	0.95%	2.06%
2	0.59%	0.91%	0.88%	2.16%
3	0.91%	2.04%	1.59%	3.22%
4	1.78%	1.75%	1.80%	3.23%
5	1.40%	2.45%	2.08%	4.01%
6	1.43%	0.81%	1.12%	2.11%
7	1.12%	2.36%	1.76%	3.86%
8	1.01%	0.83%	0.98%	1.79%
9	1.54%	1.43%	1.46%	3.05%
10	2.20%	2.37%	2.42%	3.84%
11	0.93%	0.91%	1.00%	2.41%
12	0.96%	1.02%	1.07%	2.33%
13	0.83%	0.5%	0.64%	1.76%
14	0.96%	1.28%	1.20%	2.76%
15	1.53%	1.19%	1.37%	2.62%
16	0.66%	0.62%	0.70%	1.69%
17	1.35%	2.03%	1.80%	3.69%
18	0.68%	0.96%	0.88%	1.98%
19	1.44%	0.86%	1.05%	2.06%
20	0.92%	1.61%	1.38%	2.71%
21	0.99%	0.83%	0.94%	2.28%
22	0.56%	0.88%	0.78%	2.22%
23	1.20%	2.45%	2.03%	4.20%
24	0.68%	0.87%	0.80%	1.73%

TABLE IX: PSNR-rate performance gains compared with H.265 of "PNNS substitution" and "PNNS switch" for the third test set.

Video sequence		PSNR-rate performance gain			
		"PNNS substitution"			"PNNS switch"
		Low rate	High rate	All rates	All rates
B	BQTerrace	1.66%	0.95%	1.29%	2.44%
	BasketballDrive	4.80%	2.87%	3.65%	5.20%
	Cactus	1.48%	1.51%	1.58%	3.05%
	ParkScene	0.64%	1.16%	0.97%	2.58%
	Kimono	1.28%	1.55%	1.65%	2.92%
C	BQMall	1.20%	1.30%	1.30%	3.14%
	BasketballDrill	−1.18%	1.34%	0.39%	3.50%
	RaceHorsesC	1.34%	1.58%	1.53%	3.29%
	PartyScene	1.02%	0.91%	0.96%	2.42%
D	BQSquare	0.79%	0.86%	0.86%	2.21%
	BasketballPass	1.61%	1.80%	1.48%	3.08%
	BlowingBubbles	0.66%	1.22%	1.02%	2.65%
	RaceHorses	1.32%	1.63%	1.54%	3.28%

TABLE X: PSNR-rate performance gains compared with H.265 of "PNNS substitution" and "PNNS switch" for the fourth test set.

Video sequence	PSNR-rate performance gain			
	"PNNS substitution"			"PNNS switch"
	Low rate	High rate	All rates	All rates
Bus	1.67%	1.17%	1.45%	2.74%
City	1.55%	1.19%	1.35%	2.51%
Crew	1.56%	1.24%	1.38%	3.10%
Football	1.44%	1.78%	1.78%	3.52%
Harbour	1.80%	0.73%	1.25%	2.51%
Soccer	0.96%	0.95%	1.03%	1.90%

B. Analysis of the two ways of signalling the PNNS mode inside H.265

The most striking observation is that the PSNR-rate performance gains generated by "PNNS switch" are always larger than those provided by "PNNS substitution" (see Tables VII, VIII, IX, and X). This has two explanations. Firstly, "PNNS substitution" is hampered by the suppression of the original H.265 intra prediction mode of index 18. Indeed, the PSNR-rate performance gain is degraded when the original H.265 intra prediction mode of index 18 is a relevant mode for encoding a luminance image. The most telling example is the luminance channel of the first frame of "BasketballDrill". When this channel is encoded via H.265, for the original H.265 intra prediction mode of index 18, $\nu_4 = 0.085$, $\nu_8 = 0.100$, $\nu_{16} = 0.116$, $\nu_{32} = 0.085$, and $\nu_{64} = 0.088$. This means that, compared to the average statistics in Figure 13, the original H.265 intra prediction mode of index 18 is used approximatively 10 times more frequently. This explains why the PSNR-rate performance gain provided by "PNNS substitution" is only 0.39% (see Table IX). The other way round, the luminance channel of the first frame of "BasketballDrive" is an insightful example. When this channel is encoded via H.265, for the original H.265 intra prediction mode of index 18, $\nu_4 = 0.004$, $\nu_8 = 0.005$, $\nu_{16} = 0.004$, $\nu_{32} = 0.004$, and $\nu_{64} = 0.000$. In this case, the original H.265 intra prediction mode of index 18 is almost never used. "PNNS substitution" thus yields 3.65% of PSNR-rate performance gain.

There is another explanation for the gap in PSNR-rate performance gain between "PNNS substitution" and "PNNS switch". As shown in Section IV-E, PNNS is able to model a large set of complex textures found in large image blocks. PNNS is also able to model a large set of simple textures found in small blocks. Following the principle of Huffman Coding, an intra prediction mode that gives on average predictions of good quality, such as PNNS, should be signalled using fewer bits. However, an intra prediction mode that seldom yields the highest prediction quality, such as the H.265 intra prediction mode of index 4, should be signalled using more bits. This corresponds exactly to the principle of the switch between PNNS and the H.265 intra prediction modes. Therefore, "PNNS switch" beats "PNNS substitution" in terms of PSNR-rate performance gains. Figures 14 and 15 each compares the reconstruction of a luminance image via H.265 and its reconstruction via "PNNS switch" at similar reconstruction PSNRs. More visual comparisons are available on the website[5].

Another interesting conclusion emerges when comparing "low rate" and "high rate". There is no specific range of bitrate for which "PNNS substitution" is more profitable (see Tables VII, VIII, IX, and X). Note that, in few cases, the PNSR-rate performance gain in "all rates" is slightly larger than those in "low rate" and "high rate". This happens when the area between the rate-distortion curve of "PNNS substitution" and the rate-distortion curve of H.265 gets relatively large in the range QP $\in [27, 32]$.

[5]https://www.irisa.fr/temics/demos/prediction_neural_network/PredictionNeuralNetwork.htm

C. Comparison with the state-of-the-art

Now, "PNNS switch" is compared to IPFCN-S integrated into H.265 in terms of PSNR-rate performance gains. It is important to note that the authors in [14] develop two versions of their set of 4 fully-connected neural networks for intra prediction. The first version, called IPFCN-S, is the one used in Section IV-E. The 4 fully-connected neural networks are trained on an unconstrained training set of image blocks to be predicted, each paired with its context. The second version is called Intra Prediction Fully-Connected Networks Double (IPFCN-D). The training data are dissociated into two groups. One group gathers image blocks exhibiting textures with angular directions, each paired with its context. The other group gathers image blocks exhibiting textures with non-angular directions, each paired with its context. In IPFCN-D, there are two sets of 4 fully-connected neural networks, each set being trained on a different group of training data. Then, the two sets are integrated into H.265. IPFCN-D gives slightly larger PSNR-rate performance gains than IPFCN-S. The comparison below involves IPFCN-S as our training set is not dissociated. But, this dissociation could also be applied to the training set of the neural networks of PNNS.

"PNNS switch" and IPFCN-S integrated into H.265 are compared on the third test set. The PSNR-rate performance gains of IPFCN-S are reported from [14]. We observe that the PSNR-rate performance gains of "PNNS switch" are larger than those of IPFCN-S integrated into H.265, apart from the case of the video sequence "ParkScene" (see Table XI). Note that, for several videos sequences, the difference in PSNR-rate performance gains between "PNNS switch" and IPFCN-S integrated into H.265 is significant. For instance, for the video sequence "BasketballPass", the PSNR-rate performance gain of "PNNS switch" is 3.08% whereas that of IPFCN-S integrated into H.265 is 1.1%. Therefore, the use of both fully-connected neural networks and convolutional neural networks for intra prediction, the training with random context masking and the training data augmentation for training the convolutional neural networks of PNNS help boost the PSNR-rate performance gains. This is consistent with the conclusion in Section IV-E. Note that, even when comparing the PSNR-rate performance gains of "PNNS switch" with those of IPFCN-D integrated into H.265 which are reported in [14], "PNNS switch" often yields larger gains.

D. Robustness of the neural networks to quantization noise in their input context

In this section, we show that the proposed PNNS performs well with test data compressed with different QP values despite the fact that the neural networks of PNNS have been trained only once with no quantization noise in their input context. Indeed, let us consider two different "PNNS switch". In the first "PNNS switch", our 5 neural networks, one for each block size, are dedicated to all QPs. Note that the first "PNNS switch" corresponds to the "PNNS switch" that has been used so far. In the second "PNNS switch", a first set of 5 neural networks is dedicated to QP ≤ 27 whereas a second set is dedicated to QP > 27. Unlike the first set of neural networks,

TABLE XI: PSNR-rate performance gains of our proposed "PNNS switch" and IPFCN-S [14] inside H.265 for the third test set. The reference is H.265.

Video sequence		PSNR-rate performance gain	
		our "PNNS switch"	IPFCN-S [14] inside H.265
B	BQTerrace	**2.44%**	1.8%
	BasketballDrive	**5.20%**	3.4%
	Cactus	**3.05%**	2.7%
	ParkScene	2.58%	**2.8%**
	Kimono	**2.92%**	2.7%
C	BQMall	**3.14%**	2.0%
	BasketballDrill	**3.50%**	1.1%
	RaceHorsesC	**3.29%**	2.9%
	PartyScene	**2.42%**	1.3%
D	BQSquare	**2.21%**	0.6%
	BasketballPass	**3.08%**	1.1%
	BlowingBubbles	**2.65%**	1.6%
	RaceHorses	**3.28%**	2.8%

TABLE XII: Average computation time ratio with respect to H.265.

	our "PNNS switch"	IPFCN-S [14] inside H.265
Encoding	51	46
Decoding	191	190

the second set is trained on contexts that are encoded and decoded via H.265 with QP $\sim \mathcal{U}\{32, 37, 42\}$ for each training context. For the third test set, the difference in PSNR-rate performance gain between the first "PNNS switch" and the second "PNNS switch" ranges between 0.0% and 0.1%. This means that there is no need to train the neural networks of PNNS on contexts with quantization noise.

E. Complexity

A fully-connected neural network needs an overcomplete representation to provide predictions with high quality. That is why the number of neurons in each fully-connected layer is usually much larger than the size of the context. Likewise, the number of feature maps in each convolutional layer of a convolutional neural network is usually large. This incurs a high computational cost. Table XII gives the encoding and decoding times for "PNNS switch" and IPFCN-S and shows comparable running times for both solutions. A Bi-Xeon CPU E5-2620 is used for "PNNS switch".

F. Memory consumption

In addition to the complexity, another issue arising from the integration of PNNS into H.265 is the increase of the memory consumption. For a regular intra prediction mode in H.265, the context around the image block to be predicted consists of one row of $2m+1$ pixels above the block and one column of $2m$ pixels on the left side of the block. However, for a neural network of PNNS, the context around the image block consists of m rows of $3m$ pixels above the block and m columns of $2m$ pixels on the left side of the block (see Figure 5). Therefore, the hardware must be adapted to handle the large memory required by the input contexts to the neural networks of PNNS.

VII. Conclusion

This paper has presented a set of neural network architectures, including both fully-connected neural networks and convolutional neural networks, for intra prediction. It is shown that fully-connected neural networks are well adapted to the prediction of image blocks of small sizes whereas convolutional ones provide better predictions in large blocks. Our neural networks are trained via a random context masking of their context so that they adapt to the variable number of available decoded pixels for the prediction in a coding scheme. When integrated into a H.265 codec, the proposed neural networks are shown to give rate-distortion performance gains compared with the H.265 intra prediction. Moreover, it is shown that these neural networks can cope with the quantization noise present in the prediction context, i.e. they can be trained on undistorted contexts, and then generalize well on distorted contexts in a coding scheme. This greatly simplifies training as quantization noise does not need to be taken into account during training.

Appendix A
Convolutional architectures for H.265

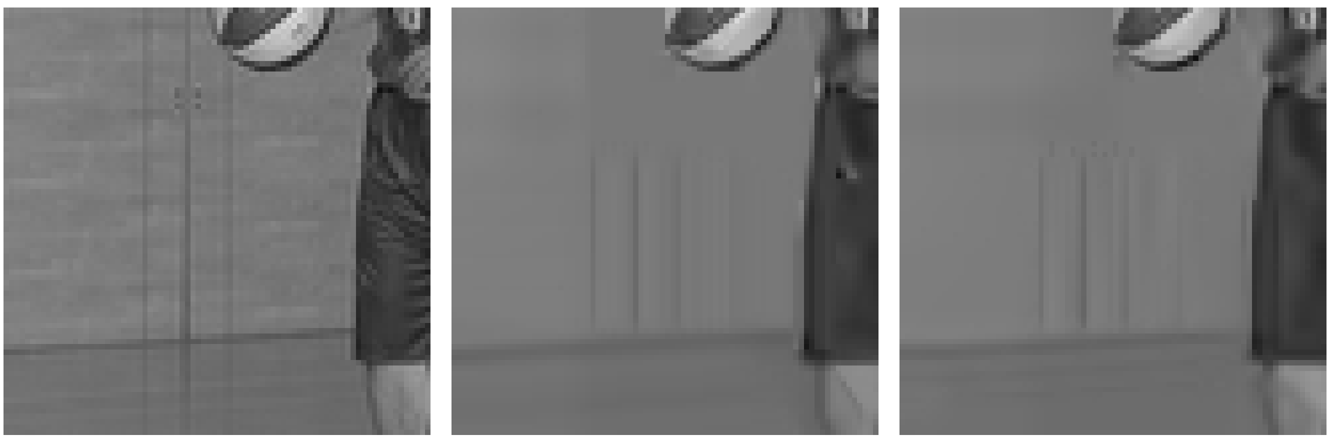
The architecture of the stack of convolutional layers $g_m^c\left(\,.\,;\boldsymbol{\phi}_m^{c,0}\right)$ that is applied to the context portion $\mathbf{X}_0$ located on the left side of the image block to be predicted is shown in Table XIII for $m \in \{4, 16, 64\}$. The architecture of the stack of convolutional layers $g_m^c\left(\,.\,;\boldsymbol{\phi}_m^{c,1}\right)$ that is applied to the context porton $\mathbf{X}_1$ located above the image block is identical to that of $g_m^c\left(\,.\,;\boldsymbol{\phi}_m^{c,0}\right)$. The slope of LeakyReLU is equal to 0.1.

The architecture of $g_8^c\left(\,.\,;\boldsymbol{\phi}_8^{c,0}\right)$ is similar to that of $g_4^c\left(\,.\,;\boldsymbol{\phi}_4^{c,0}\right)$ but, in the first layer, the filter size is $5 \times 5 \times 1$, the number of filters is 64, and the stride is 2. Moreover, in the second layer, the filter size is $3 \times 3 \times 64$ and the number of filters is 64. The architecture of $g_{32}^c\left(\,.\,;\boldsymbol{\phi}_{32}^{c,0}\right)$ is similar to that of $g_{64}^c\left(\,.\,;\boldsymbol{\phi}_{64}^{c,0}\right)$ but, in the third layer, the filter size is $3 \times 3 \times 128$, the number of filters is 128, and the stride is 1. Moreover, in the fourth layer, the filter size is $5 \times 5 \times 128$ and the number of filters is 256. In the fifth layer, the filter size is $3 \times 3 \times 256$ and the number of filters is 256.

To obtain the architecture of g_m^t, each sequence of layers in g_m^c is reversed. Besides, each convolution is replaced by a transposed convolution (see Table XIII).

(a) (b) (c)

Fig. 14: Comparison of (a) a 100×100 crop of the luminance channel of the first frame of BQMall, (b) its reconstruction via H.265, and (c) its reconstruction via "PNNS switch". QP $= 27$. For the luminance channel of the first frame of BQMall, for H.265, $\{\text{rate} = 0.670 \text{ bpp}, \text{PSNR} = 38.569 \text{ dB}\}$. For "PNNS switch", $\{\text{rate} = 0.644 \text{ bpp}, \text{PSNR} = 38.513 \text{ dB}\}$.



(a) (b) (c)

Fig. 15: Comparison of (a) a 100×100 crop of the luminance channel of the first frame of BasketballPass, (b) its reconstruction via H.265, and (c) its reconstruction via "PNNS switch". QP $= 37$. For the luminance channel of the first frame of BasketballPass, for H.265, $\{\text{rate} = 0.167 \text{ bpp}, \text{PSNR} = 32.386 \text{ dB}\}$. For "PNNS switch", $\{\text{rate} = 0.160 \text{ bpp}, \text{PSNR} = 32.407 \text{ dB}\}$.

REFERENCES

[1] M. Wien, "High efficiency video coding: coding tools and specification," *Springer*, September 2014.

[2] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur, "Intra coding of the HEVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1792–1801, December 2012.

[3] A. J. Bell and T. J. Sejnowski, "Edges are the independent components of natural scenes," in *NIPS*, 1996.

[4] J. H. van Hateren and A. van der Schaaf, "Independent component filters of natural images compared with simple cells in primary visual cortex," *Proc. R. Soc. Lond. B*, vol. 265, no. 1394, pp. 359–366, March 1998.

[5] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area V2," in *NIPS*, 2007.

[6] H. Hosoya and A. Hyvärinen, "A hierarchical statistical model of natural images explains tuning properties in V2," *The Journal of Neuroscience*, vol. 35, no. 29, pp. 10 412–10 428, July 2015.

[7] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *ICML*, 2009.

[8] M. Ranzato, V. Mnih, J. M. Susskind, and G. E. Hinton, "Modeling natural images using gated MRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2206–2222, September 2013.

[9] T. K. Tan, C. S. Boon, and Y. Suzuki, "Intra prediction by template matching," in *ICIP*, 2006.

[10] M. Turkan and C. Guillemot, "Image prediction based on neighbor embedding methods," *IEEE Transaction on Image Processing*, vol. 21, no. 4, pp. 1885–1898, April 2012.

[11] T. Wiegand and H. Schwarz, "Source coding: part I of fundamentals of source and video coding," *Foundations and Trends in Signal Processing*, vol. 4, nos. 1-2, pp. 1–222, 2011.

[12] L. Theis and M. Bethge, "Generative image modeling using spatial LSTMs," in *NIPS*, 2015.

[13] A. van der Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *ICML*, 2016.

[14] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao, "Fully-connected network-based intra prediction for image coding," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3236–3247, July 2018.

[15] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory,*, vol. 44, no. 6, pp. 2325–2383, October 1998.

[16] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra, "Video (language) modeling: a baseline for generative models of natural videos," *arXiv:1412.6604v4*, April 2015.

[17] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *ICLR*, 2016.

[18] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," *Neural Networks: Tricks of the Trade*, Springer, 1998.

TABLE XIII: Architectures of the stack of convolutional layers $g_m^c\left(\,.\, ; \phi_m^{c,0} \right)$ that is applied to the context portion $\mathbf{X}_0$ located on the left side of the image block to be predicted ((a) $m = 4$, (c) $m = 16$, and (e) $m = 64$) and the stack of transposed convolutional layers $g_m^t\left(\,.\, ; \phi_m^t \right)$ ((b) $m = 4$, (d) $m = 16$, and (f) $m = 64$). "conv" means convolution and "tconv" means transposed convolution.

Layer number	Layer type	Filter size	Number of filters	Stride	Non-linearity
1	conv	$3 \times 3 \times 1$	32	1	LeakyReLU
2	conv	$3 \times 3 \times 32$	32	1	LeakyReLU

(a)

Layer number	Layer type	Filter size	Number of filters	Stride	Non-linearity
1	tconv	$3 \times 3 \times 32$	32	1	LeakyReLU
2	tconv	$3 \times 3 \times 32$	1	1	-

(b)

Layer number	Layer type	Filter size	Number of filters	Stride	Non-linearity
1	conv	$5 \times 5 \times 1$	64	2	LeakyReLU
2	conv	$3 \times 3 \times 64$	64	1	LeakyReLU
3	conv	$5 \times 5 \times 64$	128	2	LeakyReLU
4	conv	$3 \times 3 \times 128$	128	1	LeakyReLU

(c)

Layer number	Layer type	Filter size	Number of filters	Stride	Non-linearity
1	tconv	$3 \times 3 \times 128$	128	1	LeakyReLU
2	tconv	$5 \times 5 \times 128$	64	2	LeakyReLU
3	tconv	$3 \times 3 \times 64$	64	1	LeakyReLU
4	tconv	$5 \times 5 \times 64$	1	2	-

(d)

Layer number	Layer type	Filter size	Number of filters	Stride	Non-linearity
1	conv	$5 \times 5 \times 1$	64	2	LeakyReLU
2	conv	$5 \times 5 \times 64$	128	2	LeakyReLU
3	conv	$5 \times 5 \times 128$	256	2	LeakyReLU
4	conv	$5 \times 5 \times 256$	512	2	LeakyReLU
5	conv	$3 \times 3 \times 512$	512	1	LeakyReLU

(e)

Layer number	Layer type	Filter size	Number of filters	Stride	Non-linearity
1	tconv	$3 \times 3 \times 512$	512	1	LeakyReLU
2	tconv	$5 \times 5 \times 512$	256	2	LeakyReLU
3	tconv	$5 \times 5 \times 256$	128	2	LeakyReLU
4	tconv	$5 \times 5 \times 128$	64	2	LeakyReLU
5	tconv	$5 \times 5 \times 64$	1	2	-

(f)

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[20] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[21] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval," in *ESANN*, 2011.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[23] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.

[24] D. Pathak, P. Krähenbühl, J. Donahue, T. Darell, and A. A. Efros, "Context encoders: feature learning by inpainting," in *CVPR*, 2016.

[25] H. Hoh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *ICCV*, 2015.

[26] V. Dumoulin and F. Visin, "A guide to convolutional arithmetic for deep learning," *arXiv:1603.07285v2*, January 2018.

[27] Kodak suite. [Online]. Available: r0k.us/graphics/kodak/

[28] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, February 2016.

[29] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *CVPR Workshops*, 2017.

[30] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: a deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, December 2017.

[31] P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun, "Predicting deeper into the future of semantic segmentation," in *ICCV*, 2017.

[32] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung, "Phase-based frame interpolation for video," in *CVPR*, 2015.

[33] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *CVPR*, 2017.

[34] J. van Amersfoort, W. Shi, A. Acosta, F. Massa, J. Totz, Z. Wang, and J. Caballero, "Frame interpolation with multi-scale deep loss functions and generative adversarial networks," *arXiv:1711.06045v1*, November 2017.

[35] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Neural Networks: Tricks of the Trade*, Springer, 2013.

[36] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *ICLR*, 2015.

[37] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of AISTATS 2010*, vol 9., pp 249-256, May 2010.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: a large-scale hierarchical image database," in *CVPR*, 2009.

[39] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, December 2010.

[40] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *CVPR*, 2012.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[43] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV*, 2001.

[44] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometry consistency for large scale image search," in *ECCV*, 2008.

[45] R. I. Chernyak, "Analysis of the intra predictions in H.265/HEVC," *Applied Mathematical Sciences*, vol. 8, no. 148, pp. 7389–7408, 2014.

[46] O. Bougacha, H. Kibeya, N. Belhadj, M. A. Ben Ayed, and N. Masmoudi, "Statistical analysis of intra prediction in HEVC video encoder," in *IPAS*, 2016.

[47] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1667, December 2012.

[48] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," ITU-T SG16/Q6, Austin, TX, USA, Tech. Rep., 2001.

[49] C. Rosewarne, K. Sharman, and D. Flynn, "Common test conditions and software reference configurations for HEVC range extensions," document JCTVC-P1006, 16th meeting, JCT-VC, San Jose, CA, USA, January 2014.

Thierry Dumas received the Engineering degree and the Master degree from École Centrale Marseille, Marseille, France, in November 2014 and the Ph.D. degree from the University of Rennes I, France, in June 2019. He is now working as postdoctoral fellow at Interdigital, Rennes, France. His research interests include image and video compression.

Aline Roumy received the Engineering degree from École Nationale Supérieure de l'Éléctronique et de ses Applications (ENSEA), Cergy, France, in 1996, the Master degree in June 1997, and the Ph.D. degree from the University of Cergy-Pontoise, France, in September 2000. During 2000-2001, she was the recipient of a French Defense DGA/DRET post-doctoral fellowship and was a research associate at Princeton University, Princeton, New Jersey. In November 2001, she joined INRIA Rennes, France. Her current research and study interests include the area of statistical signal processing, coding theory, and information theory.

Christine Guillemot IEEE fellow, is Director of Research at INRIA, head of a research team dealing with image and video modeling, processing, coding and communication. She holds a Ph.D. degree from ENST (Ecole Nationale Supérieure des Télécommunications) Paris, and an Habilitation for Research Direction from the University of Rennes. From 1985 to October 1997, she has been with FRANCE TELECOM, where she has been involved in various projects in the area of image and video coding for TV, HDTV, and multimedia. From January 1990 to mid 1991, she has worked at Bellcore, NJ, USA, as a visiting scientist. Her research interests are signal and image processing, and in particular 2D and 3D image and video processing for various problems (compression, super-resolution, inpainting, classification).

She has served as Associate Editor for IEEE Trans. on Image Processing (from 2000 to 2003, and from 2014-2016), for IEEE Trans. on Circuits and Systems for Video Technology (from 2004 to 2006), and for IEEE Trans. on Signal Processing (2007-2009). She has served as senior member of the editorial board of the IEEE journal on selected topics in signal processing (2013-2015) and is currently senior area editor of IEEE Trans. on Image Processing.