Context-adaptive neural network based prediction for image compression
Thierry Dumas, Aline Roumy, Christine Guillemot

To cite this version:
Thierry Dumas, Aline Roumy, Christine Guillemot. Context-adaptive neural network based prediction for image compression. 2018. hal-01841034

HAL Id: hal-01841034
https://hal.archives-ouvertes.fr/hal-01841034
Submitted on 17 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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.

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 \( B \) of observed random variables. The authors in [11] demonstrate that the optimal prediction \( \hat{X}^\star \) of \( X \), i.e. the prediction that minimizes the mean squared prediction error, is the conditional expectation \( E[X|B] \). Yet, no existing model of natural images gives a reliable \( E[X|B] \).

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 \( 4 \times 4, 8 \times 8, 16 \times 16 \), and \( 32 \times 32 \) 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...
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.

1https://www.irisa.fr/temics/demos/prediction_neural_network/
PredictionNeuralNetwork.htm

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

$$
\begin{align*}
X_c &= X - \alpha \\
\hat{Y}_c &= \begin{cases} 
  f_m(X_c; \theta_m) \\
  g_m(X_c; \phi_m)
\end{cases} \\
\hat{Y} &= \max \left( \min \left( \hat{Y}_c + \alpha, 255 \right), 0 \right)
\end{align*}
$$

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 pixels intensity $\alpha$ is learned over the training set and subtracted from each image block to be predicted and its context during training. During the test phase, $\alpha$ 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 intensity is added to the output and the result is clipped to $[0, 255]$ (see (1)).
The first task of the convolutional architecture is the computation of features characterizing the dependencies between the elements in \( X_0 \). \( X_0 \) is thus fed into a stack of convolutional layers. This yields a stack \( Z_0 \) of \( l \in \mathbb{N}^* \) feature maps (see Figure 3). Similarly, \( X_1 \) is fed into another stack of convolutional layers. This yields a stack \( Z_1 \) 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. Unfortunately, convolutions only account for short-range spatial dependencies. That is why the next layer in the convolutional architecture merges spatially \( Z_0 \) and \( Z_1 \) (see Figure 3). More precisely, for \( i \in [1, l] \), all the coefficients of the \( i^{th} \) feature map of \( Z_0 \) and of the \( i^{th} \) feature map of \( Z_1 \) are merged through affine combinations. Then, LeakyReLU with slope 0.1 is applied, yielding the merged stack \( Z \) 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 \( Z \). \( Z \) is thus fed into a stack of transpose convolutional layers [25], [26]. This yields the predicted image block (see Figure 3). Note that all convolutional layers and transpose convolutional layers, apart from the last transpose convolutional layer, have LeakyReLU with slope 0.1 as non-linear activation. The last transpose 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 \( D \) 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 \( Y \) to develop intuitions regarding this tradeoff. When \( m \) is small, the long range spatial dependencies between these decoded pixels are not relevant for predicting \( Y \) (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 \( Y \). The size of the context should now be large, despite the issue. Therefore, the context size should be a function of \( m^3, q \geq 1 \).
Fig. 4: Dependencies between $D$ and $Y$. The luminance channel of the first image in the Kodak suite [27] is being encoded via H.265 with Quantization Parameter $QP = 17$.

From there, we ran several preliminary experiments in which $q \in \{1, 2, 3\}$ and the PSNR between $\hat{Y}$ and $Y$ 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, $X_0$ is a rectangle of height $2m$ and width $m$. $X_1$ 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 $m$.

The main distinction between fully-connected neural networks and convolutional neural networks lies in the fact that fully-connected neural networks do not profit from the statistical properties of natural images such as stationarity and multi-resolution structure whereas convolutional neural networks do [35], [36], [37]. As an example, in a convolutional layer, stationarity is leveraged by sharing the convolutional kernels across the input space, hence increasing the expressive capacity of the convolutional neural network for a given number of parameters. On the contrary, in a fully-connected layer, power of representation is wasted as some learned filters are simply translated version of other filters [38]. Therefore, a convolutional neural network is selected as the context exhibits the above-mentioned properties. However, when the block width $m$ is small, the size of the context is so small that the context has no multi-resolution structure. Therefore, there is less need to use a convolutional architecture. In this case, a fully-connected neural network is used as its setup is less complicated than that of the convolutional one. We observed that block widths $m \leq 8$ are well suited to fully-connected architectures, and larger widths to convolutional architectures.

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 $64 \times 64$ pixels. A CTB can be directly used as Coding Block (CB) or can be split into $4 \times 32 \times 32$ CBs. Then, each $32 \times 32$ 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 $8 \times 8$ 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 $4 \times 4 \times 4$ 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 partitioning, $m \in \{4, 8, 16, 32, 64\}$. 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 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 of pixels intensity over the training set so that, after subtracting it from 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 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 $4 \times 4 \times 4$ 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 partitioning, $m \in \{4, 8, 16, 32, 64\}$. 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 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 of pixels intensity over the training set so that, after subtracting it from 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 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 $4 \times 4 \times 4$ 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 partitioning, $m \in \{4, 8, 16, 32, 64\}$. 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 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 of pixels intensity over the training set so that, after subtracting it from 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 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 $4 \times 4 \times 4$ 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.
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 impractical 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 \(\theta_m\). Moreover, regularization by L2 norm of the weights (not the biases), denoted \([\theta_m]_W\), is applied [39] (see (2)).

\[
\min_{\theta_m} \mathbb{E}[\|Y_c - f_m(X_c; \theta_m)\|_2] + \lambda \|\theta_m\|_W^2
\]

(2)

The expectation \(\mathbb{E}[\cdot]\) is approximated by averaging over a training set of image blocks to be predicted, each paired with its context. \(\lambda = 0.0005\). For learning the parameters \(\phi_m\), (2) is used, replacing \(\theta_m\) with \(\phi_m\) and \(f_m\) with \(g_m\).

The optimization algorithm is ADAM [40] 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 [40] are used.

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 x 320 luminance crop \(I\) is, if possible, extracted from each RGB image in the ILSVRC2012 training set [41]. This yields a set \(\mathbf{T} = \{I^{(i)}\}_{i=1...1048717}\).

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 [42], [43], [44], [45] is used. In contrast, 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 \( \Gamma \) are exclusively randomly rotated and flipped. Precisely, for each step of ADAM, the scheme is Algorithm 1. \( s_{\text{rotation}} \) rotates its input image by angle \( \psi \in [0, 2\pi] \) radians. \( s_{\text{flipping}} \) flips its input image horizontally with probability 0.5. \( e_{\text{train}} \) is the same function as \( e_{\text{test}} \), except that \( e_{\text{train}} \) extracts \( \{X(i), Y(i)\} \) from potentially rotated and flipped \( I \) instead of extracting \( X \) from \( D \) and the position of the extraction is random instead of being defined by the order of decoding. For training a fully-connected neural network \( \{X(i)_{c}, Y(i)_{c}\}_{i=1,\ldots,10000000} \) is generated offline from \( \Gamma \), i.e. before the training starts.

Algorithm 1 Training data augmentation for the convolutional neural networks

**Inputs:** \( \Gamma, m, \alpha \).

\[
\forall i \in [1,100], \\
i \sim \mathcal{U}[1, 1048717] \\
\psi \sim \mathcal{U}\left\{0, \frac{\pi}{2}, \frac{3\pi}{2}\right\} \\
n_0, n_1 \sim \mathcal{U}\{0, 4, \ldots, m\} \\
J = s_{\text{flipping}}(s_{\text{rotation}}(1(|i|), \psi)) \\
\{X(i), Y(i)\} = e_{\text{train}}(J, m, n_0, n_1, \alpha) \\
X_{c}(i) = X(i) - \alpha \\
Y_{c}(i) = Y(i) - \alpha \\
\]

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.

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 width 4 pixels, \( f_4 \), and convolutional neural networks for block widths 16 and 64 pixels, \( g_{16} \) 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 PSNR, denoted \( \text{PSNR}_{\text{PNNS}, m} \), between the image block and its prediction via PNNS, averaged over the 960 blocks, 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.

<table>
<thead>
<tr>
<th>Test ( {n_0, n_1} )</th>
<th>Training ( f_4 ) with ( {n_0, n_1} )</th>
<th>Training ( g_{16} ) with ( {n_0, n_1} )</th>
<th>Training ( g_{64} ) with ( {n_0, n_1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {0, 0} )</td>
<td>34.63 34.39 34.44 34.23</td>
<td>29.25</td>
<td>21.47</td>
</tr>
<tr>
<td>( {0, 4} )</td>
<td>32.90 34.39 32.82 34.23</td>
<td>29.11</td>
<td>21.38</td>
</tr>
<tr>
<td>( {4, 0} )</td>
<td>32.79 32.68 34.44 34.23</td>
<td>29.12</td>
<td>21.34</td>
</tr>
<tr>
<td>( {4, 4} )</td>
<td>30.93 32.68 32.82 34.23</td>
<td>28.98</td>
<td>21.27</td>
</tr>
<tr>
<td>( {0, 0} )</td>
<td>29.25</td>
<td>29.11</td>
<td>21.47</td>
</tr>
<tr>
<td>( {0, 16} )</td>
<td>28.65 29.12 24.66 23.99</td>
<td>29.12</td>
<td>21.34</td>
</tr>
<tr>
<td>( {16, 0} )</td>
<td>28.43 22.60 29.06 22.99</td>
<td>29.12</td>
<td>21.34</td>
</tr>
<tr>
<td>( {16, 16} )</td>
<td>27.87 28.37 28.35</td>
<td>28.98</td>
<td>21.27</td>
</tr>
</tbody>
</table>

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.

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-
TABLE II: Comparison of success rates in percentage (a) $\mu_{\text{PNNS},4}$, (b) $\mu_{\text{PNNS},16}$ and (c) $\mu_{\text{PNNS},64}$ for different pairs $\{n_0, n_1\}$ during the training and test phases.

<table>
<thead>
<tr>
<th>Test ${n_0, n_1}$</th>
<th>Training $f_4$ with ${n_0, n_1}$</th>
<th>Training $g_{16}$ with ${n_0, n_1}$</th>
<th>Training $g_{64}$ with ${n_0, n_1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${0, 0}$</td>
<td>$22%$ $17%$ $19%$ $16%$ $19%$</td>
<td>$55%$ $18%$ $20%$ $9%$ $54%$</td>
<td>$68%$ $39%$ $43%$ $27%$ $67%$</td>
</tr>
<tr>
<td>${0, 4}$</td>
<td>$15%$ $18%$ $13%$ $15%$ $17%$</td>
<td>$42%$ $51%$ $18%$ $18%$ $50%$</td>
<td>$63%$ $64%$ $41%$ $44%$ $65%$</td>
</tr>
<tr>
<td>${4, 0}$</td>
<td>$11%$ $11%$ $20%$ $17%$ $19%$</td>
<td>$40%$ $15%$ $51%$ $17%$ $53%$</td>
<td>$62%$ $36%$ $66%$ $38%$ $68%$</td>
</tr>
<tr>
<td>${4, 4}$</td>
<td>$11%$ $12%$ $13%$ $16%$ $15%$</td>
<td>$33%$ $36%$ $40%$ $52%$ $49%$</td>
<td>$57%$ $61%$ $63%$ $64%$ $66%$</td>
</tr>
</tbody>
</table>

Fig. 7: Prediction of a block of size $16 \times 16$ 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 in terms of PSNR, and (e) predicted block via PNNS.

Fig. 8: Prediction of a block of size $16 \times 16$ 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.

Fig. 9: Prediction of a block of size $64 \times 64$ 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. 10: Prediction of a block of size $64 \times 64$ 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.

connected neural networks in this case. Figures 7, 8, 9, and 10 each compare 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 9, 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 10, 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. Therefore, unlike the H.265 intra prediction modes, the convolutional neural networks of PNNS can model a large set of complex textures found in large image blocks.
During the test phase, as this block width is not considered in [14].

III and IV, there is no comparison for block width prediction PSNR for large block sizes. Note that, in Tables g in intra prediction modes.

A difference between them exists. The first signalling is the signalling PNNS into H.265. The purpose of setting up these two ways is to identify later on which signalling yields 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 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 \(\nu_{m} \in [0, 1], m \in \{4, 8, 16, 32 64\}\), is the number of times a mode has the lowest rate-distortion cost when \(m = \overline{m}\). To be generic, \(\nu_{m}\) should not be associated to luminance images of a specific type. That is why 100 \(380 \times 480\) luminance crops extracted from the BSDS300 dataset [46] and 100 \(1200 \times 1600\) luminance crops extracted from the INRIA Holidays dataset [47] are encoded with H.265 to compute \(\nu_{m}\).

In this case, the mode of index 18 has on average the lowest \(\nu_{m}\) when \(m \in \{4, 16, 32, 64\}\) (see Figure 11). Note that this conclusion is verified with QP \(\in \{4, 8, 16, 32, 64\}\). Note also that statistics about the frequency of selection of each H.265 intra prediction mode have already been analyzed [48], [49]. 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.

### 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 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 \(\nu_{m} \in [0, 1], m \in \{4, 8, 16, 32, 64\}\), is the number of times a mode has the lowest rate-distortion cost when \(m = \overline{m}\). To be generic, \(\nu_{m}\) should not be associated to luminance images of a specific type. That is why 100 \(380 \times 480\) luminance crops extracted from the BSDS300 dataset [46] and 100 \(1200 \times 1600\) luminance crops extracted from the INRIA Holidays dataset [47] are encoded with H.265 to compute \(\nu_{m}\).

In this case, the mode of index 18 has on average the lowest \(\nu_{m}\) when \(m \in \{4, 16, 32, 64\}\) (see Figure 11). 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 [48], [49]. 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 V. In addition, we modify the process of selecting the 3 Most Probable Modes (MPMs) [50] 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

---

\(^{1}\)fast stress that the cost computation is relatively low.

\(^{2}\)This implies 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
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.

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.

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 [51], 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 ∈ {32, 34, 37, 39, 42}. The second range, called “high rate”, corresponds to QP ∈ {17, 19, 22, 24, 27}. The third range, called “full rate”, computes the Bjontegaard metric with the complete set of QP values from 17 to 42. The H.265 common test condition [52] 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 V: Signalling of the modes described in [14].

<table>
<thead>
<tr>
<th>Mode</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural network mode</td>
<td>1</td>
</tr>
<tr>
<td>First MPM</td>
<td>010</td>
</tr>
<tr>
<td>Second MPM</td>
<td>0110</td>
</tr>
<tr>
<td>Third MPM</td>
<td>0111</td>
</tr>
<tr>
<td>Non-MPM and non-neural network mode</td>
<td>00 [5bits]</td>
</tr>
</tbody>
</table>
TABLE VII: PSNR-rate performance gains compared with H.265 of “PNNS substitution” and “PNNS switch” for the first test set.

<table>
<thead>
<tr>
<th>Image name</th>
<th>PSNR-rate performance gain</th>
<th>’’PNNS substitution’’</th>
<th>’’PNNS switch’’</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low rate</td>
<td>High rate</td>
<td>Full rate</td>
</tr>
<tr>
<td>Barbara</td>
<td>2.47%</td>
<td>1.31%</td>
<td>1.79%</td>
</tr>
<tr>
<td>Lena</td>
<td>1.68%</td>
<td>2.11%</td>
<td>2.05%</td>
</tr>
<tr>
<td>Mandrill</td>
<td>0.77%</td>
<td>0.58%</td>
<td>0.67%</td>
</tr>
<tr>
<td>Peppers</td>
<td>1.61%</td>
<td>1.50%</td>
<td>1.71%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Video sequence</th>
<th>’’PNNS substitution’’</th>
<th>’’PNNS switch’’</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low rate</td>
<td>High rate</td>
</tr>
<tr>
<td>BJ Terrace</td>
<td>1.06%</td>
<td>0.95%</td>
</tr>
<tr>
<td>BasketballDrive</td>
<td>4.80%</td>
<td>2.87%</td>
</tr>
<tr>
<td>Cactus</td>
<td>1.48%</td>
<td>1.51%</td>
</tr>
<tr>
<td>ParkScene</td>
<td>0.64%</td>
<td>1.16%</td>
</tr>
<tr>
<td>Kimono</td>
<td>1.28%</td>
<td>1.55%</td>
</tr>
<tr>
<td>BJ Mall</td>
<td>1.20%</td>
<td>1.30%</td>
</tr>
<tr>
<td>BasketballDrill</td>
<td>1.81%</td>
<td>1.34%</td>
</tr>
<tr>
<td>RaceHorsesC</td>
<td>1.34%</td>
<td>1.58%</td>
</tr>
<tr>
<td>PartyScene</td>
<td>1.02%</td>
<td>0.91%</td>
</tr>
<tr>
<td>BJ Square</td>
<td>0.79%</td>
<td>0.86%</td>
</tr>
<tr>
<td>BasketballPass</td>
<td>1.61%</td>
<td>1.80%</td>
</tr>
<tr>
<td>BlowingBubbles</td>
<td>0.66%</td>
<td>1.22%</td>
</tr>
<tr>
<td>RaceHorses</td>
<td>1.32%</td>
<td>1.63%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Video sequence</th>
<th>’’PNNS substitution’’</th>
<th>’’PNNS switch’’</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low rate</td>
<td>High rate</td>
</tr>
<tr>
<td>Bus</td>
<td>1.67%</td>
<td>1.17%</td>
</tr>
<tr>
<td>City</td>
<td>1.55%</td>
<td>1.19%</td>
</tr>
<tr>
<td>Crew</td>
<td>1.56%</td>
<td>1.24%</td>
</tr>
<tr>
<td>Football</td>
<td>1.44%</td>
<td>1.78%</td>
</tr>
<tr>
<td>Harbour</td>
<td>1.80%</td>
<td>0.73%</td>
</tr>
<tr>
<td>Soccer</td>
<td>0.96%</td>
<td>0.95%</td>
</tr>
</tbody>
</table>

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 VI, VII, VIII, and IX). 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, \( \nu_{16} = 0.116 \), \( \nu_{32} = 0.085 \), and \( \nu_{64} = 0.088 \). This means that, compared to the average statistics in Figure 11, the original H.265 intra prediction mode of index 18 is used approximately 10 times more frequently. This explains why the PSNR-rate performance gain provided by “PNNS substitution” is only 0.39% (see Table VIII). 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. Figure 12 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\(^1\).
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 VI, VII, VIII, and IX). Note that, in few cases, the PSNR-rate performance gain in “full rate” 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 $QP \in [27, 32]$.

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 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 X). 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

Section VI-C just showed the effectiveness of the proposed PNNS in a rate-distortion sense. The last issue is that the neural networks of PNNS are trained on contexts without quantization noise but, during the test phase inside H.265, these neural networks are fed with contexts containing H.265 quantization noise. It is thus natural to ask whether, during the test phase inside H.265, the quality of the predictions provided by the neural networks of PNNS is degraded by the fact that no quantization noise exists in the contexts during their training. To answer this, 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, the second set is trained on contexts that are encoded and decoded via H.265 with $QP \sim U\{32, 37, 42\}$ 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 1.1%. 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 XI 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.”
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

ARCHITECTURE OF $g_{m}^{c}$ AND $g_{m}^{t}$ FOR H.265

The architecture of $g_{m}^{c}$, $m \in \{16, 32, 64\}$, is shown in Figure 13. $\text{conv}:p,s|\text{LeakyReLU}$, $p \in \mathbb{N}^*$, $s \in \mathbb{N}^*$, denotes the convolutional layer with spatial stride $s$, $p$ output feature maps and LeakyReLU with slope 0.1 as non-linear activation. Note that the height and width of the $p$ convolutional kernels in $\text{conv}:p,s|\text{LeakyReLU}$ are $2s+1$. For $i \in \{0, 1\}$, $\phi_{m}^{c,i}$ gathers all the weights and biases in the $g_{m}^{c}$ that is applied to $X_i$.

To obtain the architecture of $g_{m}^{t}$, the architecture of $g_{m}^{c}$ is reversed. This means that each sequence of layers in Figure 13 is reversed. Besides, each convolution is replaced by a transpose convolution. For instance, Figure 14 illustrates the result of reversing the architecture of $g_{16}^{c}$, “tconv” is an abbreviation for “transpose convolution”. $\phi_{m}^{t}$ gathers all the weights and biases in $g_{m}^{t}$.

Fig. 13: Illustration of the architecture of (a) $g_{16}^{c}$, (b) $g_{32}^{c}$ and (c) $g_{64}^{c}$. Fig. 12: Comparison of (a) a $100 \times 100$ 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, \{rate = 0.670 bpp, PSNR = 38.569 dB\}. For “PNNS switch”, \{rate = 0.644 bpp, PSNR = 38.513 dB\}. 
Fig. 14: Illustration of the architecture of $g_{16}$.