Dynamic Neural Network for Lossy-to-Lossless Image Coding

Tassnim Dardouri, Mounir Kaaniche, Amel Benazza-Benyahia,
Jean-Christophe Pesquet

Dynamic Neural Network for Lossy-to-Lossless Image Coding

Tassnim Dardouri, *Student Member, IEEE,* Mounir Kaaniche, *Senior Member, IEEE,* Amel Benazza-Benyahia, and Jean-Christophe Pesquet, *Fellow, IEEE,*

***Abstract*—Lifting-based wavelet transform has been extensively used for efficient compression of various types of visual data. Generally, the performance of such coding schemes strongly depends on the lifting operators used, namely the prediction and update filters. Unlike conventional schemes based on linear filters, we propose, in this paper, to learn these operators by exploiting neural networks. More precisely, a classical Fully Connected Neural Network (FCNN) architecture is firstly employed to perform the prediction and update. Then, we propose to improve this FCNN-based Lifting Scheme (LS) in order to better take into account the input image to be encoded. Thus, a novel dynamical FCNN model is developed, making the learning process adaptive to the input image contents for which two adaptive learning techniques are proposed. While the first one resorts to an iterative algorithm where the computation of two kinds of variables is performed in an alternating manner, the second learning method aims to learn the model parameters directly through a reformulation of the loss function. Experimental results carried out on various test images show the benefits of the proposed approaches in the context of lossy and lossless image compression.**

***Index Terms*—Lifting scheme, image compression, adaptive wavelets, optimization, neural networks, adaptive learning.**

I. Introduction

Discrete Wavelet Transforms (DWTs) have been widely used in various signal and image processing tasks such as denoising, super-resolution, and compression [1], [2]. The main interest of these transforms is their ability to provide a multiresolution (or multiscale) representation of the input signal with a good energy compaction. For instance, these transforms have been retained in the JPEG2000 image compression standard [3]. Many research works have been developed showing their benefits in the compression of other stereo, hologram and video data [4], [5], [6]. More precisely, the second generation of wavelets, based on the Lifting Scheme (LS), has attracted much attention thanks to its low computational cost and its ability to guarantee a perfect reconstruction of the original image [7], [8].

Part of this work was supported by the ANR Chair in Artificial Intelligence BRIDGEABLE and by the Institut Universitaire de France.

T. Dardouri and M. Kaaniche are with Université Sorbonne Paris Nord, L2TI, UR 3043, F-93430, Villetaneuse, France. E-mail: tasnim.dardouri@edu.univ-paris13.fr, mounir.kaaniche@univ-paris13.fr.

A. Benazza-Benyahia is with University of Carthage SUP'COM, LR11TIC01, COSIM Lab., 2083, El Ghazala, Tunisia. E-mail: benazza.amel@supcom.rnu.tn.

J.-C. Pesquet is with Centre de Vision Numérique, Université Paris-Saclay, CentraleSupélec, Inria, 91190 Gif-sur-Yvette, France. E-mail: jean-christophe@pesquet.eu.

A conventional lifting structure consists of prediction and update filters that generate the detail and approximation wavelet coefficients, respectively [9], [10]. Generally, the performance of LS-based coding schemes strongly depends on the choice of these filters. For this reason, great attention has been paid to the design of the prediction and update operators in order to build compact wavelet representation that is well adapted to the input data contents [11], [12], [13], [14]. To this end, the prediction filters are often optimized by minimizing the variance of the detail coefficients [15]. In addition to such ℓ_2 minimization approach, the use of sparse criteria, e.g. ℓ_1 and weighted ℓ_1 measures, has also been investigated in [16]. Moreover, in [12], [17], the authors proposed to minimize the detail signal entropy. However, such an optimization problem is solved empirically by using the Nelder-Mead simplex algorithm. Unlike for prediction filters, the optimization of the update filter is less straightforward. Two main approaches have thus been developed for that purpose. The first one consists in minimizing the reconstruction error while assuming that the detail coefficients are set to zero in the synthesis stage [15], [18]. This optimization method leads to a complex linear system of equations. To reduce this complexity, another approach has been developed in [14], [19]. It aims at minimizing the error between the approximation coefficients and the output of an ideal low-pass filter applied to the original image. It is also worth noting that other research efforts have been made to design adaptive directional transforms based on the concept of lifting scheme [20], [21]. However, the main drawback of such transforms is that they require transmitting side information to the decoder, which generally affects the compression performance, in particular at low bitrates.

To further improve the efficiency of conventional image coding schemes, it is worth pointing out that particular attention has been paid recently to Neural Networks (NN). In this context, most of the developed NN-based coding schemes proceed as follows. First, a given NN architecture is selected to convert the input image into a compact representation. Then, a quantization step followed by an entropy encoding process is performed. Finally, a synthesis stage is applied to reconstruct the image. Such schemes are known as end-to-end compression methods [22], [23], [24], [25], [26], [27], [28]. It should be noted here that these methods mainly differ in the employed neural network model or the loss function used to learn the weights of the model. Moreover, intra prediction techniques using neural networks for image and video coding have also been developed [29]. To this end, Convolutional Neural Networks (CNNs) and Fully Connected

Neural Network (FCNN) architectures have been investigated in [30] and [31], respectively. In [32], the authors propose to apply FCNNs to small image blocks and CNNs to large blocks. In addition, other studies have also been devoted to the improvement of DCT (Discrete Cosine Transform) and DWT-based coding schemes [33], [34], [35], [36]. Indeed, in [33], a CNN architecture is used to design a DCT-like transform for image compression. In [34], the authors apply a DWT to the input image, and the resulting wavelet subbands are fed into a CNN to generate new detail coefficients. In [35], the authors propose also to apply a DWT to the input image as a pre-processing step. Then, a convolutional autoencoder is trained end-to-end for a target bitrate set to 0.15 bits per pixel. While this method outperforms JPEG, it is much less performant than the JPEG2000 image compression standard. Recently, a separable lifting structure has been proposed which makes use of CNNs during the prediction stage [36]. However, the update stage simply consists of a mean filter. It should be emphasized that the above methods, as well as most of the existing neural networks-based compression approaches, do not generate integer coefficients and so, they are not suitable for lossless image compression. On the other hand, a few deep learning approaches devoted to lossless compression have been developed [37], [38], [39], [40].

The objective of this paper is to design a LS based on neural network architecture in the context of lossy-to-lossless image compression. The main contributions of this paper are described below:

- While adopting a conventional 2D non-separable lifting scheme composed of three prediction steps followed by an update step, we first propose to learn the prediction filters using an FCNN. The same network is then used to perform the update stage, where the update optimization problem is reformulated as a prediction problem.
- While the previous FCNN-based lifting scheme relies on a fixed network, another major contribution of this work aims to build a *dynamical* FCNN model adapted to the contents of the input image to be encoded. To this end, two adaptive learning strategies taking into account the input training image are proposed. The first one resorts to an *iterative* algorithm that alternates between the learning of the hidden layer weight parameters and the computation of the linear weights of the output layer. The second one aims to *directly* learn the weights of the hidden layers while integrating the analytical expression of the optimal linear weights in the loss function of the FCNN model.
- Unlike the reported deep learning based coding methods which have been devoted to either lossless or lossy compression, the proposed FCNN based LS architectures present the advantage of allowing lossy as well as lossless compression.

For the sake of simplicity, the proposed FCNN-based prediction and update steps have been designed based on the analysis transform stage without taking into account the quantization/coding module as well as the synthesis transform. Compared to the recent CNN-based LS design method in [36], this work presents the following contributions. First, we propose here to perform the update using FCNNs, which results in a fully nonlinear transform unlike [36] where the update is simply replaced by a mean filter. Second, we resort to a non separable LS which leads to a reduction of the number of decomposition stages instead of a separable LS based decomposition that is used in [36]. Third, our FCNN weight parameters are trained and learned for each resolution level, while the weights of the CNN models are kept fixed across the different resolution levels of the LS-based wavelet decomposition in [36]. Finally, our dynamical FCNN model is *adaptive* and takes into account the input image to be encoded, which leads to a better generalization performance. Note also that a preliminary version of our work has been presented in [41]. While the latter focuses on the alternating optimization technique, we introduce here a second optimization approach. A study of both optimization techniques is also conducted. Finally, compared to [41], more extensive experiments have been carried out in the context of lossy as well as lossless compression.

The remainder of this paper is organized as follows. In Section II, we provide the necessary background on lifting schemes while emphasizing optimization issues for the involved operators. The principle of the proposed FCNN-based LS design method is presented in Section III. The adaptive learning techniques of our dynamical FCNN model are then described in Section IV. Finally, experimental results are shown and discussed in Section V, and some conclusions are drawn in Section VI.

II. Background on LS-based image compression

The concept of LS has been widely used in image coding and lies at the core of the JPEG2000 compression standard. A conventional LS incorporates three fundamental steps, namely a polyphase decomposition, a prediction, and an update. The first step aims at building two disjoint subsets composed of even and odd samples of the input 1D signal. Then, the prediction step consists in predicting the samples of one subset (for example the even one) from those of the other subset (i.e. the odd one). The resulting prediction error represents the detail signal. Finally, the odd samples are updated using the computed detail coefficients, yielding the approximation signal. By performing this 1D decomposition along the lines and the columns of an input image, one approximation subband and three detail ones are produced.

However, such separate 1D processing is not the most efficient way to capture the two-dimensional characteristics of image edges that are neither horizontal nor vertical. For this reason, some research efforts have been devoted to the design of 2D Non Separable Lifting Scheme (NSLS). In this respect, an efficient architecture composed of three prediction steps followed by an update has been proposed and widely investigated in the literature [14], [42]. The principle of this architecture will be described in what follows.

A. Principle of the considered lifting structure

Let x and x_j respectively denote the input image and its approximation subband at resolution level j (where $x_0 = x$).

A typical 2D NSLS is shown in Fig. 1.

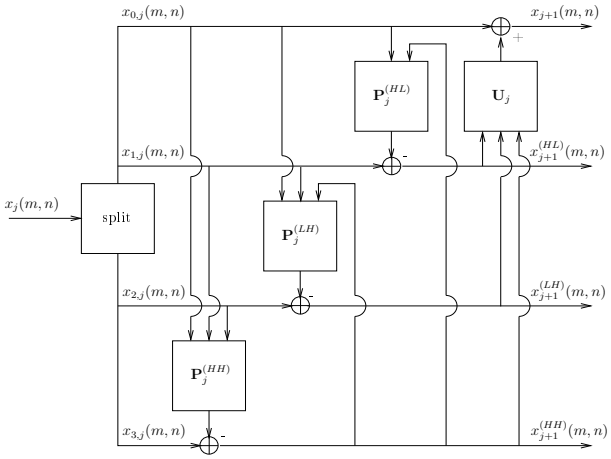


Fig. 1. NSLS decomposition structure.

This decomposition involves different steps. The first one, designated in Fig. 1 by the split box, aims at partitioning the input signal $x_j(m,n)$ into four polyphase components given by

$$\begin{cases} x_{0,j}(m,n) = x_j(2m,2n), \\ x_{1,j}(m,n) = x_j(2m,2n+1), \\ x_{2,j}(m,n) = x_j(2m+1,2n), \\ x_{3,j}(m,n) = x_j(2m+1,2n+1). \end{cases} \quad (1)$$

Then, three prediction filters, denoted in Fig. 1 by $\mathbf{P}_j^{(HH)}$, $\mathbf{P}_j^{(LH)}$ and $\mathbf{P}_j^{(HL)}$, are applied to generate three detail subbands oriented diagonally $x_{j+1}^{(HH)}$, vertically $x_{j+1}^{(LH)}$ and horizontally $x_{j+1}^{(HL)}$, respectively. Finally, an update filter $\mathbf{U}_j$ is used to produce the approximation subband x_{j+1}. Therefore, the resulting wavelet coefficients are obtained as follows:

$$\begin{aligned} {x_{j+1}}^{(HH)}(m,n) = x_{3,j}(m,n) - \Big(& (\mathbf{P}_{0,j}^{(HH)})^\top \mathbf{x}_{0,j}^{(HH)}(m,n) \\ & + (\mathbf{P}_{1,j}^{(HH)})^\top \mathbf{x}_{1,j}^{(HH)}(m,n) + (\mathbf{P}_{2,j}^{(HH)})^\top \mathbf{x}_{2,j}^{(HH)}(m,n) \Big), \end{aligned} \quad (2)$$

$$\begin{aligned} {x_{j+1}}^{(LH)}(m,n) = x_{2,j}(m,n) - \Big(& (\mathbf{P}_{0,j}^{(LH)})^\top \mathbf{x}_{0,j}^{(LH)}(m,n) \\ & + (\mathbf{P}_{1,j}^{(LH)})^\top \mathbf{x}_{1,j}^{(LH)}(m,n) + (\mathbf{P}_{2,j}^{(LH)})^\top \underline{\mathbf{x}}_{j+1}^{(HH)}(m,n) \Big), \end{aligned} \quad (3)$$

$$\begin{aligned} x_{j+1}^{(HL)}(m,n) = & x_{1,j}(m,n) - \Big((\mathbf{P}_{0,j}^{(HL)})^\top \mathbf{x}_{0,j}^{(HL)}(m,n) \\ & + (\mathbf{P}_{1,j}^{(HL)})^\top \overline{\mathbf{x}}_{j+1}^{(HH)}(m,n) \Big), \end{aligned} \quad (4)$$

$$\begin{aligned} x_{j+1}(m,n) = x_{0,j}(m,n) + \Big(& (\mathbf{U}_{0,j}^{(HL)})^\top \mathbf{x}_{j+1}^{(HL)}(m,n) \\ & + (\mathbf{U}_{1,j}^{(LH)})^\top \mathbf{x}_{j+1}^{(LH)}(m,n) + (\mathbf{U}_{2,j}^{(HH)})^\top \mathbf{x}_{j+1}^{(HH)}(m,n) \Big), \end{aligned} \quad (5)$$

where for every $i \in \{0,1,2\}$ and $o \in \{HL, LH, HH\}$,

- $\mathbf{P}_{i,j}^{(o)} = (p_{i,j}^{(o)}(s,t))_{(s,t)\in\mathcal{P}_{i,j}^{(o)}}$ and $\mathbf{U}_{i,j}^{(o)} = (u_{i,j}^{(o)}(s,t))_{(s,t)\in\mathcal{U}_{i,j}^{(o)}}$ are the prediction and update weight vectors whose supports are respectively denoted by $\mathcal{P}_{i,j}^{(o)}$ and $\mathcal{U}_{i,j}^{(o)}$.
- $\mathbf{x}_{i,j}^{(o)}(m,n) = (x_{i,j}(m+s,n+t))_{(s,t)\in\mathcal{P}_{i,j}^{(o)}}$ is a reference vector used to generate $x_{j+1}^{(o)}(m,n)$.
- $\underline{\mathbf{x}}_{j+1}^{(HH)}(m,n) = (x_{j+1}^{(HH)}(m+s,n+t))_{(s,t)\in\mathcal{P}_{2,j}^{(LH)}}$ and $\overline{\mathbf{x}}_{j+1}^{(HH)}(m,n) = (x_{j+1}^{(HH)}(m+s,n+t))_{(s,t)\in\mathcal{P}_{1,j}^{(HL)}}$ are two reference vectors used to compute $x_{j+1}^{(LH)}(m,n)$ and $x_{j+1}^{(HL)}(m,n)$, respectively.
- $\mathbf{x}_{j+1}^{(o)}(m,n) = (x_{j+1}^{(o)}(m+s,n+t))_{(s,t)\in\mathcal{U}_{i,j}^{(o)}}$ is the reference vector containing the set of detail samples used in the update step.

Note that a multiresolution representation of the input image is finally obtained by recursively repeating this decomposition to the resulting approximation subband x_{j+1}. For image reconstruction, the synthesis stage of such NSLS can be easily deduced by reverting the order of lifting operations.

Thus, it can be noticed that the key step in the design of LS-based decomposition is the choice of the predictor and update filters. While the JPEG2000 compression standard employs filters with fixed weights [3], it appears interesting to make these weights adaptable to the image contents and to build more efficient optimized lifting schemes for image coding. While the state-of-the-art prediction and update optimization methods have been summarized in Section I, let us now describe the most commonly used approaches.

B. Optimization of the prediction and update filters

Since the detail signal can be seen as a prediction error, the prediction filters $\mathbf{P}_j^{(o)}$ are often optimized by minimizing the variance of the detail coefficients $x_{j+1}^{(o)}$:

$$\mathcal{J}(\mathbf{P}_j^{(o)}) = \sum_{m=1}^{M_j} \sum_{n=1}^{N_j} \Big(x_{i,j}(m,n) - (\mathbf{P}_j^{(o)})^\top \tilde{\mathbf{x}}_j^{(o)}(m,n) \Big)^2 \quad (6)$$

where $M_j \times N_j$ represents the number of samples $x_{i,j}(m,n)$ to be predicted, $\mathbf{P}_j^{(o)}$ is the prediction filter to be optimized, and $\tilde{\mathbf{x}}_j^{(o)}$ is the reference vector gathering the samples used to generate the detail coefficients $x_{j+1}^{(o)}$. For example, for the diagonal detail subband $x_{j+1}^{(HH)}$, according to Eq. (2), these prediction and reference vectors read

$$\mathbf{P}_j^{(HH)} = (\mathbf{P}_{0,j}^{(HH)}, \mathbf{P}_{1,j}^{(HH)}, \mathbf{P}_{2,j}^{(HH)})^\top \quad (7)$$

$$\tilde{\mathbf{x}}_j^{(HH)}(m,n) = (\mathbf{x}_{0,j}^{(HH)}(m,n), \mathbf{x}_{1,j}^{(HH)}(m,n), \mathbf{x}_{2,j}^{(HH)}(m,n))^\top \quad (8)$$

By minimizing the criterion $\mathcal{J}$, given by Eq. (6), it can be noticed that the optimal predictor $\mathbf{P}_j^{(o)}$ can be found by solving the well-known Yule-Walker equations:

$$\mathsf{E}[\tilde{\mathbf{x}}_j^{(o)}(m,n)\tilde{\mathbf{x}}_j^{(o)}(m,n)^\top]\mathbf{P}_j^{(o)} = \mathsf{E}[x_{i,j}(m,n)\tilde{\mathbf{x}}_j^{(o)}(m,n)], \quad (9)$$

where $\mathsf{E}[\cdot]$ denotes the mathematical expectation which is empirically estimated by an average operation.

Once the diagonal, horizontal and vertical prediction filters are obtained, the update filter remains to be set. However, its optimization is more challenging and few works have been developed for that purpose. As mentioned in Section I, a simple and efficient method has been proposed in [14], [19], which aims at minimizing the quadratic error between the approximation signal and the decimated version of the output of an ideal low-pass filter. This error is expressed as follows:

$$\begin{aligned}\tilde{\mathcal{J}}(\mathbf{U}_j) &= \sum_{m=1}^{M_j}\sum_{n=1}^{N_j}\Big(x_{j+1}(m,n) - y_{j+1}(m,n)\Big)^2 \\ &= \sum_{m=1}^{M_j}\sum_{n=1}^{N_j}\Big(x_{0,j}(m,n) + \mathbf{U}_j^\top \tilde{\mathbf{x}}_{j+1}(m,n) \\ &\qquad - y_{j+1}(m,n)\Big)^2, \end{aligned} \tag{10}$$

where
• $\mathbf{U}_j = (\mathbf{U}_{0,j}^{(HL)}, \mathbf{U}_{1,j}^{(LH)}, \mathbf{U}_{2,j}^{(HH)})^\top$ is the update vector to be optimized,
• $\tilde{\mathbf{x}}_{j+1}(m,n) = (\mathbf{x}_{j+1}^{(HL)}(m,n), \mathbf{x}_{j+1}^{(LH)}(m,n), \mathbf{x}_{2,j}^{(HH)}(m,n))^\top$ is the update reference vector composed of the detail signals,
• $y_{j+1}(m,n) = (h * x_j)(2m, 2n)$, and
• $h(m,n) = \frac{1}{4}\,\mathrm{sinc}(\frac{m\pi}{2})\,\mathrm{sinc}(\frac{n\pi}{2})$ is the impulse response of the 2D ideal rectangular low-pass filter.
By minimizing the criterion $\tilde{\mathcal{J}}$, given by Eq. (10), it can be shown that the optimal update filter $\mathbf{U}_j$ satisfies the following equation:

$$\begin{aligned}&\mathsf{E}[\tilde{\mathbf{x}}_{j+1}(m,n)\tilde{\mathbf{x}}_{j+1}(m,n)^\top]\mathbf{U}_j \\ &\quad = \mathsf{E}[y_{j+1}(m,n)\tilde{\mathbf{x}}_{j+1}(m,n)] - \mathsf{E}[x_{0,j}(m,n)\tilde{\mathbf{x}}_{j+1}(m,n)].\end{aligned} \tag{11}$$

Therefore, Equations (9) and (11) show that the optimal prediction and update filters are the solutions of a linear system of equations.
In section III, we will present a novel approach for the optimization of the prediction and update.

III. Proposed FCNN-based LS

A. Motivation

To go beyond the conventional linear models and achieve a more accurate nonlinear approximation, we now propose to perform the prediction and update steps by using a neural network, and more specifically an FCNN model. The choice of this NN architecture is motivated by the following reasons. First, the FCNN is a simple and efficient NN for both intra- and inter-prediction tasks [31], [43]. Moreover, it does not require handling a large amount of data during the training phase. Indeed, in our context, the training samples correspond to a set of vectors containing the neighboring pixels used to predict the different image pixels. Thus, even with a relatively small size training set (for example including a few hundred images), by performing the prediction for each pixel of the image, the number of training samples becomes large enough for learning.
Therefore, in the chosen NSLS analysis structure, the three prediction lifting operators followed by the update one will be replaced by FCNN blocks denoted by $f_j^{(o)}$ with $o \in \{HH, LH, HL, LL\}$. The new FCNN-based NSLS analysis structure is depicted in Fig. 2.

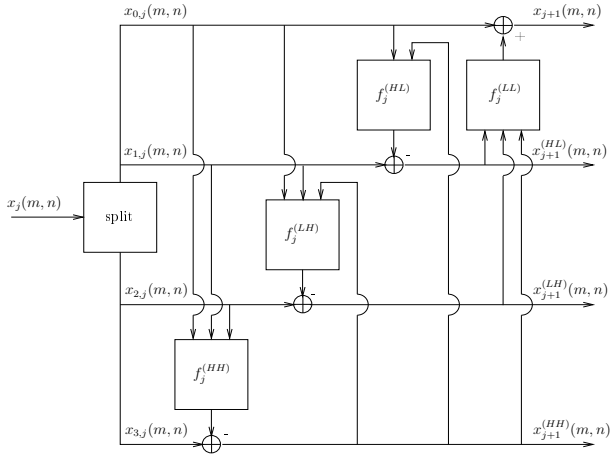


Fig. 2. FCNN-based NSLS analysis structure.

B. FCNN-based prediction stage

As the detail signal $x_{j+1}^{(o)}$ corresponds to a prediction error, the first three prediction steps in the proposed FCNN-based LS can be rewritten as follows:

$$\begin{aligned}&\forall\, o \in \{HH, LH, HL\}, \\ &\qquad x_{j+1}^{(o)}(m,n) = x_{i,j}(m,n) - \hat{x}_{i,j}(m,n) \\ &\qquad\qquad = x_{i,j}(m,n) - f_j^{(o)}(\tilde{\mathbf{x}}_j^{(o)}(m,n))\end{aligned} \tag{12}$$

where $x_{i,j}(m,n)$ (with $i \in \{1,2,3\}$) is the polyphase component to be predicted and $\hat{x}_{i,j}(m,n) = f_j^{(o)}(\tilde{\mathbf{x}}_j^{(o)}(m,n))$ is the predicted value. The latter corresponds to the output of a standard FCNN architecture applied to the input vector $\tilde{\mathbf{x}}_j^{(o)}(m,n)$ which represents the reference vector used in the prediction step for producing the associated detail signal $x_{j+1}^{(o)}(m,n)$.
More precisely, as illustrated by Fig. 3, in order to find the predicted value $\hat{x}_{i,j}(m,n)$ from the input reference vector $\tilde{\mathbf{x}}_j^{(o)}(m,n)$, H hidden layers are firstly used. The output values of their neurons (or units) are computed by applying a linear combination (with bias) followed by a nonlinear activation function. The different parameters involved in the computation of all the neuron values in these layers are denoted by $\mathbf{\Theta}_j^{(o)}$. Finally, an output layer with a single neuron is performed yielding the computation of $\hat{x}_{i,j}(m,n)$ from the unit values of the last hidden layer based on a linear combination. The parameters associated with this last operation will be designated by $\mathbf{w}_j^{(o)}$.

The set of weights $\left(\mathbf{\Theta}_j^{(o)}, \mathbf{w}_j^{(o)}\right)$ is learned by performing the forward and backward propagation passes, while minimizing a loss function. To this end, each FCNN-based prediction

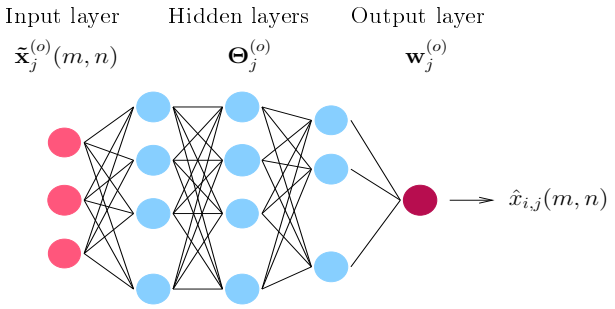


Fig. 3. FCNN-based prediction stage.

stage is trained by minimizing the ℓ_2-norm of the prediction error (i.e. the Mean Square Error (MSE)) given by

$$\mathcal{L}(x_{i,j},\hat{x}_{i,j}) = \frac{1}{M_j \times N_j}\sum_{m=1}^{M_j}\sum_{n=1}^{N_j}\big(x_{i,j}(m,n)-\hat{x}_{i,j}(m,n)\big)^2. \tag{13}$$

To optimize the above loss function, a Mini-Batch Gradient Descent (MBGD) algorithm has been used [44]. Thus, by computing the gradient of the loss function at each iteration (i.e. for each mini-batch) and updating the model parameters from one mini-batch to another within the whole training dataset, and by repeating this process many times (i.e. many epochs) until convergence of the algorithm, we will obtain the optimal weights $\left(\mathbf{\Theta}_j^{(o)},\mathbf{w}_j^{(o)}\right)$ that allow us to compute $\hat{x}_{i,j}(m,n)$. Then, we deduce the detail wavelet coefficients $x_{j+1}^{(o)}(m,n)$ by using Eq. (12). For lossless compression purposes, it is worth pointing out that, in Eq. (12), a rounding operator is applied to the predicted value $\hat{x}_{i,j}(m,n)$ (i.e. to the output of the FCNN model $f_j^{(o)}(\tilde{\mathbf{x}}_j^{(o)}(m,n))$ in order to generate integer wavelet coefficients of the input test images.

Note also that in our FCNN-based LS design method, the learning process is separately performed for each prediction/update stage of the wavelet decomposition. For instance, at each resolution level j, three separate FCNN models $f_j^{(HH)}$, $f_j^{(LH)}$ and $f_j^{(HL)}$ are employed to generate the three detail subbands oriented diagonally, vertically and horizontally, respectively. The latter are then used to generate the approximation subband based on the FCNN model $f_j^{(LL)}$.

Example: FCNN-based diagonal prediction stage

To better illustrate the use of the previous described FCNN-based prediction stage, we will consider the example of the first prediction stage dedicated to the generation of the diagonal detail coefficients $x_{j+1}^{(HH)}(m,n)$. As seen in Fig. 2, this stage aims at predicting the samples $x_{3,j}(m,n)$ from their neighboring samples $x_{0,j}(m,n)$, $x_{1,j}(m,n)$ and $x_{2,j}(m,n)$. The selected neighboring samples will form the reference vector $\tilde{\mathbf{x}}_j^{(HH)}(m,n)$ associated with the input layer of the FCNN model (see Fig. 3). After training the FCNN model using the MBGD algorithm, the optimal weights $\left(\mathbf{\Theta}_j^{(HH)},\mathbf{w}_j^{(HH)}\right)$ are obtained. These weights are then applied to the test images to compute the predicted samples $\hat{x}_{3,j}(m,n)$ given the reference vectors $\tilde{\mathbf{x}}_j^{(HH)}(m,n)$. Finally, the difference between the original pixels $x_{3,j}(m,n)$ and the predicted ones $\hat{x}_{3,j}(m,n)$ will correspond to the diagonal detail coefficients $x_{j+1}^{(HH)}(m,n)$.

C. FCNN-based update stage

Unlike the generation of the detail wavelet coefficients that is easily achieved by using the previous FCNN approach for prediction, the computation of the approximation coefficients requires more care. For this purpose, we propose to focus on the optimization approach described in Section II-B. Indeed, inspired from (10), we define the following error signal:

$$\begin{aligned} e_j(m,n) &= y_{j+1}(m,n) - x_{j+1}(m,n) \\ &= y_{j+1}(m,n) - x_{0,j}(m,n) - \mathbf{U}_j^\top \tilde{\mathbf{x}}_{j+1}(m,n). \end{aligned} \tag{14}$$

In the above equation, we propose to replace the linear term depending on the update operator by a nonlinear one which corresponds to the output of an FCNN. Thus, Eq. (14) can be rewritten as

$$\begin{aligned} \tilde{e}_j(m,n) &= t_j(m,n) - \hat{t}_j(m,n) \\ &= t_j(m,n) - f_j^{(LL)}(\tilde{\mathbf{x}}_{j+1}(m,n)) \end{aligned} \tag{15}$$

where $t_j(m,n) = y_{j+1}(m,n) - x_{0,j}(m,n)$.

Therefore, according to Eq. (15), the update optimization problem is reformulated as a prediction problem that consists in optimally predicting the target input signal $t_j(m,n)$ from the reference signal $\tilde{\mathbf{x}}_{j+1}(m,n)$. Similarly to the previous FCNN-based prediction optimization tasks, the FCNN-based update aims at finding the predicted value $\hat{t}_j(m,n) = f_j^{(LL)}(\tilde{\mathbf{x}}_{j+1}(m,n))$ from the input reference vector $\tilde{\mathbf{x}}_{j+1}(m,n)$. The associated FCNN weight parameters $\left(\mathbf{\Theta}_j^{(LL)},\mathbf{w}_j^{(LL)}\right)$ will be optimized by minimizing a loss function similar to the previous one, namely $\mathcal{L}(t_j,\hat{t}_j)$. Once the network is trained, the resulting optimal weights are used to compute its output $\hat{t}_j(m,n)$ from the input vector $\tilde{\mathbf{x}}_{j+1}(m,n)$. The approximation coefficients resulting from the FCNN-based update stage are deduced as

$$x_{j+1}(m,n) = x_{0,j}(m,n) + \hat{t}_j(m,n). \tag{16}$$

Again, to generate integer approximation coefficients, for lossless compression, a rounding operator is also applied to $\hat{t}_j(m,n)$ in Eq. (16) during the test phase.

To illustrate the proposed FCNN-based transform, Fig. 4(b) shows the obtained wavelet subbands for a given image at the first resolution level. Note that the FCNN-based prediction and update steps are performed by using the spatial supports shown in Fig. 7. Thus, similar to the wavelet subbands obtained with the conventional LS (see Fig. 4(a)), the FCNN-based decomposition results in subbands representing the approximation coefficients as well as the detail ones oriented horizontally, vertically and diagonally.

Once the FCNN models of the different prediction operations as well as the update are obtained, the wavelet coefficients are generated and then encoded using an entropy coder. For image reconstruction purposes, and similar to classical LS, the FCNN-based LS synthesis stage can be easily obtained by reverting the order of the lifting operations, inverting the signs

(a) (b)

Fig. 4. Image subband representations using: (a) conventional LS (9/7 wavelet transform), (b) FCNN-based NSLS.

(+ and −), and replacing the splitting step by a merging step. Note that this synthesis stage is performed using the same FCNN models as those obtained at the analysis stage. As in the case of conventional LS, the proposed FCNN-based LS guarantees the perfect reconstruction property when the integer version of the wavelet coefficients is employed while including a rounding operator applied to the output of the FCNN.

IV. Dynamical FCNN-based LS construction

A. Motivation

In the previous prediction and update design methods, the FCNN model is trained by minimizing a loss function. Then, the learned weights are kept fixed after training, and used to generate the wavelet representation of any image in the test set. However, in practice, the test dataset images may have some characteristics or some specific contents that are different from those of the training set. In this case, the performance of the previous *static* FCNN-based models may be sub-optimal. Therefore, it may appear interesting to design a *dynamical* or *adaptive* FCNN model where some of the parameters depend on the input image, while maintaining an acceptable complexity.

To achieve this goal, let us recall the main operations performed inside the classical FCNN model. The architecture of the latter consists of two main blocks. The first one, parameterized by $\mathbf{\Theta}_j^{(o)}$, allows to compute the values of the inputs of the neurons in the hidden layers. The second block, parameterized by $\mathbf{w}_j^{(o)}$, produces the output of the FCNN (i.e. predicted value) based on a linear combination of the neuron output values at the last hidden layer. In order to develop a dynamical FCNN model, we propose adjusting the computation of the weight parameters in the second block by taking into account the input image of the network. As a result, the weights associated to the last layer of our new model will depend on the input image $x_{j,k}$, and so, they will be denoted by $\mathbf{w}_{j,k}^{(o)}$ where k indicates the index of the image in the dataset and j represents the resolution level. Thus, the new dynamical FCNN-based model can be summarized by the block diagram depicted in Fig. 5.

To make the computation of the output layer weights adaptive to the input image, different strategies will now be explored.

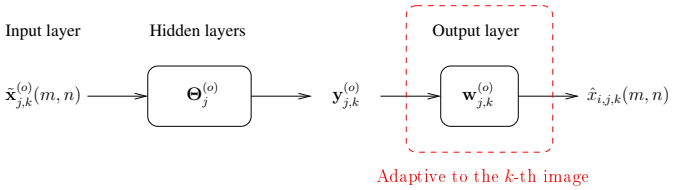


Fig. 5. Proposed dynamical FCNN-based model.

B. Transfer learning approach

A straightforward strategy consists of a transfer learning approach based on the use of a pre-trained model. More precisely, by assuming that the traditional FCNN-based model has been learned on the training dataset (as explained in Section III), let us explain how to handle each input image $x_{j,k}$ in the test set.

First, the vector $\mathbf{\Theta}_j^{(o)}$ is extracted from the learned parameters of the pre-trained FCNN model. The extracted weights are then applied to each image $x_{j,k}$ to compute the vector $\mathbf{y}_{j,k}^{(o)}$ associated to the last hidden layer. Finally, a fine-tuning is performed to deduce the optimal weights of the output layer $\mathbf{w}_{j,k}^{(o)}$ by minimizing the following criterion:

$$\begin{aligned}\mathcal{J}(\mathbf{w}_{j,k}^{(o)}) &= \sum_{m=1}^{M_{j,k}}\sum_{n=1}^{N_{j,k}}\Big(x_{i,j,k}(m,n) - \hat{x}_{i,j,k}(m,n)\Big)^2 \\ &= \sum_{m=1}^{M_{j,k}}\sum_{n=1}^{N_{j,k}}\Big(x_{i,j,k}(m,n) - (\mathbf{w}_{j,k}^{(o)})^\top \mathbf{y}_{j,k}^{(o)}(m,n)\Big)^2\end{aligned} \tag{17}$$

where $M_{j,k}$ and $N_{j,k}$ are the width and the height of the image $x_{j,k}$, divided by 2.

As shown in Eq. (9), the optimal weights of each image $x_{j,k}$ are given by

$$\begin{aligned}\mathbf{w}_{j,k}^{(o)} =& \Big(\mathsf{E}[\mathbf{y}_{j,k}^{(o)}(m,n)\mathbf{y}_{j,k}^{(o)}(m,n)^\top]\Big)^{-1} \\ &\times \mathsf{E}[x_{i,j,k}(m,n)\mathbf{y}_{j,k}^{(o)}(m,n)].\end{aligned} \tag{18}$$

Once the optimal weights $\mathbf{w}_{j,k}^{(o)}$ are obtained, the predicted values $\hat{x}_{i,j,k}(m,n)$ are computed as

$$\hat{x}_{i,j,k}(m,n) = (\mathbf{w}_{j,k}^{(o)})^\top \mathbf{y}_{j,k}^{(o)}(m,n). \tag{19}$$

and the final detail coefficients are deduced from (12).

Note that, in a similar way, we compute $\hat{t}_{j,k}(m,n)$ for each image and deduce the corresponding approximation subband from (16).

C. Adaptive learning techniques

While the main advantage of the previous approach lies in its simplicity, a more sophisticated one consists in implementing an adaptive learning process by taking into account the input image of the training set. Thus, the traditional training process for an FCNN model will be modified to take into account that the weight parameters of the output layer $\mathbf{w}_{j,k}^{(o)}$ depend on the input image. Therefore, the objective of the new training process is to find the

vector of parameters $\mathbf{\Theta}_j^{(o)}$ while adapting $\mathbf{w}_{j,k}^{(o)}$ to each input image $x_{j,k}$. In the following, we propose two techniques for achieving this objective: an alternating approach and a direct one.

Alternating optimization technique

In this technique, because of the dependence between the parameters $\mathbf{\Theta}_j^{(o)}$ and $\mathbf{w}_{j,k}^{(o)}$, we propose an iterative algorithm that alternates between the computation of $\mathbf{w}_{j,k}^{(o)}$ and the update of $\mathbf{\Theta}_j^{(o)}$. More specifically, the learning process will be achieved as follows. For each image $x_{j,k}$, two main steps are performed in an iterative way. Let us start from a given initialization of $\mathbf{\Theta}_j^{(o)}$. At each iteration ℓ, we will produce in the first step the output of the last hidden layer $\mathbf{y}_{j,k}^{(o)}$. Then, the weight parameters of the output layer $\mathbf{w}_{j,k}^{(o,\ell)}$ will be computed by minimizing the ℓ_2-norm of the prediction error (see Eq. (18)). After that, in the second step, we update the weights $\mathbf{\Theta}_j^{(o,\ell)}$ while setting the weight values for the output layer to those obtained at the first step. This leads to

$$\mathbf{\Theta}_j^{(o,\ell)} = \underset{\mathbf{\Theta}_j^{(o)}}{\operatorname{argmin}} \sum_{k,m,n} \left(x_{i,j,k}(m,n) - (\mathbf{w}_{j,k}^{(o,\ell)})^\top \mathbf{y}_{j,k}^{(o,\ell)}(m,n;\mathbf{\Theta}_j^{(o)}) \right)^2 \tag{20}$$

The different steps of this alternating optimization technique are summarized in Algorithm 1.

Algorithm 1: *Alternating optimization*

At each resolution level j and subband o, starting from a given initialization of $\mathbf{\Theta}_j^{(o)}$, do

① Set $\mathbf{\Theta}_j^{(o,0)} = \mathbf{\Theta}_j^{(o)}$

② for $\ell = 1, 2, \ldots, \ell_{\max}$

(a) For every image k in the training set, compute $\mathbf{y}_{j,k}^{(o,\ell)}$ using the weights $\mathbf{\Theta}_j^{(o,\ell-1)}$ and deduce the optimal weights of the output layer $\mathbf{w}_{j,k}^{(o,\ell)}$ using Eq. (18).

(b) Update the parameters $\mathbf{\Theta}_j^{(o,\ell)}$ according to (20).

③ Set $\mathbf{\Theta}_j^{(o)} = \mathbf{\Theta}_j^{(o,\ell_{\max})}$

In practice, a mini-batch approach is implemented where the batch size (i.e. number of training samples used to compute the gradient and update the parameters of the model) corresponds to the number of pixels to be predicted for each input image $x_{j,k}$.

Direct optimization technique

A potential drawback of the previous adaptive learning strategy is computational complexity since it alternates between the computation of $\mathbf{w}_{j,k}^{(o)}$ and the update of $\mathbf{\Theta}_j^{(o)}$. In order to alleviate this drawback and avoid the iterative process applied to each image of the training dataset, we resort to a one-pass optimization technique that allows a *direct* update of the weights $\mathbf{\Theta}_j^{(o)}$. This technique aims at finding the vector of parameters $\mathbf{\Theta}_j^{(o)}$ without explicitly computing the weights $\mathbf{w}_{j,k}^{(o)}$ for each image, as performed in Step ②(a) of the alternating optimization algorithm. For this purpose, we propose to re-write the loss function of our dynamical FCNN model to make it only depending on the vector $\mathbf{\Theta}_j^{(o)}$. More precisely, the cost function used in the standard FCNN model can be expressed as follows:

$$\tilde{\mathcal{L}}(\mathbf{\Theta}_j^{(o)}) = \sum_k \sum_{m=1}^{M_{j,k}} \sum_{n=1}^{N_{j,k}} \Big(x_{i,j,k}(m,n) - \left(\mathbf{w}_{j,k}^{(o)}(\mathbf{\Theta}_j^{(o)}) \right)^\top \mathbf{y}_{j,k}^{(o)}(m,n;\mathbf{\Theta}_j^{(o)}) \Big)^2. \tag{21}$$

Since our main idea is to make the weights $\mathbf{w}_{j,k}^{(o)}$ adaptive to the input image $x_{j,k}$, we propose to integrate in the above cost function the analytical expression of the optimal weights given by Eq. (18). To this end, and in order to show the dependence of the latter weights with respect to the parameter to be optimized $\mathbf{\Theta}_j^{(o)}$, let us first rewrite the solution as

$$\mathbf{w}_{j,k}^{(o)}(\mathbf{\Theta}_j^{(o)}) = \left(\mathbf{\Gamma}_{j,k}(\mathbf{\Theta}_j^{(o)}) \right)^{-1} \mathbf{c}_{j,k}(\mathbf{\Theta}_j^{(o)}) \tag{22}$$

where

$$\mathbf{\Gamma}_{j,k}(\mathbf{\Theta}_j^{(o)}) = \sum_{m=1}^{M_{j,k}} \sum_{n=1}^{N_{j,k}} \mathbf{y}_{j,k}^{(o)}(m,n;\mathbf{\Theta}_j^{(o)}) \mathbf{y}_{j,k}^{(o)}(m,n;\mathbf{\Theta}_j^{(o)})^\top \tag{23}$$

$$\mathbf{c}_{j,k}(\mathbf{\Theta}_j^{(o)}) = \sum_{m=1}^{M_{j,k}} \sum_{n=1}^{N_{j,k}} x_{i,j,k}(m,n) \mathbf{y}_{j,k}^{(o)}(m,n;\mathbf{\Theta}_j^{(o)}). \tag{24}$$

By inserting (22) in (21), and after some calculations, the final expression of the cost function is given by

$$\tilde{\mathcal{L}}(\mathbf{\Theta}_j^{(o)}) = \sum_k \sum_{m=1}^{M_{j,k}} \sum_{n=1}^{N_{j,k}} \Big(x_{i,j,k}(m,n) \Big)^2 - \left(\mathbf{c}_{j,k}(\mathbf{\Theta}_j^{(o)}) \right)^\top \left(\mathbf{\Gamma}_{j,k}(\mathbf{\Theta}_j^{(o)}) \right)^{-1} \mathbf{c}_{j,k}(\mathbf{\Theta}_j^{(o)}). \tag{25}$$

As the optimization algorithm requires the computation of the gradient of the loss function with respect to the weights $\mathbf{\Theta}_j^{(o)}$, the last criterion can be again simplified as follows:

$$\tilde{\mathcal{L}}(\mathbf{\Theta}_j^{(o)}) = -\sum_k \left(\mathbf{c}_{j,k}(\mathbf{\Theta}_j^{(o)}) \right)^\top \left(\mathbf{\Gamma}_{j,k}(\mathbf{\Theta}_j^{(o)}) \right)^{-1} \mathbf{c}_{j,k}(\mathbf{\Theta}_j^{(o)}). \tag{26}$$

Therefore, one can see that the resulting loss function depends explicitly on the vector of parameters $\mathbf{\Theta}_j^{(o)}$.

Once the cost function is defined, the weights $\mathbf{\Theta}_j^{(o)}$ are optimized by applying a mini-batch gradient descent algorithm on the training data. It should be noted here that, for the classical as well as the dynamical FCNN-based prediction and update stages, we have used the entire image $x_{j,k}$ as an input of the neural network.

Test phase: Validation of the adaptive FCN models on the test images

For a given subband o, once the adaptive training process is achieved and the optimal learned parameters $\mathbf{\Theta}_j^{(o)}$ are obtained, the dynamical FCNN models is applied to the test images by following the test phase procedure described in the transfer learning approach. For instance, for the different pixels to be predicted $x_{i,j,k}(m,n)$ of each test image, the

vector of parameters $\mathbf{\Theta}_j^{(o)}$ is first used to generate $\mathbf{y}_{j,k}^{(o)}(m,n)$ (i.e. the neuron output values of the last hidden layer) from the input reference vectors $\tilde{\mathbf{x}}_{j,k}^{(o)}(m,n)$. Then, the optimal weights $\mathbf{w}_{j,k}^{(o)}$ of the output layer, adaptive to the input image, are deduced based on Eq. (18). Finally, the predicted values $\hat{x}_{i,j,k}(m,n)$ are computed using Eq. (19). The difference between the original pixels $x_{i,j,k}(m,n)$ and the predicted ones $\hat{x}_{i,j,k}(m,n)$ constitutes the wavelet coefficient set in subband o.

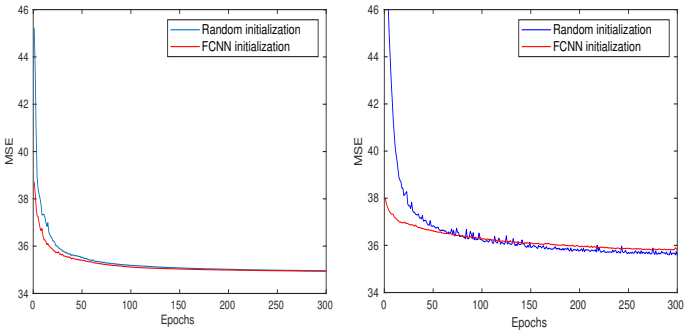
D. Discussion

To conclude this section, we study the behavior of the two proposed optimization techniques to build the dynamical FCNN model. As stated in subsection IV-C, one advantage of the direct optimization technique compared to the alternating one is that it leads to a lower computational complexity. Indeed, the training phase with the direct optimization technique is three times faster than that with the alternating optimization technique. For example, for the CLIC dataset described at the beginning of Section V, by using the spatial supports of the prediction filters shown in Fig. 7, the training phase of the first dynamical FCNN model, conducted on an NVIDIA Tesla V100 32 Gb GPU, takes about 14 hours (resp. 45 hours) with the direct (resp. alternating) optimization technique. Note that, for the remaining subbands as well as at the next resolution levels, this training time is significantly reduced. Fig. 6 illustrates the variations of the loss function with respect to the number of epochs for both optimization techniques using random and FCNN initializations. Note here that the FCNN initialization step corresponds to the case where the weight parameters $\mathbf{\Theta}_j^{(o)}$, used in the first iteration of the MBGD algorithm, are set to those obtained with the classical FCNN model. For the alternating optimization technique, the number of iterations used in each epoch was set to 1 (i.e. $\ell_{\max} = 1$). From these plots, two main observations can be made. First, for each optimization technique, it can be noticed that the final MSE values obtained with both FCNN and random initializations are close. This shows the robustness of our algorithms with respect to the initialization step. Secondly, it can be observed that both alternating and direct optimization techniques converge to close MSE values. This observation suggests that both optimization algorithms are expected to yield similar compression performance. Note that these curves are obtained with the validation images of the standard CLIC dataset that will be described in Section V.

E. Transmission cost of the adaptive weights

While the proposed dynamical FCNN model has the advantage to take into account the input image, it should be noted that the adaptive weights $\mathbf{w}_{j,k}^{(o)}$ obtained at each resolution level j and orientation o of each image need to be sent to the decoder to perform the inverse transform. This will result in a transmission overhead $\mathcal{O}_{\mathbf{w}}$ (in bpp) given by

$$\mathcal{O}_{\mathbf{w}} = \frac{32 \times n^{(H)} \times (4J)}{M \times N} \tag{27}$$



(a): Alternating optimization (b): Direct optimization

Fig. 6. Evolution of the loss function with respect to the number of epochs for alternating and direct optimization techniques based on FCNN and random initializations.

where $n^{(H)}$ represents the number of neurons in the last hidden layer, J is the number of resolution levels and, M and N are respectively the width and the height of the input image. For example, for an image of size 600×600 decomposed over $J = 3$ resolution levels, and using an FCNN model with $H = 4$ hidden layers of dimensions $128 \times 64 \times 32 \times 16$, the transmission cost of the weight coefficients is equal to 0.017 bpp. In our simulations, we have used a half-precision format for the weights of the last layer, which further reduces the transmission cost to 0.0085 bpp. We have checked that using half-precision format of these adaptive weights has no detrimental impact on the final compression results.

V. Experimental Results

In this section, we evaluate the proposed FCNN-based LS and compare it to state-of-the-art methods. In this respect, after describing the experimental settings, we will compare various lifting scheme-based wavelet representations in terms of compression performance. Then, the merits of our approach in the context of lossless as well as lossy compression will be illustrated.

A. Experimental settings

Our simulations are performed on various image datasets. More precisely, the CLIC image dataset[1] has been chosen for training. The latter, built in the context of a workshop related to the Challenge on Learned Image Compression (CLIC 2018), is composed of 585 images with different sizes. For the test phase, three image datasets have been used. The first one consists of 40 crop images, of size 512×512, taken from different images of the test CLIC dataset. The second and third ones are taken from the Tecnick sampling dataset[2] [45], [46]. Indeed, the second dataset is composed of 30 images of size 600×600 whereas the third one contains 35 larger images with size 1200×1200. Note that these two datasets will be denoted by Tecnick-R1 and Tecnick-R2, respectively. It is worth pointing out that our proposed methods have been designed for grayscale images. Thus, the color CLIC and

[1] http://www.compression.cc/2018/challenge/
[2] https://testimages.org/

Tecnick-R2 images are converted to grayscale before applying our methods as well as its competitors. Note that coding the luminance component is generally more challenging than the chrominance ones.

Moreover, the static as well as the dynamic FCNN architectures are implemented using $H = 4$ hidden layers with $128 \times 64 \times 32 \times 16$ neurons. In the training, the loss functions are evaluated without considering the rounding operator (as written in Eqs. (13), (20) and (21)). The Parametric Rectified Linear Unit (PReLU) has been employed as an activation function. ADAM algorithm is used with a learning rate equal to 10^{-3} while applying a decay of 10^{-4}, exponential decay rates for the first and second moment estimates which are respectively equal to 0.9 and 0.999, and $\epsilon = 10^{-7}$ to prevent any division by zero. These implementations were carried out by using Keras with TensorFlow backend on an NVIDIA Tesla V100 32 Gb GPU. Finally, regarding the dynamical FCNN architecture, and after showing the impact of the initialization in Fig. 6, the alternating and direct optimization techniques have employed the pre-trained FCNN model in the initialization step. Note that the source code of our methods is available on github[3].

B. Comparison methods

Let us recall that lifting scheme-based wavelet transforms are often designated by $(L, \tilde{L})$ where L and $\tilde{L}$ indicate the number of vanishing moments of the analysis and synthesis high pass filters, respectively [7], [19]. In our experiments, we have considered the NSLS(4,2) transform. Note that the supports of the different 2D prediction and update filters of the retained NSLS are shown in Fig. 7.

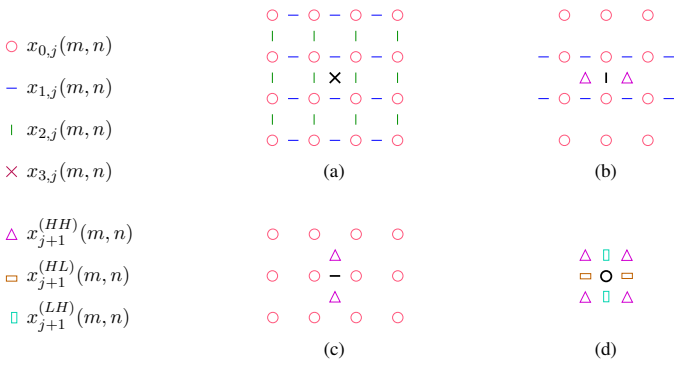


Fig. 7. Spatial supports of the prediction and update filters used to generate: (a) the diagonal detail coefficients $x_{j+1}^{(HH)}$, (b) the vertical detail coefficients $x_{j+1}^{(LH)}$, (c) the horizontal detail coefficients $x_{j+1}^{(HH)}$, and (d) the approximation coefficients x_{j+1}. Note that for every step, the pixels to be predicted and updated are highlighted in black.

Once the spatial supports of the different prediction and update filters are defined, the different proposed methods described in Sections III and IV are tested. The latter will be denoted by

- FCNN-LS: It represents the first proposed FCNN-based lifting scheme design method described in Section III.
- D1-FCNN-LS: It corresponds to the first version of the dynamical FCNN model based on the alternating optimization technique described in Section IV-C.
- D2-FCNN-LS: It corresponds to the second version of the dynamical FCNN model based on the direct optimization technique described in Section IV-C. Note that for both dynamical FCNN models, the overhead cost for transmitting the adaptive weights to the decoder is taken into account in our simulations.
- H-FCNN: It is a hybrid coding method which aims at selecting for each test image the best model between the classical and dynamical ones. In this case, an overhead of only one bit per image, indicating the coding mode, needs to be transmitted to the decoder to reconstruct the image. Note that this extra bit per image is negligible.

Our proposed nonlinear transforms will be firstly compared to the following conventional linear transforms:

- OPT-LS: This is the optimized linear transform NSLS(4,2) where the prediction and update operators are obtained by using the state-of-the-art optimization methods described in Section II.
- JPEG2000: It represents the JPEG2000 compression standard which employs the 5/3 and 9/7 transforms for the lossless and lossy compression modes, respectively.

Moreover, the proposed methods are also compared to recent neural networks-based compression methods. The latter are designated by

- Toderici [22]: In this method, Toderici *et al.* use an architecture where the encoder and decoder are based on a Recurrent Neural Network (RNN).
- Ballé 2017 [23]: This method, developed by Ballé *et al.*, is an end-to-end optimized image compression scheme based on nonlinear transform composed of three successive stages of convolution linear filter and nonlinear activation functions.
- Ballé 2018 [26]: It is based on a variational autoencoder that incorporates a hyperprior to capture the spatial dependencies of the latent representation.
- CNN-LS [36]: This method employs a neural networks-based update-predict lifting structure where the update step is a mean filter and the prediction one is performed using a CNN.

These methods, as well as most of the previous works, are only suitable for lossy compression. For this reason, the above learning based compression methods will be only considered for comparison in terms of Rate-Distortion (R-D) performance. However, since the CNN-LS [36] is based on lifting scheme, it can be slightly modified to be applicable in the context of lossless compression. In this respect, we have reversed the order of the operations by performing a CNN-based prediction first, then applying an update step to generate the approximation coefficients from the previously computed detail signal.

Lossless and lossy compression implementation

The resulting wavelet-based representations (i.e. the classical linear ones as well as the learned FCNN and CNN based ones) are produced using three resolution levels and encoded with the JPEG2000 codec while selecting the appropriate

[3]https://github.com/TassnimDardouri/FCNN_LS

compression mode. The latter is used while disabling the wavelet transform module of JPEG2000 based on 5/3 and 9/7 transforms, for the lossless and lossy compression modes, respectively.
For instance, in the case of lossless compression, the integer wavelet coefficients of the test images are firstly produced (using the rounding operator as described in Section III). Then, the JPEG2000 encoding module is *only* used as an entropy coder to generate the bitstream and evaluate the final bitrate.
Regarding the lossy compression mode, the non-integer wavelet coefficients of the test images are firstly produced using (12) and (16). Then, the JPEG2000 encoding module is employed for quantization and entropy coding. In this respect, for each given target bitrate, the wavelet subbands are partionned into blocks (called codeblocks) of size 64×64 and a rate-distortion optimization is performed by the encoder to find the optimal quantization steps and the embedded bitstream for each codeblock.

C. Performance metrics

The performance of the proposed methods has been evaluated in the context of lossless as well as lossy compression modes using different metrics. In this respect, the compactness of the different wavelet representations is firstly measured in terms of energy of the detail coefficients. The latter allows us to identify the best methods in terms of prediction accuracy and sparsity of the wavelet representation. To this end, for each wavelet representation, the energy of the detail wavelet subbands $x_j^{(o)}$ has been computed. Since $x_j^{(o)}$ represents a prediction error (as defined in Eq. (12)), this energy can be seen as a MSE evaluated over the different prediction steps as follows:

$$\text{MSE} = \sum_{j=1}^{J} \sum_{o \in \{HL,LH,HH\}} \sum_{m=1}^{M_j} \sum_{n=1}^{N_j} \frac{(x_j^{(o)}(m,n))^2}{4^j \times M_j \times N_j} \quad (28)$$

where $x_j^{(o)}$ is the detail subband, of size $M_j \times N_j$, at resolution level j and orientation o.
In addition, the entropy of all the wavelet coefficients is also employed as another measure of compression. It is defined as follows:

$$\mathcal{H} = \sum_{j=1}^{J} \sum_{o \in \{HL,LH,HH\}} \frac{1}{4^j} \mathcal{H}_j^{(o)} + \frac{1}{4^J} \mathcal{H}_J^{(LL)} \quad (29)$$

where $\mathcal{H}_j^{(o)}$ is the entropy of the wavelet subband $x_j^{(o)}$ at resolution level j and orientation o.
While the lossless compression performance is simply evaluated in terms of bitrate, the lossy compression results are generally reported in terms of Rate-Distortion (R-D). In order to quantify this distortion, four metrics have been used. They correspond to Peak Signal-to-Noise Ratio (PSNR), Structural SIMilarity (SSIM), Multi-Scale Structural SIMilarity (MS-SSIM), and Perceptual Image-Error Assessment through Pairwise Preference (PieAPP) metrics [47]. The latter was found to be better-correlated with human opinion compared to the existing image quality assessment methods [47]. Finally, to further compare R-D curves in terms of bitrate saving and quality gain, the Bjøntegaard metric [48] has also been used.

D. Results and discussion

Compactness of wavelet representations
The first round of our experiments has been focused on the evaluation of the compactness of the wavelet representations produced by the aforementioned wavelet-based methods. Tables I and II illustrate the average MSE and entropy values computed over all the test images of both the CLIC and Tecnick-R1 datasets. In terms of energy of the detail wavelet subbands, it can be firstly observed that FCNN-based LS improves the prediction accuracy with respect to the conventional optimized LS. Further improvements are obtained by resorting to the proposed dynamical FCNN architectures. Moreover, in terms of entropy, the proposed FCNN-LS achieves an average gain of about 0.1-0.15 bits per pixel (bpp) compared to the conventional optimized LS. This gain reaches 0.2 bpp with the dynamical FCNN versions. It is important to note here that the two proposed alternating and direct optimization techniques lead to a similar performance in terms of prediction accuracy as well as entropy of the resulting wavelet coefficients. Their MSE and entropy values are also equal to those obtained with the hybrid approach. This shows that the dynamical FCNN-LS outperforms the static FCNN-LS for all images of both datasets. Finally, our FCNN-based methods also outperform the recent state-of-the-art CNN-based lifting scheme architecture in terms of prediction accuracy as well as entropy of the resulting wavelet representation. This may be explained by the fact that CNN-based LS employs a separable structure and is trained by using the reconstruction error as a loss function, which makes it more appropriate for lossy compression as we will show later.
While Tables I and II provide the average gain between the different methods, Figures 8 and 9 illustrate the entropy values of the different images of CLIC and Tecnick-R1 datasets, respectively. These plots show the maximum gain achieved by the proposed methods compared to the conventional ones. For instance, it can be seen that the proposed FCNN-LS leads to a gain of about 0.5 bpp compared to JPEG2000. Moreover, the adaptive FCNN model results in an additional entropy gain that reaches 0.15 bpp.
Lossless compression performance
The second round of our experiments has been devoted to the evaluation of the compression performance of the proposed approaches. In addition to the aforementioned comparison methods, we have also included the HEVC-based lossless coding scheme (using the HEVC Test Model (HM) version HM-16.2) [49], [50] as well as a recent learning based lossless compression method [40]. The latter, designated by Residual Coding (RC), consists in transforming the input image into a lossy reconstruction version and a residual one. Thus, in the context of lossless compression, Table III provides the average bitrates obtained with the CLIC, Tecnick-R1 and Tecnick-R2 image datasets. While the conventional optimized lifting scheme shows similar performance to the JPEG2000 compression standard, an improvement is achieved by the proposed

FCNN-LS design method. The average gain becomes around 0.1-0.14 bpp by resorting to the dynamical FCNN models. Moreover, as mentioned in the previous energy compaction study, one can notice that the two proposed alternating and direct optimization techniques yield similar performance in terms of bitrates. Based on the results of these approaches, one can still conclude that the dynamical FCNN-LS outperforms the static FCNN-LS for all images of both datasets. Finally, it can be again observed that the state-of-the-art CNN-based method [36] has a lower performance in the context of lossless compression whereas the residual based coding method [40] results in similar performance compared to the proposed dynamical FCNN based LS. While this recent method [40] is devoted to lossless compression, our FCNN based LS presents the advantage of being applicable to both lossless and lossy compression.

Lossy compression performance: objective evaluation

Regarding the lossy compression mode, an objective evaluation is firstly performed in terms of R-D results. Fig. 10 illustrates the average results for the three image datasets using PieAPP, SSIM, MS-SSIM and PSNR metrics. According to these plots, different observations can be made. First, JPEG2000 often leads to better PSNR and MS-SSIM values than the different deep learning based image compression methods. However, the SSIM plots show that our proposed static and dynamical FCNN-based methods as well as the hybrid one lead to better results compared to JPEG2000 and the different neural networks-based compression methods, especially at low bitrates. It is worth pointing out that, in a recent study on the quality assessment of deep learning based compressed images [51], it has been shown that simple pixel-based metrics, such as PSNR, are much less accurate to judge the visual quality of decoded images obtained with the emerging deep learning based image compression methods. Based on a careful visual inspection of the quality of the reconstructed images at different bitrates, we have also observed that the proposed FCNN-based approaches yield better subjective results. For this reason, we proposed to quantify the R-D results using a sophisticated perceptual metric (PieAPP) [47]. Note that lower PieAPP values indicate better visual quality. Thus, the PieAPP plots show that neural networks-based compression methods often outperform JPEG2000. Moreover, the proposed FCNN-based approaches often lead to the best compression performance. Fig. 11 illustrates the R-D curves for some images of the CLIC and Tecnick-R2 datasets. We further study the coding performance of the proposed methods for test images with different resolutions and various textures. Fig. 12 illustrates the gain of the hybrid FCNN-LS with respect to the static version for each image of the CLIC and Tecnick-R2 datasets. These plots demonstrate again the interest of the proposed dynamic FCNN-based compression method.

In addition, Tables IV and V show the relative gains of the proposed H-FCNN-LS method compared to CNN-LS [36] and JPEG2000 methods, respectively. The results are provided for low and middle bitrates corresponding respectively to the bitrates $\{0.07, 0.1, 0.15, 0.2\}$ and $\{0.2, 0.25, 0.3, 0.4\}$ bpp. Note that a negative value in terms of bitrate saving (resp. PieAPP difference) indicates a decrease of bitrate (resp. PieAPP) for the same PieAPP (resp. bitrate). Thus, it can be observed that our proposed method outperforms the reference methods in terms of bitrate saving and quality of reconstruction. For instance, the gain is much more significant at low bitrate where it reaches 94% and 35% in terms of bitrate saving compared to JPEG2000 and CNN-LS [36], respectively. It should be noted here that the improvement achieved by the proposed FCNN-LS compared to the recent CNN-LS is mainly due to the main differences between these two approaches summarized in Section I.

Lossy compression performance: subjective evaluation

The proposed dynamical FCNN model is compared to the recent CNN-based LS method as well as the JPEG2000 compression standard in terms of visual reconstruction quality. Figures 13-15 illustrate some reconstructed images of the CLIC and Tecnick-R2 datasets at different bitrates. It can be clearly noticed that the proposed dynamical FCNN-based compression method leads to better perceptual quality yielding reconstructed images with sharp edges.

Ablation study

The impact of some parameters on the coding performance of the proposed FCNN-LS has been also studied. After setting the number of resolution levels to three (i.e. $J = 3$), the influence of the number of hidden layers H is first analyzed while considering four configurations of the FCNN models. The latter corresponds to one hidden layer of size 64, two hidden layers of size 128×16, four hidden layers of size $128 \times 64 \times 32 \times 16$ and six hidden layers of size $128 \times 64 \times 64 \times 64 \times 32 \times 16$. Note that the resulting number of parameters, corresponding to these four configurations, is equal to 36876, 60492, 167244 and 268620, respectively. As expected, Fig. 16 shows that increasing the number of layers generally improves the R-D results. However, it can be noticed that using four or six hidden layers yields similar R-D performance. For this reason, we have chosen an FCNN with four hidden layers in our experiments.

Moreover, we have also studied the impact of the number of resolution levels J used to generate the FCNN-LS based representation (with $H = 4$). With the retained four hidden layers configuration, the FCNN-LS based decomposition requires $55748 \times J$ parameters. Fig. 17 illustrates the R-D results using different resolution levels. It can be seen that increasing the number of levels improves the R-D results. In terms of performance improvements, it appears enough to use three resolution levels as we did in our experiments.

Complexity

Finally, we propose to compare our FCNN-LS and the recent CNN-LS [36] in terms of number of parameters and execution time. Indeed, using a multiresolution representation over three resolution levels, the number of parameters involved in the architecture of the FCNN-LS (resp. CNN-LS) is equal to 167244 (resp. 97489). This difference is mainly due to the fact that the CNN-LS uses a single model for both the horizontal and vertical decomposition of the image. This model is also kept fixed across the different resolution levels. In contrast, our FCNN-LS generates a model for each wavelet subband which will result in $4 \times J$ FCNN models.

Regarding the execution time, using an Intel Xeon(R) pro-

cessor (4 GHz) and a Python implementation, the encoding/decoding time of the FCNN-LS (resp. CNN-LS) is equal to 2.2/0.8 (resp. 1.1/0.7) seconds for an image of size 600×600. Note that the encoding/decoding time of the FCNN-LS based decomposition becomes 8.5/3.2 seconds for an image of size 1200×1200. This runtime is reduced to 0.3/0.08 seconds when the code is executed on an NVIDIA Tesla V100 32Gb GPU. All these results confirm the benefits of the proposed FCNN-based models for adaptive lifting-based coding schemes.

VI. Conclusion and perspectives

New FCNN-based models for the design of adaptive wavelet-based image coding methods are proposed in this paper. While the first model aims to perform the prediction and update lifting stages using a static FCNN architecture, the second one is based on a dynamical architecture that takes into account the characteristics of the input image. Two adaptive learning approaches, based on alternating and direct optimization techniques, have been developed and studied. Experimental results have shown the excellent performance of the proposed approaches in the context of lossless and lossy compression. These results suggest that incorporating neural networks in LS might lead to a third generation of wavelets for compression purposes. In future work, an extension of the proposed FCNN-based models to more sophisticated LS (in particular vector lifting scheme [52]) could be envisaged, to deal with color images. Moreover, end-to-end learning approaches will be investigated. Indeed, while our FCNN models are trained using a loss function depending only on the wavelet coefficients (i.e. the analysis transform), resorting to a rate-distortion loss function taking into account the coding module as well as the synthesis transform could lead to improved results.

TABLE I
Energy compaction performance of wavelet representations of the CLIC image dataset.

Method	MSE	entropy (in bpp)
JPEG2000 (5/3)	188.60	4.41
OPT-LS	149.38	4.36
CNN-LS [36]	292.38	4.89
FCNN-LS	133.17	4.23
D1-FCNN-LS	113.58	4.16
D2-FCNN-LS	113.52	4.16
H-FCNN-LS	113.52	4.16

TABLE II
Energy compaction performance of wavelet representations of the Tecnick-R1 image dataset.

Method	MSE	entropy (in bpp)
JPEG2000 (5/3)	100.24	3.90
OPT-LS	78.73	3.86
CNN-LS [36]	153.91	4.40
FCNN-LS	76.27	3.77
D1-FCNN-LS	62.03	3.72
D2-FCNN-LS	62.36	3.72
H-FCNN-LS	62.36	3.72

TABLE III
Lossless compression performance: average birates (in bpp), using JPEG2000 entropy encoder, for the different image datasets.

Method	Bitrate of CLIC	Bitrate of Tecnick-R1	Bitrate of Tecnick-R2
JPEG2000 (5/3)	4.16	3.73	3.72
HEVC	4.09	3.63	4.03
OPT-LS	4.17	3.66	3.70
CNN-LS [36]	4.66	4.22	4.17
RC [40]	4.02	3.55	3.65
FCNN-LS	4.12	3.62	3.66
D1-FCNN-LS	4.05	3.57	3.63
D2-FCNN-LS	4.05	3.57	3.63
H-FCNN-LS	4.05	3.57	3.63

TABLE IV
Bjøntegaard metric: the average PieAPP differences and the bitrate saving. The gain of "H-FCNN-LS" w.r.t "CNN-LS [36]".

	bitrate saving (in %)		**PieAPP difference**	
Datasets	**low**	**middle**	**low**	**middle**
CLIC	-35.75	-21.67	-0.41	-0.22
Tecnick-R1	-34.79	-17.12	-0.37	-0.14
Tecnick-R2	-35.89	-17.73	-0.32	-0.11

TABLE V
Bjøntegaard metric: the average PieAPP differences and the bitrate saving. The gain of "H-FCNN-LS" w.r.t "JPEG2000".

	bitrate saving (in %)		**PieAPP difference**	
Datasets	**low**	**middle**	**low**	**middle**
CLIC	-94.26	-59.57	-1.36	-0.69
Tecnick-R1	-77.19	-45.14	-0.98	-0.42
Tecnick-R2	-78.30	-45.03	-0.86	-0.33

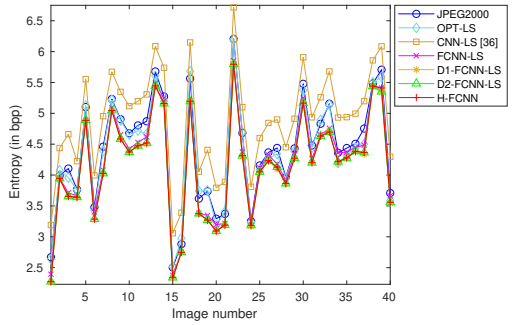


Fig. 8. Entropy values for the different images of the CLIC dataset.

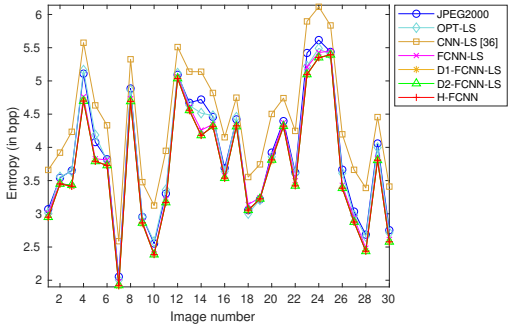


Fig. 9. Entropy values for the different images of the Tecnick-R1 dataset.

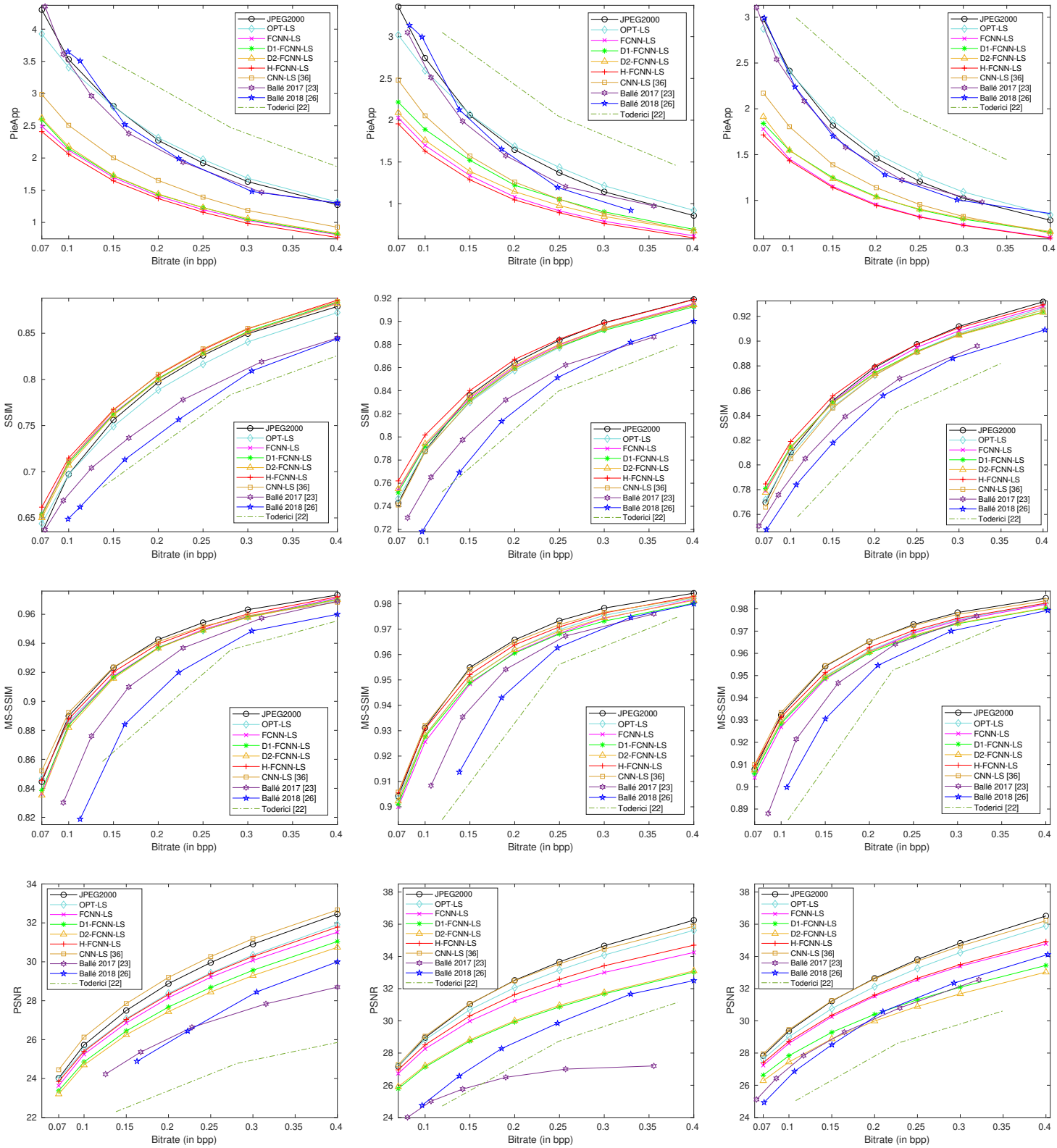


Fig. 10. Average R-D results of the CLIC (first column), Tecnick-R1 (second column) and Tecnick-R2 (third column) image datasets using PieAPP, SSIM, MS-SSIM and PSNR metrics.

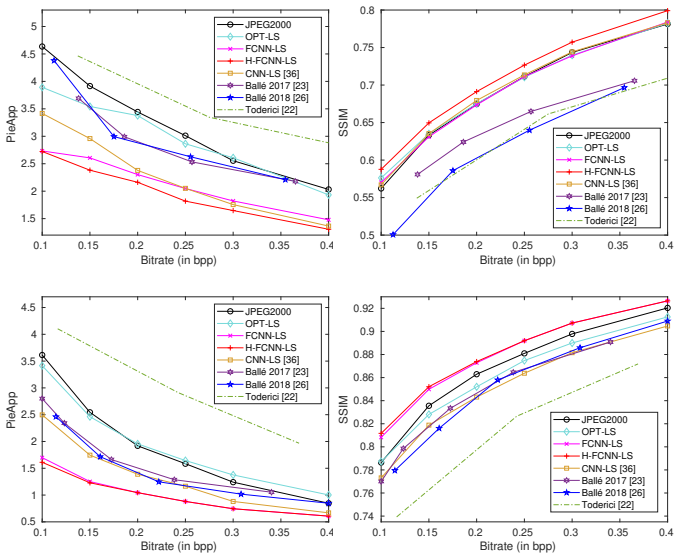


Fig. 11. R-D results in terms of PieAPP and SSIM for image 39 (top) of CLIC and image 19 (bottom) of Tecnick-R2.

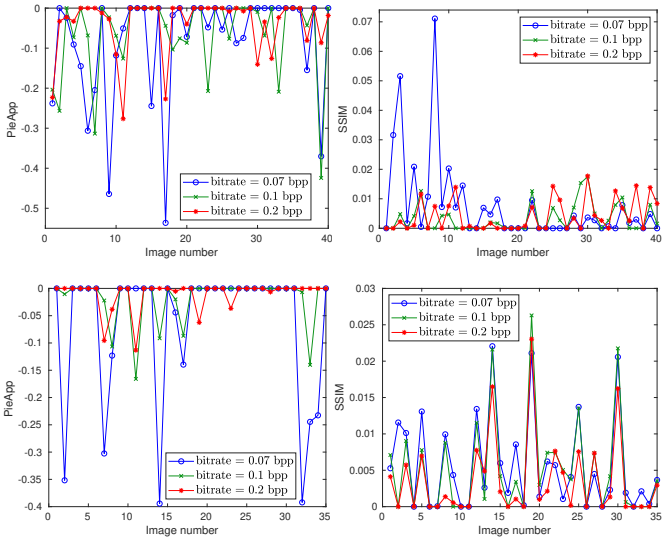
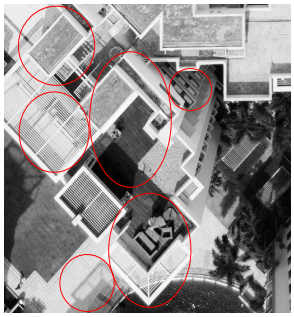


Fig. 12. The gain of the H-FCNN-LS with respect to the FCNN-LS in terms of PieAPP and SSIM for each image of the CLIC (top) and Tecnick-R2 (bottom) datasets.



(a)

(b): PSNR=22.13 dB
PieAPP=4.21

(c): PSNR=22.47 dB
PieAPP=2.81

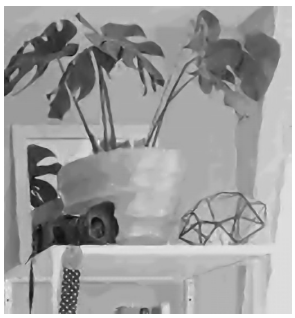
(d): PSNR=21.46 dB
PieAPP=2.26

Fig. 13. (a) Original test image taken from the CLIC dataset. Reconstructed images at 0.25 bpp using: (b) JPEG2000, (c) CNN-LS [36], (d) D2-FCNN-LS.

(a)

(b): PSNR=27.73 dB
PieAPP=3.8

(c): PSNR=28.30 dB
PieAPP=3.1



(d): PSNR=27.31 dB
PieAPP=1.86

Fig. 14. (a) Original test image taken from the CLIC dataset. Reconstructed images at 0.1 bpp using: (b) JPEG2000, (c) CNN-LS [36], (d) D2-FCNN-LS.

(a)

(b): PSNR=26.73 dB
PieAPP=3.16

(c): PSNR=26.86 dB
PieAPP=2.69

(d): PSNR=25.83 dB
PieAPP=1.68

Fig. 15. (a) Original test image taken from the Tecnick-R2 dataset. Zoom on the reconstructed images at 0.1 bpp using: (b) JPEG2000, (c) CNN-LS [36], (d) D2-FCNN-LS.

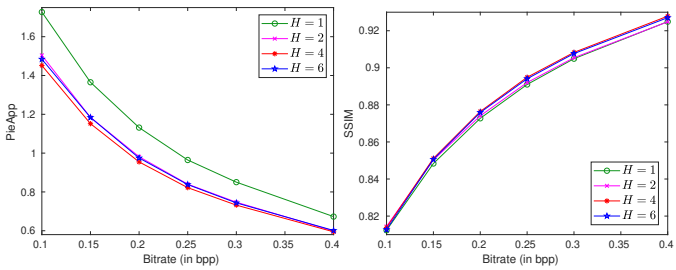


Fig. 16. Effect of the number of hidden layers H on the R-D performance of the FCNN-LS (with $J = 3$) for the Tecnick-R2 dataset.

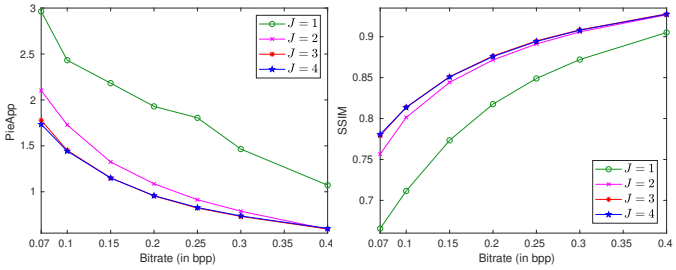


Fig. 17. Effect of the number of resolution levels J on the R-D performance of the FCNN-LS (with $H = 4$) for the Tecnick-R2 dataset.

REFERENCES

[1] T. Guo, H. S. Mousavi, T. H. Vu, and V. Monga, "Deep wavelet prediction for image super-resolution," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Honolulu, HI, USA, July 2017, pp. 104–113.

[2] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, April 1992.

[3] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 2000.

[4] B. Pesquet-Popescu and V. Bottreau, "Three-dimensional lifting schemes for motion compensated video compression," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, Salt Lake City, UT, May 2001, pp. 1793–1796.

[5] M. Kaaniche, A. Benazza-Benyahia, B. Pesquet-Popescu, and J.-C. Pesquet, "Vector lifting schemes for stereo image coding," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2463–2475, 2009.

[6] Y. Xing, M. Kaaniche, B. Pesquet-Popescu, and F. Dufaux, "Adaptive non separable vector lifting scheme for digital holographic data compression," *Applied Optics*, vol. 54, no. 1, pp. A98–A109, January 2015.

[7] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, 1998.

[8] J.-H. Jacobsen, A. W. M. Smeulders, and E. Oyallon, "i-RevNet: Deep invertible networks," in *International Conference on Learning Representations*, Vancouver, Canada, May 2018, pp. 1–11.

[9] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Applied and Computational Harmonic Analysis*, vol. 3, no. 2, pp. 186–200, April 1996.

[10] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[11] F. J. Hampson and J.-C. Pesquet, "M-band nonlinear subband decompositions with perfect reconstruction," *IEEE Transactions on Image Processing*, vol. 7, pp. 1547–1560, November 1998.

[12] J. Solé and P. Salembier, "Generalized lifting prediction optimization applied to lossless image compression," *IEEE Signal Processing Letters*, vol. 14, no. 10, pp. 695–698, October 2007.

[13] Y. Liu and K. N. Ngan, "Weighted adaptive lifting-based wavelet transform for image coding," *IEEE Transactions on Image Processing*, vol. 17, no. 4, pp. 500–511, April 2008.

[14] M. Kaaniche, J.-C. Pesquet, A. Benazza-Benyahia, and B. Pesquet-Popescu, "Two-dimensional non separable adaptive lifting scheme for still and stereo image coding," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Dallas, Texas, USA, March 2010.

[15] A. Gouze, M. Antonini, M. Barlaud, and B. Macq, "Design of signal-adapted multidimensional lifting schemes for lossy coding," *IEEE Transactions on Image Processing*, vol. 13, no. 12, pp. 1589–1603, December 2004.

[16] M. Kaaniche, B. Pesquet-Popescu, A. Benazza-Benyahia, and J.-C. Pesquet, "Adaptive lifting scheme with sparse criteria for image coding," *EURASIP Journal on Advances in Signal Processing: Special Issue on New Image and Video Representations Based on Sparsity*, vol. 2012, no. 1, pp. 1–22, January 2012.

[17] A. Benazza-Benyahia, J.-C. Pesquet, J. Hattay, and H. Masmoudi, "Block-based adaptive vector lifting schemes for multichannel image coding," *EURASIP International Journal of Image and Video Processing*, vol. 2007, no. 1, p. 10 pages, January 2007.

[18] B. Pesquet-Popescu, *Two-stage adaptive filter bank*. first filling date 1999/07/27, official filling number 99401919.8, European patent number EP1119911, 1999.

[19] M. Kaaniche, A. Benazza-Benyahia, B. Pesquet-Popescu, and J.-C. Pesquet, "Non separable lifting scheme with adaptive update step for still and stereo image coding," *Elsevier Signal Processing: Special issue on Advances in Multirate Filter Bank Structures and Multiscale Representations*, vol. 91, no. 12, pp. 2767–2782, January 2011.

[20] W. Ding, F. Wu, X. Wu, S. Li, and H. Li, "Adaptive directional lifting-based wavelet transform for image coding," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 416–427, February 2007.

[21] E. Martinez-Enriquez, J. Cid-Sueiro, F. D. de Mari a, and A. Ortega, "Directional transforms for video coding based on lifting on graphs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 4, pp. 933–946, November 2016.

[22] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Computer Vision and Pattern Recognition*, Las Vegas, USA, June 2016, pp. 5306–5314.

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

[24] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *International Conference on Machine Learning*, Sydney, Australia, August 2017, pp. 1–9.

[25] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June 2018, pp. 3214–3223.

[26] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations*, Vancouver, Canada, May 2018, pp. 1–47.

[27] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *International Conference on Neural Information Processing Systems*, Montréal, Canada, December 2018, p. 10794–10803.

[28] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and V. G. Luc, "Generative adversarial networks for extreme learned image compression," in *International Conference on Learning Representations*, New Orleans, Louisiana, USA, May 2019, pp. 1–31.

[29] M. A. Yilmaz and A. M. Tekalp, "Effect of architectures and training methods on the performance of learned video frame prediction," in *International Conference on Image Processing*, Taipei, Taiwan, September 2019, pp. 1–5.

[30] I. Schiopu and A. Munteanu, "Macro-pixel prediction based on convolutional neural networks for lossless compression of light field images," in *International Conference on Image Processing*, Athens, Greece, October 2018, pp. 445–449.

[31] 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.

[32] T. Dumas, A. Roumy, and C. Guillemot, "Context-adaptive neural network-based prediction for image compression," *IEEE Transactions on Image Processing*, vol. 29, no. 1, pp. 679–693, August 2019.

[33] D. Liu, H. Ma, Z. Xiong, and F. Wu, "CNN-based DCT-like transform for image compression," in *International Conference on Multimedia Modeling*, Bangkok, Thailand, February 2018, pp. 61–72.

[34] E. Ahanonu, M. Marcellin, and A. Bilgin, "Lossless image compression using reversible integer wavelet transforms and convolutional neural networks," in *Data Compression Conference*, Snowbird, UT, USA, March 2018, p. 1.

[35] P. Akyazi and T. Ebrahimi, "Learning-based image compression using convolutional autoencoder and wavelet decomposition," in *Conference on Computer Vision and Pattern Recognition Workshops*, Long Beach, CA, USA, June 2019, p. 5.

[36] H. Ma, D. Liu, R. Xiong, and F. Wu, "iWave: CNN-based wavelet-like transform for image compression," *IEEE Transactions on Multimedia*, vol. 22, no. 7, pp. 1667–1679, July 2020.

[37] I. Schiopu and A. Munteanu, "Deep-learning based lossless image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 1829–1842, April 2019.

[38] ——, "A study of prediction methods based on machine learning techniques for lossless image coding," in *International Conference on Image Processing*, Abu Dhabi, United Arab Emirates, October 2020, pp. 1–5.

[39] F. Mentzer, E. Agustsson, M. Tschannen, and R. Timofte, "Practical full resolution learned lossless image compression," in *Conference on Computer Vision and Pattern Recognition*, Long Beach, California, October 2019, pp. 1–14.

[40] F. Mentzer, L. V. Gool, and M. Tschannen, "Learning better lossless compression using lossy compression," in *Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, June 2020, pp. 6637–6646.

[41] T. Dardouri, M. Kaaniche, A. Benazza-Benyahia, and J.-C. Pesquet, "Optimized lifting scheme based on a dynamical fully connected network for image coding," in *accepted in International Conference on Image Processing*, Abu Dhabi, United Arab Emirates, October 2020, pp. 1–5.

[42] Y.-K. Sun, "A two-dimensional lifting scheme of integer wavelet transform for lossless image compression," in *International Conference on Image Processing*, vol. 1, Singapore, October 2004, pp. 497–500.

[43] Y. Wang, X. Fan, C. Jia, D. Zhao, and W. Gao, "Neural network based inter prediction for HEVC," in *IEEE International Conference on Multimedia and Expo*, San Diego, USA, July 2018, pp. 1–6.

[44] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, San Siego, USA, May 2015, pp. 1–15.

[45] N. Asuni and A. Giachetti, "Test images: a large-scale archive for testing visual devices and basic image processing algorithms," in *Eurographics Italian Chapter Conference*, Cagliari, Italy, September 2014, pp. 1–3.

[46] ——, "Test images: A large data archive for display and algorithm testing," *Journal of Graphics Tools*, vol. 17, no. 4, pp. 113–125, February 2015.

[47] E. Prashnani, H. Cai, Y. Mostofi, and P. Sen, "PieAPP: Perceptual image-error assessment through pairwise preference," in *IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June 2018, pp. 1–10.

[48] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves," ITU SG16 VCEG-M33, Austin, TX, USA, Tech. Rep., 2001.

[49] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," vol. 22, p. 1649–1668, Dec 2012.

[50] F. Bossen, "Common hm test conditions and software reference configurations," JCT-VC document, JCTVC-G1100, San Jose, CA, USA, Tech. Rep., 2012.

[51] G. Valenzise, A. Purica, V. Hulusic, and M. Cagnazzo, "Quality assessment of deep-learning-based image compression," in *International Workshop on Multimedia Signal Processing*, Vancouver, Canada, August 2018, p. 1–6.

[52] O. Dhifallah, M. Kaaniche, and A. Benazza-Benyahia, "Efficient joint multiscale decomposition for color stereo image coding," in *European Signal and Image Processing Conference*, Lisbon, Portugal, September 2014, pp. 1–5.