Lossless Image Compression Based on Image Decomposition and Progressive Prediction Using Convolutional Neural Networks

Jae Hoon Shim, Hochang Rhee, Yeong Il Jang, Geonsu Lee, Seyun Kim, Nam Ik Cho*
* Department of ECE, INMC, Seoul National University, Seoul, Republic of Korea
E-mail: joinjhs@ispl.snu.ac.kr

***Abstract*—This paper presents a lossless image compression method based on the image decomposition and progressive prediction of decomposed images and their coding contexts using convolutional neural networks (CNNs). We first decompose a given input into sub-images by sub-sampling into horizontal and vertical directions. The first sub-image is encoded by an existing non-learning lossless compressor, and the others are encoded progressively using the already encoded ones. While state-of-the-art learning-based encoders proceed with the auto-regressive prediction (pixel-by-pixel raster-scan order prediction) so that the runtime is not practical, our method predicts each sub-image at once and thus executes in practical time. We also design the CNNs to predict pixel values and coding contexts jointly and send the prediction error to the arithmetic encoder for the given coding context. In the case of color input, it is converted to YUV using a reversible transform, and each channel is partitioned in the same way above. Then, all the sub-images are processed progressively from Y to V. Experiments on high-resolution datasets show that the proposed method outperforms all the non-learning codecs and practical-time CNN-based encoders. Compared to learning-based auto-regressive approaches, our method shows comparable results while requiring much less runtime.**

I. Introduction

With the advancement of imaging and display devices, ultra-high-definition (UHD) cameras and displays are getting more affordable. Thus, a demand for more efficient image compression methods is growing as well. Lossy compression is widely adopted for image archiving and transmission due to limited storage capacity or channel bandwidth. On the other hand, lossless compression is required for medical and science applications where the integrity of image information is needed. Also, as the price of memory is getting lower and the quality of cameras are getting higher, people may wish to save their artistic and valuable shots losslessly.

The approaches for lossless compression can be separated into two, engineered (non-learning) codecs and learning-based codecs. Some examples of engineered lossless codecs are PNG [4], WebP [25], JPEG2000 lossless mode [21], JPEG-LS [22], BPG-lossless [3], FLIF [23], etc., which achieve desirable performances with reasonable computation time. These methods focus on reducing spatial redundancy in images through pixel prediction and entropy encoders.

Deep neural networks have achieved significant successes in artificial intelligence and signal processing fields, providing promising solutions in image compression as well. The deep learning-based lossless encoders can be roughly classified into two; one is based on the raster scan order pixel prediction conditioned on causal neighbors, and the other is to find the probability model of the entire image. In other words, the methods in the first category are exploiting auto-regressive pixel prediction, and some examples are pixel-RNN [24], pixel-CNN [11], MS-pixel-CNN [13], and Schiopu's methods [16], [18], [17], [19]. Another type of auto-regressive method is the integer discrete flow (IDF) [6], which achieves similar performance to the pixel-CNN. The deep-learning-based auto-regressive methods are shown to provide better performance than all the engineered codecs. However, since the pixel prediction is performed one by one due to the nature of auto-regressive prediction that can use only causal neighbors, they take very long computation times. Specifically, the CNN-based prediction should be performed $H \times W$ times where H and W are the height and width of the image, and hence takes impractical runtime for HD images. Hence, the performances of pixel-CNN and pixel-RNN are reported only for very small images such as ImageNet-32 and ImageNet-64 [5]. The fastest CNN-based auto-regressive methods may be Schiopu's [19], but according to their paper, it takes more than 12 minutes for compressing a 4K grayscale image.

To exploit CNN's capability for lossless compression while requiring fewer computations, Mentzer *et al.* [10], [9] introduced methods to predict a whole image instead of predicting each pixel. In [10], they first encode an image by a non-learning lossy compressor (BPG), and then find the probability model of the image conditioned on the lossy compressed image. In terms of bits per pixel (bpp), it outperforms existing non-learning-based methods except for the FLIF and shows better or worse results than the FLIF depending on the dataset. The virtue of these methods is the practical computation time compared to the auto-regressive methods. It needs less computation time compared to the FLIF, though requiring GPU computation.

In this paper, we propose a new lossless image compression method that belongs to the second category. Specifically, our method does not proceed with pixel by pixel but predicts partitioned blocks in a progressive manner. Hence, the proposed method needs practical GPU runtime closer to the

CPU time of FLIF while providing lower bpp. For a given input, we first partition the image into sub-images by subsampling. Then, one of the sub-images is compressed by the best non-learning encoder FLIF, and it is used to predict the next sub-image. These "already-encoded" sub-images are concatenated and used to predict the next, and so on. Since the sub-images are encoded one by one progressively, we name our method Progressive Lossless Image Compression via Image Partitioning (PLIC-IP). By the partitioning and block-wise progressive coding, we can reduce the computation time and also exploit noncausal neighbors from the second sub-image. For the color input, the RGB is converted by an integer reversible color transform [12], and Y is compressed by the above mentioned process. For U and V, they are also partitioned the same, and the first sub-image is also compressed by the FLIF. From the second sub-image, the sub-images are also progressively compressed using the already-encoded ones. Experiments on several high-resolution datasets show that our method requires practical runtime as stated above, and the bpp comes in between the FLIF and CNN-based auto-regressive methods.

II. Related work

A. Engineered lossless codecs/arithmetic coding

One of the widely used engineered lossless image compression codecs is PNG, which uses LZ77 and Huffman coding to compress an image. Also, JPEG2000 has both lossy and lossless compression modes, where the lossy JPEG2000 is also a competent method. More recent codecs, such as WebP, BPG, and FLIF, generally show better performances than the PNG and JPEG 2000. Specifically, the WebP improves the performance by applying different entropy codecs for each channel. BPG is based on HEVC/H.265, which shows faster encoding/decoding speed than WebP. FLIF outperforms all of the codecs above, which uses the reversible YCoCg color space and applies a context-adaptive arithmetic coder named MANIAC (Meta-Adaptive Near-zero Integer Arithmetic Coding).

The key ingredients of lossless compression are the predictor and entropy coder, where arithmetic coding is widely adopted as the entropy coder in recent codecs. When the approximate distribution of specific data is known, the arithmetic coder assigns fewer bits to frequently appearing symbols and vice versa. According to Shannon's source coding theorem [20], without information loss, the size of the compressed bitstream has a lower bound defined by the entropy. If the predicted distribution gets closer to the true distribution, the size of the compressed bitstream gets closer to the optimal bits. In this regard, various neural network-based attempts have been introduced to precisely predict the distribution of natural images. In an image, there are high-frequency pixels, such as the ones around the edges and within textured areas, where the pixel variation is relatively large. Since the distribution of the high-frequency pixels differs from that of the low-frequency ones, applying various arithmetic coders for various frequencies yields a high compression rate. Hence, the selection of the arithmetic coder needs to be based on context (property of pixel values around the encoding pixel), where we adopt the arithmetic coder from [14], [15].

B. Pixel Prediction

Various approaches to compressing data via deep learning have been introduced, along with the prosperity of deep learning technologies. As stated above, pixel prediction is one of the most important parts of lossy/lossless codecs that influences compression performance. In the following subsections, we categorize the pixel prediction method into two types, pixel by pixel prediction (auto-regressive type) and block prediction.

1) Pixel-by-pixel prediction: In natural images, each pixel value is highly correlated with adjacent ones. Oord [11] put causal neighbors into the neural network to predict the pixel value, where a masked convolution was introduced to prevent a non-causal neighbor from affecting the prediction. This method ensures precise pixel value prediction, leading to a great decrease in the size of the encoded bitstream. However, the pixel-by-pixel prediction method has a critical limitation. Specifically, a tremendous amount of computational cost is needed since one neural network computation is needed per one-pixel prediction.

2) Block prediction-based codecs: The pixel-by-pixel prediction using CNN takes a very long computation time and large power consumption with GPU, and thus it is impractical to encode a high-quality image over 2K resolution, as stated previously. For reducing the computation time in pixel prediction, block prediction methods were also introduced, where a neural network predicts a whole image or a block of pixels at once. For example, Mentzer *et al.* [10] used BPG lossy coding for the block prediction. A BPG coded image is treated as a prediction, then the prediction is put into the neural network and the arithmetic coder. The BPG is known to be a fast lossy coder, and thus they could significantly reduce the computation time for the prediction compared to the auto-regressive types. Our work is also based on a block-based prediction framework to achieve practical lossless compression.

III. Proposed method

A. Image partitioning

Different from Mentzer *et al.* [10], we propose image partitioning for the prediction of a block of pixels, the process of which is illustrated in Figure 1. Before partitioning the image, the RGB input is converted to YUV by using an invertible transform, like in most lossless color image compression methods. Then, image partitioning is applied for each channel, specifically partitioning of each channel into 12 blocks.

For the YUV-transformed image $X \in \mathbb{Z}^{H \times W \times C}$, we apply the image partitioning and set the sub-images in order. We call this PP (partition-prioritized) ordering. For channel types $c = 0, 1, 2$ (each denotes Y, U, V channel, respectively), the PP ordered images $X_{D_k} \in \mathbb{Z}^{H/2 \times W/2} (k = 0, 1, ..., 11)$ are expressed as

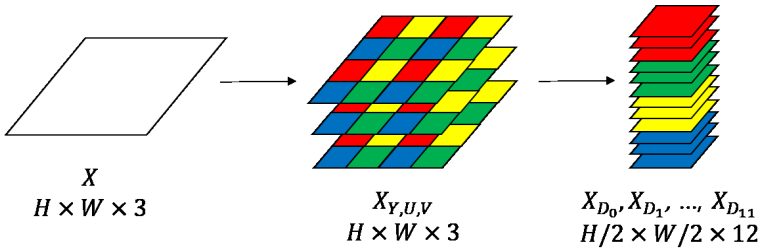

Fig. 1. The decomposition of an image.

$$X_{D_c}(m,n) = X(2m, 2n, c) \quad (1)$$
$$X_{D_{c+3}}(m,n) = X(2m+1, 2n+1, c) \quad (2)$$
$$X_{D_{c+6}}(m,n) = X(2m, 2n+1, c) \quad (3)$$
$$X_{D_{c+9}}(m,n) = X(2m+1, 2n, c). \quad (4)$$

B. Hierarchical network

We first encode the first partition of each channel using the engineered codec FLIF. The firstly-encoded data $X_{\downarrow 2} \in \mathbb{Z}^{H/2\times W/2\times C}$ can be expressed as

$$X_{\downarrow 2} = \bigcup_{i=0}^{2} X_{D_i}, \quad (5)$$

and its pixel-wise brief expression can be described as as

$$X_{\downarrow 2}(m,n,c) = X(2m, 2n, c). \quad (6)$$

Given that we know $X_{\downarrow 2}$, the other 9 ordered images can be defined as the conditional probability of previous images, described as

$$p(X_{D_i}, X_{D_0}, X_{D_1}, ..., X_{D_{i-1}}) = p(X_{D_i}|X_{D_0}, X_{D_1}, ..., X_{D_{i-1}}) \cdot (X_{D_0}, X_{D_1}, ..., X_{D_{i-1}}). \quad (7)$$

Since $X_{D_0}, X_{D_1}, X_{D_2} \in X_{\downarrow 2}$ are already encoded, there are 9 possible i values for equation (7), $i = 3, 4, ..., 11$.

In our work, inspired by Kim [7], 9 neural networks(CNN_i) are modeled to obtain the prediction $\hat{X}_{D_i}$, where the overall encoder network is shown in Figure 2. Each network also predicts the context of the target image, $C_i \in \mathbb{Z}^{H/2\times W/2}$. Then, the context value and the prediction are forwarded to the adaptive arithmetic coder. Returning to the neural network, by the equation (7), the prediction $\hat{X}_{D_i}$ and the context C_i can be defined as a function of other causal sub-images as

$$\hat{X}_{D_i} = f_i(X_{D_0}, X_{D_1}, ..., X_{D_{i-1}})$$
$$C_i = g_i(X_{D_0}, X_{D_1}, ..., X_{D_{i-1}}). \quad (8)$$

However, for already encoded three sub-images $\{X_{D_i}|X_{D_i} \subset X_{\downarrow 2}\}$, the prediction and the context are not calculated. This is the reason why 9 neural networks are needed, instead of 12.

The context is learned to predict the prediction error, *i.e.*, the difference between the original image and the predicted image. If a pixel is easy to predict, in other words, if it is in the low-frequency area, the context value for the pixel will be near zero. On the contrary, if a pixel value is hard to estimate, the context value will be relatively large. Based on the size of the context value $C_i(m,n)$, the pixel is assigned to one of 24 arithmetic coders by the mapping function $L \in \mathbb{Z}$. Thus, the adaptive arithmetic coder (AAC) takes the original pixel value $X_{D_i}(m,n)$, the predicted pixel value $\hat{X}_{D_i}(m,n)$, and the context value $C_i(m,n)$ as inputs, and generates bitstream, described as

$$bitstream_i(m,n) = AAC(X_{D_i}(m,n), \hat{X}_{D_i}(m,n), L(C_i(m,n))). \quad (9)$$

For each arithmetic coder, accumulated prediction values form a distribution function $pdf_j(j = L(C_i(m,n)), j \in \{1, 2, ...24\})$. These 24 different distribution functions ensures the effective compression. In the training stage and encoding stage, all networks are computed parallel since all original sub-images X_{D_i} are known, while in the decoding stage, the images are restored in sequential order.

C. Loss

For each neural network, two losses are defined; one is the prediction loss and the other the context loss. The prediction loss is defined as the mean difference between the original and the predicted images, where the difference ($E_{D_i} \in \mathbb{Z}^{H/2\times W/2}$) and its mean (the loss $L_{pred,i}$) are expressed as

$$E_{D_i}(m,n) = |\hat{X}_{D_i}(m,n) - X_{D_i}(m,n)| \quad (10)$$
$$L_{pred,i} = \frac{1}{HW}\sum_{m,n}^{H,W} E_{D_i}(m,n). \quad (11)$$

The context loss is defined as the mean difference between the context and the prediction error, described as

$$L_{ctx,i} = \frac{1}{HW}\sum_{m,n}^{H,W} |C_i(m,n) - E_{D_i}(m,n)|. \quad (12)$$

Thus, the overall loss function is denoted as

$$L_{overall} = \sum_{i=3}^{11} L_{pred,i} + \lambda L_{ctx,i}, \quad (13)$$

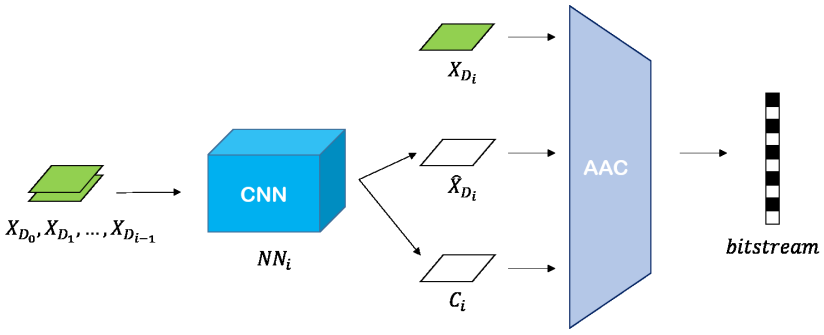

Fig. 2. The overall encoding structure.

where λ is a parameter that balances two losses.

IV. Experiments

A. Experimental Settings

1) Dataset: To evaluate the performances under practical circumstances, we test on high-resolution images (2K images). Thus we select DIV2K [2] dataset and Flickr2K [1] dataset for training and evaluation. DIV2K dataset is a widely-used high-resolution image dataset. It is divided into 800 training data and 100 validation data. We use all 800 training data for training. For evaluation (encoding/decoding), we use DIV2K original validation dataset and the randomly cropped version of DIV2K validation dataset (denoted as DIV2K (crop)). The crop size is set to 512×512 for the fair comparison to L3C[9]. Flickr2K dataset consists of 2,650 2K images, and we sampled 100 images among them. For the evaluation of Flickr2K dataset, the network is trained with DIV2K training dataset.

2) Network details: All neural network NN_3, NN_4, ..., NN_{11} are composed of 4 $3 \times 3 \times 64$ convolutional layers and 3 ReLU layers between them. We designed such simple networks for the low computational cost.

B. Training Procedures

We first randomly crop the original image to size 256×256 and batch size 4. We used Adam optimizer [8] for the optimization. Setting the learning rate to 10^{-4} for the better convergence, we first train each neural network CNN_i individually by using only loss terms ($L_{pred,i} + \lambda L_{ctx,i}$) related to the CNN_i, for 3,000 epochs (2,400K iterations) each. After this step, we train all networks with the overall loss defined in equation (13) for 500K epochs (400,000K iterations). We used 1 for the value of the balancing factor λ.

V. Result

A. Comparison with other codecs

We evaluate the coding result on DIV2K and Flickr2K in terms of bpp. We compare the result with the other codecs in Table I. Compared to ours, PNG and BPG show almost 50% higher bpp in DIV2K full-resolution dataset. Our method also outperforms WebP and FLIF. Our method shows the highest performance as well on the 512×512 cropped dataset (DIV2K(crop)). It outperforms all the other engineered codecs and L3C, the high-performance learning-based codec.

B. Runtime comparison

We also compare the encoding speed between our method and other codecs in Table II. While PNG shows the fastest speed, followed by BPG, our method is 4.8 times faster than WebP and 2.4 times faster than FLIF on 512×512 cropped DIV2K dataset in terms of encoding time. On the other hand, L3C shows a faster speed than ours by about three times.

C. Performance comparison between different experimental settings

We also test another way of ordering the partitioins, such that the sub-images from Y channel come first, followed by sub-images from U and V. We call this CP (channel-prioritized) ordering, described as

$$X_{D_c}(m,n) = X(2m, 2n, c) \tag{14}$$
$$X_{D_{3c+3}}(m,n) = X(2m+1, 2n+1, c) \tag{15}$$
$$X_{D_{3c+4}}(m,n) = X(2m, 2n+1, c) \tag{16}$$
$$X_{D_{3c+5}}(m,n) = X(2m+1, 2n, c) \tag{17}$$

For comparing the effectiveness of the hierarchical prediction schemes CP and PP (Section III), we set a baseline that does not exploit the dependency of channels, and compare their performances in Table III. In addition, we add experiments with the extension of layers. As shown in Table III, models with eight layers do not show any improvements in terms of bpp compared to the models with four layers, which means that only four layers are enough to generate accurate predictions and contexts. The result also shows that the bpp of the baseline is worse than that of PP and CP orderings. With this result, we can see that the correlation between the different channels is significant, and thus better results can be obtained by exploiting the hierarchical encoding. Also, PP ordering provides better compression performance than CP. This shows that the order of the image partitioning is also important.

TABLE I
BITS PER PIXEL (BPP) OF LOSSLESS COMPRESSION METHODS ON HIGH RESOLUTION DATASETS.

Type	Method	Flickr2K	DIV2K	DIV2K(crop)
Engineered Codecs	PNG[4]	12.74	12.69	14.62
	BPG[3]	13.69	13.26	14.05
	WebP[25]	9.32	9.34	9.68
	FLIF[23]	8.73	8.74	9.35
Learning-based Codecs	L3C[9]	9.53	9.52	9.70
	Ours(PLIC-IP)	8.69	8.57	9.14

TABLE II
ENCODING TIME(S) OF OUR METHOD COMPARED TO OTHER CODECS ON 512×512 CROPPED DIV2K DATASET.

Type	Method	DIV2K(crop)
Engineered Codecs	PNG[4]	0.001
	BPG[3]	0.224
	WebP[25]	3.105
	FLIF[23]	1.572
Learning-based Codecs	L3C[9]	0.242
	Ours(PLIC-IP)	0.647

TABLE III
PERFORMANCE COMPARISON BETWEEN DIFFERENT EXPERIMENTAL SETTINGS ON DIV2K DATASET.

Setting	bpp
baseline	9.004
PP	8.574
PP(layer 8)	8.596
CP	8.632
CP(layer 8)	8.655

D. Computation Bottleneck

Table IV shows the average running time for each step in encoding and decoding. FLIF enc/dec denote the times for encoding and decoding of $X_{\downarrow 2}$ using FLIF. Encoding/decoding NN denotes the encoding/decoding time, including neural network computing and arithmetic coding. Merge means the time consumption for merging all 12 restored sub-images into one whole RGB image. Note that it takes more time to decode an image than encoding. It is because parallel computing is performed in the encoding stage, while serial computing is necessary for the decoding stage.

TABLE IV
RUNNING TIME FOR EACH STEP IN ENCODING/DECODING ON DIV2K DATASET.

Stage	Elapsed time(s)	Elapsed time(%)
FLIF enc	4.684	38.8%
encoding NN	2.091	17.3%
FLIF dec	0.022	0.2%
decoding NN	3.332	27.6%
merge	1.931	16.0%
total	12.060	100.0%

VI. CONCLUSION

In this paper, we have proposed a lossless image compression framework that utilizes image partitioning, adaptive arithmetic coder, and hierarchical network. The correlation between the partitioned images is enhanced in the image partitioning stage so that we could use the correlation in the hierarchical network. The outputs of the network are forwarded to the adaptive arithmetic coder, yielding a competent result. Our method outperforms engineered codecs, PNG, BPG, WebP, and FLIF while achieving practical running time on high-quality 2K images from DIV2K and Flickr2K datasets.

VII. ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (2021R1A2C2007220).

REFERENCES

[1] *Flickr Dataset*. Accessed: July. 13, 2021. [Online]. Available: https://github.com/LimBee/NTIRE2017.

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. IEEE, July 2017.

[3] Umar Albalawi, Saraju P Mohanty, and Elias Kougianos. A hardware architecture for better portable graphics (bpg) compression encoder. In *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, pages 291–296. IEEE, 2015.

[4] Thomas Boutell and T Lane. Png (portable network graphics) specification version 1.0. *Network Working Group*, pages 1–102, 1997.

[5] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[6] Emiel Hoogeboom, Jorn WT Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. *arXiv preprint arXiv:1905.07376*, 2019.

[7] S. Kim and N. I. Cho. Hierarchical prediction and context adaptive coding for lossless color image compression. *IEEE Transactions on Image Processing*, 23(1):445–449, 2014.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10629–10638, 2019.

[10] Fabian Mentzer, Luc Van Gool, and Michael Tschannen. Learning better lossless compression using lossy compression, 03 2020.

[11] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

[12] S. Pei and J. Ding. Improved reversible integer-to-integer color transforms. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 473–476, 2009.

[13] Scott Reed, Aäron Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando Freitas. Parallel multi-scale autoregressive density estimation. In *International Conference on Machine Learning*, pages 2912–2921. PMLR, 2017.

[14] H. Rhee, Y. I. Jang, S. Kim, and N. I. Cho. Channel-wise progressive learning for lossless image compression. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1113–1117, 2020.

[15] Hochang Rhee, Yeong Il Jang, Seyun Kim, and Nam Ik Cho. Lossless image compression by joint prediction of pixel and context using duplex neural networks. *IEEE Access*, 2021.

[16] I Schiopu and A Munteanu. Residual-error prediction based on deep learning for lossless image compression. *Electronics Letters*, 54(17):1032–1034, 2018.

[17] Ionut Schiopu, Moncef Gabbouj, Atanas Gotchev, and Miska M Hannuksela. Lossless compression of subaperture images using context modeling. In *2017 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4. IEEE, 2017.

[18] Ionut Schiopu and Adrian Munteanu. Macro-pixel prediction based on convolutional neural networks for lossless compression of light field images. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 445–449. IEEE, 2018.

[19] Ionut Schiopu and Adrian Munteanu. Deep-learning-based lossless image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(7):1829–1842, 2019.

[20] Shannon. A mathematical theory of communication. In *Bell System Technical Journal*, volume 27, pages 379–??23, 623–656. IEEE, 1948.

[21] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[22] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[23] Jon Sneyers and Pieter Wuille. Flif: Free lossless image format based on maniac compression. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 66–70. IEEE, 2016.

[24] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

[25] WebEngines Blazer Platform Version. 1.0 hardware reference guide, xp-002202892, network engines. *Inc., Jun*, 1:92, 2000.