entropy

MDPI

Article

Lossy P-LDPC Codes for Compressing General Sources Using Neural Networks

Jinkai Ren [1], Dan Song [1], Huihui Wu [2] and Lin Wang [1,*]

[1] Department of Information and Communication Engineering, Xiamen University, Xiamen 361005, China
[2] Department of Electrical and Computer Engineering, McGill University, Montreal, QC H4H 1R3, Canada
* Correspondence: wanglin@xmu.edu.cn

Abstract: It is challenging to design an efficient lossy compression scheme for complicated sources based on block codes, especially to approach the theoretical distortion-rate limit. In this paper, a lossy compression scheme is proposed for Gaussian and Laplacian sources. In this scheme, a new route using "transformation-quantization" was designed to replace the conventional "quantization-compression". The proposed scheme utilizes neural networks for transformation and lossy protograph low-density parity-check codes for quantization. To ensure the system's feasibility, some problems existing in the neural networks were resolved, including parameter updating and the propagation optimization. Simulation results demonstrated good distortion-rate performance.

Keywords: lossy compression; neural networks; general sources; P-LDPC codes; distortion-rate performance

Citation: Ren, J.; Song, D.; Wu, H.; Wang, L. Lossy P-LDPC Codes for Compressing General Sources Using Neural Networks. *Entropy* **2023**, *25*, 252. https://doi.org/10.3390/e25020252

Academic Editors: T. Aaron Gulliver, Jun Chen and Sadaf Salehkalaibar

Received: 28 November 2022
Revised: 14 January 2023
Accepted: 29 January 2023
Published: 30 January 2023

1. Introduction

It is known that low-density parity-check (LDPC) codes have capacity-approaching performance as channel codes [1,2]. As a consequence, LDPC codes have been widely used in modern communication standards and in industrial applications. To simplify the structure, more constructive protograph LDPC (P-LDPC) codes are introduced with lower decoding complexity [3]. Furthermore, P-LDPC codes have good coding properties, and they can be easily optimized by convergence analysis of mutual information [4].

The duality between the lossy source coding and channel decoding is found with the compression of the Bernoulli sources [5,6]. Existing works show that LDPC codes have been developed for compressing the binary symmetric sources [7,8]. For instance, the belief propagation (BP) and its modifications are employed to be good candidates for lossy source coding [9,10].

Following this fact, more constructive P-LDPC code was introduced to replace the LDPC code. In [11], the BP algorithm based on the P-LDPC code was firstly proposed to compress the binary source with good performance. Then, Ref. [12] demonstrated the P-LDPC-based encoding algorithm can simultaneously overcome the source-compression distortion and channel-noise impact. Furthermore, the BP based on the P-LDPC code was firstly used for Gaussian source compression in [13]. In these cases, one P-LDPC code could be used simultaneously in source coding and channel coding to implement different functions. This is friendly to hardware manufacturing by reusing the P-LDPC decoding chip.

However, the aforementioned conventional algorithms and methods are complicated and time-consuming. It should be noted that the BP algorithm needs more iterations in the coding procedure. Moreover, the "quantization-compression" scheme is complicated, including two steps; see [13]. First, the Gaussian source is quantized to a binary sequence by using a high rate quantizer. Then, the binary sequence is compressed by P-LDPC codes.

It should be noted that the lossy compression inevitably brings bit errors, and each quantized bit contains different information. This uneven distribution of bit information

results in large distortion in the source reconstruction. One solution is to use the multilevel coding (MLC) with set partitioning and rate allocation at different levels to homogenize distortions [13]. However, the MLC still has some problems. First, the MLC is more complex than the single-level coding structure. Second, the set partitioning cannot completely homogenize distortions by increasing coding levels. Third, it is difficult to find an optimal rate-allocation scheme. Table 1 briefly describes the mentioned literature, and Table 2 compares the methods and sources of the literature.

Table 1. Literature description.

Literature	Main Contribution
Braunstein [9]	Lossy compression of binary sources using reinforced belief propagation decoding algorithm of LDPC
Fang [10]	Lossy compression of binary source using sliding-window BP decoding algorithm of LDPC
Liu [11]	Use P-LDPC code for binary source compression
Wang [12]	Performance of binary source lossy compression using P-LDPC in AWGN channel
Deng [13]	Use P-LDPC code for Gaussian source compression
Proposed scheme	Designed the RMD algorithm, combining the neural network with P-LDPC, and realized the lossy compression of general information sources

Table 2. Literature comparison.

Literature	LDPC Type	Method	Sources
Braunstein [9]	LDPC	RBP	Binary source
Fang [10]	LDPC	sliding-window BP	Binary source
Liu [11], Wang [12]	P-LDPC	RBP	Bianry source
Deng [13]	P-LDPC	MLC and RBP	Gaussian source
Proposed scheme	P-LDPC	Tranformation and RMD	General source

To conquer the aforementioned problems, a new route of "transformation-quantization" is proposed to design an efficient lossy compression based on P-LDPC codes. With the rapid development of deep learning, neural networks are used in a variety of tasks due to their excellent ability for information extraction, and they are used in data compression fields such as image and video compression [14–16]. The authors of [17] optimized autoencoders for lossy image compression, and the paper [18] presents a learned image compression system based on generative adversarial networks. Moreover, the paper [19] proposes an enhanced invertible encoding network with invertible neural networks to mitigate the information-loss problem for better compression. Recently, the diffusion model is also used in image compression fields [20]. In the aforementioned papers, the image data are modeled as the Gaussian source. To extend the source properties, the general source is considered to be compressed by the neural networks in this work. Here, the "transformation-quantization" scheme combines neural networks and lossy P-LDPC (NN-LP-LDPC) codes, as shown in Figure 1, where the transformation using neural networks and quantization using lossy P-LDPC codes modules are unified to be the encoder.

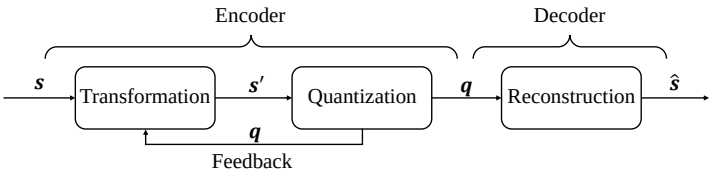

Figure 1. The "transformation-quantization" route based on the NN-LP-LDPC system.

In Figure 1, the encoder includes the transformation and quantization, and there is feedback from quantization module to transformation module. The decoder contains the source reconstruction. Here, the neural networks are employed in the transformation and reconstruction modules, and the quantization is designed with a restrict minimum distortion (RMD) algorithm based on the P-LDPC code. In this encoder, the transformation performs a nonlinear conversion on the continuous sequences, and the quantization converts the continuous sequences into binary sequences. In the decoder, the binary sequences are reconstructed as continuous sequences.

Some key issues are resolved by the NN-LP-LDPC system. First, the conventional BP algorithm cannot be directly used for compressing continuous sources, since it will bring larger distortion. The RMD algorithm is proposed to improve the quantizer. Second, neither the BP nor the RMD has an index; thus, the source is restored without reference. The powerful function-fitting ability of the neural networks serves for the reconstruction to overcome this problem. Third, the quantization function is difficult to be implemented by the neural networks. Since the derivative function of the quantizer is almost zero, the gradient backpropagation is interrupted, and the coefficients of the neural networks cannot be updated. To address the problem, a new derivative function is evaluated to successfully realize the gradient backpropagation from the previous layer. Finally, the adaptation of the compression rate between the quantization and transformation modules is also important. A multi-level feedback mechanism is designed to provide the prior output as the input of the next quantizer. In this way, the sequence length increases with the growing number of quantizers. In addition, the compression rates can be changed by the mask in the RMD algorithm.

The main contributions are summarized as follows.

(1) The NN-LP-LDPC system is proposed for compressing general continuous sources, which complements the vacancy of continuous source compression based on binary LDPC codes. Furthermore, the proposed system is robust to different source distributions.

(2) A new route of "transform-quantization" is designed for the NN-LP-LDPC system, which efficiently replaces the conventional "quantization-compression" scheme. The simulation results validate the usefulness of new route. In addition, they provide a good reference to diversely process different kinds of sources.

(3) The P-LDPC code was efficiently combined with the neural networks, by which the emerging technical problems were successfully resolved. This enormously enriches the designing methodology for the source coding based on the P-LDPC code.

The rest of this paper is organized as follows: Section 2 introduces the proposed scheme. Some key techniques and system optimization are discussed in Section 3. The simulation results and analyses are shown in Section 4. Section 5 concludes this work.

2. NN-LP-LDPC System

As shown in Figure 1, a memoryless continuous sequence $\boldsymbol{s}$ of length n is input into the transformation module, and its output is $\boldsymbol{s}'$ of length mn, where m and n are integers. Then, the continuous sequence $\boldsymbol{s}'$ is quantized as a binary sequence $\boldsymbol{q}$ with the same length mn. It should be noted that the quantized $\boldsymbol{q}$ is also the feedback information to transformation module. Next, the updated $\boldsymbol{q}$ seves as the output of the encoder, and it is used to reconstruct the source $\hat{\boldsymbol{s}}$.

2.1. Transformation Module

The encoder consists of the transformation and quantization modules that are detailed in Figure 2. First, the continuous sequence $\boldsymbol{s} = \{s_1, s_2, \ldots, s_n\}$ is input, and it is transformed as $\boldsymbol{s}'_i = \{s'_{i,1}, s'_{i,2}, \ldots, s'_{i,n}\}$, where $i \in [1, m]$, and $[a, b]$ represents the set of integer numbers from a to b. The sequence $\boldsymbol{s}'_i$ is sent to the ith quantizer Q_i. Then, the continuous sequence $\boldsymbol{s}'_i$ is quantized as a binary sequence $\boldsymbol{q}_i = \{q_{i,1}, q_{i,2}, \ldots, q_{i,n}\}$. For $i \in [1, m-1]$, each $\boldsymbol{q}_i$ is fed back to the MLP of the transformation module, and $\boldsymbol{q}'_i$ is the result. The consolidated $\boldsymbol{s}$ and $\boldsymbol{q}'_i$ are as the input of the CNN, and the output $\boldsymbol{s}'_{i+1}$ is as the input of $i+1$th quantizer Q_{i+1}.

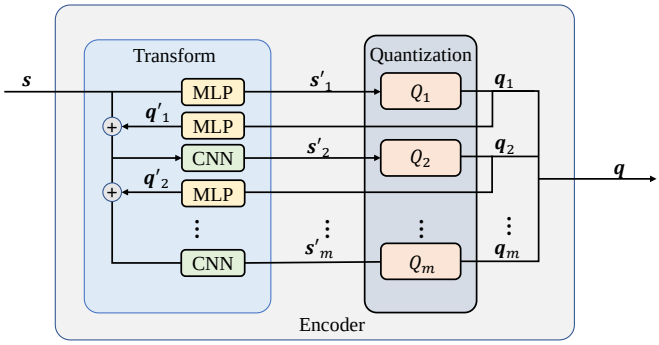

Figure 2. The encoder of proposed scheme: the transformation module contains MLP and CNN networks, and the quantization module consists of multiple quantizers.

The transformation module contains the multi-layer perceptron (MLP) and the convolutional neural network (CNN). An MLP provides a nonlinear transformation to change $\boldsymbol{s}$ into $\boldsymbol{s}'_i$, and the quantization function Q_i obtains the corresponding $\boldsymbol{q}_i$. After that, $\boldsymbol{q}_i$ is returned to another MLP and transformed to $\boldsymbol{q}'_i$, and then it is appended on the $\boldsymbol{s}$, which is presented as:

$$\boldsymbol{s} \oplus \boldsymbol{q}'_i \Rightarrow \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{q}'_1 \\ \vdots \\ \boldsymbol{q}'_i \end{bmatrix}, \tag{1}$$

Then, the appended result increases one dimension of the channel, and it is sent to CNN as the input. The resulting $\boldsymbol{s}'_{i+1}$ is as the input of Q_{i+1}, and $\boldsymbol{q}_{i+1}$ is acquired.

As shown in Figure 3, the MLP is the structure of the fully-connected (FC) layer, including an input layer, an output layer and a hidden layer. The FC layer is expressed as:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}_h + \boldsymbol{b}_h, \tag{2}$$

where $\boldsymbol{X} \in \mathbb{R}^{p \times n}$ is a small batch of inputs; p represents the batch size; the dimensions of the input are n; $\boldsymbol{Y} \in \mathbb{R}^{p \times k}$ is the output of dimension k; $\boldsymbol{W}_h \in \mathbb{R}^{n \times k}$ and $\boldsymbol{b}_h \in \mathbb{R}^{1 \times k}$ are the weight and bias parameters, respectively; and $\mathbb{R}$ indicates the set of real numbers. It should be noted that $\boldsymbol{X}$ and $\boldsymbol{Y}$ refer to the input and output variables in general, respectively.

The activation functions are used to implement nonlinear transformations in the hidden layers. For the MLPs, the active function of the hidden layer is ReLU, which is shown as follows:

$$\mathrm{ReLU}(x) = \max(x, 0) \tag{3}$$

Generally, the CNN contains several convolution layers. It is commonly used in the field of computer vision with a 2D stride and kernels [21,22], and the 1D convolution is confronted with the sequence data [23]. Furthermore, the convolution kernel larger than one is designed to increase the receptive field [24,25]. However, the memoryless continuous source has no spatial locality; hence, a larger field is unnecessary. In addition, it

is convenient that the CNN in the transformation module processes multi-channel data, where the kernel size is one and the pooling layer is not needed.

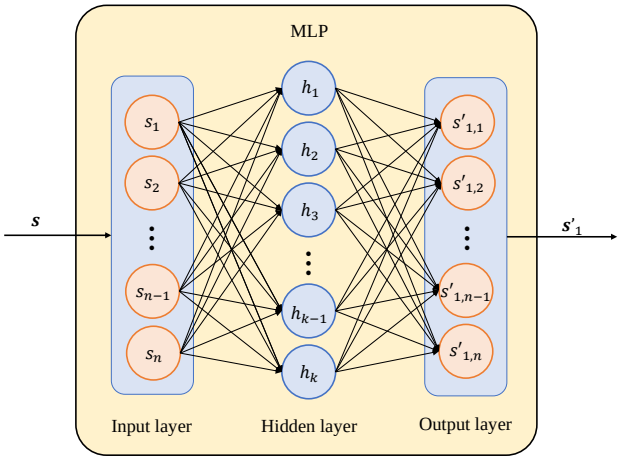

Figure 3. The MLP with the structure of the FC layer.

Considering the aforementioned facts, the CNN only has a 1D convolutional layer with kernel size one, as shown in Figure 4. Similarly to the MLP, the CNN has one hidden layer and uses ReLU as the active function. Actually, for each $y_{l,i} \in \boldsymbol{y}$, the convolution layer with kernel size one is calculated as:

$$y_{i,l} = \sum_{j=1}^{c} k_l x_{i,j} + b_l, \tag{4}$$

where $\boldsymbol{y}$ is the output, $y_{i,l}$ is the ith y in the lth out-channel, $\boldsymbol{x}$ is the input with channel c, $x_{i,j}$ is the ith x in the jth in-channel, $\boldsymbol{k}$ represents the convolution kernel, k_l is the lth channel of $\boldsymbol{k}$ and b_l is the bias. This function can be seen as a FC layer operation in the channel dimension. Thus, in the transformation module, it is more convenient for the CNN to process the multi-channel data with fewer parameters and lower complexity than the FC layer.

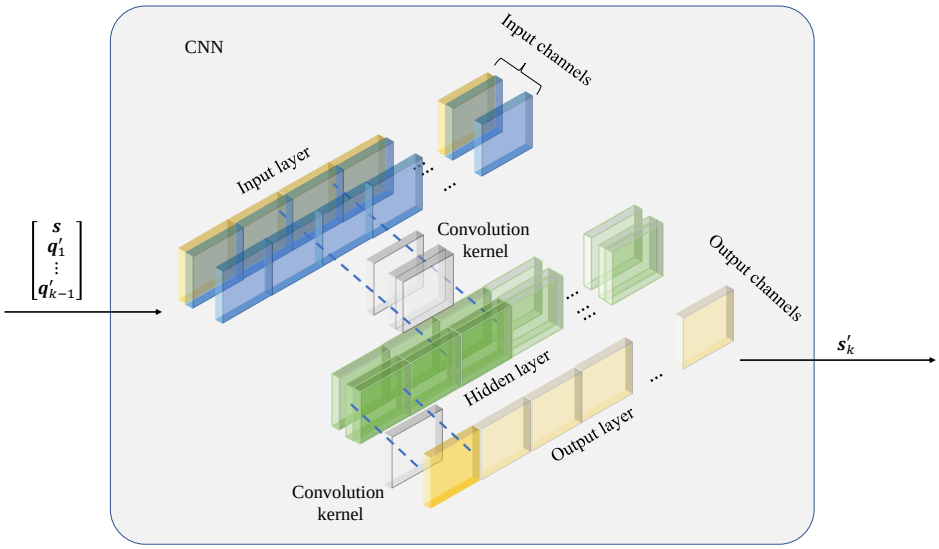

Figure 4. The CNN with 1D convolution layer, and its kernel size is 1.

2.2. Quantization

For a binary source, the BP algorithm is usually employed as the quantization for source compression. The principle of the BP quantization is based on the LDPC codebook satisfying $\boldsymbol{HC}^T = \mathbf{0}$ in GF(2), where $\boldsymbol{C}$ is the correct result, and the codebook $\boldsymbol{H}$ is the parity check matrix of the LDPC code.

However, the continuous source is quite different from the codebook of GF(2). If the continuous sequence is directly compressed by the BP based LDPC code, it will generate a larger distortion. Hence, a new quantizer based on the RMD strategy is designed to replace the BP. The RMD strategy is described in Algorithm 1. In this condition, the compression distortion is minimized to satisfy $\boldsymbol{HC}^T = \mathbf{0}$.

Firstly, the symbols in Algorithm 1 are defined as follows:
$\boldsymbol{s}$: the input source data;
iter: the maximum number of iterations;
λ : the allocation of cost weight between variable and check nodes;
$\boldsymbol{m}$: the mask vector, for which the masked nodes are set to zero;
$\boldsymbol{q}$: the quantized $\boldsymbol{s}$, and also the output of the RMD algorithm;
$(\cdot)_v, (\cdot)_c$: the subscripts represent variable and check parts of symbol $(\cdot)$, respectively;
$\mathrm{g}(\cdot)$: the generation function of the LDPC code;
coe, **map**: the coefficients of the RMD algorithm, and **map** contains $\mathbf{map}^0$ and $\mathbf{map}^1$;
fc: the cost vector of each node, and it contains $\mathbf{fc}^s$ and $\mathbf{fc}^t$;
$\mathscr{V}, \mathscr{C}$: the sets of variable and check nodes, respectively;
$\mathscr{V}(k)$: the variable nodes connected with the k-th check node;
$\mathscr{C}(k)$: the check nodes connected with the k-th variable node;
lr: the learning rate of the training stage;
$[\cdot|\cdot]$: merging of two variables;
$\mathrm{argmin}(\cdot)$: the positioning function of the minimum element.
In addition, the $\mathrm{map}(\cdot)$ function is calculated by

$$\mathrm{map}(\boldsymbol{q}[\mathrm{i}]) = \begin{cases} \mathbf{map}^0[\mathrm{i}], & \boldsymbol{q}[\mathrm{i}] = 0, \\ \mathbf{map}^1[\mathrm{i}], & \boldsymbol{q}[\mathrm{i}] = 1. \end{cases} \tag{5}$$

where $\mathbf{map}^0[\mathrm{i}]$, $\mathbf{map}^1[\mathrm{i}]$ and $\boldsymbol{q}[\mathrm{i}]$ represent the ith element of $\mathbf{map}^0$, $\mathbf{map}^1$ and $\boldsymbol{q}$, respectively.

In Algorithm 1, the variables $\boldsymbol{q}$ and **map** are initialized from lines 1 to 6. The variable **coe** is initialized according to **map** and the input mask $\boldsymbol{m}$ from lines 7 to 12. $\mathbf{fc}^s$ and $\mathbf{fc}^t$ are calculated from lines 13 to 17. In the while loop, $\boldsymbol{q}_c[\mathrm{k}]$ is flipped, and it is determined by the minimum $\mathbf{fc}_c^t[\mathrm{k}]$. If $\mathbf{fc}_c^t[\mathrm{i}] < 0$, the flipping will reduce the distortion; then, $\boldsymbol{q}_v$, $\mathbf{fc}^s$ and $\mathbf{fc}^t$ need to be updated. From lines 29 to 31, **map** is updated by using the gradient descent with learning rate lr, and it is saved for the next use at line 33. When the RDM algorithm is not implemented at the training stage, lines 5 and 6 will be replaced by loading **map**, and lines 29 to 32 will be removed. Algorithm 2 presents the cost function of the RMD algorithm, which calculates the flip cost of each nodes and assigns them to $\mathbf{fc}^s$ according to **coe**. The flow chart of Algorithm 1 is shown in Figure 5.

Algorithm 1 RMD algorithm.

Input: $\boldsymbol{H}, \boldsymbol{s}$, iter, $\lambda, \boldsymbol{m}$
Output: $\boldsymbol{q}$

$\boldsymbol{s}_v, \boldsymbol{s}_c \leftarrow \boldsymbol{s}$
$\boldsymbol{q}_c \leftarrow \text{sign}(\boldsymbol{s}_c)$
$\boldsymbol{q}_v \leftarrow \text{g}(\boldsymbol{q}_c)$
$\boldsymbol{q} \leftarrow [\boldsymbol{q}_v|\boldsymbol{q}_c]$
map $\leftarrow \mathbf{0}$
$\textbf{map}^1 \leftarrow \textbf{map}^1 + 1$
$\textbf{coe} \leftarrow 1/(\textbf{map}^0 - \textbf{map}^1)^2$
for i in $\boldsymbol{m}$ **do**
  if $\boldsymbol{m}$[i] $= 0$ **then**
    coe[i] $\leftarrow 0$
  end if
end for
$\textbf{fc}^s \leftarrow \text{cost}(\boldsymbol{q}, \boldsymbol{s}, \boldsymbol{coe})$
$\textbf{fc}_v^s$, $\textbf{fc}_c^s \leftarrow \textbf{fc}^s$
for i in $\mathscr{C}$ **do**
  $\textbf{fc}_c^t$[i] $\leftarrow \lambda \textbf{fc}_c^s$[i] $+ (1-\lambda) \sum_{j \in \mathscr{V}(i)} \textbf{fc}_v^s$[j]
end for
$n \leftarrow 0$
while $n <$ iter **do**
  $n \leftarrow n + 1$
  $k \leftarrow \text{argmin}(\textbf{fc}_c^t)$
  if $\textbf{fc}_c^t$[k] < 0 **then**
    $\boldsymbol{q}_c$[k] $\leftarrow 1 - \boldsymbol{q}_c$[k]
    $\boldsymbol{q}_v$, $\textbf{fc}^s$, $\textbf{fc}^t \leftarrow \text{update}(\boldsymbol{s}_v, \boldsymbol{q}_v, k, \textbf{fc}^s, \textbf{fc}^t)$
  else
    break
  end if
end while
$\hat{\boldsymbol{s}} \leftarrow \text{map}(\boldsymbol{q})$
gradient $\leftarrow 2(\boldsymbol{s} - \hat{\boldsymbol{s}})$
map $\leftarrow$ **map** $+$ **gradient** $\cdot$ lr
save **map**
return $\boldsymbol{q}$

Algorithm 2 $\textbf{fc}^s = \text{cost}(\boldsymbol{q}, \boldsymbol{s}, \textbf{coe})$.

Input: $\boldsymbol{q}, \boldsymbol{s}$, **coe**
Output: $\textbf{fc}^s$

$\textbf{fc}^s \leftarrow \mathbf{0}$
for i in $\boldsymbol{q}$ **do**
  $\textbf{fc}^s$[i] $\leftarrow$ **coe**[i] $\cdot ((\boldsymbol{s} - \text{map}(1 - \boldsymbol{q}[\text{i}]))^2 - (\boldsymbol{s} - \text{map}(\boldsymbol{q}[\text{i}]))^2)$
end for
return $\textbf{fc}^s$

In the RMD algorithm, q is flipped with the minimum fc^t in each iteration satisfying $\boldsymbol{qH} = \mathbf{0}$. Here, the minimum fc^t indicates the maximum quantization error between itself and the associated variable node; therefore, the flipping will effectively reduce the total quantization distortion. In the training process, **coe** and **map** are updated by gradient descent. With **coe** and **map** updating, the $\textbf{fc}^t$ will be calculated more accurately.

Algorithm 3 presents a quick way to update $\boldsymbol{q}_v, \textbf{fc}^s$ and $\textbf{fc}^t$. Only if $\boldsymbol{q}_v$[i] satisfying $\text{i} \in \mathscr{V}(k)$ is flipped, the corresponding $\textbf{fc}_v^s$[i] can be updated. Then, the corresponding $\textbf{fc}_c^t$[j] is refreshed by calculating $\textbf{fc}_c^t[\text{j}] = \textbf{fc}_c^t[\text{j}] + (1-\lambda)(\textbf{fc}_v^s[\text{i}] - t)$. In this case, it does not need to recalculate $\boldsymbol{q}_v, \textbf{fc}^s$ and $\textbf{fc}^t$. The flow chart of Algorithm 3 is shown in Figure 6.

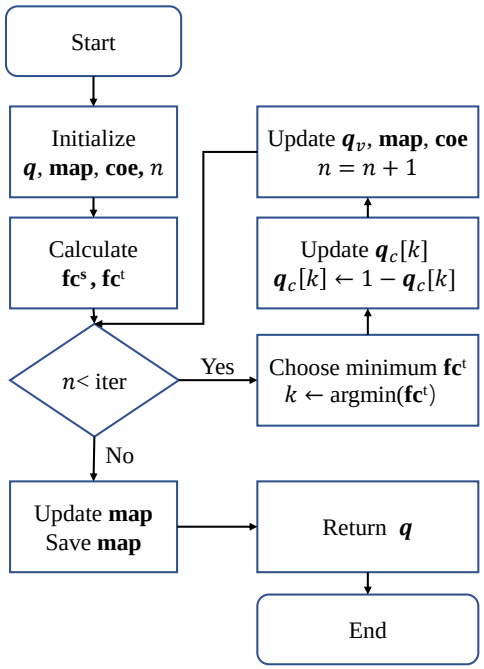

Figure 5. The flow chart of Algorithm 1.

Algorithm 3 $\boldsymbol{q}_v, \mathbf{fc}^s, \mathbf{fc}^t = \text{update}(\boldsymbol{s}_v, \boldsymbol{q}_v, k, \mathbf{fc}^s, \mathbf{fc}^t)$.

Input: $\boldsymbol{s}_v, \boldsymbol{q}_v, k, \mathbf{fc}^s, \mathbf{fc}^t$
Output: $\boldsymbol{q}_v, \mathbf{fc}^s, \mathbf{fc}^t$
for i in $\mathscr{V}(k)$ **do**
  $\boldsymbol{q}_v[\text{i}] = 1 - \boldsymbol{q}_v[\text{i}]$
  $t \leftarrow \mathbf{fc}_v^s[\text{i}]$
  $\mathbf{fc}_v^s[\text{i}] = \text{cost}(\boldsymbol{q}_v[\text{i}], \boldsymbol{s}_v[\text{i}])$
  for j in $\mathscr{C}(i)$ **do**
    $\mathbf{fc}_c^t[\text{j}] = \mathbf{fc}_c^t[\text{j}] + (1 - \lambda)(\mathbf{fc}_v^s[\text{i}] - t)$
  end for
end for
return $\boldsymbol{q}_v, \mathbf{fc}^s, \mathbf{fc}^t$

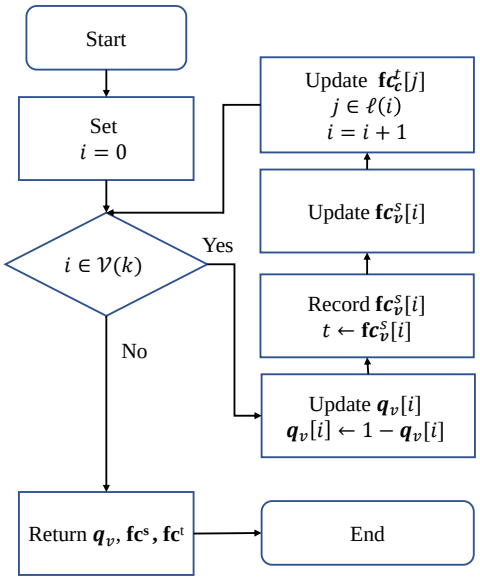

Figure 6. The flow chart of Algorithm 3.

Each node i in the check matrix of the LDPC code with mask vector $\boldsymbol{m}[\mathrm{i}] = 1$ is filled with $\boldsymbol{s}'_m[\mathrm{i}]$ before the RMD training. The output $\boldsymbol{q}$ is compressed as $\boldsymbol{q}_c$ following $\boldsymbol{qH} = \boldsymbol{0}$, and it can be reconstructed by $\boldsymbol{q} = g(\boldsymbol{q}_c)$, where $g(\cdot)$ is the generation function of LDPC code. This allows the rate $r = \frac{n-k}{n-m'}$ to be changed from $(n-k)/n$ to 1 according to the variable m', where n and k are the code length and numbers of variable nodes, respectively, and m' represents the number of element 1 in mask vector $\boldsymbol{m}$.

The computational complexity of the proposed RMD algorithm is

$$\begin{aligned} O_{RMD} &= O_{initial} + O_{iterate} \\ &= O(n \times dv \times dc) + O(t \times dv \times dc) \\ &= O(n \times dv \times dc), \end{aligned} \tag{6}$$

where n is the number of check nodes; t is the number of iterations satisfying $t < n$; and dv and dc are the degrees of the variable and check nodes, respectively. In addition, the number of iterations is limited to 30 in the RMD algorithm, and the BP algorithm needs over 100 iterations. Overall, the computational and time complexities of the RMD algorithm are both lower than those of the BP algorithm.

2.3. Decoder

The decoder structure is shown in Figure 7. Referring to the encoding scheme, $\boldsymbol{q}$ is divided into $\boldsymbol{q}_1 \sim \boldsymbol{q}_m$, and they are input into the MLP. After that, $\boldsymbol{q}'_1 \sim \boldsymbol{q}'_m$ are unified as a matrix:

$$\boldsymbol{q}'_1 \oplus \boldsymbol{q}'_2 \oplus \cdots \oplus \boldsymbol{q}'_m \Rightarrow \begin{bmatrix} \boldsymbol{q}'_1 \\ \boldsymbol{q}'_2 \\ \vdots \\ \boldsymbol{q}'_m \end{bmatrix}. \tag{7}$$

Then, the joint result is sent to the CNN and reconstructs $\hat{\boldsymbol{s}}$. The corresponding parameters and structure of the CNN can be referred to from Figure 4.

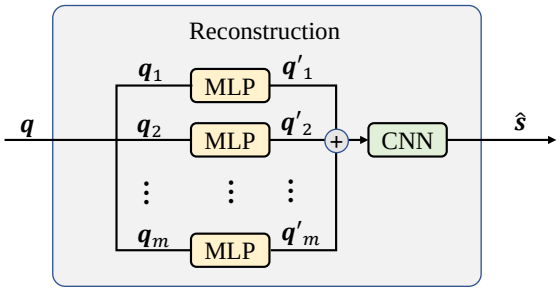

Figure 7. The decoder of the proposed scheme.

3. System Optimization and Technical Details

3.1. Gradient Backpropagation

In this section, the non-differentiability problems in the RMD algorithm and the neural network are resolved. Since the RMD algorithm is used as one layer of the neural networks, the backpropagation of this layer needs to provide the effective gradients. In this case, the coefficients of the neural networks are updated before the quantization according to the gradients, so that the loss function can be minimized.

However, in the quantization procedure, the values of the derivative function are mostly zeros. In this case, the gradient backpropagation of the neural network will be terminated [26]. To solve this problem, an existing work considers adding random noise to the quantization for training [14]. However, there is a larger discrepancy between the testing and training procedures, which significantly affects the quantization results.

According to [27], a gradient expectation is theoretically computed with the finite difference; i.e.,

$$\begin{aligned}\frac{d}{ds}\mathbb{E}[Q(s+u)] &= \frac{d}{ds}\int_{-t/2}^{t/2} Q(s+v)dv \\ &= Q(s+t/2) - Q(s-t/2),\end{aligned} \tag{8}$$

where $Q(\cdot)$ is the quantization function, $\mathbb{E}(\cdot)$ is the expectation calculation, t is the length of granular cells of the quantizer and the distribution follows $u \sim \mathcal{U}(-t/2, t/2)$. Equation (8) allows one to evaluate the derivative even if Q is non-differentiable. By extending the Q function to a vector $\boldsymbol{s}+\boldsymbol{u}$, where $\boldsymbol{u} \sim \mathcal{U}(-t/2, t/2)^D$, and the superscript D represents the dimension of the input vector, it has

$$\begin{aligned}&\frac{\partial}{\partial s_i}\mathbb{E}[(Q(\boldsymbol{s}+\boldsymbol{u}))] \\ &= \mathbb{E}\left[\frac{\partial}{\partial z_i}(\boldsymbol{Z})|_{\boldsymbol{Z}=Q(\boldsymbol{s}+\boldsymbol{u})} \cdot \frac{\partial}{\partial s_i}Q(s_i+u_i)\right], \\ &\approx \mathbb{E}\left[\frac{\partial}{\partial z_i}(\boldsymbol{Z})|_{\boldsymbol{Z}=Q(\boldsymbol{s}+\boldsymbol{u})}\right] \cdot \mathbb{E}\left[\frac{\partial}{\partial s_i}Q(s_i+u_i)\right].\end{aligned} \tag{9}$$

Here, $\boldsymbol{Z}$ is an independent variable at the next layer. From Equation (8), the derivative of the backpropagation is replaced by the following expectation:

$$\mathbb{E}\left[\frac{\partial}{\partial s_i}Q(s_i+u_i)\right] = Q(s_i+t/2) - Q(s_i-t/2). \tag{10}$$

By replacing the original derivative function with the expectation value, the gradients from the next layer can correctly calculate the quantization output. In this way, the compression rate is converted in a larger interval before the quantization.

3.2. Training the Network

The proposed scheme can be seen as an end-to-end network, where the labels should be the input continuous sequences themselves, the mean square error (MSE) loss is selected as the loss function and Adam is the optimizer. In the network, the learning rate is set to 0.005, and the batch size is 1024; see Table 3 for details. However, if the MSE loss is only used on the output of the total system, the network is hard to converge. It is recommended to add the loss function to each $\boldsymbol{q}_i'$ after the MLP in the transformation module, which can speed up the network's convergence.

Table 3. Training parameters.

Loss	Learning Rate	Optimizer	Batch Size	Number of Epochs
MSE	5×10^{-3}	Adam	1024	1000

4. Simulation Results and Discussions

In this section, we take Gaussian sources following the distribution $\mathcal{N}(0,1)$ as an example. By using the MSE measurement, the distortion-rate limit [28] is expressed as

$$d = 2^{-2r}, \tag{11}$$

where d is the theoretical distortion, and r represents the compression rate in bits/symbol.

The check matrix of the P-LDPC code is extended by using the progressive edge-growth algorithm [29], and the compression rate r is calculated by

$$r = m - 1 + \frac{n-k}{n-m'}, \tag{12}$$

where $m-1$ indicates that there are $m-1$ quantizers, $Q_1 \sim Q_{m-1}$, which are set as the sign functions of rates 1, and the rate of Q_m is $\frac{n-k}{n-m'}$.

In Figure 8, the distortion-rate performances are analyzed based on different benchmark P-LDPC codes in [30], including AR3A, AR4JA and ARA codes. The extending times were 5 and 20. It can be seen that he code after extending five times had a better distortion-rate performance than the 20-times-extended code. That is, the fewer dimensions the check matrix uses, the more the system's distortion is reduced. Note that the dimensions of the check matrix significantly increase the time consumption of the proposed scheme. Hence, the proposed system has lower complexity by employing less P-LDPC code.

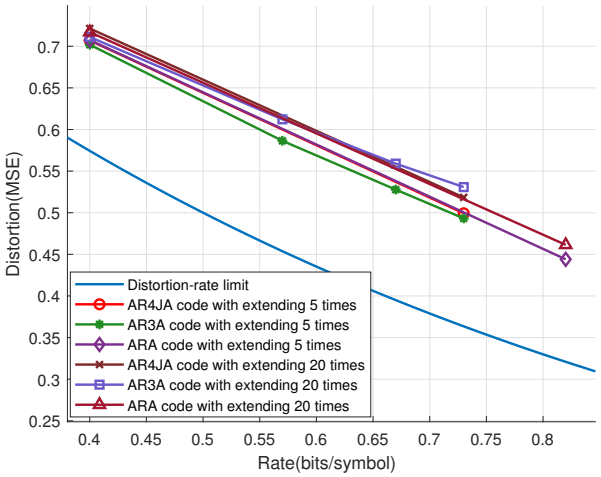

Figure 8. The distortion-rate analyses based on different P-LDPC codes and extending times.

In Figure 9, the distortion-rate performance is compared for the BP and the RMD algorithms. Three benchmark P-LDPC codes [30] were used for simulations. It is clear that the AR3A code achieved better results, approaching the distortion-rate limit. Furthermore, instead of the BP algorithm, the RMD algorithm using AR3A code was closer to the distortion-rate limit. Hence, the RMD algorithm is more efficient than the BP algorithm.

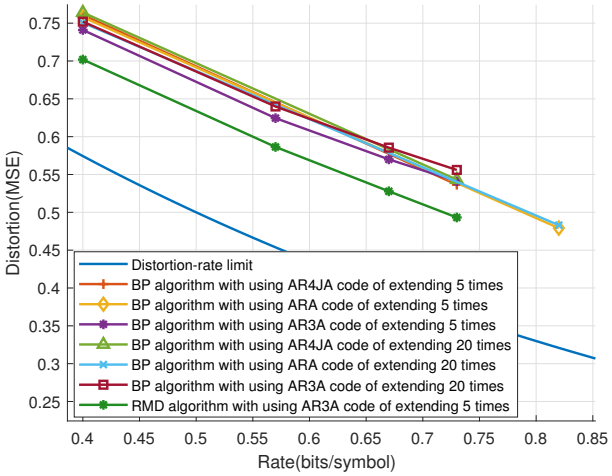

Figure 9. The distortion-rate comparison between the BP and RMD algorithms.

In Figure 10, the distortion-rate performance is shown at the high-rate regime. When the original derivative function is replaced by the new one, the neural network obtains correct gradients to update the coefficients. In this case, it is obvious that the simulation with the new derivative function is closer to the distortion-rate limit. In the rate interval from 0 to 1, these two curves approach one another, since the feedback mechanism does not

need to work. Overall, the replaced derivative function and feedback mechanism ensure the system to work well when the rate goes higher.

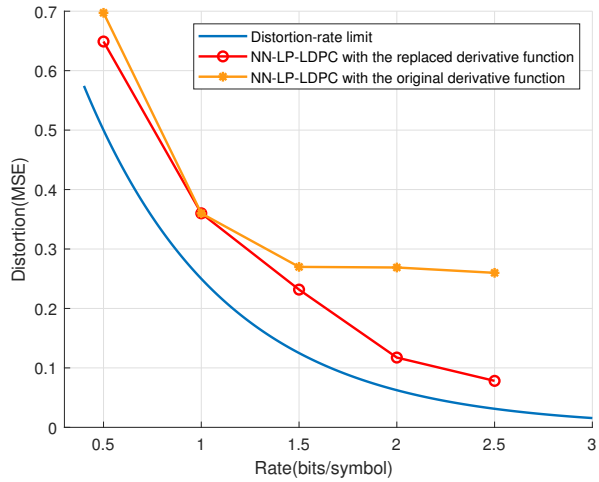

Figure 10. The distortion-rate analyses of the proposed scheme based on the distinct derivative functions.

In Figure 11, the proposed scheme, is further compared with the MLC system [13]. By using the AR3A code, it is clear that the proposed system brings a performance improvement over the MLC scheme. Even though an optimally-designed code in [13] is implemented by the MLC system, its performance is still worse than the NN-LP-LDPC. Therefore, the NN-LP-LDPC code is not only an efficient system for the lossy compression, but it also has simpler structure than the MLC system.

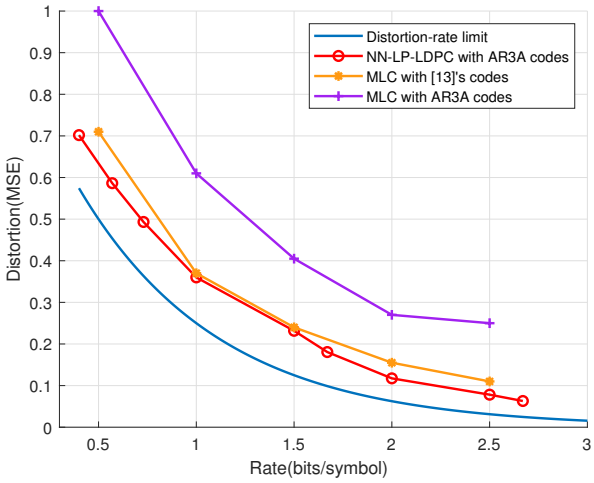

Figure 11. The distortion-rate comparison between the NN-LP-LDPC and MLC [13] systems.

In addition, since the proposed scheme was designed based on neural networks, it is demonstrated that the system input is applicable to a general source—for example, the continuous sequences following Gaussian, Laplacian and other distributions. The related simulations are shown in Figure 12. The Laplacian source follows $f(x) = \frac{\lambda}{2} e^{-\lambda|x|}$, and the distortion-rate limit of the Laplacian source with the MSE distortion is [31]:

$$D_\delta = \frac{2}{\lambda^2} - \frac{\Delta}{\lambda}\left(1 + \coth\frac{\lambda\Delta}{2}\right)e^{-\frac{\lambda\Delta}{2}}, \tag{13}$$

$$R_\delta = -p'(0)\log_2 p'(0) - e^{-\frac{\lambda\Delta}{2}}\log_2 \sinh\frac{\lambda\Delta}{2} + \frac{\lambda}{\log 2}S, \tag{14}$$

where the distortion-rate limit is expressed as a parametric equation, the parametric is Δ, $\Delta \in (0, +\infty)$, $p'(0) = 1 - e^{-\frac{\lambda\Delta}{2}}$, and S is calculated by

$$S = 2\Delta \cdot \sinh \frac{\lambda\Delta}{2} \sum_{i=0}^{n} i e^{-i\lambda\Delta}. \tag{15}$$

In our system simulation, the variance of the Laplacian source is given as $\frac{2}{\lambda^2} = 1$, and λ is set to $\sqrt{2}$.

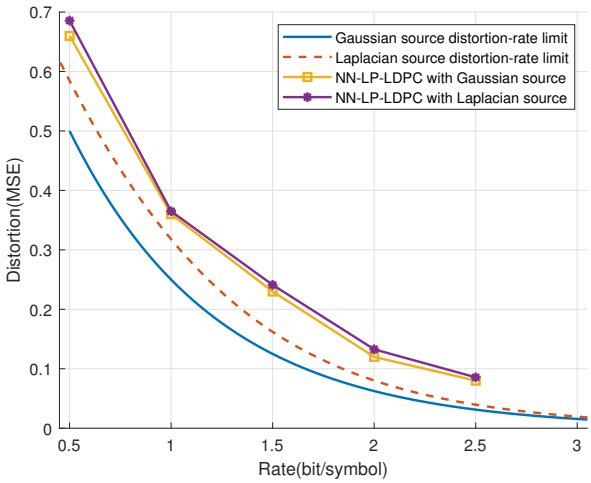

Figure 12. The distortion-rate analyses of the general source, including the Gaussian and the Laplacian sources.

It is clear that both the Gaussian and the Laplace sources have good performances to approach the distortion-rate limit. Furthermore, the simulating performances of the two types of sources were similar, which indicates that the proposed scheme has good robustness for different source distributions.

5. Conclusions

In this paper, it is demonstrated that the new route of "transform-quantization" significantly outperforms the conventional "quantization-compression" by using the neural networks. This provides a different method with which to design the lossy compression system. In addition, the P-LDPC codes were inserted into the neural networks as the NN-LP-LDPC system, which is obviously different from the existing work. The effectiveness of the proposed scheme was verified by simulation results. Compared with the existing works, the proposed scheme achieved both better performance and lower complexity. Furthermore, it has versatility and is suitable for compressing different sources. However, due to its simple structure, one drawback is that the current scheme may not work well for image/video compression, which is left as future work. In addition, the P-LDPC codes used in this paper are not optimized for lossy compression. Our future work will focus on the system optimizations, including the design of P-LDPC codebooks, the improvement of the RMD algorithm and the design of practical neural networks.

Author Contributions: Conceptualization, D.S. and L.W.; Methodology, J.R.; Software, J.R.; Validation, D.S. and H.W.; Formal analysis, J.R.; Investigation, D.S.; Resources, L.W.; Data curation, J.R.; Writing—original draft, J.R.; Writing—review & editing, H.W.; Visualization, D.S.; Supervision, H.W. and L.W.; Funding acquisition, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 61671395.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gallager, R. Low-density parity-check codes. *IRE Trans. Inf. Theory* **1962**, *8*, 21–28. [CrossRef]
2. MacKay, D.J. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inf. Theory* **1999**, *45*, 399–431. [CrossRef]
3. Thorpe, J. Low-density parity-check (LDPC) codes constructed from protographs. *IPN Prog. Rep.* **2003**, *42*, 42–154.
4. Liva, G.; Chiani, M. Protograph LDPC Codes Design Based on EXIT Analysis. In Proceedings of the IEEE GLOBECOM 2007—IEEE Global Telecommunications Conference, Washington, DC, USA, 26–30 November 2007; pp. 3250–3254.
5. Gupta, A.; Verdú, S. Operational duality between lossy compression and channel coding. *IEEE Trans. Inf. Theory* **2011**, *57*, 3171–3179. [CrossRef]
6. Wainwright, M.J.; Maneva, E.; Martinian, E. Lossy Source Compression Using Low-Density Generator Matrix Codes: Analysis and Algorithms. *IEEE Trans. Inf. Theory* **2010**, *56*, 1351–1368. [CrossRef]
7. Liveris, A.; Xiong, Z.; Georghiades, C. Compression of binary sources with side information at the decoder using LDPC codes. *IEEE Commun. Lett.* **2002**, *6*, 440–442. [CrossRef]
8. Matsunaga, Y.; Yamamoto, H. A coding theorem for lossy data compression by LDPC codes. *IEEE Trans. Inf. Theory* **2003**, *49*, 2225–2229. [CrossRef]
9. Braunstein, A.; Kayhan, F.; Zecchina, R. Efficient LDPC codes over GF (q) for lossy data compression. In Proceedings of the 2009 IEEE International Symposium on Information Theory, Seoul, Republic of Korea, 28 June–3 July 2009; pp. 1978–1982.
10. Fang, Y. LDPC-Based Lossless Compression of Nonstationary Binary Sources Using Sliding-Window Belief Propagation. *IEEE Trans. Commun.* **2012**, *60*, 3161–3166. [CrossRef]
11. Liu, Y.; Wang, L.; Wu, H.; Liu, S. Performance of lossy P-LDPC codes over GF (2). In Proceedings of the 2020 IEEE 14th International Conference on Signal Processing and Communication Systems (ICSPCS), Adelaide, SA, Australia, 14–16 December 2020; pp. 1–5.
12. Wang, R.; Liu, S.; Wu, H.; Wang, L. The Efficient Design of Lossy P-LDPC Codes over AWGN Channels. *Electronics* **2022**, *11*, 3337. [CrossRef]
13. Deng, H.; Song, D.; Miao, M.; Wang, L. Design of Lossy Compression of the Gaussian Source with Protograph LDPC Codes. In Proceedings of the 2021 IEEE 15th International Conference on Signal Processing and Communication Systems (ICSPCS), Sydney, Australia, 13–15 December 2021; pp. 1–6.
14. Ballé, J.; Laparra, V.; Simoncelli, E.P. End-to-end optimization of nonlinear transform codes for perceptual quality. In Proceedings of the 2016 IEEE Picture Coding Symposium (PCS), Nuremberg, Germany, 4–7 December 2016; pp. 1–5.
15. Toderici, G.; Vincent, D.; Johnston, N.; Jin Hwang, S.; Minnen, D.; Shor, J.; Covell, M. Full resolution image compression with recurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5306–5314.
16. Zhang, Z.T.; Yeh, C.H.; Kang, L.W.; Lin, M.H. Efficient CTU-based intra frame coding for HEVC based on deep learning. In Proceedings of the 2017 IEEE Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Kuala Lumpur, Malaysia, 12–15 December 2017; pp. 661–664.
17. Theis, L.; Shi, W.; Cunningham, A.; Huszár, F. Lossy image compression with compressive autoencoders. *arXiv* **2017**, arXiv:1703.00395.
18. Choi, Y.; El-Khamy, M.; Lee, J. Variable Rate Deep Image Compression With a Conditional Autoencoder. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
19. Xie, Y.; Cheng, K.L.; Chen, Q. Enhanced invertible encoding for learned image compression. In Proceedings of the 29th ACM International Conference on Multimedia, Virtual Event China, 20–24 October 2021; pp. 162–170.
20. Yang, R.; Mandt, S. Lossy image compression with conditional diffusion models. *arXiv* **2022**, arXiv:2209.06950.
21. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
22. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
23. Dewantara, D.S.; Budi, I.; Ibrohim, M.O. 3218IR at SemEval-2020 Task 11: Conv1D and word embedding in propaganda span identification at news articles. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, Barcelona, Spain (Online), 12 December 2020; pp. 1716–1721.
24. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
25. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
26. Werbos, P. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Harvard University, Cambridge, MA, USA, 1974.
27. Agustsson, E.; Theis, L. Universally quantized neural compression. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 12367–12376.
28. Thomas, M.; Joy, A.T. *Elements of Information Theory*; Wiley-Interscience: Hoboken, NJ, USA, 2006; pp. 463–508.

29. Hu, X.Y.; Eleftheriou, E.; Arnold, D.M. Regular and irregular progressive edge-growth tanner graphs. *IEEE Trans. Inf. Theory* **2005**, *51*, 386–398. [CrossRef]
30. Divsalar, D.; Dolinar, S.; Jones, C.R.; Andrews, K. Capacity-approaching protograph codes. *IEEE J. Sel. Areas Commun.* **2009**, *27*, 876–888. [CrossRef]
31. Rajpoot, N.M. Simulation of the Rate-Distortion Behaviour of a Memoryless Laplacian Source. In Proceedings of the 4th Middle Eastern Symposium on Simulation and Modelling (MESM 2002), Sharjah, United Arab Emirates, 28–30 October 2002.